**CST 133**

**Digital Electronics Design with Verilog**

**Lab 1 - Designing a Full Adder Using Altera Quartus II Design Flow**

Quartus Prime is used in this lab. For simplicity, in our discussion we will refer to this software package simply as Quartus. In this lab we introduce the design of logic circuits using Quartus. Step-by-step instructions are presented for performing design entry with Verilog code. The lab also illustrates the design process and simulation of a Verilog program.

***Getting Started***

Each logic circuit being designed in Quartus is called a *project.* The software works on one project at a time and keeps all information for that project in a single directory in the file system. To begin a new logic circuit design, the first step is to create a directory to hold its files.

Download the Quartus Introduction Using Verilog Designs.

[Link here](ftp://ftp.altera.com/up/pub/Altera_Material/15.0/Tutorials/Verilog/Quartus_II_Introduction.pdf)

This tutorial presents an introduction to the Quartus, the design flow and Simulation. The flow involves the following steps.

1. **Design Entry** – the desired circuit is specified by using a hardware description language, such as Verilog.

1. **Synthesis** – the entered design is synthesized into a circuit that consists of the logic elements (LEs) provided in the FPGA chip.

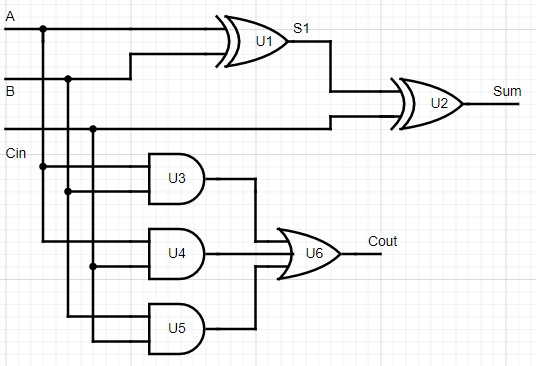
1. **Simulation** – the synthesized circuit is tested to verify its functional correctness;

The code for the design is in the given below:

|  |
| --- |
| // <student name>  // Structural design for a full adder using Verilog module fulladder(  input A, B, Cin, output Sum, Cout);  wire S1, T1, T2, T3;  // Structural code for 1 bit full adder xor U1 (S1, A, B);  xor U2 (Sum, S1, Cin);    and U3 (T3, A, B); and U4 (T2, B, Cin);  and U5 (T1, A, Cin);    or U6 (Cout, T1, T2, T3); endmodule |

# Full Adder Questions

1. Please draw the schematic given the structural code above. There’s also a mistake in the code above. Please fix it.

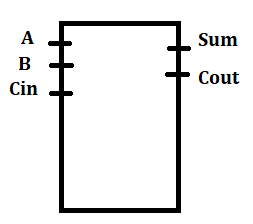


1. Please provide the output equations for Sum and Cout given the full adder above.

sum = (A ^ B) ^ Cin

1. Please provide the truth tables for both Sum and Cout.

Cout = (A & B) |(A & Cin) | (B & Cin)

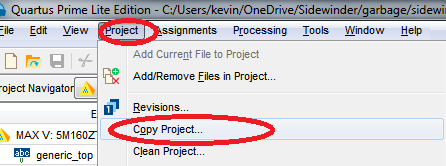
1. Please provide the block diagram for the full adder 

# 1. Introduction to development board

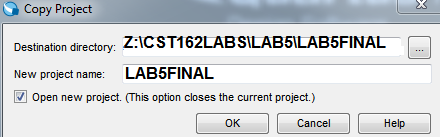
1. Watch these two YouTube videos if you didn’t get this instruction in the last lab period.
   1. [**https://www.youtube.com/watch?v=9bOBzSCxWhs**](https://www.youtube.com/watch?v=9bOBzSCxWhs)
   2. [**https://www.youtube.com/watch?v=lc-Q8OX3Cb0**](https://www.youtube.com/watch?v=lc-Q8OX3Cb0)

