Fair and Comprehensive Performance Evaluation of 14 Second Round SHA-3 ASIC Implementations

Xu Guo, Sinan Huang, Leyla Nazhandali and Patrick Schaumont

Bradley Department of Electrical and Computer Engineering,
Virginia Tech, Blacksburg VA 24061
{xuguo,shuang86,leyla,schaum}@vt.edu

Abstract. Hardware implementation quality will be considered as an important factor for evaluating the NIST SHA-3 competition candidates in the second round. The most traditional and popular hardware implementation method is designing ASICs with standard cells. However, to benchmark 14 second round SHA-3 ASIC designs based on a fair and comprehensive methodology can be very challenging because of the undefined application scenarios, various choices of technologies and multiple optimization goals. In this paper we describe a consistent and systematic approach to move a SHA-3 hardware benchmark process from FPGA prototyping to ASIC implementation, and we present our latest results for ASIC evaluation of the 14 second round SHA-3 candidates. The effort reported in this paper is complementary to the effort reported in the SHA-3 conference submission "How can we conduct fair and consistent hardware evaluation for SHA-3 candidates?" [4].

1 Introduction

The SHA-3 competition organized by NIST aims to select, in three phases, a successor for the mainstream SHA-2 hash algorithms in use today. By the completion of Phase 1 in July 2009, 14 out of the 51 hash candidate submissions were identified for further consideration as SHA-3 candidates. These 14 candidates will be further analyzed with respect to security, cost and performance, and algorithm and implementation characteristics [1].

For the second phase of the competition, NIST is looking for additional cryptanalytic results, as well as for performance evaluation data on hardware platforms. The SHA-3 submissions were made as a software reference implementation in combination with a set of test vectors [2]. This pragmatic approach leverages ubiquitous computer infrastructure as a standard evaluation platform, and it suits the purpose of cryptanalysis. However, the reference implementations in C are also far away from actual hardware design. As a result, significant additional design work is required before the SHA-3 candidates can be evaluated in terms of hardware cost.

In contrast to software implementations, which can be characterized based on performance (execution time) only, hardware implementations have at least

one additional dimension: resource cost, in addition to performance. Indeed, for hardware implementations, the architecture of the design represents an additional degree of design freedom. As a result, there is no single optimal hardware implementation. Every design has to be considered as a combination of performance under a given resource cost. This aspect complicates the comparison of designs. One may look for minimal resource cost under a given performance, or else for maximal performance under a given resource cost. Hence, a hardware benchmarking methodology needs to take this duality into account.

eBACS is a well known benchmarking environment, including a scripting environment and a performance database, for the evaluation of crypto-software [3]. This environment already supports 14 Phase-2 candidates. Compared to the proposed methodology for benchmarking crypto-software, benchmarking crypto-hardware is ad-hoc. There are several reasons why the same progress is not seen in the hardware design community. All of them boil down to a lack of standardized approaches towards the design process.

First, there are no standard methodologies to quantify the cost and performance of a hardware implementation. In the average crypto-hardware conference proceedings, one will find that no two authors measure resource cost or performance of hardware implementations using the same metrics. For example, the 11 tables that compare hardware implementations in the proceedings of CHES 2008 contain 18 different metrics for hardware cost and 10 different metrics for hardware performance [4]. While one author may use clock cycles, another one may use nanoseconds, and a third one blocks-per-second. It is up to the reader to provide the proper context.

A second reason is that hardware implementations show a larger heterogeneity compared to software processors. This includes the design target (ASIC or FPGA), the technology node, and the optimization scenario being used. Again, it is up to the reader to provide the proper context when making comparisons.

A third reason is the lack of standardized interface mechanisms for crypto-hardware modules. Because the architecture of a hardware design is a design decision, designers tend to count the interface as part of that freedom. This, however, significantly complicates benchmarking. Indeed, a standard Application Programming Interface (API) is a key enabler in existing software benchmarking environments such eSTREAM [5] and eBACS [3].

