BLAKE HASH Function Family on FPGA: From the Fastest to the Smallest

Nicolas Sklavos
Informatics & MM Dept., Branch of Pyrgos
Technological Educational Institute of Patras
Pyrgos, ZIP 27100, GREECE
e-mail: nsklavos@ieee.org

Paris Kitsos
Computer Science
Hellenic Open University, GREECE
e-mail: pkitsos@ieee.org

***Abstract*—Hash functions form an important category of cryptography, which is widely used in a great number of protocols and security mechanisms. SHA-2 is the up to date NIST standard, but is going to be substituted in the near future with a new, modern one. NIST has selected the Second Round Candidates of the SHA-3 Competition. A year is allocated for the public review of these algorithms, and the Second SHA-3 Candidate Conference is being planned for August 23-24, 2010, after Crypto 2010. This paper deals with FPGA implementations of BLAKE hash functions family, which is one of the finalists. In this work, a VLSI architecture for the BLAKE family is proposed. For every hash function of BLAKE (-28, -32, -48, & -64), a hardware implementation is presented. The introduced integrations are examined and compared with hardware implementation terms. Computational efficiency of SHA-3 finalists in silicon, is one of the evaluation criteria of SHA-3.**

Keywords-BLAKE, SHA-3, FPGA, VLSI, Hashing, Security

I. INTRODUCTION

Hash functions is a fundamental primitive category of security science, which is also named one-way hash functions. A hash function, in bibliography, is defined as computationally efficient function, which maps binary strings of arbitrary length to binary strings of fixed length. The last ones are the outputs of a hash computation and they are called hash values [1]. The importance of hash functions, in modern cryptography is clearly proven by the different applications and multi-purposes that these are used, in cryptographic protocols [2]. Hash functions are applied to support digital signatures, data integrity, random number generators [3], authentication schemes, and data integrity mechanisms [2]. They could also be met in every security protocol such like Transport Layer Security, Internet Protocol Security etc [2].

SHA-1 (Secure Hash Algorithm Version 1), was introduced by NIST (1995) as the first hash functions standard. The security level of that standard is limited to a level comparable to an 80-bit block cipher. Some years after, on 2002, NIST announced a new standard, SHA-2 (Secure Hash Algorithm Version 2), which consists of three new hash functions with longer hash value lengths: 256-, 384-, 512- bit [4]. Up to these days, the National Institute of Standards (NIST), has organized a competition for new hash functions designs, in order to conclude to a new standard. The competition received 64 proposed algorithms, while 51 of them progressed after the first round. The Second SHA-3 Candidate Conference will be held at the University of California, Santa Barbara, on August 23-24, 2010, following Crypto and CHES 2010. The purpose of this conference is to discuss the 14 second-round candidates, and to obtain valuable feedback for the selection of the finalists, soon after the conference. The process and the criteria for this selection will be a major topic of the conference. It has been announced by NIST that the computational efficiency of the hash functions, will be addressed, during the second round of the contest. BLAKE hash functions family [5], is one of the most promising functions, since it meets all NIST criteria for SHA-3 standard, which are mainly centered to the following ones:

- produced message digests of final size equal to 224-, 256-, 384-, and 512- bit
- similar parameter sizes to SHA-2
- streaming mode of one pass
- maximum message length of at least (2^{64}-1)-bit

In this paper, a hardware generic architecture for the BLAKE hash functions family is introduced. Based on this architecture, each one of BLAKE family functions (-28, -32, -48, & -64), has been implemented separately by using FPGAs devices, due to the different specifications and the way of operation. Explorations of the proposed designs are presented. FPGAs are a flexible choice for cryptographic engineering. One of their main advantages is that they can be re-programmed and allows rapid prototyping of the implementation designs. All the introduced hash functions implementations are examined and compared in the performance, by using hardware implementation terms. Comparisons with other related works are also given, in order to prove the superiority of the introduced designs, and any possible bottleneck of the proposed architecture.

The rest of this work is organized as follows: Section 2 presents briefly the four BLAKE hash functions. The next section introduces the proposed generic architecture of BLAKE family, in detail. Furthermore, specifications and implementation needs for every one of the four examined hash functions are also presented. FPGA synthesis results are given in the following section. Comparisons with other FPGA implementations are presented. Future work steps are proposed in the next section. The paper concludes with conclusions & outlook, which are discussed in the last section.

