A Zynq-based Testbed for the Experimental Benchmarking of Algorithms Competing in Cryptographic Contests

Farnoud Farahmand, Ekawat Homsirikamol, and Kris Gaj
George Mason University
Fairfax, Virginia 22030
Email: {ffarahma, ehomsiri, kgaj}@gmu.edu

Abstract—**Hardware performance evaluation of candidates competing in cryptographic contests, such as SHA-3 and CAESAR, is very important for ranking their suitability for standardization. One of the most essential performance metrics is the throughput, which highly depends on the algorithm, hardware implementation architecture, coding style, and options of tools. The maximum throughput is calculated based on the maximum clock frequency supported by each algorithm. A common way of determining the maximum clock frequency is static timing analysis provided by the CAD toolsets such as Xilinx ISE, Xilinx Vivado, and Altera Quartus Prime. In this project, we have developed a universal testbed, which is capable of measuring the maximum clock frequency experimentally, using a prototyping board. We are targeting cryptographic hardware cores, such as implementations of SHA-3 candidates. Our testbed is designed using a Zynq platform and takes advantage of software/hardware co-design. It supports two separate clock domains, one for a hardware module under test, and the other for the communication between an ARM core and hardware accelerator. We measured the maximum clock frequency and the execution time of 12 Round 2 SHA-3 candidates experimentally on ZedBoard and compared the results with the frequencies reported by Xilinx Vivado. Our results indicate that depending on the characteristics of each algorithm, we may achieve either much higher or the same experimental frequency than the results reported by the tools using static timing analysis. This behavior is then further analyzed, and the relevant conclusions drawn.**

I. Introduction

In November 2007, NIST announced a public competition to develop a new cryptographic hash function standard, called SHA-3. About four years have been spent on evaluating candidates submitted to this contest in terms of security, software and hardware efficiency, simplicity, and flexibility. Fifty one candidates were qualified to the first round of the contest. Their number was reduced first to fourteen and then to five candidates, in the second and the third round of the competition, respectively. Majority of candidates qualified to Round 2 were judged to have adequate security, and thus their performance in software and hardware became a decisive factor. Throughput, area, and throughput to area ratio were the most important metrics used for hardware evaluation. Throughput of a hash function is the number of message bits for which a hash value (digest) can be computed per unit of time. In hardware, the maximum throughput depends on the maximum clock frequency supported by each algorithm, the block size, and the number of clock cycles required to process a block. Maximum clock frequency that can be achieved by a hardware implementation can be estimated or measured at different stages of the design process. The main stages are synthesis, placing and routing (P&R), and actual implementation on the board. The post-synthesis and post place & route results are determined by the FPGA tools using static timing analysis. Hardware evaluation of 14 round 2 SHA-3 candidates, based on post-placing and routing results, is reported in [1].

In this paper, we demonstrate that the interface of Hash Core proposed in [1] can be easily combined with the de-facto industry standard, AMBA AXI [2], in order to achieve the practical, industry-grade designs for hardware accelerators implemented using reconfigurable logic of All Programmable Systems on Chip (SoC), such as Zynq. We also investigate the communication overhead introduced by the transfer of data between these hardware accelerators and the microprocessor core, for various sizes of data inputs. Secondly, we explore how the maximum clock frequency reported by Xilinx Vivado and the actual frequency measured experimentally compare to each other. Thus, we are verifying the accuracy of the worst-case values of the maximum clock frequency, reported by static timing analysis. Our expectation is that the experimental clock frequency should be greater or equal than the worst-case value returned by the tools for all investigated algorithms. Finally, we also compare the ratios of the maximum experimental clock frequency to the maximum frequency returned by the tools. The straightforward expectation could be that these ratios should be approximately the same for all algorithms, as any given instance of the Zynq device is likely to operate with a frequency higher by a certain specific percentage than the worst case instance of the same integrated circuit. However, as shown in this paper, this naive expectation appears to be very far from the true behavior of a particular Zynq device observed for various investigated algorithms.

It should be stressed that we do not advocate replacing the well-established benchmarking methodology based on post-place and route results with the comparison of experimental values, as these values are more accurate only for a specific instance of the Zynq device, and cannot be generalized to

This work is supported by NSF Grant #1314540

978-1-5090-3707-0/16/$31.00 ©2016 IEEE

millions of similar devices operating in the field, affected to a different extant by variations in the fabrication process. Our study should be used just to confirm (or question) the worst-case values returned by the tools, which are likely to better represent the entire population of Zynq devices fabricated in a given technology.

VHDL implementation of all Round 2 SHA-3 candidates is available at [3]. We selected 12 of them and applied our testing environment to these algorithms. BMW (Blue Midnight Wish) and SIMD were excluded due to poor performance. Additionally, the BMW module has two different clock domains for the i/o interface and the main hash core, which substantially complicates experimental testing. The results generated for each algorithm, collected using our testbed, included: 1) hardware execution time for various input sizes, 2) software execution time, 3) HW/SW speed up, 4) maximum clock frequency after placing and routing (obtained using Vivado Design Suite), and 5) maximum experimental clock frequency.

