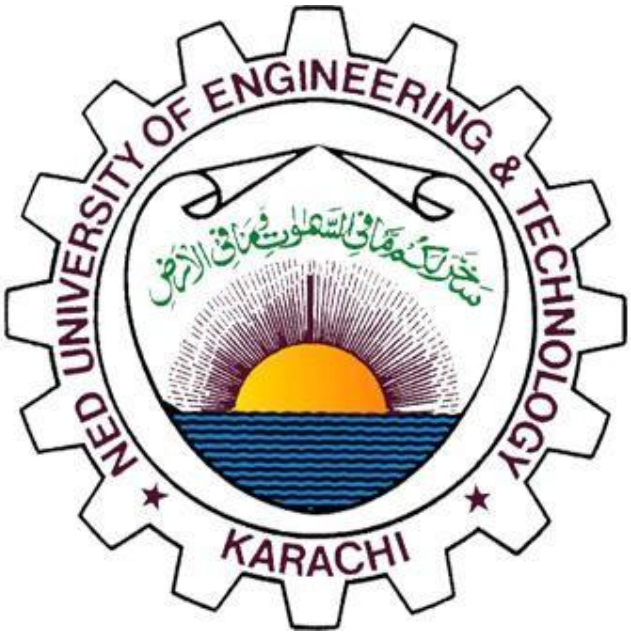


DIGITAL LOGIC DESIGN (TC-201)

OPEN ENDED LAB

NAME:

M.OWAIS (TC-23045)

SUBMITTED TO: DR FAHIM-UL-HAQUE

DEPARTMENT OF TELECOMMUNCIATION ENGINEERING

OPEN ENDED LAB

1. OBJECTIVE:

Design and Implement a sequence detector that meets the following specifications:

- The detector contains single input ‘w’ and output ‘z’
- All changes in the circuit occur on the positive edge of a clock signal
- The output ‘z’ is equal to ‘1’ if input ‘w’ receives “01111110” in the immediately preceeding clock cycles.

2. INTRODUCTION:

In this lab, we will design a circuit with flip-flops and logic gates that will be able to recognize the sequence “01111110”. The detector has a single input w and an output z which goes high when the sequence in question is received. The system works in synchronism with the clock, and the changes of the state take place at the positive transition of the clock signal.

3. WORKING OF THE CIRCUIT:

The working principle of the sequence detector is based on the concept of finite state machines (FSM), where the system transitions between various states depending on the input value and the current state.

3.1. Finite State Machine (FSM):

 The sequence detector functions as an FSM with nine states (S0 to S8).Each state represents how many bits of the sequence have been detected.

3.2. State Transitions:

- The circuit starts at state S0.
- It moves to the next state (S1, S2, etc.) when the input w matches the expected bit of the sequence.
- Each state transition occurs on the rising edge of the clock. If the sequence breaks, the system resets back to an earlier state to start detecting again from the beginning.

3.3. Initial State (S0):

- The circuit begins in the initial state S0, which waits for the first bit (0) of the sequence.
- If w = 0, the circuit transitions to the next state S1, which corresponds to detecting the first 0 of the sequence.
- If the input w = 1, the system remains in state S0, waiting for the sequence to start.

3.4. Intermediate States (S1 to S7):

- As each bit of the sequence is detected (w = 1 for subsequent bits), the system transitions through states S2, S3, …, up to S7.

- Any deviation from the expected sequence (for example, receiving w = 0 when the system expects w = 1) causes the state machine to reset to an appropriate earlier state, typically back to S0, depending on the error.

3.5. **Final State (S8):**

- When the system reaches state S8 after detecting "01111110", the output z is set to 1, signaling the sequence has been detected.
- Afterward, the system resets back to S0 to detect the next sequence

3.6. **Output Logic:**

- The output z is controlled by the state of the system. It remains 0 in all states except S8, where it is set to 1.
- This ensures that the output only goes high (1) when the full sequence has been detected.

