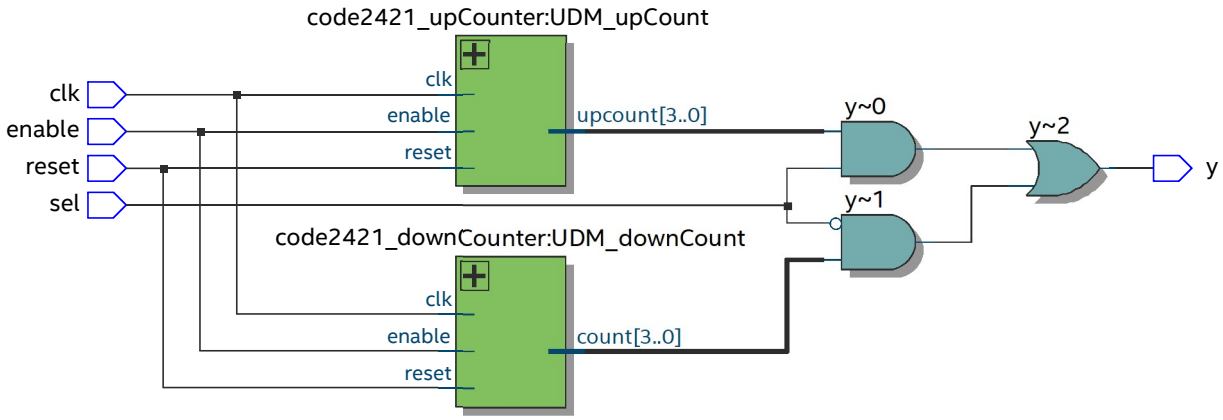
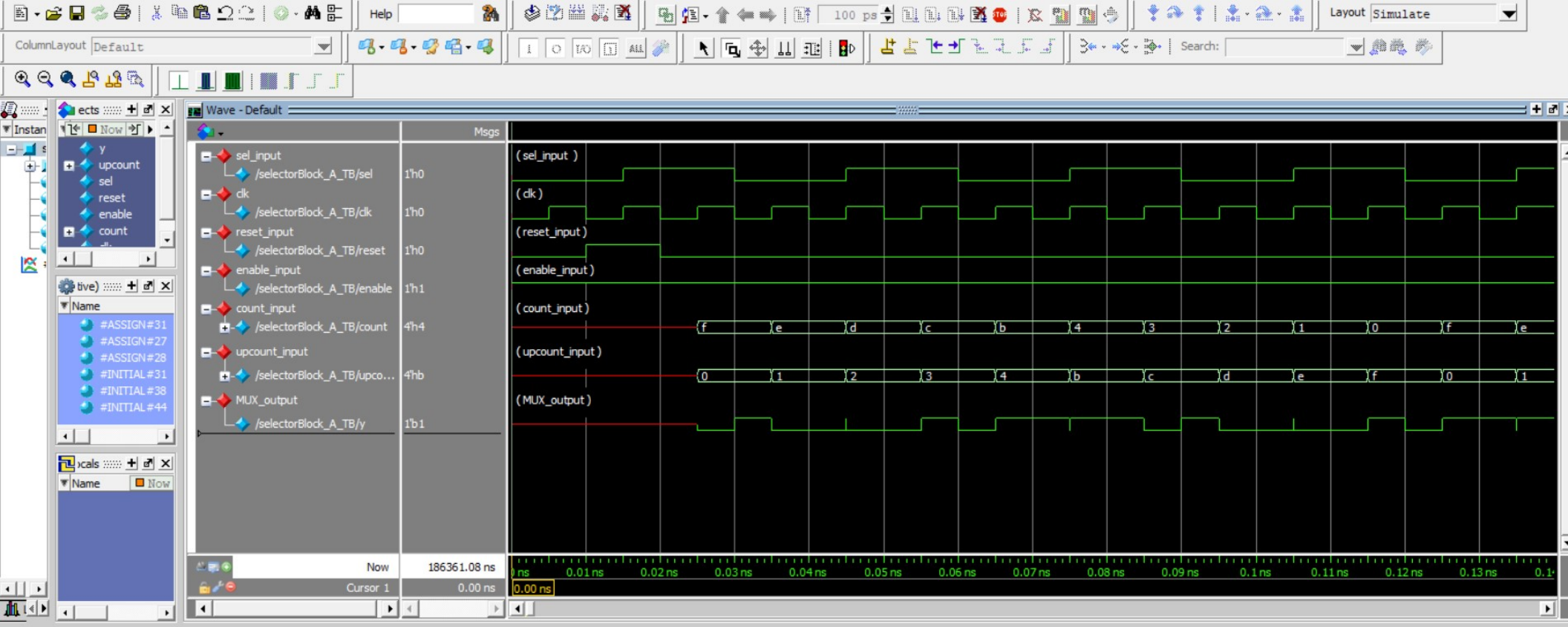
```
/*--------------------------------------------------------------------------------
Name Lamin Jammeh
CLass: EE417 Summer 2024
Lesson 04 HW Question 4 Part 3
Group: Ron Kalin/ Lamin Jammeh
Project Description: This portion takes uses a 2-to-1 MUX to select between up and down
counter. when the sel bit is high the upcounter output is transfered to the next stage
else the downcounter output is transfered to the next state
--------------------------------------------------------------------------------*/
//Step1: define the module and (port list)
   module selectorBlock_A (output wire y,
                           input sel,
                           input clk,
                           input reset,
                           input enable);

   wire [3:0] count;
   wire [3:0] upcount;

   //step 2 call the file for the code2421_downCounter
   code2421_downCounter UDM_downCount(
                                      .count(count),
                                      .clk(clk),
                                      .reset(reset),
                                      .enable(enable)
                                      );

   //call the file for the code2421_upCounter
   code2421_upCounter UDM_upCount(
                                      .upcount(upcount),
                                      .clk(clk),
                                      .reset(reset),
                                      .enable(enable)
                                      );
   //Instantiate the wire and logic gates
   //note when sel is high we count up and when low we count down
   //equation for output of mux Y = (In_1 & sel) | (In_2 & ~sel)
   assign y = (upcount & sel) | (count & ~sel);

   endmodule
```


code2421_upCounter:UDM_upCount
clk
enable
reset
sel
clk
enable
reset
upcount[3..0]
y~0
y~2
y
y~1
code2421_downCounter:UDM_downCount
clk
enable
reset
count[3..0]

```
1
2   module selectorBlock_A_TB ();
3
4       // Inputs
5       reg sel;
6       reg clk;
