ES204 Digital Systems
LAB Assignment - 3

Indian Institute of Technology, Gandhinagar
January 25, 2024

Submission deadline: Jan 29, 2024
Marks : 40

Submission instructions:

(a) Only one student from the team will submit with the word doc name Rollno1_Rollno2.pdf. The PDF will contain the code, testbench and simulation results.
(b) Make a tar-ball / Zip of the project and upload.

For each of the questions, write a Verilog code. You also need to create a **testbench** and show the simulation results.

Design a 4-bit combined BCD/Binary Up/Down counter using **Structural code**. Use Toggle FFs for designing this synchronous counter.

The counter outputs are connected to a 4-bit Shift register. The shift register allows no-shift/left-shift/right-shift operations. The shifter code needs to be written as **Behavioral code**.

Write a top_module that instantiates counter and shift register appropriately with all the mode bits to allow selection of counter mode and shift mode.

Lab assignment 3 ES 204

Birudugadda Srivibhav (22110050)
Reddybathuni Venkat (22110220)

#Structural implementation code of 4-bit combined BCD/Binary Up/Down counter

1. Code

```
`timescale 1ns / 1ps
module T_ff(
//T flopflop with always block
    input clk, reset, en, T,
    output reg Q
    );
    always@(posedge clk)
    begin
    if(!reset & en)
        Q <= 0;
    else if(reset & en)
        Q <= T ^ Q;
    else
        Q <= Q;
    end
endmodule

module counter(
    input clk, reset, en,
    input [1:0]M,
    output [3:0]Q
    );
    wire [3:0]T_binary; // for flipflop inputs T in case of binary
    wire [3:0]T_BCD;// for flipflop inputs T in case of BCD
    // populating array T_binary with binary outputs using multiplexer for up and down counters.
    and binary0(T_binary[0], 1,1);

    xor binary1(T_binary[1],Q[0],M[0]);

    and binary20(bin20, Q[0], Q[1]);
    and binary21(bin21, ~Q[0], ~Q[1]);
    and binary22(bin22, bin20, ~M[0]);
and binary23(bin23, bin21, M[0]);
or binary24(T_binary[2], bin22, bin23);

and binary30(bin30, Q[0], Q[1], Q[2]);
and binary31(bin31, ~Q[0],~Q[1], ~Q[2]);
and binary32(bin32, bin30, ~M[0]);
and binary33(bin33, bin31, M[0]);
or binary34(T_binary[3], bin32, bin33);

// populating array T_binary with BCD outputs using multiplexer for up and down counters.
and BCD0(T_BCD[0], 1,1);

and BCD10(bcd10, ~Q[3], Q[0]);
or BCD11(bcd11, Q[1], Q[2], Q[3]);
and BCD12(bcd12, bcd11, ~Q[0]);
and BCD13(bcd13, ~M[0], bcd10);
and BCD14(bcd14, bcd12, M[0]);
or BCD15(T_BCD[1], bcd14, bcd13);

and BCD20(bcd20, Q[1], Q[0]);
and BCD21(bcd21, Q[2], ~Q[1]);
or BCD22(bcd22, bcd21, Q[3]);
and BCD23(bcd23, bcd22, ~Q[0]);
and BCD24(bcd24, ~M[0],bcd20);
and BCD25(bcd25, bcd23, M[0]);
or BCD26(T_BCD[2], bcd24, bcd25);

and BCD30(bcd30, ~Q[0], ~Q[1], ~Q[2]);
and BCD31(bcd31, Q[3], Q[0]);
and BCD32(bcd32, Q[0], Q[1], Q[2]);
or BCD33(bcd33, bcd31, bcd32);
and BCD34(bcd34, ~M[0], bcd33);
and BCD35(bcd35, M[0], bcd30);
or BCD36(T_BCD[3], bcd34, bcd35);

wire [3:0]T;// For final T flipflop inputs using multiplexer for selecting BCD and binary

and T0(T[0], 1,1);

and T10(t10, ~M[1], T_binary[1]);
and T11(t11, M[1], T_BCD[1]);
or T12(T[1], t10, t11);

and T20(t20, ~M[1], T_binary[2]);
```

