HIGH THROUGHPUT PIPELINED FPGA IMPLEMENTATION OF THE NEW SHA-3 CRYPTOGRAPHIC HASH ALGORITHM

George S. Athanasiou[(1)], *George-Paris Makkas*[(2)], *Georgios Theodoridis*[(2)]

[(1)] Antcor - Advanced Network Technologies S.A, Marousi, Athens, Greece, www.antcor.com
[(2)] VLSI Design Lab, Electrical and Computer Engineering Dept., University of Patras, Patras, Greece

ABSTRACT

In this paper a two-staged pipelined architecture of the new SHA-3 (Keccak) algorithm is presented. The core can operate on both one-block and multi-block messages, realizing all possible modes of Keccak. Special effort has been paid and different design alternatives have been studied to derive efficient FPGA implementations in terms of throughput and throughput/area metrics. The proposed core has been implemented in Xilinx Virtex-5, Virtex-6, and Virtex-7 FPGA technologies and achieves significant improvements compared to existing FPGA implementations. Specifically, for Virtex-5 the proposed architecture achieves better throughput and throughput/area results from 45.8% to 248× and from 8.9% up to 17.9×, respectively. Regarding Virtex-6, the improvements in throughput and throughput/area are from 47.2% up to 18.1× and from 8% up to 27.3×, respectively.

Index Terms— Security, SHA-3, FPGA, pipeline

1. INTRODUCTION

Hash functions are vital cryptographic modules used for data authentication in security systems and applications. They are used as single modules or included in hash-based authentication mechanisms such the Hashed Message Authentication Code (HMAC) [1]. Furthermore, hash functions are used in the Public Key Infrastructure (PKI) [2], and Secure Electronic Transactions (SET) [3].

In previous years, the most-widely used hash functions were the SHA-1 and SHA-2 [4]. However, an international competition for developing the new SHA-3 hash algorithm had been launched by the U.S. National Institute of Standards and Technology (NIST) [5]. The competition was finalized in October 2012 with the Keccak [6] algorithm being the winner. The final version of the new SHA-3 standard will be published by June 2014.

To meet the real-time constraints as well as the high data rates of modern communication systems, cryptographic modules are usually implemented in hardware. Thus, there are several published works dealing with FPGA implementations of SHA-3 algorithm [7-18].

In this paper a two-staged pipelined architecture of the new SHA-3 (Keccak) algorithm is proposed. The core is fully autonomous and can operate on both one-block and multi-block data. Additionally, it supports all versions of Keccak[r, c], i.e. [r, c] = [1152, 448], [1088, 832], [832, 768], and [576, 1024], as well as the Keccak [1024, 576] with 256-bit output, as submitted in NIST's SHA-3 competition by the developers. Beyond pipeline, to achieve FPGA implementations optimized in terms of throughput and throughput/area factors, several design alternatives have been investigated and efficient circuit optimizations have been applied. The proposed design has been implemented in Virtex-5, Virtex-6, Virtex-7 FPGA technologies and experimental results in terms of frequency, area, and throughput were collected. Based on the experimental results, the proposed architecture achieve significant improvements in terms of throughput and throughput/area factors compared to the existing FPGA implementations.

The rest of the paper is organized as follows. Section 2 presents the SHA-3 algorithm, whereas in Section 3 the proposed architecture is introduced. Section 4 includes the experimental results and comparisons with existing architectures, while Section 5 concludes the paper.

2. THE SHA-3 HASH FUNCTION

The SHA-3 algorithm, originally known as Keccak, is a cryptographic hash function designed by G. Bertoni et. al. [6]. It is based on the sponge construction (absorb/ squeeze), which is a class of algorithms with finite internal state that take an input bit stream of any length and produce an output bit stream of any desired length [6].

The algorithm receives as input a 3 dimensional matrix, called *state-matrix* of size: (5 × 5 × word-length). According to the desired output length, the algorithm uses two parameters for the sponge construction. These two input parameters are the bitrate *r* and capacity *c* [6].

During the absorbing phase (main hash computation), the bitrate of the initialized state is XORed with the first part of the input. The result is used as a new bitrate which forms, together with the capacity of the initialized state matrix, a new state. This state is fed in the main process of the algorithm. The resulting new state is then used as the new

initial state and the process continues for a number of iterations [6]. Moving on, during the squeezing phase the output is considered to be half the size of the capacity and thus, there is no further calculations. The potential SHA-3 output widths of 224, 256, 384 and 512-bit are produced with bitrate, *r*, of 1152, 1088 or 1024, 832, and 576 bits, respectively [6].

The transformation round (absorbing process) of SHA-3 consists of five separate functions and iterates 24 times (64-bit word-length). The functions operate on a 1600-bit state matrix *a* and are described below. The *Iota* function utilize a round constant that is different for each round, but can be pre-calculated before the hash computation. More information about SHA-3 algorithm can be found in [6].

$$\boldsymbol{Theta}: a[x][y][z] \leftarrow a[x][y][z] + \sum_{y'=0}^{4} a[x-1][y'][z] + \sum_{y'=0}^{4} a[x+1][y'][z-1] \quad (1)$$

$$\boldsymbol{Rho}: a[x][y][z] \leftarrow a[x][y]\left[z - \frac{(t+1)(t+2)}{2}\right], \quad with\ t: 0 \le t < 24\ and \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}^{t} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix}\ in\ GF(5)^{2*2}\ or\ t = -1\ if\ x = y = 0 \quad (2)$$

$$\boldsymbol{Pi}: a[x][y] \leftarrow a[x'], with \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix} \quad (3)$$

$$\boldsymbol{Chi}: a[x] \leftarrow a[x] + (a[x+1] + 1)a[x+2] \quad (4)$$

$$\boldsymbol{Iota}: a \leftarrow a + RC[i_r] \quad (5)$$

3. PROPOSED SHA-3 ARCHITECTURE

For clarity reasons, the proposed SHA-3 architecture is gradually presented. Firstly, in sub-Section 3.1 the complete SHA-3 pipelined core is shown. Then, in sub-Section 3.2 the pipelined transformation round, which realizes the main hash computation, is described.

3.1. SHA-3 Pipelined Hash Core

The introduced SHA-3 pipelined hash core is shown in Fig. 1. It consists of the *Transformation Round* (that is discussed in 3.2), the *VSX Module* which comprises the Version Selection and initial XORing, a Zero State Register that holds the initial zero stage for the first iteration, and finally the *Control Unit* (that is described in 3.3).

As imposed by the algorithm, during the first iteration an initial zero-state is used [6]. This zero-state is held in the *Zero State REG* component, which consists of simple registers. Next to *Zero State REG* there is a 2to1 multiplexer that is responsible for realizing the feed-back action when a multi-block message is processed. In this case, the hash output of the current block (state) is fed in the *VSX* so as to be XORed with the new input state (next input block).

The *VSX module* is shown in Fig. 2. It consists of a 1152-bit XOR for the initial storing and five concatenation components, each one of them responsible for forming the appropriate state per algorithm version. Then, a 5to1 multiplexer is used to pass the appropriate state to the Transformation Round, based on the version selection. In contrast to existing SHA-3/Keccak architectures, the version selection is after the initial XORing. This design choice is made so as this module to be effectively implemented as one component in the FPGA, leading to low routing overhead and thus improving delay and area metrics.

Also, there is a register just after the multiplexer. This register prevents the *VSX* logic to increase the critical path,

