Research article Available online www.ijsrr.org ISSN: 2279–0543

International Journal of Scientific Research and Reviews

Implementation of Secure Hash Algorithm1 for High Throughput and Optimized Performance

PanduSagara*[1] and A. Sindhura[2]

Dept. of Electronics and Communication Engineering, G. V. P College of Engineering (A), Visakhapatnam, AP-India. E-mail:pandu_sagara@gvpce.ac.in, Email:sindhuarya17@gmail.com.

ABSTRACT

Information security provides protection of information against unauthorized disclosure, data collapse, and modification. Using virtual networks, the increase in demand for encryption of data is major concern in computer systems and digital communication. Field Programmable Gate Arrays (FPGAs) enter a significant growth due to their reprogrammable nature, flexibility and compatibility in fabrication process. Generally, secure communication includes authentication, confidentiality, non-repudiation, and integrity of the data. Cryptography is an essential part of secure communication. Cryptographic algorithms are designed to provide strong bit authentication. Compared to three classes of cryptography hash algorithms plays a major role. In this thesis, Secure Hash Algorithm-1 (SHA-1) is implemented for achieving high throughput and optimized performance using different modules in Xilinx FPGA devices. The entire architecture is coded using Verilog HDL RTL constructs. All the modules of the architecture are been functionally synthesized and simulated using Xilinx ISE Design tool 14.7. The proposed work is to provide security which is the major challenge for FPGAs and SOCs (System on Chip).

KEYWORDS: FPGA; Secure Communication; Hash Algorithms; SHA-1.

***Corresponding author**

Sri SagaraPandu

Assistant Professor of ECE Department,

G. V. P. College of Engineering (A),

Visakhapatnam, AP, India.

Email:pandu_sagara@gvpce.ac.in

Phone number- 7989496718

INTRODUCTION

Information security is one of the real issues in this world. The prerequisites of information security have experienced changes in the most recent couple of decades. Before the widespread use of data handling equipment, the security of information must be profitable to an organization. Network security measures the need to protect data during their transmission. It is significant to decide the authenticity of the received information.

Cryptography is one of the most useful fields in the wireless communication area and personal communication systems where information security has become important. A cryptographic hash function is designed to ensure message integrity. Cryptographic hash functions are used to compress and encrypt large messages into smaller message digest.

The authenticity of the confidential information like bank account number, passwords, credit card numbers, FPGA bitstream and other personal data is mandatory in order to avoid fraud and false representations. Various cryptographic algorithms look after confidentiality and data authentication for the security of the information.

Any unplanned or intentional alteration to the data will result in a different hash value. A good hash algorithm must make it impossible to either create an initial input that produces a specific hash value or allow the original input to be intended from the hash value. A hash function is a function that takes some message of any length as input and transforms it into a fixed-length output called a hash value, a message digest, a checksum, or a digital fingerprint.

The paper is organized as follows: Section 2 describes cryptographic hash functions and features of modern cryptography. Section 3 explains the Hash Functionsand properties of hash functions. Section 4 determines Secure Hash Algorithm-1and processing steps of algorithm and Section 5 describes the results and corresponding output. In section 6 provides conclusion and future directions for improving work.

CRYPTOGRAPHIC HASH FUNCTIONS

Nowadays, a lot of applications make use of cryptographic hash functions to provide the security for transmitted information. Cryptography stands for the study of techniques which deals with numerous security matters similar to message authentication, confidentiality, integrity, and non-repudiation. Most cryptographic hash capacities which are intended to take a string of any length as information and deliver specified length hash string. Modern cryptography is mainly based on numerical theory and computer science. The features of modern cryptography are authentication, data integrity, confidentiality and non-repudiation.

- **Confidentiality:**

The purpose of confidentiality is to maintain information private, which must not be disclosed to unauthorized parties. Confidentiality makes the effort for converting simple information to an unreadable message, except by authorized parties. This conversion is known as encryption. Decryption is applied for unreadable information to get original information. Encryption can be provided by symmetric and asymmetric algorithms.

