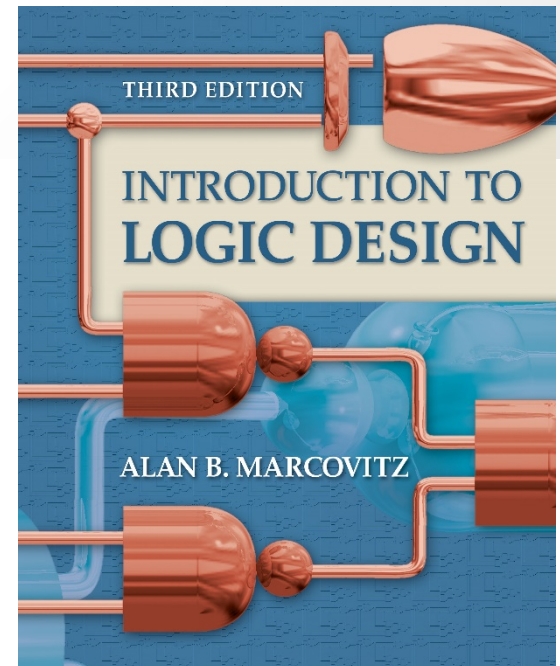


EE2000: Logic Circuit Design

Chapter 1

Acknowledgements: Dr. Wang Hong, Dr. Ray Cheung, Dr. Ricky Lau
Dr. Chang Chip Hong, Dr. Goh Wang Ling



Revision on Number System

- By the end of this topic, we will know:
 - ❑ Terminology used in Number Systems
 - ❑ Number representations in Binary, Octal and Hexadecimal
 - ❑ How to convert numbers among Decimal, Binary, Octal and Hexadecimal
 - ❑ Signed number representation with 2's Complement
 - ❑ How to convert between Decimal numbers and 2's Complement numbers
 - ❑ How to perform Binary Addition and Subtraction
 - ❑ Weighted codes, e.g. BCD
 - ❑ Non-weighted code, e.g. Gray Code, ASCII code

Digital Device and Design Levels

- Digital systems or digital computers use digits or numbers to represent and process information: data, text, graphics, sound, video, etc.
- **Binary numbers** are used in digital systems.
- Binary numbers consist **two digits** (**0** and **1**).
- Each digit is called a BIT (**B**inary dig**I**).
- **BITs** are implemented with digital electronic circuits and have voltage levels: **LOW & HIGH**.
- These circuits are called **Logic Gates**, which operates on 0 & 1 (**False & True**).

Number System

- A number system specifies how values are represented.
- Human uses DECIMAL Number System.
- There are ten digits in Decimal Number System: 0,1,2,3,4,5,6,7,8,9.
- Digital Computers use Binary Numbers, which have only two digits: 0,1.
- There are other number systems, including:
 - **Octal**
 - **Hexadecimal**

Number System Examples

<i>Binary</i>	<i>Decimal</i>	<i>Octal</i>	<i>3-Bit String</i>	<i>Hexadecimal</i>	<i>4-Bit String</i>
0	0	0	000	0	0000
1	1	1	001	1	0001
10	2	2	010	2	0010
11	3	3	011	3	0011
100	4	4	100	4	0100
101	5	5	101	5	0101
110	6	6	110	6	0110
111	7	7	111	7	0111
1000	8	10	—	8	1000
1001	9	11	—	9	1001
1010	10	12	—	A	1010
1011	11	13	—	B	1011
1100	12	14	—	C	1100
1101	13	15	—	D	1101
1110	14	16	—	E	1110
1111	15	17	—	F	1111

Positional Number System

- Values are represented by the digits and their positions in the number.
- This type of number system is called **Positional Number System**.
- E.g., in Decimal Number System, 8973 is eight thousand nine hundred and seventy three:

$$8 = 8000 = 8 \times 10^3 \quad (\text{Thousands Place})$$

$$9 = 900 = 9 \times 10^2 \quad (\text{Hundreds Place})$$

$$7 = 70 = 7 \times 10^1 \quad (\text{Tens Place})$$

$$3 = 3 = 3 \times 10^0 \quad (\text{Units Place})$$

Number System Terminology

- In Number System, a value of an **n-digit number** $a_{n-1}a_{n-2}\dots a_1a_0$ is:

$$N = a_{n-1} \times r^{n-1} + a_{n-2} \times r^{n-2} + \dots + a_1 \times r^1 + a_0 \times r^0$$

where $a_{n-1}, a_{n-2}, \dots, a_1, a_0$ are coefficients.

