PBL#13 Report: Counter 0-9

Name: Goutam Krapa
ID: 2215855

1. Introduction

In this design project PBL13, our goal is to implement a 0 to 9 counter circuit. The circuit counts 4-bit hexadecimal values from 4'ho to 4'h9 and the resets to 4'h0, when the counter reaches 4'h9 as shown in Figure 2. With the help of the block diagram in the Figure 1 we have designed our Counter 0 to 9 with below design specification:

- The circuit contains a register which plays a vital role as it works with control signals such as clock (clk) and reset signal (rst)
  - Clock signal is the signal that makes our register work on time period. In our design based on the test bench we are working with 100MHz clock cycle.
  - The counter works with an asynchronous reset signal. When "rst" signal is 0 then the output it set to 4'h0
- The counter also includes enable signal as a control signal for the counter to function. When the enable signal "en" is set to 1 the counter counts.
- We have 2 more clocks of circuit to make the counter function as expected they are the comparator and multiplexer.
  - A comparator is used to compare and validate if the counter count has is 4'h9.
  - The multiplexer is used to increment the count or reset the count to 4'h0 based on the results of the comparator. If the comparator finds the count to be 4'h0 then the comparator would reset the count to 4'h0 or would calculate the next count signal "nxt_cnt" by incrementing count by 1 (cnt+1)

This report provides a comprehensive explanation of the design process, design code, test bench code, and key comments.

