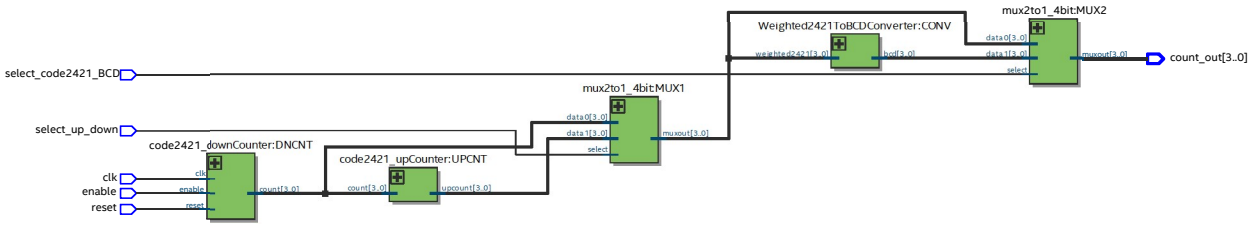
```
1   //ee417 lesson 4 Assignment 3 Part 2, L4A3P2
2   // Name: Lamin Jammeh, Date: 06-09-24  Group: Kalin/Jammeh
3   //top module BCD_or_2421_up_down_counter
4   module BCD_or_2421_up_down_counter(output [3:0] count_out,
5                                      input clk,
6                                      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;
```

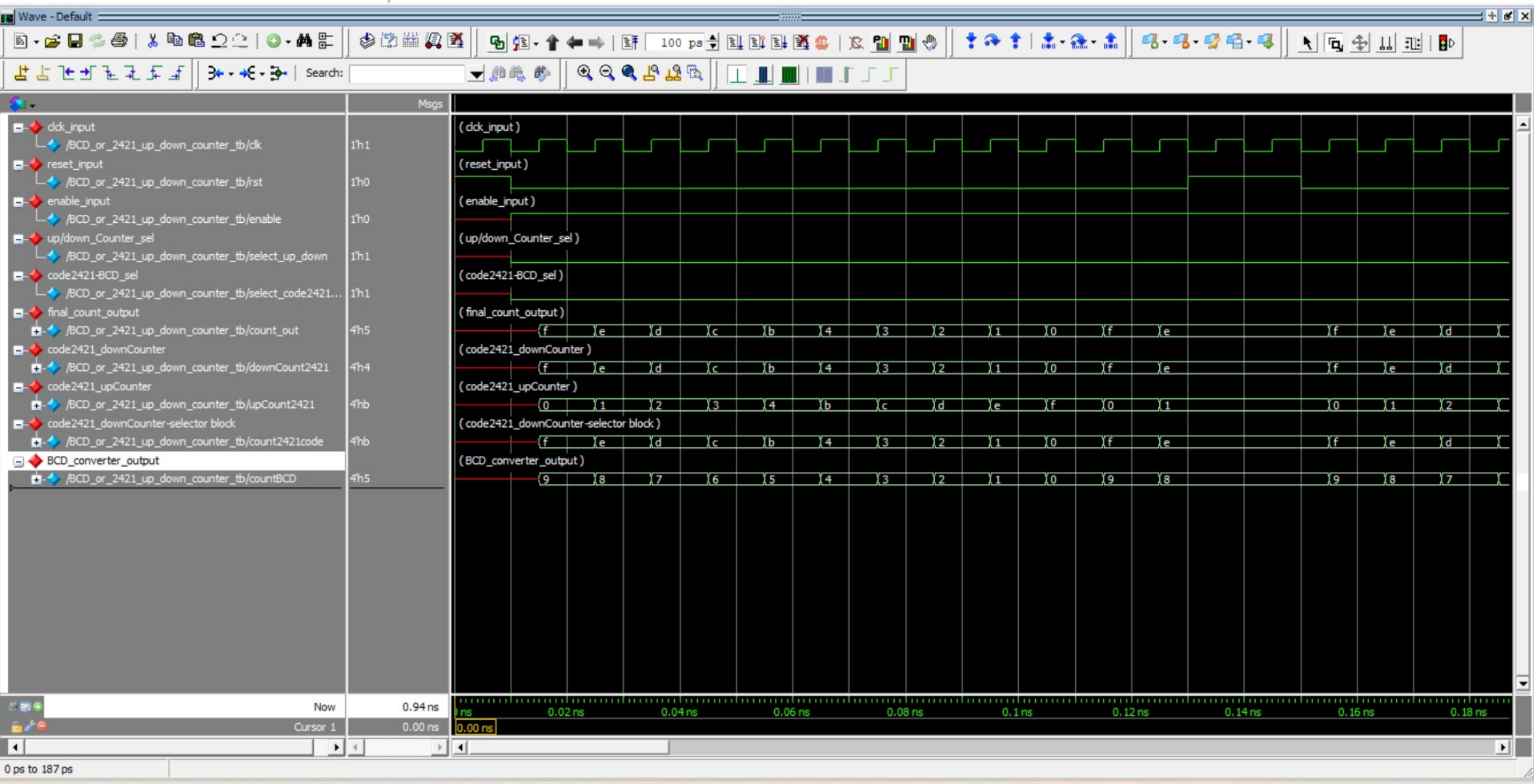
```
            count=4'b0011;
          end
        7: begin //state 7
            next_state =8;
            count=4'b0010;
          end
        8: begin //state 8
            next_state =9;
            count=4'b0001;
          end
        9: begin //state 9
            next_state =0;
            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
```


Weighted2421ToBCDConverter:CONV
mux2to1_4bit:MUX2
mux2to1_4bit:MUX1
code2421_downCounter:DNCNT
code2421_upCounter:UPCNT
select_code2421_BCD
select_up_down
clk
enable
reset
count_out[3..0]

```
1   //ee417 Lesson 4 Assignment 3 Part 2, L4A3P2
2   //Name: Lamin Jammeh, Date: 06-09-24, Group: Kalin/Jammeh
3   //Testbench for BCD_or_2421_up_down_counter using Moore FSM with case structure
4   //can select down or up counting for 2421code & also select (2421code or BCD) counting
5   //Step1 define test bench name
6   module BCD_or_2421_up_down_counter_tb();
7   /* original module declaration
8   module BCD_or_2421_up_down_counter(output [3:0] count_out,
9                                      input clk,
10                                     input reset,
11                                     input enable,
12                                     input select_up_down,
13                                     input select_code2421_BCD );*/
14  //Step2 define inputs as registers, outputs as wires
15  reg clk, rst,enable, select_up_down, select_code2421_BCD;
16  wire [3:0] count_out;
17  //internal probe wires: observe change in state..this gives Questa error if not enough
    digits
18  wire [3:0] downCount2421;
19  wire [3:0] upCount2421;
20  wire [3:0] count2421code;
21  wire [3:0] countBCD;
22  //wire [2:0] state, next_state;
23
24  //Step3 define unit under test
25  BCD_or_2421_up_down_counter UUT (count_out,clk,rst,enable,select_up_down,
26                                   select_code2421_BCD);
27
28  //internal probes to track logic and troubleshoot
29  assign downCount2421= UUT.downCount2421;
30  assign upCount2421= UUT.upCount2421;
31  assign count2421code= UUT.count2421code;
32  assign countBCD= UUT.countBCD;
33  //assign next_state= UUT.next_state;
34
35  //Step4 open initial block, define all possible input combinations
36  // Clock generation (adjust the period as needed)
37  initial begin
38    clk=0;
39    forever
40    #5 clk = ~clk;
41  end
42
43  initial  //reset is active high, longer time to count when reset is inactive (low)
44   begin   //4 cases with two selects
45    rst  = 1'b1; //reset on
46    # 10 rst = 1'b0; enable=1'b1; select_up_down=1'b0; select_code2421_BCD=1'b0;
47    //reset off, enable on, select down, select 2421
48    #120 rst = 1'b1;  //reset on
49    # 20 rst = 1'b0; enable=1'b1; //reset off, enable on
50    # 60 enable=1'b0;  //enable off: freeze count
51
52      # 10 rst = 1'b0; enable=1'b1; select_up_down=1'b0; select_code2421_BCD=1'b1;
53    //reset off, enable on, select down, select BCD
54    #120 rst = 1'b1;  //reset on
55    # 20 rst = 1'b0; enable=1'b1; //reset off, enable on
56    # 60 enable=1'b0;  //enable off: freeze count
57
58    # 10 rst = 1'b0; enable=1'b1; select_up_down=1'b1; select_code2421_BCD=1'b0;
59    //reset off, enable on, select up, select 2421
60    #120 rst = 1'b1;  //reset on
61    # 20 rst = 1'b0; enable=1'b1; //reset off, enable on
62    # 60 enable=1'b0;  //enable off: freeze count
63
64    # 10 rst = 1'b0; enable=1'b1; select_up_down=1'b1; select_code2421_BCD=1'b1;
65    //reset off, enable on, select up, select BCD
66    #120 rst = 1'b1;  //reset on
67    # 20 rst = 1'b0; enable=1'b1; //reset off, enable on
68    # 60 enable=1'b0;  //enable off: freeze count
```

```
69
70      #100 $stop; //close debug window to view waveform viewer
71     end
72
73    //Step5 Display the results
74    initial begin //monitor counter value
75      $display("_____output_count_out = -count-");
76      $monitor("clk_in = %b: rst_in = %b:  enable_in=%b: select_up_down=%b:
      select_code2421_BCD=%b: output_count_out = %d ",
77      clk, rst, enable, select_up_down, select_code2421_BCD, count_out);
78     end
79    endmodule
80
```


