UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

DEPARTAMENTO DE ELECTRÓNICA E COMPUTACIÓN

Tesis doctoral

PARALELIZACIÓN Y OPTIMIZACIÓN DE UN SIMULADOR 2D MONTE CARLO SOBRE ARQUITECTURAS GRID Y CLUSTER: ESTUDIO DE FLUCTUACIONES EN TRANSISTORES MOSFET BASADOS EN SOI

Raúl Valín Ferreiro

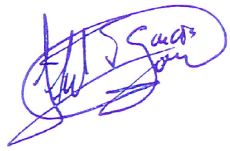
Santiago de Compostela, 12 de septiembre de 2011

Dr. Antonio García Loureiro, Profesor Titular de Universidad del Área de Electrónica de la Universidad de Santiago de Compostela,
Dra. Natalia Seoane Iglesias, Investigadora de la Universidad de Santiago de Compostela,
Dr. Carlos Sampedro Matarín, Profesor Titular de Universidad del Área de Electrónica de la Universidad de Granada.

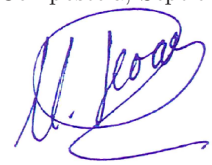
HACEN CONSTAR:

Que la memoria titulada **Paralelización y Optimización de un Simulador 2D Monte Carlo sobre Arquitecturas Grid y Cluster: Estudio de Fluctuaciones en Transistores MOSFET basados en SOI** ha sido realizada por D. **Raúl Valín Ferreiro** bajo nuestra dirección en el Departamento de Electrónica e Computación de la Universidad de Santiago de Compostela, y constituye la Tesis que presenta para optar al grado de Doctor en Ciencias Físicas.

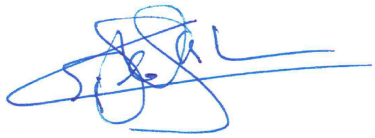
Santiago de Compostela, Septiembre de 2011



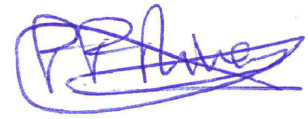
Fdo: **Dr. Antonio García Loureiro**
Codirector de la Tesis



Fdo: **Dra. Natalia Seoane Iglesias**
Codirectora de la Tesis



Fdo: **Dr. Carlos Sampedro Matarín**
Codirector de la Tesis



Fdo: **Dr. Francisco Fernández Rivera**
Director del Departamento de Electrónica y Computación

Fdo: **Raúl Valín Ferreiro**
Autor de la Tesis

A mis padres y a Ángela

Agradecimientos

En primer lugar me gustaría expresar mi agradecimiento a todas aquellas personas que me han ayudado de un modo u otro en la realización de esta tesis doctoral. Ya sea por sus aportaciones en forma de comentarios, críticas constructivas o con su apoyo anímico.

De modo especial quisiera expresar mi agradecimiento a los directores de este trabajo Antonio García Loureiro, Natalia Seoane Iglesias y Carlos Sampedro Matarín que me han guiado hasta aquí y en muchas ocasiones aconsejado sabiamente. De modo individual debo agradecer a Antonio su confianza, su comprensión y el apoyo que me ha dado cuando las cosas se complicaban. A Natalia por su complicidad, su dedicación para acabar esta memoria y su amistad. A Carlos por sus aportaciones, paciencia y ayuda con el simulador Monte Carlo.

Durante estos años he tenido la suerte de trabajar con un grupo de personas que me han ayudado a sortear numerosas dificultades y a las que les estoy sinceramente agradecido. A Manuel Aldegunde Rodríguez, me gustaría agradecerle su ayuda e infinita paciencia desde los inicios de esta tesis y a Enrique Comeseña Figueroa los divertidos rifirrafes que alegran los momentos de trabajo y su ayuda con las dificultades del día a día. Gracias también a los miembros del Departamento de Electrónica y Computación, y en especial a los integrantes del grupo de Arquitectura de Computadores por darme el apoyo necesario para acabar este trabajo, y a mis compañeros de laboratorio por la buena convivencia y ayuda durante estos años.

También me gustaría agradecer a los integrantes del grupo de investigación en Nanoelectrónica de la Universidad de Granada su ayuda, amistad y hospitalidad durante mi estancia en Granada. Así como al Dr. Karol Kalna del College of Engineering de la Universidad de Swansea, por sus consejos e ideas y al Prof.

Malcolm Atkinson, del Data Intensive Research Group por su apoyo y amabilidad durante mi estancia en Edimburgo.

Finalmente agradecer a mi familia y amigos su apoyo durante estos años. En especial a mis padres y abuelos, por su confianza y esfuerzo. A mis amigos Ignacio, Maica, Santi y Lucía, con quienes comparto los buenos momentos. Y a Ángela, con la que he superado todas las vicisitudes que me han llevado hasta aquí.

Santiago de Compostela, 12 de septiembre de 2011

Todos somos muy ignorantes. Lo que ocurre es que no todos ignoramos las mismas cosas.

A. Einstein

Índice general

Índice de figuras

Índice de tablas

Introducción

La simulación de dispositivos semiconductores ha sido una herramienta fundamental en el diseño de los transistores actuales y lo será todavía más, en el diseño de los dispositivos futuros. Esto es debido a que facilita la implementación de nuevos modelos físicos que permiten predecir el comportamiento de los transistores cuando modificamos diferentes parámetros de diseño. Los resultados obtenidos con las herramientas de simulación, combinados con aquellos que se obtienen en las pruebas experimentales, facilitan la elección del mejor diseño y reducen tanto el tiempo de desarrollo como el coste de investigación.

El desarrollo tecnológico experimentado por el escalado de los transistores, en su intento por aumentar la capacidad de integración y la velocidad de operación así como por reducir el consumo de potencia, ha permitido alcanzar tamaños del orden de decenas de nanómetros. Sin embargo, para estas dimensiones los modelos de simulación se vuelven extremadamente complejos incrementando significativamente el tiempo de cálculo. Además, desde el punto de vista tecnológico, estas dimensiones suponen el límite de la tecnología tradicional siendo necesario explorar nuevas posibilidades como por ejemplo la introducción de nuevos materiales y geometrías para continuar mejorando las prestaciones de los transistores que se fabriquen

en el futuro. Un ejemplo de este cambio es la tecnología de 45 nm de Intel que reemplaza la combinación tradicional de dióxido de silicio y polisilicio, empleados como materiales básicos para la fabricación de la puerta de los transistores MOSFET desde los años 60, por dióxido de hafnio y un metal de puerta [CBD+05]. Por si este cambio no fuera suficiente, hace unos pocos meses Intel anunciaba la producción en masa de transistores de triple puerta (*Tri–Gate*) para el nodo tecnológico de 22 nm, lo que supone la aparición en este mercado del primer transistor 3D. Estos ejemplos muestran cómo, posiblemente durante los tres últimos años hemos asistido a los cambios tecnológicos más importantes en el diseño y fabricación de los transistores MOSFET desde su creación.

Una de las consecuencias directas del escalado de los transistores es el mayor impacto que tiene sobre las prestaciones de los dispositivos los efectos de variabilidad que aparecen en el proceso de fabricación. La disminución del tamaño facilita que aumente la capacidad de integración y por lo tanto podemos aumentar el número de transistores por unidad de área en un circuito integrado (CI). De este modo, si incrementamos la capacidad de integración es importante conocer cuales son límites de variación de los parámetros que determinan el funcionamiento de los transistores que forman parte de nuestro CI, como la tensión umbral, la corriente de conducción o la corriente de corte.

Para realizar estudios de variabilidad y probar nuevos diseños, modelos físicos y materiales, es necesario reducir al máximo el tiempo de ejecución de las simulaciones e incrementar en la medida de lo posible el número de recursos computacionales. Con la aparición de los procesadores multi–núcleo es posible ejecutar aplicaciones paralelas en ordenadores de escritorio o en pequeños *clusters* [Gor07]. Además, gracias al desarrollo de las tecnologías Grid [Too, GLi] y a la reciente creación de las infraestructuras Grid Nacionales, como parte de la Iniciativa Grid Europea (EGI) [egi], disponemos de una cantidad importante de núcleos de computación que podrían ser muy útiles para los estudios de fluctuaciones.

En esta tesis doctoral se pretende aplicar las ventajas de las arquitecturas multi–núcleo y Grid en el estudio del impacto de varias fuentes

de fluctuaciones sobre el comportamiento de diferentes dispositivos. Para ello, se ha comenzado realizando una optimización y paralelización con el estándar OpenMP de las herramientas de simulación empleadas. A continuación, se han evaluado las posibles ventajas del uso de arquitecturas de tipo Grid, fundamentalmente en el estudio de fluctuaciones en donde se requieren cientos o miles de simulaciones. Finalmente, se ha estudiado con los simuladores paralelizados el impacto de varias fuentes de fluctuaciones sobre el comportamiento de diferentes dispositivos. La estructura de esta memoria es la que se muestra a continuación:

En el capítulo uno introducimos el transistor MOSFET y el estado actual de la tecnología, haciendo especial hincapié en las alternativas al MOSFET convencional, basado en tecnología *Bulk*, que han sido objeto de estudio durante los últimos años.

En el capítulo dos se presenta inicialmente una breve introducción a los diferentes métodos de simulación de dispositivos semiconductores. A continuación se explican las características del método Monte Carlo y se describen los simuladores empleados en esta memoria.

El capítulo tres empieza con una introducción a los sistemas distribuidos y a las arquitecturas multi–núcleo. A continuación se describe el estándar OpenMP empleado en la paralelización de los simuladores. Después de esta introducción de la arquitectura y del estándar OpenMP, describimos el proceso de paralelización de los dos simuladores de transistores MOSFET empleados, el 2D Multivalle Monte Carlo con correcciones cuánticas y el 2D Multisubbanda Monte Carlo. Finalmente, se muestran los resultados obtenidos tras ejecutar los códigos paralelizados en diferentes procesadores multi–núcleo. Además, se analizan los valores de aceleración en función del número de núcleos empleados, así como las posibles causas que limitan estos valores y el efecto de diferentes parámetros de la simulación sobre la aceleración de las ejecuciones.

El capítulo cuatro se inicia con una introducción a la computación Grid, su arquitectura y tecnologías empleadas. A continuación, se describen las características de la infraestructura Grid española, que ha servido de infraestructura de prueba para evaluar las posibilidades de la compu-

tación Grid en la simulación de dispositivos semiconductores. Finalmente, se muestran los resultados de dos pruebas de evaluación del Grid con el simulador, la primera orientada a reducir el tiempo de ejecución de las simulaciones Monte Carlo estacionarias y la segunda centrada en su aplicación en el estudio de fluctuaciones, en la que se analiza el impacto de la heterogeneidad de los recursos en el tiempo de ejecución de las simulaciones.

En el capítulo cinco se presentan los resultados de la simulación de dos fuentes de variabilidad en transistores SOI MOSFET. En el primer estudio se analiza, con el simulador paralelo 2D Multivalle Monte Carlo con correcciones cuánticas, el impacto del desalineamiento de las puertas en transistores MOSFET de doble puerta de silicio sobre aislante. En el segundo estudio analizamos, con el simulador paralelo 2D Multisubbanda Monte Carlo, el impacto de la variabilidad en el espesor del óxido debida a la presencia de una interfaz rugosa de HfO_2/SiO_2 en el dieléctrico de un transistor MOSFET de silicio sobre aislante con una única puerta.

Finalmente, se presentan las principales conclusiones y líneas futuras de investigación que han surgido en este trabajo, siguiendo dos líneas diferentes y al mismo tiempo dependientes. Una orientada hacia la computación paralela y el uso de infraestructuras Grid en la simulación de dispositivos semiconductores, y una segunda línea relacionada con la simulación Monte Carlo de transistores MOSFET y el estudio de fluctuaciones.

Introduction

The simulation of semiconductor devices is a very important tool in the design of the current and future devices. Thanks to simulation tools it is possible to implement new physical models to predict the electric behaviour of the devices for different design parameters. Results from simulations combined with experimental data enable us to evaluate the best design reducing the development time and research cost.

Historically, the scaling of transistors has been focused on reaching higher integration levels, faster transistors and lower power consumption, enabling the reduction of the dimensions of the transistors to some tens of nanometres. Furthermore, from the technological point of view it is necessary to carry out research on new materials and geometries to continue the scaling process because current dimensions are in the physical limit of Bulk technology. The 45 nm Intel transistor is an example of this research, since it replaces the traditional SiO_2 used as gate dielectric since 1960's decade by HfO_2. A more radical example related to the research of new architectures was announced by Intel, several months ago, concerning the mass production of FinFETs for the 22 nm technological node. This will be the first 3D transistor on the market. These examples demonstrate that during the last three years we have probably seen the most impor-

tant changes related to design and fabrication of MOSFET transistors since their creation. However, the physical models for these dimensions are extremely complex and require more simulation time.

The increase of variability effects is one of the consequences of the scaling of transistors. These effects cause fluctuations of the electric parameters on manufactured devices. High integration capacities require the knowledge of the variability limits of electric parameters such as, threshold voltage, on/off current or DIBL (Drain Induced Barrier Lowering).

The study of variability effects along with the testing of new designs, physical models and materials require fast simulations and a large number of computational resources. Nowadays, thanks to multicore architectures it is possible to run parallel applications on desktop computers or small clusters. Furthermore, the development of Grid technologies and the recent creation of National Grid infrastructures, as part of the European Grid Initiative (EGI), enable us the access to a large amount of computational cores which allow us to study the impact of different sources of fluctuations.

In this work we have increased the performance of 2D Monte Carlo simulators through the parallelisation and optimisation of their codes using OpenMP. We have also assessed the advantages of using Grid architectures for fluctuations studies where hundreds or thousands of simulations are required. Finally, we studied the impact of several sources of fluctuations using the previously parallelised simulators. The structure of this dissertation is divided as follows:

Chapter one introduces the MOSFET transistor and the state-of-the-art technology, paying special attention to the recently developed alternatives to Bulk technology.

The first part of chapter two introduces the possible methods used to simulate semiconductor devices. The second one describes the Monte Carlo method and the main characteristics of the parallelised simulators

Chapter three starts with an introduction to distributed systems, multicore architectures and a general description of the OpenMP standard used to parallelise the Monte Carlo simulators. This chapter continues

with an explanation of the parallelisation process of the 2D quantum–corrected Multi–Valley and 2D Multi–Subband Monte Carlo simulators. Finally, the speed up results obtained from different multicore processors are shown and its dependence on several simulation parameters is also analysed.

Chapter four starts with an introduction to Grid architectures and technologies, and the description of the Spanish Grid infrastructure that has been employed to assess the advantages of using Grid to simulate semiconductor devices. Finally, we present the results obtained from two tests on this infrastructure using the Monte Carlo code. The objective of the first test is to speed up stationary Monte Carlo simulations and the objective of the second one is to analyse the influence of the Grid infrastructures heterogeneity on the execution time.

Chapter five presents the simulation results of the impact of two sources of variability on the performance of SOI MOSFET transistors. Initially, we analyse the impact of the gates misalignment in a 10 nm gate length Double Gate SOI device. This study has been performed using the parallelised 2D quantum–corrected Multi–Valley Ensemble Monte Carlo (MV–ECBE–EMC). After that we study the impact of the oxide thickness variability induced by a rough HfO_2/SiO_2 interface on a SGSOI MOSFET. This analysis has been done with the parallelised 2D Multi–Subband Ensemble Monte Carlo simulator (MSB–EMC).

Finally, this dissertation ends up showing the main conclusions and future research lines that have arisen from this work.

CAPÍTULO 1

Dispositivos MOSFET

Durante más de 30 años, el transistor de efecto campo metal–óxido–semiconductor (MOSFET) ha sido la base de los circuitos integrados. Ya en el año 1965, Gordon E. Moore, observando que hasta el momento existía un crecimiento exponencial en el número de transistores en un chip, postuló que esta tendencia continuaría en el futuro [Moo65]. Esta afirmación, conocida hoy en día como la ley de Moore, expresa que cada 18 meses se duplica el rendimiento de los transistores y se cuadriplica el número de dispositivos presentes en un chip [Won02]. Esta ley se ha cumplido hasta hoy en día gracias al escalado agresivo de los dispositivos, que permite mejorar la velocidad de operación y la densidad de integración. Sin embargo, al alcanzar escalas nanométricas no es posible basar esta mejora únicamente en la reducción de dimensiones, siendo necesario encontrar nuevas alternativas para mejorar el rendimiento de los transistores. Por ejemplo, las corrientes de fuga imponen un límite al escalado de la tensión de alimentación y es necesario el uso de un principio de escalado generalizado [BWD84] que disminuye el cambio en la tensión de alimentación pero aumenta el dopado del canal más rápido para balancear los

campos lateral y vertical.

En la primera parte de este capítulo describiremos los transistores MOSFET. Empezamos con una breve descripción del dispositivo en su estructura convencional, estudiando a continuación las limitaciones que aparecen con el escalado y como influyen en su diseño. A continuación describimos algunas alternativas propuestas para superar las limitaciones de la arquitectura convencional. Finalmente, se presentan las ventajas de la tecnología de silicio sobre aislante y una clasificación, en función del número de puertas, de los dispositivos que se basan en esta tecnología de fabricación.

1.1 El MOSFET convencional, escalado y estado actual de la tecnología

Los transistores de efecto campo son dispositivos unipolares que involucran principalmente el transporte de los portadores, ya sean electrones o huecos, en una capa paralela a la superficie bajo la puerta. El más utilizado hoy en día es el MOSFET, que se caracteriza por aislar la capa de transporte del metal (o polisilicio) de puerta mediante la utilización de un óxido. El funcionamiento del dispositivo está basado en la modulación de la conductividad del canal mediante la aplicación de una tensión en la puerta. Este principio se ilustra, para un MOSFET estándar típico de canal n, en la figura 1.1 mediante una representación esquemática del transistor. En este caso hay dos regiones con dopado tipo n alto, la fuente y el drenador, y el resto del dispositivo está dopado con impurezas aceptoras. Al aumentar el voltaje aplicado en la puerta disminuye la barrera que ven los electrones para pasar de fuente a drenador, lo que hace que aumente la conductividad.

En las tecnologías convencionales el semiconductor utilizado es el silicio. Este material tiene la ventaja de ser oxidado fácilmente para formar SiO_2 de una manera altamente controlable y reproducible. Además, la superficie de separación Si–SiO_2 se puede crear con una muy buena regularidad, produciendo muy pocos defectos. Esto permite que los MOSFETs

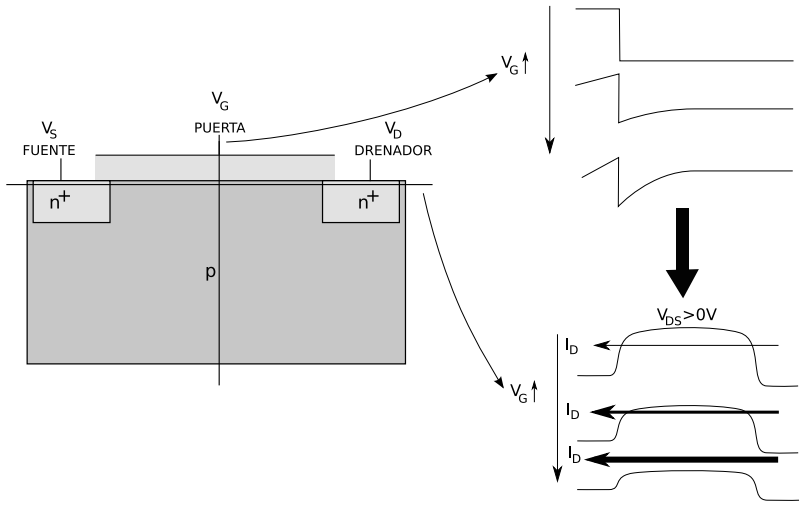


Figura 1.1: Esquema básico del funcionamiento de un MOSFET.

de Si se puedan fabricar en grandes cantidades, siendo fácilmente integrables para formar circuitos a gran escala.

La teoría original de escalado fue presentada en la década de los 70 [DGKY72, DGR$^+$74], ofreciendo parámetros de diseño concretos para la fabricación de MOSFETs de dimensiones reducidas. De acuerdo con esta teoría, si las dimensiones físicas y el potencial aplicado se escalan con un factor $1/k$ ($k > 1$) y la concentración de impurezas se aumenta un factor k, entonces la forma del campo eléctrico permanece constante. Sin embargo, al diseñar dispositivos con longitud de canal por debajo de 1 μm, el escalado del voltaje umbral y de la alimentación pasa a ser demasiado agresivo y se presenta una nueva teoría de escalado generalizado [BWD84] que permite escalar los potenciales de forma independiente, dando una mayor flexibilidad al diseño. Una vez más, al disminuir las dimensiones físicas de los dispositivos, la teoría del escalado empieza a acercarse a su límite por motivos prácticos. Muchos de los problemas que ahora surgen son intrínsecos a la naturaleza de los semiconductores y no pueden ser eliminados por medio de mejoras en el procesamiento o en el equipamiento [Ase98]. Por ejemplo, el escalado de los dispositivos MOSFET a nodos

tecnológicos inferiores al de 50 nm requiere la superación de barreras de naturaleza física que limitan su rendimiento [TBC+97, FDN+01, Fis03]. Algunos de los problemas más frecuentes son:

- Mayor sensibilidad a los efectos de canal corto.
- Corrientes de fuga a través del óxido de puerta.
- Efecto túnel de portadores desde la fuente al drenador o desde el drenador hacia el interior del dispositivo.
- Control de la densidad y posición de los átomos dopantes en el canal y en las regiones de fuente y drenador para poder proporcionar una elevada relación entre las corrientes de conducción y de corte.

En los nodos tecnológicos actuales se han introducido numerosas variaciones sobre el MOSFET convencional. Entre estos cambios está el uso de puertas metálicas, óxidos de alta permitividad en la puerta o el uso de silicio tenso para aumentar la movilidad [MAA+07]. Además, compañías como IBM o AMD también introducen en nodos actuales diseños que van un paso más allá del MOSFET convencional utilizando la tecnología de silicio sobre aislante (SOI) [IN07, SCB+08] que, entre otras ventajas, permite reducir los efectos de canal corto (SCE) [CC03].

Pese a las innovaciones introducidas para mejorar el rendimiento y continuar el escalado, hay dificultades tecnológicas que, en el estado actual de la tecnología, no tienen solución. El documento conocido como *SIA's International Technology Roadmap for Semiconductors* (ITRS) [ITR10] es una guía para la industria de los semiconductores que revisa los parámetros de diseño necesarios para poder ajustarse tanto a la ley de Moore como a las barreras tecnológicas a superar. En su última versión (*2010*) predicen la desaparición de la tecnología convencional en el nodo tecnológico de 22 nm, con lo que para nodos posteriores será necesaria la introducción de tecnologías y/o materiales nuevos que permitan la continuación del escalado.

1.2 Alternativas al MOSFET convencional

Una de las principales limitaciones de la tecnología convencional se encuentra en que el Si es un material de baja movilidad. La mayoría de los semiconductores compuestos (GaAs, InP, etc.) tienen movilidades mayores que la del silicio pero en la actualidad, a nivel comercial, no son utilizados en la fabricación de MOSFETs principalmente debido a la dificultad de obtener aislantes válidos [YWK+03], pese al progreso realizado en los últimos años [Pas05, Dat07, DRA+07].

Los límites asociados con el escalado y el rango de aplicación de los MOSFETs convencionales han dado un impulso a la investigación y desarrollo de propuestas alternativas en el diseño de transistores y en el uso de nuevos materiales. A continuación se muestra una posible clasificación de estas propuestas en función de la técnica que usan para mejorar el rendimiento del dispositivo:

- Inducir una densidad de carga más elevada para una tensión de puerta dada. Por ejemplo, esto puede lograrse reduciendo la temperatura de funcionamiento del sistema [Won02] o usando un dispositivo FET de doble puerta (*Double Gate FET*) [IWN+02, SGZ+03].

- Aumentar el transporte de portadores elevando la movilidad, la velocidad de saturación o el transporte balístico. Para ello, entre otras técnicas, es posible disminuir la temperatura de funcionamiento, reducir la influencia de factores que degradan la movilidad, minimizando el campo eléctrico transversal o la dispersión coulombiana debida a átomos dopantes, o utilizar materiales de alta movilidad [Vog07] y velocidad de saturación, como pueden ser Ge, InGaAs o InP. Entre los ejemplos de dispositivos que utilizan estas técnicas se encuentran los HEMT [YES+02], PHEMT [CCW05], MHEMT [WMH+00], SiGe MOSFET [BOC02], HBT [WML+03] y DHBT [IKW02].

- Asegurar la escalabilidad del dispositivo a una longitud de puerta más pequeña. Esto se puede conseguir utilizando perfiles de dopado

más bruscos, a través de una capacidad de puerta elevada o manteniendo un buen control electrostático del potencial en el canal. Esto ocurre en dispositivos FET de doble puerta, *Ground Plane* FET y *Ultra Thin Body* SOI MOSFET [JTC02] entre otros.

- Reducir capacidades y resistencias parásitas. Estas técnicas son también empleadas en dispositivos SOI y FET de doble puerta [WFS98].

A continuación describiremos con más detalle dos caminos posibles para conseguir alcanzar los requerimientos descritos en el ITRS: los MOSFET multipuerta y la introducción de nuevos materiales en el canal.

1.2.1 MOSFETs multipuerta

La disminución de la longitud de canal de los transistores que implica el escalado lleva a la aparición de los efectos de canal corto, que básicamente consisten en la pérdida del control de la carga por parte del terminal de puerta. Una posibilidad para mejorar esto es el uso de varias puertas para controlar la carga. Hay principalmente dos variedades: transistores multipuerta verticales y planares. Los primeros corresponden, por ejemplo, a los FinFET [HLK$^+$00, YCA$^+$02] o los TriGate [KDD$^+$06], mientras que los segundos son principalmente derivados de la tecnología SOI [CC03] que veremos en la sección 1.3.2.

1.2.2 Nuevos materiales

Aunque durante las últimas décadas la tecnología de fabricación de transistores se ha basado fundamentalmente en Si, cuando escalamos por debajo de 100 nm es necesario introducir nuevas técnicas de escalado para mejorar el rendimiento. Además de las nuevas arquitecturas, la industria también ha introducido nuevos materiales para lograr mejoras en el rendimiento. Un ejemplo es la utilización de materiales de alta permitividad para sustituir al dióxido de silicio como óxido de puerta y disminuir así las corrientes de fuga. La introducción de nuevos materiales en el canal de los transistores supone un cambio mucho más drástico. Actualmente

las mejoras en la movilidad se obtienen mediante la aplicación de tensión al silicio del canal, bien inducida en el proceso de fabricación [TAA+04], bien mediante ingeniería del sustrato [RKH+01]. Desafortunadamente, este beneficio es limitado y disminuye con el escalado [SKK+07], por lo que existe un gran interés en la introducción de otros materiales para alcanzar movilidades todavía mayores.

La introducción de nuevos materiales de alta movilidad como el germanio para MOSFETs tipo p [SCB+04] y semiconductores III–V para MOSFETs tipo n [Dat07] y su integración en la plataforma del silicio es, actualmente, una solución ampliamente reconocida.

Hay tres problemas principales que dificultan la introducción de los transistores III–V en la tecnología CMOS [SKK+07]:

1. El más importante es que, a diferencia del silicio, no existe una buena solución para el óxido de puerta. Pese a los progresos de los últimos años [Pas05, Dat07, SKK+07], sigue siendo un problema para la implementación de aplicaciones CMOS.

2. Los sustratos de materiales III–V son caros, frágiles y difíciles de hacer en un tamaño grande. Además, el éxito de una tecnología no basada en silicio depende de su compatibilidad con los procesos utilizados para este material. Por este motivo hacen falta métodos para integrar los nuevos materiales sobre sustratos de silicio.

3. La movilidad de los huecos en los materiales III–V no es mejor que la del silicio. Hace falta un esquema de integración que permita combinar, por ejemplo, transistores III–V para n–MOSFETs y de germanio para p–MOSFETs. Además, la baja densidad de estados en los materiales III–V también podría ser un problema [PKK+06].

Existen principalmente dos estructuras que se estudian para MOSFETs III–V para aplicaciones digitales: de canal superficial y de canal enterrado. La primera estructura sería equivalente a los MOSFETs de silicio convencionales, ésta requiere la formación de una interfaz semiconductor–óxido de muy alta calidad para mantener una densidad de defectos baja

cerca de la capa superficial de conducción. En la segunda, correspondiente a los MOSFETs III–V de canal enterrado, el canal suele estar entre dos capas de materiales III–V con un ancho de la banda prohibida mayor de manera que se forma un pozo cuántico.

El potencial para el escalado a dimensiones por debajo de los 20 nm de estas estructuras fue estudiado mediante simulaciones Monte Carlo. En este estudio [KAAM+08] se ve que los MOSFETs de canal superficial ofrecen una buena escalabilidad hasta longitudes de puerta del orden de los 20 nm, pero el óxido de puerta es una limitación fundamental para un mejor rendimiento de los transistores escalados. Por otro lado, en la simulación de una arquitectura con canal enterrado (*Implant Free* MOSFET), el rendimiento escala adecuadamente hasta longitudes de puerta de 15 nm si se emplea un material de canal apropiado [SGLA+09, Ben11], en caso contrario la baja densidad de estados del material de canal puede ser el factor limitante en el escalado del rendimiento.

1.3 Alternativas al MOSFET convencional basadas en tecnología SOI

En las últimas décadas se ha desarrollado una nueva tecnología que permite crear obleas de Si sobre las que se deposita una lámina de SiO_2, construyéndose así los dispositivos a partir de una capa de silicio monocristalino colocada sobre esta capa de dieléctrico. Esta tecnología se denomina Silicio sobre Aislante (SOI) y se ha convertido en una alternativa a la tecnología convencional (*Bulk*).

La finalidad de la tecnología SOI es mejorar las carencias de los dispositivos de tecnología convencional de Si y su desarrollo tiene tres causantes principales [CC03]:

1. Durante las décadas de los 70 y 80 del siglo pasado y debido a la guerra fría, existía una gran preocupación por realizar circuitos resistentes a los efectos de las radiaciones ionizantes que pudiesen operar en un hipotético escenario de guerra nuclear. La delgada capa de Si

activo minimiza el impacto de la radiación ionizante en las propiedades del dispositivo. La mayoría de la carga generada por las radiaciones ionizantes es detenida por el óxido enterrado (Buried Oxide o BOX) de forma que la corriente extra generada es muy pequeña.

2. La mejora de las prestaciones de los circuitos basados en SOI ha llevado a que muchas compañías se decidan por el cambio a esta tecnología. Por ejemplo, en aplicaciones digitales para una misma tensión de alimentación los circuitos SOI permiten mayores velocidades de operación, lo que implica un menor consumo de potencia.

3. El comportamiento de los dispositivos con longitudes de puerta inferiores a 25 nm no es el adecuado cuando son fabricados con tecnología convencional debido básicamente a los efectos de canal corto (SCE), resultando éstos cada vez más difíciles de controlar.

Desde la perspectiva de la fabricación de dispositivos, una vez superado el reto tecnológico de crear láminas de Si cristalino sobre un sustrato dieléctrico, el cambio de tecnología no ha resultado demasiado crítico ya que el diseño de los circuitos es bastante parecido al de la tecnología convencional. De hecho muchas de las estructuras que se creaban en los MOSFETs estándar durante la etapa de implementación física para asegurar el correcto aislamiento de los dispositivos y evitar efectos parásitos tales como corrientes de fuga, fotocorrientes inducidas por radiación o el *latch–up*, no son ahora necesarias debido al BOX y al aislamiento lateral dieléctrico, de forma que los chips obtenidos son más simples y compactos. La tecnología SOI permite una mayor variedad de estructuras que no eran realizables en otras tecnologías. Así, es posible integrar en un mismo chip dispositivos tan diferentes como MOSFETS, MEMS o guías de onda ópticas.

1.3.1 Ventajas de la tecnología SOI

La mayoría de los procesos de fabricación para dispositivos SOI son compatibles con los procesos estándar de la industria semiconductora y

el coste del producto final es un poco más elevado que en el caso de los basados en tecnología convencional. Esto es debido principalmente a que las obleas deben ser preprocesadas para conseguir el sustrato deseado para cada tipo de aplicación. El incremento en el gasto está en cierta medida justificado por las ventajas que la tecnología SOI presenta sobre la convencional, siendo en algunos casos la única opción para obtener ciertas estructuras o para integrar algunos componentes. Entre las principales ventajas que se pueden enumerar se encuentran las siguientes [CC03]:

- Tecnología completamente compatible con los procesos de fabricación tradicional.
- Reducción del número de pasos en los procesos de fabricación.
- Aumento, en algunos casos, del nivel de integración debido a una simplificación en los diseños de los circuitos a fabricar.
- Resistencia a las radiaciones ionizantes.
- Menor tensión de alimentación para una determinada velocidad de operación.
- Mayores velocidades de operación para una determinada tensión de alimentación.
- Mayor control sobre los efectos de canal corto.
- Reducción de capacidades parásitas.
- Mayor flexibilidad en las estructuras.
- Integración en un mismo chip de diferentes tipos de dispositivos tales como los de alta velocidad, de potencia, MEMS y elementos ópticos.
- Posibilidad de fabricar no sólo dispositivos planares sino también tridimensionales.
- Fabricación de circuitos integrados tridimensionales mediante la aplicación de sucesivos procesos de transferencia de láminas o pegado de obleas [RKWT10].

1.3.2 Clasificación de los dispositivos SOI

El dispositivo más usado en los circuitos basados en tecnología SOI continúa siendo el MOSFET convencional de puerta simple. Sin embargo, el hecho de que exista un óxido enterrado bajo la lámina de silicio y el desarrollo de nuevas técnicas para la obtención de estructuras SOI han dado lugar a la aparición de una gran cantidad de dispositivos nuevos que amplían las posibilidades a la hora de elegir el dispositivo adecuado para cada aplicación. Por ejemplo, sobre sustratos SOI se pueden fabricar, aparte de MOSFETs, transistores bipolares (BJT), transistores de unión de efecto campo (JFET) o diodos, e incluso se pueden integrar en el mismo chip elementos que no era posible combinar cuando eran fabricados con tecnología convencional.

A continuación haremos una clasificación de los tipos de dispositivos MOSFET atendiendo al número de puertas.

Dispositivos de una puerta

Los dispositivos de una puerta (SGSOI) son una continuación tecnológica de los MOSFET realizados con tecnología CMOS convencional, como el que se muestra en la figura 1.2.a. Podemos hacer una clasificación de este tipo de dispositivos en función de que exista o no una zona neutra debajo de la zona de canal. De este modo diferenciaremos dos tipos de dispositivos, los parcialmente deplexionados y los completamente deplexionados:

- Dispositivos parcialmente deplexionados (*Partially Depleted*, PD). El comportamiento de este tipo de dispositivos es similar al de los MOSFET convencionales, con las ventajas que añade el aislamiento total que introduce la capa de óxido enterrado. Sin embargo, la existencia de una zona neutra puede traer ciertos problemas, como por ejemplo la aparición del denominado *floating body* [SF94, Col04] que cambia la tensión umbral del dispositivo dependiendo de la historia del mismo (proceso de histéresis), los efectos causados por el transistor bipolar parásito formado por las zonas de drenador, zona neutra

y fuente o el denominado efecto *kink* para altas polarizaciones entre drenador y fuente. Un modo de reducir estos efectos consiste en realizar una conexión entre la puerta y el sustrato del transistor. Con esta técnica se obtiene el denominado MOS Bipolar Controlado por Tensión (VCBM) o MOS de voltaje umbral dinámico (DTMOS), cuyas características son un aprovechamiento del transistor bipolar parásito y una disminución de la tensión umbral con la tensión de puerta aplicada, obteniéndose comportamientos casi ideales en el régimen sub–umbral y una drástica reducción de los efectos de *floating body*.

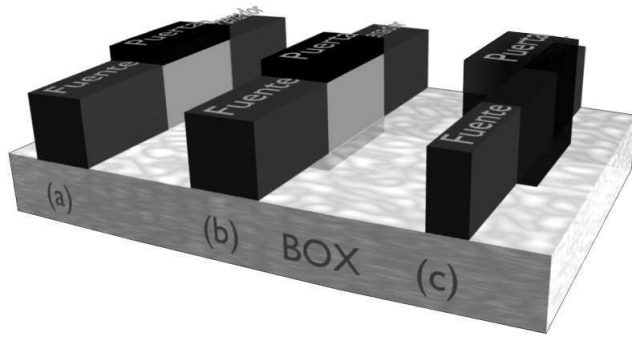


Figura 1.2: Representación esquemática de dispositivos SOI. (a) SGSOI, (b) DGSOI, (c) FinFET.

- Completamente deplexionados (*Fully Depleted*, FD). Cuando se disminuye el espesor de la lámina de silicio, la zona neutra se va reduciendo de forma que puede llegar a desaparecer. En ese momento el sustrato pasa a estar completamente deplexionado (ocupando el canal toda la zona de silicio situada entre el óxido de puerta y el BOX) desapareciendo así los efectos relacionados con el *floating body*. El comportamiento de estos dispositivos se acerca al del caso ideal, aunque existe un problema desde el punto de vista práctico. La

carga en inversión depende no sólo de la polarización sino también del espesor de la lámina de silicio, de forma que las fluctuaciones aparecidas en ésta inducen variaciones en la tensión umbral. Este fenómeno resulta crítico en dispositivos ultradelgados (UTB).

Dispositivos multipuerta

Históricamente, en el diseño de transistores MOSFET ha existido un esfuerzo constante para continuar escalando los transistores sin perder el control que la puerta ejerce sobre la carga. Sin embargo, la disminución de la longitud del canal produce los denominados efectos de canal corto (SCE) que, básicamente, consisten en la pérdida del control de la carga por el terminal de puerta. Como solución a estos problemas, surgieron los dispositivos multipuerta [Col07] de los que hablaremos a continuación.

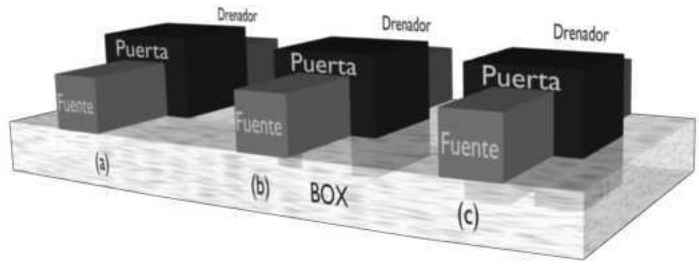


Figura 1.3: Representación esquemática de dispositivos SOI. (a) Triple puerta, (b) Puerta Π, (c) Puerta Ω.

El primer dispositivo que se propuso basado en este concepto fue el XMOS [SH84], representado en la figura 1.2.b. Este doble puerta (DGSOI), que debía su nombre a la semejanza con la letra Ξ, presentaba un mejor control sobre la carga en el canal que su equivalente con una única puerta. El primer doble puerta que se fabricó denominado DELTA [HKKT89], era un dispositivo vertical y fue el predecesor del FinFET, representado en la

figura 1.2.c, cuya fabricación es más sencilla que la de un dispositivo de canal horizontal.

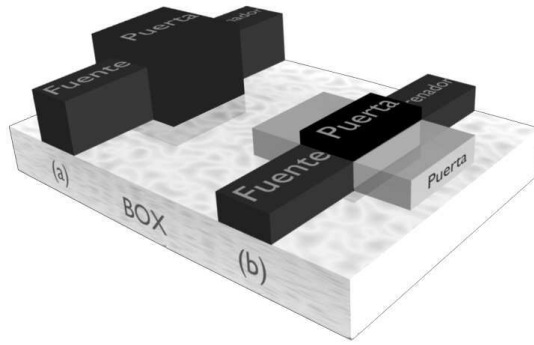


Figura 1.4: Representación esquemática de dispositivos SOI. (a) GAA, (b) G^4FET.

Cuando se comprobó que la inclusión de una segunda puerta era una mejora del dispositivo, el siguiente paso fue aumentar el número de puertas de forma que el control sobre la carga en el canal fuera mayor y de este modo se pudieran reducir los efectos de canal corto (SCE) y aumentar la corriente. Siguiendo esta iniciativa surgieron los dispositivos de triple puerta (*trigates*) en los que sobre una isla fina y estrecha de silicio cristalino se coloca una puerta sobre tres de sus lados quedando el canal completamente rodeado por capas de óxido, tal y como se representa en la figura 1.3.a. Existen también versiones más sofisticadas de estos dispositivos que se sitúan entre los de tres y cuatro puertas. Son los denominados *triple+* y entre ellos se encuentran los puerta Π y los puerta Ω, representados en las figuras 1.3.b y 1.3.c respectivamente. En el primero de ellos la puerta penetra en la zona de óxido enterrado (BOX) proporcionando un mayor control de la carga. En el segundo, la puerta además de penetrar en el BOX se extiende lateralmente bajo la lámina de silicio sin llegar a cerrarse. Esta configuración permite, en aquellos casos en los que el dispositivo sea lo suficientemente estrecho, la aparición de una puerta bajo la isla

de silicio inducida por el campo creado en las extensiones inferiores de la puerta [Col04].

Finalmente tenemos los dispositivos en los que la lámina de silicio se encuentra completamente rodeada por estructuras destinadas al control de la carga en su interior, son los denominados dispositivos de cuatro puertas. Una de las estructuras propuestas para la fabricación de estos dispositivos consiste en recubrir completamente la zona central de la lámina de silicio cristalino con puerta. Estas estructuras se reducen al concepto de hilos cuánticos [BCB+87] cuando el espesor y la profundidad del canal son lo suficientemente pequeños. En la figura 1.4.a se muestra el denominado *gate–all–around* MOSFET (GAA), que consiste en un dispositivo horizontal de sección rectangular completamente rodeado por la puerta. La figura 1.4.b consiste en un dispositivo con dos puertas MOS en los lados superior e inferior y dos JFET en los laterales. Operando normalmente en acumulación, las puertas laterales controlan el ancho efectivo del canal pudiéndose obtener distintos modos de operación que van desde el de un hilo cuántico rodeado por regiones de vaciamiento al de un dispositivo en fuerte acumulación en función de la tensión aplicada a cada una de las puertas. La reducción en las dimensiones de la estructura, especialmente en el espesor de la lámina de silicio, provoca que ciertos fenómenos de naturaleza cuántica predominen a la hora de explicar el comportamiento de estos dispositivos.

CAPÍTULO 2

Simulación Monte Carlo de Transistores MOSFET

La simulación de dispositivos semiconductores es una de las herramientas más importantes en el diseño de nuevos dispositivos. Los retos tecnológicos a los que se enfrenta la industria de semiconductores, en sectores tan diversos como la computación, las telecomunicaciones, la instrumentación electrónica o las células solares, son abordables gracias a estas técnicas de simulación. Estas herramientas permiten prever el comportamiento de nuevas estructuras y seleccionar, a priori, el diseño más favorable para afrontar el problema tecnológico bajo estudio. Posteriormente, los resultados obtenidos por estos programas pueden ser contrastados con los datos experimentales obtenidos en la caracterización de los dispositivos de prueba. Gracias a la simulación es posible seleccionar, entre un amplio rango de diseños, aquel que mejor puede satisfacer nuestros requisitos tecnológicos, consiguiendo de esta forma ahorrar tiempo y dinero en la fabricación de los posibles dispositivos objeto de estudio.

La complejidad de los modelos empleados en la simulación de dispositivos semiconductores, así como los requisitos computacionales, dependen en gran medida del reto tecnológico que queremos superar junto con el

nivel de detalle con el que se desea simular los fenómenos físicos que tienen lugar en el dispositivo estudiado. Este capítulo lo iniciamos con una introducción a las diferentes técnicas de simulación de dispositivos semiconductores haciendo hincapié en su utilidad y complejidad computacional. Posteriormente, describimos de forma general el método Monte Carlo, por ser el método que emplean los simuladores en los que se ha centrado este trabajo. Para finalizar detallamos las características específicas de los simuladores Monte Carlo paralelizados.

2.1 Técnicas de simulación

La figura 2.1 muestra un ciclo estándar del proceso de desarrollo de un dispositivo electrónico, donde se especifica de forma esquemática cada uno de los niveles del diseño tecnológico asistido por computador (TCAD). Estas herramientas de diseño se impulsaron con el incremento de la demanda comercial de componentes electrónicos que requería una mejora de los productos fabricados y un ahorro de costes. En general podría considerarse la siguiente dinámica de funcionamiento: inicialmente las necesidades del mercado demandan un dispositivo con unas características determinadas del que se simula su proceso de fabricación, a continuación se realiza la simulación del dispositivo de la cual se extraen sus características de salida (tensiones y corrientes de funcionamiento) así como los parámetros de pequeña señal. El siguiente paso es realizar la extracción de parámetros necesarios para la evaluación del transistor a nivel de circuito. Con los resultados obtenidos evaluamos si cumple las necesidades comerciales que estábamos buscando y en caso de que no lo haga el proceso se repite de nuevo modificando ciertos parámetros de fabricación.

Si nos centramos en la etapa de simulación de dispositivos, en la actualidad existe un conjunto de técnicas que pueden ser utilizadas en esta etapa del proceso diseño [Rav98]. En la figura 2.2 presentamos estas técnicas en función de dos parámetros, la complejidad computacional y el tiempo de simulación.

Atendiendo a la complejidad computacional, la técnica menos exigente

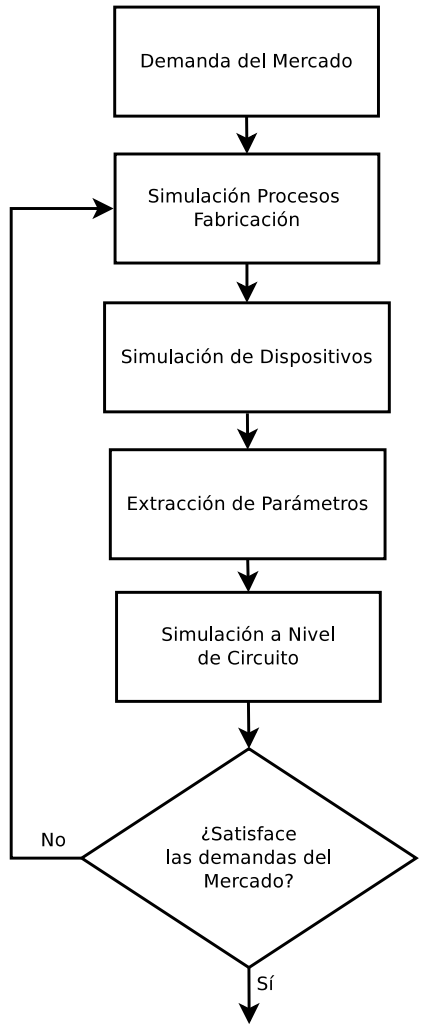


Figura 2.1: Representación esquemática de los niveles TCAD que intervienen en el desarrollo de un dispositivo electrónico.

es la empleada por los modelos compactos. Ésta contiene muy poca información sobre la física presente en el sistema y en general imita el comportamiento del dispositivo utilizando aproximaciones analíticas y parámetros

empíricos. Estos modelos requieren poco tiempo de cálculo pero su validez es limitada puesto que ignoran la naturaleza distribuida de los parámetros y otras cuestiones como podría ser la geometría del dispositivo.

En el siguiente nivel se encuentra la aproximación de arrastre–difusión (DD, *drift-diffusion*) a la ecuación de transporte de Boltzmann (BTE) [Sel84]. La aproximación DD considera sólo los primeros dos momentos de la BTE, la ecuación de continuidad de corriente y la ecuación de conservación del momento. Estas ecuaciones están acopladas a la ecuación de Poisson por el potencial electrostático. La aproximación DD incluye una relación local entre la velocidad y el campo eléctrico y no puede representar apropiadamente efectos de transporte fuera del equilibrio.

Por encima de la complejidad del modelo DD está la aproximación hidrodinámica a la ecuación de transporte de Boltzmann. En este caso se incluye el tercer momento de la BTE, la ecuación de conservación de la energía, lo que a su vez hace más compleja la ecuación de conservación del momento. Esto permite el tratamiento de efectos fuera del equilibrio ya que incluye una relación no local entre el campo eléctrico y la velocidad.

La técnica Monte Carlo para la resolución de la BTE se encuentra en el siguiente nivel de complejidad con respecto al modelo hidrodinámico. En esta técnica un conjunto de partículas evoluciona a través de un espacio de aceleración real y de eventos de dispersión escogidos aleatoriamente. Estos métodos requieren un elevado tiempo de computación, por lo que son utilizados principalmente para dispositivos de dimensiones muy pequeñas en los que los modelos de arrastre–difusión no son válidos.

Finalmente, la técnica más compleja es la empleada por las aproximaciones de transporte cuántico que utilizan las ecuaciones acopladas de Poisson y de Schrödinger independiente del tiempo, la matriz de densidad o la función de distribución Wigner. Todas ellas son extremadamente costosas computacionalmente. Otro método que está ganando popularidad es el uso del formalismo de las funciones de Green fuera del equilibrio (NEGF *Non–Equilibrium Green Functions*), que permite la inclusión de la dispersión en la formulación del transporte cuántico.

A continuación, en las siguientes subsecciones, describiremos con más

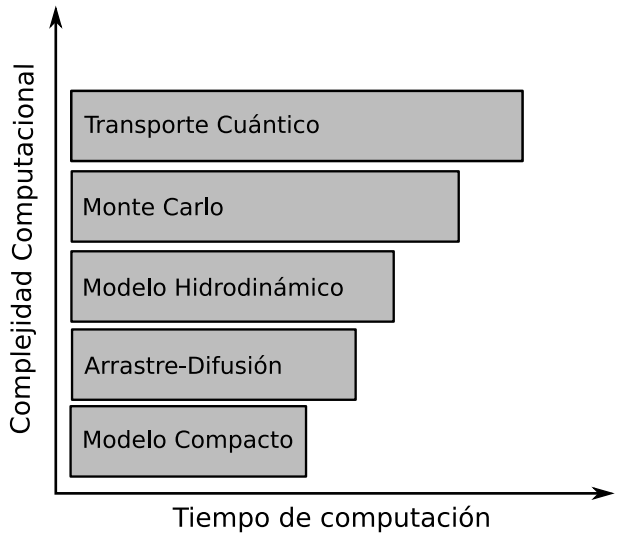


Figura 2.2: Representación de las técnicas de simulación en función de la complejidad y el tiempo de ejecución requerido.

detalle estas técnicas numéricas de simulación multidimensional.

2.1.1 Arrastre–difusión

El modelo de arrastre–difusión es el más simple empleado en simulaciones numéricas multidimensionales [ZWK+00, GLKA05, GLAS+09]. Utiliza los dos momentos más bajos de la ecuación de transporte de Boltzmann. Las densidades de corriente de electrones J_n y huecos J_p se obtienen por medio de la suma de dos componentes, una de arrastre gobernada por el campo eléctrico y otra de difusión dirigida por el gradiente de la densidad de portadores.

Esta aproximación no tiene en cuenta la temperatura de los portadores y de forma estricta es solamente válida para campos en los que la velocidad de los portadores esté directamente relacionada con el campo eléctrico. Sin embargo, la validez de la aproximación de arrastre–difusión puede extenderse empíricamente introduciendo modelos de movilidad de-

pendientes del campo eléctrico, lo que permite el uso de campos eléctricos más elevados en el proceso de simulación. Aun incluso con esta extensión el modelo arrastre–difusión sólo funciona en el cuasi–equilibrio, donde el campo eléctrico varía lentamente y la velocidad está relacionada localmente con el campo. La principal ventaja de esta aproximación es su menor coste computacional en comparación con las otras técnicas de simulación.

2.1.2 Hidrodinámico

Con el escalado de los dispositivos a dimensiones muy reducidas se produce una disparidad entre el rápido descenso de las dimensiones físicas y la mucho más lenta reducción de la tensión aplicada. Esto provoca un aumento del campo en el interior de los dispositivos. Para tener en cuenta estos campos elevados el modelo de arrastre–difusión, explicado en el apartado anterior, introduce modelos de movilidad dependientes del campo que relacionan de un modo local la velocidad de los portadores con la componente del campo en la dirección del flujo de corriente. Sin embargo, esta aproximación ignora fenómenos de transporte no locales en los que la velocidad de los portadores en un punto está determinada por la distribución del campo a lo largo del camino de corriente.

Una aproximación que supera estas limitaciones es el modelo hidrodinámico [Gar94, ALD96] al considerar el tercer momento de la ecuación de transporte de Boltzmann. En este caso la corriente tiene un término adicional proporcional al gradiente de la temperatura de los electrones y además se añade al sistema una ecuación de balance energético. Uno de los principales inconvenientes de esta aproximación es la estabilidad numérica de la solución debido a la inclusión de los momentos más elevados de la BTE, lo que implica un aumento considerable de la complejidad computacional.

2.1.3 Monte Carlo

Un método alternativo para simular el transporte de portadores, que no implica la discretización de la BTE o de sus momentos, es la apro-

ximación Monte Carlo (MC) [JR83, JL89]. Este método es una técnica estocástica que utiliza números aleatorios para obtener una aproximación estadística a la solución exacta de la BTE. Se basa en trazar las trayectorias clásicas de los portadores en un dispositivo simulado, considerando la dispersión de cada partícula después de un período de vuelo libre determinado estocásticamente a través de las tasas de dispersión acumulativas. Estas tasas de dispersión se calculan cuánticamente e incluyen, entre otras, interacciones electrón–fonón, electrón–impureza y electrón–electrón.

Una simulación Monte Carlo es por tanto una serie de vuelos libres intercalados con eventos de dispersión que cambian el momento y posiblemente la energía de la partículas. El movimiento de las partículas está acoplado a la solución de la ecuación de Poisson para permitir el cálculo actualizado de la fuerza que las dirige. La aproximación Monte Carlo es ampliamente usada en la simulación de dispositivos semiconductores [FSVF02, BAY$^+$03, SGG$^+$10], aunque el gran número de partículas que deben ser simuladas, la enorme cantidad de números aleatorios y el acoplamiento con la ecuación de Poisson hacen que este método sea muy costoso computacionalmente.

2.1.4 Transporte cuántico

La modelización del transporte dentro de un sólido mediante una aproximación cuántica completa es muy costosa computacionalmente. Una técnica que está ganando popularidad en el campo del transporte cuántico es la aproximación de las funciones de Green fuera del equilibrio (NEGF) [Dat00, MSA$^+$05, MSB$^+$10]. Utiliza un método matemático conocido como Funciones de Green para obtener la solución de un Hamiltoniano independiente del tiempo. La función de Green, a una energía dada, tiene dos entradas que pueden ser relacionadas con dos posiciones del espacio real permitiendo simular áreas de un transistor. Esta función considera la influencia de una perturbación que tiene lugar en una entrada sobre la otra entrada, y tiene, en teoría, la habilidad de modelar las propiedades físicas del sistema, tales como la densidad electrónica, la densidad de corriente y la densidad de estados. Una de las principales limitaciones del uso de

NEGF es su elevadísimo coste computacional.

Como acabamos de comentar, las simulaciones completamente mecanocuánticas son muy elevadas en términos de tiempo computacional, pero es posible incluir efectos cuánticos en simulaciones clásicas mediante correcciones cuánticas [AI89] con un coste computacional mucho menor. Estas correcciones permiten considerar efectos de confinamiento cuántico y ciertos aspectos del efecto túnel, que son imprescindibles cuando se produce un escalado agresivo de los dispositivos a dimensiones nanométricas. Los dos métodos más conocidos para incluir correcciones cuánticas en simulaciones clásicas de dispositivos son las aproximaciones *density gradient* [WSF01] y potencial efectivo [FAV00].

La aproximación *density gradient* introduce un potencial cuántico que da lugar a un término adicional de arrastre a la expresión de la densidad de corriente. Este potencial cuántico es proporcional a la segunda derivada de la densidad de portadores y disminuye las variaciones de los electrones comparado con el caso del potencial clásico, además de reducir la concentración cerca de la interfaz.

La técnica del potencial efectivo representa a los portadores por medio de un paquete de ondas gaussiano de dispersión mínima. El potencial efectivo está relacionado con el potencial a través de una integral de convolución. El suavizado del potencial asociado con la operación de convolución representa los efectos mecánico–cuánticos que alejan a la concentración de portadores de la interfaz y reducen picos bruscos debidos al uso del potencial clásico.

2.2 Fundamentos del método Monte Carlo

El método Monte Carlo es una técnica numérica que trata de resolver la ecuación de transporte de Boltzmann. La teoría de transporte es de naturaleza semiclásica. De hecho, se considera que los electrones se mueven de acuerdo a las leyes de la mecánica clásica entre dos dispersiones que ocurren sucesivamente, mientras que las secciones de dispersión de las imperfecciones del cristal se deducen de la teoría cuántica de la dispersión.

La magnitud fundamental de la teoría clásica del transporte es la función de distribución f($\vec{r}$, $\vec{k}$, t), que se define como la probabilidad de encontrar un electrón en la posición $\vec{r}$ con momento $\vec{k}$ en un determinado instante temporal t. La constante de normalización se puede elegir de forma que

$$2/\left(2\pi\right)^3 \int d\vec{k} \int d\vec{r}\, f\left(\vec{r},\vec{k},t\right) = N \tag{2.1}$$

en donde N es el número total de electrones.

La ecuación que describe el transporte en la aproximación semiclásica es la ecuación de Boltzmann,

$$\partial f/\partial t + \vec{v}\cdot\nabla_{\vec{r}} f + \dot{\vec{k}}\cdot\nabla_{\vec{k}} f = \partial f/\partial t|_{coll} \tag{2.2}$$

en donde el lado derecho de la ecuación indica las variaciones de la función de distribución debido a las colisiones.

