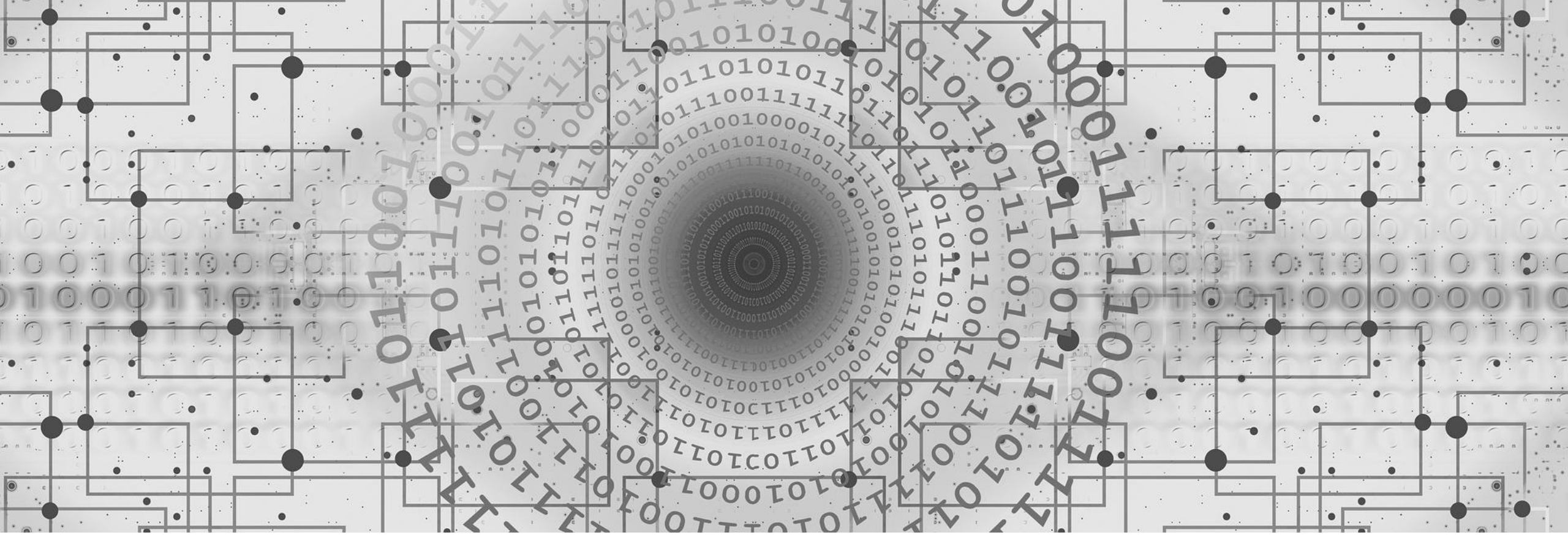


Homework-6a : Vending Machine

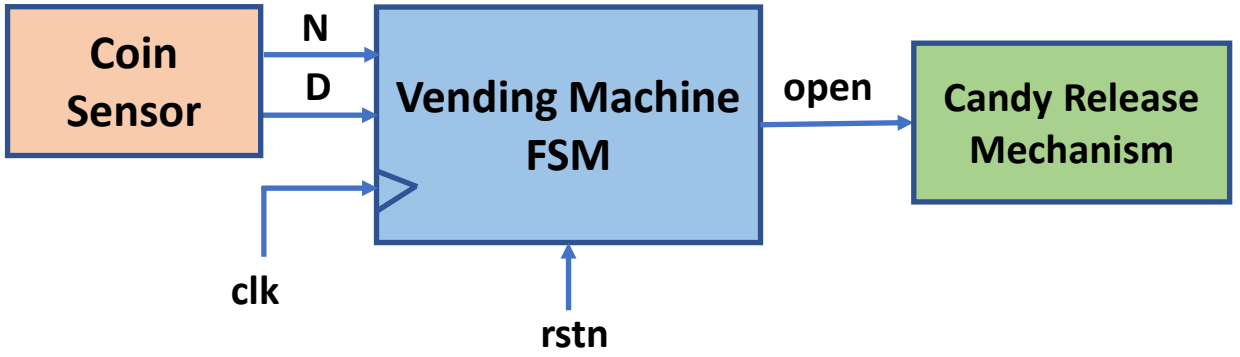
ECE-111 Advanced Digital Design Project

Vishal Karna

UC San Diego
JACOBS SCHOOL OF ENGINEERING
Electrical and Computer Engineering

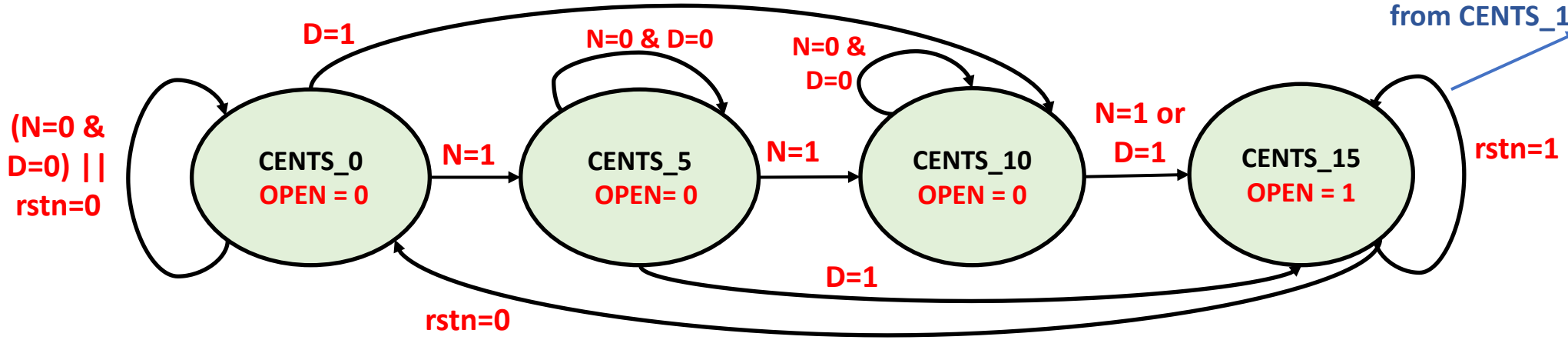
Homework-6a : Vending Machine

- **Design FSM for Vending Machine with following mentioned requirements:**
  - Dispatches a pack of candy upon deposit of 15 cents by the customer
