

SW/HW Codesign of the Post-Quantum Cryptography Algorithm NTRUencrypt Using HLS and RTL Design Methodologies

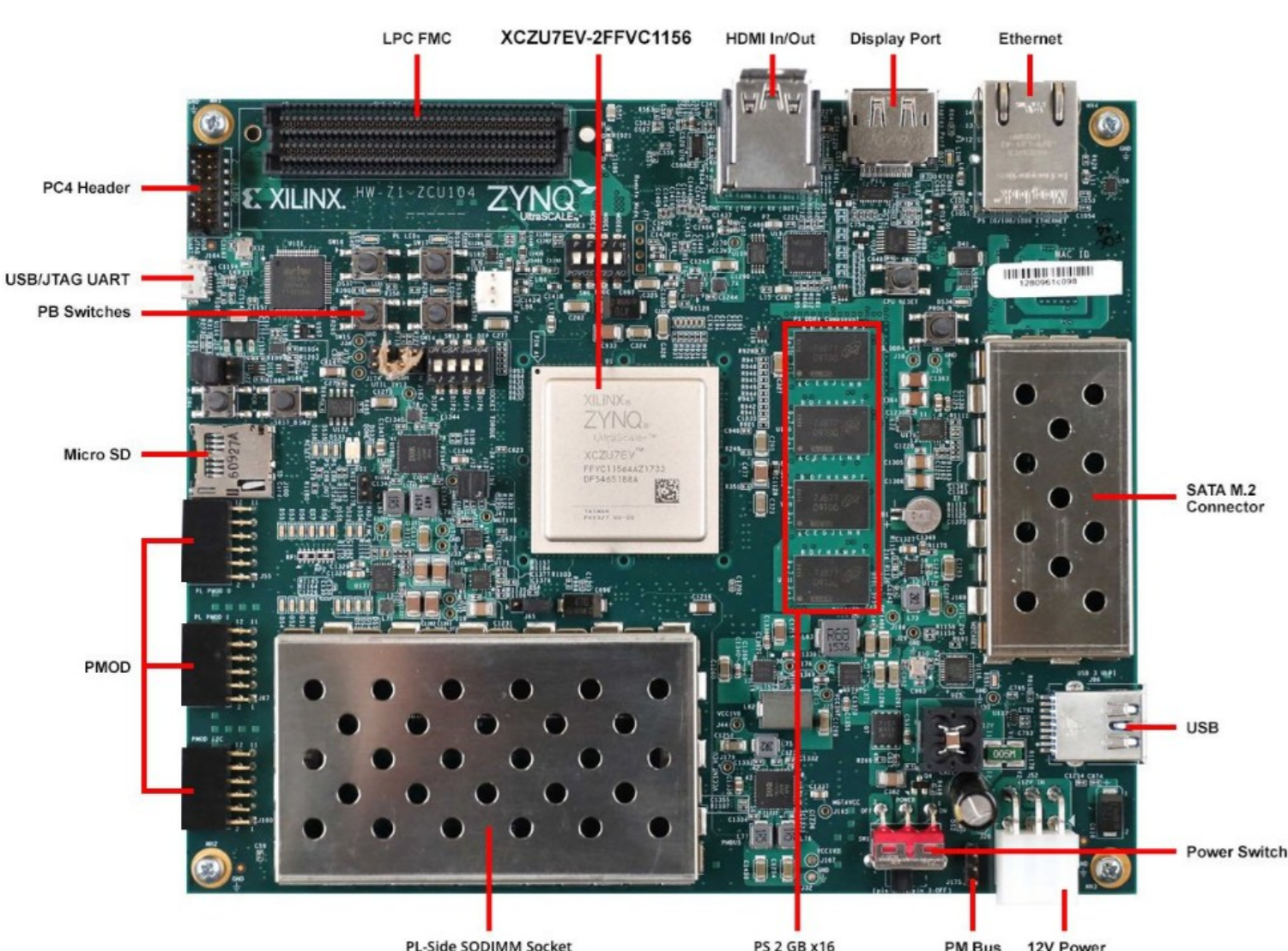
Farnoud Farahmand, Duc Tri Nguyen, Viet B. Dang, Ahmed Ferozpur and Kris Gaj

Department of Electrical and Computer Engineering, George Mason University, Fairfax, Virginia 22030, USA

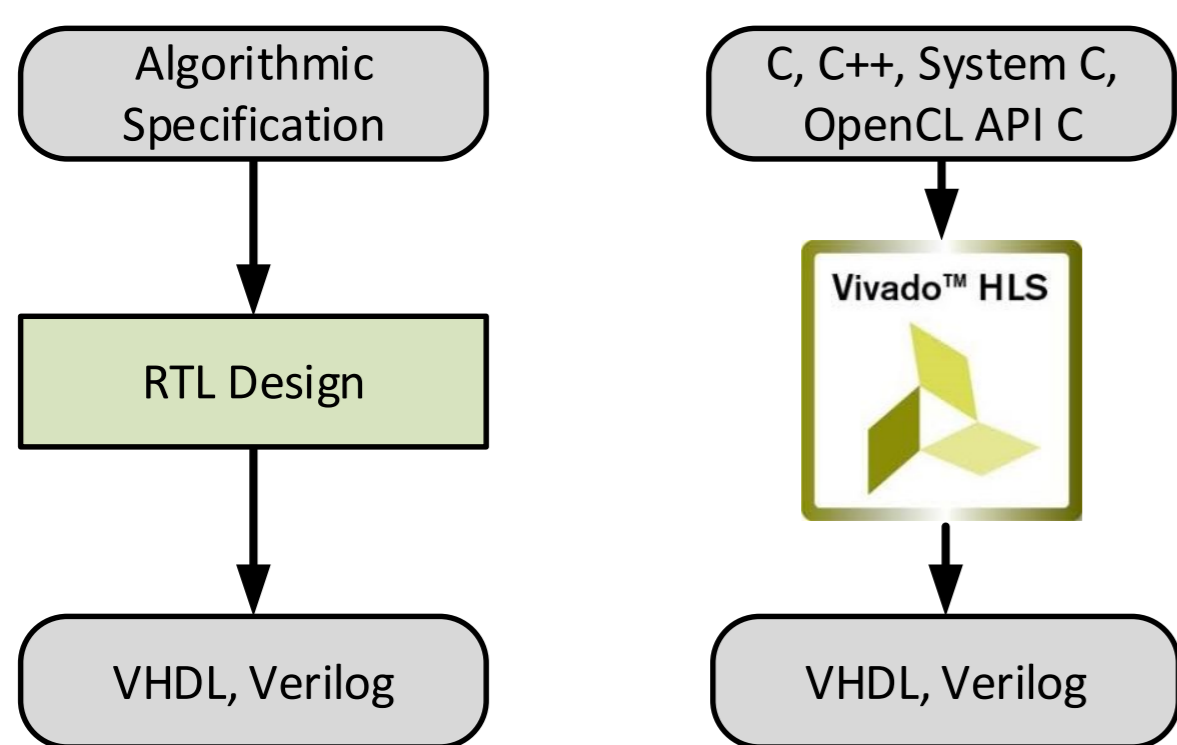
INTRODUCTION & METHODOLOGY

- When **quantum computers** become scalable and reliable, they are likely to **break all public key standards**, such as RSA and Elliptic Curve Cryptography.
- U.S. National Institute of Standards and Technology (NIST) initiated the **Post-Quantum Cryptography (PQC) Standardization Process** aimed at replacing existing public key standards with new quantum-resistant algorithms.
- NTRUencrypt** is one of the most well-known PQC algorithms that has **withstood cryptanalysis**.
- The **speed of NTRUencrypt** in software, especially on embedded software platforms, is **limited by the long execution time of polynomial multiplication**.
- We implement two variants of the NIST Round 1 PQC candidate **NTRUencrypt**: ntru-pke-443 and ntru-pke-743 in **bare-metal mode**.

We investigate speeding up NTRUencrypt using software/hardware codesign on a Xilinx Zynq UltraScale+ Multi-Processor System-on-Chip (MPSoC).

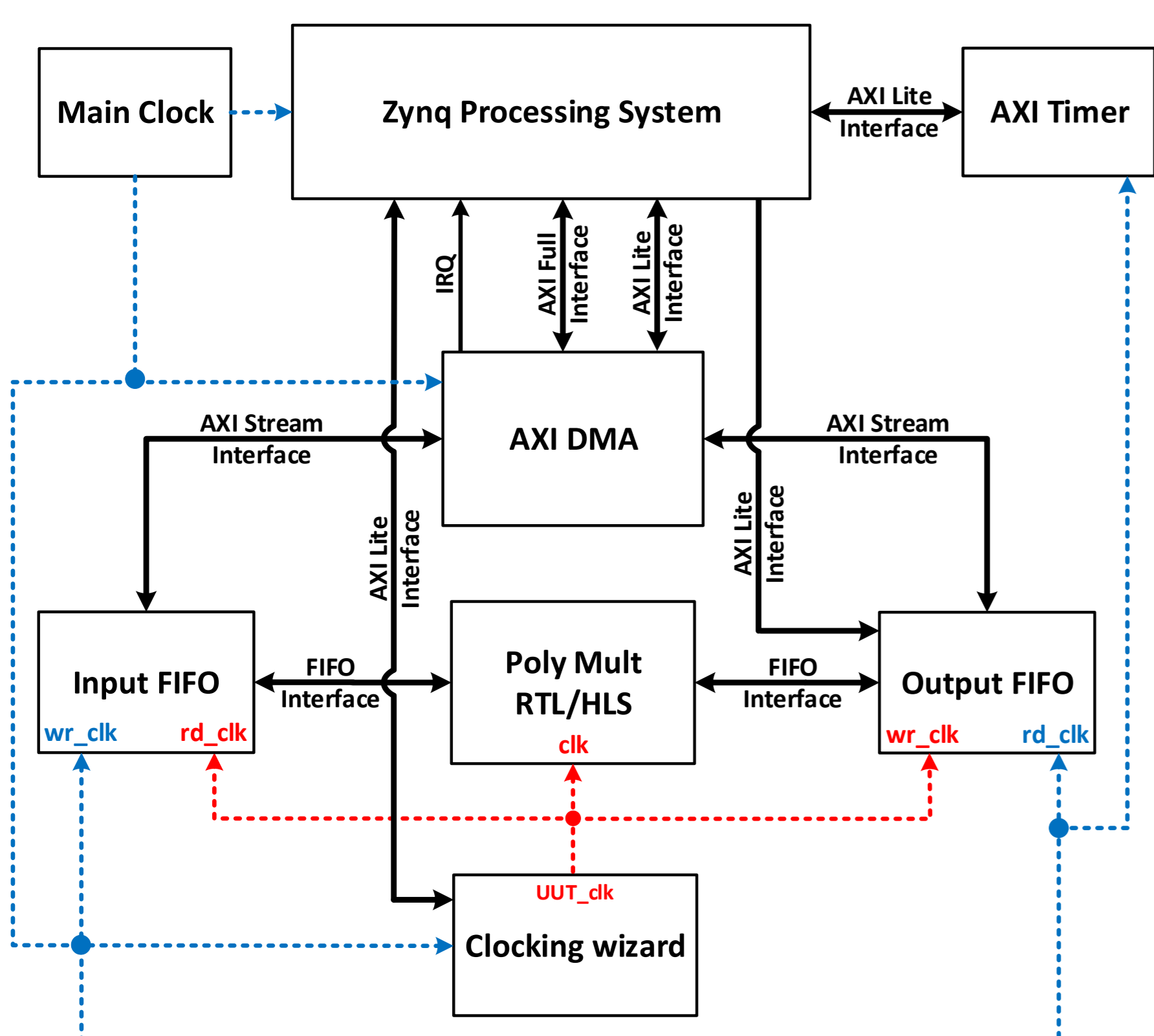


- Polynomial multiplication is implemented in the Programmable Logic (PL) of Zynq using two approaches: traditional Register-Transfer Level (RTL) and High-Level Synthesis (HLS).

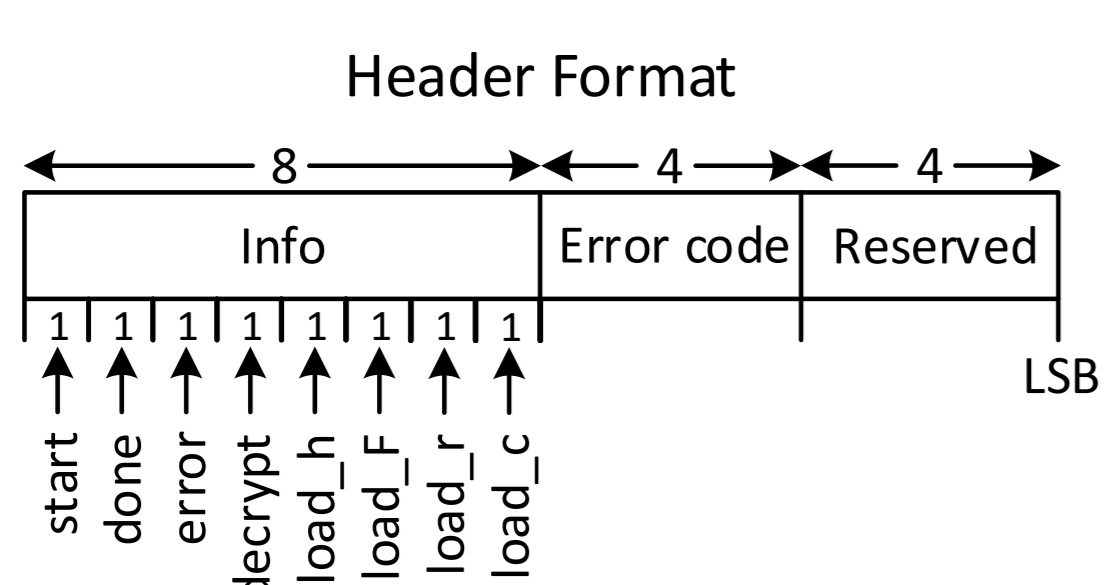


SYSTEM DESIGN

- High-level block diagram of the experimental platform**



Format of a header used to pass instructions from software to the hardware accelerator, as well as status and error codes for transmission in the opposite direction



SYSTEM DESIGN

Algorithm 1 Polynomial Multiplication, Poly Mult

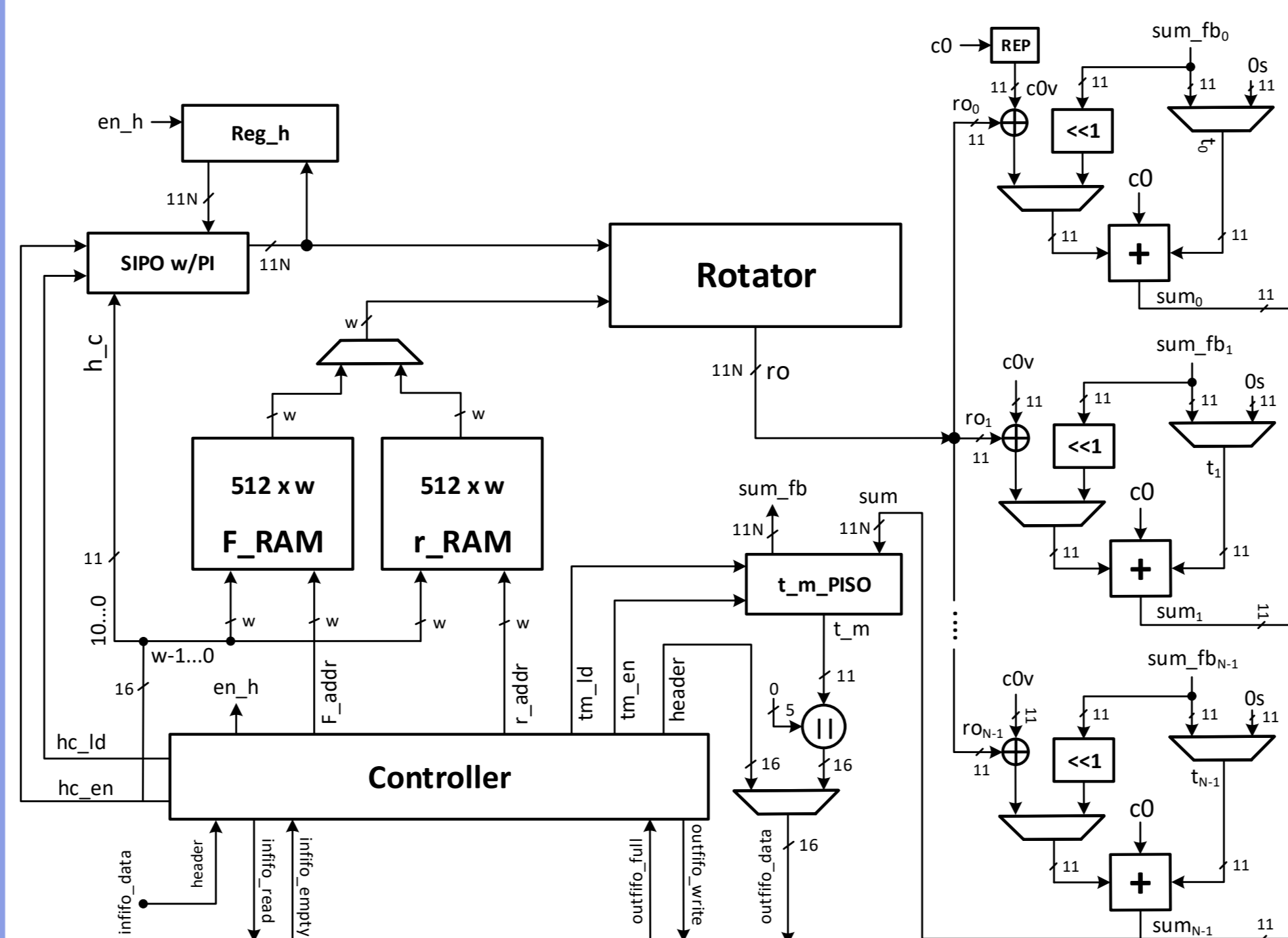
```

1: Inputs:
2: Polynomial a(X) with N "big" coefficients in the range [0, q - 1].
3: Polynomial b(X) with d coefficients 1 at the locations b_0, b_1, ..., b_{d-1} and d coefficients -1 at the locations b_d, b_{d+1}, ..., b_{2d-1}, where 0 ≤ b_i ≤ N - 1.
4: Output:
5: c(X) = a(X) * b(X) mod X^N - 1
6: Pseudocode:
7: for i := 0 to 2d - 1 do
8:   if i < d then
9:     for j := 0 to N - 1 do
10:      c_j = c_j + a_j * b_{i-j} mod N
11:     end for
12:   else
13:     for j := 0 to N - 1 do
14:      c_j = c_j - a_j * b_{i-j} mod N
15:     end for
16:   end if
17: end for

```

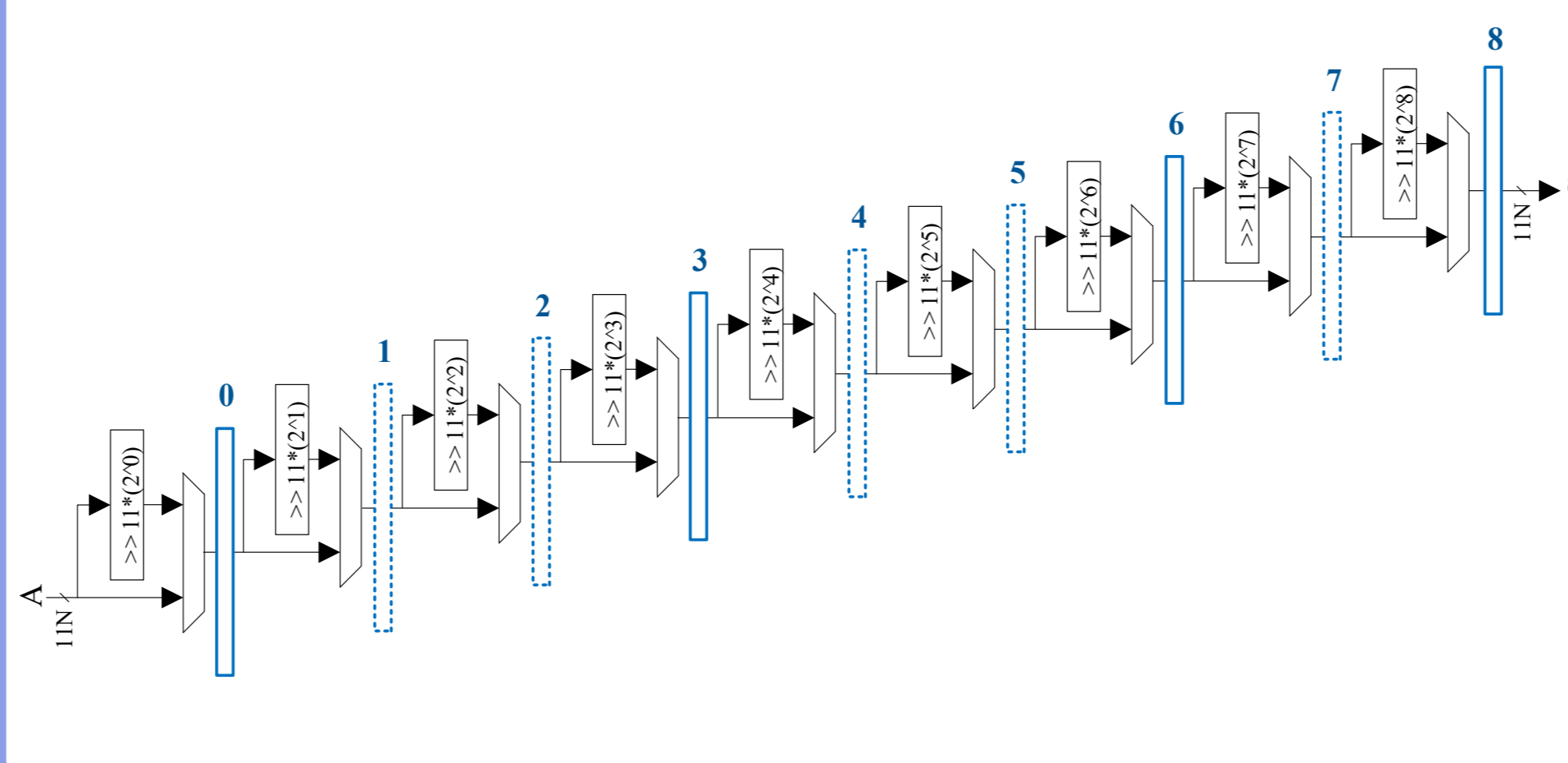
- The for loop in lines 9-11 corresponds to adding to the temporary polynomial $c = c_{N-1}..c_0$ the input polynomial $a = a_{N-1}..a_0$ rotated by b_i locations to the left, namely, $a \lll b_i = a_{N-b_i-1}..a_0 a_{N-1}..a_{N-b_i}$.
- Similarly, the for loop in lines 13-15 corresponds to the subtraction of the same value from c .

