La hoja de ruta de la ingeniería de computadores al final de la ley de Moore y el escalado de Dennard

Julio Ortega, Mancia Anguita, Alberto Prieto, Antonio Cañas, Miguel Damas, Antonio F. Díaz, Javier Fernández, Jesús González

Departamento de Arquitectura y Tecnología de Computadores
E.T.S.I.I.T., Universidad de Granada

{jortega, manguita, aprieto, acanas, mdamas, afdiaz, jfernand, jesusgonzalez}@ugr.es

Resumen. En el presente trabajo se hace una revisión sobre la situación de la ingeniería de computadores al inicio de la década de los 2020 con objeto de perfilar algunos de los cambios que deberían establecerse en la enseñanza superior de esta disciplina. Se considera la gran relevancia del control del consumo energético y de las aplicaciones relacionadas con clasificación y optimización que requieren cantidades ingentes de datos (big data) y tiempos de respuesta difícilmente alcanzables utilizando las técnicas tradicionales de la ingeniería de computadores, y dada la reducción del ritmo que marca la ley de Moore y el final del escalado de Dennard. El artículo proporciona referencias bibliográficas recientes sobre la situación de la ingeniería de computadores, e identifica los nuevos requisitos de las interfaces presentes en la jerarquía de capas propia de los sistemas de cómputo, fundamentalmente los relacionados con la seguridad, el consumo energético, y el aprovechamiento del paralelismo heterogéneo. También se reflexiona sobre los límites teóricos que se pueden establecer para la computación y las expectativas que ofrece la computación cuántica.

Palabras clave: Arquitectura de Computadores, Arquitecturas de uso específico (DSA, *Domain-Specific Architectures*), Ataques *meltdown* y *spectre*, Computación energéticamente eficiente, Computación cuántica, Ingeniería de computadores, Paralelismo heterogéneo.

Abstract. This paper reviews the state of Computer Engineering at the beginning of the 2020s in order to outline some of the changes that should be established in higher education in this discipline. It is considered the great relevance of controlling energy consumption and applications related to classification and optimization that require huge amounts of data (*big data*) and response times difficult to achieve using traditional techniques of computer engineering, and given the reduction of the improvement rate set by *Moore's law* and the end of *Dennard scaling*. The article also provides recent bibliographical references on the situation of Computer Engineering, and identifies the new requirements of the interfaces present in the hierarchy of layers of computer systems, mainly those related to security, energy consumption, and the use of heterogeneous parallelism. It also reflects on the theoretical limits that can be established for computation and the expectations that quantum computation offers.

Keywords: Computer Architecture, Computer Engineering, Domain-Specific Architecture (DSA), Heterogeneous parallel computing, *Meltdown* and *Spectre* Attacks, Power-aware computing, Quantum computing.

1 Introducción

En 2005, algunos de los autores de este artículo publicaron un libro de texto de Arquitectura de Computadores [1], que se ha venido utilizando en diversas asignaturas del ámbito de la Ingeniería de Computadores. Han transcurrido casi quince años desde 2005, y la Ingeniería de Computadores ha experimentado cambios en los paradigmas que marcan los principios cuantitativos y cualitativos para el desarrollo de nuevos computadores, que deben tenerse en cuenta tanto en el diseño de nuevos planes de estudio para las carreras relacionados con la Ingeniería de Computadores, como en la elaboración de nuevos libros de texto para asignaturas en dicho ámbito. Así, en este artículo también se proponen los contenidos que debería incluir una posible nueva edición de un libro de texto de Arquitectura de Computadores para que cubra los contenidos relevantes de las asignaturas relacionadas de Ingeniería de Computadores en los próximos años. Para realizar esta propuesta se han considerado iniciativas que marcan los nuevos *roadmaps* a seguir, como es el caso de la última hoja de ruta de semiconductores publicada por la ITRS (*InternationalTechnology Roadmap for Semiconductors* 2.0), www.itrs2.net. Hay que tener en cuenta que el número de transistores incluidos en un circuito integrado seguramente seguirá creciendo, durante algunos años más, según lo establecido por la ley de Moore. Sin embargo, el problema aparece en el rendimiento que se extrae al añadir más transistores. Así, cuantas más etapas tenga un cauce, mayor es el número de instrucciones que se necesitan para mantenerlo lleno y maximizar el rendimiento de los transistores añadidos. El número de instrucciones disponibles para entrar en el cauce depende del número de instrucciones de salto que aparezcan en el código, aunque mediante el procesamiento especulativo se pueden seguir introduciendo instrucciones en el cauce tras una instrucción de salto condicional no resuelto (aunque no todas las instrucciones que se empiecen a procesar tengan que finalizar y actualizar sus resultados). A medida que los cauces son más “profundos”, desperdician una mayor cantidad del trabajo que realizan en sus etapas dado que se pueden introducir inútilmente más instrucciones ubicadas en las alternativas de las instrucciones de salto condicional que no hay que tomar. Hacia 2003, los procesadores superaban consumos de 200W por circuito integrado, marcando un umbral de potencia a partir del cual es más cara la tecnología para *enfriar* el procesador que la del resto de elementos del computador. De hecho, la frecuencia de reloj del procesador se mantuvo alrededor de los 2 GHz durante un periodo de tiempo considerable en esos años.

La Figura 1, elaborada con datos obtenidos de [2], muestra que las prestaciones de ejecución del *benchmark* SPECint respecto al VAX 11/780 pasaron de crecer un 52% anual entre 1986 y 2003, a un 12% entre 2011 y 2015, y *sólo* un 3.5% desde entonces.

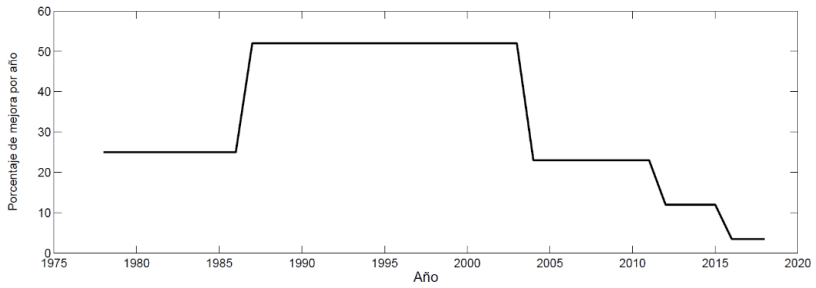

Figura 1. Evolución temporal de las prestaciones de ejecución del *benchmark* SPECint respecto al VAX 11/780 (datos obtenidos de [2]).

Además, desde aproximadamente 2004, y precisamente por la continua reducción del tamaño de los transistores, no se cumple el denominado escalado de Dennard, según el cual, para un área de silicio dada, la densidad de potencia instantánea se mantendría constante al incrementar el número de transistores. Desde alrededor de 2004 se viene observando una reducción considerable del porcentaje de mejora anual, habiéndose pasado de un 52% a un 23%. Es cada vez más difícil obtener mejoras de prestaciones basadas en aumentar la eficiencia del paralelismo entre instrucciones (ILP) en un núcleo de procesador segmentado de uso general. Además, si la frecuencia de reloj sigue creciendo en cauces cada vez más complejos, se hace más difícil conseguir que las salidas de una etapa del cauce puedan llegar, en un ciclo de reloj, a las entradas de la siguiente etapa. Una alternativa para abordar esta interacción entre aumento de la frecuencia de reloj y núcleos con cauces cada vez más complejos, ha sido la integración de más núcleos por circuito integrado, aprovechando el aumento del número de transistores disponibles. No obstante, la posible mejora de prestaciones asociada al aprovechamiento del procesamiento paralelo en estas arquitecturas de microprocesador multi-núcleo deja de ser transparente al usuario, que debe paralelizar convenientemente las aplicaciones para optimizar el uso de la microarquitectura multi-núcleo correspondiente.

Tras esta introducción, la Sección 2 describe los criterios a partir de los que se definirán las hojas de ruta para el desarrollo de las arquitecturas del computador y de los distintos elementos de la ingeniería de computadores. Precisamente la Sección 3 describe la aproximación al estudio del computador en términos de la jerarquía de capas entre las que actúan las distintas interfaces características del computador, al tiempo que se presentan los agentes que determinan la evolución de los computadores. La Sección 4 se dedica a los requisitos para las interfaces a desarrollar en un futuro próximo. Finalmente, la Sección 5 analiza los límites físicos para las prestaciones de los computadores y la Sección 6 introduce las principales características de los computadores cuánticos, que actualmente constituyen una alternativa contemplada con gran interés, para aproximar las prestaciones de los computadores a sus límites físicos, más allá de lo que se prevé conseguir mediante la tecnología electrónica utilizada actualmente. Para terminar, la Sección 7 del artículo discute una propuesta de contenidos para una asignatura de Arquitectura de Computadores (y también para el libro de texto correspondiente) a partir de las conclusiones alcanzadas a lo largo del artículo, y resumidas en la Sección 8.

