ECE 545 Project
Specification

Hardware Implementation of Signature Verification in the New Post-Quantum Signature Scheme MAYO

Stewart Schuler

Develop a hardware implementation of major operations of the MAYO.Verify function (other than those involving SHAKE256) described below using the pseudocode (without lines 15-17 and 28-31):

Algorithm 9 MAYO.Verify(epk, M, sig)

Input: Expanded public key epk $\in \mathcal{B}^{\text{epk\_bytes}}$
Input: Message M $\in \mathcal{B}^*$
Input: Signature sig $\in \mathcal{B}^{\text{sig\_bytes}}$
Constant: $\mathbf{E} \in \mathbb{F}_q^{m \times m}$ # Represents multiplication by z in $\mathbb{F}_q[z]/(f(z))$
Output: An integer result to indicate if sig is valid (result $= 0$) or invalid (result < 0).

1: //Decode epk.
2: P1_bytestring $\leftarrow$ epk$[0 :$ P1_bytes$]$
3: P2_bytestring $\leftarrow$ epk[P1_bytes : P1_bytes + P2_bytes]
4: P3_bytestring $\leftarrow$ epk[P1_bytes + P2_bytes : P1_bytes + P2_bytes + P3_bytes]
5: $\{\mathbf{P}_i^{(1)}\}_{i\in[m]} \leftarrow \text{Decode}_{\mathbf{P}^{(1)}}(\text{P1\_bytestring})$ // $\mathbf{P}_i^{(1)} \in \mathbb{F}_q^{(n-o)\times(n-o)}$ upper triangular
6: $\{\mathbf{P}_i^{(2)}\}_{i\in[m]} \leftarrow \text{Decode}_{\mathbf{P}^{(2)}}(\text{P2\_bytestring})$ // $\mathbf{P}_i^{(2)} \in \mathbb{F}_q^{(n-o)\times o}$
7: $\{\mathbf{P}_i^{(3)}\}_{i\in[m]} \leftarrow \text{Decode}_{\mathbf{P}^{(3)}}(\text{P3\_bytestring})$ // $\mathbf{P}_i^{(3)} \in \mathbb{F}_q^{o\times o}$ upper triangular
8:
9: //Decode sig.
10: salt $\leftarrow$ sig$[\lceil nk/2 \rceil : \lceil nk/2 \rceil +$ salt_bytes$]$
11: $\mathbf{s} \leftarrow \text{Decode}_{vec}(kn, \text{sig})$
12: **for** i from 0 to $k-1$ **do**
13: $\quad \mathbf{s}_i \leftarrow \mathbf{s}[i*n : (i+1)*n]$
14:
15: //Hash message and derive $\mathbf{t}$.
16: M_digest $\leftarrow$ SHAKE256(M, digest_bytes) // M_digest $\in \mathcal{B}^{\text{digest\_bytes}}$
17: $\mathbf{t} \leftarrow \text{Decode}_{vec}(m, \text{SHAKE256}(\text{M\_digest} \,\|\, \text{salt}, \lceil m \log(q)/8 \rceil))$ // $\mathbf{t} \in \mathbb{F}_q^m$
18:
19: //Compute $\mathcal{P}^*(\mathbf{s})$.
20: $\mathbf{y} \leftarrow \mathbf{0}_m$ // $\mathbf{y} \in \mathbb{F}_q^m$
21: $\ell \leftarrow 0$
22: **for** i from 0 to $k-1$ **do**
23: $\quad$ **for** j from $k-1$ to i **do**
24: $\quad\quad \mathbf{u} \leftarrow \begin{cases} \left\{ \mathbf{s}_i^\mathsf{T} \begin{pmatrix} \mathbf{P}_a^{(1)} & \mathbf{P}_a^{(2)} \\ \mathbf{0} & \mathbf{P}_a^{(3)} \end{pmatrix} \mathbf{s}_i \right\}_{a\in[m]} & \text{if } i = j \\ \left\{ \mathbf{s}_i^\mathsf{T} \begin{pmatrix} \mathbf{P}_a^{(1)} & \mathbf{P}_a^{(2)} \\ \mathbf{0} & \mathbf{P}_a^{(3)} \end{pmatrix} \mathbf{s}_j + \mathbf{s}_j^\mathsf{T} \begin{pmatrix} \mathbf{P}_a^{(1)} & \mathbf{P}_a^{(2)} \\ \mathbf{0} & \mathbf{P}_a^{(3)} \end{pmatrix} \mathbf{s}_i \right\}_{a\in[m]} & \text{if } i \neq j \end{cases}$ // $\mathbf{u} \in \mathbb{F}_q^m$
25: $\quad\quad \mathbf{y} \leftarrow \mathbf{y} + \mathbf{E}^\ell \mathbf{u}$
26: $\quad\quad \ell \leftarrow \ell + 1$
27:
28: //Accept signature if $\mathbf{y} = \mathbf{t}$.
29: **if** $\mathbf{y} = \mathbf{t}$ **then**
30: $\quad$ **return** 0
31: **return** -1

The notation used in this pseudocode, the data types, and the lower-level functions are explained in the specification of MAYO, available at

https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures

under

Multivariate Signatures.

See, in particular,

- 2.1.1 Parameters
- 2.1.2 Preliminaries and notation
- 2.1.4 Data types and conversions
- Table 2.1: Our selection of parameter sets for MAYO.

The reference implementations of these functions in C are available in the MAYO submission package, which can be downloaded from the above web page by clicking on Zip file (19 MB).

See:

Reference_Implementation/mayo.c.

Locate the lower-level functions.

Assume optimization for the minimum execution time in time units, assuming the availability of the largest and fastest device of the Xilinx Artix-7 FPGA family.

Your project should involve at least the following steps:

1. Full understanding of the specifications of MAYO.Verify, including the notation, basic operations, pseudocode, parameters, etc.
2. Full understanding of the corresponding reference implementation in C.
3. Choice of an interface for a hardware unit capable of implementing MAYO.Verify without lines 15-17 and 28-31 of the pseudocode.
4. Block diagram of the Datapath.
5. Interface divided into the Datapath and Controller.
6. RTL VHDL code of the Datapath.
7. Testbench for the Datapath.
8. Simulation, verification, debugging, synthesis, and optimization of the Datapath.
9. ASM chart of the Controller.
10. RTL VHDL code of the Controller and Top-Level Unit.
11. Testbench for the Top-Level Unit.
12. Simulation, verification, debugging, synthesis, and optimization of the Top-Level Unit.
13. Timing analysis and simulation aimed at determining the latency of the implemented circuit in clock cycles.
14. Implementation and timing simulation.
15. Determining the maximum clock frequency, execution time in time units, and resource utilization.

Important Assumption: By default, implementing one parameter set, e.g., $MAYO_1$, is sufficient. Bonus points will be awarded for supporting multiple parameter sets at the time of the circuit synthesis or at runtime.