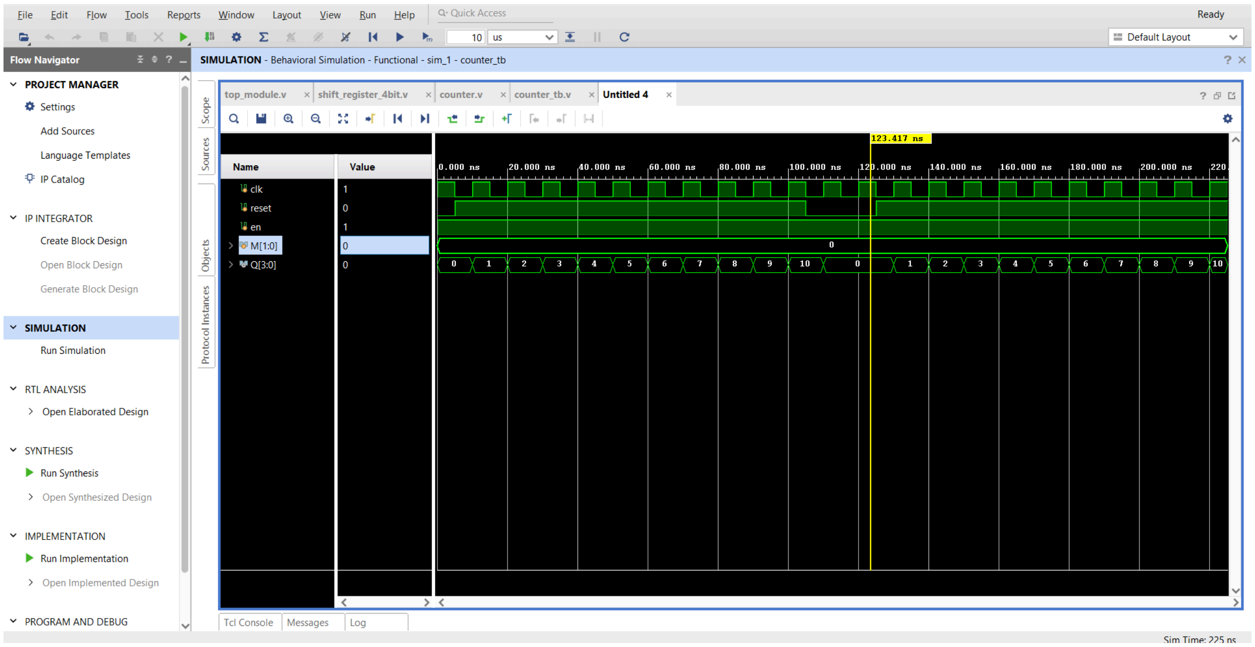
```
   and T21(t21, M[1], T_BCD[2]);
   or T22(T[2], t20, t21);

   and T30(t30, ~M[1], T_binary[3]);
   and T31(t31, M[1], T_BCD[3]);
   or T32(T[3], t30, t31);
   //implementation of 4 bit counter
   T_ff bit0(.clk(clk), .reset(reset), .en(en), .T(T[0]), .Q(Q[0]));
   T_ff bit1(.clk(clk), .reset(reset), .en(en), .T(T[1]), .Q(Q[1]));
   T_ff bit2(.clk(clk), .reset(reset), .en(en), .T(T[2]), .Q(Q[2]));
   T_ff bit3(.clk(clk), .reset(reset), .en(en), .T(T[3]), .Q(Q[3]));
endmodule
```

2. Test bench

```
`timescale 1ns / 1ps
module counter_tb();
   reg clk;
   reg reset;
   reg en;
   reg [1:0]M;
   wire [3:0]Q;
   counter uut(.clk(clk),.reset(reset), .en(en), .M(M), .Q(Q));
   initial
   begin
   clk = 1;
   forever #5 clk = ~clk;
   end
   initial
   begin
   reset = 0; en =1; M = 2'b00;
   #5;
   reset = 1; en =1; M = 2'b00;
   #100;
   reset = 0; en =0; M = 2'b00;
   #20;
   reset = 1; en =1; M = 2'b00;
   #100;
   $finish();
   end
endmodule
```

