

**This lab assignment is to be completed individually. You are allowed to use course materials, the internet, and ask the instructor or teaching assistant for limited assistance. If you give or receive assistance you will receive a grade of 0 for the assignment.**

**You have one week(s) to complete this lab. It is your responsibility to manage your time in such a way that your lab is complete before the due date without requiring last minute instructor assistance or checkoff. While instructor will attempt to assist and check off the lab at the last minute, there is no guarantee. You must demo this lab and submit the lab report before due date.**

## 1 Introduction

The purpose of this lab is to learn how to create a simple testbench in ModelSim. Recall from lecture, a testbench file is just another SystemVerilog or Verilog file, with no inputs or outputs.

```
module counter4_b; //No I/O in testbench

    counter4 il(clock , reset , updown, counter );

endmodule
```

### 1.1 Time units and initial blocks

You can use an initial block along with # to define what happens at specific points in time. Initial blocks execute sequentially and only run once. Below, updown is set to 1 after 1 time unit, and then after another time unit set to 0. Note that you must include the timescale directive or write the time unit such as #1ns

```
initial begin
    #1 updown = 1;
    #1 updown = 0;
end
```

### 1.2 Clock generation

You can generate a clock by using the following statement. The following statement changes the clock edge every 5 time units.

```
always #5 clk = ~clk;
```

You can also use the following:

```
intial begin
    forever begin
        #5 clk = ~clk;
    end
end
```

## 1.3 Looping constructs

Use while() or for() for repetitive tasks.

## 1.4 Printing to console

You can print to console by using \$display or \$monitor system tasks. Recall display displays once, but monitor will update the console every time the variable changes.

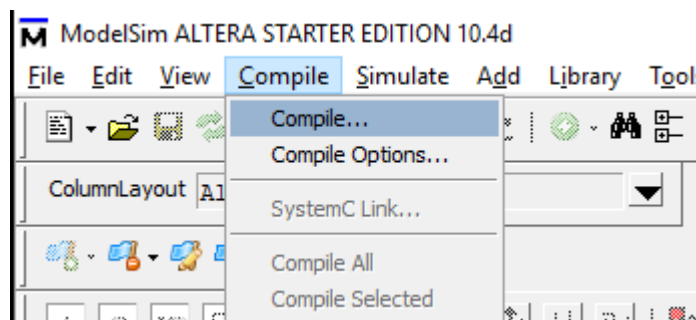
```
$display ("Hello World"); //Displays Hello World to console once.  
$display ("The value is = %d", dat1); //Displays the value of dat1 once.  
$monitor ("%b, $b", var1, var2); //Updates the console every time var1 or var2 changes.
```

## 1.5 Stop vs finish

Stop suspends simulator. Finish closes simulator.

