

Homework 7 – Verilog State Machines (65 points)

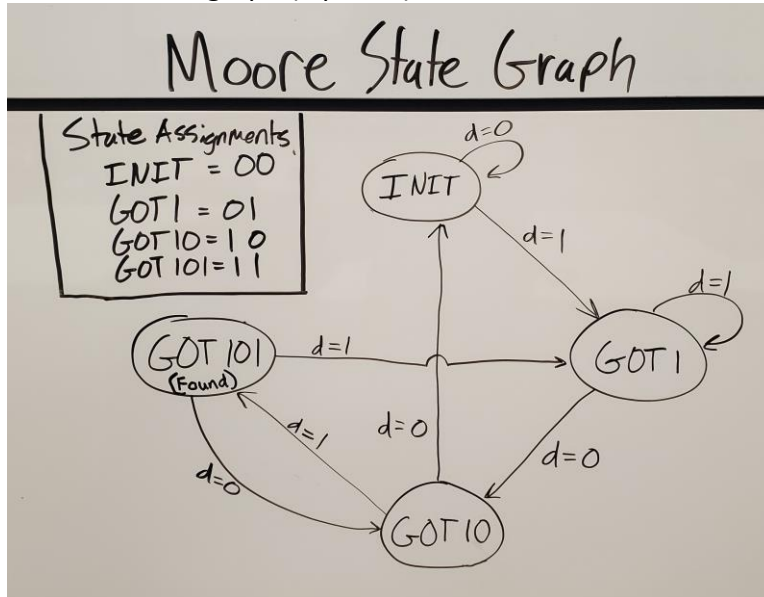
- Use D-type flip-flops and combinational logic to design a synchronous Moore finite-state machine that monitors input “A” and asserts a binary output “B” if the sequence “101” is observed. For example:

If the input data is: **A=010101101**

The output will be: **B=000101001**

-----> time

- Draw the state graph (5 points)



- Create the next state/output table using state names (5 points)

Transition Table

Input (d)	Current State	Next State	Output (found)
0	INIT	INIT	0
1	INIT	GOT1	0
0	GOT1	GOT10	0
1	GOT1	GOT1	0
0	GOT10	INIT	0
1	GOT10	GOT101	0
0	GOT101	GOT10	1
1	GOT101	GOT1	1

- c) Make state assignments and substitute numbers for names in the next state table (5 points)

Next Transition Table

Input (d)	Current State	Next State	Output (found)
0	00	00	0
1	00	01	0
0	01	10	0
1	01	01	0
0	10	00	0
1	10	11	0
0	11	10	1
1	11	01	1

- d) Determine the minimal circuit realization of the next state logic and output (5 points)

