

CSE 460

VLSI LAB

ASSIGNMENT 04

**Submitted by**

Md. Nasimuzzaman

Id: 19101051

CSE-460 Fall-2022

Submission Date: 16th November 2022

**Lab Assignment 04**

**Problem statement**

**Problem 1:**

You have to design a vending machine in Quartus for a **15 Tk product**. User’s money, returned money by the machine, and product bought condition is represented as **cash\_in (2-bit input), chg (output), and buy (1-bit output)** respectively.

The vending machine can only accept **three** inputs: no money (**cash\_in** = 00), Tk 10 (**cash\_in** = 01), and Tk 20 (**cash\_in** = 10). Once an acceptable input is more than or equal to 20 Tk, the machine immediately generates an output (**buy=1**),  goes back to the initial state, and gives back the change (if required).

Requirements:

1. Draw the state diagram.
2. How many types of changes (return) will the machine produce? How many bit/bits should **chg** output have to represent the returned money in the code?
3. Write the state-assigned table.
4. Write the Verilog code.
5. Run the simulation, and verify your answer. Add two ss (full screen and zoomed). Describe it briefly.

**Code**

module VM(clk, reset, cash\_in, cash\_return, purchase, current\_state, next\_state);

input clk, reset;

input [1:0] cash\_in;

output reg purchase;

output reg [1:0] cash\_return;

output reg [1:0]current\_state, next\_state;

parameter s0 = 2'b00, s1 = 2'b01;

parameter [1:0] in\_0tk = 2'b00,

in\_10tk = 2'b01,

in\_20tk = 2'b10;

parameter [1:0] ret\_0tk = 2'b00,

ret\_5tk = 2'b01,

ret\_15tk = 2'b10;

always @(posedge clk, posedge reset)

begin

if (reset == 1)

begin

next\_state = s0;

current\_state = s0;

purchase = 0;

cash\_return = ret\_0tk;

end

else

begin

current\_state = next\_state;

case(current\_state)

s0:

begin

if(cash\_in == in\_0tk)

begin

next\_state =s0;

purchase = 0;

cash\_return = ret\_0tk;

end

else if(cash\_in == in\_10tk)

begin

next\_state =s1;

purchase = 0;

cash\_return = ret\_0tk;

end

else if (cash\_in == in\_20tk)

begin

next\_state =s0;

purchase = 1;

cash\_return = ret\_5tk;

end

end

s1:

begin

if(cash\_in == in\_0tk)

begin

next\_state =s0;

purchase = 0;

cash\_return = ret\_0tk;

end

else if(cash\_in == in\_10tk)

begin

next\_state =s0;

purchase = 1;

cash\_return = ret\_5tk;

end

else if (cash\_in == in\_20tk)

begin

next\_state =s0;

purchase = 1;

cash\_return = ret\_15tk;

