Hardware Acceleration

Team Members

Fiona O'Connell	oconnefa@mail.uc.edu	CS
Kelly Deal	dealka@mail.uc.edu	CS
Danni Krauser	krausedc@mail.uc.edu	CS

Project Advisor:

Prof. Yu Zhao	zhao3y3@ucmail.uc.edu

Hardware Acceleration: benefits of migrating compute-intensive software to hardware

- Purpose
  - to investigate how migrating a compute-intensive program from a software-only implementation to a hardware-accelerated design can improve performance
- Goals
  - Develop a CPU-only baseline of a compute-intensive algorithm and analyze performance bottlenecks
  - Implement a hardware-accelerated version on an FPGA
  - Measure and compare performance metrics, including execution time, throughput, and energy efficiency, between software and hardware implementations to quantify acceleration benefits and trade-offs

Project Abstract

This project will investigate the performance benefits of migrating compute-intensive algorithms from a software-only implementation to a hardware accelerator. We will first develop a benchmark CPU-only version to run on the 4 core ARM processor, analyze the performance and identifying performance bottlenecks. From there, using high-level synthesis (HLS), the algorithm will be converted to VHDL, simulated, and integrated with an ARM processor. The design will then be implemented on an FPGA, with an application developed to handle data transfers between the CPU and hardware accelerator. Finally, we will analyze and visualize the performance metrics to quantify improvements, and discuss the advantages and trade-offs of hardware acceleration.

Intellectual Merits

– This project demonstrates the engineering decision-making process of identifying bottlenecks in a C++ algorithm and refactoring them for hardware synthesis.

– Utilizing AXI-Lite for control signaling and AXI-Full for high-bandwidth memory access represents a real-world implementation of modern SoC architecture.

– Using Cocotb (Python) for hardware simulation bridges the gap between traditional software testing and hardware verification, ensuring functional parity before physical deployment.

Broader Impacts

Energy Efficiency: Hardware accelerators can perform the same math as a CPU at a fraction of the clock speed and power, contributing to "Green Computing" initiatives.

Edge Computing: Demonstrating high-performance processing on a System-on-Module (SOM) like the UltraZed-EG has direct applications in many fields such as telemetry systems and real-time medical imaging where space and power are limited.

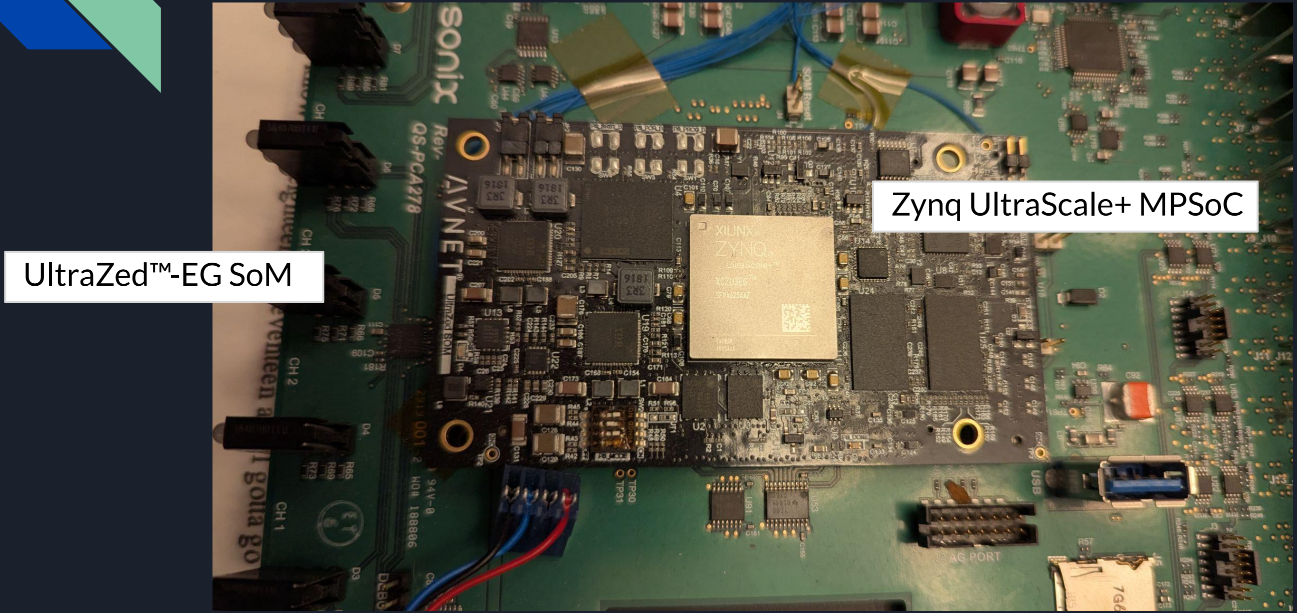
Design Specifications

System Overview: The design utilizes the UltraZed™-EG SOM. The software stack runs on Arch Linux ARM, while the hardware accelerator is implemented in the FPGA fabric.