N_1 K-map/bodean equation

Q_1, Q_0					
d		00	01	11	10
	0	0	1	1	0
	1	0	0	0	1

$$N_1 = \bar{d}Q_0 + dQ_1\bar{Q}_0$$

N_o K-map/boolean equation

Q_1, Q_0		00	01	11	10
d	0	0	0	0	0
	1	1	1	1	1

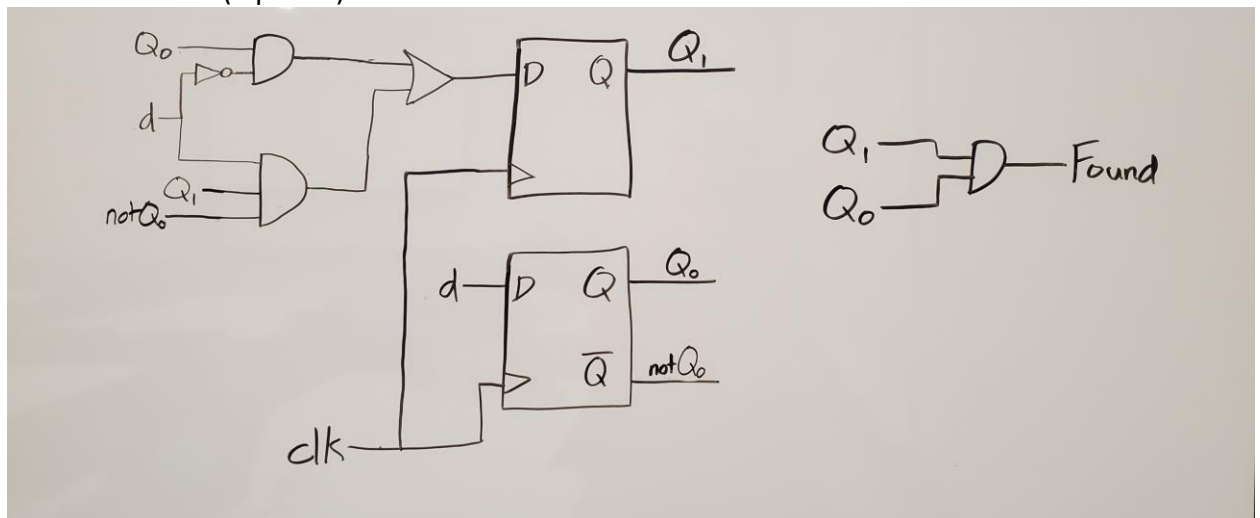
$$N_o = d$$

"Found" K-map/boolean equation

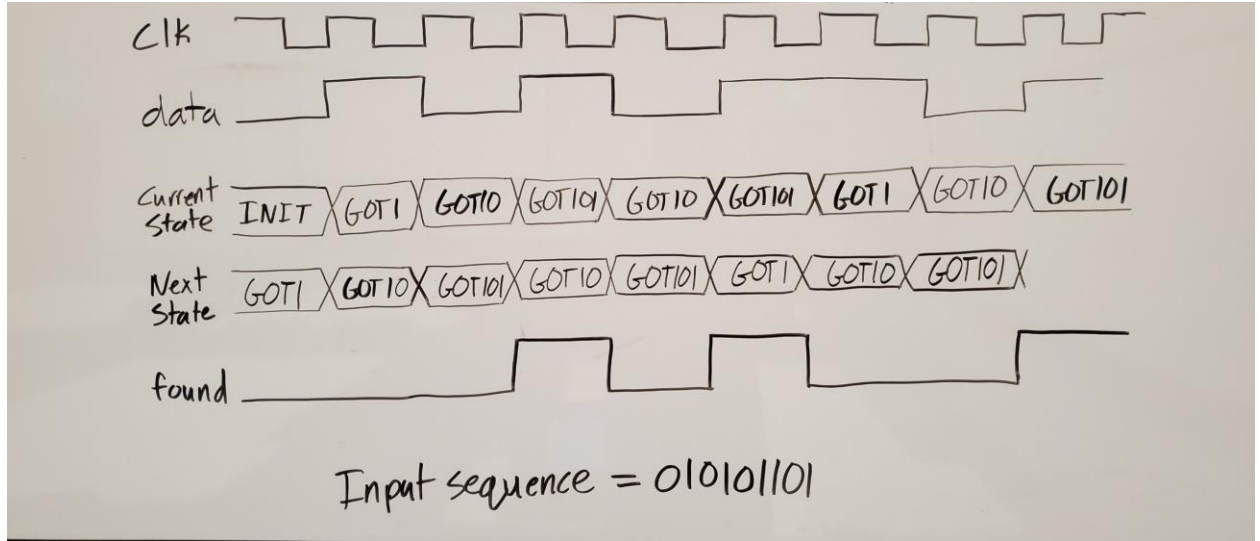
Q_0	0	1
0	0	0
1	0	1

$$\text{Found} = Q_1 Q_0$$

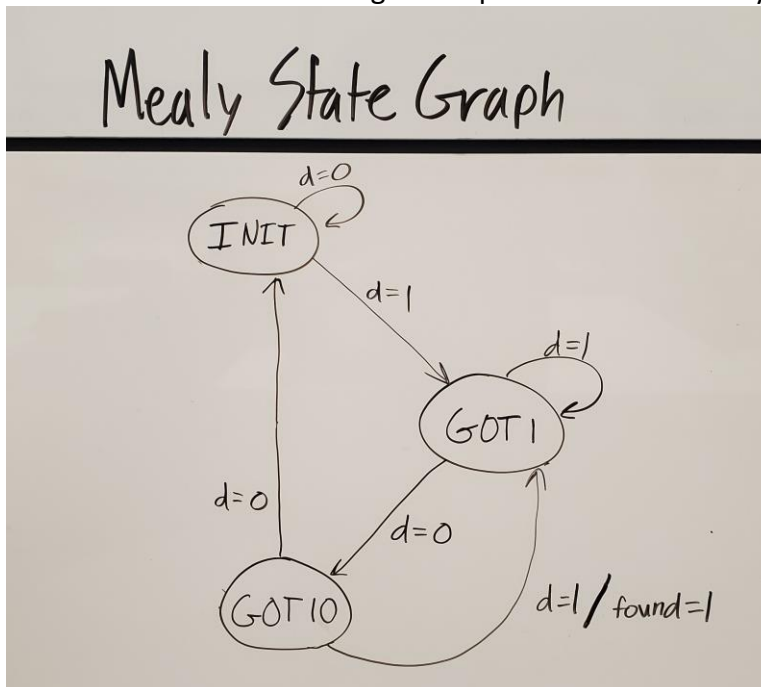
e) Draw the circuit (5 points)



- f) Draw a timing diagram using the input sequence, A = 010101101. Show the input, output, next state, and current state (5 points).



2. Draw the modified State diagram of problem 1 for a Mealy-type State machine (5 points).



3. Write the Verilog code for the above state machine using Moore-type outputs. When writing the code, please do the following:
- Use parameters for the state names (5 points).
For example:
parameter got1 = 2'b00;
 - Use a case statement to determine the next state based on the current state (5 points).
 - Place the state flip-flops in their own "always" block (5 points).
 - Use a separate case statement for the "found" signal (5 points).

```
find_101_sequence.v
s:\young\OneDrive\Documents\Hardware Labs\digital_systems-verilog\ecen340\HW7_Finite_State_Machines\HW7_Finite_State_Machines\HW7_Finite_S

1  `timescale 1ns / 1ps
2  // Company:
3  // Engineer:
4  //
5  // Create Date: 02/21/2025 09:39:22 PM
6  // Design Name:
7  // Module Name: find_101_sequence_moore
8  // Project Name:
9  // Target Devices:
10 // Tool Versions:
11 // Description:
12 //
13 // Dependencies:
14 //
15 // Revision:
16 // Revision 0.01 - File Created
17 // Additional Comments:
18 //
19 //
20 //
21 // Moore state machine to assert "found" as true when a
22 // sequence of "101" occurs from input signal "data"
23 module moore_find_101_sequence(
24     input clk,
25     input data,
26     output reg found
27 );
28
29 // state names
30 parameter INIT = 2'b00;
31 parameter GOT1 = 2'b01;
32 parameter GOT10 = 2'b10;
33 parameter GOT101 = 2'b11;
34
35 reg [1:0] next_state;
36 reg [1:0] state;
37
38 initial state = INIT;
39
40 // next state forming logic
41 always @(*) begin
42     case(state)
43     INIT:
44         if(data) next_state = GOT1;
45         else next_state = INIT;
46     GOT1:
47         if(data) next_state = GOT1;
48         else next_state = GOT10;
49     GOT10:
50         if(data) next_state = GOT101;
51         else next_state = INIT;
52     GOT101:
53         if(data) next_state = GOT1;
54         else next_state = GOT10;
55     default:
56         next_state = INIT;
57     endcase
58 end
59
60 // update the state to the next state at posedge of the clk
61 always @(posedge clk) begin
62     state <= next_state;
63 end
64
65 // output forming logic
66 always @(*) begin
67     case(state)
68     INIT:
69         found <= 0;
70     GOT1:
71         found <= 0;
72     GOT10:
73         found <= 0;
74     GOT101:
75         found <= 1;
76     default:
77         found <= 0;
78     endcase
79 end
80 endmodule
81
82
83
```

