2013 10th International Multi-Conference on Systems, Signals & Devices (SSD)
Hammamet, Tunisia, March 18-21, 2013

# An FPGA implementation of the SHA-3: The BLAKE Hash Function

Fatma Kahri, Belgacem Bouallegue, Mohsen Machhout and Rached Tourki
Electronics and Micro-Electronics Laboratory (E. µ. E. L)
Faculty of Sciences of Monastir, Tunisia
Kahrifatma@gmail.com

***Abstract*— Following the attacks considerable standard SHA-2, In this paper, a new version of hash was developed known as SHA-3. We discussed the study of the SHA-3 hash exposing the protocol chosen for our is BLAKE-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. A comparison between our implementation SHA3 and the Blake 256; 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 BLAKE-256 was developed using VHDL, and synthesized using Virtex-5, Virtex-6 and Virtex-7 chips. Blake-256 show tremendous throughput increase of 179% when compared with the implementation of the original Blake -256.**

***Index Terms*— SHA-2, SHA-3, BLAKE, FPGA, Hardware.**

## I. Introduction

Hash functions are widely used in modern security protocols and applications like digital signatures, message authenticity, and password protection.

The popular SHA-1 algorithm has been seriously weakened [1, 2, 3]. Following the weakening of the widely-used SHA-1 hash algorithm and concerns over the similarly-structured algorithms of the SHA-2 family. Consequently, NIST has decided to develop one or more additional hash functions through a public competition in order to specify its future hash standard SHA-3.

Following the official call for submissions in November 2007, 64 algorithms were proposed for SHA-3 in October 2008. The meant timeline of the competition aims at the selection of a winner in the middle of 2012 after two evaluation rounds of roughly 18 months each. In round three of the SHA-3 Contest, 5 finalists remain for consideration, and their results have been published [4].

In order to estimate the suitability of some promising SHA-3 candidates, we have implemented BLAKE. The hardware module has been evaluated in regard to here size, throughput, and frequency. With this work we hope to make a valuable contribution to the hardware evaluation effort for the SHA-3 competition.

This paper is structured as follows. First, we state the general properties of our hardware modules, wish are themselves are described. The implementation and experimental results are shown and discussed in second part. Finally, we conclude the efficient and interesting SHA-3.

## II. Background

Some descriptions of SHA-1, SHA-2 and SHA-3 algorithms can be found in the official NIST standard [5]. Table 1 shows a comparative study of four hash functions characteristics.

The security of these hash functions is controlled by the size of their outputs, referred to as hash values. All functions have a similar internal structure and process each message block using multiple rounds. . These hash functions enable the determination of a message's integrity: any change to the message will result in a different produced message digest, with a very high probability.

TABLE I. Functional characteristics of four hash functions

| **Hash function** | **SHA-1** | **SHA-2** | **SHA-3** |
|---|---|---|---|
| Constants Kt number | 4 | 64 | 16 |
| Size of hash value (n) | 160 | 256 | 256 |
| Complexity of the best attack | $2^{80}$ | $2^{128}$ | - |
| Message size | $<2^{64}$ | $<2^{64}$ | |
| Message block size (m) | 512 | 512 | 512 |
| Word size | 32 | 32 | 32 |
| Numbers of words | 5 | 8 | |
| Digest rounds number | 80 | 64 | 10 |

## III. Secure Hash Functions Blake 256

In the following part, we briefly describe the main concepts used in hash function Blake 256 and the general design choices we have taken for our hardware implementations.

BLAKE is our candidate for SHA-3. We did not reinvent the wheel; BLAKE is built on previously studied components, chosen for their complementarily.

### *A. Algorithme Description*

This section defines BLAKE-256, going from its constant parameters to its compression function. The Blake-32 compression function works on an internal state of 512 bits, represented as a (4 × 4) matrix of 32-bits words Figure 2 shows the matrix

$$\begin{bmatrix} v_0 & v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 & v_7 \\ v_8 & v_9 & v_{10} & v_{11} \\ v_{12} & v_{13} & v_{14} & v_{15} \end{bmatrix} \leftarrow \begin{bmatrix} h_0 & h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 & h_7 \\ s_0 \oplus c_0 & s_1 \oplus c_1 & s_2 \oplus c_2 & s_3 \oplus c_3 \\ t_0 \oplus c_4 & t_0 \oplus c_5 & t_1 \oplus c_6 & t_1 \oplus c_7 \end{bmatrix}$$

