UNIVERSITY OF CALIFORNIA, DAVIS
Department of Electrical and Computer Engineering

EEC180A **DIGITAL SYSTEMS I** **Spring 2000**

LAB 9: ARITHMETIC CIRCUIT DESIGN

The purpose of this lab is to use the Altera circuit design software to investigate the ripple-carry adder and the carry lookahead adder designs.

Hardware Required:

None. You will not build this circuit. You will enter and simulate the design using the Altera design tools.

Part I. Carry Lookahead Adder Modules (Background)

You will design some basic circuit modules that can be used to build a hierarchical 16-bit Carry Lookahead Adder. The basic modules are described below.

1. Bit-Adder with Propagate and Generate outputs

The Bit-Adder module has inputs A_i, B_i and C_i (carry-in) and outputs S_i (sum), P_i (carry propagate), and G_i (carry generate) as shown in Figure 1.

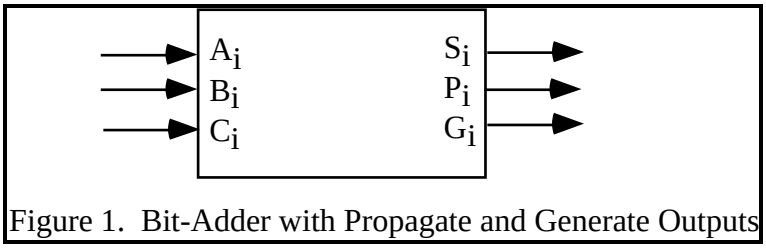

Figure 1. Bit-Adder with Propagate and Generate Outputs

The sum output, S_i, is equivalent to the sum output of a Full-Adder. However, instead of producing a carry-out similar to the Full-Adder, the Bit-Adder module produces two new signals - the carry generate and the carry propagate. The carry propagate signal will be 1 if A_i and B_i would propagate a carry-in of 1 to a carry-out of 1. The carry generate signal will be 1 if A_i and B_i would generate a carry-out of 1 independent of C_i. Thus, the outputs of the Bit-Adder module can be expressed as follows:

$$P_i = A_i \oplus B_i$$

$$G_i = A_i * B_i$$

$$S_i = A_i \oplus B_i \oplus C_i = P_i \oplus C_i$$

2. Four-bit Lookahead Carry Generator (LCG) with Group Propagate and Generate Outputs

In a Carry Lookahead Adder, each C_i signal except C_0 is produced by the Lookahead Carry Generator module and is based on P_{i-1} - P_0, G_{i-1} - G_0 and C_0. Thus, a four-bit LCG module would have inputs and outputs as shown in Figure 2. The LCG module can be used along with four Bit-Adder modules to build a 4-bit Carry Lookahead Adder. The group carry propagate, P, and carry generate, G, outputs indicate whether the inputs A_3-A_0 and B_3-B_0 would propagate or generate a carry. The group carry propagate and carry generate functions can be expressed in terms of the input signals P_3-P_0 and G_3-G_0. For example, the group propagate signal will be a 1 if each of the input P_i signals is a 1. Examples of some of the output signal equations are shown below. By following the example equations given for C_1 and C_2, you should be able to derive the equations for C_3 and C_4.

$P = P_3*P_2*P_1*P_0$

$G = G_3 + P_3*G_2 + P_3*P_2*G_1 + P_3*P_2*P_1*G_0$

$C_1 = G_0 + P_0*C_0$

$C_2 = G_1 + P_1*C_1 = G_1 + P_1*G_0 + P_1*P_0*C_0$

$C_3 = f(G_2, G_1, G_0, P_2, P_1, P_0, C_0)$

$C_4 = f(G_3, G_2, G_1, G_0, P_3, P_2, P_1, P_0, C_0)$

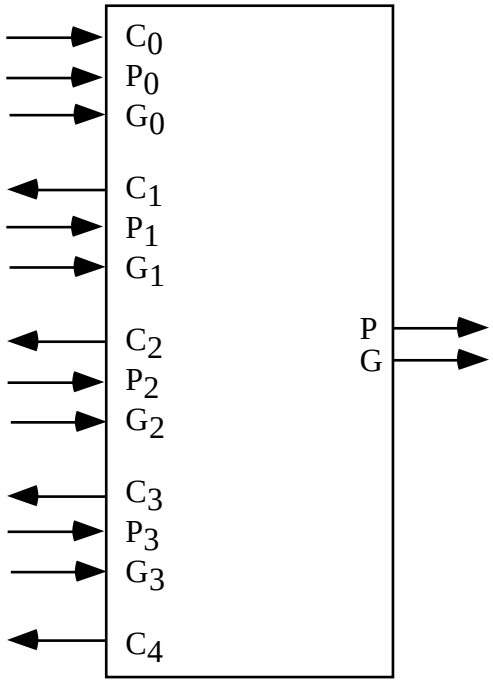

Figure 2. Four-bit Lookahead Carry Generator with Group Propagate and Generate Outputs

3. Four-bit Carry Lookahead Adder (CLA4)

A four-bit Carry Lookahead Adder can be designed using four Bit-Adder modules and a four-bit LCG module. The inputs of the CLA4 module are A_3-A_0, B_3-B_0 and C_{in}, while the outputs are S_3-S_0, P, G and C_4.

