Use of Embedded FPGA Resources in Implementations of Five Round Three SHA-3 Candidates

Malik Umar Sharif, Rabia Shahid, Marcin Rogawski, Kris Gaj

Abstract

In this paper, we present results of the comprehensive study devoted to the optimization of FPGA implementations of SHA-2 and five SHA-3 finalists using embedded FPGA resources, such as Digital Signal Processing (DSP) units and Block Memories. Our methodology involves implementing, characterizing, and comparing all algorithms with a focus on minimizing the amount of reconfigurable logic resources, and achieving a better balance between the use of reconfigurable and embedded resources. All designs are implemented using four FPGA families, representing major low-cost and high-performance families of Xilinx and Altera.

1 INTRODUCTION

Practically all FPGA vendors incorporate in modern FPGAs, apart from basic reconfigurable logic resources, also embedded resources, such as large memory blocks, DSP units, microprocessors, etc. Improved hardware performance and good balance in terms of the overall FPGA utilization can be achieved with the use of these embedded elements for multiple applications, such communications, digital signal processing, and scientific computing.

Cryptographic algorithms have been demonstrated in the past to take advantage of these resources as well. For example, the fastest to date FPGA implementation of the Montgomery multiplication, a major building block of public key cryptographic algorithms, such as RSA, have been demonstrated using DSP units in Virtex 5 FPGAs [1]. Advanced Encryption Standard, a major secret key cryptosystem used for bulk data encryption, has been sped up first by using Block Memories of Xilinx and Altera FPGAs [2], and then by using a combination of DSP units and Block RAMs in Virtex 5 FPGAs [3, 4].

In this paper we focus on the use of embedded resources in efficient implementations of five Round 3 SHA-3 candidates. Hardware performance evaluation plays a major role in the evaluation cycle of these hash functions because it provides a clear and objective measure that can be used to rank all candidates, especially in the absence of security weaknesses, which are much harder to identify in a relatively short period devoted to analysis.

Major results generated by multiple groups for FPGA implementations of 14 Round 2 candidates are summarized in [7, 8, 9]. The detailed review of these implementations

reveals that very few of them take advantage of DSP units or Block Memories of modern FPGAs. One of the reasons for this approach is the difficulty in optimizing such implementations in terms of the throughput to area ratio, as area is not easy to define (or measure) when multiple resources are used.

We believe that our study is the first one in the literature that looks comprehensively at utilizing embedded resources in a large class of hash functions, including all 5 Round 3 SHA-3 candidates.

2 DESIGN ENVIRONMENT AND METHODOLOGY

All investigated hash functions have been modeled in VHDL-93. Xilinx ISE Design Suite v.12.3 and Altera Quartus II v.10.0 were used for synthesis and implementation of all designs. A benchmarking tool, called ATHENa, was used to collect results for each hash function [10, 11].

Similarly to earlier papers [7, 8, 9], we use Throughput (Mbits/sec) as our major speed metrics. We use the resource utilization vector to indicate the resource utilization of each SHA-3 candidate, as shown in Table 1.

Our primary optimization target is the improvement of the ratio: throughput over the amount of reconfigurable logic resources. We define the amount of reconfigurable logic resources as the number of Configurable Logic Block slices (#CLB_slices) for Xilinx FPGAs, as the number of Logic Elements (#LEs) for low-cost Altera families, and as the number of Adaptive Look-Up Tables for high-performance Altera families (#ALUTs). Our secondary optimization targets are the improvement in throughput, and the reduction in the amount of reconfigurable logic resources.

Table 1. Resource Utilization Vectors.

