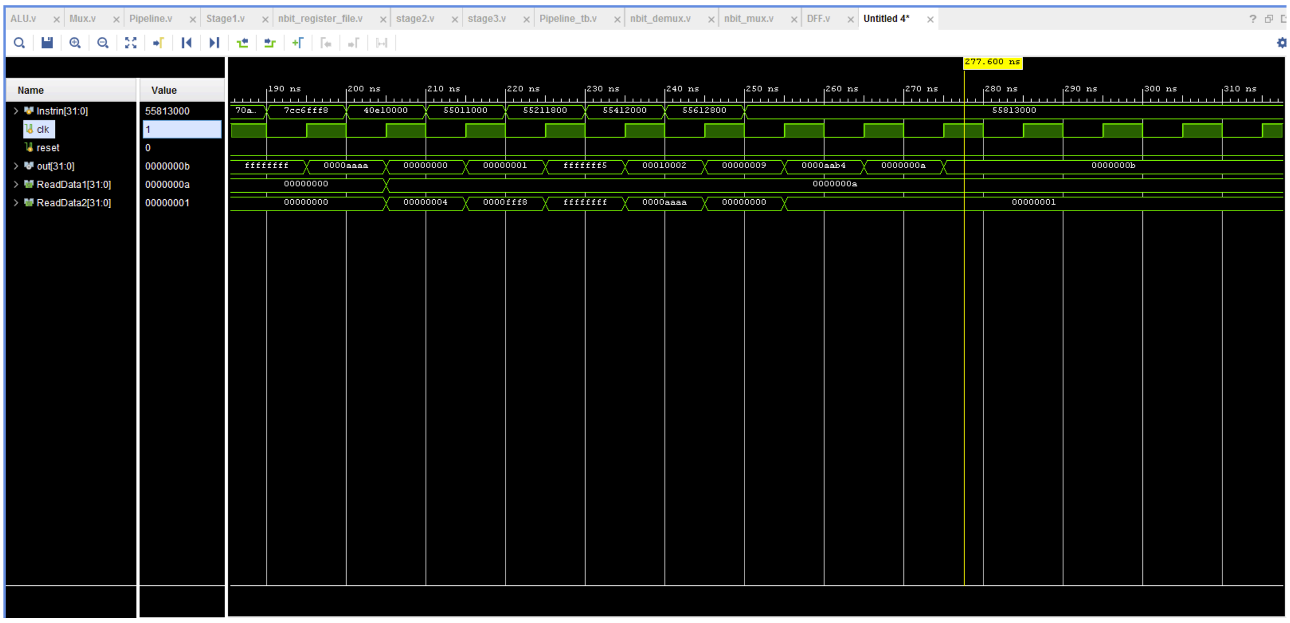
19/3/21
EC413
Ivan Isakov

Lab6

In this Verilog lab I learned how to create a pipeline data path. Inside my Top module called Pipeline, I had 6 total modules that created the datapaths. The modules were called, Stage1, nbit_register_file, Stage2, Mux, ALU, and Stage3. To create a datapath, the outputs for one module went into the inputs of another module.

In my module called stage 1, It had the inputs as a 32 bit InstrIn, a 1-bit clk, and a 1bit reset. My outputs were 32 bits S1_ReadSelect1 and S1_ReadSelect2, 16 bits Imm, 5 bit S1_writeselect, 3 bit ALUOP, 1 bit datasource, and 1bit S1_write enable. This module took in the input InstrIn and separated it into the appropriate variables that will be used as the input for stage 2. This all happened on the positive edge of the clock. Also this had a condition that checked to see if reset was true or not. If it was then the values reset to 0.

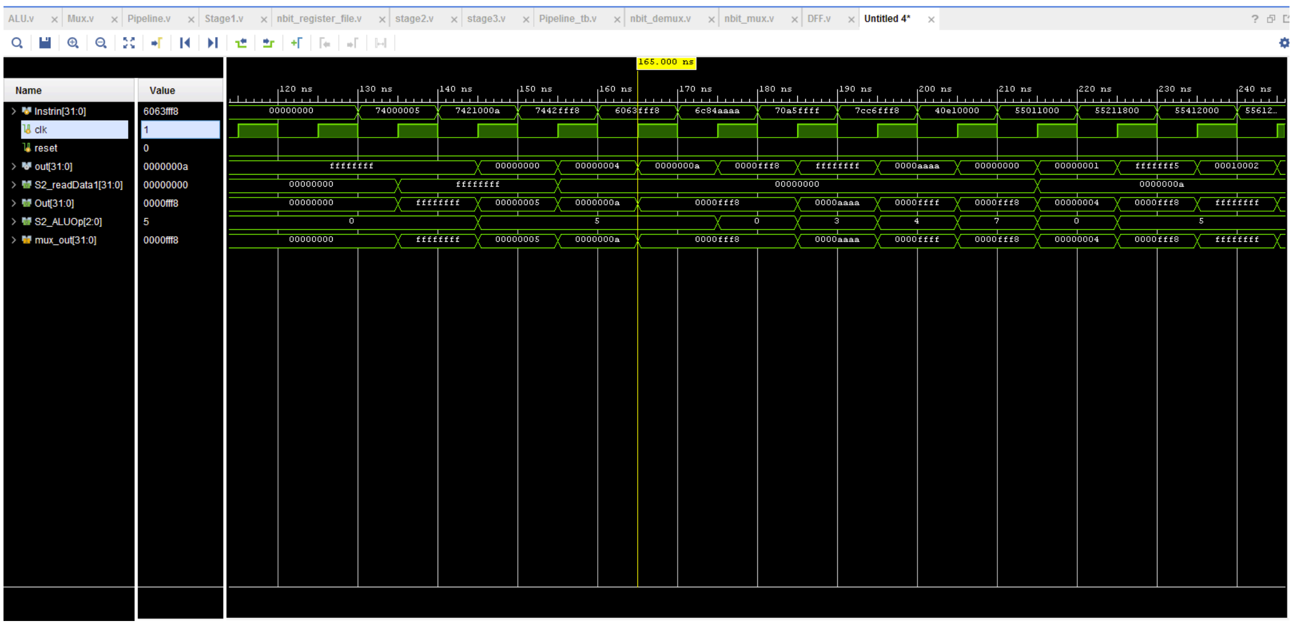
In the module called nbit_register_file, it had the inputs from both stage 1 and stage 3. This took in the inputs of Readselect1 and Readselect2 from stage 1, but took in Write data, Write Enable, and Write select from stage 3. With the given inputs this module gave Read data1 and Read data2 as outputs. I know that this module works correctly because I tested it with the values from the provided test bench and it matched up. This is shown in the picture below.



```
Name              Value
Instrin[31:0]     55813000   70a.. | 7cc6fff8 | 40e10000 | 55011000 | 55211800 | 55412000 | 55612800 | 55813000
clk               1
reset             0
out[31:0]         0000000b   ffffffff | 0000aaaa | 00000000 | 00000001 | fffffff5 | 00010002 | 00000009 | 0000aab4 | 0000000a | 0000000b
ReadData1[31:0]   0000000a   00000000 | 0000000a
ReadData2[31:0]   00000001   00000000 | 00000004 | 0000fff8 | ffffffff | 0000aaaa | 00000000 | 00000001
```

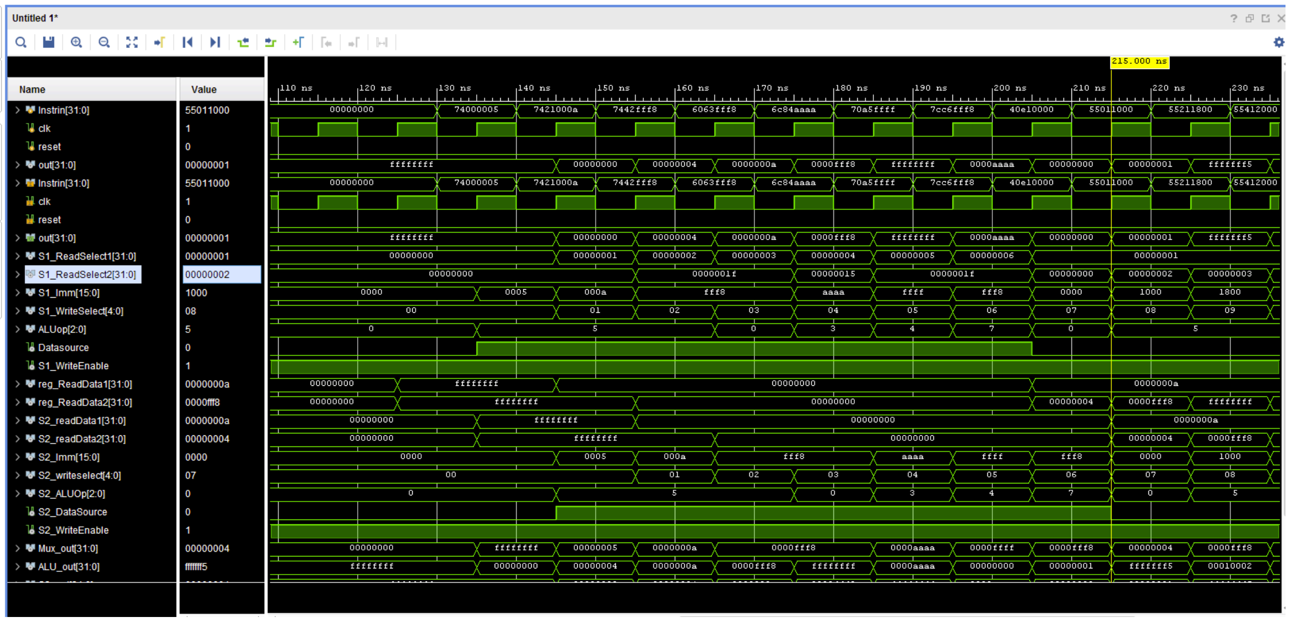
In the module called stage2, it took in the outputs from the register file and the outputs from stage1 as its inputs. The inputs Read data1 and read data 2 were from the register file, while the inputs IMM, data source, ALUOP, Write select, and Write enable are from Stage 1. Inside the module, if reset is not true, you set the outputs to stage 2 to equal the outputs of stage 1. Otherwise you set the values to 0.
In the module called MUX, it took the inputs of Immediate, ReadData2, and datasource from stage 2 and outputted the result into the ALU. In the module, it looked at the Datasource to determine whether or not it was an I type or a R type. Once it determines this, it will either output an immediate, but in 32 bits instead of 16, or the ReadData2 from stage 2.

In the ALU module, it took in the outputs of the mux, the ALUop and Read Data1 from stage 2. The module looks at the value of ALUop and determines which kind of operation to perform. I know the operation is correct because I checked the values the ALU outputs with the given test bench instructions. This is shown in the picture below.



The last module is called stage3. In this stage you take in the input from the ALU and two inputs from stage2, which are write select and write enable. If the reset in this stage is not true, the values from the stage 2 outputs are assigned to be the stage 3 outputs and ALUoutput becomes the output for stage 3 as well. Otherwise the values become zero.

The top module that incorporates all these modules is called Pipeline. It calls stage1, nbit_register_file, stage2, Mux, Alu, and stage3. This is where the outputs of each module are wired to the inputs of the other modules. I know this is wired correctly because the waveform diagram that was generated by the test bench matched the given values. This is shown in the picture below. Also the timing is correct since the values gets outputted three positive edge cycles after the IntrIn gets inputted.



Untitled 1*
215.000 ns
Name
Value
Instrin[31:0] 55011000
clk 1
reset 0
out[31:0] 00000001
Instrin[31:0] 55011000
clk 1
reset 0
out[31:0] 00000001
S1_ReadSelect1[31:0] 00000001
S1_ReadSelect2[31:0] 00000002
S1_Imm[15:0] 1000
S1_WriteSelect[4:0] 08
ALUOp[2:0] 5
Datasource 0
S1_WriteEnable 1
reg_ReadData1[31:0] 0000000a
reg_ReadData2[31:0] 0000fff8
S2_readData1[31:0] 0000000a
S2_readData2[31:0] 00000004
S2_Imm[15:0] 0000
S2_writeselect[4:0] 07
S2_ALUOp[2:0] 0
S2_DataSource 0
S2_WriteEnable 1
Mux_out[31:0] 00000004
ALU_out[31:0] ffffff5