ECE 2140 – 103

Colby Alley, Andrew Marin, and Jacob Hendrix

ckalley42@tntech.edu, amarin42@tntech.edu, and jthendrix42@tntech.edu

Project Report Part 1 – Sequence Generator

Date: 11/25/25

Due Date: 11/30/25

Lab Instructor: Marim Mahmoud

Part 1 Project Report – Sequence Generator

In part 1 of our project, we are designing a sequence generator using two D flip flops to generate two different sequences. We will have an input, S (selector), and an output, O (sequence out), that will provide the sequence depending on S and the current state of the machine. When S is one, the sequence should generate "1110," from output O. When S is zero, the sequence should generate "0001," from output O. S will "select" the sequence to be generated. First, we will design the state diagram.

Since we have four bits, we will use two D flip flops. In our design, we will be using the chip 74HC74 dual flip flop integrated chip. Here is figure one that displays the state diagram of our sequence generator.

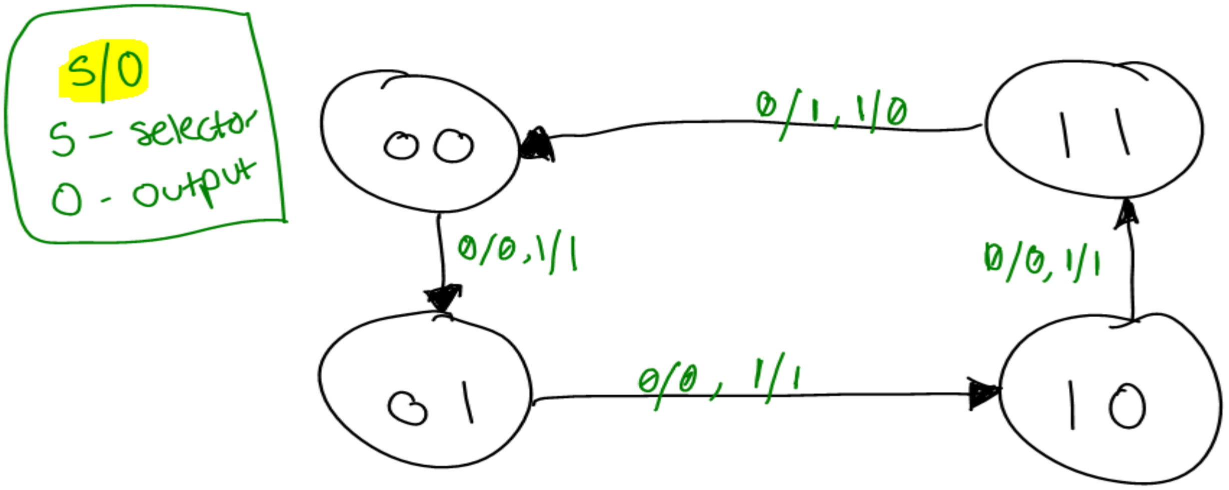
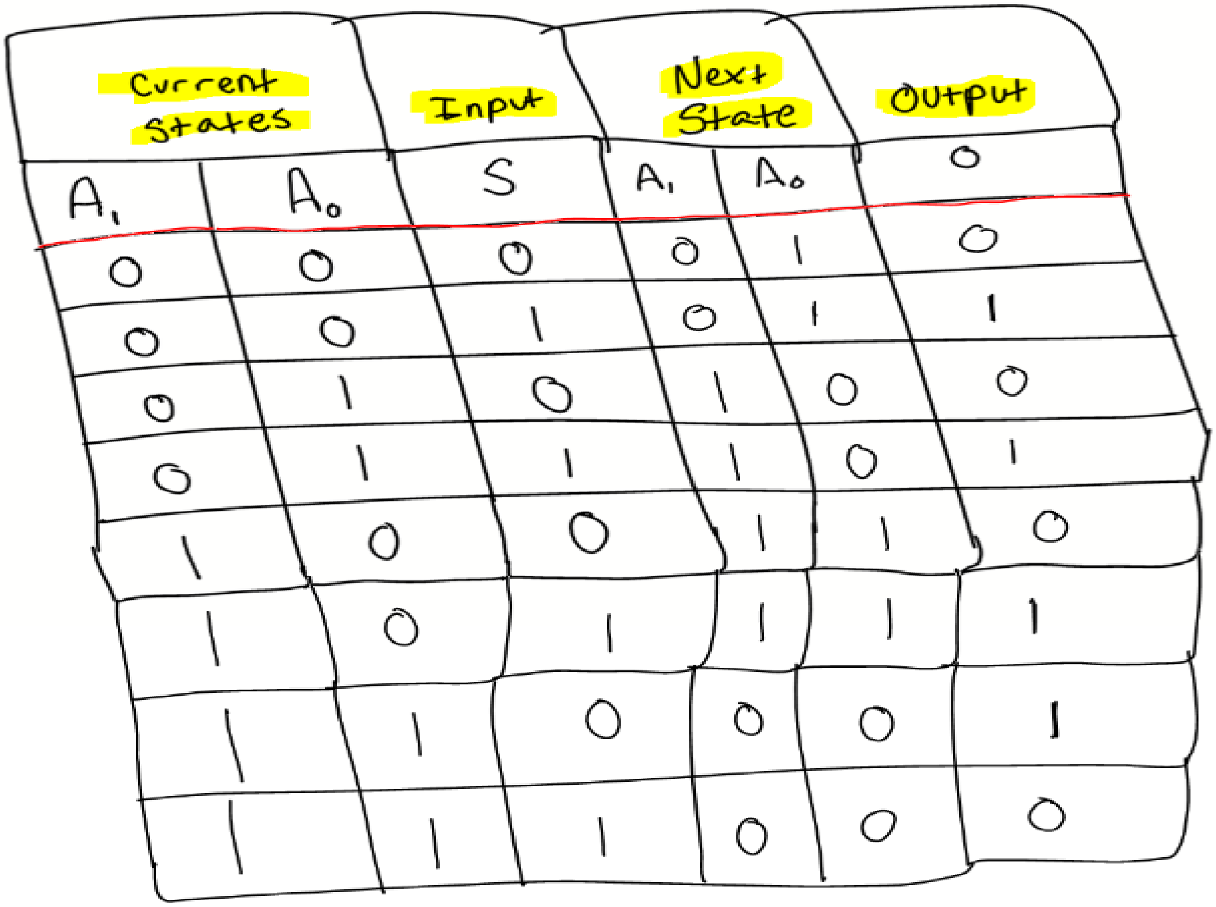


