Pipelined SHA-3 Implementations on FPGA: Architecture and Performance Analysis

Harris E. Michail
Department of Electrical Engineering, Computer Engineering and Informatics
Cyprus University of Technology
Lemesos 3036, Cyprus
harris.michail@cut.ac.cy

Lenos Ioannou
Department of Electrical Engineering, Computer Engineering and Informatics
Cyprus University of Technology
Lemesos 3036, Cyprus
lc.ioannou@cut.ac.cy

Artemios G. Voyiatzis
Industrial Systems Institute
"Athena" RIC in ICT and Knowledge Technologies
PSP building, Stadiou Str., Platani Patras, GR-26504, Greece
bogart@isi.gr

ABSTRACT

Efficient and high-throughput designs of hash functions will be in great demand in the next few years, given that every IPv6 data packet is expected to be handled with some kind of security features.

In this paper, pipelined implementations of the new SHA-3 hash standard on FPGAs are presented and compared aiming to map the design space and the choice of the number of pipeline stages. The proposed designs support all the four SHA-3 modes of operation. They also support processing of multiple messages each comprising multiple blocks. Designs for up to a four-stage pipeline are presented for three generations of FPGAs and the performance of the implementations is analyzed and compared in terms of the throughput/area metric.

Several pipeline designs are explored in order to determine the one that achieves the best throughput/area performance. The results indicate that the FPGA technology characteristics must also be considered when choosing an efficient pipeline depth. Our designs perform better compared to the existing literature due to the extended optimization effort on the synthesis tool and the efficient design of multi-block message processing.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Design Studies; B.7.1 [**Integrated Circuits**]: Types and Design Styles—*Algorithms implemented in hardware, VLSI*; B.6.1 [**Logic Design**]: Design Styles—*Parallel Circuits*

General Terms

Design, Performance, Algorithms

CS2 '15 January 19-21 2015, Amsterdam, Netherlands

ACM 978-1-4503-3187-6/15/01 ...$15.00.
http://dx.doi.org/10.1145/2694805.2694808

Keywords

Hash function, Pipeline, FPGA, Cryptography, Security

1. INTRODUCTION

Cryptographic hash functions constitute an elementary building block of network security protocols and infrastructures, such as TLS, SSL, SET, IPSec, and PKI [25, 15, 16, 18]. Hash functions can be used to verify integrity of data in transit. They can also be used as message authentication codes, as for example in the case of the Hash Message Authentication Code (HMAC) [19].

The MD5, SHA-1, and SHA-2 algorithms are three cryptographic hash functions that are widely used. As cryptanalysis of hash functions steadily improves over the years, there is a concern regarding the protection these algorithms can offer in the future. The National Institute of Standards and Technology (NIST) of the USA launched a worldwide competition for selecting SHA-3 i.e., a new algorithm that could be used as an alternative option to the existing ones [20]. The NIST opted for the Keccak algorithm in October 2012; the final details and the modes of operation were published as FIPS 202, a draft version of a Federal Information Processing Standard for comments in May 2014 [21].

In the next few years and once the standardization process concludes, it is expected that the SHA-3 will become a mandatory or optional cryptographic hash algorithm for all mainstream and future network security protocols and standards. The increasing speeds for wired and wireless data networks introduces the need for hardware implementations of the cryptographic primitives and algorithms so as to meet the high-throughput requirements. The envisioned omnipresence of the IPv6 network stack and its mandatory IPSec security protocols will push further the need for high-performance implementations. Data encryption and secure hashing are the basic functional blocks for realizing the security protocols of the IPSec suite.

In this paper, the pipeline optimization technique in conjunction with underlying FPGA device technology are explored for the efficient design and implementation of the SHA-3 algorithm. The performance analysis is realized by a generic pipeline hardware architecture that supports all the four modes of the SHA-3 hashing algorithm on FPGAs. Based on this architecture, four different pipelined designs are derived, namely designs for no (one-), two-, three-, and four-stage pipelines. All the four designs can handle

single- and multi-block messages and can support multi-message processing. The designs are implemented in Xilinx Virtex FPGA technologies and experimental results on frequency, area, throughput, and throughput/area ratio are reported. The design space concerning the number of the applied pipeline stages is explored, aiming to support the selection process of an appropriate pipeline depth. The experimental results indicate that the underlying FPGA technology characteristics strongly affect the selection. In all cases, the implementations exhibit enormous improvements in terms of the throughput and the throughput/area metrics compared to the published research literature.

