Hacettepe University

Computer Engineering Department

BM233 Logic Design Lab - 2022 Fall

Experiment 5 - Sequential Circuits in Verilog

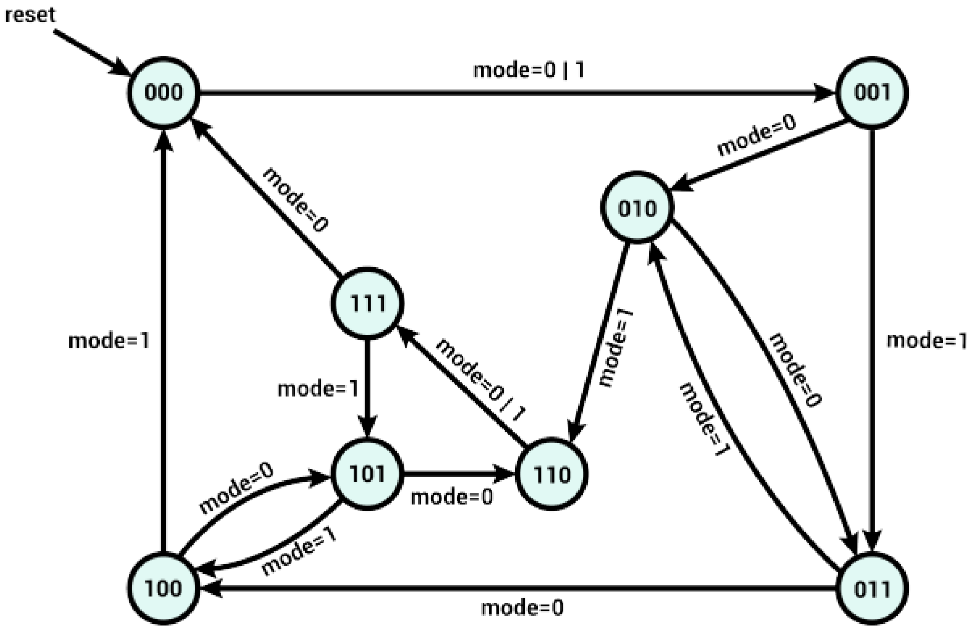
December 31, 2022

Student name:
Musa Enes ERKUVAN

Student Number:
b2200356848

1 Problem Definition

For this project, we need to implement a binary and gray counter, which counts up. We also need to implement a design that changes its counting mode with the given input.



THE STATE TRANSITION TABLE – D FLIP FLOP MODE 0

Q2 present	Q1 Present	Q0 Present	Q2 Next	Q1 Next	Q0 Next	D2 Input	D1 Input	D0 input
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0

1 D0 FLIP FLOP

Q2,Q1 Present / Q0 present	00	01	11	10
0	1	1	1	1
1	0	0	0	0

D0 = ((~Q2 present) & (~Q1 present) & (~Q0present)) | ((~Q2 present) & (Q1 present) & (~Q0present)) | ((Q2 present) & (~Q1 present) & (~Q0present)) | ((Q2 present) & (Q1 present) & (~Q0present))

D0 = ~Q0present

2 D1 FLIP FLOP

Q2,Q1 Present / Q0 present	00	01	11	10
0	0	1	1	0
1	1	0	0	1

D1 = ((~Q2 present) & (~Q1 present) & (Q0present)) | ((~Q2 present) & (Q1 present) & (~Q0present)) | ((Q2 present) & (~Q1 present) & (Q0present)) | ((Q2 present) & (Q1 present) & (~Q0present))

D1 = ((~Q1 present) & (Q0present)) | ((Q1 present) & (~Q0present))

3 D2 FLIP FLOP

Q2,Q1 Present / Q0 present	00	01	11	10
0	0	0	1	1
1	0	1	0	1

D2 = ((~Q2 present) & (Q1 present) & (Q0present)) | ((Q2 present) & (~Q1 present) & (~Q0present)) | ((Q2 present) & (~Q1 present) & (Q0present)) | ((Q2 present) & (Q1 present) & (~Q0present))

D2 = ((~Q2 present) & (Q1 present) & (Q0present)) | ((Q2 present) & (~Q1 present)) | ((Q2 present) & (Q1 present) & (~Q0present))

