3C7 Digital System Design

Assignment 2

Shane O'Donnell – 21364002

<u>Lab Description:</u>

The purpose of this session is to bring together elements from all your Labs, but particularly Lab F-G, and give you experience with developing and testing a larger sequential design and targeting it to the FPGA board.

The objective was to design a system that comprises an LFSR, a sequence detecting Finite State Machine (FSM), and a counter, in order to count the number of times a certain codeword (12 bits long) is issued in the stream of bits generated by the LFSR in a full cycle of that LFSR.

Method

In order to implement the necessary specifications for the objective of this laboratory, the LFSR module constructed for a past investigation was reused and altered to fit the specifications of this investigation. One of these alterations was changing the LFSR from a 22 bit to a 24 bit LFSR. This entailed changing the tapped bits that would be used to generate the next bit to be added in the zero position of the 24 bit binary number. The choice of tapped bits was chosen by confiding in a Xilinx data sheet for shift registers [1]. The chosen bits to be tapped can be seen in figure 1 below.

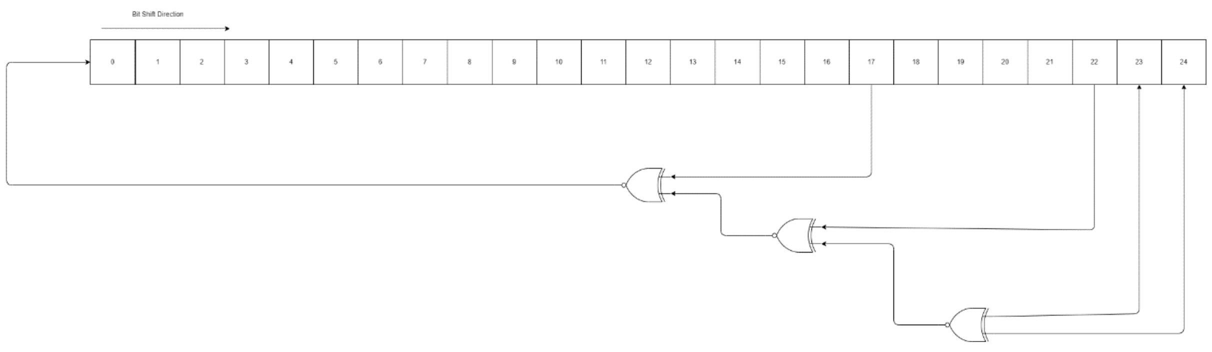


Figure 1: 24 bit LFSR with indication of tapped bits

Finite State Machine

In order to successfully detect the desired sequence a finite state machine needed to be designed and tailored to detect the desired sequence. In order to successfully implement this into Verilog a diagram was first constructed in order to understand how the finite state machine should be designed.

Essentially, a finite state machine is used to design logic circuits that have a finite number of states. The machine will transition states depending on the input signal it receives. Once the predefined number of state changes are complete an output signal is sent which will flag when the predefined state pattern has been satisfied.

In the context of this objective, the finite state machine will transition through each state depending on the input signal received, in this case, each bit which is taken from the LFSR and sent to the finite state machine.

In order to minimise the amount of state changes, the diagram seen in figure 2 displays each of the 12 states only having two possible output paths each. As seen in the diagram to ensure no sequences are missed, if the next bit in the desired sequence is a 0, if the LFSR input is a 1 the sequence is returned to S1. This is due to the fact that the first desired bit in the sequence is a 1 so this first bit has already been satisfied with the last input. If the next desired bit in the sequence is a 1 and the LFSR inputs a 0, the sequence is sent back to S0 as the desired sequence starts with a 1.

