International Journal of Scientific Engineering and Applied Science (IJSEAS) – Volume-3, Issue-12, December 2017
ISSN: 2395-3470
www.ijseas.com

An Efficient Hardware Implementation of KECCAK Algorithm

Hassen Mestiri, Fatma Kahri, Belgacem Bouallegue and Mohsen Machhout
Electronics and Micro-Electronics Laboratory (E. µ. E. L)
Faculty of Sciences of Monastir, Tunisia
hassen.mestiri@yahoo.fr

Abstract

Following the attacks considerable standard SHA-2, in this paper, a new version of hash was developed known as the family SHA-3. We discussed the study of the SHA-3 hash exposing the protocol chosen for our KECCAK-256 application. The optimization of this function and all steps taken to achieve this implementation was done are performed the synthesis of IP hash and optimization. The resulting hardware requirements as well the computation time are presented and compared with previous work. In addition, the proposed design is implemented on the most recent Xilinx Virtex FPGAs.

The number of occupied slices, the maximum working frequency (in megahertz), the throughput (in gigabits per second), and the efficiency (in gigabits per second/slice) have been compared. An FPGA architectural for KECCAK-256 was developed using VHDL, and synthesized using Virtex-6 chips. Our KECCAK-256 show tremendous throughput increase of 195.27% when compared with the implementation of the original KECCAK-256.

Keywords: *SHA-3, KECCAK, Implementation, FPGA Hardware.*

1. Introduction

The hash function is the one of the methods and techniques to ensure the information integrity. Until now, it is a goal related to protect the information.

The SHA-1 and SHA-2 are most-widely used in previous years [1]. However, the NIST announced an international competition in order to developing a new hush function SHA-3. In August 2015, the competition was finalized. The KECCAK is a final version of the new SHA-3 [2].

The new hash function KECCAK is used in very large application requiring high security integrity such as internet banking, online shopping, e-mail and other sensitive digital communications. Since then, different hardware implementation architectures of KECCAK hush function algorithm have been proposed for different applications and their performances have been evaluated by using ASIC libraries and FPGA [3], [4], [5],[6]

The remainder of this paper is organized as follows. Section 2 presents the specification of KECCAK algorithm. The experimental results are explained and discussed in section 3. Section 4 concludes this paper.

2. KECCAK Specification

For having provable security against all attacks, the design construction of the new hush family is based on sponge strategy.

The sponge design perform on a state of b = r + c bits, where r is the bitrates and c is the capacity which determines the security level. So, In Keccak, the function is a permutation chosen in a set of seven Keccak-f permutations, denoted Keccak-f[b], b can be {25, 50, 100, 200, 400, 800, 1600}, where b is the width of the permutation. The state is organized as an array of 5×5 lanes, each of length w ∈ {1, 2, 4, 8, 16, 32, 64} (b=25w). This state (A) is a three-dimensional binary matrix. [7].

The sponge design consists of three phases:

- The initialization phase: In this stage, first all the bits of the state are set to zero. Second, the input data is padded and divided into blocks of r bits.
- The absorbing phase: first, the input message (r-bit) is XORed with the first r-bit of the state. Second, the outputs results are interleaved with the permutation function. Finally, all blocks are processed; the sponge design alters to the third phase.
- The squeezing phase: in this stage the output blocks are the first r-bit of the state. Moreover, the user can be chosen the number of output blocks.

Figure 1 shows the sponge construction:

International Journal of Scientific Engineering and Applied Science (IJSEAS) – Volume-3, Issue-12, December 2017
ISSN: 2395-3470
www.ijseas.com

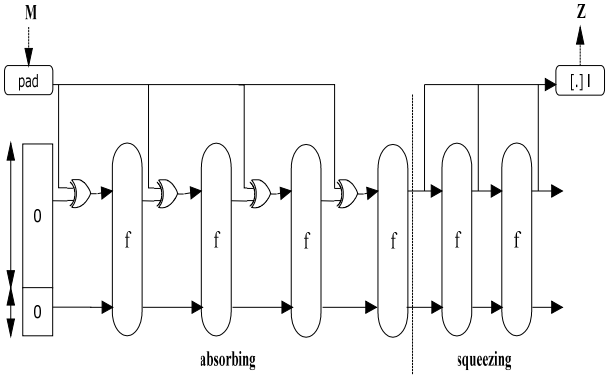

Fig. 1 The sponge construction

Actually, the state is grouped into matrix 5X5 of 64-bit words. The KECCA-f is composed of 24 rounds. Each round has five steps (θ, ρ, π, χ and τ). in each round the initial state is all zero and the data is mixed with the current state.