THE STATE TRANSITION TABLE – D FLIP FLOP MODE 1

Q2 present	Q1 Present	Q0 Present	Q2 Next	Q1 Next	Q0 Next	D2 Input	D1 Input	D0 input
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	1
0	1	1	0	1	0	0	1	0
0	1	0	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	1	0	1	1	0	1
1	0	1	1	0	0	1	0	0
1	0	0	0	0	0	0	0	0

1 D0 FLIP FLOP

Q2,Q1 Present / Q0 present	00	01	11	10
0	1	0	1	0
1	1	0	1	0

D0 = ((~Q2 present) & (~Q1 present) & (~Q0present)) | ((~Q2 present) & (~Q1 present) & (Q0present)) | ((Q2 present) & (Q1 present) & (~Q0present)) | ((Q2 present) & (Q1 present) & (Q0present))

D0 = ((~Q2 present) & (~Q1 present)) | ((Q2 present) & (Q1 present))

2 D1 FLIP FLOP

Q2,Q1 Present / Q0 present	00	01	11	10
0	0	1	1	0
1	1	1	0	0

D1 = ((~Q2 present) & (~Q1 present) & (Q0present)) | ((~Q2 present) & (Q1 present) & (Q0present)) | ((~Q2 present) & (Q1 present) & (~Q0present)) | ((Q2 present) & (Q1 present) & (~Q0present))

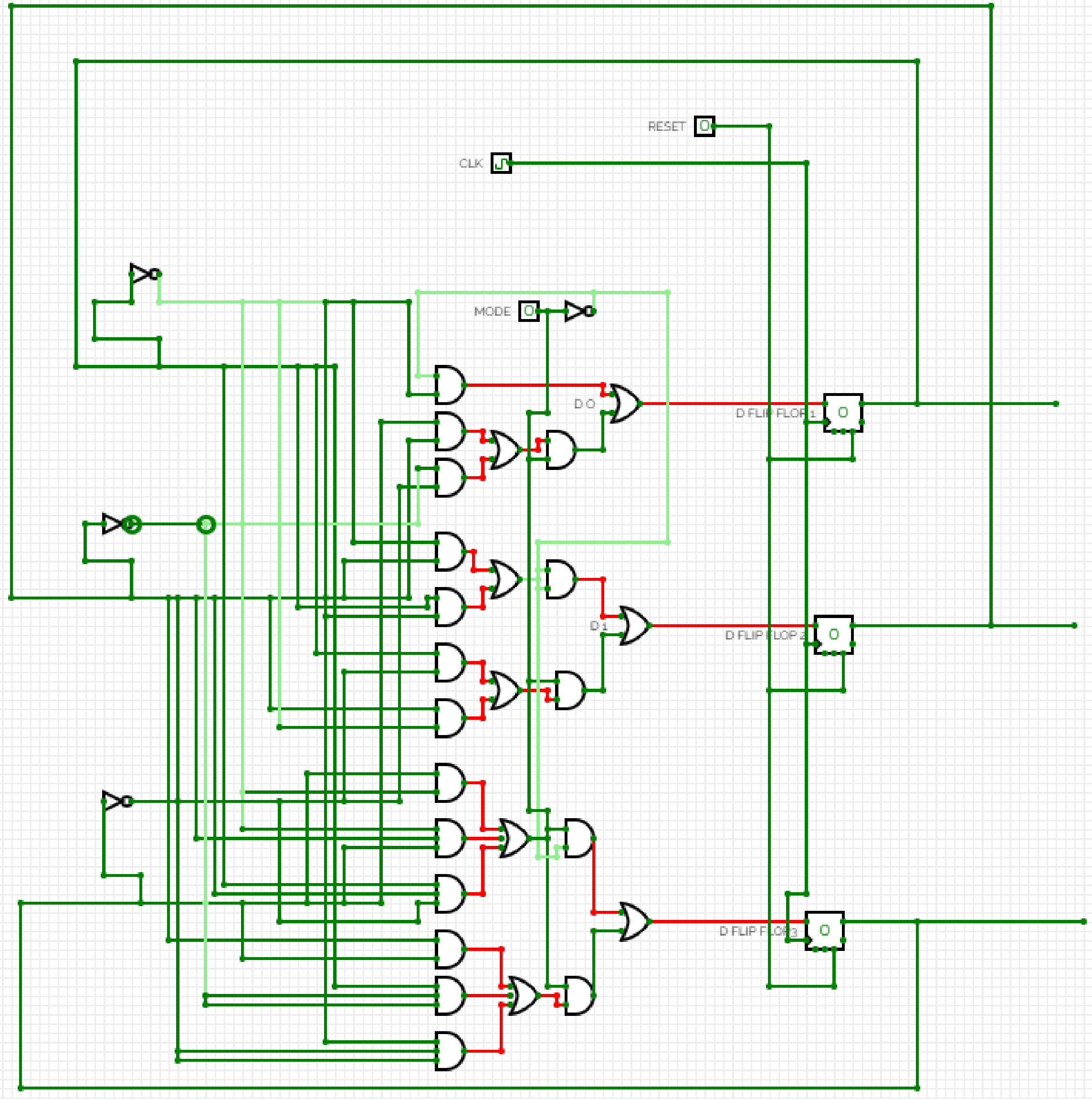
D1 = ((~Q2 present) & (Q0 present)) | ((Q1 present) & (~Q0present)))

