UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

FACULTAD DE INGENIERÍA

Diseño e implementación de la unidad de control de un neuromodulador implantable basado en FPGA

Santiago Leandro Martínez Bentancor

Programa de Posgrado en Ingeniería Eléctrica
Facultad de Ingeniería
Universidad de la República

Montevideo – Uruguay
Noviembre de 2018

UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

FACULTAD DE INGENIERÍA

Diseño e implementación de la unidad de control de un neuromodulador implantable basado en FPGA

Santiago Leandro Martínez Bentancor

Tesis de Maestría presentada al Programa de Posgrado en Ingeniería Eléctrica, Facultad de Ingeniería de la Universidad de la República, como parte de los requisitos necesarios para la obtención del título de Magíster en Ingeniería Eléctrica.

Director:
Ph.D. Prof. Juan Pablo Oliver

Director académico:
Ph.D. Prof. Juan Pablo Oliver

Montevideo – Uruguay
Noviembre de 2018

Martínez Bentancor, Santiago Leandro

Diseño e implementación de la unidad de control de un neuromodulador implantable basado en FPGA / Santiago Leandro Martínez Bentancor. - Montevideo: Universidad de la República, Facultad de Ingeniería, 2018.

XI, 79 p. 29, 7cm.

Director:

Juan Pablo Oliver

Director académico:

Juan Pablo Oliver

Tesis de Maestría – Universidad de la República, Programa en Ingeniería Eléctrica, 2018.

Referencias bibliográficas: p. 76 – 79.

1. FPGA, 2. Neuromodulación, 3. Consumo de potencia. I. Oliver, Juan Pablo, . II. Universidad de la República, Programa de Posgrado en Ingeniería Eléctrica. III. Título.

INTEGRANTES DEL TRIBUNAL DE DEFENSA DE TESIS

Ph.D. Prof. Eduardo Boemo

Ph.D. Prof. Conrado Rossi

Ph.D. Prof. Julián Oreggioni

Montevideo – Uruguay
Noviembre de 2018

Agradecimientos

Quisiera agradecer a Juan Pablo Oliver por haber aceptado participar en este trabajo, por su ayuda, sus comentarios y su gran disposición, desde Uruguay y desde España. También quisiera agradecer a Eduardo Boemo, Conrado Rossi y Julián Oreggioni por participar en la evaluación de la tesis y así poder hacer de esta un mejor trabajo.

A Pablo Monzón y Alvaro Giusto, que además de ser mis compañeros de Sistemas Lineales 1 y Sistemas Lineales 2, me han dado el espacio dentro del Departamento de Sistemas y Control para realizar de esta maestría. También quiero agradecer a Andrés Alcarraz, que además de un gran compañero de clases siempre ha sido un buen amigo.

No quiero dejar de mencionar al Instituto de Ingeniería Eléctrica de la Facultad de Ingeniería de la Universidad de la República por dar la posibilidad de estudiar en un entorno de gran nivel y en forma gratuita.

Quisiera agradecer a la empresa CCC del Uruguay por darme la posibilidad de practicar en forma activa la profesión que he elegido y poder aprender acerca de los dispositivos médicos implantables y activos. Particularmente quiero agradecer a Federico de Mula que ha revisado este documento y dado muy buenas recomendaciones, a Cristina Cornes que me ha supervisado en estudios dirigidos relativos a la ciberseguridad de sistemas médicos y a Diego Gurri que me ha supervisado en tareas que involucran el ejercicio profesional y siempre ha sido muy considerado respecto a mi trabajo en facultad.

Quiero agradecer a mi familia que siendo ajena a los detalles técnicos en los cuales uno quiere profundizar, siempre me han dado su apoyo. A Stephanía, por darme su visión técnica y estética de este trabajo, por su apoyo continuo y mucho más.

A mis amigos de Santa Rosa: Estefan, Diego, Pepe, Pablo, Fernando y Marcelo que los veo menos de lo que desearía pero siempre los tengo muy presentes.

RESUMEN

En esta tesis se presenta el diseño de la unidad de control de un circuito de neuromodulación y su implementación basada en FPGA. El foco fue puesto, esencialmente, en el consumo de potencia, variable que resulta crítica en los sistemas médicos implantables y activos.

Se implementó el diseño en tres FPGA, de diferente familia y fabricante: Cyclone V (Intel), IGLOO2 (Microsemi) e iCE40 (Lattice). Se midió el consumo de potencia en cada plataforma, obteniéndose los mejores resultados (tanto estáticos como dinámicos) en la FPGA iCE40-HX-8K con un consumo total del *core* de 3,6mW. Los resultados obtenidos fueron contrastados contra los datos de consumo obtenidos de las hojas de datos y de las herramientas de estimación, obteniéndose errores cercanos al 400 % para algunos casos.

Se determinaron los requerimientos mínimos en términos de cantidad de pines y tamaño para cada familia utilizada y se pudo calcular el menor consumo alcanzable para cada una de ellas.

Finalmente, se propusieron y evaluaron tres métodos sencillos para la medida del consumo dinámico en escenarios de gran consumo estático, obteniéndose resultados con un error cercano al 10 %.

Palabras claves:
FPGA, Neuromodulación, Consumo de potencia.

ABSTRACT

This thesis presents the design of a FPGA based control unit for a neuromodulation circuit. The work was focused on power consumption since this is one of the most critical variables addressed by active implantable medical devices designers.

The design was implemented in three different FPGA, from different families and manufacturers: Cyclone V (Intel), IGLOO2 (Microsemi) e iCE40 (Lattice). Power consumption was measured for each platform; the best results for both static and dynamic power were achieved with the iCE40-HX-8K FPGA, with 3,6mW of total core power consumption.

The results measured were compared to those provided both by the power estimator tools and by the devices datasheets. The differences were close to 400 % for some cases.

For each family, the minimum requirements in terms of FPGA size and pin quantity were determined as well as the lowest achievable power consumption.

Finally, three simple methods were proposed in order to measure dynamic power consumption in high static power consumption scenarios. For all methods, the results show errors close to 10 %.

Keywords:
FPGA, Neuromodulation, Power consumption.

Lista de siglas

Lista de siglas

4-LUT *look up table* de 4 entradas 12

AIMD Dispositivos Médicos Implantables y Activos 2

ALM *Adaptive Logic Module 12*

ASIC *Application-specific integrated circuit 9*

CMOS *MOS complementario 12*

CU *unidad de control 6*

DAC *Digital to Analog Converter 34*

DBS *deep brain stimulator 1*

FF *flip-flop 12*

FLASH-FPGA *FPGA basadas en memoria FLASH 14*

FPGA *Field Programmable Gate Array 9*

L *elementos lógicos 12*

LFSR *Linear Feedback Shift Register 28*

R *conexiones programables 13*

S *enrutadores programables 13*

SCS *spinal cord stimulator 1*

SRAM-FPGA *FPGA basadas en memoria RAM estática 14*

Tabla de contenidos

Capítulo 1

Introducción

1.1. Neuromodulación

La idea básica de la neuroestimulación es producir un cambio en la actividad del sistema nervioso en forma controlada con el fin de tratar cierta patología. Un tipo de neuroestimulador son los neuromoduladores, que utilizan campo electromagnético para conseguir un efecto terapéutico. Dentro de la áreas destacadas de la neuromodulación se encuentran los estimuladores de la médula espinal (*spinal cord stimulator* (SCS)) y los estimuladores del cerebro profundo (*deep brain stimulator* (DBS)).

Los SCS son un tipo de neuroestimulador implantable que producen campo electromagnético en zonas de la médula espinal con el fin de tratar cierto tipo de dolores, típicamente, de espalda y de piernas. Este tipo de técnicas han sido implementadas desde que en 1965 Ronald Melzack y Patrick Wall introdujeran un modelo para la transmisión de señales nerviosas denominado *gate control theory* [1]. Básicamente, el modelo propone que las fibras nerviosas más finas (encargadas de la transmisión de señales "dolorosas") y las fibras más gruesas (encargadas de la transmisión de señales relacionadas, por ejemplo, al contacto y la presión) van a parar a ciertas zonas del cuerno dorsal de la médula. Si las señales de las fibras más gruesas tienen una actividad mayor que las finas, se produce un efecto de inhibición de las últimas que se traducirá en una reducción en la sensación de dolor, dándose así un efecto anestésico. Con base en esto, los SCS tradicionales envían impulsos eléctricos al tejido a través de un arreglo de electrodos epidural, ubicado sobre aquellas vértebras vinculadas a la sensación de dolor. Su terapia produce un enmascaramiento del dolor conjuntamente a

un efecto de hormigueo y/o adormecimiento denominado parestesia [2].

Los DBS en cambio, producen campo electromagnético en zonas específicas del cerebro, cambiando la actividad cerebral en forma controlada. Los DBS se utilizan en el tratamiento de la enfermedad de Parkinson [3], temblor esencial (*essential tremor* [4]), entre otros. Si bien la forma precisa en la que los DBS operan no es aún clara, su campo de aplicación se encuentra en crecimiento [5].

1.2. Forma de onda de un neuromodulador

A rasgos generales, un estimulador de este tipo es capaz de entregar y/o absorber carga hacia o desde el tejido, a través de un arreglo de electrodos. Los electrodos pueden utilizarse para inyectar carga (*source* o "+"), absorberla (*sink* o "−") o mantenerse desconectados ("×"). El proceso de inyección/absorción de carga se realiza en forma periódica, tal y como se observa en la figura 1.1.

En la primera etapa de estimulación, el estimulador inyectará carga a corriente constante que fluirá desde los electrodos "+" hacia los electrodos "−" a través del tejido. Esta fase se denomina "fase de estímulo" y el intervalo de tiempo que esta toma se denomina ancho de pulso (w). Esta es la fase del proceso de estimulación que tiene efecto terapéutico. La segunda etapa es la de balance activo de carga, en la cual toda la carga entregada en la primer etapa será absorbida. Esto se consigue haciendo circular la misma cantidad de carga entregada pero desde los electrodos "−" hacia los "+". Idealmente, este balance activo de carga asegura que la circulación de cargas promedio hacia o desde el tejido sea nula. La presencia de corrientes promedio que circulan hacia o desde el tejido pueden resultar dañinas, por lo cual los estándares internacionales las limitan a valores muy pequeños, del orden de 1μA para los Dispositivos Médicos Implantables y Activos (AIMD) en general y $0{,}1\mu$A para los diseñados para el tratamiento de bradicardias [6], como los marcapasos ([7][8]).

En resumen, los parámetros de la forma de onda son:

- Ancho de pulso (w): como fue descrito anteriormente es el tiempo que se utiliza para la inyección y extracción de carga.

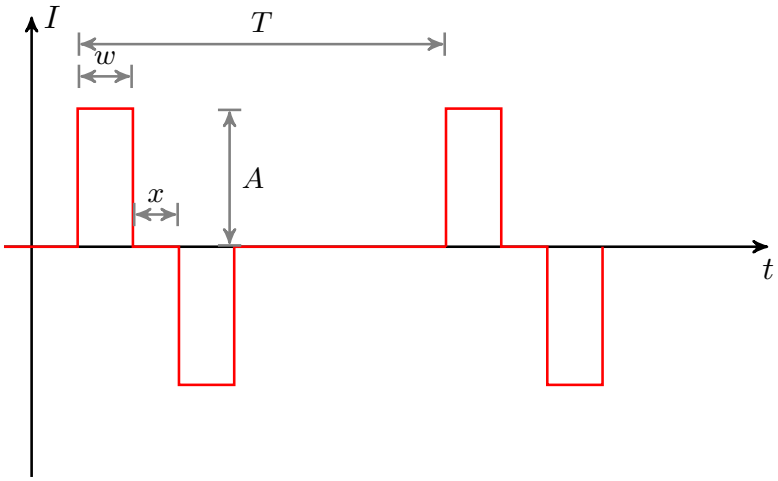

Figura 1.1: Forma de onda de la corriente entregada al tejido. La inyección y absorción de carga se realiza a corriente constante durante un tiempo w (ancho de pulso) y con una amplitud A. Entre el pulso de inyección y el de absorción (balance activo) de carga, hay un tiempo x denominado "interpulso" donde no hay flujo de cargas ni desde ni hacia el estimulador. Este proceso se repite periódicamente, con período T.

- Amplitud (A): el valor de corriente tanto en la etapa de estimulación como en la de balance activo.
- "Interpulso" (x): tiempo entre el final de la fase de estímulo y comienzo de la fase de balance.
- Período (T): tiempo entre dos pulsos de estimulación consecutivos.

Adicionalmente, múltiples formas de onda pueden ejecutarse en paralelo, cada una con sus propios parámetros (ancho de pulso, período y amplitud) y a través de un grupo de electrodos igual o diferente al de las otras. Un escenario con más de un terapia ejecutándose en forma simultánea, puede observarse esquemáticamente en la figura 1.2.

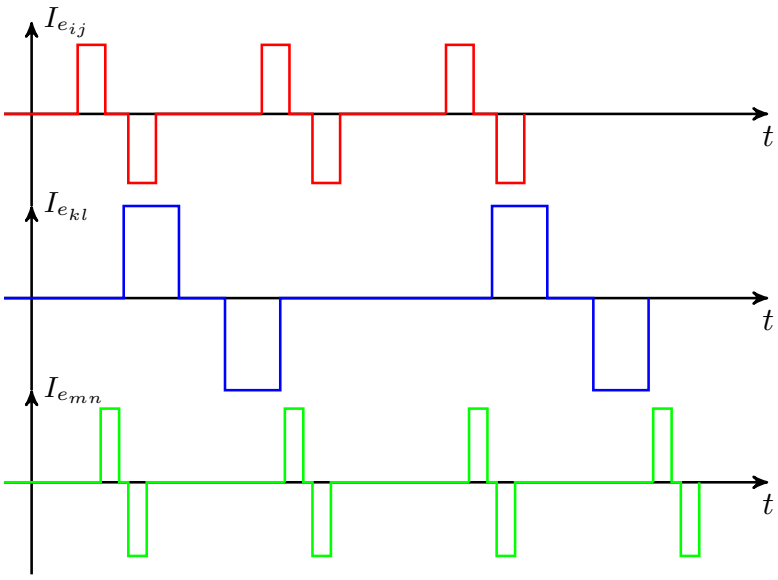

Figura 1.2: Tres terapias ejecutándose en forma simultánea. Cada una tiene su propia amplitud, ancho de pulso y período. Adicionalmente, cada una utiliza grupos de electrodos e_{ij}, e_{kl}, e_{mn} no necesariamente coincidentes ni necesariamente disjuntos.

1.3. Circuito generador de pulsos

En la figura 1.3 se presenta un circuito capaz de producir estímulos con la forma de onda descrita en la sección 1.2. El circuito está conformado por una fuente de corriente programable, un arreglo de llaves, una fuente de tensión programable y un juego de condensadores.

La fuente de corriente es la encargada de la inyección y absorción de carga hacia y desde el tejido, a través de los electrodos de salida. Las llaves programables seleccionan los electrodos por los cuales cada terapia estimulará al tejido: si el electrodo e_k es configurado como *current source* ("+"), durante la fase de estímulo la llave $S^+_{e_k}$ estará cerrada mientras que $S^-_{e_k}$ se mantendrá abierta. En la fase de balance la llave $S^+_{e_k}$ estará abierta mientras que $S^-_{e_k}$ se mantendrá cerrada. Para cada electrodo, el manejo de las llaves puede resumirse en la tabla 1.1.

Los condensadores C_i son bloqueadores de continua, que aseguran que la corriente promedio hacia el tejido sea nula ante un caso de falla.

La fuente de corriente se alimenta desde la fuente de tensión programable. Su programabilidad permite ajustar su tensión al valor mínimo necesario para poder estimular apropiadamente (es decir, que la polarización de la fuente

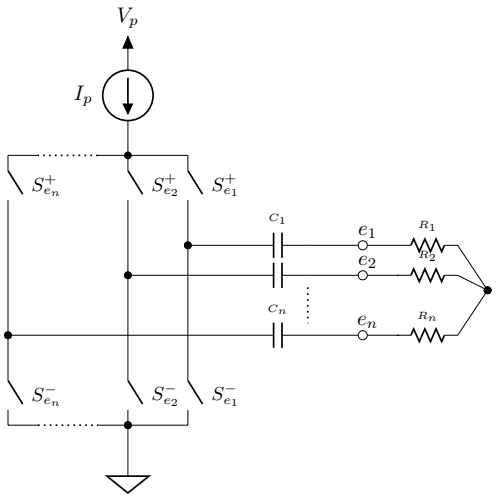

Figura 1.3: Circuito neuroestimulador. La fuente de corriente programable I_p es la encargada de la inyección como de la absorción de carga. Las llaves $S^+_{e_i}$ y $S^-_{e_i}$ $(i = 1..n)$ son las encargadas de dirigir el flujo de carga desde el circuito al tejido y desde el tejido hacia el circuito a través de los electrodos e_i $(i = 1..n)$. Las llaves $S^+_{e_k}$ y $S^-_{e_k}$ están asociadas al electrodo e_k de modo de configurarlo como *current source* ($S^+_{e_k}$ cerrada y $S^-_{e_k}$ abierta), como *current sink* ($S^+_{e_k}$ abierta y $S^-_{e_k}$ cerrada) o desconectado ($S^+_{e_k}$ abierta y $S^-_{e_k}$ abierta). El *front-end* del circuito está conformado por condensadores serie (C_i, $(i = 1..n)$) que bloquean la circulación de corrientes continuas. La red de resistencias en estrella (con resistencias R_i, $(i = 1..n)$) es un modelo resistivo de la interfaz electrodo-tejido.

	Fase de estímulo		Fase de balance	
	$S^+_{e_k}$	$S^-_{e_k}$	$S^+_{e_k}$	$S^-_{e_k}$
$e_k =$ "+"	Cerrada	Abierta	Abierta	Cerrada
$e_k =$ "−"	Abierta	Cerrada	Cerrada	Abierta
$e_k =$ "×"	abierta			

Tabla 1.1: Manejo de las llaves según la configuración del electrodo y la etapa del proceso de estimulación. De este modo, en la fase de estímulo la corriente fluye desde los electrodos positivos ("+") hacia el tejido y retorna por los electrodos negativos ("−"). Durante la fase de balance la corriente fluye en sentido opuesto, desde los electrodos negativos hacia los positivos.

de corriente sea aceptable), reduciendo así el consumo del circuito (un breve análisis del consumo del circuito se encuentra en la sección 1.3.1).

Debe notarse que en esta implementación toda la corriente deberá ser generada por una única fuente. Debido a esto no será posible generar dos formas de onda simultáneamente. En escenarios como este podría adoptarse un criterio de asignación de prioridades a las terapias, por ejemplo, priorizando a las "más

lentas" (las de mayor período). La figura 1.4 muestra un ejemplo de esto.

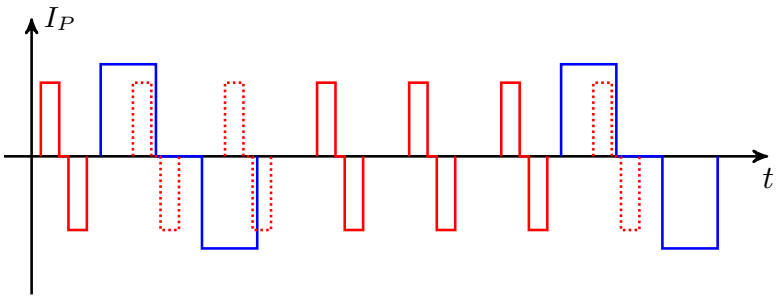

Figura 1.4: Forma de onda de la fuente de corriente programable cuando se estimula con dos terapias simultáneas. La terapia más lenta (de mayor período) tiene prioridad sobre la rápida. (Las llaves selectoras también son configuradas según este criterio, respetando la configuración de electrodos para cada terapia).

El sistema completo constará del circuito descrito anteriormente y una unidad de control (CU). Esta última está encargada de la programación de las fuentes de tensión y corriente así como de la conexión y desconexión de las llaves, respetando las amplitudes y tiempos deseados (ver figura 1.5).

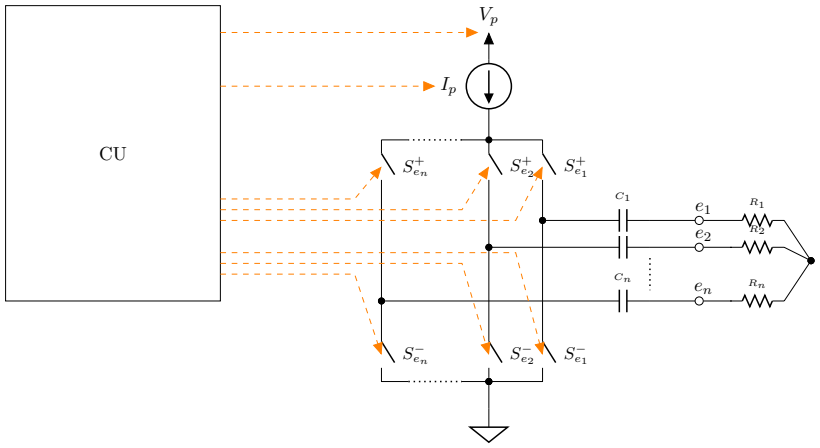

Figura 1.5: Circuito de estimulación completo. La unidad de control, *CU*, maneja la programación y señales de control de las fuentes y las llaves para conseguir la forma de onda deseada.

En el caso particular de esta tesis, los anchos de pulso tomarán valores del orden de las decenas de micro-segundos, los períodos del orden de las centenas de micro-segundos hasta las centenas de mili-segundos, las amplitudes del orden de los mili-amperios y las tensiones del orden de los voltios.

1.3.1. Consumo total del circuito de estimulación

La potencia total que consume el circuito, P_T, se puede expresar como:

$$P_T = P_V + P_{CU}$$

donde P_V es la potencia que entrega la fuente de tensión al circuito de estimulación (debido a que todos los elementos del circuito se alimentan de ella) y P_{CU} es la potencia consumida por la unidad de control. La fuente de tensión provee al circuito de estimulación de la energía necesaria para funcionar. De este modo, la potencia P_V se puede expresar como:

$$P_V = P_I + P_C + P_{SW} + P_o$$

donde P_I es la potencia consumida por la fuente de corriente, P_C es la potencia consumida por los condensadores de salida, P_{SW} es la potencia consumida por las llaves y P_o es la potencia que efectivamente se entrega al tejido.

En el caso de una sola terapia activa, la potencia promedio consumida por el circuito de estimulación ($\overline{P_V}$) no depende de los condensadores de desacople de continua, debido al balance activo. Asumiendo que la potencia consumida por los interruptores P_{SW} es despreciable:

$$\overline{P_V} \approx \overline{P_I} + \overline{P_o}$$

La potencia promedio entregada al tejido dependerá de los parámetros de la forma de onda configurada, del siguiente modo:

$$\overline{P_o} = 2A^2 R_o \frac{w}{T}$$

siendo R_o la resistencia vista desde los electrodos encargados de administrar la terapia.

