

ENGR 304

Hardware Lab 1: ASCII Addition on the Altera DE2 Board

Objectives

After completing this lab you should:

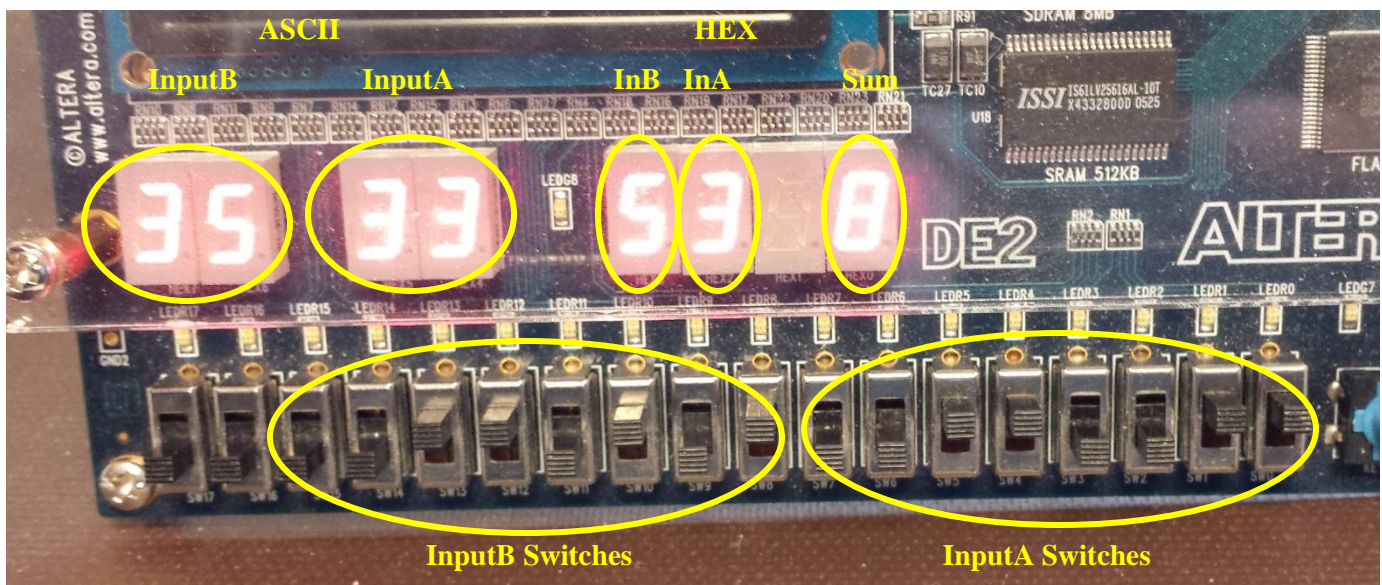
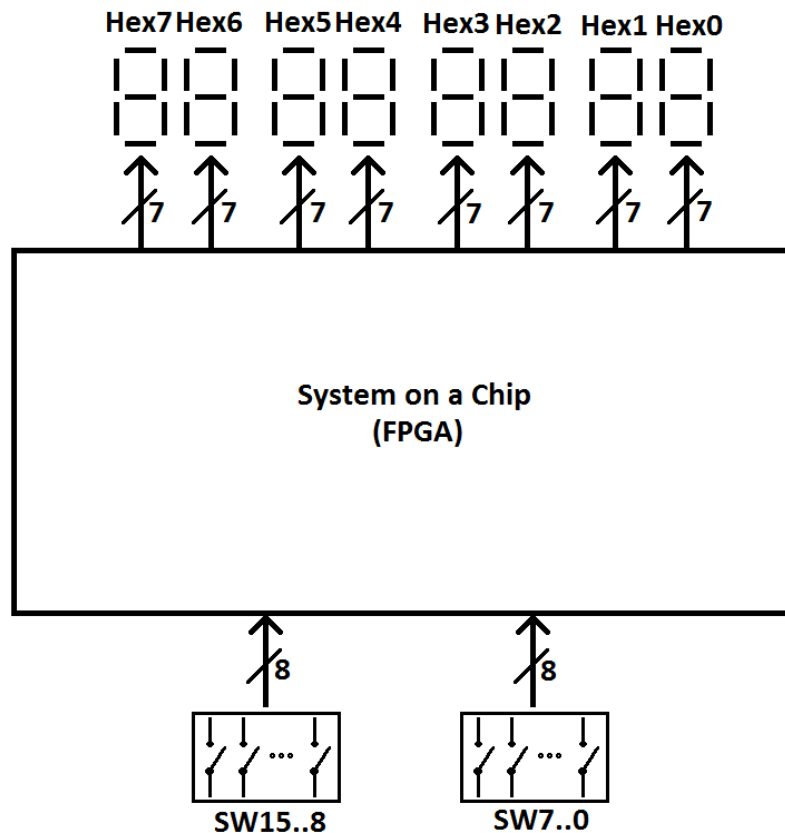
- recall binary arithmetic operations,
- recall how to use Quartus design software,
- know how to build basic block diagrams,
- be able to write simple VHDL for combinational logic, and
- be able to program the Altera part on the Altera DE2 board and test it out.

Specifications of the Design

In this design, you shall input two numbers and display the addition of the two numbers. Each number is created from one hexadecimal digit and each hexadecimal digit is entered using its 8-bit ASCII code (the characters "0-9" and "a-f" or "A-F"). Toggle slide switches shall be used to set the state of each bit for the hex digits being inputted to the system. Seven segment displays shall show the bit patterns (as hex digits) of each input number and shall show the result of the operation performed. All numbers and calculations should be done using unsigned representation and arithmetic and should allow for an 8-bit result (even though only 5 bits are needed).

Specifically, the inputs and outputs are assigned as follows:

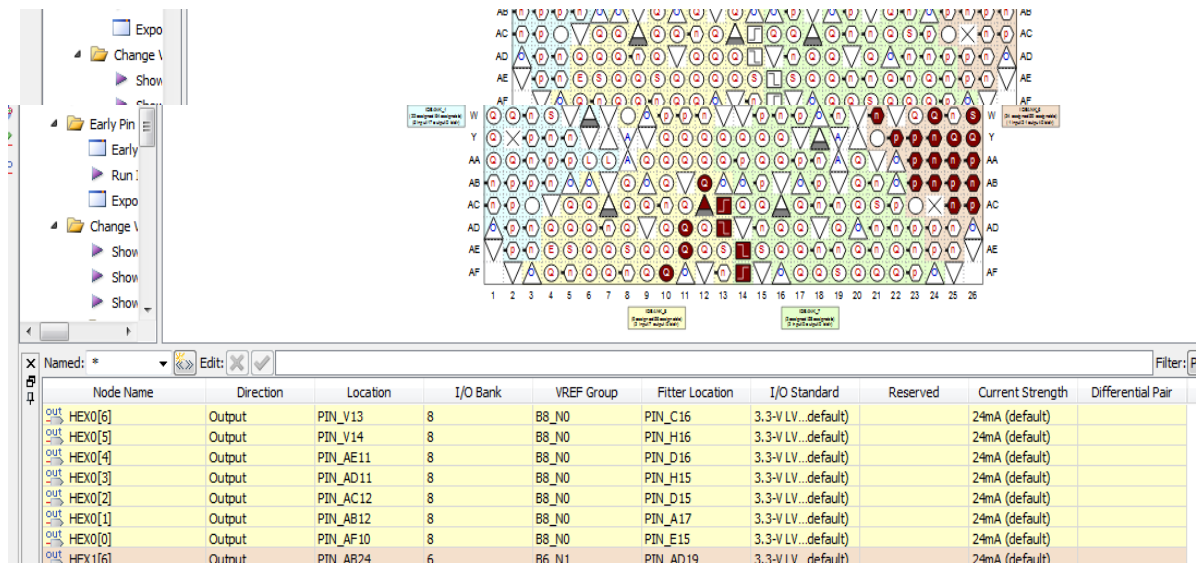
PIN Name	Description
SW(15 downto 8)	8-bit ASCII code for InputB (valid ASCII codes include: "0-9", "a-f", and "A-F")
SW(7 downto 0)	8-bit ASCII code for InputA (valid ASCII codes include: "0-9", "a-f", and "A-F")
HEXi(6 downto 0)	7 segment display signals for the 8 displays (HEX7 ... HEX0). Each display is driven by 7 signals (6 downto 0). Note that the magnitudes of each number are displayed as hex digits, not decimal digits. Assignments of specific displays are provided below.
HEX7&6	Displays the ASCII code (in hex) for InputB
HEX5&4	Displays the ASCII code (in hex) for InputA
HEX3	Displays the 4-bit unsigned equivalent of the ASCII digit of InputB. Invalid ASCII codes (see above) are indicated as an error on HEX3 by only turning on the top segment of the display.
HEX2	Displays the 4-bit unsigned equivalent of the ASCII digit of InputA. Invalid ASCII codes (see above) are indicated as an error on HEX2 by only turning on the top segment of the display.
HEX1&0	Display the 8-bit addition result. If the most significant hex digit of the sum (HEX1) is zero, then HEX1 should be turned off ("blanked") instead of displaying "0".



