Sistemas Embebidos en FPGA

RICARDO CAYSSIALS

Apoyo en la web

```
a = r;
b = m;
p = 0;
low_m = 0;
for (i = 0, i < n, i++) begin
    case {b(1) low_m}
        "01":
            p = p + a
        "10":
            P = p - a
    endcase
    p >> 1
    b >> 1
end
result = p;
```

Alfaomega

marcombo
ediciones técnicas

Sistemas Embebidos en FPGA

Ricardo Cayssials

Sistemas Embebidos en FPGA

Ricardo Cayssials

Alfaomega

marcombo
ediciones técnicas

Sistemas embebidos en FPGA
Ricardo Cayssials

Primera edición: Buenos Aires: Alfaomega Grupo Editor Argentino, 2014

Primera edición: MARCOMBO, S.A. 2014

www.marcombo.com

Diseño de cubierta: Iris Biaggini

ISBN: 978-84-267-2158-7

D.L.: B-12628-2014
Impreso en
Printed in Spain

Mensaje del editor

Los conocimientos son esenciales en el desempeño profesional, sin ellos es imposible lograr las habilidades para competir laboralmente. La universidad o las instituciones de formación para el trabajo ofrecen la oportunidad de adquirir conocimientos que serán aprovechados más adelante en beneficio propio y de la sociedad; el avance de la ciencia y de la técnica hace necesario actualizar continuamente esos conocimientos. Cuando se toma la decisión de embarcarse en una vida profesional, se adquiere un compromiso de por vida: mantenerse al día en los conocimientos del área u oficio que se ha decidido desempeñar.

Alfaomega tiene por misión ofrecerles a estudiantes y profesionales conocimientos actualizados dentro de lineamientos pedagógicos que faciliten su utilización y permitan desarrollar las competencias requeridas por una profesión determinada. Alfaomega espera ser su compañera profesional en este viaje de por vida por el mundo del conocimiento.

Alfaomega hace uso de los medios impresos tradicionales en combinación con las tecnologías de la información y las comunicaciones (TIC) para facilitar el aprendizaje.

Libros como éste tienen su complemento en una página Web, en donde el alumno y su profesor encontrarán materiales adicionales.

Esta obra contiene numerosos gráficos, cuadros y otros recursos para despertar el interés del estudiante, y facilitarle la comprensión y apropiación del conocimiento. Cada capítulo se desarrolla con argumentos presentados en forma sencilla y estructurada claramente hacia los objetivos y metas propuestas.

Los libros de Alfaomega están diseñados para ser utilizados dentro de los procesos de enseñanza-aprendizaje, y pueden ser usados como textos para diversos cursos o como apoyo para reforzar el desarrollo profesional.

Alfaomega espera contribuir así a la formación y el desarrollo de profesionales exitosos para beneficio de la sociedad.

Acerca del autor

Ricardo Cayssials

Ricardo Cayssials es Ingeniero Electrónico, Doctor en Ingeniería y Profesor Titular. También es responsable de las cátedras que cubren “Diseño de Hardware Digital”, “Lenguajes de Descripción de Hardware y dispositivos FPGA” en la Universidad Nacional del Sur y en la Facultad Regional Bahía Blanca de la Universidad Tecnológica Nacional. Fue Profesor Visitante de la Universidad Federal Fluminense en Brasil e Investigador Visitante de la Universidad de York en Inglaterra.

Es responsable y Director de la Especialización de Posgrado Profesional en Tecnologías Digitales Configurables de la Universidad Nacional del Sur y Director de Proyectos de Investigación y Desarrollo en sistemas embebidos utilizando dispositivos FPGA.

Introducción

El desarrollo de sistemas embebidos modernos demanda la implementación de funciones sofisticadas en plazos de diseño cortos. Actualmente, es muy frecuente que la implementación de un sistema embebido, además de realizar las funciones específicas deseadas, requiera, por ejemplo, comunicación con dispositivos externos cableados e inalámbricos, dispositivos de visualización y de almacenamiento masivo, compatibilidad con diferentes formatos de datos, entre otros. Es por esta razón que la flexibilidad en el diseño e implementación debe ser considerada al momento de emprender el desarrollo de un sistema embebido.

Los dispositivos FPGA permiten la implementación de todo el hardware de un sistema digital en un circuito integrado configurable, permitiendo desarrollos conocidos como *Sistemas-en-Chip-Programable* (SOPC, *System-on-Programmable-Chip*, en inglés). La especificación de todo el sistema se realiza en forma flexible debido a que su diseño e implementación involucran herramientas de desarrollo que permiten un elevado nivel de abstracción.

