Write up- Lab6

Simranpreet Kaur

May 20, 2022

Lab Section C

Objective:

The objective of this lab was to design a state machine for a game where we would be using signals to count the number of turkeys crossing the sensors which are simulated by our buttons. The sensors are high by default but when a turkey crosses either or both sensors, the output becomes low. If the turkey crosses right to left, we increment the count by one. If the turkey crosses left to right, we decrement the count by 1, and if the turkey does not cross both sensors all the way, the counter stays the same.

Methods and Design:

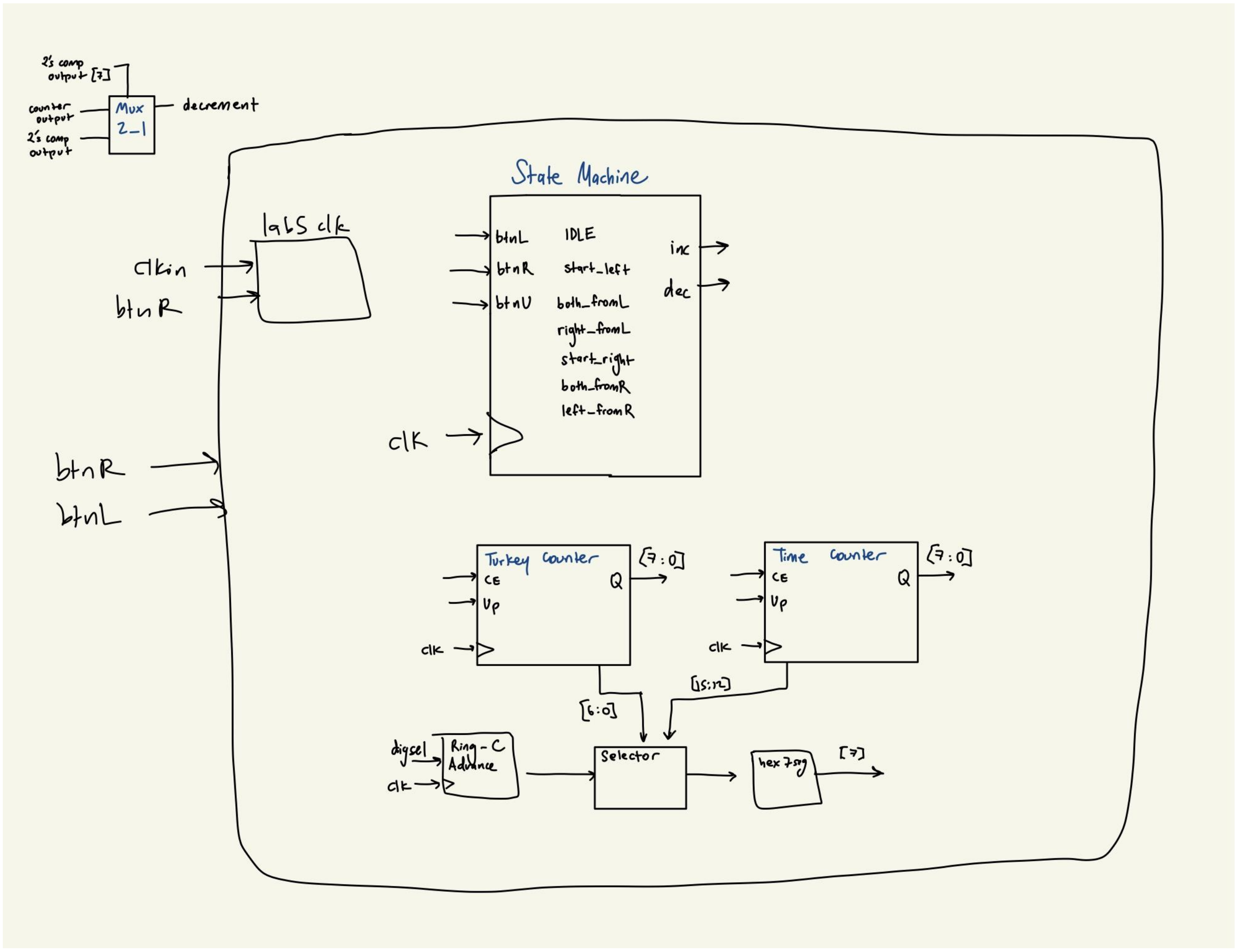
In order to complete this lab, I imported many old modules from previous labs and also built new modules which include:

- State machine: the state machine takes the sensor inputs and sends outputs to control the turkey and time counters. These outputs will be displayed on the 7 segment display
- Turkey Counter: 8-bit counter that counts up and down
- Time Counter: 8-bit counter that keeps track of time using qsec

<u>Top Level:</u>

In my top level, I called all the modules that needed to be connected and made wires for all their outputs. In my top level, I also included the anode logic where I had to make sure that the leftmost anode (an[3]) only displays the time counter, an[2] only displays the negative sign if needed, and an[1] and an[0] display the turkey counter. My LED's 9 and 15 were set to ~btnR and ~btnL respectively. The logic for negatives using muxes, and the logic for implementing two's complement were also written in my top.

Top Level Diagram:



State Machine:

I have a state diagram with 7 states starting from Idle which can either go in the sequence of Right, Both, Left from right to increment a point, or Left, Both, Right from Left to decrement a point. In detail, the way the state machine works is that you start from Idle state where the btnL and btnR buttons are set high for me in order to keep LED15 and LED9 on by default. From there, if the turkey comes from the right, which means only the buttonL is high, it will go into the right state. From the right state it can go back to idle state, or the turkey can go into the both state where none of the two buttons are high because both sensors can sense the turkey being in the middle. Then comes the left state where only the left button senses the turkey which means the turkey successfully crossed the full path and we can go back to idle state and increment our turkey counter by one. Another route the turkey can take is to start from the Left State but the only difference is that if the turkey successfully passes the path from left to right, we decrement the counter by 1.

State Machine Equations:

D0 = (Q4 * (~btnL & ~ btnR)) + (Q3 * (~btnL & ~ btnR)) + (Q6 * (~btnL & ~btnR)) + (Q1 * (~btnL & ~btnR)) + (Q0 * (~btnL & ~btnR))

D1 = (Q0 * (~btnL & btnR)) + (Q1 * (~btnL & btnR)) + (Q2 * (~btnL & btnR))

D2 = (Q3 * (btnL & btnR)) + (Q1 * (btnL & btnR)) + (Q2 * (btnL & btnR))

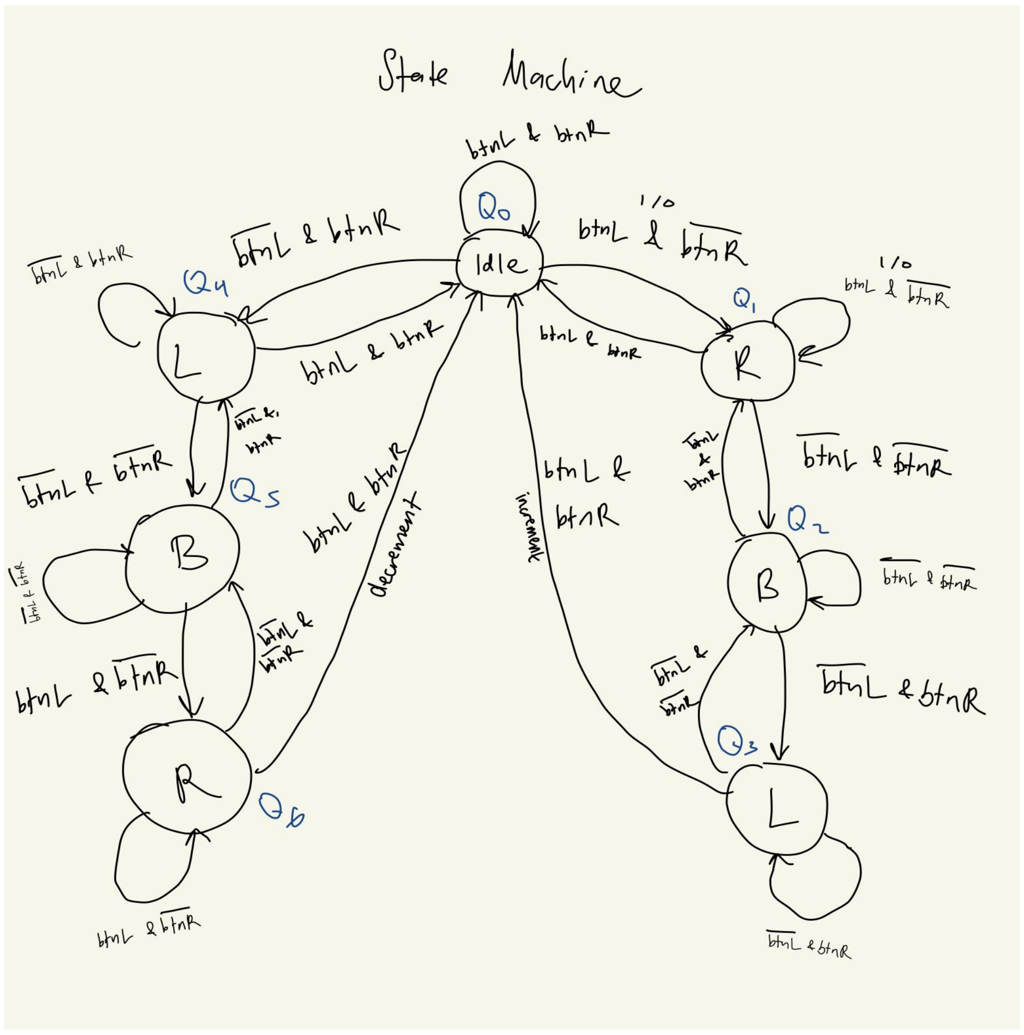
D3 = (Q3 * (btnL & ~btnR)) + (Q2 * (btnL & ~btnR))

D4 = (Q0 * (btnL & ~btnR)) + (Q4 * (btnL & ~btnR)) + (Q5 * (btnL & ~btnR))

D5 = (Q6 * (btnL & btnR)) + (Q4 * (btnL & btnR)) + (Q5 * (btnL & btnR))

D6 = (Q6 * (~btnL & btnR)) + (Q5 * (~btnL & btnR))

State Machine Diagram:



Turkey Counter:

The turkey counter is an 8 bit counter that keeps track of the turkey's movement. It can either go up or down depending on whether the turkey moves from left to right or right to left. In order to decrement into the negatives we had to implement 2's complement logic in our code to make sure that the numbers decrease as binary 0, -1, -2, -3, etc.

Time Counter:

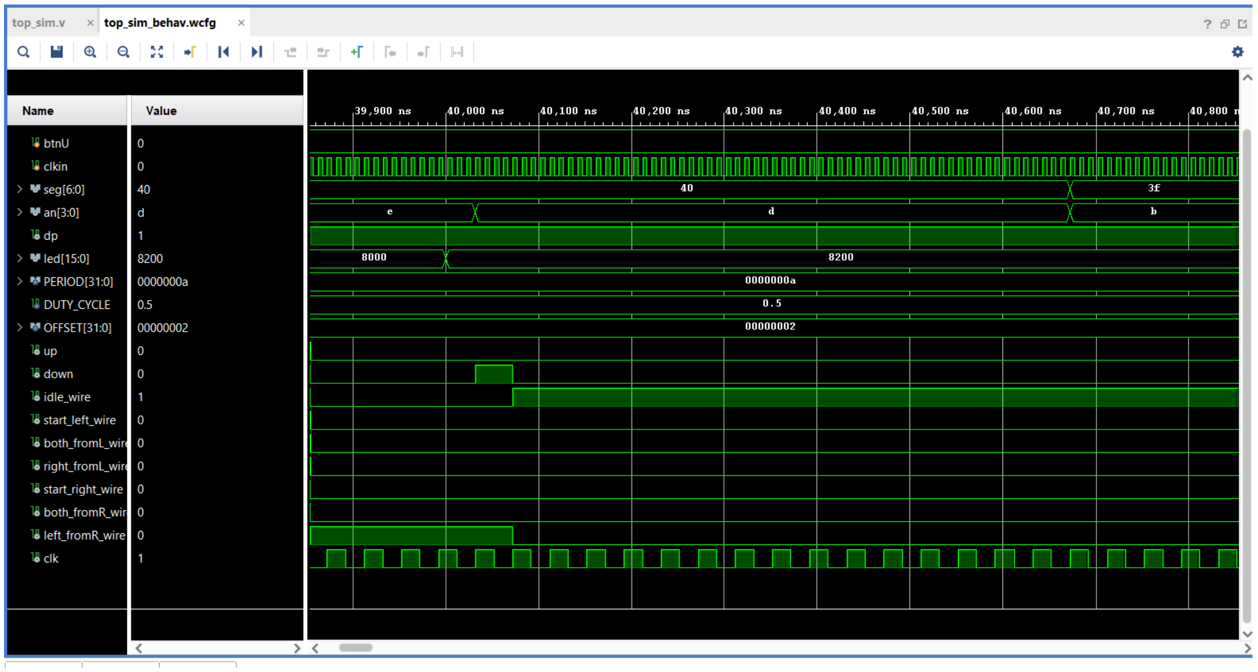
The time counter is also an 8-bit counter that tracks the time from when one of the sensors detects a turkey. The timer only resets once both buttonL and buttonR go back to high, which means that none of the sensors detect a turkey.

