International Journal Of Advanced Research In Engineering Technology & Sciences

ISSN: 2394-2819

Email: editor@ijarets.org March- 2015 Volume 2, Issue-3 www.ijarets.org

FPGA Implementation of Secure Hash Algorithm for Password Security

Kashika Garg
M.Tech Student
Department of ECE
SRM University NCR Campus
Modinagar

Manoj Vishnoi
Asst. Professor
Department of ECE
SRM University NCR Campus
Modinagar

ABSTRACT:
In this paper, an FPGA based hash algorithm is designed and implemented using a VERILOG. Hash functions are among the most important cryptographic primitives and used in the several fields of communication integrity and signature authentication. These functions are used to obtain a fixed hash value of an arbitrary long message.

I. INTRODUCTION

The US National Institute of Standard and Technology (NIST) proposed the secure hash algorithm standard and first published in 1993. The first revision to this algorithm was published in 1995 and was called SHA-1. This hash function is implemented by using cryptography. Unlike other crypto algorithms the hash function does not have a key. The use of hash function is manifold: Hash function is an essential part of digital signature schemes and message authentication codes. Cryptographic algorithms fulfil the security requirements such as data integrity, confidentiality and data origin authentication. A hash function is sort of operation that takes an arbitrary length of input and produce fixed length of output. The produce fixed length output is known as message digest. The size of the message digest depends upon the algorithm used. Furthermore a very small change in input results in different hash values.

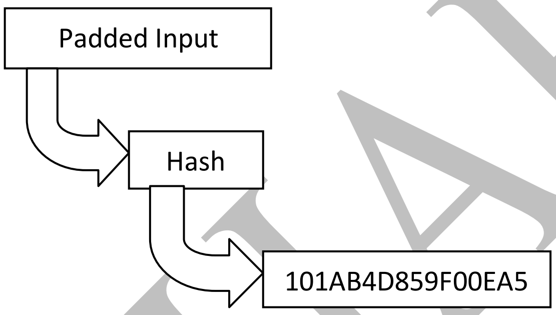

Fig 1.1: Hash Operation

A set of more function was published by NIST whose output ranges from 224 bit to 512 bit. These hash algorithms, called SHA-224, SHA-256, SHA-384 and SHA-512. The SHA-3 hash function was announced on November 2, 2007. The new standard is scheduled to be published in 2012.

II. SHA 1 DESIGN AND IMPLEMENTATION

2.1 SHA 1

The secure hash algorithm 1 is cryptographic function that was digest by National Security Agency and was first published in 1995. The SHA -1 is widely used in various application such as TLS, SSL, and SSH. The SHA-1 has a 512 bit input block size and produces a 160 bit output by using hashing function. The computation function processes the message 512 bit chunks. This compression function consists of 80 rounds which are divided into four rounds each. The steps of an SHA-1 are:

Padding: suppose that we have a message x of l bits. To obtain an overall message size of a multiple of 512 bits, we append a single "1" followed by k zeros bits and the binary 64-bit representation of l. Consequently, the number of required zeros k is given by:

K = 512-64-l-1
= 448-(l+1) mod 512

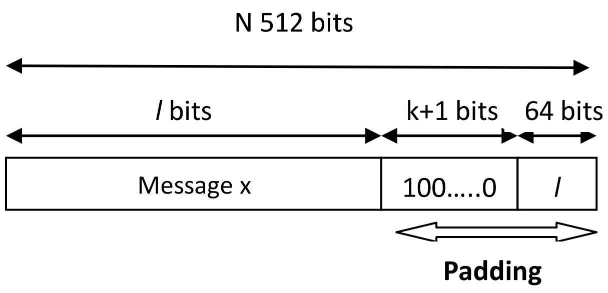

Fig 2.1.1: Padding of input message

Initial Value H_0 : For the first iteration the initial hash value is used to be hold in a 160-bit buffer. The first 32-bit are fixed and are given in hexadecimal notation as:

$A=H_0^{(0)}$=67452301
$B=H_0^{(1)}$=EFCDAB89
$C=H_0^{(2)}$=98BADCFE
$D=H_0^{(3)}$=10325476
$E=H_0^{(4)}$=C3D2E1F0

Message scheduling: Prior to applying the compression function, we need to divide the message into 512-bit blocks thus derived W_j Each 512-bits block is subdivided into 16 words of size of 32 bits. Thus, 32- bits words are computed for each of the 80 rounds by the method :

$$W_j = \begin{cases} X_i & 0 \le j \le 15 \\ (W_{j-16} \text{ XOR } W_{j-14} \text{ XOR } W_{j-8} \text{ XOR } W_{j-3}) \lll 1 & 16 \le j \le 79 \end{cases}$$

Fig 2.1.2: Message Scheduling

Table 2.1.1: Round functions and round constants for the SHA rounds

Stage t	Round J	Constant K_t	Function f
1	**0….19**	**K_1 = 5A827999**	**F_1(B,C,D)=(B&C)|(~B&D)**
2	**20…39**	**K_2 = 6ED9EBA1**	**F_2(B,C,D)=B xor C xor D**
3	**40…59**	**K_3 = 8F1BBCDC**	**F_3(B,C,D)=(B&C)|(B&D)|(C&D)**
4	**60…79**	**K_4 = CA62C1D6**	**F_4(B,C,D)=B xor C xor D**