4. IMPLEMENTATION USING FLIP FLOPS AND LOGIC GATES:

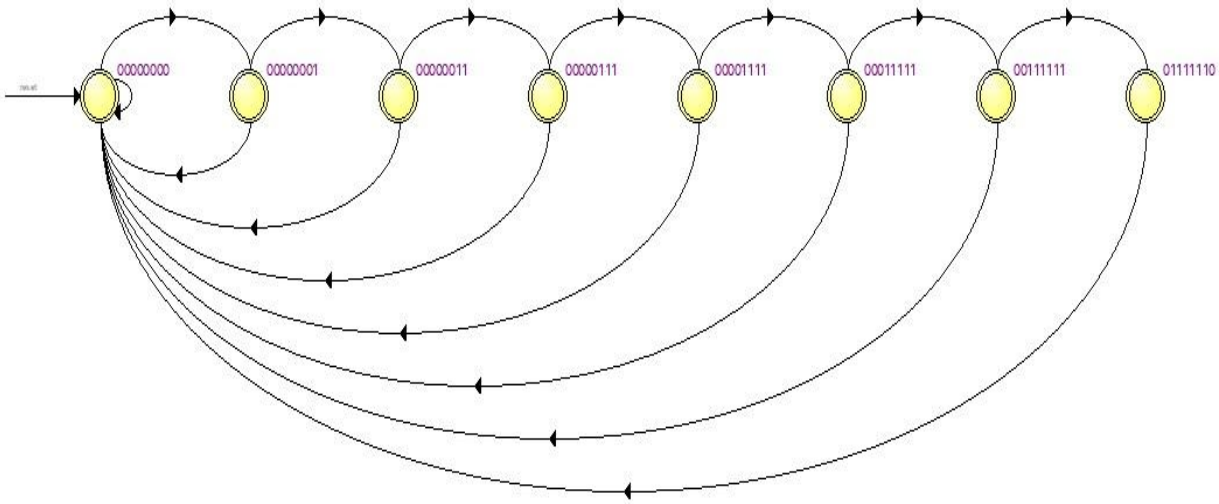
- D flip-flops are used to store the current state of the system. The output of the flip-flops represents the current state, which is used to determine the next state based on the input w.
- Logic gates are used to determine the conditions under which the system transitions from one state to another. For example, an AND gate could be used to detect the specific sequence of inputs required to move from state S7 to S8.

