ELP201 Lab Report 2

Adit Malhotra

2020EE10458

1 Synchronous 4-bit Gray-Code Counter

1.1 Counter Scheme

The scheme that my counter is going to follow would be $0 \to 1 \to 3 \to 2 \to 6 \to 7 \to 5 \to 4 \to 12 \to 13 \to 15 \to 14 \to 10 \to 11 \to 9 \to 8 \to 0$.

Q_3	Q_2	Q_1	Q_0	Decimal Num
0	0	0	0	0
0	0	0	1	1
0	0	1	1	3
0	0	1	0	2
0	1	1	0	6
0	1	1	1	7
0	1	0	1	5
0	1	0	0	4
1	1	0	0	12
1	1	0	1	13
1	1	1	1	15
1	1	1	0	14
1	0	1	0	10
1	0	1	1	11
1	0	0	1	9
1	0	0	0	8

Since total number of states $= 16 = 2^4$, thus total 4 SR flipflops would be required.

1.2 Simplification using K-Maps

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

Figure 1: K-Map for S_0

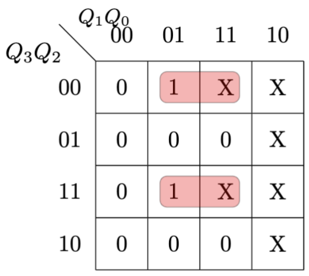
Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0	0	1	X
01	X	1	0	0
11	0	0	1	X
10	X	1	0	0

Figure 2: K-Map for R_0

From the k-maps, for first flipflop, the simplified expressions are

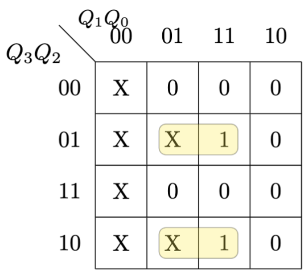
$$S_0 = \bar{Q}_3\bar{Q}_2\bar{Q}_1 + \bar{Q}_3Q_2Q_1 + \bar{Q}_1Q_2Q_3 + \bar{Q}_2Q_3Q_1 \implies S_0 = \bar{Q}_3 \oplus \bar{Q}_2 \oplus \bar{Q}_1$$

$$R_0 = \bar{Q}_3\bar{Q}_2Q_1 + \bar{Q}_1\bar{Q}_2Q_3 + \bar{Q}_3\bar{Q}_1Q_2 + Q_3Q_2Q_1 \implies R_0 = Q_3 \oplus Q_2 \oplus Q_1$$



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

Figure 3: K-Map for S_1



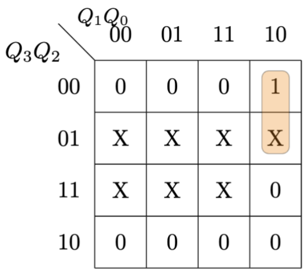
$Q_3Q_2 \backslash Q_1Q_0$	00	01	11	10
00	X	0	0	0
01	X	X	1	0
11	X	0	0	0
10	X	X	1	0

Figure 4: K-Map for R_1

From the k-maps, for second flipflop, the simplified expressions are

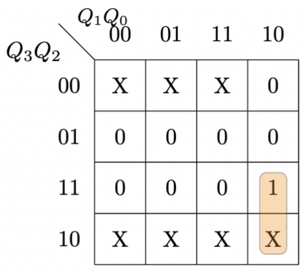
$$S_1 = Q_3Q_2Q_0 + \bar{Q}_3\bar{Q}_2Q_0 \implies S_1 = Q_0 \cdot (\bar{Q}_3 \oplus Q_2)$$

$$R_1 = \bar{Q}_3Q_2Q_0 + \bar{Q}_2Q_3Q_0 \implies R_1 = Q_0 \cdot (Q_3 \oplus Q_2)$$



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

Figure 5: K-Map for S_2



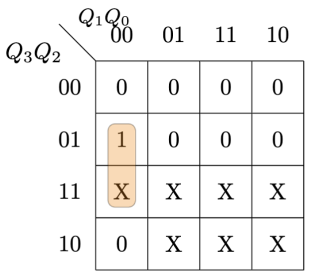
$Q_3Q_2 \backslash Q_1Q_0$	00	01	11	10
00	X	X	X	0
01	0	0	0	0
11	0	0	0	1
10	X	X	X	X

