Comprehensive Comparison of Hardware Performance of Fourteen Round 2 SHA-3 Candidates with 512-bit Outputs Using Field Programmable Gate Arrays[1]

Kris Gaj, Ekawat Homsirikamol, and Marcin Rogawski

ECE Department, George Mason University

{kgaj, ehomsiri, mrogawsk}@gmu.edu

Abstract. In this paper, we extend our evaluation of the hardware performance of 14 Round 2 SHA-3 candidates, presented at CHES 2010, to the case of high security variants, with 512 bit outputs. A straightforward method for predicting the performance of 512-bit variants, based on the results for 256-bit versions of investigated hash functions is presented, and confirmed experimentally. The VHDL codes for 512-bit variants of all 14 SHA-3 Round 2 candidates and the old standard SHA-2 have been developed and thoroughly verified. These codes have been then used to evaluate the relative performance of all aforementioned algorithms using seven modern families of Field Programmable Gate Arrays (FPGAs) from two major vendors, Xilinx and Altera. The results point to very significant differences among all evaluated algorithms in terms of both throughput and area. Only two candidates, Keccak and CubeHash, outperform SHA-512 in terms of the primary optimization target used in this study, throughput to area ratio.

1. Introduction

Both the current NIST cryptographic hash function standard, FIPS 180-3 [1] (commonly referred as SHA-2) as well as the call for a new standard, SHA-3 [2], assume that each hash function family includes variants with at least the following four output sizes: 224, 256, 384, and 512-bits. These variants should have a security equivalent to Triple DES, AES-128, AES-192, and AES-256, respectively.

Although 256-bit versions of cryptographic hash functions seem to provide adequate security for majority of current applications and common use scenarios, there exist already several recommendations suggesting the use of more secure hash function variants, with the outputs of 384 and 512 bits. For example, if a hash function is used as a part of a digital signature used to authenticate a life will, which is required to remain valid for tens of years from now, a 512-bit variant of a hash function seems to be a prudent choice.

Several recent recommendations clearly specify the need for such high-security variants [3]. Examples include

- Federal documents requiring protection well beyond the year 2030, according to the 2007 NIST recommendation [4],
- Top Secret Documents according to the NSA Suite B Cryptography Fact Sheet [5],
- Level 8 of protection according to the recent ECRYPT II Recommendations [6].

Clearly, candidates for the new SHA-3 standard, which is likely to remain in effect well beyond the year 2030, should be evaluated from the point of view of performance of their most secure variant.

Since replacing a 256-variant of a hash function with a 512-bit variant increases the resistance against the best known (birthday-paradox) attack from 2^{128} to 2^{256} (i.e., by a factor of $2^{128} \approx 3.4\ 10^{38}$), one might expect that a significant performance penalty will be incurred for such a tremendous increase in security.

[1] This work was partially supported through the NIST/ARRA grant no. 60NANB10D004.

Contrary to that, it has been observed that the 512-bit variant of the current standard, SHA-2 (known as SHA-512), is actually about 33% faster then even the 160-bit variant (SHA-1), when implemented in hardware using Xilinx Virtex FPGAs [7]. The only penalty incurred concerns the area (by a factor of about two), and not the speed of the implementation [7].

The explanation of this phenomena is quite simple - in hardware all bits of a message block are processed in parallel, so increasing the size of a message block from 512 bits to 1024 bits has a positive influence on speed, which is counteracted only by more complex operations in the critical path of the circuit (namely six 64-bit additions vs. four 32-bit additions), and thus a smaller clock frequency.

The same property clearly does not apply to traditional software implementations, where doubling message block size typically at least doubles the amount of clock cycles required for processing of this block.

In this paper, we will investigate whether the increase in the speed of a more secure hash function variant, first observed in the case of SHA-1 and SHA-2 functions, applies also to new SHA-3 candidates. We will also explore the imposed area penalty (if any), and the change in the throughput to area ratio when switching from a 256-bit variant to a 512-bit variant of a hash function.

Finally, we will also explore the relative performance of the 512-bit variants of all SHA-3 candidates in terms of the throughput, area, and throughput to area ratio.

2. Previous work

At the time of writing, relatively few hardware implementations of the 512-bit variants of SHA-3 candidates have been reported in the literature. Major results concerning FPGA implementations targeting high speed are summarized in [8, 9]. These results have been obtained using different FPGA families and different and not always clear optimization targets. The designers differ with experience and skills. Additionally, no common interface has been applied, and some of the designs are not fully autonomous but rather implement core functionality only [9].

The comparison of 256-bit variants of all candidates is somewhat more explored, with two groups reporting a full set of FPGA results [10, 11], two groups reporting ASIC results [12, 13], and several other groups reporting results for a subset of all candidates [14-20].

3. Design Methodology

Our design and evaluation methodologies follow exactly the approach outlined in our earlier CHES 2010 paper [11] on comparison of SHA-3 candidate variants with 256-bit outputs.

Our study is comprehensive as it covers all 14 SHA-3 candidates, and presents results for seven major families of FPGAs from two major vendors: Xilinx and Altera. The results for 512-bit variants of all candidates are compared with the results for 256-bit variants, implemented using the same language, tools, design methodology, and coding style, and reported earlier by our team in [11].

The fairness of our comparison is assured by using

- *firm optimization target*, the throughput to area ratio, that clearly guides the entire development process from the choice of a high-level architecture, through the implementation of basic operations, to the choice of low-level tool options;
- the *same library of basic operations* that are used in more than one SHA-3 candidate (such as AES SubBytes, Binary Galois Field multiplications by small constants (x02..x07), two-operand and multi-operand addition, etc. (see Table 2 in [11])

- *identical input/output interface*, proposed earlier by our group [11, 21], and applied consistently to all 256-bit and 512-bit variants of all compared algorithms;
- the *same assumptions and simplifications*, such as no padding in hardware, and no support for special modes of operation, such as tree hashing or MAC.
- the *same tools, options of tools*, and *identical optimization effort* in case of each of the evaluated hash functions;
- a *small team of designers* with similar skills, who closely collaborate with each other, and review each other's codes.

The results are then normalized by dividing them by equivalent results for the current SHA-2 standard (variant with equivalent security). Normalized results are then averaged for all investigated FPGA families.