Block diagram of the hardware accelerator, implemented in RTL and inferred in HLS. REP denotes a unit replicating a single bit 11 times. $w=9$ for ntru-pke-443 and $w=10$ for ntru-pke-743



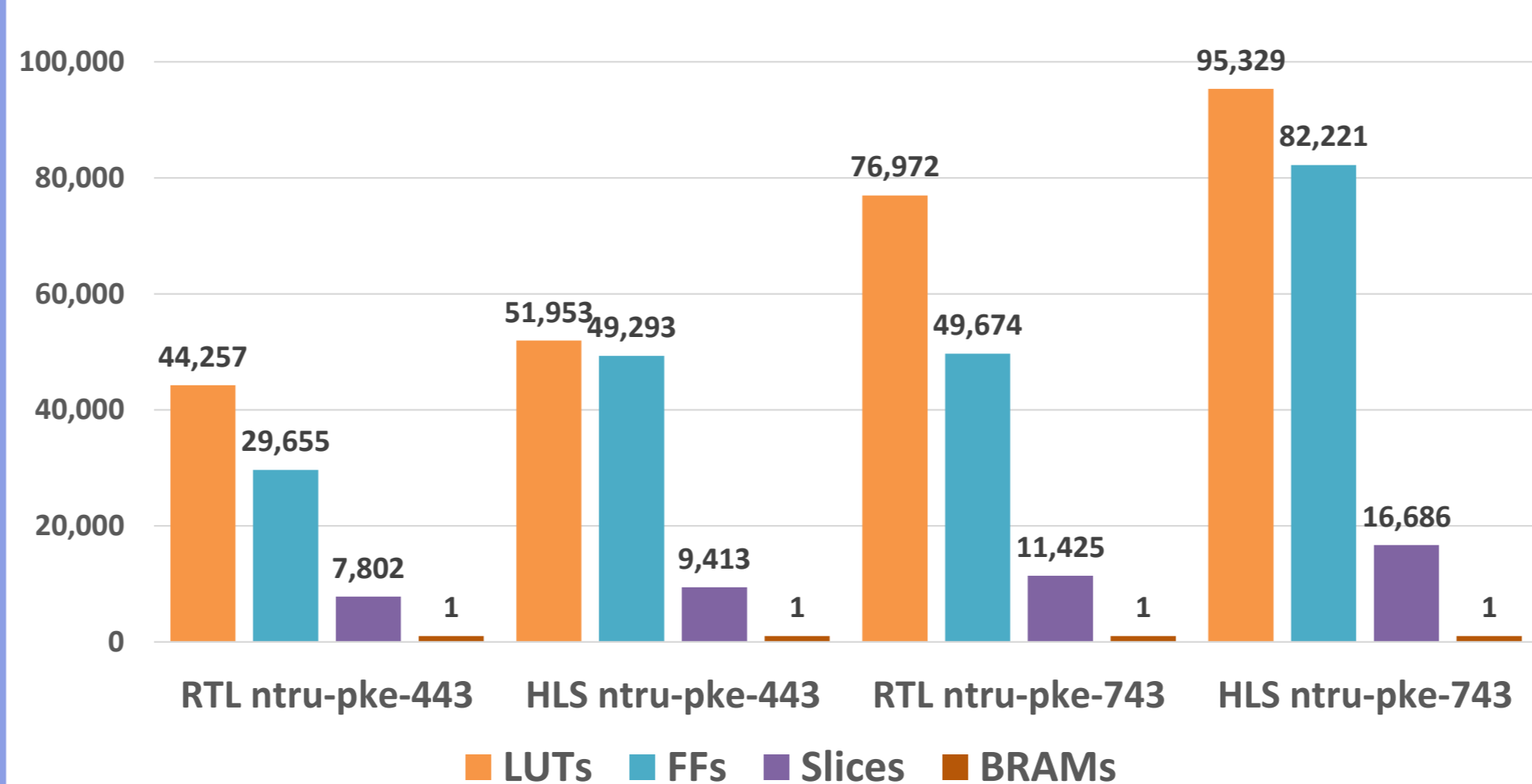
Possible locations of pipeline registers inside of the Rotator for ntru-pke-443

- Rectangles with solid lines represent positions of registers found to be optimal from the point of view of the minimal latency of the RTL and HLS implementations of Poly Mult, expressed in ns.



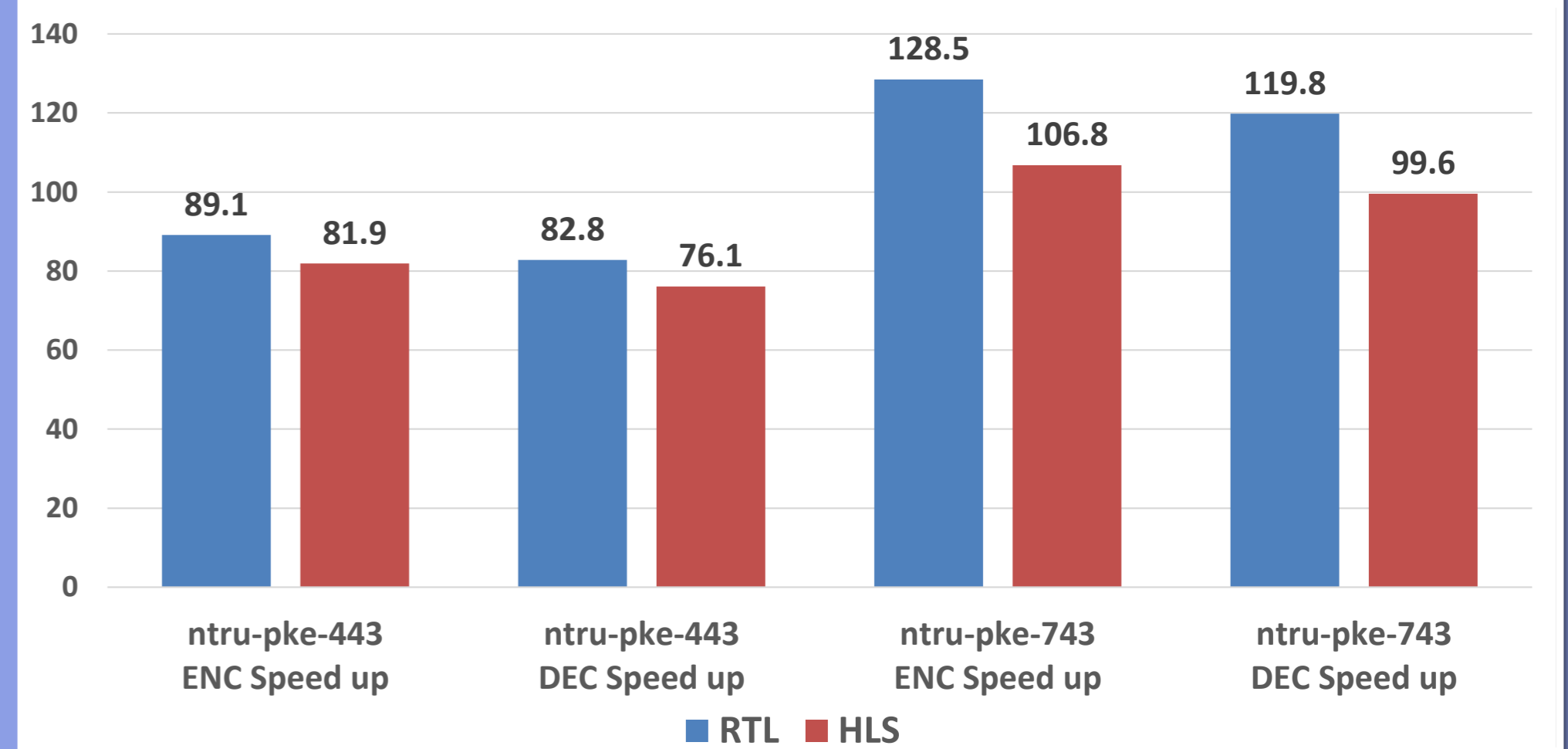
RESULTS

Comparison of the best achieved RTL and HLS designs for Poly Mult in terms of resource utilization

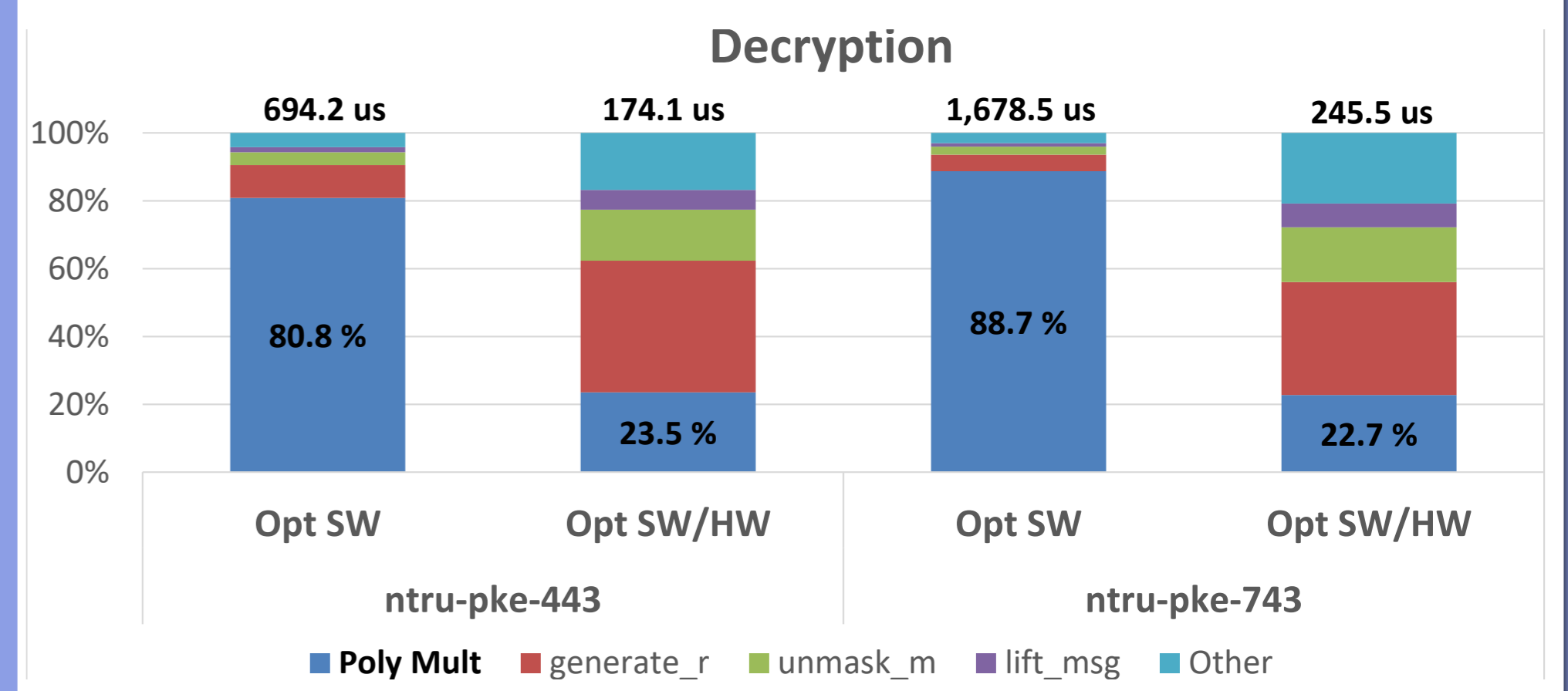
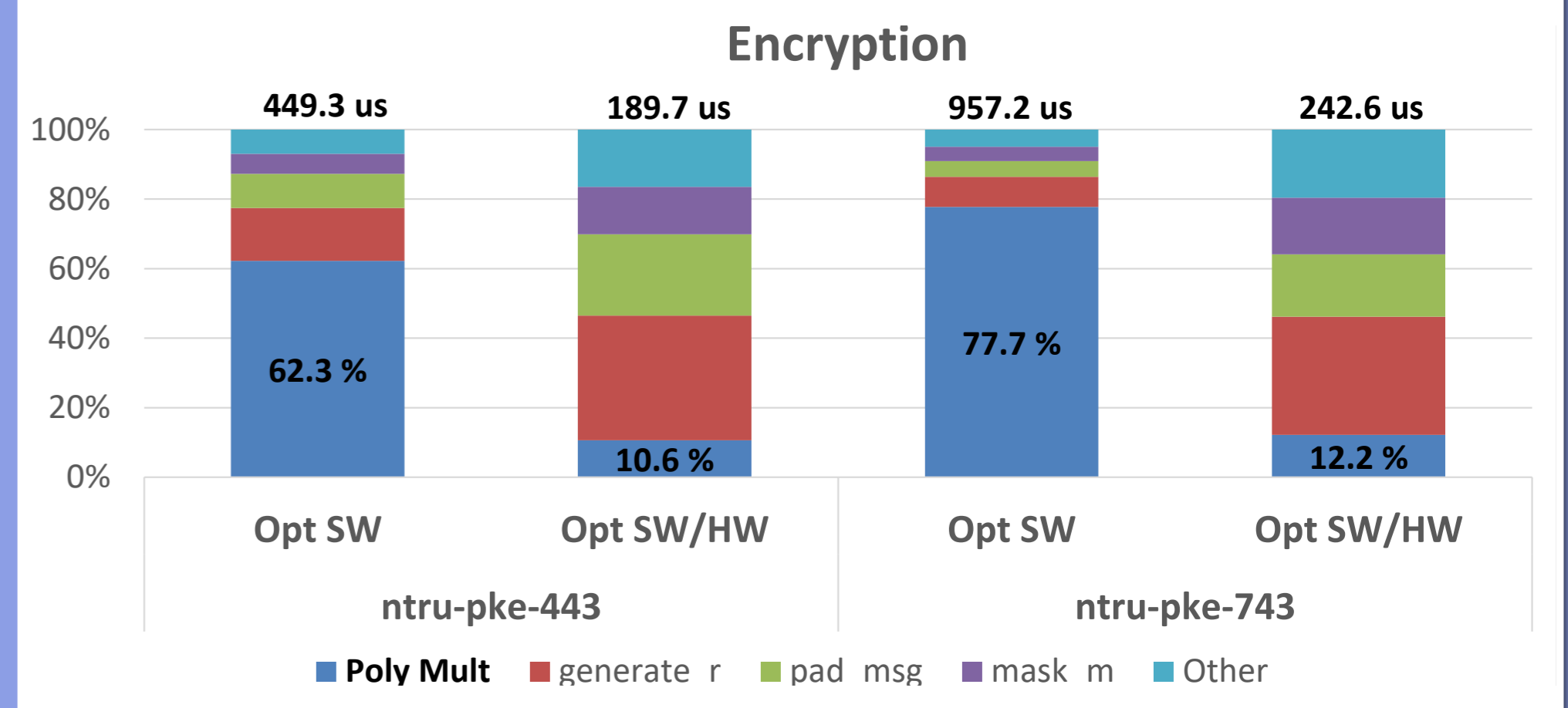