Figure 1: State diagram

In this diagram, each state alongside its input and output are displayed. When we are in state 00 and our selector equals zero, the output is zero. This cycle continues until the last state is reached, 11, and it outputs a one instead of a zero. This is the sequence we are trying to generate.

This will generate 0001 when our selector is zero. Our next sequence, 1110, will be our output when the selector is one. When we are in state 00 and our selector equals one, the output is one. This cycle continues until the last state is reached, 11, and it outputs a zero instead of a one. This is our second sequence generated, 1110. We have successfully created a state diagram to generate our sequences. It is now time to build the state table. Attached below in figure two is our state table determined by the state diagram. We will need to use k-maps to build functions based off the input and current states. This makes our circuit a Mealy design.



Current States		Input	Next State		Output
A_1	A_0	S	A_1	A_0	0
0	0	0	0	1	0
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	1	0	1
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	0	0	1
1	1	1	0	0	0

Figure 2: State Table

Next, using k-maps, we can find a relationship between our current states and selector input to our output sequence. Attached below will show our work in that process:

Finding next A_0 in terms of S, current A_0, current A_1:

A_1 \ A_0S	00	01	11	10
0	1 m0	1 m1	0 m3	0 m2
1	1 m4	1 m5	0 m7	0 m6

$$\boxed{A_0(t+1) = A_0'}$$

Finding next A_1 in terms of S, current A_0, current A_1:

A_1 \ A_0S	00	01	11	10
0	0 m0	0 m1	1 m3	1 m2
1	1 m4	1 m5	0 m7	0 m6

$$\boxed{\begin{aligned} A_1(t+1) &= A_1A_0' + A_1'A_0 \\ A_1(t+1) &= A_1 \oplus A_0 \end{aligned}}$$

Finding output in terms of S, current A_0, current A_1:

A_1 \ A_0S	00	01	11	10
0	m_0 0	m_1 1	m_3 1	m_2 0
1	m_4 0	m_5 1	m_7 0	m_6 1

$$O(t) = A_0'S + A_1'S + A_1A_0S'$$

We can deconstruct this equation as:

$$\boxed{O(t) = (A_0A_1) \oplus S}$$

Proof:

$$= (A_0 A_1)' S + S' A_0 A_1$$

$$= (A_0' + A_1') S + S' A_0 A_1$$

$$= (A_0 A_1) \oplus S$$

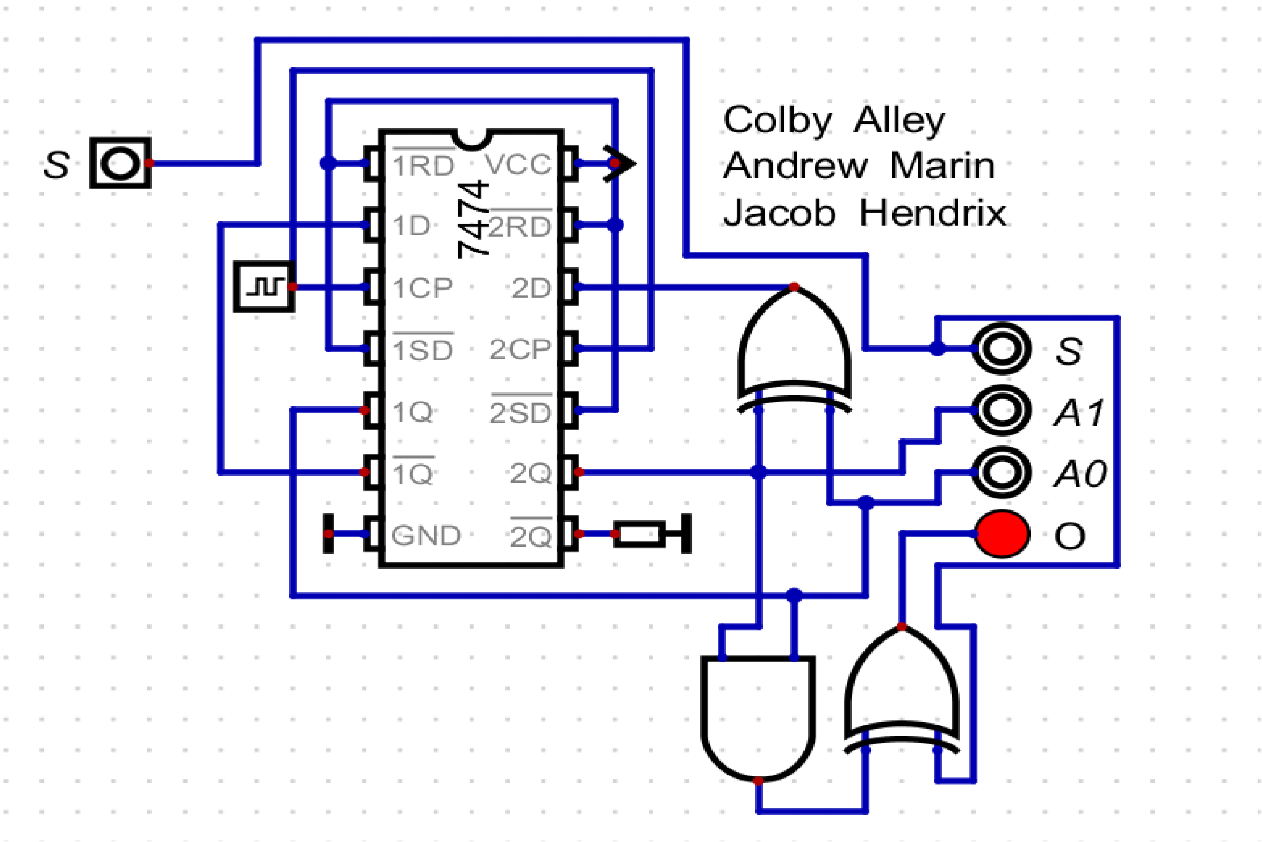
Now we have our output and next states in terms of our current states and selector input. This is all we need to construct the actual circuit. Since we are using D flip flops, we know the characteristic equation is D(t) = Q(t+1), we can use this to determine our D inputs for our flip flops. We have named these A_0 and A_1. A_1 will be our most significant state while A_0 will be our least significant state. We will have the following equations (derived above):

$DA_0(t) = A_0$'

$DA_1(t) = A_1$ XOR A_0

$O(t) = (A_0A_1)$ XOR S

We will use these equations to build our digital tool circuit, test it, and then implement it on our breadboard to ensure it works as intended. Attached below is the digital tool figure.