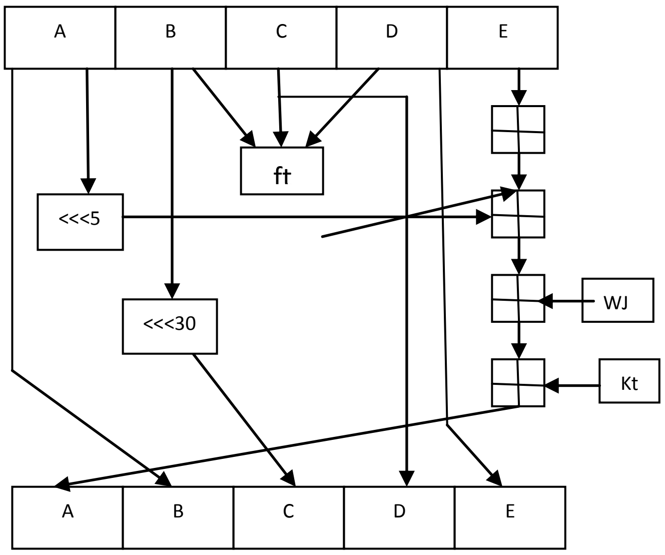

Fig 2.1.3 : SHA 1 Process

2.2 DIGITAL SIGNATURE

A message can be distributed by using digital signature in the form of plain text. Digital signature is first converted into a binary data with the help of software like MATLAB which is the input to SHA gives fixed bit length hash output of fixed length depending upon the type of SHA used. This hash value is then verified against the signature by using the public key of the signer. If the hash value and the signature match, you can be confident that the message is indeed the one the signer originally signed and that it has not been tempered.

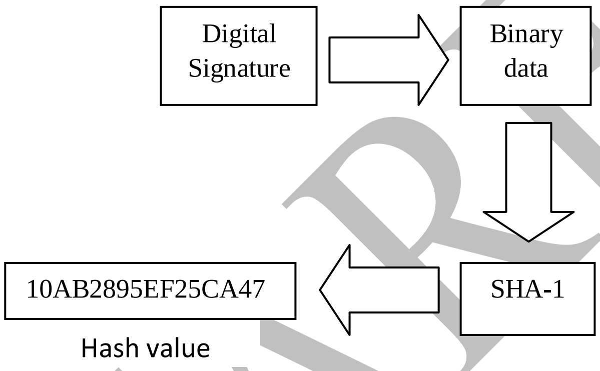

Fig 2.2.1: Digital Signature Encryption

III. RESULT AND DISCUSSION

The code is written by using VERILOG to configure the FPGA. The whole algorithm is divided into four modules:

Padding module: It collect the input data of arbitrary length and it sends to the 512 bit blocks to the next module.

Message scheduling: 512-bits block is subdivided into 16 words of size of 32 bits. Thus, 32- bit words are computed for each of the 80 rounds by the method described above.

SHA process: It performs hash calculation.

Top module: It instantiate all three module together and give the 160 bit hash value as output.

The whole secure hash algorithm module was synthesized and implement with the help of Xilinx ISE Design suit 14.6. The fig below shows the simulation result of SHA 1 implementation of input “ABC”.

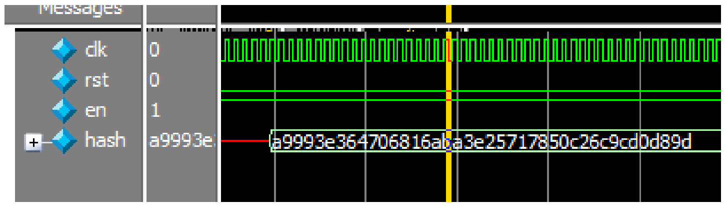

Fig 3.1: Simulation result

Table shows the FPGA simulation result synthesis on xc3s500e-4fg320
Table3.1: FPGA Synthesis Result

Logic Utilization	Used	Available	Utilization
No. of Slice FF	4001	9312	42%
No. of 4 input LUTs	5633	9312	60%
No. of occupied slices	3452	4656	74%
No. of bonded IOBs	163	232	70%

III. CONCLUSION AND FUTURE SCOPE

This work presents the hardware design and verification of the customized processor for executing the SHA-1 algorithm by timing simulation. The design is implemented on xc3s500e-4fg320 using Xilinx ISE 14.2 tool. The verilog description of the processor is well mapped to the LUT’s contained in common FPGA’s, making an efficient use of the available area. In future the speed can be increased by merging several register cycles into a single cycle. By doing efficient floor planning the area and delays can further be reduced.
.

IV.REFERENCES

1. NIST, “Secure Hash Standard”, FIPS PUB 180-1, May 1993.
2. NIST, “with change notice Secure Hash Standard”, FIPS PUB 180-2 August 2002.
3. NIST, “Digital Signature Standard (DDS)”, FIBS PUB 186 May 1994.
4. “An Overview of Cryptographic Hash Functions and Their Uses”, SANS Institute, 2003
5. R. Rivest, “MD5 Message-Digest Algorithm”, RFC 1321. MIT Laboratory for Computer Science and RSA Data Security Inc, 1992.
6. Yong Kyu Kang, Dae Won Kim, Taek Won Kwon, Jun Rim Choi, “An Efficient Implementation of Hash Function Processor for IPSEC”, In Proceedings of the IEEE AsiaPacific Conference on ASIC, pp. 93-96, August. 2002
7. Volnei A. Pedroni, “Circuit Design with VHDL”, MIT Press Cambridge, Massachusetts, 2004.
8. Clive Maxfield, “The Design Warrior’s Guide to FPGAs”, Elsevier Newnespress, 2004.
9. http://encryption_policies.tripod.com/us/denning_1095_future.htm
10. https://www.schneier.com/essay-005.html
11. www.xilinx.comChipScope Pro 10.1 Software and Cores User Guide UG029 (v10.1) March 24, 2008.