Design Of Superscalar SHA-1 & SHA-2 MIPS Processor Using FPGA

Laith F. Jumma
Asst.Prof.
Safaa S. Omran
Computer Engineering Techniques College of Electrical and Electronic Techniques/Iraq

Abstract

According to the wide developments in the area of communications, there is a demand for secure system for data transmissions. Hash function has important usage in the cryptography for information security. Cryptographic hash functions are used to protect information integrity and authenticity in a wide range of applications. In this paper, a Hash system SHA-1 and SHA-2 Processor is designed using Xilinx Spartan-3AN. The implementation of the processor is done by using Superscalar MIPS Processor (Microprocessor without Interlocked Pipelines) single cycle by choosing a certain number of instructions that are necessary to invoke the SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512 algorithms.

Keywords—VHDL; SHA-1; SHA-2; MIPS Processor; Superscalar.

1.INTRODUCTION

Cryptographic hash functions are one of the most widely used cryptographic primitives for security applications. They have been extensively deployed over the last few decades, especially after digital signatures became more and more popular. Moreover, many other "ancillary" applications, including hash-based message authentication codes, pseudo random number generators and key derivation functions, make use of cryptographic hash functions as their main underlying primitive.

A variation on the message authentication code is the one-way hash function.

A hash function accepts a variable-size message *M* as input and produces a fixed-size output, referred to as a hash code *H(M)*. Hash functions are used in conjunction with symmetric ciphers for digital signatures. In addition, hash functions are used for message authentication [7].

A Superscalar Hash MIPS processor is designed to contain only the instructions and blocks that required in invoking the

algorithms of SHA-1, SHA224, SHA-256, SHA-384 and SHA-512, so it does not take large number of transistors to implement comparing with the complete design of MIPS Processor.

VHDL (Very High Speed Integrated Circuit Hardware Description Language) is a hardware description language. It describes the behaviour of an electronic circuit or system, from which the physical circuit or system can then be attained (implanted)[6].

The rest of this paper is structured as follows. Section II gives a brief explanation for related work. Section III describes the different Hash

algorithms. Section IV gives description for the MIPS processor and section V is our complete superscalar processor design. Results and Conclusions are drawn in Sections VI and VII.

2. Related Work

One of the earliest hardware design of hash function is reported by selimis in [5]. The authors in [2] propose a processor for both SHA-1 and

MD5 algorithms, and implemented on an Altera Apex 20k. While in [1] a high-performance SHA-1 design in presented and implemented on a Xilinx Vertex-E FPGA.

The authors in [3] implemented a Hash function on embedded-system. In [4] a processor is designed to support the algorithm of SHA-1 and SHA-256 hashing. However, in this paper SHA-1 SHA-224, SHA-256, SHA-384, and SHA-512 hash MIPS processor is designed and implemented using Xilinx-Spartan-3AN.

3. SHA-1 & SHA-2 Algorithms

The SHA-2 comprises four algorithms, namely, SHA-224, SHA-256, SHA-384, and SHA-

512. SHA-224 and SHA-256 have several commonalities. Likewise, SHA-384 and SHA-512 have also several commonalities. These five algorithms are one-way hash functions that process a message to produce a message digest. Each algorithm can be classified into two stages. The first stage is the Pre-processing stage and the second is Hash Computation.

A. Pre-processing

SHA-1 and SHA-2 input block size various depending on what algorithm is used. In this stage the message is padded into block size called chunk. The input blocks size of SHA-1, SHA-224, SHA-256 is equal to 512-bits divider into 16 words of 32-bit W[0-15], While SHA-384 and SHA-512 is equal to 1024-bits divider into 16 words of 64-bit W[0-15].Then extending the W[0-15] depends on the used algorithm. Table.1 shows comparison of hash functions.

Tabl.1 Hash Function Comparison

Algorithms	Rounds & W	Block Size (bits)	Output Size (bits)	Max Message Size (bits)
SHA-1	80	512	160	2^64 - 1
SHA-224	64	512	224	2^64 – 1
SHA-256	64	512	256	2^64 – 1
SHA-384	80	1024	384	2^128 - 1
SHA-512	80	1024	512	2^128 - 1

B. Hash Computation

Hash Computation various depending on what algorithm is used so it can be classified as following.