2 Las nuevas hojas de ruta para el siglo XXI

En los próximos años, tras el agotamiento de la ley de Moore y el escalado de Dennard, el aumento de prestaciones en las arquitecturas no va a provenir fundamentalmente de la disponibilidad de arquitecturas con velocidades pico que se doblan cada dieciocho o veinticuatro meses, sino que habrá que prestar más atención al uso que se hace de las arquitecturas de cómputo para mejorar las prestaciones en las aplicaciones más demandadas en cada momento. Ha sido el desarrollo de las tecnologías de diseño e implementación hardware, así como la popularización de cierto tipo de aplicaciones, relacionadas con la inteligencia artificial y el *big data*, las que han impulsado el desarrollo de una tendencia en la mejora de las prestaciones basada en el uso de arquitecturas de dominio específico (DSA, *Domain-Specific Architectures*). En estas aplicaciones, gran parte del tiempo de ejecución corresponde al procesamiento de núcleos de código, o *kernels*, que actualmente implementan sobre todo tareas relacionadas con problemas de clasificación, optimización, aprendizaje, agrupamiento (*clustering*), etc, que pueden usarse en diferentes aplicaciones, con distinto nivel de granularidad. Teniendo en cuenta la ley de Amdahl, la aproximación más eficiente para acelerar la aplicación es, precisamente, buscar formulaciones de la misma que permitan aprovechar los tipos de paralelismo posibles. Así, las nuevas hojas de ruta para conseguir arquitecturas de cómputo cada vez más eficientes deben basarse en el aprovechamiento del paralelismo heterogéneo, en sus diversos niveles y para las aplicaciones más demandadas. Así, en cada momento habría que identificar los principales núcleos de procesamiento en los que la aplicación estaría involucrada la mayor parte del tiempo. Se identificarían elementos de cómputo que, al encontrarse repetidos, incluso en aplicaciones de dominio específico (elementos neuronales en redes neuronales profundas, o módulos de funciones comunes en metaheurísticas de optimización) contribuirían a reducir el coste de diseño y fabricación de las arquitecturas.

En relación con el consumo energético, las arquitecturas de uso específico, o de dominio específico (DSA) [2] permiten disponer de hardware eficiente para la ejecución de la aplicación considerada [3]. Por ejemplo, se intentan reducir las distancias entre las que hay que mover datos, incluyendo en la microarquitectura memorias controladas por software en lugar de cachés, cuyo hardware de control las hace menos eficientes energéticamente. No obstante, el diseño de DSA implica que el ingeniero de computadores debe ser consciente del proceso de optimización del hardware para la aplicación en cuestión. Es decir, no solo debe conocer la aplicación a fondo, sino también las partes de la arquitectura y la microarquitectura a optimizar: (1) el tamaño, la ubicación, y la gestión de las memorias, (2) incluir recursos hardware optimizados para los elementos de la microarquiectura que son muy frecuentemente utilizados (unidades aritmético-lógicas específicas), (3) implementar el paralelismo que precisa el dominio, (4) usar el lenguaje de programación específico para el dominio, y (5) usar el tamaño de datos ajustado a los tipos característicos del dominio.

Esta nueva forma de definir tendencias evolutivas, hojas de ruta, o *roadmaps*, en las arquitecturas de cómputo [4], constituye un cambio de paradigma del que estamos siendo testigos en esta segunda década del siglo XXI, y según la opinión de algunos

ingenieros de computadores (conferencia de Hennessy y Patterson al recibir el premio Turing de la ACM de 2017 [5]), va a ocasionar un nuevo resurgimiento de alternativas y posibilidades para la Ingeniería de Computadores. La forma usual de prefijar objetivos de mejora que se evalúan a través de conjuntos de *benchmarks* de uso general, ha dado paso a la necesidad de reenfocar las evaluaciones a aplicaciones específicas pero muy demandadas, que requieren mayores necesidades de cómputo, tanto en tiempo como en almacenamiento y eficiencia de consumo energético. Las nuevas propuestas de la Ingeniería de Computadores compiten no ya utilizando medidas como la velocidad pico, sino comparando tiempos de ejecución, consumo de energía, o necesidades de almacenamiento en versiones específicas hardware/software de instancias de aplicaciones reales muy demandadas socio-económicamente. Esta situación condiciona el tipo de recursos que se deben incluir en la arquitectura de la plataforma a considerar y, previsiblemente, a corto y medio plazo, las tendencias dominantes buscarán una redefinición de los niveles de capas. Se trata de aprovechar más eficazmente el paralelismo, e idear nuevos procedimientos para la optimización de arquitecturas orientadas a la implementación eficiente, en cuanto a velocidad, consumo energético, y aprovechamiento del paralelismo en todos sus niveles. Los nuevos conceptos deben proyectarse para las aplicaciones frecuentemente demandadas, rentabilizando la implementación de arquitecturas de dominio específico y disminuyendo los costos hardware/software no recurrentes de ingeniería, sin olvidar que el diseño de jerarquías de memoria debe ser energéticamente eficiente.

Con este panorama, el diseño de las interfaces que permitan el aprovechamiento eficiente del hardware, no solo sigue siendo fundamental, sino que debe incorporar elementos que den respuesta a las nuevas circunstancias resultantes de la situación actual de la ley de Moore y el escalado de Dennard.

3 Jerarquía de capas e ingeniería de computadores

La aproximación al estudio de un sistema complejo, como es el caso de un computador, se lleva a cabo dividiéndolo en partes que interactúan entre sí de acuerdo con un orden jerárquico. En un computador es usual definir un conjunto de capas o niveles de abstracción que se suceden unas a otras de forma que la del nivel inferior, con un menor nivel de abstracción por estar más próxima al nivel de las tecnologías físicas utilizadas para implementar el computador, genera elementos utilizables por la capa del nivel superior, y así sucesivamente hasta el nivel de aplicación, que considera programas implementados con lenguajes de alto nivel. Así, en un computador tendremos las capas de dispositivo, de lógica, de unidad funcional, de microarquitectura, de arquitectura del conjunto de instrucciones, arquitectura, de API (interfaz de programación de aplicaciones), de lenguaje, y de algoritmo [6]. Por otro lado, las distintas capas o niveles se pueden agrupar para definir bloques de elementos que interactúan y permiten entender, entre otras cosas, la evolución temporal de los computadores. Al estructurar las funciones del computador en capas es posible diseñarlo y analizarlo a través de subsistemas dotados de cierta independencia.

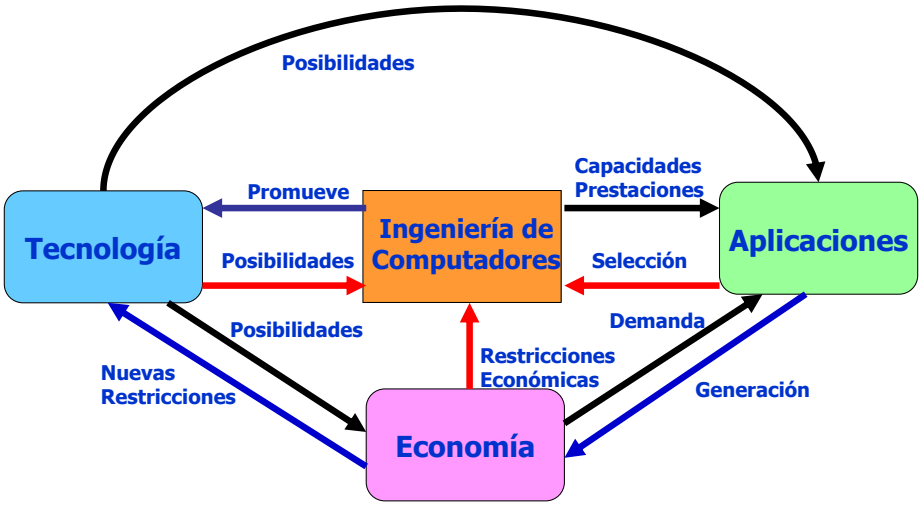

Figura 2. Interacciones entre los agentes que intervienen en la evolución de los computadores

Precisamente, la Figura 2 muestra la interrelación entre tecnología, aplicaciones, economía, e ingeniería de computadores. Así, las capas correspondientes a las tecnologías físicas que se utilizan para implementar el hardware del computador definen el bloque de ***tecnología*** de la Figura 2. También están las capas que atañen a las aplicaciones y al diseño de algoritmos y programas en lenguajes de alto nivel que se ejecutan en el computador para abordar dichas aplicaciones (en la Figura 2 se utiliza el término ***aplicaciones***), y las capas que definen lo que se conoce como ***ingeniería de computadores***, o interfaz entre el hardware y los distintos niveles de abstracción del software (capas de dispositivo, de lógica, de unidad funcional, de microarquitectura, de arquitectura del conjunto de instrucciones, arquitectura, de API). Por otra parte, la relevancia y la demanda social de las aplicaciones determinan la intervención de los factores económicos en la evolución de los computadores (bloque ***economía*** en la Figura 2). En el diagrama de la Figura 2 también se ilustra la interrelación entre tecnología, aplicaciones, economía, e ingeniería, que condiciona el desarrollo de esto sistemas. Las aproximaciones que se propongan en un futuro para mantener el crecimiento de prestaciones de las arquitecturas se pueden contemplar también considerando en qué capa, o capas, la nueva propuesta ocasiona una mayor disrupción, como se hace en [6]. Así, la denominada aproximación "*más Moore*", que busca continuar mejorando gracias al aumento del número de transistores incluidos en los circuitos integrados, afecta fundamentalmente a la capa de dispositivo. La aproximación de *"cambios ocultos"* busca mejoras a partir de innovaciones en la capa de microarquitectura y capas inferiores, y la aproximación de "cambios en la arquitectura" propone modificaciones desde la capa del conjunto de instrucciones a las más próximas al hardware. Finalmente, los computadores *"no von Neumann*" implican cambios disruptivos prácticamente en todas las capas. Aquellas capas en las que la evolución de la Ingeniería de Computadores pueda producir cambios, deben abarcar los contenidos cuyo estudio deben incluirse en un libro de texto de ingeniería de

computadores, para así proporcionar los elementos que definen el espíritu de la materia, y permitir al ingeniero mantenerse activo en su contribución al progreso futuro de la disciplina.