  - Has single coin slot and only accepts nickels and dimes. Doesn't take pennies or quarters
  - Does not provide any change back to the customer in case more than 15 cents are deposited
  - Once candy is dispatched, vending machine should move to reset state



This Vending machine requires manual reset to go back to CENTS_0 state after candy is dispatched. Hence rstn=0 is required for FSM to transition from CENTS_15 to CENTS_0

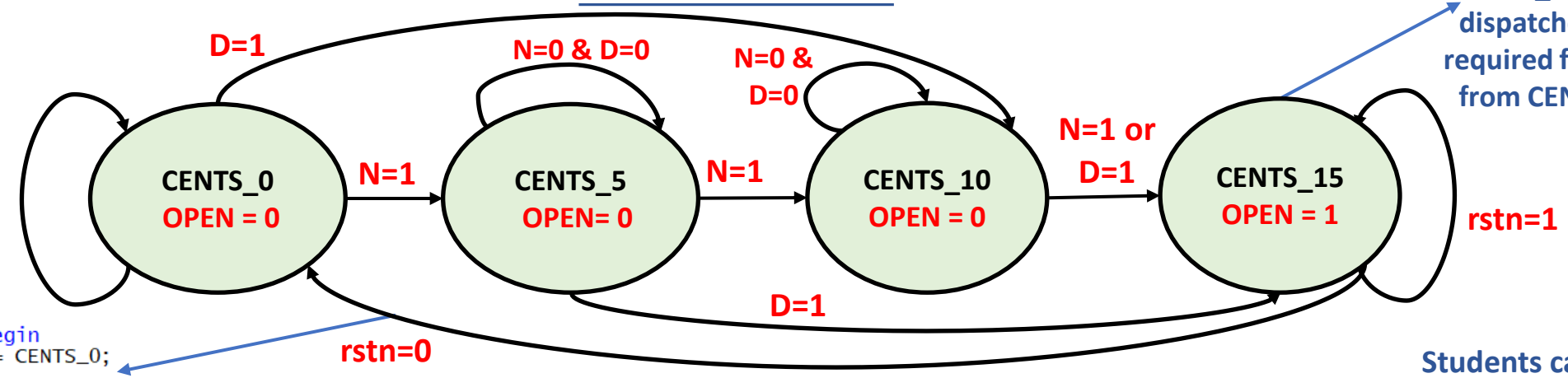
Vending Machine Moore State Transition Diagram