Respecto a la fuente de corriente, si se denomina V_{min} a la tensión mínima entre sus bornes que asegura una adecuada polarización de la misma, el consumo promedio de la fuente de corriente estará acotado inferiormente por:

$$\overline{P_I} \geq 2V_{min} A \frac{w}{T}$$

En el caso general (con hasta n terapias activas en forma simultánea) los

consumos promedios pueden escribirse como:

$$\overline{P_o} \leq \sum_{i=1}^{n} 2\delta_i \alpha_i A_i^2 R_{oi} \frac{w_i}{T_M}$$

donde T_M es el mínimo común múltiplo de los períodos de las n terapias, α_i es la cantidad de períodos de la i-ésima terapia que son capaces de entregarse en el tiempo T_M, w_i y A_i son el ancho de pulso y la corriente de estimualción de la i-ésima terapia y δ_i es cero o uno dependiendo si la terapia i está activa ($\delta_i = 1$) o no ($\delta_i = 0$). La cota superior se debe a que, como fue comentado anteriormente, sólo sería posible entregar una terapia a la vez. Esto redunda en que la cantidad de estímulos asociados al área i que se entregan pueden ser menores que α_i. Si la fuente de tensión se ajusta para cada estímulo a su mínimo valor, el consumo de la fuente de corriente puede estimarse como:

$$\overline{P_I} \approx \sum_{i=1}^{n} 2\delta_i \alpha_i V_{min} A_i \frac{w_i}{T_M}$$

donde no sería posible determinar una cota precisa. De todas maneras, para el consumo total del circuito de estimulación puede estimarse una cota inferior:

$$\overline{P_V} \geq \sum_{i=1}^{n} 2\delta_i \alpha_i V_i A_i \frac{w_i}{T_M}$$

donde V_i es el valor de la fuente de tensión cuando la corriente A_i es entregada al tejido.

El consumo de la unidad de control P_{CU} no es tan sencillo de estimar. Este dependerá tanto de la plataforma elegida como de la implementación y si bien se encarga del control de las amplitudes y el manejo las llaves, podría poseer un consumo no despreciable.

En el contexto de los dispositivos médicos implantables y activos, el bajo consumo es una característica deseable. Un diseño con un bajo consumo de potencia permite, por ejemplo:

- Si el dispositivo es recargable, maximizar el tiempo entre cargas del implantable, mejorando la experiencia del paciente.
- Si el dispositivo es no-recargable, maximizar la vida útil del implantable, maximizando el tiempo antes que el dispositivo deba ser explantado (para

implantar uno nuevo), evitándole al paciente el estrés de la operación por el mayor tiempo posible.

- Reducir el tamaño de la batería al mínimo (que permita cumplir con los requerimientos de tiempo entre cargas ó vida útil) y por lo tanto reducir el tamaño del implantable.

Una posible plataforma para implementar la unidad de control, puede ser un microcontrolador. Los modos de bajo consumo de los microcontroladores actuales permiten reducir su consumo de potencia a valores muy bajos cuando no necesitan realizar ninguna tarea. En este caso particular, el tiempo en el que un microcontrolador podría estar en un modo de bajo consumo, dependerá de la frecuencia de estimulación. Si la frecuencia de estimulación es relativamente baja, un microcontrolador podría estar más tiempo en modo de bajo consumo. Si la frecuencia de estimulación fuera relativamente alta, los tiempos en bajo consumo se reducirían (al punto de anularse para frecuencias muy altas).

Otra posible plataforma son las *Field Programmable Gate Array* (FPGA). A grandes rasgos, las FPGA consisten en un conjunto relativamente alto de compuertas lógicas y unidades de memoria, que pueden ser interconectadas entre sí para formar funciones lógicas y máquinas de estado complejas. Suelen utilizarse como un paso previo a la utilización de los *Application-specific integrated circuit* (ASIC) gracias a que pueden ser reconfiguradas, pudiendo depurar los diseños. Adicionalmente, cuando las cantidades a producir son relativamente bajas, las FPGA suelen ser más convenientes que los ASIC, por motivos económicos.

Las FPGA se encuentran en varias capacidades (cantidad de elementos lógicos que pueden ser utilizados), varias tecnologías y de varios fabricantes.

1.4. Trabajo y antecedentes

En esta tesis se presenta un diseño para la unidad de control y su implementación, tomando como plataforma las FPGA. El énfasis de la tesis fue puesto en el consumo de potencia, con el fin de aportar información relevante acerca del uso de las FPGA en aplicaciones de bajo consumo.

Presumiendo que el diseño requeriría una baja cantidad de elementos lógicos, el estudio fue realizado considerando al consumo estático de las FPGA como uno de los términos con más peso en el consumo de potencia. En esta

línea, se evaluaron FPGA de diferentes tecnologías y fabricantes (Cyclone V de Intel [9], iCE40 de Lattice [10] e IGLOO2 de Microsemi [11]). Todas las FPGA utilizadas fueron adquiridas en *kits* de desarrollo, por lo cual los recursos disponibles en cada *kit* son considerablemente mayores a los necesarios. La verificación funcional del diseño fue hecha con un prototipo provisto por la empresa CCC del Uruguay [12].

1.4.1. Trabajos previos

Respecto a los antecedentes, hasta la fecha no han sido encontradas más que unas pocas publicaciones que hacen uso de las FPGA en dispositivos implantables.

En [13] se presenta el diseño de un neuroestimulador para recuperar la función renal en pacientes con parálisis. Para su implementación se utilizó una FPGA A1020B de Microsemi. Se dice que fue probado en perros y pasó ciertas pruebas, lamentablemente no se dicen cuáles y no hay ensayos de consumo.

En [14] se presenta la unidad de control de un circuito implantable que es capaz de monitorear parámetros fisiológicos. Previo a la implementación en un ASIC, el diseño se realizó en una FPGA Virtex-7 de Xilinix. Su consumo dinámico fue de $163mW$ y su consumo estático de $136mW$. Lamentablemente de estos ensayos se desconoce la temperatura. Simulaciones de su diseño en un ASIC con tecnología CMOS de 130nm resultaron en un consumo dinámico de $14{,}5mW$.

En [15] se presenta un filtro para aplicaciones Cerebro-Máquina. El mismo es basado en una FPGA IGLOO nano (AGLN250) de Microsemi basada en FLASH. El consumo del *core* de la FPGA se encontró en un rango entre $4{,}68mW$ a $4{,}92mW$ para sus tres modos de operación. No se encuentra presente información acerca de la temperatura de los ensayos.

Finalmente, en [16] se presenta el diseño de un circuito implantable capaz de recolectar información cerebral. El mismo fue implementado en una FPGA IGLOO nano (AGLN250) de Microsemi y es capaz de recolectar datos por 32 canales a una tasa de 25kHz. El consumo de la implementación fue de $5{,}19mW$. La temperatura del ensayo no fue reportada.

1.4.2. Descripción de los próximos capítulos

Las FPGA, como sistemas digitales, poseen ciertas características cuyas principales se presentan en la sección 2.1. Un análisis de las tecnologías utilizadas (SRAM y FLASH) respecto a su implementación y su consumo asociado, son presentados en los capítulos 2 y 3.

Evidenciando el hecho que el consumo requerido para la comunicación entrada/salida de la FPGA es independiente de la plataforma (ver sección 3.3), el foco de esta tesis estará puesto esencialmente en el consumo del *core* de cada FPGA utilizada.

En el capítulo 4 se presenta el diseño completo de la unidad de control y su verificación, tanto a nivel lógico como funcional.

En el capítulo 5 se presentan los resultados obtenidos para el consumo dinámico, estático y total para cada plataforma y la estimación realizada mediante las herramientas de estimación provistas por cada fabricante. Adicionalmente, en este capítulo se determina la mejor FPGA en términos de consumo para cada familia utilizada, gracias a los ensayos realizados.

En el capítulo 6 se evalúan algunos métodos para mejorar la medida de consumo dinámico de potencia.

Finalmente, en el capítulo 7 se presentan las conclusiones a las que se arribó y se especifica una plataforma óptima en términos de consumo para esta implementación.

Capítulo 2

Tecnología

2.1. Arquitectura de las FPGA

Como fue comentado en la sección 1.3.1, el diseño de la unidad de control para el circuito estimulador estuvo basada en FPGA. Las siguientes secciones del presente capítulo tratarán de introducir brevemente a las FPGA como plataforma de diseño, sus posibles tecnologías de fabricación y su estructura interna.

2.1.1. Estructura interna de las FPGA

Las FPGA son sistemas digitales basados en tecnología MOS complementario (CMOS) que poseen, como una de sus principales características, un conjunto relativamente grande de elementos configurables e interconectables. En la figura 2.1 se muestra una versión simplificada de la estructura interna de las FPGA. Los elementos lógicos (L) contienen la lógica programable que puede ser utilizada para la generación de un diseño. La arquitectura más común para los elementos lógicos consiste en una *look up table* de 4 entradas (4-LUT) y un *flip-flop* (FF) (ver figura 2.2). La arquitectura 4-LUT + FF resulta en una de las mejores relaciones funcionalidad-área, para un rango muy amplio de tecnologías de fabricación [17].

Existen otro tipo de arquitecturas para los elementos lógicos, una de ellas son los *Adaptive Logic Module* (ALM) (ver figura 2.3). Son los bloques utilizados por las FPGA de la familia Cyclone V de Intel (la cual fue utilizada en este trabajo). Estos bloques son más complejos, con 4 *flip flops* y una 8-LUT que puede "fraccionarse" de forma de concentrar la mayor cantidad de

funcionalidad por bloque.

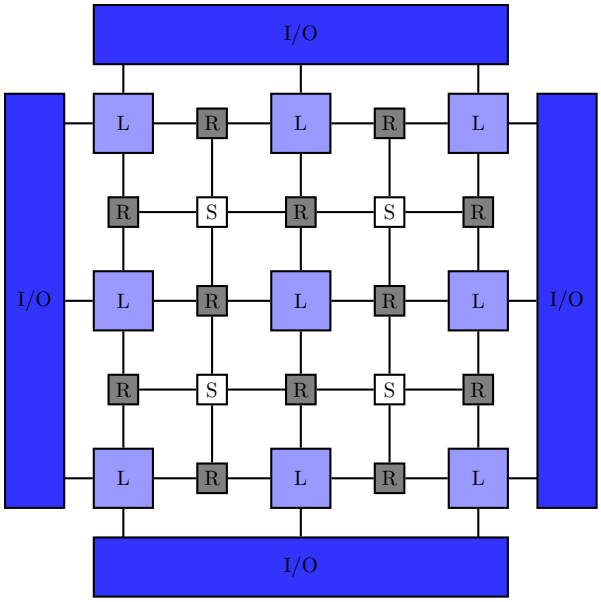

Figura 2.1: Estructura interna (simplificada) de una FPGA. Los elementos lógicos (L) están conformados típicamente por una 4-LUT y un FF el cual puede utilizarse o no. Las conexiones programables (R) y los enrutadores programables (S) interconectan los elementos lógicos para obtener funciones lógicas y máquinas de estado complejas. En esta arquitectura no son mostrados otros bloques típicos sino necesarios: generadores de reloj, PLL, memorias, multiplicadores, etc. También puede haber bloques para ciertas funciones específicas, como interfaces de alta velocidad e inclusive microprocesadores.

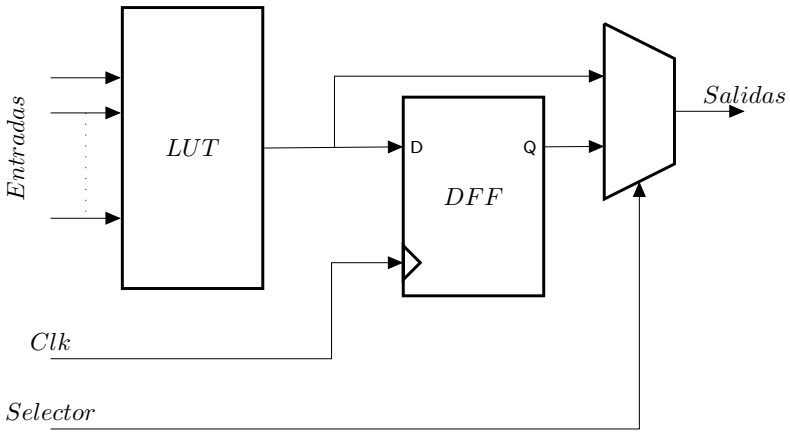

Figura 2.2: Estructura de un elemento lógico.

Naturalmente, un sistema digital requerirá de más capacidad lógica de la

que se podría obtener de un solo bloque L. La interacción de más de un bloque se obtiene a través de las conexiones programables (R) y los enrutadores programables (S). Las conexiones programables interconectan elementos lógicos a través de conexiones de diversa longitud. Algunas de las conexiones son fijas, mientras que otras son programables. Los enrutadores programables son capaces de interconectar bloques R entre sí, de este modo es posible interconectar diversos elementos lógicos de forma compleja.

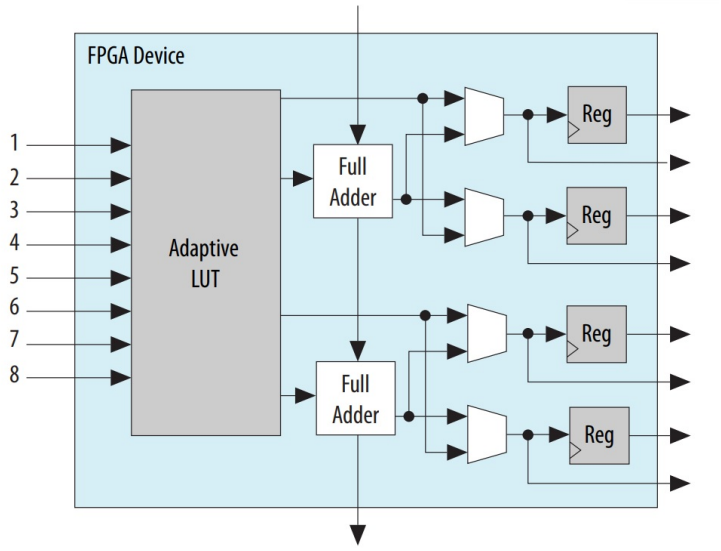

Figura 2.3: Estructura de un bloque ALM. La imagen fue tomada de [18].

Entonces, el proceso de "programación" de una FPGA consiste en programar los elementos lógicos (resultados de las 4-LUT y la utilización o no del FF) y programar las conexiones e interconexiones programables. Este proceso depende de la tecnología de la FPGA. En las secciones siguientes se dará un vistazo a dos de las tecnologías más populares (siendo estas las utilizadas en esta tesis): FPGA basadas en memoria RAM estática (SRAM-FPGA) y FPGA basadas en memoria FLASH (FLASH-FPGA).

2.1.2. FPGA basadas en SRAM

Las FPGA basadas en SRAM almacenan su configuración en celdas de memoria RAM estática (las cuales no requieren refresco, a pesar de ser volátiles). Las ideas básicas de este tipo de memoria y detalles de implementación de algunos bloques de las FPGA son presentados a continuación.

Celda de memoria RAM estática

Una celda de memoria RAM estática consta de 6 transistores (ver figura 2.4) y puede retener dos estados. Si bien no requiere refresco, la celda es volátil, de modo que pierde el valor almacenado cada vez que es apagada. Algunos de sus usos en las SRAM-FPGA son presentados más adelante. Su volatilidad implica que las SRAM-FPGA deben ser "re-programadas" cada vez que se encienden. Este proceso puede realizarse mediante la lectura de una memoria FLASH externa al momento del encendido o a través de una interfaz JTAG, por ejemplo.

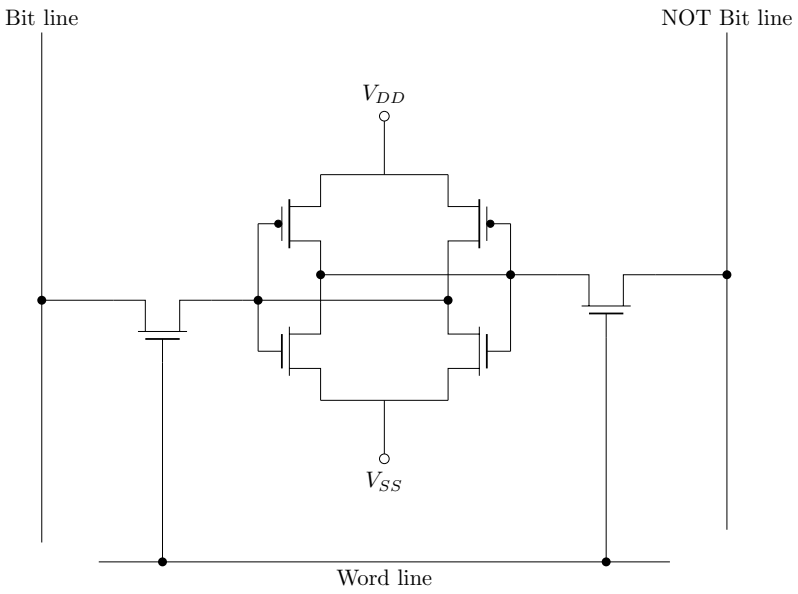

Figura 2.4: Estructura clásica de una celda de memoria RAM estática construida con 6 transistores MOS.

Algunas implementaciones en SRAM-FPGA

En la figura 2.5 se puede observar esquemáticamente una LUT de 2 bits de entrada. Cada resultado de la LUT es efectivamente un bit programable, directamente mapeado en la memoria de configuración. Una red de transistores de paso direccionan los elementos de la tabla hacia la salida, en función del valor de los dos bits de entrada.

Las conexiones entre bloques se realizan mediante transistores de paso, que con capaces de conectar dos lineas especificas (ver figura 2.6). Los *gates*

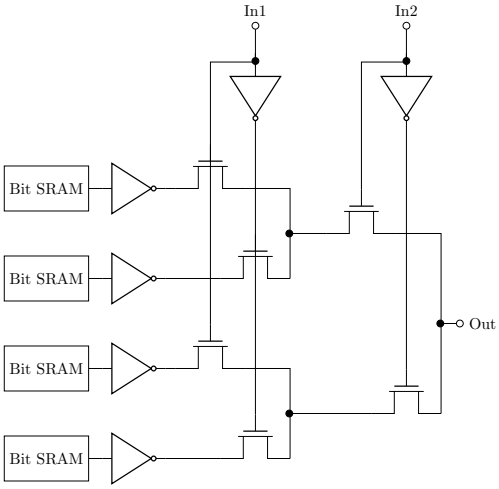

Figura 2.5: Arquitectura de una *look up table* de dos entradas. Los dos bits de entrada seleccionan el bit de la LUT que será seleccionado como salida. Cada bit (4 en este caso) que direcciona la LUT es un bit de SRAM de la memoria de configuración. De este modo, implementar una función lógica $f(In_1, In_2)$ consistirá en almacenar en cada uno de los cuatro bits de la 2-LUT el resultado correspondiente a la tabla de verdad de $f(In_1, In_2)$.

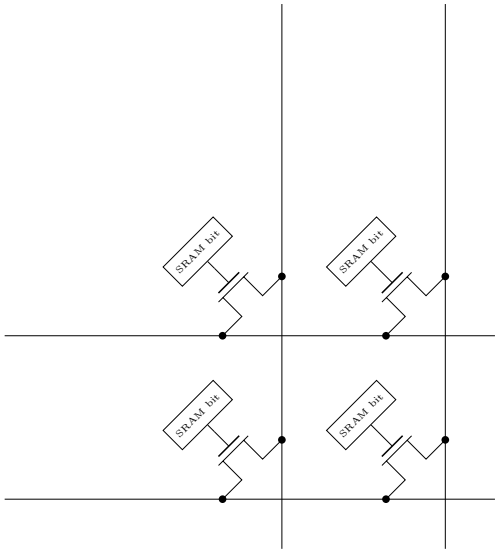

Figura 2.6: Método para la interconexión de bloques de una FPGA. El transistor de paso es controlado por un bit SRAM de la memoria de configuración. De este modo es posible "escribir" una conexión entre bloques.

de los transistores están conectados directamente a bits programables en la memoria de configuración. De este modo, es posible conectar o no dos lineas a través de la memoria de configuración. Finalmente, en la figura 2.7 se muestra

un multiplexor de enrutamiento, que dirige las entradas hacia los elementos lógicos. Las señales de selección del multiplexor se corresponden con elementos de memoria en la memoria de configuración.

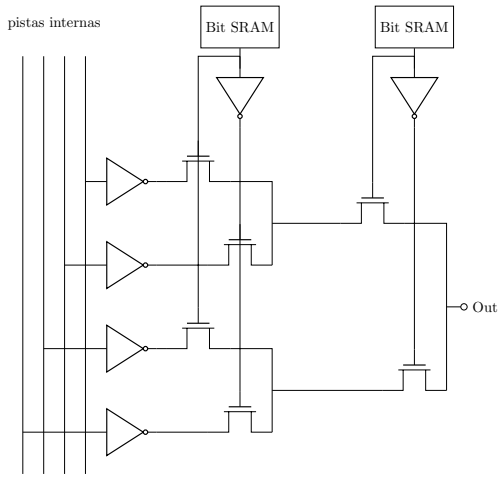

Figura 2.7: Multiplexor de enrutamiento. Las lineas que comunican bloques son las entradas de grandes redes de multiplexores que permiten conectar bloques con otros. Las señales de selección son bit SRAM de la memoria de configuración.

En [19] y [20] se muestra el diseño e implementación de una FPGA sencilla, el cual ilustra muchos conceptos comentados en estas secciones.

2.1.3. FPGA basadas en FLASH

Celda de memoria FLASH

Una celda de memoria FLASH consta de un único transistor construido con tecnología *floating gate* [21] (ver figura 2.8). La polarización del *gate* flotante del transistor se mantiene pese a estar desalimentado. Este efecto permanente dota de memoria al transistor, el cual puede utilizarse como una llave que conducirá o no dependiendo de la polarización del *gate* flotante y de la tensión aplicada. En la figura 2.9 se muestra una configuración con dos transistores FLASH que, dependiendo de las alimentaciones $VCGP$ y $VCGN$ y los valores grabados en cada transistor, se puede obtener tanto VSP ("1" lógico) como VSN ("0" lógico).

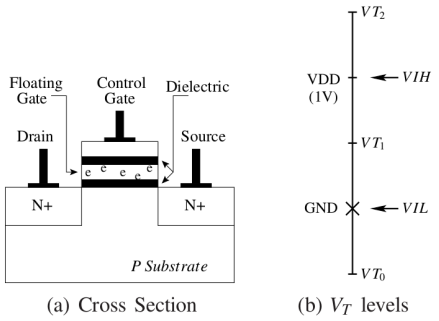

Figura 2.8: Celda de memoria FLASH (imagen obtenida de [22]). A la izquierda se aprecia una sección de un transistor con *floating gate*, el cual es capaz de modificar la tensión umbral de conducción del transistor (como se observa a la derecha).

