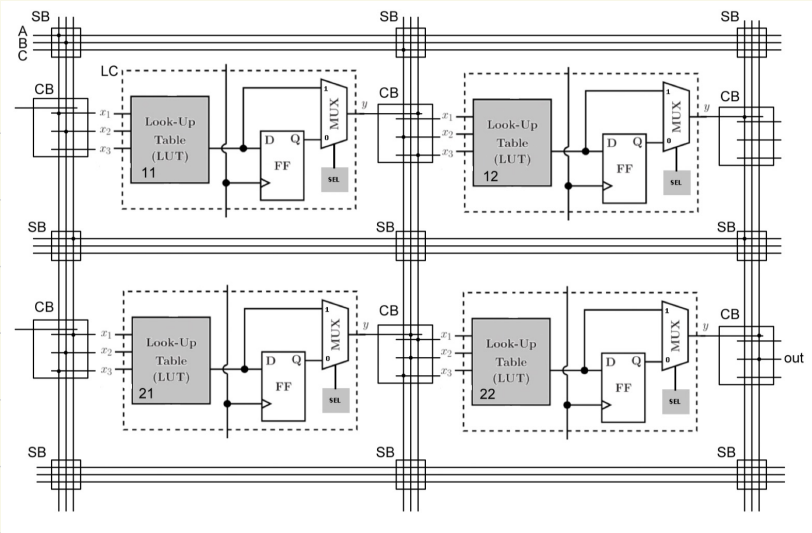
کسری امانی - 98101171



LC 11	
$x_3x_2x_1$	y
000	0
001	1
010	1
011	0
100	0
101	1
110	1
111	0
SEL	0

LC 12	
$x_3x_2x_1$	y
000	0
001	0
010	1
011	1
100	1
101	1
110	0
111	0
SEL	0

LC 21	
$x_3x_2x_1$	y
000	0
001	1
010	1
011	0
100	1
101	0
110	0
111	1
SEL	1

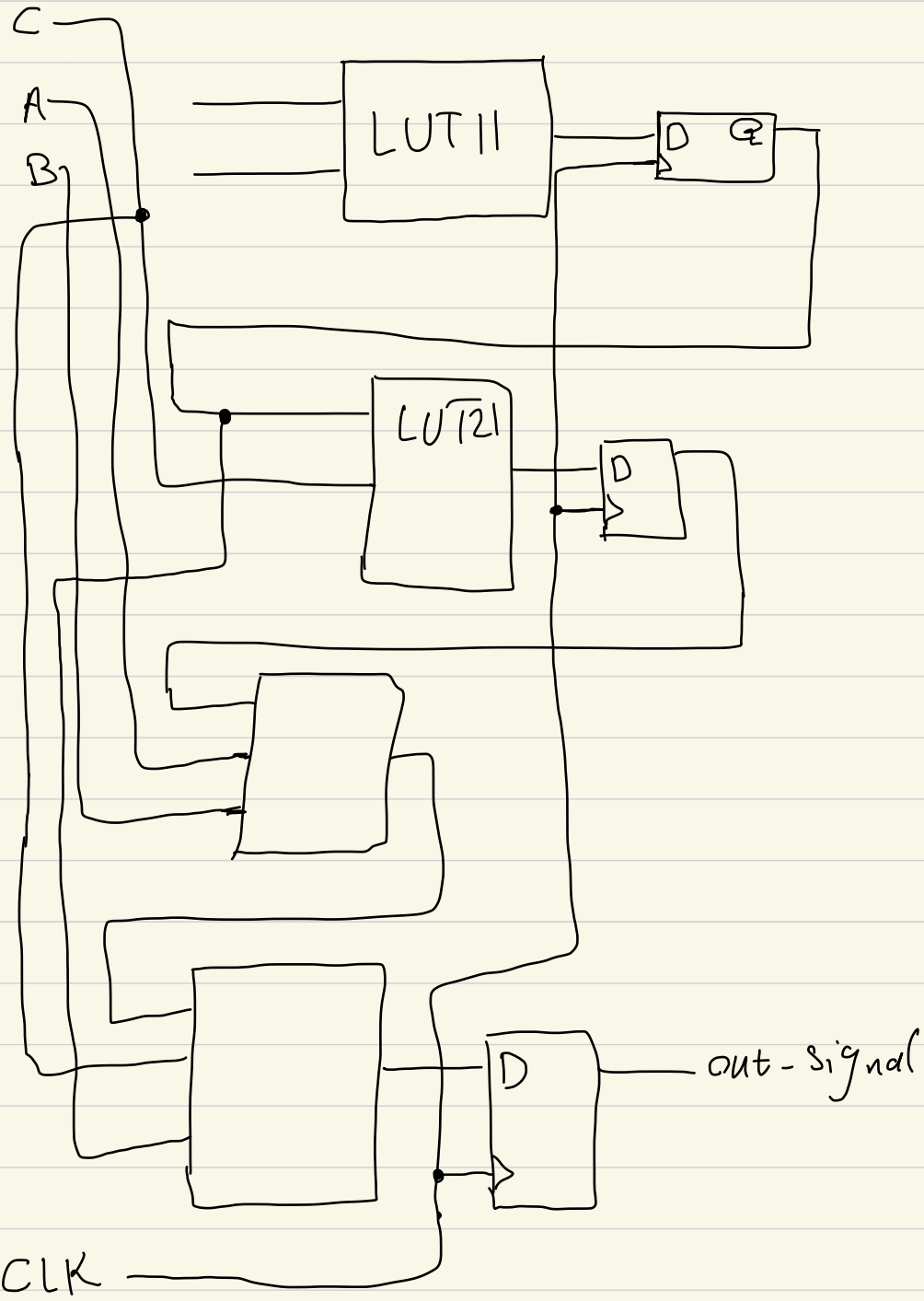
LC 22	
$x_3x_2x_1$	y
000	1
001	0
010	0
011	1
100	0
101	1
110	1
111	0
SEL	0

