International
Journal
Of Advanced Research in Engineering & Management (IJAREM)
ISSN: 2456-2033 || PP. 12-17

Performance Analysis of Cryptographic Hash Function Using SHA-256

Ms. Shalini Patel
(M Tech GGITS, Jabalpur, Madhya Pradesh)

Mr. Sunil Shah
(Ast. Prof. GGITS, Jabalpur, Madhya Pradesh.)

Abstract: Cryptographic hash functions are used to protect information and authenticity in a wide range of applications. Providing security to data and information which is to be transferred to and fro has its increasing demand where various remedy measures are to be undertaken for that.
To enhanced security it is ensured by providing out some cryptographic algorithms. The ultimate objective of this project is to develop a FPGA based cryptographic hash algorithms which are being employed in network security. The light is thrown mainly on SHA-2 algorithm and also comparisons are examined between with some other cryptography techniques. The proposed result obtained with the comparative analysis of the Above mentioned methods of Cryptography. A few among these were MD5 (Message Digest Algorithm), SHA (Secured Hash Algorithm) and so on.
In this paper, we investigate high speed and low area hardware architectures of SHA-2 Algorithm. Comparative performance results of the proposed high-speed designs indicate a throughput improvement as compare to others. Additionally, we propose a compact implementation of SHA-2 with pipelined architecture. Measurements reveal a minimal power dissipation which suggests that SHA-2 is suitable for resource-limited systems.
Keywords: Cryptographic hash functions, SHA-2, VLSI implementations, low-power, latch memory

I. INTRODUCTION

Cryptography means crypto- security; graph- writing. It conceals the context of some message which can be reveals by the sender and the recipient. With the insurance of authentication and verification of correctness of the message to the recipient. Through cryptography we encompassing the principles and methods of transforming an intelligible message into one that is unintelligible and then retransforming that message back to its original form.

Data Security is the main aspect of secure data transmission over any network. Today data Security is a challenging issue of data communications in many areas including secure communication channel, strong data encryption technique and trusted third party to maintain the database. The rapid development in information technology, the secure transmission of confidential data herewith gets a great deal of attention.

Cryptographic algorithm SHA-2 is currently used e-commerce and financial transactions, which have strong security requirements. Thus, the necessity to protect the hardware implementation against the Cryptographic attacks [12-14].

In this paper, we propose a reconfigurable design for the SHA-256 algorithms. We present its details implementation. The organization of this paper is as follows. Section II describes the related background knowledge. Section III presents describes the SHA-256 algorithms. Section IV presents the proposed a reconfigurable design for the SHA-256 algorithm. Experimental synthesis results show that the proposed architecture achieves a high performance in term of area, frequency and throughput.
The conventional methods of encryption can only maintain the data security. The information could be accessed by the unauthorized user for malicious purpose. Therefore, it is necessary to apply effective encryption/decryption methods to enhance data security. An algorithm for transforming an intelligible message into an unintelligible one using a code-book

Hash Functions

Hash functions are used as a building block in various cryptographic applications. The most important it uses as a tool for digital signature schemes for the protection of information authentication. A hash function is a function that maps an input of arbitrary length into a fixed number of output bits, the hash value. Hash functions can be divided into two basic categories:

- *One way hash functions*: these functions should be pre image and second pre image resistant, that is it should be hard to find a message with a given hash (preimage) or that hashes to the same value as a given message (second preimage).
- *Collision resistant*: it is one-way hash function for which it is hard to find two distinct messages that hash the same value.

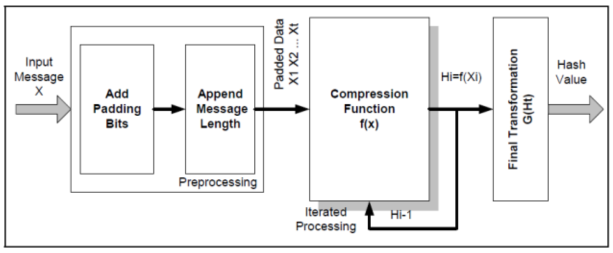


Figure 1: General Model of Hash Function

Most hash functions are designed to operate as iterative processes which hash input messages of arbitrary length. These functions process on fixed-size blocks of the input and produce a hash value of specified length (Fig. 1). The procedure is divided to pre-processing, compression and final transformation.

