Sequence Detector

Indian Institute of Technology Hyderabad

Lab Assignment : 09

EE1200: Electrical Circuits Lab

Harshil Rathan Y	**EE24BTECH11064**
Y Akhilesh	**EE24BTECH11066**

Contents

1 **Experiment Objective** 2

2 **Components Used** 2
- 2.1 **7411** IC 2
- 2.2 **7432** IC 3
- 2.3 **7476** IC 3
- 2.4 **7404** IC 4

3 **FSM** 4
- 3.1 What is a Finite State Machine (FSM)? 4

4 **Moore Machine** 4

5 **Why Moore?** 5
- 5.1 Characteristics 5

6 **Detecting 11011 pattern** 5
- 6.1 General rule for no of states required 5
- 6.2 State Definitions 6

7 **FSM State Diagram** 6

8 **State Transition Table** 6

9 **K-Maps** 7
- 9.1 K-Maps for T_2 7
- 9.2 K-Maps for T_1 8
- 9.3 K-Maps for T_0 8
- 9.4 K-Maps for Z 9

10 **Circuit Diagram and Implementation** 9

11 **Arduino Code** 10
- 11.1 Code Explanation 11

12 **Input/Output Waveform Observation** 11

13 **Conclusion** 12

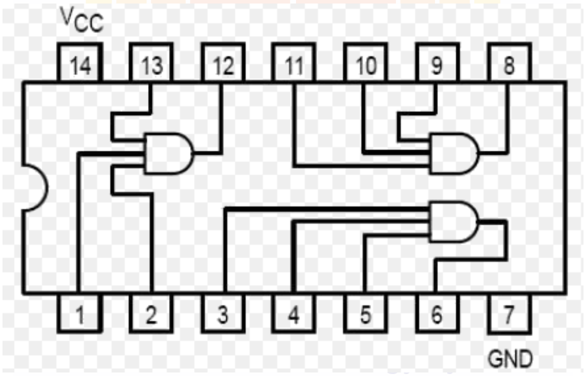
1 Experiment Objective

The objective of this experiment is to design and implement a **Moore sequence detector** that identifies a specified binary input sequence. The detector is realized using **flip-flops and combinational logic gates**, where the **output depends solely on the current state** of the system (Moore model). The design includes defining all states, creating the state transition diagram and table, and verifying the behavior through simulation and waveform analysis.

2 Components Used

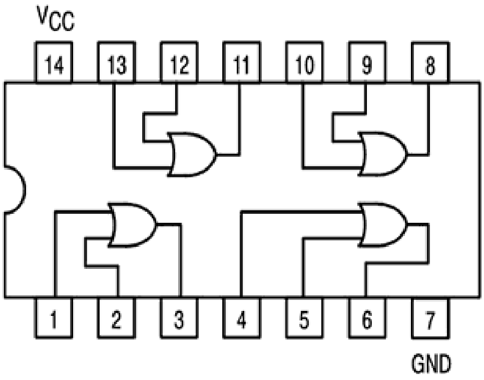
1) Arduino Uno
2) two 7476 JK Flip-Flop
3) 7411 AND Gate IC's
4) 7432 OR Gate IC's
5) 7404 IC (for **NOT** gate)
6) Breadboard & Jumper Wires
7) LEDs

2.1 ***7411*** *IC*



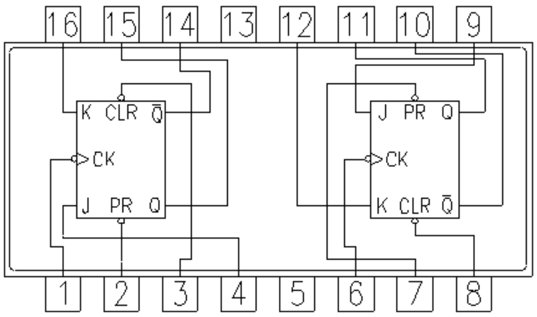
- 7411 Triple 3-Input AND Gate IC .
- We set one input to high when we need only 2 input and gate.
- Used to build multiplication logic

2.2 *7432 IC*



- 7432 Quad 2-Input OR Gate IC .
- It has four independent gates as given in the diagram.
- Used to build addition logic

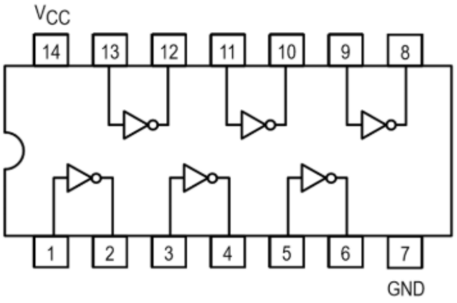
2.3 *7476 IC*



7476
Dual J/K M/S Flip-Flop
with Preset and Clear

- 7476 Dual JK FLip Flop IC's
- Each IC has 2 independent JK Flip Flops
- Used to build increment and decremnet logic based on state diagram