3- we have 8 total States and each one can lead to 8 other States. First we simplify the Circuit and then create a table and then a State machine accordingly: (for simplicity, we seprate it into two tables)

A	B	C	Q_{11}	Q_{12}	Q_{22}	S_1	S_2	S_3	out-signal
0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0	0	0
0	1	1	0	0	0	1	1	1	0
1	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	1	1	1	0
1	1	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	1	1	0
0	0	0	0	0	1	0	0	1	1
0	0	1	0	0	1	0	1	0	1
0	1	0	0	0	1	1	0	0	1
0	1	1	0	0	1	1	1	1	1
1	0	0	0	0	1	1	0	0	1
1	0	1	0	0	1	1	1	1	1
1	1	0	0	0	1	0	0	1	1
1	1	1	0	0	1	0	1	0	1
0	0	0	1	1	0	0	0	1	0
0	0	1	1	1	0	1	1	0	0
0	1	0	1	1	0	1	0	1	0
0	1	1	1	1	0	1	1	0	0
1	0	0	1	1	0	1	0	1	0
1	0	1	1	1	0	1	1	0	0
1	1	0	1	1	0	0	0	0	0
1	1	1	1	1	0	0	1	1	0
0	0	0	1	1	1	0	0	1	1
0	0	1	1	1	1	0	1	0	1
0	1	0	1	1	1	1	0	0	1
0	1	1	1	1	1	1	1	1	1
1	0	0	1	1	1	1	0	0	1
1	0	1	1	1	1	1	1	1	1
1	1	0	1	1	1	0	0	1	1
1	1	1	1	1	1	0	1	0	1
0	0	0	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1
0	1	0	1	1	1	1	0	0	1
0	1	1	1	1	1	1	1	1	1
1	0	0	1	1	1	1	0	0	1
1	0	1	1	1	1	1	1	1	1
1	1	0	1	1	1	0	0	1	1
1	1	1	1	1	1	0	1	0	1