La ecuación de Boltzmann se puede obtener del teorema de Liouville. Un modo fundamental para obtener la ecuación se basa en considerar un pequeño volumen en el espacio de fases y contar el número de partículas en movimiento que entran y salen del volumen por unidad de tiempo.

Una vez determinada la función de distribución, podemos obtener aquellas magnitudes que son de interés, tales como la velocidad de arrastre de los electrones, el promedio de energía o el coeficiente de difusión, como funciones del campo aplicado, la temperatura o el gradiente de la concentración de electrones.

En general si se aplica un campo electromagnético externo a un semiconductor, los electrones cambiarán continuamente su estado $\vec{k}$ de acuerdo con la siguiente ley

$$\hbar\dot{\vec{k}} = -e\left(\vec{\varepsilon} + \frac{1}{c}\vec{v}\times\vec{B}\right) \tag{2.3}$$

en donde $-e$ es la carga del electrón, $\vec{\varepsilon}$ y $\vec{B}$ son los campos eléctrico y magnético, respectivamente, y

$$\vec{v} = \frac{1}{\hbar}\frac{\partial E}{\partial \vec{k}} \tag{2.4}$$

es la velocidad de grupo del electrón.

Del análisis de la ecuación de Boltzmann y de las ecuaciones 2.3 y 2.4 se deduce que el conocimiento básico que se requiere para el estudio de cualquier fenómeno de transporte está relacionado con la estructura de bandas $E\left(\vec{k}\right)$ del material específico que queremos estudiar.

De la mecánica estadística sabemos que la función de distribución en el equilibrio f_0 es la función de distribución de Fermi–Dirac. Por lo tanto, para $E - E_F \gg K_B T$ (en donde ε_F es el nivel de Fermi) f_0 se puede reemplazar por la distribución de Maxwell

$$f_{MB} = C \, \exp\left(-E/K_B T\right) \tag{2.5}$$

en donde C es la constante de normalización.

El termino de colisión situado en el lado derecho de la ecuación 2.2 es debido a que la existencia de imperfecciones en la red cristalina ideal, incluyendo la existencia de fonones, puede inducir transiciones entre diferentes estados de Bloch. Si consideramos la probabilidad de transición por unidad de tiempo de un estado $\vec{k}$ a otro estado $\vec{k}'$ como $P(\vec{k}, \vec{k}')$, inducido por las imperfecciones de la red (asumiendo que no existen cambios en el espín causados por las transiciones), el término de colisión se puede expresar como la diferencia entre los electrones dispersados al estado $\vec{k}$ y los electrones dispersados al estado $\vec{k}'$:

$$\begin{aligned} &\left(\frac{\partial f}{\partial t}\right)_{coll} = \\ &= \frac{V}{(2\pi)^3} \int \left[f(\vec{k}') P(\vec{k}', \vec{k}) \left(1 - f(\vec{k})\right) - f(\vec{k}) P(\vec{k}, \vec{k}') \left(1 - f(\vec{k}')\right) \right] d\vec{k}' \end{aligned} \tag{2.6}$$

en donde V representa el volumen y los coeficientes $(1 - f)$ se tienen en cuenta para considerar el principio de exclusión de Pauli. Sin embargo, para la aproximación que hemos hecho en la ecuación 2.5 estos factores no contribuyen, por lo tanto asumiremos siempre que $f \ll 1$. Si ahora sustituimos la ecuación 2.6 en la ecuación de Boltzmann, obtenemos una ecuación integro diferencial. La complejidad de esta ecuación depende de los modelos usados en los mecanismos de dispersión y de la estructura de bandas.

Encontrar una solución a la ecuación de Boltzmann no es una tarea sencilla. Incluso en el caso de respuesta lineal, con mecanismos simples de dispersión, resulta necesario emplear aproximaciones.

Desde el punto de vista analítico, los fenómenos de transporte en regímenes no lineales quedarían completamente descritos por la ecuación de Boltzmann, pero la resolución de esta ecuación es un problema matemático de difícil resolución. Sin embargo, las técnicas analíticas aplicadas a modelos sencillos de semiconductores nos permiten obtener una interpretación física del problema de transporte no lineal, mediante la introducción de conceptos simples tales como los tiempos de relajación de la energía y el momento, y la temperatura del electrón. Por otro lado, para conseguir una solución analítica que describa el problema con detalle, resulta necesario desarrollar aproximaciones drásticas que no dejan suficientemente claro si las propiedades de interés que se extraen de los resultados se deben al modelo microscópico o a las aproximaciones matemáticas.

Sin duda, la introducción de técnicas numéricas ha supuesto un gran avance en la resolución de la ecuación de Boltzmann. Dos técnicas importantes de resolución son la técnica iterativa y el método Monte Carlo.

La técnica iterativa alcanza la solución de la ecuación de Boltzmann por medio de un procedimiento iterativo y procesado de la función de distribución para cada paso del procedimiento. Por esta razón resulta una técnica útil cuando tratamos con fenómenos físicos que dependen fuertemente de la función de distribución.

El método Monte Carlo se puede ver como una técnica de simulación directa de la dinámica de los electrones de carga dentro del cristal, lo que nos permite extraer cualquier información física mientras se desarrolla la solución de la ecuación de transporte.

Este método, aplicado al transporte de carga en semiconductores, consiste en la simulación del movimiento de uno o más electrones dentro del cristal sometidos a la acción de fuerzas externas debidas a los campos eléctricos y magnéticos aplicados y a los mecanismos de dispersión. La duración de los vuelos libres entre dos colisiones sucesivas y los fenómenos de dispersión implicados en la simulación se seleccionan de modo estocástico,

de acuerdo con la probabilidad de dispersión que describe un determinado proceso microscópico. Por lo tanto, el método Monte Carlo depende de la generación de una secuencia de números aleatorios.

Cuando el propósito de la simulación es el estudio del estado estacionario o un fenómeno homogéneo, la simulación del movimiento de un único electrón resulta suficiente. Si consideramos el principio de ergodicidad podemos asumir que obtendremos la información del comportamiento del gas de electrones si realizamos una simulación lo suficientemente larga del movimiento del electrón que hemos tomado como muestra. En este caso la simulación se denomina Monte Carlo de una única partícula (*Single Particle Monte Carlo, SPMC*). Cuando el proceso de transporte no es homogéneo, o estacionario, resulta necesario simular un número muy grande de electrones y realizar un seguimiento de su dinámica para obtener la información que deseamos. En este caso la simulación se denomina Monte Carlo de un conjunto de partículas (*Ensemble Monte Carlo, EMC*).

El simulador Monte Carlo con el que realizamos este trabajo es de tipo *EMC*. Por simplicidad, vamos a presentar primero el funcionamiento de un *SPMC* para después extender estos conceptos al funcionamiento de un *EMC*. El caso más sencillo que nos permite describir el simulador Monte Carlo de forma general, es el que describe el movimiento de los electrones en un semiconductor simple sometido a un campo eléctrico externo $\vec{\varepsilon}$. La simulación comienza con un electrón de vector de onda $\vec{k}_0$ y unas determinadas condiciones iniciales. A continuación se elige la duración del primer vuelo en función de una distribución de probabilidad determinada por las probabilidades de dispersión. Durante el vuelo las fuerzas externas actúan de acuerdo con relación

$$\hbar\dot{\vec{k}} = -e\vec{\varepsilon} \tag{2.7}$$

Durante esta parte de la simulación se guardan todas las cantidades físicas de interés, tales como la velocidad, energía, etc. La finalización del vuelo se determina mediante la elección de un mecanismo de dispersión de acuerdo con las probabilidades relativas de todos los posibles mecanismos de dispersión. De la sección eficaz diferencial de este mecanismo se elige de

modo aleatorio un nuevo estado $\vec{k}$ como el estado inicial del nuevo vuelo libre. Todo el proceso se repite de forma iterativa, de tal modo que los resultados se vuelven más y más precisos cuando la simulación avanza. Este proceso termina cuando las cantidades de interés alcanzan la precisión que deseamos. En la figura 2.3 se representa el diagrama de flujo genérico de un simulador MC. A continuación trataremos con más detalle cada una de las etapas del proceso de simulación.

2.2.1 Definición del sistema físico

Si nos fijamos en el diagrama de flujo de la figura 2.3 podemos observar que la definición del sistema físico es el punto de partida del simulador incluyendo los parámetros del material y los valores de las magnitudes físicas, tales como temperatura de la red cristalina y campo eléctrico.

También resulta posible en este nivel definir los parámetros que controlan la simulación, tales como la duración de la misma o la precisión de los resultados.

El siguiente paso en el programa es realizar el cálculo de cada probabilidad de dispersión en función de la energía del electrón e inicializar las cantidades acumulativas a cero en este punto del programa.

2.2.2 Condiciones iniciales

Inicialmente, si consideramos la simulación de un estado estacionario, el tiempo total de simulación debe ser lo suficientemente largo como para que las condiciones iniciales del movimiento del electrón no influyan en el resultado final. La elección del tiempo de simulación es un compromiso entre el principio de ergodicidad ($t \to \infty$) y la necesidad de minimizar el tiempo de computación. Por ejemplo, cuando se elige un valor inicial del vector de onda $\vec{k}$ que resulta muy poco probable, la primera parte de la simulación puede estar muy influenciada por la inapropiada elección inicial. Cuando aplicamos un campo eléctrico muy alto y le damos al electrón una energía inicial del orden de $K_B T$, esta energía será mucho menor que el promedio de la energía bajo condiciones de estado estacionario, y durante

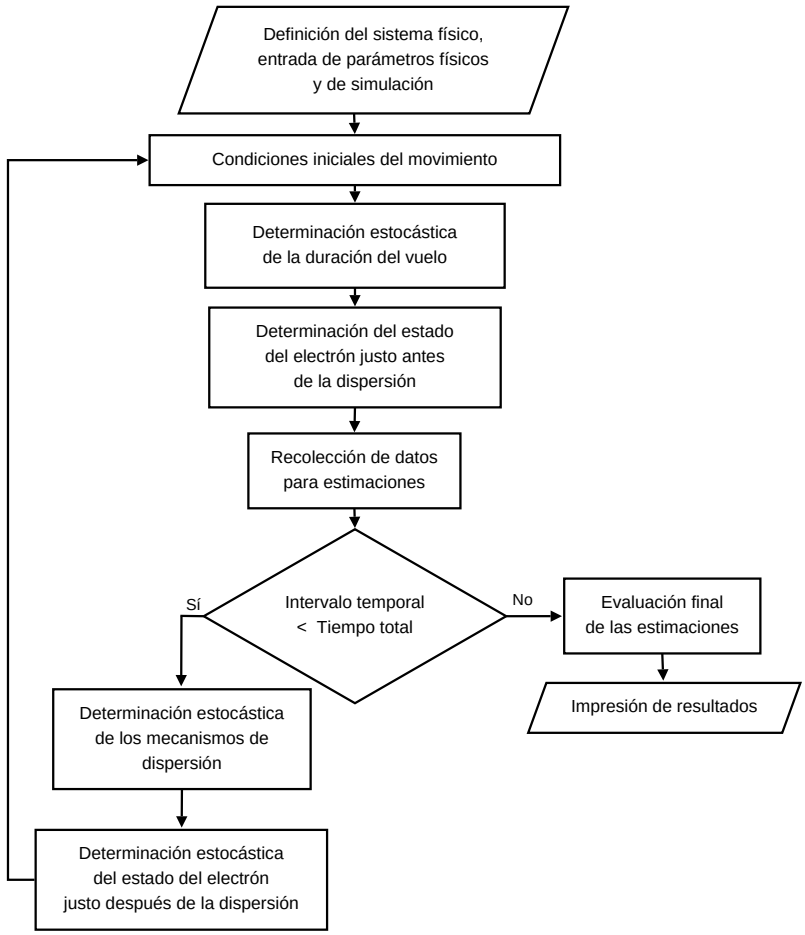


Figura 2.3: Diagrama de flujo genérico de un simulador Monte Carlo.

la simulación se irá incrementando hasta alcanzar el valor que le correspondería en el estado estacionario. Como consecuencia, la respuesta del electrón al campo eléctrico, en términos de movilidad, puede ser inicialmente muy superior a la que le correspondería en el estado estacionario.

Aumentar el tiempo de simulación disminuirá la influencia que tendrán las condiciones iniciales en el promedio de los resultados. Sin embargo

puede resultar ventajoso eliminar de la estadística la primera parte de la simulación, con la intención de minimizar los efectos indeseados originados por una elección inicial inapropiada, y así conseguir mejor convergencia con los resultados estacionarios. En los casos en los que la simulación se divide en muchas simulaciones más pequeñas, es posible que los resultados obtenidos converjan mejor a los valores estacionarios si tomamos como estado inicial de cada simulación nueva el estado final de la simulación anterior. De esta forma, tan sólo las condiciones iniciales de la primera simulación tendrán influencia en los resultados finales.

La situación opuesta a la simulación de un estado estacionario surge cuando tratamos de simular un estado transitorio o el transporte en un sistema inhomogéneo, tal y como ocurre en el simulador que estudiamos. En este caso resulta necesario simular muchos electrones por separado, por lo que, debemos considerar la distribución inicial de estados de los electrones y el transitorio inicial se convierte en una parte esencial del problema que tenemos que resolver.

2.2.3 El vuelo libre. Autodispersión

El vector de onda $\vec{k}$ del electrón cambia continuamente durante el vuelo libre debido al campo eléctrico aplicado de acuerdo con la ecuación 2.7. Si $P\left[\vec{k}\,(t)\right] dt$ es la probabilidad de que un electrón en un estado $\vec{k}$ sufra una colisión durante un intervalo de tiempo dt, la probabilidad de que un electrón que haya tenido una colisión en $t = 0$ no haya sufrido otra colisión después de un tiempo t es

$$exp\left[-\int_0^t P\left[\vec{k}\,(t')\right] dt'\right] \tag{2.8}$$

Consecuentemente, la probabilidad $\mathcal{P}\,(t)$ de que el electrón sufra un nueva dispersión durante un tiempo dt en torno al tiempo t viene dada por la expresión

$$\mathcal{P}\,(t)\,dt = P\left[\vec{k}\,(t)\right] exp\left[-\int_0^t P\left[\vec{k}\,(t')\right] dt'\right] dt \tag{2.9}$$

Debido a la complejidad del término del exponente de la ecuación 2.9, no resulta demasiado práctico generar vuelos libres de forma estocástica a partir de esta expresión, ya que necesitaríamos resolver una ecuación integral para cada dispersión. Un modo simple de superar esta dificultad es considerar que $\Gamma \equiv 1/\tau_0$ es el máximo valor que toma $P\left[\vec{k}\right]$ en la región de interés en el espacio $\vec{k}$. De esta forma se introduce un nuevo concepto ficticio llamado "autodispersión" tal que la probabilidad total de dispersión, incluyendo esta autodispersión, es constante e igual a Γ. Si el electrón sufre una autodispersión, su estado $\vec{k'}$ después de la colisión es igual al estado $\vec{k}$ que tenía antes de la colisión, por lo que en la práctica el estado del electrón sigue siendo el mismo, como si no hubiera sufrido ninguna dispersión.

Con este método, $P(\vec{k}) = \tau_0^{-1}$ es constante y la ecuación 2.9 queda de la forma

$$\mathcal{P}(t) = \frac{1}{\tau_0} exp\left[-t/\tau_0\right] \tag{2.10}$$

Para obtener el tiempo de vuelo libre a partir de la ecuación 2.10 empleamos una aproximación matemática denominada técnica directa [Ham64, BGS+66]. Esta técnica consiste en considerar una función de distribución $f(x)$ normalizada a uno en el intervalo de definición (a,b) y una función $F(x)$ que representa la integral de f. Entonces, a partir de un número aleatorio r uniformemente distribuido en un intervalo (0,1), podemos obtener un x_r tal que

$$r = F(x_r) = \int_a^{x_r} f(x)dx \tag{2.11}$$

La probabilidad $P(x)dx$ de que el valor de x_r que hemos obtenido de esta forma se encuentre dentro del intervalo dx en torno a x es igual a dF, ya que r es una distribución plana.

$$P(x)\,dx \;=\; dF \;=\; f(x)dx \tag{2.12}$$

Así pues, si aplicamos la técnica directa sobre la distribución de probabi-

lidad de la ecuación 2.10 tenemos que,

$$r = \int_0^{t_r} \frac{1}{\tau_0} exp\left[-t/\tau_0\right] dt. \tag{2.13}$$

Resolviendo la integral de la ecuación 2.13 y despejando t_r obtenemos

$$t_r = -\tau_0 \, ln(1-r). \tag{2.14}$$

Sin embargo, como r está uniformemente distribuido entre 0 y 1, también lo estará (1–r). Así en la práctica, en lugar de la ecuación 2.14 se utiliza

$$t_r = -\tau_0 \, ln(r). \tag{2.15}$$

El tiempo de computación que se emplea en considerar los procesos de autodispersión, en general, se compensa sobradamente por la simplificación en el cálculo de la duración del vuelo libre.

Por lo que respecta a la elección de la constante Γ, debemos tener en cuenta que en general $P(\vec{k})$ es una función de la energía del electrón E. Una elección adecuada de Γ podría ser tomar el máximo valor de $P(E)$ en la región de energías que se van a emplear en la simulación. En este caso debemos considerar que el rango de energías que toma el electrón durante la simulación no se conoce, que es cuando debemos elegir el valor de Γ. Por lo tanto, se debe hacer una estimación de E_{max} teniendo en cuenta que no debemos tomar un valor demasiado grande para evitar un consumo innecesario de tiempo de computación en los procesos de autodispersión. Por otro lado, se debería planificar durante la creación del programa una alternativa por si durante la simulación se alcanza un valor mayor que E_{max}.

En el caso de los modelos de semiconductores que consideran varios valles, se debe tomar un valor diferente de E_{max} que sea apropiado a cada valle.

Existen otros métodos que nos permiten elegir el valor de Γ, pero la elección de una técnica determinada depende mucho del tipo de simulación y de los recursos computacionales de que se disponga.

2.2.4 El proceso de dispersión

Durante un vuelo libre la dinámica del electrón está determinada por la ecuación 2.7. Al final de su movimiento podemos conocer el vector de onda del electrón y su energía, así como calcular todas las probabilidades de dispersión $P_i(E)$, en donde i indica el mecanismo i–ésimo de dispersión. La probabilidad de autodispersión vendrá dada entonces por el complemento en Γ de la suma de los P_i. Debemos elegir un mecanismo de todos los posibles, si consideramos

$$\Gamma = \sum_i P_i \tag{2.16}$$

la forma de hacerlo es tomando un número aleatorio r y comparando el producto $r\,\Gamma$ con las sumas sucesivas de los P_i,

$$P_1, P_1 + P_2, ..., P_1 + P_2 + ... + P_j, ..., \Gamma \tag{2.17}$$

de tal modo que se selecciona el mecanismo j–ésimo si la primera de las anteriores sumas parciales mayor que $r\,\Gamma$ es P_1+P_2+...+P_j.

Si se han probado todos los mecanismos de dispersión y no se ha seleccionado ninguno de ellos, significa que $r\Gamma > P\,(E)$, y entonces se selecciona el mecanismo de autodispersión. Por lo tanto el mecanismo de autodispersión será el que consuma mayor tiempo de computación ya que se deben calcular previamente de forma explícita todos los P_i. Sin embargo se puede hacer un proceso más corto, mediante el uso de un recurso que algunos investigadores denominan autodispersión rápida [JL89]. La idea es crear una matriz al principio de la simulación, en donde se tabulan todas las probabilidades de dispersión para cada intervalo de energía considerando los valores que puede tomar ésta. De este modo al final del vuelo, si la energía del electrón cae en el n–ésimo intervalo (P^n), antes de probar cada uno de los P_i, se compara $r\,\Gamma$ con P^n. Entonces si $r\,\Gamma > P^n$ tiene lugar un proceso de autodispersión, de lo contrario se procederá a la evaluación de todos los P_i. Por lo tanto para que tenga lugar una autodispersión que requiera la evaluación de todos los P_i debe cumplirse que $P\,(\varepsilon) < r\Gamma < P^n$.

Finalmente, debemos tener en cuenta que algunos mecanismos de dispersión pueden depender de la función de distribución en sí misma, que

se obtiene como resultado final de la simulación y por lo tanto al principio, cuando se configura el programa, no se conoce. En estos casos resulta necesario establecer un procedimiento autoconsistente, tal y como se hace en el Monte Carlo de muchas partículas (EMC). Por ejemplo, la probabilidad de dispersión asociada al mecanismo de dispersión entre electrones depende de la función de distribución de los electrones. Pero la función de distribución de portadores es uno de los resultados de la simulación en un SPMC, por lo tanto, emplearemos la salida del programa para calcular la probabilidad de dispersión que es un parámetro necesario para continuar la simulación.

Elección del estado del electrón después de la dispersión

Una vez que se ha determinado el mecanismo de dispersión que causa el final del vuelo del electrón, se debe elegir el nuevo estado que tendrá el electrón $\vec{k}_b$ después de la dispersión. Si el vuelo libre termina con una autodispersión, $\vec{k}_b$ debe ser igual a $\vec{k}_a$, el estado que tenía el electrón antes de la dispersión.

En el caso de que ocurra otro tipo de dispersión, entonces el valor de $\vec{k}_b$ se determina de forma aleatoria, considerando la sección eficaz diferencial de ese mecanismo.

2.2.5 Mecanismos de dispersión para gases 3D

Las transiciones electrónicas de interés para el transporte de carga en semiconductores se pueden clasificar como intravalle cuando los estados inicial y final permanecen en el mismo valle, o intervalle cuando las transiciones cambian de valle. Las fuentes de dispersión más importantes que determinan estas transiciones en los cristales homogéneos son los fonones, las impurezas y la dispersión con otros portadores.

La interacción de los fonones con los portadores de carga se debe a las deformaciones del cristal, producidas por los fonones a través de los mecanismos de deformación del potencial o a través de las fuerzas electrostáticas producidas por las ondas de polarización que acompañan a

los fonones. El primer tipo de interacción es típica en semiconductores covalentes. La interacción electrostática, típica en materiales polares, se conoce como interacción piezoeléctrica para el caso de los fonones acústicos y como interacción polar cuando nos referimos a los fonones ópticos. Con frecuencia se asigna directamente el atributo de polar o piezoeléctrica a los fonones, por lo tanto nos referiremos a fonones piezoeléctricos o fonones polares ópticos. Del mismo modo cuando los fonones induzcan transiciones electrónicas intervalle hablaremos de fonones intervalle.

Por lo que se refiere a las impurezas, estas pueden ser neutras o ionizadas. Cuando las impurezas son ionizadas la interacción es de tipo coulombiano, mientras que si son neutras la interacción es de más corto alcance y el efecto total de estas impurezas es más débil. Generalmente, la dispersión de Coulomb debida a las impurezas ionizadas, tan sólo debe tenerse en cuenta en transiciones intravalle, debido a que la sección eficaz de Coulomb decrece rápidamente cuando aumenta la transferencia de momento $\triangle\vec{k}$ y en las transiciones intervalle las variaciones en $\triangle\vec{k}$ son muy grandes.

Además de los mecanismos de dispersión anteriores existen otros mecanismos que se deberían tener en cuenta y que pueden ser de cierta importancia en determinadas circunstancias. Por ejemplo, la dispersión por rugosidades en la interfaz que separa dos regiones diferentes (dispersión superficial), la dispersión debida a las fluctuaciones de composición existente en aleaciones desordenadas (dispersión por aleación), la dispersión entre portadores o los procesos de pérdida y generación de portadores, tales como la ionización por impacto y la generación–recombinación.

Dispersión por fonones.

Debido a la energía térmica de la red cristalina se produce una vibración de los iones que forman la red en torno a la posición de equilibrio. El término fonones es una forma de denominar los modos normales de vibración de la red cristalina [AM76], en donde cada uno de los estados se representa por un par ($\vec{q}$, j) y está ocupado por $n_j(\vec{q})$ fonones de energía

$\hbar\omega_j(\vec{q})$. La energía de vibración viene dada entonces por la expresión

$$E = \sum_{j,\vec{q}} \hbar\omega_j(\vec{q}) \left[n_j(\vec{q}) + \frac{1}{2} \right] \tag{2.18}$$

cumpliendo $n_j(\vec{q})$ la estadística de Bose–Einstein

$$n_j(\vec{q}) = \frac{1}{e^{\frac{\hbar\omega_j(\vec{q})}{K_B T}} - 1} \tag{2.19}$$

En el caso de un cristal de base diatómica como el silicio existen 6 ramas de vibración, de las cuales 3 son acústicas (modos en los que se produce un desplazamiento neto de la celda elemental) y las 3 restantes son ópticas (modos en los que se produce un desplazamiento relativo entre los iones de la celda unitaria).

La interacción entre electrones con los iones de la red puede ser descompuesta en dos contribuciones: la interacción de los electrones con los iones en su posición de equilibrio, que describe la interacción con el potencial periódico determinando la estructura de bandas del electrón en el cristal, y las vibraciones de la red, cuya contribución se corresponde con la interacción electrón–fonón.

Si se suponen pequeños desplazamientos (α) de los iones de la red (n) con respecto de las posiciones de equilibrio, $\vec{s}_{n,\alpha}$, se puede expresar la interacción entre un electrón (e) y un ión, en primer orden de aproximación, por la expresión

$$V_\alpha\left(\vec{r}_e - \vec{R}_{n,\alpha} - \vec{s}_{n,\alpha}\right) = V_\alpha\left(\vec{r}_e - \vec{R}_{n,\alpha}\right) - \vec{s}_{n,\alpha} \cdot \nabla V_\alpha\left(\vec{r}_e - \vec{R}_{n,\alpha}\right) \tag{2.20}$$

El primer término de 2.20 sumado sobre (n,α,e) se corresponde con la interacción con los iones en equilibrio. En el segundo término tenemos la interacción electrón–fonón que podríamos expresar como,

$$H_{ep} = -\sum_{n\alpha e} \vec{s}_{n,\alpha} \cdot \nabla V_\alpha\left(\vec{r}_e - \vec{R}_{n,\alpha}\right) \tag{2.21}$$

Podemos diferenciar dos tipos fundamentales de interacción electrón–fonón: las interacciones en las que se crea un fonón con vector de onda $-\vec{q}$

(emisión) y aquellas en las que desaparece un fonón con vector de onda $\vec{q}$ (absorción). Ambos procesos están acompañados por una transición del electrón de un estado $\vec{k}$ a otro de momento $\vec{k} \pm \vec{q}$. Por lo tanto, el problema se limita al cálculo del potencial de interacción $\nabla V_\alpha \left(\vec{r}_e - \vec{R}_{n,\alpha} \right)$.

- Transiciones acústicas intravalle.

  Este tipo de transiciones son de especial interés en el caso de las interacciones de los electrones con los fonones de la rama acústica. En ella, los átomos de la celda unitaria se mueven en la misma dirección, produciéndose así un desplazamiento efectivo de la celda unitaria completa. Para el caso de longitudes de onda lo suficientemente grandes, las amplitudes de vibración cambiarán poco de una celda unitaria a otra, con lo que el papel de la estructura atómica es poco determinante para estos casos, pudiéndose realizar un paso al continuo utilizando el potencial de deformación, [Kit95], quedando el potencial de interacción electrón–fonón de la forma

  $$H_{ep} = \vec{\epsilon} \, \nabla \cdot \vec{s} \tag{2.22}$$

  $\vec{\epsilon}$ es un tensor que describe el desplazamiento de las bandas por unidad de deformación. En esta aproximación continua, el desplazamiento puede ser escrito en función de los operadores que describen la creación, y la aniquilación de fonones, $a_q^\dagger$ y a_q respectivamente. Siguiendo el desarrollo realizado en [JL89] se encuentra que, para el caso elástico utilizando la aproximación de equipartición de la energía con bandas no parabólicas, la probabilidad de dispersión por unidad de tiempo para procesos de emisión y absorción es de la forma

  $$\Gamma_{ac}(E) = \frac{\sqrt{2} m_d^{3/2} K_B T_0 \epsilon_l^2}{\pi \hbar^4 u^2 \rho} E^{1/2} \left(1 + 2\alpha E\right) \left(1 + \alpha E\right)^{1/2} \tag{2.23}$$

  en donde u es la velocidad del sonido en el medio, α el factor de no–parabolicidad, ϵ_l^2 el valor medio del tensor deformación a lo largo de la dirección de $\vec{q}$ y ρ la densidad.

- Transiciones intervalle.

  En este tipo de transiciones el estado inicial y final se encuentran situados en valles diferentes de forma que los fonones involucrados son de alto momento. El vector de onda del fonón involucrado en transiciones de este tipo permanece muy próximo a la distancia entre los mínimos de los dos valles de la transición, por tanto $\Delta\vec{k}$ es prácticamente constante y, para una rama determinada, la energía $\hbar\omega_i$ también lo es. Teniendo en cuenta estas consideraciones la dispersión intervalle puede ser tratada con el mismo formalismo que la intervalle acústica teniendo en cuenta que el potencial de deformación y la energía del fonón involucrado deben ser independientes de $\vec{q}$. Por lo tanto siguiendo el desarrollo propuesto en [JL89], la probabilidad de dispersión para orden cero y bajo la aproximación de no–parabolicidad estará determinada por la expresión

$$\Gamma_{e,i}(E) = A\binom{N_i}{N_i+1}\left(E \pm \Xi_{\mp}\right)^{1/2}\left[1+2\alpha\left(E \pm \Xi_{\mp}\right)\right]\left[1+\alpha\left(E \pm \Xi_{\mp}\right)\right]^{1/2} \tag{2.24}$$

  siendo

$$A = \frac{m_d^{3/2}\left(D_t K\right)_i^2 \gamma}{\sqrt{2}\pi\hbar^3\omega_i\rho} \tag{2.25}$$

  y

$$\Xi_{\mp} = \hbar\omega_i \mp \Delta E \tag{2.26}$$

  en donde γ representa el número de posibles valles finales equivalentes en la transición y los símbolos superior e inferior se tienen para procesos de absorción y emisión respectivamente.

  Considerando la posición de los valles en la banda de condución del silicio se observa que pueden darse dos tipos de transiciones intervalle:

  – Procesos tipo g, en los cuales se producen transiciones entre valles en la misma dirección aunque opuestos, por ejempo <100> y <$\bar{1}$00>.

– Procesos tipo f, resultantes de transiciones entre valles con diferentes direcciones como por ejemplo de la dirección <100> a la <010>.

Ambos procesos resultan ser de tipo *unklapp* e involucran a un vector de la red recíproca. La figura 2.4 representa de forma esquemática los posibles procesos que pueden ocurrir.

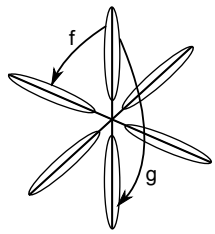


Figura 2.4: Transiciones intervalle posibles en la banda de conducción del silicio.

A partir de consideraciones geométricas sobre la zona de Brillouin se demuestra que el modo de un fonón involucrado en una transición de tipo f es igual a la distancia entre los puntos Γ y X de la zona de Brillouin del silicio, formando un ángulo de 11º con la dirección ΓX. En el caso del fonón involucrado en las transiciones de tipo g su módulo es $0{,}3\Gamma X$ en la dirección ΓX. Para estos procesos, la multiplicidad de los valles finales es 4 en el caso de las transiciones f y 1 en el caso de las de tipo g.

A partir de un análisis de la teoría de grupos se puede demostrar que, en orden cero de interacción, los procesos de tipo g sólo pueden ser asistidos por fonones de la rama longitudinal óptica (LO), mientras que para el caso de los f sólo es posible para fonones longitudinales acústicos (LA) y transversales ópticos (TO). Siguiendo este análisis, el resto de fonones quedarían prohibidos. Sin embargo, otras referencias [JL89] sugieren que, debido a que los estados inicial y final no coinciden con puntos de muy alta simetría, las reglas de selección no

tienen que cumplirse en su totalidad. Por continuidad es razonable esperar que estos procesos, que estarían prohibidos para el caso en que el estado inicial y final cayeran en puntos de alta simetría, sean muy escasos en comparación con aquellos permitidos. Los efectos de estos fonones prohibidos han sido observados de forma experimental, por lo que su modelado se realiza vía interacción de primer orden obteniendo de este modo las probabilidades de absorción y de emisión

$$\Gamma_{e,i}(E) = \frac{\sqrt{2} m_d^{5/2} D_1^2 \gamma}{\pi \hbar 4 \omega_i \rho} \binom{N_i}{N_i + 1} \left[E\left(1 + E\alpha\right) + \left(E \pm \Xi_{\mp}\right) \right.$$
$$\left. \left(1 + \alpha\left(E \pm \Xi_{\mp}\right)\right) \right] \left(1 + 2\alpha\left(E \pm \Xi_{\mp}\right)\right) \left[\left(E \pm \Xi_{\mp}\right) \left(1 + \alpha\left(E \pm \Xi_{\mp}\right)\right) \right]^{1/2} \tag{2.27}$$

Dispersión Coulombiana

En un dispositivo existen distintos tipos de carga que afectan de diferentes formas al movimiento de los portadores. A continuación se describe brevemente las distintas procedencias de estas cargas:

- Cargas atrapadas en la interfaz (Q_{it}): Son debidas a la existencia de defectos en la interfaz Si–SiO_2 y crean estados accesibles en la banda prohibida. Estos defectos pueden interactuar con las bandas de valencia y conducción capturando o emitiendo huecos y electrones respectivamente. La densidad de trampas depende de la orientación de la lámina de silicio, del estrés térmico y eléctrico al que sea sometida y de la radiación que pueda recibir la interfaz. Todos estos agentes pueden producir no sólo variación en la ocupación de las trampas, sino también la aparición de nuevos defectos y trampas que degradarán las propiedades eléctricas de la estructura.

- Cargas fijas en la interfaz (Q_f): Estas cargas, de signo positivo, se encuentran localizadas en una lámina muy delgada ($10 - 20$ Å) de óxido de silicio no estequiométrico manteniendo su posición fija independientemente de la polarización de puerta. La cantidad depende

de las condiciones de oxidación y recocido. Para interfaces de buena calidad el valor de esa carga suele ser del orden de 10^{10} cm^{-3} para superficies <100>.

- Carga atrapada en el óxido (Q_{ox}): Estas cargas están asociadas a defectos existentes en el óxido tales como impurezas o enlaces rotos y se distribuyen por todo el volumen del aislante. Al igual que las anteriores son independientes de la polarización de la puerta y pueden ser eliminadas mediante recocido a baja temperatura. Las trampas creadas por los defectos pueden cargarse a través de cargas inyectadas dentro del óxido mediante la generación de portadores calientes durante la operación del dispositivo o mediante la generación de pares electrón–hueco en el óxido de silicio en procesos de radiación por rayos X.

- Cargas de iones móviles (Q_m): Esta carga es debida principalmente a la presencia de iones de metales alcalinos tales como el sodio o el potasio. Debido a su naturaleza móvil, estas cargas son atraídas a la interfaz silicio–óxido de silicio por el campo eléctrico que se aplica a la puerta.

- Carga de las impurezas ionizadas (Q_B): Debido a la curvatura de bandas producida por la tensión aplicada en la puerta las impurezas del canal se encuentran polarizadas prácticamente en su totalidad incluso para bajas temperaturas. Esta carga tiene efectos significativos sobre la movilidad de los portadores.

Todas estas cargas se agrupan en las conocidas como cargas externas. En el caso de los simuladores paralelizados en este trabajo se considera el efecto de las impurezas ionizadas, ya que su influencia en el estudio del transporte de portadores es muy importante debido a que son las principales responsables de la dispersión Coulombiana. En el caso de las cargas atrapadas en el óxido deben considerarse en la resolución de la ecuación de Poisson.

La descripción de la interacción entre los electrones y las impurezas ionizadas se basa en el formalismo de Brooks y Herring [Bro51] en el que la probabilidad de dispersión por unidad de tiempo en la aproximación de bandas no parabólicas viene dada por

$$\Gamma_C = \frac{\sqrt{2} N_I Z^2 e^4 m^{*3/2}}{\pi \varepsilon_{Si} \hbar^4} \frac{(1+\alpha E)^{1/2} (1+2\alpha E)}{L_D^2 \left(4k^2 + L_D^2\right)} \tag{2.28}$$

donde N_I representa la densidad de impurezas ionizadas, Z es la carga de dichas impurezas y L_D es la denominada *longitud de Debye*

$$L_D = \sqrt{\frac{e^2 n_0}{\varepsilon_{Si} k_B T}} \tag{2.29}$$

donde n_0 es la concentración de portadores libres.

El mecanismo de dispersión es anisótropo, de forma que el ángulo azimutal se elige aleatoriamente entre 0 y 2π, mientras que el ángulo polar θ, que está defindo por las direcciones de $\vec{k}$ y $\vec{k'}$, viene dado por

$$cos\theta_r = 1 - \frac{2(1-r)}{1 + 4r\frac{k^2}{L_D^2}} \tag{2.30}$$

donde r es un número aleatorio comprendido entre 0 y 1.

2.2.6 El método Monte Carlo para un conjunto de partículas

Este método se emplea en aquellos problemas cuya solución depende del espacio o del tiempo, es decir, en procesos de transporte inhomogéneos o transitorios, como por ejemplo el simulador que paralelizamos en este trabajo. En estos casos no podemos emplear, como hicimos hasta ahora en el método Monte Carlo de una única partícula, el principio de ergodicidad del sistema. Debemos pues simular un conjunto de partículas.

Para analizar la simulación de un transitorio consideraremos el gas homogéneo de electrones dependiente del tiempo. Un caso de especial interés es el estudio de la respuesta dinámica del transitorio a los cambios

del campo aplicado al gas. En este caso resulta necesario hacer, de forma independiente, la simulación de muchas partículas con las distribuciones apropiadas para las condiciones iniciales. Para un número de partículas simuladas lo suficientemente grande, podemos obtener las funciones de distribución $f(\vec{k},t)$ o $f(E,t)$ a partir de los histogramas que se obtienen de la lectura a intervalos regulares de tiempo de los vectores de onda y las energías. Si queremos calcular una magnitud A que nos resulta de interés, el valor promedio lo podemos obtener a partir de este conjunto de muestras como una función del tiempo. El valor promedio de la magnitud se puede obtener haciendo uso de la expresión

$$< A >= \frac{1}{N} \sum_{i} A_i \left(t\right) \tag{2.31}$$

siendo N el número total de partículas. La duración de la respuesta transitoria se conoce a priori y será del orden de los mayores valores que tengan los tiempos característicos del sistema de electrones. Este tiempo depende, en general, de los valores del campo aplicado y la temperatura.

Para determinar la precisión de los resultados obtenidos, se separa el conjunto entero en varios subconjuntos y se estima para cada uno de ellos la magnitud A que deseamos calcular. Entonces su valor promedio y desviación estándar se pueden tomar, respectivamente, como el valor más probable y la incertidumbre estadística de A.

La simulación del transporte de portadores en sistemas inhomogéneos (dependientes de la posición) es de especial interés en el análisis y modelado de dispositivos. En este caso, también resulta necesario emplear un conjunto de partículas independientes y los promedios se deben tomar sobre las partículas que se encuentran en una posición dada. La incertidumbre estadística de los resultados se puede obtener con subconjuntos, tal y como se indicó en el caso transitorio.

A continuación, en los siguientes apartados, describiremos de un modo más concreto las principales características de los simuladores MC empleados en este trabajo.

2.3 Descripción del simulador 2D Multivalle Monte Carlo de transistores MOSFET con correcciones cuánticas

La principal característica del simulador 2D Multivalle Monte Carlo (MV–ECBE–EMC), con respecto al algoritmo de simulación Monte Carlo que hemos visto hasta ahora, es que considera la corrección cuántica del potencial mediante la inclusión del modelo denominado MV–ECBE (*Multi–Valley version of the Effective Conduction Band Edge*) [SGGR06]. En esta aproximación se tienen en cuenta los efectos cuánticos mediante la corrección del potencial a partir de la expresión

$$V_j^* \simeq V + \frac{\hbar^2}{4erV_T}\left\{\nabla\cdot\left[\left(\frac{\overleftrightarrow{1}}{m}\right)_j\cdot\nabla V_j^*\right] + \frac{1}{2V_T}\left[\nabla V_j^*\cdot\left(\frac{\overleftrightarrow{1}}{m}\right)_j\cdot\nabla V_j^*\right]\right\} \tag{2.32}$$

con la que se obtiene el potencial efectivo V_j^* para cada valle. $V_T = K_BT/e$ representa el potencial térmico y r es un parámetro cuyo valor varía entre 1 para estados puros (temperaturas bajas y un elevado confinamiento) hasta 3 para estados mezcla (temperaturas altas y confinamiento débil). Esta expresión permite tener en cuenta los efectos de un tensor de masa efectiva arbitrario, $\frac{\overleftrightarrow{1}}{m}$, que describe las características de cada valle y las direcciones de confinamiento.

El impacto de cada uno de los valles se considera, tanto en las propiedades electrostáticas como del transporte, acoplando las ecuaciones de Poisson y las de transporte de Boltzmann con la solución de la ecuación ECBE para cada valle, tal y como se muestra en la figura 2.5 donde se representa el diagrama de flujo de este simulador. Cuando hemos definido el sistema físico y estimado las condiciones iniciales del movimiento, en las que se incluye una primera solución de la ecuación de Poisson y el cálculo

correspondiente del potencial corregido, se inicia el proceso iterativo de la simulación Monte Carlo con el vuelo de las partículas y la posterior evaluación de la carga asignada a cada nodo del mallado del dispositivo. Debido a las variaciones de la concentración en los nodos causadas por el transporte de carga, es necesario resolver de forma autoconsistente la ecuación de Poisson y la corrección cuántica del potencial, para continuar con el cálculo de los parámetros de salida necesarios, velocidad, corriente, etc. Finalmente, se procede a la actualización de la tabla de dispersión debido a que al emplear este método de simulación surgen mecanismos de dispersión que dependen de las concentraciones de cada valle.

El potencial efectivo que se obtiene de la ecuación 2.32 sustituye al potencial electrostático para el cálculo del campo eléctrico, como se muestra en la ecuación,

$$\overrightarrow{\varepsilon_j} = -\nabla_r V_j^* \tag{2.33}$$

de modo que el campo eléctrico es diferente para cada valle. El hecho de que exista una fuerza de arrastre diferente en cada valle provoca que sea necesario calcular las poblaciones correspondientes de forma auto–consistente durante la simulación. En este sentido, la dispersión intervalle juega un papel fundamental en los cálculos de las poblaciones ya que es el único mecanismo implementado que permite la transferencia de carga de un valle a otro. En su implementación se ha seguido la aproximación a orden cero con tres fonones [FL93a]. Además este simulador incluye otros mecanismos de dispersión tales como fonones acústicos, dispersión coulombiana y dispersión superficial [GRLV$^+$99]. Para implementar estos mecanismos de dispersión en el simulador MV–ECBE–EMC ha sido necesario adaptar las expresiones que determinan las probabilidades de dispersión a la aproximación para gases pseudo–2D.

2.3.1 Modelos de dispersión corregidos para gases pseudo–2D

Como ya hemos mencionado anteriormente, el simulador MV–ECBE–EMC requiere conocer las poblaciones relativas de cada uno de los valles

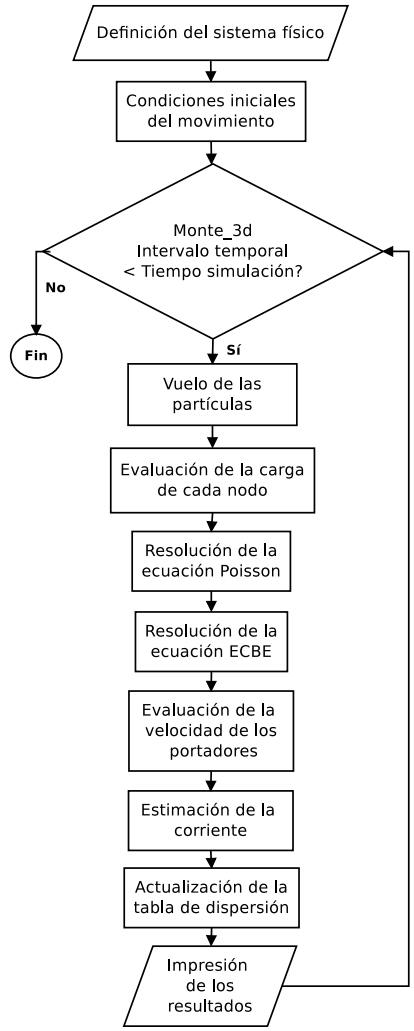


Figura 2.5: Diagrama de flujo del simulador 2D Multivalle Monte Carlo.

para poder reproducir la aportación de cada valle a la electrostática y al transporte del sistema. Sin embargo, éste no es un dato que se pueda conocer de un modo sencillo debido a que las poblaciones relativas van cambiando para cada posición del canal, cuando se aplica una tensión entre

drenador y fuente, de forma que el valor obtenido a partir de la solución exacta unidimensional tampoco resulta ser adecuado. Una solución a este problema es el replanteamiento de algunos mecanismos de dispersión de forma que se puedan reproducir de un modo adecuado y autoconsistente los cambios en las poblaciones relativas de los valles en diferentes puntos del dispositivo.

La dispersión superficial es otro mecanismo de dispersión que debe ser modelado para gases pseudo–2D debido a las correcciones cuánticas del MV–ECBE–EMC. Los modelos que consideran la dispersión superficial en simulaciones EMC semiclásicas se basan principalmente en el modelo de Fuchs [Fuc38], modelando las interfaces como difusores. Por lo tanto, cuando un portador alcanza alguna de las interfaces sufre una reflexión que, en el caso más general, podrá ser especular o difusa con una cierta probabilidad dependiendo de los parámetros considerados para la misma. Cuando se consideran correcciones cuánticas se deben utilizar nuevos modelos que tengan en cuenta el efecto de las interfaces rugosas aún cuando los electrones se encuentren, en promedio, alejados una cierta distancia de las mismas.

Dispersión por fonones en gases pseudo–2D

El modelo de dispersión de fonones para gases pseudo–2D empleado en el MV–ECBE–EMC trata de calcular de forma autoconsistente la población de cada valle sin tener que ajustar las probabilidades de dispersión para cada caso y cada transición. Para ello considera el solapamiento entre las envolventes de las funciones de onda entre los estados inicial y final de la transición [FL93a]. En el caso de las correcciones cuánticas utilizadas habitualmente (Gradiente de la densidad, Potencial efectivo o ECBE), la energía se aproxima por un continuo y, por tanto, se considera una única subbanda. Para el caso del método MV–ECBE–EMC se considera una subbanda en cada valle y, de esta forma, la concentración de electrones normalizada desempeña el papel de la envolvente de la función de onda en el caso de un gas 2D. Por lo tanto, la envolvente de la función de onda para cada valle considerado en la aproximación de gas pseudo–2D se puede

expresar como

$$|\Psi_i(y)|^2 = \frac{n_i(y)}{\int_0^{T_{Si}} n_i(y)dy} \tag{2.34}$$

donde y es la dirección de confinamiento, i representa el índice del valle considerado, $n_i(y)$ es la concentración del valle en la dirección y y T_{Si} el espesor de la lámina de silicio. Las probabilidades de dispersión para fonones quedan corregidas por el factor de forma que incluye la integral de solapamiento entre los valles i y j

$$F_{ij} = L_D \int_0^{T_{Si}} |\Psi_i(y)|^2 |\Psi_j(y)|^2 dy \tag{2.35}$$

donde L_D representa la longitud de Debye. El factor de forma F_{ij} es adimensional y para $i = j$ da idea del tamaño del paquete de ondas en el caso confinado en comparación con el del *Bulk*, representado por L_D. En el caso de procesos f de dispersión intervalle, transiciones entre valles no equivalentes, se debe considerar la masa del valle final debido a la anisotropía del material. De este modo, resultan más probables las transiciones de tipo f hacia los valles con una masa de confinamiento mayor, lo que está de acuerdo con los cálculos cuánticos, ya que estos valles se corresponden con los niveles de menor energía del sistema y deben ocuparse en primer lugar. Por lo tanto, se consigue realizar un ajuste autoconsistente de la población de electrones de cada valle sin necesidad de utilizar un valor de la población a priori. Además este modelo permite reproducir efectos puramente cuánticos como la modulación intersubbanda [GRLV$^+$01], que no pueden ser reproducidos de forma natural mediante modelos de un valle.

Dispersión por rugosidad superficial en gases pseudo–2D

Debido a las correcciones cuánticas del simulador MV–ECBE–EMC, el máximo de la distribución de electrones se encuentra alejado de las interfaces con los materiales aislantes y los modelos basados en reflexiones difusas [Fuc38] dejan de resultar válidos. Por lo tanto, es necesario emplear un modelo adecuado para describir la rugosidad superficial.

El modelo que se implementa en el simulador MV–ECBE–EMC está basado en un método para gases 2D [GRLV+99], en el que se considera un gas pseudo–2D suponiéndose una única subbanda ocupada en cada valle y para el cual se obtiene la probabilidad de dispersión para un electrón en el valle i con un vector de onda $\vec{k}$ mediante la expresión

$$\Gamma_{SRi}\left(x,\vec{k}\right) = \frac{m_i e^2 \left|\int \Psi_i\left(x,y\right) \frac{\Delta V_m(x,y)}{\Delta_{SR}} \Psi_i\left(x,y\right) dy\right|^2 \Delta_{SR}^2 L_{SR}^2}{2\hbar^3} F_\theta \quad (2.36)$$

donde

$$F_\theta = \int_0^{2\pi} \frac{d\theta}{\left(1+\frac{L^2q^2}{2}\right)^{3/2}} \quad (2.37)$$

$$q^2 = 2k^2\left(1-cos\left(\theta\right)\right) \quad (2.38)$$

Δ_{SR} y L_{SR} son parámetros de rugosidad que representan la varianza y la longitud de correlación de la superficie respectivamente, m_i es la masa efectiva asociada a la dirección de confinamiento, $\Delta V_m\left(x,y\right)$ representa la perturbación del potencial electrostático debida a una variación Δ_{SR} en el espesor de la lámina semiconductora y $\Psi_i\left(x,y\right)$ es la envolvente de la función de onda en el valle i calculada como

$$\Psi_i(x,y) = \sqrt{\frac{n_i(x,y)}{\int_0^{T_{Si}} n_i(x,y)dy}} \quad (2.39)$$

con $n_i\left(x,y\right)$ la contribución a la concentración de electrones del valle i. Para el desarrollo de la expresión final se considera que los procesos de dispersión son elásticos y anisótropos, y las transiciones intervalle tienen una probabilidad despreciable. El ángulo que forma el vector de onda resultante tras la dispersión con el inicial, se calcula utilizando una técnica combinada de forma que el ángulo seleccionado, θ, es el que cumple la condición

$$r = \frac{\int_0^\theta \frac{d\theta'}{\left(1+\frac{L^2q^2}{2}\right)^{3/2}}}{F_\theta} \quad (2.40)$$

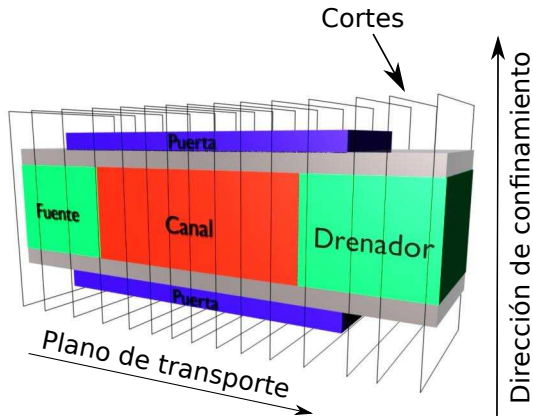


Figura 2.6: Representación de un DGSOI MOSFET simulado con el MSB–EMC. La ecuación de transporte se resuelve en el plano de transporte y la ecuación de Schrödinger 1D se resuelve en la dirección de confinamiento para los puntos de la malla situados a lo largo de la dirección de transporte.

2.4 Descripción del simulador 2D Multisubbanda Monte Carlo de transistores MOSFET

El simulador 2D Multisubbanda Monte Carlo (MSB–EMC) está basado en la aproximación *mode–space* del transporte cuántico [VRD$^+$02a], de modo que se considera que los problemas de transporte y confinamiento están desacoplados. Por lo tanto, se resuelve la ecuación de Schrödinger en la dirección de confinamiento mientras que el problema de transporte se resuelve de forma semiclásica, mediante la ecuación de transporte de Boltzmann. La limitación principal de este método se encuentra en la aproximación de desacoplamiento, ya que no permite la inclusión de fenómenos coherentes de un modo directo para el estudio de algunas estructuras. Si embargo, este problema puede ser evitable. Por ejemplo, en el caso de los transistores MOSFET en donde los efectos de transporte cuántico como el

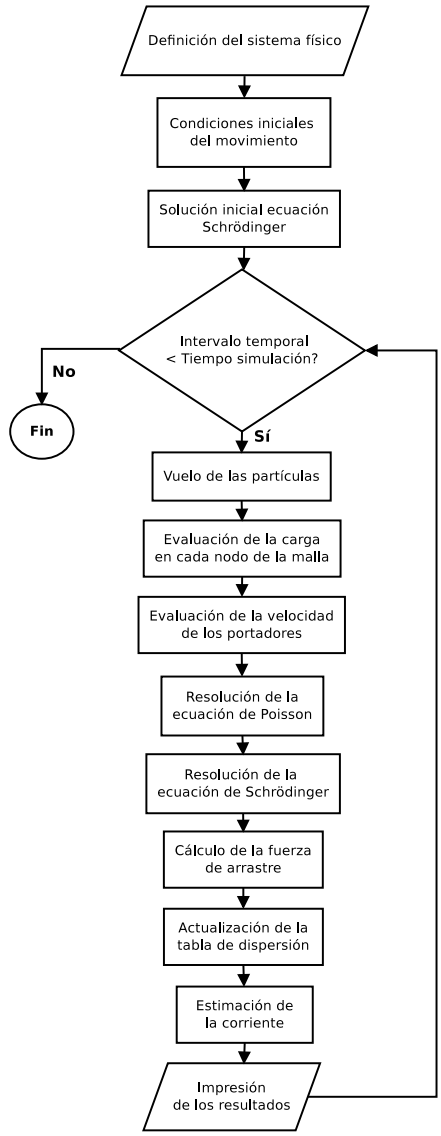


Figura 2.7: Diagrama de flujo del simulador MSB–EMC.

túnel banda–banda se pueden incluir de forma independiente cuando sea necesario [FL88].

Desde el punto de vista de la simulación podemos considerar que el transistor está formado por un conjunto de cortes, en la dirección de confinamiento, que se extienden de la fuente al drenador en el plano de transporte, tal y como se muestra en la figura 2.6. La ecuación de Schrödinger 1D se resuelve, en cada uno de los cortes en los que hemos dividido el dispositivo y para cada valle de la banda de conducción, de forma auto–consistente con la solución de la ecuación de Poisson 2D del dispositivo. Como resultado de esta solución se obtiene la evolución en la dirección de transporte de los niveles de energía de la i–ésima subbanda y v–ésimo valle, $E_{i,v}(x)$, así como de las funciones de onda $\psi_{i,v}(x,z)$. La solución del problema de transporte se realiza mediante la solución Monte Carlo de la BTE en la dirección correspondiente. Tanto para la resolución del problema de transporte como para el problema de confinamiento, se considera la aproximación analítica y no parabólica de la banda de conducción [FL93b], con un valor del coeficiente de no parabolicidad (α) igual a 0.5 (eV^{-1}).

La figura 2.7 muestra el diagrama de flujo del simulador MSB–EMC. A diferencia del simulador MV–ECBE–EMC (figura 2.5), se puede observar que se realiza una solución inicial de la ecuación de Schrödinger de la que se obtienen las funciones de onda para cada subbanda y valle, permitiéndonos conocer las poblaciones de cada valle y subbanda así como calcular una versión inicial de la tabla de dispersión. Para obtener la población de cada subbanda en cada punto de la malla de la dirección de transporte es necesario hacer una nueva asignación de las partículas a los nodos de la malla correspondiente, empleando para ello el método *cloud–in–cell*. La población de la subbanda se emplea para pesar la densidad de probabilidad correspondiente $|\psi_{i,v}(x,z)|^2$ y de este modo calcular la densidad de electrones, $n(x,z)$.

De acuerdo con la aproximación *mode–space*, el campo de arrastre que experimentan los electrones durante el transporte viene dado por la

variación de la energía de la subbanda para cada valle en dicha dirección,

$$-e\vec{\varepsilon}_{i,v} = \frac{\partial E_{i,v}(x)}{\partial x}\hat{x}. \tag{2.41}$$

Por lo tanto, el campo de arrastre es diferente para cada una de las subbandas correspondientes a un valle dado. Esta es una de las principales diferencias con respecto al MV–ECBE–EMC, en donde el campo de arrastre depende del potencial corregido para cada valle.

Una vez iniciada la simulación Monte Carlo y después de simular el transporte de electrones es necesario actualizar el potencial electrostático resolviendo la ecuación 2D de Poisson con las nuevas concentraciones, $n(x, z)$, obtenidas después del vuelo.

