Implementación de un Módulo SHA-1 para una Plataforma Reconfigurable Criptográfica en FPGA a 1 Gbps

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

Coordinación de Ciencias Computacionales
Instituto Nacional de Astrofísica Óptica y Electrónica, INAOE.
e-mail: talion00z@ccc.inaoep.mx, rcumplido@inaoep.mx, cferegino@inaoep.mx.
Luis Enrique Erro #1, CP 72840, Sta. Ma. Tonantzintla, Puebla, México.

Abstract: This work reports the implementation of the SHA-1 algorithm [1] for a throughput of 1 Gbps without the use of pipeline stages. This implemented module will be part of a library of configurations for a cryptographic reconfigurable platform. In the communication networks, it is necessary to manage diverse authentication standards to obtain a highest security considering that at the present time the transmission speeds manage superior levels to the Mbps and the Gbps, so the main objective is to obtain an implementation that processes data at 1Gbps. This article presents an initial implementation and the followed strategy to reach the required throughput.

Resumen: Este trabajo reporta la implementación del algoritmo SHA-1 [1] para un procesamiento de 1 Gbps sin el uso de etapas de pipeline. El objetivo principal es obtener una implementación que procese datos a una velocidad mayor a 1Gbps e integrarla en una biblioteca de configuraciones para una plataforma reconfigurable criptográfica. La motivación se debe a que en las redes de comunicación, el manejo de diversos estándares de autenticación ofrece una mayor seguridad, considerando que en la actualidad las velocidades de transmisión manejan niveles superiores a los Mbps y los Gbps. En este artículo se presentan las diversas estrategias que se utilizaron para alcanzar la capacidad de procesamiento requerida.

Palabras Clave: FPGA, SHA-1, *hash*, criptografía.

I. Introducción

La criptografía asimétrica permite autentificar información, es decir, asegura que un mensaje proviene de un emisor legal y no de cualquier otro, aunque 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 [2]. En la figura 1 se puede ver 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.

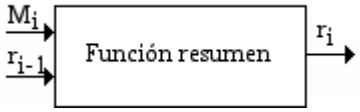

Fig. 1. Función resumen.

II. Algoritmo SHA-1

El algoritmo SHA-1 (*Secure Hash Algorithm*) fue publicado por NIST (*National Institute of Standards and Technology*), siendo una versión mejorada del SHA. SHA-1 es una función *hash* basada en el algoritmo MD4, por lo que tiene similitudes con el MD5 [3].

SHA-1 es una función *hash* 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 esta 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. Para detalles ver [1].

III. Trabajo Relacionado

La búsqueda de información se realizó en base a implementaciones en hardware, especialmente en FPGA's, tanto trabajos de investigación como productos comerciales que utilizan diferentes técnicas de diseño.

El trabajo en [4] desarrolla las implementaciones de tres funciones *hash* (SHA-1, HAS-160 y MD5). La estrategia de diseño considera utilizar un mínimo número de compuertas. Las implementaciones se realizaron en el dispositivo Altera EP20K1000EBC652-3. El resultado de procesamiento presentado es de 114 Mbps a una frecuencia de reloj de 18MHz utilizando 81 ciclos de reloj para el proceso de un bloque. En [5] se diseñan las implementaciones de las funciones MD5 y SHA-1 en un FPGA Xilinx Virtex 2V3000. 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 [6] 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 el mínimo período de

reloj de 13.2ns (del analizador temporal) y alcanzan 530 Mbps con un período mínimo de 11.5ns experimentalmente. Los recursos utilizados del FPGA son del 18% para la implementación del SHA-1.

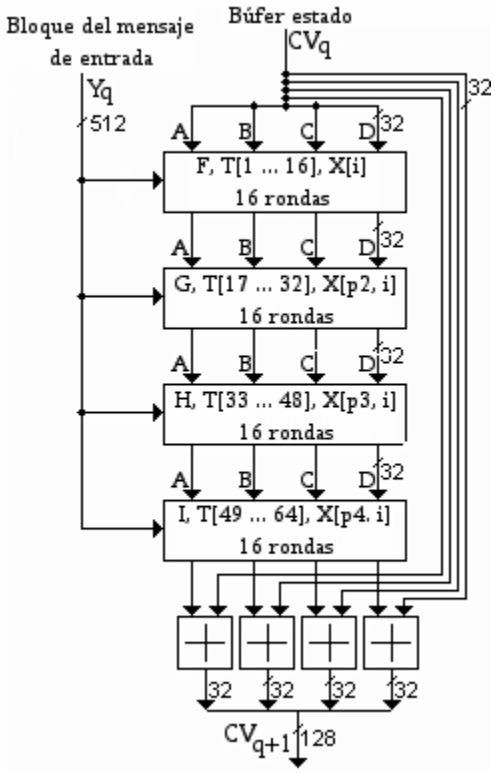

Fig. 2. Procesamiento SHA-1 de un solo bloque de 512 bits.

