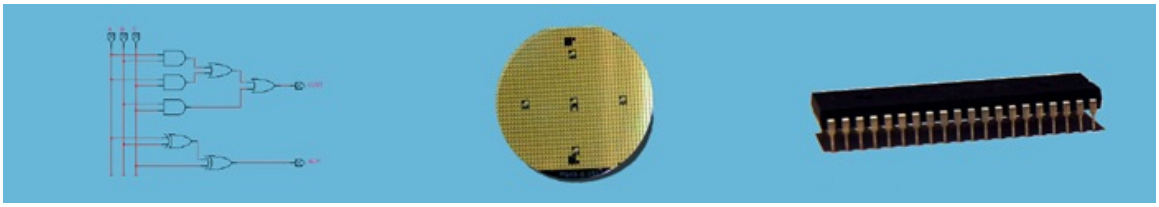
CE1900 WEEK 6 LABORATORY EXERCISES



LABORATORY SUMMARY

These laboratory exercises implement some of the basic computer arithmetic circuits studied in lecture by using the Quartus toolset to design, simulate, and program the circuits into programmable logic devices on the Altera DE1 laboratory board. The DE1 hosts one Altera Cyclone II field programmable gate array. The laboratory board is a fixed platform and the chips should not be removed. A set of switches and wire headers are included to allow input voltages to be connected. A set of light-emitting diodes (LEDs) and 7-segment displays are included to allow system results to be displayed to the user. This week, students practice VHDL description, schematic blueprints, simulation, and implementation.

Students will complete three design tasks:

- **Enter** a VHDL description for a full-adder component.
- **Enter** a VHDL description for a 7-segment display decoder.
- **Enter** the schematic diagram of an RCA4 component.
- **Enter** the top-level system schematic diagram.
- **Implement** the top-level system using the Cyclone II.
- **Test** the system in the laboratory.

The laboratory exercises reinforce these CE1900 learning objectives:

Use schematic entry to design and simulate arithmetic circuits.
Use VHDL description to design and simulate arithmetic circuits.
Implement completed schematic designs in programmable logic devices.

Print this document and **bring** it to lab with the preliminary activities completed in the spaces provided.
Hint: save your color print cartridge by selecting "black-and-white" printing from your print options.

REMEMBER

Complete all homework and preliminary lab exercises ***before you come to lab***.
Review the in-class laboratory exercises to preview the work you will do in the lab.

CE1900 WEEK 6 LABORATORY EXERCISES

PRACTICE PROBLEMS

Take a break from practice exercises this week. Instead, focus on reading the on-line VHDL tutorials, working the extensive pre-laboratory exercises, and reviewing your lecture notes during your study times.

PRELIMINARY LAB EXERCISES

READ THESE DIRECTIONS CAREFULLY AS YOU COMPLETE THE PRELAB WORK.

1. **Create** a new Quartus project called **lab6**. This is the **only project** you will create during this lab.
2. **Create** a new VHDL description file for entity FA inside your project.
    a. FA is the standard name for a full adder.
    b. **Implement** the full adder architecture using with-select statements for COUT and S.
    c. **Save** the file as fa.vhd. **Note** that Quartus will try to autoname the file lab6 because that is the project name. **Override** this autoname with fa.vhd.
    d. **Do not** use Processing → Start → Analysis and Elaboration this week to check your VHDL files because you cannot "elaborate" a project until it is completely built. This week, you will create four different design files before the project is completely built!
    e. **Use** Processing → Analyze Current File to check your VHDL. **Correct** your syntax errors if needed.
    f. **Use** File → Create/Update → Create Symbol File for Current File to generate a schematic diagram symbol that you will use later in the laboratory. A symbol file is a blueprint symbol.
3. **Create** a new VHDL description file for entity SEG7DECODE.
    a. This entity decodes a 4-bit binary number into left and right 7-segment displays. The input binary number 0100 would be shown as 04 on the 7-segment displays. Similarly, the number 1101 would be displayed as 13 on the 7-segment displays.
    b. This entity uses a 4-bit input bit_vector named ABUS. VHDL bit vectors are busses. Declare the bus using **ABUS : in bit_vector(3 downto 0)**
    c. This entity produces a 14 bit output bit_vector named SEGBUS. Declare the bus using **SEGBUS : out bit_vector(13 downto 0)**
    d. The upper 7 bits of SEGBUS correspond to segments a-g of the left seven segment display.
    e. The lower 7 bits of SEGBUS correspond to segments a-g of the right seven segment display.
    f. **Implement** the SEG7DECODE architecture using a single with-select statement for SEGBUS. It would look like this:

```
-- DE1 7SEG LEDs turn on with 0, off with 1
with ABUS select
SEGBUS <= B"00000010000001" when B"0000",
          B"00000011001111" when B"0001",
          …,
```