Wave - Default
Search:
Msgs
clck_input
/BCD_or_2421_up_down_counter_tb/clk
1'h1
reset_input
/BCD_or_2421_up_down_counter_tb/rst
1'h0
enable_input
/BCD_or_2421_up_down_counter_tb/enable
1'h0
up/down_Counter_sel
/BCD_or_2421_up_down_counter_tb/select_up_down
1'h1
code2421-BCD_sel
/BCD_or_2421_up_down_counter_tb/select_code2421...
1'h1
final_count_output
/BCD_or_2421_up_down_counter_tb/count_out
4'h5
code2421_downCounter
/BCD_or_2421_up_down_counter_tb/downCount2421
4'h4
code2421_upCounter
/BCD_or_2421_up_down_counter_tb/upCount2421
4'hb
code2421_downCounter-selector block
/BCD_or_2421_up_down_counter_tb/count2421code
4'hb
BCD_converter_output
/BCD_or_2421_up_down_counter_tb/countBCD
4'h5
(clck_input)
(reset_input)
(enable_input)
(up/down_Counter_sel)
(code2421-BCD_sel)
(final_count_output)
(code2421_downCounter)
(code2421_upCounter)
(code2421_downCounter-selector block)
(BCD_converter_output)
Now
0.94 ns
Cursor 1
0.00 ns
0.02 ns
0.04 ns
0.06 ns
0.08 ns
0.1 ns
0.12 ns
0.14 ns
0.16 ns
0.18 ns
0 ps to 187 ps

Part of the Simulation transcript

```
# ________________________________output_count_out = -count-
# clk_in = 0: rst_in = 1:  enable_in=x: select_up_down=x: select_code2421_BCD=x: output_count_out =  x
# clk_in = 1: rst_in = 1:  enable_in=x: select_up_down=x: select_code2421_BCD=x: output_count_out =  x
# clk_in = 0: rst_in = 0:  enable_in=1: select_up_down=0: select_code2421_BCD=0: output_count_out =  x
# clk_in = 1: rst_in = 0:  enable_in=1: select_up_down=0: select_code2421_BCD=0: output_count_out = 15
# clk_in = 0: rst_in = 0:  enable_in=1: select_up_down=0: select_code2421_BCD=0: output_count_out = 15
# clk_in = 1: rst_in = 0:  enable_in=1: select_up_down=0: select_code2421_BCD=0: output_count_out = 14
# clk_in = 0: rst_in = 0:  enable_in=1: select_up_down=0: select_code2421_BCD=0: output_count_out = 14
# clk_in = 1: rst_in = 0:  enable_in=1: select_up_down=0: select_code2421_BCD=0: output_count_out = 13
# clk_in = 0: rst_in = 0:  enable_in=1: select_up_down=0: select_code2421_BCD=0: output_count_out = 13
# clk_in = 1: rst_in = 0:  enable_in=1: select_up_down=0: select_code2421_BCD=0: output_count_out = 12
# clk_in = 0: rst_in = 0:  enable_in=1: select_up_down=0: select_code2421_BCD=0: output_count_out = 12
# clk_in = 1: rst_in = 0:  enable_in=1: select_up_down=0: select_code2421_BCD=0: output_count_out = 11
# clk_in = 0: rst_in = 0:  enable_in=1: select_up_down=0: select_code2421_BCD=0: output_count_out = 11
# clk_in = 1: rst_in = 0:  enable_in=1: select_up_down=0: select_code2421_BCD=0: output_count_out =  4
# clk_in = 0: rst_in = 0:  enable_in=1: select_up_down=0: select_code2421_BCD=0: output_count_out =  4
# clk_in = 1: rst_in = 0:  enable_in=1: select_up_down=0: select_code2421_BCD=0: output_count_out =  3
# clk_in = 0: rst_in = 0:  enable_in=1: select_up_down=0: select_code2421_BCD=0: output_count_out =  3
# clk_in = 1: rst_in = 0:  enable_in=1: select_up_down=0: select_code2421_BCD=0: output_count_out =  2
# clk_in = 0: rst_in = 0:  enable_in=1: select_up_down=0: select_code2421_BCD=0: output_count_out =  2
# clk_in = 1: rst_in = 0:  enable_in=1: select_up_down=0: select_code2421_BCD=0: output_count_out =  1
# clk_in = 0: rst_in = 0:  enable_in=1: select_up_down=0: select_code2421_BCD=0: output_count_out =  1
# ** Note: $stop    : C:/Users/lmnjm/OneDrive/Documents/Summer 2024/EE417 Programmable Logic Devices/Lecture 04/Quartu
#    Time: 940 ps  Iteration: 0  Instance: /BCD_or_2421_up_down_counter_tb
# Break in Module BCD_or_2421_up_down_counter_tb at C:/Users/lmnjm/OneDrive/Documents/Summer 2024/EE417 Programmable L
```

The simulation summary shows the following

- The system takes in a code2421 and counts it down in the first block
- In the second block the output sends to code2421 upCounter and a selector block
- Block three is the selector block 1 it takes input from the downCounter and the upCounter
    - It uses a select bit to determine which input to output
- The third block is a code2421 to BCD converter
    - Takes input from the selector or third block and outputs BCD
- The final block is another selector block
    - Takes input from the first selector block and code2421 to BCD converter block
    - Outputs either a BCD code or code2421 depending on a select bit