Comparative Analysis of high speed and low area architectures of Blake SHA-3 candidate on FPGA

Muhammad Arsalan
Department of Electrical Engineering, PNEC
National University of Sciences and Technology
(NUST), H-12 Islamabad, Pakistan
m_arsal4@yahoo.com

Dr. Arshad Aziz
Department of Electrical Engineering, PNEC
National University of Sciences and Technology
(NUST), H-12 Islamabad, Pakistan
dr_arshadaziz@yahoo.com

Abstract— **On Nov. 2, 2007, NIST announced a public competition to develop a new cryptographic hash algorithm SHA-3. After long run selection process, five finalists were selected for Round 3. Winner of this competition will be announced later in 2012. Blake is one of the candidates of round three of this competition. Along with the strength of security, efficient hardware implementation is also major evaluation criteria for final selection. Blake algorithm compression function is based on G-Function which executes 8 times in one round. In this paper, different architecture schemes named as 8G, 4G and 1G has been implemented on FPGA; based on serialization of Round Function processes. Optimization is performed by selecting appropriate numbers of LUTs and Slice Registers according to the Virtex 5 Device Architecture Resources. Implementation results of each design are compared with each other and with other design contributions. Full autonomous design for each scheme is implemented on Virtex 5 xc5vlx50t-3 FPGA. Common I/O and control interface is provided to find out the fair comparison results. For tradeoff analysis three design optimization techniques based on 'area', 'speed' and 'balance' designs are used. We found 8G architecture provides the best through-put, 1G provides least area implementation and 4G provides the most efficient results in terms of throughput per area (TPA). 4G design gives Tpa of 2.1. Our design methodology and optimization strategy gives improved results from previous contributions.**

Keywords- SHA-3; Hash Functions; Blake; Encryption

I. INTRODUCTION

A hash function is a transformation that takes a variable-size input m and returns a fixed-size string, which is called the hash value. A cryptographic hash function aims to guarantee a number of security properties. Most importantly that it is hard to find collisions and the output appears random. The primary application of hash functions in cryptography is message integrity. The hash value provides a digital fingerprint of a message's contents, which ensures that the message has not been altered by an intruder, virus, or by other means. Hash algorithms are effective because of the extremely low probability that two different plaintext messages will yield the same hash value [5].

Secure Hash Algorithm (SHA-2) family of hash functions developed by the National Institute of Standards and Technology (NIST) is currently used as standard for Hash Functions. Serious attacks have been reported in recent years against cryptographic hash algorithm, including SHA-1, and SHA-2 family, NIST has decided to standardize an additional hash algorithm to augment the ones currently specified in FIPS 180-2 [15]. NIST issued a Call for a New Cryptographic Hash Algorithm (SHA-3) Family in a Federal Register Notice on Nov. 2, 2007 [6].

Five Secure Hash Algorithms were selected after Round-3 of SHA-3 competition. The result of the contest is expected to be announced at the end of this year. A basic evaluation criterion for selection SHA-3 standard is based on the strength of security and efficiency in Hardware implementation for wide verity of platforms [6].

Blake is one of the candidates of round three of SHA-3 competition. Blake is based on Haifa iteration mode. Internal structure of Blake is the local wide-pipe, which is already used with the LAKE hash function. Its compression algorithm is a modified version of Bernstein's stream cipher 'ChaCha'. Blake algorithm consists of family of four hash functions BLAKE-224, BLAKE-256, BLAKE-384 and BLAKE-512 [6]. Performance evaluation of Blake algorithm on FPGA is depending on the type of architecture implemented. The core of the design is based on compression function, which includes G-Function calculation. Due to the symmetric nature of compression function of Blake Algorithm, different design architectures can be implied based on serialization of common procedures. These design variations resulted in low to high area and speed results.

In this work, we have implemented three design architectures of Blake algorithms based on number of G-Functions execution at a time. The initialization and finalization step are same for all designs and variation is based only on round function execution. Optimization is performed by selecting appropriate numbers of LUTs and Slice Registers and their placement according to the Virtex 5 Device Architecture Resources. In the first design we have used 8 G-Functions in parallel, second design used hardware of 4 G-Functions and the third design is based on 2 half-G Functions in parallel with pipeline registers between them. Pros and cons of these architectures are discussed in this paper. In fully autonomous design implementation, common I/O interface is provided and only Slice resources of FPGA are used for design implementation for fair comparison. For tradeoff analysis, each design is implemented using three different optimization strategies, these are "Area", "Speed" and "Balance" design approaches. Mechanism for selection of message and constant values for each round is also discussed for all three architectures. The evaluation concluded that 4G design gives best results in terms of Tpa.