Por otro lado, es necesario contar con un soporte de software que nos permita disponer de todas las funcionalidades que requieren los desarrollos modernos. Generalmente, este soporte está considerado y desarrollado en diversos sistemas operativos. Existen algunos que permiten ser ejecutados en diferentes plataformas de hardware y otros desarrollados para sistemas embebidos específicos. Sin embargo, la mayoría de ellos satisfacen el estándar POSIX lo que permite relativamente una migración sencilla de aplicaciones de software a diferentes plataformas de hardware.

Los fabricantes de dispositivos FPGA brindan diversas herramientas para la realización de SOPC y plataformas de desarrollos (*kits*) que incrementen la eficiencia y flexibilidad de diseño que utilicen sus dispositivos.

En este libro, se introducen los conceptos y metodologías necesarios para la realización de Sistemas Embebidos en dispositivos FPGA. Las metodologías de diseño, implementación y desarrollo estarán realizadas utilizando las plataformas y dispositivos de Altera, pero los fundamentos conceptuales permitirán su utilización en plataformas y dispositivos de otros fabricantes.

Dr. Ing. Ricardo Cayssials

Antes de comenzar a leer

En este libro, se utiliza la tipografía `courier new` en los casos en que se hace referencia a código o acciones que se han de realizar en la computadora, ya sea en un ejemplo o cuando se refiere a alguna función mencionada en el texto. También se utiliza para indicar teclas o direcciones URL.

El texto **resaltado** es utilizado para indicar acciones que deben ejecutarse mediante la selección de comandos de la barra de menús o para indicar el nombre de componentes.

Los términos o definiciones cuyos significados están muy asociados al inglés se expresan en dicho idioma en *cursiva*. Por ejemplo, el archivo *netlist* para indicar el archivo que posee la descripción del sistema como interconexión de componentes.

Podrá acceder al código fuente y a los proyectos que se desarrollan en esta obra desde la página Web: http://libroweb.alfaomega.com.mx

TABLA DE CONTENIDOS

CAPÍTULO 1
DISEÑO DE SISTEMAS EMBEBIDOS

INTRODUCCIÓN

Existen diferentes definiciones de *sistemas embebidos* (*Embedded Systems*, en inglés), pero, en general, se define en términos de lo que pueden o no pueden realizar o de las aplicaciones en las cuales pueden ser utilizados.

> *"Un sistema embebido es un sistema basado en microprocesador que es construido para controlar una o varias funciones y que no es diseñado para que sea programado por el usuario final en la misma manera que, por ejemplo, lo es una Computadora Personal* ((1))".

Generalmente, el usuario puede realizar cambios en la configuración de la funcionalidad del sistema embebido, pero no lo puede efectuar mediante el cambio o remplazo del software del mismo. Tal tarea de programar y realizar el software del sistema embebido está a cargo del diseñador/programador quién evaluará las diferentes alternativas de configuración que estarán disponibles para el usuario final.

Un sistema embebido se contrapone en funcionalidad con una Computadora Personal, en la cual el usuario final puede seleccionar el software que desea ejecutar para modificar la funcionalidad de la misma. De esta manera, una Computadora Personal puede utilizarse como, por ejemplo, un procesador de texto, una base de datos, una consola de videojuegos, etc. Por supuesto que un diseñador podría configurar una Computadora Personal para una determinada aplicación y ejecutar un software específico para tal tarea, dejando al usuario final las opciones de solo configurar tal aplicación. En este último caso, se trataría, por su aplicación, de un sistema embebido por más que se utilice una Computadora Personal para su implementación.

Por lo visto, un sistema embebido generalmente queda definido por las aplicaciones a la que está destinado y, por consiguiente, define metodologías de diseño e implementación específicas.

Actualmente, las áreas de aplicación de sistemas basados en microprocesadores son de las más variadas y, en consecuencia, las de utilización de sistemas embebidos. Automóviles, electrodomésticos, componentes industriales, equipamiento para la comunicaciones de voz y datos, instrumental de medicina, consolas de entretenimiento, dispositivos de audio y video son solo algunas de las áreas en donde se aplican los sistemas embebidos.

