Hardware Accelerator for CKKS Encryption

Athithan Elamaran[1,2], Lohit Daksha[2], Seyda Guzelhan[2], Ajay Joshi[2]

Foothill High School, 4375 Foothill Rd, Pleasanton, CA 94588[1],

Department of Electrical and Computer Engineering, Boston University, 8 St. Mary's Street, Boston, MA 022152[2]

athithan.elamaran@gmail.com

Introduction

- Fully Homomorphic Encryption (FHE) enables computation on encrypted data

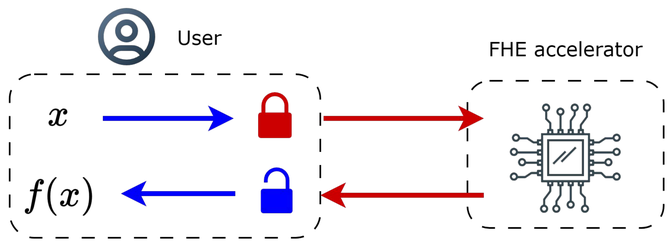


Figure 1. FHE enables direct computation on encrypted data

- CKKS scheme supports arithmetic on real-valued inputs, paving the way for privacy-preserving ML
- NTT is a key component in polynomial multiplication

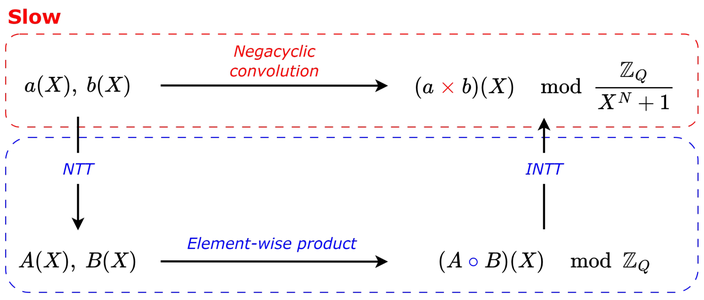


Figure 2. Polynomial multiplication in coefficient and evaluation representations

What is an FPGA?

- FPGA offers parallelism and low-latency pipelines, ideal for accelerating NTT
- Xilinx Zynq SoC combines an ARM CPU for control with FPGA for high-throughput compute
- Enables tight CPU- FPGA integration via AXI interconnect

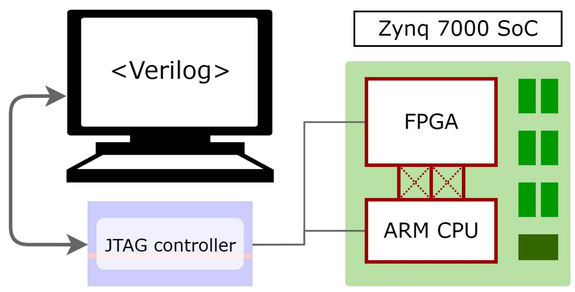


Figure 3. Hardware-software co-design using a Zynq SoC

Pipelining

- Dividing a circuit into discrete stages with intermediate registers so multiple operations can overlap, increasing overall throughput

Objectives

- Design a hardware-software co-implementation of CKKS encryption on a Xilinx Zynq SoC that offloads the NTT stage to FPGA fabric
- Investigate combinational vs. pipelined NTT accelerator designs to assess trade-offs between performance and area

Methods

- Baseline Design
  - Entire NTT computed in software using ARM Cortex-A9

NTT Hardware Accelerator Variants

- Single-Cycle Combinational NTT (CN)
  - Purely combinational circuit, with 3×8 butterfly
- Pipelined NTT with 3×8 Butterfly Row (PN3)
  - All 3 NTT stages are implemented as separate layers with registers in between for optimized frequency
- Pipelined NTT with 1×8 Butterfly Row (PN1)
  - Reuses a single row of 4 butterflies across all stages, with a bit-reversal shuffle network between stages

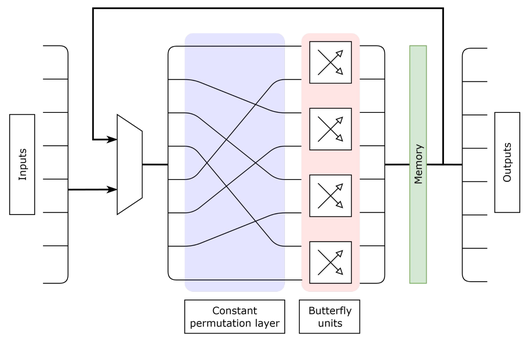


Figure 5. Single stage sequential circuit of an 8-point NTT using Cooley-Tukey butterflies

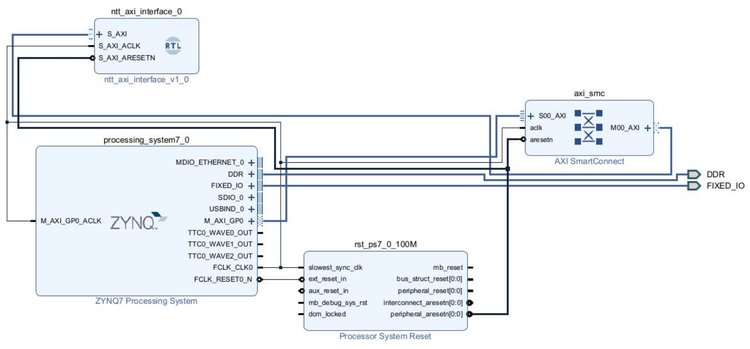


Figure 6. Block diagram from Vivado

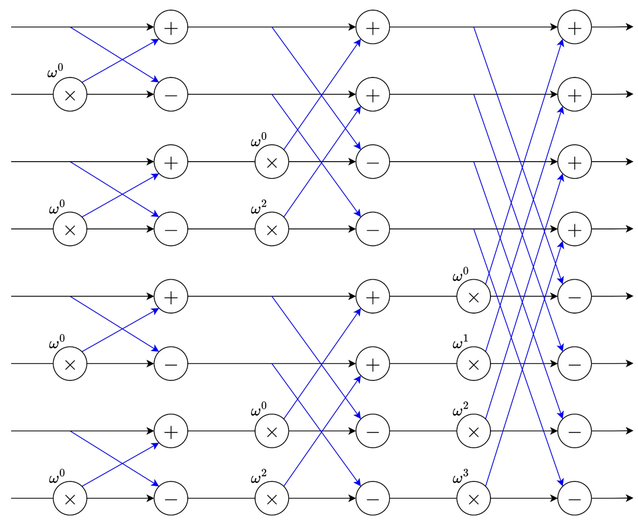


Figure 4. Fully unrolled combinational circuit of an 8-point NTT using Cooley-Tukey butterflies

Hardware-Software Partition

- CPU handles:
  - Coefficient arithmetic and randomness sampling.
- FPGA handles:
  - Forward/inverse NTT (radix-2 Cooley-Tukey and Gentleman-Sande) for Negacyclic Polynomial Multiplication
  - Optimized modular multiplication via Barrett Reduction

Integration and Communication

- The ARM CPU configures and triggers the NTT accelerator using an AXI-Lite control interface, and transfers polynomial data via memory-mapped AXI registers

Implementation Flow

- Hardware accelerators were designed in Verilog, synthesized and placed using Vivado, and integrated with the software stack in Vitis

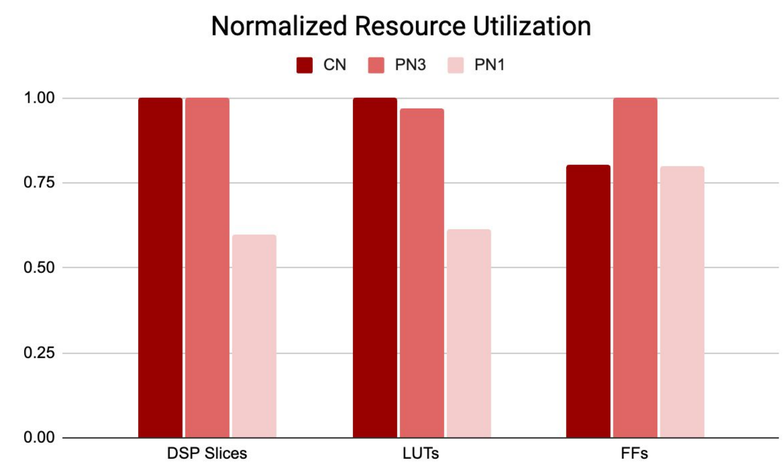
Results

CKKS Encryption Time

ARM	FPGA (CN)	Speedup
33.21 µs	17.42 µs	**1.91x**

NTT Implementation Metrics

	Frequency	Latency	Input Size
CN	10.82 MHz	1	1
PN3	39.54 MHz	3	3
PN1	23.40 MHz	3	2



Discussions/Conclusion

Benefits and Limitations of FPGAs

- FPGA allows for easy reconfiguration, allowing for flexibility across different algorithms
- Enables real-time encryption for high-throughput systems

Limitations of FPGAs

- Limited on-chip resources restrict the size of inputs

Impact of FPGA Offloading

- Offloading the NTT to FPGA fabric greatly reduces encryption latency compared to CPU-only execution
- FPGA offload enables higher throughput by exploiting instruction level parallelism in modular arithmetic

NTT Design Trade-Offs

- Pipelined designs improve throughput at the cost of performance by overlapping butterfly stages
- Pipelined designs also require greater resource utilization

Future Steps

- Use our pipelined NTT designs in the full encryption process to overlap FPGA compute with CPU-side processing, reduce idle time, and boost overall throughput.

Conclusion

- We successfully accelerated polynomial multiplication in the encryption step on an FPGA and got an overall speedup of **1.91x**

References

[1] Agrawal, R.; de Castro, L.; Yang, G.; Juvekar, C.; Yazicigil, R.; Chandrakasan, A.; Vaikuntanathan, V.; Joshi, A. FAB: An FPGA-based Accelerator for Bootstrappable Fully Homomorphic Encryption. https://ieeexplore.ieee.org/document/10070953

[2] Liang, Z.; Zhao, Y. Number Theoretic Transform and Its Applications in Lattice-based Cryptosystems: A Survey. https://arxiv.org/pdf/2211.13546

Acknowledgements

I would like to thank Professor Ajay Joshi for his guidance in my research and for the opportunity to learn in his lab. I would also like to thank Lohit Daksha and Seyda Guzelhan for mentoring and teaching me throughout the research process.