International Journal of Engineering Inventions
e-ISSN: 2278-7461, p-ISSN: 2319-6491
Volume 3, Issue 12 [Jul. 2014] PP: 111-115

Implementation of SHA-1 Algorithm on FPGA

[1]Mukaramudin, [2]D Narendara chari, [3]L raja
[1,2,3] *ECE Department,Sree Dattha Institute of Engineering & Science*

Abstract: SHA (Secure Hash Algorithm) is famous message compress standard used in computer cryptography, it can compress a long message to become a short message abstract. The algorithm can be used in many Secure Algorithm, especially for DSS. In this paper, the improved version SHA-1 is implemented and analysed. It is then improved and implemented in Verilog and FPGA. Xilinx is used to compile and generate the function modules, RTL level description circuit and simulated waveform. RTL level description is the circuit connection in FPGA chip. It shows the connection of the modules. Simulated waveform shows us the timing and the function of the SHA-1 module. SHA-1 module that designed in this paper used less memory units and logic elements. It can be used in DSA or any protocols or secure algorithm.

Keywords: *SHA-1; Secure Hash Algorithm; FPGA; Networks Security*

I. INTRODUCTION

Nowadays, network is not only a way for us to get more information. It has already become a new life-style for people. For example, network bank, net-shopping, online chat, e-government, etc. All these need a high security network. So networks security has become a very important problem in information age.
In this paper, message compress standard SHA-1 (Secure Hash Algorithm) will be implemented by FPGA (Field Programmable Gate Array) to cooperate the DSS. And it will be used in DSA (Digital Signature Arithmetic) that implemented by FPGA.

II. SHA-1

Hash function is an important part of many cryptoalgorithm, there are 3 famous Hash Algorithm, SHA, MD (Message Digest) and RIPEMD-160 message compress algorithm. In comparison, the security of SHA-1 is better than MD and RIPEMD-160.

SHA (Secure Hash Algorithm) is designed by National Security Agency of the U.S.A. It is a message compress standard is used to cooperate DSS (Digital Signature Standard) that designed by NIST(National Institute of Standards and Technology). Though SHA is designed for DSS, it can be also used in many protocols or secure algorithm. The original version of SHA is called SHA or SHA-0. SHA-1 is the improved version of SHA-0.

Using SHA-1, a message which is no longer than 2^{64} bit can be generated a 160bit message abstract. Message abstract is much shorter than the message itself, so it will spend less time to generate a digital signature. The more important is that the digital signature generate by message abstract has the same security as generate by message.[1]
The most important of all, SHA-1 is implied easily.

III. SHA-1 ARITHMETIC DESIGN AND SYNTHESIS ANALYSIS

Message compress standard SHA is designed for DSS. The input of SHA is a message which is no longer than 2^{64} bit, and it can generate a 160 bit message abstract.
If a message no longer than 264 bit, it needs to be added zeros to make the message become a 264 bit one. And if a message longer than 264 bit, it needs to be separated into several groups. Every group contains 264 bit. Then the message groups will be converted into message abstract groups by SHA algorithm.[2,3]
When message abstract is generated, five 32 bit initial values A, B, C, D, E will be used.
A=67452301
B=EFCDAB89
C=98BADCFE
D=10325476
E=C3D2E1F0
These registers are initialized to the above 32-bit integers (hexadecimal values)
The first four values are the same as those used in MD5. However, in the case of SHA-1, these values are stored in big-endian format, which is the most significant byte of a word in the low-address byte position.
As 32-bit strings, the initialization values (in hexadecimal) appears as follows

Word A: 67 45 23 01
Word B: EF CD AB 89
Word C: 98 BA DC FE
Word D: 10 32 54 76
Word E: C3 D2 E1 F0

