**CSM152A – Lab 3**

Names: Jahan Kuruvilla Cherian; Kevin Xu

Group: Lab 6 – Group 9

UID: 104436427; 704303603

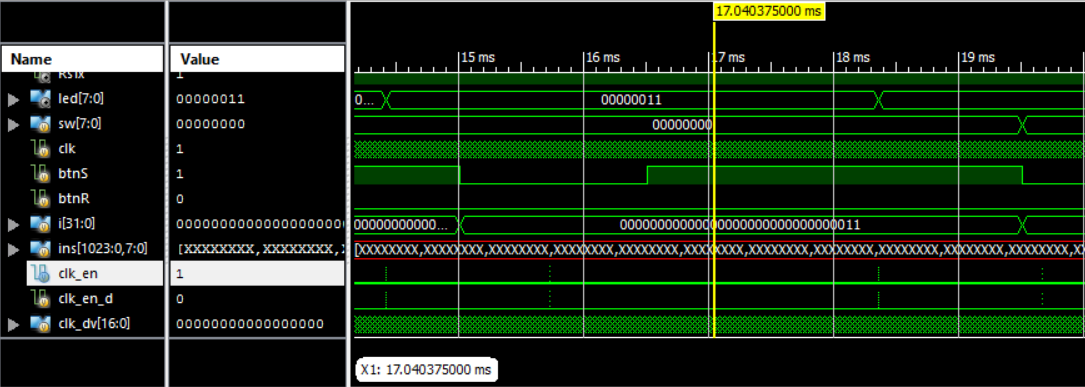
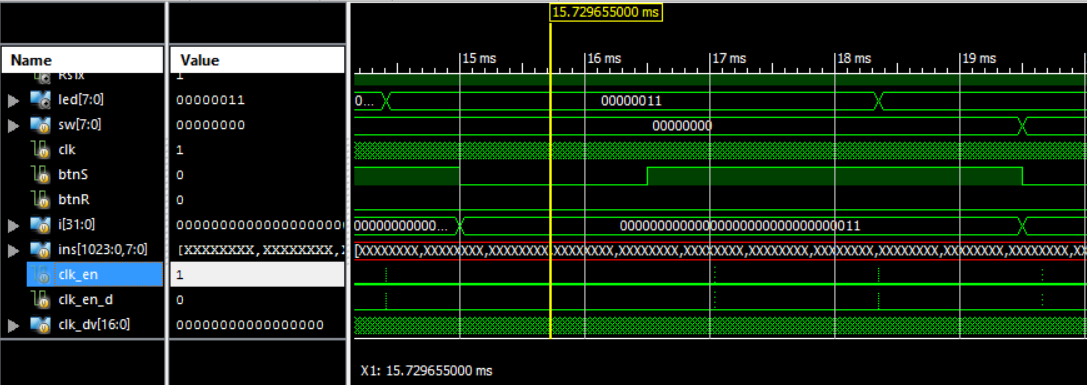
Date Performed: 4/28

TA: Babak Moatamed

**Workshop 1**

**Clock Enable**

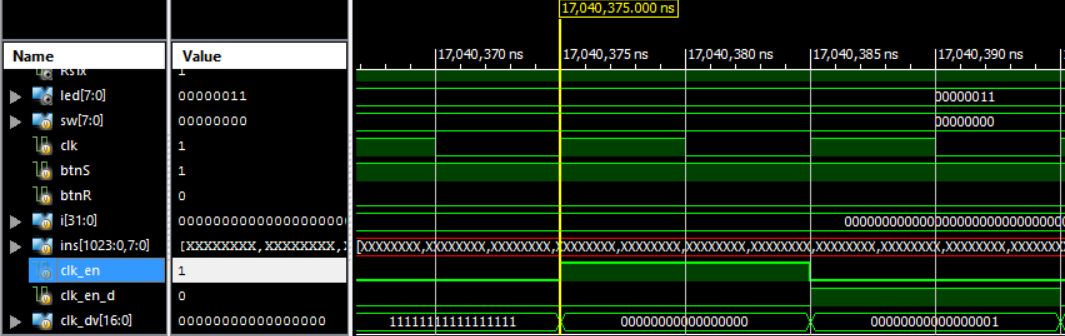
1. To understand what clk\_en really does we looked at the clock enable section of the nexys3.v file. We notice that at every positive edge of the master clock cycle, an 18-bit register clk\_dv\_inc is incremented by 1, acting almost like a counter. When the reset is enabled then all our values are reset 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.



