Implementation of SHA-2 hash function for a digital signature System-on-Chip in FPGA

Conference Paper · January 2009

DOI: 10.1109/ICED.2008.4786681 · Source: IEEE Xplore

3 authors, including:

M. Nazrin
Universiti Teknologi Malaysia
1 PUBLICATION **11** CITATIONS

SEE PROFILE

Yuan Wen Hau
Universiti Teknologi Malaysia
27 PUBLICATIONS **60** CITATIONS

SEE PROFILE

Implementation of SHA-2 Hash Function for a Digital Signature System-on-Chip in FPGA

M.Khalil[1], M.Nazrin[2], Y.W. Hau[3]
VLSI-eCAD Research Laboratory (VeCAD)
Faculty of Electrical Engineering
Universiti Teknologi Malaysia (UTM)
81310 Skudai, Johor, Malaysia.
khalil@fke.utm.my[1] , mr_nazrin@yahoo.com[2], hauyuanwen@yahoo.com[3]

Abstract

The widespread adoption of Internet as a secure medium for communication and e-commerce has made cryptography a vital part of today's information systems. However, to achieve a more pervasive deployment, the supporting cryptographic (crypto) systems should exhibit processing power of high performance and efficiency. These demanding requirements can be achieved by integrating the cryptosystems into designs based on System-on-Chip (SoC). In this paper, the design and implementation of a crypto hash SHA-2 logic core in reconfigurable hardware is presented. We also discuss a public-key crypto SoC, which uses the SHA-2 hash core in conjunction with a 2048-bit RSA co-processor to perform a digital signature security scheme. We use Verilog to model the hardware, and C to code the embedded software. With the crypto SoC implemented in an Altera Nios II Stratix FPGA-based prototyping system running on a 50 MHz system clock, we obtained a throughput of 644 Mbits/sec for our SHA-512 hardware core.

1. INTRODUCTION

A hash function is an important cryptographic (crypto) primitive that is mainly used with public-key cryptosystem in digital signature schemes. A digital signature can directly provide several security services including message authentication, message integrity, and non-repudiation [1]. The hash function is also a basic building block in the implementation of secret-key message authentication codes (e.g. HMAC) [2],[3].

The most widely accepted hash function is Secure Hash Algorithm, SHA-1. However after the introduction of AES (Advanced Encryption Standard), a new symmetric encryption standard, the security of SHA-1 does not any longer match the security level provided by the encryption standard. Three new hash SHA-224, SHA-256, SHA-384, and SHA-512 (referred to as SHA-2), which match the security level of AES, have been announced by the National Institute of Standards and Technology (NIST) [3]. Therefore, new systems should use one of these larger hash functions.

Many studies had been done on the implementation of the SHA-2 hash function, which applied different techniques, applications and technology [4-9]. Recently, FPGA-based systems have become very popular in verification and rapid prototyping of cryptosystems. For example, the work in [4] proposed a single-chip FPGA solution for SHA-384 and SHA-512, while a Xilinix Virtex FPGA implementation of SHA-512 was provided in [5]. Techniques to reduce chip area have been suggested in [6], which performed modulo-64 addition in SHA-512 as modulo-32, modulo-16 and modulo-8 additions. A VLSI implementation of digital signature scheme using 512-bit RSA and SHA-1 was also presented in [1].

In this paper, the design of the hash function SHA-2 in reconfigurable hardware is presented. The design is parameterizable, hence it can be reconfigured to an SHA-384/256/224 design, if needed. A prototype SoC is then built, implementing the SHA-2 logic core in a public-key cryptosystem based on 2048-bit RSA (Rivest-Shamir-Adleman) algorithm. The hash function is used in conjunction with the RSA function in the high-performance public-key cryptosystem to implement a digital signature scheme.

