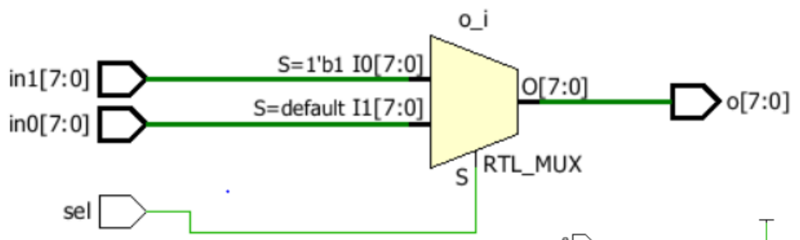
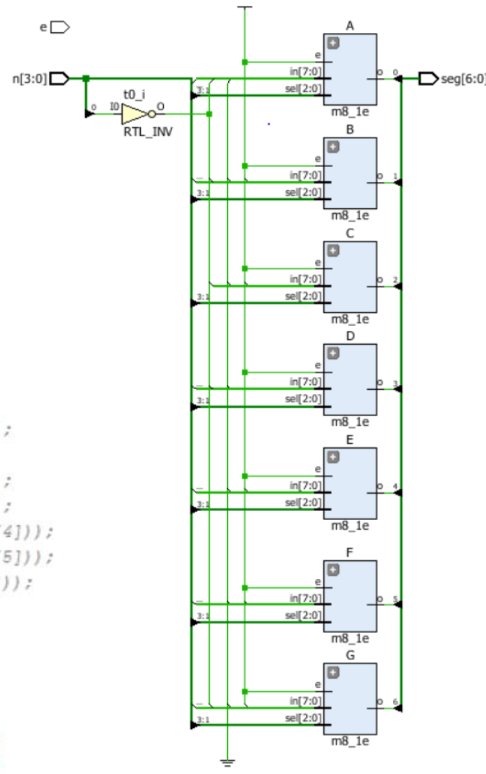
Mux 2_8x1

```
module m2_1x8(
    input [7:0] in0,
    input [7:0] in1,
    input sel,
    output [7:0] o
    );
    assign o = sel ? in1:in0;
 endmodule
```



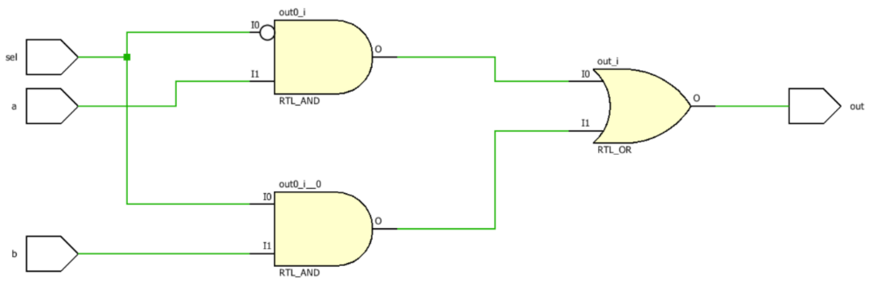


Hex7Seg

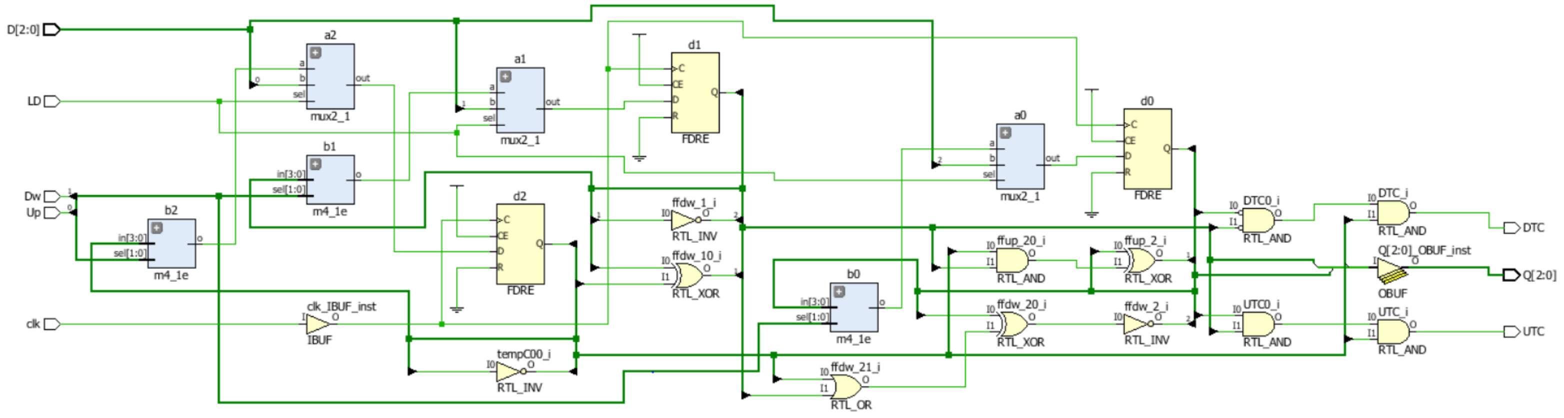
```
module hex7seg(
    input [3:0] n,
    input e,
    output [6:0] seg
    );

    wire t0 = ~n[0];
    wire [6:0] te;
    /*
    //m8_1e A( .in({n[0],1'b0,t0,1'b0,1'b0,n[0],n[0],1'b0}), .sel(n[3:1]), .e(1'b1), .o(seg[0]));
    m8_1e B( .in({1'b0,1'b0,n[0],t0,1'b0,1'b0,1'b0,1'b1}), .sel(n[3:1]), .e(1'b1), .o(seg[1]));
        m8_1e C( .in({1'b0,t0,1'b0,1'b0,1'b0,1'b0,t0,1'b1}), .sel(n[3:1]), .e(1'b1), .o(seg[2]));
        m8_1e D( .in({n[0],1'b0,t0,n[0],n[0],t0,1'b0,n[0]}), .sel(n[3:1]), .e(1'b1), .o(seg[3]));
        m8_1e E( .in({n[0],n[0],1'b1,n[0],n[0],1'b0,1'b0,1'b0}), .sel(n[3:1]), .e(1'b1), .o(seg[4]));
        m8_1e F( .in({n[0],1'b1,1'b0,n[0],1'b0,1'b0,n[0],1'b0}), .sel(n[3:1]), .e(1'b1), .o(seg[5]));