3. Simulation

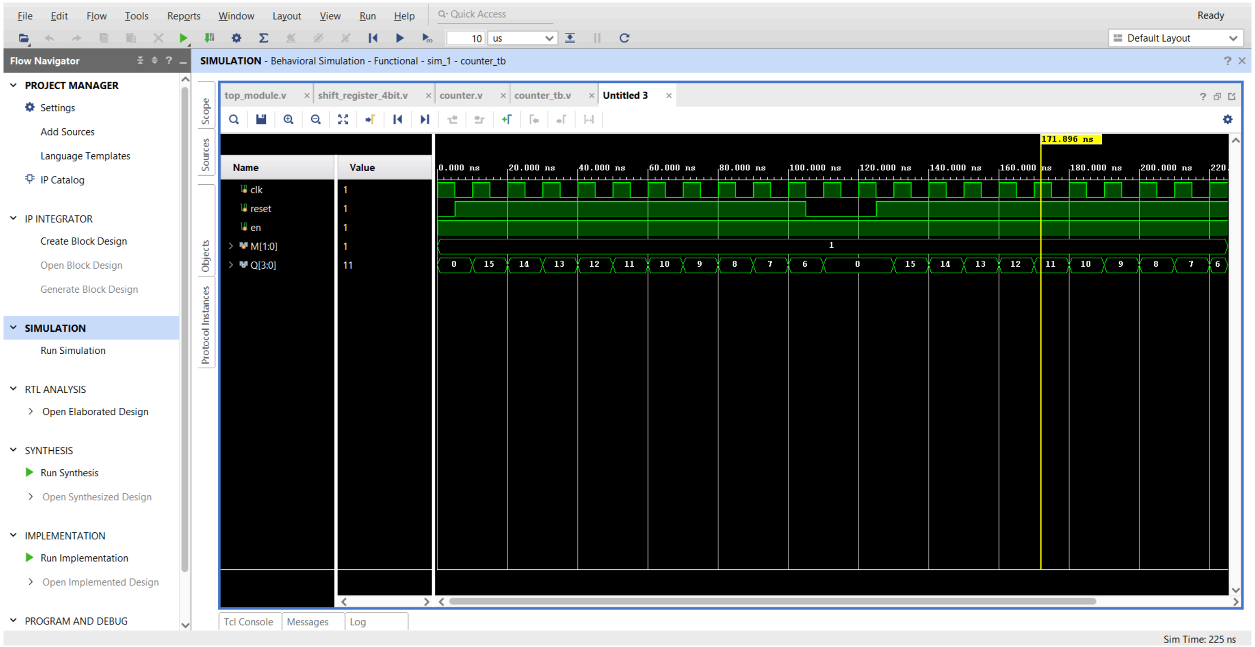
Binary Up counting



SIMULATION - Behavioral Simulation - Functional - sim_1 - counter_tb

Name	Value
clk	1
reset	0
en	1
M[1:0]	0
Q[3:0]	0

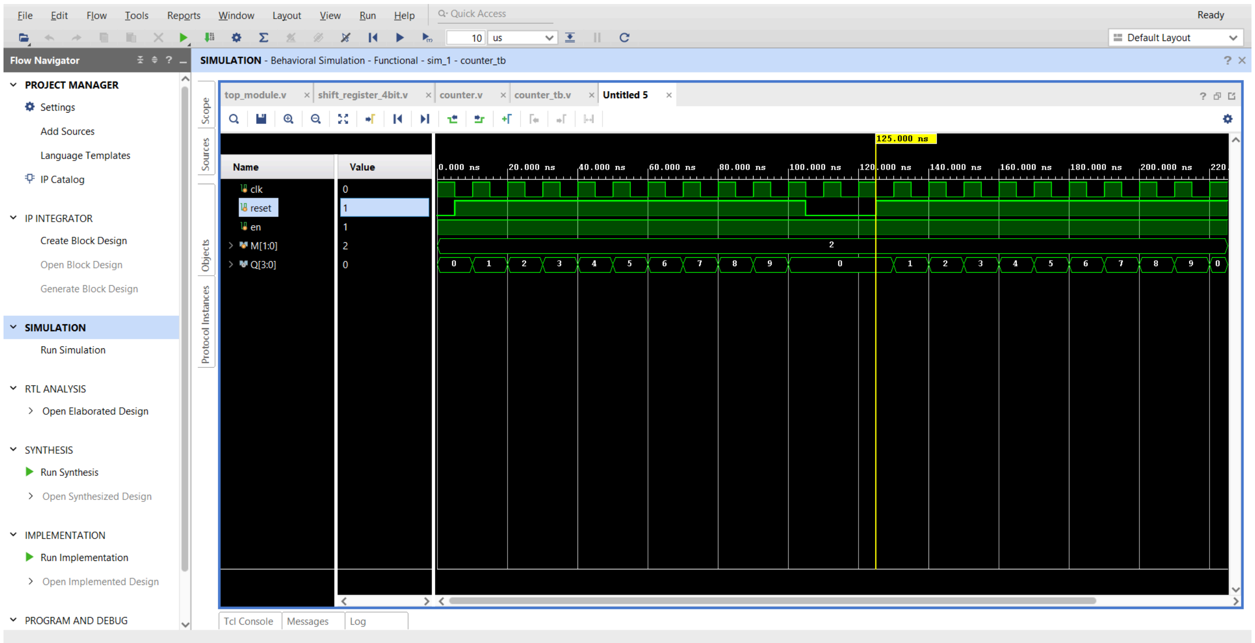
Binary down counting



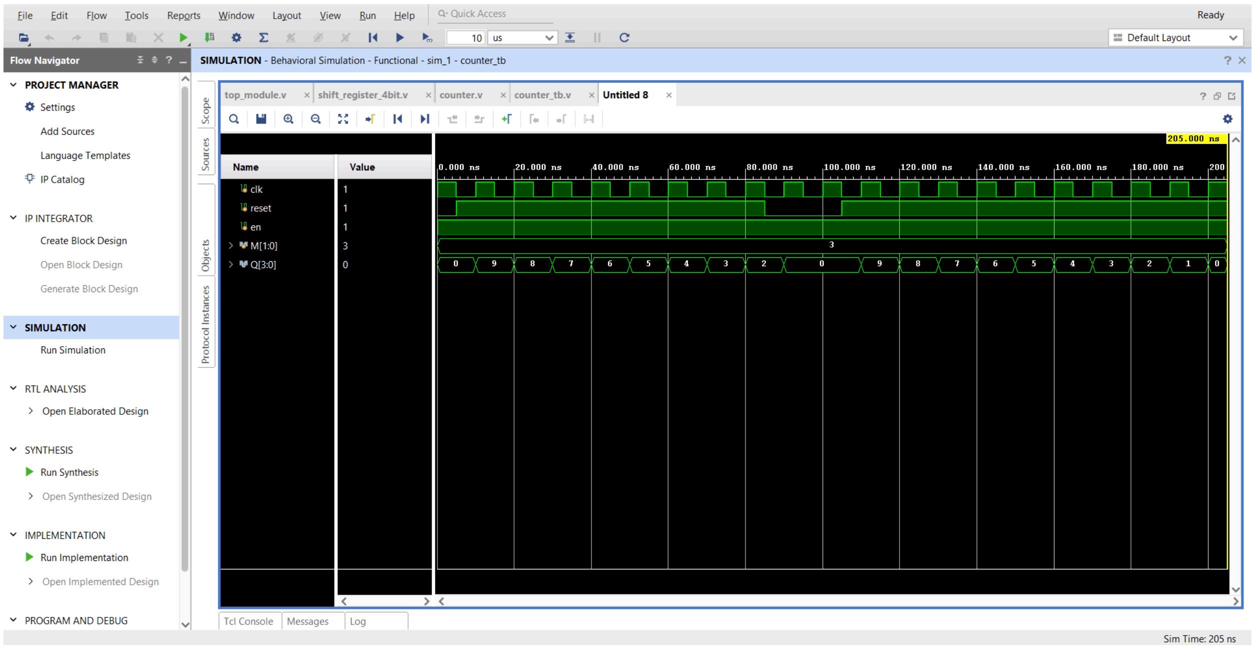
SIMULATION - Behavioral Simulation - Functional - sim_1 - counter_tb

Name	Value
clk	1
reset	1
en	1
M[1:0]	1
Q[3:0]	11

BCD up counting



BCD down counting



#Behavioral implementation code of 4-bit shift register which allows no-shift/left-shift/right-shift operations.

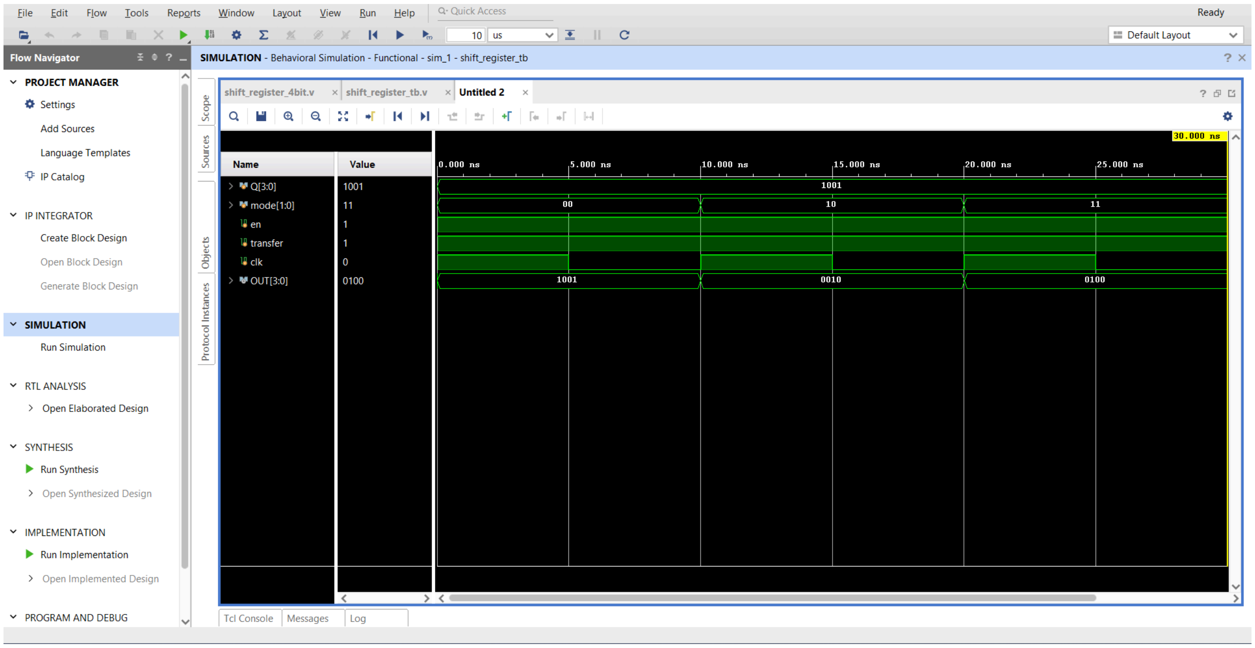
4. Code

```
`timescale 1ns / 1ps
module shift_register_4bit(
    input [3:0]Q,
    input [1:0] mode,
    // mode[1] = 0 is no shift
    //mode = 2'b10 is right shift
    // mode = 2'b11 is left shift
    input en, transfer,
    //transfer is high means parallel copy of bits from counter and low is for not copying
    input clk,
    output reg [3:0]OUT
    );
    always@(posedge clk)
    begin
    #0.01;
    if(transfer & en)
    OUT = Q;
    if(mode[1] == 1 & en)
    begin
        if(mode[0] == 1)
        begin
        OUT <= OUT >> 1;
        end
        else
        OUT <= OUT << 1;
    end
    else if(en)
    OUT <= OUT;
    else
    OUT <= 4'b0000;
    //when en = 0 OUT will be 0000
    end
endmodule
```

5. Test bench

```
module shift_register_tb();
   reg [3:0]Q;
   reg [1:0] mode;
   reg en;
   reg transfer;
   reg clk;
   wire [3:0]OUT;
   shift_register_4bit uut(.Q(Q), .mode(mode), .en(en), .transfer(transfer), .clk(clk), .OUT(OUT));
   initial
   begin
   clk = 1;
   forever #5 clk = ~clk;
   end
   initial
   begin
   transfer = 1; en = 1; Q = 4'b1001; mode =2'b00;
   #10;
   transfer = 1; en = 1; Q = 4'b1001; mode =2'b10;
   #10;
   transfer = 1; en = 1; Q = 4'b1001; mode =2'b11;
   #10;
   $finish();
   end
endmodule
```

6. Simulation



# A top_module that instantiates counter and shift register

7. Code

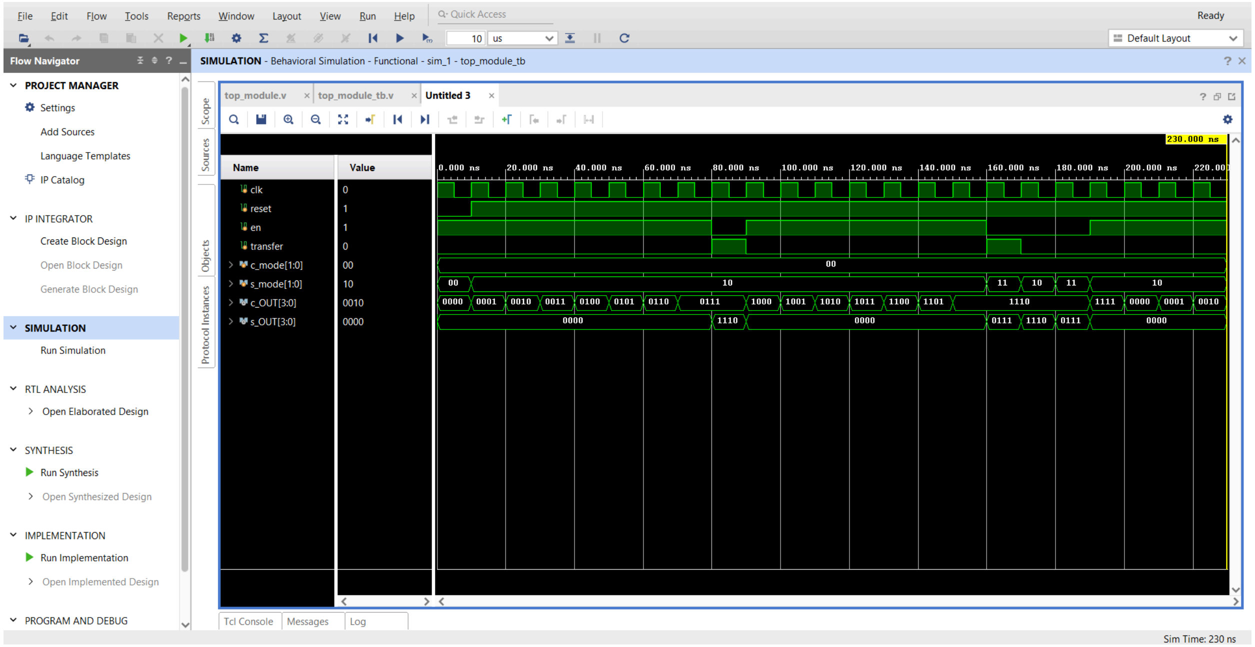
```
`timescale 1ns / 1ps
module top_module(
   input clk, reset, en, transfer,
   input [1:0] c_mode, s_mode,
   //c_mode is for counter modes
   //s_mode is for shift register modes
   output [3:0] c_OUT,s_OUT
   //c_OUT is for counter Outputs
   //s_OUT is for shift register Outputs
   );
   //en = 1 means counter is on
   // en = 0 means shift register is on
   counter inst1(.clk(clk), .reset(reset), .en(en), .M(c_mode), .Q(c_OUT));
   shift_register_4bit inst2(.Q(c_OUT), .mode(s_mode), .en(~en), .transfer(transfer), .clk(clk),
.OUT(s_OUT));
endmodule
```

8. Test bench

```
`timescale 1ns / 1ps
module top_module_tb();
   reg clk;
   reg reset;
   reg en;
   reg transfer;
   reg [1:0] c_mode;
   reg [1:0] s_mode;
   wire [3:0] c_OUT;
   wire [3:0] s_OUT;
   top_module uut(.clk(clk), .reset(reset), .en(en),.transfer(transfer), .c_mode(c_mode),
.s_mode(s_mode), .c_OUT(c_OUT), .s_OUT(s_OUT));
   initial
   begin
   clk = 1;
   forever #5 clk = ~clk;
   end
   initial
   begin
   en = 1; reset = 0; transfer = 0; c_mode = 2'b00; s_mode = 2'b00;
   #10;
   en = 1; reset = 1; transfer = 0; c_mode = 2'b00; s_mode = 2'b10;
   #70;
   en = 0; reset = 1; transfer = 1; c_mode = 2'b00; s_mode = 2'b10;
   #10;
   en = 1; reset = 1; transfer = 0; c_mode = 2'b00; s_mode = 2'b10;
   #70;
   en = 0; reset = 1; transfer = 1; c_mode = 2'b00; s_mode = 2'b11;
   #10;
   en = 0; reset = 1; transfer = 0; c_mode = 2'b00; s_mode = 2'b10;
   #10;
   en = 0; reset = 1; transfer = 0; c_mode = 2'b00; s_mode = 2'b11;
   #10;
   en = 1; reset = 1; transfer = 0; c_mode = 2'b00; s_mode = 2'b10;
   #40;
   $finish();
   end
endmodule
```