- **Data Integrity:**

For every communication, integrity of the data is important. While sending the data it must not be changed by unauthorized way. Maintenance of data integrity is done at creation, transmission, and storage. Modification of data contains inserting false data, deleting original data. To overcome these things cryptographic techniques like digital signatures and Message Authentication Codes (MAC) are used, which are able to notice both accidental and intentional alterations.

- **Non-repudiation:**

Non-repudiation is a security service that one cannot reject a transaction. This is regularly used in digital signatures and email messages. For example, if non-repudiation service is applied in the transaction, once the purchaser had placed an order, the purchaser cannot refuse the purchase order.

- **Authentication:**

Authentication is a method of identifying user's identity. Authentication has concern for source and integrity services. Identifying an entity through username is source. Usually passwords for username are kept unaltered and protected which provides authenticated integrity. Authentication is a process of giving access to entity based on their identity.

HASH FUNCTIONS

The fundamental idea of hash function is to compress the any variable length of input to fixed length output, comparatively smaller size than input [1]. A hash function is one-way and irreversible, which is computationally impossible to get the original input from hashed output. Each hash result is unique and for a change in input the output must produce differently. The hash functions are having following properties:

- **Pre-image resistance**

An arbitrary length of message 'M' is considered and hash 'h' is calculated for the message. Since it is easy to compute the hash from a message and difficult to get the message from the hashed data (one-way property).

$$h = H\ (M) \qquad (1)$$

- **Second pre-image resistance**

Second pre-image resistance property is also known as weak collision resistance in hash functions. It is hardly impossible to locate any second input that has same output as specified input.

$$M1 \neq M2 \text{ where hash } (M1) = \text{hash } (M2) \qquad (2)$$

- **Collision resistance**

It is computationally impossible to find two different pair of messages (M1, M2) with very low probability having same hash H(M1) = H(M2). This is a strong collision resistance property.

These properties of cryptographic hash functions prevent different types of attacks. Due to this secure authentication and efficiency the hash algorithms are used in Digital Signature Standards (DSS).

SECURE HASH ALGORITHM-1

The Secure Hash Algorithm-1 popularly known as SHA-1 is a cryptographic hash algorithm. This is one of the four algorithms in SHA family. This algorithm was developed by US National Security Agency (NSA) and published by NIST (National Institute of Standards And Technology) in the year 1995. SHA-1 is derived from MD4 (Message Digest 4) algorithm. The first version is SHA-0 which produces fixed length output of 160-bits [2]. Due to security threats in SHA-0, SHA-1 is replaced. This is a one-way algorithm, once the message is hashed using this algorithm it produces a message digest of fixed length and cannot get back the original message. The input is given in a variable length which must be less than 2^{64} bits. This is mainly used in Digital Signature Standard.

Hash algorithm does not have any key for encrypting the data like in symmetric and Asymmetric algorithms. It produces a unique checksums for the given input[3]. Whenever sender encrypts the message using hash algorithm, the same hash algorithm must be used by the receiver to check the message integrity. Digital Signature Algorithm (DSA) takes the input as message digest which verifies the signature of the message[4]. Since the message digest is smaller than the actual message, signing the message improves efficiency. The change in single bit of input results in completely different hash output. There will not be any similarity of digested value due to single bit change[5]. The SHA-1 algorithm consists of following steps to process a message. They are:

- **Padding the message**

Padding means nothing but expanding the message. Consider a message m having l bits, this message is padded to make the length of 448 bits. Padding is made by adding 1 at least significant bit of the message followed by zeroes to reach the length which must be multiple of 512-bit block.

- **Appending the length**

In 512-bit block size the length of the 64-bits is appended at end of the padded message. This represents the original length of the message where bytes of original message are converted to bit format. If over flow occurs only lower order 64-bits are considered. This 64-bits are again divided into 32-bit each, appending lower-order first followed by higher order word.