Fig. 1. Matrix of 32 bits

With

- $v = v_0 \ldots v_7$: initial value as SHA-256
- $c = c_0 \ldots c_{15}$: constants
- $h = h_0 \ldots h_7$: chain value
- $t = t_0, t_1$: counter
- $s = s_0 \ldots s_3$: salt

One time the state v is initialized; the compression function iterates a series of 14 rounds. A round is a conversion of the state v that computes

$G_0$ ($v_0$, $v_4$, $v_8$, $v_{12}$), $G_1$ ($v_1$, $v_5$, $v_9$, $v_{13}$), $G_2$ ($v_2$, $v_6$, $v_{10}$, $v_{14}$)
$G_3$ ($v_3$, $v_7$, $v_{11}$, $v_{15}$), $G_4$ ($v_0$, $v_5$, $v_{10}$, $v_{15}$), $G_5$ ($v_1$, $v_6$, $v_{11}$, $v_{12}$),
$G_6$ ($v_2$, $v_7$, $v_8$, $v_{13}$), $G_7$ ($v_3$, $v_4$, $v_9$, $v_{14}$).

Where, at round r, $G_i$ (a, b, c, d) sets:

$$\begin{aligned} &a \leftarrow a + b(m_{\sigma r(2i)} \oplus c_{\sigma r(2i+1)}) \\ &d \leftarrow (d \oplus a) >>> 16 \\ &c \leftarrow c + d \\ &b \leftarrow (b \oplus c) >>> 12 \\ &a \leftarrow a + b + (m_{\sigma r(2i+1)} \oplus c_{\sigma r(2i)}) \\ &d \leftarrow (d \oplus a) >>> 8 \\ &c \leftarrow c + d \\ &b \leftarrow (b \oplus c) >>> 7 \end{aligned} \qquad (1)$$

The first four calls $G_0 \ldots G_3$ can be used in computers, for the reason that each of them updates a different column of the matrix. We call the method of computing $G_0, \ldots, G_3$ a column step.
Also, the last four calls $G_4, \ldots, G_7$ update diverse diagonals thus can be parallelized as well, which we call a diagonal step.

A single G makes six additions modulo $2^{32}$, six XORs and four individual word rotations by a fixed distance. Figure 2 shows the G function for index i. A single round consists of eight invocations of the G function: Four on the columns of the state and four on the diagonals of the state. A totality of ten rounds is executed.

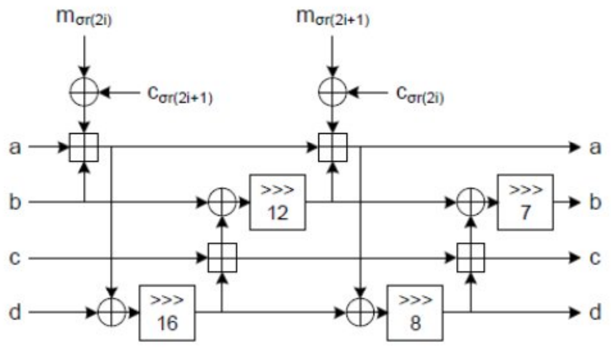

Fig. 2. The $G_i$ function.[12]

Later than the rounds sequence, the new chain value $h'_0, \ldots, h'_7$ With input of the initial chain value h0, . . . , h7 and the salt s0, . . . , s3: The finalization takes the output of the ten rounds and combines it with the input chaining value and the salt.

$$\begin{aligned} &h'_0 \leftarrow h_0 \oplus s_0 \oplus v_0 \oplus v_8 \\ &h'_1 \leftarrow h_1 \oplus s_1 \oplus v_1 \oplus v_9 \\ &h'_2 \leftarrow h_2 \oplus s_2 \oplus v_2 \oplus v_{10} \\ &h'_3 \leftarrow h_3 \oplus s_3 \oplus v_3 \oplus v_{11} \\ &h'_4 \leftarrow h_4 \oplus s_0 \oplus v_4 \oplus v_{12} \\ &h'_5 \leftarrow h_5 \oplus s_1 \oplus v_5 \oplus v_{13} \\ &h'_6 \leftarrow h_6 \oplus s_2 \oplus v_6 \oplus v_{14} \\ &h'_7 \leftarrow h_7 \oplus s_3 \oplus v_7 \oplus v_{15} \end{aligned} \qquad (2)$$

### *B. BLAKE-256 hach function*

The whole hash function operation is divided in two stages: (1) pre-processing and (2) hash computation. Pre-processing involves padding the input message, parsing the padded data into a number of m-bit blocks (m = 512) and setting the appropriate initial values, which are used in the hash computation. The calculation of hash functions requires the use functions applied to the padded data, constants and word logical and algebraic operations, to generate iteratively a series of hash values. After a specified number of transformation rounds the produced hash value turns becomes equal to the message digest.

