ISSN: 2250–3676

[IJESAT] INTERNATIONAL JOURNAL OF ENGINEERING SCIENCE & ADVANCED TECHNOLOGY Volume-2, Issue-3, 742 – 746

NEW TECHNIQUES FOR HARDWARE IMPLEMENTATIONS OF SHA

V.C.Madhavi[1], K.Hanumantha Rao[2], P.Malyadri[3], G. Rama Krishna Prasad[4]

[1]*M.Tech, Department of E.C.E, Prakasam Engineering College, A.P,* ***madhavi.vakkalagadda@gmail.com***
[2]*M.E, Assoc.prof., Dept. of E.C.E, Prakasam Engineering College, A.P,* ***hanumantharao.kilari@gmail.com***
[3]*M.Tech, HOD of E.C.E, Prakasam Engineering College, Kandukur, Andhra Pradesh,* ***paduchurim@yahoo.com***
[4]*M.Tech, Assistant Professor, Dept. of E.C.E, Koneru Lakshmaiah University, A.P,****ramguda1978@gmail.com***

Abstract

Secure Hash Algorithms are one of the forms of cryptographic algorithms. SHA hash functions are widely used security constructs. However, they are software implementations of SHA. This paper proposes techniques for hardware implementation of SHA. In order to provide security and improve performance, these methods are used in hardware reutilization and operation rescheduling. The purpose of implementing SHA at hardware level is to improve throughput. The empirical results revealed that the throughput is increased by 29 to 59% in case of SHA-1 implementation. The throughput is further increased up to 100% when SHA-2 is implemented and used. Thus it is evident that hardware implementation of SHA has more speed when compared with software implementations of SHA.

Index Terms: *Secure Hash Algorithm (SHA), cryptography, hash functions, hardware implementation, FPGA, software implementation.*

---------- *** ----------

1. INTRODUCTION

Cryptography is the branch of computer science that deals with security. It supports operations such as encryption and decryption. The cryptography is implemented in the form of hash functions, symmetric key algorithms, and public key algorithms. The symmetric and public key algorithms are used for encryption and decryption while hash functions are one way functions as they don't allow the retrieval of processed data. As MD5 and SHA are the two mostly used algorithms in the industry, this paper focuses on secure hash algorithms. MD5 can avoid collision attacks [1] with computational feasibility while SHA -1 attacks also computationally expensive [2]. As SHA-1 is not fully secure, the SHA -2 was introduced [3]. At hardware level in order to improve the performance, GPPs (General Purpose Processors) are used. SHA improvement has been done [10], [11]. This paper introduces the implementation of SHA algorithm at hardware level using techniques described here. They are pipeline techniques [5], [18]; embedded memories used to store constant values [8]; improved addition and balanced delays [4], [12]; unrolling techniques [5], [9], [10], [12]; balanced carry save address and parallel counters [4], [5], [7].

This paper proposes two architectures that can be used with hardware. This is meant for achieving high throughput. The results of implementation of SHA-1and SHA-2 algorithms at hardware level provide more speed when compared with software implementations.

2. HASH FUNCTIONS

Since from its inception in 1993, the hash algorithms are improved further to have SHA, SHA-1 and SHA-2. The original SHA was revised in 1995 [15] and named as SHA-1 while SHA-2 WAS INTRODUCED IN 2001 which makes use of DM thus making it more robust to security attacks. SHA functions are available with 128, 256 and 512 bits. From the given input message SHA-1 can produce 160 bit message digest as output. Final DM of 256 bits is the output of SHA 256. The computation of SHA 512 is identical to that of SHA 256. The difference is in the size of operands that means it uses 64 bits instead of 32 bits. Moreover the DM of this algorithm has 512 bits the logical function used are also different [15]. Fig. 1 and Fig. 2 show the round calculations of SHA-1 and SHA-2. SHA-1 needs 80 rounds while SHA 256 uses 64 rounds.

