Use of Embedded FPGA Resources in Implementations of 14 Round 2 SHA-3 Candidates

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

ECE Department, George Mason University Fairfax, VA 22030, USA

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

Abstract—**In this paper, we present results of a comprehensive study devoted to the optimization of FPGA implementations of modern cryptographic hash functions using embedded FPGA resources, such as Digital Signal Processing (DSP) units and Block Memories. Fifteen hash functions, including the current American hash standard SHA-2 and 14 candidates for the new hash standard SHA-3, have been included in our investigation. 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 logic resources and embedded resources in four FPGA families, representing major low-cost and high-performance families of Xilinx and Altera.**

I. Introduction

Apart from basic reconfigurable logic resources, practically all FPGA vendors incorporate embedded resources, such as large memory blocks, DSP units, microprocessors, etc. in modern generation of FPGAs. 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 as communications, digital signal processing, and scientific computing.

In this paper we focus on the use of embedded resources in efficient implementations of a class of cryptographic algorithms, called hash functions. Hash functions are one-way functions that transform an arbitrary-length stream of input data into a fixed-size output referred to as a hash value or message digest. Some of the most important applications of these functions include their use in digital signature schemes, message authentication codes, pseudorandom number generators, etc.

These algorithms have recently come to focus because of the contest for a new American federal hash function standard called Secure Hash Algorithm 3 (SHA-3), organized by NIST in the period 2007-2012. In January 2011, this contest entered its Round 3, with only five algorithms remaining in the competition [5, 6].

All candidates submitted to the SHA-3 contest have been evaluated in terms of their security, performance in hardware and software, and flexibility [5, 6, 7]. 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.

In this study we take into account a set of 14 Round 2 candidates, as this set gives us a good opportunity to demonstrate advantages of using embedded resources for a large class of modern hash functions, representing multiple security paradigms.

In Section II, we describe the previous work presented in literature for a better understanding of our conducted study. Section III explains the design methodology followed by an overview of embedded resources in Section IV. We then categorize these hash functions in Section V. We present and discuss results, as well as highlight some implementation details in Section VI. The conclusions from our study are drawn in Section VII.

II. Previous Work

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.

There are some cases where cryptographic algorithms have been demonstrated in the past to take advantage of these resources. For example, the fastest to date FPGA implementation of the *Montgomery multiplication*, a major building block of public key cryptographic algorithms, such as RSA, has been demonstrated using DSP units in Virtex 4 FPGAs [1].

Advanced Encryption Standard (AES), 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].

Tim Güneysu in [15] presented his study on *AES* and *Elliptic Curve Cryptosystem (ECC).* He implemented 128-bit datapath of AES using dual-port block memory and DSP units in Virtex 5 FPGA with a throughput of 55 Gbits/sec. Most of

978-1-4577-1740-6/11/$26.00 © 2011 IEEE

the field operations of ECC were also shifted to a DSP unit of Virtex 4 FPGA with a yield in throughput.

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 14 Round 2 SHA-3 candidates.

Out of 14 SHA-3 candidates, the round functions of four of them are based on AES (ECHO, Fugue, Groestl, SHAvite-3). For each of these algorithms, we implemented two versions for the round function, i.e., *S-box based* and *T-box based*. S-box based and T-box based architectures of AES are described in detail in [2]. In the S-box based architecture, the SubBytes operation of AES is implemented using a series of 256x8 bit look-up tables. The remaining operations of AES, such as ShiftRows and MixColumns are implemented using logic. In the T-box based architectures, the three basic operations of AES (SubBytes, ShiftRows, and MixColumns) are combined together and represented using a larger look-up table, called the T-table. The concept of the T-table has been used in the practical implementations of AES described for example in [3, 4, 15, 16]. A similar idea, with appropriate modifications, can be applied also to the implementations of AES-based Round 2 SHA-3 candidates. To the best of our knowledge, all reported to date implementations of these four SHA-3 candidates used the S-box based architectures. We implemented both S-box and T-box based architectures and presented results of the best architecture in terms of throughput to area ratio.

III. 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]. All results presented in this paper are results obtained after placing and routing.

Similarly to earlier papers [8, 9], we use Throughput (in 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 I.

TABLE I: 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, #Block Memory Bits, #multipliers)
	Stratix III	(#ALUTs, #Block Memory Bits, #DSP_18s)

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.

Our primary optimization target is especially appropriate for situations in which embedded resources of a target FPGA remain underutilized by an application, including a hash core. In such scenarios, the exact amount of such resources used by the hash core is irrelevant, and does not need to be explicitly taken into account in the performance metrics. This situation may appear in the implementations of complex cryptographic systems-on-chip using high-performance families if none of the components of the system relies heavily on either DSP units or block memories. This assumption may also apply for stand-alone implementations of hash cores using low-cost FPGA families.

Our study is limited to versions of all investigated functions with 256-bit output. Additionally, we consider only non-pipelined architectures, i.e., architectures capable of processing only one stream of data at a time. We intend to investigate the use of pipelined architectures in our future work.

