ELL201 Digital Electronics
Lab Report

Atif Anwer 2020EE10479

March 14, 2022

Verilog Assignment 2

1 Objectives

1. To design a synchronous 4-bit Gray Code Counter using SR Flip Flops.
2. To design a synchronous ring counter using D Flip-Flops.

2 Part 1 : Synchronous 4-bit Gray Code Counter using SR Flip Flops

2.1 State Transition in SR Flip-Flops

State Transion	S	R
$0 \rightarrow 0$	0	X
$0 \rightarrow 1$	1	0
$1 \rightarrow 0$	0	1
$1 \rightarrow 1$	X	0

Table 1: State Transition in SR Flip-Flop

2.2 State Table of SR Flip-Flop

Since a 4-bit grey code shows 16 states, there is a requirement of 4 Synchronized SR Flip-Flops.

Q_3	Q_2	Q_1	Q_0	S_3	R_3	S_2	R_2	S_1	R_1	S_0	R_0
0	0	0	0	0	x	0	x	0	x	1	0
0	0	0	1	0	x	0	x	1	0	x	0
0	0	1	1	0	x	0	x	x	0	0	1
0	0	1	0	0	x	1	0	x	0	0	x
0	1	1	0	0	x	x	0	x	0	1	0
0	1	1	1	0	x	x	0	0	1	x	0
0	1	0	1	0	x	x	0	0	x	0	1
0	1	0	0	1	0	x	0	0	x	0	x
1	1	0	0	x	0	x	0	0	x	1	0
1	1	0	1	x	0	x	0	1	0	x	0
1	1	1	1	x	0	x	0	x	0	0	1
1	1	1	0	x	0	0	1	x	0	0	x
1	0	1	0	x	0	0	x	x	0	1	0
1	0	1	1	x	0	0	x	0	1	x	0
1	0	0	1	x	0	0	x	0	x	0	1
1	0	0	0	0	1	0	x	0	x	0	x

Table 2: State Table of 4-bit Grey Code Counter

2.3 Solving Karnaugh Maps for $S_3, R_3, S_2, R_2, S_1, R_1, S_0, R_0,$

2.3.1 S_0

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	1	X	0	0
01	0	0	X	1
11	1	X	0	0
10	0	0	X	1

Minimized Expression : $S_0 = \overline{Q_3}\,\overline{Q_2}\,\overline{Q_1} + \overline{Q_3}\,Q_2\,Q_1 + Q_3\,\overline{Q_2}\,Q_1 + Q_3\,Q_2\,\overline{Q_1}$

2.3.2 R_0

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0	0	1	X
01	X	1	0	0
11	0	0	1	X
10	X	1	0	0

Minimized Expression : $R_0 = \overline{Q_3}\ \overline{Q_2}\ Q_1 + \overline{Q_3}\ Q_2\ \overline{Q_1} + Q_3\ \overline{Q_2}\ \overline{Q_1} + Q_3\ Q_2\ Q_1$

2.3.3 S_1

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0	1	X	X
01	0	0	0	X
11	0	1	X	X
10	0	0	0	X

Minimized Expression : $S_1 = \overline{Q_3}\ \overline{Q_2}\ Q_0 + Q_3\ Q_2\ Q_0$

2.3.4 R_1

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	X	0	0	0
01	X	X	1	0
11	X	0	0	0
10	X	X	1	0

Minimized Expression : $R_1 = \overline{Q_3}\ Q_2\ Q_0 + Q_3\ \overline{Q_2}\ Q_0$

2.3.5 S_2

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0	0	0	1
01	X	X	X	X
11	X	X	X	0
10	0	0	0	0

Minimized Expression : $S_2 = \overline{Q_3}\, Q_1\, \overline{Q_0}$

2.3.6 R_2

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	X	X	X	0
01	0	0	0	0
11	0	0	0	1
10	X	X	X	X

Minimized Expression : $R_2 = Q_3\, Q_1\, \overline{Q_0}$

2.3.7 S_3

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	X	X	X	X
10	0	X	X	X

Minimized Expression : $S_3 = Q_2\, \overline{Q_1}\, \overline{Q_0}$

2.3.8 R_3

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	X	X	X	X
01	0	X	X	X
11	0	0	0	0
10	1	0	0	0

