

Assignment 2

Sean Fahey

3C7

11/30/21

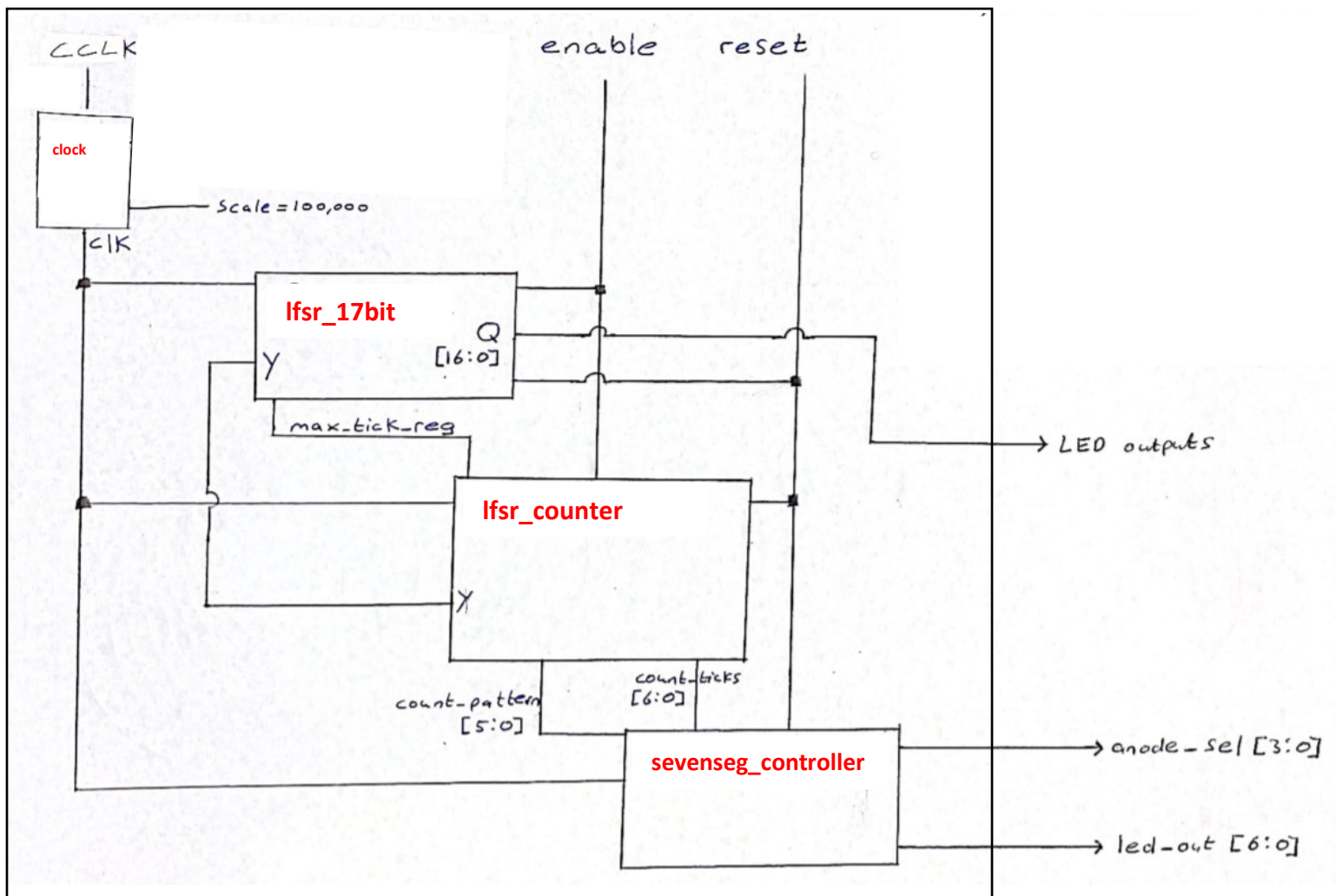
Introduction

The aim of this assignment is to implement a system comprising of a bit generator and a pattern detector. A Linear-Feedback-Shift-Register (LFSR) will generate a pseudo-random bitstream, which will be fed into a Finite-State-Machine (FSM), which will detect the given sequence, and count how many times such a sequence is seen in one full cycle of the LFSR.

