Rondas Parcialmente Desenrolladas para Implementaciones a 1 Gbps en FPGA de los Algoritmos SHA-1 y MD5

Ignacio Algredo-Badillo, René Cumplido-Parra, Claudia Feregrino-Uribe

Instituto Nacional de Astrofísica, Óptica y Electrónica, INAOE, Coordinación de Ciencias Computacionales,
Luis Enrique Erro #1, CP 72840, Sta. Ma. Tonantzintla, Puebla, México.
talion00z@ccc.inaoep.mx, {rcumplido,cferegrino}@inaoep.mx

Abstract. This work reports on the FPGA implementations for SHA-1 and MD5 cryptographic algorithms with a throughput superior to 1 Gbps. The designs do not use segmentation stages but logic replication. The main contribution of this work is the use of a partial loop unrolling technique that allows to obtain better performance hardware implementations. The design and implementation of the SHA-1 algorithm just uses partially unrolled rounds, while the MD5 besides the partially unrolled rounds it uses local memory resources.

Resumen. Este trabajo reporta las implementaciones en FPGA con un desempeño mayor al gigabit por segundo para los algoritmos criptográficos SHA-1 y MD5. Los diseños no utilizan etapas de segmentación sino la réplica de lógica. Las aportaciones de este trabajo radican en la explotación de la técnica de desenrollamiento parcial de rondas que permiten lograr un mejor desempeño en las implementaciones hardware. El diseño e implementación del algoritmo SHA-1 utiliza rondas parcialmente desenrolladas, mientras que la implementación del MD5 hace uso de recursos de memoria local y de rondas parcialmente desenrolladas.

Palabras Clave: FPGA, algoritmos criptográficos, función resumen.

1 Introducción

La criptografía asimétrica permite autentificar información, es decir, asegurar que un mensaje proviene de un emisor legal y no de cualquier otro. La autenticación debe hacerse empleando una función resumen y no codificando el mensaje completo. Las funciones resumen son también conocidas como MDC (*modification detection codes*) y permiten crear firmas digitales [1]. En la figura 1, se muestra la estructura iterativa de una función resumen. En general, las funciones resumen se basan en funciones de compresión, las cuales generan bloques de longitud *m* a partir de bloques de longitud *n*. Estas funciones de compresión trabajan en cadena, haciendo que la salida de un proceso anterior forme parte o sea dependiente de la entrada del proceso actual,

reduciendo las posibilidades de que dos mensajes con diferentes longitudes generen el mismo valor en su resumen.

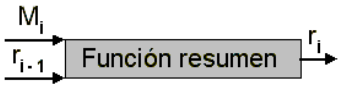

Fig. 1. Estructura iterativa de una función resumen, la cual produce una salida que es utilizada para cifrar el siguiente mensaje de entrada

Existen varios estándares para establecer la velocidad de transferencia de información y el Giga-Ethernet es uno de los más importantes y rápidos, el cual trabaja transmitiendo datos a un gigabit por segundo. La mayoría de las implementaciones de las funciones resumen, tanto en hardware como en software, no tienen desempeños mayores al gigabit por segundo, a excepción de las implementaciones en circuitos integrados de aplicación específica (ASICs).

El objetivo de este trabajo es obtener las implementaciones en un FPGA (*Field Programmable Gate Array*), dispositivo hardware configurable, de las funciones resumen SHA-1 y MD5 con un desempeño mayor a 1 Gbps.

La metodología utilizada fue la realización de los diseños modulares utilizando el número mínimo de módulos implementados en FPGA, para analizar los caminos críticos (*critical paths*) y el desempeño. Estos diseños se modificaron utilizando la técnica de desenrollamiento parcial de rondas.