Homework-6a : Vending Machine

- **Vending Machine FSM supports Manual Reset, however FSM with Auto Reset can also be implemented**
  - **Manual Reset FSM :** Requires reset to be asserted (rstn = 0) for FSM to go back to CENTS_0 state from CENTS_15 state once candy is dispatched
  - **Auto Reset FSM** : Without reset (rstn = 0), FSM transitions from CENTS_15 to CENTS_0

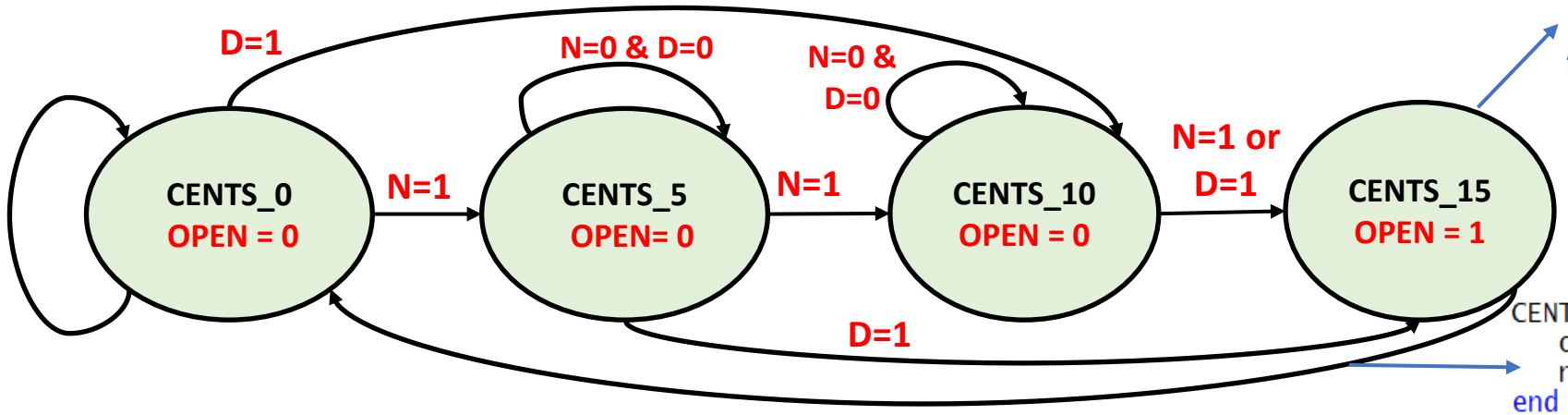
With Manual Reset



This Vending machine requires manual reset to go back to CENTS_0 state after candy is dispatched. Hence rstn=0 is required for FSM to transition from CENTS_15 to CENTS_0

```
CENTS_15: begin
   open = 1;
   if(rstn==0) begin
     next_state = CENTS_0;
   end
   else begin
     next_state = CENTS_15;
   end
end
```

With Auto Reset (Alternate Implementation)