Results:

Testing and Simulation:

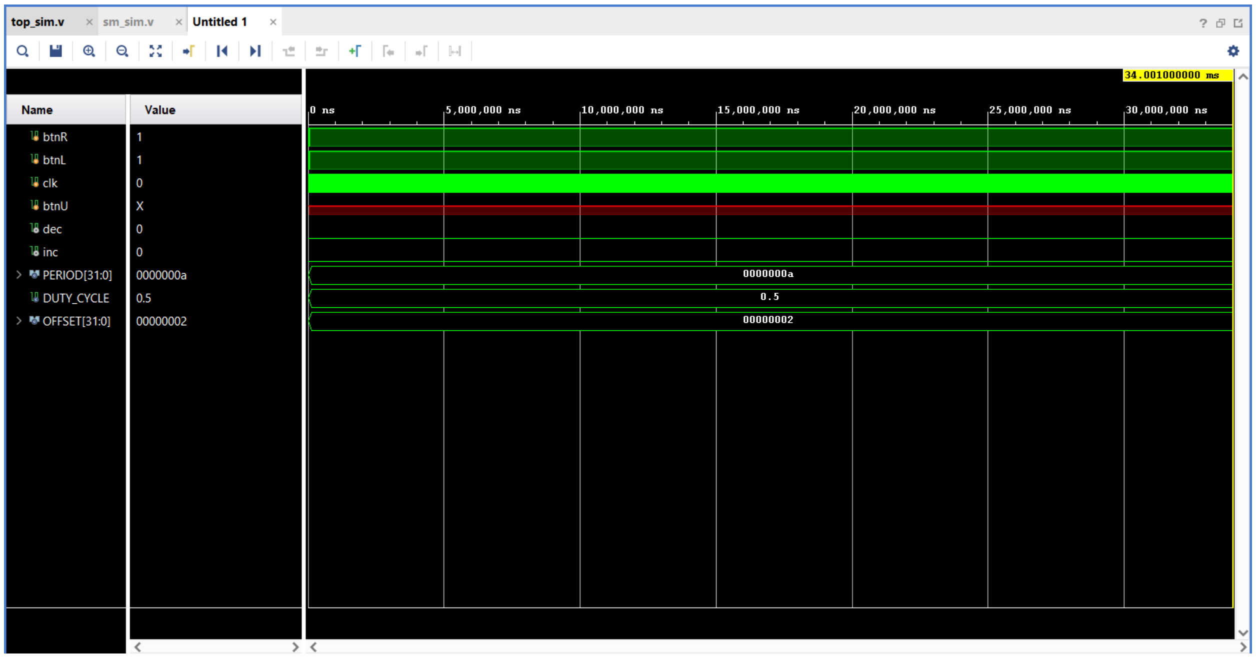
To debug and test my code, I built a simulation for my top module and for my state machine to avoid small errors that may take a while to debug.

Simulation for Top Module:



Name	Value
btnU	0
clkin	0
seg[6:0]	40
an[3:0]	d
dp	1
led[15:0]	8200
PERIOD[31:0]	0000000a
DUTY_CYCLE	0.5
OFFSET[31:0]	00000002
up	0
down	0
idle_wire	1
start_left_wire	0
both_fromL_wir	0
right_fromL_wir	0
start_right_wire	0
both_fromR_wir	0
left_fromR_wire	0
clk	1

Simulation for State Machine:



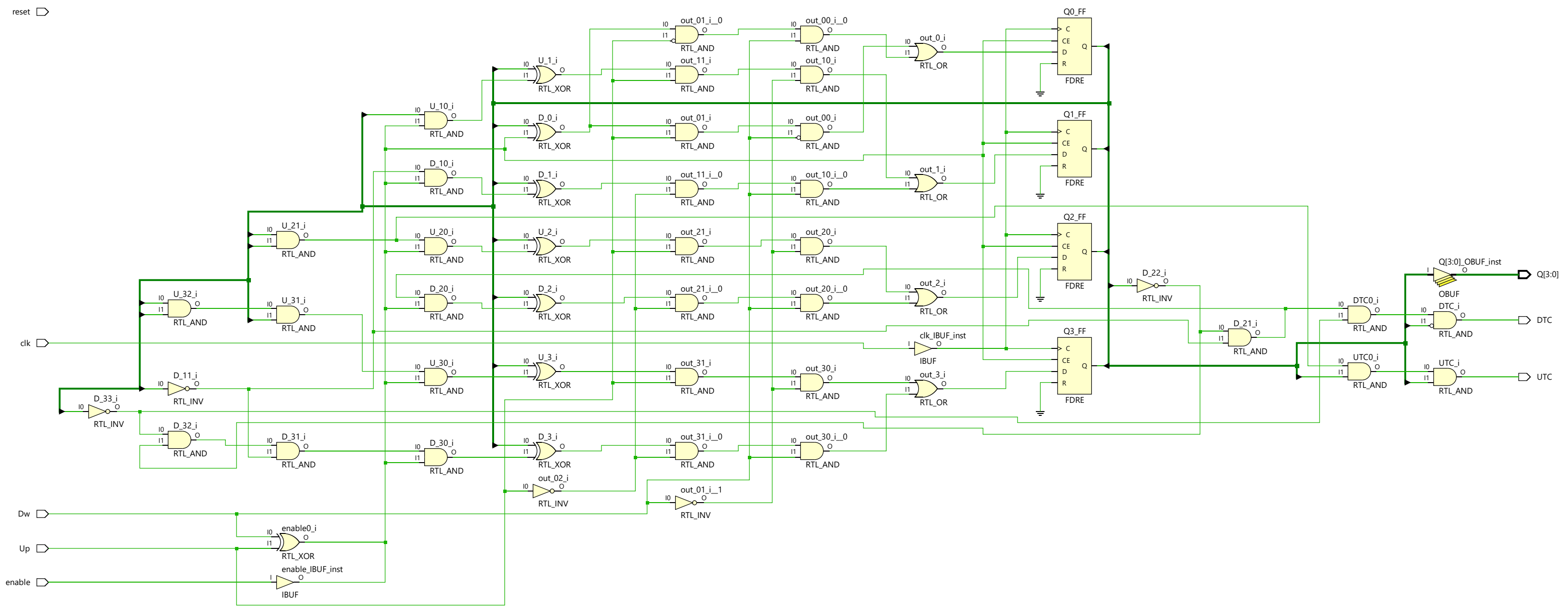
Name	Value
btnR	1
btnL	1
clk	0
btnU	X
dec	0
inc	0
PERIOD[31:0]	0000000a
DUTY_CYCLE	0.5
OFFSET[31:0]	00000002

This is an example of my state machine before I fixed my btnU reset button.

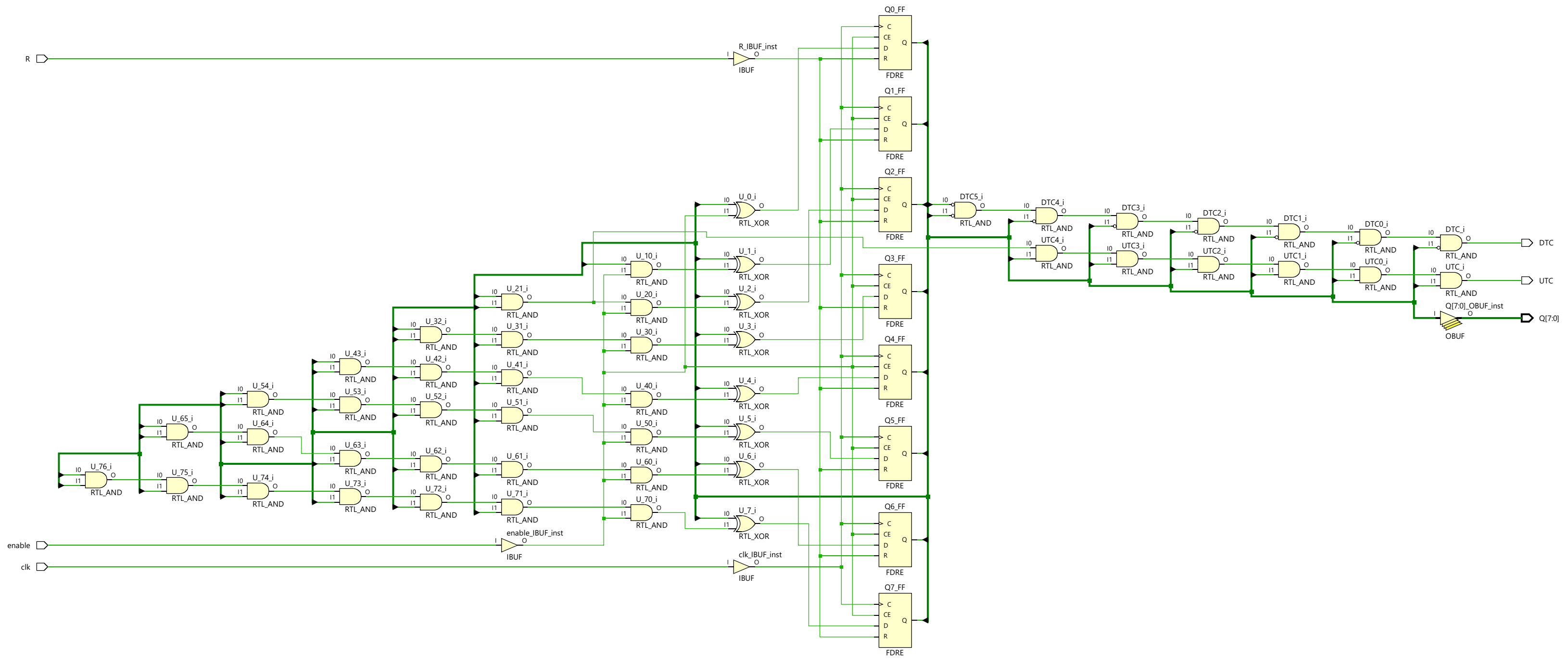
Conclusion:

In conclusion, this game was created to learn how to implement state machines, use two's complement to ensure we are decrementing correctly in binary form, and to learn about sensory inputs with the simulation of buttons. Overall this lab was interesting and fun to create but the only thing that was overwhelming was spending ours debugging small errors I made here and there. At some point I had to rewrite my state machine to ensure I was doing it right. I kept testing my simulations for the state machine when my problem was in my counter module. From this I learned how crucial simulations are since they can save you a lot of time

Appendix:

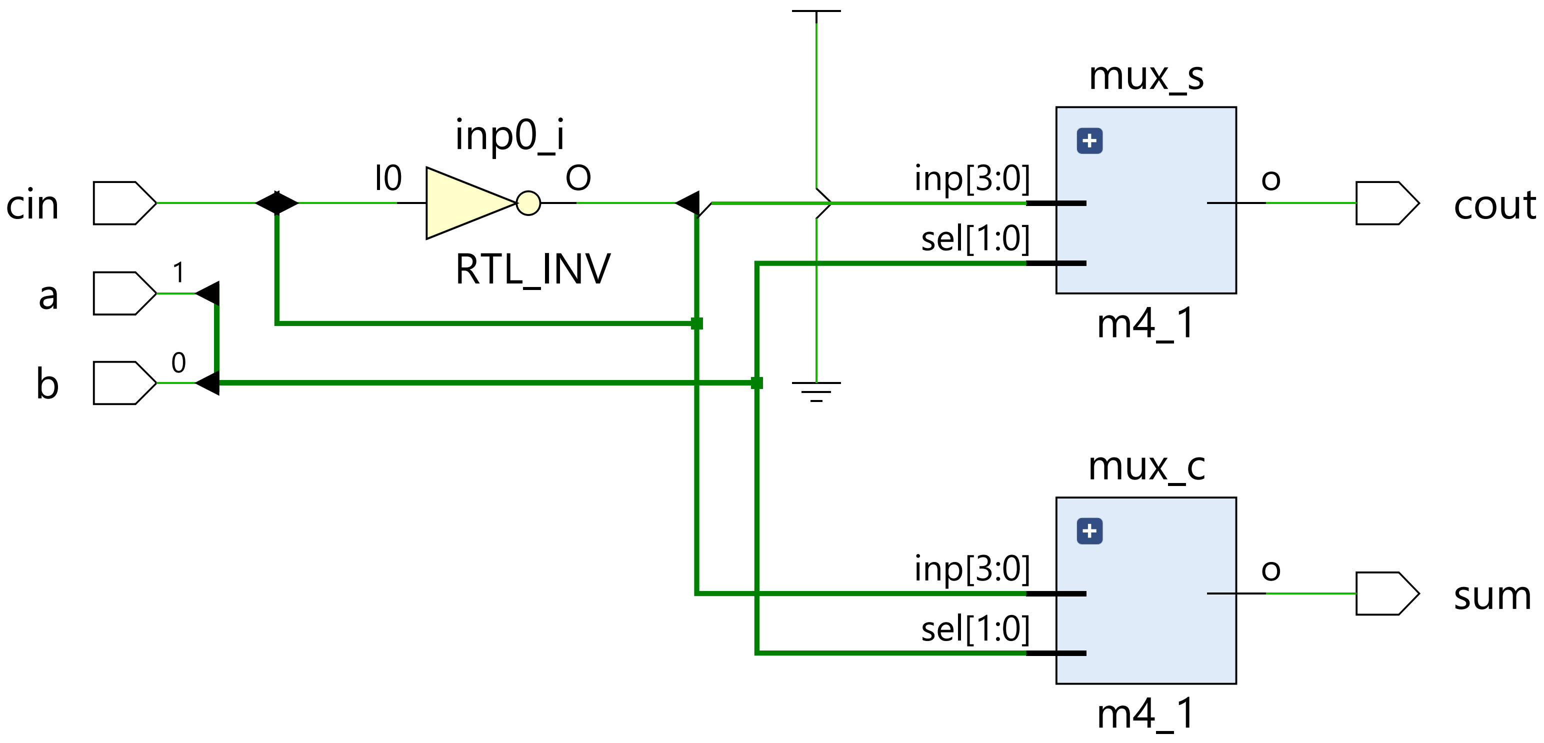

reset
clk
Dw
Up
enable
D_33_i
RTL_INV
U_32_i
RTL_AND
D_11_i
RTL_INV
D_32_i
RTL_AND
U_21_i
RTL_AND
U_31_i
RTL_AND
D_31_i
RTL_AND
enable0_i
RTL_XOR
enable_IBUF_inst
IBUF
U_10_i
RTL_AND
D_10_i
RTL_AND
U_20_i
RTL_AND
D_20_i
RTL_AND
U_30_i
RTL_AND
D_30_i
RTL_AND
U_1_i
RTL_XOR
D_0_i
RTL_XOR
D_1_i
RTL_XOR
U_2_i
RTL_XOR
D_2_i
RTL_XOR
U_3_i
RTL_XOR
D_3_i
RTL_XOR
out_02_i
RTL_INV
out_01_i__0
RTL_AND
out_11_i
RTL_AND
out_01_i
RTL_AND
out_11_i__0
RTL_AND
out_21_i
RTL_AND
out_21_i__0
RTL_AND
out_31_i
RTL_AND
out_31_i__0
RTL_AND
out_01_i__1
RTL_INV
out_00_i__0
RTL_AND
out_10_i
RTL_AND
out_00_i
RTL_AND
out_10_i__0
RTL_AND
out_20_i
RTL_AND
out_20_i__0
RTL_AND
out_30_i
RTL_AND
out_30_i__0
RTL_AND
out_0_i
RTL_OR
out_1_i
RTL_OR
out_2_i
RTL_OR
out_3_i
RTL_OR
clk_IBUF_inst
IBUF
Q0_FF
FDRE
Q1_FF
FDRE
Q2_FF
FDRE
Q3_FF
FDRE
C
CE
D
R
Q
D_22_i
RTL_INV
D_21_i
RTL_AND
DTC0_i
RTL_AND
UTC0_i
RTL_AND
DTC_i
RTL_AND
UTC_i
RTL_AND
Q[3:0]_OBUF_inst
OBUF
Q[3:0]
DTC
UTC
I0
I1
I
O

```
`timescale 1ns / 1ps
///////////////////////////////////////////////////////////////////
///////////////
// Company:
// Engineer:
//
// Create Date: 04/21/2022 01:43:56 PM
// Design Name:
// Module Name: countUD4L
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
///////////////////////////////////////////////////////////////////
///////////////


module countUD4L(
    input clk,
    input Up,
    input Dw,
    input enable,
    input reset,
    output [3:0] Q,
    output UTC,
    output DTC
    );

    wire [3:0] D, U;
    wire [3:0] out;
    assign UTC = Q[0] & Q[1] & Q[2] & Q[3];
    assign DTC = ~Q[0] & ~Q[1] & ~Q[2] & ~Q[3];
    assign enable = (Dw ^ Up);

    FDRE #(.INIT(1'b0) ) Q0_FF (.C(clk), .CE(enable), .Q(Q[0]),
.D(out[0]));
    FDRE #(.INIT(1'b0) ) Q1_FF (.C(clk), .CE(enable), .Q(Q[1]),
.D(out[1]));
    FDRE #(.INIT(1'b0) ) Q2_FF (.C(clk), .CE(enable), .Q(Q[2]),
.D(out[2]));
    FDRE #(.INIT(1'b0) ) Q3_FF (.C(clk), .CE(enable), .Q(Q[3]),
.D(out[3]));

    // increment
    assign U[0] = Q[0] ^ enable;
    assign U[1] = Q[1] ^ (Q[0] & enable);                  // Q[1]
XOR Q[0]
    assign U[2] = Q[2] ^ (Q[1] & Q[0] & enable);           // Q[2]
XOR (Q[1] AND Q[0])
    assign U[3] = Q[3] ^ (Q[2] & Q[1] & Q[0] & enable);    // Q[3]
XOR (Q[1] AND Q[0] AND Q[3])

    //decrement
    assign D[0] = Q[0] ^ enable;
    assign D[1] = Q[1] ^ (~Q[0] & enable);
    // Q[1] XOR Q[0]
    assign D[2] = Q[2] ^ (~Q[1] & ~Q[0] & enable);
    // Q[2] XOR (Q[1] AND Q[0])
    assign D[3] = Q[3] ^ (~Q[2] & ~Q[1] & ~Q[0] & enable);
    // Q[3] XOR (Q[1] AND Q[0] AND Q[3])

    assign out[0] = ((U[0] & Up & ~Dw) | (D[0] & ~Up & Dw));
    assign out[1] = ((U[1] & Up & ~Dw) | (D[1] & ~Up & Dw));
    assign out[2] = ((U[2] & Up & ~Dw) | (D[2] & ~Up & Dw));
    assign out[3] = ((U[3] & Up & ~Dw) | (D[3] & ~Up & Dw));
endmodule
```

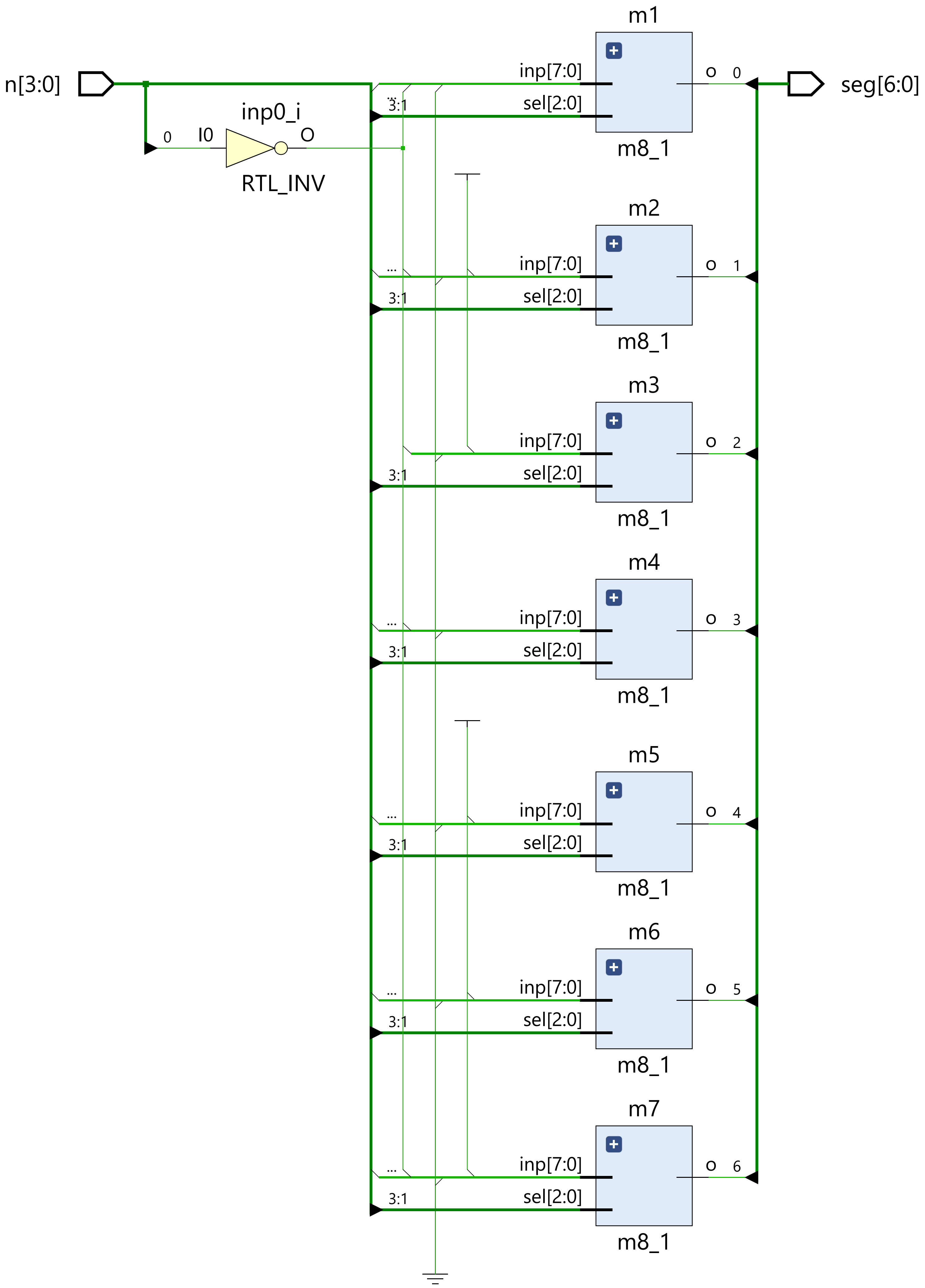

R
enable
clk
R_IBUF_inst
enable_IBUF_inst
clk_IBUF_inst
IBUF
Q0_FF
Q1_FF
Q2_FF
Q3_FF
Q4_FF
Q5_FF
Q6_FF
Q7_FF
FDRE
U_0_i
U_1_i
U_2_i
U_3_i
U_4_i
U_5_i
U_6_i
U_7_i
RTL_XOR
U_76_i
U_65_i
U_75_i
U_54_i
U_64_i
U_74_i
U_43_i
U_53_i
U_63_i
U_73_i
U_32_i
U_42_i
U_52_i
U_62_i
U_72_i
U_21_i
U_31_i
U_41_i
U_51_i
U_61_i
U_71_i
U_10_i
U_20_i
U_30_i
U_40_i
U_50_i
U_60_i
U_70_i
RTL_AND
DTC5_i
DTC4_i
DTC3_i
DTC2_i
DTC1_i
DTC0_i
DTC_i
UTC4_i
UTC3_i
UTC2_i
UTC1_i
UTC0_i
UTC_i
Q[7:0]_OBUF_inst
OBUF
DTC
UTC
Q[7:0]

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
////////////////
// Company:
// Engineer:
//
// Create Date: 04/21/2022 01:43:56 PM
// Design Name:
// Module Name: countUD4L
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////
////////////////


module countUD8L(
    input clk,
    input enable,
    input R,
    output [7:0] Q,
    output UTC,
    output DTC
    );

    wire [7:0] D, U;

    assign UTC = Q[0] & Q[1] & Q[2] & Q[3] & Q[4] & Q[5] & Q[6] &
Q[7];
```

```
    assign DTC = ~Q[0] & ~Q[1] & ~Q[2] & ~Q[3] & ~Q[4] & ~Q[5] &
~Q[6] & ~Q[7];

    FDRE #(.INIT(1'b0) ) Q0_FF (.C(clk), .R(R), .CE(enable),
.Q(Q[0]), .D(U[0]));
    FDRE #(.INIT(1'b0) ) Q1_FF (.C(clk), .R(R), .CE(enable),
.Q(Q[1]), .D(U[1]));
    FDRE #(.INIT(1'b0) ) Q2_FF (.C(clk), .R(R), .CE(enable),
.Q(Q[2]), .D(U[2]));
    FDRE #(.INIT(1'b0) ) Q3_FF (.C(clk), .R(R), .CE(enable),
.Q(Q[3]), .D(U[3]));
    FDRE #(.INIT(1'b0) ) Q4_FF (.C(clk), .R(R), .CE(enable),
.Q(Q[4]), .D(U[4]));
    FDRE #(.INIT(1'b0) ) Q5_FF (.C(clk), .R(R), .CE(enable),
.Q(Q[5]), .D(U[5]));
    FDRE #(.INIT(1'b0) ) Q6_FF (.C(clk), .R(R), .CE(enable),
.Q(Q[6]), .D(U[6]));
    FDRE #(.INIT(1'b0) ) Q7_FF (.C(clk), .R(R), .CE(enable),
.Q(Q[7]), .D(U[7]));