1.1 Algoritmo SHA-1

El algoritmo SHA-1 (*Secure Hash Algorithm*) fue publicado por el NIST (*National Institute of Standards and Technology*) [3], siendo una versión mejorada del SHA. SHA-1 es una función resumen (*hash function*) basada en el algoritmo MD4, por lo que tiene similitudes con el algoritmo MD5 [2]. SHA-1 es una función constituida por un búfer estado de 160 bits y trabaja con cuatro rondas conformadas por operaciones elementales de 32 bits. En lugar de procesar cada bloque del mensaje cuatro veces, SHA-1 utiliza una recurrencia lineal para utilizar 80 palabras de las 16 palabras de entrada del bloque que se está procesando. Esta recurrencia lineal asegura que cada bit del mensaje afectará las funciones internas al menos una docena de veces. La salida de SHA-1 es un resumen de 160 bits, ver figura 2.a. Para detalles ver [3].

1.2 Algoritmo MD5

MD5 (*Message Digest Algorithm* 5, Algoritmo de Ordenación de Mensajes 5) es un algoritmo seguro *desarrollado* por RSA Data Security, Inc [4]. MD5 es una función resumen de 128 bits, que toma como entrada un mensaje de tamaño arbitrario y produce como salida un resumen del mensaje de 128 bits. El MD5 no sirve para cifrar un mensaje ya que lo destruye completamente, la información no es recuperable de ninguna manera ya que hay pérdida de información. El primer paso es dividir el

mensaje en bloques de 512 bits. El último bloque o si el mensaje completo es menor a 512 bits, se formatea (*padding*) para tener un tamaño de 512 bits mediante el agregado de bits 0 más la longitud del tamaño del mensaje. Además, se tiene un búfer estado de 128 bits manejado como cuatro palabras de 32 bits. La función compresión tiene cuatro rondas y en cada ronda el bloque de mensaje y el búfer son combinados en el cálculo, mediante el uso de sumas modulares, XOR's, AND's, OR's y operaciones de rotaciones sobre palabras de 32 bits [2]. Para mayores detalles ver [4]. Cada ronda combina el bloque de 512 bits del mensaje con el búfer estado, así que cada palabra del mensaje es usada cuatro veces. Después de las cuatro rondas de la función compresión, el búfer estado y el resultado se suman (sumas módulo 2^{32}) para obtener la salida [5], ver figura 2.b.

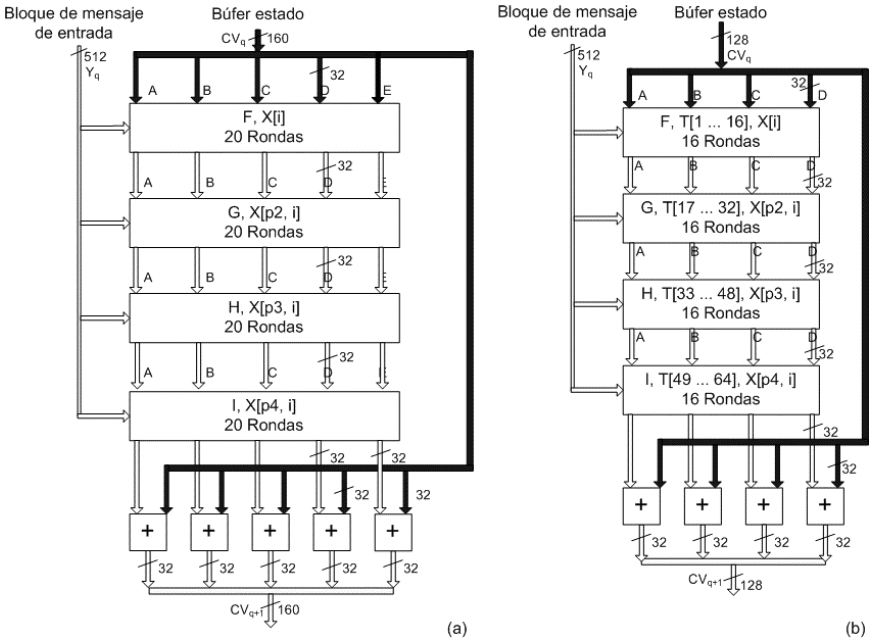

Fig. 2. (a) Estructura general del algoritmo SHA-1, para detalles ver [3]. (b) Estructura general del algoritmo MD5, para detalles ver [4]

2 Trabajo relacionado

La búsqueda de información se enfocó a implementaciones FPGAs, tanto trabajos de investigación como productos comerciales que utilizan diferentes técnicas de diseño. La característica a comparar es el desempeño de cada implementación.

