UNIVERSIDAD DE GRANADA

DEPARTAMENTO DE ELECTRÓNICA Y TECNOLOGÍA DE COMPUTADORES

PROTECCIÓN DE LA PROPIEDAD INTELECTUAL DE CIRCUITOS DIGITALES PARA LA SÍNTESIS DE DESCRIPCIONES DE ALTO NIVEL

TESIS DOCTORAL

ENCARNACIÓN CASTILLO MORALES

Editor: Editorial de la Universidad de Granada
Autor: Encarnación Castillo Morales
D.L.: GR.1774-2008
ISBN: 978-84-691-5473-1

PROTECCIÓN DE LA PROPIEDAD INTELECTUAL DE CIRCUITOS DIGITALES PARA LA SÍNTESIS DE DESCRIPCIONES DE ALTO NIVEL

ENCARNACIÓN CASTILLO MORALES

TESIS DOCTORAL

UNIVERSIDAD DE GRANADA

DEPARTAMENTO DE ELECTRÓNICA Y TECNOLOGÍA DE COMPUTADORES

Granada, junio de 2008

D. Luis Parrilla Roure, Profesor Titular de Universidad, D. Antonio García Ríos, Profesor Titular de Universidad, y D. Antonio Lloris Ruiz, Catedrático de Universidad, todos ellos pertenecientes al Departamento de Electrónica y Tecnología de Computadores de la Universidad de Granada

CERTIFICAN que el trabajo de investigación que se recoge en la presente Memoria, titulada "Protección de la Propiedad Intelectual de circuitos digitales para la síntesis de descripciones de alto nivel" y presentada por Dña. Encarnación Castillo Morales para optar al grado de Doctor por la Universidad de Granada con Mención de Doctorado Internacional, ha sido realizado en su totalidad bajo su dirección en el Departamento de Electrónica y Tecnología de Computadores de la Universidad de Granada.

Granada, a 2 de Junio de 2008

Fdo. Luis Parrilla Roure Fdo. Antonio García Ríos Fdo. Antonio Lloris Ruiz

Directores de la Tesis

AGRADECIMIENTOS

Deseo expresar mi agradecimiento a los profesores Dr. D. Luis Parrilla Roure, Dr. D. Antonio García Ríos y Dr. D. Antonio Lloris Ruiz por su dirección y asesoramiento, gracias a los cuales ha sido posible la realización de la investigación recogida en esta memoria. Asimismo quiero agradecer su apoyo y la estrecha colaboración que hemos mantenido durante estos años.

A los profesores Dr. D. Pedro García Fernández y D. Daniel González Castro, por su ayuda, colaboración y amistad.

Al profesor Dr. D. Uwe Meyer-Bäse, de la Florida State University (Tallahassee, EE.UU.) por todo su apoyo y colaboración, haciéndome sentir entre amigos desde el primer momento, así como por poner a mi disposición todos los medios de su grupo de investigación.

Al resto de miembros del Departamento de Electrónica y Tecnología de Computadores de la Universidad de Granada, especialmente a los profesores Dr. D. Carlos Sampedro Matarín y D. Miguel Ángel Carvajal Rodríguez y a Dña. María Balaguer Jiménez por su especial apoyo durante todos estos años.

Tampoco puedo dejar de agradecer especialmente a Beni, a mi familia y amigos su apoyo constante y su paciencia conmigo.

Finalmente, destacar que este trabajo ha sido financiado en parte a través del Programa de Ayudas para Estancias Breves en el Extranjero concedidas por el Ministerio de Educación, Cultura y Deportes dentro del Programa Nacional de Formación de Profesorado Universitario, y por los proyectos de investigación TIC2002–02227, financiado por el Ministerio de Ciencia y Tecnología, y TEC2007-68074–C02–01/MIC, financiado el Ministerio de Educación y Ciencia.

A Beni

A mis padres y hermanos

Índice

Índice i

Prólogo ix

Prologue xv

Capítulo 1: Introducción 1

1.1. Desarrollo de sistemas digitales 1

1.1.1. Clasificación de los circuitos integrados 2

1.1.2. Tendencias en la implementación de sistemas digitales 4

1.2. Metodologías de diseño actuales 6

1.3. Protección de la Propiedad Intelectual 8

1.4. *Watermarking* para módulos reutilizables 10

1.5. Objetivos de esta tesis 11

1.6. Conclusión 12

Capítulo 2: Desarrollo de sistemas digitales con FPGAs 13

2.1. Introducción 13

2.2. Dispositivos lógicos programables 14

2.2.1. Evolución de los dispositivos programables 15

2.2.2. FPGAs 16
2.3. Técnicas aritméticas para FPGAs 17
2.3.1. Sistemas de representación 17
2.3.2. Optimización de funciones para FPGAs 19
2.3.3. Técnicas especiales 20
2.3.3.1. El Sistema Numérico de Residuos 21
2.3.3.2. Ventajas, limitaciones y aplicaciones 22
2.3.3.3. Implementación de circuitos RNS sobre FPGAs 24
2.4. Desarrollo de sistemas digitales con FPGAs 25
2.4.1. Herramientas CAD 26
2.4.2. Lenguajes de descripción *hardware* 26
2.4.3. Flujo de diseño con HDL 28
2.5. Conclusión 31

Capítulo 3: Diseño basado en la reutilización y protección de módulos IP 33

3.1. Introducción 33
3.2. Diseño basado en la reutilización 34
3.2.1. Tipos de módulos IP 35
3.2.2. Metodología de diseño para la reutilización de módulos IP 36
3.2.3. Reutilización en SoCs e iniciativas de estandarización 38
3.3. Riesgos del diseño basado en la reutilización 42
3.4. Técnicas de protección de la propiedad intelectual 44
3.4.1. Técnicas de disuasión 45
3.4.2. Técnicas de protección 46
3.4.3. Técnicas de detección 47

3.5. Técnicas de *watermarking* 49
3.5.1. Ocultación de la información, esteganografía y *watermarking* 49
3.5.2. *Watermarking* digital 51
3.5.3. Aplicaciones del *watermarking* 53
3.5.4. Requisitos de un sistema de *watermarking* 53
3.5.4.1. Robustez y seguridad 54
3.5.4.2. Imperceptibilidad e indetectabilidad 55
3.5.4.3. Viabilidad del sistema 55
3.5.4.4. Baja probabilidad de error 55
3.6. *Watermarking* para módulos reutilizables 56
3.6.1. Aplicación de técnicas de *watermarking* para bloques IP 57
3.6.2. Objetivos y criterios de evaluación 58
3.6.3. Clasificación y revisión de las técnicas de *watermarking* 61
3.6.3.1. *Watermarking* en el nivel físico 61
3.6.3.2. *Watermarking* en el nivel de síntesis lógica 64
3.6.3.3. *Watermarking* en descripciones de alto nivel 65
3.6.3.4. *Watermarking* en distintos niveles de abstracción 68
3.7. Conclusión 69

Capítulo 4: Nueva técnica de *watermarking* para módulos IP 71

4.1. Introducción 71
4.2. Estructura general 73
4.3. Diseño firmado 75
4.3.1. Preparación de la firma 76
4.3.1.1. Funciones *hash* 79
4.3.1.2. Algoritmos asimétricos de cifrado 83

4.3.1.3. Códigos de corrección de errores 84
4.3.2. Difusión de los bits de la firma 86
4.3.2.1. Estrategia basada en la introducción de los bits de la firma ... 86
4.3.2.2. Estrategia basada en la búsqueda de los bits de la firma 91
4.3.2.3. Consideraciones adicionales 93
4.3.2.4. Elección del tamaño de los bloques de bits de la firma 94
4.3.3. Preparación del diseño para la extracción de la firma 97
4.3.3.1. Secuencia de extracción de la firma 98
4.3.3.2. Lógica para la extracción de la firma 99
4.4. Proceso de validación de la firma 101
4.5. Herramienta automática para la difusión 102
4.5.1. Características principales 103
4.5.2. Función de coste 104
4.5.3. Algoritmos de búsqueda 105
4.5.4. Enfriamiento simulado 106
4.5.5. Algoritmo de Enfriamiento Simulado 108
4.5.6. Aplicación de los algoritmos 111
4.6. Conclusión 115

Capítulo 5: Aplicación y evaluación de las estrategias de protección desarrolladas 117

5.1. Introducción 117
5.2. Filtros CIC de Hogenaour 118
5.2.1. Filtro CIC-RNS 119
5.2.2. Preparación de la firma digital y proceso de difusión 122
5.2.3. Preparación del diseño para la extracción de la firma 124

5.2.4. Proceso de extracción de la firma 126
5.2.5. Resultados experimentales 127
5.3. Filtros FIR 129
5.3.1. Filtro FIR-RNS basado en aritmética de índices 131
5.3.2. Preparación de las firmas digitales y proceso de difusión 133
5.3.2.1. Aplicación de la estrategia de protección EPIBF-LUT 134
5.3.2.2. Aplicación de la estrategia de protección EPBBF-LC 136
5.3.3. Preparación de los diseños para la extracción de la firma 138
5.3.3.1. Lógica de extracción: FSM y multiplexores 138
5.3.3.2. Lógica de extracción basada en LFSR 141
5.3.4. Diseños firmados 144
5.3.5. Resultados experimentales 145
5.4. Transformada 1-D DWT 147
5.4.1. Diseño de la 1-D DWT 148
5.4.2. Preparación de la firma y proceso de difusión 152
5.4.3. Preparación de los diseños para la extracción de la firma 156
5.4.3.1. Lógica de extracción: FSM y multiplexores 156
5.4.3.2. Lógica de extracción basada en LFSR 157
5.4.4. Resultados experimentales 158
5.5. Difusión de los bits de la firma mediante la herramienta automática .. 159
5.5.1. Aplicación al ejemplo de la 1-D DWT 160
5.5.2. Implementación de los diseños protegidos 162
5.6. Evaluación de las estrategias de protección 165
5.7. Conclusión 168

Capítulo 6: Análisis de la seguridad de las técnicas propuestas 171

6.1. Introducción 171

6.2. Clasificación de ataques 173

6.2.1. Introducción no autorizada 174

6.2.2. Detección no autorizada 175

6.2.3. Eliminación no autorizada 175

6.3. Consideraciones sobre el atacante 176

6.4. Seguridad del *watermarking* y criptografía 178

6.4.1. Prevención de detección no autorizada 179

6.4.2. Prevención de introducción no autorizada 180

6.5. Ataques contra *watermarking* para módulos IP 181

6.5.1. Falsificación de firmas 182

6.5.2. Firmas fantasma 183

6.5.3. Introducción de otra firma 184

6.5.4. Ataques de sabotaje 185

6.5.5. Ataques de detección 185

6.6. Probabilidad de la eliminación y detección 186

6.6.1. Eliminación de los bits de la firma 186

6.6.2. Invulnerabilidad de la secuencia de extracción de la firma 192

6.7. Comparación con otras alternativas 194

6.8. Conclusión 197

Capítulo 7: Conclusiones 199

7.1. Introducción 199

7.2. Consideraciones 200

7.3. Principales aportaciones 201

7.4. Líneas de investigación futuras 205

7.5. Conclusión 206

Conclusions 207

Introduction 207

Related work 208

Main contributions 209

Future research lines 212

Conclusion 214

Apéndice A: Familias comerciales de FPGAs 215

A.1. Introducción 215

A.2. Dispositivos de Xilinx 216

A.2.1. Familia Virtex II 217

A.3. Dispositivos de Altera 219

A.3.1. La familia APEX20KC 220

Apéndice B: Aritmética de residuos 223

B.1. Operaciones básicas 223

B.2. Módulos aritméticos 225

B.2.1. Sumadores 225

B.2.2. Multiplicadores 227

B.3. Esquemas de conversión RNS 229

B.3.1. Conversión de binario a RNS 229

B.3.2. Conversión de RNS a binario 231

B.4. Conclusión 235

Apéndice C: Flujo de diseño para ASICs *semi-custom*..... 237

C.1. Design Compiler..... 237

C.2. DesignWare y biblioteca de celdas estándar lsi_10k 240

Bibliografía..... 243

Prólogo

Las últimas décadas han sido testigo de los mayores avances en la historia de la humanidad. La ciencia y la tecnología han permitido la creación de dispositivos cada vez más complejos y sofisticados, siendo los sistemas electrónicos uno de los elementos claves para este desarrollo. La alta eficiencia, los altos niveles de integración y los bajos costes de producción en masa de los sistemas electrónicos han permitido su utilización en multitud de aplicaciones. La electrónica digital es sin duda la más empleada hoy en día. Aunque el uso de sistemas digitales requiere más recursos (silicio) y supone una pérdida de resolución inherente al propio proceso de digitalización, en un número elevado de aplicaciones se prefiere la electrónica digital sobre la analógica, por ser más robusta y más fácil de diseñar.

Actualmente la gran mayoría de las aplicaciones digitales se basan en las tecnologías ASIC (*Aplication Specific Integrated Circuit*). Los inconvenientes fundamentales en el desarrollo de circuito integrados ASICs son los elevados costes de diseño y fabricación y el largo período de tiempo necesario para que estos circuitos lleguen al mercado, comparado con otras tecnologías. Esta solución sólo se puede justificar cuando los volúmenes de producción son elevados, lo que permite amortizar el elevado coste, y el tiempo de desarrollo es asumible. En este sentido, el tiempo de producción de prototipos puede influir negativamente en el ciclo de diseño. Como alternativa a los ASICs tradicionales, en los últimos años han surgido una serie de dispositivos lógicos programables, tales como EPLDs (EPROM *technology-based complex Programmable Devices*) y FPGAs (*Field Programmable Gate Arrays*), que han despertado un enorme interés por sus posibilidades para el desarrollo de sistemas digitales. Estos circuitos se fabrican como dispositivos genéricos y posteriormente son programados por el usuario para realizar una función específica. Así, los dispositivos programables resultan mucho más eficientes para la implementación de aplicaciones

con volúmnes de producción medios. Además, los dispositivos programables pueden ser empleados para el desarrollo y prueba de prototipos que finalmente se producirán como ASICs, a una fracción del costo original. Dados los altos costes de fabricación de una línea de producción, en muchos casos los dispositivos programables son la única alternativa económicamente viable para desarrollar dispositivos de lógica especializada en bajos volúmenes. La posibilidad de reprogramar los chips permite mayor flexibilidad en el diseño, una mayor adaptabilidad de los dispositivos, y crea facilidades para extender, corregir o modificar *a posteriori* las funcionalidades implementadas en los productos finales.

Las tendencias actuales permiten la integración de sistemas digitales completos tanto en ASICs como en un único dispositivo programable, surgiendo así los paradigmas de diseño *System-on-Chip* (SoC) y *System-on-Programmable-Chip* (SoPC). Las metodologías de diseño y herramientas CAD (*Computer Aidded Design*) pretenden seguir este avance de la tecnología de circuitos integrados, intentando cubrir la necesidad de gestionar la complejidad generada por el nivel de integración de los ASICs y FPGAs. El diseño basado en la reutilización de módulos permite ensamblar sistemas complejos a través del uso de pequeños componentes, reduciendo así la complejidad del diseño, el tiempo de desarrollo y el ciclo de prueba y corrección. De esta forma, las estrategias basadas en módulos reutilizables, comúnmente llamados módulos IP (*Intellectual Property*) [KEA98], permiten la optimización de los recursos debido al reducido tiempo y coste de desarrollo. Sin la reutilización de módulos la industria electrónica no sería capaz de afrontar el desafío de fabricar sistemas cada vez mejores, más rápidos y más baratos.

Sin embargo, esta metodología de diseño presenta algunos riesgos. Uno de ellos es el de la protección de la propiedad intelectual (IPP: *Intellectual Property Protection*) [MAC01, KOU05]. Debido a que los bloques IP están diseñados para ser modulares y estar integrados con otros componentes, es posible para un tercero vender uno de estos bloques o módulos como si fuese de su propiedad, sin necesidad de conocer la arquitectura interna o la implementación. Existe, pues, la necesidad de desarrollar técnicas que permitan el intercambio seguro de diseños y la protección de los correspondientes derechos del autor. La tecnología actual también pone al alcance nuevos mecanismos o técnicas de protección de estos derechos sobre la propiedad

intelectual, siendo el uso de las llamadas marcas de agua, o *watermarking*, una de las opciones que más interés está despertando [LAC01, KAH01].

Este trabajo pretende abrir nuevas líneas de investigación sobre el desarrollo y protección de módulos IP para el diseño de sistemas digitales complejos en descripciones de alto nivel. Se proponen en él distintas estrategias de *watermarking* para la protección de módulos IP. Se presentan ejemplos de protección de módulos IP mediante estas estrategias, implementados sobre dispositivos lógicos programables y bibliotecas de celdas estándar, y se analizan los resultados obtenidos en términos de los criterios de evaluación más importantes. Además, se realiza un estudio exhaustivo de los niveles de seguridad que ofrece la nueva propuesta y se comparan las estrategias de protección desarrolladas con otras alternativas existentes en la bibliografía

La memoria de este trabajo está estructurada en siete capítulos, tal y como se describe continuación:

CAPÍTULO 1. Introducción

En este primer capítulo se exponen las principales motivaciones para el desarrollo de este trabajo. En concreto, se presentan las nuevas tendencias en el desarrollo e implementación de sistemas digitales, haciendo especial énfasis en el diseño basado en la reutilización y las implementaciones en dispositivos lógicos programables. A su vez, se muestra que la viabilidad de las metodologías de diseño basadas en la reutilización depende del desarrollo de mecanismos de protección de la propiedad intelectual que permitan reclamar los derechos de autor sobre los módulos reutilizables en caso de ser necesario. En este capítulo se presenta el *watermarking* como una técnica para la protección de la propiedad intelectual de los módulos reutilizables.

CAPÍTULO 2. Desarrollo de sistemas digitales con FPGAs

En el segundo capítulo se realiza una introducción a los dispositivos lógicos programables, que han dado soporte a la computación reconfigurable en el desarrollo de sistemas digitales. Además, se presentan los distintos sistemas numéricos de representación, prestando especial atención al Sistema Numérico de Residuos (*Residue Number System*, RNS) [SZA67, TAY84]. Por último, se analizan las herramientas necesarias para el desarrollo de sistemas digitales con FPGAs y se detallan los pasos

más importantes en el desarrollo de un sistema digital mediante lenguajes de descripción *hardware*.

CAPÍTULO 3. Diseño basado en la reutilización y protección de módulos IP

En el capítulo 3 se analiza la nueva metodología de diseño basada en la reutilización de módulos IP. A pesar de las enormes ventajas que ofrece esta metodología, presenta algunos inconvenientes o limitaciones entre los que se encuentra la protección de la propiedad intelectual de los módulos a compartir. Se hace un estudio de todas las técnicas empleadas para la protección de la propiedad intelectual de módulos reutilizables, entre las que se encuentra el *watermarking*, una de las técnicas más utilizadas en la actualidad. Por último, se revisan las principales técnicas de *watermarking* para módulos IP encontradas en la literatura.

CAPÍTULO 4 . Nueva técnica de *watermarking* para módulos IP

A lo largo de este capítulo se realiza una descripción completa del método de *watermarking* propuesto para módulos IP. Se presentan las ideas preliminares que llevaron al desarrollo de la primera de las estrategias de *watermarking* y se muestrra la evolución que han experimentado estas ideas para llegar a la segunda estrategia de *watermarking*. Por último, se presentará la herramienta *software* desarrollada que automatiza y optimiza la aplicación de la segunda de las estrategias de *watermarking* propuestas.

CAPÍTULO 5. Aplicación y evaluación de las estrategias de protección desarrolladas

Haciendo uso de las ideas expuestas en el capítulo 4, en el 5 se sientan las bases de las estrategias de protección propuestas con la ayuda de algunos ejemplos de aplicación, y se evalúan los resultados obtenidos, principalmente en términos del impacto sobre el área y las prestaciones del sistema y algunos otros criterios de evaluación.

CAPÍTULO 6. Análisis de la seguiridad de las técnicas propuestas

El capítulo 6 recoge el estudio realizado sobre la seguridad de sistemas de *watermarking* en general, y de las estrategias de *watermarking* desarrolladas para módulos IP en particular. Se analiza la invulnerabilidad de los sistemas de protección desarrollados contra los tipos de ataques considerados y se realiza una evaluación y

comparación de las estregias propuestas con otras alternativas de *watermarking* encontradas en la literatura.

CAPÍTULO 7. Conclusiones

Este capítulo recoge las ideas más importantes que aparecen a lo largo de esta memoria, así como las líneas de investigación futuras que quedan abiertas con el desarrollo de este trabajo.

Al final de la memoria se incluyen tres apéndices con los siguientes contenidos:

APÉNDICE A. Familias comerciales de FPGAs

Para desarrollar parte de la invetigación recogida en esta memoria se han empleado familias de FPGAs de Altera y de Xilinx, de forma que este apéndice describe la arquitectura básica y los aspectos más destacados de estos dispositivos.

APÉNDICE B. Aritmética de residuos

En este apéndice se describe con más detalle el RNS y módulos aritméticos basados en esta aritmética que muestran las metodologías clásicas para la realización de la suma y de la multiplicación. Se presentan las principales propuestas para la implementación de los esquemas de conversión de binario a RNS y de RNS a binario, que implica la menor penalización posible en el área y las prestaciones del sistema RNS considerado.

APÉNDICE C. Flujo de diseño para ASICs *semi-custom*

En este apéndice se describe brevemente el flujo de diseño para la síntesis sobre tecnologías VLSI, para lo que se ha usado la herramienta Synopsys Design Compiler, y se proporciona la información básica de la biblioteca de celdas estándar lsi_10k, empleada como tecnología objetivo en los diferetnes ejemplos de ASICs *semi-custom*.

Para finalizar este prólogo, sólo realizar algunas indicaciones para facilitar la lectura de esta memoria:

- la memoria, tal como se ha mostrado, está dividida en diferentes capítulos; estos capítulos, a su vez, constan de diferentes secciones, que se han enumerado con

dos dígitos, uno referente al capítulo y otro a la sección y así serán referidas a lo largo de la memoria. Ejemplo: sección 2.3;

- cuando sea necesario, las secciones se dividirán en apartados que llevarán los mismos dígitos que los correspondientes a la sección en la que se encuadran, acompañados de otro más que se referirá al orden de aparición del apartado dentro de dicha sección. Ejemplo: apartado 3.4.1;
- las figuras y tablas están numeradas por capítulos, con un primer dígito que señala éste y otro que se refiere al elemento en sí. Ejemplos: figura 2.1, tabla 6.2;
- las expresiones matemáticas y ecuaciones se referencian de manera similar con dos dígitos, el primero de ellos referente al capítulo en el que aparecen. Ejemplo: ecuación (4.1);
- por último, las referencias bibliográficas aparecen designadas en el texto con tres letras, correspondientes al primer autor, y dos cifras, que indican el año de publicación; en caso de existir varias referencias con la misma denominación, ésta irá acompañada de una letra al final. Las referencias se hallan ordenadas alfabéticamente al final de esta memoria. Ejemplo: [MAC01], [KAH98a], [KAH98b].

Prologue

The life style of Humankind has experienced a spectacular development in the last decades. The science and the technology have allowed the creation of increasingly complex and sophisticated devices, being the electronic systems one of the key elements for this development. The high efficiency, the high levels of integration and the low costs for mass production of electronic systems have allowed their usage in multitude of applications. In this field, the digital electronic is undoubtedly the most used nowadays. Although the use of digital systems requires a larger number of resources and supposes a loss of resolution inherent to the quantization process itself, digital electronics are preferred rather than analog in a high number of applications, due to their inherent robustness and easier design.

Actually, the great majority of the digital applications are based on the ASIC (Aplication Specific Integrated Circuit) technologies. Compared with other technologies, the most important disadvantages in the development of ASICs are the high design and manufacturing costs and the long time to market. This solution is only feasible for high production volumes, which allow amortizing the high cost and the development time. In this sense, the production time of prototypes can influence negatively into the design cycle. In the last years, logic programmable devices, such as EPLDs (EPROM technology-based complex Programmable Devices) and FPGAs (Field Programmable Gate Arrays), have emerged as an alternative to the traditional ASICs. Due to their possibilities for the development of digital systems, these programmable devices have raised an enormous interest. These circuits are fabricated as generic devices and then programmed by the user in order to realize a specific function. These programmable devices turn out to be much more efficient for the implementation of low and moderate volume applications.

The current trends allow the integration of complete digital systems both in ASICs and in only one programmable device, thus emerging the design paradigms System-on-Chip (SoC) and System-on-Programmable-Chip (SoPC). The design methodologies and CAD tools continue the advance of the integrated circuit technology, trying to cover the need to manage the complexity generated by the integration level of the ASICs and FPGAs. Reuse-based design allows the assembly of a complex system using smaller components, reducing system design, development and desing cycle and test cycle complexity. Thus, the strategies based on the use of reusable modules, commonly referred to as Intellectual Property (IP) cores [KEA98], are enabling the optimization of company resources due to the reduced development time and cost. Without reuse, the electronics industry would struggle to keep pace with the challenge of delivering better, faster and cheaper devices that consumers expect.

However, this design methodology poses significant security risks, one of the main being the intellectual property protection (IPP) of those shared modules. Because IP cores are designed to be modular and integrated with other components, it is possible for a third party to sell an IP block as their own, only requiring to understand the interface and function and without even knowing the internal architecture or implementation. This causes the need for IPP techniques that allow the safe exchange of designs and the protection of the ownership rights. The current technology also offers new IPP mechanisms or techniques, being the watermarking one of the options that arises more interest [LAC01, KAH01].

This Thesis elaborates on new research lines for the development and protection of IP cores, proposing watermarking strategies for the protection of IP cores. This work also presents some application examples for the proposed IPP strategies, implemented over programmable logic and standard-cell library based ASICs, and the results are analyzed in terms of the main evaluation criteria. In addition, an exhaustive study of the security levels offered by the new strategies is carried out and a comparison with other protection alternatives existing in the literature is performed.

This memory is composed by seven chapters, as it is described at the following:

CHAPTER 1. Introduction

The first chapter shows the principal motivations for the development of this work. New trends in the development and implementation of digital systems are

displayed, with special emphasis in the reuse-based design and the implementations over programmable logic devices. The viability of the new methodologies based on the reuse of IP cores depends on the development of mechanisms for the intellectual property protection that, in case of being necessary, allow to claim the copyrights of the reusable modules. This chapter also presents watermarking as an important tool for the intellectual property protection of reusable modules.

CHAPTER 2. Development of digital systems using FPAGs.

In the second chapter, an introduction to logic programmable devices is performed, this devices having supported reconfigurable computation in the development of digital systems. In addition, this chapter presents the number representation systems, with special emphasis in the Residue Number System (RNS) [SZA67, TAY84]. Finally, the necessary tools for the development of digital systems implemented over FPGAs are analyzed and the most important steps in the digital system design flow through hardware description languages are detailed.

CHAPTER 3. Reuse-based desing and intellectual property protection for IP cores.

Chapter 3 focuses on the new design methodologies based on the reuse of IP cores. Despite the enormous advantages that these methodologies offer, they also present some disadvantages or limitations, one of the main being the intellectual property protection of those shared modules. This chapter also includes a study of the existing techniques for the intellectual property protection of reusable modules, being the watermarking one of the best solutions. Finally, a review of the principal watermarking techniques found in the literature for IP cores is carried out.

CHAPTER 4 . New watermarking technique for IPP of IP cores

This chapter makes a complete description of the new watermarking method proposed for IP cores. It also includes the preliminary ideas that led to the development of the first one of the watermarking strategies and the evolution that these ideas have experienced, thus resulting in the second watermarking strategy presented. Finally, this chapter presents the software tool developed for the automation and optimisation of the second watermarking strategy.

CHAPTER 5. Application and evaluation of the proposed watermarking strategies

Chapter 5 shows some examples of the application of the watermarking strategies detailed in Chapter 4. The obtained IPP designs are evaluated in terms of the impact on area, system performance and other evaluation criteria.

CHAPTER 6. Security analysis of the proposed techniques

Chapter 6 includes a study of the security of watermarking systems for digital content, focusing on the security of the new watermarking strategies developed for IP cores. It analyzes the invulnerability of the developed protection systems against the considered attacks and makes an evaluation and comparison of the proposed strategies with respect to other watermarking alternatives found in the literature.

CHAPTER 7. Conclusions

This chapter summarizes the most important ideas developed in this Thesis, as well as the future research lines that remain open after the finalization of this work.

At the end of the memory, three appendices with the following contents are included:

APPENDIX A. FPGA device families

FPGAs from Altera and Xilinx have been used to develop the research included in this Thesis, so this appendix describes the basic architecture and the most important aspects for the devices used in this research.

APPENDIX B. Residue Number System arithmetic

In this appendix, the RNS and its classic arithmetic modules addition and multiplication are described in detail. It also presents the principal proposals for the implementation of the binary-to-RNS and RNS-to-binary conversion schemes that involve the minor penalty in the area and the performance of RNS-based systems.

APPENDIX C. Design flow for ASIC semi-custom

This appendix describes the design flow for the synthesis over VLSI techmologies usign Synopsys Design Compiler tool and provides the basic information about the

lsi_10k standard cell library, used as target technlogy in the ASIC semi-custom examples.

To finish this prologue, some indications to facilitate the reading of this memory are provided:

- As it has been showed, the memory is divided in different chapters; these chapters consist of different sections, which have been enumerated with two digits, one relating to the chapter and other one to the section itself, and they will be referred in such way along this memory. Example: section 2.3
- When it is necessary, the sections will be divides into subsections, enumerated three digits, the first two digits with the same value as the section where the subsection is fitted, accompanied of another digit that will refer to the order of the subsection. Example: subsection 3.4.1.
- Figures and tables are enumarated by chapters, with the first digit indicating the chapter and the second one showing the number of figure or table in the corresponding chapter. Examples: figure 2.1, table 6.2.
- The mathematical expression and the equations are refered in a similar way, with two digits. Example: equation (4.1).
- Finally, the bibliographical references are referred in this memory with three letters, corresponding to the first author name, and two numbers, that indicate the publication year; in case that more than one reference have the same denomination, an additional letter will be used. The references are alphabetically ordered at the end of this memory. Example: [MAC01], [KAH98a], [KAH98b].

.

Capítulo 1: Introducción

En este capítulo se exponen las principales motivaciones para la realización del trabajo contenido en esta memoria. Se realiza una introducción al desarrollo de sistemas digitales, destacando la importancia que los dispositivos lógicos programables han alcanzado en los últimos años. Con estos dispositivos se consigue una mayor flexibilidad en el diseño de sistemas digitales y, al igual que ocurre con los circuitos ASICs, se pueden integrar sistemas digitales completos en un único dispositivo. También se fundamenta el uso de las nuevas metodologías de diseño basadas en la reutilización de módulos como una solución muy eficiente para reducir la complejidad generada por el alto nivel de integración y, además, se introduce uno de los problemas relacionados con estas nuevas metodologías, el de la protección de la propiedad intelectual. Por último, se detallan los principales objetivos que se persiguen con este trabajo.

1.1. Desarrollo de sistemas digitales

La metodología de diseño de los sistemas digitales ha estado condicionada por la tecnología disponible, y en especial por el nivel de desarrollo de los circuitos integrados. La tecnología actual permite un alto nivel de integración, proporcionando gran miniaturización, alta velocidad de funcionamiento y minimización en el consumo de potencia de los sistemas digitales implementados. Sim embargo, a medida que se producen todos estos avances, aumenta también la complejidad de los sistemas, surgiendo así la necesidad de un flujo de diseño que permita reducir los tiempos y costes de desarrollo. En este flujo de diseño juegan un papel muy imporante las

herramientas *software*, denominadas genéricamente herramientas CAD y la metodología de diseño utilizada. En esta sección se introducen diferentes propuestas para el desarrollo e implementación de sistemas digitales, discutiendo las ventajas e inconvenientes de cada una de ellas, mientras que en las secciones posteriores se presentarán los aspectos más importante de las nuevas metodologías de diseño de sistemas digitales.

1.1.1. Clasificación de los circuitos integrados

Actualmente se dispone de un abanico muy amplio de circuitos integrados para el diseño de sistemas digitales. La elección del tipo de disposivo depende fundamentalmente de la complejidad del sistema a diseñar y del número de unidades que se vayan a producir. Los circuitos integrados se pueden clasificar tal y como muestra la figura 1.1. Los circuitos integrados estándar están totalmente terminados por el fabricante, tienen una gran versatilidad para permitir la realización de cualquier

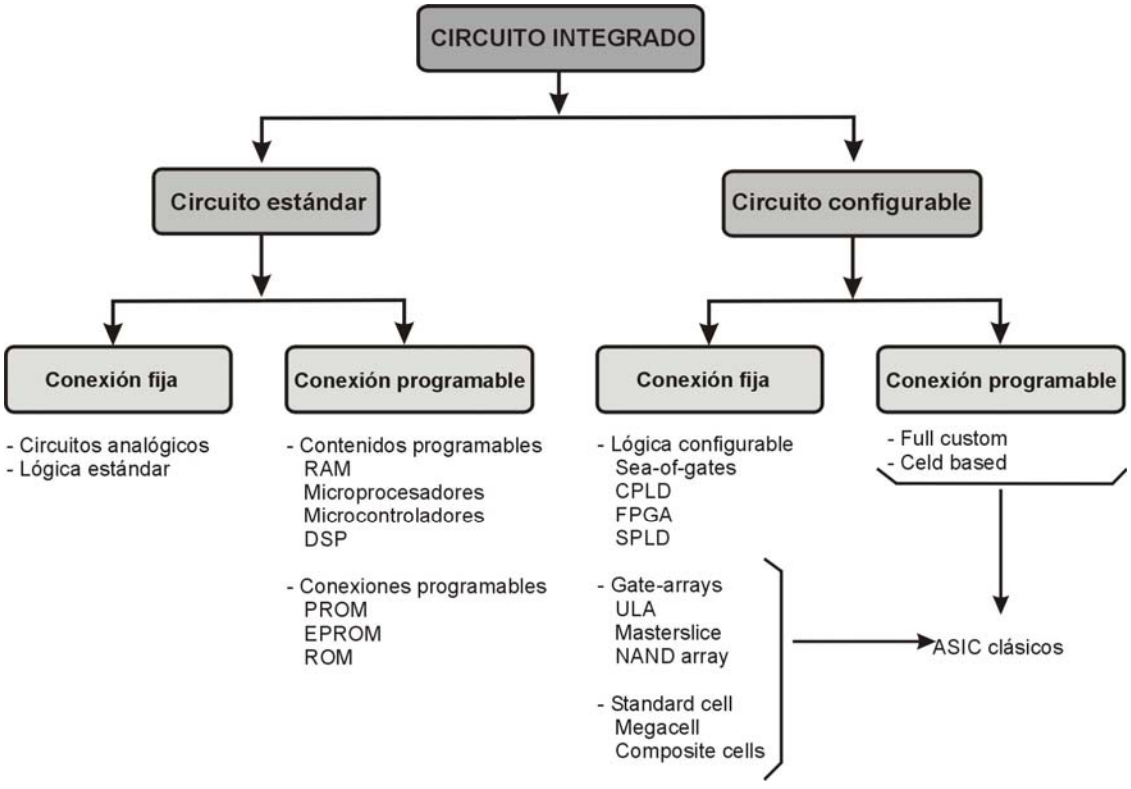

Figura 1.1. Clasificación de los CI.

sistema y son muy baratos. Sin embargo, hay numerosas aplicaciones que requieren muy altas prestaciones o funciones que los circuitos integrados estándar no pueden proporcionar. La utilización de los circuitos estándar se restringe a diseños que no sean execesivamente complejos y en los que el volúmen de producción no sea muy grande.

Debido a estas limitaciones surgen los circuitos a medida (*full-custom*) y los circuitos “casi a medida” (*semi-custom*), que son diseñados total o parcialmente por el diseñador y no por el fabricante del circuito integrado. Su utilización se justifca en el caso de necesitar grandes volúmnes de producción o si se requieren prestaciones que no se consiguen con circuitos estándar. El concepto ASIC se utiliza generalmente para referirse a cualquier tipo de dispositivo a medida o semi-a medida, pero una definición menos extensa reserva el término ASIC para todo circuito integrado VLSI diseñado mediante bibliotecas de celdas estándar o diseñado a nivel de transistor (*full-custom*). En los circuitos a medida el diseñador se encarga de realizar el trazado de las máscaras completo y remite al fabricante la descripción de las mismas. El diseño se puede realizar de forma completamente manual o utilizando objetos de una biblioteca de componentes. Aquí podría incluirse el diseño con celdas estándar (*standar cell*) en el que el constructor facilita un biblioteca *software* de celdas o módulos básicos, y el usuario los ensambla para configurar el circuito. La opción de circuitos semi-a-medida permite particularizar el circuito integrado para unas aplicaciones concretas, reduciendo considerablemente el tiempo de desarrollo de un nuevo circuito.

Dentro de los circuitos semi-a-medida se encuentran los dispositivos lógicos programables por el usuario (*Programmable Logic Devices*, PLD), en los que es el diseñador el que realiza la configuración final. Estos dispositivos están adquiriendo una gran importancia en el desarrollo de sistemas digitales debido a que consiguen una gran miniaturización, gran fiabilidad de funcionamiento y menores costes y tiempos de desarrollo del sistema. Existen distintos tipos de circuitos lógicos programables dependiendo de las funciones que puedan realizarse con los mismo y la forma en la que se realiza la programación. Los más importantes son los SPLDs (*Simple* PLD), los CPLDs (*Complex* PLD) y las FPGAs, que se verán con más detalle en el capítulo 2.

1.1.2. Tendencias en la implementación de sistemas digitales

Las alternativas para implementar un sistema digital, considerando su flexibilidad (programabilidad) y eficiencia (adaptación a la aplicación, en cuanto a velocidad consumo de potencia, etc.) se pueden agrupar en tres categorías:

- circuitos integrados estándar y ASIC;
- microprocesadores, microcontroladores;
- lógica reconfigurable.

Como se ha visto en la sección anterior, existen varias alternativas sobre el tipo de circuito integrado a utilizar para la implementación de un sistema digital [LLO03] y la elección de una de ellas dependerá, entre otros muchos, de factores tales como la velocidad, el consumo de potencia, el coste y el volumen de producción. El diseño con circuitos integrados estándar y ASIC resulta eficiente y rápido. Sin embargo, carece de flexibilidad ya que no pueden alterarse una vez fabricados. Por otro lado, la producción en serie lleva a unos costes por chip realmente bajos, además de que la inclusión en la fase de producción de tests garantiza la alta fiabilidad de estos circuitos. La tecnología actual permite crear circuitos integrados de gran complejidad y precio reducido, pero el coste de desarrollo de un ASIC sólo es justificable en el caso de grandes volúmenes de producción.

Los microprocesadores y microcontroladores son circuitos programables de arquitectura fija. Son menos eficientes que los circuitos a medida, pero se tiene la máxima flexibilidad ya que la función se cambia sin más que modificar el *software*. Aunque los primeros microprocesadores eran demasiado lentos para implementar en tiempo real la mayoría de los sistemas, a mediados de los ochenta la tecnología de los circuitos integrados había avanzado hasta permitir la realización de microcomputadores en punto fijo y punto flotante con arquitecturas especialmente diseñadas para realizar algoritmos de procesamiento de señales en tiempo discreto. A estos procesadores se les conoce por el acrónimo DSP (*Digital Signal Processor*). Algunas de sus características básicas como el formato aritmético, la velocidad, la organización de la memoria o la arquitectura interna hacen que sean adecuados para ciertas aplicaciones particulares, así como otras más genéricas.

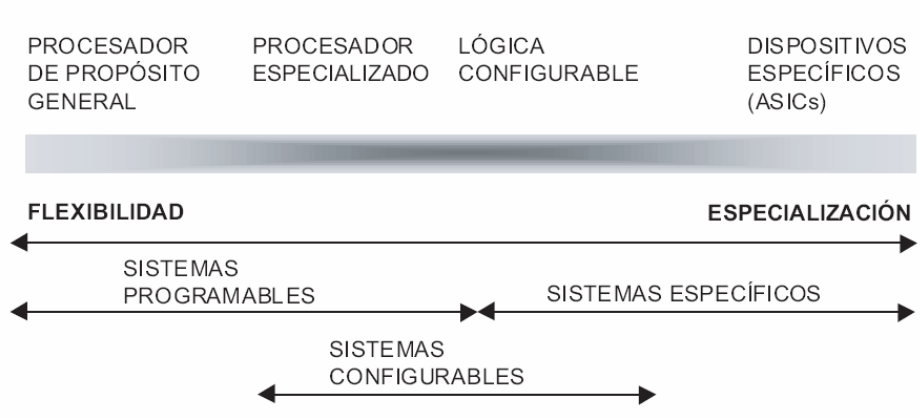

Figura 1.2. Flexibilidad y eficiencia en las distintas alternativas de diseño.

Los dispositivos de computación reconfigurable han alcanzado gran importancia en el desarrollo de sistemas digitales ya que cubren el espacio existente entre las dos opciones anteriores. En la figura 1.2 se muestra cómo los sistemas basados en dispositivos reconfigurables se sitúan en una zona intermedia en la relación flexibilidad-eficiencia (o flexibilidad-especialización) [LLO03]. Estos dispositivos son más eficientes que el uso de microprocesadores y microcontroladores, ya que el *hardware* se adapta más a la aplicación, y más flexibles que los circuitos integrados convencionales, ya que siempre es posible reconfigurar o reutilizar el sistema. En un principio, las características de estos dispositivos los habían erigido en una herramienta imprescindible para el prototipado de circuitos integrados, reduciendo enormemente los costes y el tiempo de desarrollo. Sin embargo, el aumento de prestaciones y la enorme densidad (con capacidades equivalentes superiores al millón de puertas lógicas por chip) alcanzados en los últimos años, han convertido a los dispositivos programables en una seria alternativa a los circuitos integrados digitales convencionales. Cuando se considera un número de unidades de producción reducido, la tecnología de dispositivos programables es mucho más adecuada que un ASIC en términos de tiempo y costes de desarrollo. Por otro lado, para aplicaciones que requieren una velocidad considerable, el diseño con microprocesadores o microcontroladores convencionales puede no resultar posible o más costoso que si se optase por dispositivos programables.

En lo que se refiere al procesamiento digital de señales, la implementación de cualquier algoritmo por medio de un DSP será la solución más eficiente cuando se alcance la velocidad requerida. Sin embargo, muchas de las aplicaciones tales como la

FFT (*Fast Fourier Transform*), el filtrado FIR (*Finite Impulsive Response*) o el filtrado IIR (*Infinite Impulsive Response*), que se implementaban de forma natural por medio de ASICs o DSPs, están siendo actualmente materializados de forma más eficiente sobre dispositivos programables [MEY01a]. Estos sistemas explotan técnicas de paralelismo e implementan múltiples unidades de multiplicación y acumulación con el objetivo de incrementar la velocidad de procesamiento y, por tanto, el ancho de banda del sistema.

En este trabajo se abordará el desarrollo de los sistemas digitales considerados mediante ASIC *semi-custom* y circuitos lógicos programables, en concreto mediante FPGAs. Como ya se ha comentado anteriormente, la gran complejidad de los sistemas que se diseñan actualmente hacen casi imprescindible la utilización de nuevas estrategias o metodologías de diseño. A continuación se tratan los aspectos más relevantes relacionados con esta nueva tendencia en el diseño de sistemas digitales complejos.

1.2. Metodologías de diseño actuales

La tecnología del silicio ha ido progresando hasta permitir chips con cientos de millones de transistores, lo que permite aumentar la complejidad de los sistemas y las aplicaciones. Esta tendencia ha permitido la integración de sistemas digitales completos en ASICs surgiendo el paradigma *System-on-Chip* (SoC). Un sistema SoC está compuesto por un conjunto de módulos y subsistemas (*hardware* y *software*) que se interconectan de forma apropiada para cumplir las funciones requeridas en una determinada aplicación. La complejidad del diseño SoC es mayor respecto a la del diseño convencional de sistemas distribuidos en circuitos impresos y en distintos circuitos integrados; sin embargo aporta numerosas mejoras a estos sistemas como son reducción del coste del producto, descenso significativo del consumo, reducción global de la superficie de silicio e incremento del rendimiento del sistema.

De igual forma que sucede con los ASICs, la capacidad de integración en las FPGAs se incrementa cada año de forma espectacular. En la actualidad se dispone de FPGAs con capacidades de integración de millones de puertas lógicas equivalentes.

Esta situación ha llevado recientemente al concepto *System-on-Programmable-chip* (SoPC), integrando en un mismo dispositivo un microprocesador, una matriz lógica del tipo de las FPGAs y las conexiones configurables correspondientes, lo que permite integrar en un solo chip todo un sistema digital basado en microprocesador. Se ha de mencionar que existen incluso alternativas que incluyen en el mismo dispositivo celdas analógicas configurables o programables (*Field Programmable Analogic Array*, FPAA) que actúan como interfaz al sistema digital anteriormente descrito. Las FPAAs no fueron aceptadas completamente hasta principios de los 90 [LEE91], [PIE98] y son el equivalente analógico de las FPGAs. Estos dispositivos brindan una solución eficaz a los problemas de rápido prototipaje y simplifican la tarea de diseñar circuitos electrónicos analógicos.

Frente a los SoCs basados en tecnología ASIC, los SoPCs tienen una desventaja en cuanto a precio y eficiencia puesto que no se trata de circuitos diseñados específicamente para una determinada aplicación. Sin embargo, en muchos casos compensan esta situación con su flexibilidad y accesibilidad para grupos de investigación y pequeñas y medianas empresas.

La necesidad de gestionar la complejidad generada por este nivel de integración en los SoCs y SoPCs ha incorporado nuevas metodologías basadas en el diseño a partir de módulos *hardware* y *software* (sistemas operativos en tiempo real, programas específicos para la aplicación, etc). El diseño de un SoC se basa en la reutilización de la propiedad intelectual en forma de bloques predefinidos y prediseñados, también conocidos como bloques IP, módulos IP, *cores* IP, macros, megaceldas o componentes virtuales [KEA98]. Con la disponibilidad de nuevas arquitecturas FPGA y sus herramientas de diseño, compatibles con metodologías de diseño ASIC, es posible emplear una metodología de diseño basada en la reutilización para ASICs y FPGAs. El diseño basado en la reutilización de componentes permite ensamblar sistemas complejos a través del uso de pequeños componentes, reduciendo así la complejidad del diseño de los sistemas. De esta forma, las estrategias basadas en módulos reutilizables permiten optimizar los recursos de las compañías debido a la reducción de los tiempos y costes de producción.

La cesión o venta de módulos IP se está convirtiendo en uno de los principales medios de transferencia de tecnología en la industria microelectrónica, especialmente en

el entorno de los dispositivos lógicos programables [ALT08], [XIL08]. Las nuevas tecnologías ofrecen nuevas aplicaciones y modelos, sin embargo, también se encuentra en ellas un claro objetivo para las apropiaciones indebidas. El uso de Internet, combinado con el gran desarrollo en la *World Wide Web*, ha hecho que la piratería sea mucho más fácil. El crecimiento en el negocio del mercado negro ofrece alternativas baratas y sorprendentemente confiables comparadas con las grandes marcas. El *hardware* disponible como propiedad intelectual está llegando a ser un objetivo muy lucrativo para la piratería, planteando así la necesidad de desarrollar mecanismos para su protección.

1.3. Protección de la Propiedad Intelectual

La propiedad intelectual abarca los derechos legalmente protegidos sobre los productos intelectuales, e incluye invenciones útiles, diseños novedosos, obras creativas, símbolos de origen, información comercial no pública y otros aspectos similares. Sin embargo, las ideas abstractas generalmente no están protegidas como propiedad intelectual. La propiedad intelectual comprende las patentes, los derechos de autor, las marcas registradas, la presentación comercial y los secretos comerciales. En los países industrializados y en un número creciente de países en desarrollo, los propietarios de derechos de la propiedad intelectual de distintos tipos gozan de limitados y definidos derechos exclusivos para fabricar, utilizar, vender, copiar, distribuir y/o registrar sus invenciones o creaciones. La propiedad intelectual ha surgido como un tema de plena actualidad para casi todas las empresas y una de las necesidades prioritarias que surgen es facilitar su protección.

La creciente disponibilidad y distribución de información multimedia en forma digital y la posibilidad de una fácil y barata reproducción de contenidos en industrias como las de cine, música o entretenimiento, hacen de la protección de los derechos de propiedad intelectual una cuestión cada vez mas crítica para la expansión del comercio en tales industrias. Sin embargo, la necesidad de protección de los derechos de la propiedad intelectual no es fruto de la evolución tecnológica, ya que se pueden encontrar diversos ejemplos desde la antigüedad. La inclusión de firmas manuscritas en

documentos, obras pictóricas, etc. responde a este propósito. La inclusión de una marca que permita identificar al propietario es tanto más vulnerable cuanta más información se posea sobre cómo ha sido hecha, dónde está ubicada, etc. En este sentido, es imprescindible que la marca pase lo más desapercibida posible.

Las técnicas utilizadas para la protección de la propiedad intelectual pueden buscar una funcionalidad más allá de facilitar al autor la prueba de que un determinado material es de su propiedad. Por ejemplo, resulta interesante añadir información en fragmentos musicales que indiquen título de la canción, autor, etc. de tal manera que un receptor capaz de recuperar dicha información la pueda presentar al usuario, manteniendo la transparencia para un receptor no preparado. Otra finalidad puede ser simplemente mantener la privacidad del contenido de un determinado mensaje. De esta forma, la ocultación de información podría ser una protección adicional a la criptografía. Por ejemplo, un texto criptografiado y escrito con tinta invisible presenta dos barreras a su desprotección.

En la protección de los derechos de propiedad se suelen confundir dos conceptos que corresponden a dos alternativas bien diferentes: protección anti-copia y protección del *copyright* o derechos de autor. La protección anti-copia se centra en la búsqueda de métodos que limiten el acceso al material registrado y/o inhiban el propio proceso de la copia. Algunos ejemplos de protección anti-copia son la encriptación de la transmisión de televisión digital, controles de acceso al *software* registrado mediante el uso de servidores de licencia y otros mecanismos de protección anti-copia en medios multimedia. La protección de los derechos de autor consiste en la inserción de información relativa a la autoría o propiedad en el objeto a proteger, sin una pérdida de calidad por parte de éste. Siempre que los derechos de propiedad de un objeto estén en cuestión, esta información puede ser extraída para identificar al propietario o autor legítimo. En este trabajo se hará referencia a la protección de los derechos de autor siempre que se hable de protección de la propiedad intelectual. El método más adecuado para introducir información en datos multimedia es el uso del *watermarking* digital o marcas de agua digitales. Mientras que la protección anti-copia puede ser difícil de llevar a cabo, los protocolos de protección de los derechos de propiedad basados en criptografía fuerte [LUC04] y *watermarking* parecen ser factibles para la protección de los derechos de autor.

1.4. *Watermarking* para módulos reutilizables

Las aplicaciones propuestas basadas en *watermarking* [BAR04, COX07, EGG02] son muchas y entre otras tareas incluyen la identificación del propietario de los derechos de autor. Las técnicas basadas en *watermarking* se han empleado tradicionalmente para identificar, de una forma segura, la autenticidad de texto, imágenes, vídeo y audio. Estas técnicas son una particularización de las técnicas de ocultamiento de datos con el requisito adicional de la robustez ante los posibles ataques. Consisten en la inserción de información del propietario y/o autor del módulo en el objeto digital sin pérdida de calidad del mismo. La información insertada puede ser extraída para identificar al dueño legítimo siempre que los derechos de propiedad de un objeto digital estén en cuestión.

Recientemente se han propuesto técnicas de *watermarking* para módulos IP que consiguen ocultar una firma digital o marca de agua (*watermark*) dentro del módulo. Las técnicas de *watermarking* utilizadas para bloques IP se diferencian de las utilizadas para datos multimedia. En el caso de bloques IP, los datos que representan al circuito no deberían ser modificados, pues se debe mantener el correcto funcionamiento del mismo. Algunos criterios para evaluar las distintas estrategias de *watermarking* son los siguientes: correcta funcionalidad, incremento de los recursos, transparencia, verificabilidad, dificultad para eliminar la marca y prueba de autoría.

En la literatura [HON99, KAH01, LAC01] se pueden encontrar técnicas de *watermarking* para la protección de módulos IP. Una posible clasificación de estas técnicas es la que diferencia entre técnicas aditivas y técnicas basadas en restricciones. Por ejemplo, en las técnicas aditivas, la firma se puede introducir usando tablas de consulta de la FPGA que no hayan sido utilizadas [LAC01]. En las técnicas basadas en restricciones, la firma se representa como un conjunto de restricciones adicionales. Un aspecto muy importante para el desarrollo las técnicas de *watermarking* en módulos IP es el tipo de modulo a proteger. En función de su flexibilidad para la implementación, se pueden diferenciar tres tipos de módulos IP: *soft cores*, *firm cores* y *hard cores*. Los *hard cores* están disponibles, por ejemplo, en forma de *layout*s parcialmente o totalmente enrutados, que no puede ser modificados por el usuario. Por otro lado, los *soft cores* necesitan ser procesados y pueden ser mapeados para distintos soportes. Por último, los *fir*m *cores* son empleados para representar diseños que están disponibles en

niveles de abstracción intermedios, tales como *netlists* o descripciones estructurales. Existen técnicas de *watermarking* para los distintos tipos de módulos IP. Dado que este trabajo ha centrado su interés en el desarrollo de nuevas técnicas de *watermarking* para la protección de módulos reutilizables, el capítulo 3 continuará con el estudio de las técnicas de *watermaking* para módulos IP además de tratar con más detalle los aspectos, características y particularidades vistas en esta sección.

1.5. Objetivos de esta tesis

El objetivo general de este trabajo consiste en la búsqueda y aplicación de técnicas para la protección de circuitos digitales que se utilizan como módulos reutilizables o IP *cores* para su implementación de sistemas digitales en FPGAs y ASICs. Estas técnicas pretenden permitir el poder reclamar los derechos de autor sobre estos módulos, implementando de esta manera una mecanismo eficiente para la protección de la propiedad intelectual.

En primer lugar se procederá en el desarrollo de mecanismos para la protección de la propiedad intelectual que aprovechen las particularidades del sistema numérico de residuos y los dispositivos lógicos programables. Se aprovecharán las ventajas que ofrece el RNS, no sólo en cuanto a velocidad, sino también para desarrollar módulos provistos de mecanismos para la comprobación de los derechos de autor y propiedad intelectual. De este modo, tras evaluar los mecanismos desarrollados en sistemas basados en el RNS e implementados sobre dispositivos programables, las ideas resultantes se aplicarán a sistemas digitales genéricos, tanto para tecnologías programables como para ASICs.

Las estructuras de protección resultantes habrán de ser indetectables, por lo que requieren un impacto mínimo sobre el área y prestaciones del sistema. Además, Otro aspecto importante en cualquier esquema de protección es su robustez contra ataques, por lo que se procederá a un análisis exhaustivo de la bondad de los mecanismos desarrollados en este sentido. Así, los principales objetivos propuestos se pueden resumir de la siguiente forma:

- desarrollo de técnicas de protección de la propiedad intelectual para sistemas digitales;
- desarrollo de módulos IP para FPGA y ASIC con los esquemas desarrollados para protección de la propiedad intelectual;
- análisis comparativo entre los módulos IP originales y los que incluyen los esquemas de protección para evaluar, entre otros criterios, el impacto sobre el área y las prestaciones;
- estudio de las distintas formas de ataques contra los esquemas de protección propuestos y búsqueda de mejoras en los mismos con el objetivo de aumentar su resistencia frente a ataques.

1.6. Conclusión

En este primer capítulo se han introducido conceptos clave para el seguimiento del trabajo contenido en esta memoria. Se ha realizado una introducción a los dispositivos y técnicas actuales para el diseño de sistemas digitales complejos. Los dispositivos lógicos programables, tales como FPGAs, aparecen como una alternativa atractiva para la implementación de sistemas digitales. Al igual que ocurre con los ASICs, la capacidad de integración de las FPGAs se incrementa de forma constante año a año. Este nivel de integración ha permitido que en la actualidad se incluyan sistemas digitales completos tanto en ASICs como en un único dispositivo FPGA. Para gestionar la complejidad generada por este nivel de integración se incorporan nuevas metodologías de diseño basadas en la reutilización de módulos prediseñados y reutilizables. Estas nuevas metodologías reducen los tiempos y costes de desarrollo, consiguiendo de esta forma una optimización de recursos. Sin embargo, esta tendencia en el diseño de sistemas digitales presenta algunos riesgos, siendo uno de ellos el de la protección de la propiedad intelectual. Surge así la necesidad de implementar técnicas que permitan el intercambio seguro de diseños y su protección, siendo éste el principal objetivo de este trabajo de investigación.

Capítulo 2: Desarrollo de sistemas digitales con FPGAs

En el segundo capítulo se introducen los dispositivos lógicos programables, que han dado soporte a la computación reconfigurable en el desarrollo de sistemas digitales. Además, se mencionan distintos sistemas numéricos de representación, prestando especial atención al RNS, alternativa muy eficiente para la implementación de sistemas de procesamiento digital de señales. Por último, se analizan las herramientas necesarias para el desarrollo de sistemas digitales con FPGAs y se detallan los pasos más importantes en el desarrollo de un sistema digital mediante lenguajes de descripción *hardware*. Por tanto, en este capítulo se exponen las ideas fundamentales en el desarrollo de sistemas digitales con FPGAS con el objetivo de fijar la terminología empleada en esta memoria.

2.1. Introducción

Los dispositivos lógicos programables de nueva generación se encuentran en disposición de revolucionar el campo del procesamiento digital de señales de la misma forma en que lo hicieron los procesadores digitales de señal hace dos décadas. Estos dispositivos se están empezando a considerar como un sustituto natural de los ASICs en un número amplio de aplicaciones. Además, la lógica programable es una tecnología que ofrece muchas ventajas sobre el uso de microprocesadores de propósito general, puesto que aprovecha las nuevas oportunidades que introducen las matrices de puertas configurables.

Las limitaciones en las prestaciones de los dispositivos programables, cuando se comparan con los ASICs, vienen impuestas por su arquitectura. Uno de los inconvenientes de esta arquitectura está relacionado con la velocidad de procesamiento. La mayoría de las aplicaciones de procesamiento digital de señales que están emergiendo durante los últimos años necesitan elevadas velocidades de procesamiento. Aunque la necesidad de obtener mayores velocidades de procesamiento aún se mantiene, los dispositivos alcanzan unos límites de tamaño y velocidad que se encuentran marcados por la tecnología. Por esta razón surge la necesidad de optimización de las arquitecturas en el ámbito aritmético y algorítmico. En las últimas décadas se han considerado sistemas numéricos y algoritmos alternativos para el desarrollo de sistemas de procesamiento de señales de elevadas prestaciones. Uno de los sistemas numéricos que mayor relevancia ha demostrado es el sistema numérico de residuos, o RNS.

2.2. Dispositivos lógicos programables

Los dispositivos de lógica programable han alcanzado gran importancia en el diseño de sistemas digitales. Las características de estos dispositivos pueden ser modificadas y almacenadas mediante programación, lo que los diferencia de los circuitos lógicos tradicionales, en los que la función que realiza el circuito viene predefinida de fábrica. El modo en que se implementa la programación depende de la arquitectura del dispositivo. Un PLD concreto tiene una superficie determinada, y en ella se distribuyen armoniosamente celdas lógicas, que tradicionalmente se componen de planos AND y OR, celdas de entrada/salida que añaden posibilidades a las celdas lógicas, y zonas de interconexiones, donde se encuentran los elementos que controlan la configuración del dispositivo. Con estos recursos se implementan las funciones lógicas deseadas mediante un *software* especial y un programador de dispositivos. La posibilidad de definir *a posteriori* la función a implementar da al usuario mucha mayor flexibilidad para construir diseños más complejos con mayor integración y a menor costo.

2.2.1. Evolución de los dispositivos programables

Desde los inicios de los sistemas digitales basados en tecnologías CMOS y TTL, la construcción de circuitos digitales programables ha sido una ambición de los ingenieros. Por esta razón, estos dispositivos han seguido una evolución tecnológica en los últimos 30 años. Una primera forma de sistema lógico reconfigurable son las memorias de acceso aleatorio o RAMs, que pueden ser empleadas para definir funciones. A pesar de ser muy prácticas, sus capacidades son limitadas: mucha de la información que almacenan probablemente sea redundante, están limitadas a definir un mapa fijo y su arquitectura hace que la velocidad de lectura sea notablemente más lenta que la de un circuito digital especializado.

A principios de los setenta empezaron a producirse las primeras versiones de circuitos lógicos programables. Constituidos por una matriz de puertas AND y OR interconectables, permiten definir funciones lógicas simples, con un bajo coste de puertas y alcanzando mayor velocidad. A esta generación corresponden las conocidas PAL (*Programmable AND-Array Logic*) y GAL (*Generic Array Logic*). Estos dispositivos lógicos se conocen como SPLDs. Aunque los SPLDs son muy útiles y versátiles, su tecnología estaba limitada a definir funciones simples y los recursos de interconexión se encuentran localizados en áreas muy concretas del chip.

A mediados de los ochenta surgieron los CPLDs, dispositivos lógicos configurables complejos que permitían resolver los problemas que presentaban los SPLDs. Básicamente son versiones más grandes y sofisticadas que los SPLDs en cuanto a las funciones que realizan los bloques lógicos, los bloques de E/S o las posibilidades de interconexión. Por lo general están formados por una colección de bloques tipo PAL, conectados por una serie de líneas centralizadas y programables. Aunque emplean una mayor cantidad de transistores, estos dispositivos abrieron una nueva gama de aplicaciones al permitir definir sistemas lógicos mucho más complejos.

La última etapa del desarrollo corresponde a las FPGAs, que han revolucionado el mundo del diseño digital, por su versatilidad, capacidad y facilidad de programación. Debido a su importancia, a continuación se describen sus aspectos más importantes.

2.2.2. FPGAs

Una FPGA difiere de la filosofía de los sistemas lógicos reconfigurables y dispositivos programables que le preceden (ROM, PLA, PAL, etc.). En lugar de dos niveles AND-OR programables, la FPGA incluye una matriz de elementos idénticos o bloques lógicos (*Logic Blocks*, LBs) unidos por una red de interconexiones. Gran parte del chip está dedicado a las interconexiones entre los diferentes conjuntos de elementos lógicos, que también son programables, junto a los bloques necesarios para proporcionar entradas/salidas disponibles para el usuario. La tarea del programador es definir la función lógica que realizará cada uno de los LEs, seleccionar el modo de trabajo de cada bloque de entrada y salida e interconectarlos. El diseñador cuenta con la ayuda de entornos de desarrollo especializados en el diseño de sistemas a implementarse en un FPGA. Un diseño puede definirse como esquemático, o haciendo uso de un lenguaje HDL o *Hardware Description Language* (lenguajes de descripción de *hardware*). Los HDLs más utilizados son VHDL, Verilog y ABEL .

Las FPGAs ofrecen facilidad de programación y configuración (ROM, JTAG, etc.), posibilidad de reconfiguración dinámica y gran capacidad, ya que existen dispositivos que equivalen a varios millones de puertas lógicas. Además, permiten la creación de prototipos de sistemas complejos sin necesidad de la fabricación de prototipos ASIC, reduciendo los costes y la duración del desarrollo. Para aplicaciones específicas, permiten la implementación de sistemas complejos evitando los costes de desarrollo de un ASIC. La flexibilidad de su arquitectura, la gran cantidad de elementos básicos, la capacidad de interconexión y las nuevas tecnologías de fabricación de circuitos, con un número mayor de puertas, hacen que una FPGA sea capaz de implementar prácticamente cualquier función deseada y que pueda llegar a integrar prácticamente cualquier sistema digital. De esta forma, cualquier circuito de aplicación específica puede ser implementado en un FPGA, siempre y cuando ésta disponga de los recursos necesarios. Las aplicaciones donde más comúnmente se utilizan los FPGA incluyen a los DSP, radio *software*, sistemas aeroespaciales y de defensa, prototipos de ASICs, sistemas de imágenes para medicina, sistemas de visión para computadoras, reconocimiento de voz, bioinformática, o emulación de *hardware*, entre otras. Cabe notar que su uso en otras áreas es cada vez mayor, sobre todo en aquellas aplicaciones que requieren un alto grado de paralelismo. Evidentemente, existen restricciones de

capacidad y velocidad que hacen las FPGAs poco efectivas para ciertas aplicaciones; sin embargo, los avances tecnológicos, la flexibilidad y bajo costo convierten a las FPGAs en alternativas económicamente viables.

El mercado de las FPGAs de propósito general cuenta con dos productores que están a la cabeza del mismo, Xilinx y Altera, y un conjunto de otros fabricantes que ofrecen dispositivos de prestaciones singulares, entre los que destacan Lattice Semiconductor, Actel, QuickLogic, Atmel, Achronix Semiconductor, ORCA y MathStar, Inc. En este trabajo de investigación se han empleado familias de dispositivos de Xilinx y Altera, a los que se les dedica el Apéndice A.

2.3. Técnicas aritméticas para FPGAs

Como se vio en el primer capítulo, los avances tecnológicos hacen posible implementar algoritmos cada vez más complejos. La realización eficiente de las diferentes operaciones aritméticas es clave para obtener aplicaciones reales. Por tanto, es necesario encontrar la representación y las estructuras más adecuadas para cada aplicación. La aritmética de computadores se ocupa de la implementación *hardware* de funciones aritméticas para diversas arquitecturas, así como de la mejora de algoritmos aritméticos orientados a *firmware* o *software*. De esta forma, en la aritmética de computadores hay dos aspectos de gran importancia a considerar: el sistema de representación y las operaciones algebraicas. El mayor reto es el diseño de circuitos y algoritmos para agilizar las diferentes operaciones. Esto es muy importante en campos como el procesamiento digital de señales. En esta sección se darán distintas soluciones para conseguir que la implementación de funciones aritméticas sobre FPGAs sea lo más eficiente posible.

2.3.1. Sistemas de representación

El primer problema asociado a la aritmética es el de la representación. Es necesario trasladar cantidades reales, con un sistema de representación infinito, a un sistema de representación finito y de precisión limitada. La mayoría de problemas

asociados a la aritmética de computadores son consecuencia de esta finitud, como el *overflow* y el escalado. Los diferentes tipos de datos pueden diferir en el número de bits empleados, pero lo que es más importante, en cómo el número representado es almacenado. De esta forma, se pueden encontrar dos posibilidades diferentes de representación: punto fijo (*fixed-point*) y punto flotante (*floating-point*) (figura 2.1). Decidir cuál de las dos representaciones es más apropiada es una tarea que no debe realizarse a la ligera. El tipo de aplicación y condicionantes relacionados con costo y velocidad son los que determinan la elección.

La representación de números enteros convencional está basada en sistemas de numeración con pesos posicionales. Estos sistemas (como el arábigo, de base 10) supusieron un gran avance sobre sus predecesores (como el romano). En muchas aplicaciones es necesario representar números con signo. Algunas técnicas usuales son la representación en signo y magnitud, la representación desplazada (*biased*), la representación en complementos, los dígitos y posiciones con signo, etc.. Cada una de estas alternativas implica el uso de algoritmos y circuitos diferentes. La representación en punto fijo aporta generalmente una mayor velocidad y bajo costo, pero rango dinámico y precisión limitados. Con la representación en punto flotante [MEY01a] se

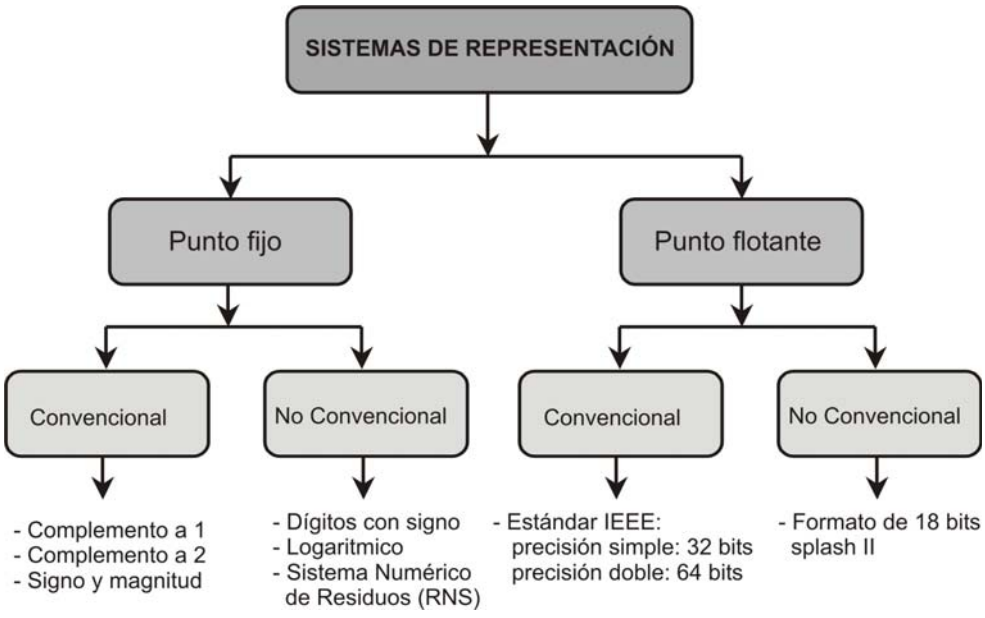

Figura 2.1. Clasificación de los sistemas de representación.

pueden representar cantidades reales y aparece una alternativa para proporcionar alta resolución sobre un gran rango dinámico, ya que éste es mucho mayor que el que se puede conseguir con punto fijo. Sin embargo, aparecen penalizaciones en cuanto a la velocidad y a la complejidad.

2.3.2. Optimización de funciones para FPGAs

Una de las desventajas tradicionales de las FPGAs radica en la implementación eficiente de funciones aritméticas. Como se acaba de ver, el sistema de representación elegido es un factor muy importante en la aritmética de computadores. En el caso concreto de funciones aritméticas para FPGAs, diversas técnicas, como la aritmética distribuida, o la aritmética de dígitos en serie, han tratado de implementar dichas funciones de una forma más eficiente. Por otro lado, la aparición de familias de FPGAs con cadenas de acarreo supone un gran avance en este aspecto. En concreto, la presencia de cadenas de propagación de acarreo entre elementos lógicos (*Logic Elements,* LEs) (para las familias de Altera) o bloques lógico configurables (*Configurable Logic Blocks*, CLBs) adyacentes (para las de Xilinx) permite la optimización de funciones aritméticas de suma. Puesto que prácticamente todas las operaciones aritméticas incluyen como bloque básico la suma, dichos recursos permiten la aceleración de un gran número de aplicaciones. Entre las familias con soporte para acarreo se encuentran la FLEX10K y APEX de Altera, y las diferentes Virtex de Xilinx, que se detallan en el Apéndice A.

En la familia FLEX10K de Altera, la cadena de acarreo permite la propagación de acarreos entre LEs adyacentes en tiempos inferiores a 0.5 ns, para dispositivos de 4 ns de tiempo característico. De este modo, es posible obtener el acarreo de salida de una suma de ocho bits en 2 ns aproximadamente. También es posible extender una cadena de acarreo entre varios bloques de memoria y de LEs que se agrupan formando una matriz de elementos lógicos (*Logic Array Blocks,* LABs). Para la familia Virtex, los tiempos característicos de dispositivos de grado – 4 para la generación de funciones de 4, 5 y 6 entradas son 0.8 ns, 1.2 ns y 1.6 ns, respectivamente. Para los mismos dispositivos, el retardo de propagación entre CIN y COUT es 0.2 ns, y 0.6 ns para la propagación de CIN a las salidas. Cada CLB incluye una puerta EXOR como soporte aritmético adicional. La presencia de estos recursos aritméticos permite optimizar la síntesis de la mayoría de funciones aritméticas, dado que la suma es su componente

esencial. Es posible extender estas cadenas más allá de bloques locales de LEs o CLBs. Sin embargo, el uso intensivo de este tipo de recursos puede producir problemas de enrutamiento si se emplean cadenas de longitud excesiva.

2.3.3. Técnicas especiales

Existen diferentes aplicaciones y tecnologías que pueden beneficiarse del uso de técnicas aritméticas especiales, particularmente en el campo del procesamiento de señales. Algunas de estas técnicas son el algoritmo CORDIC, la aritmética distribuida, la aritmética de dígitos en serie y la aritmética de cuerpos finitos. El algoritmo CORDIC (*COordinate Rotation DIgital Computer*) [VOL59, AND98] es un método de convergencia para el cálculo de funciones trigonométricas. Con este método es posible implementar diversas operaciones con números complejos con *hardware* y retardos equiparables a la división o raíz cuadrada, por lo que es de gran utilidad en procesamiento de señales y muy adecuado para implementaciones VLSI. La aritmética distribuida (*distributed arithmetic*) [WHI89] es una técnica muy eficiente para el cálculo de productos internos: filtros digitales, transformadas discretas, etc. Su nombre se debe a que las operaciones de suma y multiplicación quedan enmascaradas (distribuidas) en una realización *bit-serial*. Es de gran utilidad en el procesamiento de señales y muy adecuada para VLSI y FPGAs. La aritmética de dígitos en serie (*digit-serial arithmetic*) [ERC03] es una extensión de las arquitecturas *bit-serial*. Las estructuras *digit-serial* tienen multitud de aplicaciones, aunque se centran en el campo del procesamiento de señales usando como operadores básicos retardos de dígito y de muestra, operadores de desplazamiento y sumadores y multiplicadores.

El sistema numérico de residuos [SZA67, SOD86, TAY84] se basa en las propiedades aritméticas de anillos y cuerpos finitos, especialmente en propagación limitada de acarreos. Es una alternativa muy eficiente para la realización de aplicaciones de procesamiento digital de señales, especialmente aplicaciones de alta velocidad basadas en sumas de productos. Esto se debe a las propiedades intrínsecas del RNS, que incluye de manera natural el diseño modular y paralelo. Dado que parte de los diseños realizados en este trabajo están basados en aritmética de residuos, a continuación se tratará con más profundidad este sistema de representación.

2.3.3.1. El Sistema Numérico de Residuos

El origen del sistema numéricos de residuos se remonta al siglo III, en la antigua China [TAY84]. Una de las reglas básicas de la aritmética de residuos es el Teorema del Resto Chino, o CRT (*Chinese Remanider Theorem*) [SOD86], con el que se tiene un medio de realizar aritmética libre de acarreo. El resultado más tangible sobre el RNS fue una de las publicaciones clásicas en el campo del Sistema Numérico de Residuos [SZA67], en la que se estudiaron las principales ventajas e inconvenientes del RNS para el desarrollo de procesadores digitales avanzados. La principal conclusión de este trabajo fue que el RNS resulta únicamente eficiente para el desarrollo de sistemas específicos y que hagan uso sólo de operaciones en las que proporciona amplias ventajas como la suma, resta y multiplicación, y no de operaciones como la división, comparación o detección de signo, en las que el RNS plantea dificultad. Sin embargo, en la práctica, las investigaciones de Szabo y Tanaka encontraron poco éxito, debido a la cantidad de memoria que requerían los diseños, con lo que la tecnología del momento no podía satisfacer las demandas que exigía el RNS. Desde mediados de los años 70, tecnología y teoría han ido convergiendo y ahora es fácil implementar la memoria requerida por circuitos que usen el RNS. El RNS está encontrando su lugar en muchas aplicaciones en las que intervienen filtros digitales y transformadas, tales como realizaciones de la FFT a alta velocidad [JUL88], sistemas de corrección de errores [ORT92, KAT96], filtrado [IBR94, GAR98a], etc.

El RNS queda definido en términos de un conjunto de enteros relativamente primos entre sí, $\{m_1, m_{2,} ..., m_L\}$, llamados módulos. De esta forma, cualquier entero $X \in Z_M = \{0,1, ..., M-1\}$, donde:

$$M = m_1 \times m_2 \times ... \, m_L = \prod_{i=1}^{L} m_i \tag{2.1}$$

queda unívocamente representado en el RNS de la siguiente forma:

$$X \xrightarrow{RNS} (x_1, x_2, ..., x_L) \tag{2.2}$$

donde:

$$x_i = X \bmod m_i, \qquad i = 1,2,...,L \tag{2.3}$$

es el *i*-ésimo residuo de *X*.

Para un sistema de representación con signo sólo es necesaria una traslación del rango representado por el RNS, es decir, el rango $[-M/2, M/2]$ se hace corresponder con el rango $[0, M-1]$. Así los enteros $0 \le X < M/2$ representan su propio valor, mientras que los mayores que $M/2$ representan a los negativos. De esta forma cualquier entero X perteneciente al intervalo (-*M*/2, *M*/2), queda también representado de forma única mediante la L-tupla $(x_1, x_2, ..., x_l)$, donde cada residuo se obtiene de la siguiente forma:

$$x_i = \begin{cases} X \bmod m_i & si \quad X \ge 0 \\ (M - |X|) \bmod m_i & si \quad X < 0 \end{cases} \qquad i = 1,2,...,L. \tag{2.4}$$

2.3.3.2. Ventajas, limitaciones y aplicaciones

El RNS ha demostrado ser una buena alternativa tanto en área como en velocidad para la implementación de unidades aritméticas avanzadas. En los esquemas de representación y cálculo convencionales, la principal fuente de retardo en la operación de suma se debe a la propagación de acarreo entre las diferentes etapas, lo que intenta resolverse con circuitos especiales que suponen un incremento sustancial del *hardware* empleado. Además, en la multiplicación sucede lo mismo, ya que ésta se lleva a cabo con módulos sumadores. La principal propiedad del sistema numérico de residuos es que no existe propagación de acarreo entre canales y por tanto, permite llevar a cabo las operaciones de suma, resta y multiplicación de forma paralela, eliminando una de las principales fuentes de retardo de los circuitos aritméticos, la gestión de la información del acarreo, con lo que se puede operar a una velocidad superior a la obtenida en otros esquemas de representación.

En los esquemas de cómputo convencionales la propagación del acarreo disminuye la velocidad de funcionamiento a medida que aumentamos el rango. Esto hace que en estos esquemas, generalmente, precisión y velocidad sean parámetros antagónicos. En el RNS las operaciones se llevan a cabo de manera paralela sobre un

conjunto de módulos, cada uno de ellos menor que el rango dinámico del sistema total, con lo que la velocidad de operación vendrá limitada por la menor de las velocidades de operación de todos los módulos; de esta forma, en aritmética de residuos, es posible trabajar sobre grandes rangos sin penalizar la velocidad de operación, ya que el rango se amplía sin más que añadir nuevos módulos al RNS considerado. El RNS consigue superar así otra desventaja más que presenta el procesamiento digital en aritméticas convencionales.

Las principales objeciones al uso del RNS han sido las limitaciones impuestas por las etapas de conversión. Tradicionalmente, éstas han supuesto una penalización tanto en velocidad del sistema como en uso de recursos. Concretamente, los datos de entrada se han de convertir de cada formato de numeración pesado a su representación en el RNS y, tras su procesamiento, se han de volver a su formato original, como muestra la figura 2.2. Aunque este hecho ha limitado durante un cierto tiempo el uso del RNS a ciertas aplicaciones de elevada carga computacional y que demanden altas velocidades de procesamiento, existen esquemas eficientes de conversión que mitigan los inconvenientes que surgen en las etapas de conversión y hacen del uso de RNS una alternativa a los sistemas aritméticos tradicionales. Los esquemas de conversión más relevantes se muestran en el Apéndice B.

Algunos de los inconvenientes del RNS frente a las arquitecturas aritméticas tradicionales se deben a que el RNS es un sistema numérico de procesamiento de datos enteros y es no pesado. Como ya hemos visto, la dificultad para la comparación de

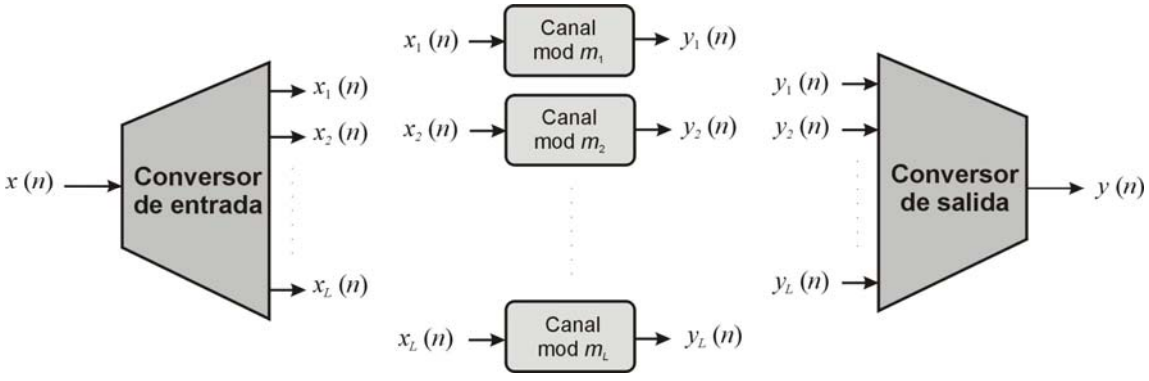

Figura 2.2. Estructura general de un sistema RNS.

magnitudes y detección de signo en el RNS hace difícil también la operación de división y la detección de desbordamiento; esto limita las aplicaciones del RNS y las restringe a las que requieren un elevado número de sumas y productos. En los sistemas pesados convencionales la pérdida de un dígito, o bit, alterará el significado decimal de la representación en función del peso correspondiente a este bit. Sin embargo, al no existir en el RNS dígitos más o menos significativos la pérdida de un bit puede modificar sustancialmente el significado decimal de la representación. Esto hace que se plantee otro problema relacionado con el uso del RNS, el de la corrección de errores. Sin embargo, añadir redundancia a los dígitos RNS es una tarea sencilla y en la literatura se pueden encontrar distintos métodos para la corrección de errores [MAN72, ETZ80].

A pesar de sus limitaciones, el RNS ha sido objeto de gran interés en los últimos años, ya que los sistemas digitales basados en el RNS pueden desempeñar un papel muy importante en sistemas de alta velocidad y tiempo real que soportan el procesamiento paralelo de datos. Todas las ventajas que ofrece este sistema de representación, desde el punto de vista aritmético, se suman a la propia naturaleza de los algoritmos considerados en DSP y hacen del RNS una herramienta muy útil en este campo.

2.3.3.3. Implementación de circuitos RNS sobre FPGAs

Hasta la aparición de las familias de FPGAs tales como APEX II o Stratix [ALT08] de Altera y las diferentes Virtex [XIL08] de Xilinx, los dispositivos programables han mostrado que las prestaciones aritméticas son su punto más débil frente a los ASICs [DIC00]. Estas limitaciones suponen un obstáculo para el uso de estos dispositivos si se tiene en cuenta la elevada velocidad de procesamiento que demandan las nuevas aplicaciones. Incluso para las familias más modernas, el incremento de prestaciones conlleva el sacrificio de parte del área del dispositivo para la inclusión de bloques aritméticos específicos; más aún, la reducción en velocidad se hace más evidente al incrementar la precisión, efecto propiciado por la propia estructura y conectividad en pequeños canales o bloques locales de los dispositivos programables y que ha sido llamado el efecto “barrera de canal” [RAM02b]. Este hecho refleja que las limitaciones en las prestaciones no sólo se debían a la capacidad limitada de las primeras familias sino que éstas también vienen impuestas por la propia arquitectura de

este tipo de dispositivos. Es aquí donde el RNS hace más evidentes sus ventajas frente a los sistemas aritméticos convencionales, como se verá a continuación.

Los dispositivos programables se componen de una matriz reconfigurable de elementos de memoria. Los dispositivos de las familias de Altera se encuentran organizados formando canales de elementos lógicos, normalmente de 8 bits. Dentro de cada uno de estos canales se pueden encontrar rutas de bajo retardo de propagación, incluyendo rápidas cadenas de propagación de acarreo y conexión en cascada, así como bloques de memoria que se pueden utilizar para implementar un gran número de funciones. Esta arquitectura presenta el siguiente inconveniente: cuando los bits de acarreo se tienen que propagar de un canal a otro a través de los canales de conexión del dispositivo, la velocidad de procesamiento se degrada rápidamente. Así todos los sistemas de procesamiento digital de señales en aritmética en complemento a dos se ven perjudicados por esta limitación en el dispositivo que se agrava con la creciente precisión de las nuevas aplicaciones. El RNS consigue solucionar estas limitaciones puesto que su aritmética se reduce a un conjunto de operaciones concurrentes que pueden realizarse en estos canales de reducida longitud de palabra [MEY99, MEY01, GAR99b, RAM01]. Aunque las ventajas del RNS son independientes de la tecnología VLSI empleada, todo esto convierte al RNS en una aritmética muy atractiva para implementar sistemas de procesamiento digital de señales de elevadas prestaciones en dispositivos programables [RAM02a, HAM95, SAF97, MEY01].

2.4. Desarrollo de sistemas digitales con FPGAs

En la actualidad, el desarrollo de la mayor parte de sistemas digitales es de gran dificultad, no sólo por la cantidad de elementos a interconectar, sino además por la variada complejidad de los mismos. Las aproximaciones tradicionales son inabordables manualmente y la descripción del sistema requiere una estructura jerárquica. Por tanto, el proceso de diseño y construcción de sistemas digitales complejos requiere el uso de herramientas automáticas de síntesis. Estas herramientas permiten, con un coste menor y con un grado de confianza mayor, una exploración rápida de las diferentes alternativas de implementación de un sistema digital integrado. Si a esto se le añade la posibilidad

de realizar prototipos sobre FPGAs (soportado también por las herramientas CAD), es posible disponer en muy poco tiempo de un abanico de circuitos *hardware* con diferentes prestaciones.

2.4.1. Herramientas CAD

Para el desarrollo de sistemas digitales existen herramientas *software* que facilitan su diseño y construcción. Los programas orientados a un diseño de alto nivel se suelen denominar CAE (*Computer-Aided Engineering*). Las herramientas orientadas al diseño físico se denominan programas para diseño con ayuda de computador o CAD. El término CAD suele abarcar también al concepto CAE. Estas herramientas han permitido que el diseñador desplace su actividad hacia niveles de mayor abstracción que requieren de su parte menor grado de detalle. Las herramientas CAD intervienen en todas las fases del diseño siendo sus cometidos los siguientes: describir el sistema sin ambigüedad, simular en computador el comportamiento del sistema, ayudar a la realización del diseño físico, analizar sistemáticamente el diseño para detectar posibles errores, sintetizar de forma automática una parte o la totalidad del sistema y facilitar la forma más eficiente para el test del sistema una vez construido. Hay herramientas disponibles para abordar uno o varios de esos cometidos. En concreto, para la descripción del sistema existen diferentes formas de descripción de alto nivel de sistemas digitales: descripciones algorítmicas, descripción RTL (*Register Transfer Language*) y lenguajes de descripción de *hardware* o HDL. Los lenguajes HDL son lenguajes de descripción formal, pensados para la descripción, documentación y diseño de elementos *hardware*.

2.4.2. Lenguajes de descripción *hardware*

En los años ochenta se impusieron dos de estos lenguajes sobre los demás: Verilog y VHDL [PAR99, TOR01]. En esta memoria se ha utilizado VHDL para el desarrollo de los sistemas digitales correspondientes. VHDL (*Very High Speed Integrated Circuit Hardware Description Language*) es un lenguaje de descripción *hardware* utilizado para describir circuitos en un nivel alto de abstracción, que está siendo rápidamente aceptado como un medio estándar de diseño y representa una herramienta de gran valor para la síntesis de sistemas por medio de PLDs o ASICs.

VHDL es un producto del programa *Very High Speed Integrated Circuit* (VHSIC) desarrollado por el Departamento de Defensa de los Estados Unidos a finales de la década de los 70. El propósito era obtener un estándar para diseñar, modelar y documentar circuitos complejos de tal manera que un diseño desarrollado por una empresa pudiera ser entendido por otra y, además, pudiera ser procesado por *software* con propósitos de simulación. Fue reconocido como un estándar de los lenguajes HDL por el IEEE en 1987 (IEEE-1076). En 1993 el estándar se actualizó y se adoptó un estándar adicional, el IEEE-1164. Posteriormente, en 1996, el estándar IEEE-1076.3 se convirtió en un estándar de VHDL para síntesis, siendo éste y el IEEE-1164 los estándares más utilizados en síntesis de circuitos digitales por la mayoría de las herramientas de diseño.

VHDL permite diferentes tipos de descripción:

- comportamiento (*behavioral*): se describe la forma en que se comporta el circuito. Esta es la forma que más se parece a los lenguajes de *software*, ya que la descripción es secuencial. Estas sentencias secuenciales se encuentran dentro de los llamados procesos en VHDL. Los procesos son ejecutados en paralelo con asignaciones concurrentes de señales y con las instancias a otros componentes. Es una descripción abstracta de alto nivel;
- flujo de datos (RTL): flujo de datos entre registros y bloques funcionales;
- estructural: se describe el circuito con instancias de componentes. Estas instancias forman un diseño de jerarquía superior, al conectar los puertos de estas instancias con las señales internas del circuito, o con puertos del circuito de jerarquía superior;
- Mixta: combinación de todas o algunas de las anteriores.

Además, VHDL permite tanto la síntesis como la simulación del sistema descrito y la creación de *testbench* (bancos de prueba) para simulación. Entre las ventajas que proporciona la utilización de este lenguaje se pueden destacar las siguientes:

- permite especificar un diseño con alto nivel de abstracción para así poder hacer uso de herramientas eficientes de síntesis que saquen el mayor partido posible de la tecnología empleada y optimizan el sistema: ahorro en área, minimización del retardo o reducción del consumo de potencia;

- permite portar las aplicaciones desarrolladas a otras tecnologías de forma sencilla y desarrollar entidades IP de posible interés comercial, todo esto debido a que es un lenguaje estandarizado y de alta aceptación, tanto por la comunidad científica como por la industria.

VHDL se ha convertido, junto a Verilog, en un estándar de facto para la industria. Por todo, esto VHDL se utiliza actualmente para la síntesis de circuitos digitales utilizando dispositivos lógicos programables [CHA97]. Es así como los dispositivos lógicos programables y VHDL constituyen los elementos fundamentales para las nuevas metodologías de diseño y, en concreto, para el diseño, desarrollo e implementación de los circuitos que se mostrarán en este trabajo.

2.4.3. Flujo de diseño con HDL

Las etapas en el desarrollo de un sistema digital sobre FPGAs llevan al diseñador desde los requerimientos sobre el circuito lógico a crear, hasta la etapa en que finalmente se reconfigura el dispositivo lógico para implementar la nueva funcionalidad. En la figura 2.3 se ilustran las etapas de este desarrollo.

La primera etapa es la de definición o descripción del diseño y consiste en definir en HDL el circuito a implementar. Esto puede hacerse también empleando módulos reutilizables, que permiten, por lo general, lograr rápidamente diseños más eficientes. Seguidamente el diseño debe ser verificado, existiendo diversas metodologías para realizar este proceso, entre las que se encuentra la verificación formal, que emplea reglas lógicas que el circuito debe cumplir. En caso de presentar dificultades se inicia un proceso de iteración en que se modifica el código HDL hasta lograr un diseño que cumpla con los requerimientos lógicos.

Después de describir el diseño se realiza la síntesis del mismo. En este paso se adapta el diseño a un *hardware* en concreto, en este caso una FPGA. Hay sentencias del lenguaje que no son sintetizables. El hecho de que no todas las expresiones en VHDL sean sintetizables se debe a que es un lenguaje genérico para modelado de sistemas, no sólo para diseño de circuitos digitales, por lo que hay expresiones que no pueden ser transformadas a circuitos digitales. El proceso de síntesis consiste en reducir el HDL a una descripción de más bajo nivel compuesta exclusivamente de componentes

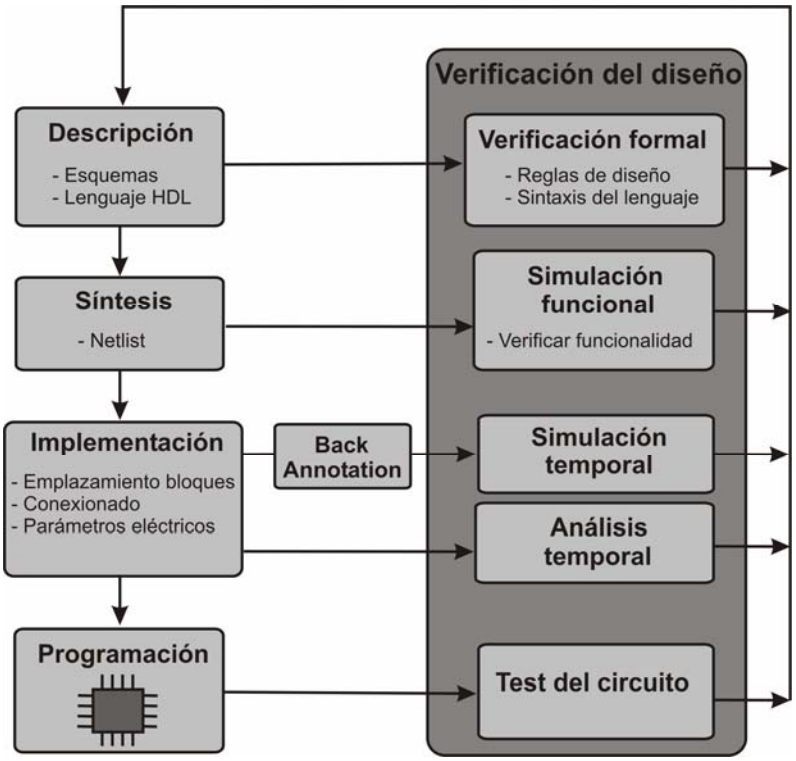

Figura 2.3. Etapas en el diseño de un sistema digital.

elementales y sus interconexiones. Esta descripción sin ningún nivel de abstracción se denomina *netlist*. Es en esta etapa en la que el sintetizador identifica elementos de alto nivel del HDL, si es que los hay, realiza reducciones lógicas y optimizaciones varias (uso de recursos, consumo de energía, etc.). El sintetizador optimiza las expresiones lógicas con objeto de que ocupen menor área, o bien son eliminadas las expresiones lógicas no usadas por el circuito. Además, se tiene en cuenta la estructura interna del dispositivo, y se definen restricciones, como la asignación de pines. En la simulación funcional se comprueba que el código HDL corresponde a las especificaciones del diseño. Además, se comprueba que el sintetizador ha realizado correctamente la síntesis del circuito, al transformar el código HDL en bloques lógicos conectados entre sí. Este paso es necesario ya que, a veces, los sintetizadores producen resultados de síntesis incorrectos, o bien realizan simplificaciones del circuito al optimizarlo.

La etapa de emplazamiento y enrutamiento (*Place&Route*) se encarga de realizar el vínculo final entre los elementos lógicos del circuito y los elementos físicos de la FPGA. El emplazamiento consiste en definir qué bloque lógico configurable

implementa cada función elemental, y situar los bloques digitales obtenidos en la síntesis de forma óptima, de manera que aquellos bloques que se encuentran muy interconectados se sitúen próximos. El proceso de enrutamiento consiste en rutar adecuadamente los bloques entre sí, intentando minimizar retardos de propagación para maximizar la frecuencia máxima de funcionamiento del dispositivo. Evidentemente la elección de los emplazamientos está restringida por la capacidad de enrutamiento. Los algoritmos de emplazamiento y enrutamiento deben resolver un complejo problema de optimización. En esta etapa debe lograrse cumplir los requerimientos de velocidad del diseño (las rutas más largas son más lentas), cumpliendo la limitaciones físicas del dispositivo. Una vez completado el emplazamiento y enrutamiento, mediante el *back-annotation* se extraen los retardos de los bloques y sus interconexiones, con objeto de poder realizar una simulación temporal (también llamada simulación *post-layout*). Estos retardos son anotados en un fichero SDF (*Standart Delay Format*) que asocia a cada bloque o interconexión un retardo. Debido a los retardos internos del chip, es posible que, aunque con la simulación se obtengan buenos resultados, cuando la FPGA se progame, el diseño no funcione correctamente. Con un análisis temporal se puede comprobar estos retardos, y si hay errores se tiene que volver a uno de los pasos anteriores.

Una vez terminado el proceso de emplazamiento y enrutamiento, cumpliendo las restricciones temporales del diseño, se realiza la programación o configuración sobre la FPGA. La programación queda registrada en un archivo de bits (*bitstream*), que es el que finalmente se cargará en la FPGA. Los métodos de programación en la FPGA dependen de su diseño particular, pero suelen ser estándares, permitiéndose cargas por protocolos serie, paralelo y por puerto JTAG (un estándar industrial para grabación y verificación de dispositivos).

Como en todo procedimiento de desarrollo, cada etapa puede dar lugar a una revisión del diseño original. En el caso de los dispositivos de lógica reconfigurable, además debe verificarse su buen funcionamiento en interacción con los demás dispositivos del circuito físico. Es sabido que la detección de errores en sistemas complejos es una tarea complicada. Debido a esto, es importante disponer de buenas herramientas de simulación y verificación del diseño en cada etapa del desarrollo. Uno de los inconvenientes para trabajar con los dispositivos de lógica reconfigurable es que

deben emplearse herramientas altamente especializadas y de alto coste. Esto se debe a que los dispositivos emplean arquitecturas no estandarizadas, en constante innovación, y que son dispositivos que se emplean en líneas de producción con grandes presupuestos.

Existen distintos proveedores de herramientas de diseño, simulación y síntesis. Las herramientas de emplazamiento y enrutamiento, por su naturaleza, suelen ser ofrecidas exclusivamente por los fabricantes de los propios dispositivos. Para tratar de romper la barrera de acceso a la tecnología, tanto Xilinx como Altera ofrecen herramientas gratuitas.

2.5. Conclusión

En este segundo capítulo se ha realizado una introducción a los dispositivos lógicos programables que han dado soporte a la computación reconfigurable en el desarrollo de sistemas digitales. A lo largo del mismo, se presenta la evolución sufrida desde las PLAs hasta las últimas familias de FPGAs y se describen las características más relevantes de las FPGAs. Se ha incluido una sección donde se presentan los distintos sistemas numéricos de representación y la importancia que éstos tienen en el desarrollo de sistemas digitales de elevadas prestaciones con dispositivos lógicos programables. Entre ellos se encuentra el RNS, que constituye una alternativa muy eficiente a los sistemas numéricos tradicionales para la implementación de sistemas digitales. Por último, se ha realizado un resumen de las herramientas necesarias para el desarrollo de sistemas digitales con FPGAs, entre las que se encuentran los lenguajes de descripción *hardware*, y se han detallado los pasos más importantes en el desarrollo de un sistema digital.

Capítulo 3: Diseño basado en la reutilización y protección de módulos IP

En este capítulo se da una perspectiva de las nuevas metodologías de diseño basadas en la reutilización de módulos que consiguen reducir la complejidad, los costes y los tiempos de desarrollo. Se diferencian los distintos tipos de módulos IP y se estudian los aspectos más importantes que han de caracterizar el flujo de diseño basado en la reutilización. Sin embargo, la viabilidad de estas metodologías dependen del desarrollo de técnicas apropiadas para la protección de la propiedad intelectual, entre las que se encuentra el *watermarking*, que actualmente es una de las técnicas más utilizadas para la protección de bloques IP. Se realiza un análisis de los principales objetivos a conseguir con la aplicación de técnicas de *watermarking*, que además ayudarán a establecer los criterios de evaluación de los diseños resultantes. Por último, se clasifican y revisan las principales técnicas de *watermarking* desarrolladas para módulos IP.

3.1. Introducción

El gran avance en la tecnología de fabricación de los chips lleva a circuitos con nuevos niveles de integración y mayor complejidad. Esto hace posible que en las nuevas aplicaciones se piense en la implementación de sistemas complejos que ocupen un tamaño pequeño. Sin embargo, la experiencia de los equipos de trabajo, la productividad y las herramientas de diseño no avanzan al mismo ritmo que la tecnología

de fabricación. El aumento en la complejidad de los sistemas hace que se necesiten ciclos de verificación y de diseño más largos, lo que hace que exista una gran diferencia entre lo que puede ser construido y lo que puede ser diseñado.

Con las metodologías de diseño convencionales, la barrera entre la capacidad de integración de los chips y la productividad de los diseños se iría haciendo mayor, lo que reduciría el crecimiento de la industria de los semiconductores. En este sentido, un gran número de diseñadores de ASICs ha empezado a cuestionarse estas metodologías de diseño y las herramientas empleadas en busca de otras que hagan más efectiva la gestión de la enorme cantidad de puertas que hay disponibles en estos momentos. Con estas metodologías convencionales los diseños de los procesos para los ASICs se llevaban a cabo por un único grupo de diseño trabajando en un solo circuito y todos sus módulos. No había ninguna reutilización, o muy poca, de los diseños anteriores. La lógica de control y los puertos de comunicación se diseñaban para cumplir con las necesidades de cada circuito. A medida que aumentan las capacidades de funcionalidad y de integración del sistema, este flujo de diseño se hace menos productivo y más caro, es decir, esta metodología de diseño es inadecuada para favorecer el aumento en la productividad que implicaría el incesante incremento en el número de puertas lógicas de los chips. Para solucionar estas barreras y limitaciones se necesita un importante cambio en el diseño y en el centro de este cambio se encuentra la reutilización.

3.2. Diseño basado en la reutilización

El diseño basado en la reutilización de módulos IP [KEA98, JAC01] ofrece una solución para saltar la barrera existente entre el número de puertas disponibles y la productividad de los diseños. El uso de módulos reutilizables permite ensamblar sistemas a través del uso de módulos o bloques funcionales previamente diseñados y verificados, reduciendo así la complejidad de los sistemas [D&R08]. Por otro lado, se reduce el ciclo del diseño, ya que se libera al diseñador del desarrollo y prueba de algunas partes del circuito, y se reducen costes, ya que se permite la reutilización de una misma porción del diseño en diferentes proyectos. Es decir, el diseño basado en la reutilización es capaz de reducir la complejidad, los costes y tiempos de desarrollo

permitiendo la producción de circuitos de elevada complejidad. El requisito fundamental para la reutilización es que la función que realiza un determinado bloque se utilice en un gran número de aplicaciones. Los circuitos aritméticos, tales como sumadores y multiplicadores, son ejemplos de bloques que se emplean en una gran variedad de aplicaciones. El número de compañías que se dedica a la producción y a la venta de módulos IP está en constante aumento. Sin la reutilización de módulos, la industria electrónica no sería capaz de afrontar el desafío de fabricar dispositivos cada vez mejores, más rápidos y más baratos. Por todo esto, las estrategias de diseño se están dirigiendo hacia el uso, cada vez más generalizado, de módulos o bloques IP.

3.2.1. Tipos de módulos IP

De modo general, el término propiedad intelectual hace referencia a productos del intelecto humano tales como ideas, invenciones, expresiones, métodos de negocio, fórmulas, información, datos, etc. La propiedad intelectual hace también referencia a bloques prediseñados conocidos como bloques IP, *cores* IP, macros IP o componentes virtuales. En este contexto, la propiedad intelectual puede ser descrita de varias formas y dar lugar así a distintos bloques IP [KEA98]. La figura 3.1 muestra los distintos tipos de componentes virtuales que se pueden encontrar en el flujo de diseño normal de una

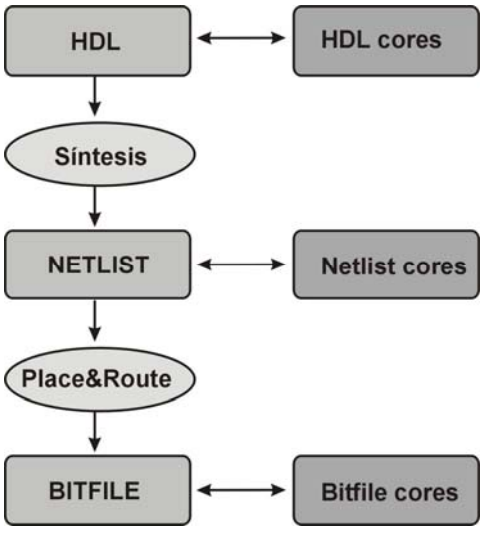

Figura 3.1. Tipos de módulos IP en el flujo de diseño de una FPGA.

FPGA o ASIC. Se diferencian tres tipos de módulos IP en función de su etapa en el diseño: *HDL–cores, netlist–cores* y *bitfile–cores*. Esta clasificación está relacionada con la flexibilidad para su síntesis, que hace que estos puedan aparecer también en la literatura con la siguiente nomenclatura: ***soft cores, firm cores*** y ***hard cores.***

Los *HDL–cores* o *soft–cores* están descritos en lenguajes de descripción *hardware* a nivel de transferencia entre registros y su código representa entidades *hardware* sintetizables. Este tipo de componentes virtuales son flexibles y relativamente independientes de la tecnología, pudiéndose sintetizar para distintos dispositivos, incluso ASICs o lógica programable [PRE95]. En los *netlist–cores* o *firm–cores* la descripción es a nivel de puertas y biestables (*netlist*), optimizada en tamaño y rendimiento para un dispositivo o familia de dispositivos en concreto. Por tanto, la asignación de recursos del dispositivo para realizar esas tareas ya está realizada antes de su utilización por parte del diseñador, restando únicamente la fase de interconexión que se realizará en la implementación. Son menos flexibles pero permiten la protección de la propiedad intelectual y, en general, son más eficientes. Por último, se encuentran los *bitfile–cores* o *hard–cores*. En el caso del diseño ASIC este tipo de bloques están preparados para ser integrados directamente en el silicio. Disponen de la información a bajo nivel de los recursos necesarios que hay que implementar en el silicio y de la información del enrutamiento exacto. Están optimizados en aspectos tales como el rendimiento, área o consumo. En el caso de los *hard–cores* para dispositivos lógicos programables, son bloques ya integrados en el silicio del dispositivo. Estos módulos IP son muy poco flexibles, ya que solamente pueden ser usados para un diseño específico. Los *soft–cores* son los que admiten mayor flexibilidad mientras que los *hard–cores* son los que resuelven las necesidades más complejas (procesadores avanzados, multiplicadores de alto rendimiento, etc.) y se encuentran mejor optimizados.

3.2.2. Metodología de diseño para la reutilización de módulos IP

Como se ha visto, la reutilización es la decisión natural para reducir los costes y la complejidad de un diseño, porque divide este último en varios proyectos. Sin embargo, la experiencia demuestra que importar fragmentos de otro circuito y añadirlos a uno nuevo no es el enfoque más adecuado: implica un importante trabajo de rediseño para

combinarlos en cada circuito. Un flujo de diseño más fiable y económico consiste en desarrollar los circuitos a partir de una biblioteca de módulos diseñados específicamente para su fácil integración y reutilización.

La efectividad del diseño basado en la reutilización requiere de una extensa variedad de módulos o bloques reutilizables. Es muy importante tener en cuenta que para que un diseño sea reutilizable debe ser usable. Es decir, el diseño debe ser robusto y correcto en su definición, lo que implica una documentación y uso de código correctos, comentarios minuciosos sobre el código RTL, entornos de verificación bien diseñados, uso de *scripts* robustos, etc. Los diseñadores de módulos reutilizables deben, además, emplear una metodología de diseño que haga que estos módulos se puedan reutilizar. Es conveniente tener claros algunos de los principios que debe cumplir la metodología de diseño basada en la reutilización:

- diseñar pensando en la reutilización teniendo en cuenta que el bloque será modificado y reutilizado en otros proyectos por otros equipos de diseñadores;
- empleo de herramientas y procesos que capturen la información del diseño de una forma consistente y fácil de comunicar;
- uso de herramientas y procesos que faciliten la integración de los módulos en otros diseños, incluso cuando el diseñador del módulo no supervise la integración.

El primero de estos principios indica que el diseño de cada módulo se debe realizar pensando en su reutilización. Algunos criterios o requisitos que se deben tener en cuenta para que un bloque sea completamente reutilizable son los siguientes:

- **diseñar el bloque para solucionar un problema genérico**: debe ser fácilmente configurable para su uso en diferentes aplicaciones;
- **diseñar el bloque para su materialización sobre varias tecnologías**: para los *cores soft* esto significa que los *scripts* de síntesis deben producir resultados satisfactorios con una variedad de bibliotecas; para los *cores hard* esto implica tener una estrategia efectiva para implementar el *core* en nuevas tecnologías;

- **diseñar el bloque para su simulación con una variedad de simuladores**: un *core* que funcione sólo para un único simulador no es portable; el bloque debe funcionar con los principales simuladores comerciales;
- **verificar el funcionamiento del bloque en el chip de forma independiente**: a menudo, los *cores* son diseñados y testeados sólo parcialmente antes de ser integrados en un chip para su verificación; de esta forma se ahorra el esfuerzo de desarrollar un banco de pruebas completo para un diseño;
- **verificar el bloque para asegurar un alto nivel de fiabilidad:** esto implica una verificación rigurosa además de la construcción de un prototipo físico;
- **documentar los bloques perfectamente en términos de aplicaciones apropiadas y restricciones**: se deben documentar las configuraciones válidas del diseño, los valores de los parámetros, las restricciones sobre las configuraciones de sistema o los valores de los parámetros y los requerimientos y restricciones de las interfaces.

Por otro lado, las herramientas de diseño tienen una influencia muy importante en la reutilización. Es muy conveniente elegir herramientas que ayuden al flujo de reutilización en lugar de imponer limitaciones al mismo. Algunos fabricantes de herramientas de diseño pueden ayudar a configurar un flujo de diseño reutilizable. Las nuevas arquitecturas FPGA disponen de herramientas de diseño compatibles con metodologías de diseño ASIC. Esto hace posible que se pueda emplear una metodología de diseño basada en la reutilización de diseños implementados en ASICs y FPGAs y su integración en plataformas SoC y SoCP.

3.2.3. Reutilización en SoCs e iniciativas de estandarización

Como se vio en el primer capítulo, los SoCs y los SoCPs son una de las mejores opciones para la integración de sistemas digitales completos de elevada complejidad en un solo chip. Debido a la exigencia de reducir los tiempos de desarrollo y a los requerimientos de elevada funcionalidad, los diseñadores están adoptando, cada vez más, el flujo de diseño SoC. En la actualidad, la reutilización de módulos IP es una de las estrategias más importantes en el diseño de SoCs y SoCPs.

Un SoC contiene un gran numero de módulos o componentes, que juegan un papel muy importante en el rápido desarrollo de un SoC y además permiten reducir costes. Una de las claves del éxito en un diseño SoC reside en el buen diseño de los módulos que lo componen. No importa cómo de bueno sea el flujo de integración de un SoC si los bloques que se están empleando no están bien diseñados. Por otro lado, módulos que estén bien diseñados pueden ser integrados en cualquier flujo SoC y el resultado final será muy satisfactorio. La figura 3.2 muestra el flujo global de diseño basado en la reutilización de módulos IP. A partir de las especificaciones del sistema, los diseñadores utilizarán los módulos IP necesarios desde las bibliotecas IP, que pueden ser internas o externas, y proveedores IP. Para bloques IP de bibliotecas externas o de proveedores se necesita un proceso de verificación IP. Tras el cual, los diseñadores pueden utilizar la metodología de reutilización para desarrollar el módulo de una forma más eficiente que el diseño original. Por último, se realiza el testeo del módulo IP, y una vez superado el diseño, puede añadirse a la biblioteca IP interna para su posterior reutilización.

La estandarización, en lo que se refiere a la búsqueda de descripciones del sistema y formatos de biblioteca de síntesis que sean portables, es todavía una de las cuestiones

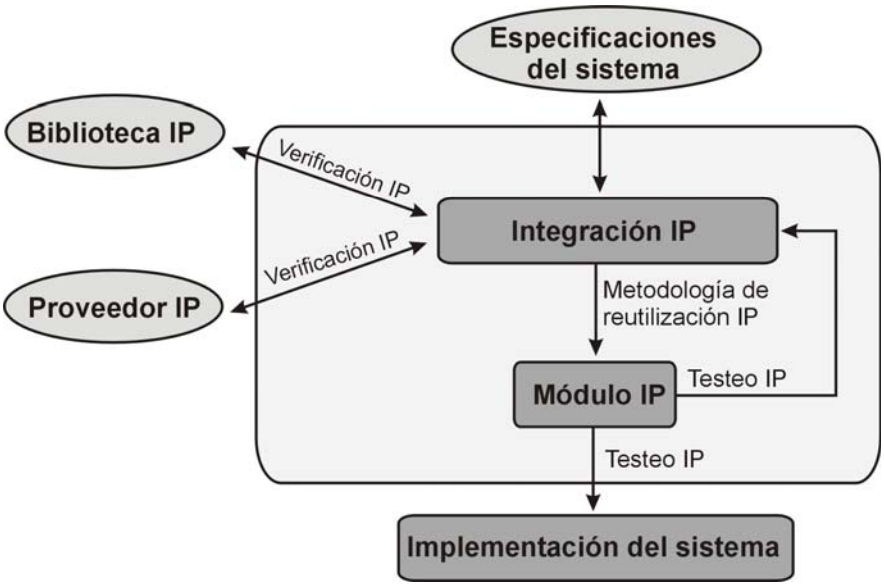

Figura 3.2. Flujo de diseño basado en la reutilización de módulos IP.

críticas en el diseño de un SoC. La figura 3.3 muestra los principales sectores de la industria involucrados en el diseño de SoCs. En estos sectores se pueden incluir los diseños que intervienen en el desarrollo de aplicaciones embebidas, vendedores de herramientas para la automatización del diseño electrónico (*Electronic Design Automation*, EDA), proveedores de módulos reutilizables y vendedores de obleas de silicio. Se están realizando grandes esfuerzos para la estandarización de todos los aspectos relacionados con el diseño de un SoC, entre ellos los relacionados con la reutilización de módulos.

En los últimos años, varias organizaciones, como *Virtual Socket Interface Alliance* (VSI *Alliance*) [VSI08, VSI02] y *Virtual Component Exchange* (VCX) [VCX08], han trabajado en el desarrollo de SoCs, propiedad intelectual y estándares para la reutilización, con el fin de aumentar la productividad del diseño SoC. VSI Alliance fue creada en 1996 por líderes mundiales en semiconductor, Semiconductor IP (SIP) y empresas de EDA, para ayudar a la industria en la definición y direccionamiento

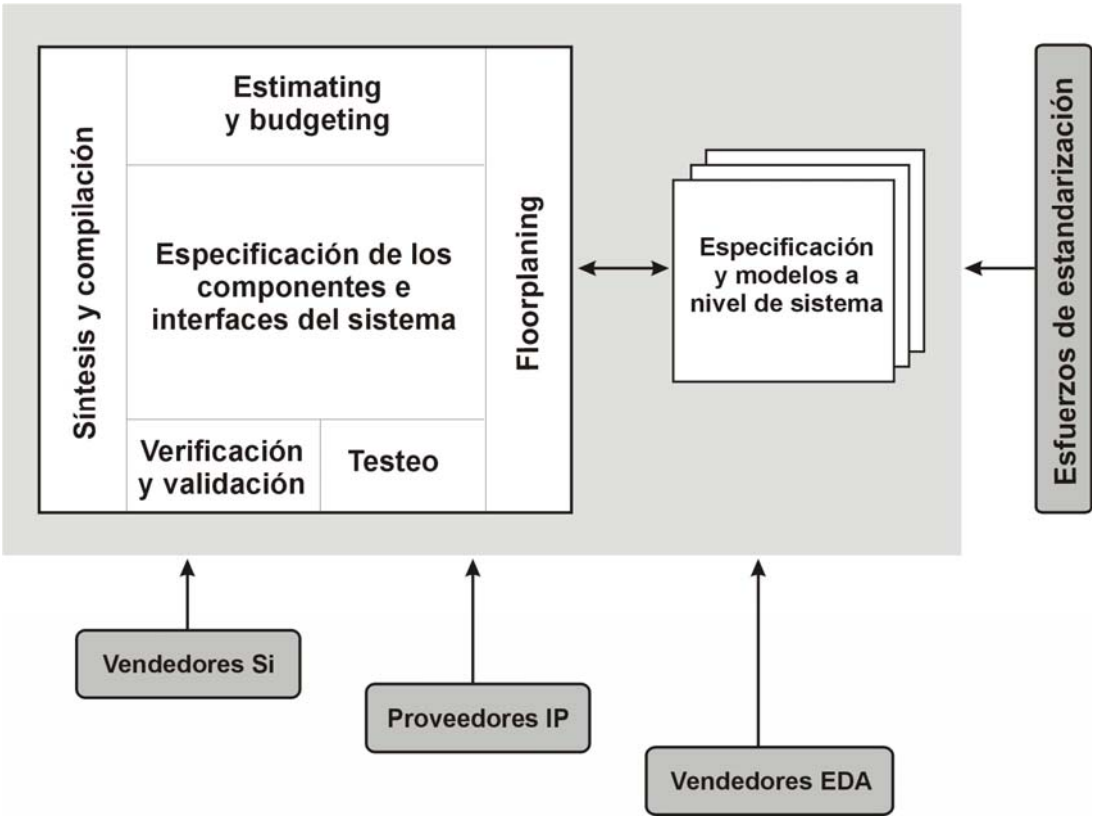

Figura 3.3. Entorno de diseño de la arquitectura de un SoC.

de los cambios tan importantes que ha ido experimentado en los últimos años el desarrollo de los SoCs. VSI Alliance ha realizado desde entonces un importante trabajo para que la industria sea capaz de entender cada vez más la importancia del diseño basado en la reutilización. Al mismo tiempo, ha conseguido avances muy importantes en la búsqueda de aspectos clave que deben abordarse para dar soporte a la efectividad y extensión de esta metodología de diseño. *Virtual Component Exchange* fue creada en 1998 con el objetivo de desarrollar un modo de gestionar el intercambio de módulos IP y determinar cuestiones legales relacionadas con estos componentes virtuales que pueden reducir considerablemente el proceso de desarrollo del producto. La VCX ha lanzado varios grupos de trabajo de desarrollo que colaboran con importantes empresas, como Hitachi, Motorola, Sun Microsystems y Toshiba, para definir los estándares y normas de intercambio de los módulos IP.

En la reutilización de diseños los procesos de simulación, verificación, estimación, síntesis y test son muy importantes. En este sentido, la *International Technology Roadmap for Semiconductors* [GAR00, INT07] recomienda nuevos estándares para las descripciones de sistemas y formatos de bibliotecas portables para la síntesis de los mismos. Por otro lado, *System Level Design Language* (SLDL) [KAM99, SLD08, STE98] es una iniciativa de estándares que desarrolla entornos de lenguaje interoperables para la especificación y diseño a nivel de sistemas embebidos. SLDL describe el comportamiento del sistema y las restricciones del diseño antes de la división *hardware*/*software*. De esta forma, y con el uso de las técnicas y herramientas apropiadas, es posible la verificación y validación a nivel de sistema.

El modelado orientado a objetos puede ayudar a reducir la complejidad de diseño. Los lenguajes orientados a objetos soportan el modelo de computación reactivo, que es más abstracto que los típicos lenguajes de descripción *hardware*. Este modelo es muy útil durante los primeros niveles del diseño para verificar las decisiones fundamentales de diseño de alto nivel antes de invertir un esfuerzo adicional en el diseño detallado. El grupo *IEEE Design Automation Standards Committee Object Oriented-VHDL* trabaja en el desarrollo de un estándar para VHDL orientado a objetos.

En [KEA00] se recoge una metodología efectiva para crear diseños reutilizables que van a formar parte de un SoC. En este sentido, trata de desarrollar y documentar el diseño basado en la reutilización y mostrar las ventajas que ofrece para cumplir

objetivos de tiempos de desarrollo, costes y complejidad de los sistemas. En este manual se describen unas directrices a seguir para el diseño de bloques reutilizables y para integrar estos bloques en chips de gran capacidad. Se describen métodos para ajustar los módulos reutilizables dentro de la metodología de desarrollo de un SoC, para diseñar los distintos tipos de módulos reutilizables, integrar estos módulos dentro de un diseño SoC y verificar la funcionalidad y tiempos en los diseños SoCs de gran escala.

A pesar de estos intentos para la estandarización de todo lo relacionado con el diseño basado en la reutilización, el continuo avance en la tecnología del silicio y las herramientas de diseño hace que no haya una única metodología que pueda ofrecer una solución permanente que sea adecuada. Como se ha visto en la sección anterior, existen reglas y guías eficaces para la producción de módulos IP; sin embargo, será el diseñador de dichos módulos el que deberá verificar estas reglas manualmente ya que no hay ningún método que lo haga automáticamente.

3.3. Riesgos del diseño basado en la reutilización

El diseño basado en la reutilización de módulos es una metodología que presenta grandes expectativas de futuro debido a las ventajas que ofrece en cuanto a reducción de la complejidad, costes y tiempos de desarrollo del diseño. Además, es posible conseguir todo lo anterior con una alta productividad. Sin embargo, existen varios problemas asociados a esta metodología de diseño que no se habían considerado hasta su aparición. Uno de ellos es el de la protección de la propiedad intelectual de los módulos IP a compartir [CHA00, VSI08]. La reutilización de diseños obliga a que los ingenieros cooperen entre sí y que compartan su datos, conocimientos y experiencia. Los detalles del diseño son documentados y, en muchos casos, hechos públicos para una mejor y más adecuada reutilización. En los últimos años, han surgido muchas herramientas de diseño de módulos IP que permiten que los equipos de diseño puedan colaborar, aún estando geográficamente separadas, por medio de Internet. Pero al mismo tiempo esto hace que la piratería de los módulos IP y las infracciones relacionadas con el uso de estos módulos sean ahora mucho más fáciles.

Debido a que los bloques reutilizables están diseñados para ser modulares e integrados con otros componentes, es posible para un tercero vender este bloque como si fuese de su propiedad, teniendo sólo que conocer la interfaz y la función que realiza y sin tener que conocer la arquitectura o la implementación. Un problema asociado a este hecho es el riesgo del uso ilegal. Por ejemplo, al utilizar un módulo de diseño, la empresa compradora puede incumplir parte de los acuerdos e incluir el módulo en otros diseños para los que no disponga de licencia o redistribuirlo a terceras empresas. En estos casos conviene disponer de mecanismos para que el autor de los módulos pueda reclamar los derechos que le correspondan. Es más, la viabilidad de las metodologías de diseño basadas en la reutilización depende del desarrollo de estos mecanismos de protección de la propiedad intelectual. Entre los objetivos de la protección de la propiedad intelectual [MAC01] se incluyen los siguientes:

- que los autores puedan proteger sus módulos contra el uso no autorizado;
- proteger todos los tipos de datos del diseño de los módulos reutilizables;
- poder realizar un rastreo de dicho componente para conocer de alguna forma la utilización del mismo.

En la industria del diseño electrónico hay varias formas de protección para la propiedad intelectual que incluyen patentes, derechos de autor (*copyrights*), marcas registradas y secretos comerciales [FER94]. Sin embago, como se detallará más adelante, todos estos métodos son insuficientes y/o no aplicables para la protección de la propiedad intelectual de módulos reutilizables [KEA00]. En los últimos años se ha experimentado un rápido desarrollo de técnicas que ocultan una firma, también conocidas como técnicas de *watermarking* o de ocultación de la información (*data hiding*). Las técnicas de *watermarking* para soportes compartidos introducen información digital en el módulo IP para conseguir objetivos como identificación y derechos de autor. Aunque se han propuesto numerosas técnicas que ocultan una firma digital en imagen, video, voz y texto [COX07], éstas no ofrecen una solución adecuada para la protección de la propiedad intelectual de diseños. Ésto se debe a que la mayoría de estas técnicas modifica un gran número de componentes en el objeto. Mientras que en datos multimedia estas modificaciones son inapreciables para la vista y el oído humano, cuando se trata de diseños y herramientas CAD estas alteraciones tienen un impacto que, en la mayoría de los casos, es inaceptable.

De entre todas las barreras técnicas y no técnicas asociadas a la metodología de diseño basada en la reutilización que dificultan su avance y desarrollo, la protección de la propiedad intelectual es una de las más preocupantes. En los últimos años, la protección de la propiedad intelectual en soportes compartidos ha generado gran interés en la comunidad científica, y en la actualidad es el centro de muchas líneas de investigación [LAC98a, KAH98a, CHA98]. En los siguientes apartados se estudian las técnicas de protección de la propiedad intelectual, en concreto de las técnicas de *watermarking* para contenidos digitales en general. Tras ver los aspectos más importantes de estás técnicas se estudiarán más profundamente aquéllas que van dirigidas a la protección de módulos reutilizables.

3.4. Técnicas de protección de la propiedad intelectual

De forma general, en el desarrollo de un diseño basado en la reutilización se necesita conocer las regulaciones y los principios de los métodos de protección asociados a los módulos reutilizables. Con ésto se consigue la protección de los derechos de autor de los propietarios de los módulos IP, y al mismo tiempo, la protección de los derechos de autor de los que desarrollan sus diseños a partir de estos módulos.

Existen varias formas de protección de la propiedad intelectual, pero no todas se aplican de igual forma a los diferentes tipos de bloques IP. De acuerdo con la *VSI Alliance* [VSI02] existen tres técnicas o métodos para la seguridad de un módulo IP: técnicas disuasorias, técnicas de protección y técnicas de detección. Con las técnicas disuasorias se puede disuadir de una infracción cuando ésta se está intentando realizar, con la ayuda de medios legales y sin que se proporcione ningún tipo de protección física. Por otro lado, las técnicas de protección intentan prevenir los usos no autorizados de los módulos IP por medio de mecanismos tales como los acuerdos de licencia y la encriptación. Por último, con las técnicas de detección se pueden detectar usos autorizados y no autorizados de un determinado bloque IP, de tal forma que si el uso es no autorizado se pueda llegar a conocer la fuente u origen de tal infracción y poder así emprender acciones legales contra tal abuso. La elección de la técnica de seguridad más

apropiada va a estar determinada por el punto de aplicación específico del módulo reutilizable en su ciclo de desarrollo.

3.4.1. Técnicas de disuasión

Los mecanismos tradicionales empleados como técnicas de disuasión son las patentes, los derechos de autor y los secretos comerciales. Estos métodos proporcionan varios niveles de protección y el principal objetivo de su aplicación es animar a la comercialización de los diseños y proporcionar los derechos que correspondan a los autores, todo esto en un corto período de tiempo. En el caso de que se hayan empleado técnicas de disuasión, antes de iniciar el desarrollo de un módulo IP, se debe realizar una búsqueda y análisis de las patentes, derechos de autor y secretos comerciales. Con esto se pueden llegar a determinar abusos de otros derechos de propiedad intelectual y ayudar así a poder establecer el valor que posee el autor de un determinado diseño.

Una patente es un derecho exclusivo concedido a una invención, es decir, un producto que aporta, en general, una nueva manera de hacer algo o una nueva solución técnica a un problema. Para solicitar una patente se necesita una amplia documentación sobre la invención, el autor debe proporcionar pruebas de novedad y utilidad y además debe dar las directrices necesarias para la implementación del objeto de patente. La protección de una patente significa que la invención no puede ser confeccionada, utilizada, distribuida o vendida comercialmente sin el consentimiento del titular de la patente. La patente tendrá validez en en el país en el que se solicitó, de conformidad con la legislación aplicable. El cumplimiento de los derechos de patente normalmente se hace respetar en los tribunales que tienen la potestad de sancionar las infracciones a la patente. El titular de una patente tiene el derecho de decidir quién puede o no puede utilizar la invención patentada durante el período en el que está protegida.

Los derechos de autor son un conjunto de normas y principios que regulan los derechos morales y patrimoniales que la ley concede a los autores por el solo hecho de la creación de una obra literaria, artística o científica, tanto publicada o que todavía no se haya publicado. Los derechos morales son aquellos que se encuentran ligados al autor de manera permanente y son irrenunciables e imprescriptibles. Los derechos patrimoniales son aquellos que permiten de manera exclusiva la explotación de la obra

hasta un plazo contado a partir de la muerte del último de los autores que, en la mayoría de los países, está establecido en 70 años. El derecho de autor es una técnica de disuasión que sólo prohíbe copiar las expresiones de una idea, no la idea en sí misma, como lo harían las patentes. Por lo tanto, los derechos de autor son mucho más fáciles de conseguir y ofrecen un mayor período de protección que una patente.

Las leyes que regulan los secretos comerciales tienen un campo de aplicación mayor que las patentes y los derechos de autor. Por una parte, los secretos comerciales pueden concernir a invenciones o procesos de fabricación que no satisfagan los criterios de patentabilidad y, por consiguiente, puedan protegerse únicamente como secretos comerciales. Este podría ser el caso de las listas de clientes o de procesos de fabricación que no sean lo suficientemente inventivos para que se les conceda una patente. Por otra parte, los secretos comerciales pueden concernir a invenciones que satisfagan los criterios de patentabilidad y, por consiguiente, puedan ser protegidos por patentes. En este caso, el organismo competente deberá decidir si patenta la invención o la considera como secreto comercial. La protección de los secretos comerciales tiene la ventaja de no estar sujeta a límites temporales (la protección continúa de manera indefinida siempre que el secreto no se revele al público), tener un efecto inmediato, no entrañar costos de registro y no tener que obedecer a requisitos como la divulgación de la información a una autoridad gubernamental. No obstante, un secreto comercial es más difícil de hacer respetar que una patente y, una vez que el secreto se divulga, todo el mundo puede tener acceso al mismo y utilizarlo como le plazca. Además, un secreto comercial puede ser patentado por cualquier otra persona que haya obtenido la información pertinente por medios legítimos.

3.4.2. Técnicas de protección

Para los propietarios de módulos IP de gran coste, la pérdida de control de los datos de diseño EDA pueden ocasionar grandes pérdidas. Por este motivo, es importante proteger dichos módulos con un alto nivel de seguridad, como la que proporciona la encriptación. Al mismo tiempo, es conveniente proporcionar a los usuarios los medios adecuados para evaluar la adquisición de estos módulos reutilizables.

La encriptación es el proceso mediante el cual cierta información es cifrada de forma que el resultado sea ilegible a menos que se conozcan los datos necesarios para su interpretación. Es una medida de seguridad utilizada para, que en el momento de almacenar o transmitir información sensible, ésta no pueda ser obtenida con facilidad por terceros. Existe además un proceso de desencriptación a través del cuál la información puede ser llevada de nuevo a su estado original, aunque existen métodos de encriptación que no pueden ser revertidos. Cualquier esquema de encriptación utilizado debe pasar mínimos tests que verifiquen su uso. La encriptación de código HDL [OBR94] ofrece a los usuarios la posibilidad de tener acceso a una versión sintetizable de un bloque IP pero sin llegar a tener acceso al código fuente. Este mecanismo de protección permite manipular el bloque, integrarlo con otros componentes y procesarlo a través de las herramientas EDA utilizadas en el flujo de diseño hacia la implementación, sin necesidad de que el usuario conozca la estructura específica de dicho bloque. El problema es que no todos los vendedores de herramientas EDA disponen de otras que soporten la encriptación. Cuantos más vendedores de herramientas EDA establezcan sus estrategias de protección de módulos IP, más alto llegará a ser el poder de la encriptación como mecanismo de protección, a pesar de los problemas de la disposición de los usuarios a pagar por estos servicios.

La protección del *hardware* es una forma eficiente de proteger directamente los datos de diseño EDA de un determinado bloque IP. Este mecanismo de protección consiste en no publicar los datos de diseño, excepto en varias formas indirectas. Por ejemplo, en forma máscaras para el chip completo, o en forma de diseño programable como FPGA para usarlo en plataformas *hardware*. Ninguna de estas dos formas permite tener acceso directo al diseño y ambas aumentan el nivel de dificultad para acceder a la fuente de información que define el bloque IP.

3.4.3. Técnicas de detección

Las técnicas de detección pueden ser consideradas como un compromiso entre las técnicas disuasorias y las de protección. Para diseños de media y gran escala, las técnicas de detección son la mejor opción para la protección de estos diseños porque tienen un bajo coste y proporcionan una verificación relativamente confiable de los módulos IP. Existen varios mecanismos que permiten la identificación del propietario

de un bloque IP. Estos mecanismos se diferencian en sus niveles de protección; algunos de ellos son indetectables y otros se muestran abiertamente, son fáciles de observar y son usados como un medio para seguir el rastro de un determinado bloque IP.

La marcación y el rastreo son métodos de detección que consisten en colocar etiquetas a los componentes para poder realizar un rastreo de estos (generalmente en la fase de la fabricación) y permiten guardar los registros apropiados. Un ejemplo de tal esquema es el de *VSI Alliance*, "*Virtual Component Identification: Physical Tagging Estándar*" [VSI08]. Esta técnica simplemente crea una etiqueta para cualquier componente que forma parte de un diseño de un circuito integrado. Esta etiqueta contiene información sobre el título, la propiedad, la fecha de origen, el número de acontecimientos, etc., y permite a una entidad registrar empleos, reconocer la propiedad y administrar pagos de derechos y acontecimientos. Es probable que en el futuro la infraestructura evolucione de tal forma que un cuerpo independiente lleve los registros de propiedad intelectual, etiquetaje, marcación e incluso firmas digitales.

La firma o huella (*fingerprint*) digital es una secuencia de símbolos finita y que aplicada a un bloque IP puede actuar como un identificador exclusivo y único. La huella es, generalmente, la característica nativa del componente, mientras que una firma puede ser la representación de esa huella. El *fingerprinting* digital [BON95, CAL99, PFI96] a veces es llamado *watermarking* pasivo. La firma es una representación de características únicas de un bloque, se utilizan características o atributos pre-existentes e inherentes de un bloque IP. Las ventajas que ofrece este esquema de detección es que consigue combatir una forma de ataque denominada *tampering* [MIN05] o cambios en el bloque, el empleo de flujos de diseño estándar, y la velocidad de implementación sin cambios en el funcionamiento. Las huellas digitales no se prestan para realizar ingeniería inversa del bloque y son muy convenientes para ser recogidas en bases de datos (por ejemplo la del FBI). Las limitaciones incluyen el hecho de que una huella digital no incluye información útil como el dueño, el nombre del componente, etc., y por tanto presenta alguna debilidad en relación con mecanismos de marcación simple. Una revisión simple de un componente establece una nueva huella digital. Es posible introducir la huella digital de un bloque en la mayor parte de niveles de abstracción en la jerarquía de diseño.

Por otra parte, el *watermarking* digital [COX07] es un esquema de detección que proporciona una fuerza disuasoria a posibles infractores ofreciendo la posibilidad de demostrar la propiedad de un determinado bloque IP. El proceso de *watermarking* consiste en la introducción de una firma digital en un bloque IP en un determinado nivel de abstracción del diseño, utilizando los rasgos intrínsecos y la estructura de dicho nivel. El *watermarking* es un área muy interesante para la investigación, tanto en el entorno de círculos industriales como en el de los académicos. De entre todos los métodos de detección, el *watermarking* es uno de los más prácticos y utilizados. Antes de presentar las técnicas de *watermarking* para módulos IP, en la siguiente sección se va a dar una visión general de las técnicas de *watermarking* para contenidos digitales en general. Será en la sección 3.6 donde se hará un estudio detallado de la adaptación y aplicación de esta técnica de detección a módulos IP.

3.5. Técnicas de *watermarking*

La inclusión de información oculta con la finalidad de proteger los derechos de la propiedad intelectual pretende añadir marcas en la información, siendo deseable que no sea posible eliminarlas sin degradar seriamente el contenido de la información. Por otra parte, a diferencia de la criptografía, la introducción de información oculta no restringe el acceso a la información. El *watermarking* está íntimamente relacionado con la ocultación de información (*data hiding*) y la esteganografía [COX07], que se estudia a continuación. Estos tres campos, el *watermarking*, la esteganografía y la ocultación de la información, se superponen y comparten muchas técnicas. Sin embargo, hay diferencias fundamentales que afectan a las exigencias, y así el diseño, de una solución técnica.

3.5.1. Ocultación de la información, esteganografía y *watermarking*

La ocultación de información es un término general que abarca una amplia gama de problemas más allá de los de mensajes que se introducen en el contenido. El término ocultación en este contexto puede referirse a hacer la información imperceptible o a

ocultar la existencia de la información secreta. Las distintas ediciones del *International Workshop on Information Hiding* recogen algunas de las investigaciones realizadas en este campo [FUR07].

El término esteganografía [COX07] deriva de las palabras griegas *steganos*, (encubierto), y *graphia*, (escritura); literalmente significa "escritura encubierta". Es el arte de la comunicación oculta, la existencia misma de un mensaje es secreta y normalmente se interpreta como la ocultación de una información en otra. Es decir, la esteganografía consiste en ocultar en el interior de una información, aparentemente inocua, otro tipo de información (cifrada o no). Como el propósito de esteganografía es el de tener una comunicación encubierta entre dos partes cuya existencia es desconocida para un posible atacante, un ataque exitoso consistiría en descubrir la existencia de esta comunicación. Algunas de las aplicaciones de la esteganografía son las comunicaciones secretas y el control de acceso a datos multimedia.

El *watermarking* [BAR04, EGG02] consiste en alterar de forma imperceptible un determinado contenido (imagen, video, música, etc.) para introducir un mensaje sobre ese mismo contenido y sobre el autor y/o propietario, que sea difícil de eliminar. Por tanto, el *watermarking* es una particularización del ocultamiento de datos con el requisito adicional de la robustez ante los posibles ataques. Las aplicaciones propuestas basadas en el *watermarking* son muchas y entre otras tareas incluyen la identificación del propietario de los derechos de autor. El uso de marcas de agua (*watermarks*) es casi tan antiguo como la fabricación de papel. Tradicionalmente, el *watermarking* se ha empleado para identificar la autenticidad de documentos oficiales, en formato de papel. El nombre de esta técnica proviene de la técnica original que usaba marcas semitransparentes hechas en el papel. Durante cientos de años, cualquiera que poseyera o fabricase un documento u obra de arte de apreciado valor lo marcaba con un sello de identificación o marca de agua (visible o no), no sólo para establecer su propiedad, origen o autenticidad, sino para desalentar a aquellos que pudieran intentar robarlo. Recientemente, el interés en la aplicación de técnicas similares para proteger e identificar contenidos en otros formatos ha crecido considerablemente. En particular, el *watermarking* se ha aplicado para la protección de la propiedad intelectual en forma digital.

3.5.2. *Watermarking* digital

A finales de la década de los noventa surgió un gran interés en el *watermarking* para sistemas y contenidos digitales [COX01]. El aumento repentino del interés en el *watermarking* se produjo debido a la creciente preocupación e interés por la protección de los derechos de autor. El riesgo de piratería está exacerbado por la proliferación de los dispositivos de grabación digital de elevada capacidad, los propietarios buscan tecnologías que prometen la protección de sus derechos con lo que la primera tecnología usada con este propósito fue la criptografía [LUC04, MEN96, STI95]. Es, probablemente, el método más común para la protección de cualquier contenido digital. El contenido es encriptado antes de la entrega, y después se proporciona una clave de desencriptación sólamente a aquellos que han conseguido copias legítimas. El archivo encriptado puede estar disponible vía Internet, pero sería de poca utilidad para un pirata que no disponga de la clave apropiada. Desafortunadamente, esta técnica no puede controlar cómo maneja un cliente legítimo el contenido después de la desencriptación. Un pirata puede conseguir el producto, usar la clave de desencriptación para obtener una copia desprotegida, y después distribuir copias ilegales. En otras palabras, la criptografía puede proteger el contenido en tránsito, pero una vez desencriptado, el contenido no tiene ninguna protección. Por tanto, los procesos criptográficos permiten proteger la adquisición legal de la información, pero una vez obtenida la información se pueden revender copias exactas. Debido a esto hay una fuerte necesidad de buscar una alternativa o complemento a la criptografía que sea capaz de proteger el contenido incluso después de que sea desencriptado. La idea es encontrar o crear un sistema de seguimiento de las copias para la protección de los derechos de autor, que también se utilice en el caso de adquisición legal para distribución fraudulenta (copias ilegales).

El *watermarking* puede cubrir esta necesidad porque coloca la información dentro del contenido donde nunca se pueda eliminar durante el uso normal de este contenido. Una marca digital puede ser diseñada para sobrevivir a todos los procesos de desencriptación, nueva encriptación, compresión, conversión digital/analógico y cambios en el formato del archivo. El principal foco de atención para la aplicación del *watermarking* digital han sido las fotografías, audio y vídeo digitales, pero en otros contenidos, tales como imágenes binarias, texto, modelos de tres dimensiones, código

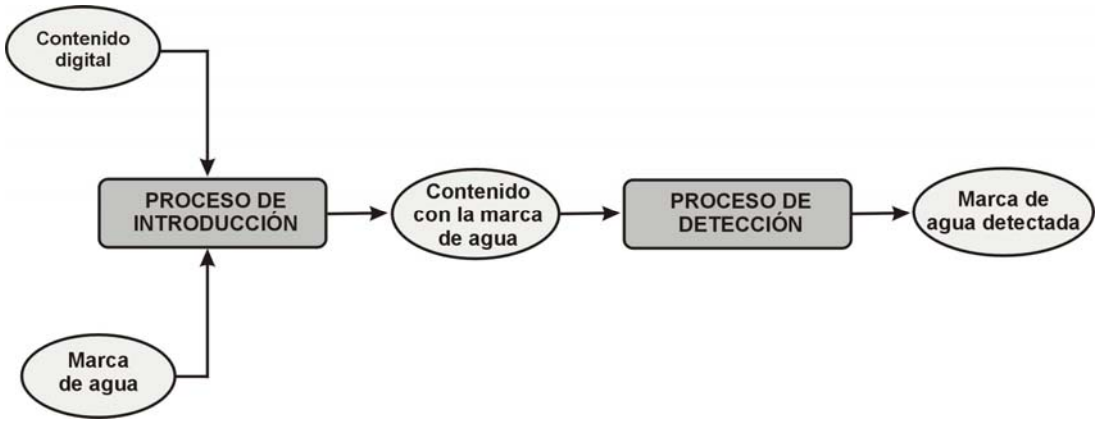

Figura 3.4. Procesos en un sistema genérico de *watermarking* digital.

ejecutable y circuitos integrados [GAN03, LAC98a, KAH98a], también se han introducido marcas de agua.

En general, un sistema de *watermarking* consiste en un proceso de introducción (*embedder*) y otro de detección (*detector*), como se puede ver en la figura 3.4. En el proceso de introducción se toman dos entradas, una de ellas el mensaje codificado en forma de marca de agua y la otra el contenido digital en el que se quiere introducir dicho mensaje. La salida de este proceso de introducción corresponde al contenido con la marca de agua. Una marca de agua digital se puede definir como un código de identificación que se encuentra permanentemente incrustado en un determinado contenido digital y que puede contener información acerca del propietario, de los derechos de autor, el creador, el usuario autorizado, el número de copias o reproducciones autorizadas, el terminal autorizado, etc. Las marcas de agua no siempre necesitan ser ocultadas; además de sistemas que usan marcas de agua invisibles, existen otros que emplean marcas de agua digitales visibles, diseñadas para ser fácilmente percibidas y claramente identificadas por el propietario. Sin embargo, la mayoría de los desarrollos se han encaminado hacia el *watermarking* imperceptible, invisible o inaudible. En el proceso de detección se intenta determinar si hay una marca de agua en un determinado contenido digital, y si es así, extraer la información codificada.

3.5.3. Aplicaciones del *watermarking*

El *watermarking* se puede utilizar en una gran variedad de aplicaciones. En general, si para una determinada aplicación resulta útil asociar alguna información adicional con un determinado contenido, se puede intentar hacer con la introducción de marcas de agua. Existen otras formas de asociar información con un contenido, aunque la cuestión es llegar a saber cuándo el *watermarking* es la mejor opción y qué puede hacer esta técnica que no se pueda conseguir con otras alternativas. Hay tres aspectos muy importantes en el *watermarking* que pueden llevar a elegir esta técnica como la más apropiada para una determinada aplicación: las marcas de agua son imperceptibles, inseparables de los contenidos en los que han sido introducidas y sufren las mismas transformaciones que estos contenidos. Al mismo tiempo, estos aspectos pueden llevar a considerar que el *watermarking* no es adecuado para ciertas aplicaciones.

