EC 413

# Pipelined Datapath Lab 6 Report

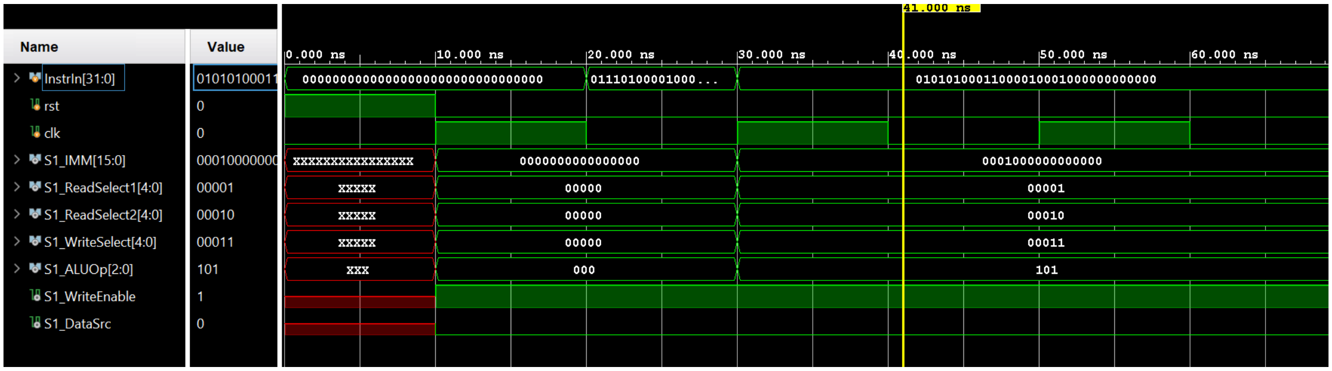
Ruben Carbajal
Claire Cropper

# Modules, Descriptions and Waveforms

## nbit_register_file:

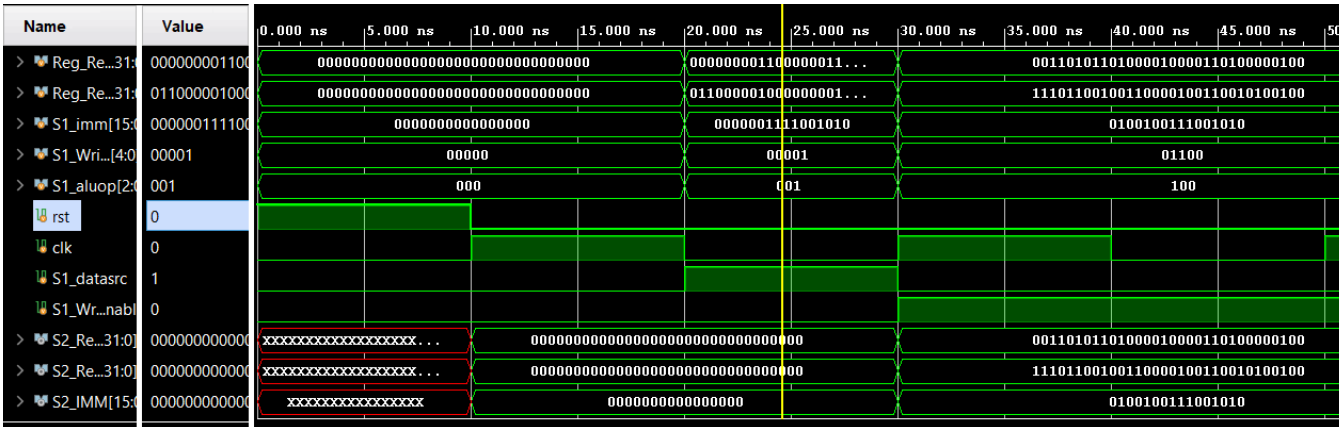
This module initializes the register values as 10 times the register, as given in Pre Lab 6.

## S1_Register:



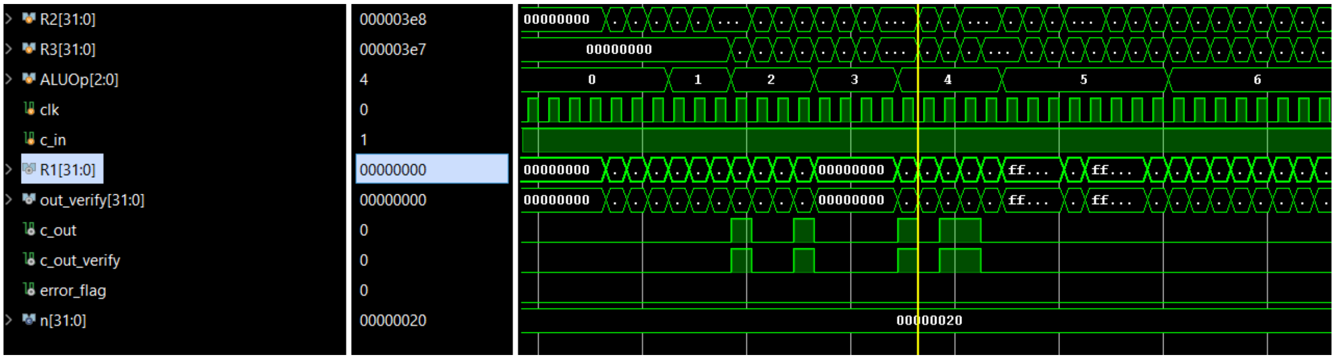
The S1_Register module takes in inputs clk, rst, and InstrIn, and then depending on what InstrIn receives, it writes the corresponding instructions to the proper outputs, S1_ReadSelect1 being bits [20:16], S1_ReadSelect2 being bits [15:11], S1_WriteSelect being bits [25:21], S1_WriteEnable set to [1’b1], S1_IMM being bits [15:0], S1_ALUOp being bits [28:26], S1_DataSrc being bit [29]. It assigns the correct part of each instruction to each output to ensure it operates as expected. We tested this module by assigning values to the inputs, resetting, and ensuring that the values were correct.

## S2_Register:



The S2_Register module takes in inputs clk, rst, Reg_ReadData1, Reg_ReadData2, S1_WriteSelect, S1_WriteEnable, S1_imm, S1_datasrc, S1_aluop. At every clock cycle, it writes Reg_ReadData1 to S2_ReadData1, Reg_ReadData2 to S2_ReadData2, S1_WriteSelect to S2_WriteSelect, S1_WriteEnable to S2_WriteEnable, S1_imm to S2_IMM, S1_datasrc to S2_DataSrc, and S1_aluop to S2_ALUOp. This module receives the output values from S1_Register and then writes them to the inputs of S2_Register. We tested this module by assigning values to the inputs, resetting, and ensuring that the values were correct.

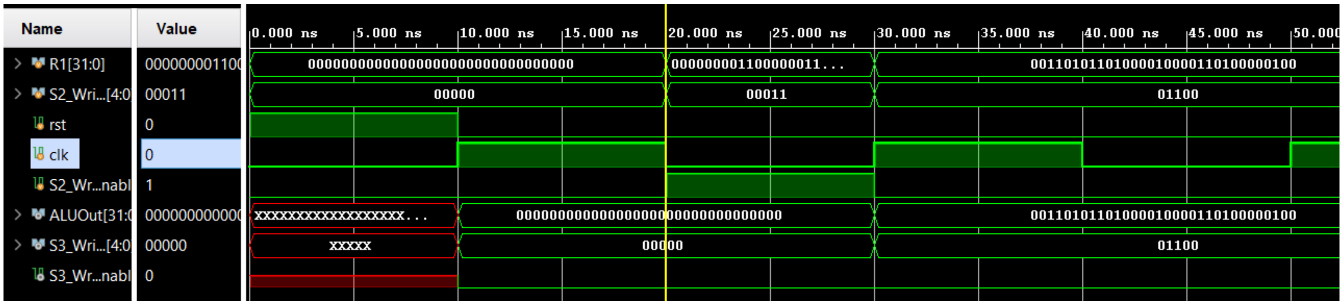
## ALU:



| | |
|---|---|
| R2[31:0] | 000003e8 |
| R3[31:0] | 000003e7 |
| ALUOp[2:0] | 4 |
| clk | 0 |
| c_in | 1 |
| R1[31:0] | 00000000 |
| out_verify[31:0] | 00000000 |
| c_out | 0 |
| c_out_verify | 0 |
| error_flag | 0 |
| n[31:0] | 00000020 |

This is the ALU we made during Lab 5 where we wrote multiple submodules to create it. Some of these functions include ADD, SUB, NOR, NAND, MOV, NOT, and SLT. We then instantiated all of these into the top ALU module, and connected all of their respective outputs to wires. Depending on the ALU select, we used test cases that will connect the desired output to the final output to the ALU.

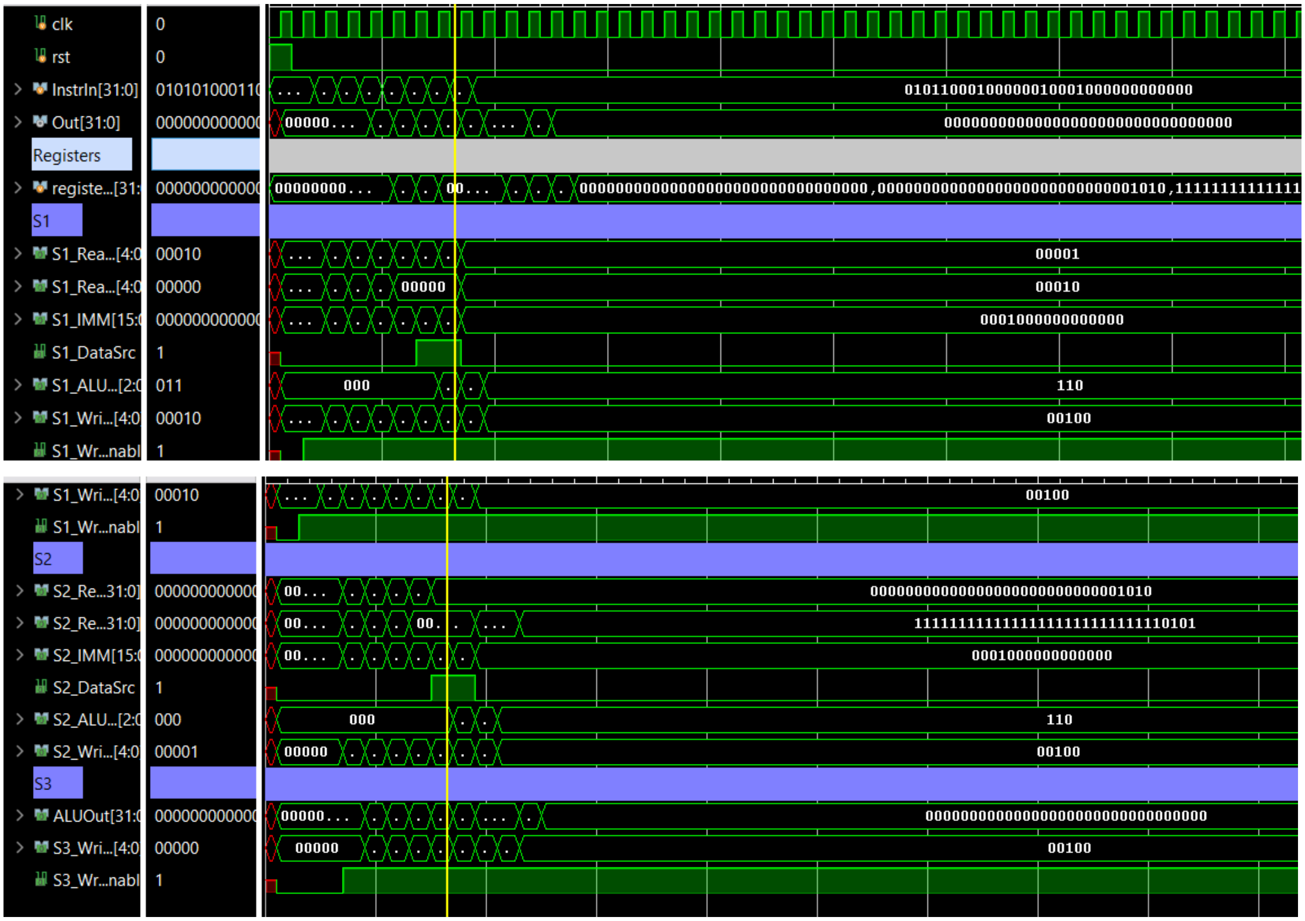
## S3_Register:



| Name | Value |
|---|---|
| R1[31:0] | 000000001100 |
| S2_Wri...[4:0] | 00011 |
| rst | 0 |
| clk | 0 |
| S2_Wr...nabl | 1 |
| ALUOut[31:0] | 000000000000 |
| S3_Wri...[4:0] | 00000 |
| S3_Wr...nabl | 0 |

In Stage 3 of the Pipeline, it holds the purpose of passing the correct output based on ALU output that it is given. For our S3_Register module, it receives and passes S2_WriteSelect and S2_WriteEnable from S2_Register to determine if the output is written to. This can be seen when the S2_WriteEnable goes to 1, the given ALU output is passed to the overall output.

## Pipeline:



```
clk             0
rst             0
InstrIn[31:0]   010101000110
Out[31:0]       000000000000
Registers
registe...[31:  000000000000
S1
S1_Rea...[4:0   00010
S1_Rea...[4:0   00000
S1_IMM[15:0     000000000000
S1_DataSrc      1
S1_ALU...[2:0   011
S1_Wri...[4:0   00010
S1_Wr...nabl    1
```

```
S1_Wri...[4:0   00010
S1_Wr...nabl    1
S2
S2_Re...31:0]   000000000000
S2_Re...31:0]   000000000000
S2_IMM[15:0     000000000000
S2_DataSrc      1
S2_ALU...[2:0   000
S2_Wri...[4:0   00001
S3
ALUOut[31:0     000000000000
S3_Wri...[4:0   00000
S3_Wr...nabl    1
```

In the Pipeline module, we instantiate S1_Register, S2_Register, S3_Register, nbit_register_file, and ALU to create the pipelined datapath. In order to test this, we modified the given Pipeline_tb.v file from the Pre Lab 6 to include an ALU and S3_Register instantiation.