ISSN 1688-2784

UNIVERSIDAD DE LA REPÚBLICA
FACULTAD DE INGENIERÍA

Técnicas de bajo consumo en FPGAs

TESIS PRESENTADA A LA FACULTAD DE INGENIERÍA DE LA UNIVERSIDAD DE LA REPÚBLICA POR

Juan Pablo Oliver

EN CUMPLIMIENTO PARCIAL DE LOS REQUERIMIENTOS
PARA LA OBTENCIÓN DEL TÍTULO DE
DOCTOR EN INGENIERÍA ELÉCTRICA

DIRECTOR DE TESIS
Eduardo Boemo ..... Universidad Autónoma de Madrid

TRIBUNAL
Valentín Obac Roda .... Universidade Federal do Rio Grande do Norte
Luciano Volcan Agostini ..... Universidade Federal de Pelotas
Gregory Randall ..... Universidad de la República

DIRECTOR ACADÉMICO
Fernando Silveira ..... Universidad de la República

Montevideo,
31 de octubre de 2014

Técnicas de bajo consumo en FPGAs, Juan Pablo Oliver.

ISSN 1688-2784

A Juanito y Adela,
María y Agustín

Resumen

Todo diseño electrónico tiene tres restricciones principales que son área, velocidad y consumo. De las tres, el consumo es la variable más complicada de manejar para un diseñador: tiene incertidumbres, es difícil de estimar, y depende de varios parámetros, algunos tan dispares como el funcionamiento del sistema o los datos que ingresan al mismo. Pero por otro lado el consumo se está volviendo la restricción más importante para un gran número de aplicaciones.

En esta tesis se presentan una serie de herramientas y metodologías para poder manejar adecuadamente la variable consumo en FPGAs. Se trabaja con un enfoque fuertemente experimental y desde el punto de vista de un usuario de este tipo de dispositivos, validando los resultados con más de 350 experimentos.

Uno de los aportes de esta tesis, es una metodología completa de medición de consumo para FPGAs, que permite la calibración de herramientas de estimación. Como parte de esta metodología se incluyen los circuitos eléctricos necesarios para realizar las medidas y un conjunto de diseños o *benchmarks* para realizar pruebas incluyendo generadores de vectores de entradas. Se desarrolla además una herramienta específica de automedida de consumo para FPGAs. La misma tiene varias aplicaciones que permitirán ampliar y mejorar los experimentos de bajo consumo en futuras investigaciones, usando recursos muy sencillos y de muy bajo costo.

Se presentan una serie de experimentos con varias técnicas de reducción de consumo en FPGAs, y se cuantifican los resultados obtenidos con cada una de ellas. Finalmente se concluye con un caso de estudio, la reducción de consumo de un circuito en particular: el microcontrolador openMSP430

Abstract

Every electronic design has three main restrictions that are: area, speed and power consumption. Among the three, power consumption is the most difficult to handle by a designer, it has uncertainties, it is difficult to estimate, and it depends on several parameters, some as diverse as the operation of the system or the data dependency. But, on the other hand, power consumption is becoming the most important constraint for a large number of applications.

A number of tools and methodologies to properly handle the power consumption in FPGAs are presented in this thesis. The work was done with a strongly experimental approach and from the point of view of a user of these kind of devices, the results were validated with more than 350 experiments.

One of the contributions of this thesis, is a complete power measurement methodology for FPGAs, which allows calibration of estimation tools. As part of this methodology electrical circuits necessary to take measurements and a set of designs or benchmarks to test vector generators are included. Additionally a specific tool for self-measurement of consumption FPGAs was developed. This tool, which also uses very simple and very low cost resources, has potential applications that will allow to expand and improve the low-energy experiments in future research.

A series of experiments with various techniques to reduce power consumption in FPGAs were conducted, and the results obtained with each of them were quantified. Finally it is concluded with a case study, the reduction in power consumption of a particular circuit: the microcontroller openMSP430.

Agradecimientos

En primer lugar quiero agradecer a Eduardo Boemo, por su apoyo y aporte en todos los aspectos técnicos y académicos de este trabajo, por haber sido mucho más que un director de tesis, con consejos sobre la vida universitaria y por su generosa hospitalidad.

Agradezco a Fernando, por haber aceptado el rol de director académico y por su apoyo en todas las etapas de este posgrado.

Al todo el personal del IIE, especialmente al el equipo de Electrónica Aplicada: Julio, Sebastián, Javier, Fiorella, Francisco, Andrés, Leonardo, Juan y Diego.

A Mariela por su hospitalidad madrileña.

A Virginia por acompañarme durante toda esta etapa.

Fuentes de financiamiento:

- Beca de la CAP de doctorado, Universidad de la República
- Proyecto ANII Fortalecimiento de Posgrados en Ingeniería Eléctrica PR-POS-2008-003
- Ministerio Español de Ciencia e Innovación, contrato TEC2007-68074-C02-02
- Proyecto de cooperación Santander-UAM Latinoamérica
- Proyecto Euroform Polo Español.
- Convenio marco UAM-Universidad de la República
- Régimen de Dedicación Total de la Universidad de la República

Tabla de Contenido

Lista de Figuras

Lista de Tablas

Glosario

A/D	Analog to Digital
AES	Advanced Encryption Standard
ASIC	Application Specific Integrated Circuit
AWG	Arbitrary Waveform Generator
BLIFF	Berkeley Logic Interchange Format File
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
CMOS	Complementary Metal Oxide Semiconductor
COTS	Commercial Off-The-Shelf, componente que puede ser adquirido en el mercado comercial
CUT	Circuit Under Test
Cyclone	Familia de FPGAs de Altera
DES	Data Encryption Standard
DCM	Digital Clock Manager
DLL	Delay-Locked Loop
DSP	Digital Signal Processor
DUT	Device Under Test
EDA	Electronic Design Automation
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
GPIO	General Purpose Input Output
GPS	Global Positioning System
GPU	Graphics processing unit
Hardcore	Circuito prediseñado implementado directamente en silicio, el circuito no es programable por el usuario.
HEX	Formato de archivo HEX originario de Intel
HDL	Hardware Description Language

HW	Hardware
IIE	Instituto de Ingeniería Eléctrica de la Facultad de Ingeniería, Universidad de la República
I/O (IO)	Input/Output
IP core	Intellectual Property core, circuito prediseñado que se comercializa en forma similar a un paquete de software
LFSR	Linear Feedback Shift Register
LUT	Look-Up Table
MD5	Message-Digest algorithm
MOS	Metal-Oxide-Semiconductor
NMOS	N-type Metal-Oxide-Semiconductor
OpenCores	Sitio web de diseños IP core de código abierto, http://www.opencores.org
PDA	Personal Digital Assistant
PC	Personal Computer
PLL	Phase-Locked Loop
PMOS	P-type Metal-Oxide-Semiconductor
RAM	Random Access Memory
SAIF	Switching Activity Interchange Format
SEM-FPGA	Sistema propuesto en esta tesis para que una FPGA mida su propio cosumo. El nombre viene de Self Energy Meter - FPGA
SRAM	Static RAM
SoC	System-on-Chip
SoPC	System on a Programmable Chip
Softcore	Circuito prediseñado que se comercializa en forma similar a un paquete de software, y es implementado en las celdas lógicas de un dispositivo programable. También llamado IP core, o simplemente core.
Spartan	Familia de FPGAs de Xilinx
USB	Universal Serial Bus
VCD	Value Change Dump
VCO	Voltage-Controlled Oscillator

Verilog	Lenguaje de especificación hardware
VFC	Voltage-to-Frequency Converter
VHDL	Lenguaje de especificación hardware VHSIC (Very-high-speed Integrated Circuit) Hardware Description Language
Virtex	Familia de FPGAs de Xilinx
WSN	Wireless Sensor Network

Capítulo 1

Introducción

Hoy en día es ampliamente aceptado que las tres restricciones principales de un diseño electrónico digital son el área, la velocidad y el consumo; pero históricamente no siempre fue así [1]. Durante muchos años el área y la velocidad fueron las restricciones dominantes, y el problema del consumo estaba reservado a nichos de aplicaciones muy específicas como los relojes pulsera, marcapasos, audífonos, calculadoras portátiles, circuitos de aplicación militar, y algunos sistemas alimentados a baterías [2]. Si bien hay algunas publicaciones de mediados de los '60 [3], [4] acerca de circuitos de muy bajo consumo, puede afirmarse que hasta 1990 el consumo de los circuitos integrados digitales no fue tenido en cuenta por la amplia mayoría de los diseñadores [5]. A partir de esa fecha se produce un cambio debido fundamentalmente al avance de la densidad de integración de los circuitos CMOS alcanzada a principios de los '90, que combinada con el aumento de la frecuencia de funcionamiento, hizo que un encapsulado pudiera disipar varias decenas de Watts, obligando los diseñadores a prestar cada vez más atención a los problemas de consumo y disipación de calor [6].

La década de los '90 fue también explosiva en el crecimiento de los dispositivos portátiles: PDAs, *laptops*, GPSs, pero sobre todo se dio un gran crecimiento de la telefonía móvil.

Aparecen también las primeras conferencias dedicadas al bajo consumo. En 1993 se organiza en Arizona la *Low-Power Electronics Conference*, y en 1994 el *Workshop*

on Low-Power Design en Napa, CA; estas dos conferencias se fusionaron en 1995 para dar lugar al ISLPED (*International Symposium on Low Power Electronics and Systems*) que se organiza anualmente. En Europa el proyecto PATMOS (*Power and Timing Modelling for Optimisation and Specification*) realizado durante los años 1990-1992 dio lugar a la creación de una conferencia anual que se mantiene hasta la actualidad.

En los últimos 20 años el consumo de los circuitos electrónicos pasó entonces de ser una restricción de diseño reservada solamente a algunos nichos específicos, a ser una de las variables más importantes en una gran mayoría de los diseños electrónicos. Tal importancia queda de manifiesto cuando el *International Technology Roadmap for Semiconductors* (ITRS) del año 2011 dice lo siguiente:

> *"Power consumption is now the major technical problem facing the semiconductor industry"* [7]

El informe del año 2013 del ITRS es aún más contundente, y afirma que la minimización del consumo es la variable que impulsa el diseño de circuitos integrados:

> *"The heterogeneous integration of multiple technologies in a limited space (e.g., GPS, phone, tablet, mobile phones, etc.) has truly revolutionized the semiconductor industry by shifting the main goal of any design from a performance driven approach to a reduced power driven approach. In few words, in the past performance was the one and only goal; today minimization of power consumption drives IC design."* [8]

Más aún, los dispositivos portátiles del pasado se caracterizaban por tener relativamente bajos requerimientos de cálculo, pero actualmente, además de masificarse, han evolucionado, y dispositivos como *smartphones*, *tablets* y *notebooks*, incorporan requisitos de altas prestaciones: procesamiento de señales, procesadores gráficos y comunicación digital de alta velocidad; aumentando así la presión por el

bajo consumo [9]. En el caso de los dispositivos alimentados a baterías el consumo de energía es una variable crítica, la cantidad de energía que puede dar una determinada batería es fija, por lo tanto el consumo del dispositivo establece el intervalo entre recargas. A esto se suma que la evolución histórica de la capacidad de las baterías que puede contener un equipo portátil, por unidad de volumen o por peso, no aumenta con la misma tasa de crecimiento que la electrónica, sino que lo hace más lentamente que el consumo esperado de los circuitos integrados [10].

Por otro lado, en equipos conectados a la red eléctrica, si bien el consumo no es una variable tan crítica, existe también una gran presión por el ahorro energético. Esto se da a todo nivel. En el ambiente doméstico hay fuertes exigencias para bajar el consumo, sobre todo el consumo en *standby* del equipamiento que está todo el tiempo encendido. A nivel industrial el costo de la energía impacta muy fuerte a los grandes consumidores de equipo electrónico: *data centers* [11], [12], redes de telecomunicaciones, etc.

Dentro de los diferentes tipos de circuitos electrónicos hay algunos que se adaptaron más rápidamente a las exigencias del bajo consumo que otros, un claro ejemplo es en el área de los microcontroladores donde casi todos los fabricantes ofrecen sus versiones *low power*. En cambio, un ejemplo en el otro sentido, está dado por las FPGAs basadas en RAM estática, que prácticamente no cuentan con versiones de bajo consumo. Dentro de los dispositivos lógicos programables actuales, los de menor consumo son los que almacenan su programación en tecnología *antifuse* o *flash*, y los que utilizan RAM estática son los que presentan el consumo más elevado. En contrapartida las FPGAs basadas en RAM estática son los dispositivos que presentan mayor flexibilidad por su capacidad para ser reprogramadas o reconfiguradas. Es en esta flexibilidad en donde se encuentra el mayor atractivo de esta tecnología, ya que permite pasar una idea a silicio en tiempos muy cortos; pudiendo además cambiar el diseño configurado en un mismo circuito integrado cuantas veces se quiera. Es por esto que en esta tesis se ha elegido exclusivamente este tipo de FPGAs, basadas en RAM estática, como objeto de estudio.

Los dispositivos lógicos programables ya tienen más de 30 años de existencia, y se han consolidado en varios nichos de aplicaciones de diseños digitales. En sus comienzos estos dispositivos fueron utilizados para sustituir diseños realizados con lógica discreta y posteriormente para sustituir diseños *full-custom* con volúmenes de producción bajos.

La lógica programable permite obtener velocidades hardware con flexibilidad software. La posibilidad de reutilización del hardware programable abarata su costo ya que puede utilizarse exactamente el mismo hardware para varias aplicaciones cambiando exclusivamente su programación interna [31].

La tecnología de la lógica programable o reconfigurable nos ofrece entonces un cambio de paradigma: hardware que puede modificarse vía software. De la misma manera que una computadora puede escribir datos en una memoria, la misma computadora puede grabar un determinado circuito dentro de un chip, y cambiarlo tantas veces como se quiera. El circuito se modifica internamente, sin la necesidad de que haya cambios físicos externos.

Sin embargo la flexibilidad de estos circuitos tiene como consecuencia un significativo aumento del área de silicio utilizada, debido principalmente la memoria de configuración, los caminos de interconexión y la llaves configurables. Esto hace que las FPGAs utilicen muchos más recursos para resolver una determinada función si se las compara con un circuito de diseño específico (ASIC). En un extenso trabajo de comparación entre circuitos de similares características implementados en ASICs y en FPGAs, Kuon y Rose [13] llegan a la conclusión que las implementaciones en FPGAs son 4 veces más lentas, ocupan un área 35 veces mayor y consumen 14 veces más. Otra comparación entre la potencia consumida por ASICs y FPGAs puede verse en [14], en este trabajo los resultados obtenidos son mucho peores, y en el mejor caso el consumo de las implementaciones en FPGAs es 20 veces superior al del ASIC correspondiente. Un tercer trabajo compara multiplicadores realizados en FPGAs y en *Standard Cells* [15], las medidas de consumo en este caso resultan en que la versión en FPGA es 70 veces superior.

Desde la introducción de las FPGAs a mediados de la década de los '80 el foco de la investigación tanto en sus arquitecturas como en las herramientas de desarrollo se centró en mejorar el área y la velocidad de los diseños; dejando de lado el consumo. Los módulos de análisis de consumo han sido tardíamente incorporados a las herramientas de diseño tanto comerciales como académicas, por ejemplo Xilinx anunció su módulo Xpower en diciembre de 2000 [2], Altera introdujo el PowerPlay en 2004 [3], y en 2002 un módulo de estimación de consumo se agregó al VPR que es una herramienta de diseño comúnmente utilizada en ambientes académicos [4].

El consumo de las FPGAs basadas en RAM estática continúa siendo un problema abierto y el principal obstáculo para que estos dispositivos puedan ser utilizados en una gran cantidad de aplicaciones.

1.1 Objetivos y propuesta de la tesis

El objetivo general de este trabajo es el desarrollo de una metodología para el manejo de la variable "consumo" en FPGAs con un enfoque fuertemente experimental y desde el punto de vista de un usuario de este tipo de dispositivos.

La pregunta que se busca responder es si las FPGAs comerciales basadas en SRAM pueden ser utilizadas en aplicaciones de bajo consumo, y dónde están los puntos críticos para que esto sea así.

Para poder dar algunas respuestas a esta pregunta, primero hubo que lidiar con varios aspectos particulares que tiene la variable consumo, y que la hace intrínsecamente diferente a las otras dos variables de un diseño que son el área y la velocidad.

Se desarrolla una técnica de medición de consumo, y se comparan medidas con estimaciones; se propone la utilización de un conjunto de circuitos de *benchmarks*

especialmente adaptados para el análisis de consumo, y se exploran distintas técnicas de reducción de consumo.

A lo largo de esta tesis se han realizado más de 350 experimentos con diferentes circuitos diseñados y construidos, a los cuales se les ha medido y estimado su consumo.

1.2 Marco tecnológico

En esta tesis se utilizaron exclusivamente FPGAs que almacenan su configuración en RAM estática, trabajando con dispositivos de los dos principales fabricantes Xilinx y Altera.

En el caso de Xilinx se utilizaron dos plataformas diferentes:

- Avnet Spartan-6 LX16 Evaluation Kit, que incluye una FPGA de 45nm modelo Spartan-6 XC6SLX16-2CSG324C
- Digilent Spartan-3 Board, que incluye la FPGA XC3S200-FT256 de 90nm

En el caso de Altera se utilizó siempre la misma plataforma:

- Altera DE0 Board, fabricada por Terasic, con la FPGA de 65 nm Cyclone III EP3C16F484

En cuanto a las herramientas de desarrollo se optó por utilizar las herramientas de los fabricantes en las versiones disponibles al inicio del trabajo de tesis. Si bien a lo largo del trabajo fueron saliendo nuevas versiones de las mismas estas no se utilizaron como forma de no agregar nuevos grados de libertad al trabajo.

- Xilinx ISE Design Suite 13.1
- Altera Quartus II 9.0

Otro punto importante es que a lo largo de toda la tesis solamente se tiene en cuenta el consumo interno de las FPGAs, descartando el consumo de entrada/salida de los

dispositivos. La justificación para haber tomado esta opción es que el consumo de entrada/salida, principalmente el de salida, depende fuertemente de la carga vista en los pines y una vez fijado este parámetro no hay márgenes para actuar sobre él.

Todo el trabajo está hecho desde el punto de vista de un usuario de FPGAs, esto implica que se utilizan FPGAs comerciales en condiciones normales de funcionamiento recomendadas por los fabricantes en sus hojas de datos.

1.3 Sinopsis de la Tesis

Describiremos brevemente la estructura del documento.

En el Capítulo 2 se presentan los fundamentos teóricos del consumo de las FPGAs basados principalmente en el consumo de los dispositivos CMOS.

En el Capítulo 3 es donde se establece toda la metodología de medida. Se comparan medidas con estimaciones dadas por las herramientas de desarrollo, se detallan las plataformas utilizadas y los bancos de prueba para medir consumo. Se propone un dispositivo que permite que la FPGA mida su propio consumo. Finalmente la metodología se valida caracterizando el consumo de un multiplicador reconfigurable.

El Capítulo 4 presenta los *benchmarks* de consumo propuestos. Se detallan un conjunto de circuitos que pueden ser utilizados en FPGAs tanto de Altera como de Xilinx para medir consumo real y para realizar simulaciones y estimaciones. El capítulo finaliza con datos experimentales utilizando esos circuitos.

El Capítulo 5 describe la aplicación de algunas técnicas de reducción de consumo con ejemplos experimentales. Las técnicas aplicadas fueron: *pipelining*, utilización de bloques hardware, *clock gating*, utilización del modo *suspend* y variaciones de las herramientas de diseño. El capítulo concluye con un caso de estudio, la reducción de consumo de un circuito en particular, el microcontrolador openMSP430.

Por último en el Capítulo 6 se dan algunas conclusiones, se analiza el trabajo realizado y se plantean líneas de investigación para el futuro.

Capítulo 2

Consumo de potencia en FPGAs

En este capítulo se describe el consumo de potencia en FPGAs. Se presenta muy brevemente la arquitectura básica de una FPGA. Luego, dado que las FPGAs actuales están basadas en circuitos CMOS se comienza por una revisión de las componentes que generan el consumo en dicha tecnología. Finalmente se detallan las particularidades del consumo en FPGAs.

2.1 Arquitectura de una FPGA

Básicamente los dispositivos lógicos programables están compuestos por un gran conjunto de bloques lógicos cuya función es programable y una red de interconexiones de estos bloques también programable (Figura 2-1). La red de interconexiones programable puede ser jerárquica o plana, y con caminos de diversa longitud entre bloques. Los bloques lógicos tienen una parte combinatoria y un elemento de memoria (*flip-flop*) que puede o no ser utilizado, la función combinatoria de estos bloques está implementada con una *lookup-table* (LUTs) programable que puede ser de 4, 6 o más entradas (Figura 2-2).

Toda la configuración del dispositivo se almacena internamente en bits de RAM estática. Esos bits de RAM controlan el enrutamiento de las señales, las funciones combinatorias de las LUTs, y si se utiliza o no el elemento de memoria o *flip-flop*.

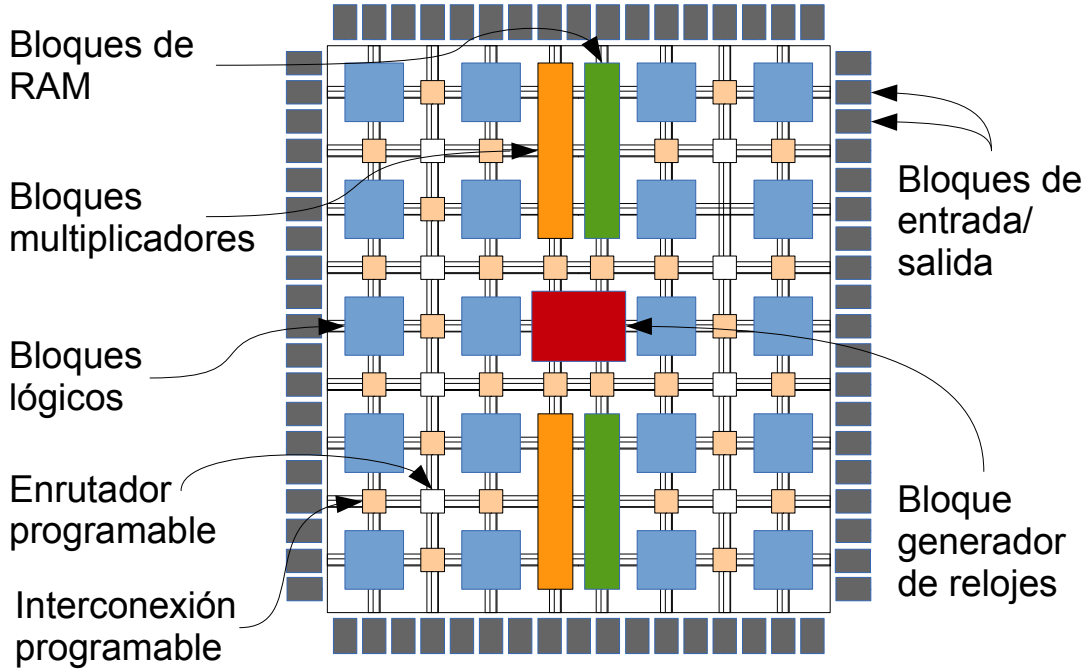


Figura 2-1 Estructura interna de una FPGA

Las FPGAs modernas, además de la estructura básica descrita, incluyen otro tipo de elementos que son bloques hardware dedicados. Estos pueden ser bloques de memoria disponible para la aplicación, bloques de control de relojes (PLL, DCM), bloques aritméticos, generalmente multiplicadores-acumuladores (DSP, *Embedded multipliers*), microprocesadores, e interfaces de varios tipos de entrada/salida.

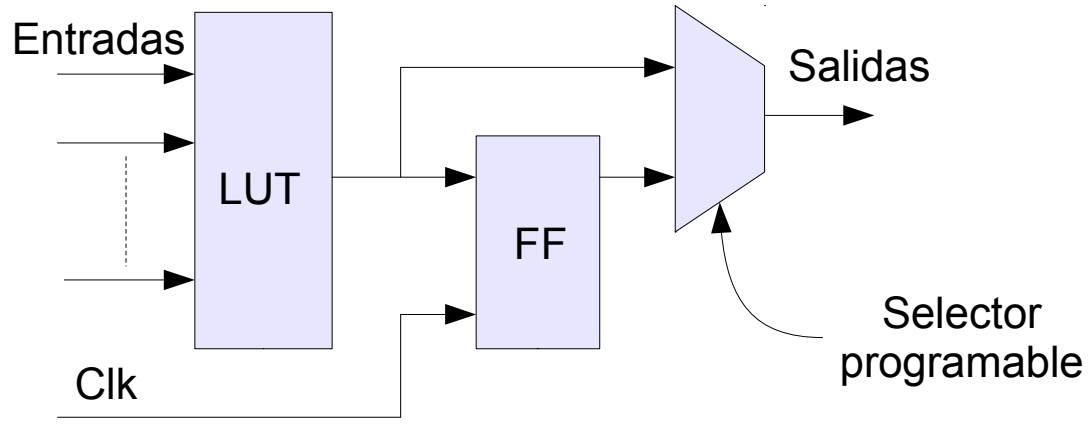


Figura 2-2 Estructura básica de un bloque lógico

2.2 Consumo en circuitos CMOS

El consumo de un inversor CMOS y en general de todos los circuitos basados en CMOS puede dividirse en tres componentes:

1. Consumo dinámico producido por la carga/descarga de la capacidad vista hacia la salida.
2. Consumo dinámico producido por la corriente de corto circuito.
3. Consumo estático causado por las corrientes de fugas.

Comenzaremos viendo la forma analítica de la primer componente del consumo. Este primer término representa la potencia útil, ya que sirve para conmutar el valor lógico de un determinado nodo, mientras que los otros dos son pérdidas.

En la Figura 2-3 puede verse un inversor CMOS con una carga capacitiva, esta carga en realidad está compuesta de varias capacidades tanto a fuente como a tierra, pero se pueden representar con un único condensador C_L equivalente al total de la capacidad vista por la salida del inversor.

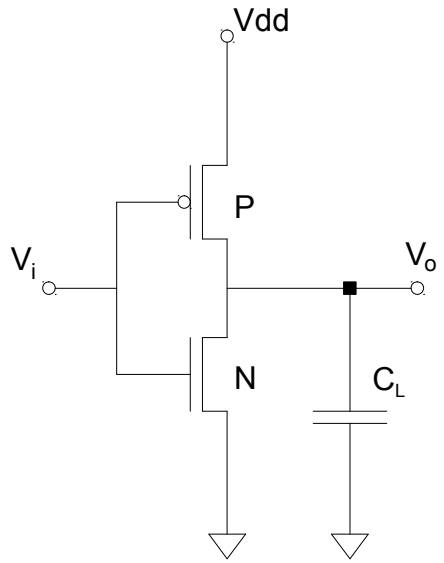


Figura 2-3 Inversor CMOS

Analizando primero la carga del condensador desde 0V a V_{dd} a través de la conducción del transistor P podemos ver que se disipa una energía

$$E_P = \frac{1}{2} C_L V_{dd}^{\ 2} \tag{2-1}$$

y el condensador a su vez almacena la misma cantidad energía

$$E_{CL} = \frac{1}{2} C_L V_{dd}^{\ 2} \tag{2-2}$$

Esta energía almacenada en el condensador se disipa en el transistor N al descargarse el mismo, por lo tanto la energía disipada en el transistor N es una vez más:

$$E_N = \frac{1}{2} C_L V_{dd}^{\ 2} \tag{2-3}$$

La energía total disipada por el inversor CMOS en un ciclo subida y bajada de su salida, o lo que es lo mismo de carga y descarga de la capacidad C_L, es entonces:

$$E = C_L V_{dd}^{\ 2} \tag{2-4}$$

Si la entrada V_i del inversor CMOS es una onda cuadrada, periódica de frecuencia f_{clk} entonces la potencia de un ciclo completo de carga y descarga está dada por la ecuación:

$$P = C_L V_{dd}^{\ 2} f_{clk} \tag{2-5}$$

Esta ecuación puede generalizarse a una red de compuertas CMOS con n nodos en donde la potencia total tiene la siguiente expresión:

$$P = V_{dd}^{\ 2} f_{clk} \sum_{i=1}^{n} C_i \alpha_i \tag{2-6}$$

donde C_i es la capacidad del nodo i, y α_i representa la cantidad de transiciones completas de carga-descarga de dicho nodo por ciclo de reloj, también llamada tasa de actividad (*activity rate*) de un nodo.

Un detalle a tener en cuenta es que el factor α_i no es directamente el valor que dan las herramientas de simulación, conocido como *toggle rate*, ya que las mismas cuentan o acumulan los cambios de nivel (o los flancos) de cada nodo del circuito, y no las

transiciones completas carga-descarga. Es debido a esto que muchas veces esta ecuación aparece con un factor de 1/2.

Vale la pena señalar que en un circuito síncrono sin azares el valor de α_i está generalmente entre 0 y 1, ya que un nodo puede cambiar a lo sumo una vez en cada período de reloj; pero este valor puede ser mayor si en el circuito hay *glitches*.

Los *glitches*, que son pulsos no deseados debidos a los diferentes retardos de propagación de las señales en la etapa combinatoria, pueden aumentar mucho el consumo de un circuito. Otro comportamiento posible, que influye en el consumo, es que estos pulsos no tengan una excursión completa entre fuente y tierra, ya que si son muy angostos las capacidades pueden no llegar a cargarse o descargarse completamente. Para que el cálculo de la potencia sea correcto estos efectos se deben incluir en el modelo, y una forma de hacerlo es incluirlos en el valor de α_i. En este factor α_i se deben incluir no solo las transiciones de cada nodo, sino también las transiciones debidas a los *glitches*, ya sean completas o incompletas.

Como puede verse en la ecuación (2-6) la potencia consumida depende en forma cuadrática del voltaje, por eso la tensión de alimentación es la principal variable en la reducción de consumo dinámico. Las otras 3 variables involucradas: frecuencia, capacidad y cantidad de transiciones presentan una dependencia lineal.

La segunda componente del consumo del inversor CMOS es la potencia de cortocircuito, que se debe a la conducción simultánea de los dos transistores N y P del CMOS, y por lo tanto es producida por una corriente que circula a través de los mismos entre V_{dd} y tierra. Este fenómeno se da durante breves transitorios en las conmutaciones, debido a que la forma de onda de entrada no sube o baja en forma instantánea. Cuando la entrada recorre valores intermedios se presentan zonas de conducción de los dos transistores.

En [16] puede verse un cálculo analítico de la potencia de cortocircuito para un inversor CMOS sin carga con una entrada en rampa con tiempos τ de subida y bajada, la ecuación a la que se llega es:

$$P_{SC} = \frac{\beta}{12}(V_{dd} - 2V_{th})^3 f_{clk} \tau \qquad (2\text{-}7)$$

donde β representa la ganancia en continua de los transistores. Esta ecuación sin embargo no es muy útil, ya que siempre los inversores tienen a la salida algún tipo de carga, generalmente capacitiva. En la misma publicación pueden verse simulaciones numéricas de potencias de cortocircuito en inversores con cargas capacitivas, y las mismas se reducen considerablemente con respecto a la ecuación (2-7). En [17] el cálculo de la potencia de cortocircuito con carga capacitiva se reduce al 30% del valor obtenido si se utiliza la ecuación (2-7), y al 1.5% de la potencia disipada en la carga/descarga de capacidades.

No incluiremos un análisis detallado de la corriente de cortocircuito con carga capacitiva, que sí pueden verse por ejemplo en [18], [19], [20] o en [21], pero daremos algunos detalles de su comportamiento. El primero es que, como se produce en las conmutaciones, es lineal con la frecuencia de reloj. Puede demostrarse que depende fuertemente de los tiempos de subida y bajada de la entrada al inversor, y en general podemos decir que:

- crece al incrementarse los tiempos de subida y bajada de la señal de entrada
- decrece para grandes capacidades de salida
- decrece cuando baja el voltaje de alimentación, incluso puede llegar a ser completamente eliminada si $V_{dd} < V_{thN} + |V_{thP}|$ ya que en este caso es imposible que los dos transistores estén conduciendo al mismo tiempo.

Por lo visto anteriormente hay técnicas que permiten reducir y acotar la potencia de cortocircuito, y además esto es válido también para las nuevas tecnologías submicrónicas. Lo contrario a este se da para el consumo estático, causado por corrientes de fuga, que se ha visto incrementado enormemente con la reducción del tamaño de los transistores especialmente por debajo de los 100nm.

Estudios detallados de las corrientes de fuga en transistores submicrónicos pueden verse en [22], [23], [24] o [25]. La Figura 2-4 muestra todas las corrientes que se producen en transistores MOS de escala nanométrica en ambos estados conduciendo y cortado, y un resumen de las mismas puede verse en la Tabla 2-1.

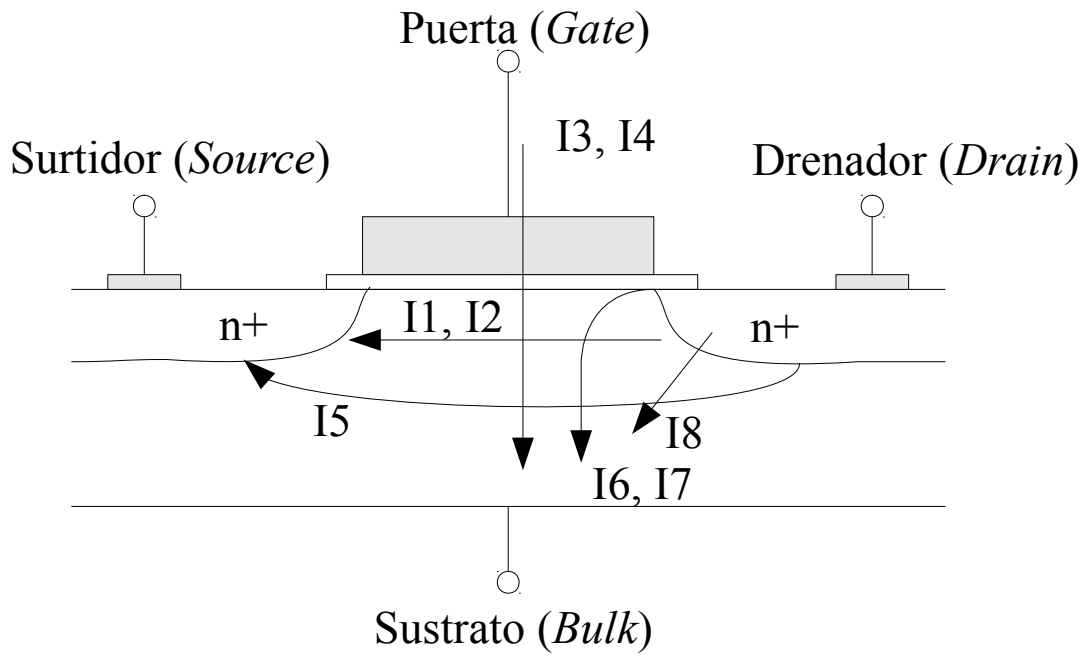


Figura 2-4 Corrientes en nano-transistores MOS

Tabla 2-1 - Listado de corrientes en nano-transistores MOS.

	Estado		Descripción
	ON	OFF	
I1	X		*drain-to-source active current*
I2		X	*subthreshold leakage*
I3	X	X	*gate leakage (oxide tunneling current)*
I4	X	X	*gate current due to hot-carrier injection*
I5		X	*channel punch-through current*
I6		X	*gate-induced drain leakage (GIDL)*
I7		X	*band-to-band tunneling current (BTBT)*
I8	X	X	*reverse bias PN junction leakage*

En los transistores actuales con bajos voltajes Vth, I2-*subthreshold leakage* e I3-*gate leackage* son las componentes dominantes de las fugas, seguidas de I6-GIDL e

I7-BTBT [24]. A esto se suma el hecho que I2 aumenta fuertemente con la temperatura. Existen varias técnicas para lidiar con estas corrientes de fugas ya sea a nivel del proceso tecnológico o a nivel de circuito [22], [26], [27], pero escapan al alcance de esta tesis.

2.3 Consumo en FPGAs

Si bien las FPGAs están basadas en CMOS, y por lo tanto se cumple lo desarrollado en el apartado anterior, las mismas presentan algunas particularidades que vale la pena destacar.

El primer punto a mencionar es la programación, las FPGAs basadas en RAM estática almacenan su programación en SRAM y por lo tanto hay dos consumos asociados. Uno es el consumo transitorio inicial debido a la programación del chip, y el otro, ya en modo de funcionamiento, es el consumo de todas esas celdas de SRAM que aportarán al término de potencia estática.

Otra particularidad de las FPGAs es el uso intensivo de transistores de paso, ya sea en las interconexiones programables o en la lógica. Las ecuaciones de consumo de potencia en un transistor de paso difieren un poco de las del inversor CMOS:

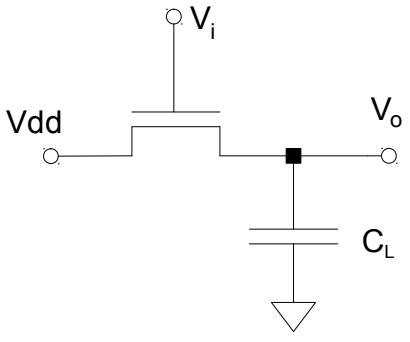


Figura 2-5 Transistor de paso NMOS

Para el caso de un solo transistor de paso NMOS, la salida V_o no tendrá una excursión completa *rail-to-rail* sino que llegará al valor $(V_{dd} - V_{th})$ [28], por lo tanto si suponemos que el condensador se encuentra inicialmente descargado, y V_i pasa de 0 a 1, entonces la energía consumida de la fuente en la carga del condensador es igual a:

$$E_{0\to 1} = C_L V_{dd}(V_{dd} - V_{th}) \qquad (2\text{-}8)$$

Si bien es menor que la energía utilizada en la carga del condensador en la versión *rail-to-rail* (como es el caso del inversor CMOS analizado anteriormente), y por lo tanto la potencia activa se reduce, esto perjudica la potencia estática. Veremos un ejemplo conectando un inversor en cascada con el circuito anterior tal como se muestra en la Figura 2-6.

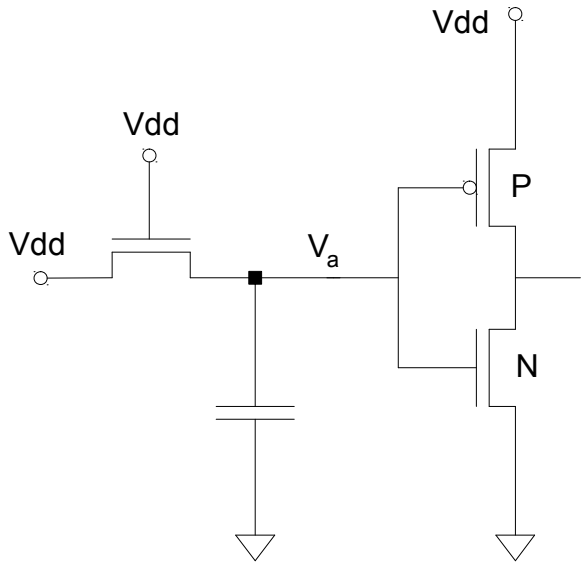


Figura 2-6 Transistor de paso alimentando un inversor CMOS

Como ya vimos la tensión de salida del transistor de paso llega $(V_{dd} - V_{th})$, esto alcanza para que el inversor esté con su salida en 0, pero el transistor PMOS no quedará cortado sino en inversión débil, (dado que $|V_{gs} - V_{th}| \approx 0$), de modo que queda circulando una corriente desde la fuente a tierra, lo cual puede generar consumos importantes. Existen soluciones a esto, agregando un transistor extra que sirva de

pull-up entre V_a y la alimentación, o utilizando compuertas de "*gate boosting*" o "*re-buffering*", que son utilizadas en FPGAs, para recomponer los niveles lógicos y evitar estas corrientes [29].

En las FPGAs los transistores de paso se utilizan en las interconexiones programables (Figura 2-7), dentro de las LUTs (Figura 2-8) y en los multiplexores de enrutamiento (Figura 2-9). La estructura completa de una FPGA a nivel de transistor puede consultarse en dos interesantes artículos de 1999 en donde se da un ejemplo de diseño de una FPGA completa: [30] y [31]. Actualmente hay algunos artículos que proponen utilizar *transmission gates* como una alternativa a los transistores de paso [29], y durante un tiempo existieron chips con *buffers* tri-estado, pero hoy los transistores de paso son ampliamente utilizados.