In this contribution we report on a methodology to address these issues for the SHA-3 ASIC benchmark process with two major steps. First, we propose the use of an FPGA platform which serves as the starting point for ASIC evaluation. Second, we compare the SHA-3 ASIC results, and we address the impact of different factors that are quite relevant for fair and comprehensive evaluation. These factors include technology differences, ASIC layout overhead over the post-synthesis results, various application-specific constraints, and different hash operation modes.

2 Related Work

This paper is complementary to the paper "How can we conduct fair and consistent hardware evaluation for SHA-3 candidates", a joint submission by National Institute of Information and Communications Technology (NICT), Katholieke Universiteit Leuven (KUL), Virginia Tech (VT), National Institute of Advanced Industrial Science and Technology (AIST), University of Electro-Communications (UEC) [4]. Hence, we will not repeat information of that submission in this paper, but instead will refer to that paper for the following results:

- A description of related work.
- A description of a standard hardware interface for SHA-3 hash modules.
- A description of hardware performance evaluation metrics.

3 ASIC Evaluation Methodology

In this section, we describe our efforts in ASIC performance evaluation. We describe the overall design flow that combines FPGA prototyping with ASIC design, and next elaborate the efforts to automate and standardize the ASIC implementation process.

3.1 Overview

This work starts with an international collaboration among several research groups in developing RTL designs of the 14 second round candidates. The benefits of this collaboration not only make us finish all the RTL coding with decent quality in a very short time but also let us hear suggestions from worldwide experts to improve the methodology. Currently, we use two sets of 14 SHA-3 designs in this flow. The first was designed through collaboration between VT, KUL and UEC. The second was contribute by George Mason University (GMU). In this paper, we discuss results from the first set.

Figure 1 illustrates the overall design flow in our ASIC implementation. A set of RTL SHA-3 candidates is implemented in Verilog or VHDL. These hardware descriptions are next mapped to FPGA technology or ASIC technology. We use the same RTL descriptions for both types of design flow. Our objective is to use the FPGA as a prototyping technology for the ASIC, rather than a direct technology target. Hence, dedicated FPGA optimizations, such as the use of specialized multipliers or memory cells, are not used.

The ASIC and FPGA design flows look very similar, and cover the same two technology mapping steps. The first step is synthesis and maps the RTL code (in Verilog or VHDL) to a netlist of technology primitives. The second step is place and route, and this step decides the spatial relationships of technology primitives in a layout. Both of these steps can be automated using scripts. The results of technology mapping are performance estimates such as circuit area and circuit delay. The performance delays obtained after place-and-route are

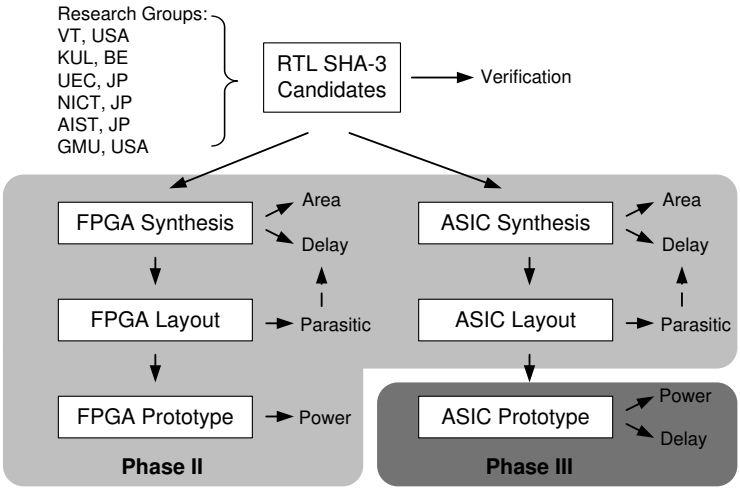

Fig. 1. An overview of the SHA-3 ASIC evaluation project.

more accurate than those obtained after synthesis. With respect to the circuit area, place-and-route will reveal the precise dimensions of the ASIC design. With respect to the circuit delay, place-and-route reveals implementation effects (annotated as parasitics in Fig. 1) which characterize delay effects caused by the interconnections.

