UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

DEPARTAMENTO DE ELECTRÓNICA E COMPUTACIÓN

PhD Thesis

Split and Shift Methodology: Overcoming Hardware Limitations on Cellular Processor Arrays for Image Processing

Author:
Natalia A. Fernández García

PhD supervisors:
Diego Cabello Ferrer
Víctor M. Brea Sánchez

Santiago de Compostela, Xullo de 2012

Dr. **Diego Cabello Ferrer**,
Catedrático de Universidade da Área de Electrónica da Universidade de Santiago de Compostela

Dr. **Víctor M. Brea Sánchez**,
Profesor Contratado Doutor da Área de Electrónica da Universidade de Santiago de Compostela

FAN CONSTAR:

Que a memoria titulada **Split and Shift Methodology: Overcoming Hardware Limitations on Cellular Processor Arrays for Image Processing** foi realizada por Dna. **Natalia A. Fernández García** baixo a nosa dirección no Departamento de Electrónica e Computación e no Centro de Investigación en Tecnoloxías da Información (CITIUS) da Universidade de Santiago de Compostela, e constitúe a Tese que presenta para optar ao grao de Doutora pola Universidade de Santiago de Compostela.

Santiago de Compostela, Xullo de 2012

Asdo: **Diego Cabello Ferrer**
Codirector da Tese

Asdo: **Víctor M. Brea Sánchez**
Codirector da Tese

Asdo: **Natalia A. Fernández García**
Autora da Tese

A Breo e Martina,
os meus amores,
o meu sentido.

Agradecementos

I would like to thank my supervisors Dr. Víctor Brea Sánchez and Dr. Diego Cabello Ferrer for their constant support and the many rewarding discussions during the development of this work. I specially thank Dr. Brea for his spirit of self-improvement, and his commitment and perseverance, which have been fundamental in the completion of this thesis. I admire his professionalism, his great capacity for work, and his constant search for new professional challenges. My also special gratitude to Dr. Cabello whose wide mature experience has provided us with an invaluable constant guidance in the development of this work. I want also to thank Dr. David López Vilariño at the Universidade de Santiago de Compostela for the proposal of the initial challenge of this thesis, the implementation of large neighborhood templates over locally connected binary architectures, and for providing us with the enough information about the PLS algorithm. Without his contribution this work would have been completely other.

I am also very grateful to Dr. Jordi Albó i Canals at EALS - Electrònica (Universitat Ramon Llull) for his effort during the intensive and enriching working days in Barcelona. Gràcies tambè a tots amb els quals vaig coincidir en el grup d'recerca de La Salle, especialment Jordi A., Jordi R., Xavi, Giovanni, Olga, Sergio i Enric, pel vostre acolliment i el vostre afecte; guardo extraordinaris records del poc temps passat amb vosaltres. My deep gratitude to Dr. Piotr Dudek, Dr. David Barr, Dr. Jayawan Wijekoon, and Dr. Alexey Lopich at The University of Manchester for the very enriching discussions, and, very specially, for their personal and professional support. I would also like to thank Manuel Suárez Cambre, PhD student at the Universidade de Santiago de Compostela, his collaboration in the statistical analysis of the CNN templates shape. And finally, a very special space here for all my lab-mates at labs 26 and 30 in the Departamento de Electónica e Computación, Raquel, DG, Pili, Nando, Levo, Bea, ..., we have shared interesting and enriching discussions over research and not research issues, but above all, thanks a lot for your friendship in the hard moments.

The work in this PhD thesis was partially supported by the Ministerio de Ciencia y Tecnología (Spain) under the projects TIC2003-09521, TEC2009-12686, and the "Programa Nacional de Formación de Profesorado Universitario FPU" (REF.: AP-2004-3741), and by the Xunta de Galicia (Spain) under the projects PGIDIT04PXI20606PN, PGIDIT06TIC10502PR, and 10PXIB206037PR.

Máis alá do meramente profesional, esta tese non tería sido posible sen o constante apoio, cariño e confianza dos meus pais, Julio Abel e Luisa, para vós o meu máis fondo e sincero agradecemento e cariño. Grazas tamén a todos aqueles que pasastes e pasades deixando esa boa pegada na miña vida, esta tese tamén ten un pouquiño de cada un de vós. E grazas, por suposto, a Breo e Martina, vós déstesLle sentido a este e a todos os outros proxectos e infundístesme o ánimo e a forza para levalos a cabo.

Xullo de 2012

(...)

lento pero viene
el futuro real
el mismo que inventamos
nosotros y el azar

cada vez más nosotros
y menos el azar

(...)

Mario Benedetti

Resumo

Seguindo o regulamento dos estudos de terceiro ciclo da Universidade de Santiago de Compostela, aprobado na Xunta de Goberno o día 7 de abril de 2000 (DOG de 6 de marzo de 2001) e modificado pola Xunta de Goberno de 14 de novembro de 2000, o Consello de Goberno de 22 de novembro de 2003, de 18 de xullo de 2005 (artigo 30 a 45), de 11 de novembro de 2008 e de 14 de maio de 2009; e, concretamente, cumprindo as especificacións indicadas no capítulo 4, artigo 30, apartado 3 do devandito regulamento, móstrase a continuación un resumo en galego da tese.

Na era multimedia, o procesado de imaxe converteuse nun elemento de singular importancia nos dispositivos electrónicos. Dende as comunicacións (p.e. telemedicina), a seguranza (p.e. recoñecemento retiniano) ou control de calidade e de procesos industriais (p.e. orientación de brazos articulados, detección de defectos do produto), pasando pola investigación (p.e. seguimento de partículas elementais) e diagnose médica (p.e. detección de células estrañas, identificación de veas retinianas), hai un sinfín de aplicacións onde o tratamento e interpretación automáticas de imaxe é fundamental. O obxectivo último sería o deseño de sistemas de visión con capacidade de decisión. As tendencias actuais requiren, ademais, a combinación destas capacidades en dispositivos pequenos e portátiles con resposta en tempo real. Isto propón novos desafíos tanto no deseño hardware como software para o procesado de imaxe, buscando novas estruturas ou arquitecturas coa menor área e consumo de enerxía posibles sen comprometer a funcionalidade e o rendemento.

Pola súa banda, o procesado de imaxe é unha tarefa complexa que pode dividirse en tres niveis diferenciados de subtarefas que están conectadas xerarquicamente. As tarefas de procesado de imaxe de baixo nivel, ou 'visión temperá', non requiren de coñecemento adicional e actúan na imaxe localmente, independentemente do contido, preparando os datos para o seguinte nivel. As tarefas incluídas neste nivel son normalmente de tipo convolución, repetitivas, sinxelas e de baixa precisión, e están normalmente orientadas á restauración ou mellora das características da imaxe para o seu posterior procesamento. Con todo, son moi esixentes computacionalmente debido á gran cantidade de datos a procesar. As tarefas de procesado de imaxe de nivel intermedio extraen información simbólica sobre a imaxe dos datos proporcionados polo nivel anterior mediante métodos globais principalmente. A cantidade de información requirida aquí é baixa, as tarefas son máis complexas e operan sobre os datos preprocesados, é dicir, sobre unha pequena parte dos datos contidos na imaxe orixinalmente. Finalmente, o procesado de alto nivel supón tarefas complexas dirixidas dalgunha maneira á comprensión do contido da imaxe. Estas utilizan a descrición simbólica proporcionada polo nivel intermedio e requiren unha cantidade significativa de información adicional para interpretar a imaxe.

Un sistema de visión inclúe estes tres niveis co obxectivo de tomar decisións autónomas. É o que se denomina Visión por Computador. Aínda que estas operacións se poden realizar nunha computadora von Neumann clásica, a alta carga computacional e o paralelismo inherente (especialmente, ambos, na fase de baixo nivel) fai dela unha opción non axeitada para o procesado de imaxe. Débese ter en conta que, aínda que as operacións realizadas a alto nivel son moito máis complexas, son as operacións de nivel máis baixo as que en moitas ocasións definen o 'colo de botella' no algoritmo, pois representan máis do 50% da carga computacional. Os procesadores modernos inclúen unidades paralelas e unidades replicadas e explotan o paralelismo a nivel de instrución. Con todo, a súa orientación a punto flotante de propósito xeral fainos pouco eficientes para o procesado de baixo nivel de imaxe máis aló do desperdicio de recursos non necesarios. Os Procesadores Dixitais de Sinais (DSP) están optimizados para o procesado de sinal e poden ser adecuados para aplicacións de baixa esixencia.

As matrices bidimensionais de procesadores celulares (CPAs) cunha memoria de programa almacenado principal dentro dunha unidade de control global que transmite as instrucións a efectuar por un conxunto de elementos de procesamento (PEs), son, pola súa banda, unha implementación natural das operacións de procesado de imaxes de baixo nivel. Ademais, os sensores CMOS permitiron integrar o sistema de adquisición da imaxe e o procesador no mesmo chip, eliminando o 'colo de botella' entre ambos e dando orixe aos procesadores de plano focal ou chips de visión. A máquina universal baseada en redes non lineais celulares (RNCs ou CNNs) é unha proposta específica de CPA de propósito xeral para procesamento de imaxe de baixo nivel integrable neste tipo de chips. As conexións locais, a robustez da saída non-lineal e a simplicidade do control SIMD fan das CNNs opcións axeitadas para a implementación hardware. Á súa vez, os operadores de convolución xunto coa dinámica de evolución con capacidade de procesamento global e o paralelismo masivo fan as CNNs axeitadas para o procesado de imaxe de baixo nivel.

As contribucións desta tese céntranse neste tipo de implementacións masivamente paralelas para procesamento de imaxe de baixo nivel. No desenvolvemento do traballo contemplamos unha vertente hardware e unha vertente funcional. No primeiro aspecto traballamos na redución da área ocupada ao través da limitación na conectividade entre PEs, entendendo nesta tanto os circuítos que ponderan as contribucións dos veciños como as propias conexións entre PEs. Na segunda cuestión abordamos a mellora da funcionalidade de sistemas de visión baseados en matrices de procesadores celulares (CPAs) permitindo a aplicación de operacións que involucren comunicacións de longa distancia ou, o que é o mesmo, operacións de grande veciñanza, mantendo a conectividade hardware local e incluso reducíndoa.

Estas dúas vertentes son abordadas mediante o desenvolvemento da denominada metodoloxía de División e Desprazamento (Split and Shift (S&S)). Esta metodoloxía está destinada a xestionar a aplicación de matrices ou *kernels* de tamaño superior ao permitido pola conexión física (conexións local e circuíto de ponderación) en CPAs, incluíndo tanto a realización de operacións de gran veciñanza como a redución do número de conexións inter-cela (inter-PE) para a redución do consumo de área. No desenvolvemento da metodoloxía propoñemos varias técnicas con dous obxectivos principais: a mínima penalización en tempo de procesamento, e absolutamente ningunha penalización a nivel funcional. Ademais búscase tamén a minimización nas restricións do hardware CPA de partida e o menor impacto posible no mesmo. A única condición estrita é o requirimento de saídas definidas e predicibles, que nos leva a xeneralizar a arquitecturas discretas no tempo, aínda que arquitecturas continuas como o modelo continuo de CNNs permiten a aplicación parcial da metodoloxía con tamén interesantes resultados. Para o desenvolvemento da metodoloxía centrámonos nas CNNs de tempo discreto. O traballo de investigación desenvolvido está a medio camiño entre o nivel algorítmico e o nivel de implementación física no campo de matrices de procesador celulares orientadas ao procesamento de imaxe.

O compromiso entre área e tempo de procesamento derivado da aplicación da metodoloxía é valorada a través dunha Figura de Mérito (FdM) definida a tal fin. Xunto coa análise de forma das operacións a aplicar, este FdM permite: 1) propoñer conxuntos reducidos de circuítos de ponderación máis adecuados, e 2) xustificar a elección clásica da conexión NEWS (Norte-Leste-Oeste-Sur). A validación da proposta realízase a través de estimacións realizadas sobre implementacións físicas reais e con algoritmos do estado da arte ("state-of-the-art") como son o SIFT (Scale Invariant Feature Transform) e o SURF (Speed-Up Robust Features), que, por outra banda, non foron previamente aplicados sobre CPAs, o que constitúe outra contribución deste traballo.

O punto de partida da investigación foi permitir as comunicacións de longa distancia (traducidas en operacións de gran veciñanza) en redes de elementos de procesamento localmente conectados e con paralelismo de grado fino para o procesamento de baixo nivel de imaxes. Posteriormente, observouse que a metodoloxía desenvolvida para abordar este problema podía ser aplicada sobre a conectividade local co obxecto de reducir o número das conexións e dos circuítos de ponderación asociados a elas e, en consecuencia, a área dos elementos de procesamento.

Esta memoria comeza cunha revisión da particularización da arquitectura CPA adoptada, as redes celulares non lineais, concluíndo que representan unha boa opción na realización de microprocesadores visuais dado que proporcionan conectividade local, clave para alcanzar o paralelismo masivo esixido no baixo nivel de procesamento de imaxe. A metodoloxía proposta impón, con todo, unha restrición na elección do modelo de hardware; estados internos ben definidos e predicíbeis en calquera momento. En CNNs isto significa usar o modelo de tempo discreto ou o template de control do modelo de tempo continuo. Non hai máis restricións con respecto ó tipo de datos, de feito, a metodoloxía considerara dous modos de aplicación, un para implementacións traballando con niveis de grises e outro que é válido tanto para implementacións en niveis de grises como implementacións binarias que traballan con imaxes en branco e negro.

Os obxectivos de preservar a funcionalidade e mesmo ampliala coas comunicacións de longa distancia con modificacións mínimas de hardware tanto en procesamento binario como en escala de grises exclúen solucións suxeridas na literatura baseadas en dispositivos ou arquitecturas específicas, pero tamén metodoloxías que requiren manipulación de valores en niveis de grises ou que só sexan aplicables a arquitecturas binarias, así como propostas só aplicables a certos tipos de *kernels* como a obtención de comunicación de longa distancia mediante a aplicación reiterada de patróns de tamaño mínimo 3×3 . Nós buscamos unha proposta aplicable a calquera tipo de modelo lineal e adaptable a calquera particularización hardware con limitacións mínimas. É por iso que diriximos os nosos esforzos a solucións a nivel de sistema. Outras dúas metas para o desenvolvemento da nosa proposta son: 1) a sinxeleza de deseño e aplicación, e 2) unha penalización aceptable a nivel de tempo de procesamento. Tomando isto como base desenvolvemos a metodoloxía de Split e Shift (S&S) con base na propiedade asociativa da adición, mediante a cal un determinado patrón é dividido en patróns máis pequenos ou con menos coeficientes que son aplicados de forma independente e que recuperan o valor proporcionado polo patrón orixinal mediante a acumulación axeitada dos seus resultados individuais. Propuxemos así mesmo dous modos de aplicación, dependendo de como apliquemos os desprazamentos para reunir os resultados dos sub-patróns: des- prazando a imaxe para que a aplicación de cada sub-patrón ofreza o seu resultado na cela correcta (modo de desprazamento de imaxe) ou desprazando ós resultados parciais unha vez obtidos ata a cela onde deben ser acumulados (modo de desprazamento de resultado). A primeira opción fai que a metodoloxía sexa aplicable sobre implementacións completamente binarias que manexan imaxes en branco e negro.

Para a aplicación da metodoloxía desenvolvimos diversas técnicas tanto para a división de patróns como para o desprazamento de imaxe ou resultado e analizamos as consecuencias destas técnicas en hardware e tempo de procesamento.

Na emulación de gran veciñanza (LN) medimos o custo de estender a funcionalidade CPA como o número de operacións necesarias para a aplicación das operacións LN. Da análise das diversas técnicas desenvoltas concluímos principalmente que os métodos de separación deben comezar a partir dun canto dos patróns, solapando os sub-patróns incompletos segundo conveña para manter os seus centros próximos á cela central. Sobre as técnicas de desprazamento observamos a conveniencia de compartir desprazamentos tanto para desprazamento de imaxe como para desprazamento de resultado. Un proceso regular unido ao compartimento de desprazamentos ofrece beneficios en termos de simplicidade e de automatización. Suxerimos, como boas opcións, a técnica de descomposición concéntrica e as técnicas de desprazamento espiral ou zig-zag. Con todo, o desprazamento central ofrece resultados lixeiramente mellores en número de operacións que pode compensar a maior irregularidade en implementacións máis esixentes.

No caso de redución de hardware, temos un valor de compromiso entre o beneficio obtido na redución de hardware e o número de operacións necesarias para manter a funcionalidade do sistema. Isto non depende só do número de circuítos de ponderación (CC), senón tamén da configuración de cela seleccionada. Para a elección da configuración da cela propoñemos catro criterios.

O primeiro criterio asegura a preservación da funcionalidade completa, sen restricións sobre a forma de patrón orixinal ou tamaño do mesmo. Este criterio impón un número mínimo de 3 CCs e unha distribución de CCs que permita de forma directa ou indirecta todos os desprazamentos necesarios para a comunicación con todos os veciños.

O segundo criterio ten en conta o rendemento do sistema a través da definición dunha figura de mérito, a RPO (Redución de hardware Por Operación incrementada por cada operación orixinal), que permite escoller as técnicas e configuracións máis axeitadas. A RPO defínese como a relación entre a porcentaxe de CCs reducida (identificada como HR) e o incremento no número de operacións por operación orixinal. En xeral, para un único patrón seleccionaremos unha configuración de 6 CCs sen CCs na columna central como a mellor opción de valor de compromiso. Con todo, se se permite a distribución do CC nos dous patróns característicos dunha operatión CNN, obtemos mellor valor de compromiso cunha configuración de 3+1 CC en disposición lateral con desprazamento de resultados parciais, dado que se require un menor número de operacións con menor número de CCs debido ao solape entre operacións permitido. Para unha operación CNN de dous patróns, re-empregar o mesmo hardware para a aplicación de ambos patróns, tanto cos CCs nun único patrón como en dous, é a mellor opción.

O terceiro criterio na elección da configuración da cela xorde dunha análise máis profunda da definición de RPO e da evidencia de que a coincidencia da forma do patrón coa configuración da cela proporciona un mellor resultado. Afondando neste criterio realizamos un estudo da forma dos patróns contidos na librería de patróns máis representativa das CNN, a CSW. A partir deste estudo, concluímos que a maioría das operacións CNN mostran unha distribución diamante dos seus coeficientes e mostran simetrías, o que, combinado co desprazamento de resultado, pode servir para reducir o número de operacións. Operacións cun único CC central para realizar operacións aritméticas ou Booleanas son tamén significativas. Como consecuencia, a configuración diamante de 5 CC, é dicir, o NEWS clásico incluíndo o coeficiente de realimentación, representa unha boa opción de valor de compromiso, o que, ademais xustifica a eficiencia xeralmente aceptada para a conectividade limitada desta configuración NEWS.

O criterio final é, obviamente, o cumprimento dos obxectivos de deseño da implementación, que marcarían os límites reais no tempo de procesamento e ocupación da área. En consonancia con eses criterios, a aplicación da metodoloxía de división e desprazamento non ofrece técnicas rigorosas para a súa aplicación, senón que dá liñas xerais. Isto significa que podemos desenvolver distintas técnicas ou modos de aplicación con resultados similares, que serán mellores canto máis particularizadas sexan. A principal achega da análise de técnicas e configuracións é, pois, un conxunto organizado de directrices de aplicación para obter unha penalización mínima no tempo de procesamento e absolutamente ningunha penalización a nivel funcional.

A combinación da emulación LN e da simplificación do hardware é completamente asumible. Con todo, como a emulación LN esixe un número significativo de desprazamentos, a configuración da cela e as técnicas de desprazamento en LN deben escollerse de forma conxunta. O posible uso de simetrías (con desprazamento de resultado) e as configuracións con dous patróns tamén se amosan como un dos recursos máis vantaxosos.

Para validar as propostas aplicamos as técnicas desenvoltas a implementacións CNN reais documentadas na literatura e a algoritmos de baixo nivel no procesamento de imaxes. A partir da análise de implementacións físicas concluímos que, como se esperaba, a aplicación da redución de hardware é moito máis rendible en arquitecturas G/S onde os CCs son en xeral de maior tamaño, e onde a memoria analóxica local está normalmente incluída. Con todo, a redución de hardware obtida polas técnicas de división e desprazamento en implementacioóns binarias pode compensar a superficie ocupada pola LAM extra necesaria en xeral para estas implementacións, tamén se simplemente precisamos dotalas de comunicacións de longa distancia coa metodoloxía de división e desprazamento.

A análise de implementacións CNN sobre FPGA coa axuda da nosa metodoloxía confirma en xeral as nosas previsións de redución de hardware. Como a realización de celas con 9 CCs non cabe na nosa área de FPGA, os datos da súa execución non poden ser tomados como referencia numérica estrita para, por exemplo, a avaliación da porcentaxe de redución de hardware. Con todo, podemos tomar os valores relativos de área entre as distintas configuracións. Observamos que obtemos valores lixeiramente diferentes para disposicións diferentes de CCs, pero, en xeral, o factor HR demostrou ser unha boa ferramenta para a avaliación da área reducida na comparación de configuración de celas. Ademais, considérase probada a non significativa contribución das conexións inter-PE, neste caso, o que reforza a nosa elección da definición máis simple de HR. Debe notarse tamén que esta conclusión non se pode xeneralizar, pero en calquera caso, a diferenza entre o número de CCs e enlaces eliminados é, como máximo, de 2 e non implica diferenzas na comparación de configuracións máis alá da consideración de que configuracións co mesmo número de CCs ocupan menos espazo se un ou dous dos CCs se empregan para realimentación.

No aspecto de rendemento temporal estudamos a aplicación da metodoloxía a algoritmos de procesamento de imaxes de baixo nivel incluíndo comunicacións de LN. Neste caso, non nos limitamos a algoritmos CNN. De feito, os algoritmos SIFT e SURF non foron aplicados en CPAs antes, e a primeira conclusión é que a nosa metodoloxía permite a súa aplicación sobre elementos de procesamento localmente conectados e masivamente paralelo, a pesar das súas necesidades de operacións de gran veciñanza. Os resultados son de feito prometedores, estimándose que a xeración dun espacio de escalas de 4 oitavas no SIFT pode levar $\sim$ 1 ms con preto de 1000 operacións 3×3 nunha configuración 5 CC NEWS, e que a aplicación completa dos 12 filtros *Spin* 7×7 realízanse con un total de 136 operacións 3×3 nunha configuración NEWS con 4 CCs.

O caso da xeración do espazo de escalas no algoritmo SURF é un pouco diferente. A aplicación da imaxe integral sobre CPAs leva á paralelización do seu cálculo, algo buscado na literatura. Con todo, debido á especificidade da definición de imaxe integral, o paralelismo limítase a unha liña de cada vez na imaxe. Isto lévanos a propoñer o uso de LPAs no canto de CPAs, porque, ademais, o seu menor número de PE permite a utilización de memorias de maiores dimensións, o que resulta ser fundamental para a imaxe integral. Esta primeira parte do algoritmo redúcese a desprazamentos e acumulacións, coa excepción de aplicación do sub-patrón inicial empregado para reducir o número de operacións necesarias. A segunda parte da xeración do espazo de escalas no SURF implica a aplicación dos filtros *Box*, que poden ser considerados como operacións LN. Aplicados á imaxe integral, estes filtros son reducidas a unhas poucas adicións de valores situados a gran distancia, que poden ser realizadas coas técnicas de división e desprazamento sobre unha CPA. Neste caso, o número de operacións depende do tamaño de imaxe para o cálculo de imaxe integral, resultando en $N+M$ 3×3 operacións para un tamaño $M \times N$ de imaxe. A aplicación destes filtros pode levar preto de 3000 operacións para catro oitavas, tanto para un patrón completo de 9 CCs como para nha configuración NEWS de 5 CCs, sendo outro exemplo da ineficiencia de implementarunha configuración con 9 CCs.

Para a análise completa do valor de compromiso optamos por unha implementación orientada á aplicación do algoritmo PLS. Á vista dos resultados temos que coa metodoloxía proposta non só se aumenta a funcionalidade da implementación permitindo as operacións LN, senón que as melloras de área superan as inconveniencias da introdución dunha memoria analóxica de acumulación. Nótese que, neste caso, o principal aforro de área ven da retirada de conexións locais xa que ocupan o 80% da área reducible.

A análise do valor de compromiso tamén nos permitiu comprobar o grao de correspondencia entre a configuración da cela e os resultados esperados da análise xeral das técnicas de división e desprazamento. Obviamente, eliximos configuracións que respectan a plena funcionalidade da cela. Tamén analizamos a forma dos patróns implicados, concluíndo, como no estudo xeral, que a conectividade NEWS é a máis axeitada. Vimos tamén que en configuracións con poucos CCs e aplicando o modo de desprazamento de resultado é interesante distribuír o CCs en dous patróns.

Recollemos tamén a análise de forma dos patróns implicados en varios algoritmos suficientemente detallados na literatura CNN, incluíndo o PLS. As estatísticas a partir desta análise apoian a elección da conectividade NEWS en 4 dos 5 algoritmos analizados. Tamén é interesante o número de ocorrencias de operacións que implica só o CC central como operacións lóxicas locais, operacións aritméticas ou incluso de saturación, a partir do que se pode concluír a conveniencia de incluír tamén o CC de realimentación, polo menos nun dos patróns.

Como na validación, a nosa perspectiva sobre o traballo futuro ten dúas liñas principais, a algorítmica e a hardware. Dentro da liña algorítmica propoñemos a xeración do espazo de escalas do algoritmo SIFT en plataformas CPA. A aplicación da metodoloxía de división e desprazamento sobre arquitecturas totalmente dixitais comprendendo só unha ALU por PE, ou un MAC é tamén unha cuestión de traballo futuro. Un segundo obxectivo na liña algorítmica é a adaptación da metodoloxía para a súa aplicación sobre arquitecturas con menor grao de paralelismo, onde os elementos de procesamento tratan con varios píxeles no canto de só un.

Dentro da liña de hardware, temos tres aspectos principais: a análise das implicacións da metodoloxía sobre o consumo de enerxía e sobre a precisión requirida polos circuítos de ponderación, e a implementación de memorias analóxicas axeitadas ao labor de acumulación requirido pola metodoloxía.

Sobre o consumo de enerxía esperamos un menor consumo instantáneo debido ao menor nmero de CCs, pero quizais maior consumo medio debido ao maior número de operacións e ao consecuente maior tempo de procesamento. Con todo, se se considera que as ponderacións por coeficientes nulos tamén consumen enerxía, a redución do número de circuítos de ponderación aplicada xunto coa elevada incidencia de patróns pouco densos dentro das operacións CNN conduciría a unha mellora neste aspecto.

Con todo, a precisión requirida impón un mínimo no consumo de enerxía dun circuíto. E este, xunto coa maior área requirida por unha maior precisión, lévanos á segunda análise sobre o hardware. Nun principio agardamos que a non igualdade entre transistores nominalmente idénticos, a fonte principal de erro nun circuíto analóxico, diminúa a medida que o número de compoñentes tamén se reduce. Ademais, a área liberada pola eliminación de CCs pode usarse tamén para mellorar a precisión.

Finalmente, aínda que as arquitecturas de tipo G/S xa ofrecen memorias analóxicas que poden usarse para a metodoloxía de división e desprazamento, sería interesante atopar unha memoria de tamaño mínimo para as arquitecturas binarias. Ademais, os moitos ciclos necesarios para unha aplicación real poden obrigar tamén a adoptar algunhas estratexias para refrescar a memoria, a fin de evitar a degradación de valores almacenados en memorias analóxicas.

Contents

Preface

In multimedia era, image processing has become a very important element on electronic devices. From communications (e.g. telemedicine) to security (e.g. retinal recognition) or industrial processes/quality control (e.g. articulated arms guidance, product defects detection) going through research (e.g. elemental particles tracking) and medical diagnosis (e.g. strange cells detection, retinal vessels identification), there is a huge number of applications where the automatic image treatment or even understanding is fundamental. The ultimate goal would be the design of vision systems with decision-making capability. In addition, current trends require the combination of these capabilities on small and portable devices with real-time or at least fast response. This poses new challenges in both hardware and software design in image processing, looking at new architectures or structures with the lowest possible area and power consumption and without compromising the functionality and performance. The contributions of this thesis focus on the optimization of area usage and the improvement of the functionality of vision systems based on Cellular Processor Arrays (CPAs), being particularized for Cellular Neural Networks (CNNs). The research presented is placed midway between the algorithm and hardware level design. In the following we try to contextualize the realized work by going through the different abstraction levels.

Image processing (Task level)

Image processing is a complex task that can be divided in three differentiated levels of sub-tasks that are connected hierarchically [Dudek, 2000]. Low-level image processing tasks, or 'early vision', require no additional knowledge and act locally in the image, independently of the content, preparing the data for the next level. Tasks included at this level are usually very simple low precision repetitive convolution-like operations, usually oriented to restoration or feature enhancement. Nevertheless, they are computationally highly demanding due to the large quantity of data to process. Intermediate level image processing tasks extract symbolic information about the image from the data provided by the previous level through global methods mainly. The quantity of information required here is low, tasks are more complex and they operate over the preprocessed data, i.e. over a little part of the data originally contained in the image. Finally, the high level processing involves complex tasks directed to understand in some way the content of the image. They use the symbolic description provided by the intermediate level and require a significant quantity of additional information to interpret the image.

Image processors (Hardware level)

A vision system includes these three levels with the aim of making autonomous decisions. This is what is called Computer Vision. Although these operations can be done on a classical von Neumann computer, the high computational load and the inherent parallelism (specially, both, in the low-level phase) make it a non-suitable option for image processing. Note that although the operations performed at high-level are far more complex, they are the lower level operations which on many occasions set the bottleneck in the algorithm as they represent more than the 50% of the computational load [Nudd, 1980]. Modern processors include parallel units and replicated units and exploit instruction level parallelism. Nevertheless their general-purpose floating point orientation makes them not particularly efficient in low-level image processing (more than 50% of the load) apart from the waste of resources not needed. Digital Signal Processors (DSP) are optimized for signal processing and can be suitable for low demanding applications.

But market drivers in the semiconductor industry demand ever more functionality on portable gadgets with as high a reliability as possible. Some medical instrumentation, cell phones or any other portable device in consumer electronics like digital cameras are clear examples of such demands. From the perspective of the circuit designer, these specifications are translated into programmable integrated circuits with as low a power dissipation as possible and small area. On many occasions, the resultant circuits become actual systems-on-chip (SoC) with heterogeneous technologies. This might be the case of a digital camera, or a mobile phone, where sensing and processing could be built up on different semiconductor technologies, and where analog and digital computation could be laid down on the same substrate. Systems-on-chip comprising processing elements (PEs) working in parallel and customized for specific functions combined with local and global memory along with peripheral control circuitry are posed by the ITRS (International Technology Roadmap for Semiconductors) as power-efficient architectures to meet the demands of some of the above market drivers [ITR, 2009-2010].

Cellular Processor Arrays for Computer Vision: Vision Chips

Cellular Processor Arrays (CPAs) suit this architecture. CPA chips usually contain a main stored-program memory within a global control unit that issues and broadcasts the instructions to be executed by an array of PEs. This array executes the same instruction over different data on every PE appearing as a massive data parallel system (Single Instruction Multiple Data, SIMD computation). 2-dimensional CPA mesh with a pixel to processor correspondence is, then, a natural implementation of low-level image processing operations.

The work of Unger in the 1950s represents the initial work in this sense [Nudd, 1980]. Since then, technological evolution and reduced complexity PEs have made it possible to have these SIMD solutions implemented even onto a single chip. Particular implementations go from dedicated hardware implementing a specific algorithm to universal machines, and from Application Specific Integrated Circuits (ASICs) to reconfigurable hardware as Field-Programmable Gate Arrays (FPGAs). CMOS sensors have allowed to integrate imager and processor on the same die, eliminating the imager-processor bottleneck and giving birth to the focal-plane processors or Vision Chips [Moini, 2000]. In addition, the need to optimize critical performance parameters like area, processing time or power consumption leads to CPAs with PEs partitioned into several customized modules, each specialized in a particular function and being the general program which decides in which module a certain function will be executed [Földesy et al., 2007, Lopich and Dudek, 2011a]. Another aspect that influences the suitability of the PEs as it affects area and processing time, is the number of connections, i.e. the number of weighting circuits employed for collecting the contributions from neighboring PEs in convolution type operations and their associated routing. This is particularly important when large neighborhood operations are involved. The research work presented in this thesis deals with this aspect.

All in all, focal-plane processing is particularly suitable for low-level image processing but it is not efficient when dealing with high-level image representation. In a whole vision system, the combination of SIMD with other paradigms of computation on the same monolithic solution would be, then, an option. The advent of new emerging technologies like CMOS-3D opens the way for such solutions. In the particular case of a CMOS-3D-based architecture the functionality is distributed among different tiers, which might lead to low- as well as medium- and high-level processing on the same monolithic solution [Rodríguez-Vázquez et al., 2010].

Cellular Non-linear Networks

Cellular Non-linear Network (CNN)- Universal Machine (CNN-UM) [Roska and Chua, 1993] is a specific proposal of general purpose CPAs that can be integrated on a single chip. The original CNN paradigm [Chua and Yang, 1988a] includes the possibility of spatial dependent (i.e. Multiple Instruction Multiple Data -MIMD- architecture) and non linear operators. For us CNN chips are conceived as vision chips for low-level image processing. In this case it is generally enough with linear spacial invariant operators or “cloning templates” (i.e. SIMD) that operate identically over each pixel and taking into account a certain neighborhood. Local connections, non-linear output robustness and simple SIMD control make CNNs suitable for hardware implementation, convolution-like operators with global processing capability and massive parallelism make them suitable for low-level image processing.

We have developed our work over the CNN paradigm. Nevertheless, our proposals are general enough to be extended to similar CPA implementations with the same restrictions. This work contributes to two main issues in CNN architecture, namely the extension of the functionality to operations implying large neighborhood communications initially limited by the local connectivity and the area saving through the reduction of the number of local connections and weighting circuits.

Research contributions

In this work we develop the so-called Split and Shift (S&S) methodology. This methodology is intended to deal with the implementation of kernels of sizes that overflow the physically implemented connectivity (local connections and weighting circuits) on

CPAs, including the realization of large neighborhood operations and/or the reduction of the inter-PE connectivity in order to drop the area consumption. In the development of the methodology we propose several techniques under two main goals: minimum penalty at processing time, and absolutely no penalty at functional level. The area-processing time trade-off derived from the application of the methodology is assessed through an ad-hoc Figure of Merit (FoM). Together with a kernel shape analysis, this FoM allows us to propose more adequate reduced sets of weighting circuits and to justify the classical choice of NEWS (North-East-West-South) connectivity. The validation of the proposal is realized by means of estimates over actual physical implementations and state-of-the-art algorithms as SIFT (Scale Invariant Feature Transform) and SURF (Speeded-Up Robust Features) algorithms, that, on the other hand, have not been previously implemented over CPAs. The methodology is applicable in general over synchronous binary (B/W) or gray-scale (G/S) image-processing CPA implementations. For the development of the methodology we have focused on the Discrete-Time CNN model [Harrer and Nossek, 1990].

During the research time we have gathered the contributions in several publications that are listed below:

N. A. Fernández, D. L. Vilariño, V. M. Brea, D. Cabello, " **On the Emulation of Large-Neighborhood Templates with Binary CNN-Based Architectures**," in *Proceedings of the* 9th *IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA 2005*, pp. 274-277, Hsinchu, Taiwan, May 2005.

N. A. Fernández, D. L. Vilariño, V. M. Brea, D. Cabello, " **Large Neighborhood Templates with Nearest-Neighbor Connected Patterns in Binary-Based Cellular Neural Networks**", in *Proceedings of the XX Conference on Design of Circuits and Integrated Systems, DCIS 2005*, Lisbon, Portugal, November 2005.

N. A. Fernández, V. M. Brea, D. L. Vilariño, D. Cabello, " **On the Reduction of the Number of Coefficient Circuits in a DTCNN Cell**," in *Proceedings of the* 10th *IEEE International Workshop on Cellular Neural Networks and their Applications,CNNA 2006*, Istanbul, Turkey, August 2006.

N. A. Fernández, V. M. Brea, D. L. Vilariño, D. Cabello, " **Hardware Simplification in Cellular Non-linear Networks for Complex Algorithms**," in *Proceedings of the XXI Conference on Design of Circuits and Integrated Systems, DCIS 2006*, Barcelona, Spain, November 2006.

N. A. Fernández-García, V. M. Brea, D. Cabello, " **Area and Time Efficient Cellular Non-linear Networks**," in *Proceed. of IEEE International Symposium on Circuits and Systems, 2007. ISCAS 2007*, pp.2682-2685, New Orleans, USA, May 2007.

N. A. Fernández-García, J. Albó-Canals, V. M. Brea, J. Riera-Baburés, D. Cabello, X. Vilasís-Cardona, "**Verification of Split&Shift techniques for CNN hardware reduction**,"in *Proceedings of the* 18th *European Conference on Circuit Theory and Design, 2007. ECCTD 2007*, pp.88-91, Seville, Spain, 27-30 August 2007.

N. A. Fernández García, M. Suárez, V. M. Brea, D. Cabello, "**Template-oriented hardware design based on shape analysis of 2D CNN operators in CNN template libraries and applications**," in *Proceedings of the 11th International Workshop on Cellular Neural Networks and Their Applications, 2008. CNNA 2008*, pp.63-68, Santiago de Compostela, Spain, July 2008.

N. A. Fernández, V. M. Brea, M. Suárez, D. Cabello, " **Scale- and Rotation-Invariant Feature Detectors on Cellular Processor Arrays**," in *Proceedings of IEEE International Symposium on Circuits and Systems, 2012. ISCAS 2012*, pp.2657-2660, Seoul, Korea, May 2012.

N. A. Fernández, V. M. Brea, D. Cabello, "**Split and Shift Methodology on Cellular Processor Arrays: Area Saving vs. Time Penalty**," under review in *International Journal of Circuit Theory and Applications* with major revisions (May, 2012).

Other published contributions not belonging to the main line of the thesis but related to it:

V. M. Brea, M. Laiho, N. A. Fernández, A. Paasio, D. Cabello, " **Relating Cellular Non-linear Networks to Threshold Logic and Single Instruction Multiple Data computing models**,"in *Proceedings of the* 18*th* *European Conference on Circuit Theory and Design, 2007. ECCTD 2007*, pp.92-95, Seville, Spain, August 2007.

Jordi Albó-Canals, N.A. Fernández-García, Jordi Riera-Baburés, Victor M. Brea, Diego Cabello, " **Discrete Time Cellular Non-linear Networks Implementation over FPGA**," in *Proceedings of the XXIII Conference on Design of Circuits and Integrated Systems, DCIS 2008*, Grenoble, France, November 2008.

A. Nieto, N.A. Fernández-García, Jordi Albó-Canals, V. M. Brea, D. L. Vilariño, Jordi Riera-Baburés, Diego Cabello-Ferrer, " **Single Instruction Multiple Data and Cellular Non-linear Networks as Fine-Grained Parallel Solutions for Early Vision on FPGAs**," in *Proceedings of the XXIII Conference on Design of Circuits and Integrated Systems, DCIS 2008*, Grenoble, France, November 2008.

J. Albó-Canals, J.A. Villasante-Bembibre, J. Riera-Baburés, N.A. Fernández-García, V.M. Brea, "**An efficient FPGA implementation of a DT-CNN for small image gray-scale pre-processing**," in *Proceedings of the European Conference on Circuit Theory and Design, 2009. ECCTD 2009*, pp.839-842, Antalya, Turkey, Aug. 2009.

Thesis overview

This manuscript is divided in five chapters and two appendices. The first chapter reviews the main characteristics of the CPA paradigm over which we have developed the research work. The research motivation and the related work found in the literature are gathered in the second chapter. In the third chapter we describe the methodology proposed in depth, and we develop the required techniques and guidelines for the two challenges considered, large neighborhood templates application and connectivity reduction, and for the combination of both. A fourth chapter presents the validation tests realized and analyzes the results obtained. The proposed CPA implementation of the SIFT and SURF algorithms deserves a particular comment as they are not only part of the validation of the methodology, but also part of the main contributions of the thesis as it is the first time in the literature where those algorithms are proposed to be implemented over CPAs with a pixel per processor approach. The final chapter summarizes the main conclusions drawn from the research work, and presents future work lines.

Finally, two appendices gather the papers where the main contributions of the thesis were published. The first appendix conveys the papers directly related to development and validation of the methodology. In the second appendix we have included those papers that gather a particular usage of the methodology, including a theoretical reflection over the relationship between the SIMD paradigm, the Threshold Logic and the application of our methodology to CNNs. Along the text we refer the published papers in several occasions from a critic point of view to provide other ways of illustrating the proposals as long as to indicate discrepancies that illustrate the evolution of the research work.

Chapter 1

Cellular Non-linear Networks

In this chapter we introduce the main aspects of the CPA particularization we have chosen to develop our research. We present the original paradigm, its generalization to universal machine and the modifications introduced in order to improve its versatility and hardware implementation. In the last section we review some hardware implementations found in the literature to give an overview of the main milestones and the state-of-the-art.

1.1 The CNN Paradigm

The Cellular Non-linear Network (CNN) is a massively parallel paradigm that has appeared as very suitable for both real time image processing and hardware implementation. First introduced by Chua and Yang in 1988 [Chua and Yang, 1988a,b], this paradigm gathers the key characteristics of neural networks (parallelism and global interaction) within a structure typical of cellular automata (local interactions and regular spatial distribution together with the parallel processing).

The original CNN paradigm is conceived as an analog massive parallel processor composed of an N-dimensional mesh of identical, analog and dynamical processing elements (PEs), cells in CNN literature. These cells interact in a limited radius n but the system keeps the global interaction capacity thanks to the propagative effects of the local interactions [Chua and Roska, 1993, 2002]. For image processing we typically consider 2-dimensional topologies with $n = 1$, what is, in addition, more suitable for integrated circuit implementation (Fig. 1.1). It is usual to consider a pixel to cell correspondence, natural assignment if we think of vision chips with sensor integration [Moini, 2000].

Cells interaction is governed by two synaptic operators. The first one, initially called *feedback operator/template/kernel* and identified as $\mathbf{A}$, acts over the *state* variable X (through the output value Y obtained after applying the output function) of the cells in a radius n, including the own cell under consideration. The second operator is identified as $\mathbf{B}$ and was called *control operator/template/kernel*. It acts over the *input* data U (either an external image or stored data) of the cells in the same radius. The bias term I modules the effect of the operators. The global result is gathered in the central cell state. This dependence of the cell state on the neighboring cells' values

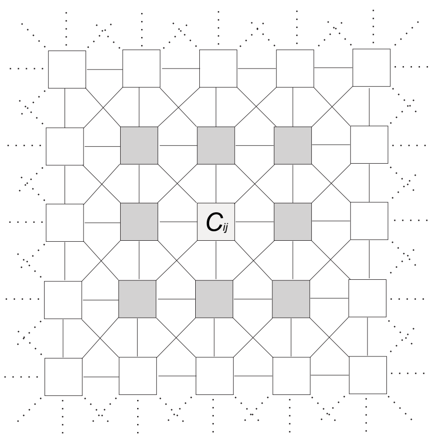


Figure 1.1: 2D square mesh with 8-connectivity and $n = 1$. Cells in gray are physically connected to the cell under consideration C_{ij}.

explains the capacity of global processing of the system.

The general case includes the possibility of non linear and spacial dependent operators what would mean a Multiple Instruction Multiple Data (MIMD) architecture. For image processing it is generally enough with linear spacial invariant operators or *cloning templates* that operate identically over each pixel (i.e. Simple Instruction Multiple Data -SIMD-). In our case the operators are, then, weighting templates or matrices that act over the input ($\mathbf{U}$) and output ($\mathbf{Y}$) images in a convolution-like way. On its side, the bias term could be in general different from cell to cell but it is typically constant in the mesh.

The dynamics of the original model, considering linear operators, is gathered in a system of coupled differential equations of first order like that shown in Eq. 1.1. ij indices refer to the cell to be processed and kl ones refer to the cells in the neighborhood N_n around the ij cell. a_{ijkl} are the corresponding template A coefficients, b_{ijkl} the corresponding template B coefficients, i_{ij} the bias term at the processing cell position, and y_{kl}, u_{kl} and x_{ij} the output, input and internal state at the indicated cells.

$$\frac{d}{dt}\, x_{ij}(t) = -x_{ij}(t) + \sum_{k,l \in N_n(i,j)} a_{ijkl}\, y_{kl}(t) + \sum_{k,l \in N_n(i,j)} b_{ijkl}\, u_{kl} + i_{ij} \qquad (1.1)$$

The output of the cell is determined by a non-linear function of the cell state that can be of several types [Chua and Roska, 1993, Roska and Rodríguez-Vázquez, 2002]. Typically we choose an output function with strictly monotone-increasing behavior within a linear region and saturated values outside that linear region. The choice of the output function affects not only its own hardware implementation but the processing type and the physical implementation of the whole cell as long as it influences the range of variables' values. The original proposal in [Chua and Yang, 1988a] is a piecewise linear function (Eq. 1.2, Fig. 1.2(a)). Nevertheless, it is usually substituted by a sigmoid function (Eq. 1.3 Fig. 1.2(b)) in physical implementations as it is easily obtained from an amplifier. The threshold function (Eq. 1.4, Fig. 1.2(c)) implemented with a comparator is an interesting option that allows optimizations in the physical implementation but it can only be applied to binary output processing [Brea Sánchez, 2002].

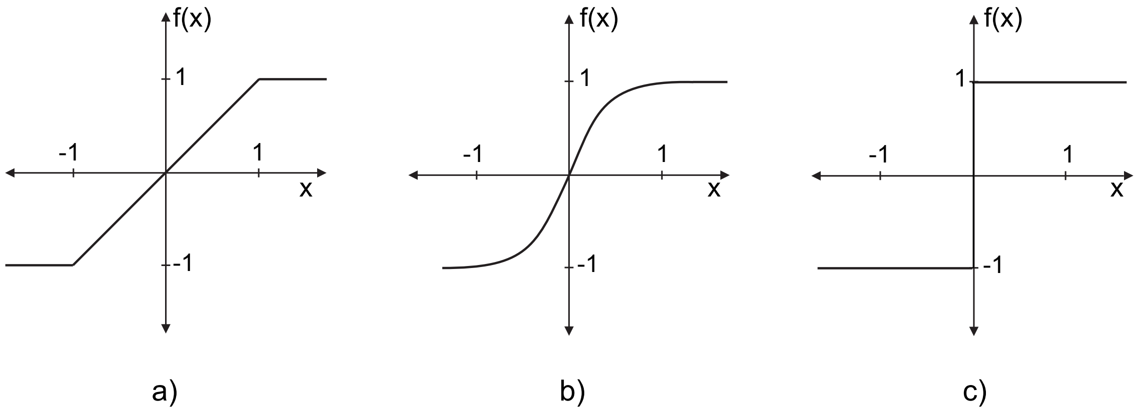


Figure 1.2: Output functions: *a)* piece-wise linear (Eq. 1.2), *b)* sigmoid (Eq. 1.3) and *c)* threshold (Eq. 1.4).

$$f(x) = \frac{1}{2}\left(|x-1| - |x+1|\right) \tag{1.2}$$

$$f(x) = \frac{2}{e^{-mx}} - 1 \tag{1.3}$$

$$f(x) = \begin{cases} -1 & x < 0 \\ 1 & x > 0 \end{cases} \tag{1.4}$$

The system level structure of a basic CNN cell is summarized in the block diagram of Fig. 1.3. **U** and **Y** refer here to the inputs and outputs of the cells in the neighborhood selected, including the cell to be processed. **I** is the bias term. **x** and **y** are, respectively, the internal state and the output of the central cell (the cell under processing). **A** and **B** are the weighting templates. All these elements take continuous values as a general rule. **G** is the law that governs the system dynamics that is described in the state equation (Eq. 1.1) and **F** is the output or transfer function.

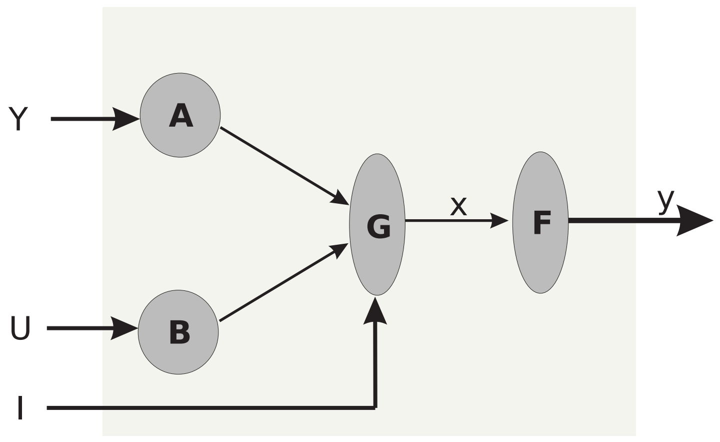


Figure 1.3: Functional block diagram of a CNN cell.

Once the cell dynamics is fixed, the interaction templates (A and B) and the offset term I determine the functionality of the CNN mesh. Considering a typical two-dimensional 8-connectivity topology, translation-invariant templates, and neighborhood order of $n = 1$, we need 19 coefficients at most to implement a particular task (9 coefficients per template and one more for $\mathbf{I}$), independently of the number of cells in the array.

1.2 The CNN Universal Machine

To allow the execution of complex algorithms on a CNN system it is necessary to add new functionalities, namely, template programmability, local memories and global control mainly. In this direction authors in reference [Roska and Chua, 1993] introduce the CNN Universal Machine (CNNUM) concept to fully exploit the CNN processing capabilities. In fact, authors in reference [Crounse and Chua, 1996] show CNNUMs as universal in Turing-sense. Besides, it opens the possibility of stored-program computation, i.e. software reconfigurability, making it possible to realize an enormous quantity of tasks over the same hardware. It is necessary, nevertheless, that the area and time for storing and changing an instruction is negligible compared to the processing area and the execution time of a nontrivial instruction respectively. This is only possible if we consider a translation-invariant cloning template (**A, B** and **I**), that requires the specification of just 19 coefficients per operation for $n = 1$.

The CNNUM model consists of a matrix of extended cells and a *Global Analogic*[1] *Programming Unit* (**GAPU**), see Fig. 1.4. The extended cell includes the original CNN core, some *Local Analog Memories* (**LAM**), and some *Local Logic Memories*, (**LLM**). In addition, it includes units to realize simple analog operations (additions and subtractions) and basic logic operations at cell level (*Local Analog Output Unit* -**LAOU**- and *Local Logic Units* -**LLU**- respectively), although these operations could be done through CNN templates. They are redundant but require little hardware and are more efficient in terms of computation time than the equivalent CNN operation. The set is completed with a *Local Communication and Control Unit* (**LCCU**) that governs the extended cell behavior in relation to the other cells and to the GAPU. The GAPU is mainly composed of analog and digital program and configuration registers (*Analog Program Register* -**APR**-, *Logic Program Register* -**LPR**-, *Switch Configuration Register*, **SCR**), used to keep the analog and digital instructions and the configuration of the cells during the program execution, and a *Global Analogic Control Unit* (**GACU**), that controls the instructions issuing, timing, data transferences and synchronism of the communication with external control devices. Finally, it is possible to include an *Optical Sensor* (**OPT**) to acquire the images directly over the processing network what results in a proper vision chip.

[1]The CNNUM introduces a new kind of computation that incorporates analog and logic operations separately, without A/D and D/A converters. It is called ‘analogic’.

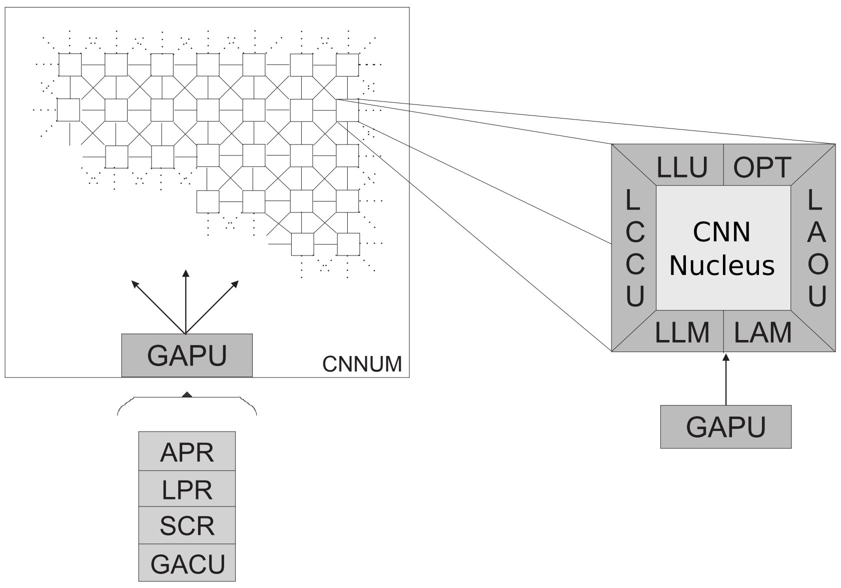


Figure 1.4: Global architecture of a CNNUM [Roska and Chua, 1993].

1.3 Discrete Time CNNs

Although time discretization was already used to show the suitability of CNNs for image processing in [Chua and Yang, 1988b], the implementation of CNNs working in discrete time was firstly proposed in 1990 in [Harrer and Nossek, 1990].

The DTCNN dynamics is governed, according to its original proposal, by a system of synchronous linear equations like that in Eq. 1.5, where T represents the temporal step, ij is the central cell location and kl the location of a neighboring cells within the n order neighborhood N_n. DTCNN definition is completed with a threshold output function (Eq. 1.4) that restricts the outputs to binary values.

$$x_{ij}\,(T+1) = \sum_{k,l \in N_n(i,j)} a_{ijkl}\; y_{kl}(T) \; + \sum_{k,l \in N_n(i,j)} b_{ijkl}\; u_{kl}(T) \; + i_{ij} \tag{1.5}$$

Thanks to the temporal discretization it is possible to control and even predict the values of all variables in each temporal step without the uncertainty of controlling a transient by estimations of the time constant of a CTCNN. As a consequence, processing velocity is improved for non-propagative operations, being it not necessary to wait up to 5 times the time constant (the estimated settling time) to consider an output as stable, what, furthermore, makes unimportant the lack of a global convergence theorem for DTCNNs ([Harrer and Nossek, 1990]). On the other hand, global processing capacity with local connectivity is preserved through a synchronized feedback of the binary outputs in a radius n in each temporal step, instead of the transient dynamics of the continuous time model (CTCNNs) that takes continuous feedback in time and value. Nevertheless, this makes CTCNNs faster in propagative operations.

Another advantage of DTCNNs is the easy interconnection between different levels of processing, even with different architectures [Harrer and Nossek, 1993]. This allows the implementation of multilayer architectures that can be implemented with one-layer reconfigurable architectures, i.e. through time variant templates. This advantage is completed with the ease of template designing either heuristically or through the resolution of linear non-equalities systems provided by the binary outputs, both thanks to the exact prediction of outputs [Harrer and Nossek, 1990]. Furthermore, template **A** can be used independently and interchangeably with template **B** thanks to output control and threshold output function that makes it not necessary to wait for output saturation. In addition, this makes it possible the combination of two operations in one.

At implementation level we have advantages in intercommunications, chip testing, chip design and even chip simulation. In the first one, the characteristic of binary and synchronous output make interconnections between different circuits and communication with the outer world easier and more reliable. Secondly, it is possible to control the propagation velocity through the modification of the system clock, what simplifies the chip testing process. With reference to chip design threshold function implies an improvement in the system robustness [2] and the physical design can be eased by an adequate selection of the template coefficients. Finally, chip simulation is less costly due that it is not necessary to implement numeric integration algorithms [Brea Sánchez, 2002, Vilariño, 2001].

1.4 Hardware Oriented Variations of the CNN Model

Since the original model was introduced in 1988, several modifications to improve the implementability of CNNs systems have been proposed. These modifications affect the highest levels of design, i.e. the model description. Apart from improvements in the output function implementation, the proposals are mainly focused on improving the weighting or coefficient circuits implementation given their importance in the main figures of merit, namely area and power consumption and their influence in the processing time. The modifications basically affect the output function definition and the variables (inputs, outputs, state and template coefficients) range.

With reference to the variables range, variables are originally continuous and restricted to $[-1, 1]$ (-1 corresponds to white and 1 to black in CNNs for image processing) in the case of input and output, and are non-restricted in the case of the state and template coefficients. It is important to take into account that modifications over the range of some variables will affect the values of other variables to keep the input-output mapping. The same occurs with the output function definition and the variables' ranges, what stresses the importance of no strict restrictions over the output function.

Further improvements can be obtained by focusing on template design. Sparse templates will lead, for example, to smaller power consumption and better robustness values and robustness can be improved as well for particular architectures [Paasio and Dawidziuk, 1999, Brea et al., 2005a].

[2] An important issue in the determination of template coefficients is the robustness, defined as the capacity of preserving the input-output mapping from variations over the nominal values of the physical elements of the circuit. It can be translated into the coefficient values tolerance and will mark the accuracy required in the circuit, that is key in the circuit size [Paasio and Dawidziuk, 1999].

1.4.1 Full-Signal Range Model (FSR)

This model modification restricts the state range to $[-1, 1]$. With this, output and state are equivalent at every moment and, as a consequence, it is not necessary to implement the output function if we consider a one-slope linear function between $[-1, 1]$. This proposal reduces area and power consumption at the same time that the limited state excursions improve the processing time. It led to the largest gray-scale (G/S) implementation at that time with 128×128 cells [Rodríguez-Vázquez et al., 1993, Espejo et al., 1994]. This model can be combined with a high-gain non-linearity to add to the limited state improvements the inherent robustness and fast convergence of this output function.

1.4.2 2Q, 1Q and 1Q-1bit Coefficient Circuits

The coefficient or weighting circuits are the circuits associated to the local connections that play the function of weighting the neighbors' contributions if we see them from the template point of view (Fig. 1.5), or that weights the cell value to send it to the neighbors if seen from the hardware point of view (Fig. 1.6).

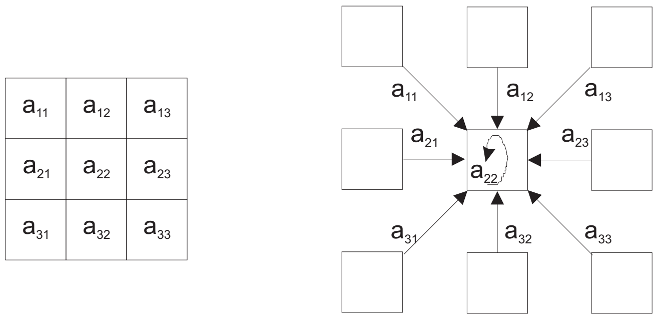


Figure 1.5: Inter-cell communications. Template perspective.

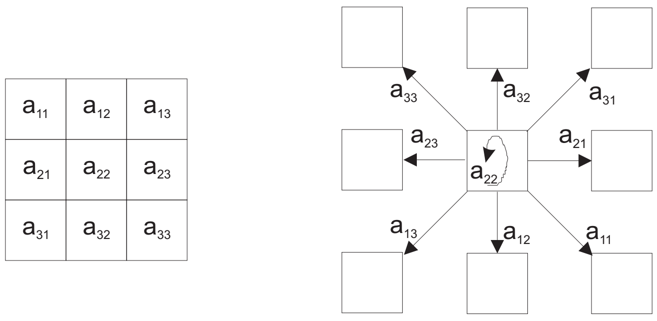


Figure 1.6: Inter-cell communications. Hardware perspective.

The high level optimizing proposals focused on the multipliers implementing the coefficient circuits can be summarized in the reduction of the number of quadrants of operation (Fig. 1.7) and the operands programmability reduction.

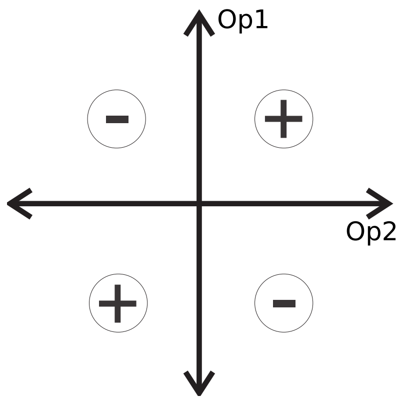


Figure 1.7: Two operator product quadrants.

2Q Coefficient Circuits

The first step is the reduction from four to **two quadrants** of operation. In this case, inputs and outputs sign is limited to positive or negative and the result of the weighting can only fall in two of the four possible quadrants (Fig. 1.7). The use of 2Q multipliers [Mead, 1989] improves area and power consumption, and processing time with respect to the full four quadrant multipliers like those in [Gilbert, 1968].

For input range, the transformation is directly realized in the codification of the image, independently of the CNN cell. On the other hand, the output range transformation requires the output function modification as is shown in [Hegt et al., 1998] or, more generally, in [Fernández García, 2006].

In [Paasio, 1998] it is defined a positive range model. In this case the input and output range changes from $[-1, 1]$ to $[0, 1]$. Fig. 1.8 shows the piece-wise-linear and threshold output functions for positive range outputs. To keep the input-output mapping these transformations will entail modifications over the template coefficients [Paasio, 1998, Fernández García, 2006].

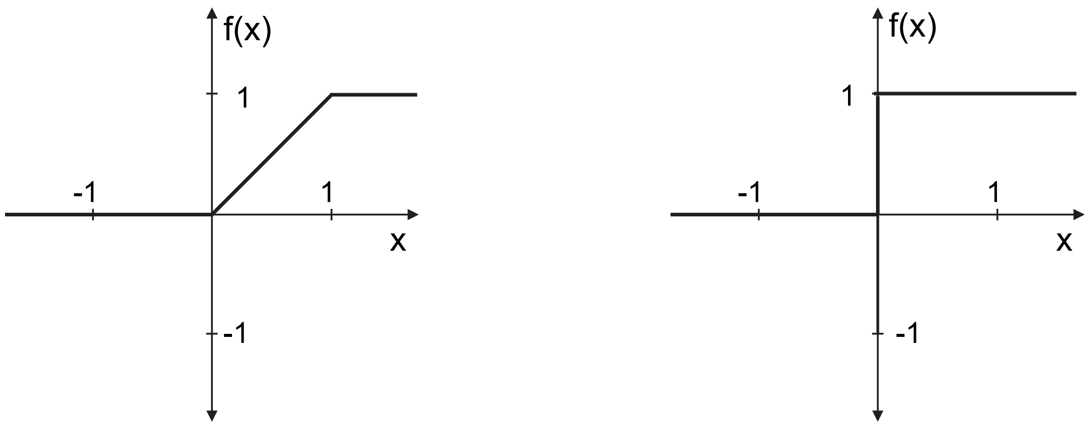


Figure 1.8: Positive range output functions.

Authors in reference [Paasio and Halonen, 2001] introduce as output function non-linearity a combination of the positive-range, high-gain and limited state range proposals, i.e. the Positive range High gain State limited CNN model (PHS-CNN). This is an easily implementable option that offers a very simple structure for multipliers implementation. In combination with the robustness and fast convergence of the high-gain model a reduced area consumption is expected as well as a higher processing capacity. As a consequence the processing is limited to binary outputs.

1Q Coefficient Circuits

A step forward is to add to the input/output sign restriction, the limitation in sign of the template coefficients. With this, we have just positive or negative operands and the weighting result can only fall into **one quadrant** (Fig. 1.7), with positive or negative values. This made it possible to implement the coefficient circuits with a reduced area consumption, just with NMOS or PMOS transistors, improving as well the processing time and the power consumption. Apart from the heuristic decomposition of the operations that can be used to obtain the new template coefficients, an analytic method is introduced in [Brea et al., 2004a].

1Q-1bit Coefficient Circuits

In addition to the 1Q implementation, the restriction of one or two of the operands (template coefficients or input/output values) to **1-bit** values (0 or 1) makes the programming and the computation simpler and faster, and reduces the circuit connections given that there is a unique digital signal programming [Paasio et al., 2004, Flak et al., 2004]. This is called *reduced programmability*. Binary template coefficient utilization requires the redefinition of the templates and will usually increase the number of operations [Laiho et al., 2005], partly compensated for the processing time improvement. On the other hand, the utilization of a high-gain non-linearity to provide binary outputs contributes to a less restrictive hardware design thanks to its inherent robustness, at the same time that it simplifies the output function implementation [Paasio, 1998, Paasio and Halonen, 2001].

The restriction of input/output range to binary values introduces limitations in the processing and in the initial conditions. In particular, the limitation to binary image processing with the introduction of a high-gain non-linearity, makes it impossible to realize operations like gray-scale gradients detection, for example.

It is interesting to note that the positive range models (both 2Q and 1Q) and even the reduced programmability templates are less aggressive modifications than the high-gain non-linearity and/or the restriction to binary inputs, given that in the first case there is not a limitation on the system functionality but just affects to the template design and ranges definition.

DTCNNs experience the same evolution as CTCNNs with respect to the number of quadrants required for weighing circuits. 4Q to 2Q system transformation were adapted to classical DTCNNs in [Brea Sánchez, 2002]. 1Q architecture was analyzed in general for both, discrete and continuous time, in [Brea et al., 2004a]. In [Brea et al., 2005b] and [Brea et al., 2005c] the reduced programmability 1Q-1bit architecture is taken in order to reach significant improvements in area, processing velocity and power consumption. Coherently, DTCNNs inherit the same limitations given by quadrants reduction. Nevertheless, in this case, binary outputs are part of the starting point.

1.5 CPA and CNN Implementations

CPA implementations are realized both over general purpose platforms (CPA emulation), and over specific or reconfigurable hardware. The first option allows more flexibility in the implementation but it is less efficient than the massive parallel computation offered by specific hardware implementations. The latter option is, consequently, the one chosen nowadays for final implementations, especially in real time and/or portable applications. Nevertheless, there are very competitive emulated implementations that should also be considered. In this review we focus on CPA architectures developed for the performance improvement of the low-level image processing, where 2D-CNN implementations have a significant contribution.

The first implemented CNN circuits lacked programmability, being devoted to the application of just one weighting template. The earliest realization we have found in literature, [Cruz and Chua, 1991], implemented a typical connected component detection (CCD) operation. Since that, different proposals were shaping the implementation-oriented simplifications of the original model: time discretization [Harrer et al., 1992], high gain [Espejo, 1994], full range [Espejo et al., 1994, Espejo, 1994] or positive range [Anguita et al., 1996], confirming the hardware and performance improvements expected. The work in reference [Espejo et al., 1994, Espejo, 1994] already included photo-sensors for the direct focal plane image capture and [Espejo, 1994] gave the first steps towards programmability.

We have chosen some representative implementations to illustrate the evolution and the state of the art of the CPA for image processing implementations. We have divided them in ASICs (Application-Specific Integrated Circuits), implementations over reconfigurable hardware (basically Field-Programmable Gate Array -FPGA-), software implementations over commercial parallel processors, and novel technologies including 3D architectures and nanotechnology.

1.5.1 ASIC Implementations

Specific implementations are typically mixed-signal circuits that have as basis a matrix of analog processing elements with extensions for local operations and a digital control system. Analog nature of the processing matrix allows the integration of photo-sensors within the same processing element without the need of A/D converters, eliminating the bottleneck of image transfer. The implementation of a distributed processor with a pixel-PE correspondence and with sensor integration is the basis of the Vision Chips and the horizon of CNN implementations for image processing.

ACE family are general purpose CNNUMs that make use of the FSR CTCNN model. They include programmability and stored-program capabilities and all operative implementations, ACE400 [Domínguez-Castro et al., 1997], ACE4K [Liñán et al., 2002] and ACE16K [Rodríguez-Vázquez et al., 2004], include integrated photo-sensors for focal plane processing. Including D/A and A/D converters, the 128×128 ACE16K gray-scale implementation is prepared to be integrated in a fully digital system [Carranza et al., 2005]. In fact, the ACE16K was integrated in the **Bi-i Vision System** [Zarándy and Rekeczky, 2005] that has been recently used to realize a bionic eyeglass prototype [Karacs and Radvanyi, 2010].

A redesigned version of the ACE16K was employed as the front-end of the first **Eye-RIS Vision Systems** generations. The Eye-RIS family unifies into a single chip the low and high level processing. They are conceived as the core of embedded real time image processing systems that include the whole vision process (sensing - processing - understanding and decision-making) at a high speed. In the last Eye-RIS generations the ACE16K chip is substituted by the QCIF **Q-Eye chip**. The Q-Eye chip significantly differs from ACE16K both at architectural and circuit design level. Mainly, it incorporates a MAC (Multiplier Accumulator Circuit) unit that processes the template application serially, despite of what computation times are similar to those obtained with its predecessors. The Q-Eye improves the chip robustness, the cells density and the power consumption and it even includes new functions in the cells thanks to the area saving given by the MAC utilization instead of the replication of multipliers [Rodríguez-Vázquez et al., 2008]. The Eye-RIS Vision Systems implement actual commercial solutions by Anafocus [AnaFocus].

High gain [Paasio et al., 1996] and positive range [Paasio et al., 1998, Paasio, 1998] output non-linearities led to significant simplifications into a completely binary image processing CNN cell with binary (B/W) images in both inputs and outputs. On this basis, the work in reference [Paasio et al., 1999a] achieves the QCIF standard video format resolution (176×144). Furthermore, [Paasio et al., 2002] proposes several new optimizations starting from a separate implementation of B/W and gray-scale processing cores. For example, gray-scale facilities can be conceived as dedicated while the B/W core is programmable as implemented in [Paasio et al., 2003]. Another proposal is the simplification to templates with 1-bit of programmability. On the one hand, [Laiho et al., 2005] showed that this simplification does not imply any functionality limitation, as any template operating over B/W images can be decomposed in a set of 1-bit programmable templates with a 2-bit programmability bias. On the other hand, this proposal allows significant improvements in B/W implementations [Flak et al., 2006c].

Based on these features it is proposed the **MIPA4k** a mixed-mode 64×64 cell array image processor. This implementation includes image sensors, A/D/A converters, embedded digital and analog memories and hardware optimized gray-scale (5-input order filter and absolute value extraction) and binary processing cores [Poikonen et al., 2009]. In addition it can implement global OR and summation functions, synchronous and asynchronous propagating neighborhood logic operations and space-dependent template and bias operations [Laiho et al., 2009]. Although it does not implement the theoretical universality of the original model, it implements a wide range of low-level image processing operations with improved performance over other more universal implementations.

SCAMP family represents a different approach for massively parallel focal plane image processor. These implementations does not start from the CNN model but share with it defining characteristics as analog processing with digital control and the horizon of vision chip (SIMD paradigm with a pixel to cell correspondence and integrated sensors). SCAMP processing elements are programmable and general purpose. Local connections are limited to the main cardinal points (4-neighbors connections, NEWS) and are non simultaneously accessible but controlled by switches. System evolution is governed by switches configurations [Dudek and Hicks, 2005] that realize the corresponding analog switched-current operations in a discrete-time fashion. Latest implementation SCAMP-3 ([Dudek, 2005]) consists of a 128×128 mesh of general purpose highly optimized digitally programmable PEs. The SCAMP3 vision chip has been successfully integrated in a low power vision system in [Carey et al., 2011].

ASPA family does not follow the CNN model either. It prefers a digital implementation, more robust and more immune to noise, specially important as CMOS technology evolves [Lopich and Dudek, 2011a]. In this case each PE combines a photosensor with an A/D converter. ASPA2 [Lopich and Dudek, 2010] is the latest implementation of this family. It includes 80×80 processing elements in a rectangular grid with NEWS local connections and photo-sensor integration that operates in a SIMD way with a central controller. It supports global operations (OR and summation) by asynchronous binary propagation. A vision system including the ASPA2 vision chip has been presented in [Lopich et al., 2011].

1.5.2 FPGA Implementations

Realizations over FPGA offer shorter time-to-market and lower price than ASIC in exchange for parallelism reduction and no photo-sensor integration, what implies penalizing processing time, power consumption, and form factor or footprint. Although mainly used for fast prototyping, as technology advances this is becoming more and more feasible as a final product option.

Falcon architecture iterates the forward-Euler discretization of the CNN equation under the FSR model to digitally emulate a CNNUM [Nagy and Szolgay, 2003]. This architecture allows accuracy, and template and matrix size reconfigurability. Furthermore, the GAPU implemented over the Xilinx MicroBlaze [Vörösházi et al., 2008] makes it possible to implement complex CNN algorithms making it feasible a low cost programmable CNNUM.

The implementation in [Nieto et al., 2008] proposes a topographic 48×48 FPGA implementation. It is devoted to B/W image processing with an SIMD type computation and NEWS local connections. It is area optimized and it provides a CNN-UM functionality, although it does not implement the CNN model. In this proposal, operations are realized through the combination of Boolean functions. Another general purpose FPGA SIMD for image processing implementation is presented in [Nieto et al., 2009]. In this case, it processes 8-bit gray-scale images by windowing images over 90 PEs. PEs comprise in this case an ALU providing addition, subtraction and multiplication operations apart from the Boolean ones.

1.5.3 Software Implementations

As commercial CPUs and GPUs are improved in terms of parallelism, speed and power consumption, software implementations are becoming an interesting low-cost option for cellular processor arrays (CPAs) in general and CNN in particular.

Cell heterogeneous multi-processor array and **Storm-1** stream processor were chosen, for example, for the implementation of a CNN simulation kernel [Nagy et al., 2007, Furedi and Szolgay, 2009]. In both cases the CNN array is implemented from the Euler-like discretized form of the original equation and the FSR model. The implementations achieve very good performance in the application of linear and, especially, non-linear templates.

We have found as well several CNN emulators/simulators realized over GPUs [Soos et al., 2008, Fernández et al., 2008, Dolan and DeSouza, 2009] that implement a discretized version of the CT-CNN model by making use of the CUDA (Compute Unified Device Architecture) multiprocessor core programming language of NVIDIA. They are intended to provide an accessible and fast CNN algorithm development environment, but they can also deal with simple image processing algorithms offering real-time execution. In reference [Potluri et al., 2011] it is presented a GPU DT-CNN implementation using the **OpenCL** framework as programming language. It makes the applications vendor-independent, and makes it possible to develop the image processing algorithms on multi-core CPUs, on GPUs or on clusters of GPUs. In all these cases, having that GPU is a co-processor, the CPU still executes several tasks like those related to the communication of data with the local memory, for example.

From another perspective, the platform-independent module **APRON** appears as a general CPA fast emulation by making use of the CPU resources. It is intended to support the whole CPA design cycle from the initial conception, modeling and prototyping of the hardware to serving as algorithm development platform, simulator and even hardware interface [Barr and Dudek, 2008]. Furthermore and thanks to its high performance it could be used as a stand-alone array processing system in several applications.

1.5.4 Novel Current Working Lines

The evolution in massively parallel systems requires nowadays new architectural and device features to deal with the technology scaling problems.

CMOS 3D implementation technology has appeared as a good solution for the low *fill factor* (ratio of photosensitive area to the total pixel area) associated to smart sensors. This is due to the processor's placement next to the sensors that at the same time provides the pixel to processor correspondence and the avoidance of the transmission bottleneck. With 3D CMOS technology it is possible to keep the advantages of smart sensors, providing full autonomous Vision-System-on-Chip (VSoC), and reduce its impact to spatial resolution and optical sensitivity. The main idea in 3D technology is splitting the multi-functional feature of the pixel among several stacked layers vertically connected: the upper one is reserved for sensor integration and some others for processing units and memory. In addition, it allows the use of different fabrication technologies for CMOS sensing and processing circuitry to obtain an optimal implementation of both. Two smart sensors prototypes following this approach are presented in [Lopich and Dudek, 2011b] and [Rodríguez-Vázquez et al., 2010].

In a different aspect, as downscaling in CMOS technology advances, undesirable quantum effects appear. At the point where these effects become dominant new devices that make use of the quantum mechanics emerge. They are the so-called **Quantum**

nanodevices. Single-Electron Tunneling (SET) transistors are a good example of these new nanodevices that could take up the baton of CMOS ones even without requiring any new fabrication technology. Another devices in the same line are Resonant Tunneling Diodes (RTD), Carbon Nanotubes (CNT), Memristors, or molecular, ferromagnetic or spin logic devices. At the same time architectures as Quantum Cellular Automata or CNNs reveal as more suitable to combine with these nanodevices in order to avoid the routing downscaling limitations [Flak et al., 2006a]. CNN implementation with SET transistors was analyzed in [Gerousis et al., 2002]. In [Flak et al., 2006b] it was already introduced a neuron structure suitable for CNN implementation in SET technology that could be used to build an extremely dense CNN for B/W image processing. In [Khitun and Wang, 2005], authors introduce a nanoCNN scheme for image processing based on RTDs, and in [Laiho and Lehtonen, 2010] it is suggested a 4-connected CNN implementation using memristors.

