Using D flip-flops and logic gates, design a Mealy machine that detects the sequence 0010.

A. First, list all the states (s0, s1, etc.) that we need to define, and briefly define them.

s0: first 0 not detected
s1: 0 detected
s2: 00 detected
s3: 001 detected

B. (Four points.) Write out the state diagram (not the circuit diagram).

![State Diagram]

C. Write out the necessary truth table, showing how Next State and Output depend on Present State and Input.

<table>
<thead>
<tr>
<th>q1</th>
<th>q0</th>
<th>In</th>
<th>D1</th>
<th>D0</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>s0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>s0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>s1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>s1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>s2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>s2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>s3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>s3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
D. (Three points.) Write a logic equation for each bit’s Next State and for the Output.

\[ D_0 = \neg(q_0 \text{ OR } q_1 \text{ OR } \text{In}) \text{ OR } q_1 \text{ AND } (q_0 \text{ XOR } \text{In}) \]

\[ D_1 = q_1 \text{ AND } \neg q_0 \text{ OR } \neg \text{In} \text{ AND } q_0 \text{ AND } \neg q_1 \]

\[ \text{Out} = q_1 \text{ AND } q_0 \text{ AND } \neg \text{In} \]

E. (Three points.) Write out the circuit diagram.

F. (Ten points.) Verilog module

module test2 (input clk, input In, output reg LatchedOut);
    reg q0;
    reg q1;
    always@(posedge clk)
        begin
            q0 <= \neg(q_0 \text{ OR } q_1 \text{ OR } \text{In}) \text{ OR } q_1 \text{ AND } (q_0 \text{ XOR } \text{In});
            q1 <= q_1 \text{ AND } \neg q_0 \text{ OR } \neg \text{In} \text{ AND } q_0 \text{ AND } \neg q_1;
        end
    LatchedOut <= q0;
endmodule
LatchedOutput <= q1 & q0 & ~In;
end
endmodule