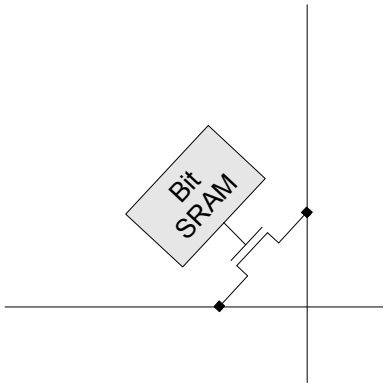


Figura 2-7 Interconexión programable

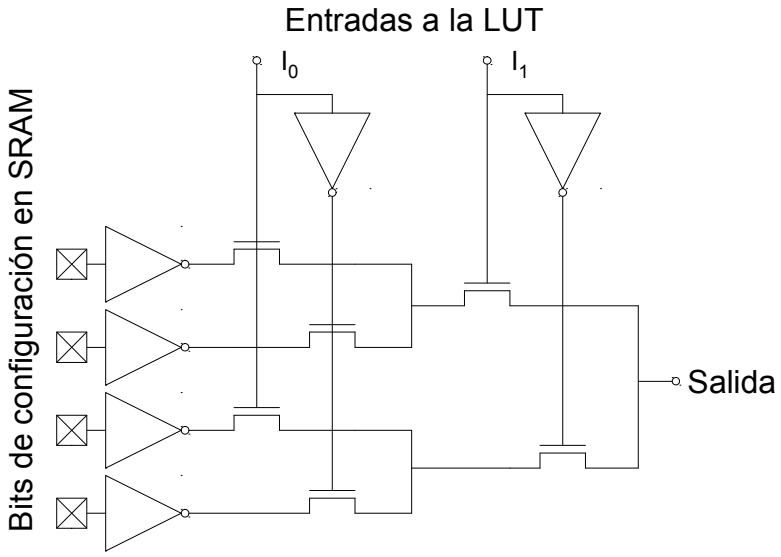


Figura 2-8 Tabla de *look-up* (LUT) a nivel transistor (ejemplo de 2 entradas)

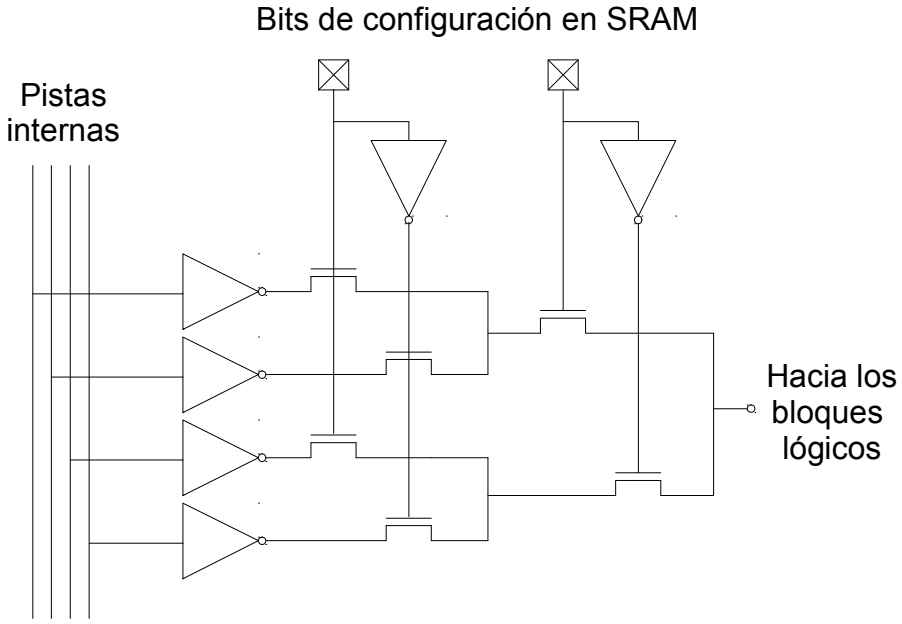


Figura 2-9 Multiplexor para enrutar entradas hacia las LUTs

Las corrientes de fuga que se presentan en transistores de paso dependen fuertemente de las tensiones aplicadas, por lo tanto la potencia estática será también dependiente de los datos. Algunos trabajos sobre las corrientes de fugas en FPGAs y las técnicas para reducirlas, principalmente reordenando entradas a datos y conexiones, pueden verse en [32], [33], [34] y [35].

El cableado interno de las FPGAs está formado por líneas de distinta longitud con transistores de paso en las interconexiones. Si se quiere realizar un cálculo de la potencia dinámica disipada es necesario modelar esto como una red de capacidades y resistencias.

Continuando con las particularidades de las FPGAs en cuanto a su consumo debemos mencionar la distribución de relojes dentro del chip, y los bloques para generar relojes. Estos bloques, que incluyen PLLs o DLLs, a veces cuentan con fuentes de alimentación separadas. En cuanto a la distribución de reloj dentro del integrado, se hace mediante líneas especiales dedicadas, y puede ser una parte muy importante del consumo total, en [36] se pueden ver ejemplos entre que van entre 4 y el 22%.

Diremos algunas palabras acerca de los bloques hardware internos a las FPGAs, los más usuales son bloques de memorias RAM y bloques de aritmética entera, generalmente multiplicadores-acumuladores, a veces llamados bloques DSP. También existen dispositivos que incluyen microprocesadores, o interfaces dedicadas. Lo que puede decirse es que con respecto al consumo estos bloques se comporten como lo que son, es decir un circuito ASIC y no como una FPGA; aunque se encuentran dentro de una FPGA.

Las celdas de entrada-salida de un FPGA no presentan mayores diferencias con respecto al consumo que las de cualquier integrado digital, si bien son programables y admiten una gran cantidad de protocolos, se ajustan a la ecuación de potencia dinámica dependiente de la capacidad externa de carga.

Se han publicado varios trabajos que modelan en forma completa el comportamiento de las FPGAs desde el punto de vista de su consumo, utilizados para el desarrollo de herramientas de estimación, por ejemplo pueden consultarse [37], [38], [39], [40] o [41]. En ellos puede verse cada una de las particularidades de las FPGAs y cómo pueden modelarse, ya sea mediante ecuaciones o mediante caracterización de sus bloques.

Capítulo 3

Medidas vs. Estimaciones

De las tres características que determinan un diseño: área, velocidad y consumo, ésta última es la que trae más problemas al intentar estimarla o acotarla. El área se calcula en forma exacta, y las herramientas de diseño calculan en forma muy acertada la frecuencia máxima de funcionamiento, pero las estimaciones de consumo pueden presentan errores muy importantes, sobre todo en etapas tempranas de un desarrollo.

Esto se debe principalmente a dos razones, la primera es la actividad de cada nodo, es decir qué tantas variaciones entre 0 y 1 tendrá la salida de una determinada compuerta en funcionamiento real; y la segunda es el problema de los *glitches*[1] que se propagan en los bloques combinatorios.

Tanto la actividad de los nodos como la propagación de *glitches* hace que el problema de estimar consumo sea fuertemente dependiente no solo de un diseño sino de los datos que ingresan al mismo, más allá que, obviamente, se debe tener muy bien caracterizado el semiconductor, interconexiones, pistas, etc.

Como las FPGAs son circuitos caracterizados por sus fabricantes, y las herramientas de desarrollo de bajo nivel también son producidas por éstos, en este capítulo se

[1] La palabra *glitch,* ampliamente utilizada en electrónica y que significa "error o fallo menor", es relativamente nueva en inglés, probablemente incorporada desde el Yiddish en la década de 1940. Por una historia del término puede verse [168].

analizan en profundidad las herramientas que proporcionan los fabricantes para realizar estimaciones de consumo.

Los resultados obtenidos en este capítulo van a definir la modalidad de trabajo experimental del resto de esta tesis.

En primer lugar se realiza una revisión bibliográfica de publicaciones que efectuaron comparaciones entre medidas y estimaciones. Luego se describen en detalle los experimentos realizados, incluyendo los bancos de pruebas, las herramientas utilizadas en las estimaciones y el procedimiento de medida. Por último se presentan y analizan los resultados obtenidos.

Hay varios trabajos que publican medidas directas de consumo de FPGAs, los primeros de ellos fueron realizados antes que las herramientas de estimación de consumo más comúnmente utilizadas fueran desarrolladas: [42], [43], [44] y [45], trabajando con familias Xilinx XC3000 y XC4000. En [36] se analiza el consumo dinámico de una Virtex II, dividiendo la distribución del consumo en 3 factores que contribuyen a la potencia total: capacitancia, utilización de recursos y actividad del circuito. Una comparación entre el consumo medido y el estimado con la herramienta Xpower en diferentes aplicaciones con reconfiguración dinámica sobre dispositivos Virtex fue presentado en [46], y una comparación entre herramientas de Xilinx Xpower y de Altera PowerPlay puede verse en [47] (vale aclarar que esta referencia es publicada por Altera).

En [48] se presenta una metodología para realizar estimaciones de alto nivel en consumo dinámico; aplicando esta metodología en dispositivos Spartan-3 se obtuvo un error del 18% con respecto al valor medido. Errores bastante superiores se obtuvieron en [49], donde se comparan medidas con estimaciones realizadas con las herramientas de los fabricantes. Para el caso de Xilinx los autores obtuvieron diferencias que varían entre 15% y 208% para dispositivos Virtex II-Pro y Spartan-3; mientras que en Altera con un Cyclone II los errores van desde 5% hasta 21%. Los circuitos de prueba que se utilizaron corresponden a 4 diferentes algoritmos de

seguridad: AES, DES, 3DES y MD5. En [50] y [51] también se comparan medidas contra estimaciones y se obtienen errores que van entre 3 y 37% para el primer trabajo y 20% para el segundo.

Hay también publicaciones que no realizan medidas, y comparan sus estimadores de consumo con simulaciones realizadas en SPICE. Un ejemplo bastante completo puede verse en [39] en donde se presenta un estimador de consumo con un error de 8% con respecto a la simulación de SPICE para un conjunto de *benchmarks* de Microelectronics Center of North Carolina (MCNC, 1991).

Para medir la energía consumida ciclo a ciclo por una FPGA se puede utilizar un capacitor tal como se presenta en [52]. La ventaja de este método es que se puede determinar la energía estática y dinámica por cada ciclo de reloj para un determinado diseño. Los autores reportan que el Xpower da resultados en algunos casos por encima y en otros por debajo de las mediciones con diferencias muy grandes que llegan casi a 300% de error.

Una metodología de medida similar a la utilizada en esta tesis es presentada en [53], en donde se realizan mediciones y estimaciones en circuitos con diferentes etapas de *pipeline*.

En [54] se realizan medidas y estimaciones para comparar, desde el punto de vista del consumo, la utilización de LUTs o de bloques de RAM para la implementación de lógica, llegando a la conclusión que consume más la RAM.

Un trabajo más reciente [55] presenta una metodología para estimar el consumo dinámico de multiplicadores embebidos en una Xilinx Virtex-II Pro; y en [56] los mismos autores proponen una metodología para medir potencia en FPGAs con medidas indirectas de las capacidades internas del chip.

Resumiendo, podemos decir que las diferencias entre medidas y estimaciones reportadas tienen un rango muy amplio, encontrándose entre -100% y 295% para Xilinx y entre -89% y 37% para Altera.

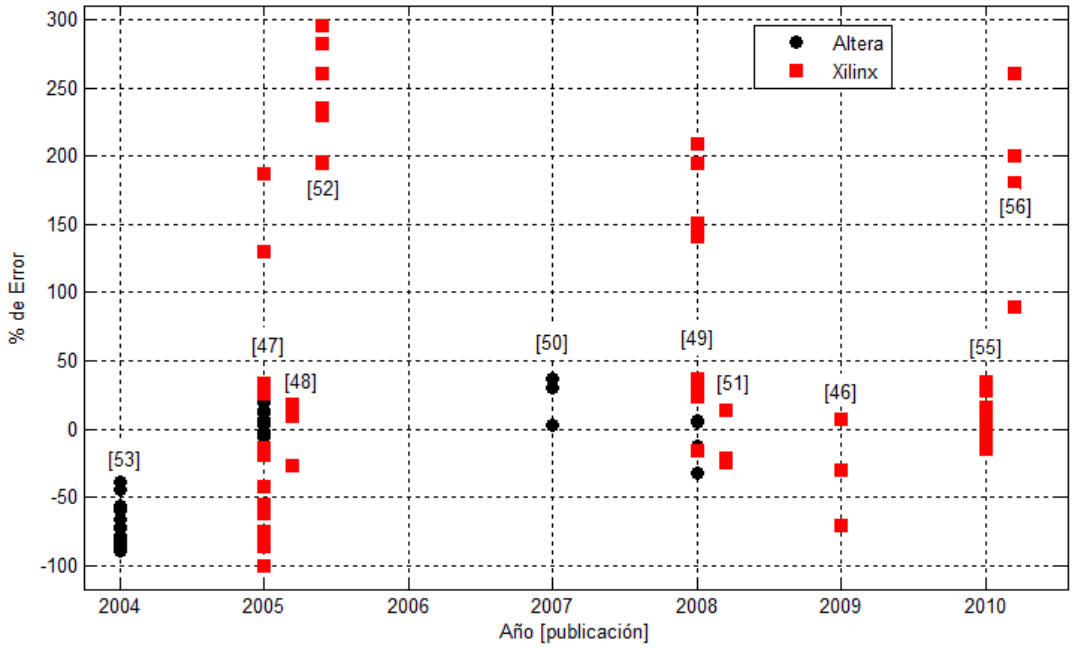


Figura 3-1 Errores de estimación de consumo en publicaciones

En la Figura 3-1 puede verse una gráfica con la distribución de los errores de estimación de consumo de las publicaciones mencionadas. En el eje horizontal se grafica el año de la publicación y entre corchetes la referencia. En el eje vertical el porcentaje de error, de acuerdo a la ecuación (3-1).

$$Error(\%) = \frac{Potencia\_estimada - Potencia\_medida}{Potencia\_medida} \qquad (3\text{-}1)$$

3.1 Bancos de medidas

Se utilizaron integrados de los dos principales fabricantes de FPGAs para hacer las medidas. En el caso de Xilinx se utilizó una placa Avnet Spartan-6 LX16 Evaluation Kit y una placa Digilent Spartan-3, y para Altera se usó una placa DE0 fabricada por Terasic que cuenta con un Cyclone III.

En todos los casos se midió solamente el consumo de la fuente que alimenta el *core* o la parte interna de los chips, descartándose los consumos de las fuentes de las entradas y salidas o las fuentes auxiliares (que alimentan PLLs, DCMs, etc.).

En el caso del Kit de Spartan-6 la placa cuenta con un sistema de medición de consumo incluido. Está formado por un *shunt* serie intercalado en la línea de la alimentación que se quiere medir. La tensión sobre este *shunt* es pasada por un amplificador diferencial, y luego por un conversor analógico digital perteneciente a un microcontrolador del tipo PSoC de Cypress. Este microcontrolador promedia los valores obtenidos y los transmite a un PC a través de una interfaz USB. También se puede medir directamente el voltaje sobre el *shunt* con un osciloscopio o un multímetro.

No sucede lo mismo en el caso de la placa DE0 de Cyclone III, esta placa no está adaptada para medidas de consumo, ni cuenta con facilidades para medir corrientes, por lo tanto ha sido necesario adaptarla. Se ha removido el regulador de voltaje que alimenta el *core* del chip y se ha sustituido por un regulador externo y un *shunt* en serie, como puede verse en la Figura 3-2. La misma adaptación se hizo para la placa Spartan-3 de Digilent.

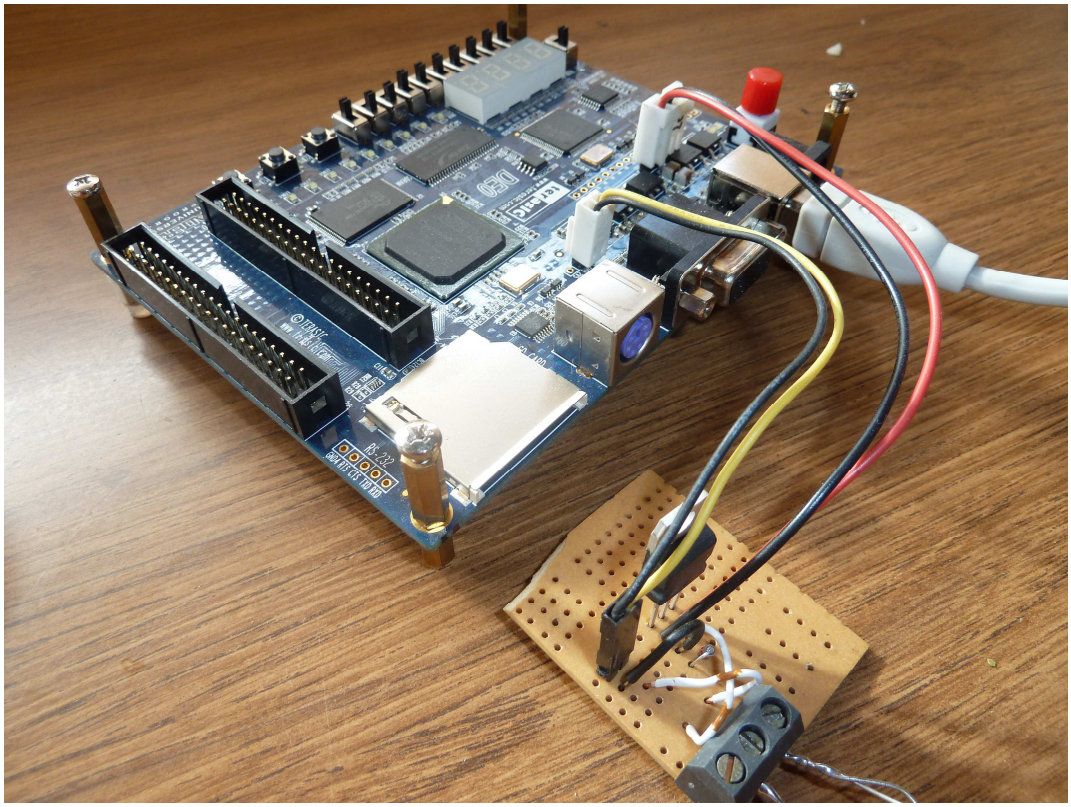


Figura 3-2 Placa DE0 con regulador externo y *shunt*

Con estas plataformas se puede medir el consumo medio en el *core* de las FPGAs si se utiliza un multímetro sobre el *shunt*, o en forma instantánea, si se adquiere la corriente con un osciloscopio digital.

Para minimizar errores se deben calibrar los *shunts*. Los voltajes en los *shunts* fueron medidos con un osciloscopio Tektronix TDS3052C y con un multímetro Fluke 45. En el caso del osciloscopio se calcularon los valores RMS, y se compararon con los del multímetro. La diferencia entre ambos instrumentos fue menor que 0,5% en los casos medidos. Como consecuencia y para simplificar las medidas se decidió medir solamente valores promedio y utilizar en multímetro. Cada medida fue repetida varias veces, la máxima variación observada fue ± 1 en el tercer dígito significativo, por lo tanto el error relativo de las medidas es menor que 1,5%.

En casos de utilizar diseños que difieran mucho en el rango de consumo se hace necesario cambiar el *shunt*. En el caso de corrientes muy bajas se debe aumentar el

valor para mantener el error relativo bajo. En cambio en el caso de diseños que consumen mucho es necesario bajarlo para mantener la caída de tensión dentro de los rangos tolerables por los reguladores y las FPGAs.

3.2 Circuitos de prueba y procedimiento de medida

Para obtener una medida efectiva del consumo de un determinado circuito, el mismo debe estar funcionando, no alcanza solamente con que reciba la señal de reloj, debe contar con datos reales, o similares a los reales, en sus entradas. En una gran cantidad de casos se pueden utilizar entradas pseudo aleatorias, pero en otros no habrá más remedio que utilizar señales reales.

Un tipo de circuitos muy utilizados para generar vectores de entrada pseudo aleatorios son los *Linear Feedback Shift Register* (LFSR). Son muy fáciles de implementar en hardware ya que cuentan solamente con un registro de desplazamiento del largo necesario y una realimentación realizada con compuertas XOR o XNOR. Dependiendo de una adecuada elección de los puntos de realimentación se pueden obtener LFSR que generen secuencias de máxima longitud, es decir secuencias de largo 2^n-1, siendo n la cantidad de bits del registro de desplazamiento.

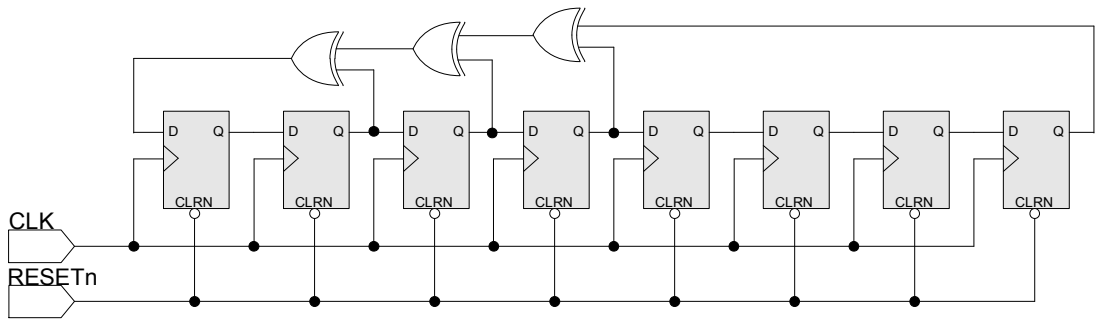


Figura 3-3 Esquemático de un LFSR de 8 bits

En el caso de los LFSR de máxima longitud realimentados con compuertas XOR se recorrerán todos los estados menos el 0, y para los realimentados con compuertas XNOR faltará el estado con todos los *flip-flops* en 1. Un detalle de su implementación en FPGAs puede verse en estas notas de aplicación de Xilinx [57] y en [58].

Un problema que es necesario resolver es que las herramientas de síntesis suponen que las salidas útiles de un circuito siempre terminan en algún pin. Si las mismas no se conectan a patas de salida de la FPGA es muy difícil garantizar que en los procesos de optimización no se elimine esa parte del circuito. Es posible incluir opciones de síntesis para que esto no suceda, pero eso no siempre funcionan correctamente. Como generalmente no se cuenta con los pines necesarios para conectar todas las salidas de un circuito, hay varias estrategias para solucionar este problema. La más utilizada es realizar un XOR con todas las salidas del circuito y conectar a un pin externo solamente la salida de la XOR. Esto es simple de hacer en hardware y tiene como ventaja que reduce el número de patas de salida a una sola, impidiendo que la herramienta minimice parte del circuito. Ejemplos de esta implementación pueden verse en [53] o en [54].

En una primera instancia se pensó que la utilización de una compuerta XOR para juntar todas las salidas de un circuito podría dar problemas ya que la XOR es la compuerta que estadísticamente tiene más alto consumo, debido a que cada vez que cambia un bit de entrada provoca un cambio en la salida. Si bien el consumo de los circuitos auxiliares finalmente será restado para obtener el consumo del circuito bajo pruebas, a veces se requiere que la influencia del consumo de los circuitos auxiliares se mantenga lo más baja posible.

Otra alternativa manejada para evitar que la herramienta de diseño minimice parte del circuito cuando no se pueden conectar las salidas hacia pines de la FPGA fue utilizar un registro de desplazamiento, que cargue en paralelo las salidas del circuito bajo test y conectar hacia afuera solamente la salida serie del registro de desplazamiento, aunque finalmente nunca se utilice. La señal de control de carga paralelo o desplazamiento serie se conecta hacia un pin de entrada.

Estas dos alternativas fueron comparadas con mediciones reales y no arrojaron diferencias significativas en cuanto a consumo, como puede verse en la Tabla 3-1. El análisis se hizo solamente para la Spartan-6, se utilizaron LFSR de 64 bits, reloj de entrada de 66.6MHz y un DCM dividiendo por 1 (se agrega solamente por compatibilidad con otros casos).

Tabla 3-1 - Consumos de los circuitos auxiliares.

Sistemas	Consumo medido
LFSR 64 bits XOR salidas	8.2mA
LFSR 64 bits *ShiftRegister* salidas	7.8mA modo carga paralelo (8.5mA modo *shift*, este modo no se utiliza en los experimentos)

La diferencia para los anchos de palabra y para las frecuencias de reloj utilizadas en los experimentos no es significativa, por lo tanto en muchos casos se utilizará la XOR por razones de simplicidad.

En la Figura 3-4 puede verse el sistema completo, propuesto en esta tesis, con un generador de entradas pseudo aleatorias implementado con un LSFR, el circuito bajo pruebas y una XOR en las salidas. Otro elemento que puede apreciarse en la figura es el DCM o *Digital Clock Manager* para Xilinx, o directamente un PLL en el caso de Altera. Estos bloques se utilizan para generar distintas frecuencias de reloj a partir de un oscilador externo. De esta forma es posible medir el consumo a diferentes frecuencias, o simplemente dividir el reloj para poder medir circuitos lentos.

Para obtener el consumo del circuito de pruebas se deberá restar el consumo de los circuitos auxiliares utilizados. En cada caso, es necesario entonces, generar un circuito que esté formado solamente por los bloques auxiliares interconectando el LFSR generador de entradas directamente con la XOR de salidas y medir su consumo, para después restarlo al del sistema completo.

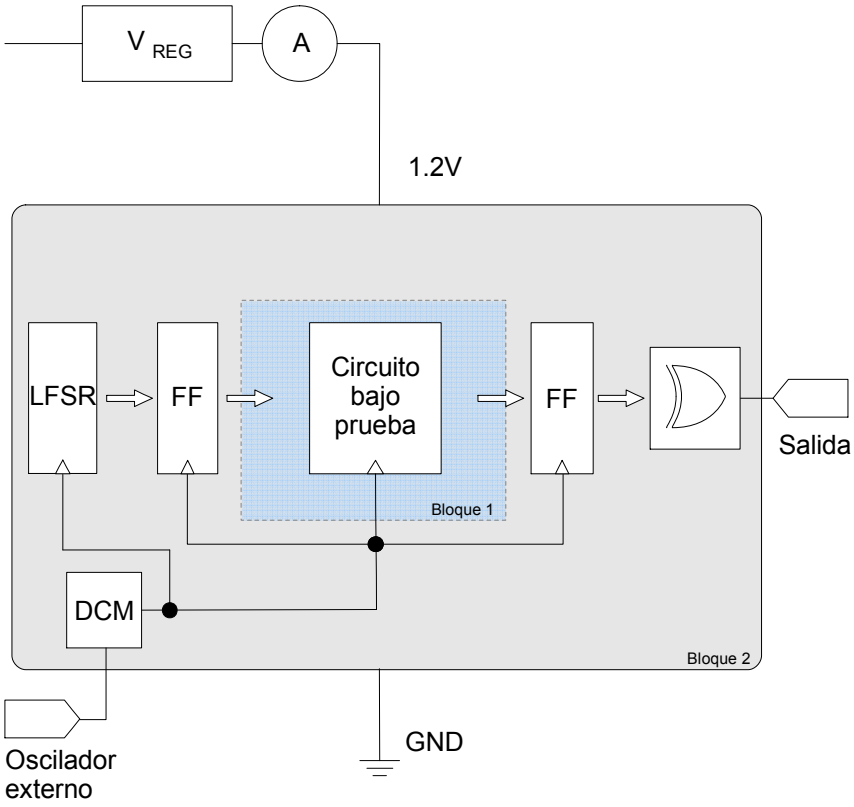


Figura 3-4 Diagrama completo del sistema de medida propuesto

Esta metodología de circuitos auxiliares fue puesta en marcha y ajustada en los diferentes bancos de medida mencionados en el apartado anterior.

3.3 Automedidor de consumo: SEM-FPGA

En este apartado se describe un sistema de medida de consumo instantáneo de un dispositivo programable, que puede ser utilizado por el mismo dispositivo con el cual se está trabajando. Si bien existen transductores que permiten conocer el consumo instantáneo o el estado de las baterías en una gran cantidad de equipos portátiles, su aplicación completa en FPGAs, incluyendo el desarrollo de un *IP core* de aplicación es novedosa: la FPGA se mide a si misma la potencia consumida.

El medidor de consumo propuesto, llamado SEM-FPGA, permite, de una forma simple, que la aplicación que está funcionando en la FPGA cuente con una entrada que le indica cuánto está consumiendo en determinado instante de tiempo. Las aplicaciones pueden ser de varios tipos: poder controlar o corregir consumos excesivos, saber cuánta energía se está consumiendo o cuál es el remanente en un banco de baterías, o realizar perfiles de consumo detallados de circuitos en funcionamiento.

El sistema cuenta con dos bloques, un bloque externo al FPGA que se intercala en la línea de alimentación que se quiere medir, y un *core* interno desarrollado en VHDL que realiza y procesa las lecturas.

El sistema se probó en una placa DE0 con un integrado Cyclone III de Altera EP3C16, pero puede adaptarse fácilmente a cualquier sistema con una FPGA donde pueda intercalarse un *shunt* con la fuente de alimentación, incluso pueden medirse varias fuentes replicando el mismo sistema.

3.3.1 Antecedentes

Existen varios trabajos que realizan medidas de consumo dentro del sistema, la gran mayoría de ellos están orientados a dispositivos con microprocesadores, ya sea en el área de redes de sensores inalámbricos WSN (*Wireless Sensor Networks*) o en equipos portátiles a batería.

En cuanto a la forma de medida hay algunas variantes, por ejemplo en [59] se utiliza un par de capacitores que conmutan carga entre la fuente y el dispositivo, y se cuenta la cantidad de veces que se intercambia carga, este método es fácil de utilizar y no requiere conversión A/D, pero generalmente produce un *ripple* en la fuente. El método más extendido es el uso de una resistencia *shunt* serie, intercalada entre la fuente y el sistema a medir, siendo la caída de voltaje en dicha resistencia proporcional a la corriente consumida. Hay que tener algunos cuidados en la elección del valor de la resistencia para que por un lado la caída de voltaje no afecte el

funcionamiento del sistema, y por otro que no sea una tensión tan chica que de errores muy grandes al medirla. Un ejemplo de la utilización de este método, junto a conversores A/D, en nodos de una WSN puede verse en [60].

Se han desarrollado algunas alternativas para evitar la inclusión de conversores A/D, una de ellas es utilizar un conversor de voltaje a frecuencia (VFC), como puede verse en [61] en donde el consumo es adquirido por el propio nodo de la WSN.

Las únicas publicaciones que se han encontrado que realizan medidas en FPGAs y las utilizan dentro de la propia FPGA son muy recientes: [62] y [63], en realidad es un mismo trabajo publicado primero en un congreso en 2012 y luego en una revista en 2013. Este trabajo se centra principalmente en el circuito externo, que es un conversor de corriente a ancho de pulso, y en el estudio de errores; y no explica en forma detallada el diseño interno dentro de la FPGA. Tampoco se especifica el consumo propio del sistema de medida, pero no parece haber sido tenido en cuenta.

El método desarrollado en esta tesis se diferencia en que el circuito externo de medida utiliza un conversor de corriente a frecuencia, de muy bajo consumo y bajo costo; y se agrega una descripción detallada del diseño de la parte interna dentro de la FPGA. Es decir un *IP core* diseñado específicamente para permitir una rápida integración del medidor de consumo con otras aplicaciones.

El módulo del conversor de corriente a frecuencia está adaptado de un trabajo previo realizado para sistemas con microcontroladores de una red de sensores inalámbricos que puede verse en [64] y [65].

3.3.2 Descripción del circuito externo del SEM-FPGA

En la Figura 3-5 puede verse un diagrama de bloques del circuito externo que básicamente cuenta con un *shunt* serie y un bloque que convierte la tensión de caída del *shunt* en frecuencia. Esta señal de frecuencia se conecta a un pin de entrada del

FPGA, y el *IP core* desarrollado en VHDL permite realizar una medida de dicha frecuencia.

El objetivo de diseño es entonces contar con un circuito simple, de bajo costo, que tenga un consumo propio bajo, y que sea capaz de convertir la corriente que pasa por el *shunt* en una frecuencia manteniéndose lineal en el rango de trabajo.

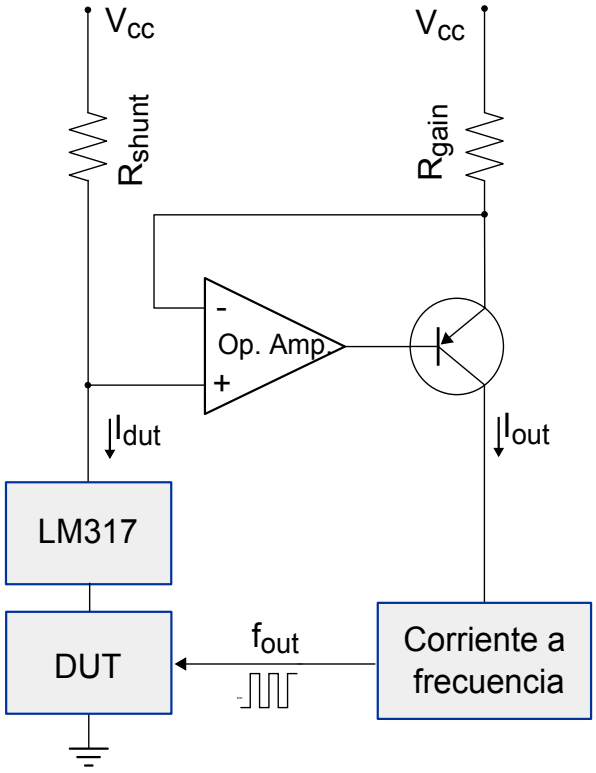


Figura 3-5 Circuito externo de conversión de corriente a frecuencia

Dentro de cierto rango de corrientes la frecuencia de salida es lineal con la corriente $I_{dut.}$. Una diagrama más detallado del sistema puede verse en la Figura 3-6, en donde se ven los bloques que conforman el sistema. Primero hay un conversor de corriente-corriente, implementado con un operacional y un transistor, que toma una muestra de la corriente de la FPGA (y del regulador LM317) y la convierte en I_{out}. Esa corriente se convierte a frecuencia utilizando una versión de ultra bajo consumo del clásico 555, concretamente el CSS555C de la firma Custom Silicon Solutions.

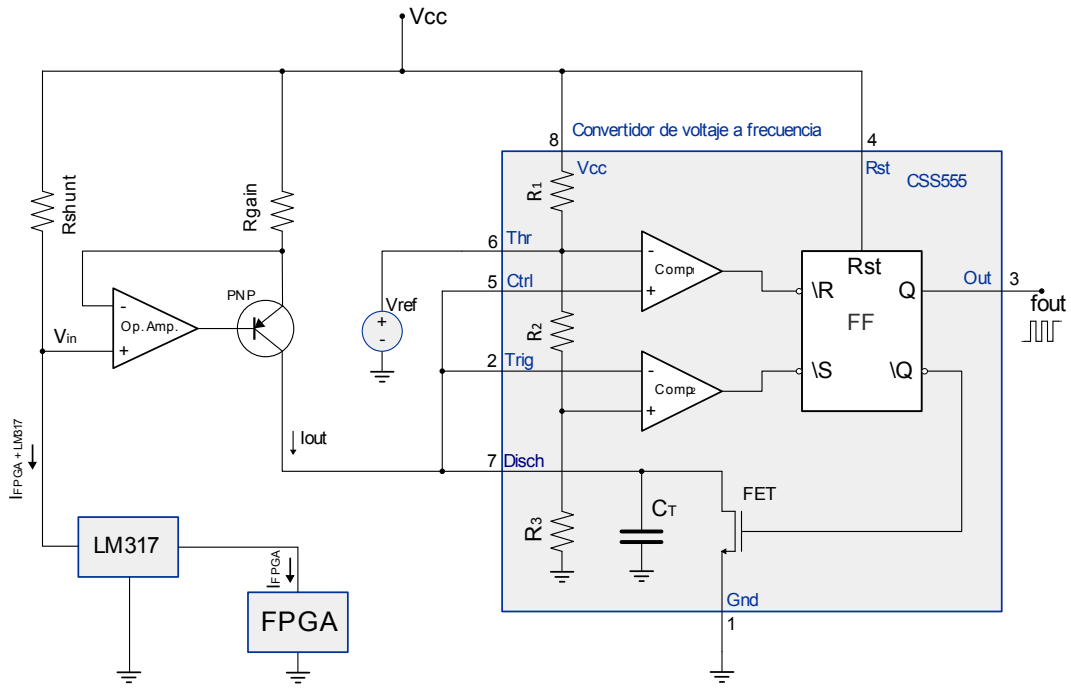


Figura 3-6 Detalle del circuito externo de conversión de corriente a frecuencia

En la resistencia R_{shunt} se genera una caída de voltaje de modo que la tensión a la entrada no inversora del operacional se tiene:

$$V_{in} = I_{dut} R_{shunt} \tag{3-2}$$

Se puede ver que la relación entre las corrientes I_{dut} e I_{out} es la siguiente:

$$I_{out} = I_{dut} \frac{R_{shunt}}{R_{gain}} \tag{3-3}$$

Para que la ecuación (3-3) se cumpla los voltajes de ambas ramas deben ser el mismo, en este caso V_{dd}. Pero el circuito del 555 debe estar alimentado con 3.3V y como se quiere medir el consumo interno de la FPGA que es de 1.2V se intercala el regulador LM317, por lo tanto se estará midiendo el consumo interno del FPGA más el consumo del regulador, que deberá calibrarse y restarse al consumo medido.

$$I_{fpga} = I_{dut} - I_{LM317} \tag{3-4}$$

La segunda etapa es un convertidor de corriente a frecuencia construido con el CSS555C. El circuito implementado con el 555 genera una salida cuya frecuencia depende de la corriente I_{out}, de modo que midiendo esa frecuencia se puede conocer la corriente I_{dut}.

El funcionamiento detallado de un circuito similar puede verse en [66], aunque en esta versión se utiliza solamente el capacitor interno, y la relación de carga-descarga es diferente. Esto se debe a que como el circuito es de muy bajo consumo los comparadores y el FF-RS tienen retardos muy grandes que hay que tener en cuenta en los cálculos de la carga-descarga. La selección de componentes se realizó para obtener una buena linealidad dentro de los rangos típicos de consumo de una FPGA, para un cálculo detallado de los mismos puede consultarse [67].

A partir de la hoja de datos del CSS555C [68] se ve que su consumo es de 3.1μA. La etapa de conversión voltaje-corriente se implementa mediante un amplificador operacional de bajo consumo, por ejemplo el OPA379. A partir de la hoja de datos de este componente se puede ver que su consumo es de 2.9μA, por lo que en este caso se obtiene un consumo promedio del método de 6.0μA.

Una relevamiento del circuito construido dio los siguientes valores: V_{cc} = 3.3V, V_{fpga} = 1.24V y en la Tabla 3-2 se pueden ver los valores de corriente y frecuencia, graficados en la Figura 3-7.

Tabla 3-2 Medida de la frecuencia en función de la corriente

Corriente (mA)	Frecuencia (kHz)
1,5	1,9
5,0	8,4
15,0	27,9
20,3	38,1
29,8	56,5
50,7	94,9
60,6	112,3
71,5	130,1
78,4	141,7
89,1	158,3
99,2	173,6
111,4	190,7
120,3	202,9
131,1	217,2
145,0	234,8
149,8	241,2

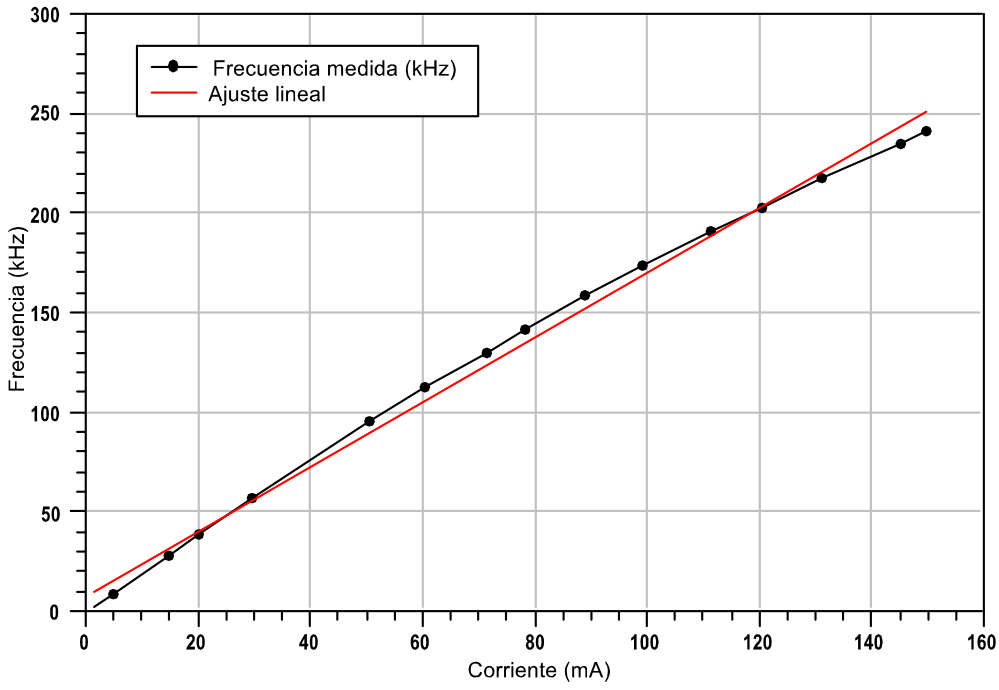


Figura 3-7 Frecuencia vs. corriente del VFC ajustado al rango de trabajo

Se midió el consumo medido del LM317, que no varía significativamente dentro del rango de trabajo: $I_{LM317} = 52\mu A$, este valor puede o bien restarse, o bien despreciarse dado que los consumos mínimos de las FPGA utilizadas están en el orden de los 5mA.

3.3.3 Descripción de los bloques internos del SEM-FPGA (*IP cores*)

En esta sección se describen los bloques internos desarrollados para medir la frecuencia que es proporcional al consumo. Las ideas de estos circuitos fueron validadas mediante el proyecto [69] dirigido por el autor de esta tesis.

Teniendo en cuenta el rango de consumos de la FPGA y los valores seleccionados para las resistencias R_{shunt} y R_{gain}, la frecuencia de la señal de entrada tendrá el rango de variación dado por (3-5).

$$f_{input} = [1\ kHz, 200\ kHz] \tag{3-5}$$

Adicionalmente se fijó un error del 1% como requerimiento para las medidas.

Hay dos métodos básicos para medir la frecuencia de una señal. El primero consiste en utilizar un reloj mucho más rápido que la señal a medir, y contar la cantidad de períodos de reloj que caben en un período de la señal de entrada. El error de este método se calcula como ±1 períodos de reloj, o sea que el peor error relativo se da para la frecuencia máxima que se desea medir. El segundo método consiste en invertir las cosas, es decir utilizar una señal auxiliar de frecuencia mucho menor que la señal a medir, y contar los períodos de la señal a medir que caben en un período de la señal auxiliar. En este caso el error es ±1 períodos de la señal a medir, y se obtiene con la entrada más lenta.

Cuál de estos métodos conviene utilizar depende de cada aplicación, por ejemplo si lo que se busca es obtener un perfil detallado de consumo en tiempos muy cortos,

entonces el método 1 es el indicado; en cambio si se quieren obtener consumos medios durante un tiempo largo es mejor utilizar el método 2.

Para este caso se seleccionó el método 2 porque es el que mejor se adapta a las medidas que se quieren realizar con este sistema. Para acotar el error relativo al 1%, entonces la cuenta de pulsos debe ser mayor a 100 en el peor caso que como dijimos es con la frecuencia mínima, o sea 1*KHz*. Esto determina que la señal auxiliar debe ser menor a 10*Hz*.

Debido al reloj disponible en la placa, y a la posibilidades de dividirlo sin gastar demasiados recursos de la FPGA, finalmente se utilizó una frecuencia de 2.44*Hz*. Esto quiere decir que se contarán los pulsos de la señal de entrada cada 1/2.44 segundos, y ese valor será proporcional a la corriente media consumida durante ese intervalo. Para esto es necesario entonces que el diseño contenga un contador.