1.6 Summary and Conclusions

In this chapter we depict the characteristics of the CPA particularization we will use along this thesis to illustrate our proposals. The Cellular Non-linear Network is a well defined paradigm that has been completed as a universal machine and that has been widely implemented from different approaches and with different optimizations, and that in any case it is considered as a a good option for the implementation of visual processors offering massive parallelism and, still, implementability with its characteristic local connectivity. We will focus on the discrete-time model as our proposals will require well-defined and predictable internal states at any moment. Nevertheless, our proposals could be applied to the B template in a CTCNN, as its application also fits those requirements.

Moreover, although the methodology proposed in this thesis is intended to be applicable to any discrete time hardware realization under the classical CNN system level architecture, we have mainly focused on the B/W implementations with 1Q coefficient circuits and 1-bit of programmability in the coefficient circuits, as they are more restrictive in the kind of techniques applicable (they do not admit gray-scale image feedbacks and require binary template coefficients), and it is more difficult to have significant optimizations at hardware level due to their intrinsic reduced area. We consider this the worst case in the application of our proposal.

Chapter 2

Research Motivation and Related Work

This thesis deals with two interesting challenges in CPA implementations: 1) the realization of large neighborhood kernels while keeping local connectivity and 2) the execution of any-sized kernels (3×3 minimum sized or larger) with a reduced number of local inter-PE connections and weighting circuits, and thus a reduced area compared to conventional solutions. Actually, these two goals can be considered as two aspects of the same objective: the implementations of kernels that overflow the hardware resources, i.e. the number of weighting or coefficient circuits (CC). Both aspects are tackled from the system-level point of view, trying to make the approach applicable to any hardware realization with minimal modifications. In this chapter we give a brief overview of the challenges to be tackled and we review the main works which deal with them.

2.1 Large Neighborhood Challenge. Related work

A CPA is characterized by being a massively parallel system with global processing capacity but local connections. This implies that a PE is physically connected only with its nearest neighbors but it can interact with separated PEs thanks to the propagative effects of the array dynamics of kernel application. The basic characteristic of local connections makes this kind of systems very suitable for its hardware implementation. But, as a consequence, the natural size of the templates to be applied is limited to the smallest one (3×3).

This is, on the other hand, an important limitation in the functionality of a CPA as larger neighborhoods are needed in several image processing primitives as diffusion or low-pass filtering operations [Vilariño, 2001], halftoning [Crounse, 1997], texture analysis [Roska et al., 2000] or matching and hit&miss operations [ter Brugge et al., 1998b], some of them used in algorithms like modern scale- and rotation-invariant feature extractors like Scale Invariant Feature Transform (SIFT) and Speed-Up Robust Features (SURF) [Lowe, 2001, Bay et al., 2008].

As it was previously indicated, a CPA can realize global processing taking into account the whole image information thanks to the propagative effects of the architecture. According to this we can think in solving the remote neighbors interaction through the recursive application of templates. In fact, a recursive process can be summarized in the application of a large neighborhood template. Nevertheless, the inverse process, the decomposition of a LN template into several templates to be applied recursively, is not trivial and different approaches have been developed to deal with this challenge.

The challenge is then to implement any kind of Large Neighborhood (LN) operations while keeping the local connectivity with affordable penalties in performance. Our goal is to do it, in addition, with the minimum impact in the architecture at hardware level. In this section we review the different approaches in the emulation of LN templates found in the literature and we analyze the pros and cons of each one of them. To ease the review we divide the approaches into three groups. The first group gathers those solutions based on the mathematical decomposition of templates. The second group goes through different approaches at hardware level. And finally, the third group is based on the partition of large templates and the shifted-accumulation of the results from applying the minimum size templates obtained. To complete the analysis we add a forth group that gathers other solutions that do not fit in the previous categories.

2.1.1 Template Decomposition Solutions

The most representative work of this kind of approaches is the one presented by Slot in [Ślot, 1994]. In this work the author addresses a set of solutions to have a suitable decomposition of an any large neighborhood template into summations of applications of 3×3 templates. The proposals are based on the associative property of convolution and convolution relationship with correlation (basic operation in CNNs) and are focused on DTCNNs with a piece-wise linear output function.

These solutions can be applied to a generic $N \times N$ template that is recursively decomposed in two $(N-2) \times (N-2)$ sub-templates up to have 3×3 sub-templates. The recursive nature of the approach is at the same time its main drawback as it leads to a number of operations growing exponentially with the neighborhood order as a worst-case upper-bound [ter Brugge et al., 1998c]. Likewise, the computational cost of obtaining the adequate sub-templates can limit the utilization of this methodology. Nevertheless these methods can offer good decompositions for small neighborhoods (5×5) and/or symmetric operators.

This proposal has two main disadvantages that we will try to overcome with our approach. On the one hand Slot's approach implies a high pre-application computational cost due to the mathematical decomposition in convolutions required. The complexity of the decomposition can also be a drawback when implementing automatic compilers to execute LN templates. On the other hand, Slot's methods show exponential growths in the number of resultant operations. It is interesting to note that there are cases for which there exists a 3×3 that, applied recursively, emulates the effect of the original larger template, being the number of iterations directly related to the template size. This results in the most efficient general emulation technique and it can be considered that Slot's approach collapses to it in those particular cases.

2.1.2 Hardware Solutions

Within this group there are solutions that implement the so-called LN-CNN implementing the long distance connectivity in some way. For example, the authors in [Akbari-Dilmaghni, 1998] suggest the grouping of the template coefficients by lines and to pre-calculate the possible values of the contribution of each line when applied to a bipolar image with $+1$ or -1 values. The actual image values select the actual contribution to be added to the next line contribution. This proposal, limited in scalability and versatility, requires bipolar image processing and it is more suitable for symmetric templates. On its side, [Paasio et al., 2002] introduces the reduced programmability (1 or 2 bits) in binary image processing, which results in an efficient implementation even with extended, but limited and fixed, second order neighborhood. Also, the authors of [Wu and Chen, 2009] introduce the use of the so-called "propagating connections", extra weighting circuits within the cell that communicate the cell neighbors in horizontal or vertical directions without incrementing the number of connections. With this it is possible to realize diamond-shaped LN templates and approximate 5×5 templates with some restrictions: the LN diamond-shaped template has to be a diffusing template (the coefficient values have to decrease as the diamond neighborhood order increases) and a coefficient in an inferior layer cannot be zero if the following layer coefficient is not zero. The circuitry proposed can be shared between A and B templates to save area.

In a complete different line there are proposals of modification at device level. For example, the POAC (Programmable Optical Array/Analogic Computer) introduces an optical processing system that provides improvements at processing time, parallelism, image resolution (array size), and template size ($> 100 \times 100$), allowing the realization of feedforward operations at the speed of light. This implementation is combined with a CNN-UM chip with optical input that performs the computations required for feedback and complex algorithms [Tökés et al., 2000, Ayoub and Tökés, 2003, Ayoub et al., 2004]. Another example is the implementation realized in [Yen and Wu, 1999, Wu and Yen, 2001]. With the neuron BJT (νBJT), they take advantage of the diffusion transport mechanism providing reduced chip area and easy photo-sensor integration as well as large neighborhood connections [Chen and Wu, 2004] in a 4-connectivity. Quantum-dot Cellular Automatas are used for cell implementation with connection through the Coulomb law in [Chen and Wu, 2001], where the large neighborhood interaction is implemented in a natural way. In a more classical CNN way, authors in [Laiho and Lehtonen, 2010] suggest a 4-connected (NEWS) implementation using memristors, nano-scale resistive memories of resistance programmable by passing charge through the device, as CNN weights. In this case LN operations can be implemented at the only cost of increasing the programming time with a very reduced area occupation. Unfortunately, CMOS/memristor hybrids are still just a promising solution, but they are not commercially available yet.

The main disadvantage of this group of proposals is the dependence on the hardware particularization. Our goals in the realization of large neighborhood communications are not impose special constraints over the characteristics of the LN kernel and to keep the hardware modifications to a minimum. Of course, the area improvement in absolute terms would depend on the original size of the realization in comparison with the area occupied by the possible additional hardware required.

2.1.3 Template Partition Solutions

The basic technique that brings together the solutions in this group is the direct partition of the original template and the utilization of shifts to correctly accumulate the partial outputs from applying the resultant fragments. We have found two main references in the literature in this line, namely, [Crounse, 1997] and [ter Brugge et al., 1998c].

In the first reference, [Crounse, 1997], the author starts from the associative and distributive properties of convolution to establish a general partition-shift algorithm to implement any-neighborhood templates, although the proposal presented is limited to rectangular templates of dimensions multiple of 3. This approach is focused on B-templates into CTCNNs, as it requires to know precisely the outputs at every moment. In this case, the partition phase groups spatially the template coefficients in 3×3 templates, starting from the coefficients around the center of the original template. The shifting is realized through convolutions over the partially-weighted images. The number of operations grows following approximately a cubic law in the order of neighborhood. Although not analyzed it is also commented the possibility of sharing shifts to reduce the number of operations required, and the possibility of shifting the input image before applying the sub-templates instead of shifting the weighted images.

An interesting contribution of this work is that it presents a canonical form based on the convolution properties that gathers the so-called partition-shift algorithm and the decomposition approach of Slot [Ślot, 1994]. Nonetheless, the kind of templates provided and the way of applying them sets a significant difference in the applicability of the different solutions. The solution in [Crounse, 1997] is considered more useful when implementing LN templates that are sparse, asymmetric or have a complicated non-zero support.

The second work, [ter Brugge et al., 1998c], starts from mathematical morphology, stating its correspondence with DTCNNs operations, and the associative and distributive properties of the dilation and erosion operations. The basic idea is to decompose the large structural element (SE) of the morphological operation into the union of several shifted minimum-sized sub-elements of 3×3 pixels. This implies that the application of each sub-element can be independent of the rest (distributive property) and so realized by means of minimum size DTCNN templates. This analysis has two main limitations. On the one hand, it is limited to those DTCNN operations that are straight derivatives of morphological functions [ter Brugge et al., 1998a], and it limits the image processing to binary images and templates or implies the utilization of non-linear templates (gray-scale morphology, [Zarándy et al., 1996]). On the other hand, as in [Crounse, 1997], it is limited to templates with dimensions multiple of 3. Other methods of fragmentation devoted to obtain an optimal fragmentation ([ter Brugge et al., 1998c], [ter Brugge et al., 2004]) do not have this restriction, but are strongly based on the structural element shape and are restricted to certain characteristics of it. The shifts are referred to the sub-elements, although we understand that in the actual application they would be applied over the initial image or the partially-weighted image as in [Crounse, 1997], what is coherent within the description thanks to the dilation property that implies the equivalence of shifting the SE with shifting the image to be dilated or the dilated image. As in [Crounse, 1997], the number of operations grows with a cubic order in the template size. This growing is improved by the sharing of the shifts, also mentioned in [Crounse, 1997], leading to a quadratic law.

Another reference to a kind of partition and shift method is found in [Koskinen et al., 2004]. In this case it starts from splitting templates and shifting results to implement a 9×9 ($n = 4$) template with a certain hardware modification. The novelty of this proposal is that it gathers the contribution of the 3×3 sub-templates in their correspondent central cell and shifts the results by means of physical connections between the central cell in the original template and the sub-templates central cell. This proposal lies midway between a physical implementation of the large neighborhood and the partition and shift concept. The main limitation of this proposal is that it is no more effective in large neighborhood orders because of the hardware complexity that implies.

The methodology and techniques presented in this thesis belongs to this category of approaches and can be considered together with [Crounse, 1997] and [ter Brugge et al., 1998c] ones. In our case we set a simpler scientific foundation (the associative property of addition) and we develop particular techniques for each phase extending the methodology applicability to templates of any size in both G/S and B/W realizations.

2.1.4 Other Solutions

Within the oldest approaches for the LN template emulation we find [Akbari-Dilmaghni and Taylor, 1996]. This work is based on the assumption that there is a 100% correlation between adjoining pixels from different neighboring orders in the input image. Under this supposition the pixel value is extracted as a common factor in the weighting equation, making the large neighborhood template to collapse into a 3×3 template. The possible error introduced by the initial assumption grows with the neighborhood and it is corrected through a space-variant polarization map I_{ij}. The work is set for CTCNNs with bipolar $[-1, 1]$ images. The still complicated application of this approach together with the requirement of polarization maps are the main disadvantages of this approach.

Finally, in [Geese and Dudek, 2010] Geese proposes the use of Marching Pixels to realize operations requiring long distance data transfer in SIMD. Although the proposal does not refer specifically to LN operations, these could be thought as a possible application of the same technique. This approach consists of the movement and switching of pixel values in the grid and is based on the use of memory to save the pixel and desired positions and in the use of a checkerboard mask. The use of an extra register makes them affordable in number of steps. The main disadvantages are the required usage of memory and the requirement of a checkerboard mask. Nevertheless it is just an application hypothesis as it was not applied to LN operations yet.

2.2 Area Reduction Challenge. Related Work

The CPA characteristic of massive parallelism represents a great contribution to image processing but it is, at the same time, one of the main challenges in its physical implementation. In fact, on focal-plane processors with a pixel-to-PE/processor assignment, the area occupation is not only a matter of cost or hardware size but it affects the image resolution and, in the classical 2D implementations, the pixel fill-factor. The second challenge to be tackled is the reduction of the area occupied by the PEs in order to implement a significant number on a single chip to actually exploit the inherent parallelism of the CNN paradigm.

The different ideas found in the CNN literature are mainly devoted to the weighting or coefficient circuits, either reducing their size or their number, as they represent an important part of the PE area. Of course, the most significant results would come from the combination of both possibilities.

Leaving aside pure circuit improvements, the most important approaches at system-level within the hardware simplification option are the transitions to systems limited in sign (from 4Q systems to 2Q or even to 1Q [Hegt et al., 1998, Brea et al., 2004a]) and value (binary images and kernels with 1 bit of programmability [Paasio et al., 2004]).

On the reduction of the number of coefficient circuits we have different contributions that come from the simple analysis of operations (in reference [Paasio et al., 1999b] authors reduce from 18 to 10 coefficient circuits at the sight of most common operations), to time multiplexing between A and B kernels (just one complete 3×3 template physically implemented -9 CC- as in [Paasio et al., 2002]) or multiplexing in time the application of the elements of the kernel over a single coefficient circuit implementation ([Sargeni et al., 2005]), both for DTCNNs. Within this line we have as well solutions that emulate an 8-connectivity with a 4-connectivity as that shown in [Ślot, 1994] or the referred in [ter Brugge et al., 1998c].

Supporting this second line, the work in [Dudek, 2004] discusses the inefficiency of 9 coefficients implemented per template and it suggests that a drop in the number of coefficient circuits might lead to a better performance, i.e. to a cell with better figures of merit. Following this, we look for a reduction in the number of CC required without drawbacks at functional level, allowing even the LN kernels implementation.

2.3 Summary and Conclusions

In this chapter we have gone through two challenges in CPA visual processing solutions: the implementation of operations implying large neighborhood interaction and the reduction of the area occupation. Both challenges can be seen as two aspects of the same one: the application of templates that overflow the hardware resources, seen these as the coefficient circuits physically implemented. We have also reviewed related work and previous solutions found in the literature for both challenges.

The contributions about LN implementation found in the literature go from different ways of physically implementing the long distance connections, including new devices, to mathematically decomposition or the straightforward breaking of the template. Our goals in the realization of large neighborhood communications are do not impose special constraints over the characteristics of the LN kernel and keep the hardware modifications to a minimum. In so doing, we look for making the proposal applicable to any kind of linear template and adaptable to any hardware particularization with minimum limitations. Taking this into account, we have headed our efforts to algorithmic or system-level general solutions. We look, as well, for the applicability of the approach to both B/W and G/S processing. This implies that approaches as the simple template recursion, when possible, or even the template decomposition of Slot, cannot be taken as solutions as they require G/S image feedback, not applicable in architectures that only deal with B/W images like [Paasio et al., 2004]. Template partition solutions fit this requirement when considering image shifting as the minimum sized templates are applied independently and the partial output is not fed back. Furthermore, these techniques imply smaller performance drawbacks than Slot's decomposition in G/S processing for templates of neighborhood larger than two. The complexity of obtaining the 3×3 templates is another advantage of the partition and shift techniques as they only require spatial grouping of coefficients from the original template. The simple template recursion is, whenever possible, the best performance option but, unfortunately, it is not generally applicable.

The second of the challenges is the reduction of the area occupation by acting over the weighting circuits. Apart from pure circuit improvements, we have found approaches within two lines in the literature: the circuit optimization by means of system-level restrictions and the reduction of the number of coefficient circuits. The first line leads to B/W implementations with even reduced programmability and the second line leads, in some cases, to a reduction in the functionality. Both lines are combined to obtain larger area reductions. Our goal in area reduction looks for no penalties at functional level in being applied to both B/W and G/S realizations. Although our proposal goes through the reduction of the number of coefficient circuits, we also consider the possibility of combination with demanding system-level restrictions leading to B/W implementations.

Chapter 3

Split and Shift Methodology

In this chapter we develop the Split and Shift (S&S) methodology, our proposal for dealing with templates that overflow the weighting circuits availability on a CPA, which includes long distance communications in the application of large neighborhood (LN) kernels, and the application of any-size kernels over a reduced connectivity CPA implementation. In the methodology development we set guidelines and propose techniques within an in-depth and rigorous analysis of their implications at hardware and processing time level. For the assessment in the area occupation reduction we have defined a Figure of Merit (FoM) to evaluate the benefit-penalty trade-off (area reduction vs. processing time increment) and we have used it to choose the most adequate techniques.

Although the same methodology deals with both challenges (LN and reduced connectivity), the techniques and required analysis are different in each case, and they are treated separately after the introduction of the general lines of the methodology. The combination of both challenges are thoroughly analyzed at the end of the chapter.

3.1 S&S Methodology General Lines

Split and Shift (S&S) is the name we give to the partition and shift methodology that we have developed to allow the application of templates that require more weighting or coefficient circuits (CC) than those available on a particular CPA implementation. We have focused, then, on dropping the number of required inter-PE connections along with their corresponding CCs, however the template dimension, 3×3 or larger neighborhood ones.

In short, our methodology is based on the DTCNN state equation (Eq. 1.5) seen as a summation of products (Eq. 3.1). With this, the associative property of addition can be applied and the equation can be re-written into several sub-additions. This, in turn, suggests splitting large neighborhood or 3×3 templates into smaller ones by grouping the coefficients spatially. These sub-templates are applied separately as an associated group of template coefficients. The partial outputs are then summed to complete the original template application, obtaining the new state *x(T+1)* from which the output is calculated. The result is exactly the same as that of applying the original template over a full-coefficient-circuit implementation. Although illustrated for DTCNNs, the S&S methodology is applicable to CPA architectures in general with the only requirement of having accessible, predictable and stable states at every clock cycle, i.e. it can be applied to synchronous deterministic implementations, including the control template B in a classical Continuous Time CNN implementation. In addition, although DTCNN definition is originally completed with a threshold output function (Eq. 1.4) that restricts the outputs to binary values, similarly to the expected saturated outputs in the continuous time version, this is not a S&S requirement, and different output functions allowing G/S outputs can also be possible.

$$\begin{aligned} x_{ij}\,(T+1) \;&= a_{ijkl}\; y_{kl}(T) \;+ a_{ijpq}\; y_{pq} \;+ a_{ijrs}\; y_{rs}(T) \;+ \ldots \;+ \\ &+ b_{ijkl}\; u_{kl}(T) \;+ b_{ijpq}\; u_{pq}(T) \;+ b_{ijrs}\; u_{rs}(T) \;+ \ldots \;+ i_{ij} \end{aligned} \tag{3.1}$$

Based on the interchangeability of A and B templates in DTCNNs, we generally start our analysis from the consideration of an 8-connected kernel of 9 elements, and not from the whole CNN classical operation comprising two, A and B, templates. Furthermore, this one-kernel is a more usual case on a CPA executing classical low-level image processing operators. The two-template CNN case is considered as a particular extension, where the two templates are executed in parallel if the required hardware is available, or run successively to combine their results in other cases. As it will be shown, the physical availability of hardware for two templates widens the possibilities of the proposed techniques. The bias term does not have influence in the application of the S&S techniques as it can be added at any moment prior to the application of the output function. 4-connected patterns are also considered as a particular case within the hardware reduction part.

The S&S methodology comprises two phases that can be understood as a preparation phase and an application phase. The first one, *Split Phase*, consists of grouping the coefficients that compound a template into several minimum-sized 3×3 sub-templates, either full dense, or sparse if considering a reduced connectivity pattern (reduction of the CC), always respecting their original relative positions. That is, we "split" the original $(2n+1) \times (2n+1)$ template (with n being the neighborhood order, an integer number greater or equal one) into several sub-templates. This phase has to take into account the final resources availability in order to adapt the new sub-templates to them. The second phase (*Shift Phase*) comprises the appropriate application of the resulting sub-templates and the collection, by means of shifts, of their outcomes at the central cell of the original $(2n+1) \times (2n+1)$ neighborhood. The $(2n+1) \times (2n+1)$ original template response is then approached by the combination of two types of minimum-sized kernels or templates:

- *Decomposition templates or sub-templates*, obtained from spatially grouping the coefficients of the original $(2n+1) \times (2n+1)$ template.
- *Shift templates*, needed to have the contributions gathered by the sub-templates at the central cell of the $(2n+1) \times (2n+1)$ window to be accumulated.

The methodology has two variants depending on the order of application of these templates. If we first apply a sub-template the weighting result is obtained in general in a cell different from the original central one and it has to be shifted to it. On the other

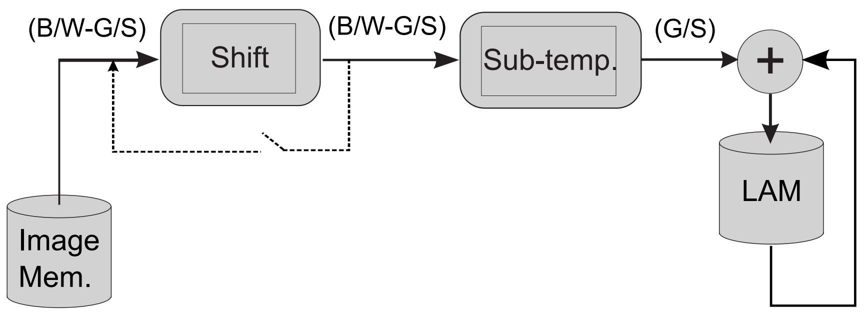


Figure 3.1: System-level architecture for the application of S&S methodology in image shifting mode.

hand, if we adequately shift the original image prior to a sub-template application, we have the partial results directly in the referred central cell. In so doing, we have two different system-level architectures and application algorithms:

- *Partial result shifting mode.* It is the most straightforward approach. In this variant the sub-templates are applied over the original image, and the results have to be shifted from the cell where they have been obtained to the central cell to be accumulated. We also refer to it as the *fixed image mode.*

- *Image shifting mode.* This variant shifts the image to be weighted to make coincide the large neighborhood template center with the center of the sub-template to be applied. The partial output is directly obtained at the cell of interest, where it is accumulated. This variant has the advantage of not requiring G/S feedback when working with B/W images. We also refer to this variant as the *shifted image mode.*

Additionally, we can consider the possibility of sharing shifts in order to reduce the number of operations. In the fixed image mode, to share shifts implies that the partial outputs are gathered on their way to the central cell, being added to the next sub-template partial result at the cell where the latter is obtained. In the shifted image mode it means that new shifts are applied to the previously shifted image, without the need to keep the original image if we always use shift-sharing.

The system-level architecture for the image shifting mode is shown in Fig. 3.1. The image (original or shifted) is taken from a locally distributed memory (either analog - Local Analog Memory, LAM - or logic - Local Logic Memory, LLM) and it is shifted. Afterwards, if no other shift is required, the sub-template is applied over the shifted image. The internal state (the partial result is taken before the output function application) is accumulated in a LAM as a gray-scale value. The process is repeated until all the sub-templates have been run and all their contributions are gathered in the LAM. At that moment, the value in the LAM is exactly the same as the internal state that would be provided by the original template application. The last step will be the application of the output function to this value.

The system-level architecture for the fixed image mode (Fig. 3.2) interchanges the order of application of the two kinds of operations. First, the sub-template is applied and then the partial output is shifted to the LN template central cell if we opt for the

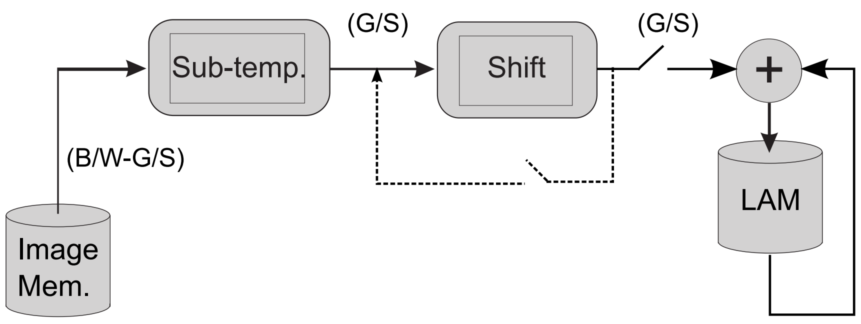


Figure 3.2: System-level architecture for the application of S&S methodology in partial result shifting mode.

non-sharing option, or to the central cell of the next sub-template to be run, if we decide to share the shifts. The accumulation is realized in the LN template central cell in the first case and in the subsequent centers of the sub-templates in the second one. This fixed image mode demands the feedback of gray-scale images to be shifted, the partial outcomes to be accumulated, and could be affected by internal state range restrictions that should have to be tackled. In both, image shifting and result shifting modes, the combination of shift-sharing and no-shift-sharing will require the availability of one or two more memories.

3.2 S&S for LN Template Emulation

After the general introduction of the methodology, in this section we tackle the application of the methodology in the emulation of large neighborhood templates by the application of minimum-sized templates (3×3). The objective is to improve the functionality of locally connected implementations with minimum penalty at hardware or processing time level.

In Section II of the CNNA05 paper (Appendix A, page 99) we introduce the S&S methodology through its particularization for 5×5 templates to ease the understanding of the process. Section III of the same paper and Section II (erroneously named "Large-neighborhood splitting methods" instead of "Large-neighborhood S&S methods") in the DCIS05 one (Appendix A, page 105) refer the methodology application for a general $(2n+1) \times (2n+1)$ template. Both papers depict the system-level architecture for the image shifting mode, what is the only option for the binary implementation they consider, with some differences if compared to Fig. 3.1 shown above. Fig.2 in CNNA05 paper represents the system-level architecture limited to a 5×5 template realization. Fig.1 in DCIS05 paper redraws the architecture including recursive shifts, needed for larger templates (more than one shift step is required per sub-template) and for shift-sharing. According to the binary implementation, in both papers the image (original or shifted) is taken from an LLM and shifted. In addition, in those figures we show shifts as complete CNN operations including the output function application. Actually, this step is not necessary in the shifting operations and it is avoided in the sub-template application. Whether or not it is applied would depend on the particular circuit realization.

Splitting Techniques

To choose the adequate technique for the splitting is critical for the final number of operations (both shifts and sub-templates). If we have templates with a size multiple of 3×3, the spatial grouping and split is straightforward and we will have sparser or denser templates in function of the value of the coefficients in the original template. Nevertheless, with a different template size (e.g. 7×7 or 11×11), we obtain incomplete 3×3 templates during the split phase that have to be completed with zeros. We have several options for grouping the template coefficients, leading to different number of sub-templates and shifts in the following phase, hence different time performances.

We have observed that, as a general rule, we reach the minimum number of sub-templates with a regular grouping process starting from the template corners, against the intuitive thought of beginning from the central sub-window adopted in Crounse [1997] or ter Brugge et al. [1998c], which were devised for templates of sizes multiples of a 3×3 neighborhood. Otherwise, corner coefficients might be left isolated, yielding more sub-templates. In addition, incomplete sub-template overlapping reduces the number of shifts as the sub-template centers are moved closer to the LN template central cell. Overlapped template elements are substituted by zeros that can be distributed within the neighboring sub-templates to make the power consumption more homogeneous in the sub-template application (assuming that zero coefficients imply lower power consumption).

These issues are illustrated for a generic 5×5 template in the second section of the (CNNA05) paper (p.99). Clearly, in a 5×5 neighborhood the minimum number of 3×3 sub-windows is four and it comes out from corner starting (see Fig.1 in this paper). The sub-templates centers are chosen taking into account the sub-template overlapping option as the distance between sub-templates and template centers are clearly shorter with it. Overlapped template elements made null are shown in Fig.3 in the paper. Sub-template overlapping is introduced there as an interesting way of obtaining a more robust template by reducing the number of non-null template elements. Nevertheless, this statement is not completely true as the considered robustness definition is applied over complete CNN operations, i.e. including the output function application. In our case the partial output provided by the sub-templates application have to be summed before applying the output function and so we cannot extract any conclusion from that definition.

With these guidelines we propose three split techniques that are shown over a 13×13 template in Fig. 3.3: concentric (Fig. 3.3.a), by rows (Fig. 3.3.b), and recursive (Fig. 3.3.c). As 13×13 is not a multiple of 3×3 the splitting results in incomplete sub-templates that have to be completed with zeros. In the image we have grouped the coefficients over the original template, and we have marked the groups with thick lines. We have also marked with dashed thick lines the starting groupings. The first two techniques directly group the template coefficients in minimum-sized templates starting in the four corners in case a), and in the upper-left one, for example, in case b). They produce the same number of sub-templates, which is given by Eq. 3.2, where n is the order of neighborhood and we assume squared templates of size $(2n+1)\times(2n+1)$. The ceiling function $\lceil\ \rceil$ produces the smallest upper integer of its argument.

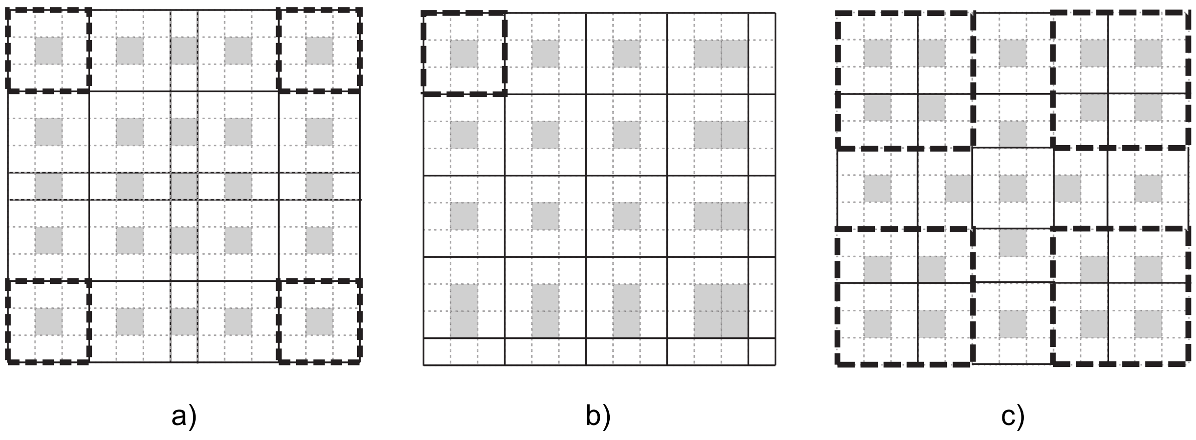


Figure 3.3: Different splitting methods over a 13×13 template. a) Corner starting and concentric decomposition. b) Corner starting and by rows process. c) Corner starting with recursive decomposition. Starting groupings in dashed thick lines. Sub-templates centers shadowed.

$$D(n) = \left\lceil \frac{2n+1}{3} \right\rceil^2 \tag{3.2}$$

The third technique implies recursive decomposition into four main sub-templates that are sub-sequentially divided until achieving 3×3 ones. Fig. 3.3.c shows the four 5×5 starting sub-templates in dashed thick lines, that are afterwards sub-divided in already minimum-sized sub-templates, mostly incomplete in this case. Due to the recursion, it provides larger or equal number of incomplete sub-templates and, consequently, larger or equal number of total sub-templates than the first two techniques.

As a rule of thumb, we will have less number of decomposition templates if 1) we choose non-recursive, i.e. direct 3×3, splitting; 2) if we start the splitting from the corners; and 3) if we follow a continuous ordered process by rows or in a concentric way. Nevertheless, the recursive technique could render less shift operations due to the natural spatial result gathering if we decide to combine CNN and hardware shifting applying the proposal of Koskinen et al. [2004] for the final shifts. Still, provided that this technique implies a more complex decomposition and it would be interesting only with hardware specifically dedicated to shifting operations, we will center the study over the two first techniques. Finally, the centers of the incomplete sub-templates, and thus the allocation of coefficients in them, will be selected with a view to having a minimum number of shifts in the partial-outputs path to the LN template central cell. This will depend on the shifting technique.

Shifting Techniques

The minimum number of shift operations (i.e. the number of shifting templates) relies 1) on the window-split method that determines the position of the sub-templates, and the distance between their centers and the LN template center, and 2) on the shifting technique chosen, that determines the shifting path/s. In the incomplete sub-templates the center is not fully determined by the splitting technique and can be chosen in the most favorable position according to the shifting technique. The shift-sharing avoids redundant shifts, diminishing the actual number of operations and, thus, the computation time.

Neither the number of shifting operations, and clearly nor the number of sub-templates, depend on the S&S mode chosen, image or partial-result shifting. Nevertheless, it is important to take into account that the choice implies a different order in the application of the sub-templates when applying shift-sharing. In the case of partial result shifting with shift-sharing we start from the outer part of the LN template. We apply an outer sub-template and we shift the result to the central cell of the next sub-template (i.e. the cell where we are going to obtain the next sub-template contribution) to pick up the new partial result, and so on until reaching the LN template central cell. In this case, shift-sharing can imply keeping partial result accumulations to wait for partial accumulations of different shifting paths that converge in the same path to the LN central cell. In the case of image-shifting we start shifting the image to make the central pixel of an inner sub-template coincide with the LN template central cell. The result of applying the sub-template is calculated, then, directly in the LN template central cell where we will accumulate all the partial results. The next shifts are applied over the shifted or the original image as convenient to yield the minimum number of shifting operations. According to the shifting technique, shift-sharing can imply to keep different shifted versions of the image when the route to the LN template central cell is divided in branches.

Regarding hardware implications, the number of memories required is the same for image and result shifting S&S modes. The difference lies in the type of memories required: in general we would require analog or digital memories with several bits but they could be 1-bit memories for image shifting if we have binary images except for the S&S partial results accumulation memory. This is coherent with the processing type required in each case.

Taking into account these considerations we have developed the shifting techniques. Fig. 3.4 displays the three main techniques. They are shown over a 13×13 template with a corner starting concentric way splitting. We have chosen this splitting option because it is more symmetric around the central cell, which can improve the technique homogeneity and, depending on the chosen shifting technique, even reduce the number of shifts. We consider shift-sharing in all of the selected proposals. Of course, not sharing shifts is also an option, but it implies a larger increment in the number of operations, more important as the neighborhood order increases. In that case each partial result is independently shifted to the LN template central cell in output shifting, and the image is shifted for the application of each sub-template starting from the original image in image shifting.

In Fig. 3.4 the arrow heads mark the beginning of the shifting path along the sub-templates centers (shadowed) over the 13×13 region covered by the original template. In the case of image-shifting the image is shifted to apply the sub-template over the correct part of the image and provide the result directly at the cell of interest. The arrow heads mark in this case the last pixel shifted to the central cell on a given route. In the case of result shifting the sub-template is applied over the original image and the

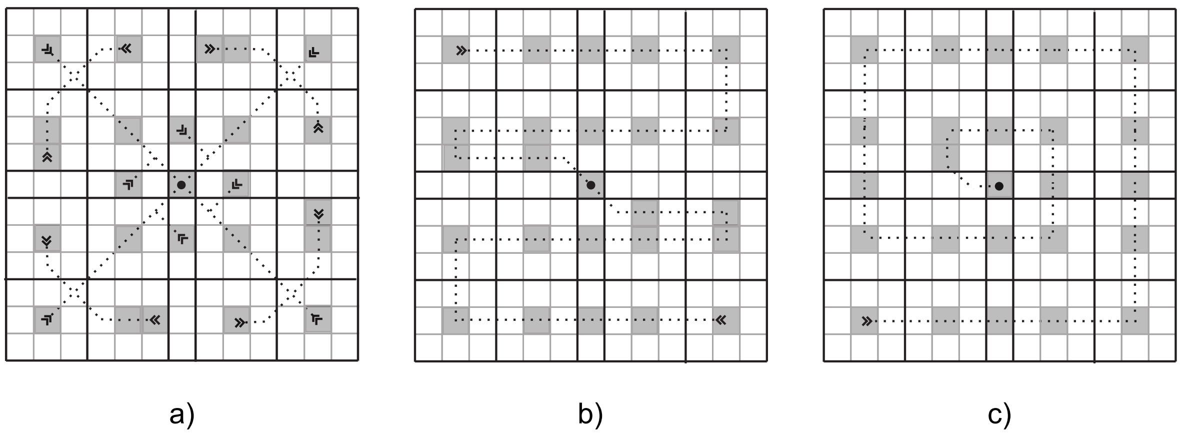


Figure 3.4: a) Central shifting. b) Zig-zag shifting. c) Spiral shifting. Sub-templates centers shadowed. Shifting path beginnings marked by the arrow heads.

result is obtained at the cell that corresponds to the center of the sub-template. The result is afterwards shifted to the cell of interest following the shifting path through the rest of the template centers and gathering the rest of the partial results in the path. All the paths showed consider shift-sharing.

In the central shift technique (Fig. 3.4.a) we have to re-start the shifting several times, i.e. taking several times the original image in image shifting or realizing several partial accumulations in result shifting. In both cases this technique implies more usage of memories. In the case of the zig-zag technique (Fig. 3.4.b) we have a two-starting-point process, which means to start again from the original image to apply the second half of templates or to accumulate the partial results in two parts. As shown in Fig.5.b in CNNA05 paper (p. 99), zig-zag could be realized as a continuous process with only one starting point with some more shift operations, those required to start with a corner template application in image shifting or to shift the final result accumulated from a corner, but with less usage of memories. The spiral technique (Fig. 3.4.c) implies a one starting point process, i.e. a continuous result accumulation or consecutive image shifting. One of the advantages of the latter approaches over the central one is the regularity in the template application what makes the process simpler both for manual and automatic application. In addition it has a consequence as well on the number of memories required. In the first case (central) we will need four memories, one for the original image, one for the final result accumulation, and two for the intermediate steps. In the second case (zig-zag) we will need three, one for the intermediate step. In the spiral case we will need just two memories, one for the image (shifted or original) and one for the partial output accumulation and final result. They will be two as well for the zig-zag case if we realize a continuous process instead of starting from two different points.

The number of shifts is given by Eq. (3.3), Eq. (3.4) and Eq. (3.5) for the central, zig-zag and spiral shifting respectively. These equations have been obtained by induction, taking into account three different classes of templates: those that are multiple of 3×3 ($n = 1 + 3i$, being i an integer ≥ 0); those that provide incomplete sub-templates with dimension 2 (2×3, 3×2 or 2×2), being the representative template the 5×5

($n = 2 + 3i$); and those that provide sub-templates with dimension 1 (1×3, 3×1 or 1×1), being the representative template the 7×7 ($n = 3 + 3i$). These different classes present particular situations in the sub-templates center distribution and require corrections in the general equations. These corrections are gathered in Eq. (3.3), Eq. (3.4) and Eq. (3.5) governed by the *remainder* function (*rem*), that provides the remainder of the division contained, and the *floor* ($\lfloor\ \rfloor$) and *ceiling* ($\lceil\ \rceil$) functions, that provide the nearest lower and upper integers of their respective arguments. Eq. (3.3) is corrected for the 5×5 template class; Eq. (3.4) is corrected for the 5×5 template class in the first correction term and for the 7×7 in the second one; and Eq. (3.5) is corrected for both, 5×5 and 7×7 classes, in the same term. Note that Eq. (3.3) is valid for $n > 1$ (i.e. $> 3 \times 3$). In all of the equations we have a quadratic behavior, but with slightly better results for the central technique. Fig. 3.5 shows graphically the behavior of the number of shifts with the neighborhood order for the three techniques. Note that the spiral and the zig-zag techniques result in the same number of shifts in template sizes multiple of 3×3 but spiral technique slightly improve the zig-zag numbers in the rest of the cases. The main advantage of the zig-zag and spiral approaches over the central one is the regularity in the template application, which makes the process simpler both for manual and automatic application. We can improve the spiral results for the 5×5 and 7×7 classes, and the zig-zag results for the 5×5 one if we combine the techniques with the central shifting in the 5×5 and 7×7 resulting central templates in a concentric split. It is shown in Fig. 3.6 for the spiral shifting technique where 5×5 and 7×7 central cores are shadowed and the corresponding sub-templates centers are shown in a darker gray. Arrow heads indicate the beginning of the shifting routes in the template sizes shown. The zig-zag improvement process for the 5×5 class is the same as that illustrated for the spiral technique in Fig. 3.6.a. Nevertheless, these improvements imply more irregularity in the S&S application, what is the main advantage of these two approaches over the central one, with a not very significant lower number of operations.

$$S(n) = \frac{4}{3}(n^2 - n + 3 - 2 \cdot \lfloor \frac{rem\left(\frac{2n+1}{3}\right)}{2} \rfloor) \qquad (n > 1) \tag{3.3}$$

$$S(n) = \frac{4}{3}(n^2 + n - 2 + \frac{1}{2}(n-1) \cdot \lceil \frac{rem\left(\frac{2n+1}{3}\right)}{2} \rceil + (n - \frac{5}{2}) \cdot rem(\frac{rem\left(\frac{2n+1}{3}\right)}{2})) \tag{3.4}$$

$$S(n) = \frac{4}{3}(n^2 + n - 2 + \frac{5}{4} \cdot \lceil \frac{rem\left(\frac{2n+1}{3}\right)}{2} \rceil) \tag{3.5}$$

In the LN emulation generalization presented in Section III of CNNA05 paper (p. 99) we also consider the non-shift-sharing approach (”independent shift approach” in the paper) and it is clearly shown its inefficiency in Fig. 6 of the same paper. The three shifting techniques considered (independent a), zig-zag b), and spiral - named concentric- c)) are shown in Fig. 5 in the paper. Note that the independent technique is

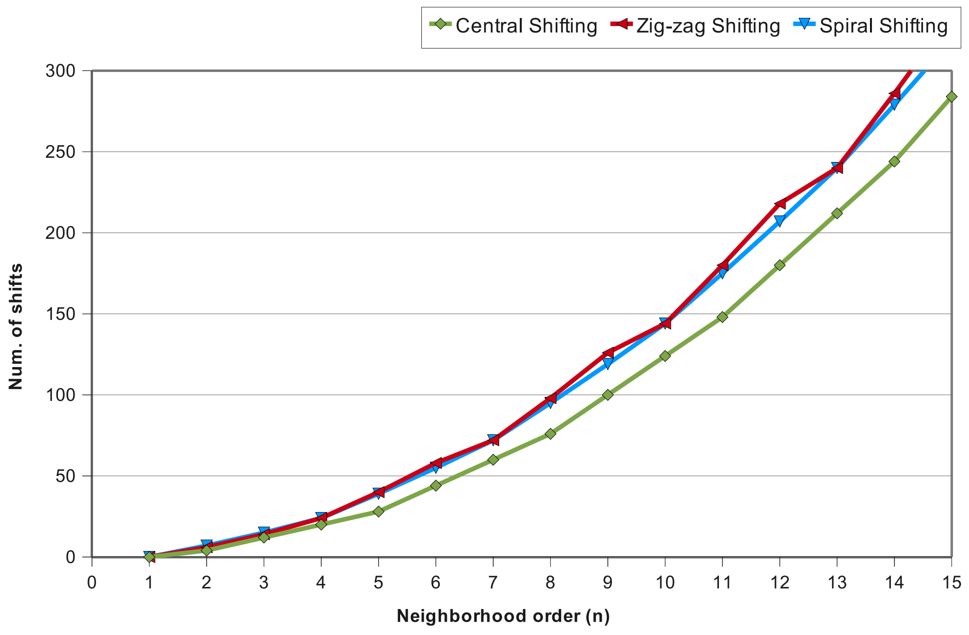


Figure 3.5: Comparative of the number of shifts required for different shifting techniques in function of the neighborhood order.

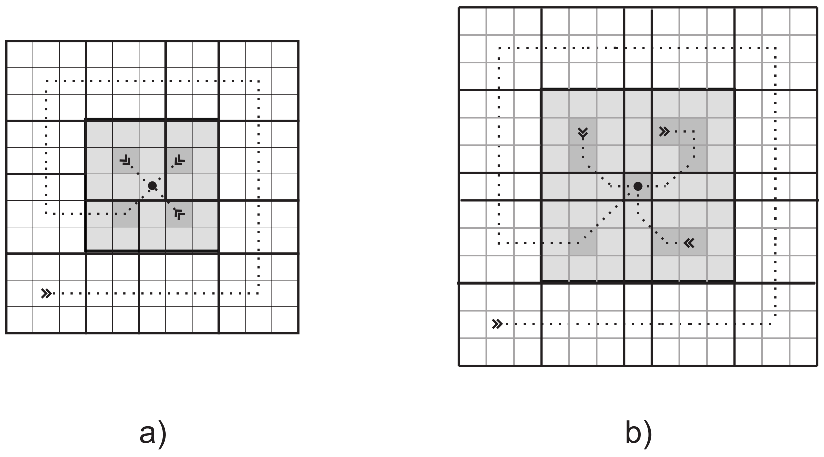


Figure 3.6: a) Shift optimization for a 5 × 5 core. b) Shift optimization for a 7 × 7 core.

a central technique, non-optimized if we apply shift-sharing. The figure corresponding to the spiral technique (”concentric” in the paper) presents an error that is corrected in Fig. 2 of DCIS05 paper (p.105). Fig. 3.4 c) in this section shows a different election of the sub-templates centers that does not affect to the total number of shifts but to the distribution of template elements in the sub-templates affected. The zig-zag technique in CNNA05 paper (p. 99) uses the by-rows split technique just affecting the allocation of sub-template centers and the distribution of template elements. It does not affect the number of operations in the zig-zag technique but it is not adequate for the spiral one. Its regularity could be advantageous in automating when choosing zig-zag shifting technique.

A final interesting comment is that we can apply two operations simultaneously in a DTCNN with a two-template hardware implementation. In the output shifting mode we could apply simultaneously a sub-template and the shift-accumulation of the previous partial result. Similarly, in the image-shifting option we could apply simultaneously two sub-templates over different shifted images, which, on the other hand, would lead to an extra image memory. If we have a one-template operation this combination of two operations in one would reduce significantly the number of operations. We reduce the number of operations in half of the sub-template applications in the case of image-shifting and in almost as many shift operations as sub-templates obtained in the splitting phase for output-shifting. If we have a two-template operation both templates emulation can be applied simultaneously, except for the image shifting in image shifting mode if input and initial state are different.

S&S for LN Emulation: Example of Application

Fig. 3.7 illustrates the whole LN emulation process for a 5×5 diffusion template with the S&S image shifting mode. Original 5×5 template is shown in the upper-left part of the figure. The template is already broken in four groups of elements. We opt to distribute the elements in the groups in a more homogeneous way to balance the power consumption in the sub-template application. The resultant sub-templates are all incomplete and they have to be completed with zeros. We overlap the sub-templates in those new-null positions to reduce the distance between original template central position and sub-templates central positions, what reduces the number of shifts required. The resultant sub-templates (D_i) and the required shifting templates (S_i) are shown in the upper-left part of the figure. In the process representation, shifts are circled and sub-template application squared. Original image and final internal state are framed. Internal states resultant of shifts and sub-templates application and partial accumulations are shown along the sequence. Shifts outputs are shown over a grid to ease the perception of the effect. The final image matches the internal state given by the direct application of the 5×5 template. All these images come from simulations of CNNs of minimum and large neighborhood connectivity in Matlab.

Figures 3 and 4 in CNNA05 paper (p. 99) show another view of this application example. Despite the G/S final output, this example was thought to be implemented with a particular binary implementation with a G/S module that operates with this output Brea et al. [2004b]. The diffusion template is considered, there, homogeneous with all its coefficients set to 1. Here the diffusion template is kept in real numbers

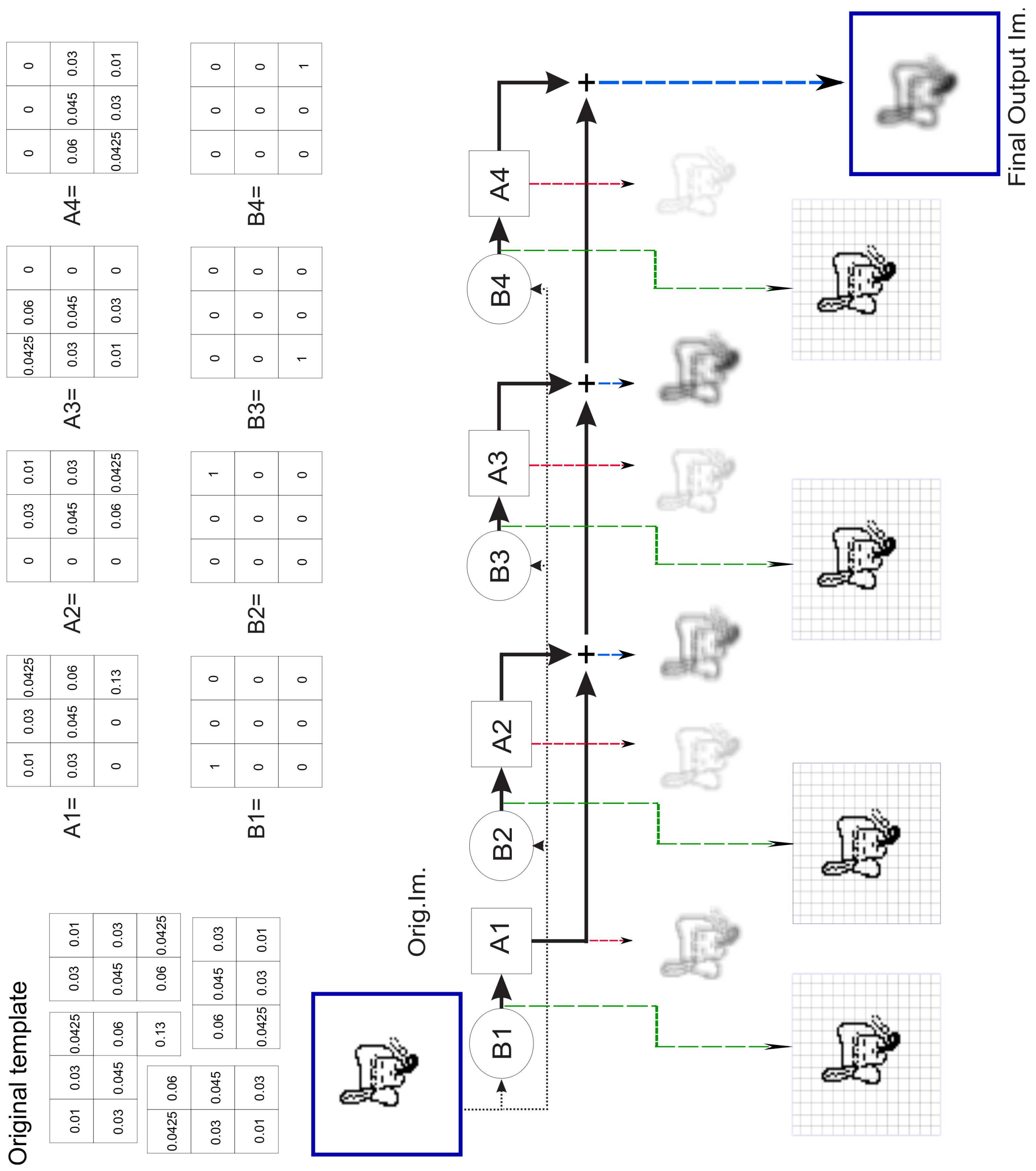


Figure 3.7: Application of a 5×5 low pass filtering template with 3×3 templates and image shifting.

to easier the following of the example. The application of this example is shown in Sections III and IV of DCIS05 paper (p. 105). The extraction of an internal potential that guides the binary contours to smooth their shapes is originally realized through LN diffusion kernels. This kind of kernels are substituted in G/S circuits by the recursive application of minimum-sized kernels. With the S&S methodology they can be applied even in binary implementations with an additional LAM implementation. The effect of the internal potential in smoothing contours is shown in Fig. 6 and 7 in DCIS05 paper. Fig. 8 in that paper illustrates the application of this feature to a robot guiding algorithm.

S&S Techniques Discussion

Fig. 3.8 shows the total number of operations of the S&S techniques considered (central, spiral and zig-zag with concentric splitting) together with the number of operation required by other LN emulation techniques, in particular the Slot's decomposition proposal (Ślot [1994], considering the number of operations given in ter Brugge et al. [1998b]) and the Crounse and Brugge's template partition approaches (Crounse [1997], and ter Brugge et al. [1998c] and ter Brugge et al. [1998b] respectively). We have used an upper bound approximation for the Crounse proposal as the function given in Crounse [1997] is recursive and dependent on the LN template size. For Brugge we have included both shift-sharing and no shift-sharing approaches. Crounse approach is probably closer in number of operations to the no shift-sharing Brugge's approach. Note that we do not include line connection in these three last representations. This indicates that these proposals are restricted to certain neighborhood orders, in particular to those that provide 3×3 multiple templates. We include as well the unit slope function that represents the number of operations required by the simply recursive application of a 3×3 seed template when it exists.

It is apparent that, when possible, the 3×3 seed is the best option. Within the generally applicable decomposition techniques, we observe in this figure the inefficiency of the general Slot's proposal versus the partition techniques. Partition techniques are the only ones that can apply LN template emulation in binary architectures. Within the S&S approaches those that do not consider shift-sharing offer worse results. As it could be expected Brugge's approach coincides with zig-zag and spiral approaches in template sizes multiple of 3×3 . Central technique is the best of the S&S proposal in number of operations but it is irregular.

At Fig. 4 in DCIS05 paper (p. 105) we can see that, under supposition of 50 ns per operation for binary implementations (for S&S) and 1 μs per operation for general gray-scale implementations (for seed recursion), the proposed techniques can be comparable (and even better) than the 3×3 recursive seed technique up to orders of neighborhood of 10 (i.e. 21×21 templates). This is also true up to n=5 if we consider 100 ns per CNN operation in the binary implementations. Finally, we could have up to 400 μs per operation to keep video rate processing (25 $frames/s$) with 100 operations per frame. Considering 1 μs per 3×3 CNN operation we could apply 100 of 31×31 templates. The numbers shown were obtained for the spiral technique.

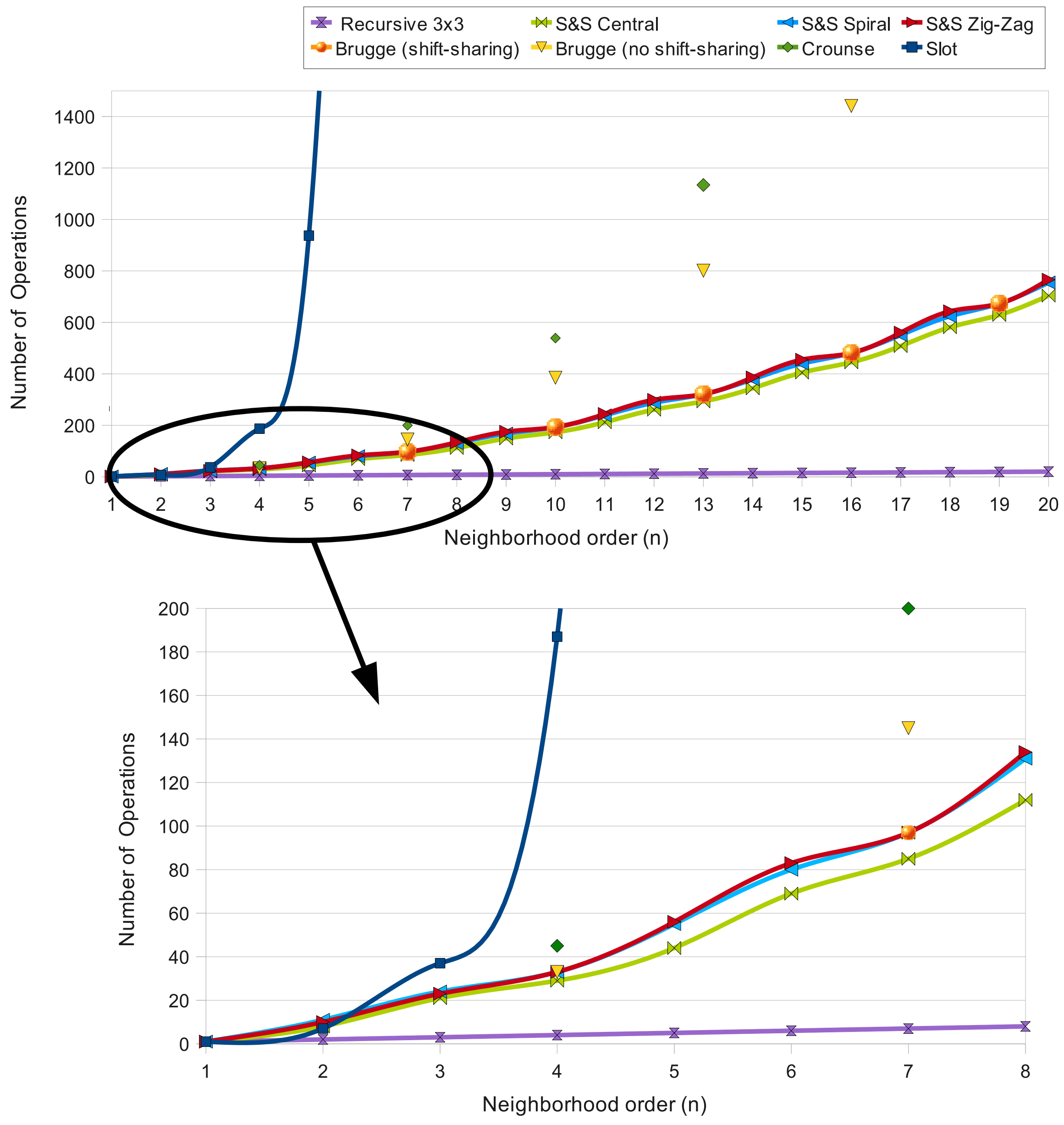


Figure 3.8: Comparative of the number of operations required for different LN emulation techniques in function of the neighborhood order.

3.3 S&S for the Hardware Reduction

In this section we apply the S&S methodology to shrink the PE area by reducing the number of coefficient circuits physically implemented, with no drawbacks at application level and affordable processing time penalty.

S&S Methodology over 3×3 Kernels

Our proposal for hardware reduction consists of the application of the S&S methodology to 3×3 kernels. The basic idea is the same as in the LN emulation: the coefficients in the original template are distributed in several sub-templates that are run together with shifting templates that allow the gathering of the correct contributions of the neighbors at the cell of interest.

In this case the original template is a 3×3 kernel, i.e. the minimum size template, and so the sub-templates are not smaller, they are 3×3 kernels too, but sparser, i.e. with several new null elements. The objective is to re-use the same coefficient circuits and neighbors' connections in the application of all the sub-templates in which we break the original 3×3 template. In so doing, we can remove the non-used coefficient circuits and connections. This idea implies that all the sub-templates have the same non-null elements arrangement. The issue now is how we choose the sub-templates sparse shape.

As we are working at the interface of system and hardware-levels it is important to have in mind both perspectives in the template application, the template perspective shown in Fig.1.5, and the hardware perspective shown in Fig. 1.6. In the system-level convention the cell gathers the weighted contributions of the neighboring cells (Fig.1.5). In the hardware-level convention each cell weights its own value prior to sending it to the neighbors (Fig. 1.6).

In Fig. 3.9 we show the inter-PE connections and the contributions that can be gathered with a particular restriction in the coefficients circuits (remaining CCs marked as dots). From a hardware point of view, the usual implementation is to weight the own value at the cell under study and to distribute adequately the weighted results to the surrounding cells. In dashed arrows (Fig. 3.9) we have the neighbors' contributions that will be gathered at the central cell (template point of view), and with solid arrows we have the contributions of the central cell to its neighbors (hardware point of view). With this configuration the information flows to the right neighbors only. Note that template coefficients allocation is a mirror of the corresponding CC positions. The *cell configuration* will represent the actual connections and remaining coefficient circuits in a cell, i.e. the hardware-level perspective. The importance of the cell configuration election lies in that it sets the neighbors that are connected to the cell of interest, and so the template elements that can be applied, and the shifts that can be realized. This issue is also illustrated in the second section of the CNNA06 paper (Appendix A, page 113).

Split and Shift Phases

Now the split phase consists of grouping template elements in the shape selected, but preserving the original relative positions. There is no limitations imposed to the sparse

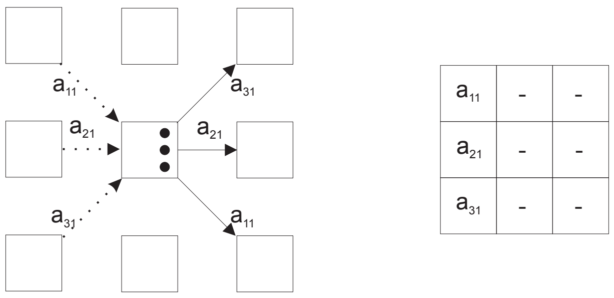


Figure 3.9: Cell communications with a particular reduced set of CC and its correspondence to the template coefficients application.

shape by the split phase as we can correctly place all the elements in any shape with the enough number of sub-templates. On the other hand, the shift phase has to gather all neighbors' contributions at the correct cell. The required shifts are determined by the sub-templates center allocation but they have to be allowed by the cell configuration as it limits the neighbors communication. This imposes a restriction in number and shape in the remaining coefficient circuits to allow all the required shifts. Coherently, our analysis now deals about the implications of choosing different cell configurations. The split and shift techniques are mostly determined by the sparse shape.

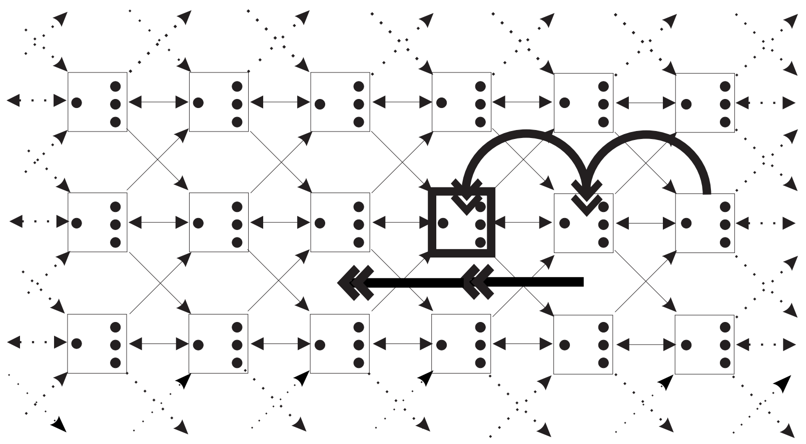
Finally, as in the LN emulation, we have the option of shifting the image to be weighted or the weighted image, and the option of sharing or not sharing the shifts. Of course, the consequences of choosing one or another option are as well the same as in the LN emulation: result shifting implies G/S processing and memories, shift-sharing can imply a greater usage of memories, and non-shift-sharing a greater number of operations. The application of these different options in hardware reduction is illustrated in Fig. 3.10 for a particular cell configuration. In this example, we need three sub-templates to have the 9 original coefficients placed over allowed positions. Shifting operations require one extra coefficient circuit that is only set to one on the shifting template (S). Image-shifting is represented by straight arrows and result shifting by convex arrows over the grid. The non-shift-sharing option implies one extra shifting operation that is represented in dashed circle and lines in both image and partial result shifting modes.

The operation sequences proposed can easily be described by identifying the pixels of the image around a cell through the cardinal points (N, NE, E, SE, S, SW, W, NW), and the pixel that coincides with the cell as C. In so doing, the first sequence, based on the image-shifting, can be described in the following steps:

1. Gathering of the NW, W and SW contributions in the cell of interest by means of the application of the left side coefficients of the original template.

2. One pixel shift to the left of the original image.

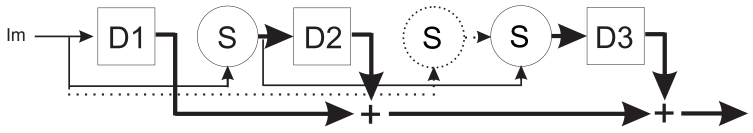
3. Gathering of the N, C and S contributions in the cell of interest by means of the application of the central coefficients of the original template to the shifted image and accumulation to the previously obtained result.



Original template
$$\begin{array}{|c|c|c|} \hline a_{11} & a_{12} & a_{13} \\ \hline a_{21} & a_{22} & a_{23} \\ \hline a_{31} & a_{32} & a_{33} \\ \hline \end{array} \qquad S= \begin{array}{|c|c|c|} \hline 0 & - & - \\ \hline 0 & - & 1 \\ \hline 0 & - & - \\ \hline \end{array}$$

$$D1= \begin{array}{|c|c|c|} \hline a_{11} & - & - \\ \hline a_{21} & - & 0 \\ \hline a_{31} & - & - \\ \hline \end{array} \quad D2= \begin{array}{|c|c|c|} \hline a_{12} & - & - \\ \hline a_{22} & - & 0 \\ \hline a_{32} & - & - \\ \hline \end{array} \quad D3= \begin{array}{|c|c|c|} \hline a_{13} & - & - \\ \hline a_{23} & - & 0 \\ \hline a_{33} & - & - \\ \hline \end{array}$$

Operations sequence for image shifting mode (with and whithout shift-sharing)



Operations sequence for partial result shifting mode (with and whithout shift-sharing)

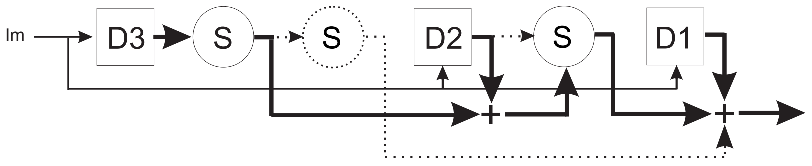


Figure 3.10: CPA with 4 CC per PE. Full-dense 3×3 template emulation employing image shifting mode or partial result shifting mode. Cell under study marked with a thick square.

4. One more pixel shift to the left of the shifted image (step 2). (Two pixel shifts of the original image if shift-sharing is not used.)

5. Gathering of the NE, E and SE contributions in the cell of interest by means of the application of the right side coefficients of the original template over the shifted image of step 4 and accumulation to the previously obtained results.

The second sequence, based on the output-shifting follows these steps:

1. Gathering of the NE, E and SE contributions in a cell placed two cells to the right of the cell of interest by means of the application of the right side coefficients of the original template to the original image.

2. Shifting the obtained result two pixels to the left to reach the cell of interest. (Shifting of one pixel to the contiguous cell where the next partial result will be obtained if we consider shift-sharing.)

3. Gathering of the N, C and S contributions in the cell on the right of the cell of interest by means of the application of the central coefficients of the original template over the original image.

4. Shifting of the obtained result one pixel to the left to reach the cell of interest and accumulation to the previous shifted result, or accumulation to the previous shifted result and shifting of that accumulation value one pixel to the left to reach the cell of interest if we consider shift-sharing.

5. Gathering of the NW, W and SW contributions in the cell of interest by means of the application of the left side coefficients of the original template. Accumulation of the obtained result to the previously accumulated results.

If we do not consider shift-sharing the order of coefficients application can be the same as that considered in the image-shifting sequence.

Figs. 5 and 6 in CNNA06 (p.113) also illustrate the different alternatives for the application of a full-dense 3×3 template over a reduced connectivity realization. The non-shift-sharing option is represented there with the number "2" as a two-step-shifting operation. Especially illustrative is the Fig. 6 of CNNA06 for the output shifting. There it is illustrated in which cell are collected the corresponding weighted contributions along with the shifts required to move them to the cell under study. In the image shifted option (Fig. 5 in CNNA06) all the contributions are obtained at the cell under study by weighting differently shifted images. In both cases the shifting template element should be set to zero in the sub-templates instead or being considered as "indiferent". Note the error in the number of remaining CC and inter-cell connections in explanation of these figures in the paper: in both cases the number is 4 instead of 3.

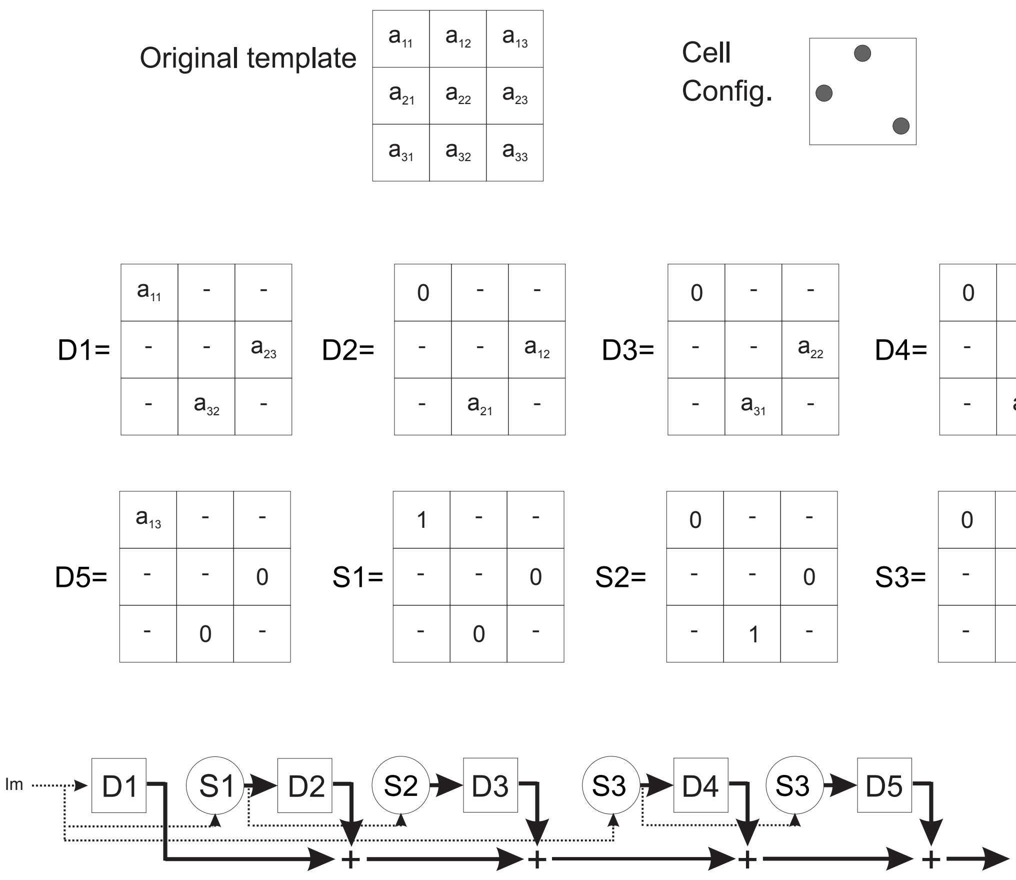

Original template
a_{11} a_{12} a_{13}
a_{21} a_{22} a_{23}
a_{31} a_{32} a_{33}
Cell Config.
D1=
D2=
D3=
D4=
D5=
S1=
S2=
S3=
Im
D1 S1 D2 S2 D3 S3 D4 S3 D5
+

Figure 3.11: Emulation of a full-dense 3×3 template over a 3 CC cell configuration with image shifting and shift-sharing when convenient.

Application Example

Fig. 3.11 illustrates the application of the methodology over a 3 CC configuration with image-shifting and shift-sharing when advantageous. In this case we need five sub-templates to cover all the elements in the original template and 3 shifting directions to gather all the contributions. Sub-templates are squared and identified as D (decomposition templates) and shifting templates are circled and identified as S. In this example we save one shift at the cost of an extra memory as we allow to re-start the shifting from the original image instead of always shifting the previously shifted image. Other alternatives with different sub-templates choice are also possible but we expect the same number of operations under the same considerations.

Cell Configuration Election

We have selected four criteria for the cell configuration election to help in finding configurations with no functional penalty and good time performance.

Functionality Criterion

The first concern in the cell configuration election is to keep the functionality of the CPA (and even extend it to LN kernels). Consequently, any election imposing restrictions to the kernel shape is rejected. Nonetheless, within the allowed configurations, it is very interesting to adapt the election to the most used shapes as it is going to lead to the minimum number of operations.

As it was previously indicated, the restrictions come from the required matching between the shifts needed by the sparse sub-templates and the shifts allowed by the correspondent cell configuration. This can be translated in being communicated, either directly or indirectly, with all the eight neighboring cells. In our analysis we assume that shifts are realized by kernels application and we do not consider time multiplexing in the CC usage, i.e. one CC is connected to only one neighboring cell. With this, the remaining CC have to offer a basis of movements from what all the connections can be recovered. The set of the four cardinal connections (NEWS) is the most straightforward primitive set. Nevertheless, we can reduce the number of CC to three by using the diagonal coefficient circuits taking into account the following rules:

1. At least one CC has to be on vertical or horizontal connections to avoid the chess bishop effect: we cannot achieve NEWS neighbors by just diagonal movements. This implies as well that the configuration with the four diagonal CCs is not allowed.

2. Two CC movements cannot cancel each other, and the third one has to complete the directions set. That is, if we have two diagonal CCs they have to belong to different diagonals, and the non-diagonal CC has to be on the direction not covered by the diagonal ones. Similarly, if we have two non-diagonal CCs one of them has to be on the vertical direction, and the other one on the horizontal. The diagonal CC has to cover the non-covered directions.

For example, if we want to keep the NE and NW coefficient circuits we have to keep the S CC as well and we can substitute the S and E CC by the SE if we keep the N and W, the W and NE or the SW and N.

Fig. 3.12 shows three cell configurations that, together with their rotations, represent the basic primitive sets that can be obtained under these rules. Starting from one of these 4 or 3 CC basis we can add other coefficient circuits in order to reduce the number of operations required by the full dense template emulation. A cell configuration of any size has to contain a 3 or 4 CC allowed configuration. The first configurations we reject are, then, those with one or two CC that cannot reproduce the communication to all neighbors. In cell configurations with more than two CC it is the shape what marks if the configuration is allowed or not. As an example, Fig. 3.9 (Figs. 3 and 4 in CNNA06 paper (p.113)) shows a 3 CC configuration that is not allowed (the horizontal CC does not complement the diagonal ones) and it has to be completed for its usage with an extra CC (Figs. 5 and 6 in CNNA06 paper and Fig. 3.10).

Performance Criterion

Having discarded the not-allowed configurations we look at the performance as the criterion in the cell configuration election. The performance is assessed in this case as

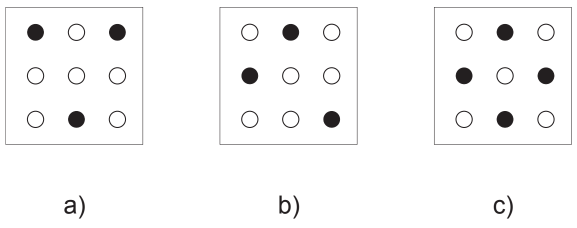


Figure 3.12: Possible minimal cell configurations.

a function of the number of coefficient circuits remaining and the number of operations required for a full dense template emulation, i.e. the area-processing time trade-off. The relationship between the number of operations required and the number of CC kept is not univocal but it depends on the CC allocation. Fig. 3.13 (Fig. 7 in CNNA06 paper in page 113) shows the minimum number of operations required by the best configurations of a given number of coefficient circuits. These numbers are obtained when the required shifts are mostly directly implemented. We apply shift-sharing whenever it is convenient. Note that in Fig. 7 of CNNA06 paper the number of operations for 3 CC configuration is 5+5 instead of 5+4. This difference is due to the fact that we have considered shift-sharing in any case in the paper, but we can have 5+4 operations if we combine shift-sharing with independent shifts (shifts from the original image in the image-shifting case). As a first filter we observe that configurations with 6, 4 and 3 multipliers (we consider 9 CC as the starting CPA configuration) require the same number of operations for a generic full-dense template emulation as configurations with more CCs. Examples of these configurations are also depicted in Fig. 7 of CNNA06 paper. Along this section we will see that configurations with 5 coefficient circuits, initially discarded, can be particularly interesting.