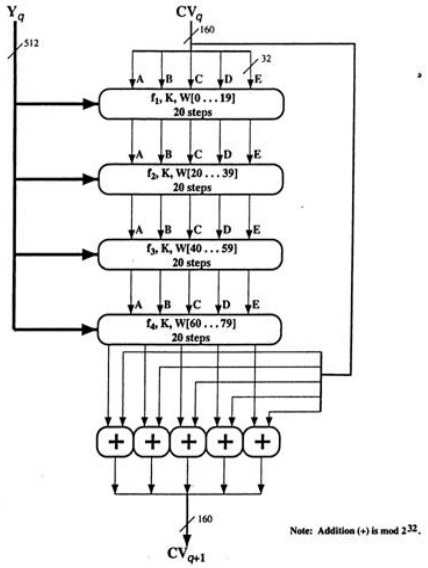

Figure 1: SHA-1 processing of single 512 bit block (Compression function)

The heart of the algorithm is a module that consists of four rounds of processing of 20 steps each. The four rounds have a similar structure, but each uses a different primitive logical function, which we referred to as f1, f2, f3 and f4.

Each round takes as input the current 512-bit block being processed (Y_q) and the 160-bit buffer value ABCDE and updates the contents of the buffer. essh round also makes use of an additive constant K_t, where $0 \le t \le 79$ indicates one of the 80 steps across five rounds. In fact, only four distinct constants are used. The values, in hexadecimal and decimal, are as follows

Table 1: Four Rounds Hash Algorithm

Step Number	Hexadecimal	Take integer part of
$0 \le t \le 19$	K_t = 5A827999	$[2^{30} \times \sqrt{2}]$
$20 \le t \le 39$	K_t = 6ED9EBA1	$[2^{30} \times \sqrt{3}]$
$40 \le t \le 59$	K_t = 8F1BBCDC	$[2^{30} \times \sqrt{5}]$
$60 \le t \le 79$	K_t = CA62C1D6	$[2^{30} \times \sqrt{10}]$

The output of the fourth round (eightieth step) is added to the input to the first round (CV_q) to produce CV_{q+1}. The addition is done independently for each of the five words in the buffer with each of the corresponding words in CV_q, addition modulo 2^{32}.

After all L 512-bit blocks have been processed, the output from Lth stage is the 160-bit message digest. We can summarize the behavior of SHA-1 as follows:

$CV_0 = IV$
$CV_{q+1} = SUM_{32}(CV_q, ABCDE_q)$
$MD = CV_L$
Where
IV: initial value of the ABCDE buffer, defined in step 3
$ABCDE_q$: The output of the last round of processing of the qth message block
L : The number of blocks in the message (including padding and length fields)
SUM_{32} : Addition modulo 2^{32} performed separately on each word of the pair of inputs
MD : Final message digest value

The logic in each of the 80 steps of the processing of one 512-bit block. Each round is of the form
$A,B,C,D,E \leftarrow (E+f(t,B,C,D)+S^5(A)+W_t+K_t), A, S^{30}(B), C, D$

Where
A,B,C,D,E = the five words of the buffer
t = step number; $0 \leq t \leq 79$
f(t,B,C,D) = primitive logical function for step t
S^k = circular left shift(rotation) of the 32-bit argument by the k bits
W_t = 32-bit word derived from the current 512-bit input block
K_t = an additive constant; four distinct values are used, as defined previously
+ = addition modulo 2^{32}

Each primitive function takes three 32-bit words as input and produces a 32-bit word output. Each function performs a set of bitwise logical operations; that is, the nth bit of the output is a function of the nth bit of the three inputs.
The functions can be summarized as follows:

Table 2: Logic Operations In Sha

Step	Function Name	Function Value
$0 \leq t \leq 19$	f1= f(t,B,C,D)	$(B \Lambda C) V (\bar{B} \Lambda D)$
$20 \leq t \leq 39$	f2=f(t,B,C,D)	B xor C xor D
$40 \leq t \leq 59$	f3=f(t,B,C,D)	$(B \Lambda C) V (B \Lambda D) V (C \Lambda D)$
$60 \leq t \leq 79$	f4=f(t,B,C,D)	B xor C xor D