The pre processing mainly appends the necessary number of bits to the input message, in order to generate the padded data block of specified length. The padded data are divided to t blocks of equal length. Each block Xi serves as input to the compression function h, which computes each time a new transformed data message Hi, as a function of the previous Hi-1 and the input Xi. After a certain number of processing rounds, the data are finally modified by the final transformation. In this way the hash value (message digest) is generated corresponding to the input message x.

The proposed architecture guarantees high security level, in all the applications requiring message authentication, via the construction of a message authentication code. The security strength and the advantages of the SHA-2 hash function that the proposed architecture is based on, ensures high security level, in the implementation of this authentication scheme Hash function 2 are cryptographic algorithms that take as input a message of arbitrary length, and that return a digest (or hash value) of fixed length (between 160 and 512 bits, in most applications). Hash functions are used in a multitude of protocols be it for digital signatures within high-end servers, or for authentication of embedded systems. Proposed design is a family of hash functions with internal state sizes: 256.

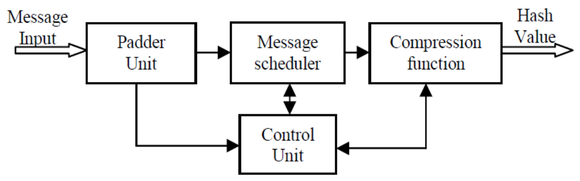
- Proposed Design-256 is our primary proposal.
- Proposed Design-256 is our low-memory variant.
- This allows the design to hash configuration data along with the input text in every block, and make every instance of the compression function unique.
- This property directly addresses many attacks on hash functions, and greatly improves Proposed Design's flexibility.

Efficient FPGA Hardware Implementation of Secure Hash Function SHA-2 Using VHDL Language to improve its performances in terms of **area**, frequency or throughput.

In this Thesis, we proposed a new design for the SHA-256 functions. Moreover, the proposed design has been implemented on Xilinx Virtex-6 FPGA. Its area, frequency and throughput have been compared and it is shown that the proposed design achieves good performance in term of area with a bit compromise in speed.

ALGORITHM SPECIFICATION

In previous year research paper several hardware optimization techniques for the SHA-2 hashing functions were explored. A new architecture that is Round Pipelined Technique was proposed for the SHA-2 core, which eliminates the data dependency between iteration using data forwarding to improve the throughput per area. The fully iterative and Round Pipelined Techniques were investigated and developed using HDL. Implementation result indicate that the Round Pipelined technique can help to achieve good tradeoff between throughput and area. Proposed research investigates optimization techniques in terms of area and resources for SHA-2 hash functions on the FPGA and achieves higher stable circuit with lowest number of hardware used hence increases the power efficiency although speed is bit slow.



Simplified architecture of the SHA-2 algorithm

SHA-2 has two main versions: SHA-32 and SHA512-64. This section gives a brief specification of these algorithms. A complete specification can be found in [7]. The BLAKE-32 algorithm operates on 32-bit words and returns a 256-bit hash value. It is based on the iteration of a compression function, described below.

1) Compression Function: Henceforth we shall use the following notations: if m is a message (a bit string), mi denotes its i-th 16-word block, and mij is the j-th word of the i-th block of m. Indices start from zero, for example a N-block message m is decomposed as

m= m0,m1,m2........ mN-1 and

the block m0 is composed of words .

m0= m01 ,m0 2,m03........ m015

Idem for other bit strings. Endianness conventions are described in [7]. The compression function of SHA-256 takes as input four values:

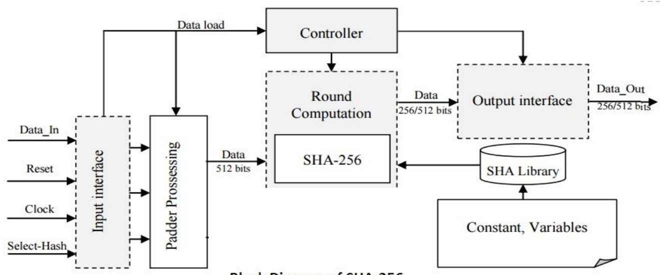
- a chaining value h = h0, . . . , h7.
- a message block m = m0, . . . ,m15.
- a salt s = s0, . . . , s3.
- a counter t = t0, t1.