#### *1) Pre-Processing:*

The n-bit message is padded so that a single 1-bit is added into the end of the message. Then, 0-bits are added until the length of the message is congruent to 448 modulo 512.
A 64-bit representation of n is appended to the result of the padding. Thus, the result message is a multiplicity of 512 bits. This message is denoted here as M(i). M(i) message blocks are passed individually to the message expander. Padding can be represented as:

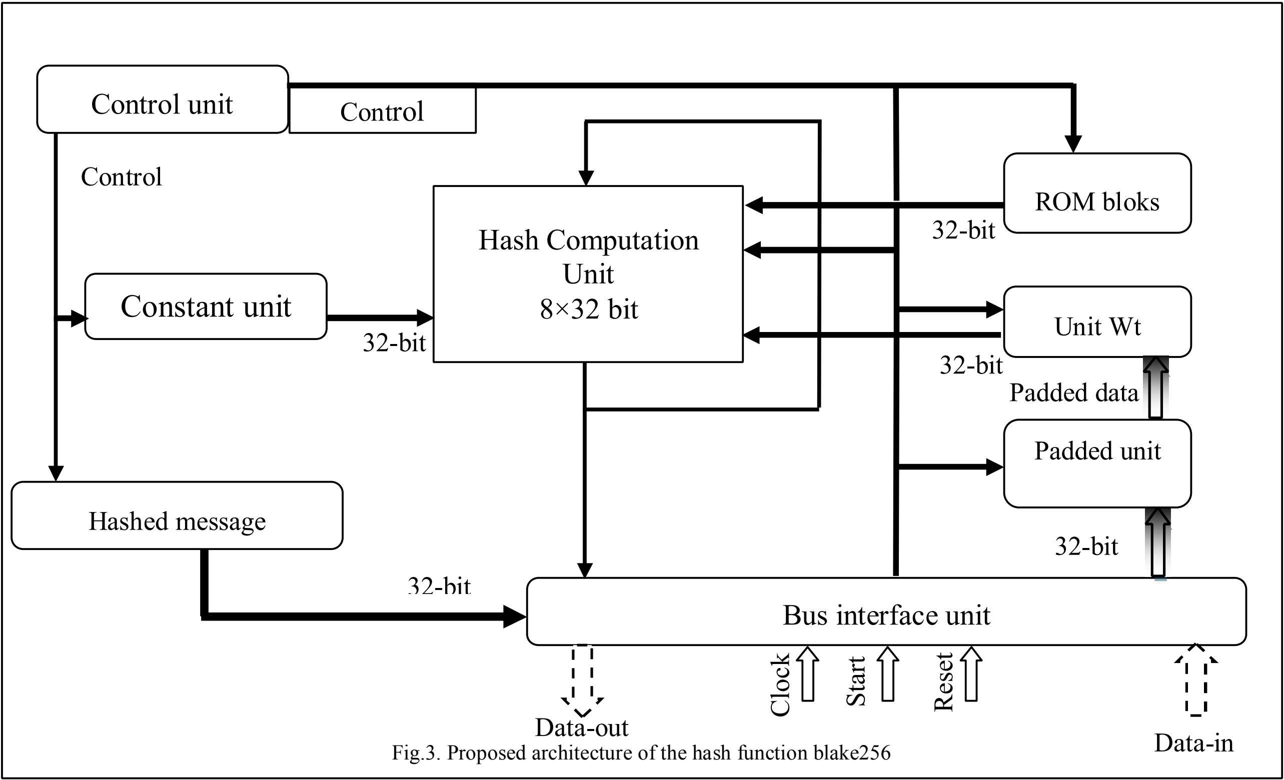

Fig.3. Proposed architecture of the hash function blake256

$$m \leftarrow m \,||\, 1000...0001 < 1 >_{64} \qquad (3)$$

*2) Hash computation:*

To determine hash of a message superior than 512-bits (16 32-bit words), BLAKE-256 compression function is used iteratively as follows:

```
h_0←v
for i=0,…,N-1
  h^{i+1}← compress(h^i, m^i ,s,l^i)
return h^n
```

## IV. Blake-256 Processor Design Implementation And Comparison

### A. Design processor

This section presents the architectural design of our programmable BLAKE-256 processor, our implementations.