In this digital tool, I have the D flip flops as the main chip alongside AND and XOR components used for the equations derived above. One thing to note is I have A_0 (D_1) connected to A_0' (Q_1') since it is just inverted. It saves a longer connection and works correctly this way. I have output O connected to an LED and the input S and states (A_1 and A_0) next to the output to be able to see what they are doing as the clock cycles. I have tested and confirmed with the digital tool that the circuit above works as intended and we can now go ahead and implement the circuit physically on our breadboard. We will be using the 74HC74 dual flip flop chip, 74HC08 dual AND gate chip, and the 74HC86 dual XOR chip to implement this circuit. I will be using the StaticIO in the WaveForms program for our S input alongside the states (A_1 and A_0) to share what they are doing during the circuit. I will use a physical LED to show the sequence on our breadboard. Attached below is figure three that shows how the connections will be made using our Analog Discovery kit.

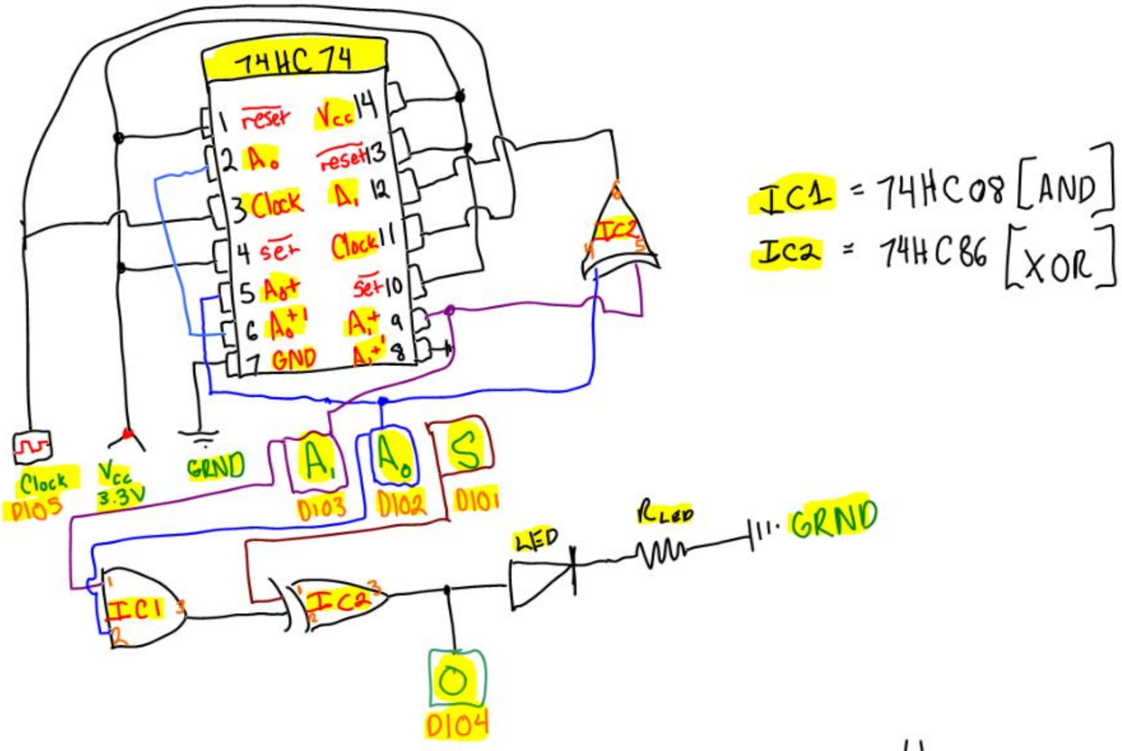
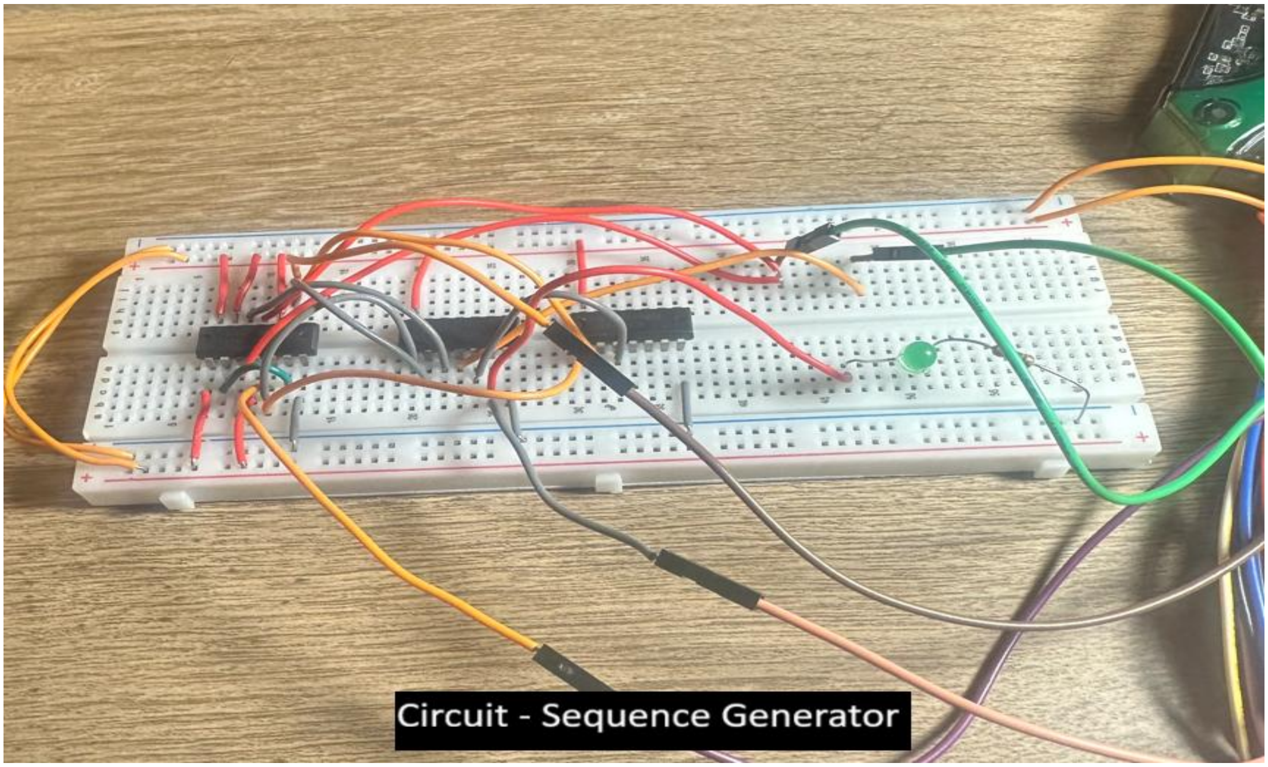


