An Alternative Approach to Hardware Benchmarking of CAESAR Candidates Based on the Use of High-Level Synthesis Tools

Ekawat Homsirikamol and Kris Gaj
George Mason University

Abstract

The growing number of candidates competing in the cryptographic contests, such as CAESAR, makes the hardware performance evaluation extremely time consuming, tedious, and imprecise, especially at the early stages of the competitions. The main difficulties include the long time necessary to develop and verify the Register-Transfer Level (RTL) hardware description language (HDL) code of all candidates, and the need of developing (or at least tweaking) code for multiple variants and architectures of each algorithm. On the other hand, recent years have brought dramatic improvements in the quality, ease of use, and effectiveness of general-purpose High-Level Synthesis (HLS) tools. In this presentation, we will investigate the use of one of such tools, Xilinx Vivado HLS [1], for efficient benchmarking of CAESAR candidates in hardware.

In our project, the reference C code, developed by the authors of each algorithm, was used as a starting point for the corresponding HLS design. Each reference C implementation was then manually modified, to create a close-to-optimum, HLS-ready C code, verified in software. This code was then passed as an input to Vivado HLS, and the corresponding VHDL code was automatically generated. The obtained HDL code was then simulated for functional correctness. Through this simulation, the number of clock cycles required to process a block of input data was determined. If the results were incorrect, or the number of clock cycles higher than the number of cipher rounds plus a small constant (to be expected for the basic iterative architecture), the HLS-ready code was modified, and the entire process repeated. If the HDL code performed as expected, this code was used to generate the final post-place and route results, as well as the final netlist verified using timing simulation.

In order to verify the potential validity of our approach, we have applied both the traditional RTL-based methodology, as well as our proposed HLS-based methodology to the evaluation of over 20 Round 2 CAESAR candidate families (with existing RTL implementations fully conforming to the single-pass CAESAR API), compared against each other and against the current NIST standard, AES-GCM. Our study aimed at verifying the following hypothesis:
Ranking of candidate algorithms in cryptographic contests in terms of their performance in modern FPGAs will remain the same, independently whether the HDL implementations are developed manually or generated automatically using High-Level Synthesis tools.

As a reference for comparison, we used RTL VHDL and Verilog code developed by about a dozen of hardware design teams close to the end of Round 2 [2–4]. All implementations, both

RTL and HLS-based, used the same standard Application Programming Interface (API) approved by the CAESAR Committee [5]. The entire RTL code, both that developed manually and that produced by HLS, was post-processed using the same FPGA tools and the same optimization strategies based on the use of ATHENA [6–7] or Xilinx Vivado [8].

Our study has demonstrated a very good correlation between RTL and HLS rankings according to three major performance metrics: throughput, area, and the throughput to area ratio. At the same time, a documented speed-up by a factor of at least 3 to10 in terms of the development time has been observed. The main sources of this speed-up were the simpler implementation and verification. In the HLS-based design approach, the designer could focus on the functionality of the design (datapath), while the control logic was inferred automatically. Additionally, the testbench development (now in C++ rather than VHDL) and the code verification were also much easier to conduct.

In order for the HLS-based design approach to be an effective replacement for the manual RTL-based design approach, there are several obstacles that need to be still overcome. These obstacles include for example an inconsistent operation of Xilinx FPGA tools, generating large swings in the relative ratio of the RTL to HLS results, depending just on the selected FPGA family, without any changes in the input HDL code. Additionally, the preparation of the HLS-ready C code may need to be simplified, either by following a well-defined and documented sequence of manual optimizations or by an automatic preprocessing of the generic C code, developed without the high-level synthesis in mind.

It should be also stressed that Vivado HLS does not target Application Specific Integrated Circuits (ASICs), and its use with non-Xilinx devices (either FPGAs or ASICs) is considered a breach of the license agreement. This issue could be possibly mitigated in future work through the use of the most-advanced academic HLS tools, such as Bambu, DWARV, and LegUp [9].

In any future cryptographic competitions, the proposed approach could be applied at even earlier stages of the contests, e.g., during the design phase of the candidate algorithms, when multiple variants, with different security vs. efficiency trade-offs, are considered, or in Round 1, where the large number of candidates makes RTL implementations prohibitively expensive in terms of time and effort.

References

[1] Xilinx, Vivado High-Level Synthesis, available at http://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html

[2] VHDL/Verilog Code of CAESAR Candidates: Summary I, available at https://cryptography.gmu.edu/athena/CAESAR_HW_Summary_1.html

[3] VHDL/Verilog Code of CAESAR Candidates: Summary II, available at https://cryptography.gmu.edu/athena/CAESAR_HW_Summary_2.html

[4] "GMU Source Code of Round 2 CAESAR Candidates, AES-GCM, AES, AES-HLS, and Keccak Permutation F available at https://cryptography.gmu.edu/athena/index.php?id=CAESAR

[5] E. Homsirikamol, W. Diehl, A. Ferozpuri, F. Farahmand, P. Yalla, J.-P. Kaps, and K. Gaj, "CAESAR Hardware API," Cryptology ePrint Archive: Report 2016/626, available at http://eprint.iacr.org/2016/626.
[6] K. Gaj, J.P. Kaps, V. Amirineni, M. Rogawski, E. Homsirikamol, B.Y. Brewster, "ATHENa – Automated Tool for Hardware EvaluatioN: Toward Fair and Comprehensive Benchmarking of Cryptographic Hardware using FPGAs," 20th International Conference on Field Programmable Logic and Applications, Milano, Italy, Aug. 31st - Sep. 2nd, 2010.
[7] ATHENa, available at https://cryptography.gmu.edu/athena/index.php?id=ATHENa
[8] Xilinx. Vivado Design Suite, available at http://www.xilinx.com/products/design-tools/vivado.html
[9] R. Nane, et al., "A Survey and Evaluation of FPGA High-Level Synthesis Tools," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (in print).