La utilización de sistemas embebidos fue originada por la flexibilidad que provee un sistema basado en microprocesador para adaptarse a diferentes aplicaciones. De esta manera, se pudo remplazar diseños electrónicos, mecánicos o eléctricos específicos por sistemas programables que, sin mayores cambios que el software, pudieron ser utilizados tanto para el control de un ascensor como para el de un horno de microondas o un control de acceso de personal, por nombrar solo algunos ejemplos.

Es de esta forma que aparecieron diferentes microcontroladores que permitían al diseñador pequeños sistemas embebidos en muy pocos pasos. Los microcontroladores de Microchip (de la familia PIC), de AVR y de Motorola abarcaron muchas aplicaciones. Posteriormente, la función que se requiere de un sistema embebido no solo se restringe a las tareas que debe cumplir según la aplicación, sino que es casi obligatorio incluir más funciones para satisfacer las exigencias de los usuarios. Por lo tanto, ya no es suficiente que una máquina fotográfica saque fotos, sino que es necesario que: almacene las fotos en una memoria compatible con una Computadora Personal, transfiera los archivos de fotos y video mediante un cable USB, detecte la sonrisa del objetivo, etc. Este tipo de exigencias en el diseño de sistemas embebidos requieren recursos de hardware y software adecuados para satisfacer los requerimientos de diseño modernos. En este sentido, las plataformas de lógica programable y el soporte de sistemas operativos permiten abarcar estas nuevas exigencias en el desarrollo de sistemas embebidos actuales.

SISTEMAS EMBEBIDOS EN LÓGICA PROGRAMABLE

Los sistemas embebidos están en continuo desarrollo, ofreciendo nuevas posibilidades día a día mediante la incorporación de nuevos dispositivos y tecnologías. Existen diferentes plataformas para la implementación de estos sistemas. Muchos de ellos poseen plataformas de desarrollo en las cuales se pueden implementar todo el software necesario para un hardware específico. Dichas plataformas presentan una solución para el desarrollador que generalmente abarca un determinado grupo de aplicaciones (automatización, video, procesamiento de señales, etc.). Por lo general, el soporte de sistemas operativos es restringido a las particularidades del hardware del sistema embebido y muy difícilmente puedan ser utilizadas en otros sistemas o aplicaciones.

Los dispositivos de lógica programable modernos, especialmente los dispositivos FPGA (*Field Programmable Gate Array*, en inglés), permiten la implementación de diferentes arquitecturas de procesamiento en hardware. El diseño y la implementación se pueden realizar mediante herramientas de desarrollo que permiten una gran flexibilidad al momento de adaptar las arquitecturas de hardware a las particularidades propias de una determinada aplicación. La capacidad de las FPGA modernas para la implementación de hardware permite la implementación de todo el sistema digital en un solo chip, lo que se denomina *Sistema-en-Chip-Programable* (SOPC, *System-on-Programmable-Chip*, en inglés). Esta flexibilidad en la

implementación del hardware posibilita cubrir una gran diversidad de aplicaciones, pero debe ser soportada al mismo tiempo por una "capa de abstracción de hardware" (*Hardware Abstraction Layer*, en inglés) que pueda ser adaptada fácilmente a las particularidades del software de aplicación. Es en este sentido que debe tenerse un conocimiento adecuado tanto del hardware como del software del sistema para permitir un diseño, desarrollo e implementación adecuada.

DESAFÍOS DEL DISEÑO DIGITAL

Las características propias de los sistemas embebidos han impuesto el desarrollo de filosofías de diseño flexibles y eficientes. El desafío de diseño digital ha sido tratado por diferentes autores.

Muchas de las filosofías de diseño consideran a todo el proceso como un procedimiento que posee un comienzo y un final en el que se entrega un producto terminado. Todo este proceso se define como "ciclo de vida del diseño" (*Design Life Cycle*). Existen diferentes maneras de implementarlo, todas ellas pudiendo producir el mismo resultado.

Una de las metodologías más tradicionales para la realización de diseños digitales es el "modelo de rebalse" (*Waterfall Model*, en inglés). La industria de software, debido a sus características específicas, ha adoptado metodologías de diseño más flexibles como ser el "modelo de prototipo" (*Prototype Model*, en inglés). Actualmente, el diseño de hardware digital basado en dispositivos digitales configurables (CPLD y FPGA) ha permitido incorporar estas metodologías de diseño puesto que poseen características similares a las del diseño de software.

