Detail Power Analysis of the SHA-3 Hashing Algorithm Candidates on Xilinx Spartan-3E

Amar Jaffar and Christopher J. Martinez

Abstract—**The United States National Institute of Standards and Technology (NIST) created an open competition to find a new standard for cryptographic hashing in 2007. The competition was composed of a number of rounds that ended with 5 finalists (BLAKE, Grostl, JH, Keccak, Skein) in December of 2010. This research paper will examine the power consumption of the 5 finalists on the Xilinx Spartan3E FPGA. The results in this paper will allow developers examining including a hash function in a hardware design to have a full understanding of the power requirements and aid the developer in making an inform decision. The results show that the Keccak algorithm has the best tradeoff of power to speed and the BLAKE algorithm is the best choice for low power design.**

Index Terms—**FPGA, hashing, power analysis, SHA-3.**

I. Introduction

The United States National Institute of Standards and Technology (NIST) has standardized security algorithms for the United States. Examples of algorithms the NIST has standardized include DES, Skipjack, AES, SHA-1, SHA-2, DSA, and RSA. These algorithms are not only used by contractor conducting business with the United States government but are the de facto standard in the technology field.

The state of the art cryptographic hashing function SHA-2 created in 2001 was reaching the end of its approved lifecycle and the NIST set forth to create a new standard for adoption in 2012. The new standard would be known as SHA-3. In order for SHA-3 to be the most cryptographic secure hashing algorithm available the NIST created a NIST hash function competition [1]. The competition started in 2007 and was narrowed down to five finalist (BLAKE, Grostl, JH, Keccak, and Skein) in December 2010. The competition ended in October of 2012 with the selection of Keccak as the winner of the competition. While Keccak was selected as the SHA-3 algorithm all five finalist will continue to be used in many application.

The NIST competition chose the winner based on the following important factors: performance, security, analysis, and diversity. This paper focuses on examining the performance of all five finalists. Performance is no longer a measurement of only execution time. Power has become as important or more important than speed in many computer domains, especially mobile and embedded systems. This paper examines the performance in terms of power consumption of all the finalists on the Xilinx FPGA platform. The Xilinx FPGA chip Spartan3E was used in this research project.

This paper is meant to be a guide for researchers and developers in understanding the power consumption of the five hashing algorithms with a comparison to power also provided. Developers in the field will be able to use this information to aid in the planning stage of devices that require hashing algorithm. Developers will be able to use our information to determine which algorithms would work with their power budget for a design. The work will allow researchers to have a baseline for power consumption and examine new ways to reduce power in future designs.

II. SHA-3 Finalist

A. BLAKE

The BLAKE algorithm is created by the team of Jean-Philippe Aumasson, Luca Henzen, Willi Meire and Raphael Phan [2]. The BLAKE algorithm is an adaptation of the ChaCha stream cipher and can produce message digests of 224-bits, 256-bits, 384-bit, and 512-bits. The BLAKE algorithm is composed of a ChaCha function that performs transformations on 4 words. The transformation involved an XOR and bit rotation leading to a fast implementation. A total of 10 to 14 rounds of ChaCha functions are used depending on the size of the message digests required.

B. Grostel

The Grostl algorithm is created by the team from Technical University of Denmark and TU Graz [3]. The Grostel algorithm borrows elements from the AES cipher algorithm. The Grostel algorithm has high throughput since many optimization for AES have been done in software and hardware over the years. Grostl uses the AES S-box function and similar permutation functions.

C. JH

The JH algorithm was created by Hongjun Wu [4]. The JH algorithm is inspired by the AES and Serpent cipher algorithms and is made up of 42 rounds of execution. Each of the 42 rounds consists of four S-boxes and MDS transformations.

D. Keccak

The Keccak algorithm was the winner of the SHA-3 competition. The Keccak function is created using a number of sponge functions [5]. The Keccak sponge function is made up of seven permutation functions of different bit lengths. The seven permutation functions are then used in XOR and

Manuscript received December 15, 2012; revised February 22, 2013.