    // increment
    assign U[0] = Q[0] ^ enable;
    assign U[1] = Q[1] ^ (Q[0] & enable);                   // Q[1]
XOR Q[0]
    assign U[2] = Q[2] ^ (Q[1] & Q[0] & enable);            // Q[2]
XOR (Q[1] AND Q[0])
    assign U[3] = Q[3] ^ (Q[2] & Q[1] & Q[0] & enable);     // Q[3]
XOR (Q[1] AND Q[0] AND Q[3])
    assign U[4] = Q[4] ^ (Q[3] & Q[2] & Q[1] & Q[0] & enable);
    assign U[5] = Q[5] ^ (Q[4] & Q[3] & Q[2] & Q[1] & Q[0] &
enable);
    assign U[6] = Q[6] ^ (Q[5] & Q[4] & Q[3] & Q[2] & Q[1] & Q[0] &
enable);
    assign U[7] = Q[7] ^ (Q[6] & Q[5] & Q[4] & Q[3] & Q[2] & Q[1] &
Q[0] & enable);

    //decrement
    assign D[0] = Q[0] ^ enable;
    assign D[1] = Q[1] ^ (~Q[0] & enable);
    // Q[1] XOR Q[0]
    assign D[2] = Q[2] ^ (~Q[1] & ~Q[0] & enable);
    // Q[2] XOR (Q[1] AND Q[0])
    assign D[3] = Q[3] ^ (~Q[2] & ~Q[1] & ~Q[0] & enable);
    // Q[3] XOR (Q[1] AND Q[0] AND Q[3])
    assign D[4] = Q[4] ^ (~Q[3] & ~Q[2] & ~Q[1] & ~Q[0] & enable);
    assign D[5] = Q[5] ^ (~Q[4] & ~Q[3] & ~Q[2] & ~Q[1] & ~Q[0] &
enable);
    assign D[6] = Q[6] ^ (~Q[5] & ~Q[4] & ~Q[3] & ~Q[2] & ~Q[1] &
~Q[0] & enable);
    assign D[7] = Q[7] ^ (~Q[6] & ~Q[5] & ~Q[4] & ~Q[3] & ~Q[2] &
~Q[1] & ~Q[0] & enable);

endmodule
```


cin
a
b
1
0
inp0_i
I0
O
RTL_INV
mux_s
inp[3:0]
sel[1:0]
o
m4_1
cout
mux_c
inp[3:0]
sel[1:0]
o
m4_1
sum

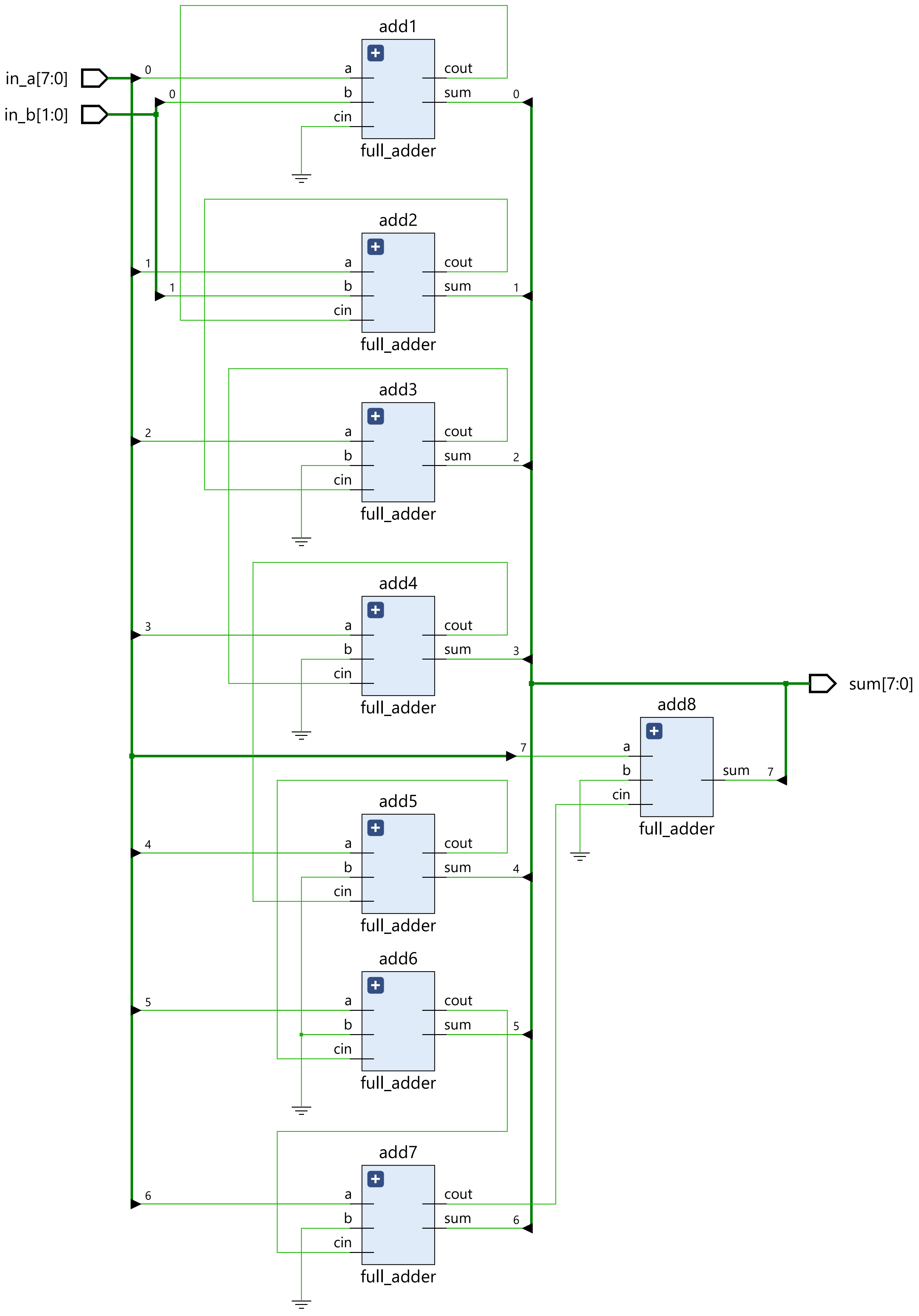
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
////////////////
// Company:
// Engineer:
//
// Create Date: 04/12/2022 01:44:49 PM
// Design Name:
// Module Name: full_adder
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////
////////////////


module full_adder(
    input a,
    input b,
    input cin,
    output cout,
    output sum
    );

    m4_1 mux_s (.inp({1'b1,cin,cin,1'b0}), .sel({a,b}), .o(cout));
    m4_1 mux_c (.inp({cin, ~cin, ~cin, cin}), .sel({a, b}),
.o(sum));

endmodule
```



n[3:0]
seg[6:0]
inp0_i
I0
O
0
RTL_INV
m1
m2
m3
m4
m5
m6
m7
inp[7:0]
sel[2:0]
o
m8_1
3:1
...
0
1
2
3
4
5
6

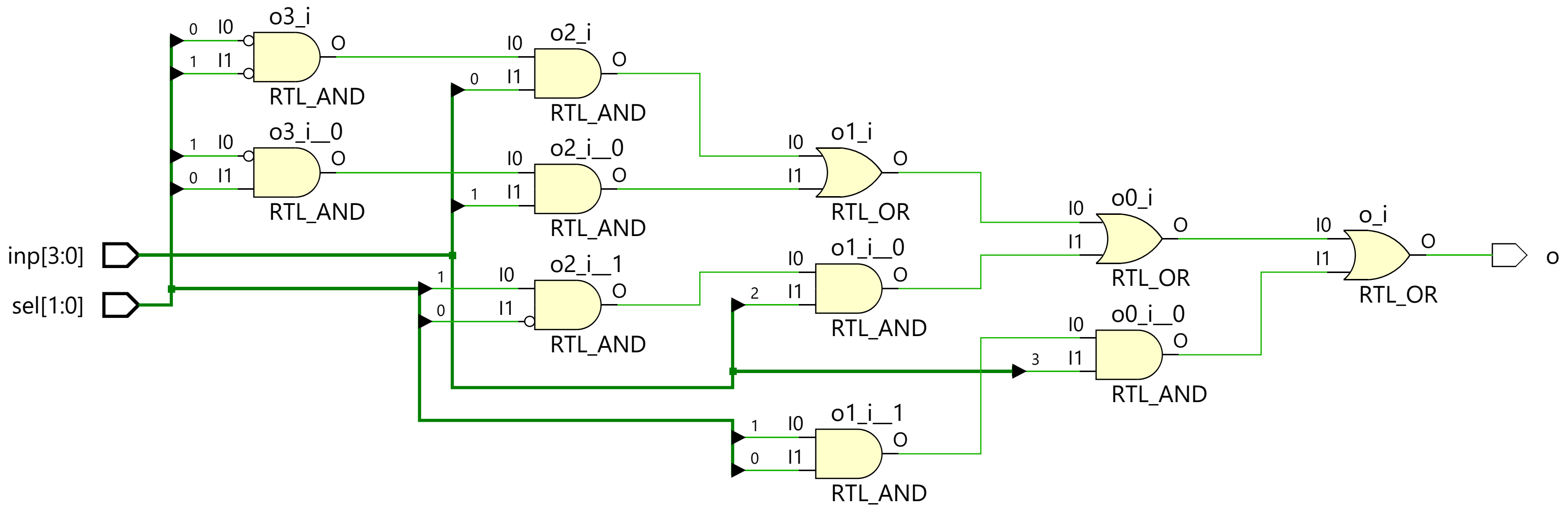
```
`timescale 1ns / 1ps
///////////////////////////////////////////////////////////////////
///////////////
// Company:
// Engineer:
//
// Create Date: 05/03/2022 02:36:09 PM
// Design Name:
// Module Name: hex7seg
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
///////////////////////////////////////////////////////////////////
///////////////


module hex7seg(
input [3:0] n,
    output [6:0] seg
    );

    m8_1 m1 (.inp({1'b0, n[0], n[0], 1'b0, 1'b0, ~n[0], 1'b0,
n[0]}), .sel({n[3], n[2], n[1]}), .o(seg[0]));
    m8_1 m2 (.inp({1'b1, ~n[0], n[0], 1'b0, ~n[0], n[0], 1'b0,
1'b0}), .sel({n[3], n[2], n[1]}), .o(seg[1]));
    m8_1 m3 (.inp({1'b1, ~n[0], 1'b0, 1'b0, 1'b0, 1'b0, ~n[0],
1'b0}), .sel({n[3], n[2], n[1]}), .o(seg[2]));
    m8_1 m4 (.inp({n[0], 1'b0, ~n[0], n[0], n[0], ~n[0], 1'b0,
n[0]}), .sel({n[3], n[2], n[1]}), .o(seg[3]));
```

```
    m8_1 m5 (.inp({1'b0, 1'b0, 1'b0, n[0], n[0], 1'b1, n[0], 
n[0]}), .sel({n[3], n[2], n[1]}), .o(seg[4]));
    m8_1 m6 (.inp({1'b0, n[0], 1'b0, 1'b0, n[0], 1'b0, 1'b1, 
n[0]}), .sel({n[3], n[2], n[1]}), .o(seg[5]));
    m8_1 m7 (.inp({1'b0, ~n[0], 1'b0, 1'b0, n[0], 1'b0, 1'b0, 
1'b1}), .sel({n[3], n[2], n[1]}), .o(seg[6]));

endmodule
```