A high- level block based of our proposed processor is shown in figure 2. The given architecture supports four operation modes for reconfigurable BLAKE-256 processor.

- A Bus Interface Unit has been integrated in order for the proposed processor to communicate efficiently with the external environment.
- The Control Unit is designed to control the flow of data in the design, as well as data exchange between the Padded procedure Unit and Hash Computation Unit. A FSM is used for this function.
- Padded Process Unit pads the input data messages and converts them to 512 bit.
- The Hash Computation Unit is the principal data path component of the system architecture. BLAKE -256 requires 14 cycles to produce the 256-bit message digests. Each cycle requires the previous round's, as well as the constant value $C_i$, the core utilize eight 32-bit words: a-d, wish are initialized to predefined values IV0(0)… IV7(0),[12] at the start of each call to the hash function.

### B. FPGA IMPLEMENTATION AND COMPARAISON

In this section, we present the implementation of blake-256 VHDL it is used as the hardware description language thanks to the flexibility to exchange among environments. The code is pure VHDL that could easily be implemented on other devices, without changing the design. The software used for this work is Xilinx - Project Navigator, ISE 14.1 suite. This is used for writing, debugging and optimizing efforts, and also for fitting, simulating and checking the performance results using the simulation tools available on ModelSim 6.1 software.

In the design of the hash function architectures described in this paper, *the goal* was to give a baseline comparison between the hash functions using area and throughput. We calculate the throughput as follows:

$$Throughput = \frac{block\_size}{T.(HTime(N+1) - HTime(N))} \qquad (4)$$

Where block_size is a message block size, characteristic for each hash function, HTime (N) is a total number of clock cycle necessary to hash an N-block message, T is a clock period.

Table 2 shows that the number of occupied slices decreases depending on the platform used, according to the circuit using FPGA there is a change in frequency. The increase in the frequency leads to an increase of the dynamic power with a high Throughput.

TABLE II. RESULTS FOR BLAKE -256

| | **Reference** | | |
|---|---|---|---|
| | ***VIRTEX 5 (65nm)*** | ***VITREX 6 (40nm)*** | ***VIRTEX 7 (28 nm)*** |
| Area | 691 (6%) | 594 (1%) | 522 (<1%) |
| Frequency (Mhz) | 79 | 81 | 82 |
| Throughput (Gbps) | 2.53 | 2.59 | 2.62 |
| Efficiency (Gbps/slices) | 3.66 | 4.42 | 4.97 |

The results of the implementations are shown in Table III. The design has been implemented on Xilinx Virtex 5, Virtex 6 and Virtex 7 FPGAs. Detailed device specifications are: Virtex 5 155T, speed grade 3, package FF1136 (5vlx155tff1136-3), Virtex 6 LX365T, speed grade 3, package FF1156 (6vlx365tff1156-3) and Virtex 7 585T, speed grade 3, package FG1157 (7v585tffg1157-3), respectively.

The resulting clock frequencies and area utilization after place and route are reported. Table II shows achieved area consumption, clock frequency, throughput, and efficiency for implemented design. Throughput comparisons amongst FPGA-based BLake-256 designs have recently been drawn in the literature [11, 7]. The highest reported data throughput for BLAKE-256 was 0.64 Gbps [9], 1.69 Gbps [12] and 2.47 [11] and using 1660, 1247 and 1566 slices respectively. For completeness, synthesis of our designs on the Virtex platform resulted in BLAKE-256 processor with a throughput of 2.53 Gbps (691 slices) in Vertex5, 2.59 Gbps (522 slices) in Virtex-6 and a throughput of 2.62 Gbps (594 slices).

The Efficiency decreased by 179% in all Virtex; we calculate Efficiency as follow:

$$Efficiency = \frac{Throughput}{Area} \qquad (5)$$

## V. CONCLUSION

In this paper we have presented efficient hardware implementation of SHA-3: BLAKE-256. SHA-3 secure hash algorithm is the newest hash function. We reported the implementation results of BLAKE-256 on Xilinx Virtex 5, Virtex 6 and Virtex 7 FPGAs. We reported the performance of our implementation in terms of area, throughput, frequency and efficiency and compared it with previously reported implementation results.