IV. OVERVIEW OF AVAILABLE EMBEDDED RESOURCES

A. *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 four subunits (called DSP_18s) and a total of eight 18x18-bit multipliers. Two neighboring 18x18 multipliers share a 37-bit adder/subtractor. The outputs of two 37-bit adders are fed to second stage 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.

B. *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, of the size of 9 kbits and 144 kbits, respectively.

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.

V. CATEGORIES

In Table II, we present a list of internal operations of 13 Round 2 SHA-3 candidates (256-bit variants), suitable for implementing using embedded resources of modern FPGAs. The 14th Round 2 candidate, Luffa, is not listed in the Table because none of its operations can be efficiently implemented using either DSP units or Block Memories.

The SHA-3 candidates can be divided into the following three categories based on the potential use of embedded resources.

- Functions using DSP Units
- Functions using DSP Units and Block Memories
- Functions using Block Memories

BMW is the only algorithm that uses multi-operand addition with more than six operands. The core of this algorithm is an adder with 17 32-bit inputs. Skein is the only algorithm that uses 64-bit addition. SIMD combines the use of adders, multipliers and ROMs.

TABLE II: Internal operations of 13 Round 2 SHA-3 candidates (256-bit variants), suitable for implementing using embedded resources of modern FPGAs. Notation: xN multiplication by a constant N, mADDn - multi-operand addition with n operands, ADD/SUB - 32-bit addition/subtraction with two operands, ADD-64, 64-bit addition with two operands. Luffa is not listed in the table as none of its operations were suitable to be implemented with embedded resources.

Hash Function	Tables of Constants	MUL	mADDn	ADD/SUB
DSP Units				
BMW			mADD17	ADD, SUB
CubeHash				ADD
Shabal		x3, x5		ADD, SUB
Skein				ADD-64
DSP Units and Block Memories				
BLAKE	Message Expansion Tables		mADD3	ADD
SHA-2	Round Constants		mADD6	ADD
SIMD	Twiddle Factors	x185, x233	mADD3	ADD
Block Memories				
ECHO	AES S-box or T-box			
Fugue	AES S-box or T-box			
Groestl	AES S-box or T-box			
SHAvite-3	AES S-box or T-box			
Hamsi	Message Expansion Tables			
JH	Round Constants			
Keccak	Round Constants			

BLAKE and Hamsi include large ROMs used to implement the message expansion tables. JH and Keccak use Block Memories to store round constants.

As stated earlier, for all hash functions with rounds based on Advanced Encryption Standard (AES) (namely, ECHO, Fugue, Groestl, and SHAvite-3), we have implemented two architectures: first based on S-boxes and the second based on T-boxes [2].

VI. 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. All basic designs are identical to those described in detail in [9]. The differences between hardware architectures used in basic designs and embedded designs are summarized in Table IV. Results for 256-bit versions of all investigated hash functions and four FPGA families are summarized in Tables III, A.1, V-VII, and Figs. 1 and 2.

It is worth mentioning here that BLAKE, SHA-2 and SIMD exist in the category of *"DSP units and Block Memories"* in Table II. However in subsequent Tables III, V-VII and A.1, these functions are placed in different categories. SIMD is moved to category *"DSP Units"*, SHA-2 remains in category *"DSP units and Block Memories"*, where as BLAKE is shifted to the category *"Block Memories"*. The decision is made on the basis of best *Throughput to Area* ratio results obtained for these functions in all three configurations *(DSP only, DSP units and Block Memories, Block Memories)*.

