Comprehensive Evaluation of High-Speed and Medium-Speed Implementations of Five SHA-3 Finalists Using Xilinx and Altera FPGAs

Kris Gaj, Ekawat Homsirikamol, Marcin Rogawski, Rabia Shahid, and Malik Umar Sharif
George Mason University

{kgaj, ehomsiri, mrogawsk, rshahid, msharif2}@gmu.edu

Abstract. In this paper we present a comprehensive comparison of all Round 3 SHA-3 candidates and the current standard SHA-2 from the point of view of hardware performance in modern FPGAs. Each algorithm is implemented using multiple architectures based on the concepts of iteration, folding, unrolling, pipelining, and circuit replication. Trade-offs between speed and area are investigated, and the best architecture from the point of view of the throughput to area ratio is identified. Finally, all algorithms are ranked based on their overall performance, and the characteristic features of each algorithm important from the point of view of its implementation in hardware are identified.

Keywords: benchmarking, hash functions, SHA-3, hardware, FPGA.

1. Introduction

Performance in hardware is one of the major criteria used in the SHA-3 competition [17]. Typically, this performance is evaluated using two major technologies: Field Programmable Gate Arrays (FPGAs) and Application Specific Integrated Circuits (ASICs). Comparison using FPGAs offers several important advantages, such as short development time, accurate post place & route results, existence of tools for optimum choice of program options and automated collection of a large number of results [1], and relatively small number of vendors and device families that dominate the market. As a result, our FPGA performance evaluation covers significantly broader design space than any ASIC comparison we are aware of. In particular, in this paper, each of the SHA-3 finalists is implemented in both basic *variants, with a 256-bit and a 512-bit output*, and each variant is implemented using from *5 to 10 different hardware architectures* based on the concepts of iteration, folding, unrolling, pipelining, and circuit replication. Each architecture is equipped with a *realistic FIFO-based interface* with a modest pin requirement, and the capability for simultaneous processing of the current message block, reading the next message block, and writing the previously computed hash value to the output FIFO [4,10,11]. Unlike any ASIC implementations, and majority of earlier reported FPGA implementations, our SHA-3 candidate cores are equipped with full *padding units*, capable of processing any messages ending on a boundary of a byte. All VHDL source codes have been developed by *two primary designers*, closely collaborating with each other, which substantially minimizes the potential influence of different designer skills. Majority of *source codes and the corresponding block diagrams* have already been published on the web and made available for public scrutiny [1]. The remaining source codes will be made publicly available by the time of the conference. All cores have been implemented and characterized using *four modern high-performance FPGA families* from two major vendors, Xilinx and Altera. All implementation results have been optimized and generated using ATHENa (Automated Tool for Hardware Evaluation) [1]. The details of all 600+ results are available in the ATHENa database [1], where they can be interactively accessed, reviewed, ranked, searched for, and compared to one another. For each set of results, ATHENa database holds also a set of replication scripts and configuration files that can be used by a third party to efficiently reproduce all results without using ATHENa. Finally, we also demonstrate in this paper that selected FPGA results show very good correlation with the corresponding ASIC results obtained using a typical standard-cell library based on the similar 65nm CMOS technology.

2. Previous work

Previous results on comparison of Round 2 SHA-3 candidates in hardware are summarized in [18]. These results are classified into four major categories, based on the technology (FPGA vs. ASIC), and the optimization target (High-Speed vs. Low-Area). The previous results most relevant to the subject of this paper belong to the category of High-Speed Implementations in FPGAs. The most comprehensive results belonging to this category were reported by Baldwin et al. [3], Gaj et al. [4], Homsirikamol et al. [10], and Matsuo et al. [15]. All these groups have published results for all 14 Round 2 candidates. Majority of published results concern 256-bit variants of the candidates, implemented using Xilinx Virtex 5 FPGAs. In [10], results for 256-bit and 512-bit variants of all algorithms, implemented using 10 FPGA families from Xilinx and Altera are discussed. Additionally, pipelined implementations of three Round 2 SHA-3 candidates have been investigated in [2]. In our earlier paper, published at CHES 2011 [11], we investigated the throughput vs. area trade-offs in implementations of SHA-2 and five SHA-3 finalists. In this paper, we present results obtained by extending each architecture with a padding unit, and optimizing selected pipelined implementations of the SHA-3 candidates.

Three comprehensive comparisons of low-area implementations of Round 3 SHA-3 candidates have been presented in [12,13,14]. The most comprehensive studies of ASIC implementations of the Round 3 SHA-3 candidates are presented in [7,8]. These studies follow previous investigation of Round 2 SHA-3 candidates described in [6,9,20].

All results obtained based on the Round 2 specifications of SHA-3 candidates carry without any changes for Keccak and Skein. The specifications of BLAKE, Groestl, and JH have been tweaked at the start of Round 3, in January 2011. The throughput of the Round 3 BLAKE and JH can be calculated based on the results from Round 2 by decreasing it by a factor proportional to the increase in the number of rounds. The area of these implementations will remain practically the same. The change in the throughput and area of Groestl is much more difficult to approximate, as demonstrated in [16].

3. Performance Metrics

Three major performance metrics used in our study are throughput, area, and throughput to area ratio. Throughput is understood as the throughput for long messages, or cumulative throughput for a large number of small messages (where processing and input/output functions overlap in time). Such defined throughput does not take into account the time taken for reading the very first block of the first message, message initialization, message finalization, and writing the last hash value to the output memory.
The resource utilization in FPGAs is a vector, with coordinates specific to the given FPGA family, e.g.

$$\textit{Resource Utilization}_{\text{Virtex 5}} = (\#\textit{CLB slices}, \#\textit{BRAMs}, \#\textit{DSPs}) \quad (1)$$

$$\textit{Resource Utilization}_{\text{Stratix III}} = (\#\textit{ALUTs}, \#\textit{memory\_bits}, \#\textit{DSPs}). \quad (2)$$

In these formulas: *#CLB_slices* is the number of Configurable Logic Block slices, BRAM stands for Block RAM, DSP is a Digital Signal Processing unit, #ALUTs represents the number of Adaptive Look-Up Tables, and *#mem-bits* is the number of bits stored in dedicated Altera FPGA memories.

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 *#CLB_slices* for Xilinx Virtex 5 and Virtex 6, *#ALUTs* in Stratix III and Stratix IV.

We believe that the capability of using embedded resources should be treated as a measure of the algorithm flexibility, and should be investigated independently from this study. This issue is discussed in more detail in Section 7.

4. Investigated Hardware Architectures

Investigated architectures are described in more detail in our earlier paper presented at CHES 2011. Additionally, full VHDL source codes and corresponding hierarchical block diagrams of majority of these architectures have been published at [1]. Below, we present only a short summary of major features of known to us high-speed and medium-speed hardware architectures of SHA-3 finalists.

A starting point for our exploration is the basic iterative architecture, shown in Fig. 1a. This architecture is the most efficient (in terms of the throughput to area ratio) non-pipelined architecture of SHA-2, Groestl, JH, and Keccak.

In order to reduce area necessary to implement a given hash algorithm, at the cost of decreasing its throughput, folded architectures can be used. These architectures can be employed only if a round of a hash function has a symmetric structure with respect to either horizontal or vertical axis (with input to a round shown at the top and output shown at the bottom of the round block), as illustrated in Fig. 1.

In Fig. 1b, horizontal folding by a factor of two is demonstrated. We will denote this architecture by /2(h). In this architecture, a half of a round is implemented as combinational logic, and the entire round is executed using two clock cycles. As a result, the block processing time (and thus also throughput) stays approximately the same, and area decreases. These dependencies lead to the overall increase of the Throughput to Area ratio. In general, folding by a factor of k might be possible, and the corresponding architecture will be denoted by /k(h). Among the five finalists, the only candidate that can benefit substantially from horizontal folding is BLAKE. The round of BLAKE consists of two horizontal layers of identical G functions, separated only by a permutation. By implementing only one layer in combinational logic, horizontal folding by a factor of two can be easily achieved. Additionally, each G function has a very symmetric structure along the horizontal axis, and can be easily folded horizontally by a factor of 2. As a result, a folding factor of 4 can be achieved for the entire round. Other SHA-3 finalists do not demonstrate any similar symmetry.

In Fig. 1c, we demonstrate vertical folding by a factor of 2. We will denote this folding by /2(v). In this architecture, the datapath width is reduced by a factor of two. As a result two clock cycles are required to complete a round. In the first clock cycle, only bits of the internal state affecting the first half of the round output are provided to the input of R/2. In the second clock cycle, the remaining bits of the internal state are processed. The first output is stored in an auxiliary register of the size of s/2 bits. This output is concatenated with the output from the second iteration to form a new internal state. The clock period of this architecture is approximately equal to the clock period of the basic iterative architecture. As a result, the block processing time, increases approximately by a factor of two compared to the basic architecture. The area reduction is also smaller than in case of horizontal folding, because of the need for an extra s/2-bit register and a multiplexer. As a result the throughput to area ratio is likely to go down. In general, vertical folding by a factor of k might be possible, and the corresponding architecture will be

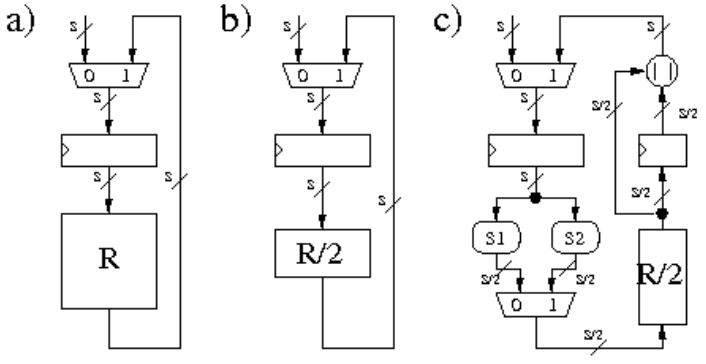

Fig. 1. Three hardware architectures of a hash function: a) basic iterative, x1, b) folded horizontally by a factor of 2, /2(h), c) folded vertically by a factor of 2, /2(v). R – round, S1, S2 – selection functions.

denoted by /k(v). Out of five final SHA-3 candidates, BLAKE and Groestl are most suitable for vertical folding. JH can be folded, but the gain in area is not expected to be substantial, because the round of JH is very simple, and does not dominate the total area of the circuit. For Skein and Keccak, the internal round symmetry, necessary for implementation of vertical folding, is limited.