En concreto, se ha utilizado como una herramienta muy útil para muchas aplicaciones con objeto de prevenir copias ilegales y proteger los derechos de autor. En la prevención de copia ilegial, la marca de agua se utiliza para informar a los dispositivos *hardware* o *software* de que la copia no está permitida. En las aplicaciones de protección de los derechos de autor la marca de agua se usa para identificar al propietario de los derechos de autor y asegurar el pago adecuado. Estas dos aplicaciones han sido los principales focos de atención en el desarrollo e investigación de técnicas de *watermarking*. Sin embargo, existen otras aplicaciones más concretas para las técnicas de *watermarking*, por ejemplo, control de difusión, pruebas de autor, rastreo de transacciones, autentificación, control de copias y control de dispositivos [COX07]. El potencial del *watermarking* para llegar a hacer cumplir el propósito de algunas de estas aplicaciones serán consideradas más adelante, teniendo en cuenta el caso particular del *watermarking* para módulos IP.

3.5.4. Requisitos de un sistema de *watermarking*

La bondad de una determinada técnica de *watermarking* se puede evaluar en términos de un conjunto de propiedades. Existe un gran número de publicaciones en las que se discuten los requisitos que deben cumplir las marcas de agua [COX07, PET00, PIV98]. La importancia de los requisitos o propiedades de los sistemas de *watermarking*

depende de la aplicación y contenido digital para las que dicho sistema esté diseñado. En general, algunas de las propiedades deseables de un sistema de marcas de agua son la robustez, la seguridad, la imperceptibilidad, el bajo costo computacional y la baja probabilidad de error, tal como se detalla a continuación.

3.5.4.1. Robustez y seguridad

Muchas de las aplicaciones necesitan que las marcas de agua puedan seguir siendo detectadas incluso si el contenido en el que fueron insertadas ha sido modificado de alguna forma después de la introducción de la marca de agua. Las marcas de agua diseñadas para poder superar un uso normal y legítimo del contenido reciben la denominación de marcas de agua robustas. En el diseño de una marca de agua robusta es importante identificar los procesos específicos a los que probablemente puede verse sometido el contenido entre la introducción y la detección de la marca de agua. Es importante establecer bien la diferencia entre marcas de agua robustas y marcas de agua seguras. Aunque algunos autores [ABD06] incluyen dentro de la robustez la resistencia contra los distintos tipos de ataques, en este trabajo se acepta como robustez la resistencia frente a aquellas modificaciones producidas en el procesamiento normal al que estarán expuestos los contenidos con la marca de agua [FRI99, COX07]. Por lo tanto, mientras que las marcas de agua robustas están diseñadas para poder superar un procesamiento normal del contenido, las marcas de agua seguras estan diseñadas para resistir cualquier intento de sabotaje [PET98]. Debido a que en la mayoría de las aplicaciones una marca de agua no puede desempeñar su función si se hace indetectable, la robustez es una propiedad necesaria si la marca de agua tiene que ser segura. En otras palabras, si una marca de agua se puede eliminar por un procesamiento normal, no se puede considerar que esta marca de agua sea segura. Sin embargo, la robustez no es suficiente para garantizar la seguridad, porque las marcas de agua seguras deben además ser capaces de superar procesos que estén diseñados específicamente para eliminarlas. De esta forma, mientras que el diseñador de una marca de agua segura debe considerar todos los posibles ataques, el diseñador de una marca de agua robusta puede limitarse a considerar el rango de todos los procesamientos posibles.

3.5.4.2. Imperceptibilidad e indetectabilidad

La imperceptibilidad y la indetectabilidad de las marcas de agua son dos conceptos que tienden a confundirse frecuentemente, aunque son distintos. La imperceptibilidad o transparencia de la marca tiene como base el comportamiento del sistema perceptual humano. Una marca de agua es imperceptible (transparente) si la degradación que causa en los contenidos digitales donde se ha insertado es muy difícil de apreciar. La indetectabilidad está relacionada con el modelo estadístico del contenido antes y después de ser marcado. Se dice que la marca es indetectable si después de haberla insertado, el contenido marcado conserva las mismas propiedades estadísticas que su original.

3.5.4.3. Viabilidad del sistema

Toda tecnología que pretenda ser comercializada, debe tener en cuenta varios aspectos, entre ellos el coste computacional, el coste económico y la escalabilidad del sistema. Los requerimientos computacionales exigen a los sistemas de marcas de agua simplicidad, pero ésta puede significar la reducción de la resistencia a las manipulaciones. Sin embargo, hay que tener en cuenta que la velocidad de los ordenadores se dobla anualmente, de manera que un algoritmo que hoy no parezca razonable, podrá rápidamente convertirse en factible. Es muy deseable diseñar sistemas de marcas de agua que sean escalables con cada generación de ordenadores.

3.5.4.4. Baja probabilidad de error

En la mayoría de los sistemas de marcas de agua es muy importante distinguir entre los objetos que contienen una marca y los que no [MIL99]. La probabilidad de error al detectar una marca debe ser muy pequeña. Se denomina probabilidad de falso negativo a la probabilidad de que, estando presente una marca en determinado contenido, el detector asuma que no hay tal marca. Por otro lado, la probabilidad de falso positivo es la probabilidad de que no estando la marca presente en un contenido, el detector determine que la marca está presente.

En general, todos los aspectos y requisitos en el desarrollo de un sistema de marcas de aguas digitales requieren de la particularización a un tipo de contenido digital, pues el diseño de un sistema de estas características logra mejores resultados si se realiza en función de la respuesta humana al medio a percibir. Hasta ahora se han tratado los sistemas de *watermarking* digital y los aspectos relacionados con estos sistemas para cualquier contenido digital. El resto de este capítulo centra su atención en sistemas de *watermarking* para la protección de componentes virtuales o módulos reutilizables empleados en las nuevas metodologías de diseño de sistemas digitales.

3.6. *Watermarking* para módulos reutilizables

A pesar de las numerosas leyes y normativas existentes para poner fin a las infracciones relacionadas con la propiedad intelectual, los autores y propietarios de módulos reutilizables están realizando grandes esfuerzos para mantener estos contenidos fuera del alcance de la piratería. Los primeros intentos realizados en el desarrollo de métodos de protección de módulos reutilizables consistieron en la grabación de números de serie en el chip, adición de código redundante al código original, empleo de varios estilos de programación, etc. Sin embargo, todos estos métodos de protección son vulnerables a distintos tipos de ataques. Por ejemplo, los números de serie se pueden modificar o eliminar, los trozos del código que son menos utilizados se puede detectar y eliminar, las variables pueden ser renombradas, etc. Estos esfuerzos de protección no son adecuados porque, entre otros motivos, el proceso de protección se realiza de forma independiente al de diseño e implementación del módulo reutilizable. En estos casos, el diseñador no tiene muchas ventajas sobre los atacantes. Es más, los diseñadores no tienen la experiencia que tienen los atacantes profesionales y no son conscientes de lo poderosas que pueden llegar a ser las herramientas empleadas en esos ataques. Como conclusión se podría decir que, para que un método de protección de la propiedad intelectual sea efectivo, dicha protección tiene que realizarse al mismo tiempo que los procesos de diseño e implementación, ya que durante esos procesos el diseñador posee un control absoluto sobre el módulo, que es imposible tener cuando estos procesos finalizan y el módulo esté listo para ser utilizado.

El *watermarking* es actualmente una de las técnicas más utilizadas para la protección de bloques IP ya que puede aplicarse durante los procesos de diseño e implementación. Para contenidos digitales tales como texto, imagen, audio y video, la marca de agua se puede introducir realizando unos ligeros cambios en dicho contenido. A pesar de que ésto modifica el contenido original, la técnica empleada es útil mientras los usuarios no puedan apreciar la diferencia entre el contenido original y el que incorpora la marca de agua. En la aplicación del *watermarking* para módulos IP hay que tener en cuenta que la utilidad de módulo depende de su correcto funcionamiento. De esta forma, el desafío más importante en la aplicación de técnicas de *watermarking* para módulos IP consiste en ocultar la marca de agua dentro del módulo al mismo tiempo que se está realizando su diseño y sin que esto afecte a su funcionamiento.

3.6.1. Aplicación de técnicas de *watermarking* para bloques IP

El *watermarking* aplicado a módulos IP consiste básicamente en introducir una firma digital o marca de agua en dicho módulo con el propósito de identificar a su autor y/o propietario y demostrar así los derechos de autor. La copia del módulo implica también la copia de la firma digital. En caso de ser necesario, esta firma puede ser utilizada ante un tribunal para demostrar la propiedad del módulo o para comprobar los usuarios legalmente autorizados que tienen el derecho de distribuir copias. Es posible emplear una marca de agua a través de un diseño entero, dentro de múltiples bloques, dentro de un único bloque o incluso dentro de pequeñas áreas funcionales. Con lo anterior se consigue que estas partes del diseño no pueden ser copiadas sin el riesgo de que las marcas de agua permanezcan inalterables y puedan ser extraídas.

Para proteger los derechos de autor resulta muy interesante intentar utilizar las marcas de agua no sólo para identificar irrefutablemente al propietario de estos derechos. Esto es algo que no se consigue con otras técnicas que asocian información a un determinado módulo, de forma que esta información se pueda eliminar o falsificar fácilmente. Con el *watermarking* se puede llegar a conseguir el nivel de seguridad necesario para ofrecer una gran prueba de autoría si se restringe la disponibilidad del detector. Cuando un atacante no dispone de un detector, la eliminación de la marca de agua es extremadamente difícil de conseguir. Sin embargo, incluso en el caso de que la marca de agua no se pueda eliminar fácilmente, un atacante podría llegar a crear

incertidumbre sobre la prueba de autoría. Por ejemplo, este atacante podría utilizar su propio sistema de *watermarking* para hacer que su marca de agua estuviera presente en el módulo reutilizable considerado, que además contiene la marca de agua del autor y/o propietario. De esta forma, el organismo competente sería incapaz de determinar a quién corresponden los derechos de autor. Este problema se podría solucionar si se prueba que el módulo del atacante se ha obtenido a partir del módulo original.

Otra posible aplicación de las técnicas de *watermarking* para bloques reutilizables es la autentificación del contenido. El problema de la autentificación de los mensajes ha sido ampliamente estudiado en criptografía [STI95]. Un método criptográfico para el problema de la autentificación del contenido consiste en la creación de una firma digital. Básicamente, la firma digital es el resultado de aplicar cierto algoritmo matemático, denominado función *hash* [SCH96], a un determinado mensaje. Además se puede utilizar un algoritmo de encriptación de clave asimétirca, de tal forma que la clave para la encriptación de la firma digital es diferente de la que se necesita para desencriptarla. En el capítulo 4 se verá con más detalle la generación de una firma digital a partir de una determinada marca de agua. Como se estudiará en el capítulo 6, el uso de firmas digitales para la aplicación de las técnicas de *watermarking*, en concreto los algoritmos criptográficos empleados para su creación, resultan de gran utilidad en la prevención de distintas formas de ataque.

3.6.2. Objetivos y criterios de evaluación

Para los investigadores que trabajan en el desarrollo de las técnicas de *watermarking* es necesario realizar una evaluación y comparación de estas técnicas [ABD03]. Antes de evaluar un sistema de *watermarking* es necesario determinar qué propiedades o características hacen que este sistema sea mejor que otro. Los criterios de evaluación dependerán de la aplicación concreta para la que se desea utilizar el *watermarking*. En los últimos años, se ha incrementado el interés de la comunidad científica por el establecimiento de las definiciones preliminares de los requisitos o propiedades de un sistema de marcas de agua. En el apartado 3.4, se vieron algunas de las propiedades más importantes de los esquemas de *watermarking* para contenidos digitales. La importancia de cada una de ellas depende de las exigencias u objetivos marcados para la aplicación particular y del papel que juegue la marca de agua. De

hecho la interpretación de una determinada propiedad de *watermarking* puede variar con la aplicación. Aunque inicialmente la mayoría de las propiedades de las técnicas de *watermarking* fueron desarolladas para aplicaciones multimedia, algunas de ellas se pueden extrapolar para la evaluación de las técnicas de *watermarking* para módulos IP [ABD05, CHA00, KIR06]. Teniendo en cuenta estas propiedades y las necesidades específicas del diseño basado en la reutilización, a continuación se realiza un breve análisis de los requisitos u objetivos más importantes que cualquier técnica de *watermarking* para módulos reutilizables debería conseguir:

- **correcto funcionamiento**: este es uno de los aspectos más importantes. El correcto funcionamiento del módulo IP en cuanto a tiempos y requisitos del diseño no deberá verse afectado por la introducción de la marca de agua;
- **mínimas modificaciones en los recursos:** algunas técnicas de *watermaking* necesitan de recursos adicionales; la introducción de la marca de agua deberá ocasionar el menor impacto posible sobre características del sistema tales como el área, la velocidad y el consumo de potencia;
- **mínimo coste**: se refiere al coste computacional, de desarrollo, esfuerzo, etc., que supone la aplicación de una determinada técnica de *watermarking*; este coste ha de ser tan pequeño como sea posible;
- **transparencia**: el procedimiento para la introducción de la marca de agua no ha de afectar al diseño; es decir, este procedimiento debe ser fácil de integrar en el flujo de diseño sin modificar las herramientas de diseño;
- **información de la marca de agua**: el *watermarking* debe añadir suficiente información para identificar al autor del diseño, tal que pueda ser considerada como una prueba de autoría evidente ante un tribunal; sin embargo, la cantidad de información debe ser lo suficientemente pequeña para que su introducción en no suponga un alto incremento del tamaño del diseño y tampoco afecte a su funcionamiento;
- **identificación del autor**: la técnica de *watermarking* debe ser capaz de identificar al autor de forma única y resultar imposible que otros pudieran reclamar la propiedad sobre el módulo;

- **protección por partes**: la marca de agua deberá ser distribuida por todo el diseño para facilitar la protección, no sólo del diseño completo, sino también de sus partes, es decir, de los distintos módulos IP que formen parte del mismo;
- **robustez**: particularizando para módulos IP, la robustez hace referencia a la habilidad del sistema de *watermarking* para detectar la marca de agua después de las operaciones de procesamiento normales a las que se puede ver sometido el módulo en cuestion;
- **resistencia contra ataques:** la resistencia a ataques de un sistema de marcas de agua es un aspecto que puede relacionarse con la seguridad del mismo; según la aplicación de que se trate, unos ataques serán más importantes que otros; en general, una técnica de *watermarking* para módulos reutilizables deberá asegurar:
  - ***la prevención de la introducción de otra marca de agua***: uno de los desafíos más importantes de los esquemas de *watermaking* es la autenticidad de la marca de agua; es necesario proteger los diseños de los intentos de terceras partes que traten de introducir otra marca de agua en el diseño o, al menos, poder anular la autenticidad de esta segunda marca de agua ante un tribunal;
  - ***la dificultad para la localización***: la marca de agua debe ser perceptualmente invisible; es importante tener en cuenta que si la marca de agua es difícil de localizar, se necesitará especial conocimiento para poder llegar a detectarla y por tanto eliminarla;
  - ***la dificultad para la eliminación***: la marca de agua debe ser resistente contra los ataques que traten de eliminarla, al menos sin degradar de forma importante el diseño original; el esfuerzo necesario para eliminar la marca de agua del bloque reutilizable debe ser mayor que el necesario para desarrollar un nuevo bloque; en caso de que la marca de agua se pudiera llegar a eliminar, se debería ver afectado el correcto funcionamiento del módulo; además, si una parte de la marca de agua es eliminada, la información que identifica el autor no debería ser afectada.
- **extracción de la firma**: en caso de ser necesario, la marca de agua debe extraída fácilmente para identificar así un determinado bloque IP; además, el coste que

suponga el proceso de extracción debe ser el menor posible y es preferible que con este proceso no se revele la ubicación de la marca de agua.

Es importante tener en cuenta que algunos de los objetivos expuestos son mutuamente contradictorios. Por ejemplo, si la marca de agua que se quiere introducir en el diseño es más grande, se puede llegar a conseguir una mayor prueba de autoría y al mismo tiempo, será más difícil eliminar dicha marca de agua pero, probablemente, esto hará que se obtengan mayores modificaciones en los recursos del diseño. Por lo tanto, en el desarrollo y aplicación de una determinada técnica de *watermarking* es muy importante establecer los principales objetivos a conseguir. Al mismo tiempo, los objetivos marcados pueden servir como criterios para la evaluación del sistema de *watermarking* resultante. En el capitulo 5 se realiza un estudio más completo de la seguridad de sistemas de *watermarking*, tratando el caso particular de cada forma de ataque posible contra estos módulos IP.

3.6.3. Clasificación y revisión de las técnicas de *watermarking*

En la literatura se pueden encontrar distintas técnicas de *watermarking* para módulos reutilizables. Teniendo en cuenta los tipos de módulos IP, una posible clasificación de las técnicas de *watermarking* que han sido propuestas para la protección de módulos reutilizables se realiza atendiendo al nivel de abstracción en el que han sido aplicadas. Se pueden distinguir así técnicas de *watermarking* en el nivel físico, en el nivel de síntesis lógica y en un nivel alto de descripción.

3.6.3.1. *Watermarking* en el nivel físico

Varios autores proponen técnicas de *watermarking* para módulos IP basadas en restricciones durante el diseño y la implementación del mismo para insertar la marca de agua que identifica de forma única al autor. Estas técnicas se han considerado como algoritmos genéricos que pueden utilizarse en diferentes niveles del flujo de diseño, aunque el nivel físico ha sido en el que se han realizado más investigaciones. Se basan en la utilización de las herramientas disponibles para resolver problemas NP-*hard*

(polinómico no determinista, *Non-Deterministic Polynomial-time*)[*] pero añadiendo además restricciones adicionales para crear el nuevo diseño que contiene la marca de agua. Este método se compone de las siguientes partes: un problema de optimización, que consiste en un problema NP-*hard* y que para resolverlo necesita restricciones y heurísiticas; un algoritmo o *software* de optimización que resuelva el problema; un conjunto de restricciones que han de ser aplicadas al diseño; un procedimiento adecuado para añadir restricciones adicionales a las anteriores que ayuden a crear el diseño firmado. Esta última es la principal herramienta del método y está compuesta por funciones de encriptación unidireccionales que convierten la marca de agua del código en un conjunto de restricciones bien definidas. En primer lugar se encripta la marca de agua y se transforma mediante una función *hash*. Tras este procesamiento la marca de agua transformada se convierte en un conjunto de restricciones adicionales que se añaden a las que ya existían. El diseño y el conujunto de restricciones alimentan un optimizador que genera un diseño que contiene la marca de agua. De esta forma, la marca de agua es un conjunto de restricciones adicionales que limita el conjunto de posibles soluciones.

Kahng y compañía [KAH98a], [KAH98b] y [KAH01] ilustraron este método usando un sencillo problema de satisfacibilidad (SAT). Por ejemplo, supóngase que SAT (U, C) es un conjunto finito de variables U y una colección $C = (c_1, c_2, ..., c_n)$ de clausulas sobre U. El problema SAT consiste en encontrar el conjunto de todas las asignaciones de C que satisfacen todas las clausulas de U. Añadiendo restricciones adicionales a este problema se llega a una solución que identifica de forma única la mejor opción para introducir la marca de agua en el diseño. [KAH01] recoge la evolución de las investigaciones realizadas y muestra varios ejemplos de *watermarking* para módulos IP en dominios muy diferentes, entre ellos el nivel físico, de diseños implementados sobre FPGAs y sobre celdas estándar.

La técnica basada en el uso de restricciones también ha sido utilizada por otros autores para la introducción de huellas digitales dentro de módulos IP. En [LAC98a] se

[*] En teoría de la complejidad computacional, la clase de complejidad NP-*hard* se traduce como NP-completo, o NP-difícil.

presenta una técnica de *watermarking* para diseños implementados sobre FPGAs que consiste en utilizar técnicas criptográficas para seleccionar un subconjunto de restricciones de diseño físicas de un conjunto de restricciones que no han sido usadas por las especificaciones del diseño. La marca de agua se transforma en un vector de bits que se introducen dentro de la FPGA, utilizando algunos de los CLBs no usados. Cada tabla de consulta no utilizada de estos CLBs puede almacenar 16 bits de la firma. Tras la introducción de todos los bits de la firma, los CLBs utilizados para esta técnica de *watermarking* son enrutados en el diseño de forma que el funcionamiento del diseño original no se vea afectado. Además en este trabajo la firma se modifica antes de ser introducida en el diseño para convertirla en una secuencia de bits que reproduzca las propiedades estadísticas del diseño, reduciendo así la la posibilidad de detección por atacantes. En la preparación de la firma digital también se utiliza una codificación que permite la corrección de errores y se intercalan múltiples bloques ECC. Esta técnica se evaluó para tres diseños implementados sobre FPGAs y para firmas digitales de diferente longitud, desde tamaños muy pequeños hasta el máximo tamaño posible dada la disponibilidad de bits en las LUTs de los CLBs no utilizados. Aplicar directamente esta técnica de *watermarking* para crear diseños con una marca de agua distinta para cada usuario autorizado (huella digital) hace que la técnica sea susceptible a los ataques de colisión (que se estudian con más detalle en el capítulo 6), que consisten en comparar los aechivos de configuración de dos de los diseños anteriores para comprobar las diferencias entre ambos, que se deben a las marcas de agua. Eliminando estas diferencias de cualquiera de los diseños se puede llegar a obtener un circuito que funcione correctamente y que no contenga la marca de agua. Otro punto débil de este método es el impacto sobre el área y la posibilidad de detectar los CLBs que contienen la firma debido a su inactividad durante un funcionamiento normal del diseño.

La vulnerabilidad que presenta el método anterior a los ataques de colisión ha sido tratada posteriormente en otros trabajos [LAC98b], [LAC98c] y [LAC01]. En estos trabajos la preparación de la firma digital es muy similar a la realizada en [LAC98a] y en ellos se utilizan técnicas de *watermarking* basadas en la introducción de la firma mediante división en baldosas y configuración de CLBs. Con un pequeño esfuerzo adicional se pueden generar múltiples diseños físicos con diferentes configuraciones de bloques lógicos. En este caso, las diferencias fundamentales entre el circuito original y el que incluye la firma se producen en la fase de emplazamiento y enrutamiento

(*place&route*). No está claro cómo pueden afectar estas restricciones al rendimiento del sistema, especialmente en el caso de circuitos pequeños, donde las posibilidades de modificación de configuraciones son menores. Al igual que el método presentado en [LAC98a], todos los trabajos anteriores ofrecen técnicas de *watermarking* para diseños que son implementados en FPGAs.

Otra técnica de *watermarking* utilizada en este nivel de diseño está basada en la introducción de marcas de agua a través de restricciones de tiempo, consideradas también en [KAH01], y consiste en especificar restricciones de tiempo en rutas con diferentes caminos críticos y, por tanto, la comprobación de la firma se hace a través del fichero de configuración de la FPGA. Estas técnicas de *watermarking* presentan el inconveniente de que el módulo IP utilizado precisará una FPGA completa y, además, es vulnerable a la modificación de bits de configuración que el atacante sepa que no alterarán el funcionamiento del diseño. En [JAI03] también se utiliza una técnica de watermarking basada en restricciones de tiempo. En este trabajo la aplicación de esta técnica está restringida a diseños implementados sobre FPGAs y, aunque los resultados experimentales muestran que no hay penalización en área ni en velocidad, la firma digital utilizada en todos los diseños es de 16 bits, obtenida con código ASCII a partir de la marca de agua ”MD”, no haciendo uso de herramientas criptográficas o más sofisticadas para crear una firma segura.

Todas estas técnicas, además de las propuestas en [CAL99], [NAR01] o [NEW02] introducen marcas de agua en el nivel físico. Este nivel en el diseño de un sistema digital no debería ser el dominio exclusivo para la protección de la propiedad intelectual, puesto que en ese caso sólo se estaría protegiendo este nivel, al ser el último en el flujo de diseño. Además, al aplicar cualquier técnica de *watermarking* en este nivel de diseño se corre el riesgo de no llegar a conseguir que la firma se introduzca como una parte funcional del diseño, siendo éste un factor muy importante en *watermarking* para módulos IP para evitar distintos tipos de ataques.

3.6.3.2. *Watermarking* en el nivel de síntesis lógica

Aunque se han realizado muchas investigaciones en el nivel de diseño físico, se ha dirigido poco interés hacia la otros niveles de descripción. En [CUI06], y [KIR06] se

proponen técnicas de *watermarking* en el nivel de síntesis lógica combinacional. Estos dos métodos ofrecen opciones de diseño muy atractivas porque se pueden llegar a demostrar los derechos de autor en el nivel de síntesis lógica o en niveles de abstracción inferiores a éste. En [KIR06] se desarrollan protocolos para introducir información especifica de la herramienta o del diseñador en la red lógica mientras se realiza la minimización de la lógica y el mapeo. Se ha aplicado el método al mapeado sobre FPGAs basadas en LUTs utilizando un cojunto de diseños de prueba. El trabajo incluye un estudio de la resistencia contra ataques, en concreto un método estadístico para calcular un parámetro que ayude a determinar la prueba de autoría. Se utiliza una función hash y encriptación RSA para la firma digital, pero en los resultados experimentales no se deja claro la longitud de dicha firma. Por otro lado el trabajo presentado en [CUI06] está basado en la idea de utilizar los períodos de poca actividad de un diseño sintetizado para identificar algunas redes locales en las que introducir la marca de agua. Aunque los resultados experimentales muestran que la aplicación de este método tiene bajo impacto sobre el área y las prestaciones, la firma digital utilizada es sólo de 32 bits, en la que no se ha utilizado una función hash para prevenir algunos de los ataques contra los diseños protegidos mediante ténicas de *watermarking* y, no queda claro si el procedimiento descrito y los diseños utilizados como ejemplo permiten la introducción de firmas digitales más seguras.

3.6.3.3. *Watermarking* en descripciones de alto nivel

Las técnicas de *watermarking* en la descripciones de alto nivel se basan en trasladar al *hardware* técnicas de *watermarking* del *software*, aplicándolo a las representaciones de alto nivel, como diagramas de estados, descripciones VHDL, etc. La principal ventaja de utilizar técnicas de *watermarking* en un nivel alto de diseño reside en la mayor facilidad de añadir la marca de agua como parte funcional del diseño, siendo así más difícil de eliminar. Algunas de las referencias más antiguas que utilizan técnicas de *watermarking* en este nivel de diseño y que ofrecían resultados prometedores son [CHA98], [HON97], [HON99] y [KAH98b]. En general, en estos trabajos se utiliza superimposición de restricciones adicionales junto con las especificadas por el usuario.

Por otro lado, los trabajos [OLI01] y [TOR00] presentan la introducción de técnicas de *watermarking* aplicadas a partes secuenciales del diseño. Los dos algoritmos están basados en la adición de nuevas secuencias de entrada/salida en la representación de la máquina de estados finita (FSM) del diseño. La principal ventaja de ambos métodos es la facilidad de detectar la presencia de la marca de agua en todos los niveles de diseño inferiores. En [TOR00] se introduce el primer método de protección a través de FSMs. El algoritmo se basa principalmente en la extracción de las transiciones que no se utilizan el diagrama de estados de la máquina. Las transiciones no utilizadas se introducen en el diagrama con una nueva secuencia de entrada/salida asociada, que corresponde a la marca de agua. El método se aplica en un nivel alto del flujo de diseño, lo que proporciona mayor fortaleza, de forma que la seguridad del diseño no depende de que el algoritmo de *watermarking* se guarde en secreto. La marca de agua se puede detectar en niveles de diseño inferiores, algunas veces hasta después de la manufacturación. Los autores utilizan la probabilidad de coincidencia como la única medida de robustez, que sólo cubre el caso de falsos positivos. El estudio de la fortaleza de este algoritmo realizado en [ABD04] a través de un conjunto de probabilidades, muestra que el algoritmo presenta una alta vulnerabilidad a los ataques de enmascaramiento (no eliminan la marca de agua por completo). Por otro lado, la longitud de la firma digital introducida en los ejemplos de aplicación de este método no es fija, depende de las posibilidades que ofrezcan los diseños. Como era de esperar, para diseños que ocupan poca área hay un gran impacto sobre los recursos, que llega a superar el 100%, mientras que para diseños que ocupan mayor área este impacto es despreciable.

Otra de las técnicas de *watermarking* propuestas en este nivel de abtracción es [OLI01]; se aplica a partir de la descripción del diseño mediante FSMs y utiliza la manipulación del diagrama de estados para introducir la marca de agua. La preparación de la firma requiere una cadena de caracteres arbitraria de gran tamaño que defina claramente al autor y que posteriormente es encriptada con una clave pública y convertida en una firma compacta de 128 bits de longitud mediante la función *hash* MD5. La firma obtenida se descompone en combinaciones de secuencias de entrada; por ejemplo, si el diseño tiene 16 entradas y la firma es de 128 bits, se define una única secuencia de 8 combinaciones de entrada. Para la introducción de la firma es necesario modificar el diagrama de estados de tal forma que la secuencia de estados alcanzada con

la secuencia de entradas que define la firma muestre una propiedad específica, poco frecuente en el diagrama de estados original. Esta propiedad es puramente topológica y no depende de una codificación específica. Cuando se quiera extraer la firma, el diseñador debe proporcionar esta secuencia de entrada y la propiedad definida. Para realizar lo anterior se añaden estados y transiciones adicionales al diagrama de estados de una forma sistemática para satisfacer la propiedad buscada. Este algortimo tiene un impacto pequeño en el flujo de diseño, porque no es necesario examinar la FSM para encontrar las transiciones no utilizadas. Incluso se propone un método para introducir la marca de agua sin la necesidad de construir la FSM del diseño. Una posible desventaja del método es que los estados adicionales que se añaden se podrían eliminar usando algún método de reducción de estados. Los autores proponen como solución a lo anterior modificar ligeramente la funcionailidad del diagrama de estados. Sin embargo esto es difícil de ralizar automáticamente y puede llegar a afectar la funcionalidad del dsieño. Por otro lado, en el estudio realizado sobre la fortaleza del método propuesto no se muestra cómo ha sido calculada la probabilidad de falsos positivos.

En [FAN03] se propone una técnica para la protección de módulos IP que usa secuencias de test aleatorias. Después de integrar el módulo IP en un SoC, se tienen que crear las señales de test. Teniendo en cuenta lo anterior, el método propuesto en este trabajo combina la secuencia de test con el circuito generador de la marca de agua. Esto se hace integrando un circuito generador de la marca de agua en el chip del módulo de test, de forma que cuando el diseño se pone en el modo de test, la marca de agua se generará automáticamente. Se proponen diferentes formas de integrar el circuito de la marca de agua en el circuito de test, o la marca de agua completa puede generarse directamente al principio o al final. De acuerdo con la información de la marca de agua, el proveedor de módulo IP es capaz de verificar los derechos de autor y no necesitan realizar un análisis más exhaustivo. El método presenta algunos inconvenientes, por ejemplo, la primera forma de ataque sobre el diseño protegido que se podría intentar consiste en eliminar el circuito de test y añadir otro nuevo, lo que no afectaría las prestaciones del diseño. Lo anterior obliga a que para mantener la seguridad de la marca de agua es necesario mantener en secreto el algoritmo utilizado. El cálculo de probabilidades de ataque con éxito no es posible, porque el diseño no se modifica y la eliminación de la marca de agua es simple y directa.

De forma general, la introducción de la marca de agua en este nivel de diseño proporciona el esquema más resistente contra ataques puesto que ésta se introduce en etapas preliminares, y es arrastrada a través de todo el diseño, estando presente en todos los niveles posteriores. Además, en esta fase del diseño, la marca de agua se podría llegar a introducir como una parte funcional del circuito. Las principales desventajas están relacionadas con las restricciones introducidas en el proceso de diseño y con un mayor incremento de área del circuito resultante. Las ventajas que pueden llegar a ofrecer la introducción de marcas de agua en el nivel alto de descripción del diseño hace que sea muy interesante el desarrollo de nuevas técnicas de *watermarking* que consigan además superar los inconvenientes que presentan algunas de los trabajos previos realizados en este nivel de abstracción.

3.6.3.4. *Watermarking* en distintos niveles de abstracción

Finalmente, la aplicación de marcas de agua en distintos niveles de abstracción [CHA98, RAS99] es probablemente una de las técnicas de *watermarking* más robustas. En [CHA98] se introduce un esquema de *watermarking* jerárquico basado en un método genérico que puede utilizarse en diferentes niveles de abstracción. Los autores proponen la utilización de múltiples marcas de agua en diferentes niveles de abstracción para conseguir un sistema más seguro. El esquema aumenta la robustez de la marca de agua ya que el atacante debe eliminar la marca de agua en los diferenetes nivele de abstracción. Por otro lado, este esquema no debería suponer un incremento muy elevado en el ciclo de diseño ni en el área o prestaciones, puesto que la información de la marca de agua se puede dividir en varios niveles. Así, la resistencia contra ataques de *tampering* y de rediseño se consigue generando una marca de agua que se distribuye por varios niveles de abstracción. Sin embargo, esta esquema de *watermarking* puede resultar muy costoso y complejo.

Como se acaba de ver, los esquemas de protección se pueden aplicar en cualquier nivel del diseño y propagarse a niveles posteriores. Por otro lado, si la marca de agua que se ha introducido no es funcional, es decir, no se incluye en la parte funcional del diseño, la protección que se ha conseguido podría llegar a ser eliminada por ingeniería inversa. De este modo, si se pretende conseguir una protección difícil de eliminar, no es suficiente introducir y esconder la marca de agua, siendo además necesario encontrar la

forma y mecanismos eficientes para hacer que la marca de agua sea parte funcional del circuito. Esto se puede conseguir más fácilmente con las técnicas de *watermarking* en el nivel de descripción más alto. Al mismo tiempo que se está realizando la descripción HDL del diseño, se podría estudiar la metodología adecuada para introducir una marca de agua de forma que ésta quede integrada dentro del mismo diseño. Además, como ya se ha comentado, estas ténicas introducen marcas de agua que estarán presentes en todos los niveles de diseño posteriores. Por lo tanto, el desarrollo de una estrategia de *watermarking* en un nivel alto de descripción consituye una opción muy interesante para la protección de módulos IP. Además de conseguir que la marca de agua sea parte funcional del diseño y que no afecte a su correcto funcionamiento, se debe conseguir superar algunos de los inconvenientes que presentan otras técnicas de *watermarking* de este tipo, que, como ya se ha comentado, principalmente son las restricciones introducidas en el diseño, el aumento del área y la reducción de las prestaciones.

3.7. Conclusión

En este capítulo se han analizado los aspectos más importantes relacionados con las metodologías de diseño basadas en la reutilización de módulos, que ofrecen un aumento en la productividad y una reducción importante en los tiempos de desarrollo. En concreto, la reutilización de módulos juega un papel muy importante en el diseño de SoCs. Sin embargo, los componentes involucrados en estas metodologías de diseño son susceptibles a infracciones y abusos. La protección de los derechos de la propiedad intelectual de módulos reutilizables es un campo multidisciplinar que está íntimamente relacionado con los aspectos del proceso de diseño de un CI. Los procedimientos habituales para la protección de propiedad intelectual en módulos reutilizables consisten en la ocultación de algún tipo de firma mediante técnicas de *watermarking*. Algunos de los criterios de evaluación de una determinada técnica de watermarking son el correcto funcionamiento, el coste, la robustez, la seguridad y prueba de autoría. Por último, se han revisado las distintas técnicas de *watermarking* propuestas en la bibliografía, clasificadas de acuerdo con el nivel de abstracción del diseño en el que han sido aplicadas. Aunque la mayoría de las investigaciones se han centrado en el diseño físico,

las ventajas que ofrece la introducción de marcas de agua en un nivel alto de descripción han hecho que recientemente se haya más interés en el desarrollo de nuevas técnicas de *watermarking* en este nivel de abstracción.

Capítulo 4: Nueva técnica de *watermarking* para módulos IP

En este capítulo se presenta un nuevo procedimiento para la protección de la propiedad intelectual de circuitos digitales (implementados sobre FPGAs o ASICs). El objetivo en el desarrollo del procedimiento es la protección de los derechos de los autores de módulos reutilizables mediante la introducción de una marca de agua. A lo largo de este capítulo se realiza una descripción completa del método de protección propuesto, desde las ideas preliminares que llevaron al desarrollo de la primera de las estrategias de protección hasta los últimos avances conseguidos.

4.1. Introducción

Como ya se comentó en el capítulo anterior, en la literatura se pueden encontrar distintas técnicas de *watermarking* para módulos IP. La mayor parte de estas técnicas introducen marcas de agua en el nivel físico; sus desventajas más importantes son la dificultad de añadir la marca de agua como parte funcional del diseño y que sólo se protegería el nivel físico. Es por este motivo que surge la necesidad de explorar el desarrollo y la aplicación de técnicas de *watermarking* en otros niveles de descripción del diseño más altos. Entre las posibilidades existentes, se encuentran las técnicas de *watermarking* en el nivel de descripción más alto [CHA03, HON99], cuyas principales ventajas son: la dificultad de eliminar la marca de agua, puesto que ésta se introduce en etapas preliminares y es además arrastrada a través de todo el proceso de diseño estando

presente en todos los niveles posteriores, y la facilidad para poder introducir esta marca de agua como una parte funcional del diseño.

En este trabajo de tesis se ha abordado el estudio y desarrollo de mecanismos de protección de la propiedad intelectual de módulos IP a través de técnicas de *watermarking* en descripciones de alto nivel. A partir de la necesidad de *watermarking* en un nivel elevado de descripción del diseño y la experiencia previa del grupo de trabajo en aritmética de residuos y de índices, surge la idea de usar posiciones de memoria libres para introducir los bits de una firma digital al mismo tiempo que se realiza la descripción del diseño. Como ya se ha comentado anteriormente, los circuitos basados en el RNS que se implementan sobre FPGAs suelen hacer un uso extensivo de las tablas incluidas en los dispositivos programables [MEY01]. Para la mayor parte de estas tablas de consulta quedan patrones de entrada sin asignar, de modo que la idea de introducir los bits de una firma digital dentro de posiciones de memoria vacías es directamente aplicable en circuitos basados en aritmética de residuos [PAR04].

Por otro lado, existe también la necesidad de extender y aplicar las ideas desarrolladas para diseños basados en el RNS a sistemas digitales genéricos y permitir así la protección de la propiedad intelectual de los mismos, tanto para su implementación para tecnologías programables como para ASICs. Sin embargo, para sistemas digitales genéricos no se puede partir del hecho de que haya posiciones de memoria libres, porque en este caso la técnica desarrollada perdería generalidad. De esta forma, es necesario buscar otra alternativa para la difusión de los bits de la firma digital, por ejemplo, identificar grupos de bits de esta firma con partes del diseño necesarias para el correcto funcionamiento del mismo [CAS07a].

Otro aspecto muy importante a tener en cuenta en la protección de módulos reutilizables está relacionado con la detección de la firma digital introducida. Esta firma digital debe ser fácil de extraer para identificar al autor de un determinado bloque IP y poder así reclamar los derechos correspondientes sobre dicho bloque. Además, el coste que suponga el proceso de extracción debe ser el menor posible y esta extracción ha de no revelar la ubicación de los bits de la firma digital. Por lo tanto, será necesario el desarrollo de un mecanismo que permita la extracción de la firma digital y que, en la medida de lo posible, cumpla los requisitos anteriores [CAS07a].

El resto de la memoria describe con más detalle estas ideas para la difusión de los bits de una firma digital dentro de módulos reutilizables, las estrategias de *watermarking* desarrolladas a partir de estas ideas y los resultados obtenidos.

4.2. Estructura general

La técnica de *watermarking* para bloques IP que se propone pretende que, a partir de la descripción de alto nivel del diseño, se genere un sistema que incluya una firma digital y la lógica necesaria para extraerla con la menor penalización en área y velocidad y una alta invulnerabilidad frente a distintos tipos de ataques.

En la aplicación de cualquier técnica de protección para módulos IP se debe seguir una serie de pasos. En concreto, para el método de *watermarking* propuesto en este trabajo, los procesos que intervienen en la generación del diseño firmado son: la preparación de la firma digital, la difusión de los bits de la firma y la preparación del diseño para la extracción de la firma. Por otro lado, el proceso de validación de la firma digital requiere la extracción de la misma aplicando al diseño firmado el mecanismo o procedimiento desarrollado para ello, y la comparación de esta firma extraída con la firma digital del autor que reclama los derechos de propiedad sobre el diseño. La figura 4.1 muestra un esquema de los procesos que intervienen en la obtención del diseño firmado y la validación de la firma.

El proceso de preparación de la firma convierte la marca de agua o firma elegida para identificar el diseño (por ejemplo, un documento de texto o una imagen), que será guardada en un documento de dominio público, en una secuencia de bits denominada firma digital o electrónica. En esencia, la firma digital obtenida es el resultado de aplicar a la marca de agua elegida para el diseño, un algoritmo matemático, denominado función *hash*, que la convierte en una secuencia fija de bits. La función *hash* debe satisfacer dos importantes requisitos: debe ser difícil encontrar dos marcas de agua cuyo valor para la función *hash* sea idéntico, y, dado uno de estos valores, debe ser difícil recuperar la marca de agua que lo produjo. Además de la función *hash* se pueden utilizar algoritmos de cifrado (en los que se emplea una clave privada) y algoritmos de

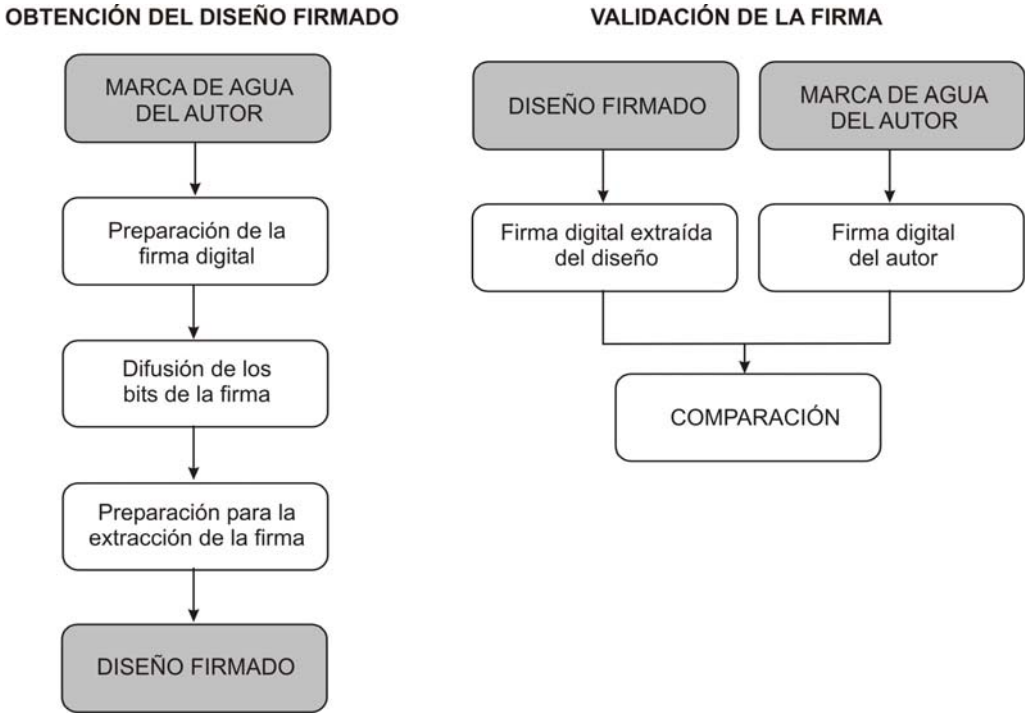

Figura 4.1. Procesos para la aplicación del método de protección.

corrección de errores. La aplicación de la función *hash* y del algoritmo de cifrado aseguran la integridad de la marca de agua, y el algoritmo de corrección de errores permite recuperar la firma digital en el caso de que algunos de los bits de esta firma hayan sido modificados.

El proceso de difusión de los bits de la firma consiste en "difundir" los bits de la misma a través de las estructuras de memoria o de la lógica combinacional que se incluyen en la descripción de alto nivel del diseño a proteger. Dependiendo de cómo se realice esta difusión, se distinguirá entre una de las dos estrategias de protección que se detallarán más adelante, "Estrategia de Protección basada en la Introducción de los Bits de la Firma en LUTs", EPIBF-LUT, o "Estrategia de Protección basada en la Búsqueda de los Bits de la Firma en la Lógica Combinacional", EPBBF-LC. Independientemente de la estrategia de protección aplicada, la firma se propaga a través de todo el flujo de diseño, llegando hasta la implementación física, y de forma totalmente independiente de la tecnología de implementación (ASIC, FPGA, etc.).

Por otro lado, la nueva técnica de protección proporciona un procedimiento o método para una fácil y segura extracción de la firma. Este procedimiento es no destructivo, es decir, se puede extraer dicha firma sin que ello suponga pérdida total o parcial alguna de los bits de la firma. El proceso de preparación del diseño para la extracción de la firma sólo necesita la inclusión de una lógica de extracción dentro del diseño original. Esta lógica adicional, tras detectar una petición de extracción de la firma, generará las señales de control necesarias e irá mostrando la firma como una secuencia de datos de salida del sistema. Tras la difusión de los bits de la firma y la preparación del diseño para su extracción ya se obtendrá el diseño protegido, que podría ser distribuido como módulo reutilizable. Finalmente, para ofrecer seguridad al código HDL desarrollado, se pueden utilizar herramientas de encriptación [OBR94].

Al igual que cualquier módulo IP, los protegidos mediante esta técnica se distribuyen para su uso empleando acuerdos de licencia. En caso de que sea necesario demostrar los derechos de autor sobre un determinado módulo se utilizaría el proceso de validación. Para ello se tiene que extraer la firma digital del diseño mediante el proceso de extracción. Por otro lado, se tiene que obtener la firma digital a partir del documento de dominio público que guarda la marca de agua correspondiente al individuo que reclama los derechos de autor sobre el módulo. Tras comparar ambas firmas, la extraída del diseño y la que ha sido generada a partir del documento de dominio público, su coincidencia demostraría los derechos del autor sobre el módulo reutilizable.

En los siguientes apartados se describen con más detalle cada uno de los procesos necesarios en la aplicación del método de protección propuesto, a la vez que se trata de reflejar la evolución que ha experimentado este método de protección desde sus ideas preliminares hasta los últimos avances conseguidos.

4.3. Diseño firmado

Tras haber expuesto en la sección anterior la estructura general del nuevo método de protección, a continuación se van a describir en detalle los procesos de preparación de la firma digital, difusión de los bits de la firma y preparación del diseño para la

detección de la firma. Tras seguir todos estos pasos, se obtendría el diseño firmado, que podrá ser distribuido y utilizado de la misma forma que el diseño original.

4.3.1. Preparación de la firma

Dentro del método de protección propuesto, la preparación de la firma es el proceso que transforma una determinada marca de agua o firma en una secuencia de bits o firma digital que más tarde será incluida dentro de un determinado diseño. En primer lugar, se debe elegir la marca de agua o firma con la que se pretende identificar el diseño. La información que contiene la marca de agua o firma puede ser el nombre del autor, el nombre de la compañía u otra información que se encuentre directamente relacionada con el propietario y/o usuarios autorizados. Si la marca de agua contiene información sobre el usuario autorizado, además del uso ilegal o no autorizado del módulo protegido, se podría llegar a conocer el origen de la apropiación indebida del módulo. Esta información quedará almacenada en un documento de dominio público, como un documento de texto, una imagen, un fichero de audio, etc. Antes de detallar el procesamiento que debería aplicarse a esta marca de agua para la generación de la firma digital, se van a considerar algunos aspectos que ayudarán a determinar los algoritmos implicados en este procesamiento.

La seguridad de una firma digital es un aspecto muy importante. Para que una firma digital sea segura, debe cumplir algunas características:

- que sea de longitud fija, independientemente de la marca de agua elegida;
- que a partir de la marca de agua, sea fácil calcular la firma digital;
- que partir de la firma digital sea computacionalmente imposible recuperar la marca de agua;
- que a partir de la marca de agua sea computacionalmente imposible encontrar otro documento de dominio público que genere la misma firma digital.

Otro aspecto directamente relacionado con la seguridad de una firma digital es la resistencia contra ataques. El ataque más conocido a una firma digital es el “ataque de

cumpleaños"[*] y se refiere a una clase de ataques de fuerza bruta. La paradoja del cumpleaños [SCH96] demuestra que aunque dado un mensaje x resulta muy difícil calcular un y tal que $firma(x)=firma(y)$, es considerablemente menos costoso generar muchos valores aleatoriamente y, posteriormente, buscar entre ellos una pareja cualquiera (x, y) tal que $firma(x)=firma(y)$. Por ejemplo, en el caso de una firma digital de 64 bits obtenida a partir de un determinado mensaje, necesitaríamos 2^{64} mensajes para obtener una firma digital que coincida con la primera. Sin embargo, bastaría con generar aproximadamente 2^{32} mensajes aleatorios para que aparezcan dos que generen la misma firma digital. En general, si la primera cantidad es muy grande, la segunda cantidad es aproximadamente su raíz cuadrada. Suponiendo que una computadora genera un millón de mensajes por segundo, el primer ataque nos llevaría aproximadamente 600.000 años mientras que el segundo apenas necesitaría una hora.

Teniendo en cuenta esta forma de ataque sobre una firma digital, es conveniente añadir una característica más a la lista anterior:

- debe ser difícil encontrar dos documentos de dominio público aleatorios tales que sus firmas digitales sean iguales.

Por este motivo se recomienda emplear firmas digitales de al menos 128 bits, siendo 160 bits el valor más usado.

La aplicación de una función criptográfica, denominada función *hash*, a la marca de agua ayuda a conseguir en gran medida las características que debería cumplir una firma digital. Una función criptográfica transforma la marca de agua en una secuencia fija de bits y hace que sea extremadamente difícil conocer la marca a partir de la secuencia de bits obtenida, es decir, realizar la función inversa.

---

[*] o *birthday attack*. Es un tipo de ataque criptográfico que se basa en la matemática detrás de la paradoja del cumpleaños, haciendo uso de una situación de compromiso espacio-tiempo informática. La paradoja del cumpleaños establece que si hay 23 personas reunidas, hay una probabilidad del 50,7% de que al menos dos de ellas cumplan años el mismo día. Para 60 o más personas la probabilidad es mayor del 99%. En sentido estricto esto no es una paradoja ya que no es una contradicción lógica; es una paradoja en el sentido que es una verdad matemática que contradice la común intuición. Mucha gente piensa que la probabilidad es mucho más baja, y que hacen falta muchas más personas para que se alcance la probabilidad del 50%.

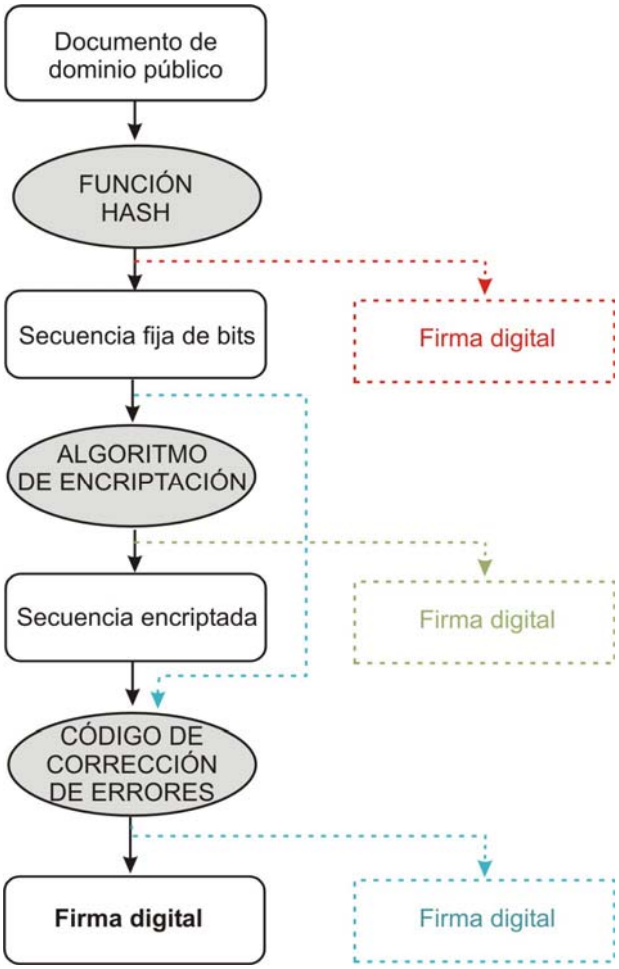

Figura 4.2. Obtención de una firma digital.

La secuencia de bits obtenida podría ser la firma digital con la que se firme el diseño, como muestra la figura 4.2. Sin embargo, se pueden aplicar algunos algoritmos más que ayuden a crear una firma digital más segura; por ejemplo, un proceso de encriptación sobre la secuencia de bits a través de un algoritmo de clave privada/pública, como el RSA [RIV78]. Aunque en este trabajo se ha considerado una definición menos restringida para la firma digital, para algunos autores la definición de firma digital corresponde estrictamente a la aplicación de una función *hash* y de un algoritmo de encriptación. El proceso de encriptación cifraría la secuencia *hash* obtenida mediante el uso de la clave privada del firmante o autor del diseño a proteger, de modo que cualquiera pueda comprobar la firma usando la clave pública correspondiente. Para tratar de conseguir que la firma digital resultante (obtenida

mediante la función *hash* u obtenida mediante la función *hash* y la encriptación) sea resistente a ataques que traten de eliminar una parte de ésta, se pueden aplicar también códigos de corrección de errores (*Error Correction Codes*, ECC) [BLA84, LIN83, PUR95], como muestra la figura 4.2. La aplicación de un determinado código de corrección de errores permite restaurar un número de bits de la firma digital que hayan sido modificados.

Las firmas con las que se ha trabajado en el desarrollo de las estrategias de *watermarking* propuestas son, por un lado, firmas obtenidas mediante una función *hash*, en concreto MD5 y SHA1, y por otro estas mismas firmas pero con corrección de errores. Como se verá en sección 4.3.1.2, la utilización de una función *hash* y de un algoritmo de cifrado asimétrico aumenta la seguridad de la firma digital y del método de protección. Sin embargo, hasta el momento no se ha trabajado con algoritmos de encriptación para las firmas utilizadas en este trabajo, quedando pendiente para futuros trabajos la incorporación de estas firmas en el desarrollo de nuevos resultados. No obstante, también se hace un análisis más detallado de la obtención de firmas mediante la encriptación de la función *hash*. Por tanto, en las siguientes secciones se van a describir con más detalle las funciones *hash*, los algoritmos de encriptación y los códigos de corrección de errores que, como se ha visto, intervienen en la preparación de la firma digital.

4.3.1.1. Funciones *hash*

En general, las funciones *hash* son funciones de compresión, que dan como resultado bloques de longitud n a partir de bloques de longitud m ($m>n$). Estas funciones se encadenan de forma iterativa, haciendo que la entrada en la iteración i sea función del i–ésimo bloque del mensaje y de la salida del paso i–1, como muestra la figura 4.3. Se suele incluir en alguno de los bloques del mensaje m, al principio o al final, información sobre la longitud total del mensaje. De esta forma se reducen las probabilidades de que dos mensajes con diferentes longitudes den el mismo valor *hash*. Las funciones *hash* son una herramienta fundamental en la criptografía y se emplean fundamentalmente para resolver el problema de la integridad de los mensajes, así como su autenticidad y su origen. Algunas de las aplicaciones de las funciones *hash* [KAM98] son contraseñas, integridad o autentificación de mensajes y firmas digitales.

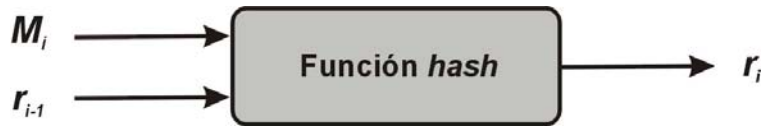

Figura 4.3. Estructura iterativa de la función *hash*.

Entre las propiedades más importantes de las funciones *hash* de una dirección se pueden destacar las siguientes:

- ***unidireccionalidad***: es imposible obtener, en un tiempo razonable, el mensaje a partir del valor *hash*, siendo únicamente posible calcular el *hash* a partir del mensaje. Así, dado un valor *hash h*, no es posible encontrar una entrada o mensaje *x* tal que $H(x)=h$.
- ***las funciones hash deben estar libres de colisiones***, es decir, dados dos mensajes distintos, *x* e *y,* no se puede obtener un mismo valor *hash* tal que $H(x)=H(y)$.

En el momento que alguna o ambas condiciones se incumple, se dice que la función *hash* ha sido rota. Se han desarrollado varias funciones tratando de mejorar las versiones anteriores, para tener una mayor seguridad y evitar que se lleven a cabo ataques con éxito. Las funciones *hash* más usadas actualmente son MD5 (*Message Digest* 5) y SHA (*Secure Hash Algorithm*) [SCH96], aunque existen otras como MD4, SHA256, SHA512 y SHA384.

MD5 es uno de los algoritmos de reducción criptográficos desarrollados por el profesor Ronald Rivest del MIT (*Massachusetts Institute of Technology*). Es uno de los algoritmos más populares de generación de firmas digitales y fue desarrollado en 1991 como reemplazo del algoritmo MD4 después de que Hans Dobbertin descubriese su debilidad. Produce como salida un valor *hash* de 128 bits [STA98], que se presentará típicamente como un número de 32 dígitos hexadecimales. A pesar de haber sido considerado criptográficamente seguro en un principio, ciertas investigaciones han revelado vulnerabilidades que hacen cuestionable el uso futuro del MD5. En agosto del 2004 (Xiaoyun Wang, Dengguo Feng, Xuejia Lai y Hongbo Yu). se anunció el

descubrimiento de colisiones de *hash* para MD5, con un ataque que se consumó en una hora de cálculo con un clúster IBM P690. Aunque dicho ataque era analítico, el tamaño del *hash* (128 bits) es lo suficientemente pequeño como para que resulte vulnerable frente a "ataques de cumpleaños", citados anteriormente. Debido al descubrimiento de métodos sencillos para generar colisiones de *hash*, muchos investigadores recomiendan su sustitución por algoritmos alternativos tales como SHA-1 o RIPEMD-160.

La familia SHA es un sistema de funciones *hash* criptográficas relacionadas de la Agencia de Seguridad Nacional de los Estados Unidos y publicadas por el NIST (*National Institute of Standards and Technology*). El primer miembro de la familia, publicado en 1993, es oficialmente llamado SHA. Sin embargo, hoy día, no oficialmente se le llama SHA-0 para evitar confusiones con sus sucesores. Dos años más tarde el primer sucesor de SHA fue publicado con el nombre de SHA-1. Existen cuatro variantes más que se han publicado desde entonces y cuyas diferencias se basan en un diseño modificado y rangos de salida incrementados: SHA-224, SHA-256, SHA-384, y SHA-512 (llamándose SHA-2 a todos ellos). Su diseño tiene mucha relación con MD5 pero con ciertas diferencias; por ejemplo genera una salida de 160 bits [STA98]. Es más lento que el MD5, pero al ser mayor la longitud de la clave, la resistencia contra ataques de colisión por fuerza bruta y de inversión también es mayor. SHA-1 se ha examinado muy de cerca por la comunidad criptográfica, y no se ha encontrado ningún ataque efectivo. No obstante, en el año 2004, se divulgó un número de ataques significativos sobre funciones criptográficas *hash* con una estructura similar a SHA-1, lo que ha planteado dudas sobre la seguridad a largo plazo de SHA-1. En la figura 4.4 se muestran ejemplos de aplicación de las funciones *hash* MD5 y SHA1; se obtienen secuencias de 128 y 160 bits, respectivamente.

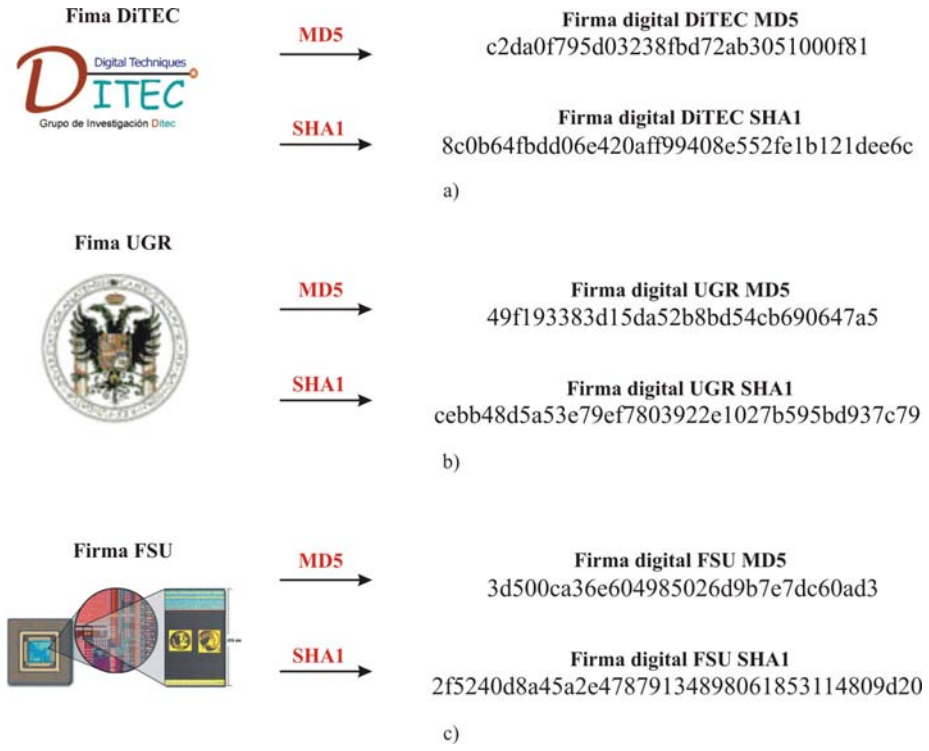

Figura 4.4. Ejemplos de obtención de firmas digitales mediante las funciones *hash* MD5 y SHA1: a) Firma DiTEC, b) firma UGR y c) firma FSU.

En la generación de la firma digital es importante el uso de una función *hash* por las dos propiedades citadas anteriormente. Por un lado, la función *hash* asegura que a partir de la secuencia de bits obtenida sea extremadamente difícil revertir el proceso y obtener la marca de agua correspondiente. En el caso de que la firma digital se obtenga mediante una función *hash* y esta firma se introduzca dentro de un diseño mediante un determinado procedimiento, el proceso de validación de la firma deberá extraer la firma digital del diseño, obtener la firma digital mediante la aplicación de esta función *hash* al documento de dominio público que contiene la marca de agua y comparar ambas firmas. Por otro lado, se evitan las colisiones, que, como ya se ha mencionado, hacen que no sea posible (computacionalmente intratable) que la aplicación de la misma función *hash* a dos marcas de agua puedan generar la misma secuencia de bits. Esto es muy importante para evitar algunos de los ataques; en concreto, asegura que la firma extraída del diseño corresponda a un único documento de dominio público, que se utilizó para generar la firma digital con la que se firmó el diseño.

4.3.1.2. Algoritmos asimétricos de cifrado

Los algoritmos de clave pública, o algoritmos asimétricos [RIV78], fueron introducidos por Whitfield Diffie y Martin Hellman a mediados de los años 70. Su novedad fundamental con respecto a la criptografía simétrica es que las claves no son únicas, sino que forman pares. Poseen dos claves diferentes, denominadas clave pública y clave privada. Una de ellas se emplea para codificar, mientras que la otra se usa para decodificar. Dependiendo de la aplicación del algoritmo, la clave pública será la de cifrado o viceversa. Por motivos de seguridad, es aconsejable que a partir de una de las claves resulte extremadamente difícil calcular la otra.

Una de las aplicaciones de los algoritmos asimétricos es la autentificación de mensajes, con la ayuda de funciones *hash*, que permiten obtener una firma digital a partir de un mensaje. Hasta la fecha han aparecido multitud de algoritmos asimétricos, la mayoría de los cuales son inseguros; otros son poco prácticos, bien sea porque el criptograma es considerablemente mayor que el mensaje original, bien porque la longitud de la clave es enorme. Se basan en general en plantear al atacante problemas matemáticos difíciles de resolver. El más popular por su sencillez es RSA, que ha sobrevivido a multitud de ataques, si bien necesita una longitud de clave considerable. Otros algoritmos son los de ElGamal, el de Rabin y el DSA (*Digital Signature Algorithm*).

El algoritmo de clave pública RSA (R. **R**ivest, A. **S**hamir y L. **A**delman) fue desarrollado en 1978 en el MIT. De entre todos los algoritmos asimétricos, quizá RSA sea el más sencillo de comprender e implementar. En muchos de los algoritmos simétricos, como el RSA, ambas claves sirven tanto para codificar como para decodificar. La seguridad de RSA radica en la dificultad de la factorización de números grandes. Las claves pública y privada se calculan a partir de un número que se obtiene como producto de dos primos grandes. Si el atacante quiere recuperar el mensaje a partir del criptograma y la clave pública, se ha de enfrentar a un problema de factorización. Por otro lado, el algoritmo DSA es una parte del estándar de firma digital DSS (*Digital Signature Standard*), propuesto por el NIST en 1991.

En cuanto a firmas digitales, las primeras versiones del proyecto PGP (*Pretty Good Privacy*) [ZIR08], iniciado a principios de los 90 por Phill Zimmerman, obtienen

una secuencia de bits mediante la función *hash* MD5, que posteriormente se codifica empleando la clave privada RSA correspondiente. Las versiones actuales implementan el algoritmo DSS, que emplea la función *hash* SHA-1 y el algoritmo asimétrico DSA.

Para los métodos de protección de firma digital en bloques IP, la utilización de una función *hash* y de un algoritmo de cifrado asimétrico aumenta la seguridad de la firma digital y del método de protección. Para la aplicación considerada en este trabajo, y teniendo en cuenta que la validación de la firma deberá ser realizada por un equipo oficial de validación, el autor del diseño utilizará una clave privada y una clave pública y el equipo de validación utilizará claves privada y pública, distintas de las anteriores. El cifrado se realizará con la clave privada del autor y la pública del equipo de validación, mientras que el descifrado empleará la clave privada del equipo de validación y la pública del autor. De esta forma, a partir de la marca de agua que se guarda en un documento de dominio público, se podría llegar a conocer la secuencia de bits que genera la función *hash*, pero no se conocería la firma digital que fue incluida en el diseño si no se dispone de la clave privada del autor. Se pueden evitar así ataques que traten de eliminar o modificar dentro del diseño firmado los bits de la firma digital, ya que no se puede llegar a conocer esta firma sin la clave privada del autor. De forma análoga, a partir de la firma extraída del diseño, cuya extracción resulta una tarea muy difícil si no conoce el mecanismo diseñado para ello, sólo el equipo de validación podría realizar el descifrado a través de su clave privada.

4.3.1.3. Códigos de corrección de errores

Con frecuencia los sistemas digitales están sometidos a posibles errores, principalmente durante la transmisión o el almacenamiento de la información, que puede alterar el valor de uno o varios bits. Los códigos de detección de errores permiten detectar estos posibles errores. Un método muy utilizado para la detección de errores en un solo bit consiste en añadir un bit de paridad, de manera que la secuencia de bits resultante con este bit extra tenga un número par (paridad par) o impar (paridad impar) de unos. Una vez detectado un error, habrá que desechar la secuencia correspondiente y tratar de volver a captar esa información de su lugar de origen. Pero esto no siempre es posible, como es el caso de una firma digital incluida en un diseño.

Los códigos de corrección de errores permiten recuperar los bits dañados a partir de la información restante. En el caso concreto de una firma digital, al introducir en la secuencia de bits de la firma las técnicas de corrección de errores, se pueden superar ataques que modifiquen o eliminen parte de ésta. El objetivo de estas técnicas de corrección de errores es introducir una redundancia controlada, de forma que sólo un subconjunto de todas las posibles combinaciones de secuencias de bits corresponda a secuencias válidas. Además, en el diseño de códigos correctores de errores con redundancia, es muy importante que la decodificación sea rápida.

El sistema más simple de corrección de errores se basa en la fuerza bruta, como los códigos de corrección 2 en 3. Este tipo de códigos se caracteriza por repetir la información 3 veces de modo que si se pierde uno de los datos, quedarán los otros dos para descubrir cual era el adecuado. Tiene el grave inconveniente de que triplica la cantidad de información, por lo que no es muy utilizado.

Existe otro sistema mucho más fiable, consistente en añadir unas palabras de control intercaladas entre los datos. Es un sistema de redundancia cíclica. Sin embargo, no se debe confundir con el CRC, pues el CRC es sólo un sistema de detección de errores, pero no permite su corrección a partir de los datos recibidos. Hay varias maneras de generar estas palabras de control; la más simple es tomar dos datos adyacentes e intercalar entre ellos su suma. De este modo, si se tienen n datos de m bits cada uno, se le añadirán $n-1$ datos de $m+1$ bits cada uno. Si se produce un error en una palabra, será fácil detectar cuál es y recuperar su valor original con una simple resta.

Un código binario simbolizado como (n, k, d) codifica k bits de información como un vector de n bits, donde se tienen $m=n-k$ bits de información redundante para poder llevar a cabo la corrección de errores y d es la distancia Hamming [PUR95]. Los códigos Hamming [BLA84] son la primera clase de códigos diseñados especialmente para la corrección de errores. Estos códigos y sus distintas versiones han sido empleados ampliamente para el control de errores en comunicaciones digitales y sistemas de almacenamiento de datos. Los límites del código muestran la capacidad máxima del código. Para un código de corrección de errores que permita corregir errores en t bits, los siguientes límites proporcionan una buena estimación:

$$2^{n-k} \geq \sum_{i=0}^{t} \binom{n}{i} \tag{4.1}$$

Por ejemplo, un código Hamming con distancia 3, capaz de corregir un error, tiene una estructura (2^m–1, 2^m–m, 3), donde m es el número de bits de paridad, 2^m–1=n es la longitud del código y 2^m–m=k es el número de bits de información. Cada bit de paridad establece paridad par entre él mismo y determinados bits de información.

4.3.2. Difusión de los bits de la firma

Tras obtener la firma digital que identifica el diseño, los siguientes pasos a seguir en la aplicación del método de protección propuesto, mostrados en la figura 4.5, son la difusión de la firma dentro del diseño y la preparación del diseño para la extracción de la firma. Esta sección se dedicará a estudiar en detalle el proceso de difusión de la firma y en la sección 4.3.3 se estudiará el proceso de preparación del diseño para la extracción de la firma.

En este trabajo de tesis se han analizado diferentes opciones para la difusión de los bits de la firma. Este análisis es fruto de la evolución que ha experimentado el método de protección desde las ideas preliminares. Como se detalla en la sección 4.1, las técnicas de protección se clasifican en dos grupos: estrategia basada en la introducción de los bits de la firma y estrategia basada en la búsqueda de los bits de la firma. En los siguientes apartados se describe cada una de estas estrategias, mostrando algunos ejemplos que ayuden a clarificar la forma en la que estas estrategias se pueden aplicar a un determinado diseño, y se comentan las principales ventajas que ofrecen comparadas con otras técnicas de protección.