A. Jaffar and C. Martinez are with the University of New Haven, West Haven, CT 06516 USA (e-mail: ajaff1@unh.newhaven.edu; cmartinez@newhaven.edu).

DOI: 10.7763/IJCEE.2013.V5.742

rotation operations.

E. Skein

The Skein algorithm is based on the Threefish block cipher [6]. Skein uses addition, XOR and rotation to create a MIX function. A number of MIX functions are used and the outputs of the functions go through permutation functions. The Skein algorithm requires 72 or 80 round depending on the block size used to run the algorithm.

III. Power Analysis

The SHA-3 finalist algorithms have been examine and evaluated by many different researchers. The work has mainly focus on implementation in terms of area and speed. A number of works have used FPGAs as the implementation technology to analysis. The work in [7] used Virtex 5 FPGAs for the Grostel algorithm, [8] examined all five finalist for size on Spartan-3 and Virtex 5 FPGAs, and [9] was another study on all five finalist. The implementation of the SHA-3 finalists were written in VHDL and adapted from [10]. The power analysis was done on Xilinx ISE 13.1 using Spartan-3E. The Xilinx tool allows for synthesis to optimize for speed, area, and memory type. The FPGA has two synthesis optimization options normal and high for speed and area. Memory can be distributed memory or block memory.

A. BLAKE

The BLAKE algorithm was ran on two different configuration of block size BLAKE-32 and BLAKE-64, each is used to produce different message digest sizes. The results of the BLAKE algorithm on the Spartan-3E are shown in Table I.

The power analysis shows that the average power consumption for all the different configurations of optimization for BLAKE-32 was 258.25 mW and the average power consumption for BLAKE-64 was 313.50 mW. By using a larger message digest size the power consumption increase by 21.39%. The type of RAM used by the algorithm also has an important impact on the power consumption. For BLAKE-32 using block-RAM requires on average an additional 21.90% more power than using distributed-RAM; for BLAKE-64 the additional power for block-RAM is 41.48%. If power is the top design requirement our analysis shows that distributed-RAM should be used in the design. The reason for the additional power in block-RAM is that block-RAM is a dedicated memory resource on the FPGA; by selecting distributed-RAM the design is able to disable the dedicated memory from drawing power. The use of block-RAM does have the ability to improve the processing speed. In the BLAKE-32 algorithm, using block-RAM with the speed-optimized synthesis configuration an average of 9.76% speedup in the maximum clock frequency was achieved. The BLAKE-64 algorithm had no noticeable difference in clocking frequency between block-RAM and distributed-RAM. In the BLAKE-64 algorithm there was no improvement in the number of slices required for implementation when using the area-optimized synthesis configuration compared to the speed-optimized synthesis configuration. The reason is that the hash algorithm is mostly composed of basic logic functions that cannot be reduced.

Overall for BLAKE-32 the average power consumption for the speed-optimized synthesis option was 262.5 mW and for the area-optimized synthesis option was 254 mW. The difference is minimal at 3.35% between speed and area optimization. The difference in power for BLAKE-64 increases to 6.25% between the speed and area optimization options; the average power consumption for speed-optimization was 323 mW and 304 mW for area-optimization. By using the speed-optimized synthesis option an approximate speedup of 20% in clock frequency would be expected.

Examining the results of the research one can conclude that the best synthesis option to use is the speed-optimized synthesis configuration. The difference between power consumption is minimal (3%) between speed-optimized and area-optimized but the difference in clock speed is large (20%).

