Implementation of Reconfigurable SHA-2 Hardware Core

Todor Mladenov & Saeid Nooshabadi
Department of Information and Telecommunications
Gwangju Institute of Science and Technology
Gwangju, Republic of Korea
{todor,saeid}@gist.ac.kr

***Abstract*—Secure Hashing Algorithm (SHA) is increasingly becoming popular for the online security applications, specifically, for mobile and embedded system platforms. This necessitates a high performance hardware implementation of the four most utilized SHA algorithms (SHA–224, SHA–256, SHA-384, SHA-512). This paper presents a reconfigurable SHA-2 IP core utilizing the available hardware for computing SHA-384/ SHA-512 and SHA-224/ SHA-256 with double the throughput.**

I. INTRODUCTION

The SHA-2 (secure hash algorithm) [1]–[10], is a unifying name for the four hashing algorithms SHA-224, SHA-256, SHA-384 and SHA-512 introduced by the US National Institute of Science and Technology (NIST) [1]. The number in the name corresponds to the message digest size (in bits), which the particular hash method outputs.

SHA-256 and SHA-512 are novel hash functions computed with message processed in units 32 and 64-bit words, respectively. Based on the virtually identical structures, they employ different shift amounts, additive constants and number of rounds to compute the message digests. SHA-224 and SHA-384 are special versions of the SHA-256 and SHA-512, respectively, where message digests are truncated to 224 and 384. They are also computed with different initial values.

Some of the applications of SHA include generating and verifying digital signatures, generating and verifying message authentication codes, data integrity protection and increasing the entropy in pseudo random number generators.

This paper presents a novel reconfigurable hardware core able to compute all four hashing functions with superior throughput, in an efficient manner. To the best of our knowledge the four hash algorithms haven't been integrated in a single IP. In particular, in our design the SHA-512 resources are used to perform two SHA-256 hash functions in parallel thus doubling the throughput with minor increase in the hardware resources. The proposed design also performs better than the designs in [11]–[13] as it avoids use of memory involving slow access. Further unlike the designs in [11]–[14] we have also implemented the preprocessor unit. The design was tested and analyzed on Altera field programmable gate array (FPGA) device; Stratix EP1S10F484C5.

The paper is organized as follows: Section II presents the SHA-2 algorithms. Section III explains the proposed architecture. Section IV reports the implementation results, while Section V concludes the paper.

II. SHA-2 ALGORITHMS

Table I shows the main characteristics of the four algorithms: output message digest size, data block size on which the hashing is performed, the maximum data length that can be hashed and the number of computation cycles per block of data.

Since SHA-224 and SHA-384 are truncated versions of SHA-256 and SHA-512 with only different initial values, for simplicity, in this section, only the latter two will be discussed in details. Message digest generation of SHA-256 and SHA-512 are identical and will be presented simultaneously.

The hashing procedure consists of two main parts: preprocessing and hash computation.

TABLE I. SHA-2 CHARACTERISTICS

Algorithm	Output size (bits)	Internal state size (bits)	Block Size (bits)	Max message size (bits)	Word size (bits)	Rounds
SHA-256/224	256/224	256	512	2^{64}-1	32	64
SHA-512/384	512/384	512	1024	2^{128}-1	64	80

978-1-4244-2342-2/08/$25.00 ©2008 IEEE.

A. Preprocessing

Preprocessor first pads the message to make it a multiple of 512 (SHA-256) and 1024 bits (SHA-512), and then divides the message into N blocks of size 512 and 1024 bits, where the last block should contain information about the length of the message.

B. Hash computation

Each of the N preprocessed blocks contains 16 message words (M_t) of 32 (SHA-256) and 64 bits (SHA-512) each. Each block of data is processed as follows for $i = 1$ to N in the following three steps:

Step1. Loading of the eight working registers *a, b, c, d, e, f, g* and *h* with the initial hash value for the first iteration or the previous iteration's hash value $H^{(i-1)}$ for every consequent one:

$$a = H_0^{(i-1)}$$

$$b = H_1^{(i-1)}$$

$$c = H_2^{(i-1)}$$

$$d = H_3^{(i-1)}$$

$$e = H_4^{(i-1)}$$

$$f = H_5^{(i-1)}$$

$$g = H_6^{(i-1)}$$

$$h = H_7^{(i-1)}$$

Step2. For cycle index $t = 0$ to 63 (SHA-256) and $t = 0$ to 79 (SHA-512) the following computations are performed:

$$h = g$$

$$g = f$$

$$f = e$$

$$e = d + T_1$$

$$d = c$$

$$c = b$$

$$b = a$$

$$a = T_1 + T_2$$

The T_1 and T_2 functions involved in the computation of work registers *a* and *e* are:

$$T_1 = h + \sum_1^{SHA}(e) + Ch(e,f,g) + K_t + W_t \quad (1)$$

$$T_2 = \sum_0^{SHA}(a) + Maj(a,b,c) \quad (2)$$

$$Ch(x,y,z) = (x \wedge y) \oplus (\neg x \wedge z) \quad (3)$$

$$Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \quad (4)$$

$$\sum_0^{\{512\}}(x) = ROTR^{28}(x) \oplus ROTR^{34}(x) \oplus ROTR^{39}(x) \quad (5)$$

$$\sum_1^{\{512\}}(x) = ROTR^{14}(x) \oplus ROTR^{18}(x) \oplus ROTR^{41}(x) \quad (6)$$

$$\sum_0^{\{256\}}(x) = ROTR^{2}(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x) \quad (7)$$

$$\sum_1^{\{256\}}(x) = ROTR^{6}(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x) \quad (8)$$

The superscript indexes {256} and {512} in the functions refers to SHA-256 and SHA-512, respectively. Rotation and shift to the right are abbreviated as "ROTR" and "SHR", respectively. The differences in the algorithms for SHA-256 and SHA-512 are size of the working registers and the amount of shift and rotation.

The value K_t in equ. (1) is a 32 and 64-bit constant in case of SHA-256 SHA-512, respectively. The value W_t is formed according to equs. (9) and (10).

$$W_t = \begin{cases} M_t^{(i)} & 0 \le t \le 15 \\ \sigma_1^{256}(W_{t-2}) + W_{t-7} + \sigma_0^{256}(W_{t-15}) + W_{t-16} & 16 \le t \le 63 \end{cases} \quad (9)$$

$$W_t = \begin{cases} M_t^{(i)} & 0 \le t \le 15 \\ \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16} & 16 \le t \le 80 \end{cases} \quad (10)$$

M_t quantities are the 16 input message words in every preprocessed block. The W_t functions for t larger than 15 are computed using following functions:

$$\sigma_0^{\{512\}}(x) = ROTR^{1}(x) \oplus ROTR^{8}(x) \oplus SHR^{7}(x) \quad (10)$$

$$\sigma_1^{\{512\}}(x) = ROTR^{19}(x) \oplus ROTR^{61}(x) \oplus SHR^{6}(x) \quad (11)$$

$$\sigma_0^{\{256\}}(x) = ROTR^{7}(x) \oplus ROTR^{18}(x) \oplus SHR^{3}(x) \quad (12)$$

$$\sigma_1^{\{256\}}(x) = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x) \quad (13)$$

Step3. Computing the hash value for the current message block:

$$H_0^{(i)} = a + H_0^{(i-1)}$$

$$H_1^{(i)} = b + H_1^{(i-1)}$$

$$H_2^{(i)} = c + H_2^{(i-1)}$$

$$H_3^{(i)} = d + H_3^{(i-1)}$$

$$H_4^{(i)} = e + H_4^{(i-1)}$$

$$H_5^{(i)} = f + H_5^{(i-1)}$$

$$H_6^{(i)} = g + H_6^{(i-1)}$$

$$H_7^{(i)} = h + H_7^{(i-1)}$$

After repeating these three steps N times, the message digest for input data is computed.