4.3.2.1. Estrategia basada en la introducción de los bits de la firma

En esta estrategia de protección la difusión de la firma se realiza introduciendo los bits, en concreto bloques de bits de la firma (BBFs), en celdas no utilizadas en estructuras de memoria [PAR04, CAS05], teniendo en cuenta que estas estructuras de memoria están incluidas y descritas en el código HDL del sistema que se intenta proteger. El hecho de que las estructuras de memoria que se utilicen para la difusión de

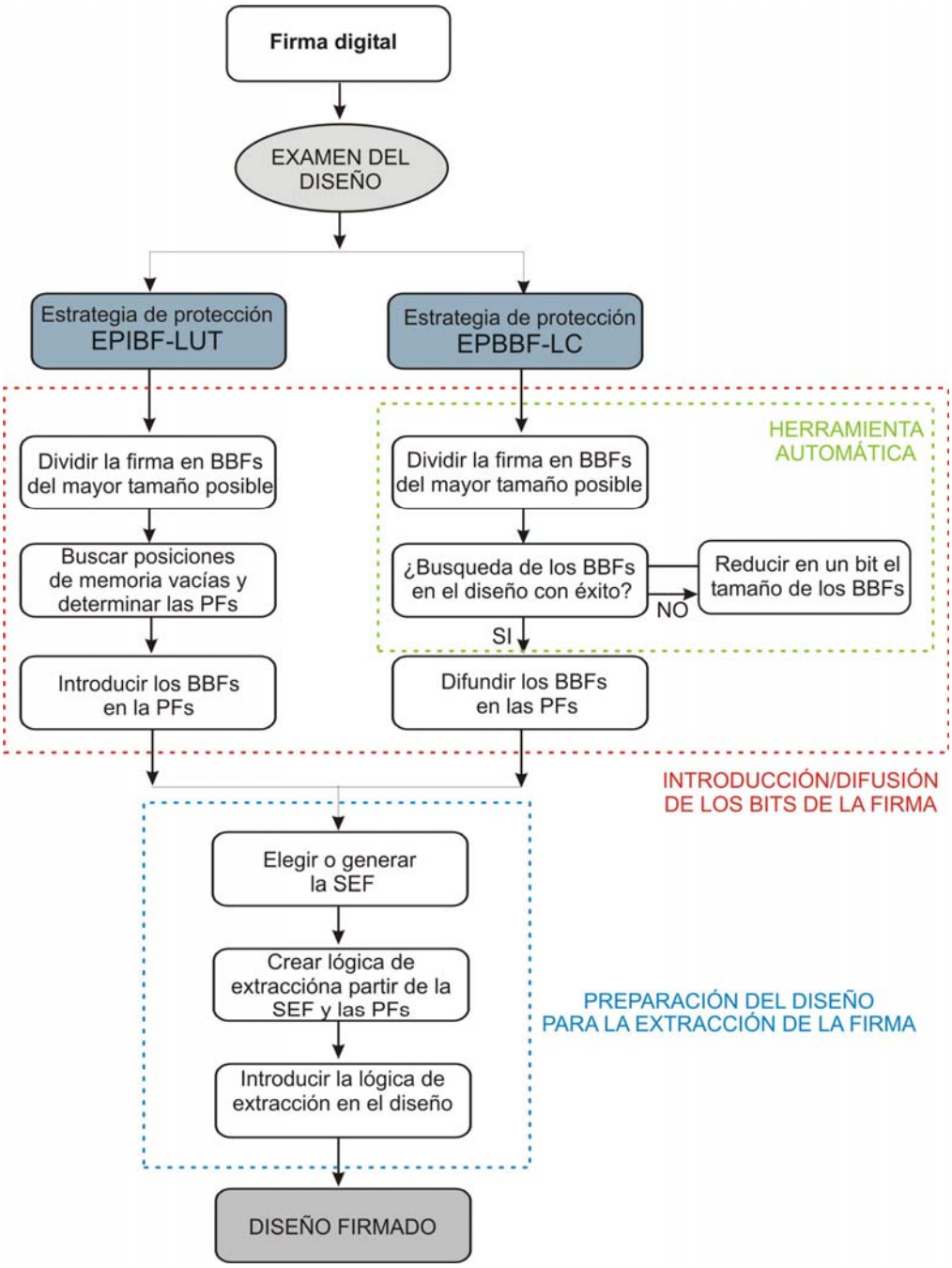

Figura 4.5. Obtención de diseño firmado mediante el método de *watermarking* propuesto.

la firma estén en uso en el diseño original dificulta enormemente el que un atacante pueda hallar dichos bits. Para extraer la firma, la lógica necesaria tendría que responder a una petición de extracción direccionando, en el orden adecuado, las posiciones de memoria que contienen los bits de la firma (posiciones de la firma o PMFs). De este

```
block_definition : process (CLK, table_in)
begin
 if CLK'event and CLK = '1' then
  case table_in is
   when "0000" => table_out <= "0000000000000000000";
   when "0001" => table_out <= "1000101110100010111";
   when "0010" => table_out <= "0001011101000101110";
   when "0011" => table_out <= "1010001011101000101";
   when "0100" => table_out <= "0010111010001011101";
   when "0101" => table_out <= "1011101000101110100";
   when "0110" => table_out <= "0100010111010001011";
   when "0111" => table_out <= "1101000101110100010";
   when "1000" => table_out <= "0101110100010111010";
   when "1001" => table_out <= "1110100010111010001";
   when "1010" => table_out <= "0111010001011101000";
   when others => table_out <= "0000000000000000000";
  end case;
 end if;
end process block_definition;
```

(a)

```
  when "1011" => table_out <= 19-bit_signature_block_1;
  when "1100" => table_out <= 19-bit_signature_block_2;
  when "1101" => table_out <= 19-bit_signature_block_3;
  when "1110" => table_out <= 19-bit_signature_block_4;
  when "1111" => table_out <= 19-bit_signature_block_5;
  when others => table_out <= “0000000000000000000”;
```

(b)

Figura 4.6. Ejemplo del código VHDL de una estructura de memoria:

a) código VHDL original, b) líneas de código que se añadirían para introducir 5 bloques de 19 bits de la firma digital.

modo, tras la correspondiente latencia del sistema, en su salida irían apareciendo los bloques de bits de la firma.

Por ejemplo, el código VHDL que se muestra en la figura 4.6.a) corresponde a una estructura de memoria que utiliza patrones de entrada de 4 bits y proporciona salidas de 19 bits. Esta estructura de memoria puede ser implementada como lógica combinacional o directamente utilizando una tabla de consulta. En concreto, esta estructura de memoria es parte de un canal módulo 11 en RNS y, por lo tanto, los patrones de entrada del 11 al 15 no serán aplicados en ningún caso durante una operación normal del canal. Estas cinco posiciones de memoria se podrían utilizar para introducir parte de los bits de la firma. En este caso sólo sería necesario añadir cinco

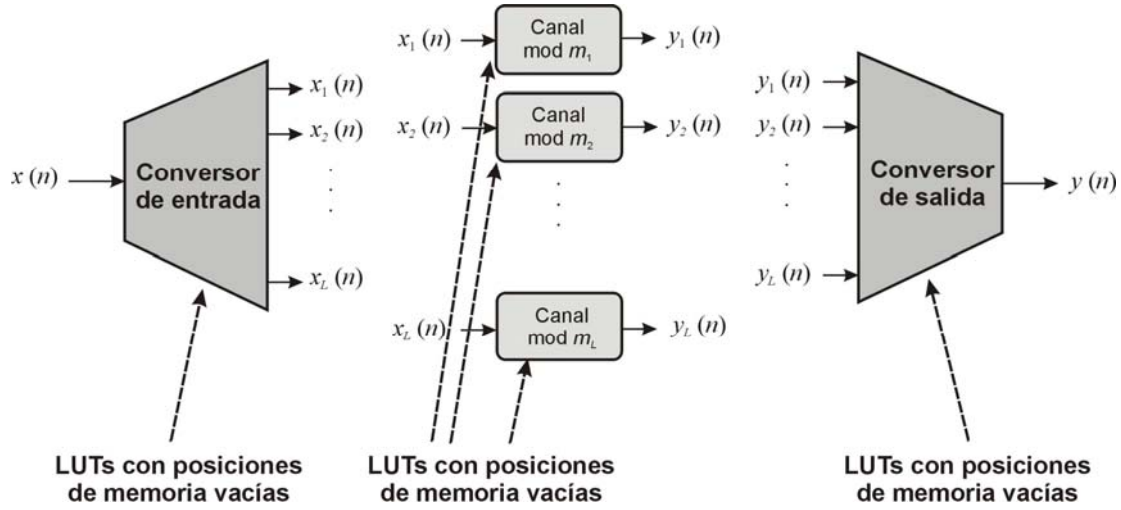

Figura 4.7. Estructura básica de un sistema RNS que muestra las LUTs con posiciones vacías.

asignaciones concurrentes al código VHDL, como se muestra en la figura 4.6.b), conteniendo cada una 19 bits de la firma en la señal `table_out`. Esta simple modificación en la descripción HDL del sistema introduciría en éste 95 bits de la firma digital. De hecho, el proceso de la figura 4.6 forma parte de la descripción HDL de uno de los ejemplos de diseño basados en el RNS descritos en el siguiente capítulo.

Como ya se ha comentado en la sección 4.1, las aplicaciones basadas en el RNS tradicionalmente han hecho un uso extensivo de estructuras de memoria. En estos sistemas se pueden encontrar varios tipos de tablas de consulta con posiciones de memoria que no han sido utilizadas, como se muestra en la figura 4.7:

- ***LUTs de la conversión de entrada***: en caso de que el procesamiento interno de los datos se realice con aritmética de índices [SZA67], el conversor de entrada ha de incluir una conversión binario-RNS y RNS-índices. Las tablas de consulta utilizadas en la conversión RNS-índices contienen posiciones de memoria que no se utilizan y que se pueden usar para introducir los bits de la firma;
- ***LUTs del bloque de procesamiento de datos***: cuando se usa aritmética de índices se procesan índices en lugar de residuos. Las tablas de conversión índices-RNS que se incluyen en el procesamiento tienen posiciones de memoria vacías, que también pueden ser usadas para la firma. Una ventaja que presenta esta alternativa es que estas tablas no son parte de la conversión de entrada o salida.

De esta forma, un bloque RNS podría ser usado junto a otros para formar un sistema más complejo, con una conversión de entrada y salida global, manteniéndose así la protección de la propiedad intelectual de cada bloque RNS;

- ***LUTs de la conversión de salida***: el algoritmo ε-CRT [GRI89] se puede usar en la conversión RNS-binario, como se detalla en el Apéndice B. Las tablas de consulta usadas por esta conversión incluyen posiciones de memoria vacías que pueden ser explotadas para la introducción de los bits de la firma.

Los bits de la firma se pueden introducir en estas posiciones al mismo tiempo que el sistema está siendo descrito, por ejemplo, en la descripción de alto nivel del diseño, e independientemente de la tecnología usada para la implementación final del sistema. De esta forma, como ya se ha mencionado anteriormente, los diseños basados en el RNS fueron los primeros que se utilizaron en este trabajo para la aplicación y evaluación de la estrategia de protección propuesta [PAR04]. Los diseños RNS propuestos para esta estrategia de protección y los resultados obtenidos se verán en detalle en el capítulo 6.

Teniendo en cuenta que para implementar una determinada lógica combinacional las estructuras de memoria son usadas en ocasiones como tablas de consulta, las posiciones de memoria no utilizadas también representan los patrones de salida de dicha lógica combinacional que no han sido utilizados. De esta forma, estos patrones de salida podrían ser utilizados para introducir los bits de la firma en la lógica combinacional del sistema durante la descripción HDL del mismo. Así, aunque en el ejemplo que se ha mostrado se han empleado estructuras de memoria con posiciones vacías para la difusión de los bits de la firma, la estrategia de protección propuesta está orientada en general a circuitos en los que la lógica combinacional disponga de patrones de entrada que no han sido utilizados en el diseño.

Como se ha visto en los ejemplos propuestos, la difusión de los bits de la firma según esta estrategia de protección es una tarea sencilla, sólo es necesario añadir líneas de código a la descripción VHDL del diseño original. Como ya se ha dicho, la localización de los bits de la firma por posibles atacantes presenta gran dificultad, ya que la lógica combinacional empleada para la difusión de los bits de la firma está en uso en el diseño original. Sin embargo, esta estrategia de protección sólo se puede aplicar a diseños que dispongan de una lógica combinacional en la que algunos de los patrones de salida no se utilicen. Un avance más en el método de protección consiste extender la

estrategia de protección para permitir la protección de cualquier diseño, no sólo de aquellos que incluyan lógica combinacional con patrones de salida sin utilizar. A continuación se realiza una generalización de esta estrategia de protección a cualquier tipo de circuito.

4.3.2.2. Estrategia basada en la búsqueda de los bits de la firma

Esta nueva estrategia de protección consiste en buscar bloques de bits de la firma en los patrones de salida de la lógica combinacional incluida en el diseño, de forma totalmente independiente de la estructura lógica utilizada para su implementación (tablas de consulta, redes de puertas lógicas, etc.) [CAS06, CAS07a]. Los patrones de salida que resulten de esta búsqueda serán las posiciones de la firma (PFs) y se tendrá que acceder a ellas para la extracción de los bits de la firma.

Además de extender la protección a cualquier tipo de sistema, la principal diferencia y ventaja que presenta la nueva estrategia de protección con respecto a la anterior es que, en lugar de aprovechar los patrones de salida de la lógica combinacional del sistema que no hayan sido utilizados e introducir en ellos los bloques de bits de la firma, se realiza una búsqueda para tratar de identificar los bloques de bits de la firma con el contenido de los patrones de salida que ya están siendo utilizados en el sistema original. De esta forma, cualquier intento de modificar o eliminar tan sólo uno de los bits de la firma del diseño protegido, modificará la funcionalidad de las partes combinacionales implicadas, afectando así al correcto funcionamiento del sistema.

El código VHDL de la figura 4.6.a) se puede utilizar también para mostrar un ejemplo de la aplicación de esta estrategia de protección. Es posible identificar una parte de la firma digital elegida con algunos de los patrones de salida que muestra el código de este ejemplo de estructura de memoria. Por ejemplo, si un bloque de bits de la firma digital es “1101000101110100010”, es posible identificar este bloque con el patrón de salida correspondiente al patrón de entrada “0111”. Es decir, en la posición de memoria “0111” el diseño ya alberga este bloque de bits de la firma. Otro ejemplo para la aplicación de la estrategia de protección propuesta es la tabla 4.1; en ella se muestra la tabla de verdad de un decodificador BCD a 7 segmentos. Una descripción HDL de tal decodificador traduce esta tabla verdad en el conjunto de asignaciones de señal

Entradas BCD (Patrones de entrada)				**Segmentos de salida (Patrones de salida)**							**Display**
D	**C**	**B**	**A**	**a**	**b**	**c**	**d**	**e**	**f**	**g**	
0	0	0	0	1	1	1	1	1	1	0	
0	0	0	1	0	1	1	0	0	0	0	
0	0	1	0	1	1	0	1	1	0	1	
0	0	1	1	1	1	1	1	0	0	1	
0	1	0	0	0	1	1	0	0	1	1	
0	1	0	1	1	0	1	1	0	1	1	
0	1	1	0	0	0	1	1	1	1	1	
0	1	1	1	1	1	1	0	0	0	0	
1	0	0	0	1	1	1	1	1	1	1	
1	0	0	1	1	1	1	0	0	1	1	

Tabla 4.1. Tabla de verdad de un decodificador BCD de 7 segmentos.

```
with DCBA select
  abcdefg <="1111110" when "0000",
            "0110000" when "0001",
            "1101101" when "0010",
            "1111001" when "0011",
            "0110011" when "0100",
            "1011011" when "0101",
            "0011111" when "0110",
            "1110000" when "0111",
            "1111111" when "1000",
            "1110011" when "1001",
            "0000000" when others;
```

Figura 4.8 Código VHDL para un decodificador BCD de 7 segmentos.

condicionales que se muestran en la figura 4.8. Si el decodificador forma parte de un diseño, es posible identificar parte de la firma digital elegida con los patrones de salida de tal decodificador. Por ejemplo, si uno de los bloques de bits de la firma digital es “1111110”, se puede comprobar que el diseño original incluye estos bits de la firma en el patrón de salida correspondiente al patrón de entrada “0001” (*v*. figura 4.8).

En estos dos ejemplos se muestra cómo con esta estrategia de protección los bloques de bits de la firma no se introducen en el diseño, si no que se realiza una

búsqueda para hallar aquellos patrones de salida, incluidos en la descripción del diseño original, que coincidan con estos bloques de la firma, sin que sea necesario añadir nuevos fragmentos de código al diseño original. Por tanto, con esta estrategia se consigue que los bloques de bits de la firma sean patrones de salida de la lógica combinacional (la estructura de memoria para el primer ejemplo y el decodificador BCD a 7 segmentos para el segundo ejemplo) que son necesarios para el correcto funcionamiento tanto del sistema original como del sistema protegido.

Para extraer los bits de la firma hay que aplicar las posiciones o patrones de entrada en el orden adecuado, es decir, las posiciones donde se encuentran los bloques de bits de la firma. Esto, como en el caso de la anterior estrategia de protección, se hace a través de una lógica de extracción que se introduce en la descripción de alto nivel del diseño. Esta lógica llevará las salidas de los patrones de entrada identificados como posiciones de la firma hacia la salida del sistema. Los detalles sobre el proceso de extracción de la firma, incluyendo las distintas posibilidades para la lógica de extracción necesaria, se verán con más detalle a lo largo de este capítulo y en los ejemplos de aplicación del capítulo 6.

4.3.2.3. Consideraciones adicionales

Es importante aclarar que las estrategias de protección propuestas no están relacionadas con los recursos que se emplearán para la implementación física del diseño. Tampoco guardan relación alguna con los recursos lógicos o primitivas derivadas del proceso de síntesis lógica. Los bits de la firma son difundidos dentro del diseño durante su descripción HDL, a través de patrones de salida de la lógica combinacional incluida en la descripción del diseño original. Teniendo en cuenta todo esto, las estrategias de *watermarking* que se proponen hacen que la protección del sistema se propague a través de todo el flujo de diseño. El sistema es protegido desde los primeros niveles de síntesis lógica sin que sea necesario un proceso de resíntesis. La protección permanece tras el proceso de emplazamiento y enrutamiento (*place&route*), de forma que la implementación física final del diseño está también protegida.

Por otro lado, en los apartados anteriores se han tratado las dos estrategias de protección como técnicas independientes, es decir, tras realizar un examen del diseño

que se quiera proteger, se elegiría una de las dos estrategias de protección y se aplicaría sobre dicho diseño. Sin embargo, es posible utilizar de forma conjunta las dos estrategias para la protección de un mismo diseño, siempre que éste disponga de estructuras de memoria con posiciones vacías o con una lógica combinacional en la que no todos los patrones de salida estén en uso. En este caso, parte de los bits de la firma digital se introducirían en el diseño, aprovechando las posiciones de memoria vacías o los patrones de salida que no han sido utilizados, y el resto de los bits se identificarían con patrones de salida que ya están en uso en el diseño original.

La ventaja que ofrece el uso de la primera de las estrategias de protección es que parte de los bits de la firma se introducen directamente en el diseño, añadiendo nuevas líneas de código. De este modo, al utilizar las dos estrategias como técnica de protección, sólo una parte de los bits de la firma tienen que ser identificados con patrones de salida. Esto hace que la búsqueda necesaria para la difusión del resto de los bits sea más fácil, se tendrían que identificar sólo parte de los bits de la firma, agrupándose en bloques de bits, con patrones de salida que ya estén en uso. Por otro lado, la segunda estrategia de protección hace que el diseño ofrezca mayor resistencia contra ataques, ya que los bits de la firma que se hayan difundido en el sistema formarán parte del diseño original y cualquier intento que trate de eliminarlos estará modificando el correcto funcionamiento del circuito. De este modo, se puede decir que al utilizar conjuntamente las dos estrategias para proteger un determinado módulo IP se consigue llegar a un compromiso entre dos de los factores a tener en cuenta en cualquier método de protección: facilidad para la aplicación y seguridad contra posibles ataques.

4.3.2.4. Elección del tamaño de los bloques de bits de la firma

Hasta ahora se ha hecho referencia a los bloques de bits de la firma asumiendo que su tamaño coincide con el de los patrones de salida en los que el diseño albergará los bits de la firma. Conviene analizar, para cada estrategia de protección, la elección más adecuada del tamaño de los bloques de la firma.

En la primera estrategia de protección, la mejor elección es hacer coincidir el tamaño de los bloques de bits de la firma con el tamaño de las posiciones de memoria en las que los bloques de la firma van a ser introducidos. De esta forma, se consigue

aprovechar al máximo la capacidad de las posiciones de memoria vacías, con lo que se minimizaría el número de posiciones de memoria necesarias para introducir todos los bits de la firma, es decir, número de líneas de código a añadir en la descripción VHDL del diseño y, por tanto, también se reducirían los recursos que emplearía la lógica de extracción. Por otro lado, con esta elección, se consigue que sea menor el número de ciclos de reloj en los que, tras una petición de extracción de la firma, en la salida del sistema irán apareciendo los bloques de bits de la firma.

Sin embargo, para la segunda estrategia de protección esta elección podría dificultar la aplicación de la misma. Como ya se ha comentado anteriormente, en esta estrategia la difusión de los bits se realiza a través de la búsqueda de patrones de salida utilizados por la lógica combinacional incluida en el diseño, en lugar de introducir nuevos patrones de salida. De esta forma, podría suceder que no todos los bloques de bits de la firma coincidiesen con alguno de los patrones de salida de la lógica combinacional. En los ejemplos que se han propuesto para esta estrategia de protección, se ha considerado sólo la posibilidad de que los bloques de la firma tengan el mismo tamaño que los patrones de salida. Sin embargo, también es posible dividir la firma en bloques de bits de un tamaño menor que el de los patrones de salida. Esto aumentaría la probabilidad de encontrar todos los bloques de la firma dentro del diseño original.

Para el ejemplo del decodicador BCD a 7 segmentos de la figura 4.7, si se supone que la firma digital es “11110000” y se tienen en cuenta los bits de los patrones de salida que indica el código VHDL de este decodificador, para facilitar la aplicación de la estrategia de protección y poder difundir todos los bits de la firma, ésta se podría dividir en dos bloques, “1111” y “0000”, que coinciden con los 4 bits menos significativos de las salidas correspondientes a las direcciones “0110” y “0001”, de forma que la firma podría quedar albergada en estas posiciones. Como ya se ha mencionado, para extraer los bits de la firma se deben direccionar estos patrones de entrada en el orden adecuado, pero en este caso sólo los 4 bits menos significativos de los correspondientes patrones de salida serán llevados hasta la salida del sistema para ir formando la firma digital.

En la aplicación de la segunda estrategia es muy importante el tamaño de los bloques de bits de la firma elegido. Conviene elegir tamaños de bloque tan grandes como sea posible, siempre que con este tamaño se puedan identificar todos los bloques

con ciertas posiciones de memoria. Por lo tanto, es necesario realizar un estudio previo de la probabilidad que existe de encontrar, para cada tamaño de bloque posible, todos los bloques de bits de la firma dentro del diseño. Supóngase un diseño que utilice estructuras de memoria con *N* posiciones de memoria de capacidad *C*, o que utilice una lógica combinacional con *N* patrones de entrada y patrones de salida de C bits, y una firma digital descompuesta en *m* bloques de *b* bits. Si se cumple la condición $C \geq b$, la probabilidad de encontrar todos los bloques de bits de la firma dentro de las posiciones de memoria se puede puede obtener a través del siguiente análisis:

$$P = \frac{2^b - 1}{2^b} \tag{4.2}$$

es la probabilidad de no identificar un bloque de bits de la firma con el contenido de una posición de memoria;

$$Q = \left(\frac{2^b - 1}{2^b} \right)^N \tag{4.3}$$

es la probabilidad de no identificar el bloque de bits de la firma con alguna de las *N* posiciones de memoria disponibles dentro del diseño;

$$R = 1 - \left(\frac{2^b - 1}{2^b} \right)^N \tag{4.4}$$

es la probabilidad de encontrar el bloque de bits de la firma en alguna de las *N* posiciones de memoria disponibles dentro del diseño,

$$P_{búsqueda} = \left(1 - \left(\frac{2^b - 1}{2^b} \right)^N \right)^m \tag{4.5}$$

es la probabilidad de encontrar los *m* bloques de bits de la firma en alguna de las *N* posiciones de memoria disponibles dentro del diseño. Además, siempre que $C \geq b$, se puede considerar la posibilidad de descomponer cada una de las *N* posiciones de memoria de capacidad *C* en *n* posiciones de memoria de capacidad *b*. Al sustituir *N* por $N \cdot n$ en (4.5), se puede comprobar que la probabilidad aumentaría.

Analizando la expresión (4.5) se puede comprobar que tamaños de bloque grandes reducen la probabilidad de encontrar todos los bloques de la firma dentro del diseño. Sin embargo, para estos tamaños de bloque la lógica adicional necesaria para la extracción

de la firma, básicamente una máquina de estados finita, necesitará menos estados y, por lo tanto, será menor el incremento del área. Por otro lado, tamaños de bloque pequeños aumentan la probabilidad de identificar estos bloques con patrones de salida dentro del diseño, pero en este caso, la lógica de extracción necesitará mayor número de estados, lo que se traduce en mayor número de recursos. Teniendo en cuenta lo anterior, la mejor opción será la que suponga el mínimo número de recursos adicionales, aunque ésto implique mayor esfuerzo en la difusión de los bits de la firma. De este modo, se debería comenzar la búsqueda de los bloques de bits de la firma eligiendo el mayor tamaño de bloque posible, es decir, aquél que coincida con el tamaño de los patrones de salida de la lógica combinacional involucrada en esta búsqueda, $b=C$. Cuando alguno de los bloques de bits de la firma no se encuentre, se debería reducir en uno el tamaño de los bloques y reanudar la búsqueda.

Como se verá en la sección 4.5, el desarrollo de una herramienta automática de búsqueda va a facilitar la localización de los bloques de bits de la firma para los distintos tamaños de bloque y, de este modo, minimizarán los recursos de la lógica de extracción, sin que esto suponga grandes esfuerzos.

4.3.3. Preparación del diseño para la extracción de la firma

Después del proceso de difusión, el diseño se debe preparar para poder realizar la extracción cuando sea necesaria (figura 4.5). La extracción debe garantizar una fácil, segura y rápida detección de la firma. Con este propósito se ha desarrollado el método que se detalla en esta sección, en lo que se refiere a los recursos a añadir en el diseño, y en la sección 4.4, en cuanto al funcionamiento del método de extracción en sí mismo.

El método de extracción que se propone requiere una lógica adicional a incluir dentro del diseño. La principal función de esta lógica es detectar una secuencia de datos de entrada previamente seleccionada, la secuencia de extracción de la firma (SEF), y generar las señales de control necesarias para dirigir los bloques de bits de la firma hacia la salida del sistema. La lógica de extracción va a depender de la secuencia de extracción de la firma que se haya elegido o de la forma en la que esta secuencia se haya generado. A continuación se van a analizar distintas posibilidades para la secuencia de extracción de la firma y la lógica adicional necesaria para cada uno de los casos.

4.3.3.1. Secuencia de extracción de la firma

La primera opción que se tuvo en cuenta para la secuencia de extracción de la firma fue una secuencia de datos elegida manualmente [PAR04] que reducía de forma importante los recursos necesarios en la implementación de la lógica de extracción, como se verá en la siguiente sección. Por ejemplo, para los primeros casos de aplicación de la técnica de protección, la secuencia de extracción de la firma que se utilizó, teniendo en cuenta datos de entrada de 16 bits, fue: 0, 1, 0, 1, 0, 0. Sin embargo, esta secuencia de extracción es fácil de detectar por posibles atacantes e incluso de aplicar por pura coincidencia.

Para aumentar la invulnerabilidad del método de extracción, se pueden utilizar generadores de secuencias pseudoaleatorias. Los registros de desplazamiento realimentados, o LFSRs (*Linear Feedback Shift Register*) [LEW73], son la base de muchos generadores de secuencias. La segunda opción propuesta para la secuencia de extracción de la firma consiste en generar ésta utilizando un LFSR, desarrollado a partir de polinomios primitivos en GF(2). Los LFSRs que consiguen generar secuencias de longitud máxima son un buen método para claves de seguridad, ya que poseen buenas propiedades estadísticas [MAC76]. Para un registro de desplazamiento con n biestables, las secuencias de longitud máxima tienen 2^n-1 elementos. Para cada valor de n hay varios LFSRs que generan secuencias de longitud máxima. Además, estas secuencias de longitud máxima son pseudoaleatorias, pues cumplen propiedades muy próximas a las secuencias estrictamente aleatorias. Por tanto, la secuencia de datos que genera un LFSR resulta muy difícil de analizar en un ataque criptográfico. La figura 4.9 muestra dos posibles realizaciones de longitud 8 para LFSRs.

Cuando se quiera extraer la firma del circuito, se deberá aplicar en la entrada de datos la secuencia de extracción de la firma y se deberá detectar esta secuencia de datos para activar el proceso de extracción. Para realizar esta detección, la lógica de extracción incluirá un bloque LFSR, basado en el mismo algoritmo que se utilizó para generar la secuencia de extracción de la firma. Como parte de la lógica de extracción, este bloque LFSR deberá ser sintetizado mediante código HDL y ser incluido en la descripción HDL del diseño.

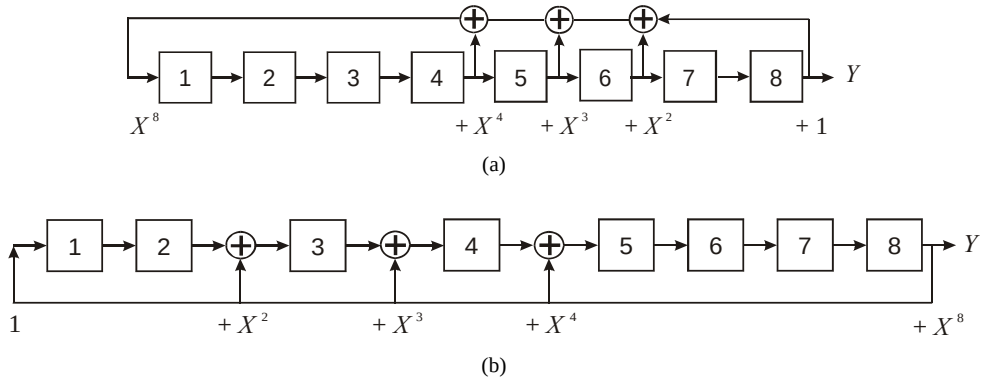

Figura 4.9. Posibles realizaciones de LFSRs: a) Configuración tipo 1 , b) Configuración tipo 2.

4.3.3.2. Lógica para la extracción de la firma

La función que desempeña la lógica de extracción de la firma es detectar la secuencia de extracción de la firma, generar las señales adecuadas para direccionar las posiciones de la firma y enrutar los bloques de bits de la firma hacia la salida del sistema. La lógica adicional se describe en HDL, y se incluye de forma apropiada en la descripción del diseño. Como se mencionó anteriormente, esta lógica de extracción depende de la secuencia de extracción de la firma:

- si la SEF se elige manualmente, la lógica de extracción consiste básicamente en un FSM y multiplexores. La máquina de estados, tras detectar cada dato de la secuencia de extracción, activará el proceso de extracción y generará las señales necesarias para direccionar las posiciones de la firma y acceder así a los bloques de bits de la firma. Los multiplexores se utilizan para el enrutado de los bits de la firma hacia la salida del sistema;
- si la secuencia de extracción de la firma se genera mediante un LFSR, la lógica de extracción estará compuesta por el LFSR correspondiente, un comparador, un contador iniciado a cero, un FSM y multiplexores, como muestra la figura 4.10. El LFSR que forma parte de la lógica de extracción sigue el mismo algoritmo que el que se utilizó para la generación de la secuencia de extracción. Esta lógica de extracción funciona como se detalla a continuación: el LFSR genera el primer dato

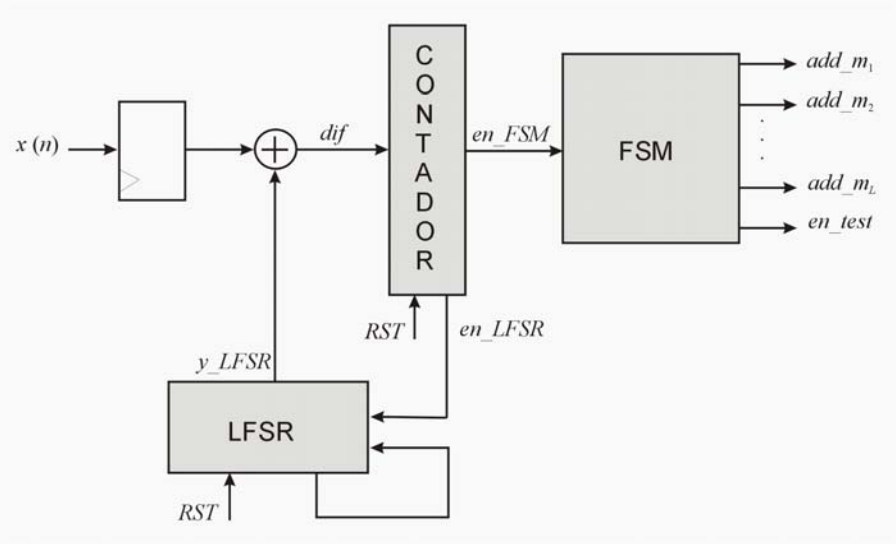

Figura 4.10. Lógica adicional para la extracción de la firma basada en un LFSR .

de esta secuencia de extracción y se comprueba si coincide con el que llega por la entrada de datos del sistema. En el caso de que coincidan, el contador se incrementa y el LFSR genera el siguiente dato de la secuencia de extracción, que volverá a ser comparado con el nuevo dato de entrada y así sucesivamente mientras los datos coincidan. Si en algún momento los datos no coinciden, se reinician el contador y el LFSR. Cuando se hayan aplicado en la entrada del sistema sucesivamente todos los datos que componen la secuencia de extracción de la firma y en el orden adecuado, el contador habrá alcanzado un determinado valor, que le indicará al FSM que se ha activado el proceso de extracción de la firma. Una vez activado, el FSM generará como señales de salida las PFs para tener acceso así a su contenido, es decir, a los bloques de bits de la firma. Como en el caso anterior, los multiplexores hacen posible que los bloques de la firma se dirijan hacia la salida del sistema.

Es importante destacar que las PFs forman parte de los mismos patrones de entradas de la lógica combinacional utilizada durante el funcionamiento normal del sistema.

4.4. Proceso de validación de la firma

En el caso de que el propietario de un determinado bloque IP crea que éste ha sido distribuido de forma ilegal y ha tenido lugar un uso no autorizado, se pueden reclamar los derechos de autor sobre dicho módulo a un equipo oficial de validación. El autor debe proporcionar a este equipo el documento de dominio público que guarda la marca de agua que, según el propietario, fue utilizada para firmar el bloque IP. El proceso de validación requiere extraer la firma del diseño. De esta forma, el propietario debe proporcionar también a este equipo de validación la secuencia de extracción de la firma. Además, en el caso de que se utilice un algoritmo de cifrado asimétrico en la creación de la firma digital, el equipo de validación dispone de las claves necesarias para el descifrado de la firma que se extraiga del diseño (su clave privada y la clave pública del autor del diseño). Este proceso de validación supone que el equipo oficial de validación no hará pública la secuencia de extracción de la firma, las claves de cifrado y descifrado, ni otra información que pudiera poner en peligro la integridad del método de protección propuesto.

El equipo de validación ha de realizar las siguientes acciones:

- extraer la firma digital del diseño (figura 4.11), que se activa aplicando la SEF. El sistema continúa funcionando según su modo normal de operación, pero durante unos cuantos ciclos de reloj en la salida del sistema van apareciendo los bits de la firma, que se deben agrupar para obtener la firma digital del diseño;
- en caso necesario, aplicar corrección de errores a la firma digital extraída y decodificarla utilizando la clave privada del equipo de validación y la clave pública del propietario para obtener la secuencia *hash* correspondiente;
- por otro lado, obtener la secuencia *hash* a partir del documento de dominio público que contiene la marca del autor y/o propietario que reclama los derechos sobre el módulo IP;
- comparar las secuencia *hash* generada a partir del documento de dominio público y la que proviene de la extracción del diseño; si ambas coinciden, quedarían demostrados los derechos de autor sobre el módulo IP.

Figura 4.11. Proceso de extracción de la firma.

4.5. Herramienta automática para la difusión

Después del análisis de la sección 4.3.2.4, es claro que para la estrategia de protecccción EPBBF-LC, la elección del tamaño del bloque de la firma más adecuado y la búsqueda de los bloques de bits de la firma depende de cada diseño y de la firma digital con la que se pretende identificar el diseño. Realizar estas tareas manualmente requiere tiempo y esfuerzo considerables. Además, es bastante difícil llegar a conseguir

la mejor solución para la difusión de los bits de la firma, es decir, aquella que consiga generar un diseño firmado con la menor repercusión posible en el área y las prestaciones. Con la estrategia EPBBF-LC es por tanto necesario el desarrollo de alguna herramienta que ayude a facilitar la tarea de difusión y que, tras encontrar un conjunto de soluciones, elija la mejor. Con éste propósito se ha desarrollado una herramienta *software*, cuyas principales características se detallan a continuación.

4.5.1. Características principales

En el desarrollo de la herramienta automática para un determinado módulo HDL se han teniendo en cuenta los siguientes objetivos:

- facilitar el proceso de difusión de los bits de la firma dentro de la lógica combinacional que forma parte del módulo;
- realizar esta difusión considerando la posibilidad de poder elegir diferentes tamaños de bloque para la secuencia de bits de la firma;
- seleccionar la función de coste que ayude a encontrar la solución que consiga las mínimas modificaciones en área y prestaciones con respecto al diseño original;
- seleccionar el algoritmo de búsqueda para reducir el valor de la función de coste sin que el tiempo de ejecución de dicho algoritmo sea excesivo.

El primer objetivo de la herramienta automática es facilitar el proceso de difusión de los bits de la firma. Al igual que mediante una búsqueda manual, la herramienta automática realiza una búsqueda dentro de la lógica combinacional del diseño a proteger. La secuencia de bits de la firma se divide en bloques de igual o menor tamaño que los patrones de salida de la lógica combinacional. El procedimiento de búsqueda se puede realizar de dos formas: teniendo en cuenta las tablas de verdad de la lógica combinacional, o reproduciendo las operaciones de la lógica combinacional. De cualquier forma, como resultado de la búsqueda la herramienta automática genera un grupo de patrones de entrada correspondientes a patrones de salida que coinciden con los bloques de bits de la firma.

Como ya se mencionó, se pueden minimizar recursos de implementación si el tamaño de los bloques de bits de la firma es igual al tamaño de los patrones de salida en

los que estos bloques de bits van a ser difundidos. Teniendo en cuenta lo anterior, la herramienta automática comenzará la búsqueda de la firma dividiendo ésta en bloques de tamaño igual a los patrones de salida de la lógica combinacional que forma parte del diseño a proteger. En el caso de que no se hayan encontrado todos los bloques de la firma dentro del diseño, la herramienta volverá a realizar una búsqueda reduciendo en un bit el tamaño de los bloques de la firma. Esta búsqueda continuará hasta que se consigan localizar todos los bloques de bits de la firma.

4.5.2. Función de coste

Otro aspecto importante en la herramienta automática es la posibilidad de que un determinado bloque de bits se pueda identificar con la salida de diferentes patrones de entrada o con una parte de ella, en el caso de que el bloque de bits de la firma tenga un tamaño menor al del patrón de salida. De esta forma, para cada bloque de la firma, la herramienta automática tiene que elegir uno de los patrones de entrada. La opción más sencilla es realizar la elección de forma aleatoria; así se hace en el primero de los algoritmos de búsqueda desarrollados. Sin embargo, se pueden aprovechar las distintas posibilidades para la difusión de cada bloque de bits de la firma y elegir una solución que, de acuerdo con un determinado criterio, sea óptima. Por ejemplo, la elección de una determinada solución de difusión puede estar controlada por una función de coste que optimice los recursos de implementación adicionales que necesitará la lógica de extracción de la firma [CAS07b,CAS07c]. Considerando la arquitectura de esta lógica de extracción, se ha elegido como función de coste la distancia Hamming entre patrones de entrada que generarán dos bloques de la firma adyacentes.

Por ejemplo, supóngase que la lógica combinacional que se va a emplear para la difusión de los bits de la firma es la de la suma de dos datos de entrada de 4 bits, llamados *x* e *y*, y datos de salida de 4 bits. Puesto que la lógica combinacional es la suma y siempre va a ser posible obtener un dato como la suma de otros dos, el tamaño de los bloques de bits de la firma coincide con el tamaño de los datos de salida. Supóngase también que dos bloques de la firma adyacentes son *A*=”1110” y *B*=”1010”. Existen varias opciones para realizar la difusión de los bits de la firma dentro del diseño. Para el bloque de la firma *A*, las posibilidades son *x*=“0001” e *y*=“1101” (1+13=14), *x*=“0010” e *y*=“1100” (2+12=14), *x*=“0011” e *y*=“1011” (3+11=14) y así

sucesivamente. Por otro lado, para el bloque de la firma *B*, las opciones son *x*="0001" e *y*="1001" (1+9=10), *x*="0010" e *y*="1000" (2+8=10), *x*="0011" e *y*="0111" (3+7=10) y así sucesivamente. Si se tiene en cuenta la condición de distancia Hamming mínima, *x*="0000" *y*="1110" para el bloque de la firma *A* y *x*="0000" *y*="1010" para el bloque de la firma *B*, ó *x*="0001" *y*="1101" para el bloque de la firma *A* y *x*="0001" *y*="1001" para el bloque de la firma *B* son dos de las elecciones que minimizan la distancia Hamming, ya que el patrón de entrada *x* no cambia y el patrón de entrada *y* sólo cambia en 1 bit. El objetivo final de la elección de las posiciones de la firma teniendo en cuenta la función de coste, es hacer que se optimicen las transiciones o cambios de las señales necesarias en la máquina de estados que forma parte de la lógica de extracción, lo que ayuda a conseguir que ésta necesite menos recursos. Como ya se ha mencionado, esta lógica de extracción va a generar los correspondientes bloques de bits de la firma y los va a dirigir hacia la salida del sistema.

4.5.3. Algoritmos de búsqueda

Se han considerado tres opciones para el desarrollo de la herramienta automática. La más sencilla consiste en elegir una solución aleatoria de todas las que ofrece la lógica combinacional. De este modo, para cada bloque de bits de la firma, este algoritmo tomaría como solución para la difusión de cada bloque la primera que encuentre. Con este algoritmo no se tiene en cuenta la función de coste y, por lo tanto, no se realiza ninguna optimización. Este algoritmo obtiene resultados similares a los obtenidos de la difusión manual, con la ventaja de que se automatiza el proceso y, por tanto, se reducen el esfuerzo y tiempo necesario por la búsqueda. Por otro lado, otra ventaja que ofrece este algoritmo en comparación con los otros dos, es el tiempo de ejecución que va a precisar, ya que para un determinado bloque de bits la búsqueda finaliza cuando encuentra la primera solución posible.

La segunda alternativa tiene en cuenta la función de coste elegida y consiste en encontrar la distancia mínima mediante búsqueda exhaustiva entre la descomposición de un bloque y el siguiente. Con este algoritmo se encuentran todas las posibles combinaciones para difundir los dos primeros bloques de la firma y se fijaría como solución aquella que consiga la menor distancia Hamming entre estos dos bloques. Para difundir el tercer bloque de la firma se buscan todas las posibles soluciones entre el

segundo y este tercer bloque de bits, teniendo en cuenta que la solución para el segundo bloque ha quedado fijada en el paso anterior. Después de esta búsqueda se elegiría como solución para el tercer bloque aquélla que consiga la menor distancia Hamming. Este proceso de búsqueda se repite para el resto de bloques, de forma que para obtener la solución de difusión optima de un bloque se buscan todas las soluciones con respecto al bloque anterior, cuya solución es fija. Con este algoritmo no se consigue un mínimo exacto, puesto que al minimizar no se tienen en cuenta todos los bloques simultáneamente, pero se sitúa cerca del mínimo total. Dada una lógica combinacional, el tiempo de ejecución para este algoritmo aumenta con el tamaño de los bloques de bits de la firma. Como se verá más adelante, para algunos tamaños alcanza unos valores muy elevados.

Por último, se ha considerado un algoritmo en el se tiene en cuenta también la función de coste pero usando Enfriamento Simulado (*Simulated Annealing*, SA) [DOW03, KIR83]. Se minimiza la distancia Hamming entre bloques adyacentes de forma global: las soluciones para cada bloque de la firma se optimizan con respecto a las soluciones para todos los bloques. Además, presenta la ventaja de que, comparado con el algorimo anterior, se consigue reducir considerablemente el tiempo de ejecución cuando aumenta el tamaño de bloque. A continuación se describen algunos de los aspectos más importantes sobre este algoritmo.

4.5.4. Enfriamiento simulado

El concepto de Enfriamiento Simulado se introdujo hace más de veinte años [KIR83] y desde entonces ha demostrado ser una herramienta muy útil para resolver una amplia gama de problemas de optimización en diversos campos, como el problema del viajante de comercio [AAR88], el particionamiento de grafos [BHA87], diseño VLSI [BAN90], diseño de códigos [BEE85], procesamiento de imágenes [GEM87], minimización de funciones lógicas expresadas mediante las primitivas AND-EXOR [PAR94], [PAR99], etc. Es una variante de la búsqueda local que permite movimientos ascendentes para evitar quedar prematuramente atrapado en una mínimo local. Por tanto, se plantea como un algoritmo para la minimización de la función a optimizar.

Los algoritmos tradicionales de búsqueda local parten de una solución inicial que, de modo paulatino, es transformada en otras que a su vez son mejoradas al introducir pequeñas perturbaciones o cambios. Si este cambio da lugar a una solución mejor que la actual, se sustituye ésta por la nueva, continuando el proceso hasta que no es posible ninguna nueva mejora. Esto significa que la búsqueda finaliza en un óptimo local, que no tiene por qué ser forzosamente el global. Un modo de evitar este problema es permitir que algunos movimientos sean hacia soluciones peores. Si la búsqueda está realmente yendo hacia una buena solución, estos movimientos deben realizarse de un modo controlado. En el caso del Enfriamiento Simulado, esto se realiza controlando la frecuencia de los movimientos hacia soluciones peores mediante una función de probabilidad que hará disminuir estos movimientos conforme avanza la búsqueda, de forma que se consigue estar más cerca del óptimo global. Los fundamentos de este control están basados en los trabajos realizados en el campo de la termodinámica estadística [MET53] en los que se modeló el proceso de enfriamiento simulando los cambios energéticos en un sistema de partículas conforme decrece la temperatura, hasta que converge a un estado estable correspondiente al congelamiento. Las leyes de la termodinámica dicen que a una temperatura T la probabilidad de un incremento energético de magnitud δE es:

$$P(\delta E) = e^{\left(\frac{-\delta E}{kT}\right)} \tag{4.3}$$

siendo k la constante de Boltzmann. Los cambios de energía resultantes se calculan generando una perturbación aleatoria en el sistema. Si como resultado de esta perturbación hay caída energética, el cambio se acepta automáticamente; sin embargo, si se produce un incremento energético, el cambio se acepta con una probabilidad dada por la expresión (4.3). El proceso se repite durante un número predefinido de iteraciones en series decrecientes de temperaturas, hasta que el sistema está “frio”.

Cualquier implementación de lógica combinatoria puede convertirse en una implementación de Enfriamiento Simulado mediante una elección de los elementos de entorno de modo aleatorio, la aceptación automática de todos los movimientos hacia una mejor solución y la aceptación de movimientos hacia una peor solución de acuerdo con la probabilidad dada por la expresión (4.3), pero sin considerar la constante k, ya que para problemas de optimización esta constante no tiene significado. Siguiendo la

analogía del enfriamiento físico, en los métodos de optimización basados en el Enfriamiento Simulado existe una parámetro de control denominado temperatura, *t*. Una determinada solución que suponga un incremento en δ en la función de coste se aceptará con probabilidad:

$$P(\delta) = e^{\left(\frac{-\delta}{t}\right)} \tag{4.4}$$

El parámetro *t* permite moverse a soluciones peores que la del estado actual; cuanto mayor es *t* más facilidad habrá para hacer esto. Al inicio *t* es grande y según se avanza en el proceso este parámetro va disminuyendo; al final es tan bajo que el algoritmo se comporta como uno de búsqueda local simple. Un algoritmo de búsqueda local simple busca una solución mejor, y nunca tiene la opción de moverse a un estado peor que el actual. Si este enfriamiento es lo suficientemente lento, se pueda alcanzar una solución óptima absoluta y no una solución óptima local.

4.5.5. Algoritmo de Enfriamiento Simulado

El Enfriamiento Simulado se considera como una estrategia heurística que necesita de varias decisiones para que quede totalmente diseñado, las cuales tienen una gran influencia en la calidad de las soluciones generadas. Un conjunto de estas decisiones están relacionadas con la forma de controlar la temperatura, incluyendo la definición de su valor inicial y la función de crecimiento, el número de iteraciones antes de bajar la temperatura y las condiciones que permitirán considerar que el sistema ya está “frio”. Por otro lado, otro conjunto de soluciones comprenden la definición del espacio de soluciones y la estructura de entornos, la función de coste y cómo se obtiene la solución inicial. El algoritmo sigue el esquema de la figura 4.12.

Como se ha comentado anteriormente, se ha desarrollado un algoritmo de difusión de los bloques de bits de la firma con los patrones de salida de una determinada lógica combinacional utilizando un proceso de Enfriamiento Simulado. En este algoritmo el espacio de soluciones está formado por todas las posibles combinaciones de patrones de entrada que dan lugar a una difusión de los bits de la firma; la solución inicial se elige de forma adecuada para explorar lo mejor posible el espacio de soluciones, la función de coste elegida es la misma que para el algoritmo no optimizado, la distancia Hammnig

```
Sea f(s) la función de coste elegida y sea N(s) su entorno:
Seleccionar:
     una solución inicial s0;
     una temperatura inicial t0>0;
     una función de reducción de la temperatura α;
     un número de iteraciones i;
     un criterio de parada c;

REPETIR
     REPETIR
          Seleccionar aleatoriamente una solución s ∈ N(s0);
          Sea δ=f(s)−f(s0);
          Si δ<0 ENTONCES s0=s;
          SI NO
               Generar aleatoriamente u ∈ U(0,1);
               Si u<exp(−δ/t) ENTONCES s0=s;
          FIN SI NO
     HASTA QUE cuenta_iteraciones=i;
     t= α(t);
HASTA QUE c=CIERTO.
```

Figura 4.12. Esquema del algoritmo de Enfriamiento Simulado.

entre patrones de entrada que generan dos bloques de la firma adyacentes, y el resto de parámetros o decisiones se han ajustado experimentalmente, procurando obtener una buena solución sin que los tiempos de ejecución sean excesivamente altos. Así, en todos los casos se ha partido de una temperatura inicial t=20, que implica una probabilidad de aceptación bastante alta de soluciones que incrementen la función de coste. La temperatura se decrementa simplemente reduciéndola en Δt, que se puede variar entre 0.01 y 0.05 dependiendo de la bondad de la solución que se desee obtener. A efectos de comparación con los otros algoritmos se ha seleccionado finalmente un Δt =0.05, que proporciona un buen compromiso entre tiempo de ejecución y distancia Hamming obtenida. Como criterio de fin del proceso se utiliza una temperatura final igual a 0.1, puesto que a dicha temperatura prácticamente ya no se aceptan soluciones que no mejoren la función de coste y tampoco modificaciones en la solución final. A cada t el número de iteraciones utilizado es i=3.

Otros parámetros se deben definir teniendo en cuenta la lógica combinacional utilizada para la descomposición de los bloques de bits de la firma, como sucede con la solución inicial y el estado siguiente. Se van a considerar dos lógicas combinacionales independientes para la descomposición de cada bloque mediante el algoritmo de Enfriamiento Simulado. Estas lógicas combinacionales son la suma, y la suma de productos. Para la suma, cada bloque de bits B_i se descompone de la siguiente forma:

$$B_i = S_{i1} + S_{i2} \tag{4.5}$$

Mientras que para la suma de productos, cada bloque de bits B_i se descompone de esta otra forma:

$$B_i = P_{i1} + P_{i2} = a_i \times b_i + c_i \times d_i \tag{4.6}$$

En ambos casos, el estado inicial para la descomposición del primer bloque es el valor medio del dato a descomponer, de forma que si el dato a descomponer es par se divide en dos partes iguales, y si es impar se coge la mitad ± 1. De esta forma quedarían fijados S_{11} y $S_{12,}$ para la suma, y P_{11} y P_{12}, para la suma de productos, donde el primer subíndice hace referencia al número de bloque de bits de la firma que se está descomponiendo. Para la suma la primera descomposición habría finalizado, pero para la suma de productos cada uno de los términos fijados, P_{11} y P_{12}, se tiene que descomponer como un producto, según (4.6). El proceso iterativo para conseguir esta descomposición es:

1. iniciar a 1 a_i y c_i,
2. comprobar si, teniendo en cuenta los valores de p_{11} y p_{12}, la descomposición es posible, es decir si los valores de b_i y d_i necesarios se encuentran dentro del intervalo de valores posibles,
3. si es así, el proceso de descomposición habría finalizado,
4. si no, se incrementa el valor de a_i y c_i y se vuelve al paso 2.

Tanto para la suma como para la suma de productos no habría estado siguiente para la descomposición del primer bloque de bits de la firma. Para los restantes bloques de bits de la firma el estado inicial es $S_{i1}=S_{i-1,1}$, $S_{i2}=B_i-S_{i1}$ para la suma ó $P_{i1}=P_{i-1,1}$, $P_{i2}=B_i-P_{i1}$ para la suma de productos. En el caso de que $B_i<S_{i-1,1}$ el estado inicial es $S_{i1}=0$, $S_{i2}=B_i$ para la suma ó $P_{i1}=0$, $P_{i2}=B_i$ para la suma de productos

El estado siguiente para S_{i1} ó p_{i1} se calcula sumando o restando de forma aleatoria el 20% al valor que se tiene para S_{i1} ó P_{i1} y se calculan los nuevos valores de S_{i2} ó P_{i2}. En el caso de la suma de productos, P_{i1} y P_{i2} se tiene que descomponer como un producto utilizando para ello el mismo proceso iterativo que se ha descrito para el primer bloque de bits de la firma. La aceptación del nuevo estado generado se lleva a cabo aplicando la función de probabilidad indicada en el algoritmo de la figura 4.13. El proceso continúa, reduciendo la temperatura progresivamente hasta que se alcanza el criterio de fin, que en esta ocasión correpsonde a alcanzar una temperatura final igual a 0.

4.5.6. Aplicación de los algoritmos

Se han comparado los tres algoritmos propuestos considerando dos lógicas combinacionales distintas para la difusión de los bits de la firma, la suma y la suma de

		No optimizada		**Optimizada**		**SAMB**	
tipo	**t.bloque**	**MDH**	**TE**	**MDH**	**TE**	**MDH**	**TE**
MD5	2	78	0.032	43	0.084	44	8.04
	4	99	0.046	53	0.120	55	7.25
	6	109	0.051	55	0.410	56	5.16
	8	106	0.053	50	4.280	51	4.74
	10	110	0.050	49	166.83	52	4.56
	12	110	0.016	––	––	52	5.08
	16	105	0.021	––	––	53	4.20
	20	96	0.020	––	––	50	4.77
	24	80	0.020	––	––	41	5.30
	30	64	0.023	––	––	32	5.70
SHA1	2	99	0.130	55	0.150	55	9.11
	4	125	0.170	68	0.150	70	7.71
	6	136	0.078	67	1.410	69	8.89
	8	135	0.037	62	23.900	63	6.77
	10	139	0.036	63	195.86	64	6.31
	12	146	0.025	––	––	67	7.15
	16	136	0.023	––	––	69	5.40
	20	112	0.041	––	––	57	5.43
	24	96	0.030	––	––	49	6.83
	30	81	0.022	––	––	40	6.90

Tabla 4.2. Media de las distancias Hamming y tiempos de ejecución obtenidos mediante los tres algoritmos de búsqueda para la suma como lógica combiancional.

		No optimizada		Optimizada		SAMB	
tipo	**t.bloque**	**MDH**	**TE**	**MDH**	**TE**	**MDH**	**TE**
MD5	2	78	0.032	43	0.084	44	8.04
	4	99	0.046	53	0.120	55	7.25
	6	109	0.051	55	0.410	56	5.16
	8	106	0.053	50	4.280	51	4.74
	10	110	0.050	49	166.83	52	4.56
	12	110	0.016	––	––	52	5.08
	16	105	0.021	––	––	53	4.20
	20	96	0.020	––	––	50	4.77
	24	80	0.020	––	––	41	5.30
	30	64	0.023	––	––	32	5.70
SHA1	2	99	0.130	55	0.150	55	9.11
	4	125	0.170	68	0.150	70	7.71
	6	136	0.078	67	1.410	69	8.89
	8	135	0.037	62	23.900	63	6.77
	10	139	0.036	63	195.86	64	6.31
	12	146	0.025	––	––	67	7.15
	16	136	0.023	––	––	69	5.40
	20	112	0.041	––	––	57	5.43
	24	96	0.030	––	––	49	6.83
	30	81	0.022	––	––	40	6.90

Tabla 4.3. Media de las distancias Hamming y tiempos de ejecución obtenidos mediante los tres algoritmos de búsqueda para la suma de productos como lógica combinacional.

dos productos, y se han utilizando serie aleatorias de firmas digitales MD5 y SHA1. El tamaño de muestra que se ha elegido para cada serie aleatoria proporciona un error máximo del ±5% con una confianza del 95%. Para cada una de estas firmas se consideraron distintos tamaños de bloque, comprendidos entre 2 y 30 bits. Por un lado, se han realizado las descomposiciones en términos suma de cada uno de estos bloques, mediante los tres algoritmos de difusión propuestos. Por otro lado, se ha realizado otra descomposición como suma de dos productos.

En las tablas 4.2 y 4.3 se muestran los resultados obtenidos para la ejecución de los tres algoritmos de búsqueda para la suma y la combinación lineal respectivamente, con ayuda de un PentiumIV a 2.2GHz. En cada una de las filas de estas tablas se muestran los datos obtenidos para la difusión de las 100 firmas consideradas bajo los tres algoritmos propuestos. Para cada tamaño de bloque y algoritmo, las medidas

mostradas en estas tablas son la media y la varianza de las distancias Hamming globales correspondientes a los patrones de salida obtenidos para la difusión de la firma y el tiempo de ejecución del algoritmo correspondiente. Un aspecto favorable del algoritmo no optimizado es que sus tiempos de ejecución son muy reducidos; sin embargo, las medidas sobre las distancias Hamming globales muestran que este algoritmo no ofrece la solución óptima para la difusión de los bits de la firma. Por otro lado, el algoritmo optimizado consigue reducir de forma considerable la media y la varianza de las distancias Hamming globales con respecto al algoritmo no optimizado, pero presenta el inconveniente de que los tiempos de ejecución aumentan a medida que aumenta el tamaño del bloque de bits, con crecimiento exponencial, llegando a ser excesivamente elevado para tamaños de bloque superiores a 10 bits.

Las tablas 4.2 y 4.3 muestran las importantes ventajas que ofrece el algoritmo de Enfriamiento Simulado respecto de los otras dos alternativas propuestas. Con este algoritmo se consiguen reducir la media y la varianza de la distancia Hamming global, sobre todo si se compara con el algoritmo no optimizado. La figura 4.13 compara la media de las distancias Hamming globales para la suma como lógica combinacional y los algoritmos de Enfriamiento Simulado y el no optimizado como algoritmos de búsqueda. Como se puede comprobar, se obtienen mejores resultados para el algoritmo de Enfriamiento Simulado. Por otro lado, si se compara con el algoritmo optimizado, el algoritmo de Enfriamiento Simulado también puede ofrecer ventajas en cuanto a tiempos de ejecución. Para tamaños de bloque pequeños, los tiempos de ejecución del algoritmo de Enfriamiento Simulado son mayores que los obtenidos para el algoritmo optimizado. Por ejemplo, en el caso de la suma, para el menor de los tamaños de bloque considerados, 2 bits, el tiempo de ejecución es de 0.084 y 8.04 segundos para el algoritmo optimizado y el algoritmo de Enfriamiento Simulado, respectivamente. Sin embargo, para un tamaño de bloque de 10 bits, el tiempo de ejecución el algoritmo optimizado es de 166,83 segundos, mientras que para el algoritmo de Enfriamiento Simulado es tan sólo de 4,56 segundos. Se observa, que a medida que aumenta el tamaño de bloque, los tiempos de ejecución del algoritmo basado en el Enfriamiento Simulado se reducen considerablemente con respecto al algoritmo optimizado. En general, para tamaños de bloque superiores a 10 bits, los tiempos de ejecución del algoritmo optimizado son muy elevados mientras que para el algoritmo de Enfriamiento Simulado son del mismo orden de magnitud que para tamaños de bloque más pequeños.

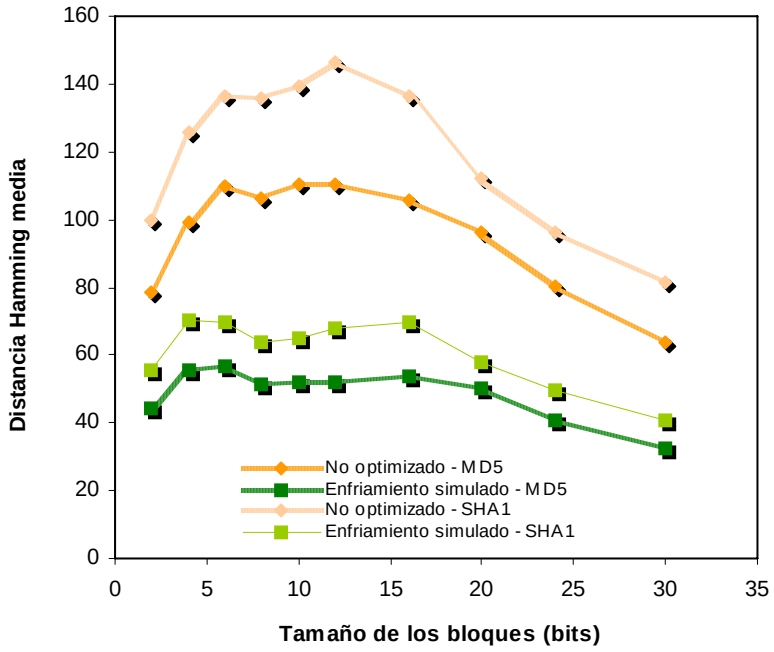

Figura 4.13. Distancia Hamming media en función del tamaño de bloque de la firma para los algoritmos no optimizado y el de Enfriamiento Simulado.

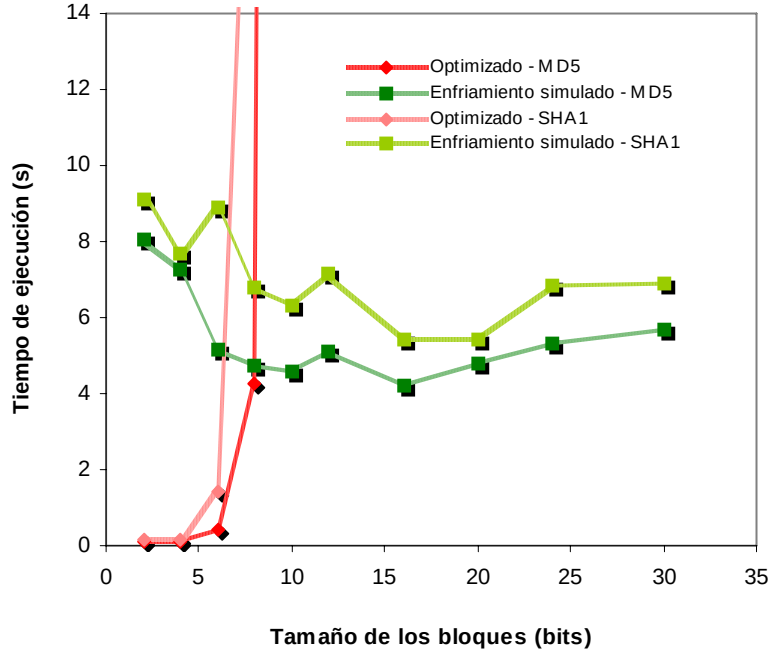

Figura 4.14. Tiempo de ejecución en función del tamaño de bloque de la firma para los algoritmos optimizado y el de Enfriamiento Simulado.

Los tiempos de ejecución obtenidos para la suma se han utilizado para elaborar el gráfico representado en la figura 4.14. Se puede comprobar que la tendencia de la curva para el algoritmo optimizado es exponencial, llegando a alcanzar valores que se consideran no aceptables para la aplicación considerada. Sin embargo, para el algoritmo de Enfriamiento Simulado los datos representados se mantienen dentro de un intervalo inferior a 5 segundos y, que para tamaños de bloque superiores a 10 bits, los tiempos de ejecución son al menos 2 órdenes de magnitud inferiores a los obtenidos para el algoritmo no optimizado.

4.6. Conclusión

En este capítulo se ha descrito un nuevo método para la protección de bloques IP mediante la difusión de los bits de una firma digital segura que identifica el origen del diseño. El método de protección se realiza en la descripción de nivel alto y dicha protección permanece tras los procesos de síntesis, emplazamiento y enrutamiento del diseño. Se ha comenzado analizando los pasos a seguir para obtener una firma digital segura a partir de la marca de agua con la que se quiere identificar el diseño. Seguidamente, se han propuesto dos estrategias de protección basadas en dos posibles formas de realizar la difusión de los bits de la firma dentro del diseño. La primera de las estrategias, llamada EPIBF-LUT, introduce los bits de la firma digital en posiciones de memoria vacías de las LUTs empleadas por el sistema o diseño a proteger y que, por tanto, serán las posiciones de la firma. La segunda estrategia de protección, llamada EPBBF-LC, identifica bloques de bits de la firma con patrones de salida de la lógica combinacional incluida en el diseño. Los patrones de entrada correspondientes serán las posiciones de la firma. Las dos estrategias de protección hacen extremadamente difícil detectar y/o eliminar la protección ya que los bits de la firma forman parte de los recursos utilizados por el diseño antes de ser protegido.

Por otro lado, el método de protección incluye un mecanismo para la extracción de la firma, válido para las dos estrategias de protección, que será necesario utilizar cuando se quieran comprobar los derechos de autor sobre un determinado diseño. La extracción de la firma se activa con una secuencia de datos específica previamente

elegida, llamada secuencia de extracción de la firma. El uso de un algoritmo generador de secuencias pseudo aleatorio como el del LFSR aumenta la seguridad del método de extracción. Además de la secuencia de extracción de la firma, el método de extracción necesita que una lógica adicional sea incluida dentro del diseño, básicamente una máquina de estados finita y multiplexores. La función de esta lógica es detectar la secuencia de extracción de la firma, generar las posiciones de la firma y enrutar el contenido de estas posiciones, es decir, los bloques de bits de la firma, hacia la salida del sistema. En el caso de utilizar un LFSR para la generación de la secuencia de extracción de la firma, la lógica de extracción incluirá además un bloque LFSR y algunos otros recursos para realizar la detección de esta secuencia de extracción.

Por último, se ha presentado una herramienta automática desarrollada para la aplicación de la estrategia de protección EPBBF-LC. Con los últimos avances conseguidos, esta herramienta, además de automatizar el proceso de difusión, consigue una optimización de los recursos necesarios por la lógica de extracción. Esta optimización se ha basado en la minimización de una función de coste y la elección de un algoritmo que consiga realizar esta tarea sin que el gasto computacional sea excesivamente elevado.

Capítulo 5: Aplicación y evaluación de las estrategias de protección desarrolladas

En este capítulo se desarrollan módulos protegidos empleando el método de *watermarking* propuesto en este trabajo de investigación. En concreto, se utilizan sistemas basados en aritmética de residuos y sistemas que utilizan aritmética convencional para ejemplificar la aplicación de las estrategias de protección propuestas. Para evaluar estas estrategias de protección se presentan resultados experimentales obtenidos en términos del impacto sobre las prestaciones y el área de cada módulo.

5.1. Introducción

Para evaluar el método de protección propuesto se comienza con la exploración de las posibilidades que ofrecen los diseños que incluyen estructuras de memoria con posiciones de memoria vacías. Como ya se ha comentado en el capítulo 4, los circuitos basados en el RNS que se implementan sobre FPGAs suelen hacer un uso extensivo de la memoria embedida incluida en los dispositivos programables. Debido a la estructura de estos sistemas, algunas de las posiciones de memoria de estas tablas están vacías y se han utilizado para la aplicación de la estrategia de protección EPIBF-LUT. Por otro lado, las posiciones de memoria en las que hay un patrón de salida definido y, en general, toda la lógica combinacional empleada en los diseños, permiten la aplicación de la estrategia de protección EPBBF-LC.

Los diseños utilizados son un filtro CIC (*Cascade Integrator Comb*) [HOG81, GAR98b], un filtro FIR [RAM02a, RAM02b] y un diseño en aritmética convencional para la 1D-DWT (*Discrete Wavelet Transform*) [RAM03a, VER95]. De esta forma, se aprovecha la estructura de estos diseños para la aplicación de las estrategias de protección propuestas, permitiendo así que se puedan comprobar los derechos de autor y propiedad intelectual de los módulos protegidos. Los diferentes módulos desarrollados permiten ilustrar las diferentes opciones para la introducción y/o difusión de los bits de la firma, los mecanismos para la extracción de la misma y los resultados para firmas digitales de diferente longitud, con y sin códigos de corrección de errores. Mediante los resultados experimentales de síntesis se ha demostrado que las estructuras de protección desarrolladas, implementadas para tecnologías programables y ASIC, presentan un impacto mínimo sobre el área y prestaciones del sistema.

En las siguientes secciones se describirán los diseños que se han utilizado como ejemplo para la aplicación de las estrategias de protección EPIBF-LUT y EPBBF-LC y se detallarán los pasos seguidos para el desarrollo de los correspondientes módulos IP protegidos.

5.2. Filtros CIC de Hogenaour

Los filtros CIC [HOG81, GAR98b] han demostrado ser de gran utilidad como estructuras de interpolación o diezmado (*decimation*). Se caracterizan porque proporcionan respuestas de tipo FIR empleando sólo sumadores y elementos de retardo; aunque la respuesta en frecuencia fuera de la banda de paso no es suficiente para algunas aplicaciones, determinados filtros convencionales de baja velocidad pueden corregirlo. La función de transferencia de un sistema CIC de S etapas referida a la frecuencia de muestreo superior viene dada por:

$$H(z) = \left(\frac{1 - z^{-RD}}{1 - z^{-1}} \right)^{S} = \left(\sum_{k=0}^{RD-1} z^{-k} \right)^{S} \tag{5.1}$$

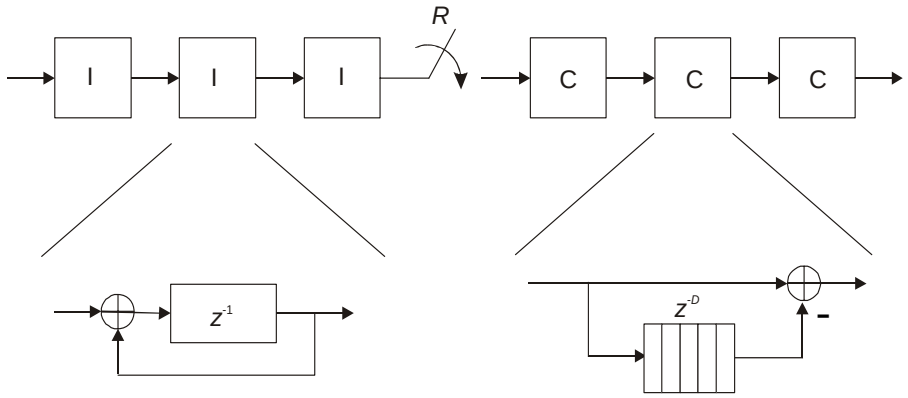

Figura 5.1. Filtro CIC de tres etapas para supresión.

La figura 5.1 muestra un filtro CIC de tres etapas para diezmado; consta de tres integradores en cascada, una reducción de la frecuencia de muestreo en un factor *R* y tres filtros peine; la estructura del filtro para interpolación es equivalente, salvo que las etapas peine se anteponen a la cascada de integradores, aunque estos siguen funcionando a una frecuencia de muestreo superior en un factor *R* a la de las etapas peine. De la ecuación (5.1) se deduce que *S* polos a frecuencia cero son anulados por *S* ceros. El resultado es que la función de transferencia es la misma que la de un filtro de media desplazada de *S* etapas. De este modo, un sistema CIC equivale a la sucesión de *S* filtros FIR idénticos, con la ventaja de que la arquitectura CIC no requiere multiplicadores.

Para un diseño efectivo de la estructura CIC es necesario calcular el máximo crecimiento de los datos. Se puede demostrar que para un filtro de tres etapas con *D*=2, *R*=32 y una entrada de 8 bits, un rango dinámico de 26 bits asegura que no se producirá desbordamiento durante el procesamiento.

