www.ijemr.net ISSN (ONLINE): 2250-0758, ISSN (PRINT): 2394-6962

Vandana Publications
IJEMR

Volume-8, Issue-1 February 2018
International Journal of Engineering and Management Research
Page Number: 1-4

Low Power and Simple Implementation of Secure Hashing Algorithm (SHA-2) using VHDL Implemented on FPGA of SHA-224/256 Core

Dipti Thakur[1] and Prof. Utsav Malviya[2]
[1]M Tech (Embedded System and VLSI Design), Gyan Ganga Institute of Science and Technology, Jabalpur, INDIA
[2]Department of Electronics and Communication, Gyan Ganga Institute of Science and Technology, Jabalpur, INDIA

[1]Corresponding Author: deeptithakuroec@gmail.com

ABSTRACT

Cryptography plays an important role in the security of data. Even though the data is encrypted it can be altered while transmitting on the network so data should be verified using a digital signature. Hashing algorithms are used to create these digital signatures for verification of the data received. Hashing algorithm like Secure Hash Algorithm-2 (SHA-2(224/256)) is designed which has a fixed output length of 512-bits.

Then to improve on power a low-power technique such as latch based clock gating technique is used. After applying these techniques all the designs are compared in terms of power, delay and frequency.

Keywords-- Data Encryption, RPT, Hashing

I. INTRODUCTION

Secure Hashing Algorithm specifications were published in federal information standard (FIPS) PUB180 in 1993 and revised version was issued as FIPS PUB 180-1 in 1995[2], FIPS PUB 180-2 in 2002[3], FIPS PUB 180-3 in 2008 [4] by the National Institute of Standard Technology (NIST). Like SHA-1, SHA-2 is also a one-way and collision-resistant cryptographic hash function with variable digest length [1] and used in cryptographic primitives for security applications and protocols such as TLS, SSL, PGP, SSH, S/MIME and IPsec. SHA-2 hashing functions were introduced to provide additional level of security in security application and protocols. SHA-2 hashing algorithm is a set of four cryptographic hash functions, SHA-224, SHA-256, SHA-384 and SHA-512. All of these SHA-2 hashing algorithms are meant to provide N/2 bits of security against collision attack.

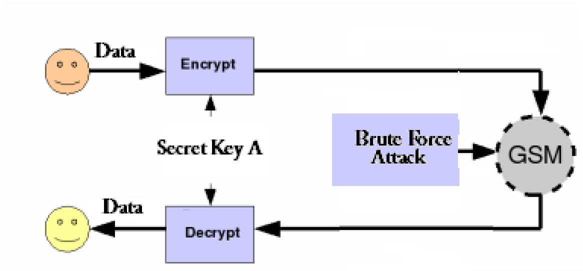

Fig. 1 Data Encryption

Considering the flexibility, physical security, performance and ease to upgrade on hardware implementation, FPGAs (Field Programmable Gate Arrays) is an appealing alternative for the implementation of cryptographic hash (SHA) algorithms.

Hardware implementations of SHA-2 present higher throughput than software, thus being more adaptable for high speed applications which may need accelerators for hashing. Moreover, hashing functions in hardware are uninterruptible contrary to software implementations. Based on the analysis of the SHA-2 algorithm and also from known publications [5] [6] [7] [8] [9] [10], SHA-2 computational path is more complex with higher data dependency and achieving high pipeline frequency and throughput is too critical. Simplified architecture of the SHA-2 hashing algorithm is shown in Fig.

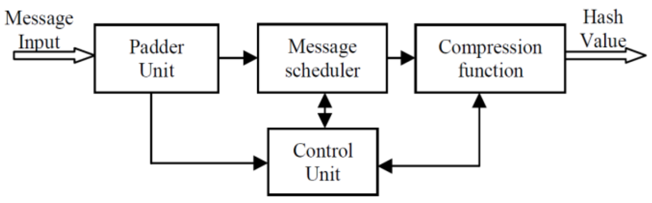

Figure 2: Simplified architecture of the SHA-2 algorithm

SHA-2 architecture uses four basic building blocks viz. padder unit, message scheduler, compression function and control unit. The critical path of the hashing function is tightly coupled with compression function. Proposed design uses Round Pipelined Technique (RPT) for implementing the hashing functions and achieves more throughput per slice.

II. PREVIOUS WORK

1. Design and Optimized Implementation of the SHA-2(256, 384, 512) Hash Algorithms

Wanzhong Sun, Hongpeng Guo, Huilei He, Zibin Dai, Institute of Electronic Technology, Information Engineering University, Zhengzhou 450004, China. 1-4244-1132-7/07-2007 IEEE.