Los fundamentos de la ingeniería de computadores parten de conceptos propios de las áreas de computación, electrónica, matemáticas y física, y deben tener en cuenta las posibilidades de aplicación de las tecnologías emergentes que, si bien pueden ser accesibles comercialmente, y se les reconoce como posibles promotores de transformaciones sociales futuras relevantes, suelen tener consecuencias que, en el momento considerado, no están claramente definidas entre los contenidos y la orientación de la ingeniería de computadores, y precisan, por tanto, un análisis más profundo [7]. Por tanto, un ingeniero competente debe ser capaz de identificar las tecnologías emergentes que contribuyen al crecimiento de la disciplina de ingeniería de computadores en un mundo en continua transformación. De hecho, los ingenieros de computadores estarán pronto involucrados en el diseño de computadores para cuya implementación se usarán tecnologías ópticas, cuánticas y/o biológicas, y/o en el desarrollo de sistemas robóticos avanzados para la nueva industria 4.0, con técnicas de realidad virtual, de análisis de datos o *big data*, que afectarán considerablemente a su trabajo.

Así, aunque los cambios en la Ingeniería de Computadores de los últimos años han venido tanto desde los niveles más próximos al hardware, tal y como se ha comentado antes en relación con el final del escalado de Dennard y el estancamiento de las prestaciones de los cauces segmentados, y con la generalización de microarquitecturas multinúcleo y la aparición de coprocesadores orientados al aumento del flujo de operaciones (throughput), como es el caso de las GPU, también se han propuesto mejoras en las capas más altas, entre los que están los cambios relacionados con el aprendizaje automático y las redes neuronales profundas [3], que permiten reemplazar el software complejo, comúnmente denominado "*Software 2.0*", gracias a que los modelos de computación que se utilizan se basan en la interconexión de conjuntos de elementos neuronales similares, y relativamente simples de programar.

4 Las interfaces de computador del siglo XXI

Más arriba se ha definido la Ingeniería de Computadores como la interfaz entre el hardware y los distintos niveles de abstracción del software. Por tanto, como disciplina, incluye el diseño, evaluación y optimización de las interfaces entre las capas de algoritmos, descritos en términos de lenguajes de alto nivel, y las relacionadas con las tecnologías utilizadas para implementar el hardware, y permiten aislar los detalles de las capas de niveles más bajos y mayor complejidad. En este contexto, el repertorio de instrucciones máquina es la parte del computador visible para el programador y el compilador, y sus prestaciones se suelen estimar utilizando conjuntos de programas de prueba o *benchmarks* ejecutados en distintos tipos de máquinas: computadores de sobremesa, servidores, y computadores móviles personales y computadores empotrados en aplicaciones. En muchas aplicaciones, la capacidad de procesamiento

con datos en coma flotante es menos importante que la necesidad de memoria [3], dado que una disminución de la memoria necesaria supone una reducción importante del coste energético de la aplicación. Desde esta perspectiva, el tamaño del código es importante. No obstante, repertorios no RISC como el 80x86 han mantenido una posición competitiva frente a los RISC para mantener la compatibilidad con códigos para PC y gracias a que el elevado número de transistores disponible ha permitido a los ingenieros implementar procesadores CISC con microarquitecturas capaces de traducir dinámicamente los códigos máquina CISC a secuencias de micro-operaciones de tipo RISC.

Hasta el momento actual, el hardware se ha implementado fundamentalmente a través de las tecnologías electrónica y electromagnética, pero los requisitos de prestaciones para nuevas aplicaciones pueden requerir cambios en las tecnologías hardware que se venían utilizando. Así, aplicaciones propias del campo de la inteligencia artificial han promovido el desarrollo de arquitecturas de dominio específico para *deep learning* [3], y la demanda de aplicaciones que no son fácilmente abordables mediante computadores electrónicos *clásicos*, está promoviendo el uso de nuevas tecnologías [7], como en el caso de los computadores cuánticos. La evolución tecnológica ha hecho aflorar nuevos requisitos de prestaciones relacionados con la eficiencia energética, la programabilidad, la fiabilidad, y la seguridad, más allá de la velocidad de cómputo [8].

Estas circunstancias también han generado nuevas exigencias de diseño para las interfaces que, más que ocultar detalles de una capa a las capas superiores, de menor complejidad, deben tener acceso a elementos de capas de abstracción diferentes para tenerlos en cuenta de cara a conseguir un comportamiento más optimizado en cuanto a las prestaciones. Se precisa, por tanto, un replanteamiento del diseño de las interfaces para que, además de permanecer estables durante periodos de tiempo considerable, proporcionen tolerancia a fallos, verificabilidad, seguridad para los datos, y velocidad de procesamiento, sobre transistores menos fiables, y con requisitos de energía más exigentes. De hecho, la arquitectura del computador define un conjunto de interfaces (el repertorio de instrucciones y la memoria virtual) que han evolucionado lentamente a lo largo de décadas, pero fueron definidas cuando la memoria era un recurso muy escaso, no había restricciones en cuanto al consumo energético, y existía poca preocupación por la seguridad. De hecho, la programabilidad (diseño y desarrollo de software ajustado a los requisitos de velocidad, consumo, fiabilidad y seguridad), que se apoyaba en técnicas de ingeniería de software como eran la modularidad y el ocultamiento de información para incrementar la productividad del programador a expensas de la eficiencia energética y la velocidad, debe buscar nuevas alternativas de optimización entre capas de distinto nivel de abstracción, el soporte hardware de la programabilidad (por ejemplo, la memoria *transaccional* busca simplificar la paralelización y la sincronización de códigos multihebra), o el diseño de lenguajes de dominio específico y script dinámico (por ejemplo, Javascript y Python).

Los repertorios de instrucciones actuales son poco eficientes a la hora de comunicar información crítica del software de alto nivel al hardware. De hecho, no se puede especificar fácilmente cuándo un programa necesita eficiencia energética, un nivel elevado de seguridad, calidad de servicio (QoS), etc. En cambio, es el hardware el que

debe recabar esa información por sí mismo, como se puede ver en el caso de la ejecución eficiente del paralelismo ILP en los procesadores superescalares. Por otra parte, hay ciertos requisitos que los repertorios máquina y las microarquitecturas disponibles actualmente no satisfacen [8]. Entre ellos están la seguridad, la eficiencia energética, y el aprovechamiento del paralelismo heterogéneo. A continuación se resumen algunos aspectos relacionados con el diseño de interfaces que proporcionan dichos requisitos entre las capas que se comunican a través de la interfaz correspondiente.

4.1 Interfaces para la seguridad

La seguridad del computador se basa fundamentalmente en aislar las distintas áreas de memoria, separando direcciones de memoria del núcleo del sistema operativo que se marcan como no accesibles para el usuario, y separando zonas de memoria para procesos de usuarios diferentes. Sin embargo, ataques como *Meltdown* [9] o *Spectre* [10] aprovechan ciertos efectos colaterales de la ejecución desordenada (*"out-of-order", o-o-o*) para permitir la lectura de posiciones arbitrarias de la memoria del núcleo, incluyendo claves. Este tipo de ataques aprovechan elementos incluidos en la microarquitectura **(**para mejorar sus prestaciones sin afectarla funcionalmente**)** y rompen el esquema de protección de memoria al permitir que cualquier usuario pueda leer todo el espacio del núcleo del sistema operativo del computador.

