NVEO
Natural Volatiles & Essential Oils

Design and FPGA implementation of folded SHA-256 using 4-2 adder compressor

P.Pavithara[1*], R.Renuka[2], P.Sabena Yasmin[3] and K.Naresh[4]

Assistant Professor, Department of Electronics and Communication Engineering,
Kongu Engineering College, Perundurai - 638060, Tamil Nadu, India[1].
UG student, Department of ECE,Kongu Engineering College, Perundurai - 638060, Tamil Nadu, India[2,3,4].
e-mail : ppavithara1993@gmail.com[1]

Abstract

The Secure Hash Algorithm is a cryptographically secure hash function that enables high security in electronic systems. SHA-256 refers to the output length of 256 bits. The folded SHA-256 algorithm is a one-way functional method that is used to represent data in a compact manner. It is utilized in a variety of authentication techniques including digital signal creation, password storage, computer vision and database access. SHA is a four step process involving pre-processing, message scheduling, digest calculation and digest update. In the partial product accumulation stage, the architecture of 4-2 adder compressor in SHA-256 ensures an improvement in computing time by lowering the number of adders. Architectural folding is used for maximum resource sharing between the stages. The delay encountered for the entire system is 2.594 ns and power consumption is 177mW. The proposed design successfully attained the throughput efficiency by reducing the critical path delay.

Keywords: SHA-256, hash function, adder compressor, architectural folding.

I. Introduction

Cryptographic algorithms like hash algorithms are developed to assure security. To perform wide range of real time operations, there needs a vast usage of digital communication systems with security. Hash functions are used to create cryptographic hashes also known as digital signatures. Hash algorithms are used to store passwords. One of the hashing algorithms is SHA (Secure Hash Algorithm). SHA-0, SHA-1, SHA-2

and SHA-3 are the several forms of SHA. SHA-2 is subdivided further into SHA-224, SHA-256, SHA-512 and SHA-384. The SHA-1 hash function can be created by the process of loop unfolding, pre-processing and multi input addition based on carry save adder [1] and it involves security issues. SHA-2 was developed to overcome the threats posed by SHA-1. The SHA-2 algorithm provides higher security than previous hash functions [2]. SHA2 includingSHA-256, 384, and 512 can be implemented in a phased manner [3]. Though the development of hash algorithm like SHA-3 is improved in recent years, the need for SHA-256 has a great impact over throughput performance and speed.SHA-256 do not have known vulnerabilities and it is unbroken. It is used to secure password hashing in Unix and Linux.

SHA being a one-way hash function, inverting a hash value for an input message is impossible. Furthermore, finding a message that generates the same hash value is impossible. These characteristics became vital in maintaining the appropriate operation of a hash function [4]. The two components of the SHA-256 architecture are the expander and compressor. These components use pipeline concept to implement both the architectures individually [5]. Pipelining involves long delay due to numerous calculations in the compressor block. Thus, architectural folding is proposed to reduce the delay.

The hardware encryption is preferred for military and commercial applications [6]. The hardware design for SHA-256 algorithm was focused on reusing data and minimizing critical paths. The SHA-256 processor was developed with a simplified arithmetic and logic unit that would result in a compact design [7, 8].

It can also be built using shift registers that require less hardware [9, 10, 11]. The hash architecture of SHA-256 is becoming very frequently used as computation has been rescheduled to take advantage of hardware capabilities. In round functions, three pipelines are employed to replace the critical path, allowing the computation chain to be broken into different parts and the long critical path to be shortened [12, 13, 14, 15].

The implementation of SHA-256 hash function at high speed exceeded throughput limit of the hash function [16]. The folded architecture of SHA-256 results in reduction of logic gates on implementation [17, 18]. The XOR-XNOR and MUX modules can be utilized for the design of 4-2 adder compressor [19, 20]. An effective hardware implementation of SHA-256 is difficult to design because of long critical path. As a result, the use of architectural folding with a 4-2 adder compressor is presented as a new architecture optimization methodology[21-24]. The proposed system improves resource sharing in the hardware circuit which reduces the area. This is achieved by sharing same unit to perform two or more computations in the system. The 4-2 adder compressor has the advantage in reducing the clock cycles.