The result of the ASIC and FPGA design flow is used in a prototype design based on the SASEBO board. In the case of ASIC design, we plan to make a tape-out after the final candidates for SHA-3 Phase-III are selected. During Phase-II, we perform prototyping on FPGA only. This prototyping is useful to evaluate power consumption, such as is discussed in the paper related to this work [4]. In the next subsection, we discuss the implementation details of the prototype design.

3.2 Platform for integrated FPGA prototyping and ASIC performance evaluation

The experimental environment for FPGA prototyping contains a PC, a SASEBO-GII board and an oscilloscope. A SASEBO-GII board contains two FPGAs: a control FPGA, which supports the interfacing activities with a PC, and a cryptographic FPGA, which contains the hashing candidate. During the ASIC prototyping phase, the cryptographic FPGA is replaced by an ASIC containing SHA-3 candidates. A board from the SASEBO-R series will be used for this purpose.

The SASEBO board was originally developed for side-channel analysis. Hence, a potential research area for the FPGA prototype is side-channel analysis of SHA-3 candidates. In our experiments, we used the SASEBO board for a more obvious application, namely the measurement of power dissipation of the SHA-3 candidates mapped to the cryptographic FPGA.

The interface of the SASEBO board on the PC side is a software driver that can read the test vectors and that can send messages to the SHA-3 FPGA through USB. The Control FPGA manages the data flow of the messages and generates control signals according to the timing requirements of a standard hash interface [6]. After SHA-3 FPGA finishes hash operations, the digest is returned to the PC through the Control FPGA. For the final ASIC prototype, the same data flow will be used.

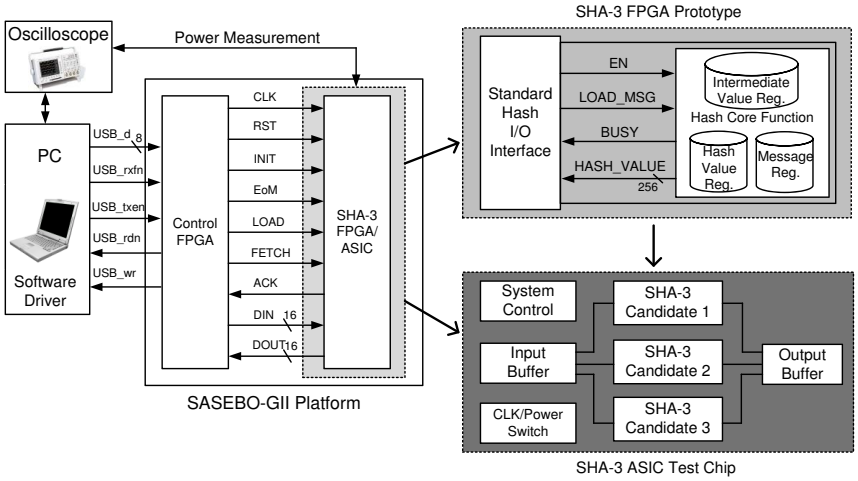

Fig. 2. Experimental environment for FPGA prototyping and final ASIC testing.

3.3 ASIC Performance Evaluation

In preparation of the ASIC prototype design, we performed a comprehensive performance analysis of the SHA-3 candidates according to the design flow of Fig. 1. The FPGA results of this design flow are discussed in a related submission [4]. In this paper, we describe the results for the ASIC design flow.

The performance evaluation of a design in ASIC technology can be done under multiple technologies. Rather than evaluating all 14 candidates under multiple technologies, we first evaluate a single candidate under different ASIC design parameters as follows.

- We evaluate the impact of different technologies. A smaller technology is smaller and faster, but may also have increased static power dissipation.
- We evaluate the impact of different constraints. During technology mapping, a given RTL design can be optimized for area, speed, or a combination of those.

– We compare Post-Synthesis results vs. Post-Layout results. ASIC layout provides additional implementation characteristics such as precise area and netlist parasitics.
– We evaluate the impact of message length. Because the regular processing, and the final processing of a hash candidate can differ, the message length may affect the average activity of a hash implementation. This will affect the power dissipation.

