

EXPERIMENT 4

Binary Arithmetic Circuits

The objectives of this Experiment:

In this lab you will learn about binary arithmetic, and you will learn to build digital logic circuits for the addition of binary numbers. The objectives of this experiment include:

- Learn to build and use binary arithmetic circuits,
- Continue to build experience with the Bit Bucket digital breadboarding system and LogicWorks digital simulation,
- Develop professional communication skills.

I. Introduction

One of the most fundamental uses of digital systems is numerical computation. Computers use the binary, or base 2, number system. Therefore we must learn how to perform binary arithmetic in order to build circuits to execute the desired operations of addition, subtraction, etc. Since the only possible outputs of a logic gate are 0 and 1, let's look at the possible arithmetic problems that arise from computing the sum of these two numbers:

Table 1. Base 10 Arithmetic Using 0 and 1			
$\begin{array}{r} 0 \\ +0 \\ \hline 0 \end{array}$	$\begin{array}{r} 0 \\ +1 \\ \hline 1 \end{array}$	$\begin{array}{r} 1 \\ +0 \\ \hline 1 \end{array}$	$\begin{array}{r} 1 \\ +1 \\ \hline 2 \end{array}$

We can do the same thing in base 2 as shown in Table 2. In Table 1 the last result is written $1+1=2_{10}$, whereas in Table 2 it is written $1+1=10_2$ where the subscript is used to indicate the base being used.

The first three examples have results that can be represented as a digital output (0 or 1). However, in the last example, $1+1=2$, the sum cannot be represented with a single binary value. (This corresponds to the base ten problem of $9+1=10$; we can't write 10 as a single digit, so we "carry the one" into the tens place to get a two-digit answer.) In Figure 1, we show a detailed anatomy of the problem $1+1=2$. In this view, we see that the binary result consists of a *sum bit* (0) and a *carry bit* (1).

Table 2. Base 2 Arithmetic Using 0 and 1			
$\begin{array}{r} 0 \\ +0 \\ \hline 0 \end{array}$	$\begin{array}{r} 0 \\ +1 \\ \hline 1 \end{array}$	$\begin{array}{r} 1 \\ +0 \\ \hline 1 \end{array}$	$\begin{array}{r} 1 \\ +1 \\ \hline 10 \end{array}$

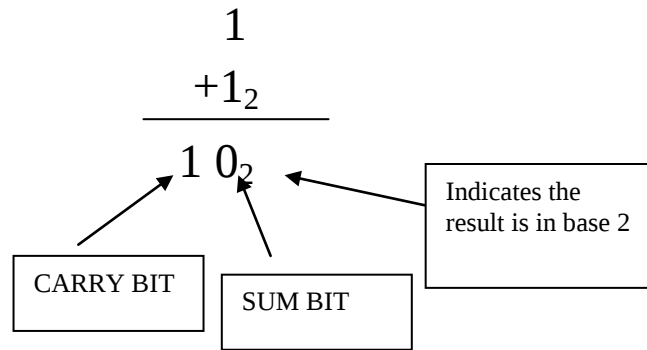


Figure 1. A detailed look at $1+1=2$ in binary.

Half-adder: adding two bits

Table 3 is a truth table for the carry bit (C) and the sum bit (S) when two bits A and B are added.

Table 3. Truth table for adding two bits. The two input bits are A and B, and the two outputs are S (sum) and C (carry).

Inputs		Outputs	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

The truth table shown in Table 3 can be implemented with the following Boolean logic:

$$S = A \oplus B \quad (1)$$

$$C = A \bullet B \quad (2)$$

The corresponding circuit is shown in Fig. 2. The symbol $\oplus$ stands for the logic function *exclusive-or (XOR)*, which is **one** if the inputs are different and **zero** if they are the same. The XOR is available as a standard IC logic gate (for example, the 74x86).

This circuit is called a *half-adder*. The reason for this name is that, although the circuit works fine for adding two bits, it has no provision for adding a carry-in bit, and therefore cannot be used directly in arithmetic involving more than two bits.

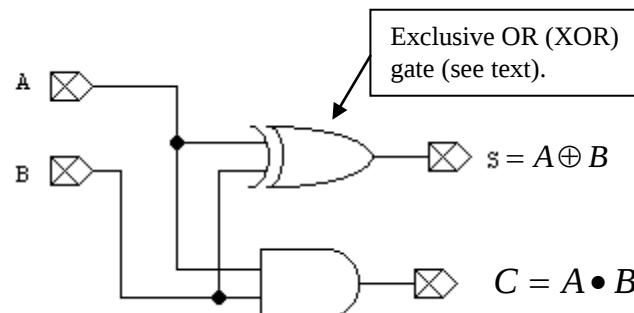


Figure 2. Half-adder logic circuit. The truth table is given in Table 3.

Full-adder: adding two bits plus a carry-in

The truth table for a two-bit full adder is shown in Table 4. The full adder accepts a carry-in, and can therefore be used for adding binary numbers with any number of bits simply by connecting together as many full adders as there are sum bits.

