2011 21st International Conference on Field Programmable Logic and Applications

Multi-Module Hashing System for SHA-3 & FPGA Integration

Nicolas Sklavos
Informatics & MM Department,
Technological Educational Institute of Patras, HELLAS
e-mail: nsklavos@ieee.org

Abstract- **Hash functions are crucial cryptographic primitives and are widely used in security protocols. Up to now SHA-2 is the proposed NIST standard for hash functions, but is going to be substituted by the end of the next year. NIST is holding up a public competition to select a new hash algorithm(s), called SHA-3 (Secure Hash Algorithm Version 3), for the purpose of completely superseding the functions of the SHA-2 family. The four finalists have been selected by NIST, and a year is allocated for the public review of these algorithms. BLAKE hash functions family is one of the most promising finalists. A multi-module hashing system for this hash functions family, and the FPGA integration of it, are proposed in this work. The introduced system supports a multi-module operation in the sense that upon to the users selection performs efficiently for all hash functions of BLAKE family: -28, -32, -48, -64. The proposed multi-module system is compared with the stand alone implementations of each of the hash functions, integrated in separate FPGA devices. Comparisons with previously published related works are also presented, in order to have a fair and detailed evaluation of the proposed design methodology. Compared with them, the proposed multi-module system performs with higher operation frequency and allocates less silicon area resources. Finally, the proposed performs better, compared with other ones, in the term of throughput on a range of 100% to 500% and better at about 50%, compared with area-delay product factor.**

Keywords: BLAKE; SHA-3; FPGA; Hash Functions; VLSI

I. INTRODUCTION

Cryptographic hash functions play a major role in modern information security protocols and applications [1]. They are extensively used for message authentication systems, digital signatures, password protection mechanisms, random number generators and fingerprinting [2]. Due to their important role in widely adopted security models, cryptanalytic attacks [3], [4] are regularly launched targeting cryptographic hash functions [5]. The attackers aim to discover vulnerabilities and "security holes" in the algorithm itself in order to compromise the security of larger protocols using the targeted family hash functions [6]. Following the cryptanalytic advances, in 2005 security flaws were identified in the widely used SHA-1 [7], indicating the imperative need for a new standard adoption. NIST (National Institute of Standards and Technology) announced SHA-2 [8] as the new standard in cryptographic hash functions. However, the core structure of SHA-2 is very similar to the structure of SHA-1, and the fear of cryptanalytic attacks on SHA-2 might be a matter of time. To forestall this possibility, NIST is currently conducting an open SHA-3 competition in order to conclude to a new standard in cryptographic hash functions. During the running period of the SHA-3 contest, cryptanalytic attacks are launched targeting the proposed structures of the hash function candidates. An overview of security results on the candidates can be found at [9] and [10]. Apart from the ongoing cryptanalytic efforts, benchmarking of software and hardware implementations of the candidates will be an important part of the evaluation. Despite the reference platform introduced by NIST, substantial evaluation projects run in parallel. From these evaluation projects, stands out the eBash project [11] in the context of the ECRYPT II network.

BLAKE Hash Functions Family [12] is one of the most promising finalists of this last phase of SHA-3 competition. The main advantages of this family are the simplicity of the algorithms, the interface for hashing with a salt, as well as security issues. Between them is the design is based on an intensively analyzed component and the resistant of the functions to generic second-preimage attacks, side channels attacks and resistant to length-extension.

In this work, a multi-module operation system for the BLAKE hash function family integration is proposed. In addition the FPGA synthesis results are also presented for the introduced design. The proposed system performs efficiently for all hash functions of BLAKE family upon to the users selection: BLAKE(-28, -32, -48, -64).

Multi-BLAKE proposed system is achieves at better throughput performance at about 250% to 500% for the different performed hash functions (28-, 32-, 48-, 64-), due to the internal parallel process design data path. Multi-BLAKE proposed system compared with the area-delay product is better at about 50% compared with BLAKE(-48) and BLAKE(-64) Stand Alone implementations.

