Farnoud Farahmand
ECE 646 – Fall 2015

Vivado HLS Implementation of Round-2 SHA-3 Candidates

Introduction

NIST announced a public competition on November 2007 to develop a new cryptographic hash algorithm, called SHA-3, for standardization. This contest required about four years for evaluation of the candidates. 51 candidates were submitted to the first round of the contest. NIST selected 14 of them for the second round, and then announced 5 finalists. A fraction of candidates was rejected based on security reasons and the remaining had to be ranked based on their hardware and software performance. Traditional hardware benchmarking consists of 3 steps: 1- Translation of specification to a hardware description language (HDL) such as VHDL and Verilog, 2- Writing a testbench based on test vectors generated by reference software implementation for functional verification, 3- Generating post-place and route result using FPGA tools and performing timing verification on the obtained netlist. These tasks take significant amount of time. The objective of this project is to use High Level Synthesis for hardware benchmarking to decrease the time that is required for these tasks. HLS let us take advantage of high-level programing language, such as C, C++ to generate synthesizable HDL. HLS implementation of 5 finalists is done [1] and we plan to implement the remaining round-2 candidates using Vivado HLS. As a result, we can rank all of round-2 candidates in term of performance in modern FPGAs and compare the results with manual RTL implementation.

Design Tools

1- **HLS tool:** Vivado HLS
2- **Coding Language:** C
3- **Output HDL language:** VHDL
4- **Simulation:** Vivado HLS C-simulation and Co-simulation, Isim
5- **Target platforms:** Xilinx Virtex 6, Altera Stratix IV, Xilinx Zynq 7000
6- **CAD tools:** Xilinx ISE, Xilinx Vivado

Design Specification

Name of candidates which I plan to implement:

1- Luffa
2- CubeHash
3- BMW
4- ECHO
5- SHAvite-3

Firstly, I will write C code in Vivado HLS environment and perform C-simulation to verify my HLS-ready C code. Then, I will synthesize my design one time and generate the corresponding VHDL code to observe the estimated latency and resource utilization in synthesize report. In the next step, I will try to optimize my design using different pragmas in Vivado HLS to achieve the target latency.

Design Source

1- **Submission packages of 14 Round 2 SHA-3 candidates:**
http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/submissions_rnd2.html

2- **Vivado HLS implementation of 5 finalists:**
https://cryptography.gmu.edu/athena/index.php?id=source_codes

3- **RTL implementation of round-2 candidates:**
https://cryptography.gmu.edu/athena/index.php?id=source_codes

4- 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, first version - Aug. 2010

Assumptions

1- Primary Optimization Target: Throughput/Area; Secondary Optimization Target: Throughput.
2- No use of embedded resources, such as multipliers, DSP units, or Block Memories, in the hash cores.

Interface

I will use GMU Hardware Interface for Secure Hash Algorithm (SHA):

