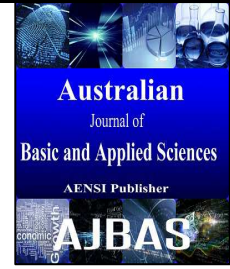




**AUSTRALIAN JOURNAL OF BASIC AND APPLIED SCIENCES**

ISSN:1991-8178 EISSN: 2309-8414  
Journal home page: www.ajbasweb.com



**Low Power Implementation of Modified Hash Algorithm in Asic**

A.Veera Lakshmi, M. Ashok Kumar, A. Kanaga Lakshmi and A. Mahadevan

*Asst.Prof, Dept of Electronics and Communication Engineering Sree Sastha Institute of Engineering and Technology, Chennai,India*

**Address For Correspondence:**

A.Veera Lakshmi, Asst.Prof, Dept of Electronics and Communication Engineering Sree Sastha Institute of Engineering and Technology, Chennai. India.  
E-mail: veerajyo.lakshmi@gmail.com

**ARTICLE INFO**

**Article history:**

Received 10 December 2015

Accepted 28 January 2016

Available online 10 February 2016

**Keywords:**

SHA-512; FPGA; Permutation;  
Digital Signature.

**ABSTRACT**

In this paper a new method is proposed for generating digital signature based on SHA-512 hash algorithm. This method uses a secret key and enjoys the benefits of the private key cryptography. Two SHA-512 modules are connected in parallel for generating two 512-bit temporary signatures, signatures are permuted by the given secret key and generates 1024-bit signature. The efficiency of the algorithm is synthesized and verified using QUATRTRUS II ALTERA by applying multiple scenarios.

**INTRODUCTION**

Data integrity assurance and data origin-authentication are more essential security services in financial transactions, electronic commerce, electronic mail, software distribution, data storage and so on. The broadest definition of authentication within computing systems includes identity verification, message origin-authentication and message content authentication (Chaves, R., 2006). To achieve the required processing capabilities, using hardware components seems necessary. These hardware cores are being implemented either in dedicated ASIC cores or in reconfigurable devices (Grembowski, T., 2002). In this paper ,Xilinx reconfigurable devices are used to implement the proposed design.

By using private key security is increased, attackers cannot generate messages offline since they don't know the secret key (William Stallings, 2005). Thus, this method is much more secure than SHA-512 facing birthday attacks. In this approach, two SHA-512 modules are utilized for generating two 512 bit signatures. Then, using the secret key it is permuted and generates a 1024-bit signature. This paper is organized as follows; Section 2 presents implementation of a SHA-512 module with FPGA. Section 3 describes the proposed design and its implementation details. In Section 4 analyzing the performance of this method and finally in section 5, the security of the proposed method and compare it with the SHA-512 approach is studied

**II. Implementation Of Sha-512:**

In 1993 the Secure Hash Standard (SHA) was first published by the NIST. In 1995 this algorithm was reviewed in order to eliminate some of the initial weakness, and in 2001 new Hashing algorithms were proposed which are called as SHA-2.It uses larger digest messages which are more resistant to possible attacks and allocate them to be used with larger blocks of data up to 2.128kilo bits, in case of SHA-512 (NIST, 2002).

SHA-512 structure is implemented by dividing in to e smaller modules to reduce the difficulty of implementing it in Verilog description language.

**Open Access Journal**

**Published BY AENSI Publication**

© 2016 AENSI Publisher All rights reserved

This work is licensed under the Creative Commons Attribution International License (CC BY).

<http://creativecommons.org/licenses/by/4.0/>



Open Access

**To Cite This Article:** A.Veera Lakshmi, M. Ashok Kumar, A. Kanaga Lakshmi and A. Mahadevan., □Low Power Implementation of Modified Hash Algorithm in Asic. *Aust. J. Basic & Appl. Sci.*, 10(1): 110-116, 2016

Each module consist of

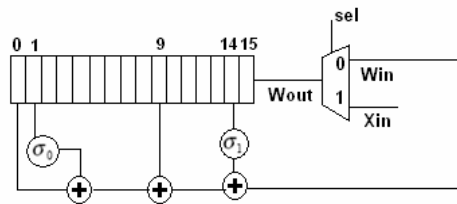
- Data path
- ❖ Message Scheduler
- ❖ K<sub>j</sub> memory
- ❖ Round
- controller

The data path issues the flow of data .The controller comprising of a state machine which controls the data flow in data path. Description of each part is as follows:

**Message Scheduler:**

It generates 80 message dependent words  $W_t$ . Among that the first 16 words are as usual considered as the first 16 words of the input message block. Remaining words are calculated using basic feedback performance based on rotations, shifts and XOR operations.