II. BLAKE HASH FUNCTIONS FAMILY

With the name BLAKE [5] we refer to modern hash functions family, which has been proposed to be the new SHA-3 hash functions standard. This hash functions family contains four alternative hash functions: BLAKE-28, BLAKE-32, BLAKE-48 and BLAKE-64. These functions are basically different in the word length they use for the data transformation, in the applied message block, the produced message digest, as well as in the used salt. Table I illustrates the specifications of each one of these four hash functions:

TABLE I. BLAKE HASH FUNCTIONS SPECIFICATIONS

HASH	WORD	MESSAGE	BLOCK	DIGEST	SALT
BLAKE-28	*32-bit*	*< 2^64*	*512-bit*	*224-bit*	*128-bit*
BLAKE-32	*32-bit*	*< 2^64*	*512-bit*	*256-bit*	*128-bit*
BLAKE-48	*64-bit*	*< 2^128*	*1024-bit*	*384-bit*	*256-bit*
BLAKE-64	*64-bit*	*< 2^128*	*1024-bit*	*512-bit*	*256-bit*

BLAKE family is a combination of some main components [5] introduced by the hash family introducers: i) the HAIFA iteration mode, an improved version of the Merkle-Damgrad paradigm proposed by Diham and Dunkelman, ii) the internal structure of LAKE hash function, iii) a modified version of Bernstein's stream cipher CHACHA, which is used as compression function. BLAKE-32 could be used as an example, in order to specify this hash family philosophy of operation. Similar to the previous is the operation of the other hash functions of the family, with the differences to be explained in detail in Section III.

The hash functions process of this family, is mainly based on two operations: the modular adder of 2^n, for unsigned integers, and the bit-by-bit XOR (exclusive OR) on n-bit words. In addition, the right rotation operation of k-bit is used.

The heart of the BLAKE-32 function ofr example is the compression function which operates with four inputs: i) the chain value h = h0, …, h7 , ii) the message block m = m0, …, m15 , iii) the salt s = s0, …, s3 and iv) the counter t = t0, t1. BLAKE starts hashing from the same initial value as SHA-2 [4], and uses a set of 16 32-bit constants [4]. The operation could be divided to a) Initialization Process, ii) Round Function, and iii) Finalization Process.

During Initialization Process a 16-word state v = v0, …, v15 is initialized such that different inputs produce different initial states. The state is represented as a (4 X 4) matrix. In the next step the round function takes place, based on the compression function. The later operates for a specified number of rounds. In every round, the state v is transformed based on addition, XOR and right rotation computations, which are the components of Gi (i=1,…,4) functions. For this purpose the Gi functions are used. When the round sequence is taken over, the new chain value h' = h'0, …, h'7 is produced from the state v, with inputs of the initial chain value h0, …, h7 and the salt s = s0, …, s3. Finalization Procedure absolutely operates on XORing data transformations, of the previous referred data words.

III. BLAKE HASH FUNCTIONS FAMILY

For the hardware implementation of BLAKE hash functions family a generic architectural platform has been designed and proposed. For each one of the functions the following architecture was implemented separately with all the appropriate modifications to the generic design. The generic proposed architecture is presented in the following Figure 1.

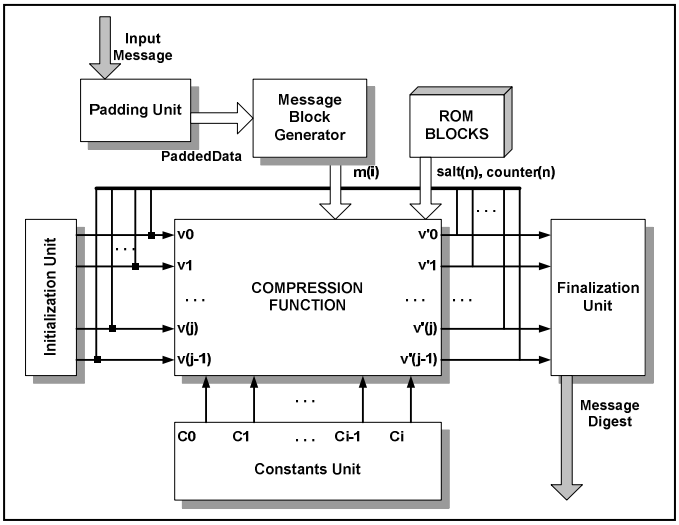

Figure 1. BLAKE proposed generic architecure

Padding Unit is used to extend the Input Message length in order to be a congruent to 447 modulo 512 for BLAKE -28, -32, and 895 modulo 1024 for BLAKE -48, -64. Length extension is achieved by appending a bit equal to logic "1", and the appropriate number of logic "0" bits. One bit at the best and 512-bit at the worst case are appended. Then logic "1" is added, followed by a 64-bit unsigned big-endian representation of the hashing message m of bit length $l<2^{64}$. Equations (1-4) show the Padding Unit transformation, which ensures that the bit length of the padded message is a multiple of 512- for BLAKE -28, -32, and 1024-bit for BLAKE -48, -64.

