

## ECE 385 – DIGITAL SYSTEMS LABORATORY

### A Simple Design in VHDL Using Altera Quartus II 6.1 Web Edition

#### Design a full-adder and use it to design a 2-bit adder:

You need to be on a machine with Windows XP, either on your own machine or in the ECE Student Lounge (167 Everitt). You cannot do this exercise on a Unix workstation. In order to perform this exercise on your own computer, you will need to install the Quartus II software from the CD included in your FPGA kit (you may also download the installer from [www.altera.com](http://www.altera.com)). Follow these steps to fully install the software:

- Install Quartus II from the included Quartus II Web Edition Software Suite CD or the web installer.
- Launch Quartus II. At the first startup, a dialog box will pop up asking for license setup. Please choose “Perform Automatic Web License Update”. This will open your web browser to the license request form on Altera’s website. Fill out the form and click *Submit Request*. You will receive an email containing your license file and instructions on how to install it shortly. Follow these instructions to complete the license setup.
  - If the automatic license request didn’t work, you can request a license manually by visiting [https://www.altera.com/support/licensing/free\\_software/lic-q2web.jsp](https://www.altera.com/support/licensing/free_software/lic-q2web.jsp) and filling out the form there. You will be asked to provide your Network Interface Card (NIC) ID. This is the MAC address of your network card, without any dashes, colons, periods, or other punctuation that might appear. For your convenience, Quartus II will automatically detect your NIC ID and display it in the *License Setup* window, in the first of the *Local System info* fields.
- Insert the included DE2 System CD. If you do not currently have the CD on hand, you can download its entire contents from <http://www.altera.com/education/univ/materials/boards/unv-de2-board.html> (link is near the bottom of the page; download size is 64 MB). Open the file `\DE2_tutorials\tut_initialDE2.pdf`. Follow the instructions in part 3 to setup the USB Blaster driver used to program the FPGA.
  - NOTE: The instructions specify `altera\quartus50\drivers\usb-blaster` as the location to find the driver. This is the default location under Quartus II v5.0, but it may be different if you installed the program to a non-default location, and it needs to be updated for Quartus II v6.x. The correct location will be `<your install location>\<version code>\quartus\drivers\usb-blaster\32`, where `<your install location>` is just that (by default, `C:\Altera`) and `<version code>` is `60` for version 6.0 (CD install) and `61` for version 6.1 (web install).

You are now ready to begin the exercise.

#### **Create a New Project:**

- From the *File* menu select *New Project Wizard*. Click *Next* to go through the intro screen, if it appears.

- The window in Figure 1 will appear. Fill in the fields from figure 1 (make sure there are no spaces in any of your entries). The program will ask you if it should create the specified directory if it does not exist; choose *yes*.
- Select *Next* on page 2 without adding any files.
- On page 3, select Cyclone II for the device family, make sure the second option under Target device is selected, and chose EP2C35F672C6 in the Available devices list. See Figure 2.
- Click *Next* on page 4 without making any changes, and click *Finish* on page 5.
- You should see an entry for the project in the *Project Navigator* window. It should appear as in Figure 3.

**New Project Wizard: Directory, Name, Top-Level Entity [page 1 of 5]**

What is the working directory for this project?

C:\vece385\2Bit\_Adder ...

What is the name of this project?

2Bit\_Adder ...

What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.

adder2 ...

Use Existing Project Settings ...