II. Previous Work

Experimental benchmarking of cryptographic algorithms has been performed previously on different platforms other than Zynq. In [4], maximum frequency of SHA-256 has been measured experimentally using the SLAAC-1V board based on Xilinx Virtex VCV 1000. In [5] and [12], an experimental measurement of the hardware performance of 14 round 2 SHA-3 candidates is performed using the SASEBO-GII FPGA board. The investigated implementations are run at their maximum clock frequency reported by the CAD tool. Hence, no investigation of higher clock frequencies is performed.

In [6], the experimental evaluation of SHA-2 and five Round 3 SHA-3 Candidates was performed using a standard-cell ASIC realized using 65nm CMOS technology. The authors combined two sets of implementations, developed using different performance targets, and implemented them on one ASIC, with a common input/output (I/O) interface. Area, throughput, and throughput to area ratio were generated for all of these algorithms. Synthesis was performed using Synopsys Design Compiler, and placing & routing using Cadence Encounter Digital Implementation. All cores have been demonstrated to operate at the experimental clock frequencies from 15% to 92% higher then the worst case estimates returned by the placing & routing tools.

In [7], the throughput and power results, from the experimental evaluation of SHA-3 finalists, using 130nm ASIC technology, are reported. SASEBO-GII FPGA board is used as a controller. The obtained results indicated that the measured throughput was always lower than the Post-layout results, with the difference less than 30%.

In [8], a comprehensive evaluation of all Round 2 SHA-3 candidates, based on post-layout reports, using the 90 nm ASIC technology, is performed. The post-layout results were reported for two target throughputs, 20 Gbps and 0.2 Gbps, and the corresponding results compared against each other. No experimental measurements were performed as a part of this work. Similarly, in [9], post-layout throughputs are reported for all Round 2 candidates, using the ASIC 180 nm technology, and no experimental measurements are reported.

III. Design & Verification

A. System Design

The Zynq-7020 All-Programmable System on Chip (SoC) has been selected as our target device, due to its high performance and flexibility. In particular, our testbed takes advantage of software/hardware co-design and the memory management provided by the ARM platform. For each algorithm, the optimized C implementation [10] is first run on the ARM core and its performance recorded. As a result, after the conclusion of the hardware measurements, hardware vs. software speed up can be easily and fairly evaluated. Our system, composed of three major parts, is shown in Fig. 1:

*1) **Processing System (PS):*** The Processing System contains two Cortex A9 ARM Processor cores, and related logic. The HP (High Performance) ports are used for communication between PS and Programmable logic (PL). The input data (message) is sent to the hash core, located in PL. After the calculations are completed, an interrupt is generated, and the hash value is transferred back to the ARM core.

*2) **Interconnects:*** Two AXI Interconnect IPs take care of the data transmission between PS and PL using the memory mapped mode, AXI Full. These IPs are added and configured automatically using Vivado Design Suite.

*3) **Programmable Logic (PL):*** Programmable Logic is used to implement the following major submodules:

a) `Input FIFO`: Input FIFO can be written to using AXI Stream Slave (AXIS) interface and read from using FWFT FIFO (First Word Fall Through FIFO) interface. It supports independent read and write clock domains. It was designed specifically for the purpose of this project, and its interface is shown in Fig. 2.

b) `Hash Core`: Hardware implementation of a hash algorithm, compliant with the top-level interface and communication protocol described in [1]. This module can be replaced with any other hardware accelerator that can communicate with the FWFT FIFO interface.

c) `Output FIFO`: Output FIFO can be written to using the FWFT FIFO interface and read from using AXIS interface. It is also capable of handling independent clock domains for reading and writing. Moreover, it has an AXI Lite interface for configuring the transfer length and start delay information. Transfer length indicates the number of output words to be sent using the AXI Stream interface. Start delay lets the user to specify the delay, in clock cycles, before this module starts transferring the output data back to the processor. Delay countdown starts when the output is ready to send. Output FIFO was designed specifically for the purpose of this project, and its interface is shown in Fig. 3.

d) `Clocking Wizard`: Clocking Wizard was added from the Xilinx IP catalog [11] and is capable of generating a clock signal with a variable frequency, configurable from software. This module can be controlled using the AXI Lite interface and let us change the clock frequency on the fly.