5.2.1. Filtro CIC-RNS

El ejemplo de diseño seleccionado para incluir la firma digital consiste en un filtro CIC de tres etapas de diezmado [HOG81] basado en aritmética de residuos [MEY01b], con entrada de 8 bits, *R*=32, *D*=2 y salida de 10 bits. Teniendo en cuenta que el rango dinámico del filtro es de 26 bits, la reducción de la complejidad y la maximización de

las prestaciones, el RNS elegido está compuesto por cuatro canales de módulos {256, 63, 61, 59}. Este filtro CIC-RNS incluye una etapa de conversión binario-RNS, cuatro canales RNS CIC paralelos y una etapa final de conversión RNS-binario.

A fin de optimizar los procesos de conversión, tradicional cuello de botella en los sistemas basados en el RNS, la conversión de una entrada de B bits $x(n)$ al dominio del RNS se realiza a través de su descomposición en p bloques de b bits x_0, x_1, … y x_{p-1}, que son procesados mediante $p-1$ tablas [RAM02b]. La salida de estas tablas y el bloque x_0 se suman usando un árbol de sumadores, que proporciona el residuo correspondiente, mientras que una tabla final realiza la conversión final al dominio de índices. Por otro lado, la conversión final desde el RNS a binario se realiza con un bloque que implementa el algoritmo ε-CRT [GRI89a, GRI89b]. Este algoritmo evita el cuello de botella que normalmente supone la conversión de RNS a binario, ya que realiza en un solo paso conversión y escalado mapeando los residuos en un número en complemento a dos escalado. Además, el ε-CRT cuenta entre sus ventajas el que sólo requiere una tabla de consulta por canal y un árbol de sumadores binarios, que suma la salida de todas estas tablas. Es evidente, pues, que el ε-CRT es la mejor opción para la implementación de la conversión de RNS a binario en aplicaciones de procesamiento digital sobre dispositivos programables.

En el esquema del filtro CIC para supresión es necesaria una etapa de reducción de frecuencia en un factor R entre las etapas de integradores y peines, tal y como se muestra en la figura 5.2. En el filtro CIC que se va a utilizar para las simulaciones el valor elegido es R=32. Como ya se ha comentado, este filtro está compuesto por integradores y peines. Ambos elementos requieren un sumador módulo m para operar en aritmética de residuos. La operación suma módulo m se realiza de acuerdo con la ecuación (B.3) que se recoge en el apéndice B. La implementación de esta operación se puede realizar con un esquema basado en sumadores completos como el mostrado en la figura B.1(c). Para conseguir que además el funcionamiento del sistema sea totalmente segmentado, se pueden realizar unas ligeras modificaciones en el sumador de la figura B.1(c) y llegar al sumador segmentado de dos estados [MEY01].

La figura 5.3 muestra el esquema general del filtro CIC basado en RNS. Como ya se ha mencionado anteriormente, el primer canal es de 8 bits y los tres restantes de 6 bits, siendo los módulos de cada canal 255, 63, 61 y 59, respectivamente. Cada uno de

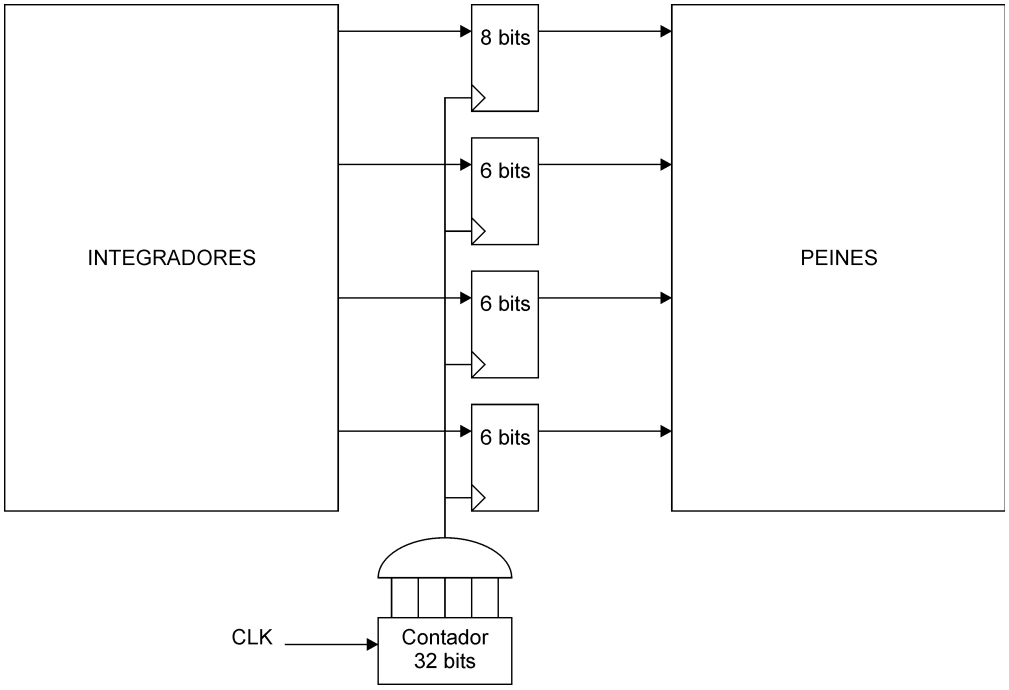

Figura 5.2. Esquema para la reducción de frecuencia en un factor 32.

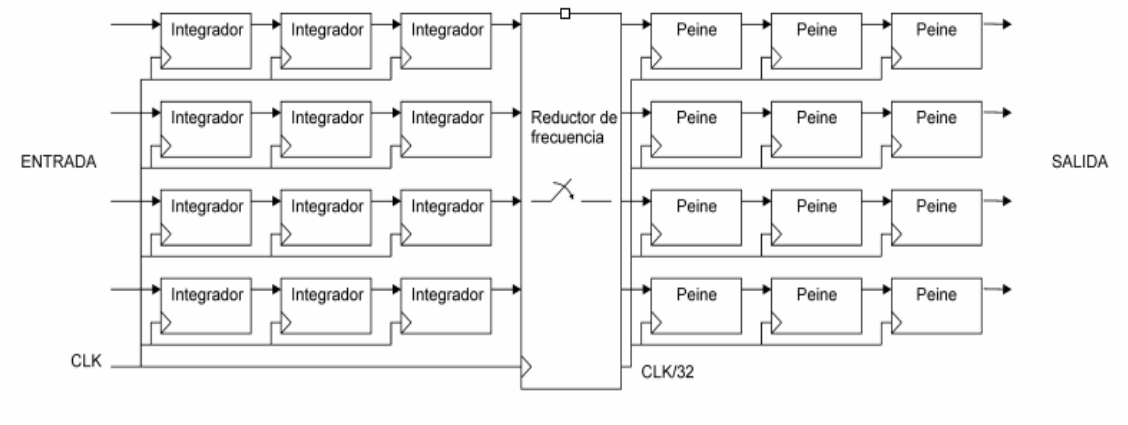

Figura 5.3. Esquema del filtro CIC con cuatro canales RNS.

los canales está compuesto por cuatro peines y cuatro integradores y la reducción en frecuencia se consigue con la ayuda de un contador. Como se ha comentado anteriormente, la conversión de entrada binario-RNS se realiza a través de la descomposición en bloques y la conversión RNS-binario se realiza mediante el algoritmo ε-CRT.

5.2.2. Preparación de la firma digital y proceso de difusión

Para este primer ejemplo se eligió la imagen de la figura 4.4.a) como firma que identifica al diseño, y se eligió el procedimiento más sencillo para la preparación de la firma digital, que consiste en la aplicación de la función *hash* MD5 a esta imagen para convertirla en la secuencia de bits: c2da0f795d03238fbd72ab3051000f81.

En la sección 4.3.2.1. se comentaron las distintas posiciones, dentro de los sistemas basados en el RNS, donde se podrían encontrar tablas de consulta con posiciones de memoria que no han sido utilizadas. Entre los distintos tipos de tablas de consulta que incluyen los sistemas basados en RNS, se distinguían tablas de consulta de la conversión de entrada, tablas de consulta del bloque de procesamiento de datos y tablas de consulta de la conversión de salida. Por ejemplo, la tabla 5.1 muestra las posiciones de memoria que se emplean para la conversión de salida de RNS a binario del filtro CIC-RNS bajo el algoritmo ε-CRT, con las posiciones ocupadas y las vacías para cada canal RNS. Cada una de estas posiciones tiene un ancho de 10 bits, y en total se dispone de 9 posiciones de memoria vacías de 10 bits. Tras el estudio de estas tablas de consulta, y teniendo en cuenta que la firma digital que se desea introducir en el

Canal	**Número de posiciones de memoria**	**Nº de posiciones de memoria ocupadas**	**Nºde posiciones de memoria vacías**	**Posiciones de memoria vacías**
256	$2^8 = 256$	256	0	- - - - - - - -
63	$2^6 = 64$	63	1	$111111111\big\vert_2 = 63$
61	$2^6 = 64$	61	3	$111111111\big\vert_2 = 63$ $111111110\big\vert_2 = 62$ $111111101\big\vert_2 = 61$
59	$2^6 = 64$	59	5	$111111111\big\vert_2 = 63$ $111111110\big\vert_2 = 62$ $111111101\big\vert_2 = 61$ $111111100\big\vert_2 = 60$ $111111011\big\vert_2 = 59$

Tabla 5.1. Análisis de las posiciones de memoria de las tablas que emplea el algoritmo ε-CRT.

diseño es de 128 bits, se aplicó una estrategia de protección mixta:

- la aplicación de la estrategia de protección EPIBF-LUT para incluir 90 bits de la firma digital en las posiciones de memoria que no han sido utilizadas. De esta forma, teniendo en cuenta la anchura de los datos en estas tablas de consulta, la firma digital se dividió en bloques de 10 bits. En la tabla 5.2 se muestran los bloques obtenidos a partir de la firma digital y la posiciones en las que se han introducido los nueve primeros;
- la aplicación de la estrategia de protección EPBBF-LC para la difusión de los 38 bits restantes de la firma digital. Estos 38 bits de la firma digital se completaron con dos ceros (estos bits aparecen en negrita en el último bloque de la tabla 5.2) para llegar hasta 40 bits, y se buscaron por grupos de 10 en posiciones de las tablas que ya estaban en uso. Las posiciones de memoria que resultaron de esta

Bloques de bits	**Estrategia de protección empleada**			
	EPIBF-LUT		**EPBBF-LC**	
	PMFs	**Canal**	**PMFs**	**Canal**
1100001011: 779	11111111	63	- - - - - - - -	- -
0110100000: 416	11111111	61	- - - - - - - -	- -
1111011110: 990	11111110	61	- - - - - - - -	- -
0101011101: 349	11111101	61	- - - - - - - -	- -
0000001100: 12	11111111	59	- - - - - - - -	- -
1000111000: 568	11111110	59	- - - - - - - -	- -
1111101111: 1007	11111101	59	- - - - - - - -	- -
0101110010: 370	11111100	59	- - - - - - - -	- -
1010101100: 648	11111001	59	- - - - - - - -	- -
1100000101: 773	- - - - - - - -	- -	111110	59
0001000000: 64	- - - - - - - -	- -	00010001	256
0000001111: 15	- - - - - - - -	- -	111111	59
1000000**100: 516**	- - - - - - - -	- -	00110010	256

Tabla 5.2. División en bloques de 10 bits y ubicación de éstos para la firma digital *c2da0f795d03238fbd72ab3051000f81* según la estrategia de protección mixta.

búsqueda, reflejadas en la tabla 5.2, se añadieron a las posiciones de la firma.

5.2.3. Preparación del diseño para la extracción de la firma

El siguiente paso en la aplicación de la estrategia de protección mixta consiste en la preparación del filtro CIC-RNS para la extracción de la firma. Para esto se debe realizar la elección o generación de la secuencia de extracción de la firma y el desarrollo de una lógica de extracción adecuada que formará parte del diseño. Como primer ejemplo, la secuencia de extracción de la firma elegida es 00000000, 00000001, 00000000, 00000001, 00000000, 00000001, que corresponde a la primera de las opciones propuestas en el capítulo 4 para la extracción de la firma.

Por otro lado, la lógica de extracción que se desarrolló está formada por una máquina de estados finita, que detecta la secuencia de extracción de la firma y genera las posiciones de memoria de la firma, y cuatro multiplexores de 2 a 1, que se utilizan para poder seleccionar las posiciones de la firma como señales de entrada de las tablas del conversor de salida. La figura 5.4 muestra la estructura básica del filtro CIC-RNS que incluye esta lógica de extracción.

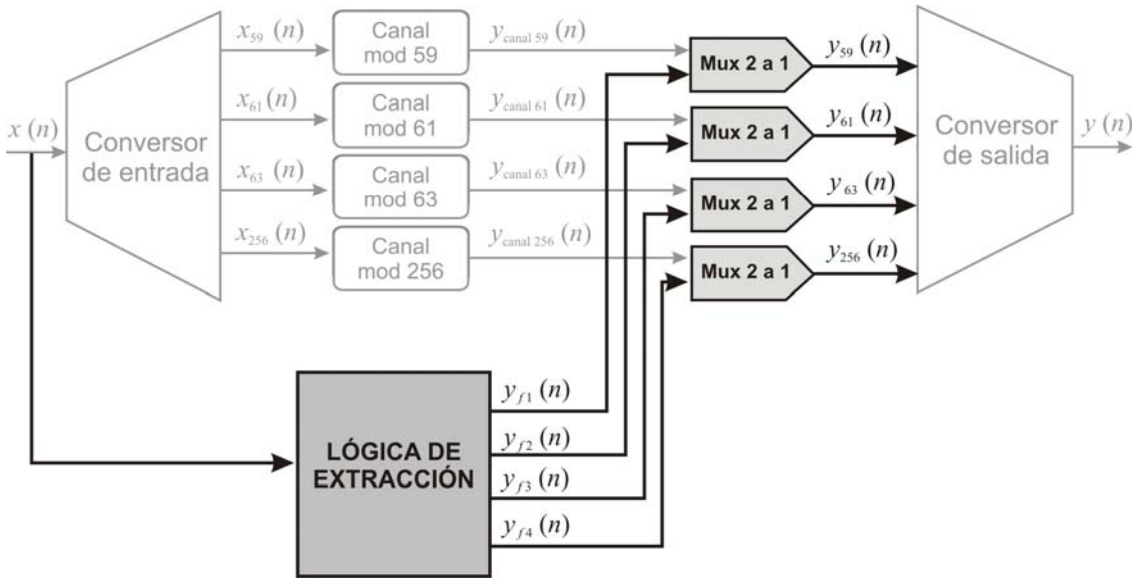

Figura 5.4. Diseño de 4 canales paralelos CIC-RNS con incorporación de la lógica de extracción.

La máquina de estados finita tiene como entrada la señal $x(n)$, que corresponde a la entrada del sistema, y como salidas las señales $y_{f59}, y_{f61}, y_{f63}, y_{f256}$ y la señal *enable*, como se puede ver en la figura 5.4. La señal *enable* es de un bit, y las señales $y_{f59}, y_{f61}, y_{f63}, y_{f256}$ tienen una longitud de 6 bits para los canales 59, 61 y 63, y 8 bits para el canal 256.

Esta máquina de estados parte de un estado inicial en el que la señal de salida *enable* está a "0" y el resto de las salidas también tienen todos sus bits a "0". Partiendo de este estado y tras detectar que se han introducido las muestras de la secuencia de extracción de la firma, se llega a un estado en el que se activa el proceso de extracción de la firma y la señal de salida *enable* pasa de "0" a "1". Sólo una del resto de las salidas, y_{fi}, cambia de cero a otro valor. Dependiendo del número de bloques de la firma, se sucederán una serie de estados en los que la señal *enable* seguirá activa, y al igual que en el estado anterior, sólo una de las salidas, que puede ser la misma del estado anterior u otra, tendrá un valor distinto de cero. Este valor distinto de cero para la salida y_{fi} corresponde a una de las posiciones de memoria de la firma o PMFs que se encuentra en la tabla de consulta del canal correspondiente. Tras llegar al último de los estados, con las salidas de la máquina de estados, se ha accedido a cada una de las PMFs.

La tabla 5.3 corresponde a la máquina de estados que interviene en el proceso de extracción de la firma para el filtro CIC-RNS considerado. En esta tabla se muestran todos lo estados posibles, 19 en total. En los seis primeros, del estado A al estado *G*, se detecta la secuencia de extracción de la firma y la transición entre estados dependerá del valor que tome la variable de entrada; todas las salidas tienen sus bits a 0. En los 13 estados restantes, del *H* al *S*, la transición entre estados ya no depende del valor que tome la variable de entrada, la señal *enable* está a "1" y las otras salidas van tomando valores que coinciden con las posiciones de memoria donde se encuentran los bloques de la firma.

Estado actual	Estado siguiente $x(n)$			Salidas				
	00000000	00000001	Otra entrada	*enable*	y_{f256}	y_{f63}	y_{f61}	y_{f59}
A	B	A	A	0	00000000	000000	000000	000000
B	B	C	A	0	00000000	000000	000000	000000
C	D	A	A	0	00000000	000000	000000	000000
D	B	E	A	0	00000000	000000	000000	000000
E	F	A	A	0	00000000	000000	000000	000000
F	B	G	A	0	00000000	000000	000000	000000
G	H	H	H	1	00000000	1111111	000000	000000
H	I	I	I	1	00000000	000000	111111	000000
I	J	J	J	1	00000000	000000	111110	000000
J	K	K	K	1	00000000	000000	111101	000000
K	L	L	L	1	00000000	000000	000000	111111
L	M	M	M	1	00000000	000000	000000	111110
M	N	N	N	1	00000000	000000	000000	111101
N	O	O	O	1	00000000	000000	000000	111100
O	P	P	P	1	00000000	000000	000000	111011
P	Q	Q	Q	1	00000000	000000	000000	111110
Q	R	R	R	1	00010001	000000	000000	000000
R	S	S	S	1	00000000	000000	000000	111111
S	A	A	A	1	00110010	000000	000000	000000

Tabla 5.3. Tabla de estados de la máquina de estados correspondiente a la lógica de extracción del filtro. CIC-RNS con canales 59, 61, 63, 256.

5.2.4. Proceso de extracción de la firma

Después de explicar el funcionamiento de la lógica que hace posible la extracción de la firma se mostrará el funcionamiento completo del circuito en este proceso de extracción, que comienza aplicando en la entrada del sistema la secuencia de extracción de la firma. Al detectar el circuito el último dato de esta secuencia, la máquina de estados activa la señal *enable*. Las etapas de conversión de entrada y salida y los canales RNS en todo momento siguen su funcionamiento normal, pero al activarse esta señal cada uno de los multiplexores conecta su salida, $y_{59}, y_{61}, y_{63}, y_{256}$, con la entrada que le

proporciona la máquina de estados, $y_{f59}, y_{f61}, y_{f63}, y_{f256}$, en lugar de con la entrada que proporciona cada uno de los canales RNS, $y_{C59}, y_{C61}, y_{C63}, y_{C256}$, que correspondería al procesamiento normal de cada residuo según el diseño considerado.

Teniendo en cuenta los valores que toman las señales de salida de la máquina de estados después de la detección de la SEF, indicadas en la tabla 5.3 como las salidas correspondientes a los estados del *G* al *S*, en cada ciclo de reloj siguiente a esta detección se empieza a acceder a cada una las posiciones de memoria de la firma. En concreto, en el ciclo de reloj siguiente a la detección de la SEF se accede a las posiciones de memoria que indica el estado *G*, es decir, a la posición de memoria "111111" = 63 de la tabla de consulta del canal 63, donde se encuentra el primer bloque de bits de la firma y, a las posiciones de memoria 000000, 000000, 00000000 de las tablas de consulta de los canales 59, 61 y 256, respectivamente, que contienen el 0.

Durante el proceso de extracción de la firma, la etapa de conversión de salida basada en el algoritmo ε-CRT sigue su funcionamiento normal, así que todas las salidas de estas tablas de consulta se suman con ayuda de un árbol de sumadores, tal y como se muestra en el esquema del algoritmo ε-CRT de la figura B.9. Al ser todas las salidas de las tablas de consulta 0 salvo la salida que corresponde a la posición de memoria de la firma, el árbol de sumadores da como salida el bloque de la firma contenido en esta posición de memoria. De esta forma, en cada ciclo de reloj siguiente a la detección de la SEF, en la salida va apareciendo cada uno de los bloques de bits que componen la firma. Al finalizar el proceso de extracción, la señal *enable* se desactivará y la salida del circuito corresponderá de nuevo a un procesamiento normal de los datos de entrada. El número de ciclos de reloj que se emplean en el proceso de extracción depende de la longitud de la secuencia de extracción y del número de bloques de la firma, que en este caso son 6 y 13, respectivamente.

5.2.5. Resultados experimentales

El diseño original del filtro CIC-RNS y el protegido fueron sintetizados desde sus descripciones VHDL usando dispositivos de la familia Virtex 2 de Xilinx [XIL] y de la familia APEX20KC de Altera [ALT]. Además, estos diseños también fueron

sintetizados para procesos ASICs *semi-custom* empleando las herramientas de Synopsys [WOL94].

En la tabla 5.4 se comparan los resultados de área y velocidad del filtro CIC con los resultados del filtro CIC protegido mediante la estrategia de protección mixta. La tabla muestra, para cada tecnología y herramienta utilizada, el área ocupada y la máxima frecuencia operativa. La tabla 5.4 sólo muestra los resultados de síntesis. En FPGAs en elementos lógicos ya que el resto de recursos (memorias embebidas, DSPs, etc.) es el mismo para todos los diseños. El análisis de estos resultados muestra que, para todas las implementaciones realizadas, el filtro protegido supone como máximo un aumento del área del 6.3% respecto del filtro original. Como se ha comentado anteriormente, este incremento de área no se debe al proceso de difusión de los bits de la firma, sino al *hardware* adicional que requiere el proceso de extracción. Sin embargo, el incremento de área que supone el procedimiento descrito es una cantidad fija, y en circuitos que ocupen mayor área, como puede ser el filtro FIR o la transformada *wavelet*, el área dedicada a la firma supondría un porcentaje menor que el obtenido en este ejemplo. Por otro lado, los datos de la tabla también reflejan que la penalización en las prestaciones es despreciable.

En la figura 5.5 se muestra un ejemplo de simulación para el filtro CIC RNS protegido. En esta figura se puede ver el proceso de extracción de la firma:

- en principio, la salida del sistema es 1022, que corresponde a un procesamiento normal de los datos de entrada;

	Xilinx		**Altera**		**Synopsys**	
Diseño original	SLICEs	F (MHz)	LEs	F (MHz)	Area	F (MHz)
CIC-filter	397	137.9	918	126.8	9321	21.55
Diseño protegido	Incr.área	Red. velocidad	Incr.área	Red. velocidad	Incr.área	Red. velocidad
CIC-FSM MD5	6.3 %	< 1%	5.7 %	< 1%	3.0 %	2.7 %

Tabla 5.4. Recursos y velocidad del filtro CIC-RNS original y del desarrallodado mediante la estrategia de protección mixta (EPIBF-LUT y EPIBBF-LC).

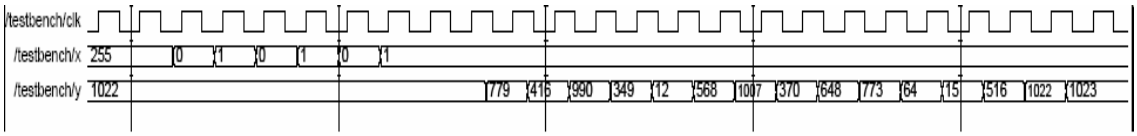

Figura 5.5. Proceso de extracción de la firma en el ejemplo de simulación.

- tras aplicar en la entrada del sistema *x* la secuencia de extracción de la firma 0, 1, 0, 1, 0, 1 (por comodidad, estos datos se muestran en decimal), en la salida, *y*, aparecen los bloques de bits de la firma (también en decimal), 779, 416, 990, 349, 12, 568, 1007, 370, 648, 773, 64, 15 y 516;
- después de que el proceso de extracción finalice, en la salida vuelven a aparecer los valores que corresponden a un procesamiento normal de los datos de entrada.

5.3. Filtros FIR

Los filtros digitales FIR o de respuesta impulsiva finita [RAM02a, RAM02b] han despertado un gran interés dado que representan de manera natural estructuras estables y son mucho menos sensibles a los errores de cuantización que los filtros de tipo recursivo. Tienen la gran ventaja de que pueden diseñarse para ser de fase lineal, lo cual hace que presenten ciertas propiedades en la simetría de los coeficientes. Este tipo de filtros son siempre estables y tienen especial interés en aplicaciones de audio. Su principal desventaja es que generalmente requieren un número elevado de términos no nulos para representar adecuadamente la respuesta en frecuencia del filtro, lo que exige el cálculo de un elevado número de sumas y productos durante el período de muestreo y limita seriamente el número de datos procesados por unidad de tiempo.

Un filtro FIR de longitud u orden N con entrada $x(n)$ y salida $y(n)$ se describe mediante una ecuación del tipo:

$$y(n) = c_0 x(n) + c_1 x(n-1) + \ldots + c_{N-1} x(n-N+1) = \sum_{k=0}^{N-1} c_k x(n-k) \qquad (5.2)$$

donde c_k son los coeficientes del filtro y $\{x(n)\}$ e $\{y(n)\}$ son las secuencias de entrada y salida del filtro, respectivamente. La figura 5.6 muestra la estructura básica de un FIR, que consiste en una serie de retardos, sumadores y multiplicadores. Pueden hacerse múltiples variaciones de esta estructura, por ejemplo situar filtros en serie, en cascada, etc. La figura 5.7 representa una variación de modelo FIR directo llamada filtro FIR traspuesto [MEY07] que, en general, representa la implementación preferida para un filtro FIR. Las ventajas de esta estructura son que no requieren un registro de desplazamiento adicional para $x(n)$ y no hay que emplear tampoco un etapa extra de segmentación de cauce en la suma de los productos para conseguir un alto rendimiento.

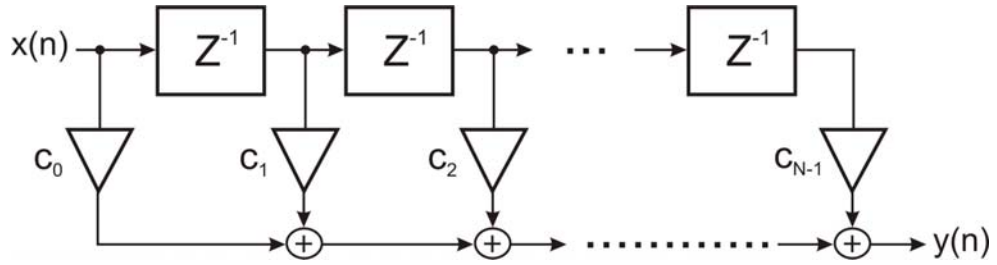

Figura 5.6. Forma directa para el filtro FIR.

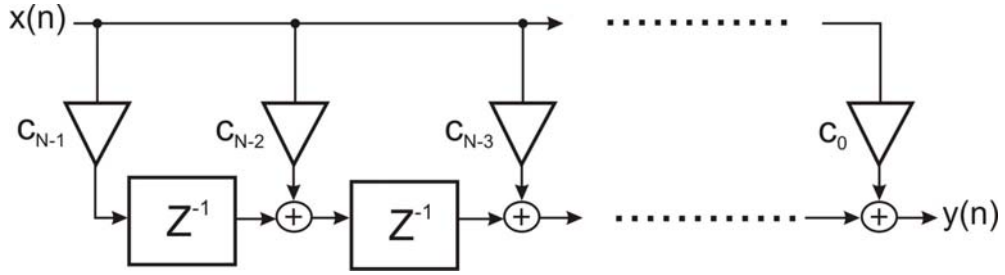

Figura 5.7. Forma traspuesta para el filtro FIR.

5.3.1. Filtro FIR-RNS basado en aritmética de índices

En la figura 5.8 se muestra la estructura interna básica de un filtro FIR basado en el RNS [RAM02b]. Este bloque está diseñado para trabajar externamente en complemento a dos [RAM03b], por lo que el filtro incluye una etapa de conversión de complemento a dos a RNS, L canales RNS paralelos y una etapa final de conversión del RNS a complemento a dos. La entrada *LOAD* habilita la actualización de los coeficientes. La conversión de entrada binario-RNS se realiza a través de la descomposición en bloques y la conversión RNS-binario se realiza con el bloque del algoritmo ε-CRT [GRI89a, GRI89b], del mismo modo que para el ejemplo anterior.

El diseño de un canal RNS para el filtro FIR se centraría en una correcta elección de los multiplicadores, puesto que de ella dependen la complejidad computacional y la velocidad. Los multiplicadores por cálculo de índices sobre cuerpos de Galois que se detallan en el Apéndice B resultan muy eficientes en el diseño del filtro digital considerado y, en general, en cualquier sistema de procesamiento de señales en el que los coeficientes no sean fijos y, por tanto, necesiten ser actualizados continuamente. En consecuencia, la figura 5.9 muestra la estructura básica de un canal para un filtro FIR de 8 etapas basado en aritmética de índices. La conversión de la señal $x(n)$ y de los coeficientes del filtro al dominio de los índices se realiza a la entrada del sistema. Por una parte, se realiza la conversión de binario a RNS a través de la conversión por bloques citada anteriormente [RAM02b] y, de otro lado, se realiza la conversión de estos residuos al dominio de los índices haciendo uso de tablas que almacenan las

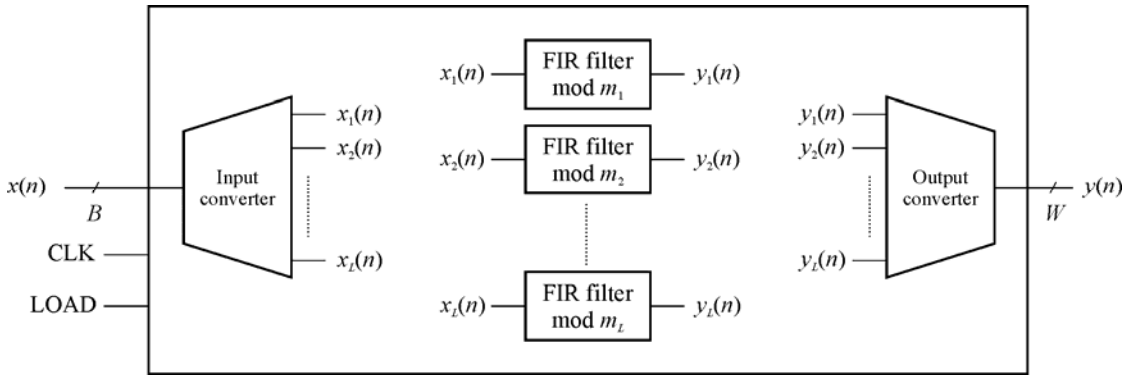

Figura 5.8. Diagrama de bloques de un filtro FIR reprogramable de L canales basado en el RNS.

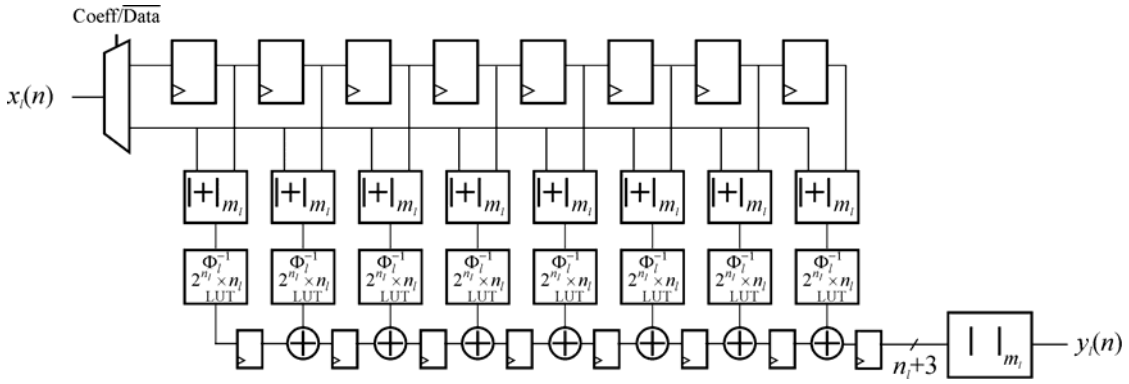

Figura 5.9. Estructura de un canal RNS de módulo primo de un filtro FIR de 8 coeficientes por medio de arimética de índices.

funciones correspondientes que asignan a cada entrada en el rango [1, m_l–1] su representación en el dominio de los índices en el intervalo [0, m_l–1]. En la figura 5.9 se puede comprobar que, al pasar al dominio de los índices, los multiplicadores necesarios en RNS se han convertido en sumadores módulo m_l–1 [JUL80], que han sido modificados para poder calcular productos por una entrada nula [GAR97] y así mantener una correcta segmentación de cauce. Una vez que cada producto ha sido calculado, las tablas correspondientes lo transforman desde el dominio de los índices al de residuos. La suma de productos se lleva a cabo a través de una cadena de sumadores módulo m_l optimizada, que consta de sumadores binarios convencionales, con precisión extendida, y una etapa final de reducción en módulo. Esto implica una importante reducción en área, especialmente a medida que aumenta el número de etapas del filtro, dado que es una estructura óptima para dispositivos programables frente a otra basada en sumadores CSA (*Carry Save Adder*), que sería la idónea para un ASIC.

En concreto, para la aplicación del método de protección, se realizó el diseño de un filtro FIR reconfigurable de 8 etapas, con entrada de datos de 16 bits y salida de datos de 16 bits. Teniendo en cuenta que el rango dinámico es de 34 bits, el conjunto de módulos elegido fue {11, 13, 17, 19, 23, 29, 31, 32} y el elemento generador depende de cada canal. A continuación se detallan los pasos que se siguieron para la aplicación del método de protección propuesto al filtro FIR basado en el RNS. En este ejemplo se emplearán firmas digitales de distinta longitud y las correspondientes firmas digitales obtenidas a partir de las anteriores, que incluyen corrección de errores. Por otro lado, se

aplicarán las dos estrategias de protección, EPBIF-LUT y EPBBF-LC, y además se utilizarán las dos opciones propuestas para la lógica de extracción de la firma que se detallaron en la sección 4.3.3. Como resultado de todo lo anterior, se obtendrán distintos diseños protegidos que servirán para comparar los recursos utilizados y las prestaciones con respecto al diseño original.

5.3.2. Preparación de las firmas digitales y proceso de difusión

En este ejemplo también se utilizó la primera de las imágenes de la figura 4.4 como firma que identifica al diseño. A esta firma se le aplicó la función MD5 para convertir la imagen en una firma digital de 128 bits correspondiente a la secuencia de cifras hexadecimales c2da0f795d03238fbd72ab3051000f81. Con el propósito de utilizar una firma de un número de bits mayor, también se utilizó la función *hash* SHA1 para convertir la misma imagen en una firma digital de 160 bits, correspondiente a la secuencia 8c0b64fbdd06e420aff99408e552fe1b121dee6c. En la aplicación de las estrategias de protección se utilizaron estas dos firmas digitales, y las mismas firmas digitales incluyendo corrección de errores en un bit. De esta forma, se generaron cuatro versiones distintas de cada uno de los diseños que se describirán más adelante. La nomenclatura utilizada para cada una de las firmas digitales es la siguiente:

- MD5 sin ECC: de 128 bits, obtenidos con MD5;
- MD5 1-bit ECC: de 136 bits, obtenidos con MD5 y con corrección de error en 1 bit;
- SHA1 sin ECC: de 160 bits, obtenidos con SHA1;
- SHA1 1-bit ECC: de 168 bits, obtenidos con SHA1 y con corrección de error en 1 bit.

En el filtro FIR-RNS propuesto se pueden encontrar posiciones de memoria vacías en los tres tipos de tablas que se distinguieron en la figura 4.7, de modo que es posible la aplicación de la estrategia de protección EPIBF-LUT. La tabla 5.5 resume las posiciones de memoria vacías dentro de las tablas de consulta empleadas para la conversión de índices a RNS y de RNS a binario del filtro FIR-RNS. Esta tabla muestra el número de bits de dirección empleados en las tablas correspondientes a cada canal,

Canal	Conversión de RNS a binario			Conversión de índices a RNS		
	Bits de dirección	Posiciones de memoria sin utilizar	Bits de memoria disponibles	Bits de dirección	Posiciones de memoria sin utilizar	Bits de memoria disponibles
32	5	--	--	5	31	1×5
31	5	31	1×16	5	[30,31]	2×5
29	5	[29,31]	3×16	5	[28,31]	4×5
23	5	[23,31]	9×16	5	[22,31]	10×5
19	5	[19,31]	13×16	5	[18,31]	14×5
17	5	[17,31]	15×16	4	--	--
13	4	[13, 15]	3×16	4	[12, 15]	4×4
11	4	[11, 15]	5×16	4	[10, 15]	6×4

Tabla 5.5. Posiciones de memoria sin utilizar y bits de memoria disponibles en de las tablas de consulta empleadas en la conversión de índices a RNS y de RNS a binario.

las posiciones de memoria sin utilizar, mostradas como intervalos, y los bits de memoria disponibles y que se podrían utilizar para la introducción de los bits de la firma. Para este último cálculo se ha tenido en cuenta que el ancho de palabra para todas las tablas de la conversión de RNS a binario es 16 bits y que para las tablas de la conversión de índices a RNS este ancho es de 5 bits para los canales 32, 31, 29, 23, 19 y 17, y de 4 bits para los canales módulo 13 y 11.

5.3.2.1. Aplicación de la estrategia de protección EPIBF-LUT

En primer lugar y para comparar resultados con los del ejemplo del filtro CIC-RNS, se utilizaron las tablas de la conversión RNS-binario para la aplicación de la estrategia de protección EPIBF-LUT al filtro FIR-RNS. Teniendo en cuenta que el número de bits de las firmas digitales que se quiere introducir dentro del diseño, como máximo 168 bits, se puede comprobar que bastaría sólo con utilizar la tabla de consulta de la conversión de RNS a binario del canal 19 ó 17, que tienen 208 y 240 bits de memoria sin utilizar, respectivamente. De esta forma, para el desarrollo de cada uno de los diseños firmados mediante la estrategia de protección EPIBF-LUT correspondientes a cada una de las firmas digitales, se emplearon las posiciones de memoria no utilizadas de la tabla de consulta correspondiente al canal 19, que forma parte de la etapa de

conversión de RNS a binario. Esta tabla de consulta tiene una anchura de 16 bits; por lo tanto, se utilizaron 8 posiciones de memoria para la firma de 128 bits, 9 posiciones de memoria para la de 136 bits, 10 posiciones de memoria para la de 160 bits y 11 posiciones de memoria para la de 168 bits.

Por otro lado, se desarrollaron también diseños firmados bajo la estrategia de protección EPIBF-LUT pero utilizando las posiciones de memoria libres de las tablas de consulta de la conversión de índices a RNS. En este caso no es posible introducir todos los bits de la firma en las posiciones de memoria libres de una sóla tabla de consulta, como se puede comprobar observando la columna correspondiente de la tabla 5.4. Sin embargo, para cada canal de la estructura propuesta del filtro FIR de 8 etapas se utilizan 8 tablas de consulta iguales para la conversión de índices a RNS. De esta forma, los bits de memoria sin utilizar dentro de cada canal se multiplican por 8, por lo que es posible introducir todos los bits de cualquiera de las firmas consideradas en las tablas de un mismo canal. Teniendo en cuenta una serie de factores como la anchura de los datos de salida del filtro (16 bits), la disponibilidad de posiciones de memoria en las tablas de conversión de índices a RNS de cada canal (mostrada en la tabla 5.4) y el número de bits de cada firma digital, se eligieron 4 de las 8 tablas de consulta del canal módulo 19 para introducir los bits de la firma. Los pasos que se siguieron para el desarrollo de los diseños firmados son los siguientes:

- teniendo en cuenta que los datos de salida del filtro son de 16 bits, cada firma digital se dividió en bloques de 16 bits;
- por otro lado, las tablas de conversión de índices a RNS del canal 19 tienen una anchura de 5 bits, por lo que cada bloque de 16 bits de la firma digital se volvió a dividir en tres bloques de 5 bits y un bloque de 1 bit;
- fue necesario, por tanto, utilizar cuatro tablas de consulta del canal 19; de tres de ellas se utilizaron los 5 bits de cada posición de memoria, y de la cuarta tabla se utilizó sólo uno de los bits de memoria;
- el número de posiciones de memoria de cada una de las cuatro tablas dependió de la longitud de las firmas digitales consideradas; se necesitaron 8 posiciones de memoria para la firma de 128 bits, 9 para la de 136 bits, 10 para la de 160 bits y 11 para la de 168 bits. Se puede comprobar que, para todos los casos, el número

de posiciones de memoria necesarias es inferior al número de posiciones de memoria vacías de las tablas de conversión índices-RNS del canal 19 (tabla 5.4);

- añadiendo la lógica de extracción, cuando se activa el proceso de extracción de la firma y durante un número de ciclos determinado, se accede simultáneamente a los bits de memoria de cada una de las cuatro tablas correspondientes a los bloques de bits de la firma, se concatenan estos bloques en el orden adecuado, y se dirije esta información hacia la salida del sistema para obtener así los bloques de 16 bits de la firma.

5.3.2.2. Aplicación de la estrategia de protección EPBBF-LC

El filtro FIR-RNS descrito también se utilizó para la aplicación de la estrategia de protección EPBBF-LC. Esta estrategia de protección (*v.* capítulo 4) consiste en la difusión de los bits de la firma a través de patrones de entrada de estructuras de memoria o, en general, de la lógica combinacional del diseño. En concreto, para la aplicación de esta estrategia de protección al filtro FIR-RNS, de entre los tres tipos de LUTs diferentes, se consideraron las LUTs utilizadas en la conversión de RNS a binario. El primer paso a seguir consiste en la división de la firma digital en bloques de bits. La elección del tamaño de bloque es una decisión importante teniendo en cuenta que de esto depende que la aplicación de la estrategia sea o no posible. Cuanto menor sea el tamaño del bloque, mayor es la probabilidad de identificar cada uno de los bloques de la firma con los patrones de salida de las estructuras de memoria, pero mayor es también la cantidad de recursos necesarios para la lógica de extracción. La expresión (4.2), que se obtuvo anteriormente para calcular esta probabilidad, se ha utilizado para realizar un estudio para los tamaños de bloque de las firmas digitales MD5 y SHA1, teniendo en cuenta la capacidad de las estructuras de memoria del filtro FIR-RNS. La figura 5.10 muestra la probabilidad de encontrar los bloques de bits de las firmas digitales de 128 y 160 bits en las tablas empleadas por el algoritmo ε-CRT para la conversión RNS-binario del diseño. En esta figura se han considerado distintos tamaños de bloque para cada una de las firmas digitales, desde 4 hasta 16 bits. Se puede apreciar que el tamaño más grande del bloque elegido coincide con el número de bits del contenido de estas estructuras de memoria, en este caso 16 bits, y que a medida que

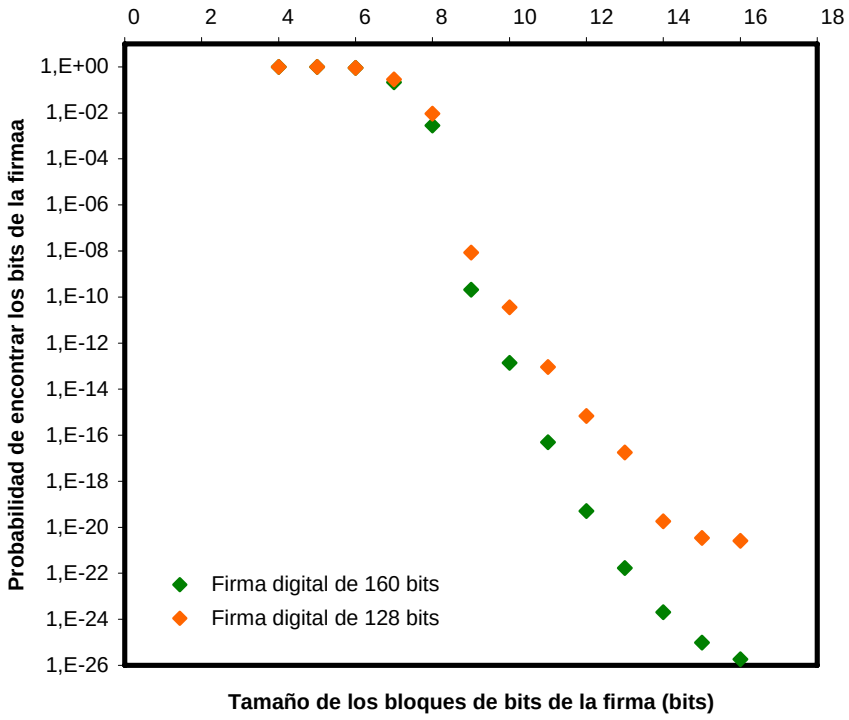

Figura 5.10. Probabilidad de encontrar los bloques de bits de la firma dentro de las tablas de consulta de la conversión de RNS a binario del filtro FIR-RNS.

disminuye este tamaño, la probabilidad de encontrar todos los bloques de bits de la firma dentro del diseño aumenta.

Los datos reflejados en esta figura muestran que para tamaños de bloque mayores que 7 la probabilidad de que la búsqueda o identificación de los bloques de bits de la firma tenga éxito es muy baja. Tras analizar estos datos, el procedimiento que se ha seguido para aplicar la estrategia EPBBF-LC al filtro FIR-RNS ha sido el siguiente:

- se ha elegido un tamaño de bloque igual a 5;
- las firmas digitales se han dividido en bloques de 5 bits;
- para cada firma digital, se ha conseguido identificar todos sus bloques de bits con los últimos 5 bits de determinados patrones de salida de la tabla de consulta del ε-CRT correspondiente al canal 32. El hecho de poder encontrar todos los bloques de bits de la firma dentro de una sola de las tablas de consulta y en la misma

posición (en este caso, en los últimos 5 bits) ha permitido minimizar los recursos necesarios por la lógica de extracción;

- los patrones de entrada correspondientes a cada uno de estos patrones de salida pasan a formar parte de las posiciones de la firma.

Es importante tener en cuenta que, para el filtro FIR-RNS considerado, hubiese sido posible elegir otro tamaño de bloque mayor para intentar así disminuir el número de recursos necesarios por la lógica de extracción. Sin embargo, se comprobó que 5 es el mayor tamaño de bloque posible que permite la identificación completa de los correspondientes bloques de bits.

5.3.3. Preparación de los diseños para la extracción de la firma

El siguiente paso tras la introducción o difusión de los bits de la firma es la preparación de los diseños para la extracción. Se utilizaron las dos lógicas de extracción propuestas en el capítulo 4, que corresponden a la elección manual y a la generación por medio de un LFSR de la secuencia de extracción de la firma. A continuación se van a considerar las dos lógicas de extracción propuestas y se va a particularizar su aplicación y funcionamiento para cada uno de los diseños en los que se van a utilizar.

5.3.3.1. Lógica de extracción: FSM y multiplexores

En primer lugar se va a considerar la lógica de extracción correspondiente a una elección manual de la SEF. Al igual que para la lógica de extración del ejemplo del filtro CIC-RNS, la secuencia de extracción de la firma elegida fue 0000_H, 0001_H, 0000_H, 0001_H, 0000_H, 0001_H. La preparación de los diseños para la extracción de la firma es muy similar a la que se describió para los diseños protegidos del ejemplo del filtro CIC-RNS. Aunque no se va a realizar una descripción tan detallada de la máquina de estados como se hizo en el ejemplo de filtro CIC-RNS, conviene particularizar la lógica de extracción desarrollada para los diseños considerados.

a) Diseños EPBIF-LUT que utilizan las tablas del ε-CRT

Se recuerda que en los diseños firmados bajo la estrategia EPBIF-LUT que utilizan las tablas del ε-CRT, los bloques de bits de la firma se han introducido sólo en la tabla de consulta del ε-CRT del canal módulo 19. La lógica de extracción para estos diseños está formada por una máquina de estados, que detecta la SEF y genera las posiciones de memoria de la firma, y ocho multiplexores de 2 a 1, que se utilizan para seleccionar las posiciones de la firma como señales de entrada de las tablas del conversor de salida. La estructura del diseño con la incorporación de la lógica de extracción es muy similar a la de la figura 5.6, pero, debido a que el diseño del filtro FIR-RNS tiene 8 canales en lugar de los 4 del filtro CIC-RNS, la FSM ha de generar 9 salidas en lugar de 5 y el número de multiplexores ha de incrementarse hasta 8.

La máquina de estados tiene como entrada *x*(*n*), que es la entrada del sistema, y como salidas las señales $y_{f11}, y_{f13}, y_{f17}, y_{f19}, y_{f23}, y_{f29,}\ y_{f31}, y_{f32}$, con anchura igual a la del residuo del canal RNS indicado por el subíndice, y la señal de un bit *enable*. Esta máquina de estados parte de uno inicial en el que todas las señales de salida (*enable* incluída) están a cero. Partiendo de este estado y tras detectar que por la señal de entrada se ha introducido la SEF, se llega a un estado en el que se activa el proceso de extracción de la firma, cambiando la señal de salida *enable* de “0” a “1”. A partir de este momento la máquina de estados ha de empezar a generar las posiciones de memoria de la firma. De esta forma, y_{f19} cambia de todo ceros al valor correspondiente a la primera de las PMFs. El resto de salidas, $y_{f11}, y_{f13}, y_{f17}, y_{f23}, y_{f29}, y_{f31}, y_{f32}$ permanecen a cero. Dependiendo del número de bloques de la firma, se sucederán una serie de estados en los que la señal *enable* está activa, la señal y_{f19} toma valores que corresponden a las PMFs y el resto de señales de salida de la máquina de estados tienen todos sus bits a 0.

Mientras la señal *enable* esté a “1”, cada multiplexor conecta su salida con la entrada correspondiente a las señales que genera la máquina de estados, en lugar de conectar con la entrada que proporciona cada uno de los canales RNS, que correspondería al procesamiento normal de cada residuo según el diseño considerado. La salida de cada multiplexor se conecta con la entrada correspondiente al bloque de conversión de algoritmo ε-CRT, que está compuesto por 8 tablas de consulta, una para cada canal, y un árbol de sumadores. De esta forma, durante la extracción de la firma, se

accede a las posiciones de memoria todo ceros de las tablas de consulta de los canales 11, 13, 17, 23, 29, 31 y 32, y a las posiciones de memoria de la tabla de consulta del canal 19 correspondientes a las PMFs. Finalmente, teniendo en cuenta que la dirección cero de las tablas de consulta del algoritmo ε-CRT contienen todo ceros, en cada ciclo de reloj el árbol de sumadores recibe una entrada con el contenido de una de las PMFs, es decir, un bloque de bits de la firma, y un conjunto de señales nulas, permitiendo así que cada bloque de bits de la firma se dirija hacia la salida del sistema. Es importante destacar que la máquina de estados fue diseñada de la forma descrita para que se permita extraer la firma sin que la etapa de conversión de salida tenga que sufrir modificación alguna: durante el proceso de extracción de la firma funciona igual que durante el procesamiento de los datos de entrada.

b) Diseños EPBBF-LC que utilizan las tablas del ε-CRT

En los diseños firmados bajo la estrategia EPBBF-LC que utilizan las tablas del ε-CRT, los bloques de bits de la firma se han difundido sólo en la tabla de consulta del ε-CRT del canal 32. La lógica de extracción para estos diseños también esta formada por una máquina de estados y ocho multiplexores de 2 a 1. La máquina de estados tiene como entrada la señal de entrada del sistema, y como salidas el conjunto de señales $y_{f11}, y_{f13}, y_{f17}, y_{f19}\, y_{f23}\, y_{f29}\, y_{f31}\, y_{f32}$ y la señal *enable*. Cuando se activa el proceso de extracción de la firma, la salida de la máquina de estados y_{f32} cambia de todo ceros al valor correspondiente a la primera de las PMFs, mientras que las salidas, $y_{f11}, y_{f13}, y_{f17}, y_{f19}, y_{f23}\, y_{f29}\, y_{f31}$ permanecen con todos sus bits a 0. Al igual que en los diseños anteriores, dependiendo del número de bloques de la firma, se sucederán una serie de estados en los que la señal *enable* seguirá activa y la señal y_{32} irá tomando valores que corresponden a las PMFs, direccionando así cada una de éstas.

No se va a seguir describiendo el resto de la lógica de extracción ni su funcionamiento ya que es muy similar a la desarrollada para los diseños EPBIF-LUT que utilizan las tablas del ε-CRT. Las diferencias más destacables de la lógica de extracción de los diseños EPBBF-LC con respecto a la de los diseños EPBIF-LUT que utilizan las tablas del ε-CRT son las siguientes:

- los bloques de la firma son de 5 bits en lugar de 16; esto supone para cada firma un número mayor de bloques, una máquina de estados con más estados y, por tanto, más recursos para la lógica de extracción;
- las PMFs se encuentran en la tabla de consulta del ε-CRT del canal 32 en lugar de en la tabla de consulta del ε-CRT del canal 19.

5.3.3.2. Lógica de extracción basada en LFSR

La segunda opción para la lógica de extracción considera que la secuencia de extracción de la firma ha sido generada con la ayuda de un LFSR. Teniendo en cuenta que la anchura de los datos de entrada de los diseños considerados es 16, se utilizó un LFSR de longitud 16 para generar los 100 datos que forman la secuencia de extracción. Es importante tener en cuenta que la generación de la secuencia de extracción de la firma mediante el LFSR se realiza de forma independiente a la descripción del diseño. De esta forma, se puede utilizar cualquier *software* y lenguaje de programación (Matlab, C, etc.) para describir el LFSR y generar los 100 datos de la secuencia de extracción. Como se estudiará a continuación, sera la lógica de extracción de la firma la que necesite incluir la descripción VHDL del LFSR basado en el mismo algoritmo que se utilizó para generar la secuencia de extracción de la firma.

Teniendo en cuenta la forma en la que se ha generado la secuencia de extracción, la lógica de extracción esta compuesta por un LFSR, un comparador, un contador, una máquina de estados y multiplexores, como se motró en la figura 4.13. El LFSR, el comparador y el contador tienen la función de detectar la SEF y activar el proceso de extracción. Una vez activado, la máquina de estados genera como señales de salida las posiciones de la firma para tener acceso así a su contenido, es decir, a los bloques de bits de la firma. Los multiplexores dirigen los bloques de la firma hacia la salida del sistema.

El funcionamiento del LFSR, el comparador y el contador, detallado en la sección 4.3.3.2, es el mismo para todos los diseños que incluyen esta lógica de extracción:

- el LFSR incluido en la lógica de extracción genera el primer dato de la SEF;

- el comparador comprueba si este dato coincide con el dato que llega por la entrada de datos del sistema;
- si los datos coinciden, el contador, que se encuentra inicialmente a 0, aumenta su cuenta, y el LFSR genera el siguiente dato de la SEF; este dato se comparará con el nuevo dato de entrada, y así sucesivamente mientras los datos coincidan;
- si en algún momento los datos no coinciden, se generan señales de control que hacen que el contador se ponga a 0 y que el LFSR se inicialice;
- cuando se hayan aplicado en la entrada del sistema sucesivamente todos los datos que componen la SEF y en el orden adecuado, el contador habrá alcanzado el valor 100, que, mediante la activación de una señal de control, llamada *en_FSM*, le indicará a la máquina de estados que se ha activado el proceso de extracción de la firma.

Por otro lado, la máquina de estados y los multiplexores incluidos en la lógica de extracción sí que dependerán de la estrategia de protección con la que los diseños se han firmado. Además de la introducción de la lógica de extracción, en los diseños firmados mediante la estrategia de protección EPIBF-LUT que hacen uso de las tablas de conversión de índices a RNS se deben realizar algunas modificaciones que permitan la extracción de la firma cuando sea necesario. A continuación se describe el resto de la lógica de extracción para cada uno de los diseños:

a) Diseños EPBIF-LUT que utilizan LUTs del ε-CRT y diseños EPBBF-LC

Las máquinas de estados para estos diseños son muy similares a las máquinas de estados que forman parte de los diseños protegidos desarrollados con la primera opción para la lógica de extracción. Sin embargo, las máquinas de estados consideradas aquí no incluyen la lógica necesaria para la detección de la SEF, ya que esta detección se realiza con ayuda del LFSR, el comparador y el contador.

De esta forma, las máquinas de estado consideradas para estos diseños tienen como entrada la señal de control que genera el comparador, *en_FSM*, y como salidas el conjunto de señales $y_{f11}, y_{f13}, y_{f17}, y_{f19}, y_{f23}, y_{f29}, y_{f31}, y_{f32}$ y la señal *en_test.* Si no se ha detectado la SEF, la señal de control *en_FSM* permanece inactiva y la máquina de estados se sitúa en un estado de espera en el que todas las señales de salida tienen todos

sus bits a 0 y la señal *en_test* está inactiva. Por otro lado, si se detecta la SEF, la señal de control se activa, indicándole así a la máquina de estados que el proceso de extracción de la firma debe comenzar. En este caso, la máquina de estados pasa por una serie de estados en los que la señal *en_test* permanece activa y se van generando cada una de las PMFs. Teniendo en cuenta que los bits fueron introducidos en la tabla de consulta del canal módulo 19, en cada estado la señal de salida y_{19} de la máquina de estados para cada uno de estos diseños va tomando los valores correspondientes a las PMFs, y el resto de señales de salida siguen con todos sus bits a 0. Por otro lado, teniendo en cuenta que en los diseños firmados mediante la estrategia de protección EPBBF-LC los bits fueron difundidos en la tabla de consulta del canal 32, la señal de salida y_{32} de la máquina de estados es la que toma los valores correspondientes a las PMFs, y el resto de señales permanecen con todos sus bits a 0.

Se necesitan ocho multiplexores de 2 a 1 controlados por la señal *en_test*. Cuando se inicie el proceso de extracción de la firma, la máquina de estados activa la señal *en_test* y cada uno de los multiplexores conecta su salida con la entrada correspondiente a las señales que genera la máquina de estados, en lugar de conectar la entrada que proporciona cada uno de los canales RNS, que correspondería al procesamiento normal de cada residuo según el diseño considerado. La forma en la que los multiplexores enrutan las señales, se accede al contenido de las PMFs y se dirigen los bloques de bits de la firma hacia la salida, es igual a la descrita en el apartado correspondiente a la primera de las lógicas de extracción.

b) Diseños EPIBF-LUT que utilizan las tablas de la conversión índices-RNS

En estos diseños los bloques de la firma se han difundido en 4 de las 8 tablas de consultas correspondientes a la conversión de índices a RNS del canal módulo 19. Para la extracción de la firma, dentro de estos diseños se deben incluir los recursos necesarios teniendo en cuenta la opción para la lógica de extracción considerada. En este caso, la lógica de extracción está formada por los bloques LFSR, el comparador y el contador descritos anteriormente, y la máquina de estados y el multiplexor que se detallan a continuación. Además de esta lógica, la preparación del diseño para la extracción de la firma requiere algunas modificaciones en el bloque de procesamiento del filtro FIR-RNS del canal módulo 19.

La FSM para estos diseños tienen como entrada la señal de control que genera el comparador, *en_FSM*, y como salidas una señal llamada *y_machine_19* y la señal *en_test.* El multiplexor de 2 a 1 estará controlado por la señal de salida de la FSM *en_test*. Las modificaciones consisten en incluir en este bloque dos señales de entrada más (una señal de control que se conecta con la señal de salida *en_test* de la FSM, y una señal que se conecta con la señal de salida *y_machine*_19 de la FSM), y desarrollar e introducir una lógica de control para permitir el acceso a las posiciones de memoria de las cuatro tablas de consulta de la conversión de índices a RNS en la que se introdujeron los BBFs.

Tras estas modificaciones, el funcionamiento del sistema sería el siguiente:

- si no se ha detectado la SEF, la señal de control *en_FSM* permanece inactiva y la FSM se sitúa en un estado de espera en el que la señal de salida tiene todos sus bits a 0 y la señal *en_test* está inactiva. El bloque de procesamiento del canal módulo 19 opera de forma normal y el multiplexor de salida conecta la salida del bloque de conversión ε-CRT con la salida del diseño correspondiente;
- si se detecta la SEF, la señal de control se activa, indicando que el proceso de extracción de la firma debe comenzar. La FSM pasa por una serie de estados en los que la señal *en_test* permanece activa y la señal *y_machine_19* va tomando valores iguales a las PMFs. Al estar activa la señal de control *en_test*, la lógica de control del bloque de procesamiento del canal módulo19 de estos diseños conecta la señal *y_machine_19* con la entrada de las 4 tablas de conversión de índices a binario en las que se encuentran los bloques de bits de la firma. De esta forma se va accediendo a cada una de las PMFs y su contenido, es decir, los BBFs se dirigen hacia la salida de este bloque. Por último, al estar activa la señal *en_test*, el multiplexor de salida conecta la salida del bloque de procesamiento del canal 19 con la salida global de estos diseños.

5.3.4. Diseños firmados

Teniendo en cuenta las dos opciones para la introducción de los bits de la firma y las dos lógicas para la extracción de la firma, los diseños desarrollados para el filtro FIR-RNS dentro de la estrategia de protección EPIBF-LUT fueron los siguientes:

- *FIR-FSM*: este esquema de protección incluye una FSM que activa el proceso de extracción de los bits de la firma, estando estos bits incluidos en las posiciones de memoria libres de las tablas del ε-CRT;
- *FIR-LFSR*: el *hardware* que interviene en la extracción incluye un LFSR puesto que éste fue utilizado para generar la secuencia de extracción de la firma. Como en el anterior, la firma se introduce en las tablas del ε-CRT;
- *FIR-Embedded-LFSR*: tiene el mismo *hardware* de extracción de la firma que el anterior; sin embargo, son las tablas de la conversión índices-RNS las que contienen los bits de la firma.

Por otro lado, teniendo en cuenta las dos diferentes lógicas para la extracción de la firma, los diseños desarrollados mediante la aplicación de la estrategia de protección EPBBF-LC fueron *FIR-FSM* y *FIR-LFSR*.

Como ya se mencionó anteriormente, se han generado cuatro versiones distintas de estos diseños, que corresponden al desarrollo de los mismos para cada una de las cuatro firmas digitales consideradas.

5.3.5. Resultados experimentales

A fin de ilustrar los resultados obtenidos, se ha implementado el filtro FIR-RNS original y los distintos diseños protegidos usando para ello dispositivos de la familia Virtex 2 de Xilinx [XIL08] y de la familia APEX20KC de Altera [ALT08]. Al igual que para el ejemplo del filtro CIC, estos diseños también fueron sintetizados para procesos ASICs *semi-custom* empleando las herramientas de Synopsys [RAB04].

En las tablas 5.6 y 5.7 se comparan los resultados de área y velocidad de filtro FIR con los resultados obtenidos para los filtros protegidos mediante la estrategia de protección EPIBF-LUT y para los filtros protegidos mediante la estrategia de protección EPBBF-LC, respectivamente. En concreto, en la tabla 5.6 se compara el filtro original con los diseños *FIR-FSM*, *FIR-LFSR* y *FIR-embedded-LFSR* protegidos mediante la estrategia EPIBF-LUT. Por otro lado, en la tabla 5.7 se compara el filtro original con los diseños *FIR-FSM* y *FIR-LFSR* firmados mediante la estrategia de protección EPBBF-LC.

			Xilinx		Altera		Synopsys	
Diseño original			SLICEs	F (MHz)	LEs	F (MHz)	Area	F (MHz)
Filtro FIR-RNS			1642	151.6	3697	192.49	43688	50.0
Diseños protegidos			Incr. área	Red. velocidad	Incr. área	Red. velocidad	Incr. área	Red. velocidad
FIR-FSM	MD5	sin ECC	2.5 %	< 1%	1.8 %	< 1%	1.6 %	< 1%
		1-bit ECC	2.6 %	< 1%	2.0 %	< 1%	1.6 %	< 1%
	SHA1	sin ECC	2.4 %	< 1%	2.2 %	< 1%	1.7 %	< 1%
		1-bit ECC	2.6 %	< 1%	2.0 %	< 1%	1.8 %	< 1%
FIR-LFSR	MD5	sin ECC	2.7 %	< 1%	2.9 %	< 1%	3.4 %	< 1%
		1-bit ECC	3.2 %	< 1%	3.0 %	< 1%	3.4 %	< 1%
	SHA1	sin ECC	3.2 %	< 1%	3.1 %	< 1%	3.5 %	< 1%
		1-bit ECC	3.5 %	< 1%	3.1 %	< 1%	3.5 %	< 1%
FIR-embedded LFSR	MD5	sin ECC	6.5 %	< 1%	4.6 %	< 1%	3.8 %	< 1%
		1-bit ECC	6.6 %	< 1%	4.6 %	< 1%	3.9 %	< 1%
	SHA1	sin ECC	6.4 %	< 1%	4.8 %	< 1%	3.9 %	< 1%
		1-bit ECC	6.2 %	< 1%	4.6 %	< 1%	4.1 %	< 1%

Tabla 5.6. Recursos y velocidad del filtro FIR-RNS y de los diseños protegidos mediante la estrategia EPIBF-LUT.

			Xilinx		Altera		Synopsys	
Diseño original			SLICEs	F (MHz)	LEs	F (MHz)	Area	F (MHz)
Filtro FIR-RNS			1642	151.6	3697	192.49	43688	50.0
Diseños protegidos			Incr. área	Red. velocidad	Incr. área	Red. velocidad	Incr. área	Red. velocidad
FIR-FSM	MD5	sin ECC	3.4 %	< 1%	2.6 %	< 1%	1.8 %	< 1%
		1-bit ECC	3.5 %	< 1%	3.0 %	< 1%	1.9 %	< 1%
	SHA1	sin ECC	3.8 %	< 1%	3.6 %	< 1%	2.0 %	< 1%
		1-bit ECC	3.9 %	< 1%	3.6 %	< 1%	2.0 %	< 1%
FIR-LFSR	MD5	sin ECC	3.8 %	< 1%	3.4 %	< 1%	3.6 %	< 1%
		1-bit ECC	4.0%	< 1%	3.5 %	< 1%	3.6 %	< 1%
	SHA1	sin ECC	4.4 %	< 1%	4.0 %	< 1%	3.7 %	< 1%
		1-bit ECC	4.4 %	< 1%	4.2 %	< 1%	3.8 %	< 1%

Tabla 5.7. Recursos y velocidad del filtro FIR-RNS y de los diseños protegidos mediante la estrategia EPBBF-LC.

Para todos los diseños se muestran los resultados obtenidos considerando las cuatro firmas digitales. Las tablas muestran el incremento de área y la reducción en la velocidad respecto del diseño original para cada uno de los diseños protegidos. Un análisis detallado de los resultados muestra que los filtros protegidos mediante la estrategia de protección EPIBF-LC ocupan un área nunca mayor al 6.6% con respecto al filtro original, mientras que para los diseños protegidos mediante la estrategia EPBBF-LC, el incremento del área no supera el 4.4 %. Por otro lado, la reducción en velocidad es despreciable para todos los diseños protegidos.

En lo que respecta al diseño *FIR-FSM* protegido mediante EPIBF-LUT empleando la firma digital MD5 sin códigos de corrección de errores en relación con el diseño CIC-RNS protegido mediante la misma estrategia y empleando la misma firma digital, si se comparan los resultados obtenidos que han quedado reflejados en las tablas 5.7 y 5.4, se puede comprobar que el incremento del área es menor para el diseño protegido que ocupa mayor área. Este resultado era predecible teniendo en cuenta que, para una firma digital de un determinado tamaño, el número de recursos adicionales necesarios por la lógica de extracción es fijo (128 bits en el caso considerado).

5.4. Transformada 1-D DWT

Durante la últimos años, la transformada *wavelet* [RAM03a, VER95] ha demostrado ser de gran utilidad como herramienta de procesamiento. Las aplicaciones de la transformada *wavelet* son amplias e incluyen la compresión de señales, el análisis estadístico, el filtrado adaptativo o la resolución de ecuaciones lineales. El interés en la transformada *wavelet* ha aumentado conforme las aplicaciones se han hecho más numerosas y ésta se está convirtiendo en parte de los estándares de compresión de imagen y video. Por ejemplo, en los últimos años se han desarollado e implementado robustos y sofisticados esquemas de compresión de imágenes basados en *wavelets* que han hecho que el estándar JPEG 2000 [ISO97] utilice la transformada *wavelet* como núcleo de su esquema de compresión.

La transformada *wavelet* se construye mediante funciones o *wavelets* de valor medio nulo y definidas sobre intervalos finitos. La idea básica de la transformada *wavelet* es la de representar una función arbitraria $f(t)$ por medio de una superposición de un conjunto de funciones base o *wavelets* [VET95, STR97]. Estas funciones base se obtienen a partir de una única función, llamada *wavelet* madre, por medio de operaciones de escalado y traslación de esta *wavelet* madre en el dominio temporal. Lo atractivo de esta transformada es que el desarrollo en serie de *wavelets* conduce a una estructura de descomposición multiresolución por medio de un algoritmo eficiente de cálculo basado en la implementación piramidal de un banco de filtros [RAM01].

5.4.1. Diseño de la 1-D DWT

La transformada discreta *wavelet* unidimensional (1-D DWT) se puede calcular por medio del algoritmo piramidal desarrollado por Mallat [MAL89a, MAL89b]. La transformada *wavelet* descompone una señal en sus secuencias de aproximación y detalle en diferentes niveles u octavas. El cálculo de la transformada *wavelet* se inicia con una señal de entrada, $a_n^{(0)}$. La aproximación y detalle de la señal en el nivel i se definen como:

$$a_n^{(i)} = \sum_{k=0}^{N-1} g_k a_{2n-k}^{(i-1)} \quad \text{(Filtro paso baja)} \tag{5.3}$$

$$d_n^{(i)} = \sum_{k=0}^{N-1} h_k a_{2n-k}^{(i-1)} \quad \text{(Filtro paso alta)} \tag{5.4}$$

siendo $a_n^{(i-1)}$ y $d_n^{(i-1)}$ sus secuencias de aproximación y detalle en el nivel de descomposición i–1, respectivamente, mientras que g_k y h_k representan los coeficientes de los filtros paso baja y paso alta de orden N respectivamente.

Las ecuaciones (5.3) y (5.4) describen el cálculo de la transformada *wavelet* unidimensional. Los coeficientes de los filtros se obtienen a partir de la representación de la *wavelet* madre en tiempo continuo. La resolución de las diferentes aproximaciones así obtenidas disminuye al aumentar i. Puesto que las señales (i= 1, 2, ..., J) representan a la señal original en varias resoluciones, la DWT también se conoce como una técnica de descomposición multiresolución de señales. Esta estructura recibe el nombre de

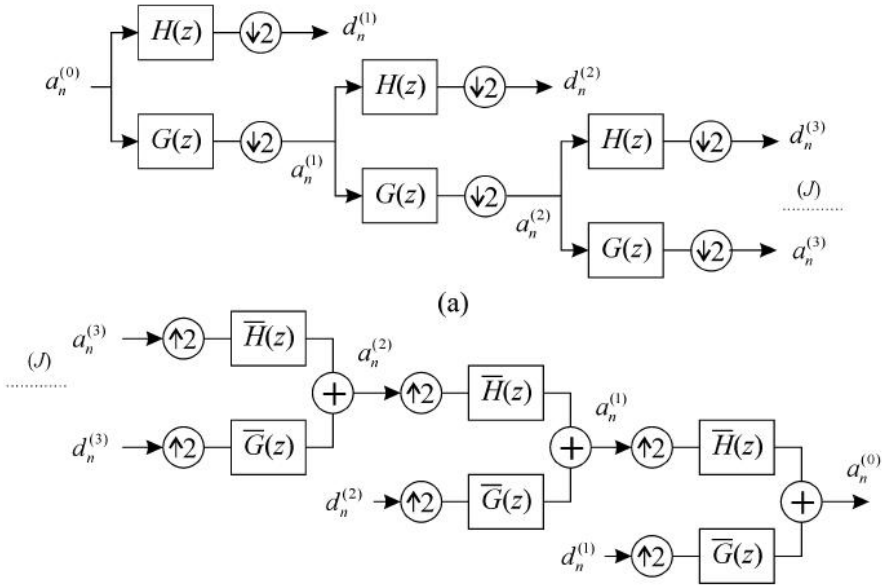

Figura 5.11. Diagrama de bloques de la descomposición multiresolución de una señal por medio de la transformada wavelet. (a) Banco de análisis y (b) síntesis.

banco de filtros de análisis. La señal de entrada $a_n^{(0)}$ se puede reconstruir a partir de su descomposición multiresolución [RAM01] de acuerdo con:

$$a_n^{(i-1)} = \sum_k \left\{ a_k^{(i)} \overline{g}_{n-2k} + d_k^{(i)} \overline{h}_{n-2k} \right\} \tag{5.5}$$

Esta ecuación describe el banco de filtros de síntesis, que consta de un filtro paso baja y otro paso alta de coeficientes $\overline{g}_n$ y $\overline{h}_n$, respectivamente, y que se encuentran fijados por las condiciones de reconstrucción perfecta [VET95, STR97] y de la *wavelet* madre en tiempo continuo. La figura 5.11 muestra un diagrama de bloques de los bancos de filtros de análisis y síntesis. El diseño del banco de filtros de análisis ha de tener en cuenta la etapa de diezmado a la salida de los filtros para conseguir una realización eficiente del sistema. Nótese que sólo se necesita una de cada dos muestras de la señal filtrada para la obtención de las secuencias de aproximación y detalle en cada nivel de descomposición. De esta manera, la etapa de diezmado se puede realizar eficientemente a la entrada y así se consigue que el sistema sólo calcule los valores que se necesitan a la salida.

