EC-204

<LAB - 7>

NITK SURATHKAL

INBASEKARAN.P

201EC226

Prof: Sumam S

1. Decade Counter

Solution:

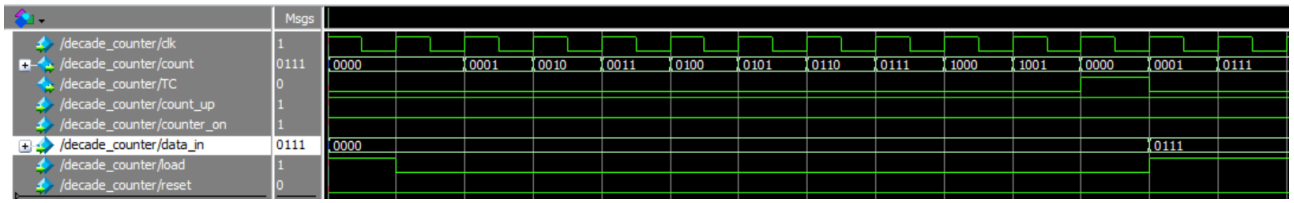
a) Code

question1 > question1.v

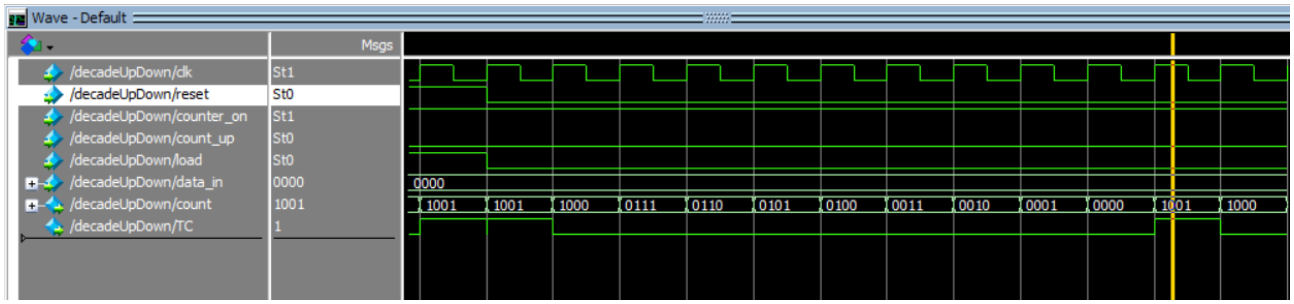
```
module decadeUpDown (count_up, load, reset, counter_on, data_in,clk, count, TC);
  input count_up, load, reset, counter_on;
  input [3:0]data_in; input clk;
  output reg [3:0]count; output reg TC;
  always @(posedge clk, negedge reset) begin
    if(reset == 1)
      count <= 0;
      // If the load = 1, the data data_in is loaded into the counter.
    else if(load == 1)
      count <= data_in;
    else if(counter_on == 1)
    begin
      if(count_up == 1)
        count <= count + 1;
      else
        count <= count - 1;
    end
    // If the counter_on=counter_up=1, the counter is incremented, the Terminal carry
    // output TC=1 when the counter is in state 9.

    if(count_up == 1 && count == 9)
    begin
      TC <= 1;
      count <= 0;
    end
    // If the counter_on=1 and counter_up=0, the counter is decremented, the Terminal
    // carry output TC=1 when the counter is in state 0.
    else if(count_up == 0 && count == 0)
    begin
      TC <= 1;
      count <= 9;
    end
    else
      TC <= 0;
  end
endmodule
```

Simulation waveform UP Count



Simulation waveform DOWN Count



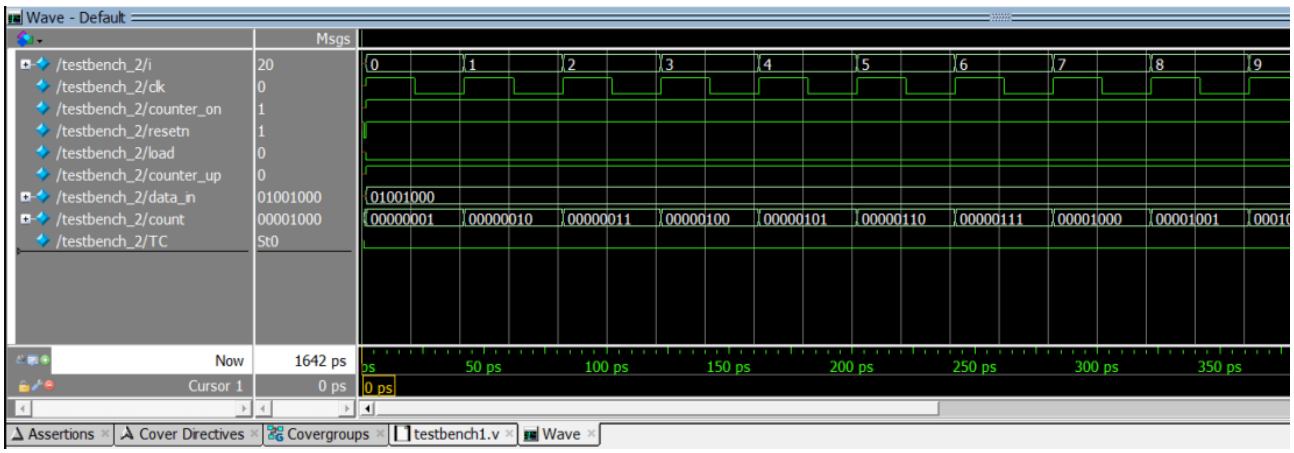
2. Decimal up down counter

Solution:

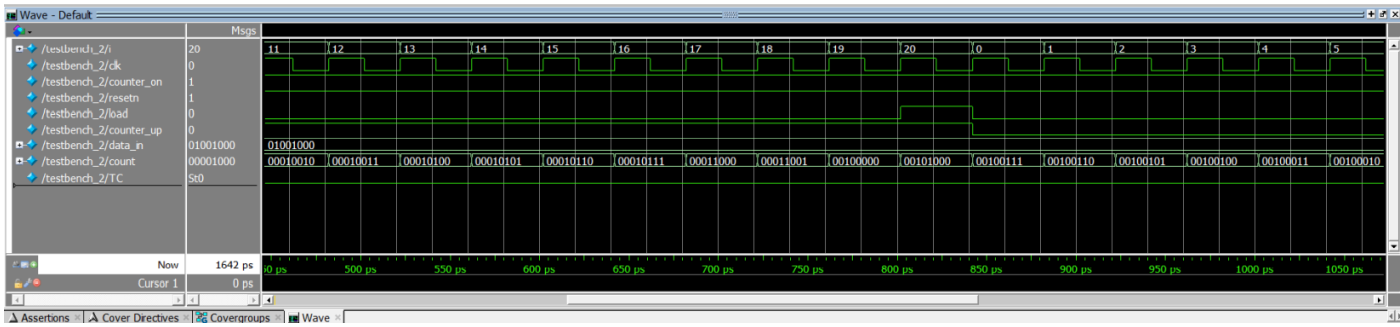
Verilog code

```
question2 > question2_f.v
1   module twoDigitDecadeCounter(input counter_up,
2     input load, input reset, input counter_on, input clk, input [7:0] data_in,
3     output [7:0] count, output TC);
4     wire n1;
5     decadeCounter cunit(counter_up, load, reset, counter_on, clk, data_in[3:0], count[3:0], n1);
6     decadeCounter cdigit(counter_up, load, reset, counter_on, n1, data_in[7:4], count[7:4], TC);
7   endmodule
```

Simulation UP count



Simulation DOWN count



3. N bit program counter

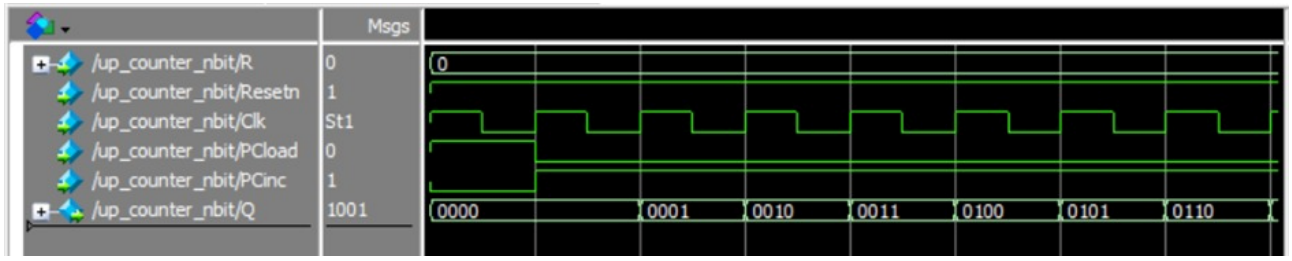
Solution:

Verilog code

question3 > question3.v

```
1   module up_counter_nbit(R, Resetn, Clk, PCinc, PCload,
2     parameter n = 4;
3     input [n-1:0] R;
4     input Resetn, Clk, PCload, PCinc;
5     output reg [n-1:0] Q;
6     always@(negedge Resetn, posedge Clk)
7       if (!Resetn)
8         Q <=0;
9       else if (PCload)
10        Q <= R;
11      else if (PCinc)
12        Q <= Q + 1;
13  endmodule
```

Simulation waveform



4. Shift Register 74194

Solution:

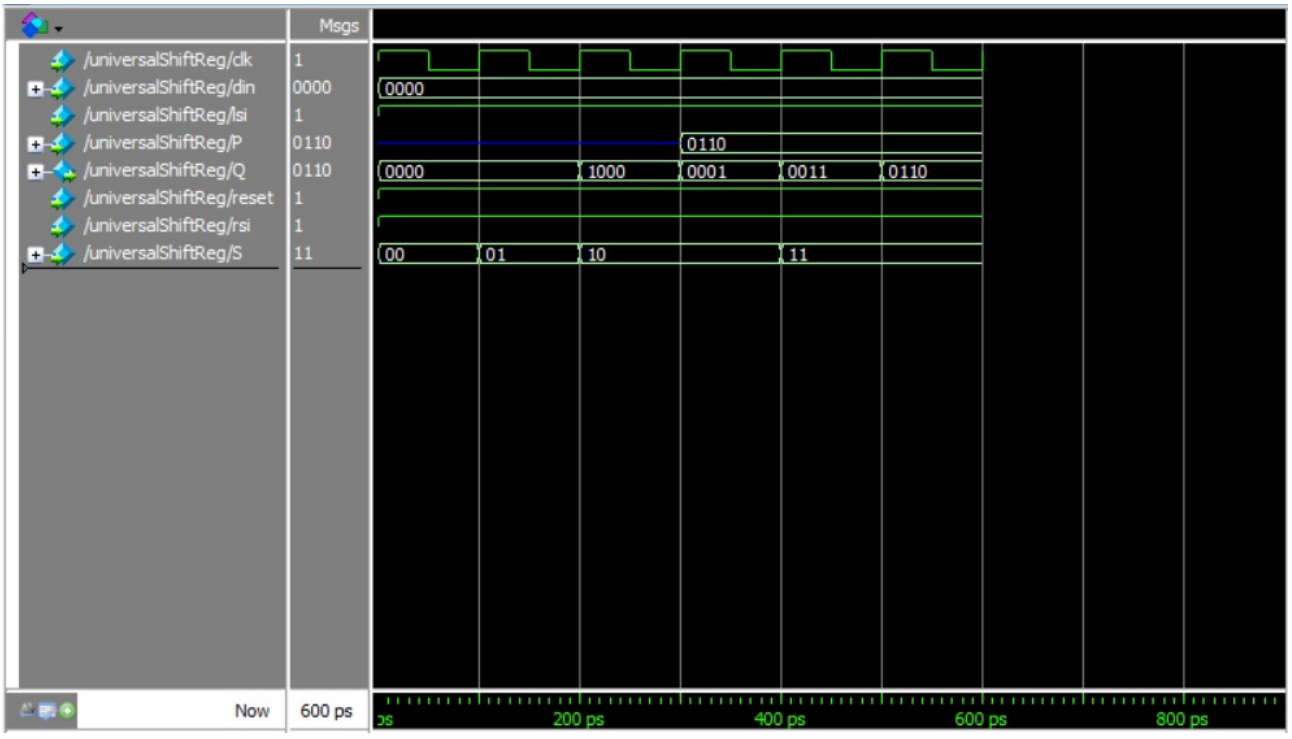
Verilog code

question4 > question4.v

```
1   module universalShiftReg(din,S,clk,Q, reset, lsi, rsi,P);
2     input [3:0]din;
3     input [3:0]P;
4     input [1:0]S;
5     input reset, clk, lsi, rsi;
6     output reg [3:0]Q;
7     assign Q = din;
8     wire [3:0]din;
9     always@(posedge clk, negedge reset)
10      if (!reset)
11        Q <= 4'b0000;
12      else
13        begin
14          Q <= din;
15          case (S)
16            // no change
17            2'b00:
18              begin
19                Q[3] <= Q[3];
20                Q[2] <= Q[2];
21                Q[1] <= Q[1];
22                Q[0] <= Q[0];
23              end
24            // shift right
25            2'b01:
26              begin
27                Q[3] <= rsi;
28                Q[2] <= Q[3];
29                Q[1] <= Q[2];
30                Q[0] <= Q[1];
31              end
32            // shift left
33            2'b10:
34              begin
35                Q[0] <= lsi;
36                Q[1] <= Q[0];
37                Q[2] <= Q[1];
```

```
25              2'b01:
26                begin
27                  Q[3] <= rsi;
28                  Q[2] <= Q[3];
29                  Q[1] <= Q[2];
30                  Q[0] <= Q[1];
31                end
32              // shift left
33              2'b10:
34                begin
35                  Q[0] <= lsi;
36                  Q[1] <= Q[0];
37                  Q[2] <= Q[1];
38                  Q[3] <= Q[2];
39                end
40              // load
41              2'b11:
42                begin
43                  Q[3] <= P[3];
44                  Q[2] <= P[2];
45                  Q[1] <= P[1];
46                  Q[0] <= P[0];
47                end
48            endcase
49        end
50    endmodule
```

Simulation waveform



5. Self-correction ring counter

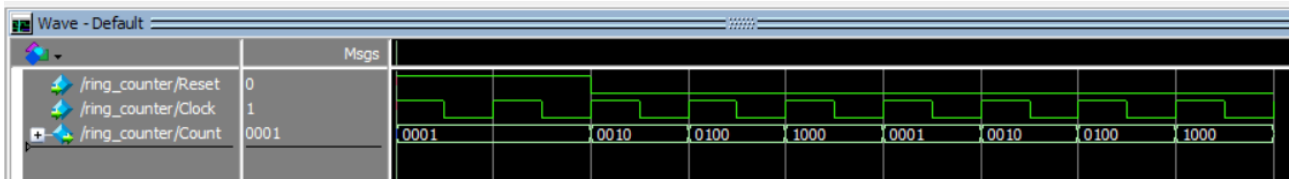
Solution:

Verilog code

question5 > question5.v

```
1   module ring_counter(
2       Clock,
3       Reset,
4       Count
5       );
6       input Clock;
7       input Reset;
8       output [3:0] Count;
9       reg [3:0] Count_temp;
10      always @(posedge(Clock),Reset)
11      begin
12          if(Reset == 1'b1 || Count == 4'b0000|| Count == 4'b1000)
13          begin
14              Count_temp = 4'b0001;
15          end
16          else if(Clock == 1'b1)
17          begin
18              Count_temp = {Count_temp[2:0],Count_temp[3]};
19          end
20      end
21      assign Count = Count_temp;
22  endmodule
23
```

Simulation waveform



6. Pseudo random sequence generator

Solution:

Verilog code

question6 > question6.v

```
1   module pseudoRandomSeqGenerator( resetn, clk, Q);
2     parameter n =8;
3     input resetn, clk;
4     output reg [n-1:0] Q;
5     always @(negedge resetn, posedge clk)
6     begin
7       if(!resetn) Q<={1'b1, {n-2{1'b0}}};
8       else begin
9       Q <= Q >> 1;
10      Q[n-1] <= Q[7] ^ Q[3] ^ Q[2] ^Q[1];
11    end
12    end
13  endmodule
14
```

Simulation waveform