2.1 SHA-1

El trabajo en [6] desarrolla las implementaciones de tres funciones resumen (SHA-1, HAS-160 y MD5). La estrategia de diseño considera utilizar un número mínimo de compuertas. Las implementaciones se realizaron en el dispositivo Altera EP20K1000EBC652-3. La implementación del SHA-1 presentada procesa 114 Mbps a una frecuencia de reloj de 18MHz utilizando 81 ciclos de reloj. En [7] se diseñan las implementaciones de las funciones MD5 y SHA-1 en un FPGA Xilinx Virtex 2V3000, donde se presentan resultados de la síntesis en ISE 4.1 para la implementación del algoritmo SHA-1, el cual procesa 899.8 Mbps con una arquitectura parcialmente desenrollada con un bloque combinacional de 4 rondas.

El trabajo en [8] implementa el SHA-1 y el SHA-512 en un Xilinx XCV-1000-6, donde el diseño del SHA-1 alcanza un procesamiento de 462 Mbps con la máxima frecuencia de reloj de 75.76 MHz (del analizador temporal) y alcanzan 530 Mbps con una frecuencia de 86.96 MHz de manera experimental. Amphion en [9] ofrece la implementación comercial del MD5 y SHA-1 en un mismo dispositivo Xilinx Virtex-II, procesando a 350 Mbps a una frecuencia de reloj de 56 MHz para el SHA-1. Cast [10] muestra la implementación comercial del algoritmo SHA-1 en hardware. Reportan varias implementaciones en diferentes dispositivos Xilinx, pero el Virtex-II XCV2V500-6 presenta el mayor procesamiento, 498.4 Mbps a 79 MHz. En [11] se presenta la implementación comercial del SHA-1 de acuerdo al estándar FIPS 180-1, con 81 ciclos para el cálculo del resumen y no realiza el formateo del mensaje de entrada. Esta hoja de especificaciones reporta resultados de implementaciones en diferentes dispositivos FPGA, el mayor procesamiento se alcanza con el Virtex-II procesando, 644 Mbps a 102 MHz.

Los trabajos revisados muestran que las implementaciones en FPGA tienen desempeños menores a 1Gbps, donde el diseño SHA-1 con mayor velocidad de procesamiento presenta una arquitectura de rondas desenrolladas mostrando sólo resultados de síntesis, es decir, falta la implementación en un FPGA que causa retardos extras al enrutar los recursos del dispositivo y obtener una funcionalidad determinada.

2.2 MD5

El trabajo [6], revisado en la sección 2.1 muestra las implementaciones de tres funciones resumen, donde el desempeño de la implementación del MD5 es de 142 Mbps a una frecuencia de reloj de 18MHz utilizando 65 ciclos de reloj. El trabajo [7], ver sección 2.1, reporta las implementaciones de las funciones MD5 y SHA-1, donde la implementación del algoritmo MD5 procesa 467.3 Mbps con un diseño simple sin optimización. El trabajo en [12] presenta dos diseños para la implementación del algoritmo MD5, sobre un Virtex V1000FG680-6. El diseño iterativo procesa 165 Mbps a 21 MHz, mientras que el diseño con lazos desenrollados trabaja a 71.4 MHz con un procesamiento de 354 Mbps. Amphion presenta en [13] la implementación comercial del algoritmo MD5. La mayor velocidad de procesamiento reportado es de 472 Mbps en un dispositivo Xilinx Virtex-II a 60 MHz. En [9], ver sección 2.1, se tiene la implementación del MD5, la cual procesa 400 Mbps a 56 MHz.