- r is called the **Base** or **Radix**

Decimal is **Base-10** system, $r = 10$

Binary is **Base-2** system, $r = 2$

Octal is **Base-8** system, $r = 8$

Hexadecimal is **Base-16** system, $r = 16$

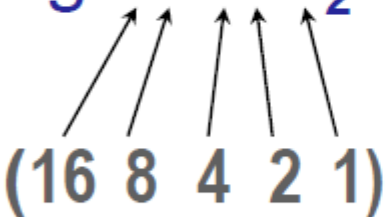
Binary Number System

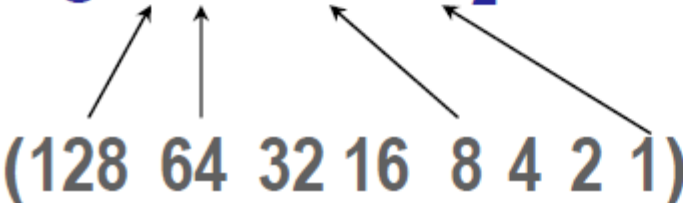
- An **n-bit binary number** $a_{n-1}a_{n-2}\dots a_1a_0$ has a value:
$$N = a_{n-1} \times 2^{n-1} + a_{n-2} \times 2^{n-2} + \dots + a_1 \times 2^1 + a_0 \times 2^0$$
- e.g. A **4-bit binary number** 1011_2 is:
$$\begin{aligned} N &= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\ &= 8 + 0 + 2 + 1 = \underline{11}_{10} \end{aligned}$$
- e.g. A **6-bit binary number** 110010_2 is:
$$\begin{aligned} N &= 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \\ &= 32 + 16 + 0 + 0 + 2 + 0 \\ &= \underline{50}_{10} \end{aligned}$$
- The leftmost bit is called the *high-order* or *most significant bit* (**MSB**); the rightmost is the *low-order* or *least significant bit* (**LSB**).

Binary to Decimal Number Conversion

- Binary numbers can be converted to Decimal numbers by using:

$$N = a_{n-1} \times 2^{n-1} + a_{n-2} \times 2^{n-2} + \dots + a_1 \times 2^1 + a_0 \times 2^0$$

- e.g. $10100_2 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$
 $= 16 + 4 = \underline{20}_{10}$


- e.g. $11001001_2 = 128 + 64 + 8 + 1 = \underline{201}_{10}$


Decimal to Binary Number Conversion

- Decimal numbers can be converted to binary numbers by finding the coefficients of

$$N = a_{n-1} \times 2^{n-1} + a_{n-2} \times 2^{n-2} + \dots + a_1 \times 2^1 + a_0 \times 2^0$$

- e.g. $17_{10} = 16 + 1$
 $= 1 \times 2^4 + 1 \times 2^0$
 $= \underline{1\ 0\ 0\ 0\ 1}_2$


- A more systematic method is to divide the decimal number by 2 successively.

Decimal to Binary by division of 2

○ For example:

○ $168_{10} = 10101000_2$ ←

○ Steps:

- Divide the decimal by 2 until the quotient is 0
- The remainders are the coefficient
- The binary number is the reverse order of the remainders

Quotient	Remainder	Coefficient
168		
84	0	a_0
42	0	a_1
21	0	a_2
10	1	a_3
5	0	a_4
2	1	a_5
1	0	a_6
0	1	a_7

Octal Number System

- Octal Numbers are **base 8**.
- Octal has **8 digits: 0, 1, 2, 3, 4, 5, 6, 7**.
- An n-bit octal number $a_{n-1}a_{n-2}\dots a_1a_0$ has a decimal value:

$$a_{n-1} \times 8^{n-1} + a_{n-2} \times 8^{n-2} + \dots + a_1 \times 8^1 + a_0 \times 8^0$$