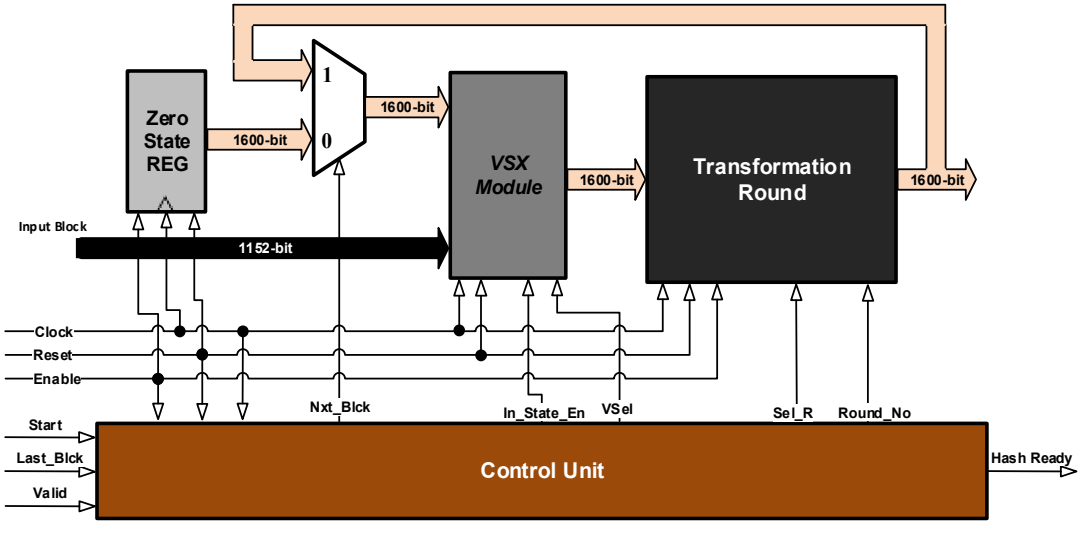

Figure 1. SHA-3 Pipelined Architecture

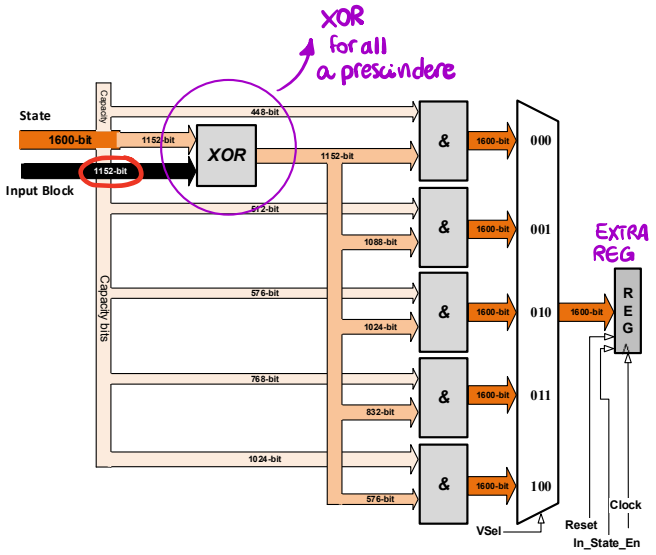

Figure 2. Version Selection and XORing (VSX) Module

which lies inside the Transformation Round. Even though this register imposes an extra clock cycle per data block, its affection to the overall throughput can be considered negligible for large multi-block messages.

The SHA-3 core is controlled by the Control Unit, which is realized using an FSM. The FSM includes 5 states, namely the state Idle, state S1 for version selection and initial XORing, states S2 and S3 for the main computation, and states S4 and S5 for the hash computation of the last block. Also, the FSM includes a counter that counts up to 48, which correspond to 24 iterations of the 2-stage pipelined Transformation Round. Due to 2-stage pipeline the propose core can be fed with 2 input blocks during the first and second clock cycle of operation. This balances the doubling of clock cycles required for one block processing.

3.2. SHA-3 Pipelined Transformation Round

The Transformation round of the proposed core is shown in Figure 3. As the algorithm imposes, it includes five modules that realize the five algorithm's functions, i.e.: *Theta*, *Rho*, *Pi*, *Chi*, and *Iota*. At the beginning of the round there is 2to1 multiplexer for the round's feedback.

The appropriate round constants that are used in *Iota* function (module) are produced by the *Round Constant Generator*. There are three design alternatives for this module: a) the on-the-fly production of the round constant values with a proper circuit that implements a binary linear feedback shift register (LFSR) **[9]**, b) the pre-calculation of the 24 constant values and the use of 24 registers for storing them and a MUX tree to feed these values in *Iota*, and c) the use of a circular buffer that contains the pre-calculated constant values and feeds the corresponding value to *Iota* in each iteration. We have developed and implement these alternatives in Xilinx Virtex-5 technology and concluded that the most effective alternative regarding throughput/area was the first one (due to strict page limit, these metrics are not included in the text).