Table 4. Full adder truth table. The three input bits are the sum bits A and B, and the carry-in bit C_{IN} . The two outputs are the S (sum) and C_{OUT} (carry-out) bits.

Inputs			Base 10 Sum of $A+B+C_{IN}$	Binary Outputs	
A	B	C_{IN}		C_{OUT}	S
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	2	1	0
1	0	0	1	0	1
1	0	1	2	1	0
1	1	0	2	1	0
1	1	1	3	1	1

The full adder Boolean logic equations for the sum (S) and carry-out (C_{OUT}) bits are:

$$S = A \oplus B \oplus C_{IN} \quad (3)$$

$$C_{OUT} = A \bullet B + B \bullet C_{IN} + A \bullet C_{IN} \quad (4)$$

The corresponding logic circuit is shown in Fig. 3. The rules of Boolean algebra have been used to reduce the carry-out part of the circuit to the minimum number of logic gates, but it can be proven equivalent to Eq. 4.

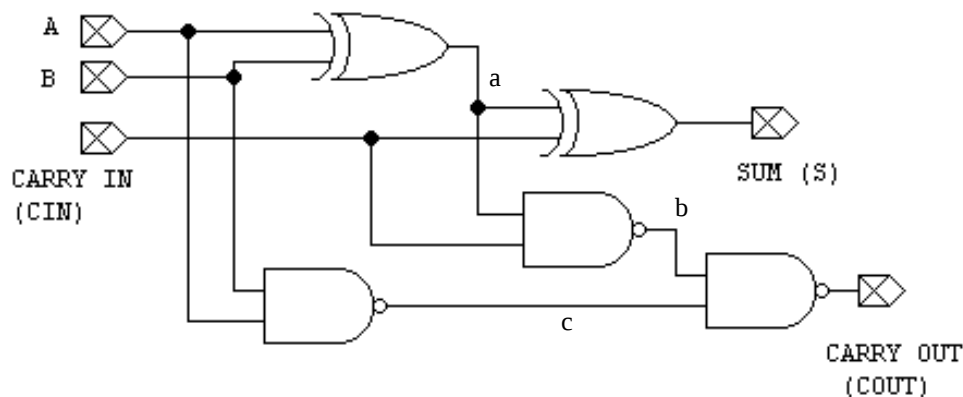


Figure 3. Full adder logic circuit. The truth table is given in Table 4.

The 7483: a 4-bit full adder IC

The 7483 is an IC that adds two 4-bit numbers plus a carry-in, and provides the four-bit sum and a carry-out. The pin-out diagram is shown in Fig. 4.

Hexadecimal (Hex): A shorthand for binary

In the binary number system (base 2), the number of bits can be quite large. Present-day microprocessors use a 32-bit wide data bus. For writing logic functions and arithmetic expressions involving such large numbers, it is convenient to combine the binary digits (bits) into groups of four and represent them with base 16, or hexadecimal (hex) notation. Table 5 lists the decimal, binary, and hex notations for some numbers. The convenience of hex notation for larger numbers is clear from this listing. Note that each group of 4 binary digits is replaced by a single hex digit.

Table 5. Decimal, binary, and hexadecimal notation.

Decimal (base 10)	Binary (base 2)	Hex (base 16)
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	1 0000	10
17	1 0001	11
32	10 0000	20
64	100 0000	40
128	1000 0000	80
1024 (1K)	100 0000 0000	400

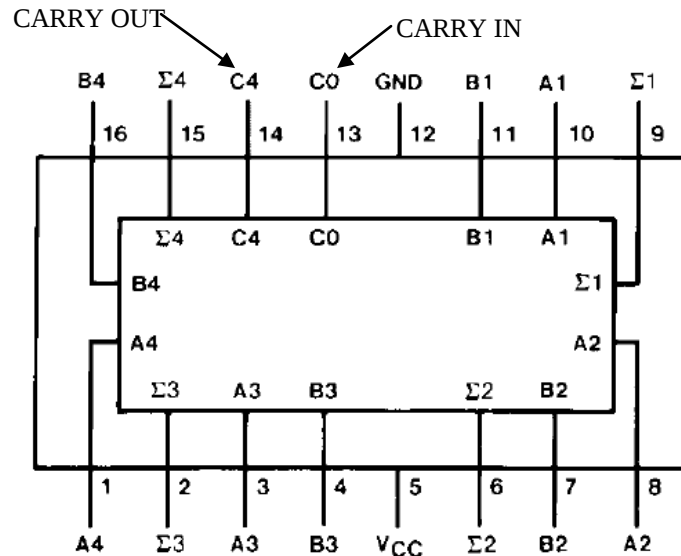


Figure 4. The 7483 four-bit full-adder. The four bits of input A are labeled A1...A4. Bit A4 is the MSB. Input B is labeled similarly. The sum bits are labeled $\Sigma 1$... $\Sigma 4$. The carry-in is labeled C0 and the carry-out is labeled C4.