end

end

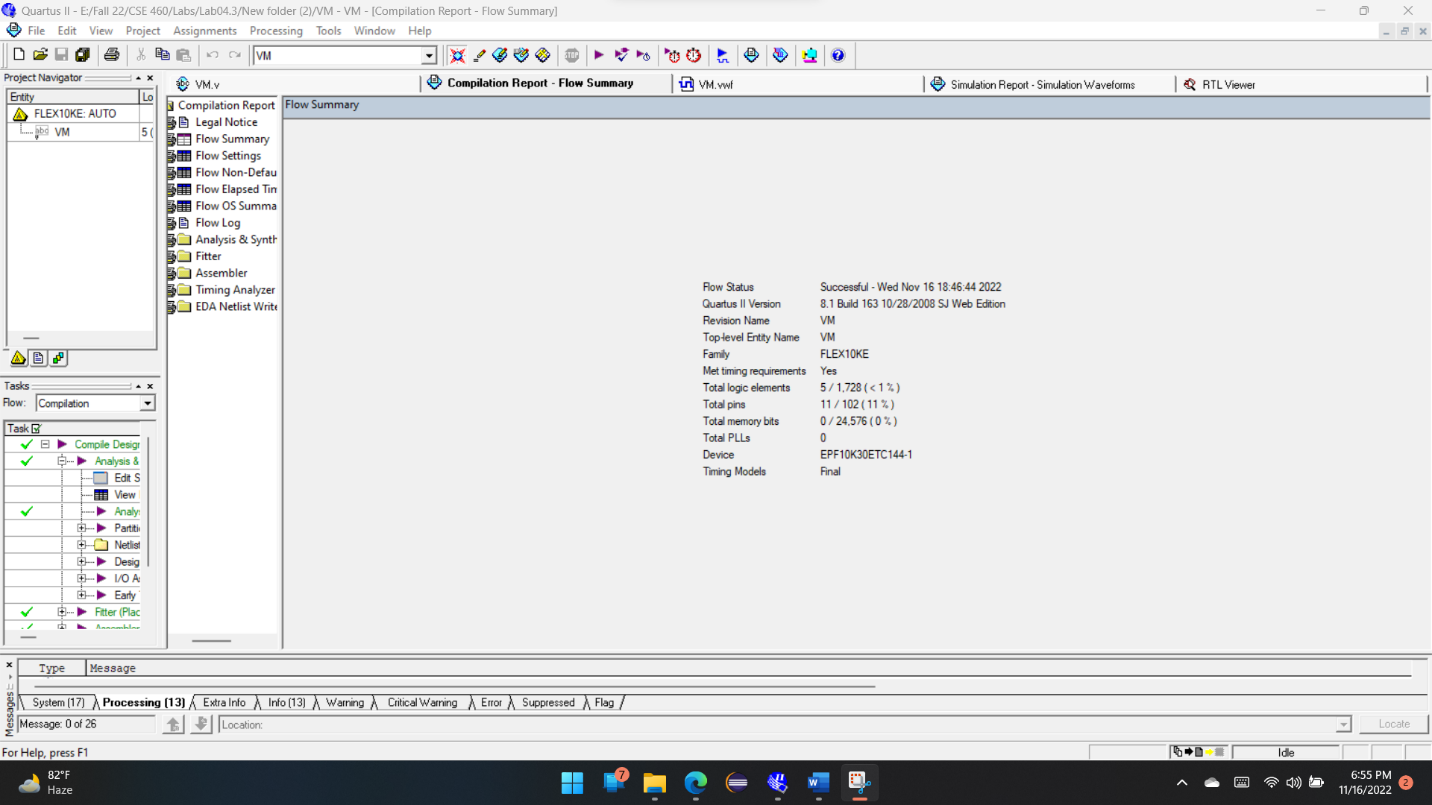
endcase

end

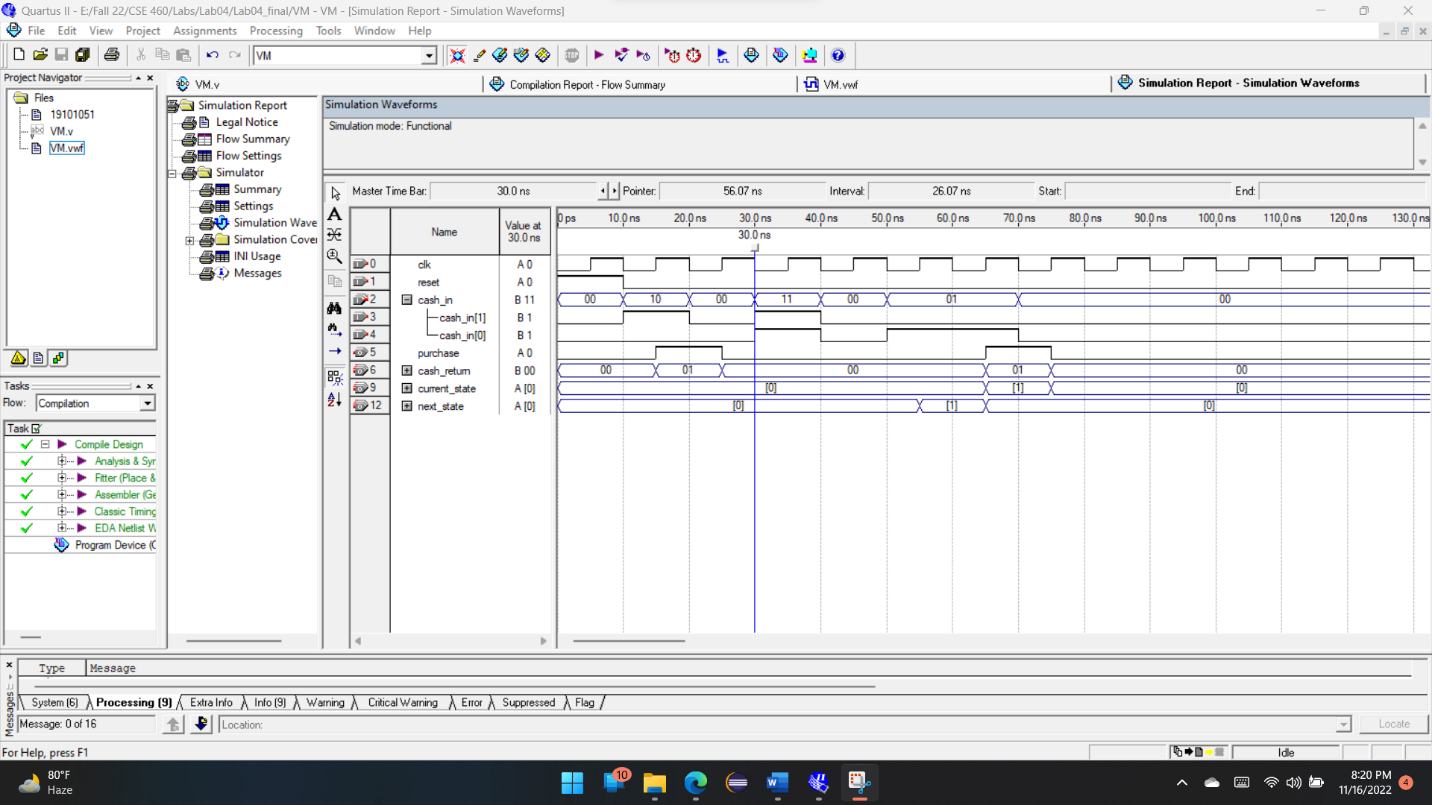
end

endmodule

**Compilation Report**



**Simulation Report**



**State Diagram**

Cash\_in = 0

Cash\_in = 20

Cash\_in = 10

S0

S1

Cash\_in = 0

Cash\_in = 10

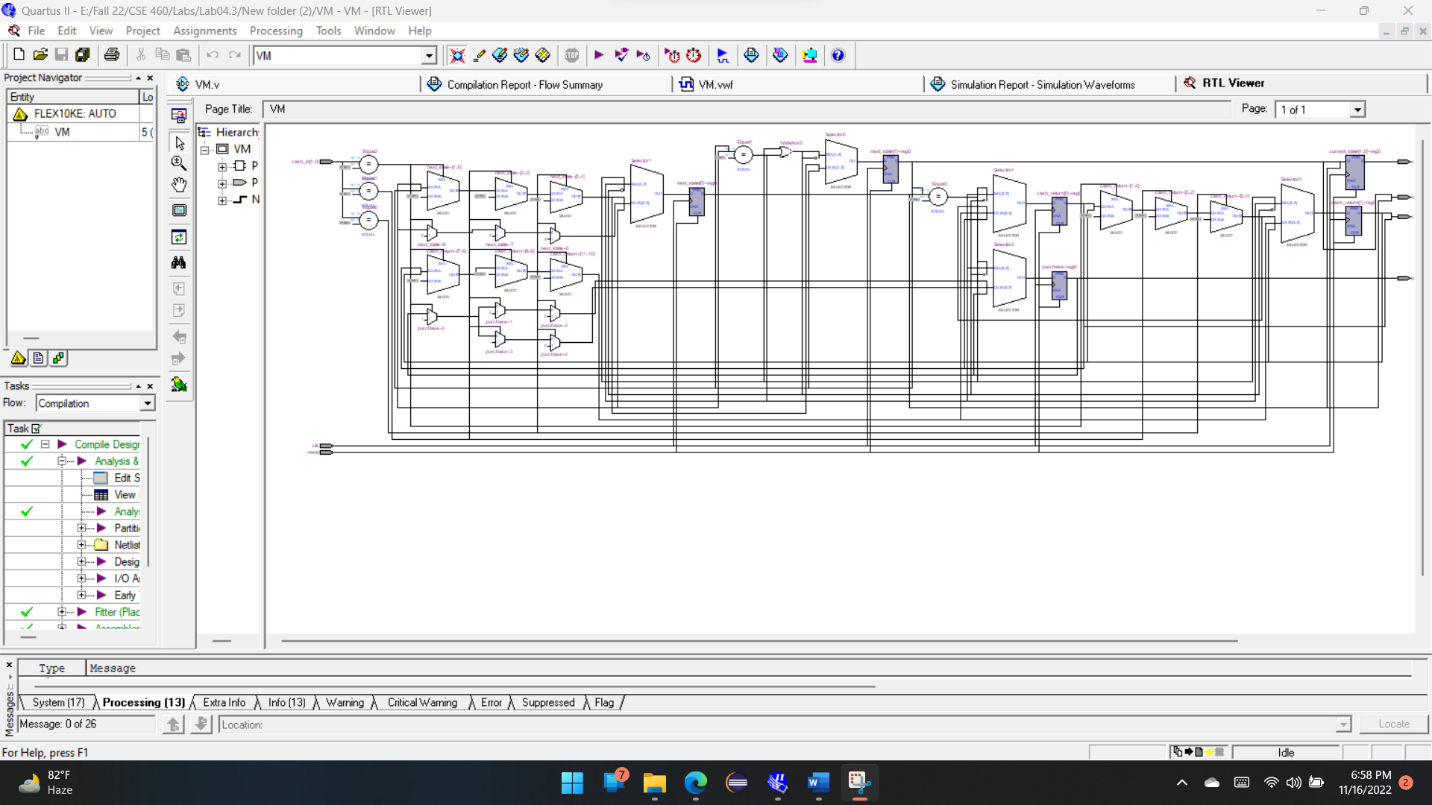
Cash\_in = 20

**State Assign Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| At state S0 |  |  |  |  |
| Cash\_in | Next\_state | Purchase | Cash\_returen |  |
| 0 | S0 | 0 | 0 |  |
| 10 | S1 | 0 | 0 |  |
| 20 | S0 | 1 | 5 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| At state S1 |  |  |  |  |
| Cash\_in | Next\_state | Purchase | Cash\_returen |  |
| 0 | S0 | 0 | 0 |  |
| 10 | S0 | 1 | 5 |  |
| 20 | S0 | 1 | 15 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**RTL View**

