ECE426 - Homework 5

Chase Lotito - SIUC Undergraduate

1 Question 1.

Design a Mealy-type FSM that can act as a sequence detector. Complete the state table, and then simulate the FSM with Verilog behavioral-level modelling.

Solution.

Present State	**Next State**		**Output**	
	w = 0	w = 1	w = 0	w = 1
A	B	C	0	0
B	B	C	1	0
C	B	C	0	1

Table 1: Mealy FSM State Table

The code for the Mealy FSM module.

Listing 1: "q1.v"

```
// Chase Lotito - SIUC Undergraduate - Spring 2024
// ECE426 w/ Chao Lu
// HW5
// Q1: Design a Mealy-type FSM that cdan act as a sequence detector.

`timescale 1us / 1us

module mealyFSM (
    clk, reset, w, out
);

// I/O
input clk, reset, w;
output out;             // wire for continuous assignment

reg [1:0] present, next; // p for present-state, n for next-state

// Parameters for cases
// 3 cases, so lets use 2-bits
```

```
parameter A = 2'b01, B = 2'b10, C = 2'b11;  // we will count 1, 2, 3
// Initial Conditions
initial begin
    present = A;
end

// Console log
always @ (w, out) begin
    $monitor("[PRESENT] = %d | [NEXT] = %d, w = %b | [OUTPUT] = %b \n",
        present, next, w, out);
end

// Next-State Combinational Logic
always @ (w, present) begin
    case (present)
        A: next = w ? C : B;
        B: next = w ? C : B;
        C: next = w ? C : B;
        default: next = 2'bxx;
    endcase
end

// Sequential Logic
// using posedge reset because not specified in problem statement.
always @ (posedge clk, posedge reset) begin
    if (reset == 1'b1)
        present <= A;
    else
        present <= next;
end

// Output Logic
assign out = ( (present == B) && (w == 1'b0) ) || ( (present == C) && (
   w == 1'b1) );

endmodule
```

The code for the testbench.

Listing 2: "tb.v (for q1)"

```
// Chase Lotito - SIUC Undergraduate - Spring 2024
// ECE426 w/ Chao Lu
// HW5
// testbench for Q1


`include "q1.v"
```

```
`timescale 1us / 1us

module testbench();

// GTKWAVE
initial begin : GTKWAVE
    $dumpfile("q1.vcd");
    $dumpvars(0, testbench);
end

// I/O
wire out;
reg reset, clk, w;

// Initial Conditions
initial begin : initalConditions
    clk = 0;
    reset = 0;
    w = 0;
end

// Clock
always begin
    #10 clk = ~clk;
end

// Stimulus
initial begin : Stimulus
    repeat(20)
        w = #10 ~w;

    reset <= #50 1'b1;
    #100 $finish;
end

// Instantiate the FSM
mealyFSM U1 (
    .clk(clk),
    .reset(reset),
    .w(w),
    .out(out)
);

endmodule
```

The terminal output from the FSM simulation is shown below in Fig. 2. The parameters A, B, and

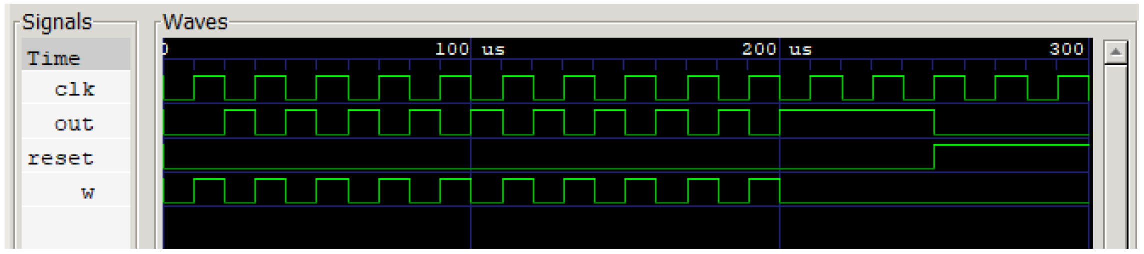
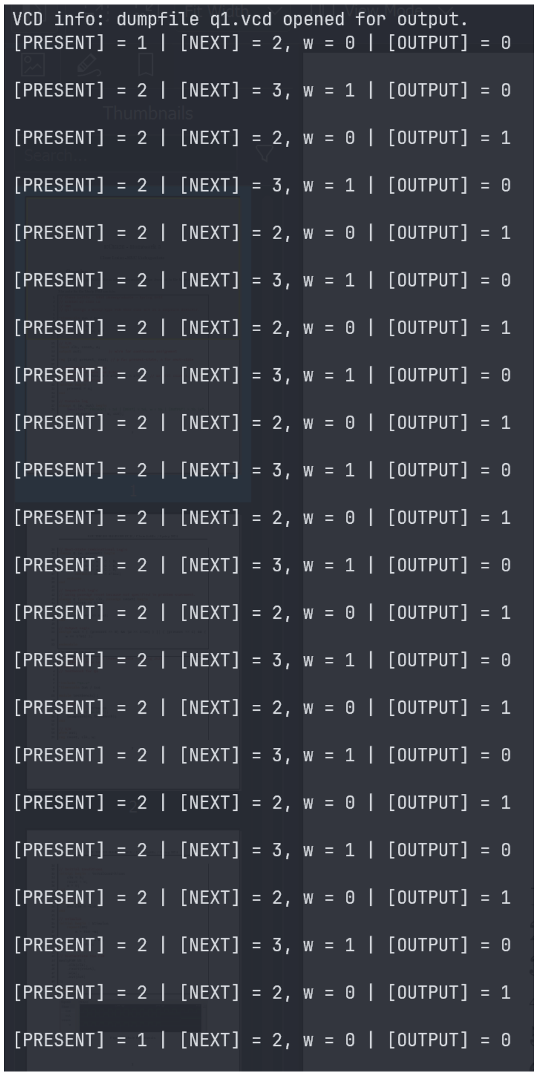