3 D2 FLIP FLOP

Q2,Q1 Present / Q0 present	00	01	11	10
0	0	1	1	0
1	0	0	1	1

D2 = ((~Q2 present) & (Q1 present) & (~Q0present)) | ((Q2 present) & (Q1 present) & (~Q0present)) | ((Q2 present) & (Q1 present) & (Q0present)) | ((Q2 present) & (~Q1 present) & (Q0present))

D2 = ((Q1 present) & (~Q0 present)) | ((Q2 present) & (Q0present))


RESET
0
CLK
MODE
0
D 0
D FLIP FLOP 1
0
D 1
D FLIP FLOP 2
0
D FLIP FLOP 3
0
0

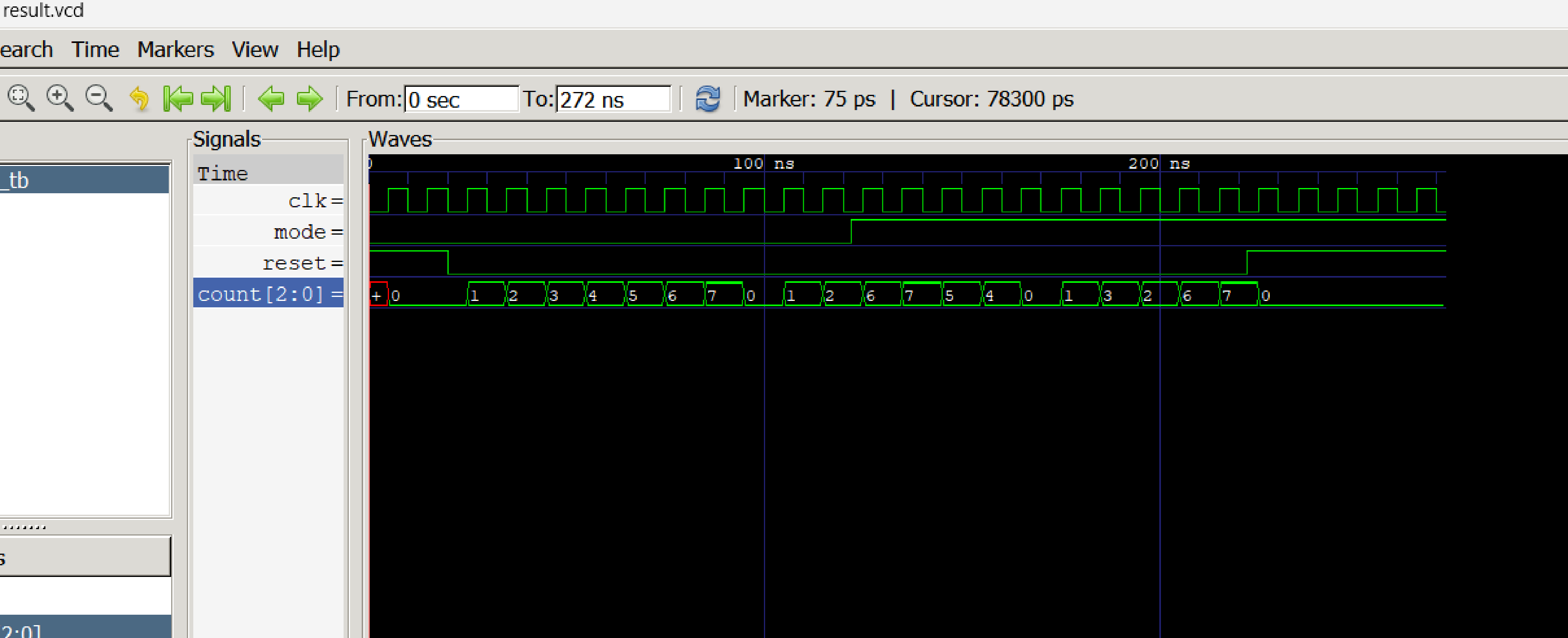
```
module counter_d(input reset, input clk, input mode, output [2:0] count);

    // Your code goes here.  DO NOT change anything that is already given! Otherwise,

    // flip flop inputs

    wire D0,D1,D2;

// 0- Inputs hex converter

    wire notc0,notc1,notc2,notmode;

    not gate90(notc0,count[0]);
    not gate91(notc1,count[1]);
    not gate92(notc2,count[2]);
    not gate93(notmode,mode);

// 1-first flip flop

    // 1a-mode 0 inputs , binary


    wire x;

    and gate1(x,notc0, notmode); // D0 = ~Q0 ,mode 0

    // 1b-mode 1 inputs , gray code

    wire y,y1,y2,y3;

    and gate2(y,count[2],count[1]);
    and gate3(y1,notc2,notc1);

    or gate4(y2,y,y1);

    and gate5(y3,y2,mode); // D0 = (Q2 & Q1) | (~Q2 & ~Q1) , mode 1

    // 1c-decide first flip flop , mode 0 or 1

    or gate6(D0,x,y3);

// 2-second flip flop

    // 2a-mode 0 inputs , binary

    wire z,z1,z2,z3;

    and gate7(z,notc0,count[1]);
    and gate8(z1,count[0],notc1);

    or gate9(z2,z,z1);

    and gate10(z3,z2,notmode); // D1 = (Q1 & Q0) | (~Q1 & Q0) , mode 0

    // 2b-mode 1 inputs , gray code

    wire v,v1,v2,v3;

    and gate11(v,notc2,count[0]);