La separación entre procesos de usuario y de núcleo se suele implementar a través de un bit del procesador que establece si puede accederse a una determinada página del núcleo del sistema. Para ello, el bit de acceso se activa (se pone a uno, por ejemplo) cuando se ejecuta código del núcleo del sistema y se desactiva para permitir el acceso a páginas de procesos de usuario. Este hardware facilita a los sistemas operativos ubicar el núcleo en el espacio de direcciones de cada proceso y conmutar eficientemente entre espacio de usuario y núcleo, por ejemplo para la gestión de interrupciones. Los ataques *Spectre* y *Meltdown* explotan los recursos incluidos en la microarquitectura del procesador para mejorar sus prestaciones, creando *"canales de fuga de información"* (*side channels*) al extender la especificación funcional del repertorio de instrucciones que constituye la arquitectura del computador. Ahora, el conjunto de instrucciones utilizado como base para la portabilidad entre máquinas diferentes no es suficiente para poder asumir con total garantía propiedades y características de seguridad, y es preciso además disponer de la información de prestaciones que se obtiene al conocer detalles de funcionamiento de la microarquitectura. Por tanto, las especificaciones funcionales y de prestaciones constituyen una nueva definición de la interfaz entre aplicaciones y máquina. Surge una nueva era para la arquitectura de computadores, en la que todavía es más importante conocer las características de la microarquitectura y su incidencia en las prestaciones de la máquina. Hay que tener en cuenta que una CPU superescalar con procesamiento especulativo, por ejemplo, está diseñada para mantener la corrección funcional y deshacer los cálculos realizados especulativamente de forma innecesaria. Un ataque *Spectre* hace que el procesador ejecute especulativamente secuencias de instrucciones que no se hubieran ejecutado en la parte no especulativa del programa. A través de la ejecución de instrucciones especulativas se pueden inducir pérdidas de

información desde el espacio de memoria de la víctima del ataque. Por ejemplo, si se ejecuta una secuencia especulativa de instrucciones que incluye instrucciones de acceso a memoria, puede haber bloques de memoria que den lugar a fallos de memoria. Marcos de esos bloques se podrían introducir en la cache, a pesar de que la correspondiente excepción de memoria impediría que finalicen las instrucciones de memoria ejecutadas especulativamente (el proceso en ejecución acabaría). Sin embargo, aunque la secuencia de instrucciones especulativa no se complete, y los datos en la memoria principal no se modifiquen finalmente, podrían haberse introducido marcos de memoria en caché, que posiblemente generarían diferencias temporales entre accesos a memoria debido a *caminos encubiertos* que se pueden "generar" en la microarquitectura Por ejemplo, instrucciones como FLUSH+RELOAD aprovechan el carácter compartido e inclusivo de la caché de último nivel. Así, instrucciones del tipo FLUSH forzarían la actualización de la línea de caché correspondiente en la memoria principal. Luego, al medir el tiempo necesario para volver a cargar el dato mediante una nueva instrucción de carga (RELOAD), el proceso atacante puede determinar si otro proceso cargó un marco de bloque en caché. Se podría identificar el proceso según haya sido el marco de bloque implicado, y diseñar códigos para implementar ataques, cambiando el estado de la microarquitectura, a pesar de que la ejecución del código se puede llevar a cabo sin cambios en los resultados. Tendríamos que, analizando únicamente el código máquina, no se podría prever *todo* lo que puede ocurrir en la microarquitectura, si incluimos en ese *todo*, por ejemplo, las cuestiones relacionadas con la seguridad de los códigos.

4.2 Interfaces para mejorar el consumo energético

El consumo de energía ha pasado a ser un factor muy importante en el diseño de las microarquitecturas, especialmente las de los procesadores embebidos, donde las restricciones relacionadas con la reducción del consumo siempre han sido esenciales. Además, la evolución hacia arquitecturas de uso general multi-núcleo ha sido una consecuencia de la necesidad de limitar la potencia por unidad de superficie disipada por el microprocesador, que podría haber llegado a valores inaceptablemente altos si se hubiera mantenido asociada la mejora de prestaciones a microarquitecturas superescalares cada vez más complejas tendentes a terminar más instrucciones por ciclo. Para entender la influencia de la tecnología en la potencia disipada de un circuito integrado CMOS se puede utilizar la expresión (1), en la que el primer término se debe al consumo dinámico de carga y descarga de la capacidad de salida, *C*, de una puerta lógica en un circuito integrado con un coeficiente de actividad, *A* que representa la fracción de puertas del circuito que conmutan en cada ciclo, a una tensión determinada, *V*, y una frecuencia de reloj dada, *f*. El segundo término de (1) se debe a la corriente entre fuente de alimentación y tierra, I_{cortoc}, durante el tiempo, t, en el que conmuta la puerta. El tercer y último término es la potencia consumida debido a la corriente de pérdidas, I_{leak}, independiente del estado de la puerta lógica. Es posible consumir menos potencia utilizando una frecuencia de reloj más baja o reduciendo la tensión de alimentación (con el consiguiente efecto en la velocidad del circuito).

$$Potencia = ACV^2 f + tAVI_{cortoc} + VI_{leak} \quad (1)$$

Hay que tener en cuenta que la frecuencia máxima a la que puede funcionar el circuito está relacionada con la tensión de la alimentación (2):

$$f_{max} \sim \frac{(V - V_{umbral})^2}{V} \quad (2)$$

Obsérvese que para incrementar la velocidad de procesamiento aumentando la frecuencia del procesador, habría que reducir la tensión umbral, V_{umbral}, de (2), junto con la tensión V, para mantener constante la diferencia $(V\text{-}V_{umbral})$, lo que aumentaría la corriente de pérdidas, que crece exponencialmente al reducirse la tensión umbral:

$$I_{leak} \sim e^{\frac{-qV_{umbral}}{kT}} \quad (3)$$

Por tanto, el incremento de la frecuencia máxima de los microprocesadores al reducir la tensión de alimentación y la tensión umbral está limitado por el aumento de la corriente de pérdidas. Un procesador puede consumir menos potencia que otro para ejecutar un programa, pero si tarda más tiempo puede ocurrir que al final la energía consumida sea mayor. Además de la energía consumida por el procesador o procesadores, existen otros factores que afectan al consumo energético en el computador y están relacionados con el funcionamiento de la memoria, los buses y los sistemas de interconexión, los dispositivos de entrada/salida, etc. Por tanto, la computación eficiente desde el punto de vista energético implica técnicas basadas no sólo en el diseño de la microarquitectura, sino también en los niveles de sistema computador y sistema operativo. En [11] se describen, con las correspondientes referencias, algunas técnicas y aproximaciones para reducir el consumo energético, y se muestra la conveniencia de incorporar e intensificar los contenidos relacionados con el consumo energético en las asignaturas de arquitectura y tecnología de computadores de grado, especialmente en los estudios de Ingeniería de Computadores.

4.3 Interfaces para el aprovechamiento del paralelismo heterogéneo

En cuanto al desarrollo y soporte de códigos paralelos, se necesitan interfaces que faciliten el trabajo del programador, permitiendo la expresión del paralelismo a alto nivel, la localidad, las dependencias de cómputo, los patrones de comunicación, y los efectos colaterales y las claves compartidas. Mediante este tipo de interfaces se puede utilizar un hardware más simple, con primitivas de comunicación y sincronización más eficientes a la hora de reducir la transferencia de datos.

Por otra parte, el paralelismo heterogéneo también necesita nuevas interfaces, dado que las aplicaciones deben programarse utilizando modelos de paralelismo y memoria diferentes y portables entre hardware con características específicas y diferentes, y capaces de aprovechar con cierta rapidez, las nuevas características del hardware.

También se necesitan interfaces que permitan identificar más claramente los datos de larga duración y las dependencias entre datos y código, para que el hardware y el software puedan poner de manifiesto, y de forma dinámica, los caminos críticos. La gestión de datos es más compleja en el caso de aplicaciones de *big data*, donde hay que orquestar los datos entre varios sistemas de tamaños considerables. También se necesitan interfaces capaces de especificar los límites de granularidad fina para la protección entre módulos de una aplicación y tener en cuenta criterios de seguridad como algo esencial. Algunas partes de la aplicación pueden tolerar fallos, o pueden asumir fallos con el objetivo de mejorar las prestaciones de velocidad de procesamiento. Todas estas interfaces pueden beneficiarse de mecanismos hardware apropiados como el seguimiento de flujos de información (*information-flow tracking*), bloques de recuperación transaccional (*transactional recovery blocks*), etc.

5 Los límites físicos de la computación

Existen límites para las prestaciones de un computador según sus características físicas. Consideremos, para empezar, la velocidad de computación máxima. Independientemente de que se sepa diseñar un supercomputador capaz de alcanzar una velocidad dada, no es inmediato pensar que haya un límite superior para la velocidad de cómputo que pueda alcanzar un computador, ni tampoco es trivial estimar dicho límite. Se podría pensar que para una masa y dimensiones físicas dadas, siempre es posible diseñar arquitecturas de cómputo y disponer de tecnologías físicas nuevas que permitan ejecutar un número cada vez mayor de operaciones por unidad de tiempo. No es trivial pensar que la velocidad de un computador no pueda crecer de forma ilimitada (suponiendo que se dispone de la energía necesaria). Igualmente, puede parecer extraño que haya un límite para el tamaño máximo de la memoria, y del paralelismo. En [12] se analizan los límites que la energía impone a la velocidad de cómputo, los que determina la entropía en el espacio de memoria, y los que el tamaño del computador establece para la paralelización. Para ello, se define un computador que denominaremos *computador esencial* con un volumen de un litro y una masa de un kilogramo. Disponer de estimaciones de los límites de la capacidad de cómputo puede ayudar en el análisis de las posibilidades que una tecnología determinada tiene para mejorar las características de los computadores actuales, y permite identificar tendencias prometedoras para la industria que, en cada momento, van a definir el paradigma dominante en la Ingeniería de Computadores. No existen garantías de que puedan alcanzarse las prestaciones máximas estimadas, pero, al menos teóricamente, surgirán tecnologías novedosas que, como la computación cuántica, mostrarán nuevas y prometedoras alternativas para seguir acercándose a ellas.

