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 Ferozpuri and Kris Gaj
Department of Electrical and Computer Engineering, George Mason University, Fairfax, Virginia 22033, 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).

SYSTEM DESIGN
- High-level block diagram of the experimental platform

Results of profiling for NTRUEncrypt

ENC: 64% | DEC: 19.8%

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)
  - 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.
- 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.

SYSTEM DESIGN

Algorithm 1 Polynomial Multiplication, Poly Mult
1. Inputs:
   a) N-1 polynomial a(x) with N "b" coefficients in the range [0, b-1] = [0, 94] in this study
   b) N-1 polynomial b(x) with coefficients in the range [0, b-1] = [0, 94] in this study

2. Outputs:
   c) Polynomial c(x) with coefficients in the range [0, b-1] = [0, 94] in this study

3. Polynomial c(x) with coefficients at the locations b0, b1, ..., bN-1 and of coefficients c0, c1, ..., cN-1 at the locations A0, A1, ..., AN-1, where: 0 ≤ b ≤ N = 1.

4. C(x) = g(x) = a(x)mod c(x) − 1

5. C(x) = g(x) = a(x)mod c(x) − 1

6. For loop over the range 0 ≤ j ≤ N - 1 do:
   a) If j = 0 then:
      i) c(j) = a(j)
      ii) end for
   b) Else:
      i) c(j) = a(j) + c(j)
      ii) end for

7. The for loop in lines 7-15 corresponds to adding to the temporary polynomial c = c−1 + a−1 rotated by b, locations to the left, namely, a = <−c, b = a−1 + b−1 + ... + b−N = −c,

8. 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

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