ISSN: 2278 – 909X
International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 6, Issue 11, November 2017

SHA Hash Function Using FPGA

Maloth Santhoshi, Ch Revathi

***Abstract*— The continued growth of both wired and wireless communications has triggered the revolution for the generation of new cryptographic algorithms. SHA-2 hash family is a new standard in the widely used hash functions category. Architecture and the VLSI implementation of this standard are proposed in this work. The proposed architecture supports a multi-mode operation in the sense that it performs all the three hash functions (256, 384 and 512) of the SHA-2 standard. The proposed system is compared with the implementation of each hash function in a separate FPGA device. Comparing with previous designs, the introduced system can work in higher operation frequency and needs less silicon area resources. The achieved performance in the term of throughput of the proposed system/architecture is much higher (in a range from 277 to 417%) than the other hardware implementations. The introduced architecture also performs much better than the implementations of the existing standard SHA-1, and also offers a higher security level strength. The proposed system could be used for the implementation of integrity units, and in many other sensitive cryptographic applications, such as, digital signatures, message authentication codes and random number generators.**

Index terms – Secure Hash Algorithm, FPGA device, Xilinx virtex-II pro, Digital signature algorithm

.

I. INTRODUCTION

This report describes an efficient hardware implementation of the SHA-1 hash algorithm which is a commonly used algorithm in cryptography. The implementation was designed using similar methods that were used in the implementation of the MD5 hash algorithm. The design is called SIG-SHA-1, where SIG is an acronym for the Signal Processing Laboratory at Helsinki University of Technology. SIG-SHA-1 is made in order to compare hardware implementations of SHA-1 and MD5. The design was used also in the evaluation of a combined MD5/SHA-1 module SIG-SHA-1 is a straightforward implementation of the SHA-1 specifications available in, and it performs well in both required area and performance.

Field Programmable Gate Arrays (FPGAs) are almost ideal candidates for implementation platforms of cryptographic algorithms, because they combine the speed of hardware with the flexibility of software. Several benefits of cryptographic algorithms on FPGAs are listed and analyzed.In the implementation described in this report, FPGA devices manufactured by Xilinx are used. Xilinx Virtex-II XC2V2000-6 FPGA device is used as an implementation platform for the presented design. Virtex-II device family offers both fast performance and large logic resources.

Hash algorithms, also commonly called as message digest algorithms, are algorithms generating a unique fixed-length bit vector for an arbitrary-length message M. The bit vector is called the hash of the message and it is here denoted as H. The hash can be considered as a fingerprint of the message. There are several essential features that a hash algorithm must have. First, H must be easy to compute for every given M. Second, it must be hard to compute M when H is given. Third, it must be hard to find another message'M′ which has the same H as M. Here, the term 'hard' means computationally infeasible.

Secure Hash Algorithm (SHA) is described in the National Institute of Standards and Technology's (NIST) Federal Information Processing Standard (FIPS) 180-2: Secure Hash Standard (SHS) . SHS describes the following algorithms: SHA-1 (SHA-160), SHA-256, SHA-385 and SHA-512, where the number is the length of the hash H in bits. In this report, only SHA-1 (SHA-160) is considered.

SHA-1 is widely used in various public-key cryptographic algorithms, e.g. in Digital Signature Algorithm (DSA). This report is organized as follows: first, the SHA-1 algorithm is introduced. Design and implementation of the SHA-1 module is considered in and the results of the implementation. Short comparisons to both MD5 and other published SHA-1 implementations.

II. SHA-1 ALGORITHM

SHA-1 is a part of the FIPS 180-2: Secure Hash Standard. It is very widely used in public-key cryptography, especially in message authentication schemes. SHA-1 calculates a 160-bit H for a b-bit M. The algorithm consists of the following steps:

A. Appending Padding Bits

The b-bit M is padded in the following manner: a single 1-bit is added into the end of M, after which 0-bits are added until the length of the message is congruent to 448, modulo 512.

B. Appending Length

A 64-bit representation of b is appended to the result of the

Manuscript received Nov, 2017.

***Maloth Santhoshi**, Electronics and communication engineering, JNTU,Hyderabad, India,8142435826*

***Chintakuntla revathi**, ECE Department,JNTUH, Hyderabad, India, 880197724.*

above step. Thus, the resulted message is a multiple of 512 bits.

C. Buffer Initialization

Let H0, H1, H2, H3 and H4 be 32-bit hash value registers. These registers are used in the derivation of a 160-bit hash H. At the beginning, they are initialized as follows:

H0 = x″67452301″
H1 = x″e f cdab89″
H2 = x″98badc f e″
H3 = x″10325476″
H4 = x″c3d2e1 f 0″ (4)

D. Processing of the message (the algorithm)

The algorithm which is used for processing of the padded message is described next. First, the padded message needs to be divided into 512-bit blocks, denoted here as Mj where $j \geq 0$ is the index of the block. The algorithm processes one Mj at once, starting from M0, until all Mj have been processed.