Decimal numbers are sometimes written with a following 'D', binary with a following 'B', and hexadecimal numbers are often written with a following 'H' to indicate the base. For example:

$$17 \text{ D} = 0001 \ 0001 \text{ B} = 11 \text{ H}$$

One other commonly used number system is base 8, called 'octal.' Although not discussed here, you are likely to run across it if you study computer systems.

PreLab:

- Build and test a 1-bit full adder in LogicWorks.
 - Construct the circuit shown in Fig. 3 using Binary Switch parts for the inputs and Logic Probe parts to display the Sum and Carry Out. Include the schematic in your Prelab.
 - Apply all 8 input combinations and record the outputs S and C_{OUT} in a truth table similar to Table 4. Are your results correct?
- Obtain the data sheets for the 74LS83 4-bit adder and the 74161 binary counter.
- Use LogicWorks to build the 4-bit full adder circuit shown in Figure 5. Include the schematic in your Prelab. Fill in a table of results similar to Table 6. Are your results correct?

Lab Exercise:

Obtain the required components. You will use the following:

Qty	Part # or value	Description
1	7483	4-bit full adder, TTL, 16-pin IC
1	74161	4-bit binary counter, TTL, 16-pin IC

(1) Build and test a 4-bit adder.

Use the 7483 to add two 4-bit binary numbers. One of the 4-bit numbers (we will designate it *A*) will come from a 74161 counter. The other (called *B*) will come from toggle switches. This is illustrated in Fig. 5.

- (a) Connect the circuit according to the pinout from your Prelab. Use four LED's on the Bit Bucket to show the value of input *A*, and use the other four LED's to show the sum *S*. Connect a separate LED / resistor combination on the breadboard to display the Carry Out. You did this previously in Experiment 3.
- (b) Build the circuit. Test it by filling in a table like the one below (Table 6). Use the exact inputs given. Put this table in your report. Have your instructor observe your circuit in operation and initial the table in your report.

(2) Cleanup

DO NOT PUT CHIPS OR ANY OTHER COMPONENTS IN THE WIRE TUBS.

- (a) Turn off the power to the Bit Bucket.
- (c) Disassemble your circuit and place all wires back in the wire tub.
- (d) Put all chips and other components back in the proper bins.
- (e) Clean up your workstation and discard any trash.

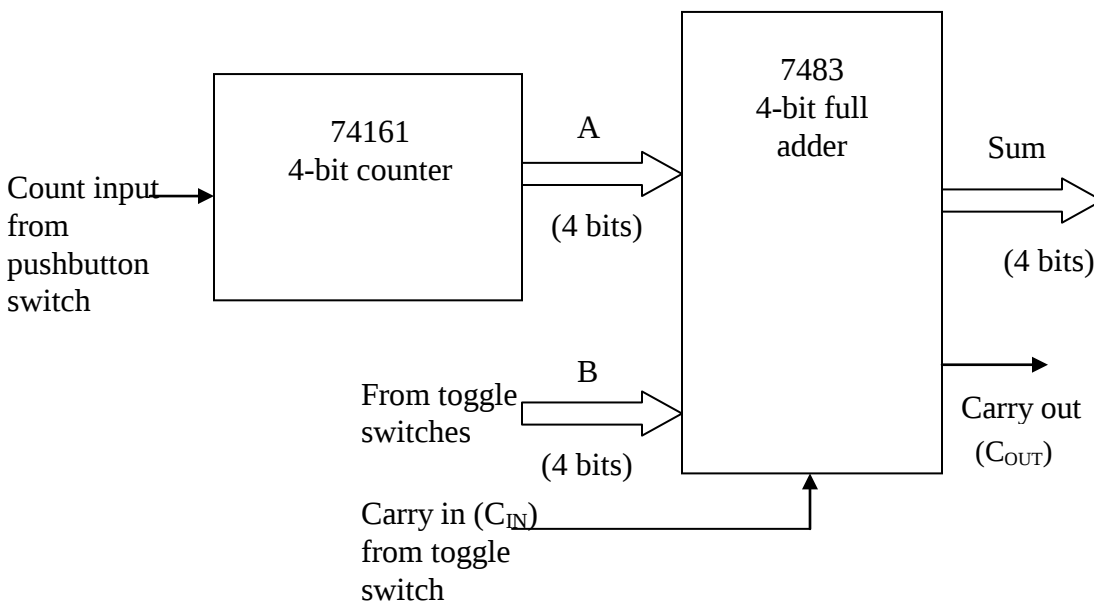


FIGURE 5. Block diagram of 4-bit full adder circuit.

Table 6. Test results for the 4-bit full adder circuit. All values in hex.

Inputs			Calculated Outputs		Actual Outputs		Check here if correct.
A	B	C _{IN}	C _{OUT}	S	C _{OUT}	S	
0	0	0					
0	0	1					
1	0	0					
7	7	1					
8	7	0					
8	8	0					
F	F	0					
F	F	1					