Los sucesivos valores de este contador se almacenarán en memoria para construir así un perfil del consumo de la FPGA. Está claro que la cantidad de bits utilizados para el contador, así como la cantidad de memoria disponible determinarán la duración máxima del perfil de consumo. Por ejemplo con un contador de 32 bits se puede registrar el consumo durante más de 5 horas sin que de *overflow*, ya que

$$\frac{2^{32}}{200kHz} = 21474{,}8\text{s} = 5.9horas \qquad (3\text{-}6)$$

Como el contador ocupa 4 bytes, la cantidad de medidas que se pueden almacenar estará limitada por la memoria interna del dispositivo. El Cyclone III utilizado cuenta con 56 bloques de 9k bits, si por ejemplo destinamos 32kB de esta memoria para almacenar medidas, entonces tenemos que la memoria se llenará en un poco menos de una hora, según muestra la ecuación (3-7).

$$\frac{32kB}{2.44Hz * 4B} = 3357\text{s} = 0.9horas \qquad (3\text{-}7)$$

Finalmente cabe destacar que con el método elegido se puede obtener el consumo total desde el reset del sistema, ya que se almacenan todos los pulsos de la onda de entrada.

Se implementaron dos módulos VHDL, el primero (SEM periódico) realiza una captura periódica del valor acumulado en el contador y lo almacena; mientras que el segundo (SEM asíncrono) solo captura y almacena el valor cuando se dispara una señal de habilitación. El segundo método necesita además conservar la información del tiempo transcurrido, para ello se debe almacenar junto al contador un índice de tiempo.

La primer implementación, llamada SEM periódico, incrementa el contador cada vez que hay un flanco de subida en la señal de entrada, y almacena el valor acumulado en dicho contador cada 2.44*Hz* en la RAM interna. El diseño es parametrizable en el número de bits utilizados para el contador, y la cantidad de memoria disponible. Cuando la memoria se llena se activa una señal ISFULL, y la adquisición de datos se detiene.

En la Figura 3-8 puede verse un esquema del circuito implementado. El bloque "Frec Div" usa los PLLs de la FPGA para generar el reloj del sistema CLK_SYS de 2*MHz* y la señal auxiliar o reloj lento CLK_SLOW de 2.44*Hz* utilizada para muestrear y almacenar el contenido del contador.

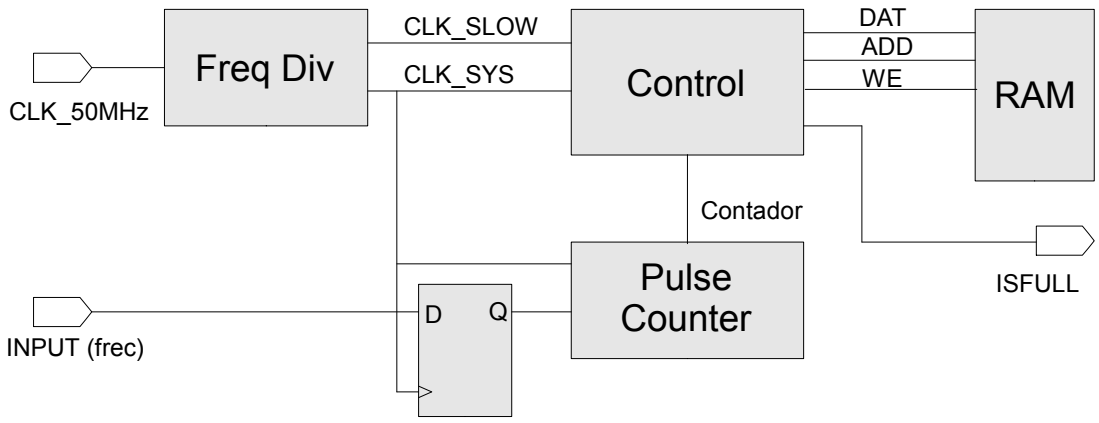


Figura 3-8 Esquema del SEM periódico

El bloque "Pulse Counter" es una máquina de estados que básicamente lo que hace es incrementar el contador con cada subida de la señal de entrada. El bloque "Control" es el encargado de manejar la RAM y almacenar los valores del contador en el momento adecuado, además genera la salida ISFULL cuando la RAM disponible se llena.

La segunda implementación, SEM asíncrono, es muy similar a esta, pero hay una entrada más, llamada SAVE, que controla en qué momentos se almacena el contador, y la etiqueta del tiempo en la RAM. Es necesario agregar entonces un contador extra para llevar la cuenta del tiempo, y otra RAM para almacenarlo como puede verse en la Figura 3-9. El contador del tiempo cuenta flancos de la señal CLK_SLOW. El almacenamiento en ambas RAMs se realiza cuando hay un flanco en CLK_SLOW y la entrada SAVE está activa.

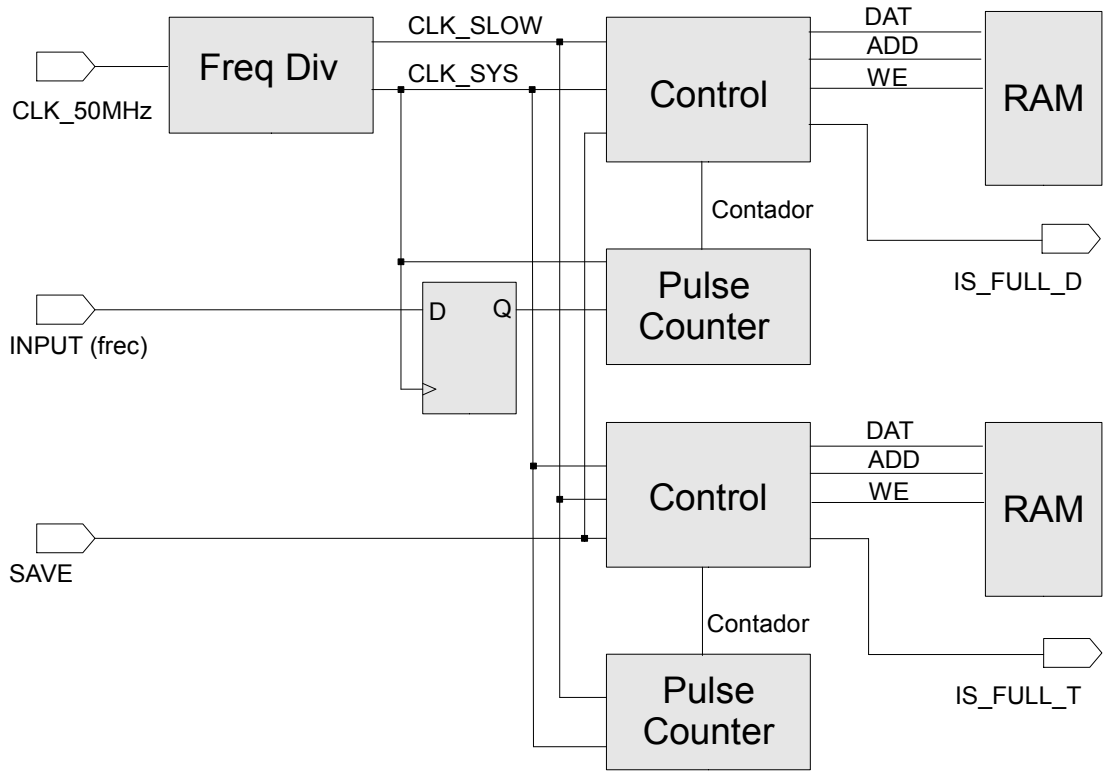


Figura 3-9 Esquema del SEM asíncrono

Con esta segunda implementación es posible realizar perfiles de consumo disparados por diversos eventos, por ejemplo en el caso de máquinas de estado anidadas es posible disparar la señal SAVE cada vez que el circuito cambia de una máquina de estados a otra. Otro ejemplo podría ser en el caso de un sistema con un microcontrolador embebido, se puede generar una señal cada vez que el programa realiza un salto o un retorno, y de esa manera obtener el perfil de consumo del programa principal y de cada una de las subrutinas.

Más adelante se verán ejemplos de medidas realizadas con este sistema, y una validación del mismo (ver sección 4.4).

Es importante caracterizar el consumo propio de los bloques del SEM dentro de la FPGA, para poder restarlo al circuito bajo prueba. Las pruebas físicas se realizaron para la placa DE0, con un integrado Cyclone, en este caso el consumo total medido de la FPGA con el SEM funcionando es de 11,5mA. Si a esto se le resta el consumo estático del Cyclone que es de 6,5mA obtenemos un consumo propio del *IP core* del SEM de 5mA.

3.4 Estimaciones de consumo

Como ya se mencionó anteriormente, estimar el consumo de un circuito presenta varios problemas, y se han citado trabajos que muestran una gran variación entre los consumos estimados y los medidos.

En esta sección se presenta un método de estimación desarrollado en esta tesis. El modelo se compara posteriormente contra medidas reales. Naturalmente, la estimación es más precisa cuanto más cerca del circuito final se realiza. Es decir, a partir de un circuito ya sintetizado, y con un PPR (*partitioning-place-route*) hacia la FPGA seleccionada.

La otra gran incógnita del estimador de consumo es la actividad de cada nodo, es decir cada cuantos períodos de reloj conmuta de valor cada nodo del sistema. Las herramientas de estimación traen un valor por defecto de 12,5%, que es un valor absolutamente empírico obtenido de la estadística de una gran cantidad de diseños. Si estamos queriendo realizar la estimación más ajustada posible no podemos utilizar este valor tan grueso para estimar la actividad de los nodos. La alternativa es simular el circuito, y con la herramienta de simulación generar un archivo de actividad. Existen dos tipos de archivos de actividad que pueden ser generados por las herramientas de simulación: archivos VDC y archivos SAIF.

Un archivo VCD (*Value Change Dump*) es un archivo ASCII que contiene, además de encabezados y definición de variables, información detallada de los cambios en cada señal para cada paso de la simulación de un circuito, es decir que contiene los cambios de valor y en qué instante de tiempo se dieron dichos cambios. En general son archivos de gran tamaño, y sirven solamente cuando es necesario analizar el consumo de un circuito en forma dinámica.

El archivo SAIF (*Switching Activity Interchange Format*, ahora parte del estándar IEEE 1801-2013 [70]) en cambio, contiene sólo la información de conmutación de los nodos de un circuito, es decir que para cada nodo contabiliza los cambios y guarda solo esta información. Cuenta además el tiempo que la señal estuvo en cada posible valor: 0, 1, X o Z. Este tipo de archivos es mucho más reducido en tamaño que el VCD, y en general es el único que se utiliza para realizar estimaciones u optimizaciones de consumo. Es posible también realizar una conversión de un archivo VDC a un SAIF.

Si la simulación se realiza con entradas similares a las reales, entonces el archivo de actividad reflejará los cambios reales en los nodos del diseño bajo prueba, siendo la mejor estimación de consumo que se puede obtener de una FPGA. Si bien persiste aún un problema que son los *glitches*. Es decir transiciones espurias y no deseadas que pueden ocurrir en un nodo antes que llegue a su estado estable final. En general estas transiciones se dan cuando la lógica combinatoria tiene varios niveles y caminos

no están balanceados convergen en una misma compuerta. Esto puede hacer que un nodo aumente significativamente la cantidad de transiciones y por lo tanto transformarse en una importante fuente de consumo no útil [71].

No siempre los simuladores lógicos dan un valor realista de los *glitches*. Esto es debido a varias razones, pero la principal es que normalmente en los circuitos sincrónicos no importan, es decir que si se trabaja a frecuencias adecuadas, los *glitches* no afectan el buen funcionamiento de un circuito ya que se extinguen antes que llegue el reloj a la siguiente etapa de *flip-flops*. Los simuladores lógicos no son precisos simulando la propagación de *glitches*, y en general los sobreestiman [72].

En simulaciones analógicas los *glitches* pueden estudiarse en forma exacta, pero está claro que para circuitos digitales de gran tamaño esto sería imposible de utilizar. En simulación lógica es posible variar cómo el simulador actúa con respecto a los *glitches* cambiando los modelos de retardo de las compuertas o, en algunos casos, realizando un filtrado de los pulsos muy angostos. Esto último no todos los simuladores los admiten, pero es bastante usual que se pueda indicar cómo un porcentaje del retardo de la compuerta. En general se define como un *glitch* parcial aquel que es más corto que el retardo de la compuerta a la cual llega, y se supone que este pulso no logra conmutar el valor de la salida de esa compuerta, aunque sí consume algo de energía en la entrada de la misma debido a las capacidades.

Lo difícil entonces es, trabajando con simuladores lógicos, manejar adecuadamente el filtrado de *glitches* para ajustarse a los valores reales de actividad de un circuito. Presentaremos a continuación una comparación entre medidas y estimaciones realizada con herramientas comerciales disponibles en Altera y Xilinx.

3.5 Medidas vs. estimaciones en Altera Cyclone

Las técnicas mencionadas anteriormente tanto para medir como para estimar consumo son utilizadas en un Cyclone III de Altera. Como circuitos de prueba se eligen multiplicadores de diferentes características: varios anchos de palabra, diferentes

etapas de *pipeline,* implementaciones en LUTs y en bloques hardware (bloques DSP incluidos en la FPGA).

3.5.1 Trabajo experimental

El objetivo es analizar las diferencias obtenidas en estimaciones de consumo y compararlas con mediciones. Las estimaciones se realizan de la forma más exacta posible, utilizando la herramienta PowerPlay Analyzer con archivos de actividad de los nodos obtenidos de simulaciones.

Se utilizan diferentes tipos de multiplicadores como circuitos de prueba. En todos los casos se usa una frecuencia de reloj fija de 50MHz.

Todas las mediciones se realizan en una placa Terasic DE0, modificada para poder medir corriente en un *shunt* externo, que cuenta con una FPGA del tipo Cyclone III 3C16 de Altera.

Como circuitos de prueba se utilizan multiplicadores enteros sin signo de 3 anchos de palabra diferentes: 32x32, 54x54 y 64x64. Para cada uno de estos multiplicadores se ensayaron diferentes alternativas de implementación:

- implementación en LUTs puramente combinatoria
- implementación en LUTs con diferentes etapas de *pipelining*
- implementación en bloques hardware combinatoria
- implementación en bloques hardware con diferentes etapas de *pipelining*

Las entradas a los multiplicadores se generan con LFSRs dentro de la FPGA y se utiliza una XOR para generar una única salida hacia un pin externo, tal como se describe en la sección 3.2. La única entrada externa al Cyclone es la señal de reloj de 50MHz.

Todos los casos son simulados durante una gran cantidad de ciclos de reloj, para garantizar que el circuito llega a un régimen estable desde el punto de vista de su consumo, y se generan archivos de actividad del tipo SAIF. Con estos archivos se

estima el consumo utilizando la herramienta PowerPlay Analyzer de Altera. Esta herramienta genera información detallada de la estimación de la potencia y las corrientes consumidas por las diferentes fuentes que alimentan el chip: VCCINT, VCCIO, VCCA y VCCD.

Nuestro banco de pruebas con la placa DE0 solamente es capaz de medir la corriente en la línea de 1.2 Volts, que alimenta las fuentes del chip VCCINT y VCCD. VCCINT es la alimentación del núcleo interno del Cyclone, y VCCD es la entrada de alimentación de la parte digital del bloque PLL.

Los resultados obtenidos se muestran en la Tabla 3-3, Tabla 3-4 y Tabla 3-5, donde cada tabla corresponde a un tamaño de multiplicador: 32x32, 54x54 y 64x64. En la primer columna se encuentra el diseño, luego la corriente medida, otra para la estimada, y en la última el error de estimación.

Tabla 3-3 Mult 32x32.

Tipo de circuito	Medidas (mA)	Estimación (mA)	Error %
LUT	52,9	55,4	4,6
LUT 1 *pipeline*	39,2	38,7	-1,4
LUT 2 *pipeline*	37,4	41,9	11,9
LUT 3 *pipeline*	40,1	43,9	9,4
LUT 4 *pipeline*	40,5	39,4	-2,7
LUT 5 *pipeline*	41,9	39,6	-5,5
LUT 6 *pipeline*	54,2	51,1	-5,8
Embedded mult	16,8	17,5	4,5
Embedded mult 1 pipe	17,4	16,5	-5,6
Embedded mult 2 pipe	18,8	18,0	-4,4
Embedded mult 3 pipe	21,8	21,3	-2,0

Tabla 3-4 Mult 54x54.

Tipo de circuito	Medidas (mA)	Estimación (mA)	Error %
LUT	119,6	130,8	9,4
LUT 1 *pipeline*	102,4	97,2	-5,1
LUT 2 *pipeline*	93,0	93,4	0,4
LUT 3 *pipeline*	92,9	90,3	-2,9
LUT 4 *pipeline*	90,3	85,0	-5,9
LUT 5 *pipeline*	83,2	77,9	-6,3
LUT 6 *pipeline*	91,5	87,6	-4,3
Embedded mult	27,5	25,8	-6,2
Embedded mult 1 pipe	26,0	26,7	2,5
Embedded mult 2 pipe	29,9	30,2	0,8
Embedded mult 3 pipe	32,5	32,3	-0,9

Tabla 3-5 Mult 64x64.

Tipo de circuito	Medidas (mA)	Estimación (mA)	Error %
LUT	164,3	185,7	13,0
LUT 1 *pipeline*	149,6	148,5	-0,7
LUT 2 *pipeline*	116,0	120,3	3,7
LUT 3 *pipeline*	102,3	103,2	0,9
LUT 4 *pipeline*	110,8	108,3	-2,2
LUT 5 *pipeline*	110,7	108,9	-1,7
LUT 6 *pipeline*	111,5	107,1	-4,0
Embedded mult	39,5	34,9	-11,7
Embedded mult 1 pipe	33,6	33,2	-1,1
Embedded mult 2 pipe	37,6	36,3	-3,5
Embedded mult 3 pipe	39,2	37,1	-5,4

El primer resultado es que el error de la herramienta de estimación en el peor caso es del 13,0%. Puede verse que el PowerPlay en algunos casos sobreestima y en otros da valores menores para la corriente analizada. El manual publicado por Altera dice textualmente que PowerPlay *"usually provides ±10 percent accuracy when used with accurate design information"* [73].

Un segundo resultado que podemos observar en estos experimentos es un tema que trataremos en profundidad en el capítulo 5.1, la reducción de consumo con el *pipelining* de un circuito. Para realizar una comparación solamente de la reducción del

consumo en los bloques multiplicadores debido al *pipelining* se debe restar el consumo de los circuitos auxiliares (LFSR, XOR, manejador de reloj), que en este caso es 12,11mA.

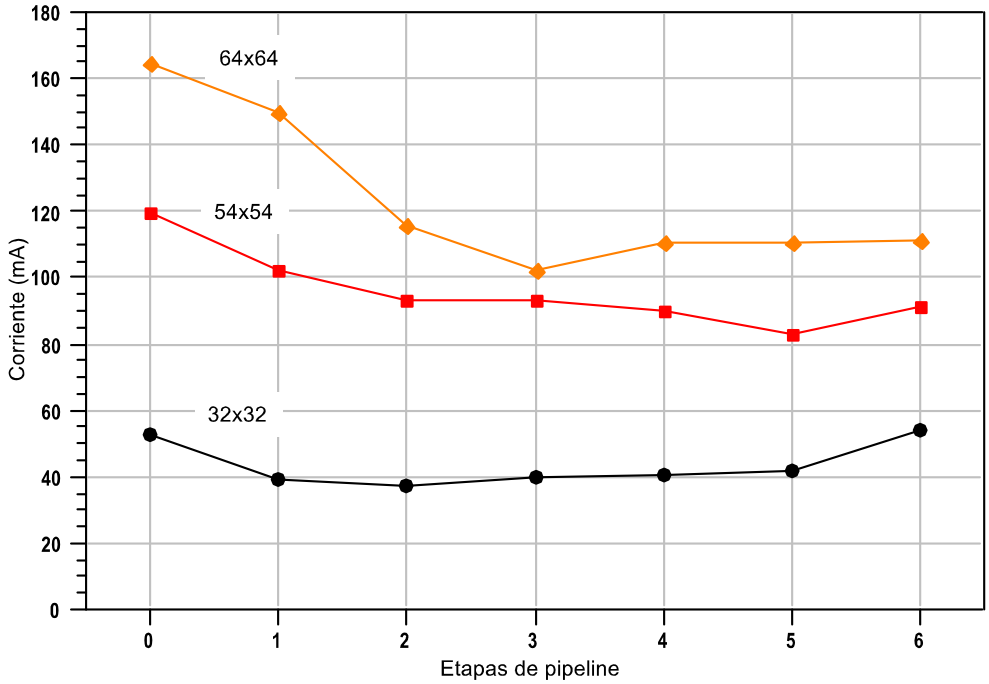


Figura 3-10 Corriente en función de las etapas de *pipeline*

En la Figura 3-10 puede verse cómo se reduce la corriente en función de las etapas de *pipeline* para los 3 tamaños de multiplicadores estudiados. Los mínimos se logran con 2 etapas de *pipeline* en el caso del multiplicador de 32x32 bits, 5 etapas para el de 54x54 y 3 etapas para el multiplicador de 64x64 bits.

Por último podemos comparar los consumos de multiplicadores implementados en lógica, es decir en LUTs comparados con su versión hardware, utilizando los bloques de multiplicación que trae el Cyclone. Esto también será analizado en detalle en el capítulo 5.2 como otra técnica para reducir consumo. En la Figura 3-11 puede verse la corriente para los tres tamaños de multiplicadores combinatorios en su versión LUT y hardware.

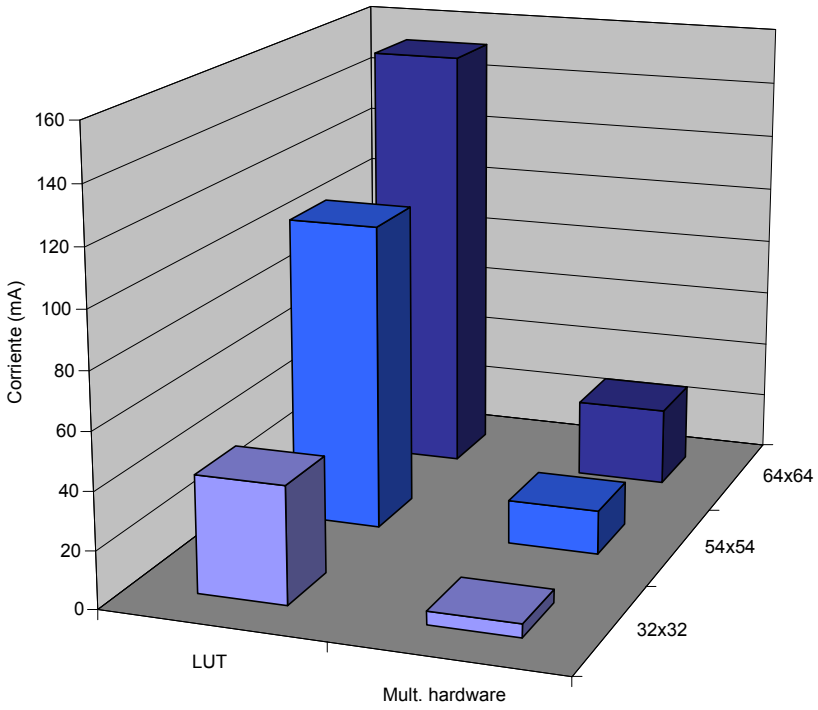


Figura 3-11 Multiplicadores combinatorios LUTs vs. bloques hardware

Como era de esperar la versión hardware de los multiplicadores consume considerablemente menos, pero el ahorro varía según el tamaño, es 8,7 veces menos para el multiplicador de 32x32 bits, 7 veces menos para el de 54x54 y 5,6 veces menos para el de 64x64.

3.6 Medidas vs. estimaciones en Xilinx Spartan-6

Con la plataforma de Avnet Spartan-6 Evaluation Kit se realizan varios experimentos con multiplicadores de diferente tipo para determinar las diferencias entre estimaciones y medidas. En todos los casos se usa la misma circuitería auxiliar para generar entradas, juntar salidas y adaptar relojes.

3.6.1 Trabajo experimental

Nuevamente el objetivo es analizar las diferencias obtenidas entre estimaciones y mediciones de consumo. Las estimaciones se realizan de la forma más exacta posible, utilizando la herramienta XPower Analyzer junto con archivos de actividad de los nodos obtenidos de simulaciones.

Todas las mediciones se realizan en una placa Avnet Xilinx Spartan-6 LX16 Evaluation Kit. Como ya se mencionó esta placa incluye la posibilidad de realizar medidas de consumo ya que cuenta con *shunts* intercalados en las diversas fuentes. El voltaje de estos *shunts* atraviesa un filtro pasa bajos y luego se conecta en forma diferencial a las entradas de un PSoC de Cypress (CY8C38). El PSoC realiza la conversión A/D de la tensión de los *shunts* y de los voltajes de las fuentes y los transmite a un PC vía USB. Del lado del PC existe un programa suministrado por Avnet llamado AvProg, que permite realizar lecturas de valores promediados de las corrientes y los voltajes de cada una de las fuentes de alimentación de la placa. Este mismo programa AvProg permite reconfigurar la Spartan con el archivo .bit de un diseño a través del USB y del PSoC.

Como los bornes de los *shunts* están accesibles en la placa mediante puntos de prueba es posible calibrar todo el sistema de medida, esto es realizado con un multímetro de precisión.

Para automatizar el proceso de medir el consumo de varios diseños se desarrolló un programa en Matlab que sustituye al AvProg. El programa Matlab busca todos los archivos .bit de un directorio, y los va enviando de a uno al PSoC para configurar la Spartan, luego que el diseño está funcionando lee los consumos y los almacena. Esto se repite para cada uno de los archivos .bit correspondientes a los diferentes diseños que se quieren estudiar o caracterizar.

El protocolo de comunicaciones es propietario de Avnet, pero fue posible realizarlo a través de Matlab gracias a que la empresa Silica nos suministró los fuentes tanto del AvProg como del firmware del PSoC.

Como circuitos de prueba se utilizan diferentes versiones de multiplicadores enteros sin signo de ancho de palabra fijo 32x32 (un multiplicador de 64x64 no cabe en el chip, y el de 54x54 cabe pero da errores):

- implementación en LUTs puramente combinatoria
- implementación en LUTs con diferentes etapas de *pipelining*
- implementación en bloques hardware combinatoria
- implementación en bloques hardware con diferentes etapas de *pipelining*

Las entradas a los multiplicadores se generan con LFSRs dentro de la FPGA y se utiliza una XOR para generar una única salida hacia un pin externo, tal como se describe en la sección 3.2. La única entrada externa al Spartan es la señal de reloj de 66.6 MHz.

Todos los casos son simulados durante una gran cantidad de ciclos de reloj, para garantizar que el circuito llega a un régimen estable desde el punto de vista de su consumo, y se generan archivos de actividad del tipo SAIF. Con estos archivos se estima el consumo utilizando la herramienta XPower Analyzer de Xilinx. Esta herramienta genera información detallada de la estimación de la potencia y las corrientes consumidas por las diferentes fuentes que alimentan el chip.

Los resultados obtenidos se muestran en la Tabla 3-6, en la primer columna se encuentra la versión del diseño, luego la corriente medida, otra para la estimada, y en la última el error de estimación.

Las simulaciones se hacen utilizando el simulador Isim que viene con la herramienta de Xilinx ISE 13.1.

Tabla 3-6 Mult 32x32.

Tipo de circuito	Medidas (mA)	Estimación (mA)	Error %
LUT	49,8	140,1	135,4
LUT 1 *pipeline*	49,3	144,3	143,0
LUT 2 *pipeline*	40,7	85,3	72,0
LUT 3 *pipeline*	34,2	55,2	40,7
LUT 4 *pipeline*	33,2	50,8	28,7
LUT 5 *pipeline*	33,7	49,0	27,7
LUT 6 *pipeline*	33,0	49,3	24,8
Embedded mult	15,1	45,7	148.6
Embedded mult 1 pipe	14,9	34,8	93,5
Embedded mult 2 pipe	13,2	21,2	37,7
Embedded mult 3 pipe	11,8	19,7	41,2
Embedded mult 4 pipe	10,9	18,8	39,6
Embedded mult 5 pipe	11,0	18,1	33,9
Embedded mult 6 pipe	11,3	19,4	36,2

La primer conclusión que se puede sacar de estos datos es que en algunos casos la herramienta tiene un error muy grande puede superar el 140%, confirmando así lo publicado en [49]. Pero analizando un poco mejor la gráfica puede verse que el error mayor se da en los casos del multiplicador combinatorio o de pocas etapas de *pipeline*. Esto lleva a pensar que la herramienta está sobrestimando la cantidad de *glitches* que se producen cuando hay combinatoria de mayor profundidad. El simulador Isim no permite un control fino de los *glitches*, solo tiene una opción *-insert_pp_buffers* que evita que se omitan pulsos en la simulación (*control pulse swallowing*), pero no permite controlar el ancho de los *glitches* a filtrar.

La magnitud de los errores hace que se busquen alternativas para hacer funcionar mejor las herramientas de Xilinx. Se prueban diferentes opciones, cambiando las configuraciones del Isim, del Modelo de simulación y las del Xpower Analyzer. De todas las pruebas realizadas la que finalmente da mejores resultados es modificar la variable *"Simulation Model Target"*. Esta variable especifica el lenguaje utilizado para representar el modelo de simulación cuando se realiza la operación *"Generate Post Place & Route Simulation Model"*. Las opciones disponibles son VHDL y Verilog.

Se comprueba que utilizando Verilog en lugar de VHDL el error de estimación se reduce significativamente.

En la Tabla 3-7 se ven los resultados para los diferentes multiplicadores de 32x32, y en las Figura 3-12 y Figura 3-13 se grafican estos datos. Como puede apreciarse la utilización de Verilog mejora muchísimo el error de estimación bajando de 140% al 17%.

Tabla 3-7 Mult 32x32.

Tipo de circuito	Medidas (mA)	Estimación VHDL (mA)	Error %	Estimación Verilog (mA)	Error %
LUT	49,8	140,1	135,4	57,1	-4,0
LUT 1 *pipeline*	49,3	144,3	143,0	56,0	-5,7
LUT 2 *pipeline*	40,7	85,3	72,0	50,2	1,2
LUT 3 *pipeline*	34,2	55,2	40,7	45,7	16,6
LUT 4 *pipeline*	33,2	50,8	28,7	42,8	8,4
LUT 5 *pipeline*	33,7	49,0	27,7	42,4	10,4
LUT 6 *pipeline*	33,0	49,3	24,8	43,1	9,0
Embedded mult	15,1	45,7	148.6	16,8	-8,4
Embedded mult 1 pipe	14,9	34,8	93,5	16,7	-7,1
Embedded mult 2 pipe	13,2	21,2	37,7	16,0	3,9
Embedded mult 3 pipe	11,8	19,7	41,2	15,3	9,5
Embedded mult 4 pipe	10,9	18,8	39,6	15,8	17,0
Embedded mult 5 pipe	11,0	18,1	33,9	15,7	15,9
Embedded mult 6 pipe	11,3	19,4	36,2	15,9	11,8

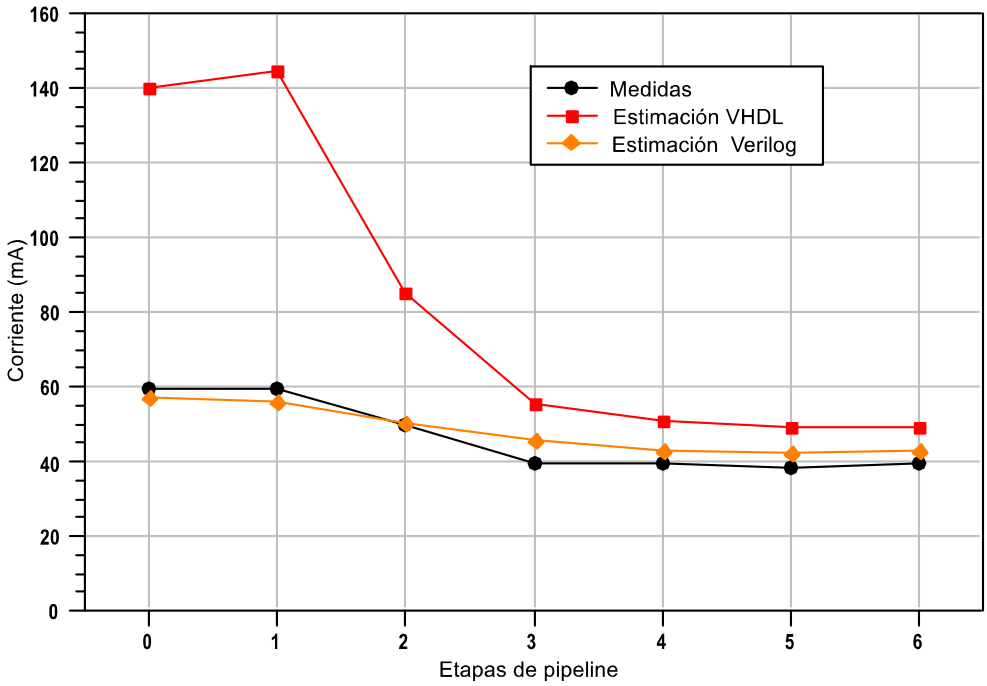


Figura 3-12 Consumo multiplicadores 32x32 en LUTs vs. estimaciones

Pensando que esta podía ser una característica general de las simulaciones con retardos, en las cuales el *netlist* expresado en Verilog simule mejor los glitches que en VHDL para la tecnología Spartan-6, hicimos una consulta a Xilinx y la respuesta fue la siguiente:

> *In ISE -insert_pp_buffers is the switch which gives you the flexibility to control pulse swallowing for simulations. May be the back-annotation of the Verilog model in your case was glitch free as compared to VHDL one but that wont be the case always.*

Si bien Xilinx afirma que esto puede no cumplirse en todos los casos y que depende del circuito particular, en este caso los multiplicadores 32x32, veremos más adelante que se cumple para casi todos los circuitos analizados.

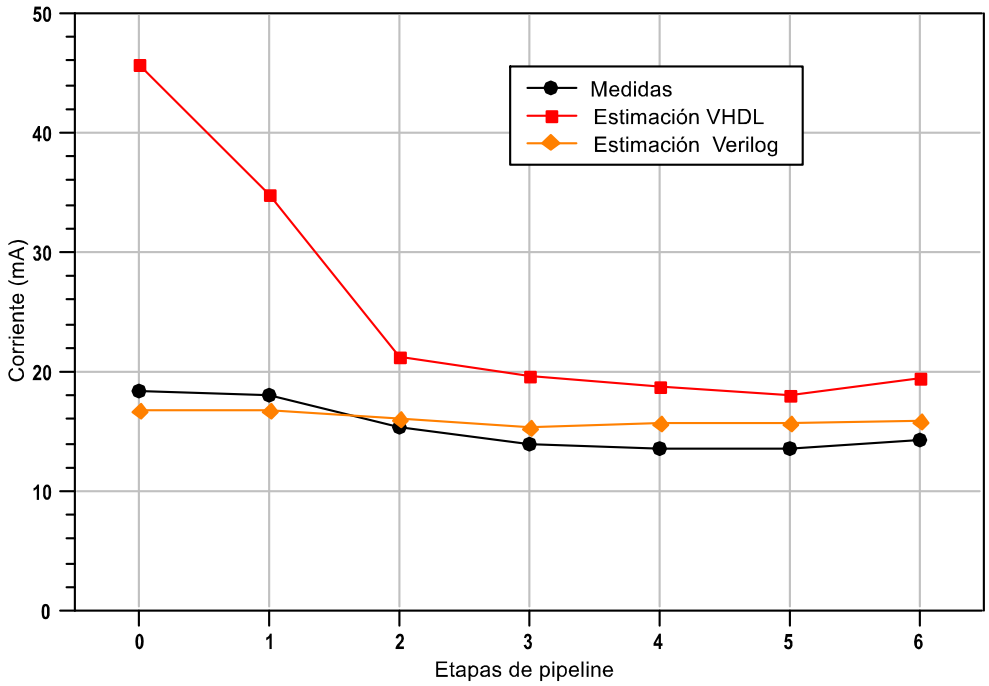


Figura 3-13 Consumo multiplicadores 32x32 en bloques DSP vs. estimaciones

Para verificar que todo esto se debe a la forma de simular los *glitches* se comprueban las formas de onda de la salida del circuito para ambas simulaciones Verilog contra VHDL.

Comparando la Figura 3-14 con la Figura 3-15 puede verse que el modelo en Verilog filtra todos los *glitches* dentro del intervalo de tiempo simulado. Una ampliación del eje de tiempos para ambos casos puede verse en la Figura 3-16 y Figura 3-17. En la Figura 3-16 se aprecia que los *glitches* que aparecen en la simulación pueden ser de diferente ancho.

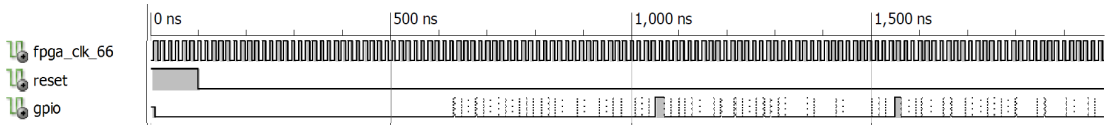


Figura 3-14 Simulación con modelo en VHDL

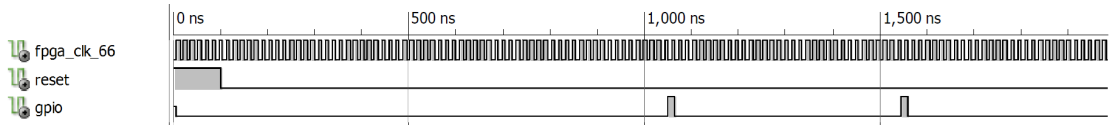


Figura 3-15 Simulación con modelo en Verilog

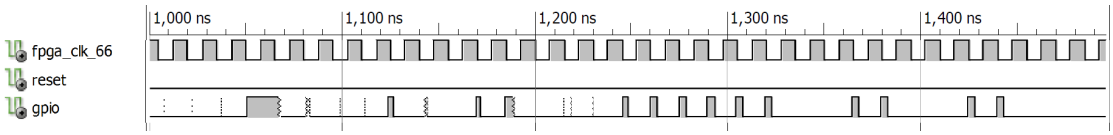


Figura 3-16 Detalle de simulación con modelo en VHDL

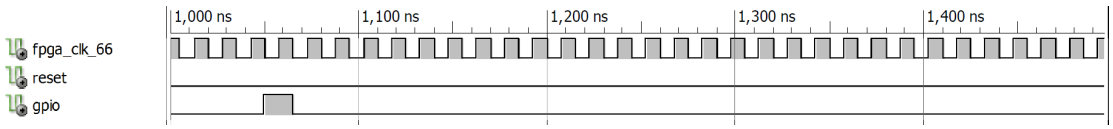


Figura 3-17 Detalle de simulación con modelo en Verilog

3.7 Validación del procedimiento de medida: caracterización de un multiplicador reconfigurable

Para validar todo el procedimiento de medida de consumo se selecciona un circuito complejo, basado en un multiplicador por constantes, pero donde las constantes son reconfigurables en tiempo de funcionamiento. Este tipo de multiplicadores es ideal para aplicaciones en las cuales la adaptación es lenta, por lo tanto las constantes se cambian a un ritmo mucho menor que el funcionamiento del reloj del sistema. El rango de aplicaciones es muy amplio abarcando filtros adaptivos, redes neuronales, ecualización de canales, controles de ganancia, criptografía, etc.

Hay varias arquitecturas propuestas para realizar multiplicadores por constantes, pero no todas ellas se adaptan a la reconfiguración de las constantes en tiempo real. El circuito presentado se basa en la arquitectura propuesta en [74] que presenta un multiplicador por constantes basado en LUTs, y le agrega la capacidad de reconfigurar las constantes en funcionamiento, sin restricciones en el valor de las mismas.

El cambio de las constantes no utiliza la reconfiguración del chip, sino que aprovecha el hecho que en los chips de Xilinx (Spartan-3 y Virtex-4) el contenido de la primitiva LUT4 puede ser visto además como un registro de desplazamiento SRL16E. Esto significa que puede alternar su funcionalidad entre registro de desplazamiento o función combinatoria. Cambiando el contenido del registro de desplazamiento se cambia la función lógica implementada en la LUT4, y de esa manera se termina cambiando la constante del multiplicador. Es decir que la nueva constante debe ingresarse por la entrada serie del SRL16E, y luego realizar 16 desplazamientos para que se almacene en forma correcta. Después de desplazar entonces los 16 ciclos de reloj, se deshabilita el desplazamiento y la LUT4 trabaja nuevamente como función lógica pero ahora con la nueva constante almacenada.

La clave de este método es conocer el contenido de las LUTs para las diferentes constantes con las cuales se quiere utilizar en el multiplicador. Para ello se ha desarrollado una metodología que permite obtener los bits que deben ingresarse en forma serial en los registros de desplazamiento para que luego las LUTs efectúen la multiplicación por una constante determinada.

La descripción detallada del circuito multiplicador y su caracterización en área velocidad y consumo dio origen a una publicación que puede verse en [75].

Para realizar las medidas en este caso hay que utilizar una nueva plataforma de medida, debido a que el multiplicador funciona en chips Xilinx con LUTs de 4 entradas y las de la Spartan-6 tienen 6 entradas. Se utiliza entonces una placa Digilent con un dispositivo Xilinx Spartan 3. Como esta placa no cuenta con la posibilidad de

medir consumo, se realiza el mismo procedimiento explicado en la sección 3.1, removiendo el regulador del 1.2V que alimenta el núcleo de la FPGA, y sustituyéndolo por un regulador externo y un *shunt* serie. Para reducir errores de medida se realiza una calibración del *shunt* junto con las puntas utilizadas para la medida. La tensión en el *shunt* se mide de dos formas diferentes, con un osciloscopio Tektronix TDS3052C que permite registro instantáneo, y con un multímetro Fluke 45. Con este instrumento se llegó a un error relativo menor al 1.5%.

La placa cuenta con un oscilador externo de 50MHz, y se utiliza el DCM para duplicar esa frecuencia y realizar todas las medidas a 100MHz. Para poder llegar a esta frecuencia se debe usar la versión de *pipeline* completa del multiplicador.

Se incluye un LFSR para generar los vectores de entradas, de modo de cambiar los datos del multiplicador en todos los períodos de reloj, y cambiar las constantes cada 1024 períodos.

Se realizan medidas del sistema con y sin multiplicador, de modo de poder restar el consumo de los elementos auxiliares (DCM, LFSR, y función de salida) y obtener así solo el consumo del circuito bajo prueba.