TABLE III. COMPARISON OF BLAKE-256 IMPLEMENTATION

| **References** | **Area (Slices)** | **Frequency (MHz)** | **Throughput (Gbps)** | **Efficiency(Gbps/slice)** |
|---|---|---|---|---|
| Virtex 5 (5vlx155tff1136-3) [7] | 1739 | 124.55 | 2.28 | 1.31 |
| Virtex 5[8] | 1653 | 91.35 | 0.83 | 0.50 |
| Virtex 5[9] | 1660 | 115 | 0.64 | 0.38 |
| Virtex 5[10] | 1871 | 117 | 2.07 | 1.1 |
| Virtex 5 [11] | 1691 | - | 2.25 | 1.33 |
| Virtex 5 (our work) | 691 | 79 | 2.53 | 3,66 |
| Virtex 6 (6vlx365tff1156-3) [11] | 1602 | 132 | 2.41 | 1.51 |
| Virtex 6[12] | 1247 | - | 1.96 | 1.57 |
| Virtex 6 (our work) | 522 | 81 | 2.59 | 4.97 |
| Virtex 7 (7v585tffg1157-3) [11] | 1566 | 135.35 | 2.47 | 1.58 |
| Virtex 7 (our work) | 594 | 82 | 2.62 | 4.42 |

## REFERENCES

[1] Jean-Philippe Aumasson, Luca Henzen,Willi Meier, and Raphael C.-W. Phan. SHA-3 proposal BLAKE, version 1.3. Available online at http://131002. net/blake/blake.pdf, 2008.

[2] Christophe De Canniére and Christian Rechberger. Findin SHA1 Characteristics: General Results and Applications. In Xuejia Lai and Kefei Chen, editors, Advances in Cryptology - ASIACRYPT 2006, 12th International Conference on the Theory and Application of Cryptology and Information Security, Shanghai, China, December 3-7, 2006, Proceedings, volume 4284 of Lecture Notes in Computer Science, pages 1–20. Springer, 2006.

[3] National Institute of Standards and Technology (NIST). FIPS-180-3: Secure Hash Standard, October 2008. Available online at http://www.itl.nist.gov/fipspubs/.

[4] SHA-3 Hardware Implementations, http://ehash.iaik.tugraz.at/wiki/SHA3Hardware_Implementations

[5] National Institute of Standards and Technology, "Secure Hash Standard", Federal Information Processing Standards 180-2, August 2002.

[5] National Institute of Standards and Technology, "Secure Hash Standard", Federal Information Processing Standards 180-2, August 2002.

[6] Kashif Latif Athar Mahboob Arshad Aziz"High Throughput Hardware Implementation of Secure Hash Algorithm (SHA-3) Finalist: BLAKE" Frontiers of Information Technology 2011

[7] B. Baldwin, N. Hanley, M. Hamilton, L. Lu, A. Byrne, M. Neill and W. Marnane "FPGA Implementations of the Round Two SHA-3 Candidates", 2nd SHA-3 Candidate Conference, Santa Barbara, August 23-24, 2010, pp. 1-18.

[8] S. Matsuo, M. Knezevic, P. Schaumont, I. Verbauwhede, A. Satoh, K. Sakiyama and K. Ota, "How Can We Conduct Fair and Consistent Hardware Evaluation for SHA-3 Candidate? " 2nd SHA-3 Candidate Conference, Santa Barbara, August 23-24,2010,pp.1-15.

[9] Kris Gaj, Ekawat Homsirikamol, and Marcin Rogawaski "Fair and comprehensive Methodologiy for Comparing Hardware Performance of Fourteen Round Two SHA-3 Candidats using FPGAs" in Proceedings of Cryptographic Hardware and Embedded Systems workshop, CHES 2010, Santa Barbara, Aug. 2010.

[10] Kashif Latif Athar Mahboob Arshad Aziz"High Throughput Hardware Implementation of Secure Hash Algorithm (SHA-3) Finalist: BLAKE" Frontiers of Information Technology 2011.

[11] 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, Tallinn, Estonia,May 19-20, 2011, pp. 1-15

[12] J;Philippe Aumasson, L Henzen W. Meier, SHA-3 proposal BLAKE varsion 1.3, decembre 16, 2010