TABLE I: Blake Power Analysis

	Power(mW)	Area Slices	Freq (MHZ)
BLAKE-32			
Normal Speed/ D-Mem	234	3813	40.82
Normal Speed/B-Mem	294	3685	46.51
High Speed/D-Mem	237	3819	39.22
High Speed/B-Mem	285	3687	41.41
Normal Area/D-Mem	231	3812	34.54
Normal Area/B-Mem	281	3682	34.54
High Area/D-Mem	229	3812	35.24
High Area/B-Mem	275	3684	35.24
BLAKE-64			
Normal Speed/D-Ram	272	7469	36.86
Normal Speed/B-Ram	375	7224	36.75
High Speed/D-Ram	269	7470	37.40
High Speed/B-Ram	376	7225	36.93
Normal Area/D-Ram	251	7479	30.17
Normal Area/B-Ram	368	7210	30.08
High Area/D-Ram	247	7477	30.66
High Area/B-Ram	350	7210	30.57

B. Grostel

The Grostel algorithm has two different configurations correspond to the different message digest sizes that are available 256 and 512. The Grostel algorithm does not require any memory so the analysis does not compare the difference between block ram and distributed ram. The Grostel-512 configuration could not be synthesized on the Spartan-3E. The results of the Grostel-245 algorithm on the Spartan-3E are shown in Table II.

The power analysis on the Grostel algorithm shows the average power consumption for the speed-optimized configurations is 287 mW and the area-optimized configuration is 283.5 mW, a difference of 1.23%. While there is minimal difference in power consumption the maximum frequency available has a noticeable difference between the size-optimization synthesis and speed-optimization synthesis option. The speed-optimized synthesis configuration has a 22.79% improvement over the area-optimized synthesis configuration in clock frequency.

The analysis shows that the speed-optimized synthesis configuration is the best implementation to use. The speed option has the benefit of a higher clock rate with a negligible difference in power.

TABLE II: GROSTEL POWER ANALYSIS

	Power(mW)	Area Slices	Freq (MHZ)
Grostel-256			
Normal Speed	321	3082	87.26
High Speed	253	3081	86.58
Normal Area	321	3080	69.16
High Area	246	3080	72.41

C. JH

The JH algorithm only has one configuration to produce all the required message digest sizes. The JH algorithm similar to the Grostel algorithm does not require memory so the analysis does not include a breakdown of the power for block ram and distributed ram. The results of the JH algorithm on the Spartan-3E are shown in Table III.

The power analysis for the JH algorithm shows that the speed-optimized synthesis configuration has an average power consumption of 281.5 mW and the power for the area-optimized synthesis configuration is 270 mW. Using the speed-optimized configuration has a 4.26% additional requirement for power. The difference is clock frequency is a 15.15% speedup for speed-optimized over the area-optimized configuration.

The synthesis analysis shows that there is no clear implementation synthesis option that stands out. The speed-optimized configuration is 15% faster and uses 4% more energy. The implementation to be used in a design will depend on the importance of speed and power in the design requirements.

TABLE III: JH POWER ANALYSIS

	Power(mW)	Area Slices	Freq (MHZ)
JH			
Normal Speed	276	3217	84.67
High Speed	287	3215	80.06
Normal Area	271	3218	73.86
High Area	269	3218	69.20

D. Keccak

The Keccak algorithm requires two different configurations for the 256-bit and 512-bit message digest sizes. The Keccak algorithm does not require the use of any RAM in the implementation. The results of the Keccak algorithm on the Spartan-3E are shown in Table IV.

The power analysis for the Keccak algorithm is divided by the two different implementations Keccak-256 and Keccak-512. The Keccak-256 algorithm had an average power consumption of 262 mW for the speed-optimized synthesis configuration and 222.5 mW for the area-optimized synthesis configuration, using the area-optimized configuration has a 17.75% reduction in power. Keccak-256 has a large difference in the clock frequency between the speed-optimized and area-optimized synthesis, the speed-optimized configuration is 48.18% faster than the area-optimized configuration.

The Keccak-512 algorithm has a large difference between the speed-optimized and area-optimized synthesis. The average speed-optimized synthesis had an average power consumption of 303 mW and the area-optimized synthesis had an average power consumption of 227 mW. The area-optimized synthesis configuration runs 33.48% more power efficient than the speed-optimized synthesis configuration. When comparing the clock frequency the speed-optimized configuration has a 39.46% speedup over the area-optimized configuration.