Minimized Expression : $R_3 = \overline{Q_2}\,\overline{Q_1}\,\overline{Q_0}$

2.4 Code

```
module grey_counter (clk,reset,count);

    input clk,reset;
    output[3:0] count;

    wire[3:0] S,R;
    wire[3:0] Q,Qbar;

    assign S[0] =   ((~Q[3]) & (~Q[2]) & (~Q[1]))
                  | ((~Q[3]) & (Q[2]) & (Q[1]))
                  | ((Q[3]) & (Q[2]) & (~Q[1]))
                  | ((Q[3]) & (~Q[2]) & (Q[1]));

    assign R[0] =   ((~Q[3]) & (~Q[2]) & (Q[1]))
                  | ((~Q[3]) & (Q[2]) & (~Q[1]))
                  | ((Q[3]) & (Q[2]) & (Q[1]))
                  | ((Q[3]) & (~Q[2]) & (~Q[1]));

    assign S[1] =   ((~Q[3]) & (~Q[2]) & (Q[0]))
                  | ((Q[3]) & (Q[2]) & (Q[0]));
    assign R[1] =   ((~Q[3]) & (Q[2]) & (Q[0]))
                  | ((Q[3]) & (~Q[2]) & (Q[0]));

    assign S[2] =   ((~Q[3]) & (Q[1]) & (~Q[0]));
    assign R[2] =   ((Q[3]) & (Q[1]) & (~Q[0]));
    assign S[3] =   ((Q[2]) & (~Q[1]) & (~Q[0]));
    assign R[3] =   ((~Q[2]) & (~Q[1]) & (~Q[0]));

    sr_flip_flop FF1(.s(S[0]),.r(R[0]),.clk(clk),.reset(reset),
                  .q(Q[0]),.qbar(Qbar[0]));

    sr_flip_flop FF2(.s(S[1]),.r(R[1]),.clk(clk),.reset(reset),
                  .q(Q[1]),.qbar(Qbar[1]));

    sr_flip_flop FF3(.s(S[2]),.r(R[2]),.clk(clk),.reset(reset),
                  .q(Q[2]),.qbar(Qbar[2]));

```

```
    sr_flip_flop FF4(.s(S[3]),.r(R[3]),.clk(clk),.reset(reset),
                     .q(Q[3]),.qbar(Qbar[3]));

    assign count = Q;

endmodule

module sr_flip_flop(s,r,clk,reset, q, qbar);

    input s,r,clk,reset;
    output reg q, qbar;

    always@(reset,posedge clk)
        begin
        if (reset == 0) begin
            q <= 0;
        end
        else
        begin

        if(s == 1)
            begin
            q <= 1;
            qbar <= 0;
            end
        else if(r == 1)
            begin
            q <= 0;
            qbar <=1;
            end
        else if(s == 0 & r == 0)
            begin
            q <= q;
            qbar <= qbar;
            end
        end
        end
endmodule
```

2.5 Test Bench

```
module greycode_testbench;

reg clk;
reg reset;
wire [3:0] count;

grey_counter G(.clk(clk), .reset(reset), .count(count));

initial begin
    clk=0;
    forever #10 clk=~clk;
end

initial begin

    $monitor ("clock %b, count = %b",clk,count);
    $dumpfile("TestBench.vcd");
    $dumpvars (0, greycode_testbench);
```

```
19
20      reset <= 0;
21      #10;
22      reset <= 1;
23
24      repeat (32)
25      begin
26          #10;
27      end
28
29      $finish;
30  end
31  endmodule
```

2.6 vvp execution output

```
VCD info: dumpfile TestBench.vcd opened for output.
clock 0, count = 0000
clock 1, count = 0001
clock 0, count = 0001
clock 1, count = 0011
clock 0, count = 0011
clock 1, count = 0010
clock 0, count = 0010
clock 1, count = 0110
clock 0, count = 0110
clock 1, count = 0111
clock 0, count = 0111
clock 1, count = 0101
clock 0, count = 0101
clock 1, count = 0100
clock 0, count = 0100
clock 1, count = 1100
clock 0, count = 1100
clock 1, count = 1101
clock 0, count = 1101
clock 1, count = 1111
clock 0, count = 1111
clock 1, count = 1110
clock 0, count = 1110
clock 1, count = 1010
clock 0, count = 1010
clock 1, count = 1011
clock 0, count = 1011
clock 1, count = 1001
clock 0, count = 1001
clock 1, count = 1000
clock 0, count = 1000
clock 1, count = 0000
clock 0, count = 0000
```