For vertical folding with the factor $k \geq 4$ it is beneficial to store the internal state in memory, rather than in registers. The obtained throughput to area ratio can be substantially increased as a result of this change in the storage element. We will denote the obtained architectures as /k(v)-m.

In order to increase circuit throughput for processing of a single message, unrolling can be used. In Fig. 2a, architecture with unrolling by a factor of two is demonstrated. The combinational logic of a round is replicated, so now two rounds are performed per clock cycle. Since the total number of clock cycles is reduced approximately by a factor of two, and the clock period increases by a factor less than two (due to optimizations on the boundaries of two rounds, and the smaller relative contributions of the multiplexer delay, the register delay, and the register setup time), the total throughput increases. Unfortunately, at the same time, the area of the circuit is likely to increase by a factor close to the unrolling factor. As a result, in most cases, the throughput to area ratio decreases substantially compared to the basic iterative architecture. As such, architectures with unrolling are typically used only when throughput for a single long message is of the utmost concern, and area is abundant.

Nevertheless, there are exceptions to this rule. Unrolling can improve the throughput to area ratio when rounds used by an algorithm in subsequent iterations are not the same, or there is a potential for substantial delay reductions on the boundary between consecutive rounds. Among the five final SHA-3 finalists, this situation happens only for Skein. As a result, the throughput to area ratio of Skein becomes optimum for one of the unrolled architectures.

Further increase in the throughput and the throughput to area ratio of SHA-3 candidates is possible by using pipelined architectures. In order to take full advantage of the pipelined architectures multiple messages must be processed at the same time. Luckily, this is exactly the situation that appears most often in practical applications of hash functions. For example, in the most widespread Internet security protocols, such as IPSec, SSL, and WLAN (802.11), the inputs to a hash unit are packets. The maximum size of a packet for Internet is limited by so called Maximum Transmission Unit (MTU). The typical size of MTU for Ethernet based networks is 1500 bytes. The Maximum Transmission Unit for the Internet IPv4 path is even smaller, and set at 576 bytes. As a result, in a typical internet node, up to 80% of packets processed have the size of 576 bytes or less, and 100% of packets have sizes equal or smaller than 1500 bytes. Such small sizes of packets mean that hundreds of packets could be easily buffered in the processing nodes, in the form of packet queues, without introducing any significant latency to the total packet travel time from the source to destination. Therefore, the capabilities for parallel processing (including pipelining) seem to be primarily limited by the total area of the hash unit, and not by the number of messages available in parallel. In this paper, we will assume that the number of messages available in parallel is large (at least 10), and we will look at the combined throughput for all available streams of data.

The easiest way to implement pipelining in hash functions is to first unroll, and then introduce pipeline registers between adjacent rounds. The simplest case is the architecture that is two times unrolled, and has two pipeline stages, as shown in Fig. 2b. We will denote this architecture as x2-PPL2. The throughput to area ratio remains roughly the same, and may be either larger or smaller than in the basic iterative architecture, depending on a particular algorithm. The more challenging way of using pipelining is to introduce pipeline registers inside of a hash function round. The improvement in throughput compared to the basic iterative architecture is then equal (either exactly or at least approximately) to the ratio of the new clock frequency to the original clock frequency. Since the critical path is reduced, the increase in throughput is guaranteed, but its level depends on how well the critical path has been divided by pipeline registers into shorter paths with approximately equal delays. At the same time, the area of the circuit increases by the area of pipeline registers, plus any logic required for simultaneous processing of multiple streams of data. The throughput to area ratio may increase, but the improvement is not guaranteed for all algorithms, and all FPGA families, and may be small or

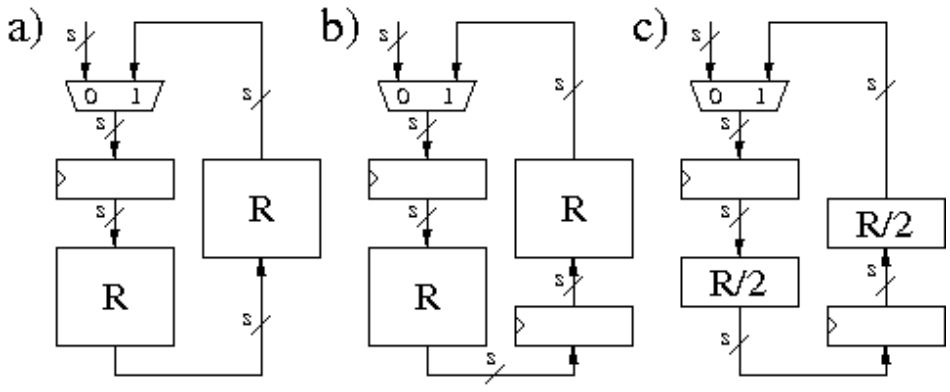

Fig. 2. Three hardware architectures of a hash function a) unrolled by a factor of 2, x2, b) unrolled by a factor of 2 with 2 pipeline stages, x2-PPL2, c) basic iterative with 2 pipeline stages, x1-PPL2.

negative in case the basic iterative architecture operates already at the clock frequency close to the maximum clock frequency supported by a given FPGA family.

The final alternative is architecture obtained by replicating the entire circuit multiple number of times. We call this architecture a multi-unit architecture, and we denote it by MUn, where n denotes the number of repetitions of the hash core. Obviously, in this architecture, throughput and area increase proportionally, and n messages are required to be present concurrently in order to take full advantage of the potential increase in throughput. A typical design approach would be to first find an architecture with the best throughput to area ratio, and then replicate it as many times as necessary in order to reach the desired throughput.

The formulas for the block processing time and the throughput of all aforementioned architectures are summarized in Table 1.

Table 1: Formulas for the time required to process a single message block, T_{block}, and the Throughput, Tp, for all investigated architectures. Notation: b – block size, r – number of rounds, f – number of clock cycles required to finalize computations for a block (f = 0 for Keccak and Groestl (P+Q), f=1 for all remaining algorithms), k – folding factor or unrolling factor, n – number of pipeline stages, T – clock period.

Notation	Architecture	Time required to process a single message block	Throughput
x1	Basic iterative	$T_{block} = (r + f) \cdot T$	$Tp = b/T_{block}$
/k	Folded by a factor of k	$T_{block} = (k \cdot r + f) \cdot T$	$Tp = b/T_{block}$
xk	Unrolled by a factor of k	$T_{block} = (r/k + f) \cdot T$	$Tp = b/T_{block}$
x1-PPLn	Basic iterative with n pipeline stages	$T_{block} = (n \cdot r + f) \cdot T$	$Tp = n \cdot b/T_{block}$
/k-PPLn	Folded by a factor of k with n pipeline stages	$T_{block} = (n \cdot k \cdot r + f) \cdot T$	$Tp = n \cdot b/T_{block}$
xk-PPLn	Unrolled by a factor of k with n pipeline stages	$T_{block} = (n \cdot r/k + f) \cdot T$	$Tp = n \cdot b/T_{block}$
MUn	Multi-unit architecture based on n repetitions of the basic iterative architecture	$T_{block} = (r + f) \cdot T$	$Tp = n \cdot b/T_{block}$

5. Design Methodology and Design Environment

Our designs for the basic, folded, and unrolled architectures use the interface and the communication protocol proposed in [4,10]. Our designs for the pipelined architectures, use the interface and surrounding logic shown in Fig. 3.

Input FIFOs serve as packet queues. Each FIFO communicates with the corresponding Serial-In Parallel-Out (SIPO) unit and the associated Finite State Machine 1 (FSM1). FSM1 is responsible for reading in the next block of data, using b/w clock cycles, possibly in parallel with the Datapath processing the previous block under the control of FSM2. Outputs corresponding to four independent packets are first stored in the corresponding Parallel-In Serial-Out Units, and then multiplexed to the output FIFO.

All architectures have been modeled in VHDL-93. All VHDL codes have been thoroughly verified using a universal testbench, capable of testing an arbitrary hash function core. 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. 13.1, including Xilinx XST,
- for Altera: Quartus II v. 11.1 Subscription Edition Software.

The generation of a large number of results and optimization of tool options was facilitated by an open source benchmarking environment, called ATHENa (Automated Tool for Hardware EvaluatioN) [1].

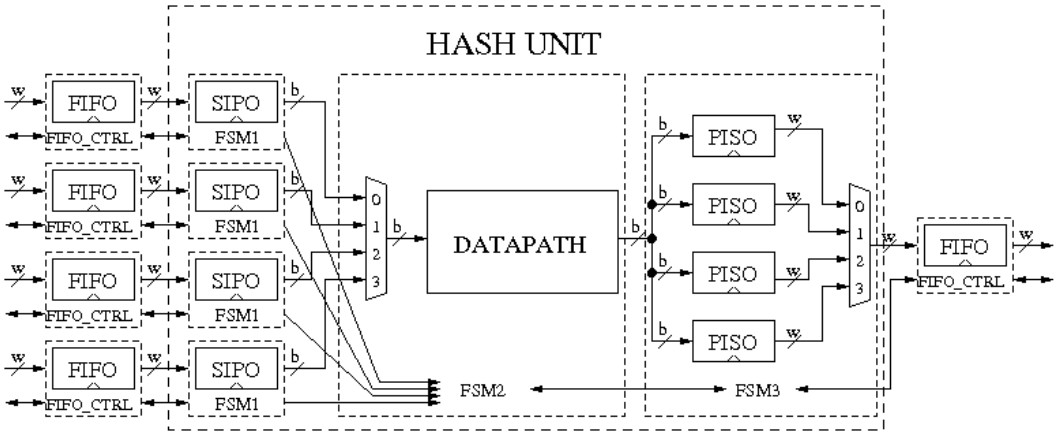

Fig. 3: The interface, high-level block diagram, and surrounding logic of the Hash Unit for the pipelined architecture with four pipeline stages. Notation: SIPO – Serial-In Parallel-Out unit, PISO – Parallel-In Serial-Out unit, w – input/output bus width, w=64 for all investigated algorithms, except SHA-2-256, where w=32.