Number of Coefficient Circuits	Number of Operations (Sub-templates + Shifts)
$\textcircled{9}$	$\underline{1}$
8	2+1
7	2+1
$\textcircled{6}$	$\underline{2+1}$
5	3+2
$\textcircled{4}$	$\underline{3+2}$
$\textcircled{3}$	$\underline{5+4}$
$\xcancel{2}$	
$\xcancel{1}$	

Figure 3.13: Possible number of CC and minimum number of operations correspondence. Minimum number of CC for equal number of operations appear circled. Not allowed number of CC showed crossed.

To formally analyze the relationship between the benefit in area and the penalty in time-consumption, and compare the different cell configurations, we define a Figure of Merit (FoM), the *RPO*. The *RPO* is defined as the percentage of hardware **R**eduction **P**er **O**peration increased per original operation for the S&S template emulation. Eq. (3.6) summarizes the RPO definition. nc is the **n**umber of **c**oefficient circuits kept, HR is the **H**ardware **R**eduction factor and it is defined as the ratio between the number of CC removed and the original number of CC, and OIF is the **O**peration **I**ncrement **F**actor and represents the ratio between the number of S&S operations required after the hardware reduction and the original number of operations.

$$RPO(nc) = \frac{HR(nc) \cdot 100}{OIF(nc) - 1} \tag{3.6}$$

The 100% RPO is never reached for the emulation of one generic full-dense 3 × 3 template with this definition. Conceptually it would imply to remove all the coefficient circuits at the cost of one extra operation. In fact, having that we require to have as minimum 3 CC, the upper limit would be set to 67%. Nevertheless, actual values for one generic template emulation are much smaller, as we can see in Table 3.1 for the configurations selected in Fig. 3.13. For the general case we can see that the 3 CC configuration is much less efficient than the ones with 4 and 6 coefficient circuits. This is due to the complexity of the shifts and coefficient distribution in realizable 3 CC cases. The 6 CC configuration is the one that offers a better trade-off value for a general case.

Table 3.1: RPO for a generic 3 × 3 template emulation over different cell configurations.

Number of CC (nc)	HR	OIF	RPO(%)
9	0	1	0/0
6	1/3 (33 %)	3	17 %
4	5/9 (56 %)	5	14 %
3	2/3 (67 %)	9	8 %

We can improve the results by allowing the distributed implementation of the CC in the two typical templates of a CNN operation A and B at the cost of complicating the control. In the case of image-shifting we could apply two different sub-templates over two shifted versions of the image. In partial-result shifting we could apply one sub-template and accumulate a previous result within the same CNN operation or apply simultaneously two different shifts over two partial-outputs to be accumulated. The advantage obtained from this two-template CC allocation will strongly depend on the cell configuration considered. In the case of image-shifting, for example, we can apply simultaneously 2 of the 5 sub-templates of a 3 CC configuration by distributing the CC in 2+1. Nevertheless, we do not obtain any benefit in applying it to a 3 CC with output-shifting or to a 6 CC parallel configuration with image or output-shifting and we can reduce to 1+1 the number of operations for a 6+1 configuration. With a 3+3 CC configuration in diamond shape we can apply simultaneously 2 of the 3 sub-templates with image-shifting requiring 2+2 operations, or the 2 shifts in just on operation for output shifting (3+1), but we require 2+1 operations if we distribute the 6 CC in parallel. In general, a significant improvement in the minimum-sized template emulation is more difficult for image-shifting as it would usually require more physically

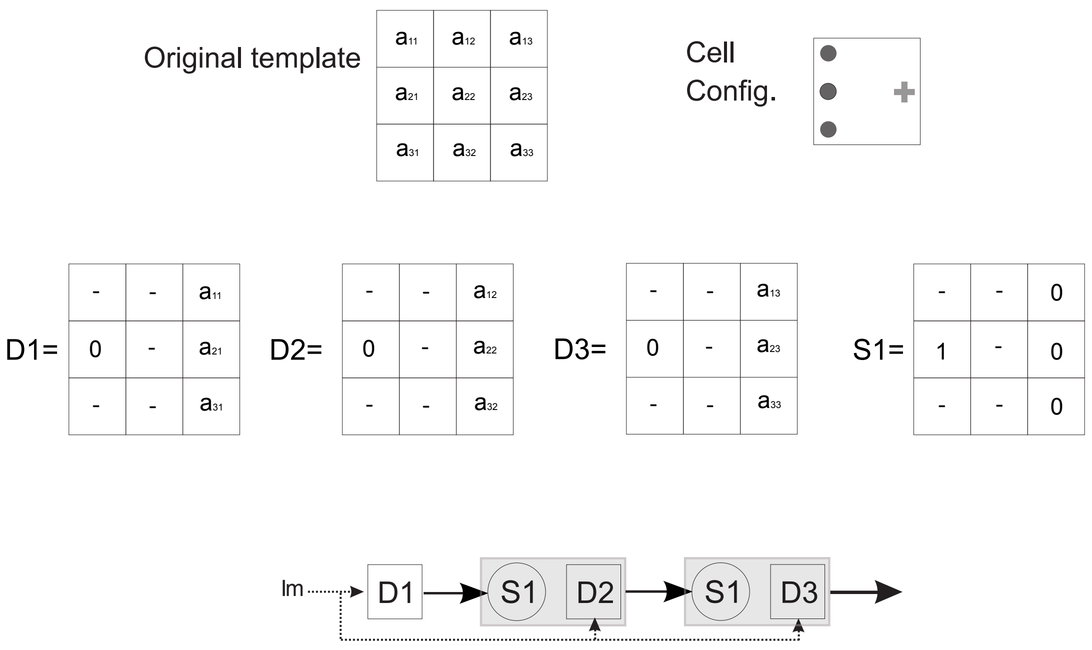


Figure 3.14: Example of CC distributed in two templates. Coefficient circuits are represented as dots or crosses depending on in which template are allocated. Sub-templates are squared and shifts are circled in the sequence of operations. Operations applied simultaneously are put together in a gray rectangle.

implemented CC in both templates to simultaneously apply two sub-templates, while result-shifting can obtain improvements with just one CC in a different template to simultaneously shift a previous partial result. With the proper election of the CC distribution we can even hide all the shifting operations for result-shifting as it is shown in Fig. 3.14 for a 3+1 configuration, while for image-shifting it is usually better to keep all the CC in the same template. With that configuration we reduce the total number of operations from 5 to 3 in a full-dense 3×3 template emulation by hiding the shifting operations. Additionally, we can consider that the CC in the second template is specialized as shifting coefficient and we can simplify it to a 1-bit programmability CC even within a G/S implementation. The same result would be obtained with a 3+2 CC configuration in diamond shape. All in all, the two-template cell configuration approach will be especially useful in the application of LN templates over a reduced PE as it will be shown in the next section.

In considering typical two-template DTCNN operations we distinguish two approaches. In the first one we keep a two template implementation with the corresponding hardware reductions (either the same in both templates or a different number of CC in each template). In this case we can apply simultaneously the sub-templates of both templates. Shifts for each template can be combined just if we have the same configuration in both templates and the same image is weighted by the two templates ($Y = U$), or if we choose partial-result shifting. In these cases we have the same performance as for one-template operations (Table 3.1). In any other case, shifts have to be performed in different CNN operations, and their outputs (shifted images) must

Table 3.2: RPO for the emulation of a generic two-template operation over cell configurations selected in Fig. 3.13.

T_{nc}	**Separated Hardware**		**Hardware Sharing**	
	(Simult. Shifts)	(Sequential Shifts)	(Sequential Shifts)	(Simult. Shifts)
9	0/0	0/0	50 %	50 %
6	17 %	11 %	13 %	17 %
4	14 %	9 %	9 %	11 %
3	8 %	5 %	4 %	6 %

be saved separately. As a consequence, the RPO drops with respect to the cases of one-template operations.

A second option is to re-use the same hardware for the application of both A and B templates (one-template implementation, hardware sharing). This was proposed, without extra reduction in the number of CC, in [Paasio et al., 2002] with the difference that we do not obtain a transient mask from the application of one of the templates but we consider the accumulation of the outputs from each template application.

Performance calculations for these cases are shown in Table 3.2 as a function of the number of remaining CC per template (T_{nc}). In this case the reference for the HR is $nc = 18$, that corresponds with the original implementation of two templates. The second column shows the results of considering separated hardware for each template (A and B) application, under the consideration of the same CC configuration for both templates, and simultaneous shifts. In absolute terms the total number of CC is doubled in this case, but the HR is the same as for the case of one-template operations as now the reference hardware is a two-template implementation with 18 CC. Together with the simultaneous shifts (A and B sub-templates application is already simultaneous in separated hardware), it provides identical results as that of the one-template analysis. The third column displays the values rendered by considering sequential shifts with the two-template hardware available. In both situations the case of 9 CC keeps the original total of $nc = 18$, and there is not increment in the number of operations ($HR = OIF = 0$ and $RPO = 0/0$). The fourth column shows the performance of the hardware sharing option. The HR for a 9 CC configuration is 1/2, and the number of operations is doubled because of the sequentially application of the A and B templates and shifts, what results in an RPO of 50%. In the last column we consider that the shifts are always applied simultaneously for both templates (restricted in this case to the $Y = U$ situation) over a shared hardware. In the 9 CC case we do not have shifts to apply simultaneously, and so the RPO is the same as in the previous column. It is in the rest of the cases where we can observe the improvement of having half of the shifting operations. In all the cases we have used the S&S shift-sharing option when advantageous.

With half of the CC, the performance in the hardware sharing option is similar to or even better than that of the separated option for the same configurations, having that the number of operations is not doubled due to the shifts particular consideration. In addition, if we compare the total number of CC we conclude that, for the general case of a two-template operation, it is a better option to keep all the CC implementing the same template, i.e. a hardware sharing option. Within this hardware sharing option we can consider as well the distribution of the CC in two templates with the same improvements as in the single template case. It should be noted that this analysis holds for a synchronous CPA implementation as DTCNN where A and B are interchangeable templates, but not for CTCNNs where S&S are only applicable to B template.

Tables 3.1 and 3.2 were also included in the CNNA06 paper (p.113) as Figs. 8 and 9. The difference in the 3 CC configuration reported before is also extended to the RPO values that in addition have been rounded.

As a final remark we should note that the HR definition used in the RPO analysis considers the reducible area as a function of the number of CCs. Nevertheless, when we remove a CC the connection to the corresponding neighbor is also removed. Having that the number of CCs and the number of connections differ in the number of central CC considered (one if we consider a one-template implementation and two in a two-template implementation), the actual area accounted for in the HR factor as defined is the area corresponding to the removed CCs plus the proportional part of the area occupied by the connections. On this basis, when we remove a central CC (not connected to any neighbor) we are overestimating the reduced area and we underestimate it when we remove a non-central CC. This is the simplest consideration of the hardware reduction, but it hides the differences between reduced cell configurations with and without central coefficient circuits for the same number of CCs. To take this into account and refine our discrimination between cell configurations we can define a new hardware reduction factor HR_{AVE} as the average of the HR_c and the HR_{CC}, defined the former as the ratio between the number of connections removed and the original number of connections and the latter as the ratio between the number of CCs removed and the original number of CCs. With this definition we consider that both groups, connections and CCs, occupy the same area, what is not true in general, but allows a fairer comparison. Due to the different initial number of CCs and connections this assumption implies that we consider that the area occupied by one CC is lower than that occupied by one connection, what is more likely in B/W implementations. Nevertheless, the numbers given by this hardware reduction definition, HR_{AVE}, can only be taken as guidance and its usage for RPO calculation can lead to erroneous comparisons. Besides, this average definition makes an error in the cell configuration area comparison if the starting point configuration is a two template configuration with two central CCs and the area of a CC is larger or equal than the area of a connection (more likely in G/S implementations). In those cases the average definition of the hardware reduction (HR_{AVE}) provides a better value for configurations with two central CC in comparison with configurations with one less CC but with no central CCs, what is just true when the area occupied by the CC is smaller than the area occupied by a connection.

For a completely fair comparison we should know the exact percentages of occupation of the CCs and the inter-cell connections. We can account for the actual area reduced by using a weighted average hardware reduction definition (HR_{weight}), where the HR_c and the HR_{CC} are weighted by they corresponding values before adding them. With this definition we observe the particular behavior of the configurations with two central CCs in implementations where the CCs are smaller than the connections, which is not observed in the opposite case. Nevertheless, at the sight of the particular shapes exhibited by the configurations with one-less-CC and no centrals, this is just a second order nuance in the cell configuration election to be taken into account in particular cases and if we have enough knowledge about the implementation characteristics. For a general analysis, and particularly when considering a one-template starting point, we should use the simple definition, just taking into account that configurations with the same number of CCs as the more central CCs (1 or 2) the lesser the area.

Shape and Symmetry Criterion

By analyzing the RPO definition we observe that it would provide a 100% value when the number of operations is increased in the same percentage in which the hardware is reduced. In this situation we could consider benefit and penalty as balanced. Values over 100% would appear with a number of operations increment (penalty) less significant than the hardware reduction (benefit). Both situations, balanced and improved, require values of OIF under 2, i.e. less than one operation increment per original operation. For just one template emulation we can have only integer values of the OIF, 1 when all the elements in the template have their corresponding CCs implemented, and 2 if they require, for example, a shift prior to the template application.

An OIF value less than 2 is achievable if we consider the particular shape of the templates to be emulated in the cell configuration election. In fact, if we consider a particular algorithm we can get a variety of RPO values depending on the grade of matching of the templates implicated with the cell configuration selected as OIF values between 1 and 2 are provided. This led us to an application-led cell configuration election. In this case, *RPO* allows us to easily compare the cell configurations not only for the general case but for particular applications. Note that infinity values are reached with hardware reduction without increment in the number of operations, i.e. without penalty. This identifies the cell configurations that fit the template or templates under consideration. It is interesting to note that the RPO behavior with the OIF value is similar to the $f(x) = 1/x$ function, and thus little variations of the OIF when its value is lower than 2 cause big variations in the RPO value that do not correspond to such big actual improvements. This application of the RPO to algorithms or complex tasks is illustrated in section IV of the CNNA06 paper (p. 113), and more widely in the DCIS06 paper (p. 121) and in the Validation chapter of this thesis.

Going further we have studied the template shapes in an attempt to find out predominant shapes and symmetries in the most usual CNN templates and determine the existence of a more adequate distribution of coefficient circuits in a general purpose CPA architecture. The study is gathered in the CNNA08 paper (Appendix A, page 141). The study was realized over the Cellular Wave Computing Library (CSW) [Roska et al., 2000]. Note that templates gathered in the CSW library are to be applied to continuous-time CNNs. The translation of the library to DTCNNs is indicated in Fig. 5 of CNNA08 paper (p. 141).

The first conclusion we draw from the statistical study of the templates featured in the CSW library is that a dense configuration might be inefficient as the percentage of CNN operations with four or more null coefficients amounts to more than 70% (see Fig.3 at CNNA08 paper), being also remarkable the high percentage of very sparse templates with only two or three non-null template coefficients (see Fig.4 at CNNA08). Note that we have not taken into account templates with only one central coefficient in these percentages as they are just used to perform Boolean operations in DTCNNs.

This can suggest even an S&S configuration with only one, two or three coefficient circuits that can be used alternatively (time-multiplexed) to weigh neighbor values in a diamond shape [Sargeni et al., 2005, Dudek, 2000]. This means one, two or three coefficients, with four NEWS inter-PE/cell connections.

From Table I (CNNA08 paper), the second main conclusion is that the S&S configuration of five coefficient circuits in diamond shape (NEWS + central) represents a good trade-off between hardware reduction and processing time penalty for a general purpose realization, especially if we take into account the presence of symmetries. In fact, it can approach more than 30% of CSW templates considered in only one step (complete matching with the template shape), with an average number of S&S operations lower than 3. Also, it is relevant the importance of the central coefficient circuit as the configurations considering it shows an improvement between 0,6 and 1 in the average number of steps, and, more significantly, they realize between 10 and 30% of the templates considered in one step versus the 0% obtained by the same configurations without central CC. Moreover, it could even be interesting to have an extra central coefficient circuit in a second template for the case of pixel-to-pixel operations between two images like Boolean or arithmetic (addition and subtraction) functions. We have checked, as well, that these tendencies are similarly followed for both gray-scale and binary (B/W) operations.

It is also interesting to remark the significant appearance of symmetries, being, for example, less than 1% the amount of dense templates without, at least, axial/mirror symmetries. With a smart use of such symmetries in the S&S methodology result shifting mode we could reach significant reductions in the number of operations. We illustrate the procedure in Fig. 3.15. We consider the most inhomogeneous dense template element distribution with axial symmetry of the CSW library according to CNNA08 paper (p. 141), i.e. distribution 5 in Fig. 1, and the 5 CC diamond cell configuration. In this situation, at the same time that the contributions of the neighbors N, S and central are collected at the cell under study the contributions of the neighbors NW, W and SW are collected in the W neighboring cell as its own N, S and central neighbors contribution. This is the same with NE, E and SE neighbors contributions in the E cell. And so, as the template shows an axial symmetry (same weights in NW and NE, W and E, and SW and SE, in this case), we can accumulate in just two operations the contribution of the six lateral neighbors by taking advantage of results calculated in different cells. Note that most of the non-diamond distributions, i.e. with non-null diagonal (SE, SW, NW or NE) elements, exhibit this kind of symmetry and this way of taking advantage of it reinforce the 5 CC diamond cell configuration preference.

From the analysis of real-life applications in the CNNA08 paper study, and far from being determinant due to the reduced number (just five) of applications analyzed, we extract some conclusions that can complete the guidelines given by the general study. The first one is that Diamond, Dense and Central (only central coefficient) are the most used shapes in these applications. Irregular or Diagonal templates are rare. Templates with only non-zero central coefficients in the applications studied are employed for Boolean operations and are usually substituted by a specialized unit. Diamond templates would be implemented with a reduced number (five) of coefficient circuits. Finally, if dense templates have symmetries, these can be easily approached started from a diamond template applying the S&S methodology. This is the case of

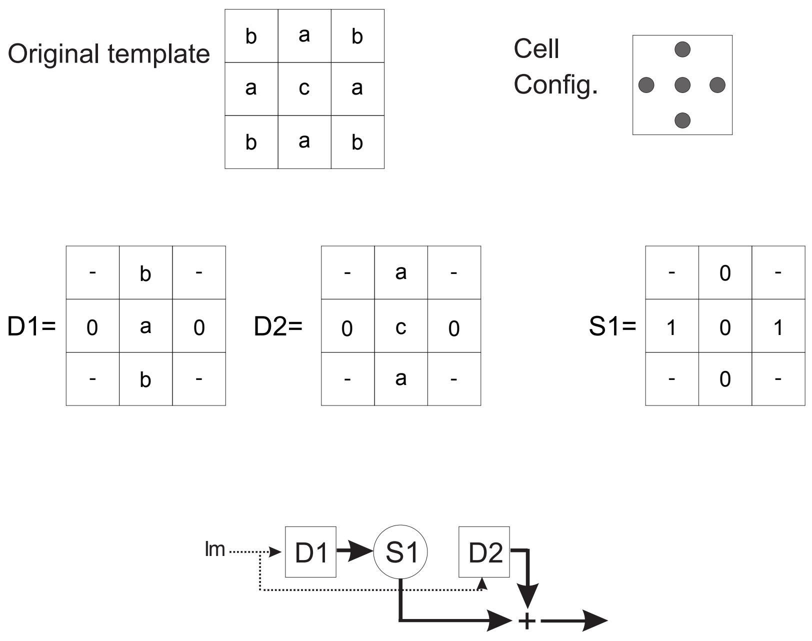


Figure 3.15: Example of taking advantage of template symmetries in S&S partial-result shifting mode.

the most of the templates of the applications discussed.

Goals Criterion

Finally, we have to take into account that the *RPO* value is a measure of the trade-off. The final decision on the particular number and arrangement of coefficient circuits in a cell would be led by the actual requirements of area and processing time in the application.

Section V in CNNA06 paper (p. 113) gathers some details about the trade-off assessment and possible consequences of the CC remove to be taken into account in reaching particular time or area goals. These details are also analyzed in the Validation chapter of this thesis.

3.4 S&S for LN Template Emulation over Simplified Hardware

As it was mentioned before, the hardware reduction by application of the S&S techniques can affect the processing time but it does not affect the functionality of the implementation. A good example of this is the application of the S&S techniques for LN templates emulation over a reduced implementation. Although it was considered as a particular complex algorithm in the CNNA06 and DCIS06 papers (pages 113 and 121 respectively) we consider it is interesting to dedicate a particular section to this combination.

The most straightforward solution is to perform each LN sub-template as if they

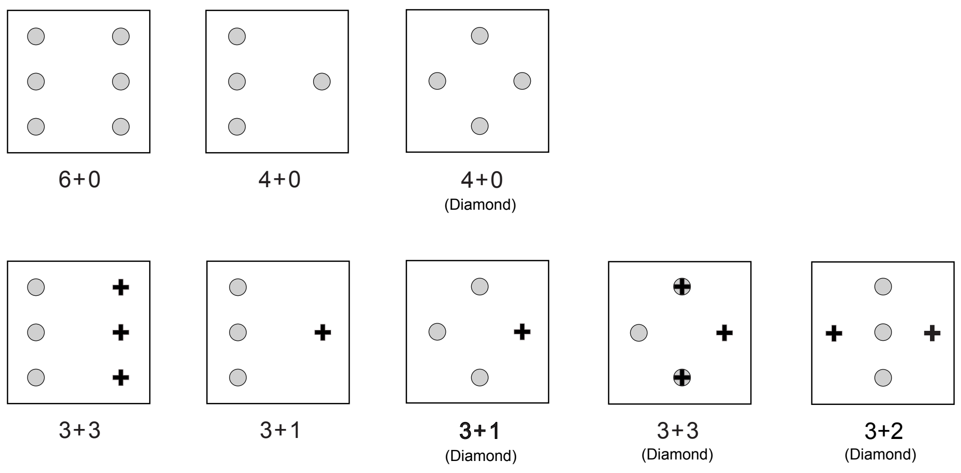


Figure 3.16: 4 and 6 CC configurations with one and two templates (templates CC differentiated as circles and crosses) for LN emulation.

were templates of a given algorithm. Nevertheless, the number of operations can be dramatically shrunk if we take into account that it is possible to directly split the original large neighborhood template taking into account the new cell configuration without the limits of the initial 3×3 splitting. In addition, the shifts required by the hardware reduction are hidden by the shifts used for LN emulation. Moreover, it is interesting to have a good fit between the cell configuration and the LN shifting technique in order to require just one step per shift operation. For example, from the analysis of the LN shift techniques we observe that zig-zag and spiral ones need a reduced number of shift directions (four cardinal points if we do not consider diagonal shortcuts). This suggests a NEWS cell configuration. Of course, we consider again the sharing of the shifts as a good way to drastically cut the final number of operations.

From the RPO values in Tables 3.1 and 3.2, we see that configurations with six and four coefficient circuits are the most efficient ones in a general template case. By considering the shape analysis we have also seen the convenience of configurations with five coefficient circuits in a diamond shape. The elected cell configuration and shifting technique have to match each other as much as possible to require a minimum number of operations. A priori we assume shift-sharing and mainly horizontal shifts as in the zig-zag shifting techniques. We choose the configurations in Fig. 3.16, that implement horizontal shift directions. We differentiate configurations with all the coefficient circuits implemented in one template ($nc + 0$), and configurations that implement two templates ($nc_A + nc_B$), where coefficient circuits from the two templates are differentiated as dots and crosses in Fig. 3.16. The second option, two templates, complicates the hardware control and compromises the memories availability, but it achieves better performance, especially if we consider partial-result-shifting. Image-shifting takes less advantage in general of the two-template implementation than the partial-result-shifting, which can apply simultaneously a sub-template and a shift operations just requiring one CC in a separated template.

Table 3.3 shows the performance values in the implementation of a full-dense 9×9 template ($n = 4$) for the selected configurations. Partial-result-shifting is considered

Table 3.3: RPO for 9×9 template emulation over cell configurations of Fig. 3.16.

Cell Config.	**HR**	**OIF** (ref. Zig-zag)	**OIF** (ref. Central)	**RPO (%)** (ref. Zig-zag)	**RPO (%)** (ref. Central)
3+1	5/9 (56%)	36/33	36/29	611	230
3+1 (D)	5/9 (56%)	40/33	40/29	262	146
3+3	1/3 (33%)	36(42)/33	36(42)/29	367(122)	138(74)
3+3 (D)	1/3 (33%)	38(48)/33	38(48)/29	220(73)	107(51)
3+2 (D)	4/9 (44%)	34/33	34/29	1467	258
4+0	5/9 (56%)	57/33	57/29	76	58
4+0 (D)	5/9 (56%)	57/33	57/29	76	58
6+0	1/3 (33%)	39/33	39/29	183	97

in the two-template configurations ($3+3$, $3+1$, $3+1(D)$ - Diamond -, $3+3(D)$ and $3+2(D)$). For the $3+3$ and $3+3(D)$ configurations we consider also image-shifting as they have enough CC in both templates to take advantage of the simultaneous application of sub-templates, and the results are presented between brackets. To adapt the shifting technique to the cell configuration we choose the previously used zig-zag for the diamond configurations as they implement both horizontal and vertical shifts. For the rest of the configurations is more efficient to shift each line contribution to the central cell directly because they do not allow direct vertical shifts, but diagonal ones. This *by-rows* technique is shown in Fig.3.17 for the non-symmetric $3+1$ configuration. Sub-templates center election depends on the cell configuration. We calculate the OIF with two references: the original S&S LN implementation (not hardware reduction) with zig-zag (regular technique) and central (very irregular but with the best results in LN) shifting techniques. HR reference is one template with 9 CC.

In the two-template configurations with partial-result-shifting we hide the shifts that can be realized with the idle hardware by applying them simultaneously to the sub-template application. Note that this situation is more significant in symmetric configurations ($3+3$, $3+3(D)$ and $3+2(D)$ in Fig. 3.16) because it can be applied regardless the direction of the shifting (left or right in the case of spiral or zig-zag techniques). For the non-symmetric configurations ($3+1$ and $3+1(D)$) we have to apply sub-templates and shifts separately, at least in the right or in the left hand-side around the central cell to reach it (right hand-side in Fig.3.17). We only apply image-shifting in the two-template symmetric configurations. In this case we shift the image in opposite directions, for example, and we can apply two sub-templates simultaneously. It is interesting to note that two-template configurations ($3+1$, $3+1(D)$ and $3+3$) overcome one-template equivalents ($4+0$, $4+0(D)$ and $6+0$ respectively) in number of operations for result-shifting, but not for image-shifting, where $6+0$ offers equal or better results than $3+3$ depending on the way of shifting the image. The $3+3(D)$ configuration could be interesting in implementations where image-shifting is mandatory (completely binary implementations, for example), the number of operations is more critical than the area occupation, and we have to implement diagonal templates frequently. Otherwise, one-template configurations are more interesting for image-shifting. In the partial result shifting mode, the $3+2(D)$ configuration offers the best trade-off value and the best number of operations, with just one more operation

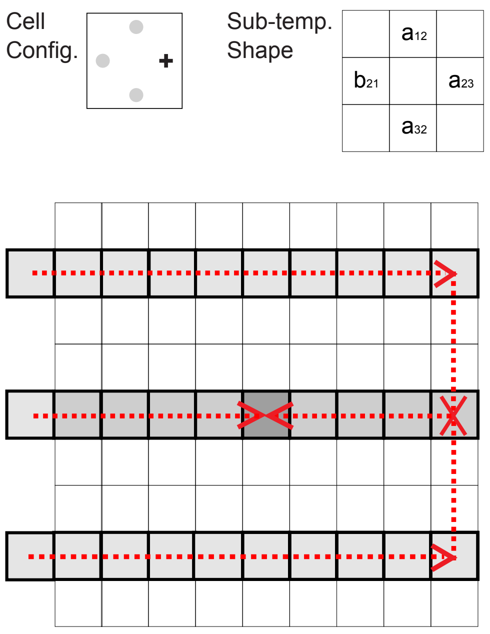


Figure 3.17: By-rows shifting technique.

than the original 9 CC configuration with a zig-zag shifting technique (Table 3.3). In addition, it offers the better shape for most usual 3×3 implementation as it was shown in a previous section. $3+1$ configuration also offers a good trade-off with lower area occupancy, but perhaps the diamond shape could make the $3+1(D)$ configuration more interesting for a generic implementation. We also observe that lateral configurations require less number of operations than diagonal ones for the LN emulation with two-template configurations. This is due to the isolated elements that diamond configurations leave in a square-shaped LN template. Finally, a $5+0(D)$ configuration does not improve the results of the $4+0(D)$, but it could be more interesting for a generic implementation with only one template according to the dominant shape extracted from the CNNA08 paper study (Appendix A, page 141).

Interesting remark is that RPO values that overpass the 100% indicate the superiority of the hardware improvement with respect to the number of operations penalty. This is the result of an average increment of less than one operation per original operation. This improvement is due to the re-splitting of the LN template, the good match of the configuration to the shift technique and, in the two-template configurations, the simultaneous application of operations.

In Fig. 3.19 we illustrate the general process of LN emulation over a limited connectivity hardware through its application to an homogeneous binary diffusion template (Fig. 3.18.a) that requires a small number of different sub-templates. This particular diffusion template was originally intended to the implementation of the internal potential of the pixel level snakes algorithm presented in [Vilariño and Rekeczky, 2004] for its application over the binary hardware introduced in [Brea et al., 2006]. The aim was to smooth a curve depicted as a series of connected black pixels over a white background, partially removing rough concavities. Sub-templates centers are emphasized by shadowing over the original template shown in Fig. 3.18.a together with the

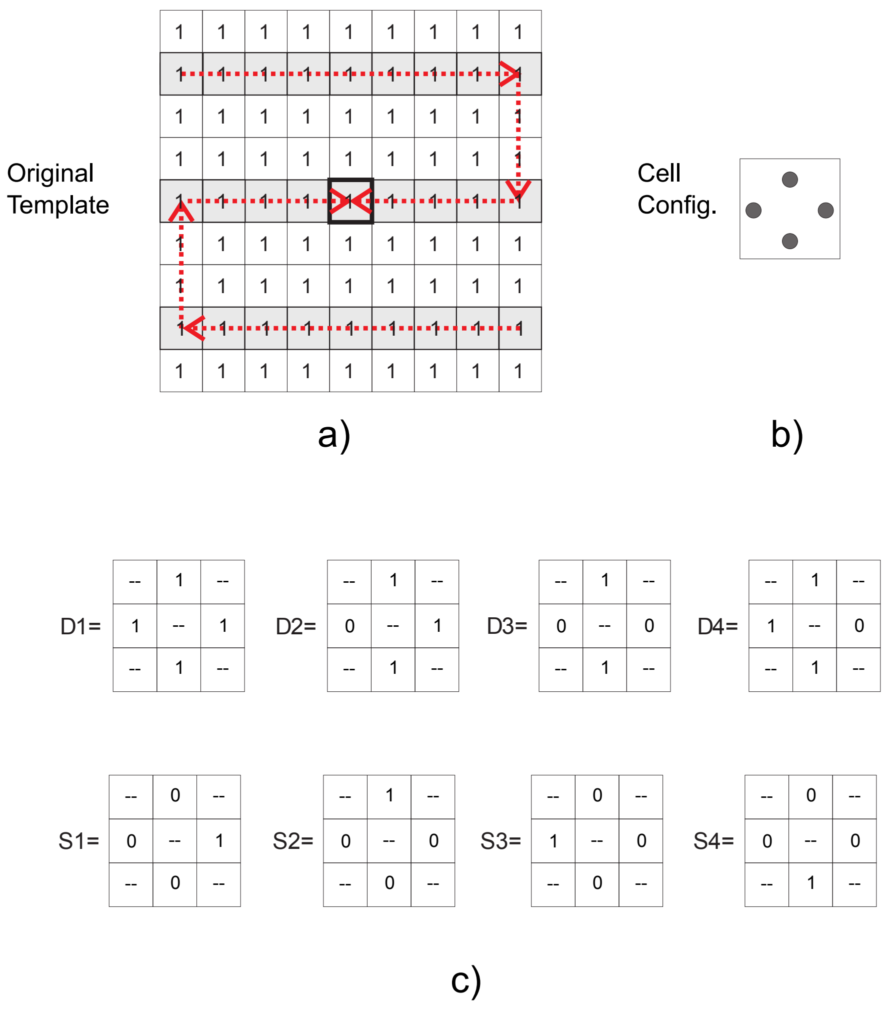


Figure 3.18: a) Binary 9 × 9 diffusion template to be emulated. Sub-templates centers shaded. Shifting directions and route marked with arrows. b) Cell configuration: 4+0. c) Sub-templates (D) and shifting templates (S).

image-shifting route. We have chosen to realize the sub-templates application in zig-zag with two starting points, as shown in Fig. 3.4, as it saves 4 shifts with respect to the one-starting point zig-zag technique. In this case we do not need diagonal shortcuts because the size of the template (9×9) is multiple of 3×3 . To use the same configuration in a different case we have to substitute the diagonal shifts by one horizontal and one vertical shift. Following the consideration of directly implemented shifts (one shift operation per shift direction required) and selecting image-shifting according to the binary implementation, we choose a 4 CC diamond configuration (Fig. 3.18.b). Required sub-templates and shift templates are depicted in Fig. 3.18.c. Fig. 3.19.a shows the system-level operations required for the image shifting mode with shift-sharing except in the starting of the second half of the template, the second starting point. Shift operations are indicated by circles and sub-template application by squares.

We had chosen for our example image-shifting and only one template configuration. In considering a G/S physical implementation, and using partial-result-shifting, we

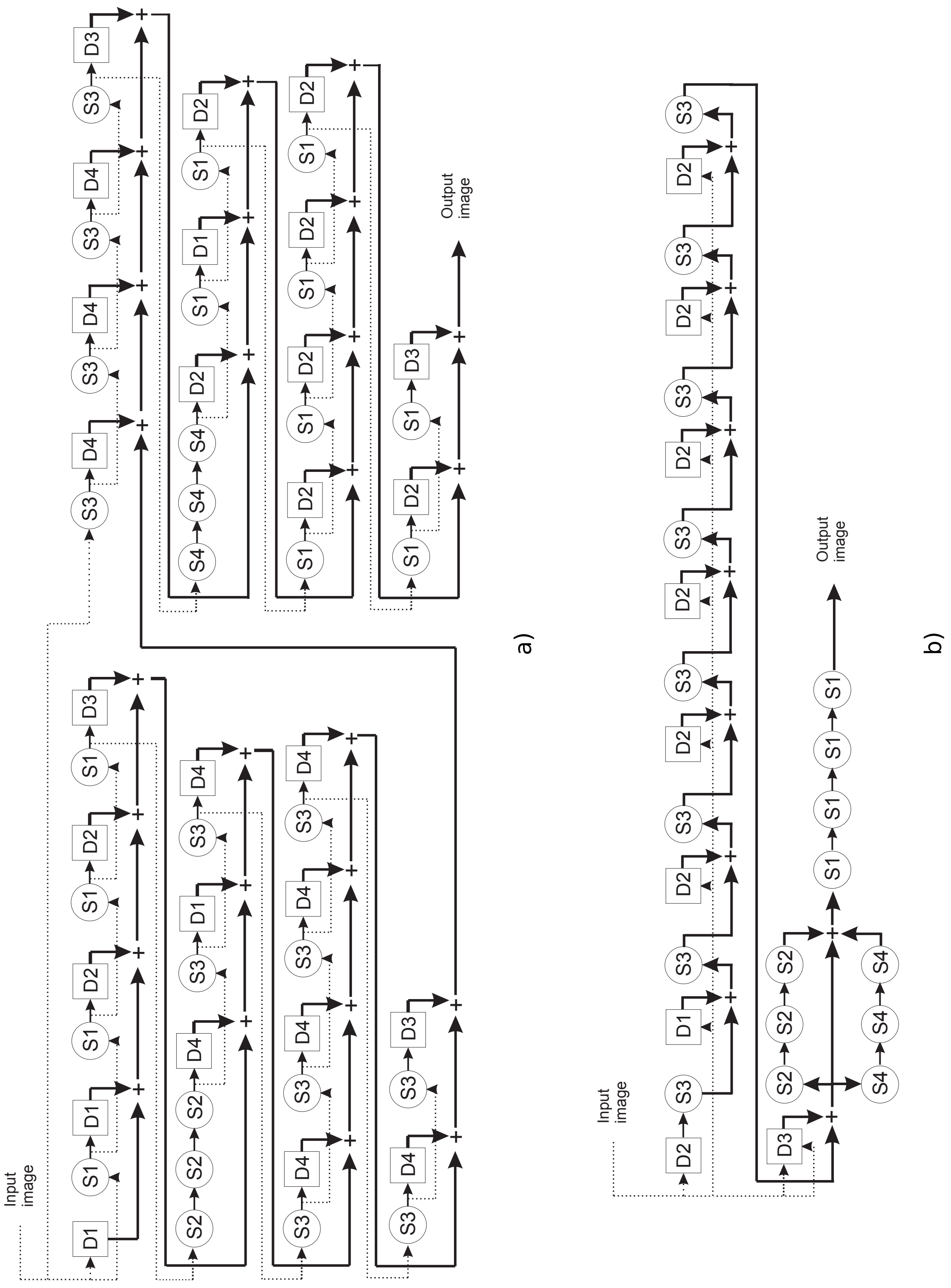


Figure 3.19: S&S emulation of a 9 × 9 diffusion operation. a) System-level implementation for S&S image-shifting mode. Image feedback for shifting as dashed arrows. b) System-level implementation for S&S partial result shifting mode and homogeneity simplification. In both cases, shift-sharing is used when convenient.

could take advantage of the template homogeneity by taking into account that while the first line of partial results is being accumulated at the central cell of the last sub-template, the corresponding accumulation is being obtained in the last sub-template of the other two lines, and thus we have only to add them to obtain the whole template result and shift it to the LN template central cell. In so doing, we would need 15 S&S operations instead of the 29 required with the central shifting technique with image-shifting in a full-dense configuration (9 CC). The system-level operations for a 4 CC NEWS configuration and zig-zag technique is shown in Fig. 3.19.b resulting in 27 S&S operations instead of the initial 57 of the original Fig. 3.19.a process, illustrated in the figure for image-shifting. We could further improve the number of operations if we distribute the four CCs into two templates, by applying a sub-template and accumulating the previous result in the same operation. Note that it could be only realized if the shifting coefficient is not required by the sub-template application, i.e. the hardware required for shifting is idle.

Fig. 3.20 shows the scheme of the application of the methodology to a G/S 9×9 diffusion template with other two different cell configurations. In the a) case we have a single template 6 CC configuration that cannot implement vertical shifts but implement the required movement through diagonal shifts with a by-rows shifting technique. b) case shows a 6 CC cell configuration where the CC have been distributed in two templates, a 3+3 CC diamond cell configuration in this case. In both cases templates centers are marked with thick lines and in the second case we differentiate the templates centers by using dash lines in those corresponding to the second one. In the application of the zig-zag shifting technique we mainly use the left template as sub-template and the right template for shifting for the first and half of the second line in b). For the implementation of the rest of the sub-templates the roles are exchanged.

In both a) and b) cases we could use the same scheme for image and partial-output shifting, just by taking into account that for image-shifting we start the shifting from the arrows head. In the second case, as we have a two-template configuration, we can apply simultaneously two sub-templates (image-shifting) or a sub-template and a shift-accumulation operation (output-shifting). Case a) requires less number of sub-templates but the same number of shifts as case b). On the other hand, for a partial-output shifting option, case b) would hide most of the shifts, what leads to better results. Fig. 3.21 detaches the templates to show in detail the operations for the upper zig-zag half in this case b). As all the sub-templates are different we directly display them in the sequence of operations. In a total of four occasions we apply both templates at the same time as sub-templates. Those are marked with a smaller rectangle within the square in Fig. 3.20. We have just one shift in the central line that cannot be hidden by the sub-templates application (it belongs to the second half and it is required to reach the central cell). The other 6 non-hidden shifts corresponds to the line change.

Note that we have mirror symmetry in the template elements but not in the derived sub-templates. In this case we could take advantage of the symmetry if we have, for example, a 5 CC diamond one-template configuration. With that configuration we could use the central coefficient circuits to apply the sub-templates and the lateral ones to shift the obtained output in both directions right and left. As the partial output should be gathered in different memories depending on the side it comes from, the shifts have to be applied in different operations. Nevertheless, in this case the

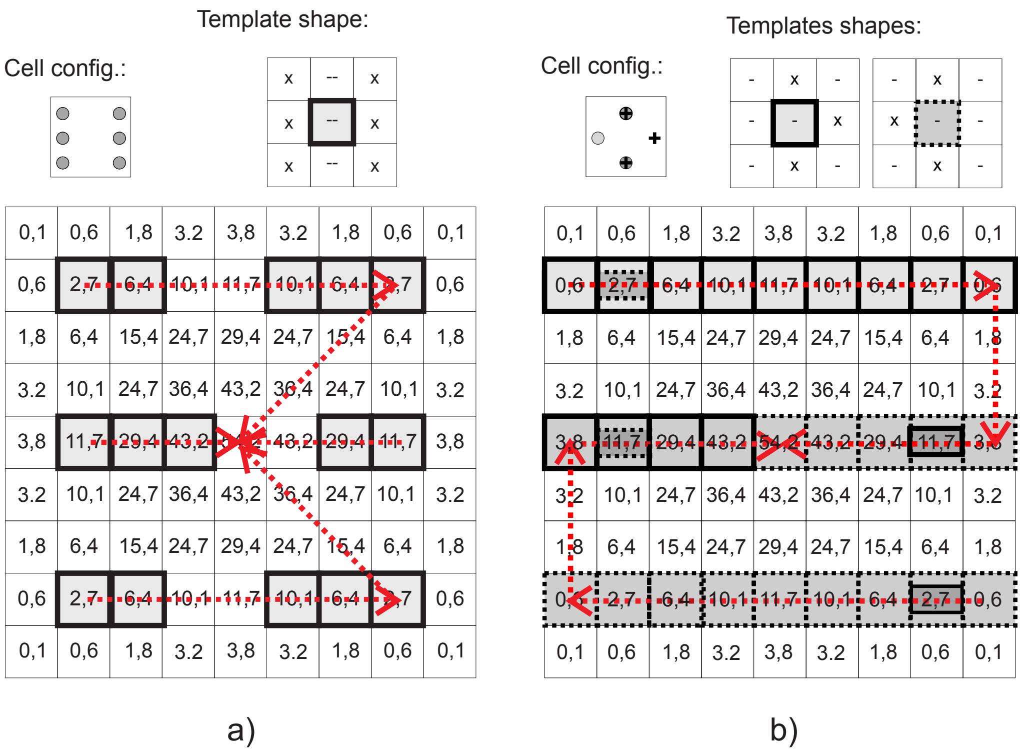


Figure 3.20: Schemes of LN implementation over two different simplified cell configurations. a) 6 CC single template cell configuration. b) 6 CC two templates cell configuration $(3 + 3(D))$. Arrows indicate the shifting routes.

symmetry usage does not provide a definitive improvement. It would provide the correct output with 45 operations (15 sub-templates and 30 shifts), 12 less shifts than the 5 CC diamond without usage of symmetry and just six more shifts than the 6 CC configuration in Fig. 3.20.a, but 11 more operations than that obtained if we divide the CC in two templates $(3 + 2(D))$, having the CC dedicated to shifting in a different template and allowing the simultaneous application of sub-templates and shifts as was shown in Fig. 3.21 for a $3 + 3(D)$ configuration.

3.5 Summary and Conclusions

In this chapter we propose a common methodology to deal with both large neighborhood emulation and hardware reduction, seen as the implementation of templates with sizes that overflow the physically implemented resources.

The so-called Split and Shift methodology can be placed within the *partition and shift* proposals for LN emulation. The main contributions of our proposal are the simplicity of the conception and an organized set of guidelines of application to obtain a minimum penalty at processing time and absolutely no penalty at functional level in the achievement of the goals.

In the LN emulation we measure the cost of widening the CPA functionality as the number of operations required for the LN operations application. From the analysis

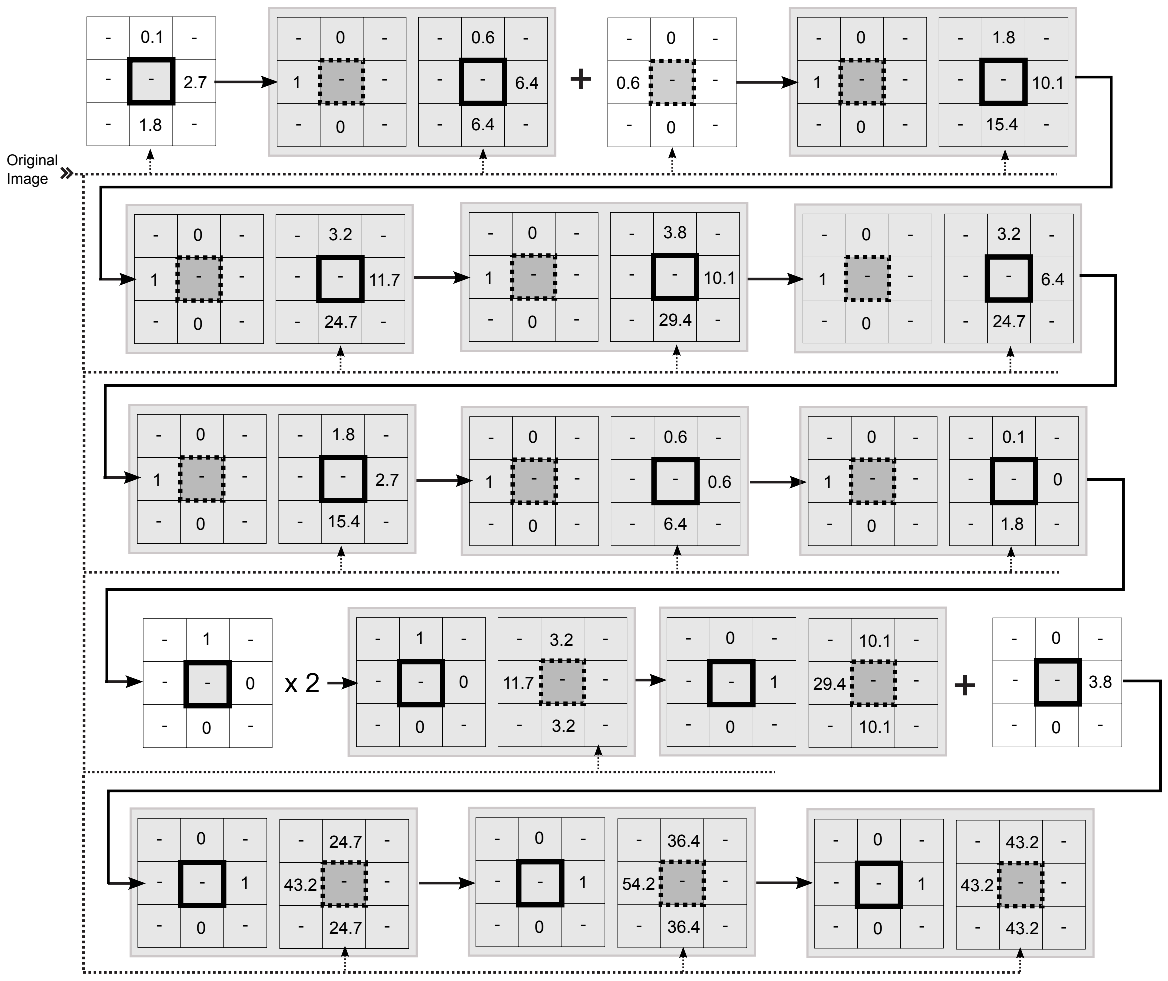


Figure 3.21: Detailed operations for the upper zig-zag half of a 9 × 9 template for the $3 + 3(D)$ cell configuration under the scheme in Fig.3.20.b

we mainly conclude that the splitting methods should begin from a template corner and overlap incomplete sub-templates when necessary to keep the sub-template centers close to the central cell. About the shifting techniques we observe the convenience of the shift-sharing option in both image and partial-result shifting modes. A regular process together with shift-sharing can benefit in terms of simplicity and automation. We suggest, as the best option, a concentric decomposition and spiral or zig-zag shifting. Nevertheless, central shifting offers slightly better results in number of operation at the cost of irregularity for more demanding implementations.

In the case of hardware reduction we have a trade-off between the benefit obtained in hardware reduction and the number of operations required to keep the functionality of the implementation. This trade-off does not depend only on the number of CCs, but on the selected cell configuration. We have gone over the cell configuration election under four criteria. The first criterion ensures the preservation of the full functionality without restrictions at kernel shape or size. This criterion imposes a minimum number of 3 CC and a distribution of CC that allows all the shifts required to communicate to all neighbors. The second criterion takes into account the performance of the implementation by defining a Figure of Merit. This FoM is called RPO and measures the relation between the percentage of CC reduced and the number of operations increased per original operation. For a general single-template operation we would select a 6 CC lateral configuration, with no one CC at the central column, as the best trade-off option. Nevertheless, if we allow the distribution of the CC in two different templates we obtain better trade-off value with a 3+1 CC lateral configuration for partial result shifting mode as it requires the same number of operations with less number of CC thanks to the operations overlapping. For a two-template operation to re-use the same hardware for the implementation of both templates, either considering the CC allocated in a single template or distributed in two, is the best option. The third criterion appears from a deeper analysis of the RPO definition and the evidence that cell configuration and template shape matching would provide a best case. We go further in this criterion and we realize a study of template shape through the most representative CNN template library, the CSW. From this study we conclude that most of the gathered CNN operations exhibit a diamond distribution of the template elements and that they are mostly symmetric, what when combined with result shifting, can be used to reduce the number of operations. Operations with just central CC as logic or arithmetic operations between others, are also significant. As a consequence, a 5 CC diamond configuration, i.e. the classical NEWS with a central CC, represents a good trade-off option, what in addition justifies the generally assumed efficiency of the NEWS limited connectivity. The study also analyzes the symmetries and proposes a way of taking advantage of them. The final criterion are, obviously, the goals to be reached in the implementation, that would set the actual limits in processing time and area occupation.

At the end of the chapter we analyze the combination of both LN emulation and hardware simplification. We have seen that it is completely assumable under the combination of LN and HR guidelines. In summary, as the LN emulation demands a significant number of shifts, the cell configuration and LN emulation shift technique should look to each other. The usage of possible symmetries (with result shifting) and two-template configurations are also shown as an advantageous resource.

Finally, we would like to remark that the application of the S&S methodology does not have strict techniques to be applied but guidelines for its application. This means that we can develop different techniques or ways of application with similar results, as better as most adjusted to the particular case.

Chapter 4

Validation

This chapter validates the presented methodology by quantitatively analyzing both their hardware improvements and their processing time penalties. Although we have treated the application of the S&S methodology to LN emulation and to hardware reduction separately, for its validation we consider the methodology globally.

Implementation requirements and time conditions are reviewed in Section 4.1. For the hardware improvements assessment (Section 4.2) we have chosen two general purpose physical CNN implementations whose area data are accessible in the literature. We present as well the results from the utilization of the S&S techniques on CNN FPGA ad-hoc realizations. In a subsequent point (Section 4.3) we evaluate the consequences at number of operations/processing time level of LN S&S techniques with and without hardware reduction over some well-established LN algorithms detailed in the literature. The realization over CPAs of two of these applications, namely the Scale Invariant Feature Transform (SIFT) and the Speed-Up Robust Features (SURF) algorithms, were not previously introduced in the literature by other authors, being another contribution of this thesis. Finally, in Section 4.4, a well-known complex real time algorithm and its physical implementation are deeply analyzed to provide trade-off data.

4.1 Implementation Requirements and Time Conditions

To begin with, it should be emphasized that the S&S methodology can only be used with CPA implementations that provide predictable stable outputs like DTCNNs or continuous time CNNs with B-type templates only (i.e. no feedback template A). In addition, it is apparent that the starting architecture determines the data type to be used. In this sense, for example, architectures that strictly realize the binary 1-bit 1Q CNN model like the one introduced in [Flak et al., 2006c] would need an extra analog memory (LAM) to accumulate partial outcomes from sub-templates. Furthermore, even with an extra analog memory, these architectures are restricted to the S&S image shifting mode as the coefficient circuits are designed to work on binary variables and the results to be shifted are in general G/S values. On the other hand, synchronous gray-scale architectures with cells of the type introduced in [Rodríguez-Vázquez et al., 2004] can easily adopt the S&S methodology as they count on analog memories to store sub-template results and can deal with both binary and gray-scale data.

Another concern is the extra time caused by the extra number of operations. Based on the general initial estimations from Table. 3.1, one can conclude that the highest number of processing steps resultant from applying the S&S methodology to an operation of 2 full dense 3×3 templates with the barest of the configurations (3 coefficient circuits only) is 18 (10 sub-template applications and 8 shifts, being 20, 10&10, if we force the usage of the previous shifted image in all the shifts). This number is sharply cut in actual applications thanks to the shape or symmetries of the templates, common images for A and B templates that allow to share the image shifting, or even the distribution in two templates of the coefficient circuits (when not the implementation of the same or different cell configurations in both templates) that adds the possibility of overlapping S&S steps as it was shown in the previous chapter. Apart from the number of operations, the time required for a processing step depends on the hardware solution. In solutions like [Rodríguez-Vázquez et al., 2004] and [Dudek, 2005], running B-type templates lasts few μs. This time is easy to cut down with today digital CMOS technologies. In fact, in current sub-micron technologies processing steps of less than 100 ns are easily achievable with 1-bit programmable architectures [Flak et al., 2006c, Brea et al., 2006]. These times include the uploading of the templates or instructions from a global memory to the cell array. Keeping all these numbers in mind, and accounting for the image acquisition and the output data downloading times, the designer can judge whether or not the S&S methodology still complies with the time requirements of the application.

4.2 Expected Hardware Improvements Evaluation

As absolute numbers will depend on the particular hardware solution, until here we have assessed the hardware reduction in function of the number of coefficient circuits removed. In order to give some numbers that allow us to evaluate the actual possibilities of the methodology we have gone through two different architectures, one G/S and one B/W, reported in the literature with enough details to allow at least rough estimations of the area savings. These estimations were introduced at the ISCAS07 paper (Appendix A, page 129). At the end of the section we include the conclusions derived from the implementation of a DT-CNNUM over an FPGA by using the S&S.

The G/S architecture is the ACE16K chip discussed in [Rodríguez-Vázquez et al., 2004] and, more detailed, in [Liñán, 2002]. This architecture consists of a 128×128 cell grid implemented with a standard 0.35-μm CMOS technology. Each cell occupies an area of 73.3×75.7 μm^2. From the references we know that the area occupied by the synapses is the 20.25% of the cell area, what means around 1124 μm^2. Every cell counts on 8 multipliers for neighborhood connectivity plus the feedback term and three multipliers for additional inputs. In order to provide higher accuracy, the latter four multipliers are doubled.

Let us apply the S&S reduction to the 8 connectivity CCs reducing their number down to 3. The number of connections removed would also be 5. We suppose that the inter-cell connections are included in the area percentage given and, in absence of further information, we consider that the connectivity area is reduced in the same percentage as that of the multipliers area. Each of the multipliers occupies a 6.25%

(1/16) of the synapses area and so it is reduced in a 31.25% with the 5 CC removing. This means an area saving of 6.3% per cell, which is around 351 μm^2. In a 128 × 128 array this amounts to 5.8 mm^2. Conceptually, and whenever we could operate in a controlled mode, the S&S methodology could be applied as a usual algorithm in the G/S architecture as it counts on several LAMs (8), just taking into account the possible range corrections required to avoid value saturation during S&S application and with no extra hardware. It is also interesting to remember that in continuous time CNNs, as it is the case, the S&S are only applicable to the template B, and that CTCNN operations can be translated to B-template operations following the equivalences shown in the statistical study of the CSW template library gathered at CNNA08 paper (Appendix A, page 141). Further considerations in the S&S application would require deeper knowledge of the particular architecture.

As a reference for implementations where we have to include the LAM, we have estimated that in the ACE16k (0.35-μm CMOS technology) each of the 8 capacitor-LAMs, occupies an area of around 145.7 μm^2, provided that the 8 LAMs available occupy a 21,01% of the cell area [Rodríguez-Vázquez et al., 2004, Liñán, 2002]. From the SCAMP3 chip [Dudek, 2005] we estimate that the analog current memories S^2I used occupies around 156 μm^2, again in a 0.35-μm CMOS technology.

The B/W architecture is the 1-bit programmable approach addressed in [Flak et al., 2006c]. It was implemented with a standard digital 0.18-μm CMOS process. In this case, the cell contains 9 coefficient circuits that occupy an approximate area of 32 μm^2 within the 155 μm^2 of the total cell area. The S&S methodology could reduce the number of multipliers until 3. In area, this means to save 21 μm^2, which is around 14% of the total cell area. In a 128 × 128 array this would mean around 0.35 mm^2. These numbers are obtained from the analysis of the layout provided in the reference. Without further information we also estimate here that cell interconnections are included in the estimated area and their particular area is reduced in the same proportion as that of the CCs. All in all, we should take into account the area to be occupied by, at least, one analog memory to accumulate the partial results that have to be included. This extra hardware may override the S&S area gains obtained over an already reduced area implementation with tiny coefficient circuits. Still, the additional memory is needed when tackling large neighborhood kernels and the hardware reduction helps to minimize the impact of the integration of the required memory.

An interesting alternative for binary DT-CNN implementations is the template partition proposal made by Brugge in [ter Brugge et al., 1998c] and introduced in Chapter 2. This proposal is applicable to binary input-output operations that can be expressed as morphological operations as indicated in [ter Brugge et al., 1998a]. The correspondence between morphological operations and DT-CNNs has been made considering 4-quadrant weightings with no restricted weight values, and they should be translated to 1-quadrant weightings and 1-bit weight values following, for example, the proposals in [Brea et al., 2004a], to apply them in implementations as the one in the example. This would imply two translations, the first one to mathematical morphology, and the second one to the limited range values. A priori, this would imply the decomposition of the original operation in several ones. After the translation we would have LN kernels that can be split in minimum-sized templates to be applied independently and combine their results to emulate the original one as propose the author in reference [ter Brugge et al., 1998c]. As the LN output is binary and it is obtained from unions and/or intersections of the saturated binary partial outputs, the "accumulation" would be also binary and it would not require a LAM. Over the DT-CNN kernels obtained (LN or even the minimum-sized if we want to reduce the area consumption) we can also apply the S&S techniques proposed in this thesis, with the only consideration that now the "accumulation", as being realized through union and intersection operations translated to DT-CNN operations, should be taken into account in the number of operations assessment. To avoid this we should apply the DT-CNN operation equivalent of combination of morphological operations shown in [ter Brugge et al., 1998a].

Finally, we have realized a topographic fine-grain parallel implementation of a DT-CNNUM over an FPGA using the S&S methodology to study the performance of the methodology over this fully digital platform programed through VHDL language, at the same time as we tested the feasibility of actual topographic DTCNN implementations with the help of the S&S methodology to reduce the area occupation. In this first incursion on the FPGA test we chose an Altera Stratix-EP1S25 FPGA to realize a 1Q 1-bit programmable binary implementation, almost reaching a 30×30 effective grid for a 9 CC configuration. Further implementations have been developed for actual applications and they are gathered in Appendix B.

We have realized different tests to analyze the consequences and the convenience of using the S&S on FPGAs. In particular we have analyzed the performance in implementations with different number of CCs in different configurations, including the analysis of the effect of keeping the feedback CC and the effect of the symmetry in the allocation of coefficient circuits. We have also implemented grids of different sizes for two configurations to study the behavior of the implementation with the number of cells implemented and to check if the conclusions are valid for any grid size.

In summary, we have obtained significant area savings by reducing the number of CCs, what leads to the possibility of implementing a larger number of processing elements for a given FPGA. In particular, we have a saving of 14 logic elements (LEs) per cell over 27 when reducing the number of multipliers from 9 to 3 CC. This leads to around half of the area occupation when comparing the 9 CC realization (103% occupation over the Altera Stratix-EP1S25 FPGA selected) with the minimum 3 CC one (55% occupation), taking into account the area occupation increment due to the higher control and dummy cells complexity and the required accumulator to perform the addition of the partial results. In fact, the 9 CC implementation cannot be actually realized over the selected FPGA, and the data are probably underestimated for it, having that with 6 CC with have a 99% occupation. If we choose a 4 CC implementation the cell area is reduced in 10 LEs and the total area saving is reduced to the 30%, but the S&S steps per template are as well reduced from 10 in the 3 CC to 5 in the 4 CC realization. As area savings can be directly translated into the possibility of implementing more cells over the same device, a reduced number of CC can lead to reducing the windowing process when tackling larger images, what partially compensates for the increment in the number of operations, even more if we take into account the highly probable template-configuration election matching for a particular application. It is also worth noting that the extra hardware required to implement the S&S techniques, an accumulator, provides the reduced CC option with the capability of applying 2-template operations or LN kernels, which is not possible in the 9 CC implementation.

Concerning to have or not to have the central coefficient circuit implemented, we conclude that in this implementation the influence of the inter-cell connections is much less significant than the size of the hardware directly related to the number of CC in the PE as the encoder and the number of registers. Finally, regarding symmetry, we only find differences in the area occupied by the dummy cells with better results in the less symmetric configuration. This difference is due to the reduced number of connections on one of the sides of the cell and not to the absence of symmetry.

From the grid size analysis we see that it does not affect the conclusions drawn from a size of 5×5 on, where the CC area reduction starts to compensate for the extra hardware required by the S&S . Only the working frequency and the global memory control complexity, that are not affected by the CC reduction, get worse with the grid size due to larger fan-out of the global lines that deliver the control signal and template values, and to the larger size of the image to be managed respectively.

The details of the implementation as long as the whole experimentation data are gathered in the ECCTD07-1 paper (Appendix A, page 135). These results were successfully applied for a B/W implementation in the DCIS08-1 paper, and for a G/S implementation in the ECCTD09 paper (both in Appendix B, page 155), and their functionality was proved in [Albó and Canals et al., 2010].

4.3 S&S Techniques over Large Neighborhood Reference Algorithms

The processing time is the other side of the trade-off. In general, the lesser the number of multipliers the higher the number of processing steps, and higher in any case if the application counts on LN operations. Nevertheless, we have seen that a good election of techniques and configurations limits the time penalty. All in all, the feasibility of the approach would be determined by the time needs of the application. To get an idea of how the S&S methodology performs we analyze its usage in some well-known algorithms with LN kernels, namely the Spin filters [Yu, 1988], the SIFT [Lowe, 2001], and the SURF [Bay et al., 2008]. Its application to the PLS algorithm [Vilariño et al., 2003] will be also throughly analyzed in the following section within the trade-off analysis.

Spin Filters

The first algorithm comprises a set of directional operators called "Spin Filters". These operations were introduced to remove noise from interferometric patterns without the loss of information caused by blurring [Yu, 1988]. The number of spin filters depends on their neighborhood size, which in turn depends on the pattern fringe curvature, fringe density and noise level. These operators can be expressed as CNN operations, and their CNN implementation has been suggested as a good option to deal with their high computational cost [Vilariño et al., 2007].

In the case of a 7×7 window, a reasonable assumption, we can define 12 directional 7×7 filters [Vilariño et al., 2007]. Fig. 4.1 shows the shape of these filters by shadowing

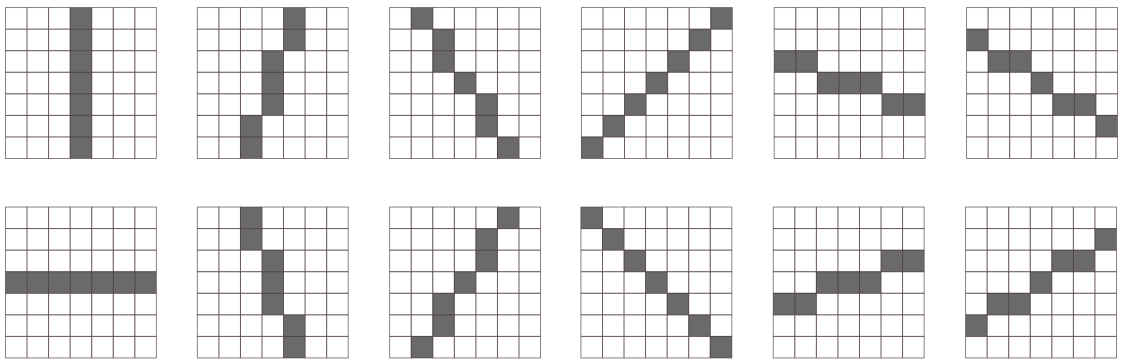


Figure 4.1: Spin filters of dimension 7×7. Non-null positions shadowed.

the corresponding non-null positions. A 9 CC one-template approach will require 7 minimum-sized CNN operations, 3 sub-templates and 4 shifts, per each one of the directional 7×7 filters. This low OIF is feasible thanks to the sparse nature of the directional filters [Vilariño et al., 2007], and it can be cut down to 6 minimum-sized CNN operations with a two-template full 18 CC CNN configuration. As a result, the original 12 7×7 directional filters can be realized with 84 3×3 operations for the 9 CC version and 72 in the case of the 18 CC CNN configuration thanks to the simultaneous application of operations both for image or result shifting mode. The sparse 3×3 resulting templates suggest us that its realization over CNNs with a reduced set of CC could be rewarding. In this case the coefficients cover all the possible template positions along the different directional filters and it is difficult to extract a more convenient configuration. An approach as the diamond configuration with 4 CC, the classical NEWS in an SIMD architecture, leads to 8 to 14 3×3 operations per original 7×7 operation, depending on the particular shape, with a total of 136 3×3 operations for the whole 12 directional filters ($OIF = 1,6$ with respect to the 3×3 LN decomposition). Again, this result can be improved with two implemented templates as for example in a configuration 4+4 CC or even just distributing the 4 CC between them. As an anecdote, in this NEWS case it would be more interesting to keep the horizontal CC in one template and the vertical CC in the other one if we consider output shifting, but if we consider image-shifting it would be more efficient to keep in each template one horizontal and one vertical.

Directional filters were successfully implemented with S&S techniques in [Perfetti et al., 2007] for retinal vessel segmentation. In this case they dealt with 12 directional 15×15 averaging filters. A proper memory usage combined with an adaptation of the S&S techniques addressed in that paper leads to only 92 steps (4 sub-template application and 88 shifts) for the whole process. In particular, they identify 4 different sub-templates to be applied just only once, and then they combine the results into the 12 desired outputs in just 92 S&S steps (4 sub-template application and 88 shifts). It was thought to be applied over the ACE16K [Rodríguez-Vázquez et al., 2004] with a slight memory extension required by the techniques adaptation. This is an example of the translation of the S&S to a particular hardware and application.

Scale Invariant Feature Transform (SIFT)

The SIFT (Scale Invariant Feature Transform) is a method for extracting distinctive scale- and rotation-invariant features that could be used, for example, for matching and object recognition [Lowe, 2001]. This method consists of four stages, of which, the first one, scale-space extrema detection, implies the application of several Gaussian filters of increasing σ values to generate a group of images, known as scales or σ levels, called octave. Depending on the image size, three to four octaves with six images each are usually needed. Every new octave starts from a half resolution image obtained by downscaling with a factor 2 the fourth image from the former octave, leading it from $M \times N$ to $M/2 \times N/2$ pixels.

If we assume 4σ as the order of neighborhood for the Gaussian filters truncation in their approach from the continuous space to discrete kernels, the kernels will be of size $(8\sigma + 1) \times (8\sigma + 1)$. Considering an initial σ value of 1.6, and a factor between sigmas of $2^{1/3}$, the scale space generation will require six kernels per octave of sizes 13×13, 17×17, 21×21, 27×27, 33×33 and 41×41. By using the *S&S* methodology to implement the 6 templates of significant LN required per octave over a CPA we would need 1796 operations (sub-templates application and shifts) with 9 CC. Given that 2D Gaussian filters are separable in horizontal and vertical 1D Gaussian filters (H/V separability), we can consider $N \times 1$ and $1 \times N$ kernels. This drops the number of *S&S* operations to 372 on a CPA with 9 CC or even with a 5 CC Diamond configuration (NEWS plus feedback circuit). This number is further dropped in a CNN approach with a two-template configuration $4 + 4$ CC Diamond, reaching 292 S&S steps.

Until here we have considered that all the σ levels or scales are provided by applying the Gaussian filters on the original image. Nevertheless, if the Gaussian filters are run on the previous filtered image or scale we can reduce the size of the kernels required to: 13×13, 5×5, 5×5, 7×7, 7×7, and 9×9. Together with the *S&S* methodology and the 1D H/V separation this result in 92 and 80 S&S steps in a 5 CC Diamond and a $4 + 4$ CC Diamond implementation respectively. Going further, Gaussian filters belong to the particular case of large neighborhood templates that can be emulated by running a 3×3 kernel recursively. In this case it would be enough with 19 recursions to obtain the 6 images required if we use the previous scale as indicated. In a CPA with a 5 CC Diamond configuration these numbers would be multiplied by 3 (2 with the $4 + 4$ CC Diamond configuration or if we consider the 1D H/V separation).

But this is only valid for the first octave. The reason is that Gaussian kernels do not change across octaves but image resolution does. In the case of a serial processor everything remains the same, but with a smaller number of operations in every new octave as it works over a smaller resolution image. Nevertheless, for massively parallel processing as CPAs with a pixel-per-processor-approach (including RC networks, the natural approach for the diffusion operation [Fernández-Berni and Carmona-Galán, 2009]), to go across octaves is not straightforward. The reason is that, unless considering a reallocation of the selected pixels by reading out, downscaling and writing back the image as stated in [Zarándy and Rekeczky, 2011], every pixel does not interact with its nearest neighbors after the downscaling. For instance, in the second octave the pixel at i, j interacts along E and S directions with the pixels located at $i, j + 2$ and $i + 2, j$. The pixels in between are not used any more. The distance is increased up to $i, j + 4$ and $i + 4, j$ for the third octave and to $i, j + 8$ and $i + 8, j$ for the fourth one due to the successive down-scalings.

On a CPA with a pixel-per-processor-approach we can deal with this situation by expanding the 3×3 Gaussian seed used in the first octave and applying the $S\&S$ methodology to implement the resultant large neighborhood kernels. Eq. (4.2) shows the result of the expansion of a generic 3×3 kernel (Eq. (4.1)) in a 5×5 template caused by the half resolution downscaling required for the second octave generation. The order of neighborhood doubles in each downscaling and thus seeds of 9×9 and 17×17 pixels appear for the third and fourth octaves. These expanded kernels cannot be achieved by the simple 3×3 recursion as the corresponding minimum-sized seed does not exist. This can be tackled, then, by changing the grid topology with every new octave with a mechanism to account for only the pixels of interest. But, it is also possible to avoid this by running the S&S techniques to approach all the seeds of every octave with minimum-sized kernels (3×3) as it was seen in the previous sections, and with a reduced number of operations thanks to the sparse character of the LN seeds to be applied.

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \tag{4.1}$$

$$\begin{bmatrix} a & 0 & b & 0 & c \\ 0 & 0 & 0 & 0 & 0 \\ d & 0 & e & 0 & f \\ 0 & 0 & 0 & 0 & 0 \\ g & 0 & h & 0 & i \end{bmatrix} \tag{4.2}$$

Taking advantage of Gaussian separability and of the sparse character of the new large neighborhood seed template (actually we expand the H and V components of the original minimum-size Gaussian filter) we just require $2N$ S&S operations to emulate the new $N \times N$ seed template ($N > 5$). The sparse 5×5 seed template for second octave (Eq. (4.2)) would require 8 S&S operations regardless 2D or 1D Gaussian filtering. The 9×9 seed in the third octave would require 33 operations in the first case and 18 for the H/V, becoming 65 and 34 respectively for the 17×17 fourth octave kernel.

Each octave requires 19 iterations of the correspondent seed to generate the six demanded scales. We have, then, that the first octave is realized with 19 3×3 operations (38 with H/V), the second octave would require 152 S&S operations for both options, the third octave would need 627 for the 2D Gaussian filtering and 342 for H/V, and the fourth octave would require 1235 steps in the first case and 646 steps in the H/V one. This makes a total of 513 steps for a SIFT process with 3 octaves and 1159 for 4 octaves, in both cases with 9 CC and H/V consideration from the second octave on. A 5 CC Diamond configuration would lead to just 19 additional operations for the first octave requiring the H/V option (for the rest of the octaves the same numbers remain) with a significant reduction in area per PE. Furthermore, a two-template CNN solution with just a central coefficient circuit in the second template to implement the central template element would drop 38 operations at the third and fourth octaves (a total of 76) in both 9 and 5 CC configurations.

Although these numbers might be affordable for SIFT-based video rate applications, given for example $1\mu s$ per operation, they can be slightly improved if we combine the application of the $S\&S$ techniques with Berni's proposal in [Fernández-Berni et al., 2011]. Therein it is said that through appropriate averages we can implement 3×3 kernels with certain symmetries (Eq. (4.3), which corresponds to the symmetries exhibited by the discretized 2D Gaussian filter Eq. (4.5)) with "2×2" kernels (Eq. (4.4), where we have completed the kernel with zeros to mark the central cell, showing a more conventional 3×3 kernel). This is especially advantageous in the SIFT for the seed kernel expansion caused by the image downscaling through octaves as it reduces the expanded kernels sizes. With this method the second octave can be generated through a new 3×3 seed kernel(Eq. (4.7), the third one with a 5×5 kernel (Eq. (4.8), and, if needed, a fourth one would need just a 9×9 seed kernel (Eq. (4.9).

$$\frac{1}{4}\begin{bmatrix} a & a+b & b \\ a+c & a+b+c+d & b+d \\ c & c+d & d \end{bmatrix} \tag{4.3}$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & a & b \\ 0 & c & d \end{bmatrix} \tag{4.4}$$

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{4.5}$$

$$\frac{1}{4}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \tag{4.6}$$

$$\frac{1}{4}\begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} \tag{4.7}$$

$$\frac{1}{4}\begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.8}$$

$$\frac{1}{4}\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.9}$$

Berni's proposal is based on the use of a resistive network (RC grid) where the averaging process comes out naturally, having that after a long enough time all the pixels involved have the same average value. Berni's method starts with two consecutive averaging operations. The first averaging involves groups of 2×2 pixels selected from the upper left corner of the image and is realized straightforwardly. This averaging is supposedly intended to produce a half resolution image through a factor 2 downsampling. For the second averaging every new 2×2 group comprises the pixels of four adjacent groups of 2×2 pixels used during the first averaging process. This procedure obliges to set a proper control mechanism to select the pixels of interest. Berni states that the application of 3×3 kernels like the one shown in Eq. (4.3) over the half resolution image obtained from the first averaged image is equivalent to the application of the "2×2" kernel in Eq. (4.4) over the half resolution image taken from the second averaged one [Fernández-Berni et al., 2011].

In the translation of Berni's proposal to CPAs we eliminate the pixel value replication (it is neither natural nor necessary on CPAs). In so doing, the first averaging step is not necessary and the 3×3 - 2×2 equivalence through the second averaging is fulfilled even considering the original size image. Our CPAs adaptation realizes the second 4-pixel local averaging through the application of the 3×3 kernel shown in Eq. (4.10). Thus, the result of applying the template in Eq. (4.3) over the original image is the same as that of applying the one in Eq. (4.4) over the averaged image. Coming back to the SIFT, in the downscaling process across octaves the averaging kernel is also expanded. Both, averaging (Eq. (4.10)) and "2×2" equivalent (Eq. (4.6)) kernels, are equal to each other through octaves from their expansion in the second octave on (Eq. (4.7), Eq. (4.8) and Eq. (4.9)).

$$\frac{1}{4}\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.10}$$

The combination of S&S methodology with our CPA adaptation of Berni's proposal yields a total of 988 operations for 4 octaves (380 for three octaves) in a 9 CC configuration. This number is increased up to 1064 (456 for three octaves) if we consider a 5 CC Diamond configuration. Slightly better results are met if we combine the H/V option for the first octave and Berni's for the rest, being 969 (361) and 1026 (418)

Table 4.1: Number of S&S operations for scale space generation in SIFT

Cell Conf. (Opt.)	3 oct.	4 oct.	Cell Conf. (Opt.)	3 oct.	4 oct.
9CC (3×3 + H/V*)	513	1159	5CC (H/V)	532	1178
9CC (Berni**)	380	988	5CC (Berni)	456	1064
9CC (H/V + Berni)	361	969	5CC (H/V + Berni)	418	1026

* H/V separability

** [Fernández-Berni et al., 2011]

respectively. In the case of having an implementation very demanding in area occupation we can even resort to a 3 CC configuration which implies 836 S&S operations for three octaves and 1748 for four octaves applying Berni's proposal with the CPA adaptation in all of them. These results do not depend on the image resolution if we have a pixel per processor correspondence.