7       reg reset;
8       reg enable;
9
10      //declare internal probes as wire
11      wire [3:0] count;
12      wire [3:0] upcount;
13
14      // Outputs
15      wire y;
16
17      // Instantiate the selectorBlock_A module
18      selectorBlock_A UUT (
19                           .y(y),
20                           .sel(sel),
21                           .clk(clk),
22                           .reset(reset),
23                           .enable(enable)
24                          );
25
26      //assign internal probe to count and upcount
27     assign count = UUT.count;
28     assign upcount = UUT.upcount;
29
30      //step4 Initialize clock
31  initial
32     begin
33          clk = 0;
34          forever #5 clk = ~clk; // 10ns period
35      end
36
37      //step5 initialize the sel bit for the test purpose
38      initial
39     begin
40          sel = 0;
41          forever #15 sel = ~sel; // 30ns period
42      end
43   // Initialize signals
44  initial
45     begin
46          clk = 0;
47          reset = 0;
48          enable = 1; // Enable the counter
49
50          // Apply reset
51          #10 reset = 1;
52          #10 reset = 0;
53
54          // Simulate some clock cycles
55          #50;
56
57          // Display output y
58          $display("y = %b", y);
59      end
60
61  endmodule
62
```


Wave - Default
Msgs
sel_input
/selectorBlock_A_TB/sel
1'h0
clk
/selectorBlock_A_TB/clk
1'h0
reset_input
/selectorBlock_A_TB/reset
1'h0
enable_input
/selectorBlock_A_TB/enable
1'h1
count_input
/selectorBlock_A_TB/count
4'h4
upcount_input
/selectorBlock_A_TB/upco...
4'hb
MUX_output
/selectorBlock_A_TB/y
1'b1
(sel_input)
(clk)
(reset_input)
(enable_input)
(count_input)
(upcount_input)
(MUX_output)
f e d c b 4 3 2 1 0 f e
0 1 2 3 4 b c d e f 0 1
Now
186361.08 ns
Cursor 1
0.00 ns
0.01 ns 0.02 ns 0.03 ns 0.04 ns 0.05 ns 0.06 ns 0.07 ns 0.08 ns 0.09 ns 0.1 ns 0.11 ns 0.12 ns 0.13 ns
Transcript
.main_pane.structure.interior.cs.body.struct
view signals
.main_pane.objects.interior.cs.body.tree
run -all
y = 1