This paper is organized as follows. In Section 2, the digital signature cryptosystem is described, and this is followed by a detailed discussion on the design of the SHA-2 logic core in section 3. Section 4 provides the

978-1-4244-2315-6/08/$25.00 ©2008 IEEE.

experimental work and results, ending with the conclusions in section 5.

2. DIGITAL SIGNATURE CRYPTOSYSTEM

This section discusses briefly, the digital signature scheme applied, and the system architecture of the proposed public-key crypto SoC that implements the SHA-2 hash logic core and the RSA-2048 co-processor to perform the digital signature scheme.

Figure 1 depicts the digital signature scheme. The scheme provides the capability to generate and verify digital signatures. A private key used in the generation of the signature, and the corresponding public key is used to verify that signature. Message, with which these digital signatures are attached, can therefore be authenticated (proved that the message is from the expected sender) at the receiver.

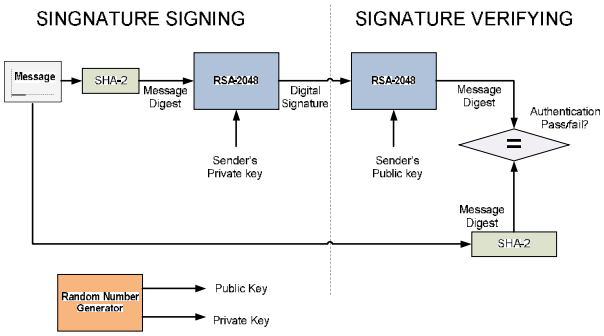

Figure 1. Digital signature scheme

In the signature generation process (left-hand side of diagram in Figure 1), the hash core is used to generate a message digest (a condensed version of the message). This digest goes through the signing process (performed by the RSA co-processor) using the sender private key. This signing process results in the generation of a digital signature, which is sent along with the message to the receiver. At the receiver side, a digest is first created out of the received message, using the same hash function. Using the sender's public key, the verifying process (performed by the RSA) produces a digest from the received signature. If this digest matches the digest produced out of the received message, then the signature is authentic, and the message is accepted; otherwise it is rejected.

Figure 2 provides a block diagram of the proposed public-key crypto SoC implementing the RSA-based digital signature scheme. The hardware part of the architecture consists of the Altera Nios II embedded processor (acting as control CPU), and three IP cores that perform hybrid random number generation (RNG), 2048-bit RSA computation, and SHA-2 hash function. Embedded software (or firmware), coded in C, provides the device driver routines and APIs.

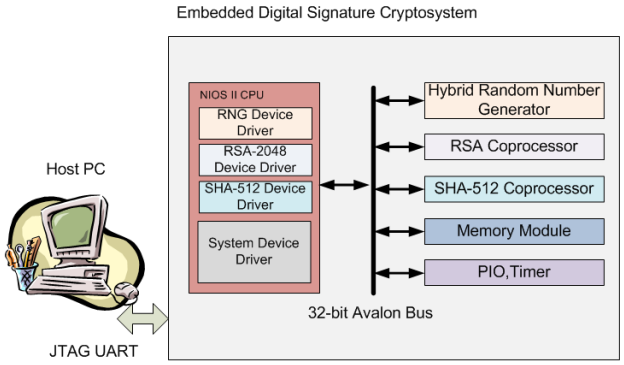

Figure 2. Block diagram of proposed crypto SoC

The hybrid RNG module provides a random number to the key generation unit to generate, on-chip, the public and private keys. The underlying function of the RSA co-processor is a compute-intensive long integer modular exponentiation operation. As this paper focuses on the design of the SHA-512 core, the designs of the RNG module and the RSA crypto accelerator is not provided here. Their detailed designs can be obtained in [11] and [12] respectively.

Hierarchical modular RTL design methodology is employed in the design of these crypto processors. In this work, hardware is modeled in Verilog, and the firmware is coded in C. The hardware cores are fully synthesized and mapped into FPGA. The Altera Nios II Stratix FPGA prototyping system is the implementation platform of the target SoC.

