EE365 – Advanced Digital Circuit Design
Fall 2003
Design Project 2 – 2-Digit BCD Adder

Project Overview:

Your task is to design and implement a 2-digit binary coded decimal adder. Given a system-level hierarchal design, you will be required to design each component and implement them in VHDL. You will also be required to test your design. As a component of the testing procedure, you will be required to develop an automated testbench.

Project Background:

Binary coded decimal numbers are a type of decimal code where each digit in a base-10 numbering system is encoded to a 4 bit-wide binary number. For instance, the number 45 in base 10 would be BCD-represented as 01000101 (0100 = 4, 0101 = 5). For a more complete description of BCD number, please refer to Wakerly, p. 48-50.

An adder (or more specifically, a full adder), takes three inputs (the two operands and a "carry in") and produces two outputs (the sum and a "carry out"). For more information on the equations behind a full adder, see Wakerly, p. 431. The presence of "carry in" (CIN) and "carry out" (COUT) signals allow full adders to be cascaded with each other; for the adder corresponding to bit N of the input, the CIN is connected to the COUT of the N-1 adder, and the COUT is connected to the CIN of the N+1 adder. This produces a cascade effect, allowing an adder to be very easily generalized to N bits.

For this project, you will be required to develop a 4-bit full adder in order to add digits of corresponding place together. You will also be required to cascade these 4-bit adders in order to create a 2-digit wide adder. Note that even though the inputs to each 4-bit adder will be BCD-compliant, the same cannot be said for the outputs. Therefore, it will be necessary to develop a logic component that converts from regular 4-bit binary numbers to a BCD-compliant number.

The system-level design for the project shall include 2 4-bit adders (each made up of 4 1-bit full adders), and 2 binary-to-BCD converters.

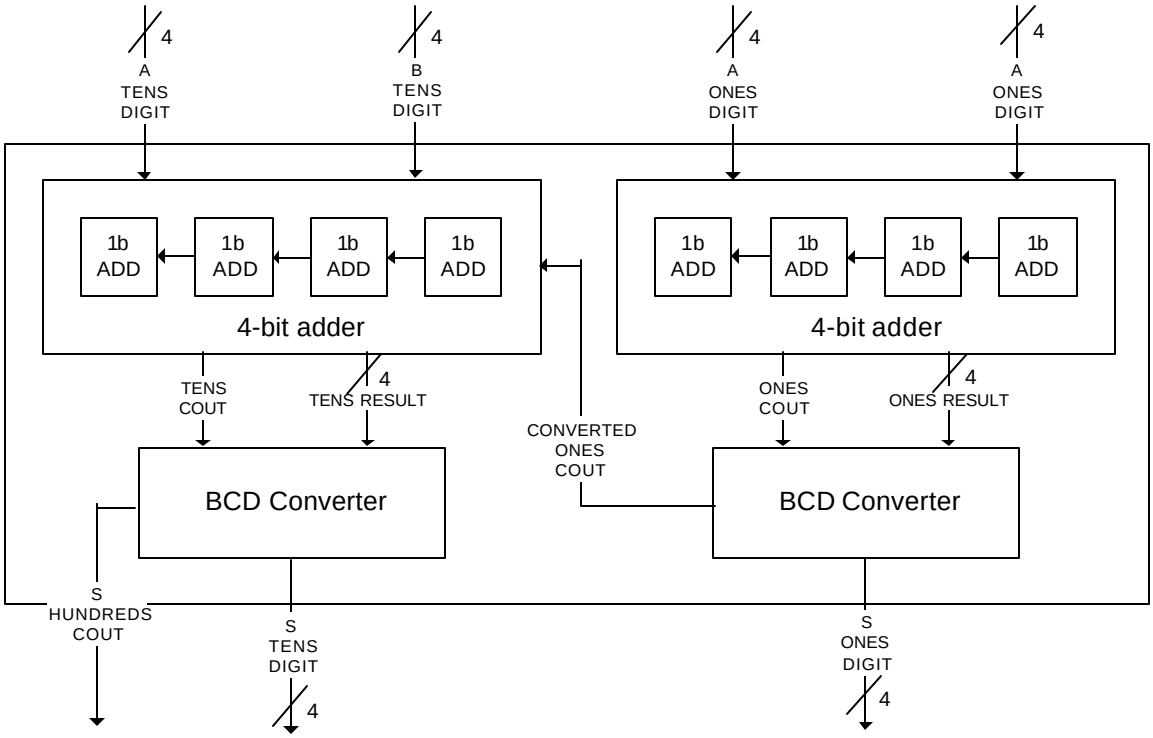

Figure 1: Top-level design drawing for Project 2.

The system shall take inputs A and B, which are two-digit BCD numbers, and add them together to produce S, also a two-digit BCD number. In the case where A + B > 99, the "hundreds" place COUT shall be asserted.

Project Specifications:

For many parts of this project, your group is free to design each component to use any method available to you in VHDL. However, particular components shall have constraints placed on their design:

1) For the 4-bit full adder, your group must produce a structural design that utilizes a "for-generate" loop to instantiate the 1-bit adders needed.
2) The BCD converter module will require two different architectures to be developed:

    a. A "look-up table"-based architecture, where the module will take certain inputs and decode them to produce the necessary outputs. This is similar to a "case-select" structure in C/C++.
    b. An architecture based on the add-6 strategy presented by Wakerly on p. 50. In this case, it will probably be useful to reuse your 4-bit full adder.

    The top level module will be required to select a particular configuration (entity-architecture pair) to be used when simulating the design. For more information about specifying configurations, refer to the following website: http://www.csee.umbc.edu/help/VHDL/design.html

3) The testbench used to test the top-level module must be automated; instead of manually testing every combination of inputs and checking the waveform afterward to ensure that they are correct, you will read in test input values from a text file, along with the expected output. You will then use assertion checking to determine whether the output from your design matches what was expected. An example of a project using this type of test bench is posted on the course website. Also, take a look at the following website for more help: http://www.emba.uvm.edu/~jswift/uvm_class/notes/files.html

   An example of this input text file (note that this example is not intended to be complete by any means):

```
24 31 055       -- 24 + 31 = 55
83 47 130       -- 83 + 47 = 130
```

For this project it is not necessary to target particular devices or include timing information for your design; therefore, only functional simulations will need to be performed.

When you turn your project in, your report should include the following:

1) A brief description of the project, as well as a description of your component designs and your method of implementation.
2) A description of your test plan (Remember, your test plan should verify that your automated testbench also works correctly. This is not hard, but it should be shown explicitly in order to increase confidence in your testing methods).
3) Timing diagrams for the functional simulations, along with an explanation (in words) of what the timing diagrams show in terms of design verification. If the simulator produces any text output in the ModelSim console window (i.e. notice of an assertion failure), please copy and paste that into your report as well.
4) A copy of the VHDL code for each module and the testbench, as well as a copy of the text file used as an input to your testbench for the simulations that you are submitting waveforms.
5) Answers to the following questions:
   a. Is the idea of a "look-up table" realistic? If you were to implement a look-up table in a design you were physically producing, what kinds of devices would you target?
   b. Note that we represent the hundreds digit of the output S with a single bit. What constraints in our design specification allow us to represent S in this way (relate the possible inputs to the possible outputs).

For extra credit, your group may attempt to generalize the top level module so that the testbench can instantiate an N-digit adder by changing a "generic" parameter within the component definition. However, it is better to submit a 2-digit adder that works than an N-digit adder that does not work!