The paper is organized as follows: in Section II a short introduction regarding hash functions is given. The proposed system multi-BLAKE architecture is presented in Section III. The next sections are dedicated to the internal basic components of the proposed multi-BLAKE system. In Section IX the implementation synthesis results are shown. Comparisons with similar works are presented in detail in the same section. Conclusions are discussed in the last section of the paper.

978-0-7695-4529-5/11 $26.00 © 2011 IEEE
DOI 10.1109/FPL.2011.37

II. CRYPTOGRAPHIC HASH FUNCTIONS

Cryptographic hash functions are one-way functions that map an input of arbitrary length to an output of fixed length [6]. The output in cryptography terminology, is called message digest or hash value (Figure 1). Hash functions are one of the most important tools in cryptography, achieving many security goals including authenticity, digital signatures, digital time stamping, entity authentication, digital steganography. They are technically related to other important cryptographic tools such as block ciphers and pseudorandom functions. Additionally, hash functions form an integral component in the functionality of other cryptographic models such as key distribution protocols and zero-knowledge proofs [1]. Their usage in several information processing applications to achieve security goals, is much more widespread than the application of block and stream ciphers [1], [2].

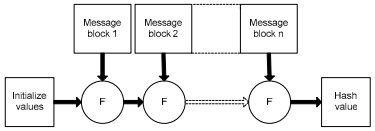


Fig. 1. Fundamental data transformation philosophy of hash functions

III. PROPOSED SYSTEM MULTI-BLAKE SYSTEM

The architecture of the proposed multi-BLAKE system is presented in the following Figure 2. The introduced design performs efficiently for all BLAKE hash family functions (-28, -32, -48, and -64), upon to the control unit commands.

On the system's initialization process, the desirable operation is being selected, upon to the user needs. The internal system's functions and procedures are coordinated upon to the Control Unit (CU) commands. The CU also takes the responsibility of the multi-BLAKE system performance, upon the selection of any one of the four modes of operations, each time. The proper constants and the appropriate word length are been defined by the CU, which also controls the ROM blocks for the right values load. Finally, the CU defines the multi-BLAKE system performance, regarding data transformation based on the proper algebraic functions and digital logic processes, for the operation of BLAKE (-28, -32, -48, and -64) hash family functions.

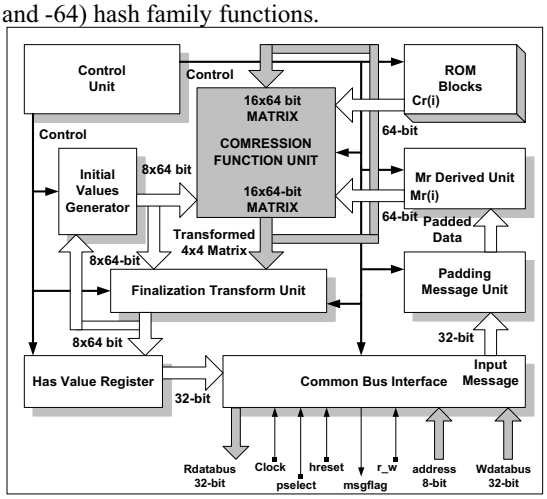


Fig. 2. Proposed multi-BLAKE System Proposed Architecture

First the Padding Message Unit extends the message length and before the data compression starts. The extended data length is a congruent to 447 modulo 512. This data transformation process ensures that the total length of the padded data is a multiple of a 512-bit vector for multi-BLAKE (-28) operation and multi-BLAKE (-32) operation also. While, both multi-BLAKE (-48) operation and multi-BLAKE (-64) operations do need the padded data to be a multiple of 1024-bit vector.

Hash Compression Function (HCF) is the main data transformation component of the proposed multi-BLAKE system. The right number of compression rounds, for each one of the functions of this family, is fully performed in this unit, based on the rolling loop technique (feedback logic). In every round, when data transformation is being performed, based on the above described extended data length of Padded Data, a new data block, Mr(i), is generated in every clock cycle. ROM blocks are used in order the Cr(i) constants set to be loaded, in order to support the operation of Hash Compression Function. These constants differs for every one of the four alternatives multi-BLAKE modes of operation (-28, -32, -48, and -64).

