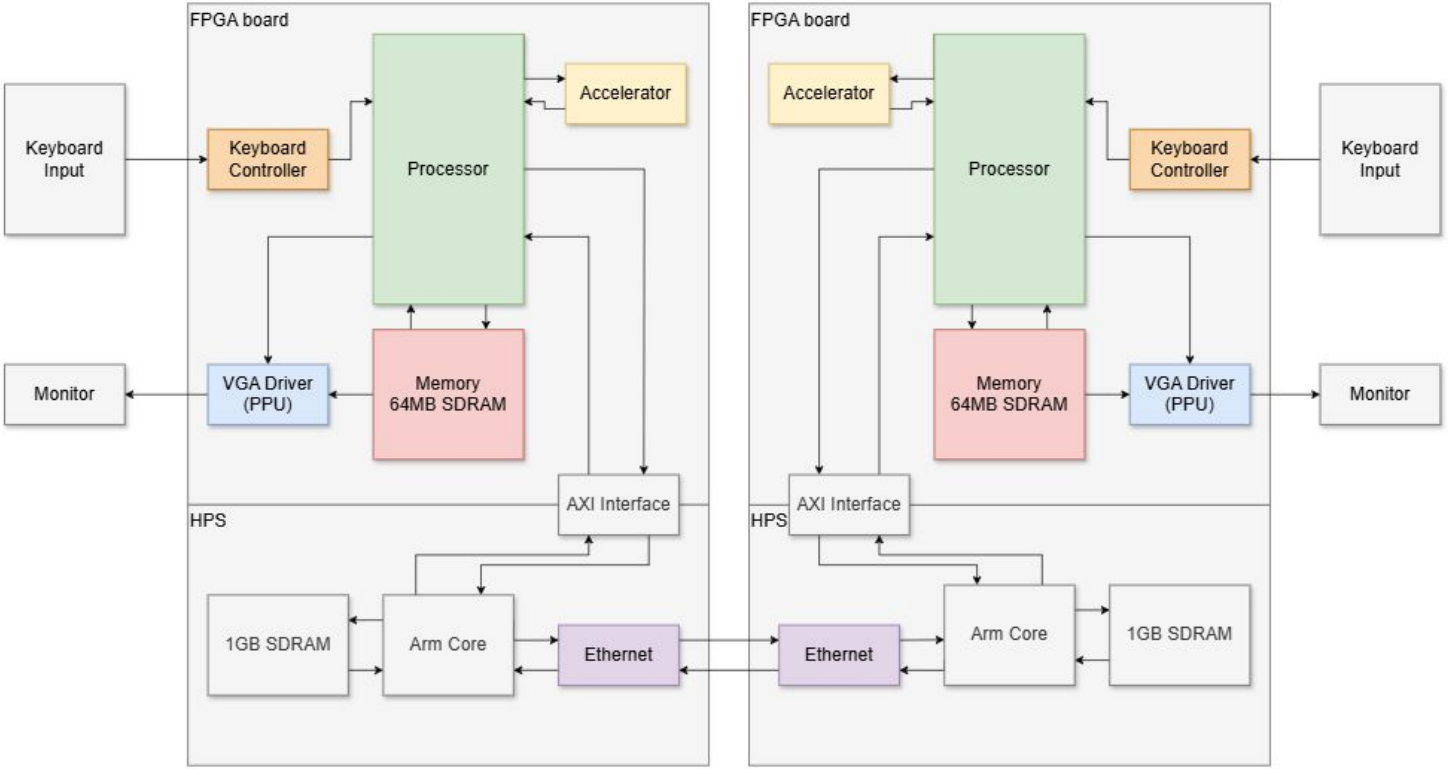
Battleship Architectural Review

By: Jaime Campos, Josh Cobian, Jacob Edmundson, Alan Ekstrand, Zach Simons

High Level Block Diagram



Processor Implementation

Custom ISA

- RV32I (Excluding FENCE, FENCE.I, ECALL, EBREAK, CSRRW, CSRRS, CSRRC, CSRRWI, CSRRSI, and CSRRCI)
- Update Grid Square (ugs xy, sel, e, m, v, type, seg, ai): Sends grid data to PPU for rendering.
- Return from Interrupt (rti): To return control flow back to PC after processing input interrupt.
- Reset from Interrupt (rsi): To return control flow to specific instruction after processing board communication interrupt.
- Load Random (lrd rd): To load a random number from Linear Feedback Shift Register.
- Start Accelerator (sac): Tells accelerator to start operation.

Algorithm to Accelerate

C. Liam Browns Battleship Probability Calculator

- Randomized Monte Carlo algorithm
- Identifies the next best move by creating a heatmap of probabilities
- Given a longer runtime or faster calculations it will provide a more accurate result
- Accelerating it will provide a smarter AI opponent

Accelerator Implementation

- Communication with processor
  - Accelerator gets game state from memory
  - Processor writes game state to memory
  - Paired with custom RISC-V processor
- Workflow
  - Processor sends instruction to start the accelerator
  - Accelerator gets game state from memory
  - Accelerator runs until turn swap interrupt
    - On swap, finish algorithm
    - Write best move to memory
    - Set done flag for processor to know the prediction is ready

Picture Processing Unit

- Stores the current state of each grid square in the game.
  - Gets game state updates from the processor
- Based on each squares state, determine the corresponding sprite to display.
  - Ship type, ship segment, hit/miss
- Drives VGA output to draw game to screen.
  - Ships, board, hit/miss markers, grid selection, best move marker

External Interfaces

- A keyboard connected through PS/2 to interact with the game
  - wasd & arrow keys to move
  - space & enter keys will fire at selected square
  - stretch goal: use mouse instead
- Ethernet to communicate between boards
  - only accessible through HPS
  - will interact with HPS and FPGA through AXI Interface
- VGA output to display game
  - store sprite data in FPGA SDRAM
  - PPU will help retrieve sprites and send to display

Verification of Original Plan

- Current Architecture aligns with goals of proposal
  - RISC-V processor
  - extended ISA to interact with PPU & interrupts
  - 2 board communication through Ethernet
  - Engaging & interactive demo through keyboard input
- Differences from initial proposal
  - Changed communication protocol from UART to Ethernet
    - lower latency & higher bandwidth
  - Accelerator
    - Battleship Probability Calculator
    - increase complexity of design and provide useful information to player