Rest of the paper is organized as follows. In section II different design approaches used by other researchers related to the both compact and high throughput designs implementation of Blake algorithm are discussed. Section III and IV explains the Blake algorithm and its architecture in detail. Section V describes our three proposed design methodology. In Section VI, design implementation results are presented and a comparison of three design approaches with each other and with other design implementations is discussed and finally in Section VII defines the conclusion of this research work.

II. Related Work

Various design techniques are used for both high-speed and compact design implementations for Blake algorithm. Kris et al. [2] implemented different architectures of Blake algorithm based on folding and pipelining technique. Detail analysis of Vertical and Horizontal Folding with different factor has been performed in his work. Horizontal folding with factor of 2 is similar to our 4G design, horizontal folding with the factor of 4 is similar to 1G and x1 basic iterative design is similar to our 8G design.

Baldwin et al. [3] design is based on older version of Blake algorithm submitted in second round of SHA-3 competition. In this version only 10 rounds are required for Hash value calculation. In his implementation, compression function is divided into two identical sections where round completion takes 4 cycles and counter, salt and message values are stored in BRAM of FPGA.

Kerchof et al. [5] used two implementation strategies; one with timing performance and other with area reduction. Fully autonomous architecture is implemented in his design where pipelining mechanism is used and rescheduling is performed to avoid pipeline stall. Baldwin et al. [3] and Kerchof et al. [5] used the similar approach as we used in our 1G design.

Benhard [7] used 2 half G-Functions in parallel with pipelining in his design. Rescheduling is applied to avoid pipeline stall. This design is comparable with our 1G design. Nikolas et al. [13] used a compression function where all message, state, constant; salt and count values are stored in Block RAM of FPGA. Kaps et al. [8] proposed two design architectures; one with block RAM and other one used logic resources. Instead of using full G function, one Half G-Function with Quasi pipelined stages are used in their design for compression function.

Kashif et al. [8] implemented both 256- and 512- variants of Blake algorithm on FPGA hardware. In his implementation, Blake algorithm is designed on Verilog HDL using LUT primitives of FPGA. In their design, 4 G-Functions were executed in parallel and each round is calculated in 2 clock cycles which is similar to our 4G design. Virtex 5, Virtex 6 and Virtex 7 FPGAs were used as hardware platform for implementation.

Beauchat et al. [11] used the approach of co-processor for the calculation of compression function of Blake algorithm. Pipelined architecture was used with interleaving of four compression functions. The design is based on Block RAM of FPGA. Both design implementations are based on Round-2 version of Blake algorithm. Miroslav et al. [12] design was also based on Round-2 of SHA-3 competition. In his fully autonomous implementation, analysis of IO interface on overall design performance is discussed.

Vaibhav et al. [14] used 4 G-Functions in their design. Blake-32 design is implemented on Virtex 5 FPGA which is based on round-2 submission of Blake algorithm. Comprehensive comparison of different design techniques used by various contributors based on 8G, 4G and 1G design with their results is given in Table 3.

III. Blake-256 Algorithm

BLAKE hash function consists of two basic variants, BLAKE-256 and BLAKE-512. BLAKE-256 operates on 32-bit words while BLAKE-512 operates on 64-bit words. The inner state of the Blake-256 compression function represents 4×4 matrix of 32-bits registers. The compression function involves 8 instances of G-function. G function consists of addition, XOR and rotation operations. Each G function operates on four elements of state matrix. BLAKE compression function constitutes 14 rounds for BALKE-256. Fig. 1 shows fully autonomous design architecture of Blake algorithm. Overall algorithm is divided into three stages Initialization, Round Function and Finalization.