The hardware described above will be implemented using a combination of a Basys3 board, Xilinx Vivado software, and the Verilog HDL.

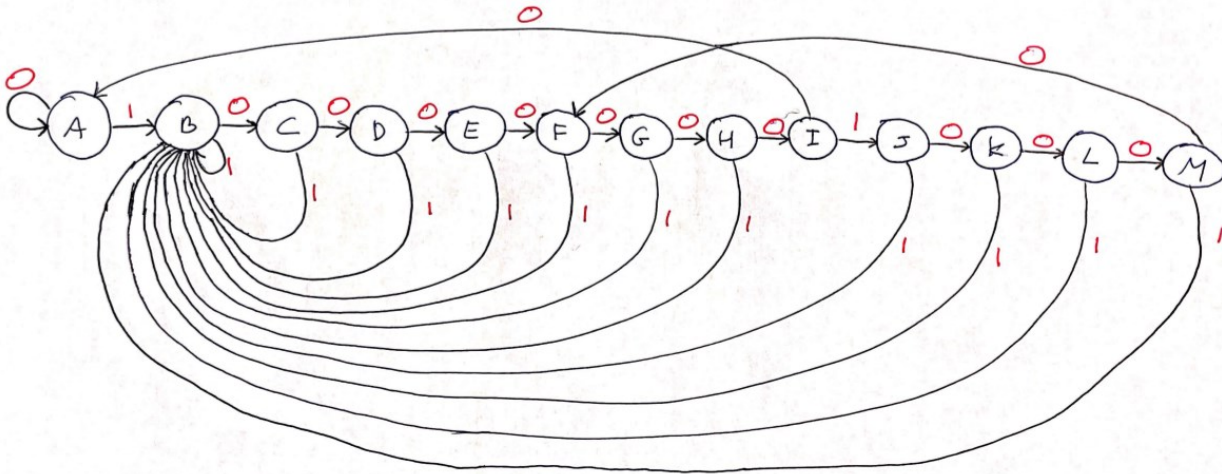
Functional Diagram

Top module



State Machine Diagram

Moore Model



This FSM needs 13 states (A-M) , as the pattern has so few transitions.

While until the first input is 1, we stay at A.

If we reach state M, we can go directly to state F or back to B.

If we reach I and the next input is 0, we go all the way back to A.

All other states will go to the next state if the input is 0, or back to B if the input is 1.