Figure 1: Mealy FSM Signals

C are detected as 1, 2, and 3, respectively. Even though the next state might say a different value as the next line's present state, this is fine because each line does not occur for a positive edge clock cycle.



```
VCD info: dumpfile q1.vcd opened for output.
[PRESENT] = 1 | [NEXT] = 2, w = 0 | [OUTPUT] = 0

[PRESENT] = 2 | [NEXT] = 3, w = 1 | [OUTPUT] = 0

[PRESENT] = 2 | [NEXT] = 2, w = 0 | [OUTPUT] = 1

[PRESENT] = 2 | [NEXT] = 3, w = 1 | [OUTPUT] = 0

[PRESENT] = 2 | [NEXT] = 2, w = 0 | [OUTPUT] = 1

[PRESENT] = 2 | [NEXT] = 3, w = 1 | [OUTPUT] = 0

[PRESENT] = 2 | [NEXT] = 2, w = 0 | [OUTPUT] = 1

[PRESENT] = 2 | [NEXT] = 3, w = 1 | [OUTPUT] = 0

[PRESENT] = 2 | [NEXT] = 2, w = 0 | [OUTPUT] = 1

[PRESENT] = 2 | [NEXT] = 3, w = 1 | [OUTPUT] = 0

[PRESENT] = 2 | [NEXT] = 2, w = 0 | [OUTPUT] = 1

[PRESENT] = 2 | [NEXT] = 3, w = 1 | [OUTPUT] = 0

[PRESENT] = 2 | [NEXT] = 2, w = 0 | [OUTPUT] = 1

[PRESENT] = 2 | [NEXT] = 3, w = 1 | [OUTPUT] = 0

[PRESENT] = 2 | [NEXT] = 2, w = 0 | [OUTPUT] = 1

[PRESENT] = 2 | [NEXT] = 3, w = 1 | [OUTPUT] = 0

[PRESENT] = 2 | [NEXT] = 2, w = 0 | [OUTPUT] = 1

[PRESENT] = 2 | [NEXT] = 3, w = 1 | [OUTPUT] = 0

[PRESENT] = 2 | [NEXT] = 2, w = 0 | [OUTPUT] = 1

[PRESENT] = 2 | [NEXT] = 3, w = 1 | [OUTPUT] = 0

[PRESENT] = 2 | [NEXT] = 2, w = 0 | [OUTPUT] = 1

[PRESENT] = 1 | [NEXT] = 2, w = 0 | [OUTPUT] = 0
```

Figure 2: Mealy FSM Log

2 Question 2.

Implement the FSM in Fig. 3. Show the code and part your simulation showing the correct operation of this FSM.

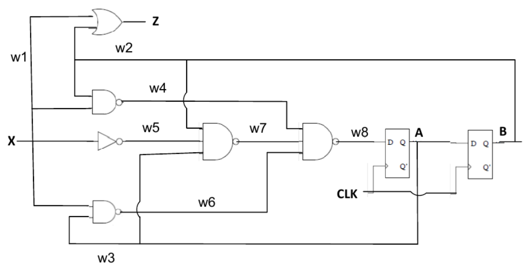


Figure 3: Q2 FSM Circuit

Solution.

Listing 3: "q2.v"

```
// Chase Lotito - SIUC Undergraduate - Spring 2024
// ECE426 w/ Chao Lu
// HW5
// Q2: Code FSM from circuit diagram

`timescale 1us / 1us

// The FSM
module fsm(
    clk, z, x
);

// I/O
input clk, x;
output z;
reg A, B;

// Intermediate wires (w1 == x, w2 == B, A == A)
wire w4, w5, w6, w7, w8;

// -- Gate-Level Modelling --
// level 1
not not1(w5, x);
or or1(z, x, B);
nand nand1(w4, x, B);
nand nand2(w6, x, A);
// level 2
```

```verilog
nand nand3(w7, B, w5, A);
// level 3
nand nand4(w8, w4, w7, w6);
//--------------------------
// -- DFFs ------------------
initial begin
    A = 0;
    B = 0;
end
always @ (posedge clk) begin
    A <= w8;
    B <= A;
end
// --------------------------
initial begin
    $monitor("[%0t us] x = %d, z = %b, w8 = %b, A = %b, B = %b", $time,
        x, z, w8, A, B);
end
endmodule
```

