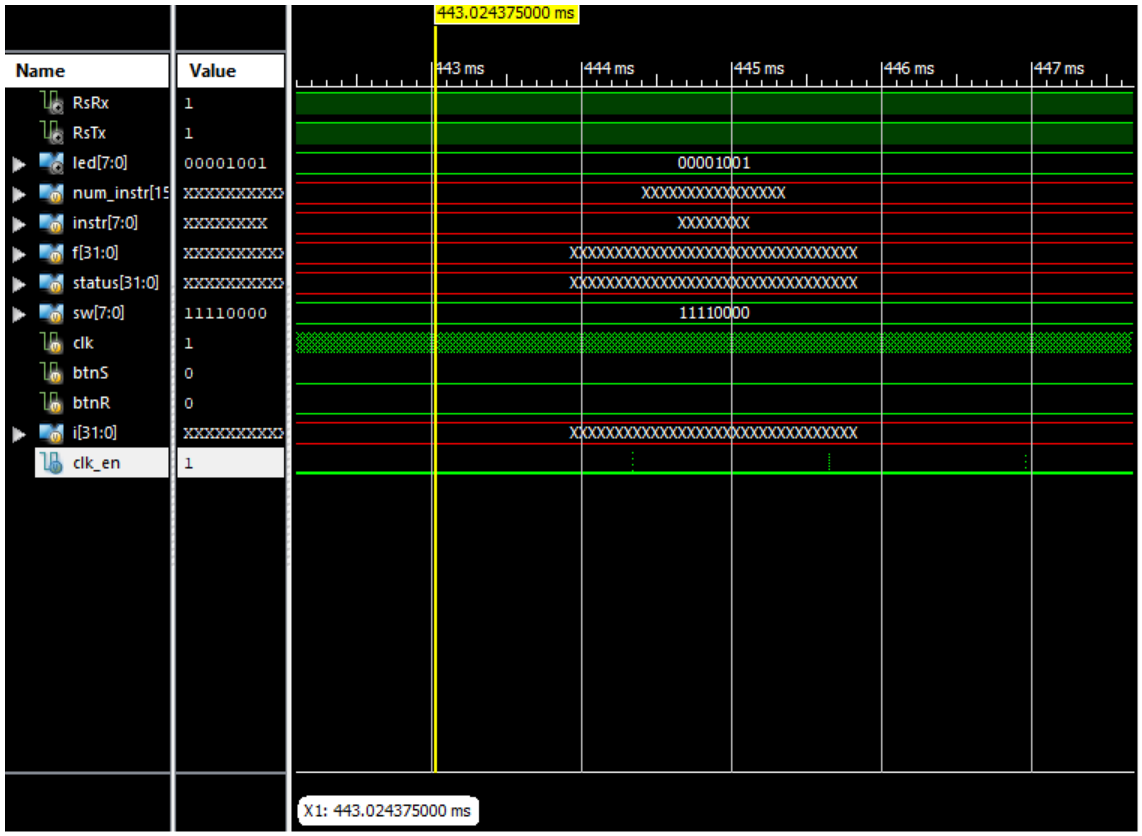
Workshop 1:

Clock Dividers:

1. **Question:** Add clk_en to the simulation's waveform tab and then run the simulation again. Use the cursor to find the periodicity of this signal (you can select the signal and use arrow keys to reach the exact edges). Capture a waveform picture that shows two occurrences of clk_en, and include it in the lab report. Indicate the exact period of the signal in the report.

Answer:



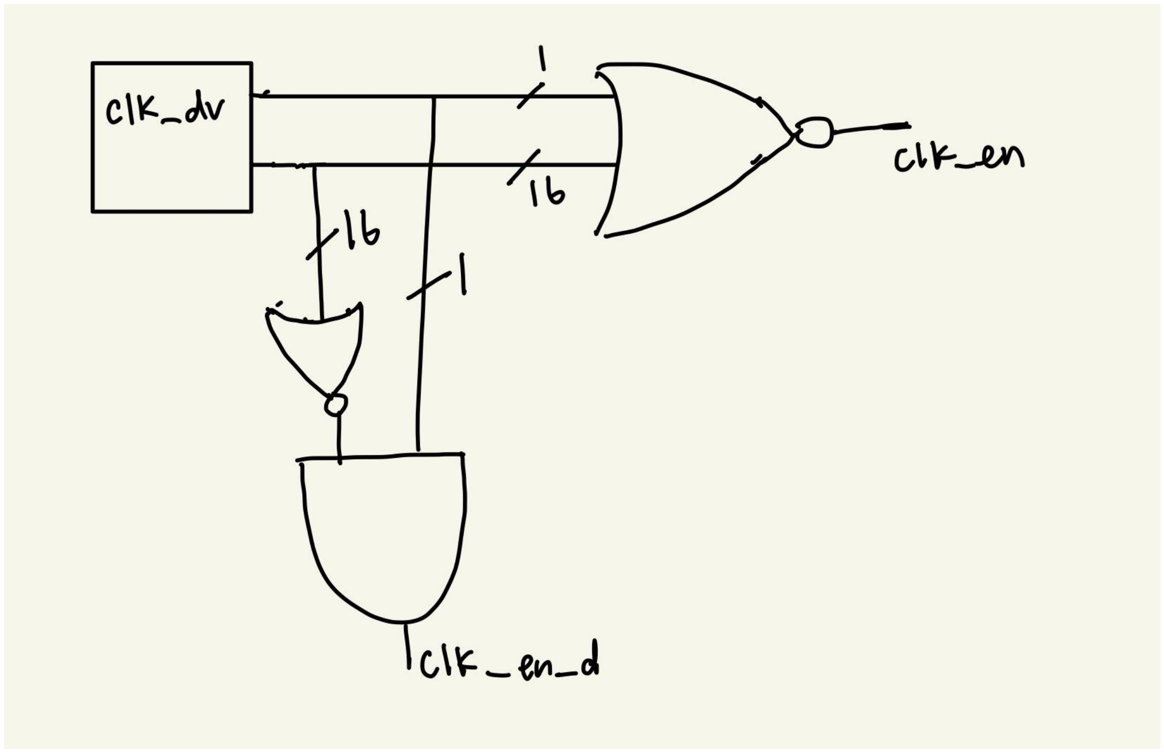
Name	Value
RsRx	1
RsTx	1
led[7:0]	00001001
num_instr[15	XXXXXXXXXX
instr[7:0]	XXXXXXXX
f[31:0]	XXXXXXXXXX
status[31:0]	XXXXXXXXXX
sw[7:0]	11110000
clk	1
btnS	0
btnR	0
i[31:0]	XXXXXXXXXX
clk_en	1

Period of signal: $1.310ms$

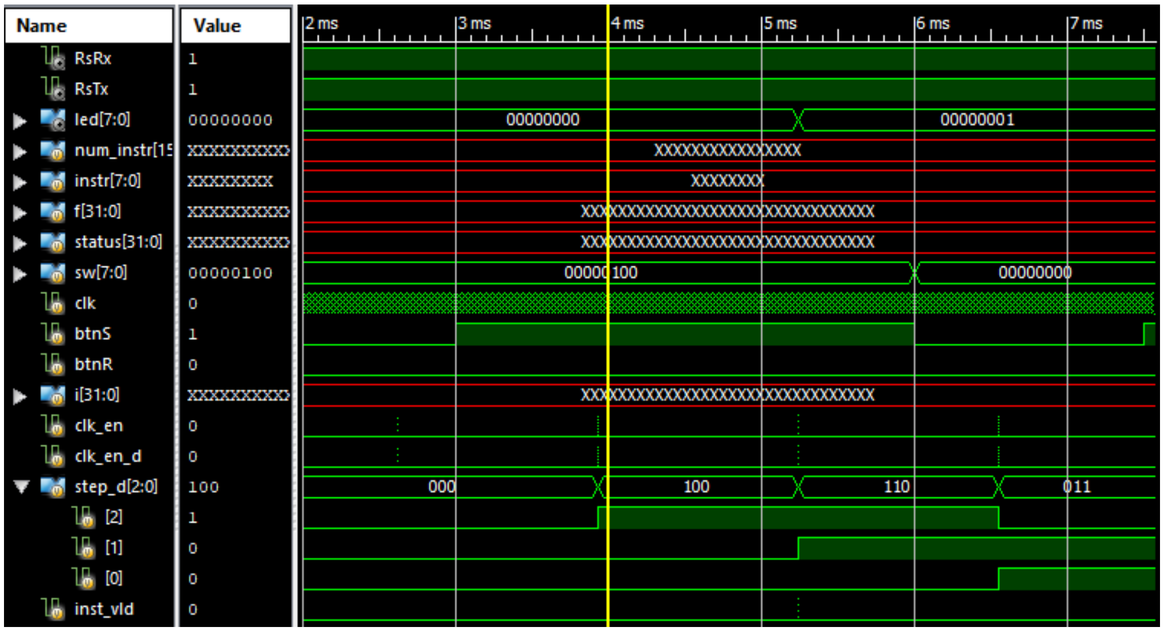
2. **Question:** A duty cycle is the percentage of one period in which a signal or system is active: $D = T / P \times 100\%$, where D is the duty cycle, T is the interval where the signal is high, and p is the period. What is the exact duty cycle of clk_en signal?