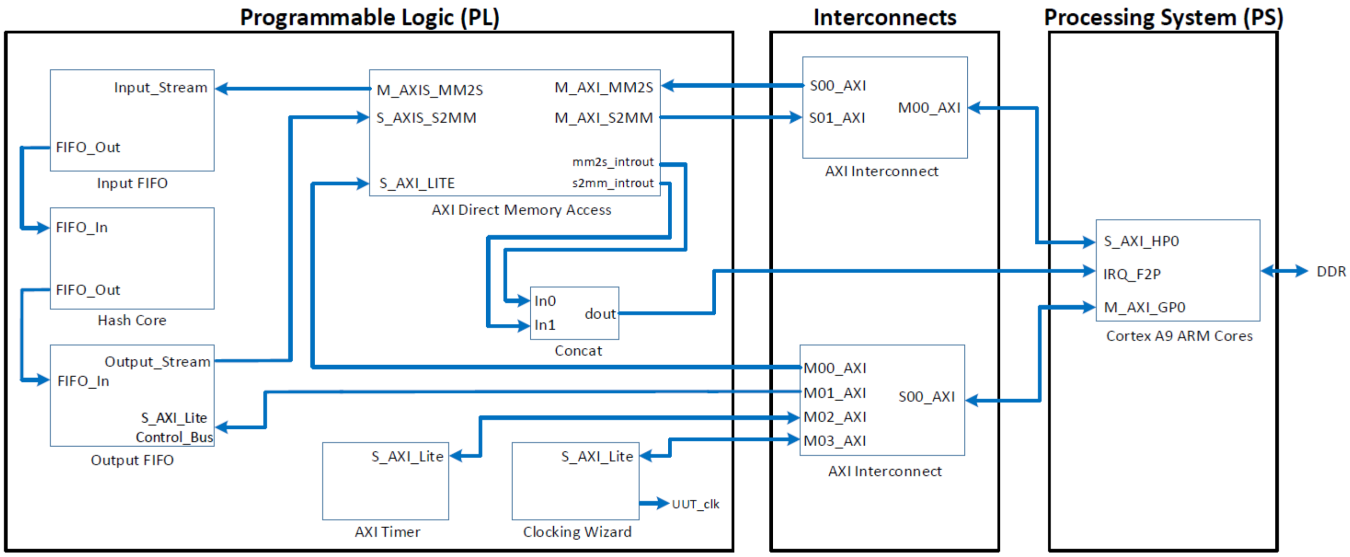


Fig. 1. Block Diagram of the Testbed with the division into Programmable logic (PL), Interconnects, and Processing System (PS)

e) AXI Direct Memory Access (DMA): AXI DMA was added to block design from the Xilinx IP catalog [11], and converts the stream transaction protocol to the memory mapped protocol. As a result, it allows the hardware accelerator to read from and write to the DDR memory. The operation of this module is fully configurable from software, and frees the ARM processor to perform other tasks.

f) AXI Timer: AXI Timer is also a standard unit, available in the Xilinx IP catalog. It is capable of performing the execution time measurements for software and hardware implementations of various functions, with the accuracy of a single clock cycle of a system clock (by default: 10 ns).

g) Concat: The Concat module is used to concatenate two input signals and produce a single output, active when any of the two inputs is active. In the circuit from Fig. 1, it is used to create an interrupt to PS, active when either an input transfer or an output transfer is completed by AXI DMA.

In case a single clock domain was used, the maximum clock frequency supported by our testbed would be limited by the maximum clock frequency of the AXI DMA and other standard IP components. To overcome this limitation, our testbed supports two separate clock frequencies, one for communication and auxiliary modules (AXI DMA, AXI Timer, etc.) and the other for our dedicated hash core. The system clock frequency, used for communication between PL and PS, is fixed. At the same time, the Clocking Wizard is used to generate the second clock, with a variable clock frequency, set on the fly, under the software control. Multiple frequencies of the second clock are then used during the binary search for the maximum frequency supported by a given hash core.

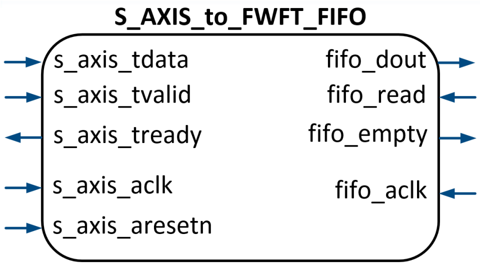


Fig. 2. Input FIFO

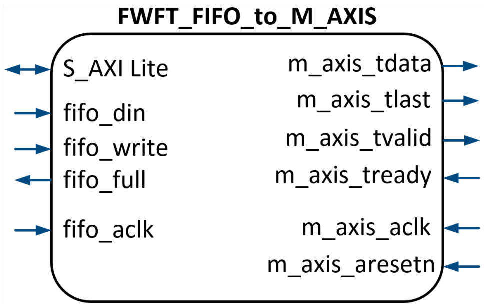


Fig. 3. Output FIFO

A simplified diagram of our testbed, with two independent clocks, is shown in Fig. 4. In this testbed, at first AXI DMA, AXI Timer, and Interrupts are initialized. Then, Output FIFO is configured with the desired transfer length and start delay, and Clocking Wizard is configured with the UUT (Unit Under Test) output clock frequency. All initializations and configurations are done through the AXI Lite interface. Afterward, a buffer is allocated in DDR memory to store input data (message) of a certain size. Software implementation of the corresponding hash algorithm is run on ARM core. The software execution time is measured using AXI Timer. AXI Timer is started before the hash algorithm function call. Next, the message is transferred from the DDR memory through AXI DMA to Input FIFO. The Hash Core starts reading data from Input FIFO and writes the corresponding hash value to

