International Journal of Information and Computation Technology.
ISSN 0974-2239 Volume 3, Number 8 (2013), pp. 757-764
© International Research Publications House
http://www. irphouse.com /ijict.htm

Implementation of Secure Hash Algorithm-1 using FPGA

Nalini C. Iyer and *Sagarika Mandal

Dept. of Electronics and Communication Engineering,
BVBCET, Hubli, Karnataka, India.

Abstract

Hash functions play a significant role in today's cryptographic applications. SHA (Secure Hash Algorithm) is a famous message compress standard used in computer cryptography, it can compress a long message to become a short message abstract. In this paper, SHA-1 is implemented using Verilog HDL (Hardware Description Language). The SHA-1 Verilog source code is divided into three modules, namely Initial, Round and Top module. The Verilog code is synthesized on Virtex5 FPGA using Xilinx ISE 14.2 software tool. The test vectors have been applied to verify the correctness of the SHA-1 functionality. A comparison between the proposed SHA-1 hash function implementation with other related works shows that it achieves a higher throughput and also higher clock frequency.

Keywords: Secure Hash Algorithm-1 (SHA-1), hash function, Verilog HDL, FPGA.

1. Introduction

Cryptography is one of the most useful fields in the wireless communication area and personal communication systems, where information security has become more and more important area of interest [1]. Cryptographic algorithms take care of specific information on security requirements such as data integrity, confidentiality and data origin authentication.

To assure that a communication is authentic, the authentication service is of much concern. The function of authentication services is to assure recipient that the message is from the source it claims to be [1]. In computer security, the process of attempting to verify the digital identity of the sender of a piece of information is known as authentication.

In order to make a very secure cryptographic portable electronic device, the selected well-known algorithm must be trusted, time-tested and widely peer-reviewed in the global cryptographic community. A one-way hash function is an algorithm that takes input data and irreversibly creates a digest of that data. One of the trusted one-way hash function are SHA-1 (Secure Hash Algorithm) [2,3], SHA-256, SHA-384 and SHA-512. SHA algorithms are called secure because, for a given algorithm, it is computationally infeasible 1) to find a message that corresponds to a given message digest or 2) to find two different messages that produce the same message digest. Any change to a message will, with a very high probability results in a different message digest. This will result in a verification failure when the secure hash algorithm is used with a digital signature algorithm or a keyed-hash message authentication algorithm.

2. Overview of SHA

SHA- 1 is a complex algorithm that involves multiple 32-bit, 5-way additions, complex logical functions, data shifting and a great deal of repetition. Generally implementations of the SHA-1 algorithm have required large die areas and so made fairly expensive portable device. A proposed method has been applied to be relatively inexpensive one. The architecture is presented for SHA-1 hash function. The implementation is conducted using Verilog HDL on Xilinx FPGA device. The synthesis results are presented and compared with other SHA-1 implementations. Here, the hardware terms of system performance (throughput), operating frequency and covered area are compared.

Hash algorithms, also called as message digest algorithms, are generating a unique fixed length bit vector for an arbitrary-length message *M* [4]. The bit vector is called the hash of the message and it is denoted as *H*. This hash value should be the same each time the same input is hashed. A hash function used in cryptography should be one way and collision resistant.

The purpose of a hash function is to produce a fingerprint of a file, message or other block of data. A hash function must have the following requirements:

- One-way property: For any given value h, it is computationally infeasible to find x with $H(x) = h$.
- Weak collision resistance: For any given block x, it is computationally infeasible to find y with $H(x) = H(y)$.
- Strong collision resistance: For any given block x, it is computationally infeasible to find x,y with $H(x) = H(y)$.

SHA (Secure Hash Algorithm) is designed by National Security Agency of the U.S.A. It is a message compress standard used to cooperate DSS (Digital Signature Standard) that is designed by NIST (National Institute of Standards and Technology) [11,12]. 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.

