Instituto Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

Implementación de una metodología para lograr la integración física "correcta por construcción (CBC)" del Sistema de Reconocimiento de Patrones Acústicos (SiRPA)

Informe de Proyecto de Graduación para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura

Luis Eduardo Abrahams Vargas

Cartago, 27 de enero de 2012

Declaro que el presente Proyecto de Graduación ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos propios.

En los casos en que he utilizado bibliografía he procedido a indicar las fuentes mediante las respectivas citas bibliográficas. En consecuencia, asumo la responsabilidad total por el trabajo de graduación realizado y por el contenido del correspondiente informe final.

Luis Eduardo Abrahams Vargas

Cartago, 27 de enero de 2012

Céd.: 1-1182-0005

Instituto Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica
Proyecto de Graduación
Tribunal Evaluador

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniera en Electrónica con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal

Dr. Pablo Alvarado Moya
Profesor Lector

Ing. Roberto Pereira Arroyo
Profesor Lector

Dr. Alfonso Chacón Rodríguez
Profesor Asesor

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica.

Cartago, 27 de enero de 2012

Resumen

Aproximadamente la cuarta parte del territorio costarricense consiste en áreas boscosas protegidas, las cuales albergan una gran proporción de la biodiversidad mundial. Sin embargo, gran cantidad de plantas y animales se ven amenazados por la presencia de la tala y la cacería ilegales y esta situación se ve agravada por las severas limitaciones en materia de vigilancia y cuido.

A través de variadas investigaciones, se han desarrollado distintas soluciones, la mayoría de ellas orientadas a la vigilancia automatizada en áreas de difícil acceso; no obstante, estas soluciones se ven limitadas por su consumo de energía, el cual tiende a ser relativamente alto en la etapa actual de los prototipos. Dada la escasez de fuentes de energía en las zonas boscosas, conviene ir un paso más adelante en su desarrollo, lo que implica implementar una solución de bajo consumo de energía como lo es un circuito integrado de aplicación específica o ASIC.

En este trabajo se detalla la utilización de un marco CAD que permite realizar la integración física "correcta por construcción" o CBC a partir de la descripción en HDL de un sistema de reconocimiento de patrones acústicos en el bosque, implementando los pasos necesarios así como las pruebas y comprobaciones pertinentes para la generación de los archivos necesarios para la integración física de este sistema mediante la utilización de las herramientas comerciales de Synopsys.

Palabras clave: circuito integrado de aplicación específica ASIC, marco CAD, correcta por construcción CBC, sistema de reconocimiento de patrones acústicos SiRPA, Synopsys

Abstract

About one fourth of Costa Rica's territory belongs to protected areas which host a significant proportion of the world's biodiversity. Nonetheless, many species of plants and animals see their existence threatened by ilegal lumbering and hunting and this situation is worsened by the serious limitations existing in the surveillance and protection of these areas.

Different solutions have been proposed to address these issues, many of them oriented to automated and remote surveillance especially in places of difficult access, but they are limited by their power consumption, which tends to be relatively high at the present stage of the prototypes. Due to the limited availability of power sources in the woods, it becomes mandatory to go one step ahead and to implement a low power-consumption solution such as application specific integrated circuit or ASIC

The present document details the use of a CAD framework that allows to get the "correct by construction" (CBC) physical integration of a digital system, derived from its HDL description such as the acoustic pattern recognition system, implementing the necessary steps and tests in order to obtain the required files for the physical layout of the system using the work flow commercial tools provided by Synopsys.

Keywords: application specific integrated circuit ASIC, CAD framework, correct by construction CBC, acoustic pattern recognition system SiRPA, Synopsys

a mi madre y a hime-chan, sin ellas no habría llegado hasta aquí.

Agradecimientos

El resultado de este trabajo no hubiese sido posible sin el apoyo del Dr. Alfonso Chacón Rodríguez y del Dr. Pablo Alvarado Moya, les agradezco por la oportunidad de participar en este proyecto. A todos los profesores que no solo haber brindado su conocimiento sino también haber contribuido con mi formación profesional. A mi familia, compañeros y amigos que siempre me han brindado su apoyo incondicional. Particular nota al Ing. Erick Salas Chaverri por su invaluable conocimiento del funcionamiento del SiRPA y al Ing. Jairo Valverde Cruz por su ayuda con las herramientas utilizadas, así como a José Pablo Castro por su incansable contribución al uso de Linux.

Finalmente a todos los que han participado y en un futuro participen en el desarrollo de este proyecto:

Muchas Gracias.

Luis Eduardo Abrahams Vargas

Cartago, 27 de enero de 2012

Índice general

Índice de figuras

Índice de tablas

Lista de Ejemplos

Lista de símbolos y abreviaciones

Abreviaciones

ASIC	Circuito Integrado de Aplicación Específica
CBC	Correcta Por Construcción
FPU	Unidad de Punto Flotante
HDL	Lenguaje de Descripción de Hardware
LDA	Análisis de Discriminantes Lineales
PCA	Análisis de Componentes Principales
RTL	Nivel de Registros de Transferencia
SiRPA	Sistema de Reconocimiento de Patrones Acústicos
VCS	Compilador y Simulador de Verilog (software propietario, Synopsys)

Capítulo 1

Introducción

Costa Rica posee más de 1,3 millones de hectáreas de áreas protegidas, las cuales representan un 26 % del territorio nacional que se encuentra distribuido entre 28 parques nacionales, 8 reservas biológicas, 31 zonas protegidas, 9 reservas forestales, 71 refugios de vida silvestre, 15 humedales y 4 zonas clasificadas como reservas naturales absolutas o monumentos; dentro de estas áreas protegidas los guardaparques son los responsables de evitar la tala de los bosques protegidos, la caza de animales en riesgo de extinción y la invasión de áreas silvestres protegidas. En el 2008 el número de guardaparques era solamente de 500 funcionarios, que si se repartieran por todo el territorio a proteger, cada uno tendría a su cargo un total de 2.654 hectáreas, el equivalente a 37 veces el Parque Metropolitano La Sabana [4]. Ello sin considerar que los guardaparques tienen que dividir su tiempo entre la protección y cuido de las áreas, el trabajo administrativo y la atención a los visitantes.

Ante la falta de capacidad para vigilar las extensas áreas protegidas es imperativo el desarrollo de dispositivos que ayuden a cumplir con la vigilancia y detección de acciones como la cacería y la tala ilegal, en particular el desarrollo de dispositivos o sensores que sean capaces de detectar disparos y motosierras al analizar los sonidos del bosque.

Tal es el objetivo del proyecto "Diseño de una red inalámbrica de telecomunicaciones para la protección ambiental en el bosque" [2] que utiliza una red inalámbrica de sensores y el reconocimiento de patrones acústicos entre otras técnicas para la detección y ubicación de disparos y lograr así dar aviso oportuno a las autoridades correspondientes.

Este proyecto consta de tres módulos generales (alimentación de potencia, detección y comunicación) y a partir de él se han generado una serie de trabajos, detallados a continuación.

1.1 Antecedentes

Como parte del proyecto "Diseño de una red inalámbrica de telecomunicaciones para la protección ambiental en el bosque" se desarrolló un sistema para reconocer patrones acústicos de motosierras y disparos, basado en la teoría de las redes neuronales artificiales utilizando discriminadores de Fisher, con un consumo de potencia de 1,8W. La simulación se realizó por medio de la herramienta MatLab y se obtuvo un porcentaje de error de un 12 % en las detecciones de motosierras y un 10 % en las de disparos en un área de cobertura de 2862m^2 [3].

En [8] se realizó un trabajo a nivel de software, que permite la extracción de características por medio de la teoría de wavelets (identifica los contenidos de tiempo-frecuencia específicos) y la clasificación de los sonidos (disparo o motosierra) por medio de clasificadores estadísticos, basados en Modelos Ocultos de Markov (HMM). Los resultados experimentales obtenidos muestran un 90 % de probabilidades de identificar la fuente sonora con tiempos de reconocimientos igual o menores a 0,05s [6].

Para analizar en forma digital las señales acústicas en tiempo real, se creó un prototipo algorítmico para la detección de disparos y motosierras. Se obtuvo una tasa de reconocimiento de disparos del 94,12 % y una tasa creciente para el modelo de motosierras [10].

En la figura 1 se muestra el esquema general de solución del proyecto “Diseño de una red inalámbrica de telecomunicaciones para la protección ambiental en el bosque", que está compuesto por los módulos de acople, preproceso, extracción de características, y clasificación. En [7] se documenta una implementación en FPGA utilizando el lenguaje de descripción de hardware VHDL de los siguientes submódulos: un normalizador de nivel, un banco de filtros digitales que divide la señal en ocho bandas de frecuencia, una etapa de extracción de características energéticas y un evaluador de HMM, tal como se observa en la figura 1.1.

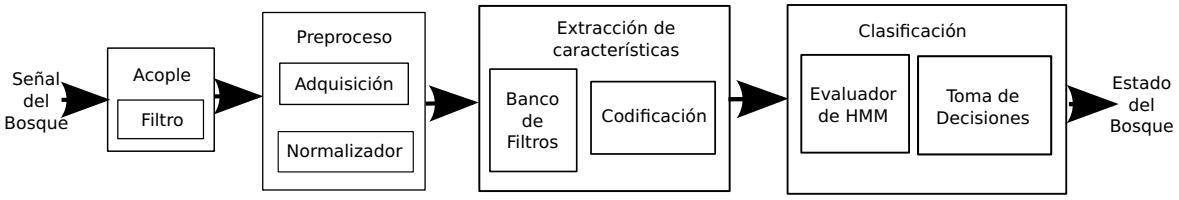

Figura 1.1: Diagrama de bloques del Sistema de Reconocimiento de Patrones Acústicos (SiRPA)

1.2 Acceso a la energía en el bosque.

Ya que el objetivo del proyecto es la instalación de los dispositivos en áreas boscosas donde el acceso a la energía es limitado, se debe orientar el desarrollo de los mismos hacia una implementación de muy bajo consumo de potencia y de bajo costo.

El desarrollo actual de los distintos submódulos del SiRPA está realizada a nivel de HDL implementado en FPGA; sin embargo, desplegar una red de sensores sobre este tipo de plataforma en el bosque se vuelve poco práctico debido a que el consumo de potencia requerido agotaría rápidamente un sistema de baterías, obligando a realizar cambios periódicos de la mismas. Además “la idea de estos circuitos es crear un sensor que pueda trabajar muchos años con una batería como la de un reloj y que no sea necesario estarla reemplazando” [5].

Esto hace pensar en la implementación sobre otra plataforma como la de circuitos integrados de aplicación específica (ASIC por sus siglas en inglés), que permite el desarrollo de las aplicaciones necesarias con un bajo consumo de potencia.

En [11] se expone el flujo necesario para el diseño ASIC aplicado al primer submódulo del SiRPA -un banco de filtros digitales segmentado- así como la implementación de un marco CAD para realizar la síntesis RTL y la verificación lógica, el proceso de obtención de la lista de nodos, realizar la implementación física, la verificación de

temporizado, y la verificación eléctrica para dicho submódulo; los resultados de esta primera aproximación generan valores estimados del consumo de potencia mucho menores a los de la implementación actual en FPGA.

1.3 Estructura del documento

Para facilitar el proceso de lectura este documento se separó en capítulos de manera que estos cubran cada etapa del desarrollo del proyecto. En el capítulo 2 se plantean la meta y los objetivos del proyecto. En el capítulo 3 se expone el flujo para el diseño de circuitos integrados de aplicación específica (ASIC) y se establecen los principios teóricos básicos de cada etapa dentro del Sistema de Reconocimiento de Patrones Acústicos (SiRPA) que fue abarcada en este proyecto. En el capítulo 4 se expone el procedimiento metodológico seguido para el desarrollo del proyecto. En el capítulo 5 se muestra el marco CAD implementado para realizar la síntesis RTL y la implementación física de un sistema descrito en alto nivel. En el capítulo 6 se presenta el resultado de la síntesis RTL y se detalla el resultado de la obtención del trazado físico para la etapa de reducción de dimensiones. En el capítulo 7 se presenta el resultado de la síntesis RTL y se detalla el resultado de la obtención del trazado físico para la etapa de codificación. En el capítulo 8 se presenta el resultado de la síntesis RTL y se detalla el resultado de la obtención del trazado físico para la etapa de modelos ocultos de Markov. En el capítulo 9 se presenta el resultado de la síntesis RTL y se detalla el resultado de la obtención del trazado físico para la etapa general del SiRPA. Finalmente en el capítulo 10 se presentan las conclusiones obtenidas y las recomendaciones del proyecto.

Capítulo 2

Meta y objetivos

2.1 Meta.

Partiendo de sistemas digitales descritos en alto nivel generar los archivos finales utilizados para la fabricación de circuitos integrados con un menor consumo de potencia y con un menor costo en comparación con la implementación en FPGA.

Indicador: El consumo de potencia y el costo se deben disminuir al implementar el sistema digital en un circuito integrado en comparación con la implementación en FPGA.

2.2 Objetivo general.

Utilizar la metodología "Correcta por Construcción" (CBC) para obtener la implementación física del Sistema de Reconocimiento de Patrones Acústicos (SiRPA).

Indicador: La integración física "correcta por construcción (CBC)" debe cumplir con las especificaciones física y lógicas del sistema minimizando su consumo de potencia en al menos un 50 %.

2.3 Objetivos específicos.

1. Obtener la síntesis lógica y la implementación física del módulo de reducción de dimensiones y generador de símbolos.

   Indicador: La implementación física debe cumplir con las especificaciones lógicas del sistema minimizando su consumo de potencia en al menos un 50 % en comparación con la implementación en FPGA.

2. Obtener la síntesis lógica y la implementación física del módulo de Modelos Ocultos de Markov.

   Indicador: La implementación física debe cumplir con las especificaciones lógicas del sistema minimizando su consumo de potencia en al menos un 50 % en comparación con la implementación en FPGA.

3. Obtener la síntesis lógica y la implementación física todos los módulos en conjunto.

Indicador: La implementación física debe cumplir con las especificaciones lógicas del sistema minimizando su consumo de potencia en al menos un 50 % en comparación con la implementación en FPGA.

Capítulo 3

Marco teórico

3.1 Marco CAD para la integración física CBC.

3.1.1 Integración física "correcta por construcción (CBC)"

Tal como se detalla en [11], la integración física "correcta por construcción (CBC)" es el proceso por el que se obtienen los archivos necesarios para la fabricación de un circuito integrado a partir de la descripción en HDL de un sistema digital. Durante este proceso la estructura interna del sistema digital puede variar pero sin alterar su funcionamiento lógico.

Para realizar la integración física "correcta por construcción (CBC)" de un sistema digital descrito en alto nivel se utiliza un flujo para el diseño ASIC. Las etapas del flujo para el diseño ASIC son descritas a continuación:

1. Idea: es el origen del flujo, que corresponde a lo que se desea diseñar.
2. Especificaciones ASIC: como especificaciones ASIC se consideran los siguientes aspectos:
    2.1. Objetivos y limitaciones del diseño.
    2.2. Especificaciones lógicas: la función lógica que debe realizar el sistema digital.
    2.3. Especificaciones físicas: el consumo de potencia y temporizado del sistema digital.
3. RTL (Register Transfer Level): la idea y las especificaciones lógicas son plasmadas en un diseño a nivel de registros de transferencia utilizando un HDL.
4. Simulación lógica: permite verificar el funcionamiento lógico del RTL. Esta simulación se realiza en las siguientes tres etapas del flujo para el diseño de circuitos integrados de aplicación específica (ASIC):
    4.1. Durante la creación del RTL.
    4.2. Después de realizar la síntesis lógica.
    4.3. Después de la implementación física.