Answer: $D = \frac{T}{P} \times 100\% = 7.63358779 \times 10^{-4}\ \%$

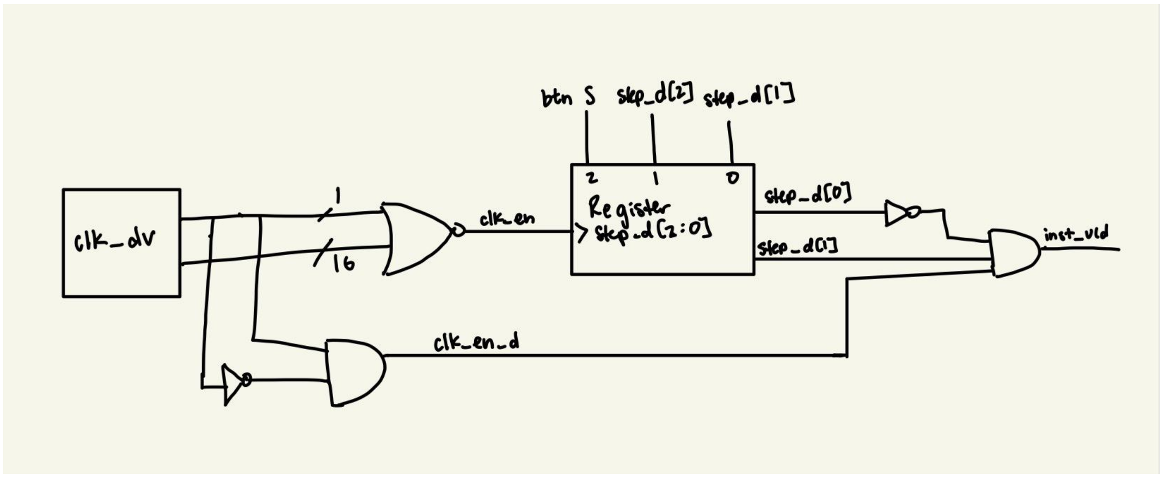
3. **Question: What is the value of clk_dv signal during the clock cycle that clk_en is high?**

	Answer: The value of the `clk_dv` signal during the clock cycle where `clk_en` is high is **0**.
4.	**Question: Draw a simple schematic/diagram of signals clk_dv, clk_en, and clk_en_d signals. It should be a translation of the corresponding Verilog code.**
	Answer: 

Debouncing:	
1.	**Question:** What is the purpose of clk_en_d signal when used in expression ~step_d[0] & step_d[1] & clk_en_d? Why don't we use clk_en?
	Answer: `clk_en_d` is used for debouncing and gets the signal from `clk_en`. `clk_en_d` will go high right after `clk_en` goes high, meaning it is delayed by one clock cycle. This enables debouncing to prevent multiple signals being sent. After the clock goes high, it will disable the next clock tick. If `clk_en_d` is high then it won't be 1 because that would be two clock ticks in a row. This is what prevents getting multiple signals in a row from pressing the button.
2.	**Question:** Instead of clk_en <= clk_dv_inc[17], can we do clk_en <= clk_dv[16], making the duty cycle of clk_en 50%? Why?

	Answer: Yes. `clk_dv[16]` gets its value from `clk_dv_inc[16]`, so the most significant bit of `clk_dv` will end up being 0 the same number of times it is 1 after enough time. In other words, yes, it will work because the counts will end up being the same.
3.	**Question:** Include waveform captures that clearly show the timing relationship between clk_en, step_d[1], step_d[0], btnS, clk_en_d, and inst_vld.
	Answer: Waveform captures that clearly show the timing relationship between `clk_en`, `step_d[1]`, `step_d[0]`, `btnS`, `clk_en_d`, and `inst_vld`. Waveform: 
4.	**Question:** Draw a simple schematic/diagram of the signals above. It should be a translation of the corresponding Verilog code.
	Answer:

Name	Value
RsRx	1
RsTx	1
led[7:0]	00000000
num_instr[15	XXXXXXXXXX
instr[7:0]	XXXXXXXX
f[31:0]	XXXXXXXXXX
status[31:0]	XXXXXXXXXX
sw[7:0]	00000100
clk	0
btnS	1
btnR	0
i[31:0]	XXXXXXXXXX
clk_en	0
clk_en_d	0
step_d[2:0]	100
[2]	1
[1]	0
[0]	0
inst_vld	0



Register File:

1. **Question:** Find the line of code where a register is written with a non-zero value. Is this sequential logic or combinatorial logic?

   Answer: Line of code:

```
    else if (i_wstb)
      rf[i_wsel] <= i_wdata;
```

   The logic is sequential since it uses non-blocking assignment, <=.

2. **Question:** Find the lines of code where the register values are read out from the register file. Is this sequential or combinatorial logic? If you were to manually implement the readout logic, what kind of logic elements would you use?