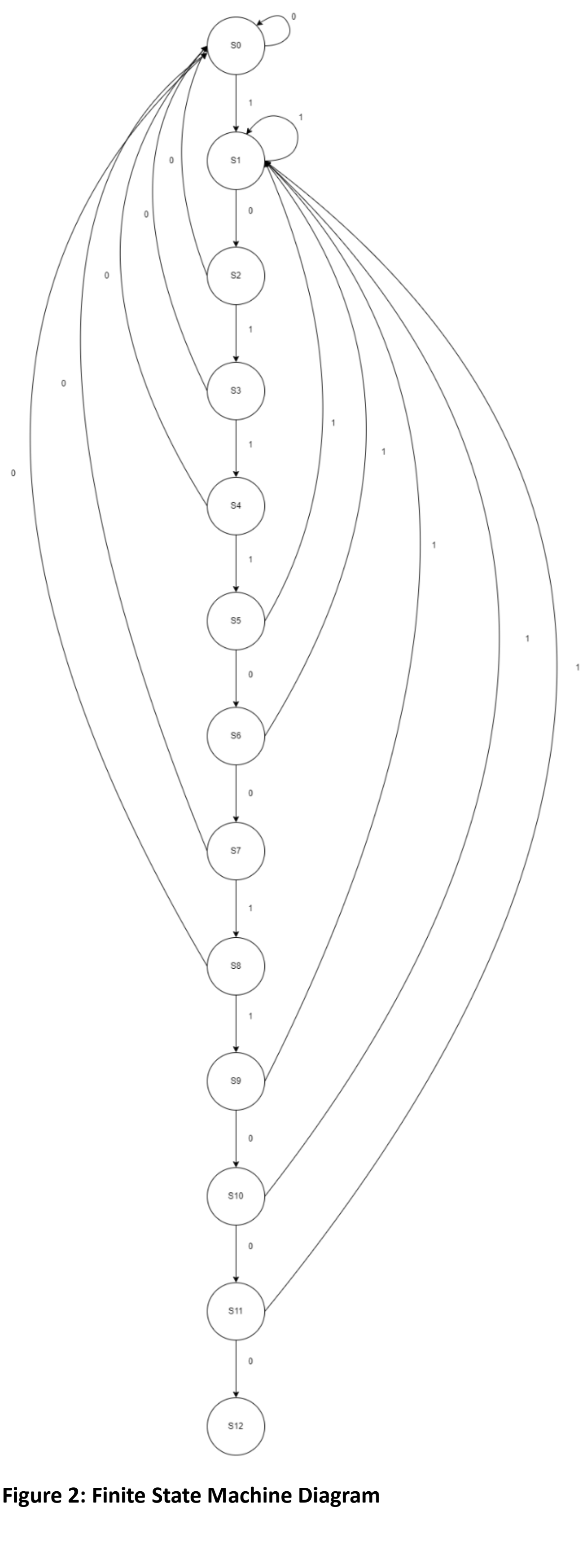
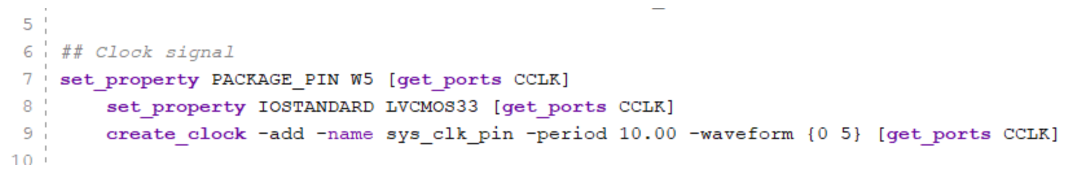