5. Síntesis lógica: es el proceso en el cual el diseño a nivel de registros de transferencia es mapeado a una lista de nodos a nivel de compuertas lógicas (*Gate Level Netlist*) que contiene sólo bloques lógicos incluidos en la biblioteca de celdas estándar.

   La biblioteca de celdas estándar posee bloques lógicos tales como compuertas lógicas, registros, y bloques de funciones aritméticas. Cada bloque lógico incluido en la biblioteca de celdas estándar contiene tres formas de representación, las cuales se describen a continuación:

   5.1. Representación CELL: contiene el trazado físico.

   5.2. Representación FRAM: posee el trazado físico en forma más simple utilizado durante la implementación física.

   5.3. Representación LM: incluye la información de temporizado y del consumo de potencia.

6. Implementación física: en esta etapa se obtiene el trazado físico (*layout*) del sistema digital a partir de la lista de nodos a nivel de compuertas (*Gate Level Netlist*) obtenida en la síntesis lógica. Además se genera la lista de nodos a nivel de compuertas con la información de las resistencias y capacitancias parásitas. La implementación física consta de las siguientes tres etapas:

   6.1. Descripción física: se transforma la descripción lógica de la lista de nodos a nivel de compuertas en una descripción física y se busca minimizar el área y el retardo de sistema digital.

   Durante esta etapa de la implementación física se hace una estimación del área del circuito integrado, se asignan los pines, y se crean los anillos de alimentación del circuito integrado.

   6.2. Colocación física: en esta etapa se colocan en un lugar específico los diferentes bloques lógicos que conforma la lista de nodos a nivel de compuertas. Se trata de minimizar el área (colocando los bloques lo más cerca posible) y se busca minimizar el largo de los cables que unen los diferentes bloques lógicos (colocando en forma adyacente los bloques necesarios para realizar determinada función).

   6.3. Enrutado físico: consiste en unir entre sí los diferentes bloques lógicos que conforman el circuito integrado. Al finalizar esta etapa, el trazado físico del sistema digital ha sido creado y se guarda como archivo GDSII .

7. Verificación de temporizado: se determinan las rutas críticas del diseño y se verifica que no existan problemas de temporizado.

8. Simulación eléctrica: permite determinar la existencia de errores en el trazado físico (circuito abierto o corto circuito).

9. GDSII: es el archivo que contiene toda la información del trazado físico y se utiliza para fabricar el circuito integrado.

El flujo para el diseño ASIC se muestra en la figura 3.1, en donde se observan las etapas necesarias para originar los archivos utilizados en la fabricación de un circuito integrado partir de una idea y especificaciones de un sistema digital.

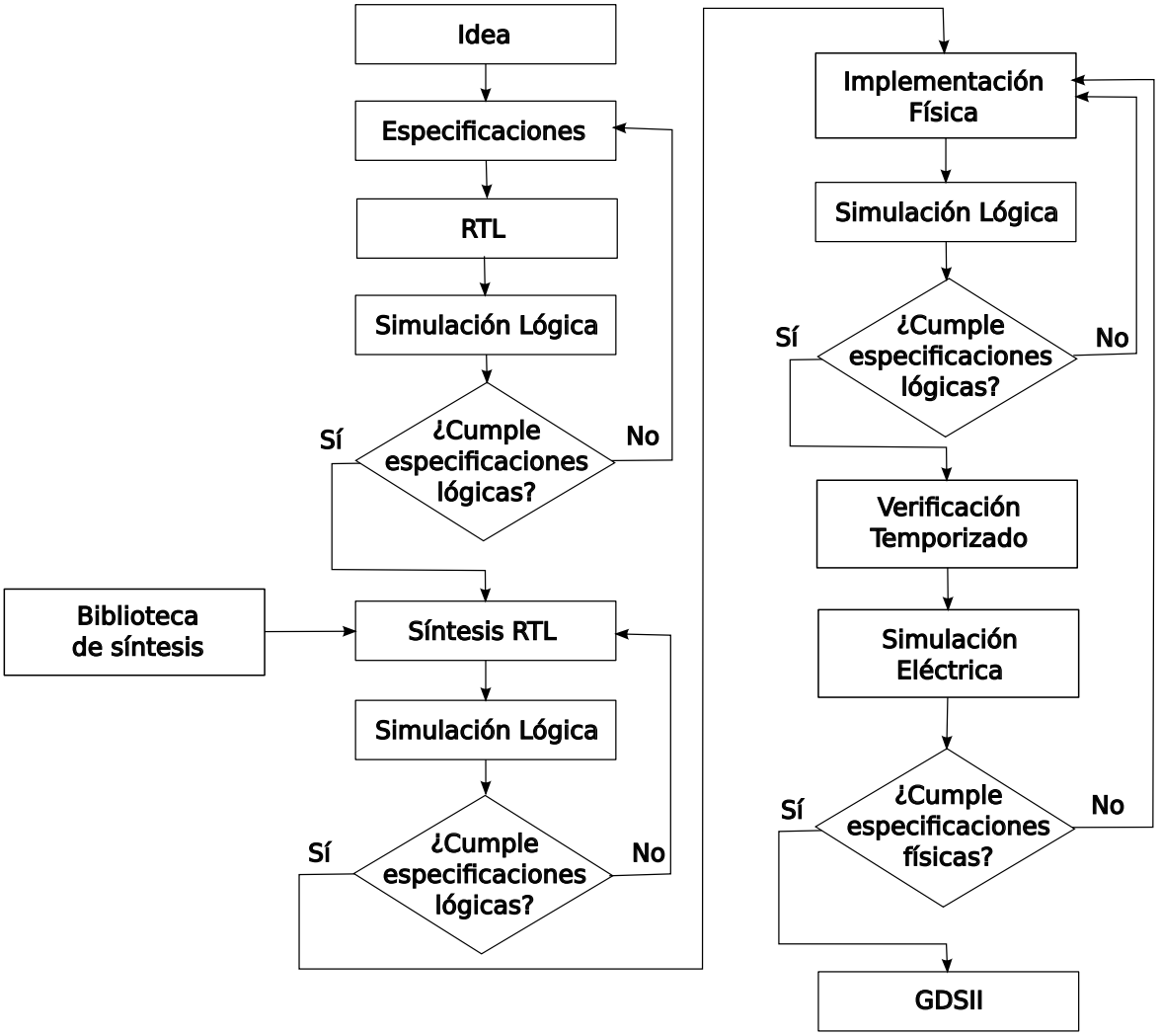

Figura 3.1: Flujo para el diseño de circuitos integrados de aplicación específica (ASIC) desarrollado en [11].

3.2 Reducción de dimensiones.

3.2.1 Reducción de dimensiones.

Tal como se desarrolla en [9], para obtener la reducción de dimensiones se realiza una transformación lineal del espacio de entrada de ocho dimensiones mediante un producto matriz-vector a partir del cual se obtiene un espacio tridimensional. El espacio de entrada corresponde a las ocho bandas de frecuencia obtenidas a través de las etapas de extracción de características del SiRPA previas al reductor de dimensiones. La figura 3.2 ilustra este proceso.

La implementación del reductor de dimensiones consta de dos etapas: una de entrenamiento y otra de ejecución. Durante la fase de entrenamiento se obtiene la matriz de transformación W mediante un entrenamiento realizado con herramientas de software en un computador externo al sistema empotrado, a partir de muestras de datos de cada una de las clases que se quieren detectar (en este caso: 'disparo', 'motosierra' y 'bosque'). Para obtener dicha matriz se utilizaron dos técnicas diferentes, LDA y PCA, para poder comparar la capacidad de separación de clases que realiza cada una durante la reducción dimensional, siendo elegida finalmente la LDA por sus características de separación de los datos, pues permite una mejor clasificación posterior de los mismos.

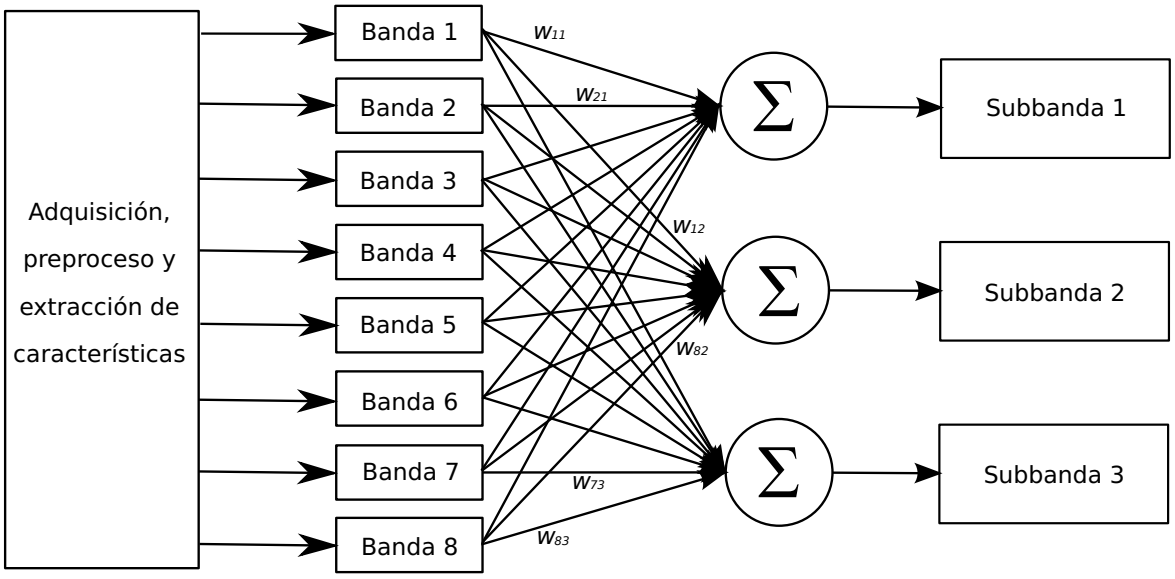

Figura 3.2: Reductor de dimensiones desarrollado en [9]. Durante una fase de entrenamiento se obtienen los 24 pesos w_{ij} que conforman la matriz de transformación.

Durante la fase de ejecución, se utilizan los pesos w_{ij} en la combinación lineal de las salidas octo-dimensionales del banco de filtros del SiRPA, para obtener vectores en el subespacio de tres dimensiones. Esto se realiza multiplicando los 24 coeficientes de la matriz W por las bandas de entrada y combinando los productos obtenidos en tres sumadores, como se muestra en la figura 3.2. Los resultados de estas sumas corresponden a las subbandas de salida de tres dimensiones.

3.2.2 Restador de media.

Las muestras en ocho dimensiones de disparos, motosierras y bosques, utilizadas para el entrenamiento, brindan una referencia del comportamiento esperado de los datos a la salida de la etapa de extracción de características del SiRPA. Durante la etapa de ejecución, el reductor de dimensiones resta a las señales de entrada la media (u *offset*) de los datos en ocho dimensiones, con el propósito de trasladar los vectores octo-dimensionales del espacio de entrada al origen del sistema de coordenadas. Es necesario hacer esto para contribuir a la estabilidad numérica del algoritmo.

3.2.3 Multiplicadores CSD.

Durante la etapa de ejecución del reductor de dimensiones, el espacio de entrada de ocho dimensiones es mapeado por la matriz de transformación para obtener la proyección a un espacio tri-dimensional. Esto implica realizar 24 multiplicaciones entre las bandas de entrada y los coeficientes de la matriz W. Las entradas del reductor corresponden a las ocho bandas de frecuencia en que se descomponen las señales de audio a través de las etapas de extracción de características. Las señales de cada banda tienen representación de 16 bits en complemento a dos. Los coeficientes de W obtenidos tanto con LDA como con PCA tienen valores reales constantes que oscilan entre $[-1, 1]$.

Para realizar los productos de las señales de entrada con los coeficientes w_{ij} , la implementación de multiplicadores CSD en lugar de multiplicadores binarios normales

permite un ahorro de hardware al reducir al menos a la mitad la cantidad de sumadores necesarios para los multiplicadores.

3.3 Codificación mediante el uso de un árbol k-D.

3.3.1 Algoritmo para la búsqueda del vecino más cercano de acuerdo con una medida de desigualdad o distancia utilizando un árbol k-D.

Este algoritmo, implementado en [1], resuelve el problema que se presenta cuando se tiene un archivo o memoria donde se almacena una cantidad N de puntos o vectores descritos por un número k de valores enteros dentro de un rango específico, lo cual se puede interpretar como N vectores en un espacio euclidiano de k dimensiones, y se requiere encontrar el punto más cercano o con los atributos más similares a los de un punto cualquiera descrito por las k dimensiones y en el mismo rango de valores.

Este algoritmo hace uso de un ordenamiento de los N vectores por medio de un aŕbol k-D, el cual es un árbol binario utilizado para el ordenamiento y la búsqueda, en el que cada nodo representa un subarchivo de vectores del archivo total. El nodo raíz es el archivo total de vectores. Cada nodo representa una partición en una dimensión específica dividiendo así el subarchivo en otros dos subarchivos hasta llegar a los nodos terminales, u hojas, los cuales no poseen partición pero dentro de los límites de cada uno de estos nodos se encuentra un vector exclusivo de los N vectores del archivo raíz.

Para el caso del SiRPA el algoritmo se implementó para un archivo que almacena 32 vectores descritos por 8 valores enteros en el rango de -32768 a 32767, es decir, rango representable con 16 bits (números con signo). El sistema digital diseñado se basó en una máquina de estados, utilizando el lenguaje de descripción de hardware VHDL, capaz de encontrar el vector, dentro de los 32 almacenados, más cercano o con la menor distancia a un vector cualquiera o vector analizado.

La búsqueda se realiza basándose en el orden, por medio del árbol k-D, en que se encuentran almacenados los vectores permitiendo una disminución en la cantidad de comparaciones de un vector almacenado con el vector analizado y con ello encontrar el vector más cercano en menor tiempo respecto a una búsqueda lineal, lo cual conlleva una ventaja en la utilización de este algoritmo cuando se requiere encontrar el vecino más cercano o con la menor medida de desigualdad respecto a un vector cualquiera.

El sistema digital posee una memoria ROM en la cual se encuentra almacenada la información del árbol k-D:

- cada nodo numerado desde el 1 al 63 y sus límites superiores e inferiores para cada dimensión
- la partición y la dimensión en la que se encuentra
- un bit que determina si es terminal o no
- otras características necesarias para realizar la búsqueda por parte del circuito

La máquina de estados principal que conforma el sistema hace uso de la información almacenada en la ROM y ejecuta el algoritmo de búsqueda el cual recorre el árbol

realizando comparaciones entre la partición de cada nodo y el valor en la misma dimensión del vector analizado, permitiendo llegar hasta el nodo terminal cuyos límites encierran los 8 valores, correspondiendo a cada dimensión, que describen al vector analizado y determinar la distancia entre este vector y el vector que se encuentra en el nodo terminal.