The Keccak algorithm implementation to use is defined by the design requirements. The difference in power and speed is great between the speed-optimized and area-optimized synthesis. The Keccak-512 algorithm can be designed to over 30% more power efficient or 40% faster.

TABLE IV: KECCAK POWER ANALYSIS

	Power(mW)	Area Slices	Freq (MHZ)
Keccak-256			
Normal Speed	261	3036	80.91
High Speed	263	3036	81.10
Normal Area	223	3037	54.70
High Area	222	3037	54.64
Keccak-512			
Normal Speed	305	2785	73.05
High Speed	301	2785	73.05
Normal Area	228	2785	52.38
High Area	226	2785	52.38

E. Skein

The Skein algorithm only requires a single configuration to create any message digest size. The Skein algorithm does not require the use of any RAM in the implementation. The results of the Skein algorithm are shown in Table V.

The Skein algorithm has an average power consumption of 380 mW for the speed-optimized synthesis configuration and 373 mW for the area-optimized synthesis configuration. The difference between the speed-optimized and area-optimized was minimal at 2.01%. When examining the clock frequency there is a difference of 5.1% between speed-optimized and area-optimized. The speed-optimized configuration had a clock frequency of 46.15 MHZ and the area-optimized configuration had a frequency of 43.91 MHZ.

The analysis shows that there is no large difference between area-optimized and speed-optimized synthesis implementation. Both speed and power difference by less than 5%.

TABLE V: SKEIN POWER ANALYSIS

	Power(mW)	Area Slices	Freq (MHZ)
Skein			
Normal Speed	389	1688	46.15
High Speed	372	1688	46.15
Normal Area	373	1655	43.90
High Area	373	1655	43.92

F. Overall Analysis

All of the five SHA-3 finalist were compared in terms of both power (Table VI) and clock speed (Table VII); the tables rank the eighteen algorithm implementations by performance measurement. The analysis shows why the Keccak algorithm was chosen as the SHA-3 winner. The Keccak algorithm is at the top of both the power and speed ranking list. The Keccak has the top two algorithm implementations in terms of power and has two algorithm implementations in the top quarter of the clock frequency ranking list.

TABLE VI: SHA-3 FINALIST RANKED BY POWER CONSUMPTION

Algorithm	Power (mW)
Keccak-256 area optimize	225.5
Keccak-512 area optimize	227
BLAKE-64 area optimize DRAM	230
BLAKE-64 speed optimize DRAM	235.5
BLAKE-32 area optimize DRAM	249
Keccak-256 speed optimize	262
JH area optimize	270
BLAKE-32 speed optimize DRAM	270.5
BLAKE-64 area optimize BRAM	278
JH speed optimize	281.5
Grostel area optimize	283.5
Grostel speed optimize	287
BLAKE-64 area optimize BRAM	289.5
Keccak-512 speed optimize	303
BLAKE-32 area optimize BRAM	359
Skein area optimize	373
BLAKE-32 speed optimize BRAM	375.5
Skein speed optimize	380.5

TABLE VII: SHA-3 FINALIST RANKED BY CLOCK FREQUENCY

Algorithm	Freq (MHz)
Grostel speed optimize	86.92
JH speed optimize	82.37
Keccak-256 speed optimize	81.01
Keccak-512 speed optimize	73.05
JH area optimize	71.53
Grostel area optimize	70.79
Keccak256 area optimize	54.67
Keccak512 area optimize	52.38
Skein speed optimize	46.15
BLAKE-64 speed optimize BRAM	43.96
Skein area optimize	43.91
BLAKE-64 speed optimize DRAM	40.02
BLAKE-32 speed optimize DRAM	37.13
BLAKE-32 speed BRAM	36.84
BLAKE-64 area DRAM	34.89
BLAKE-64 area BRAM	34.89
BLAKE-32 area DRAM	30.42
BLAKE-32 area BRAM	30.33

The analysis shows that the BLAKE algorithm should not be chosen if computation speed is a top design requirement. The BLAKE algorithm is in the bottom half of the ranking for clock speed. In terms of power the BLAKE algorithm has three implementations in the top 30% of the ranking by power consumption.