Vendor	Family	Resource Utilization Vector
Xilinx	Spartan 3	(#CLB_slices, #BRAMs, #multipliers)
	Virtex 5	(#CLB_slices, #BRAMs, #DSP48s)
Altera	Cyclone II	(#LEs, #BRAMs, #multipliers)
	Stratix III	(#CLB_slices, #BRAMs, #DSP_18s)

3 OVERVIEW OF AVAILABLE EMBEDDED RESOURCES

3.1 DSP Units and multipliers

Xilinx Virtex 5 FPGAs include DSP48E units. Each unit has a two-input multiplier followed by multiplexers and a three-input adder/subtractor/accumulator. The unit can

be configured as a 25x18 multiplier and/or 48-bit adder with up to three inputs. The third input of an adder can be used only when multiple DSP units are cascaded and an adder output of one DSP unit is connected to an adder input of an adjacent DSP unit.

The DSP unit of the Stratix III FPGAs consists of 8 DSP_18 units, each with its own 18x18-bit multiplier. Two neighboring DSP_18 units share a 37-bit adder. The outputs of two 37-bit adders can be furher combined using the following adder-accumulator.

Xilinx Spartan 3 and Altera Cyclone II contain only embedded multipliers. Spartan 3 devices support 18x18 signed multiplication. Cyclone II devices support 9x9 and 18x18 multiplication for signed and unsigned numbers.

3.2 Block Memory

The Block Memory (BRAM) in Spartan 3 FPGAs has a size of 18 kbits, including parity bits. Word size is configurable in the range from 1 to 36 bits. The maximum word size is used in the configuration 512 x 36 bits. The block memory (BRAM) in Virtex 5 FPGAs can store up to 36 kbits of data. It supports two independent 18 kbit blocks (with the word size up to 18 bits), or a single 36 kbit memory block (with the word size up to 36 bits).

Altera devices have different capacity of basic embedded memory blocks. The low-cost Cyclone II family is based on 4 kbit blocks. The high-performance Stratix III family is less homogenous. It consists of two types of memory blocks - 9 kbits and 144 kbits. All block memories have single-port and dual-port modes, and can be used to implement any operations that can be expressed in terms of table look-ups.

4 CATEGORIES

In Table 2, we present a list of internal operations of 5 Round 3 SHA-3 candidates (256-bit variants), suitable for implementing using embedded resources of modern FPGAs. The SHA-3 candidates can be divided into the following three categories based on the potential use of embedded resources.

A. Functions using DSP Units
B. Functions using DSP Units and Block Memories
C. Functions using Block Memories.

Skein is the only algorithm that uses 64-bit addition. BLAKE includes 32-bit additions and large ROMs used to implement the message expansion tables. JH and Keccak use Block Memories to store round constants.

For the Advanced Encryption Standard (AES) based algorithm, Groestl, two architectures were investigated: first based on S-boxes and the second based on T-boxes [2].

Table 2. Internal operations of 5 Round 3 SHA-3 candidates (256-bit variants), suitable for implementing using embedded resources of modern FPGAs. Notation: mADDn - multi-operand addition with n operands, ADD 32-bit addition with two operands, ADD-64, 64-bit addition with two operands.

Hash Function	Tables of Constants	MUL	mADDn	ADD
DSP Units				
Skein				ADD-64
DSP Units and Block Memories				
BLAKE	Message Expansion Tables		mADD3	ADD
SHA-2	Round Constants		mADD6	ADD
Block Memories				
Groestl	AES S-box or T-box			
JH	Round Constants			
Keccak	Round Constants			

5 IMPLEMENTATION DETAILS OF SELECTED FUNCTIONS FROM EACH CATEGORY

5.1 Functions using DSP Units

5.1.1 Skein

Skein performs 64-bit addition in the main iterative loop of the algorithm. In Xilinx Virtex 5, each 64-bit addition is performed by utilizing two cascaded DSP48E units because one DSP adder can add only two operands of the size up to 48 bits. Two adjacent DSP-based adders are cascaded by connecting the carry output of one adder to the carry input of the adjacent adder. In Altera Stratix 3, 64-bit addition is performed by using mega plugin wizard in order to instantiate 64-bit input DSP adder.

5.2 Functions using DSP Units and Block Memories

5.2.1 BLAKE

G-Functions in BLAKE constitute the main iterative task of the algorithm. In the basic implementation [9], these G-Functions are implemented as combinational logic. Each of these G-Functions has two 32-bit, 2-operand additions and two 32-bit additions with three operands. All these additions were implemented using DSP adders for the embedded version of the design.

In the embedded design, the permute block was implemented using block memory. In this design, SIPO (Serial-in-Parallel-out Unit) from [9] was removed and instead, the message block gets directly stored into block memory. Two memory banks were used to allow simultaneous loading of the next message block while the first block is being

processed. To implement the whole PERMUTE operation, two 16x256 memory block along with a 28x256 ROM to store constants was required.

5.2.2 SHA-2

Current standard SHA-2 [14] was implemented according the architecture described by Chaves et al. [12]. This architecture is the best known architecture in terms of the throughput/area ratio. SHA-2 round constants were implemented for both 256 and 512 bits version by 64x32 and 80x64 ROM respectively. All additions in the SHA-2 round function were transferred to DSP blocks.

Moreover we decided to use message scheduler architecture proposed in [13] in order to utilize DSP capabilities.

5.3 Functions using Block Memories

5.3.1 Groestl

For Groestl, which has AES-based round, we have implemented two versions: first, based on S- boxes, and the second, based on T-boxes [2,3,4].

Each round in Advanced Encryption Standard (AES) has four basic operations: SubBytes, ShiftRows, MixColumns, and AddRoundKey. The SubBytes operation is equivalent to an 8x8 substitution box (S-box). It can be implemented using a 256x8 bit look-up table, which is suitable for implementation using block memories. The other approach is a T-box based architecture. This approach allows the computation of an entire AES round using only look-up tables and XOR operations [2, 3, 4]. For the computation of entire round, 256 x 40 bit look-up tables were required.

5.3.2 JH

The key generator unit R6 from the basic implementation [9] computes the value for each key on the fly, whereas we modified the design to use pre-computed 42 256-bit values for each round and fed them to R8 block. The exact size and amount of memory used by Virtex 5, Spartan 3, Stratix III and Cyclone II can be seen in Table 5.

5.3.3 Keccak

Keccak was implemented using Block Memories to store the round constants (see Table 5). It required only one block memory to store all constants.

Table 3. Timing characteristics and resource utilization for basic architectures and architectures based on the use of embedded FPGA resources for implementations of 5 Round 3 SHA-3 candidates and the current standard SHA-2 in case of Xilinx Virtex 5 FPGAs. Notation: Clk Freq – clock frequency, Tp – throughput, Δ Tp – relative improvement in throughput, Δ #CLB slices – relative reduction in the number of CLB slices, Δ Tp/#CLB slices – relative improvement in throughput/#CLB slices ratio.

Algorithm	Architecture	Clk Freq [MHz]	Tp [Mbits/s]	Resource Utilization [#CLB slices, #BRAMs, #DSP48s]	Tp/#CLB slices	Δ Tp [%]	Δ #CLB slices [%]	Δ Tp/#CLB slices [%]	Resource Replacement Ratio
DSP Units									
Skein	Basic	97.4	2624	1559, 0, 0	1.68	-10.1	18.9	10.9	9.2 CLB slices/DSP
	Embedded	87.5	2359	1264, 0, 32	1.86				
DSP Units and Block RAMs									
BLAKE	Basic	119.7	2113	1854, 0, 0	1.13	-27.3	64.2	104.4	N/A
	Embedded	86.9	1534	662, 12, 8	2.31				
SHA-2	Basic	190.9	1504	418, 0, 0	3.60	14.3	23.0	49.3	N/A
	Embedded	218.2	1719	320, 1, 5	5.37				
Block RAMs									
(i) AES Tables									
Groestl	S-box Basic	330.5	8057	1627, 0, 0	4.95	-31.4	29.9	-2.2	27.0 CLB slices/BRAM
	S-box	226.6	5524	1141,18, 0	4.84				
	T-box Basic	252.1	6146	3738, 0, 0	1.64	-13.4	66.0	154.8	51.4 CLB slices/BRAM
	T-box	218.2	5320	1270, 48, 0	4.19				
(ii) Message Expansion Tables									
BLAKE	Basic	119.7	2113	1854, 0, 0	1.13	-11.9	60.8	126.5	86.7 CLB slices/BRAM
	Embedded	105.4	1861	726,13, 0	2.56				
(iii) Round Constant Tables									
JH	Basic	413.0	4917	1056, 0, 0	4.65	-36.5	-0.9	-37.2	-2.5 CLB slices/BRAM
	Embedded	262.2	3120	1066, 4, 0	2.92				
Keccak	Basic	298.6	13536	1352, 0, 0	10.01	-16.9	1.0	-16.0	14 CLB slices/BRAM
	Embedded	248.2	11252	1338, 1, 0	8.41				
SHA-2	Basic	190.9	1504	418,0,0	3.60	5.8	8.8	15.8	37 CLB slices/BRAM
	Embedded	201.9	1591	381,1,0	4.17				

6 RESULTS

In this section, we present a comparison between the basic designs, implemented using reconfigurable logic, and embedded designs, with DSP units and Block Memories. Results for 256-bit versions of all investigated hash functions and four FPGA families are summarized in Tables 3-8 and Figs 1 and 2. BLAKE and JH results include Round 3 tweaks introduced by the authors, whereas Groestl results are based on Round 2 specifications.

Addition operations are implemented in DSP units. Although DSP48E units in Virtex 5 slightly reduce the throughput due to the use of cascaded DSP architecture for 64-bit addition, we are able to reduce the number of CLB slices. This in turn helped to increase the overall throughput to #CLB slice ratio. On the other hand, Skein with 64 bit addition exposes a lack of good support for this precision, especially in Altera Stratix

III where extra logic needs to be generated on top of the DSP support in order to perform a 64-bit addition. This affects both throughput and area.

Table 4. Timing characteristics and resource utilization for basic architectures and architectures based on the use of embedded FPGA resources for implementations of 5 Round 3 SHA-3 candidates and the current standard SHA-2 in case of Altera Stratix III FPGAs. Notation: Clk Freq – clock frequency, Tp – throughput, Mem-bits – number of memory bits, Δ Tp – relative improvement in throughput, Δ #ALUTs – relative reduction in the number of ALUTs, Δ Tp/#ALUTs – relative improvement in throughput/#ALUTs ratio, #DSP_18s – number of DSP_18 units; one DSP unit consists of 8 such elements.

Algorithm	Architecture	Clk Freq [MHz]	Tp [Mbits/s]	Resource Utilization [#ALUTs, #Mem-bits, #DSP_18s]	Tp/#ALUTs	Δ Tp [%]	Δ #ALUTs [%]	Δ Tp/#ALUTs [%]	Resource Replacement Ratio
DSP Units									
Skein	Basic	90.0	2426	4381, 0, 0	0.55	-39.3	-30.2	-53.4	-10.3 ALUTs/DSP
	Embedded	54.7	1472	5705, 0, 128	0.26				
DSP Units and Block Memory									
BLAKE	Basic	118.2	2086	4752,0,0	0.43	-48.5	62.6	39.5	N/A
	Embedded	60.8	1073	1773,12k,32	0.60				
SHA-2	Basic	210.0	1654	988, 0, 0	1.67	-2.0	19.5	21.8	N/A
	Embedded	205.8	1621	795, 2k,16	2.03				
Block Memory									
(i) AES Based Tables									
Groestl	Sbox-Basic	247.7	6040	7498, 0, 0	0.81	-21.4	37.0	23.4	21.1 ALUTs/1kbit
	Sbox-BRAM	194.5	4743	4727, 131k, 0	1.00				
	Tbox-Basic	251.1	6122	14567, 0, 0	0.42	-4.47	80.8	397.6	18.0 ALUTs/1kbit
	Tbox-BRAM	239.8	5848	2788, 655k, 0	2.09				
(ii) Message Expansion Tables									
BLAKE	Basic	118.2	2086	4752, 0, 0	0.43	-13.3	60.0	118.6	237.6 ALUTs/1kbit
	Embedded	102.4	1808	1900, 12k, 0	0.94				
(iii) Round Constant Tables									
JH	Basic	390.9	4654	3331, 0, 0	1.39	-0.07	17.6	22.3	65.3 ALUTs/1kbit
	Embedded	390.6	4651	2743, 9k, 0	1.70				
Keccak	Basic	303.2	13746	4221, 0, 0	3.25	2.6	-1.3	1.3	-28 ALUTs/1kbit
	Embedded	311.1	14103	4277, 2k, 0	3.30				
SHA-2	Basic	210.0	1654	988, 0, 0	1.67	4.1	3.2	4.0	16 ALUTS/1kbit
	Embedded	210.9	1661	956,2k,0	1.73				