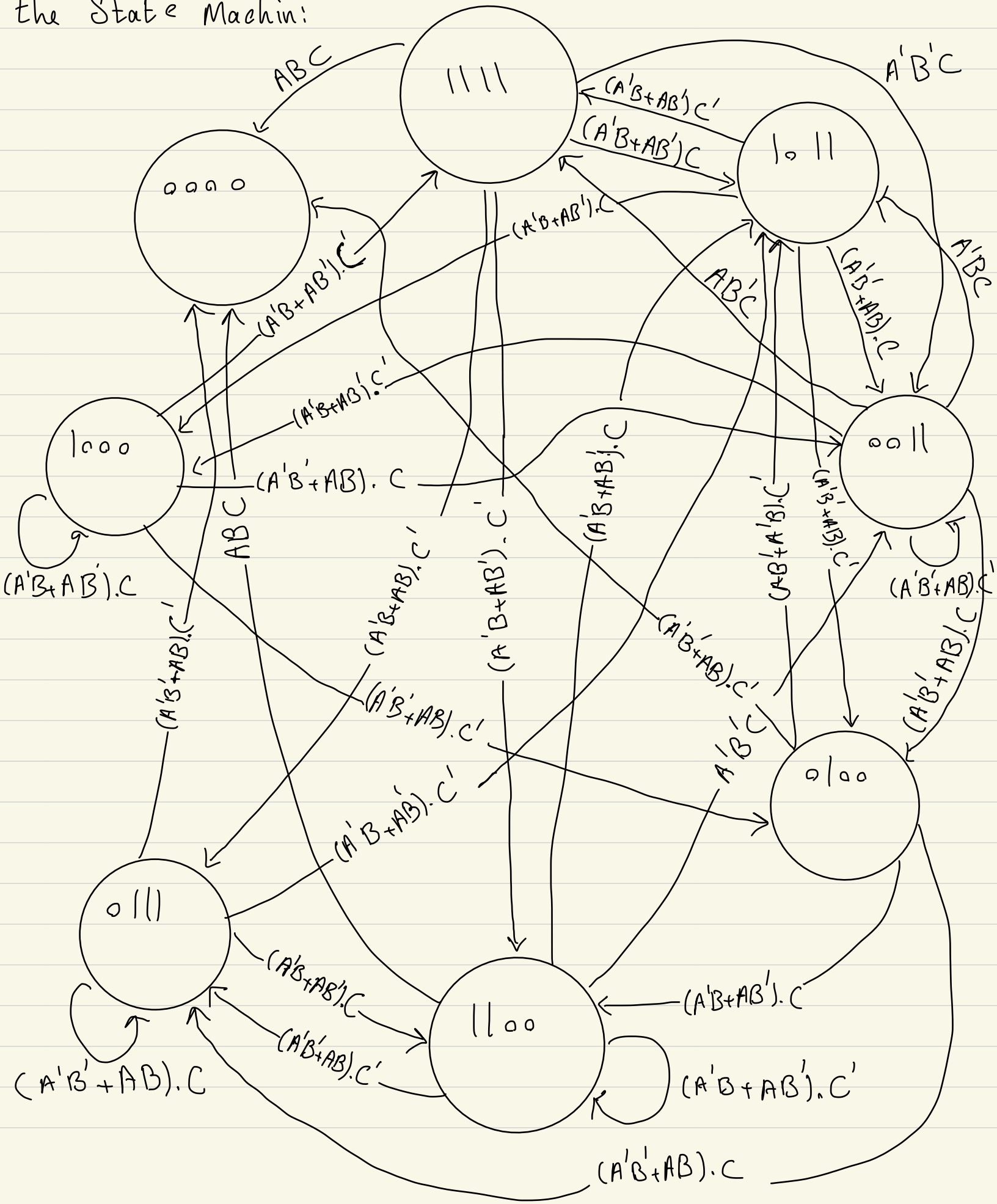
A	B	C	Q_{11}	Q_{12}	Q_{22}	S_1	S_2	S_3	out-signal
0	0	0	1	0	0	0	1	0	0
0	0	1	1	0	0	0	0	1	0
0	1	0	1	0	0	1	1	1	0
0	1	1	1	0	0	1	0	0	0
1	0	0	1	0	0	1	1	1	0
1	0	1	1	0	0	1	0	0	0
1	1	0	1	0	0	0	1	0	0
1	1	1	1	0	0	0	0	1	0
0	0	0	1	0	1	0	1	0	1
0	0	1	1	0	1	0	0	1	1
0	1	0	1	0	1	1	1	1	1
0	1	1	1	0	1	1	0	0	1
1	0	0	1	0	1	1	1	1	1
1	0	1	1	0	1	1	0	0	1
1	1	0	1	0	1	0	1	0	1
1	1	1	1	0	1	0	0	1	1
0	0	0	1	1	0	0	1	1	0
0	0	1	1	1	0	0	0	1	0
0	1	0	1	1	0	1	1	0	0
0	1	1	1	1	0	1	0	1	0
1	0	0	1	1	0	1	1	0	0
1	0	1	1	1	0	1	0	1	0
1	1	0	1	1	0	0	1	1	0
1	1	1	1	1	0	0	0	0	0
0	0	0	1	1	1	0	1	1	1
0	0	1	1	1	1	0	0	0	1
0	1	0	1	1	1	1	1	1	1
0	1	1	1	1	1	1	0	0	1
1	0	0	1	1	1	1	1	1	1
1	0	1	1	1	1	1	0	0	1
1	1	0	1	1	1	0	1	1	1
1	1	1	1	1	1	0	0	0	1