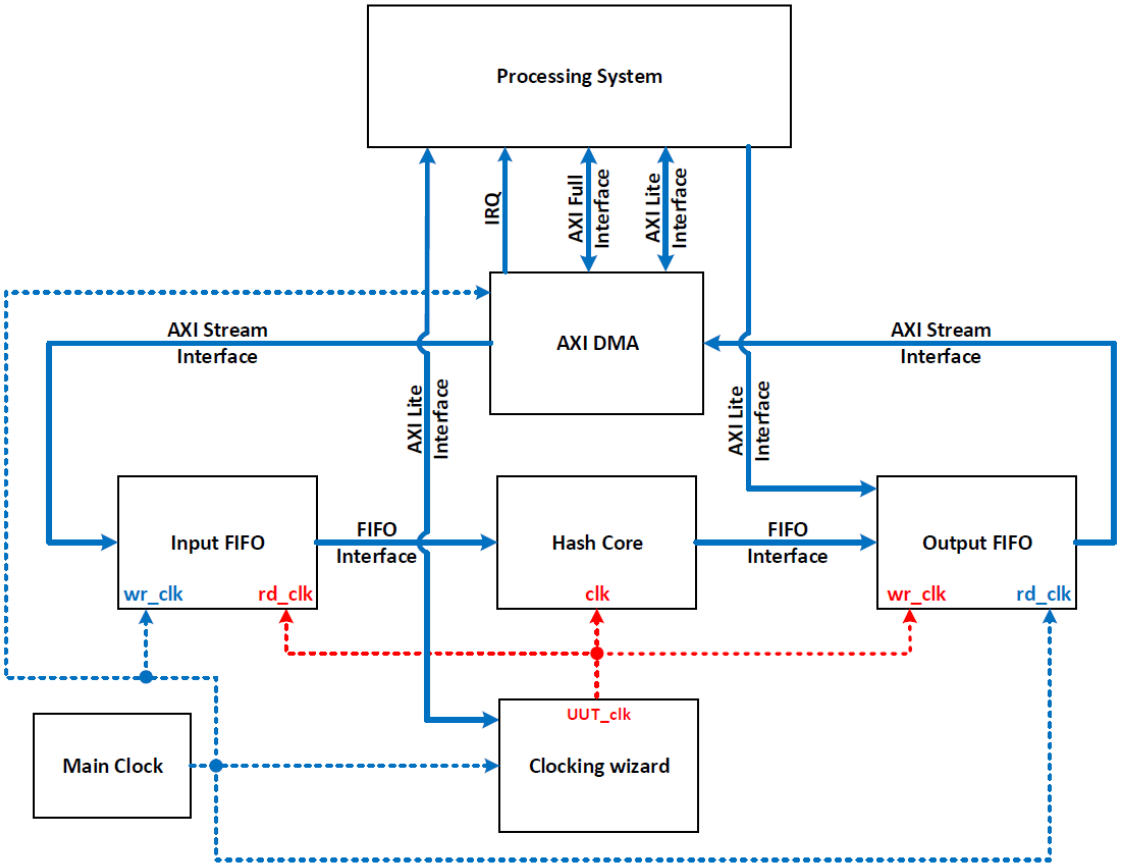


Fig. 4. Simplified block diagram of the PL side with the indication of two independent clocks

Output FIFO. Then, Output FIFO sends back the result to another buffer in the DDR memory through its AXI-Stream interface and AXI DMA. After the end of this transfer, the AXI Timer is stopped.

The entire end-to-end data transfer time and the hardware execution time are measured together using AXI Timer. The hash value received from the hardware side is compared with the value calculated by software. If they are equal, we increase the frequency of Hash core using Clocking Wizard, and rerun the entire process again. Otherwise, we decrease the frequency. The aforementioned procedure is repeated multiple times using the rules of binary search. The process ends only when we find the maximum clock frequency achievable by each hash algorithm with the precision of 0.1 MHz. The maximum experimental clock frequency, software execution time, hardware execution time at the maximum clock frequency, and the speed up (hardware vs. software) is reported.

B. *Verification methodology*

A universal testbench has been developed in the Vivado environment to verify the operation of our testbed using simulation. Apart from the circuit under test (composed of the Input FIFO, Hash Core, and Output FIFO), this testbench includes three AXI Traffic Generators (ATG) and one FIFO to simulate the functionality of the Zynq PS and AXI DMA.