< Back   Next >   Finish   Cancel

Figure 1

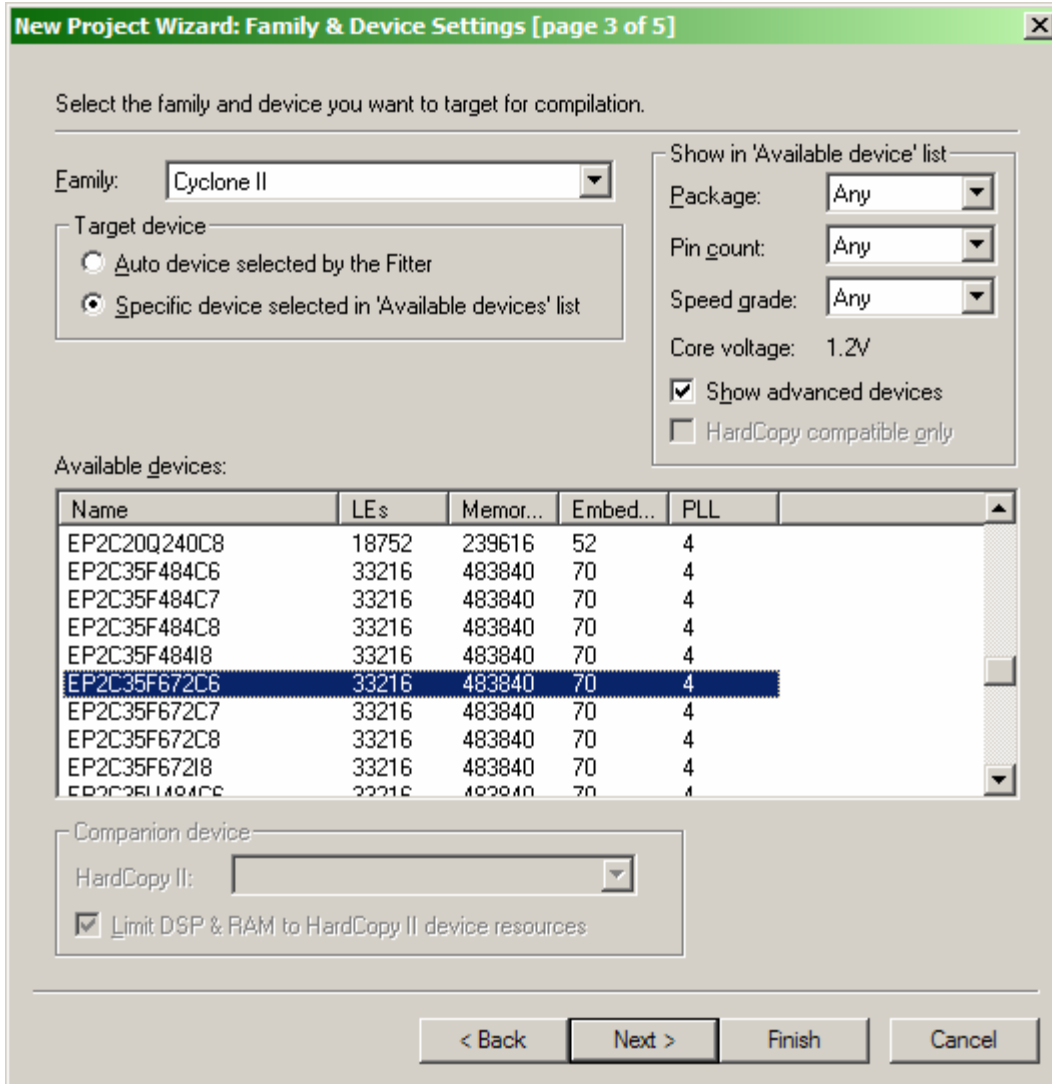


Figure 2

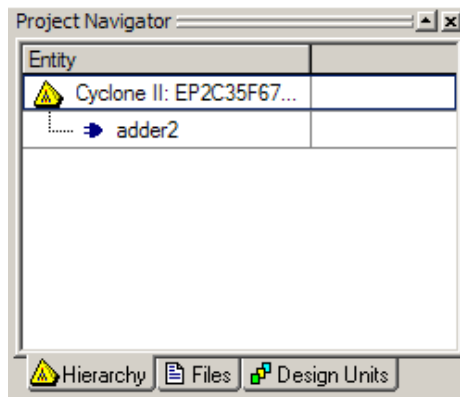


Figure 3

### Create a Full-Adder Entity:

Now you can start entering your design. In the *File* menu, select *New...* Select *VHDL File* in the window that pops up as shown in Figure 4 and click *OK*. You will be presented with a blank VHDL file. Enter the code below to define the full adder unit. Save the file as `full_adder.vhd`. Figure 5 shows schematic representations of what the code defines; if you do not understand what the code means, you should review the Introduction to VHDL on pages IVT.1-15.

