CSM152A – Lab 3

Names: Jahan Kuruvilla Cherian; Kevin Xu

Group: Lab 6 – Group 9

UID: 104436427; 704303603

Date Performed: 4/28

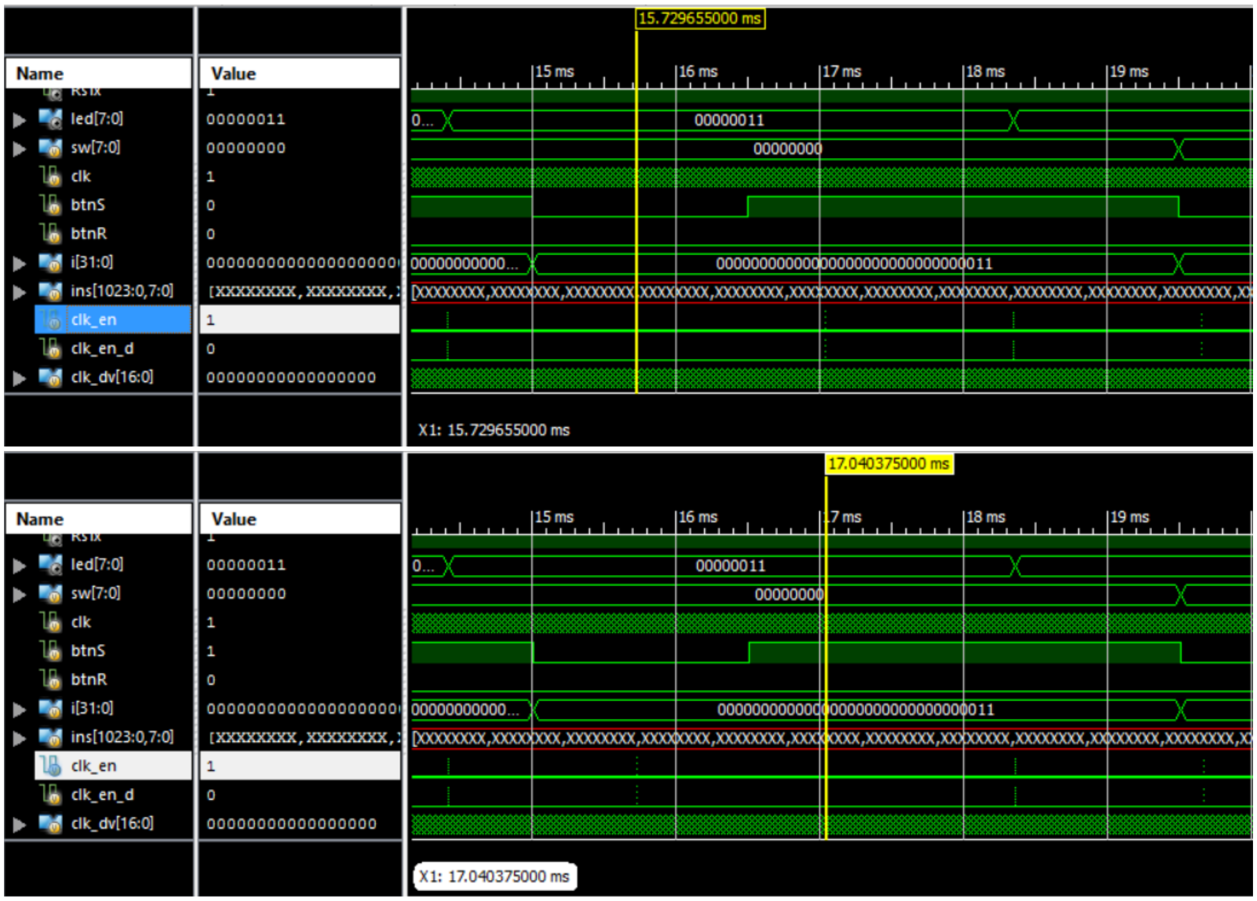
TA: Babak Moatamed

Workshop 1

Clock Enable

1.) To understand what clk_en really does we looked at the clock enable section of the nexys3.v file. We notice that at every positive edge of the master clock cycle, an 18-bit register clk_dv_inc is incremented by 1, acting almost like a counter. When the reset is enabled then all our values are reset to 0 including clk_en. However, in all other situations we notice that clk_dv is set to the lower 17 bits of the incremented value clk_dv_inc, clk_en is set to the MSB of the incremented value and clk_en_d is set to clk_en. What all of this amounts to is that this section of the code is essentially a clock divider used for the enable signal which is used in the next section of the code for instruction stepping.