3. Design and Synthesis of SHA-1

The input of SHA-1 is a message which is no longer than 2^{64}bits can generate a 160 bit message abstract. The input is processed in 512-bit blocks. The algorithm processing includes the following steps:

1. Padding: The purpose of message padding is to make the total length of a padded message congruent to 448 module 512. The number of padding bits is between 1 and 512. Padding consists of 1 single 1-bit followed by a series of 0-bit.
2. Appending Length: A 64-bit binary representation of the original length of the message is appended to the end of the message.
3. Initialize the SHA-1 Buffer: The message digest is computed using the final padded message. The computation uses two buffers, each consisting of five 32-bit words, and a sequence of eighty 32-bit words. The words of the first 5-word buffer are labeled A, B, C, D and E. The words of the second 5-word buffer are labeled Ho, H1, H2, H3 and H4. The words of the 80-word sequence are labeled W0, W1, W2, W3, W4, ..., W79. A single word buffer Temp is also employed. Before processing any blocks, the Ho, H1, H2, H3 and H4 are initialized as follows: in hex,
   H_0 = 67452301, H_1= EFCDAB89, H_2 = 98BADCFE, H_3 = 10325476 and H_4 = C3D2E1F0
4. Hash Calculation: SHA1 may be used to hash a message, M, having a length of l bits, where $0 \leq l \leq 2^{64}$.

The algorithm uses:

- A message schedule of 80x32-bit words. The words f the message schedule are labeled W0, W1, W2, ……., W79 and W80.
- Five working variables of 32-bits each. The working variables are labeled as: A, B, C, D and E.
- A hash value of five 32-bit words. The words of the hash value are labeled as: $H0_{(i)}$, $H1_{(i)}$, $H2_{(i)}$, $H3_{(i)}$, $H4_{(i)}$ which will hold the initial hash value $H_{(0)}$, replaced by each intermediate hash value (after each message block is processed) $H_{(i)}$ where i denotes the number of 512 bit block being processed in the message M and ending with the final hash value H(N) where N is the number of the last 512 bit block in the message M.
- A single temporary word, T. Previously defined constants which are labeled K_t, where t is the round number.

The calculation is carried out as follows:

The message schedule is prepared, i.e. the message word that is going to be used in that round is prepared. This computation is done as described in the following formula:

$W_t = M^i_t \; 0 \leq t \leq 15$

$W_t = ROTL^1(Wt\text{-}3 \oplus Wt\text{-}8 \oplus Wt\text{-}14 \oplus Wt\text{-}16) \; 16 \leq t \leq 79$

In the above formula M^i_t denotes the t^{th} 32-bit message word of the i^{th} 512-bit message block in the message M. The 5 working variables A, B, C, D and E that are going to be used in the computation are prepared as follows:

$A = H0^{(i-1)}$, $B = H1^{(i-1)}$, $C = H2^{(i-1)}$, $D = H3^{(i-1)}$ and $E = H4^{(i-1)}$

After these initializations, the final values of the working variables for that round are calculated as described below:

$T = S^5(A) + f(t;B,C,D) + E + Wt + Kt$, $E = D$, $D = C$, $C = S^{30}(B)$, $B = A$, $A = T$

Finally, when all 80 steps have been processed, the following operations are performed:

H0 ←H0+A
H1 ←H1+B
H2 ←H2+C
H3 ←H3+D
H4 ←H4+E

5. Output: - When all M_i have been processed with the above algorithm, the 160-bit hash *H* of *M* is available in H0, H1, H2, H3 and H4.

4. Verilog Implementation

In this study, SHA-1 is designed on hardware description language Verilog HDL [13]. Separate modules are synthesized and analyzed using Xilinx ISE 14.2 tool. The Verilog implementation was divided into three modules:

- Initial module:- It accepts a set of text data. Later 512 bits message block is prepared and broken into 16 32-bit words. Every 32-bit word is used for every round.
- Round module:- It performs the hashing calculation and operation on the input message block on previous hash output and generates new hash value.
- Top module:- This module is the control unit of SHA-1 in which the flow of the algorithm is followed and maintained.

5. Experimental Results and Discussion

Virtex-5 FPGAs offer the best solution for addressing the needs of high performance logic designers, DSP designers and embedded systems designers with unprecedented logic, DSP, hard/soft microprocessor and connectivity capabilities. Hash core is fully described using Verilog HDL on Xilinx ISE software. Target FPGA also belongs to the same company. This is an advantage since Xilinx ISE software provides full support for all the code to FPGA processes for Xilinx FPGAs.