Las distintas arquitecturas revisadas reportan desempeños menores al gigabit por segundo y el mejor rendimiento es de 472 Mbps en un diseño simple. En [12] se utiliza la técnica de diseño de desenrollamiento de lazos, la cual aumenta el desempeño del diseño simple presentado en el mismo reporte.

2.3 Discusión del trabajo relacionado

Los trabajos revisados, tanto de las implementaciones FPGA del algoritmo SHA-1 como del MD5, no tienen desempeños mayores al gigabit por segundo. La técnica de diseño utilizada para aumentar el procesamiento es desenrollar lazos de manera parcial, debido al proceso del cálculo de las funciones resumen que no puede tener una arquitectura con lazos totalmente desenrollados.

3 Implementaciones modulares

En este trabajo, la estrategia de diseño se fundamentó en utilizar eficientemente los recursos de hardware del FPGA para que las implementaciones aportaran información de caminos críticos y de esta manera seguir estrategias de diseño para alcanzar el requerimiento de procesamiento de 1Gbps.

3.1 Diseños con uso eficiente en recursos hardware

Las primeras implementaciones basadas en diseños modulares se enfocaron a usar eficientemente los recursos de hardware. El esquema general usado en la implementación, tanto para el algoritmo SHA-1 como el MD5 se muestra en la figura 3.

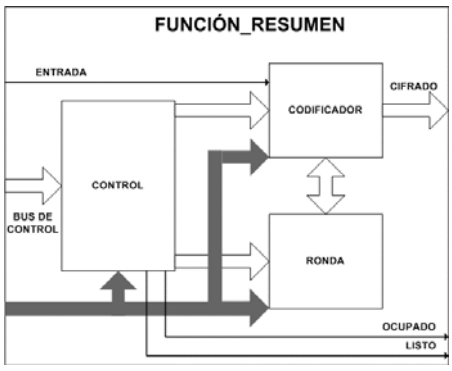

Fig. 3. Estructura general de las funciones resumen, tanto para el algoritmo SHA-1 como del algoritmo MD5

Los diseños fueron escritos y simulados en Active-HDL e implementados en Xilinx ISE 6 para la medición de parámetros de hardware tales como uso de lógica y frecuencia de operación. Los diseños fueron sintetizados, mapeados, colocados y enrutados en un FPGA Xilinx XC2V1000-FG456 con la herramienta Xilinx ISE 6, además se creó un modelo de simulación "Post-Place & Route" que validó el funcionamiento de cada diseño mediante Active-HDL 5.1. Los datos de las implementaciones de la tabla 1 son calculados en base a la información generada por los reportes del proceso Place & Route.

Tabla 1. Resultados de las implementaciones de los algoritmos SHA-1 y MD5 con diseños que usan eficientemente los recursos de hardware

Implementación	Frecuencia	IOBs	Ciclos	Desempeño
SHA1v1	99.44MHz	197	80	636.43 Mbps
MD5v1	51.60MHz	165	65	406.48 Mbps

3.2 Implementación a 1Gbps del algoritmo SHA-1

El diseño SHA1v1 de la tabla 1 usa eficientemente los recursos de hardware y se mejora a través de dos etapas: en la primera etapa se analizó el desempeño al modificar la conexión de los sumadores de la unidad Ronda (ver figura 3) al implementarlo en el FPGA y en la segunda etapa se utilizó la técnica de desenrollamiento parcial de rondas.

El análisis consistió en conectar los sumadores que conforman el módulo Ronda de manera distinta, donde 4 sumadores módulo 2^{32} dentro de la lógica combinacional manejan ocho operandos, intercambiando las entradas de cada sumador. Los resultados de estas interconexiones se muestran en la tabla 2. En la figura 4 se muestra la conexión del primer caso reportado en la tabla 2. Para los siguientes casos las entradas de los sumadores SUM1, SUM2 y SUM3 son intercambiadas conforme a la tabla 2, mientras que SUM4 no cambia. Los datos de la tabla 2 (obtenidos de la herramienta en Place & Route de ISE 6.1) muestran que pueden existir grandes diferencias en el procesamiento sólo por las diferentes conexiones de los sumadores al momento de implementar el diseño en el FPGA. Este análisis es importante al momento de implementar un diseño, ya que se muestra una diferencia de 149.10 Mbps entre el mejor caso con 636.43 Mbps y el peor caso con 487.32 Mbps.