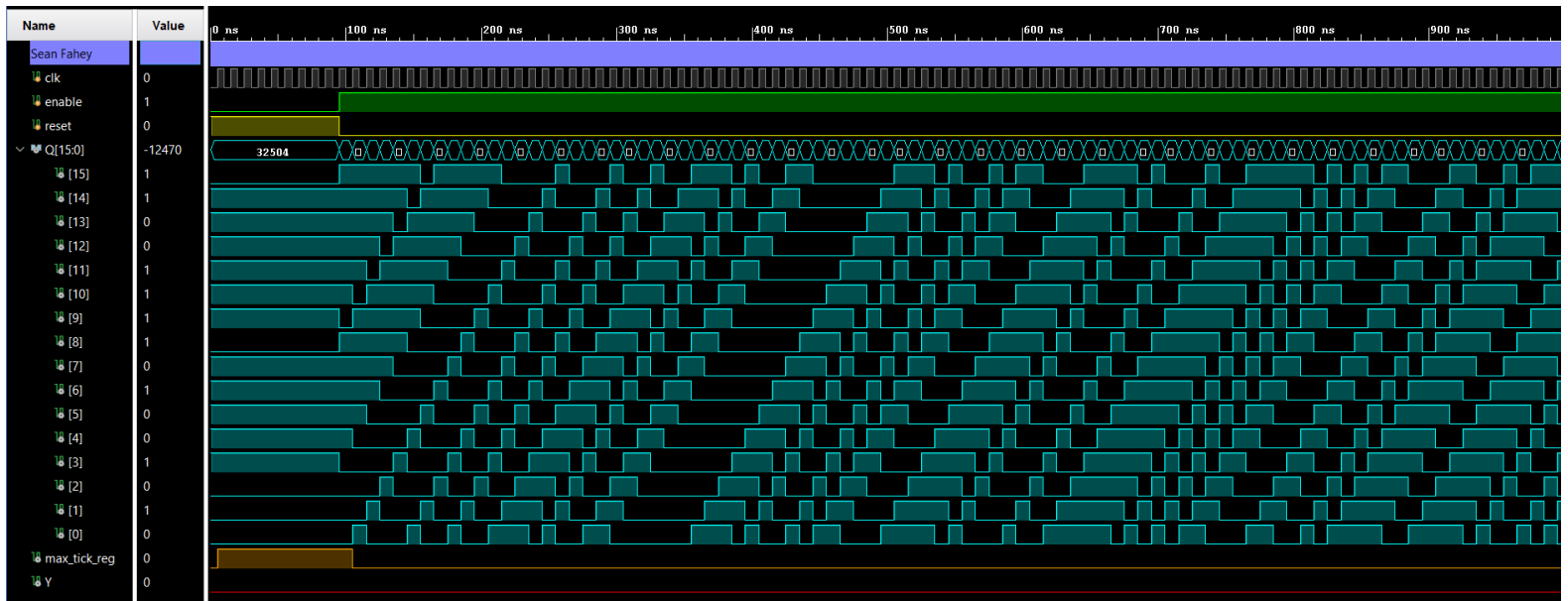
Test Plan

The testbench includes outputs from each module, as well as test vectors for the top module's inputs (enable, reset, clock). If any of the outputs have errors in them, it should be straightforward to go to the problem, as we know where it is coming from.

The waveform should show the correct transitions and the expected timing should be clear in the waveforms.

Finally, the project will be loaded onto the board and should work as expected.

