Cal Poly Pomona

Nonbinary Sequence Counter

Experiment 9

Steven Keesler, Andrew Riad
10/24/2025

0.0 PRELAB

0.1 Design the following nonbinary sequence counter with the sequence 0, 1, 2, 4, 5, 6, 7, and repeat. Use T flip-flops. Draw the state diagram, state table, and the schematic. Is the counter self-correcting? Justify your answer.

Table 0.1 : Nonbinary Sequence Counter State Table

Present State			Next State			FF Inputs		
C	B	A	C	B	A	T2	T1	T0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	1	0	0	1	1	0
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

Figure 0.1 : Nonbinary Sequence Counter State Diagram

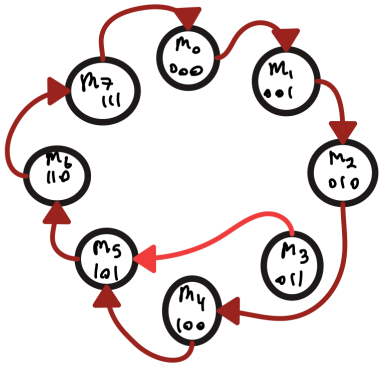


Figure 0.2 : Nonbinary Sequence Counter Schematic

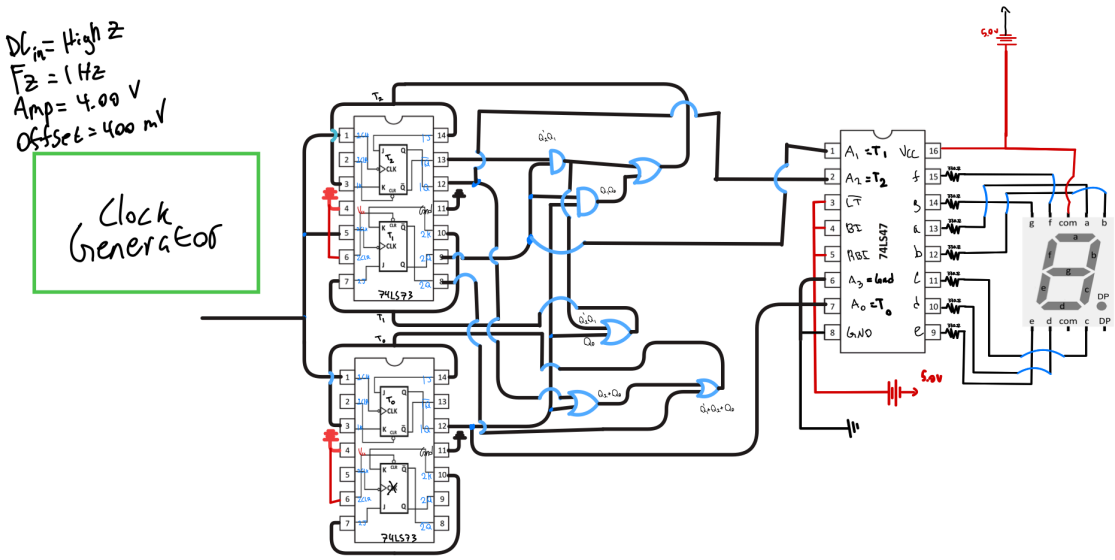
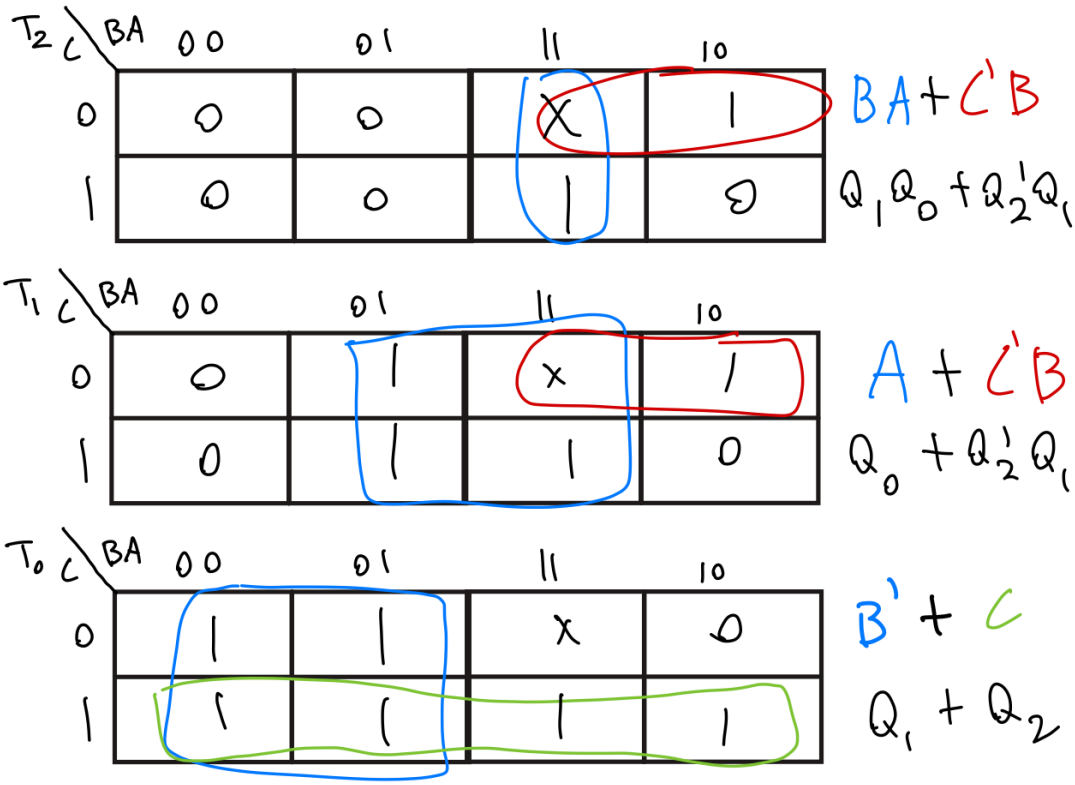