All VHDL codes have been thoroughly verified using a universal testbench [22], capable of testing an arbitrary hash function core that follows interface described in [11, 21]. A special padding script was developed in Perl in order to pad messages included in the Known Answer Test (KAT) files distributed as a part of each candidate's submission package.

For synthesis and implementation, we have used tools developed by FPGA vendors themselves:
- for Xilinx: Xilinx ISE Design Suite v. 11.1, including Xilinx XST,
- for Altera: Quartus II v. 9.1 Subscription Edition Software.

The generation of a large number of results was facilitated by an open source benchmarking environment, called ATHENa (Automated Tool for Hardware EvaluatioN), developed at George Mason University [22, 23].

4. Performance Measures

The three most important performance measures we use to characterize our hardware implementations of hash functions are: Throughput, Area, and Throughput to Area Ratio. Below we characterize each of these measures one by one.

4.1. Throughput

The Throughput is understood as the throughput for long messages, and does not take into account the time taken for reading the very first block of the message, initialization, finalization, and writing the hash value to the output memory. To be exact, we define Throughput using the following formula:

$$Thr = \frac{Block\_size}{T \bullet (Htime(N+1) - HTime(N))} \quad (1)$$

where *Block_size* is a message block size, characteristic for each hash function (as defined in the function specification, and shown in Table 2), *HTime*(*N*) is a total number of clock cycles necessary to hash an N-block message, *T* is a clock period, different and characteristic for each hardware implementation of a specific hash function.

All our designs follow the same interface, described in detail in [11, 21]. This interface has the following two variable parameters:

- w = the width of the input data bus, *din*, and the output data bus, *dout*. These buses are independent of each other, and both have the width w.
- $r_{IO} = Freq_{IO\_CLK}/Freq_{CLK}$, i.e., the ratio of the clock frequency for the fast I/O clock (used only for the fast communication with the surrounding circuits, typically Input and Output FIFOs), and

the clock frequency for the main clock used for data processing. If only one clock is used for both functions, $r_{IO}=1$.

The general formula for the time necessary to hash N blocks of the message can be written in the following form:

$$HTime(N) = c_{INIT} + c_{IN}/r_{IO} + c_{BLOCK} \cdot N + c_{FINAL} + c_{OUT}/r_{IO} \quad (2)$$

In this formula:

- c_{INIT} is the number of clock cycles necessary to establish communication with the source of data (typically, Input FIFO) and read the length of the message (in our formulas we assume that the length of the message is smaller than 2^w).
- c_{IN} is the number of clock cycles required to read the very first block of the message. $c_{IN} = Block\_size/w$.
- c_{BLOCK} is the number of clock cycles required to process one block of the message.
- c_{FINAL} is the number of clock cycles required for the finalization. We assume that only one finalization is required per entire message (if the finalization needs to be repeated for every block of the message, its number of clock cycles is included in c_{BLOCK}).
- c_{OUT} is the number of clock cycles required to write hash value to the destination circuit (typically Output FIFO). $c_{OUT}=output\_size/w$.

Table 1. The I/O Data Bus Width (in bits), Hash Function Execution Time (in clock cycles), and Throughput (in Mbits/s) for the 256-bit and 512-bit variants of all SHA-3 candidates and the current standard, SHA-2. T denotes the clock period in µs. Values different between 256-bit and 512-bit variants are shown in bold.

	256-bit variants			512-bit variants		
	I/O Bus Width	**Hash Time [cycles]**	**Throughput [Mbit/s]**	**I/O Bus Width**	**Hash Time [cycles]**	**Throughput [Mbit/s]**
BLAKE	64	2+8+21·N+4	512/(21·T)	64	2+**16**+**29**·N+**8**	**1024**/(**29**·T)
BMW	64	2+8/8+N+1	512/T	64	2+**16/16**+N+**8/16**	**1024**/T
CubeHash	64	2+4+16·N+160+4	256/(16·T)	64	2+4+16·N+160+**8**	256/(16·T)
ECHO	64	3+24+27·N+4	1536/(27·T)	64	3+**16**+**31**·N+**8**	**1024**/(**31**·T)
Fugue	32	2+N+18+8	32/T	32	2+**4**·N+**21+16**	32/(**4**·T)
Groestl	64	2+8+21·N+4	512/(21·T)	64	2+**16**+29·N+**8**	**1024**/(**29**·T)
Hamsi	32	3+1+3·(N-1)+6+8	32/(3·T)	**64**	3+1+**6**·(N-1)+6+8	**64**/(**6**·T)
JH	64	3+8+36·N+4	512/(36·T)	64	3+8+36·N+**8**	512/(36·T)
Keccak	64	3+17+24·N+4	1088/(24·T)	64	3+**9**+24·N+**8**	**576**/(24·T)
Luffa	64	3+4+9·N+9+4	256/(9·T)	64	3+4+9·N+**2**·9+**8**	256/(9·T)
Shabal	64	3+8+1+25·N+ 3·25+4	512/(25·T)	64	3+8+1+25·N+ 3·25+**8**	512/(25·T)
SHAvite-3	64	3+8+37·N+4	512/(37·T)	64	3+16+**57**·N+**8**	**1024**/(**57**·T)
SIMD	64	3+8+8+9·N+4	512/(9·T)	64	3+**16**+**9**+9·N+**8**	**1024**/(9·T)
Skein	64	2+4+19·N+4	256/(19·T)	64	2+**8**+19·N+**8**	**512**/(19·T)
SHA-2	32	2+1+65·N+8	512/(65·T)	**64**	2+1+**81**·N+8	**1024**/(**81**·T)

The ratio of the I/O clock frequency to the main clock frequency is selected in such a way that the following condition, given by Eq. (3) holds:

$$c_{IN}/r_{IO} \leq c_{BLOCK}. \quad (3)$$

This condition assures that any next message block (i.e. any block other than the very first block) can be read in parallel with processing of the previous block.

In Table 2, we summarize the formulas for the Hash Function Execution Time and the Throughput for all investigated algorithms. All formulas for the Hash Time, *HTime(N)*, are written in agreement with Eq. (2). If c_{FINAL}=0 for the given algorithm, this term is omitted in the equation.