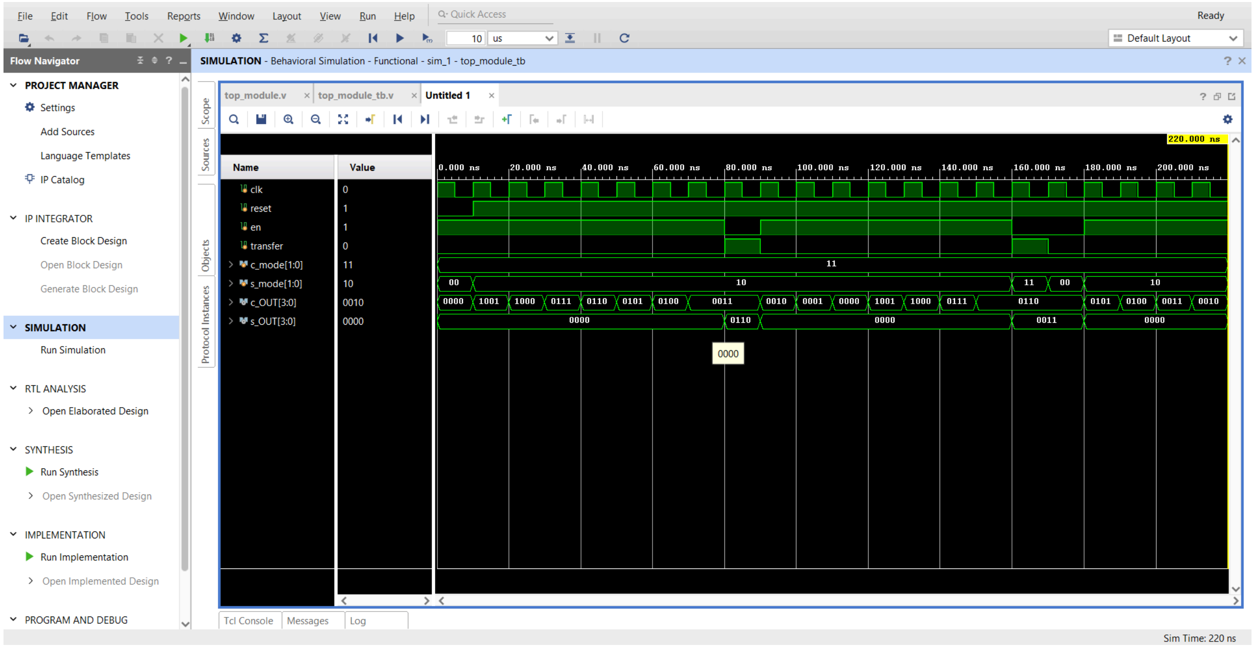
9. Simulation

Binary up counter with 1 shift and 3 shifts as shown in the figure.

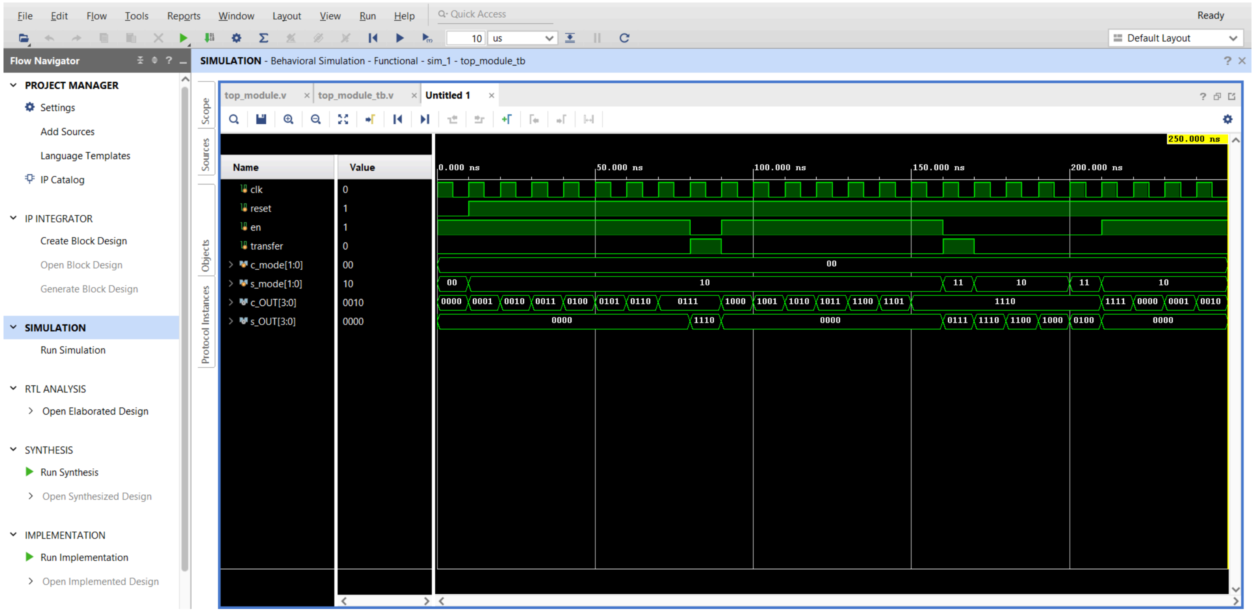


Name	Value
clk	0
reset	1
en	1
transfer	0
c_mode[1:0]	00
s_mode[1:0]	10
c_OUT[3:0]	0010
s_OUT[3:0]	0000

BCD down counter with shift 1 time followed by 2 times as shown in the figure.



Name	Value
clk	0
reset	1
en	1
transfer	0
c_mode[1:0]	11
s_mode[1:0]	10
c_OUT[3:0]	0010
s_OUT[3:0]	0000



SIMULATION - Behavioral Simulation - Functional - sim_1 - top_module_tb

Name	Value
clk	0
reset	1
en	1
transfer	0
c_mode[1:0]	00
s_mode[1:0]	10
c_OUT[3:0]	0010
s_OUT[3:0]	0000