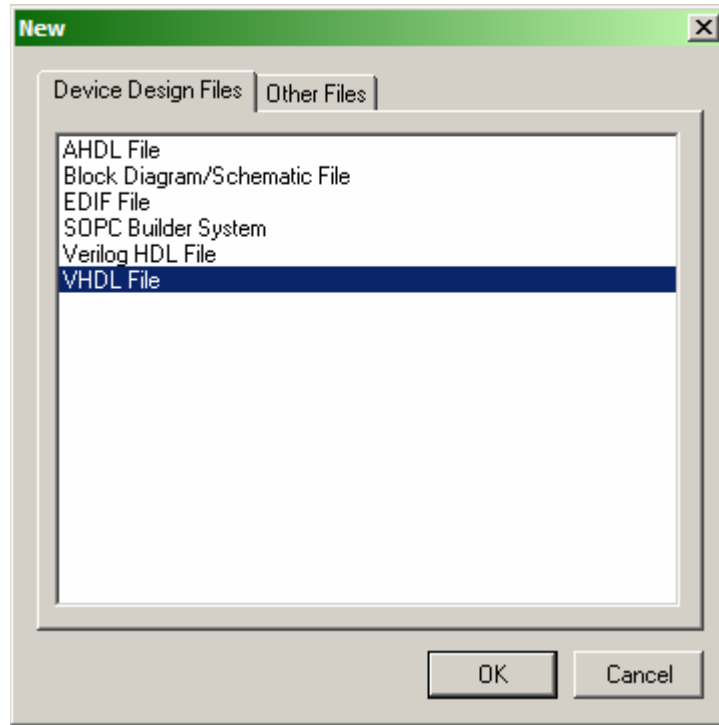


Figure 4

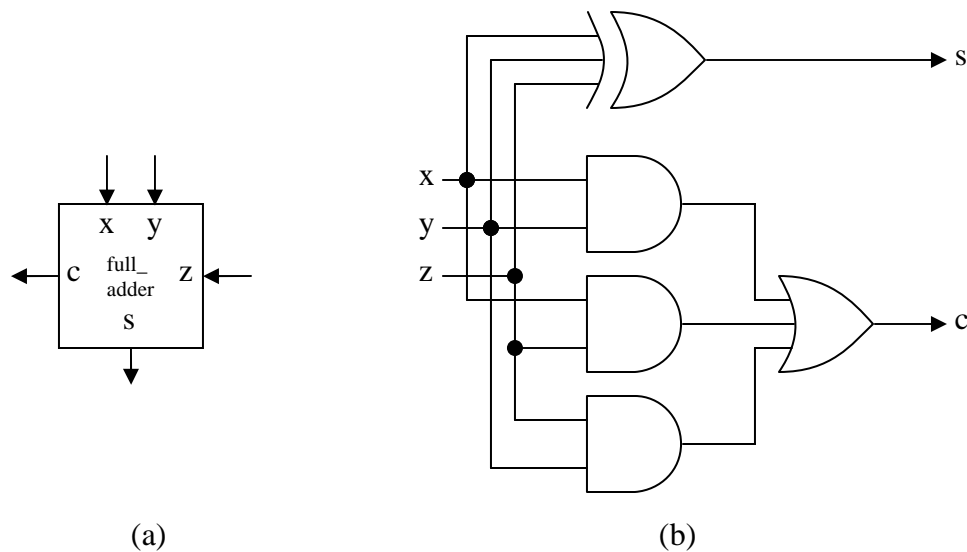


Figure 5: Schematic depiction of the meaning behind the Full Adder (a) entity block and (b) architecture block.

---

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity full_adder is
  Port ( x : in std_logic;
        y : in std_logic;
        z : in std_logic;
        s : out std_logic;
        c : out std_logic);
end full_adder;