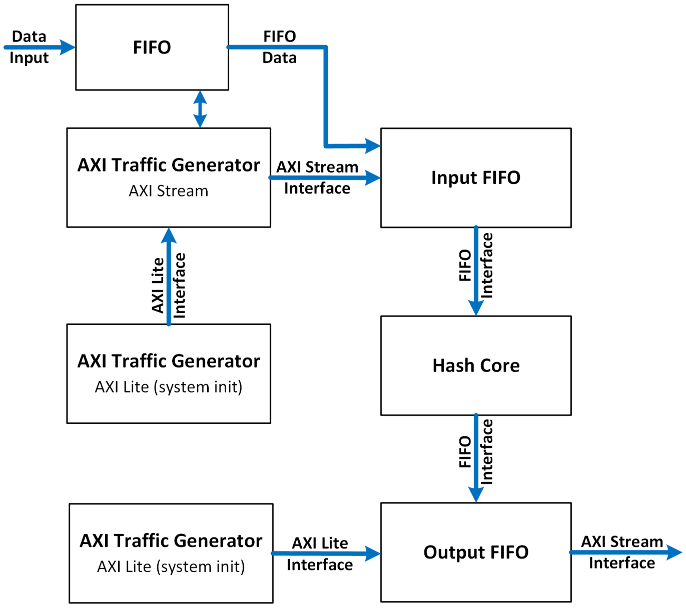


Fig. 5. Universal testbench for Vivado environment

A simplified block diagram of the testbench is shown in Fig. 5. The first ATG module is used in the AXI Stream mode to provide the control signals of the AXI Stream Interface to the Input FIFO module. Data that is provided by this ATG, configured in the AXI Stream mode, is random data. Therefore, an additional FIFO is included in the testbench to provide a source of desired input data. The second ATG module configures the first ATG module (AXI Stream ATG) with the desired configuration data, such as length of transaction, programmable delay, and the number of transactions, through AXI Lite interface. The third ATG module, configured in the AXI Lite mode, is used to configure Output FIFO. All AXI Traffic generators are preloaded with appropriate configurations, using the configuration COE files (address and data). At first the FIFO is filled with the input data. Then, all AXI Traffic Generators are started. The AXI Stream ATG and FIFO provide the input message through the AXI Stream interface to the Input FIFO module. Hash Core reads data from Input FIFO, calculates hash value, and transfers it to the Output FIFO module. Eventually, we can compare the received hash value with the expected result to verify the functionality of the design.

IV. Result

ZedBoard and Vivado 2015.4 have been used for result generation. All options of Vivado design suite including synthesis and implementation settings are set to default mode. On the software side, the bare metal environment and Xilinx SDK are used for running the C code on the ARM core of Zynq. The frequency of the primary system clock, connected to the interface IPs, including AXI DMA, AXI Timer, etc., is set to 100 MHz. Clocking Wizard generates the second clock, under the control of the C program, based on binary search. The frequency of Dual Core ARM (PS) is set to 667 MHz.

A. *Maximum Frequency*

Fig. 6 illustrates the maximum clock frequency achieved using static timing analysis and experimental testing, respectively, for each of the investigated algorithms. For all algorithms the experimental clock frequency is higher than that returned by static timing analysis. This result is expected, as CAD Tools alway take into account the worst case scenario, and thus report the pessimistic estimates in terms of speed. In particular, during any particular test, the critical path is not always triggered, even for a relatively long input. Additionally, a particular device used for testing (i.e., a particular instance of Zynq-7020 in our case) tends to have average rather than worst-case timing characteristics.

At the same our analysis reveals significant differences among the behavior of various algorithms. BLAKE, CubeHash, SHAvite-3, and Skein have an experimental frequency higher than the post-place & route frequency by 80 to 100%. In the second group, Fugue, JH, and Luffa, have the frequency higher by 55 to 65%. The third group includes Grøstl, Hamsi, and Keccak, and is characterized by the frequency improvement factor between 19 and 30%. Finally, ECHO and Shabal have almost identical frequencies returned by the static timing analyzer and the experimental test.

TABLE I
Detailed result for Maximum Frequencies and Throughputs

Algorithm	Max Freq. Static Timing Analysis [MHz]	Max Freq. Experimental [MHz]	Throughput Based on Formula and Max Exp. Freq. [Gb/s]	Throughput Based on Exp. HW Exe. Time [Gb/s]
BLAKE	76.4	145.4	3.546	3.544
CubeHash	152.9	275.8	4.413	4.399
ECHO	100.1	101.1	5.999	6.000
Fugue	122.9	200.0	3.200	3.191
Grøstl	197.2	258.6	6.305	5.821
Hamsi	105.0	124.9	1.333	1.332
JH	211.6	333.3	4.740	4.726
Keccak	102.6	123.1	5.292	5.314
Luffa	152.5	247.4	7.037	7.213
Shabal	119.7	122.5	0.981	0.983
SHAvite-3	119.0	205.7	2.846	2.828
Skein	70.6	140.3	3.782	3.772

The algorithms belonging to the same group seem to have little in common in terms of basic operations, area requirements, or absolute value of the post-place and route frequency. As a result, the most likely explanation seems to be the placement of the respective designs in different locations on the chip, affected to a different extant by parameter variations.

