Rogues Gallery - Reconfigurable Computing

Tarun Maddali, Kyle Suter, Aadi Kapur, Yihan Jiang, Roye Eshed (Fall 2020)

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.

Project Overview and Methods:

➔*SRBP (Smart Register Branch Predictor):*
- ◆ Serves as register file and branch predictor.
- ◆ Sorts registers per register-write, in an insertion-sort fashion.
- ◆ Tradeoffs: 100% accuracy, but higher energy consumption and write cycle time extended.

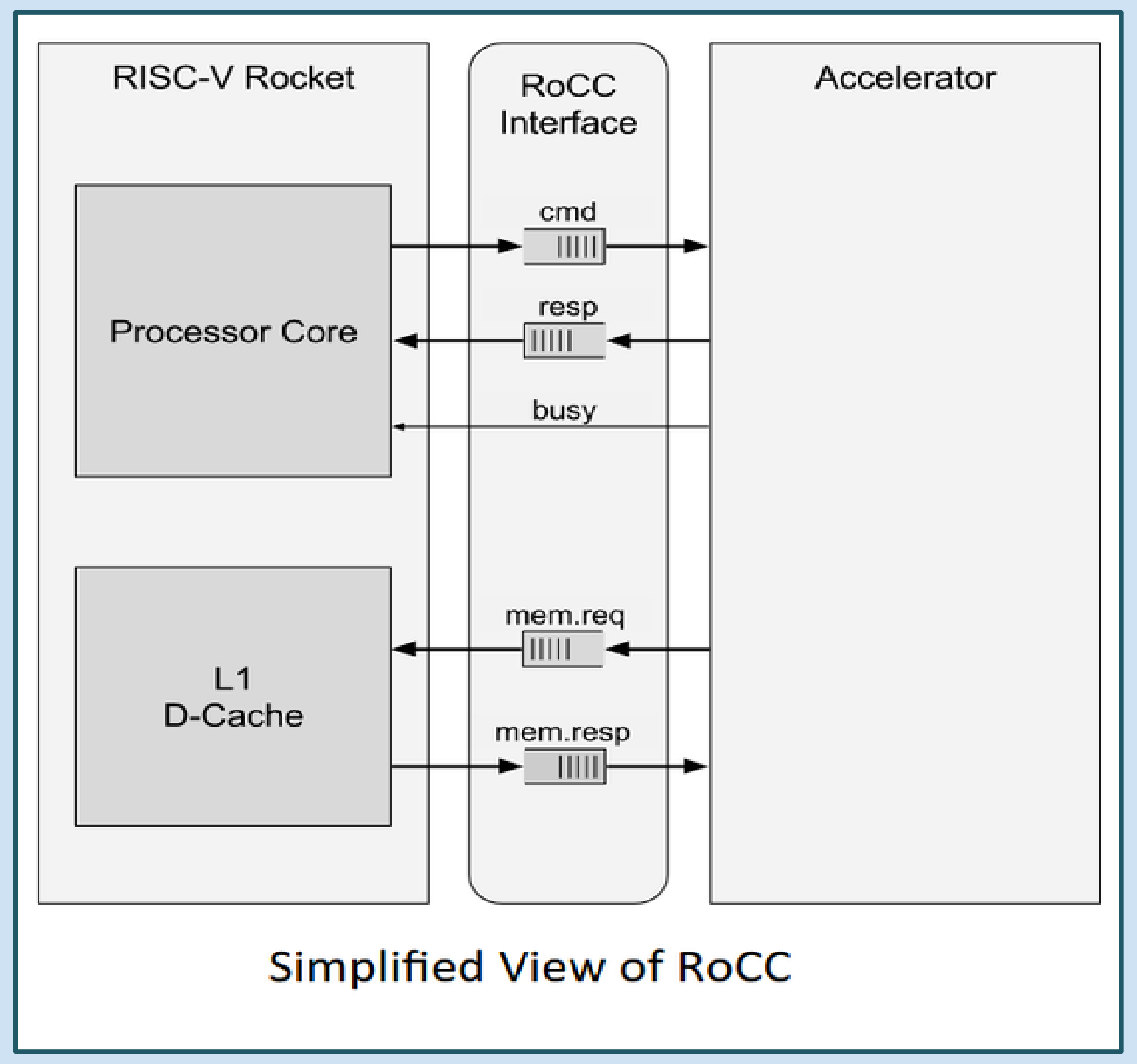
➔*Encryption: Post-quantum safe*
- ◆ AES-128 (per instructional encryption).
- ◆ Data sent to specified encryption register.
- ◆ Decryption instruction reverses changes.