- **Processing functions preparation**

SHA-1 has totally four processing functions which are used for 80 iterations. The first processing function is applied for first 20 iterations. Here, second and fourth processing functions are same. Same function is used twice.

a) f(t; B,C,D) = (B AND C) OR ((NOT B) AND D) (0<=t<=19)
b) f(t; B,C,D) = B XOR C XOR D (20<=t<=39)
c) f(t; B,C,D) = (B AND C) OR (B AND D) OR (C AND D) (40<=t<=59)
d) f(t; B,C,D) = B XOR C XOR D (60<=t<=79)

- **Processing constants preparation**

SHA-1 consists of four processing constants which are repeatedly used for 80 iterations. These constants are 32-bit each, first processing constant is applied for first 20 iterations and followed by remaining constants.

a) K(t) = 0x5A827999 (0 <= t <= 19)
b) K(t) = 0x6ED9EBA1 (20 <= t <= 39)
c) K(t) = 0x8F1BBCDC (40 <= t <= 59)
d) K(t) = 0xCA62C1D6 (60 <= t <= 79)

- **Initialization of buffers**

SHA-1 initializes MD (Message Digest) buffers in order to calculate hash of the input values. H_0, H_1, H_2, H_3, and H_4 are five 32-bit buffers which are initialized with hexadecimal values called fixed constants.

a) H_0 = 67452301
b) H_1 = EFCDAB89
c) H_2 = 98BADCFE
d) H_3 = 10325476
e) H_4 = C3D2E1F0

- **Hash Calculation**

The message is processed in 512-bit blocks, the message is scheduled in 512-bit blocks and these are divided into 16 32-bit words further extended to 80 words. For the first 16 rounds these 16

words are given for processing. After 16 rounds the process is repeated five times until (W_{79}) last round. The message schedule is done using the following expression.

$$W_t = M_t^i \qquad 0 \le t \le 15 \qquad (3)$$

$$W_t = ROTL^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) \qquad 16 \le t \le 79 \quad (4)$$

Where M_t^i represents t[th]32-bit message of the i[th] 512-bit block in the message M. The five 32-bit words A, B, C, D, and E are defined at the beginning of the process of M^i. The algorithm process one M^i at once, starting from M^0 to M^i. Here, A = H0, B = H1, C = H2, D = H3, and E = H4 at beginning of the process.

- **SHA-1 round function**

Each round has 5 32- bit inputs; bit wise logical operations are performed and produces 32-bit output. This is the process for one round operation. The round operation is dependent on non-linear function F_t. The rotate left shift operation is performed for mixing up input bits in different positions completely. This operation is repeated 80 times with 20 rounds each.

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

$$T = ROTL5(A) + f(t; B, C, D) + E + W_t + K_t$$

$$E = D$$

$$D = C$$

$$C = ROTL30(B)$$

$$B = A$$

$$A = T$$

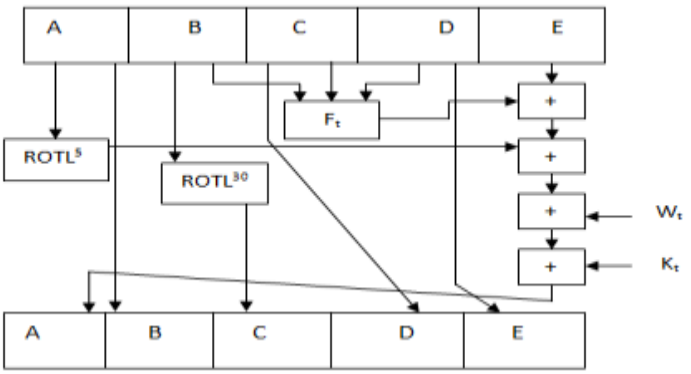

Figure 1. SHA-1 round function

- **SHA-1 compression function**

The computation of 512-bit block is performed in 80 rounds. The four rounds are similar but

designed with different logic functions.