$$m \leftarrow m \,||\, 1000\ldots0000\ (l)_{64} \quad \text{BLAKE-28} \quad (1)$$

$$m \leftarrow m \,||\, 1000\ldots0001\ (l)_{64} \quad \text{BLAKE-32} \quad (2)$$

$$m \leftarrow m \,||\, 1000\ldots0000\ (l)_{128} \quad \text{BLAKE-48} \quad (3)$$

$$m \leftarrow m \,||\, 1000\ldots0001\ (l)_{128} \quad \text{BLAKE-64} \quad (4)$$

Data transformation is based on a certain number of rounds which are specified to 10, for BLAKE -28, -32 and 14 rounds for BLAKE -48, -64. Rounds operation is supported with a certain type of constants, which are produced by the Constants Unit. BLAKE-28 uses the eight initial values of SHA-2 (224) [5], while BLAKE-32 uses the eight initial values of SHA-2 (224) [5], and 16 32-bit constants, while BLAKE-32 uses the eight initial values of SHA-2(256) [4], and the same 16, with BLAKE-28, 32-bit constants. BLAKE -48, and -64 use the initial value of SHA-2 (384) and SHA-2 (512), respectively and both of them the same as for SHA-2 (512) initial values of 64-bit.

The required constants and initial values are produced with the same way introduced at [6], [7], from the Constants Unit.

Initialization Unit is responsible for the formation of the n-word state (n=16 in the case of BLAKE-32), such that different inputs produce different initial states. State is used as a fundamental word data structure and could be represented as a 4x4 matrix, which must be filled with a specific way, according to BLAKE hash family demands. Based on the theoretical analysis of the specifications, the output vectors v0,…,v7 is very easy to be implemented on hardware, since their integration is a matter of wiring (Fig. 2). For the implementation of v9,…,v15 a XOR chain is used, between the values of salt (s0,…,s3) and the constant c (c0,…,c7).

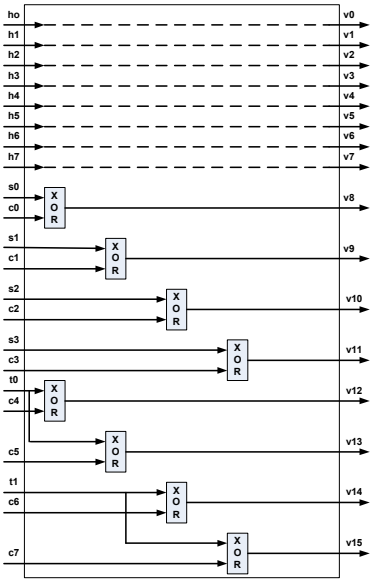

Figure 2. BLAKE-32 Initialization Unit Proposed Architecture.

Compression Function is the main core of the hash functions data transformations, for a certain number of rounds for each hash function. A round is a completely transformation of the following sets:

$$a \leftarrow a + b + \{ msr(2i) \text{ XOR } csr(2i+1) \}, \quad (5)$$

$$d \leftarrow \{ d \text{ XOR } A \} >>> 16, \quad (6)$$

$$c \leftarrow \{ c+d \}, \quad (7)$$

$$b \leftarrow \{ b \text{ XOR } c \} >>> 12, \quad (8)$$

$$a \leftarrow a + b + \{ msr(2i+1) \text{ XOR } csr(2i) \}, \quad (9)$$

$$d \leftarrow \{ d \text{ XOR } A \} >>> 8, \quad (10)$$

$$c \leftarrow \{ c+d \}, \quad (11)$$

$$b \leftarrow \{ b \text{ XOR } c \} >>> 7, \quad (12)$$

From the above equations is obvious that for each one of the variables a,…,d two district values are set during one round transformation. For each one of Gi functions an alternative implementation is proposed. All of them could be achieved based on the same generic Gi architecture, which is basically consists of n-bit modulo adders, XOR chains and, shift operations and registers. In order to implement the Gi functions for each one hash functions of BLAKE family, a Generic architecture for Gi hardware integration is proposed, which is illustrated in the following Figure 4.