Los resultados se comparan con el multiplicador generado por la herramienta de Xilinx en LUTs (sin utilizar los multiplicadores embebidos). En este caso se utilizan 5 etapas de *pipeline* que es lo recomendado por el fabricante.

Tabla 3-8 Resultados de las medidas de consumo para el multiplicador reconfigurable.

Tamaño	Mult Xilinx LUT (5 etapas *pipeline*)	Mult reconfigurable (*pipeline* completo)	% ahorro
18x18	46.6 mW	45.4 mW	2.6
24x24	73.4 mW	72.3 mW	1.5
32x32	119.1 mW	114.0 mW	4.3

Como se puede ver el multiplicador desarrollado tiene un consumo muy similar al incluido por las herramientas del fabricante, lo cual hace a esta arquitectura muy atractiva porque utiliza mucho menos celdas (un análisis completo del circuito incluyendo comparaciones de área puede verse en la publicación mencionada anteriormente).

3.8 Conclusiones del Capítulo

Queda claro que el consumo es una variable difícil de estimar. En los ejemplos anteriores nos hemos puesto en las mejores condiciones posibles, es decir que hemos hecho una estimación del mejor caso posible, contando con archivos de actividad para todos los nodos del circuito, y sin embargo las herramientas siguen dando un margen de error muy elevado para algunos de los circuitos evaluados.

Como ya mencionamos esto depende de que la simulación de los *glitches* para la tecnología que se está utilizando se ajuste a la real.

Se presenta entonces un problema y es que es imposible saber de antemano si la herramienta de simulación que se está utilizando simula bien o no los *glitches*, por lo tanto se hace inevitable realizar medidas de consumo, al menos para calibrar las herramientas.

Por otro lado, como en los circuitos completamente síncronos bien diseñados (es decir con frecuencias menores a la máxima posible) los *glitches* son irrelevantes, no hay mucho interés en garantizar que las herramientas de simulación tengan bien caracterizadas las diferentes tecnologías en este aspecto.

Los aportes más relevantes de este capítulo son:

- Se han puesto a punto 3 plataformas de medida de consumo, dos con dispositivos de Xilinx y una con Altera.

- Se ha desarrollado una metodología para medir y estimar consumo que incluye la generación de los vectores de entradas en la propia FPGA.
- Se ha desarrollado un automedidor de consumo de bajo costo y fácil utilización.
- Se ha demostrado que los errores de los estimadores comerciales, aún en las mejores condiciones posibles, pueden llegar a invalidar sus resultados.

Capítulo 4

Benchmarks para medidas de consumo

Si bien existen una enorme cantidad de *benchmarks* para FPGAs, el objetivo de este trabajo es encontrar una serie de circuitos que puedan ser utilizados tanto para estimaciones como para medidas de consumo con la metodología desarrollada en el Capítulo 3. Debido a esto, los circuitos a utilizar deben ser sintetizables, y deben poder funcionar en una placa de pruebas con una FPGA con recursos externos mínimos: una señal de reloj, una de reset, y en algún caso alguna entrada extra para poder habilitar el circuito. Además los vectores de entradas a los circuitos deben poder ser generadas internamente tal como se describió en la sección 3.2.

4.1 Antecedentes

Los antecedentes son extensos, se vienen desarrollando o compilando *benchmarks* desde que aparecieron los dispositivos programables y sus herramientas de desarrollo. Los circuitos a ser utilizados fueron creciendo en tamaño a medida que crecían los integrados, y por eso mismo algunas colecciones quedaron rápidamente obsoletas por el crecimiento de los chips. Muchas de las colecciones incluyen las publicadas anteriormente y además agregan algunos circuitos para satisfacer las nuevas necesidades. A medida que los circuitos fueron creciendo en tamaño se hizo necesario

incorporar ejemplos más grandes para que los *benchmarks* siguieran pudiendo ser considerados realistas.

Sin pretender ser exhaustivos un conjunto bastante significativo de *benchmarks* para FPGAs puede verse en la Tabla 4-1.

Los primeros conjuntos de *benchmarks* intentaban evaluar las herramientas de desarrollo, principalmente los algoritmos de síntesis, *place & route*, o el particionado en diferentes chips. Otra aplicación fue evaluar la capacidad de los dispositivos, principalmente en área, y también en frecuencia máxima.

Tabla 4-1 - Benchmarks para FPGAs

Año	Nombre
1985	ISCAS85 [76] International Symposium on Circuits and Systems http://web.eecs.umich.edu/~jhayes/iscas.restore/benchmark.html
1986	Compaction86 The Benchmark Archives at Collaborative Benchmarking and Experimental Algorithmics Laboratory http://www.cbl.ncsu.edu:16080/benchmarks/Benchmarks-upto-1996.html
1988	PRWorkshop88 The Benchmark Archives at Collaborative Benchmarking and Experimental Algorithmics Laboratory http://www.cbl.ncsu.edu:16080/benchmarks/Benchmarks-upto-1996.html
1989	Modgen89 The Benchmark Archives at Collaborative Benchmarking and Experimental Algorithmics Laboratory http://www.cbl.ncsu.edu:16080/benchmarks/Benchmarks-upto-1996.html
1989	ISCAS89 [77] International Symposium on Circuits and Systems [Brglez89] http://web.eecs.umich.edu/~jhayes/iscas.restore/benchmark.html
1989	LGSynth89 The Benchmark Archives at Collaborative Benchmarking and Experimental Algorithmics Laboratory http://www.cbl.ncsu.edu:16080/benchmarks/Benchmarks-upto-1996.html
1990	LGSynth90 The Benchmark Archives at Collaborative Benchmarking and Experimental Algorithmics Laboratory http://www.cbl.ncsu.edu:16080/benchmarks/Benchmarks-upto-1996.html
1991	MCNC91 benchmarks [78] [79] Microelectronics Center of North Carolina, published at the MCNC International Workshop on Logic Synthesis, 1991 http://lyle.smu.edu/~manikas/Benchmarks/MCNC_Benchmark_Netlists.html
1991	LGSynth91 (mismo que MCNC pero otra distribución) The Benchmark Archives at Collaborative Benchmarking and Experimental Algorithmics Laboratory

	http://www.cbl.ncsu.edu:16080/benchmarks/Benchmarks-upto-1996.html
1991	PDWorkshop91 Physical Design Workshop The Benchmark Archives at Collaborative Benchmarking and Experimental Algorithmics Laboratory http://www.cbl.ncsu.edu:16080/benchmarks/Benchmarks-upto-1996.html http://gemini.csc.ncsu.edu:8080/www/benchmarks/PDWorkshop91/
1992	HLSynth92 High Level Synthesis Workshops The Benchmark Archives at Collaborative Benchmarking and Experimental Algorithmics Laboratory http://www.cbl.ncsu.edu:16080/benchmarks/Benchmarks-upto-1996.html
1993	PDWorkshop93 Physical Design Workshops The Benchmark Archives at Collaborative Benchmarking and Experimental Algorithmics Laboratory http://www.cbl.ncsu.edu:16080/benchmarks/Benchmarks-upto-1996.html
1993	Partitioning93 The Benchmark Archives at Collaborative Benchmarking and Experimental Algorithmics Laboratory http://www.cbl.ncsu.edu:16080/benchmarks/Benchmarks-upto-1996.html
1993	LGSynth93 The Benchmark Archives at Collaborative Benchmarking and Experimental Algorithmics Laboratory http://www.cbl.ncsu.edu:16080/benchmarks/Benchmarks-upto-1996.html
1994	PREP Benchmark Suite [80], [81] Programmable Electronics Performance Corporation No disponibles online, la organización ya no existe
1997	RAW Benchmark Suite [82] http://groups.csail.mit.edu/cag/raw/benchmark/README.html
1998	ISPD98 Circuit Benchmark Suite [83] http://vlsicad.ucsd.edu/UCLAWeb/cheese/ispd98.html
1999	Toronto 20 Benchmark suite Los 20 diseños más grandes de MCNC91 http://www.eecg.toronto.edu/~vaughn/challenge/challenge.html
1999	ITC'99 Benchmarks [84] Politecnico di Torino, University of Texas at Austin http://www.cad.polito.it/downloads/tools/itc99.html http://www.cerc.utexas.edu/itc99-benchmarks/bench.html
2002	IWLS 2002 Benchmarks http://www.eecs.berkeley.edu/~alanmi/benchmarks/
2005	IWLS 2005 Benchmarks [85] http://iwls.org/iwls2005/benchmarks.html
2006	UMass RCG HDL Benchmark Collection http://www.ecs.umass.edu/ece/tessier/rcg/benchmarks/
2009	GroundHog Benchmark Suite [86][87][88][89][90] http://cc.doc.ic.ac.uk/projects/GROUNDHOG/
2010	ERCBench: An Open-Source Benchmark Suite for Embedded and Reconfigurable Computing [91] No disponible, el sitio web ya no existe.

Algunos *benchmarks* están descritos en bajo nivel, en lenguajes como BLIF (*Berkeley Logic Interchange Language*) o *netlists* [92], por lo tanto no permiten la utilización de

herramientas de alto nivel para su optimización o el mapeo en arquitecturas más modernas que contienen bloques hardware tales como multiplicadores y acumuladores.

Hay algunos antecedentes de utilización de *benchmarks* para estimaciones de consumo, pero son pocos los que realizan mediciones reales. Algunos trabajos utilizan un único circuito para optimizar su consumo, por ejemplo en [93] se optimiza el consumo de un *turbo decoder* utilizando reconfiguración dinámica, pero no se realizan medidas sino que se utiliza el PowerPlay. En [94] se optimiza el consumo de un *viterbi decoder* también utilizando reconfiguración, pero en este caso los autores sí realizan mediciones.

Existen varios ejemplos de utilización de los benchmarks MCNC para consumo, entre ellos los más destacados son los siguientes: en [37] se utilizan los MCNC benchmarks para incorporar un estimador de consumo a la herramienta de diseño VPR, en [95] se utilizan los mismos benchmarks para estimar actividad y capacidad en rutas, y en [96] se utilizan para estimar el consumo de una FPGA con doble fuente de alimentación simulando con SPICE.

En [52] se utilizan contadores de 4 bits para realizar mediciones y estimaciones ciclo a ciclo, en cambio en [55] y [56] utilizan multiplicadores de diferente tipo para comparar las estimaciones con las medidas, con vectores de entrada suministrados por otra FPGA. En [49] se utilizan diferentes tipos de circuitos de seguridad y encriptado, y se realizan comparaciones entre medidas y estimaciones de consumo con generación interna de vectores de prueba.

En el trabajo [48] se utiliza una serie de diseños que podrían servir como *benchmarks*, pero no se publican porque son extraídos de clientes de Xilinx. Algo similar sucede con [47], que es un trabajo realizado por Altera, aunque utiliza diseños conocidos tampoco se publican tal cual. En este último trabajo se emplean vectores de entradas aleatorios, lo cual no es adecuado para probar algunos de los circuitos utilizados, ya que los mismos no funcionan correctamente con datos aleatorios.

Para poder realizar mediciones de consumo los circuitos deben estar funcionado, por lo tanto hay que suministrarle las entradas adecuadas. Existen básicamente dos formas de hacer esto: una es contar con un generador de estímulos externo, que bien puede ser otra FPGA o un equipo de laboratorio que cuente con un módulo de *Arbitrary Waveform Generator* (AWG); la otra alternativa es generar las entradas internamente dentro de la misma FPGA en donde se tiene el circuito que se quiere probar. En esta última alternativa se puede trabajar con generadores aleatorios (o pseudo aleatorios como los LFSR), o con entradas más complejas generalmente almacenadas en memorias internas o generadas por un *softcore* interno de un microprocesador. Claramente contar con vectores de entradas generados externamente lleva a tener medidas más exactas del circuito bajo prueba, pero implica un banco experimental más complejo con interconexiones, cableados, compatibilizando niveles de tensión, *handshaking*, etc. La alternativa de hacer todo internamente en la misma FPGA resulta muy atractiva, ya que es mucho más sencilla, y siempre se puede restar el consumo de los circuitos auxiliares (generadores de señales, divisores de frecuencia, etc.) para obtener el consumo del circuito bajo prueba.

Hasta la fecha se ha encontrado un único ejemplo de *benchmarks* especialmente diseñados para trabajar con consumo que es el GroundHog [90], desarrollado conjuntamente por el Imperial College de London, Tampere University of Technology y Nokia, este *benchmark* publica también los estímulos necesarios para hacer funcionar los diseños, y en algunos casos un conjunto de *scripts* que sirven para generar vectores de estímulos. Sin embargo el método elegido por GroundHog para ingresar las entradas en el circuito de prueba es externo, utilizando una segunda placa con otra FPGA, e interconectándola con el dispositivo bajo pruebas.

Hay otra publicación [36] que sí utiliza varios circuitos con entradas aleatorias y generadas internamente en la misma FPGA, pero no se hace una descripción de los circuitos ni se publican como *benchmarks* para su reutilización.

4.2 El *benchmark* propuesto

El objetivo entonces es desarrollar un nuevo *benchmark* que sirva para realizar medidas y estimaciones de consumo con la premisa fundamental de su facilidad de uso. Para ello se busca un conjunto de circuitos representativos de los diseños actuales en cuanto a tamaño y aplicaciones, pero que puedan trabajar con estímulos aleatorios generados internamente en la misma FPGA. Para cada circuito se incluye como parte del *benchmark* el generador de entradas, y un *testbench* para poder simularlo generando así los archivos de actividad para cada nodo.

Los circuitos finalmente seleccionados fueron obtenidos de OpenCores, y compilados y probados tanto para Xilinx como para Altera. Se hicieron funcionar y se tomaron medidas en las placas DE0 de Altera y Xilinx Spartan-6 LX16 Evaluation Kit. También se simularon en el mejor caso posible, es decir generando los archivos de estímulos para todos los nodos del circuito, y luego se realizaron estimaciones de consumo con las herramientas de los fabricantes.

4.2.1 Multiplicadores

La utilización de multiplicadores ya fue explicada en el capítulo anterior, es el primer tipo de circuito de prueba utilizado, y es el que genera la inquietud de contar con un conjunto de circuitos de diferentes características para poder ver las dependencias de los resultados con el tipo de circuito.

Los multiplicadores presentan una gran cantidad de ventajas para el estudio de consumo, enumeraremos algunas de ellas:

- Son bloques básicos que vienen incluidos con las herramientas de diseño con la posibilidad de ser parametrizados en varios aspectos.
- Tamaño del circuito variable de acuerdo al ancho de palabra.
- Permiten agregar automáticamente etapas de *pipeline* con lo cual se puede cambiar la profundidad de la lógica.

- Pueden ser implementados en bloques lógicos (LUTs) o en bloques hardware dedicados (DSP).
- Funcionan bien con entradas pseudo aleatorias.

A lo largo de este trabajo se utilizan multiplicadores enteros sin signo, con diferentes anchos de palabra: 32x32, 54x54 y 64x64. Este último tamaño no se pudo usar en la Spartan 6 porque en el chip disponible no cabe. Se compilaron variantes con diferentes etapas de *pipeline*, tanto en lógica (LUTs) como utilizando los multiplicadores embebidos en los integrados.

4.2.2 Circuito de encriptado - desencriptado AES

Este circuito está basado en el proyecto "*High throughput and low area AES core*" publicado en OpenCores [97]. El *core* implementa el algoritmo Rijndael de encriptado y desencriptado utilizado en el estándar AES, además realiza la expansión de la clave cada vez que esta es cambiada o después de un reset. Soporta 3 largos de claves 128, 192 y 256 bits que pueden ser seleccionadas con una señal de entrada. Otra entrada selecciona si el módulo encripta o desencripta.

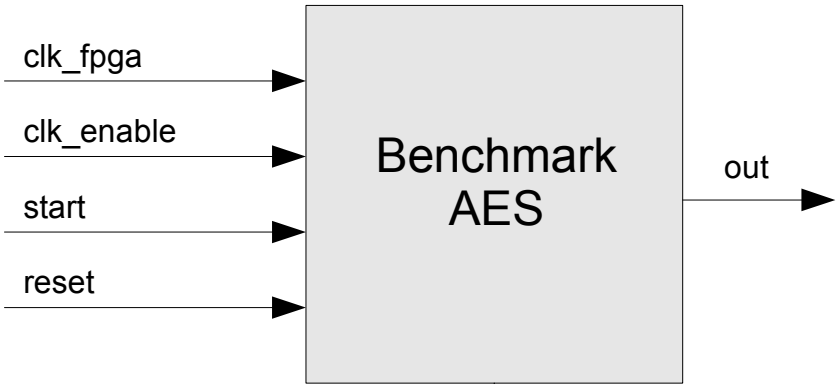


Figura 4-1 Esquema del *benchmark* AES

En este caso se utiliza una versión que encripta datos, con una clave de 256 bits. Se diseñan todos los circuitos auxiliares para el correcto funcionamiento del bloque AES. Esto implica un bloque que genera las entradas pseudo aleatorias basado en un LFSR

de 128 bits, y se diseña también una máquina de estados simple que realiza el manejo de las señales de control del bloque AES. La máquina de estados espera una señal externa (*start*) para comenzar, y luego se encarga de ingresar la clave de 256 bits, eso se logra cargando dos veces los datos que genera el LFSR que son de 128 bits. Posteriormente se espera por la señal de control que indica que la expansión de la clave finalizó; y se comienza a ingresar los datos para el encriptado.

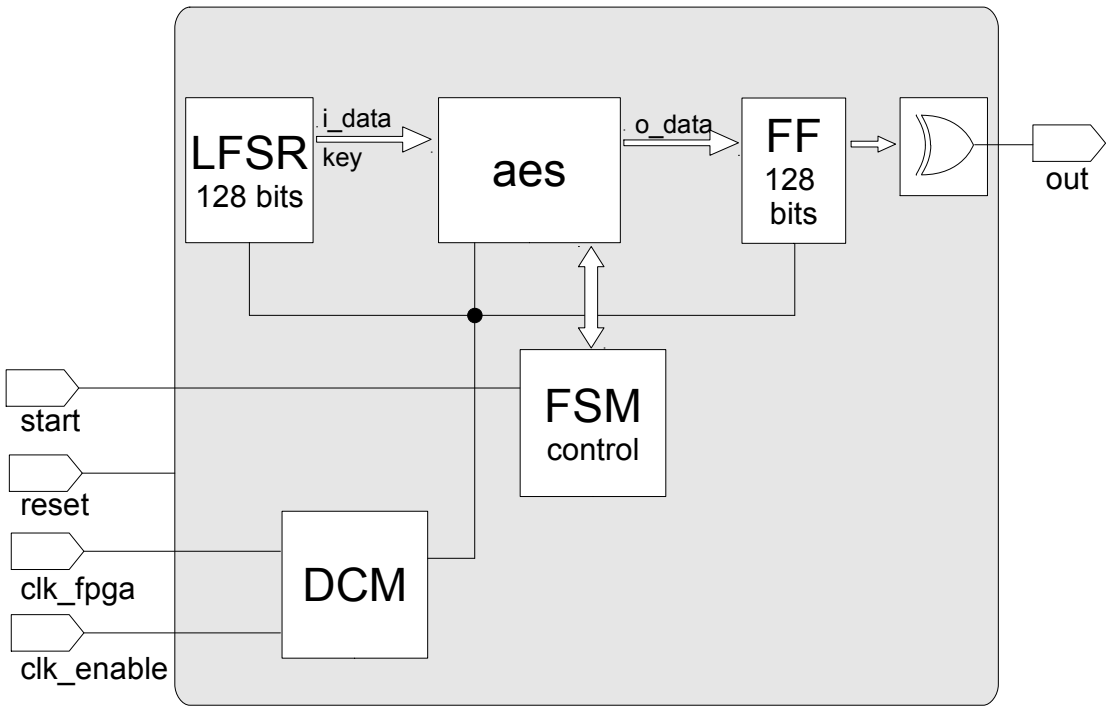


Figura 4-2 Diagrama de bloques del *benchmark* AES

Se utilizan también los DCM o PLL para llevar el reloj de cada placa a una frecuencia determinada. Con estos circuitos auxiliares el core AES puede ser probado en funcionamiento en ambas placas. Se dejan 4 señales externas: *clk*, *clk_enable*, *start* y *reset* para poder controlar el funcionamiento y realizar medidas en diferentes condiciones de actividad.

Para simularlo fue necesario desarrollar un *testbench* que genere la entrada de reloj y maneje las señales de control externas: *reset*, *clk_enable* y *start*.

Para utilizarlo en las placas se debe proveer un reloj externo a la frecuencia deseada, y cablear las demás entradas externas a botones o interruptores. Se logra el funcionamiento correcto activando primero la señal *clk*, luego dando un pulso en la entrada *reset*, y posteriormente generando un pulso en *start*. A partir de allí puede controlarse la actividad del circuito con *clk_enable*.

El *benchmark* AES completo sintetizado en una Spartan 6 (xc6slx16-3csg324) ocupa un 25% de las LUTs, un 7% de los FF, y funciona a 66MHz. En el Cyclone III (EP3C16F484C6) utiliza el 22% de los elementos lógicos, 9% de los FF, 2048 bits de memoria (< 1%), y funciona a una frecuencia de 100MHz.

4.2.3 Circuito de cálculo de la transformada rápida de Fourier (FFT)

Este módulo está basado en el proyecto de OpenCores "*Pipelined FFT/IFFT 256 points processor*" [98]. El bloque es capaz de realizar la Transformada Rápida de Fourier directa e inversa, unidimensional y para números complejos. Utiliza 256 puntos con ancho de palabra de coeficientes y datos parametrizable entre 8 y 16 bits, y opera con *pipeline*. Utiliza los multiplicadores embebidos tanto en Altera como en Xilinx.

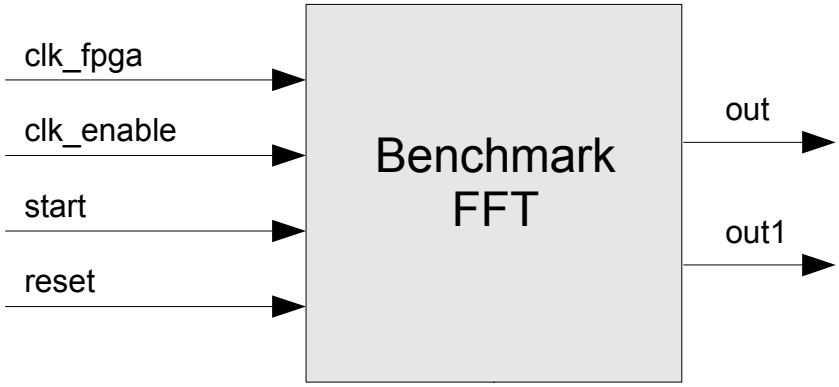


Figura 4-3 Esquema del *benchmark* FFT

Para el caso del *benchmark* se utiliza un ancho de palabra de datos de entrada de 10 bits tanto para la parte entera como para la parte imaginaria. Por lo tanto se agrega un LFSR de 20 bits que genera ambas entradas. Las salidas son números complejos con 14 bits para la parte real e imaginaria.

El circuito tiene una latencia que puede variar entre 580 y 839 ciclos de reloj según la configuración. Debido a esto es que se conecta una segunda salida (*out1*) para poder ver externamente cuando los resultados están disponibles.

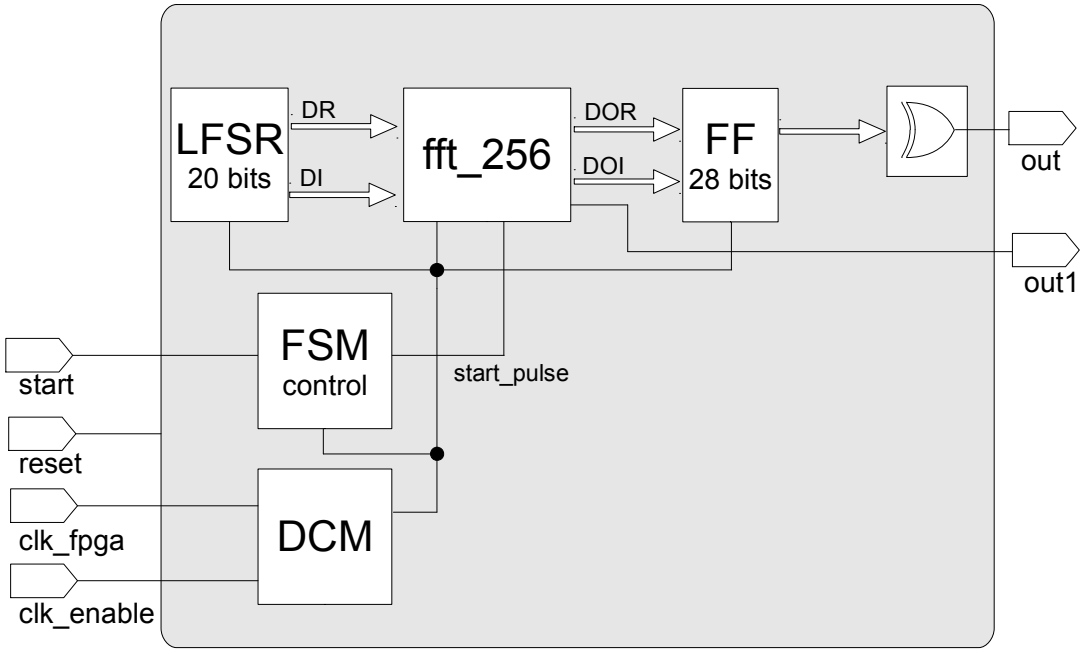


Figura 4-4 Diagrama de bloques del *benchmark* FFT

Se diseña también una máquina de estados simple que permite controlar la señal externa de arranque (*start*), y transformarla en una señal de un período de reloj que es lo que requiere el *core* de FFT.

En la Spartan 6 el *benchmark* FFT utiliza el 58% de las LUTs, el 23% de los FF, 4 bloques multiplicadores DSP48 (12%) y 3 bloques de RAMB16 (9%). El mismo circuito sintetizado en el Cyclone III utiliza el 43% de los elementos lógicos, el 32% de los registros, 8 multiplicadores embebidos (7%) y 42464 bits de memoria (8%).

4.2.4 Circuito que implementa la capa física del estándar IEEE 802.15.4

Basado en el proyecto OpenCores "*IEEE 802.15.4 Core (physical layer)*", [99], [100], que implementa la capa física de redes inalámbricas de baja velocidad (*Low-Rate Wireless Personal Area Networks* - LR-WPANs). Este protocolo es la base para sistemas de redes de sensores inalámbricos tales como ZigBee.

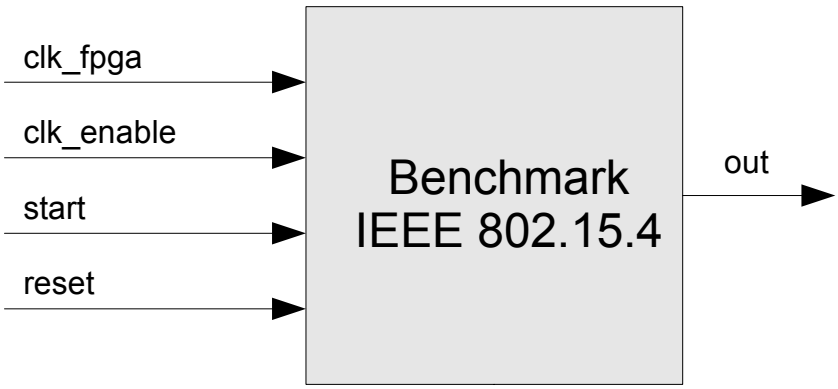


Figura 4-5 Esquema del *benchmark* IEEE 802.15.4

Para utilizar el *benchmark* se conecta en cascada la cadena de transmisión con la de recepción. El circuito original funciona con dos relojes: 8MHz y 1MHz, pero por dificultades para generar internamente esas frecuencias tan bajas en las placas de desarrollo utilizadas, las mismas se duplican. Es decir que en el *benchmark* se utilizan relojes de 16MHz y 2MHz (aunque en circuito no respetaría los tiempos para su aplicación real).

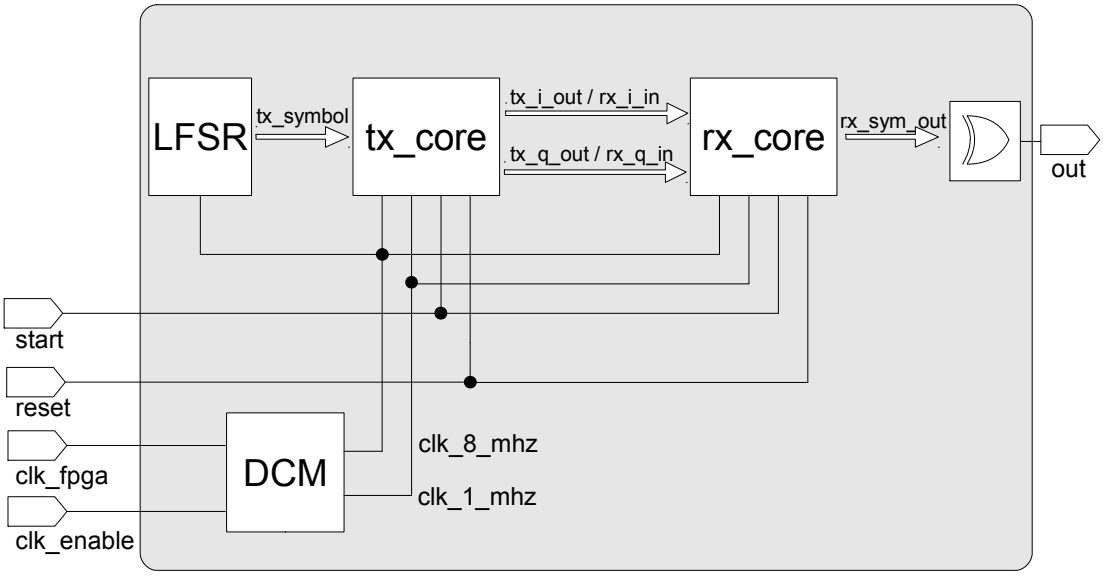


Figura 4-6 Diagrama de bloques del *benchmark* IEEE 802.15.4

Se utiliza un LFSR para generar los símbolos de entrada (*tx_symbol*) que son de 4 bits. El bloque de transmisión genera las salidas I y Q de 10 bits de ancho, que se concatenan con el bloque de recepción. La salida final del bloque de recepción (*rx_sym_out*) es nuevamente de 4 bits.

Los resultados de la síntesis para el *benchmark* IEEE802.15.4 muestran que en la Spartan 6 ocupa el 4% de los FF y el 23% de las LUTs; y para el Cyclone II un 2% de los registros, un 9% de los elementos lógicos y 20 bits de memoria.

4.2.5 Microcontrolador openMSP430

Sistema con el microcontrolador de 16 bits openMSP430 de OpenCores [101], este microcontrolador es compatible con el MSP430 de la empresa Texas Instruments, ampliamente utilizado en sistemas embebidos y aplicaciones de bajo consumo. La compatibilidad permite ejecutar código generado con herramientas de desarrollo del MSP430.

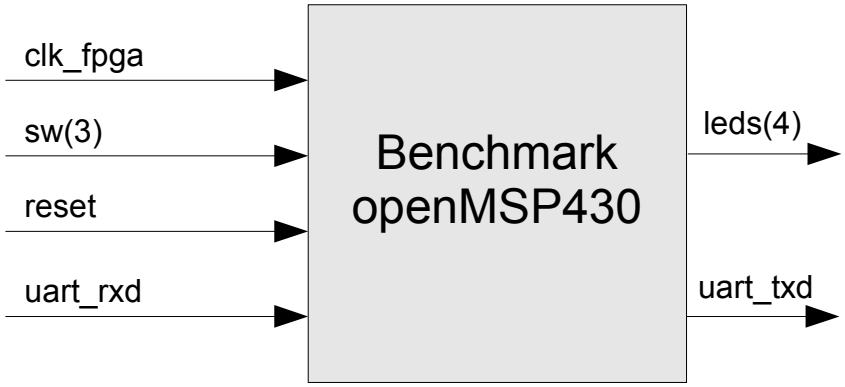


Figura 4-7 Esquema del *benchmark* openMSP430

En el *benchmark* se utiliza un programa de inversión de matrices funcionando en *loop* infinito. En este caso no es necesario utilizar los generadores de vectores pseudoaleatorios LFSR.

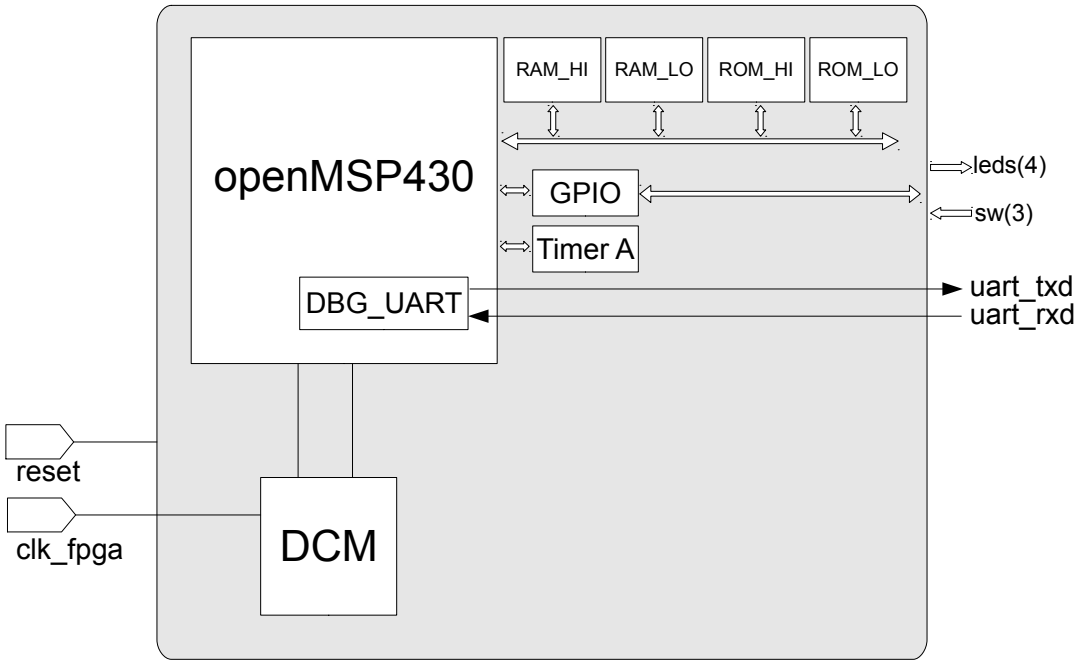


Figura 4-8 Diagrama de bloques del *benchmark* openMSP430

El sistema completo incluye el generador de relojes, memoria de datos y de programa, el bloque que maneja entradas y salidas de propósito general, un timer, y la UART de *debugging*.

Las entradas de propósito general se conectan a botones o *switches* de la placa, y las salidas a LEDs.

El código del programa puede cargarse directamente en la memoria al configurar el chip, o puede enviarse por un puerto serie a través de la unidad de *debugging*.

El microcontrolador openMSP430 consume relativamente pocos recursos, en la Spartan 6 utiliza el 5% de los FF, el 20% de las LUTs, 4 bloques de memoria y un multiplicador hardware. En el Cyclone III utiliza el 20% de los elementos lógicos, el 7% de los FF, 2 bloques multiplicadores y 5K bytes de memoria.

4.2.6 Resumen de los circuitos del *benchmark*

En las tablas siguientes se presenta un resumen de los recursos que ocupan cada uno de los circuitos de benchmark completos y su frecuencia de funcionamiento.

Tabla 4-2 - Implementación en Xilinx Spartan-6

Spartan-6 (xc6slx16-3csg324)						
	LUT	FF	Embedded Mult	RAM Blocks	Clock Blocks	MHz
AES	2343 (25%)	1419 (7%)	0	0	1	66
FFT	5314 (58%)	4278 (23%)	4	3	1	100
IEEE802154	2107 (23%)	809 (4%)	0	0	1	16/2
openMSP430	1847 (20%)	1006 (5%)	1	4	2	20

Tabla 4-3 - Implementación en Altera Cyclone III

Cyclone III (EP3C16F484C6)						
	LE	FF	Embedded Mult	Memory bits	Clock Blocks	MHz
AES	3360 (22%)	1457 (21%)	0	2048	1	100
FFT	6596 (43%)	4923 (32%)	8	42464	1	100
IEEE802154	1349 (9%)	365 (2%)	0	20	1	16/2
openMSP430	3147 (20%)	1146 (7%)	2	40960	1	24

4.3 Medidas y estimaciones utilizando los *benchmarks*

Los *benchmarks* recopilados facilitan mucho la realización de mediciones de consumo de circuitos en funcionamiento, además como incluyen un *testbench* también permiten simular registrando la actividad por nodo, y a partir de allí realizar estimaciones de consumo. La idea es brindar una herramienta de fácil utilización que permita al usuario explorar el espacio de diseño: área, velocidad y consumo; pero garantizando que el consumo puede ser medido y no solo estimado.

Se presentan algunos ejemplos de medidas y estimaciones utilizando los *benchmarks* que confirman los datos que se obtuvieron con multiplicadores en las secciones 3.5 y 3.6.

Se comienza comparando medidas y estimaciones en Spartan-6, los datos se encuentran en la Tabla 4-4. Las simulaciones son realizadas con el ISim, herramienta que viene incluída con el ISE de Xilinx. La herramienta de Xilinx no permite control de *glitches*, pero sí se puede elegir en qué lenguaje se va a expresar el netlist final del circuito para ser simulado, que puede ser VHDL o Verilog.

Como puede observarse simulando con *netlists* en VHDL los errores pueden ser muy grandes, llegando al 377% para el caso del circuito IEEE 802.15.4, pero mejoran notoriamente en el caso de utilizar Verilog. La simulación con Verilog estima muy bien en los primeros 3 casos, aunque mantiene errores demasiado grandes como para trabajar sin realizar medidas en los casos del openMSP430 y del IEEE 802.15.4.

Tabla 4-4 - Medidas y estimaciones con los *benchmarks* en Xilinx Spartan-6

	Consumo medido (mW)	Estimación VHDL (mW)	Estimación Verilog (mW)	Error estimación VHDL	Error estimación Verilog
mult 32x32 LUT comb	57,5	140,06	57,1	144%	-1%
AES	90,7	206,4	86,1	128%	-5%
FFT	218,2	277,3	235,4	27%	8%
IEEE802.15.4	14,3	68,0	23,2	377%	63%
openMSP430	21,9	29,5	26,7	35%	22%

En el caso de Altera la simulación y la estimación del consumo permiten un doble control o filtrado de *glitches*, por lo tanto se pueden analizar mejor los problemas de las estimaciones y su relación con la estimación de los *glitches* de la tecnología en un circuito determinado. Para simular se utiliza ModelSim, y para generar el archivo de actividad de los nodos se usa el formato VCD (*Value Change Dump*). Este formato genera archivos mucho más grandes que el SAIF utilizado en las herramientas Xilinx, pero es el que funciona bien en Quartus, ya que el SAIF generado por el ModelSim da problemas de formatos en la versión utilizada.

El primer control de filtrado de *glitches* es en la ventana de diálogo de simulación de Quartus, cuando se llama al ModelSim se puede activar o no un filtrado de *glitches* al generar el archivo VCD. Luego, cuando se invoca a la herramienta Power Play Power Analyzer y se suministra el archivo VCD para que extraiga la actividad de los nodos existe un segundo control de filtrado de *glitches* existentes en el archivo VCD.

En la Tabla 4-5 pueden verse los resultados en el caso de Altera, en general los resultados son mejores si se realiza algún tipo de filtrado de *glitches*, pero esto no se da en todos los casos.

Tabla 4-5 - Medidas y estimaciones con los *benchmarks* en Altera Cyclone III

	Filtrado de *glitches* en la generación del archivo VCD	Filtrado de *glitches* en la estimación usando el archivo VCD	Estimación (mW)	Medida (mW)	Error
Mult 54x54 LUT comb.	OFF	OFF	98,9	65,6	51%
	OFF	ON	50,9	65,6	-22%
	ON	OFF	70,7	65,6	8%
	ON	ON	51,9	65,6	-21%
AES	OFF	OFF	189,2	145,8	30%
	OFF	ON	135,1	145,8	-7%
	ON	OFF	157,1	145,8	8%
	ON	ON	137,0	145,8	-6%
FFT	OFF	OFF	252,5	217,2	16%
	OFF	ON	231,7	217,2	7%
	ON	OFF	245,2	217,2	13%
	ON	ON	231,0	217,2	6%
IEEE 802.15.4	OFF	OFF	26,1	22,0	18%
	OFF	ON	18,8	22,0	-15%
	ON	OFF	20,0	22,0	-9%
	ON	ON	18,9	22,0	-14%
openMSP430	OFF	OFF	40,6	30,0	35%
	OFF	ON	34,6	30,0	16%
	ON	OFF	35,6	30,0	19%
	ON	ON	31,8	30,0	6%

De estos experimentos se pueden extraer algunas conclusiones interesantes. La primera es que para las herramientas de Xilinx se confirma algo que ya se había observado en el caso de los multiplicadores, y es el altísimo error de estimación cuando se utiliza un *netlist* en VHDL. Este error disminuye considerablemente pasando a utilizar modelos en Verilog, aunque en algunos casos igual es muy elevado: 65% en el caso del circuito IEEE 802.15.4.

Las herramientas de Altera parecen funcionar un poco mejor, con errores del 51% en el peor caso si no se controlan *glitches*, pero que bajan a -22% a 19% si se incluye alguna de las opciones de filtrado.

Un problema adicional se presenta si se obtienen errores grandes pero negativos, ya que, según la convención adoptada a lo largo de esta tesis, esto significa que el consumo estimado está por debajo del real; y esto podría llevar al sub dimensionado de las fuentes de alimentación de la FPGA.

Por lo expresado anteriormente, se puede afirmar que las herramientas de estimación de consumo comerciales utilizadas presentan errores demasiado grandes como para poder trabajar con ellas sin realizar algún tipo de calibración o ajuste. Esto confirma una vez más la importancia y la necesidad de realizar mediciones de consumo.

4.4 Medidas utilizando el SEM-FPGA

Para validar el sistema de automedida desarrollado en la sección 3.3 se realizan una serie de pruebas con la placa DE0. La señal de salida del automedidor, cuya frecuencia es proporcional a la corriente consumida, se ingresa al FPGA en un pin GPIO. Luego, se instancia el módulo SEM implementado en VHDL y se asigna a la entrada INPUT el pin GPIO donde se conecta la señal del medidor externo. Además de instanciar el módulo SEM se debe instanciar el circuito que se desea medir.