Table 4.1 summarizes the number of S&S operations corresponding to the main of the presented options, always considering seed recursion and previous scale utilization. At the sight of the results we conclude that we can implement the SIFT scale-space generation over CPAs with very acceptable results by making use of the S&S . For instance, if the clock cycle was only 1 MHz, as is the case of the implementations reported in [Dudek and Hicks, 2005, Rodríguez-Vázquez et al., 2004], and each *S&S* operation takes one cycle, the scale-space generation would be ready in ~1 ms, leaving a relatively long time for the rest of operations, which might be enough for video rate processing. Best results are achieved if we combine the H/V Gaussian filter decomposition in the first octave with Berni's proposal adapted to CPAs for the rest of octaves. Little worsening is obtained if we consider a reduced configuration (5 CC Diamond, almost half of the CC). Note that it could be convenient to implement Berni's in all the octaves, with little worse numbers, in order to have 1-bit coefficient values.

In considering at the same time the H/V separability characteristic and Berni's proposal, we can decompose the Gaussian kernel in four kernels with one of their dimensions equal to 1, as shown in Eq. 4.11. In so doing, we just need a 5 CC NEWS configuration, and the requirements over the memories are reduced as the results are re-used more frequently. The number of operations is the same as that obtained for a 5 CC NEWS configuration with the only application of Berni's proposal.

$$\frac{1}{4}\begin{bmatrix} ax & (a+c)x & cx \\ a(x+z) & (a+c)(x+z) & c(x+z) \\ az & (a+c)z & cz \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} * \frac{1}{2}\begin{bmatrix} 1 & 1 & 0 \end{bmatrix} * \begin{bmatrix} 0 \\ x \\ z \end{bmatrix} * \begin{bmatrix} 0 & a & c \end{bmatrix} \tag{4.11}$$

Finally, and as a matter of fact, we state here that RC networks are the most efficient approach to the diffusion filtering, outperforming every CPA implementation [Fernández-Berni and Carmona-Galán, 2009]. Nevertheless, a CPA approach permits to have more functionality per PE/cell, which might be beneficial on a monolithic solution.

The application of the S&S methodology to the SIFT's scale space generation was introduced in the ISCAS12 paper (Appendix A, page 149). We point here the inconsistence in the numbers given as totals for three and four octaves prior to the application of Berni's proposal, 1159 and 1805 respectively in the paper. There was a mistake in the data introduced in the paper due to the double addition of the number of operations corresponding to the fourth octave. In addition, it should be indicated that those numbers (the correct ones, 513 and 1159) are obtained when considering H/V separability from second octave on and considering the full dense 3×3 in the first octave coherently with the 9 CC configuration selected in that case. In fact, the H/V separability consideration for the four octaves leads to 19 additional operations, the same as considering a 5 CC NEWS + central CC configuration. Nevertheless, these numbers are correctly shown in the Table I of the same paper.

Speeded-Up Robust Features (SURF)

The SURF (Speeded-Up Robust Features) [Bay et al., 2008] is another state-of-the-art computer vision algorithm implementing a scale- and rotation-invariant detector and descriptor. The algorithm consists of three main steps: the interest point detection, description and matching. We focus on the low-level image processing stage, the first one, that involves LN kernels.

As in SIFT, the interest points in SURF need to be found at different scales, images produced by convolving the original image with increasing size filters (increasing Gaussian σ values). The scale space is again divided into octaves that now are generated by up-scaling the filter size instead of iteratively reducing the image resolution as in SIFT. In this case, such filters are 2-D Gaussian second order derivatives along horizontal (xx), vertical (yy) and diagonal ($xy = yx$) directions that conform the Hessian matrix. These Gaussian second order partial derivatives have to be discretized and cropped for practical reasons with slight decrease in performance. A further approximation as "box filters", where the kernels are simplified to rectangular areas with a common weighting value within each region (0, 1, -1, or -2, in this case), provides similar or better performance. The 9×9 filter is considered the initial scale, and its size is increased in 6 pixels on each dimension in the first octave until having the four filtered images per octave required by the algorithm. For the first octave we have, then, filters of sizes 9×9, 15×15, 21×21 and 27×27 pixels. Each new octave begins at the second filter of the previous octave, and the neighborhood order difference between successive filters doubles with respect to the previous octave (e.g. filters in the second octave will be of sizes 15×15, 27×27, 39×39, etc.) [Bay et al., 2008]. According to the image size three or four octaves can be needed, yielding filters up to 195×195 pixels (four octaves).

Nevertheless, these filters application imply a high computational burden in a serial processor. The same happens with the S&S techniques application over a massively parallel processor as a CPA approach. Actually, this option would require around 65000 operations with the box filters approximation, which would put a real-time application in jeopardy unless the clock cycles were very short. Nonetheless, box filters are especially interesting when combined with the well known "integral image" [Viola and Jones, 2001] that, once computed, reduces the summation of any size rectangular area in an image to the four corner pixel summation ($Sum = Down_{Right} - Down_{Left} - Up_{Right} + Up_{Left}$). The integral image (II) is an intermediate image representation which gives each pixel the value of the summation of all the pixels from itself to the left and above in the original image. The box filters consist of 3 or 4 rectangular areas where the pixels have to be summed, and the results are weighted and combined. The integral image reduces the box filters application to 11 or 15 additions each (3 per rectangular area plus 2 or 3 for the areas combination depending on the box filter shape), regardless the filter size. The difficulty now roots in the integral image computational burden. Reference [Viola and Jones, 2001] introduces recurrences to avoid redundant operations. With this the number of additions is reduced to $2NM$ for an $N \times M$ image.

CPAs can parallelize the II computation reaching an order of $N + M$ steps in an image of resolution of $N \times M$ pixels and a total number of additions of around N^2M. This represents 256 CPA operations for a 128×128 image or 1120 in a 640×480 VGA one, for example. These are affordable numbers for real-time implementations. Fig. 4.2 illustrates the II calculation on a CPA through an ad-hoc S&S technique. In this case we reduce the number of sub-templates to one that adds the N, W and NW pixels to the central one (D1 in Fig. 4.2). Afterwards, the partial outputs are, first, horizontally gathered in the corresponding pixels by shifting the obtained image two shifts to the right (S1 template) repetitively, and by accumulating the successive shifted versions of the image. When the horizontal accumulation has finished we have two rows of the II calculated. The rest of the rows are obtained by repeating the same process in the vertical-down direction (S2 template) but now the shifted image is the one obtained from the horizontal accumulation. At the sight of the operations required, we can use a reduced 4 CC Diamond configuration (NEWS) by just implementing the D1 operation in three steps as is shown in the lower part of Fig. 4.2. In fact, if not required by other different operations, we could reduce the cell configuration to only the three upper CCs. I_a identifies the image obtained through any of the ways, which is the starting point for the horizontal shifting.

It is interesting to note that this is just one of the possibilities of implementing the II with CPAs. We have analyzed several different possible ways with similar results in number of CPA operations. We can, as well, directly parallelize the proposal in [Viola and Jones, 2001] by accumulating the pixel values in each row in a column fashion way and, once this row cumulative image is calculated (S in [Viola and Jones, 2001]), calculating the II in a row fashion way by accumulating the values of the pixels in the columns. It requires, again, $N + M$ CPA operations. This computation makes most of the hardware idle during the whole process (just one row/column working at a time), which makes us think of an SIMD with lesser degree of parallelism than a pixel-per-processor CPA. This might be even necessary as the integral images yields very wide words, leading to PEs with a larger area. The approach reported in [Ehsan and McDonald-Maier, 2009] reduces to 21 or even 19 bits the word length needed by SURF. Note this is not a constraint in a modern microprocessor with words of 64 bits, but it makes hard, if not impossible, to think of an analog solution with a pixel-per-processor assignment on a CPA architecture. Proposals as the Linear Processor Array (LPA) Xetal-II in [Pu et al., 2011] are promising. If its 16-bit words would suffice for SURF, their 320 16-bit PEs working at 125 MHz would lead to 560 steps for the integral image calculation in less than 5 μs for a QVGA image, under the assumption that memory accesses do not limit the processing time. In addition, the sparse shape of the box

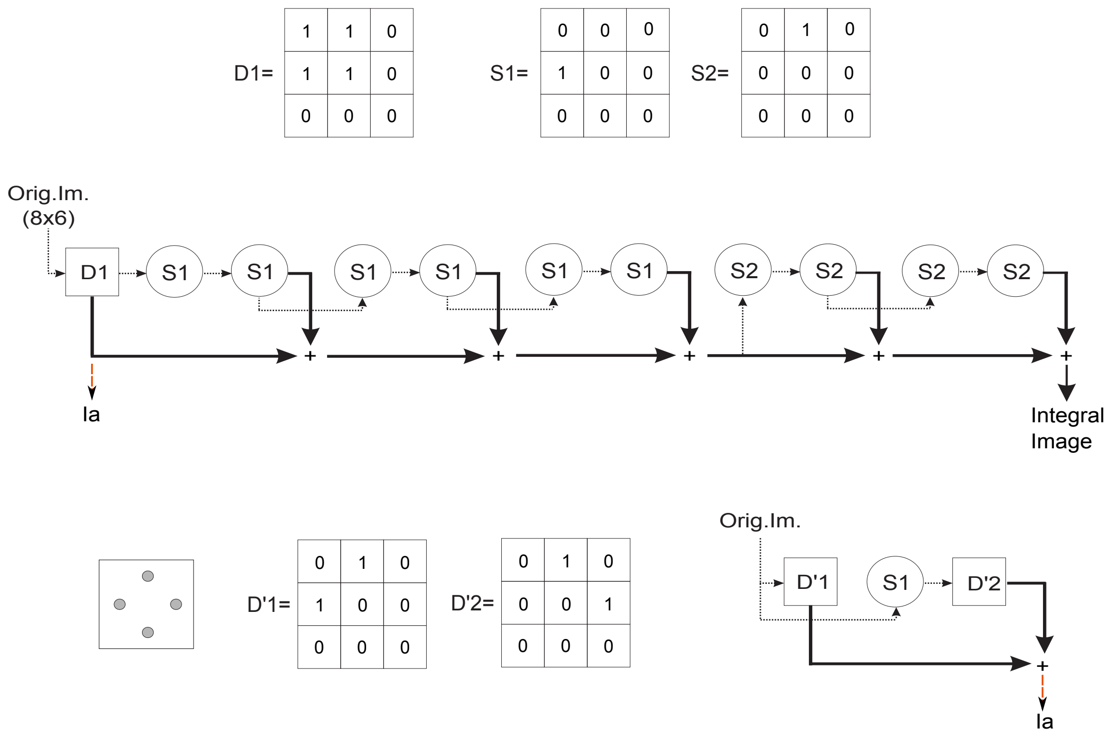


Figure 4.2: CPA 8 × 6 integral image calculation with 9 CC and 4 CC.

filters as they are applied to the II are also advantageous in this kind of architectures under the assumption of freely acceding any line of the image. This lets us think that further improvements could make it possible an LPA for the II calculation, and even SURF scale space generation.

Once we have the integral image we can apply the box filters to generate the scale-space of the SURF algorithm. Although box filters require the same number of operations on a serial processor independently of the kernel size, this does not hold for a CPA as, in the absence of global operations, the four pixels summation is implemented through sparse LN kernels. Being $Q \times Q$ the box filter size, we would require $5Q/4$ S&S operations for the xx or yy Hessian matrix elements and $4Q/3$ for xy/yx ones. Again, it would be enough with a NEWS (4 CC Diamond) configuration. If we can deal with the data size problem the whole scale space generation for an $N \times M$ image would take $N + M + 3184$ CPA operations for four octaves and $N + M + 1584$ for three octaves. If we would consider a 2-template implementation we would have $N+M+3044$ and $N + M + 1472$ respectively. A further hardware reduction (3 CC) would require $N + M + 4294$ CPA operations for four octaves and $N + M + 2128$ for three. These numbers are summarized in Table 4.2. Note that we are always considering that we have an $N \times M$ physically implemented grid and that windowing is not necessary, what is not always possible and is even more difficult with the data size requirements of the II calculation.

The SURF's scale space generation over CPAs with the aid of the S&S method-

Table 4.2: Number of *S&S* operations for scale space generation in SURF

Cell Config. (Approach)	**3 oct.**	**4 oct.**
(9 CC or 4 CC)	$N + M + 1584$	$N + M + 3185$
(9 CC or 4 CC - 2 temp.)	$N + M + 1472$	$N + M + 3044$
(3 CC)	$N + M + 2128$	$N + M + 4294$

ology was also introduced in the ISCAS07 paper (Appendix A, page 129). We should note that it uses the data from the Xetal-II LPA to give a global estimation of the time required for the scale-space generation but using the number of operations required on a CPA with pixel-per-processor correspondence for the box filters application. A more in-depth analysis of the template application process on the LPA would be required to give a more accurate estimation. In fact, Xetal-II LPA is used to implement LN kernels and it could take advantage of the sparse shape of the box filters as applied to the II without the application of the S&S techniques to them.

4.4 S&S Area-Processing time Trade-off Evaluation: a Real-Time Algorithm Implementation

To comparatively evaluate the area and time efficiency of the S&S we use the Pixel Level Snakes (PLS) algorithm version addressed in [Vilariño et al., 2003]. The PLS is an active contour-based algorithm mainly oriented to real-time contour tracking and segmentation. Its development at pixel level makes it very suitable for CPA implementations. Regarding the processing data type, PLS contains differentiated modules for gray-scale and B/W tasks. The gray-scale module extracts the guiding information for the contours from the original input image. Afterwards, contours are moved and deformed according to the guiding information by the B/W modules. In this section we analyze the area-processing time trade-off when applying the S&S methodology to the B/W modules of the PLS. The great variety of operations along with their high time-consuming nature (propagative and LN templates included) make PLS an appropriate real-time benchmark for our methodology.

We will address the algorithm version discussed in [Vilariño et al., 2003] under its implementation in [Brea et al., 2006]. B/W processing comprises morphological operations like erosion and dilation, logical functions (AND, OR), propagative templates like hole filling, large neighborhood operators like diffusion, and some other specific hit-and-miss operations. Fig. 3 and 4 in DCIS06 paper (Appendix A, page 121) show a scheme of the modules and the corresponding templates involved in the algorithm version we analyze. Nevertheless, for us, the internal potential extraction recovers its original form as a diffusion operation ([Vilariño et al., 2003]), instead of the four B/W operation approach given in [Brea et al., 2006]. We have also considered an estimation of the external potential proposed by the author of the algorithm in an internal report [Vilariño, 2005] consisting of a subtraction, a threshold, and an open/close for noise removal; and an edge detection, 15 dilations, and a 3×3 diffusion.

Regarding the physical implementation, the B/W modules are realized over a synchronous 1Q binary CNN architecture with 1-bit of programmability. It was implemented with a standard digital 0.18-μm CMOS process and each PE occupies an area of $40 \times 32\mu m^2$. The gray-scale block is implemented with specific non CNN type hardware that is kept and considered in the cell area without modification. The B/W cell core implements two templates, one of them with 9 possible non-null coefficients, and the other one with the central coefficient as the only non-zero element. Thus, the initial number of CC is 10 [Brea et al., 2006]. This is coherent with the two types of CNN operations comprised in the algorithm, either one-template operations or two-template AND/OR logical operations just requiring the central elements of both templates. The diffusion operation is allowed in this 1Q-1bit-BW architecture thanks to considering a homogeneous version of the operation (all template elements set to one) and to the S&S methodology for the LN version application. The actual usage of this alternative (homogeneous kernel) should be carefully analyzed taking into account the effect of non considering a progressive decreasing in the weighting values with the neighborhood order.

The Topological Transformation (TP) Module

Prior to the complete trade-off analysis we will use a PLS module to illustrate the analysis process. We choose the Topological Transformation (TP) module because this is the most time-consuming module in the PLS due to the hole-filling propagative task. Apart from the hole-filling, TP comprises an opening (erosion & dilation) and a binary edge detection. The S&S implementation of this module was used in CNNA06 (Appendix A, page 113) to illustrate the performance of the methodology in complex algorithms where, taking into account the particular shapes of the algorithm templates, we can achieve RPOs larger than 100% or even infinity.

First of all we have to analyze the restrictions of the original implementation and the characteristics and requirements of the involved operations. On the first aspect, the starting point implementation, [Brea et al., 2005b], is a complete binary architecture and this imposes a fundamental constraint to the processed data type. In this case the use of image-shifting is mandatory as partial-result shifting would require to feedback gray-scale images to the binary multipliers. On the second issue, this module realization implies two types of CNN operations, one consisting of one template with 5 possible non-zero coefficients, and another one devoted to logical AND/OR operations and consisting of two templates with only one non-zero template coefficient each, the central ones.

Fig. 4.3 displays the TP module operations implemented in the B/W architecture presented in [Brea et al., 2006]. It should be noted that in our synchronous architecture, and differently from a classical continuous-time CNN, A and B are interchangeable templates. A simple visual inspection reveals that it is enough to have a 4-connected NEWS configuration with central coefficient in one of the templates and only the central term in the other one. This would lead to 6 coefficients circuits ($5 + 1$ configuration). This area improvement comes without penalty at processing time, which means an $RPO \rightarrow \infty$.

A further analysis shows that simultaneous running of the two templates is just used to perform Boolean operations. It is possible, then, to choose a 4+1 configuration and execute the logical functions in two steps (we do not have the central CC in the

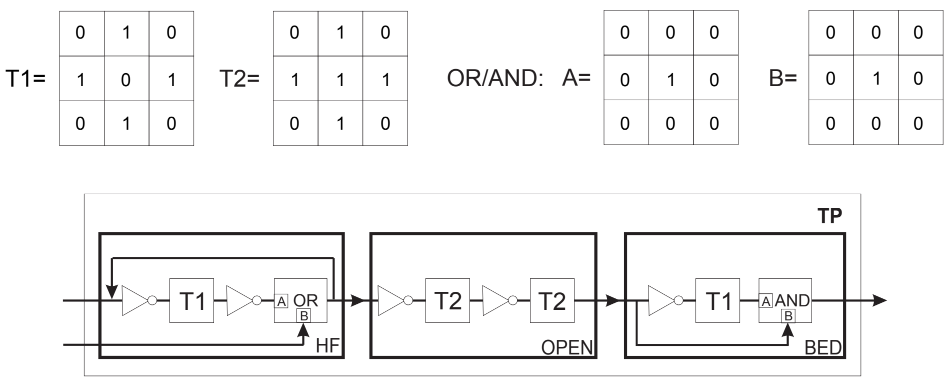


Figure 4.3: Operations and templates in the TP module of the PLS algorithm reported in [Vilariño et al., 2003] with the implementation presented in [Brea et al., 2006].

first template). In this case *RPO* drops to around 150-100% depending on the number of hole-filling (HF) iterations required (150% for just one HF iteration and 100% for a large number of iterations). Furthermore, seeking bigger area improvements, we can select an option with two templates in a 3+1 configuration. This is the situation illustrated in Fig. 4.4. The cell configuration is represented with dots for the A template coefficient circuits and with crosses for the B ones. Sub-templates, shift templates, and system level processing steps to emulate the original CNN operations functionality are also shown. This configuration would lead to an *RPO* between 45 and 60% (45% for just one HF iteration and 60% for a large number of iterations) with an HR of 6/10 (60%) with respect to our starting point.

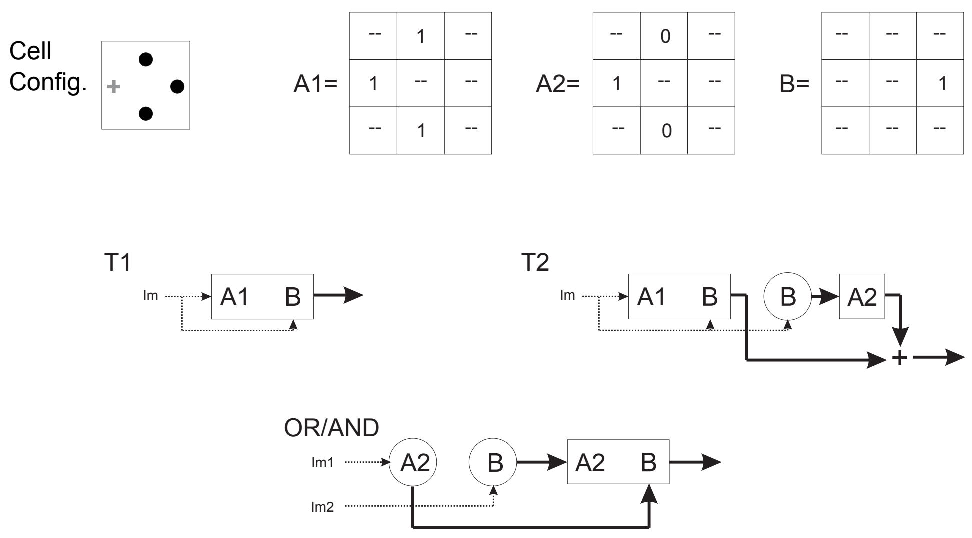


Figure 4.4: CNN operations in the TP module of a PLS algorithm implemented with a 3+1 CC configuration. Shifts appear circled and sub-template applications squared.

PLS S&S Trade-off Analysis

After the process illustration we analyze the whole algorithm to offer an appropriate and complete analysis. The results are summarized in Table 4.3.

For the selection of the cells' configurations we have followed the criteria indicated in the previous chapter. To begin with, we have chosen cell configurations respecting the full functionality of the cell and we have also analyzed the shape of the involved templates, concluding, as in the general study, that diamond configurations with feedback CCs are the most adequate. In a first rough performance evaluation of the selected cell configurations, we have chosen those configurations that starting from the diamond general shape require less number of operations for the same number of remaining CC. Note that even the selected 3 CC configurations follow the diamond shape as much as possible. A especial mention is deserved by the 4+1 configuration. This configuration takes as basis an allowed 3 CC configuration and incorporates both template central CCs. We have included this configuration because of the incidence of operations involving just central CCs and operations involving very sparse templates with just one CC, which suggests us that a very sparse configuration including feedback CCs could be interesting. In addition, in the PLS original implementation the area occupied by one CC is significantly smaller than that occupied by one connection and this makes a PLS a good candidate to perform better in a two-central-CC configuration when compared with a one-less-CC with no centrals configuration as the 3+1(D) in this case. This is the particular behavior in area observed in the analysis of HR definitions in Section 3.3. The selected configurations are shown in the first column of Table 4.3. For 3 CC we show two different configurations, with the CCs in just one template, and with them distributed in two, to illustrate the convenience of two templates, especially indicated in this case to execute pixel-to-pixel logical and arithmetic functions. In the analysis we also include for comparison the starting point configuration, a two-template implementation with 9+1 CC.

The second column in Table 4.3 lists the number of operations per frame needed to implement PLS with each configuration. In this evaluation we account for both the B/W and the initial gray-scale task. The latter is accounted as one time equivalent B/W CNN operation [Brea et al., 2006]. Ten iterations in the four cardinal directions (40 iterations per frame) were assumed. This number is high enough for applications like surveillance [Brea et al., 2006]. The total number of operations for B/W processing is calculated under the consideration of worst case for the hole-filling in a 128×128 image (64 iterations). This task is carried out twice in PLS [Vilariño et al., 2003]. The number of operations (processing steps) per frame also varies with the size of the diffusion operator. Table 4.3 gives numbers for two different orders of neighborhood, namely 3×3 and 9×9. The 9×9 is implemented through the S&S techniques over the configurations selected. The number of operations slightly increases with this order of neighborhood. The number of additional operations dedicated to the LN implementation of the 9×9 diffusion template is the same for the 6, 5, and 4 CC configurations (2040 in 10 iterations of the four cardinal directions) and around 25% more for the 3 CC configurations (2640). It is interesting to note the number of operations obtained for the 4+1 configuration, which performs even slightly better than the 4+0 (D) cell configuration with the classical NEWS connectivity thanks to the central CCs availability although it exhibits a worse shape for the general case.

Needless to say that shift-sharing is applied when convenient.

As we have selected S&S image shifting mode, we did not expect significant extra advantages of distributing the CC in two templates out of logic operations, and related to the simultaneous application of operations. Nonetheless, we have found several examples in this algorithm that take advantage of this distribution. For example, in operations involving two templates with one non-null element (central or not), the 3+1 (D) configuration performs better than the classical 4+0 (D). In particular, it saves one step in 9 of the 12 operations involved in the algorithm, including logic operations, leading to a 20% less of operations. Another example is the dense 3×3 diffusion operation emulation in the 2+1 CC configuration, where we can apply simultaneously two operations involving 1 CC each, and reduce the number of steps from 9 to 8 because of the irregular shape of the configuration. This is shown in the 3 CC configurations, where the two-template 2+1 CC configuration performs better than the one with all the CC in only one template. The case of the 5 CC (D) and 4+1(D) configurations is different as they just differ in the template allocation of the feedback CC, resulting in the same number of operations. We have selected the simpler one, the 5 CC (D).

Hardware reduction (HR) factor is calculated by two ways: the simplified (the initial definition) and the weighted way. The weighted version of the HR factor takes into account that the CCs occupy a 20% of the reducible area and the inter-cell connections the 80%. Differences are not very significant and they show the overestimated and underestimated cases, depending on the CC-connection correspondence in the removed CC. The only case where they coincide is the 5+1 (D) configuration, where the percentage of area saved in the connections area is equal to the one saved in the CCs area as they are both reduced to the half. A notable difference is the obtained for the 4+1 configuration, that actually outperforms the reduction obtained for the one-less-CC configuration 3+1(D), confirming the irregular behavior expected for two-central-CC configurations with this particular area distribution. At the bottom of the table cells we include as well the estimated absolute value of total area saving (CA_{reduc}) accounting for the reduction of CCs and connections separately. These numbers are calculated from the data gathered in the PLS cell layout shown in [Brea et al., 2006] related to the actual cell area ($40 \times 32 \mu m^2$). From this we consider that the B/W blocks occupy around a 56% of the total cell area, the CCs+connections occupy the 60% of the B/W area, and this is divided in 20% for CCs and 80% for connections. The hardware reduction entails savings in the total cell area from around 16% for the 5+1 (D) configuration to more than 21% for the 3 CC configurations, implying area savings between 205 and 275 μm^2 per cell. For a 128×128 cells grid, area savings up to 4.5 mm^2 in the 21 mm^2 approximate original area, are expected. Note that we have not taken into account the possible differences in area between considering a one-template or a two-template configuration. We have neither accounted for the extra LAM to be implemented for the S&S methodology application, as the original implementation have just binary memories. Nevertheless, taking into account the room estimated for two kinds of LAMs in the second section of this chapter (around 150 μm^2 for both in a $0.35 \mu m$ technology), we expect area savings even after the LAM inclusion (note that as in this case the images are binary we just require one LAM for the accumulation of the partial results in the S&S application). Interesting remark is the small area for the coefficient circuits as we have considered a 1Q-1bit-BW 10 CC implementation.

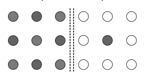
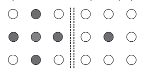
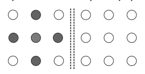
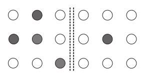
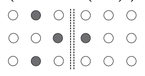
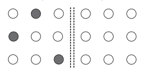
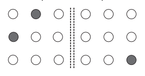
Table 4.3 also outlines the operations increment factor (OIF). It is apparent that the smaller the number of coefficient circuits, the higher the OIF, which is a prove of the good matching between the cell configurations selected and the involved templates shape. RPO formulated as Eq. (3.6) accounts for the area-time (HR-OIF) trade-off in the fifth column. The substitution of the original HR factor for the averaged version introduces slight modifications in the RPO values that are shown between brackets. Note the expected comparison difference between 4+1 and 3+1(D) configurations.

After leaving the most efficient configurations of each size we have to look at the RPO value to select the configuration that offers the best trade-off. As expected, the best allocation of the coefficient circuits is given by the templates shape. In this case, the use of 6 CC instead of 10 barely penalizes the time performance, what means that most of the PLS operations are doable with only 6 coefficient circuits in such a configuration. And so, there is a big area saving with a very small penalty in time, therefore RPO is quite high. The high incidence of the operations involving just the central CCs leads the two-central CC configurations to a better time performance, even for the 4+1 configuration that does not fully implement the diamond preferred shape. Moreover, we can see the importance of having the coefficient circuits divided in two templates (see the case of 3 CC). Also, and although not shown in Table 4.3, the largest OIF in this case for an individual template emulation is 8. This case corresponds to configurations with 3 coefficient circuits in two templates when approaching the 3×3 diffusion operator. It is interesting to say that this factor drops to 2.6 if we choose a 9×9 diffusion template (the reference in this case is the S&S implementation of the 9×9 over the 9+1 CC configuration). The reason is a more efficient use of the multipliers in the combination with LN techniques. In that case the maximum OIF is 5 and corresponds with AND/OR operations.

Because of the real-time characteristic of the algorithm, and following the goal criterion, we include a time performance evaluation for every configuration. We consider 100 ns for every processing step [Brea et al., 2006], and 0.1 ms for image uploading and downloading purposes [Brea et al., 2004a]. These times are given in order to estimate how many processing steps we can have to still meet the time needs of the application (more than 390 000 for 25 fr/s in this case, which means a maximum OIF larger than 30). In this sense, it should be noted that the acquisition time is variable and depends on the sensor implementation, the application and the scene. The time for downloading of the output image is short, as the final output is a B/W image (the contours). The numbers presented in the sixth column of Table 4.3 are easily redefined if these times change. For instance, if the combination of uploading and downloading times is 1 ms instead of 0.1 ms, the number 1.2 ms/fr would become 2.1 ms/fr as it is just the addition of the up-downloading time and the processing times. At the sight of the results we conclude that video frame rates are achievable even for the barest configurations. This is also true assuming processing steps of μs, as in the architecture in [Rodríguez-Vázquez et al., 2004], up to 4 coefficient circuits. This implies that we could try to fulfill more exigent area requirements without penalizing the goals achievement.

Finally, it is interesting to note that most of the time is consumed in executing propagative tasks, hole-filling in this case, what is shown in the last column of Table 4.3 in both, operations per frame and percentage of the total number of operations. This suggests that applications without propagative operations or propagative operations

Table 4.3: Area and time analysis of PLS with S&S.

Number of CC	**Ops/fr**	**HR**	**OIF**	**RPO**	**Time Perf.**	**Propag.**
(Config.)	3×3	$\mathbf{HR}_{weight}$			(ms/fr)/(fr/s)	**Tasks**
	9×9	$\mathbf{CA}_{reduc}$				(Ops/fr)
10 Coeffs.		0 %				10240
(9+1)	11065	0 %	1	0 %	1.21 / 826	92%
	12185	0 μm^2	1	0 %	1.32 / 758	84%
6 Coeffs.		40 %				10240
(5+1 (D))	11309	48 %	1.022	1814 (2182) %	1.23 / 813	90%
	13389	205 μm^2	1.099	405 (485) %	1.44 / 694	77%
5 Coeffs.		50 %				15360
(5+0 (D))	16749	50 %	1.513	97 (97) %	1.77 / 565	92%
	18829	215 μm^2	1.545	92 (92) %	1.98 / 505	82%
5 Coeffs.		50 %				20480
(4+1)	22094	60 %	1.997	50 (60) %	3.21 / 311	92.7%
	24459	258 μm^2	2.007	50 (60) %	3.45 / 290	83.7%
4 Coeffs.		60 %				20480
(3+1 (D))	22675	52 %	2.049	57 (50) %	2.37 / 422	90%
	24755	224 μm^2	2.031	58 (50) %	2.58 / 388	83%
3 Coeffs.		70 %				46080
(3+0)	49720	64 %	4.493	20 (18) %	5.07 / 197	94%
	52360	275 μm^2	4.297	21 (19) %	5.34 / 187	88%
3 Coeffs.		70 %				40960
(2+1)	44326	64 %	4.006	23 (21) %	4.53 / 221	92%
	47006	275 μm^2	3.858	24 (22) %	4.80 / 208	87%

in sparser templates could reach higher frame rates. Moreover, higher frame rates can be possible with specific hardware like local logic units.

In the CNN literature there are many applications with moderate to low time needs (tens to hundreds of frames per second) comprising lots of B/W operations of the type of those found in PLS [Xavier-de Souza et al., 2006, Szlavik et al., 2006, Malki et al., 2006]. In all these cases, the S&S methodology might lead to area and time efficient solutions. Applications with time goals in the order of tens of thousands of frames per second like those addressed in [Zarándy et al., 2005] might also be solved with S&S. If the acquisition time is around $1\mu s$, there is still a slot of tens of μs to process and deliver the output image. The implementation of a reduced set of CC combined with a fast architecture leading to short processing steps (tens of nanoseconds) might be good enough to keep pace with the requirements of tens of thousands of frames per second. In the end, the application time needs determines whether or not our approach is advantageous.

The analysis of the PLS implementation is gathered at ISCAS07 paper (Appendix A, page 129). Nevertheless, in the data shown in that paper we do no account for the inter-cell routing and the analog memory room, as it was accounted here. In addition, accounting for both the CC and area connections led here to the utilization of the weighted HR and to the inclusion of the 4+1 additional configuration with two central CCs in the analysis. At DCIS06 paper (Appendix A, page 121) we can find an schema of the main PLS tasks and the corresponding templates used in [Brea et al., 2006]. It also shows the S&S steps required for the emulation of the PLS operations over a 5+1 CC diamond configuration, and illustrates in detail the process for the application of the LN homogeneous diffusion template considered over the same configuration. It is interesting to note that the emulation process of this template would be exactly the same as if considering a 4 CC diamond configuration. Note, nevertheless, that the numbers given about the area savings also differ from the data given in this section. The main reason is the consideration of the original HR over the area occupied by both CCs and inter-cell connections. Additionally, the estimates of the percentage of occupation were rough and they turned out to be overestimated.

4.5 Summary and Conclusions

In this chapter we have gone through several algorithms and physical implementations in order to validate the proposed methodology by analyzing the coherence between the obtained and the expected results.

The main criterion used for the election of the physical implementations has been the availability of data and characteristics of the implementation. In addition, we look for covering both G/S and B/W data type. And finally, although the methodology is general enough to be applicable to any CPA complying with the predictable and accessible results condition, as we have developed the methodology over the CNN paradigm we have looked for the physical implementations within the CNN literature. One extra CPA non-CNN implementation, the SCAMP-3, was also analyzed to provide an estimation of the S^2I memories used in it. Finally we have realized a study of the performance of the methodology over a CNN implementation on an FPGA platform.

From the analysis of physical implementations we conclude that, as expected, the application of the hardware reduction is much more profitable in G/S solutions where the CC implemented are in general bigger and where the analog local memory is usually included. Nevertheless, the hardware reduction S&S techniques can be used to compensate for the area occupied by the extra LAM required by a binary implementation if we choose to provide it with LN functionality through the S&S methodology.

On the other side, we have chosen examples within the state-of-the-art low level image processing algorithms, and looking for LN communications. In this case we have not limited us to the CNN algorithms, as it can be seen in the SURF and SIFT election that had not been previously implemented over CPAs.

The first conclusion drawn from the S&S application to these algorithms is that they can be implemented over locally connected CPAs despite their needs of large neighborhood operations. Results are even promising, estimating that the SIFT scale space generation can take $\sim$1 ms with around 1000 3×3 operations for four scales in a 5 CC NEWS configuration, and that the whole application of 12 7×7 spin filters are realized with a total of 136 3×3 operations in a 4CC NEWS configuration. The case of the SURF is a bit different. The implementation of the integral image over CPAs leads to the parallelization of its calculation, what has been looked for in the reference literature. Nevertheless, due to the particularity of the integral image definition, the parallelization is limited to one line at a time. This leads us to propose the utilization of LPAs instead of CPAs, because, in addition, their lower number of PE allows the implementation of larger memories, what is a requirement of the integral image. In this first part the methodology is almost reduced to shifts and accumulations with the exception of the initial sub-template application that reduces the number of operations required. The second part of the SURF scale-space generation involves the box-filters application, that can be considered as proper LN operations. Together with the integral image, they are reduced to some additions of values occupying large distance positions, that can be implemented with the S&S techniques over a CPA. The box filters application can takes around 3000 3×3 operations for four octaves. In both cases for a full-dense or a 5 CC NEWS configuration, being another example of the inefficiency of having implemented a full dense template.

Of course, no biased criteria looking for a sort of better fit to the methodology was used in the implementations or algorithms test election and we have not discarded physical implementations or algorithms because of better or worse results. In this sense we have realized a blind selection as we could not have predicted the particular results prior to the algorithms analysis out of the general predictions of the methodology.

The whole trade-off analysis was realized over an application oriented implementation of the PLS algorithm. Both the algorithm and the implementation are well known for us as they have been developed within our research group. In fact, this implementation and its adaptation to LN operations was the initial motivation of this work research. At the sight of the results we observe that with the methodology proposed we can not only enlarge the functionality of the proposal by allowing LN operations, but that area improvements can be expected even after the introduction of the required LAM. Note that, in this case, the main savings come from the removing of local connections that occupy the 80% of the reducible area.

In the rest of the algorithms analyzed the LN operations have centered almost all the efforts and the cell configuration effect is less evident. Only in the SIFT space-scale generation we have openly preferred the 5 CC diamond configuration in combination with the H/V separability of the Gaussian kernel because of its almost no penalty compared to the full dense 9 CC configuration. It is then, in this trade-off analysis where we can check the correspondence between the cell configuration election with the expected results of the general S&S analysis. Obviously we have chosen cell configurations respecting the full functionality of the cell and we have also analyzed the shape of the involved templates, concluding, as in the general study that the NEWS connectivity configurations are the most adequate. We have also seen that in configurations with few CC and applying the image shifting mode, it is interesting to distribute the CC in two templates.

In the CNNA08 paper we also gather the shape analysis of the templates involved in several algorithms sufficiently detailed in the CNN literature, included the PLS reviewed here. The statistics from this analysis support the NEWS connectivity election in 4 of the 5 algorithms reviewed. In the fifth one, a vein feature extractor, the dense kernels meant around the 60% of the total. Also interesting is the number of occurrences of operations just involving the central CC as local logic operations, arithmetic operation or even threshold operations and from what we can conclude the convenience of also implementing the feedback CC, at least in one of the two implementable templates. These operations meant the 20% of the operations in two of the algorithms and around the 50% in other two, with even slightly more occurrences than the diamond shape operations in the PLS one.

Conclusions and Future Work

The research work addressed in this manuscript is placed midway between the algorithmic and the circuit design within the field of Cellular Processor Arrays oriented to image processing. We have developed a methodology as a framework to a whole series of methods and techniques that allow the simplification of the hardware avoiding functional limitations.

The first motivation of this work was to enable long distance, i.e. large neighborhood, communications in local connected networks of processing elements for fine grain low level image processing. Afterwards we observed that the same methodology could be applied over the local connectivity in order to reduce the number of connections and, consequently, the area of the processing elements. From the methodology and techniques development and the proposals validation we conclude that the initial objective can be achieved with minimal modifications in the local connectivity proposal of implementation, being even directly implementable over certain realized implementations. The amount of area reduction would depend on the starting point implementation, but, even for the tiniest implementations, the proposal can carry out the function of compensating for the extra hardware required in the worst cases for the long distance communication implementations, a local analog memory.

The manuscript begins with a review of the CPA particularization we have focused on, the Cellular Non-linear Networks (CNNs), concluding that they represent a good option in the implementation of visual microprocessors as it provides local-connectivity, key in achieving massive parallelism for low-level image processing. The proposed methodology imposes, nevertheless, one restriction in the hardware model election, to have well-defined and predictable internal states at any moment after each template/kernel application. In CNNs it is translated in using the discrete time CNN model or just in applying the methodology to the control template and not to the feedback template in continuous-time CNNs. No other restrictions are imposed in relation to the data type, for example, as the methodology considers two application modes, one for G/S implementations and another one that is valid for both G/S and B/W.

The goals of preserving the functionality and even enlarging it with long distance communications with minimum hardware modifications in both B/W and G/S processing exclude solutions based on particular devices or architectures (e.g. [Wu and Chen, 2009], [Ayoub et al., 2004], [Laiho and Lehtonen, 2010]), but also methodologies restricted to G/S (e.g. [Ślot, 1994]) or B/W (e.g. [ter Brugge et al., 1998c]) implementations or limited to a particular kind of templates as those obtained by 3×3 recursion. We look for making the proposal applicable to any kind of linear template and adaptable to any hardware particularization with minimum limitations. This is why we have headed our efforts to system level general solutions. Another two goals in the development of our proposal are: 1) the simplicity of conception and application, and 2) an affordable penalty at processing level. On this basis we have developed a template partition methodology, the Split and Shift (S&S) based on the associative property of the addition. We have proposed two modes of application depending on how we apply the shifts to correctly gather the grouped results, either applying them to the image to be weighted (image shifting mode) or by applying them to the grouped or partial results. The consideration of the first option makes the methodology applicable over completely binary implementations. We have also developed techniques both for the split and the shift phases and we have analyzed the implications of these techniques at hardware and processing time level. For the assessment in the reduction of the area occupation we have defined an FoM to evaluate the benefit (area reduction) - penalty (processing time increment) trade-off and we have used it to choose the most adequate techniques.

The main contributions of the methodology development are the simplicity of the conception and an organized set of guidelines of application to obtain a minimum penalty at processing time and absolutely no penalty at functional level in the achievement of the goals.

In the LN emulation we measure the cost of widen the CPA functionality as the number of operations required for the LN operations application. From the analysis we mainly conclude that the splitting methods should begin from a template corner and overlap incomplete sub-templates when necessary to keep the sub-template centers close to the central cell. About the shifting techniques we observe the convenience of the shift-sharing option in both image and partial-result shifting modes. A regular process together with shift-sharing can give benefits in terms of simplicity and automation. We suggest, as the best option, a concentric decomposition and spiral or zig-zag shifting. Nevertheless, central shifting offer slightly better results in number of operation at the cost of irregularity for more demanding implementations.

In the case of hardware reduction we have a trade-off between the benefit obtained in hardware reduction and the number of operations required to keep the functionality of the implementation. This trade-off does not depend only on the number of coefficient circuits (CC) but on the selected cell configuration. We have gone through the cell configuration election under four criteria.

The first criterion ensures the preservation of the full functionality without restrictions at kernel shape or size. This criterion imposes a minimum number of 3 CC and a distribution of CC that allows all the shifts required to communicate to all neighbors.

The second criterion takes into account the performance of the implementation by defining a Figure of Merit. This FoM is called RPO and measures the relation between the percentage of CC reduced and the number of operations increased per original operation. For a general single-template operation we would select a 6 CC lateral configuration, with the central column of CCs removed, as the best trade-off option. However, if we allow the distribution of the CC in two different templates we obtain a better trade-off value with a 3+1 CC lateral configuration, without CCs on the central column, for partial result shifting mode as it requires the same number of operations with less number of CC thanks to the operations overlapping. For a two-template operation, to re-use the same hardware for the implementation of both templates, either considering the CC allocated in a single template or distributed in two, is the best option.

The third criterion in the cell configuration election appears from a deeper analysis of the RPO definition and the evidence that cell configuration and template shape matching would provide a best case. We go further in this criterion and we realize a study of template shape through the most representative CNN template library, the CSW. From this study we conclude that most of the gathered CNN operations exhibits a diamond distribution of the template elements and that they are mostly symmetric, what when combined with result shifting, can be used to reduce the number of operations. Operations with just central CC as logic or arithmetic operations between others, are also significant. As a consequence, a 5 CC diamond configuration, i.e. the classical NEWS with the feedback coefficient circuit, represents a good trade-off option, what in addition justifies the generally assumed efficiency of the NEWS limited connectivity. The study also analyzes the symmetries and proposes a way of taking advantage of them.

The final criterion are, obviously, the goals to be reached in the implementation, that would set the actual limits in processing time and area occupation. According to this criterion, the application of the S&S methodology does not have strict techniques to be applied, but guidelines for its application. This means that we can develop different techniques or ways of application with similar results, which would be better as they are more adapted to the particular case.

The combination of both, LN emulation and hardware simplification, is completely assumable. Nonetheless, as the LN emulation demands a significant number of shifts, the cell configuration and LN emulation shift technique should look to each other. The usage of possible symmetries (with result shifting) and two-template configurations are also shown as an advantageous resources.

Until here we have the conclusions obtained from the methodology development gathering the main techniques and recommendations on the methodology application. To validate the proposals we have gone through actual CNN implementations and different low level image processing algorithms.

From the physical implementations analysis we conclude that, as expected, the application of the hardware reduction is much more profitable in G/S architectures where the CC implemented are in general bigger and where the analog local memory is usually included. Nevertheless, the hardware reduction S&S techniques can be used to compensate for the area occupied by the extra LAM required in general by a binary implementation if we choose to provide it with LN functionality through the S&S methodology.

The analysis of the FPGA implementations confirms in general our predictions of hardware reduction. As the 9 CC implementation does not fit our FPGA area, its implementation data cannot be taken as strict numerical reference for, for example, the HR assessment. Nevertheless, we can take the relative area values between the different actually implemented configurations, i.e. we choose a different starting point. Moreover, this election fits better the original HR definition as it just takes into account the number of CC reduced, and not the extra hardware required by the S&S, that is supposed to be the same independently of the number of CC. In fact, comparing the occupation data of the different configurations we obtain HR results similar to the obtained with the simple initial definition. Slightly different values are obtained for different CC allocations, but in general the HR is proved to be a good tool for the assessment of the area reduced for the comparison of cell configurations. Moreover, we consider proved the no significant contribution of the inter-PE connections in this case, what reinforces our election of the simplest HR definition. Note that for full-custom design this cannot be stated in general, but in any case, the difference between the number of CC and connections removed is, at most, 2, and, as it was shown in Chapter 3, it does not implies differences in the configuration comparison further than we expect that configurations with the same number of CC occupies less area if 1 or 2 of them are feedback CC.

Finally, we have also shown the feasibility of realizing actual topographic DTCNN implementations over FPGA with the help of the S&S methodology. This line was in fact followed in the B/W and a G/S implementations gathered in Appendix B for actual applications.

On the other side, we have assessed the application of the methodology to state-of-the-art low level image processing algorithms including LN communications. In this case we have not limited us to the CNN algorithms. In fact, SURF and SIFT algorithms had not been previously implemented over CPAs, and the first conclusion is that S&S methodology allows their application over these locally connected massively parallel architectures despite their needs of large neighborhood operations.

Results are even promising, estimating that the four scales SIFT scale space generation can take ∼1 ms with around 1000 3×3 operations in a 5 CC NEWS configuration, and that the whole application of 12 7×7 spin filters are realized with a total of 136 3×3 operations in a 4CC NEWS configuration.

The case of the scale space generation in the SURF algorithm is a bit different. The implementation of the integral image over CPAs leads to the parallelization of its calculation, what has been looked for in the reference literature. Nevertheless, due to the particularity of the integral image definition, the parallelization is limited to one line at a time. This leads us to propose the utilization of LPAs instead of CPAs, because, in addition, their lower number of PE allows the implementation of larger memories, what is a requirement of the integral image. In this first part the methodology is almost reduced to shifts and accumulations with the exception of the initial sub-template application that reduces the number of operations required. The second part of the SURF scale-space generation involves the box-filters application, that can be considered as proper LN operations. Together with the integral image, they are reduced to some additions of values occupying large distance positions, that can be implemented with the S&S techniques over a CPA. In this case the number of operation depend on the image size for the integral image calculation resulting $N+M$ 3×3 operations for an $N\times M$ image size. The box filters application can takes around 3000 3×3 operations for four octaves. In both cases for a full-dense or a 5 CC NEWS configuration, being another example of the inefficiency of having implemented a full dense template.

For the whole trade-off analysis we have chosen an application oriented implementation of the PLS algorithm. At the sight of the results we observe that with the methodology proposed we can not only enlarge the functionality of the proposal by allowing LN operations, but that area improvements can be expected even after the introduction of the required LAM. Note that, in this case, the main savings come from the removing of local connections that occupy the 80% of the reducible area.

This trade-off analysis also has allowed us to check the grade of correspondence between the cell configuration election with the expected results of the general S&S analysis. Obviously we have chosen cell configurations respecting the full functionality of the cell. We have also analyzed the shape of the involved templates, concluding, as in the general study, that the NEWS connectivity configurations are the most adequate. We have also seen that in configurations with few CC and applying the image shifting mode, it is interesting to distribute the CC in two templates.

We also gather the shape analysis of the templates involved in several algorithms sufficiently detailed in the CNN literature, included the PLS reviewed here. The statistics from this analysis support the NEWS connectivity election in 4 of the 5 algorithms reviewed. Also interesting is the number of occurrences of operations just involving the central CC as local logic operations, arithmetic or even threshold operations, from what we can conclude the convenience of also implementing the feedback CC, at least in one of the two implementable templates.

Future Work

As in the validation, our perspective about future work has two main lines, the algorithmic and the hardware.

Within the algorithmic line we plan the implementation of the scale space generation of the SIFT algorithm over CPA platforms. The application of the S&S methodology over fully digital implementations comprising just one ALU per PE as that in reference [Lopich and Dudek, 2011a], or a MAC as that in reference [Rodríguez-Vázquez et al., 2008] is also a matter of future work. A further objective is the adaptation of the methodology to its application over architectures with less fine grain parallelism where the processing elements deal with several pixels instead of just one.

Within the hardware line we have three main concerns to deal with over a full-custom implementation, namely, the implications of the methodology over the power consumption and over accuracy required by the weighting circuits, and the implementation of the LAMs memories when they do not exist.

About the power consumption we expect a lower instant consumption but perhaps a higher average consumption due to the higher number of operations and processing time. Nevertheless, if we consider that weightings by null coefficients also consume power, the reduction of the number of weighting circuits implemented together with the high incidence of the sparse templates within the CNN operations would lead to an improvement in this aspect.

Nevertheless, the accuracy required imposes a minimum in the power consumption of a circuit [Kinget, 2005]. And this, together with the higher area required by higher accuracy lead us to the second concern on hardware issues. We expect that the mismatch between nominally identical transistors, the main error source in an analog circuit, decreases as the number of components working at the same time decreases. In addition, the liberated area provided by the CC removal can also be used to increase the weighting circuits accuracy by increasing their transistors area.

Finally, although G/S architectures already offer local analog memories that can be used for the S&S methodology, it would be beneficial to find a minimum size LAM to allow minimum size binary implementation take advantage of the S&S methodology. Also, the many cycles needed for an actual application might well cause to adopt some strategies for memory refresh in order to avoid the degradation of values stored in analog memories.

Appendix A

Published papers gathering the thesis work

This appendix gathers the published papers that summarize the development of the research work and the contributions themselves. Papers are referred along the text indicating the name of the conference where they were presented and the year of presentation. Papers presented at the same conference are distinguished with letters. On the page before each paper we introduce the reference and the key name of the paper. Papers are ordered chronologically.

CNNA05:

N. A. Fernández, D. L. Vilariño, V. M. Brea and D. Cabello. " **On the Emulation of Large-Neighborhood Templates with Binary CNN-Based Architectures**," in *Proceedings of the* 9^th *IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA 2005*, pp. 274-277, Hsinchu, Taiwan, May 2005.

ON THE EMULATION OF LARGE-NEIGHBORHOOD TEMPLATES WITH BINARY CNN-BASED ARCHITECTURES

N.A. Fernández, D.L. Vilariño, V.M. Brea, D. Cabello

Department of Electronics and Computer Science
University of Santiago de Compostela
Santiago de Compostela, Spain
Phone:+34981563100, Ext. 13580. Fax:+34981528012. Email:natiafg@usc.es

ABSTRACT

This paper addresses the extension of applications covered by binary CNN-based architectures. The work is focused on diffusion-like tasks on binary images, traditionally tackled by either large neighborhood or propagating templates on a CNNUM architecture. The solution adopted here is to split large neighborhood into smaller templates (3×3) on a binary CNN-based architecture. Trade-offs and hardware issues arisen from such an approach, as well as examples of application, are discussed throughout the paper.

I. INTRODUCTION

The realization of large-neighborhood templates on a CNN chip results into either solutions with low density of cells or into slower applications. In a CNNUM chip, the former would lead to more coefficient circuits per cell [1], [2]. The latter solution would imply to feedback 3×3 templates, (the nearest-neighbor connected pattern), as many times as needed to have a valid approach to the large-neighborhood template to be implemented [3]. This might be a limitation to fast-time response applications. Concerning the range of applications covered by the CNNUM model, the piece-wise linear output-state relationship allows to run algorithms with B/W and gray-scale inputs/outputs [4].

Binary CNN-based architectures are an emerging approach to CNN on-chip implementation [5], [6]. Their range of applications is restricted to algorithms with B/W inputs and outputs. In these architectures, the piece-wise linear output-state relationship is exchanged for a high gain non-linear function. The major consequence is to have very simple coefficient circuits, leading to chips with a high performance, especially in area and processing speed. The solution is highly suitable for propagating B/W tasks like the hole filling, or for processing images with high resolution (number of pixels) [6]. Nevertheless, by adding new functionalities would be feasible to tackle a wider range of applications, especially algorithms with B/W inputs/outputs comprising partial gray-scale outcomes.

In [7], an extended version of a binary CNN architecture to perform a low-pass filtering function with 3×3 templates on a B/W image was reported. The result is a local processor comprising a binary CNN cell for executing B/W tasks and specific circuitry for dealing with the gray-scale output from the low-pass filtering operation. The present work is aimed at CNN image processing with B/W inputs and partial outcomes in gray-scale mode. Diffusion-like tasks fall into this category, which, as it was mentioned above, are performed either with large-neighborhood or with 3×3 gray-scale propagating templates in a CNNUM architecture. Here, we propose a solution by splitting large neighborhood into 3×3 templates running on a binary CNN-based architecture. The paper is outlined as follows. Section 2, from a 5×5 template, addresses the decomposition of large-neighborhood into 3×3 templates. Section 3 goes through the extension to greater orders of neighborhood, discussing the major trade-offs and hardware issues. Finally, conclusions and a brief outlook are given.

II. 5X5 TEMPLATE EMULATION

In order to illustrate how to emulate large-neighborhood templates with standard 3×3 templates on binary CNN-based architectures we describe in detail the operations needed in a generic 5×5 template. The first step is to split the 5×5 neighborhood into 3×3 subwindows. There are multiple window-split methods. Fig. 1 illustrates two of them. The number of subwindows is directly related with the number of resultant 3×3 templates. Clearly, in a 5×5 neighborhood the minimum number of 3×3 subwindows is four. The 5×5 template response is approached by the combination of CNN-operations based on two kind of 3×3 linear templates:

- *Decomposition templates*, dependent on the particular set of coefficients in the original 5×5 template.

- *Shift templates*, needed to drive the contribution of far neighbors into the sphere of influence of the cell under consideration (the central cell in the 5×5 window).

Concerning the decomposition templates, it is well known that sparse templates tend to be more efficient in terms of robustness [8]. Therefore the overlap of subwindows (Fig. 1) should be done with as many null template coefficients as possible.

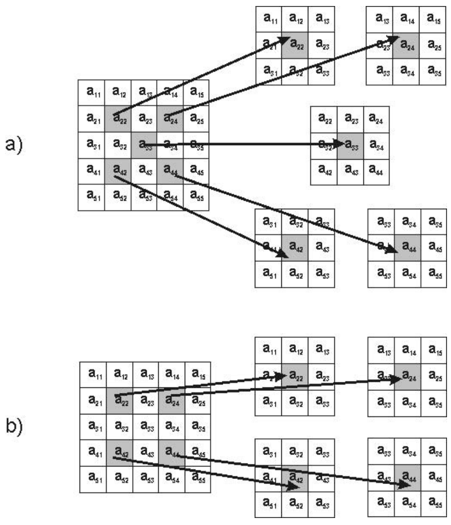


Fig. 1. Window-split methods of a 5×5 neighborhood into 3×3 subwindows.

The minimum number of shift operations (i.e., the number of shift templates) and the coefficients in the decomposition templates relies on the window-split method. Nevertheless in a 5×5 neighborhood with *N* subwindows is clear that the required number of shift operations will be either *N* or *N-1*. The latter occurs when the central 3×3 window of the original 5×5 template coincides with one of the 3×3 subwindows, as is the case of Fig. 1a.

The decomposition and the shift templates are combined in a multistep CNN algorithm. The outcome should be the same as that of an operation with the original 5×5 template. Two different multistep CNN algorithms could be followed:

- *Fixed image algorithm*, where each decomposition template is applied on the image under processing and the result is shifted by the associated shift template to the central cell in the 5×5 neighborhood.
- *Shifted image algorithm*, where the image under processing is previously shifted by a shift template in order to compute the corresponding decomposition template directly in the cell under consideration (the central cell in the 5×5 neighborhood).

Since we are constrained to CNN architectures computing binary inputs, the *shifted image algorithm* is clearly advantageous. In this algorithm, real-valued outputs are never fedback, but they are accumulated and stored in the central cell of the 5×5 neighborhood. As a consequence, the high gain non-linear activation function can be used, resulting into an efficient on-chip implementation [6]. The features added to a binary CNN architecture are so few. Fig. 2 displays a simplified view of the binary CNN-based architecture for the *shifted image algorithm*. The original image (binary data) stored in a local logic memory (LLM) is the input to the CNN module in order to compute a shift operation. The output (binary) is fedback to the CNN module to run a decomposition template. The resulting internal state (real data) is added to the data stored in a local analog memory (LAM) and subsequently the result is saved in the same LAM. Therefore, after computing all the operations the data stored in the LAM will be the sum of the partial outcomes of all the decomposition templates, being the same output as that of the original 5×5 template.

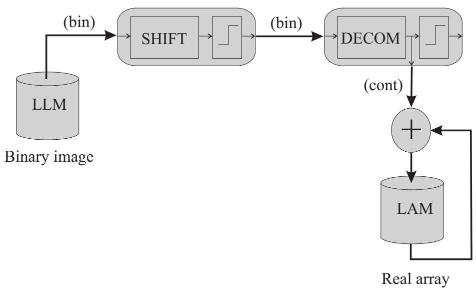


Fig. 2. Binary CNN-based architecture for the *shifted image algorithm* in a generic large-neighborhood template split into 3×3 templates.

Fig. 3 illustrates the sequence of shift and decomposition templates for the *shifted image algorithm* applied to the window-split method shown in Fig. 1b (four 3×3 subwindows). In order to illustrate the validity of the proposal, we have computed the diffusion operation with the 5×5 template of Eq.(1). This is the result of running twice on the CNNUM the 3×3 diffusion template listed in [9]. Its accuracy (robustness) should be modified according to that of the specific binary CNN-based architecture, i.e. as low as possible, in order to have as a dense on-chip implementation as possible [8].

$$\begin{bmatrix} 0.01 & 0.03 & 0.0425 & 0.03 & 0.01 \\ 0.03 & 0.045 & 0.06 & 0.045 & 0.03 \\ 0.0425 & 0.06 & 0.13 & 0.06 & 0.0425 \\ 0.03 & 0.045 & 0.06 & 0.045 & 0.03 \\ 0.01 & 0.03 & 0.0425 & 0.03 & 0.01 \end{bmatrix} \quad (1)$$

Fig. 4 shows the emulation of the 5×5 template of Eq.(1) with 3×3 templates complying with the *shifted image algorithm* for the window-split method of Fig. 1b.

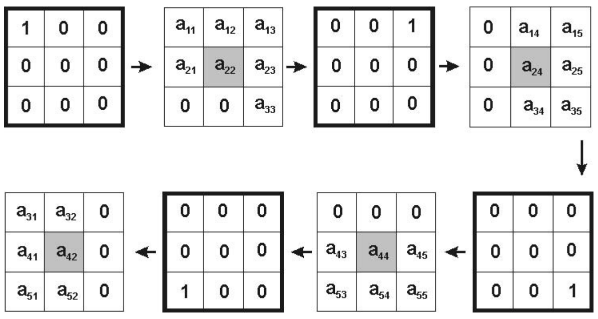


Fig. 3. 3×3 templates in the *shifted image algorithm* for the window-split method of Fig. 1b in a 5×5 neighborhood.

Every one of the eight partial outcomes is the result of every template (shift or decomposition and accumulation) depicted in Fig 3. These operations are performed in the binary CNN-based architecture displayed on Fig. 2. The final output is exactly the same as that of the original 5×5 template.

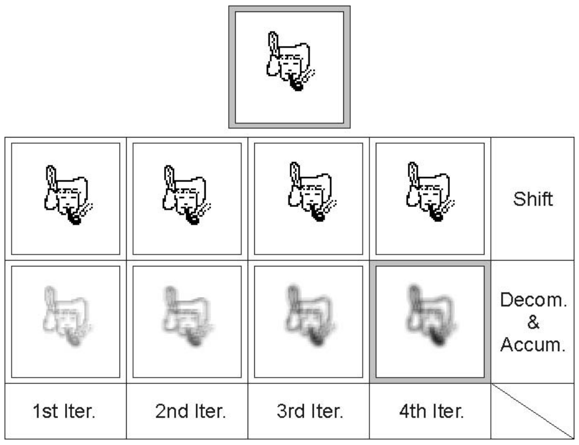


Fig. 4. Diffusion operation performed with 3×3 templates according to the architecture of Fig. 2 for the emulation of a 5×5 template.

III. LARGE-NEIGHBORHOOD TEMPLATES EMULATION

Following the strategy discussed in the previous section is possible to approach the computing of larger-neighborhood templates based on 3×3 linear templates running on a binary CNN chip-set architecture. Nevertheless, in order to adapt the described methodology there are some new alternatives to be taken into account affecting both the window-split (number of subwindows) and the shift methodology.

Concerning the window-split method, apart from splitting the neighborhood directly into 3×3 subwindows, it can be considered a modular (recursive) methodology where the original template is subdivided in subwindows $(2n + 1) \times (2n + 1)$ with $n > 1$. Every subwindow will also be split in order to compute with the nearest-neighbor connected pattern (3×3). Each subwindow can be computed as an independent template and the result shifted to the central cell. Unfortunately, this methodology has the same drawbacks as the *fixed image algorithm* for combining the shift and decomposition template outcomes (Section 2). It is required to operate with real data within the CNN module (feedback), discarding the use of binary CNN architectures, and thus its hardware benefits [6]. Therefore, the direct split of the template into 3×3 subwindows seems to be the better choice. In this case, the number of decomposition operations (templates) is:

$$DO(n) = (\frac{2n+1}{3}\rceil)^2 \qquad (2)$$

As a difference from the emulation of 5×5 templates, in larger-neighborhood templates the shift operations outnumber the decomposition operations, as there are more than one shift for decomposition template. The reason is that the distance from the central element of a 3×3 subwindow (template) to the cell under study (center of the $(2n+1) \times (2n+1)$ window) can be more than one pixel, and we are constrained to 3×3 templates. Nevertheless some shift operations can be shared, allowing to reduce the computational effort. Fig. 5 collects different techniques to approach the shift operations:

- An independent shift approach. This is a direct extension of the approach for the 5×5 templates. In this case, the shift operations are not shared.
- A dependent shift approach where the 3×3 subwindows are shifted in a *zig-zag* way.
- A dependent shift approach where the 3×3 subwindows are shifted in a concentric way.

In Fig. 6 a comparison of number of shift and decomposition operations for the three approaches shown in Fig. 5 with respect to the neighborhood order of the template is showed. As it can be seen, the three approaches described above have approximately the same number of operations when the order of neighborhood (n) is less than 6-7. From that number onwards, dependent shift approach methods are clearly advantageous. These operations are performed on a binary CNN-based architecture. In such architectures, the computation time for every 3×3 template can be programmed to be in tens of nanoseconds [6]. If a hundred of microseconds is set as the upper threshold to have video rate processing, it can be seen that templates with an order of neighborhood of n=30 (61×61 templates) could be fit in such a time slot. It is also apparent that large neighborhood templates would lead to stringent memory requirements. Also, template uploading times must be accounted in order to estimate whether or not the application meets the video rate processing. Nevertheless, orders of neighborhood inferior to n=5 are sufficient for the great majority of applications.

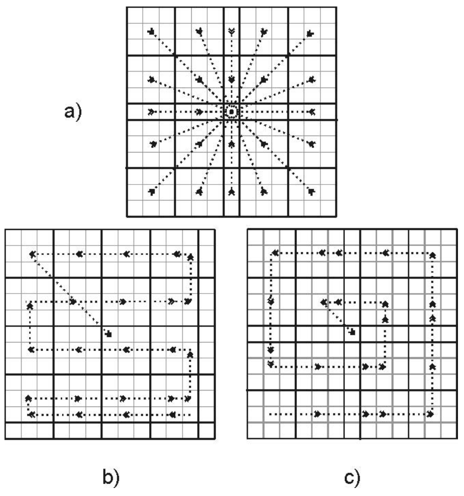


Fig. 5. Shift techniques in a large-neighborhood. a) Independent shifts, b,c) Dependent shifts.

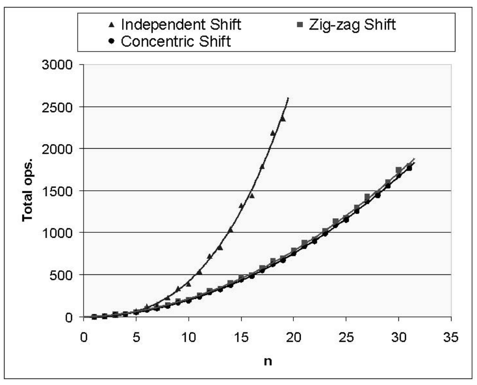


Fig. 6. Total operations (number of shift and decomposition templates) versus order of neighborhood with different shift strategies.

IV. CONCLUSION

Binary CNN-based architectures seem to be suitable to reach a good performance in area and power consumption. This is mandatory to achieve higher cell density and thus to approach new real-life applications. Unfortunately it is usually done at the expense of a considerable reduction of the application domain. This paper proposes a strategy to approach CNN-operations with large-neighborhood templates by means of binary CNN-based architectures. The final goal is to increase the application domain of this kind of architectures.

The emulation of a 5×5 template response has been deeply discussed to illustrate the proposal. Furthermore, the extrapolation to larger neighborhood templates has been addressed, including some remarks on the efficiency based on the order of the template to be emulated. The evaluation based on the hardware requirements and computing time hint that the proposed strategy can be suitable for emulating templates of moderate size.

V. ACKNOWLEDGMENTS

This work was funded by Ministerio de Ciencia y Tecnología (Spain) under the Project TIC2003-09521.

VI. REFERENCES

[1] G. Liñán, et al., "ACE4k: An Analog I/O 64 x 64 Visual Microprocessor Chip with 7-bit Analog Accuracy", Int. J. Circuit Theory Applicat., vol. 30, no. 2/3, pp. 89–116, 2002.

[2] A. Rodríguez-Vázquez et al., "ACE16k: The Third Generation of Mixed-Signal SIMD-CNN ACE Chips Toward VSoCs", IEEE Trans. Circuits Syst. I, vol. 51, no. 5, pp. 851–863, May 2004.

[3] K. Slot, "Large-neighborhood Templates Implementation in Discrete-Time CNN Universal Machine with Nearest-neighbor Connection Pattern", Proceedings CNNA94, pp. 213–218, 1994.

[4] D.L. Vilariño, Cs. Rekeczky, "Implementation of a Pixel-Level Snake Algorithm on a CNNUM-based Chip Set Architecture", IEEE Transactions on Circuits and Systems-I, vol. 51, no. 5, pp. 885–891, May 2004.

[5] V.M. Brea, D.L. Vilariño, A. Paasio, D. Cabello, "On the One-Quadrant Template Design in a High Gain CNN Model", Proceedings CNNA2004, pp. 279–284, 2004.

[6] J. Flak, M. Laiho, A. Paasio, K. Halonen, "VLSI Implementation of a Binary CNN: First Measurement Results", Proceedings CNNA2004, pp. 129–134, 2004.

[7] V.M. Brea, D.L. Vilariño, A. Paasio, D. Cabello, "Design of the Processing Core of a Mixed-Signal CMOS DTCNN Chip for Pixel-Level Snakes", IEEE Transactions on Circuits and Systems-I, vol. 51, no. 5, pp. 997–1013, May 2004.

[8] A. Paasio et al., "CNN Template Robustness with Different Output Nonlinearities", International Journal of Circuit Theory and Applications, vol. 27, pp. 87–102, 1999.

[9] Analogical, Computer Neural Computing Laboratory, and Hungarian Academy of Science, Automation Institute (MTA SzTAKI). "CNN Software Library (CSL)". http://lab.analogic.sztaki.hu/Candy/csl.html.

DCIS05:

N. A. Fernández, D. L. Vilariño, V. M. Brea and D. Cabello. " **Large Neighborhood Templates with Nearest-Neighbor Connected Patterns in Binary-Based Cellular Neural Networks**", in *Proceedings of the XX Conference on Design of Circuits and Integrated Systems, DCIS 2005*, Lisbon, Portugal, November 2005.

Large Neighborhood Templates with Nearest-Neighbor Connected Patterns in Binary-Based Cellular Neural Networks

N.A. Fernández, D.L. Vilariño, V.M. Brea, D. Cabello
Department of Electronics and Computer Science
University of Santiago de Compostela
E-15782 Santiago de Compostela, Spain
Phone +34981563100, Ext. 13580. Fax:+34981528012.
Email:natiafg@usc.es

Abstract— **This paper aims at extending the range of applications tackled with binary-based Cellular Neural Networks. Such an extension is focused on diffusion-like tasks on binary images. This is traditionally done with large-neighborhood templates. The solution adopted here is to split large neighborhood into templates with the nearest-neighbor connected pattern (3×3). Simulation results in an active contour technique illustrate the validity of the approach. Trade-offs and hardware issues are also discussed throughout the paper.**

I. Introduction

Cellular Neural Networks (CNN) make a computing/architecture paradigm especially suitable for image processing in an SIMD fashion with a pixel-to-cell correspondence at hardware level [1]. At image level, CNN architectures are usually meant for either gray-scale or B/W processing. ACE16k is the epitome of CNN on-chip solutions for gray-scale processing, having the B/W as a particular case [2]. B/W, also called binary, architectures are mainly focused on very demanding B/W image tasks. In the last case, the relatively simplicity of the problem permits to come up with binary CNN models meeting stringent hardware demands, mainly of reconfigurability/programmability, power dissipation, area consumption and processing speed [3]. It is plain, however, that their main drawback is the lack of gray-scale processing. Binary-based CNN architectures with extensions to gray-scale processing emerge as a good alternative [4]. The additional circuitry from gray-scale processing is small enough as to have very competing solutions [5].

Another issue is how to solve image tasks with large-neighborhood templates without a significant penalty at hardware level. Such templates, 5×5 and beyond, might be common in diffusion-like tasks [1]. This kind of tasks are performed either with large-neighborhood, i.e. with more coefficient circuits per cell, or with 3×3 gray-scale propagating templates, as is the case of an ACE16k-like architecture. The latter solution implies the feedback of the gray-scale output to apply the 3×3 templates needed to have a valid approach to the large-neighborhood template to be implemented [6].

The methodology used here is aimed to implement large-neighborhood templates into binary CNN architectures. This is a simple technique, previously introduced in [7], that is based on the associative property of the addition and that results in an exact emulation of the original template. The example exposed here is based on a binary-based CNN architecture with extensions to gray-scale processing, needed to deal with gray-scale outputs. The goal is to solve diffusion-like tasks on binary images by splitting large-neighborhood into 3×3 templates, and with as small a CNN cell as possible. Here, it is applied to an active contour-technique, the so-called Pixel-Level Snakes (PLS) [8].

This paper is outlined as follows. Next section recalls some of the splitting-methods discussed in [7]. Section III addresses the PLS technique, highlighting those features where the splitting methods do their best. Section IV illustrates the validity of the approach with the PLS technique. Hardware issues are also discussed throughout this section. Finally, the main conclusions along with a brief outlook are given.

II. Large-Neighborhood Splitting Methods

The first step is to split the large-neighborhood window into 3×3 subwindows (nearest-neighbor connected pattern). Accordingly, the $(2n+1) \times (2n+1)$ template, with n being the neighborhood order, i.e. an integer number greater than one, has to be broken into 3×3 templates. The partial outcome from every 3×3 template has to be collected/updated in the cell under study, central cell in the $(2n+1) \times (2n+1)$ window, in order to have the same result as that with the original $(2n+1) \times (2n+1)$ template. In so doing, there are two kind of templates. Decomposition and shift templates. The former templates result from the original large-neighborhood template. The only concern in their design is to have as many null coefficients as possible, since the robustness increases with the number of zeros [9]. The shift templates come out straightfowardly [7].

Decompositon and shift templates are subsequently combined in a multistep algorithm run on a cell like that depicted in Fig. 1. The original image (binary data) stored in a local

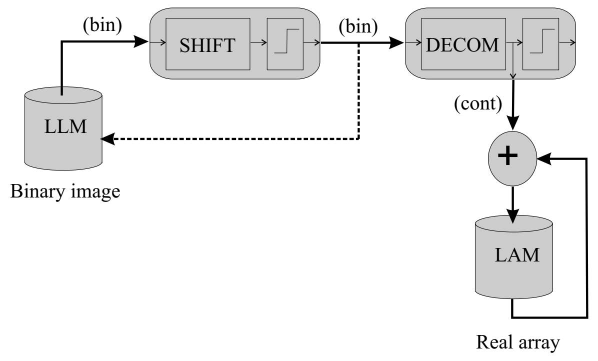


Fig. 1. Binary-based CNN architecture for a generic large-neighborhood template split into 3×3 templates.

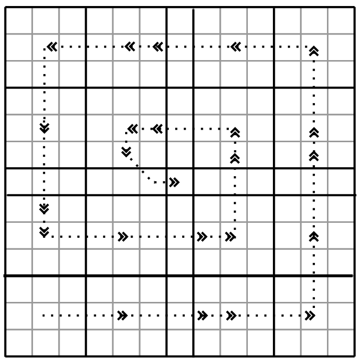


Fig. 2. Concentric shift technique in a large-neighborhood window.

logic memory (LLM) is the input to the CNN module in order to compute a shift operation. The output (binary) is fedback to the CNN module to run a decomposition template. The resulting internal state (real data) is added to the data stored in a local analog memory (LAM) and subsequently the result is saved in the same LAM. Therefore, after computing all the operations the data stored in the LAM will be the sum of the partial outcomes of all the decomposition templates, being the same output as that of the original $(2n + 1) \times (2n + 1)$ template. It is worth pointing out that the addition is supposed to be performed by KCL (current summation) in a single node with a voltage outcome. As a consequence, the coefficient circuits should be transconductance elements. This is in line with most of the CNN on-chip implementations [10]. It should also be apparent that LAM is the only additional device in analog mode with regards to an entirely binary realization. Its hardware realization does not lead to a significant extra cost. A simple SI or S^{2}I should be good enough [11]. LLM and LAM are features of the so-called Universal Machine (UM) [1].

The aforementioned multistep algorithm combining decomposition and shift templates is fundamental to determine the number of operations (performance) of the large neighborhood splitting method. In [7] was shown that when the original binary image is shifted, in order to have the outcome of every 3×3 subwindow (template) in the central cell, in either a concentric or a zig-zag way, the number of operations heavily decreases. Fig. 2 displays how the multistep algorithm works with the shift templates proceeding in a concentric way. The key is to shift the image resultant from the previous shift move, instead of the original image (initial snapshot). This way, it is possible to share shift moves. The arrows in Fig. 2 mean how to carry out the shift moves. The first arrow points to the central pixel/cell in the $(2n + 1) \times (2n + 1)$ window. The second move would go from right to left along the row where the central cell is located. This is possible because the move is made with respect to the previous shift, but not with respect to the original image. Every shift is dependent on the former one. Similarly, the third move would go upward along the column where the central pixel is, and so on. This is the way to tackle large-neighborhood templates in our binary-based CNN architecture (Fig. 1). Next section tells the application where these ideas are used.

Fig. 3 displays the number of CNN operations versus order of neighborhood for the most advantageous method of those exposed in [7](concentric shift way). Note that it can be obtained better results in particular cases, e.g. sparse templates, with ad-hoc decomposition and coherent shift way. In a binary-based CNN architecture, every operation (template execution) can be done within tens of nanoseconds [5]. General-purpose SIMD solutions like ACE16K and SCAMP need μs to run a template-like operation [2], [12]. In terms of speed, Fig. 4 shows that the large-template implementation technique used here is clearly better than that of a feedback approach. This holds up to an order of neighborhood of 10. It is not having into account template uploading times (re-programmability rate). Nevertheless, orders of neighborhood smaller than n=5 are sufficient for great majority of applications and this additional time is low enough as to keep the binary-based CNN approach as a competitive solution. As a reference, there would be needed a rate of processing of $400 \mu s$ per operation in order to achieve video rate processing ($25 frames/s$) in an application that requires 100 operations per frame.