To Do: (you should work independently on this lab)

1. Draw a detailed block diagram of the system based on the requirements listed above. Although PC tools are available, you are encouraged to do this one in pencil on a piece of paper so you can quickly draw and modify the diagram. External ports and internal signals should be labeled on your diagram and those labels should match the port/signal names in your VHDL code. Also include size indicators for connections with multiple wires. Start from the inputs and identify blocks of logic that will be needed to combine or transform those inputs into the outputs desired.

- Download the template.vhd VHDL file and the sevenseg_pkg.vhd file from the network library drive into a new design folder. In addition, download the file “DE2_pin_assignments_orig.csv” and make a second copy of that file in the same folder. Rename the template vhd file and the copied csv file appropriately. Note: you may modify the copied .csv file, but leave the orig.csv unchanged as a reference.
- Edit your vhd file: Add appropriate VHDL statements and comments, especially where you see the words “Fill_In”. The entity is complete and the structure of the architecture is complete; some additional statements and processes have to be added to the architecture to complete the system. Quartus or Notepad++ are the recommended tools for this activity. Begin by implementing only the logic needed to properly drive HEX7 and HEX6 from SW15..8. Some example statements are provided which you can modify for your needs. Unused outputs (HEX5..0) can be driven with either a fixed logic ‘0’ or ‘1’. After compiling and testing this simple version, you can then add additional capabilities to your design.
- Create a new project in Quartus in your new design folder. Specify the target device as the Cyclone II: **EP2C35F672C6** and add your two *.vhd files to the project. Hint: Keep your folder, project, and top-level entity names the same with no spaces or special characters, if possible. In this case, use the entity name, “HWLab1”.
- Setup the pin assignments for your design. Since the wire traces (from the FPGA to the various components) on the DE2 board are fixed, you must know and then map your port names to the appropriate pins on the FPGA chip. The easiest way to do this is to import the pin assignments you need from a file that contains all of the FPGA pin assignments for the DE2 board design. First, edit your .csv file using Notepad++ (**not Excel since Excel adds unwanted characters to the file**) and remove the rows of the spreadsheet for any pin not needed for your design. Include all pins needed for the complete design, not just those for HEX7 and HEX6. When finished, you should have an entry in the pin file for each port signal found in the entity’s port listings and you should have no other signals in the file that are not found in your entity listing. Do not make any other changes to the spreadsheet (e.g. leave the header information at the beginning of the file). To use this file as the pin assignment list for your design, select the menu item “Assignments/Import Assignments...” in Quartus. Locate your file using the filename navigator and then select “Ok”. It should report the success (or lack of success) in importing the pin assignments.
- Compile your design. If the compilation is successful, then you can open the tool “Assignments/Pin Planner” to view and confirm the current pin assignments for your project. Make sure the “Location”



The screenshot shows the Quartus Pin Planner interface. On the left is a file explorer showing the project structure. The main area displays a grid of pin locations (A0-AF, 0-25) with various symbols indicating pin types and assignments. Below the grid is a table with the following columns: Node Name, Direction, Location, I/O Bank, VREF Group, Fitter Location, I/O Standard, Reserved, Current Strength, and Differential Pair.

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Differential Pair
HEX0[6]	Output	PIN_V13	8	B8_N0	PIN_C16	3.3-V LV..default		24mA (default)	
HEX0[5]	Output	PIN_V14	8	B8_N0	PIN_H16	3.3-V LV..default		24mA (default)	
HEX0[4]	Output	PIN_AE11	8	B8_N0	PIN_D16	3.3-V LV..default		24mA (default)	
HEX0[3]	Output	PIN_AD11	8	B8_N0	PIN_H15	3.3-V LV..default		24mA (default)	
HEX0[2]	Output	PIN_AC12	8	B8_N0	PIN_D15	3.3-V LV..default		24mA (default)	
HEX0[1]	Output	PIN_AB12	8	B8_N0	PIN_A17	3.3-V LV..default		24mA (default)	
HEX0[0]	Output	PIN_AF10	8	B8_N0	PIN_E15	3.3-V LV..default		24mA (default)	
HEX1[6]	Output	PIN_AB24	6	B6_N1	PIN_AD19	3.3-V LV..default		24mA (default)	

Figure 1. Pin Planner showing assigned "locations" for the pins (after importing).

column is properly filled in for all inputs and outputs of the design.

7. Open up the Quartus “Programmer” tool from the “Tools” menu. Select “Hardware Setup” and make sure the USB-Blaster is selected. Then select “Add File...” to choose the *.sof file from your project. After ensuring that the check box for “Program/Configure” is selected, select the “Start” button to download your file.
8. Test your design. Consider which tests would “stress” your design and validate its function. What are some “edge” cases that should be tested?
9. If the design works, update your VHDL code for the following additional capabilities and test your design after adding each one:
 - a. drive HEX5 and HEX4 based on SW7..0
 - b. drive HEX3 and HEX2 with their respective numeric values
 - c. drive HEX1 and HEX0 with the addition result
10. Compile your complete design using Quartus to make sure there are no errors in your VHDL. Screen capture all warnings generated from the compilation. If there are ‘inferred latch’ or ‘sensitivity’ warnings, resolve them, there are other expected warnings that can be ignored for this lab.
11. Simulate your design using modelsim and the provided testbench file and do file. The testbench file provides a VHDL testbench entity that will exhaustively test your design, something you could not do with the actual circuit in a reasonable amount of time. You should add your design file, the sevenseg_pkg.vhd file, and the TestBench.vhd file to a new Modelsim project in the same folder as your Quartus project. Compile each file, but note that the order of compilation is important (lowest levels of the design first). After compilation is completed, type the command “do TestHWLab1.do” in the command window. This should run the testbench on your design. Warnings that occur at 0 ps are not uncommon and can be ignored. If any other warnings or errors are reported, then you will need to fix your design and re-run the testbench. Take a screen capture of all warnings or errors that occurred on your final simulation. Review a few spots in your simulation waveform screen and take a screen capture of at least one of them. Screen capture the transcript window with any run-time warnings that are shown.

Write Up:

Your writeup for this lab should include:

- a maximum 1-page cover sheet summarizing what you did, what you learned, and how you tested your design on the actual DE2 board – list some specific tests you tried and why you think they were important to use;
- a 1-page system block diagram (well-labeled) – can be a photo of your legible hand-drawn diagram since you may want to refer to this block diagram for the next lab;
- a screenshot of the Quartus compiler warnings for your final design version;
- a screenshot that shows the Modelsim command window after the running of the testbench on your design (any run-time warnings would be shown);
- a screenshot taken from your Modelsim simulation waveforms showing one or two add operations completing correctly;
- and a printout of your well-commented (explains “why” and not just “what”) VHDL code (the same conventions from the assembly lab code printouts still hold for VHDL code printouts) – note that poor commenting and poorly-presented code printouts may cost you points. Use a **highlighter** to mark any portions of the file that you changed from the original template.