In this paper, the proposed system architecture of SHA-2 hash family can support efficiently the security needs of modern communication applications such as WLANs, VPNs and firewall. Compared with previous published implementations, its achieved performance in the term of throughput / resources is higher. Therefore, it could substitute the implementations of the existing SHA-1 standard or MD5, in all types of applications, with better achieved performance and higher supported security level.

2. Incorporating sha-2 256 with OFB to realize a novel encryption method

Raaed K. Ibrahim, 2 Roula AJ. Kadhim, 3 Ali SH. Alkhalid Computer Engineering, Electrical and Electronics Technical Engineerig College, Middle Technical University, Baghdad, Iraq. 2015 IEEE.

Hash function is built in LabVIEW 2012 and the execution of all steps in SHA-2 256 with OFB is tested using different types of plaintext such as English, Arabic ,symbols ,and numbers to produce fixed 256 bits hash code. After generating the 160 bits hash code, these bits are entered to OFB block as a key to give a very strong code which is very difficult to be breakable. The encrypted hash code gives the confidentiality to the system in addition to the integrity and authentication that obtained from SHA-2 256 itself.

3. SHA-2 Hardware core for Virtex-5 FPGA

Anane Mohamed, Anane Nadjia, ESI (Ecole nationale Supérieure d'Informatique), BP 68M Oued Smar. CDTA (Centre de Développement des Technologies Avancées), BP 17, Baba Hassen. Algiers, Algeria. 12th International Multi-Conference on Systems, Signals & Devices 2015.

In this paper, we have presented an efficient and compact hardware core for SHA-256 hash function. This last is based on the optimization of the two critical paths to speed-up the SHA-2 iteration by using two dedicated components namely compressors 7:2 and 6:2. These components were optimized at the lowest level (LUT) which allows performing a seven operands addition in only 5.016 ns. In order to fully use the slices during the synthesis process of our architecture on FPGA, a special effort was made in the design at the Vhdl description level. The resulting hardware core is compact with good performances. This core can be easily adapted to others variants of SHA-2 by only adjusting the data path of the operands to the word length specified in each standard.

4. High Performance SHA-2 core using the Round Pipelined Technique

Manoj D Rote, Vijendran N, David Selvakumar Secure Hardware and VLSI Design Center for Development of Advanced Computing Bangalore, 560038, INDIA. 978-1-4799-9985-9/15 ©2015 IEEE.

In this research paper several hardware optimization techniques for the SHA-2 hashing functions were explored. A new architecture i.e., 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. A comparison with other published results depicts 57% improvement in throughput per slice for SHA-256 and 17% improvement in throughput per slice for SHA-512. Implementation results indicate that the Round Pipelined technique can help to achieve good tradeoff between throughput and area. As future work, implementations of the SHA-2 core may be attempted by adopting various other design techniques like a design optimized for area using better resource sharing or loop unrolling techniques. Another optimization effort could be to increase the depth of the pipeline stages to increase the throughput.

III. PROJECT CHOICES

The idea is to create a component as simple as possible to use, to pass the message a chunk (portion) at a time without waiting for any synchronization signal. Upon reaching the end of the message, monitor an output signal waiting for the hash. Everything without worrying about "preparing" the message through manual padding (in networked documents this step is always left to the user's process).

Initially, after studying the specifications of NIST, I thought I could create a 512-bit interface to which to pass an entire block every clock cycle and process it in real time. So I would get the result with a single delay clock cycle from the end of the message. The problem of compression (non-parallelizable) however forced me to abandon the single clock strategy in favor of that one "one step each clock cycle", subject to the explosion of the number of components to be used and consequent blocking of the synthesis tool .

There were three possibilities: Pass the entire message to the component and save it in a buffer with subsequent processing and waiting for the result. Easy but with obvious limits on message length far below the algorithm capability (264 bits).

Secondly, introducing a wait time between the passage of two consecutive blocks, i.e. after providing 512 bits, block the user process for several clock cycles (or a ready signal) awaiting partial partial digestion, and then proceed With the next bit train. No limit or memory, or any other type, but falls on the main idea, ease of use.

The way I have chosen involves processing a block at the same time as reading the next, in a sort of pipeline between phases, using a 512-bit scroll buffer to store the data to be processed and new ones Just read. Instructions for the user so do not change, just pass the message without interruption and wait for the final result. There is an intrinsic limit to this type of architecture, phase synchronization, or at least a number of read cycles equal to or greater than the necessary internal phases, a condition necessary to keep the buffer size fixed so as not to have Memory constraints (and therefore message lengths) and not to force the user to pause between one block and another.

Through some redundancies (signals a_2-h_2) I have been able to "embed" the final processing of the digest in the compression phase, so the number of read cycle cycles must be equal to only those needed for compression which are respectively 64 and 80 For SHA-224/256 and SHA-384/512.