Amphion en [7] ofrece la implementación comercial del MD5 y SHA-1 en un mismo dispositivo Xilinx Virtex-II utilizando 1382 *slices*, procesando a 350 Mbps a una frecuencia de reloj de 56 MHz para el SHA-1. Cast [8] 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 de 498.4 Mbps a 79 MHz, usando 739 *slices* y 199 pines de entrada. En [9] presenta la implementación comercial del SHA-1 de acuerdo al estándar FIPS 180-1, con 81 ciclos para el cálculo *hash* y no realiza el *padding* del mensaje de entrada. En esta hoja de especificaciones reporta resultados implementaciones en diferentes dispositivos Xilinx y de Altera, el mayor procesamiento se alcanza con el Vritex-II procesando a 644 Mbps a 102 MHz.

En [10] ofrece implementaciones comerciales del algoritmo SHA-1. Hi/fn Inc., tiene varios procesadores, alcanzando el 7811 el más alto procesamiento a 301 Mbps., a una frecuencia de reloj de 90MHz. SecuCore Consulting Services [11] diseña *cores*

comerciales usando tecnología ASIC de 0.18µm y trabajando a una frecuencia de reloj de 166 MHz. El procesamiento es de 1.01 Gbps usando 84 ciclos de reloj para el proceso de un bloque de 512 bits. Por último, el *core* comercial ofrecido por Ocean Logic [12] reporta resultados de implementación para el Virtex-II, el cual utiliza 612 *slices* alcanzando un procesamiento de 498.1 Mbps a 78.8 MHz. Además, presenta la implementación en tecnología ASIC de 0.18µm a 300 MHz procesa a 1.896 Mbps.

Los trabajos revisados muestran que las implementaciones en FPGA tienen velocidades menores a 1Gbps, mientras que las implementaciones en ASIC presentan procesamientos mayores al gigabit por segundo con frecuencias de reloj altas. El diseño en FPGA con mayor procesamiento presenta una arquitectura de rondas desenrolladas.

IV. Implementación y Resultados

La estrategia de diseño se fundamentó en realizar un diseño usando mínimos recursos hardware del FPGA que proveyera información de *paths* críticos y de esta manera seguir estrategias de diseño para alcanzar el requerimiento de procesamiento de 1Gbps.

1. Diseño óptimo en recursos hardware

La primera implementación basada en un diseño modular estableció usar un mínimo de recursos hardware. El diagrama general se puede ver en la figura 3.

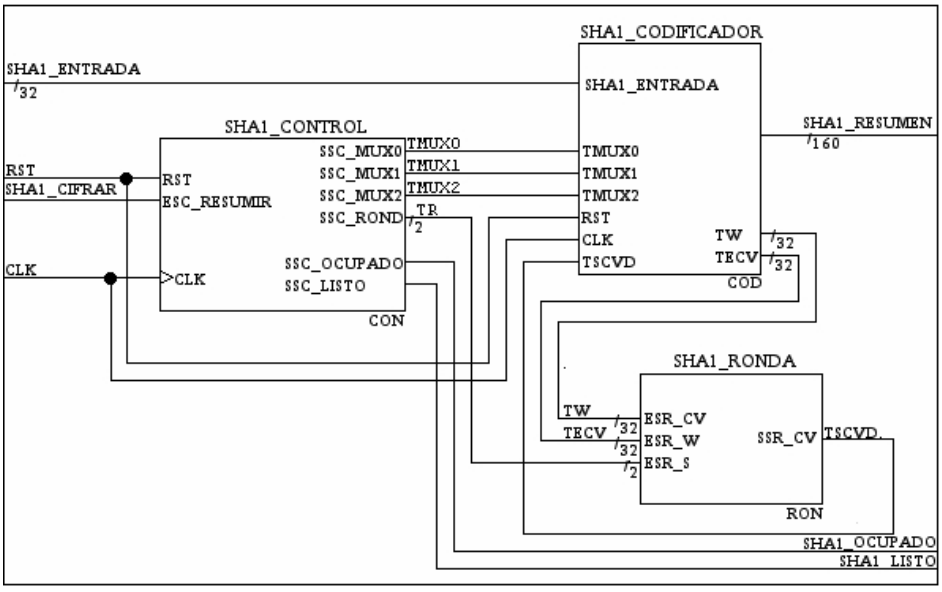

Fig. 3. Diagrama a bloques de la implementación del SHA-1.