2.) Here is the waveform that shows clk_en and clk_en_d.



Name	Value
led[7:0]	00000011
sw[7:0]	00000000
clk	1
btnS	0
btnR	0
i[31:0]	0000000000000000000
ins[1023:0,7:0]	[XXXXXXXX,XXXXXXXX,
clk_en	1
clk_en_d	0
clk_dv[16:0]	00000000000000000

X1: 15.729655000 ms

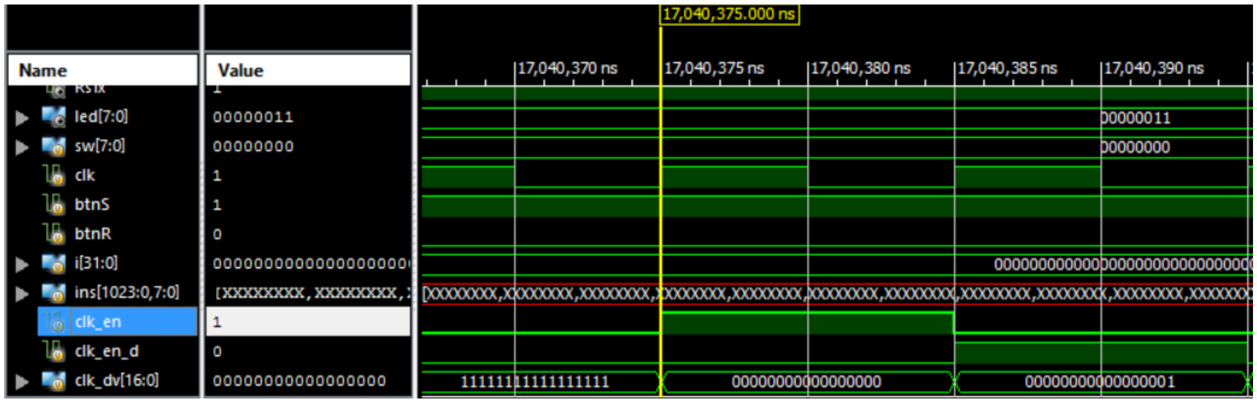
Name	Value
led[7:0]	00000011
sw[7:0]	00000000
clk	1
btnS	1
btnR	0
i[31:0]	0000000000000000000
ins[1023:0,7:0]	[XXXXXXXX,XXXXXXXX,
clk_en	1
clk_en_d	0
clk_dv[16:0]	00000000000000000

X1: 17.040375000 ms

The period is 17.030375ms – 15.729655ms = 1.31072ms.

The frequency is 763Hz.

3.) As shown in the following waveform, clk_dv is 17’b0 when clk_en is 1 because the counter for the clock divider is reset when clk_en is high.



Name	Value
RstX	1
led[7:0]	00000011
sw[7:0]	00000000
clk	1
btnS	1
btnR	0
i[31:0]	0000000000000000000
ins[1023:0,7:0]	[XXXXXXXX, XXXXXXXX,
clk_en	1
clk_en_d	0
clk_dv[16:0]	00000000000000000

4.) Using Logisim we translated the clock enable section of the code into a schematic as shown below in Figure 1.