II. Methodology

A. Hashing process

The hashing process consists of four stages as shown in Fig.1. The algorithm starts with pre-processing of input message, message scheduling of pre-processed data, digest calculation of the scheduled message and ends with digest update of the message. In pre-processing, the input message of 'n' bits length is appended with 1 followed by 'k' zero bits for a single character. A character represents 8 bit binary information. An 8 bit input is converted to 32 bit pre-processed data. The total length of the data should be multiples of 512, since the data is processed into 512 bits block as a whole. The bit appending process is known as padding and the expanded data is given to the next stage of scheduling.

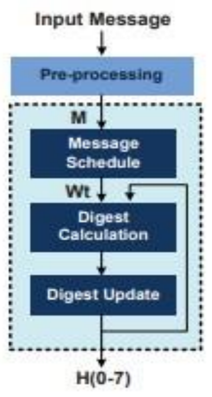

Fig.1Hashing process

The thirty-two bit data from 8 bits extending to the scheduling stage is processed with 64 iterative loops. Each of the iteration involves a scheduled value of 7 bits and each scheduled value has its unique 32 bit register value formed by right shift and right rotation operations given by equation 1.

$$W(t) = \begin{cases} M_t, & 0 \leq t \leq 15 \\ \sigma 1(W_{t-2}) + W_{t-7} + \sigma 0(W_{t-15}) + W_{t-16}, & 16 \leq t \leq 63 \end{cases} \quad (1)$$

$\sigma 0(x) = ROTR7(x) \oplus ROTR18(x) \oplus SHR3(x)$ and

$\sigma 1(x) = ROTR17(x) \oplus ROTR19(x) \oplus SHR10(x)$

where ROTR denotes a circular rotation to the right, SHR is shifting right, M represents the message bits, σ combines rotation and shift operations and W represents 32 bit words.

The core of the hash algorithm is digest calculation stage. For each scheduled value and register value, the 32 bit values of eight variables A, B, C, D, E, F, G and H are created and iterated to 64 loops respectively. The variables are initialized in the first iteration and calculated like the hash values from H0 to H7. It is governed by a set of expressions given by equation 2:

$T1 = H + \Sigma 1(E) + Ch(E, F, G) + Kt + Wt$
$T2 = \Sigma 0(A) + Maj(A, B, C)$
$A = T1 + T2 \qquad E = D + T1$
$B = A \qquad F = E$
$C = B \qquad G = F$
$D = C \qquad H = G$
and $Ch\{x, y, z\} = (x \wedge y) \oplus (\bar{x} \wedge z)$
$Maj\{x, y, z\} = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$
$\Sigma 0(x) = ROTR2(x) \oplus ROTR13(x) \oplus ROTR22(x)$
$\Sigma 1(x) = ROTR6(x) \oplus ROTR11(x) \oplus ROTR25(x) \quad (2)$

where H denotes the initiated hash variable, Ch and Maj represents EXOR operation, Σ represents rotation operations and A,B,C,D,E,F,G and H are the eight variables of hash registers.

The final stage of the algorithm is digest update. The intermediate hash values of the given data are calculated for a single character after the completion of 64 iterations. The data processed with all these stages result in digest message of 256 bits obtained by the concatenation of hash values as given by equation 3:

$$H_0^i = A+H_0^{(i-1)}$$

$$H_1^i = B+H_1^{(i-1)}$$

$$H_2^i = C+H_2^{(i-1)}$$

$$H_3^i = D+H_3^{(i-1)}$$

$$H_4^i = E+H_4^{(i-1)}$$

$$H_5^i = F+H_5^{(i-1)}$$

$$H_6^i = G+H_6^{(i-1)}$$

$$H_7^i = H+H_7^{(i-1)}$$

$$H^N = H_0^N \,||\, H_1^N \,||\, H_2^N \,||\, H_3^N \,||\, H_4^N \,||\, H_5^N \,||\, H_6^N \,||\, H_7^N \quad (3)$$

B. Architectural folding

Folding refers to reduction in the number of operations that is folded to a single unit. It is an architectural transformation technique different from unrolling process. The longest path between the input and output is referred to as critical path. Architectural folding reduces the critical path of the SHA-256 algorithm by enhancing sharing of resources in the hardware circuit. The functional units in a digital system can be minimized using folding technique that uses a factor K. The folding factor of 2 is set to reduce the number of adders in the SHA-256 circuit to a minimum. The folding process forces a substantial timing overhead in hashing process which would increase the clock cycles linearly to the corresponding folding factor. The rescheduling done on the longest critical path has two additions in the new path instead of seven additions. In hash variable calculation, the sequence of additions is arranged into mutually distinct clock cycles. The SHA-256 critical path is rescheduled, and pipelining is achieved by inserting registers in each of the shared adders. As a result, the critical path time in folded architectures is reduced.

C. 4-2 adder compressor

Adder compressors are utilized in arithmetic and Digital Signal Processing (DSP) circuits for low power and high performance applications. A compressor is basically two full adders connected together as shown in Fig.2. The full adder can be configured as XOR-XNOR gates and multiplexers. Five inputs and three outputs make up the 4-2 adder compressor as shown in Fig.3. It can compute four additions simultaneously and hence preferred for its faster operation. This parallelism feature of 4-2 adder compressor reduces the critical path of SHA-256 thereby reducing dynamic power dissipation. This adder compressor increases the speed of the circuit as Cout is not dependent on Cin, which allows for faster computation. The 4-2 adder compressor has a feature that makes it more efficient than a traditional binary adder.

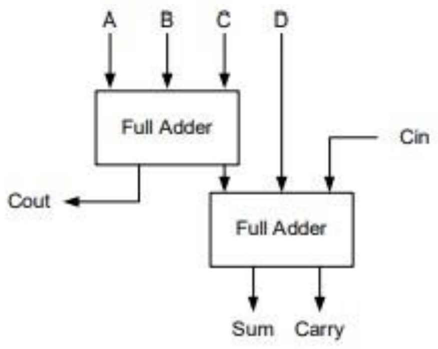

Fig.2 Internal Structure of 4-2 Adder Compressor

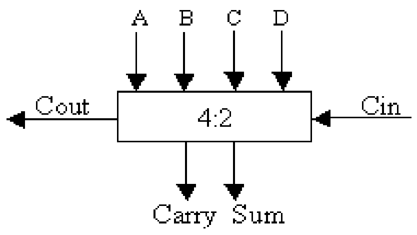

Fig.3 4-2 Adder Compressor

D. Folded SHA-256 architecture

The speed and efficiency are achieved by incorporating 4-2 adder compressor in folded SHA-256 as shown in Fig.4. This results in seven modulo 2^{32}binary adders being replaced by two compressors in folded SHA-256 algorithm. To be exact, the variables are rescheduled with lesser clock cycles. This will promptly increase the speed rate, throughput performance and decrease the hardware area size. The advantages of this architecture are fast, deterministic, more reliable and collision resistant.

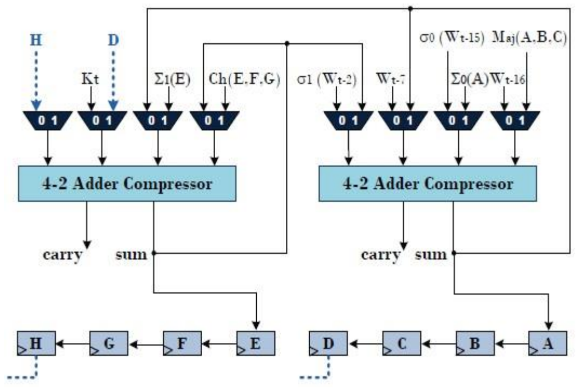

Fig.4 Folded SHA-256 using 4-2 Adder Compressor

III. Results and discussion

The following first four figures represent the simulation results of each of the processes in proposed algorithm. In pre-processing stage, padding is done. The given input data is appended with 1 followed by zeros. The reference input taken is 8 bit and it is given by 11110000 as shown in Fig.5. The padded output is 11110000100000000000000000000000 which is 32 bit length. It is performed for a single character and for multiple characters the output is of 512 bits block.

