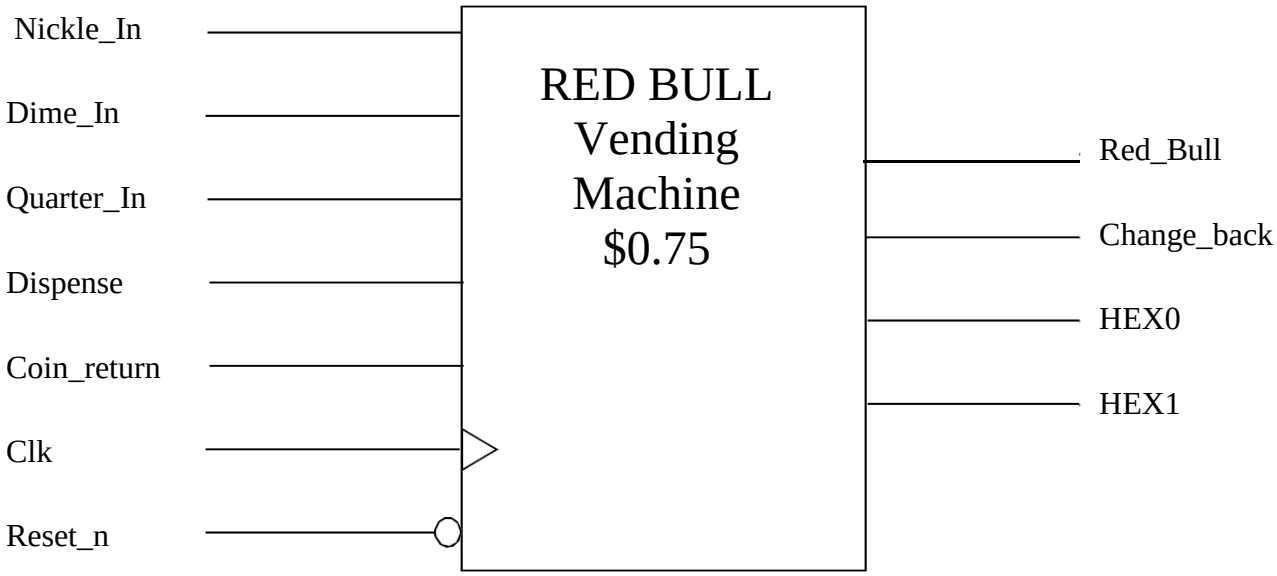
Technical Objective:
A state machine is sequential circuit that stores the status of something at a given time and can operate on inputs to change the status and/or cause an action or output to take place for any given change. In this lab, the design, coding and troubleshooting of a finite state machine will be investigated. To test the theory of designing with a state machine, a circuit to control the operations of a vending machine will be implemented.

Prelab (30%):

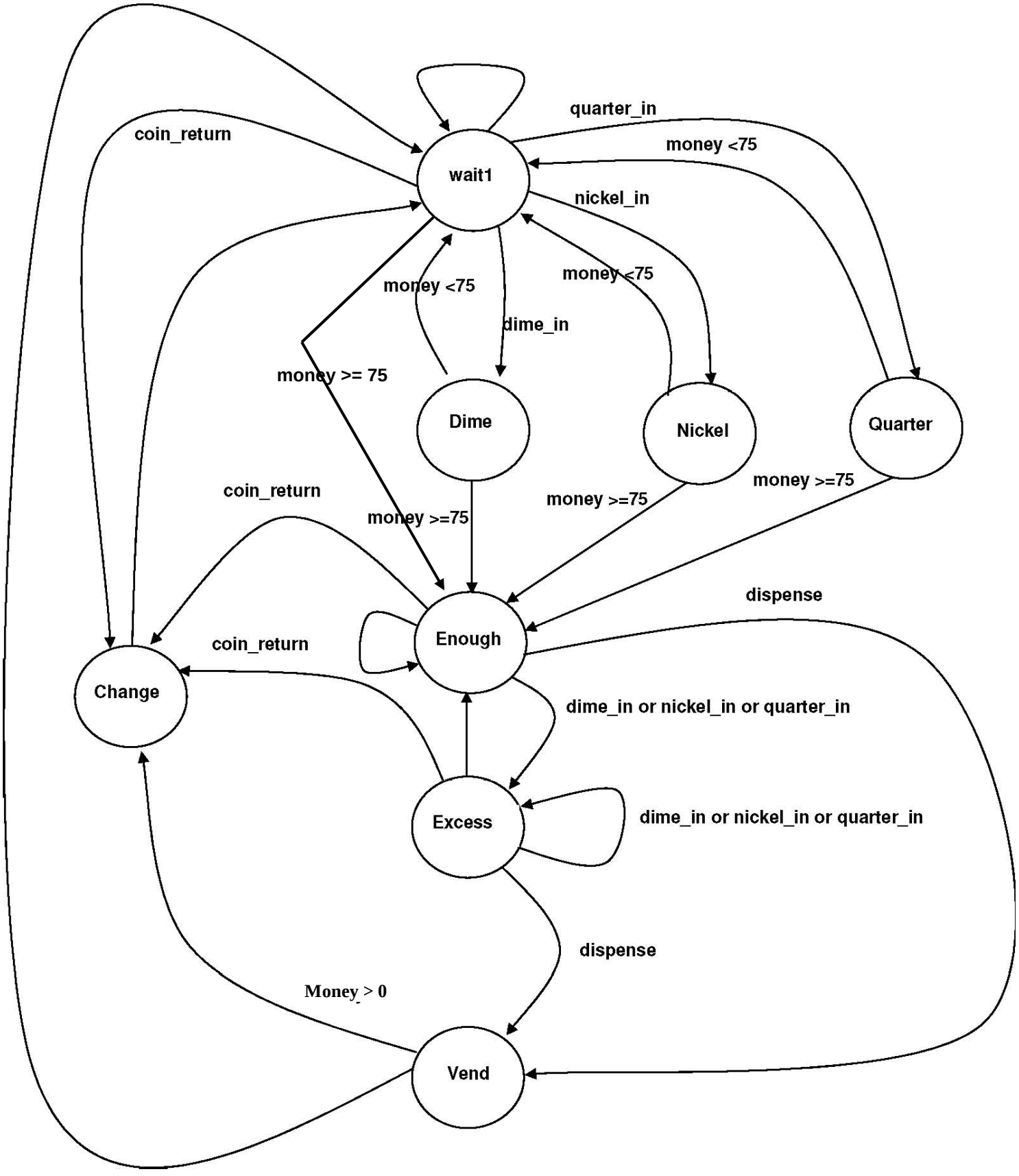
1. Design a Vending machine controller that dispenses Red Bull drinks for $0.75. The specifications are as follows.
   - The machine accepts dimes, nickels and quarters only. Only one coin can be input at a time.
   - If the dispense button is pushed before there has been $0.75 input, it is ignored. If the button is pushed after $0.75 has been deposited, the machine outputs the drink and any change, if applicable.
   - Once the total money in the machine reaches $0.75 or more, any further deposits are immediately returned to the user.
   - If the coin_return input is pressed at any time, all money is returned.
   - Two seven segment displays will always display the current balance in the machine.
   - The inputs and outputs are as follows:



- Dime_In, Nickel_in, Quarter_in: these indicate that money is being deposited. These should be connected to three slider switches on the DE2 board.
- Coin_return: This input indicates that the user would like their money back. Connect it to a slider switch.
- Dispense: This input indicates that the user would like their drink. This should also be connected to a slider switch.
- Clk: Since the 50 MHz clock on the DE2 board is much too fast for this application, there are two choices.
  - Connect clock to a pushbutton and manually clock the state machine
  - Create a much slower clock in the design and clock the state machine at a ~1 sec. rate.
- Reset_n: this is the system reset and should be connected to Key0
- Change_back: this output indicates that change is being returned. It should be a red LED.
- Red Bull: This output indicates that the drink has been dispensed. It should be a red LED.
- HEX0 and HEX1: These are two seven segment displays. They display the amount of money in the machine.

- Design Suggestions
  - Make an individual process for each output. Do not assign the outputs in your next_state logic process. The outputs must be a function of the state.
  - You will need a separate process to keep track of the current money in the machine. This should be a clocked process since there is memory involved. Do not put this in the next_state logic process either. Changes to the money variable are determined by the state as indicated in the table.

State	Money	Red_Bull	Change_back
wait1	money = money	0	0
dime	money = money + 10	0	0
nickel	money = money + 5	0	0
quarter	money = money +25	0	0
enough	money = money	0	0
excess	money = money	0	1
vend	money = money – 75	1	0
change	money = 0	0	1


wait1
quarter_in
money <75
nickel_in
coin_return
money <75
money <75
dime_in
money >= 75
Dime
Nickel
Quarter
coin_return
money >=75
money >=75
money >=75
dispense
Enough
coin_return
Change
dime_in or nickel_in or quarter_in
Excess
dime_in or nickel_in or quarter_in
dispense
Money > 0
Vend

2. Write the VHDL for your vending machine controller. Synthesize the design and correct any errors or warnings. Do not ignore latch warnings. Open the state machine in the netlist viewer and verify that the state transition diagram matches your original.
3. Submit the following to the dropbox prior to your lab section:
    a. The VHDL code
    b. The state transition diagram generated in the netlist viewer. To generate the state transition diagram, choose Tools > Netlist Viewers > State Machine Viewer

Procedure (60%):

1. Synthesize your design using Altera Quartus II.
2. Using the testbench provided, simulate your design using Modelsim. There are comments in Modelsim to help you understand what outputs you should be seeing. Be sure to run the simulation until the end.
3. Obtain a signoff for the simulation.
4. Download your design to the DE2 board and verify its operation.
5. Obtain a signoff for the working design.

Documentation (10%):

1. Include a summary (~ 1 page) describing the purpose, methodology and results of the lab
2. Include fully commented VHDL
3. Include fully commented Modelsim waveforms
4. Analyze the compilation results
    a. Based on the number of registers used (found in the compilation report), what type of encoding did Quartus default to? Don't forget to account for the registers needed for the money counter. Also note in the logic utilization the number of ALMs used.
    b. In Quartus click on Assignments > Settings. Choose 'Compiler Settings' and then click on 'Advanced Settings (Synthesis). Scroll down the list and select 'State Machine Processing' and change the value to Gray. Recompile the design.
    c. Compare the results of the gray encoding to the default encoding in terms of registers and ALMs. Report on your findings.

Signoffs and Grade:

Name:_____

Component	Signoff	Date	Time
Functional Simulation (30 pts)			
Working Board (30 pts)			

==========

Component	Received	Possible
Prelab		30
Signoff		60
Report		10
Penalties • after the first 15 minutes of lab session 11: -10 • after the first 15 minutes of lab session 12: -25	-	
Total		100