After initialization the compression function iterates a series of 14 rounds. A round is a transformation of the state v that includes G-function execution on v state values firstly column wise and secondly in diagonal in each round. The core implementation of Blake algorithm is based on G-Function. It mainly includes three operations: the modular adder of 2^n, the bit-by-bit XOR on n-bit words and right rotation operation of k-bits.

$$\begin{aligned} &G0(v0, v4, v8, v12) \quad && G1(v1, v5, v9, v13) \\ &G2(v2, v6, v10, v14) && G3(v3, v7, v11, v15) \\ &G4(v0, v5, v10, v15) && G5(v1, v6, v11, v12) \\ &G6(v2, v7, v8, v13) && G7(v3, v4, v9, v14) \quad (1) \end{aligned}$$

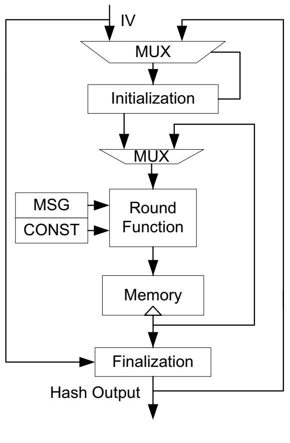

Figure 1. Blake Fully Autonomous Design Architecture

IV. Design Evaluation

In our design we have implemented Full Autonomous architecture of Blake algorithm which includes initial function, round function and finalization. In other to evaluate the hardware efficiency of different architectures, we have fixed certain features of design implementation. Since, different designers used different hardware platforms, synthesis and implementation tools and different I/O interfaces. Fair comparison is difficult to perform. We have used the features which are mostly used in research contributions. Our design used only slice resources of FPGA thus, performance evaluation metrics constitutes number of slices used and the maximum throughput achieved based on maximum design frequency calculated after place and route results. Description of common design features, used for the implementation of Blake-256 on FPGA designs is given below;

A. *IO Interfaces*

We used 64-bits input interface while 256-bits hash value is directly connected to the output port of FPGA. The wrapper consists of input FIFO as shown in Fig. 2, while padding function is not a part of our design.

B. *Message/Constant Select*

The permutation table is implemented on distributed memory of FPGA. Counter is used for memory addressing and multiplexers are used for message and constant value selection. The sigma value calculated for each round is buffered to reduce the critical path.

C. *FSM Control*

Finite state machine is used to control the overall operation of Blake algorithm. We have used Melay state machine with one-hot encoding for state register. Minimum state approach is used for efficient implementation.

D. *Implementation method & Device Platform*

Three architectures are designed using Verilog code. Xilinx ISE 13.1 is used for design synthesis and implementation. Xilinx Virtex 5 xcvlx50t-3 FPGA is used design implementation.

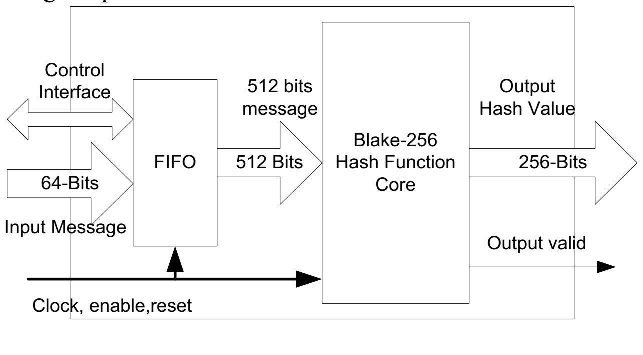

Figure 2. Blake-256 wrapper architecture

V. Design Methodology

Three different design approaches are used. These design approaches is based on number of G-Functions execution in Blake Algorithm Round Function.

A. *8G Design*

The architecture of 8G design is given in Fig. 3. It is consist of 8 G-Functions. 4x4 initial vector given to the round function through input Multiplexer in first cycle. The output generated in first cycle will be stored in a 4x4 32-bits register matrix. The register matrix values will be given back to input through input multiplexer. Since, all G-Functions are connected in parallel; each round will be executed in one cycle. Sigma values calculated from message and constant values for each round through mechanism given in Fig. 4.

The major advantage of 8G architecture is only one cycle is required for one round and whole algorithm is computed in 14 clock cycles. No additional multiplexers are required for order variation between column and diagonal step and signal are directly connected as shown in Fig. 3 and also no additional registers are required between column and diagonal step. Major disadvantage of this design is largest path delay and hence, it is difficult for synthesis and implementation tool to optimize such a large data path.

Since, 8 G-Functions are executed at a time, σa0, σb0, .., σa7, σb7 values required as given in Fig. 3, will be generated using counter, distributed memory, multiplexer and xor gates as shown in Fig. 4. The permutation table is stored in distributed memory of FPGA.