La segunda y última etapa consiste de mejorar el camino crítico, el cual se encuentra en las rondas para el cálculo de los nuevos datos. Esta conclusión es la misma que la reportada en [7], donde el camino crítico se encuentra en el cálculo del nuevo dato:

$$A = A<<5 + F(B, C, D) + E + W + K_1 . \quad (\mathbf{1})$$

El operando A representa la retroalimentación de la ronda para tener un camino no optimizable al realizar un diseño con rondas desenrolladas. Los otros operandos están disponibles y su suma puede hacerse en paralelo con el cálculo de A, es decir, con una arquitectura parcialmente desenrollada. Para una arquitectura de dos rondas desenrolladas, ver figura 5, la ganancia de desenrollar una nueva ronda es evitar el

retardo de seis sumas y no tener que calcular las ocho sumas de las dos rondas, así también se evita el retardo del cálculo de la otra función F.

Tabla 2. Resultados de la conexión entre los sumadores de la unidad Ronda (ver figura 3), cuyas entradas fueron intercambiadas, para detalles ver figura 4

SUM1	SUM2	SUM3	Período (ns)	Desempeño (Mbps)
rE	SW	FK	10.143	630.97
rE	SK	FW	10.225	625.91
rE	SF	KW	11.196	571.63
rW	SE	KF	11.319	565.42
rW	SF	KE	13.133	487.32
rW	SK	FE	11.625	550.53
rF	SK	WE	12.605	507.73
rF	SW	KE	12.337	518.76
rF	SE	WK	11.654	549.16
rK	SE	FW	10.278	622.68
rK	SF	EW	10.596	604.00
rK	SW	FE	11.441	559.39
rE	WS	KF	10.717	597.18
rE	WS	FK	10.056	636.43
Er	WS	FK	10.996	582.02

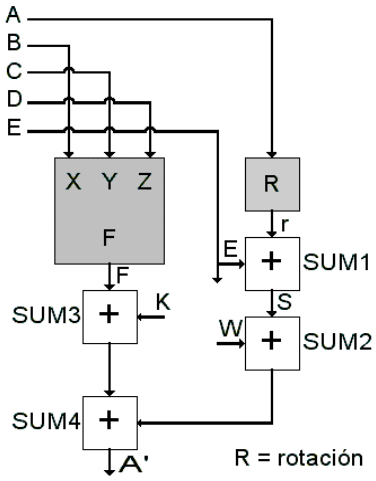

Fig. 4. Conexión de los sumadores modulares de la unidad Ronda del SHA-1, ver figura 3

La figura 5 muestra el diseño de dos rondas parcialmente desenrolladas. En la arquitectura de la nueva implementación se replicó la lógica para tener cuatro rondas parcialmente desenrolladas. Esta configuración aumentó la capacidad de procesamiento de información. Los resultados se muestran en la tabla 3.