Uno de los principales inconvenientes del simulador MSB–EMC es que requiere más tiempo de cálculo que el simulador MV–ECBE–EMC. Entre los motivos principales se encuentra la necesidad de actualizar la tabla de dispersión cada vez que se resuelve la ecuación de Schrödinger, de modo que se mantenga la autoconsistencia de la simulación. Esta actualización se debe a la evolución de los niveles de energía y las autofunciones en la dirección de transporte que originan a su vez variaciones de las probabilidades de dispersión en diferentes posiciones del dispositivo. Los cambios en la distribución de los niveles de energía modifican las energías permitidas entre subbandas y la variación de las funciones de onda con la posición afecta a los factores de forma de los procesos intrabanda.

Los mecanismos de dispersión incluidos en el simulador MSB–EMC han sido implementados siguiendo los modelos descritos para láminas en inversión. En el caso de la dispersión por fonones acústicos e intervalle se han seguido las referencias [FL93b, GRLV98], mientras que el modelo empleado en la dispersión por rugosidad superficial está descrito en [GRLV$^+$99, GCCRJM02] y para la interacción Coulombiana en [GF03, JMGD08]. Todos estos modelos consideran bandas no parabólicas y elipsoidales.

2.4.1 Optimización del simulador 2D Multisubband Monte Carlo de transistores MOSFET

Durante el proceso de paralelización de los simuladores 2D MV–ECBE–EMC y MSB–EMC, que describiremos más adelante en el capítulo 3, se han detectado algunas posibles optimizaciones que ayudarían a elevar el porcentaje de código paralelo y por consiguiente su aceleración, y que finalmente hemos implementado en el simulador MSB–EMC.

La versión inicial de estos simuladores, se basa en la descripción que hace Tomizawa en su libro [Tom93], *Numerical Simulation of Submicron Semiconductor Devices*, en donde explica como desarrollar un simulador Monte Carlo de dispositivos semiconductores. En la descripción propuesta, la subrutina *Electron Free Flights* simula el movimiento de los portadores y selecciona las partículas absorbidas por los contactos de fuente y drenador como borrables. Las partículas que salen por los contactos se marcan como borrables cambiando el índice de valle a los valores 8 ó 9 dependiendo de si la partícula es absorbida por el contacto de fuente o drenador. Una vez que la subrutina *Electron Free Flights* ha finalizado, la subrutina *Renew* evalúa el número de partículas que se encuentran dentro del dispositivo borrando todas aquellas que han salido por los contactos y cuyo valor de índice de valle es igual a 8 ó 9. Esta subrutina también se encarga de mantener la neutralidad de carga en las vecindades de los contactos óhmicos de fuente y drenador, borrando o inyectando el número de partículas necesarias. La figura 2.8 presenta una descripción de la versión propuesta por Tomizawa [Tom93] e implementada en la versión inicial del simulador 2D MSB–EMC.

Durante la paralelización del simulador 2D MSB–EMC hemos fusionado las subrutinas *Electron Free Flights* and *Renew* con la intención de optimizar el código. De esta forma, con un único lazo podemos implementar las funcionalidades de ambas subrutinas reduciendo el coste computacional. En esta propuesta el vuelo libre de las partículas se realiza del mismo modo que en la subrutina *Electron Free Flights* sin optimizar y las partículas que salen por los contactos se marcan, tal y como se hacía en la versión previa, con los valores de índice de valle 8 ó 9 dependiendo

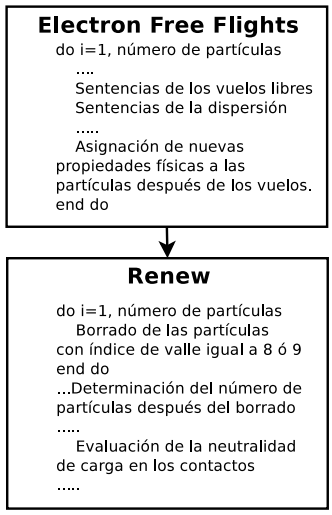


Figura 2.8: Versión inicial de las subrutinas *Electron Free Flights* y *Renew*.

de si salen por el contacto de fuente o drenador. La diferencia ahora es que las partículas que salen por el drenador son inyectadas de nuevo por la fuente y correspondientemente las que salen por fuente son inyectadas de nuevo por el drenador. Cada vez que una partícula es inyectada por un contacto se evalúa la condición de neutralidad de carga. En el caso de que la partícula inyectada incumpla dicha condición, esta partícula es marcada para ser borrada al final del lazo. De esta forma cuando se haya completado el vuelo de las partículas, tan sólo nos quedará por borrar las partículas marcadas y evaluar de nuevo la condición de neutralidad de carga en los contactos. Si el número de partículas inyectadas en los contactos no es suficiente para que se cumpla la condición de neutralidad de carga, se inyectan las partículas que restan para cumplir dicha condición. La figura 2.9 muestra una descripción simplificada de la versión optimizada de la subrutina *Electron Free Flights*.

Para ver el efecto que tiene la optimización en el peso computacional del código, hemos representado en la figura 2.10 el peso computacional

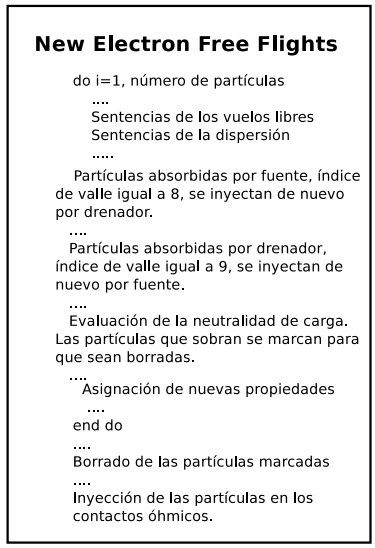


Figura 2.9: Versión optimizada de la subrutina *Electron Free Flights*.

de la subrutina *Electron Free Flights* optimizada y la suma de las pesos computacionales de las subrutinas *Renew* y *Electron Free Flights* sin optimizar. Además también se representa de forma independiente la carga computacional de cada una de estas subrutinas.

La figura representa el porcentaje del tiempo total de ejecución del simulador 2D Multisubband Monte Carlo que le corresponde a estas subrutinas para diferentes valores de la variable *scsc*. Esta variable fija cada cuantas iteraciones temporales se calcula de forma auto-consistente la solución de la ecuación de Schrödinger y la actualización de la tabla de dispersión.

Para valores muy bajos de *scsc*, *scsc* = [1,2], la auto-consistencia es muy alta y la carga computacional de las subrutinas optimizadas no es significativa en comparación con la subrutina que calcula la solución de la ecuación de Schrödinger y la encargada de actualizar la tabla de dispersión. En estos casos la optimización nos permite una reducción de la carga

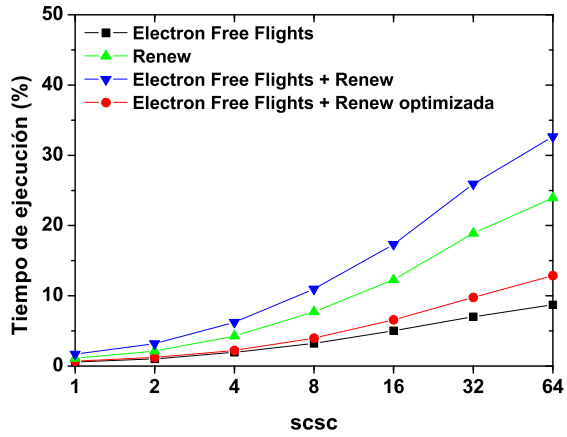


Figura 2.10: Peso computacional en función de la variable *scsc* para la versión inicial de las subrutinas *Renew* y *Electron Free Flights*. También se representa la suma de ambas subrutinas y la nueva versión optimizada de la subrutina *Electron Free Flights*.

computacional de entre el 1 y el 2 % del tiempo total de ejecución. Sin embargo, cuando el valor de la variable *scsc* es muy alto, lo que implica una auto-consistencia menor, el peso computacional de las subrutinas optimizadas es mayor y los efectos de la optimización son mayores. Por ejemplo, para $scsc = 64$ la suma de la carga computacional de las subrutinas *Renew* y *Electron Free Flights* es superior al 30 % del tiempo total de ejecución. Comparando este porcentaje con el peso computacional de la versión optimizada observamos una disminución del 20 % del peso computacional. Como consecuencia de esta disminución, el peso computacional de otras subrutinas que han sido paralelizadas en su totalidad se ha visto incrementado, permitiéndonos alcanzar un porcentaje de código paralelizado mayor y un mejor rendimiento del código paralelo.

CAPÍTULO 3

Programación paralela aplicada a la simulación de nanodispositivos semiconductores

En general la simulación en ingeniería requiere elevados recursos computacionales. En el caso de los simuladores Monte Carlo estudiados en este trabajo tenemos tiempos característicos en torno a una o dos horas de cálculo por cada intervalo temporal simulado de 1 ps de duración. Por lo tanto, completar una de las simulaciones habituales de unos 30 ps puede llevarnos varios días de cálculo dependiendo de la máquina.

Como solución para reducir el tiempo de ejecución de los simuladores empleados en este trabajo se ha optado por la paralelización de estos códigos de simulación. La reducción del tiempo de ejecución presenta dos ventajas fundamentales en la simulación de nanodispositivos: la posibilidad de realizar estudios paramétricos de las propiedades de los transistores en tiempos razonables, para los que se requieren decenas o cientos de simulaciones, y la posibilidad de acelerar la implementación de nuevos modelos en el simulador gracias a la reducción del tiempo de cálculo de las ejecuciones de prueba.

Si consideramos el fuerte desarrollo que han tenido las arquitecturas

multi–núcleo en los últimos años, poniendo a disposición del usuario procesadores de hasta 8 núcleos, parece lógico que se desarrollen simuladores paralelos que permitan obtener el máximo provecho posible de los recursos computacionales disponibles.

En este capítulo se introducen en primer lugar los diferentes tipos de sistemas distribuidos y las arquitecturas multi–núcleo. Además, se presenta el estándar de programación OpenMP y se describe el proceso de paralelización de los simuladores 2D MV–ECBE–EMC y MSB–EMC de transistores MOSFET. Finalmente, se presentan los resultados de aceleración obtenidos con las implementaciones paralelas realizadas.

3.1 Sistemas distribuidos

Existen varias definiciones de los sistemas distribuidos, como por ejemplo la que dan Tanenbaum y Maarten van Steen en su libro [TvS02].

"*Un sistema distribuido es una colección de computadores independientes que se muestran, de cara al usuario, como un único sistema coherente.*"

Como se puede apreciar, esta definición es un compendio de dos definiciones. Una hecha desde el punto de vista del hardware, en donde las máquinas deben cumplir con el requisito de ser autónomas. Y la segunda desde el punto de vista del software, en donde los usuarios ven una única máquina.

Para que el conjunto heterogéneo de redes y computadoras que forman parte de un sistemas distribuido se muestre al usuario como un sistema único, los sistemas distribuidos se organizan por medio de un nivel de software que se encuentra situado entre el nivel del sistema operativo y el de usuario y aplicaciones, tal y como se muestra en la figura 3.1, denominado *middleware*. Como se puede apreciar en esa figura el nivel de middleware se extiende sobre todas las máquinas.

Cuando construimos un sistema distribuido existen cuatro objetivos que deberíamos perseguir:

1. Facilitar la conexión de los usuarios con los recursos. El objetivo principal de un sistema distribuido es conseguir que los usuarios

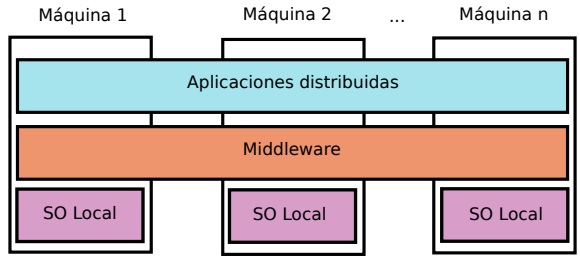


Figura 3.1: Organización de un sistema distribuido incluyendo el *middleware*.

dispongan de un acceso sencillo a los recursos remotos y que éstos se compartan entre los usuarios de un modo controlado. Estos recursos pueden ser impresoras, computadores, medios de almacenamiento, datos, páginas web, redes de conexión, etc. El motivo principal por el que se comparten los recursos es económico, ya que por ejemplo resulta más barato compartir una impresora entre todos los usuarios que comprar una impresora para cada uno.

Sin embargo, cuando se incrementa la conectividad y los recursos compartidos, la seguridad se convierte en un tema muy importante porque con frecuencia las contraseñas de usuarios y otra información sensible se envían a través de la red en texto plano, o se almacenan en servidores que no ofrecen ninguna confianza.

2. El acceso debe ser transparente. El hecho de ocultar que los recursos y procesos se encuentran físicamente distribuidos entre múltiples computadoras es otro de los objetivos que se persigue en un sistema distribuido. Se busca que el sistema sea capaz de mostrarse ante los usuarios y las aplicaciones como si fuera una única computadora.

3. El sistema debe ser abierto. Un sistema distribuido abierto es un sistema que ofrece los servicios siguiendo unas reglas estándar que describen la sintaxis y la semántica de esos servicios. Por ejemplo, en las redes de computadores, existen unas reglas estándar que rigen

el formato, los contenidos y la gestión de mensajes enviados y recibidos. Estas reglas están formalizadas en protocolos. En los sistemas distribuidos, los servicios se especifican a través de interfaces que con frecuencia se encuentran descritas en un lenguaje que define la interfaz llamado IDL (*Interface Definition Language*). Las definiciones de la interfaz que se han escrito en IDL casi siempre capturan la sintaxis de los servicios. Es decir, especifican de modo preciso los nombres de las funciones que están disponibles junto con los tipos de parámetros, valores devueltos, posibles excepciones, y demás requerimientos. La parte que resulta más complicada es aquella en la que hay que especificar de modo preciso que hacen esos servicios, lo que se denomina la semántica de las interfaces.

4. El sistema debe ser escalable. La escalabilidad de un sistema se puede medir al menos en función de tres parámetros [CS94]:

   - Su tamaño, indicando la facilidad del sistema para añadir más usuarios y recursos.
   - Escalabilidad geográfica, indicando la capacidad del sistema para mantener su utilidad y usabilidad, con independencia de lo lejos que se encuentren los usuarios y los recursos.
   - Escalabilidad administrativa, indicando la facilidad en la administración cuando los usuarios y recursos pertenezcan a organizaciones diferentes.

   Con frecuencia un sistema que es escalable en uno o más de estos parámetros deja de serlo en otro.

3.1.1 Clasificación de los sistemas distribuidos

La estructura del hardware de los sistemas distribuidos puede ser muy variada, especialmente en términos de como están interconectados y como se comunican. Existen diferentes clasificaciones de este tipo de sistemas pero nosotros hemos elegido la que considera al sistema distribuido como

una colección de computadores independientes. De este modo dividiremos todos los computadores en dos grupos:

- Los multiprocesadores, aquellos que comparten la memoria, también llamados de memoria compartida.
- Los multicomputadores, aquellos que no comparten la memoria, también llamados de memoria distribuida.

La diferencia principal entre ambos grupos es que un multiprocesador tan sólo dispone de un espacio único de direcciones de memoria para todos los procesadores. Cuando un procesador escribe un valor en una posición de memoria, por ejemplo el valor 20 en la dirección 100, otro procesador que lea la dirección 100 obtendrá el valor 20. Es decir todas las máquinas comparten la misma memoria. Por contra, en los multicomputadores, cada máquina tiene una memoria propia. Un ejemplo de un multicomputador es un conjunto de ordenadores conectados en red.

A su vez, cada una de estas categorías puede ser subdividida en dos categorías diferentes en función del tipo de conexión de red, como se puede apreciar en la figura 3.2:

- Basadas en bus. Cuando existe una única red, cable, bus u otro medio que conecta a todas las máquinas. Por ejemplo, la televisión emplea este esquema, la compañía de cable extiende un cable y todas las personas que tienen contrato con esta compañía conectan sus televisores a este cable.
- Basadas en conmutador (*switch*). Los sistemas conmutados no tienen un único cable, cada máquina dispone de un cable conectado a un conmutador y la comunicación entre ellas puede tener lugar a través de múltiples caminos. Por ejemplo, el sistema de telefonía público está organizado de esta forma.

También es posible establecer una clasificación que nos permite dividir los multicomputadores en dos grupos, los denominados heterogéneos y los homogéneos. Los multicomputadores homogéneos disponen de una

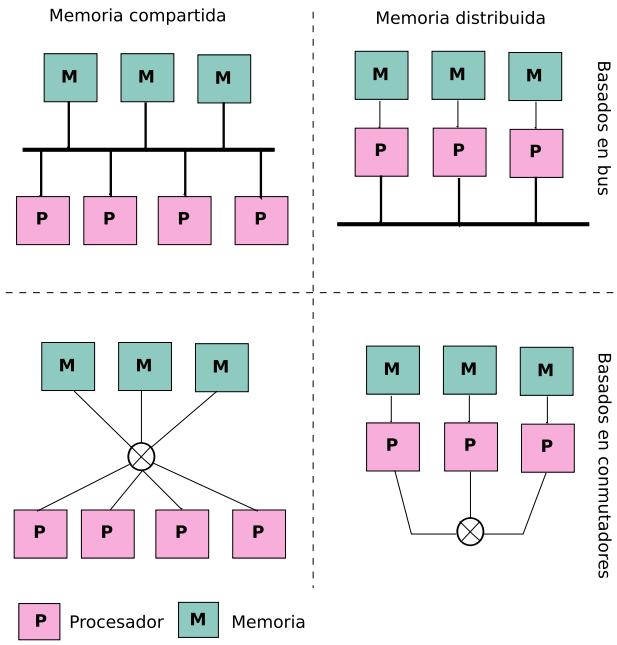


Figura 3.2: Clasificación de los sistemas distribuidos en función de la organización de la memoria y del tipo de conexión de la red.

red única que emplea la misma tecnología para todo el sistema, además todos los procesadores son iguales y disponen de la misma cantidad de memoria privada. Por contra, en los heterogéneos los ordenadores pueden ser diferentes y también estar conectados a través de redes diferentes.

A continuación trataremos en detalle los sistemas multicomputador y multiprocesador.

Multiprocesadores

En este caso todos los procesadores tienen acceso directo a la memoria compartida. Los sistemas multiprocesador basados en bus consisten en un cierto número de procesadores todos ellos conectados con un módulo de memoria a través de un único bus. Un ejemplo de una configuración simple

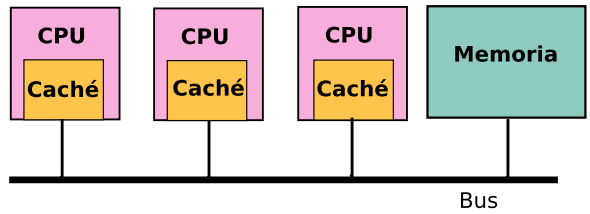


Figura 3.3: Multiprocesador basado en bus.

de este tipo de sistemas es una placa base en la cual se pueden insertar varios procesadores o tarjetas de memoria.

Cuando disponemos sólo de una memoria, si un procesador escribe una palabra en memoria y otro procesador lee esa palabra un instante después obtiene el mismo valor que escribió el primer procesador. Una memoria con esta propiedad se denomina coherente. El problema que plantea este tipo de esquema es que al aumentar el número de procesadores, el bus estará generalmente sobrecargado por el nivel de peticiones y el rendimiento caerá rápidamente.

Una solución a este problema es añadir una memoria caché por cada procesador entre la memoria y el bus para reducir el acceso a memoria, tal y como se puede ver en la figura 3.3. La caché mantiene la mayoría de las palabras cuyo acceso a memoria se ha hecho recientemente. Todas las peticiones a memoria pasan por la caché y si la palabra solicitada se encuentra almacenada en ésta, es la caché la encargada de responder la petición del procesador, sin necesidad de que la solicitud se haga a través del bus. Cuando el tamaño de la caché es lo suficientemente grande, la probabilidad de éxito, llamada *hit rate*, aumenta, y el tráfico a través del bus debido al procesador desciende drásticamente. A modo de ejemplo para tamaños de caché de 512 KB a 1 MB la tasa de acierto habitualmente es del 90 % o superior [TvS02].

Sin embargo la introducción de cachés crea el problema denominado de coherencia caché. Por ejemplo, consideremos dos procesadores, A y B, cada uno leyendo la misma palabra situada en sus respectivas cachés. Si

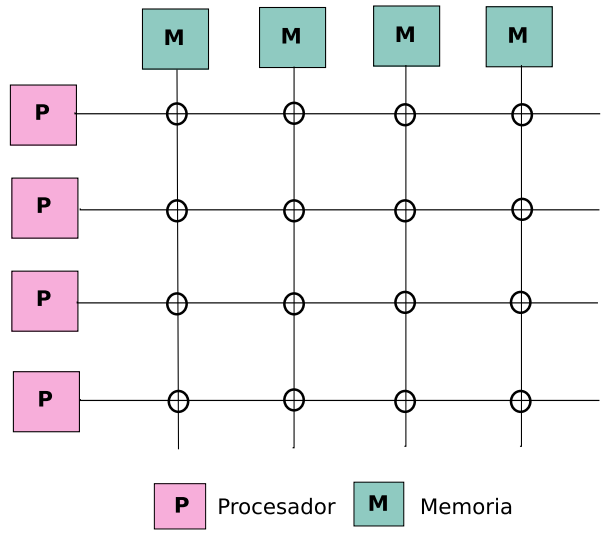


Figura 3.4: *Crossbar switch*.

el procesador A modifica la palabra, en la próxima lectura el procesador B lee el valor antiguo de la palabra en su propia caché, en vez de leer el valor que ha modificado A. La memoria ahora es incoherente. Existen varias técnicas a nivel de *hardware* orientadas a resolver el problema de la coherencia cache [HP06].

El problema de los multiprocesadores conectados en bus es su limitada escalabilidad, incluso cuando usamos cachés. Para construir sistemas multiprocesador con un número elevado de procesadores necesitamos un método diferente al de conectar todos los procesadores con la memoria. Una posibilidad es la división de la memoria en módulos que se conectan a los procesadores por medio de conmutadores entrecruzados, también llamados *crossbar switch*, tal y como se muestra en la figura 3.4. En este caso, cada procesador y cada división de la memoria disponen de una conexión que parte de ellos, cada intersección es un punto de conmutación que se puede abrir o cerrar por *hardware*. Si un procesador quiere acceder a un módulo de memoria en particular, el punto de conmutación los conecta

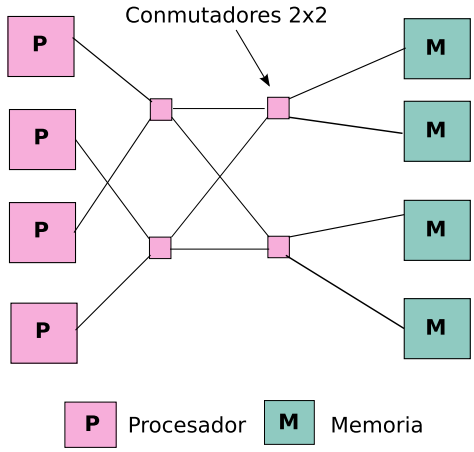


Figura 3.5: Red de conmutación omega.

entre sí e impide momentáneamente el resto de conexiones con ese módulo mientras esa conexión tenga lugar. La ventaja de este sistema es que varios procesadores pueden acceder a la memoria al mismo tiempo, la desventaja es que si quieren acceder al mismo módulo de memoria uno de ellos tiene que esperar.

Un problema del *crossbar switch* es que para n procesadores y n memorias el número de conmutadores crece con n^2. Una alternativa que trata de solucionar este problema, reduciendo el número de conmutadores empleados, es la red omega que se muestra en la figura 3.5. Esta red dispone de cuatro conmutadores con dos entradas y dos salidas cada uno. Por ejemplo en la configuración que se muestra en la figura se puede observar que cada procesador es capaz de acceder a cada módulo de memoria. El inconveniente de estas redes de conmutación es que existen varias rutas por las que un procesador puede acceder a la memoria. Por lo tanto para asegurar latencias bajas entre el procesador y la memoria, la conmutación debe ser muy rápida, lo que hace que estas redes tengan un coste muy alto.

Para disminuir el coste de las redes omega surgieron las máquinas de tipo NUMA (NonUniform Memory Access), en las que algunas memorias

están asociadas a un procesador y cada procesador puede acceder a su propia memoria local rápidamente, pero el acceso a otra memoria diferente de la local es más lento. A pesar de que este tipo de máquinas gozan en promedio de mejores tiempos de acceso, plantean la complicación de que la posición en memoria de los programas y de los datos puede llegar a ser crítica para conseguir que la mayoría de los accesos tengan lugar en la memoria local.

Multicomputadores

En los multicomputadores cada procesador tiene una conexión directa a su propia memoria local. El único problema que presenta es que los procesadores tienen que establecer algún sistema explícito de comunicación entre si que permita el intercambio de datos.

En estos sistemas los nodos se conectan a través de una red única, que en la mayoría de los casos se trata de una red de interconexión de alto rendimiento. Podemos distinguir dos tipos de sistemas en función de la red, del mismo modo que en los sistemas multiprocesador, los que están basados en un bus y los que están basados en un conmutador.

En los multicomputadores basados en bus, los procesadores se conectan a través de una red de acceso compartido como por ejemplo Fast Ethernet. Este tipo de conexión presenta el mismo problema que en los sistemas multiprocesador, su escalabilidad es limitada. En los multicomputadores basados en conmutador, los mensajes entre los procesadores se propagan a través de las interconexiones de red. Existen muchas topologías de red para este tipo de sistemas pero las dos más comunes son la topología en malla y la hipercubo, véanse las figuras 3.6 y 3.7.

Un hipercubo es un cubo n–dimensional. El que mostramos en la figura 3.7 es cuadrimensional y podemos imaginarlo como dos cubos, cada uno con 8 vértices y 12 aristas, en donde los vértices de cada uno de los dos cubos están conectados entre sí. Cada uno de ellos representa un procesador y cada arista es una conexión entre dos procesadores.

Disponemos de una gran variedad de sistemas multicomputador, como ejemplo a continuación veremos dos concepciones radicalmente opuestas.

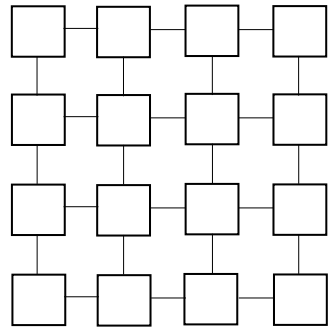


Figura 3.6: Topología en malla.

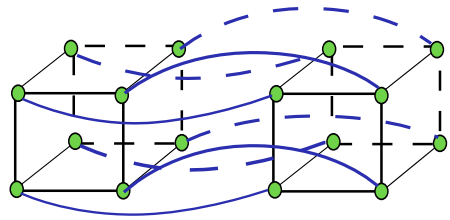


Figura 3.7: Topología en hipercubo.

Por un lado los MPPs (*Massively Parallel Processors*), que son inmensos supercomputadores, muy costosos, formados por una gran cantidad de procesadores. En muchos casos los procesadores no difieren de los que puede tener un ordenador de sobremesa. Lo que realmente los hace diferentes de otros multicomputadores es que están conectados entre sí por una red de alto rendimiento. Esta red está diseñada para alcanzar un alto ancho de banda y poca latencia. Además, dispone de medidas especiales para asegurar la tolerancia a fallos, ya que al disponer de miles de procesadores es muy probable que alguno falle, de este modo se asegura que el rendimiento de toda la máquina va a ser independiente del fallo de uno de sus procesadores.

Por otro lado, tenemos los clusters de estaciones de trabajo, que básicamente son una colección de ordenadores de sobremesa o estaciones de

trabajo conectados a través de una red de altas prestaciones tal como Gigabit Ethernet o Myrinet. Es precisamente este tipo de conexión lo que diferencia un cluster de MPPs. La red de interconexión de un cluster generalmente no dispone de medidas especiales que aseguren un ancho de banda alto, ni la protección contra fallos. Por consiguiente los cluster son más baratos y más simples.

3.2 Arquitecturas Multi–núcleo

Las arquitecturas multi–núcleo tienen la ventaja de que permiten explotar el paralelismo en equipos de sobremesa sin necesidad de utilizar costosas infraestructuras. El fuerte desarrollo experimentado por este tipo de arquitecturas en los últimos cinco años, desde los primeros procesadores con dos núcleos destinados a servidores, equipos de sobremesa y portátiles [Gee05], hasta los actuales que disponen de hasta 8 núcleos por procesador, ha convertido a los multi–núcleo en los procesadores más habituales presentes en los equipos de gama media alta.

La figura 3.8 muestra las familias de procesadores que forman parte de los supercomputadores de la lista TOP500 [top] desde el año 2005 hasta el año 2010. Los datos muestran como desde el año 2007, entre el 60 % y el 80 % de los supercomputadores que componen esta lista disponen de procesadores de la familia EMT64. Mientras que entre el 10 % y el 15 % de los sistemas está formado por procesadores de la familia AMD x86_64.

Sin ánimo de entrar en una comparación entre los dos principales fabricantes de procesadores, lo que se desea destacar con estos datos es que el 90 % de los supercomputadores de la lista TOP500 dispone de procesadores multi–núcleo. En concreto y según los datos de la lista publicada en el mes de Junio de 2010, el 85 % de los sistemas usan procesadores con cuatro núcleos y el 5 % usan procesadores con seis o más núcleos.

La figura 3.9 representa la estructura interna de un procesador de 6 núcleos de la serie Intel Xeon 7400 [KVAT10]. Como se puede ver en esta figura, la estructura típica de estos procesadores se basa en varios núcleos encargados de procesar las instrucciones y varios niveles de memoria que

pueden ser dedicados o compartidos. Por ejemplo, en el caso de la figura 3.9 cada 2 núcleos comparten una memoria caché de nivel 2 (L2) y los 6 núcleos comparten la memoria caché de nivel 3 (L3).

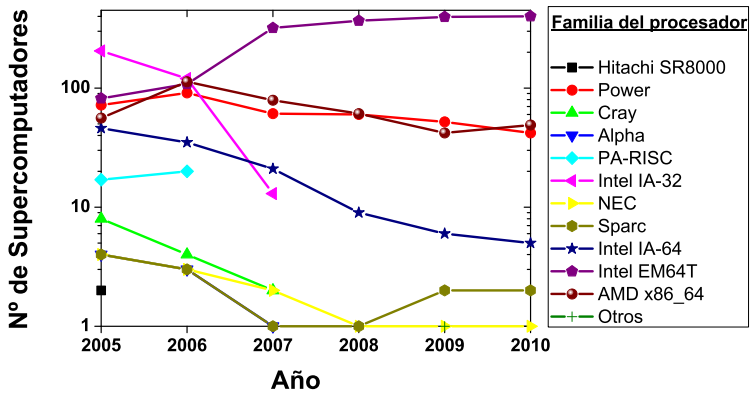


Figura 3.8: Evolución de la familia de procesadores de la lista Top500 en los últimos cinco años.

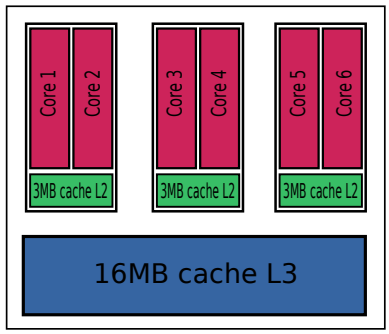


Figura 3.9: Representación de la memoria caché y los seis núcleos de un procesador de la serie Intel Xeon 7400.

Durante las últimas décadas la tendencia de diseño de los procesadores ha sido incrementar la capacidad de integración (el número de transistores por chip) y la frecuencia de los procesadores para mejorar el rendimiento, de forma que se pudieran computar un mayor número de instrucciones por ciclo [JJS06]. Sin embargo, con el aumento de la densidad de integración empezaron a surgir problemas de consumo de potencia, así como de disipación de calor. Las arquitecturas multi–núcleo son una alternativa de diseño que permiten remplazar la estrategia de computar más instrucciones por ciclo para mejorar el rendimiento de los procesadores, por otra que permite computar varias instrucciones a la vez usando varios núcleos de computación.

En la actualidad los sistemas multi–núcleo que hay disponibles en el mercado tienen un rendimiento del orden de GFLOPS (miles de millones de operaciones de punto flotante por segundo) y los núcleos disponibles son procesadores de propósito general que agrupados en un único circuito integrado dan como resultado un procesador paralelo. Sin embargo, algunos fabricantes como AMD con la tecnología AMD Fusion o Intel con el programa de investigación Tera–scale computing, están desarrollando sistemas basados en arquitecturas multi–núcleo que emplean aceleradores hardware para realizar cálculos específicos.

En el caso de la tecnología AMD Fusion la finalidad es crear un único procesador de alto rendimiento tanto para procesado serie como paralelo, denominado APU (*Accelerated Processing Units*), que añada aceleradores hardware de propósito específico [Bro10].

En el caso de la tecnología *Tera–scale computing* de Intel con la que se pretende llegar a rendimientos de TFLOPS (billones de operaciones de punto flotante por segundo), la arquitectura está basada en matrices bidimensionales en las que cada celda consta de un PE (*processing engine*) conectada a un *router* de 5 puertos [SSNS09]. El primer procesador TeraFLOPS, fabricado para pruebas, estaba basado en la tecnología de 65 nm y disponía de una matriz de tamaño 8x10 con un total de 80 celdas. El rendimiento del procesador se encuentra en el rango de 0,32 a 1,81 TFLOPS en función del escalado dinámico de la frecuencia y el consumo

de potencia.

Estas tecnologías, que durante los próximos años estarán disponibles en el mercado, muestran una tendencia clara hacia el uso de procesadores paralelos de memoria compartida y aceleradores hardware encapsulados en un único circuito integrado. El reto ahora, desde el punto de vista de la programación, está en encontrar un lenguaje de programación que permita explotar al mismo tiempo los multi–núcleo de propósito general y los aceleradores hardware. Hasta ahora, parece que el lenguaje de programación OpenCL [opeb] es el único que ha sido desarrollado con esta intención, pero sin duda habrá que esperar a que estas tecnologías se encuentren un poco más maduras [SZC+09, KM09] para portar extensas aplicaciones como los códigos de simulación Monte Carlo de dispositivos semiconductores que presentamos en esta memoria.

3.3 El estándar OpenMP

Se trata de una colección de directivas, librerías y variables de entorno para programas en Fortran, C y C++. En la actualidad es el estándar para la programación de sistemas de memoria compartida y como tal, consideramos que era el lenguaje adecuado para paralelizar los simuladores Monte Carlo con los que trabajamos con el fin de ejecutarlos en procesadores multi–núcleo.

OpenMP utiliza el modelo de ejecución paralela *fork–join*. El programa comienza su ejecución con un único hilo, llamado maestro, que cuando se encuentra la primera construcción paralela crea un conjunto de hilos que junto con el maestro se reparten el trabajo de la ejecución. Una vez acabada la región paralela sólo el maestro continúa la ejecución y el resto de hilos se destruyen.

Este lenguaje de directivas generalmente se emplea para la paralelización de lazos. El procedimiento es buscar los lazos computacionalmente más costosos y que los hilos se repartan las iteraciones del lazo. Durante la ejecución de la región paralela los hilos se comunican a través de variables compartidas. Es necesario tener cuidado al programar ya que la comparti-

ción de datos puede llevar a un mal comportamiento del programa debido al acceso simultáneo a los mismos desde diferentes hilos. El modo de evitarlo es usando directivas de sincronización que tienen la desventaja de ser computacionalmente costosas, por lo que debemos tratar de evitarlas.

3.3.1 Características principales del estándar OpenMP

OpenMP dispone una sintaxis general de las directivas de la forma **centinela nombre_directiva [cláusulas]**. Como centinelas se emplean ***C$OMP*** ó ****$OMP*** en programas de formato fijo y ***!$omp*** en programas de formato fijo o variable. Las cláusulas pueden estar separadas por espacios o comas y su orden es indiferente. Además permite la compilación condicional de líneas de programa por medio de centinelas, por ejemplo si introducimos la línea de código en lenguaje Fortran,

```
!$ A(i)=A(i)+b
```

cuando compilamos sin *+Oopenmp* el programa no ejecuta esta línea.

El mismo ejemplo para lenguaje C y C++ sería de la forma,

```
#ifdef_OPENMP
A(i)=A(i)+b
#endif
```

de modo que esta línea solamente forma parte del programa cuando se compila con OpenMP activado.

La programación con OpenMP consta básicamente de tres elementos, el control de paralelismo, el control de la sincronización y finalmente el control de datos y comunicaciones. Para el control de paralelismo existen directivas para la creación de regiones paralelas y para el reparto de trabajo. El control de datos y comunicaciones se realiza mediante el uso de variables privadas y compartidas. Finalmente la sincronización nos permite coordinar el acceso a los datos mediante el uso de barreras, secciones críticas y otras opciones que veremos más adelante.

3.3.2 Directivas para la construcción de paralelismo

Las directivas para el control de paralelismo nos permiten crear regiones paralelas y repartir las tareas entre los hilos que hemos creado. La directiva ***parallel*** es la encargada de crear la región paralela, que es un bloque de código ejecutado en paralelo por varios hilos. Esta directiva indica en OpenMP cuando empieza una región paralela y tiene la siguiente forma:

!$OMP PARALLEL [cláusulas]
bloque que deseamos paralelizar
!$OMP END PARALLEL

Cuando un hilo encuentra la directiva ***parallel***, crea un equipo de hilos y éste se convierte en el hilo maestro del equipo. El número de hilos que se crean lo controlamos por medio de una variable de entorno ***NUM_THREADS*** o mediante una llamada a una librería.

La directiva ***end parallel*** denota el final de la región paralela. Existe una barrera implícita cuando los hilos llegan a este punto, de modo que el programa no continúa hasta que todos los hilos hayan completado la ejecución del bloque de código, salvo que exista la cláusula NOWAIT. Cuando los hilos alcanzan el final de la región paralela se destruyen y tan sólo el hilo maestro continúa la ejecución del programa.

En el caso de que un equipo de hilos se encuentre, durante la ejecución de una región paralela, con una directiva de creación de otra región paralela, cada hilo crea un nuevo equipo y se convierte en el maestro de ese nuevo equipo. La segunda región paralela se denomina región paralela anidada.

El resto de directivas que se incluyen en este apartado se emplean para el reparto de trabajo. Estas directivas se encuentran dentro de una región paralela, no crean nuevos hilos y tampoco existen barreras al inicio del reparto de tareas. El primer hilo que llega realiza el trabajo que se le asigna sin esperar a que lleguen los demás. A continuación describimos con más detalle cada una de las directivas de reparto de trabajo:

- ***DO***. Esta directiva especifica que las iteraciones del primer lazo ***DO*** situado justo después de ella se deben ejecutar en paralelo. Las iteraciones del lazo se distribuyen entre los hilos ya existentes. Si queremos elegir como se reparten los hilos las iteraciones del lazo podemos hacer uso de la cláusula ***schedule(type[,chunk])***. Existen varias formas de planificar el reparto de iteraciones en función de la sintaxis que hayamos elegido dentro de las que se muestran a continuación:

  - ***SCHEDULE(STATIC, chunk)***. Cuando especificamos esta opción las iteraciones se dividen en bloques del tamaño especificado en *chunk*. Los bloques se asignan de modo estático, de forma ordenada, entre los hilos del equipo.
  - ***SCHEDULE(DYNAMIC, chunk)***. En esta opción las iteraciones se dividen en bloques de tamaño igual al valor de *chunk*. Cuando un hilo acaba un bloque de iteraciones dinámicamente se le asigna otro bloque de tamaño igual a *chunk*.
  - ***SCHEDULE(GUIDED,chunk)***. En esta opción las iteraciones se dividen en bloques de forma que el tamaño de cada bloque decrece de forma exponencial. *Chunk* especifica el tamaño del bloque más pequeño, con excepción del último bloque que puede ser menor que *chunk*.
  - ***SCHEDULE(RUNTIME)***. En esta opción el tipo de planificación y el tamaño se puede elegir en tiempo de ejecución por medio de la variable de entorno ***OMP_SCHEDULE***.

  Cuando no se especifica la cláusula ***schedule(type[,chunk])*** la planificación depende de la implementación.

  Los hilos que completan la ejecución del lazo, deben esperar al resto de hilos que todavía no acabaron la ejecución, debido a que existe una barrera implícita en la directiva ***END DO***, salvo que exista la cláusula ***NOWAIT***.

- ***SECTIONS***. Esta es una directiva de reparto de trabajo no iterativa, que especifica que la región de código que se encuentra dentro de la directiva se reparte entre los hilos del equipo. Cada sección se ejecuta una vez por un único hilo del equipo. El formato de esta directiva es el siguiente:

```
!$OMP SECTIONS [cláusulas]
!$OMP SECTION
bloque de código
!$OMP SECTION
bloque de código
...
!$OMP END SECTIONS [NOWAIT]
```

- ***SINGLE***. Con esta directiva especificamos que el código encerrado debe ser ejecutado solo por un único hilo sin necesidad de que sea el maestro. Aquellos hilos que no ejecuten la directiva tienen una barrera implícita al final de la directiva ***SINGLE*** salvo que aparezca la cláusula NOWAIT. El formato es el que se muestra a continuación:

```
!$OMP SINGLE [cláusulas]
bloque de código
!$OMP END SINGLE
```

- ***WORKSHARE***. Esta directiva permite la paralelización de expresiones con matrices en sentencias Fortran 90. Divide las tareas asociadas al bloque en unidades de trabajo y las reparte entre los hilos. El reparto depende del compilador que tiene como única restricción que cada unidad de trabajo se ejecuta una única vez. También se puede emplear con funciones intrínsecas que operan con matrices tales como *matmul*, *dot_product*, *maxval*, etc., aunque su rendimiento dependerá del compilador. El formato de esta directiva es el que se muestra a continuación:

!$OMP WORKSHARE

bloque de código

!$OMP END WORKSHARE [NOWAIT]

Al final de la directiva existe una barrera implícita, que hace que los hilos tengan que esperar a que todos hayan completado la ejecución del código, salvo que exista la cláusula NOWAIT.

También es posible combinar las directivas para la construcción de paralelismo con las directivas para reparto de tareas, para ello tan sólo tenemos que hacer uso de la directiva ***PARALLEL*** seguida de la directiva de reparto de tareas que nos convenga. A continuación explicamos con detalle las posibles combinaciones:

- ***PARALLEL DO***. Esta directiva nos permite especificar una región paralela que contenga una única directiva ***DO***. El formato de esta directiva es el que sigue:

  !$OMP PARALLEL DO [cláusulas]

  bloque de código con lazo do

  !$OMP END PARALLEL DO

  Cuando el hilo maestro llega a esta directiva crea el equipo de hilos, tal y como haría la directiva ***PARALLEL***, y reparte las iteraciones del bucle entre los hilos, cumpliendo la función de la directiva ***DO***.

- ***PARALLEL SECTIONS***. Esta directiva es una forma simple de especificar una región paralela que contiene una única directiva ***SECTIONS***. El formato de esta directiva es el que se muestra a continuación:

  !$OMP PARALLEL SECTIONS [cláusulas]

  !$OMP SECTION

  bloque de código

!$OMP SECTION

bloque de código

. . .

!$OMP END PARALLEL SECTIONS

- ***PARALLEL WORKSHARE***. Esta directiva es una forma simple de especificar una región paralela que contiene una única directiva ***WORKSHARE***. El formato de esta directiva es el que se muestra a continuación:

!$OMP PARALLEL WORKSHARE [cláusulas]

bloque de código

!$OMP END PARALLEL WORKSHARE

3.3.3 Directivas de sincronización

Cuando disponemos de varios hilos, podemos gestionar el modo en el que cada uno accede a determinados bloques de código o los accesos a memoria. Para realizar esta tarea disponemos de directivas que nos permiten gestionar los hilos creados en las regiones paralelas. A continuación describimos cada una de estas directivas:

- ***MASTER***. El bloque de código que se encuentra entre las directivas ***MASTER*** y ***END MASTER*** tan sólo va a ser ejecutado por el hilo maestro, el resto de hilos saltan el bloque de código y continúan con la ejecución. El formato de esta directiva es el que se muestra a continuación:

!$OMP MASTER [cláusulas]

bloque de código

!$OMP END MASTER

- ***CRITICAL***. Esta directiva restringe el acceso a la región de código encerrado entre las directivas ***CRITICAL*** y ***END CRITICAL*** a un sólo hilo. El formato de esta directiva es el que se muestra a continuación:

  !$OMP CRITICAL [(nombre)]

  bloque de código

  !$OMP END CRITICAL [(nombre)]

  En donde *nombre* identifica a la sección crítica.

  Cuando un hilo alcanza esta directiva durante la ejecución, espera al principio de la sección crítica hasta que ningún otro hilo esté ejecutando la sección crítica identificada con el mismo nombre. Las secciones críticas son entidades globales del programa. Si el nombre de una sección crítica coincide con el nombre de otra entidad del programa, el comportamiento del programa es imprevisible.

- ***BARRIER***. La función de esta directiva es sincronizar todos los hilos de la región paralela. Durante la ejecución del código cada hilo que encuentra esta directiva espera hasta que todos los hilos alcanzan este punto. El formato que emplea esta directiva es:

  !$OMP BARRIER

- ***ATOMIC***. Con esta directiva nos aseguramos que una posición específica de memoria se actualiza automáticamente de forma atómica (indivisible), lo que implica que no se van a producir múltiples escrituras desde diferentes hilos. Esta directiva se aplica a la sentencia inmediatamente posterior, y debe tener la siguiente forma:

  !$OMP ATOMIC

  La sentencia en Fortran para la directiva ***ATOMIC*** debe ser de la forma:

x = x operador *expr*

x = intrínseca (expr,x)

En donde operador puede ser cualquiera de los operadores matemáticos (suma, producto, resta o división), *expr* es una expresión escalar que no hace referencia a x e intrínseca puede ser MAX, MIN, IAND o IOR.

- ***FLUSH***. Es un punto de encuentro que nos permite asegurar que todos los hilos de un equipo tienen una visión consistente de ciertas variables de memoria. Esta directiva proporciona consistencia entre operaciones de memoria del hilo actual y la memoria global. Para alcanzar consistencia global cada hilo debe ejecutar una operación ***FLUSH***. En el caso de que no especifiquemos una lista de variables se aplica a todas, pero esto podría ser computacionalmente muy costoso. La forma de esta directiva es la que se muestra a continuación:

  !$OMP FLUSH [(lista de variables)]

  Esta directiva está presente de modo implícito, salvo que se use la directiva ***NOWAIT***, en las siguientes directivas:

  - ***BARRIER***
  - ***CRITICAL*** y ***END CRITICAL***
  - ***END DO***
  - ***END SECTIONS***
  - ***END SINGLE***
  - ***END WORKSHARE***
  - ***ORDERED*** y ***END ORDERED***
  - ***PARALLEL*** y ***END PARALLEL***
  - ***PARALLEL DO*** y ***END PARALLEL DO***
  - ***PARALLEL SECTIONS*** y ***END PARALLEL SECTIONS***

  - ***PARALLEL WORKSHARE*** y ***END PARALLEL WORKSHARE***

- ***ORDERED***. Las iteraciones del código encerrado entre las directivas ***ORDERED*** y ***END ORDERED*** se ejecutan en el orden que lo harían en un programa secuencial. Cuando empleamos esta directiva dentro de lazos, la directiva ***DO*** o ***PARALLEL DO*** tienen que incluir la cláusula ***ORDERED***. La forma de esta directiva es la que se muestra a continuación:

  !$OMP ORDERED

  bloque de código

  !$OMP END ORDERED

  Esta directiva secuencializa y ordena el código dentro de la sección ***ORDERED*** a la vez que permite ejecutar en paralelo el resto.

3.3.4 Cláusulas para el control de datos

Existe un conjunto de cláusulas, generalmente en las directivas de creación de paralelismo y de reparto de trabajo, que nos permiten mantener un control sobre que hilos comparten, o no comparten, ciertas variables. A continuación hablaremos con más detalle de cada una de estas cláusulas y de la directiva para el control de datos ***THREADPRIVATE***.

- ***THREADPRIVATE***. Esta directiva declara las variables que se le indican, privadas a cada hilo pero globales dentro del hilo. Debemos especificar esta directiva cada vez que la variable es declarada. Los datos ***THREADPRIVATE*** serán indefinidos dentro de la región paralela a menos que se use la cláusula ***COPYIN*** en la directiva ***PARALLEL***. La forma de esta directiva es la que se muestra a continuación:

  !$OMP THREADPRIVATE(lista de variables)

Las cláusulas que podemos emplear para el control de datos son las siguientes:

- ***PRIVATE***. Esta cláusula declara privadas a cada hilo las variables especificadas. Cada hilo tiene una copia local de las variables, siendo ésta invisible para los demás hilos. Las variables no están definidas ni al entrar ni al salir de la región paralela, tan sólo se utilizan dentro de la región paralela. La forma de esta cláusula es la que se muestra a continuación:

  PRIVATE(lista de variables)

- ***SHARED***. Esta cláusula hace que todas las variables que aparecen en la lista se compartan entre los hilos. El valor asignado a una variable compartida es visto por todos los hilos del equipo, así todos los hilos acceden a la misma posición de memoria cuando modifican variables compartidas y este acceso puede ser concurrente. En el acceso a los datos no se garantiza la exclusión mutua. La forma de esta cláusula es la que se muestra a continuación:

  SHARED(lista de variables)

  El uso de variables compartidas está recomendado cuando tenemos variables de sólo lectura a las que se accede desde diferentes hilos, los distintos hilos acceden a localizaciones diferentes de la variable (por ejemplo en una matriz cada hilo accede a diferentes índices de la matriz) y cuando queremos comunicar valores entre diferentes hilos.

- ***DEFAULT***. Con esta cláusula podemos establecer por defecto que todas la variables de una región paralela sean privadas cuando va seguida de la cláusula ***PRIVATE*** o compartidas cuando va seguida de la cláusula ***SHARED***. En el caso de que no queramos especificar un tipo por defecto podemos hacer uso de la cláusula ***DEFAULT NONE***. De esta forma todas la variables deben clasificarse de forma explícita como privadas o compartidas. La forma de esta cláusula es la siguiente:

DEFAULT(PRIVATE | SHARED | NONE)

Cuando no hacemos uso de la cláusula ***DEFAULT*** el comportamiento por defecto es el mismo que si hubiésemos especificado ***DEFAULT (SHARED)***.

- ***FIRSTPRIVATE***. Esta cláusula nos permite declarar privadas las variables especificadas y además las inicializa a los valores que tenían antes de entrar en el la región paralela. La forma de esta cláusula es la siguiente:

  FIRSTPRIVATE(lista de variables)

- ***LASTPRIVATE***. Esta cláusula declara privadas las variables de una lista y además actualiza las variables con su último valor al terminar el trabajo paralelo. Cuando una variable se declara como ***LASTPRIVATE***, el valor de la variable al salir del bucle es el valor que toma en el procesador que ejecuta la última iteración del bucle.

- ***REDUCTION***. Indica una operación de reducción sobre las variables especificadas. La forma de esta cláusula es la que se muestra a continuación:

  REDUCTION(operador:lista de variables)

  Una operación de reducción es una aplicación de un operador binario conmutativo y asociativo a una variable y algún otro valor, almacenando el resultado en la variable, como por ejemplo, x=x+a(i). Una lista de los operadores de reducción disponibles se muestra en la tabla 3.1.

  Además podemos especificar más de una operación de reducción, pero debe ser sobre diferentes variables. El valor de la variable de reducción es indefinido desde el momento que el primer hilo alcanza al cláusula hasta que la operación se completa. Si no se utiliza ***NOWAIT*** la variable está indefinida hasta la primera barrera de sincronización.

Operador	Tipo de datos	Valor inicial
+	Entero, flotante (complejo o real)	0
*	Entero, flotante (complejo o real)	1
-	Entero, flotante (complejo o real)	0
.AND.	Lógico	.TRUE.
.OR.	Lógico	.FALSE.
.EQV.	Lógico	.TRUE.
.NEQV.	Lógico	.FALSE.
MAX	Entero, flotante (complejo o real)	El menor posible
MIN	Entero, flotante (complejo o real)	El menor posible
IAND	Entero	Todos los bits
IOR	Entero	0
IEOR	Entero	0

Tabla 3.1: Operadores de reducción en Fortran.

- ***COPYIN***. Esta cláusula sólo se aplica a variables que han sido declaradas como ***THREADPRIVATE*** y con ella podemos hacer que las variables de cada hilo tomen el mismo valor que en el hilo maestro. La forma de esta cláusula es la siguiente:

  COPYIN(lista de variables)

- ***COPYPRIVATE***. Con esta cláusula el valor de las variables privadas son copiadas a las variables privadas de los otros hilos al final de la directiva ***SINGLE***. Por consiguiente esta cláusula sólo puede usarse con esta directiva ***SINGLE***. La forma de esta cláusula es la siguiente:

  COPYPRIVATE(lista de variables)

3.4 Paralelización con OpenMP de simuladores Monte Carlo de transistores MOSFET

Antes de comenzar a paralelizar cualquier código es necesario pasar por dos procesos básicos: primero localizar las regiones de código con mayor carga computacional y a continuación comprobar si existe algún tipo de dependencia que impida o dificulte la paralelización del código. Por lo tanto, la descripción del proceso de paralelización de los simuladores la dividiremos en dos apartados. Un primer apartado en el que se realiza un perfilado del código, para analizar cuales son las subrutinas con mayor carga computacional, y un segundo apartado en el que se describe la estrategia de paralelización indicando cuales han sido las subrutinas seleccionadas.

En esta sección, se describe el proceso de paralelización del simulador 2D MV–ECBE–EMC de transistores DGSOI MOSFET (3.4.1) y del simulador 2D MSB–EMC de transistores MOSFET (3.4.2).

3.4.1 Paralelización del simulador 2D MV–ECBE–EMC de transistores MOSFET

Para llevar a cabo la paralelización del simulador 2D MV–ECBE–EMC de transistores DGSOI, descrito anteriormente, debemos hacer un estudio inicial de cuales son las subrutinas que componen el programa con mayor carga computacional. Para ello hemos realizado un perfilado del código secuencial, en donde se indica el porcentaje de tiempo con respecto al tiempo total de simulación que consume cada subrutina.

Perfilado

La tabla 3.2 muestra el perfilado del código secuencial para una simulación de 1 ps de un transistor DGSOI de 10 nm de longitud de puerta como el que se muestra en la figura 3.10. Estos resultados se han obtenido en un servidor con doble procesador Intel *Quad–Core* Xeon 5355 a 2.6 GHz con 8 GB de memoria principal que forma parte del cluster SVG del

Centro de Supercomputación de Galicia (CESGA).

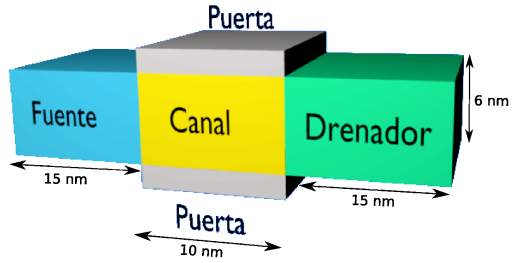


Figura 3.10: Representación de la estructura del DGSOI de 10 nm de longitud de puerta simulado. Las regiones de fuente y drenador están dopadas con una concentración de impurezas dadoras $N_D = 9$ x 10^{19} cm^{-3} y el canal tiene una concentración de impurezas aceptoras de $N_A = 10^{15}$ cm^{-3}. El grosor del óxido de puerta es de 10 Å y el grosor del silicio de 6 nm.

$\%t_{total}$	$t(s)$	*Llamadas*	(Id)
99,9	3411,75	1	*Monte3d*
31,5	1075,61	10002	*Renew*
30,2	1030,97	10002	*Ecbemc*
18,8	642,78	10001	*Free*
4,7	160,14	10002	*Charge*
3,8	129,42	10001	*Poisson*
2,9	99,17	10001	*Velocity*
1,7	56,83	10001	*Current*

Tabla 3.2: Extracto del perfilado del código secuencial realizado con el programa Gprof, para una simulación de 1 ps donde para cada subrutina *t(s)* es el tiempo en segundos que el programa invierte en la misma, *Llamadas* es el número de veces que se llama durante la ejecución e *Id* es el nombre con el cual identificamos a la subrutina.

El porcentaje de tiempo lo podemos conocer con la ayuda del programa de código abierto Gprof [gpr]. En función de estos datos, representados en la figura 3.11, hemos elegido todas las subrutinas que se muestran en la tabla 3.2 salvo la subrutina *Poisson* como candidatas para ser paralelizadas. La subrutina *Poisson*, que resuelve la ecuación de Poisson de forma numérica, no la hemos incluido porque el algoritmo de resolución empleado no es paralelizable en su forma actual debido a que existen dependencias en las iteraciones del bucle que implementa.

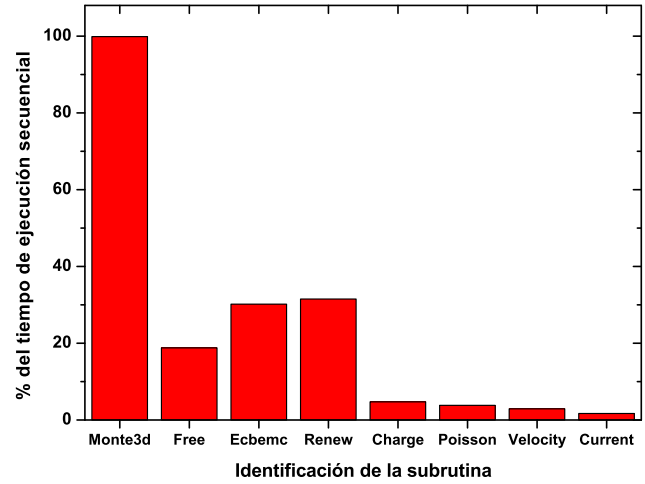


Figura 3.11: Representación gráfica del extracto del perfilado del código secuencial, mostrado en la tabla 3.2, realizado con el programa Gprof.