6. Results

The results of our implementations are summarized in Figs. 4-8, and in Tables 2 and A.1. In Fig. 4, we present the detailed throughput vs. area graphs for all implemented architectures of the 256-bit variants of six investigated algorithms in Xilinx Virtex 5 FPGAs.

For BLAKE (see Fig. 4a), the best non-pipelined architecture in terms of the throughput to area ratio is: /4(h)/4(v)-m, i.e., architecture with horizontal folding by a factor of 4, combined with vertical folding by a factor of 4, and internal state stored in memory. The best architecture overall is x1-PPL4, i.e., basic architecture with four pipeline stages. The good performance of the former of these two architectures is associated with the significant reduction of the complexity of the BLAKE PERMUTE function as a result of vertical folding by 4. The good performance of the latter is associated with the symmetric structure of the round and G function, which makes it easy to divide the datapath into four well-balanced pipeline

stages. The two less successful architectures include x1 and /2(h)-PPL4. These architectures are not included in our combined graphs shown in Figs. 5-8.

For Groestl (see Fig. 4b), we consider two major architectures: a) parallel architecture, denoted (P+Q), in which Groestl permutations P and Q are implemented using two independent units, working in parallel, and b) quasi-pipeline architecture, denoted (P/Q), in which, the same unit is used to implement both P and Q, and the computations belonging to these two permutations are interleaved [20]. The details of the basic quasi-pipelined architecture of Groestl are described in [20, Section 9] and [10, Section 3.8]. In this study, we apply vertical folding and pipelining to both architectures. The best architecture overall appears to be the parallel architecture (P+Q) in the basic version, with two pipeline stages, x1-PPL2. Vertical folding by 2 provides quite substantial reduction in area, but at the price of a slightly greater reduction in throughput. An attempt to pipeline Groestl using 7 pipeline stages (x1-PPL7), using logic-only implementation of S-boxes, appeared to be rather unsuccessful.

For JH (see Fig. 4c), we consider two major types of architectures: a) with round constants stored in memory, JH (MEM), and b) with round constants calculated on the fly, JH (OTF). Both approaches seem to result in a very similar performance for the basic iterative architectures, x1. Neither vertical folding nor pipelining seem to be efficient when applied directly to the basic architecture. Vertical folding by two, somewhat unexpectedly, increases area, and the basic architecture with two pipeline stages does not improve throughput. Both undesired effects can be tracked back to the simplicity of the main round. Folding does not reduce area, because of extra registers and multiplexers introduced to a very simple round. Pipelining does not increase throughput, because a simple basic round is hard to divide into two well balanced pipeline stages. As a result, the basic iterative architecture remains most efficient in terms of the throughput to area ratio.

For Keccak (see Fig. 4d), only vertical folding by a factor of 8, with internal state stored in memory, leads to substantial reduction in area, at the cost of a significant reduction in throughput. Two pipeline stages increase throughput, but by a factor smaller than the increase in the circuit area.

For Skein (see Fig. 4e), the unrolled by 4 architecture, x4, appears to be significantly more efficient than the basic architecture, x1. At the same time, unrolling by 8 does not give any additional improvement. The best results are obtained by first unrolling basic architecture by a factor of four, and then pipelining the obtained circuit using two pipeline stages. Five pipeline stages have been attempted as well because of an extra addition executed every fourth round, but did not improve the overall throughput to area ratio.

For SHA-2 (see Fig. 4f), none of the discussed techniques applies. The implementation of this function is already small, so reducing area is not necessary. The best way to speed up this function is by using multiple independent units of SHA-2 working in parallel. We denote this architecture by MUn, where n denotes the number of hash units.

The combined graphs for the 256-bit variants and the 512-bit variants of all algorithms, implemented using Xilinx Virtex 5 FPGAs, are presented in Figs. 5 and 6. Individual dots placed in regular intervals on the dashed lines represent multi-unit architectures. Algorithms can be ranked first in terms of the throughput to area ratio of their best architecture, as identified above. This is because this architecture can be easily replicated, allowing for processing n streams of data in parallel. Both throughput and area will increase by a factor of n.

The secondary criterion is the area of the best architecture. The smaller the area, the denser is the graph representing possible locations of a given function on the throughput vs. area graph.

The results for the 256-bit variants of hash functions, shown in Fig. 5, indicate that the order of the SHA-3 candidates in terms of throughput, for implementations using 1500 or more CLB slices is: 1) Keccak, 2) JH, 3) Groestl, 4) Skein, and 5) BLAKE. Keccak and JH clearly outperform SHA-2, while Groestl becomes faster only with more than about 3000 CLB slices. The results for the 512-bit variants of hash functions, shown in Fig. 6, are quite similar, with the exception that, JH performs almost equally well as Keccak (because of the decrease in the Keccak message block size from 1088 to 576 bits), Skein outperforms Groestl, and BLAKE is a distant fifth.

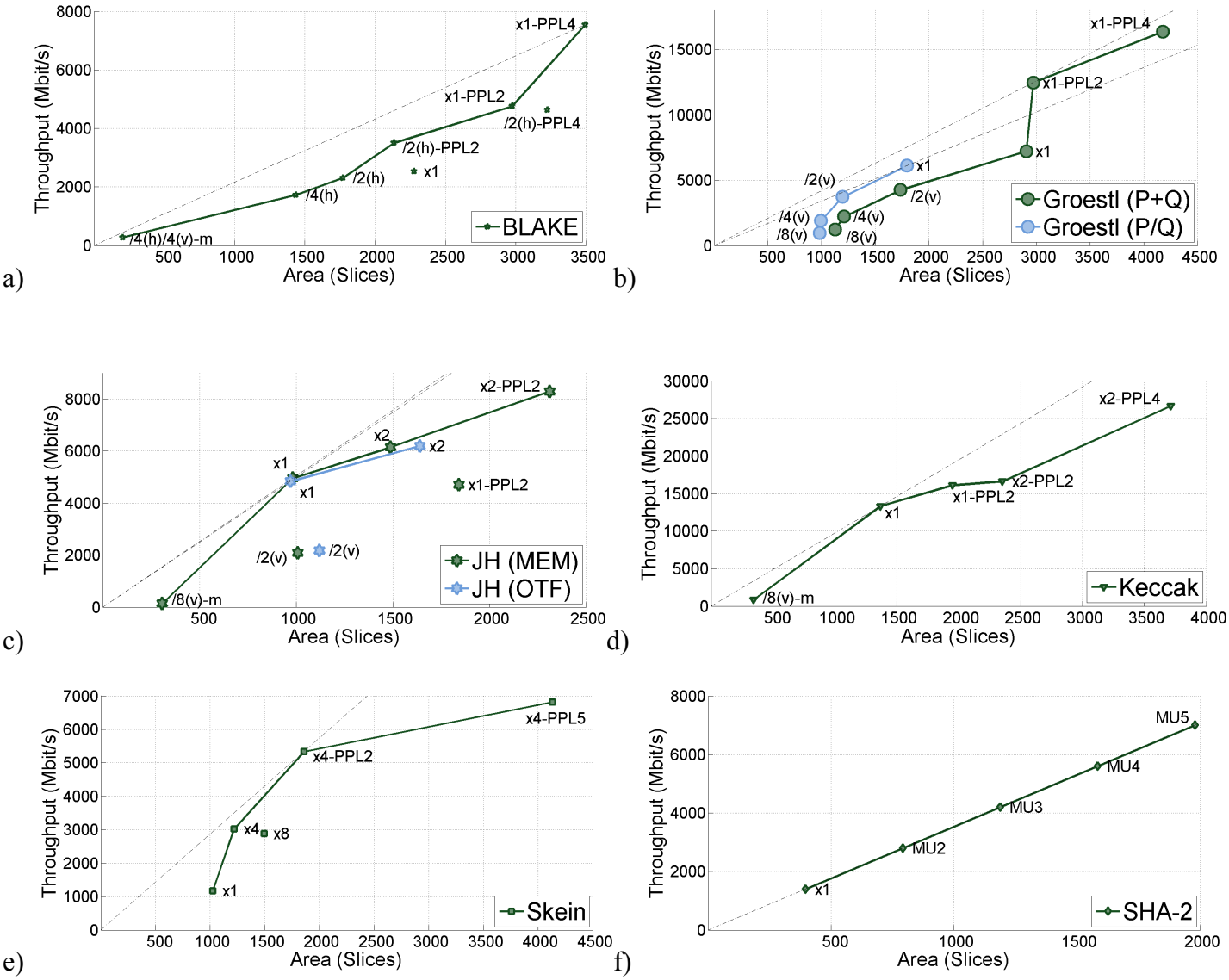

Fig. 4. Throughput vs. Area graphs for multiple architectures of a) BLAKE-256, b) Groestl-256, c) JH-256, d) Keccak-256, e) Skein-256, and f) SHA-256, implemented in Xilinx Virtex 5 FPGAs. Notation: x1 – basic iterative architecture, /k(h) – horizontally folded by a factor of k, /k(v) - vertically folded by a factor of k, /k(v)-m - vertically folded by a factor of k with internal state stored in memory, xk – unrolled by a factor of k, PPLn – pipelined with n pipeline stages, (P+Q) – parallel architecture of Groestl, P/Q – quasi-pipelined architecture of Groestl, MEM – architecture of JH with round constants stored in memory, OTF – architecture of JH with round constants calculated on the fly.