It receives a 1024-bit message block in 64-bit piece through  $X_{in}$ . Based to Fig.1 this module has a 16×64-bit shift register and one multiplexer. During the first 16 pulses of the system clock, whole message block enters into the shift register after that the multiplexer selects  $W_{in}$  to provide feedback for the shift register. Message Scheduler generates  $W_i$  for the hash algorithm in each clock pulse and  $W_{out}$ , the output signal of the module, is the  $W_i$  at Round<sub>*i*</sub>.



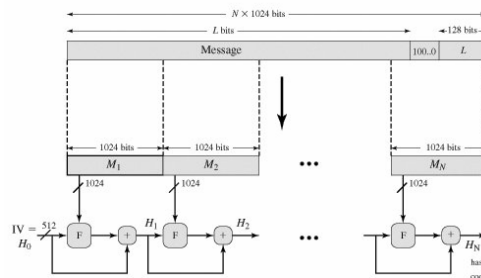
**Fig. 1:** Message Scheduler Block Diagram.

**K<sub>j</sub> Memory:**

Based on the NIST Standard, SHA-512 utilizes the sequence of eighty constant 64-bit words,  $K_0, K_1, \dots, K_{79}$ . These words represent the first sixty-four bits of the fractional parts of the cube roots of the first eighty prime numbers. The constant values are stored by using an 80×64-bit Xilinx specific Block ROM is used.

**Round:**

The standard hash algorithm for each message block runs a specific round for 80 times and it generates a 512-bit output which is an starting value of the algorithm for the next message block. After processing of the all 1024-bit message blocks, the last 512-bit output is the signature of message. Fig. 2 represents the message digest generation using SHA-512 and Fig. 3 describes the processing of single 1024 bit block.



**Fig. 2:** Message digest generation using SHA-512.

The following are the ways to implement rounds architecture: full loop unrolling and iterative looping (McEvoy, R.P., 2006; Janaka Deepakumara, 2001). The full loop unrolled architecture has an 80-step combinational logic. It requires huge amount of area beside the complexity of generating  $W_i$ . The SHA-512 algorithm is implemented the looping architecture with 80 iterations which will provide the most area-efficient solution in the first step of implementation. The block diagram of an iterative core is shown in Fig.4. At the first round, Register1 is the initial value (IV) vector which is a 512-bit constant number.

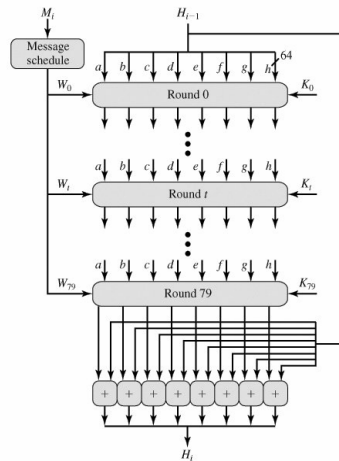


Fig. 3: SHA-512 processing of a single 1024-bit block.

Register1 holds the \$H\_i\$ of the previous round (\$H\_{i-1}\$) after the first round. The round is repeated upto 80 times and it is done by the Mux2 multiplexer which returns the output feedback of each round to the Round block. Fig.4 shows the Iterative core block diagram. The output achieved by each round is stored in ABCDEFGH register to prepare feedback value for the next round. After 80 rounds, digest of one 1024-bit message block becomes ready at \$H\_i\$.

**Controller:**

This module generate all necessary controlling signals such as multiplexer’s selectors, registers’ latch enables, ROM addresses (for \$K\_j\$) and etc. Simulation is done using ModelSIM XEI II 6.1e. In order to validate the simulation results, signatures calculated by Crypto++C library for a given message block is compared with the obtained one.