To evaluate these parameters, we used the Synopsys Design Compiler (C-2009.06-SP3) to map the CubeHash RTL codes to UMC 90nm (FSD0A_ A_ GENERIC_ CORE_ 1D0V_ TP_ 2007Q1v1.7) and 130nm (FSC0G_ D_ SC_ TP_ 2006Q1v2.0) technologies. We use the typical case condition characterization of the standard cell libraries. The 90nm technology uses 9 metal layers, and the 130nm technology uses 8 metal layers. In general, more metal layers allow for a denser interconnect, and hence a more optimal use of die area.

1. *MinArea*: A minimum-area design will minimize the use of logic resources (gates) at the expense of performance.
2. *MaxSpeed*: A maximum-speed design will minimize the computational delay of the design, at the expense of area.
3. *TradeOff0*: The first trade-off point is chosen to have a computational delay which is two-thirds between the MinArea and MaxSpeed design points.
4. *TradeOff1*: The second trade-off point is chosen to have a computational delay which is five-sixths between the MinArea and MaxSpeed design points.

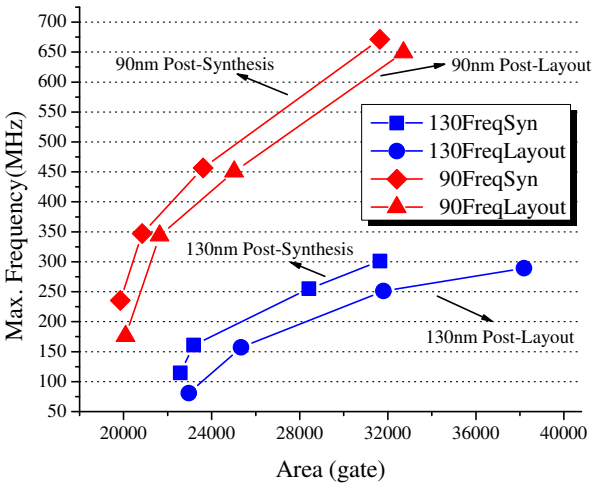

Fig. 3. CubeHash-256 area and speed results.

The TradeOff points are chosen to investigate how the relationship (speed, area) evolves when a design gradually moves from the MinArea design point to the MaxSpeed design point.

The Synopsys IC Compiler (C-2009.06-SP5) is used for the back-end process. For all the designs we start with 85% utilization of the core area. The utilization is defined as the die area devoted to active components (standard cells) as compared to the total die area. Due to the routing of signals, power, and ground between active components, utilization can never reach a 100%. The optimal value for utilization should be as high as possible. After place-and-route, design flow errors such as timing and Design Rule Check (DRC) violations may occur. In that case, the initial utilization must be lowered in order to relax the constraints to the place-and-route process.

The timing results can be obtained from the post-synthesis and post-layout steps. First, the Synopsys IC Compiler is used to extract the post-layout parasitic and generate an SDF file containing the delays of all the interconnections and instances. Second, Synopsys VCS can be used to do the post-simulation and generate the VCD file that records all the switching activities of the netlist. Finally, Synopsys Prime Time (C-2009.06-SP3) reads the final netlist, VCD file and .spef parasitic file and does the power estimation.

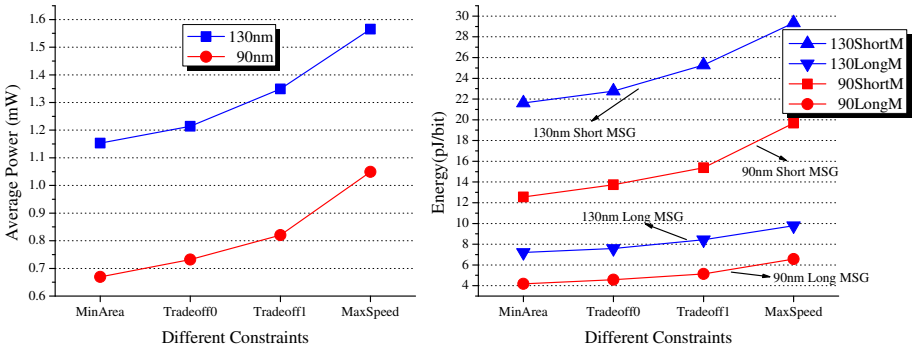

Fig. 4. CubeHash-256 power and energy results.