$$R = i \; o \; \chi \; o \; \pi \; o \; \rho \; o \; \theta \tag{1}$$

$$\theta\, step: C[x] = A[x,0] \; xor \; A[x,1] \; xor \; A[x,2] \; xor \; A[x,3] \; xor \; A[x,4]; \forall x \in [0,4] \tag{2}$$

$$\chi\, step: A[x,y] = B[x,y] \; xor \; \left((not[x+1,y]) and \; B[x+2,y] \right) \tag{3}$$

$$i \; step: A[0,0] \; = \; A[0,0] \; xor \; RC \tag{4}$$

$$\rho \& \pi \; steps: B[y,2x+3y] \; = \; rot\left(A[x,y], r[x,y]\right) \tag{5}$$

In this algorithm all the operations on the indices are done modulo 5. The state named A, A[x,y] is a particular lane. B[x,y], C[x] and D[x] are intermediate variables. The round constant is RC[i]. The constants R[x,y] are the cyclic shift offsets and are specified in the Table 1.

Table 1: Constants R[x,y] KECCAK algorithm

	x=3	x=4	x=0	x=1	x=2
y=2	25	39	3	10	43
y=1	55	20	36	44	6
y=0	28	27	0	1	62
y=4	56	14	18	2	61
y=3	21	8	41	45	15

Table 2 shows the round constants RC[i]

RC[0]	0x0000000000000001	RC[12]	0x000000008000808B
RC[1]	0x0000000000008082	RC[13]	0x800000000000008B
RC[2]	0x800000000000808A	RC[14]	0x8000000000008089
RC[3]	0x8000000080008000	RC[15]	0x8000000000008002
RC[4]	0x000000000000808B	RC[16]	0x800000000000808B
RC[5]	0x0000000080000001	RC[17]	0x8000000000000080
RC[6]	0x8000000080008081	RC[18]	0x000000000000800A
RC[7]	0x8000000000008081	RC[19]	x800000008000000A
RC[8]	0x000000000000008A	RC[20]	0x8000000080008081
RC[9]	0x0000000000000088	RC[21]	0x8000000000008080
RC[10]	0x0000000000008082	RC[22]	0x0000000080000001
RC[11]	0x000000080000000A	RC[23]	0x8000000800008008

3. Proposed KECCAK

3.1 Proposed design

Figure 2 shows the block diagram of our KECCAK architecture. This architecture takes 1600-bit for the inputs data. Then it performs the padding operation and the hash process. The output data is 512-bit. Figure 2 show the block diagram of KECCAK architecture.

The architecture of KECCAK consists of four modules: the Input/output interface, the Control Unit, the Padder Unit and the KECCAK Round.

• Input/output interface is the input blocks. The input data is 1600-bit length while the output is 512-bit wide. So the Input/output interface has to buffer the information data.

• Controller is used to ensuring the synchronization between all modules.

• Padder Unit implements the padding operation and the inversions per byte procedure and has an output

