# FPGA Based Area And Throughput Implementation of JH And BLAKE Hash Function

Vaibhav Doshi
Amity School of Engineering and Technology
Amity University Rajasthan, Jaipur

Richa Arya
Amity School of Engineering and Technology
Amity University Rajasthan, Jaipur

Rajesh Kumar Yadav
Amity School of Engineering and Technology
Amity University Rajasthan, Jaipur

*Abstract*--**Implementation of area and throughput of the main building block (compression function) for two SHA-3 candidates BLAKE and JH hash function. The National Institute of Standards and Technology (NIST) has started a competition for a new secure hash standard. A significant comparison between the submitted candidates is only possible, if third party implementations of all proposed hash functions are provided[5]. The BLAKE family contains the four hash function BLAKE-28, BLAKE-32, BLAKE-48, BLAKE-64 with the bit length of their digests being 224,256,384 and 512 , respectively and JH contains JH-224, JH-256, JH-384, JH-512.We use the proposed block diagram of JH and BLAKE compression function and find fixed size message digest from binary string of arbitrary length. The compression function of BLAKE-256 takes as the input four values chain value(h) , message block(m), salt(s) and counter(t). AES design methodology is used in JH design and the BLAKE hash function HAIFA iteration mode[7]. In this paper FPGA implementation is based on two cryptographic hash function candidates BLAKE[6] and JH. and also compare the functions which is used in JH and BLAKE and then extract the important advantages, limitations, algorithms and design principals of both candidates.**

**Keyword:** SHA-3, JH , BLAKE , Hash, Compression Function.

## I INTRODUCTION

Hash functions are an essential type of cryptography, which is generally used in protocols and security mechanisms. It is defined as computationally efficient function, which maps binary strings of arbitrary length to binary strings of fixed length and finally the outputs of a hash computation ,called hash values. NIST has selected the Second Round Candidates of the SHA-3 Competition[5]. In the present paper, we focus on the new SHA-3 competition, started by the National Institute of Standards and Technology (NIST), which searches for a new hash function in response to security concerns regarding the previous hash functions SHA-1[10] and the SHA-2[11] family. This competition requires third party software and hardware implementations of all proposed candidates to evaluate the overall performance and resource requirements. The Second SHA-3 Candidate Conference will be held at the University of California, Santa Barbara, on

August 23-24, 2010. The purpose of this conference is to discuss the 14 second round candidates[6] , and to obtain valuable feedback for the selection of the finalists. Techniques are used in the design of JH [2]. We proposed a new compression function structure to construct a compression function from a block cipher with constant key. JH compression function is constructed from a permutations , round function and bijective function (a block cipher with constant key). The block size of the block cipher is 2m bits, 2m-bit hash value $H^{(i-1)}$ and the m-bit message block $M^{(i)}$ are compressed into the 2m-bit $H^{(i)}$. Message digest size is at most m bits. The resulting digest is the first 224, 256, 384 or 512 bits from the 1024-bit final value[14]. The BLAKE family contains the four hash function BLAKE-28, BLAKE-32, BLAKE-48, BLAKE-64 with the bit length of their digests being 224,256,384 and 512 , respectively The compression function of BLAKE-256 takes as the input four values chain value(h), message block(m) , salt(s) and counter(t).These functions are basically different in word length they use for the data transformation, in the applied message block, the produced message digest, as well as in the used salt. The former to operate on 32-bit words, used for computing hashes up to 256 bits long , while the latter two work with 64-bit words , used for computing hashes up to 512 bits long.

The rest of this work is organized as follows: a review of functions which are used in compression function of JH and BLAKE , general design approach and summary of hash algorithm of both candidates in Section II to Section III. Section IV contains a comparison between the common features of both candidates and Section V contains simulation work, some concluding remarks are discussed in Section VI.