https://cryptography.gmu.edu/athena/index.php?id=interfaces

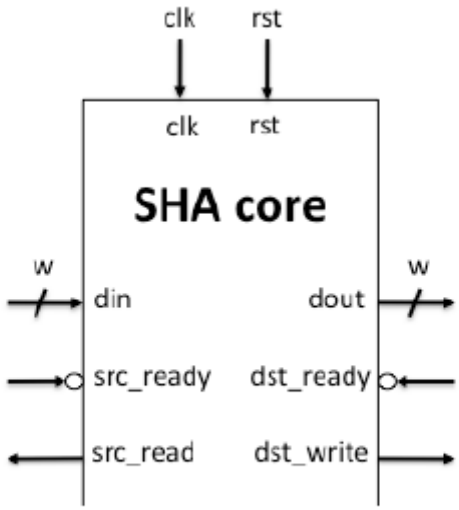


Fig. 1: Input/output interface of a SHA core

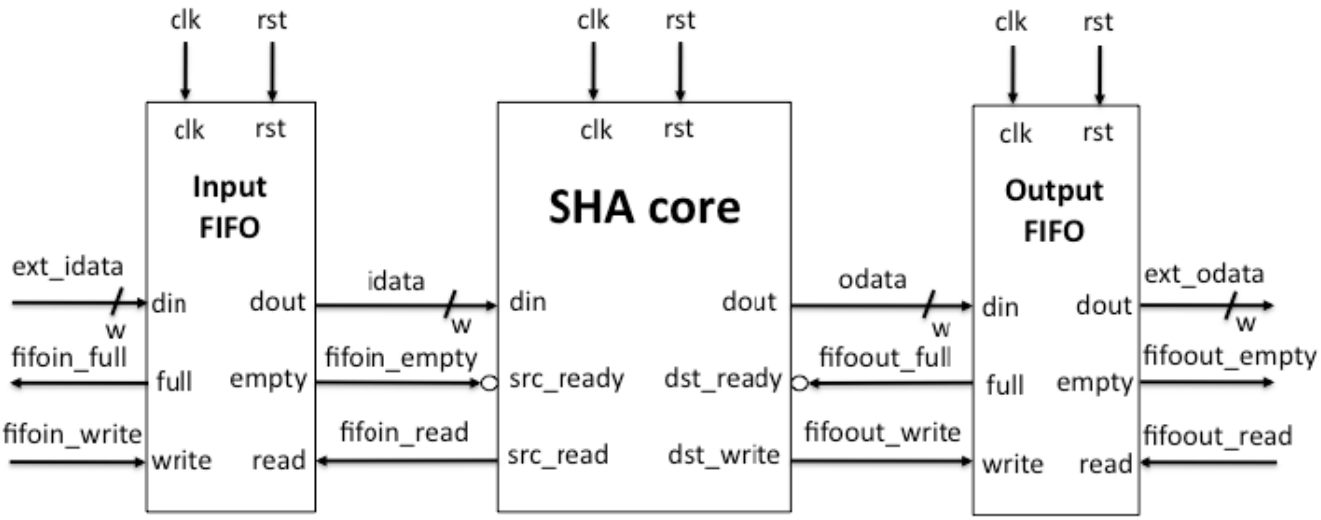


Fig. 2: A typical configuration of a SHA core connected to two surrounding FIFOs. The Input FIFO serves as a source of input data, and the Output FIFO as a destination for output data.

Global Control Signals		
clk	in	Global clock.
rst	in	Global reset, active HIGH. Minimum duration equal to one clock cycle. After the positive reset pulse, the SHA core becomes ready to hash a new message. In case the reset appears during hashing of a message, the hashing is abandoned, no output is generated, and the core becomes ready to accept a new message.

Input Data Interface		
din[w-1:0]	in	A w-bit word of input data.
src_ready	in	A control signal indicating that the source of input is ready to be read from. Active LOW.
src_read	out	A control signal used to read data from the source of data. Data from the din input is assumed to be stored in the SHA core at the next rising edge of the clock. Active HIGH.
Output Data Interface		
dout[w-1:0]	out	A w-bit word of output data (i.e., a word of a hash value).
dst_ready	in	A control signal indicating that the destination of output is ready to be written to. Active LOW.
dst_write	out	A control signal used to write data to the destination of data. Data from the dout output is assumed to be accepted by the destination circuit at the next rising edge of the clock. Active HIGH.

Previous Implementation

Hardware Benchmarking of Cryptographic Algorithms Using High-Level Synthesis Tools: The SHA-3 Contest Case Study

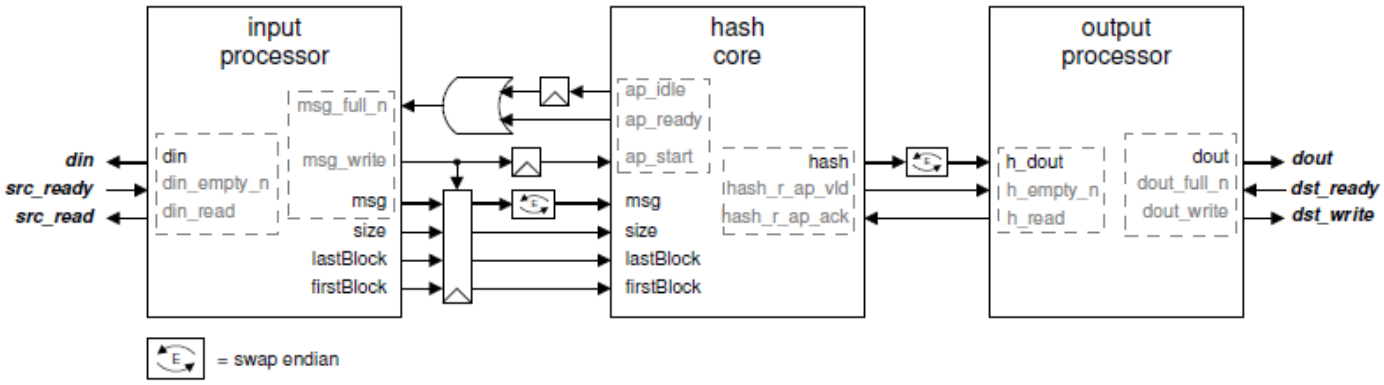
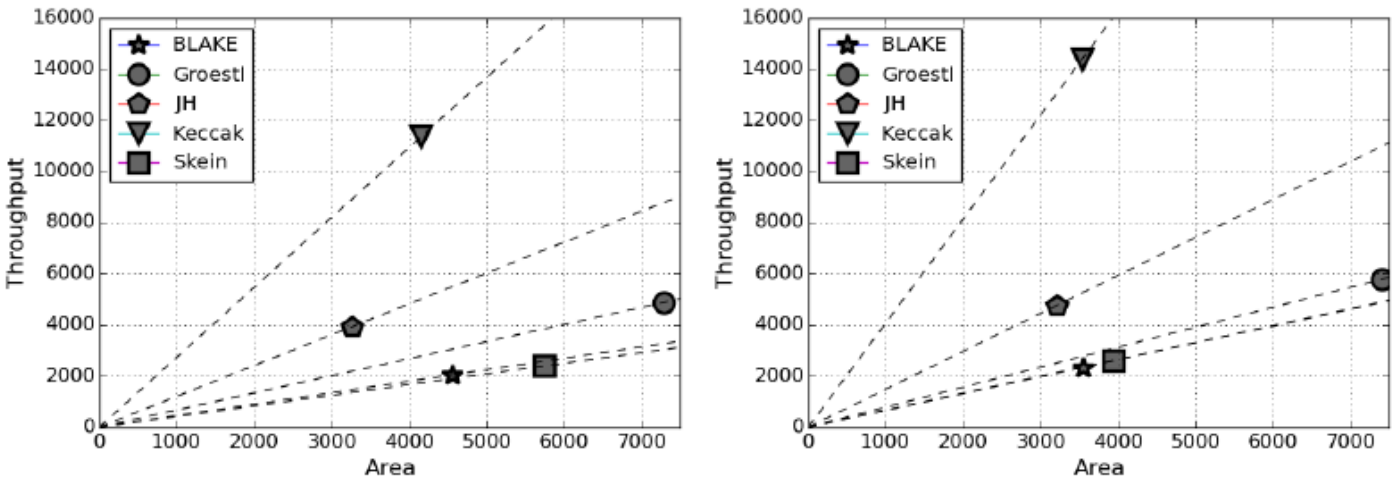


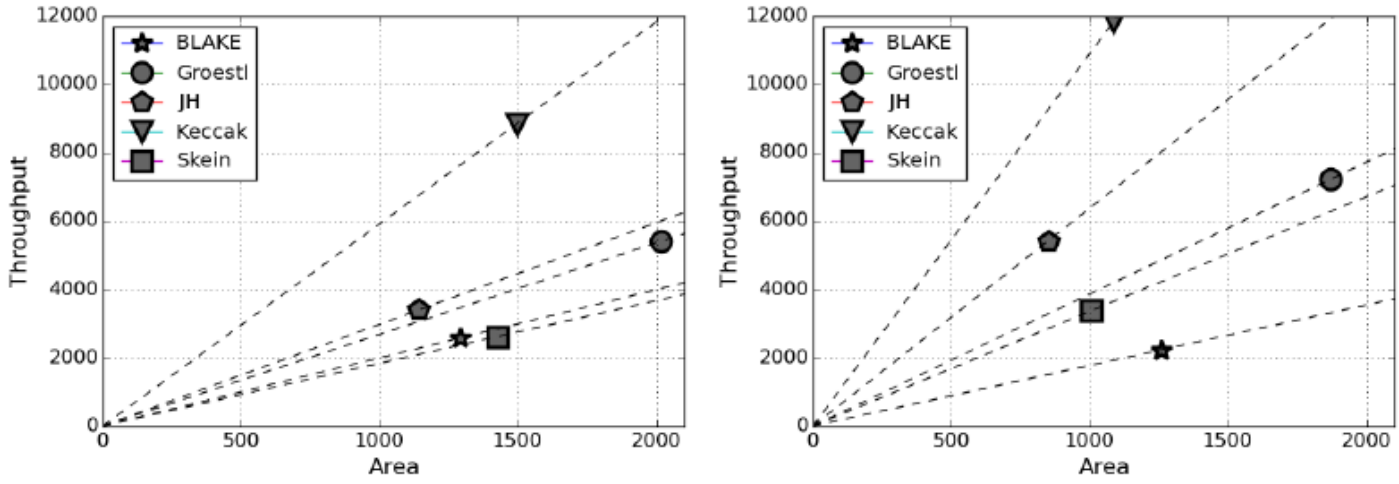
Fig. 3: Top-level diagram.