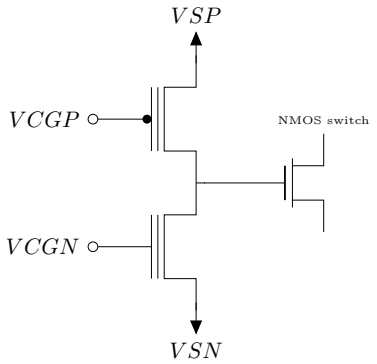

Figura 2.9: Inversor FLASH. Programando adecuadamente los transistores, se puede obtener un comportamiento similar al de un inversor CMOS clásico.

Algunas implementaciones en FLASH-FPGA

Considerando lo comentado anteriormente, la primer opción para implementar *look up tables*, conexiones y enrutadores programables basados en tecnología FLASH, sería implementar las mismas construcciones descritas en 2.1.2 reemplazando los bits de SRAM por una arquitectura como la de la figura 2.9. Esto evitaría la necesidad de grabar la memoria de configuración cada vez que se encienda la FPGA. Adicionalmente, esta topología implica una reducción en área ya que un bit de FLASH requeriría 2 transistores mientras una de SRAM requeriría 6.

De todas maneras, es posible aprovechar el carácter de llave de una celda de FLASH para reducir aún más la cantidad de transistores. En [22] se proponen otras topologías para LUT, multiplexores de enrutamiento e interconexiones programables que impactan en área y consumo.

Capítulo 3

Consumo de potencia en las FPGA

Como caso particular de circuito CMOS, las FPGA poseen un perfil de consumo de potencia que depende de las mismas variables que los circuitos CMOS más simples. Por otro lado, su estructura interna causará que esas dependencias puedan, como corolario, ser traducidas a aspectos más ligados a las FPGA. Las siguientes secciones presentarán el consumo de potencia de las FPGA desde sus elementos constructivos elementales (los transistores CMOS), hasta variables de "alto nivel" y estrictamente dependientes de este tipo de sistemas.

3.1. Consumo de potencia en circuitos CMOS

El consumo de potencia en circuitos digitales basados en tecnología CMOS suele descomponerse en tres grandes fuentes. Por un lado, el consumo dinámico, debido a las conmutaciones de tensión de los transistores ante cargas capacitivas (provocadas por otros transistores), el consumo dinámico producido por corrientes de cortocircuito en las transisiones de nivel y el consumo estático causado por fugas.

El consumo dinámico de potencia en circuitos CMOS suele entenderse a través del ejemplo de un inversor con carga capacitiva C_v (ver figura 3.1), para ser luego generalizado.

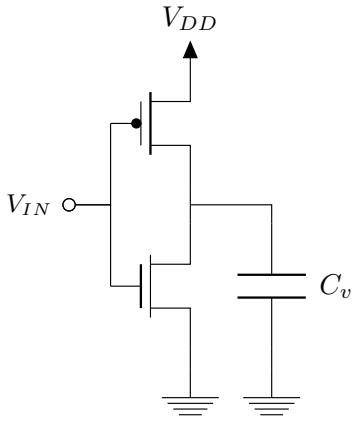

Figura 3.1: Inversor CMOS con carga capacitiva.

De este modo el consumo dinámico promedio puede modelarse como:

$$P = f V_{dd}^2 \sum_{i=1}^{n} \alpha_i C_v^i$$

Donde f es la frecuencia de reloj, V_{dd} es la tensión de alimentación, C_v^i es la capacidad vista desde el nodo i y α_i es la tasa de actividad del nodo, que representa el porcentaje de conmutaciones del nodo i respecto a la cantidad máxima que podría alcanzar (dada por la frecuencia f de operación).

El consumo dinámico por corrientes de cortocircuito se da en las conmutaciones, cuando pueden darse caminos de corriente entre V_{dd} y tierra si las tensiones de entrada varían lentamente (en el caso del inversor, cuando los transistores N y P logran estar conduciendo simultáneamente). Básicamente este tipo de consumo es lineal con la frecuencia (por darse en las conmutaciones), tiene relación directa con los tiempos de subida y bajada de las señales de entrada y mantiene una relación inversa con la tensión de alimentación, al punto de poder ser eliminada con una tensión de alimentación suficientemente baja [1].

Como fue mostrado en la sección 2.1, las FPGA hacen un uso extensivo de transistores de paso. Esto influye en el consumo dinámico debido a las

[1] En un inversor CMOS, si la tensión de alimentación V_{dd} es menor que la suma (en valor absoluto) de las tensiones umbral de cada transistor del par, entonces no podrá ocurrir que ambos transistores conduzcan, para ningún valor de entrada. La tensión de alimentación también posee una cota inferior, la cual fue determinada en [23] y se aproxima a $\frac{8kT}{q}$ (aproximadamente 0.2V a 27°C)

conmutaciones de la carga capacitiva vista desde el transistor. Además, la caída de tensión del transistor puede acarrear un consumo de cortocircuito en etapas posteriores (ya que la excursión no es *rail to rail*).

A medida que los procesos de fabricación logran conseguir transistores más pequeños, el consumo estático debido a las fugas se torna cada vez más imporante en el consumo total de potencia. En [24] las fugas se clasifican en tres grandes grupos: *gate tunneling leakage* (dado por una mala relación de aspecto de los transistores, particularmente de lo fino de la capa de óxido de los *gate*), *sub-threshold leakage* (dada por la corriente que se difunde desde el *drain* y que aumenta cuando los voltages umbral V_t de los transistores se reducen) y *junction tunneling leakage* (el cual aumenta al reducir la concentración de los dopajes).

El compromiso entre estas fuentes de fuga resulta en decisiones en la geometría de los transistores que dependerán del tipo de aplicación. Como ejemplo, en los circuitos SRAM, en comparación con los circuitos lógicos, se suelen utilizar canales de mayor longitud para minimizar la variación en la cantidad de dopantes. Esto causa que las circuitos SRAM tengan fugas dominadas principalmente por el *gate leakage*, mientras que en los circuitos lógicos las fugas serán dominadas por el *sub-threshold leakage*. Un análisis extenso de las características de las fugas en circuitos CMOS y de técnicas para convivir con ellas y/o minimizarlas también son presentadas en [24].

3.2. Consumo de potencia en el caso particular de las FPGA

Como fue mencionado en 3.1, el consumo dinámico (al estar relacionado con la evolución temporal del sistema) no puede medirse desacoplado al consumo estático. Una forma sencilla es estimarlo es a partir del consumo total y del consumo estático (básicamente como la diferencia entre estos valores).

Diversas variables tanto de la FPGA como del diseño implementado impactan en el consumo total del sistema, ya sea por aportar al consumo estático, al dinámico o a ambos.

3.2.1. Frecuencia de ejecución y tensión de alimentación

Tanto la tensión de alimentación como la frecuencia de ejecución tienen un rol significativo en el consumo total de una FPGA, particularmente en el consumo dinámico.

Decrementar la frecuencia de ejecución puede reducir la potencia pero no necesariamente en forma proporcional al decremento. Esto puede ocurrir si todos los sub-sistemas implementados no operaran a la misma frecuencia (muy comúnmente los diseños poseen, por ejemplo, algún PLL para operar ciertas partes del sistema a una frecuencia diferente a la de la fuente principal de reloj).

En lo que respecta a la tensión de alimentación, disminuirla resultará beneficioso en términos de consumo dinámico, gracias a su relación cuadrática. De todos modos, el margen para modificar la tensión de alimentación suele ser pequeño. Típicamente las FPGA tienen múltiples alimentaciones destinadas a diferentes elementos (*core*, *buffers* de entrada salida, sus PLL, etc.).

3.2.2. Tamaño del diseño

Con los argumentos dados en la sección 3.1, una mayor utilización de la FPGA implica una mayor cantidad de nodos en actividad. Esto impacta directamente en el consumo dinámico por la cantidad de transistores utilizados, el uso intensivo de transistores de paso y por la capacidad de carga de los transistores debido a largos caminos entre bloques.

3.2.3. Tamaño de la FPGA

A pesar que el consumo estático no es reducible al aporte individual de cada transistor (ya que pueden existir interelaciones entre sí que causen fugas), una aproximación razonable sería que, dada la misma tecnología de fabricación y arquitectura, una FPGA con menor cantidad de elementos lógicos tendrá un consumo estático menor que otra de mayor tamaño.

En escenarios sin reconfiguración, el tamaño de una FPGA no podrá ser menor que el tamaño del diseño que se desee implementar (adicionalmente los diseñadores suelen reservar algo de lógica con el fin de realizar cambios a futuro, mejoras, correcciones de errores, etc.). Esto significa que el consumo estático siempre estará atado al tamaño del diseño y también a los tamaños

de FPGA disponibles en el mercado.

La utilización de una FPGA de tamaño "adecuado" al diseño tendrá beneficios respecto al consumo estático. Como contrapartida, el uso de FPGA de menor tamaño impactará en el porcentaje de utilización de la misma (relación entre la cantidad de elementos lógicos necesarios y los disponibles en la FPGA). Con una utilización elevada de la FPGA, los caminos entre bloques pueden alargarse (debido a que las herramientas de *place and route* trabajarán con poco "margen" de elementos lógicos), causando un mayor consumo dinámico.

3.2.4. Tecnología y Arquitectura

Como fue comentado previamente, la tecnología de fabricación de la FPGA condiciona notáblemente al consumo estático (escencialmente por la tecnología de la memoria de configuración). Sin embargo, su influencia también se traduce al consumo dinámico. La arquitectura interna de la FPGA tiene un rol de interés, ya que los fabricantes disponen de los elementos lógicos de sus FPGA en cierta estructura que cambia entre fabricantes y modelos. Esto impacta en el nivel de utilización de la FPGA (cantidad de recursos utilizados para instanciar el diseño, respecto a la cantidad de recursos total), ya que un mismo diseño requerirá más o menos elementos lógicos en función de cómo han sido estos construidos, pudiendo incrementar el factor de actividad y la capacidad vista.

3.2.5. Herramientas de síntesis y de "place and route"

En la línea de la sección anterior (3.2.4), las herramientas de desarrollo condicionan cómo el diseño será sintetizado e instanciado con los elementos lógicos disponibles. Herramientas eficientes minimizarán la cantidad de elementos lógicos necesarios para el diseño propuesto así como la cantidad de interconexiones y el largo de las caminos entre bloques.

3.3. Consumo del core de la FPGA como consumo representativo

Como fue comentado en la sección 3.1, el consumo de un circuito CMOS puede describirse en términos de consumo estático y consumo dinámico. En

esta tesis el foco estará centrado únicamente en el consumo del *core* de la FPGA (tanto estático como dinámico). Esta decisión está fundamentada por las siguientes observaciones:

1. El consumo de los puertos de entrada y salida está gobernado esencialmente por la tensión de alimentación, la capacidad de carga por cada pin y la frecuencia de conmutación (ver 3.1). Con la tensión de alimentación adecuada, el consumo de los puertos de entrada/salida será independiente de la FPGA utilizada.
2. El consumo del *core* es característico de cada FPGA. El consumo del *core* de una FPGA habla de su tecnología, de su tamaño y de cómo se utilizan sus recursos al momento de sintetizar y generar una instancia de un diseño en ella.

Con motivo de simplificar la redacción, cuando se hable de consumo estático, dinámico y total se estará haciendo referencia a los consumos del *core*, salvo que se detalle otra cosa.

3.4. Metodología para la medida del consumo de potencia

Una de las formas más aceptadas para la medida de consumo estático de una FPGA consiste en medir su consumo, habiendo cargado el diseño que se quiere utilizar y habiendo bloqueado las entradas del sistema, particularmente la señal de reloj. Al bloquear las entradas, cualquier función combinatoria que haya sido implementada quedará completamente sin actividad. Al bloquear la señal de reloj cualquier máquina de estados implementada se detendrá, resultando así en un cese total de la actividad del *core*.

No es posible medir el consumo dinámico directamente, pero puede calcularse realizando dos medidas. La primera consiste en medir el consumo estático (como fue comentado anteriormente) y la otra consiste en medir el consumo total. Así, el consumo dinámico puede calcularse restando el consumo estático al total. La figura 3.2 muestra la medida de consumos realizada en dos pasos.

En ambos casos el consumo se determina a través de una resistencia en serie con la alimentación del *core*. El valor de esta resistencia debe ser lo suficientemente bajo como para que la tensión de alimentación se mantenga en

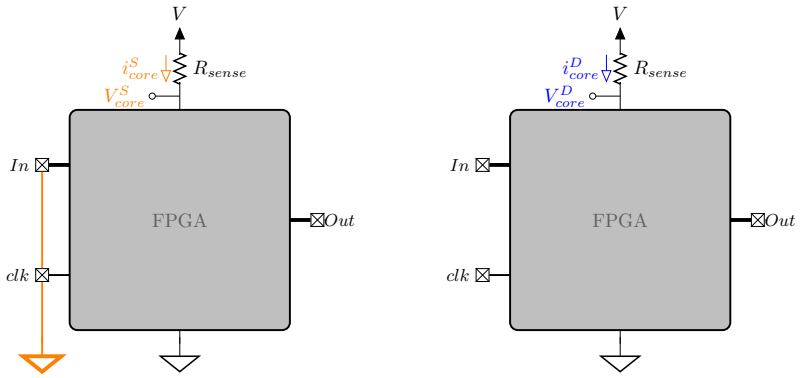

Figura 3.2: Posible ensayo para medir el consumo de potencia estático (izquierda) y total (derecha) del *core* de una FPGA. Para medir el consumo estático, todas las señales de entrada del diseño deben ser bloqueadas (inclusive la señal de reloj). En el caso del consumo total, la medida se realiza con el sistema operando normalmente. En ambos casos la medida se realiza mediante una resistencia de sensado R_{sense} en serie con la alimentación de la FPGA. Todos las señales del salida del diseño deben ser conectadas a algún pin de la FPGA. De este modo el consumo estático de potencia se obtiene mediante: $P^S_{core} = V^S_{core} i^S_{core}$ y el total como: $P^T_{core} = V^T_{core} i^T_{core}$.

un valor aceptable. A su vez, los resultados obtenidos utilizando resistencias de valor muy bajo serán más susceptibles al ruido. En el primer caso, el estático, tanto la señal de reloj como las entradas son bloqueadas. En el segundo, el circuito es utilizado como en el escenario esperado de uso.

Si bien la actividad de los *buffers* de entrada/salida tendría un aporte despreciable en el consumo del *core* de la FPGA, es necesario conectar todas las salidas del diseño bajo prueba a algún pin de salida. Esto se debe a que el compilador puede realizar ciertas optimizaciones en el diseño, en caso de haber partes del mismo que no tengan efecto sobre algún pin de salida en forma directa o indirecta. Para evitar simplificaciones sin necesidad de conectar cada salida del diseño a un pin, una técnica es la que se utiliza en [25]. Aquí, todas las señales de salida son conectadas a un único pin, a través de una red de compuertas "XOR", como se muestra en la figura 3.3.

Como se puede comprobar en la tabla de verdad de una compuerta "XOR", cualquier cambio en una de sus entradas causará un cambio en su salida. Por esta razón el compilador no podrá simplificar el diseño. Como consecuencia, el consumo dinámico será superior, no solo por el agregado de un circuito auxiliar, sino también porque la compuerta "XOR" es la que posee, en términos estadísticos, el mayor consumo dinámico.

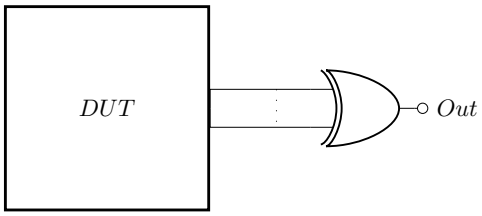

Figura 3.3: Método para tener una instancia del DUT sin necesidad de conectar todos sus señales de salida a pines de la FPGA.

Otra ventaja de esta técnica es que requiere de un solo pin de salida. Esta ventaja podría aprovecharse para evaluar el desempeño (respecto al consumo de potencia) de cierta familia de FPGA a pesar de no contar con una FPGA con la cantidad de pines suficientes.

Adicionalmente, una red de compuertas "XOR" podría utilizarse para calcular el consumo dinámico del DUT con mayor precisión, como se detalla a continuación.

3.4.1. Mejora de la precisión en el cálculo del consumo dinámico

La técnica de encausar todas las señales de salida del DUT a un solo pin mediante una red de compuertas "XOR" (para evitar simplificaciones de diseño) sigue funcionando si se generaran más instancias del DUT dentro de la FPGA (ver figura 3.4).

Si llamamos n a la cantidad de instancias del DUT, el consumo total del diseño se puede determinar como:

$$C_T^1 = nC_{DUT} + C_{XOR}$$

donde C_{DUT} es el consumo total del diseño a medir y C_{XOR} es el consumo total de la red auxiliar.

Como los circuitos auxiliares aportan términos de consumo (tanto estático como dinámico), para deducir el consumo del diseño bajo prueba se debe hacer un ensayo adicional (C_T^2)y sólo con los circuitos auxiliares.

De esta manera podría determinarse el consumo del diseño bajo prueba a partir de los siguientes ensayos:

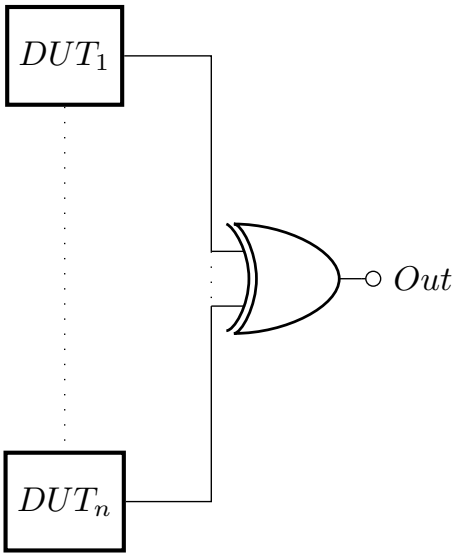

Figura 3.4: El esquema de la figura 3.3 puede generalizarse a más instancias del DUT.

$$\begin{cases} C_T^1 &= nC_{DUT} + C_{XOR} \\ C_T^2 &= C_{XOR} \\ C_{DUT} &= \frac{C_T^1 - C_T^2}{n} \end{cases}$$

El consumo dinámico del primer ensayo ($C_D^1 = nC_D^{DUT} + C_D^{XOR}$) depende del consumo dinámico del DUT, amplificado "n" veces. De este modo, el consumo total medido tendrá un componente mayor de consumo dinámico. En escenarios de grandes consumos estáticos y bajos consumos dinámicos, este método podría mejorar la precisión en el cálculo del segundo (ya que la diferencia entre los consumos estático y totales será mayor).

La medida de los circuitos auxiliares (de la que resulta C_T^2) posee dos inconvenientes para su medida: la síntesis del circuito y su actividad. La síntesis de los circuitos auxiliares posee los mismos inconvenientes que justifican su existencia, es decir, que la imposibilidad de mapear cada entrada de la red auxiliar a un pin de entrada/salida causará que el proceso de síntesis acabe por simplificarla (pudiendo anularla completamente). Como fue comentado, el consumo dinámico del circuito auxiliar depende de la actividad de sus señales de entrada, la cual no sería la misma en ambos ensayos (esto modificaría el consumo dinámico de la red).

Estos inconvenientes pueden mitigarse utilizando un bloque auxiliar, como

los *Linear Feedback Shift Register* (LFSR). Los LFSR son bloques generadores de vectores pseudo-aleatorios muy sencillos de implementar, ya que consisten en un registro de desplazamiento con algunos puntos de realimentación (ver figura 3.5).

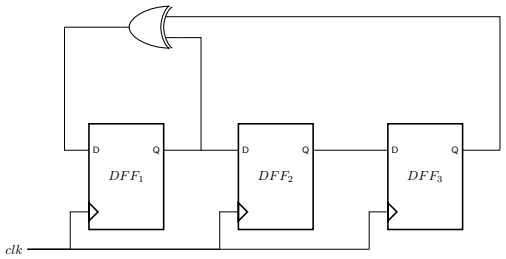

Figura 3.5: LFSR de 3 bits.

Para cada ancho de palabra (cantidad de bits del registro de desplazamiento, n) se pueden encontrar puntos de realimentación que producen que el registro recorra todos los valores posibles (2^n). En [26] se pueden encontrar algunos detalles de implementación, particularmente las posiciones del registro a realimentar, para cada largo.

Así, al primer ensayo se le agrega tantos bloques LFSR como instancias de DUT (como se muestra en el esquema de la figura 3.6).

Para evitar simplificaciones en la sintetización, cada LFSR tiene su último bit conectado a un pin de salida. El consumo total será:

$$C_1^T = nC_{DUT} + nC_{LFSR} + C_{XOR}^{DUT}$$

donde C_{LFSR} es el consumo de un LFSR y C_{XOR}^{DUT} es el consumo de la red de auxiliar cuando excitada por las instancias del DUT.

En el segundo ensayo, los DUT son reemplazados por los LFSR, conectando cada bit del registro de desplazamiento de los LFSR a una entrada de la red "XOR". Esto resulta en un consumo de:

$$C_2^T = nC_{LFSR} + C_{XOR}^{LFSR}$$

donde C_{XOR}^{LFSR} es el consumo de la red auxiliar cuando es excitada con las instancias del LFSR.

Si bien el consumo estático de la red auxiliar es igual en ambos casos ($Cs_{XOR}^{DUT} = Cs_{XOR}^{LFSR}$), no es posible afirmar que sus consumos dinámicos lo sean

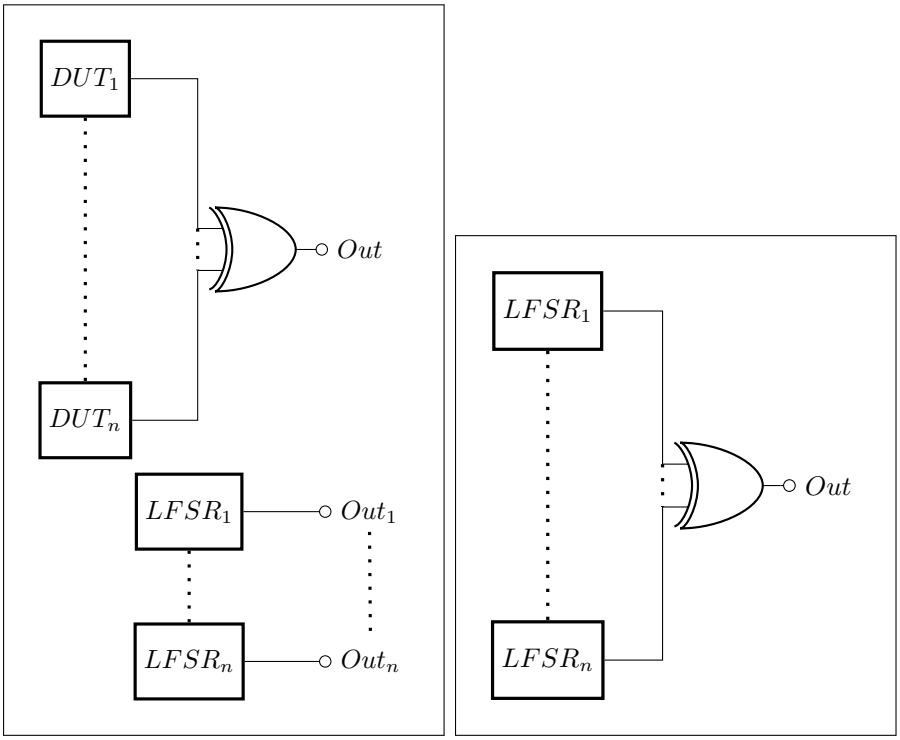

Figura 3.6: Ensayo que resuelve los inconvenientes presentados anteriormente. Aquí, en el segundo ensayo (a la derecha) la red auxiliar es ejercitada mediante bloques LFSR. Para poder descontar el consumo de los LFSR, la misma cantidad de estos bloques deben agregarse al primer ensayo (a la izquierda). El último bit del registro de desplazamiento es conectado a una salida de la FPGA para evitar simplificaciones al momento de la síntesis.