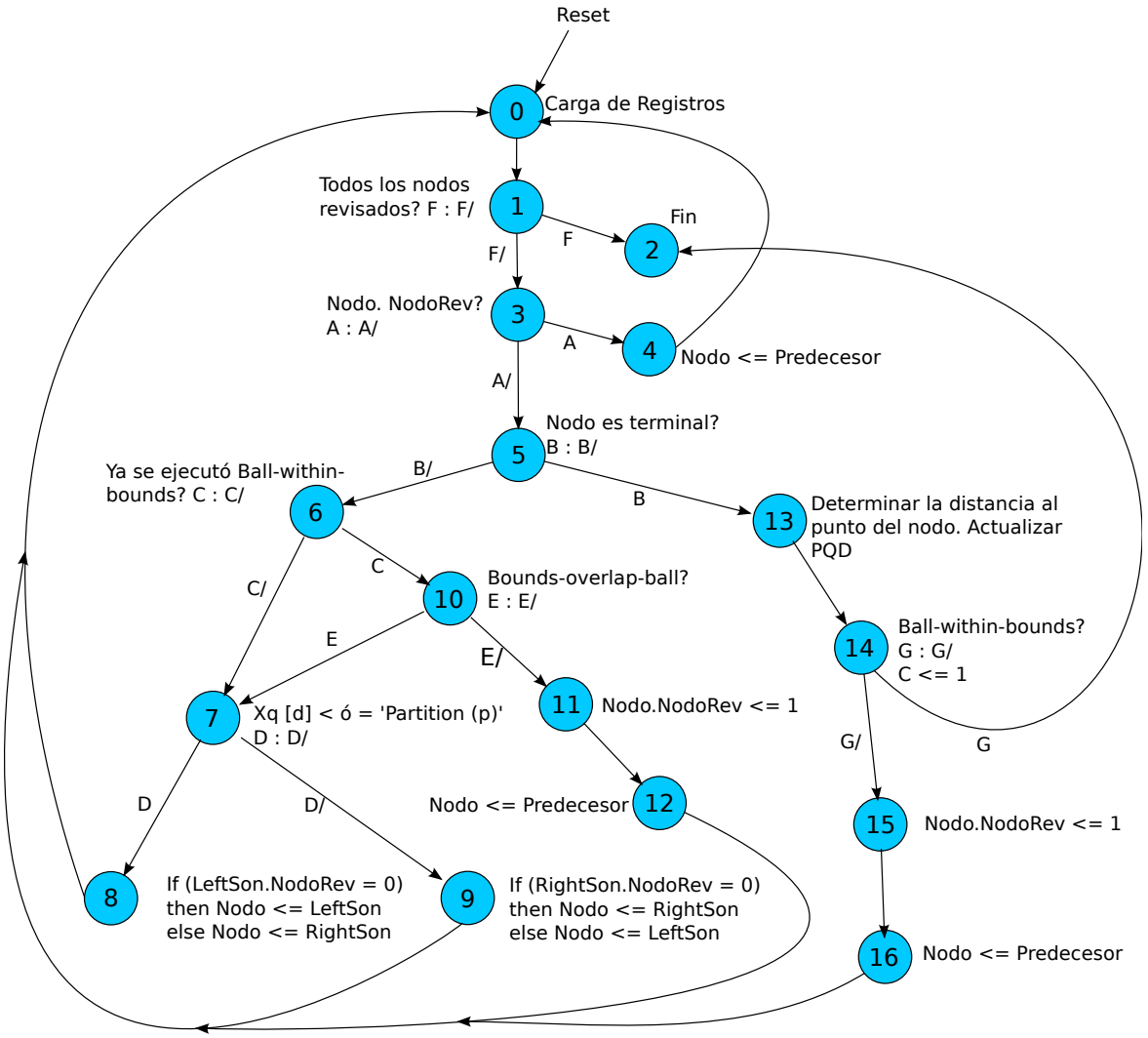

Figura 3.3: Diagrama de estados para el sistema de búsqueda del árbol k-D utilizado en [1].

El sistema ejecuta el algoritmo de búsqueda por medio del diagrama de estados que se muestra en la figura 3.3. En este diagrama se puede observar como la máquina de estados recorre el árbol k-D realizando las comparaciones necesarias y determinando la menor distancia y así el vecino más cercano.

El proceso inicia verificando si todos los nodos han sido revisados, ya que cada nodo posee un bit como bandera que determina esta condición. Si este es el caso entonces se termina la búsqueda. En la salida del sistema se encuentra un número del 0 al 31 que identifica al vector más cercano. Si esta condición no se cumple el sistema continúa la búsqueda preguntando si el nodo analizado ya fue revisado o no, y en caso de ser positiva esta condición el sistema carga el nodo predecesor y vuelve a iniciar el proceso. Si el nodo cargado o nodo por analizar no ha sido revisado entonces se procede a determinar si es un nodo terminal. En caso de serlo, el sistema ejecuta un proceso que determina la distancia entre el vector que se encuentra en el nodo y el vector analizado y determina si esta es la menor de las distancias hasta ahora encontradas.

Por tanto, se determina si el vector del nodo es el vecino más cercano de los hasta ahora encontrados. Si el nodo no es terminal el sistema realiza la comparación entre la partición del nodo y el valor del vector analizado en la misma dimensión y con ello determinar el siguiente nodo al cual analizar; es decir, si el valor del vector es menor o igual que la partición entonces el nodo por analizar será el hijo izquierdo del nodo actual y si no será el hijo derecho. Este nodo será cargado y el sistema reiniciará el proceso de búsqueda.

3.4 Clasificación utilizando Modelos Ocultos de Markov.

A partir del desarrollo efectuado en [7] se resume la estructura del circuito digital requerido para la implementación de los Modelos Ocultos de Markov, el cual analiza las cadenas de observación que provienen de la etapa de codificación mediante el algoritmo hacia adelante (*forward*). La estructura clasifica los eventos en actividad normal del bosque, disparo o sonido de motosierra, aun con ruido, distorsión o algún efecto que altere el sonido. El algoritmo hacia adelante es un conjunto de procedimientos de cálculo matricial que requiere aritmética en punto flotante debido al tratamiento de probabilidades.

3.4.1 Implementación del MAP.

Para la etapa de clasificación y toma de decisiones se diseñó el MAP (*Matrix and Arrays Processor*) [7], que es una estructura digital cuya arquitectura está optimizada para ejecutar operaciones en punto flotante de manera combinacional con entradas matriciales o arreglos. Esta arquitectura soporta el estándar de punto flotante de 32 bits (IEEE – 754).

La entrada de reloj (MAP_clk) del sistema tiene la frecuencia mínima de 800 Hz que es relativa al periodo de muestreo de la cadena de observación, que corresponde a 0,1s. La entrada $Port\_in$ corresponde al puerto de ingreso de datos al sistema que representa la entrada de símbolos del alfabeto de la cadena de observación para que sea procesado por el MAP. La entrada *int0* carga cada símbolo ingresado en el puerto. Respecto a las salidas del sistema del MAP, las señales de los pines de $HMM\_highest$ representan el modelo HMM de mayor probabilidad que denota el estado del bosque. El estado de MAP_ready indica la condición de espera por un nuevo símbolo. La condición de excepción muestra que los cálculos aritméticos internos no son válidos por causa de la ejecución de división entre cero.

3.4.2 Arquitectura del MAP que implementa el método de *forward* basado en HMM.

El circuito mostrado en la figura 3.4 detalla los bloques funcionales del MAP. La arquitectura incluye un controlador de datos de entrada ($Port$), un banco de memoria (acceso y almacenamiento temporal de cálculos aritméticos, $Mtrx\_RAM$), la memoria de programa ($Program$), un contador de programa (PC), el control general de la arquitectura ($Ctrl$), el acumulador de los HMM de probabilidad máxima (MAX), el

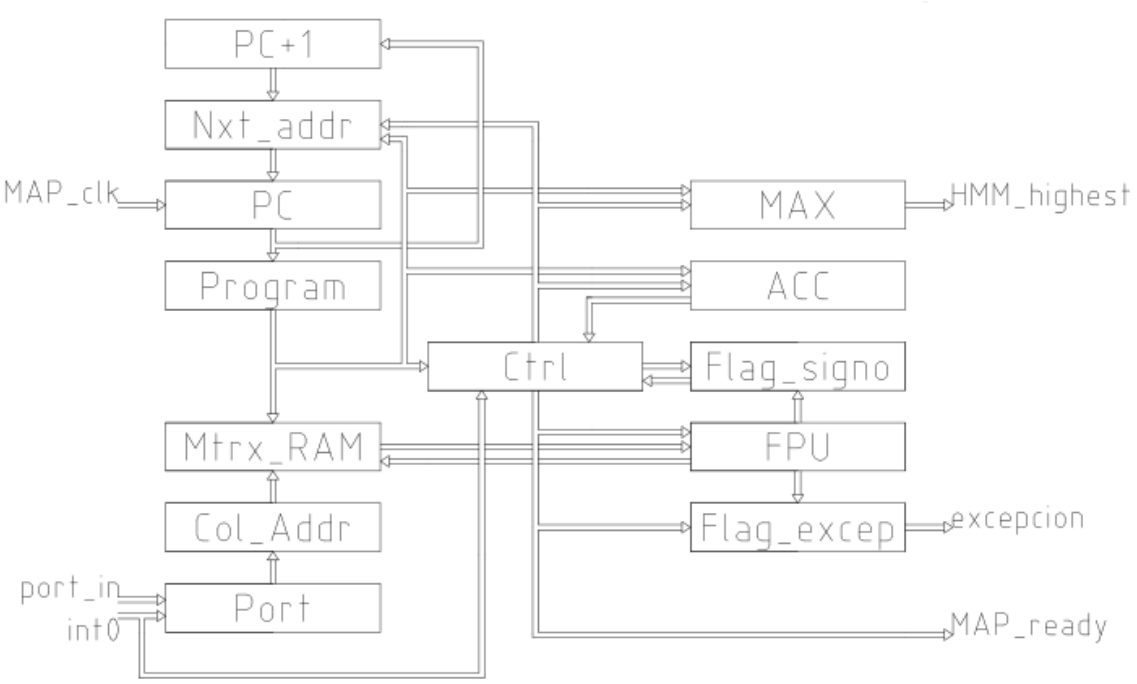

Figura 3.4: Diagrama de bloques general del MAP tomado de [7].

acumulador de número de símbolo (ACC), la unidad de aritmética en punto flotante (FPU) y el bloque de la bandera de excepción.

El MAP es un microprocesador dedicado de funcionamiento monociclo cuya arquitectura es tipo Harvard. La arquitectura del MAP soporta operaciones de punto flotante (suma, comparación, multiplicación y división), operaciones en punto fijo como el decremento de un registro de índice e instrucciones de bifurcación como el salto tipo con o sin condición. El cálculo de direcciones para el acceso de las instrucciones es distinto para el almacenamiento de los datos. A continuación se describen los bloques de banco de memoria y la unidad de punto flotante.

3.4.3 Descripción de la memoria de acceso y almacenamiento de datos.

La estructura MAP tiene un banco de memoria para el acceso de las matrices. Esta unidad preveé bloques de cálculo de direcciones para el alineamiento por filas y columnas. Para el almacenamiento de datos se tiene un arreglo de registros con la habilidad de lectura independiente de la escritura. El banco de memoria direcciona 2 elementos simultáneamente en forma separada de cualquier lugar del mapa de direcciones disponible para la obtención de los operandos.

3.4.4 Descripción de la unidad aritmética de punto flotante.

El funcionamiento del MAP está especializado en las operaciones aritméticas de punto flotante con precisión de 32 bits (IEEE – 754). Dichas operaciones son implementadas en estructuras digitales combinacionales separadas (suma, multiplicación y división). La unidad de punto flotante es una adaptación de la biblioteca para VHDL desarrollada para FPGA bajo licencia GPL. En la figura 3.5 se muestran los bloques internos de la

FPU. Para el caso de la resta, se utiliza un bloque negador del argumento. Se incluye una estructura para soportar la selección de las operaciones según la señal de control (*ctrl*).

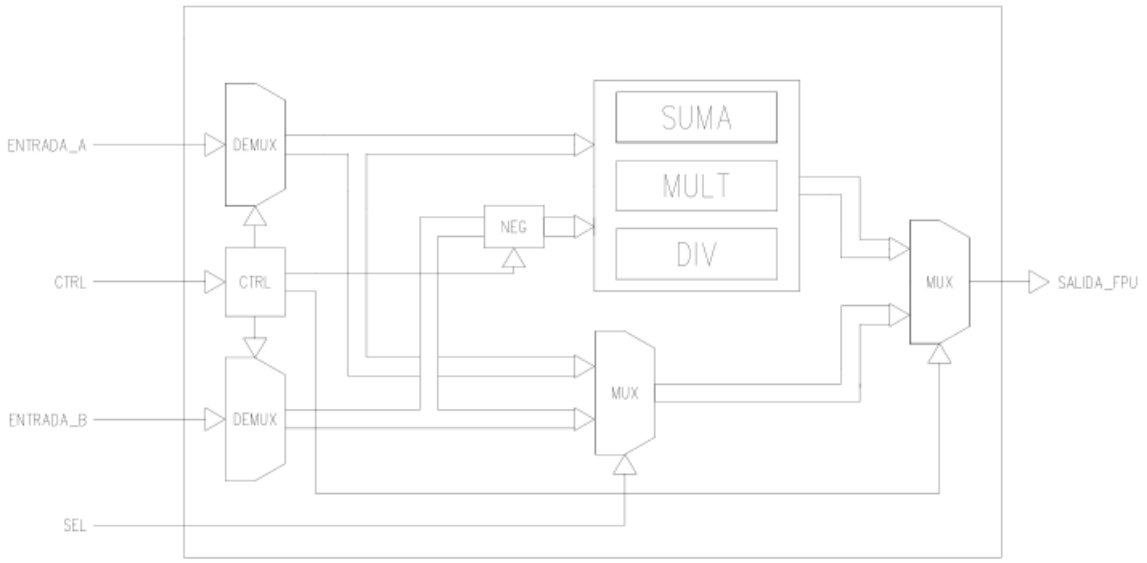

Figura 3.5: Esquema interno de la unidad de aritmética en punto flotante (*FPU*) tomado de [7].

Capítulo 4

Metodología

Para cumplir con los objetivos de este proyecto se propuso ejecutar los siguientes pasos para cada uno de los módulos y para la unión de todos ellos en una implementación final:

1. Establecer el marco CAD para realizar la síntesis lógica de un sistema digital descrito en alto nivel y la verificación lógica de la lista de nodos a nivel de compuertas.

2. Obtener la lista de nodos a nivel de compuertas de cada uno de los módulos que componen el sistema.

3. Verificar que la lista de nodos a nivel de compuertas cumpla con las especificaciones lógicas de cada módulo.

4. Establecer el marco CAD para realizar la implementación física, la verificación de temporizado, y la verificación eléctrica de la lista de nodos a nivel de compuertas de un sistema digital.

5. Obtener el trazado físico de cada uno de los módulos que componen el sistema.

6. Obtener la lista de nodos a nivel de compuertas con información de las capacitancias y resistencias parásitas de cada uno de los módulos que componen el sistema.