        m8_1e G( .in({1'b1,1'b0,1'b0,n[0],1'b0,1'b0,t0,1'b0}), .sel(n[3:1]), .e(1'b1), .o(seg[6]));
        */
    m8_1e A( .in({1'b0,n[0],n[0],1'b0,1'b0,t0,1'b0,n[0]}), .sel(n[3:1]), .e(1'b1), .o(te[0]));
    m8_1e B( .in({1'b1,t0,n[0],1'b0,t0,n[0],1'b0,1'b0}), .sel(n[3:1]), .e(1'b1), .o(te[1]));
    m8_1e C( .in({1'b1,t0,1'b0,1'b0,1'b0,1'b0,t0,1'b0}), .sel(n[3:1]), .e(1'b1), .o(te[2]));
    m8_1e D( .in({n[0],1'b0,t0,n[0],n[0],t0,1'b0,n[0]}), .sel(n[3:1]), .e(1'b1), .o(te[3]));
    m8_1e E( .in({1'b0,1'b0,1'b0,n[0],n[0],1'b1,n[0],n[0]}), .sel(n[3:1]), .e(1'b1), .o(te[4]));
    m8_1e F( .in({1'b0,n[0],1'b0,1'b0,n[0],1'b0,1'b1,n[0]}), .sel(n[3:1]), .e(1'b1), .o(te[5]));
    m8_1e G( .in({1'b0,t0,1'b0,1'b0,n[0],1'b0,1'b0,1'b1}), .sel(n[3:1]), .e(1'b1), .o(te[6]));
    //assign seg = ~seg;
   assign seg = te;
endmodule
```

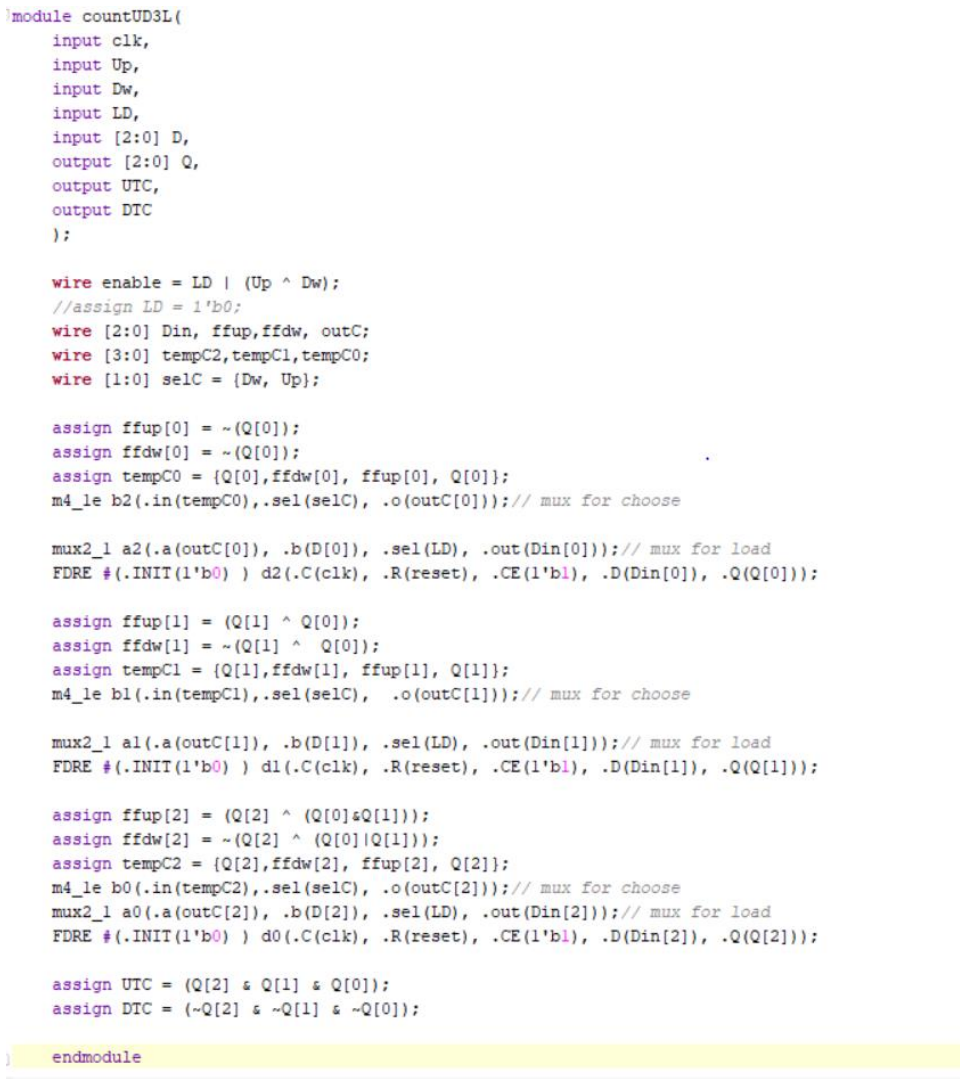
Mux 2_1

```
module mux2_1(
    input a,
    input b,
    input sel,
    output out
    );