The logical operators (AND,OR, NOT) are represented by the symbols (Λ,V,¯ ,) . As can be seen only three different functions are used. For $0 \leq t \leq 19$, the function is the conditional function: If B, then C else D.
For $20 \leq t \leq 39$ and $60 \leq t \leq 79$, the function produces a parity bit. For $40 \leq t \leq 59$, the function is true if two or three of the arguments are true. Table 12.2 is a truth table of these functions.
It remains to indicate how the 32-bit word values W_t are derived from the 512-bit message. The first 16 values of W_t are taken directly from the 16 words of the current block. The remaining values are defined as follows:
$W_t = S^1(W_{t-16} \text{ xor } W_{t-14} \text{ xor } W_{t-8} \text{ xor } W_{t-3})$

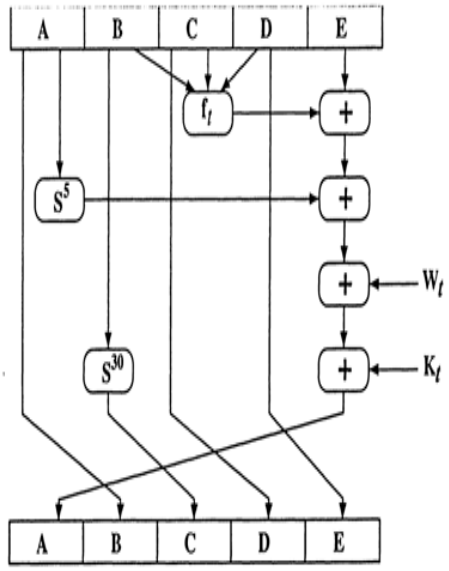

Fig 2: Elementary SHA Operations

Thus, in the first 16 steps of processing, the value of W_t is equal to the corresponding word in the message block. For the remaining 64 steps, the value of W_t consists of the circular left shift by one bit of the XOR of four of the preceding values of W_t

This is a notable difference from MD5 and RIPEMD-160, both if which use one of the 16 words of a message block directly as input to each step function; only the order of the words is permuted from round to round. SHA-1 expands the 16 block words to 80 words for use in the compression function. This introduces a great deal of redundancy and interdependence into the message blocks that are compressed, which complicates the task of finding a different message block that maps to the same compression function output.

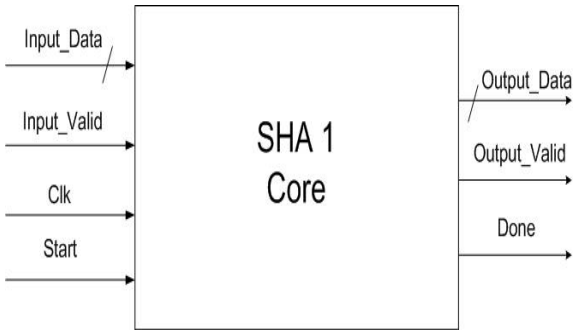

Figure 3: SHA-1 Core

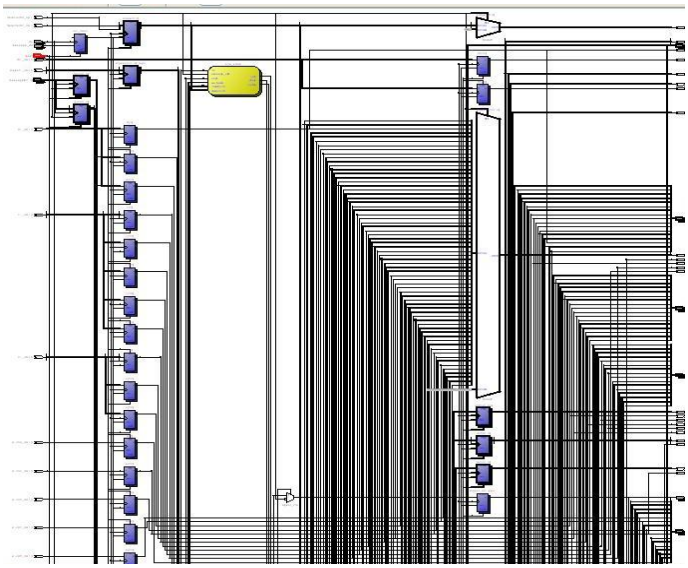

Figure 4: RTL Schematic

IV. RESULT AND ANALYSIS

The module is designed by FPGA. FPGA is a half-order circuit of ASIC (Application Specific Integrated Circuit). It contains many logic cells that has already been distributed in the chip. So it is easy for people to design a system or module for some special function. RTL is the circuit connection in chip.