Fig. 3 and Fig. 4 show the results of these technology parameters on the implementation of the Cubehash-256 SHA-3 candidate. Fig. 3 is an area-delay plot, which marks the area of a given design against the achievable performance (in this case, the maximum clock frequency). The X-axis of Fig. 3 is calibrated in equivalent gates. This means that the area is normalized to a standard 2-input NAND gate in the chosen technology. Fig. 4 is the power and energy plot that illustrates the impact of different design optimization constraints, technology, and message characteristics. The left pane of Fig. 4 indicates the average power dissipation during the processing of a very long message. The right pane of Fig.

4 indicates the energy dissipation per bit during the processing of messages of variable length.

- *The impact of different technologies.* The relationship between 130nm and 90nm technologies, as shown in Fig. 3, is non-trivial. However, one can notice that the relative relationship between the four points on each curve is similar. This means that a characterization in a single technology can also serve as a characterization in nearby technology nodes. In our experiments, we concentrated on area-delay characterization in 130nm technology.
- *The impact of different constraints.* As illustrated in Fig. 3, the impact of constraints (MinArea, MaxSpeed, TradeOff0, TradeOff1) is significant, and it varies the performance by a factor of almost 3. In exploring the 14 SHA-3 candidates, we have therefore fully characterized the 4 design points of each design in 130nm technology.
- *Post-Synthesis results vs. Post-Layout results.* Fig. 3 illustrates obvious differences between post-synthesis and post-layout results. Because post-synthesis results provide higher accuracy, we have obtained post place-and-route results for all 14 SHA-3 candidates.
- *The impact of message length.* From the energy results shown in Fig. 4, we can clearly see that energy per message bit changes a lot when considering different message lengths. Note that the power consumption is the same for CubeHash message update step and finalization step since those two steps calls the same round functions with different rounds. The cause of the energy differences is due to the different throughputs and latencies for short and long messages.

4 ASIC Implementation Results

In this section we present the performance results of the SHA-3 ASIC implementations with the UMC 130nm standard cell technology. Design space exploration is performed for all the 14 second round candidates. For each of the graphs shown below there will be 4 points on the curve representing the Min Area, Max Speed and two tradeoffs points.

In Fig. 5, the throughput is calculated based on the maximum clock frequency of the post-layout design and only consider hashing long messages. The impact of message length to the final results has been partially addressed in the analysis of results shown in Fig. 4. We also report the results for short and long message cases in Table 1.

Figure 5 illustrates how architecture differences affect the performance results. Some curves, like those of Keccak and Luffa, are very steep. This means that a small increase in area yields a significant performance improvement. Other curves however are relatively flat. For a design such as SIMD, for example, even a large addition of gates will not yield additional performance. The optimal points in Figure 5 are those with maximal performance and minimum area. This optimum is located on the upper left side of the graph. The curves of Keccak and Luffa are clearly out-shadowing other designs.

