Project Title: FPGA-Based Hardware Acceleration of NTT with RISC-V ISA Customization

Note to students: This project is a Volunteer Mentorship by Dr. Hossam Hassan
“I'm offering this project as a volunteer opportunity to help you gain practical experience with cutting-edge technology. I'm passionate about [FPGA/Embedded Systems/ RISC-V/ Cryptography] and want to share my knowledge and mentor students who are eager to learn and build their skills.”

Objective

The objective of this project is to design and implement an accelerated Number Theoretic Transform (NTT) operation integrated with RISC-V through custom ISA extensions on an FPGA. The goal is to significantly enhance the performance of NTT, which is a critical and time-consuming operation in many cryptographic protocols and proof-generation processes.

Project Scope and Breakdown

2 Project Scope

The project aims to enhance the performance of Number Theoretic Transform (NTT) operations by designing a hardware accelerator integrated with a RISC-V processor and extending its Instruction Set Architecture (ISA) to efficiently support NTT operations on an FPGA platform. The main components of the project include:

1. Research and Analysis:
   - Gain an in-depth understanding of the NTT algorithm and its computational bottlenecks.
   - Study the RISC-V architecture and the process required to extend its ISA.
   - Review existing FPGA-based accelerators and identify suitable development tools and platforms.
2. Design and Implementation:
   - Design the NTT accelerator using SystemVerilog or High-Level Synthesis (HLS).
   - Develop custom RISC-V instructions tailored for NTT operations.
   - Integrate the NTT accelerator with a RISC-V core on an FPGA.
3. Testing and Validation:

- Create comprehensive testbenches to verify the functionality of the NTT accelerator.
- Validate the integration of custom RISC-V instructions with the core processor.
- Benchmark the performance improvements compared to a software-only implementation.

4. Documentation and Presentation:
    - Document the design, implementation, and testing processes in detail.
    - Prepare a final report summarizing the project.
    - Develop a presentation and demonstration of the project outcomes.

3 Project Breakdown

1. Research and Planning (Weeks 1-4):
    - Literature Review: Study NTT algorithm, RISC-V architecture, and FPGA design techniques.
    - Tool Selection: Choose appropriate FPGA development tools and platforms.
    - Initial Planning: Define project milestones, deliverables, and risk management strategies.
2. Design Phase (Weeks 5-8):
    - NTT Accelerator Design: Design the NTT algorithm using SystemVerilog or HLS.
    - ISA Extension: Develop custom RISC-V instructions for NTT operations.
    - Integration Strategy: Plan the integration of the NTT accelerator with the RISC-V core.
3. Implementation Phase (Weeks 9-14):
    - NTT Accelerator Implementation: Implement the NTT accelerator on the selected FPGA.
    - RISC-V Core Modification: Extend the RISC-V core to support custom NTT instructions.
    - Integration: Integrate the NTT accelerator with the RISC-V core on the FPGA.
4. Testing and Validation Phase (Weeks 15-18):
    - Functional Testing: Develop and run testbenches to verify the NTT accelerator.
    - ISA Validation: Test the functionality of custom RISC-V instructions.
    - Performance Benchmarking: Measure and compare the performance of the hardware-accelerated NTT against a software implementation.
5. Documentation and Presentation (Weeks 19-22):
    - Documentation: Compile comprehensive documentation covering design, implementation, testing, and results.
    - Final Report: Write a detailed final report summarizing the project.

- Presentation Preparation: Develop slides and prepare a demonstration to present the project outcomes.

4 Detailed Project Requirements

1. Hardware Requirements
    1. FPGA Development Board:
        - Choose a suitable FPGA development board that supports custom hardware design and integration with a processor core. Examples include:
            - Xilinx Zynq-7000 series
            - Intel (Altera) Cyclone V
            - Digilent Arty A7 (for Xilinx Artix-7)
        - Ensure the board has sufficient logic elements, memory, and I/O capabilities for the project.
    2. RISC-V Core:
        - Select an appropriate RISC-V core that supports customization and extension. Examples include:
            - Rocket Chip from the Berkeley Architecture Research group
            - PicoRV32 for a lightweight implementation
            - VexRiscv for a more configurable core
        - Ensure compatibility with the chosen FPGA platform.
2. Software Requirements
    1. FPGA Development Suite:
        - Xilinx Vivado (for Xilinx FPGAs) or Intel Quartus (for Intel FPGAs):
            - Used for synthesis, place-and-route, and generating the bitstream.
        - Vivado HLS or Vitis HLS (for High-Level Synthesis if using C/C++ for hardware design).
    2. RISC-V Toolchain:
        - RISC-V GNU Compiler Toolchain:
            - For compiling and assembling code for the RISC-V core.
        - Spike Simulator or QEMU:
            - For simulating and debugging RISC-V code.
    3. Hardware Description Languages (HDL):
        - SystemVerilog or VHDL:
            - For designing the NTT accelerator and integrating it with the RISC-V core.
        - High-Level Synthesis (HLS) Tools (optional):

