Chapter 2

Binary Arithmetic

Instructor: Prof. Peter Lian
Department of Electrical Engineering & Computer Science
Lassonde School of Engineering
York University

Review of Number Systems

Four Important Number Systems

System	Why?	Remarks
Decimal	Base 10 (10 fingers)	Most used system
Binary	Base 2. On/Off systems	3 times more digits than decimal
Octal	Base 8.Shorthand notation for working with binary	3 times less digits than binary
Hex	Base 16	4 times less digits than binary

Positional Number Systems

- Have a radix *r* (base) associated with them.
- In the decimal system, *r* = 10:
  - Ten symbols: 0, 1, 2, ..., 8, and 9
  - More than 9 move to next position, so each position is power of 10
  - Nothing special about base 10 (used because we have 10 fingers)
- What does 642.391_{10} mean?

$6 \times 10^{2} + 4 \times 10^{1} + 2 \times 10^{0} \ . \ 3 \times 10^{-1} + 9 \times 10^{-2} + 1 \times 10^{-3}$

← Increasingly +value powers of radix | ↑ Radix point | Increasingly -value powers of radix →

Positional Number Systems

Number system	Radix	Symbols
Binary	2	{0,1}
Octal	8	{0,1,2,3,4,5,6,7}
Decimal	10	{0,1,2,3,4,5,6,7,8,9}
Hexadecimal	16	{0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f}

Quiz Time!!!

- Convert (10111100.00001110)b to Octal form
  A. (274.034)o
  B. (570.016)o
  C. (270.014)o
  D. (574.034)o

Binary Arithmetic

Binary Addition

a + b

Let a = 1, b = 1, what is the sum?

1
+1
2

Truth Table

a	b	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

1
+1
(sum=0, carry=1) 1 0

Binary Addition – Examples

1 1 0	6
1 1	3
+ 1 1 1	7
1 0 0 0 0	16

1 1 0 1 1 . 1 0 1	27.625
+ 1 0 1 0 . 1 1 1	10.875
1 0 0 1 1 0 . 1 0 0	38.5

Note: Addition rule for binary non-integers same as for integers.

Binary Subtraction

a - b   a = minuend
b = subtrahend

Truth table

a	b	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

0
- 1
(diff=1, borrow=1) 1

Binary Subtraction – Examples

1 1 1 0	14
- 1 0 0	- 4
1 0 1 0	10

1 1 0 1	13
- 1 0	-2
1 0 1 1	11

1 1 0 0 0 1	49
- 1 0 0 1 1	- 19
1 1 1 1 0	30

Note: Subtraction rule for binary non-integers same as for integers.

Binary Multiplication

a x b

Truth table

a	b	Product
0	0	0
0	1	0
1	0	0
1	1	1

Binary Multiplication – Example

```
              1  1  0  1  .  1
          x   1  0  1  0  .  1
-------------------------------
                 1  1  0  1  1
              0  0  0  0  0
           1  1  0  1  1
        0  0  0  0  0
     1  1  0  1  1
-------------------------------
 1   0  0  0  1  1  0  1.1  1
```

Note: Rule for positioning binary point identical to that in decimal number system.

Multiplication of PoT number

- Power-of-Two (PoT) number
  - A PoT number is defined as the sum or difference of *n* numbers of power-of-two terms, e.g. $2^p + 2^q$ or $2^p – 2^q$
- More examples of PoT numbers
  - $9=2^3+2^0$
  - $35=2^5+2^2-2^0=2^5+2^1+2^0$
  - $59=2^6-2^2-2^0$

Binary Numbers

- How many distinct numbers can be represented by *n* bits?

No. of bits	Distinct nos.
1	2 {0,1}
2	4 {00, 01, 10, 11}
3	8 {000, 001, 010, 011, 100, 101, 110, 111}
n	2^n

- Number of permutations double with every extra bit
- 2^n *unique* numbers can be represented by *n* bits

Power-of-Two Number

- A Power-of-Two (PoT) number is defined as the sum or difference of *n* numbers of power-of-two terms, i.e. $2^P \pm 2^Q \pm 2^R\ldots$
- Examples of PoT numbers
  - $9=2^3+2^0$
  - $35=2^5+2^2-2^0=2^5+2^1+2^0$
  - $59=2^6-2^2-2^0$

Multiplication of PoT number

- Assume A is a PoT number, A= $2^P + 2^Q$ (P>Q>0), and B is a *N*-bit number.
  - A*B=($2^p + 2^q$)*B

    =B*2^P+B* 2^Q
- No full multiplier is needed in A*B.

