Computer Organization (CDA – 3103)

Spring 2005

Lab # 3: Basic Arithmetical Units

In this lab, we will discuss the basic building blocks that are used for building any basic computer. These are the digital circuit used for performing arithmetical operations and are part of the Arithmetical & logical Unit (ALU) of any general-purpose computer system.

1 Adder

A key requirement of digital computers is the ability to use logical functions to perform arithmetic operations. The basis of this is addition; if we can add two binary numbers, we can just as easily subtract them, or get a little fancier and perform multiplication and division. How, then, do we add two binary numbers?

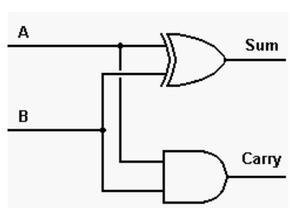
1.1 Half Adder

Let's start by adding two binary bits. Since each bit has only two possible values, 0 or 1, there are only four possible combinations of inputs. These four possibilities, and the resulting sums, are:

```
0 + 0 =  0
0 + 1 =  1
1 + 0 =  1
1 + 1 = 10
```

Whoops! That fourth line indicates that we have to account for two output bits when we add two input bits: the sum and a possible carry. Let's set this up as a truth table with two inputs and two outputs, and see where we can go from there.

A_1	A_0	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



Well, this looks familiar, doesn't it? The Carry output is a simple AND function, and the Sum is an Exclusive-OR. Thus, we can use two gates to add these two bits together. The resulting circuit is also shown above.

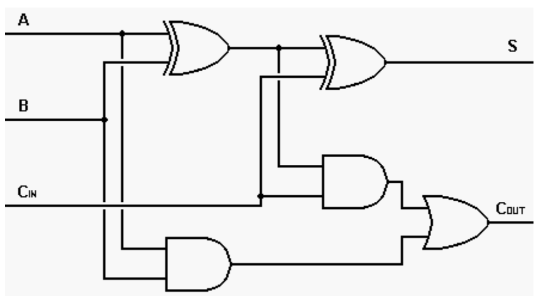
1.2 FULL ADDER

To construct a full adder circuit, we'll need three inputs and two outputs. Since we'll have both an input carry and an output carry, we'll designate them as CIN and COUT. At the same time, we'll use S to designate the final Sum output. The resulting truth table is shown to the right.

INPUTS			OUTPUTS	
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

Hmmm. This is looking a bit messy. It looks as if COUT may be either an AND or an OR function, depending on the value of A, and S is either an XOR or an XNOR, again depending on the value of A. Looking a little more closely, however, we can note that the S output is actually an XOR between the A input and the half-adder SUM output with B and CIN inputs. Also, the output carry will be true if any two or all three inputs are logic 1.

What this suggests is also intuitively logical: we can use two half-adder circuits. The first will add A and B to produce a partial Sum, while the second will add CIN to that Sum to produce the final S output. If either half-adder produces a carry, there will be an output carry. Thus, COUT will be an OR function of the half-adder Carry outputs. The resulting full adder circuit is shown below.



2 SUBTRACTION

Subtraction is also achieved by doing the addition. A – B = A + (-B)

Example : Using 8-bit 2's complement notation

$(53)_{10} - (47)_{10}$ = $(00110101)_2 - (00101111)_2$
= $(00110101)_2 + [- (00101111)_2]$
$=_{\text{2's complement}}$ $(00110101)_2 + (11010001)_2$
= $(00000110)_2$ (actually 100000110 , but most significant '1' was an overflow)
= $(6)_{10}$

3 MULTIPLICATION

Now we will build more complex circuit for multiplication using circuits like basic gates, adders, subtractors and techniques like shift operations.

Let's see how we multiply two decimal numbers on using paper & pencil. The same technique will be used for building a digital circuit for multiplication.

Suppose we are multiplying 1000 with 1001

```
   1000        multiplicand
 * 1001        multiplier
 ----------
   1000
  0000
 0000
1000
----------------
1001000        product
```

Hardware: