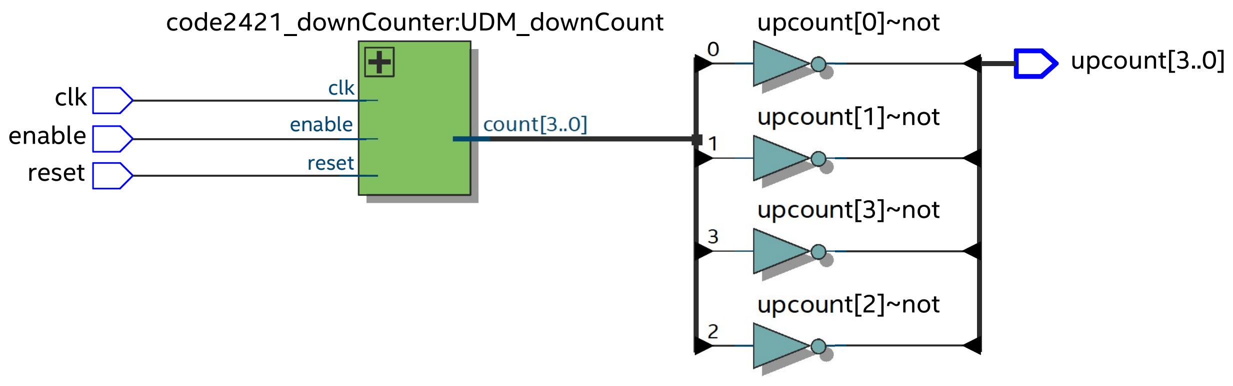
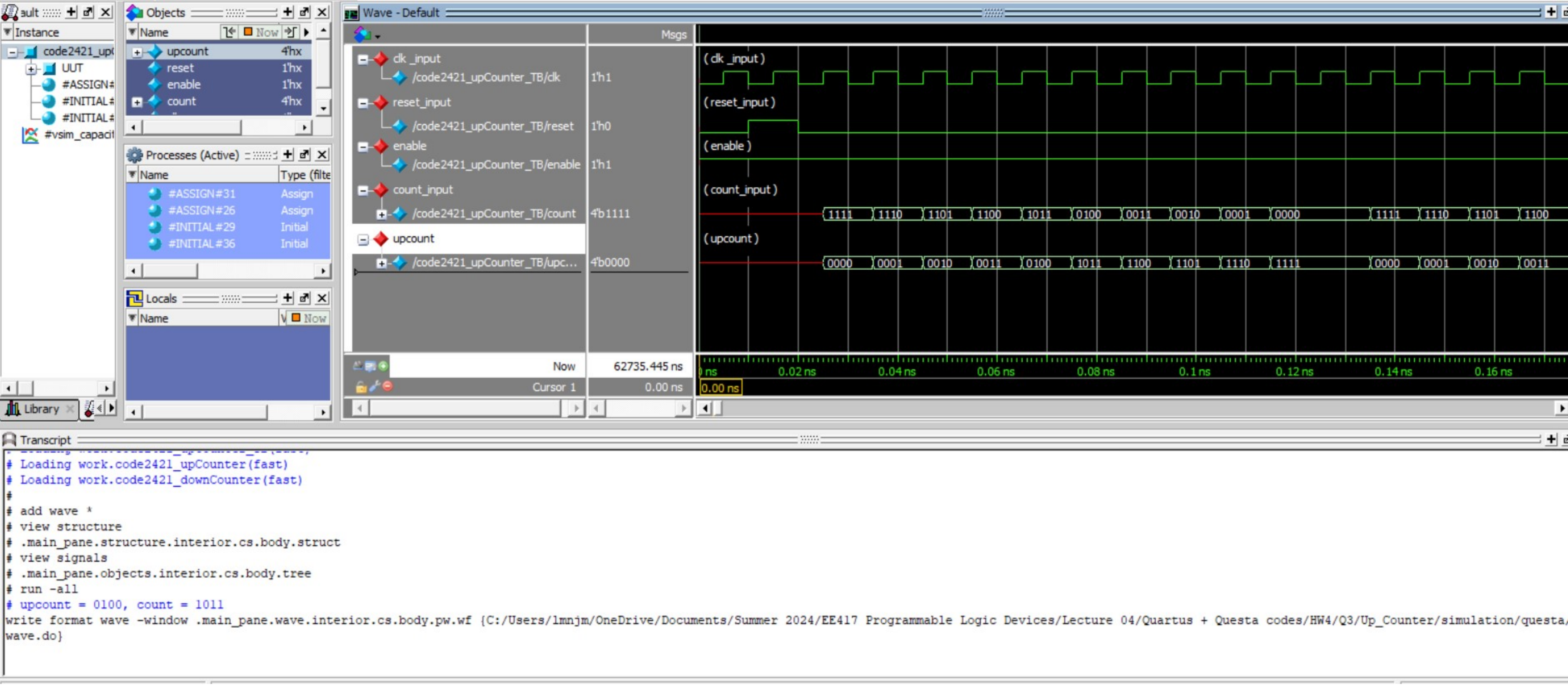
```
/*------------------------------------------------------------------------------------
Name Lamin Jammeh
CLass: EE417 Summer 2024
Lesson 04 HW Question 4 Part 2
Group: Ron Kalin/ Lamin Jammeh
Project Description: This portion takes use the output of the code2421_downCounter
and inverts it to form 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 wire [3:0] upcount,
                           input clk,
                           input reset,
                           input enable
                           );

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
```

```
/*--------------------------------------------------------------
Test Bench for code2421_upCounter
CLass: EE417 Summer 2024
Lesson 04 HW Question 3 part 2
Group: Ron Kalin/ Lamin Jammeh
---------------------------------------------------------------*/
//Step1 define the test bench
module code2421_upCounter_TB ();

//step2 define define the inputs and outputs of the testbench
reg          clk, reset, enable;
wire   [3:0] count;
wire   [3:0] upcount;



 //step3 Instantiate the code2421_upCounter module
 code2421_upCounter UUT (
                        .upcount(upcount),
                        .clk(clk),
                        .reset(reset),
                        .enable(enable)
                        );

//assign internal probe to count
assign count = UUT.count;