The memory will act like ROM and two ROM cores are generated of sizes 14x32bits each. 3216x1 Multiplexers are used for message and constant value selection. Xoring operation is performed according to the Equation 2. Registers are used to separate the critical path of round function with this DRAM chain.

$$\sigma_{air} = m_{2ir} + c_{(2i+1)r}$$
$$\sigma_{bir} = m_{(2i+1)r} + c_{2ir} \quad r = 1,2,\ldots,14 \quad i = 0,1,\ldots,7 \qquad (2)$$

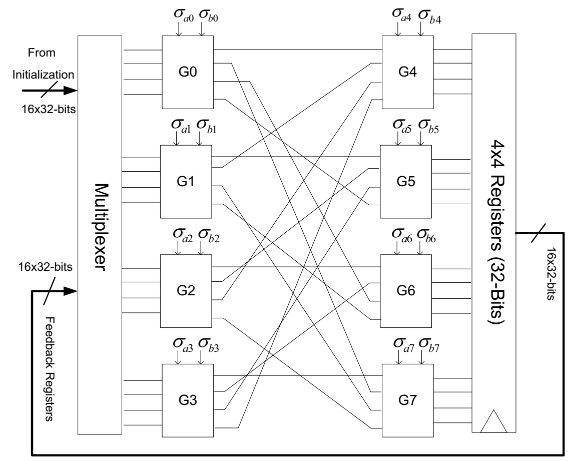
σ_{a0} σ_{b0}
σ_{a4} σ_{b4}
From Initialization
16x32-bits
G0
G4
σ_{a1} σ_{b1}
σ_{a5} σ_{b5}
G1
G5
Multiplexer
4x4 Registers (32-Bits)
σ_{a2} σ_{b2}
σ_{a6} σ_{b6}
16x32-bits
G2
G6
16x32-bits
Feedback Registers
σ_{a3} σ_{b3}
σ_{a7} σ_{b7}
G3
G7

Figure 3. 8G Design Architecture

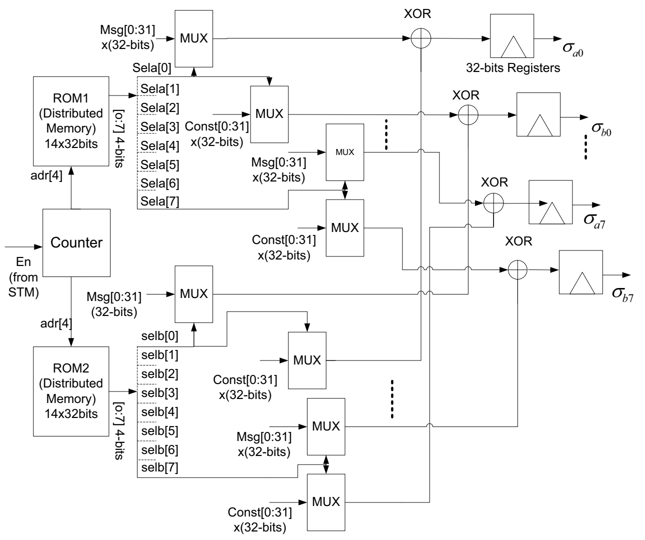
Figure 4. 8G Design message and constant selection mechanism

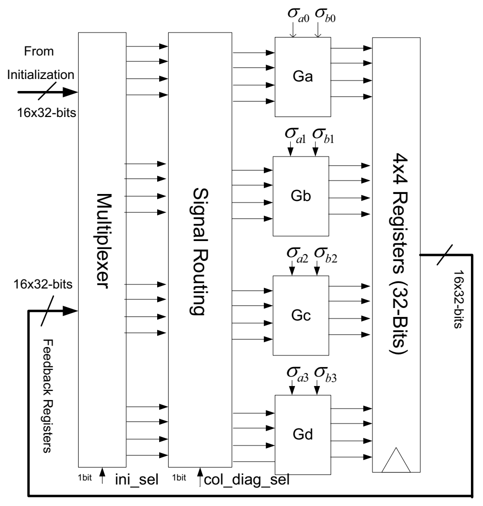
Figure 5. 4G Design Architecture

B. 4G Design

The architecture of 4G design is given in Fig. 5. It is consist of 4 G-Functions. 4x4 initial vector given to the round function through input Multiplexer in first cycle. The column step output generated in first cycle will be stored in a 4x4 32-bits register matrix. The register matrix values will be given back to input through input multiplexer and signal router will re-order the state values according to the diagonal step sequence. Since, one cycle is required for column step execution and one cycle is required for diagonal step execution; each round will be calculated in two clock cycles. Sigma values calculated from message and constant values for each round through mechanism given in Fig. 6. For diagonal step calculation the signal router will route the signal directly in straight order, while, for diagonal step order will be change depending upon the 'col_diag_sel' signal value i.e. '0' for column step and '1' for diagonal step.