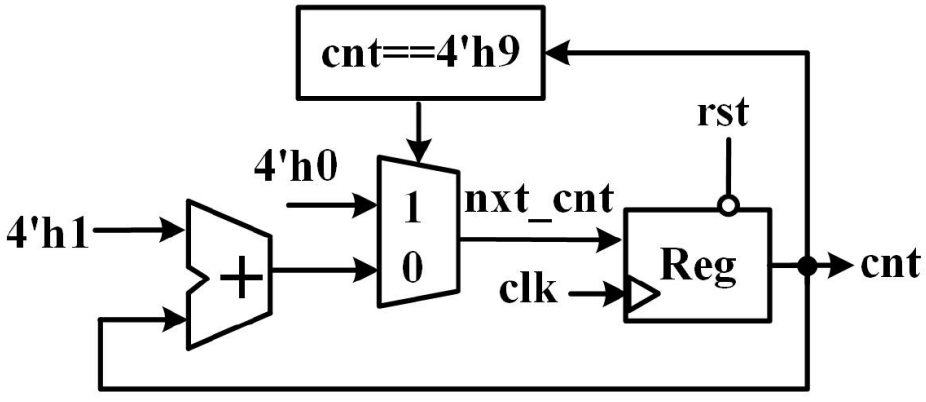

Figure 1: Block Diagram of Counter for 10 Cycles

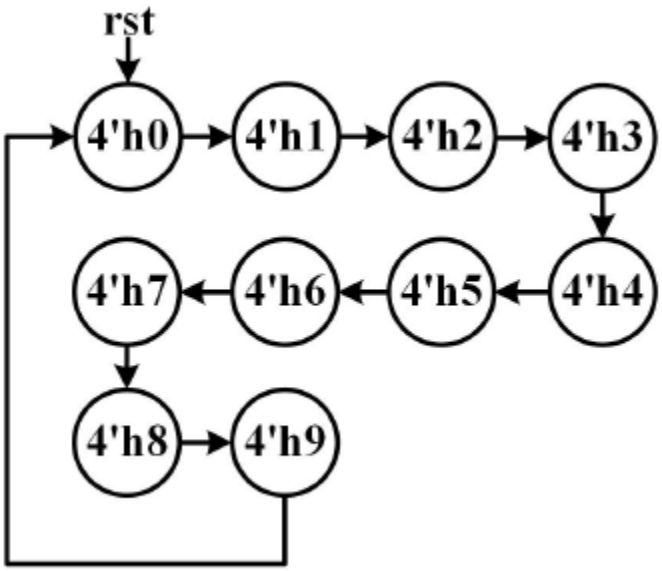

Figure 2: Design Specification of counter for 10 cycles

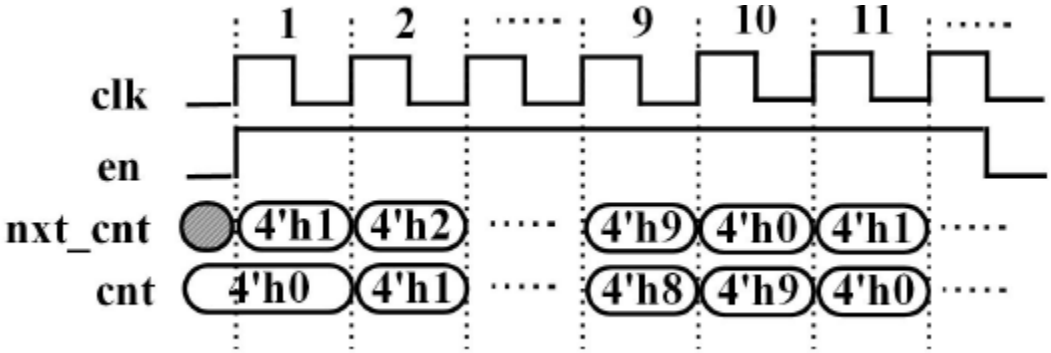

Figure 3: Timing Diagram of Counter for 10 Cycles

2. Verilog HDL Design Code

2.1. and_or_reg.v

```
module counter_0_to_9 (
   input  clk,       // clock signal
   input  rst,       // reset
   input  en,        // enable signal which is active when high
   output reg [3:0] cnt  // 4-bit counter output
);
```

```
// MUX logic for next counter value
   wire [3:0] nxt_cnt = (cnt == 4'h9) ? 4'h0 :  // reset counter to 4'h0 if cnt is 4'h9
                        (en ? cnt + 4'h1 : cnt);  // secondary MUX to increment if en, else hold current
value

          //or
          //wire [3:0] inc_cnt = en ? cnt + 4'h1 : cnt;
          //wire [3:0] nxt_cnt = (cnt == 4'h9) ? 4'h0 :  inc_cnt;

   // Counter logic with reset
   always @(posedge clk, negedge rst) begin
      if (~rst) begin
         cnt <= 4'h0;  // reset counter to 0 when reset is active that is rst=1
      end else begin
         cnt <= nxt_cnt;  // update counter with next value as next counter
      end
   end

endmodule
```

2.2. tb_and_or_reg.v

```
module tb_counter_0_to_9();
   // Inputs
   reg clk;
   reg rst;
   reg en;
   // Outputs
   wire [3:0] cnt;


   // Instantiate the counter module uut=Unit Under Testing
   counter_0_to_9 u_counter_0_to_9 (
      .clk(clk),
      .rst(rst),
      .en(en),
      .cnt(cnt)
   );

//Bus Function model
```

```
        //clock process 100 MHz
    always #5 clk = ~clk;


    initial begin
        // Initial state
                    rst=1'b0; clk=1'b0; en=1'b0;
        #10 rst=1'b1;
                    @(posedge clk) en=1'b1;
                    #150 rst=1'b0;//reset within counting
                    #30 rst=1'b1;
                    repeat(15) @(posedge clk) en=1'b0;
                    repeat(10) @(posedge clk) en=1'b1;


        // uncomment if you want to Finish the simulation
        //$finish;
    end

    // monitor signal behavior
    initial begin
        $monitor("Time: %0d ns, clk: %b, rst: %b, en: %b, cnt: %0d",
                $time, clk, rst, en, cnt);
    end
endmodule
```