The rest of the paper is organized as follows. Section 2 provides a high-level description of the SHA-3 algorithm and the implementation attempts on FPGAs. Section 3 describes our proposed pipelined architecture. Section 4 presents the experimental results, explores the design space, and draws comparisons with the existing implementations. Finally, Section 5 provides the conclusions of the paper.

2. SHA-3 AND FPGA PERFORMANCE

The SHA-3 algorithm does not use the Merkle-Damgård construction, as is the case of MD5, SHA-1, and SHA-2. Rather, it uses the so-called "sponge construction". The algorithm first *absorbs* input bits into its *hash state* and then an output of equal length is *squeezed* out of it. The hash state has length of 1,600 bits and it can be considered as a $5 \times 5 \times 2^6$ matrix α.

The hash value (message digest) is produced after 24 iterations (rounds). A *round* is composed of five sub-rounds. The sub-rounds are denoted with Greek letters $\theta, \rho, \pi, \chi,$ and ι. They consist of simple operations, like column parity computation; bitwise rotate operation; word permutation; bitwise row combination; and bitwise exclusive-OR operation with per-round constants.

The modes of operation for the SHA-3 hash function are described by the r (rate) and c (capacity) parameters. The rate of the algorithm is the size of the input message block and the capacity of the algorithm is the number of the state bits that remain untouched by the input and the output. The hash value has a size of $c/2$ bits.

The FIPS standard defines six functions for the SHA-3 family. The four of them, namely SHA3-224, SHA3-256, SHA-384, and SHA3-512 are cryptographic hash functions. The other two, namely SHAKE128 and SHAKE256, are extendable-output functions (XOFs) [21]. In all the four cases of hashing, the message digest is derived from the 512-bit output of the algorithm by truncating it at the appropriate size of 224, 256, or 384 bits.

The SHA-3 algorithm was designed to be fast in hardware. In fact, the Keccak algorithm was the second fastest of the 14 algorithms that advanced to the second round of the competition and the fastest of the finalists [6]. Its authors claim 12.6 cycles per byte on an Intel Core 2 Duo CPU [3].

A lot of researchers designed and implemented the then-Keccak algorithm on FPGA technologies during the competition period [24, 2, 13, 12, 8, 10, 9, 5, 23, 11, 22, 4, 14]. The implementation results in terms of area and throughput are varying by more than two orders of magnitude. For example, 188 slices and 0.077 Gbps are reported in [12] and 1,015 slices and 6.99 Gbps for the same (Xilinx Virtex 6) FPGA family are reported in [8].

A more recent work reports a throughput of almost 20 Gbps using a two-stage *sub-round* pipeline [1]. In that work, a pipeline stage is introduced between the π and the χ sub-rounds of the algorithm. We note that the reported throughput can only be achieved in the (rare) case of single-block messages at the input. When a multi-block message arrives at the input, only one of its blocks can be fed to the pipeline. The transformations of the next block cannot start before the output of the previous block becomes available. Moreover, the lack of a scheduling unit prohibits multiplexing in the pipeline blocks that belong in different messages.

Based on the above remarks, we consider that the attainable throughput of that work is closer to 10 Gbps in the common case of multi-block messages. This is a realistic estimation, since the two stages of the pipeline cannot be fully exploited in most of the time. In the next section, we describe a *round* pipelined architecture and designs that can overcome these limitations and achieve a higher throughput.

3. SHA-3 PIPELINED ARCHITECTURE

3.1 Base architecture

We propose a pipelined architecture that is capable of handling *multi-block* messages and of processing *multiple* messages. The architecture supports all the four standardized SHA-3 modes of operation. A high-level description is depicted in Figure 1. The SHA-3 core functionality is contained in the five sub-round blocks named `theta, rho, pi, chi`, and `iota`. These blocks realize the respective sub-round transformations ($\theta, \rho, \pi, \chi,$ and ι) of the 1,600-bit state matrix α. A control logic with a 5-bit counter drives both the 24 repetitions required for a complete SHA-3 round and the use of the appropriate 64-bit constants at each round. A 2-to-1 multiplexer selects as input the result of the previous round (if the counter is greater than 0) or the next input block of 1,600 bits. The result of each repetition is stored in a register and upon completion, the `byte inverse` and `truncate` operations take place for producing the final 512-bit hash value.