The major advantage of this architecture is reduction in critical path to half as compared to 8G architecture. Two cycles are required for one round thus whole algorithm is computed in 28 clock cycles. Since, 8 sigma values are required for one cycle thus reduces the number of multiplexers required for message and constant values selection as given in Fig. 6. Additional resources will be required for signal routing and slightly complex state machine will be required for its control. Overall design methodology best suited the Virtex 5 architecture and hence, best results obtained in terms of Tpa.

In 4G design, only column or diagonal step will executed at a time, thus, σ_{a0}, σ_{b0}, .., σ_{a3}, σ_{b3} values required as given in Fig. 5, will be generated using the same methodology described for 8G architecture but only one ROM instance of size 28x32bits is used and 16 multiplexers are used for message and constant values selection. Xoring operation is performed according to the Equation 3.

$$\sigma_{air} = m_{2ir} + c_{(2i+1)r}$$
$$\sigma_{bir} = m_{(2i+1)r} + c_{2ir} \qquad r = 1,2,\ldots,14 \quad i = 0,1,\ldots,3 \qquad (3)$$

C. 1G Design

The architecture of 1G design is shown in Fig. 6. It includes two half-G Functions, four input multiplexers, four intermediate registers and 4x4 feedback register array. Input multiplexers will be responsible for selection of inputs for Half-G Function A.

At first round, column step, initial state vector will be given to its input through input multiplxer. In diagonal step of first round input to the Half-G Function A is given through feedback register array. Intermediate registers will be act as pipeline registers, thus, two clock cycles will be require for one G-function calculation.

In first cycle first part of G-Function i.e. G0a is calculated and filled up the pipelined register. In the next clock cycle, remaining part of G0 i.e. G0b will be calculated at the same time the G1a will be calculated too. In this way after five clock cycles, column step of first round has completed and its output will be stored in 4x4 feedback register.

For diagonal step of round 1, inputs to Half-G Function A will be selected through input multiplexer from 4x4 feedback register matrix in a sequence required by diagonal step. Each output of Half-G Function B is connected with 4 Registers simultaneously. The 'en' signal of the register will determine that selection of the output register to store intermediate state value. The critical path of the design consist of input multiplexer and five operations between points 'a' to 'b' of Half-G Function represented in Fig. 6. The Xoring operation of message and count values will not be included in critical path as the two registers have been used for σ_a and σb as mentioned in Fig. 8. Therefore, total 10 clock cycles are required for each round calculation. Thus, 10*14 + 4 = 140 clock cycles are required for complete hash value calculation.

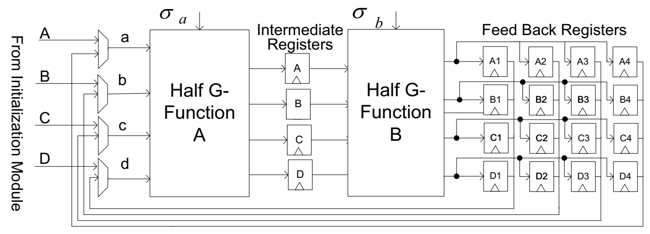

Figure 6. 1G Architecture Design

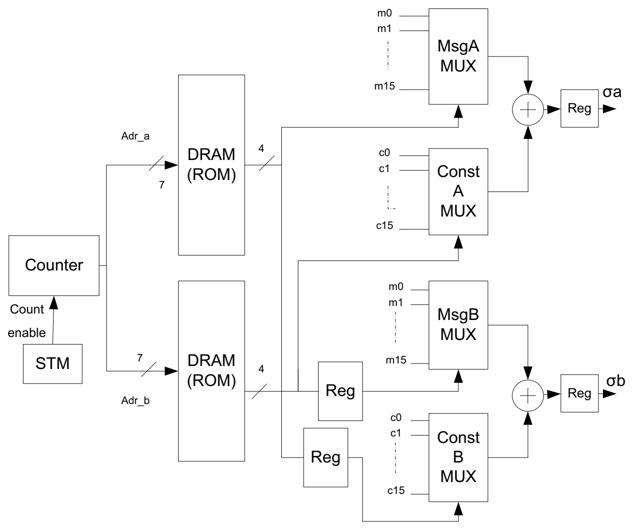

Figure 7. 1G Design message and constant selection mechanism

The mechanism for message and counter values selection is shown in Fig. 8 and similar to the 8G and 4G designs. Two ROM primitives with the size of (8x14) x 4-bits i.e. 56 bytes have been used and 7-bit Counter is used for address generation of two ROMs and it will be controlled by FSM controller. The selection for message and count values for each round can be represented as;

