Lab 9

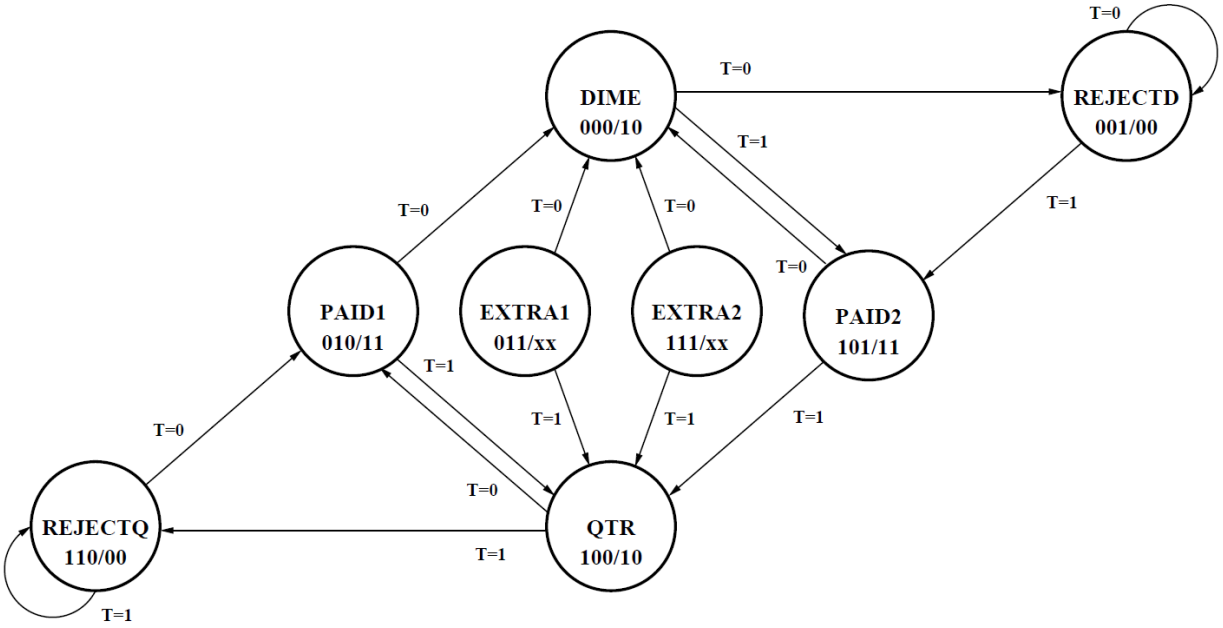
Lab 9 assignment is due on Friday, April 17, by 9pm in Blackboard.

Please ask all questions about this assignment during the office hours or post questions on piazza.

Read **Prof. Lumetta's lecture notes set 3.3: Design of the Finite State Machine for the Lab** for a detailed description of the FSM used in this lab. The notes also provide K-maps for next-state logic you need to implement in this lab.

Sequential logic design

In this lab, we will complete the design of the vending machine example in Altera Quartus. In Lab 7, you designed a combinational logic circuit that encoded the states of a finite state machine into two output signals "**A**ccept coin (**A**)" and "dispense **P**roduct (**P**)." In this lab, you will design an implementation of the state machine developed in Lecture Notes Set 3.3 and shown below.