//step4 Initialize clock
initial
   begin
        clk = 0;
        forever #5 clk = ~clk; // 10ns period
    end

 // Initialize signals
initial
   begin
        clk = 0;
        reset = 0;
        enable = 1; // Enable the counter

        // Apply reset
        #10 reset = 1;
        #10 reset = 0;

        // Simulate some clock cycles
        #50;

        // Display both upcount and count values
        $display("upcount = %b, count = %b", upcount, count);
    end

endmodule

```


code2421_downCounter:UDM_downCount
clk
enable
reset
clk
enable
reset
count[3..0]
0
1
3
2
upcount[0]~not
upcount[1]~not
upcount[3]~not
upcount[2]~not
upcount[3..0]



```
# Loading work.code2421_upCounter(fast)
# Loading work.code2421_downCounter(fast)
#
# add wave *
# view structure
# .main_pane.structure.interior.cs.body.struct
# view signals
# .main_pane.objects.interior.cs.body.tree
# run -all
# upcount = 0100, count = 1011
write format wave -window .main_pane.wave.interior.cs.body.pw.wf {C:/Users/lmnjm/OneDrive/Documents/Summer 2024/EE417 Programmable Logic Devices/Lecture 04/Quartus + Questa codes/HW4/Q3/Up_Counter/simulation/questa/
wave.do}
```

The simulation summary shows the following

- The code2421_upCounter takes count as an input
    - Count is the output of the code2421_downCounter
- The input count in this case is count upward using the complementary property of code 2421
- Therefor the upcount = ~count