- SHA-1 Computation

The message digest is 5 hash variables each of 32 bits are used. It takes 80 rounds to complete SHA-1 calculation. SHA-1 algorithm is done as follows:

In the first step hash variables shown in table.2 are initialized.

TABLE.2 INITIALIZED SHA-1 VARIABLES

H0	H1	H2	H3	H4
67452301	EFCDAB89	98BADCFE	10325476	C3D2E1F0

The second step message word W[0-15] were extended into 80 word W[16-79]

$$Wt = (W[t-3] \oplus W[t-8] \oplus W[t-14] \oplus W[t-16]) <<< 1$$

$$16 \leq t \leq 79$$

Where $\oplus$ is XOR logic and <<< is left rotate .

In the third step 5 variables (*A to E*) were added which they equal to (*H0 to H4*).

The fourth step is the main loop which is (*A to E*) will be calculated in 80 Rounds as following:

$$TEMP = (A <<< 5) + F(t,B,C,D) + E + Kt + W[t]$$

$$E = D$$

$$D = C$$

$$C = B <<< 30$$

$$B = A$$

$$A = TEMP$$

Where <<< is left rotate, *k* is round constants and *F (t,B,C,D)* is a function as shown below

$$F = (B \& C) \vee (\neg B \& D) \quad (0 \leq t \leq 19)$$

$$F = B \oplus C \oplus D \quad (20 \leq t \leq 39)$$

$$F = (B \& C) \vee (B \& D) \vee (C \& D) \quad (40 \leq t \leq 59)$$

$$F = B \oplus C \oplus D \quad (60 \leq t \leq 79)$$

Where $\vee$, $\oplus$, & *and* $\neg$ are represent bitwise logical OR, XOR, AND and complement respectively.

The final step is add (*A to E*) with (*H0 to H4*) which produce the message digest of 160-bit.

H0 = H0 + A

H1 = H1 + B

H2 = H2 + C

H3 = H3 + D

H4 = H4 + E

Fig. 1 shows a block diagram for SHA-1 Computation.

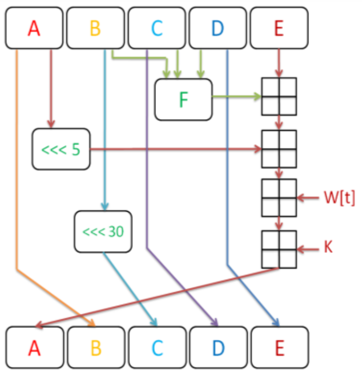


Fig.1 SHA-1 Computation

- SHA-224 & SHA-256 Computation

The message digest is 8 hash variables each of 32 bits are used. It takes 64 rounds to complete SHA-224 and SHA-256 calculation. SHA-256 algorithm is done as follows:

In the first step hash variables shown in table.3 are initialized.

TABLE.3 INITIALIZED SHA-256 V

H0	H1	H2	H3
6A09E667	BB67AE85	3C6EF372	A54FF53A
H4	H5	H6	H7
510E527F	9B05688C	1F83D9AB	5BE0CD19

In the second step message word *W[0-15]* were extended into 64 word *W[16-63]*

S0 = (W[t-15] >>> 7) ⊕ (W[t-15] >>> 18) ⊕ (W[t-15] >> 3)

S1 = (W[t-2] >>> 17) ⊕ (W[t-2] >>> 19) ⊕ (W[t-2] >> 10)

W[t] = W[t-16] + S0 + W[t-7] + S1

Where >>> is right rotate and >> is right shift

In the third step the 8 variables (*A to H*) were added which they equal to (*H0 to H7*).