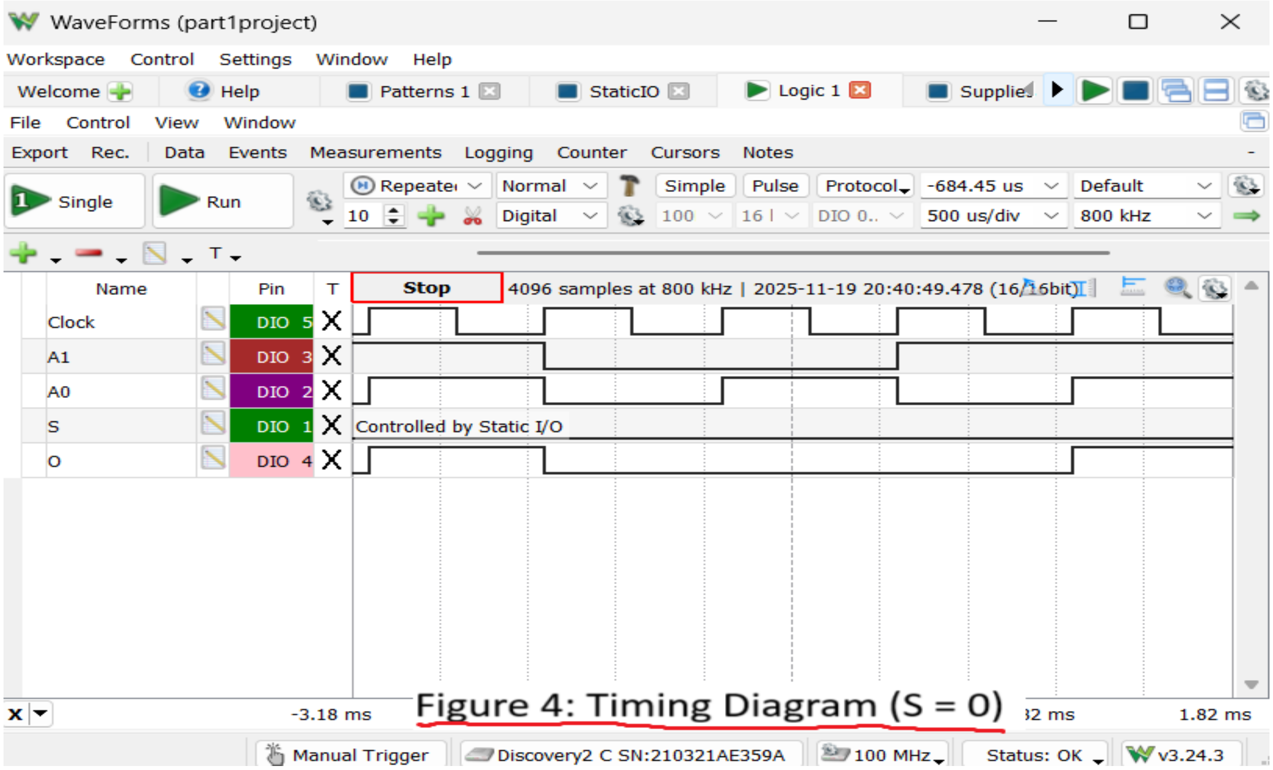
Figure 3: Schematic of Circuit

Above I have the states (A_1 and A_0) connected to DI03 and DI02 and will use StaticIO to display them as LEDs in the circuit. I will do the same with DI04 and use a physical LED on the circuit. That is shown above in figure three. DI01 will be used as the selector (S) input. Below is a picture of our physical circuit design.



Circuit - Sequence Generator

Above is a picture of the sequence generator implementation. It has been personally tested and appears to work as intended. It will also be shown to our lab instructor for verification. I used the same pins labeled in the circuit schematic and works perfectly. Figure four and five below shows the timing diagram of the circuit when selector (S) equals zero, and figure six and seven below shows the timing diagram of the circuit when selector (S) equals one. Also, these timing diagrams show the current states of the circuit alongside the outputs for each selector input. For the timing diagram, the frequency has been increased to show cycles. The demonstration will show the circuit slowed down so we are able to see the circuit generate each sequence per each state. In these timing diagrams, it is a rising edge clock.



WaveForms (part1project)
Workspace Control Settings Window Help
Welcome Help Patterns 1 StaticIO Logic 1 Supplies
File Control View Window
Export Rec. Data Events Measurements Logging Counter Cursors Notes
Single Run
Name Pin T Stop
Controlled by Static I/O
Manual Trigger

Repeated | Normal | Simple | Pulse | Protocol | -684.45 us | Default
10 | Digital | 100 | 16 | DIO 0.. | 500 us/div | 800 kHz

4096 samples at 800 kHz | 2025-11-19 20:40:49.478 (16/16bit)

Name	Pin
Clock	DIO 5
A1	DIO 3
A0	DIO 2
S	DIO 1
O	DIO 4

-3.18 ms … 32 ms … 1.82 ms

Discovery2 C SN:210321AE359A | 100 MHz | Status: OK | v3.24.3

Figure 4: Timing Diagram (S = 0)