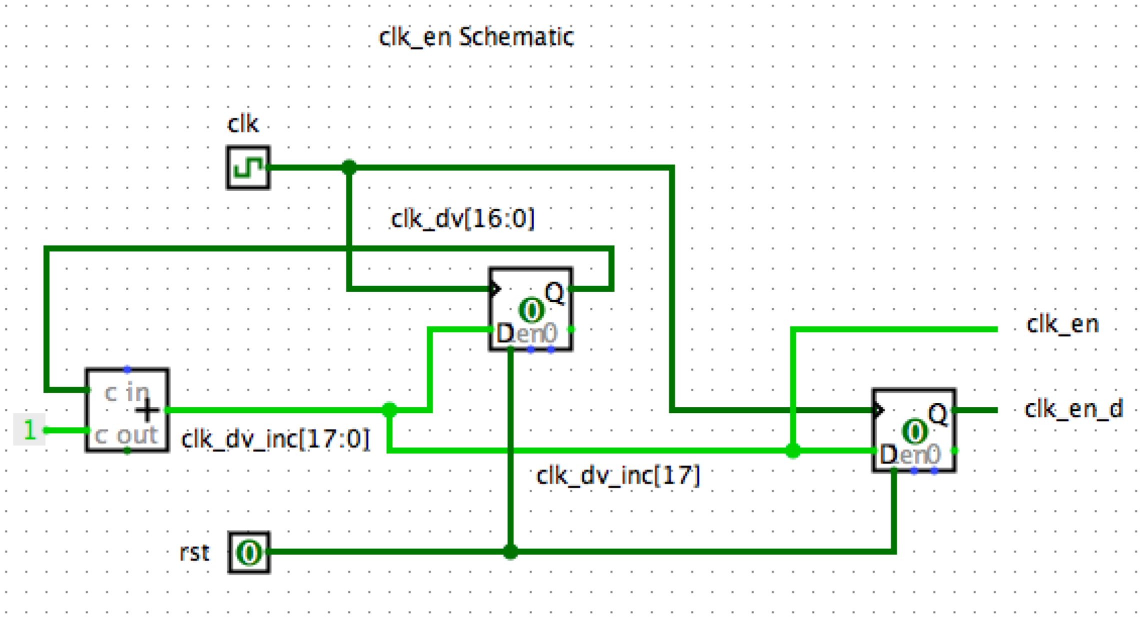
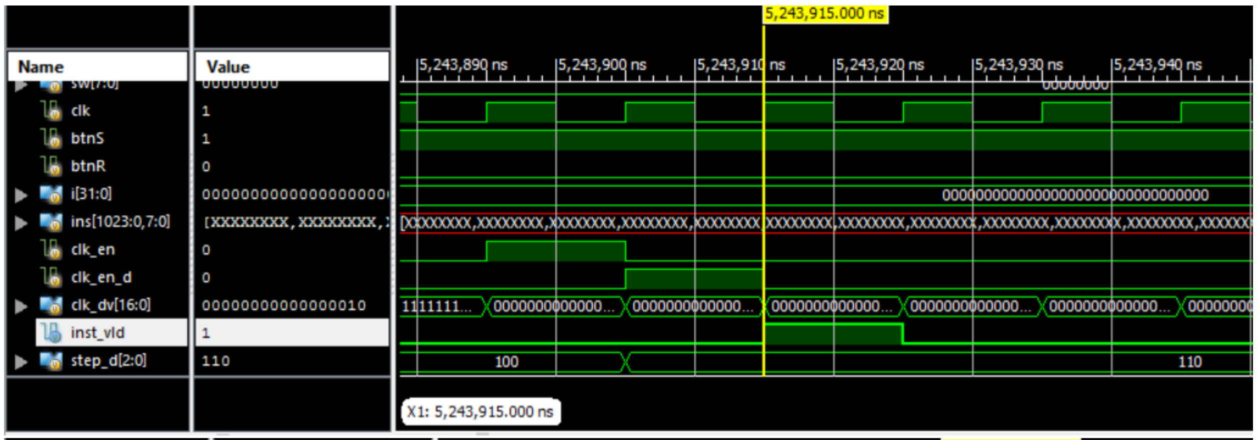


Figure 1: Logisim schematic showing how the clk_en, clk_en_d, clk_dv and clk_dv_inc signals are set from the Verilog code in nexsys3.v