The architecture utilizes a software scheduler performing three functions. The first is to prepare the input by splitting and padding long messages into blocks of 1,600 bits (multi-block messages). The second is to truncate, if necessary, the 512-bit result of the hash computation so as to match the size of the selected mode of operation i.e., 224, 256, 384, or 512 bits for the SHA3-224, SHA3-256, SHA3-384, or SHA3-512 respectively. Finally, the third function is to update the state matrix in the case of multi-block messages.

The critical path of the architecture depicted in Figure 1 is located inside the round transformation, passing through the multiplexer and the five sub-round blocks. It consists of a multiplexer; three XOR units and a cyclic left shift block at the `theta` sub-round block; a cyclic left shift block at the combined `rho` and `pi` sub-round blocks; a XOR unit at the `chi` sub-round block; and a XOR unit at the `iota` sub-round block.

The delay of the software scheduler is accommodated given that the critical path lies within the round computation. This design choice is justified, since it reduces the overall design's complexity and it does not affect its security level [17].

3.2 Pipelined designs

The base architecture described above can be easily extended into pipelined designs. For the sake of simplicity and

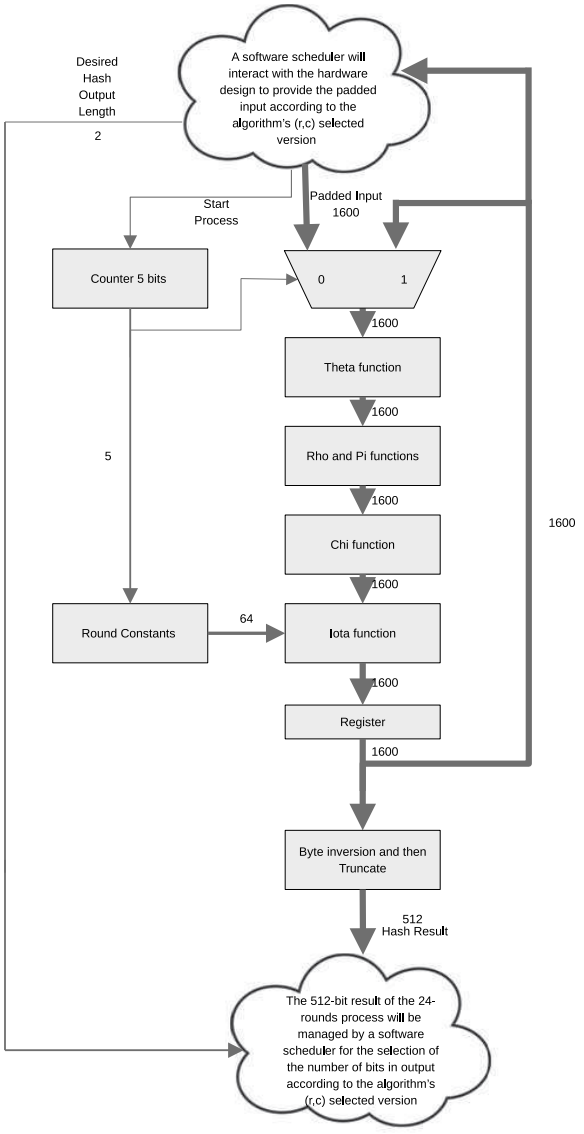

Figure 1: High-level SHA-3 architecture

space economy, we analyze here the case of the four-stage pipeline, as depicted in Figure 2. The two- and three-stage pipeline designs are an adjusted (reduced) form of this design. At each stage, the input is fed through the multiplexer to a pipeline core. This core implements the five sub-rounds and the output of the computation is written to a register. A 3-bit counter drives the six iterations at each core and the respective round constants. At the 6th, 12th, and 18th cycles, the output register is fed as input to the next stage multiplexer. At the last cycle, the output is fed to the `byte inverse and truncate` unit so as to produce the final hash value.