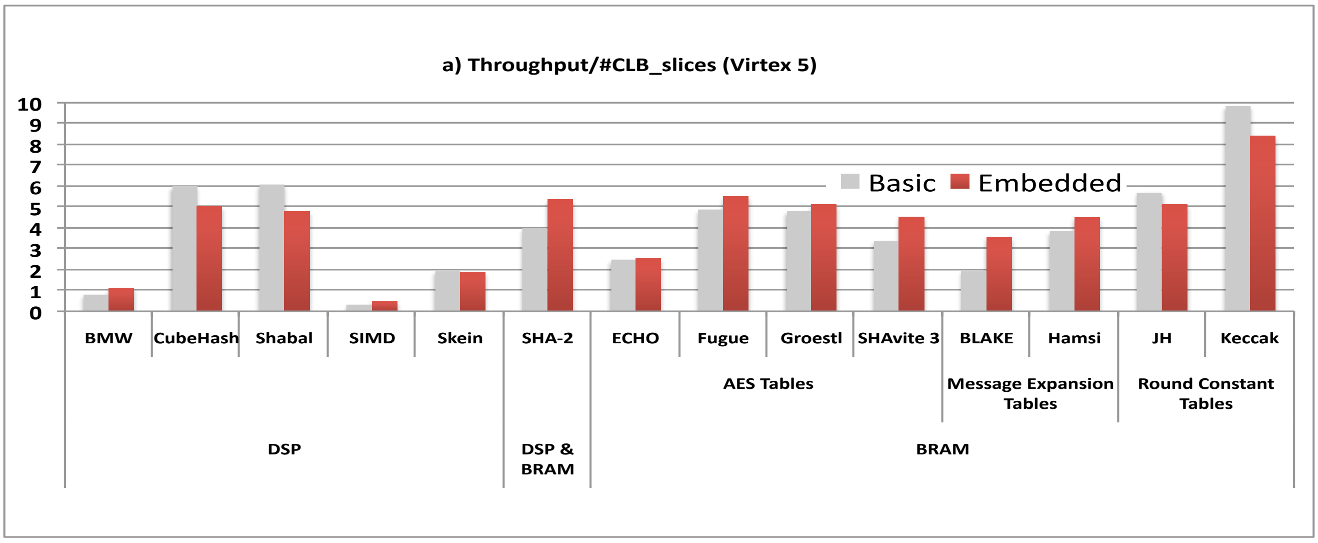
In Tables III and A.1 we present the detailed results for Virtex 5 and Stratix III, respectively. Resource Utilizations for both investigated architecture are treated as vectors. As a result, shifting resources from one category to another can be easily seen. Additionally, absolute values of maximum clock frequencies and throughputs are included as well. Finally, the Resource Replacement Ratio, defined as the amount of reconfigurable logic saved in exchange for each specific embedded block (DSP unit or memory), is provided in order to demonstrate the relative efficiency of using any given embedded block as a part of various hash functions.

In Table V, we demonstrate comprehensive results of throughput analysis across all families (Virtex 5, Spartan 3, Stratix III and Cyclone II). We optimized the designs to achieve comparable throughput while replacing logic with embedded resources. However, we observed a 17.6% drop on average in frequency and throughput across all families. For high-speed families, we observed a drop for 23 out of 28 (82.14%) hash function-FPGA family pairs. For low-cost families, this drop appeared for 9 out of 16 (56.25%) investigated pairs.

Overall, the throughput drop occurred most likely due to the delays between reconfigurable logic (used to implement majority of operations) and embedded resources. Additionally, in category *"DSP Units"*, we observed a very high throughput drop (53% on average) for embedded designs using DSP units for addition.

TABLE III: Timing characteristics and Resource Utilization for basic architectures and architectures based on the use of embedded FPGA resources for implementations of 13 Round 2 SHA-3 candidates and the current standard SHA-2 in case of *Xilinx Virtex 5* FPGAs. Notation: Clk Freq - clock frequency, Tp - throughput, ΔTp - relative *improvement* in throughput, Δ#CLB Slices - relative *reduction* in the number of CLB Slices, ΔTp/#CLB slices - relative *improvement* in throughput/#CLB slices ratio[%], Arch. - Architecture, Emb - Embedded, Util. - Utilization.

Algorithm	Arch.	Clk Freq [MHz]	Tp [Mbits/s]	Resource Util. [#CLB slices, #BRAMs, #DSPs	Tp/#CLB slices	ΔTp [%]	Δ#CLB slices [%]	ΔTp/#CLB slices [%]	Resource Replacement Ratio
A. DSP Units:									
BMW	Basic	8.7	4482	5442, 0, 0	0.82	-13.7	36.9	36.8	17.9 CLB slices/DSP
	Emb	7.5	3870	3436, 0, 112	1.13				
CubeHash	Basic	248.2	3972	663, 0, 0	5.99	-20.7	5.6	-16.0	2.3 CLB slices/DSP
	Emb	196.9	3150	626, 0, 16	5.03				
Shabal	Basic	214.9	1719	283, 0, 0	6.07	-22.2	1.4	-21.1	0.8 CLB slices/DSP
	Emb	167.1	1337	279, 0, 5	4.79				
SIMD	Basic	54.8	3121	8922, 0, 0	0.35	-24.7	46.8	41.5	43.5 CLB slices/DSP
	Emb	41.3	2350	4748, 0, 96	0.49				
Skein	Basic	95.2	2565	1306, 0, 0	1.96	-8.0	3.2	-5.0	1.3 CLB slices/DSP
	Emb	87.5	2359	1264, 0, 32	1.87				
B. DSP Units and BlockRAMs:									
SHA-2	Basic	212.6	1675	418, 0, 0	4.01	2.6	23.4	34.1	N/A
	Emb	218.2	1719	320, 1, 5	5.37				
C. Block RAMs:									
				(i) AES Tables					
ECHO	Basic	204.7	12094	4888, 0, 0	2.47	-19.4	21.1	2.3	15.4 CLB slices/BRAM
	Emb (S-box)	165.0	9748	3856, 69, 0	2.53				
Fugue	Basic	213.4	3414	700, 0, 0	4.88	-7.3	18	12.9	15.8 CLB slices/BRAM
	Emb (T-box)	197.8	3165	574, 8, 0	5.51				
Groestl	Basic	321.7	7845	1633, 0, 0	4.80	-22.3	27.3	6.9	9.3 CLB slices/BRAM
	Emb (T-box)	250.1	6098	1188, 48, 0	5.13				
SHAvite-3	Basic	271.0	3751	1104, 0, 0	3.40	-5.9	29.3	32.9	20.2 CLB slices/BRAM
	Emb (T-box)	255.1	3530	781, 16, 0	4.52				
				(ii) Message Expansion Tables					
BLAKE	Basic	130.2	3176	1623, 0, 0	1.96	-19.0	55.3	81.0	69.0 CLB slices/BRAM
	Emb	105.4	2572	726, 13, 0	3.54				
Hamsi	Basic	280.9	2997	778, 0, 0	3.85	-12.6	25.2	16.8	6.1 CLB slices/BRAM
	Emb	245.5	2619	582, 32, 0	4.50				
				(iii) Constant Tables					
JH	Basic	403.5	5739	1004, 0, 0	5.72	-12.1	1.9	-10.4	4.8 CLB slices/BRAM
	Emb	354.7	5045	985, 4, 0	5.12				
Keccak	Basic	296.7	13452	1369, 0, 0	9.83	-16.4	2.3	-14.4	31.0 CLB slices/BRAM
	Emb	248.2	11252	1338, 1,0	8.41				