Como se puede apreciar en la tabla 3.2 la subrutina denominada *Monte3d*, consume el 99,9 % del tiempo de ejecución. Esta subrutina tan sólo se llama una vez y contiene las sentencias y llamadas necesarias para completar la simulación Monte Carlo, por lo que se convierte en la subrutina perfecta para crear los hilos y que cada uno llame a las subrutinas que han sido paralelizadas.

La subrutina *Free* consume el 18,8 % del tiempo de ejecución y es llamada 10001 veces desde la subrutina *Monte3d*. La finalidad de *Free* es simular el transporte de partículas dentro del semiconductor. El número de partículas varía en cada iteración temporal y se sitúa en torno a las 180000 para el caso estudiado.

La subrutina *Ecbemc* consume el 30,2 % del tiempo de ejecución y recibe 10002 llamadas desde la subrutina *Monte3d*. Su función es realizar la corrección cuántica del potencial calculado previamente por la subrutina específica encargada de resolver la ecuación de Poisson.

La subrutina *Renew* consume el 31,5 % del tiempo de ejecución y es llamada 10002 veces desde la subrutina *Monte3d*. Esta subrutina hace un recuento del número de partículas que quedan en el dispositivo después de simular su transporte, determinando las que han salido del dominio de simulación. Además, se encarga de añadir las partículas necesarias a los contactos con la finalidad de mantener la neutralidad eléctrica.

La subrutina *Charge* consume el 4,7 % del tiempo de ejecución y recibe 10002 llamadas desde la subrutina *Monte3d*. Esta subrutina calcula la carga asociada a cada uno de los nodos de la malla que discretiza el dispositivo para su simulación.

La subrutina *Velocity* consume el 2,9 % del tiempo de ejecución y recibe 10001 llamadas desde la subrutina *Monte3d*. Su función es calcular la velocidad de las partículas simuladas para posteriormente calcular la corriente.

Finalmente, la subrutina *Current* consume el 1,7 % del tiempo de ejecución y recibe 10001 llamadas desde la subrutina *Monte3d*. Hace un cálculo de la corriente del dispositivo simulado, en función del número de partículas que entran y salen del mismo durante la simulación.

Paralelización

El diagrama de bloques de la figura 3.12 representa la paralelización del simulador 2D MV–ECBE–EMC de transistores DGSOI MOSFET. En la estrategia de paralelización propuesta se crea la región paralela justo antes de la evaluación del tiempo de simulación, en la subrutina *Monte3d*.

De esta forma se consigue que cada hilo, creado antes de la llamada a las subrutinas, evalúe el instante temporal en el que se encuentra la simulación y posteriormente haga la llamada y ejecute el código de modo independiente en cada intervalo temporal. La finalidad de esta propuesta es evitar la creación y destrucción de hilos que supondría crear una región paralela para cada subrutina paralelizada.

El inconveniente de esta técnica es que la simulación del transporte es un proceso secuencial y los resultados obtenidos durante el cálculo de una subrutina son empleados en la siguiente y además no todas las subrutinas están paralelizadas. Por lo tanto, es necesario establecer una sincronización entre los hilos para evitar que se produzcan llamadas a las siguientes subrutinas antes de que todos los hilos hayan abandonado la anterior, como se puede ver en la figura 3.13.

Además de los problemas de sincronización otro problema a tener en cuenta es que dentro de la subrutina *renew* existe código que es secuencial y que sólo puede ser ejecutado por un único hilo. Por lo tanto tenemos zonas en donde el hilo principal debe ejecutar un bloque de código y el resto de hilos deben esperar a que éste acabe.

La existencia de regiones que son ejecutadas por un único proceso y la necesidad de usar barreras limitan la eficiencia de la ejecución paralela, debido a que los hilos que ya han completado sus tareas deben esperar a que todos hayan acabado antes de iniciar la ejecución del siguiente bloque de código paralelizado.

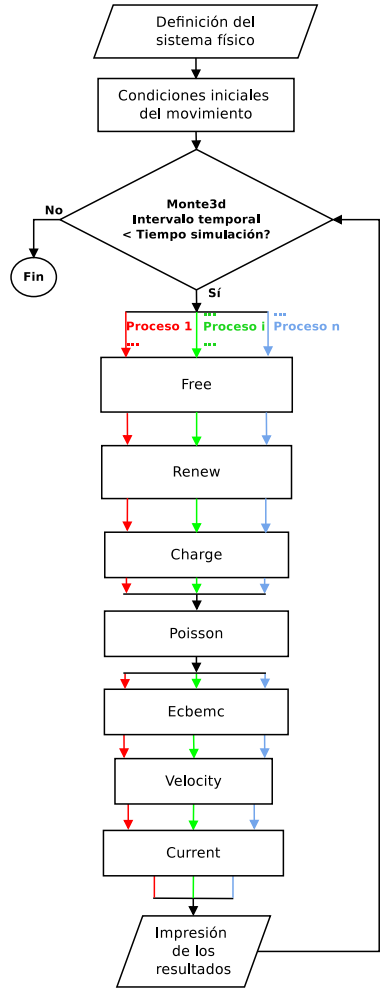


Figura 3.12: Diagrama de bloques de la paralelización del simulador 2D MV–ECBE–EMC de transistores MOSFET.

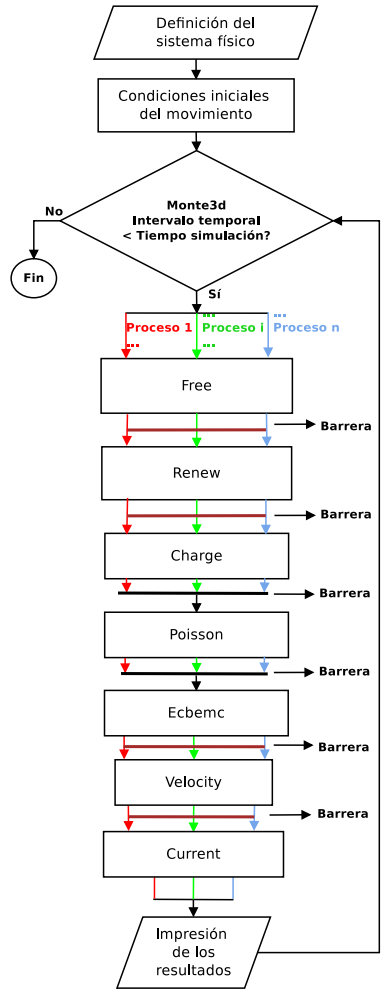


Figura 3.13: Diagrama de bloques de la paralelización del simulador 2D MV–ECBE–EMC de transistores MOSFET indicando la posición de las barreras.

3.4.2 Paralelización del simulador 2D MSB–EMC de transistores MOSFET

Como se ha mencionado en la sección 2.4, el simulador 2D MSB–EMC es una evolución del simulador 2D MV–ECBE–EMC. En este caso, la resolución de la ecuación de Schrödinger en la dirección de confinamiento para cada una de las rodajas en las que se divide el dispositivo es una de las principales ventajas para la paralelización de este código. La subrutina que resuelve esta ecuación es completamente paralelizable y como veremos a continuación en los datos del perfilado tiene una carga computacional muy importante.

La paralelización del simulador 2D MSB–EMC se ha realizado siguiendo el mismo procedimiento que se ha indicado en el apartado 3.4.1. Inicialmente hemos seleccionado las subrutinas con mayor carga computacional a partir de los datos que hemos obtenido del perfilado del código. A continuación analizamos si es posible la paralelización de las subrutinas más costosas y finalmente, en caso de que esto sea posible, se realiza la paralelización.

Perfilado

La tabla 3.3 representa el porcentaje de tiempo total de computación consumido por las principales subrutinas del simulador 2D MSB–EMC para una simulación de 1 ps, de un transistor DGSOI de 10 nm de longitud de puerta como el que se muestra en la figura 3.14. Los datos mostrados corresponden a diferentes valores de la variable *scsc* que indica cada cuantas iteraciones temporales se resuelve de forma autoconsistente la ecuación de Schrödinger y se calcula la tabla de dispersión, tal y como se ha descrito en la sección 2.4. Estos resultados se han obtenido tras haber realizado varias ejecuciones para diferentes valores de la variable *scsc* en un servidor con doble procesador Intel *Quad–Core* Xeon E5410 a 2.33GHz con 8 GB de memoria principal que forma parte de un cluster PowerEdge M1000e.

A continuación se introduce brevemente la función que desempeña cada una de las subrutinas de la tabla 3.3:

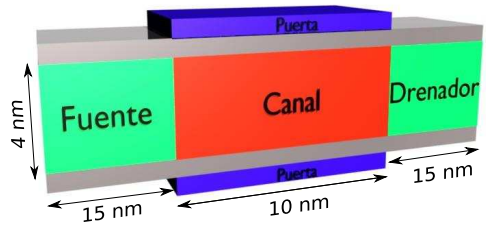


Figura 3.14: Representación de la estructura del dispositivo DGSOI simulado de 10 nm de longitud de puerta. Las regiones de fuente y drenador están dopadas con una concentración de impurezas dadoras $N_D = 10^{20}$ cm^{-3} y el canal tiene una concentración de impurezas aceptoras de $N_A = 10^{15}$ cm^{-3}. El grosor del óxido de puerta es de 1 nm y el grosor del silicio de 4 nm.

La subrutina *Multi_ini* obtiene una solución inicial de la ecuación de Schrödinger antes de iniciar la simulación Monte Carlo. Las funciones de onda asociadas a esta solución inicial se emplean para calcular la tabla de dispersión inicial en la subrutina *Sctable*.

La subrutina *Free* simula el transporte de partículas dentro del dispositivo. El número de partículas simuladas en cada iteración temporal varía en función del número de partículas que salgan por los contactos de fuente y drenador.

La subrutina *Flux* evalúa el flujo de electrones para cada uno de los cortes transversales a la dirección de transporte en los que se divide el dispositivo simulado.

La subrutina *Charge* calcula la carga asociada a cada uno de los nodos de la malla que discretiza el dispositivo para su simulación.

	scsc=1		*scsc*=8		*scsc*=64	
Subrutina	$\%t_{total}$	*Llamadas*	$\%t_{total}$	*Llamadas*	$\%t_{total}$	*Llamadas*
Main	100	1	100	1	100	1
Multi_ini	2,70	1	16,89	1	54,64	1
Free	0,69	1001	3,98	1001	12,86	1001
Flux	0,01	1001	0,08	1001	0,25	1001
Charge	0,08	1002	0,54	1002	1,72	1002
Velmu	0,05	1001	0,29	1002	0,94	1002
Poisson	0	1001	0,01	1001	0,03	1001
Multisub	37,98	1001	30,27	125	11,35	15
Sctable	58,48	1002	47,91	126	18,11	16
Current	0,01	1001	0,03	1001	0,09	1001

Tabla 3.3: Extracto del perfilado del código secuencial realizado para tres valores diferentes de la variable *scsc*, para una simulación de 1 ps, donde $\%t_{total}$ es el porcentaje del tiempo total de simulación que el programa invierte en la subrutina y *Llamadas* es el número de veces que se llama a la subrutina durante la ejecución.

La subrutina *Velmu* determina la velocidad y movilidad media de las partículas simuladas.

La subrutina *Poisson* resuelve de forma numérica la ecuación de Poisson que nos permite obtener el valor del potencial electrostático en cada nodo de la malla.

La subrutina *Multisub* obtiene la solución de la ecuación de Schrödinger cada vez que se llama durante la simulación Monte Carlo.

La subrutina *Sctable* determina la tabla de dispersión a partir de los valores de la fución de onda que se obtienen de la solución de la ecuación de Schrödinger.

Finalmente, la subrutina *Current* calcula los valores de corriente del dispositivo simulado por dos métodos: en función del número de partículas que entran y salen del dispositivo durante la simulación, y a partir del

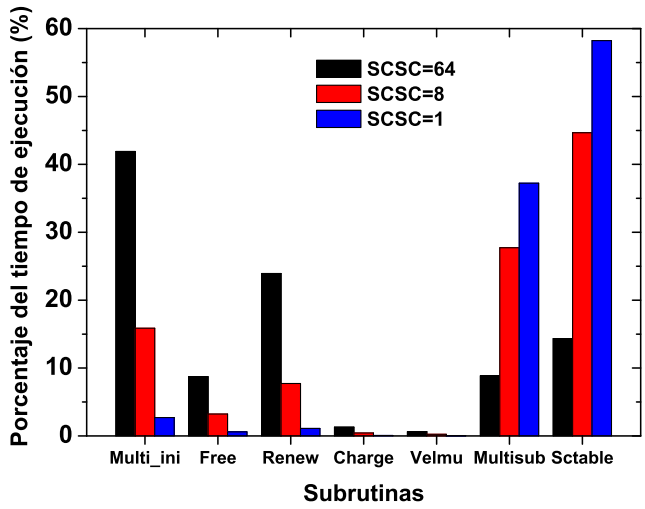


Figura 3.15: Representación gráfica de las subrutinas más costosas del simulador 2D MSB–EMC para diferentes valores de la variable *scsc*.

teorema de Ramo–Shockely [Ram39, Sho38, KMTP91].

La figura 3.15 representa el porcentaje de tiempo de computación consumido por las subrutinas más costosas de la tabla 3.3. Como se aprecia en la figura, la variable *scsc* nos permite modificar la carga computacional de las subrutinas, *Multisub* y *Sctable*, las cuales alcanzan su valor máximo cuando *scsc*=1 y el número de llamadas coincide con el número de iteraciones temporales de la simulación 2D MSB–EMC.

En función de los datos de la tabla 3.3 y de la figura 3.15, hemos elegido las subrutinas *Multi_ini*, *Free*, *Multisub* y *Sctable* como candidatas para ser paralelizadas debido a su elevado peso computacional.

La figura 3.16 representa el comportamiento del peso computacional de las subrutinas seleccionadas cuando se cambia la variable de autoconsistencia *scsc*.

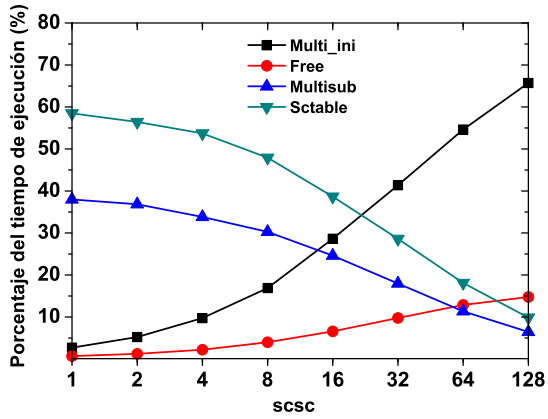


Figura 3.16: Dependencia del porcentaje de tiempo de ejecución frente a *scsc* para las subrutinas paralelizadas en el simulador 2D MSB–EMC.

Las subrutinas *Multisub* y *Sctable* son dos subrutinas fundamentales del simulador 2D MSB–EMC y el número de veces que se llaman durante la ejecución del código está determinado por la siguiente expresión,

$$llamadas = \frac{T_{sim}}{scsc \cdot dt} \tag{3.1}$$

en donde T_{sim} es el tiempo que dura la simulación, dt es el paso temporal en el que se divide la simulación Monte Carlo y *scsc* es la variable que determina la autoconsistencia. En el caso de los resultados mostrados en la figura 3.16 el tiempo de simulación es 1 ps y el paso temporal es igual a 1 fs, lo que implica un total de 1000 llamadas para un valor de *scsc* igual a uno.

Del análisis de la figura podemos observar que el peso computacional de la subrutina *Multisub* es dependiente de la variable *scsc* y oscila entre el 38 y el 6 % del tiempo total de ejecución para valores de *scsc* igual a 1 y 128 respectivamente. En el caso de la subrutina *Sctable* podemos

comprobar que se trata de una de las subrutinas más costosas de la simulación Monte Carlo, con un peso computacional del 59 % del tiempo total de ejecución cuando el valor de la variable *scsc* es igual a 1 y del 10 % del tiempo total de ejecución cuando la variable *scsc* toma el valor 128. La subrutina *Multi_ini* tan sólo se llama una vez en el código antes de iniciar la simulación Monte Carlo y su peso computacional es dependiente de la autoconsistencia. Como se puede ver en la figura 3.16 su peso computacional oscila entre el 3 % cuando la autoconsistencia es muy alta, $scsc = 1$, y el 65 % cuando la ecuación de Schrödinger se resuelve muy pocas veces, $scsc = 128$. Finalmente, en el caso de la subrutina *Free* el número de llamadas que recibe es igual al cociente de T_{sim}/dt y por lo tanto es independiente de la variable *scsc*. Sin embargo, el peso computacional de esta subrutina también depende indirectamente de la variable *scsc* y oscila, como se muestra en la figura 3.16, entre el 0,5 % cuando la variable *scsc* es igual a 1 y el 15 % cuando *scsc* vale 128.

De los resultados mostrados en la figura 3.16 del perfilado de las cuatro subrutinas más costosas del código, podemos observar que las subrutinas cuyo número de llamadas no depende de la variable *scsc*, *Multi_ini* y *Free*, aumentan su peso computacional cuando se reduce el número de llamadas (incremento de la variable *scsc*) de aquellas subrutinas que sí dependen de esta variable, *Multisub* y *Sctable*.

Paralelización

El diagrama de bloques de la figura 3.17 muestra la estrategia de paralelización en donde las líneas de colores representan los hilos paralelos. Las subrutinas encargadas de la definición del sistema y del cálculo de las condiciones iniciales se ejecutan de forma secuencial ya que no es rentable paralelizarlas debido a que su peso computacional es muy pequeño. Una vez que la ejecución alcanza la subrutina *Multi_ini* se crea una región paralela durante la llamada a la subrutina que se destruye cuando se completa su ejecución. A continuación el código continúa su ejecución de forma secuencial hasta que alcanza la subrutina *Monte2d* que contiene el código y las subrutinas fundamentales para realizar la simulación Monte Carlo

de las partículas simuladas en cada paso temporal. Dentro de esta subrutina y antes de iniciar el lazo de iteraciones temporales de la simulación Monte Carlo, se crea una región paralela en donde todos los hilos evalúan la condición del lazo. Si la condición se cumple todos ellos entran en el lazo y ejecutan las subrutinas que están paralelizadas, *Free*, *Multisub* y *Sctable*, repartiéndose la carga computacional de los bucles paralelizados. Aquellas subrutinas que no están paralelizadas, *Poisson* y *Electric Field* son ejecutadas por el hilo maestro, mientras que el resto de hilos esperan a que se complete la ejecución antes de iniciar la llamada a la subrutina siguiente.

Cuando se alcanza el tiempo de simulación que hemos indicado en el fichero de entrada, se destruye la región paralela y se escriben en el disco duro los resultados obtenidos.

Esta estrategia de simulación requiere el uso de barreras para sincronizar los hilos debido a que el hilo maestro tiene que ejecutar más código que el resto. Dada la estructura secuencial del código, en donde el estado del sistema simulado en el instante t es la condición inicial de la simulación del instante $t+dt$ (siendo dt el paso temporal), hemos considerado que el uso de barreras era inevitable.

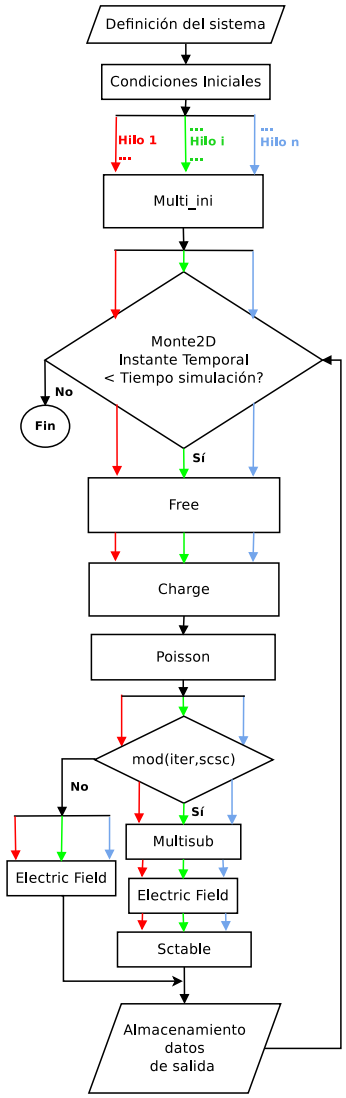


Figura 3.17: Diagrama de bloques del simulador 2D MSB–EMC paralelizado.

3.5 Medidas de la aceleración de los simuladores paralelizados

En esta sección se muestran los resultados de la aceleración de los simuladores paralelizados en función del número de núcleos empleados. Los valores de la aceleración los hemos calculado haciendo el cociente del tiempo de ejecución secuencial por el tiempo de ejecución paralelo. Las ejecuciones que hemos realizado y que se muestran a continuación evidencian la dependencia de la aceleración de los simuladores Monte Carlo paralelizados con diferentes parámetros de la simulación Monte Carlo.

3.5.1 Aceleración del simulador 2D MV–ECBE–EMC de transistores MOSFET

La tabla 3.4 muestra los resultados de la ejecución del código 2D MV–ECBE–EMC paralelizado para una simulación de 1 ps. El valor de la aceleración ideal (A_{id}) es el obtenido a partir de la ley de Amdahl [Amd67]:

$$A_{id} = \frac{1}{(\frac{P}{n}) + (1 - P - P_1) + (\frac{P_1}{n_1})} \tag{3.2}$$

en donde P es el porcentaje de programa paralelizado y n es el número de hilos utilizados. Los términos P_1 y n_1 los hemos empleado para el caso especial de la subrutina *Ecbemc* que cuenta con un bucle paralelizado de tan sólo 3 iteraciones. En este caso P_1 es el porcentaje de código paralelizado correspondiente a esta subrutina y n_1 el número de procesadores que se reparten las iteraciones de ésta, cuyo valor máximo es tres.

Para conocer cual es el porcentaje de programa paralelizado hemos empleado los datos de la tabla 3.2, que nos muestra el peso computacional de cada una de las subrutinas del programa secuencial. Así P será, asumiendo que el 100 % del código de la subrutina ha sido paralelizado, la suma de los porcentajes de las subrutinas *Free*, *Charge*, *Velocity* y *Current*, más el 50 % del porcentaje de la subrutina *Renew*, debido a que esta subrutina tan sólo tiene paralelizado aproximadamente el 50 % de su código.

Proc	$t_{ejec}(s)$	A	A_{id}
1	3795	1	1
2	2664	1,40	1,59
4	2061	1,81	2,14
6	2002	1,87	2,32
8	1973	1,89	2,43
$t_{sec}(s)$	3738		

Tabla 3.4: Resultados de la ejecución en el SVG del código 2D MV–ECBE–EMC paralelizado, para una simulación de 1 ps, donde $t_{sec}(s)$ es el tiempo de ejecución secuencial, $t_{ejec}(s)$ representa el tiempo de ejecución paralelo, A es la aceleración y A_{id} la aceleración ideal.

La figura 3.18 representa los datos de la tabla 3.4. En esta figura se observa que la aceleración conseguida alcanza el valor límite de 1,81 para 4 núcleos. La reducción del tiempo de ejecución del simulador 2D MV–ECBE–EMC en un factor 1,81 es una mejora considerable desde el punto de vista de la simulación. Sin embargo desde el punto de vista computacional la figura 3.18 indica que el porcentaje de código secuencial es todavía demasiado elevado, y la aceleración se mantiene plana para más de 4 núcleos. Además, esta gráfica también muestra que seguramente hemos sobrestimado el porcentaje de código paralelizado ya que la aceleración ideal predice valores mayores que los resultados obtenidos.

Una evidencia de que el porcentaje de código paralelo es todavía demasiado bajo, la obtenemos del estudio del comportamiento de la aceleración del simulador 2D MV–ECBE–EMC en función del número de superpartículas. En el capítulo 2 hemos visto que el método Monte Carlo aplicado al transporte de carga en semiconductores se basa en la simulación del movimiento de las partículas a través del dispositivo. Debido a la enorme carga computacional que esto supondría se simulan grupos de partículas como si fueran una única, con una carga equivalente al conjunto, que se denominan superpartículas. Inicialmente se ha fijado su carga

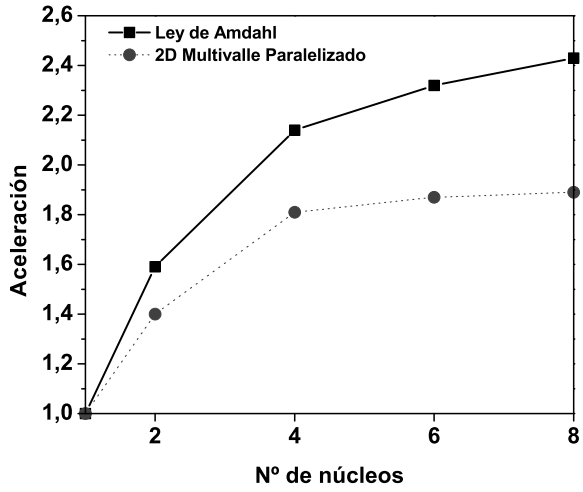


Figura 3.18: Representación de la aceleración frente al número de núcleos.

al valor de 100.000 electrones por superpartícula (eps).

En cada una de las subrutinas paralelizadas, salvo la subrutina *Ecbemc* el lazo paralelizado depende del número de superpartículas, por consiguiente podemos variar la carga computacional de estas subrutinas modificando el número de electrones que representa cada superpartícula. Así pues, por ejemplo, cuando cada superpartícula represente la carga de 50.000 electrones tendremos el doble de superpartículas que cuando represente la de 100.000 electrones.

En la figura 3.19 podemos ver como varía el tiempo de ejecución normalizado del programa sin paralelizar frente al número de superpartículas normalizado. Los valores representados están normalizados por los valores obtenidos para una simulación en la que a cada superpartícula se le asignan 100.000 electrones. De este modo en la figura podemos observar que cuando duplicamos el número de superpartículas, el tiempo de ejecución es casi 1,5 veces mayor que el tiempo de ejecución que hemos tomado como

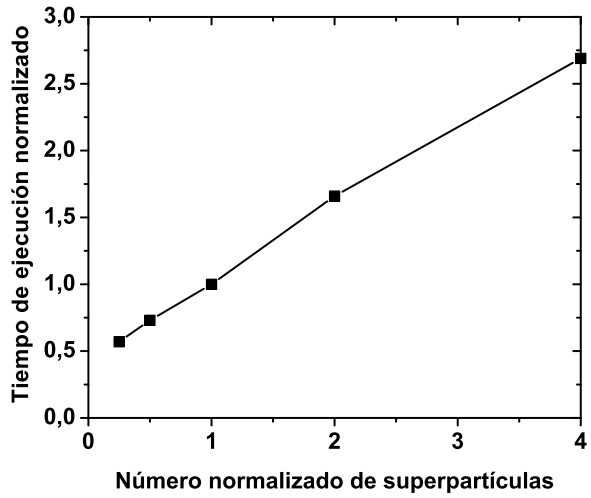


Figura 3.19: Tiempo de ejecución frente al número de superpartículas para el simulador 2D MV–ECBE–EMC sin paralelizar. Los valores del tiempo de ejecución y del número de superpartículas han sido normalizados a los valores obtenidos para una simulación de 100.000 eps.

referencia. Cuando cuadruplicamos el número de superpartículas observamos que el tiempo de ejecución es casi 2,75 veces el tiempo de ejecución de referencia. Con los resultados obtenidos se muestra que si duplicamos el número de superpartículas no se duplica el tiempo de ejecución y por lo tanto la aceleración. Esto es debido a que el incremento del número de superpartículas no afecta a todas las subrutinas del programa de igual manera.

En la figura 3.20 se representan los valores de la aceleración frente al número de procesadores cuando duplicamos y cuadruplicamos el número de superpartículas con respecto al valor que hemos tomado como referen-

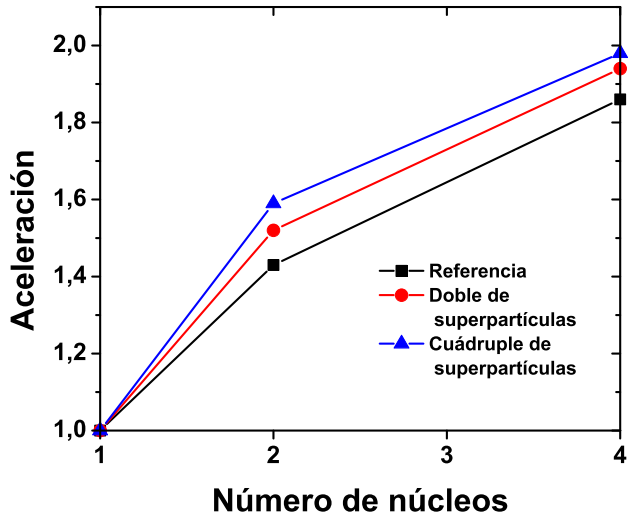


Figura 3.20: Aceleración frente al número de núcleos para diferente número de superpartículas. El valor tomado como referencia es de 100.000 eps.

cia. De los resultados representados se observa que para incrementos en el número de superpartículas de dos y de cuatro veces el valor tomado como referencia, los incrementos en la aceleración se sitúan en torno al 10 % y al 15 % por ciento con respecto al valor de referencia. Estos resultados y los representados en la figura 3.19 han sido obtenidos con un servidor con doble procesador Intel *Dual–Core* Xeon a 3 GHz con 8 GB de memoria principal.

3.5.2 Aceleración del simulador 2D MSB–EMC de transistores MOSFET

Las subrutinas que hemos elegido para paralelizar el código son las que se mostraron anteriormente, *Multisub*, *Sctable*, *Multi_ini* y *Free*. Si sumamos el peso computacional de estas subrutinas podemos aproximar

el porcentaje total del peso computacional paralelizado. Con este dato y a partir de la ley de Amdahl (3.3) podemos determinar la aceleración máxima teórica que se puede alcanzar en función del número de núcleos (n) y del porcentaje de código paralelizado (P).

$$\text{Aceleración} = \frac{1}{(1-P) + \frac{P}{n}} \tag{3.3}$$

La figura 3.21 representa la aceleración teórica y el porcentaje de código paralelizado en función de la variable *scsc*. Como se puede observar en la figura, el porcentaje de código paralelizado disminuye cuando se incrementa la variable *scsc* es decir, cuando disminuye la autoconsistencia y se reduce el número de llamadas a las subrutinas *Multisub* y *Sctable*. Esta reducción del porcentaje de código paralelizado produce un descenso de la aceleración que es mayor cuando se incrementa el número de núcleos.

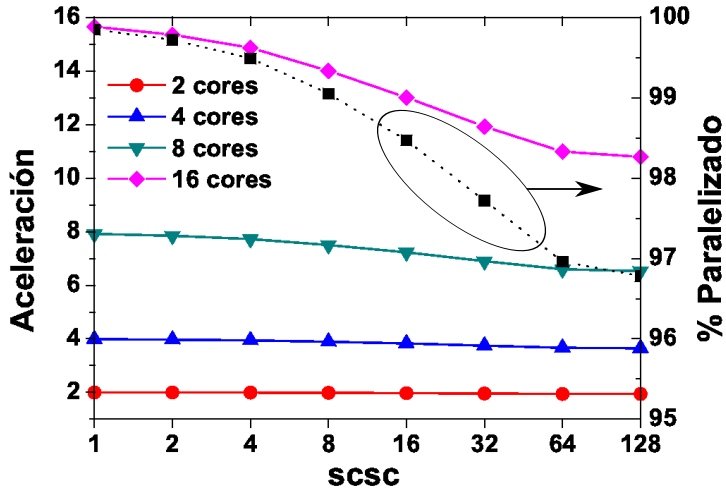


Figura 3.21: Estimación teórica de la aceleración que se puede alcanzar en función del porcentaje de código paralelizado.

Basándonos en los datos de la figura 3.21 hemos elegido el valor de la variable *scsc*=1 para comparar la aceleración del simulador 2D MSB–EMC paralelizado con los valores teóricos estimados para ese caso. Según los datos mostrados en la figura 3.21 las simulaciones con el valor de la variable *scsc*=1 son las que tienen mejor escalabilidad y mayor aceleración. Para esta caso la autoconsistencia, entre la ecuación de Schrödinger, la tabla de dispersión y la solución de la ecuación de Poisson, se realiza en cada paso temporal. Además esta condición es la más óptima desde el punto de vista físico de la simulación.

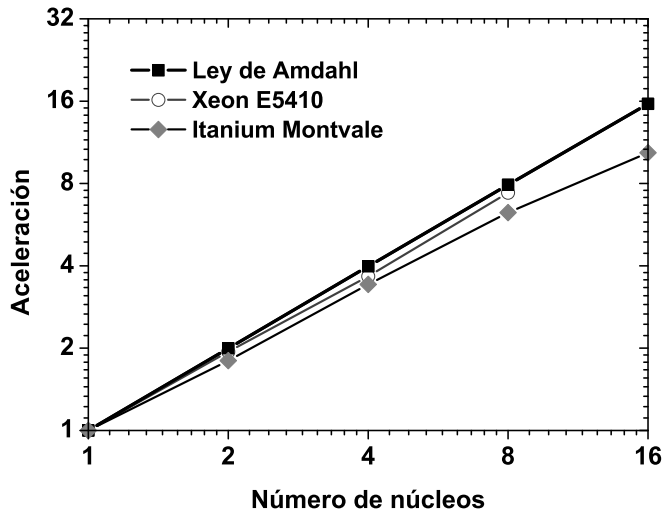


Figura 3.22: Comparación de la aceleración teórica estimada a partir de la ley de Amdahl y la aceleración real obtenida en dos nodos multiprocesador diferentes. En concreto hemos empleado un nodo con dos procesadores Intel *Quad–Core* Xeon E5410 y el otro nodo con ocho procesadores Intel Itanium Montvale.

La figura 3.22 representa la aceleración teórica estimada con la ley de Amdahl, para la simulación del apartado 3.4.2, y la aceleración real. Estos datos se han obtenido en dos nodos multiprocesador diferentes. Uno de ellos cuenta con dos procesadores Intel *Quad–Core* Xeon E5410, y el otro nodo pertenece al supercomputador Finisterrae del Centro de Supercomputación de Galicia (CESGA), con ocho procesadores Intel Itanium Montvale, y ha sido seleccionado para poder extender el estudio a 16 núcleos. Los resultados muestran que la aceleración del simulador 2D MSB–EMC escala de forma lineal con el número de núcleos utilizados y que los valores de la aceleración real obtenidos están muy cerca de los valores estimados de forma teórica. Esta mínima desviación de la aceleración real obtenida con respecto a los valores teóricos estimados, puede tener su origen en la sobrecarga computacional de las regiones paralelas y de las barreras necesarias para mantener la sincronización entre los hilos. Otra posibilidad es que hayamos sobrestimado ligeramente el porcentaje de código paralelizado. De cualquier manera, estos resultados muestran que el simulador 2D MSB–EMC es más paralelizable que el simulador 2D MV–ECBE–EMC y por lo tanto, su eficiencia en máquinas multi–núcleo es considerablemente mayor.

En la figura 3.23 podemos observar el impacto que tiene la paralelización en el tiempo de ejecución del simulador 2D MSB–EMC en el caso de la simulación estudiada anteriormente. En esta figura 3.23 se representa el tiempo de ejecución en función del número de núcleos para diferentes simulaciones de un estado estacionario de 1ps de duración, con valores diferentes de la variable *scsc*. Si consideramos una simulación secuencial para un valor de *scsc*=1 cuyo tiempo de ejecución es aproximadamente 8000 segundos es posible reducir el tiempo de ejecución a aproximadamente 1000 segundos cuando usamos 8 núcleos. En caso de que necesitáramos hacer una simulación de las mismas características, en un tiempo inferior a 1000 segundos en una máquina secuencial tendríamos que disminuir la autoconsistencia a valores de *scsc* superiores a ocho.

Estos resultados ponen de manifiesto las ventajas del paralelismo en el campo de la simulación a la hora de obtener resultados numéricos de

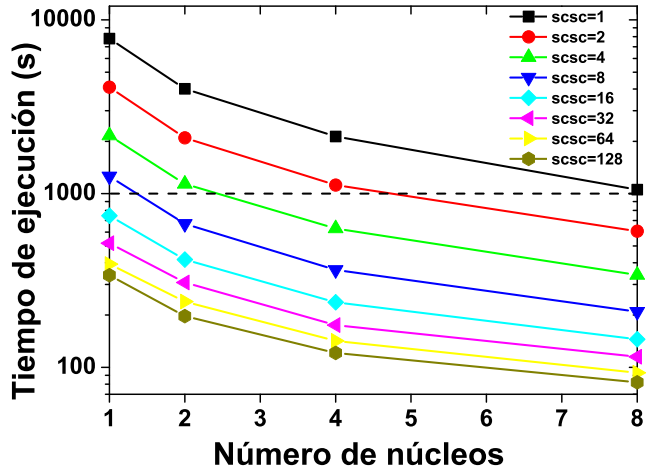


Figura 3.23: Comportamiento del tiempo de ejecución en función del número de núcleos para diferentes simulaciones de un estado estacionario de 1 ps de duración, con valores diferentes de la variable *scsc*.

los experimentos que queremos realizar. Además el hecho de que los simuladores sean más rápidos permite un ahorro considerable de tiempo a los desarrolladores de código que implementan nuevos modelos físicos en el simulador, debido a la disminución del tiempo de ejecución de las simulaciones de prueba. Sin duda la implementación de códigos paralelos permite obtener un mayor número de resultados en menos tiempo.

CAPÍTULO 4

Computación Grid aplicada a la simulación de nanodispositivos semiconductores

El Grid es una infraestructura reciente con apenas 15 años de vida. Gracias al desarrollo de nuevas tecnologías y la creación de nuevos estándares durante los últimos 10 años ha sido posible su implementación así como, la mejora de la funcionalidad y calidad de servicio. Este tipo de sistema distribuido permite la interconexión a través de Internet de una gran cantidad de sistemas, permitiendo al usuario el acceso a un conjunto muy variado de recursos que pueden estar interconectados.

En este capítulo evaluamos las ventajas y posibilidades que ofrece el Grid para la simulación de nanodispositivos, en concreto en la realización de estudios estadísticos para los que es necesario realizar cientos o miles de simulaciones. Para ello, inicialmente definimos el concepto de Grid y analizamos cual ha sido la evolución desde su creación. A continuación describimos la arquitectura y cuales son las tecnologías que facilitan su implementación y uso. Finalmente, presentamos la iniciativa Grid española y evaluamos las ventajas de usar sus recursos computacionales para la simulación de nanodispositivos, mediante la ejecución de varias pruebas

realizadas con el simulador 2D MV–ECBE–EMC.

4.1 Introducción. Definición del entorno Grid

El termino Grid surgió a mediados de los años 90 para denominar, en esa época, a las infraestructuras de computación distribuida destinadas a ciencia e ingeniería. En 1998 Ian Foster y Carl Kesselman definen el Grid como una infraestructura *hardware* y *software* que proporciona acceso a recursos computacionales de forma fiable, compatible, accesible y barata [FK04]. Según los autores de la definición esta infraestructura engloba tanto recursos de cálculo computacional, como datos o sensores conectados entre sí, con el objetivo de proporcionar un acceso universal a éstos. La figura 4.1 muestra una representación gráfica del concepto de infraestructura Grid.

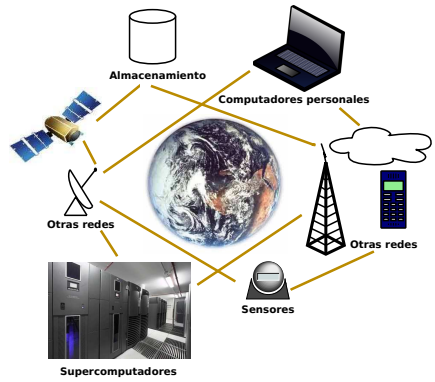


Figura 4.1: Esquema de una infraestructura Grid.

Esta definición incluía los principales requisitos para crear un Grid. Era necesaria una infraestructura fiable de modo que los usuarios pudieran asegurarse un determinado nivel de rendimiento de los recursos que la constituían. También debía ser compatible, de modo que garantizara una serie de procedimientos estándar para el acceso a los recursos, acce-

sible para que los usuarios pudieran utilizarla desde cualquier lugar sin necesidad de un sistema o dispositivo específico, y barata para que todo el mundo tuviera acceso por un precio razonable.

El desarrollo e implementación de este tipo de infraestructuras puso de manifiesto la necesidad de especificar quienes tenían acceso a ellas y el tipo de recurso al que podían acceder. Por lo tanto, en el año 2000 Ian Foster y Steve Tuecke [Fos01] refinaron la definición anterior con la intención de tratar cuestiones relacionadas con la gestión de los recursos. Para ello introdujeron el concepto de organización virtual, estableciendo que la computación Grid está relacionada con la resolución de problemas y compartición de recursos de un modo coordinado en organizaciones virtuales dinámicas y multi–institucionales.

Finalmente en el año 2002 Ian Foster aglutinó las dos definiciones anteriores en una lista de tres puntos que resume en esencia qué es un sistema Grid [Fos02]. La nueva definición precisa que se trata de un sistema que cumple las siguientes especificaciones:

1. Tiene recursos coordinados que no están sujetos a un control centralizado. A diferencia de un sistema de gestión local, el Grid integra y coordina recursos y usuarios que pertenecen a diferentes ámbitos. Por ejemplo, diferentes unidades administrativas de la misma compañía que puede que compartan recursos o tengan acceso a recursos diferentes. Por lo tanto, debe tratar los problemas relacionados con las políticas de seguridad, acceso, gestión de recursos y cobro del coste asociado al uso de éstos.

2. Emplea protocolos e interfaces estándar, abiertos y de propósito general. Estos sistemas se construyen en base a unos protocolos e interfaces orientados a resolver diferentes cuestiones técnicas como la autenticación y autorización de usuarios o el reconocimiento y acceso a los recursos. Por consiguiente, parece lógico pensar que estos sistemas se desarrollen en base a unos protocolos e interfaces estándar que permitan la interconexión de diferentes sistemas.

3. Cuenta con calidades de servicio complejas. El Grid permite que sus

recursos constituyentes se empleen de modo coordinado y ofrezcan calidades de servicio relacionadas con diferentes tipos de servicio, tales como el tiempo de respuesta, rendimiento, disponibilidad o seguridad.

A continuación vamos a presentar la evolución de esta infraestructura.

4.2 Evolución histórica del Grid

La idea del Grid surgió en la época del desarrollo de los *cluster* de computadores, del rápido crecimiento de Internet y de la creación de los primeros sistemas de computación voluntaria como Distributed.net [Dis] o el proyecto SETI@HOME [SET].

El proyecto Distributed.net se creó con la finalidad de romper el algoritmo de encriptación RSA de 56 bits mediante el uso de la computación voluntaria [Dis].

El proyecto SETI@HOME es un experimento científico que emplea computadores conectados a Internet en busca de inteligencia extraterrestre. El éxito de este proyecto se encuentra en que emplea computación voluntaria, de modo que explota los ciclos de los procesadores de aquellos usuarios que desean unirse al proyecto. Para realizar esta tarea, se ha desarrollado una aplicación denominada BOINC [BOI] que usa el tiempo de inactividad del computador. Cuando un usuario del proyecto se conecta a Internet la aplicación solicita a un servidor una tarea que se encuentra en espera para ser procesada, ésta se ejecuta en el computador del usuario aprovechando los tiempos de inactividad del computador y devuelve el resultado una vez que ha sido completada.

Este tipo de proyectos mostraban las ventajas de una infraestructura de computación distribuida de gran tamaño que fuese compartida. Ésta se podría construir de modo voluntario o a través de donaciones a cambio de emplear la infraestructura. Para hacer de esta idea una realidad fue necesario resolver una serie de cuestiones técnicas y de estandarización. En este sentido Ian Foster y Karl Kesselman [FKNT02] fueron los primeros en definir y denominar esta infraestructura con el término Grid.

La creación del GGF (*Global Grid Forum*), un foro que reunía usuarios y desarrolladores con el objetivo de impulsar el Grid dentro de las comunidades académica y de investigación, junto con el desarrollo de la EGA (*Enterprise Grid Aliance*), un consorcio creado con el objetivo de desarrollar y promocionar las infraestructuras Grid empresariales, establecieron los primeros pasos para la creación de estándares. En el 2006 estas dos organizaciones se unieron para dar lugar al Open Grid Forum (OGF) y crear de esta forma una única organización para la creación de estándares válidos tanto para la empresa como para la comunidad científica.

La primera versión de una tecnología basada en estándares que ofrecía un conjunto de servicios que posibilitaban la creación de un Grid, fue la versión 1 de Globus Toolkit [Too, Fos01, FKNT02] en el año 1998. La evolución de este *software*, hasta llegar a su versión 5 liberada en julio de 2010, ha impulsado durante esta última década la implementación de diferentes infraestructuras Grid, tanto en el ámbito académico como empresarial. Empresas como Avaki, DataSynapse, Entropia, Fujitsu, Hewlett–Packard, IBM, NEC, Oracle, Platform, Sun y United Devices han desarrollado estrategias Grid basándose en esta herramienta. En la comunidad científica proyectos de investigación como NEES (*Network for Earthquake Engineering and Simulation*) [NES], FusionGrid [Fus], ESG (*Earth System Grid*) [Gri] o EGEE (*Enabling Grids for E–sciencE*) [EGE] también han empleado *Globus Toolkit*.

4.3 Arquitectura Grid

Para gestionar este complejo sistema distribuido, que es el Grid, ha sido necesario desarrollar diferentes tecnologías que permitan coordinar el acceso a los recursos. La figura 4.2 muestra un esquema que representa la arquitectura Grid en cuatro niveles diferentes.

En el nivel de recursos se encontrarían todos aquellos recursos que pueden formar parte del Grid como supercomputadores, *clusters*, dispositivos de almacenamiento, redes o sensores. Generalmente algunos ya cuentan con gestores locales de recursos como PBS [PBS], Condor [Con] o Grid

Figura 4.2: Esquema de los diferentes niveles de la arquitectura Grid.

Engine [Eng]. En el nivel de conectividad se encontrarían los protocolos de comunicación y autenticación que permiten la conexión por red a los recursos de forma segura y sencilla. En el nivel de gestión de recursos se encuentran los protocolos de publicación, reconocimiento, monitorización y contabilidad de recursos consumidos. Finalmente, en el nivel de aplicaciones se encuentran todas aquellas aplicaciones que se comunican con los niveles anteriores para hacer uso del Grid. En este nivel podríamos situar por ejemplo los portales Grid como TeraGrid [Ter] o nanoHUB [nan].

4.4 Tecnologías Grid

Las tecnologías Grid suministran los mecanismos necesarios para asegurar el buen funcionamiento de la infraestructura. Éstas incluyen soluciones de seguridad que soportan la gestión de credenciales y políticas entre diferentes instituciones; protocolos y servicios para la gestión de los recursos, que además proveen información sobre la configuración y estado de éstos; y servicios de localización y transferencia de datos entre los sistemas de almacenamiento y las aplicaciones.

Las tecnologías Grid han surgido de la investigación y desarrollo que ha tenido lugar en esta última década y que todavía continúa hoy. Globus Toolkit ha sido la primera tecnología, considerada como estándar de facto,

que ha permitido la creación de entornos Grid gracias a la implementación de protocolos básicos como la autenticación o la transferencia de ficheros.

La creación del *Open Grid Services Architecture* (OGSA) [FKNT02] representó una evolución hacia una arquitectura Grid basada en las tecnologías de los servicios Web. Un ejemplo de este tipo de tecnología es el proyecto OGSA–DAI [FKNT02, OD] que permite el acceso y gestión mediante servicios Web de los recursos de datos (por ejemplo bases de datos relacionales o ficheros) en la infraestructura Grid.

Actualmente están surgiendo diferentes tecnologías, como SAGA [SAG] un estándar de código abierto que describe un interfaz de programación de alto nivel para la programación de aplicaciones Grid o Rapid [Rap] una tecnología que facilita la construcción de interfaces de usuario integradas y visualizadas en portales Web, que están orientadas a hacer más fácil y rentable la utilización por parte del usuario de las infraestructuras Grid.

4.5 Características de la infraestructura Grid es–NGI

Como ya hemos mencionado anteriormente en Europa existe el proyecto EGI que trata de mejorar y mantener una infraestructura Grid europea dedicada a la investigación en colaboración con las iniciativas Grid nacionales. En España la iniciativa Grid española (es–NGI) es la encargada de ofrecer una infraestructura Grid a la comunidad científica. La figura 4.3 representa los tres niveles fundamentales en los que se puede dividir esta infraestructura. En el primer nivel se encuentran las organizaciones virtuales (VOs) que agrupan a áreas comunes de investigación o aplicaciones. En el siguiente nivel se encuentran los servicios centrales de la es–NGI que proporcionan los servicios necesarios para que los usuarios puedan usar los recursos computacionales representados en el tercer nivel. Los principales servicios que permiten el funcionamiento de la infraestructura son los siguientes:

- Gestor de recursos. Se trata del servicio responsable del envío de

trabajos a los centros de recursos.

- Servicio de información. Está formado por una base de datos que suministra información del estado de los recursos al gestor de recursos.

- *Virtual Organisation Membership Service* (VOMS). Es responsable de la autenticación de los usuarios y contiene información acerca de que usuarios y organizaciones virtuales pertenecen a la es–NGI.

- Almacenamiento. Este servicio se encarga del almacenamiento de los ficheros y es accesible a todas las organizaciones virtuales.

- Catálogo de ficheros. Con este servicio se conoce cual es la localización de los ficheros usando para ello la base de datos que contiene esa información.

Además de los servicios anteriores la infraestructura es–NGI cuenta con servicios de monitorización y contabilidad con los cuales es posible supervisar el estado y uso de los recursos.

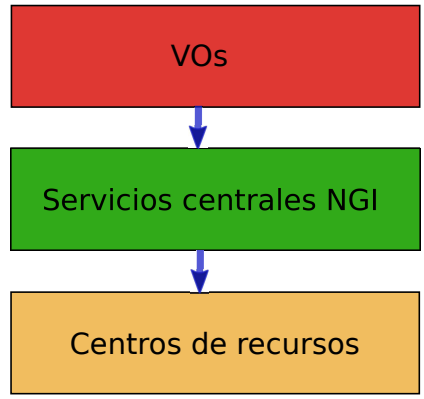


Figura 4.3: Descripción de la infraestructrua es–NGI.

En la tabla 4.1 se muestran los centros que pertenecen a la infraestructura es–NGI y los recursos computacionales que aportaban en el año 2010. En total la iniciativa es–NGI contaba con unos 5000 núcleos de ejecución y 500 Terabytes (TB) de espacio de almacenamiento dedicados a aplicaciones genéricas de grupos de investigación españoles. La previsión para el año 2011 es que exista un incremento de entorno al 50 % de los recursos disponibles de propósito general.

Centro	**Institución**	**Núcleos**	**Disco (TB)**	**Aplicación**
BIFI	UNIZAR	300	2	Propósito general
CESGA[1]	CESGA	477	144	Propósito general
CETA	CIEMAT	54	2	Propósito general
CIEMAT[1]	CIEMAT	1100	230	Propósito general: 250 núcleos LHC:850 núcleos
CIN2	CSIC	512	50	Propósito general
ESA–ESAC	ESA	24	< 1	Propósito general
GRYCAP	UPV	38	1	Propósito general
IAA	CSIC	512	2	Propósito general
IFCA[1]	CSIC	1800	354	Propósito general: 1200 núcleos/150 TB LHC: 600 núcleos/300 TB
IFIC	CSIC	1600	500	Propósito general: 1200 núcleos/200 TB LHC:320 núcleos/300 TB
IFISC	CSIC	512	50	Propósito general
PIC[1]	IFAE	1350	2000	VOs del LHC
RedIRIS	RED.es	6	< 1	Formación de usuarios
UNICAN[1]	UC	150	< 1	Propósito general
UNIOVI	UNIOVI	8	< 1	Propósito general

[1] Centros de recursos que soportan la VO eng.vo.ibergrid.eu

Tabla 4.1: Centros de recursos que forman parte de la es–NGI con sus respectivos recursos computacionales.

La organización eng.vo.ibergrid.eu es una VO perteneciente a la infraestructura es–NGI, que agrupa a los usuarios de la infraestructura cuya

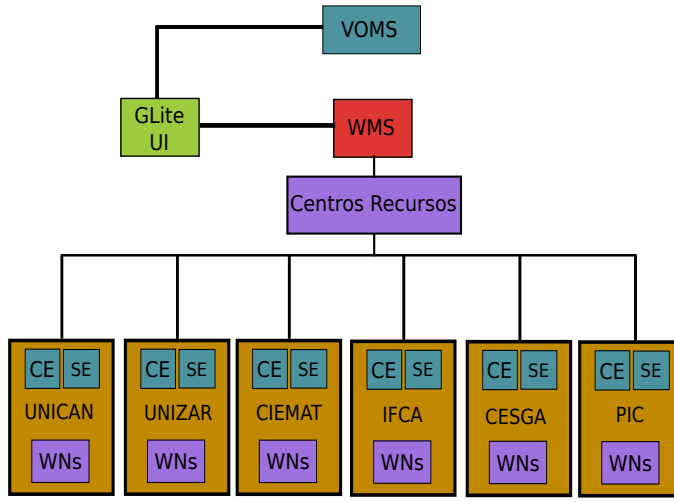


Figura 4.4: Esquema de funcionamiento de la eng.vo.ibergrid.eu.

investigación se orienta a la simulación en el área de ingeniería. De este modo el servicio VOMS puede autenticar al usuario cuando éste se conecta a la es–NGI y permitir su acceso a los recursos asignados a esta VO.

Todas las VOs tienen asociado un centro de recursos, en el caso de la VO eng.vo.ibergrid.eu el centro de recursos es el CESGA, que hace las funciones de centro de soporte. El CESGA tiene a disposición de los usuarios de esta VO una interfaz de usuario basada en gLite que permite el envío y gestión de trabajos. La figura 4.4 representa de forma esquemática su funcionamiento. Inicialmente el usuario se conecta a la interfaz de usuario (gLite–UI) del CESGA a través de una conexión *ssh*, para realizar el envío de trabajos. Como alternativa, puede instalar una máquina virtual en su propio equipo que haga las funciones de interfaz de usuario. Desde esta interfaz puede iniciar su autenticación como usuario del Grid perteneciente a la eng.vo.ibergrid.eu para lo que necesita un certificado digital X.509 de la autoridad certificadora pkIRISGrid [pkI] que le permite acreditarse en el servidor VOMS. La autenticación del usuario en el servidor VOMS

crea un certificado temporal denominado *proxy* que permite el acceso del usuario a los recursos del Grid. Si este certificado temporal caduca y el usuario no lo renueva, éste perderá el permiso de acceso a los recursos.

Una vez que el usuario está acreditado el siguiente paso es el envío de trabajos. Para enviar los trabajos es necesario crear un fichero en formato JDL [Job] en el que se especifican los ficheros y recursos necesarios para su ejecución. Después de que el trabajo ha sido enviado, es el gestor de recursos (WMS) el encargado de encontrar un centro de recursos apropiado para su ejecución. El WMS evalúa la carga de los centros disponibles y envía el trabajo al gestor de trabajos local (CE) con menor carga. Éste se encarga de su ejecución en un nodo de trabajo (WN) disponible, perteneciente a ese centro. Finalmente, los resultados de las ejecuciones se pueden almacenar en los elementos de almacenamiento (SE) en caso de que el tamaño de los ficheros de resultados lo requiera.

4.5.1 Herramientas de envío y monitorización de trabajos.

El envío y monitorización de trabajos se realiza como ya hemos mencionado en el apartado anterior con el *middleware* gLite–UI. Se trata de una interfaz de línea de comandos que nos permite realizar tareas de autenticación, envío y monitorización de trabajos.

Durante el desarrollo de las pruebas de evaluación del Grid, como recurso computacional para la simulación de nanodispositivos, hemos comprobado que para los casos en los que es necesario realizar cientos o miles de simulaciones, como ocurre en el caso de los estudios estadísticos de los dispositivos, sería conveniente mejorar el entorno de envío y monitorización de trabajos. Esto se debe a que es necesario enviar un gran número de simulaciones y el *middleware* gLite–UI asigna un identificador único a cada trabajo enviado, por lo tanto fuerza al usuario a evaluar el estado de cada una de las simulaciones durante el proceso de monitorización. A pesar de que el sistema permite enviar un conjunto de simulaciones como un único trabajo, el proceso de monitorización sigue siendo un problema cuando el número de simulaciones es elevado.

Para automatizar el proceso de envío y monitorización hemos creado una aplicación en el lenguaje de programación Python [Pyt], denominada SMNanoS, que permite el envío y monitorización de forma automática de los trabajos. De este modo una vez que éstos han sido enviados, es la aplicación la encargada de monitorizar su estado y en caso de que alguno falle, lo reenvía de nuevo. La figura 4.5 representa los tres niveles fundamentales de la aplicación.

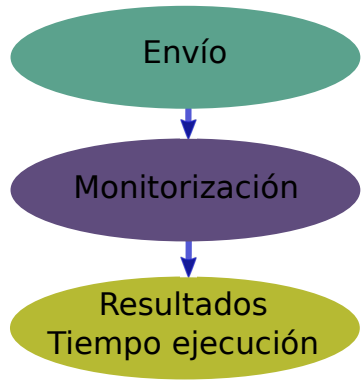


Figura 4.5: Esquema de los tres niveles fundamentales de la aplicación SMNanoS.