3. DESIGN OF SHA-2 CORE

In this section, the design and implementation of the SHA-2 logic core is discussed in detail.

SHA-2 inputs messages with the maximum length up 2^{128}-1 bits or 2^{64}-1 for 512 bits or 1024 bits block size respectively. Internally, it processes these blocks based on 32-bit words (word size for SHA-224/256) or 64-bit words for SHA-384/512. At the output, it produces a 512-bit hash value of message digest for SHA-512, 384-bit for SHA-384, 256-bit for SHA-256 and 224-bit for SHA-224.

The architecture of our hash logic core is shown in Figure 3. The preprocessing unit passes 512 or 1024-bit message blocks to the message scheduler. The iteration processing unit performs the actual hashing.

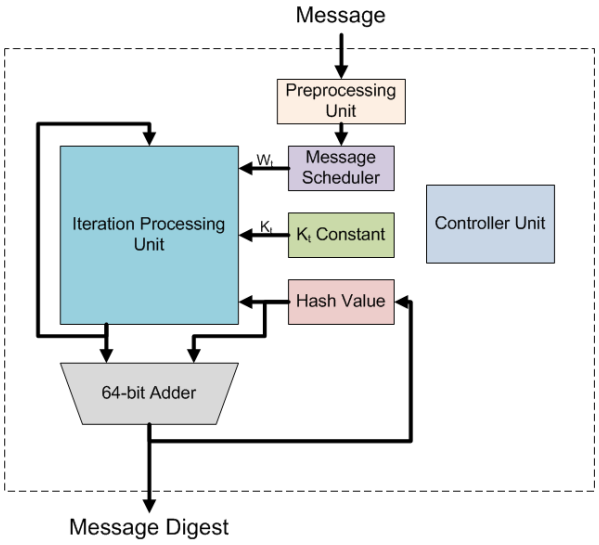

Figure 3. Architecture of SHA-2

The computation of the message digest is performed through two stages: preprocessing and hashing as illustrated in Figure 4.

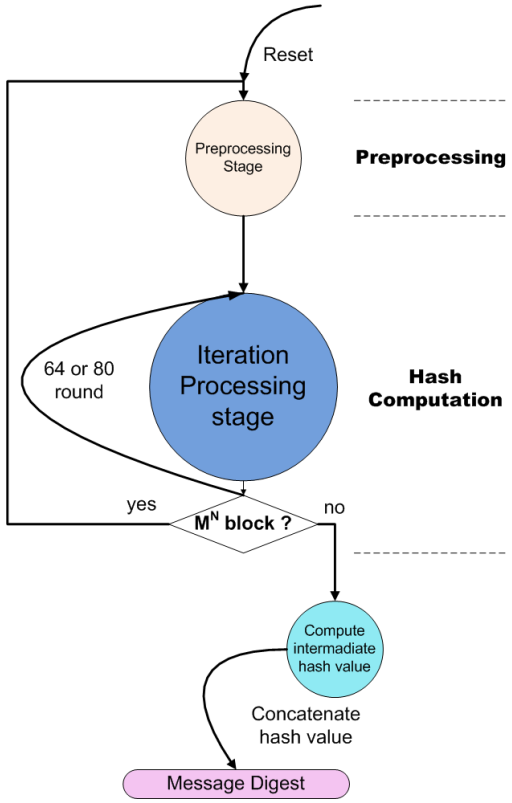

Figure 4. State diagram of SHA-2

In the preprocessing stage, the message is padded, parsed into N 512-bit or N 1024-bits blocks, $M^1, M^2, \ldots, M^N$. We performed this preprocessing function using embedded software (firmware) executed by the Nios II embedded processor. In the hashing operation, each message block $M^1, M^2, \ldots, M^N$, is processed in order. For a message block M^i, processing is performed on 64-bit words M^i_t, and there are 16 such words in a block. For each message block i (in the range 1 to N), after initial hash values are assigned, a digest round is performed to generate the hash values H_0^i to H_7^i.