Figure 2: Finite State Machine Diagram

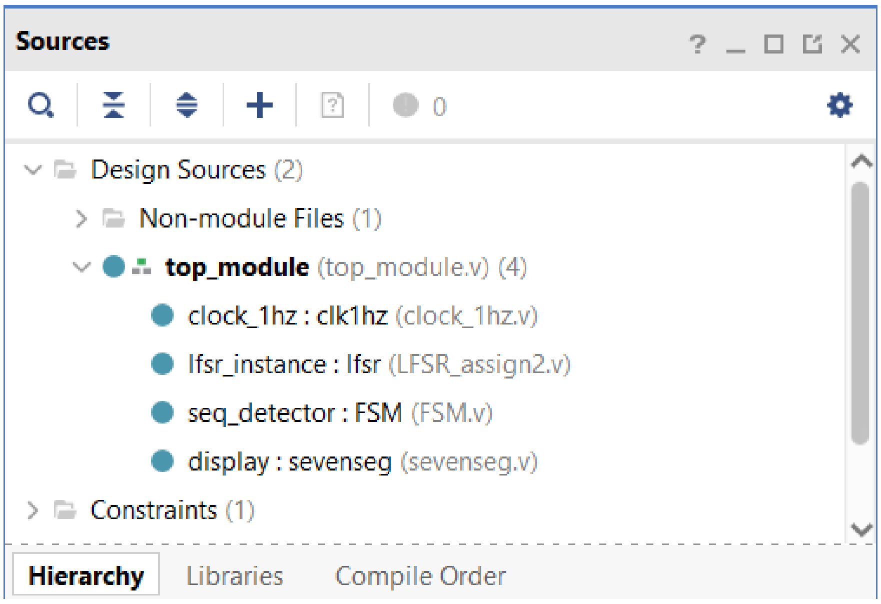
Vivado Implementation

In order to implement both the LFSR and finite state machine into Vivado, it was necessary to alter the LFSR module which was created for a previous investigation, as well as construct a module for a finite state machine that detects the desired sequence. As well as this in order to distinguish the LFSR sequences outputted to the Basys 3 LED's a 1Hz clock signal module was created, as well as a seven-segment display module which was reused from a previous laboratory. The clock module was implemented to the board in the .xdc file as seen in figure 3. These modules would work in sequence using a top module. The hierarchy of the system can be seen below in figure 3.



```
## Clock signal
set_property PACKAGE_PIN W5 [get_ports CCLK]
    set_property IOSTANDARD LVCMOS33 [get_ports CCLK]
    create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports CCLK]
```

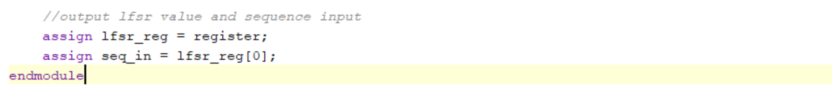
Figure 3: Clock implementation to Basys 3 Board



```
Sources
Design Sources (2)
    Non-module Files (1)
    top_module (top_module.v) (4)
        clock_1hz : clk1hz (clock_1hz.v)
        lfsr_instance : lfsr (LFSR_assign2.v)
        seq_detector : FSM (FSM.v)
        display : sevenseg (sevenseg.v)
Constraints (1)
```

Figure 4: Module Hierarchy

In order to ensure the LFSR and finite state machine would work together, the generated bit stored in the zero position of the LFSR was passed into the sequence detector each LFSR cycle. Using the diagram created previously it was possible to design the finite state machine module to detect the desired sequence. In the figure below it can be seen how the current LFSR value and the sequence input are outputted from the LFSR module to the test bench.



```
    //output lfsr value and sequence input
    assign lfsr_reg = register;
    assign seq_in = lfsr_reg[0];
endmodule
```

Figure 5: LFSR module assignment

Once the design was completed and implemented the implemented design can be seen as follows.

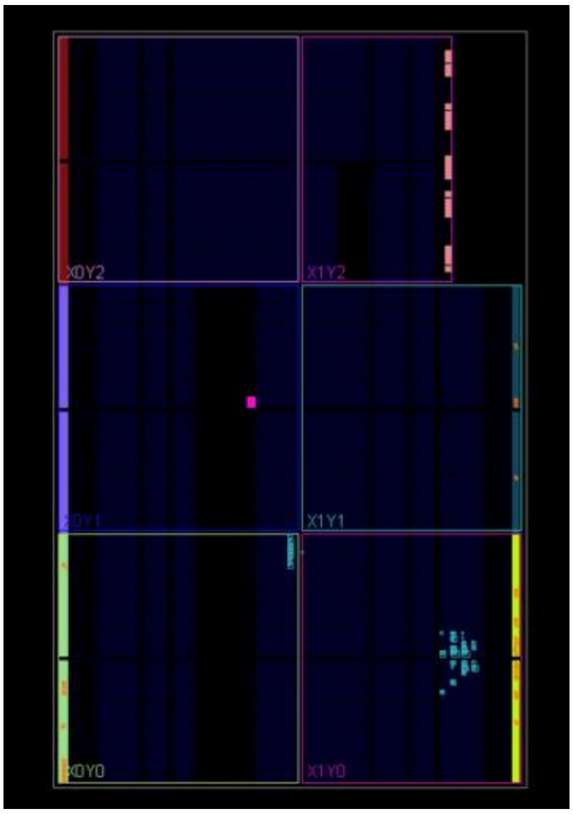


Figure 6: Implemented Device

As well as this, the RTL schematics displays an overview of how the different modules interact simultaneously in order for the overall project to work as needed. This schematic includes how the results are displayed on the basys 3 board as well as the clock module which slows down the clock signal from a 100Mhz cycle to a 100Hz cycle. As well as this the implemented schematic is provided below.