En el primer nivel se encuentra el envío de trabajos. Cuando un usuario invoca desde la línea de comandos a la aplicación, ésta le pregunta si quiere enviar una colección de trabajos o un único trabajo. Una vez que se ha elegido una opción, el usuario debe indicar el directorio en donde se encuentran los ficheros JDL (para enviar una colección) o la ruta del fichero JDL (para enviar un único trabajo). A cada trabajo enviado el *middleware* de gLite le asocia un identificador. En este primer nivel, la aplicación además de enviar el trabajo crea un diccionario que asocia el identificador con su correspondiente fichero JDL.

En el nivel de monitorización se evalúa el estado del trabajo en periodos fijos de tiempo que pueden cambiar dependiendo del tiempo estimado

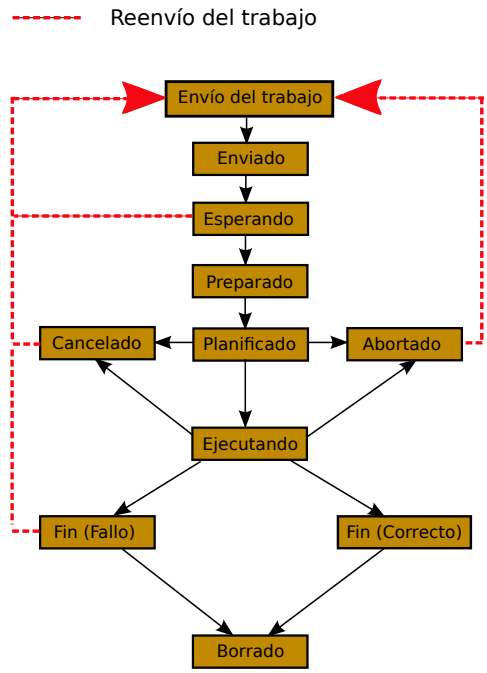


Figura 4.6: Esquema de los posibles estados por los que puede pasar un trabajo enviado con gLite–UI a la infraestructura es–NGI, indicando los estados en los que la aplicación SMNanoS puede tomar la decisión de reenviar el trabajo.

de ejecución. Durante la evaluación de los posibles estados, la aplicación puede elegir diferentes acciones en función del estado del trabajo. La figura 4.6 representa los posibles estados de un trabajo enviado y los casos en los que la aplicación SMNanoS puede tomar la decisión de reenviar el trabajo. En esta situación se encuentran cuatro estados, el primero de ellos es cuando el *middleware* devuelve el estado "esperando" que indica que en ese momento los recursos computacionales solicitados no están disponibles. En este caso cuando el tiempo de espera es mayor que la mitad del tiempo de ejecución estimado, el trabajo se reenvía automáticamen-

te. El número máximo de reenvíos está limitado a cinco en cuyo caso se notifica al usuario que debe revisar el trabajo. Los estados "cancelado" o "abortado" son un motivo suficiente para que el reenvío se produzca automáticamente. Finalmente, otra posible situación en la que el trabajo puede ser reenviado es cuando el estado es "Fin (Fallo)". En este caso solo se reenvía el trabajo si la aplicación no tiene acceso a los ficheros de salida almacenados en el SE.

Cuando se reenvía un trabajo, por cualquiera de los motivos explicados, el *middleware* le asigna un nuevo identificador. Por lo tanto, la aplicación debe reemplazar el identificador del trabajo cancelado para que se pueda realizar la monitorización del nuevo envío.

Finalmente en el último nivel de funcionalidad la aplicación SMNanoS realiza la gestión de los datos de salida. Descarga los ficheros de las simulaciones realizadas que se encuentran almacenados en el SE y ofrece la posibilidad de representar en un gráfico los tiempos de ejecución de las simulaciones.

4.6 Resultados de la simulación en la es–NGI

Para evaluar las posibles ventajas de las infraestructuras Grid en la simulación de nanodispositivos hemos realizado diferentes pruebas con el simulador 2D MV–ECBE–EMC. Los dos principales objetivos de este estudio fueron reducir los tiempos de ejecución de las simulaciones aprovechando el enorme número de recursos de la es–NGI y estudiar el impacto de la heterogeneidad de este tipo de infraestructuras en los tiempos de ejecución de los trabajos enviados.

4.6.1 Reducción del tiempo de ejecución para estudios estacionarios de los dispositivos electrónicos

Una posibilidad que nos ofrece el elevado número de recursos computacionales de la es–NGI es la reducción del tiempo de ejecución de aquellas simulaciones cuyo estudio se centra en el análisis de los estados estacio-

narios. La principal característica de los estados estacionarios es que son independientes del estado inicial cuando el tiempo de convergencia es suficientemente largo. Este tipo de simulaciones son muy habituales y su finalidad es estudiar el comportamiento del dispositivo para diferentes puntos de polarización. En este caso, se considera que cada punto de polarización es un estado estacionario de una duración determinada, durante el cual se alcanzan los valores de convergencia de las magnitudes de la simulación Monte Carlo que se quieren obtener. Por lo tanto, es posible enviar cada punto de polarización de la simulación a un nodo diferente de la es–NGI, reduciendo de esta forma el tiempo de ejecución de la simulación estacionaria al tiempo de un único punto de polarización.

Para comprobar las ventajas de este método hemos realizado una simu-

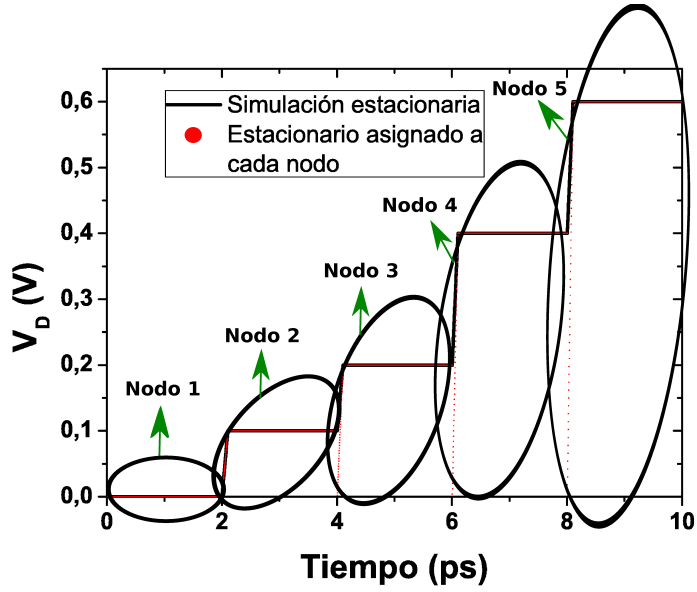


Figura 4.7: División de una simulación de 10 ps con cinco estacionarios de 2 ps para su envío a los nodos de la es–NGI.

lación estacionaria de 10 ps de duración con cinco puntos de polarización diferentes para la tensión de drenador. Cada punto de polarización es un estacionario de 2 ps de duración. La figura 4.7 muestra como se divide la simulación que hemos utilizado como ejemplo en cinco casos, asignando cada uno de ellos a un nodo computacional. Cada uno de los estacionarios que se envían a los nodos parten de la solución en equilibrio del dispositivo $V_D = 0$ V e inmediatamente después de la primera iteración de la simulación Monte Carlo, se cambia V_D al valor del estacionario correspondiente.

El dispositivo empleado en esta prueba ha sido un transistor DGSOI MOSFET de 10 nm de longitud de puerta, 6 nm de espesor de silicio y 10 Å de espesor efectivo de dióxido de silicio. El valor del dopado de las regiones de fuente y drenador es $N_D = 9 \text{ x } 10^{19}$ cm^{-3} y en el canal es $N_A = 10^{15}$ cm^{-3}. La dos puertas del dispositivo están polarizadas a 1 V mientras que la tensión de drenador toma los valores indicados en la figura 4.7.

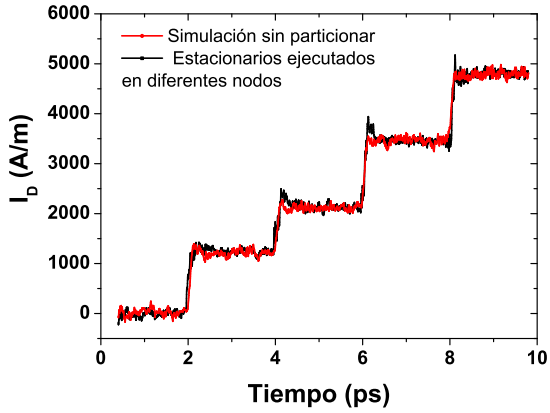
Estacionario	Tiempo (s)	Nodo	Procesador	RAM (MB)
1	6159	formiga6	Core2 Duo 2,66	512
2	10293	formiga1	Xeon 1,60	512
3	6447	formiga8	Core2 Duo 2,66	512
4	6560	formiga7	Core2 Duo 2,66	512
5	6189	formiga12	Core2 Duo 2,66	1024
Sin particionar	50655	formiga25	Xeon 1,60	512

Tabla 4.2: Tiempo de ejecución de cada estacionario en los que se ha dividido la simulación y tiempo de la simulación completa sin particionar ejecutada en uno de los nodos. Además se incluyen las características del nodo en donde se ha ejecutado: nombre del nodo, modelo del procesador y frecuencia, y tamaño de la memoria principal.

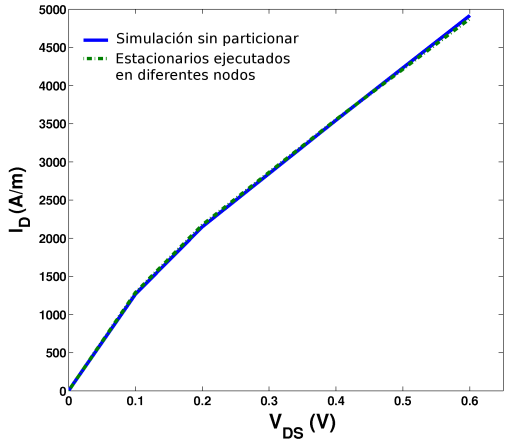
La tabla 4.2 muestra el tiempo de ejecución de cada uno de los cinco estacionarios en los que se ha dividido la simulación y el tiempo de una simulación completa de 10 ps, junto con las características de cada uno de los nodos en los que se han ejecutado las simulaciones. En los resultados de esta tabla se observa que el tiempo de ejecución de una simulación, en la

que se asigna cada estacionario a un nodo del Grid, es del orden de 5 veces inferior al de una simulación de 10 ps realizada en uno de estos nodos de cálculo. De este modo el tiempo de ejecución está determinado por el nodo más lento del conjunto de nodos que han ejecutado los estacionarios en los que se ha particionado la simulación. Los resultados obtenidos son al mismo tiempo una evidencia de la heterogeneidad de este tipo de recursos, ya que en este caso hemos obtenido variaciones en el tiempo de ejecución de hasta casi un 60 % en función de las características del nodo.

Finalmente en la figura 4.8 comparamos los resultados obtenidos cuando se reparten los estacionarios entre los nodos y cuando se realiza la simulación sin particionar en un único nodo. En la figura 4.8a representamos la evolución de la corriente de drenador con el tiempo y en la figura 4.8b se muestra la curva I_D frente a V_{DS}. En ambos casos los resultados obtenidos son prácticamente idénticos. En la figura 4.8a se puede observar un poco más de ruido al principio de cada estacionario cuando son repartidos entre los nodos. Esto es debido a que con este método, el cambio entre la polarización de la solución inicial $V_D = 0$ V y el punto de polarización correspondiente al estacionario es más brusco que si alcanzamos el punto de polarización deseado desde el estacionario anterior.



(a) Evolución de la corriente de drenador con el tiempo.



(b) Corriente de drenador frente a V_{DS}.

Figura 4.8: Comparación de la evolución de la corriente de drenador con el tiempo (4.8a) y de I_D frente a V_{DS} (4.8b) cuando se reparten los estacionarios entre los nodos y cuando se realiza la simulación sin particionar en un único nodo.

4.6.2 Impacto de la heterogeneidad de los recursos en el tiempo de ejecución del simulador 2D MV–ECBE–EMC.

En los tiempos de ejecución del apartado anterior se observaban variaciones en el tiempo de ejecución de hasta un 60 % dependiendo del nodo de cálculo que ejecutara el estado estacionario. Para comprobar la influencia de la heterogeneidad de los recursos disponibles hemos creado una colección de cinco trabajos, formada por los cinco estacionarios de la simulación utilizada en la sección anterior, y hemos medido el tiempo de ejecución en cada uno de los centros de recursos que nos dieron soporte. (Los centros de soporte han sido destacados con un [1] en la tabla 4.1).

Los resultados obtenidos para cada centro se muestran en las subtablas de la tabla 4.3, en donde se indica el tiempo de ejecución de cada estacionario así como, el tipo de procesador y el nombre del nodo que lo ha ejecutado. La figura 4.9 representa estos resultados en un diagrama de barras, en el que se comparan los centros de recursos para cada uno de los estacionarios de la colección. En esta figura se observa, tanto las diferencias que existen entre los tiempos de cálculo de los diferentes centros como las diferencias dentro de un mismo centro (por ejemplo el estacionario 1 comparado con el resto de estacionarios ejecutados en el PIC) debido a la existencia de nodos heterogéneos dentro de éste.

En la figura 4.10 se compara el tiempo de ejecución de la colección enviada a cada uno de los centros de recursos usando un gráfico de barras. Además, se muestra con una línea la variación porcentual del tiempo de ejecución con respecto al centro más rápido. Como criterio para estimar el tiempo de ejecución de la colección hemos considerado que éste es igual al tiempo del estacionario más lento. Los datos representados muestran que el PIC es el centro de recursos más rápido y con respecto a este centro existen variaciones en el tiempo de cálculo de hasta un 55 %. A pesar de que estas variaciones son importantes e inevitables debido a las características de la infraestructura, la es–NGI dispone de un elevado número de recursos que permite la ejecución simultánea de muchos trabajos, lo que supone una importante ventaja para el estudio estadístico de los dispositi-

vos electrónicos. Este tipo de estudios requiere ejecutar cientos o miles de simulaciones que podrían saturar los sistemas de colas de un único centro de cálculo.

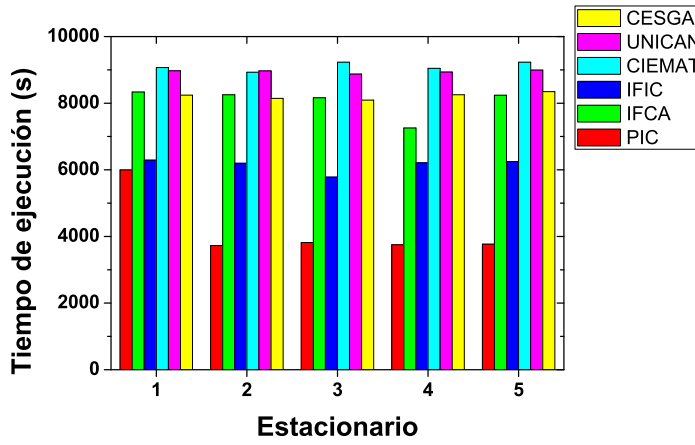


Figura 4.9: Tiempo de ejecución de cada estacionario que forma parte de la colección de trabajos enviada a cada centro de recursos.

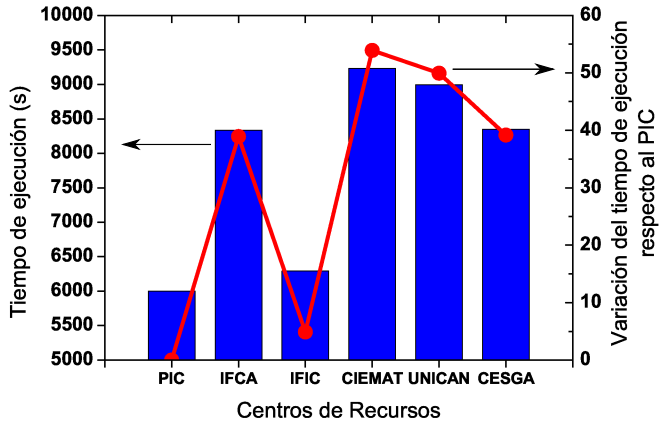


Figura 4.10: Representación del tiempo de ejecución de la colección enviada a cada uno de los centros de recursos. Hemos considerado como tiempo de ejecución de la colección el tiempo del estacionario más lento de los cinco que la componen. Además se representa el porcentaje de variación en el tiempo de ejecución entre los centros de recursos comparado con el centro más rápido.

Estacionario	Tiempo (s)	Nodo
1	5998	td160.pic.es
2	3727	td485.pic.es
3	3817	td484.pic.es
4	3750	td487.pic.es
5	3767	td472.pic.es

(a) Tiempo de ejecución de cada estacionario en el Puerto de Información Científica (PIC). La arquitectura de estos nodos es x86_64 y el modelo de procesador es Xeon L5420 2,50 GHz para el estacionario 1 y Xeon L5530 2,40 GHz para el resto de estacionarios.

Estacionario	Tiempo (s)	Nodo
1	8334	cms01wn
2	8252	cms01wn
3	8162	cms01wn
4	7257	cms02wn
5	8240	cms01wn

(b) Tiempo de ejecución de cada estacionario en el Instituto de Física de Cantabria (IFCA). La arquitectura de estos nodos es x86_64 y el modelo del procesador Xeon E5345 2,33 GHz.

Estacionario	Tiempo (s)	Nodo
1	6290	wn304.ific.uv.es
2	6198	wn304.ific.uv.es
3	5781	wn305.ific.uv.es
4	6210	wn304.ific.uv.es
5	6248	wn304.ific.uv.es

(c) Tiempo de ejecución de cada estacionario en el Instituto de Física Corpuscular (IFIC). La arquitectura de estos nodos es x86_64 y el modelo del procesador Xeon E5420 2,50 GHz.

Tabla 4.3

Estacionario	Tiempo (s)	Nodo
1	9071	ciewn037.ciemat.es
2	8930	ciewn087.ciemat.es
3	9232	ciewn036.ciemat.es
4	9045	ciewn035.ciemat.es
5	9231	ciewn036.ciemat.es

(d) Tiempo de ejecución de cada estacionario en el Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas (CIEMAT). La arquitectura de estos nodos es i686 y el modelo del procesador AMD Opteron270 2,00 GHz.

Estacionario	Tiempo (s)	Nodo
1	8974	wn007.macc.unican.es
2	8969	wn004.macc.unican.es
3	8875	wn006.macc.unican.es
4	8936	wn004.macc.unican.es
5	8995	wn006.macc.unican.es

(e) Tiempo de ejecución de cada estacionario en la Universidad de Cantabria (UNICAN). La arquitectura de estos nodos es x86_64 y el modelo del procesador PentiumD 3,00 GHz.

Estacionario	Tiempo (s)	Nodo
1	8243	compute–2–11
2	8145	compute–1–1
3	8091	compute–2–11
4	8252	compute–0–11
5	8348	compute–1–10

(f) Tiempo de ejecución de cada estacionario en el Centro de Supercomputación de Galicia (CESGA). La arquitectura de estos nodos es i686 y el modelo del procesador Pentium4 3,20 GHz.

Tabla 4.3: Resultados de los tiempos de ejecución de los cinco trabajos que forman parte de la colección enviada a cada centro de recursos. PIC (4.3a), IFCA (4.3b), IFIC (4.3c), CIEMAT (4.3d), UNICAN (4.3e), CESGA (4.3f).

CAPÍTULO 5

Estudio de fuentes de variabilidad en transistores MOSFET basados en SOI

En los últimos años, el estudio del efecto de la variabilidad estadística en las características de los transistores se ha convertido en un tema de gran importancia [Ase07, CCC+08]. La principal razón reside en que dichas variaciones tienen un mayor impacto en el funcionamiento de los circuitos integrados conforme el escalado se acerca a las dimensiones nanométricas. Las fuentes de variabilidad son inherentes a los procesos de fabricación que se emplean actualmente en la tecnología CMOS y se pueden dividir principalmente en dos categorías: las fuentes aleatorias como las fluctuaciones debidas a la naturaleza discreta de los dopantes o la rugosidad de la interfaz (*line-edge roughness*, LER), y las fuentes sistemáticas como los efectos de tensión mecánica o estrés debidos a los procesos de fabricación en la creación de silicio tenso para mejorar el rendimiento de los transistores o las fluctuaciones de la relación largo/ancho de puerta.

En este capítulo estudiaremos dos casos de variabilidad y su efecto en las características de los transistores SOI MOSFET, empleando para ello los simuladores Monte Carlo paralelizados que hemos descrito en los

capítulos anteriores. En el primer caso analizaremos el impacto de una fuente de variabilidad sistemática, el desalineamiento entre la puerta superior e inferior de un transistor DGSOI MOSFET con el simulador MV–ECBE–EMC. En el segundo caso analizaremos el impacto de una fuente de variabilidad aleatoria con el simulador MSB–EMC, se trata del estudio del efecto de una LER creada en la interfaz HfO_2/SiO_2 de un transistor SOI de una única puerta (SGSOI).

5.1 Simulación del desalineamiento de puerta en transistores DGSOI MOSFET con el simulador MV–ECBE–EMC

En la actualidad el estado del arte de la investigación de transistores MOSFET se centra en el estudio de dispositivos con tamaños de puerta del orden de decenas de nanómetros. El ITRS [ITR10] predice que a partir del año 2020 se fabricarán dispositivos con longitudes de puerta inferiores a 10 nm y, como ya mencionamos anteriormente, la tecnología Bulk MOSFET presenta importantes problemas para el control de los efectos de canal corto que dificulta de forma notable su uso para los futuros nodos tecnológicos. En la actualidad se investigan nuevas arquitecturas como los transistores multipuerta o los transistores FinFET, considerados una alternativa importante para convertirse en los dispositivos estándar de la próxima generación de circuitos integrados. Desde el punto de vista de la simulación es necesario introducir modelos numéricos que tengan en cuenta los efectos cuánticos debido a la importante influencia de éstos sobre las características eléctricas de los dispositivos [AI89, Anc90, Dat00, TR01, WR03, GF04, ARA08, QNSM$^+$09].

Dentro de los dispositivos multipuerta, los transistores MOSFET de doble puerta DG MOSFET son una de las mejores alternativas para sustituir a los transistores MOSFET de tecnología Bulk y continuar así con el escalado de los mismos hasta alcanzar tamaños del orden de unos pocos

nanómetros. La principal característica de estos dispositivos es que ofrecen un mejor control electrostático reduciendo los efectos de canal corto y mejorando la corriente de drenador en saturación. Además, estos dispositivos aprovechan los beneficios de la inversión en volumen [CC03, ITR10]. Sin embargo, uno de los problemas de los DG MOSFET surge del desalineamiento de la puerta que puede tomar especial relevancia en el caso de dispositivos de doble puerta planares o con puertas independientes.

El desalineamiento de las puertas en transistores DG MOSFET ha sido estudiada por varios autores [WFTS94, AZPC01, SMC03, YC05, WLV$^+$05, KA08] ya que su impacto se hace mayor con el escalado de los dispositivos. Principalmente estos estudios muestran resultados experimentales [YC05, WLV$^+$05] y resultados obtenidos con simuladores que emplean el método de arrastre–difusión [WFTS94, AZPC01, SMC03]. Ambos afirman que para valores de longitud de puerta en el rango de los 50–100 nm es posible asumir un desalineamiento de entre el 20–25 % de la longitud de puerta, sin que se vea afectado su rendimiento seriamente. Al mismo tiempo se observa un mejor rendimiento del dispositivo con respecto al caso alineado cuando la puerta es desalineada hacia el contacto de fuente.

Cuando se produce un escalado agresivo de estos dispositivos es necesario incluir efectos cuánticos en las simulaciones, que nos permitan describir de un modo adecuado el impacto del desalineamiento de la puerta en dispositivos de canal ultra corto y ultra delgados. Por lo tanto, debido a que en este estudio simulamos un transistor DGSOI MOSFET de 10 nm de longitud de puerta, entre 5 y 20 veces más pequeño que los publicados en trabajos previos , hemos empleado el simulador MV–ECBE–EMC (descrito en la sección 2.3) ya que incluye efectos cuánticos por medio del modelo MV–ECBE [SGGR06]. Consideramos que este es el primer estudio en el que se simula el impacto del desalineamiento de puerta con un simulador Monte Carlo, de modo que podremos comprobar si los resultados obtenidos con otros métodos de simulación para dispositivos más grandes, se siguen observando en transistores mucho más pequeños y simulados con un método diferente.

5.1.1 Descripción del dispositivo simulado. Configuraciones del desalineamiento de puerta

Para realizar este estudio hemos seleccionado un transistor DGSOI MOSFET con un tamaño de puerta de 10 nm de longitud. Las regiones de fuente y drenador tienen una longitud de 15 nm y un espesor de 6 nm. Ambas regiones cuentan con un dopado abrupto de impurezas donadoras $N_D = 5x10^{19}$ cm^{-3}. La región de canal, dopada con impurezas aceptoras con un concentración de dopantes $N_A = 10^{15}$ cm^{-3}, es de la misma longitud que la puerta y tiene el mismo espesor que las regiones de fuente y drenador. Finalmente, la región de óxido de silicio tiene un espesor igual a 1 nm y la función de trabajo considerada para el contacto metálico de la puerta es igual a 4,6 eV. La figura 5.1(a) muestra un representación esquemática de este transistor cuando las puertas se encuentran alineadas.

Para cuantificar el desalineamiento de la puerta hemos definido una magnitud que indica el porcentaje de puerta que se encuentra desalineado con respecto a su longitud total (mis). El valor de mis se obtiene de la expresión, $mis = 100 \times L_m/L_g$ (%), en donde L_m representa el desplazamiento desde la posición en que ambas puertas se encuentran alineadas (auto–alineamiento) y L_g es el la longitud total de la puerta. En la figura 5.1(c) se representa a modo de ejemplo el desplazamiento L_m. Como criterio para indicar en que sentido se desplaza la puerta se ha establecido que los valores positivos de mis indican un desplazamiento de la puerta hacia el drenador, mientras que el desplazamiento hacia la fuente se representa con valores negativos.

En la figura 5.1 se representan las tres configuraciones posibles de desalineamiento que han sido objeto de estudio. La figura 5.1(b) representa el caso en el que se desalinea la puerta superior (TGM) con valores de mis entre $-50\,\%$ y $50\,\%$. En la siguiente configuración, figura 5.1(c), ambas puertas son desalineadas en direcciones opuestas (BGOD). Para este caso se desalinea la puerta superior hacia la fuente y la puerta inferior hacia el drenador, cada una de ellas con valores de mis desde $0\,\%$ hasta $\pm 50\,\%$. El símbolo $\pm$ indica que la puerta inferior (superior) se desplaza hacia el drenador (fuente). Finalmente, se representa la última configuración en la

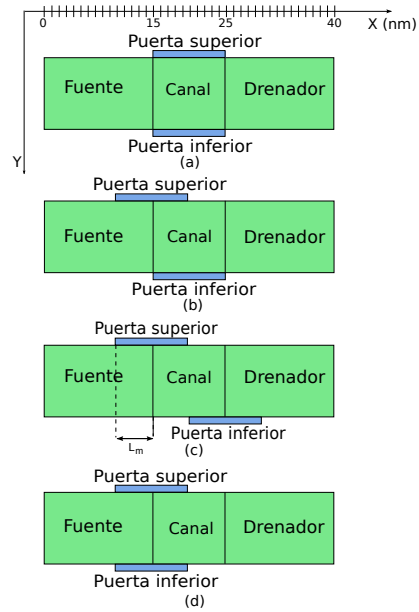


Figura 5.1: Diferentes configuraciones del dispositivo estudiado de 10 nm de longitud de canal. (a) Estructura alineada, (b) configuración del desalineamiento de la puerta superior (*Top Gate Misalignment*, TGM), (c) desalineamiento de ambas puertas en direcciones opuestas (*Both Gates in Opposite Directions*, BGOD) y (d) desalineamiento de ambas puertas en la misma dirección (*Both Gates in the Same Direction*, BGSD).

figura 5.1(d), en donde ambas puertas se desalinean en la misma dirección (BGSD) con valores de *mis* desde $-50\,\%$ hasta $50\,\%$.

Para llevar a cabo este estudio hemos realizado dos tipos diferentes de simulaciones para cada una de las configuraciones (TGM, BGOD y BGSD). En la primera de ellas, hemos fijado el voltaje de ambas puertas a 1 V mientras que el voltaje de fuente y drenador lo hemos variado desde 0 hasta 1 V. En la segunda simulación, hemos fijado el voltaje de drenador a 100 mV mientras que el voltaje de ambas puertas se ha ido variando desde 0 a 1 V en pasos de 50 mV.

5.1.2 Resultados del desalineamiento de la puerta superior

En el caso de la configuración TGM, figura. 5.1(b), la puerta superior ha sido desalineada desde $-50\,\%$ hasta $50\,\%$. La figura 5.2 muestra la corriente de drenador frente al voltaje de drenador para cada una de las posiciones de la puerta superior con $V_G = 1$ V. Los resultados obtenidos muestran que cuando el desalineamiento es inferior al $20\,\%$, con independencia de que este se produzca hacia la fuente o el drenador, los valores de I_D son similares a los que se obtienen en el caso alineado. Sin embargo, para desplazamientos de la puerta hacia el drenador mayores del $20\,\%$, los valores de I_D disminuyen notablemente con respecto al caso alineado, con caídas superiores al $5\,\%$ del valor del caso alineado, para $V_D = 1$ V. Por otro lado es interesante destacar que cuando el desalineamiento es del $10\,\%$ en la dirección de la fuente se encuentra un máximo en la corriente de drenador, observándose un mejor rendimiento del dispositivo para esta configuración.

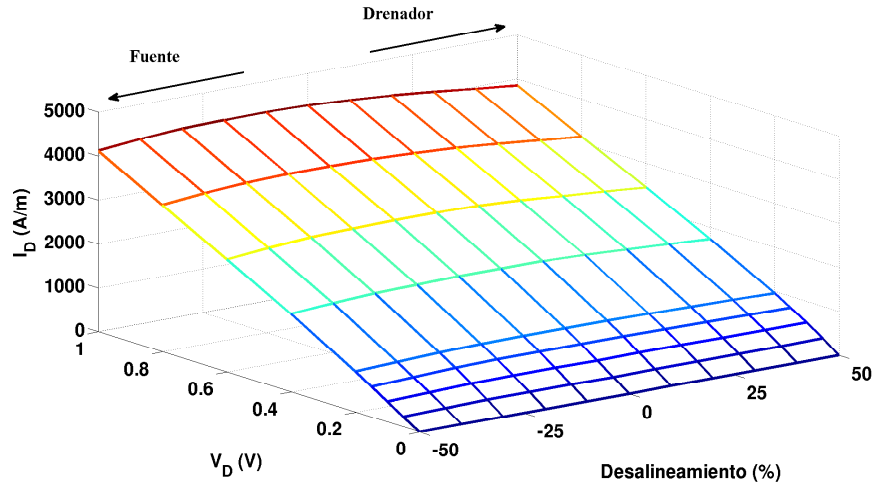


Figura 5.2: I_D frente V_D para los diferentes valores de desalineamiento de la puerta superior, con un valor del voltaje de puerta igual a 1 V.

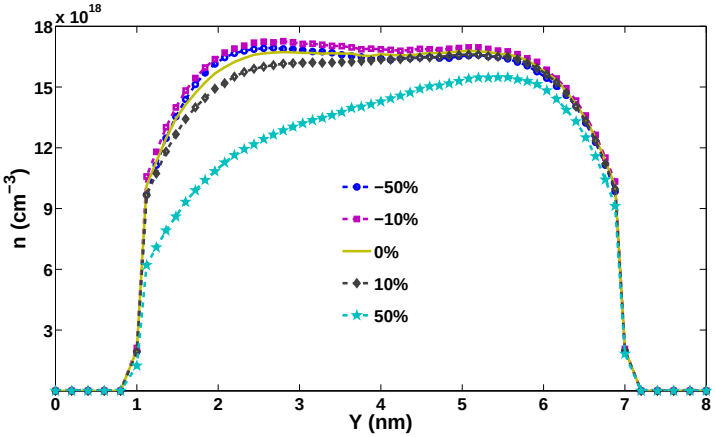


Figura 5.3: Perfil de la concentración de electrones en la posición X = 17 nm para cada una de las posiciones de puerta de la configuración TGM cuando $V_D = V_G = 1$ V.

Estos resultados parecen coincidir con los obtenidos en otros trabajos [WFTS94, AZPC01, SMC03, YC05, KA08] en los que también se muestra un incremento de la corriente de drenador cuando la puerta superior se desplaza hacia la fuente. Este incremento puede estar causado por la reducción de la resistencia serie de la fuente debido a la acumulación de carga producida en esta zona [AZPC01].

La figura 5.3 representa el perfil de la concentración de electrones en la fuente virtual (X = 17 nm) a lo largo de la dirección de confinamiento para diferentes posiciones de puerta de la configuración TGM. En general, se observa un incremento de la concentración de electrones en la región del canal próxima a la puerta desalineada, cuando ésta se desplaza hacia el contacto de fuente ($-50\,\%$ y $-10\,\%$). Sin embargo, cuando la puerta superior se desplaza hacia el contacto de drenador se observa una disminución de la concentración de electrones en la región próxima a la puerta inferior, lo que explica la reducción de la corriente comentada previamente en la figura 5.2. Este comportamiento se debe a la pérdida del control

electrostático del canal cuando la puerta se desplaza hacia el drenador así como, al incremento de la altura de la barrera de potencial como se aprecia en la figura 5.4. Esta figura representa la banda de conducción de

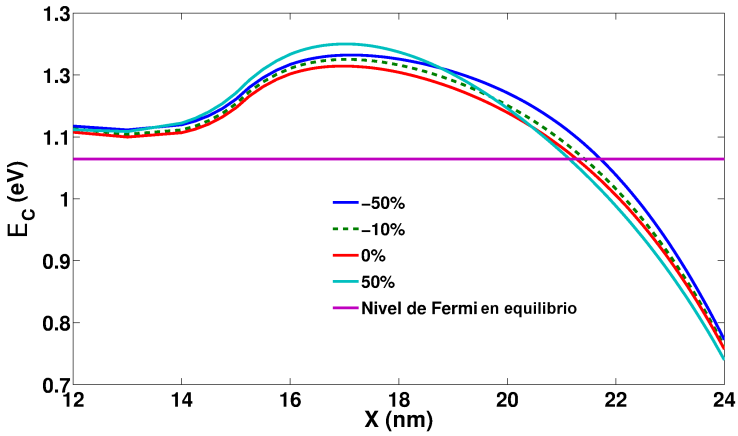


Figura 5.4: Banda de conducción de fuente a drenador a 2,2 nm de la posición de la puerta superior en la dirección Y para la configuración TGM cuando $V_D = V_G = 1$ V.

la región de fuente a drenador para la posición Y = 2,2 nm, cuando el desalineamiento de la puerta es máxima hacia la fuente (−50 %) y hacia el drenador (50 %). Además, también se muestran las bandas de conducción del caso alineado y del caso con desalineamiento −10 %. Para este último, en el que el valor de la corriente de drenador es máximo, se puede ver una disminución del máximo de la banda en el canal, lo que implica una disminución de la barrera de potencial, en contraposición a lo que ocurre en los casos límite −50 % y 50 %.

La figura 5.5 muestra los resultados de la simulación cuando variamos el voltaje de puerta V_G entre 0 y 1 V en pasos de 50 mV, cuando el valor de la tensión de drenador permanece constante a 100 mV. En concreto, la figura 5.5(a) muestra las curvas I_D–V_G para los diferentes casos de desalineamiento de la configuración TGM, en donde se observa un aumento

de las desviación de la corriente de drenador con respecto al caso alineado cuando aumenta el desalineamiento y la tensión de puerta. Para analizar con más detalle cómo y cuánto se desvía la corriente de drenador con respecto al caso alineado en función del desalineamiento y para valores de V_G altos hemos dibujado la figura 5.5(b). Esta figura muestra que la desviación de la corriente de drenador con respecto al caso alineado aumenta con el desalineamiento de puerta. Por lo tanto, si establecemos un valor de referencia o tolerable de la desviación de la corriente de drenador con respecto al caso alineado del 7 %, podemos observar que existe una ventana de tolerancia que nos indica los valores de desalineamiento para los cuales la desviación de la corriente de dranador es inferior a este valor de referencia, observándose que esta referencia se supera cuando los valores de desalineamiento de la puerta son mayores que el 30 % y menores que el -20 %.

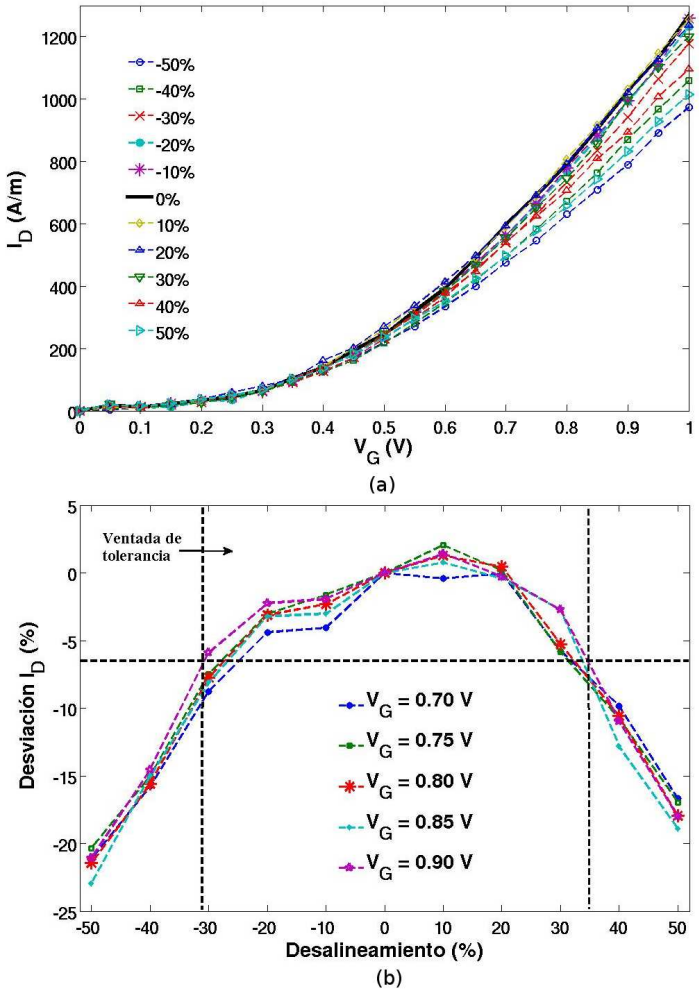


Figura 5.5: (a) Curvas I_D–V_G para la configuración TGM. (b) Comportamiento de la desviación relativa de la corriente de drenador con respecto al valor obtenido cuando la puerta se encuentra alineada para diferentes valores del voltaje aplicado en las puertas. En todos los casos V_D se ha mantenido a un valor constante de 100 mV.

5.1.3 Resultados del desalineamiento de ambas puertas en direcciones opuestas

En este caso hemos realizado un estudio similar al anterior pero moviendo ambas puertas en direcciones opuestas. La puerta superior ha sido desplazada hacia la fuente y la inferior hacia el drenador como se muestra en la figura 5.1(c).

En la figura 5.6 se muestran las curvas de la corriente de drenador frente al voltaje aplicado en drenador para los diferentes valores de desalineamiento de la puerta, cuando $V_G = 1$ V. Como se puede observar en esta figura la corriente de drenador siempre disminuye cuando ambas puertas se encuentran desalineadas. Un desalineamiento igual a $\pm 20\,\%$ causa una desviación de la corriente de aproximadamente igual al $2\,\%$ del valor obtenido cuando las puertas están alineadas. Cuando el desalineamiento es mayor que $\pm 30\,\%$ el valor de la desviación de la corriente es mayor del $5\,\%$.

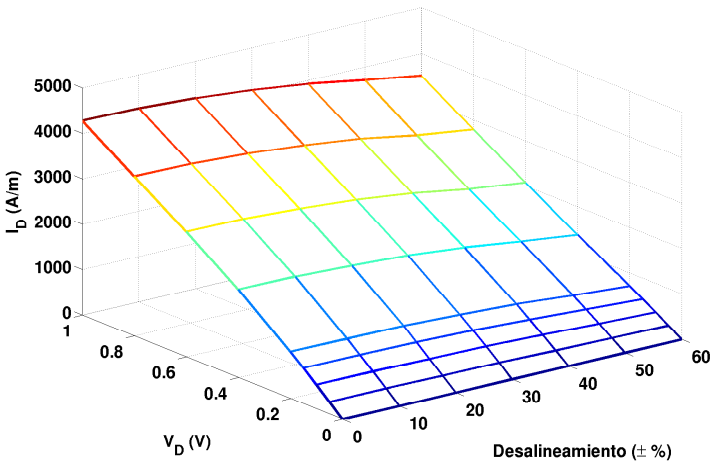


Figura 5.6: I_D frente V_D para los diferentes valores de desalineamiento simulados de la configuración BGOD, cuando el voltaje de puerta ha sido fijado a un valor constante igual a 1 V.

Para observar el impacto del desalineamiento de puerta sobre la banda de conducción hemos representado, en la figura 5.7, el perfil de la banda para cuatro valores diferentes de desalineamiento cuando $V_D = V_G = 1$ V. En esta figura se aprecia que los valores más pequeños de la barrera de energía se obtienen para el caso alineado, lo que justifica la disminución de la corriente, observada en la figura 5.6, cuando la puerta está desalineada.

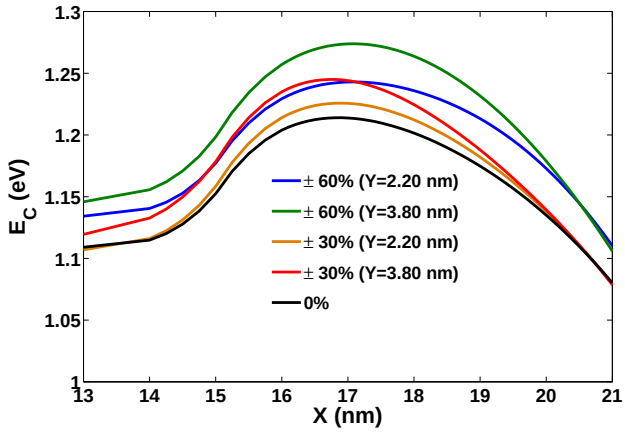


Figura 5.7: Bandas de conducción de fuente a drenador en las posiciones Y = 2,2 nm y 3,8 nm para la configuración BGOD cuando $V_D = V_G = 1$ V.

En la figura 5.8 podemos ver el efecto que causa la puerta desalineada sobre la concentración de electrones, en la posición en la que se encuentra el máximo de la barrera de potencial (X = 17 nm) representada en la figura 5.7 cuando $V_D = V_G = 1$ V. La concentración de electrones en este caso tiene un comportamiento similar al observado en el caso TGM. Cuando la puerta superior se desplaza hacia el contacto de fuente se produce un incremento en la concentración de electrones en la parte superior del canal. Sin embargo, la puerta inferior se desplaza en la misma proporción hacia el drenador, causando una disminución de la concentración de electrones en la parte inferior del canal. Por lo tanto, el control electrostático que

ejercen la puertas sobre el canal es menor debido al desalineamiento de la puerta inferior.

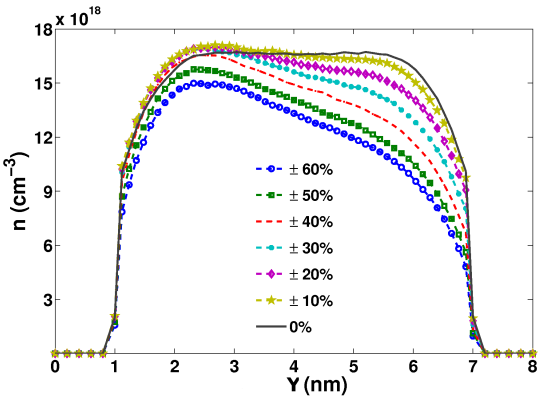


Figura 5.8: Concentración de electrones en el canal en la posición X = 17 nm para cada una de las configuraciones del caso BGOD cuando $V_D = V_G = 1$ V.

Los efectos sobre la corriente de drenador, de las diferentes configuraciones de las puerta en el caso BGOD, en función del valor del voltaje de puerta y para un valor constante de $V_D = 100$ mV se muestran en la figura 5.9 (a). En esta figura se observa en la región sub–umbral, un incremento de la corriente de drenador que degrada la relación I_{on}/I_{off}, y que se debe principalmente a la dispersión de la carga a lo largo del silicio del canal y a la disminución del mecanismo de dispersión por rugosidad superficial en la parte inferior de la puerta cerca de la fuente virtual. Además, se aprecia como para valores altos de carga en inversión, el impacto del desalineamiento de la puerta es más importante en este caso que para el TGM. Como consecuencia, cuando las puertas están desalineadas más de un 20 % se observa una degradación crítica de las características de salida, como se muestra en la figura 5.9 (b), en la que se representa la desviación relativa de la corriente de drenador, para diferentes desalineaciones de la configuración BGOD, con respecto al caso alineado.

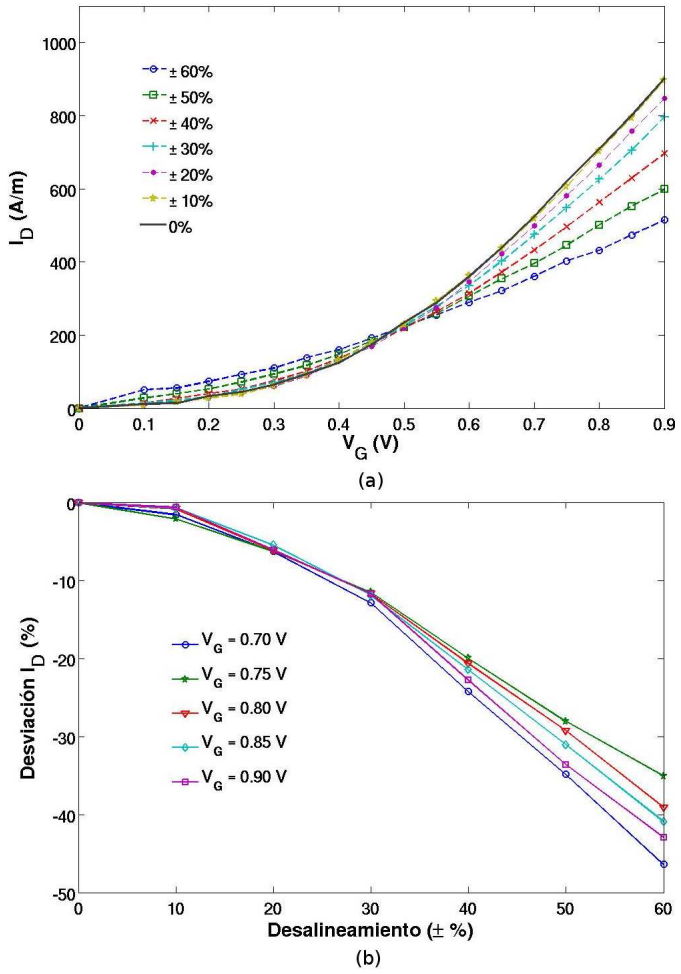


Figura 5.9: (a) Curvas I_D–V_G para cada una de las configuraciones del caso BGOD. (b) Comportamiento de la desviación relativa de la corriente de drenador, para diferentes desalineaciones de la configuración BGOD, con respecto al caso alineado. Este estudio se ha hecho para diferentes potenciales de puerta, manteniendo un valor de V_D constante igual a 100 mV.

5.1.4 Resultados del desalineamiento de ambas puertas en la misma dirección

Como ya se ha comentado en las secciones anteriores, se aprecia una disminución del rendimiento del dispositivo cuando una de las puertas se desplaza hacia el drenador. En esta sección vamos a analizar el comportamiento del transistor desalineado cuando ambas puertas se desplazan en la misma dirección.

La figura 5.10 muestra las curvas I_D–V_D en cada una de las configuraciones del dispositivo desalineado para el caso BGSD cuando $V_G = 1$ V. En este caso, la corriente de drenador disminuye rápidamente cuando ambas puertas se desplazan hacia el drenador. Cuando el desalineamiento es superior al 10 %, la disminución de corriente es mayor del 5 % con respecto a la corriente de drenador obtenida cuando las puertas están alineadas. Sin embargo, si ambas puertas están desalineadas hacia la fuente menos del 40 %, la corriente es mayor que la obtenida para el caso alineado.

La concentración de electrones en la misma posición que en las seccio-

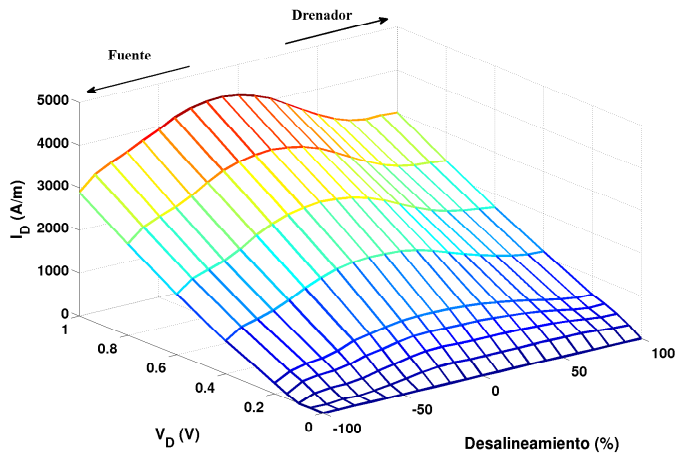


Figura 5.10: I_D frente V_D para cada una de las posiciones de puerta del dispositivo desalineado para la configuración BGSD cuando $V_G = 1$ V.

nes previas, X =17 nm, y para el mismo punto de polarización, $V_D = V_G = 1$ V, se muestra en la figura 5.11 para el caso alineado y cuatro configuraciones diferentes del caso BGSD. La figura muestra que el desalineamiento de ambas puertas produce una distribución uniforme de la concentración de electrones a lo largo del canal. La concentración es mayor cuando ambas puertas están desalineadas hacia el contacto de fuente y decrece rápidamente cuando ambas puertas están desalineadas hacia el contacto de drenador. Esta reducción incrementa la resistencia serie de la fuente causando una rápida caída de la corriente de drenador, como se aprecia en al figura 5.10.

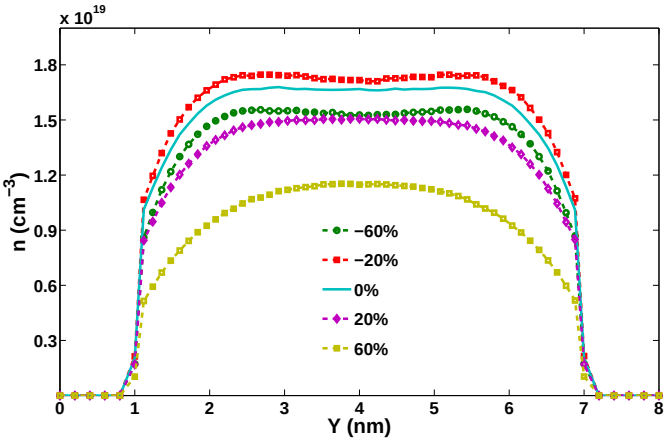


Figura 5.11: Concentración de electrones en el canal en la posición X = 17 nm para el caso alineado y para cuatro posiciones diferentes de desalineamiento de la puerta de la configuración BGSD, cuando $V_D = V_G = 1$ V.

Finalmente, representamos el comportamiento de la corriente de drenador frente al voltaje de puerta para cada una de las posiciones de la puerta de la configuración BGSD cuando $V_D = 100$ mV en la figura 5.12(a). Esta figura muestra que la corriente de drenador disminuye con respecto al

valor obtenido en el caso alineado cuando el desalineamiento es mayor del 10 %, con independencia de que las puertas estén desalineadas hacia la fuente o el drenador, exceptuando el caso en el que las puertas están desalineadas un 20 % hacia el drenador.

En la figura 5.12(b) se muestra la desviación relativa de la corriente de drenador con respecto al caso alineado para la configuración BGSD, cuando $V_D = 100$ mV. En esta figura se observa un ligero estrechamiento de la venta de tolerancia con respecto a las configuraciones anteriores, de modo que ahora las desviaciones relativas de corriente inferiores al 7 % se encuentran en el rango de desalineamiento de puerta [−15 %,18 %].

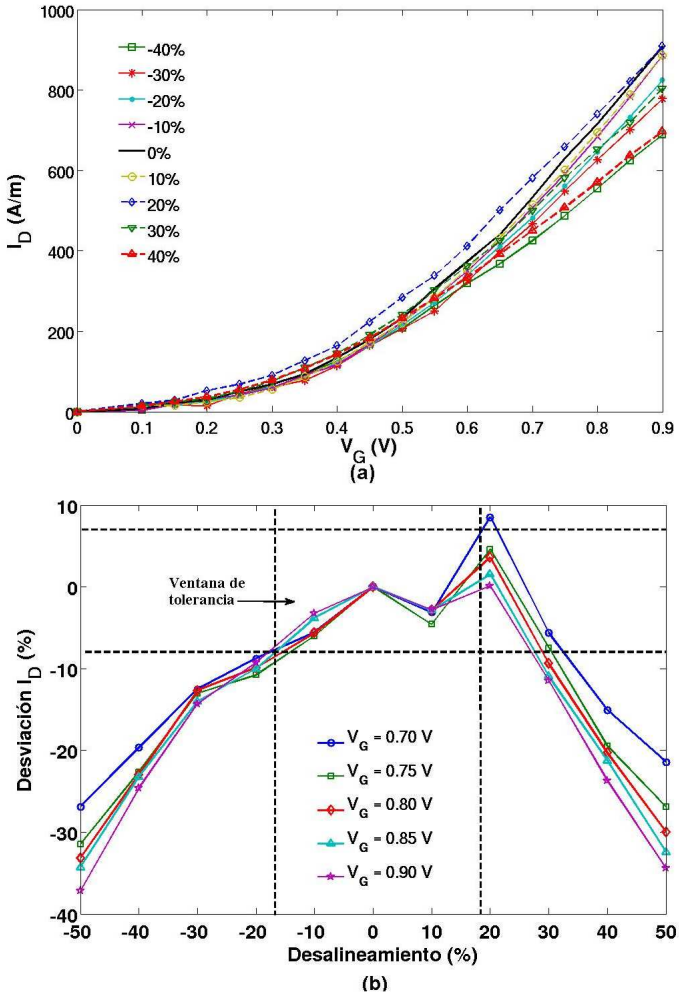


Figura 5.12: (a) Curvas I_D–V_G para cada una de las posiciones de puerta de la configuración BGSD. (b) Comportamiento de la desviación relativa de la corriente de drenador con respecto a la corriente obtenida cuando las puertas están alineadas para el caso BGSD. En todos los casos V_D se ha mantenido constante a 100 mV.

5.2 Estudio de la variabilidad de una interfaz rugosa HfO_2/SiO_2 en el dieléctrico de un transistor SGSOI con el simulador MSB–EMC

Una de las principales fuentes de variabilidad de los transistores MOSFET, conforme se escalan los dispositivos, es la fluctuación del espesor de la capa de óxido (OTF) [ABD+03, MRA10]. Hasta hace unos años el SiO_2 era el material comúnmente empleado como dieléctrico, pero su reemplazo por nuevos materiales puede ser el origen de una nueva fuente de variabilidad en los MOSFETs. Por ejemplo, durante el proceso de deposición en sustratos de silicio del HfO_2 como dieléctrico de puerta del transistor MOSFET, cuando se emplea $Hf(NO_3)_4$ en este proceso, se crea inevitablemente una interfaz irregular de SiO_2 de espesor variable, que puede influir de forma significativa en las características de funcionamiento del dispositivo [ZDZ+06]. El origen de esta lámina de SiO_2 se atribuye a diferentes orígenes, como por ejemplo a la oxidación después de la deposición cuando se expone a la atmósfera, a la oxidación de la superficie del sustrato durante el tiempo de calentamiento en el reactor, a la oxidación del silicio debido a los productos empleados en la reacción de deposición o a la oxidación del silicio debido al $Hf(NO_3)_4$.

En esta sección analizaremos el impacto de las fluctaciones locales del espesor de óxido debidas a la existencia de una interfaz rugosa aleatoria HfO_2/SiO_2, mediante la simulación de 100 dispositivos diferentes con el simulador MSB–EMC.

5.2.1 Descripción de los dispositivos simulados

El dispositivo base empleado en este estudio ha sido un transistor SGSOI MOSFET, de 32 nm de longitud de puerta como el representado en la figura 5.13. La longitud de las regiones de fuente y drenador es de 60 nm y ambas han sido dopadas de forma uniforme con $N_D = 5 \text{ x } 10^{19} cm^3$. El ancho de la longitud de canal es de 6 nm y se encuentra ligeramente dopado ($N_D = 10^{15} cm^3$). El dieléctrico de puerta lo forman dos óxidos. El

superior, por debajo del metal de puerta, está formado por HfO_2 de 4,12 nm de espesor y la capa inferior a esta es de SiO_2 con 0,7 nm de espesor. El conjunto de ambos tiene un espesor de oxido equivalente (EOT) de 1,4 nm.