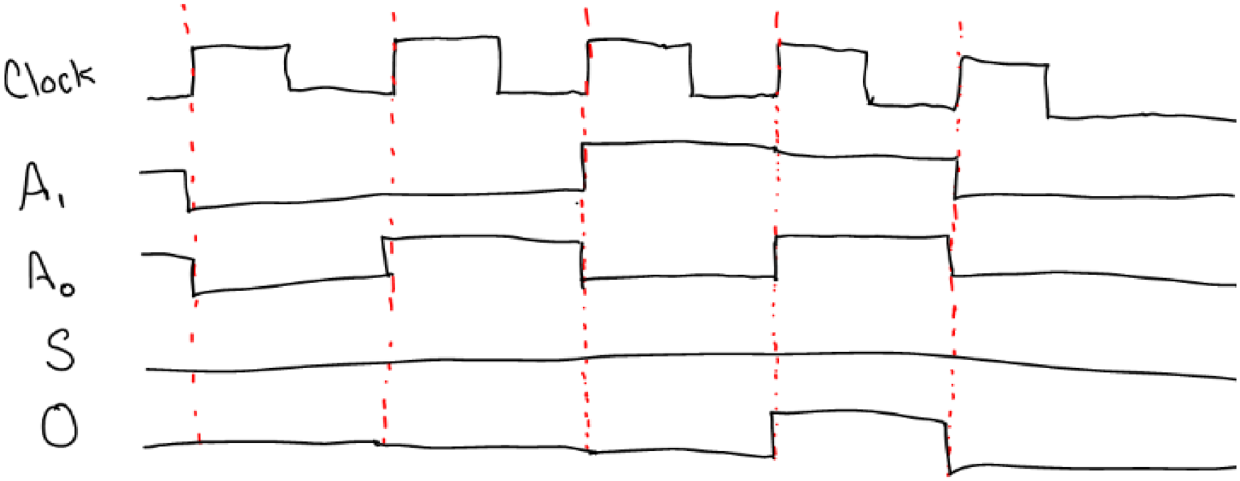
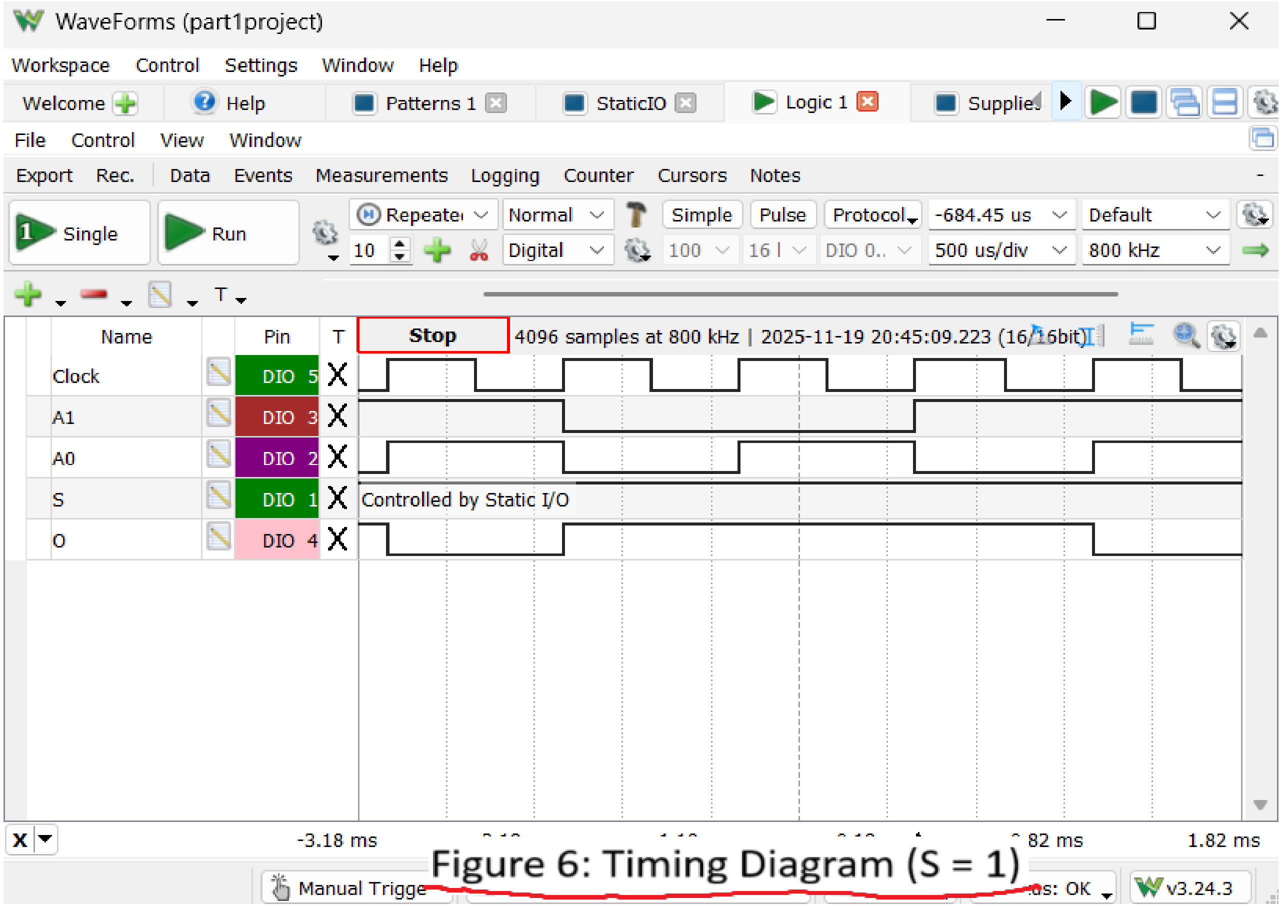