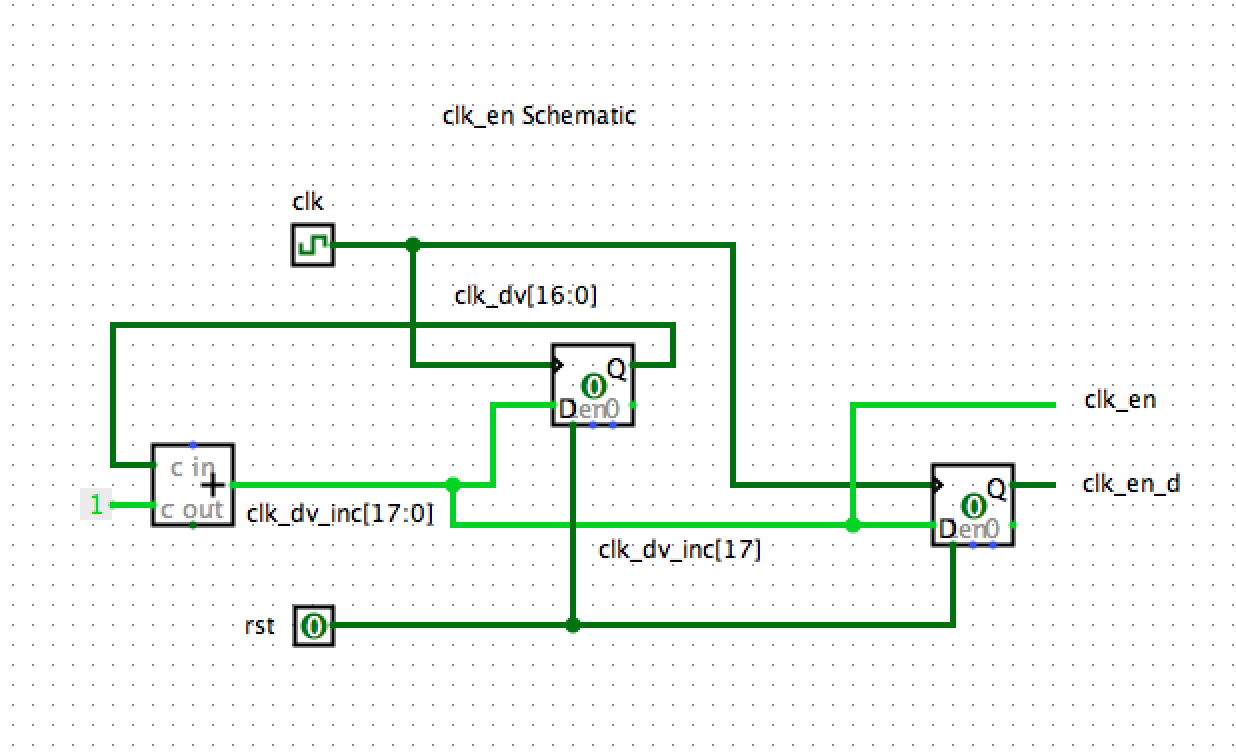
The period is 17.030375ms – 15.729655ms = 1.31072ms.

The frequency is 763Hz.

1. 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.



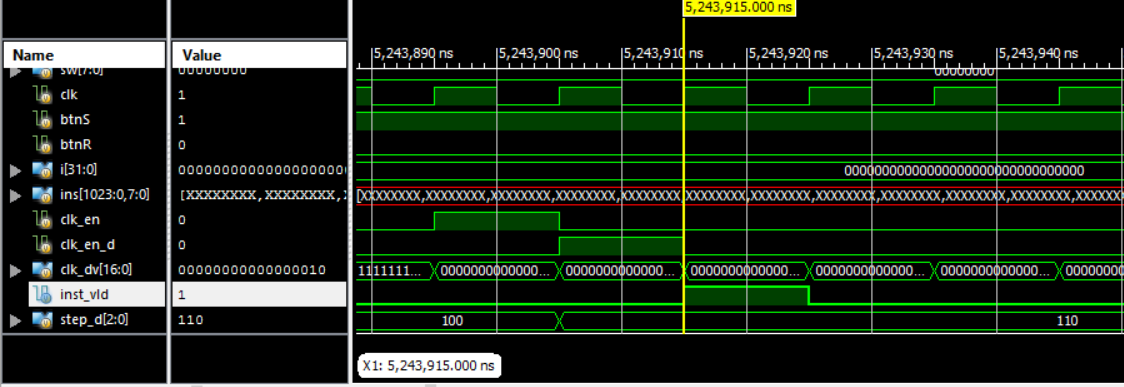
1. Using Logisim we translated the clock enable section of the code into a schematic as shown below in Figure 1.