2.3. TCL Script: run

```
#!/bin/csh -f
##############################################################################
##
# simulation top script by Goutam Krapa 2215855
# change modelsim in this file dir and run "do this_file.do"
# project dir ---+---> hdl source dir
#                +---> sim script dir
#
##############################################################################
##

# check if current dir has modelsim config file
set has_config  [file exists modelsim.do]
```

```
# config modelsim
if {$has_config==1} { do modelsim.do  ; }

echo "+====================================="
echo "| Creat Lib Work soc                  "
echo "+====================================="
vlib work
vmap work work

echo "+====================================="
echo "| Complile RTL Code of soc            "
echo "+====================================="

vlog -f ../filelist/filelist.f
##or##vlog -v ../dut/and_or.v ../tb/tb_counter_0_to_9.v
##vlog -v ../dut/counter_0_to_9.v ../tb/tb_counter_0_to_9.v

echo "+====================================="
echo "| Compiler Pass                       "
echo "| Being to Run Simulation             "
echo "+====================================="
vsim work.tb_counter_0_to_9 -t 1ns

################## ##################### mus
####################################
add wave -noupdate -format logic -radix hexadecimal tb_counter_0_to_9/clk
add wave -noupdate -format logic -radix hexadecimal tb_counter_0_to_9/rst
add wave -noupdate -format logic -radix hexadecimal tb_counter_0_to_9/en
add wave -noupdate -format logic -radix hexadecimal
tb_counter_0_to_9/u_counter_0_to_9/nxt_cnt
add wave -noupdate -format logic -radix hexadecimal tb_counter_0_to_9/cnt

add wave -noupdate -format logic -radix hexadecimal -color magenta
tb_counter_0_to_9/u_counter_0_to_9/clk
add wave -noupdate -format logic -radix hexadecimal -color magenta
tb_counter_0_to_9/u_counter_0_to_9/rst
add wave -noupdate -format logic -radix hexadecimal -color magenta
tb_counter_0_to_9/u_counter_0_to_9/en
add wave -noupdate -format logic -radix hexadecimal -color magenta
tb_counter_0_to_9/u_counter_0_to_9/cnt
add wave -noupdate -format logic -radix hexadecimal -color magenta
tb_counter_0_to_9/u_counter_0_to_9/nxt_cnt
```

run 600ns

##possible colors of waveform black
#white
#red
#green
#blue
#yellow
#cyan
#magenta

2.4. Project code

Please find the zip file of the code in the attachment.

PBL-Report-GoutamKrapa-2215855-2.zip

3. Simulation Waveform

The below following simulation waveforms are the resultant simulations generated from model sim software a tool used for simulation. We can validate that the output as follows:

- It is working with 100MHz of clock cycles.
- The initial reset and counter reset to 4'h0.
- First, we set reset to 1 at 10ns and enable to 1 at 15ns. This is when the counter starts to count from 4’h0 to 4’h9.
- We can notice that at 115ns counter changes from 4’h9 to 4’h0 there by validating the reset of counter when counter count reaches 4’h9.
- At 165ns when counter reset is 0 the counter value is forced to be 4’h0 and only starts counting when both rest and enable are set to 1.
- Counter only counts if the enables signal is set to 1.
- From 195ns we see that enable is set to 0 thus counter stops counting and holds the current count value and restarts counting when enable is set to 1 at 345ns.
- Based on the timing diagram from Figure 3 the simulation waveform is as expected.