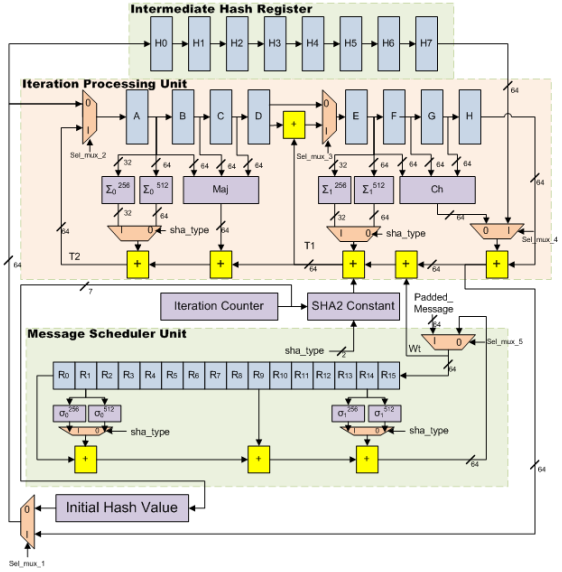

Figure 5. Functional block diagram of SHA-2

We now present the SHA-2 algorithm implemented in this work, and the corresponding functional block diagram of the complete SHA-2 core is shown in Figure 5 above.

Step 1: The hash values, H_0^{i-1} to H_7^{i-1}, are assigned to variable A, B, C, D, E, F, G and H, which can be realized with eight serial registers of 64-bit widths. The eight initial hash values are depicted in Table 1 [3].

Table 1. Initial hash value of SHA-2

	SHA-224	SHA-256	SHA-384	SHA-512
$H^0_0 \rightarrow A$	c1059ed8	6a09e667	cbbb9d5dc1059ed8	6a09e667f3bcc908
$H^0_1 \rightarrow B$	367cd507	bb67ae85	629a292a367cd507	bb67ae8584caa73b
$H^0_2 \rightarrow C$	3070dd17	3c6ef372	9159015a3070dd17	3c6ef372fe94f82b
$H^0_3 \rightarrow D$	f70e5939	a54ff53a	152fecd8f70e5939	a54ff53a5f1d36f1
$H^0_4 \rightarrow E$	ffc00b31	510e527f	67332667ffc00b31	510e527fade682d1
$H^0_5 \rightarrow F$	68581511	9b05688c	8eb44a8768581511	9b05688c2b3e6c1f
$H^0_6 \rightarrow G$	64f98fa7	1f83d9ab	db0c2e0d64f98fa7	1f83d9abfb41bd6b
$H^0_7 \rightarrow H$	befa4fa4	5be0cd19	47b5481dbefa4fa4	5be0cd19137e2179

Step2: This step is performed in the Message Scheduler, and the operation executed is as follows:
SHA-384/512:

$$W_t = \begin{cases} M_t^{(i)} & 0 \le t \le 15 \\ \sigma_1^{\{512\}}(W_{t-2}) + W_{t-7} + \sigma_0^{\{512\}}(W_{t-15}) + W_{t-16} & 16 \le t \le 79 \end{cases}$$

where,

$$\sigma_0^{\{512\}}(x) = ROTR^1(x) \oplus ROTR^8(x) \oplus SHR^7(x)$$
$$\sigma_1^{\{512\}}(x) = ROTR^{19}(x) \oplus ROTR^{61}(x) \oplus SHR^6(x)$$

SHA-224/256:

$$W_t = \begin{cases} M_t^{(i)} & 0 \le t \le 15 \\ \sigma_1^{\{256\}}(W_{t-2}) + W_{t-7} + \sigma_0^{\{256\}}(W_{t-15}) + W_{t-16} & 16 \le t \le 63 \end{cases}$$

where,

$$\sigma_0^{\{256\}}(x) = ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x)$$
$$\sigma_1^{\{256\}}(x) = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x)$$

$ROTR^n(x)$ =circular right shift operation of a variable x by n positions to the right, and
$SHR^n(x)$ = shifting of a variable x by n positions to the right.