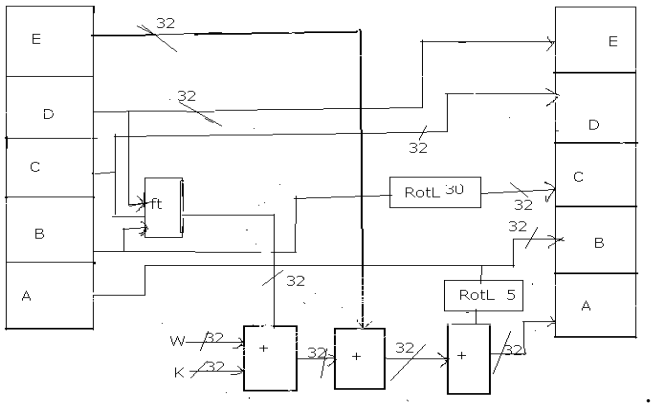

Fig-1: shows round calculation of SHA-1

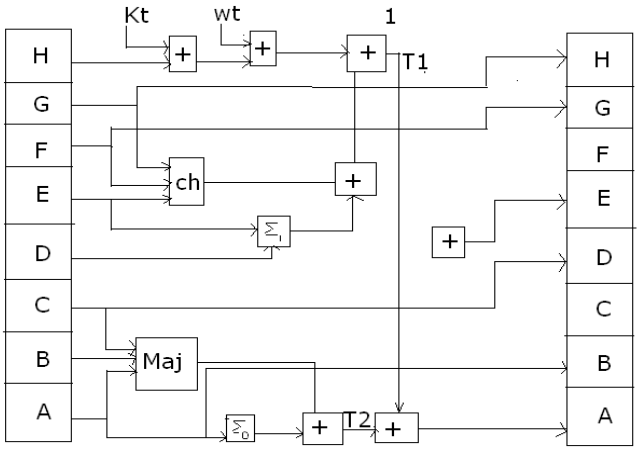

Fig. 2 shows round calculation of SHA-2

2.1. Design for SHA 1:

Each round of SHA-1 requires value from previous round. As rounds are data dependent, it is essential that rounds are carried out sequentially. By unrolling each round computations [10] attempts to speed it up. Another approach increases throughput as it makes use of pipelined structure [11]. As described in [13], high throughput is achieved in SHA-1. Operations rescheduling with respect to this paper has operations rescheduling, hash value initialization, and improved Hash Value Function. The whole computation of SHA-1 is in the A as the rest do not need any computation. The required values are provided by previous round values of various A to D. By adding zero to initialization vector, the internal hash value of first data block can be initialized. Later on this value is loaded into internal registers through multiplexer. In this case instead of the value to the register, it is set to zero as described in [6]. Improving hash value addition is done after all the rounds have been computed for a given data block. Here the internal variables are to be added to the current DM and it needs four additional adders. Finally SHA-1 data block expansion is described here. 512 bits of each data block is expanded in hardware for efficiency reasons as described in [1]. It can be implemented using XOR operations and also registers. Final output is original data for first 16 rounds and computed values for the rest of rounds.

2.2. Design for SHA 2:

As done in SHA-1, the functional rescheduling can also be applied to SHA-2 as well. However, its computational complexity of it is more. In each round values are calculated as and when required. As described in [19], the part that has to be computed is identified. With respect to operational rescheduling values for B, C, D, F, G and H are obtained directly. However, A and E values can't be computed until they are computed in the previous round. With respect to hash value addition and initialization, similar to SHA-1, the internal variables of SHA-2 also have to be added to the DM. It needs eight adders. For each SHA 256 and SHA 512 of 32 bits and 64 bits respectively an adder is required. The empirical results reveal that DM addition with a shift is more efficient. With respect to SHA-2 data block expansion done by data block expansion unit, it is similar to SHA128 in terms of computations. The XOR operation is replaced by arithmetic addition.

3. IMPLEMENTATION:

The SHA designs described above has been implemented as processor cores on a Xilinx VIRTEX II Pro FPGA. The FPGA embedded RAMs (BRAMs) are used in order to implement ROM used to store SHA256 and SHA512. Register – based structures can also be used alternatively. One is based on circular fashion in which memory blocks addressed while the other one is based on FIFOs (First – inputs –first – outputs).

