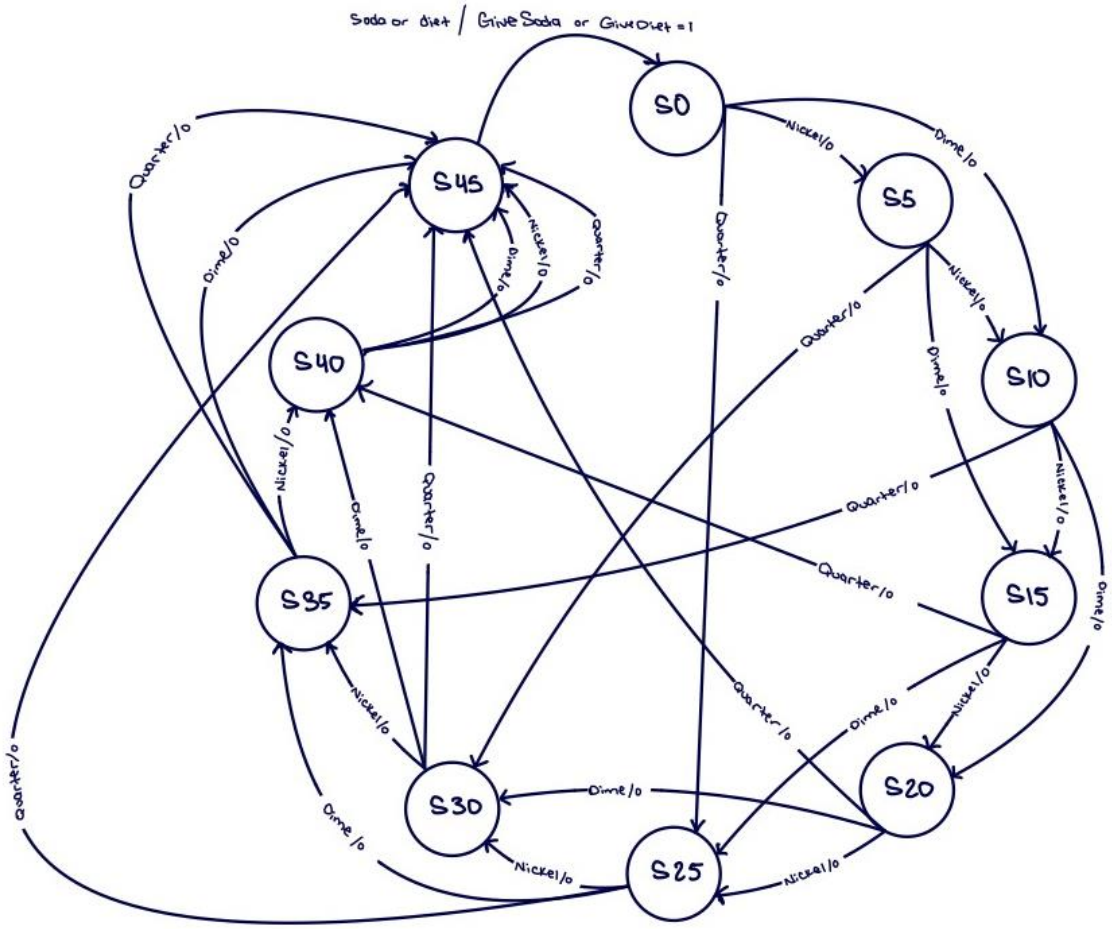
CDA 4203L
Computer System Design Lab

Lab 4 Report
Vending Machine FSM

Today's Date:	14 March 2025
Team Members:	Tanisha Dutta
	Brielle Ashmeade
	Claude Watson
Work Distribution:	Briefly explain the tasks completed by each team member All tasks were completed by all three members together for this project.
No. of Hours Spent:	6+
Exercise Difficulty: (Easy, Average, Hard)	AVERAGE
Any Other Feedback:	N/A

Problem: Show the state diagram of your vending machine controller. Briefly explain how the design works. *Use as many pages as needed.*



Description:

- The FSM starts at S0 and moves to a higher state when coins are inserted. For example, if a nickel is inserted, it moves to S5 or if a dime is inserted it moves to S10.
- The design accumulates money until 45 cents is reached (S45), with each state transition representing the total amount of money inserted.
- Once at S45, it waits for a soda or diet selection. If soda is selected, the output (GiveSoda or GiveDiet) is set to one, the selection is dispensed, and the FSM is reset.
- Additionally, if choice is selected before S45, the selection is stored and once S45 is reached, it dispenses the selection and resets.
- This FSM design follows a mealy structure, where the output depends on the input and current state.