The I/O bus width, *w*, was selected to be equal to 64 for majority of algorithms in order to limit the pin requirements of the hash modules. The only exceptions are Fugue-256, Hamsi-256, and Fugue-512, for which we choose *w*=32, because they all have block size equal to 32 bits, and thus cannot be sped up by using a wider I/O data bus. Similarly, SHA-256 can start processing data after receiving just one 32-bit word, and cannot be easily sped-up by using a wider input data bus. The fast I/O clock is required only in BMW-256 (r_{IO}=*8*) and BMW-512 (r_{IO}=*16*).

4.2. Area

In general the resource utilization in FPGAs, is a vector, with coordinates specific to the given FPGA family. For example,

$$Resource\ Utilization_{\text{Spartan3}} = (\#CLB\ slices, \#BRAMs, \#MULs) \quad (4)$$

$$Resource\ Utilization_{\text{Cyclone III}} = (\#LE, \#memory\_bits, \#MULs). \quad (5)$$

Taking into account that vectors cannot be easily compared to each other, we have decided to opt out of using any dedicated resources in the hash function implementations used for our comparison. Thus, all coordinates of our vectors, other than the first one have been forced (by choosing appropriate options of the synthesis and implementation tools) to be zero. This way, our resource utilization (further referred to as *Area*) is characterized using a single number, specific to the given family of FPGAs, namely the number of CLB slices (*#CLB_slices*) for Xilinx FPGAs, the number of Logic Elements (*#LE*) for Cyclone II and Cyclone III, and the number of Adaptive Look-Up Tables (#ALUTs) in Stratix II and Stratix III.

We believe that majority of SHA-3 candidates will be most naturally implemented without the use of dedicated logic resources. The capability of using such resources should be treated as a measure of the algorithm flexibility, and may be investigated in our future publications.

5. Relative Performance of the 512 and 256-bit Variants of the SHA-3 Candidates

In Table 2, we summarize major parameters of the 512 and 256-bit variants of the SHA-3 candidates and SHA-2.

The ratio of the area of the 512-bit variant to the 256-bit variant depends primarily on the datapath width. In all our hardware architectures, due to the optimization for the maximum throughput to area ratio, the datapath width is equal to the state size. As a result, the area ratio can be approximated very roughly as the state size ratio, as shown in Eq. (6) below:

$$\frac{Area(512)}{Area(256)} \approx \frac{Datapath\_width(512)}{Datapath\_width(256)} = \frac{State\_size(512)}{State\_size(256)} \quad (6)$$

Table 2. Major parameters of the 256-bit and 512-bit variants of all SHA-3 candidates and the current standard, SHA-2. Values different between 256-bit and 512-bit variants are shown in bold. The first approximations of the predicted area ratio (512 vs. 256-bit variant) and the predicted throughput ratio (512 vs. 256-bit variant) are given in the last two columns.

	256-bit variant				512-bit variant				Predicted Area Ratio (based on Eq. (6))	Predicted Thr Ratio (based on Eq. (8))
	State size	Block size	Round no	Word size	State size	Block size	Round no	Word size		
BLAKE	**512**	**512**	**10**	32	**1024**	**1024**	**14**	**64**	**2**	**1.43**
BMW	**512**	**512**	**16**	32	**1024**	**1024**	**16**	**64**	**2**	**2**
CubeHash	1024	256	16	32	1024	256	16	32	1	1
ECHO	2048	**1536**	**8**	32	2048	**1024**	**10**	32	1	**0.53**
Fugue	**960**	32	**2**	32	**1152**	32	**4**	32	**1.2**	**0.5**
Groestl	**512**	**512**	**10**	64	**1024**	**1024**	**14**	64	**2**	**1.43**
Hamsi	**512**	**32**	**3**	32	**1024**	**64**	**6**	32	**2**	**1**
JH	1024	512	36	64	1024	512	36	64	1	1
Keccak	1600	**1088**	24	64	1600	**576**	24	64	1	**0.53**
Luffa	**768**	256	8	32	**1280**	256	8	32	**1.67**	**1**
Shabal	1408	512	48	32	1408	512	48	32	1	1
SHAvite-3	**512**	**512**	**36**	32	**1024**	**1024**	**56**	32	**2**	**1.29**
SIMD	**512**	**512**	36	32	**1024**	**1024**	36	32	**2**	**2**
Skein	**256**	**256**	72	64	**512**	**512**	72	64	**2**	**2**
SHA-2	**256**	**512**	64	**32**	**512**	**1024**	**80**	**64**	**2**	**1.60**

The additional factors that affect this ratio include:

- *message block size*, which determines the size of the input shift register
- *output size*, which determines the size of the output shift register
- *logic of the main round*, which may be more complex in case of a 512-bit variant of a function
- logic required for *initialization and finalization*, which may not follow the datapath width
- size of the *control unit*, which is likely to remain constant between two variants, but typically contributes only small percentage to the total circuit area.

All these factors cause that the Eq. (6) is only the first approximation, and the actual results may vary and may be dependent on a particular FPGA family.

The throughput of each variant is given by

$$Thr(k) = \frac{Block\_size(k)}{c \bullet Round\_no(k) \bullet T(k, Word\_size(k))} \tag{7}$$

where

- k denotes output size, 256 or 512 bits;
- c is a number of main rounds executed in a single clock cycle (possibly a fraction). In our implementations, this number is constant and independent of the function variant.
- $T(k, Word\_size(k))$ is a minimum clock period, which is a function of the logic included in the main round, and in particular of the word size.

In majority of considered algorithms, with the exception of BLAKE, BMW, and SHA-2, the word size remains the same between the two variants. Additionally, the logic of the main round remains either the

same, or at least has the similar critical path. As a result, the following first order approximation, given in Eq. (8), can be used to estimate the throughput ratio:

$$\frac{Thr(512)}{Thr(256)} \approx \frac{\frac{Block\_size(512)}{Block\_size(256)}}{\frac{Round\_no(512)}{Round\_no(256)}} = \frac{Block\_size\_ratio}{Round\_no\_ratio} \tag{8}$$

For BLAKE, BMW, and SHA-2, the ratio is expected to be smaller because of the increase in the word size from 32 bits to 64-bits, and the influence of this change on the delay of the multi-operand additions, which appear in the critical paths of these algorithms. At the same time, this effect is expected to be significantly smaller than 2, because of

- the properties of the fast carry chain adders embedded in Xilinx and Altera FPGAs (the delay of these adders as a function of the number of bits, n, is given by $d(n) = a \cdot n + b$, with the relatively large b and small a); and
- the fact that the multi-operand adder constitutes only a fraction of the critical path.