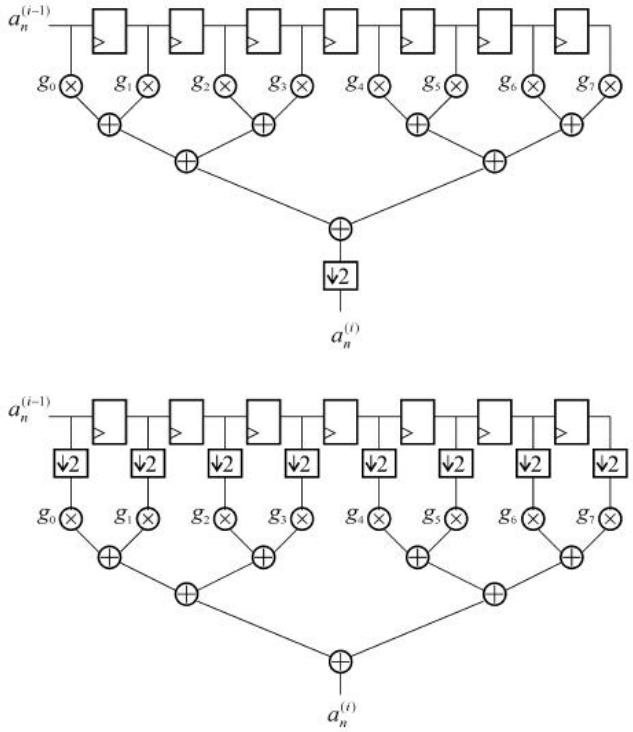

Figura 5.12. Implementación eficiente del banco de filtros de análisis por medio de la supresión a la salida del árbol de sumadores y a la entrada de los multiplicadores.

La figura 5.12 muestra la estructura original del sistema y su implementación eficiente. La cadena de supresión y registros de la estructura original se puede reestructurar de acuerdo con la siguiente observación. El multiplicador por el coeficiente g_0 realiza los productos por la secuencia de datos de índices pares, $a_0^{(i-1)} = \left\{a_{n=0}^{(i-1)}, a_{n=2}^{(i-1)}, ...\right\}$ mientras que el multiplicador por el coeficiente g_1 genera los productos por la secuencia de datos de índices impares, $a_1^{(i-1)} = \left\{a_{n=1}^{(i-1)}, a_{n=3}^{(i-1)}, ...\right\}$. Del mismo modo, los multiplicadores por los coeficientes g_2, g_4 y g_6 utilizan como entrada versiones retardadas uno, dos y tres ciclos de reloj de la secuencia $a_0^{(i-1)}$, respectivamente, y los multiplicadores por los coeficientes g_3, g_5 y g_7 calculan los productos por la secuencia de datos $a_1^{(i-1)}$ retardada con etapas de suma dispuestas en forma de árbol o en forma de cadena uno, dos y tres ciclos de reloj, respectivamente. Este hecho sugiere el tratar las secuencias $a_0^{(i-1)}$ y $a_1^{(i-1)}$ por separado y conduce a una estructura más regular en la cual el filtro se descompone en dos filtros que operan en paralelo sobre las secuencias de entrada de índices par e impar. Por tanto, el

procedimiento de modificación de la etapa de diezmado se puede realizar de una manera alternativa mediante la descomposición polifase, con lo que las secuencias $a_n^{(i)}$ y $d_n^{(i)}$ se calcularían por medio de la descomposición polifase del banco de filtros de análisis:

- los coeficientes de los filtros polifase son los coeficientes de índice par e impar de los filtros originales;
- la secuencia de entrada se descompone en sus componentes polifase que están definidas por las fases par e impar de la secuencia de entrada, respectivamente.

De esta forma, los filtros polifase usan como entradas las fases par e impar de la secuencia de entrada para calcular las salidas del banco de filtros respectivamente. La correspondiente división de la secuencia de entrada $a^{(i-1)}$ se puede realizar por medio de un conmutador controlado por el reloj de muestreo. Los filtros polifase pueden operar en paralelo y han de ser temporizados por un reloj de frecuencia mitad a la del reloj de muestreo. Por lo tanto, la 1-D DWT de orden *N* de una secuencia x_n se define mediante la descomposición multiresolución que indican las expresiones (5.3) y (5.4) con $a_n^{(0)} \equiv x_n$.

Como ejemplo de diseño 1-D DWT para la aplicación del método de protección propuesto se eligió un banco de filtros programable de 8 etapas con entrada y salida de datos de 16 y 19 bits, respectivamente, que incluye descomposición polifase. El sistema es programable ya que puede inicializarse cargando los coeficientes del filtro. La entrada se descompone en las secuencias par e impar y se calculan los productos necesarios por el filtro con ayuda de multiplicadores y un árbol de sumadores. El sistema exhibe simetría en el cálculo de las secuencias de aproximación y detalle.

A continuación se describen los pasos seguidos para la protección del diseño de la 1-D DWT. Tras analizar los recursos disponibles en el diseño, se ha aplicado la estrategia de protección EPBBF-LC, y se han utilizado las dos opciones propuestas de forma independiente para la lógica de extracción de la firma. Los diseños firmados se han evaluado en cuanto los recursos utilizados, las prestaciones y el nivel de seguridad que ofrecen.

5.4.2. Preparación de la firma y proceso de difusión

En este caso han empleado las mismas firmas digitales que para el filtro FIR-RNS: MD5, SHA1, MD5 con corrección de errores en un bit y SHA1 con corrección de errores en un bit.

En el diseño elegido para la 1D-DWT los multiplicadores y sumadores necesarios no se implementan explícitamente con estructuras de memoria. Por ello, a este diseño se le ha aplicado la estrategia de protección EPBBF-LC, que consiste en la búsqueda de los bloques de bits de la firma dentro de la lógica combinacional que describe el diseño. La lógica combinacional utilizada para la búsqueda de los bloques de bits de la firma dentro de la 1D-DWT es la de los multiplicadores y sumadores, respetando el cauce de los datos. Como este sistema es simétrico en el cálculo de las secuencias de aproximación y detalle, se pueden aplicar las mismas ideas para la difusión de los bits de la firma en ambos subsistemas. De esta forma, se reduce el número de ciclos de reloj necesarios para que el sistema genere la firma digital ya que, tras una petición de extracción de la firma, los bloques de bits de la firma se obtienen como dos secuencias de salida paralelas. Por otro lado, teniendo en cuenta que la estructura de cada subsistema corresponde a ocho productos y un árbol de sumadores de tres niveles, se consideró la posibilidad de identificar cada bloque de la firma de 16 bits como la suma de dos combinaciones lineales, siendo cada combinación lineal igual a la suma de dos productos, es decir, cada bloque de bits de la firma se identifica con la suma de cuatro productos. De esta forma, para cada subsistema cuatro de los ocho multiplicadores se utilizarán para la difusión de los bits de la firma, que recibirán el nombre de multiplicadores de difusión. La tabla 5.8 muestra lo anterior para la firma digital MD5 de 128 bits. Como se puede ver, se han organizado los multiplicadores de difusión para que se puedan observar los valores que deben ir tomando los patrones de entrada de cada multiplicador. Esa forma de organizar los datos o posiciones de la firma se entenderá mejor cuando se utilice la herramienta automática para la difusión de los bits de cada bloque de la firma. Los multiplicadores del subsistema que calcula la secuencia de aproximación que se emplean para la difusión de los bits de la firma son M_{A1}, M_{A2}, M_{A3} y M_{A4}. Por otro lado, los multiplicadores del subsistema que calcula la secuencia de detalle y, que de igual forma se emplearon para la difusión de los bits de la firma, son M_{D1}, M_{D2}, M_{D3} y M_{D4}. Como se vera más adelante x_{A1}, y_{A1}, x_{A2}, y_{A2}, x_{A3}, y_{A3}, x_{A4}, $y_{A4,}$ para

Bloques de la firma 16 bits	**Subbloques de la firma**	**Combinación lineal**	**Multiplicadores de difusión**			
c2da0f795d03238f bd72 ab3051000f81			**Multiplicadores del subsistema de aproximación**			
			$\mathbf{M_{A1}}$		$\mathbf{M_{A2}}$	
			x_{A1}	y_{A1}	x_{A2}	y_{A2}
c2da	c200	C0×F0 + 10×E0	11000000	11110000	00010000	11100000
5d03	5d00	A0×90 +10×30	10100000	10010000	00010000	00110000
bd72	Bd00	C0×F0 + 10×90	11000000	11110000	00010000	10010000
5100	5100	90×90 + 00×00	10010000	10010000	00000000	00000000
			$\mathbf{M_{A3}}$		$\mathbf{M_{A4}}$	
			x_{A3}	y_{A3}	x_{A3}	y_{A3}
c2da	00da	0E×0F + 01×08	00001110	00001111	00000001	00001000
5d03	0003	01×03 + 00×00	00000001	00000011	00000000	00000000
bd72	0072	0A×0A + 02×07	00001010	00001010	00000010	00000111
5100	0000	00×00 + 00×00	00000000	00000000	00000000	00000000
c2da0f795d03238f bd72 ab3051000f81			**Multiplicadores del subsistema de detalles**			
			$\mathbf{M_{D1}}$		$\mathbf{M_{D2}}$	
			x_{D1}	y_{D1}	x_{D2}	y_{D2}
0f79	0f00	10×F0 + 00×00	00010000	11110000	00000000	00000000
238f	2300	50×70 + 00×00	01010000	01110000	00000000	00000000
ab30	Ab00	A0×F0 + 30×70	10100000	11110000	00110000	01110000
0f81	0f00	10×F0 + 00×00	00010000	11110000	00000000	00000000
			$\mathbf{M_{D3}}$		$\mathbf{M_{D4}}$	
			x_{D3}	y_{D3}	x_{D4}	y_{D4}
0f79	0079	0B×0C + 01×01	00001010	00001100	00000001	00000001
238f	008f	0C×0D + 00×00	00001011	00001101	00000000	00000000
ab30	0030	06×08 + 00×00	00000110	00001000	00000000	00000000
0f81	0081	0A×0C+ 01×09	00001010	00001100	00000001	00001001

Tabla 5.8. Difusión de los bits de la firma digital MD5 realizada manualmente.

el subsistema de aproximación, y x_{D1}, y_{D1}, x_{D2}, y_{D2}, x_{D3}, y_{D3}, x_{D4}, $y_{D4,}$ para el subsistema de detalles, serán los 8 MSBs de los patrones de entrada que se deben aplicar a los multiplicadores de difusión durante el proceso de extracción de la firma.

Siguiendo con la difusión de los bits de la firma digital MD5, en primer lugar ésta se divide en 8 bloques de 16 bits. El primer, tercer, quinto y séptimo bloques serán difundidos en el subsistema que calcula la secuencia de aproximación y, por tanto, se obtendrán como cuatro secuencias de salida consecutivas de este subsistema, mientras que el segundo, cuarto, sexto y octavo bloques serán difundidos en el subsistema que

calcula la secuencia de detalles y se obtendrán como cuatro secuencias de salida consecutivas de este otro subsistema. Cada bloque de 16 bits se va a identificar con la suma de dos combinaciones lineales. Así la difusión de cada bloque de bits de la firma consiste en la búsqueda de los patrones de entrada de los multiplicadores de difusión cuyas salidas, procesadas como ya se ha indicado, generan el bloque de bits correspondiente.

Para entender mejor el procedimiento seguido, se va a mostrar un ejemplo que describe los pasos a seguir para la difusión de los 16 bits del primer bloque de la firma, C2DA, dentro del subsistema que calcula la secuencia de aproximación y cómo se generarían y procesarían los datos de este subsistema la extracción de la firma:

- los multiplicadores que intervienen en la difusión de C2DA son los cuatro correspondientes al subsistema de aproximación, M_{A1}, M_{A2}, M_{A3} y M_{A4};
- C2DA se identifica con dos subbloques, b_1 y b_2, de tal forma que b_1 + b_2 = C2DA; en este caso los bloques serían *C*200 y 00DA;
- a su vez, los bloques b_1 y b_2 se identifican con la suma de dos productos; para b_1 y b_2 existen varias posibilidades; una de ellas, que coincide con la que se ha tenido en cuenta en los resultados de la tabla 5.8, sería la siguiente:

$$\mathrm{C200} = x_{A1} \times y_{A1} + x_{A2} \times y_{A2} = \mathrm{C0} \times \mathrm{F0} + 10 \times \mathrm{E0} = \mathrm{B400} + \mathrm{0E00}$$
$$\mathrm{00DA} = x_{A3} \times y_{A3} + x_{A4} \times y_{A4} = \mathrm{0E} \times \mathrm{0F} + 01 \times 08 = \mathrm{00D2} + 0008$$

- el subbloque C200 se obtiene con dos de los ocho multiplicadores de la secuencia de aproximación y uno de los sumadores del primer nivel del árbol de sumadores, mientras que el otro subbloque 00DA se obtiene con otros dos multiplicadores de los seis que quedan disponibles y otro de los sumadores del primer nivel de árbol de sumadores. Las salidas del resto de los multiplicadores debe ser 0000, por lo que sus entradas durante la extracción de la firma se ponen a 0000. De esta forma, en el primer nivel del árbol de sumadores, un sumador recibe como entradas B400 y 00E0 y su salida es C200, otro sumador recibe como entradas 00D2 y 0008 y da como salida 00DA y los otros dos sumadores restantes reciben como entradas 0000 y su salida es 0000. En el segundo nivel del árbol de sumadores, un sumador recibe las entradas C200 y 00DA y sus salida es C2DA y el otro recibe como entradas 0000 y su salida es 0000. Por último el sumador del

tercer nivel recibe las entradas C2DA y 0000 y su salida es C2DA. Por tanto, para este bloque de bits de la firma, las entradas que habría que aplicar a los respectivos multiplicadores de difusión serían los siguientes:

$$x_{A1} = \text{C0} = 11000000, \quad y_{A1} = \text{F0} = 11110000,$$
$$x_{A2} = 10 = 00010000, \quad y_{A2} = \text{E0} = 11100000,$$
$$x_{A3} = 0\text{E} = 00001110, \quad y_{A3} = \text{F0} = 00001111,$$
$$x_{A4} = 01 = 00000001, \quad y_{A4} = 08 = 000010000$$

Hasta ahora se han representado los patrones de entrada de los multiplicadores de difusión como datos de 8 bits. Los datos de entrada del ejemplo de la 1-D DWT son de 16 bits, por lo que las entradas de los multiplicadores también deberán ser de 16 bits. Si en el diseño de la 1-D DWT el rango dinámico de los datos de salida coincidiera con el que requieren las operaciones realizadas, sería lógico determinar que, para tener patrones de entrada de 16 bits, bastaría añadirles ocho ceros por la izquierda a los patrones de entrada de 8 bits indicados en la tabla 5.8. Sin embargo, para que la extracción de la firma se realice de forma correcta, además de tener en cuenta que los datos de entrada del ejemplo de la 1-D DWT son de 16 bits, es importante no olvidar que las operaciones que se realizan con ellos son una multiplicación, de la que sólo se consideran los 16 MSBs, y tres sumas, correspondientes al árbol de sumadores. De esta forma, durante la extracción de la firma se deben aplicar a los multiplicadores de difusión los patrones de entradas adecuados, teniendo en cuenta que del resultado de cada multiplición sólo se consideran los 16 MSBs.

Siguiendo con el ejemplo, para extraer los bits de la firma C2DA, los multiplicadores de difusión M_{A1}, M_{A2}, M_{A3} y M_{A4} deberán generarar las salidas B400, 0E00, 00D2 y 0008, respectivamente. Teniendo en cuenta los datos de difusión de la tabla 5.8 para este bloque de bits, a continuación se dan los patrones de entrada que se deben aplicar a los multiplicadores de difusión del subsistema que calcula la secuencia de aproximación para que se generen los datos cuyos 16 MSBs coinciden con las salidas deseadas:

$$M_{A1}: \; x_{A1}\&00 \times y_{A1}\&00 = \text{C0}\mathbf{00} \times \text{F0}\mathbf{00} = \text{B4000000} \Rightarrow \text{B400}$$

$$M_{A2}: \; x_{A2}\&00 \times y_{A2}\&00 = 10\mathbf{00} \times \text{E0}\mathbf{00} = \text{0E000000} \Rightarrow \text{0E00}$$

$$M_{A3}: \; x_{A3}\&00 \times y_{A3}\&00 = 0\text{E}\mathbf{00} \times 0\text{F}\mathbf{00} = \text{00D20000} \Rightarrow \text{00D2}$$

$$M_{A4}: \ x_{A4}\&00 \times y_{A4}\&00 = 01\mathbf{00} \times 08\mathbf{00} = 00080000 \Rightarrow 0080$$

Se puede comprobar que los 8 MSBs de los patrones de entrada de 16 bits que se deben aplicar a los multiplicadores de difusión son los 8 bits que componen las señales x_{A1}, y_{A1}, x_{A2}, y_{A2}, x_{A3}, y_{A3}, x_{A4}, $y_{A4,}$ y_{D1}, x_{D2}, y_{D2}, x_{D3}, y_{D3}, x_{D4}, y_{D4} de la tabla 5.8 y los restantes bits de estos patrones de entrada, es decir, los 8 LSBs, son todo ceros.

5.4.3. Preparación de los diseños para la extracción de la firma

El siguiente paso en la aplicación del método de protección es la preparación de los diferentes diseños firmados para la extracción de la firma correspondiente. En los diseños de la 1-D DWT firmados mediante la estrategia de protección EPBBF-LC, los bloques de bits de la firma se han difundido en cuatro de los ocho multiplicadores de los subsistemas de aproximación y detalle. Para la extracción de la firma se debe incluir dentro de estos diseños la lógica necesaria para poder acceder, dentro de cada subsistema, a las posiciones de la firma y dirigir su contenido, es decir, los bloques de bits de la firma, hacia las salidas de ambos subsistemas. A continuación se describen de las lógicas de extracción desarrolladas, que son similares a las del filtro FIR-RNS.

5.4.3.1. Lógica de extracción: FSM y multiplexores

La secuencia de extracción de la firma elegida para el desarrollo de esta lógica de extracción y su introducción dentro de los diseños firmados, es 0000, 0001, 0000, 0001, 0000, 0001. En este caso, la lógica de extracción es una máquina de estados, que detecta la SEF y genera las posiciones de memoria de la firma, y 32 multiplexores de 2 a 1 que se utilizan para seleccionar como entradas de los multiplicadores las salidas de la máquina de estados. Se necesitan 32 multiplexores para controlar las 2 entradas de los 8 multiplicadores del subsistema que calcula la secuencia de aproximación y las 2 entradas de los 8 multiplicadores del subsistema que calcula la secuencia de detalles.

La máquina de estados tiene como entrada la del sistema, $x(n)$, y como salidas 16 señales que serán las entradas de los multiplicadores de difusión y la señal *en_test*. La señal *y_machine* servirá para acceder a las posiciones de la firma cuando se active el proceso de extracción, mientras que la señal *en_test*, después de activarse el proceso de extracción de la firma, permanecerá activa un número de ciclos de reloj igual al número

de bloques de bits de la firma más la latencia propia del diseño, y actuará como la señal de control de los multiplexores.

El funcionamiento de los diseños con la lógica de extracción es el siguiente:

- si la máquina de estados no detecta la SEF, se situa en un estado de espera en el que la salida tiene todos sus bits a 0 y la señal *en_test* está inactiva. El funcionamiento de los diseños protegidos corresponde al de un procesamiento normal de los datos de entrada;
- si la FSM detecta la SEF, el proceso de extracción de la firma ha de comenzar. La FSM pasa por una serie de estados en los que la señal *en_test* permanece activa y las 16 señales de salida van tomando los valores de las posiciones de la firma. Al estar activa *en_test*, los multiplexores seleccionan las 16 salidas de la máquina de estados como las entradas de los multiplicadores de difusión, y todo ceros como entradas del resto de los multiplicadores. De esta forma, se va accediendo a cada una de las posiciones de la firma y los bits de la firma se dirigen hacia la salida del subsistema correspondiente.

5.4.3.2. Lógica de extracción basada en LFSR

La estructura básica de esta lógica de extracción, que considera que la secuencia de extracción de la firma ha sido generada con la ayuda de un LFSR, es la misma que la del filtro FIR-RNS (figura 4.13). Como ya se vió, el funcionamiento del bloque LFSR, el comparador y el contador es el mismo para todos los diseños que inlcuyen esta lógica de extracción. Recuérdese que el LFSR, el comparador y el contador tienen la función de detectar la SEF y activar el proceso de extracción mientras que, una vez activado éste, la máquina de estados generará como señales de salida las PFs. Por otro lado, los 16 multiplexores de 2 a 1, cuando el proceso de extracción se activa, seleccionan como entradas de los multiplicadores las salidas de la máquina de estados. El LFSR, contador y comparador empleados para los diseños de la 1-D DWT son los mismos que los del filtro FIR-RNS.

Por otro lado, la máquina de estados es similar a la empleada para los diseños mencionados, con algunas diferencias. La máquina de estado para los diseños protegidos de la 1-D DWT tiene como entrada la señal de control que genera el

comparador, *en_FSM*, y, al igual que con la primera de las lógicas de extracción, 16 salidas que serán las entradas de los multiplicadores de difusión y la salida *en_test.* El funcionamiento del sistema sería el siguiente:

- si no se ha detectado la SEF, la señal *en_FSM* que genera el contador permanece inactiva y la máquina de estados se situa en un estado de espera, con la salida a 0 y la señal *en_test* inactiva. El sistema opera de forma normal;
- si se detecta la SEF, la señal de control *en_FSM* se activa, indicando que el proceso de extracción de la firma ha de comenzar. La máquina de estados pasa por una serie de estados en los que la señal *en_test* permanece activa y las 16 señales de salida van tomando los valores de las posiciones de la firma. Al estar activa *en_test*, los 32 multiplexores de 2 a 1 conectan las salidas de la máquina de estados con las entradas de los multiplicadores y ponen a cero las entradas del resto de los multiplicadores. De esta forma, se va accediendo a cada una de las posiciones de la firma y el propio diseño va generando los bits de la firma y dirigiéndolos hacia la salida del subsistema correspondiente.

5.4.4. Resultados experimentales

Para evaluar la aplicación de la estrategia de protección sobre los diseños firmados de la 1-D DWT, se realizó la síntesis del diseño original y de los diseños protegidos para dispositivos programables de la familia Altera APEX20KC [ALT08] y de la familia Xilinx Virtex 2 [XIL08] y para procesos ASICs *semi-custom* empleando las herramientas de Synopsys [RAB04]. Se determinaron los recursos necesarios para los diseños, así como la máxima frecuencia de operación

En la tabla 5.9 se comparan los resultados de área y velocidad. En concreto, se compara el diseño original con los diseños *FIR-FSM* y *FIR-LFSR* protegidos mediante la estrategia EPBBF-LC, considerando las cuatro firmas digitales. La tabla muestra el incremento de área y la reducción en la velocidad respecto del diseño original para cada uno de los diseños protegidos. Un análisis detallado muestra que los filtros protegidos mediante la estrategia de protección EPBBF-LC ocupan un área nunca mayor al 2.2% con respecto al diseño original. Este resultado es importante si se compara con el 4.4% de incremento de área máxima que se consiguió con la misma estrategia para los

			Xilinx		Altera		Synopsys	
Diseño original			SLICEs	F (MHz)	LEs	F (MHz)	Area	F (MHz)
1D-DWT			2927	198.9	7271	122.7	48798	159.5
Diseños protegidos			Incr. área	Red. velocidad	Incr. área	Red. velocidad	Incr. área	Red. velocidad
1D-DWT FSM	MD5	sin ECC	0.4 %	< 1%	1.7 %	< 1%	0.1 %	< 1%
		1-bit ECC	0.5 %	< 1%	1.6 %	< 1%	0.2 %	< 1%
	SHA1	sin ECC	0.7 %	< 1%	1.6 %	< 1%	0.3 %	< 1%
		1-bit ECC	0.7 %	< 1%	1.8 %	< 1%	0.3 %	< 1%
1D-DWT LFSR	MD5	sin ECC	1.4 %	< 1%	2.2 %	< 1%	1.1 %	< 1%
		1-bit ECC	1.5 %	< 1%	2.2 %	< 1%	1.2 %	< 1%
	SHA1	sin ECC	1.6 %	< 1%	2.2 %	< 1%	1.3 %	< 1%
		1-bit ECC	1.6 %	< 1%	2.2 %	< 1%	1.3 %	< 1%

Tabla 5.9. Impacto sobre el área y las prestaciones de los diseños protegidos mediante la estrategia EPBBF-LC.

diseños firmados del filtro FIR-RNS (tabla 5.7). Se puede comprobar una vez más que, para una determinada firma digital con la que se pretende identificar el diseño, el incremento de área que supone la aplicación de una determinada estrategia de protección es menor a medida que aumenta el tamaño del mismo. Por otro lado, como también se refleja en la tabla 5.9, la reducción en velocidad es despreciable para todos los diseños protegidos.

5.5. Difusión de los bits de la firma mediante la herramienta automática

En el capítulo 4 se describió la herramienta automática desarrollada para facilitar la difusión de los bits de la firma en la estrategia de protección EPBBF-LC. Esta herramienta automatiza el proceso de difusión, reduciendo el esfuerzo y tiempo necesarios con respecto a una difusión manual. A continuación se va a mostrar el uso de

la herramienta automática para el proceso de difusión y los resultados obtenidos en la aplicación de la estrategia de protección EPBBF-LC al ejemplo de la 1-D DWT y se van a comparar los resultadoss obtenidos con los que se obtuvieron mediante una difusión manual.

5.5.1. Aplicación al ejemplo de la 1-D DWT

En el desarrollo de los diseños firmados de la 1D-DWT, la búsqueda de los patrones de entrada se realizó manualmente. La tabla 5.8 muestra los resultados de la difusión de la firma digital MD5 realizada de esta forma. En estos diseños no se consideró la posibilidad de realizar una selección adecuada de todas las alternativas existentes para la difusión de un mismo bloque de bits de la firma. Por ejemplo, para C2DA otra posibilidad estaría compuesta por estas otras dos combinaciones lineales:

$$\mathrm{C200} = 30 \times 40 + \mathrm{E0} \times \mathrm{D0} = \mathrm{0C00} + \mathrm{B600}$$
$$\mathrm{00DA} = \mathrm{0B} \times 08 + \mathrm{0D} \times \mathrm{0A} = 0058 + 0082$$

En este caso, los 8 MSBs de los patrones de entrada de 16 bits que habría que aplicar a los multiplicadores de difusión serían los siguientes:

$$x_{A1} = 30 = 00110000 \quad y_{A1} = 40 = 01000000$$
$$x_{A2} = \mathrm{E0} = 11100000 \quad y_{A2} = \mathrm{D0} = 11010000$$
$$x_{A3} = \mathrm{0B} = 00001011 \quad y_{A3} = 08 = 00001000$$
$$x_{A4} = \mathrm{0D} = 00001101 \quad y_{A4} = \mathrm{0A} = 00001010$$

La herramienta automática desarrollada, además de automatizar el proceso de difusión, ayuda a realizar una selección adecuada entre todas las posibles soluciones para la difusión de los bloques de bits de la firma. Esta herramienta se utilizó también para el desarrollo de nuevos diseños firmados de la 1-D DWT, bajo la estrategia de protección EPBBF-LC. En este caso, además de las firmas digitales DiTEC MD5 y DiTEC SHA1 (que no incluyen corrección de errores), se van a considerar también las firmas digitales UGR MD5, UGR SHA1, FSU MD5 y FSU SHA1, mostradas en la figura 4.4, lo que ayudará a comprobar en un número mayor de diseños las ventajas que ofrece es la herramienta automática.

En la tabla 4.4 se mostraron los resultados obtenidos para la difusión automática de firmas digitales MD5 y SHA1 mediante los tres algoritmos propuestos, utilizando como lógica combinacional la suma de productos. Aprovechando la estructura del propio diseño, en el ejemplo de aplicación de la estrategia EPBBF-LC a la 1-D DWT mediante difusión manual, se ha utilizado también esta lógica combinacional para la difusión de la firma. Por lo tanto, los datos de la tabla 4.4 pueden ayudar a decidir cuál de los tres algoritmos es óptimo para la búsqueda automática de la firma digital dentro del diseño considerado. Teniendo en cuenta que el tamaño de bloque considerado es de 8 bits, la tabla 4.4, que muestra resultados para descomposición de bloques como suma de productos, indica que el algoritmo más adecuado es el algoritmo optimizado, ya que se consiguen mejores resultados para las medidas sobre la distancia Hamming global y el tiempo de ejecución de este algoritmo es inferior a un minuto. Sin embargo, ya que la diferencia entre las medidas de la distancia Hamming global no es demasiado acusada para el algoritmo optimizado y el de Enfriamiento Simulado, se ha realizado una búsqueda de las firmas digitales de la figura 4.4 dentro del diseño de la 1-D DWT considerando estos dos algoritmos de forma independiente.

Para mostrar un ejemplo de la difusión automática, la tabla 5.10 muestra los resultados obtenidos para la difusión automática de la firma digital DiTEC MD5 con la ayuda del algoritmo optimizado. Como se puede apreciar, de entre todas las combinaciones posibles de patrones de entrada, el algoritmo optimizado seleccionó la solución que consigue minimizar la distancia Hamming de los 8 MSBs de los patrones de entrada de cada grupo (x_{A1}, y_{A1}, x_{A2}, y_{A2}, x_{A3}, y_{A3}, x_{A4}, y_{A4}, x_{D1}, y_{D1}, x_{D2}, y_{D2}, x_{D3}, y_{D3}, x_{D4} e y_{D4}). Por ejemplo, el patrón de entrada x_{A2} permanece al mismo valor, “11010000”, para generar junto con los valores que indican los patrones de entrada correspondientes a x_{A1}, y_{A1} e y_{A2}, los bloques de bits de la firma C2, 5D, BD y 51. Como ya se mencionó anteriormente, aquí se puede entender mejor la forma elegida para mostrar los patrones de entrada de los multiplicadores de difusión: se muestran los valores que deben ir tomando los 8 MSBs de cada grupo de patrones de entrada durante el proceso de extracción de la firma, ya que para conseguir reducir al máximo el número de recursos adicionales utilizados por la lógica de extracción, la distancia Hamming entre en cada grupo de patrones de entrada se ha de reducir al máximo.

Bloques de la firma 16 bits	**Bloques de la firma**	**Combinación lineal**	**Multiplicadores de difusión**			
c2da0f795d03238f bd72 ab3051000f81			**Multiplicadores del subsistema de aproximación**			
			$\mathbf{M_{A1}}$		$\mathbf{M_{A2}}$	
			x_{A1}	y_{A1}	x_{A2}	y_{A2}
c2da	c200	40×30 + D0×E0	01000000	00110000	11010000	11100000
5d03	5d00	50×30 +D0×60	01010000	00110000	11010000	01100000
Bd72	Bd00	10×70 + D0×E0	00010000	01110000	11010000	11100000
5100	5100	10×30 + D0×60	00010000	00110000	11010000	01100000
			$\mathbf{M_{A3}}$		$\mathbf{M_{A4}}$	
			x_{A3}	y_{A3}	x_{A3}	y_{A3}
c2da	00da	0E×0C + 0A×05	00001110	00001100	00001010	00000101
5d03	0003	0E×00 + 03×01	00001110	00000000	00000011	00000001
Bd72	0072	0E×08 + 02×01	00001110	00001000	00000010	00000001
5100	0000	0E×00 + 02×00	00001110	00000000	00000010	00000000
c2da0f795d03238f bd72 ab3051000f81			**Multiplicadores del subsistema de detalle**			
			$\mathbf{M_{D1}}$		$\mathbf{M_{D2}}$	
			x_{D1}	y_{D1}	x_{D2}	y_{D2}
0f79	0f00	20×50 + 10×50	00100000	01010000	00010000	01010000
238f	2300	60×50 + 10×50	01100000	01010000	00010000	01010000
Ab30	Ab00	60×90 + 90×D0	01100000	10010000	10010000	11010000
0f81	0f00	20×10 + 10×D0	00100000	00010000	00010000	11010000
			$\mathbf{M_{D3}}$		$\mathbf{M_{D4}}$	
			x_{D3}	y_{D3}	x_{D4}	y_{D4}
0f79	0079	09×0B + 0B×02	00001001	00001011	00001011	00000010
238f	008f	0B×0B + 0B×02	00001011	00001011	00001011	00000010
Ab30	0030	03×0A + 09×02	00000011	00001010	00001001	00000010
0f81	0081	03×0A + 09×0B	00000011	00001010	00001001	00001011

Tabla 5.10. Difusión automática de los bits de la firma digital MD5 en el diseño de la 1-D DWT realizada mediante el algoritmo optimizado.

5.5.2. Implementación de los diseños protegidos

Los diseños protegidos mediante la difusión de los bits de la firma realizada con la herramienta automática fueron sintetizados para dispositivos programables de la familia Altera APEX20KC y de la familia Xilinx Virtex 2. La tabla 5.11 muestra sólo los resultados obtenidos en términos de área, ya que, en todos los casos, la velocidad de estos diseños supone una dismunición inferior al 1% con respecto al diseño original. Esta tabla también muestra los resultados de síntesis para los diseños protegidos en los

Diseño original			**Xilinx**		**Altera**	
			SLICEs		LEs	
1D-DWT			2927		7271	
Diseños protegidos			**Xilinx**		**Altera**	
	Firma digital	Difusión	SLICEs	FSM SLICEs	LEs	FSM LEs
1D-DWT FSM	DiTEC MD5	Manual	2941	18	7409	35
		Optimizado	2939	16	7393	34
		SAMB	2940	17	7394	34
	DiTEC SHA1	Manual	2946	22	7395	40
		Optimizado	2945	18	7386	31
		SAMB	2945	19	7395	33
	UGR MD5	Optimizado	2940	17	7390	33
		SAMB	2939	16	7398	33
	UGR SHA1	Optimizado	2946	19	7382	34
		SAMB	2946	20	7385	35
	FSU MD5	Optimizado	2940	16	7390	35
		SAMB	2941	16	7388	35
	FSU SHA1	Optimizado	2945	19	7384	32
		SAMB	2944	20	7384	34
1D-DWT LFSR	DiTEC MD5	Manual	2969	8	7424	15
		Optimizado	2968	7	7425	14
		SAMB	2969	8	7428	15
	DiTEC SHA1	Manual	2973	16	7431	28
		Optimizado	2972	11	7427	22
		SAMB	2971	10	7430	19
	UGR MD5	Optimizado	2969	8	7423	14
		SAMB	2968	9	7426	15
	UGR SHA1	Optimizado	2972	11	7426	20
		SAMB	2972	11	7428	22
	FSU MD5	Optimizado	2969	8	7423	15
		SAMB	2970	11	7425	15
	FSU SHA1	Optimizado	2971	10	7423	18
		SAMB	2971	8	7427	21

Tabla 5.11. Recursos de la 1-D DWT de los diseños protegidos mediante la estrategia EPBBF-LC con difusión manual y con difusión realizada mediante la herramienta automática.

que se realizó una difusión manual. Los resultados para los diseños protegidos con difusión manual para las firmas DiTEC MD5 y DiTEC SHA son los correspondientes a

la sección 5.4.3, con la única diferencia de que en la tabla 5.9 aparecían expresados en términos de incremento de área con respecto al diseño original y ahora se reflejan en términos de área absoluta. La tabla 5.11 también recoge los resultados obtenidos tras sintetizar de manera independiente las correspondientes máquinas de estados.

Si se comparan los resultados obtenidos mediante la difusión manual y las difusiones automáticas, se puede comprobar que el uso de la herramienta automática consigue que la penalización en área de los diseños protegidos sea menor debido a que la minimización de la distancia Hamming que realiza la herramienta automática consigue reducir los recursos de implementación empleados por las máquinas de estados que forman parte de las correspondientes lógicas de extracción. Sin embargo, se puede apreciar que, en algunos casos, el uso de la herramienta automática no consigue mejorar los resultados con respecto a una difusión manual. Esto se debe a que con anterioridad al desarrollo de la herramienta automática, se realizaron distintas difusiones manuales, basadas en un conjuntos de iteraciones que realizaban una minimización de la distancia Hamming entre los patrones de entrada, y los resultados de difusión que se muestran corresponden a la difusión que consiguió la optimización de los recursos para la lógica de extracción. Aunque en estos casos particulares la difusión mediante la herramienta automática consigue resultados similares a los obtenidos mediante una difusión manual optimizada, el uso de la herramienta automatiza el proceso de búsqueda y difusión de los bloques de bits de la firma, lo que supone un ahorro muy importante en esfuerzo y tiempo de aplicación de la estrategia de *watermarking*.

Por otro lado, la tabla 5.11 también muestra que para el algoritmo optimizado se obtienen mejores resultados de difusión y de implementación que los obtenidos mediante el algoritmo de Enfriamiento Simulado, como ya se había previsto con los datos experimentales de la tabla 4.4 que muestran la difusión automática bajo los tres algoritmos propuestos de las firmas digitales MD5 y SHA1. Por lo tanto, para el diseño de la 1-D DWT y las firmas digitales utilizadas, el algoritmo más adecuado es el optimizado y son los diseños obtenidos mediante este algoritmo los que finalmente deberían aceptarse como diseños protegidos mediante la estrategia EPBBF-LC. En este ejemplo no se ha podido comprobar las ventajas del algoritmo de Enfriamiento Simulado debido a que el tamaño de los bloques de la firma no es muy grande. En el caso de haberlo sido, que como muestra la tabla 4.3 sería para valores superiores a 10

bits, el reducido tiempo de ejecución del algoritmo de Enfriamiento Simulado haría que éste fuese el algoritmo elegido para la difusión automática de los bits de la firma, consiguiendo también buenos resultados en cuanto a la distancia Hamming y repercusión sobre los recursos del diseño original.

5.6. Evaluación de las estrategias de protección

A lo largo de este capítulo se han evaluado las estrategias de *watermarking* propuestas en términos de la influencia que tiene su aplicación en el área y las prestaciones con respecto a los diseños originales. Los resultados de síntesis han mostrado que el desarrollo de diseños firmados bajo la aplicación de estas estrategias tienen repercusión despreciable sobre las prestaciones de los diseños originales y una penalización muy baja en el área. Sin embargo, existen otros criterios para la evaluación de una técnica de *watermarking* para módulos IP como la identificación del autor, el coste, el correcto funcionamiento, la fácil extracción, la robustez y la seguridad (sección 3.5.4).

El parámetro P_c [KAH01] representa la probabilidad de que a partir del diseño original, que no haya sido firmado, se pueda extraer la firma digital, es decir, la probabilidad de que una secuencia de datos diferente a la SEF genere una secuencia de datos de salida que corresponda a la secuencia de bits de la firma digital. Considerando que m es el número de bloques de datos para representar la firma digital con b bits de salida:

$$P_c = \frac{1}{\left(2^b\right)^m} \tag{5.7}$$

Para obtener una elevada prueba de autoría, esta probabilidad ha de ser suficientemente baja. Dado un determinado módulo IP, esta probabilidad depende únicamente de la firma digital elegida para identificar el diseño. Como muestra la tabla 5.12, esta probabilidad ha sido calculada para el diseño protegido del filtro CIC y para los diferentes diseños protegidos del filtro FIR. Esta tabla muestra que, para todos los diseños, la probabilidad de que una secuencia de datos de entrada diferente a la SEF

Diseño	Firma	P_c
CIC-FSM	MD5	7.3×10^{-40}
FIR-FSM	MD5	2.9×10^{-39}
	SHA1	6.8×10^{-49}
FIR-LFSR	MD5	2.9×10^{-39}
	SHA1	6.8×10^{-49}
FIR-embedded LFSR	MD5	2.9×10^{-39}
	SHA1	6.8×10^{-49}

Tabla 5.12. Probabilidad de que una secuencia de datos diferente a la SEF genere una secuencia de datos de salida igual a la secuencia de bits de la firma digital.

genere la secuencia de bits de la firma como salida de estos diseños es siempre inferior a 2.9×10^{-39}. De esta forma, se puede afirmar que la técnica de *watermarking* propuesta ofrece una prueba de autoría muy firme.

Otro criterio de evaluación muy importante es el coste, que, como ya se comentó en el capítulo 3, se refiere al coste computacional, de desarrollo, esfuerzo, etc., que supone la aplicación de la técnica de *watermarking*. Para realizar esta evaluación se debe distinguir entre las dos estrategias de *watermarking* propuestas. Para la estrategia de *watermarking* EPIBF-LUT, un paso previo en la aplicación es el estudio del módulo IP para comprobar si incluye posiciones de memoria que no hayan sido utilizadas. Este estudio se puede realizar fácilmente comprobando los patrones de entrada de las estructuras de memoria que utilice el diseño o considerando las operaciones y la aritmética de dicho módulo. Como ya se ha visto, en los diseños basados en aritmética de residuos y/o índices, no es necesario observar el diseño para saber en qué parte del mismo se localizan estructuras de memoria con posiciones libres y cuáles son en concreto estas posiciones. Una vez comprobada la posibilidad de aplicar esta estrategia de *watermarking*, el proceso de aplicación está muy mecanizado. Se incluyen los bloques de la firma en las posiciones de memoria libres y se utiliza una de las dos lógicas de extracción desarrolladas, incorporando de forma adecuada las posiciones de

la firma. Para la estrategia EPBBF-LC, el estudio previo del diseño consistiría en identificar la lógica combinacional utilizada por éste. La difusión de los bits de la firma se realiza sin esfuerzo alguno con ayuda de la herramienta automática y la preparación del diseño para la extracción de la firma es una tarea muy sencilla. Los distintos algoritmos desarrollados para la herramienta automática consiguen encontrar un balance entre el coste computacional de la difusión de los bloques de bits de la firma y el impacto sobre los recursos del módulo que va a tener la protección del mismo. Por lo tanto, se comprueba que para esta estrategia de protección, el coste de la aplicación de la misma se puede considerar muy bajo.

El correcto funcionamiento del diseño firmado bajo cualquiera de las estrategias de protección no se ve alterado con respecto al diseño original. El único aspecto relevante con respecto a este criterio de evaluación es que, al aplicar como entrada del bloque la secuencia de extracción de la firma, éste no realizará un procesamiento normal de los datos de entrada, ya que estará realizando el proceso de extracción de la firma. Tras finalizar este proceso, el módulo volverá a procesar los datos de acuerdo con un funcionamiento normal de dicho módulo.

Otro de los criterios de evaluación considerados es el proceso de extracción de la firma, que debe ser fácil de extraer e identificar al autor y/o propietario del bloque IP. Las dos estrategias de *watermarking* inluyen un proceso de extracción muy sencillo y a su vez muy seguro, sobre todo en los diseños que incorporan la lógica de extracción basada en un LFSR. Por otro lado, el proceso de extracción de la firma es no destructivo, es decir, tras la extracción la firma sigue estando presente en el diseño y además no se revela la ubicación de los bits de la firma dentro del mismo. Otro aspecto importante relacionado con la extracción de la firma es que el coste que suponga el proceso de extracción debe ser lo menor posible. Como se ha comentado anteriormente, la preparación del diseño para la extracción de la firma es el único proceso que tiene repercusiones sobre los recursos originales del módulo, pero su impacto es muy pequeño.

Finalmente, la seguridad de un sistema de *watermaking* está relacionada con su habilidad de resistir los distintos tipos de ataques posibles. Dada su relevancia, se ha considerado conveniente dedicar el siguiente capítulo al estudio de la seguridad de los

sitemas de *watermarking* y a la valoración de las estrategias propuestas con respecto a este criterio de evaluación.

5.7. Conclusión

En este capítulo se ha aplicado el método de protección propuesto a varios bloques IP y se han mostrado con detalle los pasos para la difusión y la extracción de los bits de la firma en los distintos módulos IP elegidos como ejemplo. Previamente, se ha analizado cada diseño propuesto como ejemplo para la elección de la estrategia de protección más adecuada, EPIBF-LUT ó EPBBF-LC. El primer diseño protegido se desarrolló para un filtro CIC basado en el RNS mediante la estrategia de protección EPIBF-LUT empleando una firma digital de 128 bits y una secuencia de extracción elegida manualmente. Se desarrolló una lógica de extracción muy sencilla en la que se incluyeron los recursos necesarios (una máquina de estados y multiplexores de 2 a 1) para detectar dicha secuencia y activar el proceso de extracción de la firma. El siguiente diseño con el que se trabajó fue un filtro FIR basado en aritmética de residuos, que sirvió como ejemplo para la aplicación de las dos estrategias y para las dos opciones propuestas para la lógica de extracción de la firma. Además, en este diseño se utilizaron firmas digitales de distinta longitud, con y sin corrección de errores. Se aplicó la estrategia de protección EPIBF-LUT utilizando las posiciones de memoria libres de las tablas de la conversión RNS-binario y de las tablas de consulta de la conversión de índices a RNS, de manera independiente, para el desarrollo de los correspondientes diseños protegidos. Por otro lado, se aplicó la estrategia de protección EPBBF-LC utilizando las tablas de consulta incluidas en la conversión de RNS a binario. El último ejemplo considerado para la aplicación del método de protección fue la 1-D DWT. En este caso la estrategia de protección utilizada fue la EPBBF-LC y, al igual que en el ejemplo del filtro FIR-RNS, se consideraron cuatro firmas digitales y las dos opciones propuestas para la lógica de extracción de la firma. En este ejemplo, además del desarrollo de los distintos diseños protegidos mediante una difusión manual de los bits de la firma, también se desarrollaron diseños protegidos en los que la difusión de los bits de la firma se realizó mediante la herramienta automática.

Los diseños protegidos se sintetizaron utilizando dispositivos programables de la familia Altera APEX20KC y de la familia Xilinx Virtex 2 y para procesos ASICs *semi-custom* empleando las herramientas de Synopsys. Los resultados de síntesis mostraron que las distintos estructuras desarrolladas permiten la protección de los correspondientes diseños con penalización despreciable en sus prestaciones y área. Se compararon los resultados de implementación obtenidos por los diseños protegidos desarrollados para la 1-D DWT en los que la difusión de los bits de la firma se realizó de forma manual con aquéllos en los que esta difusión se realizó con la herramienta automática. Se pudo comprobar que la herramienta automática, además de facilitar la difusión de los bits de la firma, consigue optimizar los recursos necesarios para la lógica de extracción y, por tanto, reducir el impacto sobre el área de los distintos diseños protegidos con respecto a una difusión manual.

Finalmente, además de la evaluación realizada en términos de impacto sobre el área y las prestaciones, se ha realizado una evaluación de las estrategias de *watermarking* atendiendo a los siguientes criterios de evaluación: identificación del autor, coste, correcto funcionamiento, fácil extracción y robustez.

Capítulo 6: Análisis de la seguridad de las técnicas propuestas

En este capítulo se estudia la seguridad que ofrecen las técnicas de *watermarking* desarrolladas. Se presentan las diferentes formas de ataques sobre un contenido protegido mediante *watermarking* y, en concreto, se estudian los tipos de ataques cuando el contenido que se intenta proteger es un módulo IP. A continuación se evalúa la resistencia que ofrecen los módulos protegidos mediante las estrategias propuestas contra cada posible ataque, comprobando su alta invulnerabilidad.

6.1. Introducción

En la sección 3.6.2 se establecieron los criterios de evaluación más relevantes referentes a técnicas de *watermarking* para bloques IP. En la sección 5.6 se utilizaron estos criterios para evaluar las estrategias de *watermarking* propuestas. Sin embargo, en esta última sección no se ha incluído el estudio de la seguridad de la técnica de *watermarking* propuesta, ya que se ha considerado necesario realizar un análisis más exhaustivo de todos los aspectos relacionados con este criterio de evaluación.

La seguridad de un sistema de *watermaking* hace referencia a la resistencia que ofrece contra los tipos de ataques posibles [VOL01]. Está comúnmente aceptado que un determinado atacante puede lograr su propósito se dispone de los recursos suficientes y el tiempo necesario. Así, el objetivo es conseguir técnicas de *watermarking* que

precisen un esfuerzo, tiempo y dinero para ser vulneradas superiores al que representaría rediseñar el módulo o adquirir copias legales.

Los sistemas de *watermaking* pueden ser atacados de varias formas, y, para poder prevenir los ataques, se necesita de una determinada seguridad en la aplicación particular de cada técnica de *watermarking*. Por tanto, las exigencias de seguridad para cada técnica de *watermarking* dependen de manera muy importante tanto de la técnica en sí misma como de su aplicación particular a un determinado contenido. En algunas aplicaciones, las marcas de agua no necesitan ser seguras contra ataques malintencionados, porque no se puede llegar a obtener beneficio alguno de ellos. Sin embargo, estas marcas de agua han de ser robustas frente a un procesamiento normal del objeto que las contiene. Aquellas aplicaciones en las que se exige cierto nivel de seguridad deben defenderse contra diferentes formas de ataques. En algunos casos, los tipos de seguridad necesarios pueden variar dependiendo de las implementaciones de una misma aplicación. Además, el nivel de seguridad necesario para diferentes aplicaciones puede también variar dependiendo del nivel de sofistificación de los atacantes. Cualquier aplicación militar basada en el uso de marcas de agua necesitará el mayor nivel de seguridad posible, ya que se supone que los adversarios o atacantes disponen de todos los recursos posibles de un país enemigo. En el otro extremo se encontraría una aplicación el la que se trate de restringir el acceso a un determinado objeto por parte de cualquier niño que, en el peor de los casos, necesitaría ser seguro sólo contra ataques muy simples.

En esta sección se tratarán los aspectos de seguridad más importantes en sistemas de *watermarking*, siguiendo los criterios marcados en [COX01]. Se discutirán algunas restricciones que exigen determinadas aplicaciones sobre las manipulaciones de las marcas de agua. De forma específica, para una aplicación dada, algunos individuos pueden tener restricciones para la introducción, detección y/o eliminación de las marcas de agua. Dentro de cada tipo de restricción existen sutiles variaciones que dan lugar a los distintos tipos de ataques que un adversario podría intentar. Se analizan las suposiciones que se pueden tener en cuenta sobre los adversarios cuando se está juzgando la seguridad de cualquier técnica de *watermarking*. Finalmente, se estudia la relación existente entre criptografía y *watermarking* y se proponen técnicas criptográficas para restringir las acciones de introducción y detección.

Este capítulo se inicia con una introducción a los aspectos más importantes relacionados con la seguridad de un sistema de *watermarking*, entre ellos los distintos tipos de acciones no autorizadas que se pueden intentar en contenidos digitales a los que se les ha aplicado una determinada técnica de *watermarking*. A partir de esta visión general, se realiza un estudio de la seguridad de las técnicas de *watermarking* sobre módulos IP, basándose principalmente en la distinción entre los tipos de ataques posibles. Finalmente, se evalúan las estrategias de *watermarking* propuestas en este trabajo en términos de resistencia contra los distintos tipos de ataques y se sugieren algunas acciones para aumentar su seguridad.

6.2. Clasificación de ataques

En cada aplicación de *watermarking* algunos individuos pueden tener autorización para introducir, detectar y/o eliminar las marcas de agua, mientras que otros pueden tener restringida alguna de esas acciones o todas. Las marcas de agua consideradas como seguras hacen posible que se cumplan las anteriores restricciones.

Los tipos de ataques posibles a un contenido al que se ha aplicado una técnica de *watermarking* se encuentran dentro de una las siguientes categorías: introducción no autorizada, detección no autorizada o eliminación no autorizada. Una introducción no autorizada, o ataque de falsificación, consiste en introducir una marca de agua que sólo el individuo que se encarga de la aplicación de la técnica debería ser capaz de introducir. Una detección no autorizada consiste en detectar marcas de agua que sólo este mismo individuo debería ser capaz de detectar. Finalmente, una eliminación no autorizada trata de eliminar las marcas de agua que nadie debería ser capaz de eliminar. Por un lado, la eliminación e introducción no autorizada son considerados como ataques activos, ya que este tipo de ataques modifica el contenido. Por otro lado, la detección no autorizada no modifica el contenido y se considera ataque pasivo.

No todos los ataques son contra las marcas de agua. Un atacante puede frustar el sistema sin tener que realizar ninguna de las acciones no autorizadas. Los ataques que explotan las debilidades en el uso de las marcas de agua, en lugar de los puntos débiles

de las propias marcas de agua, se conocen como ataques al sistema. Como ejemplo muy simple de un ataque considérese la aplicación de un sistema de control en el que los dispositivos de grabación incluyen un chip para detectar la marca de agua. Un adversario puede abrir la grabadora y eliminar el chip para que la grabadora permita la copia ilegal. En este ataque la seguridad de la marca de agua en sí misma no es relevante.

Dentro de las categorías de acciones no autorizadas sobre un sistema de *wateramarking* existen algunas variaciones. Es posible que un adversario realice una o más partes de una acción no autorizada, incluso si no es capaz de realizar la acción completamente. Por ejemplo, generalmente en la detección de una marca de agua se detecta la presencia de la marca de agua y se decodifica el mensaje que contiene. En algunos sistemas puede que un adversario sea capaz de detectar fácilmente la presencia de la marca pero la decodificación del mensaje sea muy complicada. Dependiendo de la aplicación, estos ataques parciales pueden presentar o no problemas. Por lo tanto, es necesario examinar cada una de las tres categorías de acciones no autorizadas para poder distinguir entre los tipos de ataques posibles.

6.2.1. Introducción no autorizada

El tipo más completo de introducción no autorizada, conocido también como ataque de falsificación, implica que el atacante cree e introduzca sus propias marcas de agua. Para hacer esto, debe crear un nuevo mensaje que identifique un contenido, que incluye ya la marca de agua que indentifica el autor, como suyo, y construir una herramienta para introducirlo. Este tipo de ataque puede intentar prevernirse fácilmente con técnicas criptográficas.

Otro tipo de introducción no autorizada se produce cuando el atacante copia una marca de agua de un determinado contenido y la introduce en otro contenido distinto. Este tipo de ataque se conoce como ataque copia. Un método que puede ayudar a superar este tipo de ataque es la utilización de herramientas criptográficas para la elaboración de las firmas digitales que relacionen la marca de agua empleada con el contenido, como se verá más adelante.

6.2.2. Detección no autorizada

En algunas aplicaciones para las que no sea necesario que la capacidad de detección esté restringida, el principal objetivo es prevenir que se pueda decodificar la marca de agua, lo que se puede conseguir con el uso de herramientas criptográficas.

En otras aplicaciones, para el atacante puede ser suficiente simplemente saber si la marca de agua está presente, y no necesita saber qué mensaje se encuentra codificado. Estrictamente hablando, éste es un problema de esteganografía [COX07], no de *watermarking*. Sin embargo, se puede entender que se opte por una aplicación de *watermarking* en el caso de que el sistema pueda verse comprometido por un adversario que simplemente identifique si una marca está o no presente en un determinado trabajo. Además, detectar la presencia de una marca de agua puede ser una gran ventaja para un adversario que está intentando eliminar marcas de agua. De esta forma, si se tiene que garantizar que una marca de agua es segura contra una eliminación no autorizada, también debe ser segura contra este tipo de detección no autorizada. Sin seguridad contra la detección, la seguridad contra la decodificación no es suficiente.

Un tercer tipo de detección no autorizada, que se encuentra situada entre una mera detección y una completa decodificación, sucede cuando un adversario puede distinguir entre marcas que codifican diferentes mensajes, incluso sin que sepa qué son esos mensajes. Este tipo de detección no autorizada puede tener lugar cuando, dados dos contenidos en los que mediante una determinada técnica, se han introducido marcas de agua, el adversario puede decir de forma fiable si sus respectivas marcas codifican el mismo mensaje o si, por el contrario, codifican mensajes diferentes. Este tipo de ataque presenta un problema si el adversario puede determinar el significado de los mensajes de alguna manera alternativa a la decodificación.

6.2.3. Eliminación no autorizada

En general, se considera que se ha eliminado una marca de agua si ésta se ha hecho indetectable. En el caso más extremo se conseguirá que el contenido con la marca de agua se transforme en otro contenido exactamente igual al original. En todas las aplicaciones de *watermarking* que han de ser seguras contra eliminaciones no autorizadas, es necesario impedir que un adversario pueda recuperar el contenido

original. En la gran mayoría de las aplicaciones, el adversario intentará modificar el contenido en el que se insertó la marca de agua de tal forma que se parezca al original y, sin embargo, no consiga activar el detector de la marca de agua. En estos casos, el contenido original es sólo uno de los posibles resultados de tal acción. De esta forma, demostrar que el adversario no puede recuperar el contenido original es sólo un pequeño paso para demostrar que la marca de agua es segura contra una eliminación no autorizada. El adversario puede utilizar un restricción de fidelidad que exija que el impacto de la distorsión introducida por el ataque sea mínimamente apreciable. Esto es similar a utilizar la restricción para la introducción, pero el modelo perceptual utilizado por un adversario ha de ser diferente al que se utilizó para la introducción de la marca de agua. Normalmente, los requisitos de fidelidad para un adversario son menos estrictos que los de introducción [CHA00, MOU99]. En este caso, la fidelidad es una medida entre el contenido con la marca de agua y el contenido que es atacado.

Dentro del rango de contenidos que el adversario considera válidos como resultado de su acción, se puede distinguir entre aquellos en los que la marca de agua se elimina completamente y aquéllos en los que la marca de agua está todavía presente pero sólo puede ser detectada por un detector más sofisticado. En los primeros se habrían realizado ataques de eliminación y en los segundos ataques de enmascaramiento [COX01]. Una forma muy interesante de eliminación no autorizada es mediante un ataque de colisión [ERG99], que consiste en que el atacante obtiene varias copias de un determinado contenido con diferentes marcas de agua y las utiliza para crear una copia que no incluya marca de agua.

6.3. Consideraciones sobre el atacante

Cada categoría de ataque plantea sus propios desafíos tecnológicos a los diseñadores de técnicas de *watermarking* seguras. En general, no existen propuestas de *watermarking* inmunes a todos los tipos de ataques y en muchas ocasiones deben emplearse varias técnicas simultáneamente para alcanzar el grado deseado de resistencia. Por otro lado, como ya se ha comentado, la importancia de prevenir cada tipo de ataque depende de la aplicación. Resulta de utilidad analizar los requisitos de

seguridad de una aplicación especificando quién está autorizado para realizar determinadas operaciones y qué sabe el atacante que trata de realizar una de las acciones restringidas. Para evaluar de forma general la fortaleza de un sistema de *watermarking* contra los distintos tipos de ataque conviene hacer algunas suposiciones sobre el atacante. Por ejemplo, ¿qué conoce el adversario sobre el algoritmo de *watermarking*?; ¿qué herramientas tiene a su disposición?. En el caso de que esté intentando eliminar la marca de agua, ¿qué sabe sobre la marca insertada? Algunas de las posibles susposiciones sobre el atacante son las siguientes:

- **El atacante no sabe nada.** La suposición más simple es que el atacante no sabe nada sobre el algoritmo y no tiene herramientas especiales, como por ejemplo un detector de marca de agua. Bajo estas circunstancias, el atacante debe confiar en el conocimiento general de las debilidades más frecuentes que sufren los algoritmos de *watermarking* de la aplicación concreta.

- **El atacante dispone de más de un contenido con marca de agua.** En algunos casos, el atacante puede conseguir varios contenidos en los que se han insertado marcas de agua de acuerdo con una determinada técnica. El atacante podría aprovechar ésto para eliminar las marcas de agua, incluso sin saber el algoritmo que se utilizó (ataques de colisión).

- **El atacante conoce el algoritmo.** De forma general, para sistemas que necesiten un alto nivel de seguridad, la suposición de que el atacante no conoce nada sobre el algoritmo de *watermarking* es insegura. En muchas ocasiones resulta difícil que el algoritmo empleado se mantenga en secreto absoluto. Por otro lado, si un algoritmo se mantiene en secreto, sólo un reducido número de investigadores puede estudiar dicho algoritmo. En este caso, los aspectos de seguridad más importantes podrían no tratarse hasta después de que el sistema haya sido atacado. Por estos motivos, la suposición de que el atacante conoce todo lo relacionado con el algoritmo excepto alguna o algunas claves puede llevar a un nivel de seguridad muy alto a la técnica de *watermarking* desarrollada.

- **El atacante tiene un detector.** En todas las suposiciones anteriores se ha tenido en cuenta qué conoce el atacante sobre el algoritmo de *watermarking* pero no se ha hecho una referencia concreta sobre los aspectos relacionados con las herramientas que este individuo tiene a su disposición. Sin embargo, si la

aplicación implica que el atacante debe tener permiso para realizar alguna acción, se debe asumir que éste dispone de las herramientas necesarias para realizar dicha acción. El caso que ha recibido mayor interés es aquel en el que el adversario puede detectar las marcas de agua pero no puede eliminarlas. Incluso si el atacante no tiene información alguna sobre el algoritmo de *watermarking*, el tener acceso a un detector ofrece al atacante una gran ventaja para realizar un ataque.

6.4. Seguridad del *watermarking* y criptografía

En algunas ocasiones, la introducción y detección de la marca de agua se pueden considerar anólogas a la encriptación y desencriptación. En criptografía simétrica, se dispone de una función de encriptación, $E_K(\cdot)$, que utiliza una clave, *K*, y un determinado texto, *m*, para crear un texto encriptado, m_c:

$$m_c = E_K(m) \tag{6.1}$$

El texto encriptado puede ser desencriptado para revelar el mensaje original empleando para ello una función, $D_K(\cdot)$, y la clave correspondiente, *K*:

$$m = D_K(m_c) \tag{6.2}$$

De modo general, se puede considerar que cualquier técnica de *watermarking* utiliza una función para la introducción de la marca de agua, $\varepsilon(\cdot)$, que recibe como entradas un determinado mensaje, *m*, y contenido original, c_o, y da como salida el contenido con la marca de agua, c_w. De modo similar, para la detección de la marca de agua se tiene una función de detección, $\eta(\cdot)$, que recibe como entrada un contenido que incluye una marca de agua y da como salida un mensaje. En los sistemas de *watermarking*, la relación entre los contenidos con la marca de agua y el mensaje se controla con ayuda de una clave, *K*. Para de la detección de la información, se puede considerar un sistema que asocie una única clave con un determinado contenido. De esta

forma, cualquier sistema de *watermarking* puede caracterizarse por las siguientes ecuaciones:

$$\begin{aligned} c_w &= \varepsilon_K(c_o, m) \\ m &= \eta_K(c_w) \end{aligned} \tag{6.3}$$

que son similares a las ecuaciones (6.1) y (6.2).

Con esta analogía entre criptografía y *watermarking*, es razonable pensar que los problemas de introducción y detección no autorizada se pueden solucionar con la aplicación de las herramientas criptográficas adecuadas.

6.4.1. Prevención de detección no autorizada

En muchas ocasiones resulta imposible hacer que un sistema de *watermarking* sea totalmente seguro contra una detección y decodificación no autorizada. El problema de la decodificación no autorizada puede resolverse con una aplicación adecuada de algoritmos criptográficos. En este caso se añade un nivel de encriptación al sistema. La marca de agua se encripta antes de ser introducida en el contenido y se desencripta después de que sea detectada. Este sistema necesita dos claves: la clave de la marca de agua, K_w, que controla el nivel de *watermarking*, y la clave de encriptación, K_c, que controla el nivel de encriptación. De esta forma, la aplicación de la técnica de *watermarking* está de acuerdo con la siguiente expresión:

$$c_w = \varepsilon_{K_w}(c_o, m_c) = \varepsilon_{K_w}(c_o, E_{K_c}(m)) \tag{6.4}$$

Para la detección:

$$m = D_{K_c}\left(\eta_{K_w}(c_w)\right) \tag{6.5}$$

Este sistema se muestra en la figura 6.1. La encriptación forma parte de la capa superior que garantiza que los mensajes sean seguros mientras que el sistema de *watermarking* y el contenido forma parte de la capa inferior, que es la responsable de asegurar que estos mensajes se introduzcan de forma segura.

La encriptación de la información con la que se quiere aplicar una determinada técnica de *watermarking* garantiza la seguridad de la misma. Por ejemplo, un individuo que realice una detección no autorizada de la marca de agua no puede decodificarla y llegar a conocer la información que contiene. Además, en determinados sistemas de *watermarking*, se puede llegar a conseguir que el atacante no sea capaz de detectar la presencia de la marca de agua. Sin embargo, en la mayoría de los sistemas de *watermarking* la encriptación no previene la detección de la marca de agua.

La limitación de la encriptación queda clara con la analogía de capas mostrada en la figura 6.1. La prevención de detección no autorizada sin decodificación debe estar relacionada con la técnica de *watermarking* y su aplicación. En contraste con lo anterior, la encriptación es parte de la capa del mensaje y está relacionada con los aspectos de la confidencialidad del mensaje, autentificación e integridad [SCH96]. De esta forma, es improbable que la aplicación de herramientas criptográficas pueda solucionar los aspectos de seguridad relacionados con la detección no autorizada.

6.4.2. Prevención de introducción no autorizada

Se va a considerar ahora el problema de la introducción no autorizada. Este

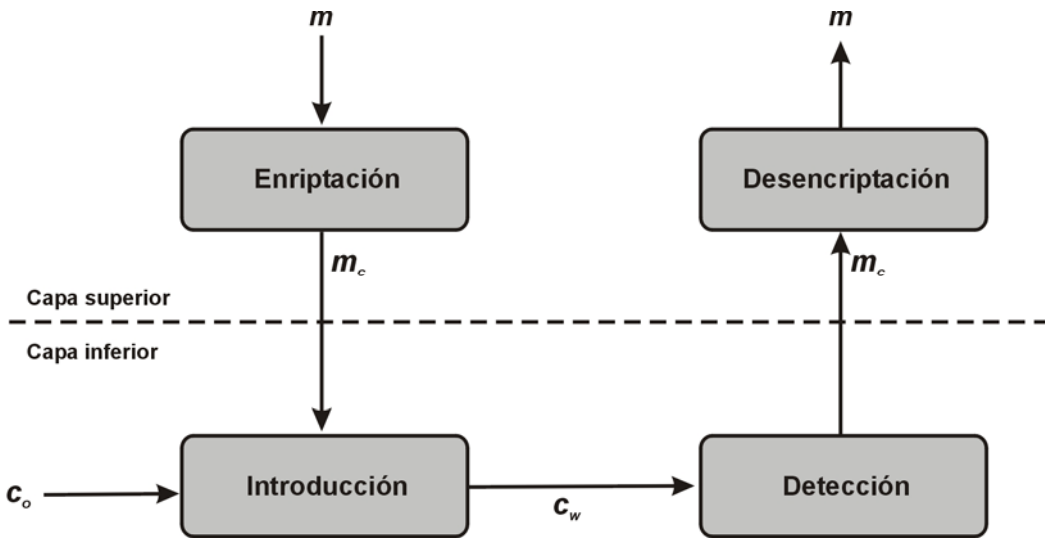

Figura 6.1. Sistema de *watermarking* de 2 niveles.

problema está estrechamente relacionado con el problema criptográfico de autentificación del emisor [COX01], que dependiendo del tamaño de los mensajes, puede resolverse con criptografía. De esta forma, un método robusto para prevenir la introducción no autorizada de una marca de agua usaría la herramienta criptográfica en la capa superior de la figura 6.1. Con este método se consigue prevenir el primero de los ataques de introducción no autorizada considerado. Este ataque consiste en que un adversario elabora un nuevo mensaje y lo introduce en el contenido considerado. Con el método anterior, incluso si el adversario conoce el algoritmo de introducción y la clave que controla la aplicación de la técnica de *watermarking*, K_w, no puede encriptar el mensaje o crear una firma criptográfica válida, a menos que conozca la clave de encriptación.

Sin embargo, esta solución no resuelve el segundo tipo de introducción no autorizada basada en que el adversario pueda encontrar un mensaje válido en un determinado contenido e introducirlo en otro contenido. Una forma de prevenir esta forma de ataque es crear una marca de agua que contenga información sobre el contenido en el que va a ser introducida, y aplicar la función criptográfica. Existen diferentes formas de aplicar esta idea. Por ejemplo, se puede relacionar el mensaje con el contenido completo y, a partir de este mensaje, crear la firma digital mediante las herramientas criptográficas. Sin embargo, este método presenta el problema de que el proceso de introducción modifica el contenido, lo que hace que la firma sea inválida. Para solucionar el problema anterior, se puede relacionar el mensaje sólo con una parte del contenido elegida de forma que el proceso de introducción no la modifique. Los aspectos más importantes relacionados con este método se detallan en [COX01].

6.5. Ataques contra *watermarking* para módulos IP

Tras el estudio sobre la seguridad en sistemas de *watermarking* para contenidos digitales, en este apartado se realiza una particularización para bloques IP. Teniendo en cuenta los tres tipos de acciones restringidas que se han estudiado, en esta sección se analizan los tipos de ataques más importantes contra módulos IP. Se detallan los aspectos más relevantes relacionados con estos tipos de ataques, que servirán para

evaluar de forma más precisa la resistencia que ofrecen las estrategias de *watermarking* propuestas en este trabajo contra cada uno de ellos.

Existen varias formas de atacar un esquema de *watermarking* diseñado para proteger módulos IP. Estos tipos de ataques se encuentran dentro de uno de las tres categorías consideradas: introducción no autorizada, detección no autorizada o eliminación no autorizada. Debido a su importancia en el estudio de la seguridad de las técnicas de *watermarking* para bloques IP, los ataques considerados en este trabajo son los siguientes: falsificación de firmas, firmas fantasma, introducción de una nueva firma, ataques de saboteo y ataques de detección. A continuación se describe cada uno de estos tipos de ataques y se analiza la resistencia que presenta el método de protección propuesto en este trabajo.

6.5.1. Falsificación de firmas

La falsificación de firmas es un tipo de ataque de ambigüedad. Consiste en que el atacante, a partir de la firma digital incluida en el bloque IP que identifica al propietario y/o autor, obtiene información que le identifique como propietario. Para conseguir lo anterior, la información obtenida ha de ser suficientemente convincente como para poder reclamar los derechos de autor sobre el bloque. Para llevar a cabo este tipo de ataque se está suponiendo que el atacante sabe cuál es la firma digital del autor y/o propietario. La aceptación de esta suposición hace que este ataque sea difícil de conseguir, ya que la firma digital incluída en un diseño es un secreto que se mantiene hasta que los derechos de autor se encuentren en cuestión y, por tanto, la única forma de conocerla es mediante una detección no autorizada. Más adelante se analizan los ataques de detección y la resistencia que ofrece el método de *watermarking* propuesto.

Este tipo de ataque se puede prevenir fácilmente mediante el uso de herramientas criptográficas [SCH96]. Como ya se ha mencionado anteriormente, la aplicación de una función criptográfica o función *hash*, transforma la información con la que el autor quiere identificar el diseño en una secuencia fija de bits o firma digital. La transformación conseguida con una función *hash* hace que sea extremadamente difícil conocer la marca de agua a partir de la secuencia de bits obtenida, es decir, realizar la

función inversa. Además, es computacionalmente improbable encontrar dos marcas de agua distintas que correspondan a la misma firma digital.

Para todas las estrategias de *watermarking* desarrolladas se han considerado firmas digitales obtenidas mediante herramientas criptográficas, MD5 ó SHA1. Por lo tanto, se puede asegurar que los módulos desarrollados bajo estas estrategias de protección ofrecen una gran resistencia contra la falsificación de firmas.