The Grosetl algorithm has the highest computation speed and is in the bottom half in the power consumption ranking. There is no clear advantage to using the Grostel in terms of speed and power. One will chose to use Grostel for the hashing algorithm properties only. The Skein algorithm has the worst power consumption of the entire SHA-3 finalist and is in the lower half of the clock frequency rankings. The analysis shows that the Skein algorithm should be avoided.

IV. CONCLUSION

The worked conducted in this paper examined the FPGA implementation analysis of the SHA-3 finalist. The analysis breaks down the power consumption and clock frequency of each SHA-3 finalist algorithm using a mixture of different synthesis options. The results show that the Keccak algorithm is the strongest finalist when examining the tradeoff between power and speed. The BLAKE algorithm has an advantage of having low power consumption at the cost of speed and is useful for power concern designs. The Skein and Grostel algorithms have no advantage in terms of speed or power and should only be chosen based on the quality of the cryptographic strength.

REFERENCES

[1] NIST, “National institute of standards and technology [docket no: 070911510-7512-01] announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (SHA-3) family,” Federal Register, November 2007.

[2] J. P. Aumasson, L. Henzen, W. Meier, and R. C. W. Phan, ‘SHA-3 proposal BLAKE version 1.3,” NIST SHA-3 Competition.

[3] P. Gauravaram, L. R. Knudsen, K. Matusiewicz, F. Mendel, C. Rechberger, M. Schlffer, and S. S. Thomsen, “SHA-3 proposal grostel version 2.0.1,” NIST SHA-3 Competition.

[4] H. J. Wu, “The hash function jh,” NIST SHA-3 Competition.

[5] G. Bertoni, J. Daemon, M. Peeters, and G. V. Assche, “The Keeccak sha-3 submission,” NIST SHA-3 Competition.

[6] N. Ferguson, S. Lucks, B. Schneier, D. Whiting, M. Bellare, T. Kohno, J. Callas, and J. Walker, “The skein hash function family,” NIST SHA-3 Competition.

[7] B. Baldwin, A. Byrne, M. Hamilton, N. Hanely, R. McEvoy, W. Pan and W. Marnane, “FPGA Implementations of SHA-3 Candidates: CubeHash, GrostEl, LANE, Shabal and Spectral Hash,” IACR report 2009/342.

[8] S. Kerckhof, F. Burvaux, N. Veyrat-Charvillon, F. Regazzoni, G. M. de Dormale, and F. X. Standaert, “Compact FPGA Implementations of the Five SHA-3 Finalist,” in *Proceedings of CARDIS*, 2012.

[9] J. P. Kaps, P. Yalla, K. K. Surapathi, B. Habib, S. Vadlamudi, S. Gurung, and J. Pham, “Lightweight Implementations of SHA-3 Candidates on FPGA,” presented at 12th International Conference on Cryptology in India, 2011.

[10] B. Baldwin, N. Hanley, M. Hamilton, L. Lu, A. Byrne, M. O’Neil, and W. Marnane, “FPGA Implementations of the Round Two SHA-3 Candidates,” in *Proceedings of the 20th International Conference on Field Programmable Logic and Applications*, August 2010.

Amar Jaffar received his M.S. in Computer Engineering from the University of New Haven, CT, USA in 2012. He is a lecturer in Computer Engineering at Umm Al-Qura University, Makkah, KSA. He is continuing his education by pursuing a Ph.D. in Computer Engineering at Florida Institute of Technology.

Christopher J. Martinez received his Ph.D. in Electrical Engineering from The University of Texas at San Antonio, TX. USA. He is currently an Assistant Professor in Computer Engineering at the University of New Haven, CT, USA. He has publications in the area of computer architecture, networking and embedded systems. His current research interest is in computer architecture, WSN, and HCI.