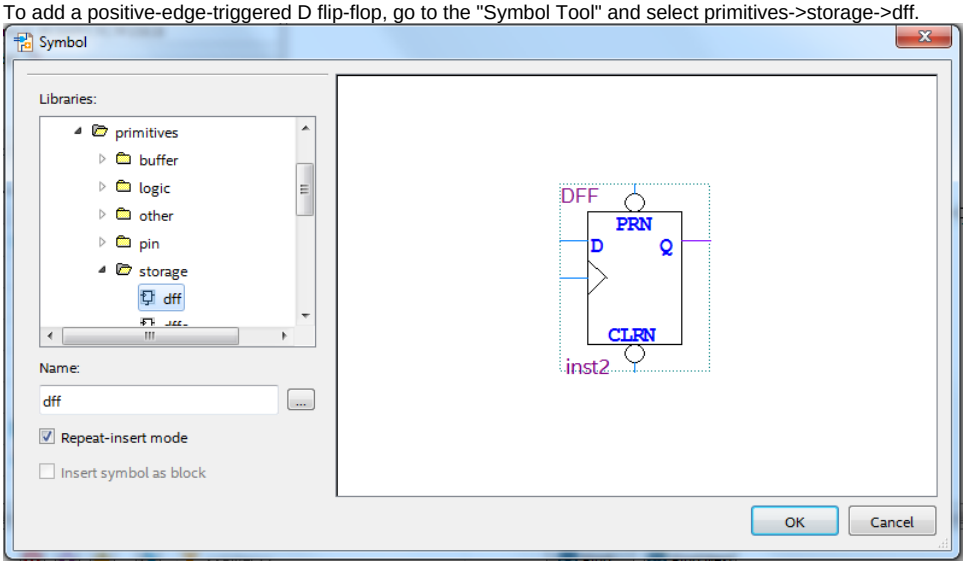


Here each state is encoded using three state variables ($S_2S_1S_0$) and the outputs are encoded using the output variables (**AP**). For example, in the "Reject Q" state, $S_2S_1S_0$ = 110 and **AP** = 00. The state transitions are labeled with the variable **T**. **T** = 1 when a quarter has been inserted and **T** = 0 when a dime has been inserted. This state machine is also driven by a clock that indicates when a coin has been inserted.

Implementing the State Machine in Altera Quartus

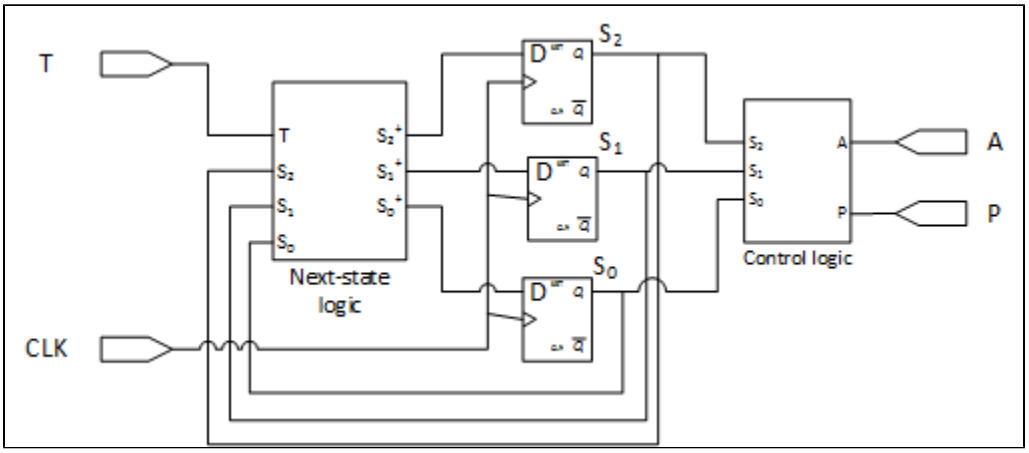
- Design the next-state logic for your circuit on paper using the techniques you have learned in class. **Design your circuit so that it uses NAND /NOR gates (inverters are OK too) and positive-edge-triggered D flip-flops.**
- Open Altera Quartus and open your lab7 project. Building this circuit in your lab7 project will allow you to reuse your circuit from lab7.
- Create the schematic that implements the next-state logic and state storage (3 D-flip flops). Pay attention to the signal names.

- To add a positive-edge-triggered D flip-flop, go to the "Symbol Tool" and select primitives->storage->dff.