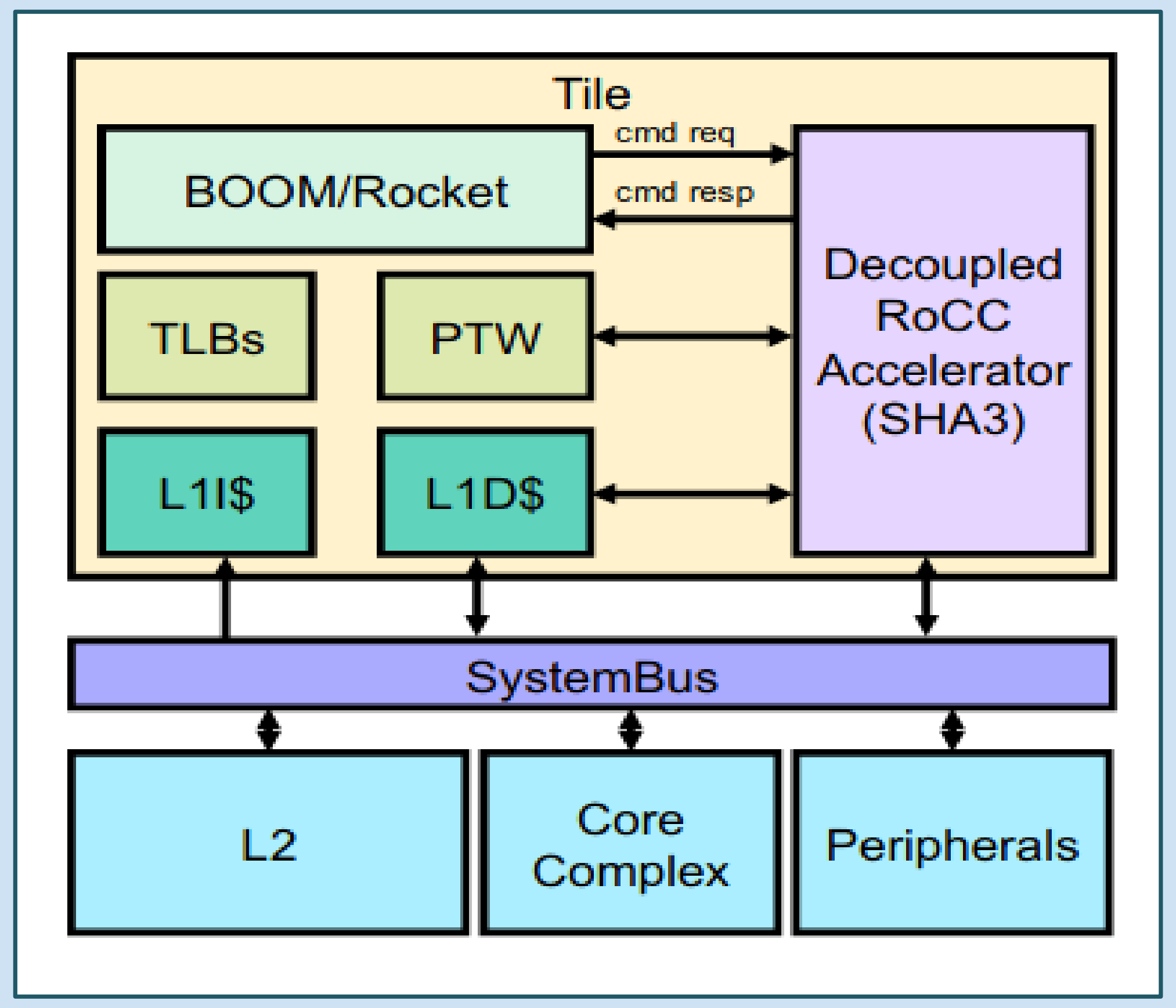
➔*Hardware I/O and Interrupt Support:*
- ◆ Act as an interface for BOOM to incorporate other devices.
- ◆ Implement Interrupt Support for I/O + Traps.

➔*RoCC Accelerator Integration:*
- ◆ Began development on a assembly instruction mapping.
- ◆ Purpose was to eventually map the Chisel modules to BOOM and memory using RoCC.