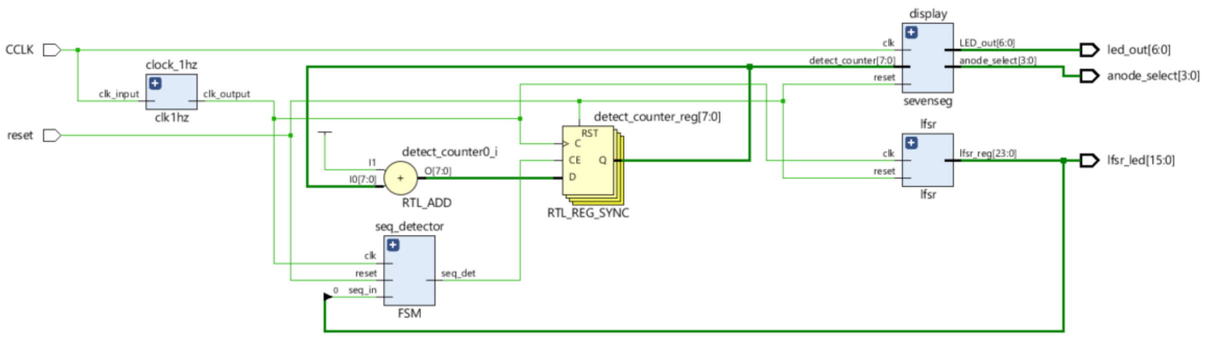


Figure 7: RTL Schematic

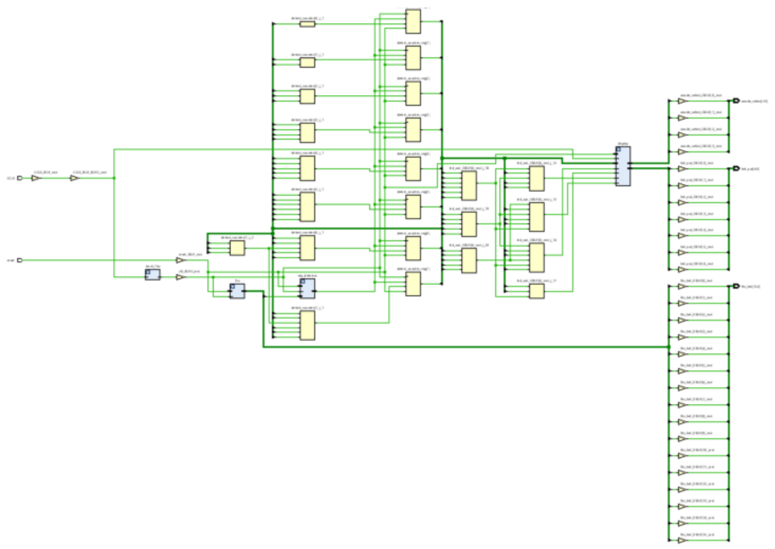
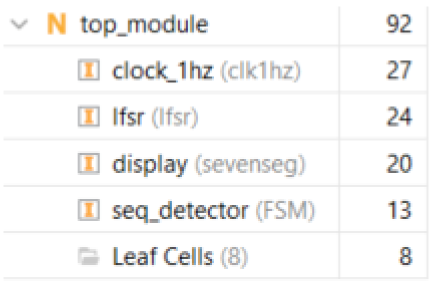


Figure 8: Implemented Schematic

Furthermore, the utilisation report provided by Vivado displayed the register as flip flop count for each module. This can be seen below.



top_module	92
clock_1hz (clk1hz)	27
lfsr (lfsr)	24
display (sevenseg)	20
seq_detector (FSM)	13
Leaf Cells (8)	8

Figure 9: Register as Flip Flop

Project Simulation

In order to ensure the design was operating as expected, a relevant simulation was run before assigning the designing to the Basys 3 board. The figure below displays what happens when a sequence is detected. It can be seen that once the sequence is outputted by the LFSR, the sequence detection is flagged once a rising edge clock signal is outputted. Once this detection flag is returned to zero the sequence counter is incremented by one to count the total detections in one LFSR cycle.