architecture Behavioral of full_adder is
begin

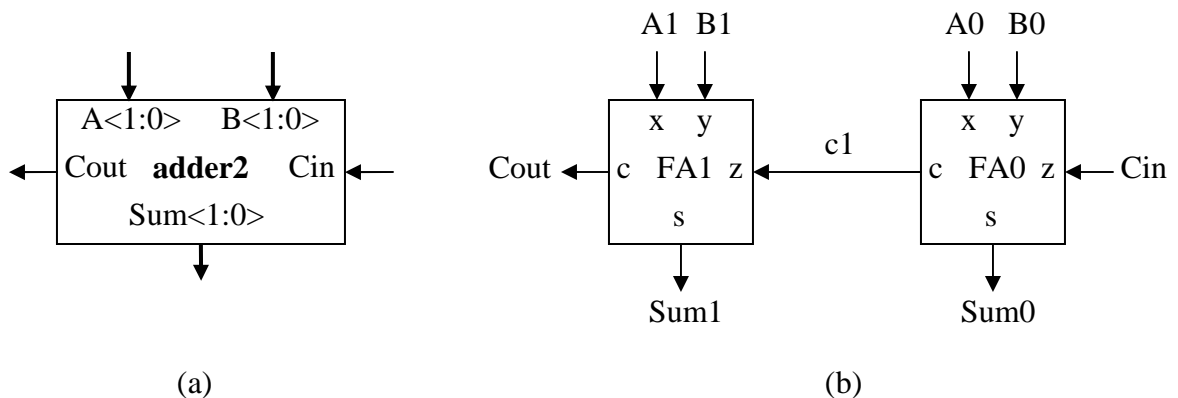
  s <= x xor y xor z; -- This line computes Sum
  c <= (x and y) or (y and z) or (x and z);
      -- This line computes Carry
end Behavioral;

```

---

### Create an Entity for 2-Bit Adder:

Now, follow the same steps to create a new file that defines a 2-bit adder with inputs A (2-bits), B (2-bits) and Cin, and outputs Sum (2-bits) and Cout. The code is given below. Save the file as adder2.vhd.



**Figure 6:** Schematic depiction of the meaning behind the 2-Bit Adder  
 (a) entity block and (b) architecture block.

---

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity adder2 is
    Port ( A : in std_logic_vector(1 downto 0);
          B : in std_logic_vector(1 downto 0);
          Cin : in std_logic;
          Sum : out std_logic_vector(1 downto 0);
          Cout : out std_logic);
end adder2;

architecture Behavioral of adder2 is

-- Declare component full_adder
-- Reproduce the entity declaration here as a component.
component full_adder is
    Port ( x : in std_logic;
          y : in std_logic;
          z : in std_logic;
          s : out std_logic;
          c : out std_logic);
end component full_adder;



signal c1: std_logic; --internal carry bit for the 2-bit adder
begin

-- Instantiate and connect two full_adders to create a 2-bit adder.
FA0: full_adder
port map( x => A(0),
          y => B(0),
          z => Cin,
          s => Sum(0),
          c => c1);

FA1: full_adder
port map( x => A(1),
          y => B(1),
          z => c1,
          s => Sum(1),
          c => Cout);

end Behavioral;
```

---

Once you have entered and saved the code for `adder2.vhd`, click the *Start Analysis & Synthesis* button (  ) in the toolbar. Quartus II will check the code for correct syntax, and generate errors or warnings if there is anything wrong or questionable about your code. A message pops up when the program has finished; you should get no errors or warnings if you have entered the code correctly. If you find errors in synthesis, correct the code, save the file, and run *Analysis & Synthesis* again. Quartus II also builds the hierarchy in the *Project Navigator* window, which should look like figure 7. Next, click the *Start Compilation* button (  ) in the toolbar. This time, you will get a few warnings about timing characteristics and load capacitances. You can ignore these. Now you have created a 2-bit adder and you can simulate the design.

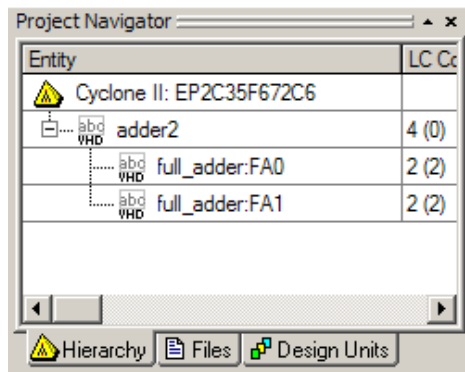


Figure 7

### Using Simulator:

To create a Test-Bench for your design, go to the *File* menu, select *New...*. On the *Other Files* tab, select *Vector Waveform File* as shown in Figure 8 and click *OK*. A blank testbench will come up.

The first thing we'll want to do is populate the testbench with the signals we're interested in. In the *Edit* menu, select *Insert -> Insert Node or Bus...*. Click the *Node Finder...* button in the window that pops up. In the *Node Finder* window, set the fields as shown in Figure 9, and click the *List* button. A list of found nodes will appear in the *Nodes Found* box. Make the selections as shown in Figure 9, and click the *>* button to move them to the *Selected Nodes* box. Your window should now look just like Figure 9. Click *OK* to return to the *Insert Node or Bus* window, where "\*\*\*Multiple Items\*\*" now appears in several of the fields. Click *OK*.