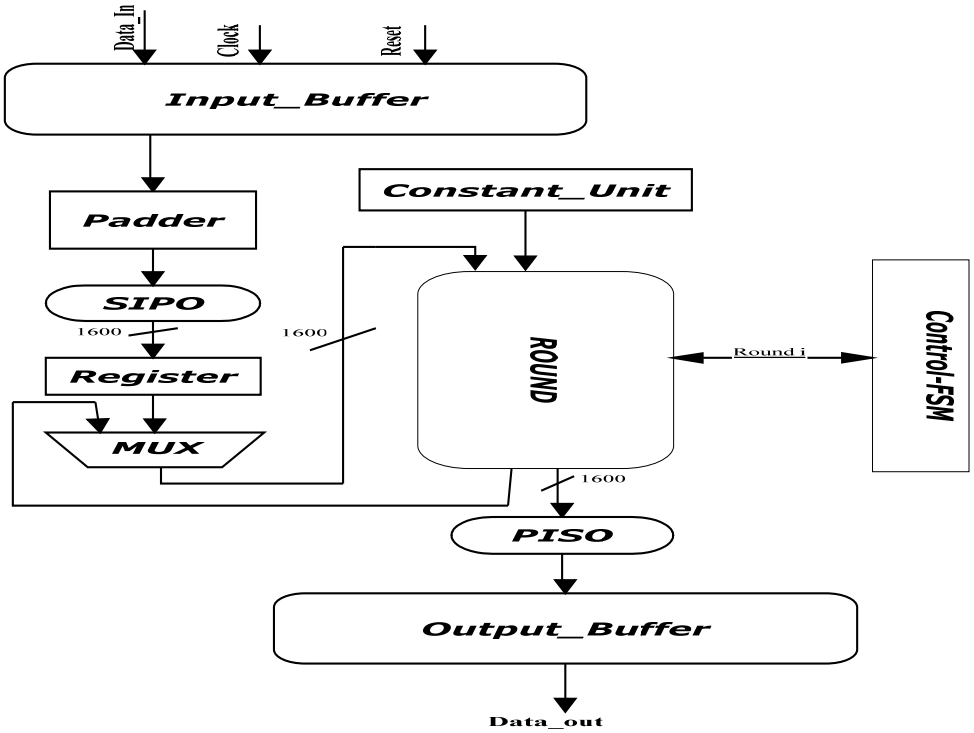

Fig. 2 Block diagram KECCAK

of 1600-bit which is the sponge function of KECCAK. Then a 2-to-1 multiplexer drives the output data from padder to the primary KECCAK components.

• KECCAK Round is the main component of proposed design. It requires 25 clock cycles to produce the 512-bit message digests where each clock cycle requires the previous round, as well as the constant value RC at the start of the each round.

3.2 Implementation: Results

Our Round function is composed of three main components: the round function, the state register and the input/output buffer. The proposed KECCAK architecture has been described using VHDL, simulated by ModelSim 6.6 and synthesized with Xilinx ISE 14.1. The FPGA target is XC6VLX75T-3ff784, from Xilinx Virtex family. As seen in Table 3, the number of occupied slices, the frequency (in Megahertz), the throughput (in Gigabits per second) and efficiency (in Megabits per second).

The throughput is obtained by using the following equation:

$$\frac{\#bit \times frequency}{\#clock\ cycles} \tag{6}$$

The architecture is simulated to verify the functionality, with use of the test vectors provided by the KECCAK standard [8]. In order to have a fair and detailed evaluation. We implemented KECCAK. Performance metrics such as area, frequency, throughput, and efficiency are derived. As seen in Table 3 the proposed KECCAK implementation takes 1167 slices for 333.361MHz frequency in virtex6.

Table.1. KECCAK FPGA implementation: Results

Performances Metrics					
	Area (Slices)	**Clock Cycle**	**Freq. (MHz)**	**Throu. (Gbps)**	**Eff. (Mbps/Slice)**
VIRTEX-6	1167	25	333.361	13.654	11.7

3.2 Implementation: Comparison

In table 4 compares our implementations with recent works reported in literature in terms of Area, frequency, Throughput, Efficiency.

	Ref.	Area (Slices)	Freq. (MHz)	Throu. (Gbps)	Eff. (Mbps/Slice)
Virtex-6	[9]	188	285	0.08	--
	[7]	1015	291.21	6.99	6.89
	Our	1167	333.36	13.65	11.7

In Virtex6 in [9] use a block size of 1088-bit for the internal data rate computation. So this implementation achieves an input throughput at 0.08Gbps in 1896 clock cycles with 285 MHz operating frequency. With the same block size, the implementation [7] has an input throughput of 6.99 Gbps with 291.21MHz operating frequency. So the proposed design increases the maximum frequency by 35.30 %. Also, this design has hardware requirements of 1015 slices.

4. Conclusion

In this work we have presented efficient hardware implementations of KECCAK. We reported the implementation results of 512-bit variants on most up-to-date Xilinx FPGAs i.e Virtex6. We reported the performance figures of our implementations in terms of Area, Frequency, Throughput and Efficiency and compared it with available results. The results of the proposed architecture are found in Section 4. It is important in term frequency, throughput, highlight that the cost of Slices consumed by Throughput. We compared and contrasted the performance figures of others works on Virtex6. This work serves as performance investigation of KECCAK on most up-to-date FPGAs.