The four-stage pipeline design allows to output a first block after exactly 24 clock cycles and can process a new block (receive the block and compute its hash value) every six clock cycles. This holds under the (common) assumption that the software scheduler handles multiple messages for hashing, as in the case of a network card servicing multiple secure streams of packets that require hashing [4]. In this sense, it is a *round* pipeline and not a *sub-round* one. In our case, all the five sub-round blocks are used at each pipeline stage.

In the case of the three-stage pipeline, three pipeline cores are used and the size of the constant unit is adjusted accordingly, while in the case of the two-stage pipeline, it is necessary to also replace the 3-bit counter with a 4-bit one so as to count up to 12. These designs produce a new block every twelve (two-stage) and eight (three-stage) clock cycles.

The proposed designs can fully occupy the pipeline. To see this, assume that multiple messages of one or more blocks are available at the input of a two-stage pipeline design. Also, denote with M_i^j the j-th block of the i-th message. Now, during the first 12 cycles, M_i^1 passes through the first stage of the pipeline and M_{i-1}^n, the last block of the previous message, passes through the second stage. During the next 12 cycles, M_1^1 passes through the second stage of the pipeline and M_{i+1}^1 passes through the first stage. During the next cycles, the next blocks $M_i^2, M_{i+1}^2, \ldots$ are fed and advance through the pipeline stages. Finally, if no more blocks of these messages are available, then the first block of the next message, M_{i+2}^1, is fed and the process continues as before.

4. EXPERIMENTAL RESULTS

We realized the proposed pipelined architecture in VHDL using the Xilinx ISE Design Suite. Three FPGA boards were used: a Xilinx Virtex 4 XC4VLX200, a Xilinx Virtex 5 XC5VLX330T, and a Xilinx Virtex 6 XC6VLX760. In alignment with the published works, we report the throughput for the SHA3-512 mode assuming that there is sufficient input load at the scheduler.

The throughput is calculated as

$$T = \frac{\#bits \times f}{\#cycles},$$

where $\#bits$ refers to the number of the processed bits, $\#cycles$ corresponds to the required clock cycles between successive messages to generate each message digest, and f is the operating frequency of the design.

Table 1 summarizes the performance of one-, two-, three-, and four-stage pipelined implementations of the SHA-3 on the three boards and in terms of frequency, area, throughput, and throughput/area ratio. The throughput/area metric is the most fair as it correlates the achieved throughput with the consumed area.

In all FPGA platforms, maximum speed grade devices were used during the synthesis. The Optimization Effort (opt level) constraint of the Xilinx ISE synthesis tools was set to High, the Optimization Goal (opt mode) was set to Speed, the design goal was timing performance, register balancing was allowed, and a ucf file was used to target a certain operating frequency.

The critical path of the architecture lies inside the round transformation unit, as explained in Section 3.1. It should be somewhat constant, regardless the number of the pipeline stages, since the proposed design architecture is modular. Table 1 reveals certain variations of the achieved frequency, caused by the different routing delays of each implementation in the Xilinx Virtex FPGAs. For all the considered FPGA families, the throughput increases almost linearly with the number of the pipeline stages. This is justified by the fact that the number of clock cycles is the dominant