Puede decirse que, independientemente del modelo de diseño seleccionado, el manejo de un proyecto debe comprender la revisión de cada etapa del proceso de diseño antes de continuar con la siguiente. Por esta razón, a medida que el proyecto avanza, existe más compromiso de recursos financieros y humanos afectados al mismo y las fallas introducidas en etapas tempranas del modelo repercuten adversamente en las siguientes.

MODELO DE REBALSE (WATERFALL MODEL)

Este modelo es un desarrollo tradicional en la industria electrónica y de hardware. Se basa en un proceso secuencial en el que cada etapa produce resultados para el comienzo de la siguiente en manera análoga al llenado de recipientes mediante el rebalse de un fluido. Este modelo (2) tiene sus orígenes en las industrias de manufactura y construcción, las cuales están fuertemente restringidas, por los costos o posibilidades, a efectuar modificaciones una vez que el producto ha sido realizado.

La Figura 1-1 muestra la representación típica del "modelo de rebalse" simulando una cascada de agua entre las diferentes etapas del proceso de diseño.

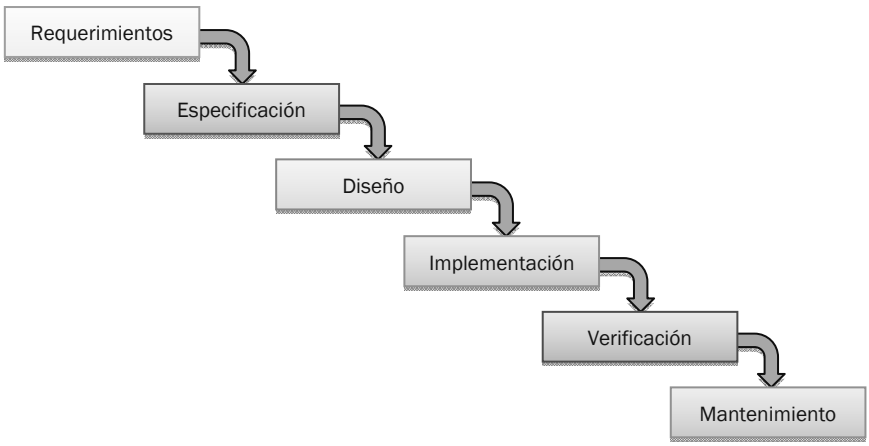

Figura 1-1: Ciclo de vida del diseño del modelo de rebalse (*Waterfall Model*).

Las etapas que generalmente se consideran en este modelo son:

- **Requerimientos**: en esta etapa, se describen los requerimientos que deberá satisfacer el producto final del diseño. Esta etapa podrá comenzar con una descripción abstracta y general para finalizar en una descripción detallada de los requerimientos que debe satisfacer el producto final del diseño. La etapa de requerimientos tiene una muy fuerte interacción con el usuario del producto final.
- **Especificación**: a diferencia de la etapa de requerimientos, esta etapa tienen una muy fuerte interacción con el desarrollador. La mayoría de las especificaciones puede provenir de la determinación técnica para la resolución de los requerimientos, pero otras pueden proceder de las alternativas técnicas introducidas por un preplanteo para su implementación. Es posible también especificar métricas que permitan verificar la correcta implementación del producto.
- **Diseño**: si bien el diseño abarca todas las etapas descriptas, esta es la etapa específica en la que el diseño es formalizado. La división entre módulos, la selección de las técnicas y metodologías, la adopción de metodologías y las estrategias de planificación y demás detalles del diseño son detalladas en esta etapa.
- **Implementación**: en esta etapa, se comienza a desarrollar todos los componentes que constituirán el diseño y se integran para su correcto funcionamiento. Siguiendo el detalle realizado en la etapa de diseño se llega al resultado final que cumple con los requisitos y especificaciones.
- **Verificación**: en esta etapa, se comprueba que la implementación cumple con los requerimientos de acuerdo a las métricas especificadas para tal fin. En caso

de no satisfacerlas, podrá ser necesario realizar nuevamente la implementación, el diseño, la especificación o los requerimientos.

- **Mantenimiento**: generalmente, luego de la verificación el producto está listo para ser entregado al usuario o para ser producido masivamente. En muchas cadenas de manufactura o de construcción, hay pocas alternativas de diseño para realizar posteriormente. En diseño de software o basados en FPGA, la etapa de mantenimiento permite la corrección de errores o fallas de diseño que no fueron detectadas anteriormente.