4. PERFORMANCE ANALYSIS AND RELATED WORK:

The resulting cores have been implemented in different Xilinx devices in order to compare the architectural gains of the proposed SHA structures. SHA -1 core, SHA 256 core and SHA 512 core performance comparisons are provided in tables I, II and III respectively.

Design	Lien[11]	Lien[11]	Our-Exp.	CAST[20]	Helion[21]	Our-Cst.	Our+IV
Device	Virtex-E	Virtex-E	Virtex-E	XCV2P2-7	XCV2P-7	XCV2P30-7	XCV2P30-7
Expansion	no	no	no	yes	yes	yes	yes
IV	Cst.	Cst.	Cst.	Cst.	Cst.	Cst.	yes
Slices	484	1484	388	568	564	533	565
Freq.(MHz)	103	73	135	127	194	230	227
TrPut.(Mbps)	659	1160	840	802	1211	1435	1420
TP/Slice	1.4	0.8	2.2	1.4	2.1	2.7	2.5

TABLE-1 shows SHA-1 Core performance comparisons

Architecture	Sklav[22]	Our	McEv.[13]	Our	Helion[23]	Our
Device	XCV	XCV	XC2V	XC2V	XC2PV-7	XC2PV-7
IV	Cst	Yes	Cst	Yes	Cst	Yes
Slice	1060	764	1373	797	815	755
BRAMS	>=1	1	>=1	1	1	1
Freq.	83	82	133	150	126	174
Cycles	n.a.	65	68	65	n.a.	65
Throughput	326	646	1009	1184	977	1370
TP/Slice	0.31	0.84	0.74	1.49	1.2	1.83

TABLE-2 shows SHA256 Core performance comparisons

Architecture	Sklav[22]	Lien[11]	Lien[11]	Our	McEv.[13]	Our	Our
Device	XCV	XCV	XCV	XCV	XC2V	XC2V	XC2VP
Expension	Yes	no	no	Yes	Yes	Yes	yes
IV	Cst	Cst	Cst	Yes	Cst	Yes	Yes
Slice	2237	2384	3521	1680	2726	1666	1667
BRAMS	n.a.	n.a.	n.a.	2	>=1	1	1
Freq.	75	56	67	70	109	121	141
Cycles	n.a.	n.a.	n.a.	81	84	81	81
Throughput	480	717	929	889	1329	1534	1780
TP/Slice	0.21	0.3	0.26	0.53	0.49	0.92	1.01

TABLE-3 shows SHA512 Core performance comparisons

5. INTEGRATION WITH PROCESSOR

SHA algorithms that have been implemented are integrated with a processor known as MOLEN polymorphic processor and its operation [14], [16] is based on the coprocessor architectureal paradigm which allows SHA cores to be embedded in a reconfigurable coprocessor with the GPP. This implementation is similar to the one given in [17]. When compared with software implementations, it is capable of achieving throughputs such as 5 Mbit/s and 4 Mbit/s for SHA 256 and SHA 128 respectively and overall speed is increased by 150 times.

6. CONCLUSION:

We have implemented SHA-1 and SHA-2 algorithms at hardware level. This achieves the reutilization and rescheduling of hardware in terms of area and speed. Critical path can be reduced with the help of operation rescheduling. It leads to the very good usage of pipeline structure. The SHA-2 which makes use of DM causes the reduction of reconfigurable resources. This also hides the extra clock cycle delay. The results of implementation reveals that the hard ware implementation of the hash algorithms are many times better than the software implementations of the same.

REFERENCES

[1] V. Klima, “Finding MD5 collisions—A toy for a notebook.” Cryptology ePrint Archive, 2005/075, 2005.

[2] X.Wang, Y. L. Yin, and H. Yu, “Finding collisions in the full SHA-1,”in CRYPTO, V. Shoup, Ed. New York: Springer, 2005, vol. 3621, Lecture Notes in Computer Science, pp. 17–36.

[3] National Institute of Standards and Technology (NIST), MD, “FIPS 180–2, secure hash standard (SHS),” 2002.