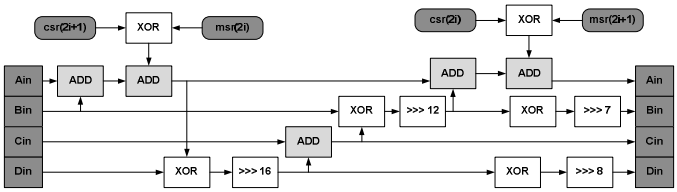

Figure 3. Gi Function Gneric Proposed Archietcture

Round transformation is based on eight different functions Gi, with i = 0,…,7. G functions, in the proposed architecture operates on data with two different ways. First operates on v matrix cell, in both column and diagonal step. In the proposed architecture the first four functions Gi are computed in parallel, due to the fact that each one of them transforms different columns of the matrix. This is taken place in column transformation. On the other hand, the next four of Gi functions, G4,…,G7 work on diagonals and can be work on parallel, in an alternative way. Figure 4 illustrates both ways of operation for all Gi functions. For BLAKE-28 the output is truncated to its first 224-bit (h'0,…,h'6) and for BLAKE-48 (h'0,…,h'5) 384-bit of the output are used. BLAKE-32 and BLAKE-64 uses all outputs (256-bit & 512-bit each one) bits.

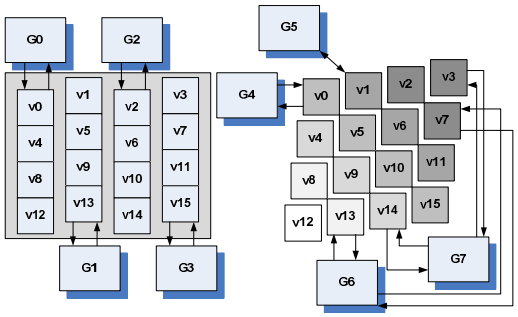

Figure 4. Gi Functions transforamtion a) column, & b) diagonal step

Finalization Unit operates on the last step and has been implemented as a chain XOR blocks, between the initial chain value h and salt s, as follows:

$$h'0 = h0 \text{ XOR } s0 \text{ XOR } v0 \text{ XOR } v8 \quad (13)$$

. . .

$$h'7 = h7 \text{ XOR } s3 \text{ XOR } v7 \text{ XOR } v15 \quad (14)$$

For this new hash functions standard, a parameter of operation must be specified, in order to allow speed confidence trade-offs. BLAKE hash functions family could be parameterized on the number of rounds. Although the achieved security level is always a factor that must be taken under consideration too, when the values of parameters are specified [4]. It has been estimated that five rounds is the minimum number that must be executed for BLAKE -28 & -32 bit, although ten rounds are recommended. For BLAKE -48, -64 fourteen rounds are recommended, although the functions could operate sufficiently with seven rounds of data transformations.

IV. IMPLEMENATION SYNTHESIS RESULTS ON FPGA

The proposed generic architecture has been used for the implementation of the four BLAKE hash functions. All of them, have been captured by using VHDL, while they were synthesized placed and routed using a XILINX FPGA Virtex device [8]. The analytical synthesis results for the introduced integrations are presented in the next Table II. Throughput is calculated as follows:

Throughput = (# Bits of Message Block X Frequency) /

(# Clock Cycles / Message Block)

TABLE II. BLAKE HASH FUNCTIONS FPGA IMPLEMENTATION RESULTS

	F (MHz)	**Throughput (10xMbps)**	**Area (Slices)**
BLAKE-28	*52*	*116,48*	*3017*
BLAKE-32	*50*	*128,00*	*3101*
BLAKE-48	*28*	*76,80*	*10986*
BLAKE-64	*27*	*98,74*	*11800*

Furthermore, the proposed FPGA integrations are compared in the terms of frequency (MHz), throughput (Mbps) and area-delay product (slices x MHz), order to provide more detailed and fair comparison for each functions of BLAKE family (Figure 5).