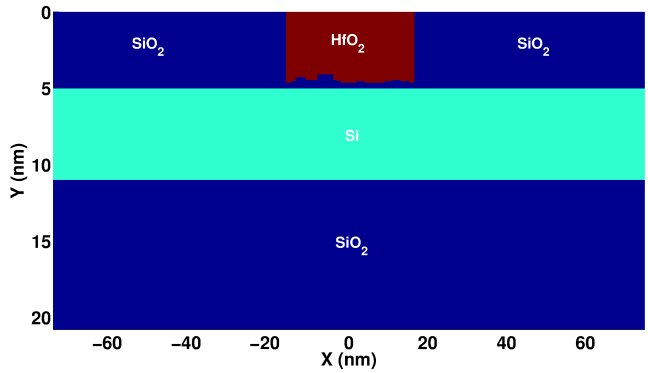


Figura 5.13: Representación de la estructura del SGSOI MOSFET. Los transistores simulados pueden variar su interfaz HfO_2/SiO_2 de forma aleatoria con una penetración máxima de 3,5 Å.

Para estudiar el efecto de la interfaz rugosa entre el HfO_2 y el SiO_2 hemos simulado diferentes dispositivos introduciendo una rugosidad aleatoria. El modelo de rugosidad lo hemos obtenido a partir de una LER, similar a las empleadas para modelar la rugosidad de la interfaz SiO_2/Si del canal [GFW+85, AKB03b], como la que se muestra en la figura 5.14. En esta figura se representa una función H(x) de unidades aleatorias que toma diferentes valores para cada uno de los 32 nodos que forman la malla longitudinal del dispositivo en la interfaz HfO_2/SiO_2. La interfaz rugosa que hemos considerado permite variaciones puntuales en el espesor de cada uno de los óxidos de hasta ±3,5 Å, estas variaciones se calculan en función de la altura de H(x) en donde los valores positivos determinan que el SiO_2 está penetrando en el HfO_2 y los valores negativos que el HfO_2 penetra en el SiO_2.

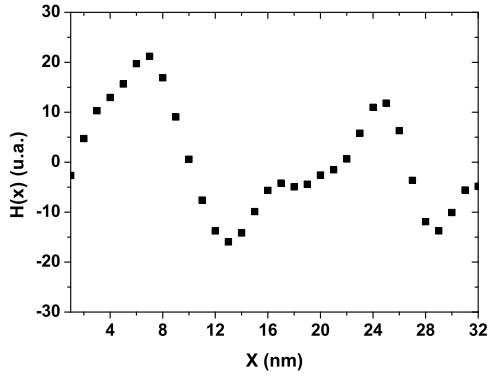


Figura 5.14: Ejemplo de la generación estadística de la interfaz de óxido.

5.2.2 Discusión de los resultados

Para comparar los efectos de la interfaz rugosa del dieléctrico hemos simulado 100 dispositivos con el mismo diseño pero cada uno con diferentes rugosidades. Además, hemos incluido 3 dispositivos en los que la interfaz no tiene rugosidad. En el primero de estos, que hemos denominado caso uniforme, se corresponde con el dispositivo de partida, con una capa de HfO_2 de 4,12 nm de espesor y otra de SiO_2 de 0,7 nm de espesor. Los dos casos restantes se corresponden con lo que hemos denominado casos límite, en donde todo el SiO_2 penetra 3.5 Å en el HfO_2 y el caso opuesto en donde se produce la misma penetración del HfO_2 en el SiO_2. Por lo tanto, para estos dos casos extremos los dispositivos simulados tienen respectivamente un espesor de HfO_2 de 3,77 nm y 4,47 nm, y de SiO_2 de 1,05 nm y 0,35 nm.

De las simulaciones realizadas se observa que la rugosidad en la interfaz de los óxidos que forman el dieléctrico puede causar una variación importante de la corriente de saturación, como se aprecia en la figura 5.15 que representa las curvas I_D–V_D para cada uno de los dispositivos simulados y sus casos límite.

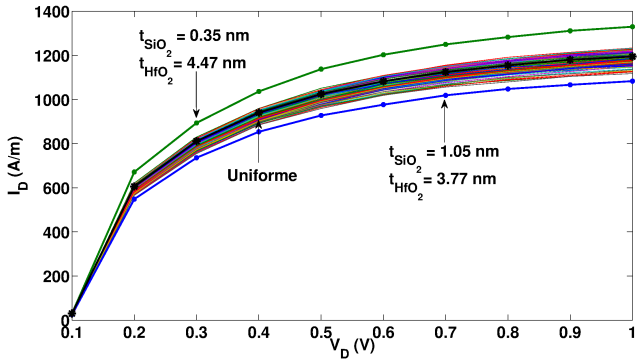


Figura 5.15: Representación de las curvas I_D–V_D de los 100 dispositivos simulados cuando $V_G = 1$ V. Para su comparación se han representado los casos límite y el caso uniforme, para los que no existe rugosidad en la interfaz.

El histograma de la corriente de saturación de los dispositivos con rugosidad en la interfaz HfO_2/SiO_2 obtenida para $V_D = 0{,}9$ V con $V_G = 1$ V se representa en la figura 5.16. Además, en esta figura también se muestran los valores obtenidos en los casos extremos y uniforme así como, el valor medio de la distribución. De los datos representados se ha obtenido que el valor medio de la corriente de saturación es 1166 A/m y un valor de la desviación estándar ligeramente superior al 2 %. En este caso la diferencia relativa entre el valor medio I_D y el valor obtenido para el dispositivo con interfaz uniforme es del 1,2 %. En la tabla 5.1 se comparan los valores de I_D para el caso uniforme con los valores medios y desviación estándar de I_D de los dispositivos con rugosidad en la interfaz cuando V_D es igual 0,5 y 0,9 V. Tanto para valores bajos de V_D (0,5 V) como para valores altos (0,9 V) los valores de la desviación estándar superan ligeramente el 2 %. Sin embargo, la diferencia relativa entre el valor medio de I_D y el valor de corriente obtenido con el dispositivo de interfaz uniforme se hace mayor para valores bajos de V_D, pasando del 1,2 % cuando $V_D = 0{,}9$ V al 2 % cuando $V_D = 0{,}5$ V.

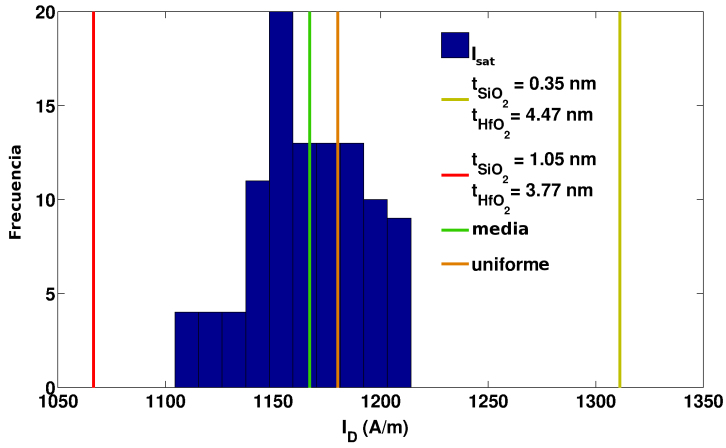


Figura 5.16: Distribución de la corriente de saturación obtenida para V_D = 0,9 V con V_G = 1 V.

	I_D uniforme (A/m)	Media I_D (A/m)	$\sigma(I_D)$ (%)
V_D=0,5V	811,3	794,7	2,13
V_D=0,9V	1180	1166	2,24

Tabla 5.1: Comparación de los valores de I_D para el caso uniforme con los valores medios y desviación estándar de I_D de los dispositivos con rugosidad en la interfaz cuando V_D es igual a 0,5 y 0,9 V, con V_G = 1 V.

La figura 5.17 muestra el efecto de la variabilidad en las curvas I_D–V_G cuando V_D = 1 V. En esta figura, también se observa una variación significativa de la corriente de drenador con respecto al caso uniforme.

En la figura 5.18 se representa el histograma de la corriente de drenador de los dispositivos con rugosidad en la interfaz para V_G = 0,9 V con V_D = 1 V. De los datos representados se ha obtenido que el valor medio de la corriente de drenador es 970,5 A/m y el valor de la desviación estándar de casi un 3 %. En este caso la diferencia relativa entre el valor medio I_D

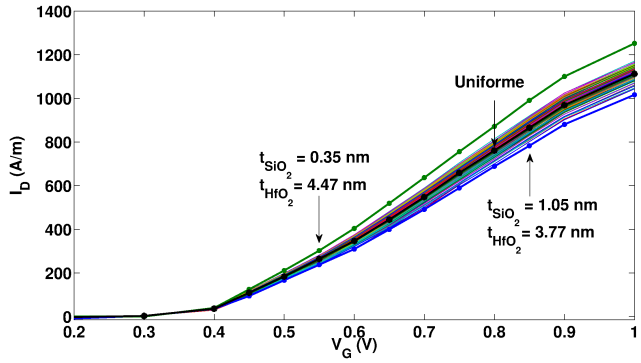


Figura 5.17: Representación de las curvas I_D–V_G para los 100 dispositivos simulados cuando $V_D = 1$ V. Para su comparación se han representado los casos límite y el caso uniforme, para los que no existe rugosidad en la interfaz.

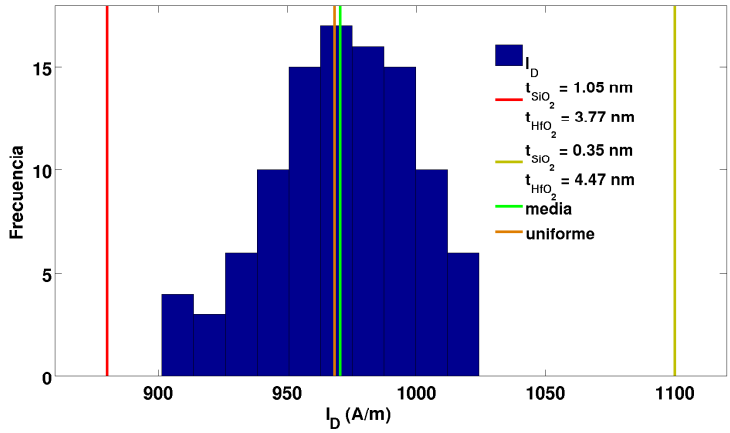


Figura 5.18: Distribución de la corriente de drenador obtenida para $V_G = 0{,}9$ V con $V_D = 1$ V.

y el valor obtenido para el dispositivo con interfaz uniforme es mínima (0,21 %). En la tabla 5.2 se comparan los valores de I_D para el caso uniforme con los valores medios y desviación estándar de I_D de los dispositivos con rugosidad en la interfaz cuando V_G es igual a 0,5 y 0,9 V. En este caso se observa mayor dispersión de la corriente en la región sub–umbral, ya que cuando $V_G = 0{,}5$ V el valor de la desviación estándar (4,21 %) es mayor que el obtenido cuando $V_G = 0{,}9$ V (2,87 %). Sin embargo, la diferencia relativa entre el valor medio de I_D y el valor de corriente obtenido con el dispositivo de interfaz uniforme es prácticamente despreciable en ambos casos e inferior al 1 %.

	I_D uniforme (A/m)	Media I_D (A/m)	$\sigma(I_D)$ (%)
$V_G = 0{,}5$V	183,5	183,9	4,21
$V_G = 0{,}9$V	968,4	970,5	2,87

Tabla 5.2: Comparación de los valores de I_D para el caso uniforme con los valores medios y desviación estándar de I_D de los dispositivos con rugosidad en la interfaz cuando V_G es igual a 0,5 y 0,9 V, con $V_D = 1$ V.

Conclusiones

La simulación de semiconductores es una herramienta básica en el diseño de nuevos transistores. Gracias a la simulación, el investigador puede estudiar nuevos dispositivos, con diferentes características de diseño y nuevos materiales. Sin embargo, el escalado de los transistores provoca que los modelos que es necesario emplear sean más complejos, aumentando el tiempo de cálculo de las simulaciones. Además, la reducción del tamaño también incrementa el efecto de las fluctuaciones, provocando una mayor variabilidad de los parámetros de funcionamiento de los dispositivos y al mismo tiempo, incrementando el número de simulaciones necesarias para que estos efectos puedan ser estudiados.

En este contexto, la simulación de semiconductores se enfrenta a dos retos computacionales: reducir el tiempo de cálculo de los simuladores sin que eso suponga reducir la complejidad de los modelos empleados y aumentar el número de recursos disponibles para atender las necesidades de los estudios de fluctuaciones. En este trabajo se han estudiado dos soluciones a estos retos, aplicándolas de modo concreto a la simulación 2D Monte Carlo de transistores MOSFET. Para reducir el tiempo de cálculo hemos propuesto la paralelización de los simuladores y para incrementar los recursos hemos evaluado el uso de infraestructuras Grid.

A continuación se detallan las aportaciones y conclusiones principales de cada una de estas soluciones:

Programación paralela aplicada a la simulación de dispositivos semiconductores

Durante los últimos 5 años hemos visto un fuerte desarrollo de las arquitecturas multi–núcleo, hasta el punto que el 90 % de los supercomputadores de la lista TOP500 dispone en la actualidad de este tipo de arquitecturas. Una ventaja adicional de este tipo de procesadores es que hoy en día se encuentran en los ordenadores de escritorio. Esto nos permite explotar el paralelismo durante la etapa de implementación de nuevos modelos. En nuestro caso, hemos intentado aprovechar las posibilidades de paralelismo que ofrecen estos recursos, paralelizando dos simuladores diferentes, eligiendo para ello el estándar para máquinas de memoria compartida OpenMP.

El primero de ellos, con un menor coste computacional, es el simulador MV–ECBE–EMC. Durante el proceso de paralelización de este código hemos comprobado que a pesar de haber elegido el estándar OpenMP por su teórica sencillez para desarrollar códigos paralelos, la paralelización del simulador ha sido una tarea compleja ya que inicialmente el código no fue desarrollado para ser paralelo. Un vez paralelizado, el simulador MV–ECBE–MC paralelo nos ha permitido reducir el tiempo de cálculo por un factor 1,81 empleando un máximo de cuatro núcleos. Para más de cuatro núcleos se alcanza el límite de escalabilidad debido a que no todo el código del simulador es paralelizable y el porcentaje de código secuencial es demasiado elevado. Por lo tanto, las pruebas realizadas resaltan la importancia que tiene el perfilado del código. Éste nos da información de la estructura del simulador y de que subrutinas consumen más tiempo de cálculo.

El segundo simulador paralelizado es el MSB–EMC, con un mayor coste computacional que el MV–ECBE–EMC pero con más porcentaje de código paralelo. El tiempo de simulación de la versión paralela del MSB–EMC se ha reducido por un factor 7.4 cuando usamos 8 núcleos y hasta por un factor 10 cuando usamos 16 núcleos. Estas reducciones en el

tiempo de ejecución son un ejemplo de las ventajas de la paralelización en la simulación Monte Carlo de transistores, ya que permite aprovechar la capacidad de paralelismo de los procesadores multi–núcleo actuales.

Computación Grid aplicada a la simulación de nanodispositivos semiconductores

Las tecnologías Grid han ido evolucionando durante los últimos 15 años y gracias a ello, han surgido nuevas infraestructuras Grid nacionales, como la es–NGI, e internacionales, como EGI o TeraGrid. En este estudio hemos utilizado la es–NGI como infraestructura de pruebas, ya que sus 5000 núcleos de ejecución y 500 TB en 2010, la convertían en una infraestructura muy interesante para realizar estudios estadísticos de dispositivos semiconductores. Sin embargo, a pesar de la potencia de cálculo que encontramos en estos recursos, creemos que el *middleware* de Grid podría ser mejorable, para facilitar y extender el uso de estos recursos a la comunidad científica. Para realizar nuestras pruebas hemos usado el *middleware* de gLite. Además, hemos desarrollado un sistema de envío y monitorización basado en este *middleware* para gestionar automáticamente aquellos trabajos que sufrían algún fallo durante su envío o ejecución.

Uno de los principales problemas del Grid es la heterogeneidad de los recursos. En el caso de la es–NGI nuestros trabajos podían ejecutarse en potentes procesadores Xeon de última generación o en antiguos PentiumD. Las pruebas realizadas confirmaron que esta heterogeneidad de recursos afectaba a los tiempos de cálculo, encontrando diferencias de hasta un 55 % en los tiempos de ejecución obtenidos en los recursos a los que teníamos acceso. A pesar de estas diferencias creemos que este tipo de infraestructuras son muy interesantes para realizar estudios estadísticos de dispositivos semiconductores. Además, disponemos de un gran número de procesadores multi–núcleo con los que podríamos reducir el tiempo de cálculo de las simulaciones, cuando el número de trabajos fuese inferior al número de núcleos disponibles. Finalmente, creemos que deberíamos destacar la importancia de desarrollar políticas de gestión y uso de estos recursos, así como de potenciar el desarrollo tecnológico basado en la creación de

nuevos servicios que permitan explotar sus posibilidades, debido a su potencial para muchas áreas de investigación que requieren de la simulación y el procesado de datos.

Para completar este trabajo, hemos estudiado algunas fuentes de variabilidad que afectan a las características de funcionamiento de los transistores SOI MOSFET. En concreto, se ha tenido en cuenta el efecto del desalineamiento de puerta en transistores DGSOI MOSFET y el impacto de la variabilidad en el espesor de óxido, debida a una interfaz rugosa de HfO_2/SiO_2 en el dieléctrico de un transistor SGSOI. En ambos estudios hemos utilizado los simuladores paralelizados previamente, el simulador MV–ECBE–EMC para estudiar el efecto del desalineamiento de puerta y el MSB–EMC para estudiar el impacto de la variabilidad en el espesor de óxido. A continuación se presentan las aportaciones y conclusiones principales a las que hemos llegado:

Simulación del desalineamiento de puerta en transistores DGSOI MOSFET con el simulador MV–ECBE–EMC

Para este estudio hemos considerado un dispositivo SOI de doble puerta por ser uno de los candidatos para reemplazar los transistores MOSFET basados en la tecnología tradicional. El DGSOI estudiado tiene 10 nm de longitud de puerta y las simulaciones del desalineamiento las hemos realizado con el simulador MV–ECBE–EMC.

Para analizar el efecto del desalineamiento de las puertas hemos simulado tres configuraciones diferentes: TGM, cuando la puerta superior está desalineada, BGOD, cuando ambas puertas están desalineadas en direcciones opuestas y BGSD, cuando ambas puertas están desalineadas en la misma dirección. Los resultados obtenidos para diferentes configuraciones de desalineamiento de las puertas muestran, de forma general para todas las configuraciones, desviaciones de la corriente de drenador con respecto al caso alineado inferiores al 7 % para desalineaciones inferiores al 20 % de la longitud de puerta. Estos resultados coinciden con los obtenidos en estudios anteriores para transistores con longitudes de puerta superiores. Además, los datos obtenidos en este estudio muestran mejores

resultados cuando el dispositivo está desalineado hacia la fuente menos de un 20 %, mejorando en algunos casos los resultados obtenidos en la región de saturación para el dispositivo alineado. Este comportamiento se debe al incremento de la concentración de electrones en la región del canal próxima a la fuente que hace que disminuya su resistencia serie.

Para la configuración BGSD también se observa que el rendimiento es mejor cuando las puertas se encuentran desalineadas hacia la fuente que cuando lo están hacia el drenador. Esto se debe a que al desalinear las puertas hacia el drenador el control sobre el canal es peor. Esta configuración tiene especial relevancia para dispositivos fabricados con técnicas de auto–alineamiento habitualmente empleadas en la fabricación de transistores FinFET.

En general y a la vista de los resultados obtenidos con el simulador MV–ECBE–EMC se observa que los efectos de desalineamiento, mostrados en estudios previos experimentales y de simulación, se producen también en el DGSOI de 10 nm que hemos estudiado cuya longitud de puerta es entre 5 y 20 veces menor que el de los transistores de los trabajos anteriormente mencionados.

Estudio de la variabilidad de una interfaz rugosa HfO_2/SiO_2 en el dieléctrico de un transistor SGSOI con el simulador MSB–EMC

En este estudio hemos analizado el impacto de las fluctuaciones locales del espesor de óxido, debidas a la existencia de una interfaz rugosa aleatoria de HfO_2/SiO_2. Para ello, hemos simulado 100 dispositivos SG-SOI con diferentes interfaces rugosas HfO_2/SiO_2 generadas aleatoriamente, utilizando el simulador paralelo MSB–EMC. Los resultados obtenidos muestran que la presencia de una interfaz rugosa aleatoria en el óxido puede afectar a las características de funcionamiento de los transistores. En concreto sobre la muestra de cien dispositivos simulados, hemos encontrado valores de la desviación estándar de la distribución de la corriente de saturación de entre un 2 y 4 %, en función del punto de polarización del transistor. Los datos de este estudio son un ejemplo de la importancia que tiene el análisis de nuevas fuentes de fluctuaciones, que surgen

al emplear nuevos materiales y técnicas de fabricación, y que de forma combinada pueden tener un impacto significativo en la variabilidad de los dispositivos.

Trabajo futuro

Durante la realización de este trabajo se ha seguido una línea de investigación orientada hacia el aprovechamiento de arquitecturas paralelas y Grid en la simulación Monte Carlo de transistores SOI MOSFET. Como consecuencia, han surgido nuevas líneas abiertas para un trabajo futuro, especialmente en lo que se refiere a la aplicación de la computación paralela y las tecnologías Grid y Cloud a la simulación de dispositivos semiconductores, así como la posibilidad de transferir este conocimiento a otras áreas científicas.

Con respecto a los simuladores 2D Monte Carlo paralelizados, el simulador MSB–EMC podría extenderse a una versión 3D resolviendo la ecuación de Schrödinger en dos dimensiones en el plano perpendicular al transporte de electrones, actualmente se hace una solución 1D de la ecuación en este plano. Esta extensión facilitaría la simulación de transistores 3D, como los Nanohilos o FinFETs. Durante el desarrollo de este simulador se continuará con la paralelización y optimizado de los nuevos modelos que se incorporen.

Siguiendo con la simulación de transistores, existen muchas opciones de trabajo abiertas en el análisis del impacto de las fluctuaciones sobre las características de funcionamiento de los transistores. Sería muy interesante continuar nuestro primer acercamiento al estudio del impacto de las variaciones del espesor del óxido inducidas por la aparición de interfaces rugosas aleatorias en la interfaz HfO_2/SiO_2. Para continuar este trabajo se podrían comparar los resultados obtenidos con el modelo implementado, que nos permite introducir directamente la rugosidad en la interfaz el óxido, con aquellos que se obtendrían de la implementación de un mecanismo de dispersión con el que se tuviera en cuenta este efecto. Además otras fuentes de fluctuaciones, relacionadas con las variaciones en los parámetros de diseño, como el espesor de silicio, longitud de puerta, etc., también

podrían ser estudiadas.

Por otro lado, nuestro grupo de investigación ha desarrollado varios simuladores 3D paralelizados con MPI, que emplean métodos de descomposición de dominios, en los que podría ser muy interesante aplicar paralelización híbrida OpenMP/MPI para reducir el tiempo de simulación. Continuando con esta línea de de investigación orientada hacia el paralelismo, resultaría muy interesante evaluar las posibilidades de uso de aceleradores *hardware* como GPUs.

Finalmente, una de las líneas de investigación más novedosa y que podría ser muy útil en el campo de la simulación en general, es la que se ha denominado como *e–Science*. En concreto en el campo de la simulación de transistores y probablemente en otra áreas que se dedican a la simulación, resulta necesario desarrollar interfaces *Web* de envío y monitorización de trabajos, desde los que se pueda elegir el uso de diferentes recursos, como el *cluster* de un grupo de investigación, o el envío de trabajos a los recursos de un Grid o un Cloud. Además, adoptando el modelo Cloud denominado *Software* como servicio (*Software as a Service*, SaaS) resultaría muy útil la gestión centralizada tanto de datos de simulación, como de datos experimentales. Estos se almacenarían en una base de datos que contendría los resultados de las simulaciones de diferentes dispositivos a partir de los cuales se podrían hacer diversas consultas como por ejemplo, comparar los resultados experimentales de un dispositivo con aquellos obtenidos a través de la simulación.

APÉNDICE A

Parallelisation and Optimisation of a 2D Monte Carlo Simulator on Grid and Cluster Architectures: Study of Fluctuations on SOI MOSFETs

Simulation of nanoelectronic devices is a research area that allows us to study the physical phenomena affecting their electrical behaviour, thanks the development of new physical models. The historical trend of the processor manufacturers is to scale down the size of the transistors to increase their performance, this size reduction makes necessary the use of very accurate simulation models in order to properly capture the physical phenomena affecting the electrical behaviour of these devices. Consequently, the increase of the complexity of the model causes a dramatic increase in the simulation time. Furthermore, the scaling down of device size may produce design deviations and material variations during the fabrication process. The influence of these design variations on the device performance can be studied simulating different device configurations that can be obtained during the fabrication processes [VSS+09, SGG+10]. However, these simulations need massive computational resources to perform

statistical studies on realistic 3–D geometries [AKB03a].

This appendix summarises the main results and achievements show in this dissertation. The structure of the appendix is as follows: Sections A.1 and A.2 describe the parallelisation and optimisation of two Quantum Monte Carlo simulators of MOSFET transistors. In Sections A.3 and A.4 we asses some advantages of Grid infrastructures for simulation of semiconductor devices. Finally, Sections A.5 and A.6 summarise the main results obtained from two variability studies performed with the parallel quantum Monte Carlo simulators.

A.1 Parallel implementation using OpenMP of a 2D Quantum–Corrected Multi–Valley Ensemble Monte Carlo simulator for MOSFETs transistors

Simulation techniques of semiconductor devices can be classified following two main criteria: the computational cost and the level of accuracy. For example, classical Drift–Diffusion simulations have been used for decades due to their low computational cost. However, their models need to be fitted with experimental results, including non–physically based parameters which can be only justified because experimental curves are well reproduced. At the other end of the spectrum, full quantum simulators based on the solution of the Schrödinger equation or the Non–Equilibrium Green's Functions theory (NEGF) have also been developed [Dat00] but such codes are very time consuming from the computational point of view and have a limited applicability since the inclusion of a comprehensive set of scattering mechanisms and actual device geometry is limited.

Among these extreme approaches, semi–classical simulators including quantum corrections have several advantages, such as a reduced computational cost, the possibility of considering a wide variety of scattering mechanisms and high accuracy for devices with thin silicon channels [RKR+05]. Ensemble Monte Carlo (EMC) simulators have been widely used, adding

a quantum term in order to correct the electrostatic potential resulting from the classical solution of the Poisson's equation to mimic the electron concentration profile obtained when the Schrödinger equation is solved. The calibration of such corrections is not free from fitting parameters (e.g., carrier effective mass in the density gradient model) but the obtained results are still accurate from the transport point of view [Anc90, WSF01, SGRG06].

The 2D Quantum–Corrected Multi–valley Ensemble Monte Carlo simulator for MOSFET transistors (MV–ECBE–EMC) solves the Boltzmann Transport Equation (BTE) self–consistently with Poison's equation. Quantum effects are included in this code via a Multi–Valley version of the Effective Conduction Band Edge (MV–ECBE) model. In this approach, the impact on both electrostatic and transport properties of each valley of the Silicon conduction band are included by coupling the 2D Poisson's equation and the Transport Boltzmann equation with the ECBE equation solved for the *j–th* valley,

$$V_j^* \simeq V + \frac{\hbar^2}{4qrV_T}\left\{\nabla\cdot\left[\left(\frac{\overleftrightarrow{1}}{m}\right)_j\cdot\nabla V_j^*\right] + \frac{1}{2V_T}\left[\nabla V_j^*\cdot\left(\frac{\overleftrightarrow{1}}{m}\right)_j\cdot\nabla V_j^*\right]\right\} \tag{A.1}$$

where V_j^* is the effective potential, $V_T = K_B T/q$ represents the thermal potential and r is a parameter whose value varies from 1 for pure states (low temperatures or very strong confinement) to 3 for mixed states (high temperatures or weak confinement). The orientiation of the simulated device is (100) and the values of the longitudinal and transversal effective masses are, respectively, $m_l = 0.91$ and $m_t = 0.19$.

The corrected potentials substitute the electrostatic potential to perform the drift calculations. In this way, the drift field for the *j–th* valley is calculated as:

$$\overrightarrow{E_j} = -\nabla_r V_j^*. \tag{A.2}$$

As a direct consequence of different driving forces for each valley, the corresponding populations have to be calculated in a self–consistent way during the simulation. This is performed via intervalley scattering which plays a fundamental role in the calculations since it is the only implemented mechanism that allows charge transfer from one valley to another. A zeroth–order approach with three phonons has been used following the work presented in [FL93a]. Acoustic phonons, Coulomb and the surface roughness scattering model introduced in [GRLV+99] have also been included. In all the cases, the scattering expressions have been adapted to the pseudo–2D gas approach. This is performed by including the calculation of overlapping factors between initial (i) and final (j) valleys.

$$F_{ij} = L_D \int_{T_{Si}} |\Psi_i(y)|^2 |\Psi_j(y)|^2 dy \tag{A.3}$$

where L_D is the Debye length, y the confinement direction and

$$|\Psi_i(y)|^2 = \frac{n_i(y)}{\int_{T_{Si}} n_i(y) dy} \tag{A.4}$$

is the envelope of the electron wave function in the pseudo 2D gas approach. The scattering rates are then weighted by F_{ij}. In this way, size quantization is taken into account and the valley population is calculated self–consistently.

A.1.1 Benchmark device and profiling of the sequential simulator

The first step to parallelise the MV–ECBE–EMC code is to know which subroutines are the most computationally demanding in order to evaluate their possible parallelisation. A simulation test has been performed to profile the code, using as benchmark device a 10 nm channel length DGSOI MOSFET transistor, see figure A.1. The silicon thickness is 6 nm, the effective oxide thickness is 10 Å and the metal gate contacts have a work–function value of 4.6 eV. The drain and source constant doping

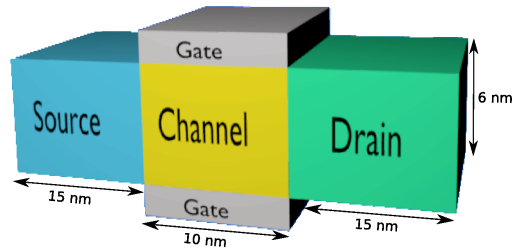


Figure A.1: Benchmark device for a DGSOI structure.

concentration is $N_D = 9$ x 10^{19} cm^{-3} and the channel is $N_A = 10^{15}$ cm^{-3}.

MC simulator was written in Fortran language and it has been parallelised using OpenMP API [opea]. Table A.1 shows the computational load of the subroutines obtained from the profiling of the code prior to the parallelisation. A graphical representation of this profiling is shown in figure A.2.

The subroutine *Monte3d* consumes 99.9% of the total execution time. This subroutine is called only once and it has the subroutines and sentences required to carry out the Monte Carlo simulation. Therefore, the rest of subroutines from table A.1 belong to the *Monte3d* subroutine.

The subroutine *Free* consumes 18.8% of the total execution time and it is called 10001 times. This subroutine simulates the electron transport in the semiconductor device. The number of particles for each iteration is variable and its value is around 180,000.

The subroutine *Ecbemc* consumes 30.2% of the total execution time and it is called 10002 times. This subroutine calculates the value of the quantum potential.

The subroutine *Renew* consumes 31.5% of the total execution time and it is called 10002 times. This subroutine updates the number of particles of the device after the transport simulation.

$\%t_{total}$	$Time(s)$	$Calls$	$Name$
99.9	3411.75	1	*Monte3d*
31.50	1075.61	10002	*Renew*
30.20	1030.97	10002	*Ecbemc*
18.80	642.78	10001	*Free*
4.7	160.14	10002	*Charge*
3.8	129.42	10001	*Poisson*
2.9	99.17	10001	*Velocity*
1.7	56.83	10001	*Current*

Table A.1: Profiling of the sequential code using the open source program Gprof. *Time (s)* is the time spent in each subroutine, *Calls* indicates the number of times that each subroutine is called and *Name* is the name of each subroutine.

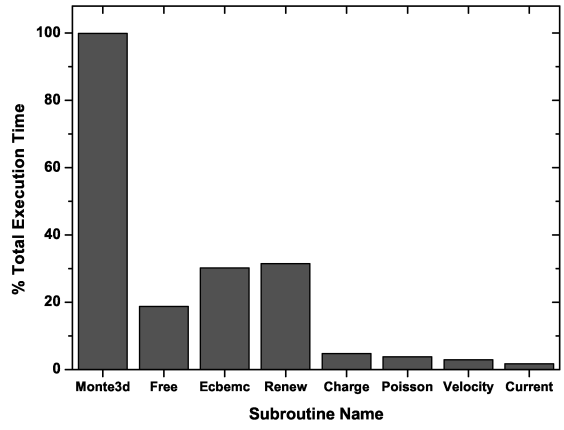


Figure A.2: Profiling of the simulation program. Percentage of the total execution time for the subroutines shown in Table A.1

.

The subroutine *Charge* consumes 4.7% of the total execution time and it is called 10002 times. This subroutine calculates the value of the electron charge for each node of the device mesh.

The subroutine *Velocity* consumes 2.9% of the total execution time and it is called 10001 times. This subroutine calculates the electron velocity.

Finally, the subroutine *Current* consumes 1.7% of the total execution time and it is called 10001 times. This subroutine calculates the value of the electron current taking into account the number of electrons which come in and leave the device.

From this analysis, a set of subroutines (74 percent of the execution time in a single processor machine) was chosen to be parallelised. All the subroutines listed in Table A.1 have been parallelised with two exceptions: *Monte3d*, because it contains the temporal loop of the Monte Carlo simulation, and *Poisson*, since it solves the Poisson equation using an algorithm with inter–iteration dependencies.

A.1.2 Parallelisation of the 2D Quantum–Corrected Multi–Valley Monte Carlo simulator

The flow chart for the parallel version of MV–ECBE–EMC simulator is shown in figure A.3. A parallel region was created just before the loop of evaluation of the iteration time and all the created threads evaluate this loop and the parallelised subroutines.

The parallelised program has been executed in a two–processor server with two quad–core Intel Xeon 5355 at 2.6 GHz and 8 GB main memory. Figure A.4 shows the speed up for the simulation as a function of the number of processors used. Using four processors the speed up is 1.81, being the ideal value 2.07. The ideal value was calculated using Amdahl's Law (A.5), where P is the proportion of a program that can be done in parallel and n is the number of processors. Table A.2 shows the execution time and the speed up as a function of the number of CPUs used.

$$A_{id} = \frac{1}{(\frac{P}{n}) + (1 - P)} \tag{A.5}$$

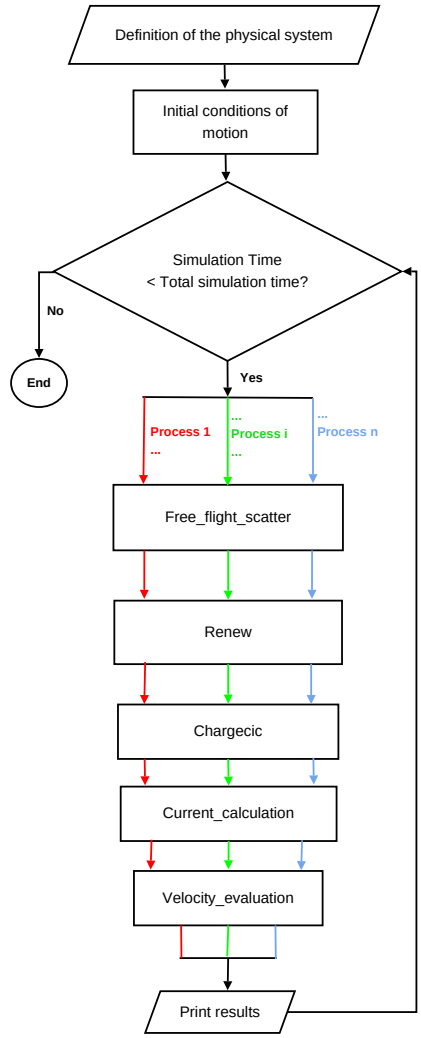


Figure A.3: Flowchart for parallel MV–ECBE–EMC simulator.

Proc	Time(s)	Sup	Sup_{id}
1	3795	1	1
2	2664	1.40	1.56
4	2061	1.81	2.14
6	2002	1.87	2.32
8	1973	1.89	2.43
$\text{T}_{seq}(s)$	5169		

Table A.2: Execution time (Time) in seconds, speed up (Sup) and ideal speed up calculated using Amdahl's law (Sup_{id}) as a function of the number of processors used (Proc).

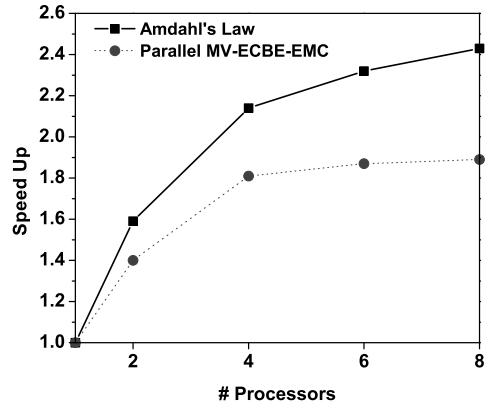


Figure A.4: Speed up for the parallel version of the MV–ECBE–EMC simulator.

Parallelisation results depending on the number of superparticles

The Monte Carlo method, as applied to charge transport in semiconductors, consists of a simulation of the motion of one or more electrons

inside the semiconductor device [JL89]. Computational limitations of simulating the motion of all carriers contained in a device force to select a subset of the electron population, containing N superparticles, as a representative sample of the overall distribution, representing each one a number of electrons, eps. Initially, this number was fixed to 100,000.

However, in most of the parallelised subroutines, the computational load depends on the number of electrons per superparticle because the number of loop iterations is dependent on the number of superparticles. Therefore, if the simulation program considers more electrons per superparticle, the number of iterations will decrease, thus reducing the simulation time.

The Monte Carlo method is a statistical procedure for the solution of mathematical problems, so the results obtained are always affected by some statistical uncertainty. As general rule, the statistical precision of the results increases as the square root of the number of trials and, therefore, the amount of computer time necessary to appreciably improve the quality of the results quickly becomes very considerable [JL89]. Figure A.5 shows, for the non–parallelised version of the code, the increase in the execution time when the number of superparticles is increased considering less electrons per superparticle.

It is possible to use a parallelised version of the program code to improve the accuracy of the simulation without increasing the execution time. In this case we have used a two–processor server with two dual–core Intel Xeon 3 GHz and 8 GB main memory. Figure A.6 shows the execution time as a function of the number of electrons per superparticle for the sequential simulation and for two and four processors simulations. If we execute the code on a single processor using 2 x 10^5 eps, it will require approximately the same time as the execution of the code on two or four processors with 1 x 10^5 eps or 0.5 x 10^5 eps, respectively.

Figure A.7 represents the dependence of the speed up on the number of processors for different numbers of superparticles, related to the computational load. An improvement in the speed up is noticed when the number of superparticles is doubled or quadrupled.

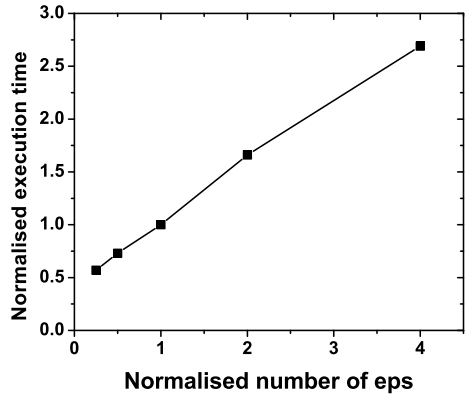


Figure A.5: Normalised execution time versus normalised number of superparticles. 100,000 electrons per superparticle were considered as a reference for both the normalised execution time and the normalised number of superparticles.

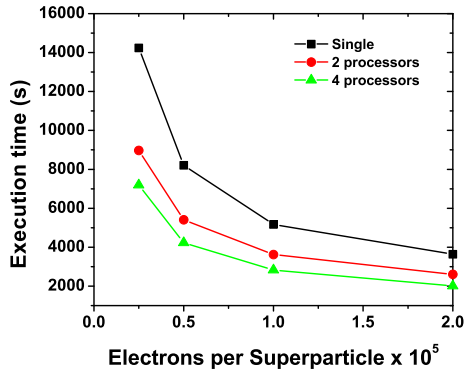


Figure A.6: Dependence of execution time on the number of electrons per superparticle for one, two and four processors.

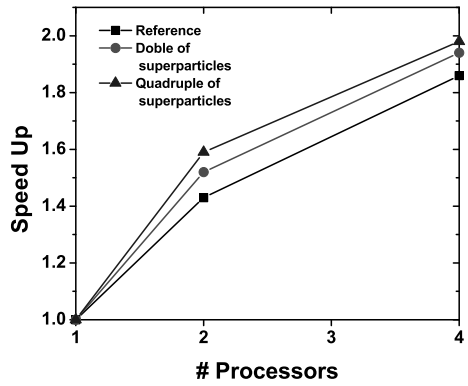


Figure A.7: Speed up versus number of processors for different number of superparticles. 100,000 electrons per superpaticle were considered as a reference.

A.2 Optimisation and parallelisation of a 2D Multi–Subband Ensemble Monte Carlo simulator for MOSFETs transistors

Semiconductor device simulators have evolved from simple approaches, such as the compact model, where transistors are replaced with basic devices like resistors and capacitors, to more complex models where quantum effects are considered [Sch98]. Currently, the very small dimensions of the new generation transistors makes mandatory the introduction of quantum effects that involve the solution of the Schrödinger equation, which greatly increases the computational cost. Furthermore, when the semiconductor dimensions are shrunk to the nanometre regime, several sources of fluctuations can significantly affect their performance and reliability [VSS$^+$09, VSA$^+$10, WLV$^+$05, AKB03a]. At these scales, the simulation of device variations requires studies on a statistical scale that are computationally very expensive [RMR$^+$09]. The large amount of memory

and floating point operations needed in this kind of calculations demands the use of parallel applications and appropriate algorithms in order to obtain the maximum performance and reduce simulation times. The sequential simulators are very inefficient because they do not exploit the parallelism available on muticore, shared-memory architectures [CGP07].

This section presents the optimisation and parallelisation of a 2D Multi–Subband Ensemble Monte Carlo simulator (MSB–EMC) using the OpenMP API [opea]. This program solves the Boltzmann Transport Equation (BTE) in the transport direction, considering the quantum effects with a self–consistent solution of the Schrödinger equation in the confinement direction, perpendicular to the transport direction [SGG+10].

A.2.1 Multi–Subband ensemble Monte Carlo method

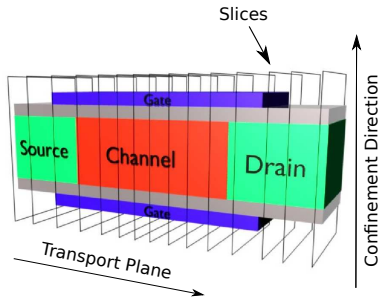


Figure A.8: Representation of the Double Gate MOSFET. The BTE equation is solved in the transport plane and the 1D Schrödinger equation is solved in the confinement direction for each grid point in the transport direction. The drain and source doping concentration in N_D=10^{20} cm^{-3} and the channel doping concentration is N_A=10^{15} cm^{-3}

The 2D MSB–EMC simulator is based on mode–space approach of the transport [VRD+02b] and solves the Boltzmann Transport Equation (BTE) in the transport direction using the Ensemble Monte Carlo (EMC) method [JL89]. Quantum effects are introduced via the self–consistent

solution of the Schrödinger equation in the confinement direction, perpendicular to the transport direction.

The BTE (A.6) describes the transport phenomena in a semiclassical approach where the fundamental quantity is the carrier distribution function $f(\vec{r}, \vec{v}, t)$.

$$\frac{\partial f}{\partial t} + \vec{v} \cdot \nabla_r f + \frac{\partial k}{\partial t} \cdot \nabla_k f = \left.\frac{\partial f}{\partial t}\right|_{coll} \tag{A.6}$$

The left–hand side describes the time evolution in the phase–space of the carrier distribution function. $\vec{v}$ is the group velocity of carriers and $\partial k/\partial t$ is proportional to the driving field. According to the mode–space approach, the drift field is calculated from the derivative of $E_{i,\nu}(x)$, e.g. the driving force is different for each subband corresponding to a given valley.

The right–hand side represents the change of $f(\vec{r}, \vec{v}, t)$ due to collisions, in our case, phonon and interface roughness scattering processes. The probabilities of the scattering mechanisms are tabulated in the simulator and they are calculated during the simulation [Lun00].

For thin body devices quantum confinement effects are quite important and must be considered properly. The MSB–EMC simulator takes these effects into account by coupling the BTE equation with the Schrödinger equation in the confinement direction [WR03]. From the simulation point of view, our transistor is considered as a stack of slices perpendicular to the transport direction (see figure A.8). The electrostatics of the system is calculated from the self-consistent solution of the 2D Poisson's equation and the 1D Schrödinger equation solved for each slice and conduction band valley [SGG$^+$10].

A.2.2 Description of the Multi–Subband Ensemble Monte Carlo simulator

Figure A.9 depicts a block diagram of the 2D MSB–EMC simulator. As first step, a starting solution of the simulated device (*Initial Conditions* and *Initial Multisubband* blocks) is required in order to obtain the initial

values of the electric field and the scattering table needed for the first iteration of the MC simulation. This initial guess is calculated through the self–consistent solution of the Poisson and Schrodinger equations at equilibrium.

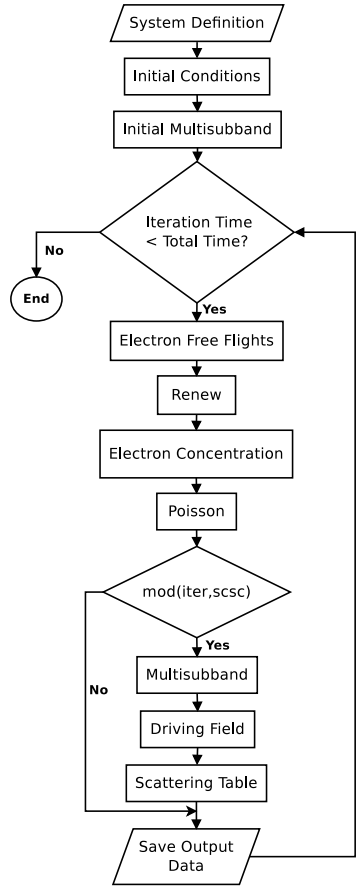


Figure A.9: Block diagram of the 2D Multi–Subband MC simulator. The *iter* variable indicates the iteration number and the *scsc* variable indicates the number of iterations that are carried out before the self–consistent solution of 1D Schrödinger and 2D Poisson equations.

The time domain of the MC simulation is discretised in short intervals, namely iteration times, where the motion of the electrons is simulated. After the flight of all superparticles a new electron distribution is obtained from the evaluation of the new positions and properties of them, and therefore, the electrostatic potential has to be updated solving the 2D Poisson equation in the transport plane.

A drawback of this simulator (compared with semi-classical EMC codes) is its computational burden due to the calculation of the scattering table for each slice for every new solution of the Schrödinger equation to keep the self-consistence of the simulation. Bearing in mind the high computational load of these calculations, it is important to look for appropriate strategies to reduce their cost. One possible strategy is based on the reduction of the number of times that the 1D Schrödinger and the 2D Poisson equations are solved self–consistently. To use this option it is necessary to assume a certain trade–off between the accuracy of the solution and the computational load of the system. This strategy is implemented in the simulator through the *scsc* variable (see figure A.9) which indicates the number of iterations that are carried out before the 1D Schrödinger and the 2D Poisson equations are solved self–consistently. In the last step of the simulation flow, the obtained values of the driving field and the probability rates of the scattering tables are employed as initial conditions for the next iteration. This iterative process is repeated up to the end of the simulation time.

Figure A.10 shows the percentage of the execution time for the five most computationally costly subroutines of the code versus the *scsc* variable. These results have been obtained for the simulation of a stationary state of 1 ps. If the self–consistent solution of the Schrödinger and Poisson equations is obtained in every iteration; i.e. *scsc*=1; the total simulation time is 6585.58 s. In this case the *Multisubband* and *Scattering Table* subroutines are the most computationally demanding. The *Multisubband* subroutine solves the 1D Schrödinger equation for each slice along the confinement direction and its computational weight depends on the number of slices of the device. For example, for the simulations of the device

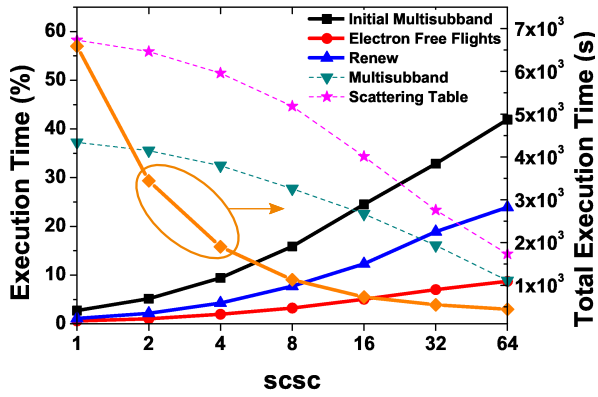


Figure A.10: Behaviour of the total execution time and computational load of the most expensive MSB–EMC subroutines versus the *scsc* variable. For *scsc*=1 the self–consistent solution of Schrödinger and Poisson equations is carried out in each step of the iteration time loop.

considered in this study, a 10 nm gate length DGSOI MOSFET, we have considered 72 grid points in the transport direction which implies the same number of Schrödinger equations have to be solved per call. This subroutine consumes a 37% of the total execution time. The *Scattering Table* subroutine, which consumes 58% of the total execution time, estimates the scattering rates and its computational weight depends on the number of scattering mechanisms implemented and the number of considered subbands.

As previously mentioned, a possible solution to reduce the execution time is to decrease the self–consistency between the solution of the Poisson and Schrödinger equations without jeopardizing the accuracy of the simulation. This strategy enable us to save computational time since it decreases the calls to the *Multisubband* and *Scattering Table* subroutines. The sum of their computational weight is reduced from 95% of the total

execution time when *scsc*=1 to 23% when *scsc*=64. As a consequence, there is also a reduction in the simulation time that is more important for large values of *scsc*. For instance, the total execution time when *scsc*=64 is 439.0 s, a value 93% lower than the one obtained when *scsc*=1. In figure A.10 the opposite behaviour is appreciated for the *Electron Free Flights* and *Renew* subroutines, responsibles for the electron free flights and for conserving the charge neutrality in the vicinity of the ohmic contacts respectively. The sum of their computational load increases from a poor 1.70% of the total execution time when *scsc*=1 to a 32.65% when *scsc*=64. Finally, the *Initial Multisubband* subroutine is just solved once since it provides a initial solution of 1D Schrödinger equation. Therefore, its execution time remains constant and independent of the *scsc* variable. This justifies the increase in its computational weight observed when the *scsc* variable is increased, which is proportional to the decrease in the total execution time.

Note that when the self–consistency is solved in every iteration, the total execution time required for a real simulation, i.e. a 30 ps stationary state, will be roughly 30 times higher than the time obtained from the simulation of a 1 ps stationary state. This means more than 2 days of simulation time. Attending to these time estimations it would be prohibitive to test new physical models without decreasing the frequency in which the 1D Schrödinger equation is solved self–consistently with the 2D Poisson equation. Moreover, for testing purposes, the accuracy of the simulation is not so decisive, and it is recommended to reduce the self–consistency to save execution time. However, there are situations which require the increase of the frequency in which self–consistency is performed in order to properly update the driving field value according to the electron density [SMQBD06]. This is the case for example, when more accurate simulations are needed to evaluate the physical properties of new devices, or when time resolution is critical such as in studies of transient states.

A.2.3 OpenMP parallelisation of the Multi–Subband Ensemble Monte Carlo simulator

The main goal of this study is to improve the simulation when using multicore architectures. For this purpose, the simulator has been parallelised using the OpenMP API. This paradigm has been selected due to the current development of multicore processors that can have up to 8 cores available in a single processor [Shi10], as for example the Intel Xeon L7555 processor. The use of these multicore processors can allow us to reduce up to 8 times maximum the total execution time.

From the profiling results of the code the most demanding subroutines were chosen to be parallelised. These subroutines are the ones presented in figure A.10. The block diagram, shown in figure A.11, depicts the parallelisation strategy where the coloured arrows represent the parallel threads. Among the initialisation subroutines of the MSB–EMC simulator, only the *Initial Multisubband* subroutine was parallelised. A parallel region was created just before the loop of evaluation of the iteration time and all the created threads evaluate this loop. This approach avoids to create and destroy the parallel regions in every time iteration. Almost all the subroutines that belong to the core of the MC simulator were parallelised with the exception of the less significant ones from the computational point of view, as for example the *Poisson* subroutine. All the created threads make a call to the parallelised subroutines, in such a way the parallel loops of these subroutines are distributed among them. Finally, the parallel region is destroyed at the end of the MC simulation when the iteration time is equal to the total simulation time.

If we compare the two block diagrams represented in Figures A.9 and A.11, we can see that in the last one the subroutine *Renew* is not included. In the parallel version of the code this subroutine has been merged with the *Electron Free Flights* one. This integration process and its computational advantages will be described in detail in Section A.2.4, because of its relevance in the development of MC simulators for semiconductor devices.

Figure A.12 shows the speed up of the parallel MSB–EMC simulator versus the number of cores. In this figure the influence of the *scsc* variable

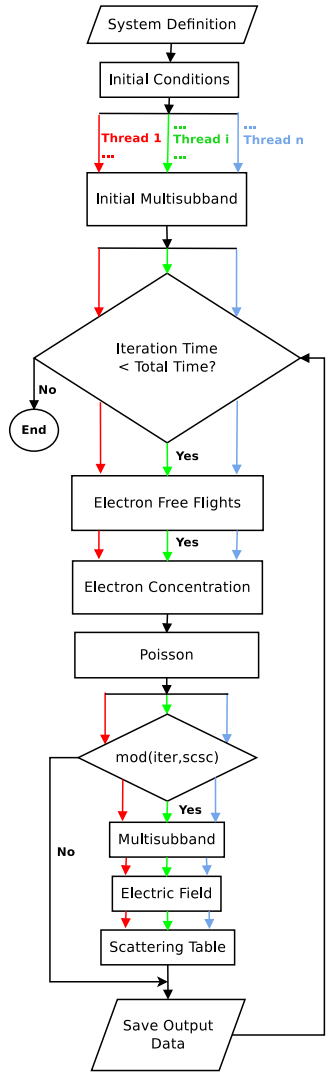


Figure A.11: Block diagram of the parallel version of the MSB–EMC simulator.

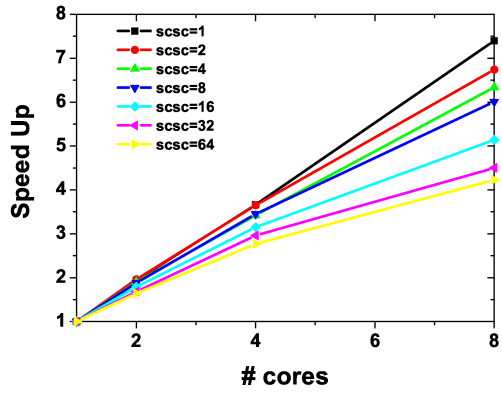


Figure A.12: Speed up of the parallel MSB–EMC simulator versus the number of cores. Several curves where depicted to show the influence of the self–consistent solution of Schrödinger and Poisson equations.

is also presented. These simulations were performed in a blade with two Quad Core processors Intel E5410 2.33 GHz that belongs to a PowerEdge M1000e cluster. Results show an increase in the speed up when the *scsc* variable is reduced. Therefore, the higher speed up is obtained when *scsc*=1. In this case, the value of the speed up for 8 cores is 7.40, very close to the theoretical maximum value equal of 8.

According to Amdahl's law (A.7) the speed up increases with the percentage of the code that is parallelised:

$$\text{Speed Up} = \frac{1}{(1-P) + \frac{P}{N}} \tag{A.7}$$

being P the parallel percentage of the code and N the number of cores. For instance, for *scsc*=1 the value of P is 99.85% and the theoretical value of the speed up is 7.92 using 8 cores. This value of the speed up is close to the experimental one, 7.4, seen in figure A.12 for that case.

The subroutines *Multisubband*, *Scattering Table* and *Initial Multisubband*, which have altogether more than 90% of the computational weight

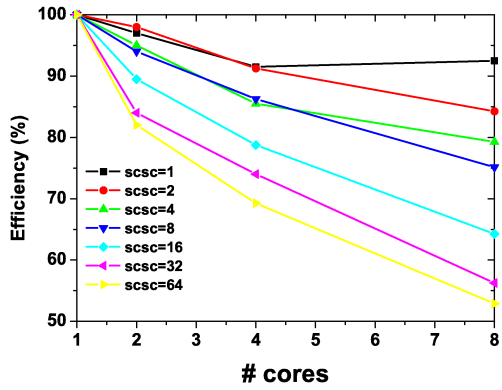


Figure A.13: Parallel efficiency of the parallel MSB–EMC simulator versus the number of cores and the *scsc* variable. The best value of the effiency is obtained when the self–consistent solution of Schrödinger and Poisson equations is solved for each iteration.

when *scsc*=1, are fully parallelised. However, the *Electron Free Flights* and *Renew* subroutines can not be parallelised completely because they have several dependencies. As seen in Section A.2.2, the computational weight of the MSB–EMC subroutines depends on the *scsc* variable. When the *scsc* variable increases the weight of the subroutines that are not completely parallelised also increases leading to a reduction in the speed up.