III. Pixel-Level Snakes

Originally introduced in [8], Pixel-Level Snakes (PLS) make up an active contour-based technique quite suitable for contour tracking and segmentation, either with still or with moving objects. In their latest version [13], PLS are placed midway between energy and level-set based models [14], [15], [16]. This makes this technique very efficient when dealing with complex applications like medical image processing with a low S/N content, or applications with several contours on the scene [4], [8].

The PLS technique comprises gray-scale and B/W operations [13]. The gray-scale processing is meant to extract the guiding information. The B/W processing entails the move of the contours. These are represented as sets of eight-connected pixels on a binary image.

Fig. 5 displays the major operations performed in the latest version of the PLS technique. The PLS algorithm is fed with two images: the external potential and the active

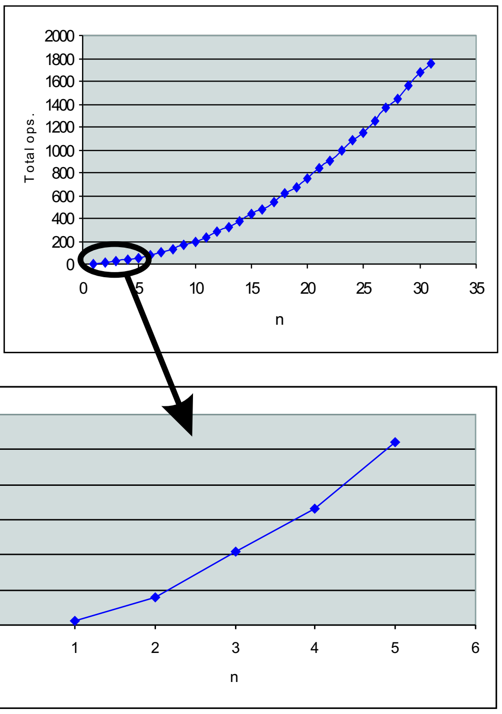


Fig. 3. Total operations (number of shift and decomposition templates) versus order of neighborhood. Zoom in most relevant neighborhood orders.

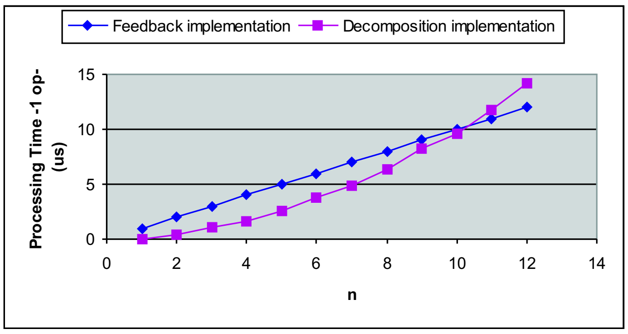


Fig. 4. Processing time for one operation performed with a large-neighborhood template versus neighborhood. Feedback implementation data is obtained by taking $1\mu s$ per CNN operation. Decomposition implementation by taking $50ns$ per CNN operation.

contour image. The external potential is a gray-scale image extracted from the image to be processed. It contains the most relevant information from the scene [14]. In the current implementation, the external potential is calculated outside, and taken as a static image for the PLS execution. The output of the Guiding Force Extraction (GFE) block is a B/W image, marking in black the locations toward the contours can go. The GFE output is a combination of the external potential with the so-called internal and balloon potentials. The two latter are extracted from the active contour image itself [17]. The Active Contour Evolution (ACE) block moves the contours according to the GFE outcome. This is carried out in the Directional Contour Expansion (DCE) and the Directional Contour Thinning (DCT) blocks. The Topologic Transformations (TPT) block deals with several contours when needed (topologic transformations). The latter encompasses morphological operations of erosion and dilation, as well as a propagating task, hole filling, and the binary edge detection [1]. Collision Point Detection (CPD) is an additional block used to spot those pixels (region in the image) where a collision is about to happen. This block can be used to make a decision on whether or not to have a topologic transformation. In order to get a better understanding of the PLS technique implemented here, the reader is addressed to [17], where an extensive set of examples with active contour applications like contour tracking or image segmentation can be found.

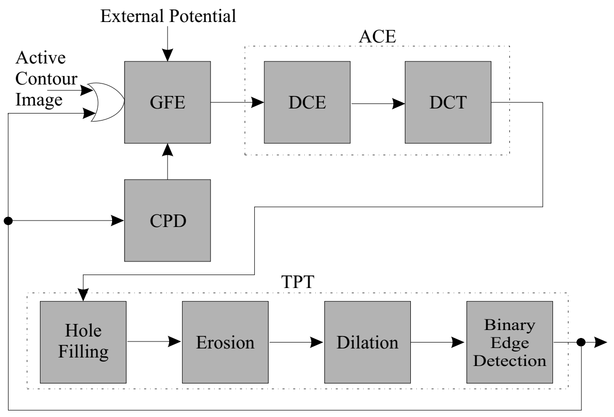


Fig. 5. Operations performed in the PLS technique.

IV. Simulation Results in the PLS Technique

As in classical active contours, in the PLS technique the contours are guided by means of three potentials: external, internal and balloon potentials [17]. As it was mentioned before, the external potential is a gray-scale image provided with the most relevant features from the scene. This is an image fed to the algorithm from outside (Fig. 5). The internal potential, however, is extracted from the active contours. Its aim is to keep the contours smooth, avoiding rough shapes (big concavities) along the contours. Likewise, the balloon potentials are updated from the contours themselves. They assist in moving them, especially in those homogeneous regions of the image to be processed, and in counteracting their tendency to shrink due to the internal potentials.

The approach of large-neighborhood templates with nearest-neighbor connected patterns is applied to the internal potential. Here, as a rule of thumb, the larger the curvature, the larger the neighborhood needed in the internal potential templates to achieve smooth contours. More concisely, the local curvature is estimated with the Derivative of the Gaussian (DoG) on the binary contour image. The higher the radio of curvature, the lower the outcome of such an estimate. This information is combined with the rest of terms involved in guiding the contours (external and balloon potentials). The goal is to lead

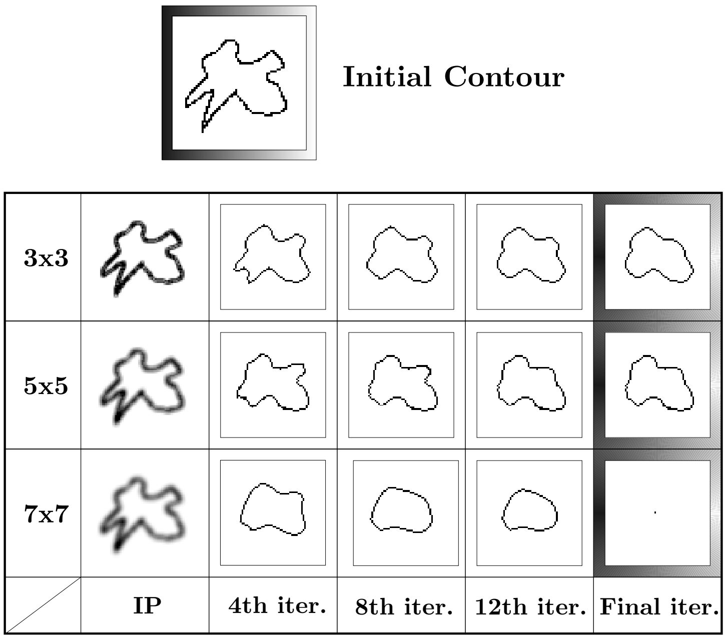


Fig. 6. Contour evolution in a closed contour guided exclusively by internal potential.

the contours to the regions of interest while keeping their shapes smooth [17].

Fig. 6 displays the evolution of a closed contour guided exclusively by internal potential. The expected outcome is a smoother shape in the contours. The contour evolution in Fig. 6 is accomplished with different orders of neighborhood, with the size of the template labeled in the leftmost column. The initial internal potential (first iteration) is also depicted in the second leftmost column. Concerning the evolution, it can be seen that a 3×3 size for the internal potential slightly smoothes the contour. In this case, a 7×7 template is sufficiently large as to collapse the initial contour into a single point (pixel). The 5×5 size gives an intermediate result.

Eq.(1) poses the 3×3 template used for the internal potential. The 5×5 and 7×7 templates run in Fig. 6 are obtained as the convolution of the 3×3 template listed in Eq.(1). This is a standard template widely discussed in the CNN literature [1]. It performs a low-pass filtering operation. Nevertheless, with a view to a custom on-chip realization in a binary-based CNN architecture, the template of Eq.(2) is far more adequate. Such a template allows to employ a positive range high gain non-linear model with 1-bit of programmability, leading to very efficient on-chip implementations [3]. At image processing level, however, its performance might differ.

$$\begin{pmatrix} 0.1 & 0.15 & 0.1 \\ 0.15 & 0 & 0.15 \\ 0.1 & 0.15 & 0.1 \end{pmatrix} \tag{1}$$

$$\frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{2}$$

Fig. 7 contains another contour evolution entirely guided by internal potential, on this occasion with an open contour. The target, only reached with a large enough template, is a straight line. The evolution displayed in Fig. 7 shows that, again, the larger the order of neighborhood, the smoother the shapes in the contour. Eventually, the straight line is attained.It should also be noted that the horizontal straight line would only be achieved with the border pixels anchored. If such pixels are not fixed, as is the case in Fig. 7, the final straight line is not horizontal. This can be clearly seen in the case of the 7×7 template.

Finally, we show an application where larger orders of neighborhood in the internal potential lead to better outcomes. This is the search of optimal routes. The field of application can be that of robot navigation. Fig. 8 illustrates how the PLS algorithm tackles the problem. These simulations were run on ACE4k [17]. The sequence reads left to right. The first frame shows the start and finish points encircled in white and black respectively. The second frame is the exploration step, where active waves (contours) are sent to the final point. This is performed with an inflating potential. Following, third frame in Fig. 8, deflating potentials are used. Finally, the route optimization step is done. It is plain that the internal potential would be fundamental in achieving more optimal paths. The larger the neighborhood in the internal potential, the straighter (shorter) the final routes (lines) would be. Simulations with different orders of neighborhood integrated in the PLS algorithm will be shown in the conference.

V. Conclusion

This paper has shown how to tackle large-neighborhood templates with nearest-neighbor connected patterns. The work is focused on diffusion-like tasks on binary images. The

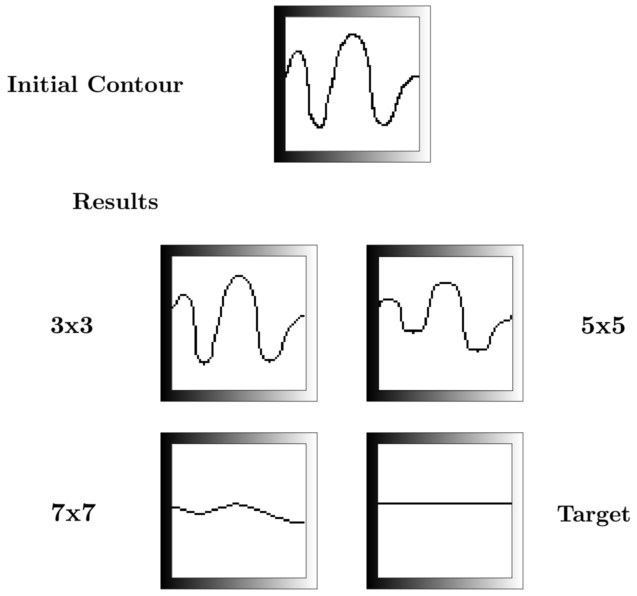


Fig. 7. Contour evolution in an open contour guided exclusively by internal potential.

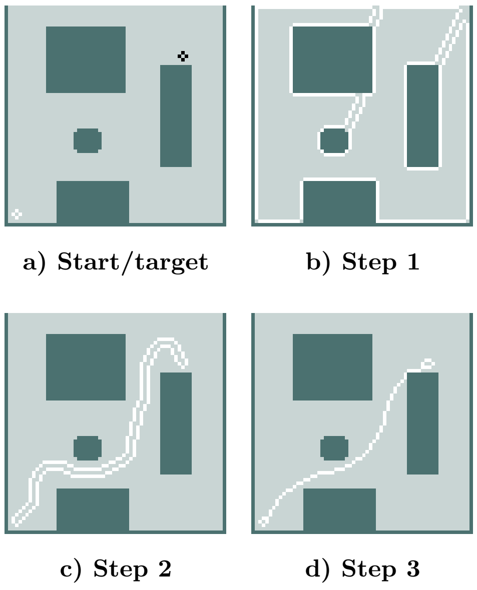


Fig. 8. Optimal route finding problem tackled by PLS.

approach is performed with a binary-based CNN model (cell). Extensions to gray-scale processing are kept as simple as possible, resulting into a hypothetical efficient CNN on-chip implementation. Such a model is tested on a relatively complex active contour-based technique, PLS. Simulation results show that the performance of the internal potential (and as a consequence that of the entire algorithm) improves significantly with larger orders of neighborhood. The binary-based CNN model guarantees the simplicity of the hardware implementation. Hardware implementations confirming system-level conclusions, however, are still to be explored in the near-term future.

ACKNOWLEDGMENT

This work was fundend by Ministerio de Ciencia y Tecnologia (Spain) under the Project TIC2003-09521 and by Xunta de Galicia (Spain) under the Project PGIDIT04PXI20606PN.

REFERENCES

[1] L.O. Chua, T. Roska, "Cellular Neural Networks and Vision Computing. Foundation and Applications", Cambridge University Press, 2002.
[2] A. Rodríguez-Vázquez et al., "ACE16k: The Third Generation of Mixed-Signal SIMD-CNN ACE Chips Toward VSoCs", IEEE Transactions on Circuits and Systems-I, vol. 51, n. 5, pp. 851–863, May 2004.
[3] Jacek Flak, Mika Laiho, Ari Paasio, Kari Halonen, "VLSI Implementation of a Binary CNN: First Measurement Results", in Proceedings 8th IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA2004, pp. 129–134, 2004.
[4] V.M. Brea, D.L. Vilariño, A. Paasio, D. Cabello, "Design of the Processing Core of a Mixed-Signal CMOS DTCNN Chip for Pixel-Level Snakes", IEEE Transactions on Circuits and Systems-I, vol. 51, no. 5, pp. 997-1013, May 2004.
[5] V.M. Brea, M. Laiho, D.L. Vilariño, A. Paasio, D. Cabello, "A One-Quadrant Discrete-Time Cellular Neural Network CMOS Chip for Pixel-Level Snakes", in Proceedings IEEE International Symposium on Circuits and Systems, ISCAS2005, pp. 5798–5801, 2005.
[6] Krzysztof Ślot, "Large-Neighborhood Templates Implementation in Discrete-Time CNN Universal Machine with a Nearest-Neighbor Connection Pattern", in Proceedings 3rd IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA-94, pp. 213–218, 1994.
[7] N.A. Fernández, D.L. Vilariño, V.M. Brea, D. Cabello, "On the Emulation of Large-Neighborhood Templates with Binary CNN-based Architectures", in Proceedings 9th IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA2005, pp. 274–277, 2005.
[8] D.L. Vilariño, D. Cabello, X.M. Pardo, V.M. Brea, "Cellular Neural Networks and Active Contours: A Tool for Image Segmentation", Image and Vision Computing, vol. 21, no. 2, pp. 189–204, 2003.
[9] A. Paasio, A. Dawidziuk, "CNN Template Robustness with Different Output Nonlinearities", Int. J. Circuit Theory Applicat., vol. 27, n. 1, pp. 87–102, 1999.
[10] IEEE Transactions on Circuits and Systems I, "Special Issue on CNN Technology and Active Wave Computing", vol. 51, n. 5, May 2004.
[11] Piotr Dudek, "A Programmable Focal-Plane Analogue Processor Array", PhD, University of Manchester, Institute of Science and Technology, May 2000.
[12] Piotr Dudek, "A General-Purpose Processor-per-pixel Analog SIMD Vision Chip", IEEE Transactions on Circuits and Systems-I, vol. 52, n. 1, pp. 13–20, January 2005.
[13] D.L. Vilariño, Cs. Rekeczky, "Implemention of a Pixel-Level Snake Algorithm on a CNNUM-based Chip Set Architecture", IEEE Transactions on Circuits and Systems-I, vol. 51, no. 5, pp. 885–891, May 2004.
[14] M. Kass et al., "Snakes: Active Contours Models", International Journal of Computer Vision, vol. 1, pp. 321–331, 1988.
[15] V. Caselles, et al., "A Geometric Model of Active Contours in Image Processing", Num. Math., vol. 66, no. 3, 1993.
[16] R. Malladi, J.A. Sethian, B.C. Vemuri, "Shape Modelling with Front Propagation: A Level Set Approach", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 17, no. 2, pp. 158–174, 1995.
[17] D.L. Vilariño, Cs. Rekeczky, "Pixel-Level Snakes on the CNNUM: Algorithm Design, On-Chip Implementation and Applications", International Journal of Circuit Theory and Applications, vol. 33, pp. 17–51, 2005.

CNNA06:

N. A. Fernández, V. M. Brea, D. L. Vilariño and D. Cabello. "**On the Reduction of the Number of Coefficient Circuits in a DTCNN Cell**," in *Proceedings of the 2006* 10^t*h IEEE International Workshop on Cellular Neural Networks and their Applications*, Istanbul, Turkey, August 2006.

On the Reduction of the Number of Coefficient Circuits in a DTCNN Cell

Natalia A. Fernández, Victor M. Brea, David L. Vilariño, Diego Cabello
Department of Electronics and Computer Science, University of Santiago de Compostela
E-15782 Santiago de Compostela, Spain
e-mail: natiafg@usc.es

Abstract— **This paper introduces a methodology to reduce the number of coefficient circuits in a DTCNN cell without penalty at application level. Trade-offs like area-processing time, and some other figures of merit like accuracy and power dissipation are considered. It is shown that it is possible to obtain efficient implementations with a reduced number of coefficient circuits. Some examples illustrate the proposal.**

Index Terms— **Hardware reduction, SIMD, CNN, PLS, trade-off area-time**

I. Introduction

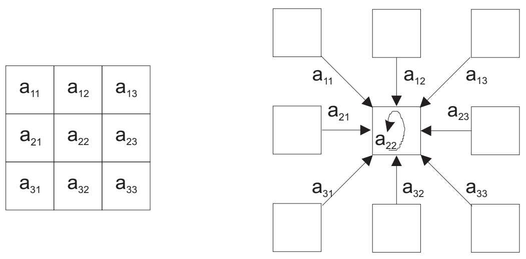
MASSIVE parallelism is one of the fundamental contributions of SIMD architecture in image processing tasks. However, at the same time, SIMD on-chip implementation is also a challenge. It is even more restrictive for classical CNN circuits comprising 18 coefficient circuits and 16 intercell connections. Many efforts have been made to reduce area consumption in such implementations. Some authors have come up with several approaches focused on coefficient circuits as an important part of the area in a CNN cell. In so doing there are two main lines: simplifying hardware implementation of coefficient circuits and reducing its number. For the sake of clarity, these two lines are analyzed separately, nevertheless better results are achieved from the combination of both. Leaving aside pure circuit improvements, the most important approach within the first option is the transition from 4Q systems to 2Q or even to 1Q [1], [2]. The limitation to binary image processing and 1-bit programmability are also important solutions for hardware simplification in a CNN cell [3]. The use of 10 coefficient circuits [4], only one template physically implemented [5] or only one coefficient circuit with time-multiplexing [6] are the main ideas in the literature to have a reduced number of coefficient circuits in a CNN cell. Supporting this, the work in [7] discusses the inefficiency of having 9 implemented coefficients per template and it suggests that a drop in the number of coefficient circuits might lead to a better performance, i.e. to a cell with better figures of merit.

In line with the above approaches, our proposal is a new methodology that attempts to shrink area by reducing the number of coefficients, without drawbacks at application level and without big penalties in processing time. The basis of our work was exposed in [8], the so called "Split & Shift" methodology to emulate large-neighborhood templates with only 3×3 coefficient circuits implemented. This means that the $N \times N$ coefficient circuits (with $N > 3$) required by the original template are turn down to 9. Now, the target of this work is to adapt those techniques to emulate 3×3 templates with a reduced number of coefficient circuits. The study presented here is general and considers full-dense 3×3 templates, so that any particular case will lead to equal or better results than the ones shown here. As in the large-neighborhood cases, it must be noted that our proposal is only valid for DTCNNs, or CTCNN operations with B templates only. The methodology presented here is also accompanied with a quantitative study of its efficiency.

This paper is organized as follows. In Section II the main steps of the methodology are presented for the case of an isolated template. Section III extends the study to CNN operations with two templates. In Section IV we introduce the modifications that must be taken into account when going through complex tasks or algorithms. We illustrate the process with an example. The special situation of large-neighborhood applications is also mentioned in this section. Hardware trade-offs are considered in Section V and, finally, conclusions and future work are gathered in Section VI.

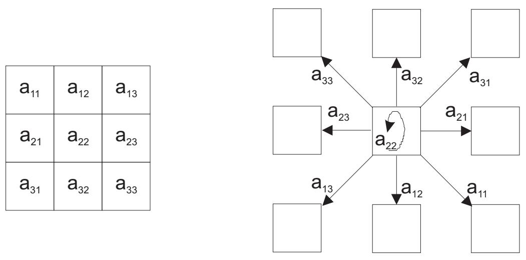
II. Reduction of the number of coefficient circuits with "split & shift" techniques

As it was mentioned before, our proposal is only valid for DTCNNs or CTCNNs operations with B templates only. This is because our methodology is based on the addition of partial results, what implies to have predictable and well-defined real-valued outputs. So, in CNNs with a piece-wise linear output-state relationship the partial outputs have to fall into the linear region, and in CNNs with a high gain output-state relationship it should be possible to get access to the state before the application of the output function. The number of coefficient circuits along with their arrangement set up the coefficient circuits configuration. General 3×3 template functionality is shown in Fig. 1, where differently from Fig. 2, the information flows inwardly instead of outwardly. Fig. 1 depicts the convention adopted at system level when listed a template. All contributions flow inwardly to the cell. Fig. 2 outlines the most frequent convention used at hardware level, when laying down the cell. All the contributions flow outwardly from the cell under study. From now on, unless otherwise stated, the hardware level convention is assumed. The cell configuration marks which of the nine template coefficients are executed, as well as which of the neighbors are connected to the cell under study. For instance we can consider a cell with three coefficient circuits like the one in Fig. 3. In this case the information flows to the right neighbors only. Having reduced the number of coefficient circuits implemented the template functionality is also limited. The functionality of a template with the configuration given in Fig. 3 is shown in Fig. 4. Both the hardware and the application (system) points of view are displayed. The methodology presented here keeps the original functionality with a reduced number of coefficient circuits.



a_{11} a_{12} a_{13}
a_{21} a_{22} a_{23}
a_{31} a_{32} a_{33}

Fig. 1. System-level convention: weighting and collecting neighbor contributions.



a_{11} a_{12} a_{13}
a_{21} a_{22} a_{23}
a_{31} a_{32} a_{33}

Fig. 2. Hardware-level convention: sending out weighted information to its neighboring cells.

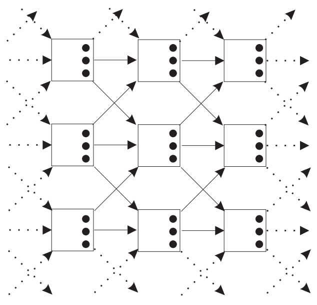


Fig. 3. CNN with only three coefficient circuits per cell. Information flow allowed to the NE, E and SE neighbors of a cell.

For a given configuration (number and allocation of the reduced number of coefficient circuits) we must rearrange the

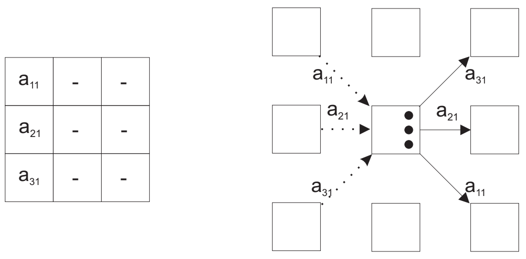


Fig. 4. Corresponding template functionality for the configuration shown in Fig. 3. Hardware point of view with solid arrows. System point of view with dashed arrows.

coefficients of the original template into several sub-templates with the selected configuration. We have to avoid repeating the same original template coefficient in two different sub-templates. Also, we must preserve the original relative positions among coefficients gathered in the same sub-template. This is the “split” phase of the methodology. The following phase consists of applying the sub-templates over the image and gather all neighbors contributions within the cell under study, emulating the functionality of the original template. To achieve this objective we have two options. The first one is to shift the image for each sub-template to be applied over the adequate neighboring pixels (see Fig. 5). In so doing, all neighbors contributions are directly collected in the cell under study. Another possibility is to apply all sub-templates over the original image and shift the partial outputs to the cell that must collect the corresponding weighted contributions (see Fig. 6). This is the “shift” phase of the methodology.

For the shift phase of the methodology, there are, sometimes, shifts that are redundant, i.e. shifts related to different sub-templates that are overlapped, and so that they can be shared. This leads to a lesser number of operations and thus, to a better processing time. Nevertheless, this option implies that either a previously shifted image or the accumulation of the previous partial outputs must be saved. It is translated, on some occasions, into a greater memory usage. In any case two memories are always needed, one to keep the original (or shifted) image and an analog one to accumulate and save partial results. In general it is beneficial to share shifts whenever it is possible.

Fig. 5 and Fig. 6 show, respectively both options image and partial result shifting, each with the two possible methods, to share or not to share shift processes. The grid in the upper part of the figures represents a reduction from 9 to 3 coefficient circuits and from 8 to 3 inter-cell connections. With only 3 coefficients permitted we have to split the original template into three sub-templates to have the 9 original coefficients placed over allowed positions. Note that template positions are the mirror image of coefficient circuit positions (see Fig. 3 and Fig. 4). Shifts that are required to gather all neighbors contributions in the adequate cell are identified with numbers 1 and 2. They are represented over the grids with convex arrows for partial result shifting and with straight ones for image shifting (Fig. 5 and Fig. 6). Operation sequences for both image and output shifting are also outlined. Between brackets it is shown that if we share shifts we only need shifts of type 1, what implies one less operation in the application of the technique.

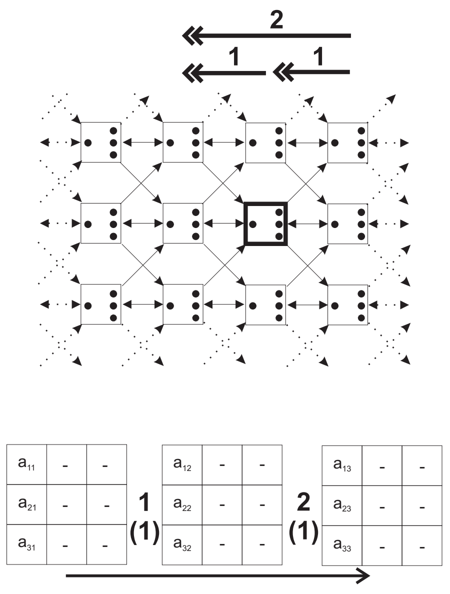


Fig. 5. Sequence of operations to approach the functionality of a full-dense 3×3 template with only three coefficient circuits with a given configuration employing image shifting. Cell under study marked with a thick square.

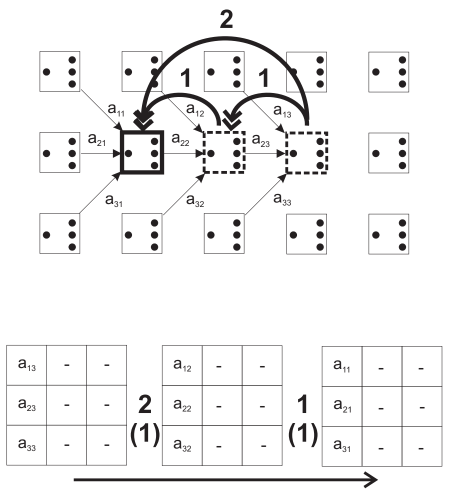


Fig. 6. Sequence of operations to approach the functionality of a full-dense 3×3 template with only three coefficient circuits with a given configuration employing partial outputs shifting. Cell under study marked with a thick square.

It must be noted that there are not limitations to the coefficient circuits configuration imposed by the original template split. Nevertheless, the shifting process forces to have not only a minimum number of coefficient circuits, but also a certain

Number of Coefficient Circuits	Number of Operations (Sub-templates + Shifts)
9	1
8	2+1
7	2+1
6	2+1
5	3+2
4	3+2
3	5+5
~~2~~	
~~1~~	

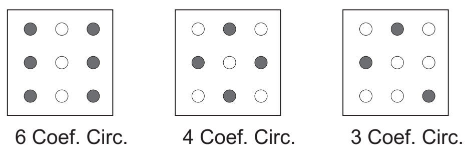


Fig. 7. Number of operations for the most efficient configuration of each number of coefficients circuits. Among equal number of operations we choose those with less coefficient circuits. These configurations are displayed in a circle. We also depict some realizable configurations with 3, 4 and 6 coefficient circuits.

arrangement within the 3×3 neighborhood of the cell under study. This is why the configuration seen in Fig. 3 is not realizable. It is not possible to shift to the left by means of a CNN operation with such a coefficient circuits arrangement. Either extra coefficient circuits or another allocation for the three coefficient circuits would be required. As another option, we can also implement shift operations through specific hardware like direct switched-connections. This means that we could avoid coefficient circuits that are only used for shifting. On this basis, the configuration depicted in Fig. 3 would be feasible. However, to simplify hardware considerations, from now on we choose to make shifts with CNN operations.

Both the number of operations (processing time) and the number of coefficient circuits (area) are key factors in assessing the performance of our methodology. In Fig. 7 we show the minimum number of operations for the most effective configurations (configurations that imply the lowest number of operations to emulate a full dense 3×3 template) of a given number of coefficient circuits. Among different configurations and coefficient cicuits we always choose those with the lowest number of operations. Configurations with the highest efficiency within each selected number of coefficients, except 9, are depicted. Configurations with 1 and 2 coefficient circuits cannot be implemented because the shifting is not doable. Note that a conventional SIMD architecture employs only one ALU. The difference with a CNN, however, is that classical SIMD implementations count on multiplexes to set up communications with the neighbors along the four cardinal directions, while CNN communications are performed with the coefficient circuits themselves, and here almost all of them are removed.

To compare the selected configurations we define an efficiency factor, the RPO (percentage of hardware Reduction Per

Number of Coefficients Circuits (nc)	Hardware Reduction (HR)	Operations Increment Factor (OIF)	RPO (%)
9	0	1	0/0
6	1/3	3	16.67
4	5/9	5	13.89
3	2/3	10	7.41

Fig. 8. Percentage of hardware reduction per CNN operation increased per original CNN operation (RPO) in 3×3 template approaches with different configurations.

CNN Operation increased for each original CNN operation). This parameter establishes the relationship between the benefit obtained in area and the harm in time-consumption. It can be a good way to compare different configurations, but design requirements mark the actual goals to be achieved. Eq. (1) gives the RPO definition. HR is the Hardware Reduction factor, OIF is the Operation Increment Factor and nc is the number of coefficients circuits (we assume that one of the best configurations for a given number of coefficient circuits is always selected).

$$RPO(nc) = \frac{HR(nc) \cdot 100}{OIF(nc) - 1} \tag{1}$$

Values of HR, OIF and RPO for the configurations selected in Fig. 7 are shown in Fig. 8. In general, we can see that the 3-coefficient configuration is much less efficient than the ones with 4 and 6 coefficient circuits. This is due to the difficulty of making shifts.

III. Application to a typical complete CNN operation comprising two templates

After having introduced our methodology for an isolated template, we look now at classical two-template CNN operations. Note that the bias term is not considered here because it does not need to be split and can be added in one or several steps while the new sub-templates.

The most straightforward solution is to reduce the number of coefficient circuits of each template as if they were isolated templates. In this case we can apply simultaneously the sub-templates from both original templates (A and B). Shifts can only be run simultaneously if either $Y = U$ (the same shifted image is valid for both templates) or if we make partial outputs shifting. In these cases we have the same performance as with isolated templates. In any other case shifts have to be performed in different CNN operations and their outputs (shifted images) must be saved separately. The efficiency drops with respect to the cases of shift-sharing and isolated templates. This is summarized in Fig. 9, where all the numbers were extracted under the consideration of equal configuration for both templates.

Hardware sharing between both templates in order to reduce area consumption was already proposed in [5]. Combining this with our proposal to reduce the number of coefficient circuits, we obtain better efficiency values than in the non hardware-sharing option for configurations with 6 coefficient circuits and no much worse for the ones with 3 and 4 (Fig. 9). Again we can consider shift-sharing in the cases with $Y = U$ or partial outputs shifting. With this improvement we achieve better results in all cases (see Fig. 9).

As mentioned before, another approach would be to specialize hardware, having specific circuits for shifting, differently from the conventional coefficient circuits used for sub-template application. Similarly to classical synchronous SIMD architectures, the shifting would be made by means of switches. Such a specialization would allow configurations with only one or two coefficient circuits.

Number of Coefficient Circuits (nc)	A&B independent hardware RPO (%)	A&B hardware sharing RPO (%)	A&B hardware and shift sharing RPO (%)
9	0/0	50	0/0
6	11.11	13.13	16.67
4	9.26	8.64	11.11
3	4.76	4.39	5.95

Fig. 9. RPO values under the consideration of two templates with specific hardware for each one (2nd column), hardware sharing (3rd column) and hardware and shift sharing (4th column).

IV. Application to complex tasks

In particular cases, the performance can be improved, even to $RPO \rightarrow \infty$. This means that there is a complete matching between the original template non null coefficients distribution and the simplified coefficient circuit configuration. We achieve hardware improvement without increasing the number of operations. Apart from that, for only one CNN original operation, the maximum RPO value is given by the percentage of hardware reduction. This happens when the number of operations is only increased in one, i.e. $OIF = 2$. Nevertheless, when we consider complex tasks comprising several CNN operations, it is possible to reach RPO values larger than 100%. This means that HR outweights OIF (see Eq. (1)). Next, these and other features of our methodology are illustrated in a complex task with several CNN operations.

A. A typical complex task

We illustrate our methodology with the design of a module of a complex active contour algorithm, the Pixel Level Snakes (PLS) [9]. We focus on the Topologic Transformation (TP) module because this is the most exhaustive time-consuming module in the PLS due to the hole-filling task. Apart from the hole-filling, TP comprises an opening (erosion & dilation) and a binary edge detection.

The starting point in this example is the synchronous binary architecture with 1-bit of programmability introduced in [10]. This imposes a fundamental constrain: the use of image shifting. In this implementation there are two types of processing steps, some with one and some with two templates. The former case is a template with 9 possible non-zero coefficients. The latter case is executed with two templates, both of them having only one non-zero template coefficient, the central one. In fact, the number of coefficient circuits implemented in [10] is ten. This is our starting point. Hence we start with 10 coefficient circuits.

Fig. 10 displays the operations implemented in the B/W architecture presented in [10]. A simple visual inspection reveals that it is enough to have a four-neighborhood configuration in one of the templates and only the central term in the other one. This would lead to 6 coefficients (5+1 configuration). This area improvement comes without penalty at processing time, which means an $RPO \to \infty$. Obviously, a more realistic approach demands to study the rest of operations involved in the PLS algorithm. This will result into a global finite RPO.

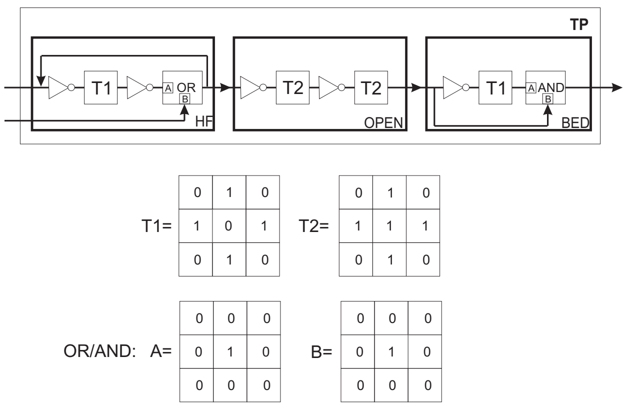


Fig. 10. Operations and templates in the topologic transformations (TP) module of the Pixel Level Snakes (PLS) algorithm reported in [9] with the implementation presented in [10].

A further analysis shows that when run at the same time the two templates are just used to perform Boolean operations. It is possible, then, to choose a 4+1 configuration and execute the logical functions in two steps. In this case, RPO drops to almost 100%. Furthermore, seeking bigger area improvements, we can select an option with two templates in a 3+1 configuration. This is the situation illustrated in Fig. 11. Cell configuration is represented with dots for the A coefficient circuits and with crosses for the B ones. Sub-templates and processing steps to emulate the original CNN operations functionality are also shown. This configuration would lead to an RPO around 60%.

B. Large neighborhood operations as special complex tasks

Large neighborhood templates implementation is a special case within complex tasks. As it is introduced in [8], this kind of tasks can be implemented with minimum size (3×3) templates. The “Split & Shift” techniques used there are the basis for the hardware reduction proposed here. Hence, a further step would go through the implementation of large

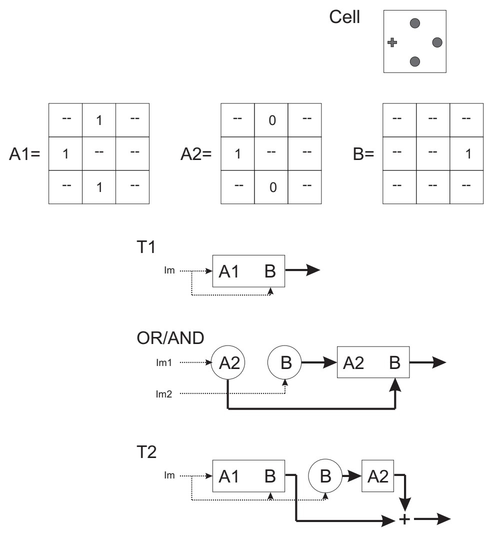


Fig. 11. CNN operations in the TP module implemented with a 4 (3+1) coefficient circuits configuration. Templates enclosed by a circle make shifts. Templates enclosed by a square are the sub-templates executed.

neighborhood templates with a reduced number of coefficient circuits.

The most straightforward solution is to implement each sub-template involved in the large neighborhood emulation with the reduced configuration as isolated templates. This option will increase significantly the number of processing steps: we need several processing steps to perform each processing step from large neighborhood emulation. Nonetheless it will be a better solution to re-split the original large neighborhood template taking into account the new configurations. In so doing we can overpass the limits imposed by the original split, i.e. we can gather into the same sub-template coefficients that were originally placed in different sub-templates. The possibility of doing this will depend on the configuration selected. On the other hand, shifts from both techniques could be shared and the use of two templates would allow to perform shifting and sub-template application at the same time. This might yield acceptable figures of merit in both area and processing time.

V. Hardware implications

Area, speed, accuracy and power dissipation are among the main parameters shown up when designing a circuit. As reported in [11], none of the these parameters can be optimized independently. Many trade-offs come out. In this section we see how our proposal affects these parameters and trade-offs. This means to be a preliminary study. An on-chip implementation would be more conclusive.

A. Area vs. processing time

Our proposal can be placed midway between a DTCNN and a classical synchronous SIMD architecture. The number of coefficient circuits is smaller than that of a CNN but greater than that of an SIMD implementation, which usually has only one ALU. Concerning processing time, our proposal needs more processing steps than a classical CNN with 18 coefficient circuits. Nevertheless, and although they were not compared, it is likely that our solution would lead to less processing steps than classical synchronous SIMD circuits.

Keeping our analysis within the CNN field, it is apparent that there is a trade-off between area and processing time. Throughout the paper the reduction of the number of coefficient circuits was taken as the measure of area improvement. Nevertheless, this is a little bit over-simplified. Coefficient circuit area consumption could represent, for instance, a 50% of the total of the area of the cell. With this the real hardware reduction will be $RH \times 0.5$. Moreover, for every less coefficient circuit, except for the central one, there is also one less connection to the neighbors, what implies smaller area for routing. The memory usage, however, grows and it can be necessary to implement more memories. Furthermore, when applied to binary architectures (B/W image processing) our methodology obliges to include an analog memory to collect all partial results.

On the other hand, an excessive number of processing steps (very few coefficient circuits and/or complicated configurations) might make an application too slow, especially for those implementations with a large resolution (e.g. 128×128 cells). It is worth noting here that this might even happen to binary architectures like the one reported in [12], where processing steps of tens of nanoseconds were measured on a chip of 4×4 cells. The reason for this is that now we need many more accesses to the global memory where the templates are, and the fan-out of the memory buffers grows with the size of the image (i.e. CNN resolution). Finally, the RPO is just an indicator of how much the trade-off between area and processing time can be improved for each single CNN operation, task or even an algorithm. However, the basic constraint is to meet the design goals, so the best solution can imply not to get the best RPO.

B. Mismatch, accuracy and power consumption

Hardware reduction makes it possible to enlarge devices preserving the same area in the cell. This leads to mismatch minimization too. The resultant circuits turn to be more accurate. As the accuracy requirements pose a minimum power dissipation [11], it is likely that a circuit realization of the methodology addressed here consumes less power than conventional CNNs. Furthermore, less number of coefficients being executed at the same time will reduce power consumption per clock cycle. This is something, however that cannot be proved for the time being, but with a circuit design.

VI. Conclusion

A methodology to reduce the number of coefficient circuits in a DTCNN cell without penalty at application level has been addressed. Studies about the performance of such a methodology has also been included. These studies show that as long as the processing time does not grow excessively, our methodology would result into an on-chip realization with very competitive figures of merit. This is true for area and accuracy, and possibly for power dissipation too. Nevertheless, this still has to be proven with a circuit implementation.

Acknowledgment

This work was partly funded by Ministerio de Ciencia y Tecnología (Spain) under the project TIC2003-0951 and the "Programa Nacional de Formación de Profesorado Universitario" (REF.: AP-2004-3741), as well as by Xunta de Galicia with the project PGIDIT04PXI20606PN.

References

[1] J. Hegt, D. Leenaerts, and R. Wilmans, "A novel compact arquitecture for a programable full-range CNN in 0.5 um CMOS technology," in *1998 Fifth International Workshop on Cellular Neural Networks and their Applications*, V. Tavsanoglu, Ed., London, England, Apr. 1998, pp. 288–293.

[2] V. M. Brea, D. L. Vilariño, A. Paasio, and D. Cabello, "On the one-quadrant template design in a high gain CNN model," in *Proceedings of the 8th International Workshop on Cellular Neural Networks and their Applications*, Budapest, Hungary, July 2004, pp. 279–284.

[3] A. Paasio, J. Flak, M. Laiho, and K. Halonen, "High density vlsi implementation of a bipolar cnn with reduced programmability," in *Proceedings of the 2004 IEEE International Symposium on Circuits and Systems. ISCAS '04*, vol. 3, Vancouver, Canada, May 2004, pp. 21–24.

[4] A. Paasio, A. Kananen, and V. Porra, "A 176x144 processor binary I/O CNN-UM chip design," in *Design Automation Day on Cellular Visual Microprocessor, European Conference on Circuit Theory and Design*, T. Roska, C. Beccari, M. Biey, P. Civalleri, and M. Gilli, Eds. Stressa, Italy: Levrotto&Bella, Torino, 1999, pp. 82–86.

[5] A. Paasio, M. Laiho, A. Kananen, and K. Halonen, "An analog array processor hardware realization with multiple new features," in *International Joint Conference on Neural networks*, Honolulu, Hawaii, 2002, pp. 1952–1955.

[6] F. Sargeni, V. Bonaiuto, and M. Bonifazi, "Time division digital programmable OTA for cellular neural networks," in *European Conference on Circuit Theory and Design, ECCTD 2005*, Cork, Ireland, 2005.

[7] P. Dudek, "Accuracy and efficiency of grey-level image filtering on vlsi cellular processor arrays," in *Proceedings of the IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA 2004*, Budapest, Hungary, July 2004, pp. 123–128.

[8] N. Fernández, D. Vilariño, V. Brea, and D. Cabello, "On the emulation of large-neighborhood templates with binary cnn-based architectures," in *Proceeding of the 9th IEEE International Workshop on Cellular Neural Networks and their Applications. CNNA 2005*, Hsinchu, Taiwan, May 2005, pp. 274–277.

[9] D. L. Vilariño and C. Rekeczky, "Implementation of a pixel-level snake algorithm on a cnnum-based chip set architecture," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, no. 5, pp. 885–891, May 2004.

[10] V. M. Brea, M. Laiho, D. L. Vilariño, A. Paasio, and D. Cabello, "One-quadrant discrete time cellular neural network architecture for pixel level snakes: B/W processing," in *Proceedings of the IEEE International Symposium on Circuits and Systems, 2005. ISCAS '05*, Kobe, Japan, May 2005, pp. 3922–3925.

[11] P. R. Kinget, "Device mismatch and tradeoffs in the design of analog circuits," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 6, pp. 1212–1224, June 2005.

[12] J. Flak, M. Laiho, A. Paasio, and K. Halonen, "VLSI implementation of a binary CNN: First measurements results," in *Proceedings of the 8th International Workshop on Cellular Neural Networks and their Applications*, Budapest, Hungary, July 2004, pp. 129–134.

DCIS06:

N. A. Fernández, V. M. Brea, D. L. Vilariño and D. Cabello. " **Hardware Simplification in Cellular Non-linear Networks for Complex Algorithms** ", in *Proceedings of the XXI Conference on Design of Circuits and Integrated Systems, DCIS 2006*, Barcelona, Spain, November 2006.

Hardware Simplification in Cellular Non-linear Networks for Complex Algorithms

Natalia A. Fernández García, *Student, IEEE,* Victor M. Brea Sánchez, *Member, IEEE*
David López Vilariño, *Member, IEEE* and Diego Cabello Ferrer, *Member, IEEE*

***Abstract*— This work aims to shrink area in Cellular Non-linear Networks (CNN). Its basis is the reduction in the number of weighting multipliers and connections in a CNN cell, classically 18 and 16 respectively. The methodology presented here comprises two phases. The first phase splits the templates into smaller sub-templates. The second phase shifts and collects the contributions of every sub-template in the cell under study. Hardware implications of such a methodology are also discussed. All these features are addressed for the case of a complex active contour algorithm, Pixel-Level Snakes (PLS).**

***Index Terms*— Hardware reduction, SIMD, CNN, PLS, trade-off area-processing time**

I. Introduction

CELLULAR Non-linear Networks (CNN) have traditionally been posed as an example of a practical solution to the on-chip realization of SIMD architectures for image computation with direct pixel to cell assignment [1]. The local connectivity is one of the key factors in having CNN on-chip implementations with high resolution (number of pixels/cells). Nevertheless, low area consumption per cell continues being a design challenge. Focusing the effort on multipliers as an important part of the CNN cell area, two main lines were followed for some authors to achieve smaller area consumptions: simplifying hardware implementation of the multipliers and reducing its number. Work about transition from 4Q to 2Q or 1Q systems [2], [3] and limitation to binary image processing and 1-bit programmability [4] are the most important approaches within the former option. The use of 10 multipliers [5], only one template physically implemented [6] or only one multiplier with time multiplexing [7] are the main ideas within the latter. A basis for this approach can be found in [8]. Nevertheless, it is worth noting that both lines can be combined with much better results.

On the other hand, a methodology to deal with large-neighborhood tasks efficiently and without penalty at hardware level in such implementations was introduced in [9]. In that work, large-neighborhood templates (5×5 or more) were executed with 9 weighting multipliers and 8 interconnections to the surrounding cells, i.e. by means of 3×3 templates. The methodology basically consisted of two phases. Firstly the large templates were split into 3×3 sub-templates. Secondly the contributions from every sub-template were collected in the current cell. The second phase could be done in two different ways, either by shifting the image before applying the sub-templates or by shifting the partial outputs of every sub-template. The difference is that when shifting the image outputs are obtained in the correct cell and we do not need to shift the outcome of every sub-template. When shifting the partial outputs, every sub-template is applied over the original (not shifted) image. Thus, we have to shift the partial outcomes of every 3×3 sub-template to collect all of them in the adequate cell.

The current paper applies the above methodology to run 3×3 templates with a reduced set of weighting multipliers physically implemented. The main goal is to improve the figures of merit (especially area) at hardware level without penalizing excessively the execution time. This comes out as an area-processing time trade-off. These and other features of our methodology are illustrated with a complex algorithm, an active contour based algorithm, namely Pixel-Level Snakes (PLS) [10].

This paper is outlined as follows. Section II introduces the methodology to reduce the number of weighting multipliers in a CNN cell. In Section III, the algorithm to be implemented and the current hardware realization system level are presented. In this section the hardware modifications needed to apply the methodology are analyzed too. Section IV goes through the selection of the best number and allocation of the multipliers. Finally some conclusions and proposals of future work are extracted in Section V.

II. "Split & Shift" techniques to preserve functionality in hardware simplified

As in large-neighborhood template emulation techniques, the methodology proposed here is valid for both DTCNNs and CTCNNs operations with B templates only. The reason is that our methodology is based on the addition of partial results, what implies to have predictable and well-defined real-valued outputs. So, in CNNs with piece-wise linear function the partial outputs must fall into the linear region. In CNNs with high gain output-state relationship it must be possible to get access to the state before the application of the output function.

This work was partly funded by Ministerio de Ciencia y Tecnología (Spain) under the project TIC2003-0951 and the "Programa Nacional de Formación de Profesorado Universitario" (REF.: AP-2004-3741), as well as by Xunta de Galicia with the project PGIDIT04PXI20606PN.

All authors are with the Department of Electronics and Computer Science, University of Santiago de Compostela, E-15782 Santiago de Compostela, Spain (phone: 0034 981563100, ext. 13580; fax: 0034 981528012;e-mail: natiafg@usc.es)

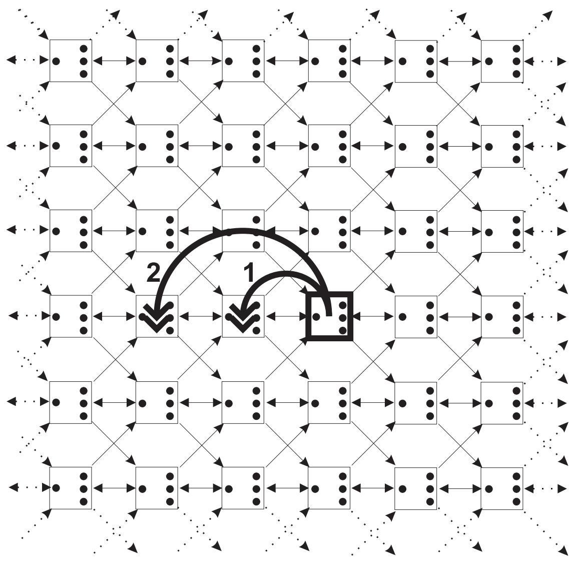
The configuration (number of multipliers and distribution) marks which of the nine coefficients are executed and which of the neighbors are connected to the cell under study. As a consequence, it determines how the split and shift steps must be applied and the hardware simplification performance. Fig. 1 shows the minimum number of steps that can be achieved for every possible number of considered multipliers. One and two multipliers are not possible options because they do not allow the required shift steps due to its limited connectivity. We can take as an example of configuration the one with four weighting multipliers shown in Fig. 2, where the weighting multipliers are plot as dots. There the information flows to the right and to the central-left neighbors. Note here that at system-level templates are matrices formulated with the information flowing from the neighboring cells toward the central cell. Thus, the weighting multipliers of the neighboring cells send their contributions into the cell under study. In a hardware implementation, the weighting multipliers are usually laid out with the information flowing out of the cell. The latter is sketched with the straight arrows in the grid of Fig. 2.

Number of Weigthing Multipliers	Number of Emulation Steps (Sub-templates + Shifts)
9	1
8	2+1
7	2+1
6	2+1
5	3+2
4	3+2
3	5+5

Fig. 1. Minimum number of steps for every number of weighting multipliers.

The first phase in our methodology is to split the original 3×3 template. The configuration depicted in Fig. 2 leads to three sub-templates. In the new sub-templates the coefficients have to be arranged in allowed sites (mirror possitions of the configuration marked with dots in Fig. 2), and in such a way that the original relative positions among template coefficients are preserved and each coefficient is only used once.

The second phase in the methodology is to shift and collect the contributions of every sub-template in the current cell. As it was said before, there are two ways, either image or partial results shifting. Furthermore, we have the option of sharing or not overlapped shifts. The latter saves processing time. Fig. 2 displays both options, image and partial result shifting, with the two possible methods, to share or not to share shifts. The grid in the upper part of the figure represents a reduction from 9 to 4 coefficient circuits and from 8 to 4 inter-cell connections. Shifts that are required to gather all neighbors contributions in the adequate cell are identified with numbers 1 and 2. They are represented over the grid with convex arrows. Operation sequences for both image and output shifting are also outlined. Between brackets is shown that if we share shifts we only need shifts of type 1, what implies one less operation in the application of the technique. It is also worth pointing out that apparently the left-central weighing multiplier is not used as its value is always set to zero in the sub-templates (see bottom of Fig. 2). Nevertheless, it should be noted that this weighting multiplier is used for shifting, not for sub-template application. In this case, possibly a simpler and straightforward solution would be to use specialized hardware (e.g. and additional data bus for inter-cell connectivity). This might simplify the shifting procedure. Nevertheless our analysis is restricted to architectures where every task is tackled with CNN operations, that is, every operation is done through the weighting multipliers. For further information about the hardware reduction methodology and its hardware-time processing implications the reader is addresed to [11].



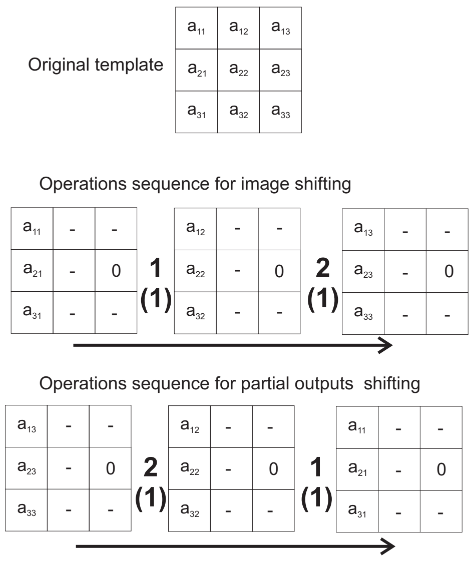

Original template
a11 a12 a13
a21 a22 a23
a31 a32 a33
Operations sequence for image shifting
a11 - - / a21 - 0 / a31 - -
1 (1)
a12 - - / a22 - 0 / a32 - -
2 (1)
a13 - - / a23 - 0 / a33 - -
Operations sequence for partial outputs shifting
a13 - - / a23 - 0 / a33 - -
2 (1)
a12 - - / a22 - 0 / a32 - -
1 (1)
a11 - - / a21 - 0 / a31 - -

Fig. 2. CNN with only four weighting multipliers. Sequence of operations for the approach of a full-dense 3×3 template with either image or partial outputs shifting.

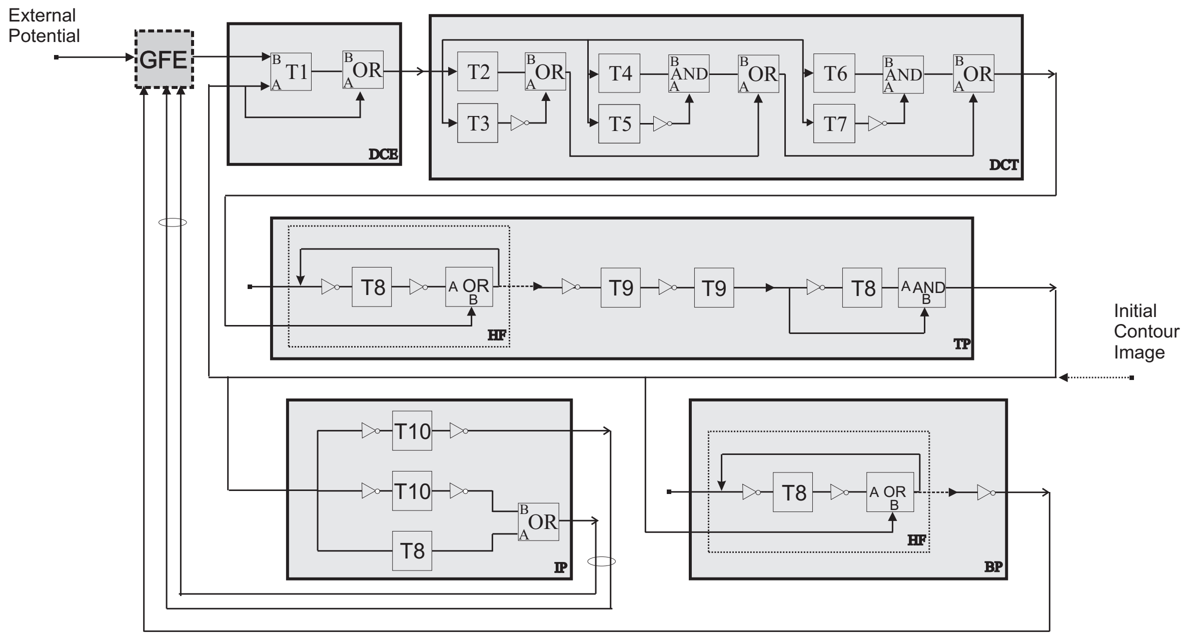


Fig. 3. Main tasks in the PLS technique with special emphasis on B/W processing. Gray-scale tasks are gathered into the dashed square. Iterative Hole-Filling (HF) operation remark into a dashed rectangle.

III. Pixel Level Snakes techniques. 1Q-1bit binary implementation

Pixel Level Snakes (PLS) is an active contour-based technique mainly oriented to contour tracking and segmentation. Its development at pixel level makes it very suitable for a CNN realization. The PLS version tackled here was introduced in [10]. PLS techniques comprise both gray-scale and B/W processing. The former is meant to extract the guiding information. The latter moves and deforms the contours according to the guiding information.

All these B/W operations are approached with a 1-bit programmable DTCNN architecture [12]. Fig. 3 outlines the flow diagram of the PLS version addressed in [12]. All the operations are run iteratively along the four cardinal directions within each algorithm iteration. B/W operations are enclosed in squares with solid lines. These operations make the move and deformation of the active contours. The active contours are one-width walls of black pixels on a white background. Gray-scale operations are meant to extract the guiding information image. This is done by the Guiding Force Extraction (GFE) module indicated in a square with dashed lines in Fig. 3. This block is implemented with specific hardware and not with CNN operations in the work presented in [12]. As the objective of the current paper is to simplify CNN hardware, the methodology presented here is only applied to B/W in the PLS algorithm.

Concerning B/W processing, the PLS algorithm sketched in Fig. 3 contains several modules. These modules have different functions in the algorithm. In Directional Contours Expansion (DCE) the contours are expanded along the current processing direction as long as the guiding information permits such a move. The contours get back to the one-wide wall shape in Directional Contours Thinning (DCT). The contours are split and/or merge in Topologic Transformations (TP). Internal Potential (IP) keeps the contours shape smooth. Finally, Balloon Potential (BP) assists the external potential in guiding the contours. For further information about PLS the reader is addressed to [10].

In Fig. 3 every module comprises several DTCNN processing steps, each indicated with *Ti*, *AND* and *OR*. Some of these processing steps require two templates. Others need only one template. Processing steps with two templates are marked with *A* and *B*. Sometimes the variables have to be inverted. This is shown with the inverter symbol. Fig. 4 depicts all the templates used for B/W processing in Fig. 3. Note that the bias term is never listed. The reason is that the bias term can be added at any time while the sub-templates are being applied.

Concerning the hardware implementation, it is important to mention that we have chosen a binary 1Q-1bit architecture [12]. This means that we can only process binary images and that templates can only have two values, 0 and 1. This implementation comprises ten coefficient circuits and six logical memories. This number of memories is enough for the application of our methodology. Nevertheless, we would have to implement at least one analog memory to accumulate the partial outcomes. It can be a current mode memory (SI, S^2I) which allows to realize the sum of the outcomes directly. The access to the cell state is easy to be implemented.

IV. Hardware reduction in the PLS B/W 1bit binary implementation

Given that PLS aims at real-time applications, we must take care with the numbers of operations incremented, as we have to comply with the speed of 25 frames/s. Special attention should be paid to the iterative Hole-Filling (HF) task in BP and TP. This might be especially troublesome for large images.

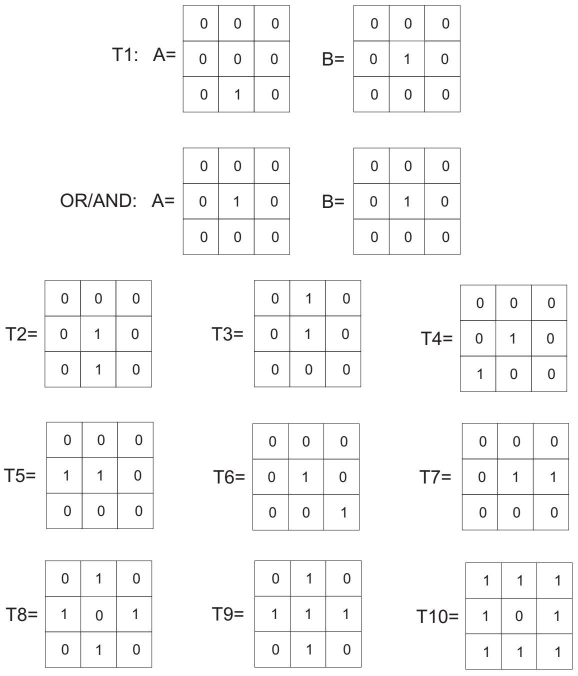


Fig. 4. Templates used in the PLS realization in [12]. Correspondence with Fig.3.

For instance, in the case of a 128×128 resolution, HF could take up to 64 CNN processing steps. In order to state how aggressive our hardware reduction can be, next we account for all the operations along the data path on an image on a hypothetical PLS on-chip implementation. We make rough and conservative estimates on how long it takes to run PLS on an image, and with this we can finally estimate how many more processing steps are possible in order to still have video rate processing.

Assuming a chip with photosensors, all the pixels are uploaded in parallel. This is the integration time. Ideally this time is controllable (programmable), yielding an adaptive sensory system. This way images with different contrast levels can be acquired. The integration time ranges μs to ms. For our analysis we can take 50 μs as the integration time [13].

The next stage on the data path of an image is the processing phase itself. This phase comprises global and local processing. On the one hand, processing an image goes through the reading and delivering of the template matrices and possibly any other signals from a global memory to the CNN array. On the other hand, the template matrices and accompanying signals are subsequently run in the 3×3 neighborhood of the cell. In current sub-0.18 CMOS technologies 100 ns (global and local processing combined) for every processing step (CNN operation execution) is not a big challenge for a B/W 1Q 1-bit implementation [12], [14].

The final stage on the data path is to download the image. As the number of pins is upper bounded, this has to be done serially in a row-wise scheme. In the case of the PLS algorithm, the images read out of the chip are always the contours, hence B/W images (binary signals). Assuming 32 pins for downloading, this amounts to less than 1 ms for reading out an image with 128×128 pixels [14].

More concerned with the PLS processing itself, as the initial contours are usually closed to their final location, ten iterations are more than enough. Observe for example the case of a video sequence. The final contours of a snapshot are the initial contours of the next frame. If the integration time is small enough compared to the speed of moving objects, the contours will be quite close to their final location. Now, assuming video rate processing (25 f/s), there is a 40 ms time slot for the complete data path of an image. Taking 1 ms for downloading and uploading purposes, we would still have more than 30 ms for doing image processing on the acquired scene. This time is rather long to achieve important area savings with the hardware reduction methodology applied here.

Thus, we take 100 ns per CNN operation, 1 ms to upload and download the image and we suppose the worst HF case within a 128×128 image. With this we have that it is possible to execute up to $390\,000$ CNN operations per frame within the real time processing rate. Note that the gray-scale module (GFE) is considered as an only operation of 100 ns [12] and that every iteration comprises the four cardinal directions. Assuming the worst case for the HF in a 128×128 image, 11 120 CNN operations are required per frame to realize ten PLS complete iterations. With this we have that we can use up to 35 new CNN operations per every original CNN 3×3 template without penalty in the real time processing rate goal. Nevertheless, as it can be seen in Fig. 1, 10 is the maximum number of CNN operations required by the barest configuration (3 multipliers).

Concerning our methodology, to choose the most adequate configuration, we must analyze the shape of all the templates in the algorithm (see Fig.4). With this and taking into account that we must avoid to penalize the Hole-Filling, we propose a 5+1 *diamond* configuration, i.e. a reduction to five multipliers in template *A* and a reduction to one in template *B*. Furthermore, due to the binary implementation we start from, we have to use image shifting emulation. Fig. 5 shows this cell configuration, depicting the *A* weighting multipliers as dots and the *B* one as a cross. As can be seen in Fig. 4, with this choice we can perform directly six of the nine original one-template CNN operations and, thanks to the one-coefficient *B* template, both two-template CNN operations (*T1* and *AND*/*OR*). Only three templates must be emulated with a reduced set of weighting multipliers. The sequences of CNN operations needed to perform them are shown in Fig. 5. In this figure shifts are signed with circles and sub-template applications with squares. In so doing we obtain a 40% reduction in the area occupied by multipliers and connections (from 10 to 6 multipliers and from 9 to 5 connections). Considering that in the implementation we started from ([12]) the area occupied by the B/W CNN arquitecture represents the 65% of the cell area and that the area occupied by multipliers and conections is the 70% of that we have that multipliers reduction implies a 18,2% reduction of the total cell area. This means that we pass from a cell of $32 \times 44 \mu m^2$ to a cell of $32 \times 36 \mu m^2$ and that in an array of 128×128 cells of 23 mm^2 the reduction is of 4,2 mm^2. The penalty at time processing level is about 2.2%, i.e. we have 1.022 operations per original operation, what is far away from 35, where the video rate processing would be still preserved.

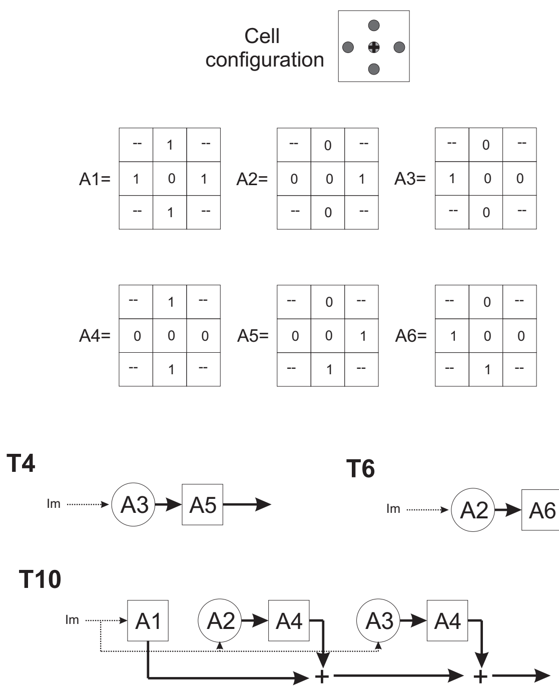


Fig. 5. System level implementation of the three original operations that cannot be implemented directly: *T4*, *T6* e *T*10 (see Fig. 4). Cell configuration for a hardware reduction from 10 to 6 multipliers. *A* weighting multipliers are represented by dots, the *B* multiplier by a cross. Templates required to emulate these operations are shown too.

Nevertheless this is a very conservative hardware simplification. As it is was said before, a barer configuration with only 3 multipliers would imply, in the worst case, 10 operations. They will be 20 if we consider the worst case of a two template CNN operation. In the case of the two template operations considered here there would be needed at most four operations for their emulation. The benefits would be a 70% multiplier and connection hardware reduction which means a 31,85% of the total area of the cell and 7,35 mm^2 in a 128×128 array.

A. Large-neighborhood implementation with hardware simplified

Recovering the original orientation of the basic methodology we propose now to combine both, the original and the new proposal to perform large-neighborhood operations with nearest neighbor connectivity and hardware simplified. This is very convenient for the application of the PLS algorithm. The Internal Potential module is aimed to smoothen the contour shape and was originally proposed as a diffusion task [10]. This proposal was changed to be adapted to the implementation we started our hardware simplification with. In this adaptation, Internal Potential is extracted as a digital word through four CNN operations. We return to the original diffusive proposal and we make it possible to smoothen bigger irregularities (rough concavities) along the contours with a large neighborhood diffusion operation.

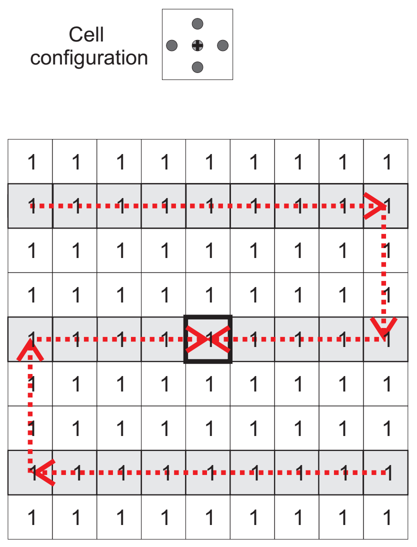


Fig. 6. Original binary large-neighborhood (9×9) template. Central coefficient and cell under study marked with a thick line square. Cell configuration for a 5+1 reduction: *A* coefficients represented by dots, *B* coefficient represented by a cross. Sub-templates centers shaded. Shifting directions illustrated with arrows.

The most straightforward solution is to use the methodology proposed in this paper to perform all the sub-templates as if they were isolated templates of a given algorithm. Nevertheless the number of operations obtained this way can be dramatically shrunk if we take into account that it is possible to re-split the original large neighborhood template with the new restrictions. In so doing, we can skip the initial sub-templates limits and gather in the same operation coefficients from two different initial sub-templates. Furthermore we notice that shifts required for hardware simplification can overlap the shifts required for large neighborhood realization if we choose adequately the way of applying the obtained sub-templates.

Zigzag shifting in Large-Neighborhood methodology [15] is the one that requires less number of shift directions (zigzag technique needs four cardinal directions to be applied). It means that it is easier to have most of the directions directly implemented, i.e. it would need less number of shifts to be realized indirectly by means of the combination of other shifts, what is translated into less number of operations. Concerning hardware configuration, for consistence, we choose the one exposed before (5+1 configuration). Nevertheless, we can choose more effective configurations to implement a large neighborhood template with six multipliers. Note that here we only use the five *A* template multipliers, we do not need the one of the *B* template.

Fig. 6 shows the original binary diffusion template and the cell configuration chosen for the emulation. Over the original template, sub-templates centers are emphasized by shadowing. Directions to be followed in the image shifting are shown. The first sub-template to be applied is the central one. Given that we have at our disposal several digital memories [12], we determine to realize the sub-templates application in two phases. Each phase starts from the original image. The first phase applies the template placed in the center of the original

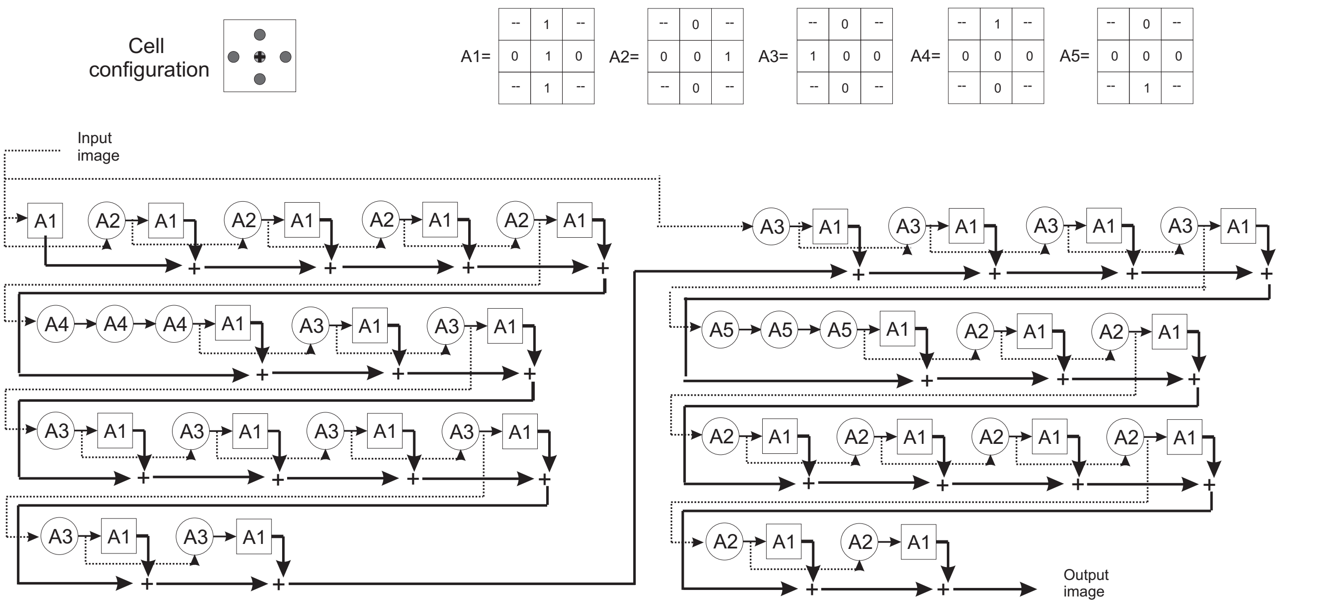


Fig. 7. System level implementation of a 9 × 9 diffusion operation. Cell configuration for a 5+1 reduction: *A* coefficients represented by dots, *B* coefficient represented by a cross. Templates required to emulate the original large neighborhood operation (see Fig. 6) are shown too.

template and continues to the up-left corner one. The second phase begins with the sub-template placed next to the central one, on its left, and continues along the arrows to the down-right corner. Fig. 7 shows all these operations at system level. Shifts are indicated by circles and sub-template application by squares.

V. Conclusion

A methodology to reduce the number of weighting multipliers in a DTCNN cell without penalty at application level has been addressed. Such a methodology was illustrated with the application of an active contour based technique, the PLS algorithm, onto a 1-bit binary programmable DTCNN architecture. The area processing-time trade-off sprung up with our methodology is the main concern at hardware level. The results given show that a significant hardware simplification (area improvement) comes with an affordable price at processing time in current sub-0.18 CMOS technologies for video processing with PLS. This proves the validity of the approach discussed here. Nevertheless, this still has to be confirmed with an on-chip implementation.

References

[1] L. O. Chua and L. Yang, "Cellular neural networks: Theory," *IEEE Transactions on Circuits and Systems*, vol. 35, no. 10, pp. 1257–1272, Oct. 1988.

[2] J. A. Hegt, D. M. W. Leenaerts, and R. T. Wilmans, "A novel compact arquitecture for a programable full-range CNN in 0.5 um CMOS technology," in *1998 Fifth International Workshop on Cellular Neural Networks and their Applications*, V. Tavsanoglu, Ed., London, England, Apr. 1998, pp. 288–293.

[3] V. M. Brea, D. L. Vilariño, A. Paasio, and D. Cabello, "On the one-quadrant template design in a high gain CNN model," in *Proceedings of the 8th International Workshop on Cellular Neural Networks and their Applications*, Budapest, Hungary, July 2004, pp. 279–284.

[4] A. Paasio, J. Flak, M. Laiho, and K. Halonen, "High density vlsi implementation of a bipolar CNN with reduced programmability," in *Proceedings of the 2004 IEEE International Symposium on Circuits and Systems. ISCAS '04*, vol. 3, Vancouver, Canada, May 2004, pp. 21–24.

[5] A. Paasio, A. Kananen, and V. Porra, "A 176x144 processor binary I/O CNN-UM chip design," in *Design Automation Day on Cellular Visual Microprocessor, European Conference on Circuit Theory and Design*, T. Roska, C. Beccari, M. Biey, P. Civalleri, and M. Gilli, Eds. Stressa, Italy: Levrotto&Bella, Torino, 1999, pp. 82–86.

[6] A. Paasio, M. Laiho, A. Kananen, and K. Halonen, "An analog array processor hardware realization with multiple new features," in *International Joint Conference on Neural networks*, Honolulu, Hawaii, 2002, pp. 1952–1955.