Los datos se almacenan en la RAM interna de la FPGA, una forma cómoda de extraerlos es utilizar la herramienta proporcionadas por Altera: "*In System Memmory Content Editor*", que permite leer los datos desde un PC vía USB y almacenarlos en un archivo en formato HEX. Se desarrolló un programa en Matlab que lee esos datos y los procesa utilizando las constantes de calibración del sistema para obtener así la corriente medida en función del tiempo.

Las dos primeras medidas se realizan con cada uno de los módulos SEM solos, es decir sin instanciar otros circuitos. Esto es para caracterizar las herramientas diseñadas, y ver cuanto va a afectar en las medidas. Estos consumos incluyen el consumo estático de la FPGA y el consumo del SEM, y deben restarse en el caso de estar analizando otros circuitos.

En la Figura 4-9(a) se observa la variación de la frecuencia de la señal INPUT en función del tiempo mientras que la Figura 4-9(b) muestra la corriente consumida, luego de hacer la conversión de frecuencia a corriente con la curva de calibración, para la primera versión implementada del SEM (SEM periódico).

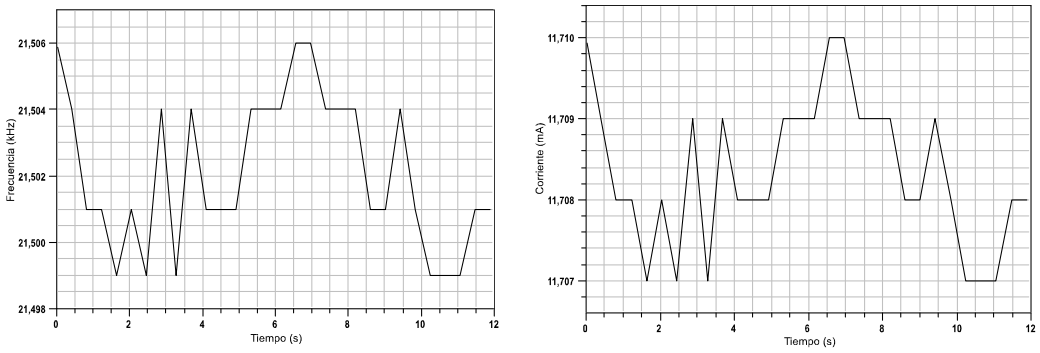


Figura 4-9 Consumo del SEM periódico: (a) Frecuencia vs. tiempo (b) Corriente vs. tiempo

Las medidas de las dos versiones del SEM, SEM periódico y SEM asíncrono sin otro circuito instanciado se muestran en la Tabla 4-6

Tabla 4-6 Medidas de las dos versiones del SEM sin otro circuito instanciado.

Circuito	Medidas con SEM (mA)	Medida con multímetro (mA)
SEM periódico	11,7	11,45
SEM asíncrono	11,7	11,52

Como puede verse el SEM tal como fue implementado tiene pequeñas variaciones de consumo que son registradas en las gráficas.

La siguientes medidas realizadas consisten en instanciar el SEM junto a un circuito al cual se le quiere medir el consumo. Se utilizan dos de los circuitos de *benchmark* para esto: el openMSP430 y el multiplicador de 32x32 bits. Los resultados promedio pueden verse en la Tabla 4-7, pero más interesante es ver las gráficas de consumos en función del tiempo.

Tabla 4-7 Medidas realizadas con el SEM.

Circuito	Medidas con SEM (mA)	Medida con multímetro (mA)
openMSP430 + SEM	31,2	32,9
Mult 32x32 + SEM	54,6	58,8

En la Figura 4-10 se muestra la corriente consumida en función del tiempo por el sistema completo con el microcontrolador openMSP430 en conjunto con la primera versión implementada del SEM. Se puede ver que el consumo promedio es de 33.1 mA. Al cabo de unos segundos del experimento, se presiona el botón de *reset* del openMSP430, se puede observar en la gráfica como el consumo baja durante el tiempo que está presionado el *reset*.

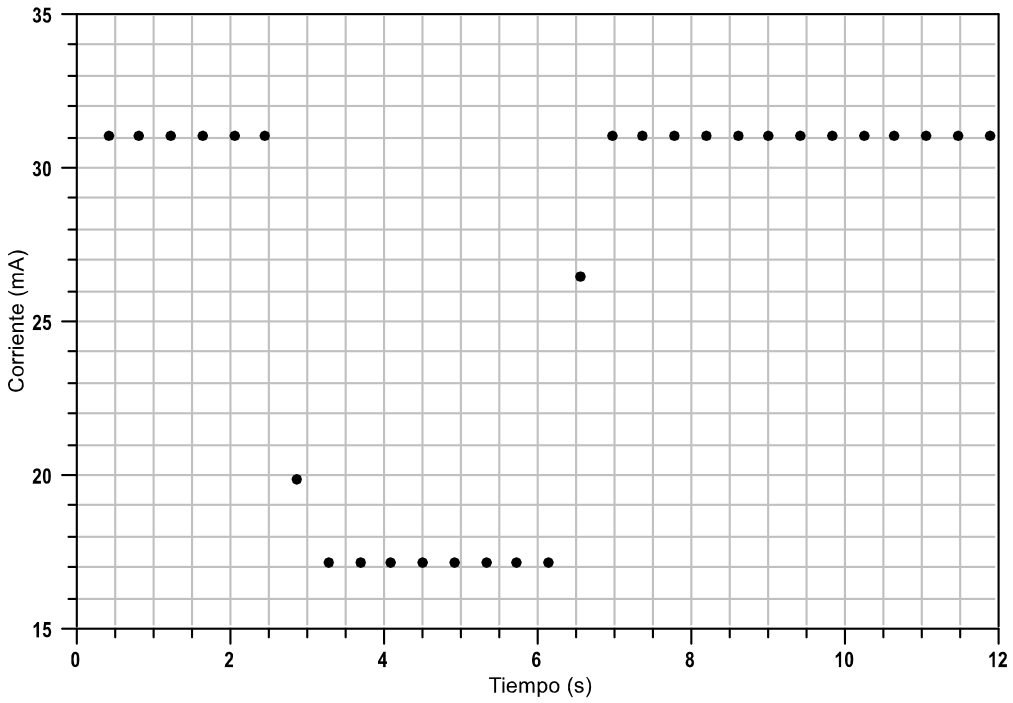


Figura 4-10 Corriente consumida por el openMSP430

El último experimento es el más interesante, se implementa un multiplicador de 32x32 bits con *clock enable* (CE) y *clock gating* (CG) de modo de poder observar las variaciones del consumo. En esta medida se desea mostrar como cambia el consumo

medido con el SEM al cambiar el consumo real durante un experimento. Para lograrlo, se utilizan *switches* externos para manejar las señales de CE y CG, y se cambian en tiempo real mientras se registra el consumo del circuito. En la Figura 4-11 se muestran los resultados obtenidos. El primer nivel es con el multiplicador funcionando normalmente, el nivel intermedio es activando CE, y el nivel de menor consumo es cuando se activa CG.

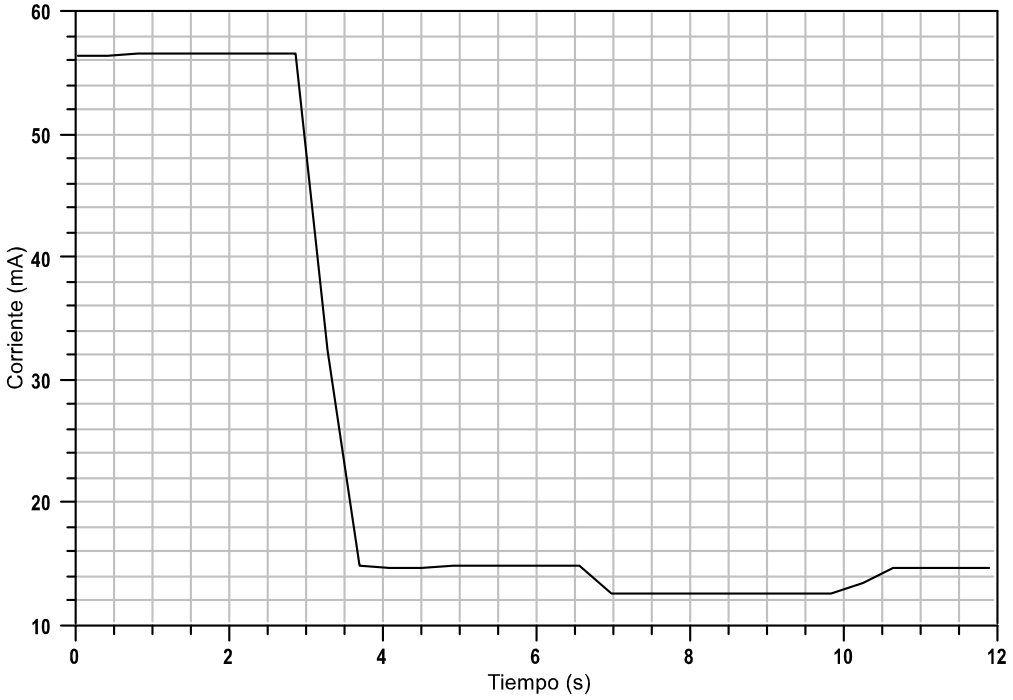


Figura 4-11 Medida con el SEM con Mult 32x32 con *clock gating*

Finalmente se puede concuir que el SEM desarrollado es una herramienta muy poderosa para estudiar consumos, sobre todo en consumos dinámicos y para extraer perfiles de consumo.

4.5 Conclusiones del Capítulo

Los resultados más importantes de este capítulo son:

- La selección de un conjunto de circuitos para ser utilizados como *benchmarks* de consumo en FPGAs.

- Estos circuitos son de código abierto, por lo tanto su utilización es libre.
- Los mismos han sido sintetizados y puestos en funcioniamiento en FPGAs de Altera y Xilinx.
- Se realizaron medidas y estimaciones de consumo para todos los circuitos en las dos plataformas disponibles.
- Se confirma que las herramientas comerciales utilizadas para estimación de consumo presentan errores demasiado grandes como para poder usarlas sin algún tipo de calibración.
- Se ha validado el automedidor de consumo desarrollado.

Capítulo 5

Aplicación de técnicas de bajo consumo

En este capítulo se exploran en forma experimental algunas técnicas de reducción de consumo. La incorporación de éstas técnicas puede ser realizada en varias de las etapas del proceso de diseño, y se pueden atacar diferentes objetivos, como por ejemplo actuar sobre la potencia dinámica o la estática, reducción de potencia o energía, etc.

Existe muy buen material bibliográfico sobre estos temas, aunque la mayor parte es dedicado al diseño digital de ASICs. No se hará en esta sección una revisión exhaustiva de la misma, pero pueden verse libros ya clásicos como Chandrakasan [102], Rabaey y Pedram [21], [103], o Piguet [5]. Sin embargo la bibliografía dedicada específicamente a técnicas de bajo consumo en FPGAs es muchísimo más escasa: se reduce a dos libros [104] y [105], y parte de un tercero [106].

Si bien en términos generales las teorías y técnicas desarrolladas para ASICs son válidas en FPGAs, a nivel de usuario solo pueden aplicarse un sub conjunto reducido de ellas dado que el chip ya está fabricado y el diseñador solo puede configurarlo.

En la Tabla 5-1 pueden verse las diferentes opciones de reducción de consumo a tener en cuenta en cada nivel de abstracción de un diseño (la tabla está hecha para desarrollo de ASICs y no para FPGAs).

Es aceptado ([107], [108]) que los ahorros de consumo mayores se dan en los niveles más altos de abstracción, es decir cuando se está diseñando el sistema completo; y que a bajo nivel, no se puede llegar a reducciones espectaculares. Es en el nivel de sistema en donde se toman opciones tales como incluir modos de *sleep* o *standby*, el apagado parcial de algunos bloques del circuito, qué tipos de algoritmos a utilizar, qué hacer en hardware y qué hacer en software; y este tipo de opciones son las que impactan muy fuertemente en el consumo final.

Tabla 5-1 Oportunidades de reducción de consumo en cada nivel de abstracción.

Nivel de abstracción	**Decisiones a tomar, oportunidades para reducir consumo**	**I (2003)**	**II (1998)**
Diseño del sistema	Algoritmos, partición software-hardware, interfaces, manejo de memoria, utilización de bloques IP	75%	10-20x
Arquitectura	Paralelismo, *pipelining*, zonas con diferentes voltajes, modos de bajo consumo	50-75%	
RTL	*Clock gating*, *power gating*	15-50%	2-5x
Circuito (*Gate Level*)	Mapeo tecnológico, optimizaciones a nivel de compuerta, distribución de reloj, reducción de *glitches*	5-15%	20-50%
Dispositivo (*Transistor Level*)	Tamaño de transistor, proceso de fabricación, Vth, encapsulado, pin-out	3-5%	

La Tabla 5-1 muestra además valores típicos de reducción de consumo en cada nivel de jerarquía del diseño. Los valores de la columna I fueron obtenidos de [107], una publicación del año 2003, que no indica el origen de los datos. Los valores de la columna II son de [108], publicados en 1998, y repetidos en [109] en 2012. Aparentemente están basados en los ejemplos que aparecen en publicaciones anteriores de 1995 [28] y 1996 [103]. Esta información entonces si bien da una idea de posibilidades de reducción de consumo en cada etapa de un diseño, la misma está basada en circuitos de 10 o 20 años atrás, y no se han encontrado datos actualizados.

Las técnicas de alto nivel en general están muy atadas a una determinada aplicación, por lo tanto escapan al alcance de esta tesis. Las técnicas que se detallan en este capítulo son a nivel de arquitectura y de circuito, se incluyen las siguientes:

- *Pipelining* o segmentación de circuitos.
- Utilización de bloques hardware.
- *Clock gating*, y *data gating.*
- Influencia de las herramientas.
- Caso de estudio el openMSP430.

5.1 *Pipelining*

Es bien conocida la aplicación de la técnica del *pipelining* o segmentación para incrementar la velocidad de funcionamiento de sistemas o circuitos. Esta misma técnica se ha aplicado también con éxito para bajar consumo.

Históricamente las primeras ideas de usar *pipelining* fueron para bajar el consumo disminuyendo la tensión de alimentación de los circuitos, y compensar la pérdida de velocidad con etapas de *pipelining*. Pero en realidad el *pipelining* funciona por sí mismo bajando el consumo por otro motivo que es la reducción de *glitches*. Esta reducción de glitches es debida a dos razones: La primera es que al insertar barreras de registros en el circuito, los *glitches* no pueden propagarse más allá del *flip-flop* de la etapa siguiente. La segunda es que al disminuir la profundidad de la lógica combinatoria, la generación de *glitches* también es menor, ya que hay menos posibilidades que se produzcan retardos diferentes entre ramas de un circuito.

Está claro que esto también tiene un límite, porque es una técnica que agrega una enorme cantidad de *flip-flops* al sistema (más de 4000 en los ejemplos estudiados), y dicho límite se alcanza cuando el consumo de los *flip-flops* agregados supera al consumo ahorrado por la reducción de *glitches*. Un análisis en profundidad de la utilización del *pipelining* como técnica de ahorro de consumo puede verse en [110].

Si bien los primeros trabajos se realizaron en ASICs CMOS [111], [112], el *pipelining* resulta particularmente atractivo en FPGAs, ya que las celdas incluyen una LUT y un *flip-flop*. Por lo tanto los *flip-flops* ya existen y no hay que agregarlos, de modo que no hay un costo extra asociado. Esto llevó a que se publicaran una enorme cantidad de trabajos de *pipelining* en FPGAs: [113], [43], [114], [115], [15], [116], [53], [117], [118], [119], [120], [121], [122].

También existen limitaciones de la aplicación de esta técnica. Por un lado, no todos los algoritmos admiten *pipelining*. Pero además, la latencia extra que agrega esta técnica, hace que a veces su utilización no sea práctica.

Utilizando la metodología de medida presentada en la sección 3.2 se muestran ejemplos de la aplicación de *pipelining* en dispositivos Altera 65-nm Cyclone III EP3C16F484 y Xilinx 45-nm Spartan-6 XC6SLX16-2. Los circuitos utilizados en todos los casos son multiplicadores de enteros sin signo con diferente ancho de palabra (32x32, 54x54 y 64x64). La única entrada externa utilizada en todos los casos es la señal de reloj, ya que los datos de los multiplicadores son generados con un LFSR (*linear feedback shift register*).

El consumo se obtiene con medidas de corriente en resistencias *shunt*, repitiendo las medidas y promediando para bajar el error relativo, que se estima menor a un 1.5%. Las Tabla 5-2 y Tabla 5-3 muestran los resultados de las medidas, y se incluye un factor de reducción de consumo, tomando como unidad el valor medido para el circuito sin *pipeline* o puramente combinatorio.

Tabla 5-2 Resultados en Cyclone III, 65 nm.

	mult 32×32 50 MHz		
Etapas de *pipeline*	**mA**	**FF**	**Factor de reducción de consumo**
1	52.9	128	1.00
2	39.2	541	0.74
3	37.4	822	0.71
4	40.1	1001	0.76
5	40.5	1300	0.77
6	41.9	1458	0.79
7	54.2	1524	1.02

	mult 54×54 50 MHz		
Etapas de *pipeline*	**mA**	**FF**	**Factor de reducción de consumo**
1	119.6	188	1.00
2	102.4	953	0.86
3	93.0	1509	0.78
4	92.9	1776	0.78
5	90.3	2338	0.76
6	83.2	2672	0.70
7	91.5	3456	0.77

	mult 64×64 50 MHz		
Etapas de *pipeline*	**mA**	**FF**	**Factor de reducción de consumo**
1	164.3	256	1.00
2	149.6	1181	0.91
3	116.0	1718	0.71
4	102.3	2401	0.62
5	110.8	3144	0.67
6	110.7	3773	0.67
7	111.5	4208	0.68

Tabla 5-3 Resultados en Spartan-6, 45 nm.

	mult 32×32 50 MHz		
Etapas de *pipeline*	**mA**	**FF**	**Factor de reducción de consumo**
1	43.2	128	1.00
3	34.1	289	0.79
5	26.6	1103	0.62
7	26.4	1182	0.61
9	---	---	---

	mult 54×54 50 MHz		
Etapas de *pipeline*	**mA**	**FF**	**Factor de reducción de consumo**
1	67.2	216	1.00
3	34.9	666	0.52
5	27.9	2346	0.42
7	---	---	---
9	23.1	3206	0.34

Como puede verse en la Figura 5-1 en algunos casos el consumo vuelve a subir si se incrementan demasiado las etapas de *pipeline*, pero en todos los casos se logran factores de entre 0.71 y 0.34 con respecto al circuito original.

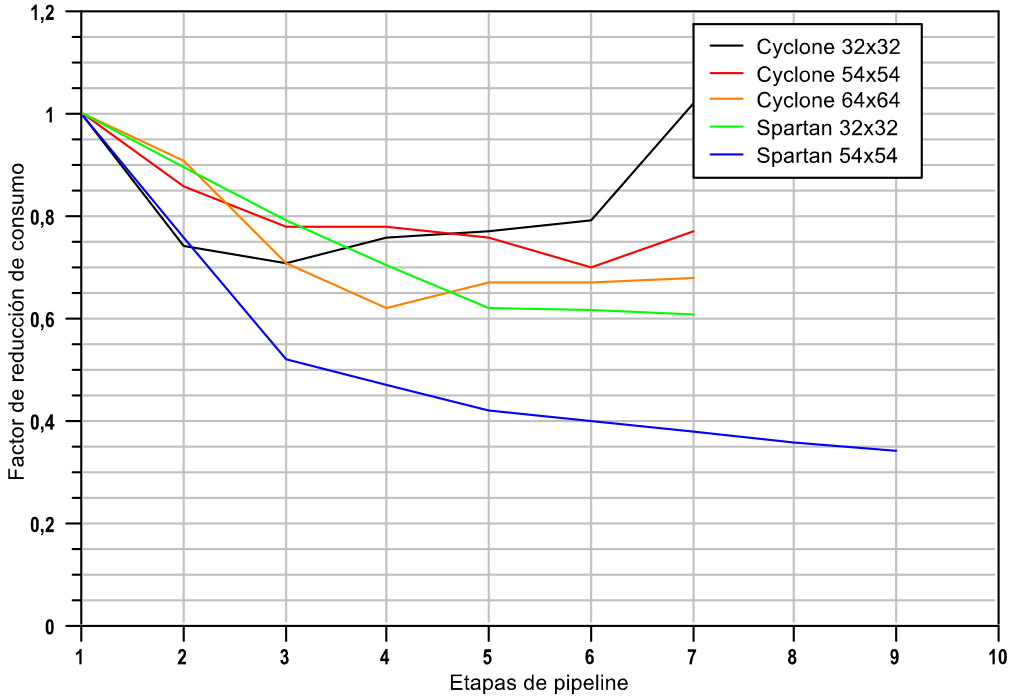


Figura 5-1 Reducción de consumo utilizando *pipelining*

La conclusión es que la técnica de *pipelining* es muy efectiva en FPGAs, y en aquellos casos en los cuales se puede aplicar es además totalmente "gratuita", debido a que el incremento de área dado por el número de *flip-flops* agregados ya se encuentra dentro del chip.

Hay que tener en cuenta también que esta técnica tiene un punto óptimo, ya que si se agregan demasiadas etapas de *pipeline* el consumo de los *flip-flops* adicionales supera el ahorro, y el consumo total vuelve a crecer como lo muestran algunas de las curvas de la Figura 5-1.

5.2 Utilización de bloques hardware

Una gran mayoría de las FPGA actuales incluyen bloques de hardware específicos. Uno de los más difundidos son las memorias RAM, seguidas de los bloques del tipo multiplicador-acumulador de enteros (también llamados *DSP blocks* o *Embedded Multipliers*).

Existen otros circuitos más específicos como controladores de memorias externas tipo DDR en varias versiones, o de entradas salidas rápidas como por ejemplo PCIe. La inclusión de microprocesadores hardware en FPGAs fue un tema controvertido. Algunos modelos los incluyeron en forma temprana por el año 2002, pero no tuvieron el éxito esperado. Por ejemplo, la familia Virtex-II Pro con el PowerPC, o la familia Altera Excalibur que incluía un procesador ARM. Luego pasaron algunos años donde la tendencia fue más bien hacia la utilización de los *softcores* como el Nios de Altera o el Xilinx MicroBlaze; y actualmente vuelven a aparecer los microprocesadores hardware. Hoy en día, tanto Altera como Xilinx, integran procesadores ARM en algunas líneas de chips: Cyclone V y Arria V para el caso de Altera; y los dispositivos Zynq de Xilinx. Altera tuvo también algunas versiones de FPGAs con el procesador Atom de Intel anunciadas en el año 2010.

Aunque el tema de los microprocesadores hardware es muy atractivo, para nuestro estudio de consumo nos vamos a limitar a los bloques de multiplicación-acumulación. Intentar hacer comparaciones entre procesadores *softcore* y hardware sería muy difícil, debido a que ambos poseen características muy diferentes. Para el caso de los multiplicadores-acumuladores en cambio la alternativa es clara, o se implementan en LUTs o se utilizan los bloques embebidos, y en cualquier caso la función final va a ser la misma. Veremos entonces la comparación de consumo para diferentes multiplicadores implementados en LUTs y en bloques hardware.

En la plataforma de Spartan-6 se analiza el consumo de multiplicadores enteros sin signo de ancho de palabra 32x32 y 54x54. Los resultados obtenidos se muestran en la Tabla 5-4 y en la Tabla 5-5 respectivamente, y son graficados en la Figura 5-2 y en la Figura 5-3, donde puede apreciarse claramente las reducciones obtenidas.

Tabla 5-4 - Multiplicadores 32x32 en Xilinx Spartan-6

	Multiplicador 32x32 66MHz		
	LUT (mA)	DSP (mA)	Ahorro
Combinatorio	43,2	8,5	80%
1 etapa *pipeline*	42,7	8,3	81%
2 etapa *pipeline*	34,1	6,6	81%
3 etapa *pipeline*	27,6	5,2	81%
4 etapa *pipeline*	26,6	4,3	84%
5 etapa *pipeline*	27,1	4,4	84%
6 etapa *pipeline*	26,4	4,7	82%

Tabla 5-5 - Consumo en multiplicadores 54x54 en Xilinx Spartan-6

	Multiplicador 54x54 16MHz		
	LUT (mA)	DSP (mA)	Ahorro
Combinatorio	67,2	15,0	78%
2 etapa *pipeline*	34,9	8,5	76%
4 etapa *pipeline*	27,9	5,5	80%
8 etapa *pipeline*	23,2	6,0	74%

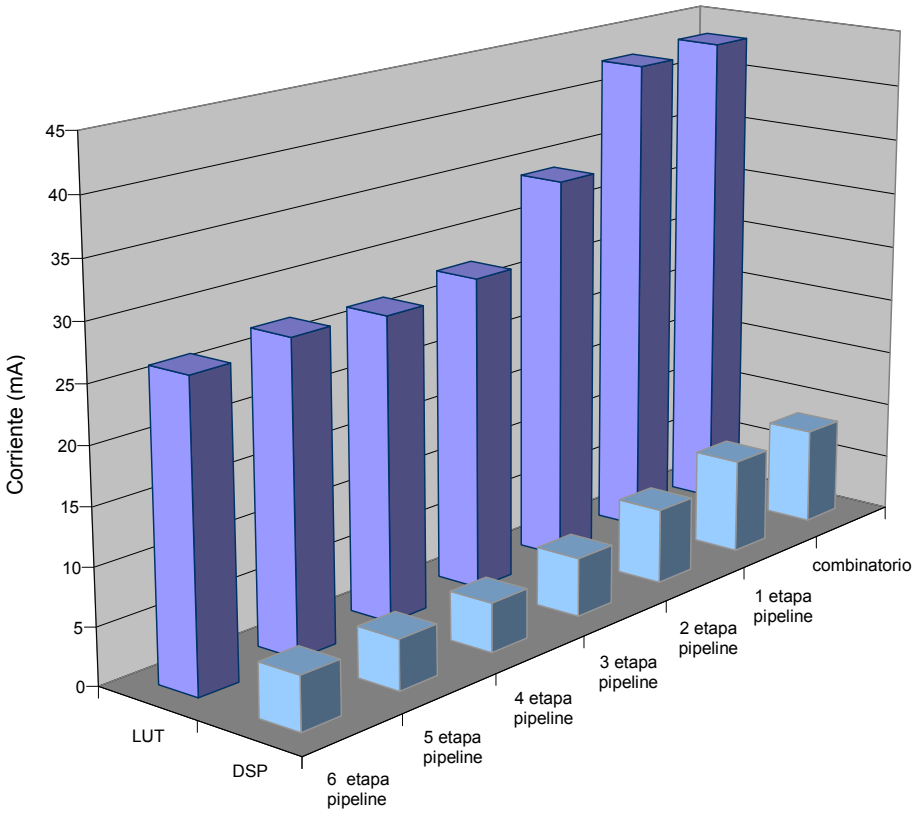


Figura 5-2 Comparación de consumo entre LUTs y bloques DSP 32x32 Spartan-6

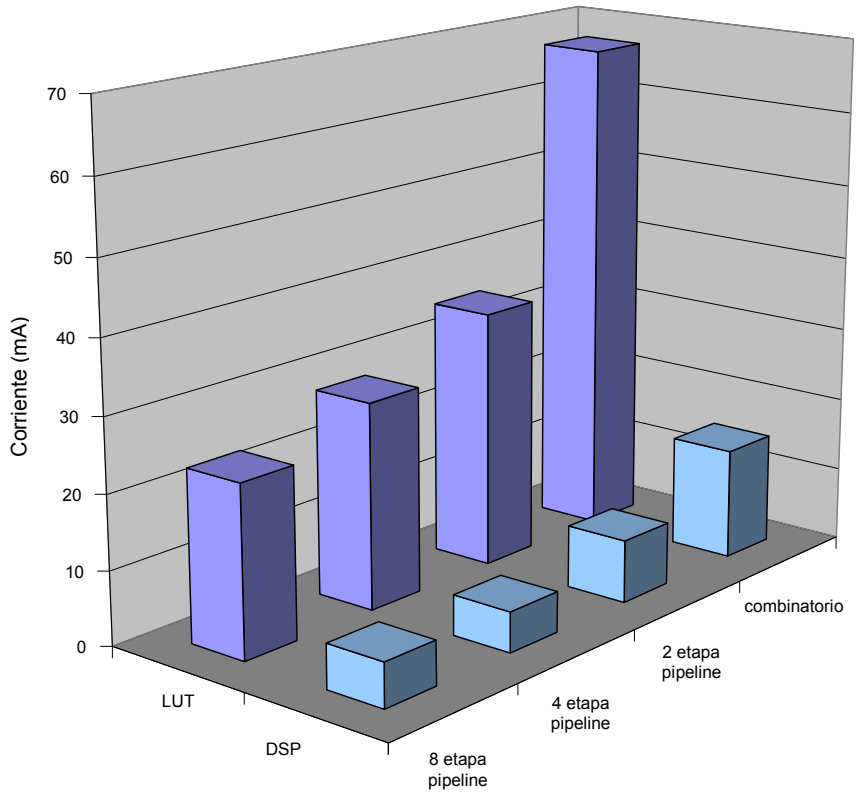


Figura 5-3 Comparación de consumo entre LUTs y bloques DSP 54x54 Spartan-6

En el caso de la Cyclone III se analizan multiplicadores sin signo de 3 tamaños: 32x32, 54x54 y 64x64. Igual que en el caso anterior los resultados se muestran en las siguientes tablas y gráficas.

Tabla 5-6 - Consumo en multiplicadores 32x32 en Altera Cyclone III

	Multiplicador 32x32 50MHZ		
	LUT (mA)	DSP (mA)	Ahorro
combinatorio	40,9	4,8	88%
1 etapa *pipeline*	27,2	5,4	80%
2 etapa *pipeline*	25,5	6,8	73%
3 etapa *pipeline*	28,1	9,8	65%

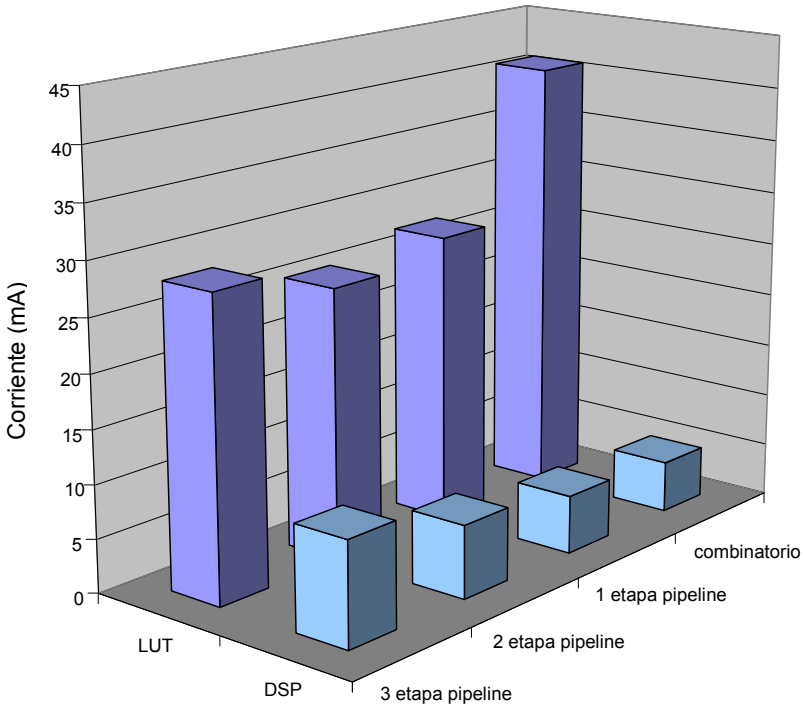


Figura 5-4 Comparación de consumo entre LUTs y bloques DSP 32x32 Cyclone III

Tabla 5-7 - Consumo en multiplicadores 54x54 en Altera Cyclone III

	Multiplicador 54x54 50MHz		
	LUT (mA)	DSP (mA)	Ahorro
combinatorio	107,4	15,3	86%
1 etapa *pipeline*	90,2	13,8	85%
2 etapa *pipeline*	80,8	17,7	78%
3 etapa *pipeline*	80,7	20,3	75%

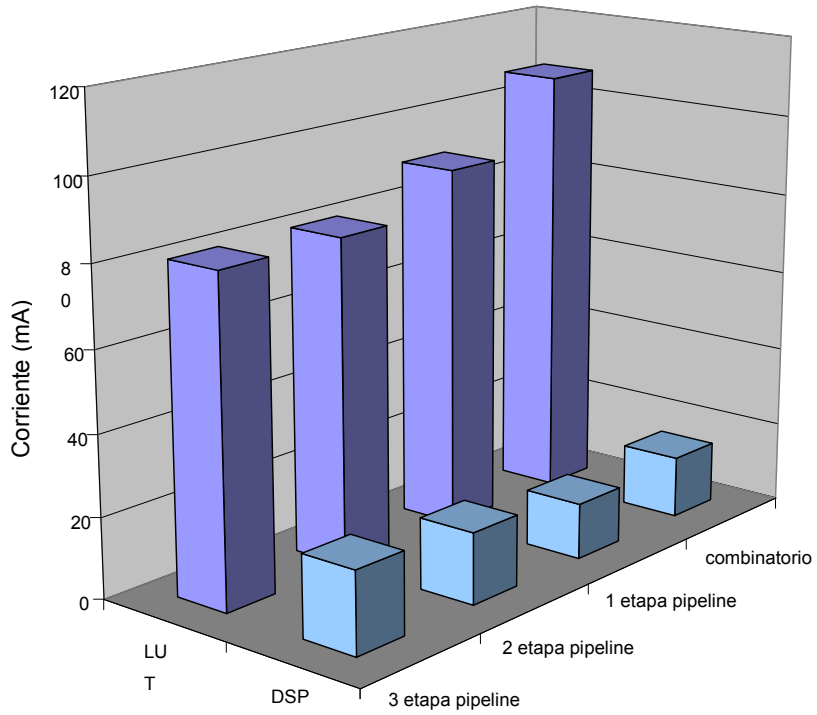


Figura 5-5 Comparación de consumo entre LUTs y bloques DSP 54x54 Cyclone III

Tabla 5-8 - Consumo en multiplicadores 64x64 en Altera Cyclone III

	Multiplicador 64x64 50MHz		
	LUT (mA)	DSP (mA)	Ahorro
combinatorio	152,1	27,2	82%
1 etapa *pipeline*	137,4	21,3	84%
2 etapa *pipeline*	103,8	25,3	76%
3 etapa *pipeline*	90,0	27,0	70%

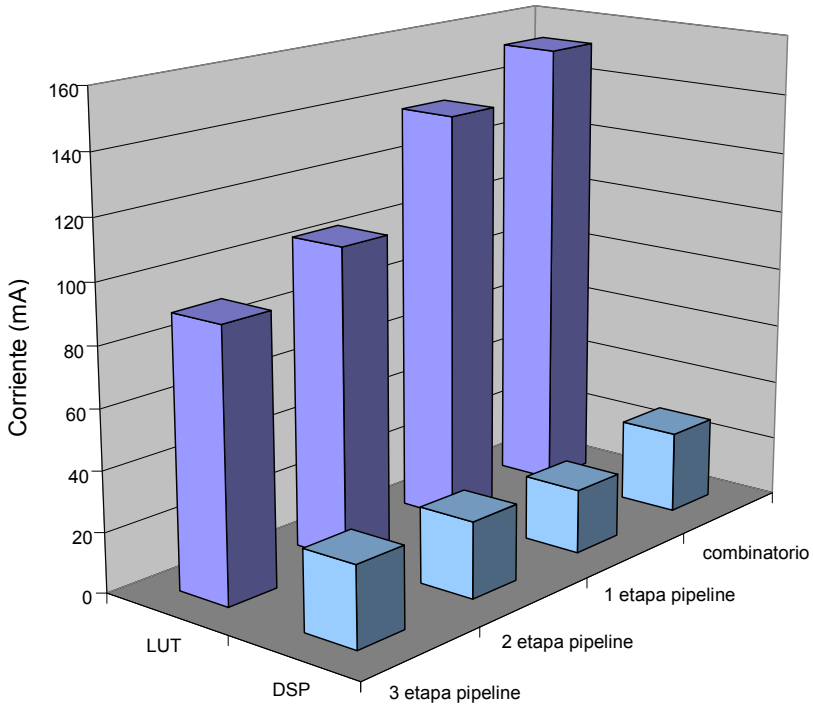


Figura 5-6 Comparación de consumo entre LUTs y bloques DSP 64x64 Cyclone III

Como comentarios finales puede decirse que la utilización de bloques de multiplicadores embebidos genera una gran reducción de consumo que puede ir desde un 65% a un 88%. En el caso de Xilinx Spartan-6 el rango de reducción es menos amplio yendo de 74% a 84%, y que prácticamente no depende de las etapas de *pipeline* de los multiplicadores. En cambio en el caso de Altera Cyclone III el rango es bastante más amplio 65% a 88%, obteniéndose mejores resultados relativos en circuitos puramente combinatorios o con pocas etapas de *pipeline*.

5.3 *Clock gating*

Una de las ideas más simples para reducir el consumo dinámico de un sistema es apagar temporalmente aquellos bloques o partes del sistema que no estén en uso. Esto puede ser hecho de varias maneras, con diferentes consecuencias: apagar totalmente la

fuente de alimentación de los bloques, deshabilitar la señal de reloj, o bloquear las entradas.

Estas técnicas son ampliamente utilizadas en ASICs donde se pueden incorporar a gusto del diseñador, pero en FPGAs los recursos disponibles están dados por el fabricante y son mucho más limitados.

Algunas FPGAs de Xilinx han incorporado modos de bajo consumo suspendido o en hibernación [123], [124], pero en general estos modos se aplican a un chip completo y no permiten su utilización parcial, de forma que no pueden ser utilizados para apagar bloques dentro de una FPGA.

Las FPGAs actuales también incorporan algunas celdas que permiten bloquear la señal de reloj en forma controlada hacia ciertas partes del circuito. En Xilinx estos bloques aparecieron por primera vez en las familias Virtex II y Spartan-3 y se mantuvieron en las versiones posteriores. En Altera pasó algo similar, se incorporaron en las familias Stratix II y Cyclone II y se mantuvieron en las versiones subsiguientes.

Considerando que una FPGA puede brindar una solución completa del tipo SoC o más bien SoPC, entonces las alternativas se reducen, y las preguntas que deberían ser contestadas son:

- ¿Cuáles son las alternativas para bajar el consumo en ciertas partes de la FPGA mientras el resto sigue funcionando?
- ¿Dentro de las alternativas disponibles cuál ofrece mayores ventajas?

En este capítulo se analizan y comparan tres técnicas diferentes, disponibles internamente en FPGAs, para poder bajar el consumo dinámico de ciertas partes de un circuito mientras el resto sigue operativo. Las técnicas que se estudian son: *clock gating*, *clock enable*, y bloqueo de entradas. Como en el resto de esta tesis las comparaciones están basadas en medidas experimentales de consumo.

El *clock gating* se define como el bloqueo de la señal de reloj a ciertas partes de un circuito. En los experimentos se utilizan para este fin solamente los elementos específicos provistos en las FPGAs. Esto se debe a que insertar compuertas comunes en las líneas de reloj no se recomienda ya que existe la posibilidad de corromper la señal de reloj con pulsos espurios o con un aumento del *skew*. Esto claramente limita la posibilidad de tener un control de grano fino, ya que los chips poseen muy pocos elementos de bloqueo de reloj. Pero, por otro lado este tipo de bloqueo de la señal de reloj parece a priori ser la mejor alternativa desde el punto de vista del consumo, porque elimina el consumo de la línea de reloj conjuntamente con el de los *flip-flops*.

La segunda técnica propuesta y estudiada es más simple, ya que consiste en utilizar *flip-flops* con entrada *clock enable* (CE), y deshabilitarla en aquellas partes del circuito que no se está utilizando. De esta forma el *flip-flop* no puede cambiar su valor. Esto no tiene restricciones de tiempo, y permite un control en grano fino del apagado de bloques; pero a priori no parece tan efectiva como la anterior debido que si bien reduciría el consumo en los *flip-flops* que están deshabilitados, la línea de reloj sigue consumiendo.

La tercer técnica estudiada se basa en deshabilitar el camino de datos de un determinado bloque o circuito bloqueando las entradas al mismo. Esto puede ser hecho de varias formas, en este caso se propone utilizar una barrera de *flip-flops*. Como en el caso anterior esto hace que tanto los *flip-flops* como la combinatoria mantengan sus valores constantes, pero la línea de reloj del sistema sigue activa.

Las medidas son realizadas en los bancos de pruebas descritos en 3.1, las placas que cuentan con un Altera Cyclone III y en la de Xilinx Spartan 6. Nuevamente se utilizan circuitos multiplicadores como ejemplos, y los generadores de entradas pseudo aleatorias se implementan con un LFSR.

Como antecedentes se puede mencionar que hay una gran cantidad de trabajos publicados que describen el rol del *clock gating* como una técnica de bajo consumo en

ASICs, la mayoría de estos estudios fue hecha en la década de 1990 [125], [126] y [127]. En los años siguientes la técnica fue incorporada en varias herramientas de diseño EDA que son capaces de generar señales de *clock gating* en varias partes del sistema en forma automática a partir de una descripción HDL [128], [129], [130], [131]. Recién en 2011 esta misma idea ha sido extendida a las herramientas para FPGAs de Xilinx [132].

Hay algunos trabajos que específicamente estudian el uso de la técnica del *clock gating* en FPGAs para reducir consumo. En [133] se realiza una partición de máquinas de estados apagando aquellas partes que no están activas. En [134] se aplica esta técnica a una implementación de un algoritmo Cordic en una FPGA de Xilinx. En [135] y [136] se presenta la utilización de *clock gating* y diferentes técnicas de bloquear señales para seleccionar entre diferentes caminos de datos. En [137] se analiza la utilización de *clock gating* para un conjunto de diseños comparando los resultados obtenidos con su implementación en FPGAs y en ASICs, pero no se realizan medidas, solamente simulaciones. Un interesante trabajo [138] propone nuevas arquitecturas de FPGAs que incluyen una red de reloj mejorada para incorporar mayor flexibilidad en el manejo del reloj y su bloqueo. Por último se puede mencionar a [139] en donde se propone una técnica de reconfiguración dinámica del enrutamiento del reloj.

