RISC-V Processor and Applications on FPGA

Oz Bejerano, Jonathan Colella, Birane I Toure, Alex Riveron, Jimmy Wen
oz.bejerano, jonathan.colella, birane.toure, ajr234, jimmy.z.wen@rutgers.edu
Advisors: Professor Phillip Southard, Manoj Viswambharan, Saurabh Singh, Gregory Mueller

RUTGERS
THE STATE UNIVERSITY OF NEW JERSEY

Goal

- ❑ Design a pipelined, RISC-V compliant, RV32I processor core
- ❑ Support arithmetic operations, memory accesses, and changing logic flow
- ❑ Implement a 5-stage, classic RISC pipelined architecture
- ❑ Include hazard detection and a stall insertion mechanism
- ❑ Build a terminal interface for output over VGA
- ❑ User input to interact with the system
- ❑ Develop applications that showcase processor capabilities
- ❑ Accelerate the advent of open source computing

Motivation and Objectives

- ❑ Motivation
  - RISC-V is a recently-introduced, modern, open-source Instruction Set Architecture (ISA)[1]
  - The RISC-V community welcomes Processor Core Implementations to help grow adoption of the ISA
- ❑ Objectives
  - Develop custom, open-source processor platform
  - Create a starting point for future related projects
  - Demonstrate the advantages of the RISC-V Platform:

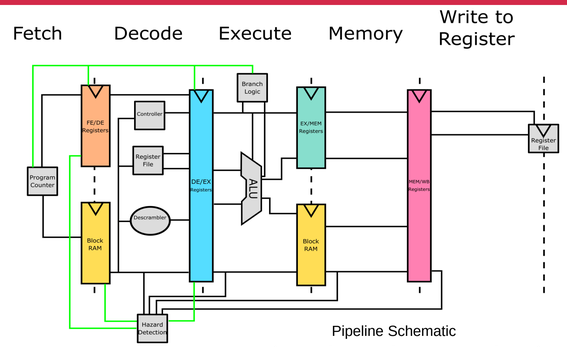
    High Efficiency | Small Size | Low Cost

Research Challenges

- ❑ Timing issues required multiple iterations of processor design and development
- ❑ Multiple references used to assist mapping out custom design of the processor[1][3]
- ❑ Some RISC-V Instruction Set extensions are in draft stage and still being developed

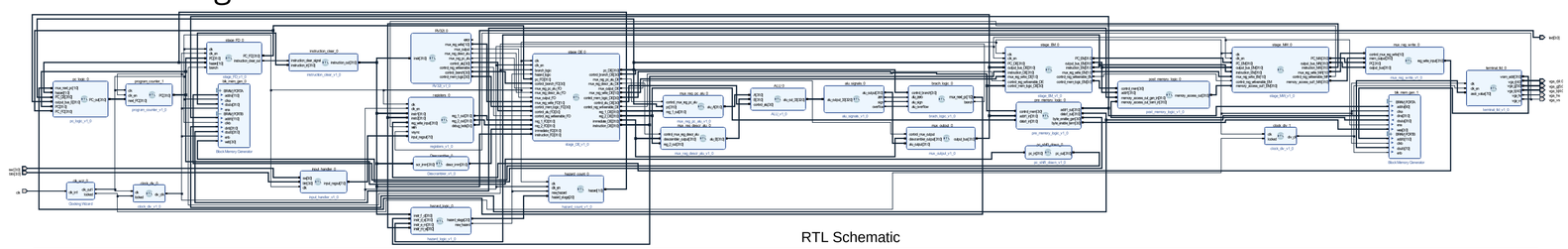
Methodology

- ❑ Translated the RISC-V ISA specification into a system of components
- ❑ Processor implementation consists of Python script generated and handwritten VHDL
- ❑ Controller component determines processor operation from the current program instruction and selects the components to use
- ❑ Controller is generated using a Python script given a table of instructions and corresponding control signals



Pipeline Schematic

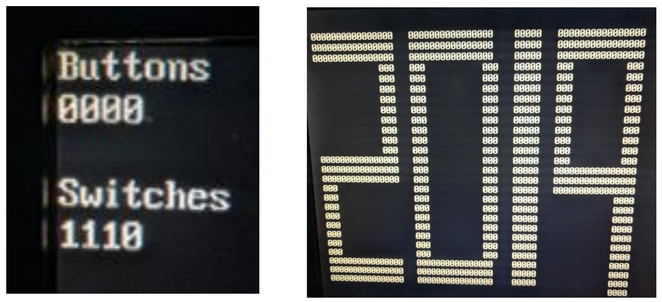
- ❑ Processor and supporting hardware designs are loaded onto a Xilinx Zynq-7000 FPGA[2] via Vivado synthesis and implementation



RTL Schematic

Results

- ❑ Demonstration programs are written in RISC-V assembly language
  - Output graphics and text to terminal
  - Games take input from user

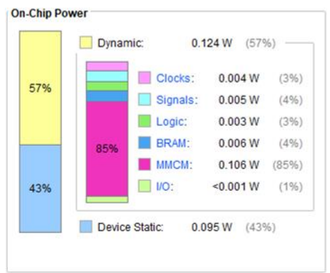


Input Test Program

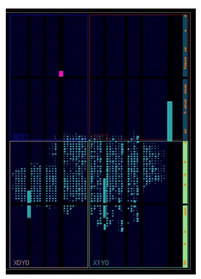
Terminal Graphics Demo

- ❑ Implemented an ASCII terminal over VGA[4]
- ❑ Switches and buttons form low-latency interfaces to the processor

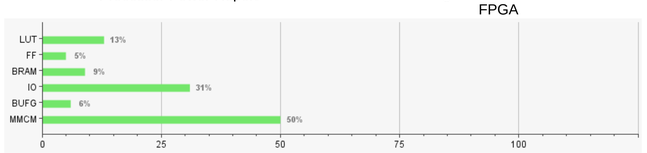
- ❑ Unit tests written in RISC-V assembly language used to verify processor features



Processor Power Report



Design Implementation on FPGA



FPGA Resource Utilization (Percentage)

- ❑ Processor runs at 25MHz and uses .22W of Power

Acknowledgement

We thank Harris Corporation for sponsoring the project and for entry into the 2019 Senior Mentor Program.

References

[1] https://riscv.org/specifications/
[2] https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html
[3] https://riscv.org/risc-v-books/
[4] https://github.com/tibor-electronics/vga_generator/tree/master/vga_text