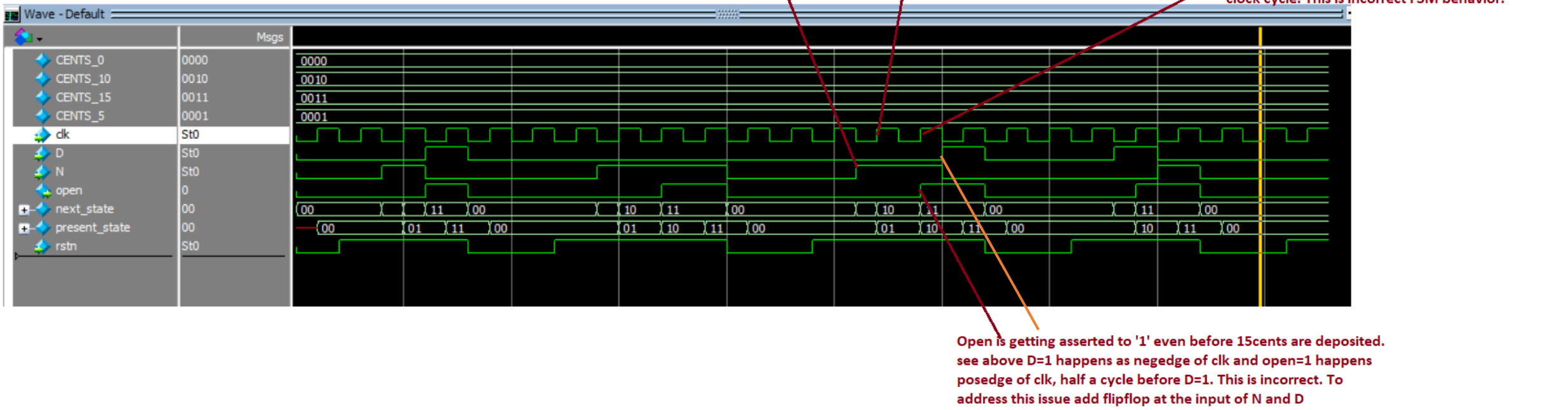
Students can implement FSM with Manual reset or Auto Reset. Auto Reset means that After entering CENTS_15 state, FSM will transition unconditionally without any reset, to CENTA_0 state

```
CENTS_15: begin
   open = 1;
   next_state = CENTS_0;
end
```

Homework-6a : Vending Machine

- **In mealy FSM register input's N and D if output "open" gets asserted to '1' before depositing 15 Cents (register means add flipflop)**
  - When implementing mealy FSM for Vending Machine, add a flipflop at the input of 'N' and 'D'
  - And these registered 'r_N' and 'r_D' should be used in always_comb next state logic procedural blocks

Without Registering N and D, See Potential Waveform Snapshot Below with pre-mature open = 1

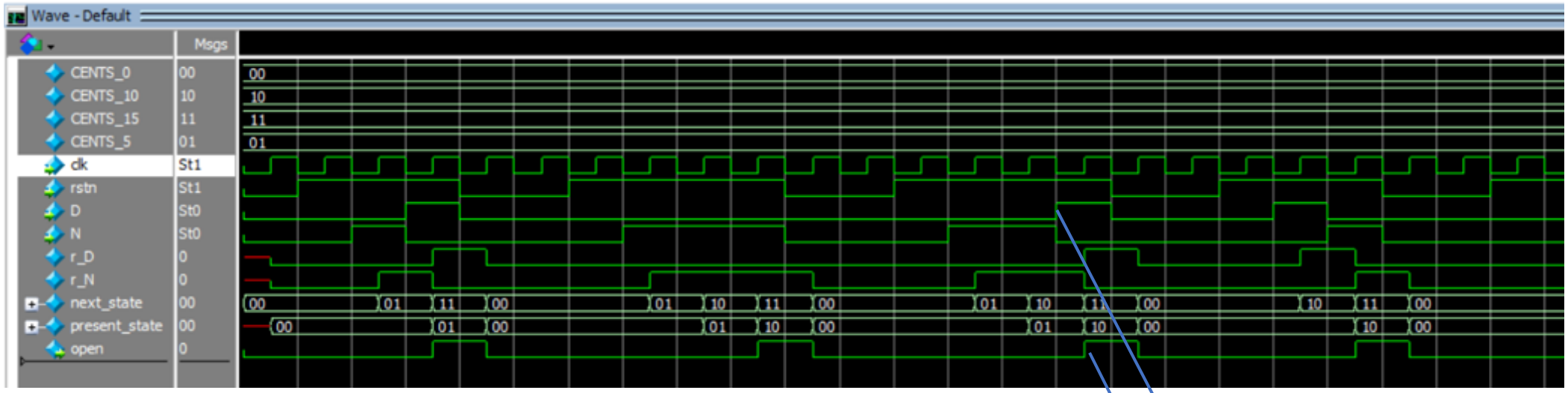


Homework-6a : Vending Machine

❑ Mealy FSM Waveform snapshot with after adding flipflop at the inputs 'N' and 'D'