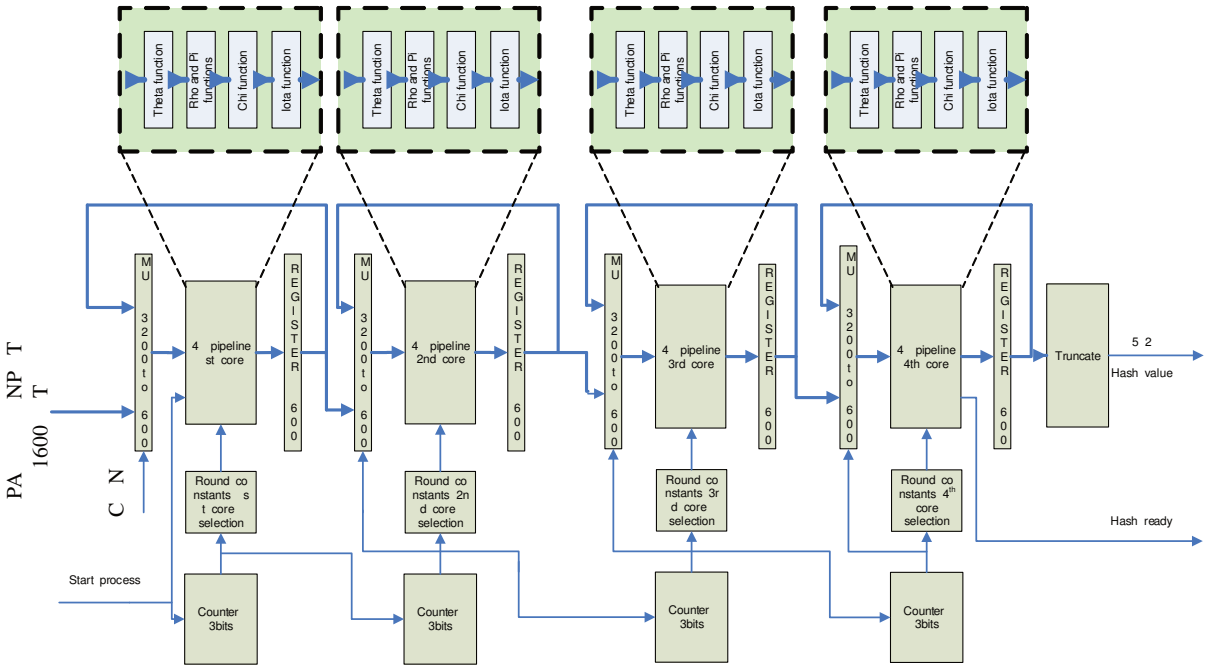

Figure 2: SHA-3 four-stage pipeline design

Table 1: Experimental results in Virtex 4, Virtex 5, and Virtex 6 platforms (A: Area, T: Throughput)

Stages	Freq. (MHz)	Area (slices)	T (Gbps)	T/A (Mbps/slice)
	Virtex 4 XC4VLX200			
1	273	2365	6.552	2.770
2	269	5494	12.912	2.350
3	291	8647	20.952	2.423
4	282	12870	27.072	2.103
	Virtex 5 XC5VLX330T			
1	382	1581	9.168	5.799
2	352	2652	16.896	6.371
3	352	3197	25.344	7.927
4	357	4632	34.272	7.399
	Virtex 6 XC6VLX760			
1	412	1115	9.888	8.868
2	391	2296	18.768	8.174
3	391	3965	28.152	7.100
4	392	4117	37.632	9.141

factor of the throughput equation. This number decreases linearly as the number of pipeline stages increases. The contribution of the frequency variations affects the linearity to some extend as shown in Table 1. As expected, the frequency and the throughput are improved when moving to more modern FPGA families.

Concerning the occupied area, there is a non-linear relation with the number of pipeline stages. This is attributed to the special nature of the FPGA devices: each slice contains one or more LUTs, multiplexers, and flip-flops to implement the logic. As the number of the pipeline stages increases, previously unused resources of the already employed slices can now be also used for implementing the additional logic. This results in the non-linear area increase.

As shown in Table 1, the throughput/area ratio does not improve linearly despite the whole architecture being modular. Moreover, the results are highly dependent on the implementation platform. Specifically, in the case of the Virtex 4 the best throughput/area factor is achieved by the non-pipelined design; in the case of the Virtex 5 the best throughput/area factor is achieved by the three-stage pipeline design; and in the case of Virtex 6 the best throughput/area factor is achieved by the four-stage pipeline design.

The comparison among certain pipelined designs varies, depending on the FPGA platform. In Virtex 4 and Virtex 5, the three-stage pipeline design performs better (in terms of the throughput/area factor) compared to the four-stage pipeline design, whereas exactly the opposite happens in the Virtex-6 platform: the three-stage pipeline design performs worse compared to any other design alternative.