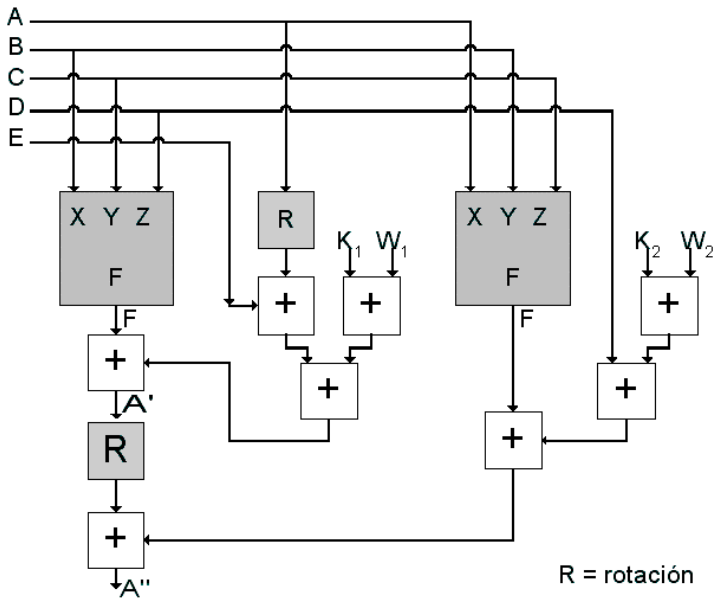

Fig. 5. Estructura de dos rondas desenrolladas parcialmente de la implementación del SHA-1

Tabla 3. Resultados de la nueva implementación del algoritmo SHA-1

Implementación	Frecuencia	Ciclos	Desempeño
SHA1v2	43.30MHz	20	1.109 Gbps

La latencia para el procesamiento de un solo bloque de 512 bits es de 20 ciclos de reloj, ya que el módulo principal consta de 4 rondas parcialmente desenrolladas y 20 ciclos por 4 rondas calculan las 80 rodas establecidas en [3].

3.3 Implementación a 1Gbps del algoritmo MD5

La implementación con un desempeño mayor al gigabit por segundo del algoritmo MD5 también consta de dos etapas: la primera que utilizó el desarrollo de la sección 3.2, la cual no alcanzó el procesamiento de 1 Gbps, por lo que fue necesaria una segunda etapa, generando la nueva descripción de las unidades funcionales en base al desenrollamiento parcial de rondas.
Los resultados obtenidos de la implementación en la sección 3.1 indican que el camino crítico se encuentra en las rondas para el cálculo de los nuevos datos:

$$A = B + ((A + F(B, C, D) + Xi + k1) ,<<< k2) . \quad (2)$$

El operando A representa la retroalimentación de la ronda para tener un camino no optimizable al realizar un diseño con rondas desenrolladas. Los otros operandos están

disponibles y su suma puede hacerse en paralelo con el cálculo de A, es decir, con una arquitectura parcialmente desenrollada. La figura 6 muestra una arquitectura de dos rondas parcialmente desenrolladas, la ganancia de desenrollar dos rondas es evitar el retardo de tres sumas y no calcular ocho sumas, lo cual es una ganancia menor comparada con la implementación del SHA-1, ver sección 3.2. Además, no se evita el retardo del cálculo de una de las funciones F como sucedió con la implementación del SHA-1.

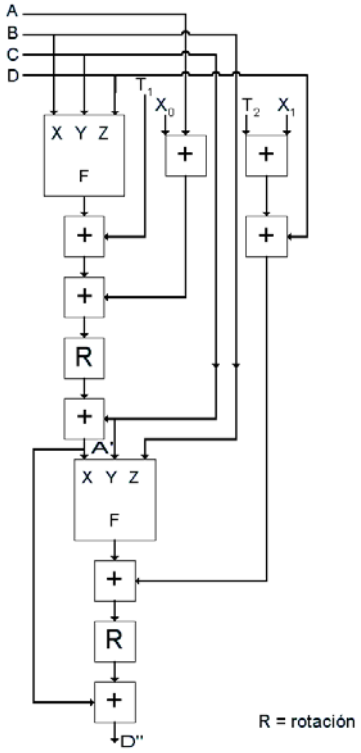

Fig. 6. Estructura de dos rondas parcialmente desenrolladas de la implementación del MD5

Las implementaciones que usaron esta técnica de replicación de lógica generaron aumentos en la velocidad de procesamiento de información hasta cierto punto, ver la figura 7, donde MD5vA1 es la implementación con dos rondas desenrolladas las cuales se utilizan iterativamente durante 32 ciclos de reloj. MD5vA2 utiliza cuatro rondas desenrollados con una latencia de 16 ciclos y MD5vA3 usa ocho rondas desenrolladas en 8 ciclos de reloj. Este decremento se debe al enrutado en el FPGA, porque los resultados de la síntesis indican que MD5vA2 procesa 956.5 Mbps y MD5vA3 procesa 1.146Gbps, es decir no debería haber decremento.

