EE120A Logic Design
Department of Electrical Engineering
University of California – Riverside

Instructor: Roman Chomko

LABORATORY # 6*
L A B M A N U A L

Datapath Components - Adders

---

* EE and CE students must attempt also to implement the assignment described in "Lab 6 Optional" Manual. It is, however, NOT THE REQUIREMENT for this laboratory and WILL NOT be graded if submitted. It provides information on usage of LED displays of Basys boards.

Objectives

1. Design of adders, synthesis and implementation;
2. Design of special purpose registers

Equipment

- PC or compatible
- Digilent's Basys Spartan-3E FPGA Evaluation Board

Software

- Xilinx ISE Design Software Suite 10.1
- ModelSim XE III modeling software
- Digilent's Adept ExPort Software

Parts

- N/A

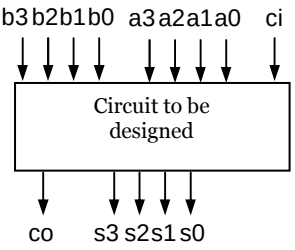
Background - Adders Design

In this FPGA application development assignment, we will implement a calculator that does just one thing – adds two 4-bit numbers.

4-bit carry-ripple adder

An N-bit adder adds two N-bit numbers plus a carry-in bit, resulting in an N-bit sum and a carry-out bit. A block diagram of a 4-bit adder appears in **Figure L6-1**.

Figure L6-1. Block diagram of a 4-bit adder.

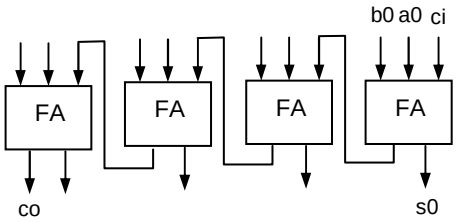


Although we could design a 4-bit adder's circuit using the combinational logic design process, the resulting circuit would be rather large. Instead, we can use a more clever design approach, which mimics how numbers are added by hand.

Addition by hand is done one column at a time. The rightmost column would add b0+a0+ci, to generate the sum bit s0 and a carry bit for the next column, which we'll call ci. We can thus first design a component, called a *full-adder*, which adds three input bits a, b, and c, and generates sum and carry output bits s and co. Using the combinational logic design process, we find that s = a xor b xor c, and co = bc + ac + ab. Design a full-adder circuit implementing those equations.

Now connect four such full-adders to create a 4-bit adder, as shown in **Figure L6-1**. The figure does not show all the connections of the inputs and outputs to the full-adders, but you should be able to determine those connections easily.

Figure L6-2. General structure of a 4-bit adder built from 4 full-adders.



Simulate the system and observe the outputs. Try adding some small numbers, like 0000+0000+0 (which should result in 0 0000) and 0001+0001+0 (which should equal 0 0010). Also try adding some larger numbers, like 1111+1111+0 (which should equal 1 1110), and 1111+1111+1 (which should equal 1 1111). Is it possible for two 4-bit numbers and a carry-in to result in a number too big to represent using 4 sum bits and a carry-out bit?

Note that the above adder assumes the inputs are unsigned numbers (i.e., the inputs are *not* in two's complement form).

Specification

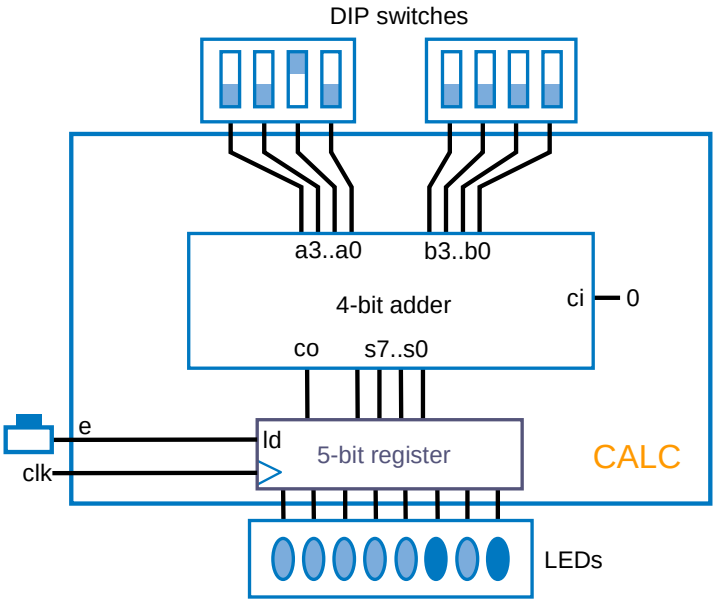


Figure L6-3. Adder Structure and Basys Board implementation hint

Implement a 4-bit adder system on the Basys development board. Use two 4-position DIP switches for the two 4-bit inputs, a single switch for the carry-in input, and five LEDs for the give outputs. Then try different combinations of the inputs and observe the outputs. Use one of the buttons to load the results to the LEDs. See **Figure L6-3** for details.

Implementation Utilities and Hints

The following CLK configuration must be used in the constraints file:

```
# clock pin for Basys Board
NET "CLK" LOC = "p54"; # Bank = 2, Signal name = CLK1
```

Demonstration

Demostrate that the application performs according to specs.

Procedures

1. Xilinx ISE Design and Synthesis environment;
2. Creation of Configuration files;
3. Usage of Adept ExPort download software;

Presentation and Report

Must be presented according to the general EE120A lab guidelines posted in iLearn.

Prelab

1. Review Lecture 14;
2. Try to answer all the questions, prepare logic truth tables, do all necessary computations