    and gate12(v1,count[1],notc0);

    or gate13(v2,v,v1);

    and gate14(v3,v2,mode); // D1 = (~Q2 & Q0) | (Q1 & ~Q0) , mode 1

    // 2c-decide second flip flop , mode 0 or 1

    or gate15(D1,z3,v3);

// 3-third flip flop


    // 3a-mode 0 inputs , binary

    wire n,n1,n2,n3,n4;

    and gate16(n,count[2],notc1);
    and gate17(n1,count[2],count[1],notc0);
    and gate18(n2,notc2,count[1],count[0]);

    or gate19(n3,n,n1,n2);

    and gate20(n4,n3,notmode); // D2 = (Q2 & ~Q1) | (Q2 & Q1 & ~Q0) | (~Q2 & Q1 & Q0)

    // 3b-mode 1 inputs , gray code

    wire m,m1,m2,m3,m4;

    and gate21(m,count[2],count[1]);
    and gate22(m1,count[2],notc1,count[0]);
    and gate23(m2,notc2,count[1],notc0);

    or gate24(m3,m,m1,m2);

    and gate25(m4,m3,mode); // D2 = (Q2 & Q1) | (Q2 & ~Q1 & Q0) | (~Q2 & Q1 & ~Q0) ,

    // 3c-decide third flip flop , mode 0 or 1

    or gate26(D2,n4,m4);

// 4-call or send inputs to flip flops

    dff_sync_res flip1 ( .D (D0) , .clk(clk) , .sync_reset(reset) , .Q(count[0]) );

    dff_sync_res flip2 ( .D (D1) , .clk(clk) , .sync_reset(reset) , .Q(count[1]) );

    dff_sync_res flip3 ( .D (D2) , .clk(clk) , .sync_reset(reset) , .Q(count[2]) );


endmodule
```

```verilog
module dff_sync_res(D, clk, sync_reset, Q);
    input D;
    input clk;
    input sync_reset;
    output reg Q;

    // Your code goes here.  DO NOT change anything that is already given! Otherwise

    always @(posedge clk ) // this means positive edge , Rising edge of clock , init
    begin

        if(sync_reset==1'b1)
            Q <= 1'b0;

        else
            Q <= D;

    end

endmodule
```