Design Components:

- Processing System (PS): Manages the OS, file I/O, and the host C++ application.
- Programmable Logic (PL): Houses the custom IP core (the accelerator).
- Interconnect: AXI4 interfaces facilitate data movement between the PS DDR memory and the PL registers.

Technologies

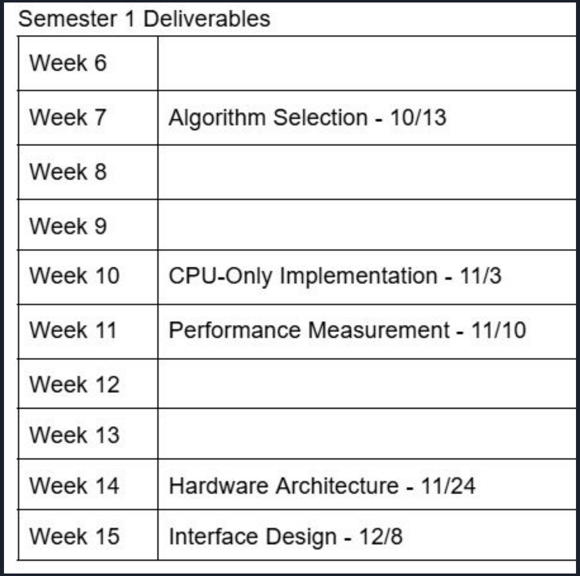


The platform we're using was loaned to us for this project by Quasonix Inc.

An UltraZed™-EG SOM on a custom carrier board. The SOM contains a Zynq UltraScale+ MPSoC device (4 core ARM64 CPU with on-chip peripherals, plus FPGA programmable logic).

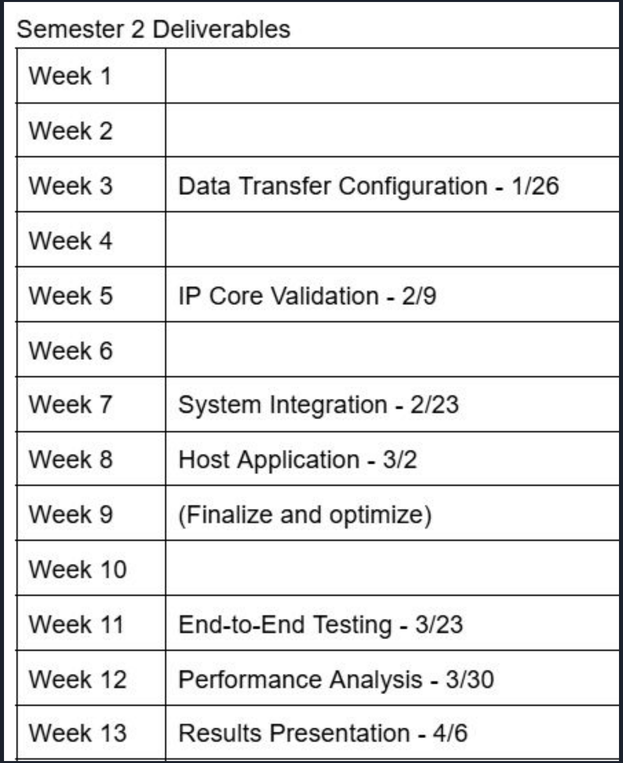
We're also using Vivado Design Suite to design the hardware, and Cocotb, a CO-simulation library that allows testing VHDL/Verilog hardware using Python.

Milestones



Semester 1 Deliverables

Week 6	
Week 7	Algorithm Selection - 10/13
Week 8	
Week 9	
Week 10	CPU-Only Implementation - 11/3
Week 11	Performance Measurement - 11/10
Week 12	
Week 13	
Week 14	Hardware Architecture - 11/24
Week 15	Interface Design - 12/8



Semester 2 Deliverables

Week 1	
Week 2	
Week 3	Data Transfer Configuration - 1/26
Week 4	
Week 5	IP Core Validation - 2/9
Week 6	
Week 7	System Integration - 2/23
Week 8	Host Application - 3/2
Week 9	(Finalize and optimize)
Week 10	
Week 11	End-to-End Testing - 3/23
Week 12	Performance Analysis - 3/30
Week 13	Results Presentation - 4/6

Results

Completed:
- Software baseline.
- Device setup, base hardware architecture design.
- AXI transfer interface.

In Progress:
- Implementing algorithm on FPGA.
- Conducting tests with comprehensive benchmarking for data comparison.

Remaining Tasks:
- Refine the host application to manage DMA transfers and minimize data-copying overhead.
- System validation by conducting tests on the integrated system to ensure stability.
- Data visualization and final presentation.

Challenges

– Data transfer overhead: Acceleration gains can be lost during the time it takes to move data across the AXI bus.

Solution: Changing the transfer interface from being through the CPU using interrupts to instead be through shared DDR memory using a bus master DMA within FPGA logic to minimize overhead and latency.