A High-Level Synthesis Approach to the Software/Hardware Codesign of NTT-based Post-Quantum Cryptography Algorithms

Duc Tri Nguyen, Viet Ba Dang, Kris Gaj

Cryptography Engineering Research Group, George Mason University, Fairfax, VA, U.S.A.

INTRODUCTION

When quantum computers become scalable and reliable, they are likely to break all public-key standards, such as RSA and Elliptic Curve Cryptography.

Due to an emerging threat of quantum computing, one of the major challenges facing the cryptographic community is a timely transition from traditional public-key cryptosystems, such as RSA and Elliptic Curve Cryptography, to a new class of algorithms, collectively referred to as Post-Quantum Cryptography (PQC).

We present an improved hardware architecture for NTT, with the hard-wired polynomial reduction, and demonstrate that this architecture can be efficiently implemented in hardware using High-Level Synthesis (HLS). The novel feature of the proposed architecture is an original memory-write-back scheme, which assists in preparing coefficients for performing later NTT stages, saving memory usage for precomputed constants.

Our design is the most efficient for the case when log₂N is even. The latency of our proposed architecture is approximately equal to (M/log₂N + 3)/4 clock cycles. As a proof of concept, we implemented the NTT operation for several parameter sets used in the PQC algorithms NewHope, FALCON, qTESLA, and CRYSTALS-DILITHIUM.

BACKGROUND

Number Theoretic Transform

By using NTT, a multiplication in \( R_2 \) can be computed as follows:

\[
C = \text{NTT}(C) = \text{NTT}(A) \times \text{NTT}(B)
\]

where \( i^2 = \omega \) and \( a, b, c \) are polynomials in \( R_2 \).

\[
A = (a_0, a_1, a_2, \ldots, a_{n-1}) \quad B = (b_0, b_1, b_2, \ldots, b_{n-1})
\]

\[
\omega = (\omega_0, \omega_1, \ldots, \omega_{n-1})
\]

\[
t = j \quad \text{for } j = 0 \text{ to } m/2 - 1 \text{ do}
\]

\[
f(k) = f(k) \times i \quad \text{for } k = 0 \text{ to } m - 1 \text{ do}
\]

\[
C(k) = C(k) \times (i^k + 1 \mod q)
\]

KRED Modular Reduction

In the reference software implementation, Montgomery modular reduction is used. However, this approach requires performing multiplication before reduction. An alternative approach towards hardware implementations: KRED Modular Reduction.

HARDWARE DESIGN

Our NTT hardware architecture has 2x2 butterfly structure, which processes layers of NTT with two butterfly units per layer. To implement this architecture in High Level Language, we actively avoid stalling, four coefficients are loaded in each clock cycle, without stalling, and placed into registers A, B, C, D. The square boxes \( m_l \) and \( m_r \) are KRED and KRED3s, respectively.

SIPO (Serial In Parallel Out) is a simple Linear Shift Register, used to convert serial input to parallel output. A different number of registers is placed inside of each SIPO to make sure that one SIPO can be full at the same time. The benefits of the SIPO design is its role in preparing coefficients for the execution of the next NTT stages on the fly.

RESULTS

The maximum clock frequency and resource utilization have been generated by performing logic synthesis, planning, and routing using Vivado 2018.3. Our target platform is Zynq UltraScale+ MPSoC. The choice of this platform is consistent with our plans to extend our hardware accelerators for NTT into full software/hardware codesigns of the entire PQC candidates.

The penalty for using HLS in terms of maximum clock frequency and latency varies between 2% and 5%. The overhead in terms of resource utilization is below 14% for all investigated candidates, except qTESLA.

Comparison with the results of the previous implementation of the NTT unit for selected Round 2 PQC Candidates, using Zynq UltraScale+

<table>
<thead>
<tr>
<th></th>
<th>NewHope &amp; Falcon</th>
<th>qTesla</th>
<th>CRYSTALS-DILITHIUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>1024</td>
<td>1024</td>
<td>296</td>
</tr>
<tr>
<td>q</td>
<td>1</td>
<td>1024</td>
<td>215</td>
</tr>
<tr>
<td>DSP</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BRAM</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LUT</td>
<td>256</td>
<td>22</td>
<td>14</td>
</tr>
<tr>
<td>Slice</td>
<td>1.07</td>
<td>1.31</td>
<td>1.31</td>
</tr>
<tr>
<td>Freq.</td>
<td>0.76</td>
<td>0.99</td>
<td>0.97</td>
</tr>
<tr>
<td>Cycles</td>
<td>1.32</td>
<td>1.36</td>
<td>1.30</td>
</tr>
<tr>
<td>Latency(billion)</td>
<td>2.78</td>
<td>2.92</td>
<td>2.96</td>
</tr>
</tbody>
</table>

Conclusions and Future Work

We have proposed a modified hardware architecture of one of the major operations of several Round 2 lattice-based PQC candidates - Number Theoretical Transform. We then implemented and benchmarked this architecture using both the novel approach based on High-Level Synthesis and using the traditional RTL-based approach.

Our results confirm the potential of the HLS-based approach to efficiently implement at least major building blocks of PQC algorithms, if not all the entire algorithms themselves. The obtained results are comparable to those obtained using the RTL approach in terms of timing, and leg behind only in terms of area.

We will also attempt full hardware implementations of all Round 2 PQC candidates that could potentially benefit from NTT, using both the HLS- and RTL-based design methodologies.

Acknowledgment

This material is based upon work supported by the U.S. Department of Commerce / National Institute of Standards and Technology under contract no. 60ANB11K0218, and by the National Science Foundation under grant no. CNS-1801512.