## II JH HASH ALGORITHM

Hongjun Wu has proposed the JH algorithm [4]. The compression process starts by exclusive-oring the input message to the first part of the previous hash. After regrouping the bits, there exist 42 rounds of the round

function. The round function uses the generalized AES design methodology and is made of three separate layers: S-Box, Linear transformation (L), and Permutation (P). There exist two different S-boxes inside the S-box layer. The parameter round constant ($C_r$) determines which of the S-boxes is used for each round. The linear transformation layer uses multiplication over the finite field GF($2^4$) using the irreducible polynomial $x^4 + x + 1$. The permutation layer is similar to the row rotation in the AES. The next steps in compression are one layer of S-Box and a de-grouping module. At the end, the msb part of the state variable is exclusive-ored with the input to create the output hash. The general block diagram used for the hardware implementation of JH-256 is shown in Fig. 1. The message M is expanded to a multiple of 512 bits, where the message M is padded with at least 512 bits. The round constant vector $C_r$ is loaded and is used later in the round function $R_d$. Next the bijective function $E_d$ is executed, which consists of a grouping function, the round function $R_d$, a trailing substitution layer $S_t$, as well as a de-grouping function. The round function $R_d$ is executed 42 times and contains two substitution-permutation networks and two linear transformations (L, $L_{Cr}$) that implement a (4, 2, 3) maximum distance separable (MDS) code over GF($2^4$). The substitution layer S within $R_d$ as well as the trailing substitution layer $S_t$ contain two 4-bit S-boxes($S_1$,$S_2$).

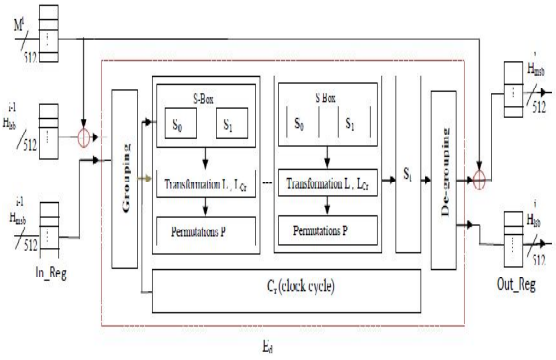



Figure 1:Proposed structure of JH-256

The round constant vector $C_r$ selects which S-boxes are used in the substitution steps and is updated in every round of $R_d$. After each invocation of $R_d$ , the value of $C_r$ it updated by passing it through the substitution-permutation network , and the linear transformation $L_{Cr}$. The last step in $F_d$ is an XOR operation of $M^i$ with the second half of H. The remaining 512-bit message blocks run through the same compression procedure to iteratively generate the hash value. After all message blocks have been processed, the n-bit message digest of M is composed of the last n bits of H.

IMPLEMENTATION of JH

Our implementation of JH-256 works with 320 instances of a combinational implementation of the S-Boxes; 256 S-boxes work on the H and 64 S-boxes work on the round constant vector $C_r$ in every round of $R_d$. This way, one round of $R_d$ can be executed in only one clock cycle. The data path of our implementation is illustrated in Figure 2.

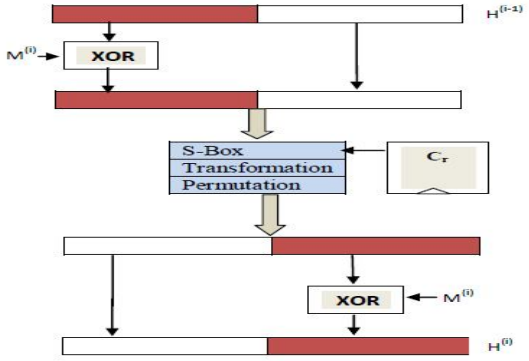



Figure 2: Proposed structure of data path of JH

In our implementation of JH the registers for the 1024-bit internal state H, the 512-bit message block $M^i$, and the 256-bit round constant $C_r$ occupy approximately one quarter of the whole area. The 320 combinational S-boxes occupy another quarter of the area.

III BLAKE HASH ALGORITHM

The BLAKE family of hash functions has been designed by Aumasson et al. [6] and follows the concept of the HAsh Iterative FrAmework" (HAIFA) [17]. Two versions of BLAKE are available: a 32-bit version (BLAKE-32) for message digests of 224 bits and 256 bits, and a 64-bit version (BLAKE-64) for message digests of 384 bits and 512 bits. BLAKE uses the local wide-pipe strategy and operates on a large inner state v that is represented as a 4X4 matrix shown..

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

Basically, the compression function consists of three steps: initialization, round updates, and finalization shown in diagram..