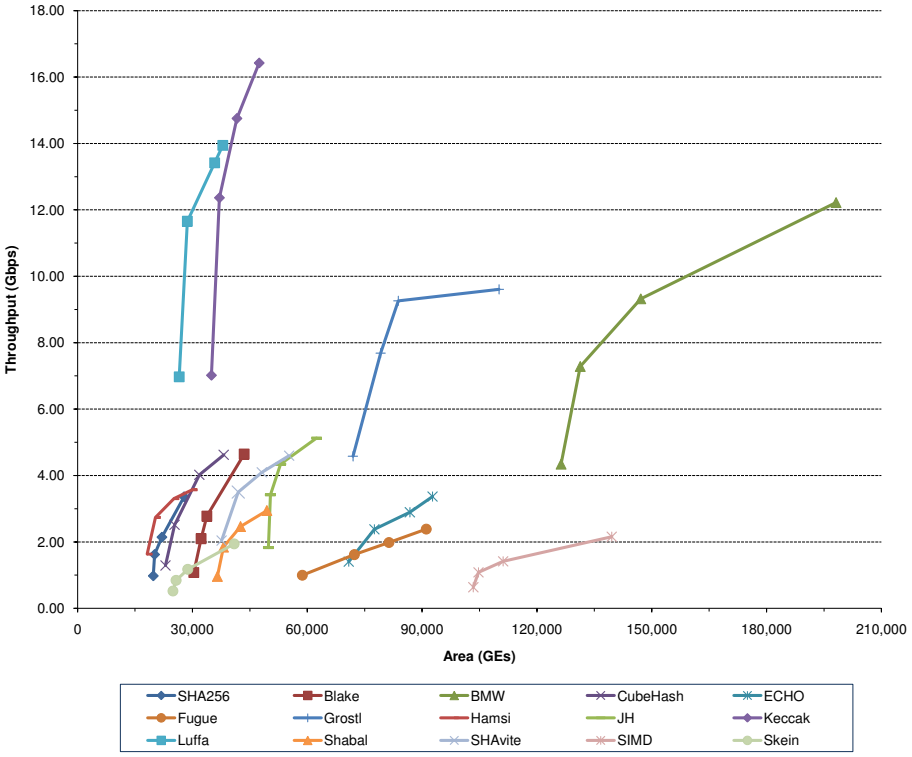

Fig. 5. Post-Layout results for throughputs and gate counts.

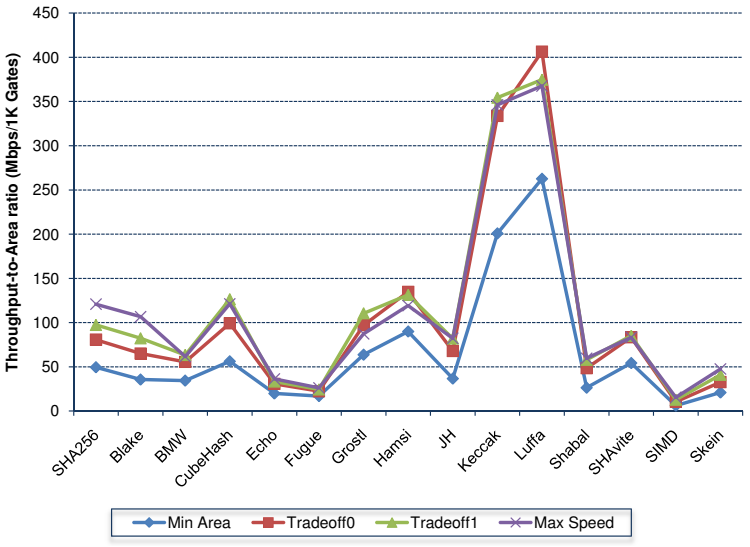

Fig. 6. Throughput-to-Area ratio for all the designs with 4 different constraints.

To compare the results of the SHA-3 candidates, we use the methodology proposed by Gaj [7]. Therefore, we utilize a uniform metric, Throughput-to-Area Ratio, as the primary metric to rank all the designs. The SHA-3 design with higher Throughput-to-Area ratio means with given fixed hardware resources this SHA-3 candidate has better efficiency (hash more message in the same period of time).

Fig. 6 shows the Throughput-to-Area ratio graph for all the 14 SHA-3 candidates. We can also observe how this 'efficiency' metric changes according to different constraints. By looking at the results shown in Fig. 6, if only considering the 'Throughput-to-Area ratio' metric, the ranking of the 14 SHA-3 designs can be found in Table 1. The SHA-256 is also included to serve as a reference.

Table 1. Ranking of the 14 SHA-3 designs in terms of Throughput-to-Area ratio metric

Rank	*MinArea*	$Tradeoff0$	$Tradeoff1$	*MaxSpeed*
1	Luffa	Luffa	Luffa	Luffa
2	Keccak	Keccak	Keccak	Keccak
3	Hamsi	Hamsi	Hamsi	CubeHash
4	Grøstl	CubeHash	CubeHash	SHA256
5	CubeHash	Grøstl	Grøstl	Hamsi
6	SHAvite	SHAvite	SHA256	Blake
7	SHA256	SHA256	SHAvite	Grøstl
8	JH	JH	Blake	SHAvite
9	Blake	Blake	JH	JH
10	BMW	BMW	BMW	BMW
11	Shabal	Shabal	Shabal	Shabal
12	Skein	Skein	Skein	Skein
13	Echo	Echo	Echo	Echo
14	Fugue	Fugue	Fugue	Fugue
15	SIMD	SIMD	SIMD	SIMD