# 2. Design

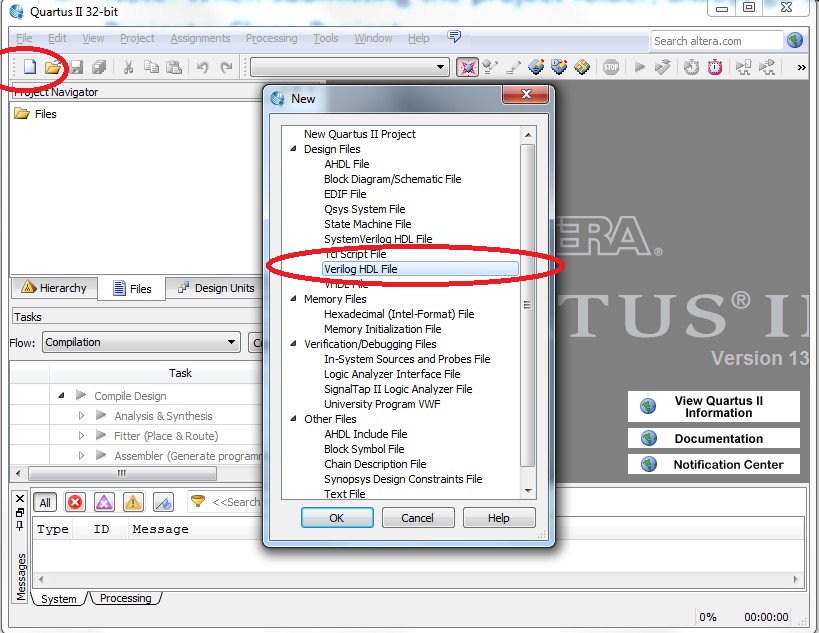
1. Use the default template file provided on the course website.
2. Unzip the template file to your Z drive.
3. Launch the project from your Z drive by double-clicking on the DE10\_generic.qpf file.
4. Go to PROJECT | COPY PROJECT



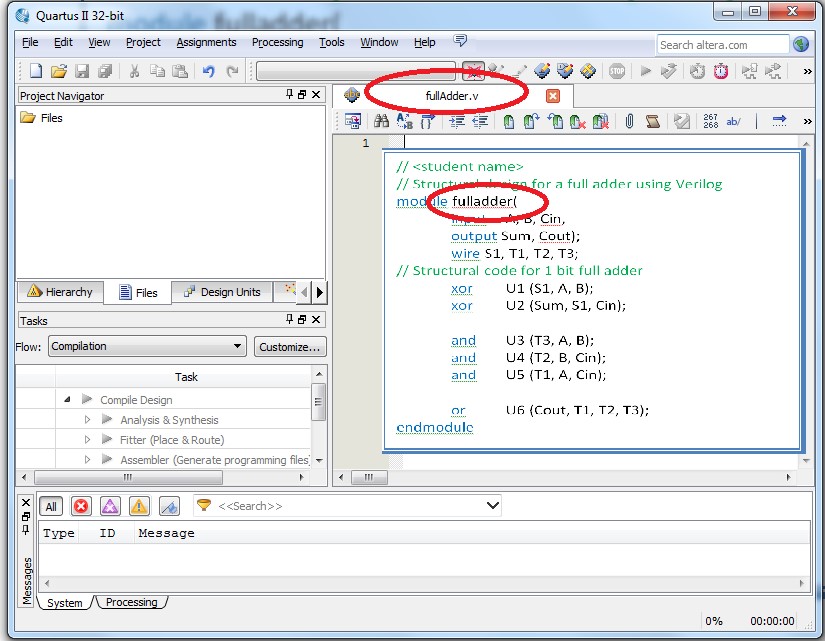
1. Enter LAB1 as the New Project Name. Create a LAB1 folder for this project. Also enter the new project name as LAB1.



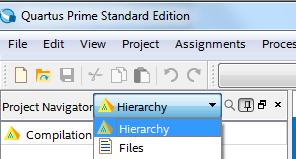
1. Create a new Verilog HDL file and call it fulladder.v. Set it as the top level file by right clicking on it in the Project Navigator File view and clicking on ‘Set as top level file’. Note that the file name must match the module name. fullAdder is not the same as fulladder.



1. Type in a dataflow design based on the simplified Boolean expression developed in step 1 of this section. This code is also listed on page 2 of this lab.

c

1. Perform a functional simulation using the ModelSim simulator (Refer to the tutorial mentioned earlier or contact your instructor). Provide a screenshot of the final functional simulation in your submission.
2. Since we are going to be putting the design on the board, set the de10\_generic.v file in the project navigator as the top level file.
3. Double-click on generic\_top. The de10\_generic.v file should appear. This is your top level file. If it doesn’t, you need to go and change to hierarchy view.