1 Testbench Implementation

State for which cases you are testing and their meaning. Example how to add Verilog code:

```
`timescale 1ns/1ps

module counter_tb;
    reg reset, clk, mode;
    wire [2:0] count;
    integer i;

        //Comment the next line out when testing your JK flip flop implementation.
    counter_d uut(reset, clk, mode, count);
    // Uncomment the next line to test your JK flip flop implementation.
    //counter_jk c1(reset, clk, mode, count);

    initial begin
        // Your code goes here.  DO NOT change anything that is already given! Othe
        // Make sure to use $finish statement to avoid infinite loops.
        $dumpfile("result.vcd");
                $dumpvars;


        reset=1'b1;
        mode=1'b0;

        #20 reset =1'b0;

        #102 mode=1'b1;

        #100 reset =1'b1;

        #50 $finish;

    end

    initial begin

        // Generate clock
        // Your code goes here.  DO NOT change anything that is already given! Othe
        clk=1'b0;
        forever #5 clk = ~clk;

    end

endmodule
```


result.vcd
earch Time Markers View Help
From: 0 sec To: 272 ns
Marker: 75 ps | Cursor: 78300 ps
Signals
Waves
_tb
Time
clk=
mode=
reset=
count[2:0]=
100 ns
200 ns
+ 0 1 2 3 4 5 6 7 0 1 2 6 7 5 4 0 1 3 2 6 7 0

THE STATE TRANSITION TABLE – JK FLIP FLOP MODE 0

Q2 present	Q1 Present	Q0 Present	Q2 Next	Q1 Next	Q0 Next	J2 Input	K2 Input	J1 input	K1 input	J0 input	K0 input
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	0	0	X	1	X	1	X	1

1 J2 FLIP FLOP

Q2,Q1 Present / Q0 present	00	01	11	10
0	0	0	X	X
1	0	1	X	X

J2 = ((~Q2 present) & (Q1 present) & (Q0 present))

2 K2 FLIP FLOP

Q2,Q1 Present / Q0 present	00	01	11	10
0	X	X	0	0
1	X	X	1	0

K2 = ((Q2 present) & (Q1 present) & (Q0 present))

3 J1 FLIP FLOP

Q2,Q1 Present / Q0 present	00	01	11	10
0	0	X	X	0
1	1	X	X	1

J1 = ((~Q1 present) & (Q0 present))

4 K1 FLIP FLOP

Q2,Q1 Present / Q0 present	00	01	11	10
0	X	0	0	X
1	X	1	1	X

K1 = ((Q1 present) & (Q0 present))

5 J0 FLIP FLOP

Q2,Q1 Present / Q0 present	00	01	11	10
0	1	1	1	1
1	X	X	X	X

J0 = (~Q0 present)

6 K0 FLIP FLOP

Q2,Q1 Present / Q0 present	00	01	11	10
0	X	X	X	X
1	1	1	1	1

K0 = (Q0 present)

THE STATE TRANSITION TABLE – JK FLIP FLOP MODE 1

Q2 present	Q1 Present	Q0 Present	Q2 Next	Q1 Next	Q0 Next	J2 Input	K2 Input	J1 input	K1 input	J0 input	K0 input
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	1	0	X	1	X	X	0
0	1	1	0	1	0	0	X	X	0	X	1
0	1	0	1	1	0	1	X	X	0	0	X
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	1	0	1	X	0	X	1	X	0
1	0	1	1	0	0	X	0	0	X	X	1
1	0	0	0	0	0	X	1	0	X	0	X

1 J2 FLIP FLOP

Q2,Q1 Present / Q0 present	00	01	11	10
0	0	1	X	X
1	0	0	X	X

J2 = ((~Q2 present) & (Q1 present) & (~Q0present))

2 K2 FLIP FLOP

Q2,Q1 Present / Q0 present	00	01	11	10
0	X	X	0	1
1	X	X	0	0

K2 = ((Q2 present) & (~Q1 present) & (~Q0present))

3 J1 FLIP FLOP

Q2,Q1 Present / Q0 present	00	01	11	10
0	0	X	X	0
1	1	X	X	0

J1 = ((~Q2 present) & (~Q1 present) & (Q0present))

4 K1 FLIP FLOP

Q2,Q1 Present / Q0 present	00	01	11	10
0	X	0	0	X
1	X	0	1	X

K1 = ((Q2 present) & (Q1 present) & (Q0present))