2.4 7404 IC



- 7404 Hex Inverter (NOT Gate) IC
- Each IC contains 6 independent NOT gates
- Used to generate complemented signals required for logic design
- Essential for implementing the state transition logic in the Moore machine

3 FSM

3.1 What is a Finite State Machine (FSM)?

A Finite State Machine (FSM) is a sequential logic circuit that transitions between a finite number of states based on:

- **Inputs**
- **Current state**

There are two main types of FSMs:

- **Moore Machine**: Output depends only on the **current state**
- **Mealy Machine**: Output depends on the **current state and inputs**

In our experiment, the goal is to detect the bit sequence 11011. We define a total of 6 states. The extra state avoids timing issues-such as the inherent delay due to negative-edge triggering of the flip-flops and provides a full clock for output to be clearly visible and valid.

4 Moore Machine

A **Moore Machine** is a type of Finite State Machine (FSM) where the output depends solely on the current state of the system, and not directly on the input. It is formally defined by a 6-tuple

- Sice the output depens only on the current state. This is beneficial as it avoids gliches caused by input changes
-

$$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$$

where:

- Q : Set of finite states
- Σ : Input alphabet
- Δ : Output alphabet
- δ : State transition function, $\delta : Q \times \Sigma \to Q$
- λ : Output function, $\lambda : Q \to \Delta$
- q_0 : Initial state ($q_0 \in Q$)

The output is associated with the state itself. Therefore, the output only changes on state transitions.

5 Why Moore?

Moore Machine: Output depends only on the state
that means:

- Output will only be 1 **after** reaching the **final state** of the sequence.
- All transitions produce an output of `0` except for that one final accepting state.

This makes the design:

- **More stable** (since outputs don't change in the middle of transitions)
- **Glitch-free**, because output doesn't depend directly on rapidly changing inputs

5.1 Characteristics

- Output depends only on the current state
- Output is associated with states
- Synchronous output change
- Slower response compared to Mealy machines
- More states are required to accomplish the same task in a Mealy machine.

6 Detecting 11011 pattern

6.1 General rule for no of states required

To detect a binary sequence of length n, you typically need:

$$\text{Number of states} = n + 1$$

so the no of states needed is '6'.
Theortical reason behind why 6 Bits:
Even though the pattern is made up of 5 bits, flip-flops (which are used to store the state) only update their values when the clock signal changes, typically on the falling edge. Because of this delay, if we only used 5 states, the output might show up too soon or might not line up properly with the rest of the system. By adding a sixth state, we give the system a full clock cycle to make sure the output (which only depends on the current state in a Moore machine) is properly synchronized. This extra state helps the system detect the pattern correctly and at the right time.

- To uniquely encode the finite states in a sequence detector FSM, we use binary encoding.

If the FSM requires N distinct states, the number of bits required is:

$$\boxed{\lceil \log_2 N \rceil} \text{ bits}$$

For example, detecting the binary sequence "11011" needs $N = 6$ states (in a Moore machine). Thus:

$$\lceil \log_2 6 \rceil = 3 \text{ bits}$$

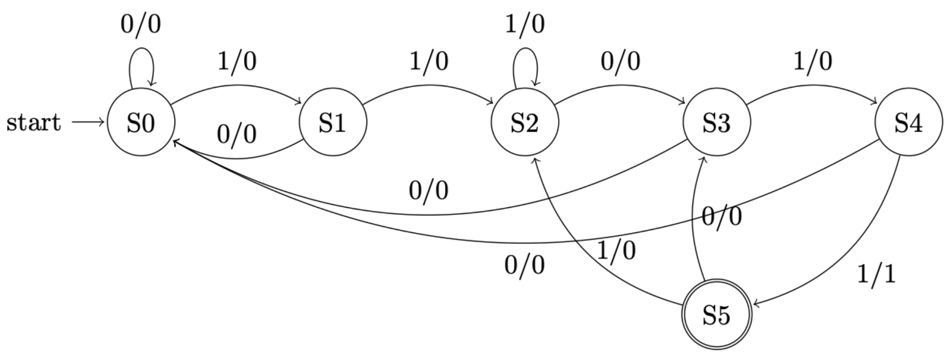
So, we use 3-bit state encoding to represent all possible states in the FSM.

Those six states are as follows:

6.2 State Definitions

- **State 0 (000)**: Initial state, no bits (or irrelevant input) have been received.
- **State 1 (001)**: The FSM has detected a leading 1.
- **State 2 (010)**: The FSM has detected 11 (two successive ones)
- **State 3 (011)**: The FSM has detected 110.
- **State 4 (100)**: The FSM has detected 1101.
- **State 5 (101)**: The FSM has detected the full pattern 11011. In this state, the output is 1.

7 FSM State Diagram



8 State Transition Table

The below table includes present state $Q_2Q_1Q_0$, X as the input and next state $Q_2'Q_1'Q_0'$

$Q_2Q_1Q_0$	X	$Q_2^+Q_1^+Q_0^+$	Z	T_2	T_1	T_0
000	0	000	0	0	0	0
000	1	001	0	0	0	1
001	0	010	0	1	1	1
001	1	001	0	0	0	0
010	0	100	0	1	1	0
010	1	011	0	0	1	1
011	0	010	0	0	0	1
011	1	001	0	0	1	0
100	0	000	0	1	0	0
100	1	101	1	0	1	1
101	0	010	0	1	1	1
101	1	001	0	1	0	0

9 K-Maps

9.1 K-Maps for T_2

Q_0Q_1 \ Q_2X	00	01	11	10
00	0	0	0	1
01	0	0	0	0
11	0	1	0	0
10	0	0	1	1

$$T_2 = \bar{Q}_1 \cdot Q_2 \cdot \bar{X} + Q_0 \cdot \bar{Q}_1 \cdot Q_2 + Q_0 \cdot Q_1 \cdot \bar{Q}_2 \cdot X$$

9.2 K-Maps for T_1

Q_0Q_1 \ Q_2X	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	0	0
10	0	1	1	1

$$T_1 = \bar{Q}_1 \cdot Q_0 \cdot X + Q_0 \cdot \bar{Q}_1 \cdot Q_2 + Q_0 \cdot Q_1 \cdot \bar{Q}_2$$

9.3 K-Maps for T_0

Q_0Q_1 \ Q_2X	00	01	11	10
00	0	1	1	0
01	1	0	0	0
11	1	1	0	0
10	1	1	1	0

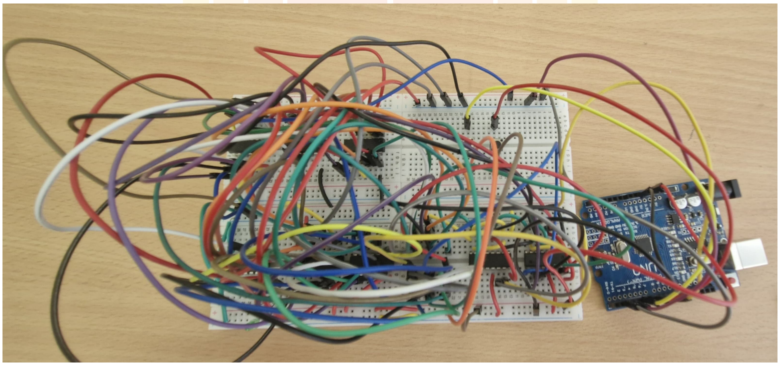
$$T_0 = \bar{Q}_1 \cdot X + Q_0 \cdot \bar{Q}_2 + Q_0 \cdot Q_1 \cdot \bar{Q}_2 \cdot \bar{X}$$

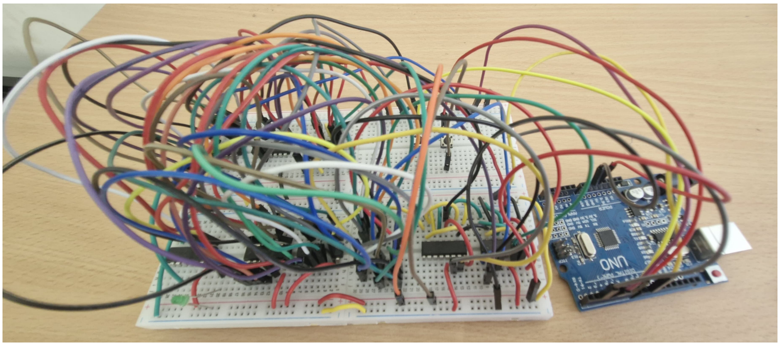
9.4 K-Maps for Z

Q_0Q_1 \ Q_2X	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	0	0
10	0	0	1	1

$$Z = Q_0 \cdot \bar{Q}_1 \cdot Q_2$$

10 Circuit Diagram and Implementation





The FSM Output Z shows 1 ony when the pattern 11011 is detected in a overlapping condition