El objetivo principal es alcanzar el procesamiento de 1 Gbps para la implementación del algoritmo MD5 en un FPGA, por lo que la figura 7 muestra que el diseño en base a rondas desenrolladas no es la solución.

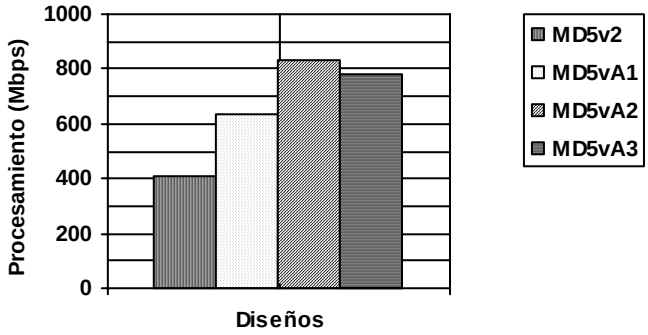

Fig. 7. Comparación del desempeño de las diversas implementaciones del MD5

La segunda etapa se basó en el diseño MD5vA2, la cual es la implementación que alcanza el mayor desempeño al manejar cuatro rondas desenrolladas, por lo cual este diseño es fundamental para mejorar la capacidad de procesamiento. La idea principal es manejar funciones de compresión especiales conformadas de cuatro rondas desenrolladas y evitar tener las unidades de corrimiento, ver figura 8.

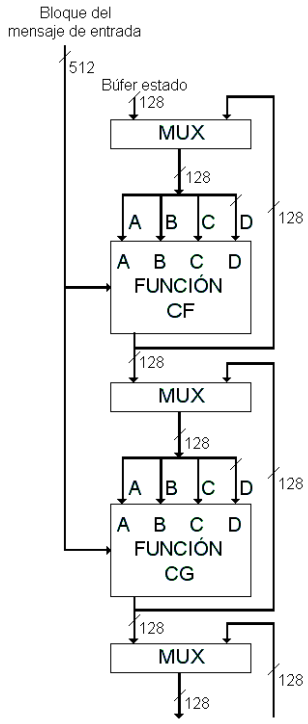

Fig. 8. Nueva descripción estructural de las unidades funcionales de la implementación MD5

La nueva descripción estructural se muestra parcialmente en la figura 8, donde se muestran dos de las cuatro nuevas funciones Ronda de compresión, las cuales están compuestas de cuatro rondas desenrolladas y cada una es utilizada cuatro veces para totalizar una latencia de 16 ciclos de reloj para 64 rondas, descritas en [4].

De esta manera, se tiene la ventaja de tener cuatro rondas desenrolladas (ver figura 4), utilizando cuatro memorias RAM de 4x128 y además se evitan unidades de corrimiento que dependían del número de rondas (64 rondas descritas en [4] con 16 valores de corrimientos). Los resultados de la implementación de esta estructura se muestran en la tabla 4.

Tabla 4. Resultados de la nueva implementación del algoritmo MD5

Implementación	Frecuencia	IOBs	Ciclos	Desempeño
MD5v2	32.47 MHz	261	16	1.039 Gbps

La latencia para el procesamiento de un solo bloque de 512 bits es de 16 ciclos de reloj, ya que cada una de las cuatro funciones Ronda son utilizadas cuatro veces (4x4 pasos) y cada función ronda se compone de cuatro rondas desenrolladas (4x4x4 pasos = 64 pasos). Los resultados de la síntesis reportan un período 25.395ns para obtener un rendimiento de 1.26 Gbps, pero al implementarlo en el FPGA presenta un desempeño de 1.039 Gbps.