5 J0 FLIP FLOP

Q2,Q1 Present / Q0 present	00	01	11	10
0	1	0	1	0
1	X	X	X	X

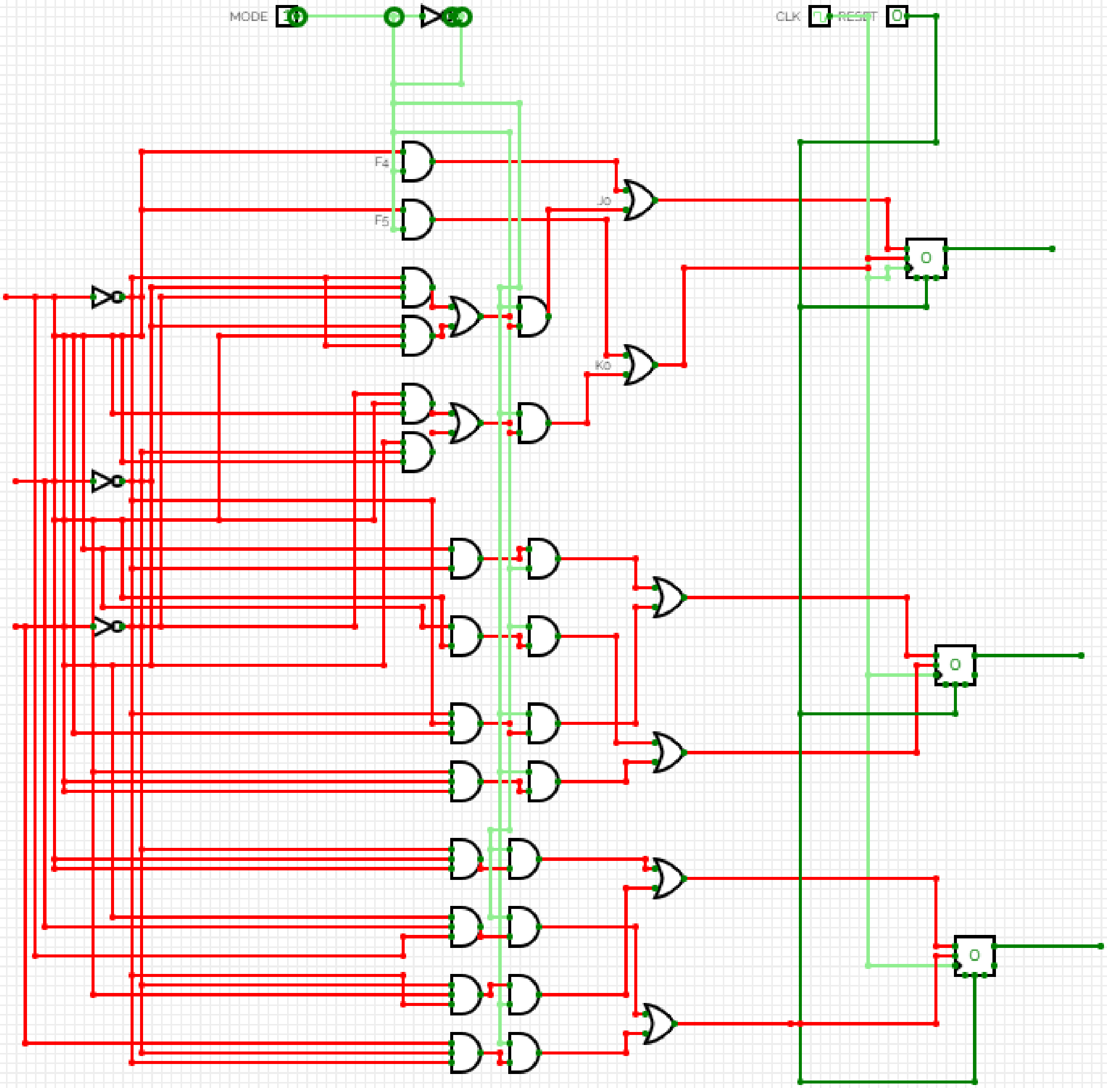
J0 = ((~Q2 present) & (~Q1 present) & (~Q0present)) | ((Q2 present) & (Q1 present) & (Q0present))

6 K0 FLIP FLOP

Q2,Q1 Present / Q0 present	00	01	11	10
0	X	X	X	X
1	0	1	0	1

K0 = ((~Q2 present) & (Q1 present) & (Q0present)) | ((Q2 present) & (~Q1 present) & (Q0present))