Figure 1: Moore state machine Verilog code

```
find_101_sequence_tb.v
/stems-verilog/ecen340/HW7_Finite_State_Machines/HW7_Finite_State_Machines/HW7_Finite_State_Machines.srscs/sim_1/new/find_101_sequence_tb.v

1  `timescale 1ns / 1ps
2  //////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date: 02/21/2025 09:41:03 PM
7  // Design Name:
8  // Module Name: find_101_sequence_tb
9  // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////
21
22 module moore_find_101_sequence_tb();
23     // Testbench signals
24     reg clk;
25     reg data;
26     wire found;
27
28     // Instantiate the Unit Under Test (UUT)
29     moore_find_101_sequence uut (
30         .clk(clk),
31         .data(data),
32         .found(found)
33     );
34
35     // Clock generation: 10ns period (100MHz)
36     always #5 clk = ~clk;
37
38     // Test sequence
39     initial begin
40         // Initialize signals
41         clk = 0;
42         data = 0;
43
44         // Apply test cases
45         $display("Starting test...");
46
47         #10 data = 1; // Input '1'
48         #10 data = 0; // Input '0'
49         #10 data = 1; // Input '1' -> should detect "101"
50         #10 data = 1; // Input '1'
51         #10 data = 0; // Input '0'
52         #10 data = 1; // Input '1' -> should detect "101" again
53         #10 data = 0; // Input '0'
54         #10 data = 0; // Input '0' (invalid sequence)
55         #10 data = 1; // Input '1'
56         #10 data = 0; // Input '0'
57         #10 data = 1; // Input '1' -> should detect "101"
58
59         // Finish simulation
60         #20;
61         $display("Test completed.");
62         $stop;
63     end
64 endmodule
65
66
```

