Computer-Aided Design Tools and Methodologies for Evaluating Candidates in Cryptographic Contests[1]

Ekawat Homsirikamol, Farnoud Farahmand, Ahmed Ferozpuri,
Marcin Rogawski[2], Panasayya Yalla, and Kris Gaj
George Mason University, USA
{ehomsiri, aferozpu, ffarahma, pyalla, kgaj}@gmu.edu, marcin@cadence.com

Abstract

Cryptographic contests have emerged as a commonly accepted way of developing cryptographic standards. A few representative examples include the AES (Advanced Encryption Standard) competition, conducted in the period 1998-2000, the SHA-3 (Secure Hash Algorithm 3) contest, organized in the period 2008-2012, and the CAESAR - Competition for Authenticated Encryption: Security, Applicability, and Robustness, started in 2014, currently in progress [1].

Performance of candidates in hardware has always been a very important evaluation factor in all major cryptographic contests, serving as a tie-breaker, when all remaining algorithms have been found to have adequate security strength. However, the growing number of candidates, from 15 in case of AES to 57 in case of CAESAR, combined with the specific requirements of the contests (such as the lack of any particular performance target and a wide range of the designers' skills) calls for the adoption of more efficient and CAD-oriented design and implementation methodologies.

During the SHA-3 contest, our group has developed ATHENa (Automated Tool for Hardware EvaluatioN), aimed at generating optimized results using heuristic strategies, by finding the best possible options of tools, target clock frequency, and starting point for placement [2–3]. The obtained results have demonstrated an improvement in terms of the throughput/area ratio by up to a factor of two.

During the CAESAR competition, three other major innovations have been introduced. First, a standard hardware *Application Programming Interface (API)* for authenticated ciphers has been developed by the GMU Team, and approved by the Organizers of the Contest – The CAESAR Committee [4]. Second, a comprehensive *Development Package*, including VHDL and Python code, supporting the development of implementations compliant with the CAESAR Hardware API, has been developed and thoroughly documented [5]. Third, the design teams have been asked to submit their *own Verilog/VHDL code* before the end of Round 2 [1].

This approach has led to the biggest hardware benchmarking effort in the history of cryptographic competitions, with 14 hardware designer groups submitting a total of 75 unique designs, covering 28 out of 29 Round 2 CAESAR candidate families. Compared to the previous SHA-3 contest, the number of candidates benchmarked in hardware has doubled and reached about 50% of all initial submissions.

This approach, however, was still based on the traditional Register-Transfer Level (RTL) design methodology, which is very time consuming, error-prone, and subject to a considerable bias caused by different skills of designers and different amount of time and effort devoted to the design optimizations.

As a result, in parallel to the well-established RTL-based approach, we have developed a new CAD-based methodology, based on the recent dramatic improvements in the quality, ease of use, and effectiveness of general-purpose High-Level Synthesis (HLS) tools.

In this new methodology, a designer starts from a reference software implementation, which is typically provided by the algorithm's designers in C. This implementation is then manually transformed into an HLS-ready C code, by the 1) addition of HLS tool directives, 2) hardware-driven code refactoring, and 3) extension of the functionality to accommodate differences between the software and hardware API. All of these transformations make the C code more suitable for the hardware description, function

[1] This work has been partially supported by NSF Grant #1314540.
[2] Currently with Cadence Designs Systems, San Jose, CA.

parallelization, and resource reuse. Once these modifications are completed, the code is verified in software for the correct functionality. Afterward, the HLS-ready C code is processed by the HLS tool to generate the corresponding RTL HDL code and the first performance estimates (e.g., the latency in clock cycles). The obtained code is then simulated for functional correctness, using a traditional HDL simulator, e.g., ModelSim. If the results are incorrect, or the number of clock cycles per block higher than expected (based on the number of cipher rounds and the target architecture), the HLS-ready code must be modified, and the entire process repeated. If the code is correct and infers an expected hardware architecture, the HLS phase of the design is completed. The obtained code is further processed using the tools and optimization strategies identical to those used in the RTL design flow.

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 26 Round 2 CAESAR candidate families (all single-pass candidate families, with existing RTL implementations fully conforming to the CAESAR Hardware API). All candidates were 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 an HLS tool, we selected a commercial tool, Xilinx Vivado-HLS. The advantages of this tool include good documentation and user support, close integration with the primary Xilinx FPGA Design Suite – Vivado, the largest number of performance optimizations, and (on average) the highest maximum clock frequency of the generated designs. The licensing limitations include the prohibition on a) targeting FPGAs of other vendors (e.g., Altera), b) targeting ASICs, and c) publishing comparative studies containing results obtained using other HLS tools.

Our study has demonstrated a very good correlation between RTL and HLS rankings for both SHA-3 and CAESAR candidates, according to the three major performance metrics: throughput, area, and the throughput to area ratio. At the same time, a documented speed-up by a factor of 3 to10, in terms of the development time, has been observed. The main sources of this speed-up were the higher level of abstraction and simpler 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 and faster to conduct.

Our results compared favorably with the results reported for the cryptographic benchmarks in [6]. In particular, our worst HLS to RTL ratio, in terms of the number of clock cycles per block, was 1.2, while the authors of [6] were able to achieve only the ratios ranging from 2.5 to 129, even after optimizations.

In the remaining stages of the CAESAR competition, the HLS-based approach can be used to detect and improve any suboptimal RTL implementations. In any future cryptographic contests, the proposed methodology could be applied at even earlier stages, e.g., during the design phase of the candidate algorithms, when multiple variants, with different security vs. implementation 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] CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness, available at https://competitions.cr.yp.to/caesar.html

[2] K. Gaj, et al., "ATHENa – Automated Tool for Hardware EvaluatioN: Toward Fair and Comprehensive Benchmarking of Cryptographic Hardware using FPGAs," Proc. FPL 2010, Milano, Italy, 2010.

[3] ATHENa, available at https://cryptography.gmu.edu/athena/index.php?id=ATHENa

[4] 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.

[5] https://cryptography.gmu.edu/athena/index.php?id=CAESAR

[6] R. Nane, et al., "A Survey and Evaluation of FPGA High-Level Synthesis Tools," IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 35, no. 10, Oct. 2016, pp. 1591-1604.