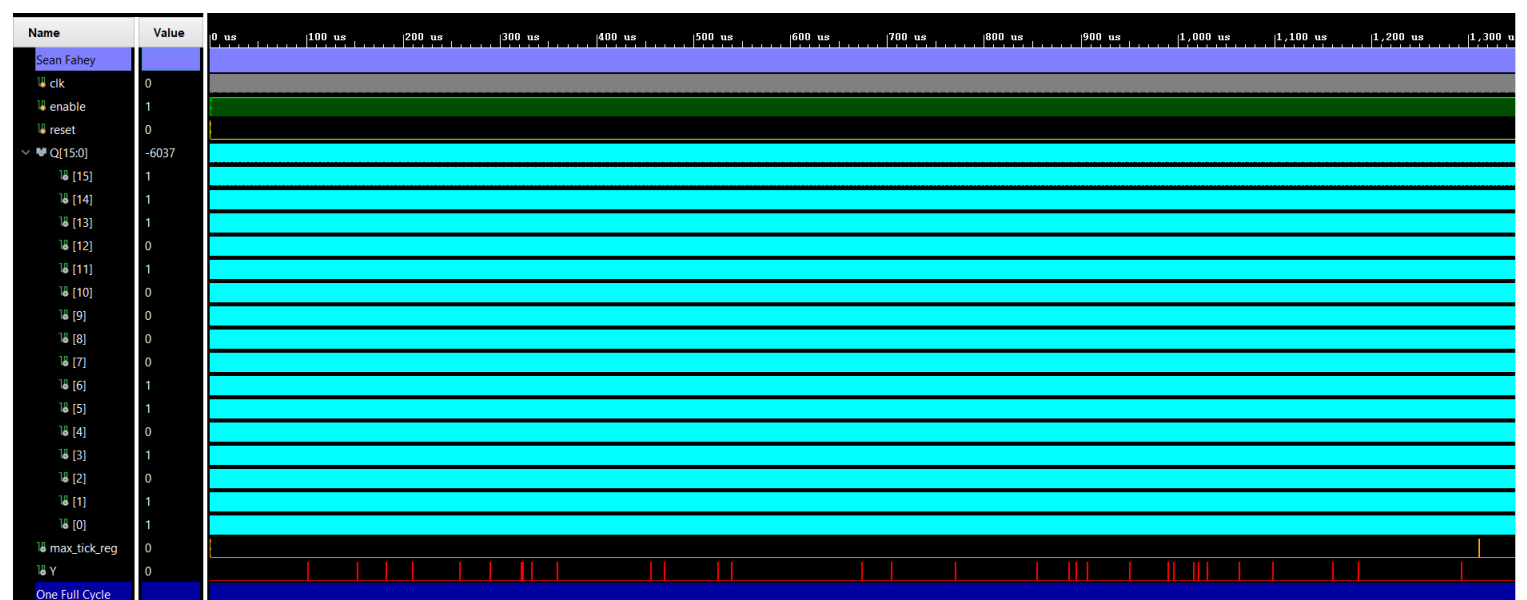
Testbench



We can see the tap outputs from the LFSR is being shifted every positive clock edge.

The tap outputs are held at the seed value while the *reset* bit is HIGH. The LFSR does not begin shifting values until *enable* is set to HIGH.

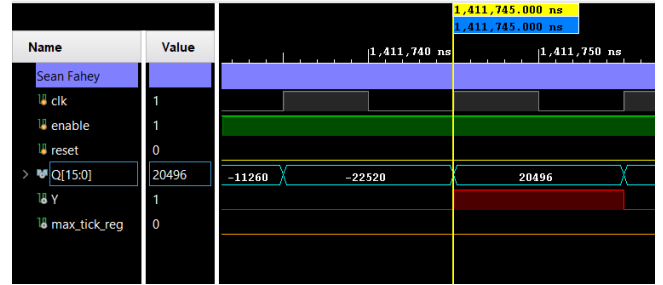
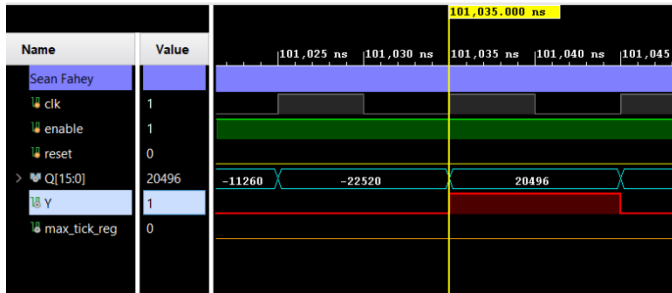
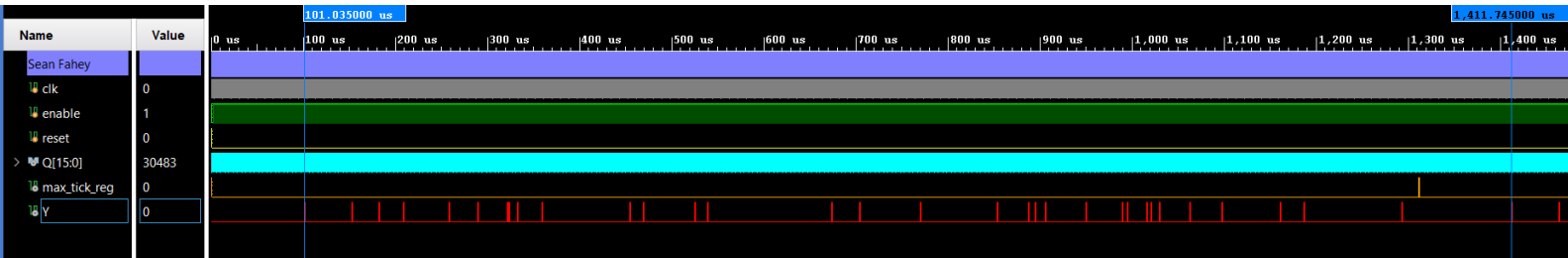
Note: at the bottom the *max_tick_reg* bit is incorrectly HIGH, as there has not been a full cycle. This has no impact on the function of the top module as it is dealt with in the *counter* module.



The outputs *Q* are too close together to see the transitions here. This waveform shows the function of *max_tick_reg* and *Y* working correctly (at the bottom)

Y is set to HIGH when the pattern is detected in the LFSR

max_tick_reg is HIGH when one full cycle of the max-length LFSR has been reached.