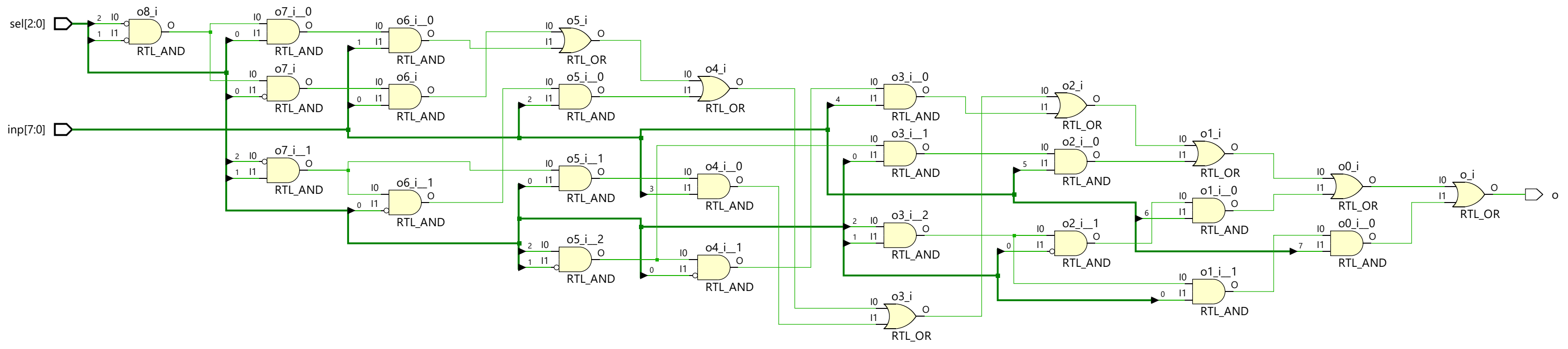

in_a[7:0]
in_b[1:0]
sum[7:0]
add1
add2
add3
add4
add5
add6
add7
add8
a
b
cin
cout
sum
full_adder

```
`timescale 1ns / 1ps
/////////////
////////////////////////////////////////////////////////////////////
//
// Company:
// Engineer:
//
// Create Date: 04/12/2022 01:59:04 PM
// Design Name:
// Module Name: incrementer
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////
///////////////


   module incrementer(
    input [7:0] in_a,
    input [1:0] in_b,
    output [7:0] sum
    );

    wire [7:0] w;

    full_adder add1 (.a(in_a[0]), .b(in_b[0]), .cin(1'b0),
.cout(w[0]), .sum(sum[0]));
    full_adder add2 (.a(in_a[1]), .b(in_b[1]), .cin(w[0]),
.cout(w[1]), .sum(sum[1]));
```

```
    full_adder add3 (.a(in_a[2]), .b(1'b0), .cin(w[1]),
.cout(w[2]), .sum(sum[2]));
    full_adder add4 (.a(in_a[3]), .b(1'b0), .cin(w[2]),
.cout(w[3]), .sum(sum[3]));
    full_adder add5 (.a(in_a[4]), .b(1'b0), .cin(w[3]),
.cout(w[4]), .sum(sum[4]));
    full_adder add6 (.a(in_a[5]), .b(1'b0), .cin(w[4]),
.cout(w[5]), .sum(sum[5]));
    full_adder add7 (.a(in_a[6]), .b(1'b0), .cin(w[5]),
.cout(w[6]), .sum(sum[6]));
    full_adder add8 (.a(in_a[7]), .b(1'b0), .cin(w[6]),
.cout(w[7]), .sum(sum[7]));

endmodule
```

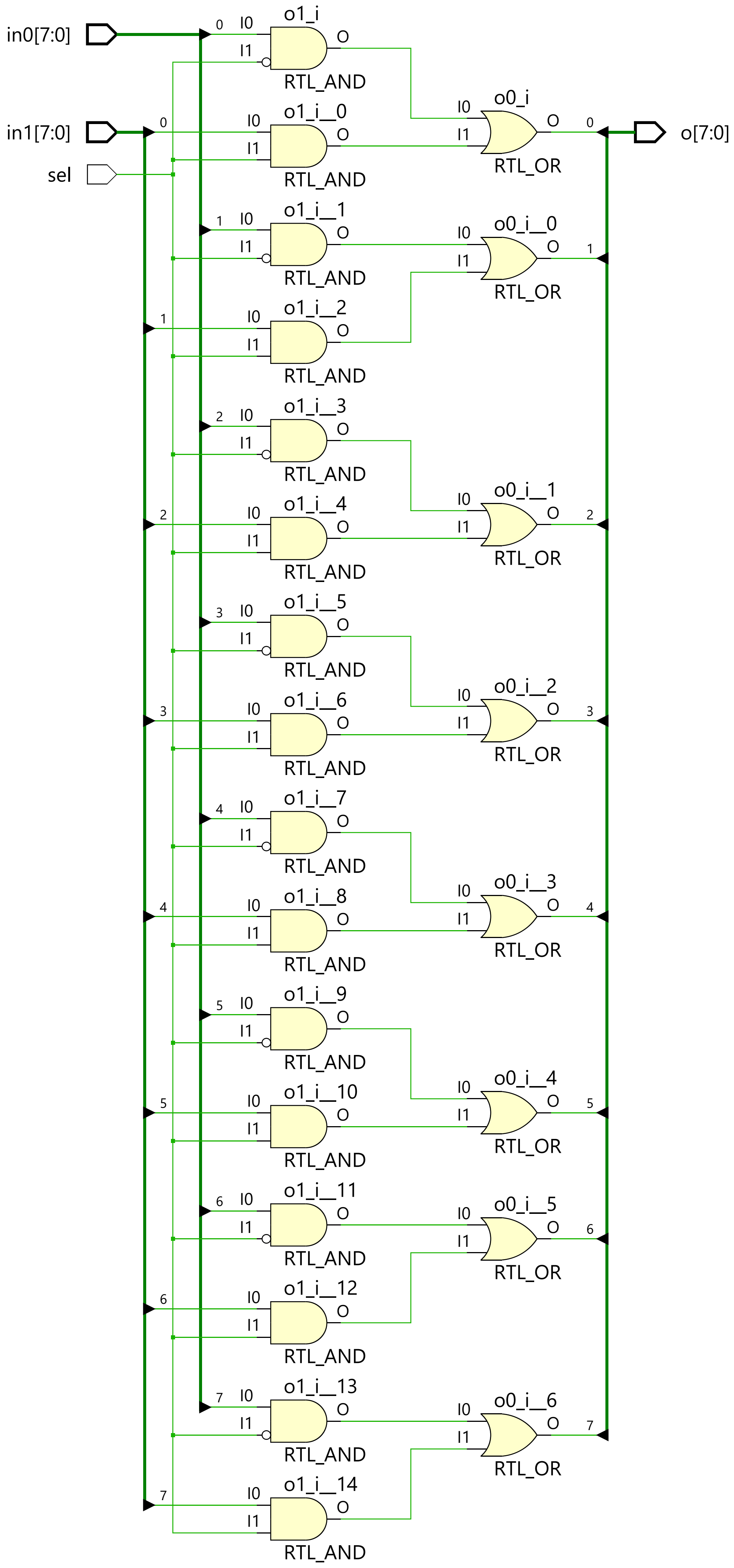

inp[3:0]
sel[1:0]
o3_i
RTL_AND
o3_i__0
RTL_AND
o2_i
RTL_AND
o2_i__0
RTL_AND
o2_i__1
RTL_AND
o1_i
RTL_OR
o1_i__0
RTL_AND
o1_i__1
RTL_AND
o0_i
RTL_OR
o0_i__0
RTL_AND
o_i
RTL_OR
o

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
///////////////
// Company:
// Engineer:
//
// Create Date: 04/11/2022 03:00:32 PM
// Design Name:
// Module Name: m4_1
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////
///////////////


module m4_1(
    input [1:0] sel,
    input [3:0] inp,
    output o
    );

    assign o = ((~sel[0]&~sel[1]&inp[0]) | (~sel[1]&sel[0]&inp[1])
| (sel[1]&~sel[0]&inp[2]) | (sel[1]&sel[0]&inp[3]));

endmodule
```

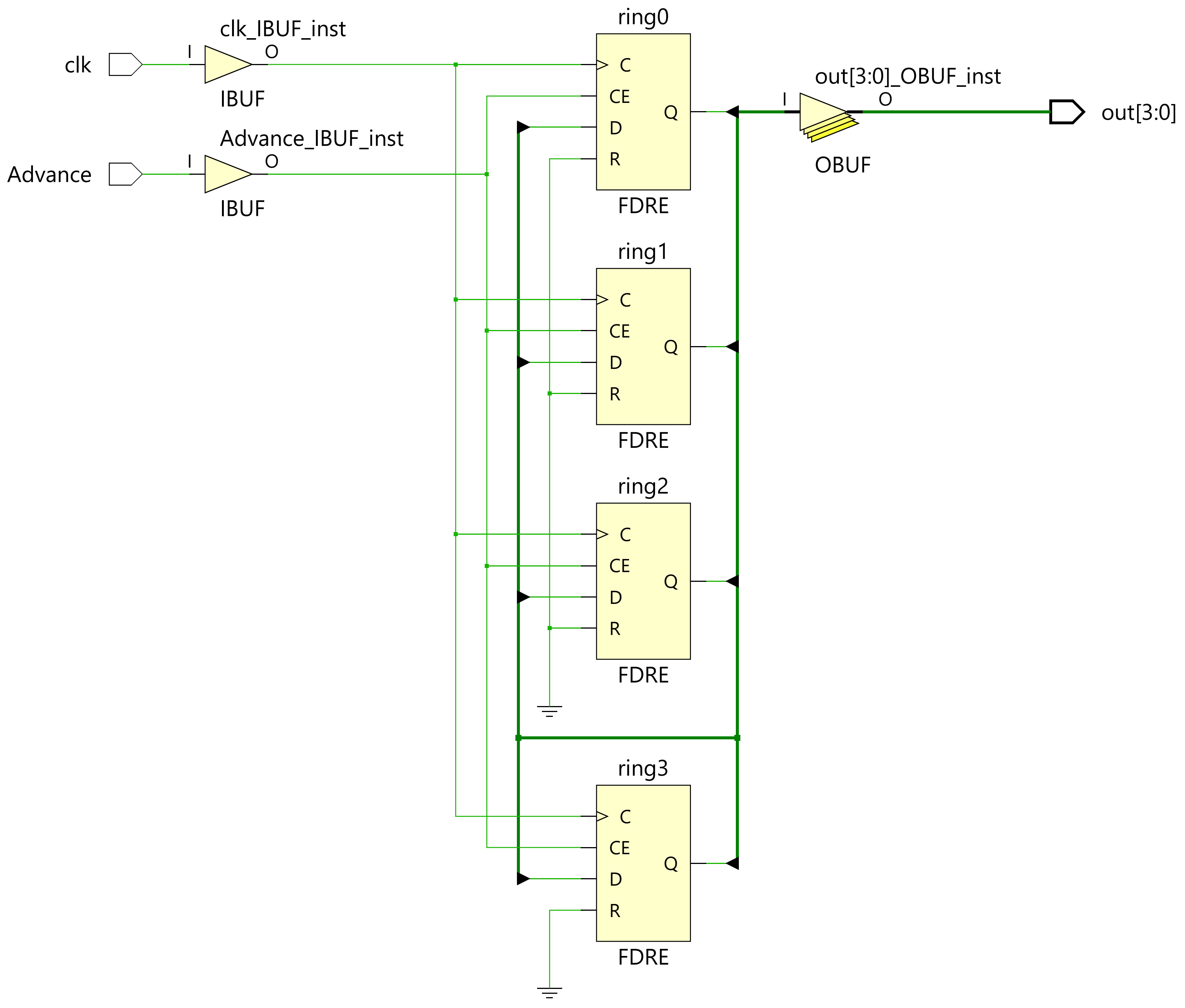


sel[2:0]
inp[7:0]
o8_i
RTL_AND
o7_i__0
o7_i
o7_i__1
o6_i__0
o6_i
o6_i__1
o5_i
RTL_OR
o5_i__0
o5_i__1
o5_i__2
o4_i
o4_i__0
o4_i__1
o3_i__0
o3_i__1
o3_i__2
o3_i
o2_i
o2_i__0
o2_i__1
o1_i
o1_i__0
o1_i__1
o0_i
o0_i__0
o_i
o

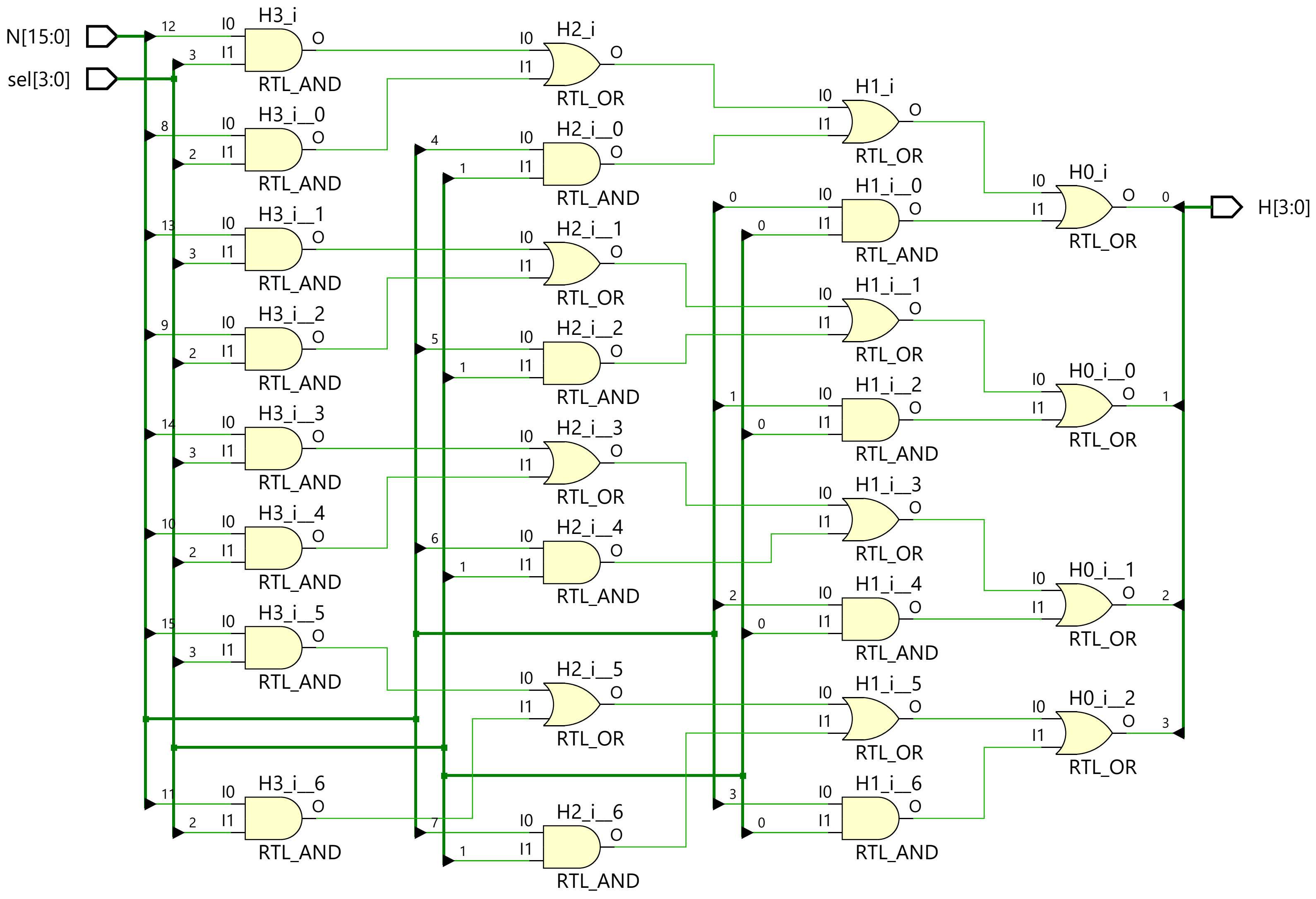
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////
///////////////
// Company:
// Engineer:
//
// Create Date: 05/03/2022 02:39:33 PM
// Design Name:
// Module Name: m8_1
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////
///////////////


module m8_1(
    input [2:0] sel,
    input [7:0] inp,
    output o
    );

    assign o = ((~sel[2]&~sel[1]&~sel[0]&inp[0]) |
(~sel[2]&~sel[1]&sel[0]&inp[1]) |
    (~sel[2]&sel[1]&~sel[0]&inp[2]) |
(~sel[2]&sel[1]&sel[0]&inp[3]) | (sel[2]&~sel[1]&~sel[0]&inp[4]) |
    (sel[2]&~sel[1]&sel[0]&inp[5]) | (sel[2]&sel[1]&~sel[0]&inp[6])
| (sel[2]&sel[1]&sel[0]&inp[7]));
endmodule
```