A partir del esquema de la figura 3 fundamentado en el FIPS-180-2, se realizaron dos diseños diferentes en el diseño de la unidad de control (basados en máquinas de estados) y la colocación de registros. Estos diseños manejaban una latencia de 82 y 80 ciclos para el proceso de un sólo bloque de 512 bits, sin la realización del *padding* descrito en [1].

Los resultados de la implementación pueden verse en las tablas 1 y 2. 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 ruteados en un FPGA Xilinx XC2V1000-FG456 con la herramienta Xilinx ISE 6, además se generó un modelo de simulación "Post-Place & Route" que validó el funcionamiento de cada diseño mediante Active-HDL 5.1. Los datos de la tabla 1 son calculados en base a la información generada por los reportes del proceso Place & Route.

Diseño	Frecuencia	IOBs	Slices	LUTs 4-E	Ciclos	Procesamiento
SHA1v1	98.88MHz	197/324	830/5120	666/10240	82	617.41 Mbps
SHA1v2	99.44MHz	197/324	579/5120	647/10240	80	636.43 Mbps

Tabla 1. Resultados de la implementación.

El diseño SHA1v1 es implementación con una latencia de 82 ciclos de reloj y con la restricción óptima en velocidad. SHA1v2 es el diseño con una latencia de 80 ciclos, además con un ciclo de reloj más rápido. La implementación SHA1v2 difiere de SHA1v1 en el diseño de la unidad de control y la colocación de registros, sin cambios en el diseño de otros módulos. Estos diseños fueron conformados, en general, por los módulos que se ven en la tabla 2.

Componentes	Descripción	MD5 Unibloque
AND2TO1_32BIT	Compuerta AND de 32 bits, 2 entradas.	4
NOT_32BIT	Compuerta NOT de 32 bits.	1
XOR2TO1_32BIT	Compuerta XOR de 32 bits, 2 entradas.	5
XOR3TO1_32BIT	Compuerta XOR de 32 bits, 3 entradas.	3
REGISTRO_160BIT	Registro de 160 bits.	1
REGISTRO_512BIT	Registro de 512 bits.	1
SUMADOR2TO1_32BIT	Sumador módulo 2^{32} de 32bits, 2 entradas	9
MUX2TO1_32BIT	Multiplexor de 32 bits, 2 entradas.	1
MUX3TO1_32BIT	Multiplexor de 32 bits, 3 entradas.	1
MUX4TO1_32BIT	Multiplexor de 32 bits, 4 entradas.	1
MUX2TO1_160BIT	Multiplexor de 160 bits, 2 entradas.	1
MUX2TO1_512BIT	Multiplexor de 512 bits, 2 entradas.	1
FSM 8 ESTADOS	Máquina de estados	1
CONTADOR_80BCD	Contador.	1

Tabla 2. Módulos utilizados en la implementación.

Estos diseños están compuestos por una función de compresión, la cual se utilizaba de manera iterativa para el proceso de los 80 pasos necesarios en el cálculo de la salida válida del SHA-1 indicado en [1].

Un análisis extra realizado consistió en conectar los sumadores que conforman la función de compresión de manera distinta. Es decir, se utilizan cuatro sumadores módulo 2^{32} dentro de la lógica combinacional, por lo tanto hay ocho operandos para la suma y como una propiedad de la suma describe que no importa el orden de los sumandos, se procedió a cambiar la entrada de cada sumador. Los resultados de estas interconexiones se pueden ver en la tabla 3.

SUM 1	SUM 2	SUM 3	Período (ns)	Procesamiento (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

Tabla 3. Resultados de la interconexión entre sumadores. Para detalles ver figura 4.