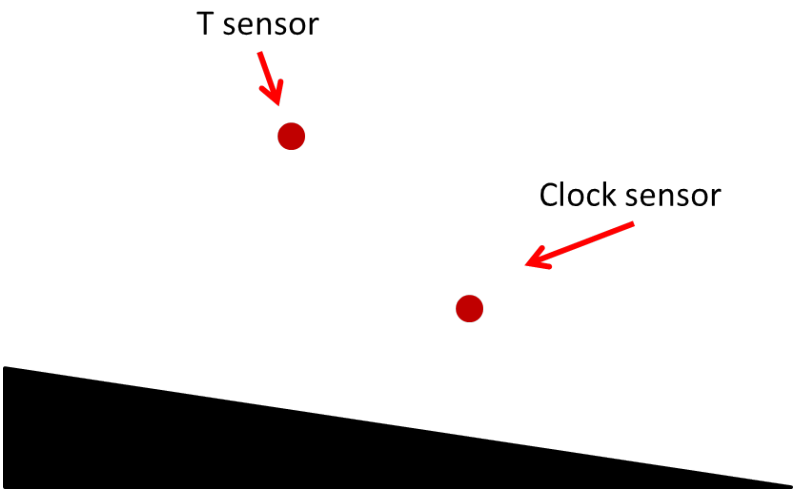
  - You should notice that the D flip-flop (picture above) has more inputs than we normally use. These additional inputs are asynchronous inputs and they alter the behavior of the D flip-flop. These asynchronous inputs allow us to set the initial state of our circuit which is handy if we just want to simulate a specific state transition.
    - The **clk** (shown as triangle) input is the clock.
    - The **D** input is the data input for the flip-flop.
    - The **Q** output is the stored state of the flip-flop.
    - **CLRN** is asynchronous active-low clear input. Clear means the flip-flop will store 0.
      - When **CLRN** is 0, the D flip-flop stores a 0 regardless of the values of any other inputs.
    - **PRN** is asynchronous active-low preset input. Preset means the flip-flop will store 1.
      - When **PRN** is 0, the D flip-flop stores a 1 regardless of the values of any other inputs.
    - **CLRN** and **PRN** cannot be 0 (low) at the same time!
    - When **CLRN** and **PRN** are both 1 (high), the flip-flop will store a value supplied on **D** input port on the next rising edge of the clock.
  - You should also notice that the above flip-flop does not have inverted output. You should invert Q yourself is such a value is needed.
- Add two more flip-flops to your design so that you can store each of the three state variables ($\mathbf{S_2S_1S_0}$). Make sure that the outputs of your flip-flops are named $\mathbf{S_2}$, $\mathbf{S_1}$, and $\mathbf{S_0}$ after you are done connecting wires.
- Add two input ports to your design. Connect one input port to all of the **CLRN** inputs to create the **rst** (reset) signal, and connect the other input port to all of the **PRN** inputs to create the **set** signal.
- Add another input port and connect it to all of the **clk** inputs to create your **clk** signal.
- Add one more input port for your **T** input from the vending machine hardware.
- Add the next state logic you designed to your circuit from Lab 7 to your schematic to complete your design.

Your final design should resemble this circuit (note that **rst** and **set** signals are omitted):