The fourth step is the main loop where (A to H) will be calculated in 80 Rounds as following:

TEMP1 = H + Z1 + CH + K[t] + W[t]

TEMP2 = Z0 + MAJ

H = G

G = F

F = E

E = D + TEMP1

D = C

C = B

B = A

A = TEMP1 + TEMP2

where *K[t]* is round constants and (*Z0,Z1,CH,MAJ*) equations are shown below

$Z0 = (A >>> 2) \oplus (A >>> 13) \oplus (A >>> 22)$

$Z1 = (E >>> 6) \oplus (E >>> 11) \oplus (E >>> 25)$

$CH = (E \,\&\, F) \oplus ((\neg E) \,\&\, G)$

$MAJ = (A \,\&\, B) \oplus (A \,\&\, C) \oplus (B \,\&\, C)$

The final step is adding (*A to H*) with (*H0 to H7*) which produce the message digest of 256-bit.

$H0 = H0 + A$

$H1 = H1 + B$

$H2 = H2 + C$

$H3 = H3 + D$

$H4 = H4 + E$

$H5 = H5 + F$

$H6 = H6 + G$

$H7 = H7 + H$

SHA-224 and SHA-256 are identical algorithms except that:

The message digest is 224-bit which constructed by omitting H7.

The initial hash values H0 through H7 are shown in table .4.

TABLE.4 INITIALIZED SHA-224 VARIABLES

H0	H1	H2	H3
C1059ED8	367CD507	3070DD17	F70E5939
H4	H5	H6	H7
FFC00B31	68581511	64F98FA7	BEFA4FA4

- SHA-384 & SHA-512 Computation

SHA-512 is identical in structure to SHA-256, but

1. the message is broken into 1024-bit chunks,
2. the initial hash values and round constants are extended to 64 bits,
3. there are 80 rounds instead of 64,
4. the message schedule array *W* has 80 64-bit words instead of 64 32-bit words,
5. to extend the message schedule array w, the loop is from 16 to 79 instead of from 16 to 63,
6. the round constants are based on the first 80 primes 2-409,
7. the word size used for calculations is 64 bits long,
8. the shift and rotate amounts used are different.

$S0 = (W[t-15] >>> 1) \oplus (W[t-15] >>> 8) \oplus (W[t-15] >> 7)$

$S1 = (W[t-2] >>> 19) \oplus (W[t-2] >>> 61) \oplus (W[t-2] >> 6)$

$Z0 = (A >>> 28) \oplus (A >>> 34) \oplus (A >>> 39)$

Z1 = (E >>> 14) ⊕ (E >>> 18) ⊕ (E >>> 41)

SHA-384 is identical to SHA-512, except that

The initial hash values h0 through h7 are different and the output is constructed by omitting h6 and h7.

Fig .2 shows a block diagram for SHA-2 Computation

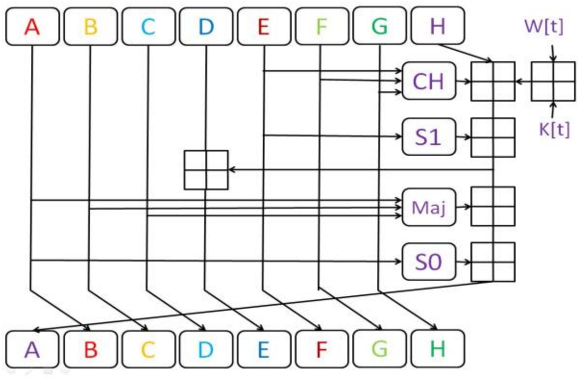


Fig. 2 SHA-2 Computation

4. INSTRUCTION SET ARCHITECTURE

MIPS processor uses 32-bit for each instruction. MIPS defining three instruction formats: R-type, I-type, and J-type. *R-type* instructions have three registers.

I-type instructions have two registers and a 16-bit immediate value. *J-type* (jump) instructions have one register and a 26-bit immediate.

- R-type Instructions

R-type instructions have three registers as operands: two as

sources and one as a destination.

It has six fields: funct(function), op(opcode), rs (source register), rt(target register), rd(destination

register) and shamt(shift operations). Each field is five or six bits, as indicated in fig.3a.