El límite máximo en la velocidad de un computador se podría estimar a partir del tiempo mínimo necesario, Δt, para realizar una operación lógica elemental. Por otra parte, la expresión del principio de incertidumbre de Heisenberg en términos de la incertidumbre en la energía y del tiempo, ΔE y Δt, establece una cota que relaciona ambas magnitudes de incertidumbre $\Delta E \times \Delta t \geq h/4$, donde h es la constante de Plank. Dado que la velocidad máxima alcanzable sería, por definición $1/\Delta t$, tendríamos que la cota superior para dicha

velocidad sería $(4\times\Delta E)/h$, operaciones por unidad de tiempo. En un computador de un kilogramo, si identificamos ΔE con el producto de la masa y la velocidad de la luz al cuadrado $\Delta E = m\times c^2 = 8.9874\times10^{16}$ J y la velocidad de cómputo máxima sería igual a 5.4258×10^{50} operaciones por segundo (la constante de Planck es igual a $h=6.6261\times10^{-34}$J×s y c es la velocidad de la luz). Para estimar esta cota superior de la velocidad de cómputo, se ha tenido que identificar la energía del computador con la incertidumbre en la energía, ΔE, y el tiempo necesario para completar una operación, con la incertidumbre en el tiempo, Δt. Esas identificaciones basadas en relacionar energía y procesamiento de la información, se han realizado en trabajos como [13, 14] y suponen que se es capaz de invertir toda la energía del computador en operaciones que ocasionan cambios en los estados de los bits del computador. Así, el dispositivo que realiza una operación podrá experimentar una variación de energía, ΔE, durante el tiempo, Δt, que tarda en realizarla. En el caso límite que se está considerando, la mecánica cuántica establece que un sistema puede pasar de un estado *ortogonal* a otro (esto es, pasar de un estado a otro distinguible del de partida) con una diferencia en la energía igual a ΔE, en un tiempo mínimo $\Delta t \geq h/(4\times\Delta E)$. Precisamente, en [15] se muestra que un sistema cuántico con energía media E, necesita un tiempo como mínimo igual a $h/(4\times E)$ para evolucionar a un *estado ortogonal*. En la estimación de la velocidad máxima del *computador esencial* descrito no interviene el tipo de arquitectura del mismo, a pesar de que se podría mejorar la velocidad mediante el procesamiento paralelo. Así, si la energía, E, se distribuye entre N puertas lógicas o procesadores, cada uno de esos elementos podría realizar operaciones a una velocidad máxima igual a $(4\times E)/(h\times N)$, es decir N veces menor: si la energía se distribuye entre más puertas (más procesamiento paralelo), los elementos que intervienen podrían ser más lentos. Así, el grado de paralelización de la computación a realizar puede hacer posible una distribución más eficiente de la energía entre los distintos elementos del computador. La diferencia entre la velocidad máxima del *portátil esencial* (con las dimensiones consideradas, el *computador esencial* de un kilogramo que hemos indicado podría ser un portátil) y la que alcanza un computador real de las mismas dimensiones se debe a dos causas fundamentales. En primer lugar, la mayor parte de la energía de un computador corresponde a la masa de las partículas que constituyen el computador, de forma que solo una pequeña parte de la masa está disponible para configurar la lógica de procesamiento. Además, en los computadores actuales se utilizan muchas partículas para configurar una puerta lógica o almacenar un bit (todas las partículas que constituyen los transistores de las puertas lógicas o los biestables). Precisamente, los computadores convencionales actuales aprovechan la redundancia para funcionar de manera fiable. En un computador cuántico, como se verá más adelante, no haría falta redundancia para realizar las operaciones que podrían implementarse gracias a puertas definidas por una única partícula, que proporcionaría las velocidades máximas de procesamiento, pero también hay que introducir elementos que proporcionen tolerancia a los errores inevitables en la computación cuántica, como por ejemplo los debidos a la *decoherencia*, concepto que se presentará en la Sección 6.

Las leyes de la naturaleza también marcan un límite máximo para la memoria de un computador. Este límite está relacionado con el número de estados físicos diferentes, W, a los que el computador puede acceder. Este número de estados accesibles, W, está relacionado con la entropía termodinámica del sistema, S, a través de la expresión

correspondiente a la ley del Boltzmann, $S=k_B \times lnW$, donde k_B es la constante de Boltzmann (k_B=1.38065×10−23 J/K). Así, un sistema con W estados accesibles podrá almacenar el equivalente a $b= log_2 W$ bits y, a partir de la anteriormente indicada ley de Boltzmann, se tendría que $b=S(E)/(k_B \times ln2)$, donde $S(E)$ es la entropía termodinámica del sistema, con un valor de energía esperada, E. A partir de la expresión previa, $(4 \times E)/h$, para el número máximo de operaciones por segundo, se tiene que el número de operaciones por segundo por bit es $(4 \times E)/h)/b= ((4 \times E)/h) \times (k_B \times ln2))/S(E)$. Por lo tanto, para obtener una estimación del espacio de memoria máximo del que podrá disponer el *computador esencial* habrá que estimar el valor de la entropía máxima de dicho computador confinado en un volumen de un litro y de una masa de un kilogramo. Para realizar ese cálculo de forma precisa se necesitaría un conocimiento completo de la dinámica de las partículas elementales presentes en el sistema, si bien se puede obtener una estimación modelando el volumen ocupado por el computador como un conjunto de modos de partículas elementales con una energía media total igual a E. También se pueden obtener distintas aproximaciones para la entropía máxima considerando los estados de funcionamiento del computador, tal y como se puede ver en [12], donde se estiman valores de $b=2.13 \times 10^{31}$ *bits*. Este tamaño de memoria estimado para el *computador esencial* es muchísimo mayor que el que encontramos en los computadores portátiles actuales (del orden de los 10^{10} bits) debido a que los computadores reales actuales utilizan muchos grados de libertad para almacenar un bit, por razones de controlabilidad y estabilidad. En un *computador esencial* un bit solo necesitaría un grado de libertad, pero para conseguirlo se tendría que poder transformar toda la materia en energía y la memoria sería un plasma de millones de grados Kelvin.

Tras este análisis de los límites físicos de la computación [12-15], a continuación se presentarán las principales características de la computación cuántica, que se encuentra aún en lo que puede considerarse su infancia, pero para la que existe una clara demanda en el ámbito científico y tecnológico y por ser una alternativa para resolver problemas que no se pueden abordar con computadores clásicos, constituyendo una opción prometedora para el acercamiento a los límites físicos de la velocidad del computador.

6 La computación cuántica

El interés por la computación cuántica surge al considerar problemas cuya complejidad los hace prácticamente irresolubles mediante *algoritmos clásicos*, pero que pueden procesarse, en tiempos relativamente pequeños, a través de los denominados *algoritmos cuánticos*. Por ejemplo, la factorización de números con una cantidad moderadamente alta de dígitos es uno de los problemas complejos para los que la computación cuántica constituye una buena alternativa. Además, puesto que este algoritmo se utiliza comúnmente en el encriptado de información, la computación cuántica tendría una incidencia considerable en las aplicaciones de ciberseguridad.

En lugar de utilizar bits, los computadores cuánticos trabajan con los denominados *qubits* como unidades de información, y aprovechan las propiedades de superposición y entrelazado (*entanglement*) cuántico. A diferencia de lo que ocurre con un bit, que

puede estar únicamente en uno de sus dos posibles estados, 0 ó 1, un *qubit* puede estar en uno de sus dos *estados base ortogonales*, |0> y |1>, o en una superposición de estados base que se expresa como una combinación lineal de los mismos, $|\Psi> = a\ |0> + b\ |1>$, donde a y b son números complejos y se denominan amplitudes de probabilidad, que verifican $|a|^2+|b|^2=1$. La medida de un *qubit* proyecta el estado de dicho *qubit* sobre uno de los dos estados base |0> y |1> con probabilidades $|a|^2$ y $|b|^2$, respectivamente. Así, mientras que un sistema digital clásico con *n* bits puede encontrarse en uno de los posibles 2^n estados, en un sistema cuántico se tendría una superposición de esos 2^n estados, que se podrían almacenar, y con los que podría operar al mismo tiempo. Esta es, precisamente, la base del aumento de velocidad exponencial que podría proporcionar un computador cuántico con respecto a uno *clásico* digital.