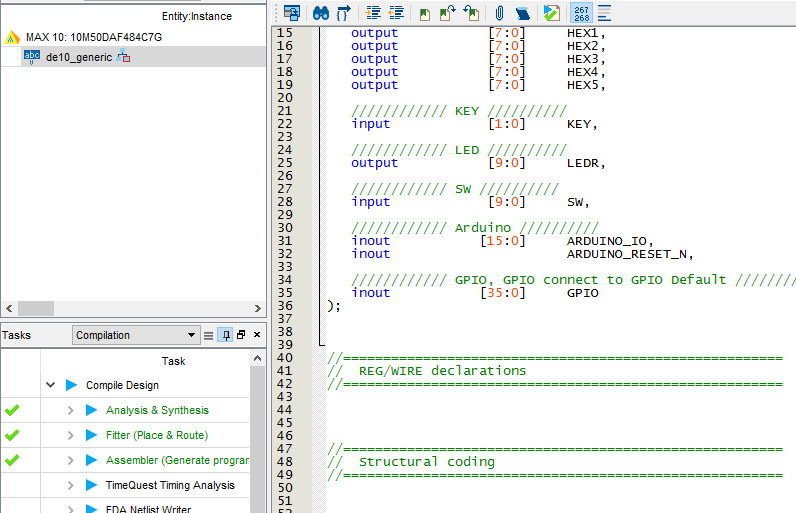
1. Assuming your Verilog module is correctly declared as module fulladder(a,b,Cin, Sum,Cout); in your fulladder.v file, you should put the following statement into the circled area:

We’re instantiating a copy of our **fulladder** module within the top level file called f1. Port A is an input to fulladder and it is being connected to SW[4] on the DE10-Lite board. Sum is an output from fulladder that is being routed to an LED called LEDR[0] on the DE10-Lite board

***If*** we wanted to make more full adders, it is possible to instantiate multiple copies of fulladder. We might call additional copies f2, f3, f4, etc. and connect them to additional switches and LEDs.

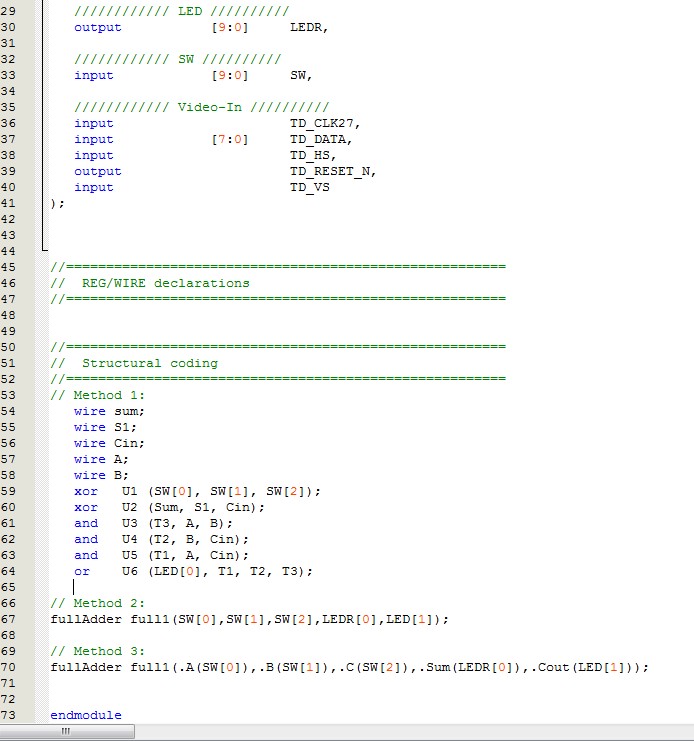
***We are connecting by port name. You can also connect by port position but I do not recommend that method.***