Table I shows detailed values of the maximum clock frequency obtained from static timing analysis and experimental testing. Maximum experimental clock frequency was determined as a worst case value across all investigated input sizes from 10 to 5000 kB. The fourth column, contains the Throughput based on the formulas for the Throughput of each algorithm, listed in Table II, with T replaced by the inverse of the Maximum Experimental Clock Frequency. The fifth column is the Throughput obtained by dividing the message input size by the actual execution time of hashing in hardware, measured using AXI Timer for the input size equal to 1000 kB.

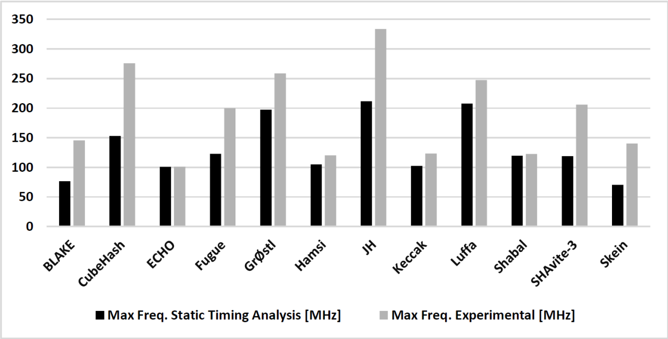
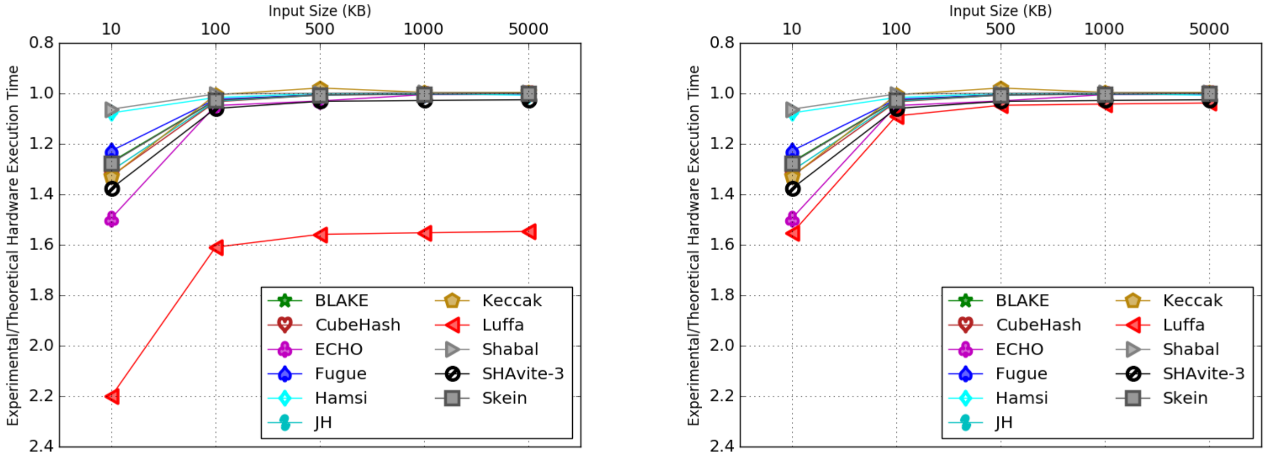


Fig. 6. Maximum clock frequencies obtained using static timing analysis and the experimental measurement, respectively

B. *Data transaction overhead*

Fig. 7(a) shows the ratio of the hardware execution time measured using AXI Timer (including any communication overhead) over the calculated hardware execution time (using



(a) DMA core running at 100 MHz for all algorithms

(b) DMA core running at 150 MHz in case of Luffa and 100 MHz for all other algorithms

Fig. 7. Ratio of the Experimental (Measured) Hardware Execution Time to the Theoretical (Calculated) Hardware Execution Time for different input sizes (expressed in kilobytes)

formulas provided in [1] and in Table II), determined for different input message sizes. As we can see, for majority of investigated algorithms, such as BLAKE, CubeHash, ECHO, etc., the trend is very similar. The overhead of the communication between the PS and PL is between 20 and 50% for the messages of the size of 10 kB. For Shabal the respective value is about 5%. This overhead decreases below 5% for messages greater or equal to 100 kB. For 500 kB and larger messages, the relative overhead is very small and the ratio is almost 1. Luffa is the only exception to this trend. For this algorithm, the ratio is 2.2 for 10 kB of data and it decreases to 1.6 for 100 kB messages. For 500 kB and larger inputs, the ratio is stable around 1.55. Thus, the overhead is very substantial. The reason for this unusual behavior is that Luffa is the only algorithm which has the Hash Core throughput exceeding the throughput of AXI DMA (6.4 Gbit/s at 100 MHz clock frequency). As a result, DMA core fails to feed the Hash Core with enough data to maintain the maximum possible throughput. Since the maximum clock frequency supported by the DMA Core on Zynq-7020 is 150 MHz [11], we can increase the DMA frequency to overcome this issue. Fig. 7(b) depicts the updated graph with new result for Luffa obtained using AXI DMA running at 150 MHz instead of 100 MHz. As illustrated by the graph, the DMA throughput bottleneck was completely eliminated and the behavior of Luffa is similar to that of other algorithms.

In addition, we repeated the same experiment using a second ZedBoard in order to demonstrate the effect of manufactoring variations on the maximum experimental clock frequency (and thus all other experimental timing measurements). In Table III, we list the maximum frequency achieved using the first and second board, average maximum frequency, and standard deviation. It can be observed that the second board has slightly better results than the first one for the majority of algorithms and the same results for the remaining ones.

TABLE II
THE I/O DATA BUS WIDTH (IN BITS), HASH FUNCTION EXECUTION TIME (IN CLOCK CYCLES), AND THROUGHPUT (IN GBITS/S) FOR THE 256-BIT VARIANT OF SHA-3 CANDIDATES. T DENOTES THE CLOCK PERIOD IN NS AND N INDICATES THE NUMBER OF INPUT BLOCKS.

Algorithm	**I/O Bus width**	**Hash Time [cycles]**	**Throughput [Gbit/s]**
BLAKE	64	$2+8+21 \cdot N+4$	$512/(21 \cdot T)$
CubeHash	64	$2+4+16 \cdot N+160+4$	$256/(16 \cdot T)$
ECHO	64	$3+24+26 \cdot N+1+4$	$1536/(26 \cdot T)$
Fugue	32	$2+2 \cdot N+37+8$	$32/(2 \cdot T)$
Grøstl	64	$3+21 \cdot N+4$	$512/(21 \cdot T)$
Hamsi	32	$3+1+3 \cdot (N-1)+6+8$	$32/(3 \cdot T)$
JH	64	$3+8+36 \cdot N+4$	$512/(36 \cdot T)$
Keccak	64	$3+17+24 \cdot N+4$	$1088/(24 \cdot T)$
Luffa	64	$3+4+9 \cdot N+9+1+4$	$256/(9 \cdot T)$
Shabal	64	$2+64 \cdot N+64 \cdot 3+16$	$512/(64 \cdot T)$
SHAvite-3	64	$3+8+37 \cdot N+4$	$512/(37 \cdot T)$
Skein	64	$2+8+19 \cdot N+4$	$512/(19 \cdot T)$

C. *Hardware/Software execution time speed up*

Table IV shows HW/SW speed up for 5 different input sizes and all evaluated algorithms. As we can see, BLAKE and Shabal demonstrate speed up below 100. Hamsi, Luffa, Skein, ECHO, SHAvite-3 and Fugue achieve the speed-up from 100 to about 620. For CubeHash, Keccak and JH the speed-up exceeds 2,000, and for JH it is over 20,000. All of the aforementioned algorithms have almost stable results for messages larger than 100 kB. For 10 kB, the majority of them show the decrease in the speed-up caused by the communication overhead between software and hardware.

TABLE III
EXPERIMENTAL MAXIMUM FREQUENCY RESULT ON 2 DIFFERENT ZEDBOARD

Algorithm	Max Freq. Exp. Board1 [MHz]	Max Freq. Exp. Board2 [MHz]	Avr. Max Freq. Exp. [MHz]	Std. Dev. Max Freq. Exp. [MHz]
BLAKE	145.4	153.8	149.6	6
CubeHash	275.8	296.3	286.0	14
ECHO	101.1	101.1	101.1	0
Fugue	200.0	200.0	200	0
Grøstl	258.6	258.6	258.6	0
Hamsi	120.2	124.2	122.2	3
JH	333.3	347.8	340.5	10
Keccak	123.1	123.1	123.1	0
Luffa	247.4	262.5	254.9	11
Shabal	122.5	140.0	131.25	12
SHAvite-3	205.7	218.7	212.2	9
Skein	140.3	148.1	144.2	6

TABLE IV
HW/SW SPEED UP FOR 5 DIFFERENT INPUT SIZES IN KB

Input Size (KB)	**10**	**100**	**500**	**1000**	**5000**
BLAKE	73	87	88	89	89
CubeHash	3,326	4,105	4,165	4,181	4,184
ECHO	287	384	397	399	400
Fugue	127	119	119	119	119
Hamsi	581	619	624	624	625
JH	19,155	24,109	24,639	24,712	24,776
Keccak	2,341	3,079	3,169	3,182	3,192
Luffa	302	398	410	411	412
Shabal	6	6	6	6	6
SHAvite-3	81	102	104	105	105
Skein	92	112	114	115	115

V. CONCLUSIONS

We have developed a novel experimental testbed, based on Zynq All Programmable System on Chip, for evaluating hardware performance of cryptographic algorithms competing in cryptographic contests, such as SHA-3, CAESAR, etc. This testbed allows determining the maximum experimental clock frequency of each core, using binary search, with the accuracy of 0.1 MHz. The operation of each hash core and surrounding FIFOs, can be first verified through simulation, and then tested experimentally using ZedBoard. The testbed can be used to correctly measure performance of designs with the maximum throughput of 64 bit · 150 MHz = 9.6 Gbit/s.