It shows us the connection of the modules, sub-module and the logic cells. It is also follow the top-to-down structure. We can see the detail structure of the sub-module when we click the sub-module block. When we use FPGA to implement the SHA-1 module, it will be organized according to the RTL level description. RTL is a top-to-down structure. Fig.4 is the top level module of SHA-1.

Fig.5 is the simulated waveform of SHA-1 module, it shows us the timing and the function of the module. The clock period is 20.0ns. The message abstract which is no longer than 264 bit can be generated in about 100ms.

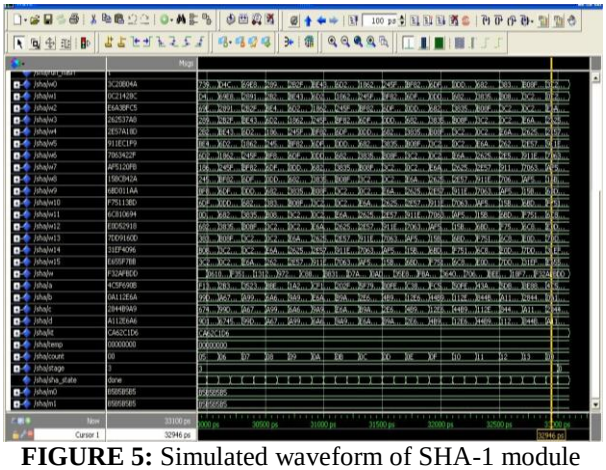

FIGURE 5: Simulated waveform of SHA-1 module

V. CONCLUSION

SHA is famous message compress standard used in computer cryptography. Its improved version SHA-1 algorithm has been analysised in this paper, and implied by Verilog. Xilinx is used to synthesis the module, and then generated RTL level description circuit and simulated waveform.

The SHA-1 module only needs 74 pins, 857 LC registers, 691 LUT-only LCs and 1029 logic elements. Using the SHA-1 module, a long message can be generated a short and safe message abstract in a very short time.

The design of SHA-1 module in this paper can be used to cooperate with DSA, or be an independent module used in many protocols or secure algorithm.

Table 3: Thermal Power dissipation by hierarchy

Sl No.	Compilation Hierarchy node	Total thermal power by hierarchy (1)	Block thermal static power (1)
1	sha	118.69mW	0.00mW

REFERENCES

[1]. Deng An-Wen. Cryptography (in Chinese). China WaterPower Press. 2006:150-152
[2]. Wang Yu-Min. Information hidden: Theory and Technology (in Chinese). Tsinghua Publishing House. 2006:30-35
[3]. Wenbo Mao.Modern Cryptography: Theory and Practice. Pearson Education. 2004:184-190
[4]. Zhang Fang-Guo. The Research on Hyperelliptic Curve Cryptosystems (in Chinese). Xidian University. 2001:22-30
[5]. Wade Trappe, Lawrence C. washington.Introduction to cryptography with coding theary (2nd Editon). Pearson Education. 2006:133-136
[6]. Deng Jian-zhi, Cheng Xiao-hui, Gui Qiong, "Design of Hyper Elliptic Curve Digital Signature," Proc. IEEE International Conference on Information Technology and Computer Science 2009(ITCS 09),IEEE Press, Jul. 2009, pp.45-47
[7]. Knuth, Donald (1973). The Art of Computer Programming, volume 3, Sorting and Searching. pp. 506–542.