($Cd_{XOR}^{DUT} \neq Cd_{XOR}^{LFSR}$) ya que el nivel de actividad de los DUT y los LFSR no serán iguales. De este modo:

$$\frac{C_1^T - C_2^T}{n} = C_{DUT} + \frac{(Cd_{XOR}^{DUT} - Cd_{XOR}^{LFSR})}{n}$$

Si el consumo dinámico de la red auxiliar fuera similar en ambos casos, podría deducirse el consumo del diseño bajo prueba como en el caso anterior:

$$si \;\; Cd_{XOR}^{DUT} \approx Cd_{XOR}^{LFSR} \Rightarrow C_{DUT} \approx \frac{C_1^T - C_2^T}{n}$$

Una posible mejora a esta técnica se muestra en la figura 3.7, donde las señales de salida de los DUT son conducidas a través de compuertas "AND". Cada compuerta tiene una señal de "habilitación", de forma que al asignar esta señal en cero lógico se minimizará la actividad de la red de compuertas "XOR" (ya que todas sus entradas estarán bloqueadas).

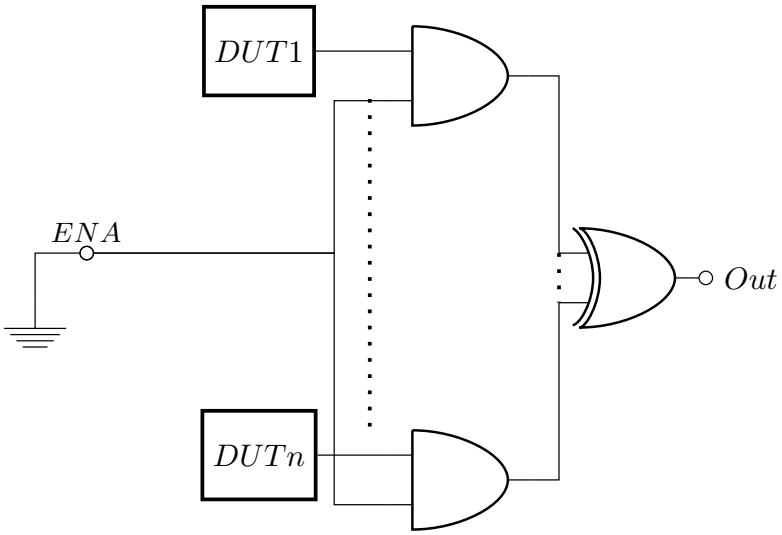

Figura 3.7: Una posible mejora al circuito de medida consiste en bloquear la salidas de los DUT antes de ser conectadas a la red auxiliar. Esto disminuiría la actividad de la misma y por tanto su consumo dinámico.

En este escenario, el consumo de la red auxiliar se reduce al estático:

$$C_1^T = nC_{DUT} + C_{AND}^{DUT} + Cs_{XOR} + nC_{LFSR}$$

donde C_{AND}^{DUT} es el consumo de las compuertas "AND" de bloqueo cuando sus

entradas son excitadas con los DUT (nótese que sólo una de las entradas de cada compuerta tiene actividad y su salida siempre es cero). La segunda medida reemplaza los DUT por los LFSR.

Así:

$$C_2^T = C_{AND}^{LFSR} + Cs_{XOR} + nC_{LFSR} \Rightarrow \frac{C_1^T - C_2^T}{n} = C_{DUT} + \frac{(Cd_{AND}^{DUT} - Cd_{AND}^{LFSR})}{n}$$

Los consumos de las compuertas "AND" de bloqueo serán presumiblemente similares en ambos casos, ya que sólo tienen actividad en una de sus entradas y sus salidas serán siempre cero. En este caso la aproximación del consumo del diseño bajo prueba

$$C_{DUT} \approx \frac{C_1^T - C_2^T}{n}$$

es más precisa.

Otra forma de evitar la simplificación del diseño en tiempo de síntesis y minimizar la actividad de los circuitos auxiliares, es mediante la utilización de multiplexores, como en la figura 3.8.

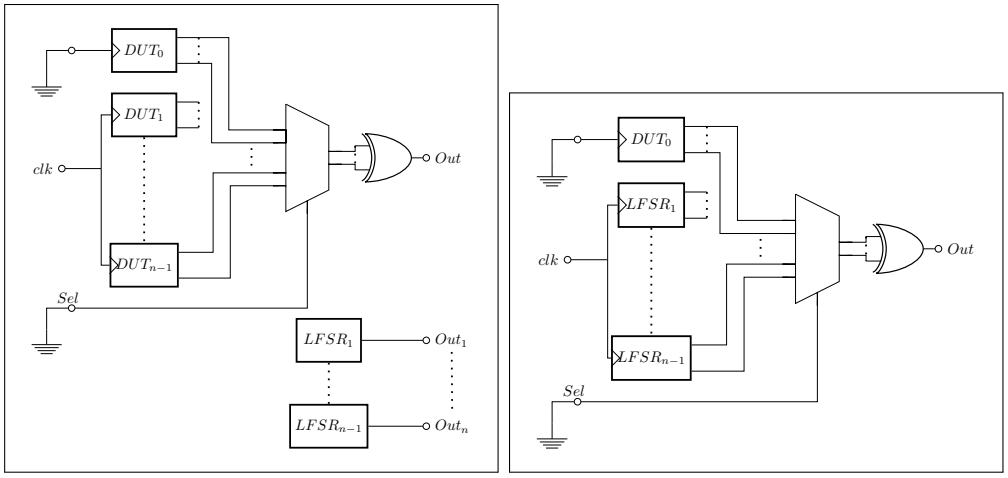

Figura 3.8: Otra forma de bloqueo de la actividad de la red auxiliar, mediante el uso de multiplexores. Las señales de selección se utilizan para conectar a la red auxiliar a un circuito sin actividad (en este caso una instancia del DUT sin reloj, aunque podría ser cualquier otro circuito).

La etapa de bloqueo previa a la red de compuertas XOR es implementada con multiplexores, los cuales canalizan las salidas de los DUT hacia la red auxiliar. Si las señales de selección son conectadas a pines de entrada, el proceso de síntesis no podrá simplificar el diseño. Adicionalmente, si una de las instancias del DUT se la deja sin fuente de reloj y se la enruta a través

de los multiplexores (con las señales de selección) entonces la red auxiliar no tendrá actividad.

En este caso también se podría obtener una buena aproximación al consumo del diseño bajo prueba con los mismos ensayos:

$$\begin{cases} C_T^1 &= (n-1)C_{DUT} + Cs_{DUT} + C_{MUX}^{DUT} + Cs_{XOR} + (n-1)C_{LFSR} \\ C_T^2 &= C_{MUX}^{LFSR} + Cs_{XOR} + (n-1)C_{LFSR} + Cs_{DUT} \end{cases}$$

$$\Rightarrow \frac{C_1^T - C_2^T}{(n-1)} = C_{DUT} + \frac{Cd_{MUX}^{DUT} - Cd_{MUX}^{LFSR}}{n-1} \approx C_{DUT}$$

En el capítulo 6 se presentan medidas de consumo de potencia del circuito implementado, utilizando los tres métodos descritos (red de compuertas "XOR" sin bloqueo, red con bloqueo mediante compuertas "AND" y con bloqueo mediante multiplexores).

Capítulo 4

Sistema implementado

En el presente capítulo se describe la arquitectura de la unidad de control diseñada para manejar el circuito de neuroestimulación descrito en 1.3. Tanto la fuente de tensión como la de corriente son regulables según una tensión analógica. Debido a esto y a que no todas las FPGA utilizadas poseen módulos *Digital to Analog Converter* (DAC) embebidos, se optó por agregar dos DAC externos con comunicación SPI (MAX5532 [27]). El circuito completo resultante es el de la figura 4.1.

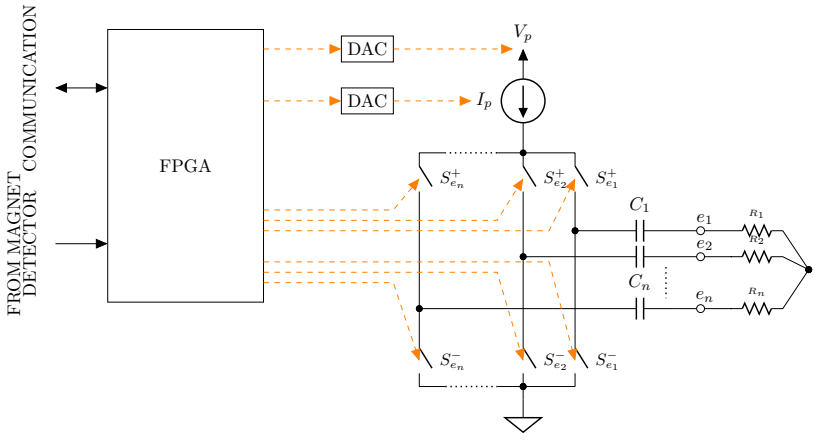

Figura 4.1: Sistema completo con dos DAC externos MAX5532.

Si bien todo el diseño fue realizado como parte de esta tesis, algunos detalles han sido omitidos porque son confidenciales.

4.1. Arquitectura del circuito

Como se observa en la figura 4.2, la solución se basa en varios tipos de bloques que se detallan a continuación.

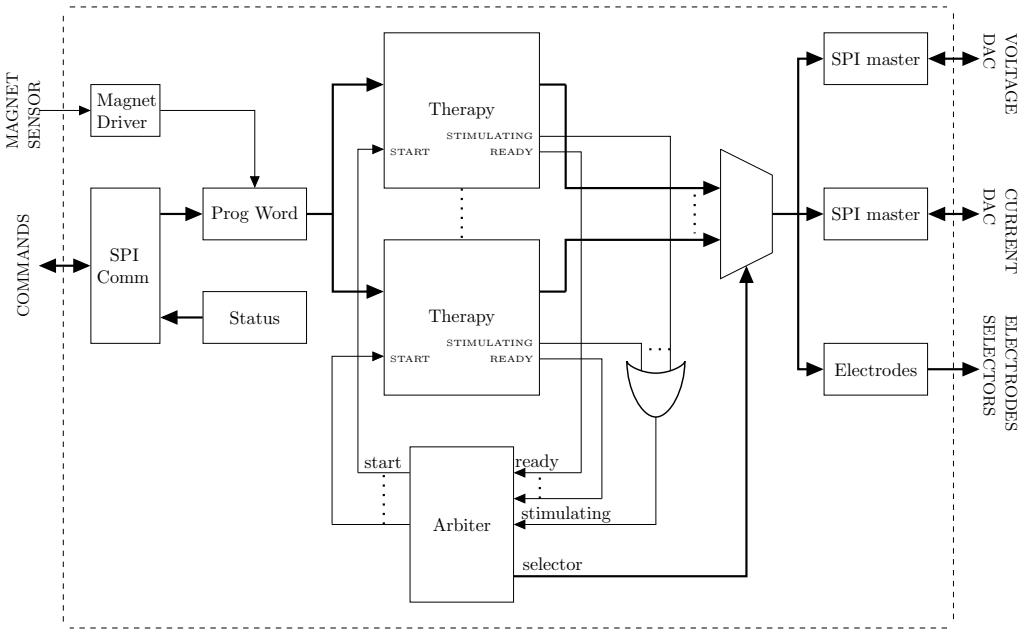

Figura 4.2: Sistema completo.

4.1.1. Bloques de terapia (Therapy)

Estos bloques son encargados de manejar todo el *hardware* relativo a la generación y entrega de los pulsos de estimulación. Por este motivo, un bloque de este tipo se encarga del encendido y configuración de la fuente de tensión y de corriente del circuito de estimulación, así como la conexión de la fuente de corriente a los electrodos. La temporización de los anchos de pulso y la frecuencia de estimulación son configurables a través de dos buses que el bloque tiene como entrada. La selección de electrodos por los cuales entregar terapia también es configurable a través de un bus dedicado que el bloque tiene como entrada[1]. Este tipo de bloque implementa una máquina de estados la cual opera según el diagrama de flujo de la figura 4.3 (nótese que la ejecución completa de esta máquina de estados desencadena un único pulso de estimulación).

[1]Los buses de entrada de los bloques de terapia han sido reducidos en la figura 4.2 para simplificar la interpretación de la arquitectura completa. En este diseño, todos los buses de

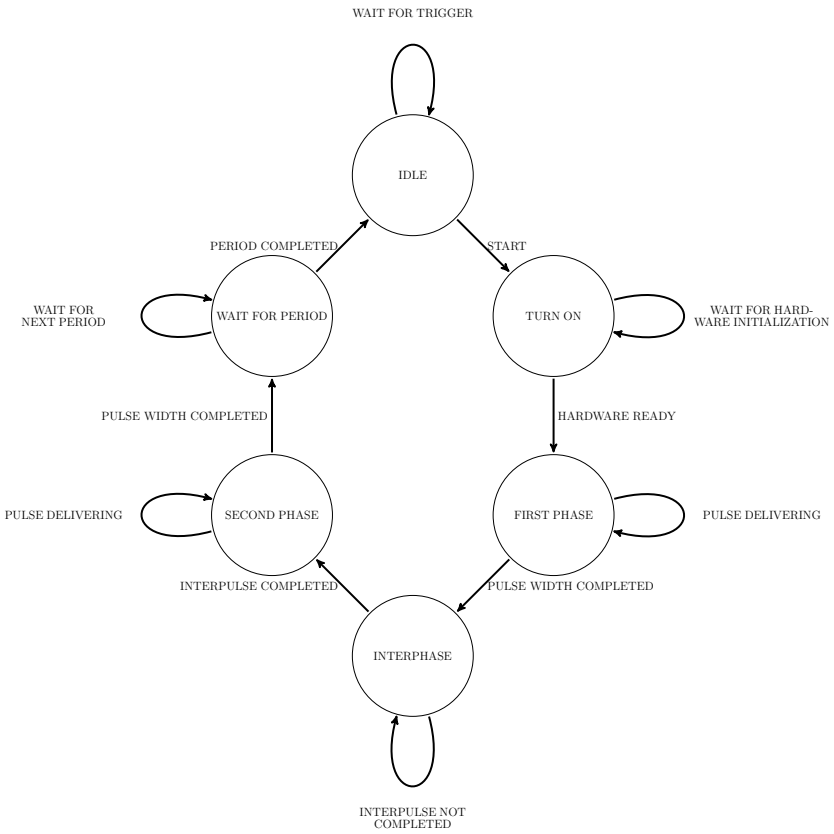

Figura 4.3: Máquina de estados de los bloques de terapia. Los mismos manejan el circuito conforme la forma de onda programada.

En la arquitectura presentada, cada bloque de terapia maneja efectivamente una terapia tal y como fue descrita en 1.2.

En este esquema paralelo, las fuentes de corriente y tensión, así como los electrodos son recursos compartidos por todas las terapias. Esto obliga a implementar algún tipo de arbitraje para controlar el circuito.

4.1.2. Árbitro (Arbiter)

Este bloque se encarga de arbitrar las terapias con el fin de utilizar apropiadamente el hardware disponible.

El bloque le asigna los recursos del circuito a una terapia a la vez y también genera la señal de "start" que desencadenará el comienzo de la estimulación. La estimulación será manejada según la terapia elegida por el árbitro. La selección es realizada en función de la disponibilidad y la prioridad que cada terapia

entrada de los bloques de terapia están conectados al registro de programación (Prog Word).

posee.

La disponibilidad de una terapia es determinada por el árbitro a través de una señal ("ready") la cual indica que una terapia se encuentra lista para generar un pulso de estimulación. La prioridad de una terapia se determina en función del tiempo de espera de la terapia (estando disponible) y de la frecuencia de estimulación a la que ha sido programada.

La figura 4.4 muestra en forma esquemática la implementación del bloque. El árbitro se encarga de monitorear un bus de entrada que contiene las señales de "ready" de todas las terapias presentes en el diseño. Cuando una terapia se encuentra lista para estimular, sus señal de salida "ready" toma el valor "1" lógico. Cuando esto ocurre, el árbitro almacena este dato como una "solicitud" de una terapia por estimular. La forma de gestionar las solicitudes se realiza mediante dos colas con prioridad, implementadas a través de dos *stack* FIFO (el diseño del *stack* es el presentado en [28]). Una cola almacena las solicitudes asociadas a terapias de baja frecuencia mientras que la otra almacena las de alta frecuencia. El criterio para determinar si una frecuencia es "alta" o "baja" es el de comparar contra una frecuencia umbral de 1,5kHz (una terapia con frecuencia de 1,5kHz se procesa como una de baja frecuencia). Cuando la terapia que genera el pulso de estimulación termina su período (esto es determinado gracias a la señal "stimulating"), el árbitro atenderá las solicitudes encoladas de baja frecuencia, de una a la vez y en orden de llegada. Cuando no se encuentren solicitudes de baja frecuencia, procesará las de alta frecuencia según el mismo principio.

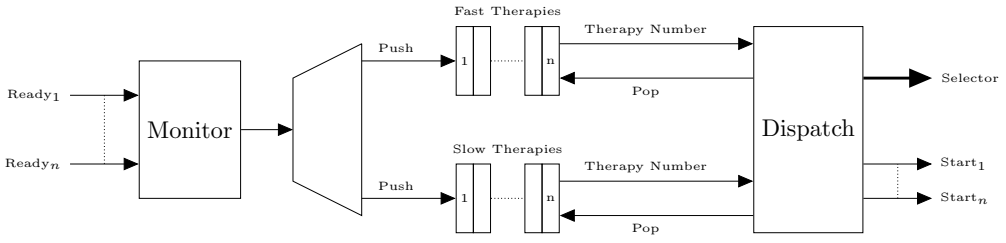

Figura 4.4: Bloque encargado del arbitraje entre terapias.

El árbitro conecta a las salidas del sistema la terapia que corresponda a través de un arreglo de multiplexores.

4.1.3. Sensor Margnético (Magnet Driver)

Este bloque monitorea el valor de la entrada "reed switch" proveniente de un sensor magnético. Se toman muestras de la señal cada 30ms y si dos muestras consecutivas son leídas en "1", el bloque asume presencia efectiva de campo magnético y lo indica asignando la salida "magnet present" en "1". Esta señal se utiliza para deshabilitar las terapias de estimulación (la deshabilitación o no de la terapia en presencia de campo magnético es una característica programable).

4.1.4. Registro de Programación (Prog Word)

Este bloque es un arreglo de elementos de memoria que almacenan todos los parámetros programables relativos a la terapia. Además de almacenar los tiempos y amplitudes que cada terapia utilizará, también permite habilitar y deshabilitar la inhibición de los pulsos de estimulación ante la presencia de campo magnético.

4.1.5. Módulos SPI maestro (SPI Comm)

Dos módulos SPI maestro son utilizados para la configuración de los DAC asociados a la fuente de tensión y a la de corriente. Debido al uso específico de este módulo (comunicación con el chip MAX5532) su diseño fue realizado como parte del trabajo de tesis, con la intención de hacer una versión "mínima" respecto a su funcionalidad.

4.1.6. Registro de estado (Status)

Este registro almacena información del estado del circuito. Para esta versión, el sistema reporta los siguientes datos:

- Terapia en curso.
- Terapia apagada.
- Terapia detenida por detección de campo magnético.

4.1.7. Interfaz de comunicación: SPI esclavo (SPI slave)

La interfaz de comunicación SPI permite:

- Programar las terapias.
- Comenzar la estimulación.
- Detener la estimulación.
- Producir el reset del circuito.
- Leer información de *status*.

El módulo fue realizado como parte del trabajo de tesis con el fin de utilizar una diseño óptimo en términos de tamaño.

4.2. Diseño paramétrico

El diseño fue descrito en *Verilog* y realizado en forma paramétrica, pudiendo configurarse al momento de la síntesis la cantidad de electrodos, la cantidad de terapias y la frecuencia del reloj del sistema. La frecuencia mínima a la que el diseño es capaz de operar está dada por la máxima frecuencia de estimulación y la cantidad de terapias. Como la estimulación está basada en la producción (por parte de los bloques de terapia) y el consumo (por parte del árbitro) de solicitudes de envío de pulsos, una frecuencia de ejecución adecuada es la que producirá una latencia despreciable. La latencia dependerá de la cantidad de terapias sintetizadas (ya que esto define el tamaño de las colas FIFO) y la frecuencia de estimulación. Para la frecuencia de estimulación máxima (10kHz) y 4 terapias (que es el caso que se evaluará en esta tesis), se requerirá de una frecuencia mínima de reloj de unos 7,5MHz para asegurar una latencia menor al 1 % del período de la terapia (1μs). Este valor fue obtenido tras la observación de que, en el caso de una única terapia habilitada, se requieren de 13 períodos de reloj desde que la terapia termina hasta que es procesada por el árbitro.

4.3. Verificación del diseño

Aquí se presentan algunos resultados de la verificación del diseño, tanto a nivel lógico como a nivel funcional.

Los módulos fueron verificados a nivel lógico utilizando *ModelSim Altera Starter Edition* (10.3d) y a nivel funcional utilizando utilizando el *kit* DE0-CV.

4.3.1. Verificación a nivel lógico

Controlador del sensor magnético

En la imagen 4.5 se muestra el comportamiento a nivel lógico del controlador del sensor magnético. A los efectos de reducir los tiempos de simulación, el tiempo para la detección de actividad magnética se estableció en 5 períodos de reloj al momento de la síntesis.

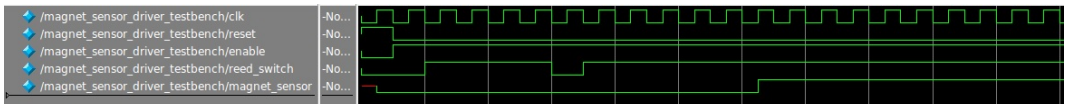

Figura 4.5: Simulación del módulo detector de actividad magnética.

Pila FIFO

En la imagen 4.6 se muestra el desempeño de del *stack* FIFO. Para su verificación, se sintetizó el *stack* con un tamaño de 2 elementos y 8 bits de ancho de palabra. Se encolaron y desencolaron diferentes números verificando el manejo apropiado de los datos y las señales de control de *stack* vacío y lleno.