To apply the pipeline technique, two registers are inserted in the round. The first one is inserted between the *Pi* and *Chi* modules, so as to divide the critical path in almost the half. The second one is placed at the end of the round, just before the feedback branch. Thus, the critical path that is created due to the feedback is also cut in almost half. The control signals of the two pipeline registers are the generic clock, reset, and enable signals that are fed as inputs in the core. Contrary, the select signal of the 2to1 multiplexer as well as the *Round Constant Generator*'s control signal are provided by the core's Control Unit.

4. EXPERIMENTAL RESULTS AND COMPARISONS

The proposed architectures were captured in VHDL, synthesized, and implemented in FPGA technology using the Xilinx ISE Design Suite, v.13.1. Their correct functionality was, initially, verified through Post-Place and Route (Post-P&R) simulation via the ModelSim simulator. A large set of test vectors, apart from the official known-answer tests (KATs), was used for this purpose. Thereafter, downloading to FPGA boards was performed and the functionality of each implementation was also verified on the board via Xilinx ChipScope tool. Also, the Virtex-5 (V5)-(xc5vlx155t-3FF1136), Virtex-6 (V6)-(xc6vlx240t-3FF784), and Virtex-7 (V7)-(xc7v855t-3FFG1157) FPGA

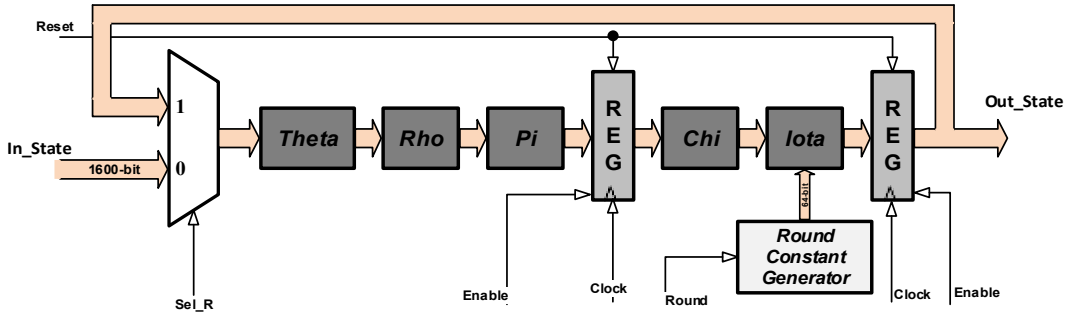

Figure 3. SHA-3 Pipelined Transformation Round

families were selected for implementation.

In Table 1 the implementation results of the proposed architecture are reported, as well as the comparisons with existing architectures. The studied design metrics are: Frequency (F), Area (A) and Throughput (T). The throughput of hash functions is calculated by Eq. (6).

$$Throughput = \frac{(\#bits) \times f}{c} \quad (6)$$

Table 1. Implementation Results and Comparisons

Ref.	FPGA	F (MHz)	A (Slices)	T (Gbps)	T/A (Mbps/S)
[7]	V5	159	393	0.864	2.19
	V6	197	397	1.071	2.69
[8]	V6	285	188	0.077	0.41
[9]	V5	189	1,117	8.518	4.32
[10]	V5	271	1414	12.3	8.68
	V6	333.4	N/A	N/A	N/A
	V7	403.4	N/A	N/A	N/A
[11]	V5	84.21	1562	5.38	3.44
[12]	V5	520	151	0.501	3.32
[13]	V5	259.2	275	0.075	0.58
	V6	299.4	106	0.136	1.28
[14]	V5	285	2573	5.7	2.21
	V5	306	2326	5.56	2.4
[15]	V5	118	1483	6.7	4.51
[16]	V5	282.7	1272	12.82	10.08
	V6	286.2	1207	12.98	10.75
[17]	V5	N/A	1498	8.06	5.38
	V6	N/A	1470	8.85	6.02
[18]	V5	205	1433	8.4	5.86
Prop.	**V5**	**389**	**1702**	**18.7**	**10.98**
	V6	**397**	**1649**	**19.1**	**11.6**
	V7	**434**	**1618**	**20.8**	**12.9**

As it can be seen, the proposed architecture outperforms significantly all the existing ones. Specifically, for Virtex-5 it achieves better throughput and throughput/area values from 45.8% to 248× and from 8.9% up to 17.9×, respectively. Regarding Virtex-6, the improvements in throughput and throughput/area are from 47.2% up to 18.1× and from 8% up to 27.3×, respectively.