- 'r_N' and 'r_D' signals in waveform snapshot are output of flipflops for inputs 'N' and 'D'
- In Moore FSM for Vending Machine, inputs N and D does not require flipflop, as output "open" is '1' only when FSM reaches CENTS_15 state (i.e. only after 15 cents or more are deposited)
- In Mealy FSM for Vending Machine, as the output "open" can be '1' in when transiting from CENTS_5 to CENTS_15 or from CENTS_10 to CENTS_15, and since inputs N and D can be changed to '1' in middle of clock cycle, adding flipflop at the inputs N and M can prevent open be go '1' pre-maturely from FSM

Mealy FSM Waveform with flipflops added for Inputs 'N' and 'M'




```
// Local Variables
logic r_N, r_D;

// Sequential Logic for present state
always_ff@(posedge clk) begin
 if(!rstn) begin
   present_state <= CENTS_0;
   r_N <= 0;
   r_D <= 0;
 end
 else begin
   r_N <= N; // register N as it can change in anytime between clock
   r_D <= D; // register D as it can change in anytime between clock
   present_state <= next_state;
 end
end
```

SystemVerilog Code demonstrating how to add flipflop at the inputs 'N' and 'D'

"open" is correctly asserted to '1' only after Nickel (N=1) and Dime (D=1) are inserted in Vending Machine.

Homework-6a : Vending Machine

❑ Mealy FSM for Vending Machine Can have 1 less state – opportunity of state reduction

- Output open = 1 is already being set when transitioning from CENTS_5 and CENTS_10 states to CENTS_15 state and in CENTS_15 state FSM machine just waits for reset condition to transition back to CENTS_0 state
  - Hence CENTS_15 state is redundant and not necessary to have in case of Mealy Machine

- **Note :** Either of the Mealy FSM design for Vending Machine (i.e. with or without CENTS_15 state) will be accepted as homework submission

Is this CENTS_15 state for Mealy State Machine Required ?
– Answer : No

```
CENTS_15: begin
    if(rstn==0) begin
      next_state = CENTS_0;
      open = 0;
    end
    else begin
      next_state = CENTS_15;
      open = 1;
    end
end
```

Move rstn==0 logic from CENTS_15 to CENTS_10 state and if N=1 or D=1 in CENTS_10 state then continue to have FSM state in CENTS_10 state. And if rstn==0 in CENTS_10 state then move to CENTS_0 state

Homework-6a : Vending Machine

- **For Vending Machine develop SystemVerilog code for More and Mealy FSM**
  - Use binary encoding for state variables
  - Review Vending state transition table and Moore FSM state transition diagram for FSM code development
  - Design state transition diagram for mealy implementation
  - Synthesize and review RTL netlist schematic, state machine viewer and resource usage
  - Simulate both Moore and Mealy implementation using testbench provided and Review waveforms
  - Design top SystemVerilog module name should be vending_machine_moore and vending_machine_mealy

Vending Machine State Transition Table (For Moore and Mealy)

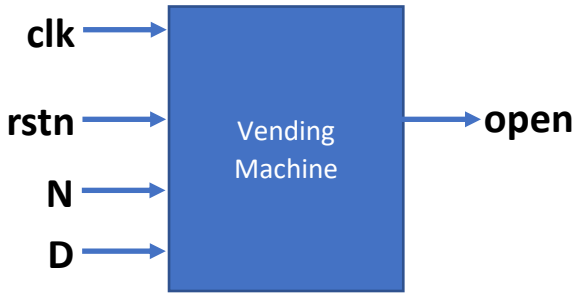
inputs						outputs				
state[3]	state[2]	state[1]	state[0]	D	N	next[3]	next[2]	next[1]	next[0]	open
0	0	0	1	0	0	0	0	0	1	0
				0	1	0	0	1	0	0
				1	0	0	1	0	0	0
				1	1	x	x	x	x	x
0	0	1	0	0	0	0	0	1	0	0
				0	1	0	1	0	0	0
				1	0	1	0	0	0	0
				1	1	x	x	x	x	x
0	1	0	0	0	0	0	1	0	0	0
				0	1	1	0	0	0	0
				1	0	1	0	0	0	0
				1	1	x	x	x	x	x
1	0	0	0	x	x	1	0	0	0	1

