**CPE 133: Digital Design**

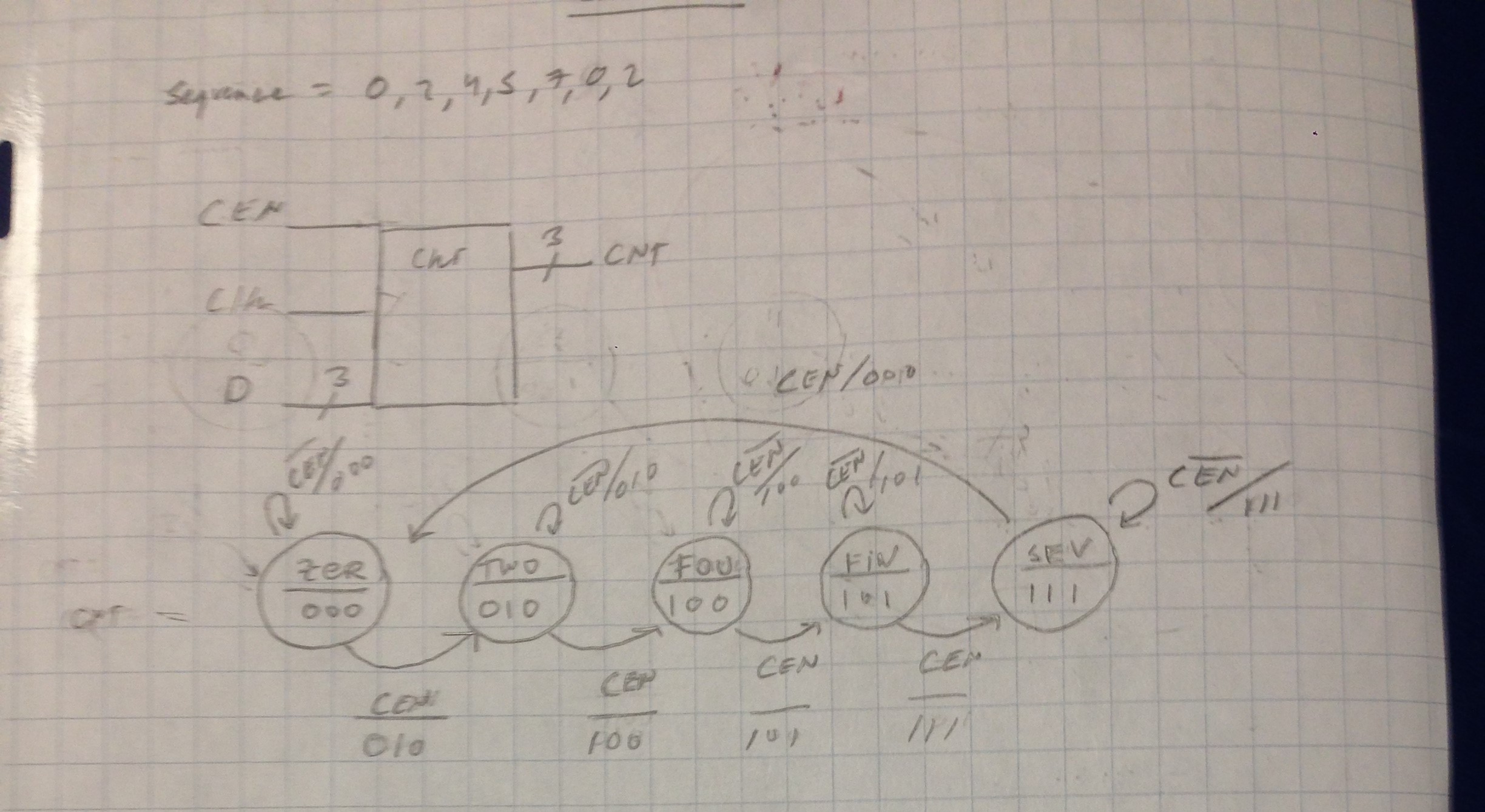
Prof John Callenes-Sloan

Luis Gomez & Richard Hua

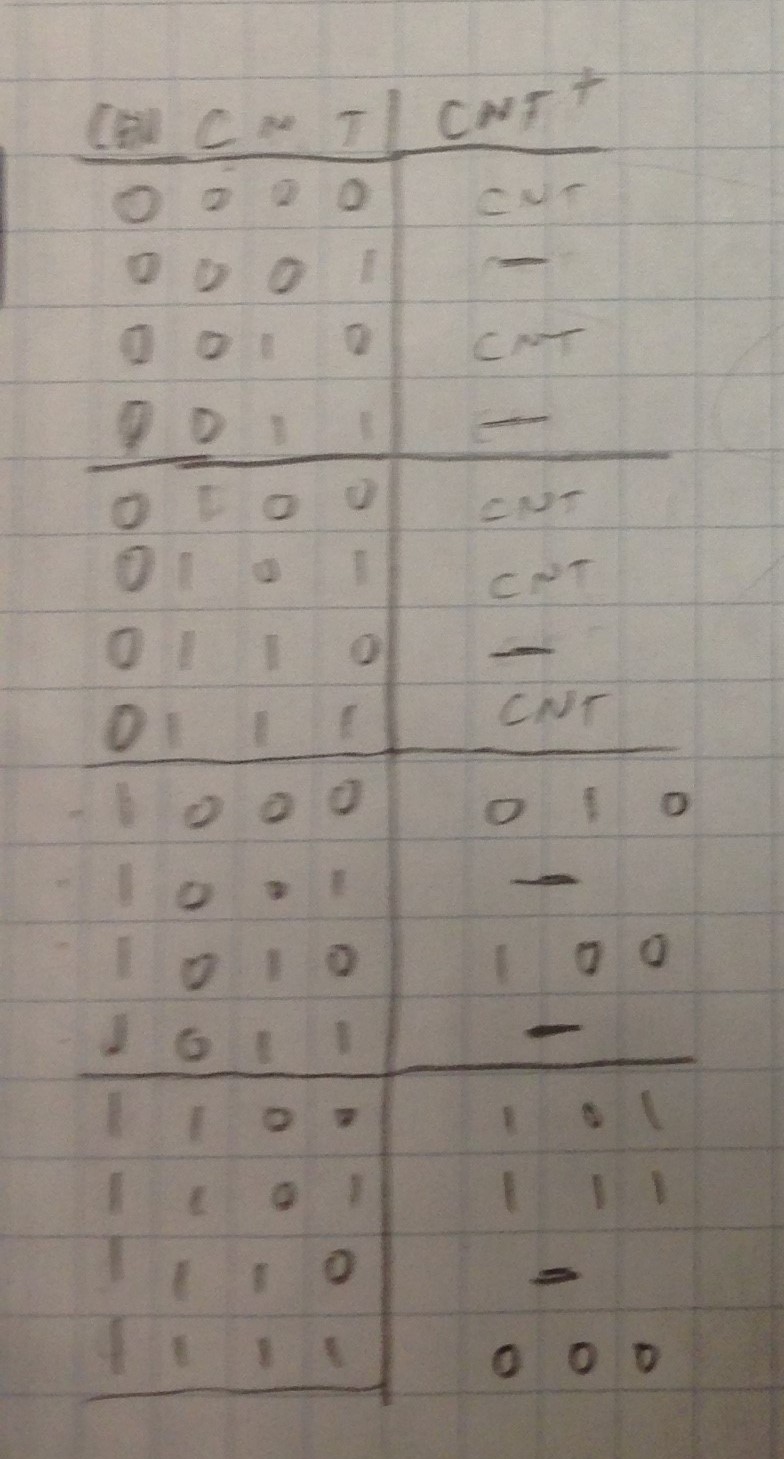
**Experiment 13: 3 Bit FSM Counter**

**Summary:**

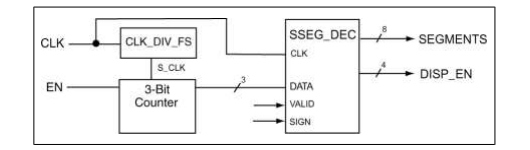
In exp 13 we successfully created a 3-bit counter for an arbitrary sequence 0,2,4,5,7... We created our own clock module to be able to display the sequence at a reasonable rate via the 7-Segment display module.