The above remarks lead to the conclusion that an increased pipeline depth can be certainly utilized to achieve higher throughput. However, if we consider the throughput/area factor, the selection of an efficient design requires not only an analysis of the pipeline depth but also of the specific FPGA platform that will be used for the implemen-

Table 2: SHA3-512 throughput (T) in Gbps and throughput/area (T/A) in Mbps/slice on Virtex 5 (V5) and Virtex 6 (V6)

	T		T/A	
Ref.	V5	V6	V5	V6
[24]	6.70	-	4.51	-
[13]	8.40	-	5.86	-
[2]	8.52	-	4.32	-
[7]	6.85		5.45	
[12]	-	0.08	-	0.41
x1,[8]	6.56	7.23	5.37	5.87
x1-PPL2, [8]	8.06	8.85	5.38	6.02
[9]	12.30	-	8.68	-
[10]	0.86	1.07	2.19	2.69
[5]	5.38	-	3.44	-
[23]	0.50	-	3.32	-
[11]	0.08	0.14	0.58	1.28
[22]	5.70	5.56	2.21	2.40
x1, [4]	7.61	7.22	5.77	6.81
x2-PPL4, [4]	12.96	15.66	3.82	6.17
[14]	6.32	6.99	5.28	6.89
[1]	9.35	9.55	5.49	5.80
our	34.27	37.63	7.40	9.14

tation.

In Table 2, we compare the attainable throughput of our pipelined architecture with the ones reported in the published literature (pipelined and not). It is clear that for both the Virtex 5 and Virtex 6 technologies, our architecture achieves at least 2.5 times better throughput compared to the previous works. This can be attributed to its optimized *round* pipeline design. As depicted in Table 2, the T/A ratio is also among the top of the published literature. This is a clear indication of a high-performance design on top of a high-throughput one. This is achieved by significantly reducing the area and by simplifying the complexity of the control circuitry as a fraction of the (core) per-round computations.

5. CONCLUSIONS

A multi-staged pipelined architecture for the SHA-3 cryptographic hash algorithm is proposed in this paper. The architecture can serve multiple multi-block messages. The design exploits a software scheduler for offloading message splitting and padding; reducing circuit complexity; and increasing performance. The scheduler carefully controls the signals so as to remove unnecessary delays for stage synchronization. The design can be considered as *a round pipeline* that can saturate all pipeline stages assuming that multiple message are available. The presented implementations in three Virtex families outperform in throughput by 250% any published results. The results indicate that the FPGA technology characteristics must also be considered when choosing an efficient pipeline depth.

As a future work and based on the promising achieved results, we aim to fully explore the design space of even deeper round pipelines, up to 24 stages. Also, to further study design trade-offs for area-throughput optimizations. Finally, it would be interesting to augment the study with power consumption results and trade-offs.

Acknowledgment

A.G. Voyiatzis was partially supported by the GSRT Action "KRIPIS" with national and EU funds in the context of the research project "ISRTDI" and by the EU FP7 COST Actions IC 1204 "TRUDEVICE" and IC 1403 "CRYPTACUS".

6. REFERENCES

[1] G. Athanasiou, G.-P. Makkas, and G. Theodoridis. High throughput pipelined FPGA implementation of the new SHA-3 cryptographic hash algorithm. In *Communications, Control and Signal Processing (ISCCSP), 2014 6th International Symposium on*, pages 538–541, May 2014.

[2] B. Baldwin, A. Byrne, L. Lu, M. Hamilton, N. Hanley, M. O'Neill, and W. P. Marnane. FPGA implementations of the round two SHA-3 candidates. In *Field Programmable Logic and Applications (FPL), 2010 International Conference on*, pages 400–407. IEEE, 2010.

[3] G. Bertoni, J. Daemen, M. Peeters, G. van Assche, and R. van Keer. Keccak implementation overview version 3.2, 2012. `http://keccak.noekeon.org/Keccak-implementation-3.2.pdf`.

[4] K. Gaj, E. Homsirikamol, M. Rogawski, R. Shahid, and M. U. Sharif. Comprehensive evaluation of high-speed and medium-speed implementations of five SHA-3 finalists using Xilinx and Altera FPGAs. Cryptology ePrint Archive, Report 2012/368, 2012. `http://eprint.iacr.org/`.

[5] A. Gholipour and S. Mirzakuchaki. High-speed implementation of the Keccak hash function on FPGA. *International Journal of Advanced Computer Science*, 2(8), 2012.