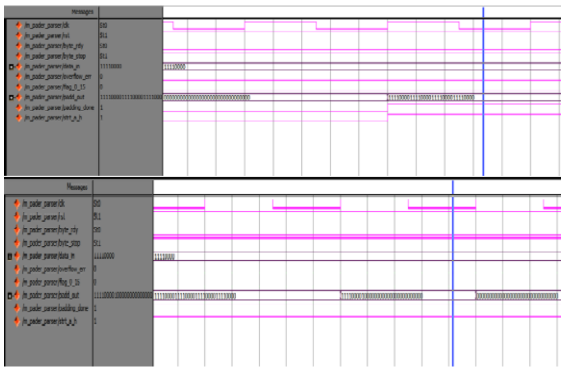
Fig.5 Pre-Process

The scheduling process is done followed by pre-processing. The padded data is fed as input to scheduling stage as shown in Fig.6. For a given padded data to be unique, each data is divided into 64 thirty two bit blocks. Each thirty-two bit block is referred to as register value. For an individual register value, a scheduled value of 7 bits is assigned starting from 0 to 64. As a whole, it performs 64 iterations for each register value and it takes hundreds of clock cycles to iterate.

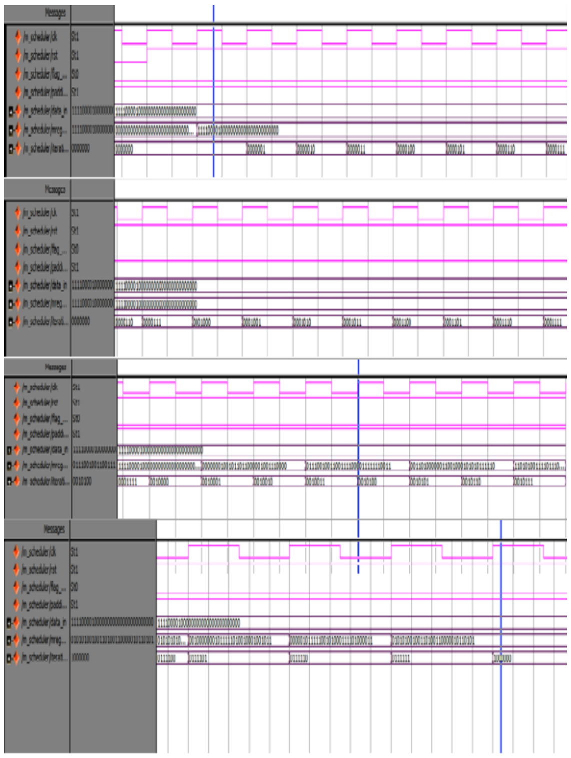
Fig.6 Message Scheduling

In digest calculation, the hash register values A, B, C, D, E, F, G and H are began for the first iteration. For 64 iterations, the eight register values get changed and iterated with its own register value and scheduled value. The highlighted values in Fig.7 are taken as reference and given as input to the digest update process.

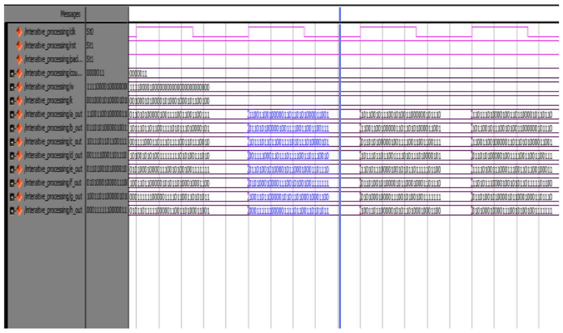
Fig.7 Digest Calculation

The final hash digest value is obtained from digest update stage. It is the result of concatenation of final values of hash registers. The digest length of 256 bits is obtained as output from SHA algorithm as shown in Fig.8.