- The conversion of octal to decimal can be done with the above equation.
- e.g. $263_8 = 2 \times 8^2 + 6 \times 8^1 + 3 \times 8^0$
 $= 128 + 48 + 3 = \underline{179}_{10}$

Hexadecimal Number System

- Hexadecimal Numbers are **base 16**.
- There are **16 digits**: **0 to 9, A, B, C, D, E, F**.
- An n-bit hexadecimal number $a_{n-1}a_{n-2}\dots a_1a_0$ has a decimal value:

$$a_{n-1} \times 16^{n-1} + a_{n-2} \times 16^{n-2} + \dots + a_1 \times 16^1 + a_0 \times 16^0$$

- The conversion of hexadecimal to decimal can be done with the above equation.
- e.g. $\mathbf{B5E}_{16} = 11 \times 16^2 + 5 \times 16^1 + 14 \times 16^0$
 $= 2816 + 80 + 14 = \underline{\underline{2910}}_{10}$

Decimal to Octal Number Conversion

- Can be done by successive division of 8.

- e.g. $938_{10} = \underline{1652}_8$

- Steps:

- Divide the decimal by 8 until the quotient is 0
- The remainders are the coefficients
- The octal number is the reverse order of the remainders

Quotient	Remainder	Coefficient
938		
117	2	a_0
14	5	a_1
1	6	a_2
0	1	a_3

Decimal to Hexadecimal Number Conversion

○ Can be done by successive division of 16.

○ e.g. $2793_{10} = \underline{AE9}_{16}$

○ Steps:

- ❑ Divide the decimal by 16 until the quotient is 0
- ❑ The remainders are the coefficients
- ❑ The hexadecimal number is the reverse order of the remainders

Quotient	Remainder	Coefficient
2793		
174	9	a_0
10	E (14)	a_1
0	A (10)	a_2

Conversion between Binary and Octal

- Conversion between Binary and Octal is convenient.
- Each Octal digit equals to **3 bits**.

0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

e.g. **3 6 2₈**
 ↓ ↓ ↓
 011 110 010₂

5 4 1₈
 ↓ ↓ ↓
 101 100 001₂

e.g. **010 101 110₂**
 ↓ ↓ ↓
 2 5 6₈

111 010 001₂
 ↓ ↓ ↓
 7 2 1₈

Conversion between Binary and Hex

- Conversion between Binary and Hexadecimal is also convenient.
- Each Hexadecimal digit equals to **4 bits**.

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

E 2 8 0₁₆

1110 0010 1000 0000₂

F B 1₁₆

1111 1011 0001₂

Representation of Signed Numbers

- In ordinary arithmetic, a plus/minus sign “+”/“-” is used to represent **positive/negative numbers**, e.g. +38/-38.
- In digital electronic circuits, everything is represented with bits (0, 1).
- There are several ways to represent the signed binary numbers with just bits, e.g.:
 - ❑ Sign & Magnitude Representation
 - ❑ 1's Complement Representation
 - ❑ 2's Complement Representation

Sign Magnitude Representation

- The most significant bit (MSB) is a sign bit.
- If the MSB is 0, the number is positive.
- If the MSB is 1, the number is negative.
- e.g. 01101 = +13
 11101 = -13
- e.g. 00000 = +0
 10000 = -0
- Disadvantages: (a) 2 patterns represent 0
 (b) Handle sign bit separately

One's complement representation

- Positive numbers and the corresponding negative numbers complement each other.
- Complement is inversion (Logic NOT):
 - The complement of 0 is 1.
 - The complement of 1 is 0.
- e.g. $01101 = +13$
 $10010 = -13$
- e.g. $00000 = +0$
 $11111 = -0$
- Advantage: Its symmetry and the ease of complementation.
- Disadvantage: 2 patterns represent 0.

Two's complement representation

- Positive numbers and Negative numbers are their corresponding 1's complement number + 1.
- e.g. $01101 = +13$
 $10010 + 1 = 10011 = -13$
- e.g. $00000 = +0$
 $11111 + 1 = 00000 = +0$
- Advantages: (a) only 1 pattern represents 0.
(b) handle sign bit as other bits.

Two's complement number conversion

- To obtain the negative value of a number in 2's complement system, we **invert ALL the bits and add 1 to it.**
- e.g. $5 = 00101$
 $-5 = 11010 + 1 = 11011$
- e.g. $-5 = 11011$
 $5 = 00100 + 1 = 00101$

This process is called: **“take the 2's complement”.**

Value of Two's complement number

- An **n-bit 2's complement number** $a_{n-1}a_{n-2}\dots a_1a_0$ has a value:

$$N = (-1)a_{n-1} \times 2^{n-1} + a_{n-2} \times 2^{n-2} + \dots + a_1 \times 2^1 + a_0 \times 2^0$$

- The equation is the same as the one for binary numbers except the MSB has a **negative** weight.
- E.g.: A 5-bit 2's complement number **11011** has the equivalent decimal value of:

$$\begin{aligned} N &= (-1)1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\ &= -16 + 8 + 0 + 2 + 1 = -5 \end{aligned}$$

- Any number having its MSB = 1 is **negative**.
- Any number having its MSB = 0 is **positive**.

Two's complement number conversion

- To find the 2's complement number for a negative decimal, we can find the binary of the positive decimal and then take its 2's complement.

e.g. -9_{10}
+9 = 01001
Invert bits: 10110
Plus 1: $10111 = \underline{-9}_{10}$
Check: $-16 + 0 + 4 + 2 + 1 = \underline{-9}$

Add a "0" bit for MSB for +ve number

Non-decimal numbers addition and subtraction

- The arithmetic of non-decimal numbers is carried out in the same way as the decimal arithmetic.
- The addition $(x + y + c_{in})$ and subtraction $(x - y - b_{in})$ table for binary digits is shown below:

c_{in} or b_{in}	x	y	c_{out}	s	b_{out}	d
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	0	1	0	1
0	1	1	1	0	0	0
1	0	0	0	1	1	1
1	0	1	1	0	1	0
1	1	0	1	0	0	0
1	1	1	1	1	1	1

Binary Addition

- Add the LSBs of the numbers with an initial carry (c_{in}) of 0, producing carry (c_{out}) and sum (S) bits.
- Continue processing the bits from right to left, adding the carry out of each column into the next column's sum.

E.g. 1

X	190	1	0	1	1	1	1	1	0
Y	+ 141	+	1	0	0	0	1	1	0
X+Y	331	1	0	1	0	0	1	0	1

Carry in

E.g. 2

X	173	1	0	1	0	1	1	0	1
Y	+ 44	+	0	0	1	0	1	1	0
X+Y	217	1	1	0	1	1	0	0	1

E.g. 3

C	011111110
X	127
Y	+ 63
S = X + Y	190

Initial carry in

E.g. 4

C	000000000
X	170
Y	+ 85
S = X + Y	255

Binary Subtraction

- Binary subtraction is performed similarly, using borrows, and producing a difference bit d .

<i>B</i>		<i>B</i>	
	001111100		011011010
<i>X</i>	229	<i>X</i>	210
<i>Y</i>	- 46	<i>Y</i>	- 109
<i>D = X - Y</i>	183	<i>D = X - Y</i>	101

11100101	- 00101110	11010010	- 01101101
10110111		01100101	

Ex. Perform the subtraction of the followings in binary.

<i>B</i>		<i>B</i>	
	170		010101010
<i>X</i>	170	<i>X</i>	10101010
<i>Y</i>	- 85	<i>Y</i>	- 01010101
<i>X - Y</i>	85	<i>X - Y</i>	01010101