in0[7:0]
in1[7:0]
sel
o[7:0]
o1_i
o1_i__0
o1_i__1
o1_i__2
o1_i__3
o1_i__4
o1_i__5
o1_i__6
o1_i__7
o1_i__8
o1_i__9
o1_i__10
o1_i__11
o1_i__12
o1_i__13
o1_i__14
o0_i
o0_i__0
o0_i__1
o0_i__2
o0_i__3
o0_i__4
o0_i__5
o0_i__6
RTL_AND
RTL_OR
I0
I1
O

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
////////////////
// Company:
// Engineer:
//
// Create Date: 04/12/2022 10:08:10 PM
// Design Name:
// Module Name: m2_1x8
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////
////////////////


module m2_1x8(
    input [7:0] in0,
    input [7:0] in1,
    input sel,
    output [7:0] o
    );

    assign o[0] = (in0[0]&~sel) | (in1[0]&sel);
    assign o[1] = (in0[1]&~sel) | (in1[1]&sel);
    assign o[2] = (in0[2]&~sel) | (in1[2]&sel);
    assign o[3] = (in0[3]&~sel) | (in1[3]&sel);
    assign o[4] = (in0[4]&~sel) | (in1[4]&sel);
    assign o[5] = (in0[5]&~sel) | (in1[5]&sel);
    assign o[6] = (in0[6]&~sel) | (in1[6]&sel);
    assign o[7] = (in0[7]&~sel) | (in1[7]&sel);

endmodule
```


clk
clk_IBUF_inst
I
O
IBUF
Advance
Advance_IBUF_inst
I
O
IBUF
ring0
C
CE
D
R
Q
FDRE
ring1
C
CE
D
R
Q
FDRE
ring2
C
CE
D
R
Q
FDRE
ring3
C
CE
D
R
Q
FDRE
out[3:0]_OBUF_inst
I
O
OBUF
out[3:0]

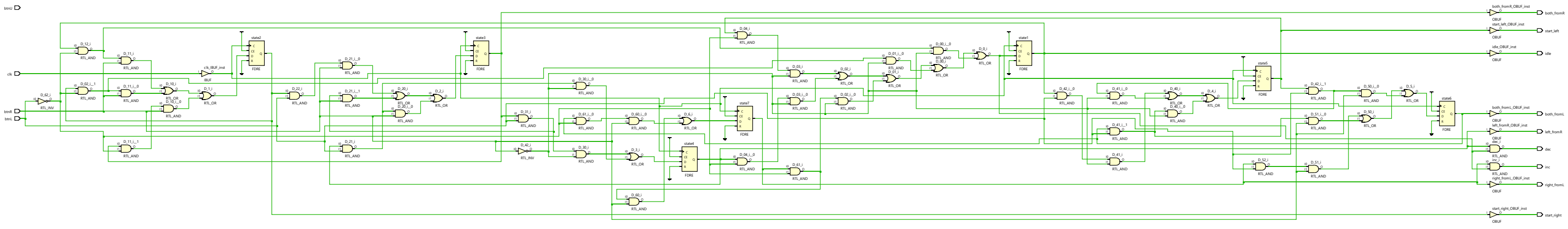
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
////////////////
// Company:
// Engineer:
//
// Create Date: 05/03/2022 02:42:09 PM
// Design Name:
// Module Name: ring_counter
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////
////////////////


module ring_counter(
    input clk,
    input Advance,
    output [3:0] out
    );

    FDRE #(.INIT(1'b1) ) ring3 (.C(clk), .R(1'b0), .CE(Advance),
.D(out[3]), .Q(out[0]));
    FDRE #(.INIT(1'b0) ) ring2 (.C(clk), .R(1'b0), .CE(Advance),
.D(out[0]), .Q(out[1]));
    FDRE #(.INIT(1'b0) ) ring1 (.C(clk), .R(1'b0), .CE(Advance),
.D(out[1]), .Q(out[2]));
    FDRE #(.INIT(1'b0) ) ring0 (.C(clk), .R(1'b0), .CE(Advance),
.D(out[2]), .Q(out[3]));

endmodule
```



N[15:0]
sel[3:0]
H3_i
H3_i__0
H3_i__1
H3_i__2
H3_i__3
H3_i__4
H3_i__5
H3_i__6
RTL_AND
H2_i
H2_i__0
H2_i__1
H2_i__2
H2_i__3
H2_i__4
H2_i__5
H2_i__6
RTL_OR
H1_i
H1_i__0
H1_i__1
H1_i__2
H1_i__3
H1_i__4
H1_i__5
H1_i__6
H0_i
H0_i__0
H0_i__1
H0_i__2
H[3:0]
I0
I1
O

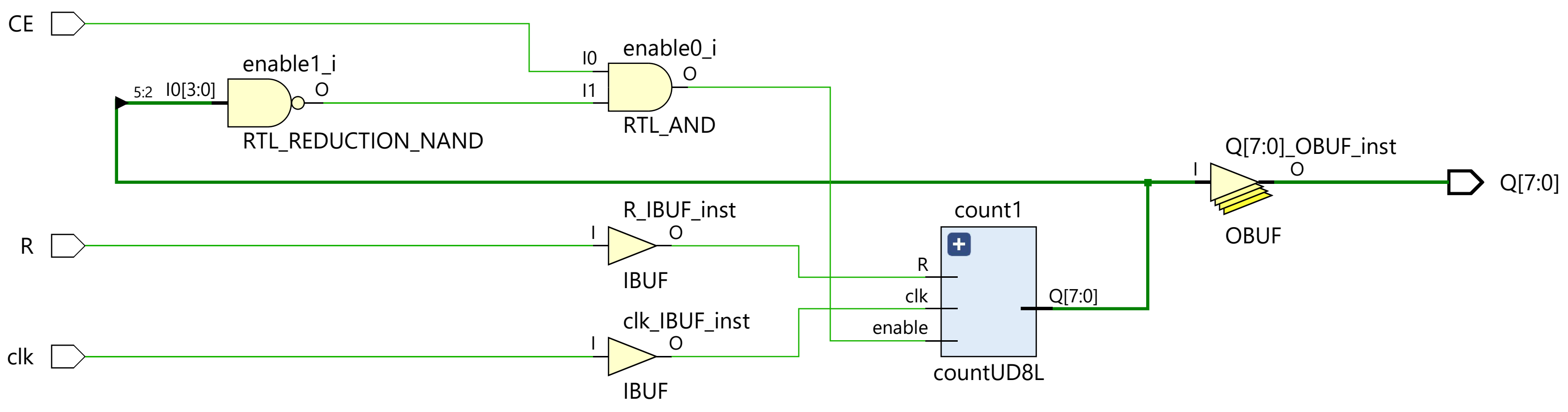
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
///////////////
// Company:
// Engineer:
//
// Create Date: 05/03/2022 02:37:49 PM
// Design Name:
// Module Name: selector
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////
///////////////


module selector(
    input [3:0] sel,
    input [15:0] N,
    output [3:0] H
    );

    //H is N[15:12] when sel=(1000)
    //H is N[11:8] when sel=(0100)
    //H is N[7:4] when sel=(0010)
    //H is N[3:0] when sel=(0001)

    assign H[0] = (N[12] & sel[3]) | (N[8] & sel[2]) | (N[4] &
sel[1]) |  (N[0] & sel[0]);
```

```
    assign H[1] = (N[13] & sel[3]) | (N[9] & sel[2]) | (N[5] &
sel[1]) |  (N[1] & sel[0]);
    assign H[2] = (N[14] & sel[3]) | (N[10] & sel[2]) | (N[6] &
sel[1]) |  (N[2] & sel[0]);
    assign H[3] = (N[15] & sel[3]) | (N[11] & sel[2]) | (N[7] &
sel[1]) |  (N[3] & sel[0]);

endmodule
```

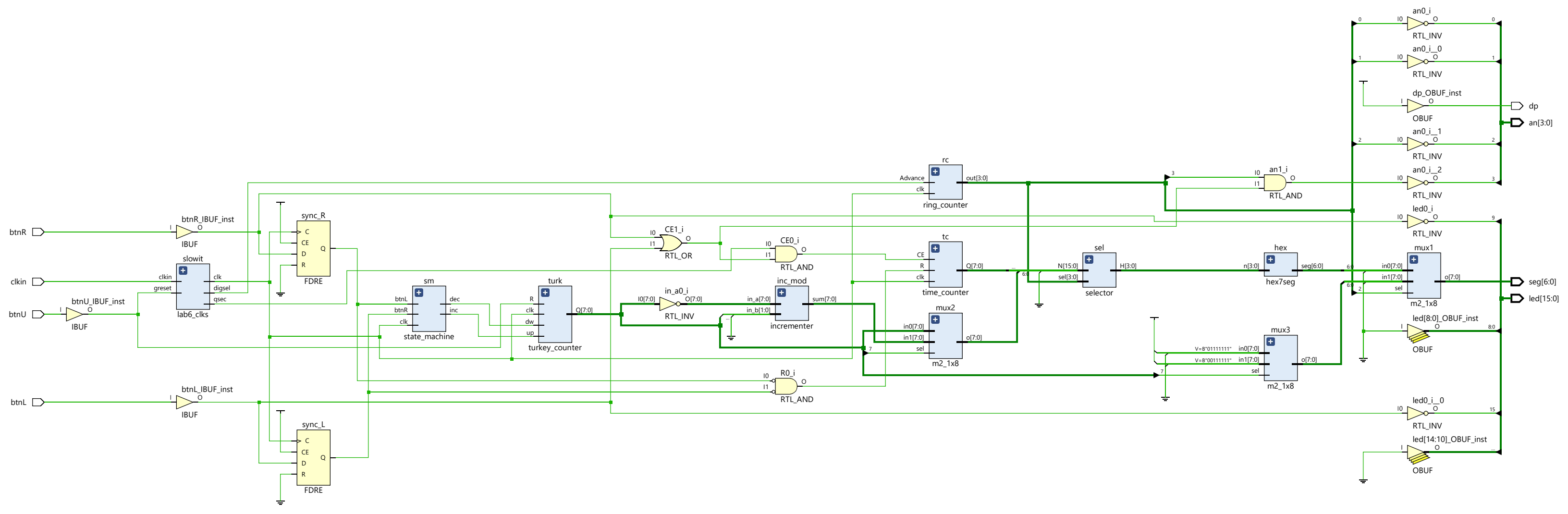

btnU
clk
btnR
btnL
both_fromR
start_left
idle
both_fromL
left_fromR
dec
inc
right_fromL
start_right

