ELL201
Assignment 2

Ritika Soni

2020MT10838

14th March 2022

Q1:

QN	Q(N+1)	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

State Table :

SN	Q3	Q2	Q1	Q0	Q3N	Q2N	Q1N	Q0N	S3	R3	S2	R2	S1	R1	S0	R0
0	0	0	0	0	0	0	0	1	0	X	0	X	0	X	1	0
1	0	0	0	1	0	0	1	1	0	X	0	X	1	0	X	0
3	0	0	1	1	0	0	1	0	0	X	0	X	X	0	0	1
2	0	0	1	0	0	1	1	0	0	X	1	0	X	0	0	X
6	0	1	1	0	0	1	1	1	0	X	X	0	X	0	1	0
7	0	1	1	1	0	1	0	1	0	X	X	0	0	1	X	0
5	0	1	0	1	0	1	0	0	0	X	X	0	0	X	0	1
4	0	1	0	0	1	1	0	0	1	0	X	0	0	X	0	X
12	1	1	0	0	1	1	0	1	X	0	X	0	0	X	1	0
13	1	1	0	1	1	1	1	1	X	0	X	0	1	0	X	0
15	1	1	1	1	1	1	1	0	X	0	X	0	X	0	0	1
14	1	1	1	0	1	0	1	0	X	0	0	1	X	0	0	X
10	1	0	1	0	1	0	1	1	X	0	0	X	X	0	1	0
11	1	0	1	1	1	0	0	1	X	0	0	X	0	1	X	0
9	1	0	0	1	1	0	0	0	X	0	0	X	0	X	0	1
8	1	0	0	0	0	0	0	0	0	1	0	X	0	X	0	X

We need 4 flip-flops to make a 4-bit gray code generator

K-maps for assigning values to each S and R achieving minimized expression for inputs.

S0 = Q1'Q2'Q3'+Q1Q2Q3'+Q1Q2Q3+Q1Q2'Q3

Q1 Q0 → Q3 Q2 ↓	**00**	**01**	**11**	**10**
00	1	X	0	0
01	0	0	X	1
11	1	X	0	0
10	0	0	X	1

R0 = Q1Q2'Q3'+Q1Q2Q3'+Q1'Q2Q3+Q1Q2'Q3

Q1 Q0 → Q3 Q2 ↓	**00**	**01**	**11**	**10**
00	0	0	1	X
01	X	1	0	0
11	0	0	1	X
10	X	1	0	0

S1 = Q0Q2'Q3'+Q0Q2Q3

Q1 Q0 → Q3 Q2 ↓	**00**	**01**	**11**	**10**
00	0	1	X	X
01	0	0	0	X
11	0	1	X	X
10	0	0	0	X

R1 = Q0Q2Q3'Q0Q2'Q3

Q1 Q0 → Q3 Q2 ↓	**00**	**01**	**11**	**10**
00	X	0	0	0
01	X	X	1	0
11	X	0	0	0
10	X	X	1	0

S2 = Q0'Q1Q3'

Q1 Q0 → Q3 Q2 ↓	**00**	**01**	**11**	**10**
00	0	0	0	1
01	X	X	X	X
11	X	X	X	0
10	0	0	0	0

R2 = Q0'Q1Q3

Q1 Q0 → Q3 Q2 ↓	**00**	**01**	**11**	**10**
00	X	X	X	X
01	0	0	0	0
11	0	0	0	1
10	X	X	X	X

S3 = Q0'Q1'Q2

Q1 Q0 → Q3 Q2 ↓	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	X	X	X	X
10	0	X	X	X

R3 = Q0'Q1'Q2'

Q1 Q0 → Q3 Q2 ↓	00	01	11	10
00	X	X	X	X
01	0	X	X	X
11	0	0	0	0
10	1	0	0	0

Code for Synchronous 4-bit Gray-Code Counter:

que_1.v

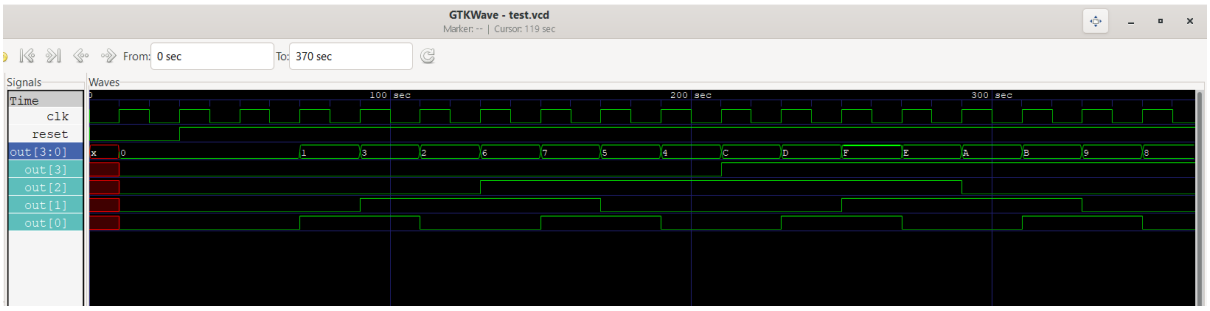
```
module gray_ctr
  (input    clk,
   input    rstn,
   output reg [3:0] out);

   reg [3:0] temp;

   always @ (posedge clk) begin
      if (!rstn) begin
      temp <= 0;
         out <= 0;
      end else begin
      temp <= temp + 1;

`ifdef for_loop
      for (int i = 0; i < 3; i= i+1) begin
         out[i] <= temp[i+1] ^ temp[i];
      end
      out[3] <= temp[3];
`else
         out <= {temp[3], temp[3:1] ^ temp[2:0]};
`endif
    end
   end
endmodule
```

que_1_tb.v

```v
module q1;
  reg clk;
  reg reset;
  wire [3:0] out;

  gray_ctr q1 (.clk(clk), .rstn(reset), .out(out));

  always #10 clk = ~clk;

  initial begin
    {clk, reset} <= 0;
    $dumpfile("q1.vcd");
    $dumpvars(0,q1);
    $monitor ("Time = %0t  Reset = %b  Output = %b ", $time, reset, out);

    repeat(2) @ (posedge clk);
    reset <= 1;
    repeat(19) @ (posedge clk);
    $finish;
  end
endmodule
```

```
I give up.
PS C:\Users\Dell\Desktop\ELL201_ASS2> iverilog -o my_file.v que_1.v que_1_tb.v
PS C:\Users\Dell\Desktop\ELL201_ASS2> vvp my_file.v
VCD info: dumpfile q1.vcd opened for output.
Time = 0  Reset = 0  Output = xxxx
Time = 10  Reset = 0  Output = 0000
Time = 30  Reset = 1  Output = 0000
Time = 70  Reset = 1  Output = 0001
Time = 90  Reset = 1  Output = 0011
Time = 110  Reset = 1  Output = 0010
Time = 130  Reset = 1  Output = 0110
Time = 150  Reset = 1  Output = 0111
Time = 170  Reset = 1  Output = 0101
Time = 190  Reset = 1  Output = 0100
Time = 210  Reset = 1  Output = 1100
Time = 230  Reset = 1  Output = 1101
Time = 250  Reset = 1  Output = 1111
Time = 270  Reset = 1  Output = 1110
Time = 290  Reset = 1  Output = 1010
Time = 310  Reset = 1  Output = 1011
Time = 330  Reset = 1  Output = 1001
Time = 350  Reset = 1  Output = 1000
Time = 370  Reset = 1  Output = 0000
Time = 390  Reset = 1  Output = 0001
que_1_tb.v:19: $finish called at 410 (1s)
Time = 410  Reset = 1  Output = 0011
PS C:\Users\Dell\Desktop\ELL201_ASS2>
```

Gtkwave output :



Q2:

4 D flip flops are required .

Assigned values to Ds using state table in the question :

D0=Q1

Q1 Q0 → Q3 Q2 ↓	**00**	**01**	**11**	**10**
00	x	0	1	1
01	0	0	1	1
11	0	0	1	1
10	0	0	1	1

D1= Q2

Q1 Q0 → Q3 Q2 ↓	**00**	**01**	**11**	**10**
00	x	0	1	1
01	1	1	1	1
11	1	1	1	1
10	0	0	0	0

D2= Q3

Q1 Q0 → Q3 Q2 ↓	**00**	**01**	**11**	**10**
00	x	0	1	1
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

D3= Q0'Q1 + Q0Q1' = Q0 XOR Q1

Q1 Q0 → Q3 Q2 ↓	**00**	**01**	**11**	**10**
00	x	0	1	1
01	1	1	1	1
11	1	1	1	1
10	0	0	0	0

My Entry no. **is 2020MT10838** so **X4 = 8**

b1 b2 b3 b4 = 1 0 0 0

No, this counter does not take all 16 bits . This counter is taking the XOR of the Q0 (XOR) Q1 and this result is put as the MSB of the next cycle, while Q3, Q2, and Q1 and shifted towards the right. On observing we will notice that this counter does not give " 0 0 0 0 " as output as it will keep on taking XOR of last two bits and shift the bits towards the right replacing first bit with output of XOR. Finally it will reach the value from where we had started.