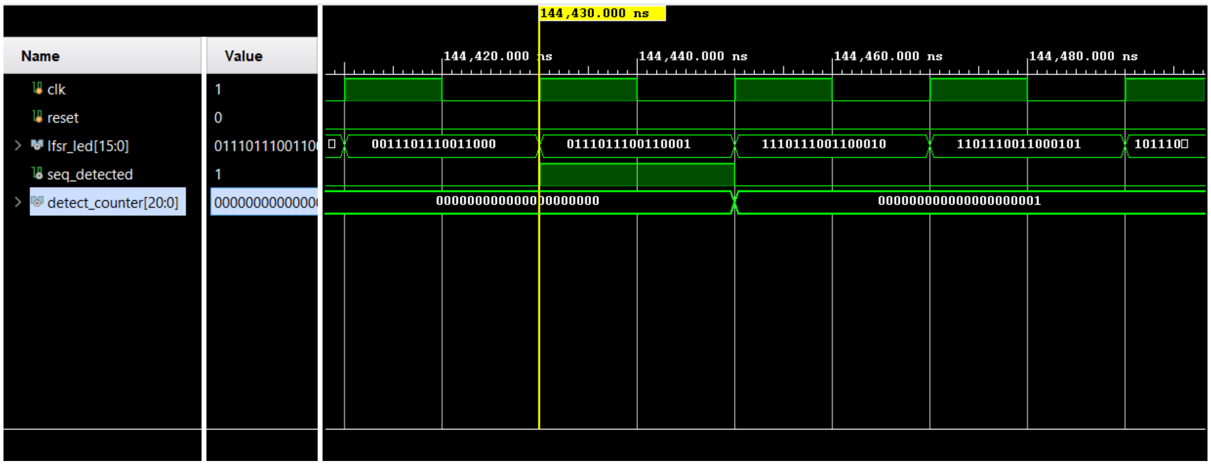


Figure 10: Sequence detection

Once a full LFSR cycle was complete, it was possible to document the final total sequence amount, this can be seen in the figure below. The final sequence detection count can be seen to be 2836 detections.

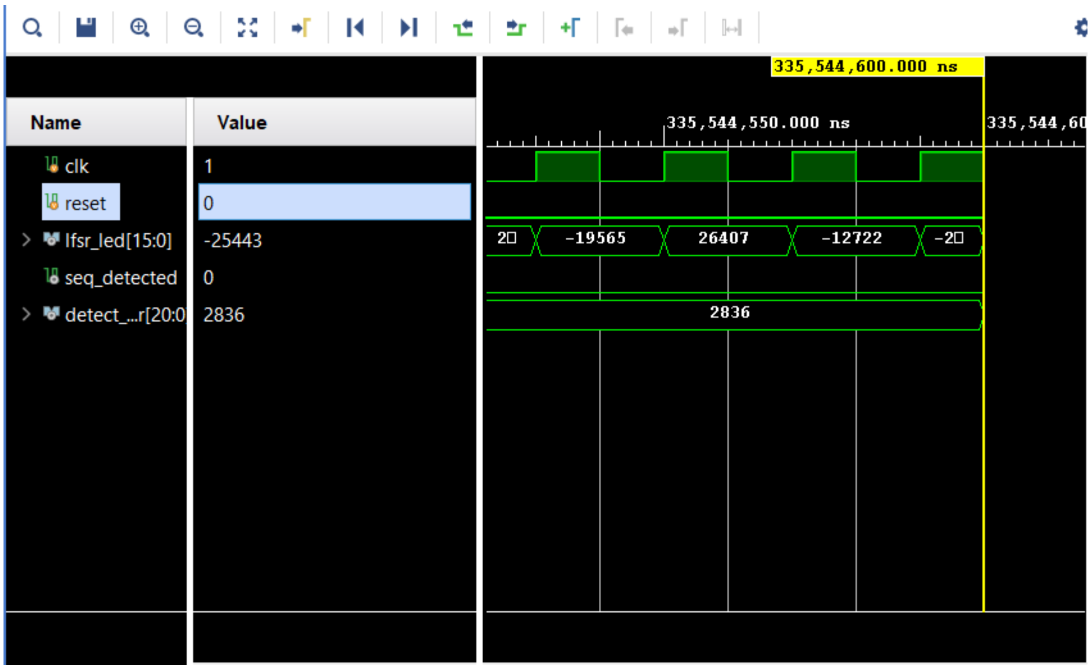
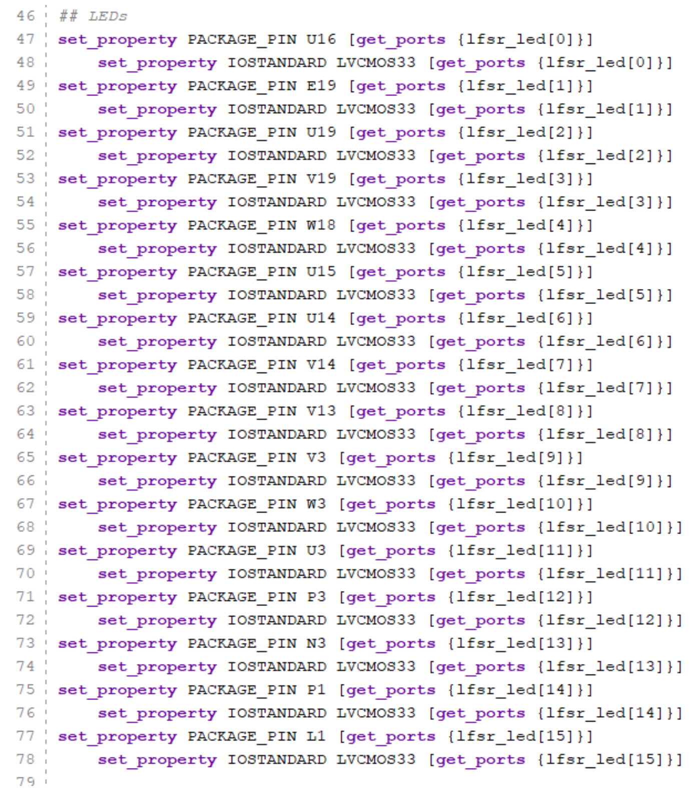