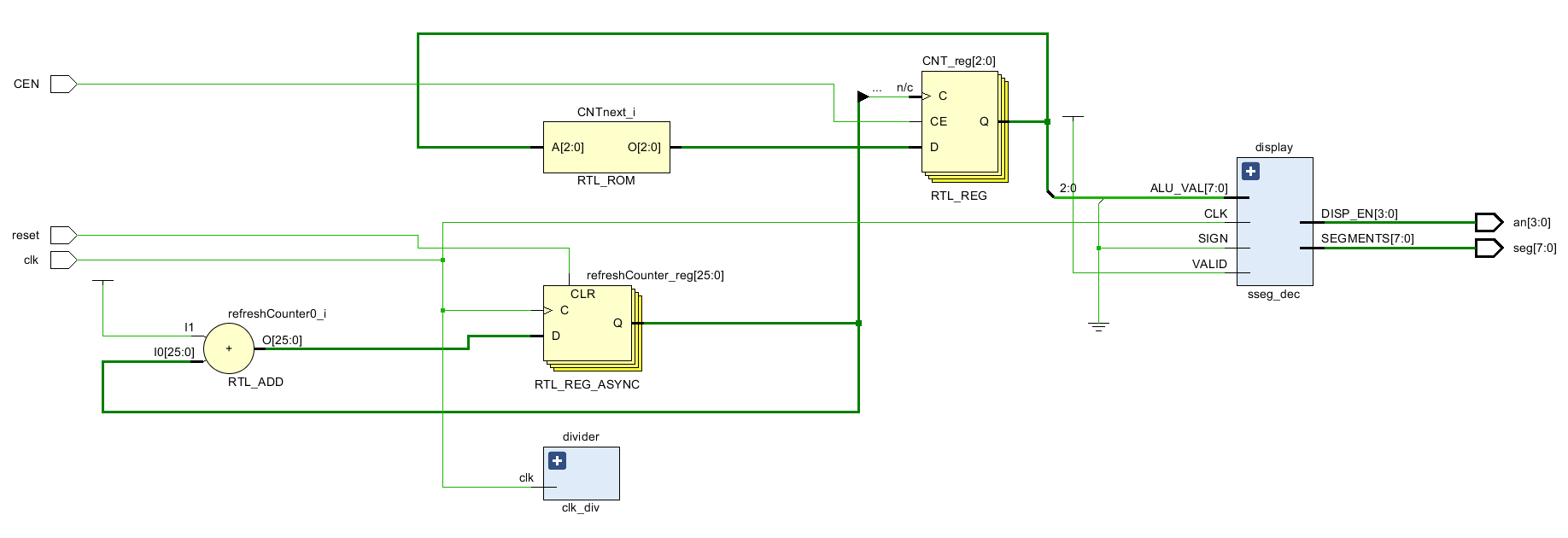
***Figure 1: Moore State Diagram of 3-Bit FSM Counter***



***Figure 2: Excitation Table for 3-Bit FSM Counter***

****

***Figure 3: BBD for 3-Bit FSM Counter***

****

***Figure 4: Elaborated Circuit Design for 3-Bit FSM Counter***

**Verification:** In the video demo provided below, we demonstrate the function of our 3-bit counter. We configured our clock-module to provide a reasonable rate of edge-triggers to toggle the sequence.

[**https://youtu.be/sUVPceepQOk**](https://youtu.be/sUVPceepQOk)

**Questions:**  
1. How many flip-flops did this FSM design use? Recall that even if you did not explicitly create a flip flop, the synthesizer would still create flip flops for you based on your behavioral model.

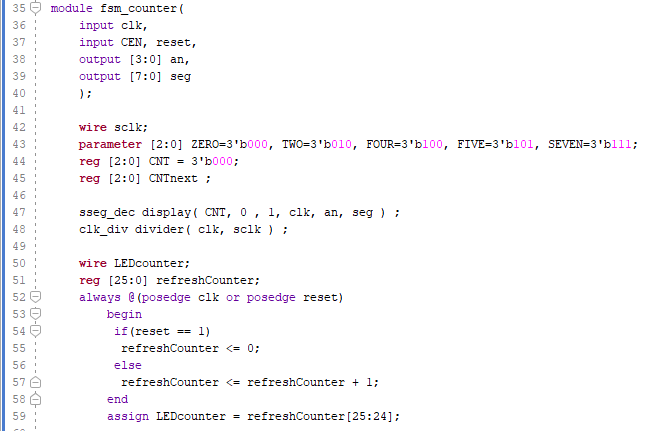
The FSM design used 3 flip-flops to represent the 5 different states.

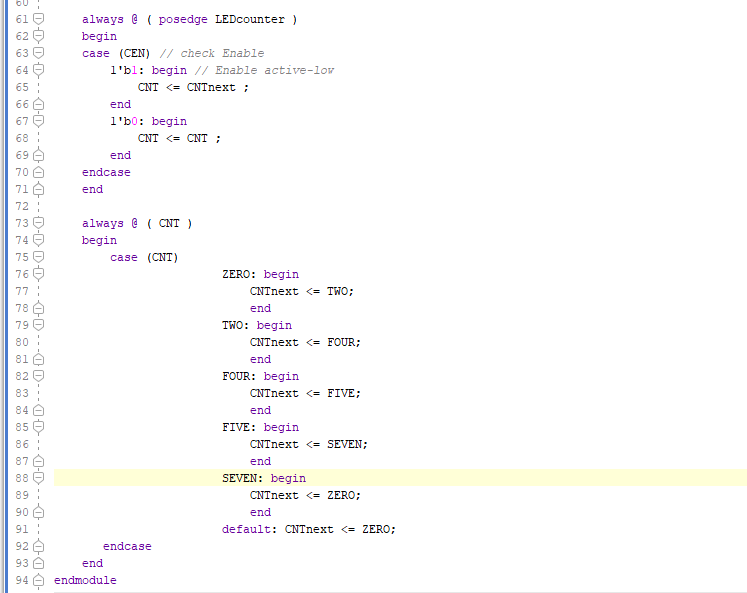
2. How hard would it to add four more numbers into the counting sequence using the  
Verilog behavioral approach to FSM design?