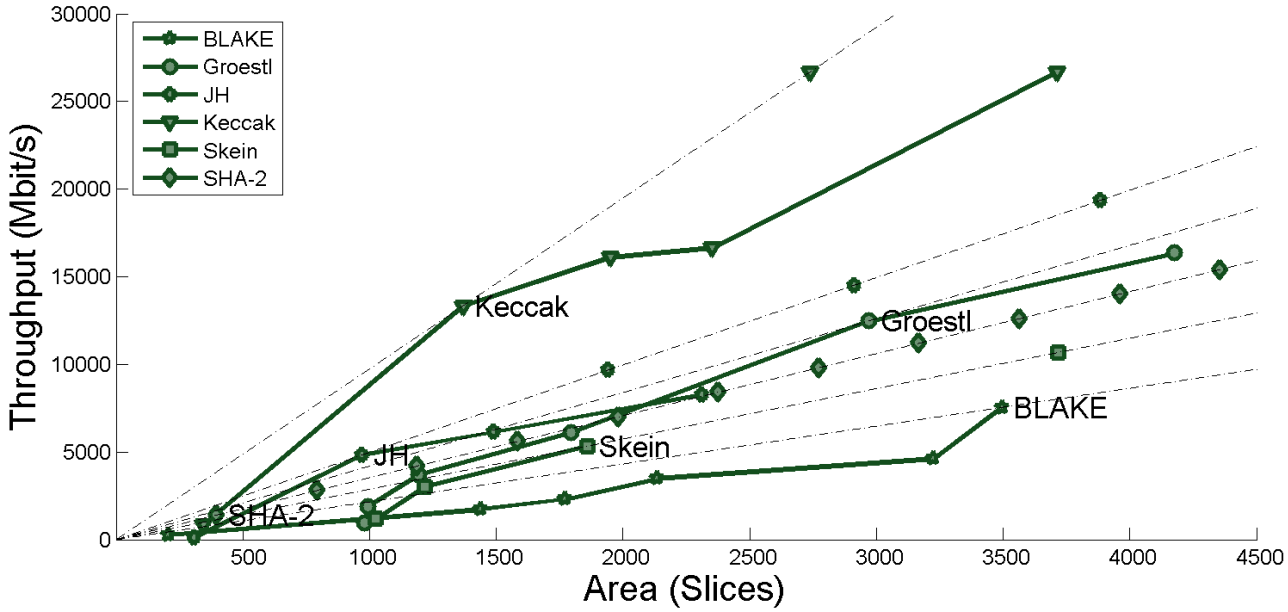

Fig. 5. Combined Throughput vs. Area graph for multiple hardware architectures of the 256-bit variants of BLAKE, Groestl, JH, Keccak, Skein, and SHA-2 implemented in Xilinx Virtex 5 FPGAs.

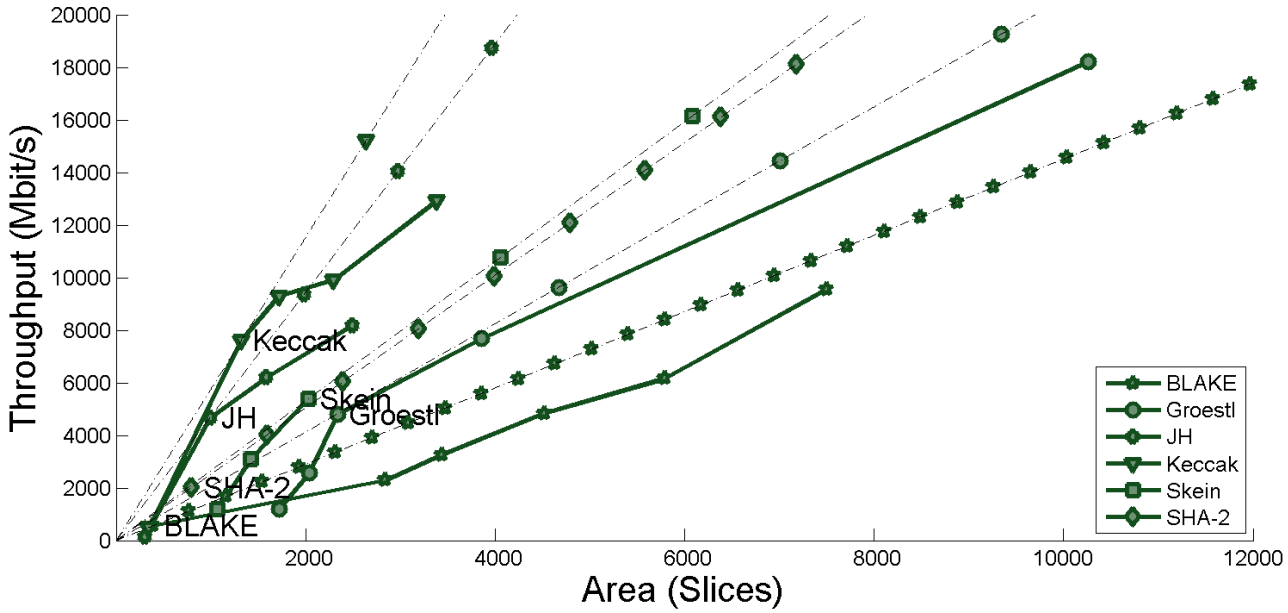

Fig. 6. Combined Throughput vs. Area graph for multiple hardware architectures of the 512-bit variants of BLAKE, Groestl, JH, Keccak, Skein, and SHA-2 implemented in Xilinx Virtex 5 FPGAs.

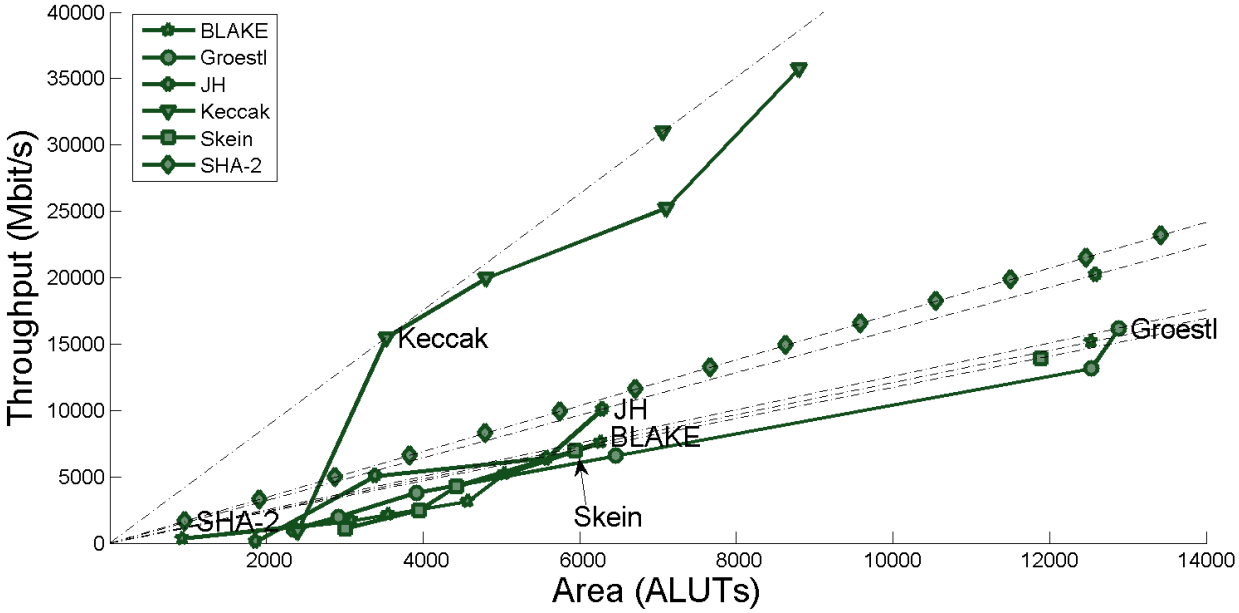

Fig. 7. Combined Throughput vs. Area graph for multiple hardware architectures of the 256-bit variants of BLAKE, Groestl, JH, Keccak, Skein, and SHA-2 implemented in Altera Stratix III FPGAs.

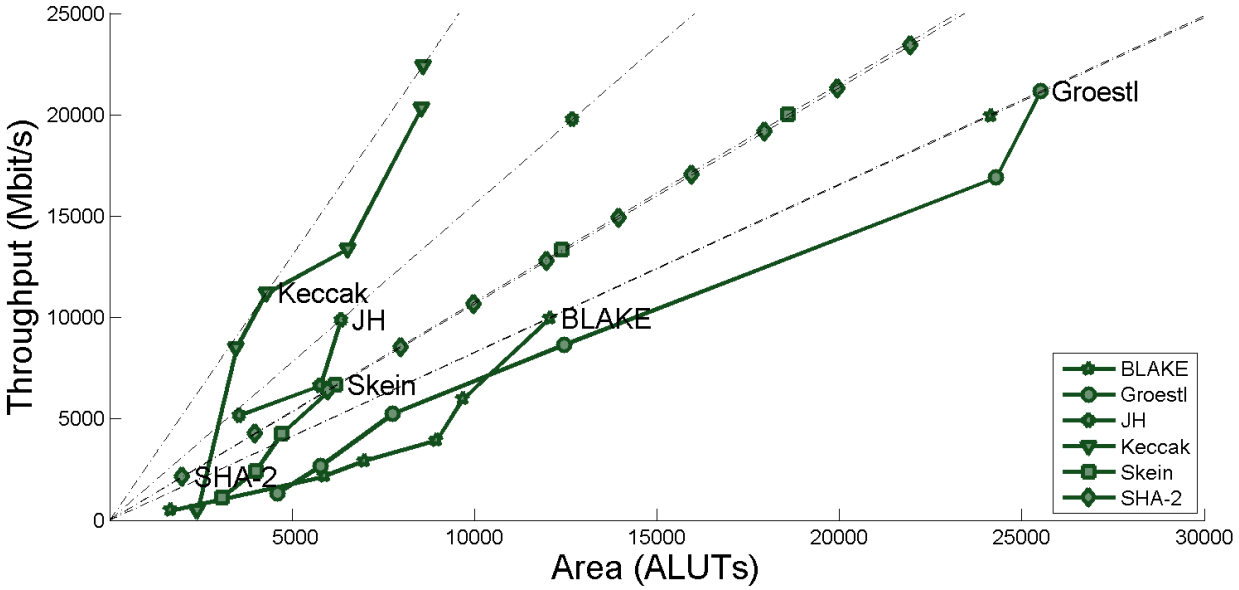

Fig. 8. Combined Throughput vs. Area graph for multiple hardware architectures of the 512-bit variants of BLAKE, Groestl, JH, Keccak, Skein, and SHA-2 implemented in Altera Stratix III FPGAs.