Figure 2: Moore state machine Verilog test bench code

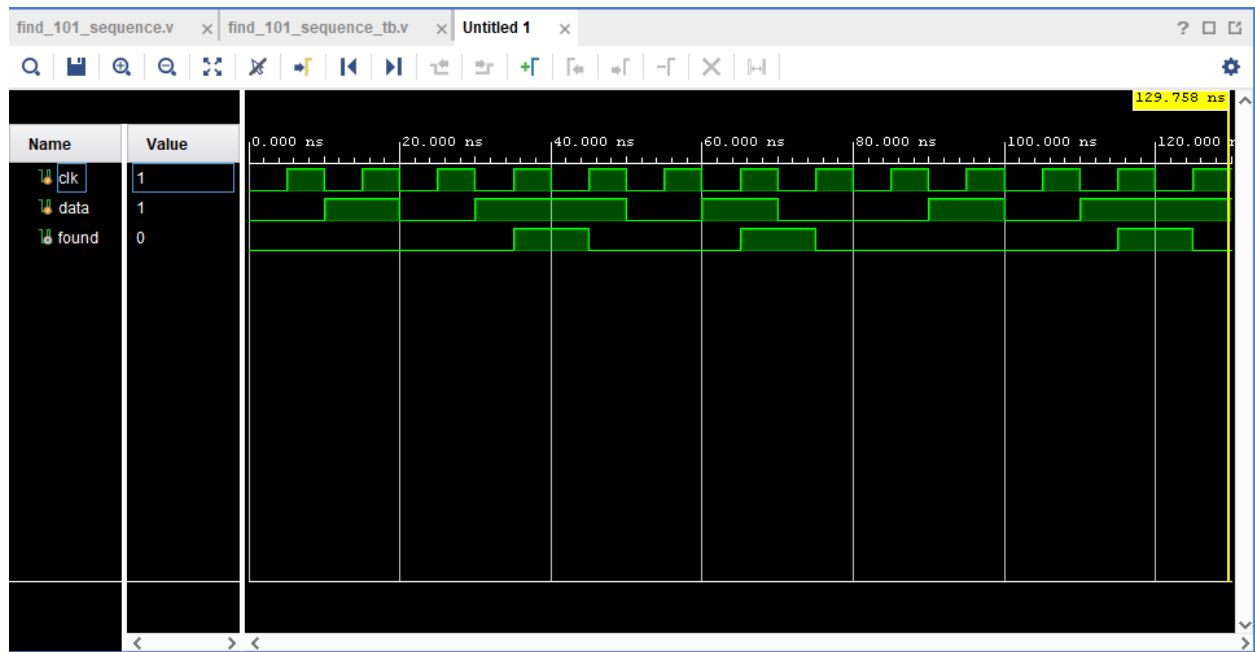


Figure 3: Moore state machine simulation

4. Modify the Moore-type output code for the Mealy-type outputs (10 points).

```
mealy_find_101_sequence.v
erilog/ecen340/HW7_Finite_State_Machines/HW7_Finite_State_Machines/srcs/sources_1/new/mealy_find_101_sequence.v

1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date: 02/21/2025 10:00:57 PM
7  // Design Name:
8  // Module Name: mealy_find_101_sequence
9  // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////
21
22
23 // Mealy state machine to assert "found" as true when a
24 // sequence of "101" occurs from input signal "data"
25 module mealy_find_101_sequence(
26     input clk,
27     input data,
28     output reg found
29 );
30
31 // state names
32 parameter INIT = 2'b00;
33 parameter GOT1 = 2'b01;
34 parameter GOT10 = 2'b10;
35
36 reg [1:0] next_state;
37 reg [1:0] state;
38
39 initial state = INIT;
40
41 // next state forming logic
42 always @(*) begin
43     case(state)
44     INIT:
45         if(data) next_state = GOT1;
46         else next_state = INIT;
47     GOT1:
48         if(data) next_state = GOT1;
49         else next_state = GOT10;
50     GOT10:
51         if(data) next_state = GOT1;
52         else next_state = INIT;
53     default:
54         next_state = INIT;
55     endcase
56 end
57
58 // update the state to the next state at posedge of the clk
59 always @(posedge clk) begin
60     state <= next_state;
61 end
62
63 // output forming logic
64 always @(*) begin
65     case(state)
66     INIT:
67         found <= 0;
68     GOT1:
69         found <= 0;
70     GOT10:
71         // if transitioning from GOT10 to GOT1, assert found as true
72         if(next_state == GOT1) found <= 1;
73         else found <= 0;
74     default:
75         found <= 0;
76     endcase
77 end
78 endmodule
79
80
```

Figure 4: Mealy state machine Verilog code


```

mealy_find_101_sequence_tb.v
verilog/ecen340/HW7_Finite_State_Machines/HW7_Finite_State_Machines/HW7_Finite_State_Machines/srcs/sim_1/new/mealy_find_101_sequence_tb.v

1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date: 02/21/2025 10:09:25 PM
7  // Design Name:
8  // Module Name: mealy_find_101_sequence_tb
9  // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////
21
22 module mealy_find_101_sequence_tb();
23     // Testbench signals
24     reg clk;
25     reg data;
26     wire found;
27
28     // Instantiate the Unit Under Test (UUT)
29     mealy_find_101_sequence uut (
30         .clk(clk),
31         .data(data),
32         .found(found)
33     );
34
35     // Clock generation: 10ns period (100MHz)
36     always #5 clk = ~clk;
37
38     // Test sequence
39     initial begin
40         // Initialize signals
41         clk = 0;
42         data = 0;
43
44         // Apply test cases
45         $display("Starting test...");
46
47         // Test the detection of "101"
48         #10 data = 1; // Input '1'
49         #10 data = 0; // Input '0'
50         #10 data = 1; // Input '1' -> should detect "101" and set found to 1
51
52         #10 data = 1; // Input '1'
53         #10 data = 0; // Input '0'
54         #10 data = 1; // Input '1' -> should detect "101" and set found to 1
55
56         // Test invalid sequence "110"
57         #10 data = 1; // Input '1'
58         #10 data = 1; // Input '1' (invalid sequence)
59         #10 data = 0; // Input '0'
60
61         // Test "101" followed by "110" and check that found stays low
62         #10 data = 1; // Input '1' -> should detect "101"
63         #10 data = 0; // Input '0'
64         #10 data = 1; // Input '1' -> should detect "101" again
65         #10 data = 1; // Input '1' (invalid)
66         #10 data = 0; // Input '0'
67
68         // Finish simulation
69         #20;
70         $display("Test completed.");
71         $stop;
72     end
73 endmodule
74
75

```

Figure 5: Mealy state machine Verilog test bench code



Figure 6: Mealy state machine simulation