Figure 5: Timing Diagram (S = 1)



4096 samples at 800 kHz | 2025-11-19 20:45:09.223 (16/16bit)

Figure 6: Timing Diagram (S = 1)

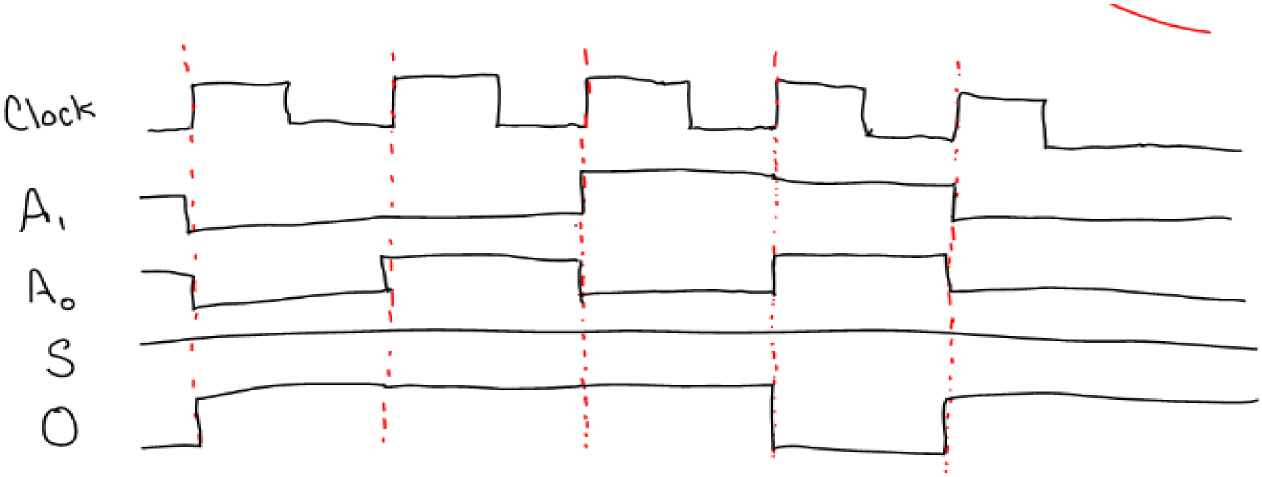


Figure 7: Timing Diagram (S=1)

Conclusion

To conclude, the sequence generator worked as intended after the design was executed. The circuit works correctly, and I am excited to combine this circuit with the sequence detector to learn how to implement a sequence detector. There were not many problems or discrepancies in the implementation of this part. The only issue I had was understanding why it was reading fast. This was because my frequency was slightly too high in the WaveForms program, causing you not to be able to notice different bits being outputted for the sequence being generated. The circuit successfully generates the sequence 0001 when selector is zero and the sequence 1110 when the selector is one.