11 Arduino Code

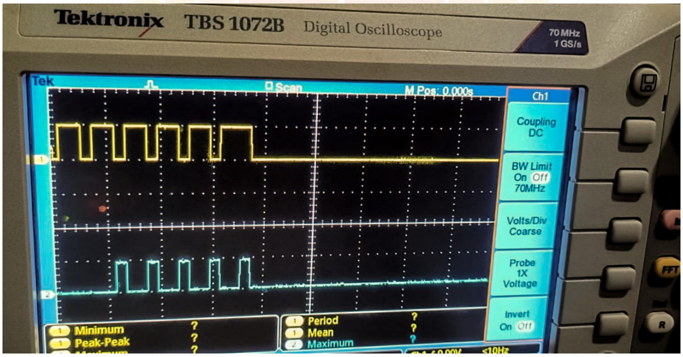
```
const int clockPin = 2; // Clock output to JK flip-flops
const int xPin = 3; // X input to first JK flip-flop (J and K tied to X)
const int xBarPin = 4; // X' input (NOT X) if needed

// Pattern to detect: 11011
int pattern[] = {1, 1, 0, 1, 1};
int len = sizeof(pattern) / sizeof(pattern[0]);

void setup() {
  pinMode(clockPin, OUTPUT);
  pinMode(xPin, OUTPUT);
  pinMode(xBarPin, OUTPUT);

  // Initialize all outputs low
  digitalWrite(clockPin, LOW);
  digitalWrite(xPin, LOW);
  digitalWrite(xBarPin, HIGH); // X' is high initially
  delay(1000); // Allow flip-flops to reset
}

void loop() {
  for (int i = 0; i < len; i++) {
    int xVal = pattern[i];

    // Set X and X'
    digitalWrite(xPin, xVal);
    digitalWrite(xBarPin, !xVal); // Complement of X for hardware logic

    delay(100); // Setup time before clock

    // Clock pulse: rising edge triggers JK flip-flops
    digitalWrite(clockPin, HIGH);
    delay(100); // Clock high duration
    digitalWrite(clockPin, LOW);
    delay(100); // Clock low duration
  }

  // Wait before restarting the pattern
  delay(2000);
}
```

11.1 Code Explanation

- `pattern[]` holds the bit sequence **11011** to be shifted into the flip-flops.
- `clockPin` sends a **rising-edge clock pulse** that triggers all JK flip-flops.
- `xPin` outputs the **current bit (X)** to the JK inputs (J and K tied together).
- `xBarPin` outputs the **complement of X (X')** for use in additional logic (e.g., for toggling or logic gates).
- `setup()` initializes all pins and waits briefly to **stabilize hardware**.
- For each bit:
  - `xPin` and `xBarPin` are updated with current bit and its complement.
  - A **short delay** ensures setup time before the clock pulse.
  - `clockPin` is toggled HIGH → LOW to simulate a **clock pulse**.
- `delay(2000)` at the end gives a **pause before restarting** the pattern.
- Can be connected to a **5-stage JK flip-flop shift register** to shift in and detect the pattern.
- Pattern detection logic can be added externally using **logic gates** (e.g., AND + NOT) to check for `11011`.

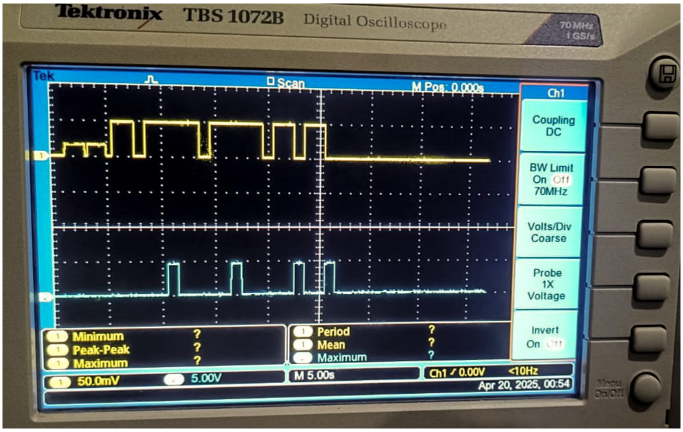
12 Input/Output Waveform Observation

A. Input = '11011011011011011', the output should show high 5 times since the pattern 11011 occurred 5 times in an overlapping case



The output is showing high 5 times which matches our theory and explanation

B. Input = '1101111011011011', we have added a non overlapping case to test. The output should show 1 four times whenever the pattern 11011 is detected



The outputs shows high (1) four times as whenever the pattern is detected

13 Conclusion

The following exp successfully implements a Moore FSM, we have used JK Flip Flops to achieve this. It detects an overlapping input pattern 11011. The report shows detailed analysis and explanation of all the state diagrams, tables and theory related to its working.