7. Verificar que la lista de nodos a nivel de compuertas con información de las capacitancias y resistencias parásitas cumpla con las especificaciones lógicas de cada uno de los módulos que componen el sistema.

8. Verificar el temporizado de la lista de nodos a nivel de compuertas con información de las capacitancias y resistencias parásitas de cada uno de los módulos que componen el sistema.

Capítulo 5

Montaje de un marco CAD para la implementación física de un sistema digital para el reconocimiento de patrones acústicos.

En este capítulo se describe el montaje del marco CAD usado para realizar la síntesis lógica de un sistema digital descrito en alto nivel y la verificación lógica de la lista de nodos a nivel de compuertas así como para realizar la implementación física, la verificación de temporizado, y la verificación eléctrica de la lista de nodos a nivel de compuertas de un sistema digital. Así mismo se incluyen los ajustes realizados para su utilización con las bibliotecas de la tecnología definida para este proyecto.

5.1 Marco CAD utilizado

El marco CAD implementado es el siguiente:

- La síntesis lógica de un sistema digital descrito en alto nivel se realiza utilizando el programa *Design Compiler* de Synopsys en conjunto con las bibliotecas para la tecnología CMOS IBM 8RF de 130nm suministradas por MOSIS.
- La verificación lógica de la lista de nodos a nivel de compuertas se realiza por medio del programa *VCS*.
- El trazado físico se obtiene con el programa *IC Compiler* de Synopsys.
- Para realizar la verificación de temporizado se utiliza el programa *Primetime*.
- La simulación lógica de la lista de nodos a nivel de compuertas con información de las capacitancias y resistencias parásitas se obtiene nuevamente por medio del programa *VCS*.

5.2 Bibliotecas de celdas estándar

Para este proyecto se propone utilizar las bibliotecas de celdas estándar ARM hechas para el proceso 8RF de IBM de 130nm suministradas por MOSIS, en preparación

de la eventual fabricación del circuito integrado. Esta tecnología tiene las siguientes características:

- Tecnología CMOS de 130nm.
- Tensión de VDD de 1,2V.
- Acepta señales de E/S de hasta 3,3V.
- Optimizadas para bajo consumo de potencia.
- Utiliza 8 capas de metal, 6 delgadas y 2 gruesas.

Entre los archivos utilizados por la herramienta *ICC* se encuentran los archivos *.tluplus (TLU+). En sí, los modelos TLUPlus son un juego de modelos que contienen características de proceso avanzadas que pueden ser utilizados para extraer información de parásitas para el modelado, como la información de los archivos *.spef. Estos a su vez son utilizados por la herramienta *Primetime* para el análisis del temporizado. Los archivos *.tluplus son generados a partir de los archivos *.itf (*Interconnect Technology File*). Sin embargo, en la nueva biblioteca de MOSIS, no se encontraban estos archivos por lo que el análisis de temporizado se realizó con la información de parásitas extraída solamente a partir del archivo de tecnología *.tf. Esta información es menos detallada que la que se puede obtener a partir de los modelos TLUPlus y además los valores contenidos son del tipo *peor caso posible* por lo que tienden a ser un poco pesimistas.

Dado que este proyecto continúa directamente con trabajos previos, en particular [11], resulta conveniente realizar una comparación entre los resultados de un módulo previamente implementado con una biblioteca de celdas estándar proporcionada por Synopsys contra la nueva biblioteca de celdas estándar y así verificar que tanto el flujo de trabajo como el marco CAD utilizado siguen siendo vigentes.

Para lograr este punto se realizaron los ajustes correspondientes a los *scripts* utilizados para la implementación física de la etapa del banco de filtros digital segmentado para incorporar las nuevas bibliotecas, así como cambios en el orden de los módulos a ser análizados pues estos deben de ir en orden de menor a mayor jerarquía, siendo el último por analizar el módulo principal de la etapa. Algunos de estos cambios se muestran en el ejemplo 1.

Dado que la implementación fue correcta y la verificación lógica de la lista de nodos a nivel de compuertas fue correcta también, se comprobó que el marco establecido sigue vigente para la nueva biblioteca de 130nm, siendo la principal diferencia en los resultados en características relacionadas con la construcción física como son la potencia y el área utilizada; estas mismas se recopilan en la tabla 5.1.

Tabla 5.1: Consumo de potencia y área para el banco de filtros segmentado utilizando diferentes tecnologías.

	Bibliotecas de Synopsys	Bibliotecas ARM - IBM
Potencia dinámica (nW)	152,9	117,7303
Potencia estática (μW)	241,9	0,0969
Área (μm^2)	77870, 2510	70511,0402

Se puede ver una disminución en el área utilizada de un 9,45 %, mientras que la potencia dinámica consumida estimada en esta etapa para una frecuencia de muestreo

Ejemplos 1 Modificaciones al *script* de síntesis del banco de filtros digital segmentado. se puede observar como los archivos con definiciones de términos se colocan primero pues son utilizados por los demás archivos fuente.

```
...
#Primero se analiza los otros módulos de adentro hacia afuera
analyze -format vhdl {def_MAPx.vhd \
def_oper_int.vhd \
def_param.vhd \
def_type.vhd \
mux.vhd \
registro.vhd \
pila_registro.vhd \
pila_reg_join_free.vhd \
pila_reg_join_mux.vhd \
pila_reg_free_mux.vhd \
codificador_prioridad.vhd \
reloj_prioridad.vhd \
coeficiente_a11.vhd \
coeficiente_a12.vhd \
coeficiente_a22.vhd \
coeficiente_b11_H.vhd \
coeficiente_b11_L.vhd \
coeficiente_b12_H.vhd \
coeficiente_b12_L.vhd \
ganancia_alto.vhd \
ganancia_bajo.vhd \
filtro_pipelining.vhd \
multi_reloj.vhd}
# Luego se analiza el módulo principal
analyze -library WORK -format vhdl {filtro_segmentado.vhd}
...
```

de 44,1kHz se reduce en un 23,0 % y la potencia estática consumida estimada es de sólo un 0,04 % con respecto a la anterior implementación. Esta reducción tan grande de la potencia estática consumida puede deberse a un error en la biblioteca de celdas estándar de Synopsys, al ser esta una biblioteca de prueba de la herramienta, puesto que el valor obtenido es muy alto para esta implementación.

5.3 Conclusiones

Al incorporar una nueva biblioteca de celdas estándar se debe distinguir qué archivos son necesarios para el proceso y cuáles no y de esta manera incorporar los adecuados. Además, de no existir alguno es necesario verificar si es posible extraerlo utilizando las herramientas disponibles con algún archivo presente o consultar con el proveedor.

Capítulo 6

Implementación física de la etapa de reducción de dimensiones.

En este capítulo se presentan los resultados de realizar la síntesis lógica y la implementación física la de la etapa de reducción de dimensiones. Se analizan los resultados de la obtención de la lista de nodos a nivel de compuertas, los resultados de la verificación lógica, la verificación de temporizado y la estimación de consumo de potencia y área a partir de las listas de nodos obtenidas tras la síntesis para esta etapa.

6.1 Síntesis lógica y verificación de resultados

El diseño original de la etapa de reducción de dimensiones consta de varios módulos, entre estos un restador de *offset*, multiplicadores CSD y sumadores. La etapa es completamente combinacional, lo que vuelve imposible la optimización de la lógica por parte del *Design Compiler*, ya que no existe una señal de reloj como referencia. Además de esta situación, se pueden presentar rutas de datos demasiado largas, que no sólo introducen retardos distintos dependiendo de las diferentes rutas de datos posibles dentro de la etapa (aunque estas diferencias pueden no ser determinantes para este diseño en particular debido a su baja frecuencia de operación) si no que al no haber una estructura de encauzamiento (como el *pipelining*) también se aumenta el consumo dinámico de la etapa debido a la presencia inevitable de riesgos lógicos o *glitches*. De igual forma, la estimación de la potencia utilizada se vuelve un asunto complejo, pues no existe una señal de referencia que indique la frecuencia de conmutación de los datos en el circuito, por lo tanto la potencia dinámica estimada se dispara a valores muy altos pues la herramienta la calcula suponiendo que cada entrada conmuta a la máxima frecuencia posible.

Al analizar el diseño para su síntesis se descubrieron incongruencias en los tamaños de algunas señales internas, tal como en las entradas de los multiplicadores CSD, pues estas señales utiiizan 19 bits mientras que la señal correspondiente a la salida del restador de *offset* es de 16 bits. Dado el tratamiento de la señal como un entero con signo en complemento a 2 [9] se debió codificar una extensión de signo para hacer el empalme de estas señales.

Cabe mencionar también que el código en VHDL de esta etapa fue generado automáticamente para facilitar su creación, pues cada uno de lo 24 multiplicadores CSD utiliza un coeficiente (o multiplicando fijo) distinto, el cual debe ser convertido al formato

CSD en una etapa de entrenamiento previa, por lo cual dicho código contiene muy poca parametrización y reutilización de recursos.

Debido a la falta de un archivo de prueba para simulación del circuito y de los datos necesarios para crear uno, no se pudo realizar una comprobación del funcionamiento lógico de esta etapa; a pesar de este hecho los resultados presentados en cuanto a las características eléctricas del circuito, como consumo de potencia, no deberían de cambiar significativamente por cualquier cambio que se realizara en el diseño siempre que este conserve la misma estructura lógica.

En la figura 6.1 se puede apreciar el esquemático obtenido de la sintetización de esta etapa utilizando la herramienta *Design Compiler*, conservando los niveles de jerarquía, mostrando los distintos módulos que componen esta etapa.

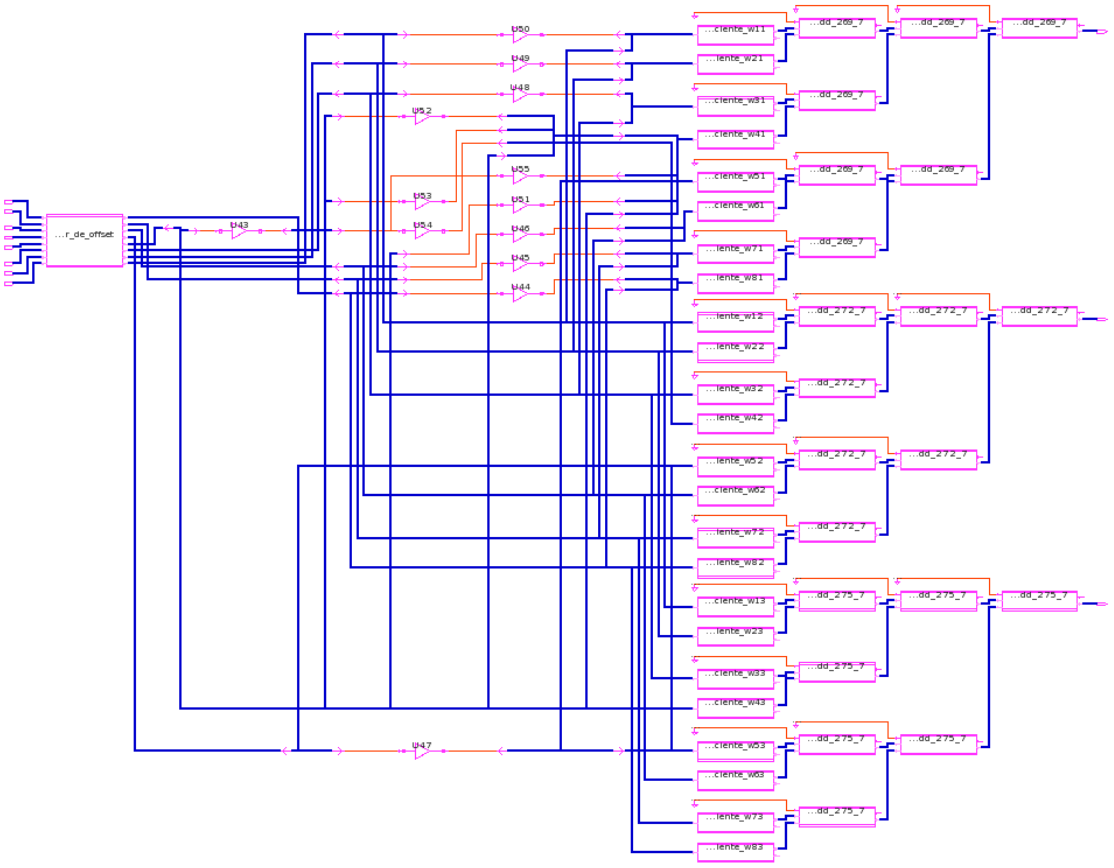

Figura 6.1: Esquemático resultante de la síntesis del módulo reductor de dimensiones. Se puede ver las 8 entradas de datos pasar al restador de offset, mientras que las salidas provienen de los sumadores.

6.2 Estimación del consumo de potencia y área utilizada

A partir de los reportes generados por la herramienta *Design Compiler* se puede ver el estimado de área utilizada y de potencia consumida por esta etapa, los cuales se presentan en la tabla 6.1. Debido a que la etapa es combinacional y no posee una señal de reloj en su diseño, la herramienta supone que las entradas conmutan a la frecuencia definida para el reloj, que al no estar definida en el *script* resulta en la utilización de

la frecuencia más alta soportada por la herramienta, que es de 1GHz; esto deriva en un cálculo de consumo dinámico excesivo.

Tabla 6.1: Estimación de área y consumo de potencia para el módulo de reducción de dimensiones.

Área (μm^2)	59.804,6408
Potencia estática (nW)	108,7669
Potencia dinámica (mW)	19,6302
Potencia total (mW)	19,6303

Se puede apreciar mejor la diferencia en el consumo de potencia cuando se compara contra los resultados obtenidos anteriormente bajo las mismas condiciones de implementación, los cuales se muestran en la tabla 6.2.

Tabla 6.2: Consumo de potencia para el reductor de dimensiones utilizando diferentes tecnologías.

	FPGA	Síntesis
Potencia estática (μW)	80980	0,1088
Potencia dinámica (mW)	-	19,6302

Se puede apreciar como la potencia estática estimada es de tan sólo un 0,00013 % de la potencia estimada para para el diseño implementado en FPGA mientras que se da una reducción del 75,76 % en la potencia total consumida para esta etapa calculada a partir de la síntesis con respecto a la implementación anterior en FPGA.

6.3 Obtención del trazado físico

La obtención del trazado físico de la etapa se obtuvo utilizando la herramienta *IC Compiler*. En la figura 6.2 se muestra el trazado obtenido. Nuevamente la falta de un archivo de prueba impide comprobar el funcionamiento de esta etapa, por lo que los resultados se muestran con caracter nominal.

6.4 Verificación de temporizado

Debido a que esta etapa es totalmente combinacional y que por tanto carece de una señal de reloj contra la cual poder medir las rutas críticas y el temporizado, no se pudo llevar a cabo una prueba utilizando la herramienta *Primetime* como se indica en el marco CAD establecido. Sin embargo, sí se cuenta con el reporte de retardo generado por la herramienta *IC Compiler*.

Este resultado indica un retardo entre entrada y salida de 15,62ns lo cual arroja una frecuencia máxima de trabajo de 64,02MHz. Esta frecuencia máxima está muy por encima de la frecuencia a la que se espera trabaje el sistema total una vez finalizado, la cual está en el orden de centenas de Hz, por lo que es de esperar que una vez incorporado al SiRPA esta etapa cumpla con las especificaciones establecidas, aunque siempre es recomendable colocar salidas registradas para evitar problemas de sincronía con el resto de las etapas.