Step3: This step is performed by the iteration processing unit, and the function is characterized by the equations below:

$$Ch(x,y,z) = (x \wedge y) \oplus (\neg x \wedge z)$$
$$Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$$

SHA-384/512:

$$\sum_0^{\{512\}}(x) = ROTR^{28}(x) \oplus ROTR^{34}(x) \oplus ROTR^{39}(x)$$
$$\sum_1^{\{512\}}(x) = ROTR^{14}(x) \oplus ROTR^{18}(x) \oplus ROTR^{41}(x)$$

SHA-224/256:

$$\sum_0^{\{256\}}(x) = ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x)$$
$$\sum_1^{\{256\}}(x) = ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x)$$

$$T_1 = h + \Sigma_1(e) + \mathrm{Ch}(e,f,g) + K_t + W_t$$
$$T_2 = \Sigma_0(a) + \mathrm{Maj}(a,b,c)$$
$$h = g$$
$$g = f$$
$$f = e$$
$$e = d + T_1$$
$$d = c$$
$$c = b$$
$$b = a$$
$$a = T_1 + T_2$$

where, Kt represents a sequence of 64 32-bit or 80 64-bit word constants for SHA-256 or SHA-512 respectively.

Step 4: In this step, the i[th] intermediate hash values, that is, H_0^i to H_7^i, are computed by modulo-64 bit adders as follows:

$$H_0^{(i)} = a + H_0^{(i-1)} \quad H_4^{(i)} = e + H_4^{(i-1)}$$
$$H_1^{(i)} = b + H_1^{(i-1)} \quad H_5^{(i)} = f + H_5^{(i-1)}$$
$$H_2^{(i)} = c + H_2^{(i-1)} \quad H_6^{(i)} = g + H_6^{(i-1)}$$
$$H_3^{(i)} = d + H_3^{(i-1)} \quad H_7^{(i)} = h + H_7^{(i-1)}$$

Let us now discuss the hardware implementation of each module in the SHA-2, in turn. As shown in Figure 5, the message scheduler unit is made up of 16 64-bit shift registers, 3 modulo-64 adders and a multiplexer. In the first 16 clock cycles, a complete message block enters into the shift register. The multiplexer then selects the feedback input to provide the shift register with data from feedback line, for the next 64 or 80 clock cycles. Wt is the output signal of this module, which is sent to the iteration processing unit.

Referring to the functional block diagram in Figure 5, the iteration processing unit uses four logical functions, Ch and Maj, $\sum_0$ and $\sum_1$. A to H registers are 64-bit wide registers, and the adders are modulo-64 units. The 64-bit values of the A to H registers are updated in each round and the new values are used in the following rounds. The initial values of these variables are given by the 512-bit constant value specified in Initial Hash Value module, this value is only set for the first data block. The consecutive data blocks use the partial digest message, computed for the previous data block.

The iteration processing unit produces a final digest message of 512 bits, which is dependent of the input message, composed by multiple blocks of 512 bits or 1024 bits each. This input block is expanded and fed to the 64 cycles (SHA-224/256) and 80 cycles (SHA-384/512) of the SHA2 functions in words of 64 bits each, denoted by Wt. In each cycle or round of the SHA-512 algorithm the introduced data is mixed with the current state. This data scrambling is performed by additions and logical operations, such as bitwise logical operations and bitwise rotations. The value Wt is the 64-bit data word, for the t-round, and the Kt value represents the 64-bit constant that also depends on the round.