[7] F. Sargeni, V. Bonaiuto, and M. Bonifazi, "Time division digital programmable OTA for cellular neural networks," in *European Conference on Circuit Theory and Design, ECCTD 2005*, Cork, Ireland, 2005.

[8] P. Dudek, "Accuracy and efficiency of grey-level image filtering on vlsi cellular processor arrays," in *Proceedings of the IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA 2004*, Budapest, Hungary, July 2004, pp. 123–128.

[9] N. A. Fernández, D. L. Vilariño, V. M. Brea, and D. Cabello, "Large-neighborhood templates with nearest-neighbor connected patterns in binary-based cellular neural networks," in *XX Conference on Design of Circuits and Integrated Systems. DCIS'05*, Lisbon, Portugal, Nov. 2005.

[10] D. L. Vilariño and C. Rekeczky, "Implementation of a pixel-level snake algorithm on a CNNUM-based chip set architecture," *IEEE Transactions on Circuits and SystemsI: Regular Papers*, vol. 51, no. 5, pp. 885–891, May 2004.

[11] N. A. Fernández, V. M. Brea, D. L. Vilariño, and D. Cabello, "On the reduction of the number of coefficient circuits in a DTCNN cell," in *Proceeding of the 10th IEEE International Workshop on Cellular Neural Networks and their Applications. CNNA 2006*, Istanbul, Turkey, 2006, pp. 29–34.

[12] V. M. Brea, M. Laiho, D. L. Vilariño, A. Paasio, and D. Cabello, "A binary-based on-chip CNN solution for pixel-level snakes," *International Journal of Circuit Theory and Applications*, vol. 34, pp. 383–407, 2006.

[13] A. E. Gamal and H. Eltonkhy, "CMOS image sensors," *IEEE Circuits and Devices Magazine*, pp. 6–20, May/June 2005.

[14] V. M. Brea, D. L. Vilariño, A. Paasio, and D. Cabello, "Design of the processing core of a mixed-signal CMOS DTCNN chip for pixel–level snakes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, no. 5, pp. 997–1013, 2004.

[15] N. A. Fernández, D. L. Vilariño, V. M. Brea, and D. Cabello, "On the emulation of large-neighborhood templates with binary CNN-based architectures," in *Proceeding of the 9th IEEE International Workshop on Cellular Neural Networks and their Applications. CNNA 2005*, Hsinchu, Taiwan, May 2005, pp. 274–277.

ISCAS07:

N.A. Fernández-García, V.M. Brea, D. Cabello, "**Area and Time Efficient Cellular Non-linear Networks**," IEEE International Symposium on Circuits and Systems, 2007. ISCAS 2007. pp.2682-2685, New Orleans, USA, May 2007.

Area and Time Efficient Cellular Non-linear Networks

Natalia A. Fernández-García, Victor M. Brea and Diego Cabello
Departament of Electronics and Computer Science
University of Santiago de Compostela
E-15782 Santiago de Compostela, Galiza, Spain
Email: {natiafg, victor}@usc.es

Abstract— **The use of a reduced set of multipliers or coefficient circuits on cellular processor arrays leads to time and area efficient solutions. The reduced set of multipliers is achievable with the so-called Split&Shift (S&S) methodology. Data resultant from applying such a methodology to implementations with Cellular Non-linear Networks (CNN) reported in the literature are presented. Also, Pixel-Level Snakes (PLS) are used as benchmark for a more in-depth analysis of our methodology.**

I. Introduction

Area consumption is one of the main design goals in hardware design. This is even more important for massive parallel architectures with a large number of on-chip processing elements (cells) like SIMD approaches. CNN architectures are a particular case of the latter. They usually need two templates and a bias term to implement an operation, which leads to 19 multipliers (coefficient circuits) per processing element, and thus big area consumption.

There are two main lines to minimize area in CNN architectures through the coefficient circuits. The first line shrinks the area in the multipliers through circuit design techniques. The second one reduces the number of multipliers. In the first line it is remarkable the 1Q-1bit-B/W proposal, where the multipliers operate only within one quadrant, with only 1 bit of programmability in the templates and over binary images [1]. Another example of contribution within this line is the implementation of multipliers with only one transistor [2] used in ACE16k [3]. Note that the implementations reported in [1] and [3] also use a reduced number of multipliers.

Within the second line we have proposals of time multiplexing as [4] and [5] and proposals of modification of the funcionality as [6]. In the same way, we have introduced in [7] a methodology that leads to less connections and coefficient circuits per processing element in Discrete Time CNN (DTCNN) architectures without penalty at functional level. This methodology, namely Split&Shift (S&S) methodology, firstly splits the initial templates into new sub-templates, and secondly gathers the partial outcomes from the sub-templates in the appropriate cell by means of outcome or input image shifting. Our technique can be easily applied to every DTCNN or synchronous SIMD architecture, whether constrained to B/W images or not, and regardless the number of quadrants in the multipliers. The only constraints are synchronous processing and access to the cell state variable. However, this methodology might imply a significant increase in the number of operations and so in the processing time. Thus it might be troublesome for applications with hard time constraints.

Nevertheless, it is feasible to compensate for such an increase of operations by means of circuit design techniques combined with today sub-micron CMOS technologies. As an example, in video-rate processing applications there is a time slot of $40ms/fr$. If the acquisition and delivery times of the input and output images lie in the range of few ms, there are still tens of ms available for computing the image. This allows to fit tens of thousands of processing cycles assuming few μs per CNN operation or processing cycle, as is the case of the solutions reported in [3] and [8]. If it is possible to use faster architectures like those addressed in [1] and [9], with tens of nanoseconds per processing cycle, hundreds of thousands of image tasks per frame would be reachable. It is apparent that in many applications there would be many processing cycles unused. In such cases, as long as the time needs are met, the S&S methodology can be applied, leading to time and area efficient solutions. Furthermore, applications with hard time requirements with tens of thousands of frames per second, like those outlined in [10], might be feasible with S&S on either fast architectures or not, depending on the number of operations and the shape of the templates.

This paper contains two main contributions. On the one hand, the data collecting on existing CNN architectures reported in the literature ([1], [11]) show that the S&S methodology would give significant area gains. On the other hand, we use Pixel-Level Snakes (PLS), a well-known active contour-based technique in cellular processors [12], to show that our methodology is efficient in both time and area consumption for real-time applications with video-rate processing. The paper addresses both contributions in sections II and III. Finally, conclusions are gathered in section IV.

II. Issues in Applying the S&S Methodology

In order to apply the S&S methodology different issues have to be taken into account. We emphasize here that the S&S methodology can only be used with either synchronous architectures or CNNs with B-type templates only.

It is apparent that the starting architecture determines the data type that can be dealt with. In this sense, architectures that strictly realize the binary 1-bit 1Q CNN model like the one introduced in [1] would need an analog memory to accumulate partial outcomes from sub-templates [7]. Furthermore, even with an extra analog memory, these architectures are restricted to have image shifting, but not data-shifting, as the coefficient circuits are designed to work on binary variables and not on real-valued ones. On the other hand, synchronous architectures with cells of the type introduced in [3] can easily adopt the S&S methodology, as they count on analog memories to store sub-template outputs and can deal with any data type.

Another concern is the extra time caused by the extra number of operations from the new sub-templates. Based on rough and conservative estimates from previous work [7], one can conclude that the highest number of processing steps resultant from applying the S&S methodology to 2 full dense 3×3 templates with the barest of the configurations (3 coefficient circuits only) is less than 20. The use of output-shifting in S&S could lead to better figures of merit than image-shifting in some cases because of the possibility of overlapping sub-template application and partial output shifting in a 2-template configuration. The time for a processing step depends on the hardware solution. In solutions like [3] and [8], running B-type templates lasts few μs. This time is easy to cut down with today digital CMOS technologies. In fact, in current sub-micron technologies processing steps of less than $100ns$ are easily achievable with 1-bit programmable architectures [1] [9]. These times include the uploading of the templates or instructions from a global memory to the cell array. Keeping all these numbers in mind, and accounting for the image acquisition and the output data downloading times, the designer can judge whether or not the S&S methodology still complies with the time requirements of the application.

Another issue when applying the S&S methodology is how much area is saved. This also depends on the particular hardware solution. In order to give some numbers we go through two different architectures. The first one is the 1-bit programmable approach addressed in [1]. In this case, the cell contains 9 coefficient circuits, occupying an approximate area of $32\mu m^2$ within the $155\mu m^2$ of the total cell area. The S&S methodology would lead to 3 multipliers. In area, this means to save $21\mu m^2$, which is around 14% of the total cell area. In a 128×128 array this would be around $0.35mm^2$. The second architecture is discussed in [11]. Every cell counts on 8 multipliers for neighborhood connectivity, plus the feedback term and three additional multipliers. Due to their much higher accuracy, the latter four multipliers are much larger than the ones used for connectivity purposes. We apply the S&S methodology to the set of 8 multipliers for connectivity, diminishing this number down to 3. This leads to area savings of 6.3% per cell, which is around $351\mu m^2$. In a 128×128 array this amounts to $5.8mm^2$. It should be noted that in the first architecture, the gains in area are from moderate to marginal. In the second architecture, the area savings are significant. In both cases we assume that the inter-cell routing is included. As it was mentioned above, the first architecture would need an analog memory to run S&S. The second architecture would not need anything else. Also, in the second case we have applied our methodology to a reduced number of multipliers, not to the whole set. In this sense, the area calculations are conservative. Much better optimizations in area would be expected if the cell presented in [3] and the S&S methodology were combined in a more exhaustive way.

As a final remark, area and processing time come up as a trade-off. The lesser the number of multipliers the smaller the area, and as a consequence more processing steps. Clearly, the feasibility of this approach would be determined by the time needs of the application. It should also be noted that the area occupied by the multipliers is strongly determined by accuracy requirements. The accuracy also influences the power dissipation. As the new subtemplates contain less coefficient terms, it is also expected to loose the accuracy requirements on the coefficient circuits [13]. Furthermore, this means less power dissipation [14]. The last two items will be studied in the short-term future.

III. Benchmarking: real time and high speed applications

To study the area and time efficiency of the S&S we use the PLS algorithm addressed in [[12] under its implementation in [9] as benchmark. Nevertheless, as a difference from [12], we also include the external potential extraction in our estimates. This is obtained as a set of B/W operations.

PLS is an active contour algorithm that deals with contours at pixel level. It has been introduced as a very useful technique for image segmentation in real-time applications thanks to its suitability for CNN implementation. According to the processing data, PLS contains a module for gray-scale and another one for B/W tasks. The gray-scale processing extracts the guiding information for the contours from the original input image. These operations are realized over a specific hardware. Processing the contours only involves B/W CNN operations. This comprises morphological operations like erosion and dilation, logical functions (AND, OR), propagative templates like hole filling, large neighborhood operators like diffusion, and some other specific hit-and-miss operations. B/W operations are realized over a 1Q-1bit-B/W CNN architecture with two templates, one of them with 9 possible non-null coefficients, and the other one with the central coefficient as the only non-zero entry. Thus, the initial number of multipliers is 10.

In this section we analyze the area-processing time trade-off when applying the S&S methodology to the B/W module of the PLS. The great variety of operations along with their long time-consuming (propagative and large-neighborhood templates included) nature make PLS an appropriate real-time benchmark for our methodology.

Table I collects the analysis of PLS for 6 different configurations of coefficient circuits. We have chosen the most time efficient configurations among those with the same number of coefficient circuits (c.c.). Templates with one and two coefficients cannot approach a general 3×3 template. For three c.c. we present two different configurations, one with one template and another with two. With this we try to illustrate the convenience of using two templates. The reason is that two

TABLE I

AREA AND TIME ANALYSIS OF PLS WITH S&S

Number of C.C. (Config.)	*Ops/fr* 3×3 9×9	*HR -%-* (μm^2)	*OIF*	*RPO -%-*	*ms/fr (fr/s)*	*Propag. Tasks Ops/fr - %*
10 Coeffs. ● ● ● ‖ ○ ○ ○ ● ● ● ‖ ○ ● ○ ● ● ● ‖ ○ ○ ○	11065 12185	0 (0)	1 1	0 0	1.21 (826) 1.32 (758)	10240 92% 84%
6 Coeffs. ○ ● ○ ‖ ○ ○ ○ ● ● ● ‖ ○ ● ○ ○ ● ○ ‖ ○ ○ ○	11309 13389	40 (35.6)	1.022 1.099	1813.9 404.8	1.23 (813) 1.44 (694)	10240 90% 77%
5 Coeffs. ○ ● ○ ‖ ○ ○ ○ ● ○ ● ‖ ○ ● ○ ○ ● ○ ‖ ○ ○ ○	16749 18829	50 (45.5)	1.513 1.545	97.3 91.7	1.77 (565) 1.98 (505)	15360 92% 82%
4 Coeffs. ○ ● ○ ‖ ○ ○ ○ ○ ○ ● ‖ ● ○ ○ ○ ● ○ ‖ ○ ○ ○	22675 24755	60 (53.3)	2.049 2.031	57.2 58.2	2.37 (422) 2.58 (388)	20480 90% 83%
3 Coeffs. ○ ● ○ ‖ ○ ○ ○ ● ○ ○ ‖ ○ ○ ○ ○ ○ ● ‖ ○ ○ ○	49720 52360	70 (62.2)	4.493 4.297	20.0 21.2	5.07 (197) 5.34 (187)	46080 94% 88%
3 Coeffs. ○ ● ○ ‖ ○ ○ ○ ● ○ ○ ‖ ○ ○ ○ ○ ○ ○ ‖ ○ ○ ●	44326 47006	70 (62.2)	4.006 3.858	23.3 24.5	4.53 (221) 4.80 (208)	40960 92% 87%

templates are advantageous to execute pixel-to-pixel logical functions on two different images.

The second column in Table I lists the number of operations needed to implement PLS with each configuration of coefficient circuits. In the following evaluation we account for both all the B/W tasks and the initial gray-scale operations. The latter is accounted in equivalent B/W CNN operations. Ten iterations with four cardinal directions each were assumed for PLS execution. This number is high enough for applications like surveillance [9]. The total number of operations for B/W processing is calculated under the consideration of worst case for the hole-filling in a 128×128 image. This task is carried out twice in PLS [12]. The number of operations (processing steps) per frame also varies with the size of the diffusion operator. Table I gives numbers for two different orders of neighborhood, namely 3×3 and 9×9. The number of operations grows slowly with the order of neighborhood. Also, and in line with what it was commented above, the configuration with 3 coefficient circuits in two templates performs better than the one with all the multipliers in only one template.

Hardware reduction (HR) is given in both percentage of multipliers and absolute area saved per cell. The latter is obtained from [9]. Area savings up to $1.02mm^2$ in a 128×128 image are achievable. Inter-cell routing and room for the analog memory are not accounted in this estimate.

Table I also outlines the operations increment factor (OIF). It is apparent that the smaller the number of coefficient circuits, the higher the OIF. RPO or "percentage of hardware Reduction Per CNN Operation increased for each original CNN operation" formulated as Eq. (1) accounts for the area-time (HR-OIF) trade-off.

$$RPO(\%) = \frac{HR(\%)}{OIF - 1} \qquad (1)$$

In the evaluation of time performance, we extract the time per frame for every configuration of coefficient circuits. For this, we consider $100ns$ for every processing step, and $0.1ms$ for downloading and uploading purposes. Note that these times are given in order to estimate how many processing steps we can have to still meet the time needs of the application (more than 390 000 for 25 fr/s in this case, which means a maximun OIF greater than 30). In this sense, it should be said that the acquisition time is variable and depends on the sensor implementation, the application and the scene. The time for downloading of the output image is short, as the final output is a B/W image (the contours). The numbers presented in Table I are easily redefined. For instance, if the combination of uploading and downloading times is 1 ms instead of 0.1 ms, the number 1.2 ms/fr would become 2.1 ms/fr.

From Table I we can conclude that video frame rates are achievable even for the barest configurations. This is also true assuming processing steps of μs, as architecture in [3], up to 4 coefficient circuits. Note that most of the time is consumed in executing propagative tasks, in this case hole-filling. This suggests that applications without propagative operations or with sparser templates on them could reach higher frame rates. Nevertheless, solutions like the one proposed in [15] or the use of a kind of cache memory to avoid repetitive template memory accesses might be very convenient approaches in applications with propagative tasks and hard-time constraints. Finally, higher frame rates can be possible with specific hardware like local logic units, or circuits for shifting.

Also, we want to remark that the most adequate allocation of the coefficient circuits will be given by the templates shape. In this case we have chosen the most symmetrical configurations because we have a significant number of isotropic templates. Note that in this case the use of 6 coefficient circuits instead of 10 barely penalizes the time performance. This means that most of the PLS operations are doable with only 6 coefficient circuits in such a configuration. And so, there is a big area saving with a very small penalty in time, therefore RPO is quite high. Moreover, we can see the importance of having the coefficient circuits split in two templates (see the case of 3 coefficient circuits). Also, and although not shown in Table I, the largest OIF was 8. This case corresponds to configurations with 3 coefficient circuits in two templates when approaching the 3×3 diffusion operator. It is interesting to say that this factor drops to 2.6 if we choose a 9×9 diffusion template. The reason is a more efficient use of the multipliers combined with large neighborhood techniques [16]. In this case the maximun OIF is 5 and corresponds with AND/OR operations.

In the CNN literature there are many applications with moderate to low time needs (tens to hundreds of frames per second) comprising lots of B/W operations of the type of those found in PLS [17]–[19]. In all these cases, the S&S methodology might lead to area and time efficient solutions. In the end, the application determines whether or not our approach is advantageous. On the other hand, applications with time goals in the order of tens of thousands of frames per second like those addressed in [10] might also be solved with S&S. If the acquisition time is around $1\mu s$, there is still a slot of tens of μs to process and deliver the output image. The implementation of a reduced set of multipliers combined with a fast architecture leading to short processing steps (tens of nanoseconds) might be good enough to keep pace with the requirements of tens of thousands of frames per second.

IV. Conclusion

This paper has shown that the use of a reduced number of coefficient circuits in DTCNNs is a time and area efficient solution. This conclusion is mainly drawn from data on CNN hardware implementations reported in the literature. The reduced set of multipliers is realizable with the so-called S&S methodology. In this paper we have also tested S&S in the PLS algorithm, which is an adequate benchmark because of its great variety of B/W operations. The main conclusion in the PLS evaluation is that even with only 3 coefficient circuits, the barest configurations for our methodogy, the PLS execution would be fast enough to reach hundreds of frames per second. Such an analysis is easy to extend to other type of algorithms and implementations by simply considering the appropriate characteristic data and time restrictions. Moreover, in applications with hard time requirements, with tens of thousands of frames per second, the S&S methodology might also be applicable.

Acknowledgment

This work is partly funded by Xunta de Galicia under the project PGIDIT06TIC10502PR and the MEC-FPU program.

References

[1] J. Flak et al., "Dense CMOS implementation of a binary-programmable CNN," *Int. J. Circuit Theory Appl.*, 34, 429–443, 2006.

[2] R. Dominguez-Castro et al., "Four-quadrant one-transistor-synapse for high-density CNN implementations," in *CNNA'98*, London, U.K., 1998, 243–248.

[3] A. Rodríguez-Vázquez et al., "ACE16K: The third generation of mixed-signal SIMD-CNN ACE chips towards VSoCs," *IEEE Trans. Circuits Syst. I-Regul. Pap.*, 51, 5, 851–863, 2004.

[4] K.K. Lai et al., "Implementation of Time-Multiplexed CNN Building Block Cell" *MicroNeuro'96*, 80–85, 1996.

[5] F. Sargeni et al., "Time Division Digital Programmable OTA for Cellular Neural Networks," in *ECCTD'05*, Cork, Ireland, 2005.

[6] A. Paasio et al., "A 176x144 Processor Binary I/O CNN-UM Chip Design," *Design Automation Day on Cellular Visual Microprocessor, ECCTD'99*, 82–86, 1999.

[7] N.A. Fernández et al., "On the reduction of the number of coefficient circuits in a DTCNN cell," in *CNNA'06*, Istanbul, Turkey, 2006, 29–34.

[8] P. Dudek, "Implementation of SIMD vision chip with 128×128 array of analogue processing elements," in *ISCAS'05*, 6, Kobe, Japan, 2005, 5806–5809.

[9] V.M. Brea et al., "A binary-based on-chip CNN solution for pixel-level snakes," *Int. J. Circuit Theory Appl.*, 34, 383–407, 2006.

[10] A. Zarándy et al., "Bi-i: A standalone ultra high speed cellular vision system," *IEEE Circuits and Systems Magazine*, 5, 2, 36–45, 2005.

[11] G. Liñán, "Design of mixed-signal programmable chips with low power dissipation for real-time vision," Ph.D. dissertation, University of Sevilla, Sevilla, Spain, 2002.

[12] D.L. Vilariño et al., "Implementation of a pixel-level snake algorithm on a CNNUM-based chip set architecture," *IEEE Trans. Circuits Syst. I-Regul. Pap.*, 51, 5, 885–891, 2004.

[13] A. Paasio et al., "CNN template robustness with different output nonlinearities," *Int. J. Circuit Theory Appl.*, 27, 87–102, 1999.

[14] P.R. Kinget, "Device mismatch and tradeoffs in the design of analog circuits," *IEEE J. Solid-State Circuit*, 40, 6, 1212–1224, 2005.

[15] P. Dudek, "Fast and efficient implementation of trigger-wave propagation on VLSI cellular processor arrays," in *CNNA'04*, Budapest, Hungary, 2004, 117–122.

[16] N.A. Fernández et al., "Hardware simplification in cellular non-linear networks for complex algorithms," in *DCIS'06*, Barcelona, Spain, 2006.

[17] S. Xavier de Souza et al., "Fast and robust face tracking for CNN chips: application to wheelchair driving," in *CNNA'06*, Istanbul, Turkey, 2006, 200–205.

[18] Z. Szlavik et al., "Visual inspection of metal objects by using cellular neural networks," in *CNNA'06*, Istanbul, Turkey, 2006, 200–205.

[19] S. Malki et al., "Vein feature extraction using DT-CNNs," in *CNNA'06*, Istanbul, Turkey, 2006, 307–312.

ECCTD07-1:

N.A. Fernández-García, J. Albó-Canals, V.M. Brea, J. Riera-Baburés, D. Cabello, X. Vilasís-Cardona. "**Verification of Split&Shift techniques for CNN hardware reduction**," 18th European Conference on Circuit Theory and Design, 2007. ECCTD 2007. pp.88-91, Seville, Spain, August 2007.

Verification of Split&Shift Techniques for CNN Hardware Reduction

N.A. Fernández-García*, Jordi Albó-Canals§, Victor M. Brea*, Jordi Riera-Baburés§,
Diego Cabello* and Xavier Vilasís-Cardona§
*Departamento de Electrónica e Computación, Universidade de Santiago de Compostela
E-15782 Santiago de Compostela, Galiza, Spain. E-mail:natiafg@usc.es
§Departament d'Electrónica, Enginyeria i Arquitectura La Salle, Universitat Ramon Llull
E-08022 Barcelona, Spain. E-mail:jalbo@salle.url.edu

Abstract— **The so-called Split&Shift (S&S) methodology has previously been introduced as an effective area saving technique for hardware implementation of Cellular Non-linear Networks. This work provides the first experimental proof of such a methodology through a circuit implementation over an FPGA platform. Results of area, processing time and functionality of different instances of the S&S methodology are given.**

I. Introduction

Cellular Non-linear Networks (CNNs) have traditionally been posed as an example of practical solution to the on-chip realization of SIMD architectures for image computation with direct pixel to cell assignment [1]. The local connectivity is one of the key factors in having CNN on-chip implementations with high resolution. Nevertheless, low area consumption per cell continues being a design goal to achieve either more cells per chip or more functionality. In line with this, in [2] we have introduced the Split&Shift (S&S) methodology. This leads to less connections and coefficient circuits per cell in synchronous CNN architectures, thereby less area consumption without penalty at functional level. The work in [3] reports estimates of area savings and processing speed of on-chip CNN implementations found in the literature. The conclusion drawn from that analysis was that the S&S methodology yields benefits in area while keeping a fast enough processing speed for video rate processing applications and beyond.

This work provides the first circuit verification of the S&S methodology. The circuits have been designed with VHDL and synthesized on an FPGA platform. Although this approach has some limitations with regards to the analog CNN on-chip solution [4], as photosensors and processing array on different chips, the design cycle is much shorter, providing a fast way of analyzing the S&S methodology at hardware level.

II. S&S Techniques

As mentioned above, the S&S methodology permits to drop the number of coefficient circuits (cc) in a cell, and still to approach dense 3×3 or even larger-neighborhood templates. The methodology works in two steps. The first step splits the original template into templates with a reduced set of coefficients (sub-templates). In the second step, the outcomes of every sub-template are properly combined in order to preserve the original template output [2].

Fig. 1 shows how the $S\&S$ technique works for a case with four coefficient circuits. The original template is split into three sub-templates, namely, $A1$, $A2$ and $A3$. There is also an extra template ($A4$) required to do shifting. The sequence of operations is displayed at the bottom of Fig. 1. In case of two templates, A and A', they are run sequentially in time. First, template $A1$ ($A1'$) is applied. Next, the original image is shifted one pixel to the left ($A4$), and over that image, the template $A2$ ($A2'$) is run. A similar process is done with $A3$ ($A3'$). The final step adds the bias term to the contribution of both templates and provides the output function Y.

This example shows the case of a generic CNN operation with dense 3×3 templates. Nevertheless, usually the templates comprise some null coefficients. Furthermore, in application oriented implementations, the coefficient circuit configuration can be suited to the most repetitive template pattern, yielding better figures of merit in area and time [3].

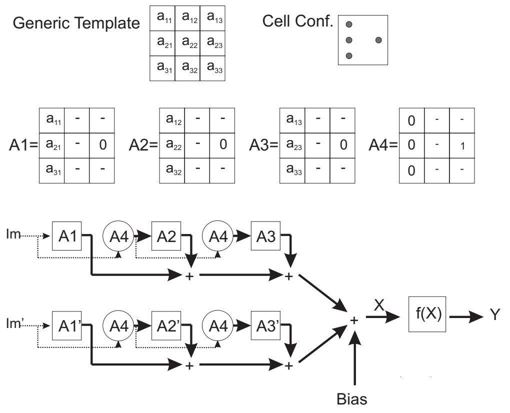

Generic Template
a_{11} a_{12} a_{13} a_{21} a_{22} a_{23} a_{31} a_{32} a_{33}
Cell Conf.
A1=
A2=
A3=
A4=
Im
Im'
A1
A2
A3
A4
A1'
A2'
A3'
X
f(X)
Y
Bias

Fig. 1. Configuration of 4 coefficient circuits. Emulation of a generic CNN operation with dense templates over this configuration.

III. FPGA implementation

A. Design Issues

S&S techniques are applicable to synchronous CNN, whether constrained to B/W images or not, and regardless the number of quadrants used in the coeffient circuits [2]. The only constraints are synchronous processing and access to the cell state variable. In this work we have chosen binary image processing and we use the so-called positive range high-gain non-linear CNN model with 1-bit of programmability [5]. In this way, the array size on the FPGA can be large enough as to clearly study the effects of the methodology. Also, as many algorithms and applications make an intensive use of B/W images the focus on this type of algorithms takes certain relevance. In fact, there are even several proposals to optimize B/W processing separately from the gray-level one as [6] or [7]. Furthermore, this case is the least favorable in terms of area savings with the application of the techniques because of the reduced size of the coefficient circuits. It is expected to have larger area savings with gray-scale cells [3].

The implementation shown here comprises the minimum possible hardware to run our methodology. The emphasis is put on the performance of the S&S techniques. The discussion on the cell particular realization is beyond the scope of this paper. Having these and the above considerations into account, our cells consist of five primary blocks that can be seen in Fig. 2. The register (D-Flip Flop) acts as local logic memory (LLM). Coefficient circuits are simplified to logic AND gates because of the binary nature of the processing. An encoder plays the role of the adder in classical on-chip CNN implementations transfering the contributions from direct connected neighbors into a binary word that represents the number of contributions equal to '1' and that makes a partial outcome. In order to apply the methodology we need an accumulator that gathers the partial outcomes obtained with the application of the different sub-templates and the bias term. For the bias term, the convention used in [7] for the positive-range high-gain non-linear CNN model with 1-bit of programmability, where the template coefficients are set to 1 and the bias is set to $X.5$, being X an integer number between 0 and 3, is now suited to the digital nature of the FPGA. In our implementation the bias term is a 2-bit positive integer number with the correspondences shown in Table I. This means to shift the threshold for the output-state relationship from 0 to 3, being the new output function as indicated in Eq. (1). As a result, the threshold function can be implemented with an OR gate that takes as inputs the bits in the output of the accumulator after the bias is added, discarding the two least significant ones, as they make 3, that is the threshold value.

$$y = f(x) = \begin{cases} 0 & x \leq 3 \\ 1 & x > 3 \end{cases} \tag{1}$$

It is worth noting that the dummy cells value is updated with the shiftings to allow the correct operation of the methodology. As a consequence, they are not simple memories but are implemented with a properly sized encoder and a number of registers and ANDs that depend on the configuration selected.

TABLE I
CORRESPONDENCE BETWEEN BIAS VALUE IN [7] AND OUR IMPLEMENTATION.

2-bit Bias [7]	-0.5	-1.5	-2.5	-3.5
New Bias Values	3	2	1	0

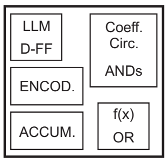


Fig. 2. Generic blocks of the CNN cell on the FPGA.

In order to check the functionality of our approach we just show an effective grid of 9×9 cells/pixels, 11×11 with dummy cells, under the four coefficient circuits configuration depicted in Fig. 1. We have chosen to run recursive dilations on this grid. The evolution of three iterations is displayed with numbers on Fig. 3. The initial image is the central gray figure with '0' number on it. Numbers '1', '2' and '3' indicates the pixels turned black after the correponding iteration. The dilation template used was Eq. (2) and it was applied three times through 9 sub-templates and 6 shifts.

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad I = 3 \tag{2}$$

With this simple example we show that the implementation works correctly by applying the S&S techniques at each template as well as several templates can be run sequentially. In spite of having used the same template in all the iterations, different templates can be applied. To apply a general algorithm we would need some more memories to keep intermediate results that have to be used after and larger neighborhood templates could be applied only with a large enough accumulator.

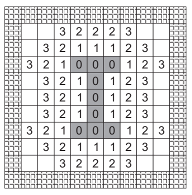


Fig. 3. Result of applying the dilation template over our implementation with the scheme of Fig. 1. Three iterations displayed in a gray-level code.

B. Experimental instances

In order to analyze the results for different number of coefficient circuits (cc) we have implemented 32×32 grids (30×30 effective cells) with 9, 6, 5, 4 and 3 cc per cell. The configurations selected are listed in Table II. We have selected the most adequate configurations for a generic and full-dense 3×3 template [2], although large neighborhood templates or sparse ones could be applied with the only restriction of the size of accumulator. Even so, we have considered two different 4-cc configurations (4 C.C. (a) and 4 C.C.(b) in Table II) to evaluate how the symmetry in the allocation of coefficient circuits affects performance. Also, we have included two implementations with central coefficients (6 C.C. (b) and 5 C.C.

TABLE II

OCCUPATION AND PROCESSING DATA FOR DIFFERENT NUMBER OF COEFFICIENT CIRCUITS IN A 32×32 GRID

32×32 (30×30)	**9 C.C.** ● ● ● ● ● ● ● ● ●	**6 C.C.(a)** ● ○ ● ● ○ ● ● ○ ●	**6 C.C.(b)** ● ○ ● ● ● ○ ● ○ ●	**5 C.C.** ○ ● ○ ● ● ● ○ ● ○	**4 C.C.(a)** ● ○ ○ ● ○ ● ● ○ ○	**4 C.C.(b)** ○ ● ○ ● ○ ● ○ ● ○	**3 C.C.** ○ ● ○ ● ○ ○ ○ ○ ●
Total Area LEs - (%)	26954* (103%)*	25293 (99%)	25297 (99%)	22512 (87%)	17941 (70%)	18033 (70%)	14189 (55%)
Area Per Cell - LEs	27*	25	25	22	17	17	13
Control Area - LEs (Mem+Grid+Cells)	109* (92+17+0)*	144 (110+17+17)	144 (110+17+17)	144 (110+17+17)	144 (110+17+17)	144 (110+17+17)	144 (110+17+17)
Dummy Cells Area - LEs	124*	673	677	647	521	613	369
Frequency - MHz	-	47.01	47.82	47.46	48.75	47.00	46.59
CNN Steps Per Template **(Clock Cycles)**	1 (2)	3 (9)	3 (9)	5 (15)	5 (15)	5 (15)	10 (30)

* Data given by the synthesizer might be under-estimated because 32×32 grid with 9 cc per cell could not be implemented properly.

in Table II). This would favor pixel-wise Boolean functions on two images. Finally, we have implemented grids of different sizes with 9 and 4 cc, the last one in its configuration (a), to study the behavior of the implementation with the number of cells implemented and to check if the conclusions are valid for any grid size (Table III).

The classical implementation of a DTCNN architecture with only one template (A or B) comprises 9 cc. In this case the template is applied in one step and as the S&S methodology is not used, the accumulator (Fig. 2) is not necessary, the bias is added through an adder and the dummy cells come down to memories. Nevertheless, the lack of accumulator shrinks the functionality of the 9 cc configuration, as it is not possible to approach neither larger neighborhood templates, nor CNN operations with two templates. This, however, is possible with the configurations of less coefficient circuits outlined in Table II, as these comprise the accumulator required for the S&S techniques. In this sense, having the same functionality would widen the area gap between the 9 cc configuration and others with less coefficient circuits.

C. Experimental data and discussion

Our implementation has been realized with the Active-HDL 7.1 software by Aldec and synthesized with the Quartus II 6.0 by Altera over the Altera Stratix-EP1S25 FPGA. This FPGA provides a total of 25660 logic elements.

Table II shows the results from a grid with 30×30 effective cells for different number of cc. We can clearly see that a reduced number of cc would lead to more cells on the FPGA. In fact, we cannot implement a 30×30 effective grid with 9 cc per cell on the selected FPGA (103% occupation), but we have almost half of the FPGA empty for the 3 cc configuration listed in Table II. The reason, apart from less coefficient circuits, is the smaller size of the encoder that is optimized for each number of cc and the number of registers that collect the direct neighbor contributions.

As we can see in Table II, the control circuitry for S&S techniques (Cells Control) consumes 17 logic elements when such techniques are applied. The control circuitry for the grid remains the same regardless the number of coefficient circuits. The control circuitry for the uploading of the image from outside to the inner RAM on the FPGA barely changes with the configuration used. The area consumed by the dummy cells depends on their connectivity with both the inner cells and other surrounding dummy cells, i.e., on the number of coefficient circuits and their allocation. For the 9-cc configuration, every dummy cell amounts to only one LE. Aside from this hardware, and for all the configurations listed in Table II, the synthesizer books 1976 LEs for communications from outside to the inner RAM memories and from there to the grid. This number only changes with the array size.

Concerning to have or not to have the central coefficient circuit, we sort out two cases (Table II): the same number of cc but different configuration (6 C.C. (a) and 6 C.C. (b)), and different number of cc but only because of the central coefficient (5 C.C. and 4 C.C.(b)). It is worth pointing out that the benefit of having a central coefficient is the reduction in the number of steps for implementing pixel-wise Boolean functions on two images. In the first case, leaving aside the area consumed by the dummy cells, the coefficient circuit allocation barely affects the data shown in Table II. From this, we conclude that in this implementation the influence of the intercell connections is much less significant than the size of the encoder and the number of registers in the cell. We can reach the same conclusion for 5 C.C. and 4 C.C.(b). In this case the intercell connectivity is the same and the only difference is the central auto-feedback coefficient and the difference in the area occupation is still notable. Finally, regarding symmetry, we only find differences in the area occupied by the dummy cells with better results in the less symmetric configuration (4 C.C. (a) vs. 4 C.C. (b)). This difference is due to the reduced number of connections on one of the sides of the cell and not to the absence of symmetry.

In reference to processing time, the working frequency is almost the same for all cases and so, the difference will come from the different number of cycles needed per CNN template application. Leaving aside uploading and downloading times for the image, we estimate processing speeds assuming 50 MHz as working frequency. As the current study is conceived for binary image processing, uploading

TABLE III
OCCUPATION AND CLOCK DATA FOR DIFFERENT GRID SIZES WITH 9 AND 4 C.C.

	9 Coefficient Circuits					**4 Coefficient Circuits**				
	Area				**Speed**	**Area**				**Speed**
Grid Size	**Total** (LEs) (%)	**Per cell** (LEs)	**Control** (Mem+Grid+Cells) (LEs)	**Dummy Cells** (LEs)	**Max. Freq.** (MHz)	**Total** (LEs) (%)	**Per cell** (LEs)	**Control** (Mem+Grid+Cells) (LEs)	**Dummy Cells** (LEs)	**Max. Freq.** (MHz)
3×3 (1 effect. cell)	197 ($< 1\%$)	27	109 $(92 + 17 + 0)$	8	67.28	235 ($< 1\%$)	26	127 $(92 + 17 + 18)$	29	67.99
4×4 (2×2 effect.)	296 (1%)	27	110 $(93 + 17 + 0)$	12	81.30	314 (1%)	18.50	128 $(93 + 17 + 18)$	46	70.21
5×5 (3×3 effect.)	466 (2%)	27	112 $(95 + 17 + 0)$	16	68.04	446 (2%)	17	129 $(95 + 17 + 17)$	63	65.52
11×11 (9×9 effect.)	2649 (10%)	27	117 $(100 + 17 + 0)$	40	58.88	1988 (8%)	17	134 $(100 + 17 + 17)$	165	58.24
32×32 (30×30 effect.)	26954* (103%)*	27*	109* $(92 + 17 + 0)$*	124*	-	17941 (70%)	17	144 $(110 + 17 + 17)$	521	48.75

* Data given by the synthesizer might be under-estimated because 32×32 grid with 9 cc per cell could not be implemented properly.

and downloading times are expected to be short, especially if pipelining strategies and FPGAs with a large pint count and fast buses are combined. For the time being, and since the emphasis has been put on the methodology, and not on a particular application, these times have not been evaluated yet. Bearing this in mind, for a working frequency of 50 MHz, i.e. a clock cycle of 20 ns, we could have $5 \cdot 10^5$ complete CNN one-template-operations per frame (ops/fr) with 25 fr/s for 9 cc, but still 62500 ops/fr for 3 cc. This includes bias addition and local memory storage, being two more clock cycles. In other words, considering an algorithm of 100 ops/fr we could reach $125 \cdot 10^3$ fr/s with 9 cc and still 15625 fr/s with 3 cc. And finally, considering a complex algorithm with 12000 ops/fr we can have 1041 fr/s for 9 cc and 130 fr/s for 3 cc. With these figures and a fast camera, to achieve video frame rate should not be a challenge, even accounting for the uploading and downloading times of the image. Note that although these numbers refers to one-3×3 template-CNN operations, we could use two-template-CNN or even large-neighbourhood templates by only adjusting properly the size of the accumulator.

We have also studied implementations with different grid sizes. The results are conveyed in Table III. By analyzing the total FPGA occupation we can see that the benefits of the reduced set of coefficient circuits appear for grids of 5×5 cells or bigger. The reason is that when using the S&S methodology we need more control hardware and dummy-cells are more complicated and this extra hardware is not compensated with the reduced cells size (on average at Table III) for such small grids. Finally, and concerning memory control and frequency, as expected, we have worse results as the grid size grows. In the case of the frequency, this is because of the larger fan-out on the global lines that must deliver the template values and control signals to the array. In the case of control, this is because its complexity grows with the size of the image that must be uploaded from outside to the inner RAM memory and from there to the grid.

IV. CONCLUSIONS

This paper presents a circuit verification of the so-called S&S methodology, previously introduced in the CNN literature. Such a verification is limited to binary image processing and validated with the implementation of a positive range high-gain non-linear CNN model on an FPGA architecture, in particular on the Stratix-EP1S25 FPGA by Altera. The use of the S&S methodology leads to area savings intended for bigger CNN arrays and/or cells with more functionality. This conclusion is not only restricted to reconfigurable VLSI architectures like FPGAs, but it is also extended to other solutions like full-custom designs. As a general conclusion, we can state that under certain constraints, and possibly for particular applications that require images of moderate to low resolution, topographic CNN architectures on FPGAs might well benefit from our methodology, especially with the advent of FPGAs that increase the number of elements.

ACKNOWLEDGMENT

This work is partly funded by Xunta de Galicia under the project PGIDIT06TIC10502PR and the MEC-FPU program.

REFERENCES

[1] L.O. Chua et al., "The CNN Paradigm," *IEEE Transactions on Circuits and Systems*, vol. 40, pp. 147-156, 1993.
[2] N. A. Fernández et al., "On the reduction of the number of coefficient circuits in a DTCNN cell," in *CNNA 2006*, Istanbul, Turkey, 2006, pp. 29–34.
[3] N. A. Fernández et al., "Area and Time Efficient Cellular Non-linear Networks," in *ISCAS 2007*, New Orleans, USA, 2007, pp. 2682–2685.
[4] A. Rodríguez-Vázquez et al., "ACE16K: The third generation of mixed-signal SIMD-CNN ACE chips towards VSoCs," *IEEE Trans. Circuits Syst. I-Regul. Pap.*, vol. 51, no. 5, pp. 851–863, 2004.
[5] J. Flak et al., "Dense CMOS implementation os a binary-programmable CNN," *Int. J. Circuit Theory Appl. s*, vol. 34, pp. 429–443, 2006.
[6] A. Paasio et al., "Different Approaches for CNN VLSI Implementations," *ECCTD'99*, vol. II, pp. 1347–1350.
[7] V. M. Brea et al., "A binary-based on-chip CNN solution for pixel-level snakes," *Int. J. Circuit Theory Appl.*, vol. 34, pp. 383–407, 2006.

CNNA08:

N. A. Fernández, M. Suárez, V. M. Brea and D. Cabello. " **Template-Oriented Hardware Design based on Shape Analysis of 2D CNN Operators in CNN Template Libraries and Applications**", in *Proceedings of the 2008* $11^t h$ *International Workshop on Cellular Neural Networks and their Applications, CNNA 2008*, Santiago de Compostela, Spain, July 2008.

11th International Workshop on Cellular Neural Networks and their Applications
Santiago de Compostela, Spain, 14-16 July 2008 CNNA2008

Template-Oriented Hardware Design based on Shape Analysis of 2D CNN Operators in CNN Template Libraries and Applications

Natalia A. Fernández García, M. Suárez, V.M. Brea and D. Cabello
Departament of Electronics and Computer Science
University of Santiago de Compostela
E-15782 Santiago de Compostela, Galiza, Spain
Email: {natiafg, victor}@usc.es

Abstract— **This work aims at finding efficient configurations of coefficient circuits in a CNN hardware cell implementation based on the shape of the templates listed in the Cellular Wave Computing Library (CSW) and some applications/algorithms addressed in the CNN literature. The paper also touches briefly on possible hardware approaches to take advantage of the shape and symmetries of the templates.**

I. Introduction

Large integration density on massively parallel ICs for early vision continues being a necessity, either to tackle images of large resolution or to include more functionality, or simply to have particular blocks able to carry out specific functions. Despite the today's impressive advancements of semiconductor technology [1], such a need of more integration density leads engineers to come up with ingenious solutions to shrink area in their processing elements.

An efficient approach to save area is to look simultaneously at both hardware and applications, applying co-design. In this line, obviously the most straightforward way to save area in the processing elements of cellular processor arrays, either CNN or SIMD, is to design application-oriented hardware. This is the case of the solutions addressed in [2], [3], [4], where digital function blocks have been designed for specific applications. In the CNN realm we find examples of implementations that use efficient area-saving models capable of carrying out general-purpose processing. This is the case of the implementation discussed in [5]. The CNN model employed therein looks at hardware and image processing tasks with the inclusion of 1-bit programmable templates [6] and the change from two templates with 9 coefficients each to only 10 coefficients distributed into one template of 9 coefficients and other one of only one coefficient. This leads to great improvements in area occupation and processing speed.

One step further in the search for low area CNN cells is to approach a general template with a reduced number of coefficient circuits by splitting one operation into several ones. This can be done by the so-called Split&Shift techniques [7]. The Split&Shift methodology can be easily applied to synchronous cellular processor arrays and to either BW or gray-scale image processing without penalty at functional level. With these techniques we achieve significant area savings with affordable increments in processing time. Nevertheless, we have seen that in some cases the focus on the algorithm/application makes it possible to use a reduced number of coefficient circuits with almost no penalty, simply because the number and allocation of the coefficient circuits chosen, what we call S&S configuration, suits the shape of the templates in the algorithm [9]. These results reinforce the observation made in [10] that states that in a CNN implementation with 18 coefficient circuits many multipliers are idle during a significant part of the computation, so that we should use a reduced number of coefficient circuits in a CNN cell. Our target now is to examine the generality of this statement. In so doing, we review the templates found in the Cellular Wave Computing Library (CSW) [11] and some applications/algorithms reported in the CNN literature.

The first goal of this paper is to find out whether or not there are predominant shapes and symmetries in the templates. The second goal of the paper is to come up with the best configuration of coefficient circuits capable of approaching general templates and algorithms with the least possible average time penalty combining Split&Shift techniques and a smart use of the symmetries found in the templates. The paper is organized as follows. Section II reports the statistics from the CSW. Section III outlines statistics from CNN applications and algorithms. Finally Section IV draws the main conclusions from the paper.

II. Statistics from the CNN Template Library

The main goal of this section is to analize the shapes and symmetries in the CSW in order to find possible trends in the templates. In view of the results found we will try to determine which S&S configuration is the most suitable one for a general purpose implementation. We analyze only 3×3 templates, as larger neighborhood templates demand additional techniques to be realized over locally connected implementations. In particular, the realization of large neighborhood templates with S&S techniques is less sensitive to the chosen configuration than the 3×3 one [8]. In the following analyses, we use either image or partial output shifting [7] when it is necessary

Dense

1			2			3			4			5			6		
b	b	b	b	b	b	b	b	b	b	a	b	b	a	b	b	a	a
b	0	b	b	a	b	b	b	b	a	a	a	a	c	a	b	0	a
b	b	b	b	b	b	b	b	b	b	a	b	b	a	b	a	b	b

Diamond

7			8			9			10			11			12			13			14			15			16		
0	a	0	0	-a	0	0	a	0	0	a	0	0	0	0	0	0	0	0	-a	0	0	0	0	0	a	0	0	0	0
a	b	a	a	a	a	-a	a	-a	0	b	0	a	b	-a	-a	a	-a	0	a	0	a	b	a	0	b	0	a	a	-a
0	a	0	0	-a	0	0	a	0	0	-a	0	0	0	0	0	0	0	0	-a	0	0	0	0	0	a	0	0	0	0

17			18			19			20			21			22			23			24			25			26		
0	0	0	0	-a	0	0	a	0	0	0	0	0	0	0	0	a	0	0	0	0	0	0	0	0	0	0	0	0	0
-a	a	a	0	a	0	0	a	0	a	a	0	0	a	a	0	a	0	0	a	0	a	b	0	0	b	a	0	b	0
0	0	0	0	a	0	0	-a	0	0	0	0	0	0	0	0	0	0	0	a	0	0	0	0	0	0	0	0	a	0

27		
0	a	0
0	b	0
0	0	0

Diagonal

28			29			30			31			32			33			34			35			36			37		
-a	0	-a	-a	0	a	a	0	-a	0	0	-a	a	0	0	-a	0	0	0	0	-a	-a	0	0	0	0	-a	a	0	0
0	a	0	0	a	0	0	a	0	0	b	0	0	b	0	0	a	0	0	a	0	0	a	0	0	a	0	0	a	0
-a	0	-a	a	0	-a	-a	0	a	a	0	0	0	0	-a	0	0	-a	-a	0	0	0	0	a	a	0	0	0	0	-a

38			39			40			41			42			43			44			45			46		
0	0	a	0	0	0	a	0	0	0	0	a	0	0	0	b	0	0	0	0	b	0	0	0	0	0	0
0	a	0	0	a	0	0	a	0	0	a	0	0	a	0	0	a	0	0	a	0	0	a	0	0	a	0
-a	0	0	a	0	0	0	0	0	0	0	0	0	0	a	0	0	0	0	0	0	0	0	b	b	0	0

Other

47			48			49			50			51			52			53			54			55			56		
-b	0	b	0	0	0	0	0	0	-a	-a	-a	-a	0	0	0	0	-a	-a	-a	0	0	-a	-a	0	0	0	0	0	0
-a	0	a	a	a	a	0	a	0	0	a	0	-a	a	0	0	a	-a	-a	a	0	0	a	-a	0	a	-a	-a	a	0
-b	0	b	b	-a	b	-a	-a	-a	0	0	0	-a	0	0	0	0	-a	0	0	0	0	0	0	0	-a	-a	-a	-a	0

57			58			59			60			61			62			63			64		
a	a	0	0	a	a	a	0	0	0	0	-a	a	0	0	0	0	a	a	a	a	0	-a	0
a	b	0	0	b	a	0	b	a	a	b	0	a	b	-a	-a	b	a	0	b	0	0	b	0
0	0	-a	-a	0	0	0	a	a	a	a	0	a	0	0	0	0	a	0	-a	0	a	a	a

Fig. 1. Template coefficient distributions found in the CSW.

to achieve the lowest possible number of processing steps with the S&S techniques. It should also be noted that S&S techniques can only be executed in synchronous implementations.

Firstly we analize the allocation of the coefficients in the templates from the CSW within the sections "Basic Image Processing" and "Spatial Logic". For anisotropic templates, we have included all the possible directions (N, E, W, S, NE, NO, SE, SO) since that in general they have a different allocation of the coefficients. In a second step we have extracted the statistics for BW tasks due to its relevance in image processing. We also classify the templates as propagating and non-propagating ones. This is important because the application of S&S techniques requires synchronous processing, and a propagating template yields an iterative execution of B-type templates in a DTCNN. This might lead to many occurences of the same template, outnumbering the occurrences of the remaining templates in the algorithm under study. In such cases, the use of a configuration with a reduced number of coefficient circuits that suits the shape of the propagating templates would reduce the penalty in processing time.

A. General Analysis

Fig. 1 shows all the template coefficient distributions we have found in the CSW. We label different coefficient values as different letters. We also show explicitly the coefficients with the same value and opposite sign in order to allow the study of symmetries. Fig. 2 displays the number of occurrences for every template coefficient distribution. In this graph we can observe that distribution 2 (dense and radialy homogeneous) is the most abundant with 16 occurrences, followed by distribution 7 (diamond and radialy homogeneous as well) with 7 appearances. Nevertheless these distributions make up only 13% and 5% respectively of the 123 templates analyzed.

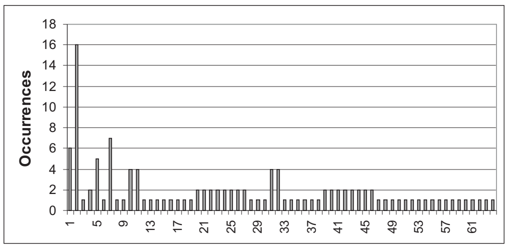


Fig. 2. Occurrences of the different template coefficient distributions (CSW).

In view of the high number of template coefficient distributions and the dispersion of occurrences, we make a new classification based on the communications of the cell under study with its neighbors. This new classification yields only four groups. In the Dense group, the cell exchanges information with all or almost all of the 8 nearest neighbors. In the Diamond group, the cell needs communications with at least one neighbor along the main cardinal directions. In the Diagonal group, the cell interacts with at least one neighbor along the NE, SE, NW and SW directions. We add a fourth category for the templates that do not fit any of the former groups. The result is shown in Fig. 3.

This graph shows that the Diamond group is the most fre-

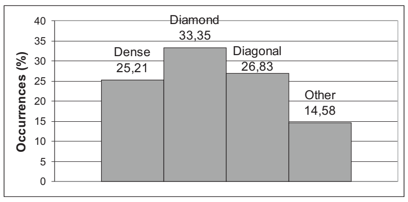


Fig. 3. The four main communication groups found in the CSW.

quent one, with up to one third of occurrences. Nevertheless, Dense and Diagonal groups have similar levels of appearances. From this graph and the analysis of the template coefficient groups from Fig. 1 we have to emphasize some issues before drawing conclusions. First of all, the percentage of templates with five or less non-null coefficients amounts to more than 70%. Such templates are those into the Diamond and Diagonal groups and some within the Other group. Furthermore, it is also interesting to note that most of the templates marked as Diamond and Diagonal have only two or three non-null coefficients, being the central coefficient one of them. This kind of sparse templates represents more than 50% of the total of the templates analyzed as it can be seen in Fig. 4. With regards to the symmetries, within the Dense group, only one (distribution 6 in Fig. 1, with a representation of less than 1% of all the templates analyzed) does not have axial symmetry.

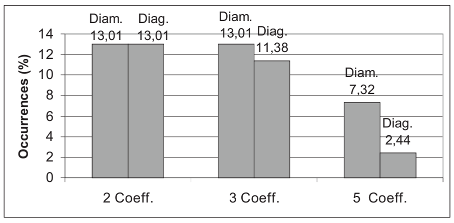


Fig. 4. Templates with less than five non-null coefficients (CSW).

From the data examined above we try to extract the most convenient S&S configuration to approach a general template. The first observation, according to the percentage of occurrences, is that a dense configuration is highly inefficient. On the other side of the spectrum, the barest of the S&S configurations, that is, three coefficient circuits (first configuration in Table I), is the best option to achieve the smallest area occupation [7]. Nevertheless, such an S&S configuration does not match any template coefficient distribution. This forces to have at least two shifts and one template application for the closest distributions. This is the case of some of the distributions with only two template coefficients (see Fig. 1).

If we add the central term to the three-coefficient-circuit (3cc) S&S configuration, which means four coefficient circuits, some of the distributions with only two template coefficients (2-c) can be done in only one step and many others in 3 steps. Also, a 4cc S&S configuration as the second in Table I permits to approach some of the 2-c templates of the Diamond group in two steps, but they mostly require three or more steps. It is also possible to improve in one step the realization of particular distributions (28-30, 47 and denses 1-5) making a smart use of symmetries by using the results that have been obtained in a different cell in a previous step[1]. Considering the central coefficient circuit with this configuration, which means five coefficient circuits, half of 2-c and 3-c distributions in Diamond, half of 2-c in Diagonal and some within Other can be done in one step. Also, the rest of 2-c and 3-c within Diamond are done in 2 steps. Again, the use of the symmetries allows for some reductions in the number of operations.

We can also consider a 4cc configuration with a diamond shape (third in Table I). This configuration has a higher matching with the distributions what leads to an improvement in the number of steps required, that is even bigger if we make use of symmetries. Furthermore, from Fig. 3, we see that the diamond configuration with the central coefficient realizes more than 30% of the templates in only one step, the Diagonal templates with 2 coefficients (13%) in two steps and the rest of diagonals in four steps (three for 28-30 distributions if we use symmetries). The dense distributions are realized with five steps in general, but almost all can be done in 3 steps if we take advantage of the symmetries that they show. Finally, all the distributions within the Other group can be approached in five or less steps, and some of them even in two steps if we take advantage of the symmetries they show.

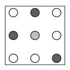
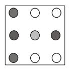
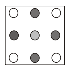
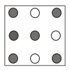
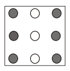
In order to make clear how to make a smart use of symmetries and without loss of generality, in the following we show the case of the five coefficient circuits configuration considered (third in Table I). We can see that at the same time that the contributions of the neighbors N, S and central are collected in the cell under study the contributions of the neighbors NW, W and SW are collected in the W neighboring cell as its own N, S and central neighbors contribution. This is the same with NE, E and SE neighbors contributions in the E cell. Taking the distribution 47 (Fig. 1) as example we can apply the (-b -a -b) weights in the central neighbors and simply collect the contributions of the right and left neigbours by gathering the results obtained in the W and E neighbors. This is done with a template that has a 1 as weight for the W neighbor and a -1 for the E neighbor. The result that was originally obtained in 5 steps is now obtained in only two steps.

The S&S configuration of six coefficient circuits shown as fourth in Table I is the smallest configuration that can approach the Diagonal group distributions in one step keeping the generality of the approach. This configuration can approach 31% of the templates in only one step, but in general we need at least four steps to implement the Diamond and Dense groups distributions. In this case (6 coefficient circuits) we can see that is more beneficial in general to have a parallel configuration (fifth in Table I) than the one with the central coefficient (fourth in Table I). The trade-off between area and speed of S&S configurations of seven and eight coefficient circuits is not worhty. Finally, dense configurations allow to

[1]For the sake of clarity we will give the explanation of how to use the symmetries with an example using the third configuration in Table I.

TABLE I

PERCENTAGE OF TEMPLATES REQUIRING A PARTICULAR NUMBER OF STEPS TO BE DONE WITH S&S FOR EACH IMPLEMENTATION, USING OR NOT USING SYMMETRIES. AVERAGE NUMBER OF STEPS AND RPO [7] TRADE-OFF VALUE FOR EACH CASE ANALYSIS FROM THE CSW TEMPLATE LIBRARY

S&S Config.	*1 Step* %-S&S %-S&S (Sym)	*2 Steps* %-S&S %-S&S (Sym)	*3 Steps* %-S&S %-S&S (Sym)	*4 Steps* %-S&S %-S&S (Sym)	*5 Steps* %-S&S %-S&S (Sym)	*More* %-S&S %-S&S (Sym)	*Average Number of Steps*	*RPO* (%)
3cc/+cent.	- / 10%	- / -	6% / 46%	20% / 4%	36% / 3%	38% / 37%	5.44 / 4.41	15.02 / 16.31
	- / 10%	- / -	16% / 46%	10% / 4%	36% / 3%	38% / 37%	5.34 / 4.41	15.36 / 16.31
4cc/+cent.	- / 21%	13% / 13%	30% / 9%	10% / 30%	47% / 27%	- / -	3.91 / 3.28	19.09 / 19.45
	- / 21%	16% / 13%	28% / 11%	23% / 53%	33% / 2%	- / -	3.72 / 3.02	20.46 / 21.95
4cc/+cent.	- / 33%	13% / 17%	44% / 7%	14% / 14%	29% / 29%	- / -	3.59 / 2.89	21.42 / 23.46
	- / 34%	20% / 17%	45% / 34%	11% / 11%	24% / 4%	- / -	3.39 / 2.36	23.24 / 32.73
5+1cc	31%	7%	13%	17%	32%	-	3.14	15.59
	31%	7%	14%	16%	32%	-	3.13	15.65
6cc	1%	13%	86%	-	-	-	2.85	17.98
	1%	23%	76%	-	-	-	2.76	18.98

apply all templates in one step. Clearly, this is the choice for applications with very tight speed requirements.

B. BW Processing and Propagative Operations

The relevance of BW processing in CNN algorithms and applications leads us to analyze this kind of templates in the CSW. We have found that in terms of percentage of occurrences they follow the same pattern as the general case (Fig. 3) but with a slight increase in Diamond and Diagonal groups at the expense of a decrease in the Dense group. On the other hand, as the templates from the CSW are designed for CTCNN, Table II provides a table of equivalences between CT and DTCNN operations, because S&S can only be applied in synchronous architectures. Fig. 5 shows the percentages of their occurrences in the CSW. From the equivalences to DTCNNs we see that the great majority of operations employs only one template (Classes A, D and E) or two but one with the central coefficient only (B or F). We have studied the templates involved in iterative DTCNN operations (Classes B,C and E) and we have concluded that they follow the same tendency of the general study presented above (Fig. 3 and Fig. 4).

C. Conclusions from the statistical study

From the statistical study of the templates from the CSW analyzed we draw several conclusions. The first one is that the S&S configuration of five coefficient circuits in diamond shape represents a good trade-off between hardware reduction

TABLE II

EQUIVALENCE BETWEEN CT AND DTCNN OPERATIONS.

	CTCNN		**DTCNN equivalence**
Class A	A = Central term only	B = Non-null template terms	Template B only
Class B	A = Non-null template terms	B = Central term only	Template B and Boolean functions with logic circuitry or two B central--term-only templates. Iteratively
Class C	A = Non-null template terms	B = Non-null template terms	Two B templates iteratively
Class D	A = Null	B = Non-null template terms	Template B only
Class E	A = Non-null template terms	B = Null	Template B iteratively
Class F	A = Central term only	B = Central term only	Boolean functions with logic circuitry or two B central--term-only templates
Class G	A = Central term only	B = Null	NA

and processing time for a general case. This is easily deduced from the average number of steps and the RPO value.Both are shown in the last columns of Table I. The RPO (area Reduction - in % - Per Operation added per initial operation) is a meassurement of the area-processing time trade-off that has been previously defined in [7]. In this case the RPO values were calulated with the average number of steps and with reference to a 9cc configuration. It is also remarkable the

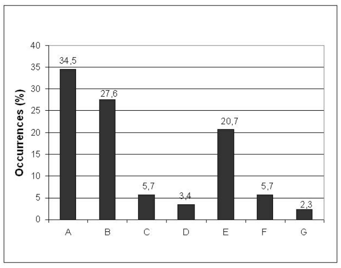


Fig. 5. Percentages of the DTCNN classes of Table II.

high percentage of templates with only two or three non-null coefficients (Fig. 4). This can suggest an S&S configuration with only one, two or three coefficient circuits that can be used alternatively (time-multiplexed) to weigh neighbors values in a diamond shape [12], [13]. This means one, two or three coefficient circuits instead of five, but four intercell connections.We also have seen the importance of the central coefficient circuit in all the cases. Moreover, it could even be interesting to have an extra central coefficient circuit in a second template, especially for those operations in class B (Table II) that are iterative. These are the conclusions drawn from the CSW. Nevertheless, we also have to review CNN algorithms and applications to see whether or not real-life applications follow the same tendency. Next section deals with this issue.

III. Statistics from CNN Applications

This Section outlines the analysis of 5 applications found in the CNN literature, three of them selected from the former CNNA [14]. We provide a brief description of every application along with their template statistics. To probe further on every application, the reader is addressed to the corresponding references. Even though it is hard to draw a general conclusion from such a short list of applications, it is still possible to find out certain trends in the shape of the templates used.

The first application is that of visual inspection of metal objects [15]. The algorithm proposed therein is quite simple. A diffusion is applied over the input image (gray-scale), followed by an image subtraction, a threshold, an AND, a variable number of erosions (1 to 9 depending on the metal object under study), and a final recall. Image subtraction and threshold would be easy to implement with non-zero central elements in the templates. Diffusion and recall are dense templates. The erosion template can be either 4-connectivity or 8-connectivity. In this analysis we have supposed 4-connectivity erosion templates, and we have set the number of erosions to 5. Fig. 6 displays a summary of template statistics for this application.

The second application is about vein feature extraction through DTCNNs [16]. The algorithm is too complex to be wholly addressed here. After a binarization, the image is subject to skeletonization, isolated pixel removal, ending detection, bifurcation detection, Boolean operators, false feature elimination and figure reconstruction. Each one of the former stages comprises several templates, most of them dense. Figure

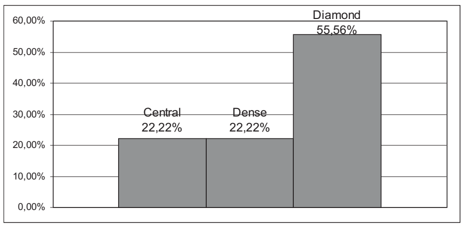


Fig. 6. Template statistics for the application of visual inspection [15].

7 shows a summary of template statistics for this application.

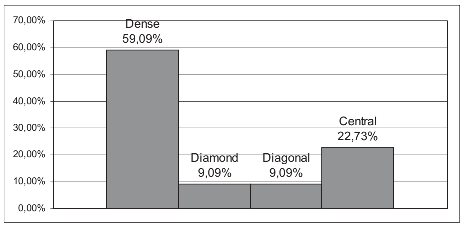


Fig. 7. Template statistics for the application of vein feature extraction [16].

The third application is wheelchair driving [17]. The algorithm contains Boolean operators, shifts, and the templates recall and shadow. The template statistics for this application can be seen in Fig. 8.

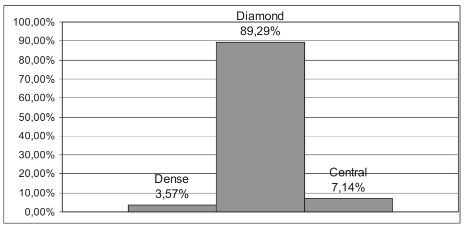


Fig. 8. Template statistics for the application of wheelchair driving [17].

The fourth application is that of Pixel-Level Snakes (PLS). We analyse the version discussed in [5]. Such a version contains Boolean operators, as well as hit-and-miss, and erosion and dilation templates. Also, a propagating template, the hole filling is included. The hole filling is performed by running a B-type template iteratively. We set an upper bound of N/2 for the number of iterations, being $N \times N$ the image size. Also, every template is formulated as a binary template, i.e. every coefficient can be either 0 or 1. The results from template statistics are depicted in Fig. 9. These data have been extracted from a 64×64 image. The percentages of Fig. 9, however, barely change with the image size.

The final case analyzed here is an application of robot guiding [18]. Here we have reformulated the original templates into binary templates. The proposed algorithm comprises hole filling, CCD, shadow, hit-and-miss, binary edge detection, small killer and Boolean operators. The application encompasses many propagating templates. As in the case of the PLS, we have made the analysis for 64×64 images, and in this case

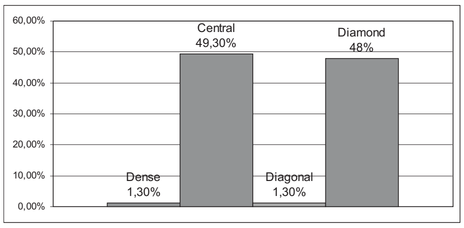


Fig. 9. Template statistics for PLS [5].

we have set an upper bound of 64 iterations for propagating templates. Again, the percentages remain the same regardless the image size. The template statistics are shown in Fig. 10.

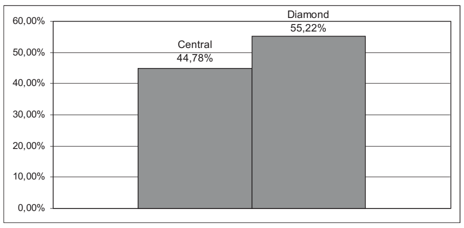


Fig. 10. Template statistics for robot guiding [18].

Although the study reported here cannot be posed as conclusive, we can draw several conclusions. The first one is that Diamond, Dense and Central are the shapes with the highest number of occurrences. Irregular or Diagonal templates are rare. Templates with only non-zero central coefficients in the applications studied are employed for Boolean operations. Diamond templates would be implemented with a reduced number (five) of coefficient circuits. Finally, if dense templates have symmetries, these can be easily approached started from a diamond template applying a split and shift methodology [7]. This is the case of the most of the templates of the five applications discussed here.

IV. Conclusion

This paper has reviewed the CSW and some CNN algorithms/applications found in the literature in order to confirm the general statement of whether a reduced number of multipliers/coefficient circuits yields area and time efficient Cellular Non-linear Networks. The study has shown that in the great majority of the cases analyzed, having 18 or even 9 coefficient circuits is not the best option, unless speed requirements are too tight. In a synchronous architecture, a reduced number of coefficient circuits combined with the S&S methodology is a better option. This study shows that the S&S configuration of five coefficient circuits allocated in a diamond shape is the best option. Such a solution covers a large percentage of the templates of the CSW and CNN algorithms/applications almost without extra processing time. Furthermore, with a smart use of symmetries, there are barely templates not implementable with only two or three steps by means of S&S techniques. In fact, they represent only around a 15% of the templates analyzed, and they are approached with only 4 or 5 steps. An extra template of only one non-zero central coefficient can be useful as well for logic operations implemented with CNN templates.

As future work, a more in-depth analysis of algorithms/applications will be presented. Also, hardware approaches that take advantage of symmetries will be dicussed more throughly.

Acknowledgment

This work is funded by Xunta de Galicia under the project PGIDT06TIC10502PR and by the Ministerio de Educación y Ciencia under the FPU program.

References

[1] International Technology Roadmap for Semiconductors (ITRS), *http://public.itrs.net*.
[2] C. Baldanza et al., *A Cellular Neural Network for Peak Finding in High-Energy Physics*, Proceedings of the 2000 6th IEEE International Workshop on Cellular Neural Networks and their Applications Proceedings, CNNA2000, pp. 443–448, 2000.
[3] A. Lopich and P. Dudek, *Architecture of Asynchronous Cellular Processor Array for Image Skeletonization*, European Conference on Circuit Theory and Design, ECCTD'05, Cork, Ireland, vol. 3, pp. 81–84, 2005.
[4] T. Komuro et al., *A Digital Vision Chip Specialezed for High-Speed Target Tracking*, IEEE Transactions on Electron Devices, vol. 50, n. 1, pp. 191–199, 2003.
[5] V.M. Brea et al., *A Binary-Based On-Chip CNN Solution for Pixel-Level Snakes*, International Journal of Circuit Theory and Applications, vol. 34, pp. 383–407, 2006.
[6] M. Laiho et al., *Template Design for Binary-Programmable Cellular Nonlinear Networks*, 2005 IEEE International Symposium on Circuits and Systems, ISCAS 2005, pp. 3938–3941, 2005.
[7] Natalia A. Fernández-García et al., *On the Reduction of the Number of Coefficient Circuits in a DTCNN cell*, Proceedings of the 8th IEEE International Workshop on Cellular Neural Networks and their Applications, pp. 29–34, 2006.
[8] Natalia A. Fernández-García et al., *Hardware Simplification in Cellular Non-linear Networks for Complex Algorithms*, XXI Conference on Design of Circuits and Integrated Systems, DCIS 2006.
[9] Natalia A. Fernández-García et al., *Area and Time Efficient Cellular Non-linear Networks*, 2007 IEEE International Symposium on Circuits and Systems, ISCAS 2007, pp. 2682–2685, 2007.
[10] Piotr Dudek, *Accuracy and Efficiency of Grey-level Image Filtering on VLSI Cellular Processor Arrays*, Proceedings of the 2006 10th IEEE International Workshop on Cellular Neural Networks and their Applications, pp. 123–128, 2004.
[11] CSW, *Cellular Wave Computing Library (Templates, Algorithms, AND Programs), Version 2.1, CSW-1-2007*, Budapest 2007.
[12] Piotr Dudek, *A Programmable Focal-Plane Analogue Processor Array*, PhD, University of Manchester, 2000.
[13] F. Sargeni et al., *Time Division Digital Programmable OTA for Cellular Neural Networks*, European Conference on Circuit Theory and Design, ECCTD'05, Cork, Ireland, vol. 1, pp. 75–78, 2005.
[14] CNNA 2006, *Proceedings of the 2006 10th IEEE International Workshop on Cellular Neural Networks and their Applications*, Istanbul, Turkey, 28-30 August 2006.
[15] Zoltan Szlavik et al., *Visual Inspection of Metal Objects by Using Cellular Neural Networks*, Proceedings of the 2006 10th IEEE International Workshop on Cellular Neural Networks and their Applications, pp. 142–146, 2006.
[16] Suleyman Malki et al., *Vein Feature Extraction Using DT-CNNs*, Proceedings of the 2006 10th IEEE International Workshop on Cellular Neural Networks and their Applications, pp. 307–312, 2006.
[17] Samuel Xavier-de-Souza et al., *Fast and Robust Face Tracking for CNN Chips: Application to Wheelchair Driving*, Proceedings of the 2006 10th IEEE International Workshop on Cellular Neural Networks and their Applications, pp. 200–205, 2006.
[18] Giovanni Egidio Pazienza et al., *Robot Vision with Cellular Neural Networks: A Practical Implementation of New Algorithms*, International Journal of Circuit Theory and Applications, vol. 35, pp. 449–462, 2007.

ISCAS12:

N.A. Fernández-García, V.M. Brea, D. Cabello, " **Scale- and Rotation- Invariant Feature Detectors on Cellular Processor Arrays** ", IEEE International Symposium on Circuits and Systems, 2012. ISCAS 2012. pp., Seoul, Korea, May 2012.

Scale- and Rotation- Invariant Feature Detectors on Cellular Processor Arrays

N. A. Fernández, V.M. Brea, M. Suárez, D. Cabello
Centro de Investigación en Tecnologías de la Información, (CITIUS)
University of Santiago de Compostela
Santiago de Compostela, Spain
Email:victor.brea@usc.es

Abstract— **This paper assesses the implementation of scale- and rotation-invariant feature detectors on Cellular Processor Arrays (CPA). Scale- and rotation-invariant feature detectors are complex image processing algorithms with a high computational burden in the low-level image processing stage due to large-neighborhood convolution-type operations. Such operations are used to generate the so-called scale-space. This paper outlines different options to provide the scale space in the Scale Invariant Feature Transform (SIFT) and the Speeded-Up Robust Features (SURF) algorithms on CPAs with pixel-per-processor assignment. The paper shows that it is feasible to do this even with a reduced set of inter-processor communications within acceptable time limits on existing CPAs.**

I. Introduction

Scale- and rotation- invariant feature detectors are used for different image processing tasks as object detection and classification, image retrieval, image registration or tracking [1]. Their invariant nature yields repeatability, which permits to deal with occlusion, or with scenes taken under different conditions such as illumination, or a different view angle.

Scale Invariant Feature Transform (SIFT) [2] and Speeded-Up Robust Features (SURF) [3] are the most widely known scale- and rotation-invariant feature detectors. In SIFT, part of the low-level image processing relies on convolution-type operations in large neighborhood regions. In SURF, large neighborhood filters are also run, but on the so-called integral image, which is the result of adding the pixel values of all the pixels located up and to the left of the pixel of interest. In both, large neighborhood convolutions and integral image calculation, to have data parallelism at pixel level is a challenge. Indeed, only the work reported in [4] presents a pixel-per-processor assignment for low-level image processing for SIFT in a reduced window, reaching 1 frame/s in VGA.

Cellular Processor Arrays (CPA) are matrices of simple Processing Elements (PEs) locally interconnected. This architecture responds to the Single Instruction Multiple Data (SIMD) paradigm. With the approppriate adaptations we can take advantage of the degree of parallelism on a CPA with a pixel-per-processor assignment in tackling both convolution-type operations with large neighborhood kernels and the integral image calculation. In the former case, either we count on inter-PE routing and circuits to connect pixels lying at long distances from one another, or we opt for specific techniques for local connections to cope with large neighborhood. The latter is the best option as it gives the highest integration density. The price is longer processing times. In the second case, integral image additions can be group to achieve enough parallelism to take adavange of CPAs. This paper addresses different techniques to approach large neighborhood kernels and the integral image with CPAs with local connectivity while keeping the low-level image processing stages of SIFT and SURF within acceptable time limits. The results are given in terms of number of clock cycles or steps. We prove that existing CPA implementations can give reasonable times.

II. Split and shift techniques

Split and Shift ($S\&S$) is a methodology aimed at executing convolution-type operations on a CPA with a set of inter-PE connections and weighing circuits overpassed by the kernel requirements [5,6]. This can imply both implement large neighborhood kernels over local connected hardware, or apply large neighborhood or minimun-sized kernels over a reduced set of inter-PE connections and weighing circuits. For instance, a 3×3 or larger kernel can be run with only 3, 4 or 5 weighing circuits per PE, and their corresponding inter-PE circuits, instead of 9 weighing circuits per PE.

$S\&S$ techniques are feasible in convolution-type operations, where the associative property holds. These techniques are valid on CPAs where the signals are predictable and accessible at every clock cycle like synchronous ones. The $S\&S$ methodology is run in two phases: 1) we group the kernel coefficients into different sub-kernels, this is the split phase, and 2) we collect the results of applying the sub-kernels at the PE under study by shifting the sub-kernel outcomes, or by shifting the inital image prior to the sub-kernels, this is the shift phase. It should be noted that shifts are executed as specific kernels on CPAs. Different time performances come out with different split and shift methods. More details of these techniques can be found in [5, 6]. This paper presents the most suitable $S\&S$ techniques for scale-space generation in SIFT and SURF. It should be noted thet shifts are executed as specific kernels on CPAs.

III. Feature detectors assessment on CPA

A. SIFT

The first stage in SIFT, scale-space extrema detection, implies several Gaussian filters of increasing σ values to generate a group of images, known as scales or σ levels, called octave. Three to four octaves with six images each are usually needed. Every new octave is generated from a half resolution image obtained by downscaling with a $1/4$ factor an image from the former octave, i.e. going from an $M \times N$ to $M/2 \times N/2$ pixels.