3 Conclusiones

Se presentan las implementaciones de los algoritmos SHA-1 y MD5 para el proceso de bloques de 512 bits, el agregar retroalimentación de la salida para procesos de multibloques no debe incluir retardos significativos en el camino crítico, ya que se registraría la salida inmediatamente y después se seleccionaría adecuadamente la retroalimentación.

Para la implementación a 1 Gbps del SHA-1, se realizan implementaciones que usan eficientemente los recursos del FPGA, analizándose los caminos críticos de las diferentes conexiones de los sumadores de la función ronda para obtener un mejor desempeño. y utilizándose la técnica del desenrollamiento parcial de rondas.

Para la implementación a 1 Gbps del MD5, los resultados de las implementaciones en FPGA indican que utilizar la técnica de desenrollamiento de rondas no es suficiente para tener un desempeño mayor a 1 Gbps (ver sección 3.3), por lo que se realizan nuevas descripciones en hardware de bloques funcionales, las cuales utilizan recursos de memoria local con rondas parcialmente desenrolladas, aumentando así la velocidad de procesamiento de información en la implementación en FPGA.

Las soluciones FPGA de los algoritmos SHA-1 y MD5 presentadas en este artículo tienen mejor desempeño que el reportado en la literatura (ver sección 2) con un rendimiento de 1.109 Gbps y de 1.039Gbps, respectivamente.

Referencias

1. Lucena, M. J.: Criptografía y Seguridad en Computadores, Libro Electrónico, Tercera Edición (2001), disponible en http://wwwdi.ujaen.es/~mlucena/lcripto.html
2. Ferguson N., Schneier B.: Practical Cryptography, Wiley Publishing, Inc., EUA (2003)
3. Federal Information Processing Standards (FIPS) Publication 180-2.: Announcing the Secure Hash Standard, US DoC/NIST (2002)
4. Rivest, R.: The MD5 Message-Digest Algorithm, RFC 1321, MIT and RSA Data Security, Inc. (1992)
5. Stallings, W.: Cryptography and Network Security. Principles and Practices, Prentice Hall, EUA (2003)
6. Yong K. K., Dae W. K., Taek W. K., Jun R. C.: An Efficient Implementation of Hash Function Processor for IPSec, The Third IEEE Asia-Pacific Conference on ASICs, Taipei, Taiwan (2002)
7. Diez J. M., Bojaniç S., Stanimiroviç Lj., Carreras C., Nieto-Taladriz O.: Hash Algorithms for Cryptographic Protocols: FPGA Implementations, 10th Telecommunications Forum TELFOR'2002, Belgrade, Yugoslavia (2002)
8. Grembowski, T., Lien, R., Gaj, K., Nguyen, N., Bellows, P., Flidr, J., Lehman, T., Schott, B.: Comparative Analysis of the Hardware Implementations of Hash Functions SHA-1 and SHA-512, Information Security, 5th International Conference ISC 2002, Sao Paulo, Brasil (2002)
9. Amphion, Hoja de especificaciones técnicas.: CS5316 High Performance SHA1/MD5 Hashing Algorithm Core, (2002), disponible en www.amphion-semi.com/acrobat/DS5316.pdf
10. CAST, Inc., Hoja de especificaciones técnicas.: SHA-1 Processor, (2002), disponible en www.cast-inc.com/sha-1/cast_sha-1.pdf
11. Alma Technologies, Hoja de especificaciones técnicas.: SHA-1 High Performance Hash Function, (2002), disponible en www.alma-tech.com/Data-Sheets/SHA-1_pre_sales.pdf
12. Deepakumara J., Heys H. M., Venkatesan.: FPGA Implementation of MD5 Hash Algorithm, Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering (CCECE 2001), Toronto, Ontario (2001)
13. Amphion, Hoja de especificaciones técnicas.: CS5315 High Performance Message Digest 5 Algorithm (MD5) Core, (2002), disponible en www.amphion-semi.com/acrobat/DS5315.pdf