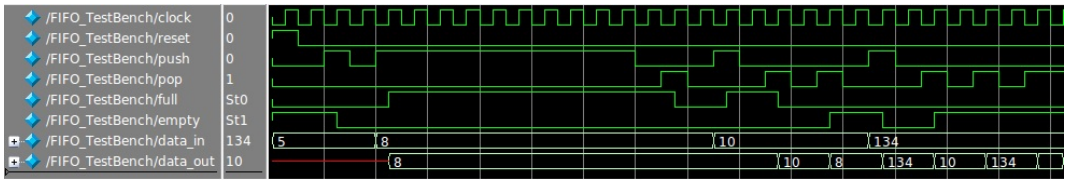

Figura 4.6: Simulación del *stack* FIFO sintetizado con 8 bits de ancho de palabra y profundidad igual a 2.

Árbitro

En la imagen 4.7 se muestra parte de la verificación del módulo de arbitraje de terapias. Se sintetizó un diseño con dos terapias "rápidas". Las terapias son desencadenadas con las el bit correspondiente del bus de *start*. El selector es modificado antes de comenzar la siguiente terapia y las terapias no comienzan hasta no terminar la estimulación anterior.

Módulo de terapia

Este módulo fue probado en una instancia del sistema completo sin el módulo de comunicación, para simplificar la simulaciones. En el sistema que

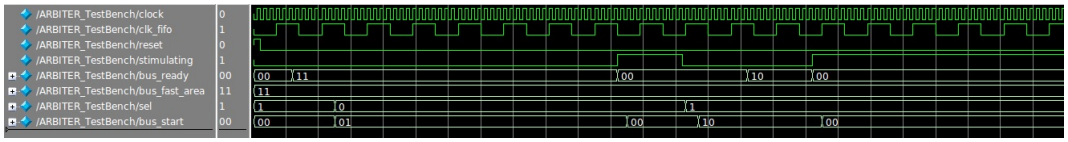

Figura 4.7: Desempeño del bloque de arbitraje de terapias, sintetizado para dos terapias. Las frecuencia de reloj del arbitro y del *stack* no necesariamente deben ser iguales.

se aprecia en la figura 4.8 existen 4 terapias con diferentes parámetros (electrodos, amplitudes y tiempos). Como puede observarse, la selección de electrodos amplitudes y tiempos se realizan respetando la terapia seleccionada.

4.3.2. Verificación a lo largo del diseño

Para cada módulo, varios archivos de simulación fueron realizados para su verificación. Se creó un *script* en *tcl-tk* que permite ejecutar en forma automática todos los archivos de simulación creados para un módulo y comparar su resultado con un conjunto de resultados esperados. Esto fue utilizado para realizar una verificación de "regresión" a medida que los módulos fueron evolucionando hasta su versión final.

4.3.3. Verificación a nivel funcional

Plataforma de verificación

El sistema completo fue verificado a nivel funcional como en el esquema de la figura 4.9, el mismo consta de un *kit* de desarrollo DE0-CV y el circuito de estimulación.

En la FPGA del *kit* existe una instancia del DUT que maneja los puertos de salida hacia el circuito de estimulación. Adicionalmente, se agregó un diseño que es capaz de enviar los comandos hacia el DUT y recibir sus respuestas. Este circuito posee un módulo SPI maestro por el cual se comunica con el DUT. Los comandos y configuración enviada hacia el DUT es seleccionada de un conjunto pre-definido con los interruptores del *kit*, del siguiente modo:

El *status* reportado por el DUT se despliega en los LEDs del *kit*.

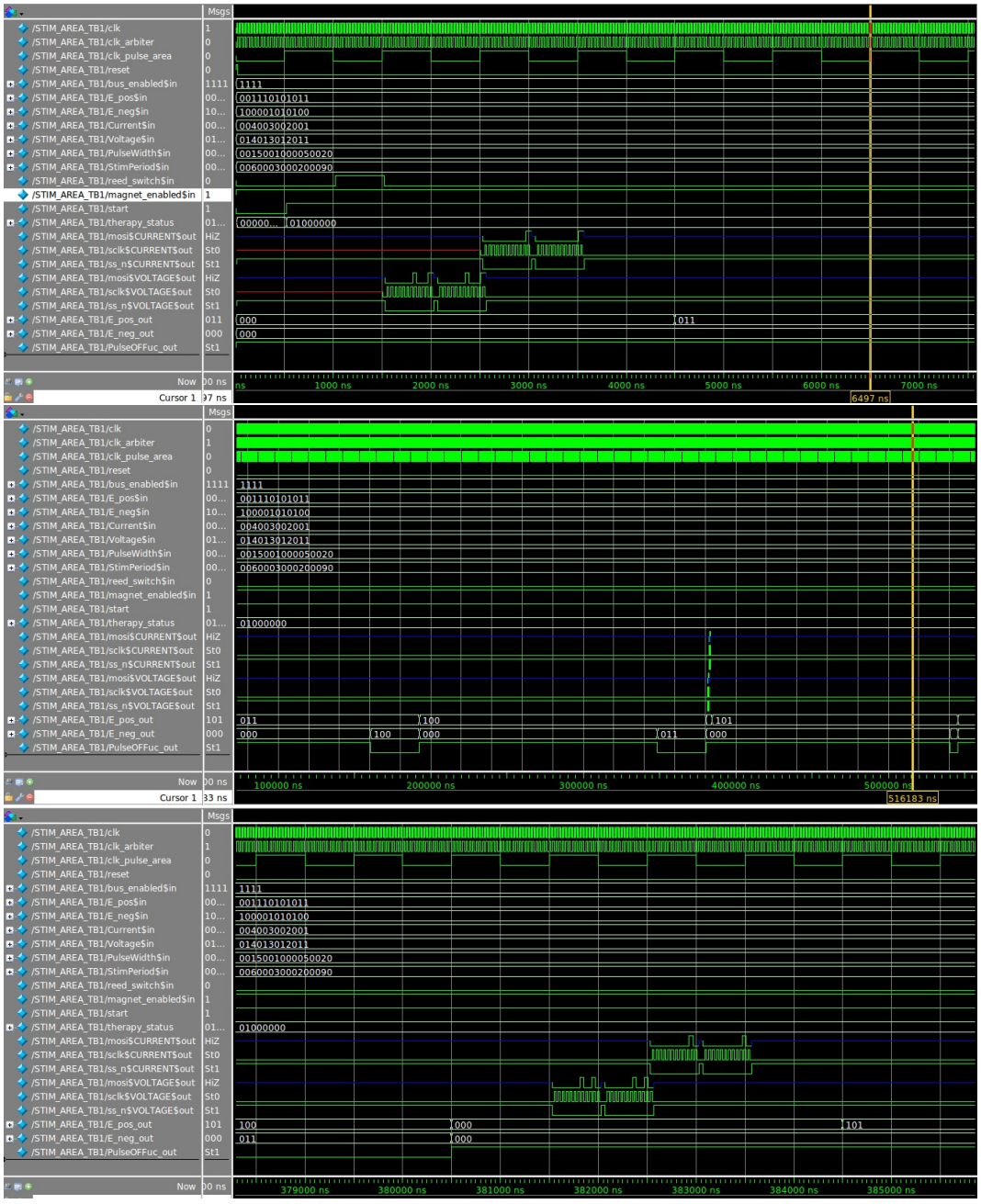

Figura 4.8: Ejecución de 4 terapias en forma simultánea.

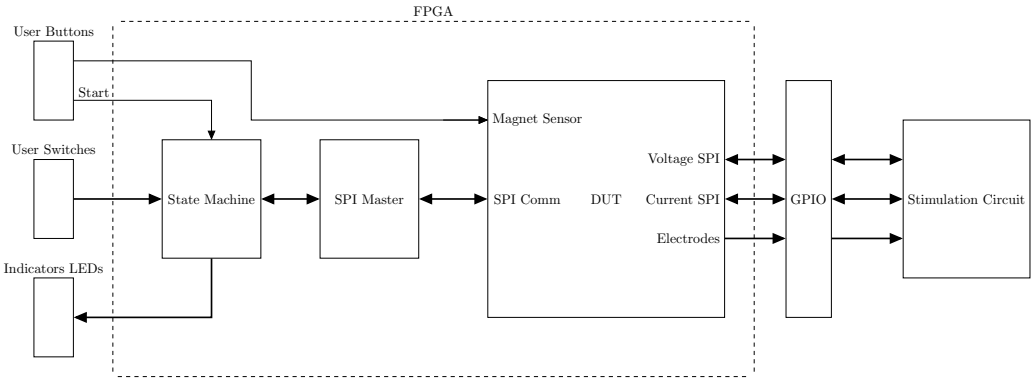

Figura 4.9: Plataforma para la verificación funcional de diseño, la configuración de las terapias y el estado del DUT se maneja y despliega con los recursos del *kit* de desarrollo.

Estimulación con 4 terapias y 16 electrodos

Para la verificación funcional se sintetizó el diseño con 4 terapias y 16 electrodos. Su configuración fue la siguiente:

- Terapia 1:
  - Habilitación: habilitada.
  - Amplitud: 1.0mA.
  - Frecuencia: 2Hz.
  - Ancho de pulso: 1000μs.
  - Electrodos: E_2 (+), E_1 (−).
- Terapia 2:
  - Habilitación: habilitada.
  - Amplitud: 0.5mA.
  - Frecuencia: 10Hz.
  - Ancho de pulso: 1000μs.
  - Electrodos: E_2 (+), E_1 (−).
- Terapia 3:
  - Habilitación: habilitada.
  - Amplitud: 0.2mA.
  - Frecuencia: 10kHz.
  - Ancho de pulso: 30μs.

- Electrodos: E_2 (+), E_1 (−).

- Terapia 4:
  - Habilitación: no habilitada.
  - Amplitud: 0.2mA.
  - Frecuencia: 1kHz.
  - Ancho de pulso: 30μs.
  - Electrodos: E_2 (+), E_1 (−).

En las figuras 4.10, 4.11 y 4.12 se pueden ver los resultados de la ejecución. Particularmente, en la figura 4.11 se pueden observar los tres tipos de pulsos de estimulación entregados. Cuando se entregan los pulsos de baja frecuencia se observa una subida lenta en la amplitud. Esto se debe al tiempo de establecimiento de los DAC utilizados, que rondan los 660μs [27].

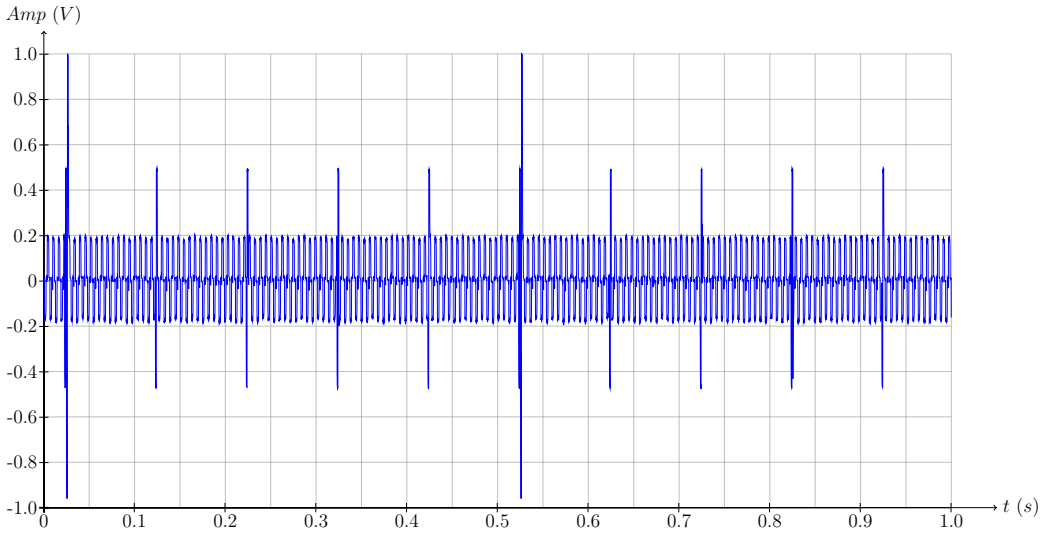

Figura 4.10: Estimulación con tres terapias habilitadas. Cada una de ellas fue configurada para entregar los pulsos de estimulación en los mismo electrodos para poder observar el solapamiento de las terapias fácilmente. Para la verificación se utilizó una red de 16 resistencias en estrella de 500Ω.

Finalmente, en la figura 4.13 se observa el envío de pulsos de estimulación con la siguiente configuración de terapias:

- Terapia 1:
  - Habilitación: habilitada.
  - Amplitud: 4.0mA.

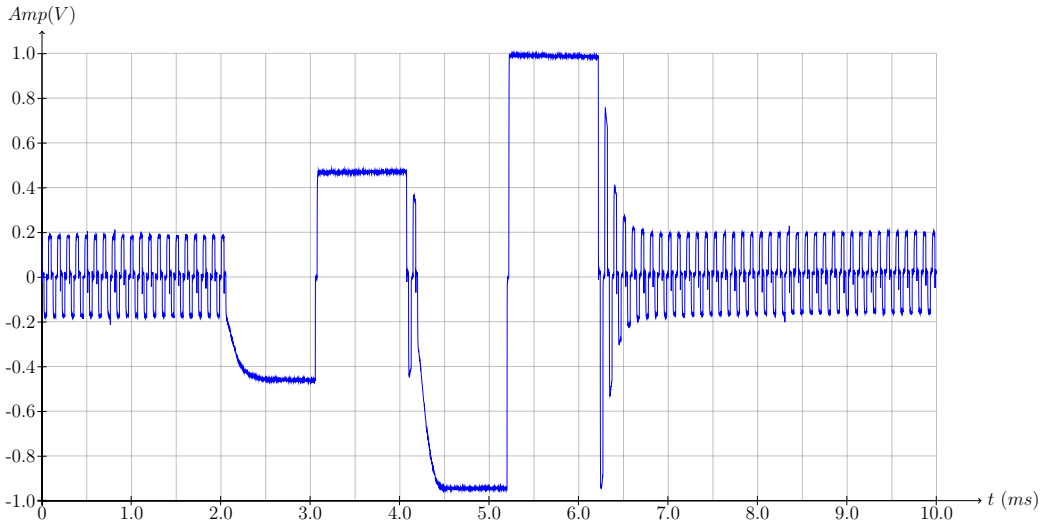

Figura 4.11: Captura con menor ventana de tiempo. Se puede observar como las terapias de frecuencias lentas son entregadas con mayor prioridad que la de alta frecuencia.

- Frecuencia: 10kHz.
- Ancho de pulso: 30μs.
- Electrodos: E_9 (+), E_1 (−).

- Terapia 2:
  - Habilitación: no habilitada.
- Terapia 3:
  - Habilitación: no habilitada.
- Terapia 4:
  - Habilitación: no habilitada.

Esta es la configuración que se utilizará para evaluar el consumo en todas las plataformas utilizadas.

4.3.4. Verificación en otras plataformas

El diseño fue verificado a nivel funcional en los *kit* de desarrollo utilizadas en esta tesis. La evaluación fue realizada verificando que las señales de salida (entradas al circuito de estimulación) sea comporten adecuadamente.

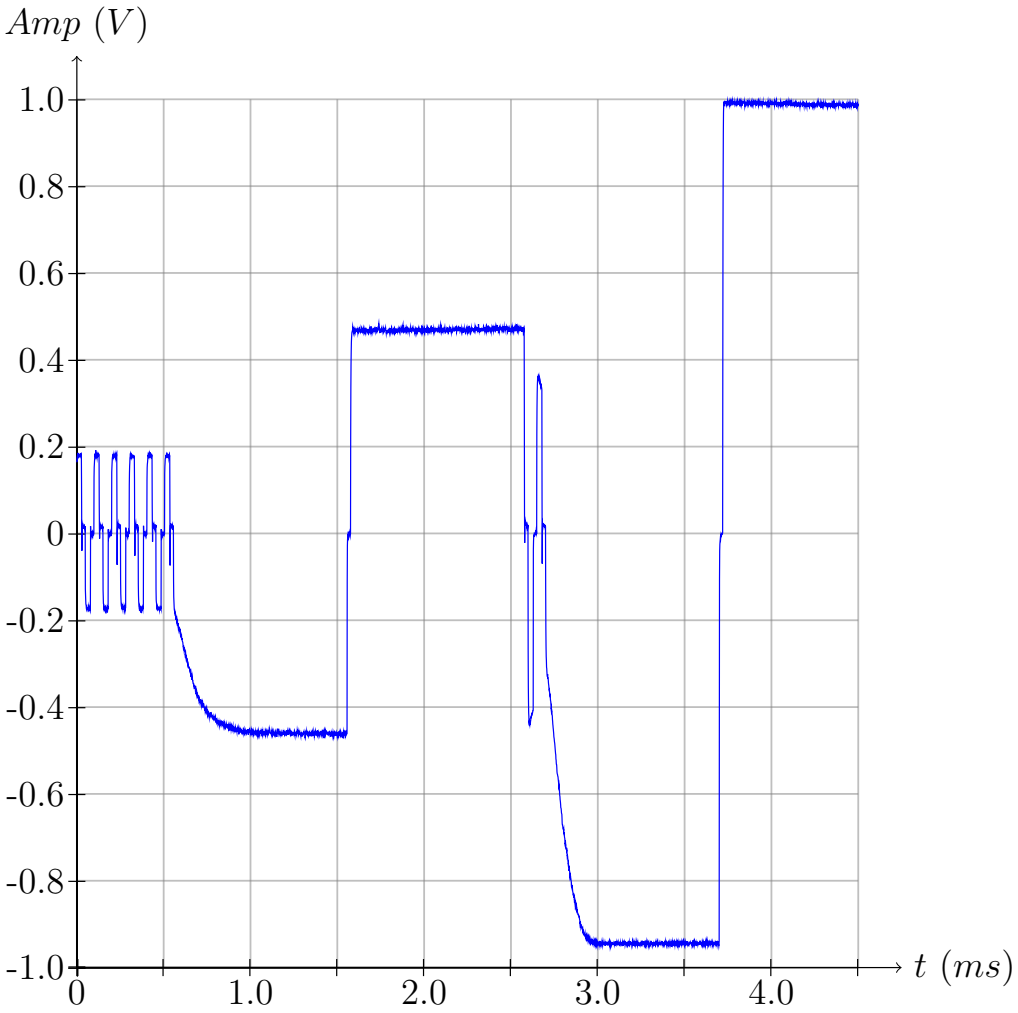

Figura 4.12: Captura con una ventana de 5ms entorno a la entrega de los pulsos de estimulación asociados a la terapia de 2Hz. En ella puede observarse que entre el pulso de 10Hz y el de 2Hz hay tiempo suficiente para poder entregar un pulso de la terapia de 10kHz.

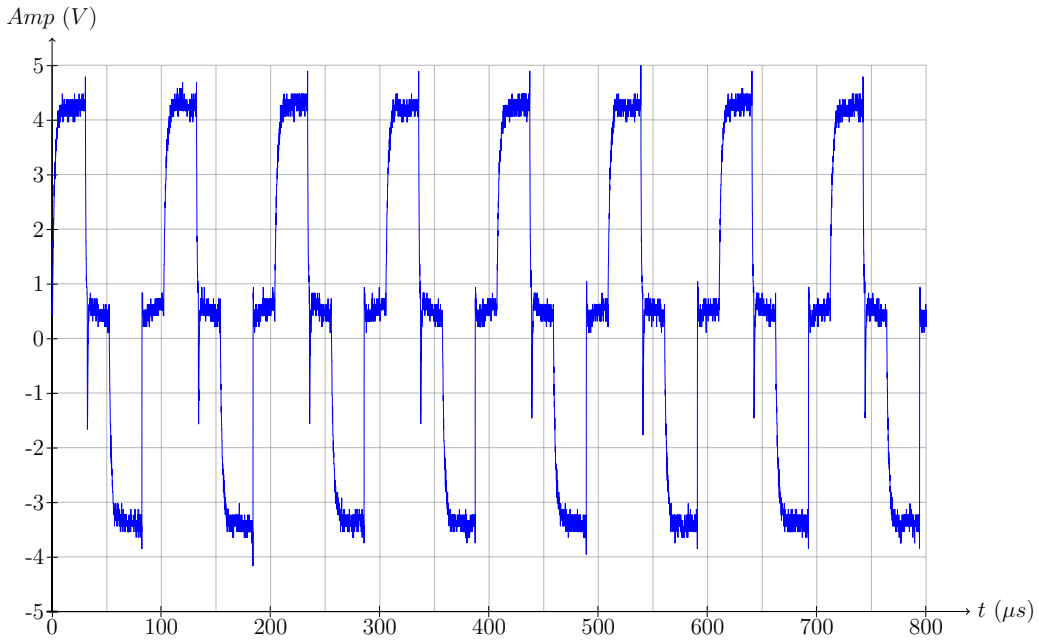

Figura 4.13: Una única terapia de 10kHz, 4.0mA y 30μs de ancho de pulso.

4.3.5. Tamaño del diseño en cada plataforma

El diseño (con 4 terapias y 16 electrodos) fue sintetizado para cada plataforma (con una modificación despreciable que se encuentra descrita en la sección 5.1). En la tabla 4.1 se puede observar el tamaño resultante del diseño y su respectivo porcentaje de ocupación.

	Recursos necesarios	Ocupación (%)
Cyclone V - 5CEBA4	776 ALM	4,2
IGLOO2 - M2GL025	3643 elementos lógicos	13,2
iCE40 - HX-8K	3840 elementos lógicos	50,0

Tabla 4.1: Tamaño del diseño y porcentaje de ocupación reportados por cada herramienta de desarrollo.

Capítulo 5

Consumo de potencia del diseño en varias plataformas

5.1. Algunas consideraciones y adaptación del diseño para la medida de consumo de potencia

En un uso normal, el sistema recibirá comandos a una tasa mucho menor que la frecuencia de estimulación. Esto implica que el consumo dinámico promedio de la interfaz SPI será despreciable respecto al consumo dinámico del resto del circuito. En otras palabras, esencialmente, la interfaz SPI aportará solo al consumo estático. Esta consideración simplifica mucho el mecanismo para la medición del consumo, particularmente del consumo total, ya que no sería necesario ejercitar periódicamente la interfaz de comunicación. Por este motivo, se agregó una señal adicional de entrada ("force") que desencadena el inicio de la estimulación utilizando los parámetros que se encuentren programados (ver figura 5.1).

El diseño fue sintetizado con la siguiente configuración:

- 4 terapias
- 16 electrodos
- frecuencia de reloj: 7,8125MHz

El valor por defecto de los parámetros programables (los que fueron utilizados para todas las medidas de consumo) son:

- una terapia habilitada
- frecuencia de estimulación: 10kHz
- ancho de pulso: 30μs
- amplitud: 4mA
- tensión de la fuente programable: 16V
- electrodos: E_1 (+), E_9 (−).