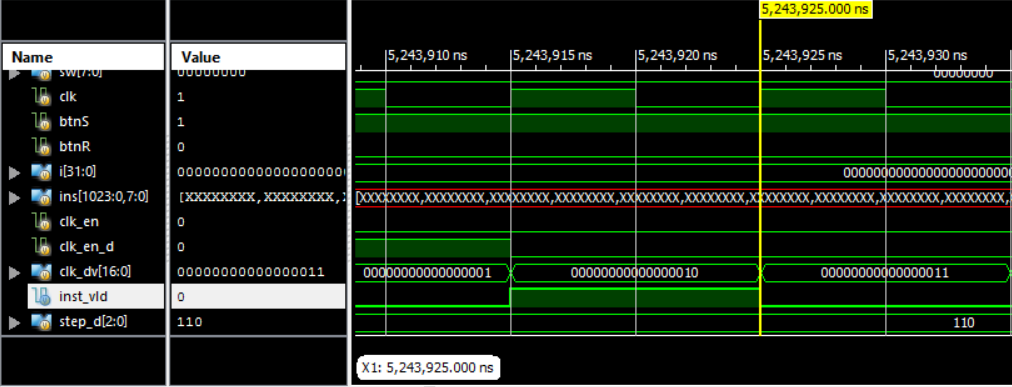


**Figure 1:** Logisim schematic showing how the clk\_en, clk\_en\_d, clk\_dv and clk\_dv\_inc signals are set from the Verilog code in nexsys3.v

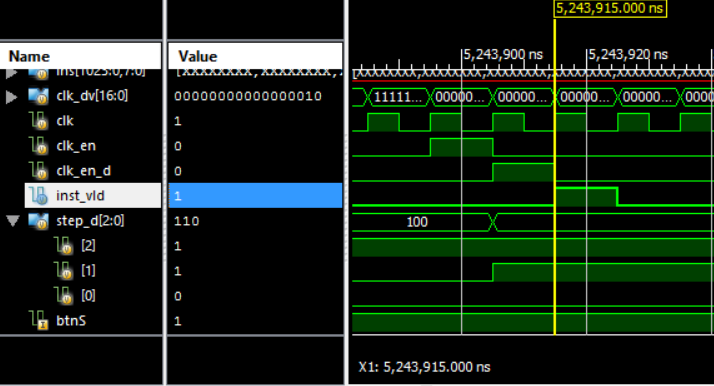
**Instruction Valid**

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.

