Rogues Gallery - Reconfigurable Computing

Aadi Kapur, Yihan Jiang, Roye Eshed, Varun Valada (Spring 2021)

Project Goals:

The Reconfigurable Computing team enhances pipelined-CPU runtimes by building upon Open Source frameworks and implementing innovative hardware schemas. Our goal is to think of hardware in a non-conventional fashion and challenge the industry standard and implementing those schemas together to create innovative hardware microarchitectures.

Project Overview and Methods:

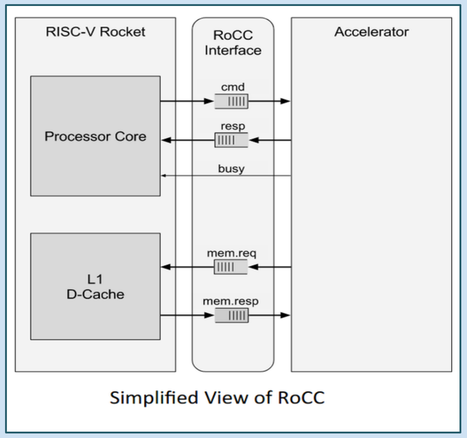
➔*AES: Post-quantum safe*
- ◆ AES-128 (per instructional encryption).
- ◆ Data sent to specified encryption register.
- ◆ Decryption instruction reverses changes.

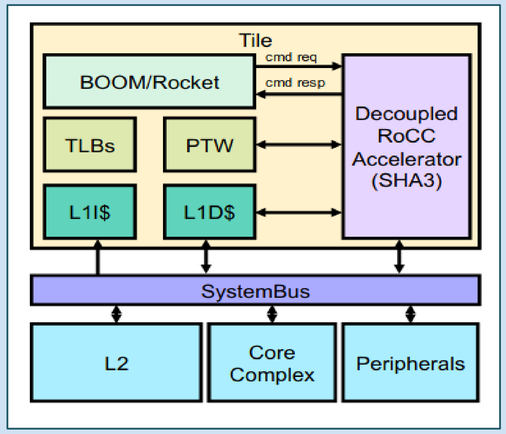
➔*Hardware I/O and Interrupt Support:*
- ◆ Act as an interface for BOOM to incorporate other devices by having an interrupt controller that stall the dispatch queue until the interrupt completes
- ◆ Implement Interrupt Support for I/O + Traps that allows for native hardware interrupts rather than merely software ones.

➔*Chipyard:*
- ◆ Finalized the VM configuration to compile BOOM and Chipyard configurations.
- ◆ Purpose was to eventually map the Chisel modules to BOOM and memory.
- ◆ A RoCC interface helps the processor communicate with our custom Chisel modules.

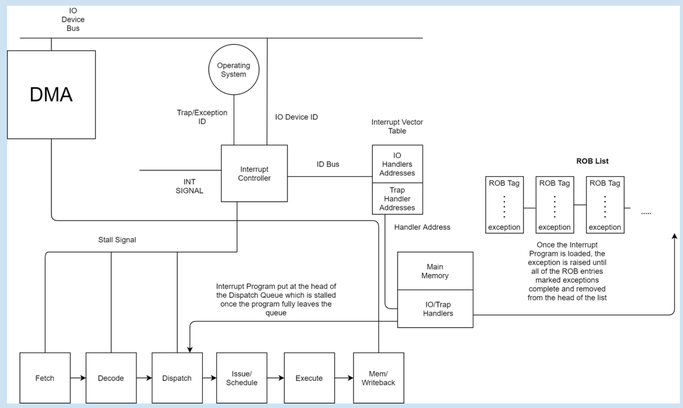
Design Methodology

Rocket Chip Coprocessor (RoCC) Interface:



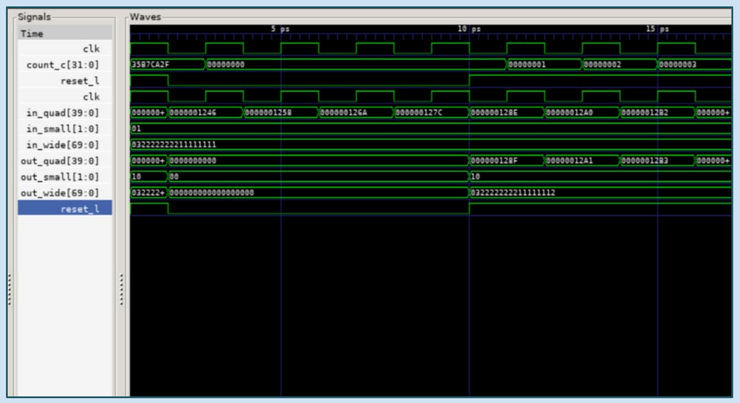


Above Left: Interface between a custom accelerator, the Rocket Chip processor and memory. Above Right: RoCC interface from the perspective of a top-level design file for a similar SHA3 encryption module.



Left: Showing the Interrupt Controller receiving an interrupt request, putting it into the head of the dispatch queue and then updating the rob entries with an exception flag raised until the interrupt program completes and then sending the date through DMA if needed.

Right:The waveform for the default sample chipyard.



Lessons Learned

➔ *Chisel*: Scala-embedded HDL
➔ *ISAs*: Risc-V and custom instructions
➔ *Encryption*: AES-128
➔ *RoCC Interface* (BOOM-Core integrations)
➔ *Hardware I/O and Interrupt Support: Challenges in Superscalar Pipelines*
➔ *Assembly*: new instr BOOM-integration
➔ *Chipyard Custom Accelerator Configuration*
➔ *Coding Practices* (GitHub, Code Review)
➔ Environment Setup

Next Steps and Future Challenges

➔ Further test completed modules, and look for edge case bugs.

➔ Readapt the work done in Interrupt Support for basic pipelines this semester into pipeline support for Superscalar Processors

➔ Ensure that Encryption and Decryption work on a wide range of values.

➔ Finish integrating completed designs with the RISC-V BOOM architecture, benchmark performance changes.

➔ Create a way for the BOOM processor to interface with our integrated AES/SRBP accelerator.