5. STATE TABLE OF THE CIRUIT:

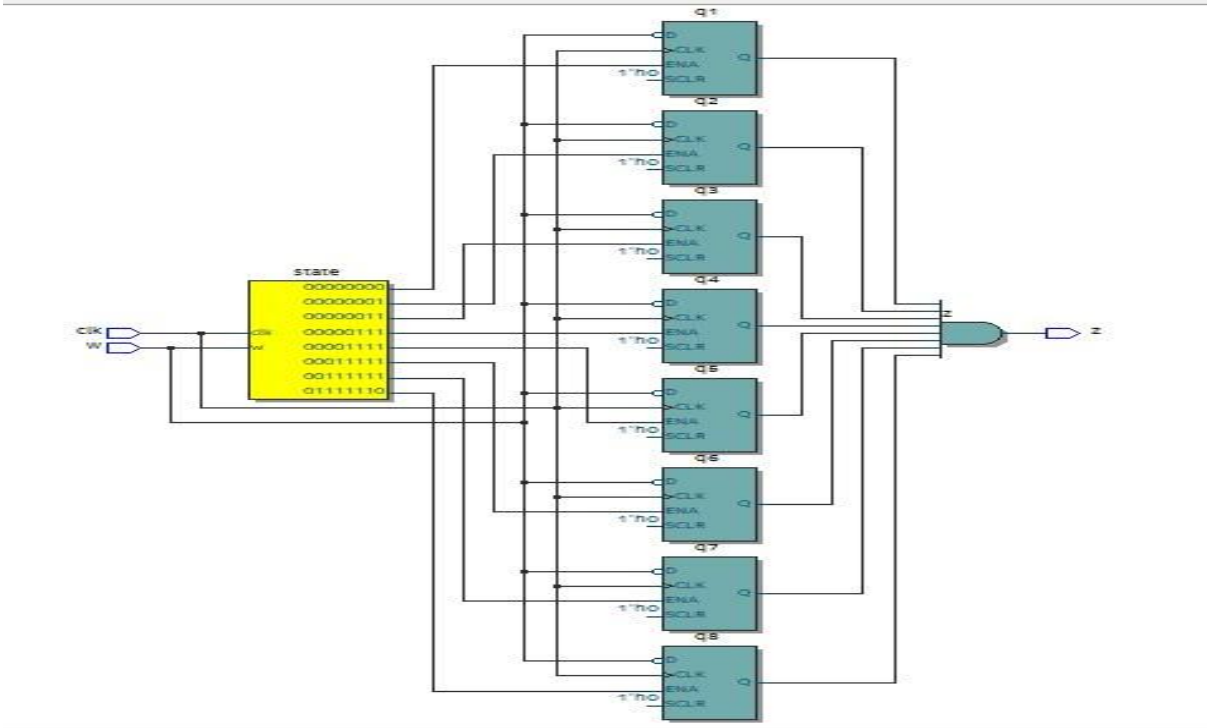
CURRENT STATE	INPUT (w)	NEXT STATE	OUTPUT Z
0000_0000	0	0000_0001	0
0000_0000	1	0000_0000	0
0000_0001	0	0000_0011	0
0000_0001	1	0000_0000	0
0000_0011	0	0000_0111	0
0000_0011	1	0000_0000	0
0000_0111	0	0000_1111	0
0000_0111	1	0000_0000	0
0000_1111	0	0001_1111	0
0000_1111	1	0000_0000	0
0001_1111	0	0011_1111	0

0001_1111	1	0000_0000	0
0011_1111	0	0111_1110	0
0011_1111	1	0000_0000	0
0111_1110	0	0000_0000	1
0111_1110	1	0000_0000	0

6. STATE DIAGRAM OF THE CIRCUIT:



7. CIRCUIT DIAGRAM CONTAING FLIP-FLOPS AND LOGIC GATE:



8. VERILOG IMPLEMENTATION:

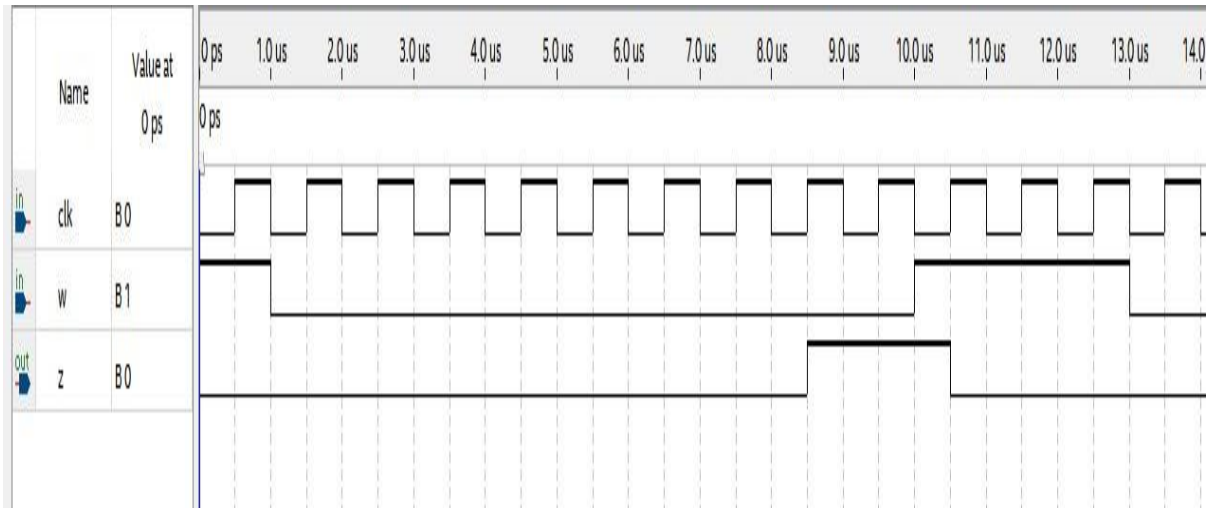
The sequence detector can be implemented in Verilog by defining the states and the transitions between them. The code assigns specific binary values to each state and uses combinational logic to describe the state transitions based on the input w.The output z is driven by the system's state, going high when the system reaches state S8.