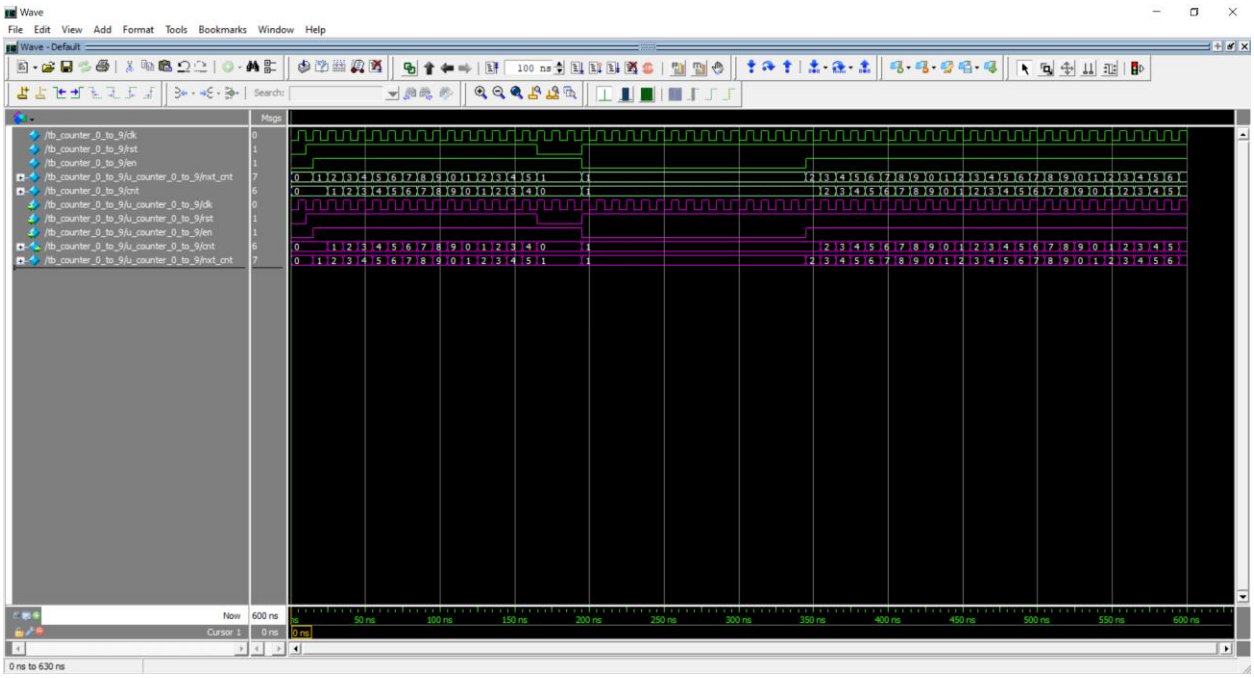

Figure 4: Simulation waveform for the counter_0_to_9.v with tb_ counter_0_to_9.v

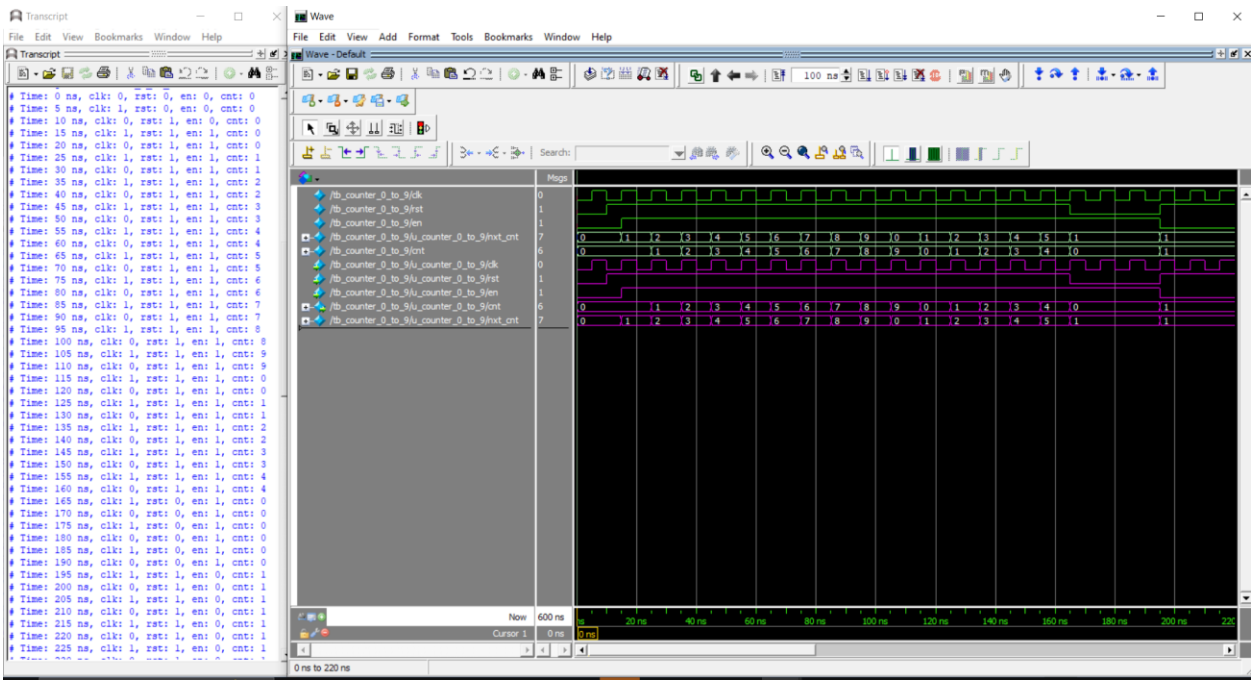

Figure 5: Simulation waveform for the counter_0_to_9.v with tb_ counter_0_to_9.v

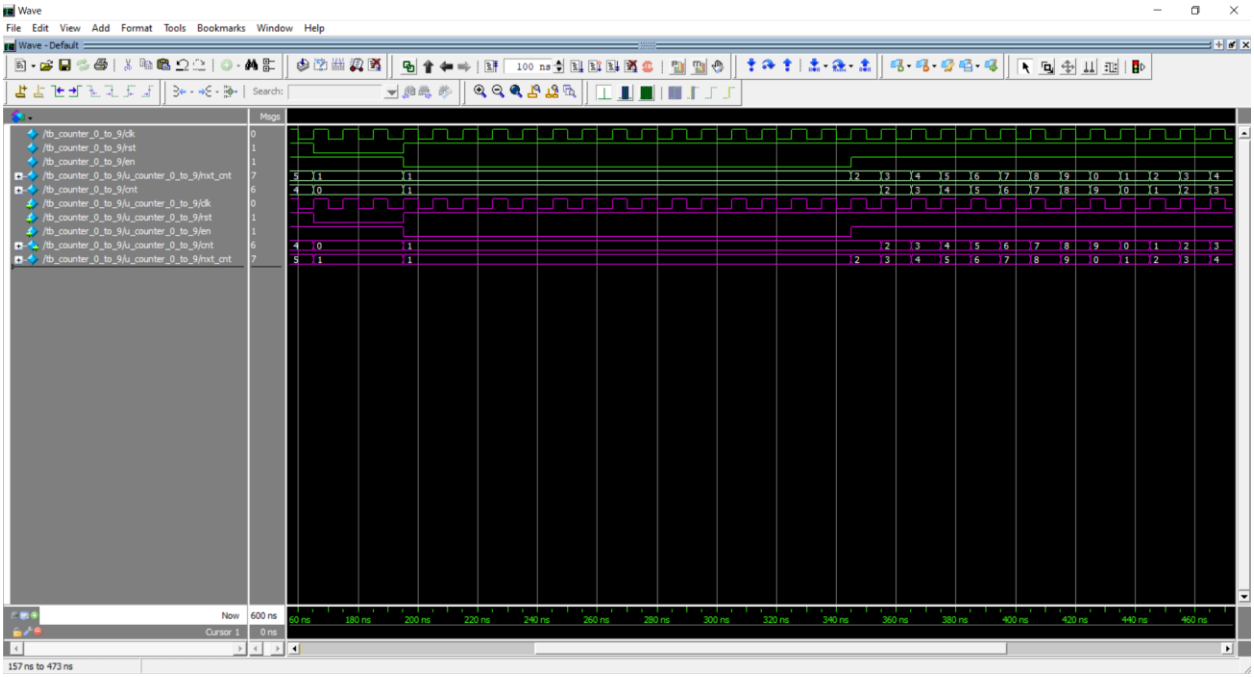

Figure 6: Simulation waveform for the counter_0_to_9.v with tb_ counter_0_to_9.v

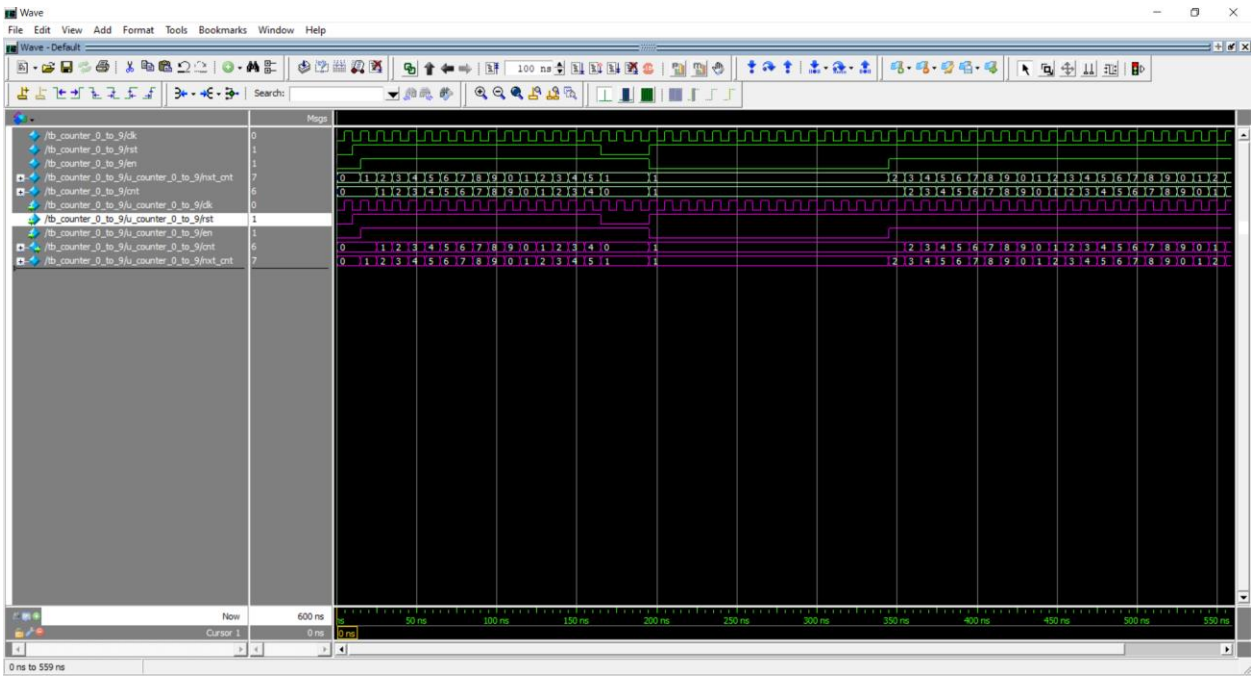

Figure 7: Simulation waveform for the counter_0_to_9.v with tb_ counter_0_to_9.v

4. Comments and Conclusion

4.1. Design project

Our project involves four main components Comparator, multiplexer, incrementor and register. Our Design code Test Bench code and TCL scripts are used to implement counter 0 to 9 as given

in the Figure 1. The output from the Comparator, Multiplexer and Incrementor is given to the register which is a 4-bit hexadecimal signal. Based on the block diagram we use rst, negedge, clk and posedge.