The first rough approximations of the area ratio (based on Eq. (6)) and the throughput ratio (based on Eq. (8)) are given in the last two columns of Table 2. Based on these approximations, we can divide 15 investigated algorithms into the following 6 major groups:

- Group 1: area and throughput are not affected by the change of the output size: *CubeHash*, *JH*, *Shabal*.
- Group 2: area and throughput both double: *BMW*, *SIMD*, *Skein*.
- Group 3: area and throughput both increase, but area increases more: *BLAKE*, *Groestl*, *SHAvite-3*, and *SHA-2*.
- Group 4: area stays the same and throughput decreases: *ECHO*, *Keccak*.
- Group 5: throughput stays the same and area increases: *Hamsi*, *Luffa*.
- Group 6: area increases and throughput decreases: *Fugue*.

Table 3. Major performance measures of SHA-3 candidates (512-bit and 256-bit variants) when implemented in Xilinx Virtex 5 FPGAs

	Max Clk Freq [MHz]			Throughput [Mbit/s]			Area [CLB slices]			Throughput/Area		
	512	**256**	**ratio**	**512**	**256**	**ratio**	**512**	**256**	**ratio**	**512**	**256**	**ratio**
BLAKE	106.01	117.06	0.91	3743.28	2853.91	1.31	3276	1871	1.75	1.14	1.53	0.75
BMW	8.45	10.89	0.78	8655.87	5576.70	1.55	10401	4400	2.36	0.83	1.27	0.66
CubeHash	215.33	219.30	0.98	3508.77	3445.31	1.02	707	764	0.93	4.59	4.87	0.94
ECHO	200.97	234.85	0.86	6430.88	13874.33	0.46	5958	5445	1.09	1.08	2.55	0.42
Fugue	138.49	98.47	1.41	1107.88	3151.17	0.35	924	956	0.97	1.20	3.30	0.36
Groestl	180.15	355.87	0.51	6361.09	8676.50	0.73	3466	1884	1.84	1.84	4.61	0.40
Hamsi	171.38	248.08	0.69	1828.05	2646.15	0.69	2201	946	2.33	0.83	2.80	0.30
JH	275.48	278.09	0.99	3917.97	3955.02	0.99	1165	1108	1.05	3.36	3.57	0.94
Keccak	276.86	238.38	1.16	6644.52	10806.51	0.61	1236	1229	1.01	5.38	8.79	0.61
Luffa	220.12	281.53	0.78	7043.81	8008.02	0.88	2164	1154	1.88	3.25	6.94	0.47
Shabal	135.30	128.12	1.06	2770.94	2623.96	1.06	1372	1266	1.08	2.02	2.07	0.97
SHAvite-3	213.45	208.55	1.02	3834.56	2885.89	1.33	1954	1130	1.73	1.96	2.55	0.77
SIMD	36.37	40.89	0.89	4138.55	2325.90	1.78	17016	9288	1.83	0.24	0.25	0.97
Skein	104.34	116.35	0.90	2811.72	1567.62	1.79	1520	843	1.80	1.85	1.86	0.99
SHA-2	215.84	207.00	1.04	2728.68	1630.49	1.67	646	433	1.49	4.22	3.77	1.12

Table 4. Major performance measures of SHA-3 candidates (512-bit and 256-bit variants) when implemented in Altera Stratix III FPGAs

	Max Clk Freq [MHz]			Throughput [Mbit/s]			Area [ALUTs]			Throughput/Area		
	512	256	ratio	512	256	ratio	512	256	ratio	512	256	ratio
BLAKE	93.41	124.55	0.75	3298.34	3036.65	1.09	3414	1779	1.92	0.97	1.71	0.57
BMW	7.44	16.45	0.45	7618.56	8422.40	0.90	25225	12632	2.00	0.30	0.67	0.45
CubeHash	218.05	236.07	0.92	3488.80	3777.12	0.92	1924	1928	1.00	1.81	1.96	0.93
ECHO	246.00	164.20	1.50	7872.00	9700.43	0.81	20085	21689	0.93	0.39	0.45	0.88
Fugue	206.27	123.64	1.67	1650.16	3956.48	0.42	2775	3594	0.77	0.59	1.10	0.54
Groestl	250.38	270.27	0.93	8841.00	6589.44	1.34	6288	3103	2.03	1.41	2.12	0.66
Hamsi	181.16	294.81	0.61	1932.37	3144.64	0.61	5668	2320	2.44	0.34	1.36	0.25
JH	358.94	364.96	0.98	5104.92	5190.54	0.98	3222	3107	1.04	1.58	1.67	0.95
Keccak	269.61	296.30	0.91	6470.64	13432.27	0.48	3575	4458	0.80	1.81	3.01	0.60
Luffa	268.02	307.31	0.87	8576.64	8741.26	0.98	6888	3304	2.08	1.25	2.65	0.47
Shabal	126.44	126.87	1.00	2589.49	2598.30	1.00	3753	3600	1.04	0.69	0.72	0.96
SHAvite-3	215.38	255.00	0.84	3869.28	3528.65	1.10	5610	2497	2.25	0.69	1.41	0.49
SIMD	43.38	47.40	0.92	4935.68	2696.53	1.83	47671	22376	2.13	0.10	0.12	0.86
Skein	7.44	16.45	0.45	7618.56	8422.40	0.90	25225	12632	2.00	0.30	0.67	0.45
SHA-2	234.80	212.81	1.10	2968.34	1676.29	1.77	1620	963	1.68	1.83	1.74	1.05

Table 5. Ratio of the respective performance measures (Throughput (Thr), Area, Throughput to Area Ratio (Thr/Area)) for a 512-bit variant vs. 256-bit variant, averaged (using geometric mean) over all 7 FPGA families (Overall), 3 Xilinx families, and 4 Altera Families.