El modelo de rebalse estructura todo el proceso de diseño, forzando a todas las personas intervinientes a definir completamente cada una de las etapas. Esto permite una interacción más eficiente entre todos los actores del proceso y reduce el riesgo de que el producto final no cumpla con las expectativas del usuario.

Con el avance de la industria del software, este modelo comenzó a recibir algunas críticas por su metodología. La principal de ellas es que, en el proceso de diseño de software, el usuario puede no tener una idea acabada del producto final al inicio del diseño y que esto requiera de una interacción permanente con el diseñador durante cada una de las etapas del proceso. Por otro lado, debido a las características en el desarrollo de software, es muy frecuente que el diseño no finalice solamente en el mantenimiento del producto final, sino que sea permanente un rediseño del mismo para producir nuevas versiones del producto que cumpla con nuevos requerimientos.

La realización de hardware digital mediante dispositivos CPLD y FPGA, ha permitido que el diseño se realice mediante la utilización de Lenguajes de Descripción de Hardware para la implementación de productos de Propiedad Intelectual (IP de *Intellectual Property* en inglés) en forma similar al desarrollo de software. Por este motivo, metodologías utilizadas en el desarrollo de software, y que permiten interacción entre las diferentes etapas del proceso de diseño, puedan ser aplicadas para el desarrollo de hardware digital en dispositivos digitales configurables.

MODELO DE PROTOTIPO (PROTOTYPE MODEL)

El concepto principal del modelo de prototipo (3) es permitir una interacción durante todo el proceso de diseño entre el usuario y el diseño en la especificación del producto. Como ocurre en el desarrollo de software, el cliente puede establecer determinados requerimientos que sería conveniente para el diseñador comprobar su posibilidad de implementación mediante prototipos. De esta manera, el cliente y el diseñador podrían interactuar para redefinir las especificaciones de acuerdo a las posibilidades tecnológicas con las que cuentan.

Asimismo, las diversas iteraciones producirán el diseño de varios prototipos. Puede llegar el momento en que se considere que el prototipo está listo para ser calificado como producto terminado y ser entregado, al mismo tiempo que las tareas de rediseño en otros prototipos continúa como trabajo interno de la compañía. Este proceso es muy frecuente en software en donde un producto posee diversas versiones.

La realización de un prototipo permite mejorar el conocimiento sobre las capacidades de implementación, los alcances de las funcionalidades y el refinamiento de los requerimientos

que pueden establecerse para el diseño final. La Figura 1-2 muestra una metodología de implementación del "ciclo de vida del diseño" de un modelo de prototipo en la que puede observarse que los requerimientos y especificación son refinados mediante el desarrollo de prototipos ligeros que permiten una interacción fluida entre el usuario y el diseñador. Este refinamiento puede continuarse aun cuando se decidiera que un determinado prototipo estuviera listo para su implementación, permitiendo el mejoramiento de futuras versiones del producto.

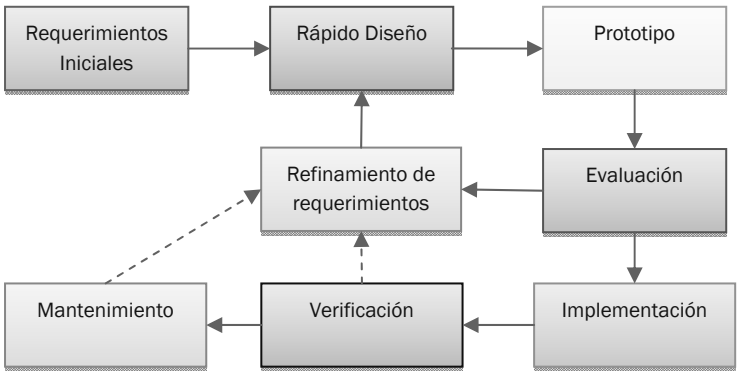

Figura 1-2: Ciclo de vida del diseño del modelo de prototipo (*Prototype Model*).

El desarrollo de un prototipo puede involucrar solamente alguna implementación parcial del diseño con el fin de evaluar la factibilidad o alcances que se puede lograr y redefinir los requerimientos y especificaciones. Algunos de los beneficios que pueden lograrse mediante este tipo de prototipos preliminares son:

- Análisis del alcance del diseño, permitiendo redefinir dicho alcance de acuerdo a la capacidad y complejidad de cada implementación.
- Demostración del funcionamiento general del sistema, permitiendo al desarrollador y al usuario obtener conocimientos para determinar las mejores alternativas para su implementación.
- Evaluación preliminar de los costos del producto y tiempo de desarrollo, dado que los cambios en etapas más avanzadas del proceso involucran costos y tiempos exponencialmente mayores.
- Obtención de volúmenes de información y procesamiento a considerar en el proyecto, para determinar las plataformas necesarias para un manejo adecuado y eficiente.