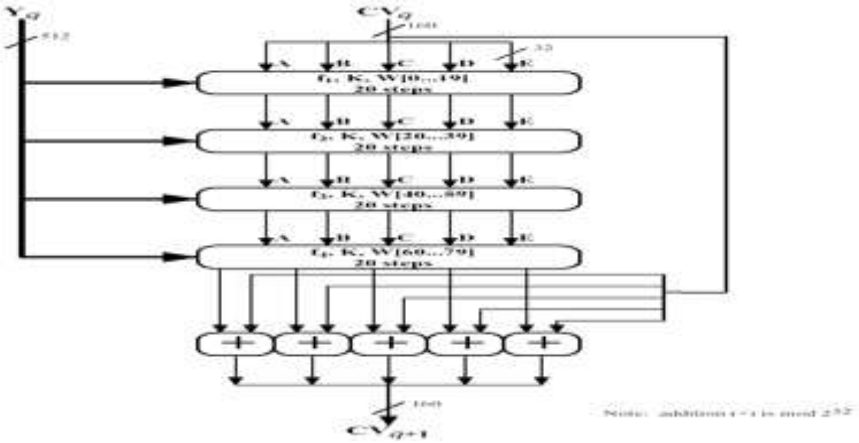

Figure 2. SHA-1 compression function

After the processing of compression function, the output of eightieth round is added to the initial hash value through modulo addition which produces the 160-bit output.

$$H0 \leftarrow H0 + A$$

$$H1 \leftarrow H1 + B$$

$$H2 \leftarrow H2 + C$$

$$H3 \leftarrow H3 + D$$

$$H4 \leftarrow H4 + E$$

RESULT

Synthesis report of Spartan-3E xc3s500e-4fg320 device specifies logic utilization of slices, slice flip flops, LUTs, and GClKs.

Table 1. Device Utilization Summary of Spartan-3E

Logic Utilization	Used	Available	Utilization
Number of Slices	659	4656	14%
Number of slice flip-flops	711	9312	7%
Number of 4 input LUTS	1174	9312	12%
Number of GCLKs	1	24	4%

Simulation results are also specified. The throughput of the design is calculated using the following notation:

Throughput = (Message block size * Clock frequency) / Clock cycles per block

Throughput of implemented design = 535.82 Mbps

Throughput per slices (TPS) = Throughput / Slices = 0.813Mbps / slice

Table 2. Timing Report

Timing report	Values obtained
Minimum Period	11.653 ns
Minimum input arrival time before clock	4.698 ns
Maximum output required time after clock	8.773 ns
Maximum combinational path delay	9.274 ns
Maximum Frequency	85.815MHz

The frequency and delay results after synthesis using Spartan-3E are shown in the table 2.