	Overall 512 vs. 256 variant			Xilinx Families 512 vs. 256 variant			Altera Families 512 vs. 256 variant		
	Area	Thr	Thr/Area	Area	Thr	Thr/Area	Area	Thr	Thr/Area
BLAKE	1.89	1.13	0.60	1.81	1.26	0.69	1.93	1.07	0.55
BMW	1.99	1.11	0.56	1.99	1.23	0.62	2.00	0.90	0.45
CubeHash	0.97	0.97	0.89	0.87	1.00	0.87	1.04	0.95	0.91
ECHO	1.02	0.73	0.71	1.07	0.68	0.64	0.97	0.77	0.80
Fugue	0.81	0.41	0.50	0.87	0.39	0.45	0.77	0.42	0.54
Groestl	1.87	1.14	0.61	1.74	0.94	0.54	1.98	1.32	0.66
Hamsi	2.40	0.69	0.29	2.37	0.75	0.31	2.41	0.65	0.27
JH	1.03	1.00	0.96	1.04	1.00	0.97	1.03	0.99	0.96
Keccak	0.89	0.54	0.60	0.95	0.56	0.59	0.85	0.52	0.61
Luffa	2.08	0.94	0.45	1.92	0.87	0.45	2.21	1.00	0.46
Shabal	1.02	1.02	1.01	1.02	1.00	0.99	1.01	1.04	1.02
SHAvite-3	2.07	1.19	0.58	1.91	1.28	0.67	2.20	1.13	0.51
SIMD	2.11	1.86	0.88	2.08	1.86	0.89	2.12	1.85	0.88
Skein	1.73	1.90	1.10	1.75	1.80	1.03	1.73	1.98	1.15
SHA-2	1.67	1.58	0.95	1.36	1.59	1.16	1.72	1.58	0.91

From the point of view of the throughput to area ratio, Groups 1 and 2 are the best, followed by Groups 3, 4, and 5, and ending with the Group 6, with the worst trend. Among the Groups 1 and 2, belonging to the Group 2 is less desirable, especially for the algorithms that already take significant area for a 256-bit variant, such as BMW and SIMD.

In Tables 3 and 4, we report the actual performance measures of the 512-bit and the 256-bit variants of all investigated algorithms, for the case of Xilinx Virtex 5 and Altera Stratix III, respectively.

In Table 5, we report ratios of all three major performance measures (Throughput, Area, and Throughput to Area Ratio) for a 512-bit variant vs. a 256-bit variant, averaged (using geometric mean) over

- all seven FPGA families,
- three Xilinx families (Spartan 3, Virtex 4 and Virtex 5), and
- four Altera families (Cyclone II and III, Stratix II and III).

Based on this table, there seems to be a pretty good agreement between the values of respected ratios for Xilinx and Altera families, with the largest discrepancies seen in case of the Throughput of BMW and Groestl, and the Area of Luffa and SHA-2.

The comparison of the rough approximations for the ratios of Throughputs and Areas based on Equations (6) and (8) (see the last two columns of Table 2) with the actual values of these ratios averaged over seven families of FPGAs (see Table 5, Overall, Thr and Area columns) reveals a very good agreement between our predictions and experimental results. A few algorithms, for which the ratios are substantially different are listed below together with the short explanation:

Hamsi: the area ratio is larger than expected (2.40 vs. 2) and the throughput ratio smaller than expected (0.69 vs. 1). Both effects seem to be caused by our implementation of the message expansion unit, which is based on look-up tables. The total size of the look-up tables for the 512-bit variant is four times bigger than for the 256-bit variant (1 Mbit vs. 256 kbit). Additionally, all table look-ups in the 256-bit version can be performed in parallel, while in the 512-bit variant, two groups of the table look-ups need to be performed sequentially, one by one, because of the data dependency.

Keccak: the area ratio is smaller than predicted (0.89 vs. 1). This effect is caused by the smaller value of the block size for the 512-bit variant of the algorithm (576 bits vs. 1088 bits). This value affects only the size of the input shift register, and has no influence on the size of the datapath. It should be noted that the size of the output shift register increases in the 512-bit variant (512-bits vs. 256-bits) but this increase is smaller than the decrease in the size of the input register.

Luffa: the area ratio is larger than predicted (2.08 vs. 1.67), and the throughput ratio smaller than predicted (0.94 vs. 1). This effect can be explained by the more complex computations performed in the 512-bit variant of Luffa during the Message Injection phase. In particular, the $GF(2^8)$ constants used as inputs in Galois Field multiplications, change from small values of {1, 2, 3, 4} to the larger values including {01, 02, 04, 08, 10, 0A, 0F}.

6. Results

In Table 6, the maximum clock frequencies are listed for each pair: hash algorithm – FPGA family. These frequencies can be used together with the formulas provided in Table 1, in order to compute the exact execution times of each algorithm (depending on the number of message blocks, N) and the values of the throughputs for long messages. The clock period (in microseconds), T, is a direct inverse of the clock frequency, f, in MHz. Thus, in the formulas from Table 1, we can replace directly $1/T$ by f, and we will obtain the Throughput in Mbits/s.

Table 6. Clock frequencies of all SHA-3 candidates (512-bit variants) and SHA-512 expressed in MHz (post placing and routing)

Candidate	Spartan 3	Virtex 4	Virtex 5	Cyclone II	Cyclone III	Stratix II	Stratix III
BLAKE	36.59	71.26	106.01	41.57	50.24	73.78	93.41
BMW	N/A	6.03	8.45	N/A	N/A	N/A	7.44
CubeHash	90.84	188.89	215.33	113.43	129.20	164.69	218.05
ECHO	85.17	190.30	200.97	N/A	135.24	166.64	246.00
Fugue	64.25	122.84	138.49	86.61	100.74	142.05	206.27
Groestl	66.18	202.76	180.15	124.10	133.96	187.72	250.38
Hamsi	69.00	158.05	171.38	103.31	117.16	128.68	181.16
JH	130.12	277.32	275.48	173.94	221.93	267.52	358.94
Keccak	94.12	208.86	276.86	161.39	173.07	207.68	269.61
Luffa	93.41	210.88	220.12	143.53	172.98	192.49	268.02
Shabal	29.87	113.62	135.30	69.38	81.70	103.58	126.44
SHAvite-3	75.31	161.97	213.45	86.71	103.73	140.53	215.38
SIMD	N/A	28.57	36.37	20.09	23.87	32.36	43.38
Skein	36.93	81.20	104.34	47.06	54.73	70.64	92.10
SHA-512	90.06	168.75	215.84	93.54	113.15	177.34	234.80