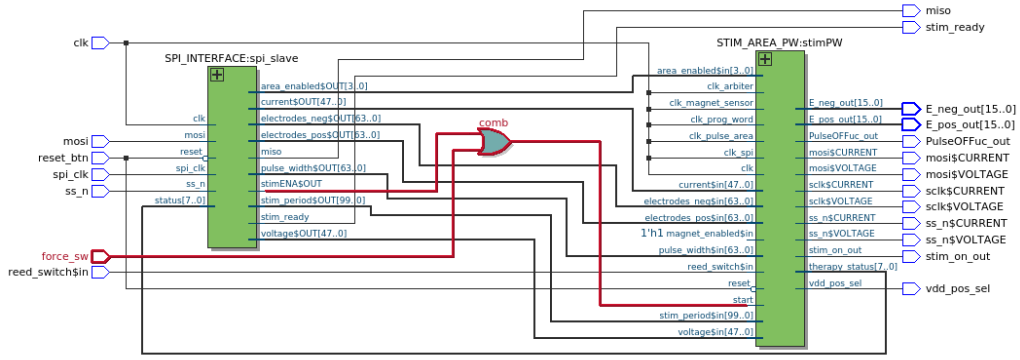

Figura 5.1: Adaptación del diseño para agregar una señal de entrada que desencadene la estimulación.

La frecuencia de reloj se estableció en las plataformas Cyclone V e IGLOO2 con un PLL ajustado a la frecuencia más cercana a 7,8125MHz. En el caso de la FPGA iCE40, se ajustó un PLL a 15,625MHz y luego se dividió con un divisor de frecuencias de 1 bit. En la tabla 5.1 se pueden ver las frecuencias de reloj obtenidas en cada plataforma.

Cyclone V - 5CEBA4	IGLOO2 - M2GL025	iCE40 - HX-8K
7,8125MHz	7,812MHz	7,8125MHz

Tabla 5.1: Frecuencias de reloj para las medidas de consumo en cada plataforma.

5.2. Resultados para cada plataforma

Las medidas de consumo fueron realizadas con el método de medida descrito en 3.3 (una resistencia de sensado en serie con la alimentación del *core* de

la FPGA y con todas las salidas del diseño conectadas a pines de la FPGA). Exceptuando las señales de reloj y *reset*, las entradas del diseño bajo prueba son las del esclavo SPI por donde se reciben los parámetros del pulso de estimulación (y sus habilitaciones). Con las consideraciones descritas en 5.1, las entradas de la interfaz SPI fueron bloqueadas y se forzó la estimulación con la señal "force". Se utilizó una resistencia de sensado de 2,1Ω y su diferencia de potencial y promedio fueron registrados y calculados con un osciloscopio en una captura de 10 segundos de duración. Todas las medidas fueron tomadas dentro de un horno a una temperatura controlada de 37°C[1]. La frecuencia de reloj fue próxima a 7,812MHz. Los resultados se pueden apreciar en la tabla 5.2 y gráficamente en la figura 5.2.

	Total (mW)	Estático (mW)	Dinámico (mW)
Cyclone V	43,86	40,14	3,72
IGLOO2 - M2GL025	20,06	16,22	3,84
iCE40 - HX-8K	3,62	2,24	1,38

Tabla 5.2: Consumos estáticos, dinámicos y totales del diseño bajo prueba en las tres plataformas consideradas

Como se aprecia en los resultados obtenidos, la FPGA de la familia iCE40 es la que muestra un menor consumo de potencia, tanto estático como dinámico.

Las FPGA de la familia IGLOO2 y Cyclone V poseen similar consumo dinámico, mientras que la segunda posee un consumo estático considerablemente mayor.

Un resultado interesante es que la FPGA con menor capacidad (es decir, menor cantidad de elementos lógicos) es la que posee menor consumo estático. Como fue descrito en la sección 2.1 las FPGA contienen un alto número de transistores dedicados a elementos lógicos, interconexiones programables y enrutadores programables. Naturalmente, a mayor cantidad de elementos lógicos, mayor cantidad de recursos dedicados a las interconexiones entre bloques. Si bien las FPGA de la familia IGLOO2 reducen el consumo estático mediante el uso de celdas de FLASH, la FPGA utilizada posee 3,6 veces más elementos

[1]Naturalmente, la temperatura de operación será próxima a los 37°C por ser un dispositivo implantable. Los estándares internacionales exigen una temperatura ambiente de 37°C +/− 2°C como condición para la verificación de algunos requerimientos [7] (donde las condiciones ambientales resultan pertinentes). Dado que el consumo estático depende de la temperatura de operación (a mayor temperatura, mayor consumo), parece sensato realizar las medidas de consumo de este diseño a la temperatura normal del dispositivo ya implantado.

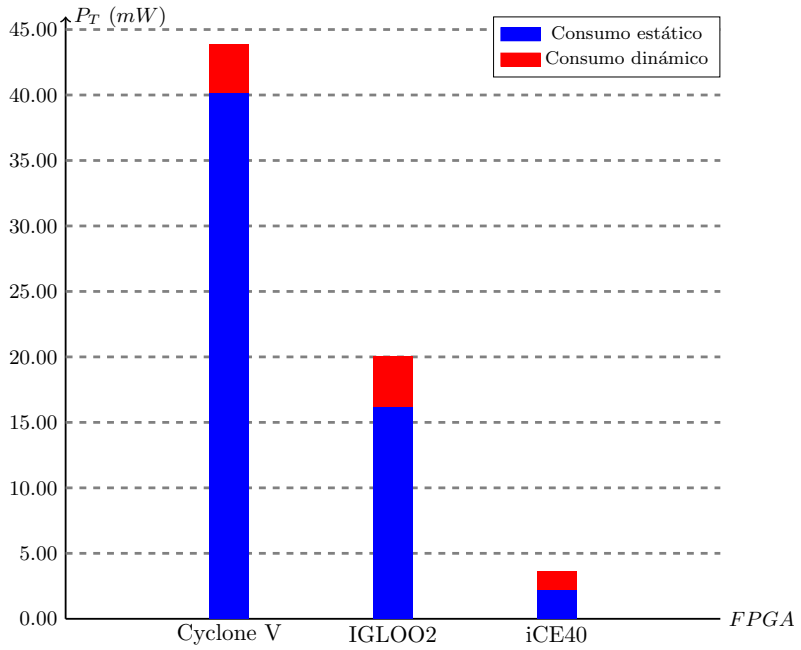

Figura 5.2: Consumo total del sistema implementado en cada plataforma. En azul puede apreciarse el aporte del consumo estático mientras que en rojo puede apreciarse el del consumo dinámico. Nótese el gran peso relativo del consumo estático en todos los casos.

lógicos que la de familia iCE40. El peso que tiene la cantidad de recursos de la FPGA parece ser tal, que resulta tener mayor influencia que el uso de celdas de FLASH para almacenar la configuración y que la tecnología de fabricación (130nm en la familia IGLOO2 contra 40nm en la familia iCE40). De este modo, la capacidad de la FPGA parece posicionarse como variable fundamental en lo que respecta al consumo estático de potencia. Reducir el tamaño de una FPGA al mínimo necesario dará el menor consumo estático posible (y el mayor porcentaje de utilización).

5.3. Resultados y sus estimaciones

Para cada una de las plataformas, los resultados medidos fueron comparados contra las hojas de datos de cada FPGA y con las herramientas de estimación de consumo. En lo que respecta al consumo dinámico, las medidas fueron comparadas contra los resultados de las herramientas de estimación solamente, ya que no es posible inferir datos significativos mediante las hojas de

datos.

5.3.1. Comparaciones para la FPGA de la familia Cyclone V

En la hoja de datos de la familia Cyclone V ([29], sección *DC Characteristics*) se sugiere que se utilice la herramienta *Early Power Estimator (EPE)* [30] para determinar el consumo estático. Básicamente EPE es una hoja de cálculo que permite estimar el consumo del diseño en forma temprana. Con el diseño finalizado se decidió no realizar esta etapa y utilizar directamente la herramienta de estimación *PowerPlay* (de Quartus2, versión: 15.0.2). Los resultados se presentan en la tabla 5.3 y en la figura 5.3.

	Estático (mW)	Dinámico (mW)	Error Estático	Error Dinámico
Medido	40,14	3,72	N/A	N/A
Herramienta de estimación	199,75	4,74	+397,6 %	+27,4 %

Tabla 5.3: Estimaciones y medidas de consumo para la FPGA de la familia Cyclone V

5.3.2. Comparaciones para la FPGA de la familia IGLOO2

El consumo estático según la hoja de datos ([31], en la tabla "*SmartFusion2 and IGLOO2 Quiescent Supply Current (VDD = 1.2 V) – Typical Process*") se encuentra el resultado medido a 25°C y 85°C y con la configuración borrada. La herramienta de estimación fue *Smart Power* (versión: 11.8.3.6). Los resultados se presentan en la tabla 5.4 y en la figura 5.4.

	Estático (mW)	Dinámico (mW)	Error Estático	Error Dinámico
Medido	16,22	3,84	N/A	N/A
Hoja de datos a 25°C	10,68	N/A	-34,2 %	N/A
Hoja de datos a 85°C	48,72	N/A	+200,4 %	N/A
Herramienta de estimación	15,01	4,68	-7,5 %	+21,9 %

Tabla 5.4: Estimaciones y medidas de consumo para la FPGA M2GL025

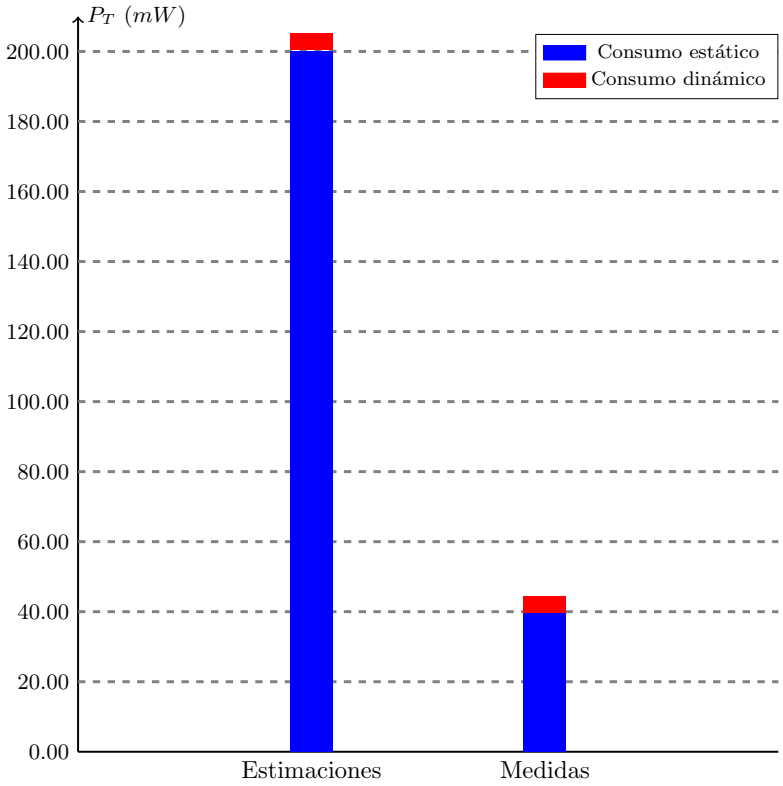

Figura 5.3: Estimaciones y medidas de consumo para la FPGA de la familia Cyclone V

5.3.3. Comparaciones para la FPGA de la familia iCE40

El consumo estático según la hoja de datos ([32], sección "*DC and Switching Characteristics*") es el medido a 25°C y con la memoria de configuración borrada. La herramienta de estimación fue *Power Estimator*, provista por el entorno de desarrollo *Lattice iCEcube* (release: 2017.08.27940). Los resultados se presentan en la tabla 5.5 y en la figura 5.5.

	Estático (mW)	Dinámico (mW)	Error Estático	Error Dinámico
Medido	2,24	1,38	N/A	N/A
Hoja de datos	1,37	N/A	-38,8 %	N/A
Herramienta de estimación	2,29	5,65	+2,2 %	+309,4 %

Tabla 5.5: Estimaciones y medidas de consumo de potencia para la FPGA iCE40 - HX-8K.

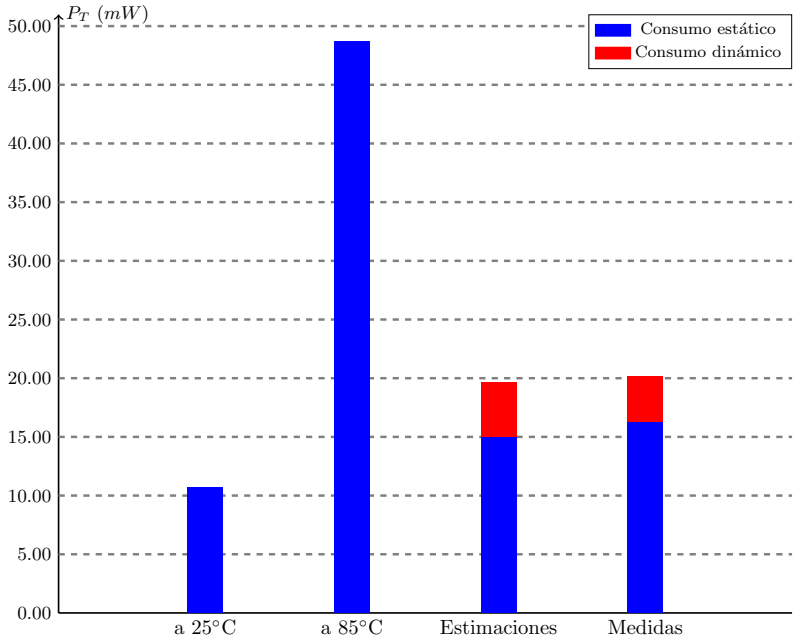

Figura 5.4: Estimaciones y medidas de consumo para la FPGA M2GL025

5.4. Determinación de una mejor plataforma con la información obtenida

Debido al uso de *kits* de desarrollo, las FPGA utilizadas no son las que mejor se ajustan al diseño realizado. Algunas tienen un tamaño excesivamente grande para esta aplicación, lo que implica un consumo estático superior al que se podría conseguir con una FPGA del tamaño adecuado.

En esta sección se determinará, para cada familia utilizada, el tamaño de un FPGA teórica con capacidad de almacenar una única instancia del diseño bajo prueba y que maximice su nivel de utilización. Conjuntamente, se calculará el consumo estático y total del diseño en esta FPGA.

Con el estudio antes comentado, se determinarán los requerimientos mínimos en cuanto a tamaño y cantidad de pines de una FPGA para este diseño. Para cada una de las tres familias utilizadas, se elegirá la FPGA óptima existente en el mercado.

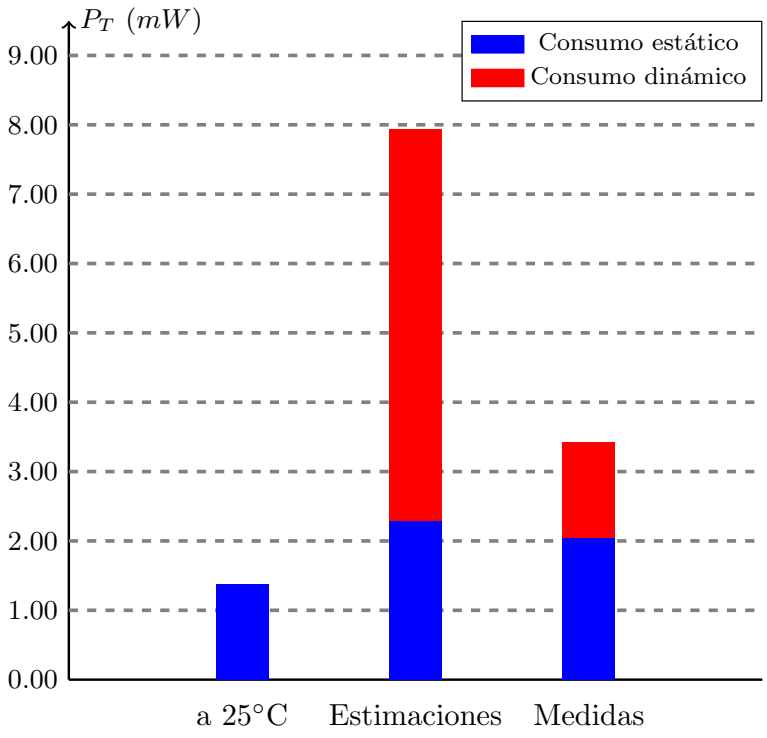

Figura 5.5: Estimaciones y medidas de consumo de potencia para la FPGA iCE40 - HX-8K.

5.4.1. El consumo estático de potencia como variable fundamental

Como puede verse tanto en la tabla 5.2 como en la figura 5.2, el consumo estático es el tipo de consumo mayoritario en todos los ensayos. En la tabla 5.6 se ve que el aporte del consumo estático en el total, varía entre 61,9 % y 91,5 %.

	Total (mW)	Estático (%)	Dinámico (%)	Utilización de la FPGA (%)
Cyclone V	43,86	91,5	8,5	4,2
IGLOO2 - M2GL025	20,06	80,9	9,1	13,2
iCE40 - HX-8K	3,62	61,9	38,1	50,0

Tabla 5.6: Peso relativo del consumo estático y dinámico en cada plataforma

Observando los niveles de utilización resultantes, se podría utilizar una

FPGA de menor tamaño y alcanzar un menor consumo estático (y por ende total). Si para cada tecnología utilizada (Cyclone V, IGLOO2 e iCE40) se utilizara la FPGA con la cantidad de elementos lógicos "mínima necesaria" para el diseño en cuestión, se tendría el máximo nivel de utilización y, por sobretodo, un consumo estático menor (de hecho el menor posible).

5.4.2. Determinación del tamaño mínimo de una FPGA para el diseño propuesto

Cálculo basado en los reportes de los entornos de desarrollo

Los reportes de las herramientas de desarrollo nos dan información relativa a, entre otras cosas, la cantidad de elementos lógicos que han sido utilizados por los diseños. Para el diseño propuesto y las FPGA utilizadas, los resultados se encuentran en la tabla 5.7. Estos datos dan una buena aproximación al tamaño mínimo que necesita una FPGA, para alojar el diseño propuesto.

	Recursos necesarios
Cyclone V - 5CEBA4	776 ALM
IGLOO2 - M2GL025	3643 elementos lógicos
iCE40 - HX-8K	3840 elementos lógicos

Tabla 5.7: Cantidad de recursos necesarios para cada familia utilizada.

Como fue mostrado anteriormente, el nivel de utilización de recursos de las FPGA ensayadas resultó en valores muy bajos. Si se piensa en una FPGA teórica que tuviera el tamaño exacto para el diseño utilizado (los tamaños de la tabla 5.7), el nivel de utilización de recursos sería máximo. Esto implicaría un esfuerzo mucho mayor de las herramientas de *place and route*, ya que no habría "margen para el error". En casos así no sería posible afirmar que efectivamente el diseño "entre" en una FPGA tan justa de recursos.

Básicamente, el método anterior (utilizar la cantidad de recursos mostrada en los reportes) asume que dada una misma familia de FPGA, la cantidad de recursos necesarios para un diseño es independiente del nivel de utilización de la FPGA (de algún modo es decir que es independiente de las herramientas de *place and route*). Con este enfoque, se podría afirmar que la cantidad de instancias del DUT que es posible generar dentro de la FPGA Cyclone V, 5CEBA4 es:

$$\frac{776 ALM}{18480 ALM} = 23 \; instancias$$

Otro método, sería el de generar la mayor cantidad posible de instancias del DUT en la FPGA. Esto permitiría evaluar la *performance* de las herramientas de desarrollo, en un escenario con un alto nivel de utilización. Con este otro enfoque, el tamaño mínimo de la FPGA se determinaría asignando proporcionalmente los recursos de la FPGA ensayada entre la cantidad de instancias del DUT que fueron generadas.

En la siguiente sección se desarrolla este análisis para las tres FPGA utilizadas.

Cálculo basado en la cantidad de instancias sintetizables dentro de una FPGA

Como fue comentado anteriormente, para determinar el tamaño de una FPGA teórica capaz de ubicar una sola instancia del diseño, se realizó un ensayo que consistió en determinar cuántas instancias del DUT es posible ubicar dentro de cada FPGA utilizada. Para evitar simplificaciones, el diseño posee la misma arquitectura presentada en la figura 3.4 (instancias del DUT conectados a un pin de salida mediante una red de compuertas "XOR"). Básicamente, el ensayo se ejecutó en dos pasos:

1. Se determinó la cantidad máxima de los DUT que es posible sintetizar y ubicar en cada FPGA (N_{DUT}). Para realizar esto se tomó la misma estrategia presentada en 3.4.1, ubicando todas las salidas de cada instancia del DUT en una red de compuertas "XOR" (esto resulta práctico y a la vez necesario, ya que no sería posible asignar todas las señales de entrada y salida a pines de entrada/salida de las FPGA). Adicionalmente, se llevó a cabo un proceso iterativo tratando de determinar la cantidad máxima de instancias. Este es un proceso lento, ya que los tiempos de *placement* y *routing* aumentan mucho (del orden de las horas) cuando el nivel de utilización es alto (en [33] se describen muy en detalle estos procesos y se formulan heurísticas nuevas para reducir estos tiempos). Si bien la red auxiliar consume cierta cantidad de elementos lógicos, en este cálculo se despreció este número respecto al total requerido.
2. Habiendo determinado este valor (N_{DUT}), se procede a calcular el tamaño mínimo de una FPGA de la misma familia que la ensayada (S_0).

El tamaño mínimo de una FPGA para poder ubicar apropiadamente un DUT (S_0), se puede calcular bajo la hipótesis de que cada instancia del DUT

insume la misma cantidad de recursos, del siguiente modo:

$$S_0 = \frac{S_{FPGA}}{N_{DUT}}$$

donde S_{FPGA} es el tamaño de la FPGA utilizada para el ensayo.

Este análisis determina, para cada familia ensayada, el tamaño de una FPGA teórica en la que se puede sintetizar una única instancia del DUT. Esta FPGA sería la "más pequeña" capaz de utilizarse para este diseño.

El método descrito se ejecutó en las tres FPGA disponibles. La cantidad de instancias sintetizables en cada caso se presenta en la tabla 5.8. También se encuentra allí el nivel de utilización alcanzado. El tamaño mínimo para cada familia se presenta en la tabla 5.9.

FPGA	N_{DUT}	Utilización (%)
Cyclone V: 5CEBA4	24	93,3
IGLOO2: M2GL025	6	66,3
iCE40: iCE40 - HX-8K	1	50,0

Tabla 5.8: Resultados del ensayo que trata de encontrar la cantidad máxima de instancias del DUT que pueden ser sintetizadas en cada FPGA. En la FPGA iCE40-HX-8K el diseño utilizaba el 50 % de los recursos de la misma, por lo cual no sería posible obtener dos instancias del DUT. Lo más llamativo es el bajo nivel de utilización alcanzado en la FPGA M2GL025, en el que se "desperdicia" más el 30 % de los recursos.