The output data, after the last round is performed, are further modified in the Finalization Transform Unit (FTU). The last one operation is based on the produced n-bit constants values of the initial state (vo, v1, …, v14, v15), with input of the initial chain (ho, h1, …, h14, h15), and the salt (so, s1, …, s14, s15). This is the final procedure in the data transformation and in this way the output hash value is produced, and stored in the Hash Value Register.

In the following sections, the fundamental components of multi-BLAKE proposed system are analyzed in detail.

IV. HASH COMPRESSION FUNCTION

Compression Function Unit (CFU) is illustrated in the Following Figure 3. It's operation is centered on 16 basic data inputs (V0in,…,V15in) and generates another set of 16 outputs (V0out,…,V15out), of compressed data. The appropriate values of initial values, (IVO,…,IV7), are loaded only during the initialization process. CFU operation is based on two other inputs: Mr(i) and Cr(i). Cr(i) constants data are loaded through ROM memory blocks, as presented also in Figure 2, while Mr(i) data are being processed from Mr Derived Unit.

The word length of both INs and OUTs of CFUis equal to 64-bit in total. In order the multi-BLAKE proposed system to perform efficiently for all the four alternative hash functions of this family, the length of both INs and OUTs is organized as two equal parts of 32-bit. The right one (R) is always used, and for all alternative ways of multi-BLAKE operation. The left one, (L), performs only for BLAKE (-48) and BLAKE (-64). For the rest two operation, in case of selection (-28 & -32) is "idle", which it means that its value is stuck at Ox"00000000". This design approach of the proposed multi-BLAKE system, fruitful the specifications of BLAKE(-28) and BLAKE(-32) for 32-bit data transformation, while the other two hash functions (-48 and -64) operations are quite different and it is based on 64-bit words. Data transformation in the CFU is basically performed by G0 to G7 functions. Each Gi function is operating on different vectors of a 4x4 matrix.

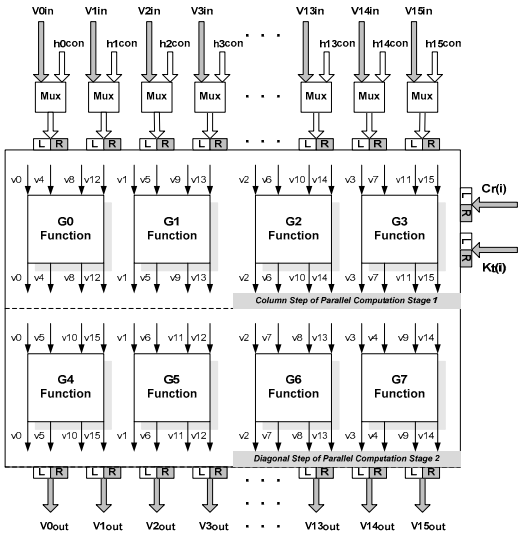


Fig. 3. Compression Function Unit Proposed Architecture

Gi functions in the proposed multi-BLAKE system operate on data with two alternative ways of transformation. The first four are performed on the column step of 4x4 matrix. These transformations are implemented in parallel, since that each one of Gi transforms data of different column of the matrix and there is no overlap between the function inputs. Furthermore the rest four of Gi functions, 4 to 7, are performed also in parallel based on an alternative way of diagonal step. Figure 4 shows both column and diagonal step of data transformation.

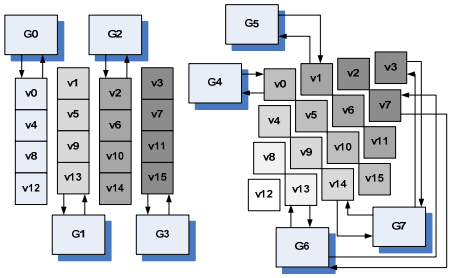


Fig. 4. State Matrix Proposed Parallel Transformation

The proposed architecture of Gi function is presented in the next Figure 5.