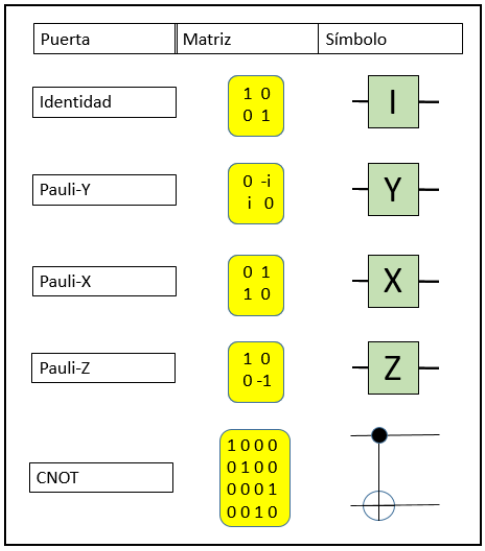

Figura 3. Algunas puertas cuánticas y sus matrices correspondientes [19].

Los algoritmos cuánticos se describen en términos de circuitos constituidos por puertas (cuánticas) que operan con *qubits*, y cuyo comportamiento se puede expresar a partir de matrices unitarias $2^n x 2^n$, para el caso de puertas que actúen sobre *n qubits*. La Figura 3 muestra algunas puertas cuánticas y sus matrices para el caso de dos *qubits* ([19]). En este caso se trata de matrices de 2x2. Por ejemplo, al aplicar la puerta Pauli-X sobre dos *qubits* |0> y |1> se obtiene $|\Psi_1> = 0\ |0> +1\ |1>$ y $|\Psi_2> = 1\ |0> + 0\ |1>$, ó $|\Psi_1> = |1>$ y $|\Psi_2> = |0>$. La puerta CNOT o NOT controlada de la Figura 3, es una puerta de 2 *qubits* (la matriz es 4x4) que realiza la operación X sobre el *qubit* de la entrada superior de la puerta, cuando el *qubit* de control (el de la entrada inferior) es |1>. Es posible encontrar conjuntos universales de puertas cuánticas, a partir de las que se puede implementar cualquier operación cuántica. La medida del estado de una puerta cuántica *colapsa* el estado cuántico de la puerta al valor que se obtiene tras la medida (valor clásico, no cuántico). Por tanto, no es posible determinar experimentalmente, con exactitud, el estado de superposición del *qubit.*

El funcionamiento de las tecnologías cuánticas actuales se ve afectado frecuentemente por errores debidos a la denominada *decoherencia*. Es decir, se producen pérdidas de información asociadas a la interacción del sistema de cómputo cuántico que ocasionan tasas de error de alrededor de 0.01 en las puertas cuánticas. Se trata de tasas bastante mayores que las tasas de 10^{-12} y 10^{-15} que se requieren en los algoritmos cuánticos complejos [16]. Esto significa que debe existir una capa que se encargue de implementar procedimientos para la corrección de errores cuánticos (Quantum Error Correction Code, QECC) y mecanismos de tolerancia a fallos. En los circuitos de corrección cuánticos (QEC) la información se protege mediante un QECC específico que codifica un *qubit* lógico a través de varios *qubits* físicos imperfectos. Además, el sistema se está monitorizando constantemente para detectar posibles errores y corregir su efecto. La codificación se lleva a cabo a través del entrelazado de *qubits* y la detección de errores mediante medidas de comprobación de paridad (ESM, *Error Syndrome Measurements*). Estas ESM permiten realizar la comprobación de paridad sin una medida directa de los *qubits*, por lo que mantiene la coherencia de los estados de dichos *qubit*s.

Uno de los QECC más populares es el código de superficie (*surface code*, SC) [17] que se puede implementar mediante una estructura 2D de *qubits* de dos tipos: *qubits de datos* (en los que se almacena la información) y ciertos *qubits* de apoyo denominados *qubits ancilla*, entre los que hay interacción únicamente entre *qubits* vecinos. Así, cada *qubit ancilla* interactúa con cuatro *qubits* de datos para realizar las comprobaciones de paridad entre los *qubits* de datos. A través de los *qubits* de datos y los *ancilla*, indicados anteriormente, se pueden discretizar e identificar los errores cuánticos continuos como errores de cambio de bit (X), o de cambio de fase (Z). Este proceso de identificación de los errores se implementa a través de circuitos de electrónica clásica.

Los computadores cuánticos, dado el uso que se proyecta para ellos, deben ser considerados de dominio específico, y presentan problemas no resueltos satisfactoriamente hasta el momento. Como se ha comentado anteriormente, uno de dichos problemas es la decoherencia producida por la interacción con el entorno no cuántico y que ocasiona errores de un factor de alrededor de 0.01 en las puertas cuánticas. Por lo tanto, los sistemas cuánticos deben incluir elementos de detección y corrección de errores cuya infraestructura debe escalarse con el aumento de *qubits*. Actualmente hay más de 50 algoritmos cuánticos básicos, de los que el que puede considerarse más representativo es el algoritmo de Short, que es el primero que se aplicó a problemas reales y requiere entre millones y miles de millones de *qubits* para factorizar números competitivamente grandes en relación con otros algoritmos de factorización. En la computación cuántica pueden combinarse varios *qubits* para generar un nuevo estado que es la composición de los 2^n estados posibles. El estado entrelazado del *qubit* no puede descomponerse en estados separados (*entanglement*), y cuando se aplica una operación cuántica sobre los *qubits* combinados, la operación se lleva a cabo sobre los 2^n estados posibles al mismo tiempo. La ganancia exponencial de velocidad se alcanzaría a través de la aplicación de un conjunto universal de operaciones cuánticas sobre *qubits* superpuestos y con entrelazado (*entanglement*).

Un computador cuántico incluye tanto elementos de computación clásica o convencional como componentes de computación cuántica. El desarrollo de un computador cuántico implica resolver retos importantes desde el punto de vista de la ingeniería de computadores, además del desarrollo de dispositivos cuánticos [18]. Desde la perspectiva de la Ingeniería de Computadores, los computadores cuánticos son computadores convencionales dotados de un coprocesador o unidad de tratamiento de datos cuántica. La interfaz del procesador cuántico con el exterior se hace a través de unidades del computador convencional, apoyándose también en unidades de este tipo para compilación de programas, almacenamiento permanente de programas y datos, etc.

De igual forma que para abordar el estudio de un computador digital convencional se recurre a una descripción del mismo a través de una serie de niveles superpuestos, en un computador cuántico también se definen una serie de capas [18, 19] que deben estar presentes y haberse desarrollado convenientemente. Estas capas van desde la de descripción de alto nivel del algoritmo cuántico (realizada mediante el correspondiente lenguaje de programación cuántico de alto nivel), hasta la capa de operaciones físicas del procesador cuántico. Las dos capas superiores incluyen los algoritmos y los compiladores que hay que desarrollar para explotar el hardware cuántico subyacente, que define la capa inferior, en la que se asume que los *qubits* y las operaciones cuánticas no sufren ningún tipo de error. En [18] se consideran distintas alternativas para implementar la capa inferior hardware en un computador cuántico. Sin embargo, en el caso de la computación cuántica, un circuito actúa sobre la superposición de los estados posibles de los *qubits*, y por ello, el talón de Aquiles de la computación cuántica es la fragilidad de los *qubits*, debido a la *decoherencia*. Como se ha comentado anteriormente, un computador cuántico debe incluir sistemas de corrección de errores cuánticos (QEC) y de tolerancia a fallos (FT). Concretamente, debajo de la capa de alto nivel, existe una capa de compilación que convierte los algoritmos cuánticos en su versión tolerante a fallos mediante el correspondiente código QEC y genera una serie de instrucciones incluidas dentro del repertorio de instrucciones cuántico (QISA), análogo al repertorio ISA en un computador digital. La infraestructura de compilación consta de un compilador que genera la lógica clásica y otro que produce los circuitos cuánticos. La capa QISA es la que separa el hardware y el software (el compilador traduce instrucciones lógicas a instrucciones físicas pertenecientes al conjunto QISA para el que el hardware cuántico proporciona soporte). Como ejemplo de instrucciones del conjunto QISA están la inicialización, las medidas de magnitudes, y otras puertas cuánticas, como la CNOT.

El bloque de ejecución cuántica (QEX) es el que se encarga de ejecutar las instrucciones QISA generadas por la infraestructura de compilación. La interfaz cuántico-clásica aplica las señales eléctricas sobre las que actúa el chip cuántico y es responsable de las conversiones entre el plano analógico de *qubits* y las capas digitales de la pila del sistema. La unidad de control cuántica (QCU) decodifica las instrucciones del QISA, y realiza las operaciones cuánticas, el control de realimentación, y el control de errores. La Figura 4 proporciona un esquema de la distribución de las capas mencionadas.