Figura 6.2: Trazado físico resultante de la implementacón del módulo de reducción de dimensiones.

6.5 Consumo de potencia del trazado físico

A partir del reporte brindado por la herramienta, se extrajeron los datos estimados de potencia consumida por el sistema una vez realizada la implementación física, que se exponen en la tabla 6.3.

Tabla 6.3: Estimación de área y consumo de potencia para el módulo de reducción de dimensiones.

Área (μm^2)	59804,6408
Potencia dinámica (mW)	19,6373
Potencia estática (nW)	108,7650

Al comparar estos datos con los obtenidos de la síntesis se corrobora que el área de implementación se mantiene mientras que se nota una ligera diferencia en la potencia consumida, siendo la potencia dinámica un 0,036 % mayor y la potencia estática un 0,0017 % menor.

6.6 Conclusiones

Debido a que la etapa es totalmente combinacional genera dificultades tanto respecto del temporizado como en el cálculo de la potencia estimada. La ausencia de registros ocasiona la existencia de distintos retardos a través de las diferentes rutas de datos posibles, mientras que la ausencia de una señal de reloj y por tanto de una frecuencia de trabajo del circuito, provoca que la herramienta calcule de manera errada la potencia dinámica consumida. Además, la falta de una señal de temporizado única impide la optimización efectiva del circuito por parte de la herramienta.

La manera en la que está diseñada esta etapa usando multiplicadores CSD con coeficientes fijos permite la economía en el área del circuito así como la consecuente disminución en el consumo de potencia; sin embargo, es un diseño estático que no permite alterar la matriz de transformación una vez que se ha realizado la integración física del diseño por lo que cualquier re-entrenamiento del circuito para incorporar nuevos datos (como pueden ser nuevas grabaciones de disparos, etc) implicaría el tener que cambiar el diseño del circuito en sí mismo volviendo obsoleta cualquier versión anterior. Puede buscarse una solución más flexible en su implementación de los coeficientes de transformación aunque esto implique un mayor uso del área del circuito y el posible aumento del consumo de potencia que esto conlleva.

La falta de una prueba o *testbench* previamente establecido para la comprobación lógica del circuito mina significativamente la confianza de los resultados posteriores a la síntesis de esta etapa. Esto debido a que en [9] el análisis se enfoca a la comprobación de los coeficientes LDA y PCA extraídos y utilizados y en sus características para su elección final; sin embargo, una vez que se elige una matriz de transformación para ser implementada no se brindan datos concretos del comportamiento del circuito como tal. Es altamente recomendable definir una estrategia de verificación más exigente para esta etapa.

Capítulo 7

Implementación física de la etapa de codificación.

En este capítulo se presentan los resultados de la síntesis lógica y la implementación física de la etapa de codificación por medio de un árbol k-D. Se incluyen la obtención de la lista de nodos a nivel de compuertas, los resultados de la verificación lógica, la verificación de temporizado y la estimación de consumo de potencia y área para esta etapa.

7.1 Síntesis lógica y verificación de resultados

La etapa de codificación es realizada a través de la implementación de un árbol k-D, el cual como se explica en el capítulo 3 es la implementación en FPGA de un algoritmo de búsqueda, que codifica datos de 16 bits en 8 dimensiones e indica cuál de las 32 palabras código es la más cercana o similar al dato recibido.

Esta etapa está compuesta principalmente por una máquina de estados que ejecuta el algoritmo de búsqueda y por módulos auxiliares en su mayoría registros y flip flops explícitos. En el diseño inicial, todos los elementos secuenciales eran activos con el flanco negativo del reloj, cuando lo usual en diseño de hardware es que sean activos con el flanco positivo del reloj; esta cualidad del diseño causó inconvenientes en el momento de realizar la síntesis. Por este motivo se analizó el diseño y no se encontró ningún motivo de peso por el cual no se pudieran cambiar todos los elementos secuenciales en activos por flanco positivo.

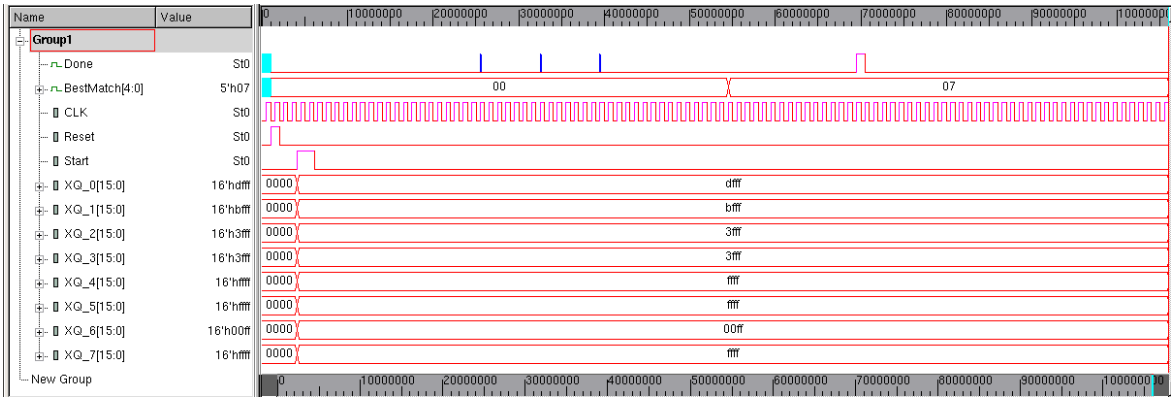

Figura 7.1: Resultado de la simulación de la etapa después de la síntesis.

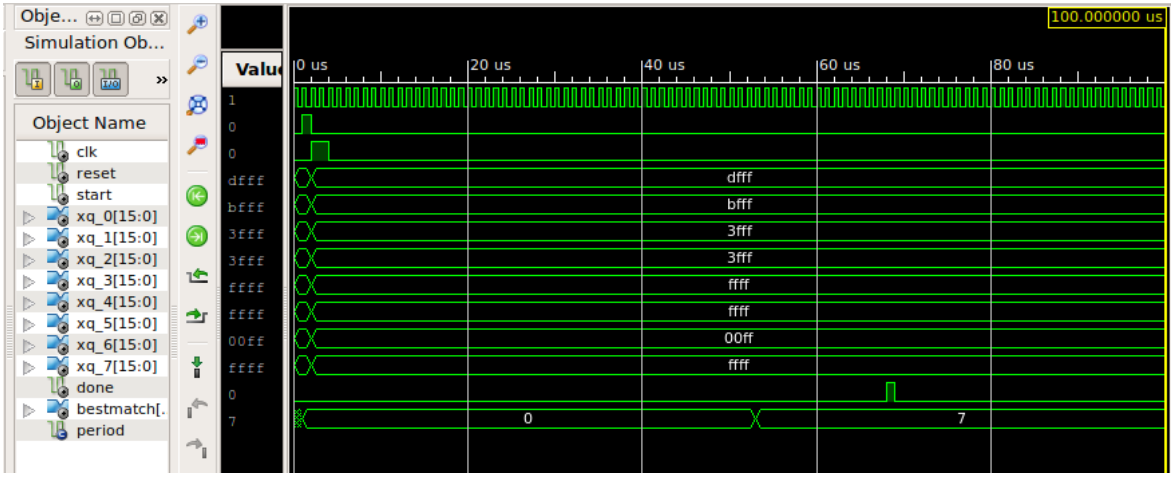

Figura 7.2: Resultado de la simulación original de la etapa.

Una vez realizados los ajustes no se presentaron nuevos inconvenientes durante la realización de la síntesis por los que se debiera volver a modificar el diseño. De esa manera se comprobó que el funcionamiento no se mostró afectado tal y como se puede deducir al comparar la simulación obtenida después de la síntesis, mostrada en la figura 7.1, y la simulación original, mostrada en la figura 7.2, puesto que en ambas simulaciones se obtuvo el mismo valor de salida al cabo del mismo número de ciclos de reloj.

7.2 Estimación del consumo de potencia y área

A partir de los reportes generados por la herramienta *Design Compiler* se puede ver el estimado de área utilizada y de potencia consumida por esta etapa, tal como se presentan en la tabla 7.1.

Tabla 7.1: Estimación de área y consumo de potencia para el módulo de codificación (KD Tree).

Área (μm^2)	33312,9602
Potencia dinámica (μW)	6,6265
Potencia estática (nW)	39,0507

Estos resultados son por lo menos un orden de magnitud menores que la potencia consumida por la implementación en FPGA, los cuales están en el orden de los mW.

7.3 Obtención del trazado físico

En la figura 7.3 se muestra el trazado obtenido. Nuevamente se confirma que el funcionamiento lógico del circuito es el correcto al realizar la simulación correspondiente utilizando la lista de nodos a nivel de compuertas con información de capacitancias y resistencias parásitas generada y siendo esta idéntica a la obtenida en la síntesis.

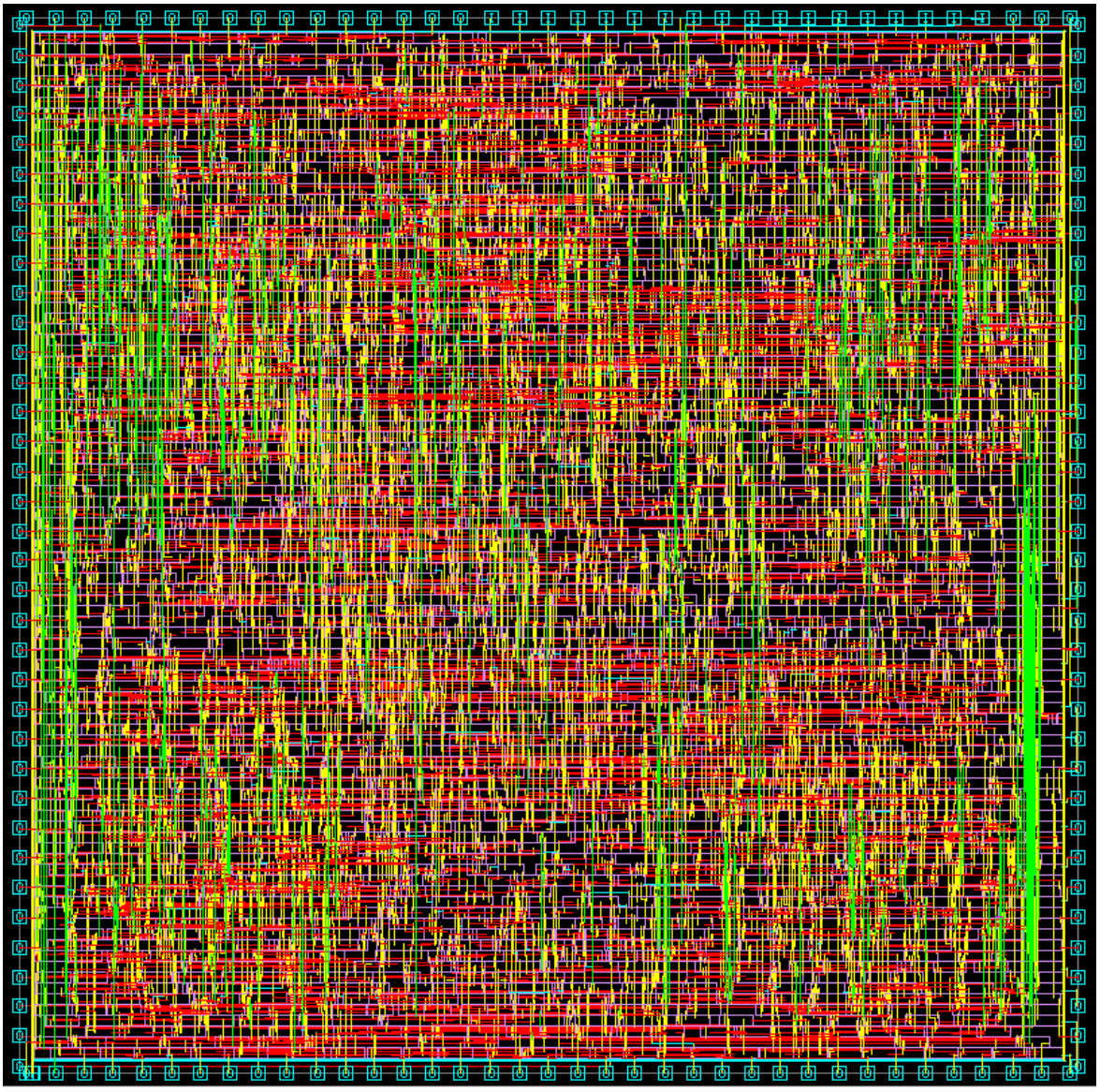

Figura 7.3: Trazado físico resultante de la implementacón del módulo de codificación.

7.4 Verificación de temporizado

A partir de los resultados obtenidos con la herramienta *Primetime*, los cuales se resumen en la tabla 7.2, se extrajo que para todas las rutas críticas el tiempo de llegada del dato es mucho menor del tiempo requerido para cumplir con esa ruta, resultando en un *slack* positivo. El diseño cumple, por tanto, con las especificaciones de temporizado para las condiciones de prueba, las cuales fueron realizadas con una frecuencia de reloj de 1MHz. Aunque esta etapa se probó a una frecuencia relativamente alta, se espera que la implementación final de esta etapa trabaje con una frecuencia en el orden de las decenas de kHz, por lo que se garantiza el cumplimiento del temporizado.

7.5 Consumo de potencia del trazado físico

A partir del reporte brindado por la herramienta, se extrajeron los datos estimados de potencia consumida por el sistema una vez realizada la implementación física, que se exponen en la tabla 7.3.

Tabla 7.2: Rutas críticas para el temporizado del módulo de codificación (KD Tree).

Punto de inicio	Punto final	Tiempo requerido (ns)	Tiempo de llegada (ns)	*Slack* (ns)
NodoDirReg1/ Data_reg_0_	BOB1/FlipFlopC/ Salida_reg	1003,03	32,46	970,58
NodoDirReg1/ Data_reg_0_	BOB1/REGSUM1/ Data_reg_18_	1002,92	35,35	967,57
NodoDirReg1/ Data_reg_0_	BWB1/FlipFlopA/ Salida_reg	1003,03	27,32	975,72
BestMatchM1/ PQDreg/Data_reg_0_	BestMatchM1/ PQDreg/ Data_reg_18_	1002,92	11,02	991,90
NodoDirReg1/ Data_reg_0_	BestMatchM1/ REGDISSIM/ Data_reg_18_	1002,92	29,80	973,12

Tabla 7.3: Estimación de área y consumo de potencia para el módulo de codificación (KD Tree).

Área (μm^2)	33312,9602
Potencia dinámica (μW)	6,6336
Potencia estática (nW)	39,0546

Al comparar estos datos con los obtenidos de la síntesis se corrobora que el área de implementación se mantiene mientras que se nota un ligero aumento de la potencia consumida, siendo la potencia dinámica un 0,11 % mayor y la potencia estática un 0,001 % mayor.