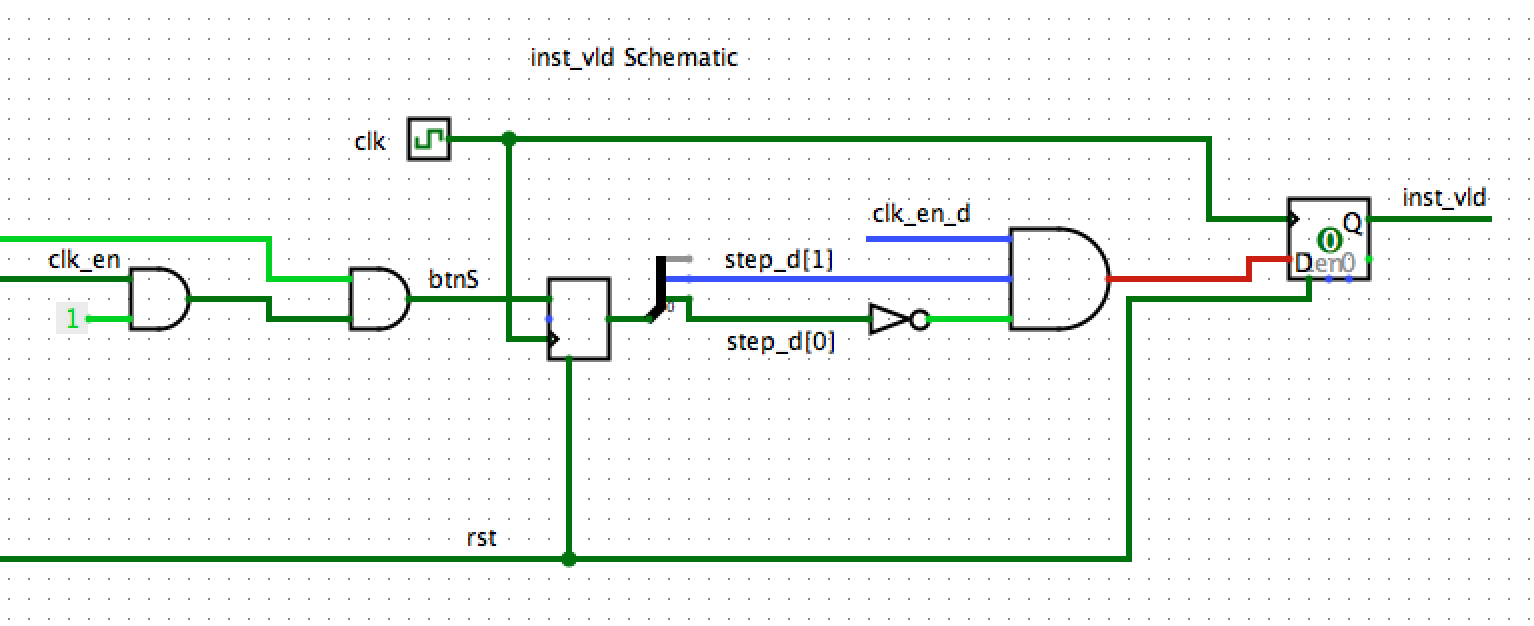




1. clk\_en\_d is essentially a delayed version of clk\_en. The reason this is used in setting the new value of step\_d (which is then later used in the inst\_vld signal) is because the clk\_en value is set back to 0 once step\_d changes. Thus using this would result in inst\_vld always evaluating to 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.
2. 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.