The device utilization summary after synthesis of top module using Virtex-5 as target device is given in Table I. Table II shows the synthesis results for the implementation. Comparisons of the proposed SHA-1 hash function implementation

with other related works are shown in Table III, from which it is observed that the proposed work achieves a higher throughput and also higher frequency.

Table 1: Device Utilization Summary after Synthesis
(Virtex5-Xc5vlx50t)

Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	2446	28800	8%
Number of Slice LUTs	8202	28800	28%
Number of LUT-FF pairs	422	10226	4%
Number of bonded IOBs	322	480	67%
Number of BUF	1	32	3%

Table 2: Implementation Result of SHA-1
(VIRTEX5-XC5VLX50T)

Parameter	**Value**
Slices	1351
Max. Clock Frequency	124.502MHz
Throughput	786Mbps`
Throughput per Slice(TPS)	0.582Mbps/slice

Table 3: Implementation Comparison Results.

Implementation	Frequency (MHz)	Area (Slices)	Throughput (Mbps)
[5] Virtex-II	117.5	1275	734
[6] Virtex v200pq240	74	2384	291, (512) 467, (256)
[8] Spartan-2 XCS-100	106	423	212
[14] Xilinx XCV1000-6	87	2212	530
Proposed work Virtex-5 XC5VLX50T	124.502	1351	786

6. Conclusion

The proposed design is verified using Xilinx ISE 14.2 tool on software by timing simulation and is implemented using Virtex-5 XC5VLX50T-1 FPGA hardware. In this study, Verilog HDL is used to design and capture SHA-1 in Xilinx ISE software environment. The design is implemented on Xilinx FPGA and results are given as shown in Table II. For verification, test vectors are used and observed that the design generates correct hash values. From Table III, the proposed SHA-1 architecture achieves a higher working frequency and also higher throughput.

References

[1] William Stallings, "Cryptography and Network Security, Principles and Practices" Fourth Edition, 2005.

[2] NIST "SECURE HASH STANDARD", Federal Information Processing Standards Publication 180-1, May 1993.

[3] NIST "SECURE HASH STANDARD", Federal Information Processing Standards Publication 180-2, August 2002.

[4] Ilya Mironov, "Hash functions: Theory, attacks and applications", Microsoft Research, Silicon Valley Campus, November 14, 2005.

[5] K. Jarvinen, "Design and Implementation of a SHA-1 Hash Module on FPGAs", Technical Report, Otakaari 5A, Espoo, Finland, November, 2004.

[6] Dai Zibin and Zhou Ning, "FPGA Implementation of SHA-1 Algorithm", IEEE Proceedings, 5th International Conference on ASIC, 2003.

[7] A.P. Kakarountas, G. Theodoridis, T. Laopoulos and C.E.Goutis, "High-Speed FPGA Implementation of the SHA-1 Hash Function", IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Sofia, Bulgaria, 2005.

[8] Guopyin Wang, "An Efficient Implementation of SHA-1 Hash Function", IEEE International Conference on Electro or Information Technology, pp. 575-579, 2006.

[9] Cheng Xiao-hui and Deng Jian-zhi, "Design of SHA-1 Algorithm based on FPGA", IEEE Second International Conference on Networks Security, Wireless Communications and Trusted Computing, (NSWCTC), Vol-1, pp. 532-534, 2010.

[10] Zhou Hua and Liu Qiao, "Hardware Design for SHA-1 Based on FPGA", IEEE International Conference Publications on Electronics, Communications and Control (ICECC), pp.2076-2078, 2011.

[11] Quynh Dang, "Recommendation for Applications Using Approved Hash Algorithm", NIST special publication 800-107, Computer security Division, National Institute of Standards and Technology, Dept of commerce, USA, pp. 1-21, 2011.

[12] NIST "Digital Signature Standard", Federal Information Processing Standards Publication 186, 1 May 1994.

[13] Samir Palnitkar, "Verilog HDL A Guide to Digital Design and Synthesis", 2nd Edition, 2003.

[14] T. Grembowski, R. Lien, K. Gaj, N. Nguyen, P. Bellows, J. Flidr, T. Lehman, and B. Schott, "Comparative analysis of the hardware implementations of hash functions SHA-1 and SHA-512" in Proc. of 5th International Conference on Information Security (ISC 2002), Sao Paulo, Brazil, Oct. 2002, pp. 75-89.