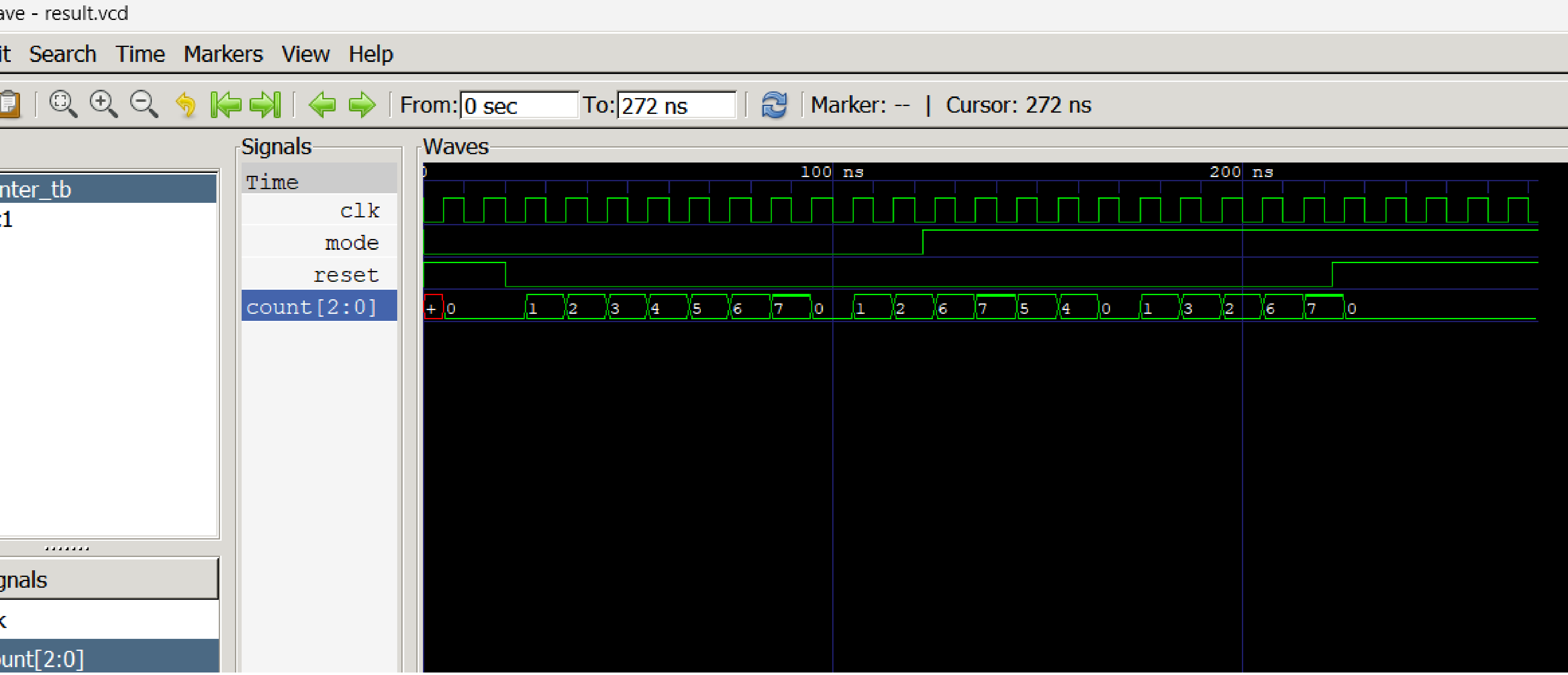
SYNC RESET	Q Present	Q Next	J	K
0	X	0	X	X
1	0	0	0	X
1	0	1	1	X
1	1	0	X	1
1	1	1	X	0