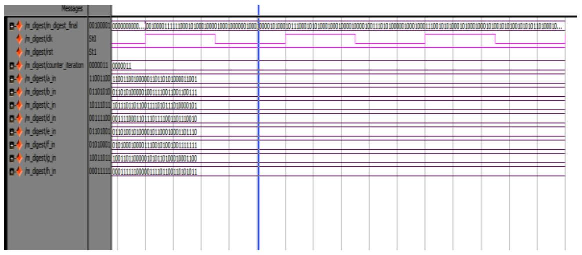
Fig.8 Digest Update

The simulation result of folded SHA-256 algorithm using 4-2 adder compressor is shown in the Fig.9. Input is named as 'data_in' and digest refers to the output. The multiplexer values are referred to as a0, b0, c0, d0, a1, b1, c1 and d1. The digest value is of 256 bits and the time period to obtain last bit is found to be 200ns.

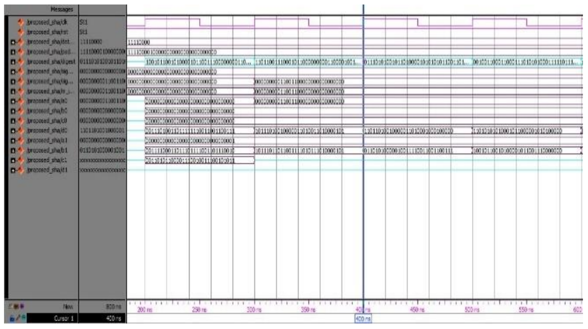
Fig.9 SHA-256 Digest Output

The algorithm steps for conventional and proposed system are synthesized and implemented in hardware to verify its functionality. The device utilization summary gives the area occupied by the design and timing summary gives delay in addition to frequency.Total power of any system refers to the combination of its static power and dynamic power. The dynamic power must be lower than static power for an efficient system and it refers to the power at runtime. The implemented results are presented in terms of area, power and delay in Table I.

Table I Performance analysis

Parameters	**Conventional**	**Proposed**
Number of slice LUTs	2381	350
Number of	274	269
Power	226 mW	177 mW
Delay	4.679 ns	2.594 ns

It is observed that the power consumed by folded SHA-256 is 177mW and conventional system is 226mW. This lowest power consumption is an added advantage to the system. The delay encountered by the folded architecture is also less compared to SHA-256 without folding and compressor.

IV. Conclusion

The folded SHA-256 algorithm has a better scope to the digitalizing world since it produces a unique hash value for every bit of information that is completely non-duplicable by other piece of information. The proposed folded SHA-256 algorithm has been implemented on FPGA using 4-2 adder compressor. It is observed that the proposed design improved the performance of entire system by decreasing the delay and power. The delay is reduced by 45% and power is reduced by 22% compared to conventional method without folding and compressor. Hence, this architecture gives an approach different from conventional way by achieving the balance between high throughput and minimum hardware area.

References

1. Selvakumar A L and Ganadhas C S. “The Evaluation Report of SHA-256 Crypt Analysis Hash Function” International Conference on Communication Software and Networks. (pp.588-592). 2009.
2. Kim M, Lee D G and Ryou J. “Compact and unified hardware architecture for SHA-1 and SHA-256 of trusted mobile computing” Pers Ubiquit Comput 17. (pp.921-932). 2013.
3. Bai L and Li S. “VLSI implementation of high-speed SHA-256” IEEE 8th International Conference on ASIC.(pp.131-134). 2009.
4. Suhaili S B and Watanabe T. “Design of high- throughput SHA-256 hash function based on FPGA” 6th International Conference on Electrical Engineering and Informatics. 2017.
5. Padhi M and Chaudhari R. “An optimized pipelined architecture of SHA-256 hash function” 7th International Symposium on Embedded Computing and System Design. (pp. 1-4). 2017.
6. Michail H, Athanasiou G, Kritikakou A, Goutis C, Gregoriades A and Papadopoulou V. “Ultra high speed SHA-256 hashing cryptographic module for IPSec hardware/software code design” International Conference on Security and Cryptography. (pp. 1-5). 2010.