The performance for Altera devices, shown in Figs. 7 and 8, is somewhat different. For the 256-bit versions of the algorithms, Keccak is the only function that outperforms SHA-2 in terms of the throughput to area ratio. JH is second in ranking, with two architectures offering the similar ratio as SHA-2. BLAKE, Groestl, and Skein are in tie with each, with Groestl being somewhat disadvantaged by approximately twice as large area of its most efficient architecture. For the 512-bit versions of the algorithms (see Fig. 8), Keccak and JH outperform SHA-2, Skein is in tie with SHA-2, Groestl and BLAKE fall significantly behind the current standard.

The numerical results for all our implementations are summarized in Tables 2 and A.1. The best values of the throughput to area ratios and the best architectures for each hash function are listed in **bold** in these tables.

Table 2: Results for 256-bit variants of the Round 3 SHA-3 candidates and SHA-2, implemented using all investigated architectures and four FPGA families, Virtex 5 and Virtex 6 from Xilinx, and Stratix III and Stratix IV from Altera. Notation: Tp – throughput, A – area, Tp/A – Throughput to Area Ratio. The best values of the throughput to area ratios and the best architectures for each hash function are listed in **bold**.

Arch	Virtex 5			Virtex 6			Stratix III			Stratix IV		
	Tp	A	Tp/A	Tp	A	Tp/A	Tp	A	Tp/A	Tp	A	Tp/A
BLAKE-256												
/4(h)/4(v)-m	276	204	1.35	399	171	2.34	370	924	0.40	399	935	0.43
/4(h)	1726	1437	1.20	1882	886	2.12	1695	3093	0.55	1735	3085	0.56
/2(h)	2308	1771	1.30	2226	1257	1.77	2157	3553	0.61	2337	3543	0.66
x1	2533	2279	1.11	2416	1711	1.41	2181	4620	0.47	2319	4618	0.50
/2(h)-PPL2	3506	2136	1.64	3178	1630	1.95	3131	4570	0.69	3409	4567	0.75
/2(h)-PPL4	4633	3226	1.44	4807	2407	2.00	5205	5039	1.03	5467	5042	1.08
x1-PPL2	4761	2976	1.60	4768	2111	2.26	4162	5436	0.77	4429	5423	0.82
x1-PPL4	7547	3495	**2.16**	8056	2530	**3.18**	7583	6267	**1.21**	8063	6271	**1.29**
Groestl-256												
/8(v) (P+Q)	1237	1124	1.10	1371	936	1.46	1240	3306	0.38	1173	3288	0.36
/4(v) (P+Q)	2215	1208	1.83	2850	1072	2.66	2576	4528	0.57	2366	4402	0.54
/2(v) (P+Q)	4254	1734	2.45	4850	1548	3.13	5028	7444	0.68	4387	6895	0.64
x1 (P+Q)	7214	2906	2.48	8754	2395	3.65	9572	11193	0.86	8962	10961	0.82
x1-PPL2 (P+Q)	12479	2971	**4.20**	13410	2873	4.67	13166	12531	1.05	12290	12203	1.01
x1-PPL4 (P+Q)	16353	4177	3.91	16213	3597	4.51	16198	12885	**1.26**	16141	12933	**1.25**
/8(v) (P/Q)	951	981	0.97	1057	705	1.50	1009	2346	0.43	976	2342	0.43
/4(v) (P/Q)	1907	993	1.92	2381	859	2.77	1998	2919	0.68	1837	2902	0.63
/2(v) (P/Q)	3721	1195	3.11	4201	898	**4.68**	3818	3914	0.98	3701	3906	0.95
x1 (P/Q)	6117	1795	3.41	7220	1870	3.86	6604	6460	1.02	6269	6421	0.98
JH-256												
/8(v)-m (MEM)	138	306	0.45	157	226	0.69	133	1865	0.07	118	1849	0.06
/2(v) (MEM)	2094	1009	2.08	2327	944	2.46	2131	3379	0.63	2138	3368	0.63
x1 (MEM)	4955	982	**5.05**	5412	849	**6.37**	5276	3221	**1.64**	4759	3210	1.48
x2 (MEM)	6149	1489	4.13	6904	1335	5.17	6418	5584	1.15	6128	5542	1.11
x1-PPL2 (MEM)	4711	1842	2.56	5202	1320	3.94	5463	4263	1.28	5439	4259	1.28
x2-PPL2 (MEM)	8289	2312	3.59	9284	2050	4.53	10116	6294	1.61	10116	6294	**1.61**
x2-PPL4 (MEM)	8526	3085	2.76	9154	2131	4.30	9927	6892	1.44	9994	6883	1.45
/2(v) (OTF)	2181	1120	1.95	1993	845	2.36	2084	3473	0.60	2035	3538	0.58
x1 (OTF)	4840	971	4.98	5255	917	5.73	5071	3388	1.50	4912	3385	1.45
x2 (OTF)	6196	1640	3.78	7046	1493	4.72	6359	6121	1.04	5817	5993	0.97
Keccak-256												
/8(v)-m	870	344	2.53	1183	365	3.24	879	2398	0.37	821	2395	0.34
x1	13337	1369	**9.74**	11839	1086	10.90	15493	3531	**4.39**	16104	3471	4.64
x1-PPL2	16121	1950	8.27	18803	1474	**12.76**	19971	4810	4.15	21184	4294	**4.93**
x2-PPL2	16651	2352	7.08	N/A	N/A	N/A	25283	7107	3.56	25291	6523	3.88
x2-PPL4	26690	3714	7.19	29825	2748	10.85	35780	8806	4.06	38451	8553	4.50
Skein-256												
x1	1179	1025	1.15	1330	858	1.55	1115	3005	0.37	1226	3003	0.41
x4	3023	1218	2.48	3373	1005	3.36	2475	3943	0.63	2592	3936	0.66
x8	2890	1492	1.94	3276	1283	2.55	3161	5432	0.58	3161	5463	0.58
x4-PPL2	5338	1858	**2.87**	6212	1628	**3.82**	4273	4423	0.97	4709	4446	1.06
x4-PPL5	6819	4130	1.65	7669	3126	2.45	6974	5941	**1.17**	7675	5925	**1.30**
x8-PPL10	N/A	N/A	N/A	10978	5844	1.88	11741	11163	1.05	11792	10992	1.07
SHA-256												
x1	1401	396	**3.54**	1634	239	**6.83**	1656	959	**1.73**	1798	959	**1.87**

7. Influence of Assumptions on the Results of Comparison

In order to resolve any possible doubts, regarding the influence of assumptions on ranking of five final SHA-3 candidates, below we investigate the impact of two important assumptions we have made in this study: 1) assumption that all our hash cores include *padding unit*, and b) assumption that *no embedded resources*, such as DSP Units or Block Memories, are used in our FPGA implementations. Both issues have been studied independently by other members of our group [19,21]. Below, we summarize the results of these investigations.

The effect of the padding unit on the performance of selected non-pipelined architectures of 5 Round 3 SHA-3 candidates in four FPGA families has been summarized in Table A.2. Based on this table, the largest decrease in the throughput to area ratio does not exceed 18%. This decrease depends on the FPGA family, and is in the range 0-10% for Virtex 5, 0-14% for Virtex 6, 0-18% for Stratix III, and 4-12% for Stratix IV. These variations do not affect the ranking of candidates as determined in Section 6.

The influence of embedded resources on performance of all 14 Round 2 candidates has been studied in [19]. This study was then narrowed to 5 finalists implemented using the most efficient non-pipelined architecture without padding unit. None of the final candidates can take substantial advantage of DSP units, because none of them uses multiplication. In both Virtex 5 and Stratix III, BLAKE and Groestl can take advantage of Block Memories. BLAKE requires memories for the implementation of the PERMUTE operation, and Groestl for the AES-based operations. The numerical effect of using embedded resources in Xilinx Virtex 5 and Altera Stratix III is summarized in Table A.3. The throughput either decreased, stayed constant, or increased by less than 10% for majority of SHA-3 finalists. The only exception was Groestl implemented using Stratix III, where throughput increased by about 21%. The amount of reconfigurable logic resources stayed within 12% of the original value for JH, Keccak, and Skein. The reduction for Groestl and BLAKE was much more significant, and stayed in the range 41-62%. It should be stressed that this reduction has been accomplished at the expense of substantial usage of embedded memories. For example in Virtex 5, BLAKE occupied 13 out of 64 and Groestl 50 out of 64 18-kbit Block RAMs of the respective FPGA device. The corresponding architectures of BLAKE and Groestl are shown in Figs. A.1 and A.2. The affect on the ratio of throughput to the amount of reconfigurable resources is negligible for JH, Keccak, and Skein, and quite substantial, between 60 and 220% for BLAKE and Groestl. It should be stressed however, that this improvement can be taken advantage of only if in the given system-on-chip including hash functions, none of the other components of the system relies heavily on block memories. If this assumption holds, and we treat the amount of reconfigurable logic resources (#CLB_slices and #ALUTs, respectively) as a sole representation of area, then the ranking of finalists based on the throughput to area ratio changes as shown in Table A.4. As a result of using embedded resources in Virtex 5 and Stratix III FPGAs, Groestl jumps from the third position to the second place ahead of JH, and BLAKE moves from the fifth position to the fourth place ahead of Skein.

