Ripple Carry and Carry Lookahead Adders

1 Objectives

- Design ripple carry and carry lookahead (CLA) adders.
- Use VHDL CAD tools.
- Use hierarchical design techniques.
- Model and simulate combinational logic using VHDL.

2 Introduction

We will start by explaining the operation of one-bit full adder which will be the basis for constructing ripple carry and carry lookahead adders.

2.1 One-bit full adder

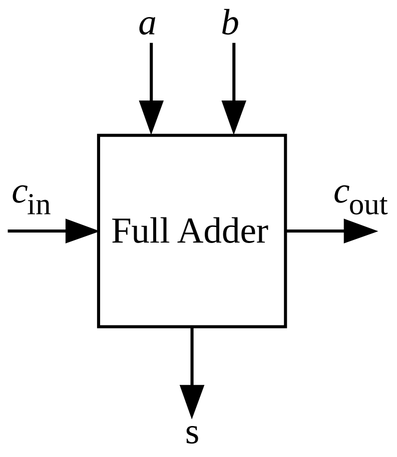

Figure 1: One-bit full adder.

A one-bit full adder is a combinational circuit that forms the arithmetic sum of three bits. It consists of three inputs $(a,\ b,$ and $c_{in})$ and two outputs $(s$ and $c_{out})$ as illustrated in Figure 1. The truth table of the full adder is listed in Table 1. The gate implementation of 1-bit full adder is shown in Figure 2.

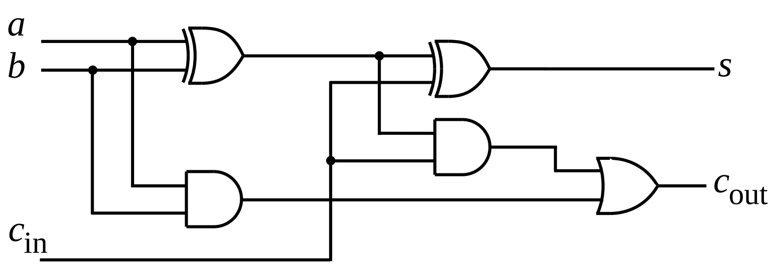

Figure 2: Gate implementation of full adder.

Table 1: Full adder truth table.

a	b	c_{in}	c_{out}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

2.2 Ripple carry adder

A ripple carry adder is a digital circuit that produces the arithmetic sum of two binary numbers. It can be constructed with full adders connected in cascaded (see section 2.1), with the carry output from each full adder connected to the carry input of the next full adder in the chain. Figure 3 shows the interconnection of four full adder (FA) circuits to provide a 4-bit ripple carry adder. Notice from Figure 3 that the input is from the right side because the first cell traditionally represents the least significant bit (LSB). Bits a_0 and b_0 in the figure represent the least significant bits of the numbers to be added. The sum output is represented by the bits s_0–s_3.

2.3 Ripple carry adder delays

In the ripple carry adder, the output is known after the carry generated by the previous stage is produced. Thus, the sum of the most significant bit is only available after the carry signal has rippled through the adder from the least significant stage to the most significant stage. As a result, the final sum and carry bits will be valid after a considerable delay.

Table 2 shows the delays for several CMOS gates assuming all gates are equally loaded for simplicity. All delays are normalized relative to the delay of a simple inverter. The table also shows the corresponding gate areas normalized to a simple minimum-area inverter. Note from the table

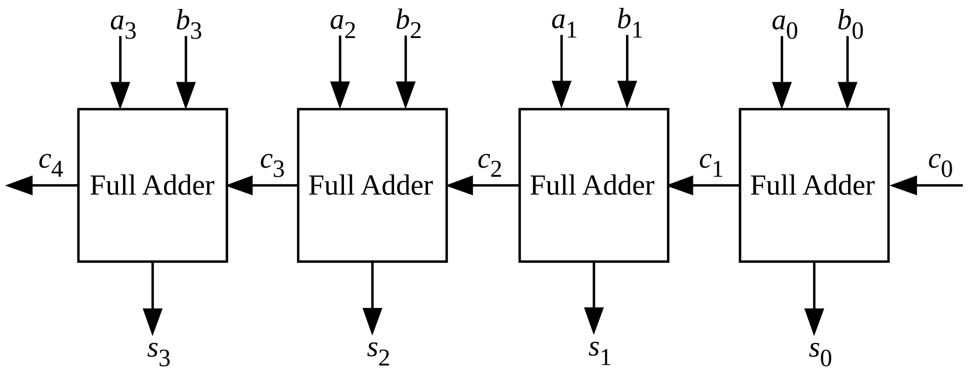

Figure 3: 4-bit full adder.

that multiple-input gates have to use a different circuit technique compared to simple 2-input gates.

Table 2: CMOS gate delays and areas normalized relative to an inverter.

