CS 3220 – Processor Design
Assignment #3: Verilog Pipeline
Spring 2019

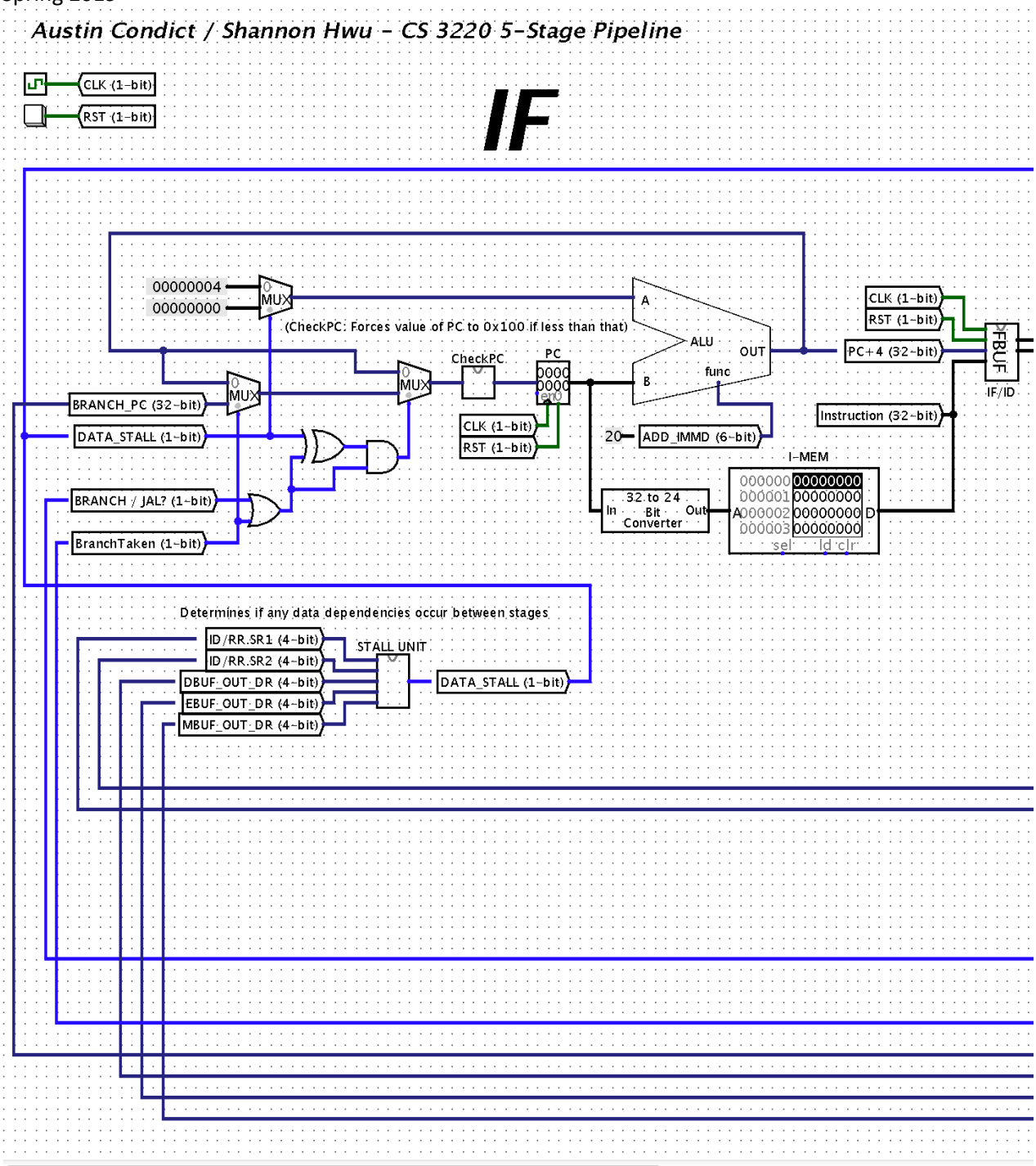
Austin Condict
[Partner: Shannon Hwu]

For this assignment, we were tasked with implemented a 5-stage pipelined processor in Verilog to run on our DE0-CV FPGA boards. We began with the provided assignment 3 frame and continued to build on that design based off of our diagrams from assignment 1. Throughout development, the HEX (7-segment) displays and LEDRs were used heavily for debugging. After implementing the overall pipeline, I wrote some small unit test programs to verify that instructions from the given ISA were handled properly and also to exercise our data dependency handling. In order to get Test.asm running as it should, Shannon and I reworked all stall logic from top to bottom and discovered a bug in our code where we had used logical negations (!) for all NOR/NAND/NXOR instructions rather than bitwise negations (~). After these revisions, we were able to run Test.asm with no problems.