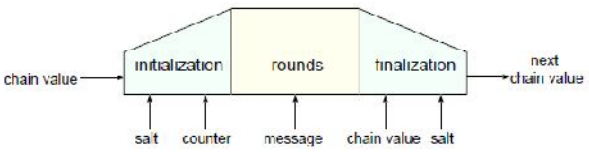



Figure 3: The local wide-pipe construction of BLAKE's compression function

During the first step, the inner state v is initialized from $h_i$ , s and t. Afterwards, v is updated several times by using a message-dependent round function. In the last step, v is compressed and the next chaining value $h_{i+1}$ is computed. The round function is based on the stream cipher ChaCha [16] and consists of the eight round-dependent transformations $G_0,..,G_7$.
$G_0(v_0 , v_4 , v_8 , v_{12})$ $G_1(v_1 , v_5 , v_9 , v_{13})$ $G_2(v_2 , v_6 , v_{10}, v_{14})$
$G_3(v_3 , v_7 , v_{11}, v_{15})$ $G_4(v_0 , v_5 , v_{10}, v_{15})$ $G_5(v_1 , v_6 , v_{11}, v_{12})$
$G_6(v_2 , v_7 , v_8 , v_{13})$ $G_7(v_3 , v_4 , v_9 , v_{14})$

All $G_i$s are derived from a single transformation operation G which is parameterized by a **permutation table σ** shown below..

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\sigma_0$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| $\sigma_1$ | 14 | 10 | 4 | 8 | 9 | 15 | 13 | 6 | 1 | 12 | 0 | 2 | 11 | 7 | 5 | 3 |
| $\sigma_2$ | 11 | 8 | 12 | 0 | 5 | 2 | 15 | 13 | 10 | 14 | 3 | 6 | 7 | 1 | 9 | 4 |
| $\sigma_3$ | 7 | 9 | 3 | 1 | 13 | 12 | 11 | 14 | 2 | 6 | 5 | 10 | 4 | 0 | 15 | 8 |
| $\sigma_4$ | 9 | 0 | 5 | 7 | 2 | 4 | 10 | 15 | 14 | 1 | 11 | 12 | 6 | 8 | 3 | 13 |
| $\sigma_5$ | 2 | 12 | 6 | 10 | 0 | 11 | 8 | 3 | 4 | 13 | 7 | 5 | 15 | 14 | 1 | 9 |
| $\sigma_6$ | 12 | 5 | 1 | 15 | 14 | 13 | 4 | 10 | 0 | 7 | 6 | 3 | 9 | 2 | 8 | 11 |
| $\sigma_7$ | 13 | 11 | 7 | 14 | 12 | 1 | 3 | 9 | 5 | 0 | 15 | 4 | 8 | 6 | 2 | 10 |
| $\sigma_8$ | 6 | 15 | 14 | 9 | 11 | 3 | 0 | 8 | 12 | 2 | 13 | 7 | 1 | 4 | 10 | 5 |
| $\sigma_9$ | 10 | 2 | 8 | 4 | 7 | 6 | 1 | 5 | 15 | 11 | 9 | 14 | 3 | 12 | 13 | 0 |

Each $G_i$ processes four words of the inner state v and is composed of modular additions, XOR, and shift operations. First, $G_0,...,G_3$ are applied in parallel on the four vertical columns of v. Consecutively, $G_4,...,G_7$ are applied in parallel on the four disjoint diagonals of v. BLAKE-32 uses 10 iterations of the round function and BLAKE-64 uses 14 iterations.

## IMPLEMENTATION of BLAKE

We have implemented various design approaches of BLAKE-32 and evaluated them with respect to the maximum achievable throughput. Results have shown that the best performance is obtained by implementing four parallel instances of the transformation operation G. Hence, two clock cycles are required for computing one round of BLAKE-32.The finalization step, which produces the next chaining value, by one clock cycle additionally increases the performance.

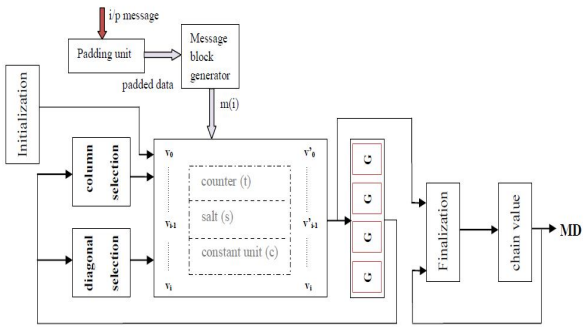



Figure 4:Proposed generic structure of BLAKE

The efficiency of this approach in terms of throughput per area can be further improved by splitting the functionality of each G operation into two equal parts and implementing only one of them in hardware. After ten rounds of transformation, the next chain value ($h_i$) is extracted from the state value in the finalization stage.