Gate	**Delay**	**Area**	**Comment**
Inverter	1	1	Minimum delay
2-input NOR	1	3	More area to produce delay equal to that of an inverter
2-input NAND	1	3	More area to produce delay equal to that of an inverter
2-input AND	2	4	Composed of NAND followed by inverter
2-input OR	2	4	Composed of NOR followed by inverter
2-input XOR	3	11	Built using inverters and NAND gates
n-input OR	2	$n/3+2$	Uses saturated load ($n > 2$).
n-input AND	3	$4n/3+2$	Uses n-input OR preceded by inverters ($n > 2$).

Using Table 2 and the schematic of Figures 2 and 3, we can estimate the delays associated with the outputs of the ripple carry adder stages as indicated in Table 3. The delays are normalized relative to an inverter delay.

For an n-bit ripple carry adder the sum and carry bits of the most significant bit (MSB) are obtained after a normalized delay of

$$\text{Sum } s_{n-1} \text{ delay} = 4n+2 \quad (1)$$

$$\text{Carry } c_n \text{ delay} = 4n+3 \quad (2)$$

For a 32-bit processor, the carry chain normalized delay would be 131. The ripple carry adder can get very slow when many bits need to be added. In fact, the carry chain propagation delay is the determining factor in most microprocessor speeds.

Table 3: Delays for the outputs of a 4-bit ripple carry adder normalized to an inverter delay.

Signal	**Delay**
s_0, c_1	6, 7
s_1, c_2	10, 11
s_2, c_3	14, 15
s_3, c_4	18, 19

2.4 Carry lookahead adder (CLA)

The carry lookahead adder (CLA) solves the carry delay problem by calculating the carry signals in advance, based on the input signals. It is based on the fact that a carry signal will be generated in two cases: (1) when both bits a_i and b_i are 1, or (2) when one of the two bits is 1 and the carry-in is 1. Thus, one can write,

$$c_{i+1} = a_i.b_i + (a_i \oplus b_i).c_i \quad (3)$$
$$s_i = (a_i \oplus b_i) \oplus c_i \quad (4)$$

The above two equations can be written in terms of two new signals P_i and G_i, which are shown in Figure 4:

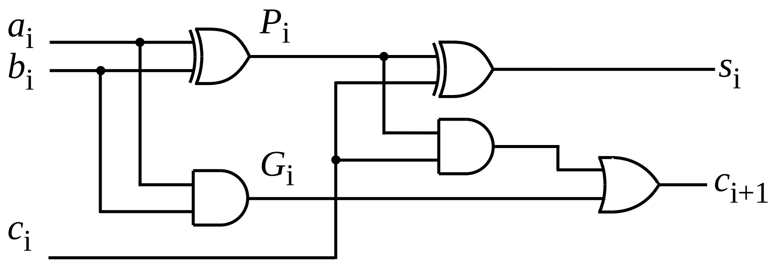

Figure 4: Full adder at stage i with P_i and G_i shown.

$$c_{i+1} = G_i + P_i.c_i \quad (5)$$
$$s_i = P_i \oplus c_i \quad (6)$$

where

$$G_i = a_i.b_i \quad (7)$$
$$P_i = a_i \oplus b_i \quad (8)$$
$$(9)$$

G_i and P_i are called the carry generate and carry propagate terms, respectively. Notice that the generate and propagate terms only depend on the input bits and thus will be valid after one and two gate delay, respectively. If one uses the above expression to calculate the carry signals, one does not need to wait for the carry to ripple through all the previous stages to find its proper value. Let's apply this to a 4-bit adder to make it clear.

Putting $i = 0, 1, 2, 3$ in Equation 5, we get

$$c_1 = G_0 + P_0.c_0 \tag{10}$$

$$c_2 = G_1 + P_1.G_0 + P_1.P_0.c_0 \tag{11}$$

$$c_3 = G_2 + P_2.G_1 + P_2.P_1.G_0 + P_2.P_1.P_0.c_0 \tag{12}$$

$$c_4 = G_3 + P_3.G_2 + P_3.P_2.G_1 + P_3.P_2.P_1.G_0 + P_3.P_2.P_1.P_0.c_0 \tag{13}$$

Notice that the carry-out bit, c_{i+1}, of the last stage will be available after four delays: two gate delays to calculate the propagate signals and two delays as a result of the gates required to implement Equation 13.

Figure 5 shows that a 4-bit CLA is built using gates to generate the P_i and G_i signals and a logic block to generate the carry out signals according to Equations 10–13.