La utilización de dispositivos digitales configurables, como CPLD y FPGA, ha permitido la utilización del modelo de prototipo y otras metodologías de diseño de software al desarrollo de sistemas embebidos en dispositivos FPGA, mejorando la eficiencia y flexibilidad de los

procesos de diseño que se adapten a los cambios del ambiente que impone el mercado actual.

MÉTRICAS DE DISEÑO

Desde sus comienzos, las comparaciones entre los diferentes sistemas digitales de procesamiento fueron realizadas basadas en su velocidad: en cuán rápido podían ejecutar una determinada aplicación o software. A medida que estos sistemas fueron siendo utilizados para la implementación de sistemas embebidos, la velocidad pasó a ser solo una más de las métricas de comparación y evaluación de la eficacia y eficiencia de un determinado diseño.

Los requerimientos y las especificaciones detallan las condiciones que el diseño debe satisfacer, mientras que las **métricas** son las medidas que determinan si el producto final efectivamente satisface dichos requerimientos y especificaciones. En otras palabras, mientras que los requerimientos y especificaciones guían el proceso de diseño, las métricas verifican su cumplimiento.

Existe otro tipo de métricas que determina un grado de eficiencia de la implementación y que puede servir como criterio de comparación. Por ejemplo, si el diseño debe satisfacer una determinada velocidad de procesamiento, una implementación más veloz será más eficiente, según esta métrica, que otra que solo la cumple estrictamente. Por ejemplo, un diseño portable que pesa 300 gramos será considerado mejor que el que pesa 600 gramos. Sin embargo, esta mejor métrica de peso debe ser considerada conjuntamente con otras métricas de diseño, como por ejemplo el costo que ello implica.

Existen diversas métricas que pueden ser tenidas en cuenta en el diseño de sistemas embebidos. A continuación, se detallan solo algunas métricas de diseño cuantificables técnicamente como son: velocidad, energía, tamaño y costo.

VELOCIDAD

Gran parte de la literatura de sistemas de procesamiento basan la medición de la eficiencia de un sistema en su velocidad de procesamiento. Cuanto más rápido sea el procesamiento de un sistema, mayor su eficiencia. De esta manera, se proponen diferentes arquitecturas para incrementar la velocidad promedio en la ejecución de un determinado software. Este "criterio de eficiencia" está basado en que, generalmente, si un procesador es "más rápido" que lo requerido el sistema funcionará correctamente, mientras que si es "más lento" degradará la funcionalidad del sistema, pudiendo traer consecuencias inesperadas en la aplicación.

Características como coprocesamiento, memorias "caché", *branch prediction*, etapas de *pipeline* son algunas de las metodologías y mecanismos propuestos para incrementar la velocidad de procesamiento de los procesadores modernos. En la mayoría de los casos, el incremento de la velocidad de procesamiento en un sistema embebido podría demandar un incremento en el costo unitario del producto. Por otro lado, la utilización de un procesador más veloz podría simplificar el desarrollo del software y, en consecuencia, el costo de diseño,

pero podría demandar un mayor consumo de energía. Todas las influencias deben ser evaluadas para determinar la conveniencia en la selección de la velocidad de procesamiento.

La velocidad de procesamiento de un sistema embebido no se restringe solo a la frecuencia de reloj con que este funciona. Existen diversos factores que intervienen en la velocidad final de un determinado sistema como ser la diagramación de las tareas en tiempo-real y la utilización de los recursos de hardware para la ejecución del software.

ENERGÍA

Todos los sistemas requieren energía para su funcionamiento. En especial, todos los sistemas digitales modernos están construidos con componentes electrónicos que requieren energía eléctrica para su funcionamiento. En sistemas embebidos alimentados por baterías, puede ser necesario reducir el consumo de energía para incrementar la autonomía de los mismos.

Desde el punto de vista energético, puede verse un circuito digital como un sistema que transforma energía eléctrica en calor ya que toda esta energía que consume el circuito es disipada en forma de calor. El control del calor generado puede requerir la utilización de disipadores o sistemas de refrigeración forzada que pueden influir negativamente en la confiabilidad, tamaño o costos del diseño.