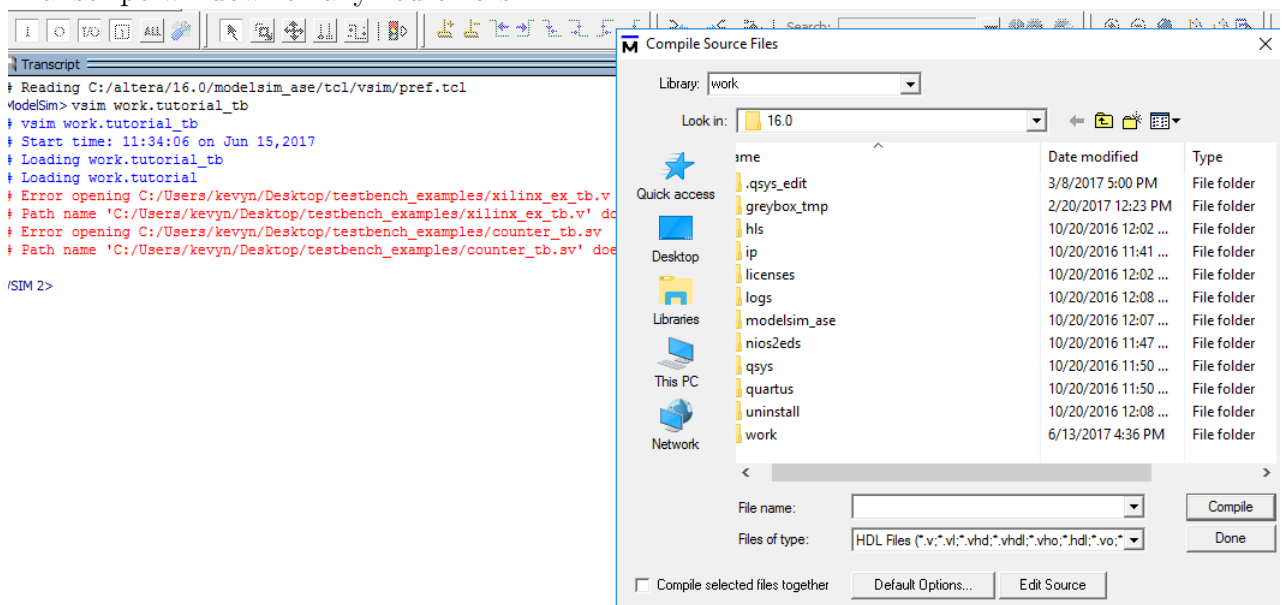
# 2 ModelSim user interface

1. Open up ModelSim.

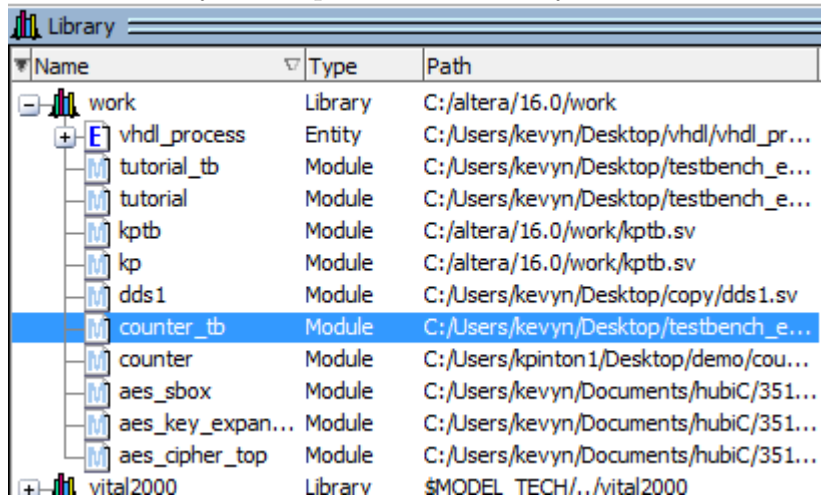


2. Click Compile in ModelSim.

3. You will need to compile the testbench and the module you are testing in ModelSim. Note that sometimes it will compile in Quartus, but will fail to compile in ModelSim. Check the Transcript window for any red errors.