2.7 Waveform on GTKwave

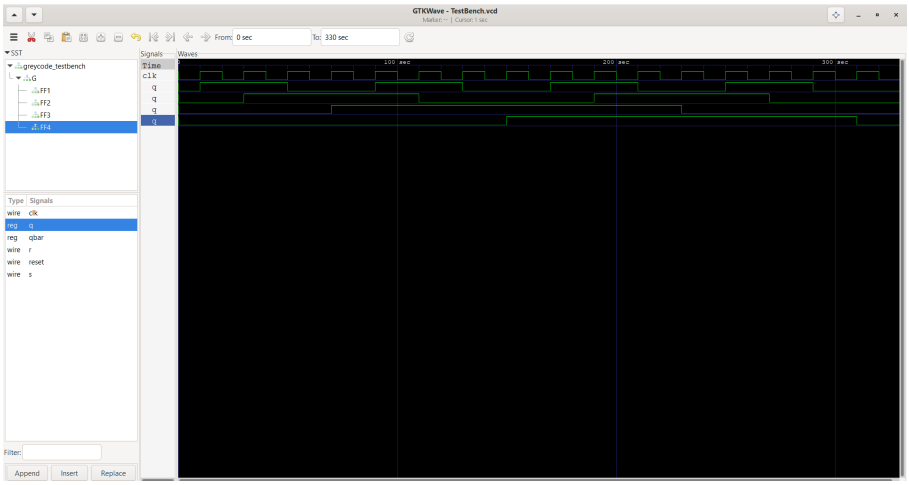

Figure 1: Waveform of simulation

3 Part 2 : Synchronous 4-bit Ring Counter using D Filp-Flops

3.1 State Transition in D Flip-Flops

State Transion	D
$0 \rightarrow 0$	0
$0 \rightarrow 1$	1
$1 \rightarrow 0$	0
$1 \rightarrow 1$	1

Table 3: State Transition in D Flip-Flop

3.2 State Table of SR Flip-Flop

Since a 4-bits can show 15 non-zero states, there is a requirement of 4 Synchronized D Flip-Flops.

Q_3	Q_2	Q_1	Q_0	D_3	D_2	D_1	D_0
0	0	0	1	1	0	0	0
1	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0
0	0	1	0	1	0	0	1
1	0	0	1	1	1	0	0
1	1	0	0	0	1	1	0
0	1	1	0	1	0	1	1
1	0	1	1	0	1	0	1
0	1	0	1	1	0	1	0
1	0	1	0	1	1	0	1
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	1	1	1
0	1	1	1	0	0	1	1
0	0	1	1	0	0	0	1

Table 4: State Table of 4-bit Ring Counter

3.3 Solving Karnaugh Map for D_0, D_1, D_2, D_3

3.3.1 D_0

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	X	0	1	1
01	0	0	1	1
11	0	0	1	1
10	0	0	1	1

Minimized Expression : $D_0 = Q_1$

3.3.2 D_1

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	X	0	0	0
01	1	1	1	1
11	1	1	1	1
10	0	0	0	0

Minimized Expression : $D_2 = Q_2$

3.3.3 D_2

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	X	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

Minimized Expression : $D_2 = Q_3$

3.3.4 D_3

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	X	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

Minimized Expression : $S_0 = \overline{Q_1}Q_0 + Q_1\overline{Q_0} = Q_1 \oplus Q_0$