Next, we will set the testbench length and view. In the *Edit* menu, select *End Time...*, enter 320 ns, and click *OK*. Finally, in the *View* menu, click *Fit in Window*. Your view should now look like Figure 10. Save the testbench as `wave1.vwf`.

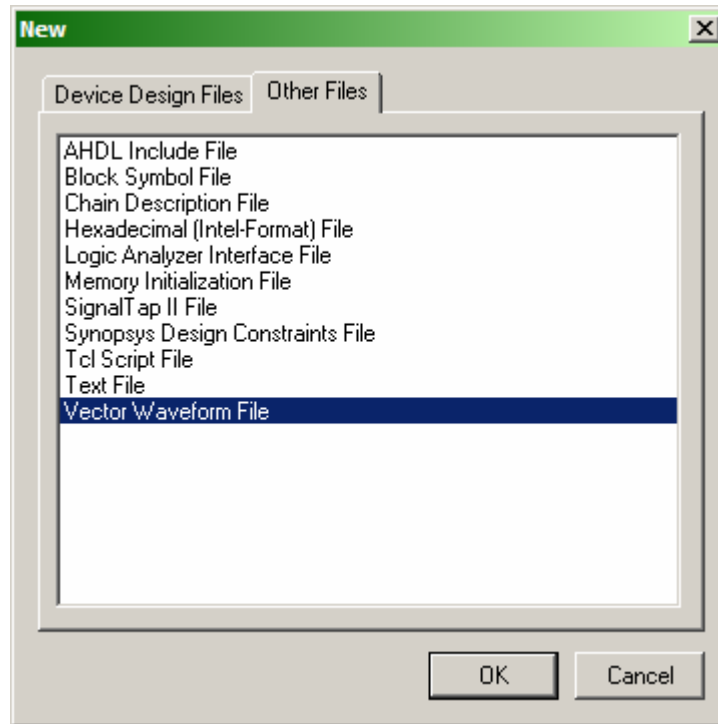


Figure 8

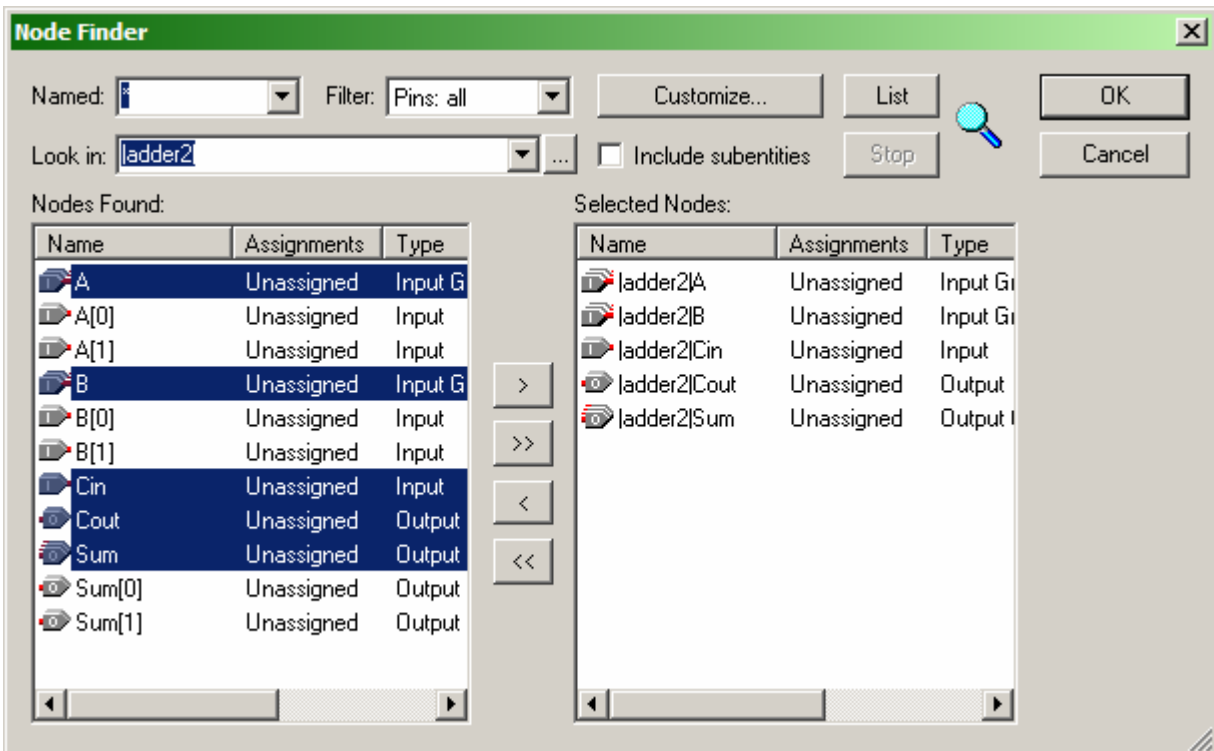


Figure 9

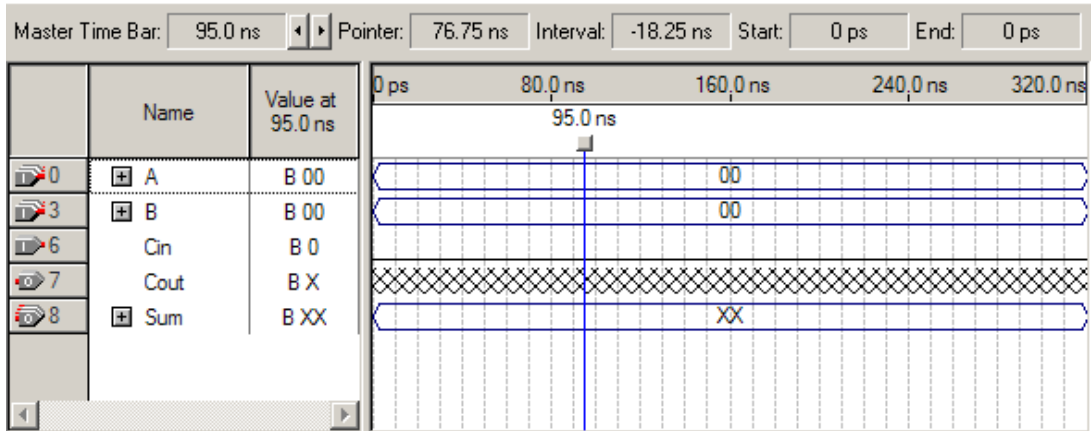







Figure 10

We now have to set values for the inputs of our circuit. Setting values in the waveform editor is done by selecting areas of time and then setting values for that time. We can select one signal for an arbitrary portion of time, select one signal for the entire simulation runtime by clicking on its name, select one signal for multiple disjoint periods of time by holding control as we select each region, and even select multiple signals and assign them all values at once. By default, the selections snap to a grid; the default grid size is 5 ns.

As an example, click on the Cin signal's name to select it for the whole simulation time. Click the *Overwrite Clock* () button to define it as a clock. You can set the clock period, duty cycle, and offset. Click *OK* and observe the result. This will be useful for defining clocks in lab 7 and beyond, but for now, click the  button to set the Cin signal back to logic low for the entire simulation, select the signal for the second half of the simulation (time 160 ns to 320 ns), and click the  button to set it to logic value 1 in this region.

Next we will set the values of A and B. A and B are buses, with two bits each. We can use the standard tools to set all signals in the bus together, but that will only let us set values of 11 and 00. There are several ways to set values for buses. One is to expand the bus by clicking on the + sign next to its name and setting each bit individually. Another is to use the *Arbitrary Value* () button to set some arbitrary value to the bus for the selected time period. Often, though, you'll want to generate some sort of pattern for testing purposes. Select bus A for the whole length of the simulation, and click the *Count Value* () button. Observe the values on the *Counting* and *Timing* tabs, and click *OK* without changing anything. A will cycle through its four possible values, changing every 10 ns. Do the same for B, but change its value for *Count Every* on the *Timing* tab from 10 to 40. You should see that B is counting once for each full cycle of A. Your waveform should now look like figure 11. Between the changed in the three signals, all possible input combinations to the circuit are represented. Save your waveform again now.