La disipación de energía, y por consiguiente el calor generado, está asociada con la **potencia** consumida por el dispositivo que es definida por la energía requerida en un determinado periodo de tiempo. Eléctricamente, la potencia (*power*, en inglés) es la tasa a la cual la **energía es transferida al circuito** eléctrico. La unidad de potencia es *watt* que equivale a un **Joule por segundo**. En un circuito digital, la energía consumida es proporcional al calor disipado al medio ambiente. Es necesario aplicar técnicas de disipación del calor generado para evitar que el circuito incremente su temperatura a valores perjudiciales para el sistema o el medio ambiente que lo rodea.

Otro factor importante es determinar el perfil del consumo de potencia, siendo conveniente, generalmente, tender a un consumo constante de la potencia evitando picos de consumo. En circuitos alimentados con baterías, los picos de potencia pueden acortar considerablemente la duración de la carga de la mismas afectando perjudicialmente la autonomía del sistema.

TAMAÑO

En sistemas embebidos modernos, el tamaño de los mismos es un requerimiento de diseño establecido por la portabilidad o espacio definido para su implementación. Eventualmente, el requerimiento de tamaño podrá limitar la velocidad del procesador para evitar un consumo que requiera un sistema de disipación voluminoso o un tamaño de baterías inadecuado.

Por otro lado, restricciones de tamaño podrían limitar la modularidad de la implementación que eventualmente imposibilitarían adaptaciones posteriores, su confiabilidad o dificultarían o afectarían las etapas de mantenimiento o la vida útil del proyecto.

COSTO

Las métricas anteriores definen la "eficiencia técnica" del diseño. En oposición a estas métricas, el costo determina la factibilidad económica o comercial del diseño. Existen diferentes costos que deben considerarse en el desarrollo de un determinado producto:

- Costo de ingeniería no retornable: es el costo que involucra el diseño y lograr un producto listo para producción. Principalmente, este costo debe considerar los costos del personal involucrado en el diseño y el de la infraestructura necesaria para su realización.
- Costo de fabricación: es el costo de fabricar el producto para su comercialización. Este costo debe considerar el costo de los componentes que conforman el producto y los costos involucrados en su manufactura.
- Costo de comercialización: es el costo económico y financiero requerido para la distribución y venta del producto y la administración del proceso de comercialización.

Todos estos costos deben ser considerados en el "costo unitario" final del producto. Los costos de ingeniería no retornable y los de comercialización debieran ser prorrateados en el cálculo del costo unitario de acuerdo a la cantidad de productos fabricados.

Los dispositivos digitales configurables permiten disminuir el tiempo necesario del ciclo de vida del diseño y consecuentemente el costo de ingeniería no tetornable. La reducción de estos costos es importante cuando la cantidad de productos fabricados es limitada y cuando el tiempo para la entrada en el mercado es reducido.

OTRAS MÉTRICAS DE DISEÑO

En los sistemas embebidos, en los cuales el diseño involucra componentes de software como hardware, puede resultar difícil o imposible obtener una implementación que optimice cada una de las métricas de diseño especificadas. Si bien las detalladas anteriormente pueden resultar las más frecuentemente especificadas, existen diversas métricas de diseño, algunas de ellas de compleja cuantificación, entre las que se pueden listar:

- Tiempo al prototipo (*Time-to-prototype*, en inglés): es el tiempo requerido para construir una versión del sistema que pueda ser utilizada para verificar su funcionalidad.
- Tiempo a la comercialización (*Time-to-market*, en inglés): es el tiempo requerido para diseñar y manufacturar el sistema hasta el momento en que se encuentre en condiciones de que pueda ser comercializado a los consumidores.
- Conformidad (*Correctness*, en inglés): es el grado de satisfacción de que el sistema cumple con las especificaciones de diseño.
- Mantenimiento (*Maintainability*, en inglés): es la posibilidad de realizar modificaciones o mejoras al producto luego de su comercialización.

- Seguridad (*Safety*, en inglés): es generalmente expresado por la probabilidad de que una falla en el sistema no causará daño.
- Violabilidad (*Security*, en inglés): es el grado de protección contra sabotajes que posee el producto, el desarrollo o su diseño.
- Confiabilidad (*Reliability*, en inglés): es la evaluación de que el sistema realizará las funcionalidades para las que ha sido diseñado, durante un determinado periodo de tiempo y bajo condiciones establecidas.