Although it is not necessary that the new SHA-3 standard has to be better than the existing SHA-256 in terms of performance, still one would be interesting to see the comparison results. In Fig. 7, for all the 4 cases, the Throughput-to-Area ratio of all the designs has been normalized to the value of SHA-256. All the points that are above the red line which denotes value one can be deemed as outperforming the SHA-256.

For detailed analysis we have shown all the results in Table 2. For the reason of selecting those metrics and how the results are derived you may refer to a complementary submission [4] for details.

Table 2. Performance results of post-layout designs of the SHA-3 14 candidates with UMC 130nm technology

		Block	Max	# of cycles		LongMSG	ShortMSG	Area
		Size	Freq.	IF+Core	Core	TP[Mbps]	Latency[us]	[Gates]
SHA256	MinA	512	130	148(196)	68(68)	450(979)	3.81(1.57)	19789
	MaxS	512	446	148(196)	68(68)	1544(3361)	1.11(0.46)	27816
Blake	MinA	512	46	121(169)	22(22)	196(1080)	8.92(1.42)	30365
	MaxS	512	200	121(169)	22(22)	845(4645)	2.07(0.33)	43521
BMW	MinA	512	17	98(148)	2(4)	89(4345)	20.27(0.35)	126315
	MaxS	512	48	98(148)	2(4)	249(12220)	7.21(0.13)	198167
Cubehash	MinA	256	81	64(272)	16(176)	323(1290)	6.58(2.98)	22968
	MaxS	256	289	64(272)	16(176)	1156(4624)	1.84(0.83)	38184
ECHO	MinA	1536	90	407(455)	99(99)	342(1404)	5.06(1.09)	70850
	MaxS	1536	217	407(455)	99(99)	819(3366)	2.11(0.46)	92727
Fugue	MinA	32	62	8(93)	2(39)	249(995)	5.61(1.66)	58705
	MaxS	32	149	8(93)	2(39)	596(2385)	2.34(0.69)	91089
Grøstl	MinA	512	89	106(164)	10(20)	432(4580)	4.24(0.45)	71933
	MaxS	512	188	106(164)	10(20)	906(9606)	2.00(0.21)	110108
Hamsi	MinA	32	204	10(63)	4(9)	653(1633)	1.96(0.70)	18159
	MaxS	32	446	10(63)	4(9)	1429(3571)	0.90(0.32)	29941
JH	MinA	512	139	135(183)	39(39)	512(1828)	3.25(0.84)	49871
	MaxS	512	391	135(183)	39(39)	1481(5128)	1.16(0.30)	62417
Keccak	MinA	1024	161	217(265)	25(25)	761(6606)	2.99(0.31)	34959
	MaxS	1024	377	217(265)	25(25)	1781(15457)	1.28(0.13)	47434
Luffa	MinA	256	245	57(114)	9(18)	1101(6972)	1.41(0.22)	26551
	MaxS	256	490	57(114)	9(18)	2202(13943)	0.70(0.11)	37942
Shabal	MinA	512	118	143(341)	50(200)	424(962)	5.33(2.87)	36516
	MaxS	512	362	143(341)	50(200)	1297(2945)	1.74(0.94)	49439
SHAvite	MinA	512	152	134(185)	38(38)	579(2041)	2.97(0.55)	37621
	MaxS	512	341	134(185)	38(38)	1304(4599)	1.32(0.33)	55245
SIMD	MinA	512	57	142(190)	46(46)	206(636)	8.30(2.42)	103379
	MaxS	512	194	142(190)	46(46)	699(2157)	2.45(0.71)	139547
Skein	MinA	256	43	75(143)	21(41)	146(521)	8.67(2.43)	24919
	MaxS	256	159	75(143)	21(41)	544(1941)	2.33(0.65)	40899

1. 'I/F+Core' cycle counts is equal to $I_{in} + I_{core}(I)$.
2. 'Core' cycle counts is equal to $I_{core}(I_{core} + I_{final})$.
3. LongMSG and ShortMSG cases include the communication overhead by interface.
4. The values in parenthesis are the case excluding the interface overhead,
e,g. only the core function block.