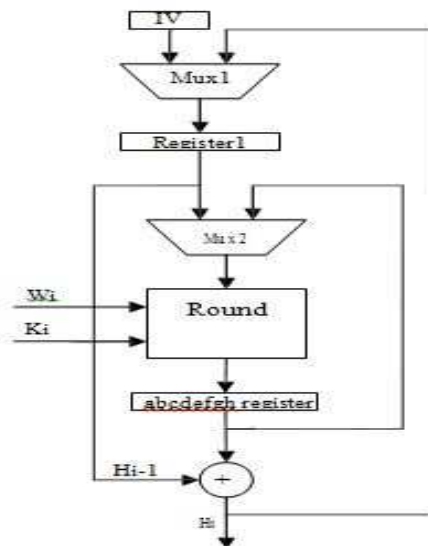


Fig. 4: Iterative core block diagram.

**III. Proposed Method:**

The proposed method is implemented using two SHA-512 modules connected in parallel. A message is conceded to SHA-512 modules in 64-bit chunks, alternatively. There will be two 512-bit temporary signatures for a given message, There is a permutation unit is applied to permute two temporary signatures with the given secret key which is stored in a 1024×10-bit Block ROM. Each word of ROM indicates the position of corresponding bit in output signature. After permutation of the 1024-bit temporary signatures, the final signature becomes ready is shown in Fig. 5. As Fig.6. shows, the structure of a round consists of multiple add operations (eq. 1 and eq. 2) in series that forms the critical path of this structure.

$$E_{t+1} = D_t + \Sigma_1 (E_t) + Ch (E_t, F_t, G_t) + H_t + K_t + W_t \tag{1}$$

$$A_{t+1} = \Sigma_0(A_t) + \text{Maj}(B_t, C_t, D_t) + \Sigma_1(E_t) + \text{Ch}(E_t, F_t, G_t) + H_t + K_t + W_t \tag{2}$$

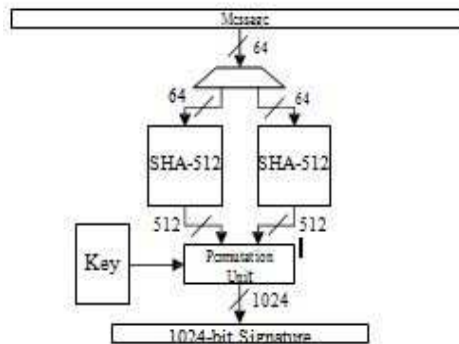


Fig. 5: Combined SHA-512.

Carry Save Adder (CSA) (Behrooz Parhami, 2000) is preferred in the place of Carry Propagate Adder (CPA) blocks for calculating  $A_{t+1}$  and  $E_{t+1}$  are shown in Fig 7 which is known for its best-parallelizable architecture. This implementation strategy reduces the critical path delay significantly.

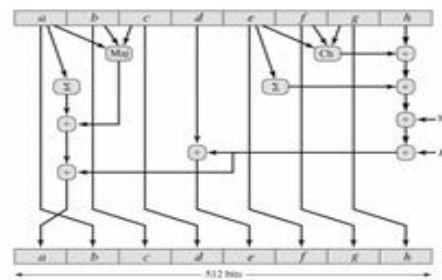


Fig. 6: Single round structure.

The design area is reduced by the shift register mode of CLB<sup>1</sup> slices available in Xilinx Virtex4 FPGA devices in Message Scheduler implementation in place of generic shift registers.

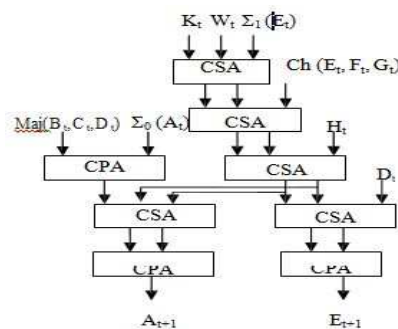


Fig. 7: Calculating  $A_{t+1}$  and  $E_{t+1}$  using CSA blocks.

**IV. Performance Analysis:**

The modules were synthesize, placed and routed on Xilinx Spartan 3e target device. In the case of SHA-512 the utilization of slices was 1921 out of 15360 (12%) and number of consumed slice flip flops was 2240 out of 30720 (7%). The timing analysis shows 118.043 MHz as the maximum frequency of the design. Maximum data throughput can be estimated using the following equation:

$$\text{Throughput} = S_{mb} \times F_{max} / R \tag{3}$$

where:

$S_{mb}$ : message block size

$F_{max}$ : Maximum clock frequency

### R: Number of rounds

Using the eq. 3 maximum expected throughput is:

1.51Gbps ( $1024b \times 118.043\text{MHz} / 80$ ).

For the combined SHA-512 method, without using CSA, the utilization of slices is 3541 out of 15360 (23%) and number of slice flip flops is 4490 out of 30720 (14%) with the maximum clock frequency of 117.845 MHz and maximum throughput of 2.51Gbps. In this model, it takes  $80 + 16$  clock pulses for two 1024-bit message blocks. Extra 16 clock pulses are needed for second SHA-512 message scheduler. This throughput is valid for one message because of the permutation delay (about 8.7us). Permutation can be overlapped by the processing of the next message if its length is at least 13 message blocks. After using carry save adders in combined SHA-512 method, because of the CSA adder overhead, the slice utilization increased to 3826 out of 15360 (24%) and number of used slice flip flops increased to 4528 out of 30720 (14%). However, the maximum clock frequency rose to 135.466 MHz which results in improved throughput of 2.89Gbps with 7.5us permutation delay for each message ( $2.89\text{Gbps} = 2 \times 1024b \times 135.466\text{MHz} / (80 + 16)$ ).

### V. Security Analysis:

The security of hash functions is determined by the size of their outputs, referred to as hash values,  $n$ . The best known attack against these functions, the “birthday attack”, can find a pair of messages having the same hash value with a work factor of approximately  $2^{n/2}$ . Therefore, for a SHA-512 module, complexity of the best attack is  $2^{256}$ . For this method each SHA-512 has the same attack complexity, but here permutation process increases the attack complexity in proportion to the size of the secret key. If we use a 1024-bit secret key, permutation complexity becomes  $2^{256}!$  yields the total complexity of  $2^{10}! \times 2^{256}$  and it's much more secure than the SHA-512 algorithm.

### Conclusion:

In this paper, a new idea is proposed to generate a signature for messages based on the well-known SHA-512 algorithm. (Behrooz Parhami, 2000) The implementation and simulation results show the method superiority over the basic algorithm in throughput and security features. Because of permutation delay, the new method is suitable for long messages (longer than 1.5kB). Also, the numerical results prove the suitability of the FPGAs as the implementation platform for cryptographic accelerators.

### Simulation Results:

All the blocks in the combined SHA 512 are integrated in a top module. The data rate is calculated as: Clock frequency = 118.043 MHz for 64 bit Input, therefore data rate = 1.51 Gbps. All the files are compiled using the software Model Sim and the simulation waveform is obtained as shown in the Fig. 8. The blocks mentioned in the receiver section is compiled separately. Then it is included in the top module and simulation is done. Compilation result is obtained using the software Quartus II.

#### simulation result

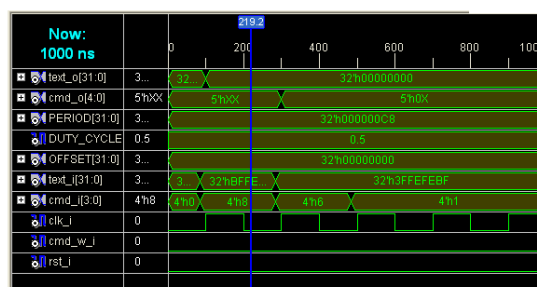


Fig. 8: Simulation result.

The compiled files in the Model Sim are included in this software. Test bench is written separately. Device name and top level entity is included. The compilation report is obtained as shown in the Fig. 9.

Data rate is calculated by using the Time Quest Analyzer. Clock frequency obtained is 118.043 MHz. Therefore the data rate is calculated for 64 bit Input as 1.51 Gbps. The timing waveform is obtained as shown in the Fig. 10.