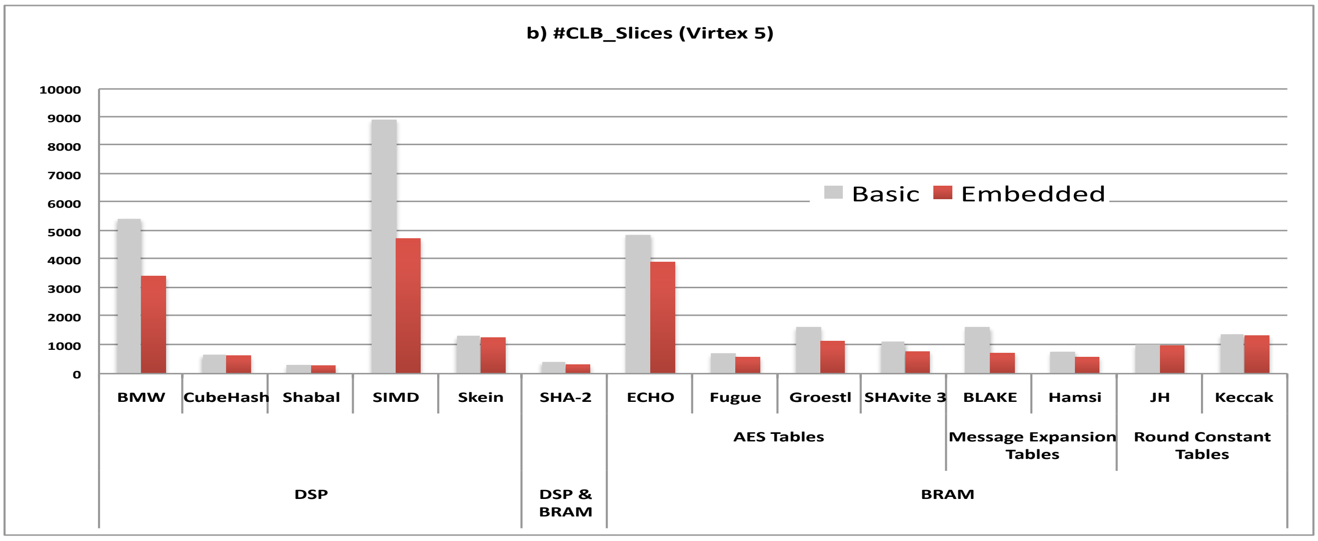
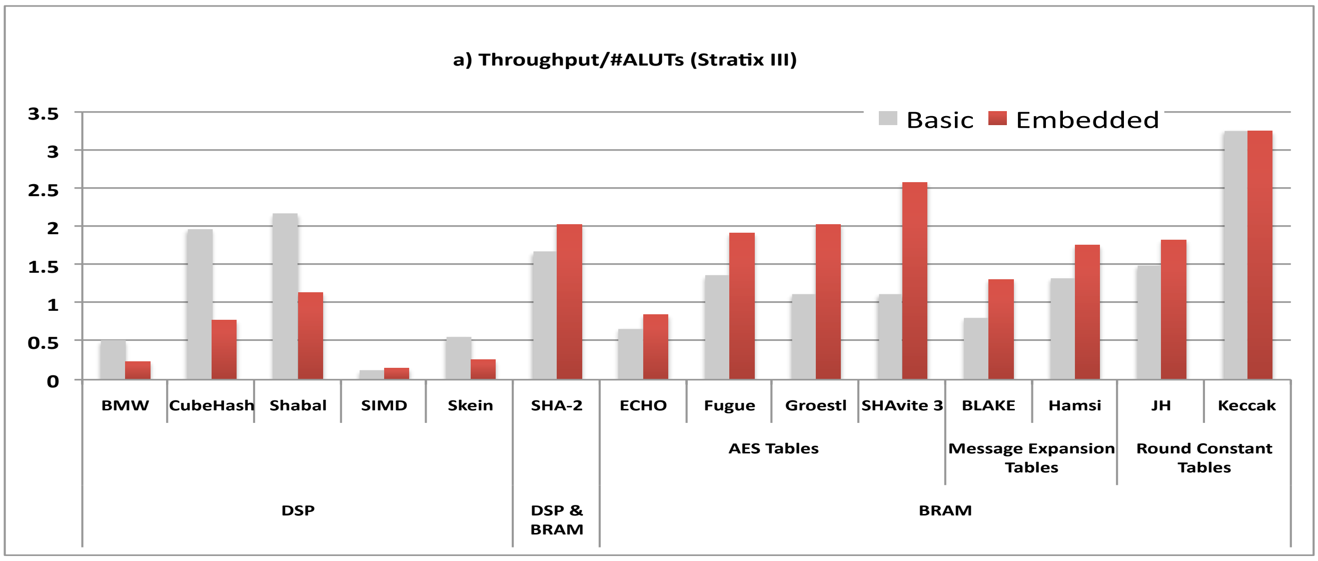


Fig. 1. Comparison of basic designs with designs based on the use of embedded resources in Xilinx Virtex 5 FPGAs in terms of a) Throughput/#CLB_slices b) #CLB_slices. Notation : DSP – Designs based on DSP units, DSP & BRAM – designs based on DSP units and BlockRAMs, BRAM – designs based on BlockRAMs.



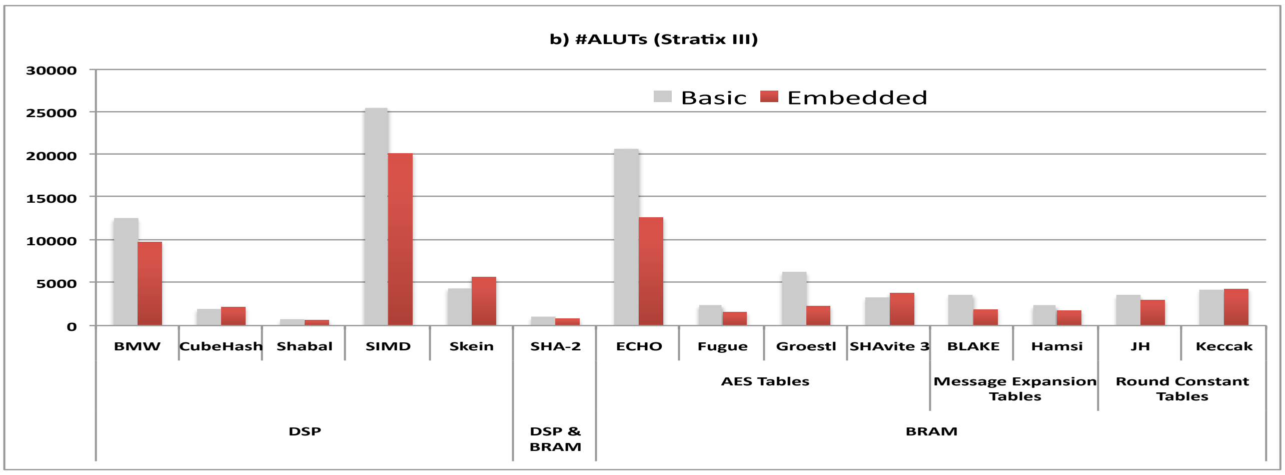


Fig. 2. Comparison of basic designs with alternative designs based on the use of embedded resources in Altera Stratix III FPGAs in terms of a) Throughput/#ALUTs b) #ALUTs. Notation: DSP – designs based on DSP units, DSP & BRAM – designs based on DSP units and Block Memory. BRAM – design based on Block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

Table IV. Characteristic features of implemented architectures, i.e., non-pipelined architectures optimized for the best ratio of Throughput to the amount of reconfigurable logic resources. Notation: ME Tables – Message Expansion Tables. Arch – Architecture.