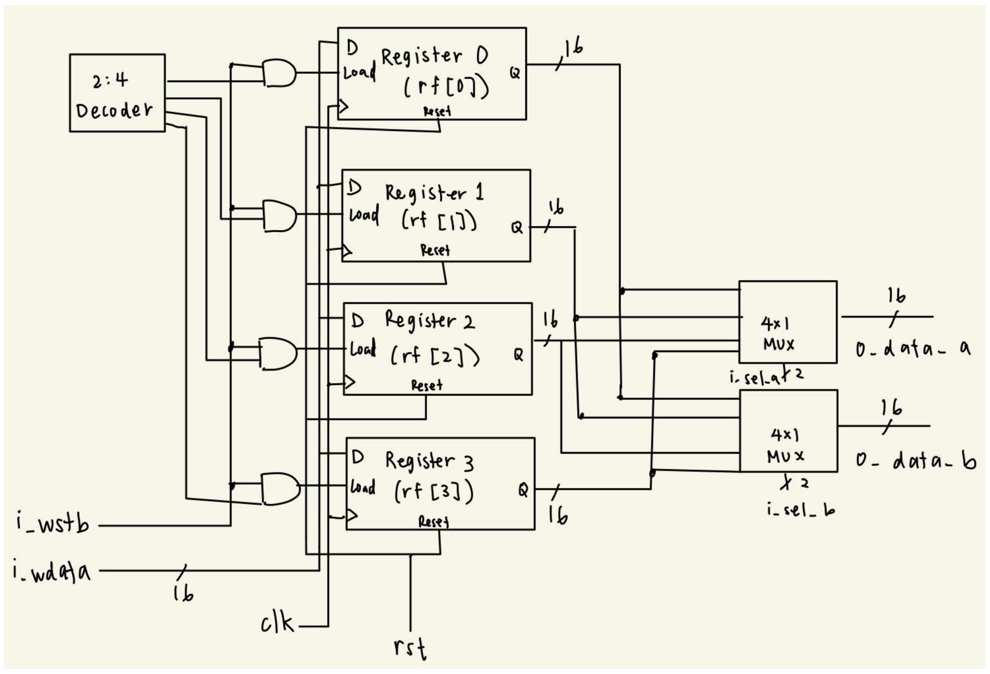
   Line of code:

```
  assign o_data_a = rf[i_sel_a];
  assign o_data_b = rf[i_sel_b];
```

   This is combinatorial logic because it doesn't use data from "history". To manually implement the readout logic, we could use two 4x1 multiplexers.

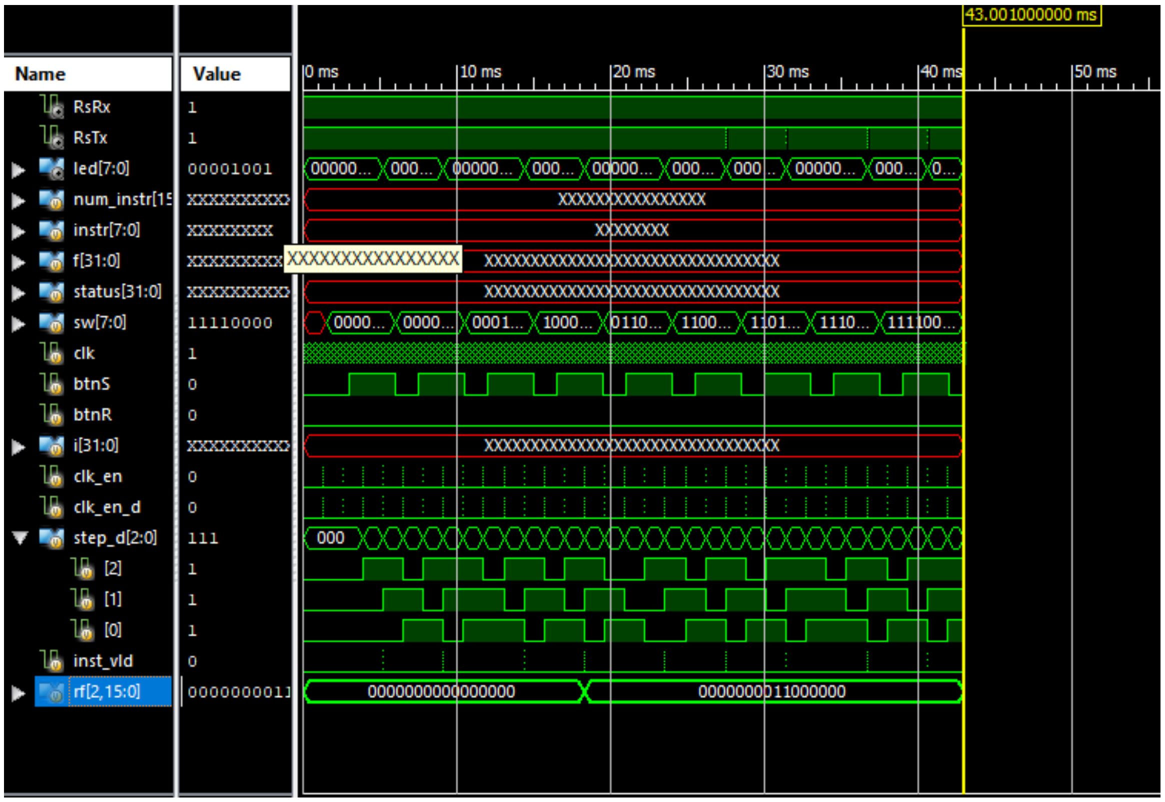
3. **Question:** Draw a circuit diagram of the register file block. It should be a translation of the corresponding Verilog code.