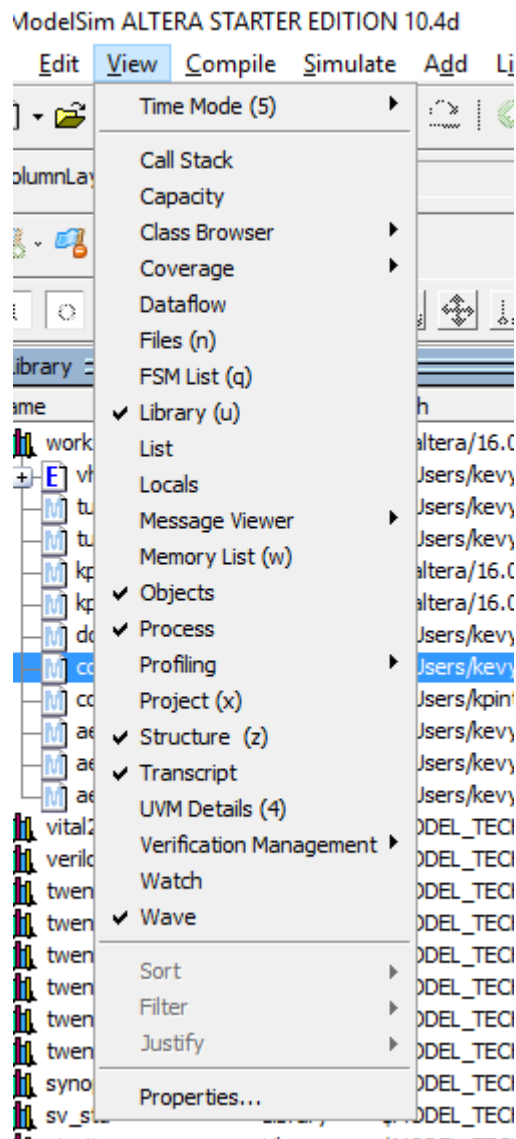


4. Successfully compiled work will show up in the ModelSim 'work' library. Double click the testbench file you compiled in the library to launch it.

