

# Lab 10: Three-out-of-Four Detection

**Arturo Salinas-Aguayo**

CSE 2301: Principles and Practice of Digital Logic Design

Dr. Mohammad Khan, Section 003L-1248

Electrical and Computer Engineering Department



College of Engineering, University of Connecticut  
Coded in L<sup>A</sup>T<sub>E</sub>X

## Theory

### A '101' or '010' Serial Sequence Detector

This task involved designing a Moore machine from the problem statement. The successful implementation will yield a 1 if either the detected sequence was 101 or 010 and is allowed to overlap.

These types of codes may represent start and stop codes to signify the start and end of a data stream. For example, if the ACK or acknowledgement code is 101, then the receiver can be programmed or built such that it starts listening after getting to this "state." Another code could be encoded such that it represents NACK or negative-acknowledgement to indicate an error with the preceding bits or operation.

### Combinational Logic vs Sequential Logic

---

#### Combinational Logic

Combinational logic circuits are those in which the outputs depend *only* on the current inputs at any given moment. They do not have any memory element, meaning they cannot retain information about previous input values. As a result, their output is purely a direct response to the present combination of inputs, and it changes immediately whenever an input changes.

- **Definition:** A combinational logic circuit is a network that processes discrete-valued inputs and provides outputs based solely on the current input values. These circuits are described as *memoryless*.
  - **Example Circuits:** Examples of combinational circuits include arithmetic circuits (such as adders and subtractors), multiplexers, decoders, and encoders.
  - **Behavior:** For each possible combination of inputs, there is a unique output determined by the circuit's logic function.
- 

#### Sequential Logic

Sequential logic circuits, unlike combinational logic, have outputs that depend on both the current inputs and the history of past inputs. These circuits contain *memory elements* (typically flip-flops or latches), allowing them to store information about previous states. This capability enables sequential circuits to use past events to influence future outputs, making them suitable for state-dependent operations.

- **Definition:** A sequential logic circuit is one whose output depends on both the current input and past input sequences. These circuits *retain memory* of previous states and can change their state over time.
- **State:** Sequential circuits operate based on a concept of *state*, which summarizes past inputs necessary to determine future behavior.

- **Example Circuits:** Examples include counters, shift registers, flip-flops, and finite state machines (FSMs), such as Moore and Mealy machines.
- 

### Key Differences between Combinational and Sequential Logic

- **Dependency on Past Inputs:** Combinational circuits depend solely on the present inputs, while sequential circuits depend on both the current and past inputs.
  - **Memory:** Combinational circuits are *memoryless*, while sequential circuits include memory elements to retain past states.
  - **Timing:** Combinational circuits respond instantly to input changes, whereas sequential circuits often have a clock signal or other control mechanisms to manage output changes.
- 

## Discussion

In this lab, a sequence detector using a Moore machine was developed to identify two specific binary sequences: 010 and 101. The design required careful attention to state transitions and the use of D flip-flops to hold and propagate the state information. This process emphasized the fundamental differences between combinational and sequential logic, as our sequence detector required memory to retain information about past inputs, which is characteristic of sequential circuits.

### Design Considerations

To achieve the required functionality, we designed a Moore machine where each state transition reflects the detection of a particular state, not input. We used D flip-flops to implement these states, ensuring that the machine could retain the current state information while processing each new bit in the input sequence. Given that two overlapping patterns needed to be detected, the state machine was carefully constructed to handle cases where multiple valid sequences could appear back-to-back within the input string. This resulted in a total of 7 states. This means that we had to have at least 3 Flip-Flops of the D variety.

### State Diagram and Transition Table

A state diagram was created to map out all possible transitions based on the input bit values. Each transition was annotated in the form  $A/B$ , where  $A$  represents the input and  $B$  the output, with the output set to 1 upon detection of either sequence. We ensured that each state had two outgoing transitions, one for an input of 0 and one for 1, allowing the detector to appropriately move between states based on incoming bits. For this lab, the flip flop inputs were labeled  $Q_0$ ,  $Q_1$ , and  $Q_2$  corresponding each to its own flip flop. The excitation states were also labeled as such but superscripted with a \* to indicate excitation. The input and the current output of the D-Flip Flops is combined to make the future or next state logic.

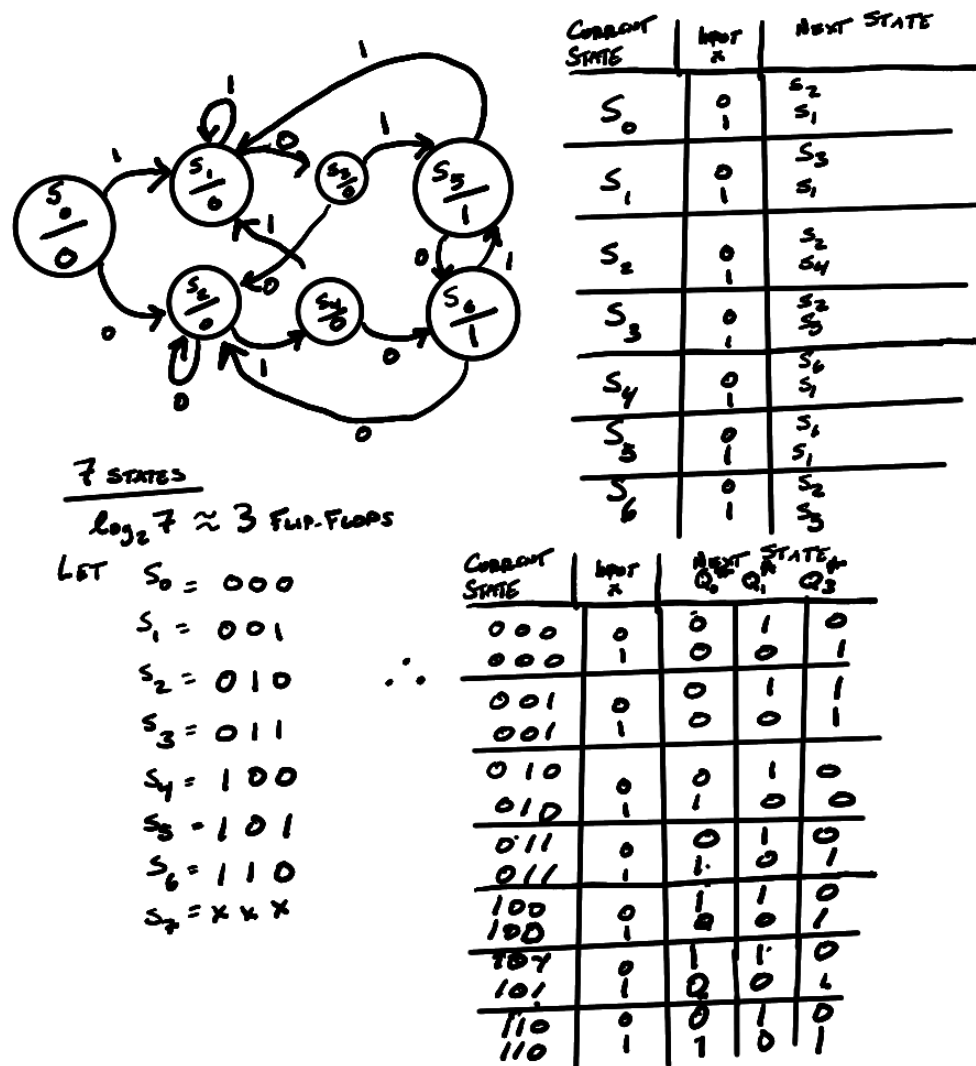


Figure 1: Moore State Diagram

Following this, a transition table was constructed, detailing all possible states and their corresponding next states based on current inputs. The table provided a clear reference for deriving the next-state logic, a crucial step in mapping the circuit behavior to D flip-flops. S0 marks the first state, S6 marks the final state for a total of 7 states as stated previously.

CURRENT STATE	INPUT	NEXT STATE	$Q_0Q_1Q_2$	Input	$Q_0^*$	$Q_1^*$	$Q_2^*$
S0	0	S2	000 (S0)	0	0	1	0
S0	1	S1	000 (S0)	1	0	0	1
S1	0	S3	001 (S1)	0	0	1	1
S1	1	S1	001 (S1)	1	0	0	1
S2	0	S2	010 (S2)	0	0	1	0
S2	1	S4	010 (S2)	1	1	0	0
S3	0	S2	011 (S3)	0	0	1	0
S3	1	S5	011 (S3)	1	1	0	1
S4	0	S6	100 (S4)	0	1	1	0
S4	1	S1	100 (S4)	1	0	0	1
S5	0	S6	101 (S5)	0	1	1	0
S5	1	S1	101 (S5)	1	0	0	1
S6	0	S2	110 (S6)	0	0	1	0
S6	1	S5	110 (S6)	1	1	0	1

Figure 2: Simple and Expanded State Tables

## Karnaugh Maps for Minimization

To simplify the circuit, Karnaugh maps were utilized for each D flip-flop input. This step minimized the Boolean expressions governing state transitions, reducing the overall complexity of the design. By optimizing the input functions, we minimized the number of logic gates required, improving efficiency and reducing potential propagation delay.

$Q_2B$ $Q_0Q_1$	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	1	X	X
10	1	0	0	1

(a)  $Q_0^* = Q_1X + Q_0\overline{Q_1}\overline{X}$

$Q_2B$ $Q_0Q_1$	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	1	0	X	X
10	1	0	0	1

(b)  $Q_1^* = \overline{X}$

$Q_2B$ $Q_0Q_1$	00	01	11	10
00	0	1	1	1
01	0	0	1	0
11	0	X	X	X
10	0	1	1	0

(c)  $Q_2^* = \overline{Q_1}X + Q_2X + \overline{Q_0}\overline{Q_1}Q_2 + Q_0X$

$Q_1Q_2$ $Q_0$	00	01	11	10
0	0	0	1	0
1	0	0	0	1

(d)  $B = Q_0Q_1\overline{Q_2} + Q_0\overline{Q_1}Q_3$

Figure 3: Karnaugh Map Excitation and Output Formulae

## Challenges and Solutions

One of the main challenges was ensuring that the sequence detector could handle overlapping sequences without error. The Moore machine design required precise state assignments and transitions to correctly identify sequences like 010 and 101 even when they appeared consecutively. To address this, we carefully analyzed the state diagram and used D flip-flops to maintain stability during transitions, ensuring accurate sequence detection.

## Conclusion

This lab reinforced key concepts in sequential logic design, particularly the importance of state retention and transition management in sequence detection. By constructing a Moore machine using D flip-flops, we successfully implemented a robust detector capable of identifying overlapping sequences in real-time. The use of Karnaugh maps further streamlined

the design, demonstrating the power of Boolean minimization techniques in digital circuit optimization.

## Practice Questions

### Example 1 A Counter Utilizing the J-K Flip-Flop

Using JK flip flops, design a counter that counts from DCBA=0000 sequentially to DCBA=1011 and then returns to 0000. Complete the table below.

#	D	C	B	A	D*	C*	B*	A*	J <sub>D</sub>	K <sub>D</sub>	J <sub>C</sub>	K <sub>C</sub>	J <sub>B</sub>	K <sub>B</sub>	J <sub>A</sub>	K <sub>A</sub>
0	0	0	0	0	0	0	0	1	0	X	0	X	0	X	1	X
1	0	0	0	1	0	0	1	0	0	X	0	X	1	X	X	1
2	0	0	1	0	0	0	1	1	0	X	0	X	X	0	1	X
3	0	0	1	1	0	1	0	0	0	X	1	X	0	X	X	1
4	0	1	0	0	0	1	0	1	0	X	X	0	0	X	1	X
5	0	1	0	1	0	1	1	0	0	X	X	0	1	X	X	1
6	0	1	1	0	0	1	1	1	0	X	X	0	X	0	1	X
7	0	1	1	1	1	0	0	0	1	X	X	1	X	1	X	1
8	1	0	0	0	1	0	0	1	X	0	0	X	0	X	1	X
9	1	0	0	1	1	0	1	0	X	0	0	X	1	X	X	1
10	1	0	1	0	1	0	1	1	X	0	0	X	X	0	1	X
11	1	0	1	1	0	0	0	0	X	1	X	0	X	1	X	1
12	1	1	0	0	X	X	X	X	X	X	X	X	X	X	X	X
13	1	1	0	1	X	X	X	X	X	X	X	X	X	X	X	X
14	1	1	1	0	X	X	X	X	X	X	X	X	X	X	X	X
15	1	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X

Table 1: J-K Flip Flop Sequence

### Example 2 Determining Flip Flop Inputs from Table

Given the task to draw a state transition diagram for this inputless table we are given:

CURRENT STATE			NEXT STATE		
$Q_2$	$Q_1$	$Q_0$	$Q_2^*$	$Q_1^*$	$Q_0^*$
0	0	0	1	1	1
0	0	1	1	0	1
0	1	0	0	0	0
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	0	1	1

Table 2: A Simple Sequencer

Using the binary encoding scheme, that is, each state starting with  $S_0$  is represented by the three binary digits counting up starting from 0, which corresponds to 000 in this case.

Implementating this with D Flip-Flops is straight forward.

INPUT		OUTPUT
$CLK$	$D$	$Q$
0	0	NO CHANGE
0	1	NO CHANGE
1	0	0
1	1	1

(b) TRUTH TABLE

Figure 4: The D Flip-Flop Truth Table

That is, to move from the current state to the next, excite the D Flip-Flop input precisely how you want the next state. Simple, really.

I use a Karnaugh Map to simplify this mapping for  $D_0^*$ , but unfortunately the logic required is not able to be simplified much due to the groupings.

$Q_0$ \ $Q_1 Q_2$	00	01	11	10
0	1	1	1	0
1	0	1	1	1

Figure 5: The  $D_0^*$  Input Excitation Karnaugh Map



This simplification corresponds with the boolean equation

$$D_0^* = \overline{Q_0}\overline{Q_1} + Q_2 + Q_0Q_1$$

---