Each 512 data block (SHA-224/256) or 1024 data block (SHA-384/512) is processed for 64 or 80 rounds respectively, after which the values of the register A to H are added to the previous digest message, in order to

obtain partial digest message. When the final message block has been processed, the hash value outputs are concatenated to produce the 512-bit message digest. The message digest is computed by $H_0^N || H_1^N || H_2^N || H_3^N || H_4^N || H_5^N || H_6^N || H_7^N$ after preprocessing all N blocks in the message.

4. EXPERIMENTAL RESULTS

The digital signature crypto SoC is prototyped in a single Altera Stratix EP1S40F780C5 FPGA chip, using Altera SOPC Builder and Nios II IDE. Each module in the SoC is synthesized, simulated, place and route, and timing analyzed using Altera Quartus II Version 7.1 EDA software. Test data are obtained from [10]. Design verification is performed via timing simulations, and the complete system is validated through the evaluation tests performed on the SoC hardware prototype. Figure 6 shows a snapshot of the timing simulation output that verifies the functionality of the SHA-512 logic core designed in this work.

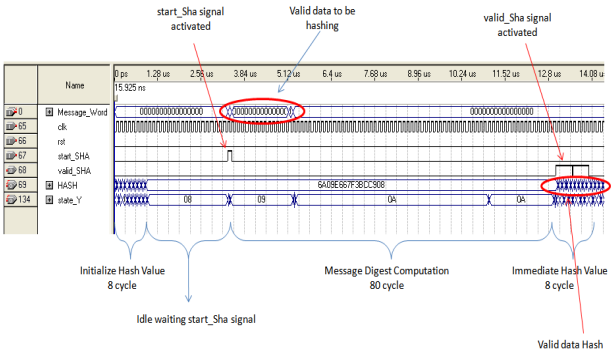

Figure 6. Timing simulation of SHA-512 core

In additions, functionality test in host PC also conducted to test the connection and data transfer between SHA-2 on hardware and API in host PC. Figure 7 shows the result for message digest of SHA-2. The *API_Hashing()* function is responsible for this operation in *UTMmyCryptoAPI* module.

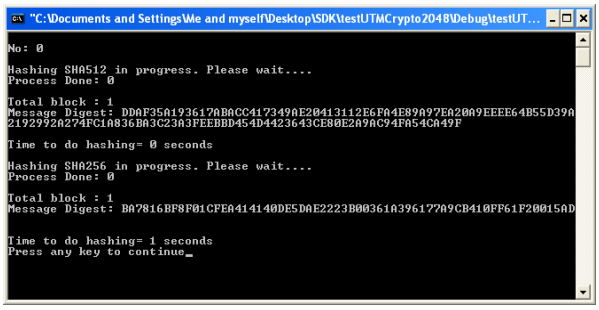

Figure 7. SHA-2 operation with *UTMmyCryptoAPI*

The performances of the various designed in this work are presented in Table 2. The no 1 gives the higher throughput but slightly lower on operation frequency compared with no 3. Our SHA-512 design is then compared with other designs reported in [6], [7] and in [13]. Overall our design has a higher throughput and comparable area consumption.

Table 2. Comparison with other designs

No	Designer	Design	Device	Area (LE)	Percentage of chip area used (%)	fmax(Mhz)	Throughput (Mbits/s)
1	M.Nazrin	SHA 384/512	Stratix EP1S40	4489	11%	50 MHz	644
		SHA 512		4408	11%	49 MHz	633
2	C.L Ping[13]	SHA 384/512		5905	14%	34 MHz	430
3	M.Nazrin	SHA 224/256	Stratix EP1S40	2245	5%	57 MHz	464
4		SHA 224/256/384/512		4957	12%	50 MHz	634
6	I. Ahmad[6]	SHA512(64)	Stratix EP1S10	4229	40%	48 MHz	613
		SHA512(32)		2800	26%	47 MHz	592
7	McLoone M[7]	SHA 384/512	Xilinx XCV600E-8	2914 CLB		38 MHz	479

Table 3 shows the time execution in the hashing a 7Mbyte sample file through serial communication from host PC to device. According to the table 3, shows that SHA-512 takes almost 45% shortest time compared to SHA-256 to compete the final message digest byte.