Algorithm	**Common for Basic and Embedded Architectures**	**Specific for Basic Architecture**	**Specific for Embedded Architecture**
BLAKE	4G arch., with horizontal folding by a factor of 2.	Message Expansion implemented using Logic	Message Expansion implemented using block memories.
BMW	Fully Unrolled	All adders implemented using Standard Carry Chain Logic (“+” in VHDL)	All 17-operand adders implemented With 7 DSP units and 5 standard carry chain adders.
CubeHash	Basic Iterative Architecture	All adders implemented using Standard Carry Chain Logic (“+” in VHDL)	Selected adders (50%) implemented using DSP units
ECHO	3 Clock cycles per round	S-box based architecture with S-box Tables implemented using distributed memory or logic.	S-box based architecture using block memories for Virtex 5, Spartan 3 and Cyclone II. T-box based architecture using block memories for Stratix III.
Fugue	Basic Iterative Architecture	S-box based architecture with S-box Tables implemented using distributed memory or logic.	T-box based architecture with T-box Tables implemented using block memory
Groestl	Quasi-pipelined Architecture with P & Q permutations interleaved.	S-box based architecture with S-box Tables implemented using distributed memory or logic	T-box based architecture with T-box Tables implemented using block memory
Hamsi	Basic Iterative Architecture	Message Expansion done using logic	ME Tables implemented using block memory
JH	Basic Iterative Architecture	On the fly generation of round constants.	Round Constants permuted and stored in block memory
Keccak	Basic Iterative Architecture	Table of constants implemented using distributed memory or logic	Table of constants implemented using block memory.
Luffa	Basic Iterative Architecture	N/A	N/A
Shabal	Basic Iterative Architecture with 32-bit datapath.	Multiplication by 3 and 5 implemented using logic	Multiplication by 3 and 5 implemented using DSP units or embedded multipliers
SHAvite-3	4 Clock cycles per round	S-box based architecture with S-box Tables implemented using distributed memory or logic.	T-box based architecture with T-box Tables implemented using block memory
SIMD	4 SIMD steps unrolled	All multiplications implemented using logic	All multiplications implemented using DSP units or embedded multipliers.
Skein	4 Threefish rounds unrolled	All adders implemented using Standard Carry Chain Logic (“+” in VHDL)	Selected adders implemented using DSP units
SHA-2	Basic Iterative Architecture	Round Constants stored in distribted memory. All adders implemented using Standard Carry Chain Logic (“+” in VHDL)	Round Constants stored in block Memory. Selected adders implemented using DSP units

Between the two high-performance families, this drop was higher for Altera Stratix III than for Xilinx Virtex 5. This is because in Stratix III, DSP addition must be always executed together with preceding dummy multiplication by 1. The drop in performance was relatively smaller for SIMD in Stratix III, because in this case DSP units were used for multiplication.

For AES-based hash functions, throughput increases for low-cost families, and decreases (in 7 out of 8 cases) for high-performance families. This behavior can be explained as follows: In Spartan 3, basic implementation of an AES S-box costs 64 slices based on 4-input LUTs. For Virtex 5, the cost is 8 slices based on 6-input LUTs. The corresponding number of LUT levels is 5 for Spartan 3, and 2 for Virtex 5. Moving to the T-box based implementations in Spartan 3 replaces the large routing delay inside of an S-box, by a medium routing delay between logic and BRAMs. The same transition in Virtex 5, replaces the small routing delay inside of an S-box, by a larger routing delay between logic and BRAMs.

TABLE V: Relative Improvement in Throughput, Δ Tp [%], across four FPGA families. Notation: N/A not applicable, N/F not fitting

Algorithm	Virtex 5	Spartan 3	Stratix III	Cyclone II
A. DSP Units:				
BMW	-13.7	N/A	-62.7	N/A
CubeHash	-20.7	N/A	-55.3	N/A
Shabal	-22.2	-35.5	-54.6	-27.4
SIMD	-22.4	N/F	-1.4	N/F
Skein	-8.0	N/A	-39.5	N/A
B. DSP Units and Block Memories:				
SHA-2	2.6	N/A	-2.0	N/A
C. Block Memories:				
(i) AES Tables				
ECHO	-19.4	N/F	-22.3	N/F
Fugue	-7.3	13.7	-6.8	23.3
Groestl	-22.3	4.8	-33.5	5.6
SHAvite-3	-5.9	25.9	15.4	19.5
(ii) Message Expansion Tables				
BLAKE	-19.0	-30.4	-15.6	-40.8
Hamsi	-12.6	-8.3	0.6	-2.3
(iii) Round Constant Tables				
JH	-12.1	0.5	4.2	-13.3
Keccak	-16.4	-11.1	1.2	-0.1

Cyclone II does not contain distributed memory (i.e., memory inside of basic Logic Elements, LE) As a result, in the basic architecture, each S-box is first converted to a set of Boolean functions, and then these functions are mapped into 4-input combinational LUTs. The result amounts to 208 Logic Elements and 7 levels of LUTs per each S-box. This transition is obviously quite costly in terms of performance. The embedded T-box based designs can take advantage of 4 kbit memory blocks present in Cyclone II, and as a result are more efficient. In Stratix III, compared to Cyclone II, larger and more flexible Adaptive Look-up Tables (ALUTs) are used for implementing S-boxes. As a result, basic designs, with a small number of ALUT levels, are relatively faster than embedded designs, which suffer from the relative large interconnect delays between reconfigurable logic and memory blocks.

In Table VI, we present consolidated results of our effort to transfer reconfigurable resources to embedded resources for all investigated families (Virtex 5, Spartan 3, Stratix III and Cyclone II).