[6] X. Guo, S. Huang, L. Nazh, and P. Schaumont. Fair and comprehensive performance evaluation of 14 second round SHA-3 ASIC implementations. NIST 2nd SHA-3 Candidate Conference, 2010. `http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/Aug2010/`.

[7] E. Homsirikamol, M. Rogawski, and K. Gaj. Comparing hardware performance of fourteen round two SHA-3 candidates using FPGAs. Cryptology ePrint Archive, Report 2010/445, 2010. `http://eprint.iacr.org/`.

[8] E. Homsirikamol, M. Rogawski, and K. Gaj. Comparing hardware performance of round 3 SHA-3 candidates using multiple hardware architectures in Xilinx and Altera FPGAs. Ecrypt II Hash Workshop 2011, 2011. `http://www.ecrypt.eu.org/hash2011/proceedings/hash2011_07.pdf`.

[9] Y. Jararweh, L. Tawalbeh, H. Tawalbeh, and A. Moh'd. Hardware performance evaluation of SHA-3 candidate algorithms. *Journal of Information Security*, 3:69, 2012.

[10] B. Jungk. Evaluation of compact FPGA implementations for all SHA-3 finalists. NIST 3rd SHA-3 Candidate Conference, 2012. `http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/March2012`.

[11] J.-P. Kaps, P. Yalla, K. K. Surapathi, B. Habib, S. Vadlamudi, and S. Gurung. Lightweight implementations of SHA-3 finalists on FPGAs. NIST

3rd SHA-3 Candidate Conference, 2012. http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/March2012.

[12] S. Kerckhof, F. Durvaux, N. Veyrat-Charvillon, F. Regazzoni, G. M. de Dormale, and F.-X. Standaert. Compact FPGA implementations of the five SHA-3 finalists. In *Smart Card Research and Advanced Applications*, pages 217–233. Springer, 2011.

[13] K. Kobayashi, J. Ikegami, M. Knezevic, E. X. Guo, S. Matsuo, S. Huang, L. Nazhandali, U. Kocabas, J. Fan, A. Satoh, et al. Prototyping platform for performance evaluation of SHA-3 candidates. In *Hardware-Oriented Security and Trust (HOST), 2010 IEEE International Symposium on*, pages 60–63. IEEE, 2010.

[14] K. Latif, M. M. Rao, A. Aziz, and A. Mahboob. Efficient hardware implementations and hardware performance evaluation of SHA-3 finalists. NIST 3rd SHA-3 Candidate Conference, 2012. http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/March2012/.

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

[16] P. Loshin. *IPv6: Theory, Protocol and Practice*. Elsevier Publications: USA, 2004.

[17] H. E. Michail, G. S. Athanasiou, V. Kelefouras, G. Theodoridis, and C. E. Goutis. On the exploitation of a high-throughput SHA-256 FPGA design for HMAC. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 5(1):2, 2012.

[18] SP 800-32, introduction to public key technology and the federal PKI infrastructure, 2001. NIST Publication, US Dept. of Commerce.

[19] FIPS 198, the keyed-hash message authentication code (HMAC) federal information processing standard, 2002. NIST Publication, US Dept. of Commerce.

[20] NIST. Cryptographic hash algorithm competition - SHA-3, 2010. http://csrc.nist.gov/groups/ST/hash/sha-3/index.html.

[21] Draft FIPS 202, SHA-3 standard: Permutation-based hash and extendable-output functions, May 2014. NIST Publication, US Dept. of Commerce, http://csrc.nist.gov/publications/drafts/fips-202/fips_202_draft.pdf.

[22] G. Provelengios, P. Kitsos, N. Sklavos, and C. Koulamas. FPGA-based design approaches of Keccak hash function. In *Digital System Design (DSD), 2012 15th Euromicro Conference on*, pages 648–653. IEEE, 2012.

[23] I. San and N. At. Compact Keccak hardware architecture for data integrity and authentication on FPGAs. *Information Security Journal: A Global Perspective*, 21(5):231–242, 2012.

[24] J. Strömbergson. Implementation of the Keccak hash function in FPGA devices, 2008. http://www.strombergson.com/files/Keccak_in_FPGAs.pdf.

[25] S. Thomas. *SSL & TLS Essentials: Securing the Web*. John Wiley and Sons Publications, 2000.