El trabajo aquí presentado aporta medidas realizadas en dos tipos de dispositivos: Cyclone III de 65 nm, y en Spartan 6 de 45 nm, comparando modos activos con modos *standby* utilizando las tres técnicas mencionadas anteriormente.

5.3.1 Trabajo experimental

A los circuitos de prueba propuestos en 3.2 se le deben hacer algunas modificaciones para poder realizar el *clock gating* y el *clock enable*. El primer caso se puede ver en la Figura 5-7 donde se intercala una compuerta de habilitación en la línea de reloj.

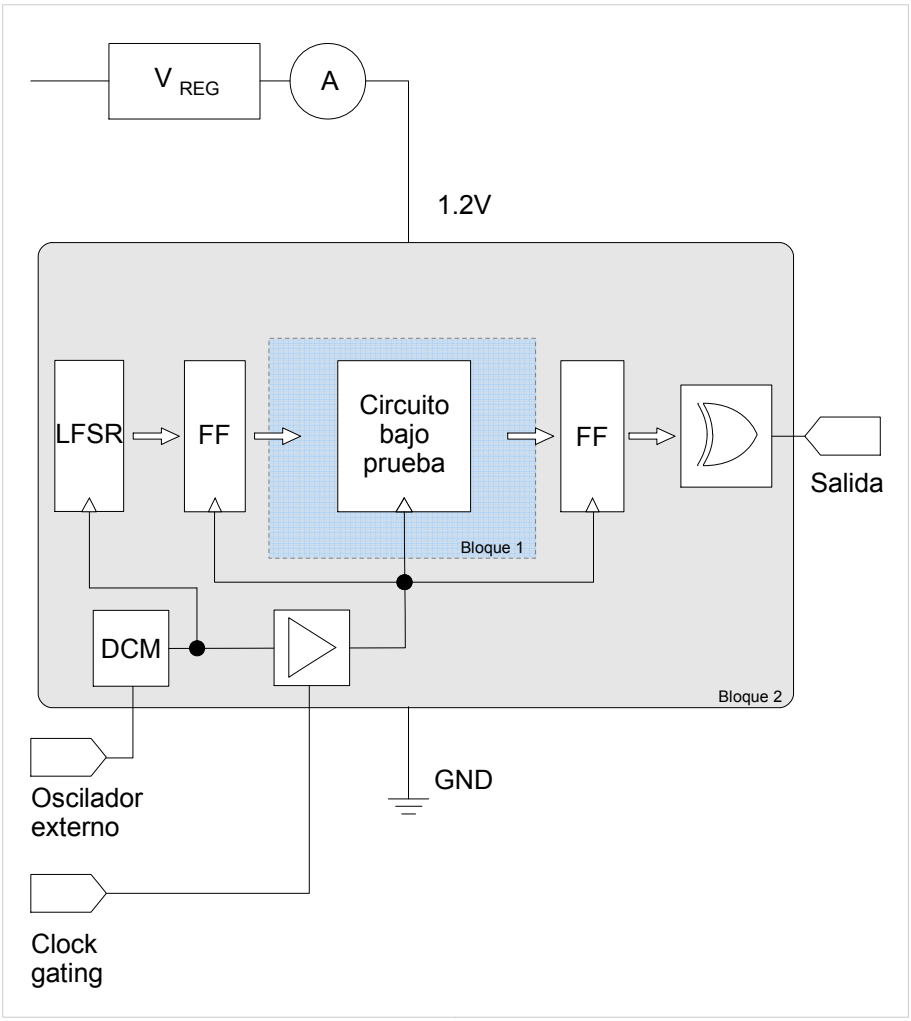


Figura 5-7 Circuito con entrada de *clock gating*

Como ya fue explicado se utilizan LFSR para generar entradas pseudo aleatorias, y al final se utiliza una compuerta EXOR para bajar la cantidad de salidas y evitar que la herramienta minimice el circuito [53], [122]. Se utiliza también un bloque DCM (*Digital Clock Manager*) para el control de la frecuencia del reloj.

Los bloques LFSR y DCM son comunes a todos los casos, mientras que los circuitos de prueba se cambian para generar diferentes tipos de experimentos. En unos casos se utiliza un multiplicador solo, y se cambia la cantidad de etapas de *pipeline*, mientras que en otros se varía la cantidad de multiplicadores utilizados. En total se analizan 6 casos que pueden verse resumidos en la Tabla 5-9.

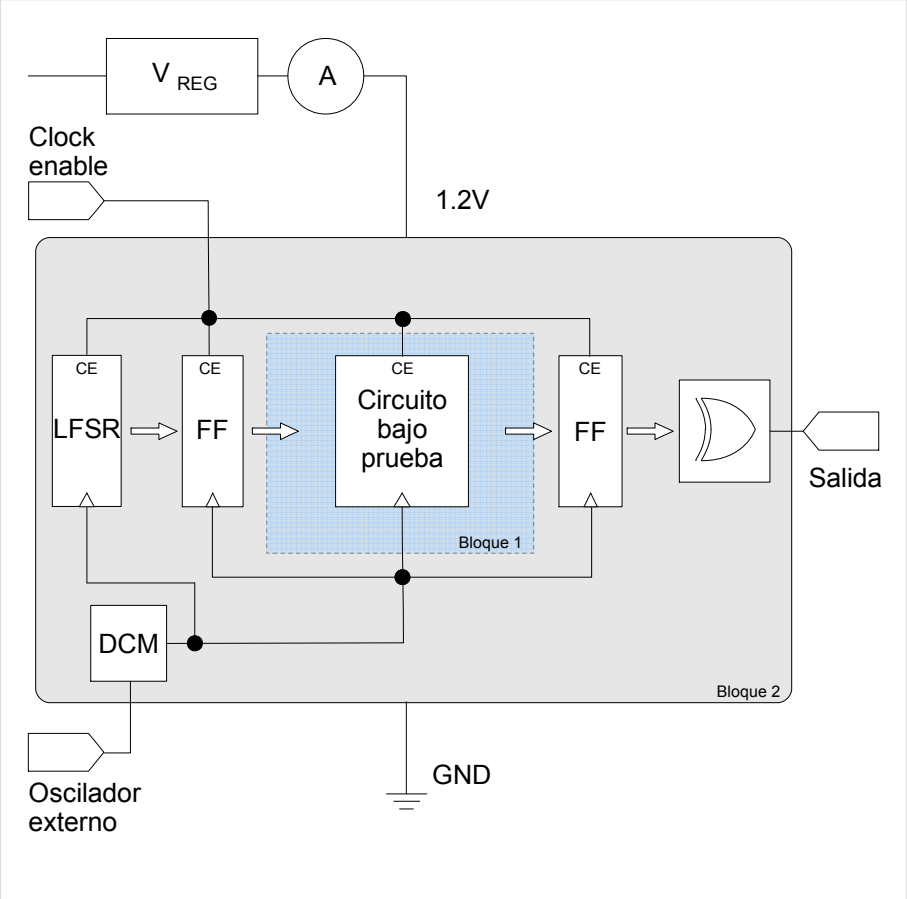


Figura 5-8 Circuito con entrada de *clock enable*

Se utilizan las placas DE0 de Altera, y Avnet Spartan-6 LX16 Evaluation Kit de Xilinx, si bien los osciladores externos de ambas placas difieren se utilizan los bloques manejadores de reloj para generar una frecuencia interna fija de 50MHz en todos los casos.

Tabla 5-9 - Descripción de circuitos utilizados

Diseño	Descripción	
	Cantidad de multiplicadores 32x32	Etapas de *pipeline* por multiplicador
M_1	1	0
M_2	1	1
M_3	1	3
M_4	1	5
M_5	2	5
M_6	3	5

En las tres técnicas mencionadas se compara el consumo con el circuito activo y con el circuito inactivo o en *standby*, la inactividad del circuito se logra en cada caso aplicando cada una de las técnicas propuestas. La implementación de cada técnica implica realizar variaciones en los circuitos.

La primer técnica utilizada es la de *clock gating* (Figura 5-7). Para implementar el *clock gating* dentro de la FPGA sin tener problemas de tiempos o de *glitches* en la línea de reloj se deben utilizar los bloques provistos por los fabricantes. En el caso de Altera es posible hacer esto con la *megafunction* ALTCLKCTRL [140]. En el caso de Xilinx debe utilizarse la primitiva BUFGCE [141], esta primitiva se construye a partir de una más compleja desarrollada para permitir el multiplexado de relojes sin tener *glitches*. Ambos bloques cuentan con una entrada de control que permite dejar pasar o bloquear el reloj del circuito bajo prueba. Esta señal de control (CG) la conectamos a un pin del FPGA cableado a un interruptor para cambiar el modo de funcionamiento del circuito en forma manual.

Para la segunda técnica se deben utilizar *flip-flops* con entrada de habilitación (Figura 5-8), esta entrada está físicamente presente en todos los *flip-flops* de la FPGA, pero debe ser definida en el diseño para poder utilizarla. Se utiliza una línea global para conectar todas las entradas de habilitación de los *flip-flops* hasta un pin de entrada del chip que llamaremos CE. Esta entrada CE será el control de modo activo o inactivo en este caso.

La tercer técnica consiste en bloquear las entradas del camino de datos al circuito bajo prueba, esta modalidad se realiza como una variante de la anterior, conectando la señal CE solamente en los *flip-flops* de entrada pero no en el resto del sistema. En este caso podrían presentarse variaciones debido a que el último dato que queda memorizado en la barrera de *flip-flops* es la entrada que queda fija en el circuito bajo prueba, y esto podría dar un consumo dependiente de ese valor. Para evitar esto se utilizan una gran cantidad de valores aleatorios y se promedia el valor final de los consumos medidos.

Para cada circuito se mide en estado activo e inactivo aplicando las 3 técnicas descritas. En estado activo las variaciones son mínimas, y se deben solamente a las modificaciones que se hacen en los circuitos para implementar cada una de las técnicas.

El consumo de los circuitos auxiliares LFSR, DCM, y XOR se mide en forma separada y se resta al consumo total, los datos presentados en la Tabla 5-10 y Tabla 5-11 muestran solamente el consumo de los circuitos bajo test.

5.3.2 Consumo de los circuitos en modo inactivo

La primer técnica de *clock gating* implica el apagado total de la línea de reloj, por lo tanto los *flip-flops* del circuito no pueden cambiar su valor, debido a esto la lógica combinatoria hecha en LUTs tampoco cambiará su estado lógico. El consumo entonces es prácticamente el consumo estático del chip, y no depende mucho del tamaño del circuito ni de la cantidad de *flip-flops*. Como técnica de apagado, y desde el punto de vista del consumo, es la mejor.

Los resultados obtenidos con la técnica de *clock enable* muestran que el consumo se reduce en forma significativa con respecto al modo activo, pero es mayor que el caso anterior. Este es el resultado esperado, ya que si bien los *flip-flops* no cambian de estado lógico, la línea de reloj sigue activa y consumiendo.

Es interesante observar que el consumo inactivo cuando se aplica esta técnica de *clock enable* crece casi linealmente con la cantidad de *flip-flops* de circuito bajo test, como lo muestran la Figura 5-9 y la Figura 5-10

Teóricamente, en modo inactivo, el uso de la técnica de entradas bloqueadas debería ser equivalente a la de *clock enable*, dado que en ambas queda funcionando la línea de reloj, pero los *flip-flops* no cambian de estado, y por lo tanto la combinatoria tampoco. En la práctica esto se cumple para los dispositivos Xilinx, como puede observarse en la Figura 5-9; sin embargo para el caso del chip de Altera, el consumo inactivo con entradas bloqueadas es mayor que el de *clock enable* (Figura 5-10).

Tabla 5-10 - Consumos medidos en Spartan 6

Diseño	Cantidad de FF	Consumo en activo (mW) @50MHz			Consumo en *standby* (mW)		
		Clock Gating	*Clock Enable*	Entradas Bloqueadas	*Clock Gating*	*Clock Enable*	Entradas Bloqueadas
M_1	192	46.0	44.3	50.2	0.2	1.1	1.5
M_2	256	43.5	45.2	47.3	0.1	1.5	1.3
M_3	947	28.2	27.1	26.4	0.1	3.6	3.5
M_4	1285	27.3	26.4	26.3	0.1	4.6	4.4
M_5	2442	54.0	55.1	40.2	0.2	8.7	8.9
M_6	3657	95.8	121.8	110.8	0.3	11.6	12.1

Tabla 5-11 - Consumos medidos en Cyclone III

Diseño	Cantidad de FF	Consumo en activo (mW) @50MHz			Consumo en *standby* (mW)		
		Clock Gating	*Clock Enable*	Entradas Bloqueadas	*Clock Gating*	*Clock Enable*	Entradas Bloqueadas
M_1	192	43.1	40.3	39.5	0.1	1.2	1.4
M_2	614	32.1	30.2	30.0	0.1	1.4	3.3
M_3	1086	29.0	28.7	27.2	0.1	2.1	4.1
M_4	1525	34.4	30.9	32.0	0.2	2.3	9.0
M_5	2915	64.4	57.7	62.2	0.3	3.6	15.5
M_6	4271	94.8	86.9	90.9	0.5	5.4	22.5

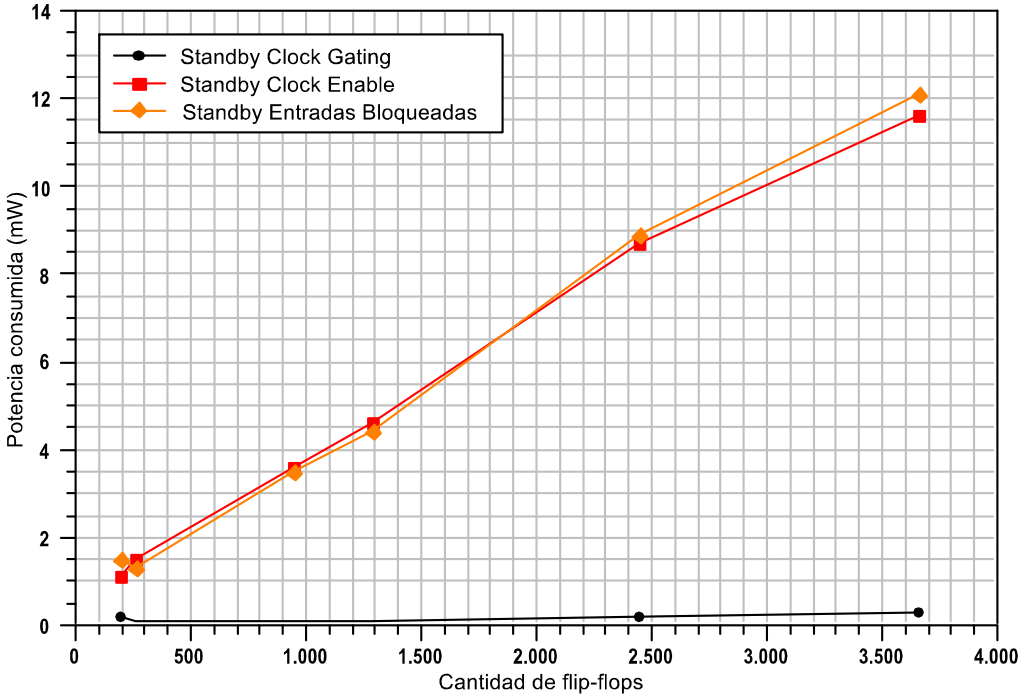


Figura 5-9 Consumo inactivo vs. cantidad de FF en Spartan 6

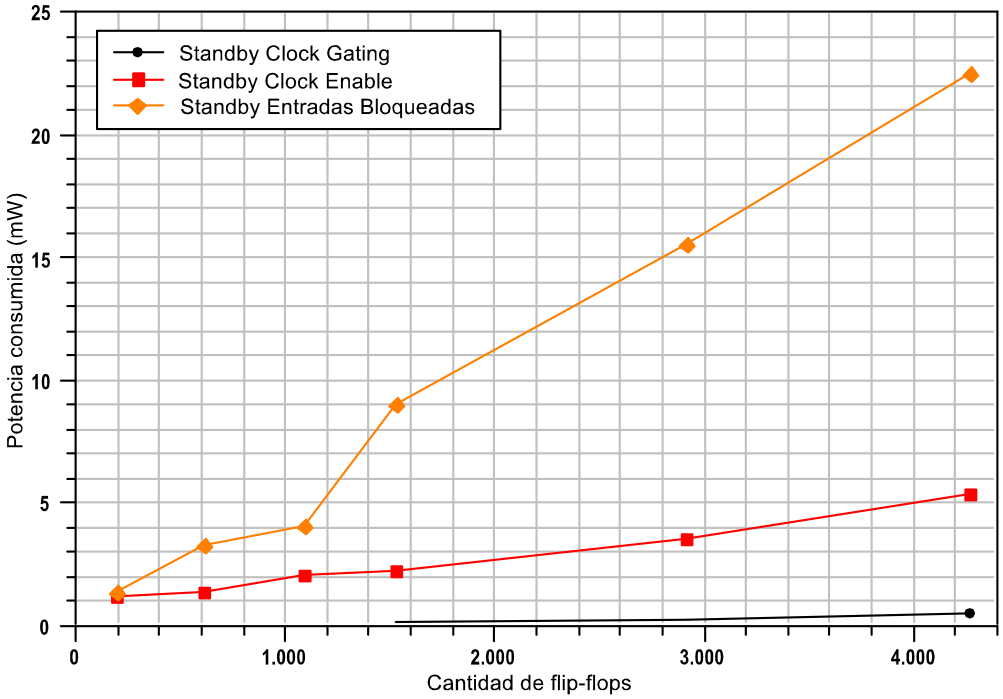


Figura 5-10 Consumo inactivo vs. cantidad de FF en Cyclone III

En la Figura 5-10 puede verse que la diferencia entre aplicar *clock_enable* y entradas bloqueadas en el caso del Cyclone es considerable, y que ambas son bastante lineales con la cantidad de *flip-flops* utilizados. Una explicación para esta diferencia, que como ya se mencionó no se presenta en la Spartan, puede encontrarse en la arquitectura interna del Cyclone. Las celdas básicas, que en la nomenclatura de Altera se llaman *Logic Elements* (LE), se encuentran agrupadas en *Logic Array Blocks* (LABs). Cada LAB contiene 16 LE, y el LAB tiene señales globales de control que se apican a todos los LEs. Entre las señales globales de control disponibles, existen 2 especialmente dedicadas a habilitación de reloj llamadas *labclkena1* y *labclkena2*. La utilización de estas señales globales está automatizada por Quartus cada vez que se utiliza un *clock enable* que afecta a los 16 *flop-flops* de un LAB, tal como explica la documentación del Cyclone:

"Deasserting the clock enable signal turns off the LAB-wide clock." [142]

Este efecto global de las líneas de *clock enable* para todo un LAB, explicaría la baja de consumo al utilizar esta técnica.

5.3.3 Consumo de los circuitos en modo activo

Debido a que la implementación de cualquiera de las 3 técnicas implica incluir modificaciones en el circuito original, es que se decide estudiar si estas modificaciones afectan el consumo en modo activo.

El principal resultado es que en modo activo la aplicación de las 3 técnicas no introducen grandes cambios en el consumo de los circuitos bajo test, aunque pueden apreciarse algunas variaciones. Esto es bueno, ya que nos permite elegir la técnica pensando en el modo inactivo, y sabiendo que cuando el sistema esté en funcionamiento su consumo no variará mucho con la técnica elegida.

Puede verse que el caso de la Spartan 6 (Figura 5-11) los consumos en activo se empiezan a diferenciar en los diseños grandes M_5 y M6; en M_5 la opción que consume menos en activo es con las entradas bloqueadas, y en el caso M_6 la que consume menos es *clock gating*; pero las curvas se cruzan.

En Cyclone III los resultados son un poco diferentes (Figura 5-12), en este caso las curvas no se cruzan y se mantiene que *clock gating* es la opción que consume más en todos los circuitos y las que consumen menos se reparten entre *clock enable* para los 3 diseños más grandes y entradas bloqueadas para los 3 diseños más chicos.

5.3.4 Conclusiones

En este apartado se analizan 3 técnicas que permiten el "apagado" dinámico de partes de un circuito que no se utilizan en determinados instantes de tiempo, dentro de las posibilidades limitadas que ofrecen las FPGAs. Las alternativas utilizadas son 3: *clock gating*, *clock enable* y bloquear entradas con barreras de *flip-flops*.

La primer conclusión es que aplicando cualquiera de las técnicas se obtiene una reducción muy significativa del consumo frente a la alternativa de dejar el circuito siempre en modo activo.

La técnica que produce mejores resultados en consumo inactivo es la de *clock gating*, utilizando bloques provistos por el fabricante y anulando líneas globales de reloj. Un resultado adicional es que la aplicación de esta técnica es prácticamente independiente del tamaño del diseño. El uso de múltiples líneas de reloj globales no siempre es posible o fácil, y puede traer aparejados diversos problemas de diseño, pero en el caso de utilizar una FPGA para resolver un sistema completo tipo SoPC, en el cual se utilizan bloques de gran tamaño que pueden desactivarse cuando no están en uso, claramente el *clock gating* es la mejor opción.

La técnica de *clock enable* es una muy buena opción cuando el diseño no tiene una gran cantidad de *flip-flops*, ya que como se mostró el consumo del circuito inactivo es lineal con la cantidad de *flip-flops*. Pero por otro lado la aplicación de esta técnica es muy simple, no introduce multiplicidad de relojes ni problemas de tiempos, y permite su utilización en grano fino.

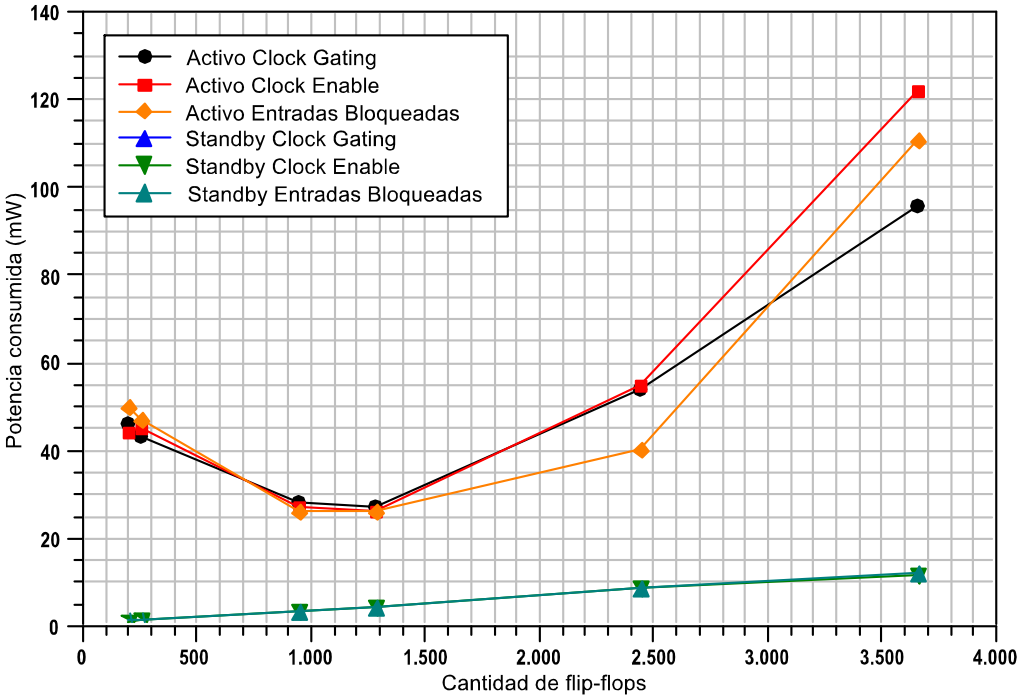


Figura 5-11 Consumo activo e inactivo vs. cantidad de FF en Spartan 6

Otro aspecto a destacar es que la aplicación de cualquiera de estas técnicas modifica el consumo en modo activo de los circuitos, por lo tanto deberá tenerse en cuenta el ciclo de trabajo de activo-inactivo de cada bloque a apagar para calcular el ahorro total de energía.

Cabe hacer notar que, como se menciona en el apartado 5.1, el *pipelining* es una de las técnicas utilizadas para reducir consumo en modo activo, y esto puede verse en las Figura 5-11 y Figura 5-12; pero incrementa enormemente la cantidad de *flip-flops*. Por lo tanto incrementará también el consumo en modo inactivo, sobre todo si se utiliza la técnica de *clock enable*. Nuevamente el balance final hay que hacerlo en energía teniendo en cuenta los ciclos de trabajo de los bloques.

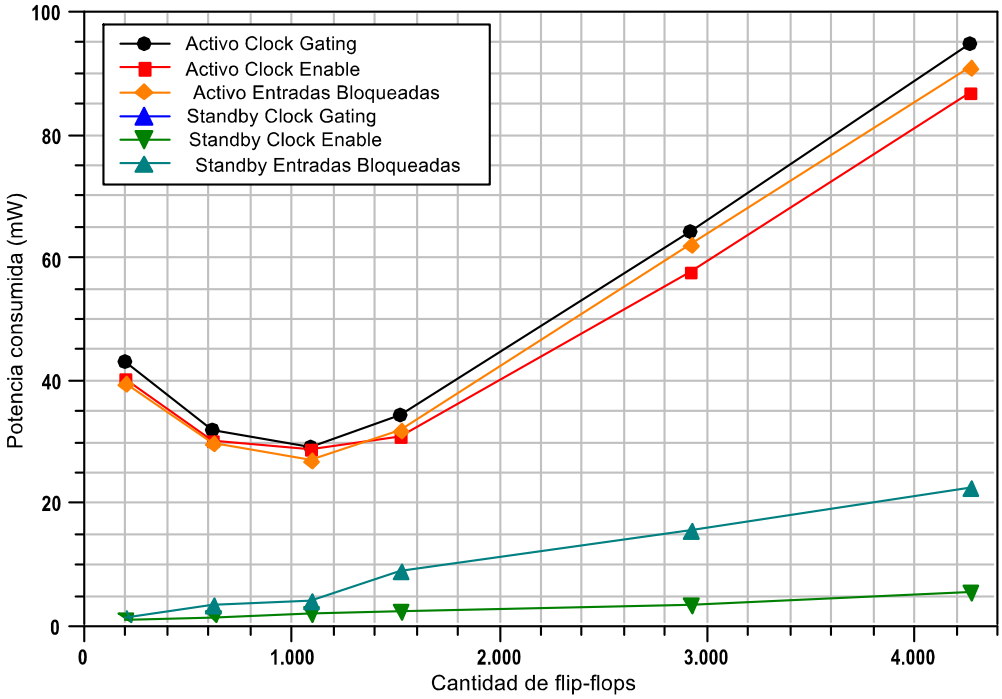


Figura 5-12 Consumo activo e inactivo vs. cantidad de FF en Cyclone III

Por último, puede verse una diferencia entre Spartan 6 y Cyclone III si comparamos *clock enable* con bloquear las entradas en los circuitos inactivos. Mientras que en el caso de Spartan 6 da prácticamente lo mismo aplicar cualquiera de estas técnicas, para la Cyclone III *clock enable* siempre da mejores resultados.

5.4 La opción *Suspend* de Spartan

Algo similar al apartado anterior, pero a nivel de toda la FPGA completa, puede obtenerse utilizando la opción *Suspend* que presenta Xilinx para los integrados de la familia Spartan. Esta opción permite el "apagado" del chip conservando su configuración. En particular para el caso de la Spartan 6, se proporcionan herramientas bastante sofisticadas para el manejo del modo *Suspend*, que incluyen opciones tales como filtrado de *glitches* o un circuito de sincronización para garantizar el estado de los *flip-flops*. Un listado completo de las opciones del modo *Suspend* puede verse en la guía de usuario [124].

La crítica que, en una primera instancia, se le puede hacer a este modo *Suspend* es que se necesita una señal externa para despertar al chip, es decir que no permite que la FPGA deje una parte activa y se despierte a si misma. Esto impide usarlo en soluciones tipo SoPC, y obliga a poner circuitería externa.

Pero más allá de esta observación, los resultados experimentales obtenidos muestran que el modo *Suspend* no es una buena opción desde el punto de vista del consumo, ya que casi no reduce el consumo estático.

Se prueba el modo para algunos casos y se obtienen los siguientes resultados:

Tabla 5-12 - Consumos medidos en modo *Suspend*

Circuito	Consumo en *Suspend* (mA)	Consumo sin reloj (mA)
AES	7.6	7.2
FFT	7.7	6.8

Por lo tanto desde el punto de vista del consumo interno de la FPGA la opción *Suspend* es mas o menos equivalente a bloquear el reloj del circuito, incluso en uno de los casos es peor . Esto coincide con lo publicado en una nota de aplicación de Xilinx para la Spartan 3A [143], que da los siguientes resultados:

Tabla 5-13 - Consumos en modo *Suspend* medidos por Xilinx

Configured Device but NO Clock	12.1mA
Suspend Mode	13.5mA

5.5 Impacto en el consumo de las opciones de las herramientas

En esta sección se analiza cómo afectan al consumo las diferentes opciones de las herramientas de diseño. Para ello se utilizan los circuitos de *benchmark* propuestos en

el Capítulo 4. Para cada opción estudiada se calcula también la estimación de consumo y su error.

Se comienza con las opciones de Xilinx, se utiliza la versión ISE Designa Suite 13.1, que presenta las siguientes metas de diseño:

- *Balanced*
- *Power Optimization*
- *Timing Performance*
- *Area Reduction*
- *Minimum Runtime* - no utilizada, no tiene sentido para nuestro objetivo de estudiar consumo minimizar el tiempo de compilación

Los resultados obtenidos de aplicar cada una de las opciones a los circuitos de *benchmark* pueden verse en la Tabla 5-14.

Tabla 5-14 - Consumos medidos y estimados variando opciones del ISE

	ISE Design Goal	Consumo medido (mW)	Estim. VHDL (mW)	Estim. Verilog (mW)	Error estim. VHDL	Error estim. Verilog
mult 32x32 LUT	*Balanced*	70,7	168,07	68,52	138%	-3%
	Power Optimization	69,7	164,24	68,29	136%	-2%
	Area Reduction	69,7	164,24	68,29	136%	-2%
	Timing Performance	69,7	164,24	68,29	136%	-2%
AES	*Balanced*	90,7	206,4	86,1	128%	-5%
	Power Optimization	66,7	148,2	65,4	122%	-2%
	Area Reduction	69,9	151,8	68,4	117%	-2%
	Timing Performance	107,6	237,1	104,3	120%	-3%
FFT	*Balanced*	218,2	277,3	235,4	27%	8%
	Power Optimization	219,0	275,9	236,5	26%	8%
	Area Reduction	212,5	266,6	228,6	25%	8%
	Timing Performance	245,8	294,0	267,9	20%	9%
openMSP430	*Balanced*	21,9	29,5	26,7	35%	22%
	Power Optimization	18,7	23,0	23,5	23%	26%
	Area Reduction	19,0	26,3	24,0	38%	26%
	Timing Performance	21,5	26,1	26,1	22%	21%
IEEE802.15.4	*Balanced*	14,3	68,0	23,2	377%	63%
	Power Optimization	15,5	69,6	23,3	350%	50%
	Area Reduction	15,4	75,8	23,5	394%	53%
	Timing Performance	14,8	46,3	22,9	214%	55%

Un primer resultado interesante surge de analizar los errores de estimación, como ya se mostró en la sección 3.6, la herramienta de Xilinx estima mejor el consumo cuando el *netlist* de la simulación se genera en Verilog y no en VHDL. Esto, que se había verificado solamente para circuitos multiplicadores, se confirma para todos los circuitos estudiados. Es interesante observar además que el error de estimación tiene una fuerte dependencia con el circuito, y no con las opciones de síntesis.

Con respecto al ahorro de consumo, se genera una tabla reducida que incluye solamente los casos máximos y mínimos:

Tabla 5-15 - Consumos máximos y mínimos con opciones del ISE

	Opción que da el consumo máximo	mW	Opción que da el consumo mínimo	mW	Ahorro
mult 32x32 LUT	*Balanced*	70,7	*Power Optimization*	69,7	1%
AES	*Timing Performance*	107,6	*Power Optimization*	66,7	38%
FFT	*Timing Performance*	245,8	*Area Reduction*	212,5	14%
openMSP430	*Balanced*	21,9	*Power Optimization*	18,7	15%
IEEE 802.15.4	*Power Optimization*	15,5	*Balanced*	14,3	8%

Como primer observación se puede notar que nuevamente las metas de compilación no son coherentes, en 3 casos los mínimos de consumo se logran con la meta *Power Optimization*, pero en los otros dos no.

La segunda observación interesante es que en la mayoría de los casos se pueden obtener reducciones de consumo nada despreciables solamente cambiando alguna estrategia o meta de compilación. Lo malo es que no se puede decir a priori cual va a ser, ni tampoco confiar en las estimaciones. Por ejemplo en el caso del IEEE 802.15.4 no se cumple que los máximos y mínimos de consumo medidos coincidan con los máximos y mínimos estimados (aunque en los otros circuitos sí parecen coincidir).

En el caso de Altera se utiliza el Quartus II 9.0, donde las opciones de optimización de síntesis y consumo se pueden manejar en forma separada. Por un lado en la síntesis hay 3 opciones:

- *Speed*
- *Balanced*
- *Area*

Por otro lado con en forma independiente se puede controlar la optimización en cuanto al consumo que tiene 3 opciones posibles:

- *Off* - no realiza ninguna optimización de consumo
- *Normal Compilation* - realiza optimización de consumo sin perder performance
- *Extra effort* - realiza optimizaciones de consumo que pueden llevar a la pérdida de performance

Tabla 5-16 - Consumos medidos y estimados variando opciones del Quartus

Diseño	Opción de síntesis	Opciones de optimización de consumo	Medidas (mW)	Estim. (mW)	Error (%)
mult 32x32 LUT	*Balanced*	*OFF*	65,8	50,9	-23%
	Balanced	*Normal Compilation*	65,6	50,9	-22%
	Balanced	*Extra Effort*	65,6	50,9	-22%
	Speed	*OFF*	61,8	48,5	-22%
	Speed	*Normal Compilation*	61,8	48,5	-22%
	Speed	*Extra Effort*	61,8	48,5	-22%
	Area	*Normal Compilation*	65,6	52,0	-21%
AES	*Balanced*	*OFF*	144,4	134,4	-7%
	Balanced	*Normal Compilation*	144,6	134,4	-7%
	Balanced	*Extra Effort*	144,6	134,4	-7%
	Speed	*OFF*	141,6	142,6	1%
	Speed	*Normal Compilation*	141,6	142,6	1%
	Speed	*Extra Effort*	152,6	156,9	3%
	Area	*Normal Compilation*	153,4	135,0	-12%
FFT	*Balanced*	*OFF*	216,2	231,6	7%
	Balanced	*Normal Compilation*	216,2	231,7	7%
	Balanced	*Extra Effort*	221,0	237,3	7%
	Speed	*OFF*	220,0	237,0	8%
	Speed	*Normal Compilation*	221,2	237,0	7%
	Speed	*Extra Effort*	222,6	240,0	8%
	Area	*Normal Compilation*	223,2	239,4	7%
openMSP430	*Balanced*	*OFF*	30,0	34,6	15%
	Balanced	*Normal Compilation*	30,0	34,6	16%
	Balanced	*Extra Effort*	29,2	33,5	15%
	Speed	*OFF*	33,4	37,7	13%
	Speed	*Normal Compilation*	33,4	37,7	13%
	Speed	*Extra Effort*	33,2	38,2	15%
	Area	*Normal Compilation*	29,9	34,1	14%
IEEE802.15.4	*Balanced*	*OFF*	22,2	18,9	-15%
	Balanced	*Normal Compilation*	22,2	18,8	-15%
	Balanced	*Extra Effort*	22,0	18,8	-15%
	Speed	*OFF*	20,6	18,0	-13%
	Speed	*Normal Compilation*	21,4	18,0	-16%
	Speed	*Extra Effort*	22,0	18,5	-16%
	Area	*Normal Compilation*	21,6	18,1	-16%

Nuevamente se pude ver que el error depende del tipo de circuito, y que para un determinado diseño se mantiene bastante estable (aunque hay una excepción en el caso del AES).

También se da que los máximos y mínimos de los consumos no siempre se obtienen para las mismas opciones, nuevamente se genera una tabla con un resumen de los máximos y mínimos obtenidos: Tabla 5-17. Lo curioso es que la opción "*Extra effort*" no da el menor consumo en ninguno de los circuitos estudiados.

Tabla 5-17 - Consumos máximos y mínimos con opciones del Quartus

	Opción que da el consumo máximo	**mW**	**Opción que da el consumo mínimo**	**mW**	**Ahorro**
mult 32x32 LUT	*Balanced/OFF*	65,8	*Speed/todas*	61,8	6%
AES	*Area/Normal comp.*	153,4	*Balanced/OFF*	144,4	6%
FFT	*Timing Performance*	223,2	*Balanced/OFF-Nc*	216,2	3%
openMSP430	*Speed/OFF-Nc*	33,4	*Area/Normal comp*	29,9	10%
IEEE 802.15.4	*Balanced/OFF-Nc*	22,2	*Speed/OFF*	20,6	7%

Las variación de las reducciones de consumo obtenidas en Altera, comprendidas entre un 3% y un 10%, son mucho menores que para Xilinx que van entre 1% y 38%. En realidad lo que sucede es que en Xilinx se obtienen dispersiones más grandes entre máximos y mínimos cuando se varían las opciones de síntesis.

5.6 Un caso de estudio: reducción del consumo del openMSP430

En esta sección se presentan los resultados de aplicar técnicas de bajo consumo a un sistema completo funcionando en una FPGA. Parte de las medidas y trabajos de este capítulo fueron realizados por el proyecto de fin de carrera [144] dirigido por el autor de esta tesis.

Del conjunto de circuitos de *benchmark* propuesto se elige el openMSP430 por ser uno de los diseños más versátiles para aplicar diferentes técnicas y estrategias de reducción de consumo.

Además la versión COTS del MSP430 de Texas Instruments es ampliamente utilizada en aplicaciones de bajo consumo, y varias de las plataformas de redes de sensores están basadas en este microcontrolador, por lo cual parece interesante caracterizarlo desde el punto de vista del consumo en su versión *softcore* dentro de una FPGA.

Está claro que la versión COTS consume mucho menos que la versión en FPGA, pero como primer dato para fijar ideas se muestra en la Tabla 5-18 el consumo aproximado de ambas implementaciones.

Tabla 5-18 - Comparación de consumos de un MSP430 y un openMSP430 en una Spartan 6

	MSP430 Texas Instruments @1.8V	MSP430 Texas Instruments @3V	openMSP430 Spartan 6 @1.2V
Standby	0.001mW	0.002mW	6-8mW
Activo a 16MHz	5-8mW	9-13mW	13-17mW

Las dos primeras columnas representan un consumo típico de un MSP430 trabajando a 16MHz con distintas fuentes de alimentación, y los valores están sacados de las hojas de datos de Texas Instruments. En la tercer columna se muestran valores obtenidos de mediciones realizadas sobre un openMSP430 en una Spartan 6, también a 16MHz. Como puede apreciarse la gran diferencia se da para el consumo en *standby*, siendo de varios órdenes de magnitud. Esto es debido a que la FPGA no cuenta con modos *standby* de ultra bajo consumo ya que debe mantener los datos de la RAM de programación y presenta fugas considerables. Por otra parte el puede verse que si bien el consumo en modo activo de la FPGA también es mayor, la diferencia no es tan grande.

5.6.1 Descripción del sistema

El sistema sobre el cual se aplican las técnicas de reducción de consumo está formado por un openMSP430 configurado con 4kB de memoria de programa y 256B de RAM, se utiliza un reloj principal de 20MHz, todo esto configurado en una Spartan 6.

El programa que corre en el openMSP430 es una multiplicación de matrices que se realiza en *loop* infinito, y verifica que el resultado sea correcto.

5.6.2 Opciones generales de la herramienta

Como ya se vio en el apartado 5.5 las opciones de las herramientas de síntesis pueden generar variaciones en los consumos del circuito implementado. En el caso del openMSP430 los resultados pueden verse en la Tabla 5-19.

Tabla 5-19 - Consumos del openMSP430 con diferentes opciones de la herramienta

Estrategias del ISE	Consumo (mW)	Factor de reducción
Balanced	21,9	1.00
Power Optimization	18,7	0.85
Area Reduction	19,0	0.87
Timing Performance	21,5	0.98

5.6.3 Opciones de la herramienta al instanciar las memorias

Para implementar las memorias del sistema se utiliza la herramienta Core Generator de ISE. El Core Generator permite controlar los parámetros necesarios para especificar la memoria requerida, por ejemplo: tamaño, ancho de palabra, puertos, etc.

Las pruebas se realizan utilizando memoria distribuída en LUTs y utilizando los bloques de memoria que dispone la Spartan 6. Para el caso de bloques de memoria, se prueban las 3 opciones relativas al algoritmo que utiliza la herramienta para concatenar bloques de memoria que son: "*Minimum Area*", "*Low Power*" y "*Fixed Primitives*". En este último caso el usuario puede seleccionar qué primitiva de memoria utilizar, ya sean bloques de 9Kbits (RAMB8BWER) o de 18Kbits

(RAMB16BWER), que se corresponden con los bloques que la Spartan tiene en hardware.

La memoria del sistema es de 4K bytes de memoria de programa y 256 bytes de memoria RAM, organizada de modo que pueda leerse en palabras de 16 bits, o sea que lógicamente son dos bloques de 2048 x 8, y dos de 128 x 8. Por el lado de la Spartan 6, los bloques de 9Kb pueden utilizarse como una memoria de 1024 x 8, y los de 18Kb como 2048 x 8. La memoria de programa utiliza bloques completos, pero siempre son necesarios 2 bloques más para alojar la RAM, el resumen de recursos utilizados después de la síntesis puede verse en la Tabla 5-20.

