DC Tutorial (using the GUI)

1. First, open your Linux account and type *addpkg* in order to install necessary software:

```
[ ] java12          OpenJDK Java 12 Development Kit (12.0.1)
[ ] java7           Java SE Development Kit 7 (1.7.0)
[ ] java8           Java SE Development Kit 8 (1.8.0)
[ ] maple           Maple - Maplesoft Maple Software (v16) HIDDEN
[ ] matlab          MATLAB (2012b)
[ ] mentor-ams      Mentor Questa ADMS  (15.4.1)
[ ] mentor-calibre  Mentor Calibre (2013.2_35.25)
[ ] mentor-questa-20 Mentor Questa (2014 10.3)
    14
[X] mentor-questa-20 Mentor Questa (2014 10.4c)
    15
[ ] ngspice-26      Ngspice version (testing) (26)
[ ] nusmv           NuSMV (2.5.4))
[ ] pgi             Portland Group Compiler 2017 (17.4)
[ ] primetime       SYNOPSYS Primetime (G-2012.06)
[ ] pycharm         PyCharm (2017.3.3)
[ ] python3.5       Python Anaconda - 3.5 (4.0.0)
[ ] synapticad      SynaptiCAD software including Verilogger (16.04d)
[ ] synopsys-coretoo SYNOPSYS Core Tools (2017)
    ls2017
[X] synopsys-designc SYNOPSYS Design Compiler (G-2012.06)
    ompiler
[ ] synopsys-dftcomp SYNOPSYS DFT Compiler, with Tetramax Overlay (2016)
    iler
[ ] synopsys-formali SYNOPSYS Formality (2017)
    ty
[ ] synopsys-formali SYNOPSYS Formality ESP (2017)
    ty_esp
[ ] synopsys-icc    SYNOPSYS IC Compiler (2016)
[ ] synopsys-icc2013 SYNOPSYS IC Compiler (I-2013.12-SP1)
[ ] synopsys-icc2017 SYNOPSYS IC Compiler (2017)
[ ] synopsys-icv2017 SYNOPSYS IC Validator (2017)
```

For this project, you will need *synopsys-designcompiler* for doing synthesis and mentor-questa-2015 for simulation (if you already do not have this installed).

2. Next, create a new folder for your project. For my project, I created a separate folder for each part using *mkdir*. To see which files you have in your current directory, use *ls*. To go into your directory of choice, use *cd*. See the commands provided below:

```
baris@walle:~$ mkdir New_Student_ECE_581_Project_3
baris@walle:~$ ls
Baris_Inan_ECE_581_Project_3              Documents
Baris_Inan_ECE_581_Project_3_Part2        Downloads
Baris_Inan_ECE_581_Project_3_Part2_Reverse New_Student_ECE_581_Project_3
Baris_Inan_ECE_581_Project_3_Part3        alib-52
Desktop                                   bin_to_gray.ddc
baris@walle:~$ cd New_Student_ECE_581_Project_3/
```

3. To write your code files, you can use *vim filename.sv*. If you are using a remote computer, you can drag your files and folders into *MobaXterm*, shown below:
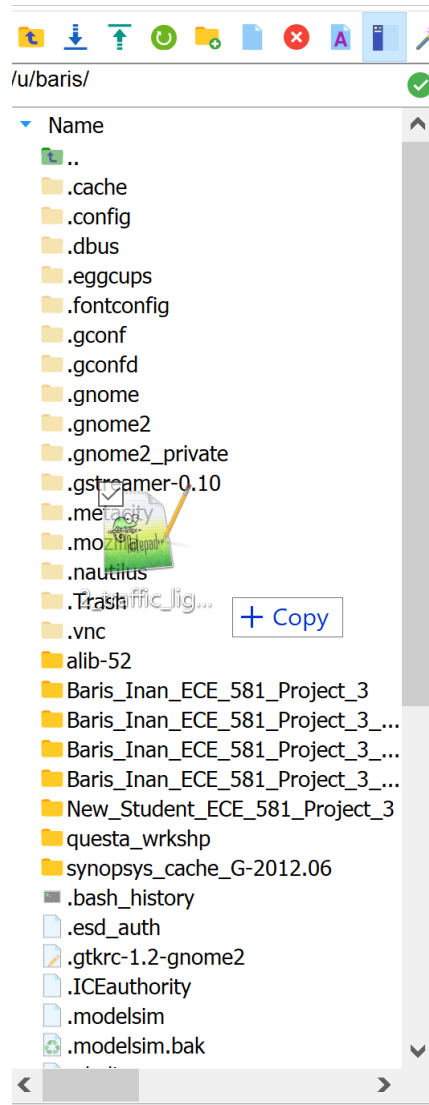
4. Next, a .tcl file can be created that contains the commands a user needs to run to do synthesis. I provide the contents of my .tcl file below:

```
lappend search_path /u/baris/Baris_Inan_ECE_581_Project_3
set target_library osu05_stdcells.db
set link_library [concat "*" $target_library]
link

read_file -format sverilog gray_adder.sv
current_design gray_adder
compile
report_area
report_cell
report_power
write -format Verilog -hierarchy -output gray_adder.netlist
link
```