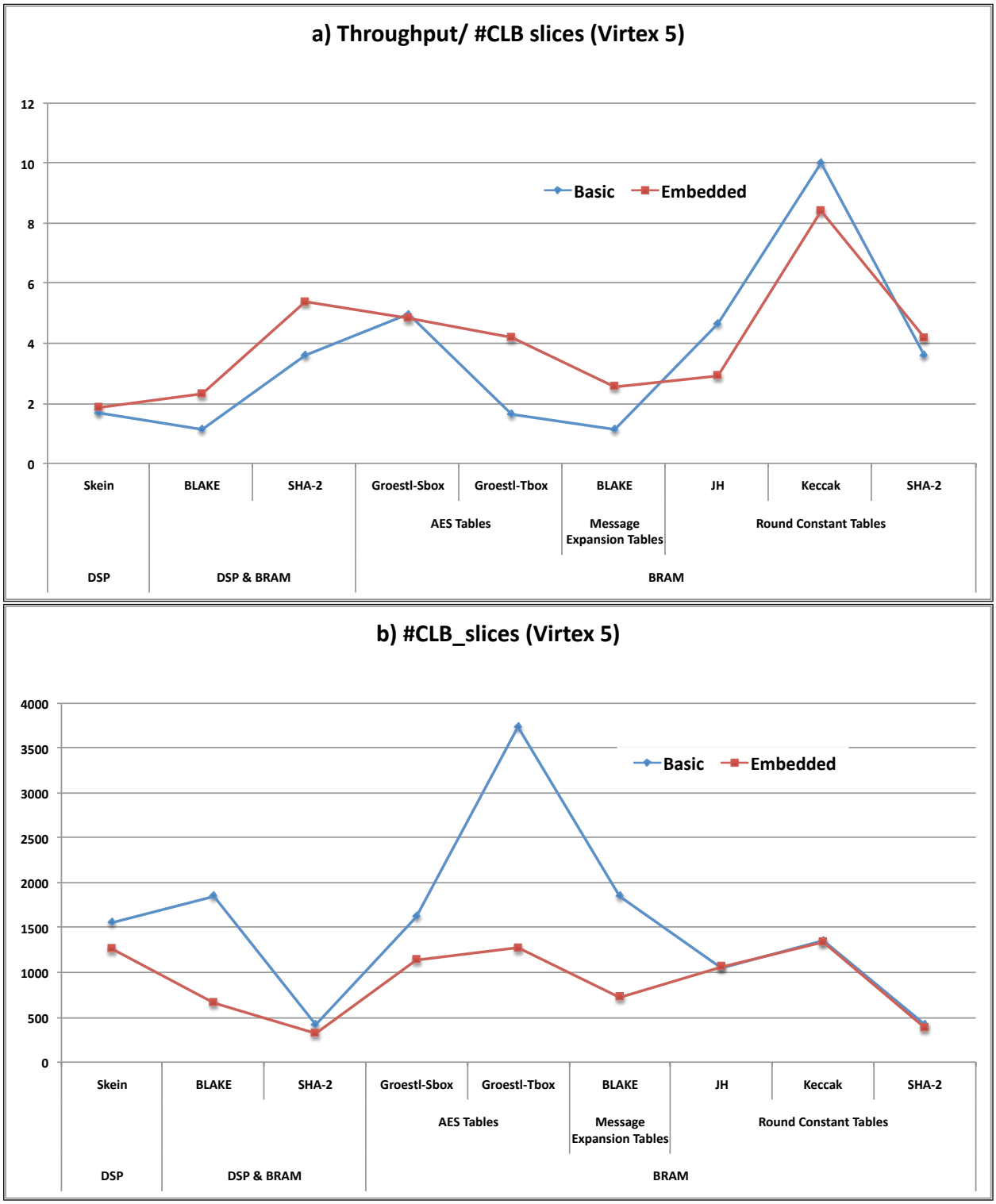

Figure 1: Comparison of basic designs with alternative designs based on the use of embedded resources in Xilinx Virtex 5 FPGAs in terms of a) Throughput/#CLB_slices b) #CLB_slices. Notation DSP : designs based on DSP units, DSP & BRAM : designs based on DSP units and Block RAMs, BRAM : designs based on Block RAMs.

Figure 2: Comparison of basic designs with alternative designs based on the use of embedded resources in Altera Stratix III FPGAs in terms of a) Throughput/#ALUTs b) #ALUTs. Notation DSP : designs based on DSP units, DSP & BRAM : designs based on DSP units Block RAMs, BRAM : designs based on Block RAMs.

Table 5. Estimate of memory used by implementations ofBLAKE, Groestl, JH, and Keccak for Spartan 3, Virtex 5, Cyclone II, and Stratix III. Notation: ()d dual-port, ()s – single-port.

Algorithm	Memory	Spartan 3	Virtex 5	Cyclone II	Stratix III
BLAKE	2 x (16 x 256) + 28 x 256 = 15 kbits	16 x (512 x 32)s + 4 x(512 x 32)d = 20 18kbit BRAMs	8 x (1k x 32)d + 4 x (1k x 32)d = 12 36kbit BRAM	8 x (128 x 32)d ; 13kbits used	8 x (512 x 32)d ; 13kbits used
Groestl S-box	64 x (256 x 8) = 128 kbits	32 x (2k x 8)d = 32 18kbit BRAMs	32 x (2k x 8)d = 16 36kbit BRAMs	32 x (512 x 8)d ; 128kbits used	16 x (1k x 8)d ; 128kbits used
Groestl T-box	64 x (256 x 40) = 640 kbits	32 x (512 x 32)d + 32 x (2k x 8bits)d = 64 18kbit BRAMs	32 x (1k x 32)d + 32 x (2k x 8bits)d = 48 36kbit BRAMs	128 x (128 x 32)s + 64 x (512 x 8)d ; 640kbits used	64 x (256 x 32)s + 16 x (1k x 8)d ; 640kbits used
JH	42 x 256 = 10.7 kbits	4 x (512 x 32) d = 4 18k BRAMs	4 x (1k x 32) d = 4 36k BRAMs	4 x (128 x 32)d ; 9kbits used	4 x (256 x 32)d ; 9kbits used
Keccak	24 x 64 = 1.5 kbits	1 x (512 x 32) d = 1 18k BRAMs	1 x (1k x 32) d = 1 36k BRAMs	1 x (128 x 32)d ; 2kbits used	1 x (256 x 32)d ; 2kbits used

BLAKE and SHA-2 have 32-bit additions in their data path. Both algorithms have functions suitable for the use of block memory i.e., BLAKE with message expansion tables and, SHA-2 with table for constants. As they can use both kinds of embedded resources, we investigated two versions for each of them, i.e., block memory only and block memory along with DSP units. BLAKE yields reduction in the throughput of both embedded designs. However, higher clock frequencies were observed for block memory only design as compared to the design with additions using DSP units (see Tables 6-8). DSP addition delays the critical path due to possible routing delays from reconfigurable logic to DSP cores. An increase in the number of adders involved in the critical path of BLAKE was also observed. However, more area was saved for the implementations using both embedded resources (BLAKE greater than 60%).

For all algorithms, except Skein on Altera Stratix III, we were able to move significant portion of logic into embedded resources. In case of functions using round constant tables (JH, Keccak), the relative improvement is not significant because those tables are relatively small (see Table 7).

AES-based function Groestl, in both S-box and T-box architectures, was giving much bigger area reduction because those operations are a big part of the entire hash function circuit (see Table 5 for the sizes of the respective tables and embedded memories used for the implementation).

Table 6. Relative Improvement in Throughput, Δ Tp [%], across four FPGA families. Notation: N/A not applicable

Algorithm	**Virtex 5**	**Spartan 3**	**Stratix III**	**Cyclone II**
DSP Units				
Skein	-10.1	N/A	-39.3	N/A
DSP Units and Block Memory				
BLAKE	-27.3	N/A	-48.5	N/A
SHA-2	14.3	N/A	-2.0	N/A
Block Memory				
(i) AES Tables				
Groestl (S-box)	-31.4	2.1	-36.9	5.6
Groestl (T-box)	-13.4	21.4	-25.6	29.4
(ii) Message Expansion Tables				
BLAKE	-11.9	-6.0	-13.3	-15.8
(iii) Round Constant Tables				
JH	-36.5	27.7	-0.07	0.9
Keccak	-16.9	-11.1	2.6	-0.6
SHA-2	5.8	-1.3	4.1	0.7

Table 7. Relative Reduction in the amount of Reconfigurable Logic Resources (CLB slices for Spartan 3 and Virtex 5, LEs for Cyclone II, and ALUTs for Stratix III) across four FPGA families. Notation: N/A not applicable

Algorithm	**Virtex 5**	**Spartan 3**	**Stratix III**	**Cyclone II**
DSP Units				
Skein	18.9	N/A	-30.2	N/A
DSP Units and Block Memory				
BLAKE	64.2	N/A	62.6	N/A
SHA-2	23.0	N/A	19.5	N/A
Block Memory				
(i) AES Tables				
Groestl (S-box)	29.9	54.0	47.3	72.0
Groestl (T-box)	66.0	64.7	82.2	93.3
(ii) Message Expansion Tables				
BLAKE	60.8	61.3	60.0	66.6
(iii) Round Constant Tables				
JH	-0.9	16.0	17.6	21.7
Keccak	1.0	0.5	-1.3	0.3
SHA-2	8.8	7.5	3.2	19.6

Table 8. Relative Improvement in the Throughput to the Amount of Reconfigurable Resources ratio across four FPGA families. Notation: N/A not applicable.

Algorithm	**Virtex 5**	**Spartan 3**	**Stratix III**	**Cyclone II**
DSP Units				
Skein	10.9	N/A	-53.4	N/A
DSP Units and Block Memory				
BLAKE	104.4	N/A	39.5	N/A
SHA-2	49.3	N/A	21.8	N/A
Block Memory				
(i) AES Tables				
Groestl (S-box)	-2.2	122.1	19.8	277.6
Groestl (T-box)	154.8	244.3	318.9	1843.6
(ii) Message Expansion Tables				
BLAKE	126.5	142.5	118.6	152.0
(iii) Round Constant Tables				
JH	-37.2	51.9	22.3	28.9
Keccak	-16.0	-10.6	1.3	-0.3
SHA-2	15.4	6.7	4.0	24.3

Our secondary optimization target was an improvement in throughput. For low-cost families, throughput increased for 7 out of 12 architecture-family pairs (see Table 6). For high performance families, we observe the frequency and throughput drop for 14 out of 18 cases (see Table 6). This drop was most likely caused by the delays between reconfigurable logic (used to implement majority of operations) and embedded resources (used to implement selected operations). Since the delays of interconnects contribute relatively more to the overall delay in high-performance families, the throughput is effected in the opposite directions for these two classes of FPGAs.

All results for basic and embedded architectures were generated using heuristic optimization methods of ATHENa [10, 11], which helped to find the best synthesis and implementation options of FPGA tools. Among all performed runs of these tools (with single run testing one set of possible options), we decided to report results that were giving us the best ratio of throughput to the amount of reconfigurable logic resources. This type of optimization criterion was selected because of two reasons. First, it helped us to identify designs with the best possible trade-off between the speed and reduction in reconfigurable logic. Secondly, these selected designs were also easier to compare with designs from [9], which used the same optimization target.

7 CONCLUSIONS

Six modern cryptographic hash functions have been implemented using four FPGA families from Xilinx and Altera. All functions have been optimized using embedded resources

of the target FPGAs, namely DSP units/ multipliers and Block Memories. Our results demonstrate significant savings in the amount of reconfigurable logic, especially high for the function based on large look-up tables, such as the AES-based candidate, Groestl, as well as BLAKE. The advantage of using DSP units and multipliers was much more limited, and typically associated with the significant performance drop. The main reason for that was that the majority of investigated hash functions, use only addition, and cannot take any advantage of multipliers present in these units.

Overall, embedded resources provide an interesting and important alternative to the use of basic reconfigurable logic resources in implementations of modern cryptographic hash functions. We hope that this paper will support the design process involving these resources, and pave the way to their extended use in future implementations of cryptography in modern FPGAs.

REFERENCES

[1] D. Suzuki, How to Maximize the Potential of FPGA Resources for Modular Exponentiation, in Proc. CHES 2007, Vienna, Austria, Sep. 2007, pp. 272-288.
[2] K. Gaj and P. Chodowiec, "FPGA and ASIC Implementations of AES," Chapter 10 in C.K. Koc (Ed.), Cryptographic Engineering, pp. 235-320, Springer, Dec. 2008.
[3] S. Drimer, Security for Volatile FPGAs, Ph.D. Dissertation, University of Cambridge, Computer Laboratory, Nov 2009, uCAM-CL-TR-763.
[4] S. Drimer, T. Guneysu and C. Paar: DSPs, BRAMs and a Pinch of Logic: Extended Recipes for AES on FPGAs, ACM Trans. Reconfig. Techn. and Systems, Issue 3, Volume 1, 1/2010.
[5] SHA-3 Contest, http://csrc.nist.gov/groups/ST/hash/sha3/index.html
[6] SHA-3 Zoo, http://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo
[7] SHA-3 Zoo: SHA-3 hardware implementations,
http://ehash.iaik.tugraz.at/wiki/SHA3_Hardware_Implementations.
[8] K. Gaj, E. Homsirikamol, and M. Rogawski, Fair and Comprehensive Methodology for Comparing Hardware Performance of Fourteen Round Two SHA-3 Candidates Using FPGAs, in Proc. CHES 2010, Santa Barbara, CA, USA, Aug. 2010, pp. 264-278.
[9] 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.
[10] ATHENaProjectWebsite, http://cryptography.gmu.edu/athena
[11] 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, in Proc. 20th Int. Conf. on Field Programmable Logic and Applications, FPL 2010, Milan, Italy.
[12] R. Chaves, G.Kuzmanov, L.Sousa, S. Vassiliadis, Improving SHA-2 Hardware Implementations in Proc. CHES 2006, Yokohama, Japan, Oct. 2006, pp. 298-310.
[13] R.P. McEvoy, F.M. Crowe, C.C. Murphy, W.P Marnane, , Optimisation of the SHA-2 Family of Hash Functions on FPGAs, in Proceedings of the 2006 Emerging VLSI Technologies and Architectures (ISVLSI06), pp. 317-322
[14] FIPS PUB 180-3