8. Correlation between FPGA results and ASIC results

The number of hardware architectures of SHA-3 candidates implemented in ASICs to date is very small compared to the number of architectures implemented in FPGAs. Typically, only the best non-pipelined architectures for the 256-bit variants of the SHA-3 finalists are reported [6,7,9,20]. At the same time, multiple applications use ASICs as a primary way of implementing cryptographic transformations, and this trend is likely to continue in the future. Therefore, it is very interesting to see, whether there exist any strong correlation between results obtained for the ASIC and FPGA implementations of the same architectures. In our experiment, performed in collaboration with the group from ETH Zurich, we have implemented selected architectures for all SHA-3 candidates and SHA-2 using standard-cell CMOS 65nm UMC ASIC technology (UMC65LL) offered through Europractice MPW services, and using a 65 nm high-performance Altera FPGA family Stratix III. The selected architectures included the following non-

pipelined architectures: basic iterative architectures, x1, for Keccak and SHA-2, basic iterative architecture with round constants computed on the fly, x1 (OTF), for JH, basic iterative parallel architecture of Groestl (P+Q), horizontally folded two times architecture of BLAKE, /2(h), and the unrolled 4 times architecture of Skein, x4. All architectures have been designed for the 256-bit variants of the functions, without padding units, and with wide input and output interface (512 or 1088 bits at the input and 256 bits at the output). Exactly the same VHDL source codes have been synthesized, mapped, placed and routed using both technologies. The results, normalized to the results for SHA-2, are presented in Fig. 9. A very good correlation between normalized throughput and normalized area in both technologies have been observed. Ranking in terms of throughput is identical in both technologies. In terms of area the biggest difference is a relatively smaller area of BLAKE in ASIC technology. In Stratix III FPGAs, JH and BLAKE have almost the same area, in ASIC BLAKE is about 20% smaller than JH.

The biggest difference appears in terms of the throughput to area ratio, where BLAKE moves from the 4th position in tie with Skein in Stratix III to the 3rd position, ahead of Groestl for ASIC. Overall correlation is however very good and indicates that evaluations using Altera FPGAs are likely to give similar results to the evaluations using ASICs built using equivalent technology. Interestingly, similar comparison using Xilinx Virtex 5 FPGAs results in much worse correlation.

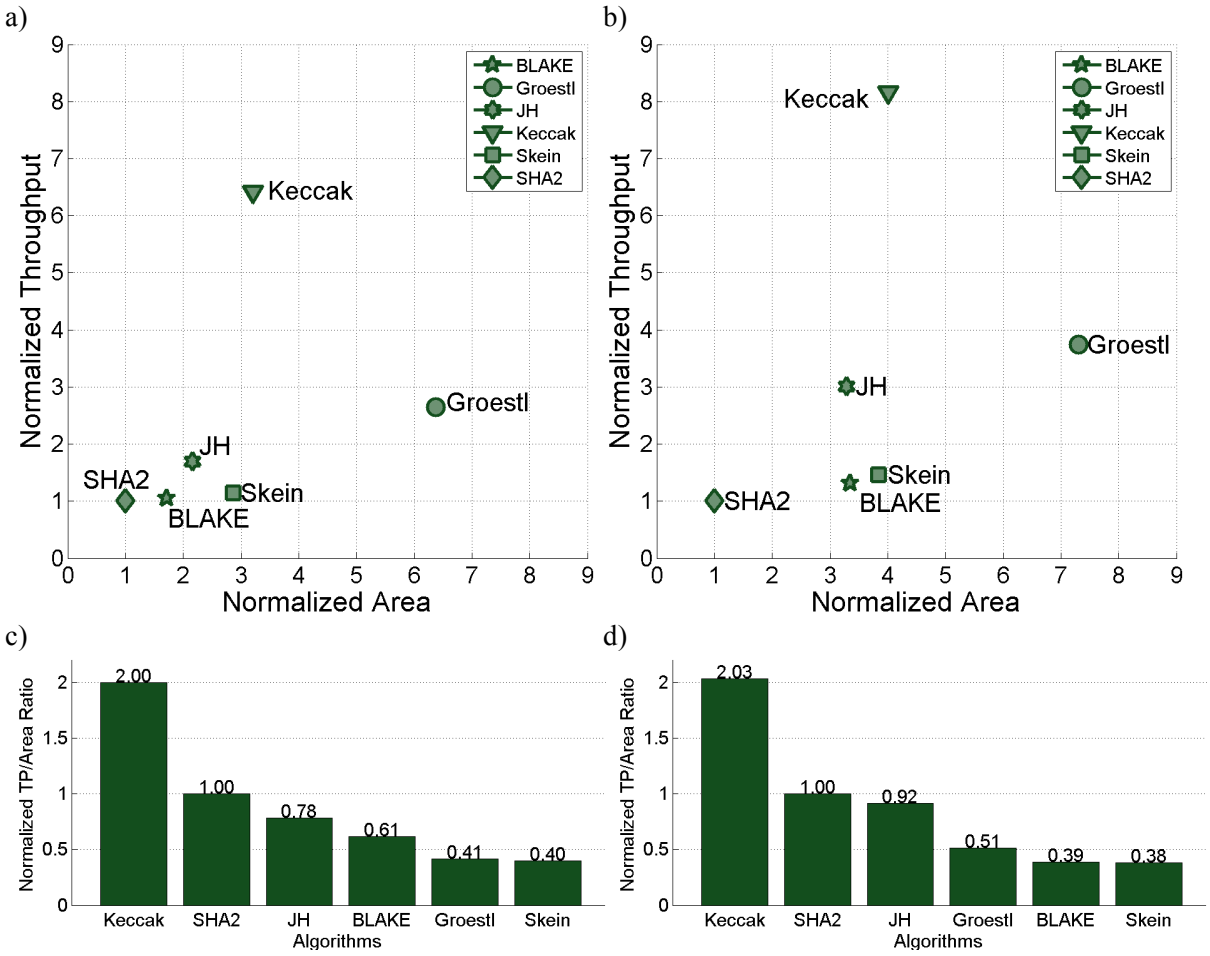

Fig. 9. Correlation between results for 65nm ASIC and 65nm Altera Stratix III FPGA. Normalized throughput vs. normalized area for a) ASIC, b) Stratix III FPGA. Normalized throughput to area ratio for c) ASIC, d) Stratix III FPGA.

9. Conclusions

In this paper, we have performed a systematic investigation of *high-speed* hardware architectures for the five final SHA-3 candidates. The investigated architectures were based on the concepts of the basic iterative architecture, horizontal folding, vertical folding, unrolling, pipelining, and parallel processing using multiple independent units. Each architecture was implemented using four high-performance FPGA families: Virtex 5 and Virtex 6 from Xilinx, and Stratix III and Stratix IV from Altera. Based on the obtained results, we have identified the most efficient hardware architecture for each of the investigated algorithm, based on the best throughput to area ratio.

In case of four out of five candidates (all except JH), the most efficient architecture for at least two out of four FPGA families appeared to be a pipelined architecture. The optimum number of pipeline stages was specific to both the algorithm and FPGA family, as shown in Tables 2 and A.1.

The results for all investigated functions, and the most successful architectures have been summarized using the comprehensive throughput vs. area graphs. These graphs (and the corresponding tables) have revealed that Keccak is the only candidate that consistently outperforms SHA-2 for all considered FPGA families and two hash function variants (with 256-bit and 512-bit output). The only drawback of this function appears to be its limited suitability for folding.

JH performed better than SHA-2 in 3 out of 4 scenarios. It was outperformed by SHA-2 only for the 256-bit function variants implemented using Altera FPGAs. Interestingly, JH is most efficient in its basic iterative architecture, and is not very suitable for either folding or inner-round pipelining.

Groestl outperformed SHA-2 in only one scenario, for the 256-bit variants implemented using Virtex 5. However this advantage was reached only for the relatively large area of about 3000 CLB slices. Although Groestl appeared to be very suitable for vertical folding, the very nature of this technique caused that the decrease in area was accompanied by an even greater decrease in speed. Additionally, Groestl can take advantage of block memories, and can share resources with AES, when both algorithms are implemented on the same chip.

Skein is the only finalist that can substantially benefit from unrolling. It is also the fastest for the pipelined versions of the 4x unrolled architecture, and is the only algorithm that can be pipelined up to 10 times. It performs particularly well compared to other algorithms for the 512-bit variants of hash functions implemented using both Xilinx and Altera FPGAs.

BLAKE is the algorithm with the highest flexibility, and the largest number of potential architectures. It can be easily folded horizontally and vertically by factors of two and four. It can also be easily pipelined even in the folded architectures. It is also the only algorithm that has a relatively efficient architecture that is smaller than the basic iterative architecture of SHA-2. Finally, BLAKE, similarly to Groestl, can benefit substantially from using embedded block memories of both Xilinx and Altera FPGAs.

Our future work will include experimental testing of all developed high-speed architectures of the SHA-3 finalists, using high-performance FPGA boards based on Xilinx and Altera FPGAs, equipped with high-speed communication interface, such as PCI Express.

Acknowledgments

The authors would like to thank Frank Gürkaynak and other members of the Microelectronics Designs Center at ETH Zurich for providing us with the post-layout results of ASIC implementations of GMU cores. We also thank Ambarish Vyas for preliminary results regarding hash cores with padding units, and Rajesh Velegalati for extensive help with multiple ATHENa runs.

References:

[1] ATHENa Project Website, http://cryptography.gmu.edu/athena/.

[2] A. Akin, A. Aysu, O.C. Ulusel, and E. Savas, "Efficient Hardware Implementation of High Throughput SHA-3 Candidates Keccak, Luffa and Blue Midnight Wish for Single- and Multi-Message Hashing," The Second SHA-3 Candidate Conference, Aug. 23-24, 2010.

[3] B. Baldwin, N. Hanley, M. Hamilton, L. Lu, A. Byrne, M. O'Neill and W.P. Marnane, "FPGA Implementations of the Round Two SHA-3 Candidates," The Second SHA-3 Candidate Conference, Aug. 23-24, 2010.

[4] 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, pp. 264-278.

[5] 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.

[6] X. Guo, S. Huang, L. Nazhandali, and P. Schaumont, "Fair and Comprehensive Performance Evaluation of 14 Second Round SHA-3 ASIC Implementations," The Second SHA-3 Candidate Conference, Aug. 23-24, 2010.