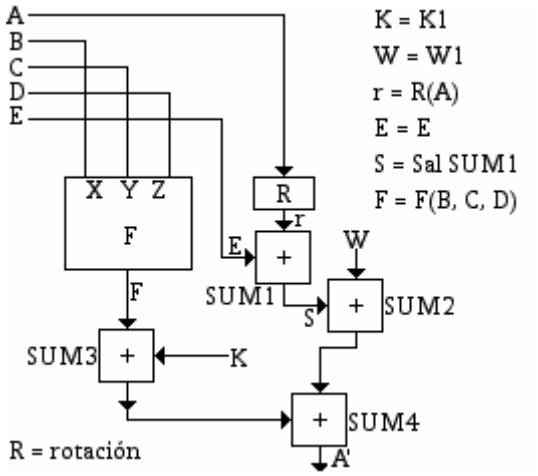

Fig. 4. Conexión entre los módulos suma de la función de compresión.

En la figura 4 se muestra la conexión del primer caso reportado en la tabla 3 y para los siguientes casos, las entradas de los sumadores SUM1, SUM2 y SUM3 son intercambiadas conforme a la tabla 3, mientras que el sumador SUM4 se mantiene sin cambios. Los datos presentados en la tabla 3 (obtenidos de la herramienta en Place & Route de ISE 6.1) muestran que pueden existir grandes diferencias en el procesamiento sólo por las diferentes interconexiones de los sumadores al momento de implementar el diseño en el FPGA. Esta consideración es importante al momento de implementar un diseño, ya que la tabla 3 muestra una diferencia de 149.10 Mbps entre el mejor caso con un procesamiento de 636.43 Mbps y el peor caso con 487.32 Mbps.

2. Diseño con réplica de lógica

Los resultados obtenidos de la primera etapa de implementación indican que el *path* crítico se encuentra en las rondas para el cálculo de los nuevos datos. Esta conclusión es la misma que la reportada en [5], donde el *path* crítico se encuentra en el cálculo del nuevo dato:

$$A = A<<5 + f(B, C, D) + E + W + K_1$$

El operando A representa la retroalimentación de la ronda para tener un *path* no optimizable al realizar un diseño con rondas desenrollados. Los otros operandos están disponibles y su suma puede ser hecha en paralelo con el cálculo de A, es decir, se puede tener una arquitectura parcialmente desenrollada. La ganancia de desenrollar una nueva ronda es evitar el retardo de tres sumas y no de las cuatro sumas de una ronda completa, así como evitar el retardo del cálculo de la función F, ver figura 5.

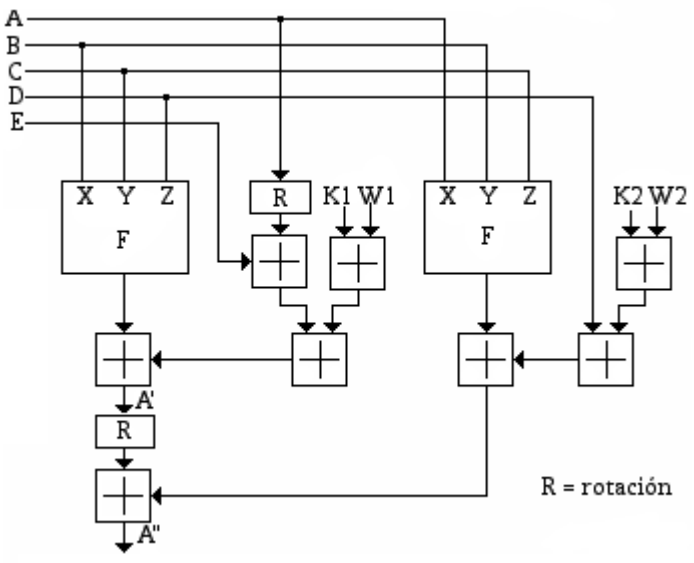

Fig. 5. Estructura de dos rondas desenrolladas

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 de los reportes del Place & Route se pueden ver en la tabla 4.

Diseño	Frecuencia	IOBs	*Slices*	LUTs 4-E	Ciclos	Procesamiento
SHA1vB	43.30MHz	293/324	1039/5120	1629/10240	20	1.109 Gbps

Tabla 4. Resultados de la implementación en la segunda etapa.

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, por lo que 20 ciclos por 4 rondas calculan las 80 rodas establecidas en [1].

La búsqueda de información que se realizó en este artículo muestra que el diseño presentado en este trabajo es la mejor opción de una implementación en FPGA, ver tabla 5.