Table 7. Results for the reference implementation of SHA-512 (architecture with rescheduling [24, 25])

	Spartan 3	Virtex 4	Virtex 5	Cyclone II	Cyclone III	Stratix II	Stratix III
Max. Clk Freq. [MHz]	90.06	168.75	215.84	93.54	113.15	177.34	234.80
Throughput [Mbit/s]	1138.51	2133.31	2728.68	1182.53	1430.44	2241.93	2968.34
Area	1367	1403	646	2916	2915	1639	1620
Throughput to Area Ratio	0.83	1.52	4.22	0.41	0.49	1.37	1.83

For several algorithms, implementing (placing & routing) the 512-bit variant was not possible for low cost FPGA families, such as Spartan 3 from Xilinx and Cyclone II and Cyclone III from Altera. These cases are denoted by "N/A" in Table 6 and in subsequent Tables 8-10. BMW-512 is a special case in the sense that we have been able to properly route this circuit for only three out of seven investigated FPGA families, namely for Virtex 4, Virtex 5, and Stratix III. For all remaining families, routing was not possible, despite the fact that the tested FPGA devices contained more than sufficient number of logic resources. This is certainly one of the major drawbacks of BMW, which is also relatively inflexible in terms of trading speed for area.

In Table 7, we summarize the absolute results obtained for our implementation of the current standard SHA-512. The results are repeated for all seven FPGA families used in our study. As hardware architecture, we have selected the architecture by Chaves et al., presented at CHES 2006 [24]. This architecture has been specifically optimized for the maximum throughput to area ratio [24, 25] and is considered one of the best known SHA-2 architectures of this type.

In the following analysis, the absolute values of the three major performance measures: throughput, area, and the throughput to area ratio, for the 512-bit variants of all SHA-3 candidates, have been normalized by dividing them by the corresponding values for the reference implementation of SHA-512. The corresponding ratios, referred to as normalized throughput, normalized area, and normalized throughput to area ratio are summarized in Tables 8, 9, and 10. In all these tables, the Overall column represents the geometric mean of all normalized results, averaged over all seven investigated FPGA

families. The candidate algorithms are ranked based on the value of this Overall metric, representing the performance for a wide range of different FPGA families.

In Table 8, the normalized throughputs are reported. Only five candidates, Luffa, Groestl, BMW, ECHO, and Keccak, outperform SHA-512 by a factor larger then two. The additional five candidates have a normalized throughput in the range from 1 to 2. Four candidates, Skein, Shabal, Hamsi, and Fugue, are slower than SHA-512, with Fugue, slower by a factor of two.

In Table 9, the normalized areas are reported. Based on this table, all SHA-3 candidates, in their 512-bit variants, are larger than SHA-512. The spread of results is much larger than in the case of the throughput, with the smallest SHA-3 candidate, CubeHash, almost the same size as SHA-512, and the largest SIMD, lagging behind by a factor of 26. The group following CubeHash in terms of area, including Fugue, Keccak, Shabal, JH and Skein, covers the range between 2.0 and 2.3, and includes only one candidate, Keccak, which excels also in terms of speed.

Table 8. Throughput of all SHA-3 candidates (512-bit variants) normalized to the throughput of SHA-512

Candidate	Spartan 3	Virtex 4	Virtex 5	Cyclone II	Cyclone III	Stratix II	Stratix III	Overall
Luffa	2.63	3.16	2.58	3.88	3.87	2.75	2.89	3.07
Groestl	2.05	3.36	2.33	3.71	3.31	2.96	2.98	2.90
BMW	N/A	2.90	3.17	N/A	N/A	N/A	2.57	2.87
ECHO	2.39	2.85	2.36	0.00	3.03	2.38	2.65	2.60
Keccak	1.98	2.35	2.44	3.28	2.90	2.22	2.18	2.45
JH	1.63	1.85	1.44	2.09	2.21	1.70	1.72	1.79
SIMD	N/A	1.52	1.52	1.93	1.90	1.64	1.66	1.69
SHAvite-3	1.19	1.36	1.41	1.32	1.30	1.13	1.30	1.28
CubeHash	1.28	1.40	1.29	1.53	1.45	1.18	1.18	1.32
BLAKE	1.13	1.18	1.37	1.24	1.24	1.16	1.11	1.21
Skein	0.87	1.03	1.03	1.07	1.03	0.85	0.84	0.96
Shabal	0.54	1.09	1.02	1.20	1.17	0.95	0.87	0.95
Hamsi	0.65	0.79	0.67	0.93	0.87	0.61	0.65	0.73
Fugue	0.45	0.46	0.41	0.59	0.56	0.51	0.56	0.50

Table 9. Area (utilization of programmable logic blocks) of all SHA-3 candidates (512-bit variants) normalized to the area of SHA-512

Candidate	Spartan 3	Virtex 4	Virtex 5	Cyclone II	Cyclone III	Stratix II	Stratix III	Overall
CubeHash	1.28	1.24	1.18	1.16	1.17	1.18	1.19	1.20
Fugue	2.22	2.16	1.43	2.54	2.52	1.72	1.71	2.00
Keccak	2.25	2.17	1.91	1.81	1.81	2.19	2.21	2.04
Shabal	2.28	2.25	2.12	2.10	2.11	2.29	2.32	2.21
JH	2.81	2.74	1.80	2.41	2.37	1.97	1.99	2.27
Skein	2.51	1.32	2.35	2.28	2.36	2.79	2.82	2.29
BLAKE	5.86	5.44	5.07	2.52	2.44	2.12	2.11	3.02
Hamsi	3.19	3.10	3.41	2.61	2.61	3.50	3.50	3.11
Luffa	3.92	3.82	3.35	3.60	3.63	4.28	4.25	3.82
SHAvite-3	5.85	6.09	3.02	7.01	7.01	3.36	3.46	4.83
Groestl	14.35	18.57	5.37	5.08	5.16	3.71	3.88	6.59
BMW	N/A	13.50	16.10	N/A	N/A	N/A	15.57	15.01
ECHO	19.56	18.43	9.22	N/A	23.89	12.26	12.40	15.15
SIMD	N/A	28.29	26.34	22.09	22.13	29.15	29.43	26.05