Five 32-bit registers, A, B, C, D and E are defined. At the beginning of processing of each Mj their values are set as follows: A←H0, B←H1, etc. The algorithm consists of 80 steps. Let t denote the index of a step, i.e. $0 \leq t \leq 79$. First, a 32-bit message block Wt is derived for every step t from the 512-bit message block Mj using a message schedule. For $t < 16$, Wt is simply the 32-bit word of Mj. When $t \geq 16$,Wt are derived recursively with the following formula. Where n denotes circular shift to the left by s bits and is a logical xor operation. Let Kt be a constant value for step t. The values of K are set as follows: A function F(X,Y,Z) depending on the step t is defined as follows .where ∧, and ¬ are bitwise logical and, xor and complement, respectively. The message is processed for $0 \leq t \leq 79$ with the following function, which is here called the SHA-1 step function

T=(A<<<5)+F(B,C,D)+Wt+Kt+E (5)

Where + denotes an addition modulo 232. After each step, the values of the registers are set as follows:

A ←T
B ←A
C ←B<<<30
D ←C
E ←D (6)

Finally, when all 80 steps have been processed, the following operations are performed:

H0 ←H0+A
H1 ←H1+B
H2 ←H2+C
H3 ←H3+D
H4 ←H4+E (7)

If all Mj have been processed, the algorithm is terminated. Otherwise, the algorithm is processed with Mj+1.

E. Output

When all have been processed with the above algorithm, the 160-bit hash H of M is available in H0, H1, H2, H3 and H4.

III. IMPLEMENTATION

The goal of the design of SIG-SHA-1 was to make an implementation comparable to the SIG-MD5-I design presented in so that the logic requirements and performance of MD5 and SHA-1 could be easily compared. The iterative structure was chosen in order to make a compact structure which could be used in the evaluation of the combined MD5/SHA-1 block introduced.

The top-level architecture used for SIG-SHA-1 implementation is almost similar to the architecture used for SIG-MD5 implementations in a block diagram of SIG-SHA-1. SIG-SHA-1 implements only the steps 3–5, because padding of M is fast to perform also with software, and thus it does not require hardware acceleration. The critical path of the implementation includes the step function block and the multiplexer in front of it. Thus, an efficient implementation of the step function is essential for a high performance hardware implementation of SHA-1. The calculation of the algorithm can be speeded up by unrolling several steps, for example. However, it was decided that this approach was not used in the SIG-SHA-1 implementation, because of the increase in area requirements and, especially, because of the reduced comparability to the combined architecture of and SIG-MD5-I.

where F implements the functions of Equation (4) and is the index of the function derived from the step t. The logic requirements of the step function are the following:

Four 32-bit adders, 5 32-bit registers and Frequires four 32-bit bitwise logical operations selected by a multiplexer. Cyclical left shifts by constant values 5 and 30 do not require any logic, as they are performed simply by reorganizing the bit vector.The constant block in Figure 1 contains the values of constant Kt . It may be implemented using dedicated memory blocks of the FPGA device, e.g. Block-RAMs in Xilinx' devices. However, in SIG-SHA-1 it was implemented on slices in order to guarantee straightforward comparison to SIG-MD5-I.

The message schedule block implements the SHA-1 message schedule described in Section 2. A 512-bit message block Mj is loaded into the block with M_in, adresand loadsignals. The width of M_incan be chosen between 32 and 512 bits, and for the implementations presented in this report the width was chosen to be 128 bits. The message schedule block includes a 16x32-bit shift register and additional logic implementing Equation (2). The five adders in Figure 1 are used for calculating the additions of Equa tion (7). The counter is a 7-bit counter, which counts the value of t from 0 to 79.

The coder derives the index i of the functions of from t, i.e.

$$i = \begin{cases} 0 & \text{if } 0 \leq t \leq 19 \\ 1 & \text{if } 20 \leq t \leq 39 \\ 2 & \text{if } 40 \leq t \leq 59 \\ 3 & \text{if } 60 \leq t \leq 79 \end{cases}$$

The ready block determines when the calculation of the

steps is finished, i.e. t = 79 = 10011112. The leftmost multiplexer is used for initializing the hash value registers. The initial values are set when a derivation of a new hash value is started with start new, i.e. when M0 is processed. If j ≥ 1 in Mj, the values from previous algorithm round are used, and the derivation is began with the continue signal. The other multiplexer is used for controlling the iterative loop. For the first step, t = 0, the initial values or the values from the previous algorithm round are taken (start new or continue), otherwise values from previous iteration step are used When the ready signal is high, a processing of Mj is finished, andMj+1 can be loaded into the design. If all Mj have been processed, the hash H of the message M is ready in hash.