4. 16-bit Carry Lookahead Adder (CLA16)

A 16-bit Carry Lookahead Adder can be designed using four CLA4 adders and a four-bit LCG module with group propagate and generate outputs. The inputs of the 16-bit Carry Lookahead Adder are A_{15}-A_0, B_{15}-B_0, and C_{in}, while the outputs are S_{15}-S_0, P, G and C_{16}.

Part II. Design Procedure

A. Carry Lookahead Adder Design

1. Using the Altera Graphic Editor, design a Bit-Adder with Propagate and Generate outputs as described in Part I, Section 1. Perform a functional simulation to be sure it works correctly.

2. Design and verify the Four-bit Lookahead Carry Generator module as described in Part I, Section 2.

3. Design and verify, by timing simulation, the four-bit Carry Lookahead Adder (CLA4) described in Part I, Section 3. (See the hints on hierarchical design in Altera.)

4. Design and verify, by timing simulation, the 16-bit Carry Lookahead Adder (CLA16) described in Part I, Section 4. Note that you do not need to do an "exhaustive" simulation covering every input possibility for the 16-bit adder. Verify enough cases to satisfy yourself that the adder works correctly. Also, use the following two test vectors:
   - 0000+FFFF with Cin=1
   - 0F0F+0201 with Cin=0

   Record the two result values and measure the output delays in the timing simulation.

5. Run the Timing Analyzer on the 16-bit Carry Lookahead Adder. With the CLA16 project open and the CLA16 design compiled with the **Timing SNF Extractor** option turned on, select the Timing Analyzer tool from the MAX+plusII pull-down menu. Run the Delay Matrix tool and print the result.

6. Identify the critical path in the 4-bit and 16-bit Carry Lookahead Adders. List the critical paths and show them on your schematic in color. How does the size of the adder affect the delay? If possible, give a function of the delay in terms of N, the size of the adder.

B. Ripple-carry Adder Design (Optional)

1. Using the Altera Graphic Editor, design a Full-Adder as described in Ch. 20 of the Roth text. The equations given in the text (p. 526) are:

$$s_i = c_i \oplus x_i \oplus y_i$$

$$c_{i+1} = c_i (x_i + y_i) + x_i y_i$$

   Perform a functional simulation to be sure your full-adder works correctly.

2. Design and verify a four-bit Ripple-Carry adder (RCA4) using full-adder modules. Perform a timing simulation and observe the typical delay from input to output.

3. Design and verify by timing simulation the 16-bit Ripple-Carry Adder (RCA16) using four RCA4 modules. Note that you do not need to do an "exhaustive" simulation covering every input possibility for the 16-bit adder. Verify enough cases to satisfy yourself that the adder works correctly. Also, use the following two test vectors:
   - 0000+FFFF with Cin=1
   - 0F0F+0201 with Cin=0

   Record the two result values and measure the output delays in the timing simulation.

4. Run the Timing Analyzer on the 16-bit Ripple-Carry Adder. With the RCA16 project open and the RCA16 design compiled with the **Timing SNF Extractor** option turned on, select the Timing Analyzer tool from the MAX+plusII pull-down menu. Run the Delay Matrix tool and print the result.
5. Identify the critical path in the 4-bit and 16-bit Ripple-Carry Adders. List the critical paths and show them on your schematic in color. How does the size of the adder affect the delay? If possible, give a function of the delay in terms of N, the size of the adder.

Lab Report

The following items must be included in your lab report:

- A TA verification sheet with the CLA4 and RCA4 (optional) timing simulations verified.
- Altera schematics for the following circuits:
  1. Bit-Adder with Propagate and Generate outputs
  2. Lookahead Carry Generator module
  3. Four-bit Carry Lookahead Adder (CLA4)

4. 16-bit Carry Lookahead Adder (CLA16)
5. Full-Adder (Optional)
6. Four-bit Ripple-Carry adder (RCA4) (Optional)
7. 16-bit Ripple-Carry Adder (RCA16) (Optional)

- Altera Timing Analyzer Delay Matrix outputs for the CLA16 and RCA16 (optional) circuits.
- Analyze the critical path for the CLA4, CLA16, RCA4 (optional) and the RCA16 (optional) circuits. Use your schematics to determine the gate count of the longest path through each adder. Highlight the critical paths in colors on your schematics. Find a function for the delay of the adder in terms on its size, N.
- Explain what the following two test vectors measure
  - 0000+FFFF with Cin=1
  - 0F0F+0201 with Cin=0

  Compare the delay values for the CLA16 and the RCA16 for these vectors. (Optional)

Altera Hints

The Altera design tools have good support for hierarchical designs. Once you have created a circuit, you can easily make a symbol for it so that it can be used as a module in another design. To create a symbol, select **Create Default Symbol** (File Menu). You will then be able to place that component on another schematic.

On many of these schematics, it is easier to make connections by name than by drawing nets. For example, if two nets are labeled P0 on the same schematic (i.e. same level of hierarchy), the Altera compiler considers those nets to be connected.

Bus notation can be used to reduce the number of input and output components needed. For example, a single input pin can be labeled **A[15..0]** and the net attached to the pin can have the same label. Make sure to use the bus net (thicker line) rather than the signal net. The net type can be changed using the net pull-down box on the toolbar or by selecting Line Style (Options menu) and choosing the appropriate line. (The first entry is the signal line style and the second is the bus line style. Often the lines will not show up in the UNIX version of Max+plusII. However, you can still change the setting by knowing the positions of the different options.)

In order to generate specific input vectors for simulation, you can use the Overwrite > Group value... option after highlighting the desired time interval of a grouped signal.