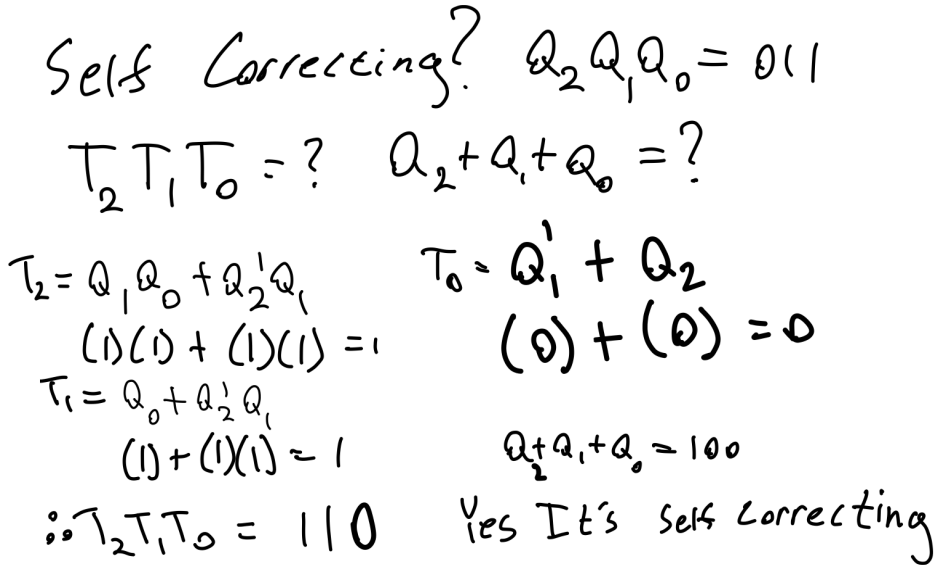
Figure 0.3: K-Map logic from table 0.1



Is our counter self correcting?

Yes I believe it is. While listing minterm 3 as Don't Care inputs, it makes it so whenever 3 does show up, it would simply change to minterm 5 once it runs through, as I've shown in figure 0.4 below.

Figure 0.4: Proof of self correction



1.0 **INTRODUCTION**

Implement the above circuit using a minimum number of chips. Demonstrate the Lab using switches, flip-flops, and seven-segment displays as needed.

2.0 **OBJECTIVES**

The goal is the following:

1. Generate a sequence counter showing the numbers 0-7, except for 3 using a JK FF as a T FF.
2. Encode the circuit to the BCD (C, B, A) and drive a 7-segment via 74LS47 (display 0, 1, 2, 4, 5, 6, & 7).
3. Verify operation across all 16 input combinations.

3.0 **REQUIREMENT**

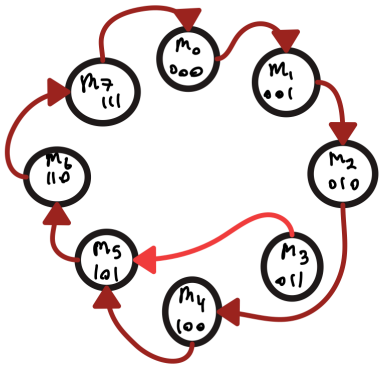
Implement the JK Flip Flop so that it functions as a sequence counter so that the clock shows in order 0, 1, 2, 4, 5, 6, & 7 on the seven-segment when power and the generator are on.

State Table 3.1 is pulled from Pre-Lab Table 0.1 & State Diagram 3.1 is pulled from Pre-lab figure 0.1

Table 3.1 - Nonbinary Sequence Counter State Table

Present State				Next State				FF Inputs		
C	B	A	Dec.	C	B	A	Dec.	T2	T1	T0
0	0	0	0	0	0	1	1	0	0	1
0	0	1	1	0	1	0	2	0	1	1
0	1	0	2	1	0	0	4	1	1	0
0	1	1	3	1	0	0	4	1	1	1
1	0	0	4	1	0	1	5	0	0	1
1	0	1	5	1	1	0	6	0	1	1
1	1	0	6	1	1	1	7	0	0	1
1	1	1	7	0	0	0	0	1	1	1

Figure 3.1 : Nonbinary Sequence Counter State Diagram



4.0 **PARTS LIST**

The Parts list for this experiment is shown below in Table 4.1

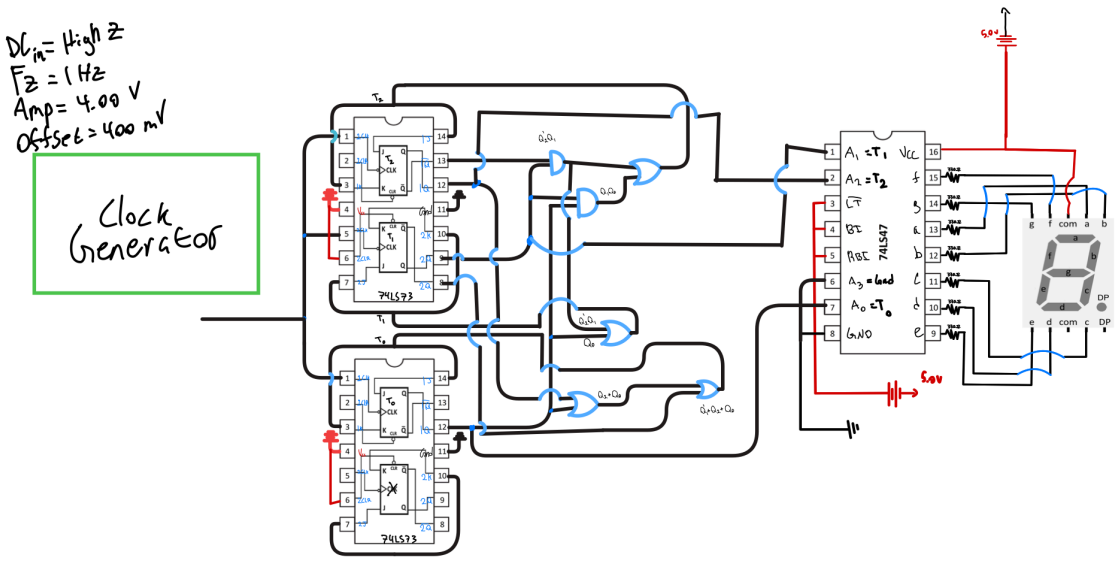
Table 4.1 - Parts List

Item No.	Part Number	Function	Quantity
1	74LS137	JK Flip Flop	2
2	74LS08	AND	1
3	74LS32	OR	1
4	74LS47	BCD-to-7-Seg	1
5	7-Segment	Output	1

5.0 **DESIGN/IMPLEMENTATION**

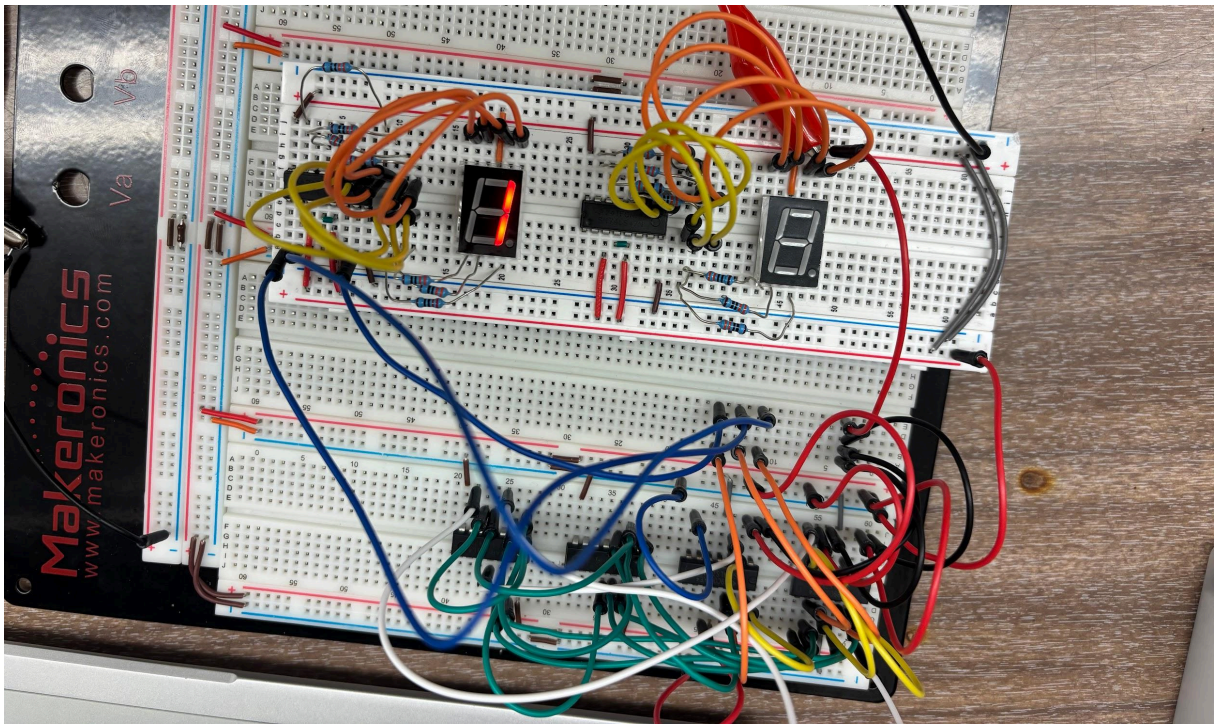
The design from figure 0.2 in the prelab will be tested, and it will be shown again below in figure 5.1. A generator will be used to provide the clock inputs to the JK FFs, and the outputs will be displayed using a 7-segment display and LED.