Based on the truth table drawn before, we can create a state machine of the circuit:



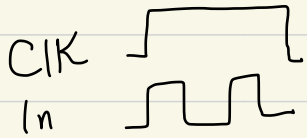
the numbers represent ABC Out

the State Machin:



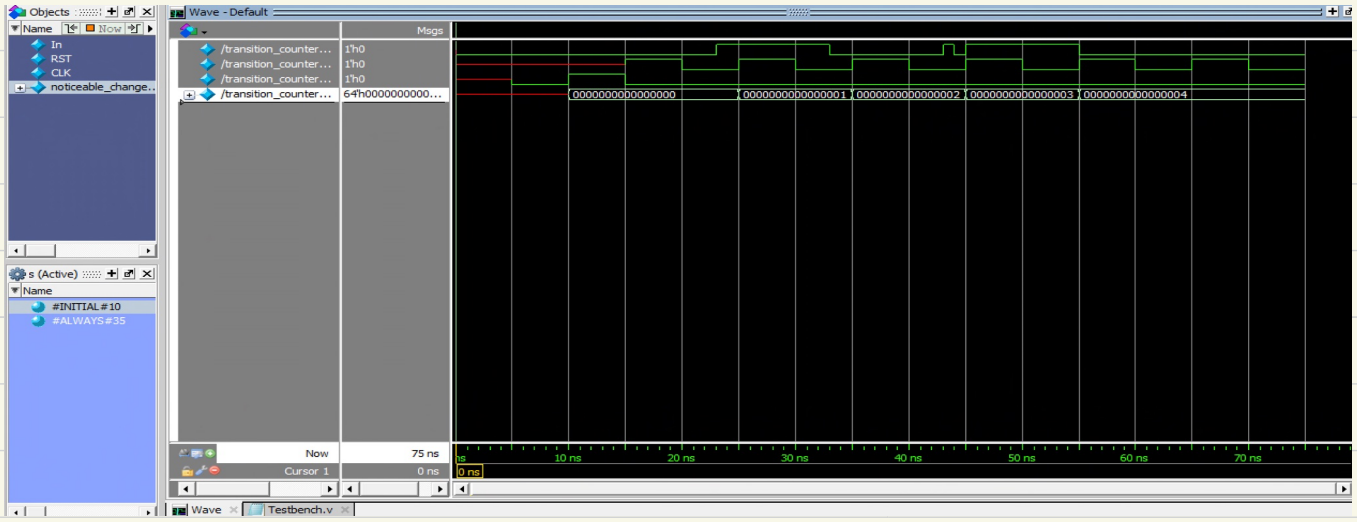
6:

A: For the changes to be Counted correctly, In's Value mustn't change more than once in a single clock So the counter Can work correctly.