## IV COMPARATIVE RESULT OF COMMON FEATURES OF BOTH CANDIDATES

| FACTOR | JH | BLAKE |
|---|---|---|
| Structure | Iterative | HAIF mode |
| Hash variants | Single design | Two variants |
| Type | Block cipher based | Add-XOR-Rotate |
| Resistant to length extension | Each message block is 64 bytes | Message length limited to respectively $2^{64}$ and $2^{128}$ BLAKE-256 and BLAKE-512 |
| Interface | Interfacing with initial values | Interfacing with salt |
| Security | Easy to analyze and large security margin | Based on intensively based component(ChaCha) resistant to generic second-preimage attacks |
| Performance | Efficiently implemented, simple component and high parallelizability | Fast in both software and hardware, parallelism and throughput/area |
| Fast in both software and hardware | 16.8 cycles/byte on 64 bit core 2 microprocessor, 21.3 cycles/byte on 32 bit core 2 microprocessor Bit-slice, suitable for 128 bit SIMD instruction set | Intel core 2 Duo,BLAKE-256 can hash at about 15 cycle/byte and BLAKE-512 can hash at about 10 cycle/byte |
| Padding scheme | 1,0's until congruent(384mod512) 128 bit message length, min 512 bits added | 1,0's until congruent(448mod512),64 bit message length for BLAKE-224.<br>1,0's until congruent(447mod512),1,64 bit message length for BLAKE-256.<br>1,0's until congruent(895mod1024),128 bit message length for BLAKE-384.<br>1,0's until congruent(894mod1024),1,128 bit message length for BLAKE-1024. |

**Table 1:** Common features of both candidates

## V SIMULATION WORK

The proposed architectures have been used for the implementation of both hash functions and different architecture of BLAKE. All of them, have been captured by using VHDL, Stratix III FPGA family from Altera. The analytical synthesis results for the introduced integrations are

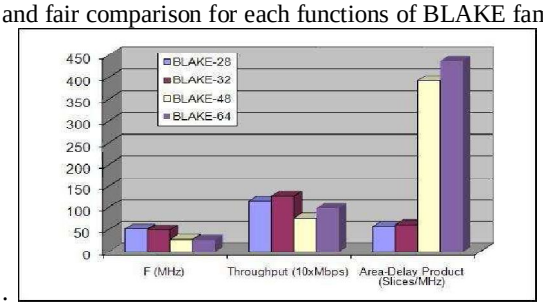
presented in the next Table 2. Throughput is calculated as follows:

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

**Table 2:**

| | 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 fig.5




.

Figure 5: BLAKE Family Comparisons Graph

**Table 3:**

| Architecture | Function | Area [KGE] | Latency [cycles] | Throughput [Mbps] | Efficiency [Kbps/GE] |
|---|---|---|---|---|---|
| [8G] | BLAKE-32<br>BLAKE-64 | 58.30<br>132.47 | 11<br>15 | 5295<br>5910 | 90.8<br>44.6 |
| [4G] | BLAKE-32<br>BLAKE-64 | 41.31<br>82.73 | 21<br>29 | 4153<br>4810 | 100.5<br>58.1 |
| [1G] | BLAKE-32<br>BLAKE-64 | 10.54<br>20.61 | 81<br>113 | 253<br>181 | 24.0<br>8.8 |
| [1/2 G] | BLAKE-32<br>BLAKE-64 | 9.89<br>19.46 | 161<br>225 | 127<br>91 | 12.9<br>4.7 |
| | JH-256 | 58.83 | 39 | 4992 | 84.85 |
| | BLAKE-32 | 45.64 | 22 | 3971 | 87.007 |

Table 3 contains the area and throughput of the BLAKE and JH with different architecture. The area is given in terms of gate equivalents (GEs)[18]. The reported clock frequency is the maximum value under typical conditions[19]. The throughput column indicates the peak throughput at the stated clock frequency. The [8G] and [4G]-BLAKE architectures maximize the throughput, so they were synthesized with speed optimization options at the maximal clock frequency. The target applications of [1G] and [1/2G]-BLAKE are resource-restricted environments, where a compact chip size is the main constraint.