$$For\ HalfG\ Fn\ A: \quad constant{:} c\sigma r2i{+}1;\ message{:}\ m\sigma r2i$$
$$For\ HalfG\ Fn\ B: \quad constant{:} c\sigma r2i;\ message{:}\ m\sigma r2i{+}1$$

It can be seen from above equations that same permutation index value for Half-G Function A is given to Half-G Function B in next clock cycle. Hence, we have used registered/delayed ROM output to the multiplexer for message b and constant b as shown in Fig. 7.

TABLE 1. DESIGN OPTIMIZATION STRATEGIES

Speed	Timing Performance without IOB packing
Area	Area Reduction with Physical Synthesis
Balance	Xilinx Default

VI. DESIGN IMPLEMENTATION RESULTS

Synthesis and implementation of design is performed on Xilinx ISE 13.1. Results are shown for three different design strategies selected in ISE software as given in Table 1. The post-route implementation results are given in Table 2. Implementation results shows that maximum throughput of 2.62 Gbps is obtained from 8G design using "Speed Optimization" strategy. 4G design implementation with "Speed Optimization strategy" gives maximum TPA of 2.1. 1G design is best suited for low-area design implementation that gives slice count of 412 when using "Area Optimization" strategy as highlighted in Table 2.

TABLE 1. PLACE AND ROUTE IMPLEMENTATION RESULTS

Strategy	Slices	Freq (Mhz)	TP (Gbps)	TPA
		8G		
Area	1450	61.767	2.259	1.558
Speed	2275	71.633	**2.620**	1.152
Balance	2653	63.816	2.334	0.880
		4G		
Area	929	98.232	1.796	1.934
Speed	900	105.263	1.925	**2.139**
Balance	1429	112.740	2.062	1.443
		1G		
Area	**412**	157.480	0.576	1.398
Speed	416	193.424	0.707	1.700
Balance	569	173.913	0.636	1.118

The comparisons for Virtex 5 implementation with other contributions are given in Table 3. Table shows that our design results have much better improvements as compare to other contributions. For 8G-Design, our design shows the TPA of 1.56 while Aumasson et al. [11] gives the TPA of 1.83. But, later one is based on Blake-32 which is older version and requires only 10 clock cycles for round completion. If TPA is calculated after considering 14 clock cycles for [11], it will give TPA of 1.4 which is less than our design results. For 4G Design, our design results shows best performance for lowest slice count and highest TPA. 1G design results gives better performance as compare to all other design except Benhard. et al. [19], who used multiple pipeline approach with G-Function reorganization.

VII. CONCLUSION

Different design architectures of Blake-256 are implemented on FPGA. Our results show much better improvements as compare to previous contributions. Design uses large hardware resources gives maximum throughput i.e. 8G design requires only 14 clock cycles for Hash value calculation resulted throughput of 2.6 Gbps. 1G design gives most efficient results in terms of number of slices utilized.

The optimized delay path is utilized in 4G design with respect to Virtex 5 architecture. That's gives maximum TPA of 2.1. Appropriate selection of number of slice LUTs and Slice registers and their placement according to the Virtex 5 Device Architecture Resources gives the optimized results. Overall research suggests that selection of architecture is dependent upon type of application either high speed requirements or low area constraints, suitable optimization could be performed in a particular domain to achieve best design results.