***Also, I made two mistakes on the line I told you to copy on purpose. Fix them.***



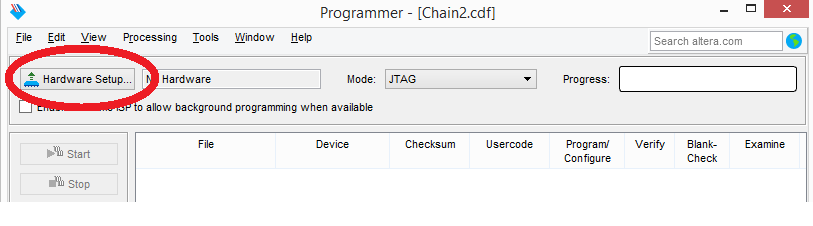
1. For your information (Don’t copy the code below…), the line I told you to copy could also be instantiated the following ways:
   1. Method 1 is instantiating all the gates in the top.v file. You can’t reuse this and have to copy and paste all the code every time.
   2. Method 2 is modular allowing for reuse of the module called fullAdder. It connects the I/O by position.
   3. Method 3 is modular allowing for reuse of the module called fullAdder. It connects the I/O by name. Method 3 is preferred. Why?

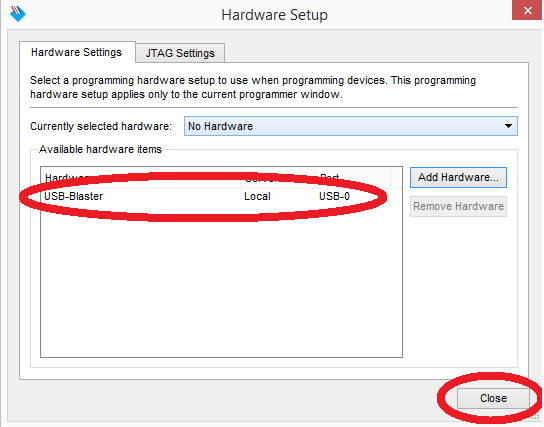
Connecting by name is preferred because connecting by position could lead to a situation where the positions of the inputs/outputs are changed and the connections are no longer going where they were before.

\

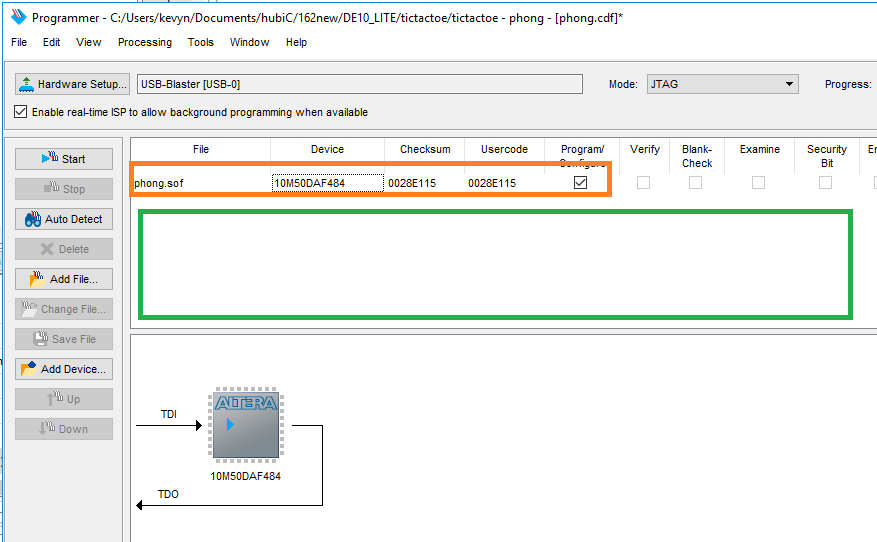
We now want to proceed to compile and program the design to the board. See next page.

# 3. PROGRAMMING AND TESTING THE DESIGN

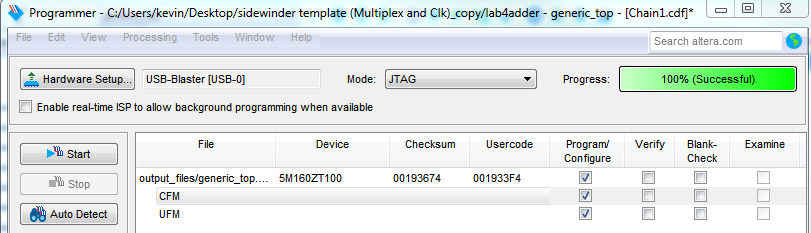
1. Compile your design before proceeding: PROCESSING|START COMPILATION.
2. Make sure the compilation is successful.
3. Click TOOLS|PROGRAMMER, then click on the Hardware Setup button. 
4. Double click on USB-Blaster until it shows up as ‘Currently selected hardware’. Then click close. If you can’t see the USB-Blaster, make sure your Sidewinder board is plugged into the USB port. Otherwise, contact your instructor.



