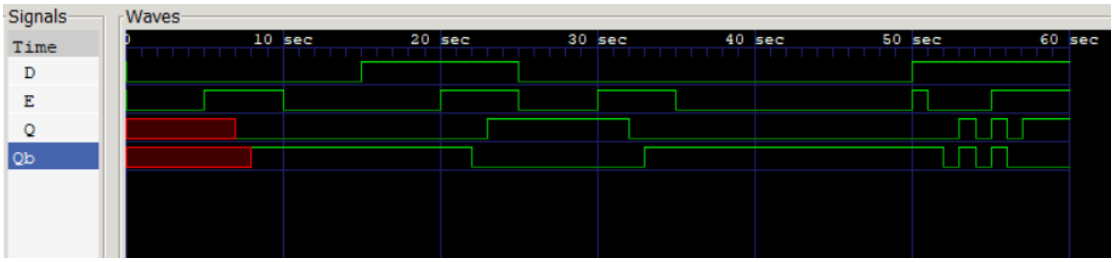
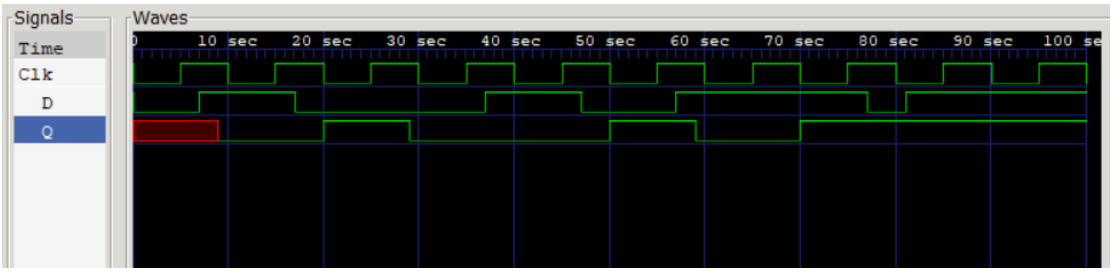
(1)



The output oscillates between 0 and 1 because the update of D-latches sometime require several rounds to update to the new state.
Though there are some delay because of the gate delays, we can still tell from the wave that Q is set to D when E is 1, so the simulation is correct.

(2)



The Q in flip-flop started to change when the clock change from 0 to 1 (the negedge), which set the Enable of the second Latch to be 1. This can not be tell directly in the graph because there are gate delays in the two not gates and the D-Latch.
The simulation is correct because it acts like what a D-flipflop should be like.

(3)

In the state diagram based model, I use three part to deal with three individual tasks, which is updating state, setting next state, and handling output.

```
module Lab3_Converter_state_diagram (input X, Clk, Rst, output Z);

    reg [2:0] state, nextstate;
    reg Z;
    parameter S0 = 3'b000, S1 = 3'b001, S2 = 3'b010, S3 = 3'b011, S4 = 3'b100, S5 = 3'b101, S6 = 3'b110;

    always @(posedge Clk, negedge Rst)  // handle the update
        if(Rst == 0) state <= S0;
        else state <= nextstate;

    always @(state, X)              // handle the change of states
        case(state)
            S0: if(~X) nextstate = S4; else nextstate = S1;
            S1: if(~X) nextstate = S5; else nextstate = S2;
            S2: nextstate = S3;
            S3: nextstate = S0;
            S4: nextstate = S5;
            S5: if(~X) nextstate = S6;
                else nextstate = S3;
            S6: nextstate = S0;
        endcase

    always @ (state, X)             // handle the output
        case (state)
            S0, S1, S5, S6: Z = ~X;
            S2, S3, S4: Z = X;
        endcase

endmodule
```

For the structure modeling

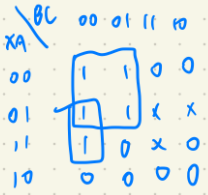
Present State	Next State X = 0	Next State X = 1	Output X = 0	Output X = 1
S_0	S_4	S_1	1	0
S_1	S_5	S_2	1	0
S_2	S_3	S_3	0	1
S_3	S_0	S_0	0	1
S_4	S_5	S_5	0	1
S_5	S_6	S_3	1	0
S_6	-	S_0	x	0

表 1：Excess-3 轉 BCD 轉換器之狀態表

Table 1: The state table of the Excess-3-to-BCD converter.