5. Before doing DC, make sure you have both *osu05_stdcells.db* and *osu05_stdcells.v* in your folder. These can be found at *d2l.pdx.edu* under course name ECE581 and then Reading and EDA Tools. The first file will be used for DC. The second file will be used for simulation purposes.

osu05_stdcells
DB File

osu05_stdcells
V File
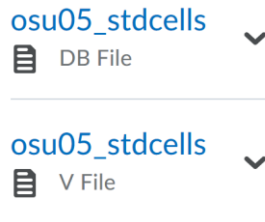
6. This tutorial uses the GUI. Type *design_vision* into the terminal prompt to launch the GUI. Make sure you are using the correct libraries. To do this, go to the *File* menu and select *Setup…*. Select the appropriate libraries from your New_Student_ECE_581_Project_3 folder. For the Link library and Target library, select *osu05_stdcells.db*.
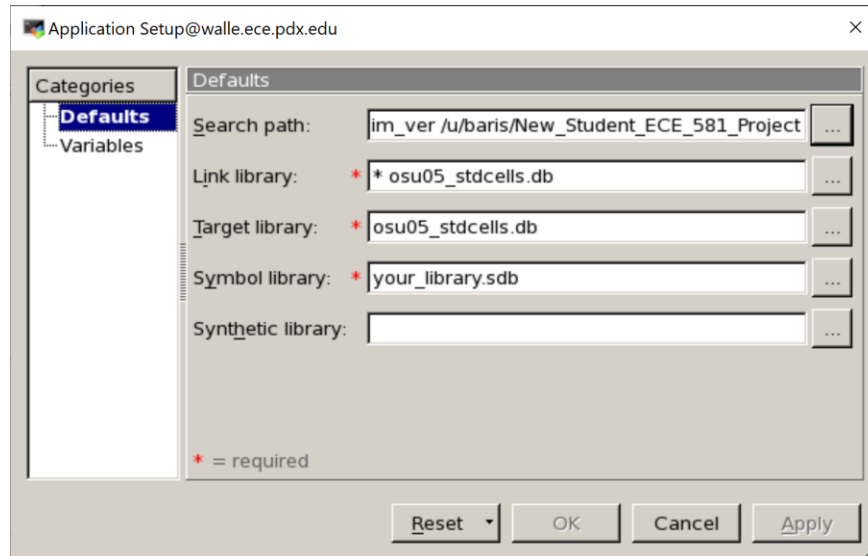

7. After selecting your Link library and Target library, type in the command shown below into the GUI.

8. Note that if you receive an "Error: Current design is not defined. (UID-4)" error in the output window of the GUI, you can just ignore it because it is resolved later.

9. At this point, the user can enter *report_cell*, *report_area* and *report_power* into the GUI to receive information about the complexity (gate count) using cell, area and power.