Figure 6: K-Map for R_2

From the k-maps, for third flipflop, the simplified expressions are

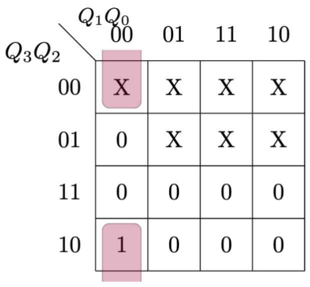
$$S_2 = \bar{Q}_0Q_1\bar{Q}_3$$

$$R_2 = \bar{Q}_0Q_1Q_3$$



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

Figure 7: K-Map for S_3



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

Figure 8: K-Map for R_3

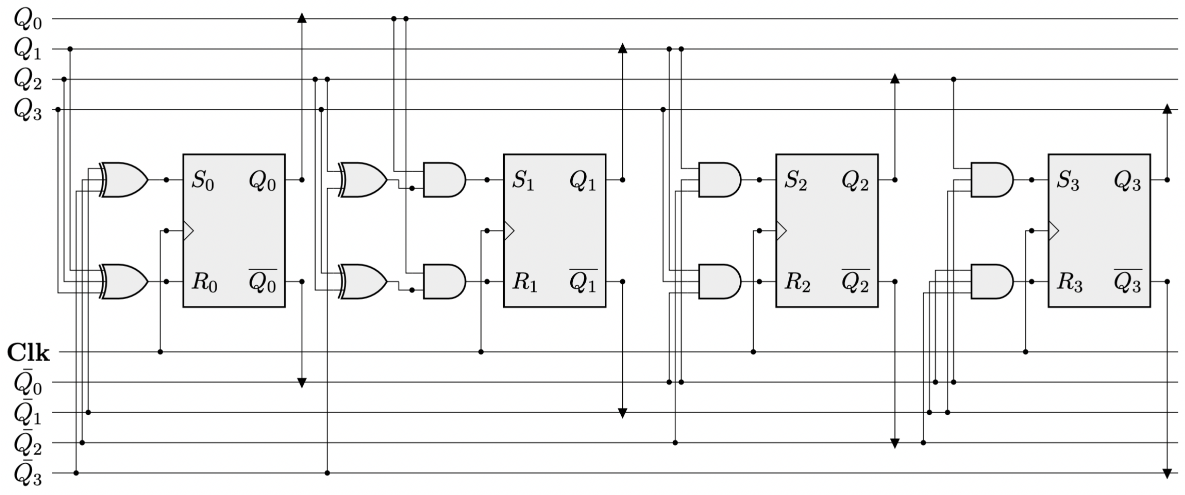
From the k-maps, for third flipflop, the simplified expressions are

$$S_3 = Q_2\bar{Q}_0\bar{Q}_1$$

$$R_3 = \bar{Q}_2\bar{Q}_0\bar{Q}_1$$

1.3 Circuit Diagram

The circuit diagram for the counter using the simplified expressions from above, is:



Q_0
Q_1
Q_2
Q_3
Clk
$\bar{Q}_0$
$\bar{Q}_1$
$\bar{Q}_2$
$\bar{Q}_3$
S_0 Q_0 R_0 $\overline{Q_0}$
S_1 Q_1 R_1 $\overline{Q_1}$
S_2 Q_2 R_2 $\overline{Q_2}$
S_3 Q_3 R_3 $\overline{Q_3}$

Figure 9: Circuit Diagram

1.4 Verilog Simulation

The code is implemented in such a way that flipflops behave as postive edge triggered not latches.

1.4.1 Code

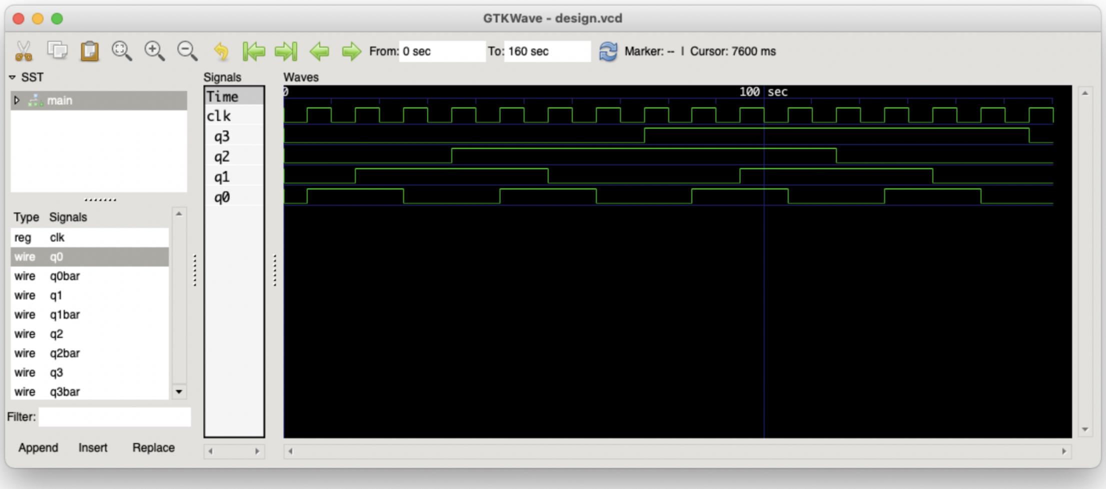
The code for the implementation of the SR flipflop file (design.v) :

```verilog
module SRFlipFlop(S,R,clock,Q,Qbar);
input S,R,clock;
output reg Q=0,Qbar=1;

always@(posedge clock)
begin
  if(S & ~R)
    begin
      Q <= 1;
      Qbar <= 0;
    end
  else if(~S & R)
    begin
      Q <= 0;
      Qbar <= 1;
    end
  else if(~S & ~R)
    begin
      Q <= Q;
      Qbar <= Qbar;
    end
  else if(S & R)
    begin
      Q <= 1'bx;
      Qbar <= 1'bx;
    end
end
endmodule
```