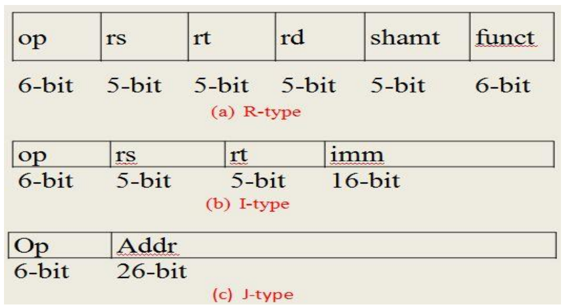
- I-Type Instructions

I-type instructions have two register operands and one immediate operand. It has four fields: imm, op, rs and rt. The first three fields.op, rs and rt, are like those of R-type instructions. The imm field holds the 16-bit immediate, as indicated in fig.3b.

- J-type Instructions

This format is used only with jump instructions. This instruction format uses a single 26-bit address operand, addr,

J-type instructions begin with a 6-bit opcode. The remaining bits are used to specify an address addr, as indicated in fig.3c.



op	rs	rt	rd	shamt	funct
6-bit	5-bit	5-bit	5-bit	5-bit	6-bit

(a) R-type

op	rs	rt	imm
6-bit	5-bit	5-bit	16-bit

(b) I-type

Op	Addr
6-bit	26-bit

(c) J-type

FIG. 3 INSTRUCTION FORMAT

5.Processor Design

The single cycle microarchitecture executes one instruction in one cycle. Because it completes the

operation in one cycle, it does not require any nonarchitectural state. However, the cycle time is limited by the slowest.

Superscalar microarchitecture executes more than one instruction in one cycle, so in this paper we designed 2-way Superscalar of 64-bit for SHA-

512 and SHA-384 and 3-way of 32-bit for SHA-1, SHA-224 and SHA-256.

In this paper a MIPS processor is chosen in our design, because implementation of MIPS processor is easy.