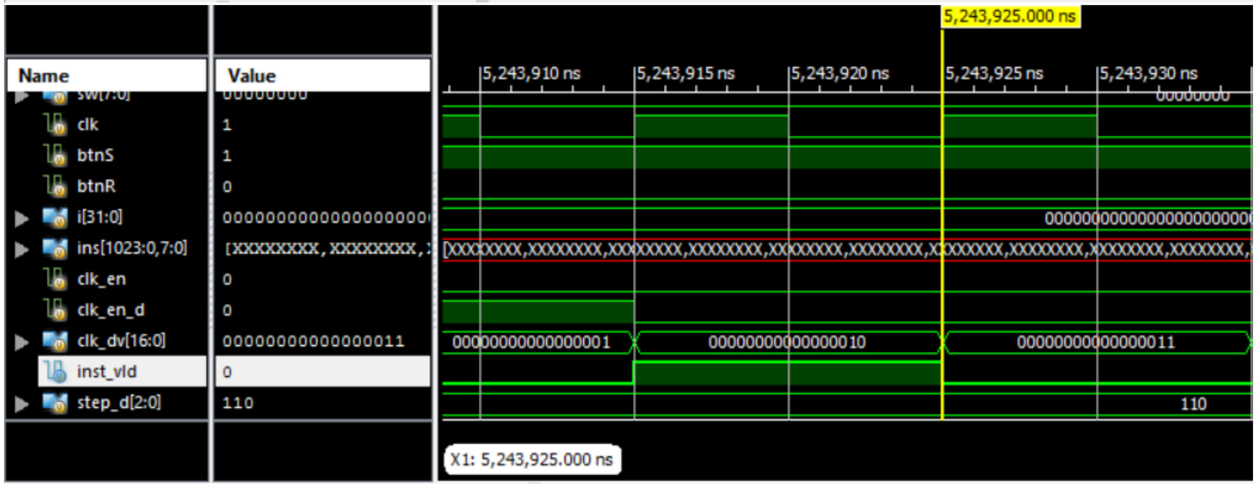
Instruction Valid

1.) We ran a simulation of the code from the nexsys3.v file and used the results to answer the questions presented in the lab manual.

2.) The first time interval during which inst_vld = ~step_d[0] & step_d[1] & clk_en_d evaluates to 1 is **10ns** following 5,243,915ns.



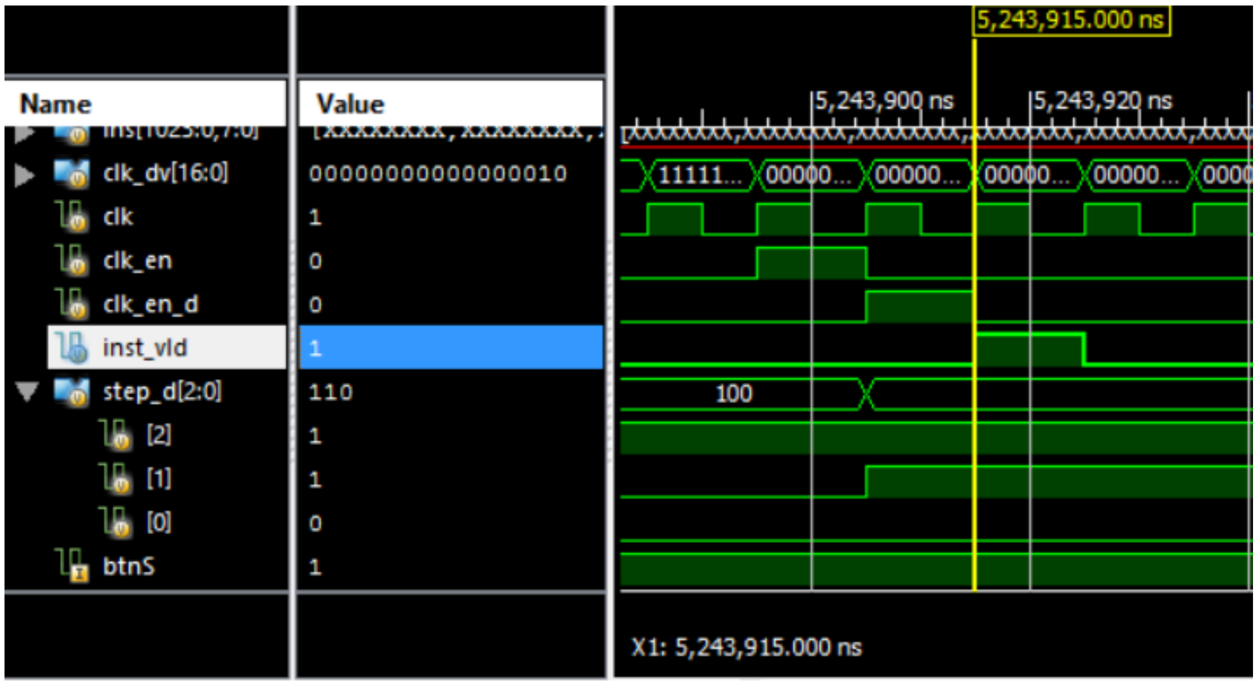
Name	Value
sw[7:0]	00000000
clk	1
btnS	1
btnR	0
i[31:0]	0000000000000000000
ins[1023:0,7:0]	[XXXXXXXX,XXXXXXXX,:
clk_en	0
clk_en_d	0
clk_dv[16:0]	00000000000000010
inst_vld	1
step_d[2:0]	110