10. In addition, the GUI allows the user to save their results. First, go to the GUI and navigate to the *Design* menu. Select *Report cells…* and then check *To file:* and provide a name for your file. The default name provided is Report.txt. The same can be done with *Report area…* and *Report power….* In my case, I included all of these lines in the .tcl file so that these results appear in the transcript file generated by *source compile_gray_adder.tcl* in Step 7 and for this reason, this step is not necessary. For more details on the .tcl file see Step 4.
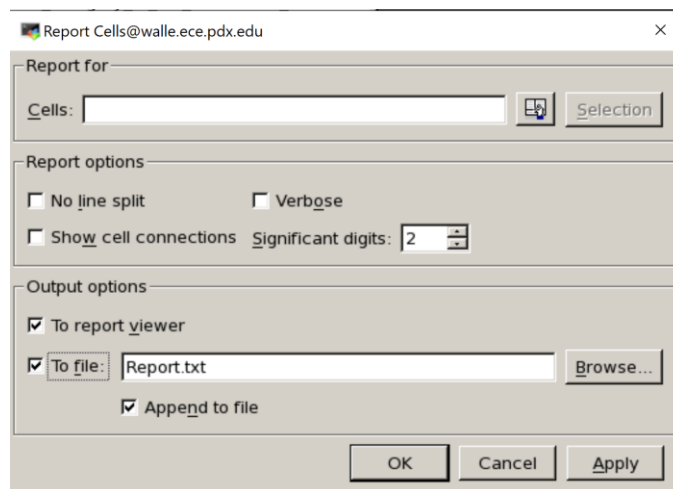
11. Now, to view your netlist, type *vim name_of_netlist.netlist*. In this example, my netlist is called *gray_adder.netlist*. You should see a netlist like the following:

```
module gray_to_bin_0 ( b, g );
  output [3:0] b;
  input [3:0] g;
  wire   \g[3] ;
  assign b[3] = \g[3] ;
  assign \g[3]  = g[3];

  XOR2X1 U1 ( .A(g[0]), .B(b[1]), .Y(b[0]) );
  XOR2X1 U2 ( .A(b[2]), .B(g[1]), .Y(b[1]) );
  XOR2X1 U3 ( .A(g[2]), .B(\g[3] ), .Y(b[2]) );
endmodule


module gray_to_bin_1 ( b, g );
  output [3:0] b;
  input [3:0] g;
  wire   \g[3] ;
  assign b[3] = \g[3] ;
  assign \g[3]  = g[3];

  XOR2X1 U1 ( .A(g[0]), .B(b[1]), .Y(b[0]) );
  XOR2X1 U2 ( .A(b[2]), .B(g[1]), .Y(b[1]) );
  XOR2X1 U3 ( .A(g[2]), .B(\g[3] ), .Y(b[2]) );
endmodule
```
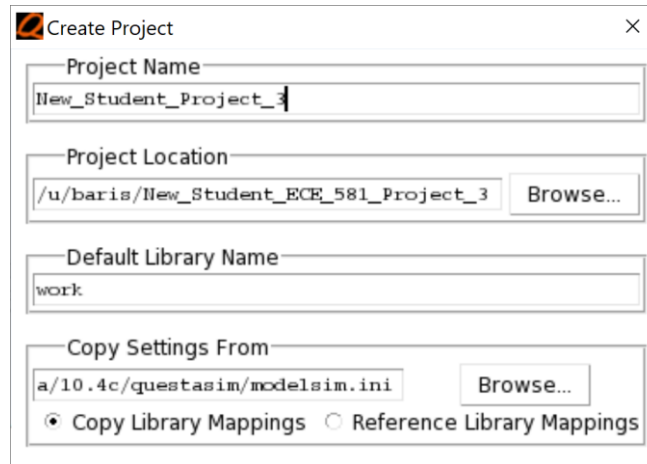
Note that if you are having problems generating a netlist, it could be because you used constructs that are part of SystemVerilog's syntax but cannot be synthesized by the synthesis tool. One example of this case is variable parameters. For example, I had to change [N-1:0] to [4:0] and [3:0] for the graycode to binary code and binary code to graycode converters respectively.

12. Now we are ready for simulation. Enter *vsim* into the terminal in order to open QuestaSim. Make sure QuestaSim is installed (See Step 1).

13. After opening QuestaSim, go to *File* then *New* then *Project*.

Questa Sim-64 10.4c

File Edit View Compile Simulate Add **Project** Tools

New	▸	Directory
Open...		Source ▸
Load	▸	Project...
Close Project		Library...
Import	▸	Debug Archive...
Export	▸	Results Analysis Database...
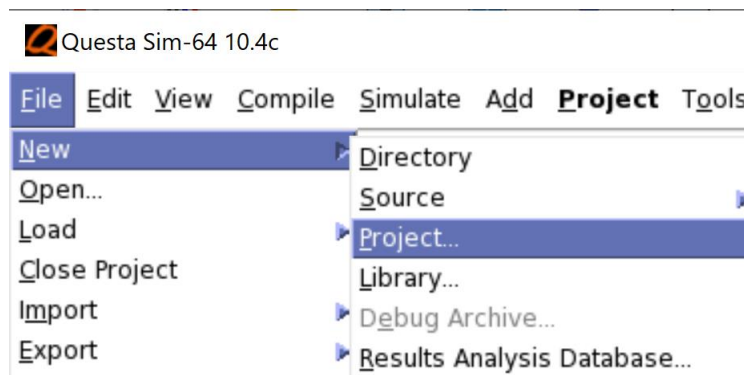
14. Change *Project Location* to the folder where you created the project. Afterwards, name
your project and hit Enter.

15. Click on *Add Existing File* and add your netlist, testbench and osu05_stdcells.v one by one.
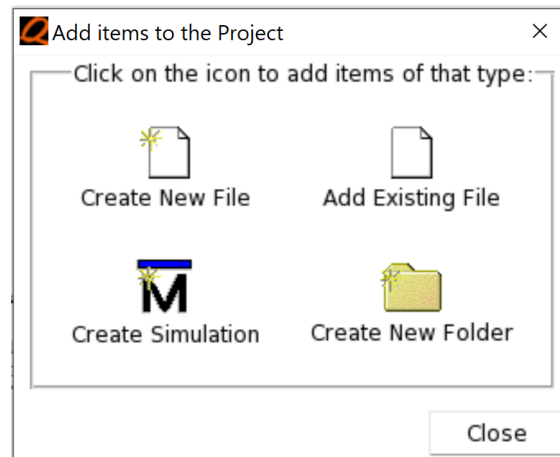

16. Hit *Close* and go to *Compile* and select *Compile All*. All three of your files should have green checkmarks beside them. This means that all of them have compiled.