7.6 Conclusiones

En el diseño de circuitos digitales orientado a síntesis, cuando se utilizan elementos secuenciales (como *Flip-Flops* por ejemplo), es común utilizar el flanco positivo del reloj como evento de cambio, en general debido a la mayor disponibilidad de registros de flanco positivo en las bibliotecas de síntesis (o en las FPGA). Si es necesario introducir un retardo de medio periodo se puede utilizar la inversión del reloj en lugar de usar el flanco negativo.

Capítulo 8

Implementación física de la etapa de modelos ocultos de Markov.

En este capítulo se presentan los resultados de realizar la síntesis lógica y la implementación física de la etapa de modelos ocultos de Markov (HMM). Estos resultados incluyen la verificación lógica, la verificación de temporizado y la estimación de consumo de potencia y área estimada para esta etapa.

8.1 Síntesis lógica y verificación

La etapa de clasificación del estado del bosque utilizando modelos ocultos de Markov es la etapa más compleja pues utiliza un procesador de arreglo de matrices o MAP, que tiene estructuras dedicadas propias, además de contar con una unidad de punto flotante, motivos por los cuales se le prestó una atención mayor a la hora de su preparación.

Como primer punto se tuvo que añadir una biblioteca de trabajo distinta de la que se usa por defecto (en general la biblioteca *work*) ya que para la unidad de punto flotante se requiere implementar una biblioteca a la medida, tal como se expone en el capítulo 3. Esto requirió modificaciones en el *script* de síntesis para agregar la biblioteca y compilar los módulos en el orden correcto; el ejemplo 2 muestra parte de estas modificaciones. En este punto se detectó una incompatibilidad producida por el nombre de una función dentro de la biblioteca, que probablemente entraba en conflicto con alguna palabra reservada de la herramienta, motivo por el cual se decidió modificar el nombre de la función por uno ligeramente distinto.

Debido a que el diseño utiliza múltiples registros y otros elementos secuenciales que requieren de ser iniciados en un valor conocido fue necesario modificarlos para incluir una señal de *reset*, la cual pudiera inicializar todos estos elementos en un valor inicial conocido así como el permitir reiniciar todo el módulo en caso de ser necesario. Esta medida puede pasarse por alto cuando se realiza un diseño orientado hacia FPGA pues este dispositivo puede en ocasiones inicializar los registros y otros elementos similares a un cero lógico, mientras que la integración física no soporta esta opción. En la figura 8.1 se muestra el resultado de la simulación que muestra esta problemática, donde se aprecia como las señales relevantes nunca salen del estado desconocido en el que inician debido a que no son forzadas a ello.

El diseño contaba además con algunos módulos los cuales no son utilizados todo el tiempo, por lo que utilizaba la técnica de bloqueo de reloj o *clock gating* para contro-

Ejemplos 2 Modificación del script de síntesis para incluir la nueva biblioteca de trabajo *fplib*.

```
...
#Definir biblioteca fplib
define_design_lib fplib -path ./work/fplib
#Primero se analiza los paquetes y archivos de la biblioteca fplib
analyze -work fplib -format vhdl {pkg_misc.vhd \
pkg_lnsadd.vhd \
pkg_fpdiv.vhd \
pkg_fpsqrt.vhd \
pkg_fpmul.vhd \
pkg_fplib_fp.vhd \
pkg_fplib_lns.vhd \
pkg_fplib.vhd \
pkg_fpadd.vhd \
pkg_fp_misc.vhd \
fplib.vhd \
...
```

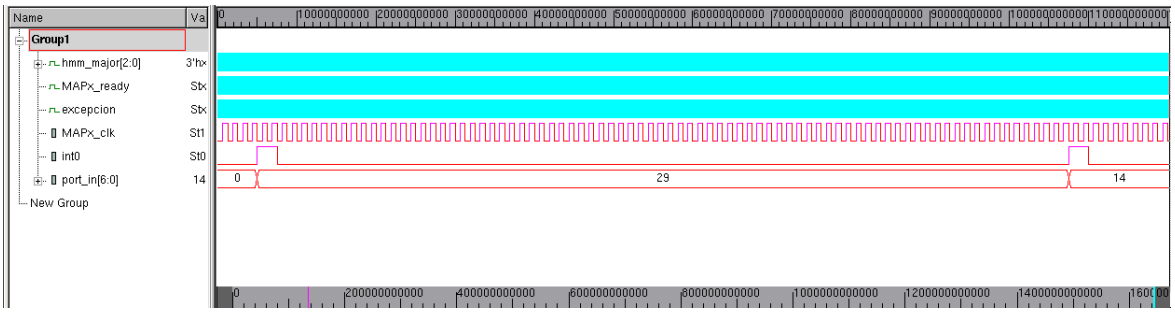

Figura 8.1: Resultado de la simulación del diseño original tras la síntesis; se puede ver como las señales superiores nunca salen del estado de alta impedancia.

larlos. Sin embargo, al realizar la comprobación lógica se comprobó que esto conducía a problemas de sincronía entre los distintos módulos, ocasionando que algunos datos se perdieran y por ende causando el incorrecto funcionamiento de toda la etapa, como se muestra en la figura 8.2.

Por este motivo se decidió implementar el uso de señales de habilitación o *enable*, las cuales cumplen con la misma función pero de un modo más transparente al no interferir con la señal de reloj utilizada en toda la etapa. Una vez implementadas estas medidas en el diseño se procedió a la síntesis de la etapa, obteniendo los resultados de la simulación que se observan en la figura 8.3 los cuales se pueden comparar con la simulación original presentada en la figura 8.4, comprobando que la señal de salida (hmm_major) cambia de valor con cada cadena de 4 símbolos que es ingresada a la etapa. Puesto que estos valores de salida son los mismos para ambas simulaciones se comprueba que se obtiene el mismo funcionamiento lógico para esta etapa.

Finalmente se obtuvo la lista de nodos a nivel de compuerta de la cual se presenta el esquemático correspondiente en la figura 8.5.

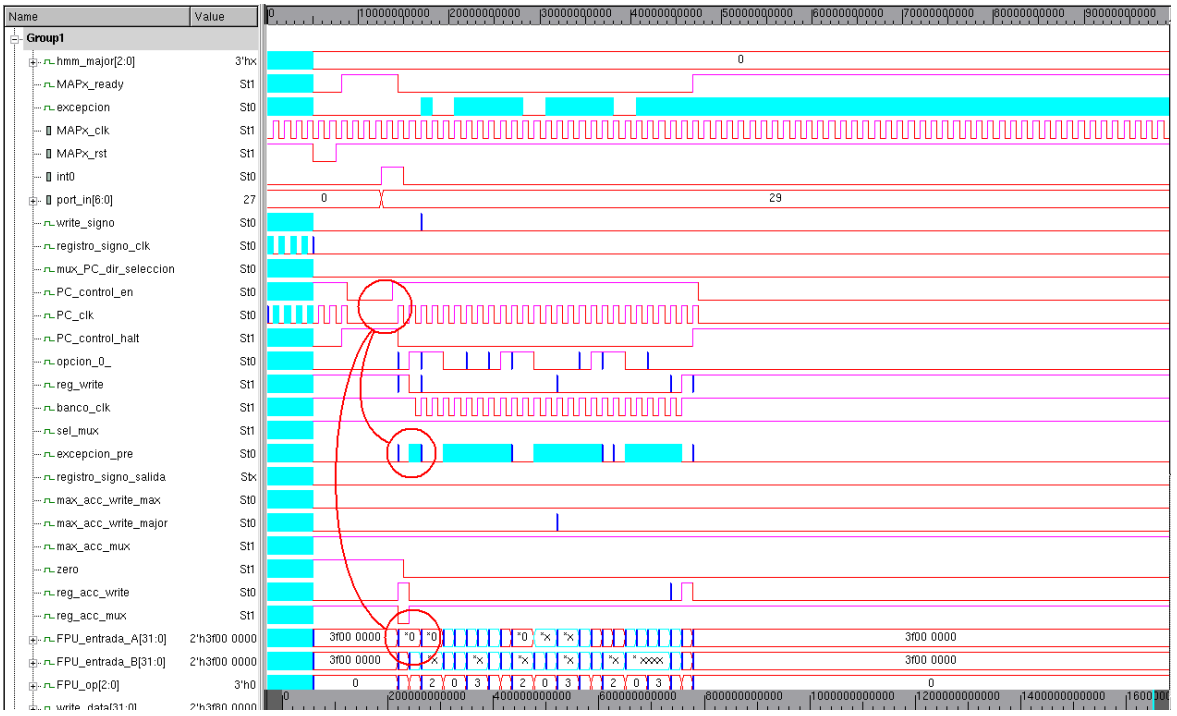

Figura 8.2: Resultado de la simulación después de agregar la señal de reset al diseño original tras la síntesis. Se señala una señal de reloj que es manejada de manera combinacional y las señales de datos que se ven afectadas.

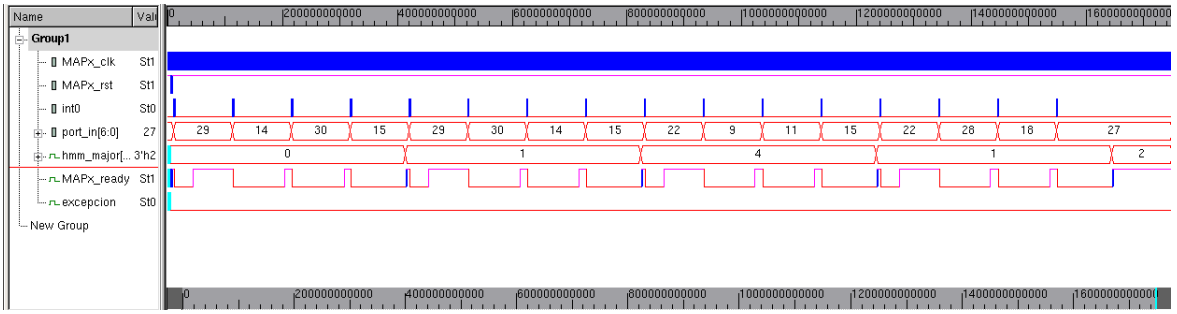

Figura 8.3: Resultado de la simulación después de agregar señales de reset y de enable al diseño original tras la síntesis. Se puede ver como la señal de salida (hmm_major) cambia de manera adecuada con cada cadena de 4 símbolos que es ingresada.

8.2 Estimación del consumo de potencia y área

A partir de los reportes generados por la herramienta *Design Compiler* se puede ver en la tabla 8.1 el estimado de área utilizada y de potencia consumida por esta etapa.

Estos resultados son por lo menos un orden de magnitud menores que la potencia consumida por la implementación en FPGA. Además se puede ver que la potencia dinámica es mucho menor que la potencia dinámica, que resulta de la baja frecuencia de trabajo para el MAP: 800Hz.

8.3 Obtención del trazado físico

La obtención del trazado físico de la etapa se obtuvo utilizando la herramienta *IC Compiler*. En la figura 8.6 se muestra el trazado obtenido. Nuevamente se confirma que

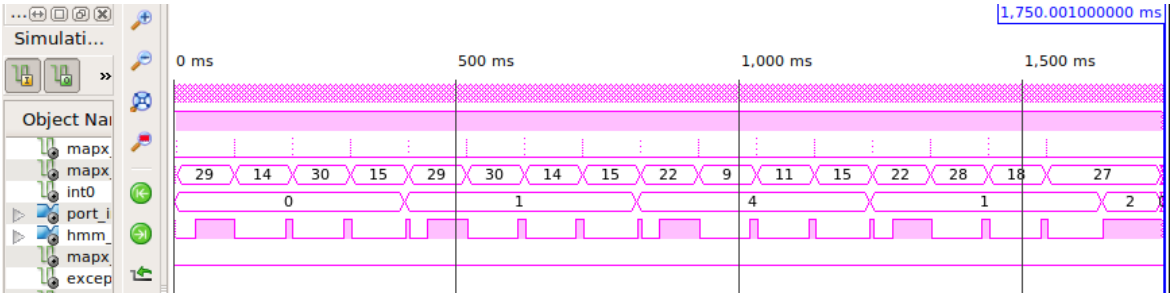

Figura 8.4: Resultado de la simulación original de la etapa. Se puede ver como la señal de salida (hmm_major) cambia con cada cadena de 4 símbolos que es ingresada.

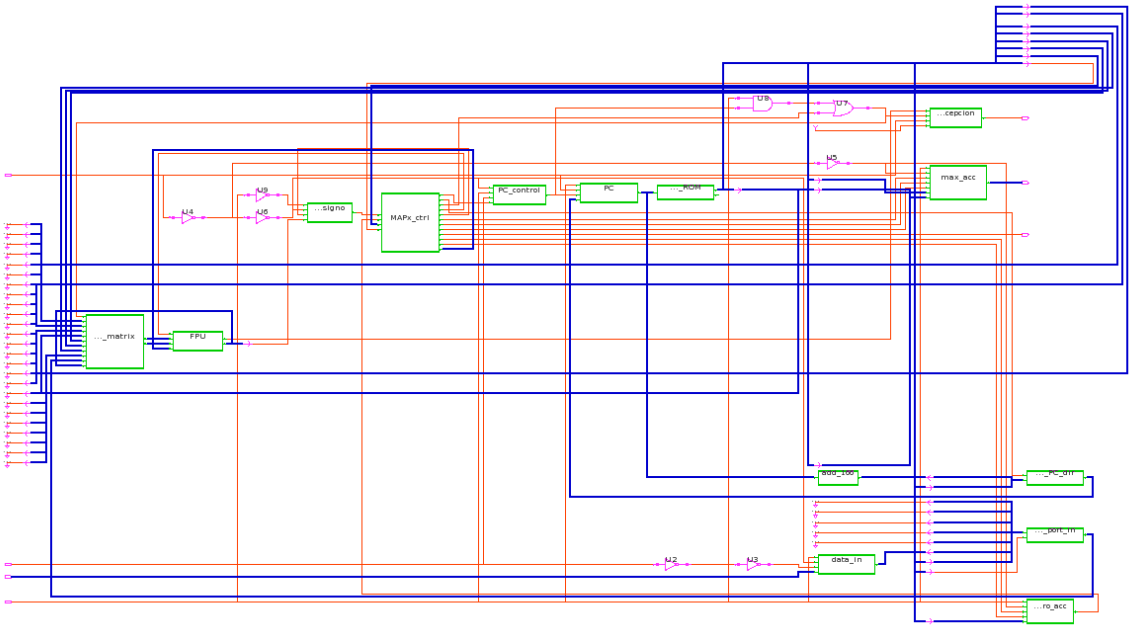

Figura 8.5: Esquemático resultante de la síntesis del módulo de modelos ocultos de Markov (HMM).

el funcionamiento lógico del circuito es el correcto al realizar la simulación utilizando la lista de nodos a nivel de compuertas con información de capacitancias y resistencias parásitas generada y siendo esta idéntica a la obtenida en la síntesis.

8.4 Verificación de temporizado

A partir de los resultados obtenidos con la herramienta *Primetime*, los cuales se resumen en la tabla 8.2, se obtuvo que para todas las rutas críticas el tiempo de llegada del dato es mucho menor del tiempo requerido para cumplir con esa ruta, resultando en

Tabla 8.1: Estimación de área y consumo de potencia para el módulo de modelos ocultos de Markov (HMM).