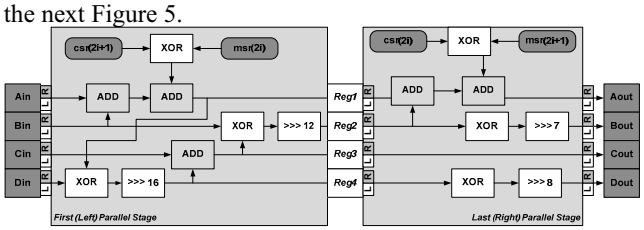


Fig. 5. Proposed Gi Functions Architecture a) Column, b) Diagonal

Since the hash functions specifications define that for each one of the variables, that Gi function applies on, two district values are set, the proposed architecture for Gi functions implementation is based on two parallel processing stage: a) the first (left part), and b) the second (right part). These two stages are working in parallel and in this way the transformation of the data has doubled performance, due to the two parallel processing stages. The penalty of this design is only the delay and the area cost of the registers chain, between the two stages. Regarding time delays the register data path is not considered critical since it is minimal, while for area resources the cost is a set of 4x64-bit. It has to be mentioned that in order the Gi Functions to perform efficiently for the four alternative ways of each mode of multi-BLAKE, the proposed design technique of "Left" and "Right" part for every IN and OUT, as well as for the internal data transformation signals is also applied to the proposed generic Gi Function.

With the S(n) name, the components that perform rotation of k-bit towards less significant bits, are defined. Since the parameters k1, k2, k3, k4 are specified as constants with fixed values for all modes of the proposed multi-BLAKE system operation, the implementation of S(n) components is a matter of right wiring between the inputs and the outputs, with no area and no delay cost for the proposed architecture.

V. Padding Message & MR(i) Units

The input message first is padded in the Padding Message Unit and before the Compression Function Unit (CFU) starts its operation. The purpose of this unit is to ensure that the length of the input data block is the appropriate one, and congruent to 447 modulo 512 for BLAKE(-28), BLAKE(-32), while for BLAKE(-48) and BLAKE(-64) congruent to 895 modulo 1024, accruing to the introduced hash family specifications. Figure 6 presents in detail the structure of Padding Message Unit.

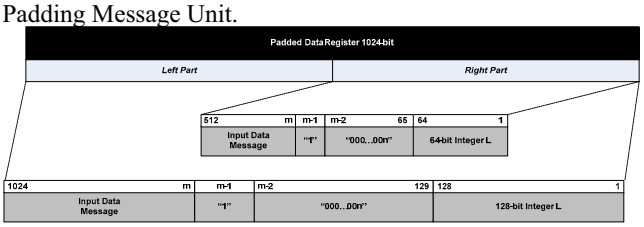


Fig. 6. Padding Message Unit: 512-bit Register

The length extension is performed by appending a logic "1" bit, following by the necessary number of logic "0" bits. For BLAKE(-28 & -32) operations, in the best case 1-bit is appended at least, or 512-bit in the worst one. After that, a bit "1" is added, which is followed by a 64-bit unsigned big endian representation of the initial message bit length L. For an input message (m), which the bit length is $L<2^{64}$, the appropriate transformation of Padding Message Unit can be implemented as an extended register of the input, which certain values of the added bits. For these two operations, only the Right part of the register is used, while the Left remains idle.

For BLAKE(-48 & -64) operations, in the best case 1-bit is appended at least, or 1024-bit in the worst one. After that, a bit "1" is added, which is followed by a 128-bit unsigned big endian representation of the initial message bit length L. For an input message (m), which the bit length is $L<2^{128}$, the appropriate transformation of Padding Message Unit can be implemented as an extended register of the input, which

certain values of the added bits. Both Left and Right parts of the register is used.

Bit n is defined as logic "1" for BLAKE(-32 & -64) and as logic "0" for BLAKE(-28 & -48).

The area cost of the Padding Message Unit is equal to a register of 512-bit, with data path of short delay.

The output data of Padding Message Unit is the input of the Mr(i) Unit, as it is shown in detail in Figure 3. In the Mr(i) Unit the padded data are split into equal words m^o, …, m^{N-1}. If we let L^i be the number of message bits in m^o,…,m^i, which is excluding the bits added by the padding. In addition, random chosen vectors used as "salt" in data transformation. The salt is chosen by the user, while is set to null value when no salt is required. In this way the hash computation of the produced data message could be described with the following feedback logic:

h(o) = IV
for i=0, …, N-1
h(i+1) = compress [h(i) , m(i) , s, L(i)]
return h^N
where: h(i) is the chain value, m(i) is the message block, s is the salt value, L message length.

VI. ROM Memory Blocks

The ROM memory blocks are used for the processes of the specified constants values Cr(i), which are different for each one of the hash functions of the BLAKE family. BLAKE(-28) and BLAKE(-32) use sixteen constants of 32-bit, while the other two ones BLAKE(-48) and BLAKE(-64) use a set of sixteen constants of 64-bit.

From the 16x64-bit constants values of BLAKE(-48) and BLAKE(-46), the first 8x64-bit are the same with the ones used in BLAKE(-28) and BLAKE(-32). In order to implement the Cr(i) constants set, for the multi operations of the proposed system, only two ROM arrays of 16x64-bit are finally used. Following this methodology, the multi-BLAKE operational modes are supported with the minimized ROM blocks allocated resources at about 33% of the total cost.

In the cases of BLAKE(-28) and BLAKE(-32) modes of operation, the Right ROM Block 32-bit output, is the Right part of the Cr(i), while the left part is "idle", which means that it is stuck at Ox"00000000".

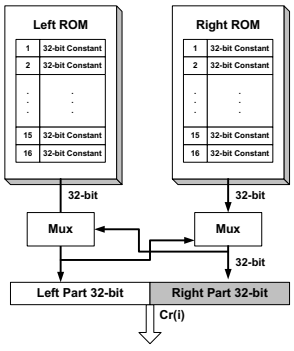


Fig. 7. ROM Blocks for Cr(i) Operation

In the cases of BLAKE(-48) and BLAKE(-64) modes of operation, the Right ROM Block 16x32-bit output, are both the Left and the Right parts of the first 8x64-bit outputs, while the for the last 8x64-bit outputs the Left ROM Block 16x32-bit outputs are used also for the Left and Right parts of the output (Figure 7).

VII. Initial Values Generator

BLAKE hash functions family defines two different sets of 8x32-bit for BLAKE(-28) and BLAKE(-32) and 8x64-bit for BLAKE(-48) and BLAKE(-64). The constants are basically used during the initialization procedure of the Compression Function Unit and during the Finalization Transform Unit. Figure 8 presents in detail the proposed Initial Values Generator.

Eight constants of 64-bit each one, are stored in total for the right operation of BLAKE(-48) and BLAKE(-64) of multi-BLAKE modes. Each one of the constants, through a multiplexers cascade is forwarded to the outputs IV0, …, IV7. In the case of BLAKE(-28) and BLAKE(-32) the appropriate 8x32-bit values are completely the same right parts of the 8x64-bit values, used for the previous described modes.

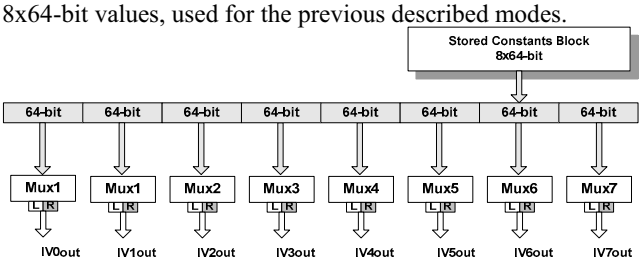


Fig. 8. Initial Values Generator Unit (IVGU) Proposed Architecture

For this reason, the cascade of multiplexers forward to the right parts of the outputs only the left parts of the 8x64-bit stored constants. In this way, a total of 33% area resources are saved. The appropriate storage of the 16x64-bit constants values could be implemented with Look Up tables (LUTs), which is a fundamental and cheap basic primitive of FPGAs of almost any kind.

VIII. FPGA Implementation Synthesis Results

The proposed multi-BLAKE system architecture, shown in Figure 2, was captured by using VHDL, based on completely structural design logic. The pure code was synthesized, placed and routed by using a XILINX FPGA Virtex4 device, by using XILINX ISE, 12.1, available at [14].The system then was simulated again, and verified considering real time operation, by using BLAKE (-28, -32, -48, -64) official test vectors. The implementation synthesis results are illustrated in the next Table I. In order to achieve a fair and detailed comparison of the proposed multi-BLAKE system, the introduced implementation is compared with stand-alone ones implementations, of each one of the four hash functions of BLAKE family [15]. The implementation results of the last ones are also illustrated in the same Table I.

BLAKE related implementations have been previously published from other research groups such as [16] and the hash functions family introducers [12], with quite different implementation criteria compared with the introduced work. In order to provide a fair and detail comparison, comparisons of the proposed multi-module system with the stand-alone implementations are given.

Table I. Synthesis Results: multi-BLAKE vs Stand Alone

Device Implementation	Covered Area (Slices)	ROM (-bit)	F(MHz)
BLAKE(-28)	3017	-	52
BLAKE(-32)	3101	-	50
BLAKE(-48)	10986	-	28
BLAKE(-64)	11800	-	27
Proposed System multi-BLAKE	11500	16x64	65

The proposed multi-BLAKE system has almost the same allocated resources with the stand alone BLAKE(-64) implementation. It is obvious that BLAKE(-28) and BLAKE(-32) hardware integrations less allocates less area resources, at about 60%. The operation frequency of the proposed multi-BLAKE proposed system reaches the value of 65 MHz. Although, it was expected to be near to the value of BLAKE(-64) since it is also performs as BLAKE(-64), the proposed parallel processing of the two stages in Gi Function of Compress Function Unit (Figure 2 and Figure 5) results to this advantage. Furthermore, in order to have a fair and detailed comparison of the proposed multi-BLAKE system, comparisons using both the achieved throughput and the area-delay produce, with the Stand Alone implementations of BLAKE for (-28,-32,-48,-64) are also presented in Figure 10.

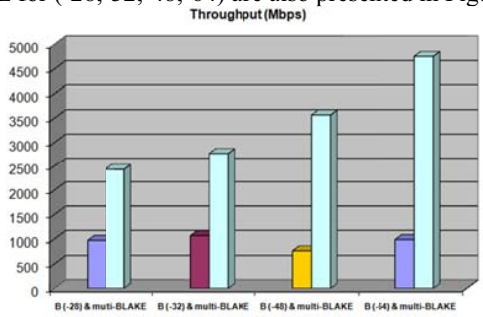
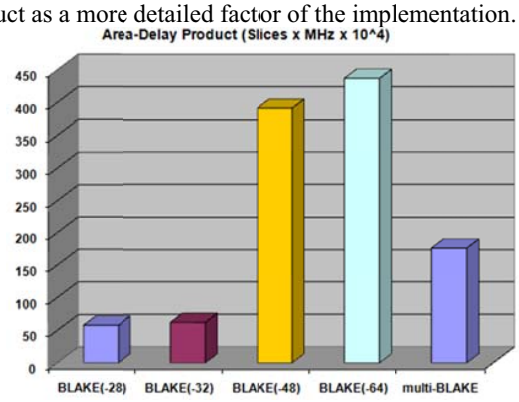


Fig. 10. a) Throughput & b) Area-Delay Product Comparisons.