   Answer:



4. **Question:** Capture a waveform that shows the first time register 3 is written with a non-zero value.

Answer:

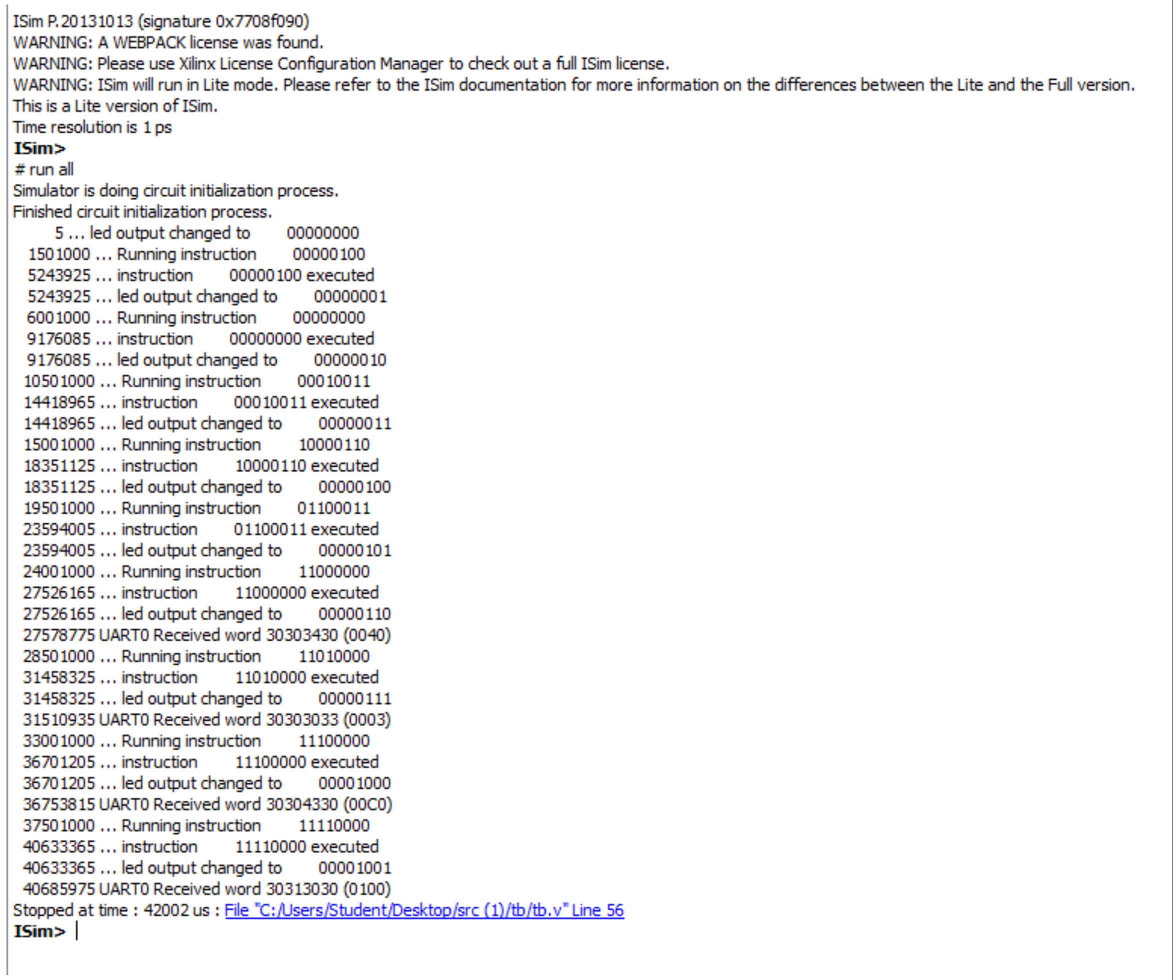


Name	Value
RsRx	1
RsTx	1
led[7:0]	00001001
num_instr[15	XXXXXXXXXX
instr[7:0]	XXXXXXXX
f[31:0]	XXXXXXXXX
status[31:0]	XXXXXXXXXX
sw[7:0]	11110000
clk	1
btnS	0
btnR	0
i[31:0]	XXXXXXXXXX
clk_en	0
clk_en_d	0
step_d[2:0]	111
[2]	1
[1]	1
[0]	1
inst_vld	0
rf[2,15:0]	000000001

```
led[7:0]     00000... 000... 00000... 000... 00000... 000... 000... 00000... 000... 0...
num_instr    XXXXXXXXXXXXXXXX
instr[7:0]   XXXXXXXX
f[31:0]      XXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
status[31:0] XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
sw[7:0]      0000... 0000... 0001... 1000... 0110... 1100... 1101... 1110... 111100...
i[31:0]      XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
step_d[2:0]  000
rf[2,15:0]   0000000000000000   0000000011000000
```