For example, if our waveform looks like 

the Counter won't function Properly.

B: to find out if we have Came across an error, we Create two Counters and Compare them. An error is produced if they are not equal. A picture of an example where errors are Produced and also the waveform of the testbench:




```
# Reading pref.tcl
# //  ModelSim SE-64 2020.4 Oct 13 2020
# //
# //  Copyright 1991-2020 Mentor Graphics Corporation
# //  All Rights Reserved.
# //
# //  ModelSim SE-64 and its associated documentation contain trade
# //  secrets and commercial or financial information that are the property of
# //  Mentor Graphics Corporation and are privileged, confidential,
# //  and exempt from disclosure under the Freedom of Information Act,
# //  5 U.S.C. Section 552. Furthermore, this information
# //  is prohibited from disclosure under the Trade Secrets Act,
# //  18 U.S.C. Section 1905.
# //
# Loading project Question6
# Compile of main.v was successful.
# Compile of Testbench.v was successful.
# 2 compiles, 0 failed with no errors.
ModelSim> vsim -voptargs=+acc work.transition_counter_tb
# vsim -voptargs="+acc" work.transition_counter_tb
# Start time: 19:53:21 on Jul 11,2021
# ** Note: (vsim-3812) Design is being optimized...
# Loading work.transition_counter_tb(fast)
# Loading work.transition_counter(fast)
VSIM 2> vsim -voptargs=+acc work.transition_counter_tb
# End time: 20:04:09 on Jul 11,2021, Elapsed time: 0:10:48
# Errors: 0, Warnings: 2
# vsim -voptargs="+acc" work.transition_counter_tb
# Start time: 20:04:09 on Jul 11,2021
# ** Note: (vsim-8009) Loading existing optimized design _opt
# Loading work.transition_counter_tb(fast)
# Loading work.transition_counter(fast)
add wave  \
sim:/transition_counter_tb/In
add wave -position end  sim:/transition_counter_tb/CLK
add wave -position end  sim:/transition_counter_tb/RST
add wave  -position end  sim:/transition_counter_tb/noticeable_changes
VSIM 6> run
# Error at                    45: real changes are at:                    5 but apparent changes are at:                    3
# Error at                    55: real changes are at:                    5 but apparent changes are at:                    4
# Error at                    65: real changes are at:                    6 but apparent changes are at:                    4
# ** Note: $stop    : C:/modeltech64_2020.4/examples/Testbench.v(32)
#    Time: 75 ns  Iteration: 0  Instance: /transition_counter_tb
```

Transcript | Wave | Testbench.v | Objects | Processes | Library | Project | sim

7: the python code is included. To test it, we simply use the code of a D Flip Flop from a previous exercise:

```
`timescale 1ns / 1ps

module DFF(
    output q,
    output qBar,
    input d,
    input clk,
    input resetBar
    );

wire tmp0;
wire tmp1;
wire tmp2;
wire tmp3;
assign tmp3 = ~(resetBar & clk);
assign tmp2 = ~(resetBar & ~(tmp1 & ~(d & tmp0)));
assign tmp1 = ~(~tmp2 & tmp3);
assign tmp0 = ~(tmp2 & tmp3);
assign qBar = ~(tmp0 & q);
assign q = ~(tmp1 & qBar);

endmodule
```

Converted with our Python Program:

```
`timescale 1ns / 1ps

module DFF(
    output reg q,
    output reg qBar,
    input d,
    input clk,
    input resetBar
    );

wire tmp0;
wire tmp1;
wire tmp2;
wire tmp3;


    always @ (tmp3) begin
        None <= tmp3;
    end


    always @ (tmp2) begin
        None <= tmp2;
    end


    always @ (tmp1) begin
        None <= tmp1;
    end


    always @ (tmp0) begin
        None <= tmp0;
    end


    always @ (qBar) begin
        None <= qBar;
    end


    always @ (q) begin
        None <= q;
    end

endmodule
```