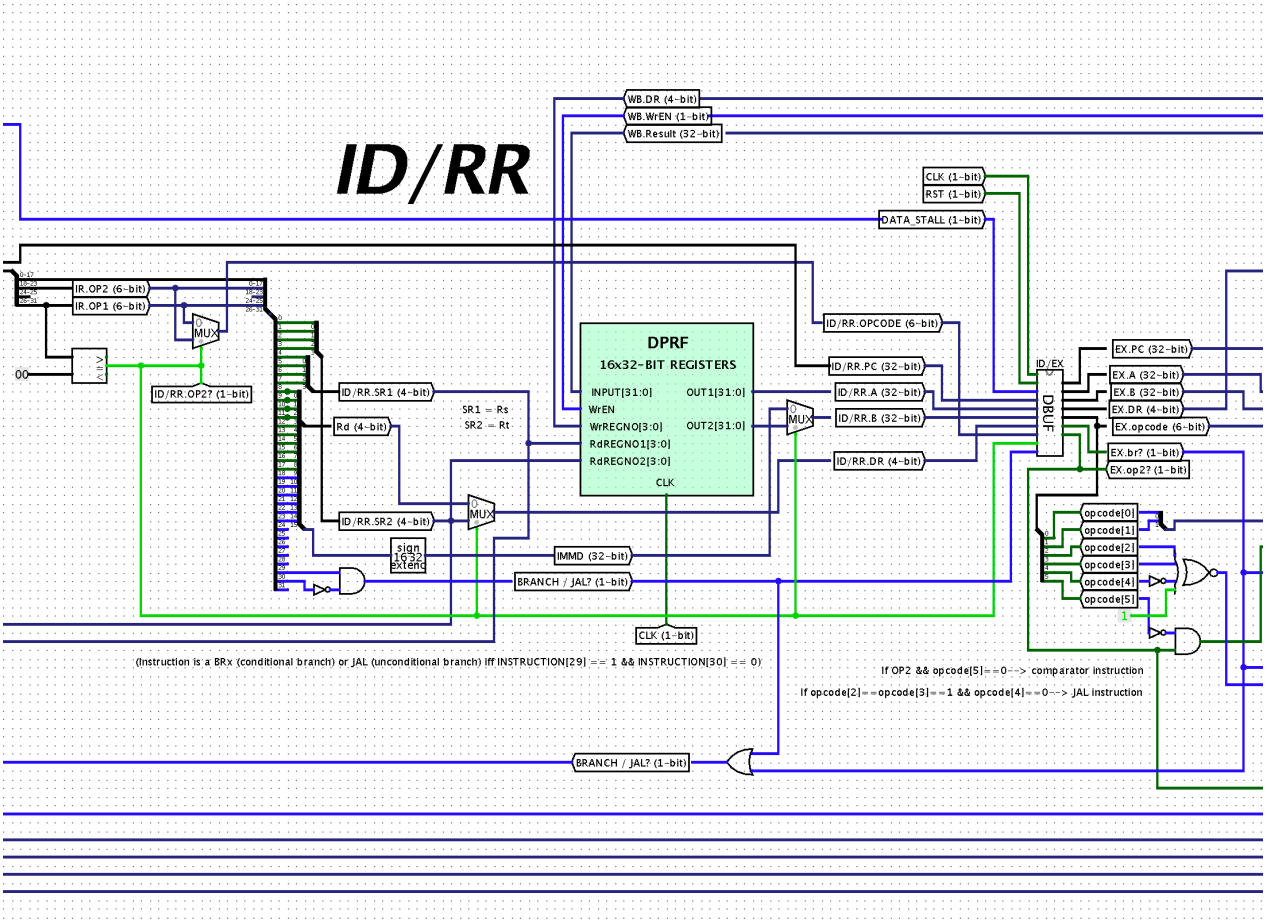
Confident in our solution, we were soon discouraged by the grueling debug sessions required for fmedian2.asm. The first issue was spotted with back-to-back CALL instructions in which only the second CALL's target address was being loaded into the PC. We fixed this by abstracting out the jump stall logic (separate from the four conditions we used for data stalls) into its own signal for detecting when any JAL or branch instruction is in the decode stage and used this for flushing the FE latch data. Our final bug after resolving the jump issues had to do with the FE latch PC value that gets passed to ID as well as back to itself for the address in IMEM from which to fetch the next instruction. A new register instance had to be created for saving the good FE PC for which to use for fetching while the other now simply forwards some PC value to the ID stage depending on two stall signals and a misprediction (i.e., "branch taken") signal. Hours of simulation waveform analysis in ModelSim helped us to decide when to use asynchronous wire signals vs. using their respective synchronous register values, based on operations occurring either one clock cycle too soon or one clock cycle too late.

Regarding contributions, I wrote all of the Verilog and assembly code and Shannon refined all stall and flush logic, and the both of us debugged cooperatively. Our overall design was very similar to our CircuitSim diagrams that were submitted in assignment 1 (shown below) other than our new stall logic.

Austin Condict
[Partner: Shannon Hwu]



Austin Condict
[Partner: Shannon Hwu]



Austin Condict
[Partner: Shannon Hwu]