Tabla 5-20 - Recursos utilizados al instanciar las memorias del openMSP430 en una Spartan 6

Opciones	LUTs usadas como memoria		9K Block RAM		18K Block RAM	
Distributed Memory	544/2176	25%	0/64	0%	0/32	0%
Minimum Area	0/2176	0%	2/64	3%	2/32	6%
Low Power	0/2176	0%	6/64	9%	0/32	0%
Fixed Primitves 9Kb	0/2176	0%	6/64	9%	0/32	0%
Fixed Primitves 18Kb	0/2176	0%	0/64	0%	4/32	12%

Por ejemplo con la opción *Minimum Area* se utilizan completamente 2 bloques de 18Kbits que sirven para alojar toda la memoria de programa de 2048 x 8, y se agregan 2 bloques más para la RAM de 128 x 8, que en este caso el sintetizador los asigna a bloques de 9K, gran parte de estos bloques queda sin utilizar. En cambio en el caso de memoria distribuída se utilizan los bits de memoria de las LUTs que para la Spartan 6 son 64 bits por LUT. La memoria total del sistema en bits es:

$$4096 * 8 + 256 = 34816$$

Si cada LUT puede alojar 64 bits entonces la cantidad de LUTs necesaria es:

$$\frac{34816}{64} = 544$$

en este caso no se desperdicia nada y no queda memoria sin utilizar.

Una vez que se hace el *mapping* es interesante ver dónde se ubica la memoria dentro del chip, esto puede hacerse con la herramienta PlanAhead. En la Figura 5-13, los bloques de memoria BRAM de 9k y 18k están ubicados en las columnas verticales, señalizados con un círculo rojo. En el primer cuadro se observan estas columnas vacías debido a que la memoria está distribuida en los bloques lógicos de la FPGA.

Distibuted Memory

Minimum Area

Low Power

Fixed Primitves 9Kb

Fixed Primitves 18Kb

Figura 5-13 Ubicación de las memorias

Ahora que se ha explicado cómo se utilizan los recursos de memoria en cada opción se verán los resultados de las medidas de consumo realizadas que están resumidos en la Tabla 5-21.

Tabla 5-21 - Consumos para diferentes opciones de memoria

Opciones	Consumo (mW)
Distributed Memory	22.5
Minimum Area	20.7
Low Power	19.9
Fixed Primitves 9Kb	20.6
Fixed Primitves 18Kb	21.0

Puede verse que la opción que consume más es la de memoria distribuída, y la que consume menos es la de *Low Power,* pero aquí las variaciones con respecto a la utilización de bloques de memorias hardware son mucho menores que en los casos en que utilizaron bloques multiplicadores de la sección 5.2.

5.6.4 *Clock gating* a las memorias

Se aplica la técnica de *clock gating* a los bloques de memoria de programa y RAM del sistema. La idea es que la señal de reloj llegue a las memorias solamente cuando están siendo accedidas y que se bloquee cuando el microcontrolador está realizando otras operaciones.

Hay que tener en cuenta que si bien las memorias están separadas en byte alto (*hi*) y bajo (*lo*), el microcontrolador las accede como una palabra de 16 bits.

También se hacen pruebas particionando las memorias originales en dos y en cuatro, aplicando *clock gating* a distintas combinaciones. La idea de particionar es tener bloques más chicos de los cuales sólo uno esté activo a la vez.

Cuando no se utiliza *clock gating*, el reloj del sistema está conectado directamente al reloj de cada bloque de memoria. La modificación que se realiza es utilizar las mismas señales de habilitación *pmem_ce* y *dmem_ce* que se conectan las memorias a través de la entrada *ena* para generar un nuevo reloj que se active con dichas señales. La conexión original de las señales de la memorias se detalla en la Figura 5-15. Puede verse que si bien el reloj llega siempre a las memorias, las mismas cuentan ya con una entrada *ena* que en cierta forma las deshabilita, por lo tanto no es de esperar que se obtengan reducciones de relevancia.

Tabla 5-22 - Configuración original de las memorias

ram_8x128 ram_8x128_hi (.addra (dmem_addr), .clka (mclk), .dina (dmem_din[15:8]), .douta (dmem_dout[15:8]), .ena (dmem_ce), .wea (dmem_we[1]));	ram_8x128 ram_8x128_lo (.addra (dmem_addr), .clka (mclk), .dina (dmem_din[7:0]), .douta (dmem_dout[7:0]), .ena (dmem_ce), .wea (dmem_we[0]));
rom_8x2k_hi rom_8x2k_hi (.addra (pmem_addr), .clka (mclk), .dina (pmem_din[15:8]), .douta (pmem_dout[15:8]), .ena (pmem_ce), .wea (pmem_we[1]));	rom_8x2k_lo rom_8x2k_lo (.addra (pmem_addr), .clka (mclk), .dina (pmem_din[7:0]), .douta (pmem_dout[7:0]), .ena (pmem_ce), .wea (pmem_we[0]));

Para implementar el *clock gating*, se utiliza un bloque que ya viene incluido en el *core* del openMSP430, llamado *omsp_clock_gate* e implementado como se muestra en la Figura 5-14.

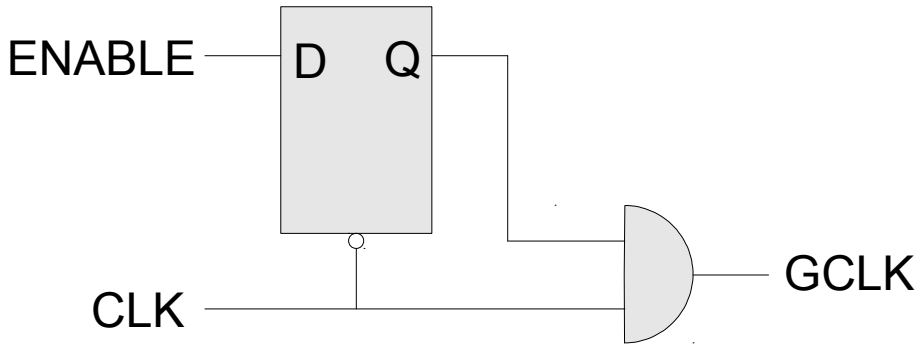


Figura 5-14 Implementación del bloque de *clock gating osmp_clock_gate*

A modo de ejemplo se muestra a continuación la instanciación de ese bloque para aplicar *clock gating* al bloque ROM_HI.

```
omsp_clock_gate gateROM_hi(
.gclk(gclk_ROM_hi),
.clk(mclk),
.enable(pmem_ce),
.scan_enable(1'b0)
);
```

En este ejemplo, *mclk* es el reloj principal del sistema, mientras que *gclk_ROM_hi* es la señal de reloj que ingresa al bloque *rom_8x2k_hi*.

En la Figura 5-15 se puede apreciar cómo la señal de reloj *gclk_ROM_hi* se activa únicamente cuando *pmem_ce*, = 1.

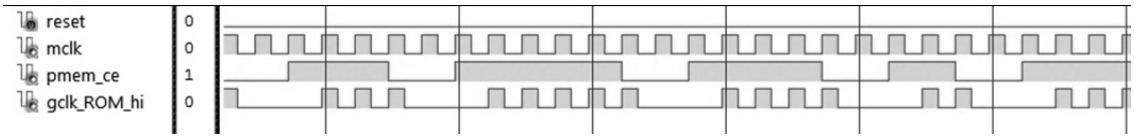


Figura 5-15 Reloj original mclk, y gated clock gclk_ROM_hi

Se utiliza la misma señal de habilitación tanto para los bloques *hi* como para los bloques *lo*, no es posible habilitarlos por separado debido a cómo están implementados los accesos del openMSP430 a memoria que son todos de 16 bits.

Tabla 5-23 - Resultados obtenidos al realizar *clock gating* en las memorias

	Consumo (mW)	Factor de reducción
Sin *clock gating*	20,9	1
ROM	19.7	0.94
RAM	18.7	0.89
ROM + RAM	20.1	0.96

La siguiente prueba consiste en particionar las memorias, primero en bloques de la mitad de su tamaño original, y después en bloques de la cuarta parte. Esta prueba implica varias modificaciones al sistema, ya que además de generar todos los bloques de memoria necesarios se requiere generar las señales de habilitación de cada bloque decodificándolas de las direcciones y las señales de control del procesador. Es necesario incluir un bloque multiplexor para juntar los datos de los bloques hacia el micro, y resolver algún problema de tiempos retardando los mismos un período de reloj. Todo este trabajo aporta muy poco ya que la mejor de las pruebas, en todos los casos, apenas puede bajar el consumo a 18.5mW, cuando en el caso de *clock gating* a la RAM ya se había obtenido 18.7mW.

5.6.5 Utilización del multiplicador hardware

Los MSP430 de Texas Instruments tienen versiones con y sin multiplicadores hardware, en el caso del openMSP430 esto es configurable en el momento de la síntesis. Para que todo funcione correctamente, hay que darle además una directiva al compilador *gcc* para indicarle que el multiplicador hardware está presente.

Se puede ver que cuando se incluye el multiplicador, la herramienta lo reconoce y en el proceso de mapeo utiliza un bloque DSP para su implementación, como se muestra a continuación en el reporte del ISE:

```
Number of DSP48A1s:       1 out of 32    3%
```

El multiplicador se mapea como un periférico del microcontrolador en las direcciones 0x0130 a 0x013C. Para realizar el *clock gating* se genera una señal de *enable* que está activa sólo cuando el bus de direcciones de los periféricos, *per_addr*, apunta a las direcciones que le corresponden al multiplicador.

La primer prueba se realiza agregando el multiplicador hardware y aplicámdole *clock gating*. Como el programa de pruebas es una multiplicación de matrices es de esperar que el multiplicador se utilice bastante. Los resultados obtenidos se ven en la Tabla 5-24.

Tabla 5-24 - Resultados obtenidos al aplicar *clock gating* al multiplicador HW

	Consumo (mW)	**Factor de reducción**
Sin *clock gating*	20,6	1
Con *clock gating*	18.7	0.91

Además de estudiar la efectividad o no de la técnica de aplicar *clock gating* al multiplicador hardware, resulta interesante comparar el consumo del sistema al realizar las operaciones utilizando el multiplicador y sin utilizarlo. El problema es que en este caso la comparación en potencia no tiene sentido, ya que el multiplicador hardware es mucho más rápido que realizar las operaciones por software. Para salvar este problema se diseña otro experimento basado en comparar la energía consumida en realizar determinada cantidad de operaciones.

Se sintetiza otra versión del openMSP430 que incluye modos de bajo consumo LPM (*Low Power Modes*), y con esta versión del microcontrolador a su vez se implementan 3 sistemas tal como se describe en la Tabla 5-25.

Tabla 5-25 - Sistemas para pruebas de energía

Sistema	**Multiplicador HW**	***Clock gating***
1	NO	--
2	SI	NO
3	SI	SI

En el sistema 1 no se incluye el multiplicador hardware y el software es compilado con las operaciones emuladas. En el sistema 2 sí se incluye el multiplicador y se fuerza su uso en la compilación del software. En la tercera prueba, se utiliza el multiplicador con el *clock gating* implementado.

El software que se utiliza en estos 3 casos realiza 50.000 veces la multiplicación de dos matrices y luego lleva al microprocesador al modo de bajo consumo LPM3.

Para obtener las medidas de potencia instantánea y calcular la energía a partir de ellas se utiliza el programa en Matlab detallado en la sección 3.6.1, con algunas modificaciones que permitieron tomar 80 muestras de la potencia instantánea del rail de 1,2V cada 0,5 segundos. Los resultados obtenidos en las tres pruebas se muestran en las Figura 5-16, Figura 5-17 y Figura 5-18.

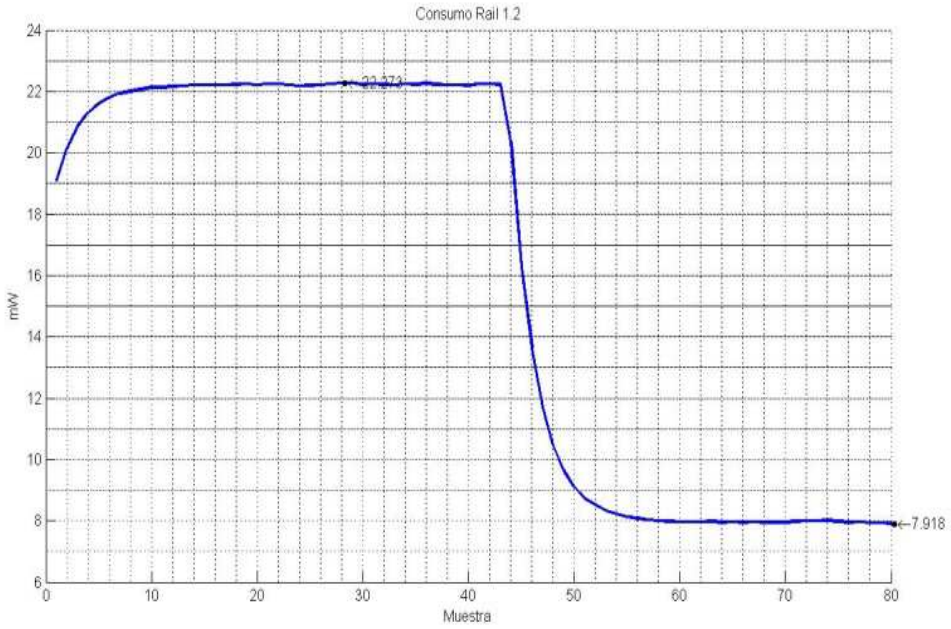


Figura 5-16 Sistema 1: sin multiplicador HW

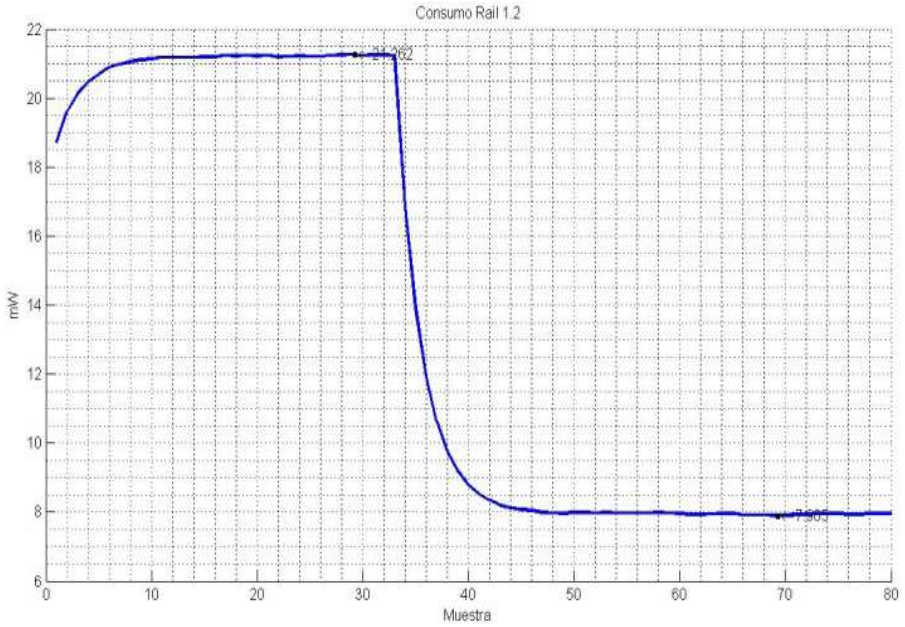


Figura 5-17 Sistema 2: multiplicador HW sin *clock gating*

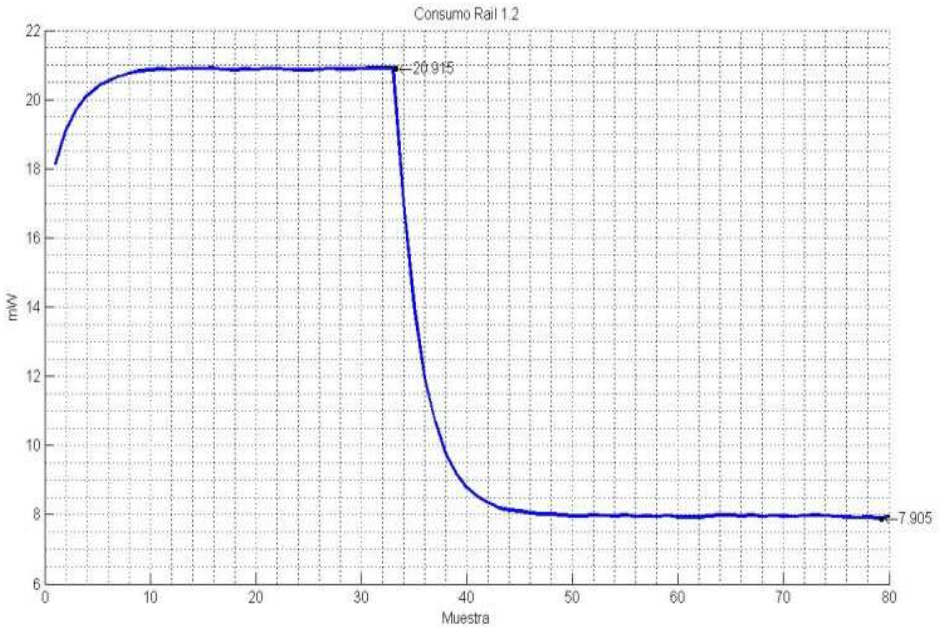


Figura 5-18 Sistema 3: con Multiplicador HW, con *clock gating*.

De acuerdo a lo esperado puede observarse que el tiempo que tarda el microcontrolador en realizar las 50.000 operaciones se reduce de aproximadamente 22 segundos a 17 segundos al utilizar el multiplicador HW.

La Tabla 5-26 muestra las medidas de potencia obtenidas y la energía calculada en un intervalo fijo de 40 segundos. Durante el tiempo T_on el microcontrolador está efectuando las 50.000 multiplicaciones de matrices y en T_off se encuentra en el modo de bajo consumo LPM3. Las medidas fueron realizadas a una temperatura ambiente de 28,6 °C y se midió una potencia estática de 5,9 mW.

Tabla 5-26 - Medidas de potencia y cálculo de energía

Sistema	Potencia ON (mW)	Potencia Off (mW)	T_on (s)	T_off (s)	Energía (mJ)
1	22.2	8.0	22.22	17.78	635.32
2	21.2	8.0	16.94	23.06	542.80
3	20.9	8.0	16.95	23.05	537.38

Hay varios análisis que pueden hacerse de estos datos. El primero es que el modo de bajo consumo del openMSP430 no es bueno, se arranca de una base mala, ya que el consumo estático de la Spartan 6 es elevado, pero el modo LPM3 lo incrementa de 5.9mW a 8.0mW. El segundo punto a destacar es que, si bien se pasa de una multiplicación por software a una multiplicación por hardware utilizando un bloque DSP, la reducción de consumo en modo activo no es demasiado significativa; y está lejos de las obtenidas en la sección 5.2. Está claro que son sistemas diferentes, en la sección 5.2 se comparan multiplicadores implementados en LUTs contra multiplicadores realizados con bloques embebidos (o DSPs), y acá se compara un sistema completo con un microcontrolador, en el cual éste realiza una multiplicación entre matrices por software o por hardware a través de un periférico implementado con un bloque DSP. De todas maneras las diferencias son abismales, en la sección 5.2 se obtienen ahorros del 80% y en este caso apenas del 6%.

El ahorro mayor se da en la energía en donde se llega a un 15%, pero tampoco es una cifra demasiado importante, y claramente el aporte del *clock gating* en este caso no es significativo.

5.6.6 *Clock gating* a los registros

El autor del openMSP430 incluye la posibilidad de realizar *clock gating* a todos los registros del microcontrolador con una opción llamada *Fine Grained Clock Gating*. Hoy en día casi todas las herramientas de síntesis tienen opciones automáticas para hacer un *fine grained clock gating,* pero en este caso se tiene la opción de hacerlo desde los fuentes Verilog.

El propio autor del *IP core* no recomienda la utilización de su versión con *fine grained clock gated* en FPGAs, y se verá a continuación que de acuerdo a las medidas obtenidas en los consumos esta recomendación es acertada.

Tabla 5-27 - Medidas de consumo con *fine grained clock gating*

Versión sintetizada del openMSP430	*Clock gating*	Potencia (mW)
Sin posibilidad de *clock gating*	---	17.6
Con posibilidad de *clock gating*	off	19.8
Con posibilidad de *clock gating*	on	17.8

En la Tabla 5-27 se ve que la opción sintetizada para poder hacer el *fine grained clock gating* consume más que la versión original, incluso con el *clock gating* activo.

A pesar del mal resultado obtenido, pero considerando que el *fine grained clock gating* es una herramienta potente, se realizan más experimentos. El primero de ellos consiste en estudiar la actividad de los registros, agruparlos según dicha actividad y aplicarles *clock gating* a los diferentes grupos. Para esto se realizan simulaciones del sistema completo, se genera el archivo SAIF y se obtiene la actividad por cada registro del microcontrolador. Los resultados no generan ni reducciones ni aumentos significativos de consumo, por lo tanto se omiten los detalles.

Un segundo conjunto de experimentos consiste en realizar *clock gating* a cada uno de los registros del microcontrolador por separado. Teniendo en cuenta que el microcontrolador tiene 40 registros esto implica realizar 40 pruebas con síntesis y medidas, los resultados se muestran en la Tabla 5-28

Tabla 5-28 - Medidas de consumo aplicando *clock gating* a cada registro por separado

***Clock gating* al registro**	**mW**	**°C**	**%**	**reduce**
(ninguno)	17,3	21,8	100%	0%
mdb_out_nxt	15,8	21,7	91%	9%
mdb_in_buf	15,9	21,7	92%	8%
irq_num	16,8	21,7	97%	3%
r0 (pc)	15,3	21,8	88%	12%
inst_sext	15,6	21,8	90%	10%
inst_dext	15,3	21,8	88%	12%
inst_type	16,9	21,7	98%	2%
inst_so	16,9	21,8	98%	2%

inst_jmp_bin	16,5	21,8	95%	5%
inst_mov	16,4	21,8	95%	5%
inst_dest_bin	17,1	21,7	99%	1%
inst_src_bin	16,8	21,8	97%	3%
inst_as	16,9	21,8	98%	2%
inst_ad	17,1	21,8	99%	1%
inst_sz	16,3	21,9	94%	6%
inst_alu	17,2	21,9	99%	1%
pmem_dout_bckup	16,4	21,9	95%	5%
op1	16,2	21,9	94%	6%
op2	16,3	23,3	94%	6%
reslo	16,3	23,3	94%	6%
reshi	16,1	23,3	93%	7%
sign_sel	16,6	23,3	96%	4%
acc_sel	15,9	23,3	92%	8%
wdtctl	15,8	23,3	91%	9%
wdtcnt	16,4	23,3	95%	5%
r1	14,9	21,9	86%	14%
r2	15,0	21,9	87%	13%
r3	15,6	23,3	90%	10%
r4	15,7	23,3	91%	9%
r5	16,5	23,3	95%	5%
r6	15,8	23,3	91%	9%
r7	16,8	23,3	97%	3%
r8	15,8	23,3	91%	9%
r9	16,8	23,3	97%	3%
r10	16,6	23,3	96%	4%
r11	16,9	23,3	98%	2%
r12	16,4	23,3	95%	5%
r13	15,3	22,0	88%	12%
r14	15,9	23,3	92%	8%
r15	16,8	23,3	97%	3%

Ordenando en forma decreciente los ahorros obtenidos, se realiza la gráfica que se muestra en la Figura 5-19. Como puede apreciarse hay un grupo de 5 registros para los cuales se produce un ahorro mayor al 10%.

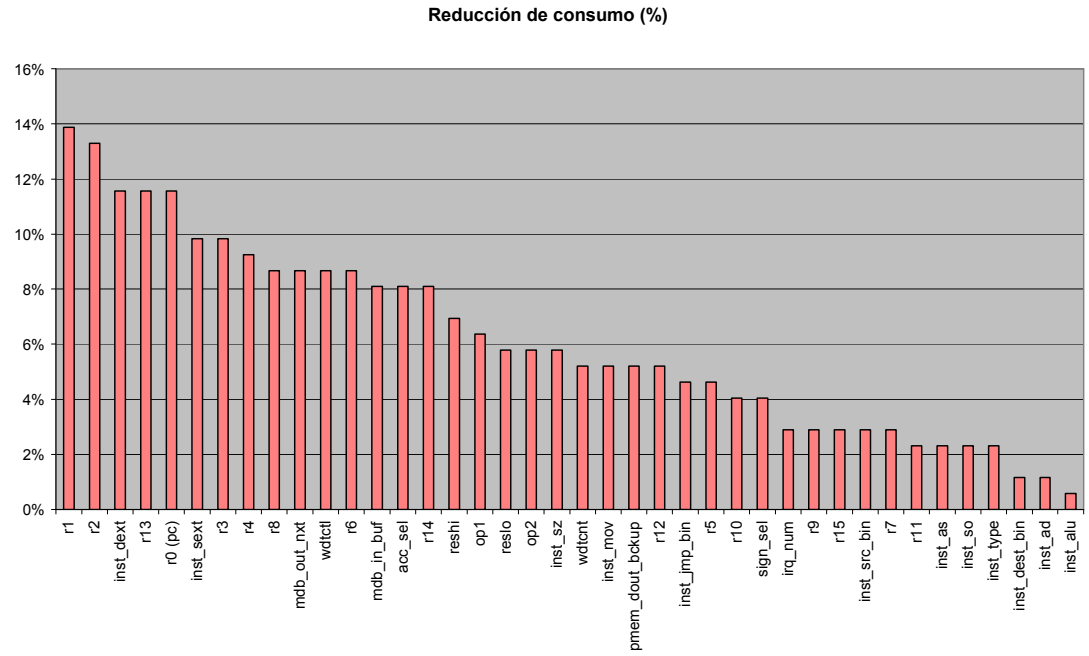


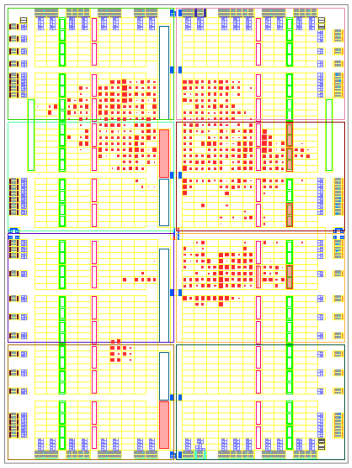
Figura 5-19 Ahorro de potencia aplicando *clock gating* a cada registro por separado.

5.6.7 Variaciones del *Placement* usando PlanAhead

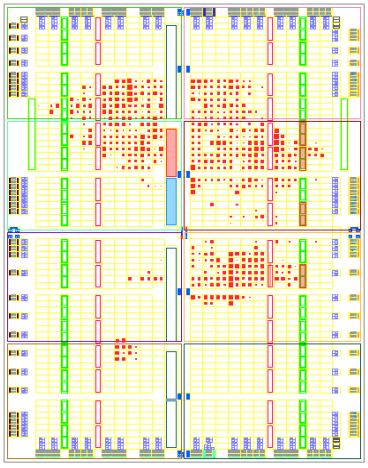
Utilizando el FloorPlanning de la herramienta PlanAhead es posible cambiar la ubicación de la lógica, las memorias y los distintos componentes del sistema, en este caso el microcontrolador y sus memorias. La idea de estos experimentos es entonces variar la posición de algunos elementos y evaluar su impacto en el consumo. En particular se busca reducir el área del chip. En todos los casos se utiliza la versión optimizada para FPGAs del openMSP430, corriendo un programa de multiplicación de matrices en bucle infinito.

Las distintas configuraciones de *placement* estudiadas se muestran gráficamente en la Figura 5-20. Pueden verse en rojo las zonas ocupadas de la lógica, en las columnas centrales los DCMs y las columnas a los lados los bloques de memoria BRAM.

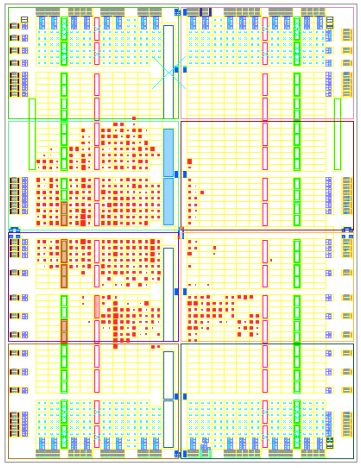
0



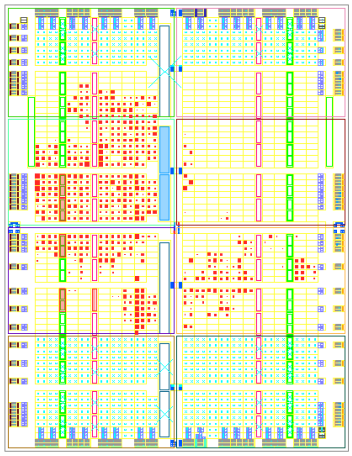
1



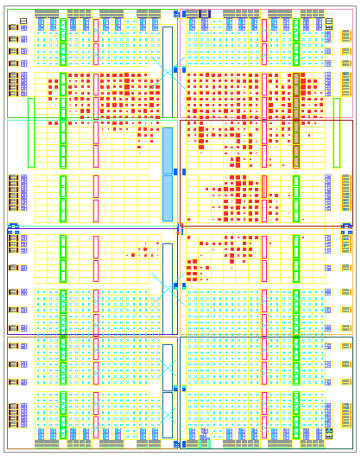
2



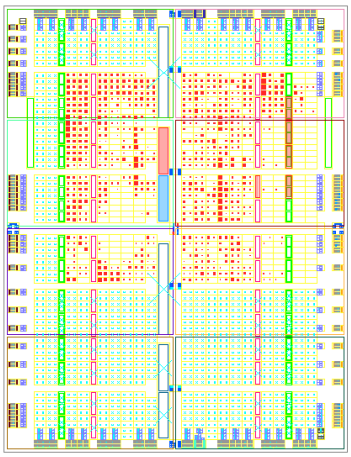
3



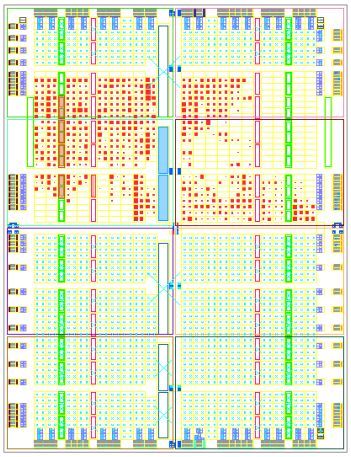
4



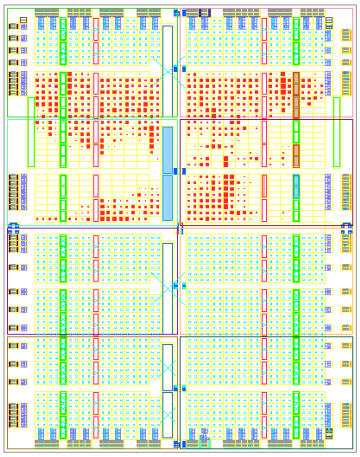
5



6



7



8

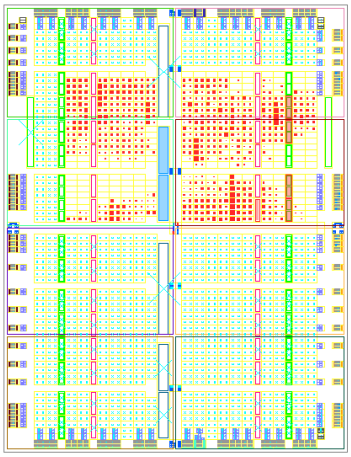


Figura 5-20 Configuraciones de *placement* estudiadas.

En las pruebas 1, 12 y 14 lo que se hace es simplemente mover uno de los DCM hacia una ubicación contigua al otro de ellos, y en la 16 se mueve la ubicación de uno de los BRAM.

Los resultados se muestran en la Tabla 5-29, en ninguna de las configuraciones realizadas se logra una reducción sustancial de consumo, en cambio en varias de ellas el consumo aumenta significativamente. El mejor resultado corresponde la configuración 6, pero la reducción es de apenas un 2,7% del consumo de potencia total, y el peor a la 11 con un aumento del 14,2%.

Tabla 5-29 - Medidas de consumo para las distintas configuraciones de *placement*.

Placement	**Potencia (mW)**	**Reducción (%)**	**Temp. (°C)**
0	18,3	0,0%	28,1
1	18,2	0,5%	28,1
2	18,4	-0,5%	28,1
3	18,8	-2,7%	28,1
4	18,2	0,5%	28,1
5	18,7	-2,2%	28,1
6	17,8	2,7%	28,1
7	18,3	0,0%	28,1
8	18,7	-2,2%	28,1
9	18,0	1,6%	28,1
10	18,5	-1,1%	28,1
11	20,9	-14,2%	28,1
12	20,8	-13,7%	28,1
13	18,2	0,5%	28,1
14	18,3	0,0%	28,1
15	18,6	-1,6%	28,1
16	18,9	-3,3%	28,1
17	19,2	-4,9%	28,1
18	19,4	-6,0%	27,9
19	20,2	-10,4%	27,9
20	19,8	-8,2%	27,9

5.6.8 Superposición de técnicas

Para el sistema completo del openMSP430 se realizan varios experimentos superponiendo las técnicas de reducción de consumo aplicadas hasta ahora. Para cada una de ellas se seleccionan los dos casos que habían dado mejores resultados en pruebas anteriores y se combinan de a dos y de a tres para ver los consumos con la superposición.

En casi todos los casos se obtienen resultados iguales o peores que al aplicar una técnica sola; y en los pocos casos en que se obtienen mejoras, éstas no son significativas.

5.7 Conclusiones del capítulo

En este capítulo se aplican varias técnicas de reducción de consumo, utilizando todas las herramientas desarrolladas previamente en esta tesis. Los aportes más destacables son:

- Confirmación y actualización de resultados en reducción de consumo utilizando *pipelining*.
- Cuantificación de la reducción de consumo por la utilización de multiplicadores embebidos.
- Caracterización de 3 técnicas de apagado parcial de zonas del diseño: *clock gating*, *clock enable* y bloqueo de entradas.
- Cuantificación del consumo de la opción *Suspend* en Spartan 6.
- Estudio en profundidad del impacto de las opciones de las herramientas en el consumo.
- Aplicación de múltiples técnicas al microcontrolador *softcore* openMSP430.

Capítulo 6

Conclusiones

En este capítulo se reseñan las conclusiones globales y los aportes de la tesis y finalmente se discuten algunas líneas de trabajo futuro.

El primer tema a destacar es que, en un diseño electrónico, la variable consumo no se puede tratar de la misma manera que el área o la frecuencia máxima. El consumo presenta particularidades que la hacen más compleja (dependencia de los datos, consumo pasivo, *glitches*, etc.). Por lo tanto es mucho más difícil de estimar, y no puede ser manejada con valores exactos en etapas tempranas de un diseño. Estas incertidumbres se agravan en las FPGAs, ya que al ser circuitos configurables por el usuario, a las dependencias antes mencionadas se agrega que el consumo dependerá de dicha configuración y de cómo fue mapeada en el chip.

Una parte de los trabajos experimentales realizados en esta tesis muestra que esta dificultad en estimar el consumo de las FPGAs, se ve reflejada en herramientas profesionales ampliamente utilizadas como son las ofrecidas por los fabricantes. Simulando circuitos relativamente sencillos, y estimando el consumo con dichas herramientas en las mejores condiciones posibles se obtuvieron, en algunos casos, errores superiores al 140%. Esta tasa de error es inaceptable para muchas de las aplicaciones actuales en donde el consumo es una de las restricciones más importantes. ¿Qué alternativas tiene entonces un usuario de FPGAs que necesite conocer en forma un poco más exacta el consumo de sus diseños? ¿Medir? Sí, una de las conclusiones de esta tesis es que, al menos con el desarrollo actual de las

herramientas de estimación, se hace indispensable realizar mediciones de consumo. Está claro que no se está afirmando que todos los usuarios de FPGAs deban medir el consumo de todos sus diseños, pero lo que sí debería hacerse, al menos, es calibrar las herramientas de estimación que se vayan a utilizar realizando algunas medidas.

Como consecuencia de esto, uno de los aportes de esta tesis, es una metodología completa de medición de consumo para FPGAs, que permite, por ejemplo, la calibración de herramientas de estimación. Como parte de esta metodología se incluyen los circuitos eléctricos necesarios para realizar las medidas y un conjunto de diseños o *benchmarks* para realizar pruebas incluyendo generadores de vectores de entradas.

Las principales características de esta metodología son:

- circuitos eléctricos muy simples
- conjunto de *benchmarks*
    - abarcan diferentes tipos de aplicaciones
    - incluyen generadores de vectores de entradas
    - son circuitos que funcionan
    - se incluyen *testbenches* para su simulación
    - probados en Altera y Xilinx
    - facilidad de cambiar parámetros:
        - frecuencia de reloj
        - en multiplicadores: etapas de *pipeline,* ancho de datos

La principal diferencia es utilizar *benchmarks* que además de sintetizarse y simularse se pueden programar y probar en una FPGA.

Se ha desarrollado además una herramienta específica de automedida de consumo para FPGAs. Esta herramienta, que también utiliza recursos muy sencillos y de muy bajo costo, tiene dos tipos de aplicaciones potenciales de interés que permitiran ampliar y mejorar los experimentos de bajo consumo en futuras investigaciones.

La primera de ellas es que permite obtener perfiles de consumo de un determinado diseño y almacenarlos, de modo que es posible tener una curva de consumo en función del tiempo. Esto puede ser muy útil en aplicaciones tipo SoC (o SoPC) con módulos heterogéneos que pasan de modo activo a *standby*. Por ejemplo se puede obtener el perfil de consumo de una aplicación que incluya un microprocesador y varios módulos de procesamiento hardware, con apagado por *clock gating* (o mejor aún con *power gating* en el caso de existir) con variaciones del ciclo de trabajo entre los módulos. Está claro que la obtención de perfiles de consumo también puede hacerse con instrumentos externos, pero la herramienta desarrollada en esta tesis es mucho más potente ya que permite generar condiciones de registro o de "disparo" con señales internas del propio diseño, similares a las de un analizador lógico.

La segunda aplicación de esta herramienta es que los circuitos programados en la FPGA pueden conocer su propia potencia consumida y con ese dato tomar acciones o seguir diferentes estrategias. Esto puede hacerse con la potencia instantánea, o bien utilizar el sistema como un totalizador de carga e ir llevando la cuenta de la energía consumida, algo utilizado en sistemas alimentados a batería.

En cuanto a las diferentes técnicas de bajo consumo que se experimentaron en esta tesis, no se pueden sacar las mismas conclusiones para todas ellas. La que da resultados más contundentes es la utilización de bloques embebidos (multiplicadores hardware o bloques DSP). La utilización de multiplicadores hardware frente a una implementación en LUTs siempre genera ahorros de potencia que pueden ir desde 65% a un 88% en los casos analizados. La conclusión entonces, es que siempre se debe optar por la utilización de los recursos embebidos de las FPGAs, en la medida que estos estén disponibles. Otra técnica que siempre da resultados positivos es el *pipelining* pero con reducciones de consumo menores, que en los ejemplos relevados van desde un 34% hasta un 71%. Esta técnica no siempre es tan fácil de utilizar, ya que la posibilidad de implementar *pipelines* en un determinado circuito va a depender de la arquitectura de la aplicación, pero por otro lado no tiene penalizaciones en área ya que en las FPGAs los *flip-flops* ya existen dentro de las celdas, se utilicen o no.

La variación de las opciones de las herramientas de diseño también pueden producir reducciones del consumo, pero con resultados bastante inciertos. No siempre las mismas opciones son las que dan el menor consumo, y algunas veces el consumo incluso aumenta. El no poder sacar reglas generales no hace que esta técnica deba ser descartada, ya que con muy pocos esfuerzos pueden lograrse reducciones de un 15%, llegando hasta un 38% en alguno de los experimentos. Pero el no poder generalizar los resultados la transforma en una técnica de prueba y error, en la cual en algunos casos se pueden obtener buenos resultados o en otros nada. La penalización en este caso no es en utilización del chip, sino que se mediría en tiempo de diseñador.

Con respecto a la aplicación de *clock gating*, se puede decir que en general es una técnica que sirve, si pueden encontrarse en la arquitectura del diseño bloques que puedan apagarse. Dado que los recursos de las FPGAs para implementar *clock gating* en las líneas de reloj globales son escasos, la granularidad del *clock gating* debe ser grande. Implementar *clock gating* en grano fino no parece ser recomendable ya que en general acarrea serios problemas de tiempos que hay que resolver en forma cuidadosa (aunque sí fue realizado en los experimentos con el Open MSP430).

El caso de estudio del openMSP430 presenta algunas conclusiones interesantes. La primera surge de la comparación entre la versión FPGA y la versión comercial del MSP4340 de Texas Instruments. Si bien la versión FPGA siempre consume más, el consumo de ambas en modo activo es del mismo orden de magnitud; mientras que en modo *standby* el consumo de la versión FPGA es 10.000 veces mayor. Esta diferencia se debe a que el MSP430 comercial se "apaga" realmente, es decir que tiene implementados varios niveles de *power gating*, mientras que la FPGA tiene un consumo estático muy elevado.

Si bien nos interesó estudiar a fondo el caso del openMSP430 por varias razones (explicadas en la sección 5.6), del conjunto de *benchmarks* utilizados es de los que consume menos. Esto es debido a que es un diseño chico, más bien orientado a bloques de control, con poca utilización de hardware y caminos de datos reducidos, lo cual lleva a que se utilice solamente el 20% de las LUTs en el caso de la Spartan 6. En

este tipo de circuitos el peso del consumo estático frente al consumo total es muy grande y por esto muchas de las técnicas aplicadas dan magros resultados. Los experimentos muestran que sin mucho esfuerzo fue posible obtener reducciones de entre 10% y 15%, pero mejorarlas fue prácticamente imposible. En este caso fue el único que se intentó bajar el consumo mediante una reorganización del *placement*, pero tampoco se obtuvieron buenos resultados.

Otra conclusión llamativa es que la superposición de técnicas dio muy malos resultados, en este circuito particular los márgenes para mejorarlo son muy estrechos, y se ve que al superponer más de una técnica, al final se terminan empeorando las cosas. Esto no es una regla general, ya que como se vio en la sección 5.2, la aplicación superpuesta de *pipelining* en los bloques multiplicadores da excelentes resultados.

6.1 Publicaciones asociadas a esta tesis

J. Hormigo, G. Caffarena, J. P. Oliver, and E. Boemo, “Self-Reconfigurable Constant Multiplier for FPGA,” ACM Trans. Reconfigurable Technol. Syst., vol. 6, no. 3, pp. 14:1–14:17, 2013.

E. Boemo, J. P. Oliver, and G. Caffarena, “Tracking the pipelining-power rule along the FPGA technical literature,” in Proceedings of the 10th FPGAworld Conference, 2013, pp. 9:1–9:5.