The Register incorporated in the given circuit additionally has two necessary control signals named CLK (clock signal) and RST (reset signal). The reset signal ensures the output is set to 4'h0 i.e. when the reset is true (RST=0) the output "cnt" is 4'h0 and if reset is True the output "cnt" is the 1-bit result of the circuit.

The design of the circuit based on above points is in the counter_0_to_9.v file and the test bench used to define the test cases is in tb_ counter_0_to_9.v file whereas the TCL script used to generate simulated waveforms is in the run file.

4.2. Design Challenges

- **Timing Control and Edge Sensitivity:** One of the challenges faced in PBL13 is to manage the edge triggering behavior of the register. It needs to respond correctly to both the rising edge of the clock signal and falling edge of the reset signal.
- **Synchronization of Signals:** During simulation we need to ensuring that all inputs are changed at the correct time while maintaining synchronization with the clock and reset.
- **Reset Logic:** The reset logic must be implemented such that a consistent reset operation is performed and it resets the register output to 4'h0 when activated (RST=0).
- **Conditional operation of counter with enable signal:** Design a counter such that it only increments when the enable is set to 1.

4.3. Constraint

- **Fixed Counting Range (4'h0-4'h9):** The design has a specific constrain to have a specific boundary between 4'h0 and 4'h9.

4.4. Conclusion

On cross validating the generated simulation waveform we can conclude that with the help of Verilog HDL code we have translated the design specification into logical functional code and through simulation we displayed the simulated waveform. The counter counts from 4'h0 to 4'h9 and resets to 4'h0 if the count reaches 4'h9. The circuit also properly resets to 4'h0 when rst signal is 0 and the counter only counts when the enable signal is active. Thus, we can conclude that the counter is working as expected from the waveform simulation.