    assign out = (~sel&a) | (sel&b);
endmodule
```



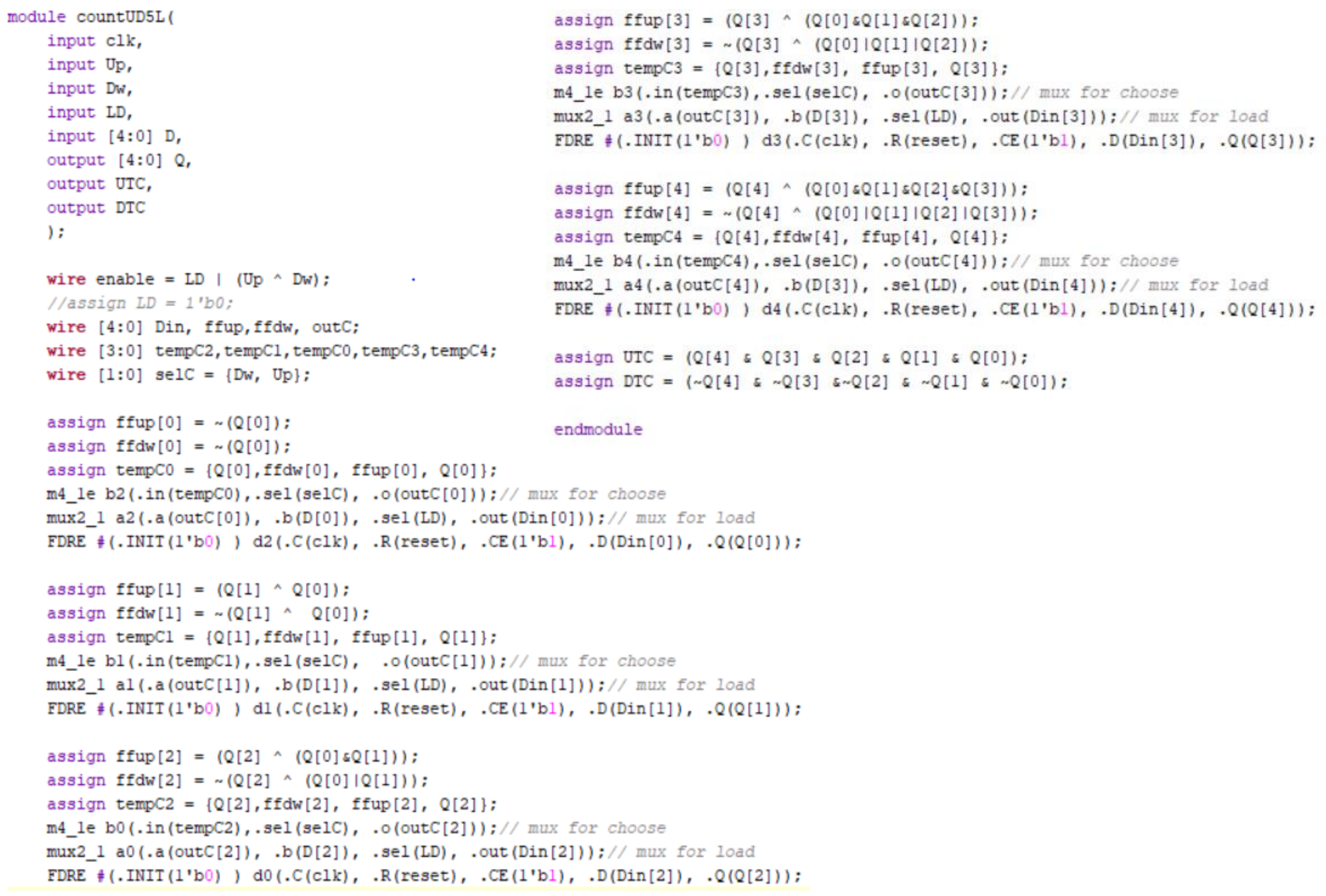
CountUD3L



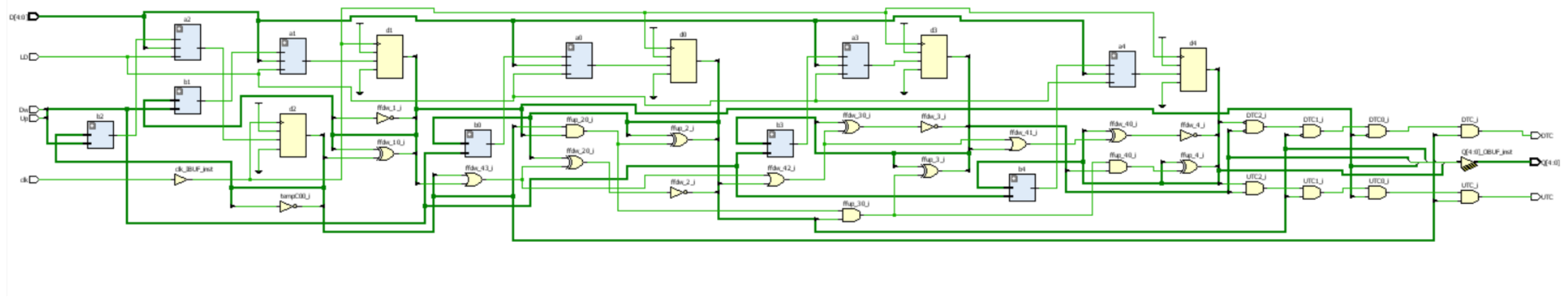


```
module countUD3L(
    input clk,
    input Up,
    input Dw,
    input LD,
    input [2:0] D,
    output [2:0] Q,
    output UTC,
    output DTC
    );

    wire enable = LD | (Up ^ Dw);
    //assign LD = 1'b0;
    wire [2:0] Din, ffup,ffdw, outC;
    wire [3:0] tempC2,tempC1,tempC0;
    wire [1:0] selC = {Dw, Up};

    assign ffup[0] = ~(Q[0]);
    assign ffdw[0] = ~(Q[0]);
    assign tempC0 = {Q[0],ffdw[0], ffup[0], Q[0]};
    m4_1e b2(.in(tempC0),.sel(selC), .o(outC[0]));// mux for choose

    mux2_1 a2(.a(outC[0]), .b(D[0]), .sel(LD), .out(Din[0]));// mux for load
    FDRE #(.INIT(1'b0) ) d2(.C(clk), .R(reset), .CE(1'b1), .D(Din[0]), .Q(Q[0]));

    assign ffup[1] = (Q[1] ^ Q[0]);
    assign ffdw[1] = ~(Q[1] ^  Q[0]);
    assign tempC1 = {Q[1],ffdw[1], ffup[1], Q[1]};
    m4_1e b1(.in(tempC1),.sel(selC),  .o(outC[1]));// mux for choose