Figure A.13 shows the parallel efficiency, the relation between the speed up and the number of used cores, of the MSB–EMC simulator as a function of the number of cores and the frequency in which the Poisson and Schrödinger equations are solved self–consistently. A decrease of the parallel efficiency is observed when the number of cores is increased. The *scsc* variable has a big impact in the parallel efficiency. For *scsc* values higher than 32 the parallel efficiency drops to almost a 50% when 8 cores are used. For the same interval of *scsc* values the efficiency is 70% when 4 cores are used. Therefore, the efficient use of the computational resources leads to 4 or less cores. In the other hand, for *scsc* values lower than

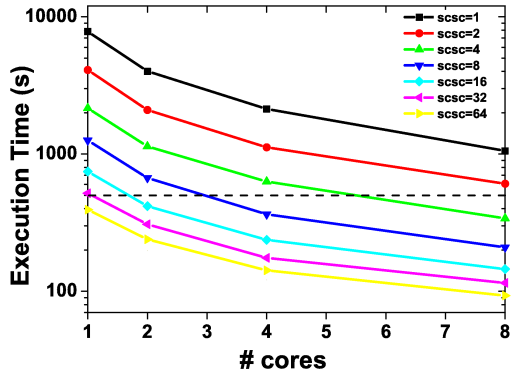


Figure A.14: Execution time of the parallel MSB–EMC simulator depending on the number of cores. The influence of the frequency in which the self–consistent solution of Schrödinger and Poisson equations is calculated (scsc variables) is also presented.

4 the parallel efficiency is higher than 80% when we are using 8 cores. These results indicate that the most accurate simulations, and hence the most demanding ones, make the most of the multicore processor without wasting available resources.

Another advantage of the use of parallel machines is the possibility of adjust the level of accuracy desired (through the *scsc* variable and depending on the simulation requests) without compromising the execution time. This can be seen in figure A.14, where the total execution time for a 1ps stationary state as a function of the number of cores and the *scsc* variable is shown. If we fixed an optimum simulation time in 500s (see dashed horizontal line), we can adjust the accuracy of the simulation by changing the number of cores employed. For instance, if 4 cores are being used we can afford to solve the Poisson–Schrödinger coupled system every 8 iterations. If we double the number of cores the self–consistent solution can be obtained every 4 iterations.

These results demonstrate the benefits of the parallelisation of the

MSB–EMC simulator. Thanks to the parallelisation it is not necessary to sacrifice the accuracy of the simulation to get the results in a reasonable time.

A.2.4 Optimisation of the 2D Multi–Subband Ensemble Monte Carlo code: integration of the *Renew* and *Electron Free Fligths* subroutines

The computational load of the *Renew* subroutine (see figure A.10) increases from 1.11% of the total execution time, when the Schrödinger and Poisson equations are solved self–consistently in every iteration, to 23.93% of the total execution time, when the equations are solved self–consistently every 64 iterations.

The initial version of the 2D MSB–EMC simulator was partially based on the Tomizawa's book [Tom93] that describes how to develop a Monte Carlo device simulator. In the presented approach, the subroutine *Electron Free Flights* simulates the carrier motion and marks for deletion the particles absorbed by the contacts. The index valley variable will be 8 or 9 if the particle is respectively absorbed by the source or by the drain. After this, the subroutine *Renew* evaluates the number of particles that are outside the device due to their absorption by the contacts. Therefore, all particles with the index valley property equal to 8 and 9 will be deleted. The subroutine *Renew* is also responsible for maintaining the charge neutrality in the vicinity of the source and drain ohmic contacts deleting or injecting the right number of particles. Figure A.15 presents a description in pseudocode of the initial version of the *Electron Free Flights* and *Renew* subroutines. Note that both subroutines need a loop that evaluates the properties of the particles.

In the optimisation that we propose, both subroutines, *Electron Free Flights* and *Renew*, are merged. In this way, only one loop that evaluates the properties of the particles is required. Using this proposal, the free flights of the electrons are carried out in the same way that in the unoptimised version. We also mark the index valley with the values 8 or 9 when the particles are absorbed by the contacts. In the initial version of

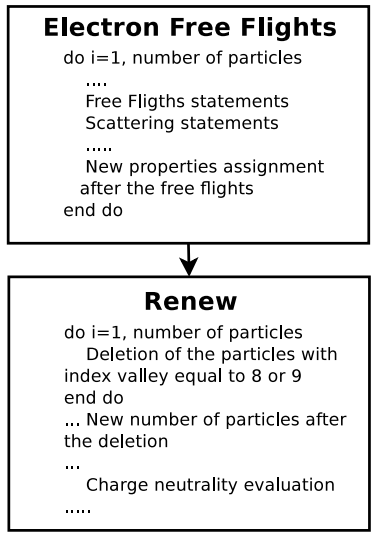


Figure A.15: Pseudocode of the initial version of the *Electron Free Flights* and *Renew* subroutines.

the code these particles were dismissed. However, in the optimised version these particles will be re–injected by the opposite contact that they were absorbed. These carriers are injected according to a Maxwellian distribution at the lattice temperature [GP93]. After that, the charge neutrality condition in the ohmic contacts will be evaluated. If the injected particle is not needed because its injection disobeys the charge neutrality condition it will be marked for deletion and deleted at the end of the subroutine. This process is done iteratively for each particle and its main advantage is that only one loop for all particles is necessary. Finally, at the end of the loop, the charge neutrality condition in the ohmic contacts has to be evaluated. If necessary, a right number of particles are injected to reach the charge neutrality condition. A simplified description of the optimised version of the *Electron Free Flights* subroutine in pseudocode is shown in figure A.16.

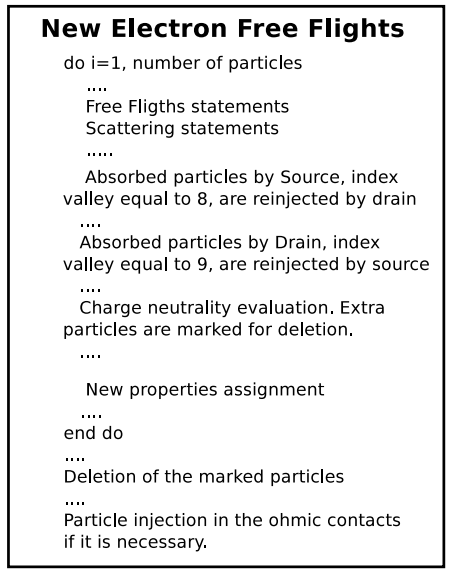


Figure A.16: Pseudocode of the optimised *Electron Free Flights* subroutine.

Figure A.17 shows the percentage of execution time versus the *scsc* variable for the initial version of *Renew* and *Electron Free Flights* subroutines. The addition of both subroutines (EFF+RE) and the new optimised version of the *Electron Free Flights* subroutine is also shown. Results were obtained from the sequential simulation of a 1 ps length stationary state. The computational weight of these subroutines depends on the *scsc* variable. Therefore, the benefits of the optimisation presented are also dependent on this variable. For instance, when *scsc*=1 our optimisation allow us to reduce just 1% of the total execution time, whereas if *scsc*=64 the total execution time is reduced by 20%.

Finally, it is important to highlight the relevance of these optimisation results since the *Renew* and *Electron Free Flights* subroutines are widely used in different kind of MC semiconductor device simulators.

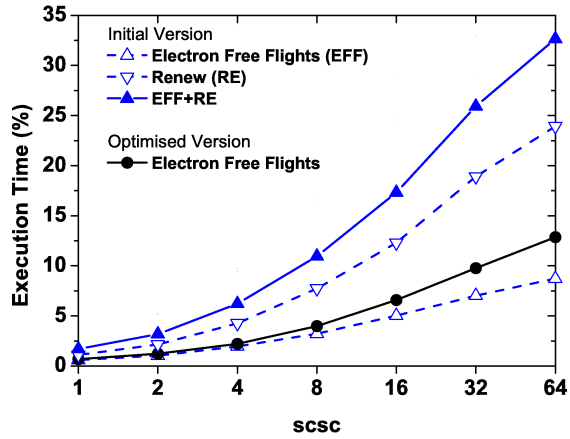


Figure A.17: Computational weight depending on the *scsc* variable for the initial version of *Renew* and *Electron Free Flights* subroutines. The addition of both subroutines (EFF+RE) and the new optimised version of the *Electron Free Flights* subroutine are included.

A.3 SMNanoS: submitting and monitoring nanoelectronic simulations in the es–NGI

The recent creation of the European Grid Initiative (EGI), where Grid infrastructure of each country will be run by National Grid Initiatives (NGI), has boosted the number of available resources of the national Grid infrastructures. In Spain, the Spanish National Grid Initiative (es–NGI) [ngi] is supported by the Spanish e–Science Network [eci] and, the computer applications of each area of knowledge have their own virtual organisation and their own resources provided by the resource centres.

The MOSFET virtual organisation (VO–MOSFET), that belongs to es–NGI, was created in 2009 to perform semiconductor device simulations using this infrastructure. It has more than 4500 nodes and users submit their jobs using the gLite middleware [GLi]. Therefore, thanks

to this infrastructure it is possible to run a large number of simulations allowing to study the influence of design variations and statistical simulations [HAB+07, FFL05, TTS+06].

Despite the important increase of computational resources, this infrastructure is not very used by researchers of VO–MOSFET due to several drawbacks. For example, the job submission and monitoring processes using the gLite user interface are not straightforward for users, and this problem increases when users have to monitor and recover the output data for tens or hundreds of simulations. In this section, we present a submission and monitoring tool to facilitate the use of the VO–MOSFET. Furthermore, we evaluate the impact of the heterogeneous resources of the es–NGI infrastructure on the execution time of the MV-ECBE-EMC simulator.

A.3.1 VO–MOSFET infrastructure

The Spanish National Grid Initiative, that belongs to the EGI, was developed to deploy a national Grid infrastructure for Spanish researchers. Figure A.18 shows a global description of this infrastructure, where two different levels are appreciated. Common areas or applications, placed in the second level of the figure (NGI VOs), are gathered in the VOs of the infrastructure where each VO has its own resources provided by the resource centres. Usually, these providers are supercomputers or research centres and, the available resources depend on the VO requirements. In the first level are located the NGI central services, being the main ones:

- Information Service: This is an information database which provides the state of the resource to the resource broker.
- Resource Broker: This service is a central service responsible of submitting jobs to the resource centres.
- Virtual Organization Membership Service (VOMS): This service stores the information about VOs belonging to es–NGI infrastructure.

- Storage: This service, which is distributed between the resource centres, is responsible for job file storage. This service must be available for all VOs.

- File Catalog: This service indicates where the files are stored using a file location information database.

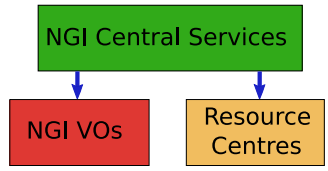


Figure A.18: Global description of NGI infrastructure.

The most important services that guarantee the smooth running of the infrastructure, such as the information service, VOMS and file catalog, must be replicated to minimize possible local failures. Moreover, this infrastructure relies on monitoring and accounting services where the state of infrastructure and consumed resources can be monitored by users.

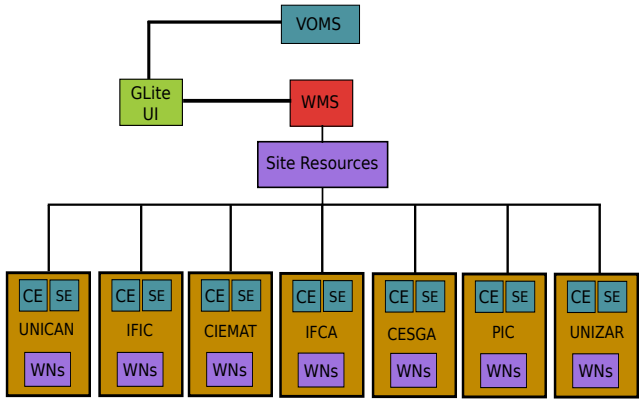


Figure A.19: Global VO–MOSFET support infrastructure.

The VO–MOSFET is a virtual organisation developed to provide computational resources for nanoelectronic simulations. This VO belongs to the es–NGI and a simplified scheme of the support infrastructure is depicted in figure A.19. Currently, the VO–MOSFET uses the gLite middleware and it is compatible with other Grid initiatives like EGEE [EGE], EELA [eel] or I2G [i2g]. If a VO user wants to submit a job from a gLite User Interface (UI), he has to authenticate to VOMS server and, after that, he can submit the job to the Workload Management System (WMS). Finally, the WMS will distribute the jobs between the worker nodes (WN) of the resource centres. The VO users have a virtual machine available that is configured as a user interface. Furthermore, they can also submit their jobs using a server configured as a user interface that is located in the Supercomputing Center of Galicia (CESGA).

# Cores	Resource Center
1378	PIC
1616	IFCA
848	IFIC
284	CIEMAT
148	UNICAN
340	CESGA
22	UNIZAR
Total #Cores	4636

Table A.3: VO–MOSFET resources supplied by resource centres. These data were obtained from the "lcg–infosites" command.

Table A.3 shows the number of cores and the name of the resource centres that support the VO MOSFET. The number of cores of each resource centre was obtained from the *lcg–infosites* command, and indicates the total number of cores.

A.3.2 Description of the job submission and monitoring application

The job submission system of the VO–MOSFET is based on the command line middleware gLite–UI. This middleware requires to know the necessary commands to submit and monitor the jobs. An interesting characteristic of the VO–MOSFET is the possibility to execute parametric jobs, changing the design parameters of a semiconductor device. Tens or hundreds of simulations are necessary in some cases to perform these studies. Therefore, with this number of jobs, the job submission and monitoring tasks may be a complicated issue. The first solution to facilitate these processes was to develop a Python [Pyt] application that could submit and evaluate automatically the state of the jobs.

This Python application that we have named SMNanoS (Submitting and Monitoring Nanoelectronic Simulations) uses the gLite–UI commands to submit and obtain the job status.

The diagram depicted in figure A.20 shows the three main levels of the SMNanoS application. When a user invokes the program from the command line, allows the submission of a single job or a job collection to the VO–MOSFET resources. Users only have to indicate the jdl directory, if they want to submit a job collection, or the jdl file if they want to submit a single job. In this first level of functionality, a python dictionary associates the identifier of the submitted job with its jdl file. In the second level of functionality, the job status is evaluated for a fixed period of time that can be changed by the user depending on the total execution time.

FigureA.21 shows the possible job status and the cases where the job may be resubmitted. The status *Waiting* is the first possible resubmission case shown in the figure. The status *Waiting* is usually assigned when the computational resources are not available. If the waiting time is longer than the half of the estimated execution time the job is cancelled and resubmitted. The number of resubmissions is limited to five for all cases, after that the user is advised to check the job. If the job status is *Cancelled* or *Aborted*, it will be automatically resubmitted. Finally, if the job status is *Done (failed)* it will be resubmitted if the simulation output is not

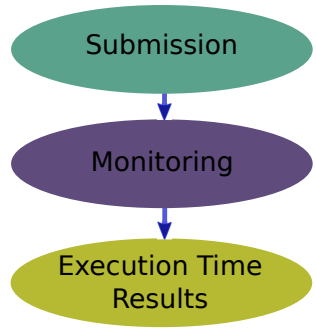


Figure A.20: Functionality levels of the SMNanoS.

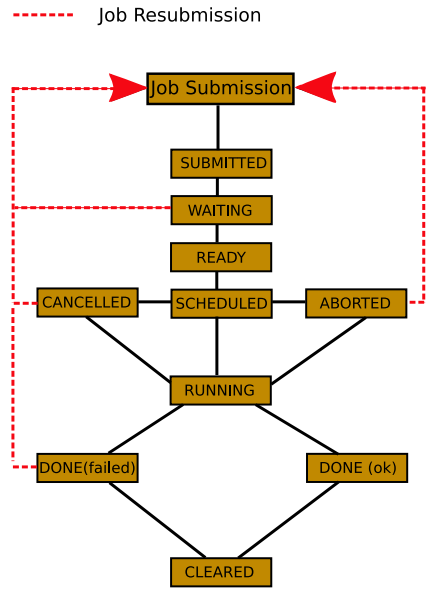


Figure A.21: VO–MOSFET possible job status and resubmission cases.

stored in the Storage Element (SE). When a job has to be resubmitted, its identifier is removed from the python dictionary and its associated jdl file is resubmitted again. Therefore, the new identifier that belongs to the

resubmission is stored in the Python dictionary with its associated jdl.

In the last level of functionality, the SMNanoS application downloads the output files of each simulation from the SE and depicts a figure with the execution time of each job. Previously, the submitted job has been configured to store the output directory of the simulation in the SE. This directory contains the simulation data, and moreover, information about the name, CPU model of the WN and the execution time of the job.

A.3.3 Testing the VO–MOSFET resource centres

The SMNanoS application has been used to test the VO–MOSFET resource centres. We have prepared a job collection with five jobs, where each job simulates a 2D DGSOI MOSFET with a stationary state of 2 ps length [RMN$^+$09, VGLA$^+$09]. We have submitted this collection to each one of the resource centres that supports the VO-MOSFET, with the exception of Unizar, and we compare the differences between the available resources.

The obtained results, with the values of the execution time and the WN description for the job collection executed in each available resource centre, are shown in Tables A.4 to A.9. Furthermore, figure A.22 depicts the execution time of each job of the collection for each resource centre following the same order as in Table A.3. These results demonstrate the differences between the worker nodes where, the largest difference for the execution time (between the fastest and slowest WN) is roughly a factor 1.4. As expected, these differences are logical due to the heterogeneity of the Grid infrastructure, where different characteristics such as hardware, system load or configuration of the worker nodes can lead to different results. However, considering the fastest resource centre and the execution time of a job collection equal to the time of the slowest job, see figure A.23, the highest difference in the execution times between resource centres is almost 55%. This important difference is compensated for the huge amount of computational resources that are not available in local clusters.

# JobId	Time (s)	CPU (GHz)	WN Name
1	5998	Xeon L5420 2.50	td160.pic.es
2	3727	Xeon L5530 2.40	td485.pic.es
3	3817	Xeon L5530 2.40	td484.pic.es
4	3750	Xeon L5530 2.40	td487.pic.es
5	3767	Xeon L5530 2.40	td472.pic.es

Table A.4: Execution time of each job that belongs to the collection submitted to the PIC resource centre. The job number (# JobId), the execution time (Time), the processor type (CPU) and the name of the worker node (WN name) are shown in the table. All these worker nodes have the same architecture x86_64.

# JobId	Time (s)	WN Name
1	8334	cms01wn
2	8252	cms01wn
3	8162	cms01wn
4	7257	cms02wn
5	8240	cms01wn

Table A.5: Execution time of each job that belongs to the collection submitted to the IFCA resource centre. The job number (# JobId), the execution time (Time) and the name of the worker node (WN name) are shown in the table. All these worker nodes have the same architecture x86_64 and processor model Xeon E5345 2.33 GHz.

# JobId	Time (s)	WN Name
1	6290	wn304.ific.uv.es
2	6198	wn304.ific.uv.es
3	5781	wn305.ific.uv.es
4	6210	wn304.ific.uv.es
5	6248	wn304.ific.uv.es

Table A.6: Execution time of each job that belongs to the collection submitted to the IFIC resource centre. The job number (# JobId), the execution time (Time) and the name of the worker node (WN name) are shown in the table. All these worker nodes have the same architecture x86_64 and processor model Xeon E5420 2.50 GHz.

# JobId	Time (s)	WN Name
1	9071	ciewn037.ciemat.es
2	8930	ciewn087.ciemat.es
3	9232	ciewn036.ciemat.es
4	9045	ciewn035.ciemat.es
5	9231	ciewn036.ciemat.es

Table A.7: Execution time of each job that belongs to the collection submitted to the CIEMAT resource centre. The job number (# JobId), the execution time (Time) and the name of the worker node (WN name) are shown in the table. All these worker nodes have the same architecture i686 and processor model AMD Opteron 270 2.00 GHz.

# JobId	Time (s)	WN Name
1	8974	wn007.macc.unican.es
2	8969	wn004.macc.unican.es
3	8875	wn006.macc.unican.es
4	8936	wn004.macc.unican.es
5	8995	wn006.macc.unican.es

Table A.8: Execution time of each job that belongs to the collection submitted to the UNICAN resource centre. The job number (# JobId), the execution time (Time) and the name of the worker node (WN name) are shown in the table. All these worker nodes have the same architecture x86_64 and processor model PentiumD 3.00 GHz.

# JobId	Time (s)	WN Name
1	8243	compute–2–11
2	8145	compute–1–1
3	8091	compute–2–11
4	8252	compute–0–11
5	8348	compute–1–10

Table A.9: Execution time of each job that belongs to the collection submitted to the CESGA resource centre. The job number (# JobId), the execution time (Time) and the name of the worker node (WN name) are shown in the table. All these worker nodes have the same architecture i686 and processor model Pentium4 3.20 GHz.

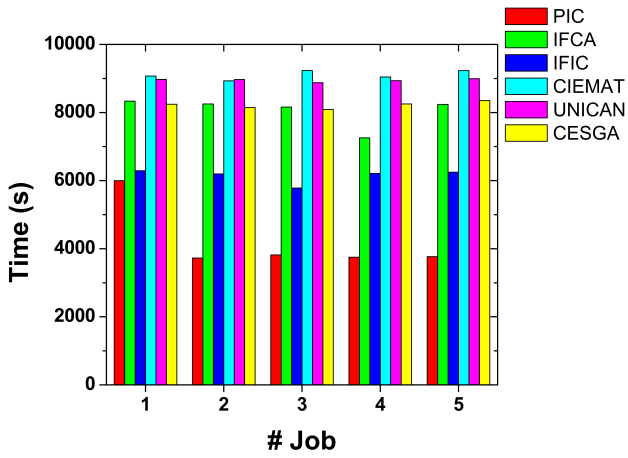


Figure A.22: Execution time of each job that belongs to the submitted collection for each resource center.

A.4 Using Grid infrastructures for a stationary DGSOI Monte Carlo simulation

Thanks to the large amount of available resources of Grid infrastructures, we can decrease the execution time of several stationary simulations distributing the stationary states among these resources. In nanoelectronics it is usual to simulate different stationary states under different bias points, ignoring the simulation results of the transients.

In this study we have distributed 5 stationary states of a MV–ECBE–EMC simulation among the computational resources of the Cesga Virtual Organisation (that belongs to the Galician Supercomputing Centre) in order to evaluate the advantages of this strategy. The benchmark device is a 10 nm channel length DGSOI transistor. The silicon thickness is 6 nm, the effective oxide thickness is 10 Å and the metal gate contacts have a work–function value of 4.6 eV. The drain and source doping concentration is N_D = 9 x 10^{19} cm^{-3}, whereas the channel concentration is $N_A = 10^{15}$ cm^{-3}.

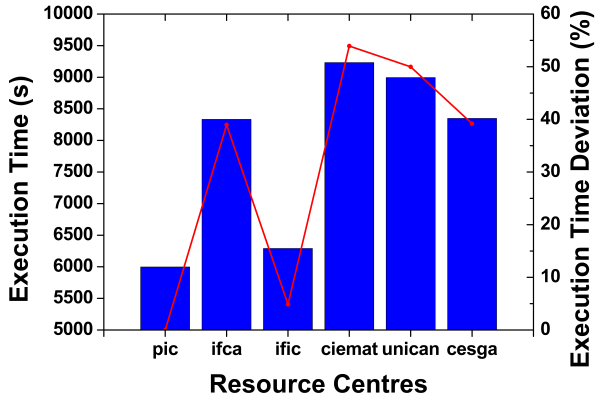


Figure A.23: Execution time of each job collection submitted to each resource centre. The execution time of a job collection is equal to the slowest execution time of its jobs. Furthermore, this figure also shows the additional percentage of required execution time for the resource centres compared to the centre with the shortest execution time.

A.4.1 Splitting the transient simulation

The main goal is to distribute the stationary states of the simulation among the worker nodes of the Grid. Therefore, the ideal achieved speed up will be equal to the number of stationary states.

Figure A.24 shows a simulation with 5 stationary states with different drain voltages. Stationary states of the total transient simulation (rhombus) are 2 ps long. The ramp time to switch between different values of drain voltage is 0.1 ps. The voltage in the gates is kept constant at 1 V. Triangles show the equivalent split simulations. Each one of split simulations starts with the drain voltage equal to 0 V and reaches the drain voltage of the stationary state 0.1 ps later.

The study that we present is limited to stationary states only, because the transient dynamic response of the simulations depend upon the initial conditions of carriers [JL89].

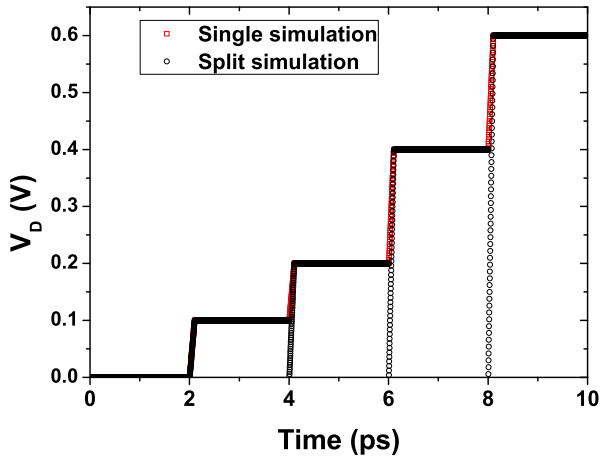


Figure A.24: Transient simulation of the DGSOI device for a gate voltage of 1 V. Drain bias point is changed each 2 ps.

A.4.2 Results

The objective of this work is to reduce the execution time of stationary Monte Carlo simulations. However, we must prove that results have not been altered when we execute the split simulation. We executed the split simulations and the single simulation in Cesga Virtual Organisation which belongs to the EGI Grid infrastructure. This virtual organisation of the Grid has several clusters with different processors and the workload management system (WMS) distributes jobs between free worker nodes.

Table A.10 shows the execution time for the split simulations in comparison with the full single transient ones. The execution time of the split simulations is almost a fifth of the execution time of the 10 ps single simulation with respect to the worst execution time in this case.

The black line in figure A.25 describes the behaviour of the time evolution of the drain current for the split simulations. The behaviour of the stationary states for split simulations is the same as the one observed

# Split	Time (s)	CPU (GHz)	Ram (MB)
1	6159	Core2 Duo 2.66	512
2	10293	Xeon 1.60	512
3	6447	Core2 Duo 2.66	512
4	6560	Core2 Duo 2.66	512
5	6189	Core2 Duo 2.66	1024
Single	50655	Xeon 1.60	512

Table A.10: Execution time for different Grid worker nodes. The number of the split simulation (# Split), the execution time (Time), the processor type and frequency (CPU), the size of the main memory (RAM) are presented.

in a single simulation.The behaviour of the drain current versus the V_{DS} voltage is very similar for a single and split simulation as showed in figure A.26. Therefore, stationary simulation results for split simulations are as valid as single simulation ones, as is proven in Figures A.25 and A.26.

Figure A.25 compares the time evolution of the drain current for the single and split simulations. Note the excellent agreement between them. The validity of the split simulations is also proved in figure A.26, where the drain current versus the drain bias for both the single and split simulations is shown.

Figure A.27 shows the standard deviation of the mean current versus V_{DS} voltage for single and split simulations when we use two different stationary states, one of 2 ps and the other of 3 ps. This figure shows that the error with 2 ps stationary states is higher than with 3 ps. Figure A.27 also shows that the error for split simulations is higher than for single ones. For split simulations, the error for 3 ps stationary states is lower than for 2 ps stationary states. This behaviour may be caused by the initial bias point since the simulation of each stationary state begins at $V_{DS} = 0$ V and the ramp time is constant for all simulations. This may cause an increase in the error because the initial conditions of the stationary states

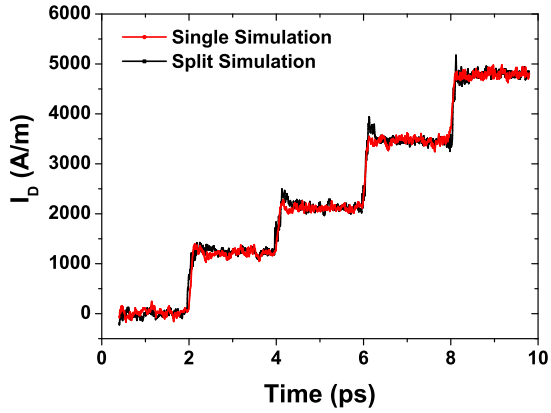


Figure A.25: Time evolution of the drain current for a single and a split simulation.

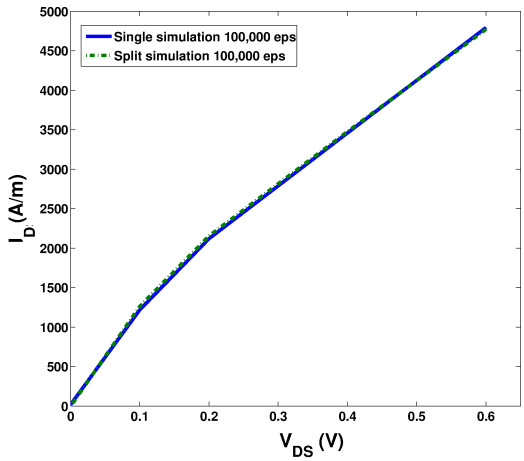


Figure A.26: Drain current versus V_{DS} voltage for a single and split simulation.

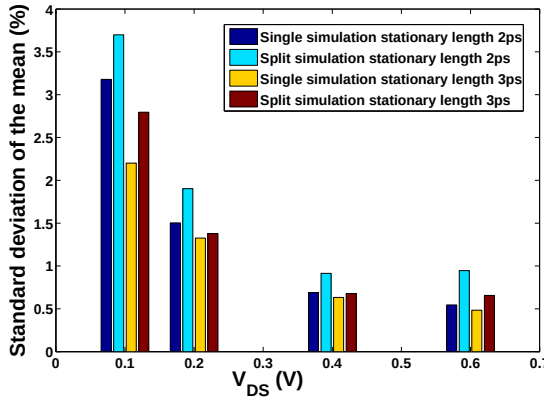


Figure A.27: Standard deviation of the mean current versus V_{DS} voltage for a single and split simulation.

for split simulations are different from those of the single simulation, and the length of stationary state may need to be increased. Figure A.27 also shows that the standard deviation of the mean is between 0.5 % and 3.5 %. These values are very low and it is possible to reduce the standard deviation of the mean current by increasing the length of the stationary states.

A.5 2D Monte Carlo Simulation of gate misalignment in a 10 nm gate length DGSOI MOSFET

State–of–the–art electronic devices have reached the decananometer range and conventional planar bulk MOSFETs will not meet these requirements because of severe Short Channel Effects (SCEs). Non–conventional MOSFETs, such as multi–gate transistors or FinFETs, are being extensively investigated as serious candidates to become standard devices for next generation integrated circuits. For such small devices, quantum mechanical (QM) effects must be considered in numerical simulations because they deeply influence their electrical characteristics [AI89, Anc90, Dat00, TR01, WR03, GF04, QNSM$^+$09, ARA08]. Classical simulations such as drift–diffusion or classical Monte Carlo are not suitable to investigate these devices.

Double Gate (DG) MOSFETs are one of the most promising candidates to substitute standard bulk MOSFETs in next generation devices continuing scaling the chanel length to a few nanometers. The unique features of these devices lead to a better electrostatic control to avoid SCEs and to an improvement of the saturation drain current. Furthermore, these devices can take advantage of volume inversion benefits [CC03, ITR10]. However, one of the fabrication problems arises from the gate misalignment which can be of special importance in planar DG Silicon–On–Insulator (DGSOI) and independent gate devices. The DG MOSFET misalignment has been widely studied by several authors [WFTS94, AZPC01, SMC03, KA08, YC05, WLV$^+$05], since its impact increases as the gate length is scaled down. Both drift–diffusion simulations [WFTS94, AZPC01, SMC03] and experimental results [YC05, WLV$^+$05] are presented in such studies. It is shown that a gate misalignment around 20 – 25% can be assumed for channel lengths in the range of 50 – 100 nm, achieving an increase in the general performance when the gate is shifted towards the source contact.

As the devices are aggressively scaled down, it is necessary to include in

the simulations quantum effects to accurately describe the impact of gate misalignment in Ultra–Short and Ultra–Thin–Body devices. In this section we carry out a thorough study by using a quantum–corrected Monte Carlo simulator for three different configurations of a 10 nm gate length misaligned device: a) the top gate misalignment (TGM), b) both gates (top and bottom) misalignment in opposite directions (BGOD), and c) both gates misalignment in the same direction (BGSD).

A.5.1 Description of gate misaligned devices

This study has been carried out considering a 10 nm gate length (L_g) DGSOI MOSFET. The silicon thickness is 6 nm, the effective oxide thickness (EOT) is 1 nm and the work–function of the metal gate contacts is 4.6 eV. The drain and source have an abrupt doping profile and the concentration is $N_D = 5 \times 10^{19}$ cm^{-3}. Doping density in the channel is $N_A = 10^{15}$ cm^{-3}.

The misalignment measurement is described by the relation $100 \times L_m/L_g$ (%) where L_m is the displacement length with respect to the self–aligned position, and L_g is the gate length (see figure A.28). A positive value of this magnitude implies a gate shift towards the drain whereas source directed gate displacements are represented by negative values. The three different configurations of the gate misaligned devices considered in this work are shown in figure A.28. The aligned structure is shown in figure A.28(a). Figure A.28(b) depicts the top gate misalignment (TGM), where the top gate is misaligned from -50% to 50%. In the next configuration figure A.28(c), both gates are misaligned in opposite directions (BGOD). In this case, the top gate is always shifted towards the source and the bottom gate is always misaligned towards the drain from 0% to $\pm 50\%$. The $\pm$ symbol highlights that the bottom (top) gate is displaced towards the drain (source). Finally, the last configuration is depicted in figure A.28(d), where both gates are misaligned in the same direction (BGSD) from -50% to 50%.

Two different simulations for each configuration (TGM, BGOD, BGSD) have been carried out. For the first one, the voltage of both gates was kept

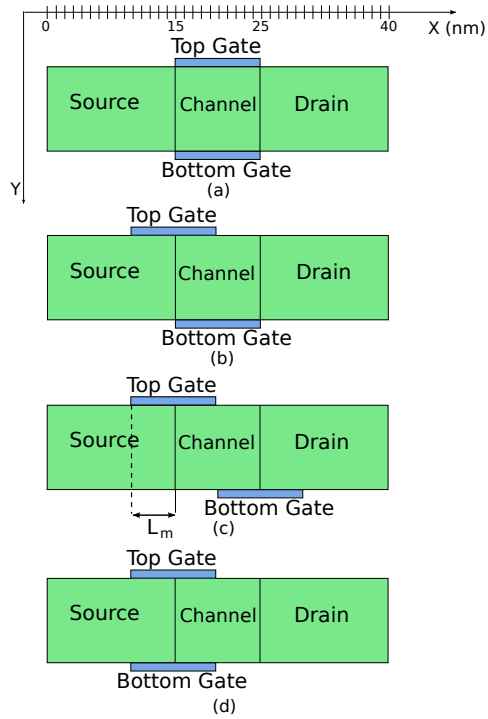


Figure A.28: Different configurations of the studied device, the channel length is 10 nm. (a) Aligned structure, (b) Top gate misalignment configuration (TGM), (c) Both gates misaligned in opposite directions (BGOD) and (d) Both gate misaligned in the same direction (BGSD).

constant at 1 V while the drain voltage was switched from 0 to 1 V for each case. In the second case, the drain voltage was kept constant at 100 mV and both gate voltages were switched from 0 to 1 V in 50 mV steps.

A.5.2 TGM. Top gate misalignment

In the TGM configuration (figure A.28(b)), the top gate has been misaligned from -50% to 50%. Figure A.29 sketches the drain current versus the drain voltage and the corresponding top gate misaligned posi-

tion for $V_G = 1$ V. The results produce a drain current maximum for a -10% misalignment, showing a better performance than the aligned device. However, the drain current decreases quickly with a dropping higher than 5% when the top gate misalignment is higher than 20% towards the drain contact for $V_D = 1$ V.

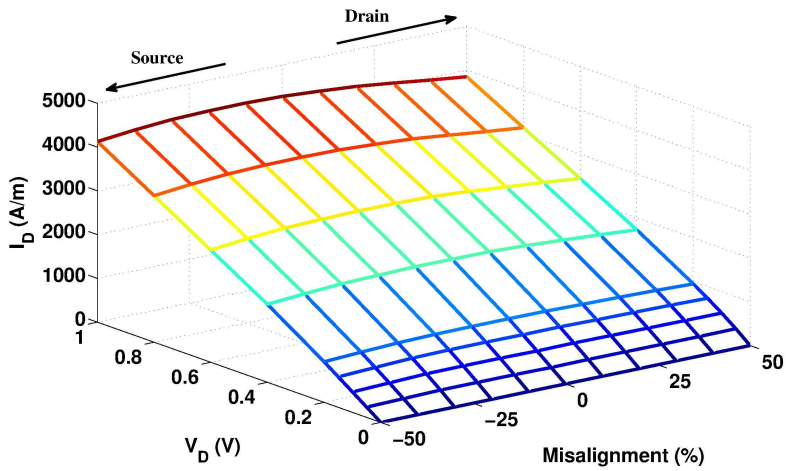


Figure A.29: Drain current versus drain voltage for each gate position of the TGM configuration. Gate voltages were kept constant at 1 V.

These results agree with other studies [WFTS94, AZPC01, SMC03, KA08, YC05] reporting a drain current increase when the gate is misaligned towards the source. Such behaviour can be explained due to the reduction of the source serial resistance because of the charge accumulation when the gate is misaligned towards the source [AZPC01].

Figure A.30 compares the electron concentration profile along the confinement direction at the virtual source ($x \simeq 17$ nm), it can be observed an increase of the charge peak as the top gate is shifted towards the source. Nevertheless, the electron concentration for the fixed bottom gate decreases when the top gate is misaligned towards the drain more than 10%, causing the drain current decrease shown in figure A.29. This behaviour is due to the loss of the electrostatic control when the gate is misaligned

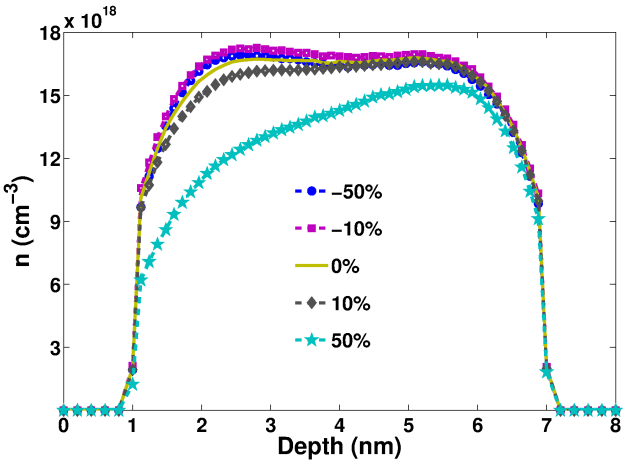


Figure A.30: Channel electron concentration profile at X = 17 nm for each top gate position of the TGM configuration when $V_D = V_G = 1$ V.

towards the drain and because of the increase of the barrier height, such as is shown in figure A.31. In this figure a barrier lowering can be observed for the misaligned configuration corresponding to the maximum enhancement of drain current (−10%), compared to the results obtained for the extreme configurations −50% and 50%.

I_D vs V_G curves for each top gate position have also been obtained considering a drain voltage of 100 mV. The results, sketched in figure A.32 (a), show drain current deviations higher than 7% when the misalignment is higher than 30% and lower than −20%. These results agree with those obtained, in Fig A.29 and A.30 for the drain current and the electron concentration respectively, where tolerable values of the misalignment are similar. Figure A.32 (b) shows the current deviation with respect to the reference device for different applied gate voltages. In all the cases, a similar behaviour is observed corresponding the maximum values of the drain current deviation for misalignment values lower than −30% and greater than 30%.

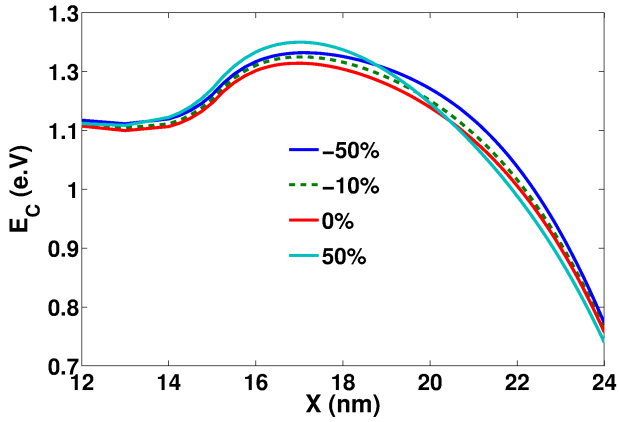


Figure A.31: Conduction band from source to drain at 2.2 nm of the top gate position (Y direction) for the TGM configuration when $V_D = V_G = 1$ V.

A.5.3 BGOD. Both gates misalignment in opposite directions

We have performed a similar study moving both gates in opposite directions. The top gate is shifted towards the source and the bottom gate towards the drain as shown in figure A.28(c).

The drain current versus the drain voltage for each gate position is depicted in figure A.33 when $V_G = 1$ V. As can be observed, I_D always decreases when both gates are missaligned. A ±20% misalignment causes about a 2% drain current deviation with respect to the drain current value of the aligned device and a drain current deviation higher than 5% is obtained when the misalignment is higher than ±30%.

The conduction band from source to drain at the point of maximum carrier concentration for the reference device is shown in figure A.34, for four different values of misalignment at $V_D = V_G = 1$ V. This figure shows that the lowest energy barrier is obtained for the aligned case, which explains the drain current decrease in figure A.33 when the misalignment increase.

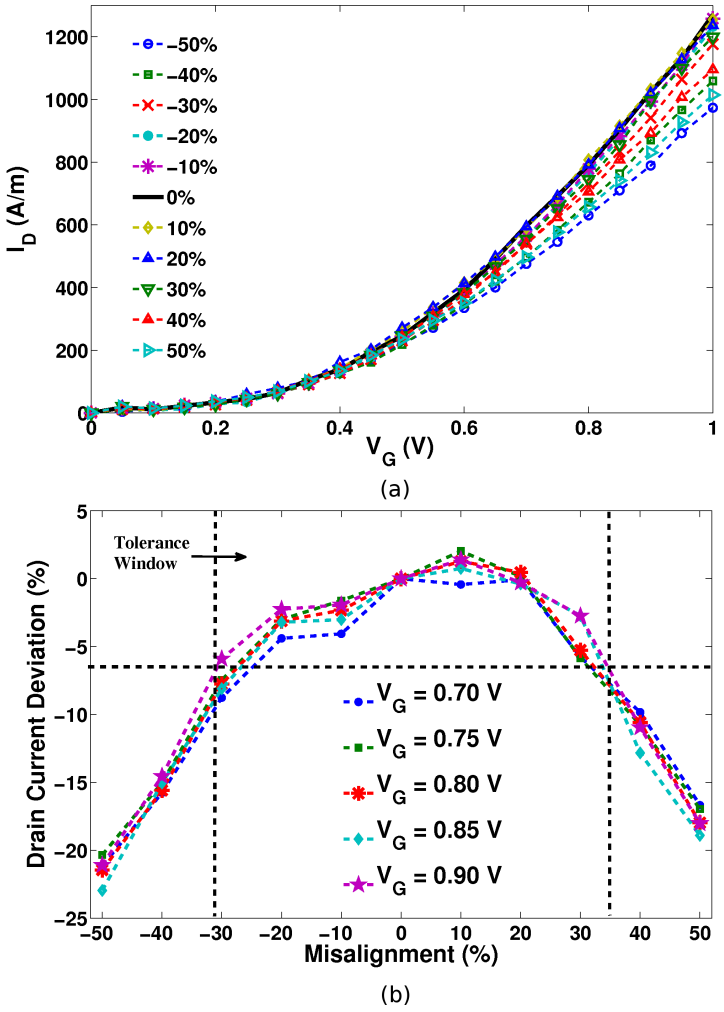


Figure A.32: (a) Drain current versus gate voltage for each gate position of the TGM configuration. (b) Behaviour of the relative drain current deviation from the aligned position for the TGM configuration and different gate voltages. For all cases V_D was kept constant at 100 mV.

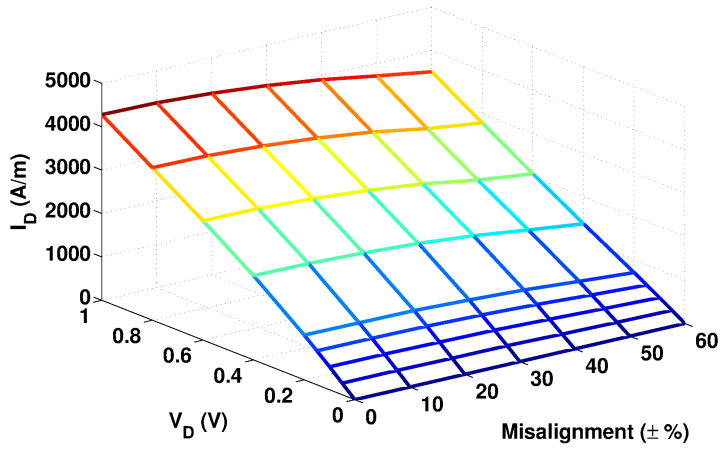


Figure A.33: Drain current versus drain voltage for each gate position of the BGOD configuration, for a gate equal to 1 V.

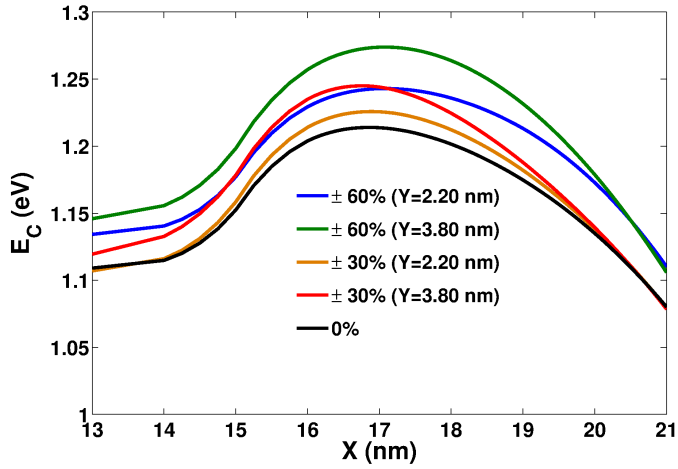


Figure A.34: Conduction bands from source to drain at Y = 2.2 and 3.8 nm for the BGOD configuration at $V_D = V_G = 1$ V.

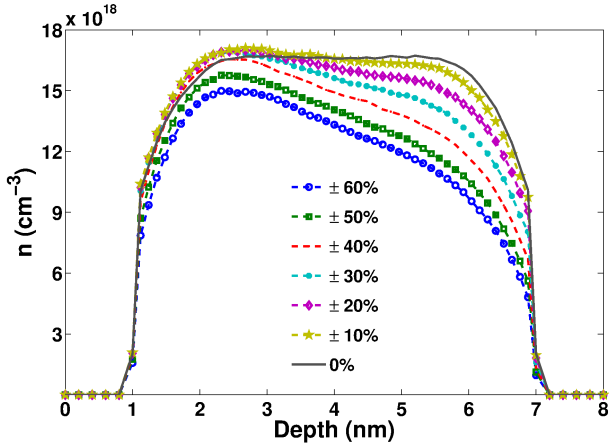


Figure A.35: Channel electron concentration at X = 17 nm for each gate possition of BGOD configuration at $V_D = V_G = 1$ V.

Figure A.35 represents the electron concentration at the maximum of the energy barrier (X $\simeq$ 17nm) depicted in figure A.34 when $V_D = V_G = 1$ V. The behaviour of the electron concentration is similar here to that shown for the TGM case but with combined effects. As the top gate is shifted to the source contact the electron concentration increases in the top channel. However, the bottom gate is shifted towards the drain and, therefore, there is a simultaneous charge decrease in the bottom channel. As a consequence, the electrostatic control is weaker in this case since the bottom gate also moves. The effects on the drain current are presented in figure A.36 (a). For high inversion charge values, the impact of misalignment is more important than in the TGM case. As a consequence, for misalignment values higher than 20% output characteristics are degraded in a critical way, as is seen in figure A.36 (b). In the case of the subthreshold regime, a higher drain current, which degrades the I_{on}/I_{off} ratio, can be observed mainly due to the spreading of charge in the whole silicon slab and the decrease of surface roughness scattering in the bottom interface near the virtual source.

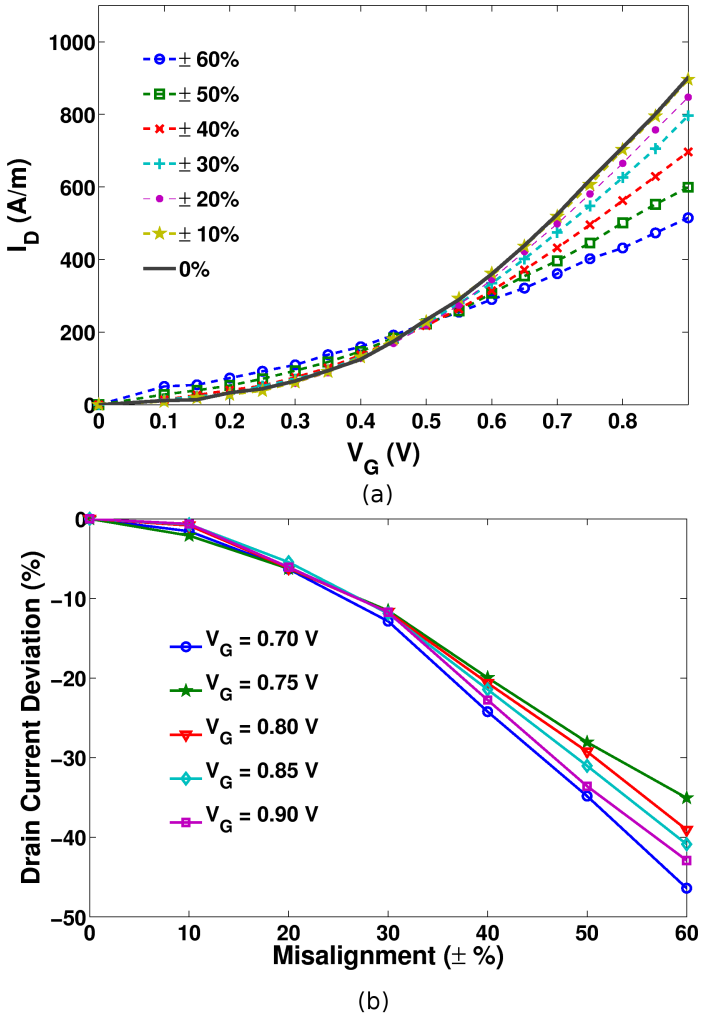


Figure A.36: (a) Drain current versus gate voltage for each gate position of the BGOD configuration. (b) Behaviour of the relative drain current deviation from the aligned position for the BGOD configuration and different gate voltages. For all cases V_D was kept constant at 100 mV.

A.5.4 BGSD. Both gates misalignment in the same direction

As it has been already commented in previous sections, device performance is degraded when one gate is shifted towards the drain. In this section we will discuss the behaviour of the misaligned device in the case of both gates shifted to the same direction. Figure A.37 shows I_D vs V_D for each gate position when $V_G = 1$ V. In this case, the drain current drops quickly when both gates are moved towards the drain, with a reduction higher than 5% of the reference drain current when the misalignment is higher than 10%. However, if both gates are misaligned towards the source, the drain current is higher than the obtained with the aligned configuration. This increase drops quickly when the misalignment of both gates towards the source is higher than -40%.

The electron concentration is depicted in figure A.38 at the same position as in previous sections, X = 17 nm, and for the same bias point, $V_D = V_G = 1$ V, for five gate positions of the BGSD configuration. It

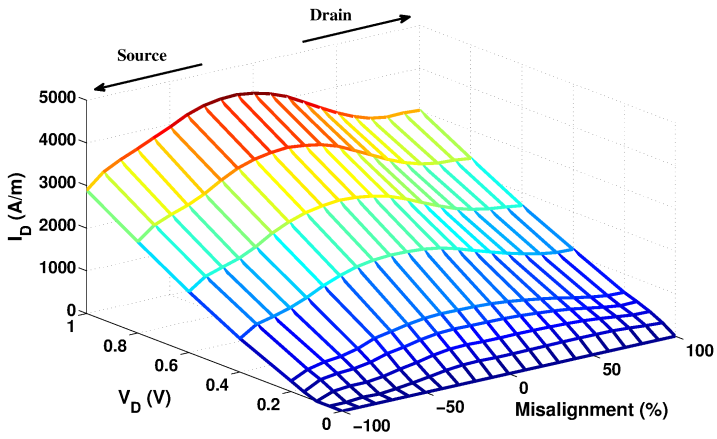


Figure A.37: Drain current versus drain voltage for each gate position of the BGSD configuration and $V_G = 1$ V.

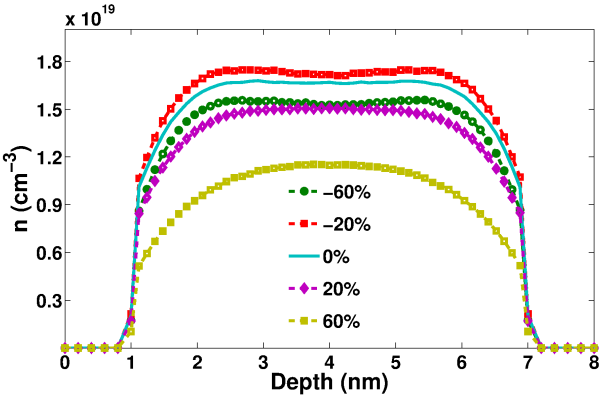


Figure A.38: Electron concentration in the channel of the device at X = 17 nm for each gate position of the BGSD configuration and $V_D = V_G = 1$ V.

shows that, a misalignment in both gates produces a uniform distribution of the electron concentration along the channel. The electron concentration is higher when both gates are misaligned towards the source and decreases sharply when the misaligned is towards the drain. This reduction in the electron concentration increases the source series resistance causing the quick drop of the drain current observed in figure A.37.

Finally, we have sketched the drain current versus gate voltage for each gate position when the drain voltage was kept constant at 100 mV in figure A.39 (a). The tolerance window for current variations is shown in figure A.39 (b), presenting affordable values for misalignment in the range of $[-15\%, 18\%]$.

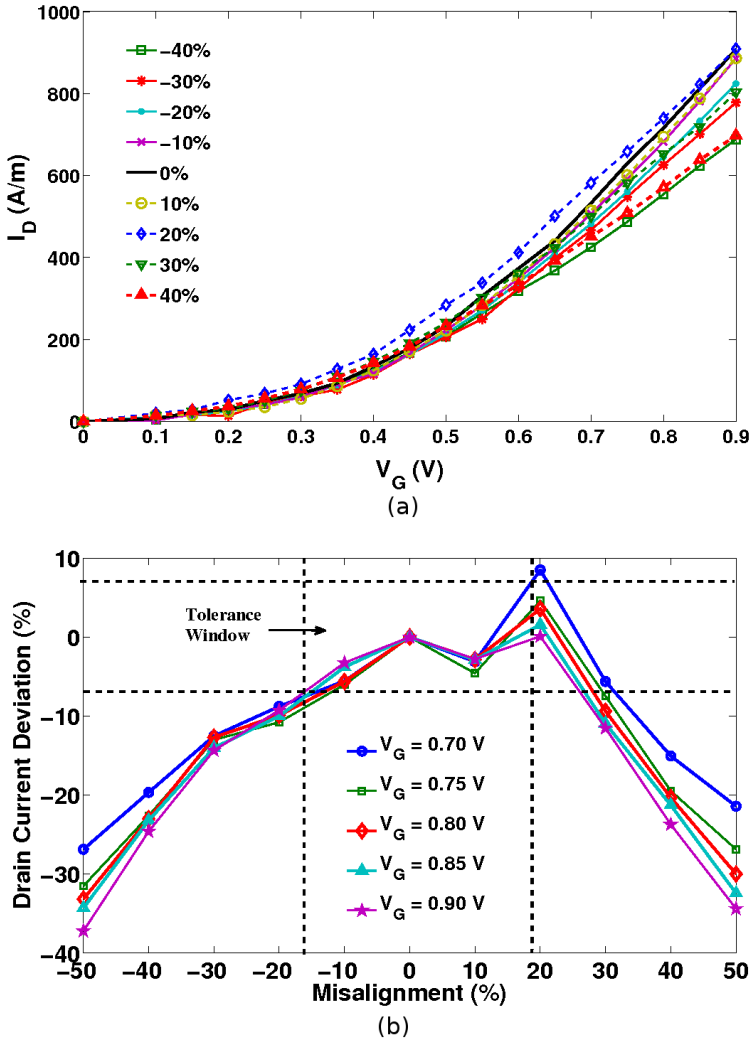


Figure A.39: (a) Drain current versus gate voltage for each gate position of the BGSD configuration. (b) Behaviour of the relative drain current deviation from the aligned position for the BGSD configuration and different gate voltages. For all cases V_D was kept constant at 100 mV.