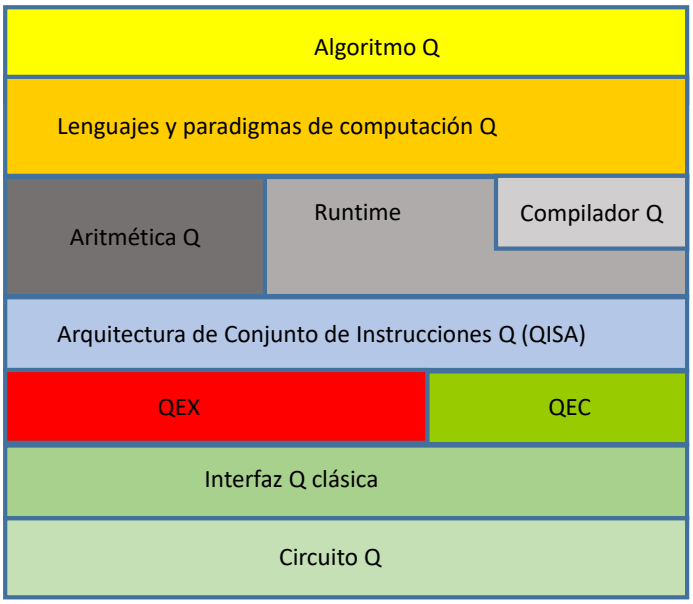

Figura 4. Organización de capas de un computador cuántico [18].

Uno de los retos que plantea el desarrollo de circuitos para los distintos módulos de los computadores cuánticos es el de aumentar el número de *qubits* por circuito integrado que pueden entrelazarse, al tiempo que se incrementa su tiempo de vida y su fidelidad, escalabilidad, y flexibilidad. La idea básica de las técnicas de control de errores cuánticos consiste en utilizar varios *qubits* imperfectos para componer unidades más fiables que se denominan *qubits* lógicos basados en códigos de corrección de error cuánticos específicos, como por ejemplo los denominados *surface codes*, en los que los *qubits* se organizan en una malla 2D regular que solo permite las interacciones más próximas y puede tolerar una tasa de error de hasta 0,01 (1%)

7 Una asignatura de Arquitectura de Computadores para 2020

En esta sección se propone un conjunto de contenidos esenciales para Arquitectura de Computadores, que deberían abordarse, con la extensión adecuada, en una o varias asignaturas del grado Ingeniería Informática y, por supuesto en grados o especialidades de Ingeniería de Computadores. A partir de esos contenidos se puede editar un libro de texto actualizado como referencia unificadora de la materia a impartir en cada una de las asignaturas relacionadas con Arquitectura de Computadores, muy útil en particular cuando se utilizan metodologías de docencia invertida. En dicho libro debería exponerse el cuerpo teórico aceptado para la materia, al tiempo que se reflejan los paradigmas del área de conocimiento surgidos de las prácticas usuales en el ámbito y de las líneas de investigación científica y tecnológica propias. No se propone una

distribución específica de los contenidos que se indican a continuación en asignaturas, ni tampoco se hacen estimaciones de tiempos de impartición dado que estos dependerían de la titulación de que se trate en cada caso y, por tanto, cambiarían según el contexto definido por los ámbitos y niveles de detalle que abordan el resto de asignaturas de la titulación considerada.

Para establecer los contenidos, se ha tenido en cuenta que, como se ha mostrado a lo largo del artículo, la ingeniería y la computación han experimentado cambios muy rápidos y, además existe la creencia de que se seguirá manteniendo esa tónica en el futuro, a velocidades incluso mayores. Por tanto, la formación de los ingenieros de computadores debe incluir los conocimientos fundamentales que les permitan adaptarse ágilmente a las tecnologías emergentes, e incluso ser capaces de identificar empresas y proyectos que, muy probablemente fracasarían por su dificultad para adaptarse a los inminentes cambios tecnológicos. Además, el ingeniero de computadores debe ser consciente de las consecuencias positivas y negativas del desarrollo de las tecnologías emergentes. De hecho, incluso en la introducción del área de conocimiento de *Arquitectura y Organización* en [20] se indica que los profesionales informáticos no pueden contemplar el computador como una caja negra que ejecuta programas por arte de magia, sino que también deben conocer el hardware en el que se basa la computación, sus componentes funcionales y los efectos de su interacción, para desarrollar programas con las prestaciones requeridas. También se debe disponer de un conocimiento exhaustivo de la interfaz que proporciona la arquitectura a las capas software superiores, y ser consciente de que la viabilidad de la ejecución de una aplicación en un computador no solo está relacionada con restricciones en el tiempo de ejecución y memoria, sino también con el consumo de energía.

En el libro de Arquitectura de Computadores de 2005 [1], el paralelismo se contempla como el paradigma que definía la evolución de la Arquitectura de Computadores (que traduce en capacidades reales y prestaciones, las posibilidades que ofrece la tecnología). El aprovechamiento de los distintos tipos y niveles de paralelismo sigue siendo una línea esencial para la mejora de prestaciones de las aplicaciones procesadas en un computador, y las alternativas según las que se contempla su uso cambian a medida que lo hacen la tecnología y las aplicaciones más demandadas. Por tanto, es importante disponer de procedimientos de optimización de aplicaciones en términos de módulos elementales, o de paralelismo heterogéneo. Además de estas consideraciones previas, al elaborar el contenido de un curso de Arquitectura de Computadores y su correspondiente libro de texto, también se deben tener en cuenta las últimas recomendaciones del IEEE y la ACM para el *Computer Engineering Curricula* de 2005 [20], y la guía docente de la asignatura en el grado de Ingeniería Informática de la UGR [21]. Igualmente, también es relevante revisar los contenidos de las ediciones más recientes de libros de texto de referencia en la materia, como [2]. Según esto, una propuesta de contenidos para Arquitectura de Computadores y, consecuentemente, de posibles capítulos a incluir en el correspondiente libro de texto es la siguiente:

1. Fundamentos cuantitativos de Arquitectura de Computadores.
2. Repertorios de instrucciones e Interfaces
3. Evaluación de prestaciones.

4. Programación paralela y jerarquía de memoria.
5. Procesadores con paralelismo entre instrucciones (ILP)
6. Procesadores con paralelismo a nivel de hebra (TLP)
7. Procesadores con paralelismo de datos (DLP): GPUs. Procesadores Vectoriales, Procesadores SIMD.
8. Computadores de dominio o uso específico
9. Tecnologías hardware y perspectivas de la evolución de los computadores

Para empezar, se abordarían los aspectos relacionados con la caracterización cuantitativa del computador, describiendo las magnitudes utilizadas para analizar sus prestaciones, tanto en cuanto a velocidad de procesamiento, como en potencia instantánea, o en energía consumida, e incluyendo información sobre los distintos tipos de computadores según las aplicaciones que ejecutan más eficientemente y los mercados que los demandan. También se abordaría la diferencia entre retardo o latencia y ancho de banda.

A continuación se introduciría el concepto de Arquitectura del Computador desde distintas perspectivas. Por un lado como repertorio de instrucciones máquina que definen una interfaz entre el hardware y el software del computador, o como diseño y organización del hardware para satisfacer los requisitos funcionales del computador.

El tercer capítulo presentaría conceptos relacionados con el procesamiento y la programación paralela, y describiría las magnitudes utilizadas para evaluar las mejoras que proporciona el paralelismo (ganancia de velocidad, eficiencia, etc.) junto con sus limitaciones. A través del estudio de las leyes de Amdahl y Gustafson se introducen límites para la ganancia de velocidad, que permiten estimar las mejoras alcanzables en función de los perfiles de paralelismo de las aplicaciones consideradas. Indudablemente, como se ha indicado más arriba, el paralelismo, sus alternativas, implementaciones, y formas de aprovechamiento, seguirán siendo parte esencial de las asignaturas de Ingeniería de Computadores. Así, independientemente de cómo se organicen los contenidos, deberían incluirse temas dedicados al paralelismo de instrucciones (ILP), de datos (DLP), y de hebras (TLP), y debería mantenerse el estudio los computadores paralelos MIMD considerando las estructuras de interconexión entre procesadores memorias y dispositivos de E/S, las clasificaciones de multiprocesadores y multicomputadores según tengan una jerarquía de memoria compartida o distribuida, y los conceptos de coherencia y consistencia de memoria junto con su relación con la sincronización y comunicación en el multiprocesador.

También es esencial el estudio de las interfaces presentes en el computador desde el punto de vista de la garantía de seguridad que ofrecen (por ejemplo frente a ataques como *Meltdown* o *Spectre*) dado el funcionamiento de las microarquitecturas actuales con paralelismo ILP, que incorporan, entre otros, recursos de procesamiento especulativo para la mejora de rendimiento de velocidad y consumo energético. Finalmente, el estudio de APIs que faciliten el aprovechamiento de arquitecturas heterogéneas resulta esencial, junto con el de las tecnologías físicas que constituirán la base de computadores cuya implementación empieza a considerarse en la actualidad. Un ejemplo lo tenemos en entornos como *OpenCL*, que permiten la programación

paralela independiente de la plataforma a través de programas que se procesan en un *host*, que envía funciones, denominadas *kernels* o núcleos, para que sean ejecutadas en otros dispositivos, definidos previamente como dispositivos *OpenCL* (*OpenCL devices*) que pueden implementar arquitecturas diferentes (CPU o GPU). De esta forma se pueden definir plataformas paralelas heterogéneas para las que deben abordarse problemas complejos de distribución de carga, no sólo para optimizar la ganancia de velocidad sino también el consumo de energía. No obstante, junto a estos entornos alternativos y algo más específicos, como el mencionado *OpenCL*, también existen otros, como *OpenMP* muy extendidos, y con implementaciones eficientes realizadas por, prácticamente, todos los fabricantes. Además, podría utilizarse en prácticamente todas las asignaturas de Arquitectura de Computadores.