Next State and Output Function

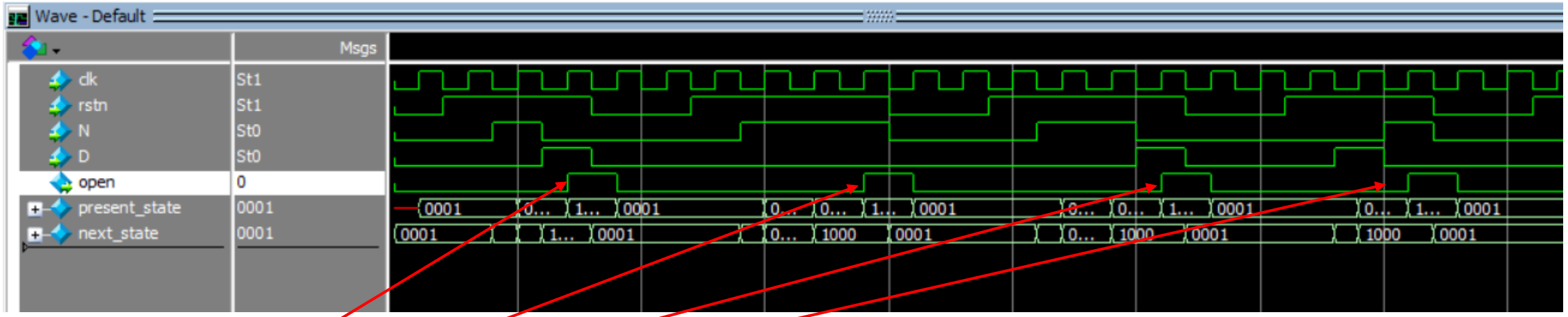
next[0] = state[0].D'.N'
next[1] = state[0].N + state[1].D'.N'
next[2] = state[0].D + state[1].N + state[2].D'.N'
next[3] = state[1].D + state[2].D + state[2].N + state[3]
open = Q3

Homework-6a : Vending Machine

- **Assume below mentioned primary Ports for Vending Machine**
  - **Input clk :** posedge clock
  - **Input rstn :** reset should be synchronous negedge reset
  - **Input N, D :** 1-bit Nickel and Dime inputs indicating Nickel and Dime are deposited if values are '1'
  - **Output open :** 1-bit open signal indicating vending machine is dispatching candy



[Reference Moore FSM Simulation Waveform]



Output “open” is asserted once 15 cents or more are deposited

Homework-6a : Vending Machine

- **Report should include :**
  - SystemVerilog design code for both Moore and Mealy FSM
  - Synthesis resource usage and schematic generated from RTL netlist viewer
  - Simulation snapshot and explain simulation result for both Moore and Mealy FSM
  - FSM state transition diagram for both Moore and Mealy implementations
  - Explanation of FPGA resource usage in report is not required.
  - **Note :**
    - State transition diagram needs to be submitted in report and it can be either hand drawn with picture taken and pasted in report or it could be drawn in word or powerpoint or auto-generated sate machine diagram from Quartus prime is also acceptable. If submitting auto-generated Quartus generated state machine diagram then also attach state transition table generated from Quartus prime.
    - For Moore FSM diagram from this requirements document can also be used
    - Simulation transcript is **not** required in report since there are no prints from testbench

- **Lab6 folder includes :**
  - Template design code for Vending machine Moore and Mealy FSM are provided
  - Full testbench code for both Moore and Mealy FSM are provided
  - For learning purpose, student can change the stimulus in initial block in testbench file.