Name	Value
sw[7:0]	00000000
clk	1
btnS	1
btnR	0
i[31:0]	0000000000000000000
ins[1023:0,7:0]	[XXXXXXXX,XXXXXXXX,:
clk_en	0
clk_en_d	0
clk_dv[16:0]	00000000000000011
inst_vld	0
step_d[2:0]	110

3.) clk_en_d is essentially a delayed version of clk_en. The reason this is used in setting the new value of step_d (which is then later used in the inst_vld signal) is because the clk_en value is set back to 0 once step_d changes. Thus using this would result in inst_vld always evaluating to 0. This delayed clk_en_d signal thus gives us the ability to evaluate the value of inst_vld at the same time the clk signal changes.

4.) The following is a simulation diagram showing the timing relationship among clk_en, step_d[1], step_d[0], btnS, clk_en_d, and inst_vld.



5.) We used Logisim to draw a schematic for the relationship between clk_en, step_d[1], step_d[0], btnS, clk_en_d, and inst_vld. This can be seen below in Figure 2.

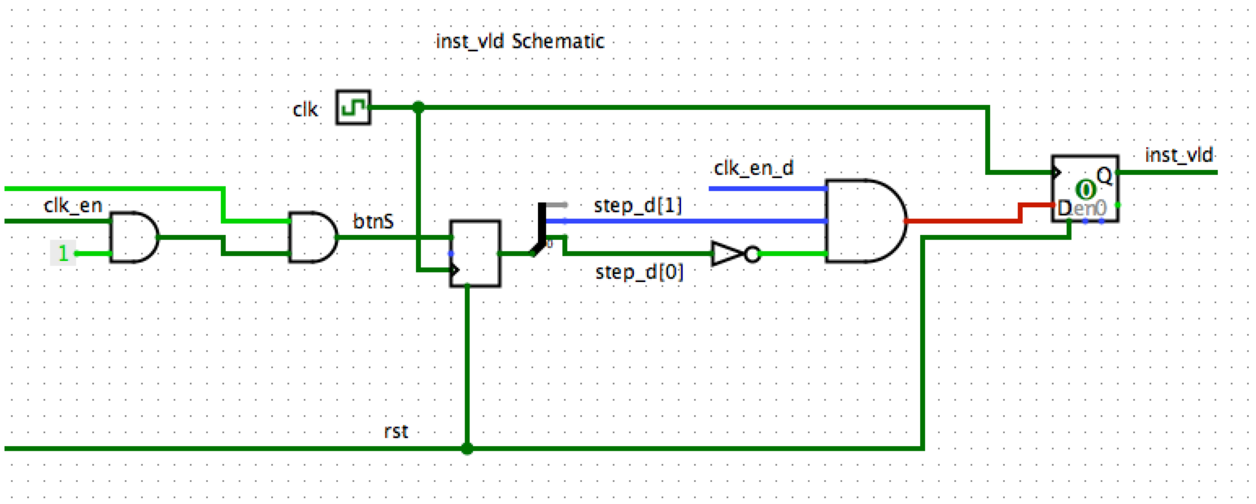


Figure 2: The Logisim schematic for the aforementioned signals. Note that this is not a complete implementation that is worthy of simulation because it focuses only on the small subpart of the overall design. It should be used merely for an overview of what this subsection of the nexsys3.v module does.

Register File

1.) Running the simulation for the program with the seq_rf.v module gave us insight into answering the questions presented in Workshop 1.

2.) The register is first assigned a non zero value on line 33 of the seq_rf.v Verilog code. The line is rf[i_wsel] <= i_wdata;. This line is clearly indicative that this statement is sequential in nature due to the use of a non-blocking assignment and the fact that it exists in an always block that is dependent on the positive edge of the clock.