17. Go to the *Simulate* menu and select *Start Simulation…*. Open *Work* and select your testbench. In my case, this is *gray_adder_tb*.
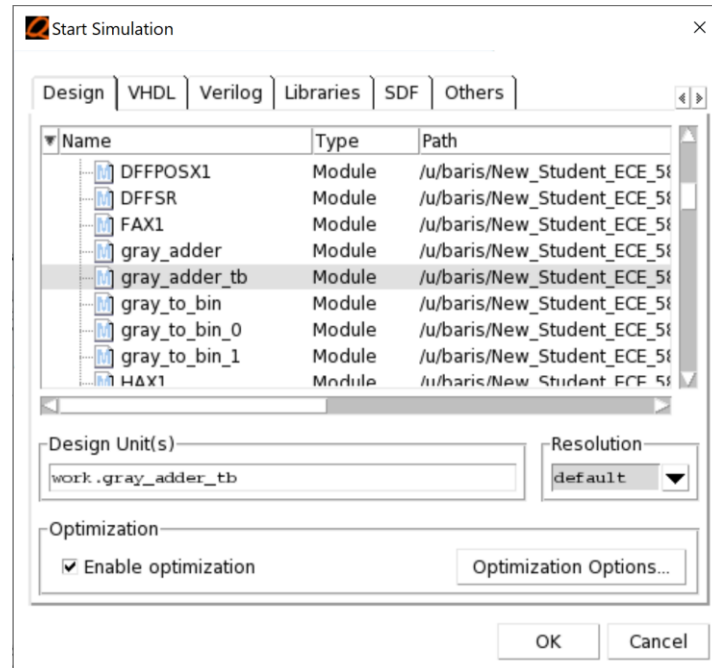