Signed Binary Numbers

Negative numbers representation

- Three kinds of representations are common:
  1. Signed Magnitude (SM)
  2. One's Complement
  3. Two's Complement

Signed Magnitude Representation

[0,1] {…………}

Sign bit (left most) — (*n* -1) magnitude bits

- 0 indicates +value
- 1 indicates -value

8 bit representation for +13 is 0 0001101

8 bit representation for -13 is 1 0001101

1's Complement Notation

Let N be an n-bit number and $\tilde{N}(1)$ be the 1's Complement of the number. Then,

$$\tilde{N}(1) = 2^n - 1 - N$$

- The idea is to leave positive numbers as is, but to *represent negative numbers by the 1's Complement of their magnitude*.
- *Example*: Let $n = 4$. What is the 1's Complement representation for +6 and -6?
  - +6 is represented as 0110 (as usual in binary)
  - -6 is represented by 1's complement of its magnitude (6)

1's Complement Notation

- 1's C representation can be computed in 2 ways:
  - Method 1: 1's C representation of -6 is:
    $2^4 - 1 - |N| = (16 - 1 - 6)_{10} = (9)_{10} = (1001)_2$
  - Method 2: For -6, the magnitude = 6 = $(0110)_2$
    - The 1's C representation is obtained by complementing the bits of the magnitude: $(1001)_2$
    - $2^4 - 1 - |N| = (16)_{10} - 1 - |N| = (15)_{10} - |N|$
      $= (1111)_2 - |N|$

2's Complement Notation

Let N be an n bit number and $\tilde{N}(2)$ be the 2's C of the number. Then,

$$\tilde{N}(2) = 2^n - N$$

- Again, the idea is to leave positive numbers as is, but to *represent negative numbers by the 2's C of their magnitude*.
- *Example*: Let $n = 5$. What is the 2's C representation for +11 and -13?
  - +11 is represented as 01011 (as usual in binary)
  - -13 is represented by 2's complement of its magnitude (13)

2's Complement Notation

- 2's C representation can be computed in 2 ways:
  - Method 1: 2's C representation of -13 is:
    $2^5 - |N| = (32 - 13)_{10} = (19)_{10} = (10011)_2$
  - Method 2: For -13, the magnitude = 13 = $(01101)_2$
    - The 2's C representation is obtained by adding 1 to the 1's C of the magnitude
    - $2^5 - |N| = (2^5 - 1 - |N|) + 1 =$ 1's C + 1

$$01101 \xrightarrow{1\text{'s } C} 10010 \xrightarrow{add\ 1} 10011$$

Comparing all Signed Notations (4-bit)

4-bit No.	SM	1's C	2's C
0000	+0	+0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	-0	-7	-8
1001	-1	-6	-7
1010	-2	-5	-6
1011	-3	-4	-5
1100	-4	-3	-4
1101	-5	-2	-3
1110	-6	-1	-2
1111	-7	-0	-1

- In all 3 representations, a –ve number has a 1 in MSB location
- To handle –ve numbers using *n* bits,
  - ≅2^{n-1} symbols can be used for positive numbers
  - ≅2^{n-1} symbols can be used for negative umbers
- In 2's C notation, only 1 combination used for 0

Addition of Signed Numbers

- SM notation awkward for computations
- 1's C is better, but not as widely used as 2's C which is very convenient
- The 4 combinations that need to be considered for signed number addition are
  1. (+) + (+)
  2. (-) + (+)
  3. (+) + (-)
  4. (-) + (-)

Addition of Signed Numbers

Examples below are shown for 4-bit 2's C arithmetic.

1.
```
  (+5)     0101
+(+2)    +0010
  (+7)     0111
```

2.
```
  (-5)     1011
+(+2)    +0010
  (-3)     1101
```

3.
```
  (+5)     0101
+(-2)    +1110
  (+3)   1 0011
```
ignore the carry

4.
```
  (-5)     1011
+(-2)    +1110
  (-7)   1 1001
```
ignore the carry

What's the meaning of overflow? When does it occur?
How can subtraction be done in 2's C arithmetic?

Overflow

- Example: 7 + 6 (each number in signed 4-bit)

```
 + 7:    0111
 + 6:    0110
+13:    1101  → -3
```
Overflow

- Overflow if result out of range

Operation	Operand A	Operand B	Result Indicating overflow
A+B	≥0	≥0	<0
A+B	<0	<0	≥0
A-B	≥0	<0	<0
A-B	<0	≥0	≥0