J. P. Oliver, J. Pérez Acle, E. Boemo, "Power Estimations vs. Power Measurements in Spartan 6 Devices", in Programmable Logic (SPL), 2014 IX Southern Conference on, 2014.

J. P. Oliver, J. Curto, D. Bouvier, M. Ramos, and E. Boemo, “Clock gating and clock enable for FPGA power reduction,” in Programmable Logic (SPL), 2012 VIII Southern Conference on, 2012, pp. 1–5.

J. P. Oliver and E. Boemo, “Power estimations vs. power measurements in Cyclone III devices,” in Programmable Logic (SPL), 2011 VII Southern Conference on, 2011, pp. 87–90.

J. P. Oliver, F. Veirano, D. Bouvier, and E. Boemo, “A Low Cost System for Self Measurements of Power Consumption in FPGAs,” 2014, submitted

6.2 Líneas de trabajo futuras

Como líneas de trabajo futuro, un primer paso importante sería la ampliación y mejora de los circuitos de *benchmarks* con las mismas características que los actuales, es decir que cuenten con generadores internos de vectores de entrada y *test benches* para simulación. Se pueden publicar en forma abierta para que estén accesibles a la comunidad de diseñadores interesados en consumo. Con un buen conjunto de circuitos de prueba se puede trabajar en el test y calibración de herramientas de estimación de consumo. Hay una gran cantidad de artículos sobre consumo de FPGAs publicados solamente con datos basados en estimaciones y los errores que presentan estas herramientas podrían hacer cuestionable su validez.

Otra herramienta que puede ampliarse en el futuro es el automedidor de consumo. Esta herramienta es muy simple, pero a la vez muy poderosa para realizar un análisis en profundidad de un diseño. La utilización de esta herramienta facilitará y mejorará la realización de trabajos de investigación en consumo por ejemplo en la utilización de FPGAs en redes de sensores, área de creciente desarrollo como se puede ver en [145] y [146].

El alto consumo estático de las FPGAs actuales lleva a pensar que para que puedan ser utilizadas en aplicaciones de muy bajo consumo con períodos largos de *standby*, la única forma debería ser apagándolas totalmente. El problema que se presenta con el apagado total es que se pierde la configuración, y por lo tanto al encenderse se debe

cargar nuevamente el *bitstream* para poder funcionar. Este procedimiento es utilizado ya en algunas aplicaciones como por ejemplo la serie de trabajos publicados con la plataforma *Cookie* [147] y sus versiones posteriores [148], [149] y [150], que proponen un sistema con una FPGA y un pequeño microcontrolador externo que se encarga exclusivamente del encendido/apagado de la misma. Para reducir el tiempo de configuración se pueden utilizar diversas técnicas, como la compresión de los *bitstreams* como se puede ver en [151] o [152].

Las redes de sensores son un área que está demandando cada vez más capacidad de procesamiento, en la medida que ese procesamiento pueda ser acelerado por una FPGA y la ecuación total de energía sea mejor que con la utilización de otras tecnologías (microprocesadores, DSPs, GPUs); puede ser un nicho para aplicar los resultados de esta tesis.

Como palabras finales se incluyen algunos comentarios generales acerca de las FPGAs basadas en SRAM, que son las únicas estudiadas en esta tesis. Estos dispositivos son sumamente atractivos por su flexibilidad, pero a priori son elementos con un comportamiento muy malo desde el punto de vista de su consumo. Por otro lado el diseñador o usuario de estas FPGAs no tiene tantos grados de libertad como un diseñador VLSI, y sólo tiene disponibles los recursos que el fabricante pone en el chip. Aún así, y como se vio a lo largo de esta tesis, hay un margen considerable para reducir su consumo; pero este margen sería mucho mayor si los fabricantes incluyeran más posibilidades a nivel de silicio.

Hay una gran cantidad de trabajos que muestran posibles mejoras a nivel del circuito integrado de las FPGAs actuales con respecto a su consumo, algunas de estas ideas se han visto reflejadas en integrados comerciales y otras no. En un breve repaso podemos ver propuestas de incluir *power gating* a nivel de los bloques de conmutación [153], a nivel global [154], estudios sobre la granularidad de *power gating* [155], y una FPGA con *power gating* de grano fino [156]. Otros trabajos incluyen el uso voltajes duales de V_{dd} y V_t [157] [96], y su combinación con *power gating* [158], [159]. También se pueden encontrar trabajos recientes que proponen

reestructurar o mejorar los recursos disponibles para aplicar *clock gating* [138] y [139].

Se han publicado diseños de circuitos de bajo consumo a nivel de LUTs [160] y en elementos de conmutación [161], [162], [163]; y hay propuestas de circuitos para reducir las corrientes de fugas [34], [159], [164]. Es interesante ver en [165] un diseño completo, a nivel de transistor, de una FPGA de bajo consumo basada en una Spartan 3. Por último se pueden ver algunos artículos que proponen el uso de técnicas más sofisticadas: en [166] se propone usar el reciclado de carga para mejorar el consumo de los bloques de conmutación, y en [167] el desarrollo de una FPGA que funcione en modo *sub-threshold* con una única fuente Vdd.

El aumento de la demanda de aplicaciones de bajo consumo, y esta gran cantidad de trabajos, con propuestas a nivel de transistor para mejorar la performance de las FPGAs, llevan a pensar que en el futuro veremos que los fabricantes incluirán en sus dispositivos cada vez más características que darán a los diseñadores más y mejores formas de reducción de consumo.

Siguiendo con un poco de futurología, se enumeraran cuales son, a juicio del autor, las características que deberían incluirse en las próximas generaciones de FPGAs a corto plazo:

- mejoras *clock gating*
- *power gating*
- modos *standby* y módulos de *power management*

Las mejoras al *clock gating* son simplemente contar con más recursos para poder implementar un *clock gating* de grano un poco más fino que el actual, esta mejora no implicaría grandes cambios a los diseños actuales de los chips.

La inclusión de *power gating*, o zonas de la FPGA a las que se le pueda apagar su fuente de alimentación, sí sería ya una modificación mayor, tanto a nivel de silicio

como a nivel de herramientas de diseño, porque un usuario debería poder seleccionar dinámicamente qué zonas del chip apagar y cuales dejar encendidas.

Otra mejora importante sería incluir modos de *standby* que lleven al chip a un estado de ultra bajo consumo sin perder la programación. Esto debería asociarse, pensando en estructuras similares a las de los microcontroladores, con módulos de *power management*, es decir distintos niveles de módulos que queden funcionando y sean capaces de "despertar" al resto de la FPGA. Por ejemplo el usuario podría apagar todo el chip y dejar funcionando el PLL que divide el reloj y un contador, y que al llegar al final de la cuenta despertar al resto de los circuitos. Nuevamente para poder realizar un diseño con estas características sería necesario incorporar el manejo de las mismas desde las herramientas de desarrollo.

Bibliografía

[1] K. Keutzer and P. Vanbekbergen, "The impact of CAD on the design of low power digital circuits," in *Low Power Electronics, 1994. Digest of Technical Papers., IEEE Symposium*, 1994, pp. 42–45.

[2] J. D. Meindl, "A history of low power electronics: how it began and where it's headed," in *Proceedings of the 1997 international symposium on Low power electronics and design*, 1997, pp. 149–151.

[3] J. D. Meindl, *Micropower circuits*. Wiley, 1969.

[4] E. Keonjian, *Micropower electronics*, no. no. 77. Published for and on behalf of Advisory Group for Aeronautical Research and Development, North Atlantic Treaty Organization by Pergamon Press; [distributed in the Western Hemisphere by Macmillan, New York], 1964.

[5] C. Piguet, *Low-Power Electronics Design*, vol. 20043681. CRC Press, 2005.

[6] J. D. Meindl, "Low power microelectronics: retrospect and prospect," *Proc. IEEE*, vol. 83, no. 4, pp. 619–635, 1995.

[7] "The International Technology Roadmap for Semiconductors," *2011 Edition, Design Chapter*, 2011. [Online]. Available: http://www.itrs.net/Links/2011itrs/2011Chapters/2011Design.pdf.

[8] "The International Technology Roadmap for Semiconductors," *2013 Edition, Executive Summary*, 2013. [Online]. Available: http://www.itrs.net/Links/2013ITRS/Home2013.htm.

[9] K. Roy and S. Prasad, *Low-Power CMOS VLSI Circuit Design*. Wiley, 2000.

[10] "The International Technology Roadmap for Semiconductors," *2012 Ediction ORTC-Tables*, 2012. [Online]. Available: http://www.itrs.net/Links/2012ITRS/2012Tables/ORTC_2012Tables.xlsm.

[11] J. Koomey, "Growth in Data center electricity use 2005 to 2010," *Analytics Press*, 2011. [Online]. Available: http://www.analyticspress.com/datacenters.html.

[12] J. G. Koomey, "Worldwide electricity used in data centers," *Environ. Res. Lett.*, vol. 3, no. 3, p. 034008, Jul. 2008.

[13] I. Kuon and J. Rose, “Measuring the Gap Between FPGAs and ASICs,” *Comput. Des. Integr. Circuits Syst. IEEE Trans.*, vol. 26, no. 2, pp. 203–215, Feb. 2007.

[14] A. Amara, F. Amiel, and T. Ea, “FPGA vs. ASIC for low power applications,” *Microelectronics J.*, vol. 37, no. 8, pp. 669–677, 2006.

[15] E. Boemo, S. López-buedo, C. S. Pérez, C. Santos, J. Jáuregui, and J. Meneses, “Logic Depth and Power Consumption: A Comparative Study Between Standard Cells and FPGAs,” in *XIII Design of Circuits and Integrated Systems (DCIS) Conference*, 1998, pp. 402–406.

[16] H. J. M. Veendrick, “Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits,” *IEEE J. Solid-State Circuits*, vol. 19, no. 4, pp. 468–473, Aug. 1984.

[17] N. Hedenstierna and K. O. Jeppson, “CMOS Circuit Speed and Buffer Optimization,” *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 6, no. 2, pp. 270–281, Mar. 1987.

[18] W. Nebel and J. Mermet, Eds., *Low Power Design in Deep Submicron Electronics*. Boston, MA: Springer US, 1997.

[19] A. Bellaouar and M. I. Elmasry, *Low-Power Digital Vlsi Design*. Boston, MA: Kluwer Academic Publishers, 1995.

[20] M. J. S. Smith, *Application-Specific Integrated Circuits*. ADDISON WESLEY Publishing Company Incorporated, 1997.

[21] J. Rabaey, *Low Power Design Essentials*. Boston, MA: Springer US, 2009.

[22] S. Narendra and A. P. Chandrakasan, *Leakage in nanometer CMOS technologies*. New York: Springer, 2006, p. 307.

[23] S. P. Mohanty, N. Ranganathan, E. Kougianos, and P. Patra, *Low-Power High-Level Synthesis for Nanoscale CMOS Circuits*. Boston, MA: Springer US, 2008.

[24] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, “Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits,” *Proc. IEEE*, vol. 91, no. 2, pp. 305–327, Feb. 2003.

[25] Z. Abbas, V. Genua, and M. Olivieri, “A novel logic level calculation model for leakage currents in digital nano-CMOS circuits,” in *2011 7th Conference on Ph.D. Research in Microelectronics and Electronics*, 2011, pp. 221–224.

[26] A. Abdollahi, F. Fallah, and M. Pedram, “Leakage current reduction in CMOS VLSI circuits by input vector control,” *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 12, no. 2, pp. 140–154, Feb. 2004.

[27] S. Singh, B. Kaur, B. K. Kaushik, and S. Dasgupta, “Leakage current reduction using modified gate replacement technique for CMOS VLSI circuit,” in *2012 International Conference on Communications, Devices and Intelligent Systems (CODIS)*, 2012, pp. 464–467.

[28] A. P. Chandrakasan and R. W. Brodersen, *Low Power Digital CMOS Design*. Boston, MA: Springer US, 1995.

[29] C. Chiasson and V. Betz, “Should FPGAS abandon the pass-gate?,” in *2013 23rd International Conference on Field programmable Logic and Applications*, 2013, pp. 1–8.

[30] P. Chow, J. Rose, K. Chung, G. Paez-Monzon, and I. Rahardja, “The design of an SRAM-based field-programmable gate array. I. Architecture,” *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 7, no. 2, pp. 191–197, Jun. 1999.

[31] P. Chow, J. Rose, K. Chung, G. Paez-Monzon, and I. Rahardja, “The design of a SRAM-based field-programmable gate array-Part II: Circuit design and layout,” *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 7, no. 3, pp. 321–330, Sep. 1999.

[32] H. Hassan, M. Anis, and M. Elmasry, “Input Vector Reordering for Leakage Power Reduction in FPGAs,” *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 27, no. 9, pp. 1555–1564, Sep. 2008.

[33] J. H. Anderson, F. N. Najm, and T. Tuan, “Active leakage power optimization for FPGAs,” in *Proceeding of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays - FPGA ’04*, 2004, p. 33.

[34] J. H. Anderson and F. N. Najm, “Active leakage power optimization for FPGAs,” *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 25, no. 3, pp. 423–437, Mar. 2006.

[35] T. Tuan and B. Lai, “Leakage power analysis of a 90nm FPGA,” in *Proceedings of the IEEE 2003 Custom Integrated Circuits Conference, 2003.*, 2003, pp. 57–60.

[36] L. Shang, A. S. Kaviani, and K. Bathala, “Dynamic Power Consumption in Virtex-II FPGA Family,” in *Proceedings of the 2002 Tenth ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA’02)*, 2002, pp. 157–164.

[37] K. K. W. Poon, A. Yan, and S. J. E. Wilton, “A Flexible Power Model for FPGAs,” in *Proceedings of the Reconfigurable Computing Is Going Mainstream, 12th International Conference on Field-Programmable Logic and Applications, FPL’02*, 2002, pp. 312–321.

[38] F. Li, D. Chen, L. He, and J. Cong, “Architecture evaluation for power-efficient FPGAs,” *Proc. 2003 ACM/SIGDA Elev. Int. Symp. F. Program. gate arrays - FPGA ’03*, p. 175, 2003.

[39] F. Li, Y. Lin, L. He, D. Chen, and J. Cong, “Power modeling and characteristics of field programmable gate arrays,” *Comput. Des. Integr. Circuits Syst. IEEE Trans.*, vol. 24, no. 11, pp. 1712–1724, 2005.

[40] E. Todorovich, “Estimación Estadística de Consumo en FPGAs,” Universidad Autónoma de Madrid, 2006.

[41] R. Jevtic, “High-level Power Estimation of DSP Circuits Implemented in FPGAs,” Universidad Politécnica de Madrid, 2009.

[42] E. I. Boemo, “Contribución al diseño de arrays VLSI con paralelismo de grano fino,” Universidad Politécnica de Madrid, 1995.

[43] E. Boemo, G. González de Rivera, S. López-Buedo, and J. M. Meneses, “Some Notes on Power Management on FPGA-based Systems,” in *Proceedings of the 5th International Workshop on Field-Programmable Logic and Applications (FPL’95)*, 1995, pp. 149–157.

[44] E. Todorovich, G. Sutter, and N. Acosta, “End-user low-power alternatives at topological and physical levels. Some examples on FPGAs,” in *Proc. DCIS’2000*, 2000.

[45] E. Kusse and J. Rabaey, “Low-energy embedded FPGA structures,” in *Proceedings. 1998 International Symposium on Low Power Electronics and Design (IEEE Cat. No.98TH8379)*, 1998, pp. 155–160.

[46] J. Becker, M. Huebner, and M. Ullmann, “Power Estimation and Power Measurement of Xilinx Virtex FPGAs : Trade-offs and Limitations,” pp. 1–6, 2009.

[47] “Stratix II vs . Virtex-4 Power Comparison & Estimation Accuracy,” no. August. Altera Corporation, Wite paper, 2005.

[48] V. Degalahal and T. Tuan, “Methodology for high level estimation of FPGA power consumption,” in *Proceedings of the ASPDAC 2005 Asia and South Pacific Design Automation Conference 2005*, 2005, vol. 1, pp. 657–660.

[49] D. Meintanis and I. Papaefstathiou, “Power Consumption Estimations vs Measurements for FPGA-Based Security Cores,” *2008 Int. Conf. Reconfigurable Comput. FPGAs*, pp. 433–437, Dec. 2008.

[50] N. Abdelli, A.-M. Fouilliart, N. Julien, and E. Senn, “High-Level Power Estimation of FPGA,” in *2007 IEEE International Symposium on Industrial Electronics*, 2007, pp. 925–930.

[51] I. Herrera-Alzu, M. A. Sanchez, M. Lopez-Vallejo, and P. Echeverria, “Experimental methodology for power characterization of FPGAs,” in *Electronics, Circuits and Systems, 2008. ICECS 2008. 15th IEEE International Conference on*, 2008, pp. 582–585.

[52] H. G. Lee, K. Lee, Y. Choi, and N. Chang, “Cycle-Accurate Energy Measurement and Characterization of FPGAs,” *Analog Integr. Circuits Signal Process.*, vol. 42, no. 3, pp. 239–251, Mar. 2005.

[53] S. E. Wilton, S.-S. Ang, and W. Luk, “The Impact of Pipelining on Energy per Operation in Field-Programmable Gate Arrays,” in *Field Programmable Logic and Application SE - 73*, vol. 3203, J. Becker, M. Platzner, and S. Vernalde, Eds. Springer Berlin Heidelberg, 2004, pp. 719–728.

[54] S. Y. L. Chin, C. S. P. Lee, and S. J. E. Wilton, “Power Implications of Implementing Logic Using FPGA Embedded Memory Arrays,” in *Field Programmable Logic and Applications, 2006. FPL ’06. International Conference on*, 2006, pp. 1–8.

[55] R. Jevtic and C. Carreras, “Power Estimation of Embedded Multiplier Blocks in FPGAs,” *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 18, no. 5, pp. 835–839, May 2010.

[56] R. Jevtic and C. Carreras, “Power Measurement Methodology for FPGA Devices,” *IEEE Trans. Instrum. Meas.*, vol. 60, no. 1, pp. 237–247, 2011.

[57] P. Alfke, “Efficient Shift Registers, LFSR Counters, and Long Pseudo-Random Sequence Generators.” Xilinx Appl. note 052, 1996.

[58] M. George and P. Alfke, “Linear Feedback Shift Registers in Virtex Devices.” Xilinx Appl. note 210, 2007.

[59] J. Andersen and M. T. Hansen, “Energy Bucket: A Tool for Power Profiling and Debugging of Sensor Nodes,” in *2009 Third International Conference on Sensor Technologies and Applications*, 2009, pp. 132–138.

[60] I. Haratcherev, G. Halkes, T. Parker, O. Visser, and K. Langendoen, “PowerBench: A scalable testbed infrastructure for benchmarking power

consumption," in *Int. Workshop on Sensor Network Engineering (IWSNE*, 2008, pp. 37–44.

[61] X. Jiang, P. Dutta, D. Culler, and I. Stoica, "Micro Power Meter for Energy Monitoring of Wireless Sensor Networks at Scale," in *6th International Symposium on Information Processing in Sensor Networks*, 2007, pp. 186–195.

[62] Z. Nakutis, "A consumption current measurement approach for FPGA based embedded systems," in *2012 IEEE International Instrumentation and Measurement Technology Conference Proceedings*, 2012, pp. 328–333.

[63] Z. Nakutis, "A Current Consumption Measurement Approach for FPGA-Based Embedded Systems," *IEEE Trans. Instrum. Meas.*, vol. 62, no. 5, pp. 1130–1137, May 2013.

[64] C. Fernandez, D. Bouvier, J. Villaverde, L. Steinfeld, and J. Oreggioni, "Low-Power Self-Energy Meter for Wireless Sensor Network," in *2013 IEEE International Conference on Distributed Computing in Sensor Systems*, 2013, pp. 315–317.

[65] J. Villaverde, L. Steinfeld, J. Oreggioni, D. Bouvier, and C. Fernández, "Self-energy meter in duty-cycle battery operated sensor nodes," in *Instrumentation and Measurement Technology Conference (I2MTC)*, 2014.

[66] R. F. Coughlin and F. F. Driscoll, *Operational Amplifiers and Linear Integrated Circuits*. Prentice Hall, Pearson Education International, 2001, p. 529.

[67] C. A. Fernández, D. Bouvier, and J. Villaverde, "Proyecto SEM - Self Energy Meter," Universidad de la República, Uruguay, 2013.

[68] "CSS555C Micropower Timer (with Internal Timing Capacitor)," *Custom Silicon Solutions, Inc.*, 2012. [Online]. Available: http://www.customsiliconsolutions.com/downloads/Revised Standard products/CSS555C_Spec_2.pdf.

[69] F. Veirano and L. Briosso, "SEM en FPGA," Universidad de la República, 2013.

[70] "IEEE Standard for Design and Verification of Low-Power Integrated Circuits." pp. 1–348, 2013.

[71] P. E. Landman, "Low-Power Architectural Design Methodologies," University of California at Berkeley, 1994.

[72] C. Baena, J. Juan-Chico, M. J. Bellido, P. R. Clavijo, C. J. Jiménez, and M. Valencia, "Measurement of the Switching Activity of CMOS Digital Circuits at

the Gate Level," in *Integrated Circuit Design. Power and Timing Modeling, Optimization and Simulation SE - 35*, vol. 2451, B. Hochet, A. Acosta, and M. Bellido, Eds. Springer Berlin Heidelberg, 2002, pp. 353–362.

[73] "Cyclone III FPGAs: Optimized for Low Power." [Online]. Available: http://www.altera.com/devices/fpga/cyclone3/overview/power/cy3-power.html. [Accessed: 09-May-2013].

[74] M. J. Wirthlin, "Constant Coefficient Multiplication Using Look-Up Tables," *J. VLSI Signal Process. Signal, Image, Video Technol.*, vol. 36, no. 1, pp. 7–15, Jan. 2004.

[75] J. Hormigo, G. Caffarena, J. P. Oliver, and E. Boemo, "Self-Reconfigurable Constant Multiplier for FPGA," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 6, no. 3, pp. 14:1–14:17, 2013.

[76] M. C. Hansen, H. Yalcin, and J. P. Hayes, "Unveiling the ISCAS-85 benchmarks: a case study in reverse engineering," *IEEE Des. Test Comput.*, vol. 16, no. 3, pp. 72–80, 1999.

[77] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *IEEE International Symposium on Circuits and Systems*, 1989, pp. 1929–1934.

[78] S. Yang, "Logic Synthesis and Optimization Benchmarks User Guide," Research Triangle Park, NC, 1991.

[79] F. Brglez, "ACM/SIGDA Benchmarks Electronic Newsletter DAC'93 Edition," 1993.

[80] S. Kliman, "Prep benchmarks reveal performance and capacity tradeoffs of programmable logic devices," in *Proceedings Seventh Annual IEEE International ASIC Conference and Exhibit*, 1994, pp. 376–382.

[81] D. McCarty, D. Faria, and P. Alfke, "PREP Benchmarks for Programmable Logic Devices," in *Proceedings of the IEEE 1993 Custom Integrated Circuits Conference*, 1993, pp. 7.7.1–7.7.6.

[82] J. Babb, M. Frank, V. Lee, E. Waingold, R. Barua, M. Taylor, J. Kim, S. Devabhaktuni, and A. Agarwal, "The RAW benchmark suite: computation structures for general purpose computing," in *Proceedings. The 5th Annual IEEE Symposium on Field-Programmable Custom Computing Machines Cat. No.97TB100186)*, 1997, pp. 134–143.

[83] C. J. Alpert, "The ISPD98 circuit benchmark suite," in *Proceedings of the 1998 international symposium on Physical design - ISPD '98*, 1998, pp. 80–85.

[84] F. Corno, M. S. Reorda, and G. Squillero, “RT-level ITC’99 benchmarks and first ATPG results,” *IEEE Des. Test Comput.*, vol. 17, no. 3, pp. 44–53, 2000.

[85] C. Albrecht, “IWLS 2005 benchmarks,” in *International Workshop for Logic Synthesis (IWLS)*, 2005.

[86] T. Becker, P. Jamieson, W. Luk, P. Y. K. Cheung, and T. Rissa, “Towards benchmarking energy efficiency of reconfigurable architectures,” in *2008 International Conference on Field Programmable Logic and Applications*, 2008, pp. 691–694.

[87] P. Jamieson, T. Becker, W. Luk, P. Y. K. Cheung, T. Rissa, and T. Pitkänen, “Benchmarking Reconfigurable Architectures in the Mobile Domain,” in *2009 17th IEEE Symposium on Field Programmable Custom Computing Machines*, 2009, pp. 131–138.

[88] T. Becker, P. Jamieson, W. Luk, P. Y. K. Cheung, and T. Rissa, “Power characterisation for the fabric in fine-grain reconfiguralbe architectures,” in *2009 5th Southern Conference on Programmable Logic (SPL)*, 2009, pp. 77–82.

[89] P. Jamieson, T. Becker, P. Y. K. Cheung, W. Luk, T. Rissa, and T. Pitkänen, “Benchmarking and evaluating reconfigurable architectures targeting the mobile domain,” *ACM Trans. Des. Autom. Electron. Syst.*, vol. 15, no. 2, pp. 1–24, Feb. 2010.

[90] T. Pitkänen, P. Jamieson, T. Becker, S. Moisio, and J. Takala, “Power consumption benchmarking for reconfigurable platforms,” *Analog Integr. Circuits Signal Process.*, vol. 73, no. 2, pp. 649–659, Sep. 2012.

[91] D. W. Chang, C. D. Jenkins, P. C. Garcia, S. Z. Gilani, P. Aguilera, A. Nagarajan, M. J. Anderson, M. a. Kenny, S. M. Bauer, M. J. Schulte, and K. Compton, “ERCBench: An Open-Source Benchmark Suite for Embedded and Reconfigurable Computing,” *2010 Int. Conf. F. Program. Log. Appl.*, pp. 408–413, Aug. 2010.

[92] J. Pistorius, M. Hutton, A. Mishchenko, and R. Brayton, “Benchmarking Method and Designs Targeting Logic Synthesis for FPGAs,” in *Proc. of International Workshop on Logic and Synthesis (IWLS)*, 2007.

[93] J. Liang, R. Tessier, and D. Goeckel, “A Dynamically-Reconfigurable, Power-Efficient Turbo Decoder,” in *12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, 2004, pp. 91–100.

[94] J. Noguera and I. O. Kennedy, “Power Reduction in Network Equipment Through Adaptive Partial Reconfiguration,” in *2007 International Conference on Field Programmable Logic and Applications*, 2007, pp. 240–245.

[95] J. H. Anderson and F. N. Najm, “Power estimation techniques for FPGAs,” *Very Large Scale Integr. Syst. IEEE Trans.*, vol. 12, no. 10, pp. 1015–1027, Oct. 2004.

[96] A. Gayasen, K. Lee, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, and T. Tuan, “A Dual-VDD Low Power FPGA Architecture,” in *Field Programmable Logic and Application SE - 17*, vol. 3203, J. Becker, M. Platzner, and S. Vernalde, Eds. Springer Berlin Heidelberg, 2004, pp. 145–157.

[97] M. Litochevski and L. Dongjun, “High throughput and low area AES core,” *OpenCores*, 2010. [Online]. Available: http://opencores.org/project,aes_highthroughput_lowarea.

[98] O. Uzenkov and A. Sergiyenko, “Pipelined FFT/IFFT 256 points processor,” *OpenCores*, 2010. [Online]. Available: http://opencores.org/project,pipelined_fft_256,Overview.

[99] A. de la Piedra, “IEEE 802.15.4 Core (physical layer),” *OpenCores*, 2013. [Online]. Available: http://opencores.org/project,802154phycore.

[100] A. de la Piedra, A. Touhafi, and G. Cornetta, “An IEEE 802.15.4 baseband SoC for tracking applications in the medical environment based on Actel Cortex-M1 soft-core,” in *2010 17th IEEE Symposium on Communications and Vehicular Technology in the Benelux (SCVT2010)*, 2010, pp. 1–5.

[101] O. Girard, “openMSP430,” *OpenCores*, 2009. [Online]. Available: http://opencores.org/project,openmsp430.

[102] A. P. Chandrakasan and R. W. Brodersen, *Low-power CMOS design*. IEEE Press, 1998.

[103] J. M. Rabaey and M. Pedram, *Low Power Design Methodologies*. Kluwer, 1996.

[104] V. George and J. M. Rabaey, *Low-Energy FPGAs — Architecture and Design*. Boston, MA: Springer US, 2001.

[105] H. Hassan and M. Anis, *Low-Power Design of Nanometer FPGAs*. Elsevier, 2010.

[106] I. Kuon, R. Tessier, and J. Rose, “FPGA Architecture: Survey and Challenges,” *Found. Trends® Electron. Des. Autom.*, vol. 2, no. 2, pp. 135–253, 2007.

[107] F. Nekoogar, *From ASICs to SOCs : a practical approach*. Upper Saddle River, NJ: Prentice Hall/Professional Technical Reference, 2003.

[108] A. Raghunathan, N. K. Jha, and S. Dey, *High-Level Power Analysis and Optimization*. Boston, MA: Springer US, 1998.

[109] S. Ahuja, A. Lakshminarayana, and S. K. Shukla, *Low Power Design with High-Level Power Estimation and Power-Aware Synthesis*. New York, NY: Springer New York, 2012.

[110] E. Boemo, J. P. Oliver, and G. Caffarena, "Tracking the pipelining-power rule along the FPGA technical literature," in *Proceedings of the 10th FPGAworld Conference*, 2013, pp. 9:1–9:5.

[111] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *Solid-State Circuits, IEEE J.*, vol. 27, no. 4, pp. 473–484, 1992.

[112] J. Leijten, J. van Meerbergen, and J. Jess, "Analysis and reduction of glitches in synchronous networks," in *Proceedings the European Design and Test Conference. ED&TC 1995*, 1995, pp. 398–403.

[113] E. Boemo, G. González de Rivera, S. Lopez-Buedo, and J. M. Meneses, "Un estudio dobre circuitos segmentados en FPGAs," in *Design of Circuits and Integrated Systems, DCIS*, 1994, pp. 549–554.

[114] E. Boemo, S. Lopez-Buedo, G. González de Rivera, and J. M. Meneses, "On the Usefulness of Pipelining and Wave Pipelining as Low-Power Design Technique," in *PATMOS Fifth Int. Workshop (Power and Timing Modelling for Performance of Integrated Circuits)*, 1995, pp. 252–263.

[115] E. Mussol and J. Cortadella, "Low-Power Array Multipliers with Transition-Retaining Barriers," in *PATMOS Fifth Int. Workshop (Power and Timing Modelling for Performance of Integrated Circuits)*, 1995.

[116] "Pipelining and re-timing techniques and their effect on power dissipation," 2000.

[117] Mouzam Khan (Altera Corporation), "Power Optimization in FPGA Designs," in *Synopsys Users Group (SNUG)*, 2006.

[118] N. H. 1977- Rollins, "Reducing Power in FPGA Designs Through Glitch Reduction," Brigham Young University, 2007.

[119] S. Bard and N. I. Rafla, "Reducing power consumption in FPGAs by pipelining," in *Circuits and Systems, 2008. MWSCAS 2008. 51st Midwest Symposium on*, 2008, pp. 173–176.

[120] J. M. Ramos Meixedo, "Metodologias de projecto de baixo consumo para implementacoes em FPGA," Universidade do Porto, 2008.

[121] M. (Xilinx I. . Klein, “Optimizing Xilinx FPGAs for Power,” *Xcell Journal*.

[122] J. P. Oliver and E. Boemo, “Power estimations vs. power measurements in Cyclone III devices,” in *Programmable Logic (SPL), 2011 VII Southern Conference on*, 2011, pp. 87–90.

[123] “Using Suspend Mode in Spartan-3 Generation FPGAs,” *XAPP480 (v1.0) May 2, 2007.*, 2007. [Online]. Available: http://www.xilinx.com/support/documentation/application_notes/xapp480.pdf.

[124] “Spartan-6 FPGA Power Management User Guide,” *Xilinx UG394 (v1.2) May 16, 2014*. .

[125] L. Benini, P. Siegel, and G. De Micheli, “Saving power by synthesizing gated clocks for sequential circuits,” *IEEE Des. Test Comput.*, vol. 11, no. 4, pp. 32–41, Jan. 1994.

[126] D. Garrett, M. Stan, and A. Dean, “Challenges in clockgating for a low power ASIC methodology,” in *Proceedings of the 1999 international symposium on Low power electronics and design - ISLPED ’99*, 1999, pp. 176–181.

[127] Q. Wu, M. Pedram, and X. Wu, “Clock-gating and its application to low power design of sequential circuits,” in *Proceedings of CICC 97 - Custom Integrated Circuits Conference*, 1997, pp. 479–482.

[128] F. Theeuwen and E. Seelen, “Power Reduction Through Clock Gating by Symbolic Manipulation,” in *VLSI: Integrated Systems on Silicon SE - 32*, R. Reis and L. Claesen, Eds. Springer US, 1997, pp. 389–399.

[129] P. Schoenmakers and J. Theeuwen, “Clock gating on RT-level VHDL,” in *Proc. of the Int. Workshop on Logic Synthesis*, 1998, pp. 387–391.

[130] F. Emnett and M. Biegel, “Power reduction through RTL clock gating,” in *Synopsys Users Group Conference*, 2000.

[131] N. Raghavan, V. Akella, and S. Bakshi, “Automatic insertion of gated clocks at register transfer level,” in *Proceedings Twelfth International Conference on VLSI Design. (Cat. No.PR00013)*, 1999, pp. 48–54.

[132] F. Rivoallon, J. Balasubramanian, and B. F. Rivoallon, “Reducing Switching Power with Intelligent Clock Gating,” *Xilinx White Paper WP370 (v1.4)*. [Online]. Available: http://www.xilinx.com/support/documentation/white_papers/wp370_Intelligent_Clock_Gating.pdf.

[133] G. Sutter, E. Todorovich, S. Lopez-Buedo, and E. Boemo, “FSM Decomposition for Low Power in FPGA,” in *Field-Programmable Logic and*

Applications: Reconfigurable Computing Is Going Mainstream SE - 37, vol. 2438, M. Glesner, P. Zipf, and M. Renovell, Eds. Springer Berlin Heidelberg, 2002, pp. 350–359.

[134] O. Cadenas and G. Megson, "Power performance with gated clocks of a pipelined Cordic core," in *2003 5th International Conference on ASIC Proceedings (IEEE Cat No 03TH8690) ICASIC-03*, 2003, vol. 2, pp. 1226–1230 Vol.2.

[135] G. Sutter, "Aportes a la Reducción de Consumo en FPGAs," Universidad Autónoma de Madrid, 2005.

[136] G. Sutter and E. Boemo, "Experiments in low power FPGA design," *Lat. Am. Appl. Res.*, vol. 37, pp. 99–104, 2007.

[137] Y. Zhang, J. Roivainen, and A. Mammela, "Clock-Gating in FPGAs: A Novel and Comparative Evaluation," in *9th EUROMICRO Conference on Digital System Design (DSD'06)*, 2006, pp. 584–590.

[138] S. Huda, M. Mallick, and J. H. Anderson, "Clock gating architectures for FPGA power reduction," in *2009 International Conference on Field Programmable Logic and Applications*, 2009, pp. 112–118.

[139] L. Sterpone, L. Carro, D. Matos, S. Wong, and F. Fakhar, "A new reconfigurable clock-gating technique for low power SRAM-based FPGAs," in *2011 Design, Automation & Test in Europe*, 2011, pp. 1–6.

[140] "Clock Control Block (ALTCLKCTRL) Megafunction User Guide," *Altera Corporation*. [Online]. Available: http://www.altera.com/literature/ug/ug_altclock.pdf.

[141] "Spartan-6 FPGA Clocking Resources User Guide," *Xilinx Inc. UG382*. [Online]. Available: http://www.xilinx.com/support/documentation/user_guides/ug382.pdf.

[142] "Cyclone III Device Handbook, Volume 1, Chapter 2, Logic Elements and Logic Array Blocks in the Cyclone III Device Family," *Altera Corporation*, 2012. [Online]. Available: http://www.altera.com/literature/hb/cyc3/cyc3_ciii51002.pdf.

[143] K. Chapman, "Practical Power Testing. A Reference Design for the SpartanTM-3A FPGA Starter Kit," *Xilinx*, 2008. [Online]. Available: http://www.xilinx.com/products/boards/s3astarter/files/s3ask_power_testing.pdf.

[144] L. Nogueira, M. Rodríguez, and G. Barreto, "Diseño de bajo consumo en FPGAs," Universidad de la República, Uruguay, 2012.

[145] G. J. García, C. A. Jara, J. Pomares, A. Alabdo, L. M. Poggi, and F. Torres, “A survey on FPGA-based sensor systems: towards intelligent and reconfigurable low-power sensors for computer vision, control and signal processing.,” *Sensors (Basel).*, vol. 14, no. 4, pp. 6247–78, Jan. 2014.

[146] A. de la Piedra, A. Braeken, and A. Touhafi, “Sensor Systems Based on FPGAs and Their Applications: A Survey,” *Sensors*, vol. 12, no. 12, pp. 12235–12264, Sep. 2012.

[147] J. Portilla, A. de Castro, E. de la Torre, and T. Riesgo, “A Modular Architecture for Nodes in Wireless Sensor Networks,” *J. Univers. Comput. Sci.*, vol. 12, no. 3, pp. 328–339, Mar. 2006.

[148] J. Valverde, A. Otero, M. Lopez, J. Portilla, E. de la Torre, and T. Riesgo, “Using SRAM based FPGAs for power-aware high performance wireless sensor networks.,” *Sensors (Basel).*, vol. 12, no. 3, pp. 2667–92, Jan. 2012.

[149] M. Lombardo, J. Camarero, J. Valverde, J. Portilla, E. de la Torre, and T. Riesgo, “Power management techniques in an FPGA-based WSN node for high performance applications,” in *7th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*, 2012, pp. 1–8.

[150] Y. E. Krasteva, J. Portilla, E. de la Torre, and T. Riesgo, “Embedded Runtime Reconfigurable Nodes for Wireless Sensor Networks Applications,” *IEEE Sens. J.*, vol. 11, no. 9, pp. 1800–1810, Sep. 2011.

[151] M. Hübner, J. Meyer, O. Sander, L. Braun, J. Becker, J. Noguera, and R. Stewart, “Fast Sequential FPGA Startup Based on Partial and Dynamic Reconfiguration,” in *2010 IEEE Computer Society Annual Symposium on VLSI*, 2010, pp. 190–194.

[152] J. Meyer, J. Noguera, M. Hübner, L. Braun, O. Sander, R. M. Gil, R. Stewart, and J. Becker, “Fast Start-up for Spartan-6 FPGAs using Dynamic Partial Reconfiguration,” in *2011 Design, Automation & Test in Europe*, 2011, pp. 1–6.

[153] A. A. M. Bsoul and S. J. E. Wilton, “An FPGA with power-gated switch blocks,” in *Field-Programmable Technology (FPT), 2012 International Conference on*, 2012, pp. 87–94.

[154] A. A. M. Bsoul and S. J. E. Wilton, “An FPGA Architecture Supporting Dynamically Controlled Power Gating,” in *International Conference on Field-Programmable Technology (FPT’10)*, 2010.

[155] A. Rahman, S. Das, T. Tuan, and S. Trimberger, “Determination of Power Gating Granularity for FPGA Fabric,” in *Custom Integrated Circuits Conference, 2006. CICC ’06. IEEE*, 2006, pp. 9–12.

[156] S. Ishihara, M. Hariyama, and M. Kameyama, “A Low-Power FPGA Based on Autonomous Fine-Grain Power Gating,” *Very Large Scale Integr. Syst. IEEE Trans.*, vol. 19, no. 8, pp. 1394–1406, 2011.

[157] F. Li, Y. Lin, L. He, and J. Cong, “Low-power FPGA using pre-defined dual-Vdd/dual-Vt fabrics,” *Proceeding 2004 ACM/SIGDA 12th Int. Symp. F. Program. gate arrays - FPGA ’04*, p. 42, 2004.

[158] C. Q. TRAN, H. KAWAGUCHI, and T. SAKURAI, “Low-Power Low-Leakage FPGA Design Using Zigzag Power Gating, Dual-VTH/VDD and Micro-VDD-Hopping,” *IEICE Trans. Electron.*, vol. E89–C, no. 3, pp. 280–286, Mar. 2006.

[159] C. Q. Tran, H. Kawaguchi, and T. Sakurai, “95% Leakage-Reduced FPGA using Zigzag Power-gating, Dual-VTH/VDD and Micro-VDD-Hopping,” in *Asian Solid-State Circuits Conference, 2005*, 2005, pp. 149–152.

[160] T. Marconi, K. Bertels, and G. Gaydadjiev, “A Novel Logic Element for Power Reduction in FPDs,” Delft, 2010.

[161] J. J. H. Anderson and F. F. N. Najm, “A novel low-power FPGA routing switch,” in *Proceedings of the IEEE 2004 Custom Integrated Circuits Conference*, 2004, vol. 2, pp. 719–722.

[162] J. H. Anderson and F. N. Najm, “Low-power programmable routing circuitry for FPGAs,” in *IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004.*, pp. 602–609.

[163] J. H. Anderson and F. N. Najm, “Low-Power Programmable FPGA Routing Circuitry,” *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 17, no. 8, pp. 1048–1060, Aug. 2009.

[164] C. Q. Tran, H. Kawaguchi, and T. Sakurai, “More than two orders of magnitude leakage current reduction in look-up table for FPGAs,” in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, 2005, pp. 4701–4704 Vol. 5.

[165] T. Tuan, A. Rahman, S. Das, S. Trimberger, and S. Kao, “A 90-nm Low-Power FPGA for Battery-Powered Applications,” *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 26, no. 2, pp. 296–300, Feb. 2007.

[166] S. Huda, J. Anderson, and H. Tamura, “Charge recycling for power reduction in FPGA interconnect,” in *2013 23rd International Conference on Field programmable Logic and Applications*, 2013, pp. 1–8.

[167] R. Sankaranarayanan and M. R. Guthaus, “A single-VDD ultra-low energy sub-threshold FPGA,” in *2012 IEEE/IFIP 20th International Conference on VLSI and System-on-Chip (VLSI-SoC)*, 2012, pp. 219–224.

[168] B. Zimmer, “The Hidden History of ‘Glitch,’” 2013. [Online]. Available: http://www.visualthesaurus.com/cm/wordroutes/the-hidden-history-of-glitch/. [Accessed: 07-Dec-2013].