        - If using HLS, tools like Xilinx Vitis HLS can be employed to convert C/C++ code into HDL.
4. Simulation and Verification Tools:
    - ModelSim or Xilinx Vivado Simulator:
        - For simulating and verifying the HDL designs.
    - Synopsys VCS (optional):
        - For advanced simulation capabilities.

3. Knowledge Requirements

1. Digital Design and FPGA Programming:
    - Understanding of digital logic design, finite state machines, and FPGA architecture.
    - Proficiency in a hardware description language (SystemVerilog/VHDL).
2. RISC-V Architecture and ISA Extensions:
    - Familiarity with the RISC-V instruction set architecture, including base ISA and extension mechanisms.
    - Knowledge of how to modify and extend the RISC-V ISA to include custom instructions.
3. Number Theoretic Transform (NTT) Algorithm:
    - In-depth knowledge of the NTT algorithm, its mathematical foundations, and its computational requirements.
4. High-Level Synthesis (Optional):
    - Experience with HLS tools if using C/C++ to design the accelerator.

4. Additional Resources and References

1. Documentation and Tutorials:
    - RISC-V Specifications: RISC-V ISA Specifications
    - FPGA Vendor Documentation: Xilinx or Intel FPGA documentation and user guides.
    - NTT Algorithm Resources: Research papers, textbooks, and online tutorials on NTT and related mathematical concepts.
2. Development Kits:
    - FPGA Development Kits: Ensure access to development kits and necessary accessories (e.g., power supplies, cables).
3. Collaboration and Support:
    - Access to forums, support communities, and potential collaboration with peers or mentors specializing in FPGA design and RISC-V.

5 Project Deliverables

1. Design Documentation
   1. NTT Accelerator Design:
      - Detailed design documents outlining the architecture and implementation of the NTT accelerator.
      - Block diagrams, data flow diagrams, and state machine diagrams as applicable.
      - Explanation of the chosen algorithmic optimizations and their impact on performance.
   2. RISC-V ISA Extension:
      - Documentation of the custom RISC-V instructions developed for NTT operations.
      - Instruction format, encoding, and integration details.
      - Rationale for the custom instructions and their expected performance benefits.
   3. Integration Strategy:
      - Detailed plan for integrating the NTT accelerator with the RISC-V core.
      - Interface specifications between the accelerator and the RISC-V core.
      - Memory and I/O considerations for the integrated system.
2. Implementation
   1. Hardware Description Code:
      - SystemVerilog or VHDL code for the NTT accelerator.
      - Modified RISC-V core with custom ISA extensions.
      - Integration code to interface the NTT accelerator with the RISC-V core.
      - HLS code (if applicable) for the NTT accelerator.
   2. Simulation and Synthesis Files:
      - Testbenches for simulating the NTT accelerator and verifying functionality.
      - Synthesis scripts and configuration files for the FPGA development tools.
      - FPGA bitstream file for programming the FPGA with the final design.
3. Testing and Validation
   1. Testbenches:
      - Comprehensive testbenches for verifying the functional correctness of the NTT accelerator.
      - Test cases for validating the custom RISC-V instructions.
      - Integration testbenches to ensure the NTT accelerator works correctly with the RISC-V core.
   2. Validation Reports:
      - Detailed reports on the results of functional testing, including pass/fail status for each test case.

        - Performance benchmarking results comparing the hardware-accelerated NTT operations to a software-only implementation.
        - Analysis of resource utilization on the FPGA, including logic elements, memory blocks, and power consumption.

4. Final Report
    1. Project Overview:
        - Introduction and background information on NTT and RISC-V.
        - Objectives and scope of the project.
    2. Design and Implementation:
        - Detailed description of the design and implementation process.
        - Challenges encountered and how they were addressed.
    3. Testing and Results:
        - Summary of testing methodologies and results.
        - Performance analysis and comparison with baseline software implementation.
    4. Conclusion and Future Work:
        - Summary of key findings and achievements.
        - Potential areas for future improvement and research.
5. Presentation
    1. Slides:
        - A comprehensive slide deck covering all aspects of the project, including introduction, design, implementation, testing, results, and conclusions.
        - Visual aids such as diagrams, charts, and graphs to illustrate key points.
    2. Demonstration:
        - Live demonstration of the working FPGA implementation showing the accelerated NTT operations.
        - Explanation of the custom RISC-V instructions and how they enhance performance.
        - Performance comparison between hardware-accelerated and software-only implementations.
6. Additional Materials (Optional)
    1. Video Presentation:
        - A recorded video presentation summarizing the project and demonstrating the key results.
        - Can be used as supplementary material for the final presentation.
    2. Code Repository:
        - A well-organized code repository (e.g., GitHub) containing all the source code, design files, and documentation.