    g. **Save** the file as seg7decode.vhd. **Note** that Quartus will try to autoname the file lab6 because that is the project name. **Override** this autoname with seg7decode.vhd.
    h. **Use** Processing → Analyze Current File to check your VHDL. **Correct** your syntax errors if needed.
    i. **Use** File → Create/Update → Create Symbol File for Current File to generate a schematic diagram symbol that you will use later in the laboratory.
4. **Create** a new schematic diagram.

a. **Draw** the schematic for the 4-bit ripple carry adder shown in Figure 1.
b. **Save** the file as **rca4** because the standard schematic name for a ripple-carry adder is a block labeled RCA. **Note** that Altera will attempt to autoname the file to lab6 because the project ***must*** have a top-level entity called lab6. **Override** that autoname with **rca4**.
c. **Note** that as a result of creating the symbol file in an earlier step, the full adder component is available for placement under the "Project" library in the symbol dialog box. Just double-click as normal, expand the "Project" library if needed. Your symbol will be called FA. In Figure 1, it is called fulladd just to illustrate that the library symbol will depend on what you name the VHDL entity!
d. **Note** from Figure 1 how input and output **busses** are used to make connecting the inputs and outputs significantly easer. **Busses** are collections of wires holding a number as binary voltages. Thus, **bus** is simply another term for **bit_vector**. Entering the bus isn't hard – here's how:

    i. **Place** the input or output connector.
    ii. **Name** the connector using **bus notation**. For example: A[3..0]. Note that the range inside the bus notation determines exactly how many signals are contained in the bus. In this case there are four signals named A[3], A[2], A[1], and A[0] collected together to form the **A bus**.
    iii. **Pull** the wire from the connector using the mouse or trackpad. It will be the appropriate width for a bus. **Note** from the diagram that busses are wider than single wires.
    iv. **Connect** single wires to the bus by pulling from the component to the bus. **Note** that the wire will be the single wire width. **Highlight** the single wire and type the name – for example, A[3].

e. **Complete** Processing → Analyze Current File to verify that all wires are connected. **Correct** wiring errors if necessary.
f. **Make** a block-diagram symbol for the component with File → Create/Update → Create Symbol Files for Current File.