This would not be very hard since all we have to do is add four more cases to represent the numbers that we want to add into the sequence.

3. What would happen if this FSM were to suddenly find itself displaying the count of ‘3’?

The FSM would change to zero on the next rising edge, effectively resetting itself, so that it can properly display the next correct number in the sequence.

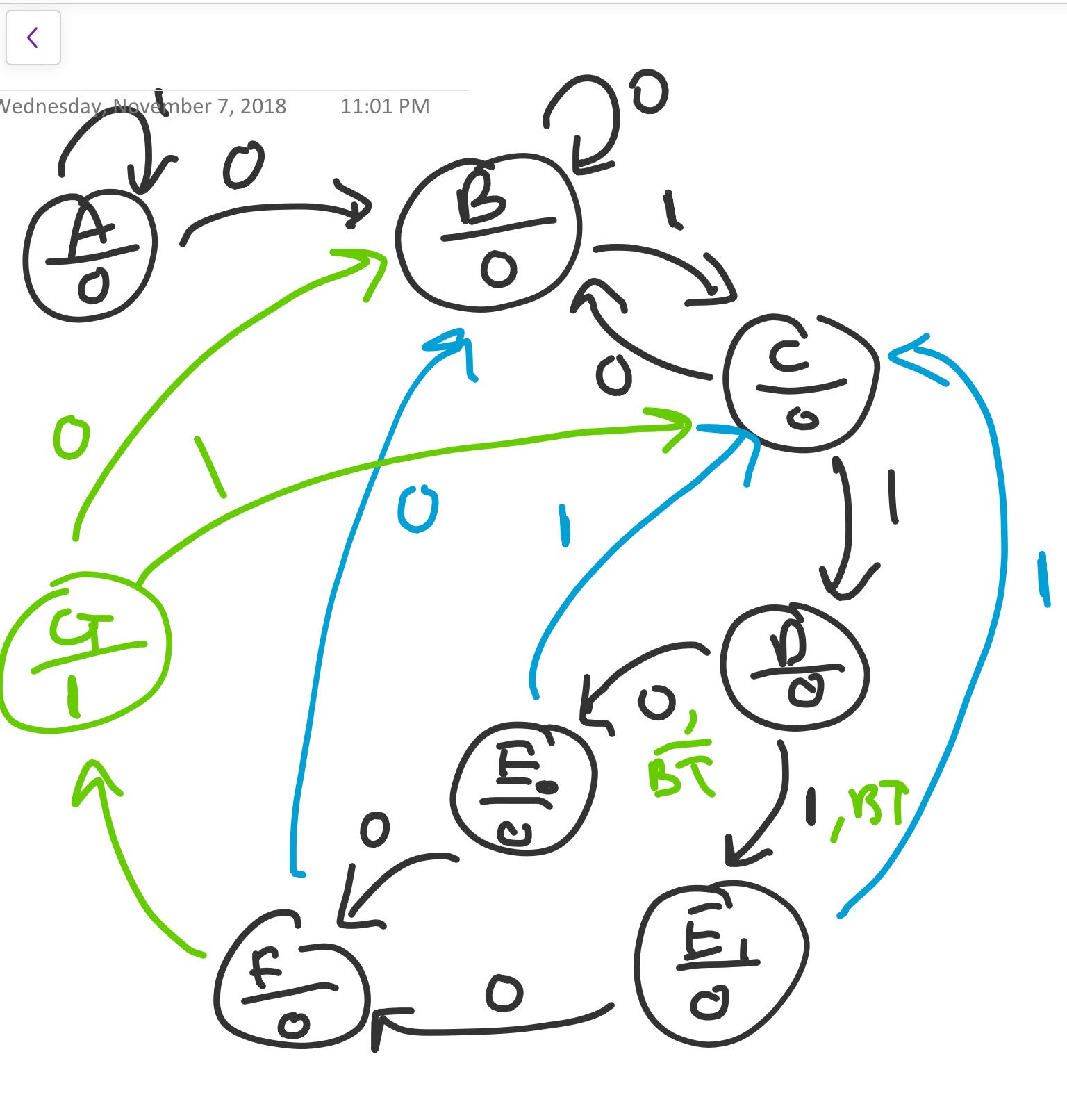
**Code:**

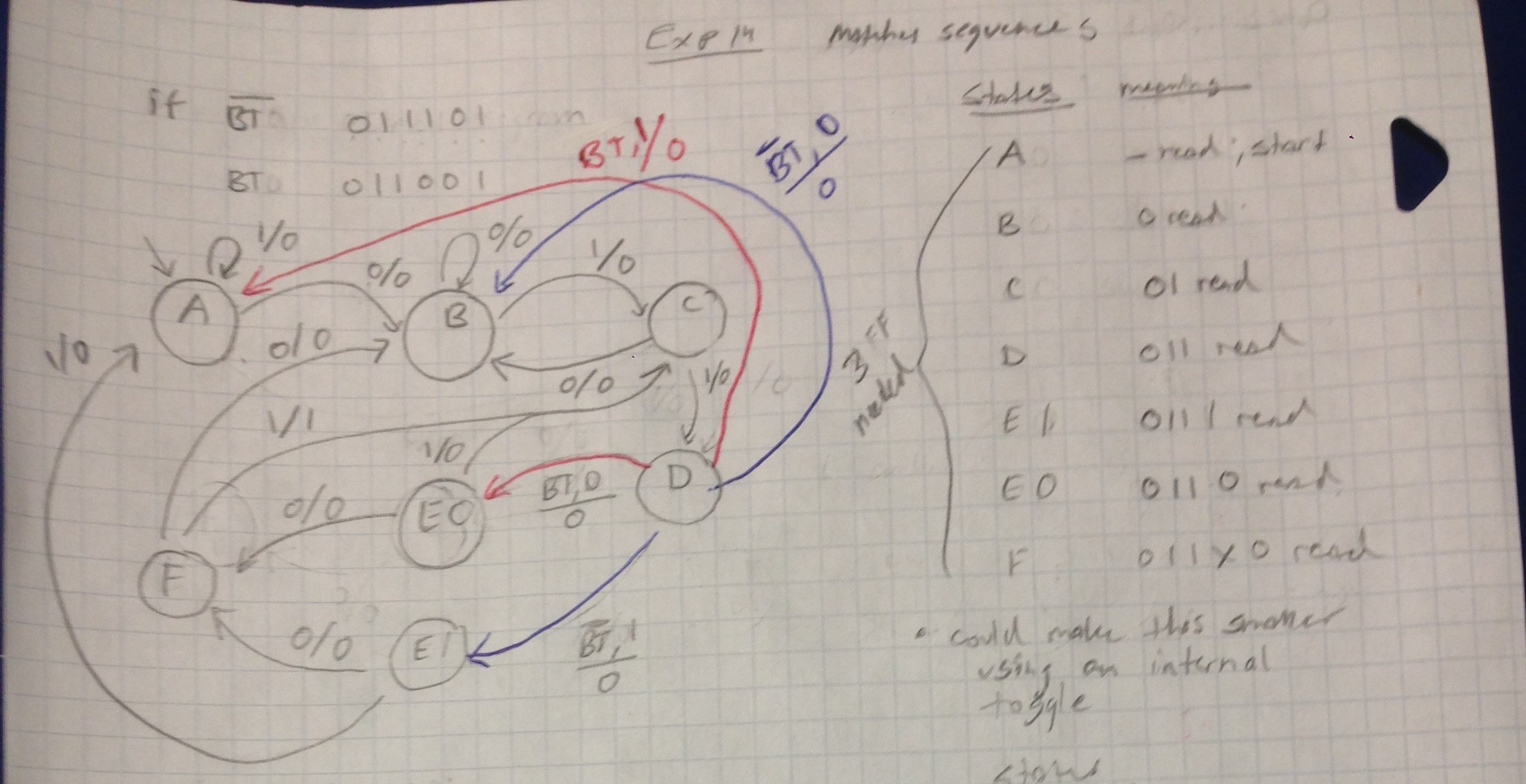


**Experiment 14: Sequence Detector FSM**

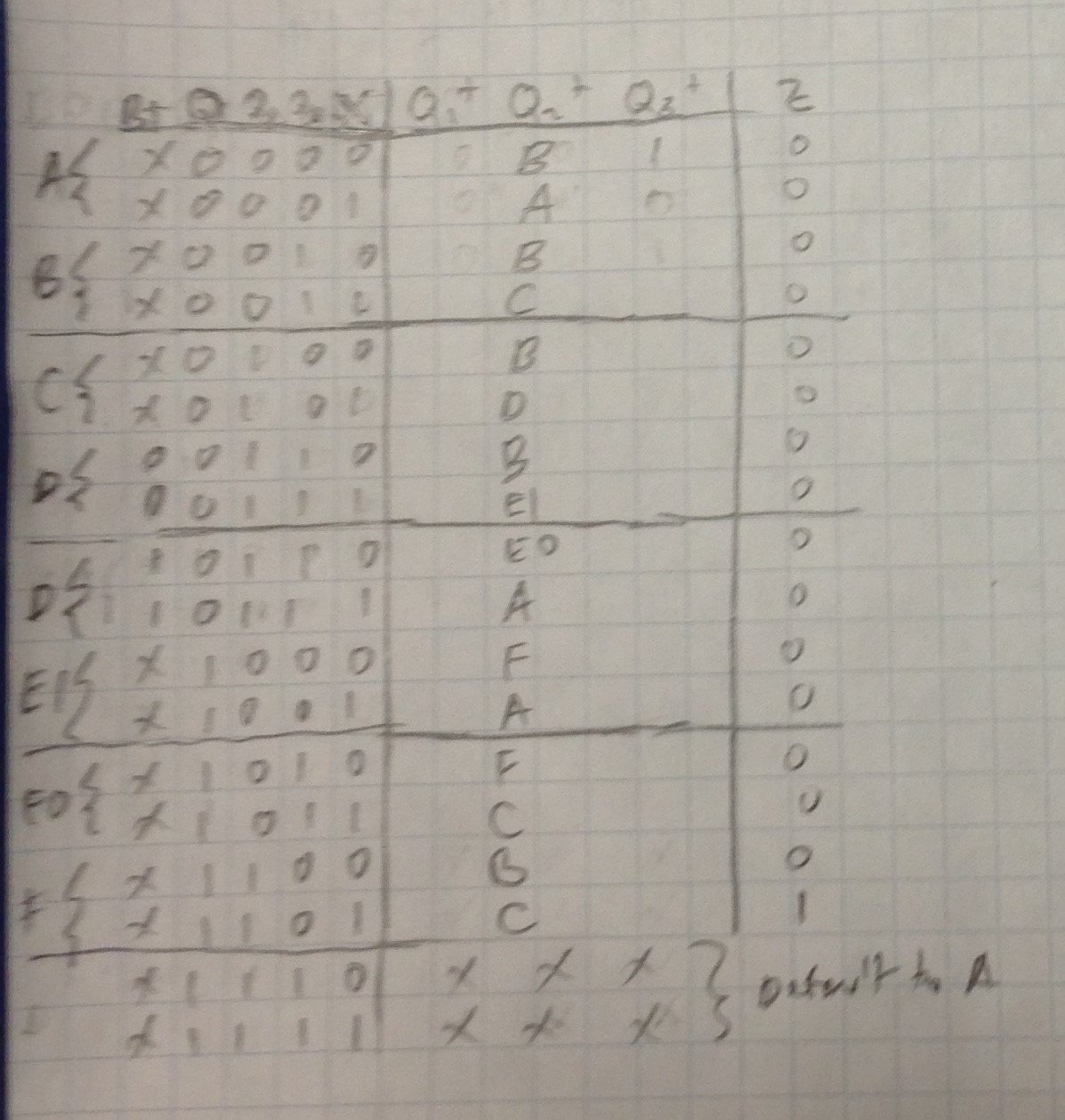
**Summary:**

In exp 14 we attempted to create a Sequence Detector FSM utilizing the provided “SEQ\_DVR” and “BC\_DEC” modules. The sequence detector has the following inputs: a Clock signal ,push-Button , and a 8-bit switch vector. The Sequence Detector has 12 bits of output via the 7-segment display module. The push-button input is a form of external control that determines which of the two different sequences is to be detected.

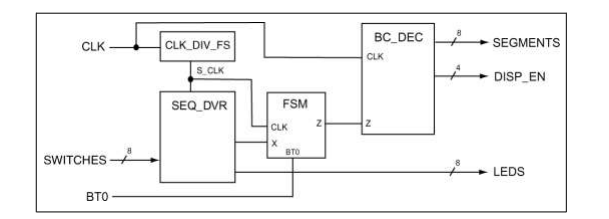


****

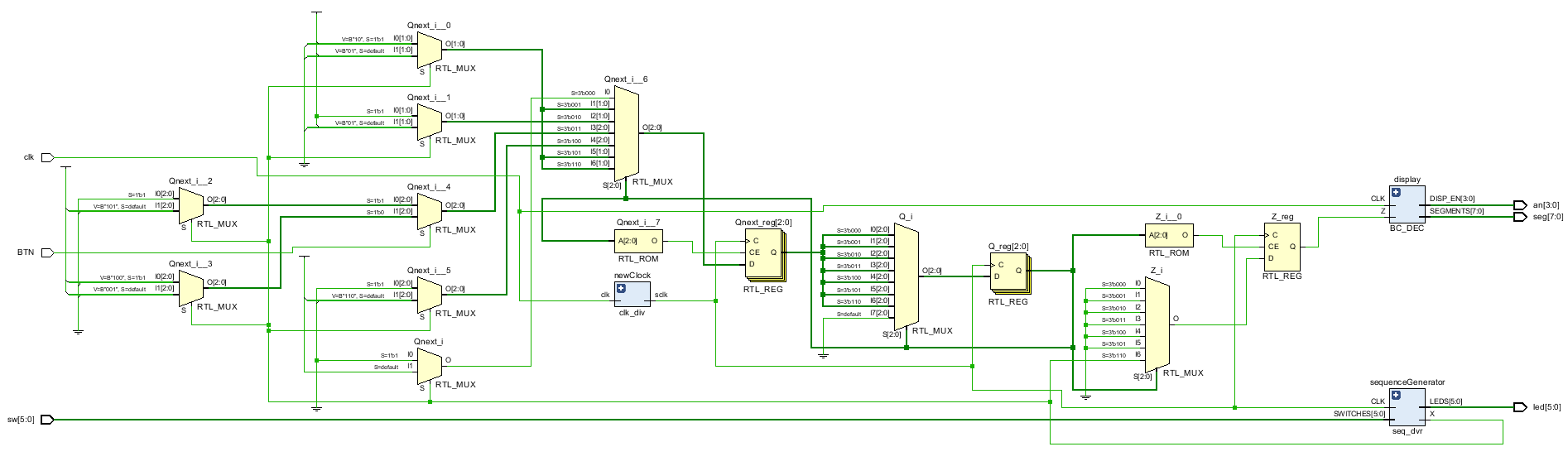
***Figure 5: Mealy & Moore State Diagram of Sequence Detector & State Table***

****

***Figure 6: Next State Table for Sequence Detector, note that we only care about the Button toggle during the D-state***

****

***Figure 7: BBD of Sequence Detector***



***Figure 8: Elaborated Circuit Design for Sequence Detector***

**Verification:**  We provided a video demo below of our attempt at creating the Sequence Detector. We believe that our state-diagram and next-state table were correct but that our implementation in Vivado requires debugging and review. We were able to detect one of the two sequences provided but our display did not display COOL but rather C8AP and Cr8P, but we haven’t discerned why.

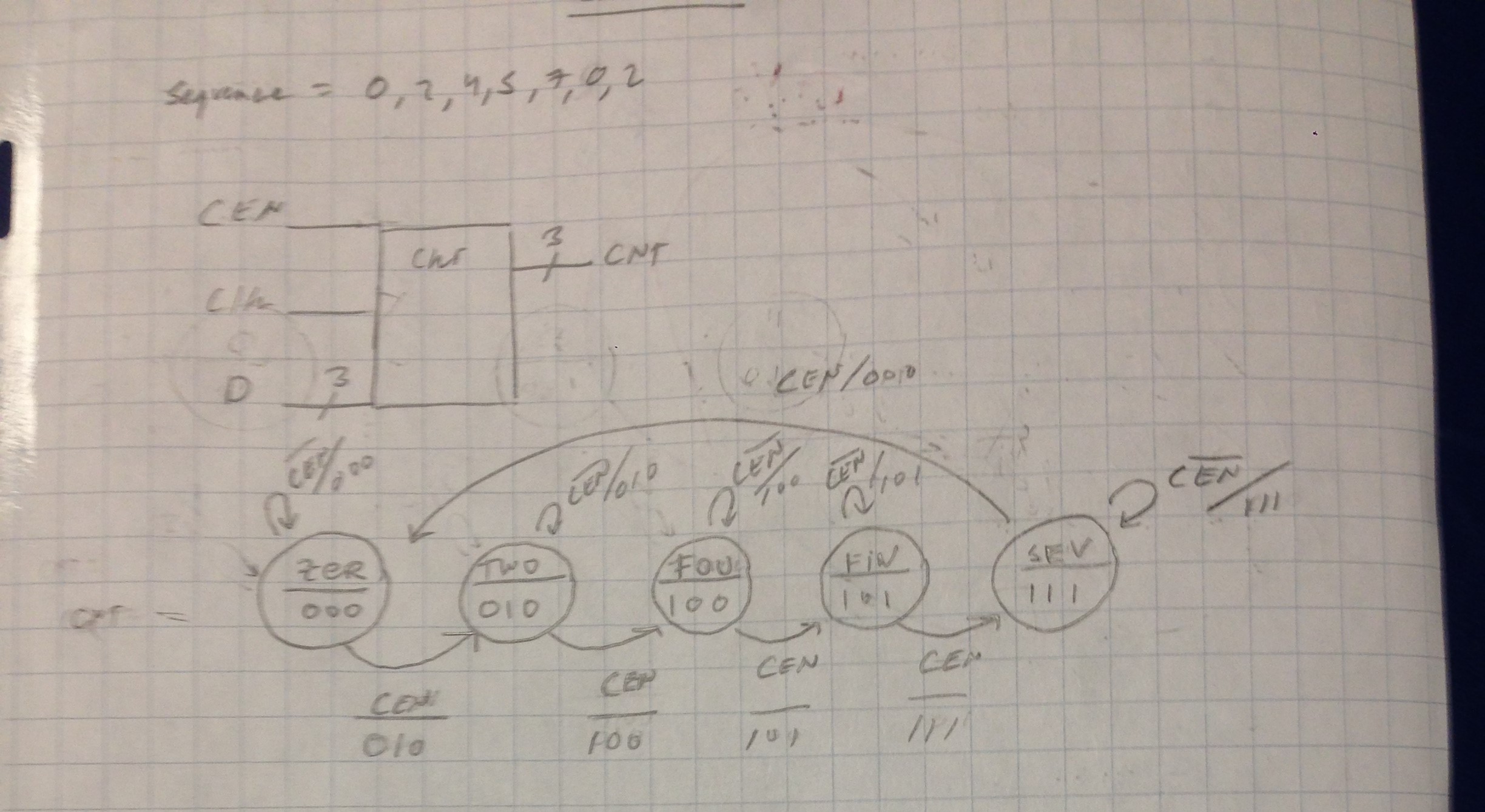
[**https://youtu.be/4I3TGv38HjY**](https://youtu.be/4I3TGv38HjY)