1. 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**

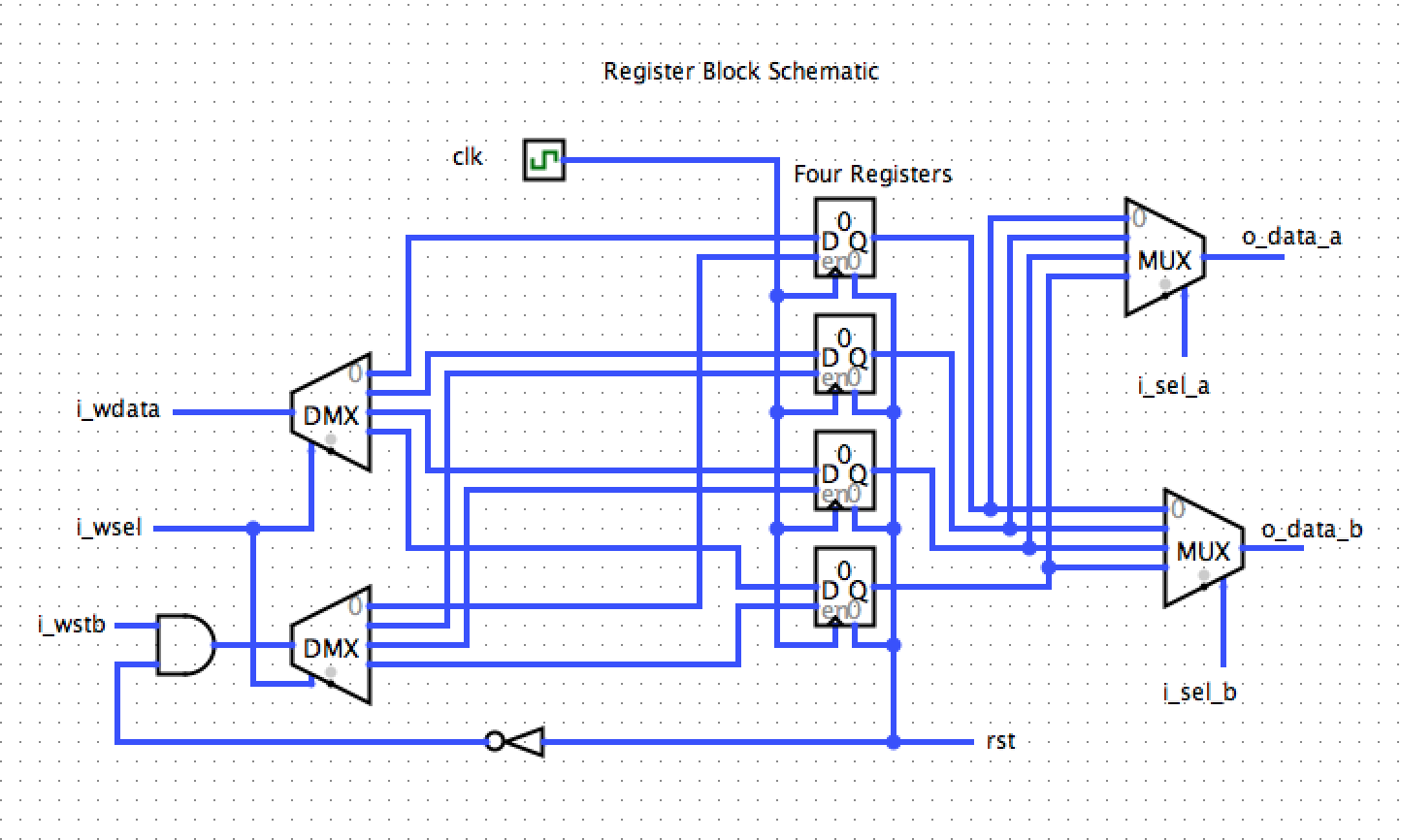
1. Running the simulation for the program with the seq\_rf.v module gave us insight into answering the questions presented in Workshop 1.
2. The register is first assigned a non zero value on line 33 of the seq\_rf.v Verilog code. The line is rf[i\_wsel] <= i\_wdata;. This line is clearly indicative 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];

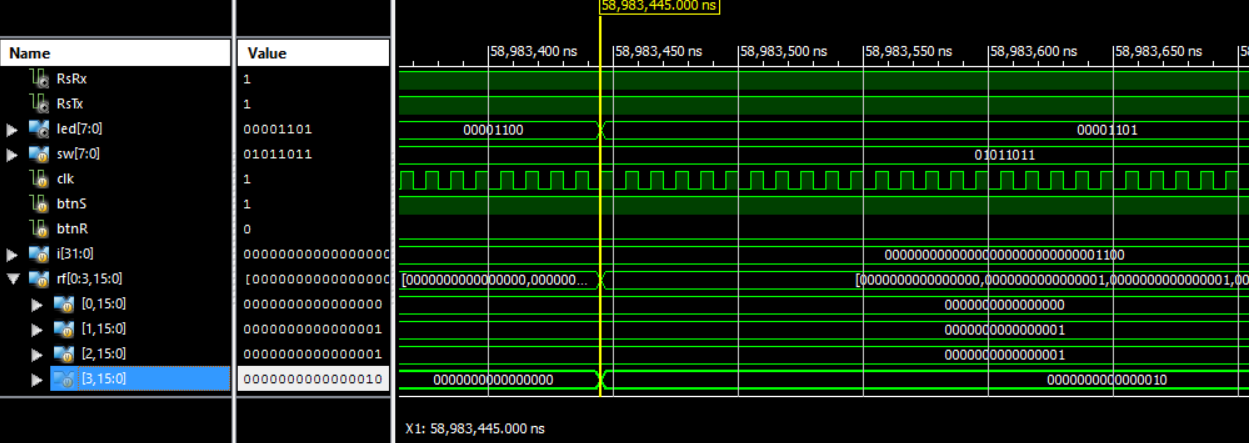
assign o\_data\_b = rf[i\_sel\_b];

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

1. Figure 3 shows us the Logisim schematic of the register file block.
2. 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.



**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.



Here is the waveform for the original given source code.