Área (μm^2)	147274,5618
Potencia estática (nW)	193.3518
Potencia dinámica (nW)	7,6571
Potencia total (nW)	201,0089

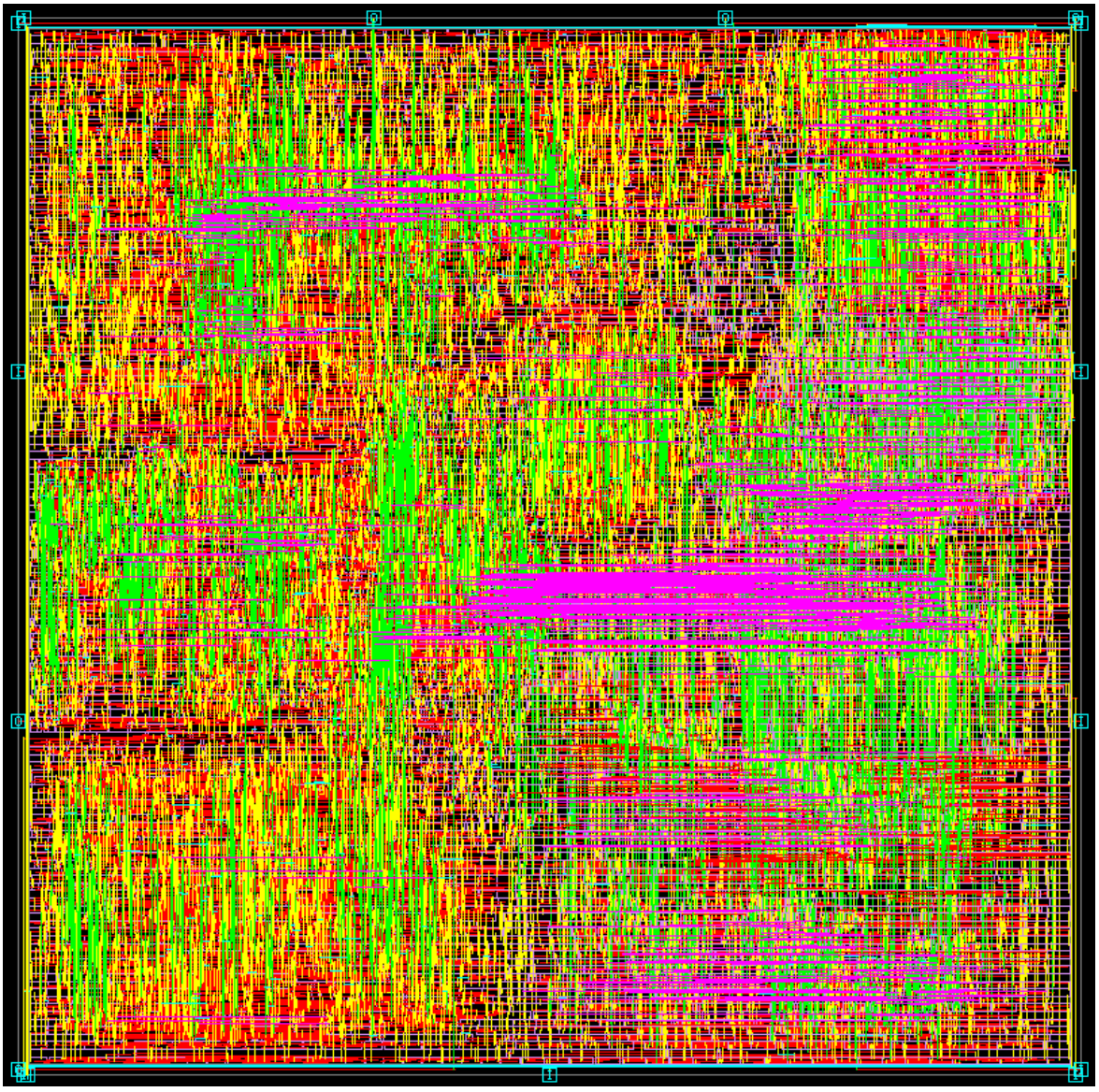

Figura 8.6: Trazado físico resultante de la implementacón del módulo de modelos ocultos de Markov.

un *slack* positivo. El diseño cumple, por tanto, con las especificaciones de temporizado para las condiciones de prueba, las cuales fueron realizadas con una frecuencia de reloj de 800Hz.

8.5 Consumo de potencia del trazado físico

A partir del reporte brindado por la herramienta, se extrajeron los datos estimados de potencia consumida por el sistema una vez realizada la implementación física, tal como se exponen en la tabla 8.3.

Al comparar estos datos con los obtenidos de la síntesis se corrobora que el área de implementación se mantiene mientras que se nota una ligera diferencia de la potencia consumida, siendo la potencia dinámica un 0,019 % mayor y la potencia estática un 0,0064 % menor.

Tabla 8.2: Rutas críticas para el temporizado del módulo de modelos ocultos de Markov (HMM).

Punto de inicio	Punto final	Tiempo requerido (ns)	Tiempo de llegada (ns)	*Slack* (ns)
registro_acc/reg0/ registro_MAPx_cuenta_ reg_0_	PC/registro_MAPx_ cuenta_reg_6_	1250002,90	625007,52	624995,38
PC/registro_MAPx_ cuenta_reg_3_	PC_control/state_reg	625003,15	11,08	624992,06
PC/registro_MAPx_ cuenta_reg_3_	banco_reg_matrix/ reg_pila_MAPx/reg_0/ reg_MAPx_cuenta_reg_0_	625002,43	242,29	624760,12
PC/registro_MAPx_ cuenta_reg_3_	banco_reg_matrix/ reg_pila_MAPx/reg_0/ reg_MAPx_cuenta_reg_0_	625002,43	242,29	624760,12
PC/registro_MAPx_ cuenta_reg_3_	flag_signo/flag_cuenta_reg	625002,92	44,74	624958,19
banco_reg_matrix/ reg_pila_MAPx/reg_13/ reg_MAPx_cuenta_reg_23_	flag_excepcion/ flag_cuenta_reg	1250002,92	625226,23	624776,69
PC/registro_MAPx_ cuenta_reg_0_	max_acc/reg_major/ registro_MAPx_cuenta_ reg_2_	625002,92	10,68	624992,25
PC/registro_MAPx_ cuenta_reg_3_	flag_signo/flag_cuenta_reg	625002,92	44,74	624958,19
PC/registro_MAPx_ cuenta_reg_3_	max_acc/reg_max/ registro_MAPx_cuenta_ reg_2	625002,92	11,79	624991,12
PC/registro_MAPx_ cuenta_reg_3_	registro_acc/reg0/ registro_MAPx_cuenta_ reg_3_	625003,02	10,72	624992,31

8.6 Conclusiones

El uso de una bilioteca personalizada, en este caso para el manejo de las operaciones en punto flotante, debe realizarse con cuidado y se debe tener en cuenta la posibilidad de tener que modificar su contenido nuevamente para evitar el uso de palabras reservadas por la herramienta de síntesis, las cuales pueden no haber presentado problemas en etapas de desarrollo anterior.

En todo diseño realizado teniendo en mente una implementación física se debe de cuidar el hecho de que todos los elementos de memoria (como registros), contadores y otros elementos que cuentan con estados iniciales, deben de tener una forma de regresar a un estado inicial conocido; siendo la más utilizada una señal de *reset*. Esto también permite reiniciar cualquier proceso que se esté ejecutando si así fuera el caso.

Al utilizar la técnica de *clock gating* para deshabilitar unidades que no están en uso todo el tiempo, no es conveniente realizarlo manipulando directamente la señal de reloj que maneja la unidad sin seguir los lineamientos generales para esta técnica, pues esto puede provocar retardos en dichas señales de reloj que causan pérdida de datos o

Tabla 8.3: Estimación de área y consumo de potencia para el módulo de modelos ocultos de Markov (HMM).

Área (μm^2)	117731,5221
Potencia estática (nW)	193,3394
Potencia dinámica (nW)	7,6586
Potencia total (nW)	200,9980

sincronía con otras unidades y por ende el funcionamiento erróneo de toda la etapa. Se recomienda añadir señales de habilitación o *enable* a estas unidades, con las cuales se logra el mismo efecto de deshabilitar una unidad pero de una manera transparente con respecto al reloj. Otra opción para realizar esta técnica es dejar que la misma herramienta *Desing Compiler* se encargue de introducir el control del reloj.

Capítulo 9

Implementación física de la etapa general del SiRPA.

En este capítulo se presentan los resultados de realizar la síntesis lógica y la implementación física de la etapa general del SiRPA la cual abarca todas las etapas expuestas en capítulos anteriores. Se incluyen la obtención de la lista de nodos a nivel de compuertas, los resultados de la verificación lógica, la verificación de temporizado y la estimación de consumo de potencia y área estimada para esta etapa.

9.1 Síntesis lógica y verificación

El módulo general del SiRPA conlleva la unión de las etapas expuestas en capítulos anteriores. En orden de entrada hacia salida estos serían: el módulo del banco de filtros digitales, el módulo reductor de dimensiones, el módulo de codificación, y el módulo de modelos ocultos de Markov o HMM. Aunque cada etapa por separado funciona de manera correcta en su propio entorno se deben de realizar algunos ajustes para que trabajen juntas, en especial cuando han sido desarrollados por separado.

El principal obstáculo en el desarrollo de este módulo es la falta de compatibilidad de la etapa de codificación con la etapa anterior, pues el árbol k-D que lo conforma fue realizado teniendo como base un algoritmo que trabaja clasificando datos en 8 dimensiones en lugar de los datos en sólo 3 dimensiones que proporciona precisamente la etapa de reducción de dimensiones. Por ende es necesario realizar el rediseño de la etapa de clasificación para que sea posible su uso en este contexto; sin embargo, debido a las limitaciones de tiempo de este proyecto y a la falta de documentación adecuada de la etapa, no fue posible realizar este rediseño. Los resultados aquí presentados fueron enfocados hacia la capacidad de interacción entre las distintas etapas y su capacidad de ser procesadas como una unidad, de manera que fuera posible realizar la síntesis y la posterior integración física de esta unidad utilizando el marco CAD planteado.

Entre los aspectos a tomar en cuenta para la interconección de las etapas está la del tipo y tamaño de las señales que son utilizadas por dos o más etapas. En este sentido se tuvieron que realizar conversiones entre tipos de señales, en particular entre la etapa de reducción de dimensiones y la de codificación, y se debió ajustar el tamaño de la salida de la etapa de codificación que ingresa a la etapa de modelos ocultos de Markov.

Por otro lado, se tiene que las etapas de codificación y de modelos ocultos de Markov pueden integrarse adecuadamente entre ellas principalmente debido a que las señales

de salida de la etapa del árbol k-D incluye una señal de "dato listo", la cual puede utilizarse directamente como señal indicadora del dato de entrada en la unidad MAP; así mismo, las salidas del MAP incluyen una señal que indica su disponibilidad para el ingreso de un nuevo dato de entrada, la cual puede usarse como señal indicadora para el árbol k-D para que clasifique el siguiente símbolo a partir de los datos de entrada.

Este funcionamiento de estas dos etapas de manera conjunta en lo que podría denominarse como "automático" está lejos de ser lo ideal pues los modelos ocultos de Markov trabaja en base a cadenas de símbolos, específicamente de 4 símbolos para este desarrollo en particular, de manera que ocupa un indicador de cuál es el primer símbolo de la cadena para empezar su análisis, mientras que la etapa de codificación trabaja cada símbolo individualmente a partir del dato presente en la entrada por lo que no se tendría la posibilidad de obtener un indicador directo de cuál es el primer símbolo de la cadena. Se requiere entonces de una etapa de predicción que pueda brindar un indicador, generalmente probabilístico, que indique cuando se está en presencia del posible inicio de una cadena; o de un detector inicial que de alguna manera ya sepa que se está en presencia de algo parecido a un disparo o una motosierra, y que dé la orden de arranque al resto del sistema.

En la figura 9.1 se puede apreciar el esquemático obtenido de la sintetización del módulo general, tal como se expuso anteriormente, utilizando la herramienta *Design Compiler*, mostrando los distintos módulos que componen esta etapa.

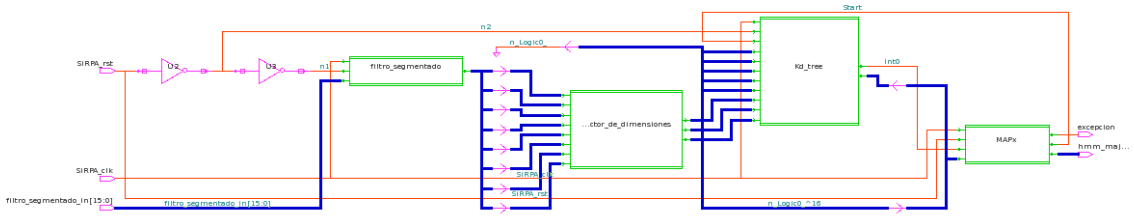

Figura 9.1: Esquemático resultante de la síntesis del módulo general del sistema.

9.2 Estimación del consumo de potencia y área

A partir de los reportes generados por la herramienta *Design Compiler* se puede ver el estimado de área utilizada y de potencia consumida por esta etapa, que se presentan en la tabla 9.1.

Tabla 9.1: Estimación de área y consumo de potencia para el módulo general del sistema.

Área (μm^2)	296916,4830
Potencia estática (nW)	442,0986
Potencia dinámica (nW)	382,6417
Potencia total (nW)	824,7403

Estos resultados son por lo menos un orden de magnitud menores que la potencia consumida por la implementación en FPGA, los cuales están en el orden de los mW.

9.3 Obtención del trazado físico

La obtención del trazado físico de la etapa se obtuvo utilizando la herramienta *IC Compiler*, habiendo realizado las correcciones necesarias para la síntesis del circuito. En la figura 9.2 se muestra el trazado obtenido.