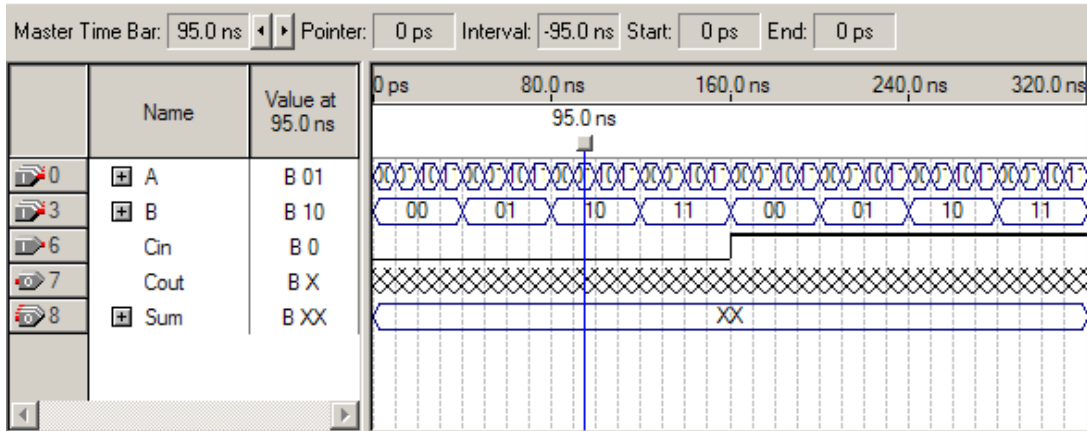



Figure 11

To run a functional simulation of your design, right-click on *adder2* in the *Project Navigator* window and select *Settings....* Click on *Simulator Settings* in the *Category* box. Change *Simulation mode* to *Functional*, and select your waveform as the *Simulation input* (you can browse for it using the “...” button). Click *OK*. Additionally, in the *Processing* menu, select *Generate Functional Simulation Netlist*. Once this finishes, run the simulation by clicking the *Start Simulation* (  ) button. Once the simulation finishes, you should see output that looks like Figure 12. You may have to go to the *View* menu and select *Fit in Window* again to see the whole simulation. Check the output values based on the inputs you provided.

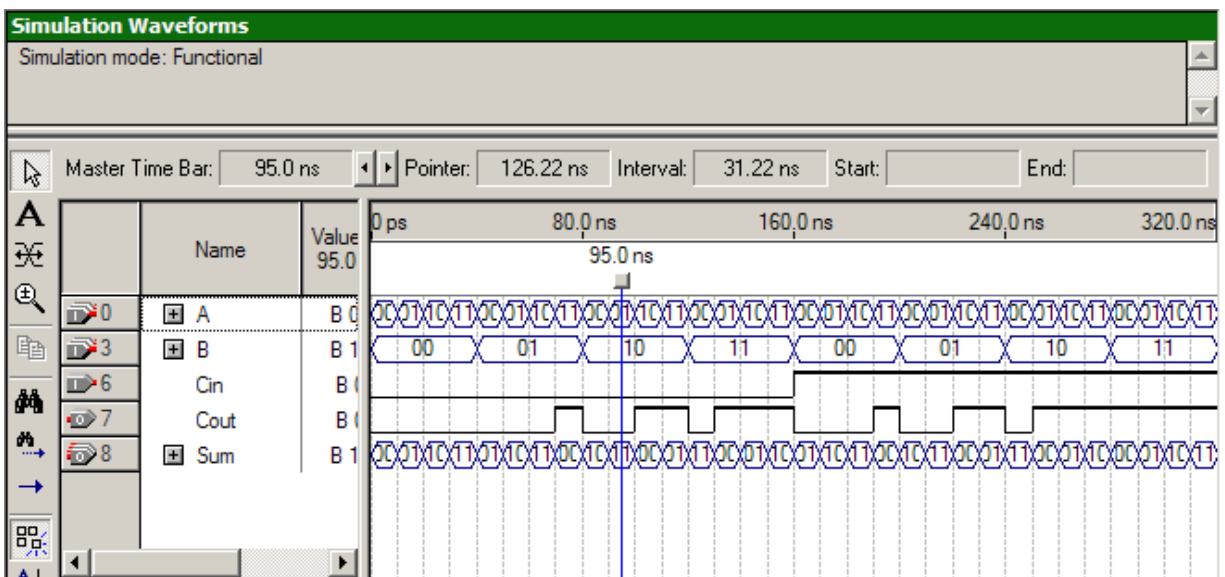




Figure 12

### Entering Pin Assignments:

Click the *Assignment Editor* button (  ) in the toolbar to open the Assignment Editor. Select *Pin* on the dropdown menu to show only pin assignments. Enter the assignments from Table 1 in sequence. The editor should look similar to Figure 13. Save the assignment editor view and recompile the design (  ).

Port Name	Location	Comments
Cin	PIN_N25	On-board slider switch (SW0)
A[0]	PIN_N26	On-board slider switch (SW1)
A[1]	PIN_P25	On-board slider switch (SW2)
B[0]	PIN_AE14	On-board slider switch (SW3)
B[1]	PIN_AF14	On-board slider switch (SW4)
Sum[0]	PIN_AE23	On-Board LED (LEDR0)
Sum[1]	PIN_AF23	On-Board LED (LEDR1)
Cout	PIN_AB21	On-Board LED (LEDR2)

Table 1: Pin Assignments

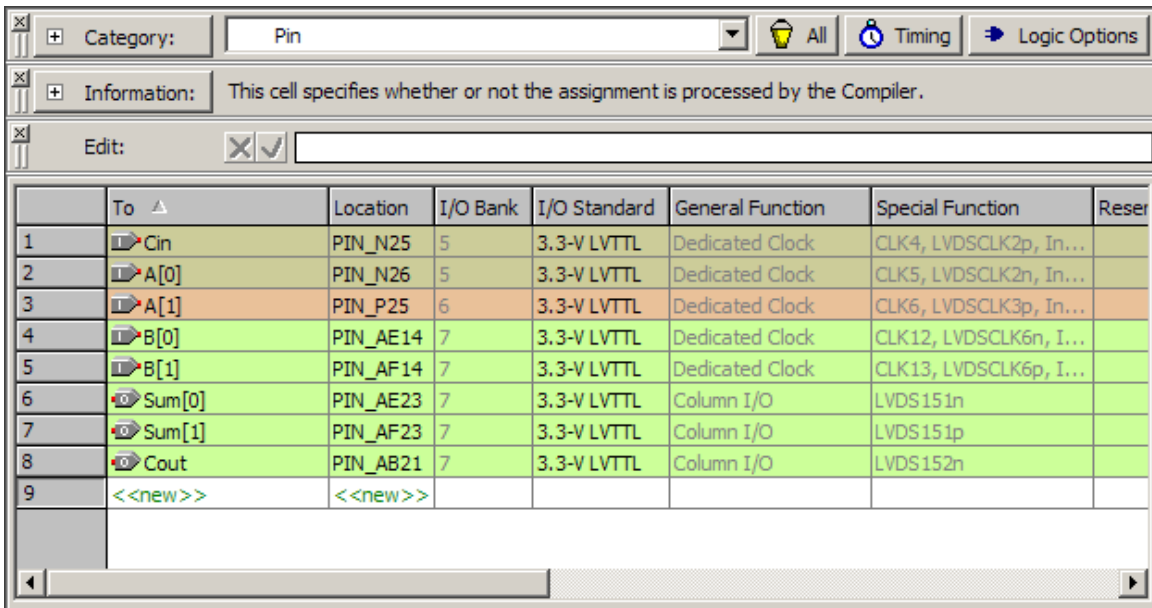



Figure 13

**Programming the FPGA:**

Plug in the FPGA power, and connect the FPGA Blaster port to the computer with the included USB cable. Click the *Programmer* button (  ) to open the programmer. The screen should list the file *adder2.sof*. Turn on the FPGA power, and click the *Hardware Setup...* button. Select *USB Blaster* from the *Currently selected hardware* dropdown list, and click *Close*. Back in the programmer, check the box next to *adder2.sof* under *Program/Configure*. Finally, click the *Start* button. The FPGA will be programmed with your design. Toggle the switches and verify that the circuit is performing correct 2-bit addition with carry-in.

**Inform your TA:** Once you have completed this exercise, e-mail your TA with “ECE 385 - Done with simple design” in the *Subject* of the e-mail.