Multi-BLAKE proposed system is achieves at better throughput performance at about 250% to 500% for the different performed hash functions (28-, 32-, 48-, 64-), mainly due to the internal parallel process design data path of Gi functions. Furthermore, multi-BLAKE proposed system compared with the area-delay product is better at about 50% compared with BLAKE(-48) and BLAKE(-64) Stand Alone implementations. BLAKE(-28) and BLAKE(-32) are proved more flexible designs, to the area resources minimization, compared with area-delay model, while they reach poor thoughput values, compared with the proposed multi-BLAKE system.

IX. CONCLUSIONS & OUTLOOK

In this work a multi-BLAKE system design and the FPGA implementation of it is proposed for the hardware integration of BLAKE hash function family, which is one of the most promising candidates for SHA-3 hash functions standards. Comparisons with the conventional BLAKE hash functions implementations prove the superiority of the proposed system, regarding throughput values, and by using the area-delay product as a more detailed factor of the implementation.



X. REFERENCES

[1] Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno, *Cryptography Engineering*, John Wiley & Sons, March 15, 2010, ISBN: 9780470474242.

[2] Nicolas Sklavos, Xinmiao Zhang, *Wireless Security & Cryptography: Specifications and Implementations*, CRC-Press, A Taylor and Francis Group, ISBN: 084938771X, 2007.

[3] K. Theoharoulis, I. Papaefstathiou, C. Manifavas, "Implementing Rainbow Tables in High-End FPGASs for Super-Fast Password Cracking", proceedings of International on Filed Programmale Logic and Applications, August 31 – Septmber 2, Milano, Italy 2010.

[4] K. Theoharoulis, C. Manifavas, and I. Papaefstathiou, "High-End reconfigurable System for Fast Windows'password Crackin", IEEE FCCM, Napa, California, USA, April 5-7, 2007.

[5] A. Bechtsoudis, N. Sklavos, "Side Channel Attacks Cryptanalysis Against Block Ciphers Based on FPGA Devices", proceedings of IEEE Computer Society Annual Symposium on VLSI (IEEE ISVLSI'10), Kefalonia, Greece, July 5-7, 2010.

[6] S. Bakhtiari, R. Safavi-Naini, J. Pieprzyk, "Cryptographic Hash Functions: A Survey", Technical Report 95-09, Department of Computer Science, University of Wollongong, July 1995.

[7] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1. In Victor Shoup, editor, Advances in Cryptology – CRYPTO 2005, 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005.

[8] SHA-2 Standard, National Institute of Standards and Technology (NIST), Secure Hash Standard, FIPS PUB 180-2, 2011.

[9] ECRYPT II, The SHA-3 Zoo Project, 2011, online at: http://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo

[10] SHA-3 Cryptanalysis by the Grøstl Team, 2011, online at: http://www.groestl.info/cryptanallysis.html

[11] eBASH: ECRYPT Benchmarking of All Submitted Hashes, online at: http://bench.cr.yp.to/ebash.html

[12] J. P. Aumasson, L. Henzen, W. Meier, and R.C.W. Phan, "SHA-3 Proposal BLAKE", online at: http://www.131002.net/blake, 2011.

[13] SHA-3, National Institute of Standards and Technology (NIST), Cryptographic Hash Algrorithm Competition, online at: http://csrc.nist.gov/groups/ST/hash/sha-3/index.html

[14] XILINX, San Jose, California, USA: www.xilinx.com, 2011.

[15] N. Sklavos, P. Kitsos, "BLAKE HASH Function Family on FPGA: From the Fastest to the Smallest", proceedings of IEEE Computer Society Annual Symposium on VLSI (IEEE ISVLSI'10), Kefalonia, Greece, July 5-7, 2010.

[16] J.L. Beuchat, E. Okamoto, and T. Yamazaki, "Compact Implemenations of Blake-32 and Blake-64 on FPGA", Cryptology ePrint Archive 2010, available at http://eprint.iacr.org/2010/0173.pdf .