Weekly Updates

3/12 - 3/18

Josh: I mainly worked on the memory stage for the processor this week. This includes determining what type of memory to use, developing the wrapper for the memory controller and creating the LFSR register. I also worked on testing the memory wrapper and also the memory pipe stage which have both passed my test benches.

Alan: This week I worked on the Execute stage of the processor. This includes creating and testing the ALU and Branch Control Unit, along with connecting all of the control signals. In addition, I began writing the Battleship C code that will run on our processor. Initially I am writing a console version of the game to test the game logic before changing the input/output code to use the custom instructions provided by our custom processor.

Zach: This week I worked on implementing the decode stage for our processor. This included creating the register file including rf bypassing, the instruction decoder to set all of the processors command signals, as well as all the other small bits within the decode stage such as the muxing and sign extending as well as the pipeline into execute. I also wrote a testbench for the register file and am working on a testbench for the instruction decoder. I also began working on the accelerator module.

Jaime: I had worked on the PPU with the goal of getting a VGA controller to work on the board. I was able to find one online made by a professor at Cornell. With this VGA controller I had tested to see if I could load the game board on the top left of the screen. By converting the png image of the game board into a 24 bit rgb hex file, I was able to have the synthesis tool create an SRAM module that had my hex values preloaded into it and got it to work. However this design took 9min to synthesize with 70% of block ram used.

Jacob: I worked on the board-to-board communication modules (ethernet) and experimented with soc configurations in Quartus. I was able to use the Platform Design tool to generate a top level HDL file allowing our FPGA logic to communicate with the pins of the PHY chip on the board. Most of the RTL to create a full ethernet frame is complete, which comprises most of the MAC logic.

3/19 - 4/1

(Spring Break)

Josh: I created and tested fetch during this week. This included creating an i-mem wrapper and then writing the fetch module. The fetch module was more complicated than initially anticipated, mainly due to the 1-cycle delay associated with the memory IP we plan on using. This 1-cycle delay means that anytime we branch or handle an interrupt, we need to stall the pipeline for 1 cycle. I was able to test the fetch module during this time and verify functionality, making it ready for integration.

Alan: I finalized the command line version of the game and tested it for errors. Then I started adapting the code to remove cmd line input/output in preparation for compilation into RISC-V assembly where our custom instructions can be added. I also began working on the RISC-V toolchain in order to compile the C code into RISC-V.

Zach: I finished the accelerator module, which included all of the ship/gamestate updating logic as well as all of the comparison logic. This logic goes through each ship location and makes sure its segments do not intersect with any other ship segment or any game square that is not empty. Currently, this is done through nested for loops that roll out the logic into a fairly large implementation. I synthesized the design to check the area and am going to start work on a different variant to try and reduce logic utilization. On top of the accelerator implementation I wrote a testbench for it that generates up to a predefined value of random game squares to be marked as invalid, generates 5 random ship location/orientations that are valid (fit on the board) and then runs the accelerator on them to see if the game state is valid or not.

Jaime: I had not made much progress on the PPU or keyboard controller due to my lack of access to a decent computer, IO devices, and an FPGA. Most of my work relied on physically interacting with the IO devices so I was limited on what I could do. However, I had looked into finding smaller sprites for the game due to the current ones being too large and taking up a significant amount of block ram.

Jacob: No progress on the ethernet this week- was on a cruise for break and unable to connect to the CAE machines

4/2 - 4/8

Zach: This week, I implemented forwarding and hazards on the processor as well as helped Josh with stalling. After this, I spent most of the week testing the processor along with Jacob and Josh. We went through every basic instruction together and validated that they worked, which came with a lot of bug fixing. This resulted in all basic instructions working, except for an error with loads that is being worked on by Josh. After the basic instructions were validated, I split off to work on validating the custom instructions which is what I am still working on now. Currently,

most of our basic custom instructions that send information out of the processor are validated to work without forwarding but need forwarding to be properly integrated, which I almost have done. Our interrupts had a lot of issues, so I have fixed most of them and am close to having them functioning.

Josh: I worked entirely on the processor this week. What I did was build the main proc.sv module and connected all of our pipeline stages together. Once that was done, I helped Zach with hazards and stalling, and then Zach, Jacob, and I worked all weekend on testing the basic instructions to see if they would work on our processor. We tested all the operational instructions, jumps and branches, and loads and stores. As of yesterday, all the basic instructions are working except for loads, which need further testing. We are really close to finishing the processor, and once that is done, we hope to start connecting our processor to our peripheral devices.

Alan: This week I finished setting up the RISC-V toolchain to compile C code into RV32I assembly. Then I wrote a custom assembler capable of assembling RV32I and our custom instructions into the machine code to run on our processor. Once that was done I started using our custom assembler to help test our processor.

Jaime: I made a lot of progress this week by completing an implementation of the PPU with the smaller sprites I found and the keyboard controller. I verified the functionality of the PPU by sending a sequence of inputs that would display all sprites onto the screen at every possible orientation. To test the keyboard controller I wrote a verilog testbench that simulated a PS/2 device and checked to see if my DUT signals were correct when certain keys were pressed. I then tested both the PPU and Keyboard together by having a demo that would move a selected square across the gridboard.