[7] X. Guo, M. Srivistav, S. Huang, D. Ganta, M. Henry, L. Nazhandali, and P. Schaumont, "Pre-silicon Characterization of NIST SHA-3 Final Round Candidates", 14th Euromicro Conference on Digital System Design Architectures, Methods and Tools, DSD 2011, Oulu, Finland, Aug. 31-Sep. 2, 2011.

[8] F. K. Gürkaynak, K. Gaj, B. Muheim, E. Homsirikamol, C. Keller, M. Rogawski, H. Kaeslin, and J.-P. Kaps, "Lessons Learned from Designing a 65nm ASIC for Evaluating Third Round SHA-3 Candidates," submission to the 3rd SHA-3 Candidate Conference, Washington D.C., March 2012.

[9] 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, pp. 248-263.

[10] E. Homsirikamol, M. Rogawski, and K. Gaj, "Comparing Hardware Performance of Fourteen Round Two SHA-3 Candidates Using FPGAs," Cryptology ePrint Archive: Report 2010/445.

[11] E. Homsirikamol, M. Rogawski, and K. Gaj, "Throughput vs. Area Trade-offs in High-Speed Architectures of Five Round 3 SHA-3 Candidates Implemented Using Xilinx and Altera FPGAs," LNCS 6917, Cryptographic Hardware and Embedded Systems workshop, CHES 2011, Nara, Japan, Sep. 28-Oct. 1, pp. 491-506.

[12] B. Jungk and J. Apfelbeck, "Area-Efficient FPGA Implementations of the SHA-3 Finalists," 2011 International Conference on ReConFigurable Computing and FPGAs, ReConFig 2011, Cancun, Mexico, Nov. 30-Dec. 2, 2011.

[13] J.-P. Kaps, P. Yalla, K.K. Surapathi, B. Habib, S. Vadlamudi, S. Gurung, and J. Pham, "Lightweight Implementations of SHA-3 Candidates on FPGAs," 12th International Conference on Cryptology, Indocrypt 2011, Chennai, Dec. 11-14, 2011.

[14] S. Kerckhof, F. Durvaux, N. Veyrat-Charvillon, F. Regazzoni, G. Meurice de Dormale, F.-X. Standaert, "Compact FPGA Implementations of the Five SHA-3 Finalists," 10th Smart Card Research and Advanced Application Conference, CARDIS 2011, Leuven, Belgium, Sep. 14-16, 2011.

[15] S. Matsuo, M. Knezevic, P. Schaumont, I. Verbauwhede, A. Satoh, K. Sakiyama, and K. Ota, "How Can We Conduct "Fair and Consistent" Hardware Evaluation for SHA-3 Candidate?" The Second SHA-3 Candidate Conference, 2010, Aug. 23-24, 2010.

[16] M. Rogawski and K. Gaj, "Groestl Tweaks and their Effect on FPGA Results," Cryptology ePrint Archive: Report 2011/635.

[17] SHA-3 Contest: http://csrc.nist.gov/groups/ST/hash/sha-3/index.html

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

[19] R. Shahid, M.U. Sharif, M. Rogawski, and K. Gaj, "Use of Embedded FPGA Resources in Implementations of SHA-3 Candidates," The 2011 International Conference on Field-Programmable Technology, FPT 2011, New Delhi, India, Dec. 12-14, 2011.

[20] 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.

[21] A. Vyas, "Implementing and Benchmarking of Padding Units and HMAC for SHA-3 Candidates in FPGAs and ASICs," Master's Thesis, George Mason University, Fall 2011.

Appendix A

Table A.1: Results for the 512-bit variants of the Round 3 SHA-3 candidates and SHA-2. Notation: Tp – throughput, A – area, Tp/A – Throughput to Area Ratio. The best values of the throughput to area ratios and the best architectures are given in **bold**.

Arch	Virtex 5			Virtex 6			Stratix III			Stratix IV		
	Tp	A	Tp/A	Tp	A	Tp/A	Tp	A	Tp/A	Tp	A	Tp/A
BLAKE-512												
/4(h)/4(v)-m	560	386	**1.45**	613	309	1.98	491	1680	0.29	543	1676	0.32
/4(h)	2300	2840	0.81	2646	1584	1.67	2186	5891	0.37	2442	5885	0.42
/2(h)	3264	3435	0.95	3478	2610	1.33	2928	6977	0.42	3318	6971	0.48
x1	N/A	N/A	N/A	N/A	N/A	N/A	2965	9033	0.33	3323	9024	0.37
/2(h)-PPL2	4841	4515	1.07	4478	2879	1.56	3954	8969	0.44	4742	8959	0.53
/2(h)-PPL4	6171	5794	1.07	6915	4575	1.51	5991	9684	0.62	7859	9694	0.81
x1-PPL2	6364	5674	1.12	6471	4571	1.42	5660	10625	0.53	6351	10615	0.60
x1-PPL4	9567	7497	1.28	10706	5267	**2.03**	9980	12074	**0.83**	11075	12082	**0.92**
Groestl-512												
/8(v) (P+Q)	1556	2251	0.69	1726	1773	0.97	1677	6549	0.26	1614	6510	0.25
/4(v) (P+Q)	3112	2393	1.30	3230	2113	1.53	3447	8727	0.39	3277	8750	0.37
/2(v) (P+Q)	5119	3289	1.56	5793	2971	1.95	6595	14318	0.46	6265	14207	0.44
x1 (P+Q)	9582	5797	1.65	11857	5234	2.27	13061	22062	0.59	11936	21902	0.54
x1-PPL2 (P+Q)	18213	10263	1.77	18998	7528	2.52	16900	24292	0.70	16114	24241	0.66
x1-PPL4 (P+Q)	N/A	N/A	N/A	N/A	N/A	N/A	21158	25515	**0.83**	20580	25407	**0.81**
/8(v) (P/Q)	1211	1722	0.70	1326	1358	0.98	1335	4598	0.29	1307	4592	0.28
/4(v) (P/Q)	2573	2036	1.26	2772	1529	1.81	2700	5786	0.47	2596	5770	0.45
/2(v) (P/Q)	4816	2336	**2.06**	5319	1761	**3.02**	5262	7763	0.68	5262	7763	0.68
x1 (P/Q)	7686	3853	1.99	8375	3630	2.31	8669	12450	0.70	8504	12368	0.69
JH-512												
/8(v)-m (MEM)	138	307	0.45	154	228	0.68	128	1817	0.07	119	1851	0.06
/2(v) (MEM)	2052	1055	1.95	2491	944	2.64	2224	3664	0.61	2175	3660	0.59
x1 (MEM)	4882	1037	4.71	5825	931	**6.26**	5011	3288	1.52	5139	3294	**1.56**
x2 (MEM)	6203	1587	3.91	6859	1377	4.98	6630	5768	1.15	6305	5786	1.09
x1-PPL2 (MEM)	4635	1990	2.33	5060	1534	3.30	5361	4521	1.19	5319	4521	1.18
x2-PPL2 (MEM)	8183	2494	3.28	9439	2128	4.44	9881	6339	**1.56**	9665	6309	1.53
x2-PPL4 (MEM)	8107	3408	2.38	8237	2288	3.60	9456	7427	1.27	8806	7392	1.19
/2(v) (OTF)	2024	1127	1.80	2104	954	2.21	2107	3680	0.57	1982	3669	0.54
x1 (OTF)	4686	992	**4.72**	5181	939	5.52	5181	3557	1.46	5043	3605	1.40
x2 (OTF)	6413	1870	3.43	7128	1501	4.75	6268	6276	1.00	6032	6314	0.96
Keccak-512												
/8(v)-m	495	344	1.44	579	320	1.81	500	2398	0.21	467	2395	0.19
x1	7612	1320	**5.77**	7208	1061	**6.79**	8526	3471	2.46	7825	3467	2.26
x1-PPL2	9306	1720	5.41	9619	1468	6.55	11215	4294	**2.61**	10816	4295	**2.52**
x2-PPL2	9915	2297	4.32	N/A	N/A	N/A	13389	6523	2.05	12984	6519	1.99
x2-PPL4	12935	3387	3.82	15661	2539	6.17	20356	8553	2.38	19300	8549	2.26
Skein-512												
x1	1201	1069	1.12	1441	983	1.47	1135	3072	0.37	1229	3073	0.40
x4	3084	1418	2.17	3462	1114	3.11	2438	4006	0.61	2736	4015	0.68
x8	2832	1577	1.80	3573	1373	2.60	3121	5589	0.56	3322	5507	0.60
x4-PPL2	5378	2026	**2.65**	5943	1702	**3.49**	4271	4705	0.91	4682	4683	1.00
x4-PPL5	N/A	N/A	N/A	7071	3486	2.03	6670	6199	**1.08**	6972	6185	**1.13**
x8-PPL10	N/A	N/A	N/A	12176	6145	1.98	11063	11205	0.99	10802	11204	0.96
SHA-512												
x1	2013	798	**2.52**	2422	553	**4.38**	2128	1995	**1.07**	2378	1996	**1.19**

Table A.2: The effect of the padding unit on the performance of 5 Round 3 SHA-3 candidates in four FPGA families, Virtex 5 and Virtex 6 from Xilinx, and Stratix III and Stratix IV from Altera. Notation: Tp – throughput, A – area, Tp/A – Throughput to Area Ratio, Δ [%] – relative change in the Throughput, Area, and Throughput to Area ratio as a result of adding padding unit to the hash unit. The relative change in the throughput to area ratio has been marked in **bold**.