The code for the testbench file (main.v) :

```verilog
`timescale 1s/100ms
`include "design.v"

module main();

reg clk;
wire q0,q0bar,q1,q1bar,q2,q2bar,q3,q3bar;

//Instantation of flip flops
SRFlipFlop ff0(q1bar ^ q2bar ^ q3bar,q1 ^ q2 ^ q3,clk,q0,q0bar);
SRFlipFlop ff1(q0 & (q3bar ^ q2),q0 & (q3 ^ q2),clk,q1,q1bar);
SRFlipFlop ff2(q3bar & q1 & q0bar,q3 & q1 & q0bar,clk,q2,q2bar);
SRFlipFlop ff3(q2 & q1bar & q0bar,q2bar & q1bar & q0bar,clk,q3,q3bar);

initial
begin
$monitor("CLK=%b,q3=%b,q2=%b,q1=%b,q0=%b",clk,q3,q2,q1,q0);
$dumpfile("design.vcd");
$dumpvars(0,main);
  clk=0; //intitially clock=0;
  #160; //simulation time (enough for all states to occur)
  $finish;
end
always
  #5 clk = ~clk; //changing clock after every 5s
endmodule
```

1.4.2 Waveform plot



2 Synchronous Ring Counter

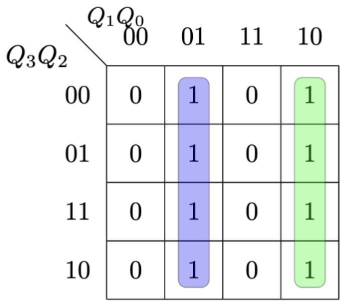
2.1 Introduction

From the table it can be observed that there are total 15 different states of the counter. Also since $2^4 \geq 15$, hence total number of D flip-flops required would be 4.

Since the connections for the ring counter are such that for all intermediate flipflops, output of one flipflop is the input for next flipflop, and for the flipflop it can take any combination of the current states.

Thus from the K-map for D_3, the simplified expression obtained is :

$$D_3 = Q_1\bar{Q}_0 + Q_0\bar{Q}_1 \implies D_3 = Q_1 \oplus Q_0$$



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

Figure 10: K-Map for D_3

2.2 Circuit Diagram

Thus the cicuit diagram for this ring counter would be:

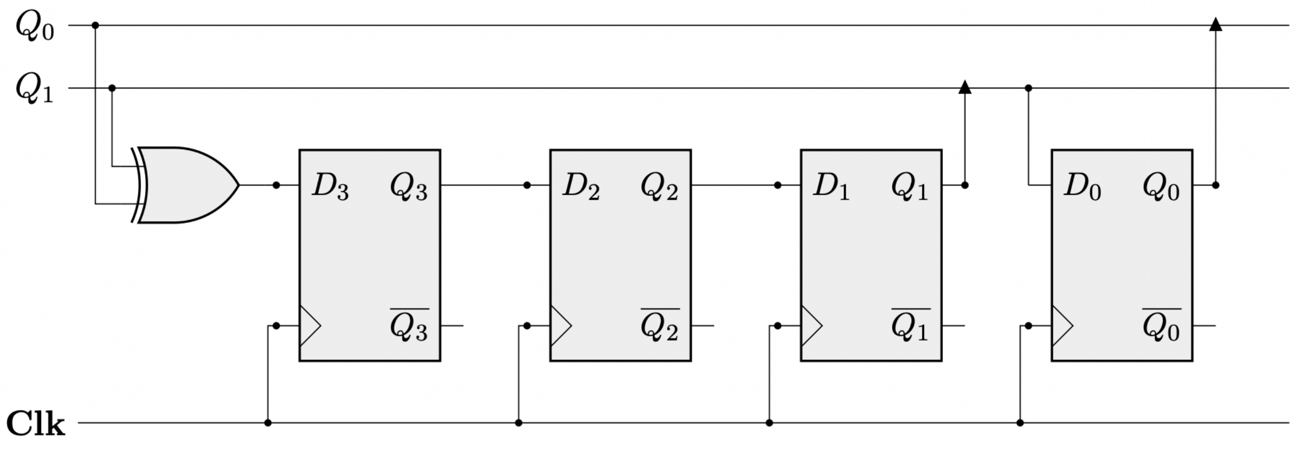


Figure 11: Circuit Diagram

2.3 Verilog Simulation

According to my entry number, the initial state should be:

$$Q_3 = 1\ , Q_2 = 0\ , Q_1 = 0\ , Q_0 = 0$$

The preset parameter is here used to input the initial condition to flipflop. It is configured as low active (hence when preset=0, then $Q = 1$). The D flipflops are coded in such a way that they behave as positive edge triggered not latches.