RESULTS

Speed-up achieved for the Polynomial Multiplication operation of encryption (ENC) and decryption (DEC)

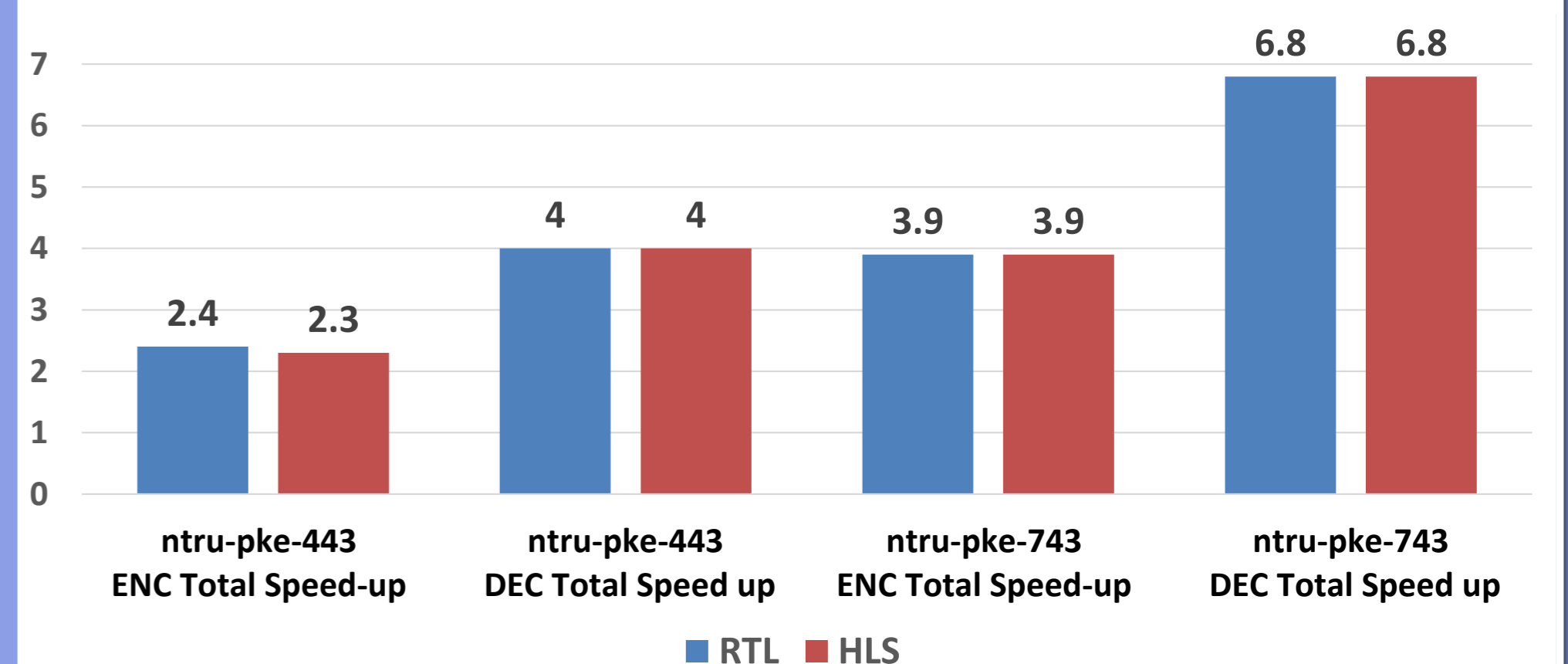


Results of profiling for NTRUencrypt



Speed-up achieved for the entire encryption (ENC) and decryption (DEC)

- Public and private keys are assumed to be precalculated, and preloaded to the appropriate arrays in software and appropriate memories and/or registers in hardware.



CONCLUSIONS

- Using SW/HW codesign allows the implementers of candidates for the NIST PQC standards to **substantially reduce the development time** compared to the use of purely-hardware implementations.
- The implementers **avoid reproducing in hardware the cumbersome and sequential operations**. Instead, they can focus on major operations that are both most **time-consuming** and most **suitable for parallelization**.
- In this study, we have clearly **demonstrated the viability of this approach in case of the Round 1 NIST PQC candidate NTRUencrypt**, and its major operation, Poly Mult.
- We have determined that the use of **HLS vs. RTL** implementation approaches had a **negligible influence on the obtained speed-ups**, while at the same time **provided quite substantial productivity gains**.
- We have identified the **areas of concerns for the HLS based methodology**, such as the need to **almost entirely rewrite the C code** of the accelerated function, as well as **over twice as large use of LUTs and CLB Slices**.

Acknowledgment

This material is based upon work supported by the U.S. Department of Commerce / National Institute of Standards and Technology under Grant no. 60NANB15D058 and Grant no. 70NANB18H218, and by the National Science Foundation under Grant no. CNS-1801512.