6.5.2. Firmas fantasma

Este tipo de ataque consiste en aplicar los recursos necesarios para encontrar una determinada firma en el bloque IP, distinta a la que se utilizó en la protección del bloque, y considerar ésta como la firma que identifica al propietario de dicho bloque. De esta forma, el atacante podría demostrar que el bloque, además de la firma del verdadero propietario, también incluye su firma y podría reclamar así sus derechos. Este tipo de ataque no modifica la circuitería del bloque IP, pero se necesita un gran esfuerzo y tiempo para encontrar una firma que se corresponda con datos que identifiquen de forma significativa al que pretende hacerse pasar como el propietario del diseño. Teniendo en cuenta el tipo de acciones no autorizadas, estos ataques pueden ser considerados como una forma de introducción no autorizada. Sin embargo, normalmente son considerados como un ataque al sistema.

Teniendo en cuenta este tipo de ataque, el organismo competente podría resolver que los derechos de autor del bloque IP pertenecen a aquél cuya firma ofrezca mayor evidencia de validez. De esta forma, la primera acción para defenderse contra estos ataques consiste en utilizar técnicas de *watermarking* que permitan la introducción de firmas que contengan información más relevante que la que pueda encontrar el atacante dentro del mismo bloque IP. De nuevo, para evaluar la fortaleza del método de protección propuesto, se debe tener en cuenta que este método utiliza firmas digitales obtenidas mediante herramientas criptográficas, que transforman cualquier marca de agua en una secuencia de bits de longitud fija. Por lo tanto, la marca de agua utilizada puede incluir toda la información que se necesite para identificar al propietario, ya que, dada una herramienta criptográfica, esto no va a suponer mayor número de bits de la firma digital. Si al tomar la decisión sobre la autoría de un determinado bloque el

organismo competente, además de la relevancia de la información de la firma digital, valorara el número de bits que forman parte de la firma digital incluida en el diseño, se podrían utilizar funciones criptográficas que proporcionen firmas digitales de un número mayor de bits (SHA256, SHA512, etc.). Por otro lado, todos los módulos desarrollados bajo las estrategias de protección propuestas incluyen un método para la extracción de la firma. El hecho de que el *hardware* necesario para dicha extracción forme parte del diseño protegido podría dar mayor credibilidad sobre la firma original que sobre la firma fantasma.

6.5.3. Introducción de otra firma

Este tipo de ataques consiste en introducir una firma digital en un determinado diseño que ya contiene la firma del verdadero autor. La introducción de una nueva firma en el módulo previamente protegido sólo es posible si el atacante dispone de las herramientas adecuadas y utiliza un procedimiento que se lo permita. Además del esfuerzo y tiempo que supone la introducción de una firma en el diseño, para que el resultado de este tipo de ataque sea satisfactorio, es necesario introducir la nueva firma de forma que el circuito siga funcionando correctamente y que no haya impacto sobre el área y las prestaciones, o que éste sea el menor posible.

Si el atacante es capaz de introducir su firma en el diseño, podría intentar demostrar sus derechos de autor. Sin embargo, en este caso, el diseño modificado incluiría la firma del autor y la firma del atacante, mientras que el diseño original sólo incluye la firma del autor. Comparando el diseño modificado por el atacante con el diseño original, el autor podría demostrar sus derechos sobre el primero, ya que la versión original sólo contiene la firma del autor y, posiblemente, sea mejor en cuanto a recursos y velocidad. Por otro lado, al igual que en el caso de ataques basados en falsificación de firmas, podría resolverse que los derechos de autor del bloque IP pertenecen a aquél cuya firma sea más consistente, tanto en la relevancia de la información que contiene la firma digital como en el número de bits de esta firma.

Las estrategias de *watermarking* propuestas en este trabajo incluyen el uso de funciones criptográficas para la creación de las firmas digitales y un método de

extracción que ofrecen al autor del módulo una alta credibilidad cuando los derechos de autor estén siendo cuestionados.

6.5.4. Ataques de sabotaje

Los ataques de sabotajeo ataques *tampering*, incluyen dos de los tipos de acciones restringidas: la eliminación no autorizada y la introducción no autorizada. Por lo tanto, para llevar a cabo este ataque, se tiene que eliminar la firma del propietario legítimo cambiando el diseño original, y se tiene que introducir otra firma.

Existen varios motivos por los que este tipo de ataque es bastante difícil de conseguir. En primer lugar, bajo la suposición de que el atacante desconoce el método o técnica de *watermarking* que se empleó para la introducción de los bits de la firma del propietario, es muy probable que intentando eliminar estos bits el atacante modifique o elimine otros bits que son necesarios para el correcto funcionamiento del diseño. En segundo lugar, el uso de códigos de corrección de errores en la firma digital del propietario permite que el diseño sea capaz de soportar la eliminación de un determinado número de bits de la firma. Por otro lado, la introducción de una nueva firma digital supone un esfuerzo considerable en cuanto a tiempo y herramientas o mecanismos de desarrollo. Por último, suponiendo que el atacante consiga eliminar la firma digital o parte de ella e introducir su firma digital en el diseño, es necesario que realice los procesos de síntesis necesarios, que pueden incrementar de forma importante el número de recursos necesarios y afectar de forma negativa a las prestaciones del diseño.

A pesar de las dificultades que plantea esta forma de ataque, en la sección 6.6 se muestra el estudio estadístico realizado sobre la fortaleza que presenta el método de protección propuesto contra una eliminación no autorizada de la firma.

6.5.5. Ataques de detección

Como ya se ha estudiado anteriormente, la detección no autorizada se puede conseguir atendiendo a tres niveles de rigurosidad. El nivel más severo de detección no autorizada sucede cuando un adversario detecta y decodifica la firma digital del bloque

IP considerado. Con una forma menos severa de ataqu, el adversario puede detectar la firma digital, y distinguir entre diferentes firmas digitales, pero no puede decodificar la información que contienen. Es importante tener en cuenta que, aunque no se pueda decodificar la información de la firma digital, su detección podría ayudar a que un ataque de sabotaje tuviera éxito. En concreto, la detección de la firma digital podría facilitar la localización de las posiciones de los bits de la firma y conseguir su eliminación sin que ésto afecte a otras partes del diseño. De esta forma, el correcto funcionamiento del diseño no se vería afectado y se habría alcanzado uno de los requisitos más importantes de los ataques de sabotaje. Por último, la forma menos severa de este tipo de ataque se produce cuando el adversario es capaz de determinar que la firma digital está incluida en el diseño, pero no es capaz ni de detectarla ni de decodificarla.

En el método de protección propuesto se utilizan herramientas criptográficas para la firma digital, que hacen que la decodificación de la misma sea extremadamente difícil. Por lo tanto, para la evaluación de este método de protección, la primera forma de detección no autorizada quedaría descartada. En la siguiente sección se detalla el estudio realizado sobre la probabilidad de detección de la firma digital en los módulos desarrollados mediante las estrategias de *watermarking* propuestas.

6.6. Probabilidad de la eliminación y detección

Entre las acciones no autorizadas involucradas en los ataques considerados en la sección anterior se encuentran la detección y la eliminación no autorizada. A continuación se realiza un estudio estadístico de la probabilidad de éxito para los ataques de eliminación y detección.

6.6.1. Eliminación de los bits de la firma

La eliminación total o parcial de los bits de la firma es uno de los objetivos de los ataques de sabotaje. En el capítulo 4 se detalló el proceso de difusión de los bits de la firma para las dos estrategias de *watermarking* desarrolladas. Las diferencias de

difusión entre estas dos estrategias hacen que en el estudio de la fortaleza del método de protección propuesto contra la eliminación de los bits de la firma, sea necesario realizar una distinción entre estas dos estrategias.

Para la estrategia de protección EPIBF-LC los bits de la firma se identifican con patrones de salida de la lógica combinacional incluida en la descripción del diseño. En esta estrategia los bits de la firma son bits que el sistema utiliza para un procesamiento normal. De esta forma, resulta imposible eliminar cualquier bit de la firma sin que esto afecte el correcto funcionamiento del sistema, es decir, P_b= 0 para cualquier valor de b. Por lo tanto, con respecto a la estrategia EPIBF-LUT, la estrategia de protección EPBBF-LC consigue un importante aumento de la resistencia contra cualquier tipo de ataques de eliminación. Los ataques de eliminación contra diseños desarrollados bajo la estrategia de protección EPBBF-LC nunca conseguirán ser efectivos.

Considerando la estrategia de protección EPIBF-LUT, la primera dificultad que encuentra el atacante para eliminar los bits de la firma es el desconocimiento del procedimiento seguido para la difusión de la misma. Suponiendo que el atacante conozca que estos bits fueron introducidos como bits de las estructuras de memoria incluidas en el diseño, se encuentra con la dificultad de no saber cuáles de los bits de las estructuras de memoria corresponden a los bits de la firma y cuáles corresponden a información necesaria para el normal y correcto funcionamiento del diseño. En este caso, la probabilidad de que uno de los bits de la firma sea eliminado de forma aleatoria es:

$$P_1 = \frac{f}{m} \tag{6.6}$$

donde f es el número de bits de la firma y m es el número total de bits de memoria incluidos dentro del diseño.

Para conseguir eliminar b bits de la firma, previamente se deben haber eliminado con éxito b–1 bits. A su vez, para eliminar b–1 bits de la firma, previamente se deben haber eliminado con éxito b–2 bits y así sucesivamente. Estos eventos son dependientes y, por lo tanto, la probabilidad de eliminar con éxito b bits de la firma es condicional [RON99] y se puede calcular de acuerdo con la siguiente expresión:

$$P_{b-cambios} = \frac{f}{m} \cdot \frac{f-1}{m-1} \cdots \frac{f-(b-1)}{m-(b-1)} = \frac{\binom{f}{b}}{\binom{m}{b}} \tag{6.7}$$

Para comparar la estrategia de EPIBF-LUT con las desarrolladas en otros trabajos [JAI03, LAC01] se utilizó la ecuación (6.7) en el cálculo de la probabilidad de eliminar un cierto porcentaje de la firma digital incluida en los diseños desarrollados bajo esta estrategia. La tabla 6.1 muestra los resultados obtenidos, siendo $P_{b\%\text{-cambios}}$ la probabilidad de eliminar el *b*% de los bits de la firma. Por otro, la figura 6.2 muestra la probabilidad de eliminar distintos porcentajes de bits de la firma digital MD5 para los diseños protegidos desarrollados a partir del filtro CIC y del filtro FIR. Analizando los resultados de la tabla 6.1 y de la figura 6.2, se puede comprobar que es extremadamente difícil que un atacante pueda eliminar incluso un pequeño porcentaje de la firma. Por ejemplo, las probabilidades de que un adversario pueda eliminar el 10% de los bits de la firma digital MD5 incluida en los diseños de los filtros CIC y FIR es 1.2×10^{-27} y 2.0×10^{-21}, respectivamente.

Por otro lado, es importante determinar si la eliminación de un determinado número de bits puede resultar en un ataque con éxito. Por ejemplo, cuando en la preparación de la firmas digital utilizada por la estrategias de *watermarking* considerada no se incluye corrección de errores, basta con eliminar sólo uno de los bits de la firma para conseguir que el ataque de eliminación cumpla su propósito, de que la firma digital sea irrecuperable. Sin embargo, en el caso de que en la elaboración de la firma se haya utilizado corrección de errores en *t* bits, será necesario eliminar *t*+1 bits de la firma para que ésta no pueda ser recuperada y, por lo tanto, el ataque de eliminación sea un éxito. Así, se va a considerar que para que un ataque de eliminación sea efectivo, se debe eliminar al menos un determinado número de bits de la firma. En el caso de que en la elaboración de la firma digital se hayan utilizado códigos de corrección de errores, este número de bits coincide con los bits de redundancia introducidos por dicho código.

Diseño	Firma	$P_{b\%\text{-}cambios}$ 5%	10%	30%	50%
CIC-FSM	MD5	8.5×10^{-12}	2.0×10^{-21}	9.6×10^{-65}	1.8×10^{-109}
FIR-FSM	MD5	3.9×10^{-15}	1.2×10^{-27}	2.0×10^{-83}	3.7×10^{-140}
	SHA1	6.0×10^{-15}	2.4×10^{-29}	1.5×10^{-86}	4.3×10^{-147}
FIR-LFSR	MD5	8.9×10^{-15}	5.8×10^{-27}	2.1×10^{-81}	7.3×10^{-137}
	SHA1	6.0×10^{-15}	2.4×10^{-29}	1.5×10^{-86}	4.3×10^{-147}
FIR-embedded LFSR	MD5	8.9×10^{-15}	5.8×10^{-27}	2.1×10^{-81}	7.3×10^{-137}
	SHA1	6.0×10^{-15}	2.4×10^{-29}	1.5×10^{-86}	4.3×10^{-147}

Tabla 6.1. Probabilidad de eliminar distintos porcentajes de la firma digital.

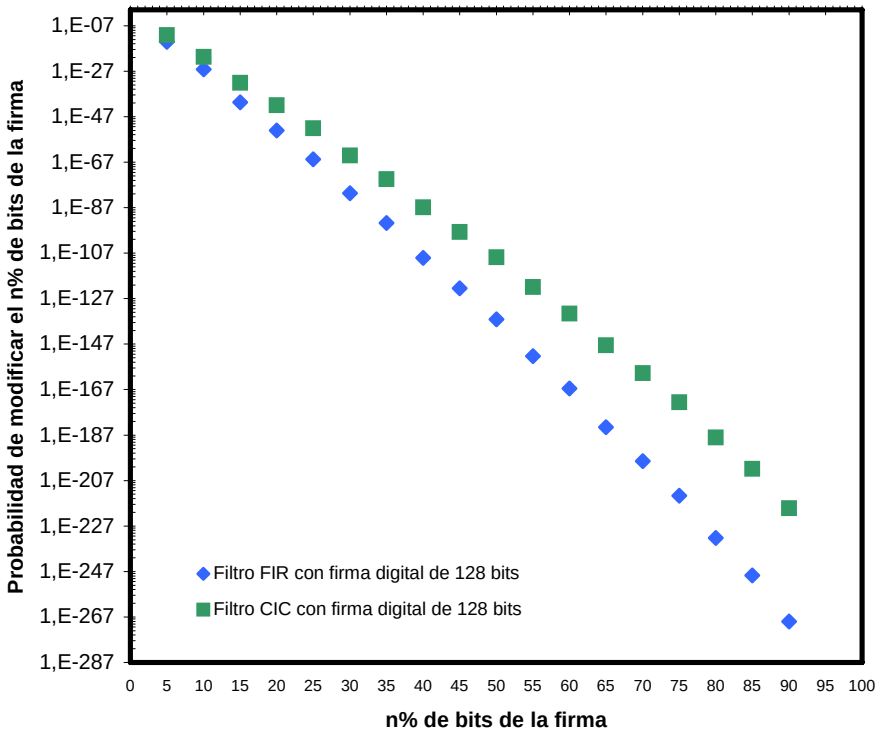

Figura 6.2. Probabilidad de eliminar distintos porcentajes de las firmas digitales introducidas en los diseños.

Suponiendo que se ha utilizado código de corrección de *t* bits para la preparación de la firma, la probabilidad de que *b* bits de la firma sean eliminados y que esta acción haga que la firma digital no se pueda recuperar se calcula con ayuda de la siguiente expresión:

$$P_b = P_{b-cambios} \cdot P_{ECC} \tag{6.8}$$

donde P_{ECC} representa la probabilidad de que no se pueda recuperar la firma digital después de eliminar *b* bits:

$$P_{ECC} = \begin{cases} 1 & if \quad b \geq t+1 \\ 0 & if \quad b < t+1 \end{cases} \tag{6.9}$$

Combinando las ecuaciones (6.5), (6.6) y (6.7) se obtiene:

$$P_b = \begin{cases} P_{b-cambios} = \binom{f}{b} \Big/ \binom{m}{b} & if \quad b \geq t+1 \\ 0 & if \quad b < t+1 \end{cases} \tag{6.10}$$

Por ejemplo, si la firma digital incluye corrección de errores en un solo bit, t=1, para los ataques que traten de eliminar un bit de la firma, b=1, se obtendrá P_{ECC} =0 y P_1=0. Como indica el valor de P_1, para estos ataques la probabilidad de éxito es 0. En estos casos, sera necesario eliminar al menos dos bits de la firma, $b \geq 2$, para que ésta no pueda ser recuperada. Según la ecuación (6.10), la probabilidad de que el intento de eliminar 2 bits de la firma se realice con éxito es de $P_2 = P_{2-cambios}$, ya que para t=1 y b=2, $b \geq t$+1 y por lo tanto P_{ECC}=1.

Como indica la ecuación (6.10), la probabilidad P_b depende del numero de bits de la firma, *f*, del número de bits de corrección posibles, *t*, y del número de bits disponibles en las memorias, *m*. La figura 6.3 representa la probabilidad de conseguir ataques de eliminación considerando diseños que incluyan tablas de consulta de distintas capacidades, y las firmas digitales MD5 y SHA1, con y sin corrección de errores. Para los diseños que incluyen las firmas digitales sin corrección de errores, se ha calculado la probabilidad de que se consiga eliminar un bit de estas firmas. Por otro lado, para los

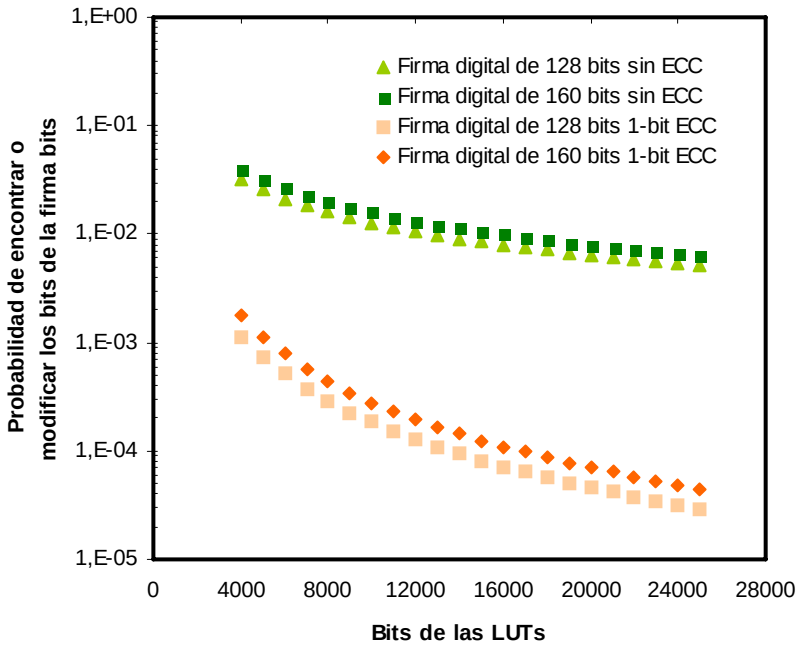

Figura 6.3. Probabilidad de éxito en ataques de eliminación para diseños con LUTs de distinta capacidad desarrollados bajo la estrategia de watermarking EPIBF-LUT.

diseños que incluyen las firmas digitales con corrección de errores en un bit, se ha calculado la probabilidad de que se consigan eliminar dos bits de estas firmas. Se puede comprobar que a medida que aumenta la capacidad de las tablas de consulta, disminuye la probabilidad de ataque de eliminación con éxito y que para las firmas digitales de menor longitud esta probabilidad es menor. Por otro lado, los resultados obtenidos para los diseños que incluyen firmas digitales con corrección de errores en un bit, muestran que para estos diseños las probabilidades calculadas disminuyen con respecto a las probabilidades correspondientes a los diseños que no incluyen corrección de errores. Por lo tanto, la inclusión de corrección de errores en la firma digital aumenta la invulnerabilidad de la misma.

La ecuación (6.10) se utilizó para calcular la probabilidad de conseguir ataques de eliminación con éxito en los diseños desarrollados bajo la estrategia de protección EPIBF-LUT. La tabla 6.2 muestra la probabilidad de que un atacante pueda eliminar un solo bit de la firma digital incluida en los distintos diseños, suponiendo que la firma no incluya corrección de errores. Esta tabla también muestra la probabilidad de conseguir

Diseño	Firma	P_b	
		Sin corrección de errores	Corrección de error en un bit
CIC-FSM	MD5	2.7×10^{-2}	8.1×10^{-4}
FIR-FSM	MD5	8.9×10^{-3}	8.9×10^{-5}
	SHA1	1.1×10^{-2}	1.4×10^{-4}
FIR-LFSR	MD5	8.9×10^{-3}	8.9×10^{-5}
	SHA1	1.1×10^{-2}	1.4×10^{-4}
FIR-embedded LFSR	MD5	8.9×10^{-3}	8.9×10^{-5}
	SHA1	1.1×10^{-2}	1.4×10^{-4}

Tabla 6.2. Probabilidad de que la firma digital introducida en los diseños desarrollados bajo la estrategia de watermarking EPIBF-LUT sea irrecuperable tras un ataque de eliminación.

ataques con éxito en aquellos diseños que incluyan firma digital con corrección de errores en un bit. Comparando los resultados obtenidos para estos últimos diseños con los obtenidos para los diseños que no incluyen corrección de errores, se puede comprobar que las probabilidades de ataque con éxito se reducen en un factor de 100. De esta forma, el uso de códigos de corrección de errores para la firma difgital consigue aumentar la invulnerabilidad contra los ataques de eliminación.

6.6.2. Invulnerabilidad de la secuencia de extracción de la firma

La detección de la firma digital puede ayudar a establecer la ubicación de estos bits dentro de un determinado diseño y, por lo tanto, facilitar su eliminación sin que esto afecte al correcto funcionamiento del sistema. En los diseños desarrollados la detección de la firma digital por parte de un atacante supone un gran esfuerzo. Por un lado, se tiene que conocer que el procedimiento para la detección consiste en aplicar una determinada secuencia de datos en la entrada del diseño para obtener la firma como la secuencia de salida correspondiente. Además, el atacante debe saber cuál es exactamente la secuencia de extracción de la firma. Suponiendo que sabe el

procedimiento pero desconoce la SEF, el ataque de detección consistiría en encontrar esta secuencia de extracción.

La probabilidad de aplicar la secuencia de extracción de la firma de forma aleatoria es la siguiente:

$$P_{SEF} = \frac{1}{\left(2^n\right)^l} \tag{6.11}$$

donde l es el número de palabras que componen la SEF y n el número de bits de estas palabras. Esta probabilidad corresponde también a la probabilidad de que un atacante pueda encontrar la SEF, pero bajo la suposición de que éste conoce la longitud de esta secuencia, es decir, conoce l. Esto es demasiado optimista, ya que *a priori* el atacante sólo conoce el procedimiento de extracción de la firma pero no conoce ni los datos ni la longitud de la secuencia de extracción.

Como indica la expresión (6.11), para que la detección de la firma por pura coincidencia o por parte de cualquier atacante sea muy difícil de conseguir es necesario que el valor de la probabilidad P_{SEF} sea tan pequeño como se pueda. Se consigue así una elevada invulnerabilidad de la SEF. Para esto se puede elegir l tan grande como sea posible sin aumentar de forma excesiva la complejidad y recursos necesarios para el *hardware* de extracción de la firma. Una buena opción para aumentar la invulnerabilidad de la SEF es utilizar un generador de secuencias pseudoaleatorias, por ejemplo el basado en un determinado LFSR.

Las estrategias de *watermarking* desarrolladas proponen la generación manual de la secuencia de extracción de la firma y la generación de esta secuencia mediante un LFSR. Para mostrar la invulnerabilidad de las secuencias de extracción de la firma utilizadas, se ha realizado un estudio para los ejemplos de diseño correspondientes a los filtros CIC y FIR. La tabla 6.3 muestra la probabilidad de que la secuencia de extracción de la firma sea aplicada aleatoriamente en la entrada del correspondiente diseño. Las probabilidades se han calculado mediante (6.11) y teniendo en cuenta que las entradas son de 8 y 16 bits para los filtros CIC y FIR, respectivamente. Para la SEF generada manualmente la longitud de esta secuencia es 6 y las probabilidades calculadas son 3.5×10^{-15} para el filtro CIC y 1.3×10^{-29} para el filtro FIR. Para aumentar la

Diseño	Entrada de datos	Longitud de la SES	Firma	P_{SES}
CIC-FSM	8-bits	6 palabras	MD5	3.5×10^{-15}
FIR-FSM	16 bits	6 palabras	MD5	1.3×10^{-29}
			SHA1	1.3×10^{-29}
FIR-LFSR	16-bits	100 palabras	MD5	$< 1.0\times10^{-300}$
			SHA1	$< 1.0\times10^{-300}$
FIR-embedded LFSR	16-bits	100 palabras	MD5	$< 1.0\times10^{-300}$

Tabla 6.3. Invulnerabilidad de la SES para los diseños desarrollados bajo las estrategias de protección EPIBF-LUT y EPBBF-LC.

invulnerabilidad de la SEF, se generó una secuencia de extracción de longitud 100 con ayuda de un LFSR. Como muestra la tabla 6.3, para los diseños *FIR-LFSR* y *FIR-LFSR embedded*, la probabilidad de activar el proceso de extracción de la firma por coincidencia es inferior a 1×10^{-300}, reflejando el importante aumento de la invulnerabilidad de la SEF que se consigue en los diseños con secuencias de extracción de la firma de longitudes elevadas, que se se pueden generar fácilmente con la ayuda de un LFSR. Analizando todos los resultados, se puede concluir que los diseños desarrollados bajo las estrategias de protección propuestas ofrecen una elevada invulnerabilidad de la SEF, en especial aquéllos que utilizan un LFSR para la generación de la SEF. La elevada invulnerabilidad de la SEF hace que la resistencia contra los ataques de detección también sea muy elevada.

6.7. Comparación con otras alternativas

La principal novedad introducida por el esquema de *watermarking* propuesto reside en que la difusión de los bits de la firma no necesita de recursos adicionales y que el mismo diseño incluye un método para la extracción de la firma, con un impacto mínimo sobre el área y prestaciones del diseño original. Los resultados de la evaluación

realizada muestran que este esquema ofrece una prueba de autor muy fiable, supone un coste bajo en cuanto a esfuerzo, tiempo y gasto computacional, no modifica el correcto funcionamiento del bloque IP y la extracción es fácil, no destructiva y no revela la localización de los bits de la firma. Además de esta evaluación basada en los criterios más importantes para *watermarking* en módulos IP, es muy interesante comparar el esquema desarrollado con algunos de los trabajos realizados por otros investigadores en este campo y que ya fueron analizados en 3.6.3.

Nuestra propuesta para la protección de módulos reutilizables se ha evaluado para firmas de 128 y 160 bits, que son de mayor longitud que las utilizadas en la mayoría de los ejemplos de aplicación propuestos en otras técnicas de *watermarking* [CUI06, FAN03]. Considerando la forma de ataque de firma fantasma, las propuestas de [CUI06, FAN03] ofrecen menor prueba de autoría que la que ofrece la técnica propuesta en este trabajo. Por otro lado, las firmas digitales se obtienen mediante funciones criptográficas *hash*. Como ya se ha analizado anteriormente, la utilización de funciones criptográficas para la elaboración de la firma digital permite la prevención de distintos tipos de ataques, entre ellos del ataque de falsificación. Otras propuestas [YUA04, CUI06, FAN03] no utilizan firmas digitales obtenidas mediante funciones *hash* y, por lo tanto, serán más vulnerables ante algunos de los tipos de ataques considerados.

El hecho de que en la ténica de *watermarking* propuesta, la difusión de los bits de la firma se realice utilizando elementos que forman parte del módulo original, estando estos bits entrelazados con los bits necesarios para un funcionamiento normal de este módulo, hace que sea extremadamente difícil que un atacante pueda econtrar y/o eliminar estos bits. Además, para la estrategia de *watermarking* EPBBF-LC, es imposible eliminar un solo bit de la firma sin que esto afecte al correcto funcionamiento del diseño. De esta forma, la nueva propuesta consigue superar los problemas que presentan otros métodos basados en la utilización de tablas o elementos lógicos que no están en uso en el diseño original [LAC01, KAH98a]. Estos problemas están relacionados con la posibilidad de detectar la inactividad de las partes del circuito que incluyen los bits de la firma y que podrían indicar a un adversario su ubicación.

La difusión se realiza durante la etapa de descripción de alto nivel del diseño; esto contribuye al incremento en la invulnerabilidad de esta propuesta cuando se compara con otras alternativas [CAL04, NEW02, JAI03] ya que los procesos de emplazamiento

y rutado se realizan sin esfuerzo adicional y hacen que la firma quede totalmente integrada dentro del diseño.

El análisis anterior compara la técnica de *watermarking* propuesta en este trabajo con otras alternativas; la mayor parte de ellas corresponden a técnicas de *watermarking* en el nivel de diseño físico. Sin embargo, conviene hacer otro análisis en el que se compare con otras propuestas que aparecen en la literatura para descripciones de nivel alto. Como ya se describió en la sección 3.6.3.3, en [FAN03] se propone un circuito generador de la marca de agua que se introduce dentro del módulo IP durante la descripción comportamental del mismo. Este circuito generador se incluye dentro del diseño como una parte independiente del mismo, hecho que hace que este método ofrezca menor seguridad que el propuesto en este trabajo. En [YUA04] se proponen tres técnicas de *watermarking* para la protección de módulos HDL. Para la primera técnica no se muestran resultados experimentales, mientras que para las otras dos se muestran resultados que utilizan firmas digitales de 15, 16, 20 y 56 bits. Estas firmas no se obtienen mediante herramientas criptográficas, de forma que no ofrecen toda la protección posible contra ataques de eliminación o ataques de falsificación. Por otro lado, son de menor longitud que las empleadas en los ejemplos de diseño propuestos en este trabajo y, por tanto, ofrecerían menor prueba de autor.

La diferencia entra las firmas digitales empleadas por las distintas propuestas que se encuentran en la literatura hacen muy difícil una comparación cuantitativa. Esto se debe a que para cada técnica de *watermarking*, el uso de firmas digitales de diferente longitud, además de dar lugar a diseños con diferentes niveles de seguridad, puede hacer que haya diferencias muy notables en el incremento del área y reducción de las prestaciones. Por este motivo, estrictamente sólo sería posible una comparación cuantitativa con aquellas técnicas que muestren resultados experimentales con firmas digitales de igual longitud.

Por último, las ténicas de *watermarking* mencionadas se aplican en diferentes etapas del flujo de diseño, lo que genera una gran variedad de resultados de implementación y una limitación de las tecnologías para las que los diferentes esquemas pueden realizarse. Una importante característica de la nueva propuesta es su independencia de la tecnología usada para la implementación e incluso del flujo de diseño, ya que la introducción de la firma digital se realiza mediante la estrategia

seleccionada en la descripción HDL del sistema. Todos los aspectos considerados sobre la nueva técnica propuesta y el análisis de las ventajas sobre otras alternativas que se encuentran en la literatura hacen que la ausencia de una comparación cuantitativa no sea imprescindible para asegurar la consolidación y, al mismo tiempo, la seguridad que esta técnica presenta.

6.8. Conclusión

En este capítulo se han estudiado los aspectos más importantes relacionados con la seguridad de un sistema de *watermarking*. En este contexto, la seguridad hace referencia a la resistencia que ofrece el sistema de *watermarking* contra aquellas acciones que traten de frustar su propósito. En general, cualquier sistema de *watermarking* necesita ser seguro contra una introducción, detección y eliminación no autorizadas. El uso de herramientas criptográficas puede ayudar a asegurar la integridad de varios aspectos relacionados con la marca de agua. El estudio de la seguridad para contenidos digitales ha servido para establecer la base necesaria en la evaluación de la seguridad de las técnicas de *watermarking* para módulos IP. Se han descrito los tipos de ataques más relevantes y analizado la resistencia que ofrece el método de *watermarking* propuesto en este trabajo contra cada uno de ellos. Para la estrategia EPIBF-LUT los estudios realizados muestran que la probabilidad de que se pueda conseguir eliminar una pequeña parte de la firma es bastante reducida. Por otro lado, las probabilidades de ataque con éxito se pueden recudir potencialmente utilizando códigos de corrección de errores para la firma digital. La estrategia EPBBF-LC supone un aumento muy importante de la seguridad contra los ataques de eliminación, ya que no es posible eliminar tan sólo un bit de la firma sin que esto afecte al correcto funcionamiento del módulo. Ambas estrategias incluyen un proceso de extracción de la firma que ofrece una gran resistencia contra detecciones no autorizadas. Por último se ha comparado la técnica de *watermarking* desarrollada con otras propuestas de *watermarking*. Se ha confirmado que ésta última ofrece numerosas ventajas frente a estas otras en muchos aspectos, especialmente en aquellos relacionados con la seguridad.

Capítulo 7: Conclusiones

En este capítulo se enuncian las principales aportaciones y conclusiones de esta memoria. Concretamente, se enumerarán los resultados más destacados que se han obtenido con las estrategias de *watermarking* propuestas. Por último, se comentan las líneas de investigación futuras que han quedado abiertas con este trabajo.

7.1. Introducción

Las estrategias basadas en módulos reutilizables proporcionan la optimización de los recursos de las compañías debido a reducidos tiempos y costes de desarrollo. Con la disponibilidad de nuevas arquitecturas FPGA y sus herramientas de diseño es posible emplear una metodología de diseño basada en la reutilización tanto para ASICs como para FPGAs. Sin la reutilización de módulos, la industria electrónica no sería capaz de afrontar el desafío de fabricar dispositivos cada vez mejores, más rápidos y más baratos. Los bloques IP están diseñados para ser modulares e integrados con otros componentes, y resulta sencillo que alguien pueda llegar a vender estos bloques como si fuesen de su propiedad sin tener que conocer la arquitectura o la implementación. Existe, pues, una necesidad de alguna técnica que permita el intercambio seguro de diseños.

El objetivo de la investigación recogida en esta memoria ha sido desarrollar un nuevo método de *watermarking* para la protección de la propiedad intelectual de módulos reutilizables. En concreto, se han desarrollado dos estrategias de *watermarking* para módulos HDL y una herramienta automática para facilitar el proceso de difusión de los bits de la firma digital y optimizar los recursos necesarios por la lógica de extracción. Esto ha permitido obtener resultados interesantes en este campo y ha abierto

nuevas líneas de trabajo durante el periodo de realización de esta investigación, que también se recogen en esta memoria.

7.2. Consideraciones

La revisión exhaustiva de la bibliografía disponible ha permitido llegar a una serie de interesantes conclusiones que han servido de base al trabajo realizado:

1. Comparado con las metodologías de diseño convencionales, el diseño basado en la reutilización presenta numerosas ventajas, entre las que destacan los reducidos tiempos y costes de desarrollo, la importante reducción de la complejidad y la alta productividad. Sin embargo, existen varios problemas asociados a la metodología de diseño basada en la reutilización que no se habían considerado hasta su aparición. Uno de estos problemas es el de la protección de la propiedad intelectual de los módulos IP a compartir.
2. En esta memoria se han descrito algunas de las opciones más representativas entre las existentes en la literatura para ilustrar el amplio abanico de posibilidades disponibles como formas de protección para la propiedad intelectual, que incluyen patentes, derechos de autor, marcas registradas y secretos comerciales. Sin embargo, como se ha analizado en esta memoria, estos métodos no son aplicables para la protección de la propiedad intelectual de módulos reutilizables o no consiguen los objetivos fundamentales de dicha protección. Para que un método de protección de la propiedad intelectual sea efectivo en contenidos como módulos reutilizables, dicha protección tiene que realizarse al mismo tiempo que los procesos de diseño e implementación, ya que durante esos procesos el diseñador posee un control absoluto sobre el módulo que es imposible tener cuando estos procesos finalizan y el módulo esté listo para ser utilizado.
3. En los últimos años se ha experimentado un rápido crecimiento y desarrollo de técnicas de *watermarking* para módulos IP y actualmente es una de las técnicas más utilizadas, ya que puede aplicarse durante los procesos de diseño e implementación. Estas técnicas están basadas en la introducción y ocultación de

una firma digital en un determinado nivel de abstracción del diseño, utilizando los rasgos intrínsecos y la estructura de dicho nivel, que consigue objetivos tan importantes como la identificación del autor y/o propietario y la posibilidad de poder reclamar los derechos de autor.

4. Aunque se han propuesto numerosas técnicas que ocultan una firma digital en datos multimedia, estas no ofrecen una solución adecuada para la protección de la propiedad intelectual de bloques IP. Esto se debe a que para datos multimedia la marca de agua se puede introducir realizando unos ligeros cambios en dicho contenido. Sin embargo, en la aplicación de técnicas de *watermarking* para módulos IP hay que ocultar la marca de agua dentro del módulo al mismo tiempo que se está realizando su diseño y sin que los cambios que puedan tener lugar en dicho módulo afecten a su funcionamiento.
5. La revisión bibliográfica ha permitido obtener una perspectiva actualizada de las diferentes propuestas para las técnicas de *watermaking* en bloques IP. Se ha podido comprobar que, aunque se han realizado muchas investigaciones en el nivel de diseño físico, se ha dirigido poco interés hacia otros niveles de descripción más elevados. La ventaja de utilizar técnicas de *watermarking* en un alto nivel de descripción reside en la dificultad de eliminar la marca de agua, ya que ésta se introduce en etapas preliminares y es arrastrada a través de todo el diseño, y la facilidad de introducir esta marca como una parte funcional del circuito. Sin embargo, existen algunas desventajas que estan relacionadas con las posibles restricciones introducidas en el proceso de diseño y con la repercusión en los recursos del diseño original.

7.3. Principales aportaciones

La investigación realizada se ha traducido en una serie de interesantes aportaciones en el campo de la protección de bloques IP que se han expuesto en esta memoria:

1. En una primera fase, las técnicas de protección se basaron en la introducción de los bits de una firma digital en posiciones de memoria que no han sido utilizadas y que forman parte de las tablas de consulta del circuito. El hecho de que las tablas estén en uso no incrementa el área del sistema, a diferencia de otras técnicas previas, y dificulta enormemente la posibilidad de que un atacante pueda hallar dichos bits.

2. Este esquema de protección incluye también un método seguro para la extracción de la firma del circuito, de forma que las modificaciones necesarias sobre el circuito no alteran de manera significativa ni el área ni las prestaciones del mismo.

3. Los circuitos basados en el RNS que se implementan sobre FPGAs suelen hacer un uso extensivo de las tablas incluidas en los dispositivos programables. Esto hace que la idea propuesta como técnica de protección sea directamente aplicable. De este modo, en primer lugar, se ha trabajado en el desarrollo de mecanismos para la protección de la propiedad intelectual basado en la inclusión de una firma digital que aprovecha las particularidades del RNS y los dispositivos lógicos programables.

4. Los resultados previos mostraron que las estructuras de protección resultantes presentaban un impacto mínimo sobre el área y prestaciones del sistema y una buena robustez ante ataques. De hecho, las primeras ideas desarrolladas proporcionaron resultados muy satisfactorios reflejadas en algunas publicaciones [PAR04, CAS05]. Estas ideas preliminares para la difusión de los bits de una firma digital y de la aplicación directa a sistemas basados en el RNS constituyen la primera de las estrategias de *watermarking* propuestas, llamada EPIBF-LUT.

5. Tras evaluar los mecanismos desarrollados en sistemas con estructuras de memoria que disponen de posiciones libres e implementados sobre dispositivos programables, surgió la necesidad de extender la aplicación de las técnicas desarrolladas para la protección de la propiedad intelectual a sistemas digitales genéricos, tanto para tecnologías programables como para ASICs. Sin embargo, para sistemas digitales genéricos normalmente no hay

posiciones de memoria libres y es necesario buscar otra alternativa para la difusión de los bits de la firma digital.

6. Con este propósito se ha desarrollado la segunda de las estrategias de *watermarking*, llamada EPBBF-LC. Esta estrategia trata de buscar o identificar fragmentos de la firma digital en patrones de salida de la lógica combinacional incluida en el diseño original. De esta forma, para acceder a los bits de la firma basta con aplicar los correspondientes patrones de entrada. También en este caso se han obtenido resultados que demuestran la viabilidad de esta estrategia de protección para sistemas digitales en general, como se puede ver en algunas de las publicaciones derivadas [CAS06a, CAS06b, CAS07a], confirmándose así las amplias posibilidades que se abren en este campo.

7. Un paso más en la evolución de la estrategia de *watermarking* EPBBF-LC ha consistido en facilitar la búsqueda de los bloques de bits de la firma dentro del diseño. Con este propósito se ha desarrollado una herramienta *software* que realiza una búsqueda automática de los bloques de bits de la firma, identificando éstos con ciertos patrones de salida de la lógica combinacional incluida en el diseño a proteger. Tras esta búsqueda, la herramienta automática genera los patrones de entrada que habrá que aplicar para acceder a los bloques de bits de la firma. En general, no existe una solución única, es decir, un determinado bloque de bits de la firma podría ser identificado con varios patrones de salida. Por lo tanto, para el conjunto de bloques de bits existirán varias soluciones posibles para la difusión de los bits de la firma.

8. El primer algoritmo que se utilizó para el desarrollo de la herramienta automática no ha tenido en cuenta este conjunto de soluciones, y ha elegido la primera de las soluciones obtenidas. Sin embargo, resulta muy interesante desarrollar un algoritmo para elegir la mejor solución.

9. De este modo, se han desarrollado otros dos algoritmos que tienen en cuenta una función de coste para realizar la difusión de los bits de la firma. Esta función de coste es la distancia Hamming entre las posiciones de la firma de bloques de bits adyacentes, que consigue optimizar los recursos necesarios para la extracción de la firma.

10. Teniendo en cuenta lo anterior, el segundo algoritmo desarrollado realiza una optimización de la función de coste para los dos primeros bloques de bits de la firma, y para el resto el algoritmo optimiza la solución de un bloque con la solución fija del anterior. Este algoritmo consigue una buena optimización, pero el problema es el elevado tiempo de ejecución cuando aumenta el tamaño de bloque considerado.

11. Por otro lado, la elección de la mejor solución se puede considerar como un problema de lógica combinatoria, una de las aplicaciones más frecuentes de técnicas de Enfriamiento Simulado. De esta forma, el tercero de los algoritmos desarrollados para la herramienta automática está basado en Enfriamiento Simulado, que consigue optimizar la función de coste empleando tiempos de ejecución considerablemente reducidos, incluso para tamaños de bloque elevados.

12. Los resultados obtenidos han demostrado que las estructuras de protección desarrolladas presentan un impacto mínimo sobre el área y las prestaciones del sistema. Además, la evaluación de las estrategias de protección desarrolladas con respecto a otros criterios de evaluación como la prueba de autor, el coste, el correcto funcionamiento, la fácil extracción, la robustez y la seguridad ha dado resultados muy satisfactorios.

13. Dada la importancia de la seguridad de cualquier sistema de *watermarking*, en este trabajo de tesis se ha realizado un estudio exhaustivo sobre la invulnerabilidad de los esquemas de protección obtenidos frente a los diferentes tipos de ataque, obteniéndose resultados muy satisfactorios. Con este estudio se han conseguido dos objetivos importantes:

    - disponer de procedimientos preliminares para medir la fortaleza de los esquemas de *watermarking* a través de una serie de parámetros;
    - establecer algunos puntos fuertes y otros a mejorar en los esquemas de protección con el fin de aumentar su resistencia contra ataques.

7.4. Líneas de investigación futuras

Las ideas desarrolladas en este trabajo han dejado abiertas amplias e interesantes líneas de investigación que podrían representar la continuación del trabajo recogido en esta memoria:

1. El desarrollo de la herramienta automática para el proceso de difusión de los bits de la firma, que facilita la aplicación de la estrategia de *watermarking* EPBBF-LC, ha conseguido ventajas tan importantes como la generación automática de la posiciones de la firma y la optimización del número de recursos necesarios por la lógica de extracción. Uno de los algoritmos utilizados como parte fundamental de la herramienta automática es el Enfriamiento Simulado. Los resultados conseguidos para este algoritmo se han obtenido recientemente, por lo que es posible profundizar en el desarrollo de este algoritmo, en concreto en el ajuste experimental de los parámetros del mismo para conseguir mejores resultados, sobre todo para tamaños de bloque más pequeños. Por otro lado, también sería muy interesante el desarrollo de una herramienta automática más compleja que, además de utilizar el algoritmo para la difusión y la optimización de recursos, sea capaz de generar automáticamente toda la lógica de extracción necesaria descrita en VHDL, que posteriormente será incluida en el diseño a proteger. Esta lógica de extracción dependerá de un conjunto de parámetros de entrada y de las posiciones de la firma obtenidas por el algoritmo de difusión que la propia herramienta incluye.

2. Se ha demostrado que la aplicación de las estrategias de *watermarking* propuestas permite el desarrollo de módulos reutilizables con protección de la propiedad intelectual. Es posible profundizar en la aplicación de estas estrategias en diseños que actualmente presentan gran interés, como los microprocesadores para *hardware* reconfigurable. Resultaría muy interesante llevar de forma paralela al desarrollo del microprocesador la aplicación de las estrategias de *watermarking* propuestas, o la exploración y el desarrollo de nuevas alternativas para permitir el uso e intercambio seguro de dichos microprocesadores como módulo reutilizable.

3. Las estrategias de protección desarrolladas se evaluaron en términos de su invulnerabilidad frente a ataques. Hasta ahora no se ha diferenciado entre ataques directos y los llamados ataques laterales [KOC99]. Los ataques que se han considerado hasta el momento son los ataques directos, por ejemplo, el sabotaje o *tampering*, que intenta alterar el diseño a nivel digital con el fin de destruir la firma digital embebida, impedir su extracción o tratar de falsear un proceso de extracción de una supuesta firma falsa. Por otra parte, el concepto de ataque lateral, o *Side-Channel Attack* (SCA), está derivado del análisis de procesadores criptográficos en los que cierta información, como el tiempo de cómputo, el consumo de potencia o las emisiones electromagnéticas, puede aprovecharse por un atacante para extraer la clave del sistema. De este modo, una de las líneas investigación de interés que quedaría abierta es el estudio de la invulnerabilidad de los esquemas de protección desarrollados frente a este tipo de ataques laterales.

7.5. Conclusión

Con la presentación de las principales aportaciones y el esbozo de las líneas de investigación que aún permanecen abiertas finaliza la exposición de la investigación desarrollada presentada en esta memoria. El objetivo con el que se inició esta investigación fue tratar de ofrecer mayor viabilidad a las metodologías de diseño basadas en la reutilización, asegurando que se puedan comprobar y reclamar los derechos de autor sobre los componentes utilizados en estas metodologías; de este modo, se han presentado de manera estructurada las principales características del diseño basado en la reutilización, de las técnicas de protección para módulos IP, en concreto de las técnicas de *watermarking*, y de los trabajos realizados hasta el momento en este campo de investigación tras lo que se han expuesto los resultados del trabajo desarrollado, que se resumen en este capítulo final.

Conclusions

This chapter presents the main achievements and conclusions obtained in this research. Concretly, the most important results obtained by the proposed watermarking strategies are enumerated. Finally, future research lines and possible improvement are also commented.

Introduction

The strategies based on reusable modules optimize company resources due to the reduced development time and costs. With the availability of new FPGA architectures and FPGA design tools it is now possible to employ design reuse methodologies for both ASICs and FPGAs. Without reuse, the electronics industry would not be able to keep on with the challenge of delivering the better, faster and cheaper devices that consumers expect. IP cores are designed to be modular and capable of being integrated with other components, and it is possible for a third party to sell an IP block as their own, once the interface and function are understood, without even knowing neither the internal architecture nor the implementation. In order to avoid such situations, it is necessary to provide authors with mechanisms for claiming their intellectual property rights, also allowing the safe exchange of designs.

The aim of the research presented in this memory has been to develop a new watermarking method for the intellectual property protection of reusable modules. Concretly, two watermarking strategies for HDL cores and an automated tool, which eases the spreading process of the digital signature bits and optimizes the necessary resources for the extraction logic, have been developed. This has allowed obtaining

interesting results in this field and has opened new research lines during the period of research, which also are presented in this Thesis.

Related work

The exhaustive review of the bibliography has allowed getting interesting conclusions that have been used as a solid platform for the related work:

1. Compared tohe conventional design methodologies, the reuse-based design presents numerous advantages, among them the reduced development time and costs, an important complexity reduction and high productivity. Nevertheless, there are several problems associated with the reuse-based design methodologies that had not been considered up to its appearance. One of these problems is the intellectual property protection of those shared modules.
2. In order to illustrate the wide range of available possibilities for the intellectual property protection, some of the most representative proposals among the existing ones in the literature have been described in this memory, which includes patents, copyrights, registered brands and commercial secrets. However, as it has been analyzed, these methods are not applicable to the intellectual property protection of reusable modules or do not accomplish the fundamental objectives of such protection. For an IPP method to be effective in reusable modules, the protection has to be realized at the same time as the design and implementation processes, since during these processes the designer absolutely controls the module in such a way that is not possible to attain once these processes are finished and the module is ready to be used.
3. In the last years, a rapid growth and development of watermarking techniques for IP cores has been experienced, and nowadays it is one of the techniques most employed, since it can be applied during the design and implementation processes. These techniques are based on the embedding and hiding of a digital signature at a certain abstraction level of the design, using the intrinsic features and the structure of such level; thus, important objectives such as the

identification of the author and the possibility of being able to claim the copyrights are attained.

4. Although numerous techniques that hide a digital signature in multimedia content have been proposed, these do not offer a good solution for the protection of IP cores. This is due to the fact that for multimedia content the watermark can be embedded making some changes to the content itself. However, in the application of watermarking techniques to IP cores it is necessary to hide the watermark inside the module at the same time it is being designed and without letting the changes that could take place in the module to modify its correct operation.
5. The bibliographic revision has allowed obtaining an updated perspective of the different proposals for watermaking techniques of IP cores. Though many researches have been carried out at the physical design level, little interest has been directed to other higher description levels. The advantage provided by watermarking techniques at high design levels resides in the difficulty to remove the watermark, since it is embedded in preliminary stages and is dragged through the whole design flow, besides the facility of embedding this watermark as a functional part of the design. Nevertheless, there are some disadvantages related to restrictions introduced in the design process and the impact in the resources of the original design, which the watermarking technique proposed in this work has overcome.

Main contributions

This research has resulted in a set of interesting contributions in the field of IP core protection that have been presented in this memory:

1. Initially, the protection techniques were based on the embedding of the bits of a digital signature into non-used cells of memory structures included and described in the HDL code of the design. The fact that memory structures are in use does not increase the area of the system and makes enormously difficult the possibility of an attacker finding these bits.

2. This protection scheme also includes a secure method for signature extraction, and the necessary modifications have a minimum impact on the design area and performance.

3. RNS-based circuits implemented over FPGAs have traditionally made extensive use of memory structures included in the programmable devices. This makes the proposed protection technique directly applicable to these circuits. Thus, first contributions were based on the development of mechanisms for the intellectual property protection through the embedding of a digital signature that takes advantage of the particularities of the RNS and programmable logic devices.

4. The previous results showed the developed protection structures to have a minimal impact on the system area and performance and a strong resistance against attacks. In fact, the first developed ideas provided very satisfactory results reflected in some publications [PAR04, CAS05]. These preliminary ideas for the spreading of a digital signature and its application to RNS-based systems constitute the first one of the proposed watermarking strategies, called EPIBF-LUT.

5. After evaluating the developed mechanisms in systems with memory structures that have unused positions and are implemented over programmable devices, there arose the need to extend the application of the developed technique for the intellectual property protection of digital generic systems, for both programmable technologies and ASICs. However, for generic digital systems usually there are no unused memory positions and it is necessary to look for another alternative for the spreading of the digital signature bits.

6. Regarding this last item, a new watermarking strategy has been developed, called EPBBF-LC. This strategy tries to look for or to identify blocks of the digital signature into output patterns of the combinational logic included in the original design. Thus, for extracting the signature bits, the corresponding input patterns have to be applied in a correct order by means of some additional circuitry added at the HDL-design level. In this case, the obtained results also demonstrate the viability of this protection strategy, as it is possible to verify

in some of the derived publications [CAS06a, CAS06b, CAS07a], thus confirming the wide possibilities that are opened in this field.

7. One more step in the evolution of the watermarking strategy EPBBF-LC was based on easing the search of the signature bit blocks inside the design. With this objective, a software tool that realizes an automatic search of the signature blocks has been developed. The tool identifies the signature blocks in output patterns of the combinational logic included in the design to protect. After this search, the automatic tool generates the input patterns that will be necessary to apply in order to extract the signature bit blocks. For a given signature block, it is possible to find different input patterns that generate the same output, where the signature block will be hosted. Thus, for the set of signature blocks there exist several solutions for the signature bit spreading.

8. The first algorithm used for the development of the automated tool is based on a random decomposition of each one of the signature blocks, without considering the set of possible solutions. Nevertheless, it is very interesting to develop an algorithm to choose the best solution.

9. Thus, two algorithms that take into account a cost function to realize the signature spreading have also been developed. The selected cost function is the Hamming distance between the input patterns that will generate two adjacent signature blocks, which optimizes the resources required for the signature extraction.

10. Considering the ideas above, the second developed algorithm realizes an optimization of this cost function for the first two signature bit blocks, and for the rest of the blocks the algorithm optimizes the solution of a block with the fixed solution of the previous one. This algorithm gets a good solution but still implies high execution times for high block sizes.

11. On the other hand, the selection of the best solution can be considered to be a problem of combinacional logic, one of the most frequent applications of Simulated Annealing algorithms. Thus, the third one of the developed algorithms for the automatic tool is based on Simulated Annealing, which optimizes the cost function using considerably reduced execution times, even for high block sizes.

12. The obtained results have demonstrated that the developed protection structures present a minimal impact on the system area and performance. In addition, the evaluation of the developed protection strategies respect to other evaluation criteria as proof of authorship, cost, correct functioning, easy extraction, robustness and security has provided very satisfactory results.
13. Given the importance of the security of any watermarking system, an exhaustive study on the invulnerability of the proposed protection schemes against the different types of attacks has been realized. It has been demonstrated that the proposed watermarking strategies offer high security agains attacks.With this study two important objectives have been obtained:
    - to have preliminary procedures to measure the strength of the watermarking schemes across a set of parameters;
    - to establish some strong points and some improved ideas for other aspects in the protection schemes in order to increase its resistance against attacks.

Future research lines

The ideas developed in this work have left open wide and interesting lines of research that might represent the continuation of the work gathered in this memory:

1. The development of the automatic tool for the hosting process, which facilitates the application of the watermarking strategy EPBBF-LC, has obtained advantages so important as the automatic generation of the signature positions and the optimization of the resources required for the extraction logic. One of the algorithms used as fundamental part of the automatic tool is the Simulated Annealing. The results for this algorithm have been obtained recently, so it is possible to investigate more into the development of this algorithm, concretely in the experimental adjustment of the parameters to obtain better results, especially for smaller sizes of block. On the other hand, it would also be very interesting the development of an automatic tool more

complex that, besides using the algorithm for the signature spreading and the resource optimization, would be able to generate automatically all the necessary extraction logic described in VHDL, which later would be included into the design. This extraction logic will depend on a set of input parameters and on the signature positions obtained by the spreading algorithm that the tool itself includes.

2. It has been demonstrated that the application of the proposed watermarking strategies allows the development of reusable modules with intellectual property protection. It is possible to extend the application of these strategies to designs of great interest, such as embedded microprocessors for reconfigurable hardware. Thus, it would be interesting to carry out in parallel to the development of microprocessor itself the application of the proposed watermarking strategies, as well as the exploration and development of new alternatives for the use and secure exchange of those microprocessors as reusable modules.

3. The developed protection strategies were evaluated in terms of their invulnerability against attacks, but without differentiating between direct attacks and the so called side attacks [KOC99]. The considered attacks up to date have been direct attacks, such as tampering, which tries to change the design at the digital level in order to destroy the embedded signature, to impede its extraction or to simulate the extraction of a fake signature. On the other hand, the concept of side channel attack (SCA) is derived from the analysis of cryptographic processors, where information such as computation time, power consumption or electromagnetic emissions may be used by an attacker to extract the system key. Thus, another research line still open is the study of the invulnerability of the developed protection schemes against these side channel attacks.

Conclusion

The summary of the main contributions and future research lines concluyes the exposition of this research. The initial objective was to try to offer improved viability to reuse-based design methodologies, ensuring the claims of author rights on the used components. Thus, the main characteristic of reuse-based design has been structured, as well as those of the protection techniques for IP cores, concretely watermarking techniques, and the research carried out in this field. Finally, the results of this research has been shown and summarized in this final chapter.

Apéndice A: Familias comerciales de FPGAs

En este apéndice se da una descripción general de los dispositivos lógicos programables de las familias de FPGAs de Xilinx y Altera. Se describe de forma más detallada los dispositvos de las familias Virtex II de Xilinx y APEX20K de Altera que se han utilizado para la implementación de los diseños incluidos en esta memoria.

A.1. Introducción

Los dispositivos de Xilinx poseen un amplio rango de niveles de interconexión, locales, lo que hace que exista una sinergia entre estos dispositivos y los DSPs, debido a que la mayor parte de los algoritmos de procesamiento digital de señales procesan los datos localmente. Por otro lado, los dispositivos de Altera están basados en una arquitectura con anchos buses. Esta arquitectura es de gran importancia en el procesamiento digital de señales ya que normalmente las operaciones de *bit slices* no sólo procesan bits individuales, sino que también procesan vectores de datos más anchos, desde 16 a 32 bits, que tienen que desplazarse al siguiente bloque de procesamiento.

Los siguientes aparatados se dedican a las familias de dispositivos Virtex II de Xilinx y APEX20KC de Altera que han sido utilizadas para la implementación de los diseños propuestos en este trabajo.

A.2. Dispositivos de Xilinx

En general, los dispositivos de Xilinx constan de un conjunto de bloques lógicos, llamados bloques lógicos configurables (*Configurable Logic Blocks,* CLBs), bloques de entrada/salida (*Input-Output-Blocks*, IOBs) y elementos de interconexión. Los CLBs y los elementos de interconexión se pueden programar cargando valores en las celdas de configuración, que para este fabricante se basan en SRAM. Por tanto, estas celdas de configuración distribuidas en el chip controlan la lógica y la interconexión que realiza la función de aplicación de la FPGA. En la figura A.1 se muestra la estructura interna simplificada de una FPGA de Xilinx. Los CLBs interconectados mediante esa red están formados por un circuito combinacional programable formado por una tabla de consulta y un biestable, que puede o no tomar su entrada de la tabla anterior.

Hay múltiples familias lógicas dentro de Xilinx. Las primeras que surgieron fueron las XC2000, XC3000 y XC4000, correspondientes respectivamente a la primera, segunda y tercera generación de dispositivos, que se distinguen por el tipo de bloque lógico configurable que contienen. En la actualidad existen también las familias Spartan II, Spartan III, Virtex, Virtex II, VirtexPro, Vritex II Pro X, Virtex 4 y Virtex 5. En este trabajo se han empleado dispositivos de las familias Virtex II. A continuación se comentarán la arquitectura interna y características más importantes de la Virtex II.

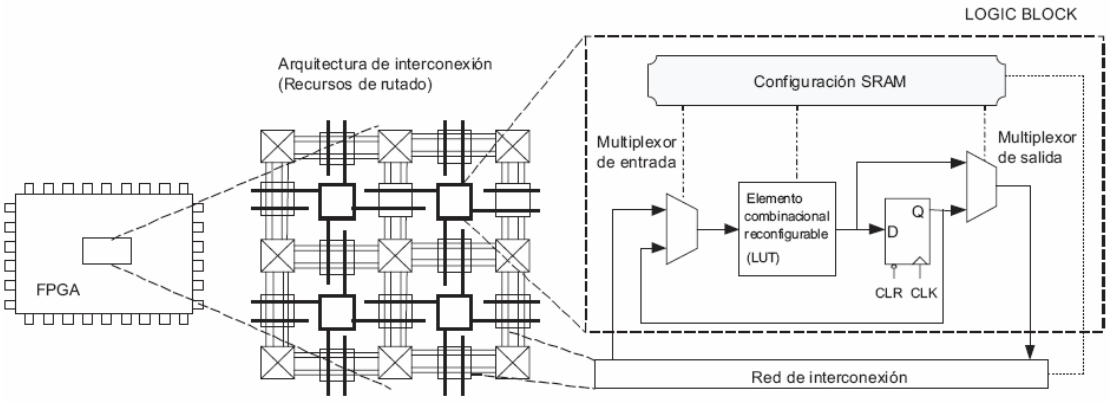

Figura A.1. Arquitectura básica de una FPGA de Xilinx.

A.2.1. Familia Virtex II

VirtexII es una plataforma pensada para conseguir diseños de altas prestaciones basados en núcleos IP y módulos hechos a medida. Entre sus características más relevantes debemos destacar: proceso de fabricación de 0.15 μm, con 8 capas de metalización y transistores de alta velocidad de 0.12 μm, bajo consumo de potencia y amplio rango de densidades de puerta. Cada dispositivo está formado por una serie

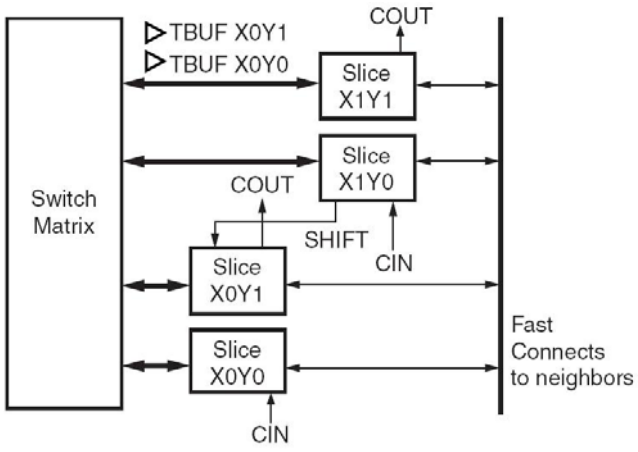

Figura A.2. CLB de la familias familias Virtex II de Xilinx.

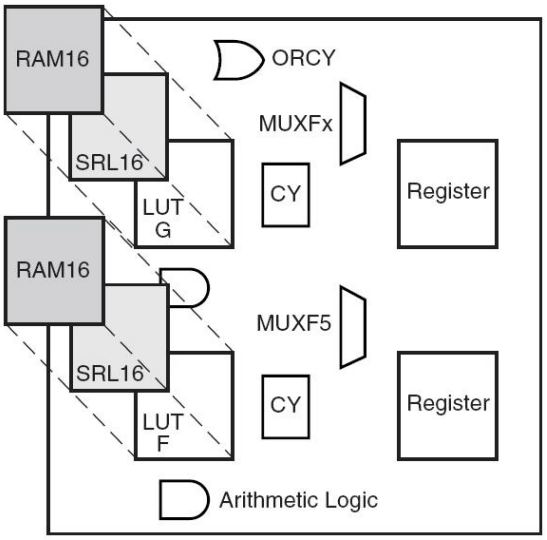

Figura A.3. Slice de la familias familias Virtex II de Xilinx.

IOBs, un conjunto de CLBs, bloques de memoria SelectRAM de 18 Kbits, elementos específicos (multiplicadores y multiplexores), controladores de reloj digitales (*Digital Clock Manager*, DCM) y recursos de interconexión. Los CLBs se distribuyen formando una matriz y con ellos se pueden construir lógica combinacional y secuencial síncrona. Los CLBs están interconectados a través de una matriz de enrutamiento general que está compuesta por un conjunto de interruptores locales ubicados en las intersecciones horizontal y vertical de los canales de enrutamiento. Cada CLB tiene una matriz de interconexiones (*switch matrix*) que sirve para acceder a la matriz general de enrutamiento como se muestra en la figura A.2. En la Virtex II un CLB está compuesto por cuatro *slices*, organizados en dos columnas con dos cadenas de acarreo lógico independientes y una cadena común de desplazamiento. Cada *slice* está formado por dos generadores de funciones de cuatro entradas, acarreo lógico, puertas lógicas artiméticas, elementos de almacenamiento y multiplexores específicos. Cada generador de funciones de cuatro entradas se puede programar como una LUT de cuatro entradas, una memoria distribuida SelectRAM de 16 bits o un registro de desplazamiento de 16 bits. La figura A.3 muestra la estructura interna de un *slice*. La presencia de registros en los *slices* permite generar circuitos secuenciales síncronos con los que se pueden almacenar resultados intermedios, generar iteraciones y permite realizar segmentación del cauce. Los DCMs tiene cuatro componentes principales: DLL (*Delay-Locked Loop*), que controlan las señales de reloj y compensan el retardo entre dichas señales, DFS (*Digital Frequency Synthesizer*), PS (*Phase Shifter*) y lógica de estado, la síntesis de frecuencia y el cambio de fase. La tabla A.1. muestra los recursos que integran los miembros de la familia de dispositivos Virtex II.

Device	System Gates	CLB (1 CLB = 4 slices = Max 128 bits)			Multiplier Blocks	SelectRAM Blocks		DCMs	Max I/O Pads(1)
		Array Row x Col.	Slices	Maximum Distributed RAM Kbits		18 Kbit Blocks	Max RAM (Kbits)		
XC2V40	40K	8 x 8	256	8	4	4	72	4	88
XC2V80	80K	16 x 8	512	16	8	8	144	4	120
XC2V250	250K	24 x 16	1,536	48	24	24	432	8	200
XC2V500	500K	32 x 24	3,072	96	32	32	576	8	264
XC2V1000	1M	40 x 32	5,120	160	40	40	720	8	432
XC2V1500	1.5M	48 x 40	7,680	240	48	48	864	8	528
XC2V2000	2M	56 x 48	10,752	336	56	56	1,008	8	624
XC2V3000	3M	64 x 56	14,336	448	96	96	1,728	12	720
XC2V4000	4M	80 x 72	23,040	720	120	120	2,160	12	912
XC2V6000	6M	96 x 88	33,792	1,056	144	144	2,592	12	1,104
XC2V8000	8M	112 x 104	46,592	1,456	168	168	3,024	12	1,108

Tabla A.1. Dispositivos de la familia Virtex II.

A.3. Dispositivos de Altera

La arquitectura básica de las FPGAs de Altera se compone de bloques de memoria y de elementos lógicos (*Logic Element*, LE) que se agrupan formando una matriz de elementos lógicos (*Logic Array Block,* LAB). Los bloques de memoria pueden utilizarse para definir pequeñas memorias o funciones lógicas especiales. Pueden además utilizarse de forma independiente o también en modo combinado para realizar funciones más complejas o elementos de memoria de mayor capacidad. Cada LE contiene una tabla de consulta de 4 entradas, un *flip-flop* programable y lógica para la generación y propagación rápida de acarreo y conexión en cascada. Las conexiones entre bloques de memoria y elementos lógicos se realizan mediante filas y columnas conectándose mediante líneas de metal denominadas *FastTracks* que se distribuyen en todo el área y transportan las señales entre los elementos de entrada-salida (*Input/Output Element*, IOE). De esta forma, se eliminan los retardos acumulativos que presentan las conexiones mediante segmentos. Los IOEs son los pines de entrada y salida de propósito general y pines dedicados a las señales de control rápidas y globales. Cada

IOE tiene un buffer que puede ser usado como entrada o salida y un *flip-flop*, que puede utilizarse para tratar con señales de entrada, salida o bidireccionales.

A.3.1. La familia APEX20KC

Los dispositivos APEX20K, con hasta un millón de puertas, se fabrican en tecnologías de 0.15 μm. De forma similar a los dispositivos de las familias APEX20K y APEX20E, los dispositivos de la famila APEX20KC representan una evolución en la arquitectura con respecto a la familia FLEX10K, incorporando nuevas características, mayor densidad y mejores prestaciones en velocidad. Ofrecen una arquitectura MultiCore, que la potencia la capacidad de los dispositivos basados en LUTs de los basados en términos producto con una estructura de memoria mejorada. La lógica basada en LUTs proporciona un funcionamiento y eficiencia optimizados para distintos tipos de aplicaciones, como el procesamiento digital de señales. La lógica basada en términos producto está optimizada para funciones combinacionales complejas, como máquinas de estado complejas. Estos dos tipos de lógicas se combinan con las funciones de memoria y una amplia variedad de MegaCores y funciones AMPP haciendo que la arquitectura APEX20KC esté disponible para diseños SoPC. Además, incluyen características adicionales como soporte estándar I/O mejorado, relojes globales adicionales y una circuitería de relol ClockLock mejorada.

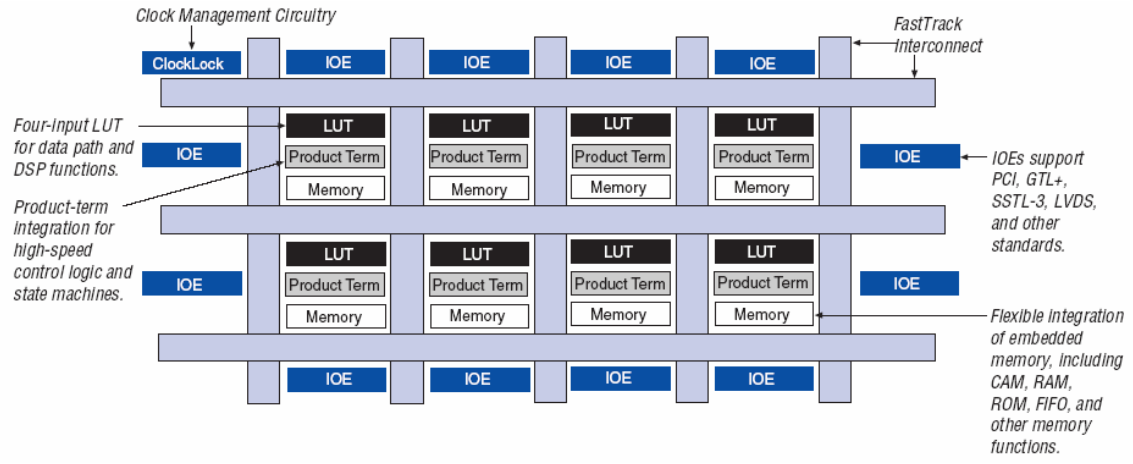

Figura A.4. Arquitectura de la familia APEX20KC

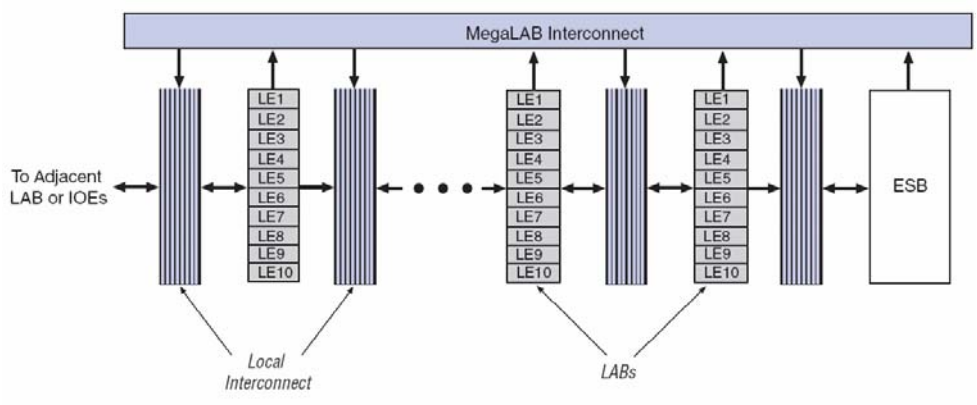

Figura A.5. Estructura de un MegaLAB,

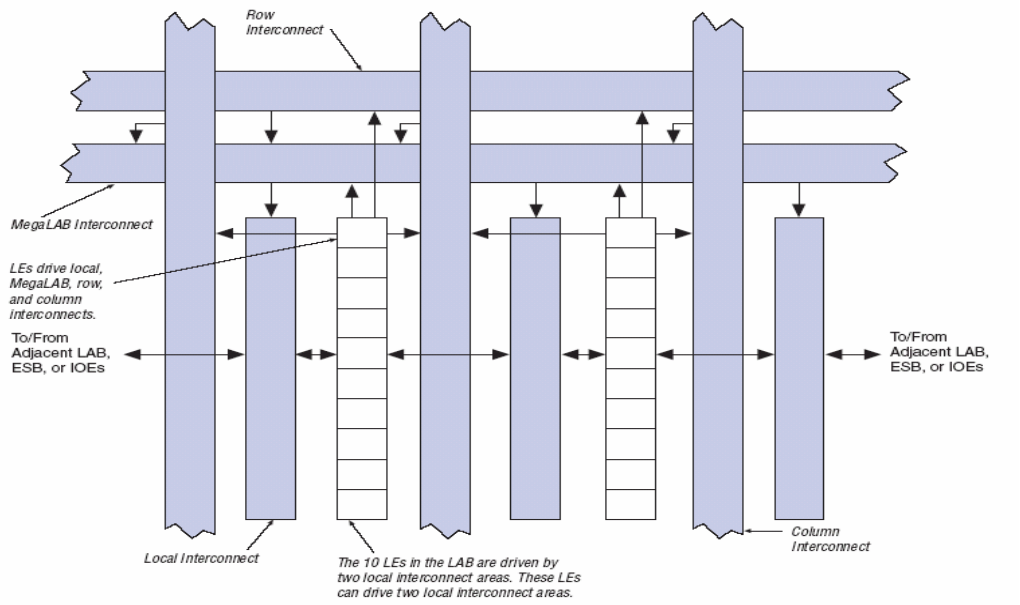

Figura A.6. Estructura de un LAB.