Una aproximación que siempre se contempla para desarrollar computadores más eficientes que los disponibles en cada momento, es la de buscar nuevas tecnologías físicas que superen las limitaciones que se observan en las utilizadas y validadas en el presente. En [7] se describen algunos de los dispositivos alternativos a los de tecnología CMOS, y en [18, 19] se referencian tecnologías y dispositivos que se están considerando para conformar la capa de tecnología física en el diseño de computadores cuánticos. Dado que en el futuro es previsible que el desarrollo de dispositivos implementados en nuevas tecnologías físicas sea una alternativa muy frecuente a la hora de plantear nuevas mejoras en Ingeniería de Computadores, es conveniente mantener en los curricula, asignaturas dedicadas a las tecnologías físicas que se están considerando y a las que se prevé utilizar para conformar los niveles hardware de los computadores futuros.

8 Conclusiones

Al aproximarse el agotamiento de la ley de Moore han aparecido nuevos paradigmas para marcar la hoja de ruta del desarrollo de la Ingeniería de Computadores, y como consecuencia, se inducirán cambios tanto en los contenidos, como en su distribución entre las asignaturas de las nuevas titulaciones que habrá que plantear. En la mayoría de los casos, en lugar de definir conjuntos de programas de prueba que caractericen el uso más frecuente de un computador y usar magnitudes como las velocidades pico para comparar arquitecturas, se pueden utilizar directamente las aplicaciones reales consideradas, cuyos códigos implementan clasificadores y/o aproximadores universales como las redes neuronales profundas, y también metaheurísticas de optimización, como las basadas en algoritmos evolutivos. Si las aplicaciones de interés se implementan a partir de modelos neuronales, o metaheurísticas evolutivas, pueden ser bastante efectivas para obtener medidas de prestaciones realistas. Además, dado que poseen estructuras computacionales basadas en la interconexión de elementos de cómputo relativamente sencillos y parecidos entre sí, su uso contribuye a evitar el problema de la recurrencia en Ingeniería: gran parte de los procedimientos y alternativas de optimización que se han analizado y desarrollado para unas aplicaciones, se pueden aprovechar en otras diferentes para las que, sin embargo, también se utilizan aproximadores o clasificadores basados en redes neuronales profundas, o

metaheurísticas. No es casual el reciente interés, en el ámbito de la Ingeniería de Computadores, por el desarrollo de aplicaciones realistas a través de redes neuronales profundas y en su implementación eficiente a través de arquitecturas de uso específico que, por otra parte permiten un aprovechamiento eficiente del paralelismo implícito en la aplicación.

También despiertan un enorme interés aproximaciones, en principio prometedoras, con objeto de abordar problemas que caen fuera del alcance de la computación *clásica*. De hecho, en la Sección 6 de este artículo proporcionamos una aproximación a la computación cuántica. Por un lado, los computadores cuánticos vienen despertando un interés considerable debido a las posibilidades que ciertos algoritmos cuánticos ofrecerían en aplicaciones inabordables mediante algoritmos y computadores clásicos, gracias al paralelismo implícito que podría aprovecharse a través de conceptos de computación cuántica que, desde hace bastante tiempo se consideran una solución adecuada para superar las prestaciones que podrían proporcionar los computadores electrónicos debido a los límites que la minimización de los transistores impondría en su ritmo de mejora de prestaciones. Por otro lado, la integración de computadores cuánticos dentro del desarrollo de la ingeniería de computadores los sitúa más bien como coprocesadores para los que habrá que desarrollar interfaces que los ubiquen dentro de los computadores existentes. Nuestro propósito al referir aquí algunos detalles de la computación cuántica no es sugerir que se incluyan asignaturas dedicadas a la implementación de computadores cuánticos en los planes de estudios actuales para estudios de grado de Informática o Ingeniería de Computadores. Se ha buscado únicamente poner de manifiesto el interés de impartir contenidos relacionados con las tecnologías emergentes que puedan ser útiles para el desarrollo de computadores cada vez más potentes, al tiempo que, en asignaturas que abordan perspectivas de la Ingeniería de Computadores, se ilustra una vez más la importancia del estudio de interfaces para integrar y programar nuevas aplicaciones en los computadores al uso. Si bien es relevante abordar conceptos generales relacionados con la necesidad de interfaces eficientes y la integración de nuevas tecnologías para mantener niveles adecuados de mejora en las Arquitecturas de Computadores, el estado actual de desarrollo de los computadores cuánticos no aconseja la inclusión de asignaturas dedicadas a la computación cuántica en estudios de grado de Ingeniería de Computadores. Por esa misma razón, tampoco encontramos necesario introducir temas relacionados con los computadores cuánticos en un libro de texto sobre Arquitectura de Computadores. La perspectiva, y los problemas que se abordarían desde dicha perspectiva serían similares a los que habría que considerar a la hora de estudiar, en cada momento, otras aproximaciones al desarrollo de coprocesadores de dominio específico que ofrezcan prestaciones elevadas en ámbitos de aplicación muy demandados. Un ejemplo de esta situación lo constituye la computación neuromórfica, y el uso de coprocesadores para redes neuronales profundas, que están suscitando un interés considerable en la actualidad.

Agradecimientos. Este artículo ha sido financiado por el proyecto PGC2018 - 098813 – B – C - 31 (Ministerio de Economía y Competitividad" y fondos FEDER).

9 Referencias

1. Ortega, J.; Anguita, M.; Prieto, A.:"Arquitectura de Computadores". Ed. Thomson-Paraninfo, 2005.
2. Hennessy, J.L.; Patterson, D. A.: "Computer Architecture: A quantitative approach (Sixth Edition)". Morgan Kaufmann, 2019.
3. Jouppi, N.P. et al.:"In-Datacenter performance analysis of a Tensor Processing Unit". 44th Symp. On Computer Architecture (ISCA). Junio, 2017.
4. Chien, A.: "Computer Architecture: disruption from above". Communications of the ACM, Vol. 81, No.9, pp.5, Septiembre, 2018.
5. Hennessy, J; Patterson, D.:"2017 Turing Award Lecture" (ISCA 2018). https://www.acm.org/hennessy-patterson-turing-lecture.
6. Conte, T.M.; DeBenedictis, E.P.; Gargini, P. A.; Track, E.: "Rebooting Computing: The Road Ahead". Computer, pp. 20-29. Enero, 2017.
7. Bourianoff, G.; Brewer, J.E.; Cavin, R.; Hutch, J. A.; Zhirnov; V.:"Boolean Logic and alternative information-processing devices". IEEE Computer, pp. 38-46, 2008.
8. 21st Century Computer Architecture. A community whitepaper, https://cra.org/ccc/wp-content/uploads/sites/2/2015/05/21stcenturyarchitecturewhitepaper.pdf, 2012.
9. Lipp, M., et al.: "Meltdown", 2018.
10. Kocher, P., et al.:"Spectre attacks: exploiting speculative execution", 2018.
11. Díaz García, A.F.; et al. Consumo de energía y asignaturas de arquitectura y tecnología de computadores. Enseñanza y Aprendizaje de Ingeniería de Computadores, 7: 79-92 (2017). [http://hdl.handle.net/10481/47374]
12. Lloyd, S.:"Ultimate physical limits to computation". arXiv: quant-ph/9908043v3, 14 Feb 2000
13. Landauer, R.:"Irreversibility and heat generation in the computing process". IBM J., pp.183-191, 1961.
14. Plenio, M. B.; Vitelli, V.: "The physics of forgetting: Landauer's erasure principle and information theory". Contemporary Physics, Vol.42, No.1, pp.25-60, 2001.
15. Margolus, N.; Levitin, L.B.:"The maximum speed of dynamical evolution". Physica D: Nonlinear Phenomena 120 (1-2), pp.188-195, 1998.
16. Xu.; et al.:"A heterogeneous quantum computer architecture", ACM CF'16, pp.323-330, 2016.
17. Fowler, et al.:"Surface code: Towards practical large-scale quantum computation. Phys. Rev. A., 86(3):032324, 2012.
18. Jones, N.C., et al.: "Layered architecture for quantum computing". arXiv:10105022v3, 2012.
19. Almudever, C.G.; et al.:"The engineering challenges in Quantum Computing". IEEE 2017 Design, Automation and Test in Europe (DATE), pp.837-845. 2017.
20. IEEE/ACM Computer Engineering Curricula 2016: http://www.acm.org/education/curricula-recommendations.
21. Guías docentes del Grado en Ingeniería Informática de la Universidad de Granada: http://grados.ugr.es/informatica/pages/infoacademica/guias_docentes/guiasdocentes_curso_actual.