Performance Analysis of Falcon Post-Quantum Cryptography in Embedded Hardware-Software Integration

John Biselx[‡] and Andrea Guerrieri[‡] *Senior Member, IEEE*
[‡]School of Engineering, HES-SO Valais-Wallis, Sion, Switzerland

***Abstract*—Post-Quantum Cryptography (PQC) algorithms are gaining significant interest as they transition from theoretical prototypes to practical implementations, particularly in embedded applications such as the Internet of Things (IoT). Falcon, a PQC signature standard candidate selected by NIST, is the focus of this study. This paper aims to (1) benchmark Falcon's performance on an embedded target and (2) explore hardware-software codesign approaches to enhance the performance of cryptographic functions. Our hardware-software codesign approach demonstrates a speedup of up to 42× for the most critical functions with a 1.8× overall performance improvement with respect to the baseline.**

I. Introduction

As the field of quantum computing advances, the need for cryptographic systems that can withstand quantum attacks has become increasingly urgent. Post-Quantum Cryptography (PQC) offers solutions that are resilient against both classical and quantum computational threats. This is particularly critical for embedded systems, which are pervasive in various applications, from Internet of Things (IoT) devices to critical infrastructure. Embedded systems are characterized by their limited computational resources, including processing power, memory, and energy consumption. Implementing PQC on such targets requires optimizing these algorithms to fit within these constraints without compromising security. Effective hardware-software codesign can play a fundamental role in achieving this balance, ensuring that PQC implementations are both secure and efficient. Falcon is designed for efficiency and security, leveraging lattice-based cryptography.

II. Related Work and Contribution

Falcon stands out due to its recursive nature and utilization of floating-point (FP) numbers, features uncommon among other PQC algorithms. However, FP arithmetic poses a persistent challenge for FPGAs, thereby introducing an additional layer of complexity to its implementation. Previous efforts tried to implement stand-alone hardware accelerators of cryptographic functions [1]. Others tried implementing Falcon using a hardware-software codesign approach targeting FPGA [2], and ASIC [3]. This paper seeks to achieve two primary objectives: firstly, to conduct a performance benchmark of Falcon on an embedded target, and secondly, to investigate hardware-software codesign strategies aimed at improving the performance of cryptographic functions. The remainder of this paper is organized as follows: Section III introduces the performance analysis of the cryptography functions. Section IV presents the hardware-software codesign strategy, the preliminary results are shown in SectionV to conclude the paper with future work.

III. Performance Analysis of Falcon PQC

The three operations performed by Falcon are *key generation*, *signature*, and *verification*. To measure the execution time of the Falcon algorithm for the operations mentioned above, the built-in performance measurement tool is provided in the code submitted to NIST. The selected target for our performance analysis is an AMD SoC integrating a dual-core Arm Cortex-A9 CPU, along with the FPGA. The Arm Cortex-A9 can be configured to include both NEON (Advanced SIMD) and VFPv3 (Vector Floating Point version 3) coprocessors. To evaluate maximum-achievable performance, the compiler has been configured to utilize the above-mentioned operations provided by the CPU's floating-point unit (FPUs). Execution results are displayed in Table I.

TABLE I
Performance evaluation of Falcon operations running on the embedded target. The CPU is a dual-core Arm Cortex-A9.

Degree [bits]	**Key Generation** $[ms]$	**Signature** $[us]$	**Verification** $[us]$
256	325	8796	1275
512	617	18315	2696
1024	1541	38169	5647

Performance analysis showed notable results: as expected, the complex FFT, named *Falcon Inner FFT*, and the function *process_block* resulted in the most expensive execution percentage for the three operations, respectively 40% and 15%.

IV. Hardware-Software Codesign

The second goal of this work is to explore hardware-software codesign strategies to improve the performance of the cryptographic functions for the embedded hardware. Particularly, in this work we explored the viability of using high-level synthesis (HLS) which would allow quicker deployment of such algorithms onto hardware design. HLS takes code written in C/C++ and generates hardware description language (HDL) for a given target. The study underscores the potential of hardware acceleration in enhancing computational efficiency while also pointing to the necessity of balancing resource allocation to ensure scalability and efficiency.

A. Process block

The exploration of hardware acceleration for the *process_block* function through HLS reveals significant performance improvements. The results showed a 35× performance increase, achieved without any code modifications or special synthesis directives (i.e. #pragmas) apart from loop pipelining,

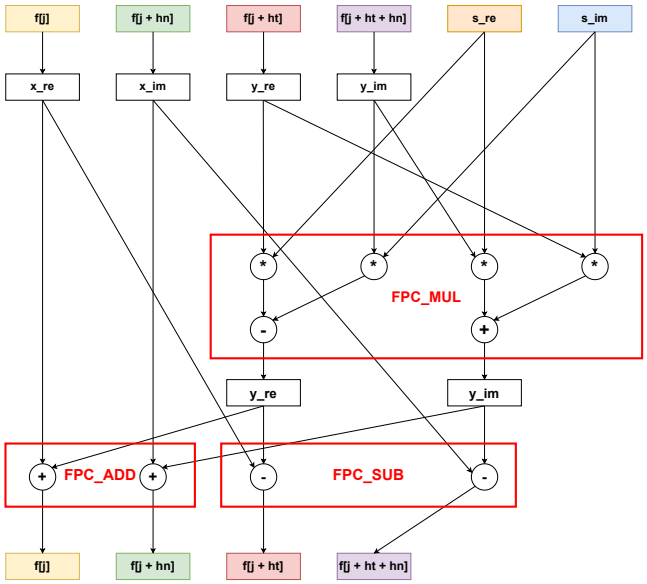

Fig. 1. Complex FFT Operation performed in Falcon. The FPC directives are macros that calculate mathematical operations with complex operands. The first two arguments are the output, the next pair is the first complex number, and the final two is the second complex operand.

highlighting the progress and advancements of EDA technology. Resource utilization for the accelerated implementation included 6% of the available Flip-Flops (FFs), 3% of the Block RAMs (BRAMs), and a considerable 32% of the Look-Up Tables (LUTs). Nevertheless, the substantial LUT usage presents a clear trade-off between area and performance, emphasizing the need for strategic resource management, especially when multiple functions require acceleration.

B. Falcon Inner FFT

Conversely to the function presented in Section IV-A, where HLS did not raise any performance issues, for the function FFT HLS generated suboptimal results [4]. Loop-carried dependencies prevent the efficient pipelining of the inner-most loop, increasing the initiation interval (number of clock cycles before starting a new iteration) up to the value of the iteration latency (the duration in clock cycle of one iteration), in this case 168. To reduce the initiation interval, manual intervention with code refactoring is needed. The data dependencies have been analyzed and solved. Figure 1 shows the dataflow graph of the complex FFT operations. The HLS results are shown in Table II. The optimized version presents an initiation interval of 4, with a speedup of 42× with respect to the original solution. The total resources do not exceed 10% of the FPGA.

TABLE II
FALCON INNER FFT HIGH-LEVEL SYNTHESIS RESULTS.

Version	Initiation Interval (cc)	Speedup (×)	LUTs	FFs	DSPs	BRAMs
Original	168	-	6048	5093	25	16
Optimized	4	42	6661	5790	17	24

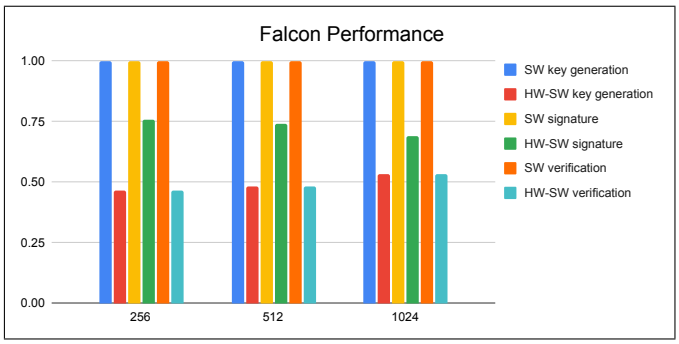

Fig. 2. Falcon Performance for the three cryptographic operations, normalized. The results are the execution runtime profiled on the embedded hardware. The hardware-software codesign approach is accelerating the operations of a factor 1.8× on average.

V. PRELIMINARY RESULTS

The experiments have been performed on a Zedboard, provided with AMD Zynq XC7Z020 SoC, which includes an Arm Cortex-A9 dual-core CPU operating at 667 MHz, and Artix-7 FPGA fabric. The HLS tool used for the experiment is Vitis HLS 2023.1, the *arm-linux-gnueabi-gcc* cross-compiler for the Arm CPU, and *gprof* CPU profiler. The final results are presented in Figure 2. The hardware-software codesign approach shows a speedup of an average of 2× for the *key generation*, 1.4× for the *signature*, and 2× for the *verification*. On one note, the speedup showcased in our results is conservative: the experiments have been performed using a clock frequency of 100MHz, which could be easily increased up to 250MHz leading to a speedup of 4.5×. Therefore, this work is expected to outperform the 1.6× speedup reported in [2], targeting the same hardware platform.

VI. CONCLUSION AND FUTURE WORK

Implementing Falcon PQC on embedded hardware is an important step toward securing embedded systems against future quantum threats. Future work includes the integration of other functions such as the random number generator as well as the evaluation of the impact on energy efficiency. To the best of our knowledge, this paper is the first to attempt to optimize Falcon PQC using HLS. Additionally, the preliminary results presented in this work are poised to surpass the state-of-the-art performance achieved in previous work [2].

REFERENCES

[1] A. Guerrieri, G. Da Silva Marques, F. Regazzoni, and A. Upegui, "H-saber: An FPGA-optimized version for designing fast and efficient post-quantum cryptography hardware accelerators," in *2023 24th International Symposium on Quality Electronic Design (ISQED)*, 2023, pp. 1–6.

[2] E. Karabulut and A. Aysu, "A hardware-software co-design for the discrete gaussian sampling of Falcon digital signature," in *2024 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2024, pp. 90–100.

[3] Y. Lee, J. Youn, K. Nam, H. H. Jung, M. Cho, J. Na, J.-Y. Park, S. Jeon, B. G. Kang, H. Oh, and Y. Paek, "An efficient hardware/software co-design for Falcon on low-end embedded systems," *IEEE Access*, vol. 12, pp. 57 947–57 958, 2024.

[4] A. Guerrieri, G. D. S. Marques, F. Regazzoni, and A. Upegui, "Optimizing post-quantum cryptography codes for high-level synthesis," in *2022 Euromicro Conference on digital systems Design (DSD22)*, Gran Canaria, Spain, 2022, pp. 361–67.