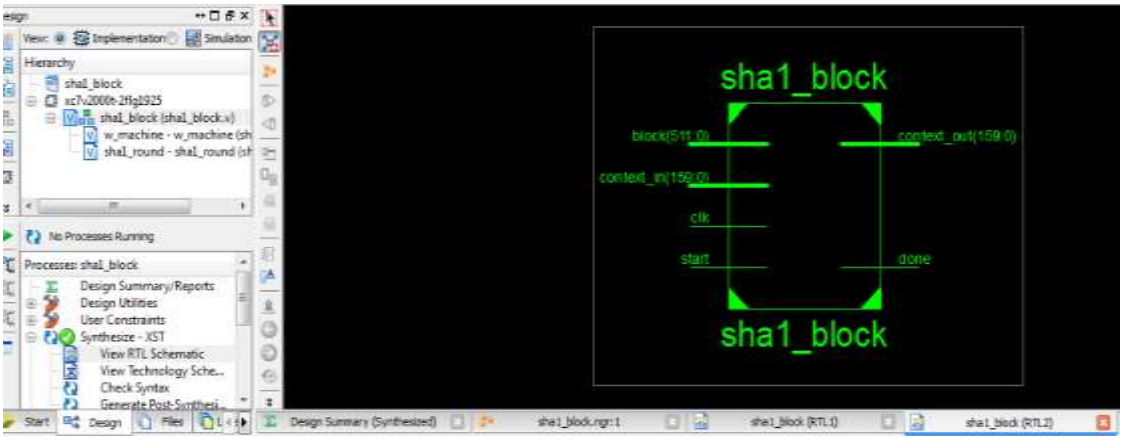

Figure 3. RTL Schematic

The RTL schematic of the design is shown in the figure 3, based on the input context output context is generated. For any given message input the output generated is not identical. It generates a unique message digest. The simulated output shows the hash result which is exactly of SHA-1 online hash calculator.

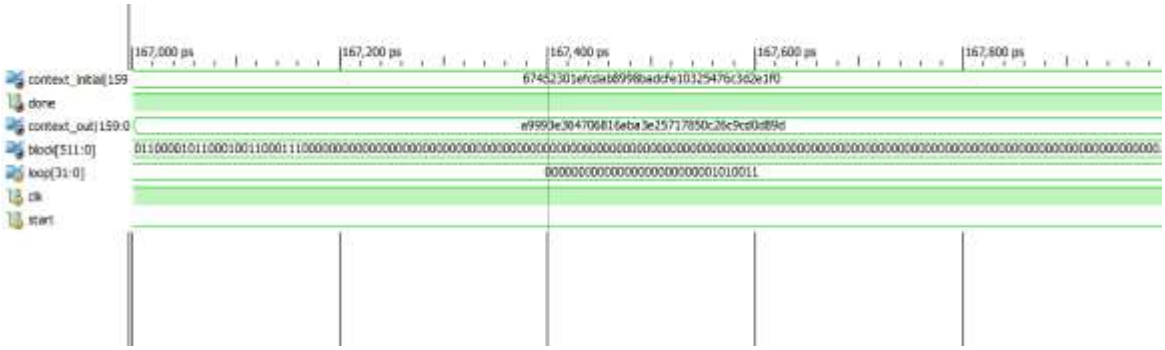

Figure 4. Simulated output waveform

For the given input message "abc", generated an output of 40-digits value represented in hexadecimal format which are located in H0, H1, H2, H3, and H4 buffers. The resulted hash output is combined form of H0-H4 to produce 160-bits fixed length as shown in figure 4. The resulted output is validated using online SHA-1 calculator as shown in the figure 5.