Table 10. Throughput to Area Ratio of all SHA-3 candidates normalized to the throughput to area ratio of SHA-512

Candidate	Spartan 3	Virtex 4	Virtex 5	Cyclone II	Cyclone III	Stratix II	Stratix III	Overall
Keccak	0.88	1.09	1.27	1.81	1.60	1.02	0.99	1.20
CubeHash	1.00	1.13	1.09	1.33	1.24	1.00	0.99	1.10
Luffa	0.67	0.83	0.77	1.08	1.07	0.64	0.68	0.80
JH	0.58	0.68	0.80	0.87	0.93	0.86	0.86	0.79
Groestl	0.14	0.18	0.43	0.73	0.64	0.80	0.77	0.44
Shabal	0.24	0.48	0.48	0.57	0.55	0.41	0.38	0.43
BLAKE	0.19	0.22	0.27	0.49	0.51	0.55	0.53	0.40
Skein	0.35	0.43	0.44	0.47	0.44	0.30	0.30	0.38
SHAvite-3	0.20	0.22	0.46	0.19	0.19	0.34	0.38	0.27
Fugue	0.20	0.21	0.28	0.23	0.22	0.30	0.32	0.25
Hamsi	0.20	0.25	0.20	0.36	0.34	0.17	0.19	0.24
BMW	N/A	0.21	0.20	N/A	N/A	N/A	0.16	0.19
ECHO	0.12	0.15	0.26	0.00	0.13	0.19	0.21	0.17
SIMD	N/A	0.05	0.06	0.09	0.09	0.06	0.06	0.06

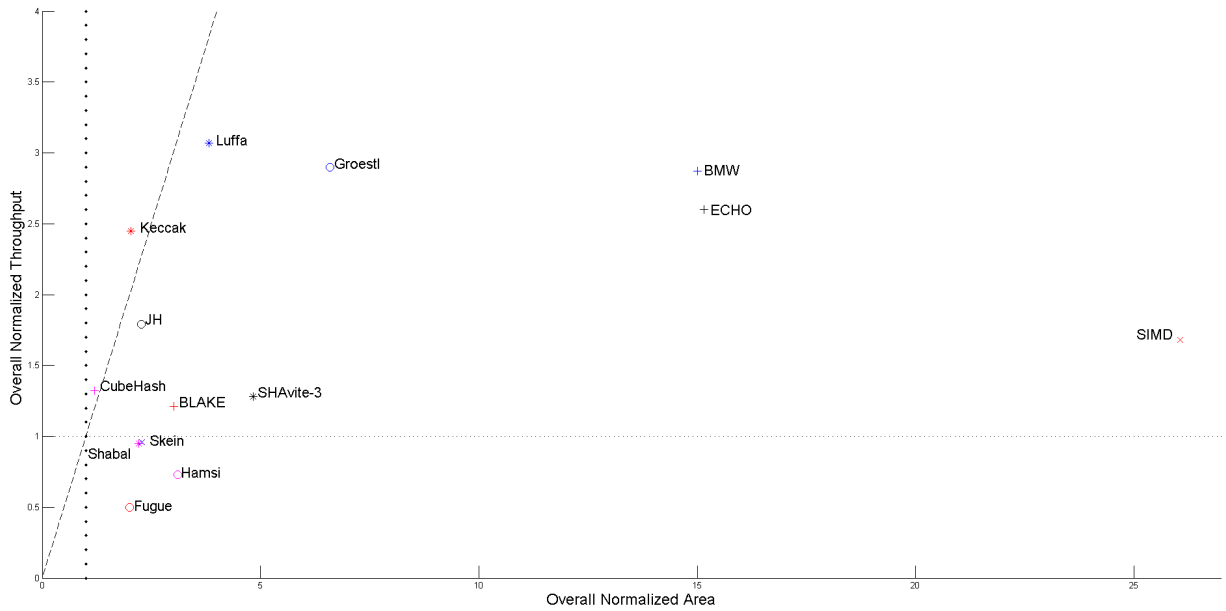

Fig. 1. Relative performance of all Round 2 SHA-3 Candidates (512-bit variants) in terms of the overall normalized throughput and the overall normalized area (with SHA-512 used as a reference point).

In Table 10, the throughput to area ratio is reported. This table is the best considered together with Fig. 1, which presents a two dimensional diagram, with Normalized Area on the X-axis and Normalized Throughput on the Y-axis. Only two algorithms, Keccak and CubeHash, outperform SHA-512 in terms of the throughput to area ratio. Out of them Keccak is almost twice as fast, but CubeHash is almost twice as small. SIMD is approximately 20 times worse than Keccak in terms of the throughput to area ratio, and ECHO and BMW are more than 6 times worse. The implementations of these algorithms are not likely to scale to the same performance region as implementations of majority of other candidates, even if significantly trading speed for reduced area.

7. Conclusions

Our evaluation methodology, applied to 512-bit variants of all 14 Round 2 SHA-3 candidates, has demonstrated large differences among competing candidates. The ratio of the best result to the worst result was equal to about 6 in terms of the throughput (Luffa vs. Fugue), about 23 in terms of area (CubeHash vs. SIMD), and about 20 in terms of our primary optimization target, the throughput to area ratio (Keccak vs. SIMD). Only two candidates, Keccak and CubeHash, have demonstrated the throughput to area ratio better than the current standard SHA-512. Out of these two algorithms, Keccak has also demonstrated very high throughputs, while CubeHash outperformed other candidates in terms of minimum area. Almost all candidates, except Fugue, Hamsi, Shabal, and Skein, outperform SHA-512 in terms of the throughput, but at the same time none of them, except CubeHash, matches SHA-512 in terms of the area.