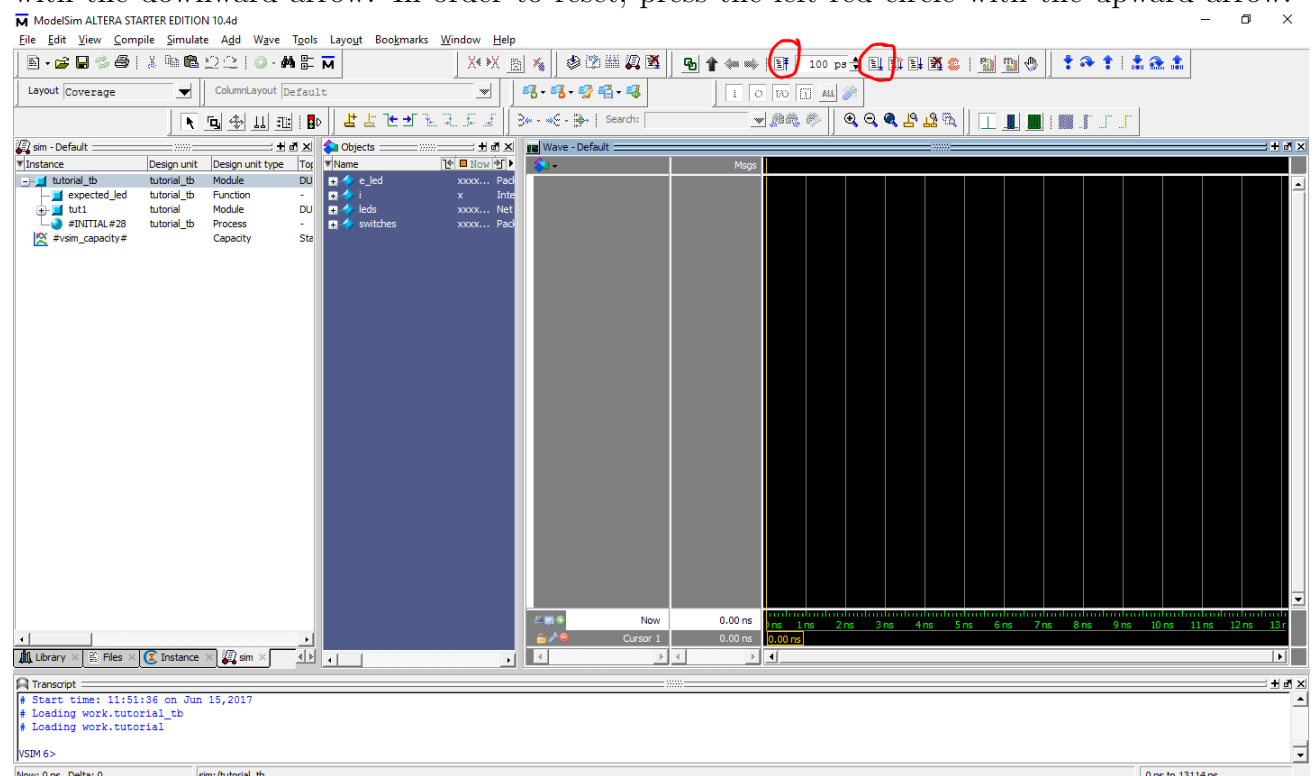


Name	Type	Path
work	Library	C:/altera/16.0/work
vhdl_process	Entity	C:/Users/kevyn/Desktop/vhdl/vhdl_pr...
tutorial_tb	Module	C:/Users/kevyn/Desktop/testbench_e...
tutorial	Module	C:/Users/kevyn/Desktop/testbench_e...
kptb	Module	C:/altera/16.0/work/kptb.sv
kp	Module	C:/altera/16.0/work/kptb.sv
dds1	Module	C:/Users/kevyn/Desktop/copy/dds1.sv
counter_tb	Module	C:/Users/kevyn/Desktop/testbench_e...
counter	Module	C:/Users/kpinton1/Desktop/demo/cou...
aes_sbox	Module	C:/Users/kevyn/Documents/hubiC/351...
aes_key_expan...	Module	C:/Users/kevyn/Documents/hubiC/351...
aes_cipher_top	Module	C:/Users/kevyn/Documents/hubiC/351...
vital2000	Library	\$MODEL_TECH/./vital2000

5. If you can't see a window you need, go to View and select the window. Make sure you have Library, Objects, Transcript, and Wave.
- Library is where your compiled files will show up.
  - Objects list the input and output and internal signals in your system.
  - Wave is the window which shows your simulation in time.
  - Transcript is where your display and monitor commands as well as other input and output will go.



6. Your setup should look like this. In order to run your testbench, press on the right red circle with the downward arrow. In order to reset, press the left red circle with the upward arrow.



## 3 Instructions

### 3.1 Task 1

Design a counter with the following specifications.

1. 5-bit counter (Counts from 0 to 31)
2. Synchronous reset
3. Up and down signal
4. enable signal

```
module counter5(  
input clock,  
input reset,  
input updown,  
output reg [4:0] counter  
);
```

```
//Insert your code here.
```

```
endmodule
```

### 3.2 Task 2

Create a testbench to test the code automatically, without user input. The testbench should test the reset, updown, enable signals. Do the following:

1. Make all required wires or registers to test your counter4 module.
2. Instantiate a copy of your counter in the testbench file.
3. Generate a clock signal.
4. Toggle reset and leave it in the enabled position.
5. set updown to count up (however you configured it in code.)
6. Count from 0 to 31.
7. Set updown to count down.
8. Count from 31 to 0.
9. Print out the count value in the console window. You can use the monitor system task.

### 3.3 Task 3- Decrypt message using provided code and testbench

Use the project provided from Canvas website and create a testbench which will test the AES 128 decryption module with the inputs on the course website. You can just take the files needed and also use them directly in ModelSim by creating a ModelSim project Pick one of the files (e1-e15) and contact me for the key. This process must be performed using the testbench and return each of the decrypted 128 bits automatically in ModelSim. Print the decrypted values for all decryptions in the ModelSim window.

Provide the following:

1. Decryption Testbench code
2. Decryption Testbench output txt file (decrypted file)
3. Decryption Modelsim window screenshot

## 4 Submission

Electronically submit the following. Note that there is no lab writeup required.

1. Testbench for counter (Submit separate file)
2. Simulation screenshot for counter showing correct counter operation (reset, up, down ,etc.)
3. Your console window screenshot for counter in text file.
4. Answer this question: If we had a 1024 bit counter and limited testing time, what strategies would you use to verify the counter is functional without testing every single value?
5. Decryption Testbench code.
6. Decryption Testbench output txt file
7. Decryption Modelsim window screenshot