Table 3: Time execution of hashing on 7MByte data via serial communication, between PC host to the device

Hash Function	Time taken
SHA-224	256 sec
SHA-256	275 sec
SHA-384	160 sec
SHA-512	151 sec

5. CONCLUSION

In this paper, we presented the design and implementation of an embedded digital signature cryptosystem based on SoC technology. The design is designed to be parameterizeable permitting easy modifications to be made on data widths and register sizes. The crypto SoC is prototyped into a single Altera Stratix FPGA microchip. The experimental results show that the design is a promising, giving a throughput of 644 Mbits/sec. At the same time, this crypto embedded system provides a suitable compromise between the constraint of speed, area and required security level based on the specific demands of targeted applications.

ACKNOWLEDGEMENT

The authors would like to acknowledge the financial support provided for this work by MOSTI Technofund grant, UTM Vot No. 79900.

REFERENCES

[1] P.Kitsos, N.Sklavos and O.Koufopavlou, "An Efficient Implementation of the Digital Signature Algorithm*", 9th Int. Conf. on Electronic, Circuits and Systems*, 2002.

[2] Alfred J.Menezes, Paul C.van Oorschot, Scott A.Vanstone, *Handbook of Applied Cryptography*, *CRC Press*, 1997.

[3] NIST Federal Information Processing Standards (FIPS) Publication,FIPS PUB 180-2, 2002.

[4] Imtiaz Ahmad and A. Shoba Das, " Analysis and Detection Of Errors In Implementation Of SHA-512 Algorithms On FPGAs", *The Computer Journal 2007* 50(6):728-738.

[5] T. Grembowski, R. Lien, K. Gaj, N. Nguyen, P. Bellows and J. Flidr et al., "Comparative analysis of the hardware implementation of hash functions SHA-1 and SHA-512" , *Proceedings of the 5th Int. Conf. on information security (ISC'2002)*, LNCS 2433, Springer-Verlag (2002), pp. 75–89.

[6] Imtiaz Ahmad and A. Shoba Das, "Hardware implementation analysis of SHA-256 and SHA-512 algorithm on FPGAs" , *Computers & Electrical Engineering*, Volume 31, Issue 6, September 2005, Pages 345-360.

[7] McLoone M, McCanny JV. "Efficient single-chip implementation of SHA-384 and SHA-512", *In: Proceedings of the IEEE Int. Conf. on field-programmable technology(FPT)*, Hong Kong , July 2002.p.311-4.

[8] Seyyed Ali Emam and Sareh Sadat Emami, "Design and implementation of a Fast, Combined SHA-512 on FPGA", *IJCSNS Int. Journal of Computer Science and Network Security*, Vol.7 No.5, May 2007.

[9] Ryan Glabb, L. Imbert, G. Jullien, A. Tisserand and N. Veyrat-Charvillon, "Multi-mode operator for SHA-2 hash functions", *Journal of Systems Architecture 53* (2007) p. 127-138

[10] National Institute of Standards and Technology, NIST Special Publication 800-21, December 2005.

[11] Arul Paniandi, "A hardware implementation of Rivest-Shamir-Adleman Co-processor for resource constrained embedded systems", Master Thesis, Universiti Teknologi Malaysia, May 2006.

[12] Norashikin M. Thamrin, "A Field Programmable Gate Array-Based Cryptographic System-On-Chip" ", Master Thesis, Universiti Teknologi Malaysia. Dec 2007.

[13] Chua Li Ping "RTL Design of SHA-512 Processor Core for Secure Message Hashing", Master Thesis, Universiti Teknologi Malaysia, May 2008.