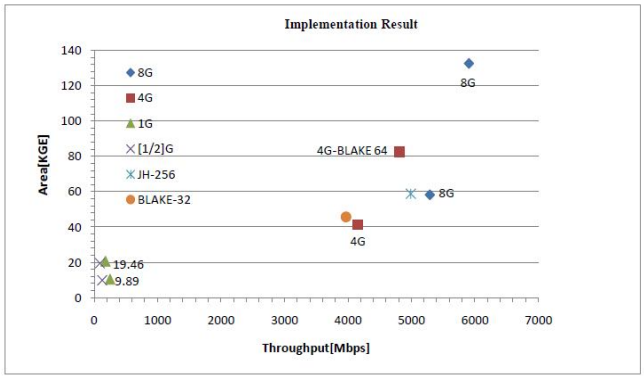



Figure 6: Both candidates comparisons graph

## VI CONCLUSION

We have presented FPGA implementation of the compression function for two candidates of the SHA-3 competition. Implementations have been carried out using the Stratix III FPGA family from Altera. Results have also been compared with the implementation results of some other candidates using the same platform.

## REFERANCES

[1] A. H. Namin and M. A. Hasan, Waterloo, Ontario N2L 3G1 Canada. Compression Function for Selected SHA-3 Candidates.

[2] Hongjun Wu, "The Hash Function JH", The First SHA-3 Candidate Conference, 2009, available on line at

http://ehash.iaik.tugraz.at/wiki/JH

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

[4] Wu, H., SHA-3 proposal JH, Website: http://icsd.i2r.astar.edu.sg/staff/hongjun/jh/.

[5] National Institute of Standard and Technology (NIST): Cryptographic Hash Algorithm Competition Website: http://csrc.nist.gov/groups/ST/hash/sha-3/.

[6] J.P. Aumasson, L. Henzen, W. Meier, R.C.W. Phan, "SHA-3 proposal BLAKE". Submission to the SHA-3 Competition, 2008.

[7] E. Biham, O. Dunkelman, "A framework for iterative hash functions- HAIFA".Cryptology ePrint Archive, Report 2007/278.

[8] Mao Ming,He Qiang ,Shaokun Zeng Xidian University Xi'an , Shanxi, China BLAKE-32 based on differential properties, 2010 International Conference on Computational and Information Sciences.

[9] George Provelengios,Nikolaos S. Voros,Paris Kitsos Greece, 2011 14th Euromicro Conference on Digital System Design

[10] SHA-1 Standard, National Institute of Standards and Technology (NIST), Secure Hash Standard, FIPS PUB 180-1,

1995, available on line at www.itl.nist.gov/fipspubs/ fip180-1.htm
[11] Secure Hash Standard (SHS), National Institute of Standards and Technology (NIST), FIPS PUB 180-3, 2008, available on line at http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf
[12] IEEE Annual Symposium Nicolas Sklavos 2010,ZIP 27100,GREECE,Paris Kitsos, Computer Science Hellenic Open University, GREEC
[13] J.-L. Beuchat, E. Okamoto, and T. Yamazaki, "Compact implementations of BLAKE-32 and BLAKE-64 on FPGA," in FPT, 2010, pp. 170–177.
[14] H. Wu. SHA-3 proposal JH, version January 15, 2009. JH
[15]Bernhard Jungk and J¨urgen Apfelbeck,Hochschule RheinMain, Wiesbaden, Germany 2011 International Conference on Reconfigurable Computing
[16] D. J. Bernstein. ChaCha, a variant of Salsa20. Available online at http://cr.yp. to/chacha/chacha-20080128.pdf, January 2008.
[17] E. Biham and O. Dunkelman. A Framework for Iterative Hash Functions - HAIFA. In Second NIST Cryptographic Hash Workshop, Santa Barbara, California, USA, August 24-25, 2006, August 2006.
[18]Cadence Design Systems. The Cadence Design Systems Website. http://www.cadence.com/.
[19].C. D. Canni_ere, H. Sato, and D. Watanabe. Hash Function Lufia, Specification Ver. 2.0. Available online at http://www.sdl.hitachi.co.jp/crypto/luffa/ Luffa_v2_Specification_20090915.pdf, September 2009.