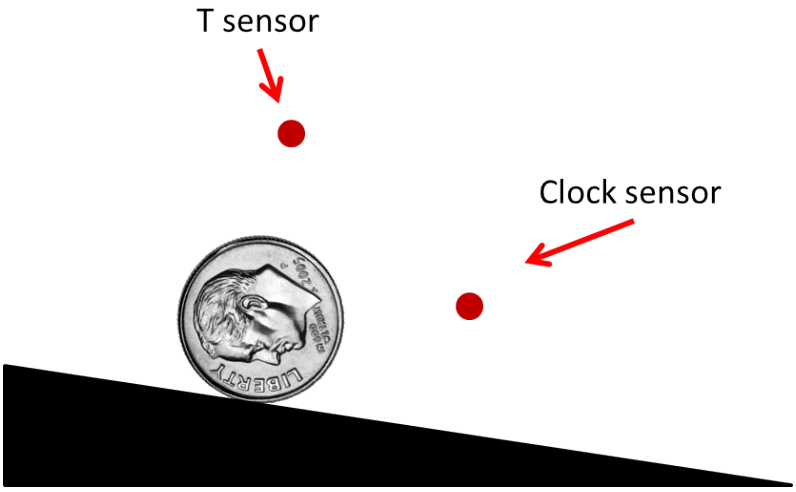


Simulating the State Machine

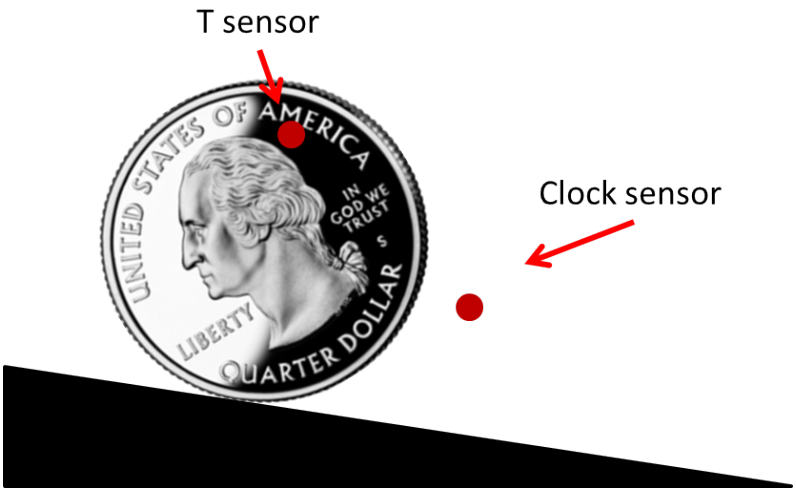
In order to simulate your state machine, you need to better understand how the physical system of the vending machine works, so you can abstract its functionality in the Quartus simulator. The vending machine accepts coins by letting them roll down a ramp. Along the ramp are a set of sensors. Each sensor sends a 0 when no coin is in front of the sensor. So as long as no coin is inserted, both the T sensor and clock sensor send out 0s.



When a dime is inserted, it will roll under the T sensor (without blocking it) and then briefly roll in front of the clock sensor creating a clock pulse (the clock rises from 0 to 1 and then returns from 1 back to 0). Consequently, the coin slot will send a clock pulse signal that will update the state stored in the flip-flops while input T=0.



When a quarter is inserted, it will roll and block the T sensor creating a 1, and then moments later it will briefly and simultaneously block the clock sensor creating a clock pulse. Consequently, the coin slot will send a clock pulse signal that will update the state stored in the flip-flops while input T=1.



To simulate this behavior, we will need to alter input signals as described below:

- To simulate a coin being inserted, we will assume that it takes 10 ns for a coin to roll past a sensor. We will also assume that the quarter will roll in front of the T sensor 5 ns before it rolls in front of the clock sensor. For example, if a quarter is inserted at 20 ns, the T sensor will become a 1 at 20 ns and will return to a 0 at 30 ns. Similarly, the clock sensor will become 1 at 25 ns and will return to a 0 at 35 ns for the same quarter.
- In the waveform editor, initialize the values of your flip-flops to state "extra2" by forcing **set** to 0 and **rst** to 1 the first 5 nanoseconds (ns) of your simulation. You will also want to initialize **T** and **clk** to 0 to start. At 5 ns, force **set** to 1.
- Next, modify your waveform to simulate the following sequence of coins (assuming a very fast human who is apparently very thirsty or hungry)
    - Quarter inserted at 20 ns.
    - Quarter inserted at 40 ns.
    - Quarter inserted at 60 ns.
    - Dime inserted at 80 ns.
    - Dime inserted at 100 ns.
    - Dime inserted at 120 ns.
    - Quarter inserted at 140 ns.
    - Dime inserted at 160 ns.
    - Quarter inserted at 180 ns.
- Add at least the **T**, **clk**, $\mathbf{S_2}$, $\mathbf{S_1}$, $\mathbf{S_0}$, **A**, and **P** signals to your waveform so you can debug.
    - Flip-flop outputs $\mathbf{S_2}$, $\mathbf{S_1}$, and $\mathbf{S_0}$, should be connected to output ports; this way they will become visible in the simulator.
- Run your simulation for 200 ns, and check whether your circuit demonstrates the desired behavior. If it does not, debug and resimulate your circuit.

What to Turn In

- Boolean Expressions for S_2^+, S_1^+, S_0^+ that correspond to the K-maps given in Prof. Lumetta's lecture notes set 3.3.
- Printout of your Altera Quartus circuit schematic.
- Printout of your simulated waveform.