2.3.1 Code

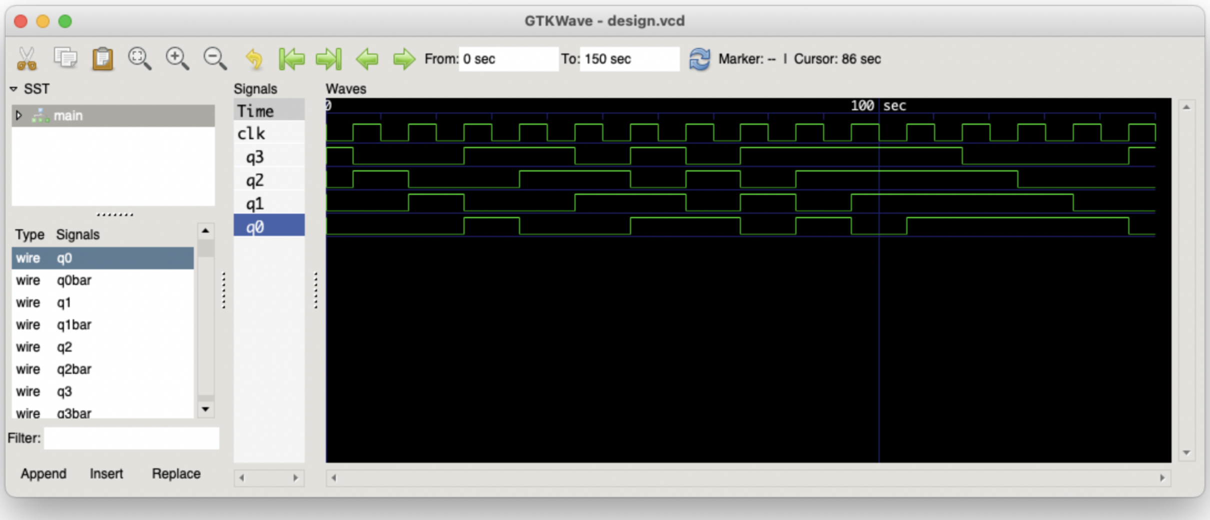
The code for the implementation of the D flipflop file (design.v):

```verilog
module DFlipFlop(D,clock,preset,Q,Qbar);
input D,clock,preset;
output reg Q,Qbar;

initial
begin
  if(~preset)
  begin
    Q=1;Qbar=0;
  end
  else
  begin
    Q=0;Qbar=1;
  end
end

always@(posedge clock)
begin
  Q <= D;
  Qbar <= ~D;
end
endmodule
```

The code for the testbench file (main.v):

```verilog
`timescale 1s/100ms
`include "design.v"

module main();

reg clk;
wire q0,q1,q2,q3,q3bar,q2bar,q1bar,q0bar;

//Instantating the flip flops with respective intial values
DFlipFlop ff3(q1^q0,clk,1'b0,q3,q3bar);
DFlipFlop ff2(q3,clk,1'b1,q2,q2bar);
DFlipFlop ff1(q2,clk,1'b1,q1,q1bar);
DFlipFlop ff0(q1,clk,1'b1,q0,q0bar);

initial
begin
$monitor("CLK=%b,q3=%b,q2=%b,q1=%b,q0=%b",clk,q3,q2,q1,q0);
$dumpfile("design.vcd");
$dumpvars(0,main);
  clk=0; //intitially clock=0;
  #150; //simulation time
  $finish;
end
always
  #5 clk = ~clk; //changing clock after every 5s
endmodule
```

2.3.2 Waveform plot



2.4 Further Analysis

Since the cycle of the counter consists of only 15 states, and the state which isn't getting covered in the cycle is 0000. If the counter starts from state 0000, then this state would keep on repeating for every clock impulse and the counter would get stuck in an lockout condition. Whereas if it starts from any other state, then the counter would follow the sequence given in the question, covering each one of the 15 states once before getting back to initial state.

3 Code Files Link

The respective code files to run the simulation for both the Synchronous Gray Code Counter and Synchronous Ring Counter can be downloaded from **here**.