7. [7] Jeong C and Kim Y. "Implementation of efficient SHA-256 hash algorithm for secure vehicle communication using FPGA" International SoC Design Conference. (pp. 224-225). 2014.
8. Garcia R, Badillo A I, Sandoval M M, Uribe F C and Cumplido R. "A compact FPGA-based processor for the secure hash algorithm SHA-256." Comput. Electr.Eng. 40(1):194–202. 2013.
9. Kumar S and Kumar M. "4-2 Compressor Design with New XOR-XNOR Module" Fourth International Conference on Advanced Computing & Communication Technologies. (pp. 106-111). 2014
10. Michail E H, George S, Athanasiou, Kelefouras V, Theodoridis G and Goutis C E. "On the exploitation of a high-throughput SHA-256 FPGA design for HMAC"ACM Trans. Reconfigurable Technol. Syst. 2012.
11. Canis A, Anderson J H and Brown S D. "Multi-pumping for resource reduction in FPGA high-level synthesis" Design, Automation & Test in Europe Conference & Exhibition. (pp. 194-197). 2013.
12. Badillo AI, Uribe F C, Cumplido R and Sandoval M M. "FPGA-based implementation alternatives for the innerloop of the Secure Hash Algorithm SHA-256" Microprocessors and Microsystems. 37 (6-7) : 750-757. 2012.
13. Zhang and Xiaoyong. "A High-Performance Parallel Computation Hardware Architecture in ASIC of SHA-256 Hash" 21st International Conference on Advanced Communication Technology, (pp. 52-55). 2019.
14. Kim M, Ryou J and Jun S. "Efficient hardware architecture of SHA-256 algorithm for trusted mobile computing". Information Security and Cryptology. (pp. 240-252). 2009.
15. Chaves R, Kuzmanov G, Sousa L and Vassiliadis S. "Cost-Efficient SHA Hardware Accelerators" IEEE Transactions on Very Large Scale Integration Systems. 16(8) : 999-1008. 2008.
16. Michail H, Milidonis A, Kakarountas A and Goutis C. "Novel high throughput implementation of SHA-256 hash function through pre-computation technique".12th IEEE International Conference on Electronics Circuits and Systems. (1-4). 2005.
17. Strollo A G M, Napoli and Meo G D, "Comparison and Extension of Approximate 4-2 Compressors for Low-Power Approximate Multipliers". IEEE Transactions on Circuits and Systems. 2020.
18. Tonfat J and Reis R. "Low power 3−2 and 4−2 adder compressor implemented using ASTRAN".IEEE 3rd Latin American Symposium on Circuits and Systems. (pp. 1-4). 2012.
19. Kahri F, Mestiri H, Bouallegue B and Machhout M. "Efficient FPGA hardware implementation of secure hash function SHA-256/Blake-256" IEEE 12th International Multi-Conference on Systems, Signals & Devices.(pp.1-5).2015.
20. Jung E, Han D and Lee J. "Low area and high speed SHA-1 implementation" International SoC Design Conference. (pp. 365-367).
21. V.R. Balaji, Maheswaran S, M. Rajesh Babu, M. Kowsigan, Prabhu E., Venkatachalam K,Combining statistical models using modified spectral subtraction method for embedded system,Microprocessors and Microsystems, Volume 73,2020.
22. Malar, A.C.J., Kowsigan, M., Krishnamoorthy, N. S. Karthick, E. Prabhu & K. Venkatachalam (2020). Multi constraints applied energy efficient routing technique based on ant colony optimization used for disaster resilient location detection in mobile ad-hoc network. Journal of Ambient Intelligence and Humanized Computing, 01767-9.

23. Miodrag Zivkovic, Nebojsa Bacanin, K. Venkatachalam, Anand Nayyar, Aleksandar Djordjevic, Ivana Strumberger, Fadi Al-Turjman, COVID-19 cases prediction by using hybrid machine learning and beetle antennae search approach,Sustainable Cities and Society, Volume 66,2021.
24. Amin Salih Mohammed, Saravana Balaji B, Saleem Basha M S, Asha P N, Venkatachalam K(2020),FCO — Fuzzy constraints applied Cluster Optimization technique for Wireless AdHoc Networks,Computer Communications, Volume 154,Pages 501-508.