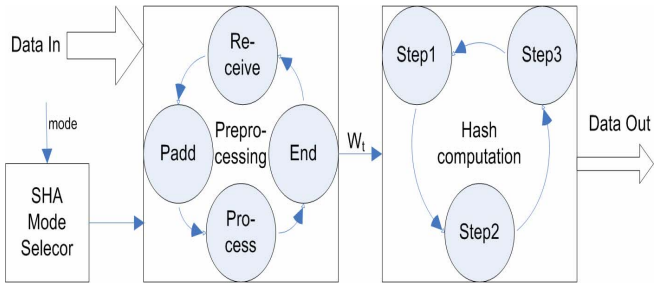

Fig.1 SHA-2 Block Diagram

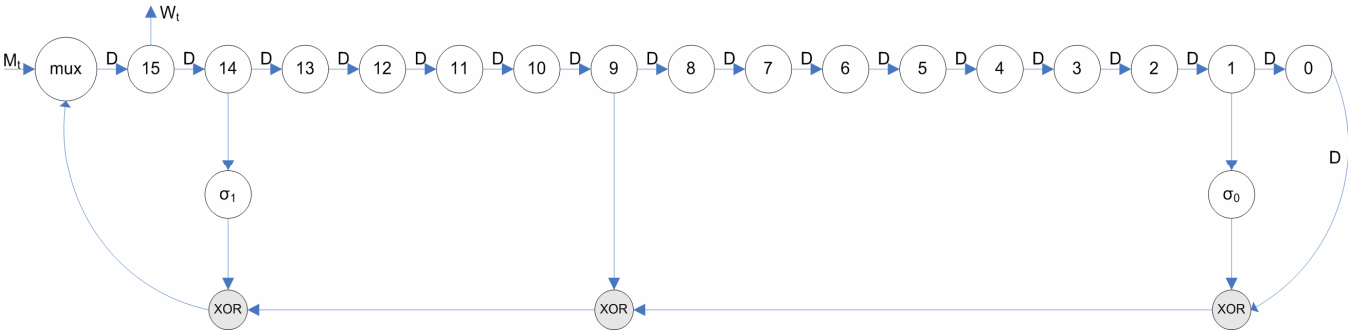

Fig.2 Directed Flow Graph for W_t generation in Preprocessing State Machine

III. HARDWARE ARCHITECTURE

The sequential nature and inter iteration dependencies of SHA hashing algorithm, does not allow for increase in the throughput through parallelization. We, however, improve the throughput by a factor of two, through an architecture that is reconfigurable. In the proposed architecture a SHA-512 is reconfigured to perform two SAH-256 hash functions in parallel.

The SHA architecture, shown in Fig.1, is based on two state machines for the preprocessor and the hash computation, and a logic block that selects the mode of operation. When SHA is implemented in software, usually the W_t values, in equ. (7), are pre-computed once for all t iterations within a block of data. Fig.2 shows a directed flow graph (DFG) that computes one t iteration in one clock. In the proposed DFG all W_t values are computer in parallel. The difference of the DFG for SHA-256 and SHA-512 is in the size of the registers (32 and 64 bits) and the σ functions.

The "SHA Mode Selector block" on Fig.2 selects the core for computing one of the four algorithms. It also disables the unused logic in order to reduce the power consumption and controls the double message processing for SHA-256, if required.

Step1 and Step3 of the hash computation can be combined to reduce the number of clock cycles to process a block by one to 65 cycles, provided that working registers are initialized using an asynchronous reset at the beginning of the hashing of every new message.

In order to improve the performance of our design, we have avoided use of memory to store the K_t constants and the initial hash constant in memory. We employ the unused registers in the logic elements to store these constants.

IV. MPLEMENTATION RESULTS

To compare our work with the best implementation to date in [11], we have synthesized our design in the same platform as in [11]; Altera Stratix EP1S10F484C5. The implementation results for SHA-512 are presented in Table II. As seen our design does not require any memory. We achieve a throughput of 1024/81 × 74Mhz = 936Mbits/s. The throughput performance result of the proposed design is 1.55 times better than the best reported design [11] on the same platform, with considerable saving in memory, and marginal increase in the number of logic elements.

TABLE II. SHA512 PERFORMANCE RESULTS FOR

Design	Area (LEs)	Percentage of chip area used (%)	Memory (bits)	Clock (MHz) (f)	Throughput (Mbits/s)
SHA-$2_{(512\text{-}64bit)}$	4365	41	0	74	936
Design in [11]	4229	40	9216	48	613
Design in [13]	-	-	5120	38	479