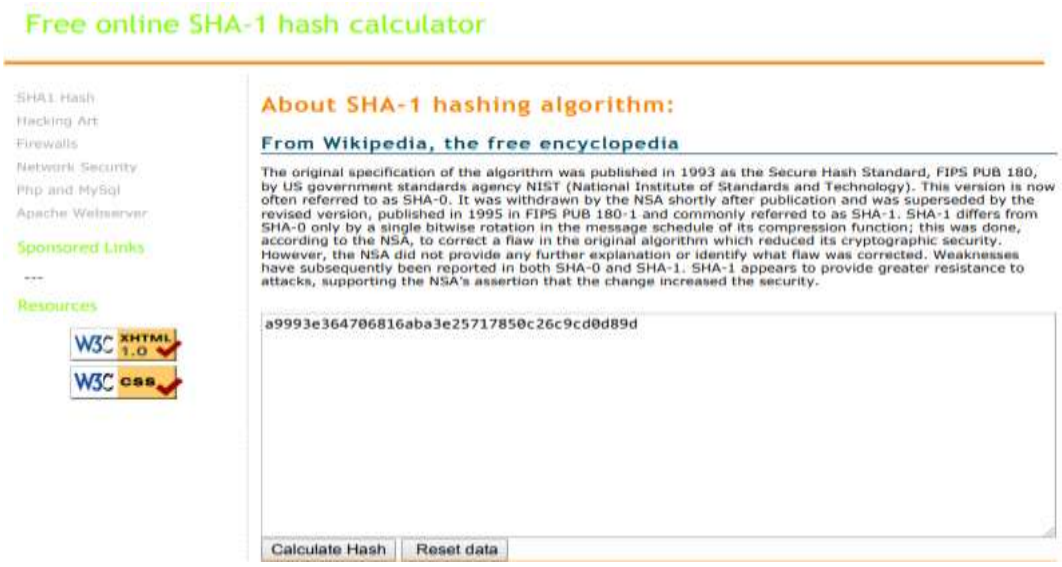

Figure 5. Validation using SHA-1 online calculator

For example, for a change in single character in the input message generates a completely different hash value. Instead of "abc" if "abe" is given as input text it is observed that different message digest obtained.

Table 3. SHA-1 Output

Test Strings	SHA-1
	da39a3ee5e6b4b0d3255bfef95601890afd80709
abe	c9a2a5540dcfc3ebad99e723223675d3f15edc54
message digest	c12252ceda8be8994d5fa0290a47231c1d16aae3
ijklmnopqrstuvwxyzABCDEFGHIJKLM NOPQRSTUVWXYZ012	ed8b55273b7180b9a64b763ffd802939834d6d6c

The design is functionally tested and synthesized using Xilinx ISE Design tool 14.7. Overall architecture is coded in Verilog HDL and test input is applied in test bench. When the output digest is same with the online hash calculator value, the input message is known to be authenticated. Considering the large input message is considered say 55 characters of the string. Although the input message is large it generates fixed length 40 digit (160-bit) output.

The output is verified using online SHA-1 generator. The input is entered in online generator and selects the option "calculate hash", hash value is generated that is exactly same hexadecimal string generated using Xilinx ISE simulation.

By these observations it is said that irrespective of the input message using hash algorithm, output produces a fixed length message digest.

CONCLUSION

In this paper, the Secure Hash Algorithm-1 is implemented with high throughput of 535.82 Mbps. The same is simulated and synthesized in Xilinx ISE design tool using Spartan-3E. The design summary specifies the optimized use of slices and slice flip flops, which achieves high speed design with maximum frequency of 85.815MHz. Verilog code is written for the algorithm which produces a fixed length output for any variable length input message. The implemented output is validated using online SHA-1 hash calculator available in the internet which offers the hash values for all the given inputs. The algorithm in this paper is designed for serial interfacing provides great flexibility. Compared to previous implementations the proposed architecture occupies less area and high speed. The hash algorithm in this paper can be used for verifying data and file integrity, protecting passwords, IP information, FPGA bit stream encryption, and various applications.

FUTURE SCOPE

In the present work, the implementation of the SHA-1 algorithm is used to achieve an optimized and high throughput by minimizing the delay. Presently the design is functionally tested and synthesized using Spartan-3E FPGA board. There is a scope to reduce the number of bonded IOBs, reduce the logic utilization further to increase the speed of the circuit. This can be further modified to implement in any of the Xilinx FPGA devices. This algorithm can be implemented in MATLAB software using HDL coder further achieving high throughput.

REFERENCES:

1. Ankit Anand and Pushkar Praveen, "Implementation of SHA on FPGA", *Int. J. Comput. Sci* (IJCSE)., E-ISSN: 2347- 2963, 2015; 3(5).
2. https://en.wikipedia.org/wiki/SHA-1.
3. Stallings, W. *Cryptography and Network Security, 4/E*. Pearson Education India.2006
4. A. P. Kakarountas, H. Michail, A. Milidonis, C. E. Goutis, and G. Theodoridis, "High-speed FPGA implementation of secure hash algorithm for IPSec and VPN applications," *J. Supercomput.*, 2006; 37(2): 179–195,
5. Parelkar, M. M. FPGA security–bitstream authentication. *Presentation Slides, available at http://mason. gmu. edu/~ mparelka/reports/FPGA_Security. ppt*. 2005.