These inputs represent 30 words in total (i.e., 960 bits). The salt is an optional input for special applications, such as randomized hashing [11]. The output of the compression function is a new chaining value h' = h'0, h'1,........ h'7. of eight words (i.e., 256 bits). We write h' := compress(h,m,s,t).
The compression function compress() can be decomposed into three main steps
a) Initialization: b) Round Function: c) Finalization:

2) Hashing a Message: When hashing a message, the function starts from an initial value (IV), and the iterated hash process computes intermediate hash values that are called chaining values. Before being processed, a message is first padded so that its length is a multiple of the block size (512 bits). It is then processed block per block by the compression function, as described below:

```
h0 := IV
for i = 0, . . . ,N − 1
hi+1 := compress(hi,mi,s,li)
return   hN
```

Here, li is the number of message bits in m0,m1,........mi, that is, excluding the bits added by the padding. It is used to avoid certain generic attacks on the iterated hash (e.g., [12]). The salt s is chosen by the user, and set to zero by default.



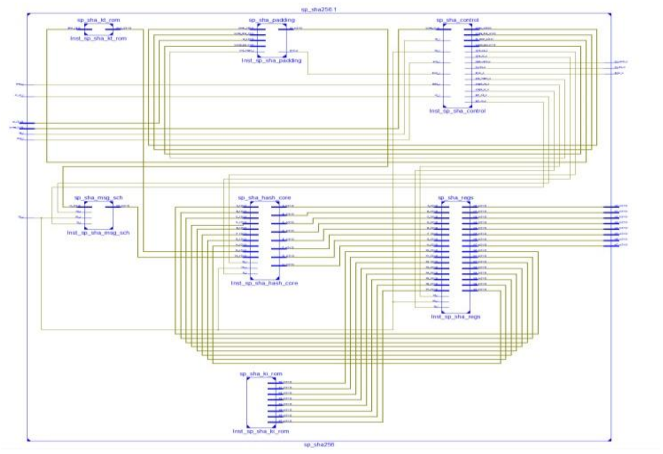
Block Diagram of SHA-256

VLSI Implementation of SHA-2

The aim is to implement the designed hash function core on VHDL. The whole package and separate modules were synthesized and analyzed using Xilinx ISE 12.1 tool for the targeted Virtex-VI FPGA.

The VHDL implementation was divided into five modules:

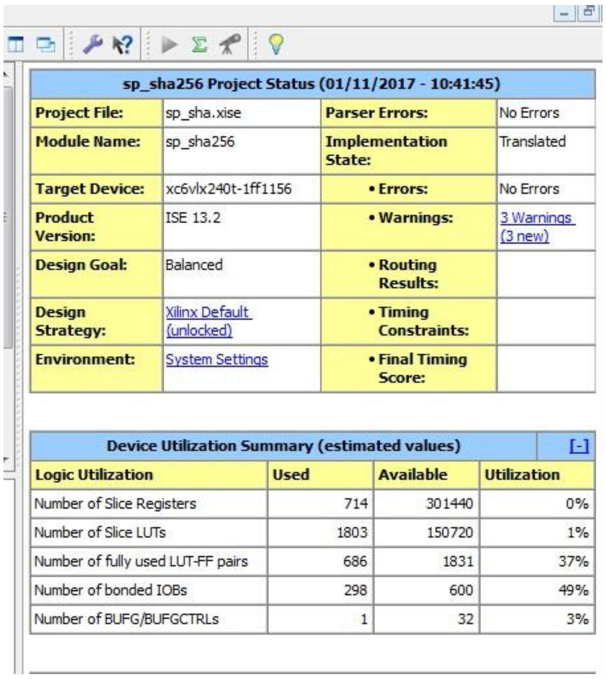
- *Initial module:* **-** It collects the serial input bits and sends 512 bit blocks to the next module.
- *Round module:* **-** It performs the hashing calculations and operations on the input message block and previous hash output to generate a new hash value.
- *Last Block module:* **-** At the end of the message bit stream the final message block of 512 bits has to be prepared by adding 64 bits of message length at the end of 448 bits of input message block, padded accordingly to suffice the word size requirement. This final message block does this function of preparing the last message block.
- *Final module:* **-** This module computes the hash value by adding the previous hash value to the new hash value achieved from the Round module. Then it sends the 256 bit hash value, bit by bit (serially).
- *Top module:* **-** This module is the control unit for controlling the functioning of the rest of the modules and to ensure that the SHA-2 algorithm flow is followed and maintained