MODE
1
CLK
RESET
0
F4
F5
J0
K0
0
0
0

```verilog
module counter_jk(input reset, input clk, input mode, output [2:0] count);

    // Your code goes here.  DO NOT change anything that is already given! Otherwis


    wire J0,K0,J1,K1,J2,K2;

// 0- Inputs hex converter
    wire notc0,notc1,notc2,notmode;

    not gate90(notc0,count[0]);
    not gate91(notc1,count[1]);
    not gate92(notc2,count[2]);
    not gate93(notmode,mode);

// 1-first flip flop
// mode 0

    // J0
    wire f4;
    and gate88(f4,notc0,notmode);

    // K0
    wire f5;
    and gate89(f5,count[0],notmode);

// mode 1

    // J0
    wire l1,l2,l3;

    and gate7(l1,notc2,notc1,notc0);
    and gate8(l2,count[2],count[1],notc0);

    or gate9(l3,l1,l2);

    wire f2;
    and gate78(f2,l3,mode);

    // K0
    wire l4,l5,l6;

    and gate10(l4,notc2,count[1],count[0]);
    and gate11(l5,count[2],notc1,count[0]);


    or gate12(l6,l4,l5);

    wire f1;
    and gate77(f1,l6,mode);

// decide which mode to choose

    or gate660(J0,f2,f4);
    or gate679(K0,f1,f5);

// 2-second flip flop

// mode 0

    // J1
    wire x1,o1;
    and gate1(x1,notc1,count[0]);

    and gate99(o1,x1,notmode); // J1 with mode 0

    // K1
    wire x2,o2;
    and gate2(x2,count[1],count[0]);

    and gate100(o2,x2,notmode); // K1 with mode 0

// mode 1

    // J1
    wire l7,o3;
    and gate13(l7,notc2,notc1,count[0]);

    and gate101(o3,l7,mode);


    // K1
    wire l8,o4;
    and gate14(l8,count[2],count[1],count[0]);

    and gate102(o4,l8,mode);

// decide which mode to choose

    or gate666(J1,o3,o1);
    or gate676(K1,o4,o2);


// 3-third flip flop

// mode 0

    // J2
    wire y1,t1;
    and gate5(y1,notc2,count[1],count[0]);

    and gate71(t1,y1,notmode);

    // K2
    wire y2,t2;
    and gate6(y2,count[2],count[1],count[0]);

    and gate23(t2,y2,notmode);

// mode 1

    // J2
    wire l9,w1;
    and gate15(l9,notc2,count[1],notc0);

    and gate44(w1,l9,mode);

    // K2
    wire l10,w2;
    and gate16(l10,count[2],notc1,notc0);

    and gate33(w2,l10,mode);

// decide which mode to choose

    or gate66(J2,t1,w1);
    or gate67(K2,t2,w2);

    // CALL FLIP FLOPS

    jk_sync_res flip1 ( .J(J0) , .K(K0)  ,  .clk(clk)  ,  .sync_reset(reset)
 ,  .Q(count[0])  );

    jk_sync_res flip2 ( .J(J1) , .K(K1)  ,  .clk(clk)  ,  .sync_reset(reset)
 ,  .Q(count[1])  );

    jk_sync_res flip3 ( .J(J2) , .K(K2)  ,  .clk(clk)  ,  .sync_reset(reset)
 ,  .Q(count[2])  );

endmodule
```

```verilog
module jk_sync_res(J, K, clk, sync_reset, Q);
    input J;
    input K;
    input clk;
    input sync_reset;
    output reg Q;

    // Your code goes here.  DO NOT change anything that is already given! Otherwise

    always@ (posedge(clk))
    begin
        if(sync_reset == 1'b1)
            Q <= 1'b0;
        else
            if(J == 0 && K == 0)
                Q <= Q;
            else if(J == 0 && K == 1)
                Q <= 1'b0;
            else if(J == 1 && K == 0)
                Q <= 1'b1;
            else
                Q <= ~Q;

    end

endmodule
```

```
`timescale 1ns/1ps

module counter_tb;
    reg reset, clk, mode;
    wire [2:0] count;
    integer i;

        //Comment the next line out when testing your JK flip flop implementation.
    //counter_d uut(reset, clk, mode, count);
    // Uncomment the next line to test your JK flip flop implementation.
    counter_jk c1(reset, clk, mode, count);

    initial begin
        // Your code goes here.  DO NOT change anything that is already given! Othe
        // Make sure to use $finish statement to avoid infinite loops.
        $dumpfile("result.vcd");
                $dumpvars;


        reset=1'b1;
        mode=1'b0;

        #20 reset =1'b0;

        #102 mode=1'b1;

        #100 reset =1'b1;

        #50 $finish;

    end

    initial begin

        // Generate clock
        // Your code goes here.  DO NOT change anything that is already given! Othe
        clk=1'b0;
        forever #5 clk = ~clk;

    end

endmodule
```


ave - result.vcd
it Search Time Markers View Help
From: 0 sec To: 272 ns Marker: -- | Cursor: 272 ns
Signals
Waves
nter_tb
:1
gnals
k
unt[2:0]
Time
clk
mode
reset
count[2:0]
100 ns
200 ns
+ 0 1 2 3 4 5 6 7 0 1 2 6 7 5 4 0 1 3 2 6 7 0