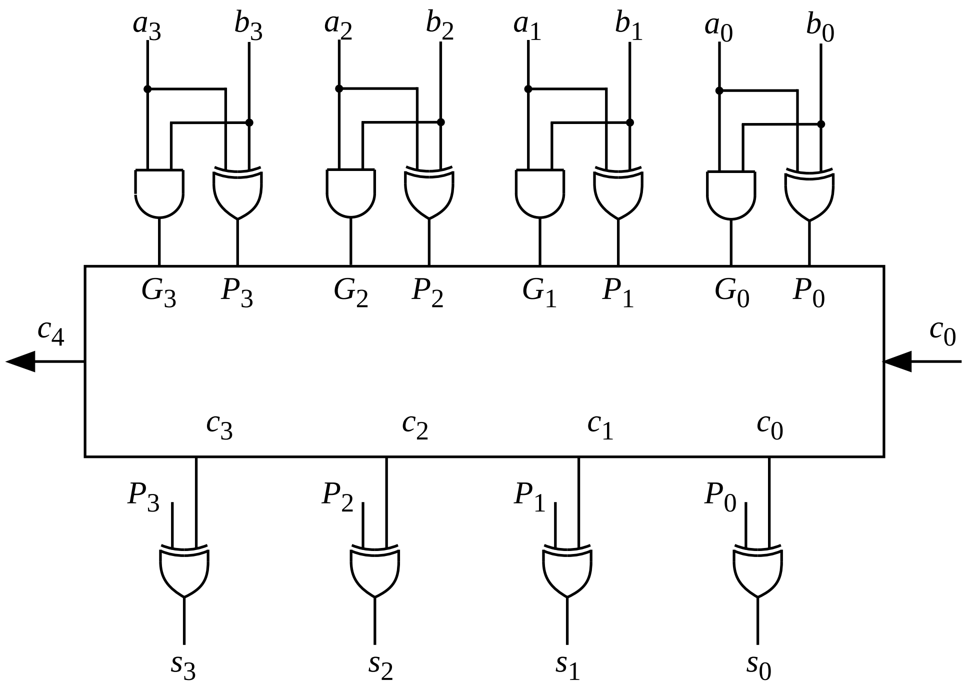

Figure 5: 4-Bit carry lookahead adder implementation detail.

The disadvantage of CLA is that the carry logic block gets very complicated for more than 4-bits. For that reason, CLAs are usually implemented as 4-bit modules and are used in a hierarchical structure to realize adders that have multiples of 4-bits.

3 Pre-Lab Report

For your prelab report, you are required to do the following.

1. Estimate the gate delays associated with the output signals for the 1-bit full adder.
2. Estimate the gate delays associated with the output signals for the 4-bit CLA.
3. Using a 4-bit CLA adder as a building block, draw a block diagram to show how to construct a 16-bit CLA adder.

4 Project Requirements

In this project you are required to design, model, and simulate a carry ripple adder and a carry lookahead adder.

4.1 Ripple carry adder requirements

1. Write VHDL behavioral models for OR, AND, and XOR gates.
2. The delays of the OR, AND, and XOR gates should be assigned with the help of Table 2 and assuming the delay of an inverter is 1 ns.
3. Model a 1-bit full adder using structural description using the OR, AND, and XOR gates as compoenents.
4. Model a 4-bit adder in a separate file using VHDL structural description. The 4-bit adder will use 1-bit full adders as components.
5. Model a 16-bit adder in a separate file using the VHDL structural description. The 16-bit adder will use 4-bit ripple carry adders as components.
6. The 16-bit adder has two inputs a and b of type `bit_vector` representing the addend and augend; and 1-bit input signal c_in of type `bit` representing the carry in. The adder produces one output signal s of type `bit_vector` representing the sum word and a 1-bit output signal c_out of type `bit` representing the carry out.
7. Write a testbench to verify the operation of the 16-bit adder. The testbench should try different number values. Simulate the behavior of the adders using the testbench you developed.

4.2 Carry lookahead adder requirements

1. Write a VHDL code for a 4-bit CLA should be defined using structural description with components defined in Figure 5.
2. The delays of the 4-bit CLA should be assigned with the help of Table 2 and assuming the delay of an inverter is 1 ns.
3. Model a 16-bit adder in a separate file using VHDL structural description. The 16-bit adder will use 4-bit CLA's as components.
4. The 16-bit adder has two inputs a and b representing the addend and augend; and 1-bit input signal c_in representing the carry in. The adder produces one output signal s representing the sum word and a 1-bit output signal c_out representing the carry out.
5. Write a testbench to verify the operation of the 16-bit CLA. The testbench should try different number values. Simulate the behavior of the adder using the testbench you developed.

5 Lab Report

Your lab report should include the following.

5.1 Ripple carry adder

1. Refer to the lab report grading scheme for items that must be present in your lab report.
2. Find the delay of the ripple carry adder using the waveform you got from the simulation.
3. Using the data of Table 2 estimate the area required for the 4-bit ripple carry adder in Figure 3.
4. Estimate the area of a 16-bit carry ripple adder.

5.2 Carry lookahead adder

1. Refer to the lab report grading scheme for items that must be present in your lab report.
2. Using the data of Table 2 estimate the area required the 4-bit CLA in Figure 5 and for a 16-bit CLA.
3. Find the delay of the adder using the waveform you got from the simulation.
4. Compare the delay of the ripple carry adder with the delay of the CLA based on the waveforms you obtained and comment on the result.

5. Compare the area of the ripple carry adder with the area of the CLA based on the waveforms you obtained and comment on the result.

6. Provide a plot showing the theoretical area and delay complexity of n-bit carry ripple adders and an n-bitCLA for values of $1 \leq n \leq 64$