**Questions:**

1. Describe at least two applications where sequence detectors could potentially be useful.

Sequence detectors could be used to decode packets of transmitted data or for very basic decryption, by mapping sequences of binary values to messages or letters of the alphabet. With a parity checker, a sequence detector could possibly do error correction/detection given a sufficiently capable hamm-code.

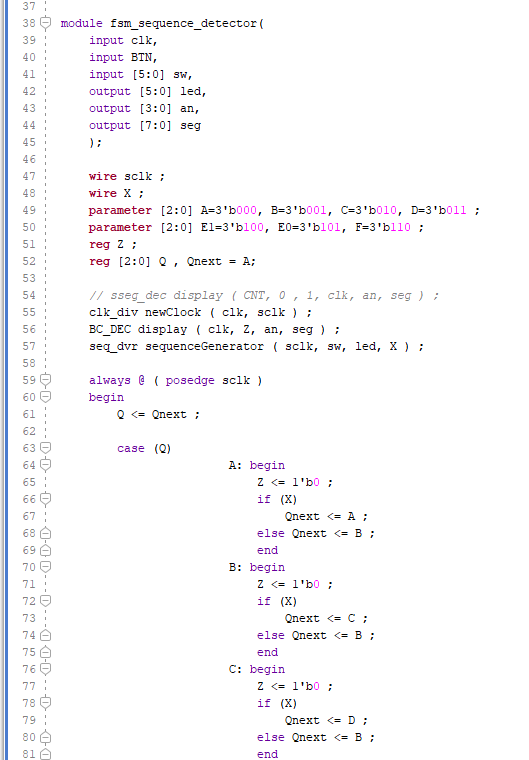
2. Provide Mealy-type state diagrams for the resetting version of the FSM you designed in this lab activity.

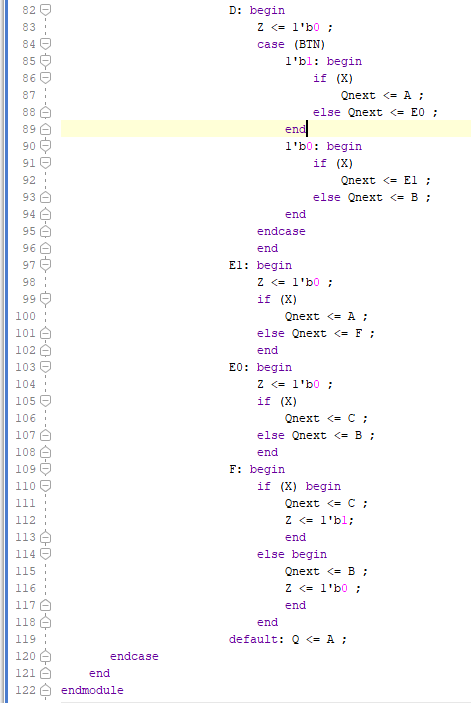


3. Briefly explain the function of the SEQ\_DVR box in Figure 22. You’ll need to take a look at the Verilog model for this question.

SEQ\_DVR at a high level is a 3-bit counter module paired with an encoder. With each edge-trigger of the clock the module increments a 3-bit value and uses this value to call the index of 1 of 8 switch inputs. The reference element in the 8b-switch vector is then assigned to the output value X. Over the course of many cycles, the module will output the state of the switch inputs in order ,which we interpret as a sequence.

**Code:**

****

****