3.4 Circuit Diagram

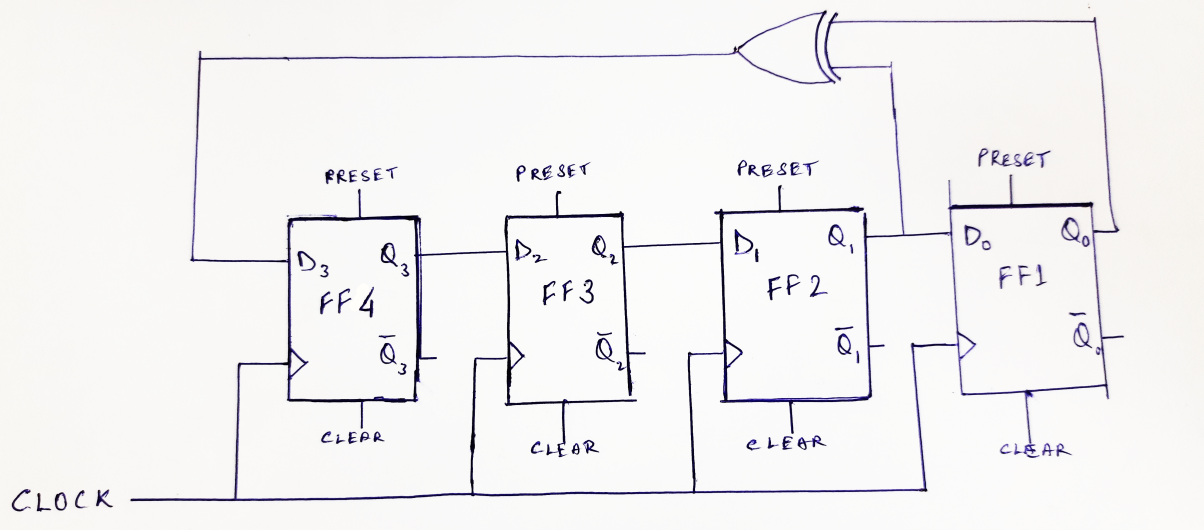

Figure 2: 4-Bit Synchronous Ring Counter

3.5 Code

```
module ring_counter (clk,override,count);

    input clk,override;
    output[3:0] count;

    wire[3:0] D;
    wire[3:0] Q,Qbar;

    assign D[0] = Q[1];
    assign D[1] = Q[2];
    assign D[2] = Q[3];
    assign D[3] = Q[1]^Q[0];

    D_flip_flop FF1(.D(D[0]),.clk(clk),.reset(1'b1),.preset(override),
                    .q(Q[0]),.qbar(Qbar[0]));

    D_flip_flop FF2(.D(D[1]),.clk(clk),.reset(override),.preset(1'b1),
                    .q(Q[1]),.qbar(Qbar[1]));

    D_flip_flop FF3(.D(D[2]),.clk(clk),.reset(override),.preset(1'b1),
                    .q(Q[2]),.qbar(Qbar[2]));

    D_flip_flop FF4(.D(D[3]),.clk(clk),.reset(1'b1),.preset(override),
                    .q(Q[3]),.qbar(Qbar[3]));

    assign count = Q;

endmodule

module D_flip_flop(D,clk,reset,preset, q, qbar);

    input D,clk,reset,preset;
    output reg q, qbar;

    always@(preset,reset,posedge clk)
        begin
        if (reset == 0) begin
            q <= 0;
        end
        else if (preset == 0) begin
            q <= 1;
        end
        else
        begin

        if(D == 1)
            begin
            q <= 1;
            qbar <= 0;
            end
        else if(D == 0)
            begin
            q <= 0;
            qbar <=1;
            end

```

```
        end
    end
endmodule
```

3.5.1 Test Bench

```
module ringcounter_testbench;

reg clk;
reg reset;
wire [3:0] count;

ring_counter G(.clk(clk), .override(reset), .count(count));

initial begin
    clk=0;
    forever #10 clk=~clk;
end

initial begin
    $monitor ("clock %b, count = %b",clk,count);
    $dumpfile("TestBench.vcd");
    $dumpvars (0, ringcounter_testbench);

    reset <= 0;
    #10;
    reset <=1;

    repeat (30)
    begin
        #10;
    end

    $finish;
end
endmodule
```

3.5.2 vvp execution output

```
VCD info: dumpfile TestBench.vcd opened for output.
clock 0, count = 1001
clock 1, count = 1100
clock 0, count = 1100
clock 1, count = 0110
clock 0, count = 0110
clock 1, count = 1011
clock 0, count = 1011
clock 1, count = 0101
clock 0, count = 0101
clock 1, count = 1010
clock 0, count = 1010
clock 1, count = 1101
clock 0, count = 1101
clock 1, count = 1110
clock 0, count = 1110
clock 1, count = 1111
clock 0, count = 1111
```

```
clock 1, count = 0111
clock 0, count = 0111
clock 1, count = 0011
clock 0, count = 0011
clock 1, count = 0001
clock 0, count = 0001
clock 1, count = 1000
clock 0, count = 1000
clock 1, count = 0100
clock 0, count = 0100
clock 1, count = 0010
clock 0, count = 0010
clock 1, count = 1001
clock 0, count = 1001
```

It is observed that the counter returns to the start state after 15 clock cycles($2^4 - 1$, Excluding 0000) irrespective of the value of start state.

3.6 Waveform on GTKwave

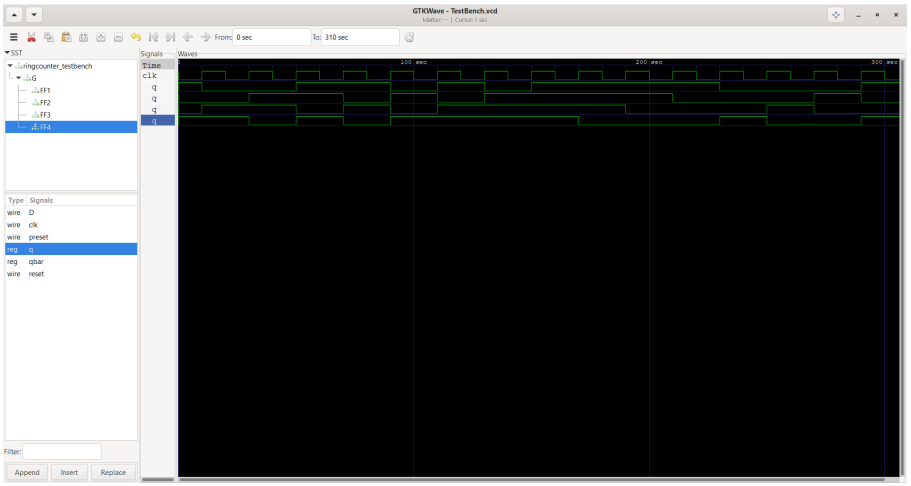

Figure 3: Waveform of simulation