Figure 11: Final sequence detection count

Demo

Now that the system has been verified to function correctly, it was now possible to assign the system to the board. For this project the Basys 3's onboard LED's were used to output the LFSR number while the seven segment display was used to output the current pattern counter. The LFSR value was assigned to the onboard LED's using the xdc file as follows.



```
## LEDs
set_property PACKAGE_PIN U16 [get_ports {lfsr_led[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {lfsr_led[0]}]
set_property PACKAGE_PIN E19 [get_ports {lfsr_led[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {lfsr_led[1]}]
set_property PACKAGE_PIN U19 [get_ports {lfsr_led[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {lfsr_led[2]}]
set_property PACKAGE_PIN V19 [get_ports {lfsr_led[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {lfsr_led[3]}]
set_property PACKAGE_PIN W18 [get_ports {lfsr_led[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {lfsr_led[4]}]
set_property PACKAGE_PIN U15 [get_ports {lfsr_led[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {lfsr_led[5]}]
set_property PACKAGE_PIN U14 [get_ports {lfsr_led[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {lfsr_led[6]}]
set_property PACKAGE_PIN V14 [get_ports {lfsr_led[7]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {lfsr_led[7]}]
set_property PACKAGE_PIN V13 [get_ports {lfsr_led[8]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {lfsr_led[8]}]
set_property PACKAGE_PIN V3 [get_ports {lfsr_led[9]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {lfsr_led[9]}]
set_property PACKAGE_PIN W3 [get_ports {lfsr_led[10]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {lfsr_led[10]}]
set_property PACKAGE_PIN U3 [get_ports {lfsr_led[11]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {lfsr_led[11]}]
set_property PACKAGE_PIN P3 [get_ports {lfsr_led[12]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {lfsr_led[12]}]
set_property PACKAGE_PIN N3 [get_ports {lfsr_led[13]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {lfsr_led[13]}]
set_property PACKAGE_PIN P1 [get_ports {lfsr_led[14]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {lfsr_led[14]}]
set_property PACKAGE_PIN L1 [get_ports {lfsr_led[15]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {lfsr_led[15]}]
```

Figure 12: LED assignment

This can be seen in the following figure which displays the board when the counter is at 1.

