1) Each of the following state machines can be implemented with a single $D$ flip-flop and some additional gates. Determine the state transition function for these state machines, and draw a circuit diagram. Be sure to include clock and reset lines.

   a. There is a single input $X$. When $X=0$, the next state $N$ is equal to the current state $S$; when $X=1$, the next state is equal to the complement of the current state.

   b. There are two inputs $A$ and $B$. When $A=0$ and $B=0$, the next state is equal to the current state; when $A=0$ and $B=1$, the next state is 0; when $A=1$ and $B=0$, the next state is 1; and when $A=1$ and $B=1$, the next state is equal to the complement of the current state.

2) Implement the state machines above in Verilog by instantiating the following $D$ flip-flop module:

   ```verilog
module dff(clk, rst, D, Q);
  input clk, rst, D;
  output reg Q;
  always @(posedge clk or posedge rst)
    if (rst)
      Q <= 0;
    else
      Q <= D;
endmodule
```

   a. Create a module `flippy(clk, rst, X, S)` implementing the first state machine. Let $S$, the current state, be the output of your module.

   b. Create a module `floppy(clk, rst, A, B, S)` implementing the second state machine.

3) Create a state machine that has 5 bits of input ($W, D_3, D_2, D_1, D_0$), and 4 bits of state ($S_3, S_2, S_1, S_0$). The transition to the next state $N_i$ from the current state $S_i$ is determined by the truth table below:

   $$
   \begin{array}{c|cccc}
   W & N_3 & N_2 & N_1 & N_0 \\
   \hline
   1 & D_3 & D_2 & D_1 & D_0 \\
   0 & S_2 & S_1 & S_0 & D_0 \\
   \end{array}
   $$

   a. Draw a gate diagram implementing the state machine. Be sure to include clock and reset lines.

   b. Create a Verilog module `shifty(clk, rst, W, D, S)` implementing the state machine by instantiating the `dff` module above four times. Let $S$, the 4-bit current state, be the output of your module.
4) An accumulator is a special type of state machine that adds up a series of numbers, one at each clock cycle. We will create a 4-bit accumulator which has 4 bits of input \((X_3, X_2, X_1, X_0)\) and 5 bits of output \((R_3, R_2, R_1, R_0, F)\). The \(R\) outputs are the 4 bits of the running total, and the \(F\) output is set to 1 if the running total cannot be properly represented in a 4-bit number (i.e. if overflow occurred while summing up the inputs).

   a. A 4-bit adder has three inputs: two 4-bit addends \(A\) and \(B\), as well as a 1-bit carry input \(C_{in}\). The adder has a 4-bit output \(S\) indicating the sum of \(A\) and \(B\) as well as a 1-bit carry output \(C_{out}\).

   Show that a 4-bit accumulator can be implemented with a 4-bit adder, five \(D\) flip-flops, and one additional OR gate. Draw a diagram. For clarity, you can leave out the clock and reset lines.

   b. Create a Verilog module \(\text{accum4}(\text{clk}, \text{rst}, X, R, F)\) implementing a 4-bit accumulator. Your module should instantiate the \(\text{dff}\) module above five times, and instantiate the 4-bit adder module \(\text{add4}\) once:

   ```
   module add4(cin, A, B, S, cout);
   input cin;
   input [3:0] A;
   input [3:0] B;
   output [3:0] S;
   output cout;
   assign {cout, S} = A+B+cin;
   endmodule
   ```

5) Create a state machine to recognize the sequence 1001. Your state machine should have a single bit of input \(X\) and a single bit of output \(Y\), which is 1 for the input sequence 1001, and 0 otherwise. Note that the sequence 1001001 should cause the output to be 1 twice: once for the second 1 in the input and once for the third 1 in the input.

   a. Draw a state diagram for a Moore state machine recognizing the sequence 1001.

   b. Draw a state diagram for a Mealy state machine recognizing the sequence 1001.

   c. How many bits of state are required for the Moore state machine? How many for the Mealy? Which design would use fewer flip-flops?

   d. Create a Verilog module \(\text{seq1001_moore}(\text{clk}, \text{rst}, X, Y)\) which uses behavioral Verilog (not using the \(\text{dff}\) module – like we saw in class on Friday) to implement your Moore state machine.
e. Create a Verilog module `seq1001_mealy(clk, rst, X, Y)` which uses behavioral Verilog to implement your Mealy state machine.

6) Bob the digital circuit designer has been assigned to implement a digital circuit for a Coke machine. A Coke costs $0.15. (Wow, cheap Coke!) The machine accepts only nickels and dimes.

The circuit has three inputs: N, D, and R. The N input is set to 1 if a nickel was inserted, D if a dime was inserted, and R if the change return button was pressed. At most one of the inputs can be set to 1 at once (it’s impossible to insert a nickel and a dime at the same time, and the change return button is simply ignored in the unlikely event that a coin was dropped in the same clock cycle). The clock input flips every 10 µs. When an input is set to 1, it turns on 5 µs before the positive edge of the clock, and stays on for 20 µs.

The circuit has two outputs: C and L. When C goes from 0 to 1, the coins in the collector are returned to the customer, and when L goes from 0 to 1, the latch to dispense a Coke is released, and the coins in the collector are stored inside the machine.

Here is the circuit that Bob designed:

![Circuit Diagram]

What’s supposed to happen is this: when a customer puts in coins, the machine waits until at least $0.15 is deposited, and then dispenses a soda. If the customer hits the change return button before $0.15 is deposited, their money is returned. However, the machine isn’t working correctly!
Help Bob find out what’s wrong with the machine:

a. Write a Boolean function for \( N_0 \) in terms of the current state and inputs. Do the same for \( N_1 \).

b. Explain, in plain English, the conditions under which \( S_0 \) will become 1 (i.e. \( N_0 \) is 1). Do the same for \( S_1 \).

c. It is given that at most one of \( N, D, \) and \( R \) can be 1. Explain why exactly one of \( N, D, R, \) and \( X \) is 1 at any time.

d. Create a truth table for \( N_1 \) and \( N_0 \) by considering each possible state of \( S_1 \) and \( S_0 \), and each possible “input” (\( N, D, R, \) or \( X \)). Since there are 4 possible states to be in (00, 01, 10, 11) and four possible inputs, your truth table should have 16 rows. The first four rows look like this:

<table>
<thead>
<tr>
<th>( S_1 )</th>
<th>( S_0 )</th>
<th>input</th>
<th>( N_1 )</th>
<th>( N_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>( N )</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>( D )</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>( R )</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>( X )</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

e. Assume the machine starts in state 00. How much money has the customer deposited when it is in state 01? How much when it is in 10?

f. Create a state diagram for the machine. Start with state 00. Each state should have 4 edges (arrows) emerging from it, labeled \( N, D, R, \) or \( X \). Leave plenty of room to add notation for the outputs (see next item).

g. Is the state machine a Moore state machine or a Mealy state machine? If Moore, label each state in your state diagram with the outputs \( C \) and \( L \). If Mealy, label each edge in your diagram with the outputs \( C \) and \( L \).

h. Can the machine ever enter state 11 through normal operation? Why or why not?

i. Explain, using your state diagram, that inserting a nickel followed by a dime results in the correct dispensing of a soda.

j. Explain two ways that a soda can be purchased for only $0.10.

k. Explain how a clever customer can get two sodas for the price of one.

l. Does Bob’s choice of Moore vs. Mealy seem incorrect to you? Explain why or why not.