Manual RTL vs. HLS-based Results for Altera Stratix IV:



(a) HLS (b) RTL

Manual RTL vs. HLS-based Results for Xilinx Virtex 6:



(a) HLS (b) RTL

RTL/HLS performance ratios across investigated algorithms:

FPGA Family	Freq.	Throughput	Area	TP/A
Altera Stratix IV	0.98-1.19	1.09-1.27	0.68-1.02	1.17-1.59
Xilinx Virtex 6	0.84-1.56	0.87-1.59	0.70-0.98	0.89-2.14

Testing Performance and Functionality

I intend to use Vivado HLS C-simulation, Co-simulation and Xilinx Isim. Test vectors are generated using references C codes.

I would be using C-simulation at the first step and after VHDL code generation I use Co-simulation for verification purpose. In the next step I'll use the VHDL code which is generated using Vivado HLS and port map these code with the VHDL code of GMU Hardware Interface and then verify my top-level design using GMU testbench, this part would be done using Xilinx Isim simulator.

Performance parameters are maximum frequency, latency in number of clock cycles, throughput and resource utilization.

I'll generate the results for Xilinx Virtex 6 and Altera Stratix IV for these 5 candidates. Therefore, we can compare them with the results that generated previously for 5 finalists.

Resource utilization parameters:

1- Number of CLB slices
2- Number of Flip Flops
3- Number of LUT

Simulation Experiments and verification using boards

I intend to implement each design on ZEDBOARD (Xilinx Zynq-7000 All Programmable SoC) and experimentally verify the design on the board and measure the performance.

I plan to test my design in the following procedure:

1- maximum clock frequency
2- maximum throughput for various input sizes, taking into account the communication overhead

Note: This section would be done in case that I have enough time to implement at least 5 candidates.

Time Schedule

10/26 -- Implementing at least 2 candidates

11/10 – Implementing third candidate and starting 4th

11/23 – Finish implementation of candidate number 4 and starting number 5

12/08 – 5 ready HLS implementation and start working on ZEDBOARD implementation

Tentative Table of Contents in Final Report

Literature list

[1] K. Gaj, E. Homsirikamol, M. Rogawski, R. Shahid, and M.U. Sharif, "Comprehensive Evaluation of High-Speed and Medium-Speed Implementations of Five SHA-3 Finalists Using Xilinx and Altera FPGAs," Cryptology ePrint Archive, Rep. 2012/368, Oct. 2012.

[2] 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.

[3] E. Homsirikamol and K. Gaj, "Hardware Benchmarking of Cryptographic Algorithms Using High-Level Synthesis Tools: The SHA-3 Contest Case Study," in ARC. Conf., Bochum, Germany, 2015, pp. 217-228.

[4] R. Shahid, U. Sharif, M. Rogawski, and K. Gaj, "Use of Embedded FPGA Resources in Implementations of 14 Round 2 SHA-3 Candidates," in FPT. Conf., New Delhi, India, 2011, pp. 1-9.

[5] E. Homsirikamol, M. Rogawski, and K. Gaj, "Throughput vs. Area Trade-offs in High-Speed Architectures of Five Round 3 SHA-3 Candidates Implemented Using Xilinx and Altera FPGAs," in CHES. Conf., Nara, Japan, 2011, pp. 491-506.

[6] 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 CHES. Conf., Santa Barbara, CA, 2010, pp. 264-278.

[7] T. Grembowski, R. Lien, K. Gaj, N. Nguyen, P. Bellows, J. Flidr, T. Lehman, B. Schott, "Comparative Analysis of the Hardware Implementations of Hash Functions SHA-1 and SHA-512," in ISC. Conf., Sao Paulo, Brazil, 2002, pp. 75-89.

[8] R. Lien, T. Grembowski, K. Gaj, "A 1 Gbit/s Partially Unrolled Architecture of Hash Functions SHA-1 and SHA-512," in RSA. Conf., San Francisco, CA, 2004, pp. 324-328.

[9] Cryptographic HASH and SHA-3 Standard Development. (2015, Aug. 5). NIST. [Online]. Available: http://csrc.nist.gov/groups/ST/hash/index.html. Accessed Sep. 22, 2015.

[10] GMU Source Codes. (2015, Apr. 15). George Mason University. [Online]. Available: https://cryptography.gmu.edu/athena/index.php?id=source_codes. Accessed Sep. 22, 2015.

[11] SHA-3 Hardware Implementations. (n.d.). ECRYPT II. [Online]. Available:http://ehash.iaik.tugraz.at/wiki/SHA-3_Hardware_Implementations. Accessed Sep. 22, 2015.

[12] A. Akin, A. Aysu, O. C. Ulusel, and Savas, "Ev cient Hardware Implementation of High Throughput SHA-3 Candidates Keccak, Luasaa and Blue Midnight Wish for Single- and Multi-Message Hashing. 2nd SHA-3 Candidate" Presented at SIN. Conf., Taganrog, Russia, 2010.

[12] A. Akin, A. Aysu, O. C. Ulusel, and Savas, "Ev cient Hardware Implementation of High Throughput SHA-3 Candidates Keccak, Luasaa and Blue Midnight Wish for Single- and Multi-Message Hashing. 2nd SHA-3 Candidate," Presented at SIN. Conf., Taganrog, Russia, 2010.

[13] B. Baldwin, "FPGA Implementations of the Round Two SHA-3 Candidates." Presented at 2nd SHA-3 Candidate Conf. 2010.

[14] J.-L. Beuchat, E. Okamoto, and T. Yamazaki, "A Compact FPGA Implementation of the SHA-3 Candidate ECHO." Cryptology ePrint Archive, Rep. 2010/364, 2010.

[15] K. Gaj, P.J. Kaps, "ATHENa {Automated Tool for Hardware EvaluatioN: Toward fair and comprehensive benchmarking of cryptographic hardware using FPGAs." in FPL. Conf., 2010.

[16] X. Guo, S. Huang, L. Nazhandali, and P. Schaumont, " Fair and Comprehensive Performance Evaluation of 14 Second Round SHA-3 ASIC Implementations.” Presented at 2nd SHA-3 Candidate Conf. 2010.

[17] S. Matsuo, "How Can We Conduct \Fair and Consistent" Hardware Evaluation for SHA-3 Candidate? "Presented at 2nd SHA-3 Candidate Conf. 2010.

[18] S. Tilich, "High-speed Hardware Implementations of Blake, Blue Midnight Wish, Cubehash, ECHO, Fugue, Groestl, Hamsi, JH, Keccak, Lu_a, Shabal, Shavite-3, SIMD, and Skein" Cryptology ePrint Archive, Rep. 2009/510, 2009.

[19] L. Henzen, "Developing a hardware evaluation method for SHA-3 candidates." Presented at CHES. Conf., Santa Barbara, CA, 2010.

[20] SHA-3. (n.d.). Wikipedia. Available: https://en.wikipedia.org/wiki/SHA-3. Accessed Sep. 22, 2015.

[21] E. Homsirikamol and K. Gaj, "Can High-Level Synthesis Compete Against a Hand-Written Code in the Cryptographic Domain? A Case Study," in ReConFig. Conf., Cancun, Mexico, 2014.