Arch	Virtex 5			Virtex 6			Stratix III			Stratix IV		
	Tp	**A**	**Tp/A**	**Tp**	**A**	**Tp/A**	**Tp**	**A**	**Tp/A**	**Tp**	**A**	**Tp/A**
BLAKE-256												
/2(h)	2308	1771	1.30	2226	1257	1.77	2157	3553	0.61	2337	3543	0.66
/2(h) - PAD	2266	1860	1.22	2363	1391	1.70	2206	3660	0.60	2316	3680	0.63
Δ [%]	-1.83	5.03	**-6.53**	6.18	10.66	**-4.04**	2.25	3.01	**-0.74**	-0.90	3.87	**-4.59**
Groestl-256 (P/Q)												
x1	6117	1795	3.41	7220	1870	3.86	6604	6460	1.02	6269	6421	0.98
x1 - PAD	6572	2020	3.25	7071	1884	3.75	6160	6466	0.95	6033	6415	0.94
Δ [%]	7.44	12.53	**-4.53**	-2.06	0.75	**-2.79**	-6.72	0.09	**-6.81**	-3.76	-0.09	**-3.67**
JH-256 (MEM)												
x1	4955	982	5.05	5412	849	6.37	5276	3221	1.64	4759	3210	1.48
x1 - PAD	4543	1001	4.54	5086	918	5.54	5024	3383	1.49	4815	3415	1.41
Δ [%]	-8.32	1.93	**-10.06**	-6.02	8.13	**-13.09**	-4.77	5.03	**-9.33**	1.17	6.39	**-4.90**
Keccak-256												
x1	13337	1369	9.74	11839	1086	10.90	15493	3531	4.39	16104	3471	4.64
x1- PAD	12745	1375	9.27	12451	1147	10.86	14624	4060	3.60	15167	3734	4.06
Δ [%]	-4.44	0.44	**-4.86**	5.16	5.62	**-0.43**	-5.61	14.98	**-17.91**	-5.82	7.58	**-12.45**
Skein-256												
x4	3023	1218	2.48	3373	1005	3.36	2475	3943	0.63	2592	3936	0.66
x4 - PAD	3127	1245	2.51	2957	1026	2.88	2495	3960	0.63	2647	3970	0.67
Δ [%]	3.43	2.22	**1.19**	-12.33	2.09	**-14.13**	0.77	0.43	**0.34**	2.10	0.86	**1.23**

Table A.3. Change in the results between the logic-only implementation (without DSP units and Block RAMs) and the implementation using these embedded resources. The respective columns represent: **Δ**Throughput [%] - Relative Improvement in Throughput, **Δ**Reconfigurable Logic [%] - Relative Reduction in the amount of Reconfigurable Logic, **Δ**Tp/Reconfigurable Logic [%] - Relative Improvement in Throughput/Reconfigurable Logic Ratio. N/A - indicates that an investigated architecture with embedded resources did not improve any of the performance measures. *All results presented in this table (unlike the results in the rest of the paper) have been obtained for architectures without padding units, using Xilinx ISE 12.3, and Altera Quartus II 10.0.*

Algorithm	**Δ Throughput [%]**		**Δ Reconfigurable logic [%]**		**ΔTp/Reconfigurable logic [%]**	
	Virtex 5	**Stratix III**	**Virtex 5**	**Stratix III**	**Virtex 5**	**Stratix III**
BLAKE	-17.4	-15.7	57.3	47.6	93.7	61.0
Groestl	-3.8	20.9	41.5	61.8	64.6	216.2
JH	-10.0	2.8	-5.1	11.3	-14.4	15.9
Keccak	8.1	0.6	-9.3	-0.5	-1.1	0.1
Skein	-20.0	N/A	2.4	N/A	-18.0	N/A

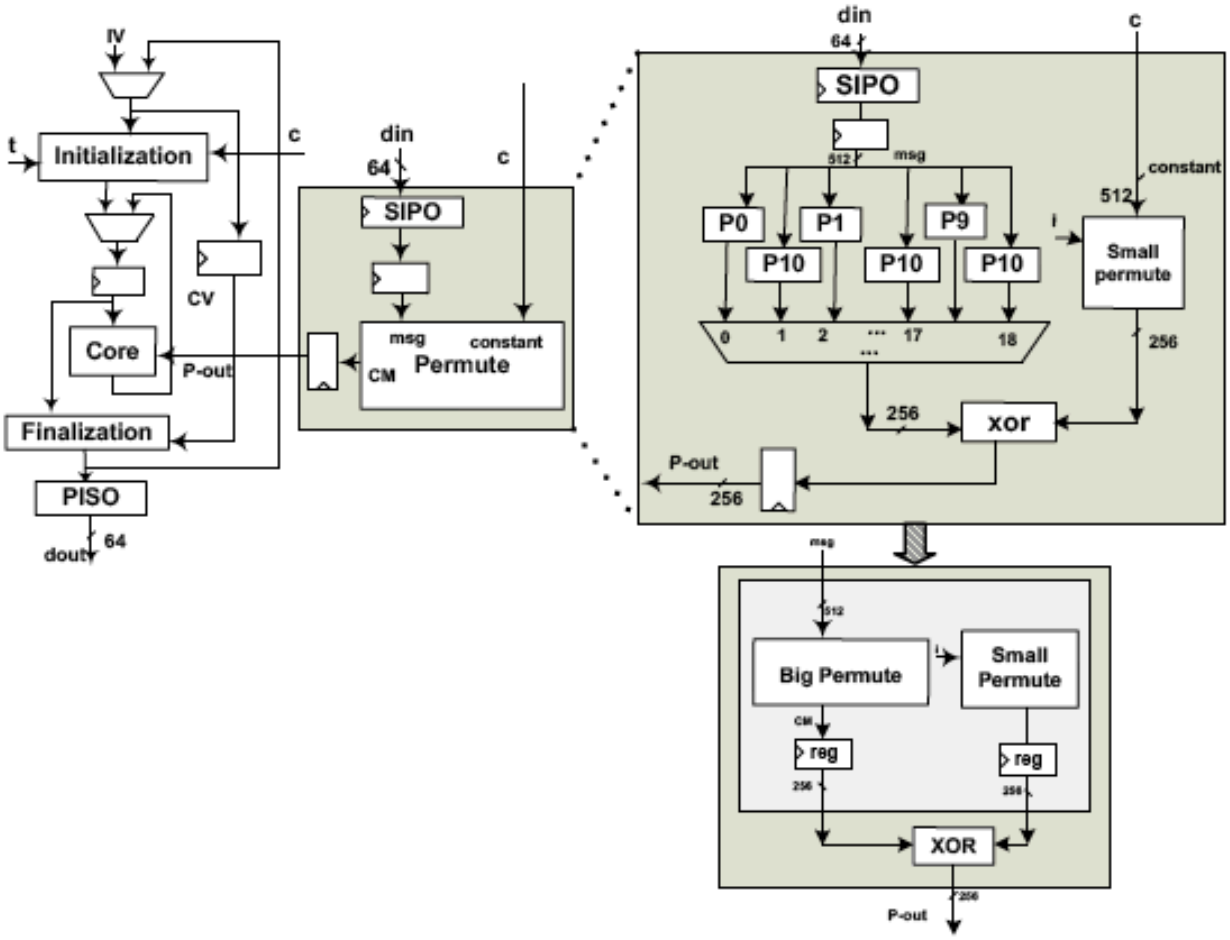

Fig. A.1. BLAKE. Transformation of the datapath from the logic-only implementation to the implementation using embedded resources.

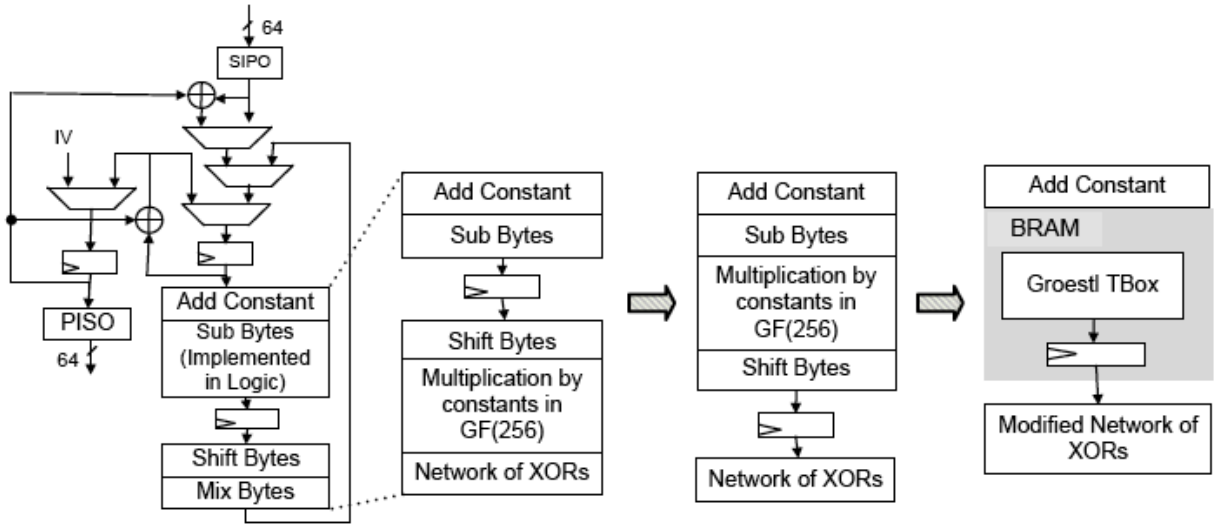

Fig. A.2. Groestl. Transformation of the datapath from the S-Box based logic-only implementation to the T-Box based implementation using embedded resources.

Table A.4. Change in the throughput to area ratio between the logic-only implementation (without DSP units and Block RAMs) and the implementation using these embedded resources. Value given in **bold** represents the best result for a given algorithm and FPGA family, and the improvement column represents a relative improvement compared to the basic architecture achieved using *the best of the two architectures*. *All results presented in this table (unlike the results in the rest of the paper) have been obtained for architectures without padding units, using Xilinx ISE 12.3, and Altera Quartus II 10.0.*

Algorithm & Architecture	**Virtex 5**			**Stratix III**		
	Logic-only	**With embedded resources**	**Improvement**	**Logic-only**	**With embedded resources**	**Improvement**
BLAKE /2(h)	1.32	**2.56**	94%	0.59	**0.95**	61%
Groestl x1 (P/Q)	3.28	**5.41**	65%	0.79	**2.49**	216%
JH x1 (MEM)	**4.67**	4.00	0%	1.37	**1.59**	16%
Keccak x1	**10.26**	10.15	0%	**3.28**	3.28	0%
Skein x4	**2.28**	1.87	0%	**0.62**	0.26	0%