```
1   //ee417 lesson 4 Assignment 3 Part 2, L4A3P2
2   // Name: Lamin Jammeh, Date: 06-09-24  Group: Kalin/Jammeh
3   //top module BCD_or_2421_up_down_counter
4   module BCD_or_2421_up_down_counter(output [3:0] count_out,
5                                      input clk,
6                                      input reset,
7                                      input enable,
8                                      input select_up_down,
9                                      input select_code2421_BCD );
10  //define internal wires
11  wire [3:0] downCount2421;
12  wire [3:0] upCount2421;
13  wire [3:0] count2421code;
14  wire [3:0] countBCD;
15  //instantiate submodules
16  code2421_downCounter DNCNT (downCount2421,clk,reset,enable);
17  code2421_upCounter UPCNT (upCount2421, downCount2421);
18  mux2to1_4bit MUX1 (downCount2421, upCount2421, select_up_down, count2421code);
19  Weighted2421ToBCDConverter CONV (count2421code, countBCD);
20  mux2to1_4bit MUX2 (count2421code, countBCD, select_code2421_BCD, count_out);
21
22  endmodule
23  //---------------------------------BLOCK
    A-------------------------------------------------------
24  //code2421_down_counter using Moore FSM with case structure
25  module code2421_downCounter (output reg [3:0] count,
26                               input clk,
27                               input reset,
28                               input enable);
29  //define the states
30  reg [3:0] state, next_state;
31
32  //define parameters
33
34
35  //sequential logic
36  always @(posedge clk or posedge reset)
37     if (reset)
38           state <= 4'b1111;
39     else if (enable) state<= next_state; //if reset high, reset, else..countdown
40
41     always @ * //any variable changes, combinational logic from given state table
42       case (state)
43         0: begin //state 0
44             next_state =1;
45             count=4'b1111;
46           end
47         1: begin //state 1
48             next_state =2;
49             count=4'b1110;
50           end
51         2: begin //state 2
52             next_state =3;
53             count=4'b1101;
54           end
55         3: begin //state 3
56             next_state =4;
57             count=4'b1100;
58           end
59         4: begin //state 4
60             next_state =5;
61             count=4'b1011;
62           end
63         5: begin //state 5
64             next_state =6;
65             count=4'b0100;
66           end
67         6: begin //state 6
68             next_state =7;
```

```
            count=4'b0011;
          end
        7: begin //state 7
            next_state =8;
            count=4'b0010;
          end
        8: begin //state 8
            next_state =9;
            count=4'b0001;
          end
        9: begin //state 9
            next_state =0;
            count=4'b0000;
          end
    default: next_state <=0;
              //count=4'b1111;
      endcase
endmodule

/*-------------------------------------------------------------------------------
Project Description: This portion takes use the output of the code2421_downCounter
and inverts it to form an up counter. this is possible because of the complementary
characteristics of 2421 code
-------------------------------------------------------------------------------*/
//define the input as the output of the downCounter and the assign and output
// note the system is already initialize from the code2421_downCounter block
//define the output as wire since the upcount block is intermediate to enable assignment
operation
module code2421_upCounter (output [3:0] upcount, input [3:0] count);

//wire  [3:0]     count;

////call the file for the code2421_downCounter
//code2421_downCounter UDM_downCount (.count(count),
                                    //   .clk(clk),
                                    //   .reset(reset),
                                    //   .enable(enable));
//assign a function to the output of the upcounter
assign upcount = ~count;

endmodule

//4-bit wide 2 to 1 multiplexer
module mux2to1_4bit(
    input [3:0] data0, // 4-bit input data 0
    input [3:0] data1, // 4-bit input data 1
    input select,      // Select signal (0 or 1), z
    output reg [3:0] muxout); //4-bit output
    always @ (data0, data1)  //put input only in sensitivity list
        if (select)  //select = 1
            muxout = data1;  //data1
        else
            muxout = data0;  //data0
endmodule

// Weighted 2421 to BCD Converter
module Weighted2421ToBCDConverter (
 input [3:0] weighted2421, // Input in weighted 2421 format
 output reg [3:0] bcd ); // output in BCD format

    always @ (weighted2421)    //put input only in sesitivity list
       case (weighted2421)
            4'b0000: bcd = 4'b0000; // 0
            4'b0001: bcd = 4'b0001; // 1
            4'b0010: bcd = 4'b0010; // 2
            4'b0011: bcd = 4'b0011; // 3
            4'b0100: bcd = 4'b0100; // 4
            4'b1011: bcd = 4'b0101; // 5
            4'b1100: bcd = 4'b0110; // 6
```

```
137                 4'b1101: bcd = 4'b0111; // 7
138                 4'b1110: bcd = 4'b1000; // 8
139                 4'b1111: bcd = 4'b1001; // 9
140                 default: bcd = 4'b1111; // Default to unique value so invalid input code can be
    recognized
141             endcase
142     // Alternative description is given below
143     //assign bcd = (weighted2421 < 5) ? weighted2421 : weighted2421 - 6;
144     endmodule
145
```