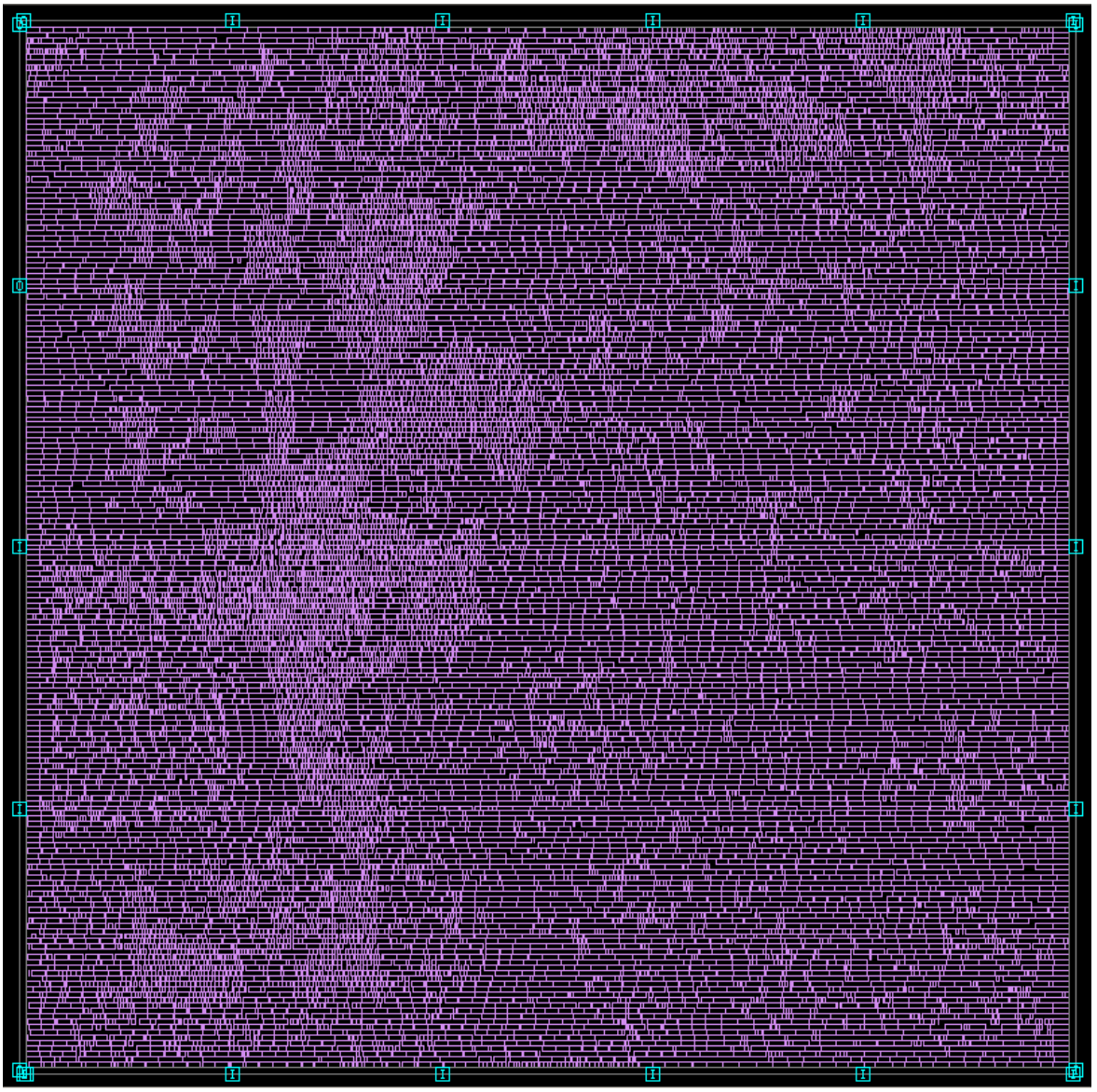

Figura 9.2: Trazado físico resultante de la implementacón del módulo general del sistema.

9.4 Verificación de temporizado

A partir de los resultados obtenidos con la herramienta *Primetime*, los cuales se resumen en la tabla 9.2, se extrajo que para todas las rutas críticas el tiempo de llegada del dato es mucho menor del tiempo requerido para cumplir con esa ruta, resultando en un *slack* positivo. El diseño cumple, por tanto, con las especificaciones de temporizado para las condiciones de prueba, las cuales fueron realizadas con una frecuencia de reloj de 44kHz.

9.5 Consumo de potencia del trazado físico

A partir del reporte brindado por la herramienta, se extrajeron los datos estimados de potencia consumida por el sistema una vez realizada la implementación física; estos se exponen en la tabla 9.3.

Al comparar estos datos con los obtenidos de la síntesis se corrobora que el área de implementación se mantiene mientras que se nota un ligero aumento de la potencia consumida, siendo la potencia dinámica un 1,4 % mayor y la potencia estática un 0,0036 % mayor.

9.6 Conclusiones

Cuando se trabaja con etapas desarrolladas de manera independiente es recomendable crear un estándar para las interfaces entre los distintos módulos, de manera previa a su desarrollo. De esta manera, al establecer convenciones de trabajo *a priori* se sabe que no se tendrá incompatibilidades al momento de realizar la fusión de las unidades aunque estas hayan sido desarrolladas de manera independiente.

Debido a la incompatibilidad de la etapa de codificación con la etapa anterior de reducción de dimensiones, no se logra una implementación satisfactoria de la etapa general del SiRPA. Realizar un rediseño del árbol k-D utilizado por esta etapa está fuera de los alcances planteados para este proyecto, debido a que requiere de una revisión detallada de los detalles teóricos de su implementación anterior, los cuales no existían al momento de la realización de este proyecto.

A pesar de ser incompatible con la etapa anterior, la etapa de codificación es totalmente compatible con la etapa posterior (la de clasificación) de manera tal que ambas pueden funcionar juntas en una especie de modo "automático" pues poseen señales de salida que controlan a la otra etapa de manera adecuada. Sin embargo, debido a que la clasificación por medio de los modelos ocultos de Markov requiere analizar cadenas de símbolos esta etapa necesita saber cuándo el símbolo proporcionado por el codificador es el inicio de una cadena, para lo cual el diseño actual no está preparado.

Es posible utilizar una sola frecuencia de reloj para toda la etapa; sin embargo, es recomendable que las etapas de codificación y la de clasificación tengan una frecuencia distinta a la de las etapas previas, pues esto permite que realicen el análisis de cada dato de entrada actual antes de que el siguiente dato de entrada esté disponible. Esto implica crear una estructura de relojes para manejar cada unidad de manera que le permita el procesamiento del dato correspondiente y la sincronía necesaria con las demás unidades.

Tabla 9.2: Rutas críticas para el temporizado del módulo general del sistema.

Punto de inicio	Punto final	Tiempo requerido (ns)	Tiempo de llegada (ns)	*Slack* (ns)
Kd_tree/NodoDirReg1/ Data_reg_0_	Kd_tree/BOB1/FlipFlopC/ Salida_reg	22678,74	32,95	22645,79
Kd_tree/NodoDirReg1/ Data_reg_0_	Kd_tree/BOB1/REGSUM1/ Data_reg_18_	22678,62	35,88	22642,74
Kd_tree/BestMatchM1/ PQDreg/Data_reg_0_	Kd_tree/BestMatchM1/ CentroideReg1/Data_reg_4_	22678,62	10,83	22667,79
Kd_tree/BestMatchM1/ Contador1/Cuenta_reg_0_	Kd_tree/BestMatchM1/ Contador1/Cuenta_reg_1_	22678,61	5,36	22673,25
Kd_tree/NodoDirReg1/ Data_reg_0_	Kd_tree/BestMatchM1/ REGDISSIM/Data_reg_18_	22678,62	29,50	22649,12
Kd_tree/presente_reg_1_	Kd_tree/RegEntrada1/ Data_reg_127_	22678,62	7,42	22671,20
MAPx/registro_acc/reg0/ registro_MAPx_cuenta_reg_0_	MAPx/PC/registro_MAPx_ cuenta_reg_6_	22678,60	11345,31	11333,28
MAPx/PC/ registro_MAPx_cuenta_reg_0_	MAPx/PC_control/state_reg	11340,47	10,47	11330,01
MAPx/PC/ registro_MAPx_cuenta_reg_0_	MAPx/banco_reg_matrix/ reg_pila_MAPx/reg_0/ reg_MAPx_cuenta_reg_0_	11340,38	245,93	11094,45
MAPx/PC/ registro_MAPx_cuenta_reg_0_	MAPx/banco_reg_matrix/ reg_pila_MAPx/reg_21/ reg_MAPx_cuenta_reg_31_	11340,37	45,59	11294,78
Kd_tree/presente_reg_2_	MAPx/data_in/ registro_MAPx_cuenta_reg_6_	22678,62	6,97	22671,65
MAPx/banco_reg_matrix/ reg_pila_MAPx/reg_13/ reg_MAPx_cuenta_reg_23_	MAPx/flag_excepcion/ flag_cuenta_reg	22678,59	11566,02	11112,57
MAPx/PC/ registro_MAPx_cuenta_reg_0_	MAPx/flag_signo/ flag_cuenta_reg	11340,82	45,75	11295,07
MAPx/PC/ registro_MAPx_cuenta_reg_3_	MAPx/max_acc/reg_max/ registro_MAPx_cuenta_reg_2_	11340,82	12,29	11328,53
filtro_segmentado/ multi_reloj/contador_reg_0_	filtro_segmentado/ filtro_pipelining/secuencia/ reloj_prioridad_salida_reg_2_	11340,50	5,88	11334,61
filtro_segmentado/ multi_reloj/contador_reg_1_	filtro_segmentado/ multi_reloj/contador_reg_7_	22678,58	6,99	22671,59
filtro_segmentado/ multi_reloj/contador_reg_7_	filtro_segmentado/multi_reloj/ reloj_secuencia_delay_reg_7_	11340,42	5,05	11335,37

Tabla 9.3: Estimación de área y consumo de potencia para el módulo general del sistema.

Área (μm^2)	296916,4830
Potencia estática (nW)	442.1144
Potencia dinámica (nW)	387,9897
Potencia total (nW)	830,1041

Capítulo 10

Conclusiones

El flujo de trabajo para la creación de los archivos necesarios para la creación de un circuito integrado y su correspondiente implementación física a partir de un diseño descrito en un lenguaje de descripción de hardware, es independiente de la tecnología empleada para la realización física del mismo y puede ser adaptado a una tecnología particular a través de la utilización de las bibliotecas de celdas estándar adecuadas. Para este fin sólo es necesario modificar las partes relevantes de los *scripts* o guías de comandos empleados, en las cuales se infiere el uso de dichas bibliotecas. Es necesario, sin embargo, tener el cuidado de añadir todos los archivos relevantes de la biblioteca para su correcto funcionamiento con todas las herramientas utilizadas.

Entre los errores más comunes que se cometen al diseñar un circuito orientado a la implementación física están:

- El empleo de elementos secuenciales activados con el flanco negativo del reloj puede causar problemas graves al momento de realizar la síntesis del circuito debido a la falta de celdas o bloques lógicos en algunas bibliotecas que permitan manejar el temporizado de manera correcta;
- El no utilizar señales de entrada que inicialicen los elementos de memoria causa que los mismos arranquen en estados desconocidos o de alta impedancia por lo que el funcionamiento del circuito se ve comprometido;
- Diseñar etapas completamente combinacionales introduce diferentes retardos al no sincronizar las distintas rutas posibles dentro de la etapa, lo que puede causar que los datos no estén al mismo tiempo en la entrada de la siguiente etapa;
- El deshabilitar módulos que no están en uso constante manipulando directamente la señal de reloj de manera combinacional, en lugar de utilizar señales de habilitación independientes del reloj para este propósito, puede provocar pérdida de sincronía o datos entre los distintos módulos de un circuito alterando su funcionamiento.

La estandarización previa de las señales de entrada y salida de las distintas etapas de un diseño es una práctica muy recomendable, sobre todo en diseños grandes que constan de varias etapas y módulos que deben de interconectarse entre ellos. Este estándar debe establecer el tipo y tamaño de de cada señal de interconexión, sobre todo si el desarrollo de las distintas partes del diseño se realiza de manera independiente y por distintos miembros de un equipo de trabajo, para así evitar complicaciones al momento de unir todos los módulos y etapas juntos.

El diseño de la etapa de reducción de dimensiones optimiza el consumo de potencia y el área utilizada del circuito por medio de la implementación de multiplicadores CSD con coeficientes fijos dentro de una matriz de transformación; sin embargo, desde el punto de vista de reusabilidad este diseño es muy rígido pues la construcción física de los multiplicadores es intrínseca a los coeficientes utilizados. Por ende los coeficientes no pueden ser alterados por un re-entrenamiento de los datos sin alterar el circuito de manera física. Por este motivo puede ser necesario realizar un nuevo diseño de esta etapa que, al no utilizar los multiplicadores CSD, sacrifique una parte del consumo de potencia para obtener flexibilidad para alterar la matriz de transformación en un futuro de ser necesario.

El diseño actual de la etapa de codificación por medio de un árbol k-D debe ser replanteado para que la utilización del espacio octo-dimensional sea adecuada a un espacio tridimensional y de esta forma codificar de manera correcta los 32 símbolos que son utilizados por la etapa siguiente de clasificación. Aunque la reducción en la cantidad de dimensiones del dato de entrada implica una reducción de la utilización de recursos no implica una disminución en la complejidad del algoritmo empleado por lo que se recomienda verificar la base teórica antes de su realización. Así mismo, es deseable el contar con un archivo de prueba o *testbench* más completo, que sea capaz de representar de manera satisfactoria el comportamiento del circuito ante un set de entradas representativo.

Dado que el algoritmo hacia adelante que implementa la etapa de clasificación para poder utilizar los modelos ocultos de Markov utiliza como entrada cadenas de muchos símbolos para su análisis, debe de existir algún mecanismo para identificar (aunque sea de manera probable) el símbolo de inicio correcto de una cadena, a partir del cual los demás símbolos sean ingresados en orden para su análisis. Esto debido a que si se deja el ingreso de cadenas de manera libre, estas pueden provocar, dependiendo de la robustez del diseño, falsos positivos o falsos negativos.

La reducción en las características físicas de los circuitos, tal como el consumo de potencia, se debe no solo al cambio de tecnología como lo es el paso de una implementación en FPGA a una implementación por medio de un ASIC, sino además a que las bibliotecas para la implementación física de 130nm están diseñadas para ser de bajo consumo, por lo que representa una seria mejora sobre las implementaciones anteriores de este proyecto.

Bibliografía

[1] R Elizondo. Algoritmo en vhdl para la búsqueda del vecino más cercano de acuerdo con una medida de desigualdad o distancia utilizando un k-d tree. Cartago, 2010. (document), 3.3.1, 3.3

[2] N Hernández. *Diseño de una red inalámbrica de telecomunicaciones para la protección ambiental en el bosque.* I.T.C.R., Cartago, 2004. 1

[3] A Leiva. *Diseño e implementación de la etapa de detección de disparo de armas de la red inalámbrica de telecomunicaciones para la protección de zonas protegidas.* I.T.C.R., Cartago, 2005. 1.1

[4] V Loaiza. Solo 500 guardaparques protegen bosques y refugios. *La Nación*, mayo 2008. 1

[5] C Ruiz. Caza ilegal sería detectada. *el financiero cr.com*, Junio 2010. 1.2

[6] M Sáenz. *Reconocimiento de patrones acústicos para la protección del ambiente utilizando wavelets y Modelos Ocultos de Markov.* I.T.C.R., Cartago, 2006. 1.1

[7] E Salas. *Reconocimiento de patrones acústicos para la protección del ambiente utilizando wavelets y Modelos Ocultos de Markov.* I.T.C.R., Cartago, 2010. (document), 1.1, 3.4, 3.4.1, 3.4, 3.5

[8] W Salas. *Diseño e implementación de un sensor para la detección de motosierras de la red inalámbrica de telecomunicaciones de la protección ambiental en el bosque.* I.T.C.R., Cartago, 2005. 1.1

[9] M Sequeira. *Módulo de reducción de dimensiones espectrales en un sistema de reconocimiento de patrones acústicos de motosierras y disparos por medio de una implementación en FPGA.* I.T.C.R., Cartago, 2011. 3.2.1, 3.2, 6.1, 6.6

[10] L E Smith. *Reconocimiento digital en línea de patrones acústicos para la protección del ambiente por medio de HMM.* I.T.C.R., Cartago, 2008. 1.1

[11] J Valverde. *Implementación de una metodología para lograr la integración física correcta por construcción (CBC) de un banco de filtros digitales descrito en alto nivel.* I.T.C.R., Cartago, 2011. (document), 1.2, 3.1.1, 3.1, 5.2

Índice alfabético