If we assume 4σ as the order of neighborhood for the Gaussian filters truncation in their approach from the continuous space to discrete kernels, the kernels will be of size $(8\sigma+1)\times(8\sigma+1)$. Considering an initial σ value of 1.6, and a factor between sigmas of $2^{1/3}$, the scale space generation will require six kernels per octave of sizes 13×13, 17×17, 21×21, 27×27, 33×33 and 41×41. It should be emphasized that the Gaussian kernels do not change across octaves. The only change lies in the image resolution.

By using the $S\&S$ methodology to implement the required large neighborhood templates over a CPA we will need 1796 operations (sub-kernels application and shifts) per octave with 9 inter-PE circuits in a neighborhood radio $r=1$ (8 neighbors plus the feedback term). Given that 2D Gaussian filters are separable in horizontal and vertical 1D Gaussian filters, we can consider $N\times 1$ and $1\times N$ kernels. This drops the number of $S\&S$ operations per octave to 372 on a CPA with 9 inter-PE circuits. A classical NEWS (North-East-West-South) configuration with 4 inter-PE connections along with a central circuit to weigh the value at the PE under study yields the same number of operations.

Until here we have considered that all the σ levels or scales are provided by applying the Gaussian filters on the original image. Nevertheless, if the Gaussian filters are run on the previous filtered image or scale, each octave would need six kernels of sizes: 13×13, 5×5, 5×5, 7×7, 7×7, and 9×9. With the $S\&S$ methodology and the 1D separation this would result in 92 steps per octave on a CPA with 9 inter-PE circuits. Nevertheless, the most efficient way of emulating a large neighborhood kernel is to run a 3×3 kernel recursively. This holds as long as the 3×3 seed is known, as is the Gaussian filtering case. This would lead to 67 3×3 Gaussian operations per octave when obtaining each image from the original one, and 19 when using the previous filtered image. In a CPA with 5 inter-PE circuits (NEWS plus feedback circuit) these numbers would be multiplied by 3 when applying the $S\&S$ techniques to the Gaussian particular case.

However, from the second octave on every pixel does not interact with its nearest neighbors, but with pixels at more distant locations due to the downscaling process. For instance, in the second octave the pixel at i,j interacts along E and S directions with the pixels located at $i,j+2$ and $i+2,j$. The pixels in between are not used anymore. The distance is increased up to $i,j+4$ and $i+4,j$ for the third octave and to $i,j+8$ and $i+8,j$ for the fourth one due to the successive downscalings. In a CPA with a pixel-per-processor-approach we can deal with this situation by expanding the 3×3 Gaussian seed used in the first octave and applying the $S\&S$ methodology to implement the resultant large neighborhood kernels. Eq. (2) shows the result of the expansion of a generic 3×3 kernel in a 5×5 one caused by a half resolution downscaling (i.e. second octave). The order of neighborhood doubles in each downscaling and thus seeds of 9×9 and 17×17 pixels appear for the third and fourth octaves.

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \tag{1}$$

$$\begin{pmatrix} a & 0 & b & 0 & c \\ 0 & 0 & 0 & 0 & 0 \\ d & 0 & e & 0 & f \\ 0 & 0 & 0 & 0 & 0 \\ g & 0 & h & 0 & i \end{pmatrix} \tag{2}$$

The $S\&S$ techniques take advantage of the sparse character of the new large neighborhood seed kernels for every octave. In addition, the decomposition of the Gaussian filter in its horizontal and vertical components is advantageous, as we reduce the number of long-distant pixels. The sparse 5×5 seed template of Eq. (2) requires 8 $S\&S$ operations regardless 2D Gaussian filtering or the 1D approach along horizontal and vertical directions (H/V). In the case of the 9×9 kernel we need 33 operations in the 2D Gaussian case, and 18 for H/V, becoming 65 and 34 steps respectively for the 17×17 kernel. With these numbers we see that the first octave is realized with 19 operations (38 with H/V), a second octave would require 152 $S\&S$ steps for both options, a third octave would need 627 for the 2D Gaussian filter, and 342 for H/V, and a fourth octave would require 1235 steps in the first case and 646 steps in the H/V one. This makes a total of 1159 steps for a SIFT process with 3 octaves and 1805 for 4 octaves with 9 inter-PE circuits in both cases and H/V consideration. A NEWS configuration with an additional feedback weighing circuit leads to just 19 additional operations for the first octave with the H/V option (for the rest of the octaves the same numbers remain) with a significant reduction in area per PE .

Although these numbers might be affordable for SIFT-based video rate applications, given for example 1 μs per operation, they can be slightly improved if we combine the application of the $S\&S$ techniques with Berni's proposal in [7]. Therein it is said that with an appropriate set of operations we can implement 3×3 kernels with certain symmetries (Eq. (3), which correspond to the symmetries shown by the discretized 2D Gaussian filter) with "2×2" kernels (Eq. (4)). This is especially advantageous in the SIFT for the seed kernel expansion caused by the image downscaling through octaves. With this method the second octave can be generated through a new 3×3 seed kernel, the third one with a 5×5 kernel, and a fourth one would need just a 9×9 seed kernel. Eq. (5) and Eq. (6) list the kernels for the second and third octaves respectively. The kernel for the fourth octave will be similar, but with a 9×9 neighborhood.

$$\frac{1}{4}\begin{pmatrix} a & a+b & b \\ a+c & a+b+c+d & b+d \\ c & c+d & d \end{pmatrix} \tag{3}$$

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & a & b \\ 0 & c & d \end{pmatrix} \tag{4}$$

$$\begin{pmatrix} a & 0 & b \\ 0 & 0 & 0 \\ c & 0 & d \end{pmatrix} \tag{5}$$

$$\begin{pmatrix} a & 0 & 0 & 0 & b \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ c & 0 & 0 & 0 & d \end{pmatrix} \tag{6}$$

Berni's proposal makes use of a resistive network (RC grid). Its method starts with two consecutive averaging processes within groups of 2×2 pixels. Berni states that if we take the half resolution image from the second averaging instead from the first one the application of 3×3 kernels like the one shown in Eq. (3) can be realized through the application of the "2×2" kernel in Eq. (4) [7]. We implement Berni's proposal on synchronous CPAs just with the averaging given by the kernel of Eq. (7). Hence, in our case the result of applying the template in Eq. (3) over the original image is the same as the one in Eq. (4) over the "averaged" image obtained with Eq. (7). Both, averaging (Eq. (7)) and "2×2" equivalent (Eq. (4)) kernels, are expanded through octaves.

$$\frac{1}{4}\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{7}$$

All in all, the combination of $S\&S$ techniques with Berni's proposal [7] on a pixel-per-PE CPA with 9 inter-PE circuits yields 988 $S\&S$ operations for four octaves (380 for three octaves). This number increases up to 1064 steps for four octaves (456 for three octaves) if we consider a NEWS configuration. Slightly better results are met if we combine the H/V option for the first octave and Berni's proposal for the rest of octaves, producing 969 (361) and 1026 (418) steps respectively. These results do not depend on the image resolution. This means that we could implement the SIFT scale-space generation over CPAs with very acceptable processing times by means of $S\&S$ techniques. For instance, if the clock cycle was only 1 MHz, as is the case of the implementations reported in [8, 9], and each $S\&S$ operation takes one cycle, the scale-space generation would be ready in $\sim$1 ms, leaving a relatively long time for the rest of operations, which might be enough for video rate processing.

B. SURF

The SURF (Speeded-Up Robust Features) [3] consists of three main steps: the interest point detection, description and matching. We focus on the detection phase. As in SIFT, the interest points in SURF need to be found at different scales (Gaussian σ values). The scale space is again divided into octaves composed by a series of images produced by convolving the original image with increasing size filters (increasing σ values). Such filters are 2-D Gaussian second order derivatives along horizontal (xx), vertical (yy) and diagonal ($xy = yx$) directions that conform the Hessian matrix. The 9×9 filter is considered the initial scale, and its size is increased in 6 pixels on each dimension in the first octave until having the four filtered images per octave required by the algorithm. For the first octave we have, then, filters of sizes 9×9, 15×15, 21×21 and 27×27 pixels. Each new octave begins at the second filter of the previous octave, and the neighborhood order difference between successive filters doubles with respect to the previous octave (e.g. filters in the second octave will be of sizes 15×15, 27×27, 39×39, etc.) [3]. According to the image size three or four octaves can be needed, yielding filters up to 195×195 pixels (four octaves). This filter application implies a high computational burden on a serial processor. The same happens with an $S\&S$ approach over CPAs.

Nonetheless, the combination of the so-called box filters with the well-known "integral image" eases the computational burden [10]. Box filters are kernels divided in rectangular areas with a common weighting value within each region. The integral image (II) is an intermediate image representation which gives each pixel the value of the summation of all the pixels from itself to the left and above in the original image. The integral image reduces the box filters application to around 10-15 additions each (3 per rectangular area plus 2 or 3 for the areas combination) regardless the filter size. The difficulty now roots in the computational burden of the integral image. Ref. [10] introduces recurrences to avoid redundant operations. With this the number of additions is reduced to 2NM for an $N \times M$ image.

CPAs can parallelize the II computation reaching $N + M$ steps in an image of resolution $N \times M$ pixels (1120 steps in a VGA image, for example). Fig. (1) illustrates the CPA II calculation through an ad-hoc optimized $S\&S$ technique. In this case we reduce the number of sub-kernels to one that adds the N, W and NW pixels to the central one (A1 in Fig. (1)). Afterwards, the partial outputs are, first, horizontally gathered in the corresponding pixels by shifting the obtained image two shifts to the right (B1 kernel) repetitively, and by accumulating the successive shifted versions of the image. When the horizontal accumulation has finished we have two

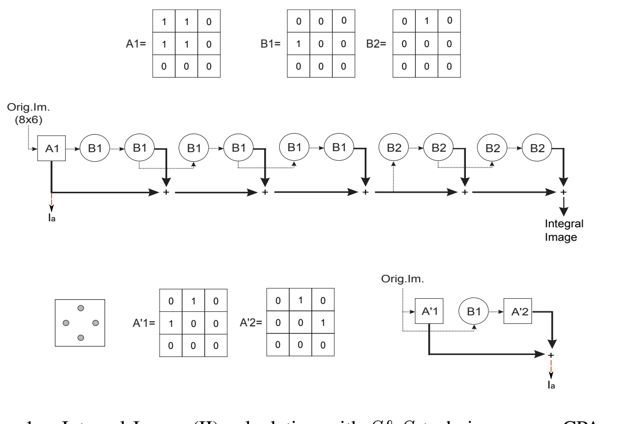


Fig. 1. Integral Image (II) calculation with $S\&S$ techniques on a CPA.

TABLE I

NUMBER OF $S\&S$ OPERATIONS FOR SCALE SPACE GENERATION IN SIFT AND SURF

	SIFT (Seed recursion with previous scale)							**SURF** (II + Box filters)
Num. Oct.	**9wc**	**9wc + HV**	**5wc (HV)**	**9wc + Berni**	**5wc + Berni**	**9wc HV/Berni**	**5wc HV/Berni**	**(9cc or NEWS)**
3	798	532	532	380	456	361	418	N+M+1584
4	2033	1178	1178	988	1064	969	1026	N+M+3185

Note: "wc" refers to the number of weighting circuits per PE.

rows of the II calculated. The rest of the rows are achieved by repeating the same process in the vertical-down direction (B2 template) but now the shifted image is the one obtained from the horizontal accumulation. We can use a reduced version of the NEWS configuration by just implementing the A1 operation in three steps as is shown in the lower part of Fig. (1). I_a identifies the image obtained through both ways which is the starting point for the horizontal shifting.

It is interesting to note that this is just one of the possibilities of implementing the II with CPAs. We have analyzed several different possible ways with similar results in number of CPA operations. We can, as well, directly translate the proposal in [10] by accumulating the pixel values in each row in a column fashion way and, once this row cumulative image is calculated (S in [10]), calculating the II in a row fashion way by accumulating the values of the pixels in the columns. It requires, again, $N + M$ CPA operations. This computation makes most of the hardware idle during the whole process (just one row/column working at a time), which makes us think of an SIMD with lesser degree of parallelism than a pixel-per-processor CPA. This might be even necessary as the integral image yields very wide words, leading to PEs with a larger area. The approach reported in [11] reduces to 19 bits the word lenght needed by SURF. Proposals as Linear Processor Arrays (LPA) in [12] are promising. If 16-bit words sufficed for SURF, their 320 16-bit PEs working at 125MHz would lead to 560 steps for the integral image calculation in less than 5μs for a QVGA image.

Once we have the integral image we can apply the box filters to generate the scale-space of the SURF algorithm. Although box filters require the same number of operations on a serial processor independently of the kernel size, this does not hold for a CPA, where the four pixels summation is implemented through sparse large neighborhood kernels. For instance in a box filter of size $Q \times Q$, we would require $5Q/4$ $S\&S$ operations for the xx or yy Hessian matrix elements, and $4Q/3$ for xy/yx ones. Again, it would be enough with a NEWS configuration. Given PEs with wide enough words (at least 19 bits) the whole scale space generation for an $N \times M$ image would take $N + M + 3184$ CPA operations for four octaves and $N + M + 1584$ for three octaves, less than 30μs for four octaves (less than 20μs for three) for a QVGA image on a processor as Xetal [12].

IV. SUMMARY AND CONCLUSIONS

This paper has assessed the feasibility of the scale-space generation for two modern scale- and rotation-invariant feature detectors, SIFT and SURF, on CPAs with pixel-per-processor assignment. The time burden of the scale-space generation in SIFT arises from convolution-type operations with large neighborhood kernels on many images of different resolutions. The challenge with SURF is to calculate the integral image and to apply the large neighborhood box filters with a CPA. The paper shows that with the so-called $S\&S$ techniques is possible to execute the scale-space of both feature detectors with a low enough number of steps or clock cycles as to have very low times with implementations in current CMOS technologies. The scale-space generation of SIFT could be available in $\sim$1ms for a pixel-per-processor approach at 1MHz. The SURF scale-space would be ready in less than 30μs with a linear processor array like the one reported in [12] for a QVGA image. Table I summarizes data on number of operations, which can be taken as clock cycles, for the different options employed in the scale-space generation of SIFT and SURF throughout this paper.

V. ACKNOWLEDGMENT

This work has been funded by Xunta de Galicia (Spain) and MICINN (Spain) through projects 10PXIB206037PR and TEC2009-12686.

REFERENCES

[1] Tinne Tuytelaars, and Krystian Mikolajczyk,*"Local Invariant Feature Detectors: A Survey"* , Foundations and Trends in Computer Graphics and Vision, Vol. 3, No. 3, (2007) 177-280.

[2] D.G. Lowe,*"Distinctive Image Features from Scale-Invariant Keypoints"* , Int. J. Comput. Vis., Vol. 60, no. 2, pp. 91-110, 2004.

[3] Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool,*"SURF: Speeded Up Robust Features"* , Computer Vision and Image Understanding, Vol. 110, no. 3, pp. 346-359, 2008.

[4] Seungjin Lee et al.,*"A 345 mW Heterogeneous Many-Core Processor With an Intelligent Inference Engine for Robust Object Detection"* , IEEE journal of Solid-State Circuits, vol. 46, no. 1, pp. 42-51, Jan. 2011.

[5] N. A. Fernández et al.,*"On the Emulation of Large-Neighborhood Templates with Binary CNN-based Architectures"* , CNNA 2005, pp. 274-277, 2005.

[6] N. A. Fernández et al.,*"On the Reduction of the Number of Coefficient Circuits in a DTCNN Cell"* , CNNA 2006, pp. 29-34, 2006.

[7] J. Fernández-Berni et al.,*"Image Filtering by Reduced Kernels Exploiting Kernel Structure and Focal-Plane Averaging"* , ECCTD 2011, pp. 229-232, 2011.

[8] P. Dudek and P. J. Hicks,*"A General-Purpose Processor-per-Pixel Analog SIMD Vision Chip"* , In *IEEE Transactions on Circuits and Systems-I: Regular Papers*, vol. 52, no. 1, pp. 13-20, 2005.

[9] Ángel Rodríguez-Vázquez et al.,*"ACE16K: The Third Generation of Mixed-Signal SIMD-CNN ACE Chips Towards VSoCs"* , In *IEEE Transactions on Circuits and Systems-I: Regular Papers*, vol. 51, no. 5, pp. 851-863, 2004.

[10] P.A. Viola et al.,*"Rapid object detection using a boosted cascade of simple features"* , In *CVPR (1)*, pp. 511-518, 2001.

[11] Shoaib Ehsan et al,*"Novel Hardware Algorithms for Row-Parallel Integral Image Calculation"* , 2009 Digital Image Computing: Techniques and Applications, pp. 61-65, 2009.

[12] Anteneh A. Aboo et al.,*"Xetal-II: A 107 GOPS, 600 mW Massively Parallel Processor for Video Scene Analysis"* , IEEE J. of Solid-State Circuits, vol. 43, no. 1, pp. 192-201, Jan. 2008.

Appendix B

FPGA implementations using S&S methodology

Along this work we have collaborated with the Departament d'Electrónica, Enginyeria i Arquitectura La Salle (Universitat Ramon Llull, Barcelona, Spain) in the development of B/W and G/S FPGA implementations with CNN Universal Machine functionality by applying the Split and Shift methodology.

The methodology was successfully applied to obtain a 25×25 B/W implementation on an Altera Stratix-EP1S25 FPGA. The first version of this implementation is shown in the DCIS08-1 paper. This implementation was compared to an SIMD fine grained implementation in the DCIS08-2 paper. The SIMD implementation was developed within our research group. It is based on the translation to logic operations of the binary templates, and it also uses the shifting of images to achieve pixels not-directly connected to the cell. Paper ECCTD07-2 relates the SIMD paradigm, the threshold-logic, and the DTCNNs with S&S, identifying the characteristics that could formulate them under a common model.

The G/S FPGA realization implementing the S&S techniques is gathered in the ECCTD09-1 paper. This implementation functionality was proved in [Albó and Canals et al., 2010] and [Albó-Canals et al., 2010]in the implementation of a robot guiding system. [Consul-Pacareu et al., 2011] evolves the last referred work to a low-cost, low-power consumption and full functionality intelligent camera device for robot vision, and it assesses the power consumption having as a result that between equal implementations realized with different number of CC, the instant power consumption is smaller in the one with less CC, but the total power consumption is bigger due to the more number of steps, as it was expected. Nevertheless the lower cell area occupation leads to a bigger grid that reduces the windowing process and can then compensate for the increment in the power consumption.

Published Papers

DCIS08-1:

Jordi Albó-Canals, N.A. Fernández-García, Jordi Riera-Baburés, Victor M. Brea, Diego

Cabello, " **Discrete Time Cellular Non-linear Networks Implementation over FPGA**," in *Proceedings of the XXIII Conference on Design of Circuits and Integrated Systems, DCIS 2008*, Grenoble, France, November 2008.

DCIS08-2:

A. Nieto, N.A. Fernández-García, Jordi Albó-Canals, V. M. Brea, D. L. Vilariño, Jordi Riera-Baburés, Diego Cabello-Ferrer, " **Single Instruction Multiple Data and Cellular Non-linear Networks as Fine-Grained Parallel Solutions for Early Vision on FPGAs**," in *Proceedings of the XXIII Conference on Design of Circuits and Integrated Systems, DCIS 2008*, Grenoble, France, November 2008.

ECCTD07-2:

V. M. Brea, M. Laiho, N. A. Fernández, A. Paasio, D. Cabello. " **Relating Cellular Non-linear Networks to Threshold Logic and Single Instruction Multiple Data computing models**,"in *Proceedings of the 18th European Conference on Circuit Theory and Design, 2007. ECCTD 2007*, pp.92-95, Seville, Spain, August 2007.

ECCTD09:

J. Albó-Canals, J.A. Villasante-Bembibre, J. Riera-Baburés, N.A. Fernández-García, V.M. Brea, "**An efficient FPGA implementation of a DT-CNN for small image gray-scale pre-processing**," in *Proceedings of the European Conference on Circuit Theory and Design, 2009. ECCTD 2009*, pp.839-842, Antalya, Turkey, Aug. 2009.

Discrete Time Cellular Non-linear Networks Implementation over FPGA

Jordi Albó-Canals[§], N.A. Fernández-García[*], Jordi Riera-Baburés[§], Victor M. Brea[*] and Diego Cabello[*]
[*]Departamento de Electrónica e Computación, Universidade de Santiago de Compostela
E-15782 Santiago de Compostela, Galiza, Spain. E-mail:natalia.fernandez@usc.es
[§]Departament d'Electrónica, Enginyeria i Arquitectura La Salle, Universitat Ramon Llull
E-08022 Barcelona, Spain. E-mail:jalbo@salle.url.edu

Abstract— **This paper introduces a parallel Cellular Non-linear Network implementation on an FPGA. Such an approach is intended for speeding-up early-vision applications, mainly in images with low resolution ($\leq 50 \times 50$). Bigger images can be processed with an efficient computation time by means of windowing. Our implementation has been realized over an Altera Stratix-EP1S25F672 FPGA achieving a parallel implementation of a 25×25 effective grid.**

I. Introduction

Massive parallel architectures have made it possible to tackle problems with high computational burden in acceptable time slots. Cellular Non-linear Networks (CNNs) have traditionally been posed as an example of practical solution to the on-chip realization of Single Instruction Multiple Data (SIMD) architectures for image computation with direct pixel to cell assignment [1]. A CNN consists of a grid of identical processing elements (PEs) or cells that are locally connected. The inherent parallelism of the processing and the local interconnection among PEs are the two main assets of the CNNs. The former makes them suitable for processing images. The latter makes it possible to implement them at hardware level. In its discrete time (DTCNN) version the general dynamic is described by the system of coupled equations and the output-state relationship listed in Eq. 1 and Eq. 2 [2]. Here, x_{ij} represents the state of the processing element ij, y_{kl} and u_{kl} are the outputs and inputs of the PE kl, n marks the temporal step and $N_r(i,j)$ comprises the neighbors of the PE ij within a radio r. Typically, on chip implementations have a 3×3 neighborhood ($r = 1$) but there are applications that need larger neighborhoods. In such applications, either larger area for routing or special techniques are used. Finally, A, A' are the templates that weight the input images (U and Y) and I is the bias term. They determine the operation to be realized.

$$x_{ij}(n+1) = \sum_{k,l \in N_r(i,j)} A_{ijkl} y_{kl}(n) + \sum_{k,l \in N_r(i,j)} A'_{ijkl} u_{kl} + I_{ij} \tag{1}$$

$$y_{ij} = f(x_{ij}) = \begin{cases} -1 & x_{ij} < 0 \\ 1 & x_{ij} > 0 \end{cases} \tag{2}$$

Several *analogic* (analog and logic processing in the array of PEs) solutions have demonstrated CNN architecture benefits, specially in realizing applications with video frame or even higher rates [3]. On the other hand, FPGAs have appeared as affordable and reconfigurable platforms with a very short design cycle, and thus quite adequate for fast prototyping and even mass production, as well for image processing tasks where they allow to exploit paralelism in vision problems, as it is addressed in [4]. Our proposal is to combine both approaches, FPGA and CNN, in a topographic implementation, that is, with a pixel per PE assignment. The result would be a very versatile and robust solution running low level image applications faster than usual software implementations, with the additional benefit of a very short design time and low prize compared to its full-custom counterpart.

In the literature we have found several examples of CNN implementations over FPGAs. There are some cases, as those in [5] and [8], in which the FPGA contains one or only a few PEs. They are general purpose CNN emulations but they hardly take advantage of the inherent CNN parallelism. Thus, with a limited speed, yet faster than usual software implementations over general purpose CPUs. There are some other approaches intended for specific tasks or applications, as those in [9] and [10], achieving more parallelism at the cost of a restricted programmability. In stand-alone systems, typically the images are acquired by a companion camera and stored in a RAM memory, either in or outside the FPGA chip. For processing, more or less exhaustive windowing algorithms, depending on the parallelism implemented, must be used. The pixel value under study and its neighbors are transferred from the RAM to the PEs. The result is read out of the FPGA. In implementations with only one PE, for example, these three steps have to be done as many times as pixels in the image, i.e. in a serial way.

The goal of our approach is to increase the parallelism in general-purpose FPGA implementations. In so doing we improve the processing time as we make windowing less exhaustive or even not necessary in early-vision processing with images of low resolution ($\leq 50 \times 50$). An example of the later is the application addressed in [11]. The key of our proposal to achieve the parallel CNN FPGA implementation is introducing efficient design techniques and methodologies aimed at reducing PEs area consumption, yet meeting time and functionality requirements. In particular we make use of the S&S techniques addressed in [12] that have appear as very efficient in CNN hardware reduction over FPGA implementations [18], as well as weight, state and output range reduction.

The usage of the S&S techniques and other important decisions made in the conception of the PE are analyzed in Section 2. Then, Section 3 addresses the FPGA implementation and Section 4 shows experimental results in functionality, area consumption and processing time. Finally, Section 5 will gather the main conclusions drawn from the paper.

II. Processing Element Design Decisions

Two main requirements have guided the decisions in the design of the PEs: to have full CNN functionality and to have a reduced area occupation. Preserving the full CNN functionality implies that any CNN algorithm could be implemented on it. In order to do that, the implementation has to provide enough local memories to keep intermediate results apart from the hardware necessary to realize any CNN and logic operation. Looking at typical CNN algorithms as can be this in [7], we consider that 4 local memories will be enough at this point. To improve the versatility of the hardware and to achieve better processing times we have considered as well a Fixed-State Map (FSM), that allows to stop the running of selected cells.

Looking for a reduced area we have combined two different lines that trade between area and number of operation. The first line is about shrinking the range of values in Eq. 1 and Eq. 2. The second one is based in using a lower number of multipliers in the PE. Within the former, the first range limitation is to restrict the implementation to binary image processing. In principle, this can be regarded as a serious functionality limitation. Nevertheless, this is not the case as many applications in the realm of early vision make an intensive use of binary tasks after a preliminary processing on the real scene (usually gray-scale processing). Going further, in our PE modeling we use one-quadrant multipliers working on positive values [13]. In this way, the PEs outputs are restricted to either 0 or 1. Besides, the templates are limited to a 1-bit programmability [14], i.e. the coefficients are restricted as well to 0 or 1 value. In our approach, as in those realized in [13] and [15], the bias is chosen 2-bit programmable. The resultant model, the so-called 1Q-1bit-B/W, has proved to yield very dense and fast CMOS chips [15] and it is expected to have good results as well in its FPGA implementation. The counterpart of this choice is that the original templates must be redesign and we have a bigger number of CNN and logic operations. Nevertheless, it is possible to take a good advantage of the FSM in this transformation.

The second step in reducing PE area is based on the number of multipliers that are used in a PE. In a typical CNN there are needed 18 coefficients to implement Eq. 1 for $r = 1$. This means that for every PE we will require 18 multipliers and 16 connections to the surrounding PEs. In order to reduce the area occupied by the PE, the number of multipliers and, with that, the number of connections to neighboring PEs are dropped by means of the Split & Shift techniques that were proposed in [12]. The main idea behind these techniques is to reuse the hardware, executing every CNN operation in several steps as it is shown in Fig. 1. This leads to an increase in processing time. Nevertheless, the actual parallelism introduced by the higher number of cells compensates for this extra time. This is specially true when A and A' are sparse templates through an appropriate S&S configuration (number and allocation of multipliers in a PE). Another important feature is that, as the original templates are executed by the addition of partial results from several sub-templates applied once, this methodology can only be adopted in synchronous CNN architectures.

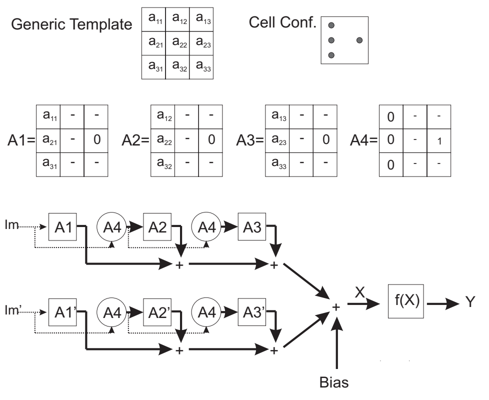


Fig. 1. Application of the S&S techniques to the realization of a CNN operation with two templates (A and A') in the general template shape, over a cell with only 4 multipliers allocated as is shown in the Cell Conf. Squares indicate the application of the subtemplates (A1, A2 and A3) in what the original template is divided. Circles mark the application of shifts implemented by the A4 template.

Among the many different S&S configurations in our topographic CNN realization on an FPGA we have selected the configuration with four multipliers in diamond shape (Fig. 2). Note that, as in a synchronous design there is no difference between variables U and Y, A and A' are interchangeable templates. This allows to gather sequentially the effect of both by re-using the same hardware (multipliers). Configurations with 4 multipliers offer good trade-offs between the hardware reduction (less multipliers) and the higher number of operations (longer processing time) that must be done to emulate a generic template, favoring hardware reduction [12]. Furthermore, the diamond configuration fits the shape of several typical sparse templates, allowing to realize them in 1 or 3 steps instead of 5 [12]. With this configuration the shifts are always done from the original image for a 3×3 template, and from either the original or the shifted image for large neighborhood templates [12], [16]. It is worth noting that this shape reminds of the typical North-East-Weast-South (NEWS) SIMD architecture, revealing the proximity between CNN and SIMD computation [17].

Besides that, we have realized the design in a parametric way allowing the user to decide the maximum size in the templates to be used and the number of multipliers and its

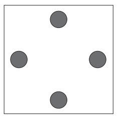


Fig. 2. Configuration with 4 multipliers in diamond shape.

configuration. This is a kind of adjustable trade-off between functionality, processing time and area consumption. Further information about the particular implementation of the hardware, gathering the functionality and area reduction decisions, is shown in next section.

III. FPGA Implementation

The complete system is made up of several elements, which are shown in Fig. 3, that we can group in three major blocks. The first block comprises the grid of processing elements (PEs) that apply the operations over the image. The second one contains the RAM memories and the hardware needed for the communication with the external camera or PC and the computing grid. Finally, the control block supervises the correct application of the templates through its subtemplates and the communications between memories and grid.

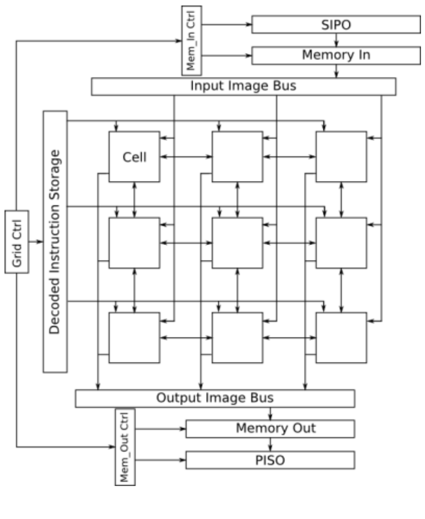


Fig. 3. Schematic showing the three major blocks of the implementation: grid of processing elements, RAM memory and communication hardware, control hardware.

The processing block comprises the PE grid with a pixel per processor correspondence. Furthermore, a ring of dummy cells is employed to deal with border effects. All the PEs are interconnected in a NEWS way according to the cell configuration selected (4 multipliers diamond). Regarding the functionality, the processing block executes Eq. 1 and Eq. 2. A PE comprises the weighting elements, the hardware to accumulate the neighbors contributions and the partial outputs, a comparator to implement the output-state relationship and a group of memories to allow the application of any algorithm. The schematic view of the PEs is shown in Fig. 4.

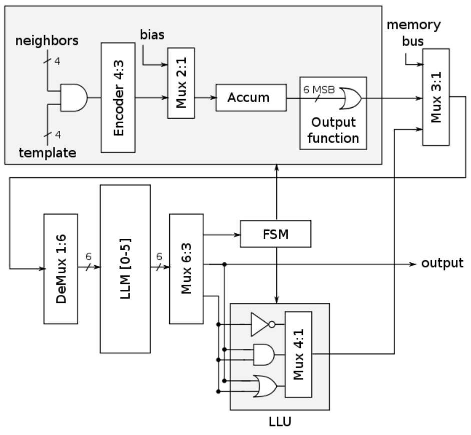


Fig. 4. Processing element schematic showing the DTCNN implementation hardware, FSM, LLU, and the local memory (LLM) and communication elements.

Due to the binary nature of the images to be weighted and the weighting coefficients, our multipliers or weighting elements are implemented in the simplest way, i.e. as AND gates as it is shown in the figure. In the following phase, to add the contributions of the neighbors that have been weighted, again binary, an encoder comes up as an efficient solution in terms of area. Note here that the size (in bits and area occupation) of the encoder depends on the number of multipliers implemented [18]. In applying the S&S techniques we need an accumulator to gather all partial results and the bias term. The size of the accumulator depends on the maximum number and size of the templates we allow in our implementation. In this case we choose 8 bits. With this size we can approach up to two (A and A') 11×11 full dense templates with a two-bit bias. This is more than enough if we take into consideration that the actual large neighborhood templates are typically sparse and that they are normally combined with smaller size templates [11], [6]. It is worthy to note that to avoid negative and non integer values, the bias term has been changed according to Table I. As a consequence the threshold value changes from 1 to 3 and the output-state relationship is realized with an OR gate gathering the six most significant bits of the output of the accumulator. The dummy cells are made up of a simple memory plus needed connections to their neighbors.

TABLE I
Correspondence between bias value in [7] and our implementation.

2-bit Bias [7]	-0.5	-1.5	-2.5	-3.5
New Bias Values	3	2	1	0

In order not to penalize pixel-to-pixel Boolean functions like AND and OR on two images it will be useful to have the central coefficients of both A and A' templates, i.e. two more multipliers. Nevertheless, we have chosen to include a Local Logic Unit (LLU) [19] while keeping the 4 multipliers configuration of Fig. 2. The functionality of the cell is completed with the so-called Fixed-State Map (FSM) [13] and, within the LLU, a NOT gate. The FSM can be applied with any template or logic operation and the image used as fixed map is determined by the programmer. In addition, six Local Logic Memories (LLMs) [19] are needed to store inputs, intermediate and final results of any algorithm and to apply the S&S methodology. These memories are implemented with D-Flip-Flops. Finally, two fixed values (0 and 1) are accessible to compute with them as they were memory values.

To design the memory and communication block we have taken into account a possible robot guiding future application, previously introduced in [11], and the hardware resources we have at our disposal. In this sense we have considered a serial data providing camera M3188A. The emulation of the camera is done through the serial RS232 port. The camera is used in its B/W mode. From the serial feeding restriction we need a Serial Input Parallel Output (SIPO) register to upload the image into a RAM memory. We consider two input memories that upload the input image alternatively to allow the pipeline of the process. We use the image row size as width of the RAM words. For the sake of similarity the output image is saved in an output RAM memory and we use a SIPO register to offer the output image in a serial way. Of course, as we have enough pins we could think in a row/column image download to improve both, the area occupation and the time consumption. These three RAM memories with the same size as the image have been implemented in the Stratix Memory Blocks. In order to simplify the control we have decided to upload the image from the RAM memory to the grid in one step. This implies waiting till having all the image loaded in one of the input RAM memories and upload the image to a bus of the image size to be transferred to the grid. This decision introduces less penalizations in area and time than the row/column scheme but we still need global routing from the memory to each cell in the grid. This can be avoided, penalizing the control, if we upload the image in rows/columns by shifting the data from line to line. The system has been pipelined in order to allow the concurrence of uploading, processing and downloading.

We have divided the control block in four different finite state machines. The first one controls the reception of the image information from the camera and the writing into the input RAM memories. The second state machine controls the writing into the output memory. The third one controls the execution of the image processing within the PEs once the image is inside the grid. Finally, the fourth finite state machine controls the fetch and the identification of the different instructions and supervises the work of the other state machines. The instructions used are 22 bits wide and coding is not worthy. Table II shows the distribution of these 22 bits in the different control signals. The first bit indicates if it is a load instruction, what means upload the image from a RAM memory to the grid. The second group, two bits, says the type of operation (00 for template execution, 01 for logic NOT, 10 for logic AND and 11 for logic OR). The next bit indicates if we have to use the transient mask or fixed state map. The four following bits are the values of the coefficients of the template to be applied and the following two are the value of the bias. The next 3 groups of 3 bits indicate the two local memories to be read and the one to write the output respectively. The second memory takes the fixed state map for template operations and the third one is the fixed state map for logic operations when the fourth bit is 1. The next bit indicates if the template to be applied is a shift template or a sub-template application within the S&S techniques. The last two bits mark the last template of the S&S application and the algorithm respectively.

The entire implementation consists of a grid of 25×25 effective cells over the Altera Stratix-EP1S25F672 FPGA. This image size is enough for some applications as that in [11]. The design has been realized with the Active-HDL 7.1 software by Aldec and synthesized with the Quartus II 6.0 by Altera. We can process larger images by using windowing techniques. The main idea here is to split a large image in windows of the grid size including dummy cells (27×27 in our case). This division has to take into account an overlap among windows equal to the biggest template neighborhood in the algorithm. Also, to avoid errors from border effects, a CNN operation has to be applied over the whole image under processing, i.e. over all the windows, before applying the next one in the algorithm. Clearly, the windowing would slow down the processing, as we have to account for up and download times of each window in the application of every CNN operation. In this case the penalty on the processing time should be analyzed. The S&S methodology, however, would not yield time increases, as all the sub-templates needed to do a complete CNN operation can be run sequentially in a window without any up/downloading. Besides, it should be apparent that a sequential up/downloading based on shifting would be more penalizing and it would appear a new trade-off between the area saved due to the avoided routing and the processing time increment. Apart from that we have to take into account that in this case the intermediate outputs of the algorithm that have to be stored have to be kept in the RAM memories and not in the LLM as it is done in the processing of 25×25 images. This makes it necessary to provide at least 3 RAM memories to hold the different intermediate images. In adition, we have to have as many different banks as the number of rows in the grid in each memory if we want to allow a one step up/download. The LLMs of the cells would be used to apply the S&S techniques. Pipeline techniques would be useful in this part to improve the processing time.

IV. Results and Discussion

Our design has been implemented over the EP1S25F672C Stratix FPGA by Altera. We have implemented a 27×27 (25×25 effective) CNN grid taking up to 19255 of the 25660 logic elements, what represents a 75%. The restriction in the size of the grid is given mainly by the number of logic-arithmetic-blocks (LABs) contained in the FPGA thus we use 2534 of the 2566 (LABs). We have to take into

TABLE II

INSTRUCTIONS AND CONTROL SIGNALS.

Load	**Op**	**Trans. Mask**	**Template**	**Bias**	**WMR1**	**WMR2**	**WMW**	**Temp. Type**	**S&S End**	**Alg. End**
1 bit	2 bit	1 bit	4 bit	2 bit	3 bit	3 bit	3 bit	1 bit	1 bit	1 bit

account, as well, that we have implemented the communication with the computer in order to emulate a serial camera, which implies certain processing time limitations and extra hardware consumption. The memory occupation in the processing of 25×25 images (two input and one output memories) is of less than 1% (2187 bits). There is not usage of DSP blocks, embedded multipliers or PLLs. We can see that there are enough pins to allow parallel image upload up to a size of 25×25 or row/column upload for bigger images. At the moment the control instructions are kept within the internal memory of the FPGA. Nevertheless, given the number of available pins they could be fed from the outside. It is worthy to note that the use of a reduced number of multipliers (4) would allow to use less pins for the template coefficient upload than having all the multipliers.

The maximum working frequency of this implementation is 55.70 MHz. However, in order to give a more general measure we will give the processing time in both clock cycles and seconds, the latter depending on the frequency of the systhesized implementation. The execution of the instructions takes from six to nine clock cycles including the three clock cycles needed for the instruction fetch. Table III shows the number of cycles for the execution of the basic operations that can be realized in the PEs: logic operations (AND, OR, NOT), templates that require only one S&S step to be realized (i.e. those that have at most 4 multipliers in diamond shape) and dense templates that require the maximum number of S&S steps, i.e. three sub-templates applications and two shifts ([16]). The template execution time includes the bias summation and the output function application. A complete CNN operation with two full dense templates would take simply double than the latter. Shifting operations are specially implemented for the S&S techniques and requires the same number of cycles than a logic operation. From these data our system is able to process more than 1400 images/s, applying 1000 full 3×3-template CNN operations to each image. This data could be improved just by pipelining and overlaping the fetch. In this calculation we have not accounted for the time needed to upload the image from the outside as this can be improved just by using fast CMOS cameras as shown in [20]. In this implementation the time required for transferring the image from the RAM memory to the array is $2,7\mu s$.

TABLE III

TIME CONSUMPTION OF BASIC OPERATIONS.

AND/OR/NOT/Shift	$6cycles$	$108ns$
Template (1 step)	$9cycles$	$162ns$
Template (S&S max)	$39cycles$	$700ns$

We can compare these results with the times provided by other implementations. The ACE16K ([21]), that is a common referent in a CNN full-custom implementation, takes $8\mu s$ for binary or gray scale convolutions ([22]). The SCAMP system ([23]) is an SIMD implementation for image processing. This system provides a processing time of $0.8\mu s$ per operation [1]. The digital implementation in [24] takes $0.44\mu s$ for a binary 3×3 kernel application. Also, the FALCON system ([22]) is an implementation over a newer FPGA from Xilinx that requires $0.6\mu s$ per gray-scale convolution. This values are given for a processing of a 128×128 image in gray-scale. The values in Table III are given for our 25×25 implementation and it does not include the $2.7\mu s$ to send the image from the RAM memory to the grid. If we consider applications that require larger images we would have to apply windowing. If we consider no modification in the RAM-grid upload we have to include the $2.7\mu s$ communication time twice per window, $135\mu s$ for a 128×128 image. If we compare our FPGA implementation with the 1Q1bitBW full custom implementation in [25] we can see that the A'-template convolution is more than one order of magnitude worse but the logic operations take only double time. Finally, in a fairer comparison, we consider the FPGA implementation in [5] as it has been implemented over the same hardware we use. In the optimized implementation we have that a 25×25 image can take around $19.23\mu s$[1] per convolution of two 5×5 templates within a CNN operation. If we consider this type of convolution in our system it takes 300 cycles, what is around $5.4\mu s$. Nevertheless, we have to take into account that the great mayority of the CNN operations involved in an image processing algorithm have just 3×3 templates and mainly consider 1 template per CNN operation and they mostly fit in the diamond configuration [**?**].

To complete the test of the implementation we have implemented six algorithms or complex operations. The first five of them have been applied over the 25×25 image of Fig. 5. The first one is the Hole-Filling. In this case we need only a four coefficient template that is applied with a transient mask 13 times for this image size. This amounts to $3.1\mu s$ for the image of Fig. 5. The second propagative operation is the Horizontal CCD. For this we need 8 CNN and 8 logic operations that have to be applied 25 times. This leads to $28.54\mu s$ of processing time. For the Shadow we have applied two operations 25 times, taking $2.16\mu s$. The Hit& Miss looks for the shape in Fig.6 and requires 10 operations ($1.68\mu s$). The Binary Edge Detection (BED) with 4-connectivity requires only 3 operations, this is 540 ns. Finally we have implemented a Shortest Path Problem algorithm based in proposed in [26] over the labyrinth shown

[1]This value is estimated from the cited paper data.

in Fig. 7. The results suggest that this implementation could be useful as an image processing accelerator, especially if we use a newer technology to implement it.

TABLE IV

PROCESS TIME OF ALGORITHMS TESTED

Hole Filling	$155 cycles$	$2,8\mu s$
Horizontal CCD	$1427 cycles$	$25,62\mu s$
SW Shadow	$108 cycles$	$1,94\mu s$
Hit&Miss	$84 cycles$	$1,51\mu s$
BED	$27 cycles$	$485 ns$
Shortest path	$2712 cycles$	$48,69\mu s$

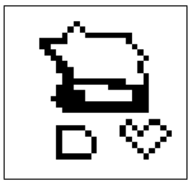


Fig. 5. Test image for the tasks listed on the top five rows of Table IV.

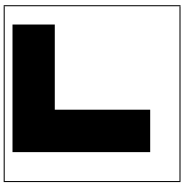


Fig. 6. Hit& Miss structural element.

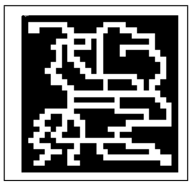


Fig. 7. 1-pixel labyrinth to solve the Shortest Path Problem.

As a final remark, we could improve the computational time if we increase the number of coefficients, as we can improve the area time using 3 coefficients configurations instead of 4.

V. CONCLUSION

We have implemented a 25×25 DTCNN effective grid that demonstrates the viability of fine-grain parallelism with a pixel per processor assignment in images of low resolution (less than 50×50) for real time image processing implementation on an FPGA architecture, in this case on the Stratix-EP1S25 FPGA by Altera. The implementation is limited to binary image processing, which is enough for most of the low level applications and in particular for the robot guide algorithm in [11], one of our pursuits in future work. Also, we have developed a reconfigurable VLSI design that permits test any type of DTCNN multipliers configuration. The migration to other FPGAs family is very easy as we only need to change the stratix memory block. The rest of elements are rescaled only changing size of the grid and size of templates. Further improvements would be achieve if we choose to optimize a parameter in particular, giving priority to either time or area and depending on the application.

REFERENCES

[1] L.O. Chua et al., "The CNN Paradigm," *IEEE Transactions on Circuits and Systems*, vol. 40, pp. 147-156, 1993.

[2] H. Harrer and J. A. Nossek, "Multilayer Discrete-Time Cellular Neural Networks Using Time-Variant Template," *IEEE Transactions on Circuits and Systems II:Express Briefs*, vol. 40, n. 3, pp. 191-199, 1993.

[3] A. Zarándy and Cs. Rekeczky, "Bi-i: A Standalone Ultra High Speed Cellular Vision System," *IEEE Circuits and Systems Magazine*, vol. 5, n. 2, pp. 36-45, 2005.

[4] W. James MacLean, "An Evaluation of the Suitability of FPGAs for Embedded Vision Systems," in *CVPR'05*, San Diego, USA, 2005, pp. 29–34.

[5] G. Pazienza et al., "Optimized Cellular Neural Network Universal Machine Emulation on FPGA," in *European Conference on Circuit Theory and Design 2007, ECCTD 2007*, Seville, Spain, 2007 pp. 815-818.

[6] Vilarino D.L. et al., "Discrete-time CNN for image segmentation by active contours" *Pattern Recognition Letters*, vol. 19, pp. 721-734, 1998.

[7] Victor M. Brea et al., "A binary-based on-chip CNN solution for pixel-level snakes," *Int. J. Circuit Theory Appl.*, vol. 34, pp. 383–407, 2006.

[8] Nagy, Z. and Szolgay, P.,"Configurable Multilayer CNN-UM Emulator on FPGA" *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 50, n. 6, pp. 774-778, 2003.

[9] Lopich, A. and Dudek, P., "Architecture of Asynchronous Cellular Processor Array for Image Skeletonization," in *ECCTD'05*, Cork, Ireland, 2005, pp. III/81 - III/84

[10] C. Baldanza et al., "A Cellular Neural Network for Peak Finding in High-Energy Physics," in *CNNA'2000*, Catania, Italy, 2000, pp. 443-448

[11] Xavier Vilasis-Cardona, et al., "Guiding a Mobile Robot with Cellular Neural Networks," in *International Journal of Circuit Theory and Applications*, Vol. 30, n. 6, pp. 611-624, 2002

[12] N.A. Fernández-García et al., "On the Reduction of the Number of Coefficient Circuits in a DTCNN Cell," in *CNNA2006*, Istanbul, 2006, pp. 29–34.

[13] Victor M. Brea et al., "On the One-Quadrant Template Design in a High Gain CNN Model," in *CNNA2004*, Budapest, Hungary, 2004, pp. 279-284

[14] A. Paasio et al., "High Density VLSI Implementation of a Bipolar CNN with Reduced Programmability," in *ISCAS '04*, Vancouver, Canada, 2004, vol. 3, pp. 21-24

[15] Jacek Flak, et al., "VLSI Implementation of a Binary CNN: First Measurements Results," in *CNNA2004*, Budapest, Hungary 2004, pp. 129-134

[16] N.A. Fernández-García et al., "Hardware Simplification in Cellular Non-Linear Networks for Complex Algorithms," in *DCIS06*, Barcelona, Spain, 2006.

[17] Victor M. Brea et al., "Relating Cellular Non-linear Networks to Thershold Logic and Single Instruction Multiple Data Computing Models," in *ECCTD'07*, Seville, Spain, 2007.

[18] N.A. Fernández-García et al., "Split&Shift Techniques for CNN Hardware Reduction: First Measurements," in *ECCTD'07*, Seville, Spain, 2007.

[19] Tamás Roska and Leon O. Chua, "The CNN Universal Machine: An Analogic Array Computer" *IEEE Transactions on Circuits and Systems-II: Express Briefs*, vol. 40, n. 3, pp. 163-173, 1993.

[20] www.Fast-Vision.com

[21] A. Rodríguez-Vázquez et al., "ACE16K: The third generation of mixed-signal SIMD-CNN ACE chips towards VSoCs," *IEEE Trans. Circuits Syst. I-Regul. Pap.*, vol. 51, no. 5, pp. 851–863, 2004.

[22] Zoltán Nagy et al., "Emualted digital CNN-UM solution of partial differential equations," *International Journal of Circuit Theory and Applications*, Vol. 34, pp. 445-470, 2006

[23] Piotr Dudek, "Implementation of SIMD Vision Chip with 128 × 128 Array of Analog Processing Elements," in *ISCAS '05*, Kobe, Japan, 2005, vol. 6, pp. 5806-5809

[24] Péter Foldesy et al., "Digital Implementation of Cellular Sensor-Computers," *International Journal of Circuit Theory and Applications*, Vol. 34, pp. 409-428, 2006

[25] Jacek Flak, et al., "Dense CMOS Implementation of a Binary Programmable CNN," in *International Journal of Circuit Theory and Applications*, Vol. 34, pp. 429-443, 2006

[26] Cs. Rekeczky, "Skeletonization and the Shortest Path Problem - Theoretical Investigation and Algorithm for CNN Universal Chips," *NOLTA99*, 1999

Single Instruction Multiple Data and Cellular Non-linear Networks as Fine-Grained Parallel Solutions for Early Vision on FPGAs

A. Nieto*, N.A. Fernández-García*, J. Albo-Canals†,
V.M. Brea*, D.L. Vilariño*, J. Riera-Babures† and Diego Cabello-Ferrer*
* Department of Electronics and Computer Science, University of Santiago de Compostela
E-15782 Santiago de Compostela, Spain. Email: alejandro.nieto@usc.es
† Departament d'Electrónica, Enginyeria i Arquitectura La Salle, Universitat Ramon Llull
E-08022 Barcelona, Spain. E-mail:jalbo@salle.url.edu

***Abstract*—This paper examines the feasibility of fine-grained parallelism on FPGAs for early vision. The paper compares the performance and functionality of Cellular Non-linear Networks and Single Instruction Multiple Data architectures on FPGAs. Area and speed data along with different examples of low-level image processing tasks are discussed throughout the paper.**

I. Introduction

The ever larger availability of resources on FPGAs as well as their rapid prototyping and high degree of reconfigurability make designers of hardware for early vision turn an eye to FPGA-based systems [1]. Still, FPGA implementations fail to deliver fine-grained parallelism in low-level image processing, lagging far behind their IC custom counterpart. State-of-the-art vision chips like Eye-Iris follow a pixel-per-processor assignment, achieving a higher degree of parallelism [2]. The work in this paper explores the feasibility of fine-grained parallelism with pixel-per-processor assignment on existing FPGAs.

The paper goes through two well-known architectures, namely Cellular Non-linear Networks (CNN) [3] and Single Instruction Multiple Data (SIMD) [4]. CNNs are a parallel computing paradigm similar to neural networks, with the difference that communication is allowed between neighbouring units only. The dynamical behavior of CNN processors can be expressed mathematically as a series of ordinary differential equations, where each equation represents the state of an individual processing unit. SIMD is a conception that is able to run mathematical operations on multiple data elements simultaneously, as in a vector processor. This is in contrast to a scalar processor which handles one element at a time. The former have found their main niche of applications in computer vision. The latter has been around in high-performance computing for many years. Today it continues being a field of interest in vision chips [5]. Concerning details of both implementations, it is worth pointing out that the connectivity of every cell/processing element in a CNN suits very well the convolution pattern of image processing. In a classical CNN implementation, cell inputs and outputs of the 9 nearest neighbors (including the cell under study) are weighed and collected at the cell under study. This is performed by 18 multipliers, or more generally coefficient or weighing circuits and a summing circuit in every cell. The information among cells is exchanged through the coefficient circuits. In the SIMD conception, every processing element (PE) is locally connected to the four nearest neighbors through the main four cardinal directions. Every PE contains an ALU and the connectivity among neighbors is done through the so-called North-East-West-South (NEWS) bus system.

It is apparent that there is a significant difference in the amount of resources used for every architecture. In principle, the greater number of connections of a CNN cell with its neighbors leads naturally to faster processing than the SIMD architecture, although at the cost of less density of integration, and thus parallelism.

The density of integration becomes important when designing FPGA-based systems. This paper explores the feasibility of fine-grained parallelism on FPGAs for low-level image processing with SIMD and CNN architectures. The paper is devoted to B/W image processing. This can be regarded as a serious limitation of this work. Nevertheless, many algorithms and applications in the realm of early vision make an intensive use of binary image processing, on occasions being even more time-consuming than the gray-scale processing itself [6]. Also, the ever larger density and higher clock frequencies of FPGA platforms make ever feasible a design with gray-scale and binary image processing on separated circuits, either on a single or multiple FPGA chips. The work presented here deals with the implementation of SIMD and CNN arrays of low to moderate size (less than 100×100 cells/PEs) on a single chip, the Altera Stratix-EP1S25F672 FPGA.

The paper is organized as follows. Section II addresses the PE implemented on the SIMD architecture. Section III goes through the design of the CNN cell. Section IV compares the performance and functionality of both solutions when dealing with typical low-level image processing tasks. Finally, Section V gathers the main conclusions drawn from the paper with a brief outlook.

II. The SIMD Processing Element

The design of the processing element on the SIMD architecture is focused on area, aiming at the smallest possible processing element, in order to get the largest possible array. With this, the resultant processing element is quite simple. Fig. 1 displays a detailed view of the PE. It has three main components: a logic unit, a memory bank and a module for local connectivity among neighbors within the array through the classical NEWS system. A global control is proposed with the aim of reducing the area of each PE. This control unit is implemented with only a few logic gates using an adequate instruction set.

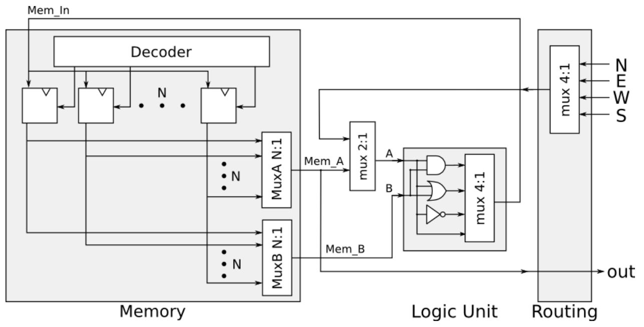


Figure 1. Details of the Processing Element for the SIMD solution

The logic unit performs the most basic Boolean functions: AND, OR, NOT and Identity. This set of operators makes up a functionally complete logic, so with enough time and memory we can implement any algorithm for B/W image processing. The logic unit takes two inputs, namely A and B, and gives one output, R. The operand B drives only the AND and OR gates. The operand A feeds the four Boolean operators. The operand B comes from memory while the operand A comes from either memory or the neighborhood (see Fig. 1). The identity operator can be used to transfer data among memory elements within the cell under study, or to save a neighbor variable into the local memory. Also, the identity operator can do synchronous shifts through the NEWS system of either columns or rows in the array. The function to be done by the logic unit is selected through a 4:1 multiplexer controlled with a 2 bits signal. In addition, it would be easy to include new operators in the logic unit to meet the time needs of specific applications or algorithms, although at the expense of area.

The memory bank comprises a configurable set of one bit registers. The memory bank takes a one bit word as input, labeled Mem_In, and provides two simultaneous outputs, labeled Mem_A and Mem_B, of one bit each. This configuration permits to handle both internal operations and data exchange among neighbors. The identity operator in the logic unit is needed for the latter. In addition, it is doable to read and write simultaneously at the same memory address. This reduces the number of storage elements needed for a generic algorithm, as it is feasible to do operations of the type $Register(0) = Register(0)\ AND\ Register(1)$. Finally, the control logic for the memory bank includes an $S : N$ decoder, being N the number of storage elements and S complying with $2^S = N$, to select the write address, and an additional 1 bit enable signal to state whether or not the memory is written. The read address is selected by means of two N:1 independent multiplexers controlled by two signals of S-bits wide each. This yields signals Mem_A and $MemB$. The number of control bits is set by the memory size, in this case $3S + 1$. Finally, it should be noted that the memory bank is one of the most critical modules in the processing element. The number of storage elements should be reduced as much as possible because it occupies most of the area in the processing element and the performance of the implementation improves when the global lines and the fan-out decrease.

The connectivity among processing elements is set through the NEWS system. Every processing element counts on four inputs and one output. A 4:1 multiplexer decides which one of the four neighbors is used for processing at the processing element under study. Inner connections draw this value to the appropriate module. The instruction determines what to do with the output from every processing element.

Concerning the connectivity within the processing elements, all the modules described above are directly connected to each other, with the exception of a 2:1 multiplexer to select where the operand A comes from, either the memory bank or a neighbor (see Fig. 1). The internal buses in the processing element provide the following functionality:

- Select neighbor
- Select two memory values
- Select between two operation modes
  1) Mode 1- with two operands, either with two memory values or with a memory value and a neighbor (AND/OR)
  2) Mode 2- with one operand, either from the memory or from a neighbor (NOT/Identity)
- Write the operation result into the memory bank
- Provide the processing element output

In our implementation, we choose a memory size of 8 bits (N=8 and S=3), with a instruction set of 15 bits wide. For a more detailed explanation, readers are addressed to [7].

III. The CNN Cell

As in the case of the SIMD processing element, the area is the main parameter to optimize in the design of the CNN cell. To this end we use one-quadrant coefficient circuits working on positive values [8]. In this way, the cells/PEs outputs are restricted to either 0 or 1 instead of -1 and 1 as in the case of the classical CNN model. Also, the templates are 1-bit programmable [9]. The resultant model, the so-called 1Q-1bit-B/W, has proved to yield very dense and fast CMOS chips [10]. In our approach, as in the 1Q-1bit-B/W implementations realized in [8] and [10] the bias is 2-bit programmable.

The second step to save area looks at the coefficient circuits. In a typical CNN there are needed 18 coefficients and 16 connections to the surrounding PEs. In our approach, the number of coefficient circuits and, with that, the number of connections to neighboring PEs are dropped by means of the Split & Shift techniques [11]. The main idea behind these techniques is to reuse the hardware, executing every CNN operation in several steps. This leads to an increase in processing time. Nevertheless, the actual parallelism introduced by the higher number of cells compensates for this extra time. This is specially true when the CNN templates are sparse and when the S&S configuration (number and allocation of coefficient circuits (cc) in the cell) suits the allocation and number of non-zero entries of the templates. Another important feature is that, as the original templates are executed by the addition of partial results from several sub-templates applied once, this methodology can only be adopted in synchronous CNN architectures.

Among the many different S&S configurations in our pixel-per-processor CNN realization on an FPGA we have selected the configuration with four cc in diamond shape (Fig. 2). Such a configuration offers good trade-offs between the hardware reduction (less coefficient circuits) and the higher number of operations (longer processing time) that must be done to emulate a generic template [11]. Furthermore, the diamond configuration fits the shape of several typical sparse templates, allowing to realize them in few steps (1 to 3). With this configuration the shifts are always done from the original image for a 3×3 template, and from either the original or the shifted image for large neighborhood templates [11], [12]. It is also worth noting that this shape reminds of the typical North-East-Weast-South SIMD architecture, revealing the proximity between CNN and SIMD computation [13].

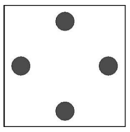


Figure 2. Configuration with four cc in diamond shape.

Fig. 3 displays the schematic view of the CNN cell. Due to the binary nature of the images to be weighed and the weighing coefficients, our coefficient circuits are just AND gates. The contributions of the neighbors that have been weighed, again binary, are collected by means of an encoder. Its size depends on the number of coefficient circuits implemented [14]. In applying the S&S techniques we need an accumulator to gather all partial results and the bias term. The size of the accumulator depends on the maximum number and size of the templates we allow in our implementation. In this case we choose 8 bits. With this size we can approach up to two (*A* and *B*) 11×11 full dense templates with a two-bit bias. This is more than enough if we take into consideration that the actual large neighborhood templates are typically sparse and that they are normally combined with smaller size templates. It is worthy to note that to avoid negative and not integer values, the bias term has been changed according to Table I. This implies to change the threshold value from 1 to 3. As a consequence, the output-state relationship is done with an OR gate gathering the six more significant bits of the output of the accumulator.

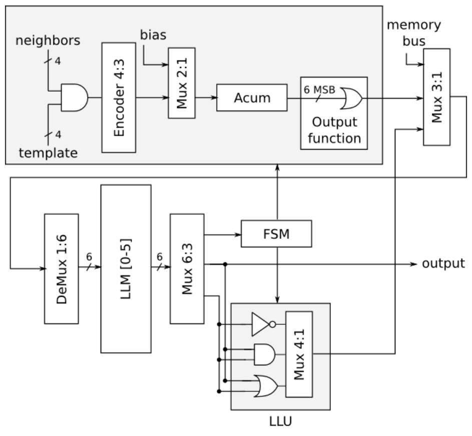


Figure 3. Details of the CNN cell.

In order not to penalize pixel-to-pixel Boolean functions like AND and OR on two images it will be useful to have the central coefficients of both *A* and *B* templates, i.e. two more coefficient circuits. Nevertheless, we can see [14] that a more efficient solution than to include the central coefficient is to include a Local Logic Unit (LLU) [15] while keeping the 4 cc configuration of Fig. 2. The functionality of the cell is completed with the so-called Fixed-State Map [8] and, within the LLU, a NOT gate. The FSM can be applied with any template or logic operation and the image used as map is determined by the programmer. In addition, six Local Logic Memories (LLMs) [15] are needed to store inputs, intermediate and final results of any algorithm and to apply the S&S methodology. These memories are implemented with D-Flip-Flops. In [17] you can find a more exhaustive description of the CNN cell.

Table I
CORRESPONDENCE BETWEEN BIAS VALUE IN [16] AND OUR IMPLEMENTATION.

2-bit Bias [16]	-0.5	-1.5	-2.5	-3.5
New Bias Values	3	2	1	0

IV. SIMD AND CNN ARRAYS ON THE FPGA

This Section examines the feasibility of fine-grained parallelism on FPGA-based systems. In so doing, we go through typical image processing tasks usually found in algorithms and applications.

A. FPGA Description

The SIMD and CNN architectures have been implemented on the Altera EP1S25F672C Stratix FPGA. This FPGA is embedded on a card that provides a serial interface to the PC. Obviously, this is not the best solution for a real-life application. Nevertheless, it gives a deeper insight into the

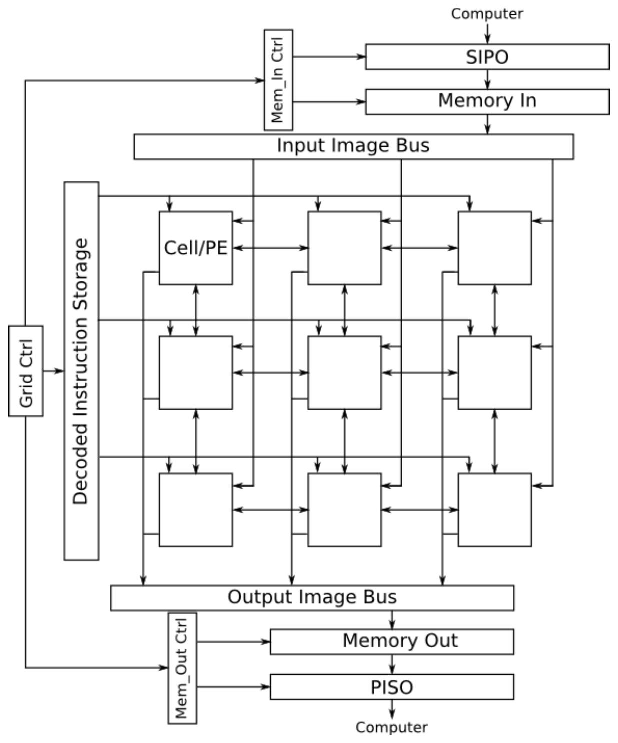


Figure 4. Floorplan of the SIMD and CNN architectures on the FPGA.

functionality and performance of SIMD and CNN architectures with pixel-per-processor assignment on FPGA-based systems. Both architectures have been developed in VHDL to obtain the highest possible performance and to be able to control at low-level the hardware to be implemented on the FPGA. The software used to design and test was the Active-HDL 7.1 provided by Aldec. The Quartus II 6.0 by Altera was used for the synthesis.

B. SIMD and CNN Arrays

The FPGA implementation is made up of three major blocks. Fig. 4 displays its schematic view. The processing block comprises the PE grid, here 25×25 PEs/cells. With the selected connectivity, both the CNN cell and the PE of the SIMD are equivalent in terms of integration into the mesh. Obviously they differ in the internal logic of the control elements. The memory block contains RAM memories and the hardware required for communications with the outside and the computing grid. There is also a block to control the execution of instructions and the communications between the memories and the grid. In our case the images upload onto the inner RAM memories of the FPGA chip is serielized from the outside. The images are transferred in parallel from the inner RAM to the grid. The FPGA floorplan is completed with a ring of dummy PEs/cells (for the sake of clarity not shown in Fig. 4) and shift registers to synchronize the communications from/to the input/output FPGA pins.

C. Synthesized Data

The resources employed by both architectures are summarized in Table II. The SIMD architecture consumes less resources than the CNN. In both cases, however, the area is small enough to fit images of low to moderate resolution (32×32 to 64×64).

	SIMD	**CNN**
Frequency (MHz)	103.0	55.70
Total logic elements	16.307 (64%)	19.255 (75%)
Total memory bits	1.458 (<1%)	2.187 (<1%)

Table II
SUMMARY OF BOTH ARCHITECTURES WITH A $25{\times}25$ ARRAY.

D. Examples

In order to illustrate the functionality and assess the differences and similiarities between the SIMD and CNN architectures discussed here, we go through a series of B/W image processing tasks. The tasks addressed here are well-known and very representative of the type of image processing done in real-life applications. We analyze hit and miss (pattern matching finder), propagative, large-neighborhood operations and a relatively complex algorithm of B/W skeletonization as an example of a morphological task. First we briefly describe every task. Subsequently we tell how to approach every example with the SIMD and CNN architectures along with their corresponding performance data.

1) Pattern Matching Finder: As its name suggests, this template finds given patterns. As an example, we will find the pattern shown in Fig. 5. The symbol ”-” means does not matter. The output is a binary image representing the locations of the 3×3 pattern of Fig. 5. In a CNN this task can be done with the template of Eq. (1), where black pixels have set to +1 and white pixels to -1.

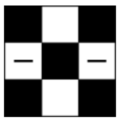


Figure 5. Pattern Matching Finder example.

$$T = \begin{pmatrix} 1 & -1 & 1 \\ 0 & 1 & 0 \\ 1 & -1 & 1 \end{pmatrix}, I = -6.5 \qquad (1)$$

2) B/W Skeletonization: This algorithm finds the skeleton of a black and white object. Fig. 6 displays the flow diagram of such an algorithm. In this case we use the templates listed in [18]. Eq. (2) and (3) are the first two templates of the algorithm. The rest of the templates are rotated versions of Eq. (2) (SkelBW3, SkelBW5, SkelBW7) and Eq. (3) (SkelBW4, SkelBW6, SkelBW8), where again black pixels have been assigned to +1 and white pixels to -1.

$$SkelBW1 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}, I = -1 \quad (2)$$

$$SkelBW2 = \begin{pmatrix} 2 & 2 & 2 \\ 0 & 9 & 0 \\ -1 & -2 & -1 \end{pmatrix}, I = -2 \quad (3)$$

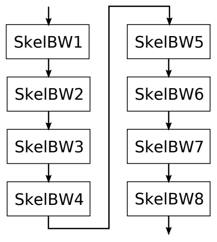


Figure 6. Flow diagram of the Skeletonization algorithm.

3) Hole Filling: This is a very representative CNN propagating template [18]. As its name suggests this template fills the gaps of all the objects found on an image. In a synchronous architecture, this template is run iteratively for n iterations, being n an upper bound set a priori and dependent on the image resolution.

4) Large-Neighborhood Template: As an example of large-neighborhood template we have realized a 5×5 line detector similar to the *LE3pixelLineDetector* found in [18]. The template addressed here deletes lines with more than three pixels in a row along the horizontal, vertical and the two diagonal directions, keeping only the lines with less than or equal to three pixels. The template works on binary images, providing a B/W output image too.

5) SIMD Approach: The pattern matching finder can be translated easily onto the SIMD architecture. The current pixel will be active if the condition $OUT = nw.\bar{n}.ne.c.sw.\bar{s}.e$ is true (the variables c, n, nw, etc. refer to central, north, north-west pixels). For its implementation we have to keep in mind that the Logic Unit of each PE can only handle 2 inputs and the access to the neighbors at the corners must be made using shifts. The following pseudo-code outlines all the steps:

- R0 $\leftarrow$ *initial image*
- R1 $\leftarrow$ *NOT* north(R0) = $\bar{n}$
- R1 $\leftarrow$ R0 *AND* R1 = $c.\bar{n}$
- R2 $\leftarrow$ *AND* south(R0) = *s*
- R1 $\leftarrow$ R1 *AND* R2 = $c.\bar{n}.\bar{s}$
- R2 $\leftarrow$ *shift*(right)
- R1 $\leftarrow$ R1 *AND* north(R2) = $c.\bar{n}.\bar{s}.nw$
- R1 $\leftarrow$ R1 *AND* south(R2) = $c.\bar{n}.\bar{s}.nw.sw$
- R2 $\leftarrow$ *shift*(left)
- R1 $\leftarrow$ R1 *AND* north(R2) = $c.\bar{n}.\bar{s}.nw.sw.ne$
- R1 $\leftarrow$ R1 *AND* south(R2) = $c.\bar{n}.\bar{s}.nw.sw.ne.e$

In terms of notation *neighbor(Register#)* like *north(R2)* means that we have to access to the register #2 of the north neighbor to operate. In this example, once the image is loaded on the array we need 10 cycles (each operation takes one cycle) and 2 registers. One of them is used to store the final result. It should be noted that to access a pixel that is not directly connected to the pixel of interest, it is only necessary to perform shifts until the value of the pixel shifted reaches one of the four neighbors and not even the position of the pixel under consideration. This allows us to increase largely the performance.

The approach of the B/W skeletonization on the SIMD architecture is not straightforward and it is necessary to perform an analysis of the cases in which the active pixel changes state. The templates of Eq. (2) and Eq. (3) can be approached as follows:

$$\begin{aligned} SkelBW1 &= \overline{\overline{nw}.\bar{n}.\bar{w}.e.\bar{s}}.c \\ &= \overline{(nw+n+w).e.s}.c \end{aligned} \quad (4)$$

$$\begin{aligned} SkelBW2 &= \overline{\overline{nw}.\bar{n}.\overline{ne}.s.(sw+se)}.c \\ &= \overline{(nw+n+ne).s.(sw+se)}.c \end{aligned} \quad (5)$$

In this case, the number of cycles required to compute these two templates are 9 for *SkelBW1* and 12 for S*kelBW2*. After loading the image, the number of registers used are 2 and 3 respectively. One of them stores the result. For the whole algorithm, we need 84 cycles and only 3 registers.

As it was mentioned above, the hole filling is a very representative CNN propagating template. The approach followed is similar to that of the Pattern Matching Finder. This template can be done by the condition $OUT = n+e+w+s$. In addition, it is necessary to perform some additional operations. First, invert the original image. Second, apply the template. Then, invert the resultant image and do an OR between this result and the original image. Once the image is loaded a hole filling iteration takes 7 clock cycles and needs 2 registers, including one to store the result. For the image processed (25×25) we have set 13 as number of iterations, leading to 91 clock cycles.

Finally, the large neighborhood template of 5×5 for detecting lines of less than or equal to three pixels along the main six directions (two diagonals, horizontal and vertical) is executed in 59 cycles on the SIMD architecture. It needs only 3 registers.

6) CNN Approach: The simplicity of the 1Q-bit CNN model usually splits one template into several ones. This is because the different conditions that turn a black (white) pixel into white (black) cannot be gathered in only one template. Additionally, the S&S methodology necessary leads to more steps. In order to illustrate our procedure we describe thoroughly how to approach the pattern matching finder realized with the template of Eq. (1). The templates for this task are listed in equations (6) to (8). The templates of Eq. (6) and Eq. (7) act on the original image. The template of Eq. (8) is run on the inverted image. Finally, the conditions assessed by the three templates are checked out with an AND gate. All in all, the example of pattern matching finder analysed here needs 33 cycles.

$$T1 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, I_1 = -2.5 \quad (6)$$

$$T2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix}, I_2 = -1.5 \quad (7)$$

$$T3 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, I_3 = -1.5 \quad (8)$$

Following a similar procedure to what has been used for the example of the pattern matching finder, the hole filling amounts to 87 cycles on a 25×25 image, setting 13 as the number of iterations.

In the case of the large-neighborhood template, when using the 1Q-1bit CNN model, the lines along the horizontal direction are detected by applying the two templates of Eq. (9) and (10). These templates need to be rotated appropriately for every direction (two diagonals, horizontal and vertical).

$$T1 = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 \end{pmatrix}, I = -3.5 \quad (9)$$

$$T2 = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \end{pmatrix}, I = -3.5 \quad (10)$$

Altogether, the large-neighborhood template requires 243 cycles.

E. Discussion

Table III shows the time needed to implement the proposed examples.

	SIMD (cycles / μs)	CNN (cycles / μs)
Pattern Matching Finder	10 / 0.097	33 / 0.592
Skel. (1 whole iteration)	84 / 0.816	432 / 7.76
Hole Filling (13 iterations)	91 / 0.883	87 / 1.56
Large-Neigh. Template	59 / 0.573	243 / 4.63

Table III
TIME IN μS TO EXECUTE THE PROPOSED EXAMPLES IN BOTH ARCHITECTURES WITH AN 25×25 ARRAY. FREQUENCY: A) SIMD AT 103.0MHZ B) CNN AT 55.70MHZ.

In view of the above results, one can conclude that the SIMD conception leads to more competitive data on both area and time consumption. Nevertheless, the FPGA CNN implementation features too many finite state machines, that can yield more innefficient solutions. In the short-term future this is an issue to be addressed. The difference in frequencies is due to the aggressive segmentation of operations under the SIMD architecture. As each instruction is very simple, the critical path is very short, obtaining a high frequency. The area difference is explained taking into account the simplicity of the PE. The CNN cell integrates, in addition to the logic for CNN operations, additional logic for Boolean operations. The main advantage of the SIMD proposal is performance. However, the translation of CNN templates into Boolean equations requires further work by the designer of the algorithm. In some cases, especially for large neighbourhood, to explore the solutions space may be too hard. It is in these cases where the CNN proposal stands out, providing a better response at design time and also in performance.

V. CONCLUSION

This paper has shown that fine-grained parallelism with pixel-per-processor assignment on FPGA-based systems is feasible for low-level image processing with binary images of low resolution (less than 50×50). This work has proved on the FPGA Altera EP1S25F672C Stratix for images of 25×25 pixels. It is possible to process larger arrays on more modern FPGA models. Also, this work shows that the SIMD conception clearly outperforms the CNN model, leading to more competitive area and time consumptions. The FPGA CNN implementation presented here, however, features a high number of finite state machines. In the short-term future, this issue will be improved. This will narrow the gap between CNN and SIMD architectures for B/W processing. Still, it is likely that the SIMD architecture continues outperforming the CNN model.

REFERENCES

[1] W. James MacLean, *An Evaluation of the Suitability of FPGAs for Embedded Vision Systems*, Proceedings of the 2005 IEEE Computer Society Conference on Computer Vison and Pattern Recognition (CVPR'05), vol. 34, pp. 445–470, 2005.

[2] Anafocus, *http://www.anafocus.com*.

[3] L.O. Chua, T. Roska, *Cellular Neural Networks and Vision Computing. Foundation and Applications*, Cambridge University Press: Cambridge, 2002.

[4] Martin C. Herbordt et al., *A System for Evaluating Performance and Cost of SIMD Array Designs*, Journal of Parallel and Distributed Computing, vol. 60, pp- 217–246, 2000.