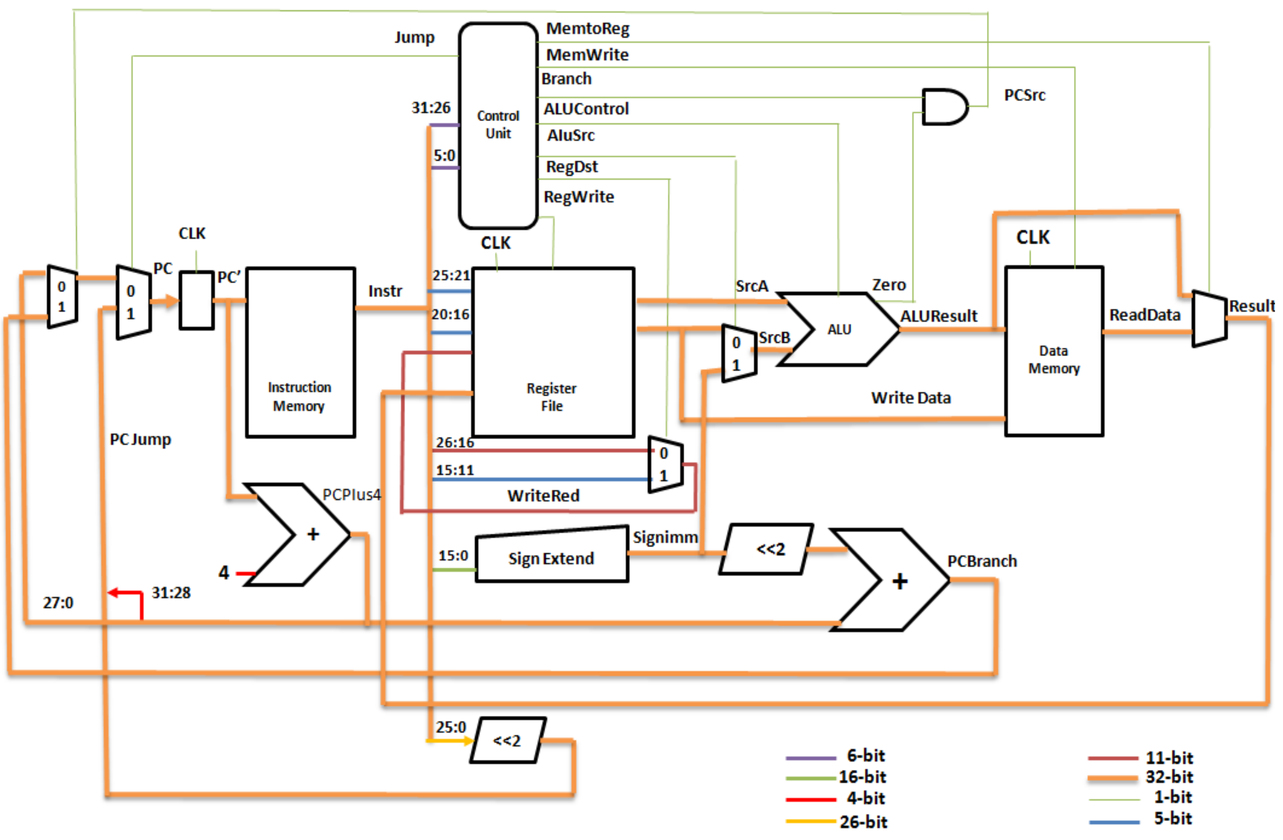


Fig. 4 Implementation of 32-Superscalar MIPS Processor with Only Component That Hash Need

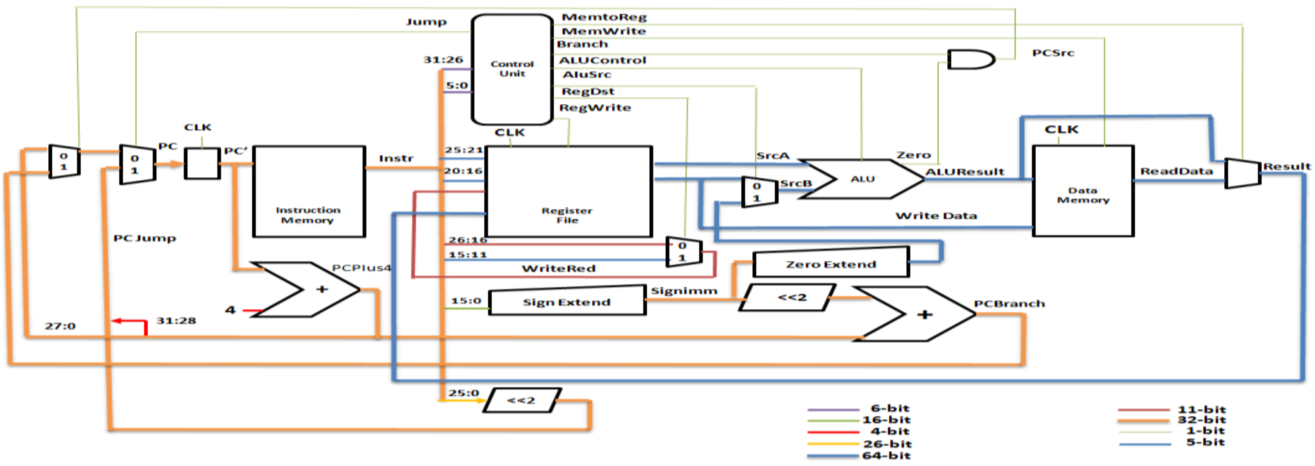


Fig. 5 Implementation of 64-Superscalar MIPS Processor with Only Component That Hash Need

The general MIPS processor contains 49 instructions, but we

don’t need all of them, so we designed a MIPS processor with only the instructions that requires invoking the algorithms of SHA-1, SHA224,

SHA-256, SHA-384 and SHA-512 using VHDL in Xilinx ISE software environment.

Since SHA, SHA224 and

SHA-256 hash variable is 32-bit and SHA-384 and SHA-512 hash variable is 64-bit, so in this paper one superscalar MIPS single-cycle of 32-bit is designed to implement algorithms for (SHA-1, SHA224 and SHA-256) and another superscalar MIPS single-cycle of 64-bit for implementing algorithms of (SHA-384 and SHA-512).