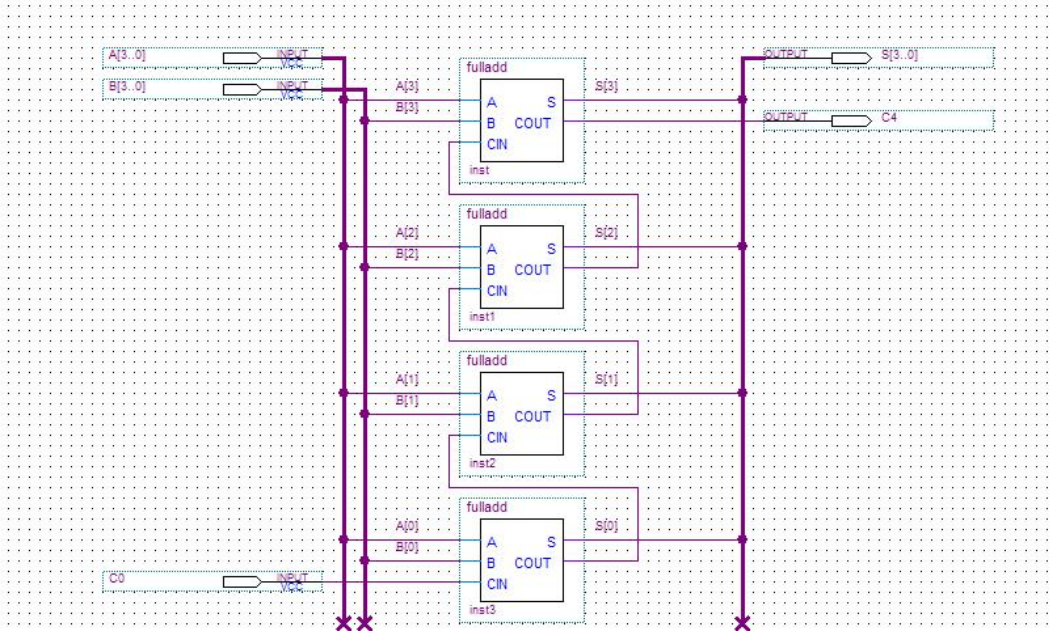


Figure 1: The RCA4 schematic

5. **Create** a new schematic diagram.
   a. **Draw** the top-level schematic for the **lab6** project shown in Figure 2. **Note** that input C0 has been "grounded" – attached to 0V permanently. You can find the "ground" symbol by typing GND into the library component-by-name search box.
   b. **Save** the file as **lab6. Note** that Altera will attempt to autoname the file to lab6 because the project ***must*** have a top-level entity called lab6. **Accept** that autoname.
   c. **Complete "**Processing → Start compilation" to verify that all wires are connected and complete the files needed for simulation. **Correct** wiring errors if necessary.
   d. **Simulate** the **lab6** entity to verify correct operation. **Note** during signal selection that you can choose just the bus name (A for example) rather than each individual bus signal – the whole bus will be added. **Overwrite** count values on the **A bus** and **B bus** and run the simulation. **Verify** that the **SEGBUS bus** output is correct. For example, 0000 + 0000 = 0000, 0001 + 0001 = 0010, 0010 + 0010 = 0100, etc. verifying that each sum produces the correct segments lighting on the seven-segment display. **Examine** each component if the simulation fails. **Note** that a double-click on the component pushes into the component's internal circuit or VHDL description. **Correct** errors and re-simulate.

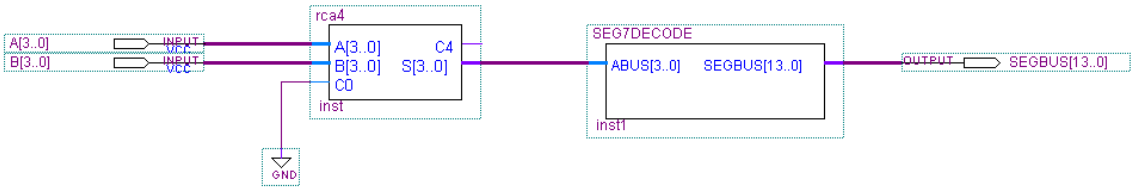


Figure 2: The lab6 project schematic

6. **Make** device and pin assignments for laboratory testing. **Use** the Cyclone II, the 7-segment displays, and the toggle switches available on the DE1 board. **Use** switches 7 down to 4 as A[3], A[2], A[1] and A[0]. **Use** switches 3 down to 0 as B[3], B[2], B[1], and B[0]. **Note** that the DE1 manual is available as a PDF on the lab page of the course website. Or, if you are a computer engineering student that has purchased the DE0 board early, use the appropriate chip, displays, and switches from that board.

LABORATORY EXERCISES

Each student must complete these exercises during the laboratory period in week 6.

1. **Take** the weekly quiz given by the instructor.
2. **Check** out a DE1 board from EECS Tech Support.
3. **Compile** the design and **program** it into the CYCLONE II FPGA.
4. **Test** your design.
5. **Demonstrate** to the instructor. The instructor will complete the signature block at the end of this document. The signature block serves as proof that you have completed the laboratory assignments. **Keep** this document in your binder for CE1900 in case you need to refer to it in future work or in case the instructor asks to see the signature block on a different date.
6. **Estimate** the total amount of time you spent working on the pre-lab and lab

   exercises. **Record** that time in minutes: _____

   NOTE: Your lab grade will be docked if you do not complete the time logging activity. Also note that each student must complete the time logging activity.

CE1900 WEEK 6 LABORATORY EXERCISES

FOR INSTRUCTOR USE ONLY: DO NOT WRITE IN THIS TABLE

ITEM	COMPLETED	SCORE
Prelab work		
Demonstrates understanding of Quartus		
Lab demonstration		

Instructor Signature: _____