[5] Alireza Moini, *Vision Chips*, Kluwer Academic Publishers, 2000.

[6] D.L. Vilariño, D.Cabello, X. M. Pardo, V.M. Brea, *Cellular Neural Networks and Active Contours: A Tool for Image Segmentation*, Image and Vision Computing, vol. 21, n. 2, pp- 189–204, 2003.

[7] A. Nieto, V.M. Brea, D.L. Vilariño, *SIMD Array on FPGA for B/W Image Processing, CNNA2008*, Santiago de Compostela, Spain, 2008.

[8] V.M. Brea, M. Laiho, D.L. Vilariño, A. Paasio, D. Cabello, *On the One-Quadrant Template Design in a High Gain CNN Model*, in *CNNA2004*, Budapest, Hungary 2004, pp. 279-284

[9] A. Paasio et al., *High Density VLSI Implementation of a Bipolar CNN with Reduced Programmability*, in *ISCAS '04*, Vancouver, Canada, 2004, vol. 3, pp. 21-24

[10] Jacek Flak, et al., *VLSI Implementation of a Binary CNN: First Measurements Results*, in *CNNA2004*, Budapest, Hungary 2004, pp. 129-134

[11] N.A. Fernández et al., *On the Reduction of the Number of Coefficient Circuits in a DTCNN Cell*, in *CNNA2006*, Istanbul, 2006, pp. 29–34.

[12] N.A. Fernández *et al.*, *Hardware Simplification in Cellular Non-Linear Networks for Complex Algorithms*, in *DCIS06*, Barcelona, Spain, 2006.

[13] V.M. Brea, M. Laiho, N.A. Fernández, A. Paasio, D. Cabello, *Relating Cellular Non-linear Networks to Thershold Logic and Single Instruction Multiple Data Computing Models*, in *ECCTD'07*, Sevilla, Spain, 2007.

[14] N.A. Fernandez-Garcia, J. Albo-Canals, V.M. Brea, J. Riera-Babures, D. Cabello, X. Vilasis-Cardona, *Verification of Split&Shift Techniques for CNN Hardware Reduction*, in *ECCTD'07*, Sevilla, Spain, 2007.

[15] Tamás Roska and Leon O. Chua, *The CNN Universal Machine: An Analogic Array Computer, IEEE Transactions on Circuits and Systems-II: Express Briefs*, vol. 40, n. 3, pp. 163-173, 1993.

[16] V.M. Brea et al., *A binary-based on-chip CNN solution for pixel-level snakes, Int. J. Circuit Theory Appl.*, vol. 34, pp. 383–407, 2006.

[17] N.A. Fernández et al., *Discrete time non-linear cellular networks over FPGA implementation*, in *DCIS08*, Grenoble, France, 2008.

[18] CSW, *Cellular Wave Computing Library (Templates, Algorithms, AND Programs), Version 2.1, CSW-1-2007*, Budapest 2007.

Relating Cellular Non-linear Networks to Threshold Logic and Single Instruction Multiple Data Computing Models

V.M. Brea*, M. Laiho§, N.A. Fernández*, A. Paasio§, and D. Cabello*
*Departamento de Electrónica e Computación, Universidade de Santiago de Compostela
E-15782 Santiago de Compostela, Spain. E-mail:victor@dec.usc.es
§Microelectronics Laboratory, University of Turku, Lemminkaisenkatu 14-18
20540 Turku, Finland

Abstract— **This paper examines three apparently different computing models, namely, Threshold Logic (TL), Cellular Non-linear Networks (CNN) and Single Instruction Multiple Data (SIMD). TL is an area of interest in modern VLSI design and computational neuroscience. CNNs are mainly employed in image processing. Conventional SIMD architectures aim at exploiting data parallelism to speed up the execution time of computation intensive algorithms. The scope of this paper is limited to the processing of binary images. Within this scope, the paper conveys three main conclusions. First, the three computing models can be used for binary image processing. Second, not only 2D-CNNs are a sub-class of SIMD architectures, but also synchronous 2D-CNNs with a reduced set of coefficient circuits act as a classical 1-bit SIMD processing element with NEWS (North-East-West-South) for nearest-neighbor communications. Third, TL gates (TLGs) are proved to be an alternative to implement binary 2D-CNNs, leading to on-chip solutions with a very high performance.**

I. Introduction

In the realm of early vision, many algorithms make an intensive use of binary image processing. This is the case of surveillance or tracking applications based on edge or region detection, in which, after a preliminary processing on the real scene to filter out noise and to provide some kind of motion vector, the algorithms work on binary images [1], [2], [3]. The large amount of data obliges to run such algorithms on hardware provided with a certain degree of parallelism in order to comply with video frame rate or even harder time constraints. On many occasions, in an attempt to design the system as programmable and general purpose as possible, the hardware is meant to work on gray-scale images, dealing with binary images as a particular case of gray-scale processing [4]. In so doing, although the hardware usually meets the video frame rate constraint, there is still room to improve the performance and go further.

Today, with the 45 nm technology node around the corner [5], and with the advent of nanotechnologies is feasible to have hardware with more and more level of parallelism and specialization. In this scenario, architectures with modules for gray-scale and binary image processing separated might be an alternative to solutions with circuitry for gray-scale processing only. Eventually, the level of specialization might increase, as is the case of the architecture presented in [6], where morphological and arithmetic operations are executed in different modules. Emerging reconfigurable architectures like Field Programmable Object Arrays (FPOAs) point to the same direction, containing modules for specific functions, instead of circuits for general purpose applications [7].

This work is focused on hardware for binary image processing. Our interest is limited to on-chip solutions with fine-grained parallelism. Approaches with coarse-grained parallelism like general purpose processors or even FPGAs are not addressed here. In this context, classical SIMD and CNN are viewed as very suitable computing models [8], [9]. The former are usually implemented as synchronous digital chips [10], even though some analog approaches exist [11]. The latter were initially conceived as an asynchronous model. Nevertheless, subsequently a synchronous version was introduced [12]. The result, namely, Discrete Time CNN (DTCNN), resembles the classical synchronous SIMD model. Furthermore, in a DTCNN the output of every processing element (PE) is determined by a threshold function. This kind of functions have been traditionally realized with TL. In summary, there is an overlap among the three aforementioned models, TL, DTCNN and SIMD, sharing design concerns and applications. This paper goes through the three models, putting the emphasis on their differences and similarities, and their suitability for processing binary images.

II. Threshold Logic

TLGs have mainly been used in artificial neural networks (ANN) and microprocessors [13]. TLGs are described by the threshold function:

$$Y = sgn[\sum_k w_k r_k + z] \tag{1}$$

where w_k mean certain weights or contributions, being these analog values, r_k are digital signals to select the corresponding w_k, and z is the threshold term. If the argument of Eq. (1) is greater than zero, the outcome becomes high (low), otherwise it turns low (high). The number of weights or contributions is determined by the summation term. This number is usually

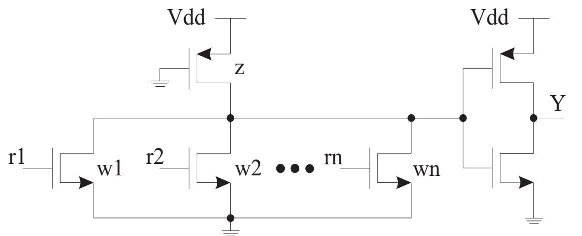


Fig. 1. Pseudo nMOS TLG.

fixed. Signals r_k are programmable. The threshold term z is either fixed or programmable. In microprocessor design, the goal is to find a certain circuit topology with proper transistor sizing in order to implement a given Boolean function. In this case, r_k are programmable and z is fixed. In ANN design, both r_k and z are programmable, as the function to do is not known a priori, or even if it is, the circuit has to perform a wide variety of functions, like for instance general purpose image processing tasks.

TLGs have come in different ways, ranging from CMOS to nanodevices. An in-depth survey on TLG implementations can be found in [13]. This work touches on pseudo nMOS (pMOS) solutions. Fig. 1 displays the most classical pseudo nMOS circuit. The NMOS branch competes with the PMOS transistor to fix the current and voltage at the inverter input node. The constraint is the Kirchhoff law, so that the current through the PMOS is equal to the sum of currents through the NMOS transistors. If the current through the PMOS transistor due to its V_{gs} was to be inferior to the total current in the NMOS branch, the voltage at the inverter input node would go low, and the output Y high (Vdd). If the opposite, the output is low (gnd). Concerning performance, more evolved versions to widen noise margins and drop power dissipation have been reported (again see [13] and references therein). Regarding functionality, the bias term z can be made programmable by means of several PMOS transistors in parallel controlled by different digital signals. As a result, pseudo nMOS gates turn to be a good candidate to implement DTCNN, or even Continuous Time (CT) CNN cells with a threshold function for the output-state relationship. Next section goes through this issue.

III. Cellular Non-linear Networks

CNN has found its main niche of applications in computer vision [9]. In hardware, one of the goals has been the design of visual processors provided with an analog or an *analogic* (analog and logic operations) core [14], taking advantage of the CNN computation model to reach a high performance. The ACE family and the Bi-i system, the latter made up of an ACE16k and companion DSP and FPGA chips, are good examples [4], [15]. Both solutions are asynchronous and capable of running general purpose gray-scale image tasks. In the case of Bi-i, algorithms executing thousands of frames per second with a 128×128 resolution have been reported [15].

In binary image processing, the circuits addressed in [16],

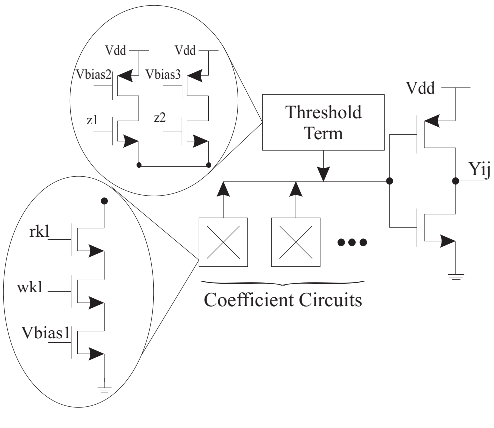


Fig. 2. CMOS implementation of a CNN cell with a threshold function for the output-state relationship.

[17] are shown to be very efficient. Both are 2D-CNN array implementations, in which a PE (cell) interacts with its neighbors within a 3×3 neighborhood to perform a wide variety of functions. In [16], the network can be set to run either synchronous or asynchronous operations. In [17] only synchronous tasks are executed. Both cases use the same CNN model. This can be defined by the next equation:

$$Y_{ij} = sgn[\sum_{k,l} w_{kl} r_{kl} - z] \tag{2}$$

where Y_{ij} is the output of the cell ij, w_{kl} are the cloning template coefficients, r_{kl} are cell variables, and z is the threshold term. The template coefficients w_{kl} determine the task to be done. These are 1-bit digitally programmable. They are the same for every cell ij, and set how the cell interacts with their neighbors within a 3×3 neighborhood. The cell variables r_{kl} are also 1-bit digital signals. Finally, the threshold term z is also programmable, and its value changes according to the image task. As it can be seen, Eq. (1) and Eq. (2) are essentially the same. Both can be realized by a TLG.

Fig. 2 depicts a circuit realization for Eq. (2). This is the solution adopted in [17]. The major difference with the pseudo nMOS gate shown in Fig. 1 is that now the programmability in w_{kl} and z causes to have more transistors. Signals $Vbias_k$ are analog fixed voltages that set the currents in the circuit. The number of transistors used for z depends on its number of bits of programmability. Also, observe that the digital programmability in w_{kl} and z could be replaced with analog programmability in $Vbias_k$. The result would be a less number of transistors, as the transistors driven by w_{kl} would not be needed. The price would be slower processing. Keeping the 1-bit digital programmability yields fast and still very dense solutions [16].

The cell density achieved in the implementation addressed in [16] can still be improved in CMOS technology by trading area for speed. The key is to drop the number of coefficient circuits. In [16], every cell contains 9 coefficient circuits (only one 3×3 template). The methodology discussed in [18] goes one step further, coming down up to 3 coefficient circuits. In so doing, every template is split into a succession of sub-templates. In the end, there is a final step to collect the outcomes of every sub-template. This is known as the split and shift methodology. The same idea is applied to large-neighborhood templates [19]. This methodology is doable because current CMOS technologies permit that every sub-template be executed fast enough as to still meet video frame rate in complex algorithms or applications. Furthermore, if the image acquisition is fast, thousands of frames per second might be feasible [20]. Also, the smaller area per cell produces more benefits. The most straightforward is to have chips with more cells. Another possibility is to occupy the saved area with modules to realize specific functions, or to enlarge transistors to have a better yield [21].

Concerning the optimal number of coefficient circuits, in [20] it is found that 4 coefficient circuits is a good compromise between area and time. In this case, if the allocation of such coefficient circuits faces North, East, West and South, and the architecture is synchronous, the resultant DTCNN cell resembles quite well the classical 1-bit SIMD PE with the North-East-West-South (NEWS) for nearest-neighbor communications. This is analyzed thoughout the next section.

IV. Single Instruction Multiple Data

SIMD is a computation paradigm that has been around in high-performance computing for many years. Today, clusters of PCs or workstations or some other technologies like vector-based are among the most successful supercomputers [22], [23]. Nevertheless, SIMD continues being a field of interest and giving high performance. In fact, the CNN chip ACE16k is a particular realization of an SIMD architecture with a peak performance about 0.19 TOPS in low-level vision applications for images with a 128×128 resolution, 7 bits of accuracy and an area around 1.44 cm^2 with 4 Watts of power consumption [4].

The mainstream of applications covered by SIMD falls into the category of numerically intensive tasks like scientific or engineering applications [8]. One of these applications is early vision. In this field, most of the operations are convolutions, being a repetitive operation for every pixel within the image. This leads to inherent data parallelism that suits SIMD quite well. Visual processors are examples of such systems. This kind of chips contains both the image acquisition plane (photodetectors) and the PE array [24]. It should also be noted that in the case of visual processors, the photodetector is normally embedded in every PE, even though there are chips in which the PE is shared among several photodetectors [6].

Fig. 3 shows a schematic view of a classical SIMD PE with its constituent parts. The PEs are arranged in a 2D array. All of them receive the same instruction from a Global Logic Unit (GLU). The Arithmetic Logic Unit (ALU) performs local processing like convolutions and logic operations. The North-East-West-South (NEWS) module is used to communicate with the nearest neighbors (located along the four cardinal directions). The flag system permits to do global operations, as for instance to know the first pixel (PE) to reach a certain state, or to inhibit the activity in certain PEs. Finally, some kind of register system or bank of memories (MEM in Fig. 3) is needed to store PE variables.

Both analog and digital PEs have been reported in the literature of on-chip solutions for early vision [11], [25]. In the digital domain, although the chips usually work synchronously, there are new solutions that combine asynchronous and synchronous operations on the same silicon substrate [26], [27]. The size of the digital word can vary from one to several bits. The use of 1-bit ALUs performing bit-slice computation to process wider digital words is also an alternative [28]. In binary image processing, the easiest approach consists of 1-bit ALUs. Also, as it has been pointed out throughout this paper, a PE composed of a dedicated ALU for binary image processing along with other circuitry for gray-scale operations can produce very efficient solutions, especially in those applications with a big percentage of B/W processing [17].

V. Discussion

In the case of a 1-bit ALU for running low-level image tasks, the modified pseudo nMOS TLG displayed in Fig. 2 seems to be a good solution. The difference from a synchronous CNN cell to a classical SIMD PE lies in the coefficient circuits and the communication among neighbors. In a CNN, the coefficient circuits transmit the information to the neighbors. Usually, cell (PE) states like memory contents do not go directly from cell to cell, but through the coefficient circuits. In this sense, in a CNN, the coefficient circuits can be regarded as the "NEWS" system. In the SIMD conception, the coefficient circuits would make part of the ALU. Coefficient circuit outputs and any other memory or flag contents would be sent to the neighbors via NEWS. The resemblance between synchronous CNN cells and classical SIMD PEs is even higher when the number of coefficient circuits drops to four and they interact with the neighbors along the four cardinal directions [20].

As the sizes of the processing arrays grow larger, capability to perform global operations effectively is becoming the key aspect in high speed applications. For example, objects of interest need to be identified and their location on the image plane needs to be extracted. A commonly used global operation is the global OR that identifies whether at least one PE fulfills a pre-defined criterion. However, the global OR does not tell what is the location of the first pixel to fulfill the criterion. The location can be found e.g. iteratively if flexible addressing [29] is included in the design. Another interesting means to notify of local events globally is the address event representation; if an event is generated, a PEs sends its address off the array and resets itself [30]. If the number of events is low enough, the temporal relations between the events are preserved and virtual connections between PEs are possible. A technique that can be used for detecting more complex events (combinations of bits) is possible by using a content addressable memory (CAM); PEs fulfilling a certain bit combination make their address available globally. When

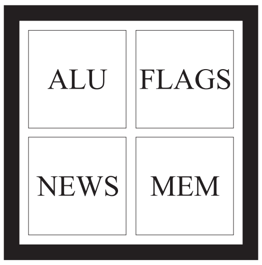


Fig. 3. Schematic view of the classical SIMD PE.

parallel write operation is added to the CAM, the resulting CAM associative processor can perform computation and data shifts by using parallel read and write operations successively [31]. A CAM associative processor and a SIMD processor can also be combined [32]. Such combinations of effective local and global processing capabilities are promising, since they result in a flexible computing platform.

VI. Conclusion

The paper has gone through three apparently different computing models, namely, TL, CNN and SIMD, examining their differences and similarities. It is concluded that when it comes to binary image processing, the synchronous CNN cell comes down to a a classical 1-bit SIMD PE. Its realization by means of pseudo nMOS (pMOS) TLGs has proved to be a very efficient solution to process binary images. In current and future CMOS technologies, this means to have either binary chips with higher resolution or PEs with modules to run more specialized functions.

Acknowledgment

This work is partly funded by Xunta de Galicia under the project PGIDT06TIC10502PR and the MEC-FPU program (Spain).

References

[1] Wiming Hu et al., *A Survey on Visual Surveillance of Object Motion and Behaviors*, IEEE Transactions on Systems, Man, and Cybernetics, vol. 34, n. 3, pp- 334–352, 2004.
[2] P. Dudek, D.L. Vilariño, *A Cellular Active Contours Algorithm Based on Region Evolution*, Proceedings of the 10th IEEE International Workshop on Cellular Neural Networks and Their Applications, CNNA2006, pp- 269–274, 2006.
[3] D.L. Vilariño et al., *Cellular Neural Networks and Active Contours: A Tool for Image Segmentation*, Image and Vision Computing, vol. 21, n. 2, pp- 189–204, 2003.
[4] A. Rodríguez-Vázquez et al., *ACE16k: the Third Generation of Mixed-Signal SIMD-CNN ACE Chips Toward VSoCs*, IEEE Transactions on Circuits and Systems-I, vol. 51,n. 5, pp. 851–863, 2004.
[5] *http://www.itrs.net/*.
[6] Péter Foldesy, Ákos Zarandy, Csaba Rekeczky, Tamás Roska, *Digital implementation of cellular sensor-computers*, International Journal of Circuit Theory and Applications, vol. 34, n. 4, pp- 409–428, 2006.
[7] *http://www.mathstar.com/index.html*.
[8] Lewis W. Tucker, George G. Robertson, *Architectures and Applications of the Connection Machine*, IEEE Computer Magazine, vol. 21, n. 8, pp- 26–38, 1988.
[9] L.O. Chua, T. Roska, *Cellular Neural Networks and Vision Computing. Foundation and Applications*, Cambridge University Press: Cambridge, 2002.
[10] Martin C. Herbordt et al., *A System for Evaluating Performance and Cost of SIMD Array Designs*, Journal of Parallel and Distributed Computing, vol. 60, pp- 217–246, 2000.
[11] P. Dudek, P.J. Hicks, *A General-Purpose Processor-Per-Pixel Analog SIMD Vision Chip*, IEEE Transactions on Circuits and Systems-I, vol. 52, n. 1, pp- 13–20, 2005.
[12] H. Harrer, J.A. Nossek, *Discrete-Time Cellular Neural Networks*, International Journal of Circuit Theory and Applications, vol. 20, n. 1, pp- 453–467, 1992.
[13] Valeriu Beiu, José M. Quintana, María J. Avedillo, *VLSI Implementations of Threshold Logic- A Comprehensive Survey*, IEEE Transactions on Neural Networks, vol. 14, n. 5, pp. 1217–1243, 2003.
[14] T. Roska, L.O. Chua, *The CNN Universal Machine: An Analogic Array Computer*, IEEE Transactions on Circuits and Systems-II, vol. 40, n. 3, pp. 163–173, 1993.
[15] Ákos Zarandy, Csaba Rekeczky, *Bi-i: A Standalone Ultra High Speed Cellular Vision System*, IEEE Circuits and System Magazine, vol. 14, n. 2, pp. 36–45, 2005.
[16] J. Flak et al., *Dense CMOS Implementation of a Binary-Programmable Cellular Neural Network*, International Journal of Circuit Theory and Applications, ol. 34, n. 4, pp- 429–443, 2006.
[17] V.M. Brea et al., *A Binary-based On-Chip CNN Solution for Pixel-Level Snakes*, International Journal of Circuit Theory and Applications, vol. 34, n. 4, pp- 383–407, 2006.
[18] N.A. Fernández et al., *On the Reduction of the Number of Coefficient Circuits in a DTCNN Cell*, Proceedings of the 10th IEEE International Workshop on Cellular Neural Networks and Their Applications, CNNA2006, pp- 29–34, 2006.
[19] N.A. Fernández et al., *On the Emulation of Large-Neighborhood with Binary CNN-Based Architectures*, Proceedings of the 9th IEEE International Workshop on Cellular Neural Networks and Their Applications, CNNA2005, pp- 274–277, 2005.
[20] N.A. Fernández et al., *Area and Time Efficient Cellular Non-linear Networks*, Accepted to International Symposium on Circuits and Sistems 2007, ISCAS2007, New Orleans, 2007.
[21] M. Laiho et al., *Effect of Mismatch on the Reliability of Binary-Programmable CNNs*, International Symposium on Circuits and Sistems 2006, ISCAS2006, pp- 3622–3625, 2006.
[22] Jack Dongarra, *Trends in High-Performance Computing*, IEEE Circuits and Devices Magazine, pp. 851–863, January/February 2006.
[23] *http://www.top500.org/*.
[24] Alireza Moini, *Vision Chips*, Kluwer Academic Publishers, 2000.
[25] T. Komuro et al., *A Dynamically Reconfigurable SIMD Processor for a Vision Chip*, IEEE Journal of Solid-State Circuits, vol. 39, n. 1, pp. 265–268, January 2004.
[26] Alexey Lopich, Piotr Dudek, *Architecture of a VLSI Cellular Processor Array for Synchronous/Asynchronous Image Processing*, IEEE International Symposium on Circuits and Systems 2006, ISCAS 2006, pp-3618–3621, 2006.
[27] Alexey Lopich, Piotr Dudek, *A Processing Element for a Digital Asynchronous/Synchronous Vision Chip*, IASTED International Conference on Circuits, Signals, and Systems, CSS, 2006.
[28] T. Komuro et al., *A Digital Vision Chip Specialized for High-Speed Target Tracking*, IEEE Transactions on Electron Devices, vol. 50, n. 1, pp. 191–199, January 2003.
[29] Piotr Dudek, *A flexible global readout architecture for an analogue SIMD vision chip* , IEEE International Symposium on Circuits and Systems 2003, ISCAS 2003, vol. III, pp- 782–785, 2003.
[30] K. A. Boahen, *Point-to-point connectivity between neuromorphic chips using address events* , IEEE Transactions on Circuits and Systems II, vol. 47, pp- 416–434, 2000.
[31] T. Ikenaga et al., *A fully parallel 1-Mb CAM LSI for real-time pixel-parallel image processing*, IEEE Journal of Solid State Circuits, vol. 35, pp. 536-544, 2000.
[32] A. Krikelis, et al., *A programmable processor with 4096 processing units for media applications*, IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 01, pp 937-940, 2006.

An Efficient FPGA Implementation of a DT-CNN for Small Image Gray-scale Pre-processing

J. Albó-Canals
Jose Á. Villasante-Bembibre
and Jordi Riera-Baburés
LIFAELS
La Salle - Universitat Ramon Llull
E-08022 Barcelona, Spain
Email: jalbo@salle.url.edu

N.A. Fernández-García
and Victor M. Brea
Departamento de Electrónica e Computación
Universidade de Santiago de Compostela
E-15782 Santiago de Compostela, Spain
E-mail:natalia.fernandez@usc.es

***Abstract*—This paper presents an 8-bit FPGA implementation of a Discrete Time Cellular Neural Network (DTCNN) suitable for small image gray-scale pre-processing (simple operations with high computational burden). It uses Split&Shift techniques to have a 31×31 grid that processes more than 2500 images per second. As this work evolves from a previous binary DTCNN implementation, results are compared in terms of area occupancy, routing complexity and processing time. Several design techniques have been applied to optimize the VHDL implementation on an Altera Stratix II-EP2S60F484C5 FPGA device. Moreover, as technology independent description allows easy migration to other devices or vendors, the benefits of FPGA technology evolution are discussed, focusing on DTCNN implementations.**

I. Introduction

Discrete Time Cellular Non-linear Networks (DTCNN) are good examples of a practical solution for image computation operating in discrete time steps, with direct pixel-to-processor assignment thanks to their local connectivity. Its basic functionality relies on templates, which are matrices of synapses values that establish the relationship between neurons. These templates are stored in libraries, and their combination allows the execution of different purpose instructions and even complex algorithms. The neighborhood level defines the maximum distance of local interconnections required between neurons, fixing also the size of the templates [1] and conditioning the physical architecture of the implementation.

Field Programmable Gate Array (FPGA) can implement parallel hardware structures very efficiently, specially locally connected ones. Moreover, their quick Time-to-Market and easy re-programmability makes them a very attractive solution. This paper shows an FPGA topographical implementation of an 8-bit gray-scale DTCNN which uses the Split&Shift techniques [2]. By applying these techniques a great reduction on the number of local interconnections between neighboring neurons and on the cell's internal arithmetic complexity is achieved. However, this is at the expense of execution time, because several sub-templates need to be applied through different stages in order to obtain the same result as with the original full template [3]. A configuration with four coefficient circuits in diamond shape, the so called North-East-West-South (NEWS), has the best trade-off between area reduction and processing time [2]. Also, it seems to be the most usual template shape in CNN algorithms and applications [4]. This is the configuration chosen for the 8-bit gray-scale DTCNN, as it was in the binary DTCNN case [5]. A full comparison with the binary DTCNN presented in [5] is performed taking into account aspects related to area, routing complexity, and performance.

A first approach to 8-bit gray-scale (which is a standard resolution) is preferred. This DTCNN implementation is a general purpose one, so being able to execute any kind of algorithm. That means to loose the opportunity of applying ad-hoc optimizations that can be realized in an application oriented implementation [6]. Still, hardware optimizations have been performed whenever possible in the design but always retaining the original general purpose nature of the DTCNN. In this sense, dedicated hardware to implement fixed state maps [1] has been removed, as they can be easily implemented using an instruction sequence. Moreover, the basic DTCNN cell operates on positive numbers, and thus the original templates need to be adapted to the physical hardware implementation. Our technology-independent VHDL description of an 8-bit DTCNN can be easily synthesized on different FPGA devices. So, it serves also as a good benchmark for testing technology evolution. We confirm significant improvements when migrating from an Altera Stratix I device to a Stratix II one.

The paper is organized as follows. Section II briefly recalls the DTCNN binary implementation. Section III explains the 8-bit gray-scale DTCNN implementation and design criteria. Discussion and comparisons of both implementations are performed in Section IV. And finally, in Section V, concluding remarks are given together with an insight of work in progress.

II. Binary DTCNN

The binary DTCNN is focused on black and white (B&W) image processing, which is enough for most of the low level applications like the robot guidance algorithm addressed in [8]. The core network consists of a grid of identical Processing Elements (PEs), or cells, that are locally connected and where each one is associated to a single pixel. This processing block interacts with a communication block, which includes the memories required as temporal storage for image input/output. Finally, there is a global Finite State Machine (FSM) acting as a control unit for all the system. It is responsible of loading the image to the grid, applying the Split&Shift techniques, managing the instruction set to apply an algorithm and finally, controls the communication block. A detailed description of the binary DTCNN implementation can be found in [5].

III. 8-Bit Gray Scale DTCNN

Although the basic functionality of the 8-bit implementation is the same of the binary one, its hardware structure is quite different. For instance, the FSM has been moved inside the PE while the local memory, implemented as an 8-bit RAM register, has been moved outside. These changes are made in order to save area. In this section the basic elements are described following a bottom-up approach, from the basic cell to the support structure of the Cellular Neural Network.

A. The Processing Element (PE)

The DTCNN cell/PE architecture is shown in Fig.1. The data received from the four neighbors is multiplied by the template data, obtaining a 64-bit result. A multiplexer chooses between accumulating the bias or the neighbors contribution. Finally, depending on the instruction being executed, the calculated data or the previous image data are send to the output. All this process, as well as the truncation needed to normalize data lengths, is controlled by a small local FSM.

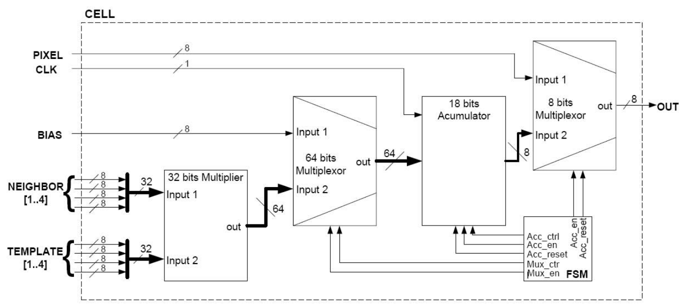


Fig. 1. Block diagram of an 8-bit gray-scale cell.

The main blocks of the PE are briefly discussed next.

1) 32-bit Multiplier: the 32-bit (8-bit*4) neighbor data are multiplied by the 32-bit (8-bit*4) template coefficients to obtain a 64-bit (16-bit*4) output which is the concatenation of each neighbor weighted contribution. This is done asynchronously to reduce time penalization and to simplify the general FSM.

2) 64-bit Multiplexer: chooses between the 64-bit calculated data and the 8-bit bias information (corresponding to the high part of the bus). In any case, the output is extended to 64-bit.

3) Accumulator: this element adds, and then accumulates, the output of 64-bit Multiplexer. First, the global contribution of the neighborhood (or bias) is calculated by adding the 16-bit blocks in which the multiplexer output is divided. After that, the result is truncated. Its output is synchronized.

4) 8-bit Multiplexer: this element is necessary to choose between the processed pixel or the original one than can be needed for the application of sub-templates or a different operation.

B. The DTCNN Grid

The DTCNN network is a grid of MxN cells connected in a NEWS way. The outer cells, due to the lack of some of their neighbors, are implemented as dummy cells that only provides the image value in the corresponding pixel.

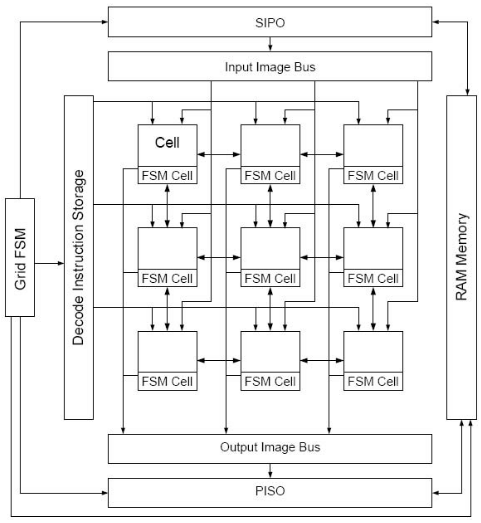


Fig. 2. 8-bit gray-scale DTCNN implementation schematic

The uploaded image is placed in a bus. The least significant byte is assigned to the North-West dummy cell, the one next to the right receives the second byte, and so on, until the East-South dummy cell receives the most significant byte. Data flow according to the local FSM, depending on the instruction being executed.

C. The CNN top

This block permits to upload and download an image to the grid. Also, it manages the instruction storage, which is necessary to apply any algorithm. This model is an upgrade of the B&W DTCNN implementation [5]. Apart from the obvious increase in the data size, from bit to byte, there are also two significant differences among both implementations. The first one is in the instruction block, that due to the cell changes has been redesigned, and the other one is in the memory manager, which will be discussed in the next section.

The instruction fields are shown in Table I. Notice that they are sets of nibbles (4-bit data groups). The increase from binary DTCNN implementation is obvious, as templates data are now 8-bit length. The reserved nibbles are to complete the 64-bit bus size and they can be used as future expansion to add others features.

The field meanings are as follows. The Bias field corresponds to the bias to be applied in the cell. When the operation value is 0 the image is put out of the grid, else it is processed.

TABLE I
INSTRUCTION ARRAY (FIELDS IN NIBBLES)

Bias	Template	Operation	WMR	WMW	OE	Reserved
2	8	1	1	1	1	2

The Which Memory Write (WMW) nibble indicates in which memory segment of M×N size (being M×N the size of the image). The Which Memory Read (WMR) reads it in case the cell has to operate with a different image than the actual one. Finally, the Output Enable (EE) starts the Parallel Input Serial Output (PISO) register in order to get the image out of the grid.

The global FSM is divided in two parts (see Fig. 2). The first one manages the grid controlling the image reception, processing it while doing the fetch and executing the instructions, and finally sending the image to the host. The second one is the memory manager, which writes and reads one of the six M×N memory blocks as needed by the algorithm being executed.

IV. FROM BINARY TO 8-BIT DTCNN IMPLEMENTATION

The DTCNN implementations using Split & Shift techniques are aimed at saving area and power consumption at expense of a reasonable time penalty. As shown in [5] and [2] some of the design decisions are only valid for 1-bit implementations, so new ones are needed for an 8-bit system. Also, as the 8-bit DTCNN is implemented in an Altera Stratix II EP2S60F484C5 device while 1-bit DTCNN was implemented on an Altera Stratix I EPS25F672C one, we can study how hardware resource usage changes with technology evolution. Finally, at the end of this section the results in terms of occupancy and performance of both implementations are compared.

A. Design Changes from 1-bit to 8-bit

Most design changes are introduced to alleviate the increase on area consumption, that is, PEs and routing complexity, when going from B&W to gray-scale DTCNN.

1) RAM registers instead of local flip-flops: in the binary DTCNN implementation each cell has 6 D flip-flop registers. Only 6-bit of storage are needed, and its direct implementation inside the PE is better than using dedicated memory to reduce routing complexity and increase performance. However, in the gray scale implementation, each flip-flop becomes an 8-bit register, and then these 6 registers are better mapped in a local memory block inside the FPGA (see Fig.2).

2) Truncation instead of Activation Function: the output-state relationship on the binary DTCNN implementation is given as an OR function of the 6 Most Significant Bits (MSB) obtained from the accumulator. Obviously, in the gray-scale DTCNN the output-state has a value range different from 1 or 0. So, the neighbor contribution is accumulated, then a bias is added and finally the output is normalized using a truncation technique. The truncation method has to be set according to the activation function chosen. As the activation function is piece wise linear, in the 8-bit implementation, a simple truncation method is used which selects the 8 LSB (Least-Significant-Bits). Of course, this can be easily changed if the application requires it.

3) Local FSM instead of global FSM: in some cases, 8-bit implementation multiply 8 times the number of connections, causing, in some cases, a not tolerable increase of in connection complexity. Distributed control system simplifies the FPGA routing permitting to increase the grid size. So the grid Control FSM has been simplified and a Local Cell FSM of only 3 states has been implemented [9].

4) Remove of the Local Logic Unit (LLU): experimental results have confirmed our supposition that in the 8-bit DTCNN implementation it is better to use template-based instructions to perform binary image operations (NOT, OR and AND) instead of having a dedicated LLU as it was used in the binary implementation [5].

B. Experimental instances

In order to achieve bigger and faster DTCNNs equivalent FPGAs from different technologies, Stratix I-EP1S25F672C and Stratix II-EP2S60F484C5 FPGAs by Altera, are tested. Comparing these two technologies it is apparent that a more efficient DTCNN can be implemented as better performance FPGA appear. The results shown in Fig. 3, Fig. 4 and Fig. 5 confirm this prediction.

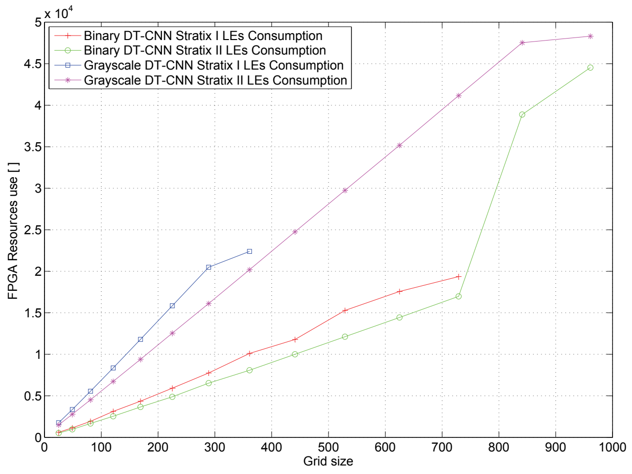


Fig. 3. FPGA Resources - Number of Logic Elements -

However, static elements created to improve features of new FPGA families can be a problem so the routing gets more complex losing area due to elements that have no possible connections.

V. RESULTS & CONCLUSIONS

A complete study to demonstrate the efficiency of an 8-bit gray-scale DTCNN FPGA implementation has been realized with Synplicity 9.6.2 version and Quartus II 8.0 through Aldec Active-HDL 8.1 Interface. This implementation allows to process a 31 × 31 8-bit gray-scale image in 394μs, having high enough performance for low resolution gray scale image processing applications (See Table II). Bigger images can be

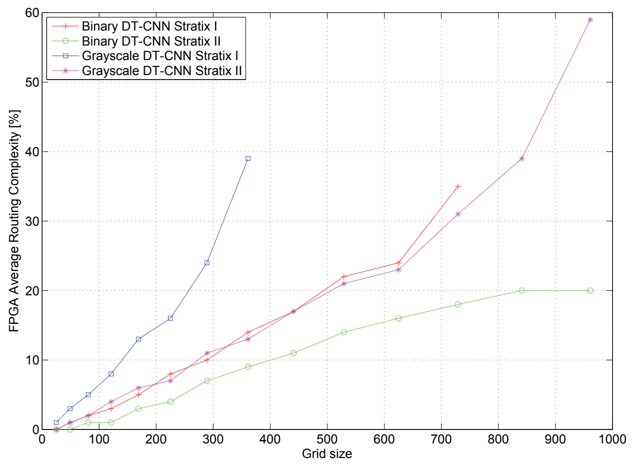


Fig. 4. FPGA Routing

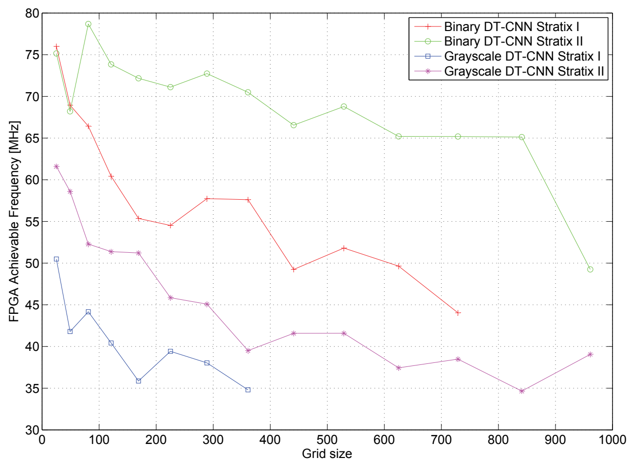


Fig. 5. FPGA Maximum Frequency

processed using windowing techniques. Moreover, as far as there is still enough hardware resources left in the Altera Stratix II-EP2S60F484C5 device, it is envisaged to implement an embedded processor inside the FPGA. This will allow the application of the windowing techniques while avoiding the performance penalty introduced by the communication, if a host PC were used instead.

In order to verify the functionality of the implementation we have evaluated a heat-diffusion template in terms of time consumption. $361\mu s$ are needed to load, process and save the output to memory of a 20×20 image (this is the maximum size achievable in the Stratix I FPGA). This is fast enough for typical FPGA image processing algorithms [10].

As expected, there is a real progression in terms of grid size if better FPGA devices are used (See Table III). So better results can be obtained as FPGA families evolve.

TABLE II
31x31 DTCNN RESULTS

RAM Memory	7688 of 2,544,192	$< 1\%$
Logic Elements	48321 of 48352	100%
Average Interconnect	59%	
Maximum Frequency	$39,06 Mhz$	

However, surprisingly enough, embedded FPGA blocks, like dedicated DSP elements, will increase the router complexity when used. The results show that they penalize DTCNN features in lieu of making it better, probably because their use hinders taking advantage of the local-connection paradigm of the DTCNN implementation. Of course, as the maximum working frequency directly depends on the routing complexity, designers have to be very careful in setting the synthesis options, thus DSPs are avoided. To obtain these results Altera Stratix III-EP3SL70F484C4 device and Altera Stratix IV-EP4SGX70DF29C4 device are used.

TABLE III
20×20 GRID DTCNN FEATURES

	Binnary Implementation			
	Area	LE	Max.Freq.(Mhz)	Routing Complexity
Stratix I	44%	11354	52,69	15%
Stratix II	27%	13253	65,08	9%
Stratix III	21%	11288	87,86	6%
Stratix IV	21%	12155	112,54	4%
	8-bit Gray Scale Implementation			
	Area	LE	Max.Freq.(Mhz)	Routing Complexity
Stratix I	93%	23809	27,98	50%
Stratix II	31%	15150	41,84	16%
Stratix III	28%	15155	66,19	13%
Stratix IV	26%	15159	56,89	8%

Finally, the hardware architecture has been designed using VHDL to be easily scalable both on image size and pixel bit depth, and benefits from being technology independent so it can be mapped to bigger and/or more powerful FPGA devices, even from a different vendor. This has the secondary effect of allowing easily benchmarking of the efficiency of DTCNN implementations on different FPGA devices, vendors, or using different synthesizing software. In the mid-term this ability, added to the natural technology evolution, will contribute on having bigger, more powerful, and application oriented DTCNN implementations on FPGA devices.

REFERENCES

[1] L.O. Chua et al., *The CNN Paradigm*, IEEE Trans. Circuits and Syst., vol. 40, pp. 147-156, 1993

[2] N.A. Fernández-García et al., *Split and Shift Techniques for CNN Hardware Reduction: First Mesurements*, ECCTD07, Seville, Spain, Aug. 2007

[3] N.A. Fernández-García et al., *On the Reduction of the Number of Coefficient Circuits in a DTCNN Cell*, CNNA06, Istambul, Turkey, Aug. 2006

[4] N.A. Fernández-García et al., *Template-Oriented Hardware Design based on Shape Analysis of 2D CNN Operators in CNN Template Libraries and Applications*, CNNA08, Santiago de Compostela, Spain, July 2008

[5] J. Albó-Canals et al., *Discrete Time Cellular Non-linear Networks Implementation over FPGA*, DCIS08, Grenoble, France, Nov. 2008

[6] A. Nieto et al., *Single Instruction Multiple Data and Cellular Non-linear Networks as Fine-Grained Parallel Solutions for Early Vision on FPGAs*, DCIS08, Grenoble, France, Nov. 2008

[7] Altera Corporation, *Stratix Programmable Logic Device Family Data Sheet*, Aug. 2002

[8] X. Vilasis-Cardona et al., *Guiding a Mobile Robot with Cellular Neural Networks*, Int. J. of Circuit Theory and App., Vol.30, n.6, pp. 611-624, 2002

[9] Z. Nagy et al., *Configurable multi-layer CNN-UM emulator on FPGA using Distributed Arithmetic*, ICECS02, Dubrovnik, Croatia, Sept. 2002

[10] T. Saegusa et al.,*How Fast is an FPGA in Image Processing?*, FPL08, Heidelberg, Germany, Sept. 2008, pp. 77-82

Appendix C

Acronyms List

ALU	Arithmetic Logic Unit
APR	Analog Program Register
ASIC	Application Specific Integrated Circuit
B/W	Black and white
CC	Coefficient Circuit
CNN	Cellular Non-linear Network
CNN-UM	Cellular Non-linear Network-Universal Machine
CNNUM, CNN-UM	CNN Universal Machine
CNT	Carbon Nanotubes
CPA	Cellular Processor Array
CPU	Central Processing Unit
CSW	Cellular Wave Computing Library
CTCNN, CT-CNN	Continuous Time CNN
DSP	Digital Signal Processor
DTCNN, DT-CNN	Discrete Time CNN
DT-CNNUM	Discrete Time CNN Universal Machine
FoM	Figure of Merit
FPGA	Field-Programmable Gate Arrays
FSR	Full-Signal Range Model
G/S	Gray-Scale

GACU Global Analogic Control Unit

GAPU Golbal Analogic Programming Unit

GPU Graphics Processing Unit

HR Hardware Reduction

H/V Horizontal and Vertical separability

II Integral Image

ITRS International Technology Roadmap for Semiconductors

LAM Local Analog Memory

LAOU Local Analog Output Unit

LCCU Local Communication and Control Unit

LLM Local Logic Memory

LLU Local Logic Unit

LN Large Neighborhood

LPA Linear Processor Array

LPR Logic Program Register

MAC Multiplier Accumulator Circuit

MIMD Multiple Instruction Multiple Data

NEWS North-East-West-South

OIF Operation Increment Factor

OPT Optical Sensor

PE Processing Element

PLS Pixel Level Snakes

RPO Reduction Per Operation, percentage of hardware Reduction Per Operation increased per original operation

RTD Resonant Tunneling Diodes

S and S Split and Shift

SCR Switch Configuration Register

SET Single-Electron Tunneling

SIFT	Scale Invariant Feature Transform
SIMD	Simple Instruction Multiple Data
SoC	System-on-Chip
SURF	Speed-Up Robust Features
TP	Topological Transformation
VSoC	Vision-System-on-Chip

Bibliography

R. Akbari-Dilmaghni. Mixed logic & analog implementation of large neighborhood cloning templates for CNNs. In *CNNA'98: Fifth International Workshop on Cellular Neural Networks and their Applications*, pages 277–281, London, UK, Apr. 1998. 23

R. Akbari-Dilmaghni and J. Taylor. Large neighborhood template implementation in continuous-time cellular neural networks with physical connectivity of r = 1. In *Proceedings of the 1996 International Symposium on Circuits and Systems. ISCAS'96. "Connecting the World"*, volume 3, pages 570–573, Atlanta, EEUU, May 1996. 25

J. Albó and Canals, J. A. Villasante-Bembibre, S. Consul-Pacareu, J. Riera-Baburés ands, and X. Vilasís-Cardona. Robot guiding with obstacle avoidance algorithm for uncertain environments based on DTCNN. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, July 2010. 71, 155

J. Albó-Canals, J. A. Villasante-Bembibre, J. Riera-Baburés, and X. Vilasís-Cardona. 8-bit gray-scale DTCNN implementation over an FPGA for robot guiding algorithm. In *2010 12th International Workshop on Cellular Nanoscale Networks and Their Applications (CNNA)*, Feb. 2010. 155

AnaFocus. *http://www.anafocus.com.* Seville, Spain. 17

M. Anguita, F. J. Pelayo, E. Ros, D. Palomar, and A. Prieto. VLSI implementations of CNNs for image processing and vision tasks: Single and multiple chip approaches. In *CNNA'96: Fourth International Workshop on Cellular Neural Networks and their Applications*, pages 479–484, Seville, Spain, June 1996. 16

A. Ayoub and S. Tőkés. The first version of the of the optical programmable array/analogic computer (POAC) template library. In *Proceedings of the 46th International Midwest Symposium on Circuits and Systems, MWSCAS'03*, volume 2, pages 1012,1015, Cairo, Egypt, Dec. 2003. 23

A. Ayoub, S. Tőkés, L. Orzó, and T. Roska. Evolution of the programmable optical array computer (POAC). In *Proceedings of the 8^{th} IEEE International Workshop on Cellular Neural Networks and their Applications. CNNA 2004*, pages 64–69, Budapest, Hungary, July 2004. 23, 91

D. R. W. Barr and P. Dudek. A cellular processor array simulator and hardware prototyping tool. In *11th International Workshop on Cellular Neural Networks and their Applications, CNNA2008*, pages 213–218, Santiago de Compostela, Spain, July 2008. 19

H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. *Computer Vision and Image Understanding (CVIU)*, 110(3):346–359, 2008. 21, 71, 78

V. M. Brea, D. L. Vilariño, A. Paasio, and D. Cabello. On the one-quadrant template design in a high gain CNN model. In *Proceedings of the 8th International Workshop on Cellular Neural Networks and their Applications*, pages 279–284, Budapest, Hungary, jul 2004a. 15, 26, 69, 86

V. M. Brea, D. L. Vilariño, A. Paasio, and D. Cabello. Design of the processing core of a mixed-signal CMOS DTCNN chip for pixel–level snakes. *IEEE Transactions on Circuits and Systems-I: Regular Papers*, 51(5):997–1013, 2004b. 39

V. M. Brea, M. Laiho, and A. Paasio. Robustness improvement in binary cellular nonlinear network architectures. In *European Conference on Circuit Theory and Design. ECCTD'05*, Cork, Ireland, August-September 2005a. 12

V. M. Brea, M. Laiho, D. L. Vilariño, A. Paasio, and D. Cabello. One-quadrant discrete time cellular neural network architecture for pixel level snakes: B/W processing. In *Proceedings of the IEEE International Symposium on Circuits and Systems, 2005. ISCAS '05*, pages 3922–3925, Kobe, Japan, may 2005b. 15, 82

V. M. Brea, M. Laiho, D. L. Vilariño, A. Paasio, and D. Cabello. A one-quadrant discrete time cellular neural network CMOS chip for pixel level snakes. In *Proceedings of the IEEE International Symposium on Circuits and Systems, 2005. ISCAS '05*, pages 5798–5801, Kobe, Japan, may 2005c. 15

V. M. Brea, M. Laiho, D. L. Vilariño, A. Paasio, and D. Cabello. A binary-based on-chip CNN solution for pixel-level snakes. *International Journal of Circuit Theory and Applications*, 34:383–407, 2006. 59, 68, 81, 82, 83, 84, 85, 86, 88

V. M. Brea Sánchez. *Design of a Mixed-Signal CMOS Integrated Circuit for Pixel-Level Snakes*. PhD thesis, Departamento de Electrónica e Computación, Universidade de Santiago de Compostela, Santiago de Compostela, Spain, Mar. 2002. 8, 12, 15

S. J. Carey, D. R. W. Barr, and P. Dudek. Demonstration of a low power image processing system using a scamp3 vision chip. In *2011 Fifth ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)*, aug. 2011. 18

L. Carranza, F. Jiménez-Garrido, G. Liñán-Cembrano, E. Roca, S. E. Meana, and A. Rodríguez-Vázquez. ACE16k based stand-alone system for real-time preprocessing tasks. In *Proceedings of Microelectronics for the New Millenium Symposium*, Seville, Spain, 2005. 17

S.-H. Chen and C.-Y. Wu. A low power design on diffusive interconnection large-neighborhood cellular nonlinear network for giga-scale system application. In *Proceedings of the 2004 11th IEEE International Conference on Electronics, Circuits and Systems. ICECS 2004*, pages 179, 182, Tel-Aviv, Israel, dec 2004. 23

S.-H. Chen and C.-Y. Wu. The quantum-dot large-neighborhood cellular nonlinear network (QLN-CNN) in nanotechnology. In *Proceedings of the 2001 1st IEEE Conference on Nanotechnology. IEEE-NANO 2001*, pages 331, 334, Maui, Hawaii, oct 2001. 23

L. O. Chua and T. Roska. *Cellular Neural Networks and Visual Computing*. Cambridge University Press, UK, 2002. 7

L. O. Chua and T. Roska. The CNN paradigm. *IEEE Transactions on Circuits and Systems*, 40(3):147–156, Mar. 1993. 7, 8

L. O. Chua and L. Yang. Cellular neural networks: Theory. *IEEE Transactions on Circuits and Systems*, 35(10):1257–1272, Oct. 1988a. 3, 7, 8

L. O. Chua and L. Yang. Cellular neural networks: Applications. *IEEE Transactions on Circuits and Systems*, 35(10):1273–1290, Oct. 1988b. 7, 11

S. Consul-Pacareu, J. Albó-Canals, X. Vilasís-Cardona, and J. Riera-Baburés. High performance DT-CNN camera device design on ACTEL IGLOO low power FPGA. In *2011 20th European Conference on Circuit Theory and Design (ECCTD)*, pages 37–40, Aug. 2011. 155

K. R. Crounse. *Image Processing Techniques for Cellular Neural Network Hardware*. PhD thesis, University of California, Berkely, Oct. 1997. 21, 24, 25, 33, 41

K. R. Crounse and L. O. Chua. The CNN universal machine is as universal as a turing machine. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 43(4):353 – 355, Apr. 1996. 10

J. M. Cruz and L. O. Chua. A CNN chip for connected component detector. *IEEE Transactions on Circuits and Systems*, 38(7):812–817, 1991. 16

R. Dolan and G. DeSouza. GPU-based simulation of cellular neural networks for image processing. In *International Joint Conference on Neural Networks, 2009. IJCNN 2009.*, pages 730 –735, June 2009. 19

R. Domínguez-Castro, S. Espejo, A. Rodríguez-Vázquez, R. Carmona, P. Földesy, A. Zarándy, P. Szolgay, T. Szirányi, and T. Roska. A 0.8um CMOS 2-D programmable mixed-signal focal-plane array processor with on-chip binary imaging and instructions storage. *IEEE J. Solid-State Circuits*, 32(7):1013–1026, 1997. 16

P. Dudek. Accuracy and efficiency of grey-level image filtering on VLSI cellular processor arrays. In *Proceedings of the IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA 2004*, pages 123–128, Budapest, Hungary, July 2004. 26

P. Dudek. Implementation of SIMD vision chip with 128 $\times$ 128 array of analogue processing elements. In *Proceedings of the IEEE International Symposium on Circuits and Systems, 2005. ISCAS '05*, volume 6, pages 5806–5809, Kobe, Japan, May 2005. 18, 68, 69

P. Dudek. *A programmable focal-plane analogue processor array.* PhD thesis, Institute of Science and Technology (UMIST), University of Manchester, Manchester, UK, may 2000. 1, 55

P. Dudek and P. J. Hicks. A general-purpose processor-per-pixel analog SIMD vision chip. *IEEE Transactions on Circuits and Systems-I: Regular Papers*, 52(1):13–20, Jan. 2005. 18, 77

S. Ehsan and K. McDonald-Maier. Exploring integral image word length reduction techniques for SURF detector. In *Computer and Electrical Engineering, 2009. ICCEE '09. Second International Conference on*, volume 1, pages 635 –639, dec. 2009. 79

S. Espejo. *Redes Neuronales Celulares: Modelado y Diseño Monolítico.* PhD thesis, Departamento de Electrónica y Electromagnetismo, Universidad de Sevilla, Seville, Spain, 1994. 16

S. Espejo, A. Rodríguez-Vázquez, R. Domínguez-Castro, J. Huertas, and E. S. Sinencio. Smart-pixel cellular neural networks in analog current-mode CMOS technology. *IEEE Journal of Solid-State Circuits*, 29(8):895–905, 1994. 13, 16

A. Fernández, R. San Martín, E. Farguell, and G. Pazienza. Cellular neural networks simulation on a parallel graphics processing unit. In *Cellular Neural Networks and Their Applications, 2008. CNNA 2008. 11th International Workshop on*, pages 208 –212, july 2008. 19

J. Fernández-Berni and R. Carmona-Galán. Accurate design of a mos-based resistive network for time-controlled diffusion filtering. In *European Conference on Circuit Theory and Design, 2009. ECCTD 2009*, pages 683 –686, Aug. 2009. 73, 77

J. Fernández-Berni, R. Carmona-Galán, and Á. Rodríguez-Vázquez. Image filtering by reduced kernels exploiting kernel structure and focal-plane averaging. In *ECCTD2011 EUROPEAN CONFERENCE ON CIRCUIT THEORY AND DESIGN*, pages 233–236, Linköping, Sweden, Aug. 2011. 75, 76, 77

N. A. Fernández García. Optimización hardware e mellora da funcionalidade de redes non lineais celulares. Master's thesis, Universidade de Santiago de Compostela, Santiago de Compostela, july 2006. 14

J. Flak, M. Laiho, A. Paasio, and K. Halonen. VLSI implementation of a binary CNN: First measurements results. In *Proceedings of the 8th International Workshop on Cellular Neural Networks and their Applications*, pages 129–134, Budapest, Hungary, jul 2004. 15

J. Flak, M. Laiho, and K. Halonen. On emerging nanodevices and architectures. In *2006 International Baltic Electronics Conference*, Tallinn, Estonia, Oct. 2006a. 20

J. Flak, M. Laiho, and K. Halonen. Programmable CNN cell based on set transistors. In *2006 10thInternational Workshop on Cellular Neural Networks and Their Applications*, Istanbul, Turkey, Aug. 2006b. 20

J. Flak, M. Laiho, A. Paasio, and K. Halonen. Dense CMOS implementation of a binary-programmable cellular neural network. *International Journal of Circuit Theory and Applications*, 34(4):429,443, 2006c. 17, 67, 68, 69

P. Földesy, A. Zarády, C. Rekeczky, and T. Roska. High performance processor array for image processing. In *International Symposium on Circuits and Systems, 2007. ISCAS 2007*, pages 1177–1180, New Orleans, LA, May 2007. 3

L. Furedi and P. Szolgay. CNN model on stream processing platform. In *European Conference on Circuit Theory and Design, 2009. ECCTD 2009*, pages 843 – 846, Antalya, Turkey, Aug. 2009. 19

M. Geese and P. Dudek. Autonomous long distance transfer on SIMD cellular processor arrays. In *2010 12th International Workshop on Cellular Nanoscale Networks and their Applications (CNNA)*, Berkeley, California, Feb. 2010. 25

C. Gerousis, S. M. Goodnick, W. Porod, and A. I. Csurgay. High-speed and low-power cellular non-linear networks using single-electron tunneling technology. In *IEEE International Symposium on Circuits and Systems, 2002. ISCAS 2002*, volume 2, pages 45–48, Phoenix-Scottsdale, AZ USA, May 2002. 20

B. Gilbert. A precise four-quadrant multiplier with subnanosecond response. *IEEE Journal of Solid-State Circuits*, 3(4):365–373, Dec. 1968. 14

H. Harrer and J. A. Nossek. Time discrete cellular neural networks: Architecture, applications and realization. Technical Report TUM-LNS-TR-90 12, Institute for Network Theory and Circuit Design, Technical University Munich, Nov. 1990. 4, 11, 12

H. Harrer and J. A. Nossek. Multilayer discrete-time cellular neural networks using time-variant templates. *IEEE Transactions on Circuits and Systems-II: Express Briefs*, 40(3):191–199, Mar. 1993. 11

H. Harrer, J. A. Nossek, and R. Stelzl. An analog implementation of discrete-time cellular neural networks. *IEEE Transactions on Neural Networks*, 3(3):466–476, 1992. 16

J. A. Hegt, D. M. W. Leenaerts, and R. T. Wilmans. A novel compact arquitecture for a programable full-range CNN in 0.5 um CMOS technology. In V. Tavsanoglu, editor, *1998 Fifth International Workshop on Cellular Neural Networks and their Applications*, pages 288–293, Londres, Inglaterra, Apr. 1998. 14, 26

International Technology Roadmap for Semiconductors. ITRS, 2009-2010. [Online]. In http://www.itrs.net/reports.html. 2

K. Karacs and M. Radvanyi. A prototype for the bionic eyeglass. In *Cellular Nanoscale Networks and Their Applications (CNNA), 2010 12th International Workshop on*, Feb. 2010. 17

A. Khitun and K. L. Wang. Cellular nonlinear network based on semiconductor tunneling nanostructure. *IEEE Transactions on Electron Devices*, 52(2):183 – 189, Feb. 2005. 20

P. R. Kinget. Device mismatch and tradeoffs in the design of analog circuits. *IEEE Journal of Solid-State Circuits*, 40(6):1212–1224, June 2005. 95

L. Koskinen, A. Paasio, and K. Halonen. 3-neighborhood motion estimation in CNN silicon arquitectures. In *Proceedings of the 2004 International Symposium on Circuits and Systems. ISCAS 2004*, volume 5, pages 708–711, Vancouver, Canada, May 2004. 25, 34

M. Laiho and E. Lehtonen. Cellular nanoscale network cell with memristors for local implication logic and synapses. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2051 –2054, june 2010. 20, 23, 91

M. Laiho, A. Paasio, J. Flak, and K. Halonen. Template design for binary-programmable cellular nonlinear networks. In *Proceedings of the IEEE International Symposium on Circuits and Systems, 2005. ISCAS '05*, volume 4, pages 3938–3941, Kobe, Japan, May 2005. 15, 17

M. Laiho, J. Poikonen, and A. Paasio. Space-dependent binary image processing within a 64×64 mixed-mode array processor. In *European Conference on Circuit Theory and Design, 2009. ECCTD 2009*, pages 189–192, Antalya, Turkey, August 2009. 17

G. Liñán. *Design of mixed-signal programmable chips with low power dissipation for real-time vision*. PhD thesis, University of Sevilla, Seville, 2002. 68, 69

G. Liñán, S. Espejo, R. Domínguez-Castro, and A. Rodríguez-Vázquez. ACE4k: An analog i/o 64x64 visual microprocessor chip with 7-bit analog accuracy. *International Journal of Circuit Theory and Applications*, 30(2/3):89–116, 2002. 16

A. Lopich and P. Dudek. An 80×80 general-purpose digital vision chip in $0.18\mu m$ CMOS technology. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems(ISCAS)*, pages 4257–4260, Paris, June 2010. 18

A. Lopich and P. Dudek. A SIMD cellular processor array vision chip with asynchronous processing capabilities. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 58(10):2420 –2431, Oct. 2011a. 3, 18, 95

A. Lopich and P. Dudek. Architecture and design of a programmable 3d-integrated cellular processor array for image processing. In *2011 IEEE/IFIP 19th International Conference on VLSI and System-on-Chip (VLSI-SoC)*, pages 349 –353, oct. 2011b. 19

A. Lopich, D. R. W. Barr, B. Wang, and P. Dudek. Live demonstration: Real-time image processing on ASPA2 vision system. In *2011 IEEE International Symposium on Circuits and Systems (ISCAS)*, page 1989, May 2011. 18

D. G. Lowe. Local feature view clustering for 3D object recognition. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, Kauai, Hawaii, December 2001. 21, 71, 73

S. Malki, Y. Fuqiang, and L. Spaanenburg. Vein feature extraction using DT-CNNs. In *Cellular Neural Networks and Their Applications, 2006. CNNA '06. 10th International Workshop on*, aug 2006. 88

C. Mead. *Analog VLSI e Neural Systems*. Addison-Wesley Publishing Company, Massachusetts, EEUU, 1989. 14

A. Moini. *Vision Chips*. Kluwer Academic Publishers, 2000. 3, 7

Z. Nagy and P. Szolgay. Configurable multi-layer CNN-UM emulator on FPGA. *IEEE Transactions on Circuits and Systems-I: Regular Papers*, 50(6):774–778, 2003. 18

Z. Nagy, L. Kek, Z. Kincses, and P. Szolgay. CNN model on CELL multiprocessor array. In *18th European Conference on Circuit Theory and Design, 2007. ECCTD 2007*, pages 276–279, Seville, Spain, Aug. 2007. 19

A. Nieto, V. M. Brea, and D. L. Vilariño. SIMD array on FPGA for B/W image processing. In *11th International Workshop on Cellular Neural Networks and their Applications, CNNA2008*, pages 202–207, Santiago de Compostela, Spain, July 2008. 18

A. Nieto, V. M. Brea, and D. L. Vilariño. FPGA-accelerated retinal vessel-tree extraction. In *International Conference on Field Programmable Logic and Applications, 2009. FPL2009*, pages 485 – 488, Prague, Czech Republic, Aug. 2009. 18

G. Nudd. Image understanding architectures. In *AFIPS Proc. National Computer Conference*, volume 49, pages 377–390, Anaheim, California, USA, May 1980. 2

A. Paasio. *Integration of Cellular Nonlinear Network Universal Machine*. PhD thesis, Electronic Circuit Design Laboratory, Department of Electrical and Communications Engineering, Helsinki University of Technology, 1998. 14, 15, 17

A. Paasio and A. Dawidziuk. CNN template robustness with different output nonlinearities. *International Journal of Circuit Theory and Applications*, 27:87–102, 1999. 12

A. Paasio and K. Halonen. A new cell output nonlinearity for dense cellular nonlinear network integration. *IEEE Transactions on Circuits and Systems-I: Regular Papers*, 48(3):272–280, 2001. 15

A. Paasio, A. Dawidziuk, and V. Porra. VLSI implementation of cellular neural network universal machine. In *The Third International Conference on Electronics Circuits and Systems*, pages 414–416, Rodes, Greece, 1996. 17

A. Paasio, A. Kananen, K. Halonen, and V. Porra. A 48 by 48 CNN chip operating with B/W images. In *Proceedings of the 5th IEEE International Conference on Electronics, Circuits and Systems*, pages 191–194, Lisbon, Portugal, Sept. 1998. 17

A. Paasio, A. Kananen, K. Halonen, and V. Porra. A QCIF resolution binary I/O CNN-UM chip. *Journal of VLSI Signal Processing Systems*, 23(2/3):281–290, 1999a. 17

A. Paasio, A. Kananen, and V. Porra. A 176x144 processor binary I/O CNN-UM chip design. In T. Roska, C. Beccari, M. Biey, P. Civalleri, and M. Gilli, editors, *Design Automation Day on Cellular Visual Microprocessor, European Conference on Circuit Theory and Design*, pages 82–86, Stressa, Italy, 1999b. Levrotto&Bella, Torino. 26

A. Paasio, M. Laiho, A. Kananen, and K. Halonen. An analog array processor hardware realization with multiple new features. In *International Joint Conference on Neural networks*, pages 1952–1955, Honolulu, Hawaii, 2002. 17, 23, 26, 52

A. Paasio, M. Laiho, A. Kananen, K. Halonen, and J. Poikonen. A 32×32 cellular test chip targeting new funtionalities. In *Proceedings of the 2003 International Symposium on Circuits and Systems. ISCAS'03*, volume 3, pages 506–509, Bangkok, Tailandia, May 2003. 17

A. Paasio, J. Flak, M. Laiho, and K. Halonen. High density VLSI implementation of a bipolar CNN with reduced programmability. In *Proceedings of the 2004 IEEE International Symposium on Circuits and Systems. ISCAS '04*, volume 3, pages 21–24, Vancouver, Canada, May 2004. 15, 26, 27

R. Perfetti, E. Ricci, D. Casali, and G. Costantini. Cellular neural networks with virtual template expansion for retinal vessel segmentation. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 54(2):141 –145, Feb. 2007. 72

J. Poikonen, M. Laiho, and A. Paasio. MIPA4K: a 64×64 cell mixed-mode image processor array. In *IEEE International Symposium on Circuit and Systems. ISCAS 2009*, pages 1927–1930, Taipei, Taiwan, May 2009. 17

S. Potluri, A. Fasih, L. K. Vutukuru, F. Al Machot, and K. Kyamakya. CNN based high performance computing for real time image processing on GPU. In *3rd International Workshop on Nonlinear Dynamics and Synchronization (INDS) 16th International Symposium on Theoretical Electrical Engineering (ISTET), 2011 Joint Conference*, July 2011. 19

Y. Pu, Y. He, Z. Ye, S. Londono, A. Abbo, R. Kleihorst, and H. Corporaal. From Xetal-II to Xetal-Pro: On the road toward an ultralow-energy and high-throughput SIMD processor. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(4):472 –484, Apr. 2011. 79

A. Rodríguez-Vázquez, S. Espejo, R. Domínguez-Castro, J. Huertas, and E. Sánchez-Sinencio. Current mode techniques for the implementation of continuous and discrete-time cellular neural networks. *IEEE Transactions on Circuits and Systems-II: Express Briefs*, 40(3):132–146, 1993. 13

A. Rodríguez-Vázquez, G. Liñán-Cembrano, L. Carranza, E. Roca-Moreno, R. Carmona-Galán, F. Jiménez-Garrido, R. Domínguez-Castro, and S. Espejo-Meana. ACE16K: The third generation of mixed-signal SIMD-CNN ACE chips towards VSoCs. *IEEE Transactions on Circuits and Systems-I: Regular Papers*, 51(5):851–863, 2004. 16, 67, 68, 69, 72, 77, 86

A. Rodríguez-Vázquez, R. Domínguez-Castro, F. Jiménez-Garrido, S. Morillas, J. Listán, L. Alba, C. Utrera, S. Espejo, and R. Romay. The Eye-RIS CMOS Vision System. In H. Casier, M. Steyaert, and A. H. M. van Roermund, editors, *Analog Circuit Design*, pages 15–32. Springer, 2008. 17, 95

A. Rodríguez-Vázquez, R. Carmona, C. D. Matas, M. Suárez-Cambre, V. Brea, F. Pozas, G. L. nán, P. Földesy, A. Zarándy, and C. Rekeczky. A 3D chip architecture for optical sensing and concurrent processing. In *Optical Sensing and Detection. Proceedings of the SPIE*, volume 7726, pages 772613–772613–12, Brussels, Belgium, Apr. 2010. 3, 19

T. Roska and L. O. Chua. The CNN universal machine: An analogic array computer. *IEEE Transactions on Circuits and Systems-II: Express Briefs*, 40(3):163–173, 1993. 3, 10, 11

T. Roska and Á. Rodríguez-Vázquez. Toward visual micropocessors. *Proceedings of the IEEE*, 90(7):1244–1257, July 2002. 8

T. Roska, L. Kek, A. Zarándy, and P. Szolgay. *CNN Software Library (CSL)*. Analogical and Neural Computing Laboratory, Computer and Automation Institute (MTA SzTAKI), Hungarian Academy of Science, 2000. [Online]. In http://lab.analogic.sztaki.hu/Candy/csl.html. 21, 54

F. Sargeni, V. Bonaiuto, and M. Bonifazi. Time division digital programmable OTA for cellular neural networks. In *European Conference on Circuit Theory and Design, ECCTD 2005*, Cork, Irlanda, 2005. 26, 55

K. Ślot. Large-neighborhood templates implementation in discrete-time CNN universal machine with nearest-neighbor connection pattern. In *Third International Workshop on Cellular Neural Networks and their Applications, CNNA'94*, pages 213–218, Roma, Italy, Dec. 1994. 22, 24, 26, 41, 91

B. G. Soos, A. Rak, J. Veres, and G. Cserey. GPU powered CNN simulator (SIMCNN) with graphical flow based programmability. In *11th International Workshop on Cellular Neural Networks and Their Applications, 2008. CNNA 2008*, pages 163 –168, July 2008. 19

Z. Szlavik, R. Tetzlaff, A. Blug, and H. Hoefler. Visual inspection of metal objects by using cellular neural networks. In *Cellular Neural Networks and Their Applications, 2006. CNNA '06. 10th International Workshop on*, aug 2006. 88

M. H. ter Brugge, J. A. G. Nijhuis, and L. Spaanenburg. Transformational DT-CNN design from morphological specifications. *IEEE Transactions on Circuits and Systems-I: Regular Papers*, 45(9):879–888, Sept. 1998a. 24, 69, 70

M. H. ter Brugge, J. H. Stevens, J. A. G. Nijhuis, and L. Spaanenburg. License plate recognition using DTCNNs. In *Proceedings of the Fifth IEEE International*

Workshop on Cellular Neural Networks (CNNA-98), pages 212–217, London, UK, Apr. 1998b. 21, 41

M. H. ter Brugge, J. H. Stevens, J. A. G. Nijhuis, and L. Spaanenburg. Efficient DTCNN implementations for large-neighborhood functions. In *Proceedings of the Fifth IEEE International Workshop on Cellular Neural Networks (CNNA-98)*, pages 88–93, London, UK, apr 1998c. 22, 24, 25, 26, 33, 41, 69, 70, 91

M. H. ter Brugge, J. A. G. Nijhuis, and L. Spaanenburg. Optimal decomposition of large-neighborhood CNN templates. In *Proceedings of the 8^{th} IEEE International Workshop on Cellular Neural Networks (CNNA-98)*, pages 160–165, Budapest, Hungary, jul 2004. 24

S. Tökés, L. Orzó, C. Rekeczky, T. Roska, and Á. Zarándy. An optical CNN implementation with stored programmability. In *ISCAS 2000. IEEE International Symposium on Circuits and Systems*, pages 136–139, Geneva, Switzerland, May 2000. 23

D. L. Vilariño. *Contornos Activos a Nivel de Píxel: Diseño e Implementación Sobre Arquitecturas de Redes No Lineales Celulares*. PhD thesis, Universidade de Santiago de Compostela, Santiago de Compostela, May 2001. 12, 21

D. L. Vilariño. Third evolution of pixel level snakes towards an efficient hardware implementation. Technical report, Department of Electronics and Computer Science, University of Santiago de Compostela, Santiago de Compostela, Dec. 2005. 81

D. L. Vilariño and C. Rekeczky. Implementation of a pixel-level snake algorithm on a CNNUM-based chip set architecture. *IEEE Transactions on Circuits and Systems-I: Regular Papers*, 51(5):885–891, May 2004. 59

D. L. Vilariño, D. Cabello, X. M. Pardo, and V. M. Brea. Cellular neural networks and active contours: a tool for image segmentation. *Image and Vision Computing*, 21(2):189204, Feb. 2003. 71, 81, 83, 84

D. L. Vilariño, V. M. Brea, V. Moreno, and D. Cabello. CNN implementation of spin filters for electronic speckle pattern interferometry applications. In *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, pages 2674 –2677, May 2007. 71, 72

P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511 – I–518 vol.1, 2001. 78, 79

Z. Vörösházi, A. Kiss, Z. Nagy, and P. Szolgay. A standalone FPGA based emulated-digital CNN-UM system. In *11th International Workshop on Cellular Neural Networks and their Applications, CNNA2008*, page 4, Santiago de Compostela, Spain, July 2008. 18

C.-Y. Wu and S.-H. Chen. The design and analysis of a cmos low-power large-neighborhood cnn with propagating connections. *IEEE Transactions on Circuits and Systems-I: Regular Papers*, 56(2):440–452, feb 2009. 23, 91

C.-Y. Wu and W.-C. Yen. A new compact neuron-bipolar junction transistor (νbjt) cellular neural network (CNN) structure with programmable large neighborhood symmetric templates for image processing. *IEEE Transactions on Circuits and Systems-I: Regular Papers*, 48(1):12–27, jan 2001. 23

S. Xavier-de Souza, M. Van Dyck, J. A. K. Suykens, and J. Vandewalle. Fast an robust face tracking for CNN chips: application to wheelchair driving. In *Cellular Neural Networks and Their Applications, 2006. CNNA '06. 10th International Workshop on*, aug 2006. 88

W.-C. Yen and C.-Y. Wu. A new compact programmable νbjt cellular neural network structure with adjustable neighborhood layers for image processing. In *Proceedings of ICECS'99. The 6th IEEE International Conference on Electronics, Circuits and Systems*, volume 2, pages 713, 716, Paves, Cyprus, set 1999. 23

Q. Yu. Spin filtering processes and automatic extraction of fringe centerlines in digital interferometric patterns. *Applied Optics*, 27(18):3782–3784, Sept. 1988. 71

A. Zarándy and C. Rekeczky. Bi-i: A standalone ultra high speed cellular vision system. *IEEE Circuits and Systems Magazine*, 5(2):36–45, 2005. 17

A. Zarándy and C. Rekeczky. 2D operators on topographic and non-topographic architectures–implementation, efficiency analysis, and architecture selection methodology. *International Journal of Circuit Theory and Applications*, 39:983–1005, 2011. 73

A. Zarándy, A. Stoffels, T. Roska, and L. O. Chua. Morphological operators on the CNN universal machine. In *1996 Fourth International Workshop on Cellular Neural Networks and their Applications*, pages 151–156, Seville, Spain, June 1996. 24

A. Zarándy, P. Földesy, P. Szolgay, S. Tökés, C. Rekeczky, and T. Roska. Various implementations of topographic, sensory, cellular wave computers. In *Proceedings of the IEEE International Symposium on Circuits and Systems, 2005. ISCAS '05*, volume 6, pages 5802–5805, Kobe, Japan, May 2005. 88