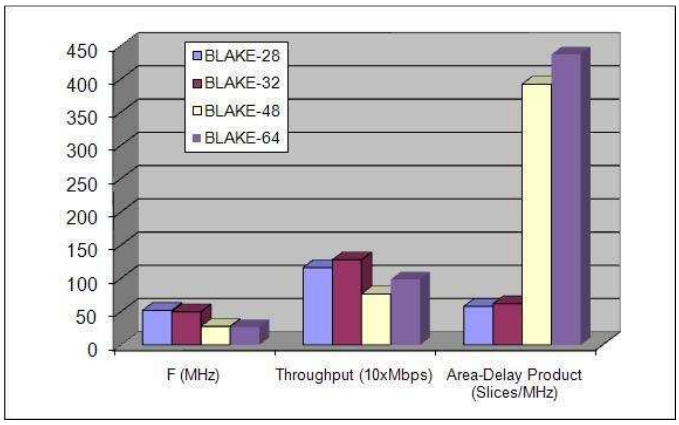

Figure 5. BLAKE Family Comparisons Graph

In addition, comparisons with other related works [6] of the previous standard, as well as with proposal for the new one [9], are presented, in order to provide a more detailed point of view of the proposed Generic Architecture (Table IV).

V. FUTURE WORK

Based on the proposed Generic Architecture for the implementation of each one of the hash function of BLAKE family, a proposed architecture that would support a multi-mode operation in the sense that it performs all the four hash functions (28, 32, 48, & 64 -bit) of BLAKE family could be design and proposed. The proposed system must be compared with the implementation of each hash function in a separate FPGA device.

TABLE III. PROPOSED BLAKE IMPLEMEANTIONS COMPARISONS

INTEGRATIONS	***AREA RESOURCES***	***F (MHz)***	***RATE (Mbps)***
SHA-2	*1060, 1966, 2237*	*83, 74, 74*	*326, 350, 480*
CubeHash	*1178*	*167*	*0,160 Gbps*
Crostl	*5878*	*128*	*3,28 Gbps*
LANE	*3442*	*133*	*1,38 Gbps*
Shabal	*2307*	*222*	*1,33 Gbps*
Spectral Hash	*951*	*125*	*0,18*
Prop. BLAKE-28	*3017*	*52*	*116,48*
Prop. BLAKE-32	*3101*	*50*	*128,00*
Prop. BLAKE-48	*10986*	*28*	*76,80*
Prop. BLAKE-64	*11800*	*27*	*98,74*

VI. CONCLUSIONS & OUTLOOK

This work deals with the FPGA implementations of BLAKE hash family functions. The proposed hardware integrations are compared based on a generic architecture in hardware terms. Comparisons with additional criteria such as area-delay products, as well as related published works are also presented. BLAKE hash functions family [10], [11] is a latest proposal finalist, for the competition of the new SHA-3.

REFERENCES

[1] 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.

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

[3] SHA-2 Standard, National Institute of Standards and Technology (NIST), Secure Hash Standard, FIPS PUB 180-2, www.itl.nist.gov/fipspubs/fip180-2.htm

[4] J. P. Aumasson, L. Henzen, W. Meier, and R.C.W. Phan, "SHA-3 Proposal BLAKE", Online: http://www.131002.net/blake 2010.

[5] N. Sklavos, O. Koufopavlou, "On the Hardware Implementations of the SHA-2 (256, 384, 512) Hash Functions", proceedings of IEEE International Symposium on Circuits & Systems (IEEE ISCAS'03), Vol. V, pp. 153-156, Thailand, May 25-28, 2003.

[6] N. Sklavos, O. Koufopavlou, "Implementation of the SHA-2 Hash Family Standard Using FPGAs", Journal of Supercomputing, Springer-Verlag, Vol. 31, No 3, pp. 227-248, 2005.

[7] N. Sklavos, P. Kitsos, K. Papadomanolakis and O. Koufopavlou, "Random Number Generator Architecture and VLSI Implementation", proc. of IEEE International Symposium on Circuits & Systems (ISCAS'02), USA, 2002.

[8] Xilinx, San Jose, California, USA: 'Virtex, Field Programmable Gate Arrays', www.xilinx.com, 2010.

[9] B. Baldwin, A. Byrne, M. Hamilton, N. Hanley, R. P. McEvoy, W. pan, W. P. Marnane, "FPGA Implemeantions of SHA-3 Candidates: CubeHash, Grostl, LANE, Shabal, Spectral Hash", proceedings of 12th EUROMICRO Conferecne on Digital System Desgin: Architetcures, Methids, and Tools, August 27-29, Patras, Greece, 2009.

[10] Stinson, D.R.: 'Cryptography: Theory and Practice', CRC Press LLC, 1995.

[11] Menezes, A., Oorchot, P.Van., and Vanstone. S.: 'Handbook of Applied Cryptography', CRC Press, Inc, 1997.