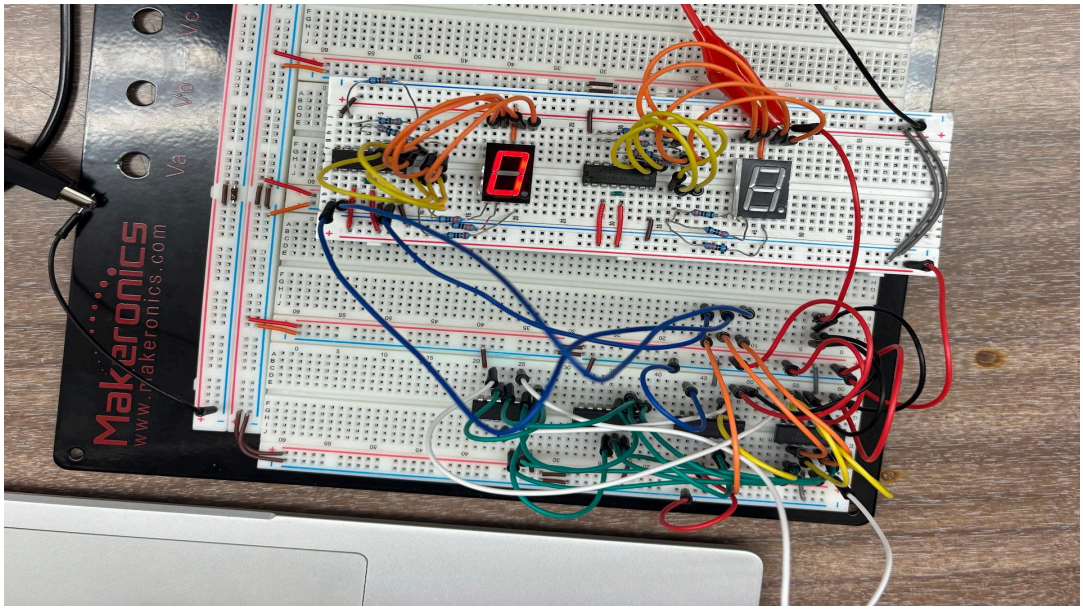
Figure 5.1



6.0 **TEST SETUP**

Below is an image of the breadboard implementing the circuit from figure 5.1. Two 74LS37s generate the T2, T1, & T0 as described. These are encoded to BCD and shown on the 7-segment via the 74LS47. They are also run back into the JK FFs as inputs to generate the next number. The breadboard photo below shows both the FFs and the display/LED interface.





7.0 **TEST RESULTS/VERIFICATION MATRIX**

The test results are summarized below in Table 7.1:

Table 7.1 - Verification Matrix of Counter

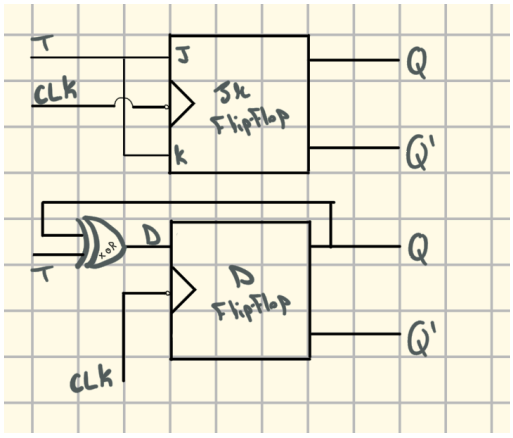
# Shown		Verification
Present State	Next State	Works? (Y/N)
0	1	Y
1	2	Y
2	4	Y
3	5	Y
4	5	Y
5	6	Y
6	7	Y
7	0	Y

8.0 Conclusion

The Counter built from two 74LS137s operated correctly, going from one BCD to another in the sequence mentioned in our state diagram 3.1. It is also self correcting, as planned and shown in the pre-lab figure 0.4. The results across all 8 state transitions, shown in table 7.1, matched the expected behavior in Table 3.1.

9.0 Post Lab

1. Is T flip-flop commercially available? If so, draw the pin assignments from the Internet. If not, draw block diagrams for obtaining T-ff in two ways.
The T flip-flop is not commercially available. So, we can recreate it using either a JK flip-flop or a D flip-flop:



2. How many flip-flops are needed to design a counter to count in the following sequence: 12, 20, 1, 0, and then repeat?
5 flip-flops
Since the highest number in the sequence is $20_{10} = 10100_2$, we need 5 bits to represent it, meaning 5 flip-flops are needed (one for each bit). A flip-flop stores a single binary digit, so 2^5=32 possible states that will cover all needed values up to 20. We can use three IC chips, since each dual flip-flop IC provides two flip-flops (totaling four), and one additional flip-flop is needed for the fifth bit.