- Instructions for setting up the development environment and reproducing the results.

6 Potential Challenges

1. Complexity of NTT Algorithm
    1. Mathematical Complexity:
        - Understanding and implementing the NTT algorithm correctly, including modular arithmetic and handling large prime numbers, can be challenging.
        - Ensuring that the algorithm is optimized for hardware implementation without sacrificing correctness.
    2. Resource Optimization:
        - Balancing the trade-offs between speed, area, and power consumption in the FPGA implementation.
        - Efficiently mapping the NTT operations to FPGA resources like DSP blocks, BRAM, and LUTs.
2. RISC-V ISA Extension
    1. Instruction Design:
        - Designing custom RISC-V instructions that effectively accelerate NTT operations while maintaining the simplicity and orthogonality of the RISC-V ISA.
        - Ensuring that the new instructions integrate seamlessly with the existing RISC-V pipeline and do not introduce hazards or performance bottlenecks.
    2. Toolchain Modifications:
        - Modifying the RISC-V toolchain (assembler, compiler, and simulator) to support the custom instructions.
        - Ensuring that the modified toolchain produces correct and optimized code for the new instructions.
3. Integration with RISC-V Core
    1. Hardware Integration:
        - Integrating the NTT accelerator with the RISC-V core requires careful design to ensure correct data transfer and synchronization.
        - Handling interface protocols between the accelerator and the processor, such as memory-mapped I/O or custom co-processor interfaces.
    2. Performance Bottlenecks:
        - Identifying and mitigating potential performance bottlenecks introduced by the integration, such as data transfer delays or pipeline stalls.
        - Ensuring that the overhead of invoking custom instructions does not negate the performance benefits of hardware acceleration.
4. FPGA Constraints

1. Resource Utilization:
    - Managing FPGA resource constraints, including logic elements, memory blocks, and I/O pins, especially if the design is complex and resource-intensive.
    - Ensuring the design meets timing requirements and fits within the available FPGA resources.
2. Debugging and Verification:
    - Debugging hardware designs can be challenging, especially when dealing with complex interactions between the processor and the accelerator.
    - Developing comprehensive testbenches and validation strategies to ensure the design functions correctly under all conditions.

5. Performance Optimization

1. Algorithmic Optimization:
    - Optimizing the NTT algorithm for parallel execution in hardware while maintaining accuracy and robustness.
    - Balancing between the depth of pipelining and the latency of individual operations to achieve optimal performance.
2. Benchmarking and Validation:
    - Accurately measuring and comparing the performance of the hardware-accelerated NTT against a software-only implementation.
    - Ensuring that the performance gains are significant and justify the complexity of the hardware design.

6. Learning Curve and Development Time

1. Tool Proficiency:
    - Gaining proficiency with FPGA development tools (e.g., Xilinx Vivado, Intel Quartus) and RISC-V toolchain modifications can take time and effort.
    - Staying up-to-date with the latest developments in RISC-V and FPGA technologies.
2. Project Management:
    - Managing the project timeline effectively to ensure that all milestones are met, and potential risks are mitigated.
    - Allocating sufficient time for testing, debugging, and optimization phases.

7 Project Timeline (Adjustable)

Phase 1: Planning and Research (Weeks 1-3)
Week 1:

- Project Kickoff:

    - Define project objectives and scope.
    - Identify key deliverables and success criteria.
- Research:
    - Study the Number Theoretic Transform (NTT) algorithm.
    - Review RISC-V architecture and ISA extension mechanisms.
    - Gather resources and reference materials.

Week 2:

- Hardware and Tools Selection:
    - Choose the FPGA development board and RISC-V core.
    - Set up an FPGA development environment (Vivado/Quartus).
    - Set up the RISC-V toolchain (GCC, Spike/QEMU).

Week 3:

- Detailed Project Plan:
    - Develop a detailed project plan with tasks, milestones, and deadlines.
    - Identify potential risks and mitigation strategies.

Phase 2: Design and Specification (Weeks 4-6)

Week 4:

- NTT Accelerator Design:
    - Develop the high-level architecture of the NTT accelerator.
    - Create block diagrams and data flow diagrams.
- RISC-V ISA Extension:
    - Define custom instructions for NTT operations.
    - Document the instruction format and encoding.

Week 5:

- Integration Planning:
    - Plan the integration strategy between the NTT accelerator and the RISC-V core.
    - Define the interface specifications (e.g., memory-mapped I/O).

Week 6:

- Documentation:
    - Complete the design and specification documentation.
    - Review and finalize the design documents.

Phase 3: Implementation (Weeks 7-12)

Week 7-8:

- NTT Accelerator Implementation:
    - Start coding the NTT accelerator in SystemVerilog/VHDL.
    - Develop initial testbenches for individual modules.

Week 9:

- RISC-V Core Modification:

    - Modify the RISC-V core to support custom ISA extensions.
    - Update the toolchain to recognize new instructions.

Week 10-11:

- Integration:
    - Integrate the NTT accelerator with the modified RISC-V core.
    - Implement the interface logic and data transfer mechanisms.

Week 12:

- Testing and Debugging:
    - Conduct initial testing of the integrated system.
    - Debug and resolve any issues encountered during integration.

Phase 4: Testing and Validation (Weeks 13-16)

Week 13:

- Functional Testing:
    - Develop comprehensive testbenches for the NTT accelerator and custom instructions.
    - Perform functional testing to verify correctness.

Week 14:

- Performance Benchmarking:
    - Benchmark the hardware-accelerated NTT operations against a software-only implementation.
    - Analyze performance improvements and resource utilization.

Week 15:

- Optimization:
    - Optimize the design for better performance and resource efficiency.
    - Iterate on design improvements based on benchmarking results.

Week 16:

- Validation and Final Testing:
    - Conduct final validation tests to ensure system stability and performance.
    - Prepare detailed validation reports.

Phase 5: Documentation and Presentation (Weeks 17-20)

Week 17:

- Final Report Preparation:
    - Compile all design, implementation, and testing documentation.
    - Write the final project report, including introduction, design details, testing results, and conclusions.

Week 18:

- Presentation Preparation:
    - Develop a comprehensive slide deck for the final presentation.

    - Prepare visual aids, diagrams, and performance charts.

Week 19:

- Final Review:
    - Review and refine the final report and presentation materials.
    - Conduct a mock presentation to gather feedback.

Week 20:

- Final Presentation:
    - Deliver the final presentation to the project committee.
    - Demonstrate the working FPGA implementation and discuss results.

Phase 6: Wrap-up and Submission (Week 21)

Week 21:

- Submission:
    - Submit the final report, code repository, and any additional materials.
    - Ensure all documentation and deliverables are complete and well-organized.

Resources

1. NTT | NTT Overview (hardcaml.com)
2. Hardware acceleration of number theoretic transform for zk-SNARK - Zhao - Engineering Reports - Wiley Online Library
3. ZKP MOOC Lecture 16: Hardware Acceleration of ZKP
4. tilk/riscv-simple-sv: A simple RISC V core for teaching (github.com)
5. HiggsBose/RiscV_CPU_with_Accelerator: A RiscV CPU with an accelerator for accelerating neural networks attached to it (github.com)
6. openhwgroup/cv32e40p: CV32E40P is an in-order 4-stage RISC-V RV32IMFCXpulp CPU based on RI5CY from PULP-Platform (github.com)
7. PulseRain/PulseRain_RISCV_MCU: PulseRain RISC-V MCU (github.com)
8. nitinm694/Tensor_Core_RISCV_VP: RISCV Vector Extension ISA to accelerate Deep Learning (github.com)
9. ssayin/riscv32-cosim-model: RISC-V processor co-simulation using SystemVerilog HDL and UVM. (github.com)
10. shrutiprakashgupta/RISCV_Formal_Verification: Formal Verification of RISC V IM Processor (github.com)

Conclusion

The successful completion of this project on accelerating Number Theoretic Transform (NTT) operations through RISC-V integration and ISA extension on an FPGA demonstrates the significant potential of custom hardware accelerators to enhance computational efficiency. By leveraging the flexibility of the RISC-V architecture and the parallel processing capabilities of

FPGAs, we were able to achieve notable performance improvements over traditional software implementations.

This project underscores the transformative impact of hardware acceleration on computational performance. By integrating custom accelerators with a flexible and extendable architecture like RISC-V, significant advancements can be made in specialized fields such as cryptography and digital signal processing. The insights and outcomes from this project lay a strong foundation for further exploration and development in hardware-software co-design and high-performance computing.
The journey through this project not only enhanced your technical skills and knowledge but also reinforced the importance of innovation and adaptability in tackling complex engineering challenges.