Diseño	Frecuencia (MHz)	Ciclos	Procesamiento (Gbps)
[4] - FPGA	18.00	81	0.114
[5] - FPGA	38.60	22	0.899
[6] - FPGA	75.75	83	0.462
[7] - FPGA	56.00	81	0.350
[8] - FPGA	79.00	81	0.498
[9] - FPGA	102.00	81	0.644
[10] -FPGA	90.00	153	0.301
[11] - ASIC	166.00	84	1.010
[12] - ASIC	300.00	82	1.896
Este trabajo	43.30	20	1.109

Tabla 5. Resultados de la implementación en la segunda etapa.

La tabla 5 muestra resultados de las implementaciones revisadas en la sección III y del diseño SHA1vB.El resultado de procesamiento de 1.109 Gbps es la mejor opción de una implementación en FPGA, ya que la búsqueda de información de trabajo relacionado no señala implementación alguna que sobrepase el gigabit por segundo. En cambio, las implementaciones en ASIC de [11] y [12] procesan a 1.01 Gbps y 1.896 Gbps, respectivamente trabajando a altas frecuencias de reloj. La implementación presentada en este artículo tiene un procesamiento mayor que la implementación en un ASIC [11] pero a una menor frecuencia de reloj.

Para tener una comparación con los resultados reportados en la síntesis del trabajo en [5], la síntesis del diseño SHA1vB reporta un período 16.943ns para obtener un procesamiento de 1.51 Gbps que es 1.68 veces mejor respecto al procesamiento. Estos dos diseños utilizan una lógica combinacional de cuatro rondas parcialmente desenrolladas.

VI. Conclusiones

Este diseño es para el proceso de bloques de 512 bits, pero agregarle retroalimentación de la salida para procesos de multibloques no debe incluir retardos significativos en el *path* crítico, ya que se registraría la salida inmediatamente y después se seleccionaría adecuadamente la retroalimentación.

La implementación en un FPGA del algoritmo SHA-1 presentado en este artículo ofrece la mejor opción respecto a la capacidad de procesamiento de 1.109Gbps, considerando que la implementación tiene un mayor procesamiento que la implementación revisada en [11] para un ASIC. Este módulo formará parte de la biblioteca de configuraciones de una plataforma reconfigurable criptográfica y la velocidad de procesamiento presentada es una buena solución entre sistemas configurables.

VII. Agradecimientos

Se agradece al CONACyT el apoyo otorgado a través de la Beca para Estudios de Maestría # 171489.

Referencias

[1] Federal Information Processing Standards (FIPS) Publication 180-2, “Announcing the Secure Hash Standard”, US DoC/NIST, Agosto 2002.

[2] Lucena, M. J., “Criptografía y Seguridad en Computadores”, Libro Electrónico, Tercera Edición, Junio 2001, http://wwwdi.ujaen.es/~mlucena/lcripto.html.

[3] Ferguson N., Schneier B., “Practical Cryptography”, Wiley Publishing, Inc., EUA, 2003.

[4] 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, Agosto 2002.

[5] 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, Noviembre 2002.

[6] Tim Grembowski, Roar lien, Kris Gaj, Nghi Nguyen, Peter Bellows, Jaroslav Flidr, Tom Lehman, Brian Schott, “Comparative Analysis of the Hardware Implementations of Hash Functions SHA-1 and SHA-512”, Information Security, 5th International Conference ISC 2002, Sao Paulo, Brasil, Octubre 2002.

[7] Amphion, Hoja de especificaciones, “CS5316 High Performance SHA1/MD5 Hashing Algorithm Core”, 2002, www.amphion-semi.com/acrobat/DS5316.pdf.

[8] CAST, Inc., Hoja de especificaciones, “SHA-1 Processor”, Abril 2002, www.cast-inc.com/sha-1/cast_sha-1.pdf.

[9] Alma Technologies, Hoja de especificaiones, “SHA-1 High Performance Hash Function”, Mayo 2002, www.alma-tech.com/Data-Sheets/SHA-1_pre_sales.pdf.

[10] Hifn Inc., Hoja de especificaciones, “7811 Network Security Processor”, 2002, www.memec-impact.ch/pdf/hifn/7811.pdf.
[11] SecuCore Consulting Services, Hoja de especificaciones, “SecuCore SHA-1/MD5/HMAC Core”, 2001, www.secucore.com/secucore-hmac.pdf.
[12] Ocean Logic Pty Ltd, Hoja de especificaciones, “OL_SHA SHA-1 Processor Rev. 1.1”, www.ocean-logic.com/pub/OL_SHA.pdf.