Familia: Cyclone V			
FPGA	N_{DUT}	S_{FPGA} (en bloques ALM)	S_0 (en bloques ALM)
5CEBA4	24	18480	770
Familia: IGLOO2			
FPGA	N_{DUT}	S_{FPGA} (en elementos lógicos)	S_0 (en elementos lógicos)
M2GL025	6	27696	4616
Familia: iCE40			
FPGA	N_{DUT}	S_{FPGA} (en elementos lógicos)	S_0 (en elementos lógicos)
iCE40-HX-8K	1	7680	7680

Tabla 5.9: Determinación del tamaño mínimo (S_0) para cada FPGA. La operación asigna, para cada DUT, el tamaño total de la FPGA, equitativamente. Como corolario, a cada DUT se le asigna una porción de los elementos lógicos utilizados y también una porción de los no utilizados.

Comparación de los métodos de cálculo

La tabla 5.10 muestra la comparación de los resultados obtenidos para los tamaños mínimos de una FPGA teórica (que es capaz de almacenar una única instancia del DUT), para cada método de cálculo.

FPGA	Según reportes	S_0 (según cantidad de instancias)
5CEBA4	776 bloques ALM	770 bloques ALM
M2GL025	3643 elementos lógicos	4616 elementos lógicos
iCE40-HX-8K	3840 bloques lógicos	7680 bloques lógicos

Tabla 5.10: Tamaño mínimo de una FPGA (para cada familia ensayada) para poder sintetizar el DUT apropiadamente. Los resultados presentados han sido calculados según los reportes de las herramientas de desarrollo y según un ensayo que trata de "llenar" las FPGA de instancias del DUT.

Los resultados obtenidos para las FPGA M2GL025 y iCE40-HX-8K son consistentes con las suposiciones tomadas, es decir, que el cálculo mediante el "llenado" de las FPGA resultó en una mayor cantidad de elementos lógicos. Esto puede atribuirse a dos factores. Por un lado, la red auxiliar de compuertas "XOR" tiene requiere de una cantidad de elementos lógicos que aumenta con la cantidad de instancias que se sintetizan, esto puede resultar es una sobre-estimación del tamaño requerido. Por otro lado, las herramientas de *place and route* pueden no ser tan eficientes cuando el tamaño de los diseños se aproxima al tamaño de la FPGA de destino. En el caso de la FPGA iCE40-HX-8K se da la mayor distancia entre los resultados, ya que el diseño ocupa exactamente la mitad de la capacidad de la FPGA.

En el caso de la FPGA 5CEBA4, una única instancia del diseño ocupa una cantidad de elementos lógicos apenas mayor que la calculada al completar la FPGA con los DUT. Esto está fuertemente relacionado a las herramientas de *place and route* conjuntamente con los bloques ALM utilizados, que por su flexibilidad pueden ser fraccionados para aprovechar mejor el hardware.

El resultado más importante está relacionado con el nivel de utilización conseguido con la FPGA M2GL025 al realizar el ensayo, ya que sólo se pudo utilizar el 66,3 % de sus elementos lógicos. Según los reportes, en esta FPGA "cabrían" casi 8 instancias del diseño. Lo que este resultado puede significar, es que para ciertos diseños, las herramientas de *place and route* disponibles puede desperdiciar muchos elementos lógicos, impidiendo obtener un alto nivel de utilización. Esto es importante, ya que implica que deba ser utilizada una

FPGA de un tamaño, por ejemplo, 30 % mayor al necesario. Los diseñadores suelen dejar libre un porcentaje generoso de elementos lógicos para correcciones futuras o modificaciones menores. De todas maneras este espacio no sería utilizable, debiendo sacrificar aún más área y consumo estático.

A priori, no parece sencillo decir cuál de los dos resultados es más preciso. Por un lado, asumir que el resultado del reporte (obtenido de la síntesis del diseño en una FPGA de tamaño mucho mayor al necesario) es muy preciso, tiene el inconveniente de que si las herramientas provistas por los fabricantes no fueran eficientes, una FPGA del tamaño "justo" al diseño podría no alcanzar. Por otro lado, el "llenado" de una FPGA con instancias del diseño tiene el inconveniente del uso de una red auxiliar (para evitar simplificaciones de diseño) que tiene un costo en tamaño. Una método sencillo y rápido para obtener respuestas a esta interrogante es simplemente realizar el proceso completo de síntesis y *place and route* para la FPGA candidata. Si este proceso falla, se procede con la siguiente FPGA en tamaño.

5.4.3. Mínimo consumo de potencia alcanzable con las familias utilizadas

El consumo estático mínimo que se podría alcanzar para cada familia (y con este diseño) será el consumo estático que tendría la FPGA teórica calculada previamente. Este valor podría aproximarse según la relación de tamaño entre las FPGA utilizadas y el tamaño "mínimo". Básicamente, la aproximación consistiría en el escalado del consumo estático de las FPGA utilizadas según la relación de tamaño entre cada una de estas y la FPGA mínima calculada.

Para realizar este cálculo se tomó como tamaño de la FPGA al obtenido a través del ensayo anterior. Esta decisión fue tomada con la intención de que la FPGA se encuentre completa de instancias del DUT al momento de medir su consumo, lo cual parece un escenario que mejor se ajustaría al escenario teórico (una instancia del DUT que ocupa la mayoría del tamaño de la FPGA). Adicionalmente, los resultados obtenidos mostraron que cierta cantidad de elementos lógicos puede ser desperdiciada por las herramientas de desarrollo y por lo tanto el tamaño teórico debería contemplar este echo.

Si se le llama P_S^N al consumo estático de cada FPGA cuando se sintetizan en ella la cantidad máxima de DUT (N_{DUT}), es posible determinar el consumo

deseado, del siguiente modo:

$$P_S = \frac{P_S^N}{N_{DUT}}$$

Asumiendo que el consumo dinámico del diseño no cambia, se puede calcular el consumo total en este escenario teórico.

Los ensayos descritos fueron ejecutados para las tres FPGA disponibles. Las medidas realizadas y los cálculos se presentan en la tabla 5.11 y gráficamente en la figura 5.6.

	N_{DUT}	$P_S^N\ (mW)$	$P_s\ (mW)$	$P_T(mW)$
Cyclone V - 5CEBA4	24	42,50	1,77	5,49
IGLOO2 - M2GL025	6	16,47	2,75	6,58
iCE40 - HX-8K	1	2,24	2,24	3,62

Tabla 5.11: Determinación del consumo promedio estático (P_s) y total (P_T) para cada una de las plataformas teóricas. El proceso para calcular el consumo estático mínimo consiste en generar la mayor cantidad de instancias del diseño bajo prueba posibles. Una vez obtenido este número (N_{DUT}), se mide su consumo estático P_S^N. Con estos datos es posible obtener un cálculo razonable. Debe notarse que el consumo para la FPGA iCE40 es el medido directamente, ya que sólo se puede generar una única instancia del DUT.

Este mecanismo de cálculo es muy interesante, ya que condensa diversas variables en forma muy simple. Una variable como la tecnología de fabricación (ver 3.2.4) tiene una incidencia muy evidente, ya que un mayor consumo estático se traduce en un mayor consumo calculado (recordar que la familia IGLOO2 tiene un proceso de fabricación basado en FLASH de 130nm, mientas que la familia Cyclone V tiene un proceso basado en SRAM de 28nm).

Una variable como el entorno de desarrollo (en particular las herramientas de *place and route*) también tienen una aporte capital. En esta línea, posiblemente el resultado más destacado es que la mejor plataforma "mínima" con tecnología igual a una Cyclone V tendría un consumo estático sensiblemente menor (de un 30,1 %) a su contraparte con tecnología IGLOO2.

Esta diferencia tan grande se debe principalmente a dos factores. Por un lado, a la gran eficiencia de las herramientas de *place and route* de Quartus2 (v15.0.2), ya que el nivel de utilización de los recursos de la Cyclone V fue de un 93,3 %, mientras que para el caso de la IGLOO2 (utilizando el entorno LiberoSoC, version 11 SP3) fue de un 66,3 %. Por otro lado, la arquitectura interna de las FPGA pueden resultar en una ventaja para la familia Cyclone

V, que utiliza ALM como bloques lógicos, dándole flexibilidad y mayor margen para la optimización.

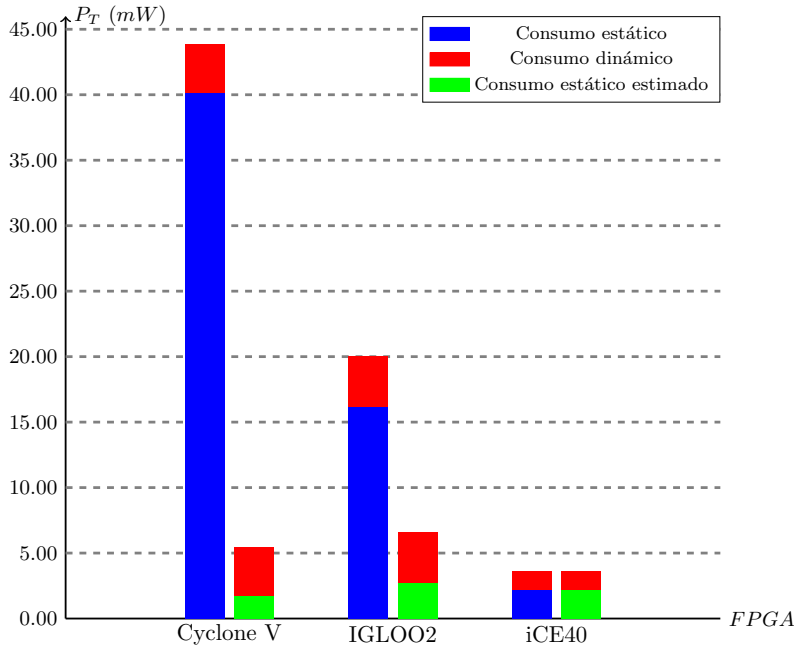

Figura 5.6: Determinación del mínimo consumo estático (y el correspondiente consumo total) alcanzable por ajuste del tamaño de la FPGA (al mínimo necesario) en cada tecnología utilizada. Básicamente, estos consumos serían los obtenidos si se utilizara una FPGA con la cantidad de elementos calculados y presentados en la tabla 5.9 (estos tamaños serían los mínimos requeridos para cada familia)

5.4.4. Requerimientos generales y específicos para cada familia

Los resultados anteriores permiten buscar la FPGA que mejor se ajuste al DUT propuesto, con las familias utilizadas.

Los requerimientos mínimos para cualquier plataforma están dados por la cantidad de pines de entrada/salida (43 en total) y la frecuencia de operación (de 7,812MHz generada externamente o mediante un PLL). El tamaño necesario para el DUT es una característica específica de cada familia y que fue calculado anteriormente.

La tabla 5.12 resumen los requerimientos para cada familia de FPGA.

	Pines I/O	Frecuencia	Tamaño
Cyclone V	43	7,812MHz	770 ALM
IGLOO2	43	7,812MHz	4616 bloques lógicos
iCE40	43	7,812MHz	7680 bloques lógicos

Tabla 5.12: Requerimientos en cuanto a cantidad de pines y elementos lógicos, para cada familia utilizada

Mejor FPGA de la familia Cyclone V

Todas las FPGA fabricadas en la familia Cyclone V están sobredimensionadas para el diseño presente. Las FPGA que mejor se ajustan al diseño son las 5CE-x-A2-xxx que poseen 9434 ALM [34] (que tiene un tamaño de unas 12 veces mayor al necesario).

Se sintetizó y se realizó el *place and route* del mismo diseño del ensayo descrito en 5.4 para la FPGA 5CEBA2F23C8. Se determinó la cantidad de instancias máxima configurable dentro de esta. Con esta información, se calculó su consumo estático. Los resultados son los de la tabla (5.13):

FPGA	N_{DUT}	Utilización (%)	P_s (mW)
5CEBA2F23C8	12	93	21,70

Tabla 5.13: Resultado del proceso de síntesis y *place and route* del diseño presentado en 5.4 para una FPGA 5CE-x-A2-xxx de la familia Cyclone V. La cantidad de instancias del DUT que pueden ser ubicadas en ella es coherente con el método utilizado. La cálculo del consumo estático (P_s) para esta FPGA se realizó en función de la relación entre la cantidad de bloques ALM entre esta FPGA y la ensayada.

La cantidad de instancias obtenidas es consistente con el método de cálculo utilizado, ya que la cantidad teórica de instancias sintetizables en esta FPGA resulta en:

$$N_{DUT}^{5CEBA2F23C8} = \frac{ALM_{5CEBA2F23C8}}{ALM_{5CEBA4F23C7N}} N_{DUT}^{5CEBA4F23C7N} = \frac{9434}{18480} 24 \approx 12,25$$

Mejor FPGA de la familia IGLOO2

Así como las FPGA de la familia Cyclone V de Intel, todas las FPGA de la familia IGLOO2 tiene un tamaño mucho mayor al necesario. Las FPGA más apropiadas para este diseño son las M2GL005, que poseen 6060 elementos lógicos [35].

La síntesis y el *place and route* del DUT para la FPGA M2GL005-VF256

resulta en los valores de la tabla (5.14), allí también se presenta el consumo estático calculado:

FPGA	N_{DUT}	Utilización (%)	P_s (mW)
M2GL005-VF256	1	58,4	3,60

Tabla 5.14: Resultado del proceso de síntesis y *place and route* del diseño presentado en 5.4 para una FPGA M2GL005 de la familia IGLOO2. La cantidad de instancias del DUT que pueden ser ubicadas en ella es coherente con el método utilizado. El cálculo del consumo estático (P_s) para esta FPGA se realizó en función de la relación entre la cantidad de elementos lógicos entre esta FPGA y la ensayada.

Este resultado es consistente con los resultados previstos, ya que no sería posible generar dos instancias del DUT en esta FPGA.

Mejor FPGA de la familia iCE40

El ensayo de la tabla 5.11 se muestra que la FPGA utilizada (iCE40-HX-8K) es mínima en el sentido descrito previamente. La cantidad de elementos lógicos que requiere el diseño es de 3840 [36]. Existen FPGA de la familia iCE40 con un volumen lógico aproximado de 4000 (código “4k”). De todos modos ninguna supera el número mínimo de elementos lógicos necesarios. A pesar de esto, se intentó implementar el diseño en la FPGA iCE40-Ultra (de 3520 elementos lógicos), proceso que falló estrepitosamente.

FPGA	N_{DUT}	Utilización (%)	P_s (mW)
iCE40-HX-8K	1	50,0	2,24

Tabla 5.15: Mejor plataforma de la familia iCE40. Los resultado presentados son los de la tabla 5.4 que fueron agregados aquí a modo de resumen.

Capítulo 6

Evaluación de los métodos de medida de consumo dinámico de potencia

6.1. Medidas y resultados

Se sintetizó y se calculó el consumo dinámico de una sola instancia del diseño descrito en el capítulo 4, conectando todas sus salidas a salidas de la FPGA (exactamente del modo en el que el diseño será utilizado).

Los métodos descritos en 3.4 (múltiples instancias del diseño conectadas a un solo pin de salida a través una red de "XOR", mejora del anterior con bloqueo de la red con compuertas "AND" y con bloqueo con multiplexores) también fueron utilizados con el diseño del capítulo 4. Los tres métodos fueron implementados y sintetizados en la FPGA IGLOO2MPGL025 de Microsemi. Se calculó el consumo dinámico del diseño según cada método.

Los resultados se presentan en la tabla 6.1 y en la figura 6.1. Todas las medidas fueron realizadas utilizando un reloj externo de 50MHz y a una temperatura ambiente de 37°C.

Los resultados obtenidos muestran que los consumos obtenidos por los tres métodos distan alrededor de 10 % respecto al calculado por la medida directa del DUT. Los tres métodos dieron resultados que distan menos de $0{,}13mW$ entre sí, mostrando muy poca variabilidad.

Si bien el consumo dinámico real del diseño no se puede conocer con exactitud, es posible concluir que los resultados obtenidos por los métodos utilizados

	Consumo calculado (mW)	Cantidad de instancias para la medida
DUT	6,35	1
XOR	5,69	6
AND	5,78	6
MUX	5,65	5

Tabla 6.1: Consumo dinámico calculado según: medida del consumo total y estático del diseño (DUT), mediante la red auxiliar de compuertas XOR (XOR), mediante el *gating* con compuertas AND (AND) y mediante el uso de multiplexores (MUX). Todas los cálculos fueron utilizando la FPGA de la familia IGLOO2. Adicionalmente, se presenta la cantidad de instancias del DUT que se utilizaron en cada método.

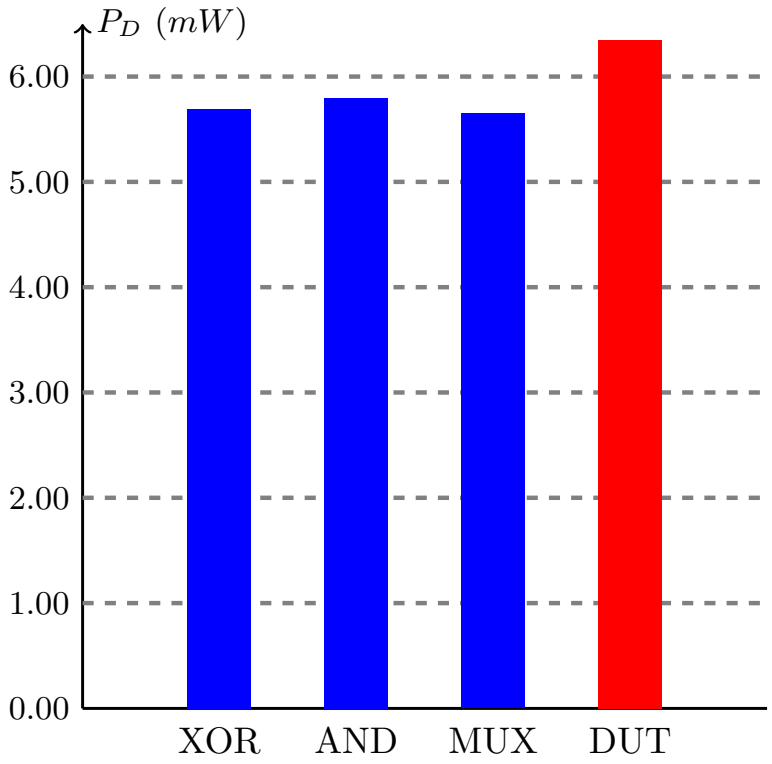

Figura 6.1: Cálculo del consumo dinámico del diseño bajo prueba mediante los tres métodos descritos. Los tres primeros resultados (en azul) son los resultados obtenidos utilizando como circuito auxiliar una red de compuertas XOR (con abscisa "XOR"), una red de compuertas XOR con habilitación basada en compuertas AND (con abscisa "AND") y basada en multiplexores (con abscisa "MUX"). A efectos comparativos, el resultado con abscisa "DUT" (en rojo) es el cálculo para una instancia del diseño bajo prueba (determinado con las medidas de consumo su estático y total). Todos los circuitos fueron medidos con un reloj externo de 50MHz y a una temperatura ambiente de 37°C.

sub-estimarán su valor. Una fuente importante de este error radica en la medición del consumo dinámico de los circuitos auxiliares.

En todos los métodos, en el primer ensayo las redes auxiliares son ejercitadas con instancias del DUT. En el segundo ensayo, las redes auxiliares son ejercitadas por instancias LFSR a 50MHz. En el segundo caso, la actividad de la red auxiliar será mayor que en el primero.

Dado que el tiempo de los pulsos de estimulación es independiente de la frecuencia del reloj, ocurre que una gran parte de la actividad de la red auxiliar (cuando es manejada por los DUT) depende de los tiempos de los pulsos de terapia. Esto no ocurre cuando la red es manejada por los LFSR, que no sólo operan a 50 MHz sino que conmutan todos sus bits en forma pseudo-aleatoria. Este efecto contribuye a sobre-estimar el consumo de la red auxiliar ($C_{XOR}^{LFSR} > C_{XOR}^{DUT}$) y por lo tanto a sub-estimar el consumo del DUT.

Idealmente, el bloqueo de las salidas de los LFSR mediante el uso de compuertas "AND" y multiplexores reduciría, al punto de anular, la actividad de la red auxiliar. Los resultados muestran apenas una mejora utilizando el bloqueo con las compuertas "AND" y ninguna mejora utilizando multiplexores.

Si no se utilizan *IP-cores* (bloques provistos por los fabricantes y que suelen hacer uso de *hardware* dedicado), todos los circuitos combinatorios se sintetizarán a las LUT. De esta manera, la arquitectura final (luego del proceso de *fitting*) podrá distar mucho de la original. En una arquitectura con bloques 4-LUT, las compuertas "AND" de más de 4 entradas y multiplexores de más de dos canales requerirán más de un 4-LUT en cascada (ya que en ambos casos se tendrán por lo menos 5 señales de entrada). En la figura 6.2 se muestra la implementación de una función lógica de 5 entradas basada en bloques 4-LUT. Adicionalmente, en [37] se muestra una implementación de un multiplexor de 4 canales mediante dos 4-LUT.

Ya sea con una señal de habilitación (caso de bloqueo con compuertas "AND") o con canales de selección fijos (bloqueo con multiplexores) no podrá evitarse cierta propagación de la actividad a través de las 4-LUT. Para ilustrar esto, los diseños de la figura 6.3 fueron implementados en la misma FPGA. Como se observa, los modelos RTL equivalente son los deseados y los diseños ya sintetizados muestran la cascada de dos bloques 4-LUT.

Esto acarrea cierta propagación de las conmutaciones, causando que el bloqueo, si bien tiene el efecto esperado, admita cierto consumo dinámico.

Como fue comentado anteriormente, no es posible conocer exactamente el

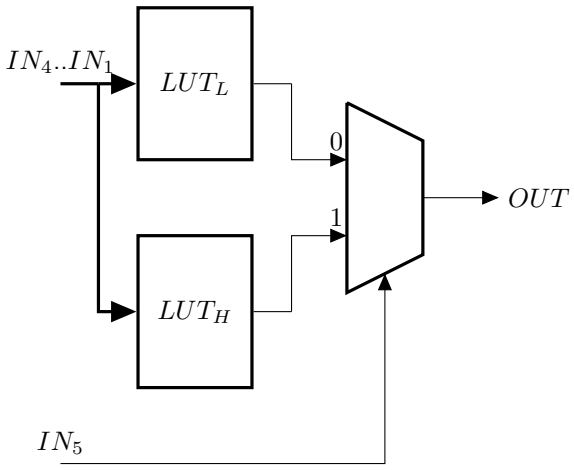

Figura 6.2: Función lógica de 5 bits de entrada implementada con dos bloques 4-LUT (que se manejan con los mismos 4 bits) y un selector binario (que se manejan con el quinto bit).

consumo dinámico real del diseño. Por un lado, el cálculo del mismo mediante la medida "directa" del consumo estático y total, tendrá un error debido al gran peso del estático en el total. Por otro lado, los métodos presentados tendrán un error que tendería a sub-estimar el consumo dinámico. No es posible saber qué resultado se encuentra más cerca del valor real, pero la distancia entre los resultados obtenidos no distan más de 10 %, por lo que el margen de error obtenido por cualquiera de los métodos resulta aceptable.