    mux2_1 a1(.a(outC[1]), .b(D[1]), .sel(LD), .out(Din[1]));// mux for load
    FDRE #(.INIT(1'b0) ) d1(.C(clk), .R(reset), .CE(1'b1), .D(Din[1]), .Q(Q[1]));

    assign ffup[2] = (Q[2] ^ (Q[0]&Q[1]));
    assign ffdw[2] = ~(Q[2] ^ (Q[0]|Q[1]));
    assign tempC2 = {Q[2],ffdw[2], ffup[2], Q[2]};
    m4_1e b0(.in(tempC2),.sel(selC), .o(outC[2]));// mux for choose
    mux2_1 a0(.a(outC[2]), .b(D[2]), .sel(LD), .out(Din[2]));// mux for load
    FDRE #(.INIT(1'b0) ) d0(.C(clk), .R(reset), .CE(1'b1), .D(Din[2]), .Q(Q[2]));

    assign UTC = (Q[2] & Q[1] & Q[0]);
    assign DTC = (~Q[2] & ~Q[1] & ~Q[0]);

    endmodule
```

CountUD5L



```
module countUD5L(
    input clk,
    input Up,
    input Dw,
    input LD,
    input [4:0] D,
    output [4:0] Q,
    output UTC,
    output DTC
    );

    wire enable = LD | (Up ^ Dw);
    //assign LD = 1'b0;
    wire [4:0] Din, ffup,ffdw, outC;
    wire [3:0] tempC2,tempC1,tempC0,tempC3,tempC4;
    wire [1:0] selC = {Dw, Up};

    assign ffup[0] = ~(Q[0]);