[4] L. Dadda, M. Macchetti, and J. Owen, “The design of a high speed ASIC unit for the hash function SHA-256 (384, 512),” in Proc. DATE, 2004, pp. 70–75.

[5] M. Macchetti and L. Dadda, “Quasi-pipelined hash circuits,” in Proc. IEEE Symp. Comput. Arithmetic, 2005, pp. 222–229.

[6] L. Dadda, M. Macchetti, and J. Owen, D. Garrett, J. Lach, and C. A. Zukowski, Eds., “An ASIC design for a high speed implementation of the hash function SHA-256 (384, 512),” in Proc. ACM Great Lakes Symp. VLSI, 2004, pp. 421–425.

[7] T. Grembowski, R. Lien, K. Gaj, N. Nguyen, P. Bellows, J. Flidr, T.Lehman, and B. Schott, “Comparative analysis of the hardware implementations of hash functions SHA-1 and SHA-512,” in ISC,A.H. Chan and V. D. Gligor, Eds. New York: Springer, 2002, vol. 2433, Lecture Notes in Computer Science, pp. 75–89.

[8] M. McLoone and J. V. McCanny, “Efficient single-chip implementation of SHA-384&SHA-512,” in Proc. IEEE Int. Conf. Field-Program.Technol., 2002, pp. 311–314.

[9] N. Sklavos and O. Koufopavlou, “Implementation of the SHA-2 hash family standard using FPGAs,” J. Supercomput., vol. 31, pp. 227–248, 2005.

[10] R. Lien, T. Grembowski, and K. Gaj, “A 1 Gbit/s partially unrolled architecture of hash functions SHA-1 and SHA-512,” in Proc. CT-RSA, 2004, pp. 324–338.

[11] N. Sklavos, E. Alexopoulos, and O. G. Koufopavlou, “Networking data integrity: High speed architectures and hardware implementations,” Int. Arab J. Inf. Technol., vol. 1, pp. 54–59, 2003.

[12] R. P. McEvoy, F. M. Crowe, C. C. Murphy, and W. P. Marnane, “Optimisation of the SHA-2 family of hash functions on FPGAs,” in Proc. IEEE Comput. Soc. Annu. Symp. Emerging VLSI Technol. Arch. (ISVLSI), 2006, pp. 317–322.

[13] R. Chaves, G. Kuzmanov, L. A. Sousa, and S. Vassiliadis, “Rescheduling for optimized SHA-1 calculation,” in Proc. SAMOS Workshop Comput. Syst. Arch. Model. Simulation, Jul. 2006, pp. 425–434.

[14] S. Vassiliadis, S. Wong, G. N. Gaydadjiev, K. Bertels, G. Kuzmanov,and E. M. Panainte, “The MOLEN polymorphic processor,” IEEE Trans. Comput., vol. 53, no. 11, pp. 1363–1375, Nov. 2004.

[15] National Institute of Standards and Technology (NIST), MD, “Announcing the standard for secure hash standard,” FIPS 180, 1993.

[16] S. Vassiliadis, S. Wong, and S. D. Cotofana, “The MOLEN __-coded processor,” in Proc. 11th Int. Conf. Field-Program. Logic Appl. (FPL),Aug. 2001, vol. 2147, pp. 275–285.

[17] R. Chaves, G. Kuzmanov, S. Vassiliadis, and L. A. Sousa, “Reconfigurable memory based AES co-processor,” in

Proc. 13th Reconfigurable Arch. Workshop (RAW), Apr. 2006, pp. 192–192.

[18] H. E. Michail, A. P. Kakarountas, G. N. Selimis, and C. E. Goutis, "Optimizing SHA-1 hash function for high throughput with a partial unrolling study," in PATMOS, V. Paliouras, J. Vounckx, and D. Verkest, Eds. New York: Springer, 2005, vol. 3728, Lecture Notes in Computer Science, pp. 591–600.

[19] R. Chaves, G. Kuzmanov, L. A. Sousa, and S. Vassiliadis, "Improving SHA-2 hardware implementations," in Proc. Workshop Cryptograph.Hardw. Embedded Syst. (CHES), Oct. 2006, pp. 298–310.