3.) We notice on lines 35 ad 36 of the file to be the points where data is read out from the register based on two select bits – i_sel_a and i_sel_b. This is combinatorial logic since the statements as follows use blocking assignments without relying on a clock:
assign o_data_a = rf[i_sel_a];
assign o_data_b = rf[i_sel_b];

If we were to manually implement this we would have to use multiplexers with the two selection bits as our select bits.

4.) Figure 3 shows us the Logisim schematic of the register file block.

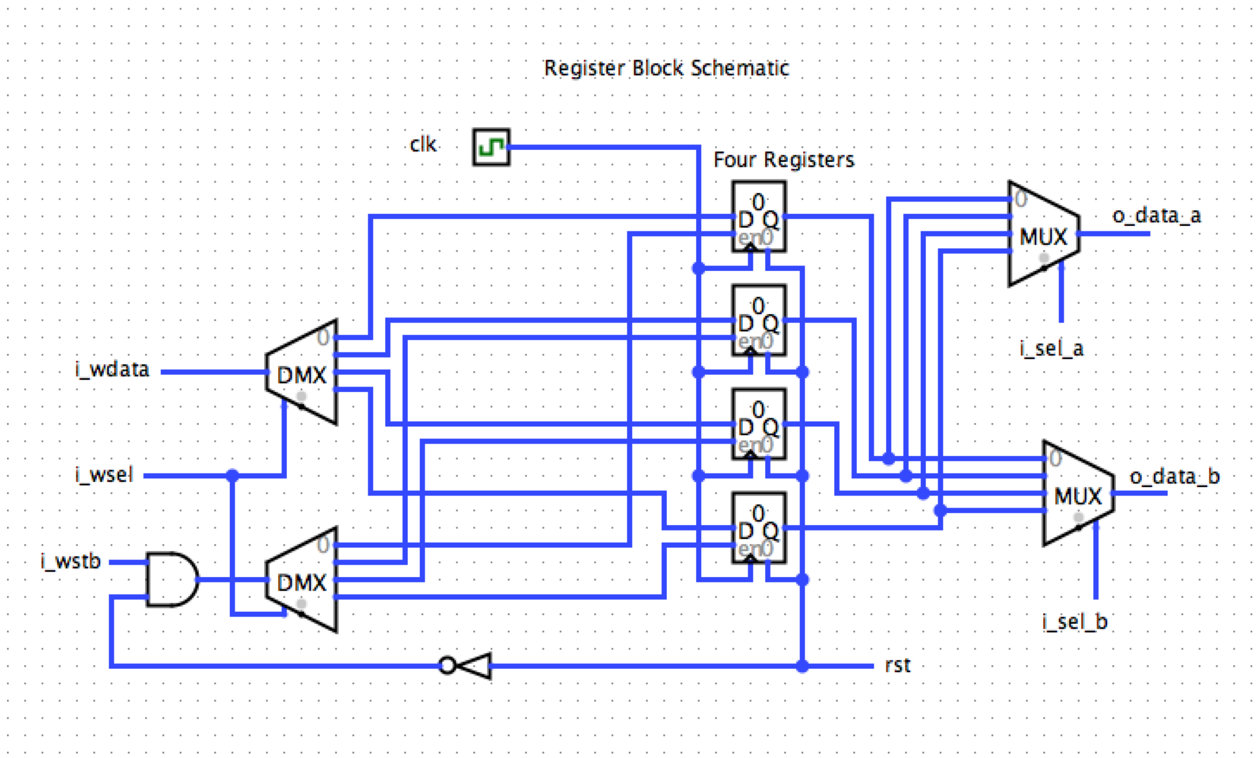
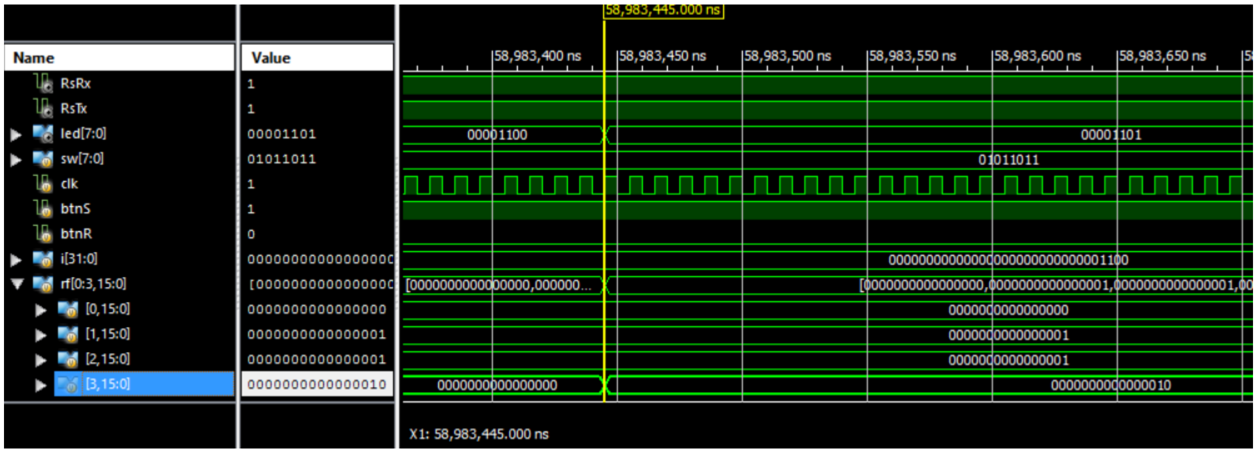