    assign ffdw[0] = ~(Q[0]);
    assign tempC0 = {Q[0],ffdw[0], ffup[0], Q[0]};
    m4_le b2(.in(tempC0),.sel(selC), .o(outC[0]));// mux for choose
    mux2_1 a2(.a(outC[0]), .b(D[0]), .sel(LD), .out(Din[0]));// mux for load
    FDRE #(.INIT(1'b0) ) d2(.C(clk), .R(reset), .CE(1'b1), .D(Din[0]), .Q(Q[0]));

    assign ffup[1] = (Q[1] ^ Q[0]);
    assign ffdw[1] = ~(Q[1] ^  Q[0]);
    assign tempC1 = {Q[1],ffdw[1], ffup[1], Q[1]};
    m4_le b1(.in(tempC1),.sel(selC),  .o(outC[1]));// mux for choose
    mux2_1 a1(.a(outC[1]), .b(D[1]), .sel(LD), .out(Din[1]));// mux for load
    FDRE #(.INIT(1'b0) ) d1(.C(clk), .R(reset), .CE(1'b1), .D(Din[1]), .Q(Q[1]));

    assign ffup[2] = (Q[2] ^ (Q[0]&Q[1]));
    assign ffdw[2] = ~(Q[2] ^ (Q[0]|Q[1]));
    assign tempC2 = {Q[2],ffdw[2], ffup[2], Q[2]};
    m4_le b0(.in(tempC2),.sel(selC), .o(outC[2]));// mux for choose
    mux2_1 a0(.a(outC[2]), .b(D[2]), .sel(LD), .out(Din[2]));// mux for load
    FDRE #(.INIT(1'b0) ) d0(.C(clk), .R(reset), .CE(1'b1), .D(Din[2]), .Q(Q[2]));