6. CONCLUSIONS

In this paper a two-staged pipelined architecture of the new SHA-3 algorithm was presented. Special effort has been paid and several design alternatives have been studied to derive efficient FPGA implementations in terms of throughput and throughput/area, achieving significant improvements compared to existing FPGA implementations. Future work include optimized FPGA implementations of the finalized SHA-3 standard.

7. REFERENCES

[1] NIST-FIPS-198, 'The Keyed-Hash message authentication code (HMAC)", USA, 2006.

[2] NIST- SP 800-32, 'Introduction to Public Key Technology and the Federal PKI Infrastructure', USA, 2001.

[3] L. Loeb, 'Secure Electronic Transactions: Introduction and Technical Reference', Artech House Publishers, 1998.

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

[5] http://csrc.nist.gov/groups/ST/hash/sha-3/index.html, Accessed: 2012.

[6] G. Bertoni, J. Daemen, M. Peeters, G.V. Assche, "The KECCAK reference", 2011, http://keccak.noekeon.org/

[7] B. Jungk, "Evaluation Of Compact FPGA Implementations For All SHA-3 Finalists", Third SHA-3 Candidate Conference, March, 2012.

[8] S. Kerckhof, F. Durvaux, N. Veyrat, F. Regazzoni, F.X. Standaert, "Compact FPGA Implementations of the Five SHA-3 Finalists", CARDIS 2011, pp. 217-233, 2011.

[9] B. Baldwin, A. Byrne, L. Lu, M. Hamilto, N. Hanle, M. O'Neill, W. P. Marnan, "FPGA Implementations of the Round Two SHA-3 Candidates", FPL 2010, pp. 400-407, 2010.

[10] Y. Jararweh, L. Tawalbeh, H. Tawalbeh, A. Mod'd, "Hardware Performance Evaluation of SHA-3 Candidate Algorithms", Journal of Information Security, 2012, 3, 69-76

[11] 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.

[12] I. San, N. At, "Compact Keccak Hardware Architecture for Data Integrity and Authentication on FPGAs", Information Security Journal: A Global Perspective, Taylor & Francis, 21:5, 231-242, 2012

[13] J.-P. Kaps, P. Yalla, K.K. Surapathi, B. Habib, S. Vadlamudi, S. Gurung, "Lightweight Implementations of SHA-3 Finalists on FPGAs", SHA-3 Conference, March 2012

[14] G. Provelengios, P. Kitsos, N. Sklavos, C. Koulamas, "FPGA-Based Design Approaches of Keccak Hash Function", 15th Euromicro Conference on Digital System Design, 2012.

[15] J. Strombergson, "Implementation of the Keccak Hash Function in FPGA Devices", http://www.strombergson.com/files/Keccak_in_FPGAs.pdf

[16] E. Homsirikamol, M. Rogawski, K. Gaj, "Comparing Hardware Performance of Fourteen Round Two SHA-3 Candidates Using FPGAs", Cryptology ePrint Archive, Report 2010/445, Dec. 2010

[17] E. Homsirikamol, M. Rogawski, K. Gaj, "Comparing Hardware Performance of Round 3 SHA-3 Candidates using Multiple Hardware Architectures in Xilinx and Altera FPGAs", Cryptology ePrint Archive: Report 2012/368, 2012.

[18] K. Kobayashi, J. Ikegami, M. Knezevic, E.X. Guo, S. Matsuo, Sinan Huang, L. Nazhandali, U. Kocabas, Junfeng Fan, A. Satoh, I. Verbauwhede, K. Sakiyama, K. Ohta, "Prototyping platform for performance evaluation of SHA-3 candidates" 2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp.60,63, 13-14 June 2010.