A. 3-way Superscalar MIPS Processor of 32-bit

The simple design of a 32-bit, single cycle MIPS processor consists of two parts:

□ Three of 32-bit Data path

□ Three of Control unit.

• *Three of 32-bit Data path*

A 32-bit single cycle MIPS requires a 32-bit data path. It contains element such as registers, memories, ALUs, sign, extenders and multiplexers.

A descriptor of each data path is shown below:

1. 1.Program counter (PC): One register represents the address of instruction to execute which has 32-bit size.

2. Instruction memory: store instruction to execute hash function and a 96-bit data as the output

3. Register file: Three of 32 register, each has 32-bit in size. It has two read port and one write port.

4. Data memory: have three input write port (WR1, WR2, WR3) and three output read port (RD1, RD2, RD3). each memory location has 32-bit size.

5. Three of sign extension simply copies the most significant bit of a 16-bit input size.

6. Three of ALUs is used in order to execute the arithmetic and logical instruction. ALUs take alucontrol(2:0) as input and generate corresponding function . Note that there is modify in some instruction.

- *Three of control units*

  The control unit received from the

current instruction of the data path which tell it what to do to execute the instruction.

Fig .4 shows the complete design of the Superscalar MIPS processor of 32-bit with all instructions required to invoke

the SHA-1, SHA-224 and SHA-256 algorithms.

A. 2-way Superscalar MIPS Processor of 64-bit

Similar to Superscalar MIPS single-cycle of 32-bit except that

Register file: Three of 32 register, each has 64-bit in size. It has two read port and one Write port.

1. Data memory: each memory location has 64-bit size.

2. some multiplexers modify to work with 64-bit.

3. ALUs is modify in order to execute the arithmetic and logical instruction of SHA-384 and SHA-512.

Fig .5 shows the complete design of the MIPS processor of 64-bit with all instructions required to invoke the SHA-384 and SHA-512 algorithms.

4. RESULTS

The word “abc” was entered by using a keyboard and the result of SHA-1, SHA-224 and SHA-256 is displayed as shown in Fig 6. The time required for processing the three hash functions is 68,435 ns. While the time required for processing SHA-384 and SHA-512 is displayed as shown in Fig 7, which it takes 54,545 ns. Fig 8 shows the summary design of superscalar MIPS process.