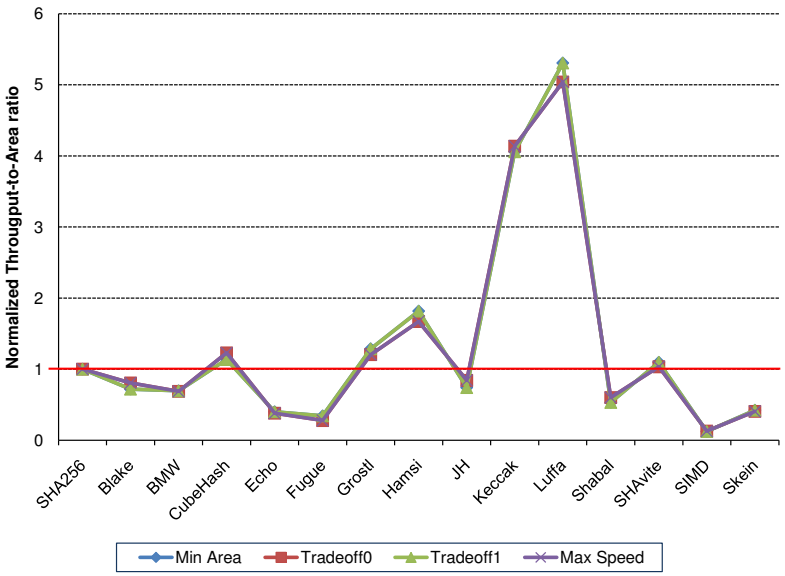

Fig. 7. Normalized Throughput-to-Area ratio for all the designs with 4 different constraints.

5 Conclusions

In this paper, we presented performance evaluation results of 14 SHA-3 candidates in a 130nm CMOS ASIC Technology. We discussed the impacts of various factors including technology, design constraints, place-and-route, and hash operating modes. We conclude that top-performing candidates in our experiment include Luffa, Keccak, Hamsi, Cubehash, and Grøestl. We intend to open-source the RTL versions of the SHA-3 designs that we evaluated [4].

Acknowledgments. The effort reported in this paper was supported by a NIST Measurement, Science and Engineering Grant ("Environment for Fair and Comprehensive Performance Evaluation of Cryptographic Hardware and Software").

References

1. Regenscheid, A., Perlner, R., Chang, S., Kelsey, J., Nandi, M., Paul, S.: Status Report on the First Round of the SHA-3 Cryptographic Hash Algorithm Competition. NIST IT 7620, Information Technology Laboratory, NIST, MD, http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/documents/sha3_NISTIR7620.pdf, Aug. 2010.
2. National Institute of Standards and Technology: Second Round Candidates, http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/submissions_rnd2.html, Aug. 2010.

3. Bernstein, D., Lange, T. (editors): eBACS: ECRYPT Benchmarking of Cryptographic Systems, http://bench.cr.yp.to, Aug. 2010.
4. S. Matsuo, M. Knezevic, P. Schaumont, I. Verbauwhede, A. Satoh, K. Sakiyama, K. Ota, "How can we conduct fair and consistent hardware evaluation for SHA-3 candidate?", The Second SHA-3 Candidate Conference by NIST, 2010.
5. ECRYPT. The eSTREAM project, http://www.ecrypt.eu.org/stream/, Aug. 2010.
6. Z. Chen, S. Morozov, and P. Schaumont, A Hardware Interface for Hashing Algorithms,IACR ePrint archive, 2008/529, 2008.
7. K. Gaj, E. Hamsirikamal, M. Rogawski, "Fair and Comprehensive Methodology for Comparing Hardware Performance of Fourteen Round Two SHA-3 Candidates using FPGA's", Proc. CHES2010, 2010.