Problem: Include the Verilog code of (a) FSM; (b) Testbench; and (c) Simulation waveforms. *Use as many pages as needed.*

FSM:

```
1  timescale 1ns / 1ps
2
3  module VendingMealy(
4      input clk, reset, quarter, dime, nickel, soda, diet,
5      output reg GiveSoda, GiveDiet
6  );
7
8      parameter S0 = 4'b0000, S5 = 4'b0001, S10 = 4'b0010, S15 = 4'b0011,
9                S20 = 4'b0100, S25 = 4'b0101, S30 = 4'b0110, S35 = 4'b0111,
10               S40 = 4'b1000, S45 = 4'b1001;
11
12     (*FSM_ENCODING = "SEQUENTIAL", SAFE_IMPLEMENTATION = "NO" *)
13     reg [3:0] state;
14     reg queued_soda, queued_diet;
15
16     always @(posedge clk or posedge reset) begin
17         if (reset) begin
18             state <= S0;
19             queued_soda <= 0;
20             queued_diet <= 0;
21             GiveSoda <= 0;
22             GiveDiet <= 0;
23         end
24         else begin
25             GiveSoda <= 0;
26             GiveDiet <= 0;
27
```

```
27
28             case (state)
29                 S0: if (nickel) state <= S5;
30                     else if (dime) state <= S10;
31                     else if (quarter) state <= S25;
32
33                 S5: if (nickel) state <= S10;
34                     else if (dime) state <= S15;
35                     else if (quarter) state <= S30;
36
37                 S10: if (nickel) state <= S15;
38                      else if (dime) state <= S20;
39                      else if (quarter) state <= S35;
40
41                 S15: if (nickel) state <= S20;
42                      else if (dime) state <= S25;
43                      else if (quarter) state <= S40;
44
45                 S20: if (nickel) state <= S25;
46                      else if (dime) state <= S30;
47                      else if (quarter) state <= S45;
48
49                 S25: if (nickel) state <= S30;
50                      else if (dime) state <= S35;
51                      else if (quarter) state <= S45;
52
```

```
52
53                  S30: if (nickel) state <= S35;
54                       else if (dime) state <= S40;
55                       else if (quarter) state <= S45;
56
57                  S35: if (nickel) state <= S40;
58                       else if (dime) state <= S45;
59                       else if (quarter) state <= S45;
60
61                  S40: if (nickel) state <= S45;
62                       else if (dime) state <= S45;
63                       else if (quarter) state <= S45;
64
65                  S45: begin
66                      if (queued_soda || soda) begin
67                          GiveSoda <= 1;
68                          state <= S0;
69                          queued_soda <= 0;
70                      end
71                      else if (queued_diet || diet) begin
72                          GiveDiet <= 1;
73                          state <= S0;
74                          queued_diet <= 0;
75                      end
76                  end

77              endcase
78
79              // if soda or diet is selected before enough money, queue the selection
80              if (state != S45) begin
81                  if (soda) queued_soda <= 1;
82                  if (diet) queued_diet <= 1;
83              end
84          end
85      end
86  endmodule
```

Testbench:

```
`timescale 1ns / 1ps

module FSMtb;

    // Inputs
    reg clk;
    reg reset;
    reg quarter;
    reg dime;
    reg nickel;
    reg soda;
    reg diet;

    // Outputs
    wire GiveSoda;
    wire GiveDiet;

    // Instantiate the Unit Under Test (UUT)
    VendingMealy uut (
        .clk(clk),
        .reset(reset),
        .quarter(quarter),
        .dime(dime),
        .nickel(nickel),
        .soda(soda),
        .diet(diet),
        .GiveSoda(GiveSoda),
        .GiveDiet(GiveDiet)
    );

    always #5 clk = ~clk;

    initial begin
        // initialize inputs
```

```
    initial begin
        // initialize inputs
        clk = 0;
        reset = 1;
        quarter = 0;
        dime = 0;
        nickel = 0;
        soda = 0;
        diet = 0;

        #10 reset = 0;

        // Test 1: insert money normally then select soda normally
        #10 nickel = 1;
        #10 nickel = 0;
        #10 dime = 1;
        #10 dime = 0;
        #10 quarter = 1;
        #10 quarter = 0;
        #10 nickel = 1;
        #10 nickel = 0; // 45 cents
        #10 soda = 1;
        #10 soda = 0; // Select soda

        #20 reset = 1;
        #10 reset = 0;
        #10;

        // Test 2: Diet is selected before enough money is input
        #10 diet = 1;
        #10 diet = 0;
        #10 dime = 1;
        #10 dime = 0;
        #10 quarter = 1;
```

```
66          #10 quarter = 0; //45 cents
67          #10 dime = 1;
68          #10 dime = 0;//   Should automatically dispense Diet
69
70          #20 reset = 1;
71          #10 reset = 0;
72          #10;
73
74          // Test 3:
75          #10 quarter = 1;
76          #10 quarter = 0;
77          #10 quarter = 1;
78          #10 quarter = 0; //50 cents
79          #10 soda = 1;
80          #10 soda = 0;
81
82          #20 reset = 1;
83          #10 reset = 0;
84          #10;
85
86          // Test 4
87          #10 quarter = 1;
88          #10 quarter = 0;
89          #10 nickel = 1;
90          #10 nickel = 0;
91          #10 soda = 1;
92          #10 soda = 0;
93          #10 dime = 1;
94          #10 dime = 0;
95          #10 quarter = 1;
96          #10 quarter = 0; // should automatically dispense
97
98          #20;
99          $finish;
100     end
101 endmodule
```

Waveform: