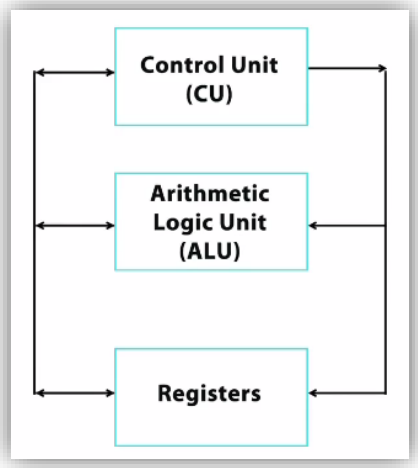
Laboratory 2: Generic Full Adder

1 Introduction

In the previous lab we created a single bit full adder via both the behavioral and hierarchical techniques. The main goal this semester is to create our own custom processor which is composed of a Control Unit, Arithmetic Logic Unit, and registers [see Figure 1 below]. In order to actually process meaningful data such as audio, a single bit adder will not suffice and we must therefore create a multiple bit adder. For example, most audio streams consist of 16 bit raw data which is actually the data format found on audio CDs. In order to process audio data we would need a 16 bit full adder. In this lab we will create a 4 bit adder and verify it via Modelsim by the use of a test bench with assert statements. In subsequent labs we will extend the adder bit width as needed for our intended purpose. We will again explore the difference between behavioral and hierarchical design techniques by viewing the generated RTL in the Quartus RTL viewer. Lastly the use of generics and generate statements will be explored in order to achieve a customizable full adder bit width. Other concepts touched upon in the lab include the differences between concurrent and sequential statements, loops, and delta delays. As usual, version control is required for this lab.



- **Note: This lab is very little code, but rather focuses on folder organization and setting up simulations and hardware compilations. Please take your time and be very intentional when unzipping, naming, and moving folders around.**
- **Note: do not modify the generic_adder_beh.vhd file during this lab.**

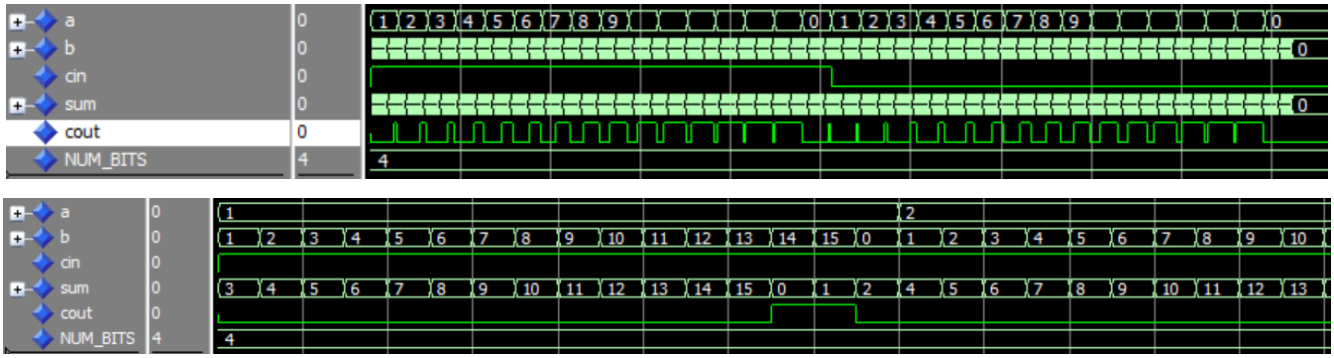
2 Generic Adder Simulation [Behavioral] with Sequential Test Bench [2 bit]

- Unzip and simulate generic_adder_beh.zip by clicking on simulate.bat in the /sim/script folder. Make sure you have the below folder structure as sometimes unzipping a file adds another folder layer.
    - Lab2
        - generic_adder_beh
            - src
            - sim
                - src
                - script

- Notice the **** sequential testbench start **** description in the transcript window
- Notice the green markers in the wave window that show the 'report' messages from the test bench.
- Take a look at the wave window do double check the math and then close out of Modelsim.
- Set the SEQUENTIAL_FLAG in the test bench to false and rerun the simulation.
- Modify the 'radix' of the wave signals to 'unsigned'.
- Receive a sign off but be ready to answer the below questions.
    - Why did the concurrent simulation run repeatedly but the sequential simulation stopped after one run?
    - What does the generate statement do in the test bench?
    - What is incomplete about this simulation?

3 Generic Adder Simulation [Behavioral] with Sequential Test Bench [4 bit]

- Convert the test bench to simulate a 4 bit adder. Make sure to set the SEQUENTIAL_FLAG to true.
- Add the 'cin' signal into the test bench to simulate out all possible input scenarios.
- Verify your waveform against the below answer keys [one is zoomed in].



- Receive a sign off but be ready to answer the below questions.
    - Why does the SEQUENTIAL_FLAG need to be true for the 4 bit case?

4 Generic Adder Compilation [Behavioral]

- Click on the compile.bat file in the /hw folder.
    - Note: on your home computer you might have to add C:\altera_lite\15.1\quartus\bin64\ or something like that to your system path. You can check if your path is correct by typing 'quartus_sh -version' at the command prompt and see if it is recognized.
- Open up Quartus 15.1 and investigate the RTL viewer.
- Receive a sign off but be ready to answer the below questions.
    - Why aren't there 'and', 'or', or 'xor' circuits in the RTL viewer?

5 GENERIC ADDER SIMULATION [HIERARCHICAL]

- Create a completely new folder so that your structure looks like the one below. This is done by simply copying the generic_adder_beh folder, pasting it, and renaming it to generic_adder_arch. All this section of the lab is doing is pulling down parts of project and weaving them all together to work. This is a very important skill to have and it is very important to take the time and to pay attention to details. If you do not pay attention to your directory structure this section will take you forever.
    - Lab2
        - generic_adder_arch
            - src
            - sim
        - generic_adder_beh
            - src
            - sim
- Remember to add everything to version control, and then do a commit.
- Pull down the generic_adder_arch.vhd file and place it into the src folder.
- Pull down the entire full_adder_single_bit_arch.zip file and unzip it into the src folder.
- Your folder should look like this. We are effectively importing the entire single bit adder into our src folder.
    - Lab2
        - generic_adder_arch
            - sim
                - script
                - src
            - src
                - generic_adder_arch.vhd
                - full_adder_single_bit_arch
                    - sim
                        - src
                        - script
                    - src
- modify your test bench to point to the arch version of the generic adder. Should only have to change the component name.
- update the sim.do file to point to all of your project files.
    - vlib work
    - vcom -93 -work work ../../src/full_adder_single_bit_arch/src/alu_and.vhd
    - vcom -93 -work work ../../src/full_adder_single_bit_arch/src/alu_or.vhd
    - vcom -93 -work work ../../src/full_adder_single_bit_arch/src/alu_xor.vhd
    - vcom -93 -work work ../../src/full_adder_single_bit_arch/src/full_adder_single_bit_arch.vhd
    - vcom -93 -work work ../../src/generic_adder_arch.vhd
    - vcom -93 -work work ../src/generic_adder_tb.vhd
    - vsim -novopt generic_adder_tb
    - do wave.do
    - run 6 us
- run the simulation for 4 bits. Make sure to set the SEQUENTIAL_FLAG to true.
- Drag in the appropriate wave signals.

- ☐ Receive a sign off.

6 GENERIC ADDER COMPILATION [HIERARCHICAL]

- ☐ Update the compile.tcl folder in the /hw folder to point to all the files just like you did with the sim.do
- ☐ Click on the compile.bat file in the /hw folder.
- ☐ Open up Quartus 15.1 and investigate the RTL viewer.
- ☐ Receive a sign off but be ready to answer the below questions.
    - What is different between the hierarchical and behavioral multi-bit adder RTL views?

7 DELIVERABLES

To receive full credit for this lab one must hand in the below items no later than 168 hrs [7 days] after the start of one's lab session. Signoffs can be obtained after the due date as long as the time stamp of the code is from before the deadline.

- ☐ Hard copy of this document.
- ☐ Hard copy of important file [no tabs and print from notepad++ with 'show symbols' on].
    - generic_adder_tb.vhd [can be targeting the beh or arch versions.. doesn't matter which one]
    - generic_adder_beh.vhd
    - generic_adder_arch

8 SIGNOFFS

Category	Initials	Date	Points
Behavioral Simulation [2 bit]			/10
Behavioral Simulation [4 bit]			/20
Behavioral RTL View			/10
Hierarchical Simulation [4 bit]			/20
Hierarchical RTL View			/10
Version Control			/10
Header/Comments/Tabs			/10
Deliverable			/10
Final Grade			/100