Adicionalmente, los tres métodos propuestos utilizan muy pocos pines de entrada/salida (ya que todas las salidas de los DUT se conducen a un único pin de la FPGA mediante las compuertas "XOR"). Considerando la simplicidad de su uso (por no tener que conectar cada señal de entrada/salida a un pin de la FPGA) y el bajo error que podrían alcanzar, cualquiera de los tres métodos propuestos resultan útiles para el cálculo del consumo dinámico de un diseño en escenarios de alto consumo estático.

Existen muchos ensayos interesantes para realizar respecto a estos mecanismos de medición y cálculo, los cuales se describen en la sección 7.2 como trabajo a futuro.

Como última observación respecto a los resultados obtenidos, la FPGA M2GL025 con el DUT fue medida a 50 MHz (ver tabla 6.1) y a 7,8125 MHz (ver tabla 5.2). Los resultados muestran que, habiendo reducido la frecuencia de reloj en un factor de 6,40, el consumo dinámico se redujo en un factor

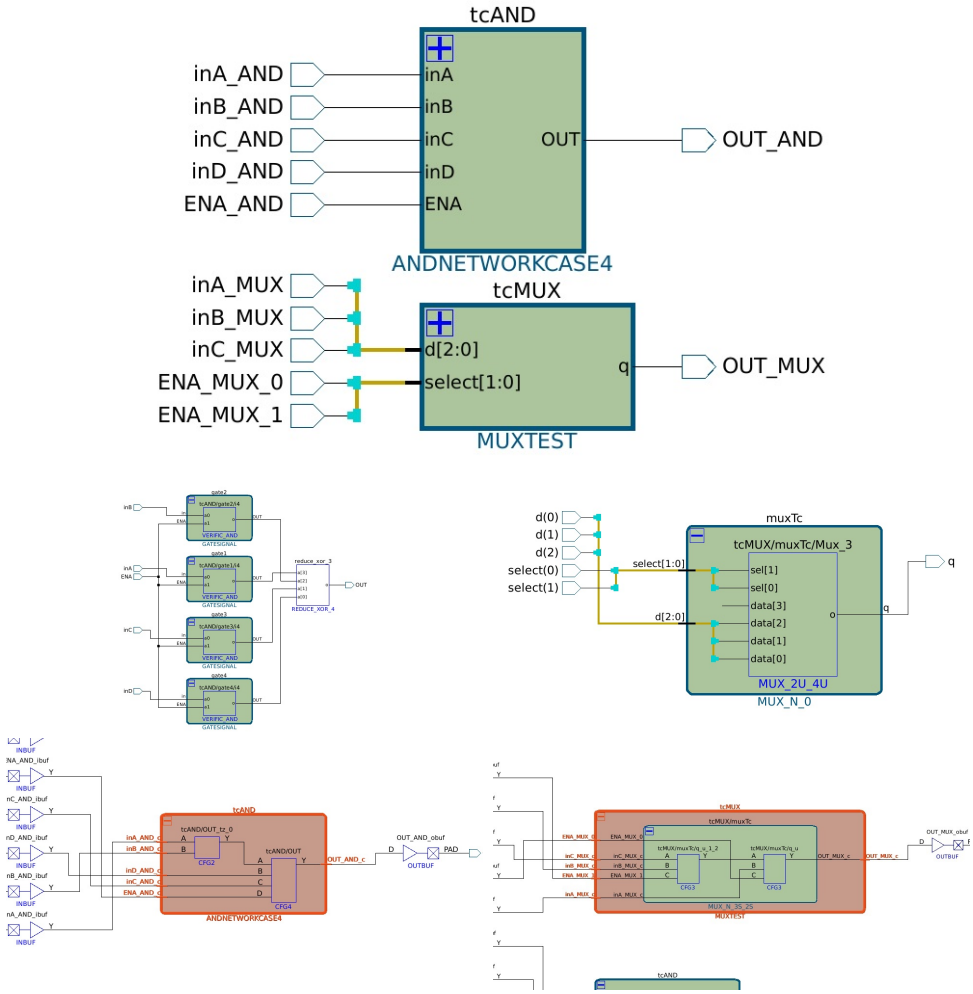

Figura 6.3: Implementación de una compuerta "AND" de 5 bits de entrada y un multiplexor de 3 canales (con 2 de selección). Las imágenes en la parte superior y media muestran el equivalente RTL *post-synthesis*. La imágenes en la parte inferior muestran el diseño *post-fitting*, donde se observa la cascada de dos 4-LUT en ambos casos.

de 1,65. Este efecto de reducción no proporcional es el comentado en 3.2.1 (no todos los nodos del diseño conmutan a la frecuencia del reloj, sino que muchos cambios se dan siempre a tiempos fijos que dependen del pulso de estimulación).

Capítulo 7

Conclusiones y trabajo futuro

7.1. Conclusiones

7.1.1. Respecto a las implementaciones y los consumos de potencia

Se diseñó la unidad de control de un neuromodulador simple. Este diseño se implementó en tres FPGA de distintos fabricantes (Intel, Microsemi y Lattice) con diferentes procesos de fabricación y tecnología, particularmente, con memoria de configuración basada en celdas de memoria FLASH y de memoria SRAM.

Se midieron los consumos estático y total de los *cores* de las FPGA con el la unidad de control implementada. De las FPGA ensayadas, el chip iCE40-HX-8K de Lattice es el que posee un menor consumo de potencia, tanto estática como dinámica (resultando en un consumo total del *core* de 3,62mW).

7.1.2. Respecto a las estimaciones de consumo

Adicionalmente, se utilizaron las herramientas de estimación de potencia provistas por los fabricantes (*PowerPlay* de Intel, *Smart Power* de Microsemi y *iCEcube* de Lattice). Se observó que los resultados de las herramientas de estimación poseen errores muy considerables, inclusive sub-estimando el consumo del diseño (lo cual puede acarrear problemas de implementación, por ejemplo en el diseño de las fuentes de alimentación).

La estimación de consumo no es trivial, ya que es necesario estimar la actividad de los nodos de la FPGA completa. Lamentablemente los grandes errores

de estimación no son novedad, como puede verse en [25] (donde se realiza un análisis detallado de los procesos de estimación de potencia) a lo largo de la historia estas herramientas han cometido errores de magnitud considerable. Por estos motivos, las herramientas de estimación disponibles actualmente no son confiables y lamentablemente no pueden ser utilizadas. Como corolario, todo diseño debe ser medido.

7.1.3. El consumo estático como el peor enemigo

Los ensayos realizados permitieron comprender que en aplicaciones de complejidad baja (de bajo volumen lógico) el consumo estático es el término más importante en el consumo total. Este define el consumo base que toda aplicación tendrá y depende de los procesos de fabricación, la tecnología y la temperatura de funcionamiento.

En el contexto de las FPGA, la cantidad de elementos lógicos de la misma se encuentra fuertemente vinculada con el consumo estático de potencia. Esto significa que en escenarios donde el consumo es crítico, la utilización de una FPGA de un tamaño acorde a las necesidades siempre será una buena opción.

7.1.4. Respecto a la estimación de la FPGA que mejor se ajusta a un diseño

En la práctica, para la verificación del diseño es muy práctico utilizar los *kit* de desarrollo provistos por los fabricantes, ya que resuelven muchos problemas prácticos (alimentaciones, circuito impreso, conexiones, etc.) que en estas instancias, generalmente, quieren ser evitados. Esto limita las FPGA a utilizar a las que las empresas decidan utilizar con tal fin, redundando en la elección de una FPGA con un volumen lógico considerablemente mayor al necesario (y por lo tanto con un consumo estático mayor al alcanzable).

El método presentado en la sección 5.4 es muy útil en esos casos, ya que permite implementar un diseño en un *kit* de desarrollo y determinar los requerimientos en volumen lógico (para la misma familia de FPGA que el *kit*) que aseguren el poder tener una instancia del diseño al menor costo en volumen lógico y por consiguiente, minimizando el consumo estático de potencia.

Este mecanismo de estimación del volumen lógico para una familia de FPGA dada es interesante, ya que condensa variables como la tecnología de

fabricación, la estructura interna y los procesos de síntesis, *placing* y *routing*.

En este punto cabe destacar que la FPGA con tecnología de fabricación más eficiente (en términos de consumo) no necesariamente resulta en la mejor elección, dado que los resultados dependen fuertemente de la eficiencia de las herramientas disponibles y de la estructura interna de las FPGA.

7.1.5. Respecto a los métodos utilizados para la medida de consumo dinámico

Se evaluaron tres métodos sencillos para la medida de consumo dinámico con mayor precisión, generando varias instancias del diseño bajo pruebas. Los métodos presentaron un error del entorno del 10 % para los ensayos realizados.

Se evidenció que el diseño genérico, sin uso de *IP-cores*, es pasible de ser alterado en forma importante al momento de realizarse el proceso de *fitting* (por reducirse los diseños a los elementos lógicos de la FPGA).

7.2. Trabajo futuro

7.2.1. Desempeño de las herramientas de estimación de consumo

Los resultados más groseros fueron los obtenidos con las herramienta de estimación de consumo, particularmente con la herramienta *PowerPlay* de Intel.

Esta herramienta permite realizar la estimación de consumo declarando la actividad del circuito a través de diversos mecanismos: restricciones temporales en el comportamiento de las señales de entrada, declaración del *toggle rate* de las mismas señales, generación de archivos de simulación o una mezcla de todas.

Para comprender mejor el comportamiento de la herramienta, sería interesante evaluarla con sus diferentes mecanismos de ingreso de datos, para diferentes diseños. Esto podría dar algo de luz respecto a las fortalezas y las debilidades de esta herramienta y así comprender mejor los errores obtenidos.

7.2.2. 4-LUT vs ALM

Como fue comentado en su momento, el uso de los bloques ALM proveen mucha flexibilidad a las FPGA, gracias a que sus LUT pueden ser divididas

de modo conveniente. Sería interesante evaluar las ventajas y desventajas de los ALM respecto a los 4-LUT (que son considerablemente más sencillos).

7.2.3. Diseño paralelo

El diseño de la unidad de control fue condicionado por el circuito utilizado para la generación de pulsos (presentado en la sección 1.3). Por esta razón cada bloque de terapia era multiplexado. Si se tuviera un diseño en *steering*, con tantas fuentes de corriente como terapias, cada bloque manejaría una fuente de corriente, simplificando notablemente el diseño y pudiendo obtenerse una reducción del consumo.

7.2.4. Área-Consumo-Velocidad

Las tres grandes aristas del diseño digital son el área de los diseños, la velocidad de los mismos y su consumo.

Típicamente, la restricción de dos de estas variables se puede resolver relajando la tercera. Si se desea un diseño que minimice el consumo y maximice la velocidad, sería sensato pensar que esto tenga un costo en área.

Si se deseara procesar datos a cierta velocidad y minimizar el consumo, puede pensarse un diseño en forma paralela, de modo de poder reducir la frecuencia de ejecución, manteniendo la velocidad de procesamiento. Esta solución reduce el consumo dinámico e implica aumentar el área del circuito.

Cuando el contexto de trabajo requiere muy bajo consumo, el consumo estático se vuelve un término fundamental. Este obedece a diversas variables como la tecnología de fabricación y la temperatura, entre otras. Como ya fue comentado, se pudo evidenciar el gran peso que tiene el tamaño de la FPGA en el consumo estático.

Esto introduce una restricción muy fuerte entre las variables área y consumo (estático) que obliga a repensar los razonamientos anteriores.

Básicamente, si se desea reducir el consumo de potencia de un diseño manteniendo la velocidad de procesamiento, el sacrificar área (con la estrategia comentada antes) disminuirá el consumo dinámico, pero aumentará el consumo estático. En contextos donde el consumo estático es mucho mayor que el dinámico, esta opción no parece ser la indicada. En contextos de consumo mixto, la decisión no parece trivial.

Escenarios como estos (donde las variables área, velocidad y consumo están fuertemente vinculadas) resultaría sumamente interesantes investigar la relación óptima de área y velocidad que minimizan el consumo de un diseño.

7.2.5. Tecnologías utilizadas

Sería muy fructífera la realización de ensayos adicionales en otras plataformas por diferentes motivos, los cuales se listan a continuación:

- Medidas de consumo estático en alguna FPGA de la familia 5CE-x-A2 (Cyclone V) y en alguna de la familia M2GL005 (IGLOO2) para verificar los valores calculados en las secciones 5.4.4 y 5.4.4 (donde se buscaron las FPGA de cada familia que mejor se ajustaran a los requerimientos del diseño).
- Con motivo de aumentar la cantidad de fabricantes, sería interesante generar una instancia del diseño en una FPGA de Xilinix [38] ya que es uno de los de mayor presencia en el mercado.
- Las FPGA con tecnología *antifuse* son una opción para reducir drásticamente el consumo estático, ya que el "grabado" de la FPGA produce una conexión no reversible similar a una vía. Si bien sólo se pueden grabar una vez, no tienen memoria de configuración. Se intentó comprar una FPGA de la familia Axcelerator de Microsemi [39] pero por costos y disponibilidad no pudo ser adquirida.
- Otras FPGA que se ajusten mejor en tamaño al diseño, de modo de poder calcular los límites en consumo que hoy en día pueden obtenerse para una aplicación de este tipo.

7.2.6. Métodos de medida de consumo dinámico

Como fue comentado en su momento, el uso de los LFSR a 50MHz no sería el mejor método para la medida de consumo de la red auxiliar, ya que no sería un fiel representante de lo que ocurre cuando la red es conectada a los DUT. Por esta razón, sería necesario realizar más ensayos variando la frecuencia de operación para observar su dependencia.

Las técnicas basadas en compuertas "AND" y multiplexores para bloquear la actividad de la red auxiliar fue implementada en forma genérica, para independizar la implementación de la plataforma. Esto parece ser innecesario, ya

que si la finalidad es medir el consumo dinámico de un diseño en cierta plataforma y con la mayor precisión posible sería sensato utilizar los *IP-cores* que el fabricante provea. En esta linea se podrían definir *wrappers* con la idea de poder instanciar, por citar un ejemplo, un "BUS-MUX" a través de un *IP-core* o con el modelo genérico.

Tomando en cuenta que la actividad de los DUT es periódica (regida a groso modo por las frecuencias de estimulación), es posible generar otro circuito de medida utilizando registros e introduciendo un *delay* igual al período de la actividad en los puertos de salida. De este modo podría excitarse la red auxiliar con un circuito que genere exactamente la misma actividad que los DUT originales. En la figura 7.1 se presenta un mecanismo que con los dos ensayos descritos podría determinar el consumo del DUT con mejor precisión.

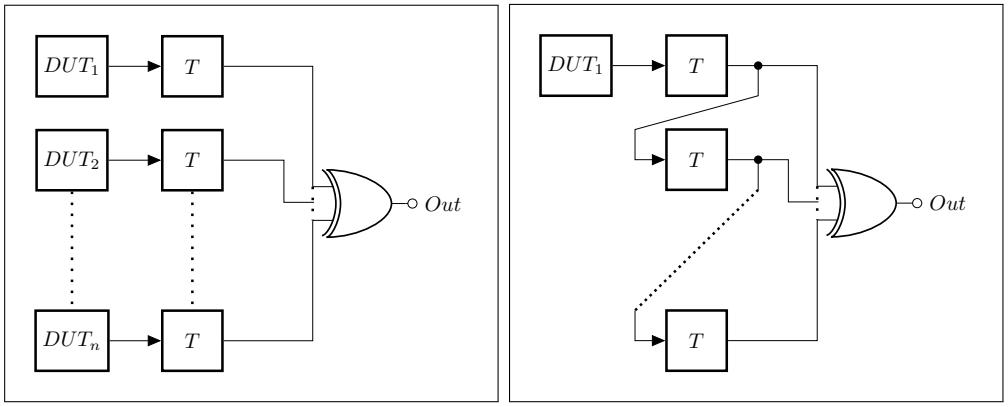

Figura 7.1: Otro método para el cálculo del consumo dinámico.

Esta configuración tiene la ventaja que la capacidad vista por los DUT es similar, pudiendo variar solamente por el ruteo a través de las interconexiones.

Referencias bibliográficas

[1] R. Melzack and P. D. Wall, "Pain mechanisms: A new theory," *Science*, vol. 150, no. 3699, pp. 971–979, 1965.

[2] "Paresthesia Information Page — National Institute of Neurological Disorders and Stroke." https://www.ninds.nih.gov/Disorders/All-Disorders/Paresthesia-Information-Page. Accessed: 2018-08-16.

[3] "Parkinson's Disease Information Page — National Institute of Neurological Disorders and Stroke." https://www.ninds.nih.gov/Disorders/All-Disorders/Parkinsons-Disease-Information-Page. Accessed: 2018-08-16.

[4] "Tremor Fact Sheet — National Institute of Neurological Disorders and Stroke." https://www.ninds.nih.gov/Disorders/Patient-Caregiver-Education/Fact-Sheets/Tremor-Fact-Sheet. Accessed: 2018-08-16.

[5] M. L. Kringelbach, N. Jenkinson, S. L. Owen, and T. Z. Aziz, "Translational principles of deep brain stimulation," *Nature Reviews Neuroscience*, vol. 8, pp. 623–635, 2007.

[6] "Arrhythmia — National Heart, Lung, and Blood Institute (NHLBI)." https://www.nhlbi.nih.gov/health-topics/arrhythmia#Types. Accessed: 2018-08-16.

[7] BSI Standards Publication, "Implants for surgery. Active implantable medical devices. General requirements for safety, marking and for information to be provided by the manufacturer. BS EN 45502-1:2015."

[8] BSI Standards Publication, "Active implantable medical devices. Particular requirements for active implantable medical devices intended to treat bradyarrhythmia (cardiac pacemakers). BS EN 45502-2-1:2003."

[9] "Cyclone® V FPGA - Intel® FPGA." https://www.intel.com/content/www/us/en/products/programmable/fpga/cyclone-v.html?wapkw=cyclone+v. Accessed: 2018-08-12.

[10] "iCE40 LP/HX/LM - Lattice Semiconductor." https://www.latticesemi.com/Products/FPGAandCPLD/iCE40. Accessed: 2018-08-12.

[11] "IGLOO2 FPGAs — Microsemi." https://www.microsemi.com/product-directory/fpgas/1688-igloo2. Accessed: 2018-08-12.

[12] "CCC del Uruguay - Medical Devices - Pacemaker." http://www.ccc.com.uy/esp/company/director.htm. Accessed: 2018-08-12.

[13] M. Sawan, S. Robin, B. Provost, Y. Eid, and K. Arabi, "A wireless implantable electrical stimulator based on two fpgas," in *Proceedings of Third International Conference on Electronics, Circuits, and Systems*, vol. 2, pp. 1092–1095 vol.2, Oct 1996.

[14] N. Modir, K. Fricke, Z. E. Abid, and R. Sobot, "Controlling and signal processing core for wireless implantable telemetry system," in *2016 IEEE International Conference on Electronics, Circuits and Systems, ICECS 2016*, 2017.

[15] F. Zhang, M. Aghagolzadeh, and K. Oweiss, "A low-power implantable neuroprocessor on nano-FPGA for Brain Machine interface applications," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2011.

[16] F. Zhang, M. Aghagolzadeh, and K. Oweiss, "A programmable and implantable microsystem for multimodal processing of ensemble neural recordings," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, 2011.

[17] J. Rose, R. J. Francis, D. Lewis, and P. Chow, "Architecture of Field-Programmable Gate Arrays: The Effect of Logic Block Functionality on Area Efficiency," *IEEE Journal of Solid-State Circuits*, 1990.

[18] "Cyclone V Device Overview." https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-v/cv_51001.pdf. Accessed: 2018-10-01.

[19] P. Chow, S. O. Seo, J. Rose, K. Chung, G. Paez-Monzon, and I. Rahardja, "The design of an sram-based field-programmable gate array. i. architecture," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 7, pp. 191–197, June 1999.

[20] P. Chow, S. O. Seo, 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 Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 7, pp. 321–330, Sept 1999.

[21] J. Brown, W. Brewer, *Nonvolatile Semiconductor Memory Technology.* IEEE Press editorial board, 1998.

[22] M. Abusultan and S. P. Khatri, "Exploring static and dynamic flash-based FPGA design topologies," in *Proceedings of the 34th IEEE International Conference on Computer Design, ICCD 2016*, 2016.

[23] R. M. Swanson and J. D. Meindl, "Ion-implanted complementary mos transistors in low-voltage circuits," *IEEE Journal of Solid-State Circuits*, 1972.

[24] S. G. Narendra and A. Chandrakasan, *Leakage in Nanometer CMOS Technologies (Series on Integrated Circuits and Systems).* Berlin, Heidelberg: Springer-Verlag, 2005.

[25] J. Oliver, *Técnicas de bajo consumo en FPGAs.* PhD thesis, 2014.

[26] M. George and P. Alfke, "Linear Feedback Shift Registers in Virtex Devices," tech. rep., 2007.

[27] "MAX5532 - Digital to Analog Converter." www.maxim-ic.com. Accessed: 2018-08-27.

[28] "Synchronous FIFO." http://www.asic-world.com/examples/systemverilog/syn_fifo.html. Accessed: 2018-10-01.

[29] Intel, "Cyclone V Device Datasheet," tech. rep., 2018.

[30] "Early Power Estimators (EPE) and Power Analyzer." https://www.intel.com/content/www/us/en/programmable/support/support-resources/operation-and-testing/power/pow-powerplay.html. Accessed: 2018-09-08.

[31] Microsemi, "DS0134 Datasheet SmartFusion2 SoC FPGA and IGLOO2 FPGA," tech. rep., 2018.

[32] Lattice, "DS1040 - iCE40 LP/HX Family Data Sheet," tech. rep., 2017.

[33] P. Du, *A Fast Heuristic Technique for FPGA Placement based on Multilevel Clustering*. PhD thesis, 2004.

[34] Intel, "Cyclone V FPGA Features," tech. rep.

[35] "IGLOO2 FPGAs — Product Table." https://www.microsemi.com/product-directory/fpgas/1688-igloo2#product-tables. Accessed: 2018-09-28.

[36] "iCE40 LP/HX/LM - Lattice Semiconductor." https://www.latticesemi.com/Products/FPGAandCPLD/iCE40. Accessed: 2018-09-28.

[37] N. Kumar, "LUT based multiplexers, United States Patent N: US 7486,110 B2." https://patentimages.storage.googleapis.com/5c/b8/3f/a91885100eeebc/US7486110.pdf, 2009.

[38] "Xilinx - Adaptable. Intelligent." https://www.xilinx.com/. Accessed: 2018-08-30.

[39] "Axcelerator — Microsemi." https://www.microsemi.com/product-directory/antifuse-fpgas/1700-axcelerator. Accessed: 2018-09-28.