Listing 4: "tb.v (for q2)"

```verilog
// Chase Lotito - SIUC Undergraduate - Spring 2024
// ECE426 w/ Chao Lu
// HW5
// testbench for Q2
`include "q2.v"
`timescale 1us / 1us

module testbench();

// GTKWAVE
initial begin : GTKWAVE
    $dumpfile("q2.vcd");
    $dumpvars(0, testbench);
end

//-------------------------
// I/O
reg clk, x;
wire z;

// Initals
initial begin : initalConditions
   clk = 0;
   x = 0;
end
```

```verilog

// CLOCK
always begin : clock
   #10 clk = ~clk;
end

// PROGRAM LIFETIME
initial begin : programLifeTime
    #100 $finish;
end

// STIMULUS
initial begin : stimmychecks
    repeat(20)
        x = #3 ~x;
end

// MODULE INSTANTIATION
fsm U1 (
    .clk(clk),
    .z(z),
    .x(x)
);

endmodule
```

A problem that I came across becomes clear in the simulation. For the first D-Flip-Flop to get an input of 1, then the next-state combinational logic requires x to equal 0 and 1, simultaneously. So, A and B stay at 0 for the entire simulation.

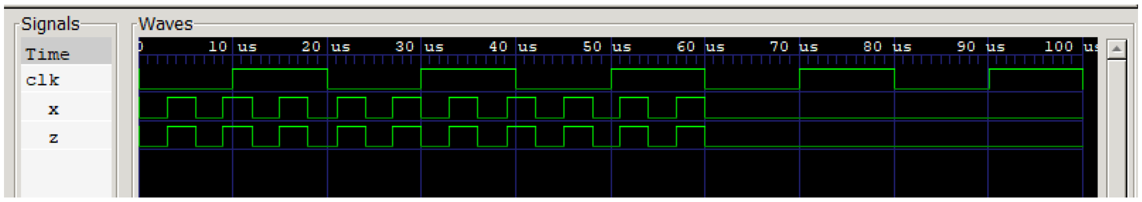
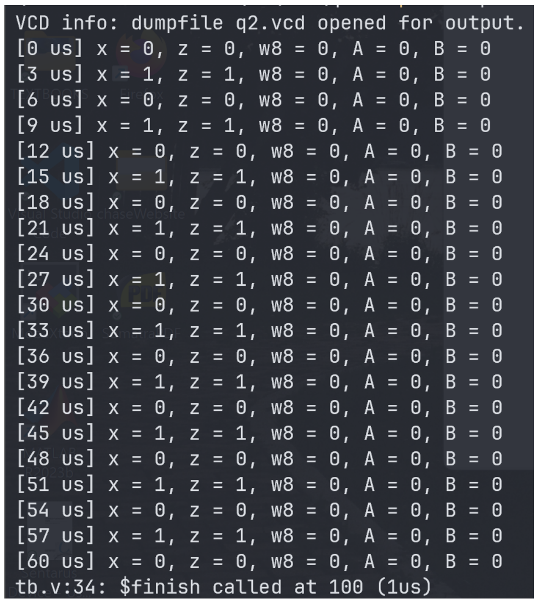


Figure 4: Q2 FSM Signals



```
VCD info: dumpfile q2.vcd opened for output.
[0 us] x = 0, z = 0, w8 = 0, A = 0, B = 0
[3 us] x = 1, z = 1, w8 = 0, A = 0, B = 0
[6 us] x = 0, z = 0, w8 = 0, A = 0, B = 0
[9 us] x = 1, z = 1, w8 = 0, A = 0, B = 0
[12 us] x = 0, z = 0, w8 = 0, A = 0, B = 0
[15 us] x = 1, z = 1, w8 = 0, A = 0, B = 0
[18 us] x = 0, z = 0, w8 = 0, A = 0, B = 0
[21 us] x = 1, z = 1, w8 = 0, A = 0, B = 0
[24 us] x = 0, z = 0, w8 = 0, A = 0, B = 0
[27 us] x = 1, z = 1, w8 = 0, A = 0, B = 0
[30 us] x = 0, z = 0, w8 = 0, A = 0, B = 0
[33 us] x = 1, z = 1, w8 = 0, A = 0, B = 0
[36 us] x = 0, z = 0, w8 = 0, A = 0, B = 0
[39 us] x = 1, z = 1, w8 = 0, A = 0, B = 0
[42 us] x = 0, z = 0, w8 = 0, A = 0, B = 0
[45 us] x = 1, z = 1, w8 = 0, A = 0, B = 0
[48 us] x = 0, z = 0, w8 = 0, A = 0, B = 0
[51 us] x = 1, z = 1, w8 = 0, A = 0, B = 0
[54 us] x = 0, z = 0, w8 = 0, A = 0, B = 0
[57 us] x = 1, z = 1, w8 = 0, A = 0, B = 0
[60 us] x = 0, z = 0, w8 = 0, A = 0, B = 0
tb.v:34: $finish called at 100 (1us)
```

Figure 5: Q2 FSM Log