18. Now select *Optimization Options* and select *Apply full visibility to all modules(full debug mode)*. Next, select the *Options* menu. Select *Disable Optimizations-(O0)* and uncheck the *Disable Timing Checks(+notimingcheck)*.
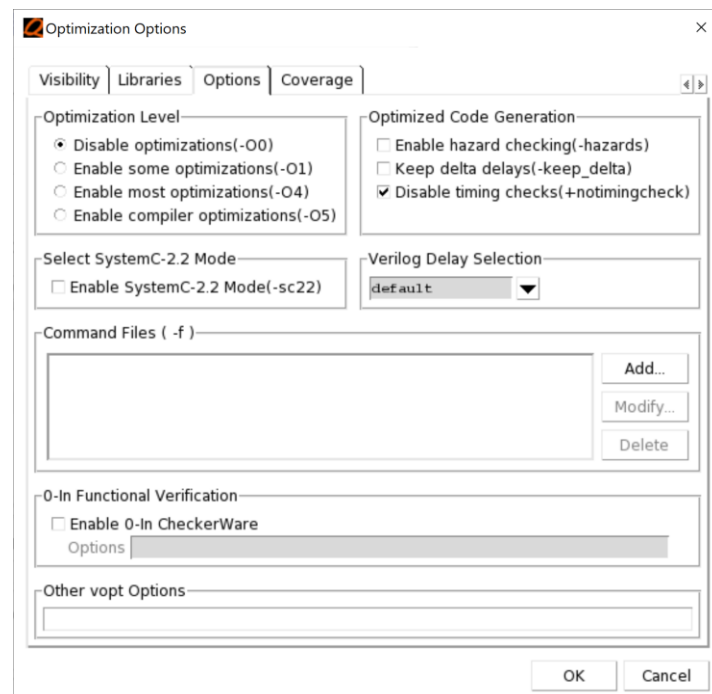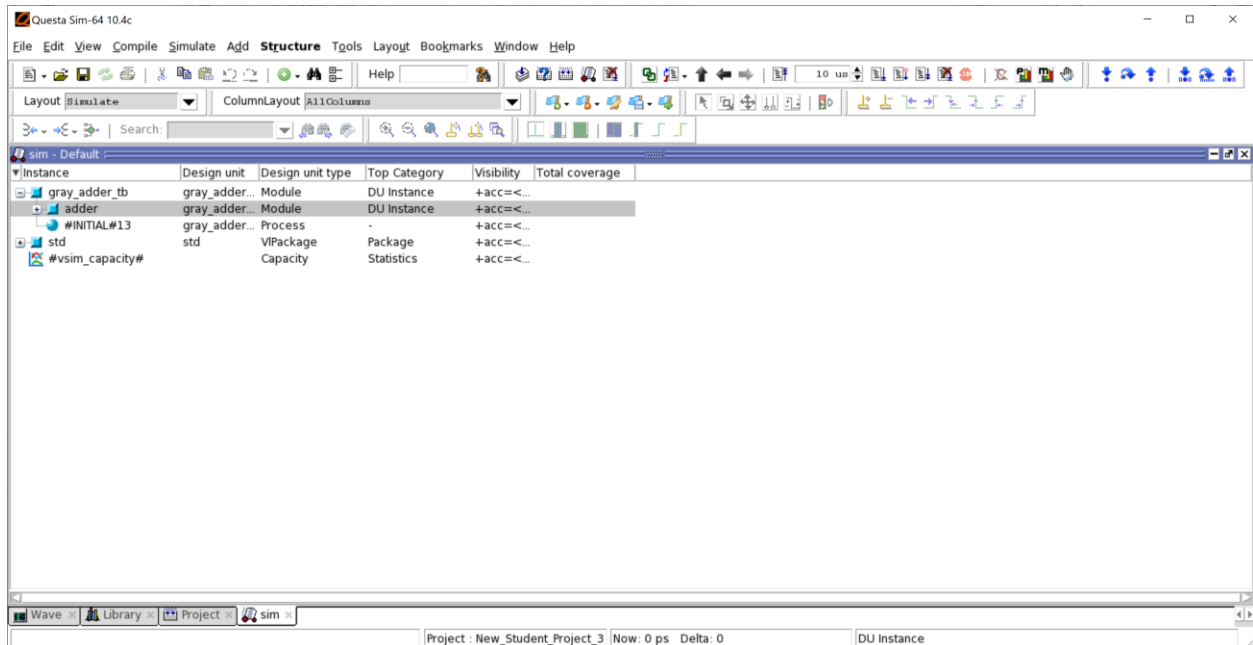
19. Click OK. Then type *run -all* to generate your output. It should look like the example below:
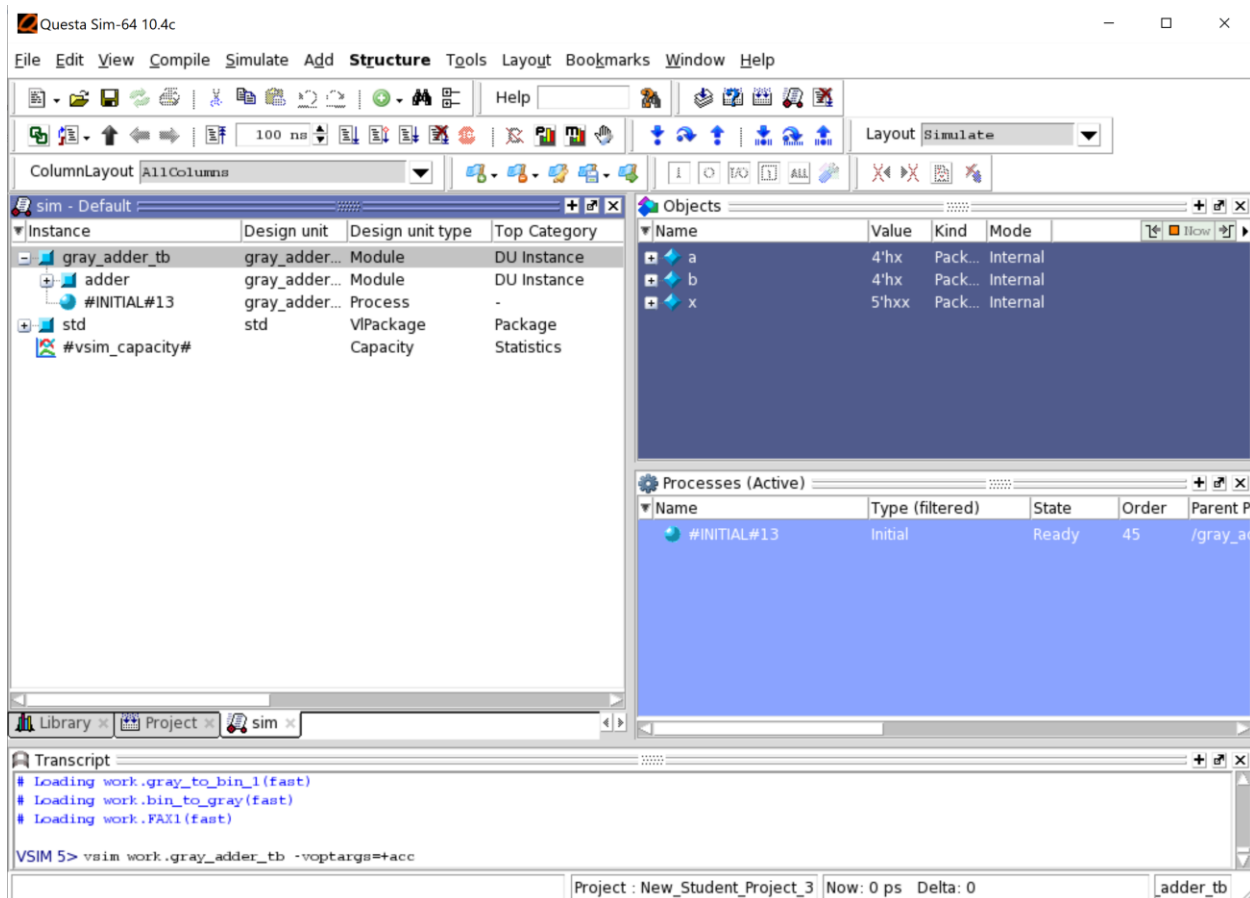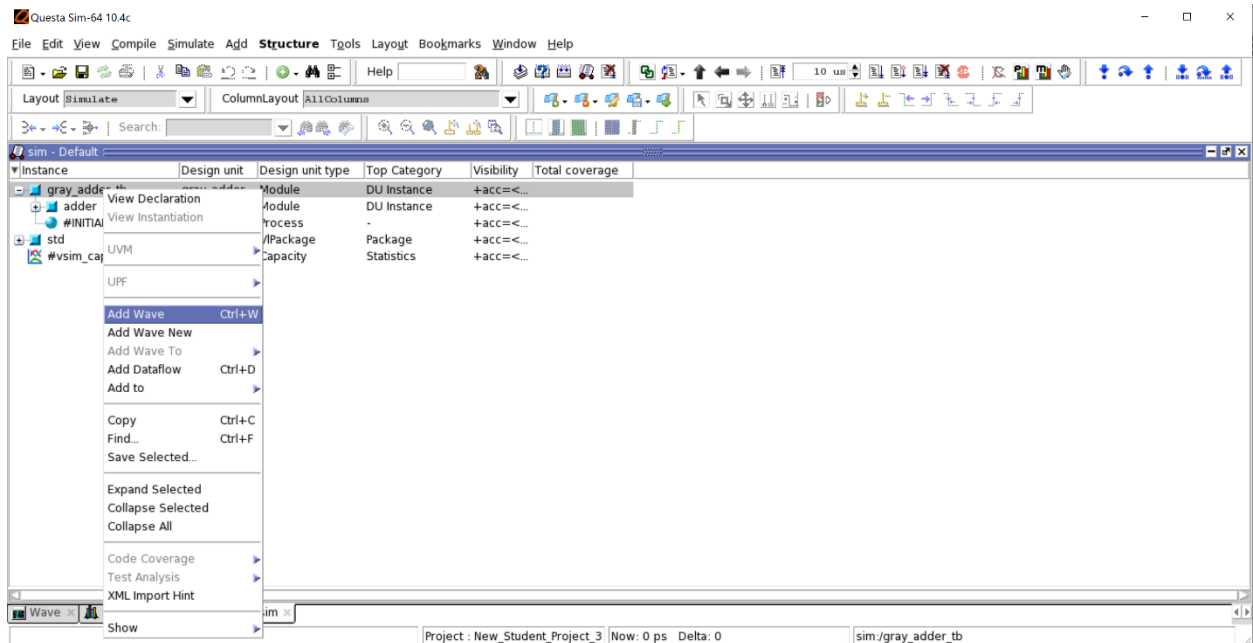