Jacob: This week I worked mostly on testing the processor. I created the testbench that will be used throughout the project with different test programs, but first just to test the execution of basic instructions. We started by testing all the R and I type instructions which went fairly quickly, and then moved on to testing loads and stores, which took longer. We created different test programs that can be easily ran with the testbench to test different aspects of the design and by the end of the week we got most basic functions working, besides some load instructions that seem to be edge cases because of hazards.

4/9 - 4/15

Alan: I spent this week writing the C code including the custom instructions for our processor that will actually run battleship. I had to adjust the code for the accelerator in order to reduce the amount of data memory required by optimizing the invalid_combos list. Finally, I made numerous adjustments to the custom assembler in order to work better with the RISC-V toolchain. The adjustments included adding support for global variables and handling specific syntax that the gcc compiler generates (even though it is not part of the general RISC-V standard).

Zach: This week I continued work on validating the custom instructions. I finished the basic ones early on, but interrupts had many issues due to our implementation of instruction memory and the fetch module. I spent most of the week fixing these issues and eventually got them to work correctly in any edge case. After this I wrote out some tests for interrupts to be added to the testing out auto testing suite.

Josh: The main thing I did this week was continue to build out the testing suite for the processor while also fixing bugs. Some of the tests I made were loading consecutive values that would cause a data hazard, more branching and jump tests, a final interrupt branching test, and many more processor tests to verify that forwarding worked properly. I also helped Jacob make an automated test bench that would test these programs without needing to view the waves. Hopefully, the last bugfix was today, and the processor should be completed.

Jaime: This week I attempted to create a different version of the PPU that had larger sprites as although the current version is functional, it would look better if we had larger sprites that covered more of the screen. I came across a good amount of issues mainly due to memory limitations on the FPGA. In order to use these sprites I would likely have to write the picture data into SDRAM, read it into a frame buffer and display it onto the screen.

Jacob: This week I continued developing the test infrastructure for the processor, mostly developing the testbench and making it easier to run with more test programs to improve our test coverage. I created a shell script that can be run from the command line that executes a list of our hex files to run with their respective tests in the testbench. I also added functionality to run just one test, and to have it open questa with waves from arguments in the script. This made it easy to run regression sweep of all tests whenever we change something in the processor to see if anything else stopped working as a result of our changes. It seems like we got all of the normal RV32i instructions to work this week. I also started doing high level integration to assemble all modules of our project in and be able to run it with quartus.

4/16 - 4/22

Josh: The main thing I did this week was continue to debug the processor and SPART modules. SPART was quite difficult to fix, as we had many errors that popped up in synthesis but didn't in simulation. We found that this was because the clocks between the FPGAs were not being synced, and fixed that problem. We also found some load and interrupt bugs that our processor testing didn't find, so we fixed those two.

Alan: I spent this week integrating the processor and the SPART modules and debugging the board to board communication. We had trouble with our original SPART module having a lot of noise in the signals, so I created a new SPART module to have less noise. Finally, I updated the C code to utilize the board to board communication to pass game state information between the boards.

Zach: This week I worked alongside the team to continue integrating each module one by one and test/debug in between these gradual changes. I also worked on debugging SPART which had a good handful of issues with it. We got it to a point where 95% of the features are integrated and working with only a small handful of bugs. Most of the remaining fixes are software-related, so we will continue to bug fix and finish up the last bit next week. This week I also worked alongside everyone to make the poster board for the final demo.

Jaime: I abandoned the second version of the PPU and focused on optimizing the older version. I found some background sprites of water and a shoreline and included them into a PPU such that the two boards are surrounded by water and there's a shoreline on the bottom of the screen. I also added labels A-J for each column on the board and numbers 1-10 for each row.

Jacob: I continued to work on integrating all parts of the system and test the board to board communication. I got it to work initially with the UART sending just 1 byte between boards, but we decided we wanted to send 3 bytes instead, which created a lot of issues. I initially created a wrapper to transmit 3 UART transactions serially, but it seemed very unreliable/buggy so we decided to just modify our protocol to send 24 bits in one transaction.

4/23 - 4/29

Alan: I spent this week debugging the final code and updating it as the PPU continued to adjust to convey more information to the player. I also worked on finalizing the poster so we could order it. Finally, I worked on integrating the accelerator with the processor, finalizing our design.

Zach: This week I continued work on the remaining debugging work but at this point the project is mostly finished. I also worked on getting our website updated and working so that it can be accessed during the demo via the QR code on the poster.

Josh: This week was a lot of debug work. I mainly worked with Alan, debugging the processor and the battleship code. I also worked with the team on creating the poster. I then ordered it and picked it up on Monday. I'm excited to present our project on Wednesday to the college!!!

Jaime: This week I added the title "Battleship" to the screen along with text that tells the player when it needs to wait or when it's their turn. I also implemented compatibility with the accelerator as the software can now send data to the PPU to highlight the output square green. I also updated the PPU to send a "Win" or "Lose" screen once the game finishes.

Jacob: This week was mostly helping with the PPU and displaying new sprites to the screen to make the game more visually appealing. I worked on creating new sprites for the game, including the game background, title, turn selector, and win/lose screens.