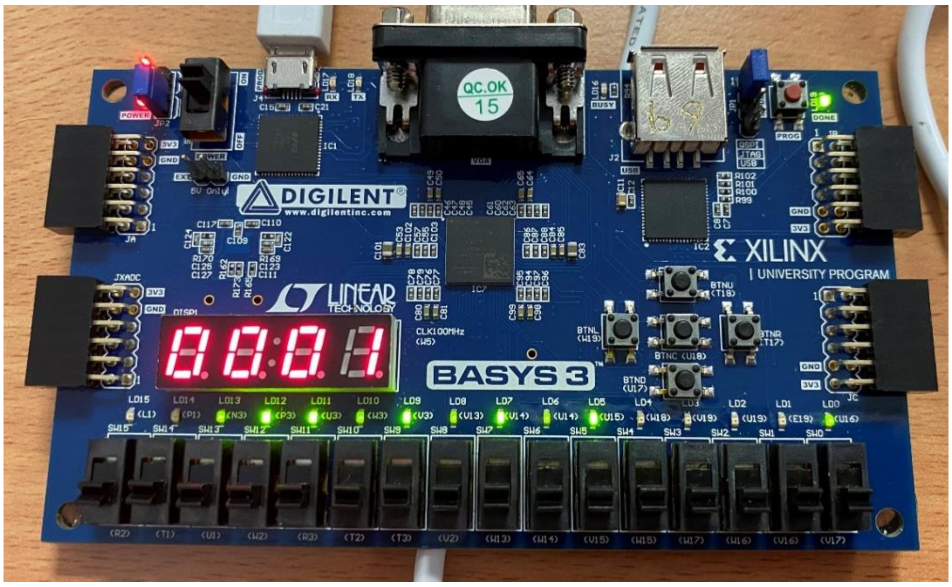


Figure 13: Project assigned to board – counter outputting 1

Due to the clock signal being slowed down, it was easier to monitor each lfsr pattern, although this did slow down the amount of patterns being detected as the LFSR was generating new patterns at a slower rate. The following figure shows the basys 3 board after letting the system run for a significant amount of time.

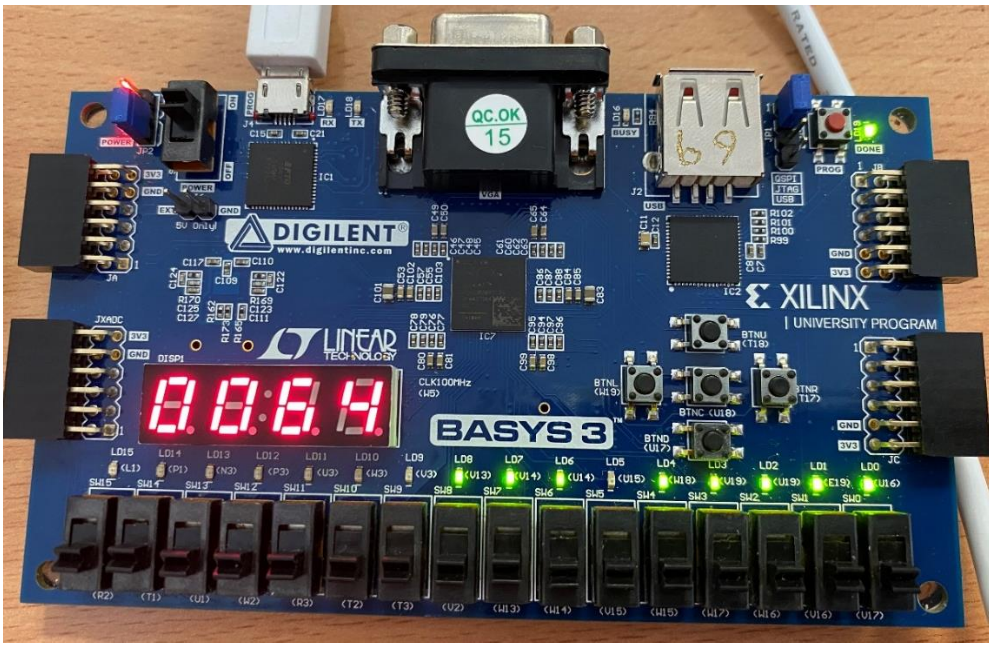


Figure 14: Basys 3 board with a detection counter of 64

Conclusion

To conclude, the 24 bit LFSR and finite state machine were successfully implemented onto the board. Connecting these modules together using a top module made it possible to successfully detect and count the number of sequences that occurred during one LFSR cycle.

Appendix

The relevant Verilog files are provided in this submission and uploaded accordingly.

Sources

[1] https://docs.amd.com/v/u/en-US/xapp052