The correct operation of the testbed was demonstrated using the implementations of 12 Round 2 SHA-3 Candidates. For all these hash functions, the overhead of the communication between PS and PL was below 5% for 100 kB messages and negligible for messages above 500 kB. All algorithms have also demonstrated significant speed up vs. their execution in software on the same chip, in spite of the substantial speed of the ARM core, operating at 667 MHz. Our experiments have also demonstrated that the maximum experimental clock frequency was always higher than the post-place and layout frequency calculated by Vivado, using static timing analysis. This fact demonstrates that the tool correctly returns the worst-case boundries, not likely to be reached in practice. At the same time, somewhat unexpectedly, the spread of ratios experimental to post-place and route frequency is very large, ranging from 1 to 2. This fact can be explained by a different influence of parameter variations, on the critical path of the each hash core, due to a different physical location (placement) of these critical paths in the FPGA fabric.

Our future work will involve an attempt at further explanation of the observed differences among various algorithms. We will also extend our environment to handle other types of cryptographic transformations, such as authenticated ciphers and post-quantum public key cryptosystems. Finally, we will also investigate at the use of other types of prototyping boards, including the FPGA boards with the PCI Express interface.

REFERENCES

[1] E. Homsirikamol, M. Rogawski, and K. Gaj, ”Comparing Hardware Performance of Fourteen Round Two SHA-3 Candidates Using FPGAs,” Cryptology ePrint Archive, Rep. 2010/445, Aug. 2010.

[2] ARM. AMBA Specifications. [Online]. Available: http://www.arm.com/products/system-ip/amba-specifications.php

[3] GMU Source Code. (2015, Apr. 15). George Mason University. [Online]. Available: https://cryptography.gmu.edu/athena/index.php?id=source_codes, Accessed Oct. 10, 2016.

[4] D. Fedoryka, ”Fast Implementation of the Secure Hash Algorithm SHA-256 in FPGA”, Master’ Thesis, George Mason University, July 2004.

[5] K. Kobayashi, J. Ikegami, M. Knežević, X. Guo, S. Matsuo, S. Huang, L. Nazhandali, U. Kocabas, J. Fan, A. Satoh, I. Verbauwhede, K. Sakiyama, K. Ohta, ”A Prototyping Platform for Performance Evaluation of SHA-3 Candidates,” IEEE International Symposium on Hardware-Oriented Security and Trust (HOST 2010), pp. 60-63, Jun. 2010.

[6] F. K. Gürkaynak, K. Gaj, B. Muheim, E. Homsirikamol, C. Keller, M. Rogawski, H. Kaeslin, and J. P. Kaps ”Lessons Learned from Designing a 65 nm ASIC for Evaluating Third Round SHA-3 Candidates”, The Third SHA-3 Candidate Conference, Washington DC, 2012. Available: http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/March2012/Program_SHA3_March2012.html

[7] X. Guo, M. Srivastav, S. Huang, D. Ganta, M. B. Henry, L. Nazhandali, and P. Schaumont, ”ASIC implementations of five SHA-3 finalists,” 2012 Design, Automation & Test in Europe Conference & Exhibition (DATE), Mar. 12-16, 2012, pp. 1006-1011.

[8] L. Henzen, P. Gendotti, P. Guillet, E. Pargaetzi, M. Zoller, and F. Gürkaynak. ”Developing a hardware evaluation method for SHA-3 candidates” In Cryptographic Hardware and Embedded Systems, CHES 2010, LNCS 6225, pp. 248-263, 2010.

[9] S. Tillich, M. Feldhofer, M. Kirschbaum, T. Plos, J.-M. Schmidt, and A. Szekely. ”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. Available: http://eprint.iacr.org/2009/510.

[10] Cryptographic HASH and SHA-3 Standard Development. (2015, Aug. 5). NIST. [Online]. Available: http://csrc.nist.gov/groups/ST/hash/index.html Accessed Oct. 10, 2016.

[11] AXI DMA v7.1 LogiCORE IP Product Guide (PG021), Xilinx, Inc., 2015.

[12] M. Knezevic, K. Kobayashi, J. Ikegami, S. Matsuo, A. Satoh, U. Kocabas, J. Fan, T. Katashita, T. Sugawara, K. Sakiyama, I. Verbauwhede, K. Ohta, N. Homma, and T. Aoki, ”Fair and consistent hardware evaluation of fourteen round two SHA-3 candidates,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 20, issue 5, 2012, pp. 827-840.