Future work will include the development of different architectures of SHA-3 candidates, representing various trade-offs between speed and area. The uniform padding units will be added to each SHA core, and their cost estimated. In terms of FPGA families, our study will be extended to the most recent families of FPGAs from two major vendors, namely Spartan 6 and Virtex 6 from Xilinx, and Cyclone IV, Stratix IV, and Arria II from Altera. We will also investigate the influence of synthesis tools from different vendors (e.g., Synplify Pro from Synopsys). The evaluation may be also extended to the cases of hardware architectures optimized for the minimum area (cost) and minimum power consumption. Each algorithm will be also evaluated in terms of its suitability for implementation using dedicated FPGA resources, such embedded memories, dedicated multipliers, and DSP units. Finally, an extension of our methodology to the standard-cell ASIC technology will be investigated.

References

[1] FIPS 180-3, Secure Hash Standard (SHS), October 2008, available at http://csrc.nist.gov/groups/ST/toolkit/secure_hashing.html

[2] NIST's SHA-3 Contest: http://csrc.nist.gov/groups/ST/hash/sha-3/index.html

[3] BlueCrypt, Cryptographic Key Length Recommendation, available at http://www.keylength.com/

[4] Recommendation for Key Management, Special Publication 800-57 Part 1, NIST, 03/2007, available at http://csrc.nist.gov/groups/ST/toolkit/key_management.html

[5] Fact Sheet Suite B Cryptography, NSA, 03/2010, available at http://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml

[6] Yearly Report on Algorithms and Keysizes (2009), D.SPA.7 Rev. 1.0, ICT-2007-216676 ECRYPT II, 07/2009, available at http://www.ecrypt.eu.org/documents/D.SPA.7.pdf

[7] T. Grembowski, R. Lien, K. Gaj, N. Nguyen, P. Bellows, J. Flidr, T. Lehman, B. Schott, "Comparative Analysis of the Hardware Implementations of Hash Functions SHA-1 and SHA-512," LNCS 2433, Information Security, Eds. G. I. Davida, Y. Frankel, 5th International Conference, ISC 2002, Sao Paulo, Brazil, Sep./Oct. 2002, pp. 75-89.

[8] SHA-3 Zoo : http://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo

[9] SHA-3 Hardware Implementations: http://ehash.iaik.tugraz.at/wiki/SHA-3_Hardware_Implementations

[10] K. Kobayashi, et al., “Evaluation of hardware performance for the SHA-3 candidates using SASEBO-GII.

Cryptology ePrint Archive, Report 2010/010, 2010. http://eprint.iacr.org/.
[11] K. Gaj, E. Homsirikamol, and M. Rogawski, "Fair and Comprehensive Methodology for Comparing Hardware Performance of Fourteen Round Two SHA-3 Candidates using FPGAs," Proc. Cryptographic Hardware and Embedded Systems workshop, CHES 2010, Santa Barbara, Aug. 2010 (in print).
[12] S. Tillich, et al. "High-speed hardware implementations of Blake, Blue Midnight Wish, Cubehash, ECHO, Fugue, Grøstl, Hamsi, JH, Keccak, Luffa, Shabal, Shavite-3, SIMD, and Skein. Cryptology ePrint Archive, Report 2009/510, 2009. http://eprint.iacr.org/.
[13] L. Henzen, P. Gendotti, P. Guillet, E. Pargaetzi, M. Zoller and F.K. Gurkaynak, "Developing a Hardware Evaluation Method for SHA-3 Candidates," Proc. Cryptographic Hardware and Embedded Systems workshop, CHES 2010, Santa Barbara, Aug. 2010 (in print).
[14] B. Baldwin, et al., "FPGA implementations of SHA-3-3 candidates: Cubehash, Grøstl, Lane, Shabal and Spectral Hash. Cryptology, ePrint Archive, Report 2009/342, 2009. http://eprint.iacr.org/.
[15] I. Verbauwhede and M. Kneevic, "Hardware evaluation of the Luffa hash family. COSIC Publication, Online, 2009. http://www.cosic.esat.kuleuven.be/publications/article-1282.pdf.
[16] J. Apfelbeck, B. Jungk, S. Reith, "On optimized FPGA implementations of the SHA-3 candidate Grøstl. Cryptology ePrint Archive, Report 2009/206, 2009. http://eprint.iacr.org/.
[17] Joachim Strombergson, "Implementation of the Keccak hash function in FPGA devices," Online, 2008. http://www.strombergson.com/files/.
[18] M. Bernet, et al., "Hardware implementations of the SHA-3 candidates Shabal and Cubehash," Circuits and Systems, pp. 515-518, 2009. 52nd IEEE International Midwest Symposium.
[19] M. Long, "Implementing Skein hash function on Xilinx Virtex-5 FPGA platform. Online, 2010. http://www.skein-hash.info/sites/default/files/.
[20] A.H. Namin and M.A. Hasan. Hardware implementation of the compression function for selected SHA-3 candidates. In CACR 2009-28, page 29, July 2009.
[21] "Hardware Interface of a Secure Hash Algorithm (SHA)," by CERG Team, George Mason University, 2010, available at http://cryptography.gmu.edu/athena/index.php?id=interfaces
[22] "ATHENa Project Website," http://cryptography.gmu.edu/athena/.
[23] K. Gaj, J.P. Kaps, V. Amirineni, M. Rogawski, E. Homsirikamol, B.Y. Brewster, "ATHENa – Automated Tool for Hardware EvaluatioN: Toward Fair and Comprehensive Benchmarking of Cryptographic Hardware using FPGAs," 20th International Conference on Field Programmable Logic and Applications, Milano, Italy, Aug. 31st - Sep. 2nd, 2010.
[24] R. Chaves, G. Kuzmanov, L. Sousa and S.Vassiliadis, "Improving SHA-2 hardware implementations, In Workshop on Cryptographic Hardware and Embedded Systems, CHES 2006.
[25] R. Chaves, G. Kuzmanov, L. Sousa, and S. Vassiliadis, "Cost efficient SHA hardware accelerators," in IEEE Transactions on Very Large Scale Integration Systems, Aug 2008, pp. 999–1008.