Complete Top Level Logic Design of SHA-256

Result

• For testing the effectiveness and efficiency of the proposed design a performance comparison has been made in terms clock frequency, latency, area (gate equivalents) and throughput with the existing competitors of same bit size. The table below depicts the comparison.

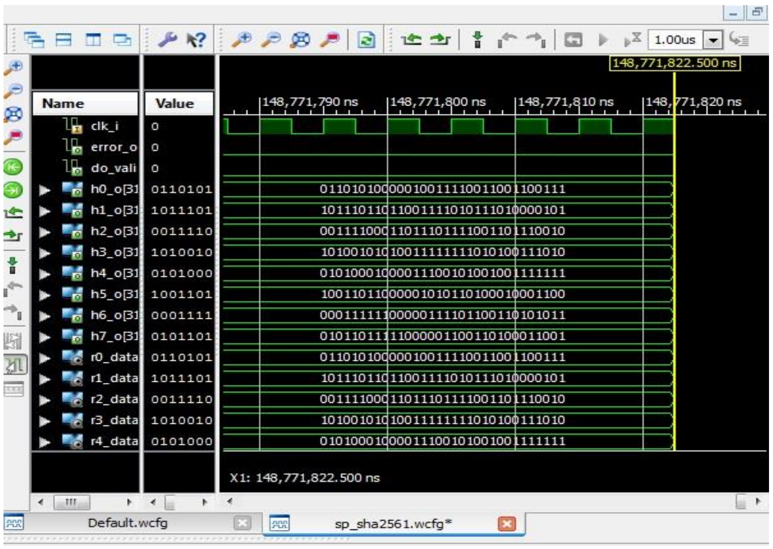
• The proposed design gives better performance. The table below shows the effectiveness of our design. In the next chapter we have given the snapshots of RTL logic and their simulation waveforms.



sp_sha256 Project Status (01/11/2017 - 10:41:45)			
Project File:	sp_sha.xise	**Parser Errors:**	No Errors
Module Name:	sp_sha256	**Implementation State:**	Translated
Target Device:	xc6vlx240t-1ff1156	• **Errors:**	No Errors
Product Version:	ISE 13.2	• **Warnings:**	3 Warnings (3 new)
Design Goal:	Balanced	• **Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	
Environment:	System Settings	• **Final Timing Score:**	

Device Utilization Summary (estimated values) [-]			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	714	301440	0%
Number of Slice LUTs	1803	150720	1%
Number of fully used LUT-FF pairs	686	1831	37%
Number of bonded IOBs	298	600	49%
Number of BUFG/BUFGCTRLs	1	32	3%