El desarrollo de sistemas embebidos basados en dispositivos FPGA y la aplicación de metodologías flexibles de diseño pueden contribuir a mejorar significativamente estas métricas, aun cuando la implementación final se realice en otro tipo de tecnología.

CONCLUSIONES

Los dispositivos digitales configurables, como son los dispositivos CPLD y FPGA, han permitido la utilización de metodologías más eficientes en el diseño de sistemas embebidos. Mientras que en un principio el hardware digital debía, por su naturaleza, ser manufacturado, actualmente puede ser descripto y concebido como una propiedad intelectual factible de ser conceptualmente rediseñada. Bajo el punto de vista de diseño, no existen mayores diferencias entre el software referido al programa ejecutado por el procesador del sistema y el software referido al código que permite configurar el hardware de un dispositivo digital configurable. Los dispositivos CPLD y FPGA permiten la implementación de hardware físico mediante un código que puede ser concebido en ciclos de vida de diseño con metodologías solo aplicables anteriormente al diseño de software.

CAPÍTULO 2
LÓGICA PROGRAMABLE Y LENGUAJES DE DESCRIPCIÓN DE HARDWARE

CIRCUITOS DIGITALES CON LÓGICA PROGRAMABLE

Los dispositivos lógicos programables pueden definirse, en principio, como circuitos integrados que permiten la implementación de circuitos digitales. Los circuitos digitales electrónicos son ampliamente utilizados en diferentes áreas (procesamiento de señales, automatización, entretenimiento, comunicaciones, medicina, entre otras).

Conceptualmente, se puede afirmar que cualquier circuito digital puede resumirse en un circuito **combinacional** o un circuito **secuencial**. En este capítulo, se describen los principales conceptos de estos diferentes tipos de circuitos digitales y las capacidades que brindan los diferentes tipos de dispositivos de lógica programable (CPLD y FPGA). Al final del capítulo, se introducen los lenguajes de descripción de hardware como alternativa de representación de circuitos digitales complejos.

CIRCUITO COMBINACIONAL

Un circuito combinacional es aquel en el que la salida depende solo de la entrada actual. La Figura 2-1 muestra un diagrama de bloques el cual representa a la lógica combinacional como una “caja negra” en la cual el valor de las salidas depende de los valores que poseen las entradas.

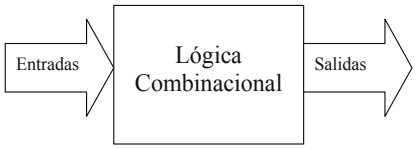

Figura 2-1: Diagrama de bloques de circuito combinacional.

Un circuito combinacional puede describirse mediante la especificación en una tabla del valor de la salida para cada una de las combinaciones de las entradas. Este tipo de tabla, que representa el funcionamiento de un circuito combinacional, se denomina **tabla de verdad**.

Por ejemplo, la expresión de una lógica combinacional de, por ejemplo, 3 entradas (**A**, **B** y **C**) y 2 salidas (**X** y **Z**), mediante una tabla de verdad, es de la forma:

TABLA 2-1: CIRCUITO COMBINACIONAL REPRESENTADO POR TABLA DE VERDAD

Entradas			Salidas	
A	**B**	**C**	**X**	**Z**
0	0	0	1	0
0	0	1	1	0
0	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	0	1	0	0
1	1	0	1	1
1	1	1	1	1

El circuito combinacional representado por una tabla de verdad puede ser implementado mediante compuertas electrónicas digitales que implementen funciones lógicas (compuertas NOT, AND, OR, NAND, etc.). Existen diferentes métodos para obtener las funciones lógicas de un circuito combinacional a partir de su especificación.

CIRCUITO SECUENCIAL

Un circuito secuencial es aquel cuya salida es función de la *secuencia* en la que se fueron dando las entradas. Un ejemplo de esto es un sistema de teclado de una alarma. En este tipo de circuito, se debe ingresar un determinado número para desactivar la alarma. Si los números son ingresados, pero en diferente orden, la alarma no se desactivará. Por otro lado, si los números son ingresados en la *secuencia* correcta, la alarma se desactivará.

En los circuitos secuenciales, no es práctico, y generalmente imposible, describir el funcionamiento del circuito mediante una tabla que enumere todas las secuencias posibles de las entradas. Es por ello que se utiliza el **estado** para expresar la condición en la que está