Code for Synchronous Ring Counter :

q2_farzi.v

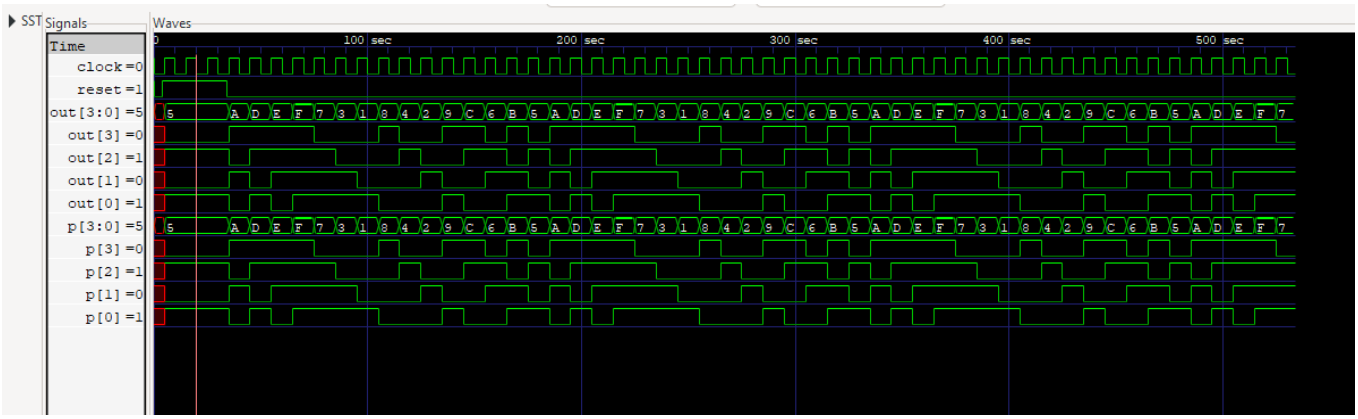
```
1   module ring_counter (
2       input clock,
3       input reset,
4       output [3:0] out
5     );
6
7     reg[3:0] temp;
8
9       always @(posedge clock)
10        if (reset)
11          temp = 4'b1000;
12        else
13          begin
14           temp = {temp[0]^temp[1],temp[3:1]};
15          end
16      assign out = temp;
17
18    endmodule
```

q2_farzi_tb.v

```v
`timescale 1ns / 1ps
module q2;
  reg clock;
  reg reset;
  wire[3:0] out;
  ring_counter r2 (
    .clock(clock),
    .reset(reset),
    .out(out)
  );

 always #10 clock = ~clock;

  initial begin
  clock = 0;
  reset = 0;
  #5 reset = 1;
  #20 reset = 0;
  #300 $finish;
   end

    initial begin
      $dumpfile("q2.vcd");
      $dumpvars(0,q2);
      $monitor($time, " Clock = %1b , Reset = %1b , Output = %4b ",clock,reset,out);
    end

endmodule
```

```
   5 Clock = 0 , Reset = 1 , Output = xxxx
  10 Clock = 1 , Reset = 1 , Output = 1000
  20 Clock = 0 , Reset = 1 , Output = 1000
  25 Clock = 0 , Reset = 0 , Output = 1000
  30 Clock = 1 , Reset = 0 , Output = 0100
  40 Clock = 0 , Reset = 0 , Output = 0100
  50 Clock = 1 , Reset = 0 , Output = 0010
  60 Clock = 0 , Reset = 0 , Output = 0010
  70 Clock = 1 , Reset = 0 , Output = 1001
  80 Clock = 0 , Reset = 0 , Output = 1001
  90 Clock = 1 , Reset = 0 , Output = 1100
 100 Clock = 0 , Reset = 0 , Output = 1100
 110 Clock = 1 , Reset = 0 , Output = 0110
 120 Clock = 0 , Reset = 0 , Output = 0110
 130 Clock = 1 , Reset = 0 , Output = 1011
 140 Clock = 0 , Reset = 0 , Output = 1011
 150 Clock = 1 , Reset = 0 , Output = 0101
 160 Clock = 0 , Reset = 0 , Output = 0101
 170 Clock = 1 , Reset = 0 , Output = 1010
 180 Clock = 0 , Reset = 0 , Output = 1010
 190 Clock = 1 , Reset = 0 , Output = 1101
 200 Clock = 0 , Reset = 0 , Output = 1101
 210 Clock = 1 , Reset = 0 , Output = 1110
 220 Clock = 0 , Reset = 0 , Output = 1110
 230 Clock = 1 , Reset = 0 , Output = 1111
 240 Clock = 0 , Reset = 0 , Output = 1111
 250 Clock = 1 , Reset = 0 , Output = 0111
 260 Clock = 0 , Reset = 0 , Output = 0111
 270 Clock = 1 , Reset = 0 , Output = 0011
 280 Clock = 0 , Reset = 0 , Output = 0011
 290 Clock = 1 , Reset = 0 , Output = 0001
 300 Clock = 0 , Reset = 0 , Output = 0001
 310 Clock = 1 , Reset = 0 , Output = 1000
 320 Clock = 0 , Reset = 0 , Output = 1000
```

Output of Gtkwave:



Thank You