The output is the same at the two times marked above, which should indicate that the cycle has been completed and then restarted;

$1411.745\mu\text{s} - 101.035\mu\text{s} = \mathbf{1310.71\mu\text{s}}$ → time between two occurrences of the same tap outputs.

And we calculate that for a 17-bit LFSR the time for a full cycle (with period 10ns) should be













$10\text{ns} * 2^{17} = \mathbf{1310.72\mu\text{s}}$ (including first period)

As we can see, the LFSR shifts through all numbers in the cycle correctly.

Also, there are 32 'Y' pulses in the full cycle. This is correct because my LFSR is 17 bits long, and my pattern is 12 bits long.

$17 - 12 = 5$, and so there can be $2^5 = 32$ different values in the LFSR containing the 12-bit pattern.

Utilization Report

Reports	Design Runs	Utilization	×	Timi
				Register as Flip Flop
Name				Used
▼  lfsr_17bit_top				199
>  sevensseg (sevensseg_controller)				35
 clk_1Hz (clock_1)				33
 clk_10Hz (clock)				33
 clk_10kHz (clock_0)				33
 clk_500Hz (clock_2)				33
 LFSR_17 (lfsr_17bit)				19
 count (lfsr_counter)				13

Note:

Only one **clk** is required, but as described in the *Demo* section, there are 4 clock speeds to choose from because having one clock speed has the following problem:

If we have a *high* clock speed then we can see the *pattern matches* and *max_ticks* numbers incrementing within reasonable time, but we cannot see the LFSR shifting, as the LEDs are flashing too quickly.

If we have a *low* clock speed then we can see the LFSR shifting through the values, but we will never see any matches for pattern or the cycle tick (it would take several hours).

So having several clock speeds is necessary to demonstrate that the top module works correctly.

XDC

```
1 | # clock
2 | set_property PACKAGE_PIN W5 [get_ports CCLK]
3 | set_property IOSTANDARD LVCMOS33 [get_ports CCLK]
4 | create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports CCLK]
5 |
6 | #clock scale
7 | set_property PACKAGE_PIN R2 [get_ports clk_sw[0]]
8 | set_property IOSTANDARD LVCMOS33 [get_ports clk_sw[0]]
9 | set_property PACKAGE_PIN T1 [get_ports clk_sw[1]]
10 | set_property IOSTANDARD LVCMOS33 [get_ports clk_sw[1]]
11 | set_property PACKAGE_PIN U1 [get_ports clk_sw[2]]
12 | set_property IOSTANDARD LVCMOS33 [get_ports clk_sw[2]]
13 | set_property PACKAGE_PIN W2 [get_ports clk_sw[3]]
14 | set_property IOSTANDARD LVCMOS33 [get_ports clk_sw[3]]
```

The Basys3 internal clock at pin W5 is connected to port CCLK, and is initiated (line 4).

clk_sw [3:0] is a 4-bit number controlled by switches 12 to 15.