La figura A.4 representa la arquitectura básica de esta familia de dispositivos. La familia AEPX20KC se construye a partir de un conjunto de estructuras de MegaLABs, cada una de las cuales contiene 16 LABs, un bloque de memoria, llamado ESB (*Embedded System Blocks*) e interconexiones, que enrutan las señales dentro de la estructura del MegaLAB como muestra la figura A.5. Cada LAB agrupa 10 LEs, canales cascada y de acarreo asociados al LE, señales de control del LAB e

Feature	EP20K200C	EP20K400C	EP20K600C	EP20K1000C
Maximum system gates	526,000	1,052,000	1,537,000	1,772,000
Typical gates	200,000	400,000	600,000	1,000,000
LEs	8,320	16,640	24,320	38,400
ESBs	52	104	152	160
Maximum RAM bits	106,496	212,992	311,296	327,680
PLLs *(2)*	2	4	4	4
Speed grades *(3)*	-7, -8, -9	-7, -8, -9	-7, -8, -9	-7, -8, -9
Maximum macrocells	832	1,664	2,432	2,560
Maximum user I/O pins	376	488	588	708

Tabla A.2. Dispositivos de la familia APEX20KC.

interconexiones locales (*v.* figura A.6). La interconexión local transfiere señales entre LEs que pertenecen a los mismos LABs, IOEs o ESBs o a adyacentes.

La tabla A.2 muestra algunos de los dispositivos de la familia APEX20K y sus principales caracterísiticas.

Apéndice B: Aritmética de residuos

Este apéndice se dedica por completo a la aritmética de residuos y en él se exponen los esquemas que servirán de base para los diseños basados en esta arimética que se incluyen en esta memoria. Se presentan las principales propuestas para la implementación de componentes aritméticos RNS y los esquemas de conversión de binario a RNS y de RNS a binario más comunes.

B.1. Operaciones básicas

La principal ventaja del RNS es que la aritmética sobre el anillo de enteros modulo M puede trasladarse a los anillos correspondientes a cada módulo. Así, si $X, Y, Z \in Z_M$ y X e Y tienen las siguientes representaciones en el RNS:

$$\begin{aligned} X &\xrightarrow{RNS} (x_1, x_2, ..., x_L) \\ Y &\xrightarrow{RNS} (y_1, y_2, ..., y_L) \end{aligned} \tag{B.1}$$

entonces $Z = X \lozenge Y$ satisface:

$$\begin{aligned} Z &\xrightarrow{RNS} (z_1, z_2, ..., z_L) \\ z_i &= (x_i \lozenge y_i) \bmod m_i, \quad i = 1, 2, ..., L \end{aligned} \tag{B.2}$$

Donde ◊ representa suma, resta o multiplicación. La importancia de este resultado puede no ser apreciada a simple vista. El *i*-ésimo residuo RNS, z_i, está definido solamente en términos de $(x_i \lozenge y_i) \bmod m_i$, es decir cada residuo del resultado es una función de sólo un dígito de cada operando, y por tanto independiente del resto de los otros residuos. De esta forma, en el RNS las operaciones suma, resta o multiplicación sobre un cierto rango dinámico *M* quedan reducidas al conjunto de operaciones paralelas sobre una serie de canales de menor rango dinámico y sin propagación de acarreo entre ellos, eliminando los problemas relacionados con las cadenas de acarreo largas cuando se tienen anchos de palabra grandes. Además, el RNS permite realizar estas operaciones de números grandes a la misma velocidad que con números pequeños, ya que la velocidad está determinada por el canal más lento.

Los sistemas numéricos convencionales son lineales, posicionales y pesados, es decir, cada cifra tiene un peso perfectamente determinado. Eso permite comparar dos números (dígito a dígito) y hace fácil la detección de signo (comprobando el bit más significativo). En aritmética de residuos esto no es posible, ya que al ser un sistema no pesado se dificulta la comparación de magnitudes y la comparación de signo. Dado que la división es una sucesión de restas y comparaciones y que el RNS es un sistema de representación de enteros, ésta operación es difícil de implementar en aritmética de residuos. Aunque en la literatura existen trabajos para la división en el RNS y su optimización [BAN81, POS96, HIA97], la complejidad de esta operación es mucho mayor que la de la suma o multiplicación.

Como se ha mencionado anteriormente, un aspecto a tener en cuenta en cualquier sistema de representación numérico es la posibilidad de que se supere el rango dinámico máximo, es decir, el desbordamiento u *overflow*. En los sistemas convencionales la detección de desbordamiento se realiza con ayuda del dígito más significativo; que como hemos visto, es una operación compleja; además, este desbordamiento no existe como tal en la aritmética en módulo, al carecer de sentido por la propia naturaleza de la misma, con lo que su aparición queda completamente enmascarada en el funcionamiento normal del sistema. Para evitar el desbordamiento durante el funcionamiento de un sistema RNS se puede aumentar el rango dinámico del sistema añadiendo módulos al sistema RNS o se puede utilizar el escalado, que es una forma

especial de división. En principio puede parecer que la realización del escalado va a presentar un problema importante por tratarse de una forma de división (operación difícil en el RNS), pero el hecho de que el divisor sea constante simplifica el problema, y lo hace abordable en determinadas aplicaciones [JUL78].

B.2. Módulos aritméticos

Para explotar el paralelismo del RNS, se pueden utilizar unidades aritméticas que implementen de forma rápida y eficiente el cálculo del dígito $(x_i \Diamond y_i) \bmod m_i$ correspondiente a cada canal (a cada módulo m_i). Para un conjunto de módulos arbitrarios no existen unidades aritméticas binarias convencionales que se puedan utilizar para la aritmética de residuos. Sólo para el caso en el que uno de los módulos sea potencia de 2, $m_i = 2^n$, se puede utilizar una unidad aritmética convencional. En este apartado se presentan diferentes propuestas de sumadores y multiplicadores módulo *m* mostrando su estrategia de diseño, arquitectura básica y ventajas e inconvenientes.

B.2.1. Sumadores

La estrategia de implementación de la aritmética de residuos ha evolucionado basada en los desarrollos de tecnológicas de los circuitos integrados. La tecnología disponible determina el método de implementación tanto en el circuito como en los niveles del sistema. En los últimos 30 años han surgido una gran variedad de implementaciones para la suma basadas en aritmética de residuos que se pueden clasificar dentro de los siguientes métodos de implementación:

- implementación mediante tablas de consulta;
- implementación lógica directa usando módulos binarios convencionales;
- implementación híbrida.

A continuación se describe los sumadores basados en módulos binarios convencionales, que han servido como módulos para los diseños basados en el RNS utilizados en esta memoria.

La suma en módulo *m* de dos enteros *x* e *y* se puede descomponer en dos casos:

$$\left|x+y\right|_m = \begin{cases} x+y & si\ x+y<m \\ x+y-m & si\ x+y\geq m \end{cases} \qquad \text{(B.3)}$$

en el primero el resultado no excede el valor de $m-1$ por lo que no se debe realizar ninguna corrección, en el segundo, para obtener el valor correcto, se le debe restar el módulo con el cual se está trabajando. Para la sumaen RNS se pueden emplear sumadores binarios [BAY87, DUG92], pudiéndose presentar los siguientes casos:

Primer caso: $m = 2^n$

En el caso $m = 2^n$, con *n* entero positivo, el residuo y la representación binaria son iguales, y un sumador binario de *n* bits proporciona la suma módulo *m* sin más que despreciar el acarreo, figura B.1.a).

Segundo caso: $m = 2^n - 1$

En este caso se suman los residuos y, de haber acarreo, se suma al resultado anterior, figura B.1.b), como en la suma en complemento a 1.

Tercer caso: $m < 2^n - 1$

La suma directa de dos residuos presentará $2n - m$ combinaciones incorrectas, a las que se le deberá restar *m*, o, sumarle su complemento al módulo $(2n - m)$. Este caso se puede resolver mediante dos sumadores binarios, el primero, calcula $x + y$ mientras que el segundo calcula $x + y - m$. El problema reside en determinar cual de los dos da la suma correcta y presentar ésta como resultado. En [BAY87, DUG92] se muestran todos los casos que posibles, y se concluye que el resultado correcto es el que se obtiene del segundo sumador siempre que se tenga un acarreo en alguno de los dos sumadores, mientras que, será el del primer sumador, cuando no exista acarreo en ninguno de los dos sumadores. El problema se puede resolver entonces con dos sumadores binarios, un multiplexor y una puerta OR de dos entradas como se puede

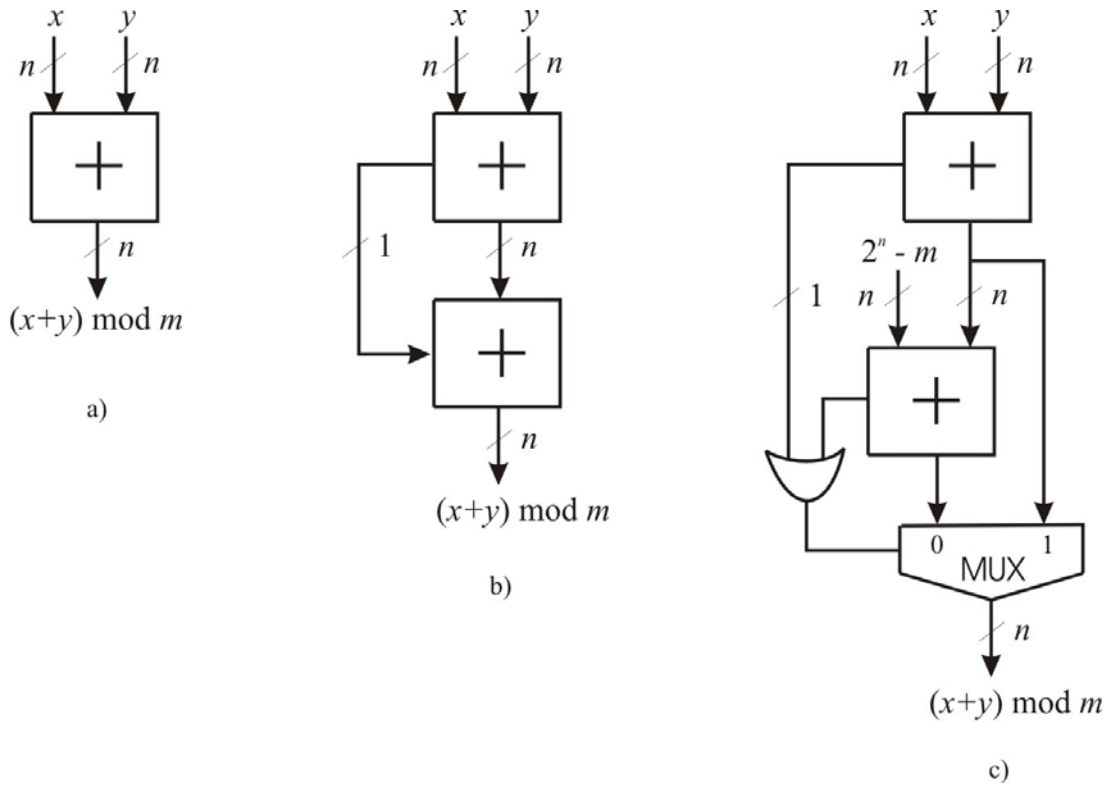

Figura B.1. Esquema de un sumador en módulo *m* mediante sumadores binarios: a) primer caso, b) segundo caso, c) tercer caso.

ver en la figura B.1.c). Esta estructura, además, es fácilmente segmentable, lo que facilitará la implementación de sistema de alta velocidad.

B.2.2. Multiplicadores

Existe una gran variedad de propuestas para la realización práctica de la multiplicación módulo *m*; entre ellas se incluyen las siguientes:

- multiplicadores basados en tablas de consulta;
- multiplicadores de ley cuadrática;
- multiplicadores por transformación al dominio de los índices;
- multiplicadores que no hacen uso de tablas de consulta.

A continuación se describe los multiplicadores basados en aritmética de índices, que han sido utilizados en el filtro FIR-RNS utilizado como ejemplo para la aplicación de las estregias de protección. El diseño de este tipo de multiplicador [JUL80, RAD91] se basa en la realización de una transformación a un dominio en el que la multiplicación se puede realizar de forma más eficiente mediante una operación de suma, pero su uso está limitado a módulos primos. Este multiplicador es comparable en complejidad y en velocidad al multiplicador basado en la ley cuadrática.

El cálculo por índices se basa en las propiedades asociadas con los cuerpos de Galois [KRI94, KRI98]. Para multiplicar sobre el cuerpo de Galois o GF(*m*) se puede utilizar la propiedad de que todos los elementos no nulos se pueden generar por medio de un elemento generador o raíz, *g*:

$$\forall\, q_n \in Q \qquad \exists\, i_n \in I \quad : \quad q_n = g^{i_n} \tag{B.4}$$

donde $Q = \{1, 2, ..., m-1\}$, $I = \{0, 1, ..., m-2\}$ e i_n recibe el nombre de *índice* de q_n. De esta manera, a través del isomorfismo entre el grupo multiplicativo *Q*, con la multiplicación módulo *m*, y el grupo aditivo *I*, con la suma módulo *m*–1:

$$\left| q_j q_k \right|_m = g^{\left| i_j + i_k \right|_{m-1}} \tag{B.5}$$

Así, la multiplicación de dos números q_j y q_k consistiría en calcular sus índices, i_j e i_k, respectivamente, sumarlos módulo *m*–1, y realizar la transformación inversa desde el dominio de los índices. La figura B.2 muestra el diseño de un multiplicador basado en aritmética de índices. Como no está definido un índice para el elemento nulo de *Q*, hay que añadir la lógica necesaria para tratar por separado el caso en el que alguno de los operandos de entrada sea cero y hacer la salida cero.

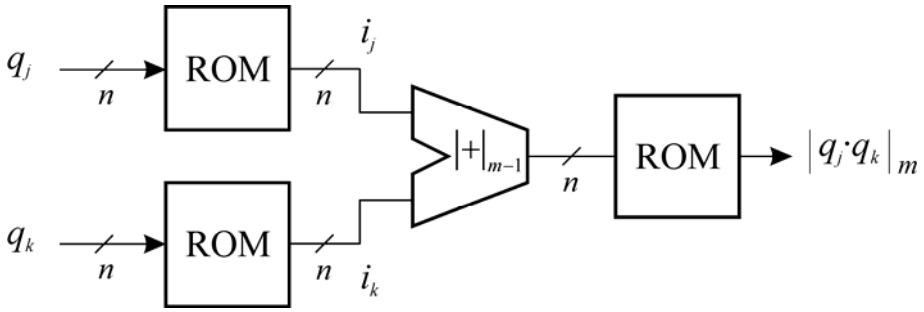

Figura B.2. Multiplicador basado en aritmética de índices.

B.3. Esquemas de conversión RNS

En este apartado se van a describir los esquemas de conversión de binario a RNS y viceversa.

B.3.1. Conversión de binario a RNS

Usando la ecuación (2.3) es posible calcular el residuo de un número. En [SZA67] se propone un método muy simple para generar los residuos a partir de la representación en complemento a dos. Suponiendo que X está representado por un número de t+1 bits:

$$X = \sum_{j=0}^{t} B_j 2^j \tag{B.6}$$

Teniendo en cuenta que B_t es el bit de signo, la expresión anterior equivale a:

$$X = -B_t 2^t + \sum_{j=0}^{t-1} B_j 2^j \tag{B.7}$$

A partir de esta expresión el residuo $x_i = X \bmod m_i$ se calcula como:

$$x_i = \left(B_t \left(m_i - 2^t \bmod m_i \right) + \sum_{j=0}^{t-1} B_j \left(2^j \bmod m_i \right) \right) \bmod m_i \tag{B.8}$$

Para generar el residuo anterior, se pueden definir las funciones:

$$F_i(j) = \begin{cases} 2^j \bmod m_i & j = 0,1,...,t-1 \\ m_i - 2^t \bmod m_i & j = t \end{cases} \tag{B.9}$$

y de esta forma la ecuación (B.9) quedaría:

$$x_i = \left(B_t \, F_t + \sum_{j=0}^{t-1} B_j \, F_i(j) \right) \bmod m_i \tag{B.10}$$

que equivale a:

$$x_i = \left(\sum_{j=0}^{t} B_j \, F_i(j) \right) \bmod m_i \tag{B.11}$$

Para el cálculo de un residuo se pueden almacenar las funciones F_i en tablas de 2^{t+1} palabras. De esta forma el cálculo consistiría en acceder a las posiciones de memoria para las que $b_j = 1$ de la tabla correspondiente y sumar los contenidos en sumadores módulo m_i.

A continuación se va a proponer otro esquema que optimiza los procesos de conversión, tradicional cuello de botella, en los sistemas basados en el RNS y que ha sido utilizado en los diseños basados en el RNS que aparecen en esta memoria. La conversión de una entrada de *B* bits *x*(*n*) al dominio del RNS se realiza a través de su descomposición en *p* bloques de *b* bits $\tilde{x}_0, \tilde{x}_1, ..., \tilde{x}_{p-1}$. Estos bloques de bits son procesados mediante *p*–1 tablas que almacenan la siguiente función:

$$\Phi_k(\tilde{x}_k) = \begin{cases} (2^{n_l - 1 + (k-1)b} \, \tilde{x}_k) \bmod m_l & k = 1,2,..., p-2 \\ \left(2^{B-q} \, \tilde{x}_p \right) \bmod m_l & k = p-1 \end{cases} \tag{B.12}$$

siendo $n_l = \lceil \log_2(m_l) \rceil$ y q el tamaño en bits del bloque más significativo $\tilde{x}_{p-1}$. En la figura B.3 se puede ver el diagrama de bloques de este esquema de conversión para uno de los módulos, el esquema de conversión completo de binario a RNS por descomposición en bloques estaría formado por una estructura similar a la de la figura B.3 para cada módulo que compone el RNS. La salida de las tablas $\tilde{y}_1, \tilde{y}_2 ..., \tilde{y}_{p-1}$ y el bloque $\tilde{x}_0$ se suman usando un árbol de sumadores modulo m_i, que proporciona el residuo correspondiente.

B.3.2. Conversión de RNS a binario

Para la conversión desde el RNS a binario, una *N*-tupla de residuos puede convertirse a un valor entero almacenando directamente en una ROM la correspondencia entre la representación en residuos y los valores binarios correspondientes. Esta conversión sería apropiada para sistemas de alta velocidad. A continuación se van a describir otros métodos de conversión que, en general, son más eficientes.

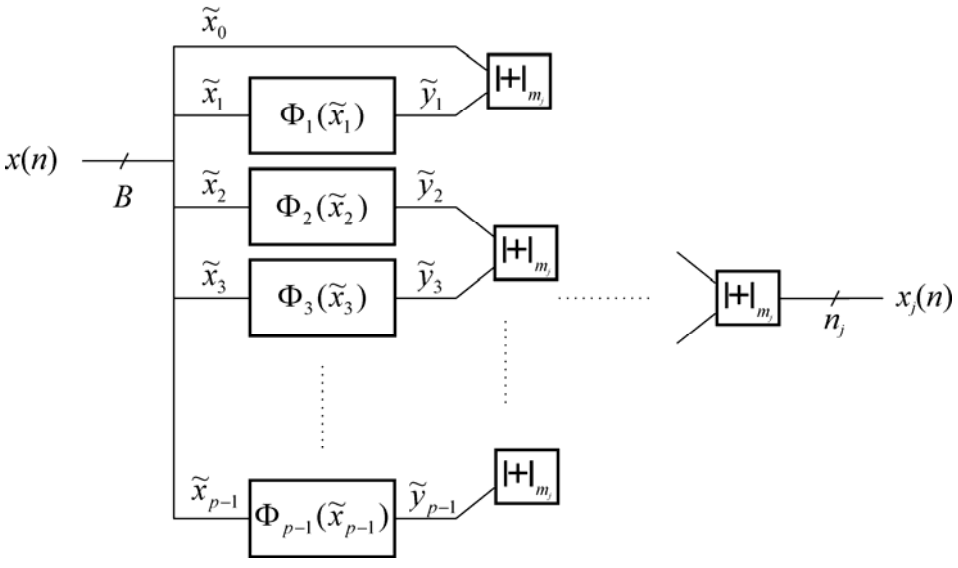

Figura B.3. Conversión de binario a RNS por descomposición en bloques para un canal.

B.3.2.1. CRT

El CRT (*Chinese Remainder Theorem*) se basa en la aplicación directa del teorema del resto chino. El Teorema del Resto Chino es la base para la conversión a un valor decimal de un número representado en el sistema numérico de residuos.

Teorema: *Considérese el conjunto de módulos primos relativos entre sí* $\{m_1, m_2, ..., m_L\}$ y el *conjunto de residuos de Y,* $[y_1, y_2, ..., y_L]$, *definidos mediante:*

$$y_i = Y \bmod m_i \qquad i = 1, 2, ..., L \tag{B.13}$$

*Entonce*s:

$$Z(M) \cong Z(m_1) \oplus Z(m_2) \oplus ... \oplus Z(m_L) \tag{B.14}$$

o, equivalentemente, el anillo de enteros módulo M, $Z(M)$*, es isomorfo a la suma directa de los anillos de enteros de módulos* $m_1, m_2, ..., m_L$ *y existe un único* $Y \in Z(M)$, $M = \prod_{i=1}^{L} m_i$ *que, de acuerdo con la representación de la ecuación (B.1), se obtiene por medio de la ecuació*n:

$$Y \bmod M = \left(\sum_{i=1}^{L} \widehat{m}_i \left(\frac{y_i}{\widehat{m}_i} \right) \bmod m_i \right) \bmod M \tag{B.15}$$

donde:

$$\widehat{m}_i = \frac{M}{m_i} \tag{B.16}$$

y $\left(\frac{x_i}{\widehat{m}_i} \right)$ *representa el producto* (*módulo* m_i) *de* x_i *por el inverso multiplicativo de* $\widehat{m}_i$ *modulo* m_i*, definido de manera que:*

$$\widehat{m}_i = \frac{M}{m_i} \tag{B.17}$$

Así, el teorema del resto chino representa un método fundamental para la división del anillo *Z*(*M*) en un número de anillos independientes $Z(m_i)$ más pequeños sobre los cuales se realizan las operaciones en paralelo. Además, el proceso de conversión basado en el CRT se encuentra definido por medio de la ecuación (B.15), que equivale a la siguiente expresión:

$$Y = \left(\sum_{i=1}^{N} s_i \left(y_i s_i^{-1} \right) \bmod m_i \right) \bmod M \tag{B.18}$$

donde $s_i = \hat{m}_i$ y $s_i^{-1} = \dfrac{M}{m_i}$ es el inverso multiplicativo de s_i de modo que:

$$\left(s_i^{-1} s_i \right) \bmod m_i = 1 \tag{B.19}$$

En la figura B.4 se puede ver el diagrama de bloques correspondiente a este esquema de conversión.

Aunque parece un método elegante para la reconstrucción de números enteros tras ser procesados en el dominio del RNS, su implementación eficiente es crucial. Una de las principales limitaciones de los convertidores basados en el CRT es el tamaño de los sumadores módulo *M*. Por ejemplo, si el rango dinámico del sistema es superior a 32 bits, la complejidad de los sumadores afectará en gran medida a la habilidad del RNS de

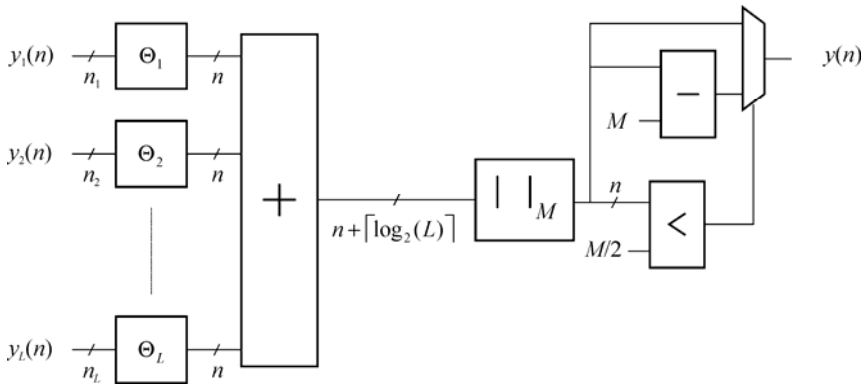

Figura B.4. Conversión de RNS a binario mediante el algoritmo CRT.

incrementar la velocidad del sistema en un orden de magnitud. Este problema se puede resolver utilizando algoritmos de conversión alternativos como el algoritmo de conversión y escalado propuesto en la siguiente sección.

B.3.2.2. ε-CRT

Este método de conversión [GRI89a], [GRI89b] realiza la etapa de conversión de RNS a binario de forma eficiente. Un convertidor basado en el ε-CRT posee una latencia mínima y puede realizar la conversión a binario y el escalado en un solo paso. Para reducir la complejidad del convertidor CRT este algoritmo sustituye los sumadores módulo *M* por sumadores convencionales como se verá a continuación. El algoritmo ε-CRT se define a partir de la ecuación (B.18) tomando una versión escalada de esta ecuación:

$$\overline{Y} \equiv \left\lfloor \frac{Y}{V} \right\rfloor = \left(\sum_{i=1}^{N} \left\lfloor \frac{s_i}{V} \left(y_i \, s_i^{-1} \right) \bmod m_i \right\rfloor \right) \bmod M \tag{B.20}$$

que equivale a:

$$\overline{Y} = \left(\sum_{i=1}^{N} \lfloor \phi(x_i) / V \rfloor \right) \bmod \lfloor M / V \rfloor \tag{B.21}$$

donde:

$$\phi(x_i) = s_i \left(y_i s_i^{-1} \right) \bmod m_i \tag{B.22}$$

Para realizar la conversión se pueden calcular y almacenar en tablas los valores de $\lfloor \phi(x_i) / V \rfloor$. La idea de este algoritmo es utilizar sumadores convencionales, para lo que sólo hay que seleccionar *V*, que puede ser cualquier número real, de manera que $\lfloor M / V \rfloor$ sea una potencia de dos. Por lo tanto para realizar la conversión sólo se necesitarían *N* tablas y un acumulador convencional módulo $\lfloor M / V \rfloor$, tal y como se puede ver en la figura B.5.

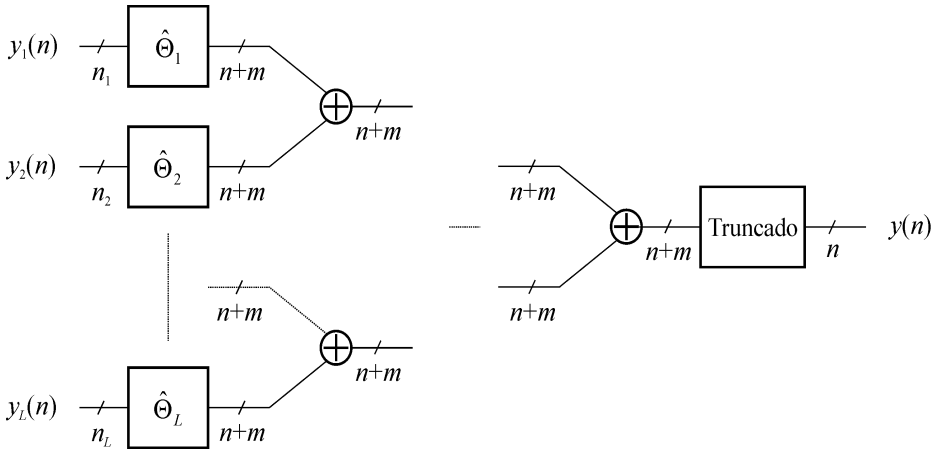

Figura B.5. Conversión de RNS a binario mediante el algoritmo ε-CRT.

Este algoritmo introduce un pequeño error en el escalado de la salida, ya que en la ecuación (B.21) se utiliza la parte entera. Este error se comete en el límite del rango dinámico, *M*, y se consigue evitar seleccionando un conjunto de módulos que proporcione un rango dinámico superior al necesario y, por tanto, se evite llegar al extremo en el que el error es inaceptable.

B.4. Conclusión

En este apéndice se han expuesto diversas propuestas sobre el diseño de sumadores y multiplicadores para el RNS, orientadas a distintas aplicaciones, a diferentes tamaños de módulos. Se han mostrado diseños de carácter general para módulos grandes, propuestas de carácter muy específico y que tratan de sacar el máximo partido de las propiedades del RNS, y se ha prestado especial atención a aquellos esquemas que han servido de base a la contribución de esta memoria. Por último se han estudiado los esquemas de conversión, y se han explicado con más detalle aquellos que se han empleado en el diseño propuesto en esta tesis.

Apéndice C: Flujo de diseño para ASICs *semi-custom*

En este apéndice se describe brevemente el flujo de diseño para la síntesis sobre tecnologías VLSI, para lo que se ha usado la herramienta Synopsys Design Compiler, y se proporciona la información básica de la biblioteca de celdas estándar lsi_10k, empleada como tecnología objetivo en los diferentes ejemplos de ASICs *semi-custom*.

C.1. Design Compiler

Design Compiler forma parte de los productos de *software* de síntesis de Synopsys y proporciona una optimización de las restricciones y soporte para un amplio conjunto de estilos de diseño. Las herramientas de Design Compiler sintetizan la descripción HDL en un diseño a nivel de puertas dependiente de la tecnología. Design Compiler optimiza los diseños combinacionales o secuenciales teniendo en cuenta parámetros como la velocidad, el área y el consumo de potencia, y permite tanto diseños con estructura jerárquica como en un único nivel. Design Compiler proporciona dos interfaces diferentes de trabajo, Design Analyzer y Design Compiler *shell* que se describen brevemente a continuación:

- Design Analyzer: es el interfaz de usuario gráfico (*Graphical User Interfaz*, GUI) para los productos de síntesis. Este interfaz inluye en su menú la mayor parte de los comandos de síntesis. Con estos comandos se puede cargar el archivo del diseño, sintetizarlo con GTECH (la biblioteca genérica), compilarlo con los

elementos ASIC de la librería elegida (lsi_10k) para la generación de los informes de área, tiempo y potencia disipada.

- Design Compiler *shell* (dc_shell): es el interfaz de línea de comando para la síntesis. Design Analyzer es útil para hacer un primera depuración del diseño. Sin embargo, una interfaz de *scripts* simplifica esta tarea porque las restricciones necesarias pueden ser escritas una vez y ejecutarse, de forma sencilla, tantas veces como se necesite mediante el correspondiente *script*. Un *script* de síntesis típico estaría compuesto por los siguientes pasos:

  1. elaborar y analizar todos los componentes y definir el diseño que va a ocupar el nivel más alto de la jerarquía,
  2. ajustar las restricciones de área y tiempo;
  3. unificar el diseño;
  4. compilar y guardar el diseño;
  5. generar informes de tiempo, área, potencia y elementos de la biblioteca.

La figura C.1 muestra el flujo de diseño completo de descripciones de alto nivel, donde las áreas sombreadas muestran las tareas de síntesis. A continuación se describe cómo se integraría Design Compiler en este flujo. Se comienza con la descripción HDL del diseño. En paralelo, se realiza una síntesis preliminar y una simulación funcional. Para la síntesis preliminar se utiliza un algoritmo que evalua las prestaciones del diseño con respecto a los objetivos marcados. Si se violan las especificaciones de tiempo en más de un 10%, se deben modificar o retocar los objetivos y las restricciones. La simulación funcional verifica que el diseño realiza la función deseada. En caso de no conseguir superar esta verificación, se debe modificar el código HDL. Seguidamente, se realiza la implementación del diseño o síntesis final en la que se utiliza Design Compiler para generar un diseño que se ajuste a los objetivos marcados. Después de sintetizar el diseño en un *netlist* a nivel de puertas, se debe verificar que el diseño cumple los objetivos. En caso de no ser así, se generan y analizan los informes para determinar las técnicas que se deben usar para corregir estos problemas. Después de verificar la funcionalidad, los requisitos de tiempo y los objetivos globales, se completa el diseño físico. Si el análisis de las prestaciones del diseño y a implementado muestra

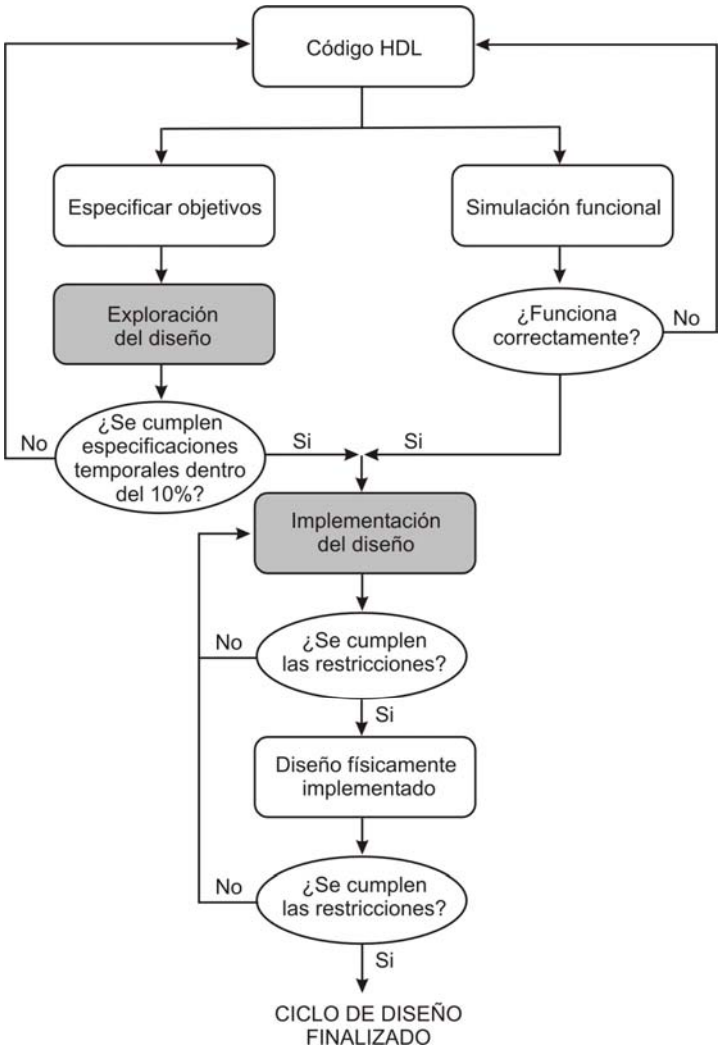

Figura C.1. Fllujo de diseño de alto nivel orientado a implementación en ASICs con herramientas de Synopsys.

resultados que se encuentran dentro de los objetivos del diseño, el ciclo de diseño habría finalizado; si no, se debe volver a la fase de implementación del diseño.

Por otro lado, la figura C.2 muestra un ejemplo del flujo de diseño de síntesis mediante Design Compiler.

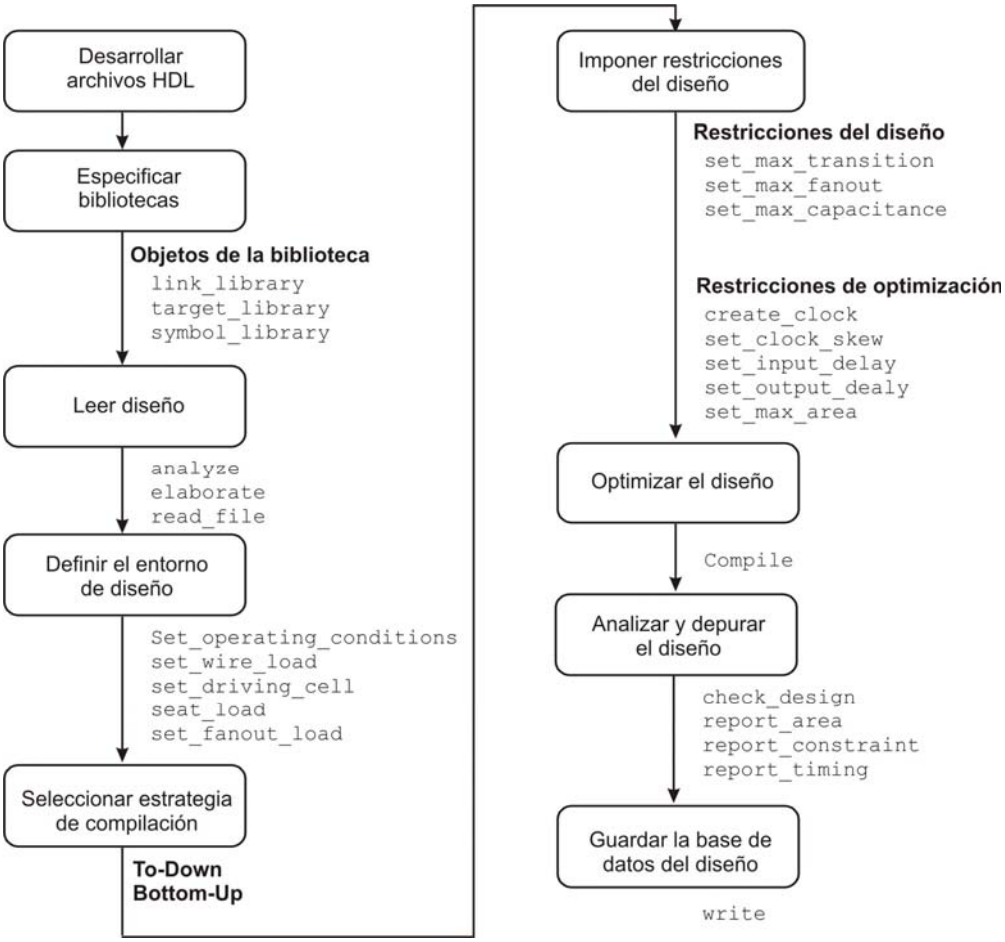

Figura C.2. Fllujo de síntesis con Design Compiler.

C.2. DesignWare y biblioteca de celdas estándar lsi_10k

En este trabajo se ha utilizado la biblioteca de celdas estándar lsi_10k bajo el peor caso típico de condiciones de funcionamiento comerciales (WCCOM). El componente DesignWare de Synopsys incluye un amplio rango de módulos IP, desde microcontroladores, estructuras de memoria, interfaces de comunicación, etc., y permite una selección automática de estructuras sumadoras RCA/CLA. Estos módulos IP puede

trasladarse a diferentes tecnologías objetivo, según estén disponibles en el flujo de diseño. Para la investigación realizada se seleccionó la biblioteca clásica lsi_10k, ya que es parte del conjunto de herramientas de diseño incluidas en el paquete de Synopsys disponible y permite la verificación de los resultados presentados. En este sentido, para escalar el proceso de 1μm a la tecnología de proceso actual (por ejemplo, 0.15μm) es necesario escalar la velocidad por el factor de los dos tamaños del proceso. Las medidas de área proporcionadas por las herramientas Synopsys, expresadas en puertas equivalentes, no están escaladas por la tecnología de proceso. En cualquier caso, no está permitido especificar la estructura interna de estas bibliotecas ni proporcionar más detalles debido a las restricciones impuestas por los fabricantes para su uso, siempre bajo licencia.

Bibliografía

[ABD03] A. T. Abdel-Hamid, S. Taha, El Mostapha Aboulhamid, "IP watermarking techniques: Survey and Comparison", *Proc. on the 3rd IEEE International Workshop on System-on-Chip for Real-Time Applications*, págs. 60-65, 2003.

[ABD05] A. T. Abdel-Hamid, S. Taha, El Mostapha Aboulhamid, "A Survey on IP Watermarking Techniques", *Design Automation for Embedded Systems,* vol. 9, págs. 211-227, 2004.

[ABD06] A. T. Abdel-Hamid, S. Taha, El Mostapha Aboulhamid, "Finite State Machine IP Watermarking: A Tutorial", *Proc. of the first NASA/ESA conference on Adaptive Hardware and Systems table of contents*, págs. 457-464, 2006.

[ALT01] Altera Corporation, *FLEX10K Embedded Programmable Logic Device Family*, 2001.

[ALT08] Altera Corporation, *Altera Homepage 2008.* `http://www.altera.com.`

[ANA99] Analog Devices, *ADV601LC Ultralow Cost Video Codec*, 1999.

[AND98] R. Andraka, "A survey of CORDIC algorithms for FPGAs", *Proc. of the 1998 ACM/SIGDA Sixth International Symposium on Field Programmable Gate Arrays*, págs.191-200, 1998.

[AAR89] E. H. L. Aarts, J. H. M. Korst, "Blotzamann machines for solving travelling salesman problems", *European Journal of Operatinal Research*, vol. 39, nº 2, págs. 144-147, 1989.

[BAN81] D. K Banerji, "A High-Speed Division Method in Residue Arithmetic", *Proc. of the 1981 IEEE Fifth Symposium on Computer Arithmetic*, págs. 158-164, 1981.

[BAN90] P. Banerjee, M. H. Jones, J. S. Sargent, "Parallel simulated annealing algorithms for standard cell placement on hypercube multiporcessors", *IEEE Transactions Parallel and Distributed Systems*, vol. 1, págs. 91-06, 1990.

[BAR04] M. Barni, F. Bartolini, *Watermarking Systems Engineering.* Marcel Dekker, 2004.

[BAY87] M. A Bayoumi, G. A. Jullien, W. C. Miller, "A VLSI Implementation of Residue Adders", *IEEE Transactions on Circuits and Systems*, vol. 34, nº 3, 1987.

[BEE85] G. F. M. Beenker, T. A. C. M. Claasen, P. W. C. Hermens, "Binary sequences with maximally flat amplitude spectrum", *Philips Journal of Research*, vol. 40, págs. 289-304, 1987.

[BHA87] J. Bhasker, S. Sahni, "Optimal linear arrangement of circuit component", *Journal of VLSI and Computer Systems*, vol. 2, págs. 87-109.

[BLA84] R. E. Blahut, *Theory and Practice of Error Control Codes*. Addison-Wesley, 1984.

[BON95] D. Boneh, J. Shaw, "Collusion-secure fingerprinting for digital data", *Proc. 15th Annual International Cryptology Conference*, págs. 452–465, 1995.

[BRI94] K. 0'Brien, S. Maginot, "Non-Reversible VHDL Source-Source Encryption", *Proc. of the Conference on European Design Automation*, págs. 480 – 485, 1994.

[CAL99] A. E. Caldwell, H. Choi, A. B. Kahng, S. Mantik, M. Potkonjack, G. Qu, J. L. Wong, "Effective Iterative Techniques for Fingerprinting Design IP", *Proc. of the 36th ACM/IEEE conference on Design automation conference*. págs. 843-848, 1999.

[CAS05] E. Castillo, U. Meyer-Baese, L. Parrilla, A. García and A. Lloris, "RNS-based Watermarking Technique for IP Core Protection", *XX Conferece on Design of Circuits and Integrated Systems (DCIS 2005)*, 2005.

[CAS06] E. Castillo, U. Meyer-Baese, L. Parrilla, A. García and A. Lloris, "IPP Watermarking Technique for IP Core Protection on FPL Devices", *Proc. 16th International Conference on Field Programmable Logic and Applications FPL 2006*, págs. 487-492, 2006.

[CAS07a] E. Castillo, U. Meyer-Baese, A. García, L. Parrilla, A. Lloris, "IPP@HDL: Efficient Intellectual Property Protection Scheme for IP Cores", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 15, págs. 578-591, 2007.

[CAS07b] E. Castillo, U. Meyer-Baese, A. García, L. Parrilla, and A. Lloris, "Intellectual Property Protection of IP Cores through High-level Watermarking", *Proc. of SPIE, Independent Component Analyses, Wavelets, Unsupervised Nano-Biomimetic Sensors, and Neural Networks V*, vol. 6576, 2007.

[CAS07c] E. Castillo, U. Meyer-Baese, A. García, L. Parrilla, and A. Lloris", Intellectual Property Protection of HDL IP Cores Through Automated Signature Hosting", *17th International Conference on Field Programmable Logic and Applications,* págs. 183-188, 2007.

[CHA95a] C. Chakrabarti, M. Vishwanath, "Efficient Realizations of the Discrete and Continuous Wavelet Transform: from Single Chip Implementations to Mappings on SIMD Array Computers", *IEEE Transactions on Signal Processing*, vol. 43, págs. 759-771, 1995.

[CHA95b] Y. T. Chan, *Wavelet Basis*. Kluwer Academic Publishers, 1995.

[CHA96] C. Chakrabarti, M. Vishwanath, R. M. Owens, "Architectures for Wavelet Transforms: A Survey", *Journal of VLSI Signal Processing*, vol. 14, nº 2, págs. 171-192, 1996.

[CHA97] K. C. Chang, *Digital design and modeling with VHDL and synthesis*. IEEE Computer Society Press, California, 1997.

[CHA98] E. Charbon, "Hierarchical watermarking in IC design", *Proc. of the IEEE Custom Integrated Circuits Conference*, págs. 295–298, 1998.

[CHA00] E. Charbon, I. H. Torunoglu, "On Intellectual Property Protection", *Proc. 2000 IEEE Custom Integrated Circuits Conference*, págs. 517-523, 2000.

[CHA03] E. Charbon, I. Torunoglu, "Watermarking techniques for electronic circuit design", *Lecture Notes on Computer Science*, 2613, págs. 147-169, 2003.

[COO65] J. W. Cooley, J. W. Tukey, *"An Algorithm for the Machine Calculation of Complex Fourier Series", Math of Computer*, vol. 19, págs. 297-301, 1965.

[COX97] I. J. Cox, M. L. Miller. "A Review of Watermarking and the Importance of Perceptual Modeling", *Proc. of SPIE, Human Vision & Electronic Imaging II*, vol. 3016, págs. 92-99, 1997.

[COX07] I. Cox, M. Miller, J. Bloom, J. Fridrich, *Digital Watermarking and Steganography*, Second Edition, The Morgan Kaufmann Series in Multimedia Information and Systems, 2007.

[CUI06] A. Cui, C. H. Chang, "Stego-signature at Logic Synthesis Level for Digital Design IP Protection", *Proc. IEEE Int. Symp. on Circuits and Systems*, Greece, 2006.

[DIC00] C. Dick, *FPGAs: The High-End Alternative for DSP Applications*. DSP Engineering, 2000.

[DOW03] K. A. Dowsland, B. Adenso, "Heuristic design and fundamentals of the Simulated Annealing", *Revista Iberoamericana de Inteligencia Artificial,* vol.19, págs. 93-102, 2003.

[DUG92] M. Dugdale, "VLSI Implementation of Residue Adders Based on Binary adders", *IEEE Trans. on Circuits and Systems II*, vol. 39, nº5, págs. 325-329, 1992.

[D&R08] *Designs&Reuse Homepage* 2008, Catalyst of Collaborative IP based SoC design, `http://www.design-reuse.com/`.

[EGG02] J. Eggers, B. Girod, *Informed Watermarking.* Kluwer 2002.

[ELL95] K. M. Elleithy, M.A. Bayoumi, "A Systolic Architecture for Modulo Multiplication*", IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 42, nº 11, págs. 725-729, 1995.

[ERC03] M. D. Ercegovac, Tomas Lang, *Digital Arithmetic*. Morgan Kaufmann, 2003.

[ERG99] F. Ergun, J. Kilian, R. Kumar. "A Note on the Limits of Collusion-resistant Watermarks", *Advances in Cryptology: EUROCRYPT'99*, págs. 140-149, 1999.

[ETZ80] M. Etzel, W. K. Jenkins, “Redundant Residue Number Systems for Error Detection and Correction in Digital Filters”, *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 29, nº5, 1980.

[FAN03] Y. C. Fan, H. W. Tsao, “Watermarking for intellectual property protection”, *Electronics Letters*, vol. 39, nº 18, págs. 1316-1318, 2003.

[FAS97] FastMan, Inc. *Biorthogonal Wavelet Filter Megafunction*, 1997.

[FER94] D. Fernández, "Intellectual Property Protection in the EDA Industry", in *Proc. 31st Design Automation Conference*, págs. 161–163, 1994.

[FRI99] J. Fridrich, M. Goljan. “Comparing Robustness of Watermarking Techniques”, *Proc. of the SPIE/IS&T Conference on Security and Watermarking of Multimedia Contents*, vol. 3657, 1999.

[FUR07] T. Furon, F. Cayre, G. DoërrG, P. Bas, *Proc. on the 9th International Workshop on Information Hiding: IH 2007*, Revised Selected France Papers, 2007.

[GAN03] Q. Gang, *IP Protection on VLSI CAD: Theory and Practice.* Secaucus, NJ, USA: Kluwer Academic Publishers, 2003.

[GAR97] A. García, A, Lloris, “Pipelined submodular isomorphic mapped RNS multipliers”, *Proc. of XII Circuits and Integrated Systems Conference*, págs. 733-738, 1997.

[GAR98a] A. García, U. Meyer-Baese, F. J. Taylor, “Pipelined Hogenauer CIC filtres using field-programmable logic and residue number system”, *Proc. of 1998 IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 5, págs. 3085-3088, 1998.

[GAR98b] A. García, U. Meyer-Baese, F. J. Taylor, “Pipelined Hogenauer CIC Filters Using Field-Programmable Logic and Residue Number System”,

Proc. 1998 IEEE Int. Conf. on Acoustics, Speech and Signal Processing, vol.5, págs. 3085-3088, 1998.

[GAR99a] A. García, U. Meyer-Baese, F .J. Taylor, "RNS Implementation of FIR Filters Based on Distributed Arithmetic Using Field-Programmable Logic", *Proc. of the 1999 IEEE International Symposium on Circuits and Systems*, vol. 1, págs. 486-489, 1999.

[GAR99b] A. García, *Procesamiento digital de señales de altas prestaciones utilizando el Sistema Numérico de Residuos*. Tesis Doctoral, Universidad de Granada, 1999.

[GAR00] P. Gargini, "The International Technology Roadmap for Semiconductors (ITRS): Past, Present and Future", *Gallium Arsenide Integrated Circuit (GaAs IC) Symposium,* págs. 3-5, 2000.

[GAU66] C. F. Gauss, *Disquisitiones Arithmeticae*. Yale University Press, New Haven, 1966.

[GEM87] S. Geman, *Stochastic relaxation methods for image restoration and expert systems*, Automated Image Analysis: Theory and Experiments, Academic Press, 1987.

[GRI89a] M. Griffin, M. Sousa, F. J. Taylor, "Efficient Scaling in the Residue Number System", *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, págs. 1075-1078, 1989.

[GRI89b] M. F. Griffin, *The Residue Number System, Complex Residue Number Systems, and Digital Signal Processing*. Tesis doctoral, Universidad de Florida, Gainesville, 1989.

[HAM95] V. Hamann, M. Sprachmann, "*F*ast Residual Arithmetic with FPGAs", *Proc. of the Workshop on Design Methodologies for Microelectronics*, 1995.

[HIA97] A. Hiasat, H. S. Abdel-Aty-Zohdy, "Design and Implementation of an RNS Division Algorithm", *Proc. of the 13th IEEE Symposium on Computer Arithmetic*, págs. 240-249, 1997.

[HIA00] A. A. Hiasat, "New Efficient Structure for a Modular Multiplier for RNS", *IEEE Transactions on Computers*, vol. 49, nº 2, págs. 170-174, 2000.

[HOG81] E. B. Hogenauer, "An Economical Class of Digital Filters for Decimation and Interpolation", *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 29, no. 2, págs. 155-162, 1981.

[HON99] I. Hong, M. Potkonjak, "Behavioral Synthesis Techniques for Intellectual Property Protection", *36th Annual Conference on Design Automation*, págs. 849–854, 1999.

[IBR94] K. M. Ibrahim, "Novel digital filter implementation using hybrid RNS-binary arithmetic", *Signal Processing*, vol. 40, págs. 287-294, 1994.

[INT07] *International Technology Roadmap for Semiconductors*, Homepage 2007. `http://www.itrs.net/Links/2007ITRS/Home2007.htm`.

[ISO97] ISO/IEC/JTC1/SC29/WG1 N390R, *JPEG 2000 Image Coding System*, Mar, `http://www.jpeg.org/public/wgln505.pdf`, 1997.

[JAC01] M. F. Jacome, H. P. Peixoto, "A survey of digital design reuse", *IEEE Design & Test of Computers,* vol. 18, no. 3, págs. 98–107, 2001.

[JAI03] A. K. Jain, L. Yuan, P. R. Pari, G. Qu, "Zero Overhead Watermarking Technique for FPGA Designs", *Proc. of the 13th ACM Great Lakes symposium on VLSI*, 2003.

[JEN78] W. K. Jenkins, "A Highly Efficient Residue-Combinatorial Architecture for Digital Filters", *Proc. of the IEEE*, vol. 66, págs. 700-702, 1978.

[JEN79] W. K. Jenkins, "Recent Advances in Residue Number Techniques for Recursive Digital Filtering", *IEEE Transactions on Acoustics, Speech and Signal Processing,* vol. ASSP-27, págs. 19-30, 1979.

[JOH80] E. Johnson, "A Digital Quarter Squarer Multiplier", *IEEE Transactions on Computers*, Marzo 1980.

[JUL78] G. A. Jullien, "Residue Number Scaling and Other Operations Using ROM Arrays", *IEEE Transactions on Computers*, vol. 27, n° 4, pags. 325-357, 1978.

[JUL80] G. A. Jullien, "Implementation of Multiplication, Modulo a Prime Number, with Applications to Number Theoretic Transforms", *IEEE Transactions on Computer*, vol. C-29, nº 10, págs. 899-905, 1980.

[JUL88] G. A. Jullien, M. Taheri, W.C. Miller, "A VLSI radix-2 two-dimensional FFT butterfly with fault tolerance", *Conference on Computers*, 1998.

[KAH98a] A.B. Kahng, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, G. Wolfe, "Robust IP Watermarking Methodologies for Physical Design", *Proc. of the 35th Design and Automation Conference*, págs. 782-787, 1998.

[KAH98b] A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, G. Wolfe, *"*Watermarking Techniques for Intellectual Property Protection*." Proc. Of the Design Automation Conference*, págs. 776-781, 1998.

[KAH01] A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, G. Wolfe, "Constraint-Based Watermarking Techniques for Design IP Protection", *IEEE Transactions on Computer-Aided Design*, vol. 20, nº 10, págs. 1236-52, 2001.

[KAM98] D. L. Kambanis. “DES, one-way hash functions and the MD5”, `http://www.cs.nyu.edu/yingxu/privacy`, 1998.

[KAM99] R. Kamath, P. Alexander, D. Barton, “SLDL: a systems level design language”, *Proc. on the 12th Annual IEEE International ASIC/SOC Conference,* , págs. 71-75, 1999.

[KAT96] S. Katti, “A new residue arithmetic error correction scheme”; *IEEE Transactions on Computers*, vol. 45, nº1, págs. 13-19, 1996.

[KEA98] M. Keating, P. Bricaud, *Reuse Methodology Manual for System-on-a-Chip Designs*, Kluwer Academic Publishers, 1998.

[KIR83] S. Kirkpatrikk, C. D. Gelai‘t, “Optimization by simulated annealing”, *Science*, vol. 220, págs. 671-680. 1983.

[KIR06] D. Kirovski, Y.-Y. Hwang, M. Potkonjak, J. Cong, “Protecting Combinational Logic Synthesis Solutions”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, issue 12, págs. 2687-2696, 2006.

[KOC99] P. Kocher, J. Jaffe, “Differential Power Analysis : Leaking Secrets”. *Proc. of Advances in Cryptology, CRYPTO'99*, págs. 388-397. 1999.

[KOU05] F. Koushanfar, I. Hong, M. Potkonjak, “Behavioral synthesis techniques for intellectual property protection”, *ACM Transactions on Design Automation of Electronic Systems*, vol. 10, no. 3, págs. 523-545, 2005.

[KRI94] H. Krishna, B. Krishna, K. Y. Lin, J. D. Sun, *Computational Number Theory and Digital Signal Processing. Fast Algorithms and Error Control Techniques*. CRC Press, 1994.

[KRI98] H. Krishna, *Digital Signal Processing Algorithms, Number Theory, Convolution, Fast Fourier Transforms, and Applications*. CRC Press, 1998.

[KUT00] M. Kutterf, F. A. P. Peticolas. "The Watermak Copy Attack", *Proc. on the SPIE, Security and Watermarking of Multimeduia Contents Conference*, vol. 3971, págs. 371-380, 2000.

[LAC98a] L. Lach, W.H. Mangione-Smith, M. Potkonjak, "Signature Hiding Techniques for FPGA Intellecual Property Protection", *Proc. IEEE International Conference on Computer Aided Design*, págs. 194-198, 1998.

[LAC98b] L. Lach, W. H. Mangione-Smith, M. Potkonjak, "FPGA fingerprinting techniques for protecting intellectual property", *Proc. IEEE Custom Integrated Circuit Conference*, págs. 299-302, 1998.

[LAC98c] L. Lach, W. H. Mangione-Smith, M. Potkonjak, "Fingerprinting Digital Circuits on Programmable Hardware", *Proc. of the Second International Information Hiding Workshop*, págs. 16-31, 1998.

[LAC01] J. Lach, W. H. Mangione-Smith, M. Potkonjak, "Fingerprinting Techniques for Field Programmable Gate Array Intellectual Property Protection", *IEEE Transactions on Computer-Aided Design*, vol. 20, nº 10, págs.1253-61, 2001.

[LEE91] E. K. F. Lee, P. G. Gulak, "A CMOS Field-Programmable Analog Array", *IEEE Journal Solid-State Circuit*, vol. 26, nº12, págs. 1860–1867, Dec. 1991.

[LEW73] T. Lewis, W. Payne, "Generalized Feedback Shift Register Pseudorandom Number Algorithm", *Journal of the Association for Computing Machinery*, vol. 20, no. 3 , págs. 456-458, July 1973.

[LIN83] S. Lin, D. J. Costello, *Error control coding: Fundamentals and Applications*. Prentice-Hall, Inc. Englewood Ciffs, New Yersey, 1983.

[LIP81] J. D. Lipson, *Elements of algebra and algebraic computing*. Addison-Wesley, 1981.

[LLO03] A. Lloris, A. Prieto, L. Parrilla, *Sistemas Digitales*. Mc Graw Hill, 2003.

[LUC04] M.J. Lucena, *Criptografía y Seguridad en Computadores,* 3ª ed., 2004.

[MAC76] F. MacWilliams, J. Sloane, "Pseudo–Random Sequences and Arrays", *Proc. of the IEEE*, págs. 1715-29, 1976.

[MAC01] I. R. Mackintosh, "Intellectual Property Protection White Paper: Schemes, Alternatives and Discussion Version 1.0", *Virtual Socket Interface Alliance*, January 2001.

[MAL89a] S. G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation", *IEEE Transactions on Pattern Recognition and Machine Intelligence*, vol. 11, nº 7, págs. 674-693, 1989.

[MAL89b] S. G. Mallat, "Multifrequency channel decompositions of images and wavelet models", *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, nº 12, págs. 2091-2110, 1989.

[MAN72] D. Mandelbaun, "Error Correction in Residue Arithmetic", *IEEE Transactions on Computer*, vol. 21, n° 6, págs. 538-545, 1980.

[MEN96] A. J. Menezes, P. C. van Oorschot, S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

[MES06] D. Mesquita, B. Badrignans, L. Torres, G. Sassattell, M. Robert, J. C. Bajard, F. Moraes, "A Leak Resistant Architecture Against Side Channel

Attacks", *Proc. 16th International Conference on Field Programmable Logic and Applications FPL'2006*, págs. 487-492, 2006.

[MET53] **N.** Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, E. Teller, "Equation of state calculation by fast computing machines", *Journal of Chemistry Physics*, vol. 21, págs. 1087-1091, 1953.

[MEY99] U. Meyer-Baese, J. Buros, W. Trautmann, F. Taylor, "Fast Implementation of Orthogonal Wavelet Filterbanks Using Field-Programmable Logic", *Proc. of the 1999 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 99)*, vol. 4, págs. 2119-2122, 1999.

[MEY01] U. Meyer-Baese, A. García, F. Taylor, "Implementation of a Communication Channelizer Using FPGAs and RNS Arithmetic", *Journal of VLSI Signal Processing*, vol. 28, nº 1/2, págs. 115-128, 2001.

[MEY07] U. Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*. Springer, 2007.

[MIL99] M. L. Miller, J. A. Bloom, **"**Computing the Probability of False Watermark Detection Source", *Lecture Notes In Computer Science*, *Proceedings of the Third International Workshop on Information Hiding*, vol. 1768, págs.146-158 , 1999.

[NI05] M. Ni, Z. Gao, "Detector-based watermarking technique for soft IP core protection in high synthesis design level", *Proc. of the International Conference on Communications, Circuits and Systems, 2005*. vol. 2, págs -352, 2005.

[MOU99] P. Moulin, J.A. 0'Sullivan. "Information–theoric Analysis of Information Hiding" *Proc. of the IEEE International Symposium on Information Theory*, pág. 19, 2000.

[NAR01] N. Narayan, R. D. Newbould, J. D. Carothers, J. J. Rodriguez, W. T. Holman, "IP protection for VLSI designs via watermarking of routes", *Proc. of 14th Annual IEEE International ASIC/SOC Conference*, págs. 406–410, Sep. 2001.

[NEW02] R. D. Newbould, D. L. Irby, J. D. Carothers, J. J. Rodriguez, W. Holman, "Watermarking ICs for IP protection", *Electronics Letters*, vol. 38 , nº 6 , págs. 272–274, 2002.

[NUS76] H. Nussbaumer, "Digital filters using Read-Only Memories", *Electronics Letters*, vol. 12, 1976.

[OBR94] K. 0'Brien, S. Maginot, "Non-Reversible VHDL Source-Source Encryption", *Proc. of the Conference on European Design Automation*, págs. 480-485, 1994 .

[OTR92] G. A. Orton, L. E. Peppard, S. E. Tavares, "New fault tolerant techniques for residue number systems", *IEEE Transactions on Computers,* vol. 41, nº11, págs. 1453-1464, 1992.

[PAR94] L. Parrilla, J. Ortega, A. Lloris, "Using simulated annealing in the minimisation of AND-EXOR functions", *Electronics Letters,* vol. 30**,** no. 22, págs. 1838-1839, 1994.

[PAR99] F. Pardo, J. A. Boluda, *VHDL: Lenguaje para síntesis y modelado de circuitos*. Ed. Ra-Ma, abril 1999.

[PAR99] L. Parrilla, J. Ortega, A.Lloris, "Nondeterministic AND-EXOR minimisation by using rewrite rules and simulated annealing", *IEE Proc. Comput. Digital Technology*, vol. 146, no. 1, 1999.

[PET98] F. A. P. Peticolas, R. J. Anderson, M. G. Kuhn. "Attacks on Copyright Marking Systems", *Lecture Notes in Computer Science,* págs. Vol. 1525, págs. 218-238, 1998.

[PET99] F. A. P. Peticolas, R. J. Anderson, M. G. Kuhn. “Information Hiding, A Survey”, *Proc. of the IEEE*, vol. 87, nº 7, págs. 1062-1077, 1999.

[PET00] F. A. P. Petitcolas, “Watermarking Schemes Evaluation”, *IEEE Magazine on Signal Proceesing*, vol. 17, no. 5, pp. 58-64, 2000.

[PFI96] B. Pfitzmann, M. Schunter, “Asymmetric fingerprinting”, *Proc. Int. Conf. Theory and Application of Cryptographic Techniques*, págs. 84–95, 1996.

[PIE98] E. Pierzchala, G. Gulak, L. O. Chua, A. Rodríguez-Vázquez, *Field-Programmable Analog Arrays*, Kluwer Academic Publishers,1998.

[PIV98] A. Piva, M. Barni, F. Bartolini. “Copyright protection of digital images by means of frequency domain watermarking. Mathematics of Data/Image Coding, Compression, and Encryption”, *Proc. of SPIE*,vol. 3456, págs. 25-35, 1998.

[POL76] J. M. Pollard, “Implementation of Number-Theoretic Transforms”, *Electronics Letters*, vol. 12, nº 22, págs. 378-379, 1976.

[POS96] K. C. Posch, R. Posch, “Division in Residue Number Systems Involving Length Indicators”, *Journal of Computational and Applied Mathematics*, vol. 66, págs. 411-419, 1996.

[PRE95] V. Preis “Reuse Scenario for the VHDL Based Hardware Design Flow”, *Proc. European Design Automation Conference with EURO-VHDL 95, IEEE CS Press*, págs. 464-469, 1995.

[PUR95] M. Purser, *Introduction to error correcting codes*. Boston, Artech House, 1995.

[RAD91] D. Radhakrishnan, Y. Yuan, "Fast and Highly Compact RNS Multipliers", *International Journal of Electronics*, vol. 70, nº 2, págs. 281-293, 1991.

[RAM01] J. Ramírez, *Nuevas estructuras RNS para la síntesis VLSI de sistemas de procesamiento digital de señales*. Tesis Doctoral, Nov. 2001.

[RAM02a] J. Ramírez, U. Meyer-Baese, "High Performance, Reduced Complexity Programmable RNS-FPL Merged FIR Filters", *Electronics Letters*, vol. 38, nº 4, págs. 199-200, 2002.

[RAM02b] J. Ramírez, A. García, U. Meyer-Baese, A. Lloris, "Fast RNS-based FPL Communications Receiver Design and Implementation", *Lecture Notes in Computer Science*, vol. 2438, págs. 472-481, 2002.

[RAM03a] J. Ramírez, U. Meyer-Baese, F. Taylor; A. García; A. Lloris, "Design and Implementation of High-Performance RNS Wavelet Processors Using Custom IC Technologies", *Journal of VLSI Signal Processing*, vol. 34, págs. 227-237, 2003.

[RAM03b] J. Ramírez, A. García, U. Meyer-Baese; D. González; L. Parrilla; A. Lloris, "Benchmarks for Programmable FIR Filters Built in RNS-FPL Technology", *Jornadas sobre Computación Reconfigurable y Aplicaciones*, págs. 167-172, 2003.

[RAM03c] J. Ramírez, A. García, D. González, E. Castillo, L. Parrilla, A. Lloris, "Implementación de filtros adaptativos usando el Sistema Numérico de Residuos", *III Jornadas de Computación Reconfigurable y Aplicaciones (JCRA 2003)*, págs. 10-12, 2003.

[RAS99] A. Rashid, J. Asher, W. H. Mangione-Smith, M. Potkonjak, "Hierarchical Watermarking for Protection of DSP Filter Cores", *Proc of the IEEE Custom Integrated Circuits*, págs. 39-42, 1999.

[RIV78] R. Rivest, A. Shamir, L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", *Communications of the ACM*, vol. 21, nº2, págs.120–126. 1978.

[RSA] Página web del RSA, `http://www.rsa.com/rsalabs/node.asp?id=2182`.

[RON99] X. Rong Li, *Probability, Random signals and statistics*. CRC press, 1999.

[SAF97] H. Safiri, H. Ahamadi, G. Jullien V. Dimitrov, "Design of FPGA Implementation of Systolic FIR Filters Using Fermat Number ALU", *Proc. of the Asilomar Conference on Signals, Systems and Computers*, 1997.

[SCH96] B. Schneier, *Applied Cryptography*. 2nd Edition Wiley & Sons, 1996.

[SLD08] *VI System Level Design Language Home Page***,** `http://www.eda.org/sldl/`

[SOD80] M. Soderstrand, C. Vernia, "A High-Speed Low-Cost Modulo Pi Multiplier with RNS Arithmetic Application", *Proc. of the IEEE*, vol. 68, págs. 529-532, 1980.

[SOD86] M. A. Sodersterand, W. K. Jenkins, G. A. Jullien, F. J. Taylor, "Residue Number System Arithmetic: Modern Applications in Digital Signal Processing", *IEEE Press*, 1986.

[STA98] W. Stallings, *Cryptography and Network Security: Principles and Practice.* Second Edition, Prentice Hall, 1998.

[STE98] S. E. Schulez, "The New System-level Design Language", *Integrated System Design*, págs. 26-34, Jul. 1998.

[STI95] D. R. Stinson. *Cryptography : Theory and Practice*. Boca Raton, FL: CRC Press, 1995.

[STR97] G. Strang, T. Nguyen, *Wavelets and Filter Banks*. Wellesly-Cambridge Press, 1997.

[SZA67] N. S. Szabo, R.I. Tanaka, "Residue Arithmetic and its Applications to Computer Technology", McGraw-Hill, 1967.

[TAY84] F. Taylor, "Residue Arithmetic: A Tutorial with Examples", *IEEE Computer*, vol. 17, n° 5, págs. 50-62, 1984.

[TIW08] 5th international workshop on information hiding `http://vision.ece.ucsb.edu/ihw08/`.

[TOR00] I. Torunoglu, E.Charbon, "Watermarking-Based Copyright Protection of Sequential Functions", *IEEE Journal of Solid-State Circuits*, vol. 35, nº. 3, págs. 434–440, 2000.

[TOR01] F. J. Torres-Valle, *Síntesis y descripción de circuitos digitales utilizando VHDL*. Universidad Autónoma de Guadalajara, México, 2001.

[TSE79] B. D. Tseng, G. A. Jullien, W. C. Miller, "Implementation of FFT Structures Using the Residue Number System", *IEEE Transactions on Computers*, vol. C-28, págs. 831-844, 1979.

[VCX08] Virtual Component Exchange (VCX); *VCX Homepage 2008*, `http://www.vcx.org`.

[VET95] M. Vetterli, J. Kovacevic, *Wavelets and Subband Coding*. Prentice Hall, 1995.

[VOL59] J. E. Volder, "The CORDIC trigonometric computing technique", *IRE Transactions on Electronics and Computers*, vol. 8, págs. 330-334, 1959.

[VOL01] S. Voloshynovskiy, S. Pereira, T. Pun, J. J. Eggers, J. K. Su, "Attacks on Digital Watermarks: Classification, Estimation-based attacks and Benchmarks", *IEEE Communication Magazin*, 2001.

[VSI08] Virtual Socket Interface Alliance (VSIA); *VSIA Homepage 2008*; `http://www.vsi.org`.

[VSI02] VSI Alliance: *The Value and Management of Intellectual Assets*. White Paper, V1.0. June, 2002.

[WAL93] C. D. Walker, "Systolic Modular Multiplier", *IEEE Transactions on Computers*, vol. 42, nº 3, págs. 376-378, 1993.

[WAS82] S. Waser, M.J. Flynn, *Introduction to arithmetic for digital systems designers*. Holt, Rinehart & Winston, 1982.

[WHI89] S. A. White, "Applications of distributed arithmetic to digital signal processing, L tutorial review", *IEEE Acoustics, Speech and Signal Processing Magazine*, págs. 4-19, 1989.

[WOL94] W. Wolf, *Modern VLSI Design*. Englewood Cliffs, NJ: Prentice–Hall, 1994.

[XIL08] *Xilinx Homepage* `http://www.xilinx.com/products/silicon_solutions/fpgas.`

[YUA05] L. Yuan, P.R. Pari, G. Qu, "Soft IP Protection: Watermarking HDL codes", *Lecture Notes in Computer Science*, vol. 3200, 224-238. 2005.

[ZIR08] P. Zimmerman, *Por qué escribí PGP*, Declaraciones de Phill Zimmerman. `http://www.pgpi.org/pgpi/`.