8.1. SOURCE CODE:

```
module OEL(w, clk, z);
   input w, clk;
   output z;
   reg q1, q2, q3, q4, q5, q6, q7, q8;
   reg [7:0] state;
   always @(posedge clk) begin
case (state)
8'b0000_0000: begin
        if (w) begin
          state <= 8'b0000_0000;
            q1 <= 1'b0;
     end else begin
       state <= 8'b0000_0001;
           q1 <= 1'b1;
      end
end
8'b0000_0001: begin
          if (w) begin
               state <= 8'b0000_0000;
               q2 <= 1'b0;
              end else begin
     state <= 8'b0000_0011;
       q2 <= 1'b1;  // Proceed to next bit in
sequence
   end
end
8'b0000_0011: begin
        if (w) begin
             state <= 8'b0000_0000;
               q3 <= 1'b0;  // Reset if sequence
breaks
       end else begin
            state <= 8'b0000_0111;
          q3 <= 1'b1;  // Proceed to next bit in
sequence
     end
end
8'b0000_0111: begin
      if (w) begin
            state <= 8'b0000_0000;
  q4 <= 1'b0;  // Reset if sequence breaks
 end else begin
    state <= 8'b0000_1111;
         q4 <= 1'b1;  // Proceed to next bit in
sequence
    end
end
8'b0000_1111: begin
       if (w) begin
                state <= 8'b0000_0000;
```

```
                q5 <= 1'b0;  // Reset if sequence
breaks
        end else begin
                state <= 8'b0001_1111;
            q5 <= 1'b1;  // Proceed to next bit in
sequence
        end
    end
 8'b0001_1111: begin
             if (w) begin
                state <= 8'b0000_0000;
            q6 <= 1'b0;  // Reset if sequence breaks
        end else begin
             state <= 8'b0011_1111;
          q6 <= 1'b1;  // Proceed to next bit in
sequence
       end
end
8'b0011_1111: begin
        if (w) begin
            state <= 8'b0000_0000;
                q7 <= 1'b0;  // Reset if sequence
breaks
             end else begin
                state <= 8'b0111_1110;
            q7 <= 1'b1;  // Detected part of sequence
         end
    end
       8'b0111_1110: begin
     if (w) begin
                state <= 8'b0000_0000;
           q8 <= 1'b0;  // Reset if sequence breaks
      end else begin
                state <= 8'b0000_0000;
            q8 <= 1'b1;  // Sequence completed
        end
    end
    default: begin
      state <= 8'b0000_0000;  // Default state is the
initial state
       end
     endcase
end
assign z = q8 & q7 & q6 & q5 & q4 & q3 & q2 &
q1; //IF ANY IS 0 Z=0
endmodule
```

8.2. SIMULATION OF THE VERILOG CODE:



9. CONCLUSION:

In this lab, we successfully designed and implemented a sequence detector for the sequence "0111_1110" using flip-flops and logic gates. By utilizing the concept of a finite state machine (FSM), we ensured accurate state transitions and output generation based on the input sequence. The circuit operates synchronously with the clock, ensuring reliable and predictable behavior. The output z correctly goes high when the desired sequence is detected.