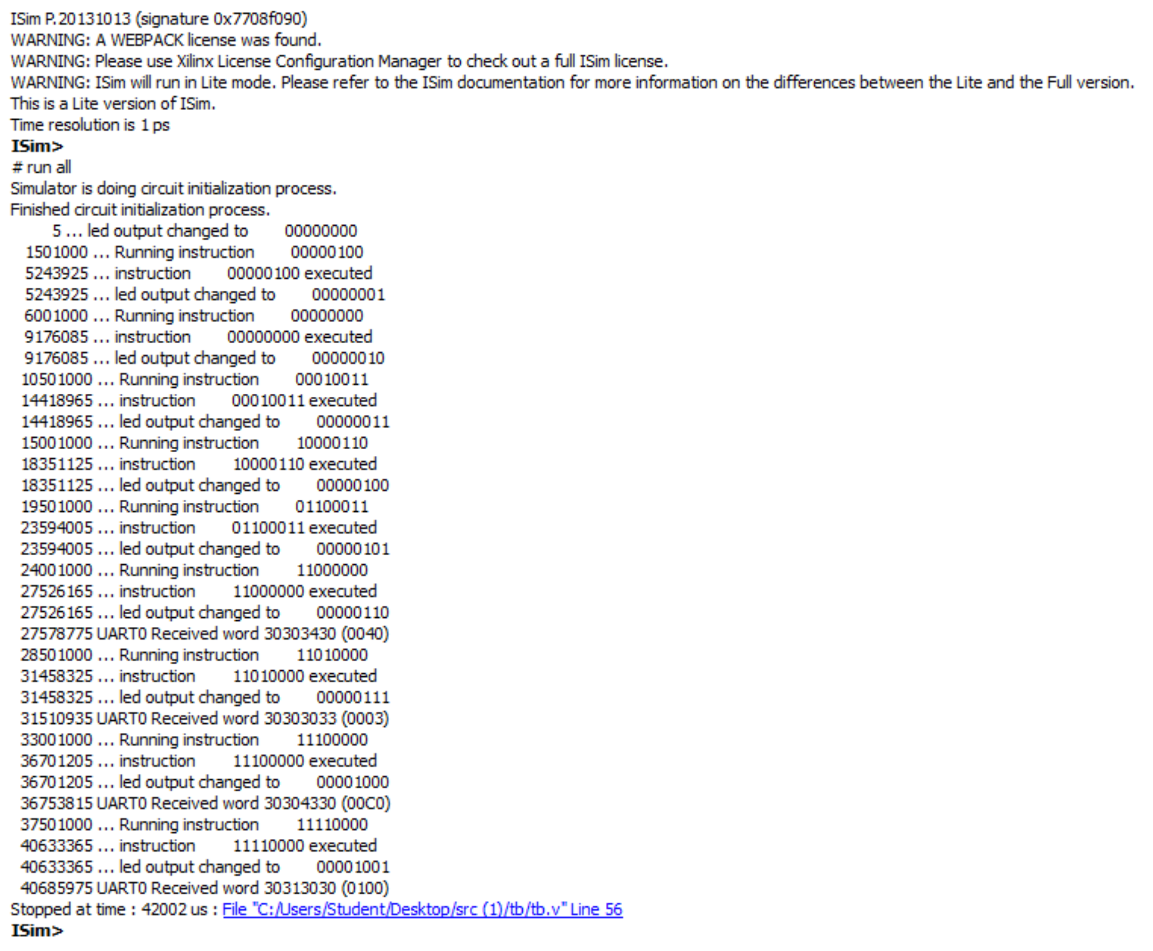
Workshop 2:

Nicer UART Output



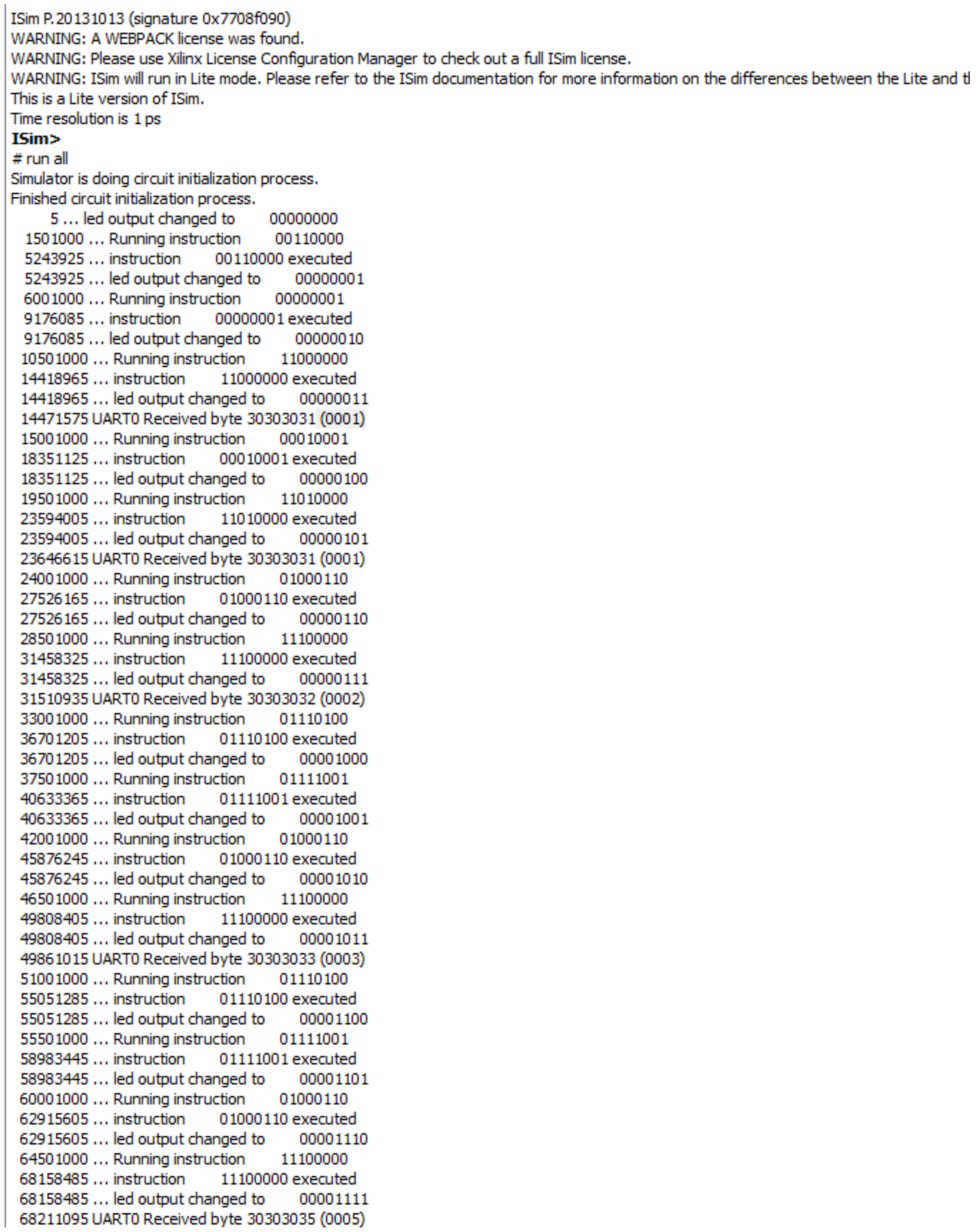
```
ISim P.20131013 (signature 0x7708f090)
WARNING: A WEBPACK license was found.
WARNING: Please use Xilinx License Configuration Manager to check out a full ISim license.
WARNING: ISim will run in Lite mode. Please refer to the ISim documentation for more information on the differences between the Lite and the Full version.
This is a Lite version of ISim.
Time resolution is 1 ps
ISim>
# run all
Simulator is doing circuit initialization process.
Finished circuit initialization process.
       5 ... led output changed to      00000000
 1501000 ... Running instruction      00000100
 5243925 ... instruction      00000100 executed
 5243925 ... led output changed to      00000001
 6001000 ... Running instruction      00000000
 9176085 ... instruction      00000000 executed
 9176085 ... led output changed to      00000010
10501000 ... Running instruction      00010011
14418965 ... instruction      00010011 executed
14418965 ... led output changed to      00000011
15001000 ... Running instruction      10000110
18351125 ... instruction      10000110 executed
18351125 ... led output changed to      00000100
19501000 ... Running instruction      01100011
23594005 ... instruction      01100011 executed
23594005 ... led output changed to      00000101
24001000 ... Running instruction      11000000
27526165 ... instruction      11000000 executed
27526165 ... led output changed to      00000110
27578775 UART0 Received word 30303430 (0040)
28501000 ... Running instruction      11010000
31458325 ... instruction      11010000 executed
31458325 ... led output changed to      00000111
31510935 UART0 Received word 30303033 (0003)
33001000 ... Running instruction      11100000
36701205 ... instruction      11100000 executed
36701205 ... led output changed to      00001000
36753815 UART0 Received word 30304330 (00C0)
37501000 ... Running instruction      11110000
40633365 ... instruction      11110000 executed
40633365 ... led output changed to      00001001
40685975 UART0 Received word 30313030 (0100)
Stopped at time : 42002 us : File "C:/Users/Student/Desktop/src (1)/tb/tb.v" Line 56
ISim> |
```