TABLE 2. PLACE AND ROUTE RESULTS COMPARISON

Reference	Version	Slices	F Mhz	TP (Mbps)	TPA
8G					
Kris et al. [2]	Blake-256	2306		2561	1.11
Aumasson et al. [1]	**Blake-32**	1694	67	3103	1.83
This Work	**Blake-256**	**1450**	**61.76**	**2258**	**1.56**
4G					
Kris et al. [2]	Blake-256	1691		2253	1.33
Kashif et al. [9]	Blake-256	1382		2290	1.66
Vaibhav et al. [14]	**Blake-32**	1301	50	1280	0.98
Aumasson et al. [1]	**Blake-32**	1217	100	2438	2.00
This Work	**Blake-256**	**900**	**105.3**	**1925**	**2.14**
1G					
Kris et al. [2]	Blake-256	1547		1770	1.14
Baldwin et al. [3]	**Blake-32**	1118	118.1	1169	1.05
Benhard et al. [7]	Blake-256	374	163	725	1.94
Kirchof et al. [5] -area	**Blake-32**	192	240	183	0.95
Kirchof et al. [5] -speed	**Blake-32**	215	304	232	1.08
Aumasson et al. [1]	**Blake-32**	390	91	575	1.47
This Work	**Blake-256**	**416**	**193.4**	**707**	**1.70**

REFERENCES

[1] J.-P. Aumasson, L. Henzen, W. Meier, and R. Phan, "SHA-3 proposal BLAKE (version 1.3)," 2009, available at http://www.131002.net/blake

[2] Ekawat Homsirikamol, Marcin Rogawski, and Kris Gaj "Throughput vs. Area Trade-off in High-Speed Architectures of Five Round 3 SHA-3 Candidates Implemented Using Xilinx and Altera FPGAs" , CHES 2011, pp. 491-06

[3] Brian Baldwin, Andrew Byrne, Liang Lu, Mark Hamilto, Neil Hanle, Maire O'Neill and William P. Marnan "FPGA Implementations of the Round Two SHA-3 Candidates", FPL 2010, pp. 400-407

[4] Kris Gaj, Ekawat Homsirikamol, Marcin Rogawski, Rabia Shahid, and Malik Umar Sharif "Comprehensive Evaluation of High-Speed and Medium-Speed Implementations of Five SHA-3 Finalists Using Xilinx and Altera FPGAs" , Third SHA-3 Candidate Conference, March, 2012

[5] Stephanie Kerckhof, Francois Durvaux, Nicolas Veyrat, Francesco Regazzoni, Fstandae "Compact FPGA Implementations of the Five SHA-3 Finalists", CARDIS 2011, pp. 217-233

[6] National Institute of Standards and Technology (NIST), "Cryptographic Hash Algorithm Competition, http://csrc.nist.gov/groups/ST/hash/sha-3/index.html

[7] Benhard Jungk "Evaluation Of Compact FPGA Implementations For All SHA-3 Finalists", Third SHA-3 Candidate Conference, March, 2012

[8] Jens-Peter Kaps, Panasayya Yalla, Kishore Kumar, Bilal Habib, Susheel Vadlamudi, Smriti Gurung "Lightweight Implementations of SHA-3 Finalists on FPGAs" , Third SHA-3 Candidate Conference, March, 2012

[9] Kashif Latif, M Muzaffar Rao, Arshad Aziz and Athar Mahboob "Efficient Hardware Implementations and Hardware Performance Evaluation of SHA-3 Finalists" , Third SHA-3 Candidate Conference, March, 2012

[10] Knezevic, Kobayashi, K. , Ikegami, J. , Matsuo, S. , Satoh, A. , Kocabas, U. , Junfeng Fan ; Katashita, T. , Sugawara, T. , Sakiyama, K. ; Verbauwhede, I. ,; Ohta, K. , Homma, N. , Aoki, T. , "Fair and Consistent Hardware Evaluation of Fourteen Round Two SHA-3 Candidates", VLSI Systems, IEEE Transactions, May, 2012. pp. 827-840

[11] Jean-Luc Beuchat, Eiji Okamoto, and Teppei Yamazaki "Compact Implementations of BLAKE-32 and BLAKE-64 on FPGA" , FPT 2010, pp. 170-177

[12] S. Matsuo, M. Knežević, P.Schaumont, I. Verbauwhede, A. Satoh, K. Sakiyama, and K. Ohta, "How Can We Conduct Fair and Consistent Hardware Evaluation for SHA-3 Candidate", 2nd SHA-3 Conference, 2010, pp. 15

[13] Nicolas Sklavos, Paris Kitsos "BLAKE HASH Function Family on FPGA: From the Fastest to the Smallest" IEEE ISVLSI'10, pp. 139-142

[14] Vaibhav Doshi, Richa Arya, Rajesh Kumar Yadav, "FPGA Based Area and Throughput Implementation of JH And BLAKE Hash Function", International Journal of Computer Trends and Technology, vol.3 Issue.2, 2012.