20. QuestaSim may prompt you asking if you want to finish as shown below. Press No.
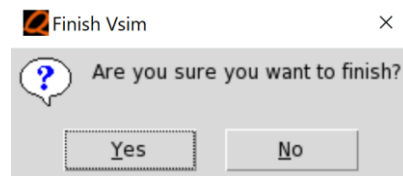
21. Click on the *sim* tab. The QuestaSim simulation window should look like the figure below.


22. To open the waveform window, right click on *gray_adder_tb* and select *Add Wave* or press *Ctrl+W*.

23. This step will bring you to the window where you can view your waves. However, at this point, you can only see the variables a, b and x and not the actual waveforms. To address this issue press the *Run -All* button.
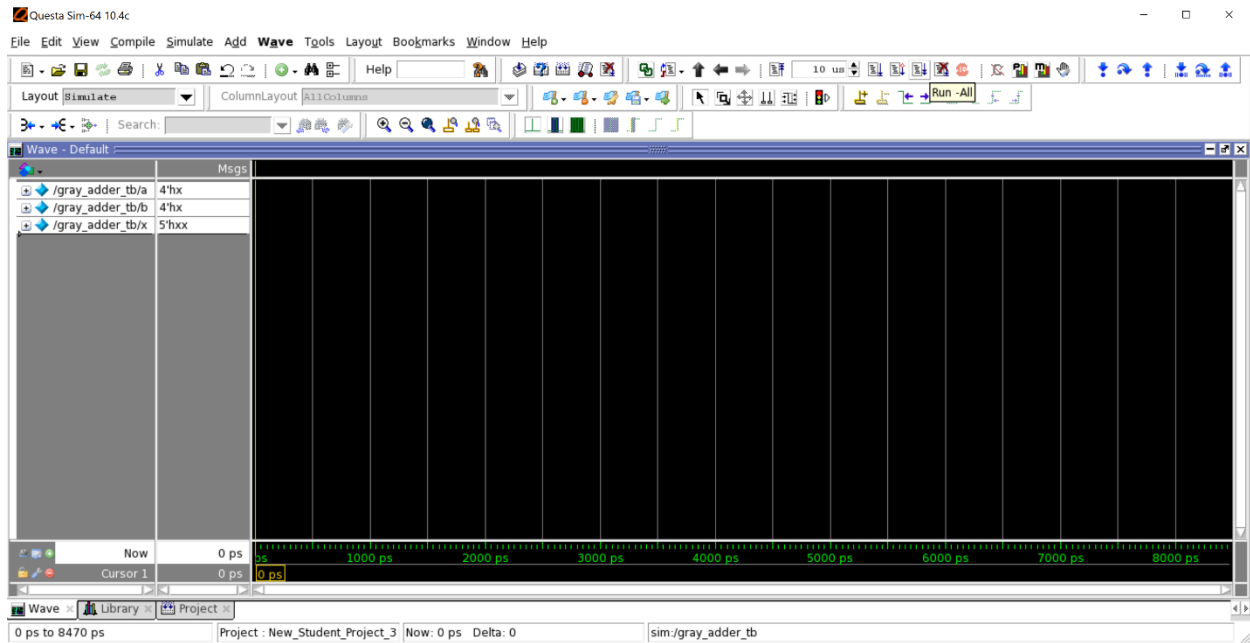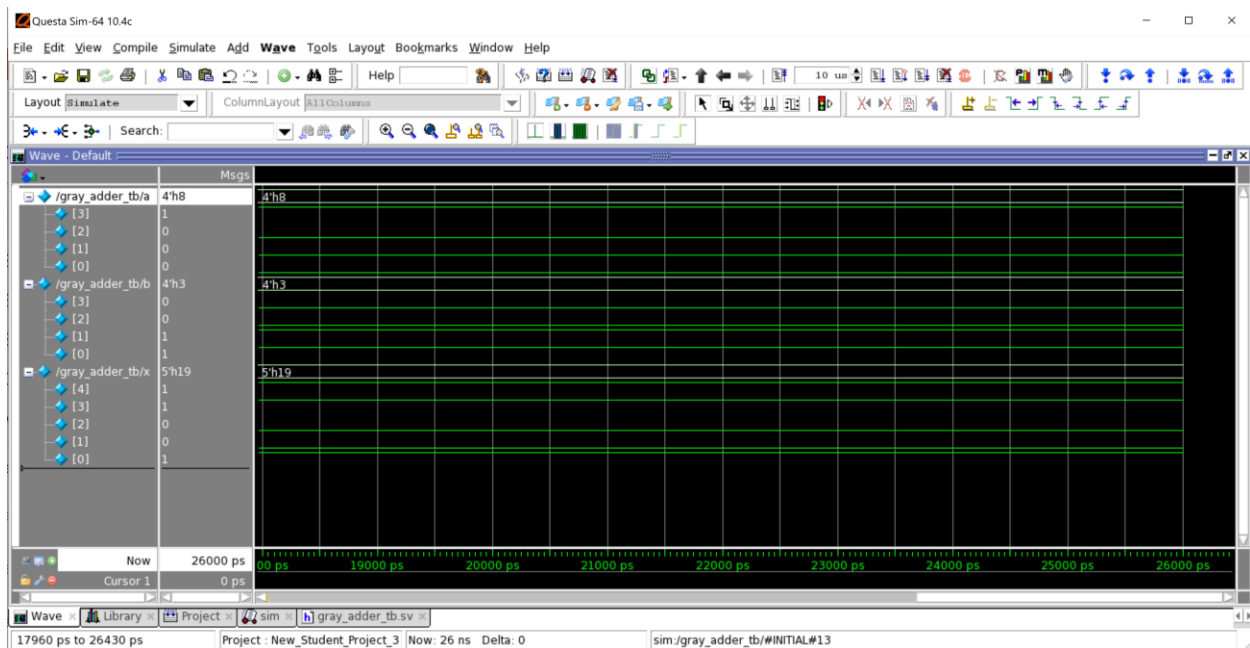