A.6 Study of the oxide thickness variability induced by a rough HfO_2/SiO_2 interface on SGSOI MOSFETs

An important source of variability for MOSFET transistors is the oxide thickness fluctuation (OTF) [ABD+03, MRA10]. SiO_2 has been the most common material used as gate dielectric for MOSFET transistors until some years ago. However, its replacement by new materials could be the origin of new sources of variability for these devices. For instance, an avoidable irregular interface of SiO_2 with variable thickness is created during the deposition process of HfO_2 dielectric on silicon substrates when $Hf(NO_3)_4$ is employed [ZDZ+06]. The origin of this SiO_2 layer has been attributed to different sources such as, oxidation of the substrate surface during the time that it is being heated in the reactor, post–deposition oxidation upon exposure to the atmosphere or direct oxidation of the silicon by the $Hf(NO_3)_4$.

In this work we have studied 100 devices with different rough interfaces of HfO_2/SiO_2 using the MSB–EMC simulator, in order to analyse the impact of local oxide thickness fluctuations.

A.6.1 Description of the simulated devices

The simulated device is a 32 nm gate length Single Gate Silicon on Insulator (SGSOI) MOSFET, such as is shown in figure A.40. The length of the source and drain regions is 60 nm and they are doped with a uniform doping profile $N_D = 5 \text{ x } 10^{19} cm^3$. The channel thickness is equal to 6 nm and is undoped. The gate dielectric has two oxide layers. The top oxide, the nearest to the metal gate, is a HfO_2 layer with a thickness equal to 4.12 nm. The bottom oxide, the nearest to the channel, is a SiO_2 layer with a thickness equal to 0.7 nm. The value of the equivalent oxide thickness (EOT) is equal to 1.4 nm.

The roughness model has been obtained from a Line Edge Roughness (LER), as shown on figure A.41, following the model employed to study the

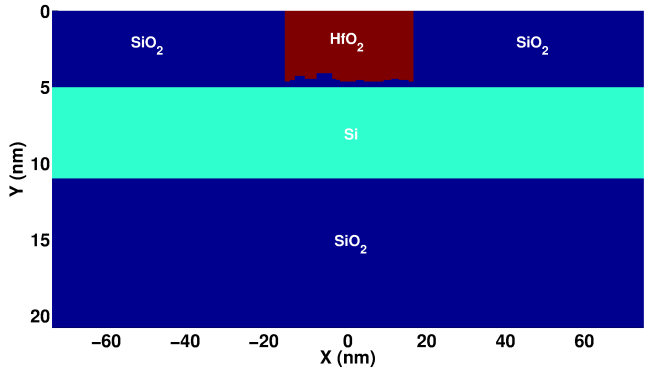


Figure A.40: Representation of the SGSOI MOSFET structure. This structure shows an example of a random fluctuation in the HfO_2/SiO_2 interface.

effect of the SiO_2/Si interface roughness [GFW+85, AKB03b]. This figure shows a random H(x) function with arbitrary units and different values for each node of the longitudinal mesh that belongs to the HfO_2/SiO_2 interface. This rough interface has local variations of the oxide thickness in the range ±3.5 Å. The value of the local variation of oxide thickness is calculated as a function of the H(x) height, where positive values of the function indicate a SiO_2 penetration into the HfO_2 and negative values indicate a penetration of the HfO_2 into the SiO_2.

A.6.2 Simulation results

We have simulated 100 devices with different rough interfaces between gate oxides. Furthermore, we have simulated three devices without roughness in order to analyse the impact of the roughness in the interface between oxides. The first device without roughness (uniform case), has a 4.12 nm thick layer of HfO_2 and a 0.7 nm thick layer of SiO_2. For the other two devices (limit cases), the interface of the SiO_2 oxide layer penetrates 3.5 Å into the HfO_2 for one of them and for the other one, the HfO_2

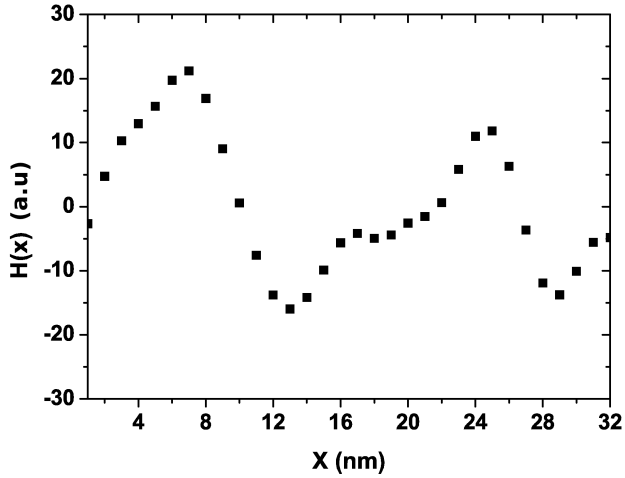


Figure A.41: Example of the statistical generation of the oxide interface.

interface penetrates 3.5 Å into the SiO_2. Therefore, the thickness of the HfO_2 layer for these devices is 3.77 nm y 4.47 nm respectively and, 1.05 nm y 0.35 nm for the SiO_2 layer.

Figure A.42 represents $I_D - V_D$ curves for each simulated device. The drain current fluctuations due to the roughness of the interface between oxides regarding the limit cases and the uniform case are shown. The distribution of the drain current for the devices with rough interfaces between the gate oxides is shown in figure A.43. The drain and gate bias are 0.9 and 1 V respectively. Furthermore, this figure shows the drain current values of the uniform case, the limit cases, and the mean of the distribution. The obtained mean is equal to 1166 A/m and standard deviation is slightly higher than 2%. For this case the relative difference between the mean of the distribution and the uniform case is around 1.2%. A comparison of the I_D for the uniform case, the mean and the standard deviation is shown in Table A.11, for $V_D = 0.5$ and 0.9 V. The values of the standard

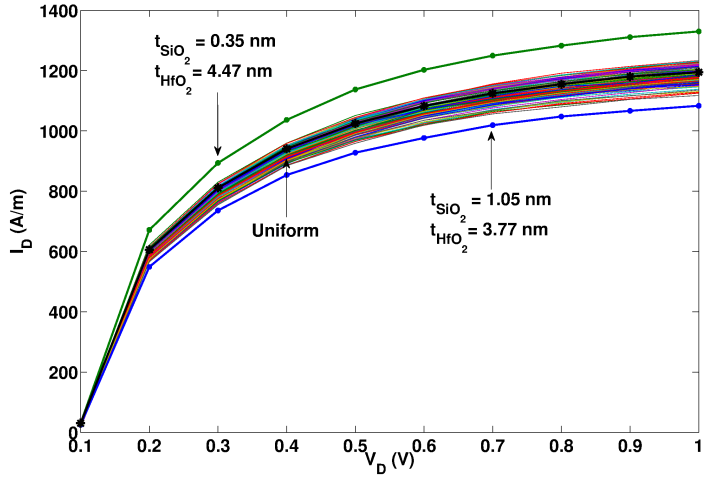


Figure A.42: I_D–V_D curves for a SGSOI with $V_G = 1$ V and with different configurations of the oxide interface. The limiting cases of the SiO_2 and HfO_2 thicknesses and, the curve for a uniform device are shown for comparison.

	Uniform I_D (A/m)	Mean I_D (A/m)	$\sigma(I_D)$ (%)
V_D=0.5 V	811.3	794.7	2.13
V_D=0.9 V	1180	1166	2.24

Table A.11: Comparison of the I_D for the uniform case, the mean value of the drain current distribution and the standard deviation for $V_D = 0.5$ and 0.9 V. V_G was kept constant to 1 V.

deviation are higher than 2% for both drain voltages. However, the relative difference between the mean value of the drain current distribution and the uniform case increases when the drain bias is reduced.

Figure A.44 shows the I_D–V_G fluctuations when $V_D = 1$ V and drain current fluctuation regarding the uniform case.

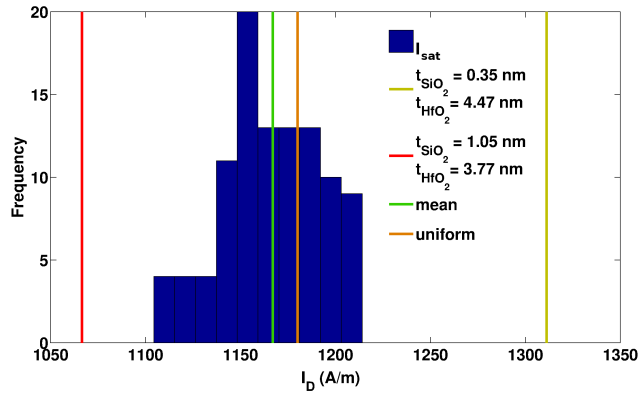


Figure A.43: Distribution of the on–current obtained from the I_D–V_D figure for $V_D = 0.9$ V and $V_G = 1$ V.

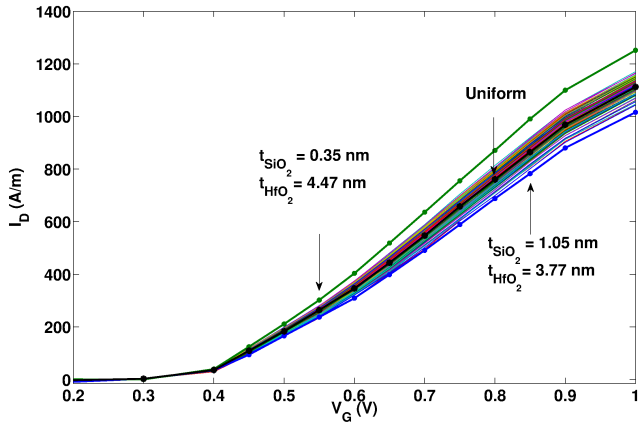


Figure A.44: I_D–V_G curves for a SGSOI with $V_D = 1$ V and with different configurations of the oxide interface. The limiting cases of the SiO_2 and HfO_2 thicknesses and, the curve for a uniform device are shown for comparison.

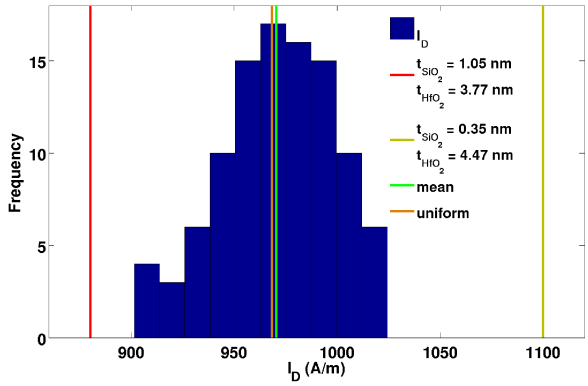


Figure A.45: Distribution of the drain current obtained from the I_D–V_G figure for $V_G = 0.9$ V and $V_D = 1$ V.

The distribution of the drain current for the devices with rough interfaces between the gate oxides is shown in figure A.45, for $V_G = 0.9$ V and $V_D = 1$ V. Furthermore, this figure shows the drain current values of the uniform, the limit cases and the mean of the distribution. The obtained value of the mean is equal to 970.5 A/m and the standard deviation is around 3%. For this case the relative difference between the mean of the distribution and the uniform case is insignificant (around 0.21%). A comparison of the I_D values for the uniform case, the mean and the standard deviation is shown in Table A.12, for $V_G = 0.5$ and 0.9 V. In this case the standard deviation for sub–threshold region (4.21%) is higher than the obtained value for $V_G = 0.9$ V equal to 2.87%. However, the relative difference between the mean value of the drain current and the uniform case is insignificant (around 1%).

	Uniform I_D (A/m)	Mean I_D (A/m)	$\sigma(I_D)$ (%)
$V_G = 0.5$ V	183.5	183.9	4.21
$V_G = 0.9$ V	968.4	970.5	2.87

Table A.12: Comparison of the I_D for the uniform case and the I_D mean, and I_D standard deviation for $V_G = 0.5$ and 0.9 V. V_D was kept constant to 1 V.

Conclusions

The simulation of semiconductor devices is a fundamental design tool in the nanoelectronics research field, since it makes possible the study of new devices, design characteristics and new materials.

However, due to the aggressive scaling of devices more complex simulation models are required and therefore, longer computational times. The reduction of the device dimensions also increases the impact of variability effects in their performance. To account for these effects statistical analysis are mandatory, which greatly increase the computational cost.

From the computational point of view simulation of semiconductor devices has two challenges: (i) decreasing the execution time without limiting the complexity of models and (ii) increasing the available computational resources to perform studies of statistical fluctuations.

This work proposes two solutions to these challenges, applying them to 2D Monte Carlo simulation of MOSFET transistors: (i) parallelisation of Monte Carlo simulators in order to decrease the execution time and (ii) the usage of Grid infrastructures to increase the computational resources available.

The main achievements and conclusions of these proposed solutions are commented below:

Parallel Programming Applied to Simulation of Semiconductor Devices

We have observed a strong development of multicore architectures during the last 5 years. For instance, a 90% of the supercomputers which belong to the TOP500 list have multicore architectures. An important advantage of these processors is that they are incorporated into desktop computers and therefore, we can take advantage of parallelism during the development of the simulators in our personal computers.

In this work, we have parallelised two different Monte Carlo simulators using the OpenMP standard. The first one, called Multi–Valley Effective Conduction Band Edge Ensemble Monte Carlo (MV–ECBE–EMC), is the lest computational demanding code. Despite OpenMP is a simple parallel language, the parallelisation process of this simulator was a complex task due to the fact that this code was not design with parallelism in mind. Finally, the parallel version of the MV–ECBE–EMC simulator has reduced the execution time by a factor 1.81 using 4 cores. The scalability limits have been achieved with more than 4 cores due to the influence of the percentage of sequential code that can not be parallelised. This fact highlights the importance of profiling to know the structure of the simulator and the most computational demanding subroutines of the code.

The second parallel simulator is the Multi–Subband Ensemble Monte Carlo (MSB–EMC). This simulator is more computational demanding than the MV–ECBE–EMC but it has a larger parallel percentage of the code. The execution time of the parallel version of MSB–EMC has decreased by a factor 7.4 using 8 cores and by a factor 10 with 16 cores.

Grid Computing Applied to the Simulation of Semiconductor Devices

Grid technologies have evolved during the last 15 years boosting the development of new National and International Grid Infrastructures, such as es–NGI, EGI or TeraGrid. In this study, we used the es–NGI infrastructure because of the large amount of available resources, 5000 cores and

500 TB in 2010, since they could be very useful for variability studies of semiconductor devices. Despite these resources, in our opinion, the Grid middleware needs to simplify the usage of Grid infrastructures. We have employed the gLite middleware to perform our tests and we have developed a submitting and monitoring system to manage failed jobs automatically.

The heterogeneity of computational resources is an important problem of Grid infrastructures. For instance, the submitted jobs to es–NGI could run on last generation Xeon processors or older PentiumD processors. Our tests confirm the effect of these heterogeneous resources on the execution time of Monte Carlo simulations, with up to a 55% of difference among the execution times depending on the computational resource. Despite these differences, the use of this kind of infrastructures are very useful to perform statistical studies of semiconductor devices, since the large amount of multicore processors are available can be employed to decrease the execution times.

Finally, we would like to emphasize the importance of developing new services to exploit the advantages of these infrastructures for scientific simulations and data processing.

To end up this dissertation we present a study of the impact of two sources of variability on the characteristics of SOI MOSFETs. Therefore, we study (i) the effect of gate misalignment on DGSOI MOSFETs using the parallelised MV–ECBE–EMC simulator and (ii) the impact of the oxide thickness variability induced by a rough HfO_2/SiO_2 interface on the dielectric of SGSOI MOSFETs using the parallel MSB–EMC simulator. The main achievements and conclusions of these studies are commented below.

Simulation of gate misalignment on DGSOI MOSFETs using the MV–ECBE–EMC simulator

A 10 nm channel length double gate SOI MOSFET has been chosen to carry out this study. Three different configurations of gate misalignment have been studied: (i) TGM, when the top gate is misaligned, (ii) BGOD, when both gates are misaligned in opposite directions, and (iii) BGSD,

when both gates are misaligned in the same direction.

Obtained results for all the considered configurations show that, for a gate misalignment up to 20% the drain current deviations are smaller than 10% with respect to the ideal perfectly aligned device. The three studied device configurations confirm that a gate misalignment towards the source lower than 20% is better than towards the drain (especially for the saturation region), achieving in some cases a better performance than in the aligned device. This behaviour is due to the increase in the electron concentration in the channel region close to source which decreases the source serial resistance.

In the BGSD configuration a higher performance is achieved when both gates are misaligned towards the source whereas the channel control is reduced when the gate is misaligned towards drain. This situation is specially relevant when the devices are fabricated using self–alignment techniques which is the usual case for FinFETs.

Study of the oxide thickness variability induced by a rough HfO_2/-SiO_2 interface on the dielectric of SGSOI MOSFETs using the MSB–EMC simulator

This work analyses the impact of local oxide thickness fluctuations due to a HfO_2/SiO_2 rough interface. We have simulated 100 SGSOI devices with different rough interfaces using the parallel MSB–EMC simulator. Simulation results of studied devices show that a rough interface of the gate oxide modifies the SGSOI MOSFET characteristics. The standard deviation of the drain current, obtained for the 100 simulated rough devices, is around 2 and 4% depending on the bias point of transistors.

Future Work

In this dissertation we have benefited from parallelism and Grid architectures to improve the Monte Carlo simulation of MOSFET transistors. From this research area, new lines have arisen for future development regarding the application of parallel computing techniques and Grid and

Cloud technologies to the simulation of semiconductor devices. Furthermore, this possible applications could also be useful in other research areas. Several examples of possible future works are listed bellow:

- The parallel 2D MSB–EMC simulator could be extended to 3 dimensions. Currently, this simulator solves a 1D Schrödinger equation for the perpendicular plane to transport direction. A 3D parallel simulator could be implemented with a 2D solution of this equation for the same plane. The 3D simulator will enables us to simulate three-dimensional devices such as, Nano–wires or FinFETs. Furthermore, the future 3D version and new implemented methods can also be parallelised and optimised taking advantage of the experience obtained from this work.

- Simulation of the impact of other sources of statistical variability on MOSFETs characteristics. For instance, an interesting work could be to extend the study of the impact of local fluctuations on oxide thickness due to a HfO_2/SiO_2 rough interface. This work could compare the current results (obtained from a direct method) with a new ones obtained from a indirect method. This indirect method could be a scattering mechanism which takes into account the effects of oxide thickness fluctuations. Furthermore, a comparison with experimental results will be also very interesting because we can properly model the HfO_2/SiO_2 interfaces. Other studies related to variations of the design parameters of MOSFETs transistors are also possible such as, variability of silicon thickness or gate length.

- Application of parallel computing techniques to the simulation of semiconductor devices. Our group has developed several 3D parallel simulators parallelised with MPI, using domain decomposition methods. A very interesting computational work taking advantage of the OpenMP experience could be the hybrid parallelisation OpenMP/MPI of these simulators. Moreover, it would be interesting to assess the advantages of using hardware accelerators such as GPUs.

- Finally, the most recent research line for future work is focused on e–Science. Simulation of semiconductor devices and other research areas require Web portals for submitting and monitoring jobs. These web portals must provide access to different resources, i.e. clusters, Grid or Cloud, and they must include Web services to enable researchers to compare data from different sources, such as experimental and simulation data.

Acknowledgements

This work has been carried out under the Spanish projects TEC2010-17320 and TIN2007-67537-C03-01, the Xunta de Galicia projects DX-IDI09TIC001CT, INCITE08PXIB206094PR and the Royal Society International Joint Project 2009/R2. Furthermore, I would like to thank the grants of EUROSOI network and the HPC-EUROPA2 project (project number: 228398) with the support of the European Commission – Capacities Area – Research Infrastructures to stay in Granada and Edinburgh respectively.

Bibliography

[ABD+03] A. Asenov, A.R. Brown, J.H. Davies, S. Kaya, and G. Slavcheva. Simulation of intrinsic parameter fluctuations in decananometer and nanometer–scale MOSFETs. *Electron Devices, IEEE Transactions on*, 50(9):1837–1852, sept. 2003.

[AI89] M. G. Ancona and G. J. Iafrate. Quantum correction to the equation of state of an electron gas in a semiconductor. *Phys. Rev. B*, 39(13):9536–9540, May 1989.

[AKB03a] A Asenov, S Kaya, and A R Brown. Intrinsic parameter fluctuations in decananometer MOSFETs introduced by gate line edge roughness. *IEEE Trans. Electron Devices*, 50(5):1254–1260, May 2003.

[AKB03b] A. Asenov, S. Kaya, and A.R. Brown. Intrinsic parameter fluctuations in decananometer MOSFETs introduced by gate line edge roughness. *Electron Devices, IEEE Transactions on*, 50(5):1254–1260, may 2003.

[ALD96] N.R. Aluru, K.H. Law, and R.W. Dutton. Simulation of the hydrodynamic device model on distributed memory parallel computers. *Computer–Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 15(9):1029–1047, September 1996.

[AM76] N.W. Ashcroft and N.D. Mermin. *Solid State Physics*. Saunders College, Philadelphia, 1976.

[Amd67] Gene M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18–20, 1967, spring joint computer conference*, AFIPS '67 (Spring), pages 483–485, New York, NY, USA, 1967. ACM.

[Anc90] M Ancona. Density–gradient theory analysis of electron distributions in heterostructures. *Superlattices and Microstructures*, 7(2):119–130, 1990.

[ARA08] C. Alexander, G. Roy, and A. Asenov. Random–Dopant–Induced Drain Current Variation in Nano–MOSFETs: A Three–Dimensional Self–Consistent Monte Carlo Simulation Study Using *Ab Initio* Ionized Impurity Scattering. *IEEE Transactions on Electron Devices*, 55(11):3251–3258, 2008.

[Ase98] A. Asenov. Random dopant induced threshold voltage lowering and fluctuations in sub–0.1 μm MOSFET's: A 3–D "atomistic" simulation study. *Electron Devices, IEEE Transactions on*, 45(12):2505–2513, December 1998.

[AZPC01] F. Allibert, A. Zaslavsky, J. Pretet, and S. Cristoloveanu. Double–Gate MOSFETs: Is Gate Alignment Mandatory? In *Solid–State Device Research Conference, 2001. Proceeding of the 31st European*, pages 267–270, September 2001.

[BAY+03] F.M. Bufler, Y. Asahi, H. Yoshimura, C. Zechner, A. Schenk, and W. Fichtner. Monte Carlo simulation and measurement of nanoscale n–MOSFETs. *Electron Devices, IEEE Transactions on*, 50(2):418–424, 2003.

[BCB+87] F. Balestra, S. Cristoloveanu, M. Benachir, J. Brini, and T. Elewa. Double–gate silicon–on–insulator transistor with volume inversion: A new device with greatly enhanced performance. *IEEE Elec. Dev. Lett.*, 8(9):410–412, September 1987.

[Ben11] Benbakhti, B. and Kalna, K. and Chan, K. and Towie, E. and Hellings, G. and Eneman, G. and De Meyer, K. and Meuris, M. and Asenov, A. Design and analysis of the $In_{0.53}Ga_{0.47}As$ implant–free quantum–well device structure. *Microelectronic Engineering*, 88(4):358–361, 2011.

[BGS+66] N. P. Buslenko, D.I. Golenko, Y.A. Shreider, I.M. Sobol, and V.G. Sragovich. *The Monte Carlo method: The Method of Statistical Trials*. Pergamon, 1966.

[BOC02] S.G. Badcock, A.G. O'Neill, and E.G. Chester. Device and circuit performance of SiGe/Si MOSFETs. *Solid–State Electronics*, 46(11):1925–1932, 2002.

[BOI] BOINC. http://boinc.berkeley.edu/.

[Bro51] H Brooks. Scattering by ionized impurities in semiconductors. *Phys. Rev.*, 83:879, 1951.

[Bro10] Nathan Brookwood. AMD Fusion Family of APUs: Enabling a Superior, Immersive PC Experience. AMD White Paper, 2010.

[BWD84] G. Baccarani, M.R. Wordeman, and R.H. Dennard. Generalized scaling theory and its application to a 1/4 microme-

ter MOSFET design. *Electron Devices, IEEE Transactions on*, 31(4):452–462, April 1984.

[CBD+05] Robert Chau, Justin Brask, Suman Datta, Gilbert Dewey, Mark Doczy, Brian Doyle, Jack Kavalieros, Ben Jin, Matthew Metz, Amlan Majumdar, and Marko Radosavljevic. Application of high–κ gate dielectrics and metal gate electrodes to enable silicon and non–silicon logic nanotechnology. *Microelectronic Engineering*, 80:1–6, 2005.

[CC03] G. K. Celler and Sorin Cristoloveanu. Frontiers of silicon–on–insulator. *Journal of Applied Physics*, 93(9):4955, April 2003.

[CCW05] Hsien-Chin Chiu, Yi-Chyun Chiang, and Chan-Shin Wu. High breakdown voltage $(Al_{0.3}Ga_{0.7})_{0.5}In_{0.5}P$/InGaAs quasi–enhancement–mode pHEMT with field–plate technology. *Electron Device Letters, IEEE*, 26(10):701–703, oct. 2005.

[CGP07] Lei Chai, Qi Gao, and Dhabaleswar K Panda. Understanding the Impact of Multi–Core Architecture in Cluster Computing: A Case Study with Intel Dual–Core System. *Cluster Computing and the Grid, IEEE International Symposium on*, 0:471–478, 2007.

[Col04] J. P. Colinge. *Silicon–On–Insulator Technology: Materials to VLSI*. Kluwer Academic Press, 3rd edition, 2004.

[Col07] J.P. Colinge. Multi–gate SOI MOSFETs. *Microelectronic Engineering*, 84(9–10):2071–2076, 2007. INFOS 2007.

[Con] Condor. http://www.cs.wisc.edu/condor/.

[CS94] Thomas L. Casavant and Mukesh Singhal. *Readings in Distributed Computing Systems*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1994.

[Dat00] Supriyo Datta. Nanoscale device modeling: the Green's function method. *Superlattices and Microstructures*, 28(4):253–278, 2000.

[Dat07] Suman Datta. III–V field–effect transistors for low power digital logic applications. *Microelectronic Engineering*, 84(9–10):2133–2137, 2007. INFOS 2007.

[DGKY72] R.H. Dennard, F.H. Gaensslen, L. Kuhn, and H.N. Yu. Design of micron MOS switching devices. In *Electron Devices Meeting, 1972 International*, 1972.

[DGR+74] R. H. Dennard, F. H. Gaensslen, V. L. Rideout, E. Bassous, and A. R. LeBlanc. Design of ion–implanted MOSFET's with very small physical dimensions. *Solid–State Circuits, IEEE Journal of*, 9(5), oct 1974.

[Dis] Distributed.net. http://www.distributed.net/.

[DRA+07] R. Droopad, K. Rajagopalan, J. Abrokwah, L. Adams, N. England, D. Uebelhoer, P. Fejes, P. Zurcher, and M. Passlack. Development of GaAs–based MOSFET using molecular beam epitaxy. *Journal of Crystal Growth*, 301–302:139–144, 2007.

[eci] http://www.e–ciencia.es/.

[eel] http://www.eu–eela.eu/.

[EGE] EGEE. http://www.eu–egee.org/.

[egi] http://www.egi.eu/.

[Eng] Grid Engine. http://gridengine.sunsource.net/.

[FAV00] D.K. Ferry, R. Akis, and D. Vasileska. Quantum effects in MOSFETs: use of an effective potential in 3D Monte Carlo

simulation of ultra–short channel devices. In *Electron Devices Meeting, 2000. IEDM Technical Digest. International*, 2000.

[FDN+01] D.J. Frank, R.H. Dennard, E. Nowak, P.M. Solomon, Y. Taur, and Hon-Sum Philip Wong. Device scaling limits of Si MOSFETs and their application dependencies. *Proceedings of the IEEE*, 89(3):259–288, March 2001.

[FFL05] J A B Fortes, R J Figueiredo, and M S Lundstrom. Virtual computing infrastructures for nanoelectronics simulation. *Proceedings of the IEEE*, 93(10):1839–1847, 2005.

[Fis03] M.V. Fischetti. Scaling MOSFETs to the Limit: A Physicists's Perspective. *Journal of Computational Electronics*, 2:73–79, 2003.

[FK04] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2004.

[FKNT02] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Technical report, 2002.

[FL88] Massimo V. Fischetti and Steven E. Laux. Monte carlo analysis of electron transport in small semiconductor devices including band–structure and space–charge effects. *Phys. Rev. B*, 38(14):9721–9745, Nov 1988.

[FL93a] M. V. Fischetti and S. E. Laux. Monte Carlo Study of Electron Transport in Silicon Inversion Layers. *Physical Rev. B*, 48(4):2244–2274, 1993.

[FL93b] M. V. Fischetti and S. E. Laux. Monte Carlo study of electron transport in silicon inversion layers. *Phys. Rev. B*, 48(4):2244–2274, Jul 1993.

[Fos01] I. Foster. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, 15(3):200–222, August 2001.

[Fos02] I. Foster. What is the Grid? A Three Point Checklist. *GRID today*, 1(6):22–25, 2002.

[FSVF02] G.F. Formicone, M. Saraniti, D.Z. Vasileska, and D.K. Ferry. Study of a 50 nm n–MOSFET by ensemble Monte Carlo simulation including a new approach to surface roughness and impurity scattering in the Si inversion layer. *Electron Devices, IEEE Transactions on*, 49(1):125–132, January 2002.

[Fuc38] K. Fuchs. The conductivity of thin metallic films according to the electron theory of metals. *Mathematical Proceedings of the Cambridge Philosophical Society*, (34):100–108, 1938.

[Fus] FusionGRID. http://nees.org/.

[Gar94] Carl L. Gardner. The quantum hydrodynamic model for semiconductor devices. *SIAM J. Appl. Math.*, 54:409–427, April 1994.

[GCCRJM02] F. Gámiz, P. Cartujo-Cassinello, J. B. Roldán, and F. Jiménez-Molinos. Electron transport in strained Si inversion layers grown on SiGe–on–insulator substrates. *Journal of Applied Physics*, 92(1):288, 2002.

[Gee05] D. Geer. Chip makers turn to multicore processors. *Computer*, 38(5):11–13, 2005.

[GF03] F. Gámiz and M. V. Fischetti. Remote Coulomb scattering in metal–oxide–semiconductor field effect transistors: Screening by electrons in the gate. *Applied Physics Letters*, 83(23):4848, 2003.

[GF04] M. J. Gilbert and D. K. Ferry. Efficient quantum three–dimensional modeling of fully depleted ballistic silicon–on–insulator metal–oxide–semiconductor field–effect–transistors. *Journal of Applied Physics*, 95(12):7954, June 2004.

[GFW+85] S. M. Goodnick, D. K. Ferry, C. W. Wilmsen, Z. Liliental, D. Fathy, and O. L. Krivanek. Surface roughness at the Si(100)–SiO_2 interface. *Phys. Rev. B*, 32(12):8171–8186, Dec 1985.

[GLAS+09] A. Garcia-Loureiro, M. Aldegunde, N. Seoane, K. Kalna, and A. Asenov. 3D Drift–Diffusion Simulation with Quantum–Corrections of Tri–Gate MOSFETs. In *Electron Devices, 2009. CDE 2009. Spanish Conference on*, pages 200–203, feb. 2009.

[GLi] GLite. http://glite.web.cern.ch/glite/.

[GLKA05] A.J. García-Loureiro, K. Kalna, and A. Asenov. Efficient three–dimensional parallel simulations of PHEMTs. *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, 18(5):327–340, 2005.

[Gor07] Pam Frost Gorder. Multicore Processors for Science and Engineering. *Computing in Science and Engineering*, 9:3–7, 2007.

[GP93] T. González and D. Pardo. Ensemble Monte Carlo with Poisson solver for the study of current fluctuations in homogeneous GaAs structures. *Journal of Applied Physics*, 73(11):7453, 1993.

[gpr] GNU gprof. http://www.gnu.org/software/binutils/manual/gprof–2.9.1/gprof.html.

[Gri] ESG. Earth System Grid. http://www.earthsystemgrid.org/.

[GRLV98] F. Gámiz, J. B. Roldán, and J. A. López-Villanueva. Phonon–limited electron mobility in ultrathin silicon–on–insulator inversion layers. *Journal of Applied Physics*, 83(9):4802, 1998.

[GRLV+99] F. Gámiz, J.B. Roldán, J.A. López-Villanueva, P. Cartujo-Cassinello, and J.E. Carceller. Surface roughness at the Si–SiO_2 interfaces in fully depleted silicon–on–insulator inversion layers. *J. Appl. Phys.*, 86(12):6854–6863, 1999.

[GRLV+01] F. Gámiz, J. B. Roldán, J. A. López-Villanueva, P. Cartujo-Cassinello, J. E. Carceller, P. Cartujo, and F. Jiménez-Molinos. Electron transport in silicon–on–insulator devices. *Solid–State Electronics*, 45(4):613–620, 2001.

[HAB+07] Liangxiu Han, Asen Asenov, Dave Berry, Campbell Millar, Gareth Roy, Scott Roy, Richard Sinnott, and Gordon Stewart. Towards a Grid–Enabled Simulation Framework for Nano–CMOS Electronics. In *E–SCIENCE '07: Proceedings of the Third IEEE International Conference on e–Science and Grid Computing*, pages 305–311, Washington, DC, USA, 2007. IEEE Computer Society.

[Ham64] D. C. Hammersley, J. M. & Handscomb. *Monte Carlo Methods*. Chapman and Hall, 1964.

[HKKT89] D. Hisamoto, T. Kaga, Y. Kawanoto, and E. Takeda. A fully depleted lean–channel transistor (delta)–a novel vertical ultra thin SOI MOSFET. *Electron Devices Meeting, 1989. Technical Digest., International*, pages 833–836, 1989.

[HLK+00] D. Hisamoto, Wen-Chin Lee, J. Kedzierski, H. Takeuchi, K. Asano, C. Kuo, E. Anderson, Tsu-Jae King, J. Bokor,

and Chenming Hu. FinFET–A self–aligned double–gate MOSFET scalable to 20 nm. *Electron Devices, IEEE Transactions on*, 47(12):2320–2325, December 2000.

[HP06] John L. Hennessy and David A. Patterson. *Computer Architecture: A Quatitative Approach*. Morgan Kaufmann, 4^{th} edition, 2006.

[i2g] http://www.grid.ie/i2g/.

[IKW02] M. Ida, K. Kurishima, and N. Watanabe. Over 300 GHz f_T and f_{max} InP/InGaAs double heterojunction bipolar transistors with a thin pseudomorphic base. *Electron Device Letters, IEEE*, 23(12):694–696, December 2002.

[IN07] S.S. Iyer and E.J. Nowak. 45 nm SOI and Beyond – Getting to a General Purpose Technology. In *SOI Conference, 2007 IEEE International*, pages 1–4, 2007.

[ITR10] ITRS. The International Technology Roadmap for Semiconductors, http://www.itrs.net/, 2010.

[IWN+02] M. Ieong, H.-S.P. Wong, E. Nowak, J. Kedzierski, and E.C. Jones. High performance double–gate device technology challenges and opportunities. In *Quality Electronic Design, 2002. Proceedings. International Symposium on*, 2002.

[JJS06] Held Jim, Bautista Jerry, and Koehl Sean. From a Few Cores to Many: A Tera–scale Computing Research Overview. White paper Intel Corporation, 2006.

[JL89] Carlo Jacoboni and Paolo Lugli. *The Monte Carlo Method for Semiconductor Device Simulation*. Springer-Verlag Wien New York, 1989.

[JMGD08] F. Jiménez-Molinos, F. Gámiz, and L. Donetti. Coulomb scattering in high–κ gate stack silicon–on–insulator metal–

oxide–semiconductor field effect transistors. *Journal of Applied Physics*, 104(6):063704, 2008.

[Job] Job Description Language Attributes Specification for the GLite Middleware. *https://edms.cern.ch/document/555796/1.*

[JR83] Carlo Jacoboni and Lino Reggiani. The Monte Carlo method for the solution of charge transport in semiconductors with applications to covalent materials. *Rev. Mod. Phys.*, 55(3):645–705, Jul 1983.

[JTC02] P. Jong-Tae and J.-P. Colinge. Multiple–gate SOI MOSFETs: device design guidelines. *Electron Devices, IEEE Transactions on*, 49(12):2222–2229, dec 2002.

[KA08] A. Kranti and G.A. Armstrong. High Tolerance to Gate Misalignment in Low Voltage Gate–Underlap Double Gate MOSFETs. *IEEE Electron Device Lett.*, 29(5):503–505, May 2008.

[KAAM+08] K. Kalna, A. Asenov, J.S. Ayubi-Moak, A.J. Craven, R. Droopad, R. Hill, M.C. Holland, X. Li, A.R. Long, P. Longo, D. MacIntyre, M. Passlack, G. Paterson, C.R. Stanley, S. Thoms, H. Zhou, and I.G. Thayne. III–V MOSFETs for Digital Applications with Silicon Co–Integration. In *Advanced Semiconductor Devices and Microsystems, 2008. ASDAM 2008. International Conference on*, pages 39–46, 2008.

[KDD+06] J. Kavalieros, B. Doyle, S. Datta, G. Dewey, M. Doczy, B. Jin, D. Lionberger, M. Metz, W. Rachmady, M. Radosavljevic, U. Shah, N. Zelick, and R. Chau. Tri–Gate Transistor Architecture with High–k Gate Dielectrics, Metal Gates and Strain Engineering. In *VLSI Technol-*

ogy, 2006. Digest of Technical Papers. 2006 Symposium on, pages 50–51, 2006.

[Kit95] C. Kittel. *Introduction to Solid State Physics*. Wiley, 2011 edition, 1995.

[KM09] Kurt Keutzer and Tim Mattson. A design pattern language for engineering (parallel) software. *Intel Technology Journal*, 13(4):118–129, 2009.

[KMTP91] H. Kim, H. Min, T. Tang, and Y. Park. An extended proof of the Ramo–Shockley theorem. *Solid–State Electronics*, 34(11):1251–1253, November 1991.

[KVAT10] Shesha Krishnapura, Lal Vipul, Shaji Achuthan, and Tang Ty. Faster Silicon Design with Intel Xeon Processor 7500 Series. White paper Intel Corporation, 2010.

[Lun00] Mark S. Lundstrom. *Fundamentals of Carrier Transport*. Cambridge University Press, 2000.

[MAA+07] K. Mistry, C. Allen, C. Auth, B. Beattie, D. Bergstrom, M. Bost, M. Brazier, M. Buehler, A. Cappellani, R. Chau, C.-H. Choi, G. Ding, K. Fischer, T. Ghani, R. Grover, W. Han, D. Hanken, M. Hattendorf, J. He, J. Hicks, R. Huessner, D. Ingerly, P. Jain, R. James, L. Jong, S. Joshi, C. Kenyon, K. Kuhn, K. Lee, H. Liu, J. Maiz, B. Mclntyre, P. Moon, J. Neirynck, S. Pae, C. Parker, D. Parsons, C. Prasad, L. Pipes, M. Prince, P. Ranade, T. Reynolds, J. Sandford, L. Shifren, J. Sebastian, J. Seiple, D. Simon, S. Sivakumar, P. Smith, C. Thomas, T. Troeger, P. Vandervoorn, S. Williams, and K. Zawadzki. A 45 nm Logic Technology with High–k+Metal Gate Transistors, Strained Silicon, 9 Cu Interconnect Layers, 193 nm Dry Patterning, and 100% Pb–free Packaging. In *Electron*

Devices Meeting, 2007. IEDM 2007. IEEE International, pages 247–250, 2007.

[Moo65] G.E. Moore. Cramming More Components onto Integrated Circuits. *Electronics*, 38(8):114–117, 1965.

[MRA10] S. Markov, S. Roy, and A. Asenov. Direct Tunnelling Gate Leakage Variability in Nano–CMOS Transistors. *Electron Devices, IEEE Transactions on*, 57(11):3106–3114, nov. 2010.

[MSA+05] A. Martinez, A. Svizhenko, M.P. Anantram, J.R. Barker, A.R. Brown, and A. Asenov. A study of the effect of the interface roughness on a DG–MOSFET using a full 2D NEGF technique. In *Electron Devices Meeting, 2005. IEDM Technical Digest. IEEE International*, pages 4 pp.–616, dec. 2005.

[MSB+10] A. Martinez, N. Seoane, A.R. Brown, J.R. Barker, and A. Asenov. Variability in Si Nanowire MOSFETs Due to the Combined Effect of Interface Roughness and Random Dopants: A Fully Three–Dimensional NEGF Simulation Study. *Electron Devices, IEEE Transactions on*, 57(7):1626–1635, july 2010.

[nan] nanoHub. http://nanohub.org/.

[NES] NESS. Network for Earthquake Engineering Simulation. http://nees.org/.

[ngi] http://knowledge.eu–egi.eu/knowledge/index.php/Spain.

[OD] OGSA-DAI. http://www.ogsadai.org.uk/index.php.

[opea] http://www.openmp.org.

[opeb] OpenCL. Open Computing Language. http://www.khronos.org/opencl/.

[Pas05] Matthias Passlack. Development methodology for high–κ gate dielectrics on III–V semiconductors: $Gd_xGa_{0,4-x}O_{0,6}/Ga_2O_3$ dielectric stacks on GaAs. *Journal of Vacuum Science & Technology B: Microelectronics and Nanometer Structures*, 23(4):1773–1781, 2005.

[PBS] PBS. http://www.pbsworks.com/.

[pkI] pkIRISgrid. http://www.irisgrid.es/pki/index.en.phtml.

[PKK+06] A. Pethe, T. Krishnamohan, Donghyun Kim, Saeroonter Oh, H-S.P. Wong, and K. Saraswat. Investigation of the Performance Limits of III–V Double–Gate n–MOSFETs. In *University/Government/Industry Microelectronics Symposium, 2006 16th Biennial*, pages 47–50, 2006.

[Pyt] Python. http://www.python.org/.

[QNSM+09] Damien Querlioz, Huu-Nha Nguyen, Jérôme Saint-Martin, Arnaud Bournel, Sylvie Galdin-Retailleau, and Philippe Dollfus. Wigner–Boltzmann Monte Carlo approach to nanodevice simulation: from quantum to semiclassical transport. *Journal of Computational Electronics*, 8(3–4):324–335, August 2009.

[Ram39] S. Ramo. Currents Induced by Electron Motion. In *Proceedings of the IRE*, volume 27, pages 584–585, September 1939.

[Rap] Rapid. http://www.omii.ac.uk/wiki/rapid.

[Rav98] Umberto Ravaioli. Hierarchy of simulation approaches for hot carrier transport in deep submicron devices. *Semiconductor Science and Technology*, 13(1):1, 1998.

[RKH+01] K. Rim, S. Koester, M. Hargrove, J. Chu, P.M. Mooney, J. Ott, T. Kanarsky, P. Ronsheim, M. Ieong, A. Grill, and

H.-S.P. Wong. Strained Si NMOSFETs for high performance CMOS technology. In *VLSI Technology, 2001. Digest of Technical Papers. 2001 Symposium on*, 2001.

[RKR+05] R. Ravishankar, G. Kathawala, U. Ravaioli, S. Hasan, and M. Lundstrom. Comparison of Monte Carlo and NEGF simulation of double gate MOSFETs. *J. Comput. Electron.*, 4(1/2):39–43, April 2005.

[RKWT10] P. Ramm, A. Klumpp, J. Weber, and M. Taklo. 3D System–on–Chip technologies for More than Moore systems. *Microsystem Technologies*, 16:1051–1055, 2010.

[RMN+09] R.Valin, M.Aldegunde, N.Seoane, A.J.Garcia-Loureiro, C.Sampedro, A.Godoy, and F.Gamiz. Using Grid Infrastructures for a Stationary DGSOI Monte Carlo Simulation. In *2009 Spanish Conference on Electron Devices*, pages 172–175, 2009.

[RMR+09] Dave Reid, Campbell Millar, Scott Roy, Gareth Roy, Richard Sinnott, Gordon Stewart, Graeme Stewart, and Asen Asenov. Enabling cutting–edge semiconductor simulation through grid technology. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 367(1897):2573–84, June 2009.

[SAG] SAGA. http://saga.cct.lsu.edu/.

[SCB+04] Huiling Shang, J.O. Chu, S. Bedell, E.P. Gusev, P. Jamison, Ying Zhang, J.A. Ott, M. Copel, D. Sadana, K.W. Guarini, and Meikei Ieong. Selectively formed high mobility strained Ge PMOSFETs for high performance CMOS. In *Electron Devices Meeting, 2004. IEDM Technical Digest. IEEE International*, pages 157–160, dec 2004.

[SCB+08] A.A. Salman, Shuqing Cao, S.G. Beebe, M.M. Pelella, and R.W. Dutton. Double–well field effect diode vs. SCR behavior under CDM stress in 45 nm SOI technology. In *SOI Conference, 2008. SOI. IEEE International*, pages 143–144, 2008.

[Sch98] Andreas Schenk. *Advanced Physical Models for Silicon Device Simulation Advanced Physical Models for Silicon Device Simulation*. Springer, 1998.

[Sel84] S. Selberherr. *Analysis and Simulation of Semiconductor Devices*. Springer–Verlag, 1984.

[SET] SETI@HOME. http://setiathome.ssl.berkeley.edu/.

[SF94] Dongwook Suh and J.G. Fossum. Dynamic floating–body instabilities in partially depleted SOI CMOS circuits. In *Electron Devices Meeting, 1994. IEDM '94. Technical Digest., International*, pages 661–664, dec 1994.

[SGG+10] C. Sampedro, F. Gámiz, A. Godoy, R. Valín, A. García-Loureiro, and F.G. Ruiz. Multi–Subband Monte Carlo study of device orientation effects in ultra–short channel DGSOI. *Solid–State Electronics*, 54(2):131–136, 2010.

[SGGR06] C. Sampedro, F. Gamiz, A. Godoy, and F.J. García Ruiz. The Multivalley Effective Conduction Band–Edge Method for Monte Carlo Simulation of Nanoscale Structures. *IEEE Trans. Electron Devices*, 53:2703–2710, 2006.

[SGLA+09] N Seoane, A Garcia-Loureiro, M Aldegunde, K Kalna, and A Asenov. Impact of intrinsic parameter fluctuations on the performance of $In_{0.75}Ga_{0.25}As$ implant free MOSFETs. *Semiconductor Science and Technology*, 24(5):055011, 2009.

[SGRG06] C Sampedro, F Gamiz, Francisco J García Ruiz, and A Godoy. The Multivalley Effective Conduction Band–Edge Method for Monte Carlo Simulation of Nanoscale Structures. *IEEE Trans. Electron Devices*, 53:2703–2710, 2006.

[SGZ+03] P.M. Solomon, K.W. Guarini, Y. Zhang, K. Chan, E.C. Jones, G.M. Cohen, A. Krasnoperova, M. Ronay, O. Dokumaci, H.J. Hovel, J.J. Bucchignano, Jr. Cabral, C., C. Lavoie, V. Ku, D.C. Boyd, K. Petrarca, J.H. Yoon, I.V. Babich, J. Treichler, P.M. Kozlowski, J.S. Newbury, C.P. D'Emic, R.M. Sicina, J. Benedict, and H.-S.P. Wong. Two gates are better than one [double–gate MOSFET process]. *Circuits and Devices Magazine, IEEE*, 19(1):48–62, January 2003.

[SH84] T. Sekigawa and Y. Hayashi. Calculated threshold–voltage characteristics of an XMOS transistor having an additional bottom gate. *Solid–State Electron*, 27(8–9):827–828, 1984.

[Shi10] Robert (Intel Corporation) Shiveley. A Catalyst for Mission–Critical Transformation, 2010.

[Sho38] W. Shockley. Currents to Conductors Induced by a Moving Point Charge. *Journal of Applied Physics*, 9(10):635, April 1938.

[SKK+07] Y. Sun, S.J. Koester, E.W. Kiewra, J.P. de Souza, N. Ruiz, J.J Bucchignano, A. Callegari, K.E. Fogel, D.K. Sadana, J. Fompeyrine, D.J. Webb, J.-P. Locquet, M. Sousa, and R. Germann. Post–Si CMOS: III–V n–MOSFETs with High–k Gate Dielectrics. In *Proceedings of the 2007 Compound Semiconductor Integrated Circuit Symposium*, pages 231–234, 2007.

[SMC03] Jian Shen, Tsz Yin Man, and Mansun Chan. 2D Analysis of Bottom Gate Misalignment and Process Tolerant for sub–100 nm Symmetric Double–Gate MOSFETs. In *Electron Devices and Solid–State Circuits, 2003 IEEE Conference on*, pages 201–204, Dec. 2003.

[SMQBD06] Jerome Saint-Martin, Damien Querlioz, Arnaud Bournel, and Philippe Dollfus. Efficient multi sub-band Monte Carlo simulation of nano-scaled Double Gate MOSFETs. In *2006 International Conference on Simulation of Semiconductor Processes and Devices*, pages 216–219. IEEE, September 2006.

[SSNS09] Dighe Saurabh, Vangal Sriram, Borkar Nitin, and Borkar Shekhar. Lessons learned from the 80–core Tera–scale research. *Intel Technology Journal*, 13(4):118–129, 2009.

[SZC+09] Bratin Saha, Xiaocheng Zhou, Hu Chen, Ying Gao, Shoumeng Yan, and Sai Luo. A programming model for heterogeneous INTEL X86 platforms. *Intel Technology Journal*, 13(4):42–61, 2009.

[TAA+04] S.E. Thompson, M. Armstrong, C. Auth, S. Cea, R. Chau, G. Glass, T. Hoffman, J. Klaus, Zhiyong Ma, B. Mcintyre, A. Murthy, B. Obradovic, L. Shifren, S. Sivakumar, S. Tyagi, T. Ghani, K. Mistry, M. Bohr, and Y. El-Mansy. A logic nanotechnology featuring strained–silicon. *Electron Device Letters, IEEE*, 25(4):191–193, 2004.

[TBC+97] Yuan Taur, D.A. Buchanan, Wei Chen, D.J. Frank, K.E. Ismail, Shih-Hsien Lo, G.A. Sai-Halasz, R.G. Viswanathan, H.-J.C. Wann, S.J. Wind, and Hon-Sum Wong. CMOS scaling into the nanometer regime. *Proceedings of the IEEE*, 85(4):486–504, April 1997.

[Ter] TeraGrid. https://www.teragrid.org/.

[Tom93] K Tomizawa. *Numerical Simulation of Submicron Semiconductor Devices*. Artech House, Boston, 1993.

[Too] Globus Toolkit. http://www.globus.org/toolkit/.

[top] TOP500 Supercomputer Sites. http://www.top500.org/.

[TR01] Hideaki Tsuchiya and Umberto Ravaioli. Particle Monte Carlo simulation of quantum phenomena in semiconductor nanostructures. *Journal of Applied Physics*, 89(7):4023, 2001.

[TTS+06] Hiroshi Takemiya, Yoshio Tanaka, Satoshi Sekiguchi, Shuji Ogata, Rajiv K. Kalia, Aiichiro Nakano, and Priya Vashishta. Sustainable adaptive grid supercomputing: multiscale simulation of semiconductor processing across the pacific. In *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 106, New York, NY, USA, 2006. ACM.

[TvS02] Andrew S. Tanenbaum and Maarten van Steen. *Distributed Systems*. Prentice Hall, 2002.

[VGLA+09] R. Valin, A. Garcia-Loureiro, M. Aldegunde, N. Seoane, T.F. Pena, J.C. Cabaleiro, F.F. Rivera, D. Cordero Placer, C. Fernandez Iglesias, J. Lopez Cacheiro, C. Fernandez Sanchez, and J.I. Lopez Cabido. Gridification of a Nanodevice Monte Carlo Simulator for the Formiga Project. In *3rd Iberian Grid Infrastructure Conference Proceedings*, pages 109–116, Valencia, Spain, 20/05/09 2009. Netbiblo.

[Vog07] Eric Vogel. Technology and metrology of new electronic materials and devices. *Nat. Nano.*, 2(1):25–32, December 2007.

[VRD+02a] R. Venugopal, Z. Ren, S. Datta, M. S. Lundstrom, and D. Jovanovic. Simulating quantum transport in nanoscale

transistors: Real versus mode–space approaches. *Journal of Applied Physics*, 92(7):3730, sep. 2002.

[VRD+02b] R. Venugopal, Z. Ren, S. Datta, M. S. Lundstrom, and D. Jovanovic. Simulating quantum transport in nanoscale transistors: Real versus mode–space approaches. *Journal of Applied Physics*, 92(7):3730, September 2002.

[VSA+10] R. Valin, N. Seoane, M. Aldegunde, A. Garcia-Loureiro, C. Sampedro, A. Godoy, and F. Gamiz. Channel Inversion Charge Dependence on Silicon Thickness in Ultra Thin Double–Gate SOI MOSFETs. pages 83–84, Grenoble, France, 2010.

[VSS+09] R Valin, C Sampedro, N Seoane, A Godoy, M Aldegunde, A.J. Garcia Loureiro, and F Gámiz. Study of the Influence of Gate Misalignment on DGSOI MOSFETs. In *Fifth Workshop on the Thematic Network on Silicon–On–Insulator*, pages 47–48, Gotemburgo, Suecia, 2009.

[WFS98] H.-S.P. Wong, D.J. Frank, and P.M. Solomon. Device design considerations for double–gate, ground–plane, and single–gated ultra–thin SOI MOSFET's at the 25 nm channel length generation. In *Electron Devices Meeting, 1998. IEDM '98 Technical Digest., International*, pages 407–410, dec 1998.

[WFTS94] Hon-Sum Wong, D.J. Frank, Yuan Taur, and J.M.C. Stork. Design and Performance Considerations for Sub–0.1μm Double–Gate SOI MOSFET's. In *Electron Devices Meeting, 1994. IEDM '94. Technical Digest., International*, pages 747–750, Dec 1994.

[WLV+05] J Widiez, J Lolivier, M Vinet, T Poiroux, B Previtali, F Dauge, M Mouis, and S Deleonibus. Experimental Evaluation of Gate Architecture Influence on DG SOI MOSFETs

Performance. *IEEE Trans. Electron Devices*, 52(8):1772–1779, 2005.

[WMH+00] C.S. Whelan, P.F. Marsh, W.E. Hoke, R.A. McTaggart, C.P. McCarroll, and T.E. Kazior. GaAs metamorphic HEMT (MHEMT): an attractive alternative to InP HEMTs for high performance low noise and power applications. In *Proceedings of the 2000 International Conference on Indium Phosphide and Related Materials*, 2000.

[WML+03] R.J. Welty, K. Mochizuki, C.R. Lutz, R.E. Welser, and P.M. Asbeck. Design and performance of tunnel collector HBTs for microwave power amplifiers. *Electron Devices, IEEE Transactions on*, 50(4):894–900, April 2003.

[Won02] H.-S. P. Wong. Beyond the conventional transistor. *IBM J. Res. Dev.*, 46:133–168, March 2002.

[WR03] B. Winstead and U. Ravaioli. A quantum correction based on Schrödinger equation applied to Monte Carlo device simulation. *IEEE Transactions on Electron Devices*, 50(2):440–446, February 2003.

[WSF01] A. Wettstein, A. Schenk, and W. Fichtner. Quantum device–simulation with the density–gradient model on unstructured grids. *Electron Devices, IEEE Transactions on*, 48(2):279–284, February 2001.

[YC05] Chunshan Yin and P.C.H. Chan. Investigation of the Source/Drain Asymmetric Effects due to Gate Misalignment in Planar Double–Gate MOSFETs. *IEEE Trans. Electron Devices*, 52(1):85–90, Jan. 2005.

[YCA+02] Bin Yu, Leland Chang, S. Ahmed, Haihong Wang, S. Bell, Chih-Yuh Yang, C. Tabery, Chau Ho, Qi Xiang, Tsu-Jae King, J. Bokor, Chenming Hu, Ming-Ren Lin, and

D. Kyser. FinFET scaling to 10 nm gate length. In *Electron Devices Meeting, 2002. IEDM '02. Digest. International*, 2002.

[YES+02] Y. Yamashita, A. Endoh, K. Shinohara, K. Hikosaka, T. Matsui, S. Hiyamizu, and T. Mimura. Pseudomorphic $In_{0.52}Al_{0.48}As/In_{0.7}Ga_{0.3}As$ HEMTs with an ultrahigh f_T of 562 GHz. *Electron Device Letters, IEEE*, 23(10):573–575, October 2002.

[YWK+03] P.D. Ye, G.D. Wilk, J. Kwo, B. Yang, H.-J.L. Gossmann, M. Frei, S.N.G. Chu, J.P. Mannaerts, M. Sergent, M. Hong, K.K. Ng, and J. Bude. GaAs MOSFET with oxide gate dielectric grown by atomic layer deposition. *Electron Device Letters, IEEE*, 24(4):209–211, 2003.

[ZDZ+06] L. Zhong, W.L. Daniel, Z. Zhang, S.A. Campbell, and W.L. Gladfelter. Atomic Layer Deposition, Characterization, and Dielectric Properties of HfO_2SiO_2 Nanolaminates and Comparisons with Their Homogeneous Mixtures. *Chemical Vapor Deposition*, 12(2–3):143–150, 2006.

[ZWK+00] Y. P. Zhao, J. R. Watling, S. Kaya, A. Asenov, and J. R. Barker. Drift diffusion and hydrodynamic simulations of Si/SiGe p–MOSFETs. *Materials Science and Engineering B*, 72(2–3):180–183, 2000.