IV. PROPOSED DESIGN

The aim of the project is to implement VHDL of the Hash Algorithm version 2 (SHA-2) hashing cryogenic algorithm, which will only provide partial and useful information to understand my work.

From an arbitrary length message (<264), the algorithm generates a fixed length hash, equal to the number present in the name of the four variants of the SHA-224, SHA-256, SHA-384, SHA-512 algorithm.

A preprocessing (or padding) phase is provided in which the message is bit 1 and so many bits 0 to its length to a value divided by 512 for the rest 448 (i.e., 512 to 64) and then accodated The length of the original message (64 bits).

The message is divided into 512-bit blocks and the algorithm is applied sequentially to cascade on the blocks, using the result obtained from the previous block as the input data for the next calculation. Each block passes through three phases: data expansion, compression cycle, and hash processing (or intermediate digestion).

The most critical step for implementation is compression that consists of a 64-step loop (80 for SHA-384/512) that is dependent on each other and therefore not parallelizable. Even expansion in the official drafting of the algorithm is a loop, however, it can be carried out without loosing clock cycles while loading the block through a buffer and a pointer over it.

I will look for other modifications and optimizations (rescheduling, pipelines between the stages of the different blocks, redundancies) to reduce the total delay, some of the many documents available on the net, the main ones in the bibliography (2) (3), others introduced by Tailored to my design. I have examined and discarded many other possibilities (unrolling, quasi-pipeline, special additions) because they are not suitable for my architecture.

V. RESULT SUMMERY

To test the correct functioning of the component we create a test bench directly in the VHDL project that simulates the user process.

Using the VHDL development tool, we can have both automatic and visual feedback.

In the test bench I give a number of test vectors with the corresponding length and control hashes pre-calculated by me with generators found on the web (8) (9). Just choose which test (be careful only include the package of constants for sha-256 or sha-224 in the project) and run the simulation.

Manual / visual control can be done by creating a waveform and adding useful signals (as in the figures in the previous section).

In addition, an automatic control is performed, and a confirmation message is displayed in the output console (or cyclically if the do Loop option is activated) if the result is equivalent to the pre-calculated hash or error otherwise.

VI. CONCLUSION

In this paper several hardware optimization techniques for the SHA-2 hashing functions were explored. A new architecture i.e., 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. A comparison with other published results depicts 57% improvement in throughput per slice for SHA-256 and 17% improvement in throughput per slice for SHA-512. Implementation results indicate that the Round Pipelined technique can help to achieve good tradeoff between throughput and area. As future work, implementations of the SHA-2 core may be attempted by adopting various other design techniques like a design optimized for area using better resource sharing or loop unrolling techniques. Another optimization effort could be to increase the depth of the pipeline stages to increase the throughput.

REFERENCES

[1] William Stallings. (2005). *Cryptography and network security, Principles and Practices* (IV). New Jersey: Prentice Hall.
[2] NIST. (1995 August). Secure Hash Standard. *Federal Information Processing Standards Publication,* 180-1.
[3] NIST. (2002 August). Secure Hash Standard. *Federal Information Processing Standards Publication*, 180-2.
[4] NIST. (2008 August). Secure Hash Standard. *Federal Information Processing Standards Publication*, 180-3.
[5] Harris E. Michail, & Athanasios S. Milidonis. (2009 October-December). A Top-Down Design Methodology for Ultrahigh-Performance Hashing Cores. *IEEE Transaction on Dependable and Secure Computing, 6*(4), 255-268.
[6] N. Skluvos, G. Dimitroulakos & O. Koufopavlou. (2003 December). An Ultra High Speed Architecture for VLSI Implementation of Hash Functions. *Electronics, Circuits and Systems, IEEE International Conference*.

[7] Marco Macchetti, & Luigi Dadda. (2005). Quasi-Pipelined Hash Circuits. *Proceedings of the 17th IEEE Symposium on Computer Arithmetic*. 83-98.
[8] Robert P. McEvoy, Francis M. Crowe, Colin C. Murphy & William 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*. 317-322.
[9] Ricardo Chaves, Georgi Kuzmanov, Leonel Sousa, & Stamatis Vassiliadis. (2008). Cost-Efficient SHA Hardware Accelerators. *IEEE Transaction on Very Large Scale Integration (VLSI) systems, 16*(8), 999-1008.
[10] Shay Gueron, & Vlad Krasnov. (2012 November). Parallelizing message schedules to accelerate the computations of hash functions. *Journal of Cryptographic Engineering, 2*(4), 241-253.
[11] Helion Technology Ltd. (http://www.heliontech.com)
[12] Chanjuan Li1, Qingguo Zhou, & Yuli Liu, Qi. (2011), Cost efficient Data Cryptographic Engine Based on FPGA. *Fourth International Conference on Ubi-Media*. 48-52.