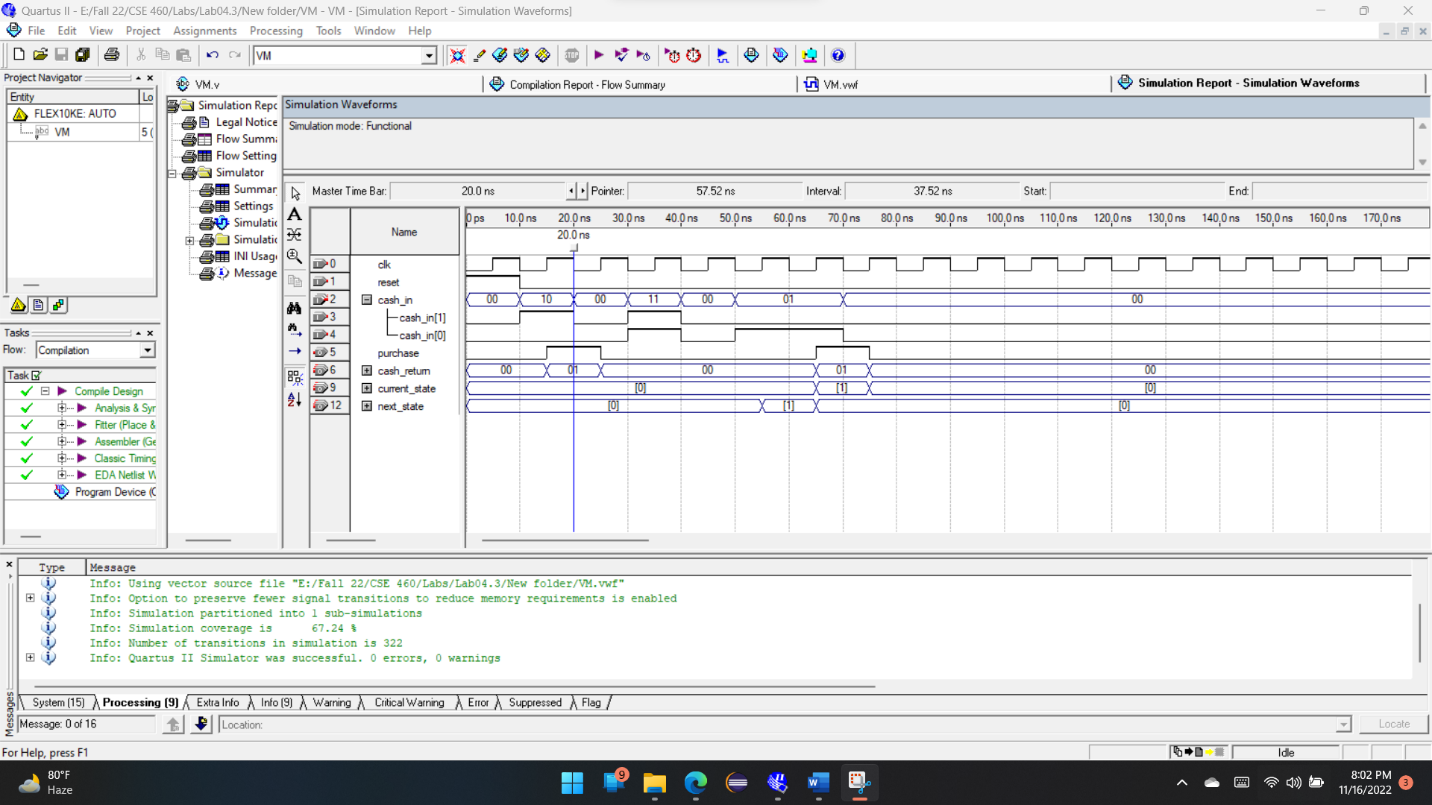
****

**Explanation**

In this task, I’ve simulated a Vending Machine, which accepts three inputs, no money (Cash\_in = 00), Tk 10(Cash\_in = 01), and Tk 20 (Cash\_in = 10). The vending machine will purchase a product (cost 15Tk) only if the input is >= 20 Tk. Once this condition meets then the machine generates an output (buy/purchase = 1), then return back to the initial state and gives back the change if needed. In this case I’ve assigned the inputs, 0Tk as 00, 10 Tk as 01 and 20 Tk as 10. Also, the changes assigned 0Tk as 00, 5Tk as 01 and 15Tk as 10.

Now, we have two states here which are S0 and S1. If we analyze the whole scenario, when our input is 0tk in state S0, the machine remains in the initial state and the purchase is 0 also cash\_back is 0. Then if we give 10tk input in state S0 it will go to the next state which is S1 but still purchase will be 0 and cash\_back will be 0 as, our product value is 15tk and our input is still less than 20tk. After that if we give 20tk as an input in state S0, it will remain in the present state S0 and as our condition now meets, purchase will be 1 and there will be cash\_back of 5tk.

Now, in the next state (S1) if we give 0tk as an input it will move back to the previous state(S0) also purchase will be 0 and there will be no cash\_back. Again, if we give 10tk as an input in S1 there will be 1 purchase as we have 10tk previously which means now we have 20tk in S2. So, purchase will be 1 and also, we get cash\_back of 5tk. Lastly when we give 20tk as an input in S1, we have previously 10tk in S2, now it becomes 30tk. So, purchase will happen and we will get 15tk as cash\_back.



If we look at the simulation report, 10ns to 20ns input is 10 which means 20tk and it meets our condition (input>=20) so we get the output(purchase) high in this part. Also, we get cash\_bask 01 which was denoted for 5tk and it works perfectly.