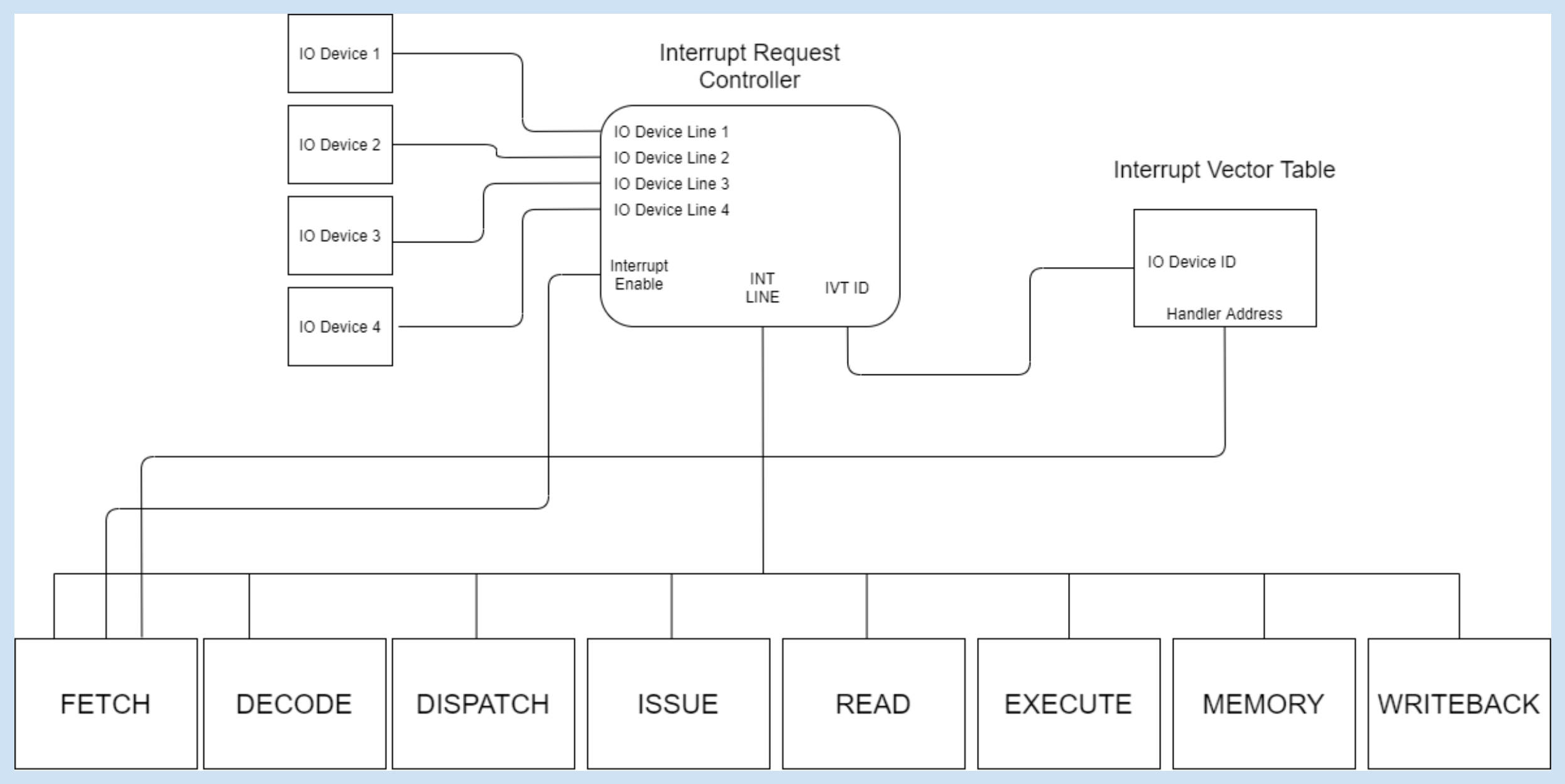
Design Methodology

Rocket Chip Coprocessor (RoCC) Interface:



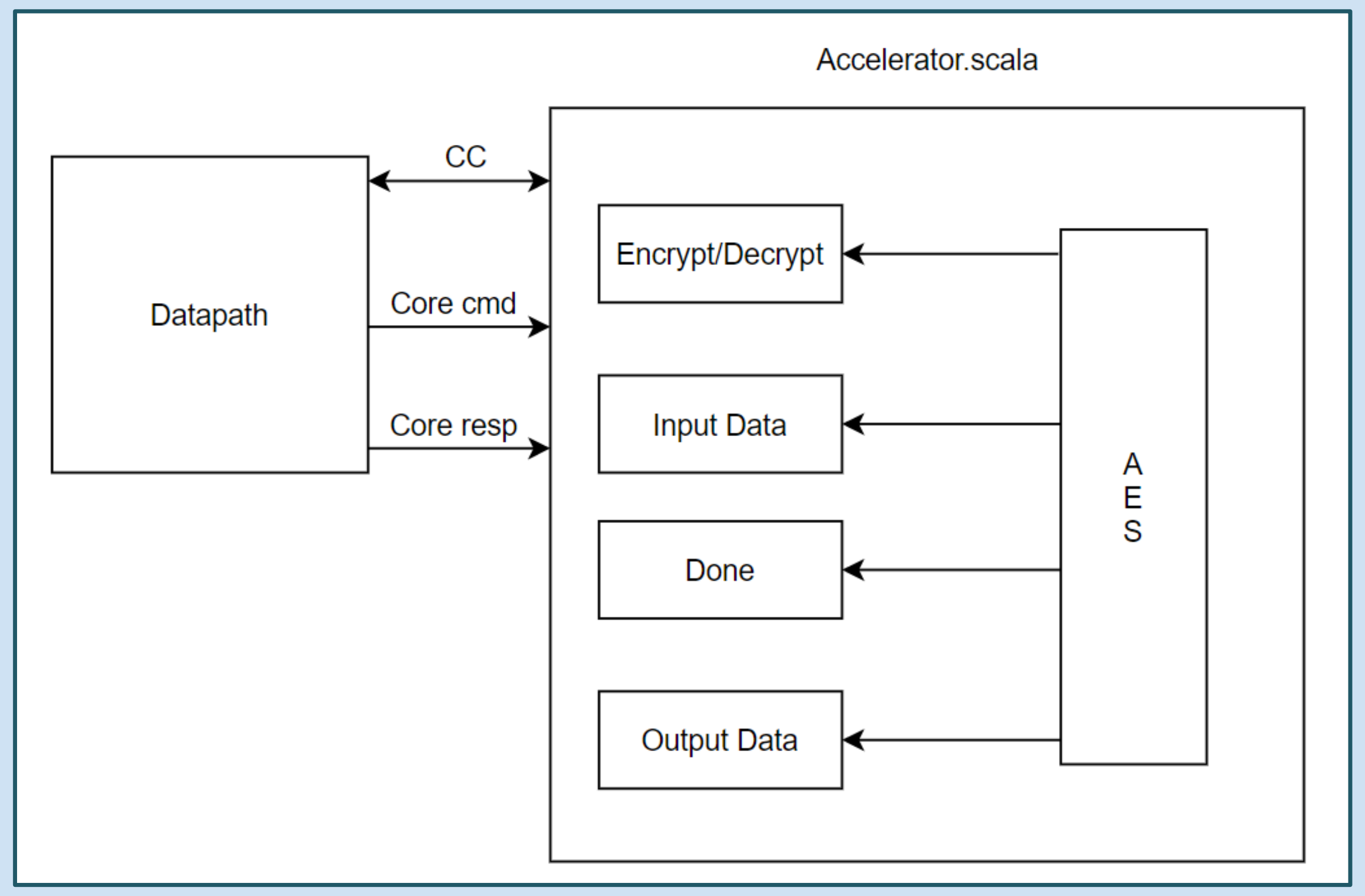


Above Left: Interface between a custom accelerator, the Rocket Chip processor and memory.
Above Right: RoCC interface from the perspective of a top-level design file.



Above: Display of the use of the Interrupt Request Controller to handle Interrupts from IO Devices and execute the Interrupt Handler for devices..

Right: A block diagram that shows the integration of the AES and the accelerator by the Rocc interface.



Lessons Learned

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

Next Steps and Future Challenges

- ➔ Further test completed modules, and look for edge case bugs.
- ➔ Implement a Branch Target Buffer, so that the Branch *Decision* and Branch *Target* are both known in 1-cycle.
- ➔ Ensure that Encryption and Decryption work on a wide range of values.
- ➔ Integrate completed designs with the RISC-V BOOM architecture, benchmark performance changes.
- ➔ Create new assembly instructions that account for our modules within the BOOM processor.