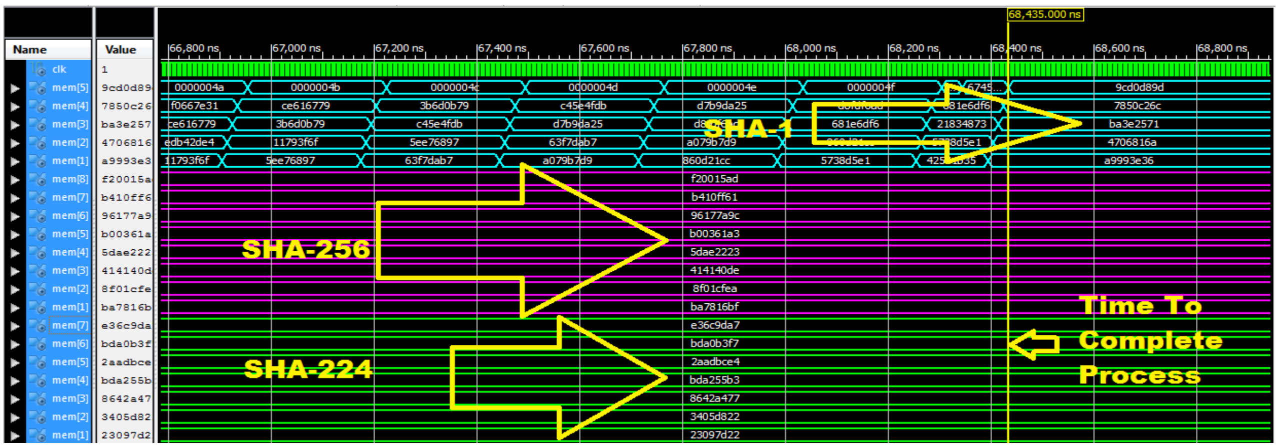


Fig. 6 Word of “abc” and the result of SHA-1, SHA-224 and SHA-256

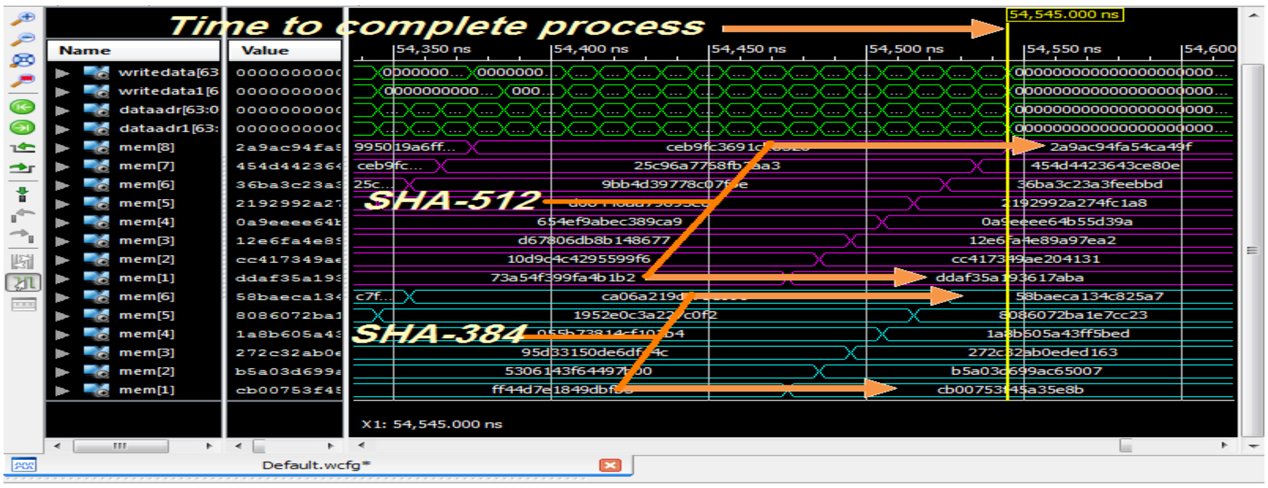
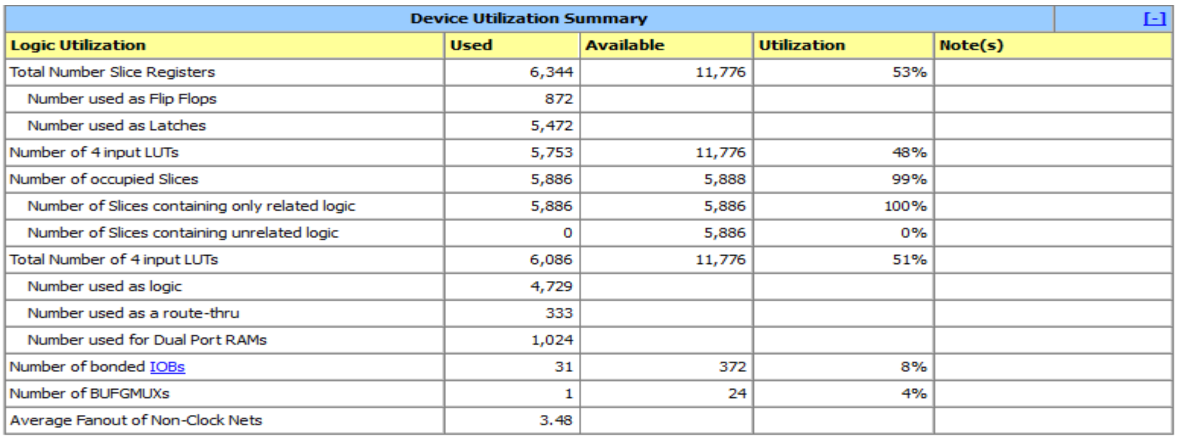