1. Usually, the default file that populates is correct. In this case, my project name was phong. Your .sof file will have the name of your project. Verify that the 10M50DAF484 device is selected.



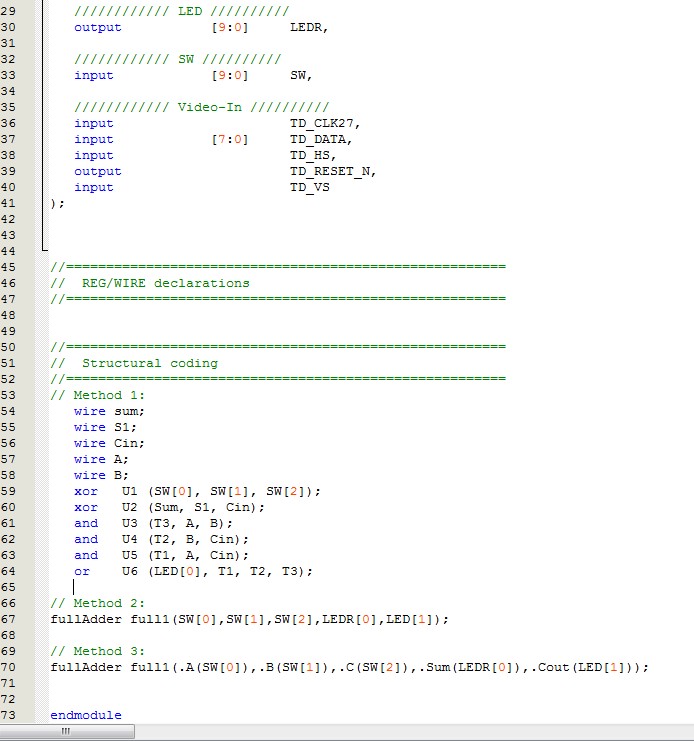
1. If you find that the file is incorrect, right click in the green area and navigate to your project on the student file server (the Z drive) or wherever you decided to put the project and find the \*.sof, which is the SRAM Object File from your project. Once you have located it, click **Open**. In the programmer window, you should now see your File name, the Device (10M50DAF484). There will be several checkboxes to the right of this information. Make sure the checkbox labeled Program/Configure is checked. If there are any other devices, click on them and delete them as they will cause your programming to fail.
2. Click on STARTin the left toolbar. Wait until the configuration of the FPGA successfully completes.

It is now configured and ready to test. 

1. Test your design and have the instructor check off the design.

Design 2

1. Go back to your code and comment out the full adder you instantiated by putting // in front. The code should turn green.



2. Make a new file called fullAdder2. In this file, you should have the same I/O as before.

Inputs: A, B, C

Outputs: Sum, Cout.

This time inside the fullAdder2 file, use the equations you developed in part 0 for Sum and Cout and the assign statement. Assign statements are another way to implement combinational logic.

assign sum = Your equation

assign cout = Your equation

3. Instantiate a copy of your fullAdder2 in the top.v file. To instantiate a copy, call the module you want followed by a name for the module, and map the I/O. See line 67 in step 1 above.

Additional Questions

1. Why should we connect by port name instead of port position (method 2 vs method 3)?

Connecting by port name is preferrable because it is more explicit in that it specifies that the exact ports match up as expected, even if positions of inputs/outputs change at a later time.

2. How do we instantiate a copy of a module? Can we instantiate more than one copy of a module? Is there a limit to the number of copies you can create?

module\_name variable\_name(.label\_name(port\_name))

More than one copy can be instantiated, and as many copies can be instantiated as space permits on the device being used.

3. Why don’t we just write everything in the top file (method 1) What are the benefits of creating these modules such as fullAdder and fullAdder2 (method 2, method 3)?