TABLE VI: 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, N/F not fitting

Algorithm	Virtex 5	Spartan 3	Stratix III	Cyclone II
A. DSP Units:				
BMW	36.9	N/A	22.0	N/A
CubeHash	5.6	N/A	-11.5	N/A
Shabal	1.4	6.0	13.2	1.4
SIMD	46.8	N/F	20.8	N/F
Skein	3.2	N/A	-30.3	N/A
B. DSP Units and Block Memories:				
SHA-2	23.4	N/A	19.5	N/A
C. Block Memories:				
(i) AES Tables				
ECHO	21.1	N/F	38.9	N/F
Fugue	18.0	37.5	34.2	55.6
Groestl	27.3	54.2	63.5	79.3
SHAvite-3	29.3	50.0	49.6	65.9
(ii) Message Expansion Tables				
BLAKE	55.3	57.5	47.8	55.7
Hamsi	25.2	23.9	24.0	20.0
(iii) Round Constant Tables				
JH	1.9	9.4	14.7	17.2
Keccak	2.3	0.5	-1.3	0.3

For high-speed families, we were successful in reducing the amount of reconfigurable logic resources in 26 out of 28 (92.8%) hash function-FPGA family pairs. For low cost families, the equivalent percentage was 100 (16 out of 16 relevant pairs). Biggest savings can be observed in BMW, SIMD, all AES based functions (ECHO, Fugue, Groestl and Shavite-3), as well as in BLAKE and Hamsi. Here, the only exceptions are CubeHash and Skein, DSP Units, when implemented in Stratix III.

In SIMD, multiplications consume large chunk of logic resources. Big improvement for SIMD in Xilinx and Altera FPGAs results from efficiently utilizing embedded DSP multipliers. BMW architecture involves multi-operand addition. In order to best exploit the potential of DSP units, they are used in cascaded mode. This mode enables the designer to take third input from adjacent DSP unit. This feature reduces the total number of adders required in multi-operand addition, and thus improves the ratio of the number of DSP units added to the number of logic resources saved.

Finally, Skein hash function with 64-bit addition exposes a lack of good support for this precision in Altera Stratix III. In order to perform a 64-bit addition, the DSP unit needs to be supported with extra logic implemented using regular reconfigurable logic resources.

AES-based functions, in both S-box and T-box architectures, resulted in much bigger area reduction because the functions implemented using embedded resources are a big part of the entire hash function circuit. In case of functions using round constant tables (JH, Keccak), the relative improvement is not significant because these tables are relatively small.

In Table VII, we illustrate overall throughput to area results across all families (Virtex 5, Spartan 3, Stratix III and Cyclone II). It was our primary goal to optimize Throughput to Area ratio. For high-speed families, we obtained an improvement for 19 out of 28 (67.8%) hash function-FPGA family pairs. For low-cost families, we saw an improvement for 13 out of 16 (81.2%) pairs.

In the category Block memories, similar improvement was shown for 92.15% pairs. Biggest achievers are *AES and Message Expansion* based function with 100% improvement in each case. Results of AES based functions really stand out with biggest potential of improvement. These benefit from throughput improvement for low-cost families (see Table V), and area reduction for all families (see Table VI), thus scoring highest in Throughput to Area Ratios (see Table VII).

In order to evaluate the usefulness of embedded resources, we also calculated the *"Resource Replacement Ratio"* (see Table III). This ratio represents a relative benefit from conversion of logic to embedded resources. Results depict that BMW, SIMD in category *"DSP Units"*, all AES based functions (ECHO, Fugue, Groestl and Shavite-3), and BLAKE in category *"Block Memories"* give a largest benefit in replacing logic resources with embedded resources.

TABLE VII: Relative Improvement in the Throughput to the Amount of Reconfigurable Resources ratio across four FPGA families. Notation: N/A not applicable, N/F not fitting

Algorithm	Virtex 5	Spartan 3	Stratix III	Cyclone II
A. DSP Units:				
BMW	36.8	N/A	-52.2	N/A
CubeHash	-16.0	N/A	-59.9	N/A
Shabal	-21.1	-31.4	-47.8	-26.3
SIMD	41.5	N/F	24.4	N/F
Skein	-5.0	N/A	-53.5	N/A
B. DSP Units and Block Memories:				
SHA-2	34.1	N/A	21.8	N/A
C. Block Memories:				
(i) AES Tables				
ECHO	2.3	N/F	27.2	N/F
Fugue	12.9	82.0	41.6	177.9
Groestl	6.9	128.8	82.0	410.5
SHAvite-3	32.9	151.5	129.0	50.1
(ii) Message Expansion Tables				
BLAKE	81.0	63.6	61.7	33.8
Hamsi	16.8	20.5	32.4	22.1
(iii) Round Constant Tables				
JH	-10.4	10.9	22.1	4.8
Keccak	-14.4	-10.6	-0.4	0.2

VII. CONCLUSIONS

Fourteen 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. They involve addition (32-bit, 64-bit, and multi-operand), multiplication, and use of block memories (AES, Message Expansion, and Round Constant based tables). As a result, it is a non-trivial problem to generalize the usefulness of embedded resources across all FPGA families (Virtex 5, Stratix III, Spartan 3 and Cyclone II).