```
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////
///////////////
// Company:
// Engineer:
//
// Create Date: 05/10/2022 11:21:49 AM
// Design Name:
// Module Name: state_machine
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////
///////////////


module state_machine(
    input clk,
    input btnL,
    input btnR,
    input btnU,
    output dec,
    output inc,
    output idle,
    output start_left,
    output both_fromL,
    output right_fromL,
    output start_right,
    output both_fromR,
    output left_fromR
    );

    wire [6:0] D;
    wire [6:0] Q;

    assign idle = Q[0];
    assign start_right = Q[1];
    assign both_fromR = Q[2];
    assign left_fromR = Q[3];
    assign start_left = Q[4];
    assign both_fromL = Q[5];
    assign right_fromL = Q[6];

    assign inc = Q[6] & D[0];
    assign dec = Q[3] & D[0];

    assign D[0] = (Q[4] & ~btnL & ~btnR) | (Q[3] & ~btnL & ~btnR) |
(Q[6] & ~btnL & ~btnR) | (Q[1] & ~btnL & ~btnR) | (Q[0] & ~btnL &
~btnR);
    assign D[1] = (Q[0] & ~btnL & btnR) | (Q[1] & ~btnL & btnR) |
(Q[2] & ~btnL & btnR);
    assign D[2] = (Q[1] & btnL & btnR) | (Q[2] & btnL & btnR) |
(Q[3] & btnL & btnR);
    assign D[3] = (Q[2] & btnL & ~btnR) | (Q[3] & btnL & ~btnR);
    assign D[4] = (Q[0] & btnL & ~btnR) | (Q[4] & btnL & ~btnR) |
(Q[5] & btnL & ~btnR);
    assign D[5] = (Q[6] & btnL & btnR) | (Q[5] & btnL & btnR) |
(Q[4] & btnL & btnR);
    assign D[6] = (Q[6] & ~btnL & btnR) | (Q[5] & ~btnL & btnR);

    FDRE #(.INIT(1'b1) ) state1 (.C(clk), .CE(1'b1), .Q(Q[0]),
.D(D[0]));
    FDRE #(.INIT(1'b0) ) state2 (.C(clk), .CE(1'b1), .Q(Q[1]),
.D(D[1]));
    FDRE #(.INIT(1'b0) ) state3 (.C(clk), .CE(1'b1), .Q(Q[2]),
.D(D[2]));
```

```
    FDRE #(.INIT(1'b0) ) state4 (.C(clk), .CE(1'b1), .Q(Q[3]),
.D(D[3]));
    FDRE #(.INIT(1'b0) ) state5 (.C(clk), .CE(1'b1), .Q(Q[4]),
.D(D[4]));
    FDRE #(.INIT(1'b0) ) state6 (.C(clk), .CE(1'b1), .Q(Q[5]),
.D(D[5]));
    FDRE #(.INIT(1'b0) ) state7 (.C(clk), .CE(1'b1), .Q(Q[6]),
.D(D[6]));

endmodule
```


CE
enable1_i
5:2
I0[3:0]
O
RTL_REDUCTION_NAND
I0
I1
enable0_i
O
RTL_AND
Q[7:0]_OBUF_inst
I
O
OBUF
Q[7:0]
R
R_IBUF_inst
I
O
IBUF
count1
R
clk
enable
Q[7:0]
countUD8L
clk
clk_IBUF_inst
I
O
IBUF

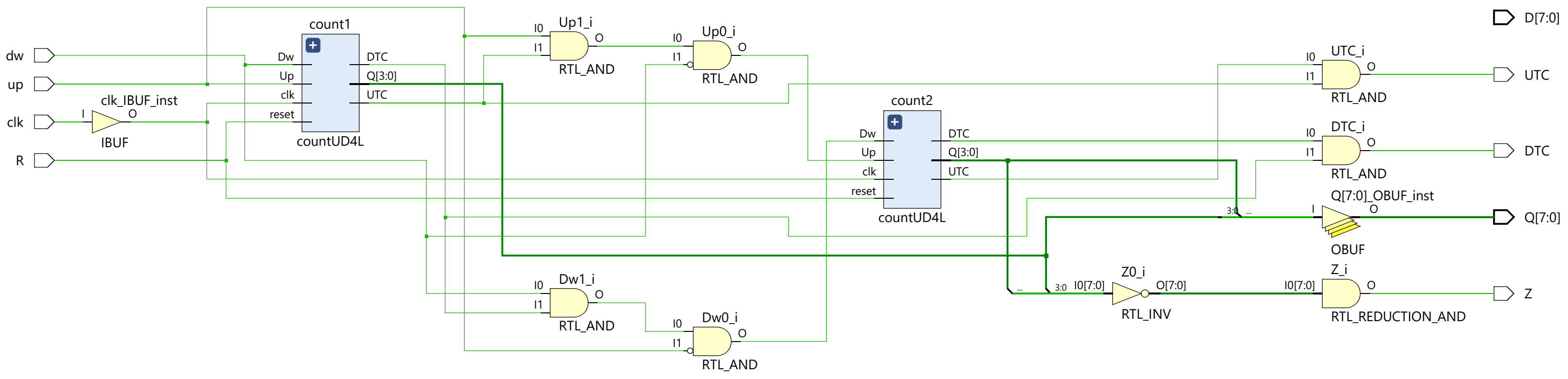
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
///////////////
// Company:
// Engineer:
//
// Create Date: 05/04/2022 01:40:45 PM
// Design Name:
// Module Name: time_counter
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////
///////////////


module time_counter(
    input CE,
    input R,
    input clk,
    output [7:0] Q
    );

    wire utc_wire;
    countUD8L count1 (.clk(clk), .enable(CE & ~&Q[5:2]), .Q(Q),
.R(R), .UTC(utc_wire));

    //wire [1:0] sel;
    //assign sel[0]= CE;
    //assign sel[1] = CE & utc_wire;

    //countUD4L count1 (.clk(clk), .enable(CE & ~utc_wire),
.reset(R), .Q(Q), .UTC(utc_wire));
    //countUD4L count2 (.clk(clk), .enable(sel[1]), .reset(R),
.Q(Q[7:4]));

endmodule
```


btnR
clkin
btnU
btnL
btnR_IBUF_inst
IBUF
btnU_IBUF_inst
IBUF
btnL_IBUF_inst
IBUF
slowit
clkin
greset
clk
digsel
qsec
lab6_clks
sync_R
C
CE
D
R
Q
FDRE
sync_L
C
CE
D
R
Q
FDRE
sm
btnL
btnR
clk
dec
inc
state_machine
turk
R
clk
dw
up
Q[7:0]
turkey_counter
CE1_i
I0
I1
O
RTL_OR
CE0_i
I0
I1
O
RTL_AND
in_a0_i
I0[7:0]
O[7:0]
RTL_INV
inc_mod
in_a[7:0]
in_b[1:0]
sum[7:0]
incrementer
R0_i
I0
I1
O
RTL_AND
rc
Advance
clk
out[3:0]
ring_counter
tc
CE
R
clk
Q[7:0]
time_counter
mux2
in0[7:0]
in1[7:0]
sel
o[7:0]
m2_1x8
sel
N[15:0]
sel[3:0]
H[3:0]
selector
an1_i
I0
I1
O
RTL_AND
hex
n[3:0]
seg[6:0]
hex7seg
mux3
V=B"01111111"
V=B"00111111"
in0[7:0]
in1[7:0]
sel
o[7:0]
m2_1x8
an0_i
RTL_INV
an0_i__0
RTL_INV
dp_OBUF_inst
OBUF
an0_i__1
RTL_INV
an0_i__2
RTL_INV
led0_i
RTL_INV
mux1
in0[7:0]
in1[7:0]
sel
o[7:0]
m2_1x8
led[8:0]_OBUF_inst
OBUF
led0_i__0
RTL_INV
led[14:10]_OBUF_inst
OBUF
dp
an[3:0]
seg[6:0]
led[15:0]

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
////////////////
// Company:
// Engineer:
//
// Create Date: 05/10/2022 11:22:18 AM
// Design Name:
// Module Name: top_level
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////
////////////////


module top_level(
    input btnR,
    input btnL,
    input btnU,
    input clkin,
    output [6:0] seg,
    output dp,
    output [3:0] an,
    output [15:0] led
    );

    //wires for lab6_clks
    wire clk;
    wire digsel;
    wire qsec;

    //wires for two's comp and neg logic
    wire [6:0] twos_final;
    wire [7:0] inc_wire;

    wire [6:0] seg_wire;
    wire [6:0] seg2_wire;

    //wires for state machine outputs
    wire up, down;
    wire idle_wire;
    wire start_left_wire;
    wire both_fromL_wire;
    wire right_fromL_wire;
    wire start_right_wire;
    wire both_fromR_wire;
    wire left_fromR_wire;

    //counter output buses
    wire [15:0] sel_input;
    wire [7:0] time_counter_wire;
    wire [7:0] turkey_counter_wire;

    wire btnL_wire;
    wire btnR_wire;
    wire [3:0] ring_wire;
    wire [3:0] sel_wire;

    // flip flops for syncronization of buttons
    FDRE #(.INIT(1'b0) ) sync_L (.C(clk), .CE(1'b1), .Q(btnL_wire),
.D(btnL));
    FDRE #(.INIT(1'b0) ) sync_R (.C(clk), .CE(1'b1), .Q(btnR_wire),
.D(btnR));

    assign sel_input[15:12] = time_counter_wire[5:2];
    assign sel_input[11:7] = 5'b00000;
    assign sel_input[6:0] = twos_final;
    assign led[9] = ~btnR;
    assign led[15] = ~btnL;
    assign led[8:0] = 9'b000000000;
    assign led[14:10] = 5'b00000;

    // muxes for negative logic
    m2_1x8 mux1 (.in0(seg_wire), .in1(seg2_wire),
.sel(ring_wire[2]), .o(seg));
    m2_1x8 mux3 (.in0(7'b1111111), .in1(7'b0111111),
.sel(turkey_counter_wire[7]), .o(seg2_wire));

    //logic for two's complement
    incrementer inc_mod (.in_a(~turkey_counter_wire),
.in_b(turkey_counter_wire[7]), .sum(inc_wire));
    m2_1x8 mux2 (.in0(turkey_counter_wire), .in1(inc_wire),
.sel(turkey_counter_wire[7]), .o(twos_final));

    //logic for an
    assign an[0] = ~ring_wire[0];
    assign an[1] = ~ring_wire[1];
    assign an[2] = ~(ring_wire[2]);
    assign an[3] = ~(ring_wire[3] & (btnR | btnL));
    assign dp = 1'b1;



    // call all functions
    lab6_clks slowit (.clkin(clkin), .greset(btnU), .clk(clk),
.digsel(digsel), .qsec(qsec));

    time_counter tc (.clk(clk), .R(~btnR_wire & ~btnL_wire),
.CE(qsec & (btnL | btnR)), .Q(time_counter_wire));

    turkey_counter turk (.clk(clk), .up(up), .R(btnU), .dw(down),
.Q(turkey_counter_wire));
```

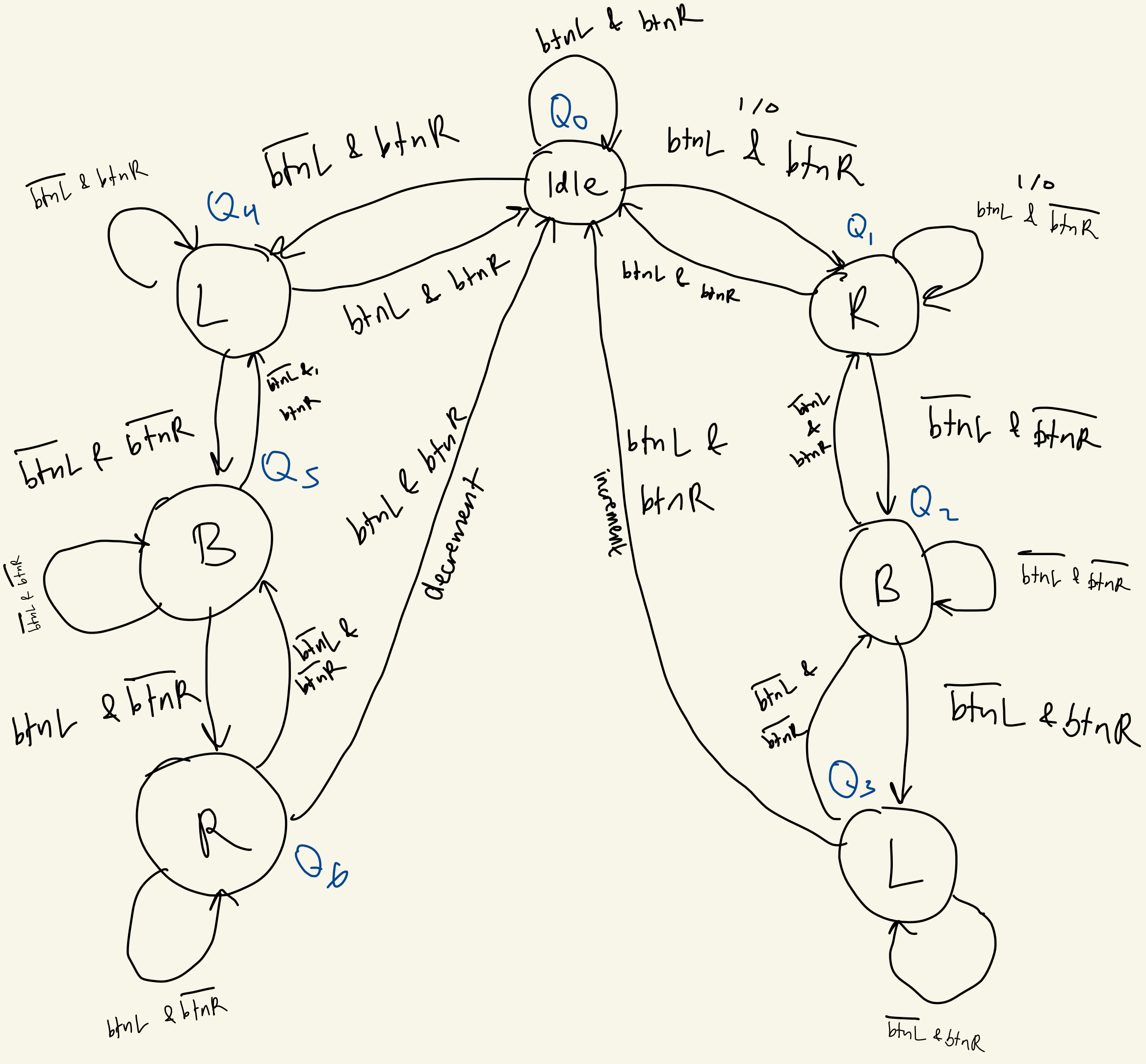
```
    state_machine sm (.clk(clk), .btnR(btnL_wire),
.btnL(btnR_wire), .inc(up), .dec(down), .idle(idle_wire),
.start_left(start_left_wire), .both_fromL(both_fromL_wire),
.right_fromL(right_fromL_wire), .start_right(start_right_wire),
.both_fromR(both_fromR_wire), .left_fromR(left_fromR_wire));