24. QuestaSim should prompt you and ask you if you are sure you want to finish. Press No.
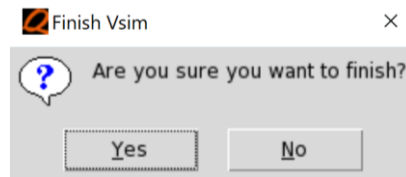
25. The waveforms for the three signals a, b and x should appear as shown in the image below. Click on the plus (+) sign on these signals to expand them and look at their individual bits.
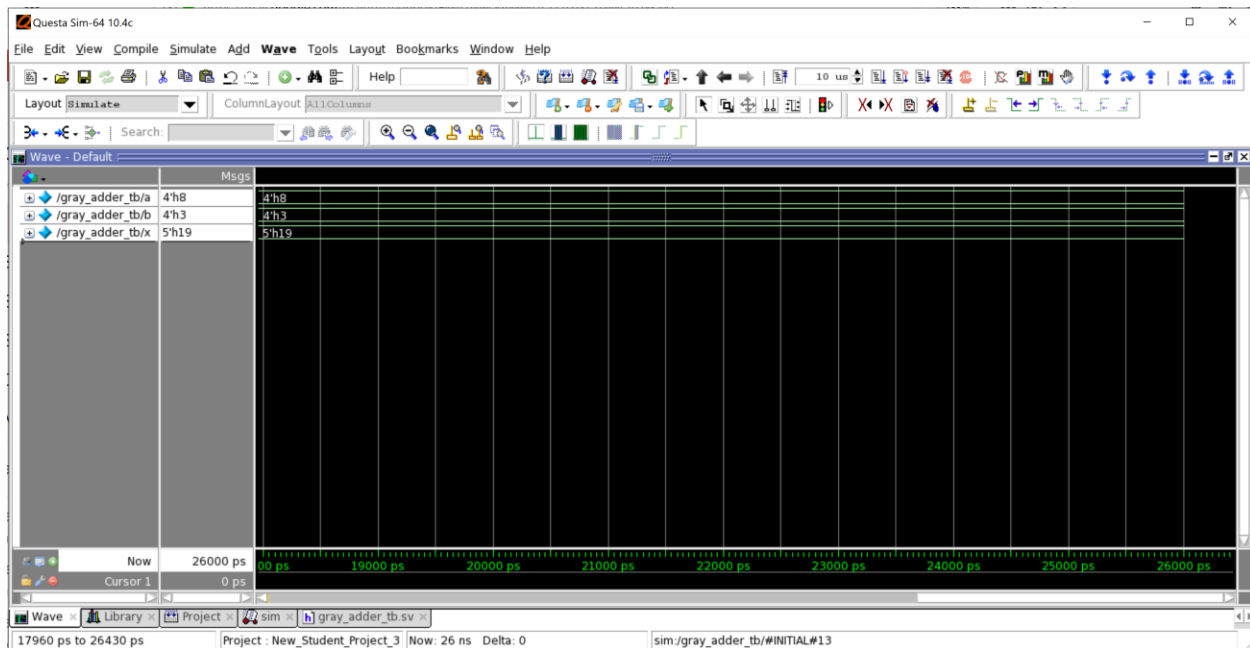
26. It is now possible to view your entire waveform. This is beneficial for debugging your signals and provide their analysis in your report.

27. To exit QuestaSim and to view your output results, press the x on the top right corner of QuestaSim. Next, QuestaSim should prompt you asking if you are sure you want to quit. Press Yes.
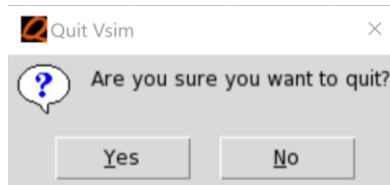
28. In the folder you created for your project, type *vim transcript* to open the transcript file with your results. Your file should be like the example below.

```
0x=xxxxx,a=xxxx,b=xxxx
1x=xxxxx,a=0000,b=0000
1x=0xxxx,a=0000,b=0000
2x=00xxx,a=0000,b=0000
2x=00xxx,a=0001,b=0000
2x=000xx,a=0001,b=0000
2x=000x0,a=0001,b=0000
2x=00000,a=0001,b=0000
3x=00001,a=0001,b=0000
3x=00001,a=0010,b=0000
4x=00000,a=0010,b=0000
4x=00010,a=0010,b=0000
4x=00010,a=0011,b=0000
5x=00011,a=0011,b=0000
5x=00011,a=0000,b=0000
6x=00010,a=0000,b=0000
6x=00000,a=0000,b=0000
6x=00000,a=0000,b=0001
7x=00001,a=0000,b=0001
7x=00001,a=0000,b=0010
8x=00000,a=0000,b=0010
8x=00010,a=0000,b=0010
8x=00010,a=0000,b=0011
```