We employed criteria like Best Throughput to Area Ratio and Resource Replacement Ratio in different categories and across FPGA families to comprehend the usefulness of embedded resources. Our results demonstrate significant savings in the amount of reconfigurable logic, especially high for functions based on large look-up tables, such as four AES-based candidates, as well as BLAKE and Hamsi. 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.

Future designers interested in using embedded resources do need to consider right FPGA family selection for their implementations because FPGA vendors have different features and architectures for embedded resources. Our results show a significant difference in performance across FPGA families.

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. Güneysu 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] ATHENa Project Website, 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] Federal Information Processing Standard Publication (FIPS PUB) 180-3, Secure Hash Standard (SHS), October 2008 (FIPS 180-3)

[15] T. Güneysu, "Utilizing hard cores of modern FPGA devices for high-performance cryptography", J. Cryptographic Engineering, Vol. 1. 2011, pp. 37-55.

[16] V. Fischer and M. Drutarovsky, Two methods of Rijndael implementation in reconfigurable hardware, in Proc. CHES 2001, Paris, France, May, 2001, pp. 81-96.

Appendix

Table A.1. Timing characteristics and resource utilization for basic architectures and architectures based on the use of embedded FPGA resources for implementations of 13 Round 2 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 #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 [#ALUTs, #Mem-bits, #DSPs]	Tp/#ALUTs	Δ Tp [%]	Δ #ALUTs [%]	Δ Tp/#ALUTs [%]	Resource Replacement Ratio
DSP Units									
BMW	Basic	12.5	6408	1254,0,0	0.51	-62.7	22.0	-52.2	6.2 ALUTs/DSP
	Embedded	4.6	2389	9783, 0, 448	0.24				
CubeHash	Basic	233.1	3730	1919,0,0	1.94	-55.3	-11.5	-59.9	-3.4 ALUTs/DSP
	Embedded	104.3	1668	2139, 0, 64	0.78				
Shabal	Basic	202.2	1618	744,0,0	2.17	-54.6	13.2	-47.8	4.9 ALUTs/DSP
	Embedded	91.7	734	646, 0, 20	1.14				
SIMD	Basic	55.3	3146	25434,0,0	0.12	-1.4	20.8	24.4	41.6 ALUTs/DSP
	Embedded	54.5	3101	20150, 0, 127	0.15				
Skein	Basic	90.2	2432	4380,0,0	0.56	-39.5	-30.3	-53.5	-10.4 ALUTs/DSP
	Embedded	54.7	1472	5705, 0, 128	0.26				
DSP Units and Block Memory									
SHA-2	Basic	209.9	1654	988,0,0	1.67	-2.0	19.5	21.8	N/A
	Embedded	205.8	1621	795, 2048,16	2.03				
Block Memory									
(i) AES Based Tables									
ECHO	Basic	235.4	13910	20754,0,0	0.67	-22.3	38.9	27.2	15.8 ALUTs/1kbit Mem
	Emb (S-box)	175.9	10802	12675, 512k, 0	0.85				
Fugue	Basic	202.5	3241	2391,0,0	1.36	-6.8	34.2	41.6	8.3 ALUTs/1kbit Mem
	Emb (T-box)	188.9	3022	1574, 99k, 0	1.92				
Groestl	Basic	286.7	6990	6260,0,0	1.12	-33.5	63.5	82.0	7.5 ALUTs/1kbit Mem
	Emb (T-box)	190.7	4650	2288, 528k, 0	2.03				
SHAvite-3	Basic	238.1	3295	2930,0,0	1.12	15.4	49.6	129.0	5.7 ALUTs/1kbit Mem
	Emb (T-box)	275.0	3804	1477, 256k,0	2.58				
(ii) Message Expansion Tables									
BLAKE	Basic	121.3	2958	3637,0,0	0.81	-15.6	47.8	61.7	144.8 ALUTs/1kbit Mem
	Embedded	102.4	2498	1900, 12k, 0	1.31				
Hamsi	Basic	286.8	3060	2304,0,0	1.33	0.6	24.0	32.4	30.8 ALUTs/1kbit Mem
	Embedded	288.6	3078	1750, 18k, 0	1.76				
(iii) Round Constant Tables									
JH	Basic	383.0	5299	3526,0,0	1.50	4.2	14.7	22.1	57.4 ALUTs/1kbit Mem
	Embedded	388.3	5523	3009, 9k, 0	1.83				
Keccak	Basic	303.2	13746	4221,0,0	3.26	1.2	-1.3	-0.1	-28.0 ALUTs/1kbit Mem
	Embedded	306.9	13913	4277, 2k, 0	3.25				