    assign ffup[3] = (Q[3] ^ (Q[0]&Q[1]&Q[2]));
    assign ffdw[3] = ~(Q[3] ^ (Q[0]|Q[1]|Q[2]));
    assign tempC3 = {Q[3],ffdw[3], ffup[3], Q[3]};
    m4_le b3(.in(tempC3),.sel(selC), .o(outC[3]));// mux for choose
    mux2_1 a3(.a(outC[3]), .b(D[3]), .sel(LD), .out(Din[3]));// mux for load
    FDRE #(.INIT(1'b0) ) d3(.C(clk), .R(reset), .CE(1'b1), .D(Din[3]), .Q(Q[3]));

    assign ffup[4] = (Q[4] ^ (Q[0]&Q[1]&Q[2]&Q[3]));
    assign ffdw[4] = ~(Q[4] ^ (Q[0]|Q[1]|Q[2]|Q[3]));
    assign tempC4 = {Q[4],ffdw[4], ffup[4], Q[4]};
    m4_le b4(.in(tempC4),.sel(selC), .o(outC[4]));// mux for choose
    mux2_1 a4(.a(outC[4]), .b(D[3]), .sel(LD), .out(Din[4]));// mux for load
    FDRE #(.INIT(1'b0) ) d4(.C(clk), .R(reset), .CE(1'b1), .D(Din[4]), .Q(Q[4]));

    assign UTC = (Q[4] & Q[3] & Q[2] & Q[1] & Q[0]);
    assign DTC = (~Q[4] & ~Q[3] &~Q[2] & ~Q[1] & ~Q[0]);

endmodule
```


D[4:0]
LD
Dw
Up
clk
a2
a1
a0
a3
a4
b0
b1
b2
b3
b4
d0
d1
d2
d3
d4
clk_IBUF_inst
tempC00_i
DTC
UTC
Q[4:0]
Q[4:0]_OBUF_inst