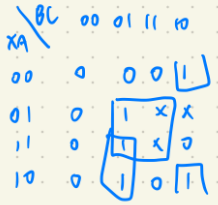
(ABC stands for each column of states, and $A^tB^tC^t$ is the new state)

for A^t



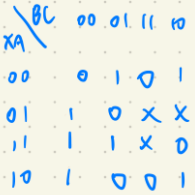
$A^t = X'B' + AB'C'$

for B^t



$B^t = AC + XB'C + A'BC'$

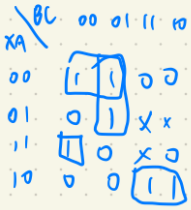
for C^t



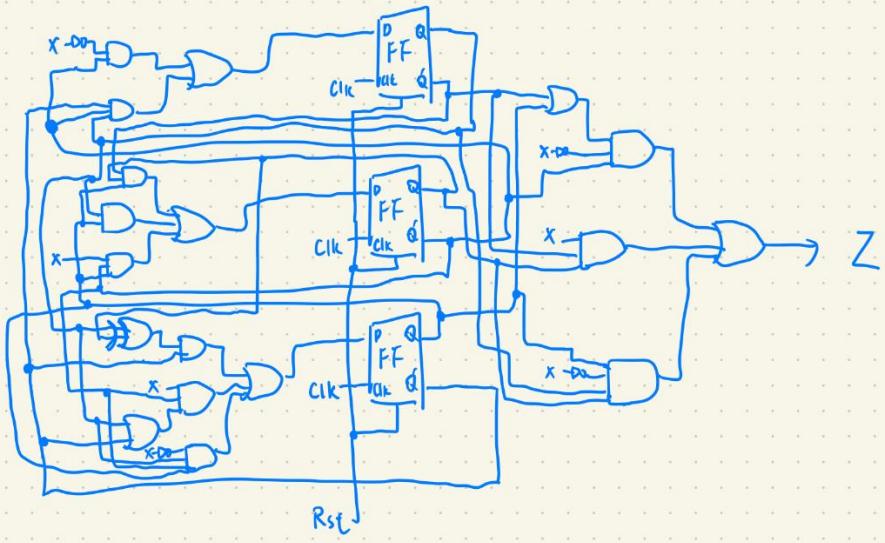
$C^t = A'BC' + XBC' + XAB' + AB'C' + X'A'B'C$

$= C'(A \oplus B) + XB'(A+C') + X'A'BC$

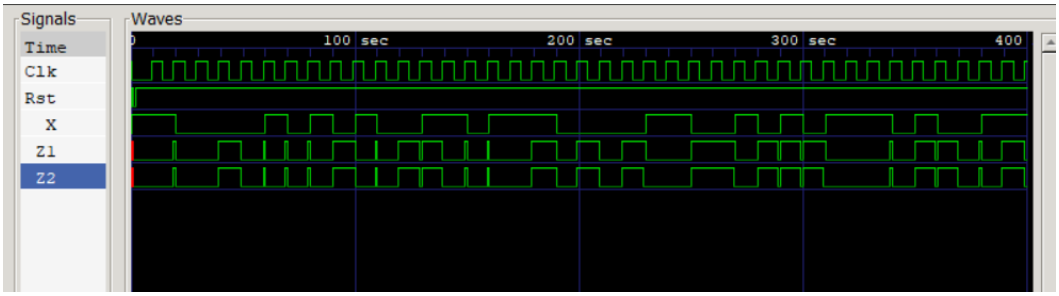
for Z



$Z = X'B'C + XA'B' + XA'B + XABC'$



Simulation result:



Testbench design:

First set the Rst to reset the Flipflop. Set the input X from 3 to 13, having all the numbers (in binary) in the reverse order, i.e. (1100) (0010) (1010) ….. which can be done by the following code.

```
initial begin
    for(i=3;i<13;i++)  begin
        for(j = 0;j<4;j++) begin
            X = i[j];
            #(10);
        end

    end
    $finish;
end
```

The result of each 4 pair should be 0 to 9 in reverse order, i.e. (0000) (1000) (0100) (1100)……, which is same as the simulation result, so the simulation is correct.

(4)

What I have learned & problems I encountered:

This is my first time writing the sequential circuit, and I find it to be a little difficult. It took me quite a long time to get familiar with of the usage of syntax and function, and I finally have idea of how this circuits should be done.

The last question is the most difficult one to me. I was being careless and kept making typing mistakes, which makes the module wrong. I tried to traceback all the things through the simulation result by comparing the state trasitions, and it took me quite a while to figure out where I got wrong. After that, I found my testbench will cause some error because I did not set the Rst well, so I made several trial on the testbench and finally make the simulation result easier to read.