References

[1] NIST-FIPS 180-3, 'Secure Hash Standard (SHS)', USA, 2008

[2] F. Kahri, H. Mestiri, B. Bouallegue, M. Machhout " Fault Attacks Resistant Architecture for KECCAK Hash Function" , International Journal of Advanced Computer Science and Applications, Vol. 8, No. 5, 2017, pp 237-243

[3] K. Latif, M. Muzaffar Rao, A. Aziz and A. Mahboob, Efficient hardware implementations and hardware performance evaluation of SHA-3 finalists, NIST Third SHA-3 Candidate Conf., Washington, DC, 22–23 March, 2012.

[4] E. Homsirikamol, M. Rogawski and K. Gaj, Comparing hardware performance of fourteen round two SHA-3 candidates using FPGAs, Cryptology ePrint Archive Report 2010, George Mason University.

[5] F. Kahri, H. Mestiri, B. Bouallegue, M. Machhout " High Speed FPGA Implementation of Cryptographic KECCAK Hash Function Crypto-Processor" Journal of Circuits, Systems, and Computers Vol. 25, No. 4 (2016), 1650026 (15 pages)

[6] S. Bayat-sarmadi, M. Mozaffari-Kermani, and A. Reyhani-Masoleh, "Effcient and concurrent reliable realization of the secure cryptographic SHA-3 algorithm", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 33(7), July 2014.

[7] G. Bertoni, J. Daemen, M. Peeters and G. Van Assche, The KECCAK SHA-3 submission (2011), Submission to NIST (Round 3), http://keccak.noekeon.org/Keccak-submission- 3.pdf [Also see NIST, Keccak hash function (2014), http://csrc.nist.gov/groups/ST/ hash/sha-3.

[8] S.Kerckhof, Fr.Durvaux, N.Veyrat-Charvillon, F.Regazzoni, G. M. de Dormale, F.-X.Standaert, "Compact FPGA implementations of the five SHA-3 finalists", 10th IFIP Smart Card Research and Advanced Applications 2011 (CARDIS 2011), Leuven, Belgium, pp. 217-233, September 14-16, 2011.

[9] A. Gholipour, S. Mirzakuchaki"High-Speed Implementation of the KECCAK Hash Function on FPGA" International Journal of Advanced Computer Science, Vol. 2, No. 8, Pp. 303-307, Aug., 2012.

First Author

Hassen Mestiri received his M.S. and Ph.D degrees in Microelectronic Systems from the Faculty of Sciences of Monastir, Tunisia, in 2011 and 2016 respectively. Dr MESTIRI is currently Assistant Professor at University of Gabes, Tunisia. His research interests include implementation of standard cryptography algorithm, security of embedded system and Hardware/Software Codesign.

Fatma Kahri received here M.S. degree in Microelectronic Systems from the Faculty of Sciences of Monastir, Tunisia, in 2012. She is a PhD student. Her research interests include implementation of standard hash algorithm and security of embedded system on FPGA.

Belgacem Bouallegue received his MSc in Physic Microelectronic, his DEA in Electronic Materials and Dispositifs and Ph.D. degrees in Electronics from University of Monastir, Tunisia, in 1998, 2000 and 2005, respectively. His research interests include High Speed Networks, Multimedia Application, Network on Chip: NoC, flow and congestion control, interoperability, Security Networks implementation of standard cryptography algorithm, key stream generator and electronic signature on FPGA and performance evaluation. He is working in collaboration with Lab-STICC à Lorient Laboratory, Lorient Cedex France and LIP6, Laboratoire d'Informatique de Paris 6, Université Pierre et Marie Curie, UPMC - CNRS UA 7606, Département SoC, Systèmes Embarqués sur Puce, 4, place Jussieu; 75252 PARIS Cedex 05, France.

Mohsen Machhout was born in Jerba-Tunisia, on January 31 1966. He received MS and Ph.D. degrees in electrical engineering from University of Tunis II, Tunisia, in 1994 and 2000 respectively. Dr Machhout is currently Associate Professor at University of Monastir, Tunisia. His research interests include implementation of standard cryptography algorithm, key stream generator and electronic signature on FPGA and ASIC, security of smart card and embedded system with resource constraints.