    ring_counter rc (.Advance(digsel), .clk(clk), .out(ring_wire));

    selector sel (.N(sel_input), .sel(ring_wire), .H(sel_wire));

    hex7seg hex (.n(sel_wire), .seg(seg_wire));



endmodule
```

```
`timescale 1ns / 1ps
///////////////////////////////////////////////////////////////////
///////////////
// Company:
// Engineer:
//
// Create Date: 05/10/2022 11:21:14 AM
// Design Name:
// Module Name: turkey_counter
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
///////////////////////////////////////////////////////////////////
///////////////


module turkey_counter(
    input up,
    input dw,
    input clk,
    input R,
    output [7:0] Q,
    output [7:0] D,
    output DTC,
    output UTC,
    output Z
    );

    wire [1:0] utc_wire;
    wire [1:0] dtc_wire;
    wire [7:0] invert;
    wire [7:0] c;

    assign Z = &(~(Q[7:0]));
    assign UTC = utc_wire[1] & utc_wire[0];
    assign DTC = dtc_wire[1] & dtc_wire[0];

    countUD4L count1 (.clk(clk), .Up(up), .Dw(dw), .reset(R),
.Q(Q[3:0]), .UTC(utc_wire[0]), .DTC(dtc_wire[0]));
    countUD4L count2 (.clk(clk), .Up(up & utc_wire[0] & ~dw),
.reset(R), .Dw(dw & dtc_wire[0] & ~up), .UTC(utc_wire[1]),
.DTC(dtc_wire[1]), .Q(Q[7:4]));

endmodule
```



D[7:0]
count1
Up1_i
Up0_i
UTC_i
I0
I1
O
dw
up
clk
R
Dw
Up
clk
reset
DTC
Q[3:0]
UTC
clk_IBUF_inst
IBUF
countUD4L
RTL_AND
count2
DTC_i
Q[7:0]_OBUF_inst
OBUF
Q[7:0]
Z0_i
Z_i
I0[7:0]
O[7:0]
RTL_INV
RTL_REDUCTION_AND
Z
Dw1_i
Dw0_i

State Machine



State Machine Equations

$$D_0 = \left(Q_4 \cdot \left(\overline{btnL} \cdot \overline{btnR}\right)\right) + \left(Q_3 \cdot \left(\overline{btnL} \cdot \overline{btnR}\right)\right) + \left(Q_6 \cdot \left(\overline{btnL} \cdot \overline{btnR}\right)\right) + \left(Q_1 \cdot \left(\overline{btnL} \cdot \overline{btnR}\right)\right)$$
$$+ \left(Q_0 \cdot \left(btnL \cdot btnR\right)\right)$$

$$D_1 = \left(Q_0 \left(\overline{btnL} \cdot btnR\right)\right) + \left(Q_1 \cdot \left(\overline{btnL} \cdot btnR\right)\right) + \left(Q_2 \cdot \left(\overline{btnL} \cdot btnR\right)\right)$$

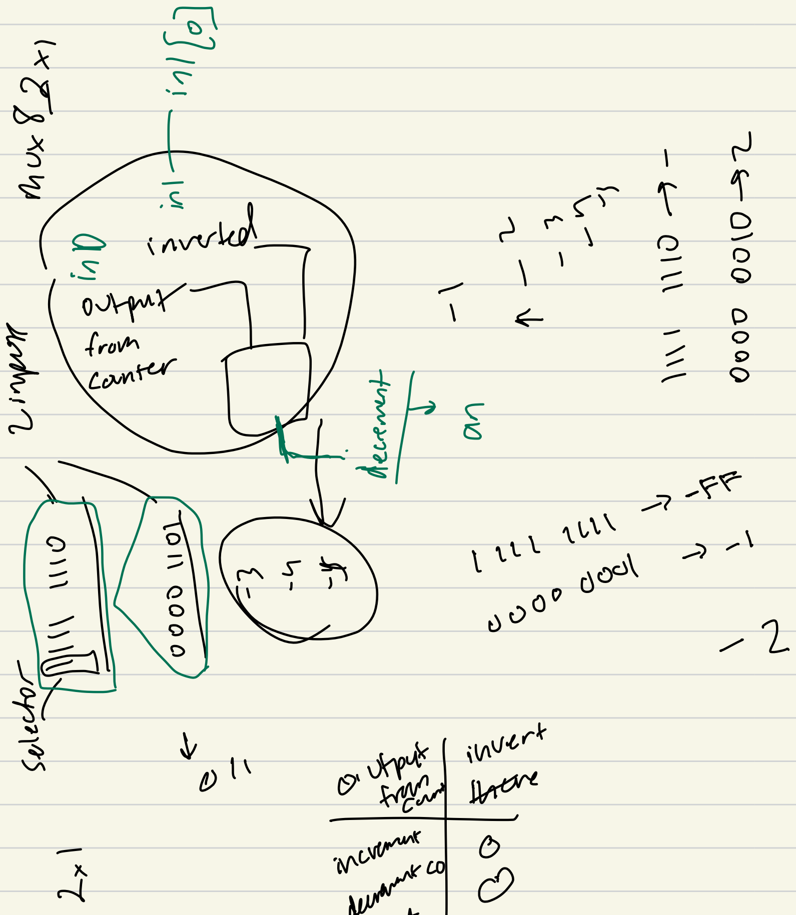
$$D_2 = \left(Q_1 \cdot \left(btnL \cdot btnR\right)\right) + \left(Q_2 \cdot \left(btnL \cdot btnR\right)\right) + \left(Q_3 \cdot \left(btnL \cdot btnR\right)\right)$$

$$D_3 = \left(Q_2 \cdot \left(btnL \cdot \overline{btnR}\right)\right) + \left(Q_3 \cdot \left(btnL \cdot btnR\right)\right)$$

$$D_4 = \left(Q_5 \cdot \left(\overline{btnL} \cdot btnR\right)\right) + \left(Q_4 \cdot \left(\overline{btnL} \cdot btnR\right)\right)$$

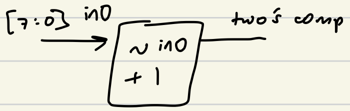
$$D_5 = \left(Q_6 \cdot \left(btnL \cdot btnR\right)\right) + \left(Q_5 \cdot \left(btnL \cdot btnR\right)\right) + \left(Q_4 \cdot \left(btnL \cdot btnR\right)\right)$$

$$D_6 = \left(Q_0 \cdot \left(btnL \cdot \overline{btnR}\right)\right) + \left(Q_6 \cdot \left(btnL \cdot \overline{btnR}\right)\right) + \left(Q_5 \cdot \left(btnL \cdot \overline{btnR}\right)\right)$$



output from Counter	invert ~~there~~
increment	0
decrement C0	00
decrement	0

2's complement



in0 ^ in0[7]

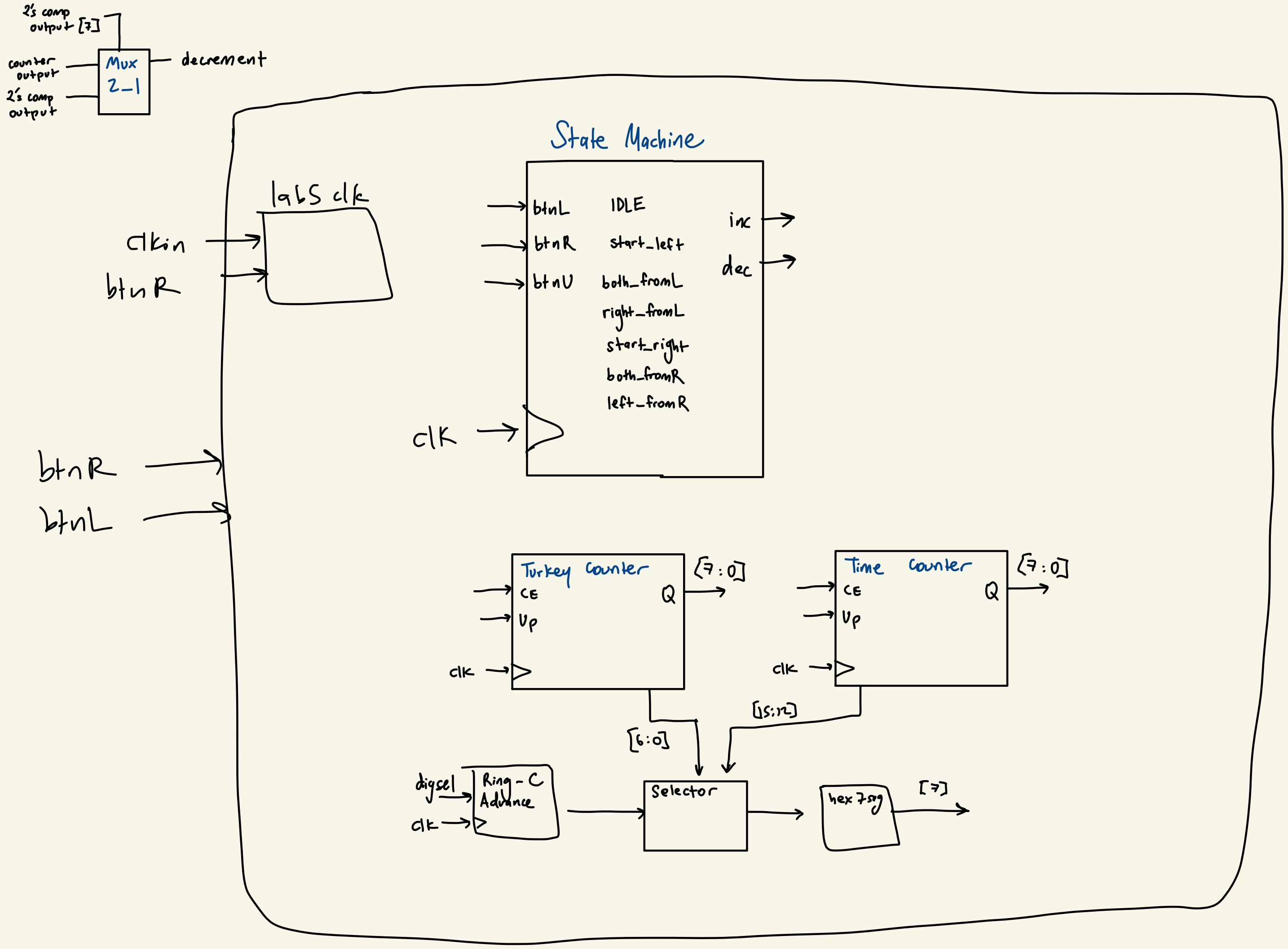
+1

in0 = 0000 0001 → +1

1111 1110

+ 1

1111 1111 → -1


2's comp output [7]
counter output
2's comp output
Mux 2_1
decrement
State Machine
lab5 clk
clkin
btnR
btnL
btnR
btnU
IDLE
start_left
both_fromL
right_fromL
start_right
both_fromR
left_fromR
inc
dec
clk
btnR
btnL
Turkey Counter
CE
Up
clk
Q
[7:0]
Time Counter
CE
Up
clk
Q
[7:0]
[6:0]
[15:12]
digsel
clk
Ring-C Advance
Selector
hex 7seg
[7]