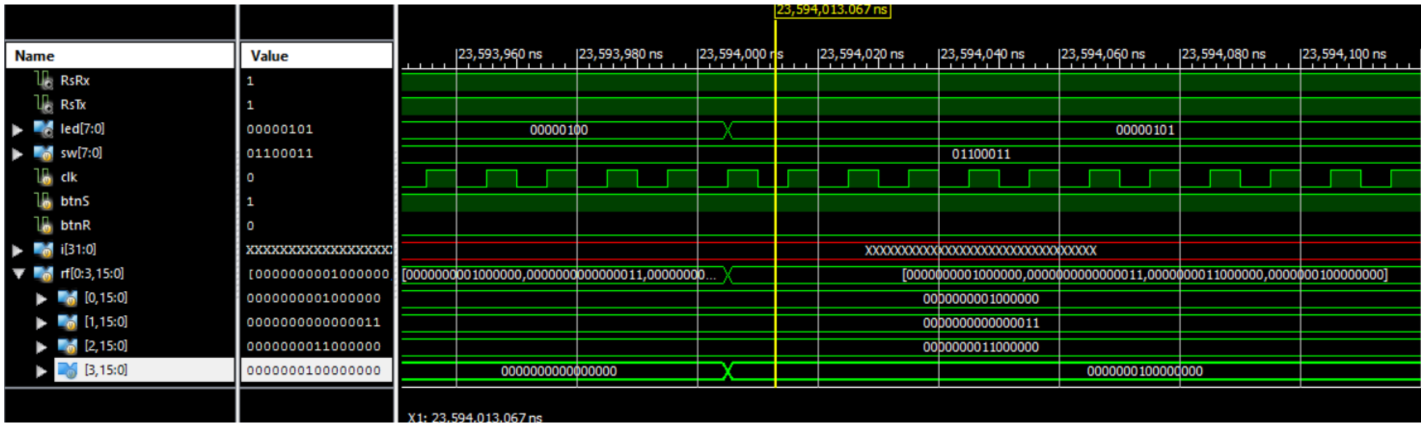
Figure 3: The basic Logisim schematic that shows the selection of data from the four different registers used for the four different instructions fed into the Nexsys3 board.

5.) Here is the waveform that shows the first instance that Register 3 stores a non-zero value in our Fibonacci program. The highlighted name is of Register 3.



Name	Value
RsRx	1
RsTx	1
led[7:0]	00001101
sw[7:0]	01011011
clk	1
btnS	1
btnR	0
i[31:0]	0000000000000000
rf[0:3,15:0]	[0000000000000000
[0,15:0]	0000000000000000
[1,15:0]	0000000000000001
[2,15:0]	0000000000000001
[3,15:0]	0000000000000010

Here is the waveform for the original given source code.



Name	Value
RsRx	1
RsTx	1
led[7:0]	00000101
sw[7:0]	01100011
clk	0
btnS	1
btnR	0
i[31:0]	XXXXXXXXXXXXXXXXXX
rf[0:3,15:0]	[0000000001000000
[0,15:0]	0000000001000000
[1,15:0]	0000000000000011
[2,15:0]	0000000011000000
[3,15:0]	0000000100000000