Total thermal power dissipation is about 139.94mW. The power estimation confidence is about low: user provided insufficient toggle rate data. The power analysis report is shown in Fig.11.

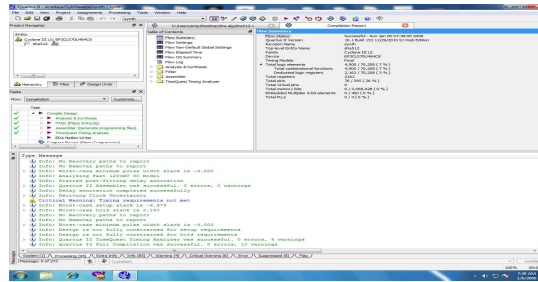


Fig. 9: Compilation report.

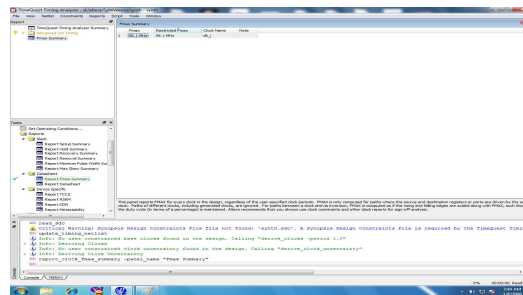


Fig. 10: Timing waveform.

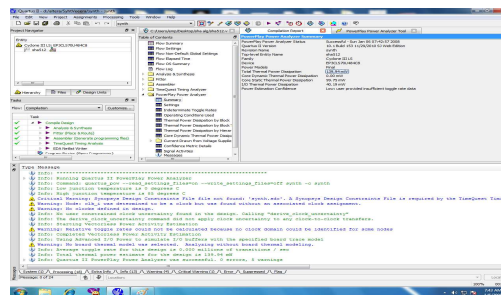


Fig. 11: Power analysis.

RTL shows the schematic representation of input and output. 64 bit Inputs and 512 bit outputs are shown in the Fig.12.

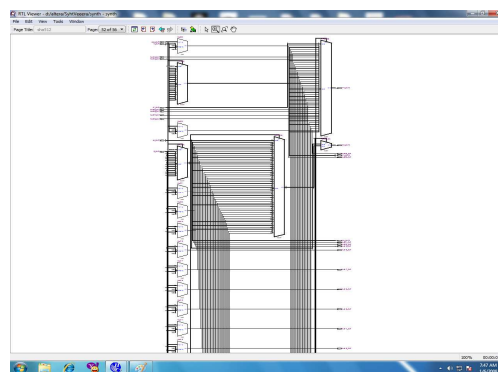


Fig. 12: RTL view.

REFERENCES

Chaves, R., G.K. Kuzmanov, L. Sousa, S. Vassiliadis, 2006. "Improving SHA-2 Hardware implementations", Workshop on Cryptographic Hardware and Embedded Systems (CHES).  
 Grembowski, T., R. Lien, K. Gaj, N. Nguyen, P. Bellows, J. Flidr, T. Lehman, B. Schott, 2002. "Comparative analysis of the hardware implementations of hash functions SHA-1 and SHA-512", in ISC (A. H. Chan and V. D. Gligor, eds.), 2433 of Lecture Notes in Computer Science, 75-89, Springer.

William Stallings, 2005. "Cryptography and Network Security Principles and Practices, Fourth Edition", Prentice Hall.

McEvoy, R.P., F.M. Crowe, C.C. Murphy, W.P. Marnane, 2006. "Optimisation of the SHA-2 family of hash functions on FPGAs", IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures ISVLSI'06).

Janaka Deepakumara, Howard M. Heys, R. Venkatesan, 2001. "FPGA Implementation of MD5 Hash Algorithm". In Canadian Conference on Electrical and Computer Engineering (CCECE).

NIST, 2002. "Announcing the Standard for Secure Hash Standard, 180-2", tech. rep., National Institute of Standard.

Behrooz Parhami, 2000. "Computer Arithmetic, Algorithms and Hardware Design", Oxford University Press.