Summery Report



```
148,771,822.500 ns
Name        Value      |148,771,790 ns   |148,771,800 ns   |148,771,810 ns   |148,771,820 ns
clk_i       0
error_o     0
do_vali     0
h0_o[31     0110101    01101010000010011110011001100111
h1_o[31     1011101    10111011011001111010111010000101
h2_o[31     0011110    00111100011011101111001101110010
h3_o[31     1010010    10100101010011111111010100111010
h4_o[31     0101000    01010001000011100101001001111111
h5_o[31     1001101    10011011000001010110100010001100
h6_o[31     0001111    00011111100000111101100110101011
h7_o[31     0101101    01011011111000001100110100011001
r0_data     0110101    01101010000010011110011001100111
r1_data     1011101    10111011011001111010111010000101
r2_data     0011110    00111100011011101111001101110010
r3_data     1010010    10100101010011111111010100111010
r4_data     0101000    01010001000011100101001001111111

X1: 148,771,822.500 ns
Default.wcfg    sp_sha2561.wcfg*
```

Simulation Table

CONCLUSION

Along with an optimal security strength the future cryptographic hash standard SHA-2 should be suitable and flexible for a wide range of applications. In this work we presented a complete hardware characterization of the SHA-2. A round rescheduling technique and a special-purpose memory design are also proposed. I believe that a similar approach for compact VLSI implementations of cryptographic protocols is a valuable choice to reduce the area and power consumption of the integrated circuit. Post-synthesis results , a low-power compact implementation of SHA-2 has been Implemented. The wide spectrum of achieved performances paves the way for the application of the SHA-2 function to various hardware implementations.
Concluding remarks are,

- ‘Lightweight’ is the rising star of cryptography. The term ‘lightweight’ alone covers a very wide range, such as lightweight in terms of area, speed, power consumption, energy consumption, or a combination of these, depending on the specific application.
- This research solely concentrates on the lightweight for area, which also results in lightweight for average power consumption in most applications.
- The use of block memories is avoided for compatibility on different platforms.
- We have been successful in reaching our target of lowest gate count, and even managed to surpass some of the recently proposed lightweight hash functions in terms of compactness and throughput.

REFERENCES

[1]. Manoj D Rote, Vijendran N, David Selvakumar “High Performance SHA-2 core using the Round Pipelined Technique “ published in ieee 2015

[2]. Roar Lien, Tim Grembowski, and Kris Gaj “A 1 Gbit/s Partially Unrolled Architecture of Hash Functions SHA-1 and SHA-512” published in IEEE 2004.

[3]. Ricardo Chaves , Georgi Kuzmanov , Leonel Sousa , Stamatis Vassiliadis “Improving SHA-2 hardware implementations, Proceedings of the 8th international conference on Cryptographic Hardware and Embedded Systems” published in October 2006.

[4]. Algredo-Badillo , C. Feregrino-Uribe , R. Cumplido , M. Morales-Sandoval “FPGA-based implementation alternatives for the inner loop of the Secure Hash Algorithm SHA-256, Microprocessors & Microsystems” published in August, 2013.

[5]. Rommel García , Ignacio Algredo-Badillo , Miguel Morales-Sandoval , Claudia Feregrino-Uribe , René Cumplido “A compact FPGA-based processor for the Secure Hash Algorithm SHA-256” Computers and Electrical Engineering, published in January, 2014.

[6]. Ryan Glabb , Laurent Imbert , Graham Jullien , Arnaud Tisserand , Nicolas Veyrat-Charvillon “Multi-mode operator for SHA-2 hash functions” published in February, 2007.

[7]. Marcin Rogawski, Kris Gaj, "A High-Speed Unified Hardware Architecture for AES and the SHA-3 Candidate Grøstl", Digital System Design (DSD) 2012 15th Euromicro Conference on, pp. 568-575, 2012

[8]. Cheng-Fu Chou , William C. Cheng , Leana Golubchik “Performance study of online batch-based digital signature schemes” Journal of Network and Computer Applications, v.33 n.2, p.98-114, March, 2010

[9]. Dan Cao, Jun Han, Xiao-yang Zeng, Shi-ting Lu, "A core-based multi-function security processor with GALS Wrapper", Solid-State and Integrated-Circuit Technology 2008. ICSICT 2008. 9th International Conference on, pp. 1839-1842, 2008.

[10]. Luminita Scripcariu has proposed “A Study of Methods Used To Improve Encryption Algorithms Robustness” Published On IEEE in July 2015.