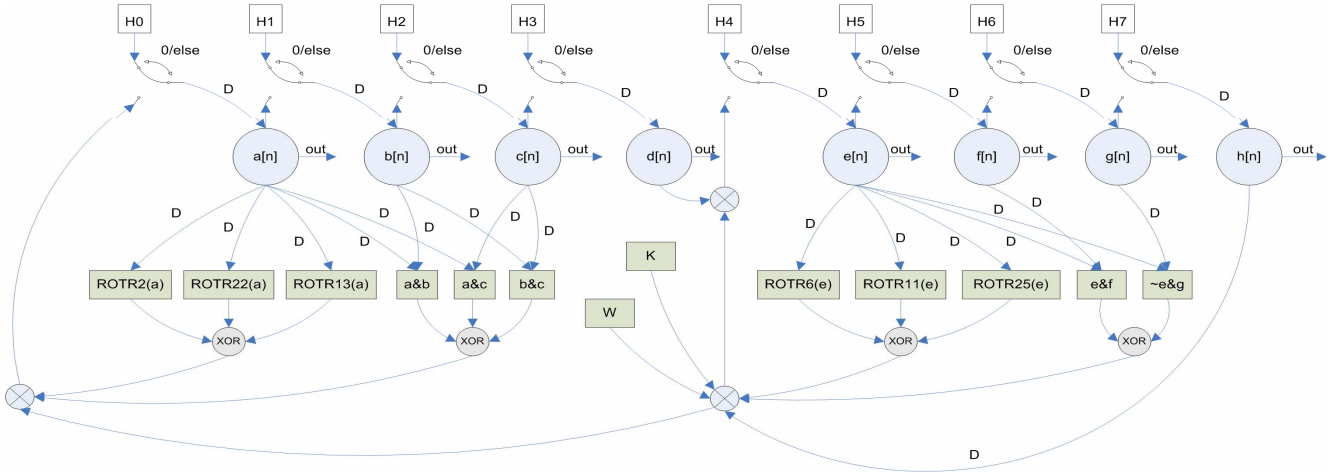

Fig.3 Directed Flow Graph for SHA-256 Computation

TABLE III. SHA256 PERFORMANCE RESULTS.

Design	Clock (MHz)	Equiv. Gate	Throughput (Mbits/s)
SHA-$2_{(256)}$	74	104760	595
SHA-$2_{(256\text{-double})}$	74	–	1189
Design in [12]	88	167190	87

Unlike the designs in [2] – [4], we have also preprocessor unit. It requires 4026 logic elements. Without the inclusion of the preprocessor unit the clock frequency for our design increases to 88 with a throughput rate of 1075.

The implementation results for SHA-256 are presented in Table III. Our performance results are in comparison with the only published work in [12] on the hardware implementation of SHA-256. Since work in [12] is on a different platform (Xilinx Virtex-XCV300E-8 FPGA) we have used the equivalent gate count and the throughput for comparison of two designs. The throughput performance result of the proposed design is 6.8 times better than the best reported design in [12].

V. CONCLUSIONS

This paper proposed a high performance implementation of SHA256/SHA512/SHA224/SHA384 as a single reconfigurable IP core, without any hardware duplication. The proposed core was synthesized and compared with most of the reported in the literature previous design.

The proposed design also avoids use of memory storage improving performance substantially. The design also includes the pre-processor unit. The throughput performance result of the proposed design is 1.55 times better than the best reported design [11] on the same platform.

REFERENCES

[1] US NIST, Secure Hash Standard, Draft FIPS PUB 180-2, August 2002.

[2] C. Kaufman, R. Perlman and M. Speciner, Network security: private communication in a public world (2nd ed.), Prentice-Hall (2002).

[3] FIPS Publication 198. The Keyed-hash message authentication code (HMAC). US Doc/NIST, March 6, 2002.

[4] Rivest RL. The MD5 message digest algorithm. RFC 1321, April 1992.

[5] FIPS Publication 180-1. Secure hash standard (SHS). US Doc/NIST, April 17, 1995.

[6] FIPS Publication 197. Advanced encryption standard (AES). US Doc/NIST, November 26, 2001.

[7] FIPS Publication 180-2. Secure hash standard (SHS). US Doc/NIST, May 30, 2001.

[8] FIPS Publication 180-2. Secure hash standard (SHS) change notice 1. US Doc/NIST, February 2004.

[9] A. Booselaers, R. Govaerts and J. Vandewalle, Fast hashing on Pentium, Proceedings of Crypto'96, LNCS 1109, Springer-Verlag (1996), pp. 298–312

[10] A. Booselaers, R. Govaerts and J. Vandewalle, SHA: a design for parallel architectures?, Proceedings of the EUROCRYPT'97, LNCS 1233, Springer-Verlag (1997), pp. 348–362.

[11] A. Imtiaz and A. Shoda Das, "Hardware implementation analysis of SHA-256 and SHA-512 algorithms on FPGA," Computers and Electronic Engineering, vol. 31, pp. 345-360, October 2005.

[12] K. Ting, S. Yeun, K. Lee, P. Leong, "An FPGA based SHA-256 Processor", 12th International Conference on Field Programmable Logic and Applications, September 2002.

[13] M. McLoone, J. McCanny, "Efficiant Single-chip Implementation of SHA-384 & SHA-512",.Proceedings of the IEEE international Conference on Field-Programmable Technology (FPT), July 2002, Hong Kong , p.311-314