If switch 15 is ON, the clock ticks at 1Hz

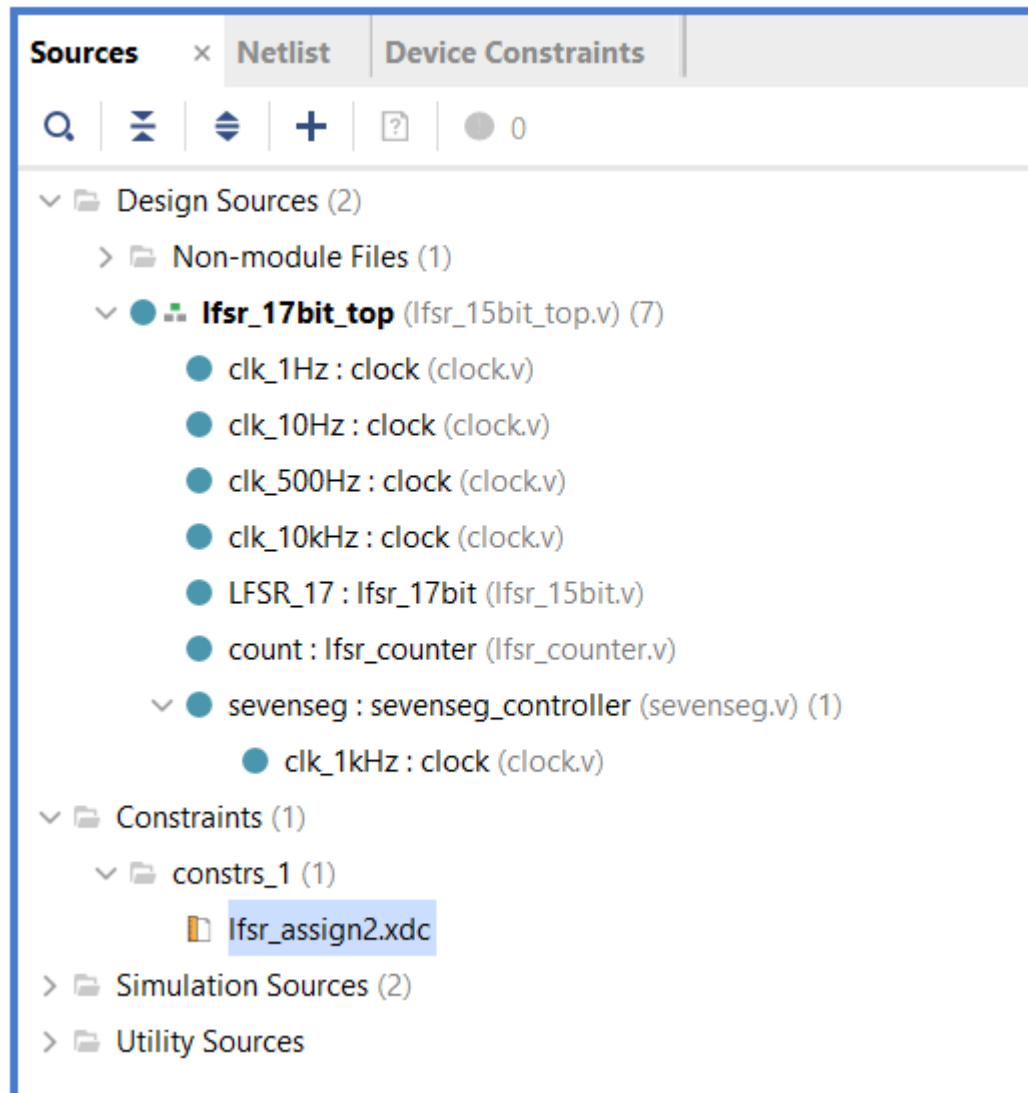
If switch 14 is ON, the tick ticks at 10Hz