An Easier Way to Load Sequencer Program



```
ISim P.20131013 (signature 0x7708f090)
WARNING: A WEBPACK license was found.
WARNING: Please use Xilinx License Configuration Manager to check out a full ISim license.
WARNING: ISim will run in Lite mode. Please refer to the ISim documentation for more information on the differences between the Lite and the Full version.
This is a Lite version of ISim.
Time resolution is 1 ps
ISim>
# run all
Simulator is doing circuit initialization process.
Finished circuit initialization process.
       5 ... led output changed to       00000000
 1501000 ... Running instruction       00000100
 5243925 ... instruction       00000100 executed
 5243925 ... led output changed to       00000001
 6001000 ... Running instruction       00000000
 9176085 ... instruction       00000000 executed
 9176085 ... led output changed to       00000010
10501000 ... Running instruction       00010011
14418965 ... instruction       00010011 executed
14418965 ... led output changed to       00000011
15001000 ... Running instruction       10000110
18351125 ... instruction       10000110 executed
18351125 ... led output changed to       00000100
19501000 ... Running instruction       01100011
23594005 ... instruction       01100011 executed
23594005 ... led output changed to       00000101
24001000 ... Running instruction       11000000
27526165 ... instruction       11000000 executed
27526165 ... led output changed to       00000110
27578775 UART0 Received word 30303430 (0040)
28501000 ... Running instruction       11010000
31458325 ... instruction       11010000 executed
31458325 ... led output changed to       00000111
31510935 UART0 Received word 30303033 (0003)
33001000 ... Running instruction       11100000
36701205 ... instruction       11100000 executed
36701205 ... led output changed to       00001000
36753815 UART0 Received word 30304330 (00C0)
37501000 ... Running instruction       11110000
40633365 ... instruction       11110000 executed
40633365 ... led output changed to       00001001
40685975 UART0 Received word 30313030 (0100)
Stopped at time : 42002 us : File "C:/Users/Student/Desktop/src (1)/tb/tb.v" Line 56
ISim>
```

Fibonacci Numbers



```
ISim P.20131013 (signature 0x7708f090)
WARNING: A WEBPACK license was found.
WARNING: Please use Xilinx License Configuration Manager to check out a full ISim license.
WARNING: ISim will run in Lite mode. Please refer to the ISim documentation for more information on the differences between the Lite and t
This is a Lite version of ISim.
Time resolution is 1 ps
ISim>
# run all
Simulator is doing circuit initialization process.
Finished circuit initialization process.
        5 ... led output changed to        00000000
  1501000 ... Running instruction        00110000
  5243925 ... instruction        00110000 executed
  5243925 ... led output changed to        00000001
  6001000 ... Running instruction        00000001
  9176085 ... instruction        00000001 executed
  9176085 ... led output changed to        00000010
 10501000 ... Running instruction        11000000
 14418965 ... instruction        11000000 executed
 14418965 ... led output changed to        00000011
 14471575 UART0 Received byte 30303031 (0001)
 15001000 ... Running instruction        00010001
 18351125 ... instruction        00010001 executed
 18351125 ... led output changed to        00000100
 19501000 ... Running instruction        11010000
 23594005 ... instruction        11010000 executed
 23594005 ... led output changed to        00000101
 23646615 UART0 Received byte 30303031 (0001)
 24001000 ... Running instruction        01000110
 27526165 ... instruction        01000110 executed
 27526165 ... led output changed to        00000110
 28501000 ... Running instruction        11100000
 31458325 ... instruction        11100000 executed
 31458325 ... led output changed to        00000111
 31510935 UART0 Received byte 30303032 (0002)
 33001000 ... Running instruction        01110100
 36701205 ... instruction        01110100 executed
 36701205 ... led output changed to        00001000
 37501000 ... Running instruction        01111001
 40633365 ... instruction        01111001 executed
 40633365 ... led output changed to        00001001
 42001000 ... Running instruction        01000110
 45876245 ... instruction        01000110 executed
 45876245 ... led output changed to        00001010
 46501000 ... Running instruction        11100000
 49808405 ... instruction        11100000 executed
 49808405 ... led output changed to        00001011
 49861015 UART0 Received byte 30303033 (0003)
 51001000 ... Running instruction        01110100
 55051285 ... instruction        01110100 executed
 55051285 ... led output changed to        00001100
 55501000 ... Running instruction        01111001
 58983445 ... instruction        01111001 executed
 58983445 ... led output changed to        00001101
 60001000 ... Running instruction        01000110
 62915605 ... instruction        01000110 executed
 62915605 ... led output changed to        00001110
 64501000 ... Running instruction        11100000
 68158485 ... instruction        11100000 executed
 68158485 ... led output changed to        00001111
 68211095 UART0 Received byte 30303035 (0005)
```

seq.code:

```
100100
```

```
00110000
00000001
11000000
00010001
11010000
01000110
11100000
01110100
01111001
01000110
11100000
01110100
01111001
01000110
11100000
01110100
01111001
01000110
11100000
01110100
01111001
01000110
11100000
01110100
01111001
01000110
11100000
01110100
01111001
01000110
11100000
01110100
01111001
01000110
11100000
01110100
01111001
```

Reflection on what we learned

As a group we learned how difficult it is to use hardware description languages like Verilog. However, through practice we learned how to use the Xilinx software with the FPGA boards, how to modify and write testbench files, and how to run simulations and read the waveforms. We also gained insight on how machine instructions like push, add, multiply, etc. work in a computer system.