Creating modules allows for code reuse where you can instantiate copies of simpler circuits to implement more complex circuits without having to rewrite or rework anything.

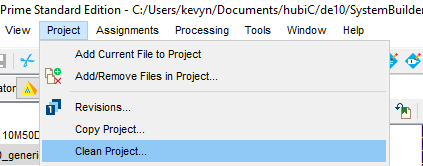
Submissions

The instructor does not guarantee last second availability. It is your responsibility to submit the lab in a timely manner. Note that the instructor’s mailbox may not be open after 5 PM.

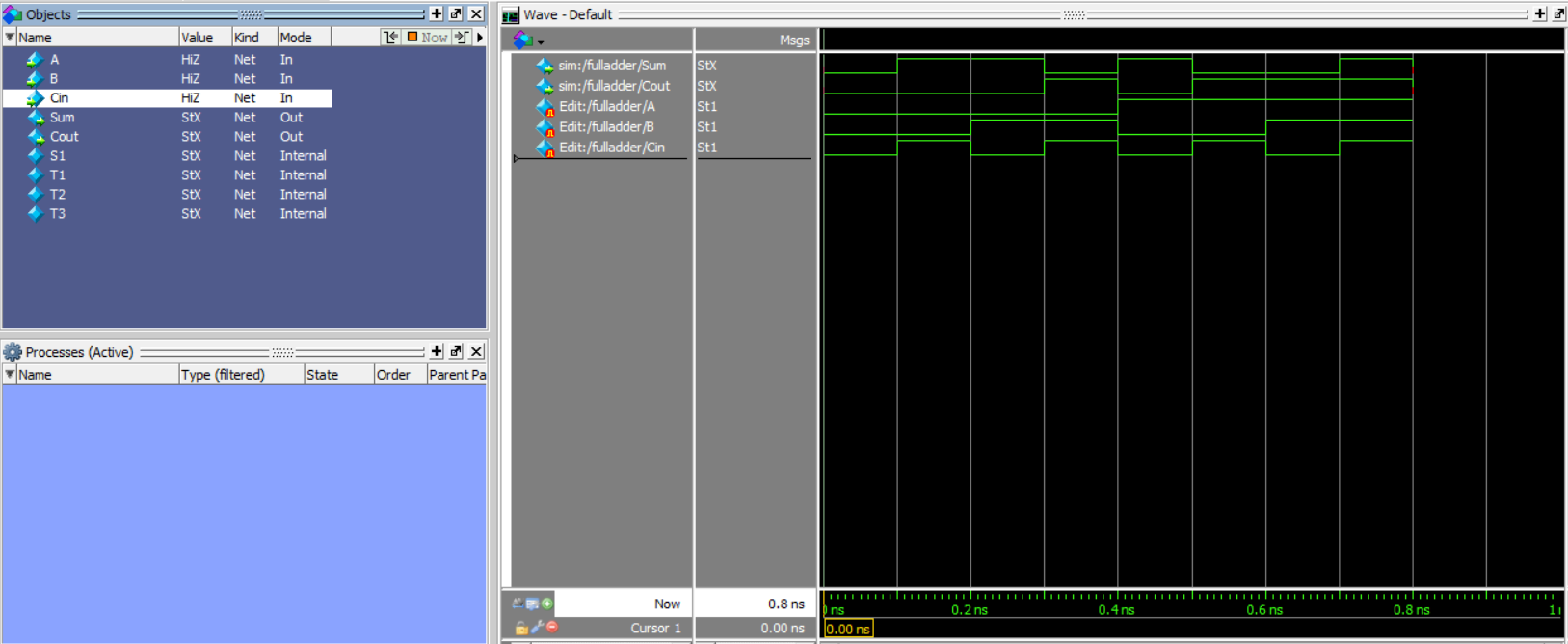
A penalty of 15% per day up to 4 days will apply. After 4 days, no submissions will be accepted and you will receive a score of 0 for the lab.

For this lab you must submit the following to the course website

1. Demo the assignment to instructor in person before the start of the next lab.
2. Cleaned and zipped project folder. If you do not clean the project you will be subject to a 10% deduction in grade for this assignment.



1. Screen capture of project simulation



1. Questions answered