Binary Arithmetic

COE 202

Digital Logic Design

Dr. Muhamed Mudawar

King Fahd University of Petroleum and Minerals

Adding Bits

- 1 + 1 = 2, but 2 should be represented as $(10)_2$ in binary
- Adding two bits: the sum is S and the carry is C

X	0	0	1	1
+ Y	+ 0	+ 1	+ 0	+ 1
C S	0 0	0 1	0 1	1 0

- Adding three bits: the sum is S and the carry is C

0	0	0	0	1	1	1	1
0	0	1	1	0	0	1	1
+ 0	+ 1	+ 0	+ 1	+ 0	+ 1	+ 0	+ 1
0 0	0 1	0 1	1 0	0 1	1 0	1 0	1 1

Binary Addition

- Start with the least significant bit (rightmost bit)
- Add each pair of bits
- Include the carry in the addition, if present

carry		**1**	**1**	**1**	**1**				
	0	0	1	1	0	1	1	0	(54)
+	0	0	0	1	1	1	0	1	(29)
	0	1	0	1	0	0	1	1	(83)
bit position:	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	

Subtracting Bits

❖ Subtracting 2 bits (X – Y): we get the difference (D) and the **borrow-out** (B) shown as 0 or -1

X	**0**	**0**	**1**	**1**
– Y	**– 0**	**– 1**	**– 0**	**– 1**
B D	**0 0**	**-1 1**	**0 1**	**0 0**

❖ Subtracting two bits (X – Y) with a **borrow-in = -1**: we get the difference (D) and the **borrow-out** (B)

borrow-in	**-1**	**-1**	**-1**	**-1**	**-1**
	X	**0**	**0**	**1**	**1**
	– Y	**– 0**	**– 1**	**– 0**	**– 1**
	B D	**-1 1**	**-1 0**	**0 0**	**-1 1**

Binary Subtraction

- Start with the least significant bit (rightmost bit)
- Subtract each pair of bits
- Include the borrow in the subtraction, if present

borrow			-1	-1			-1		
	0	0	1	1	0	1	1	0	(54)
–	0	0	0	1	1	1	0	1	(29)
	0	0	0	1	1	0	0	1	(25)
bit position:	7	6	5	4	3	2	1	0	

Binary Multiplication

- Binary Multiplication table is simple:

  0×0=0,   0×1=0,   1×0=0,   1×1=1

```
Multiplicand           1100₂ = 12
Multiplier        ×    1101₂ = 13
                  ----------------
                       1100
                      0000
                     1100
                    1100
                  ----------------
Product           10011100₂ = 156
```

> Binary multiplication is easy
>
> 0 × multiplicand = 0
>
> 1 × multiplicand = multiplicand

- *n*-bit multiplicand × *n*-bit multiplier = 2*n*-bit product
- Accomplished via shifting and addition

Hexadecimal Addition

- Start with the least significant hexadecimal digits
- Let Sum = summation of two hex digits
- If Sum is greater than or equal to 16
  - Sum = Sum – 16 and Carry = 1
- Example:

```
carry         1   1       1
      9   C   3   7   2   8   6   5
+     1   3   9   5   E   8   4   B
      -----------------------------
      A   F   C   D   1   0   B   0
```

5 + B = 5 + 11 = 16
Since Sum ≥ 16
Sum = 16 – 16 = 0
Carry = 1

Hexadecimal Subtraction

- Start with the least significant hexadecimal digits
- Let Difference = subtraction of two hex digits
- If Difference is negative
  - Difference = 16 + Difference and Borrow = -1
- Example:

```
borrow     -1   -1         -1
        9  C  3  7  2  8  6  5
-       1  3  9  5  E  8  4  B
        ----------------------
        8  8  A  1  4  0  1  A
```

Since 5 < B, Difference < 0
Difference = 16+5–11 = 10
Borrow = -1

Shifting the Bits to the Left

❖ What happens if the bits are shifted to the left by 1 bit position?

Before	0	0	0	0	0	1	0	1	= 5
After	0	0	0	0	1	0	1	0	= 10

Multiplication By 2

❖ What happens if the bits are shifted to the left by 2 bit positions?

Before	0	0	0	0	0	1	0	1	= 5
After	0	0	0	1	0	1	0	0	= 20

Multiplication By 4

❖ Shifting the Bits to the Left by *n* bit positions is multiplication by 2^n

❖ As long as we have sufficient space to store the bits

Shifting the Bits to the Right

❖ What happens if the bits are shifted to the right by 1 bit position?

Before	0	0	1	0	0	1	1	**0**	= 38
After	0	0	0	1	0	0	1	1	= 19, **r=0**

Division By 2

❖ What happens if the bits are shifted to the right by 2 bit positions?

Before	0	0	1	0	0	1	**1**	**0**	= 38
After	0	0	0	0	1	0	0	1	= 9, **r=2**

Division By 4

❖ Shifting the Bits to the Right by n bit positions is division by 2^n

❖ The **remainder r** is the value of the bits that are **shifted out**