The above architecture was written in VHDL and it required 580 lines of code. Having the experience of implementing MD5, the design of SHA-1 was simple and straightforward. The architecture presented in Section 3 was implemented on a Xilinx Virtex-II XC2V2000-6 FPGA device. The logic synthesis was performed with Simplify 7.3.4 and implementation, including translation, mapping, place & route and timing, as performed with Xilinx ISE 6.2. Alde Active-HDL 6.2 was used for management and simulations. Virtex-II XC2V2000-6 includes logic resources of 10,752 slices. A Virtex-II slice consists of two 4-to-1-bit Look-Up Tables (LUTs), two flip-flops and some additional logic.

Implementation results of the SIG-SHA-1 design on Virtex-II XC2V2000-6 are presented, where the throughput and throughput per slice (TPS) values are

$$\text{throughput} = \frac{\text{block size} \times \text{clock frequency}}{\text{clock cycles per block}}$$

Comparisons

Slices	1,275
Equivalent gate count	25,467
Max. clock frequency	117.5 MHz
Latency of a SHA-1 round	698 ns
Throughput	734 Mbps
TPS	0.576 Mbps / slice

$$\text{TPS} = \frac{\text{throughput}}{\text{slices}}.$$

Implementation results of SIG-SHA-1 on Virtex-II XC2V2000-6 Where block size is 512 bits and clock cycles per block is 82. Based on the values presented in Table 1 it is stated that SHA-1 can be efficiently implemented on FPGAs with minimal logic resources. The performance of SIG-SHA-1 is sufficient for most imaginable applications, but if the throughput falls short for certain applications, similar methods that were used in for increasing the throughput of MD5 can be used also for SHA-1. That is, parallel SHA-1 blocks can be added so that several SHA-1 calculations can be processes simultaneously in parallel. Throughput can be also increased by unrolling several SHA-1 steps and then pipelining the design so that a different SHA-1 calculation can be simultaneously processed in every pipeline stage. These approaches increase the throughput of the design, but they do not decrease the delay of a single SHA-1 calculation. If the delay of a single calculation needs to be reduces, unrolling of several steps may be used.

First, SHA-1 and MD5 are compared. The comparison is straightforward, because similar implementation techniques as well as target devices were used Second,SIG-SHA-1 designs are compared to other published FPGA-based implementations. First of all, it must be stated that SIG-SHA-1 was not designed to be the fastest or most compact SHA-1 implementation. It was implemented merely to compare SHA-1 and MD5 and to provide a rightful reference point for the design of a compact and combined MD5/SHA-1 architecture.

IV. Conclusion

Algorithms in HAS-160 are a hash algorithm developed by Korea Telecommunications Technology Association, and it is not widely used. The structure of HAS-160 is similar to SHA-1 and they were combined together in the design, but MD5 was included merely as a separate block. A more efficient combination of the MD5 and SHA-1 algorithms. There is a great scope for developments in the .we are successful in bringing the SHA in our design implementation using VHDL. In our, we also learned about the SHA that are in market today. Thus there is scope for quantitative, qualitative and total security improvement..

REFERENCES

[1] S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk. Cryptographic hash functions: A survey. Technical Report95-09, Department of Computer Science, University of Wollongong, July 1995

[2] S. Dominikus. A hardware implementation of MD4-family hash algorithms. In IEEE Proc., International Conference on Electronics Circuits and Systems (ICECS), vol. III, pp. 1143–1146, 2002.

[3] P. N. Green and M. D. Edwards. Object oriented development method for reconfigurable embedded systems. IEE Proc., Comput. Digit. Tech., 147(3):153–158, 2000.

[4] A.J. Elbirt, W. Yip, B. Chetwynd, and C. Paar. An FPGA Implementation and Performance Evaluation of the AES Block Cipher Candidate Algorithm Finalists. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 9:545 – 557, August 2001.

[5] Federal Information Processing Standards. Secure Hash Standard. *FIPS PUB 180-2*, August 1, 2002. With changes, February 25, 2004, URL: fips18h0t-2pw:i/thw a.n gsern o.tni set.pgdofv/public ations/fips/fips180-2/ , (visited September 21, 2004).

[6] T. Wollinger, J. Guajardo, and C. Paar. Security on FPGAs: State of the Art Implementations and Attacks. *ACM Transactions on Embedded Computing Systems, Special Issue on Security and Embedded Systems*, 3:534 – 574, 2004.

[7] B. Schneier. *Applied Cryptography*. John Wiley & Sons, 2nd edition, 1996.

Maloth Santhoshi author completed my M.tech at vardhaman engineering college in electronics and communication (DECS) digital electronics and communication system .Author published some journals to GJRE Global Journal of Research in Engineering the paper is about "FIR filter memory based algorithm" in 2014 December. And also for IJSPR International Journal Of Scientific Progress And Research paper on" Compression Technique " in 2017 September. Preparing to join Ph.D in Osmania University. Author also attend workshop at IIIT Basara for NTPEL National Technology Programme For Enhanced Learning.

Ch Revathi author completed her m.tech at a Arvindaksha Educational Socity's Group Of Institutions, in electroics and communication(embedded systems). Author published some journals in IJSPR international journal of scientific progress and research paper on "Accident Report System For Highway's Using GSM"Preparing to join Ph.D in Osmaia university.