Fig. 7 Word of “abc” and the result of SHA-512 and SHA-384



Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Total Number Slice Registers	6,344	11,776	53%	
Number used as Flip Flops	872			
Number used as Latches	5,472			
Number of 4 input LUTs	5,753	11,776	48%	
Number of occupied Slices	5,886	5,888	99%	
Number of Slices containing only related logic	5,886	5,886	100%	
Number of Slices containing unrelated logic	0	5,886	0%	
Total Number of 4 input LUTs	6,086	11,776	51%	
Number used as logic	4,729			
Number used as a route-thru	333			
Number used for Dual Port RAMs	1,024			
Number of bonded IOBs	31	372	8%	
Number of BUFGMUXs	1	24	4%	
Average Fanout of Non-Clock Nets	3.48			

Fig. 8 Device Utilization Summary of Superscalar

5. Conclusion

A hash SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512 was implemented using FPGA Spartan-3an. A Superscalar MIPS processor was designed using VHDL Xilinx ISE software language with only instructions that Hash need. This study is a starting point for future studies and can be extended to invoke the algorithm for SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512 by using a Multicore MIPS processor.

References

1. A.Kakarountas, A.Miliadonis, and C.Coutis H.Michail, "Efficient implentation of the keyed-hash message authentication code (HMAC) using the SHA-1 hash Function," in 11th IEEE International Conference on Asia South Pacific Design Automation (ASP-DAC'04), 2004, pp. 567-570.

2. C.P.Su, C.T.Huamg, and C.W.Wu M.W.Wowg, "An HMAC processor with integrated SHA-1 and MD5 algorithms," in In Proceeding of the 2004 Conference on Asia South Pacific Design Automation, 2004, pp. 456-458.

3. G. Meliolla, K. A. Nugroho, F. I. Hariadi, "Implementation of hash function on embedded-system platform using chaotic tent map algorithm," in International Symposium on Electronics and Smart Devices (ISESD), Bandung, Indonesia,Nov.2016.

4. M.Askar and T.S.Celebi, "Design and FPGA implementation of Hash processor," in Information Security and Cryptology Conference (ISC 2007), Ankara, Turkiya, Dec. 2007, pp. 85-89.

5. N.Sklavos, and O.koufopavlou G.Selimis, "VLSI implementation of the keyed-hash message authentication code for the wireless ," in *10th IEEE* international Coference on Electronics, Circuit, and System (ICECS'03), Dec. 2003, pp. 24-27.

6. Volnei A. Pedroni, *Circuit Design with VHDL*. London, England: MIT Press, 2004.

7. William Stallings, Cryptography and Network Security Principles and Practices.: Prentice Hall, November 16, 2005.

تصميم من سوبر سكيلر SHA-1 وSHA-2 MIPS المعالج عن طريق FPGA

ليث فؤاد جمعة

قسم هندسة تقنيات الحاسوب

الكلية التقنية الهندسية الكهربائية

صفاء عمران

استاذ مساعد

قسم هندسة تقنيات الحاسوب

الكلية التقنية الهندسية الكهربائية

الخلاصة

ووفقا للتطورات الواسعة النطاق في مجال الاتصالات، هناك طلب على نظام آمن لإرسال البيانات. وظيفة هاش لديها استخدام مهم في التشفير لأمن المعلومات. وتستخدم دالات التجزئة التشفير لحماية سلامة المعلومات والأصالة في مجموعة واسعة من التطبيقات. في هذه الورقة تم تصميم نظام

يتم تنفيذ المعالج.3 ANسبارتان XILINXبرويسور باستخدام SHA-1 وSHA-2

المعالج (المعالج بدون خطوط الأنابيب المتشابكة) MIPSبأستخدام سوبيرسكالار دورة واحدة عن طريق اختيار عدد معين من التعليمات التي ضرورية لاستدعاء الخوارزميات.SHA-512وSHA-384-SHA-256, SHA-224, SHA-1