If switch 13 is ON, the clock ticks at 500Hz

If switch 12 is ON, the clock ticks at 10kHz

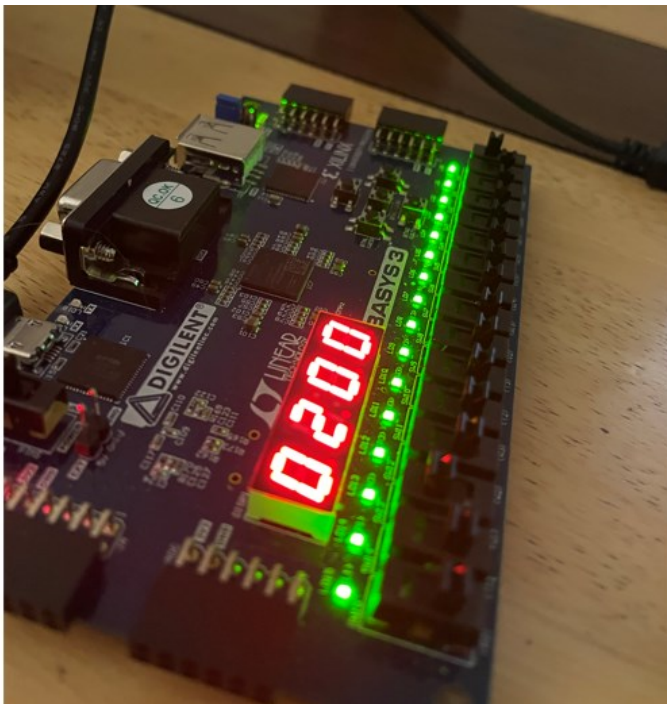
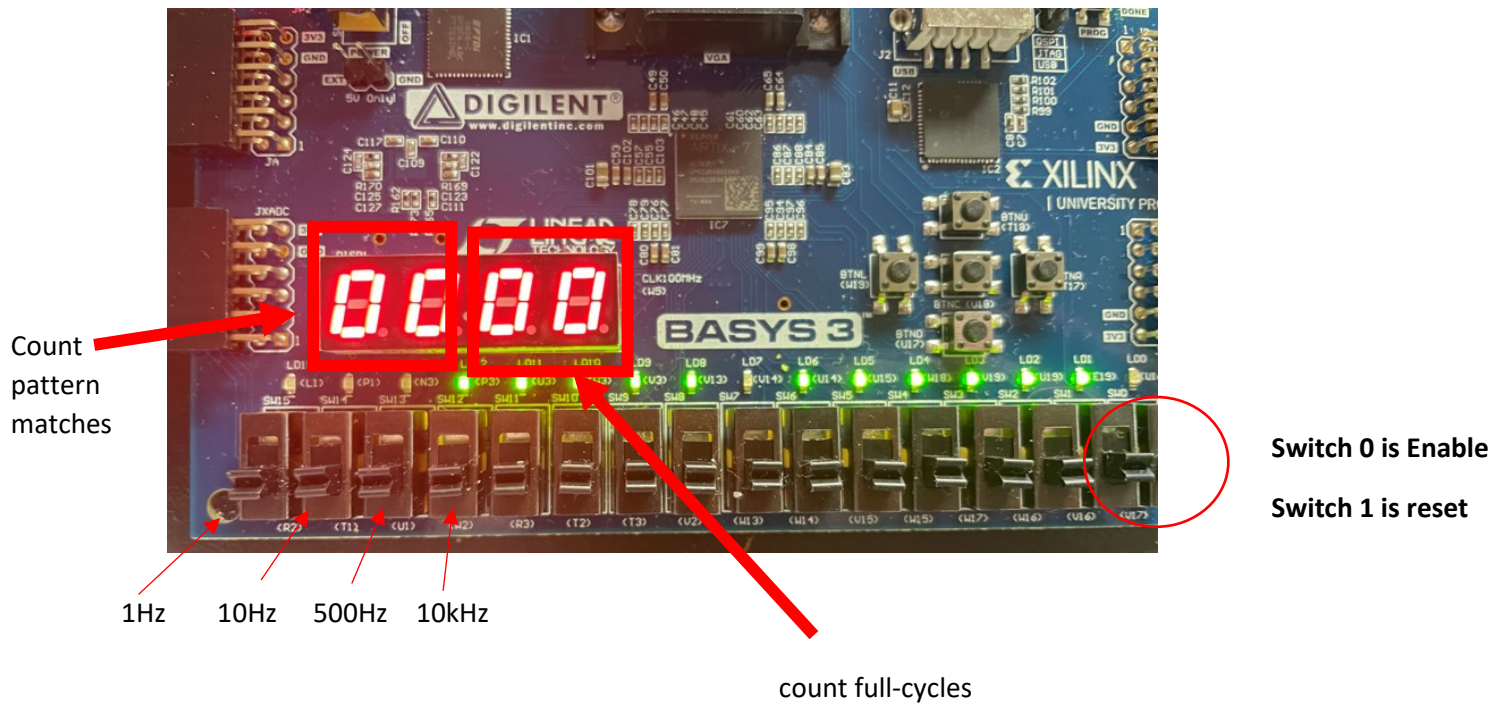
(if more than one switch is on, then the clock is a linear combination of all the ON switches)

Hierarchy



Demo

The board starts with the seed value (the 16 MSB are displayed).



LFSR at 500Hz clock speed

<https://youtu.be/W6151vL7fls>

this is a video showing the board going through one full cycle, with 32 pattern matches