Two's complement addition & subtraction

- Two's complement numbers can be added in the same way as the ordinary addition.
- Any carry beyond the MSB is ignored.
- The result will always be the correct sum as long as the range of the number system is not exceeded.
- Examples of 4-bit two's complement additions are:

$$\begin{array}{r} +3 \quad 0011 \\ + +4 \quad + 0100 \\ \hline +7 \quad 0111 \end{array}$$

$$\begin{array}{r} -2 \quad 1110 \\ + -6 \quad + 1010 \\ \hline -8 \quad 11000 \end{array}$$

$$\begin{array}{r} +6 \quad 0110 \\ + -3 \quad + 1101 \\ \hline +3 \quad 10011 \end{array}$$

$$\begin{array}{r} +4 \quad 0100 \\ + -7 \quad + 1001 \\ \hline -3 \quad 1101 \end{array}$$

Octal & Hex addition/subtraction

$$\begin{array}{r}
 1\ 1\ 1\ 0 \leftarrow \text{Carry in} \\
 1\ 4\ 7\ 6_8 \\
 +\ 3\ 5\ 5\ 6_8 \\
 \hline
 5\ 2\ 5\ 4_8
 \end{array}$$

$$\begin{array}{r}
 1\ 1\ 0\ 0 \leftarrow \text{Carry in} \\
 1\ 9\ B\ 9_{16} \\
 +\ C\ 7\ E\ 6_{16} \\
 \hline
 E\ 1\ 9\ F_{16}
 \end{array}$$

Ex. Subtract the following hexadecimal numbers using 2's complement method.

(a) $85_{16} - 3B_{16}$ (Ans. $4A_{16}$)

(b) $E4_{16} - 0D_{16}$ (Ans. $D7_{16}$)

Solution

(a) $3B_{16} = 00111011$

2's complement of $3B_{16} = 11000101$

$= C5_{16}$

$$8\ 5_{16}$$

$$+\ C\ 5_{16}$$

$$\underline{1\ 4\ A_{16}}$$

Discard carry → Answer

Value range

- For **4-bit binary**, the range is:

Unsigned	0000 → 1111	0 → 15
1's Complement	1000 → 1111	-7 → -0
	0000 → 0111	0 → 7
2's Complement	1000 → 1111	-8 → -1
	0000 → 0111	0 → 7

- For **n-bit binary**, the range is:

Unsigned	0 → $+2^n - 1$
1's Complement	$-(2^{n-1} - 1) \rightarrow +2^{n-1} - 1$
2's Complement	$-2^{n-1} \rightarrow +2^{n-1} - 1$

- e.g. 5-bit binary

Unsigned	0 → 31
1's Complement	-15 → +15
2's Complement	-16 → +15

Overflow

- If an addition operation produces a result that exceeds the range of the number system, **overflow** is said to occur.
- Addition of two numbers with different signs can never produce overflow, but addition of two numbers of the same sign can.

○ e.g.

$\begin{array}{r} -3 \\ + -6 \\ \hline -9 \end{array}$	$\begin{array}{r} 1101 \\ + 1010 \\ \hline 10111 = +7 \end{array}$	$\begin{array}{r} +5 \\ + +6 \\ \hline +11 \end{array}$	$\begin{array}{r} 0101 \\ + 0110 \\ \hline 1011 = -5 \end{array}$
$\begin{array}{r} -8 \\ + -8 \\ \hline -16 \end{array}$	$\begin{array}{r} 1000 \\ + 1000 \\ \hline 10000 = +0 \end{array}$	$\begin{array}{r} +7 \\ + +7 \\ \hline +14 \end{array}$	$\begin{array}{r} 0111 \\ + 0111 \\ \hline 1110 = -2 \end{array}$

Binary codes for decimal numbers

- Binary numbers are used by computers but human beings are familiar with decimal numbers.
- A set of n -bit strings in which different bit strings represent different numbers or other things is called a **code**.
- A particular combination of n -bit values is called a **code word**.
- At least 4 bits are needed to represent ten decimal digits. There are billions of different ways to choose ten 4-bit code words and some of the common decimal codes are listed in the Table shown next.

Binary codes for decimal numbers

<i>Decimal digit</i>	<i>BCD (8421)</i>	<i>2421</i>	<i>Excess-3</i>	<i>Biquinary</i>	<i>1-out-of-10</i>
0	0000	0000	0011	0100001	1000000000
1	0001	0001	0100	0100010	0100000000
2	0010	0010	0101	0100100	0010000000
3	0011	0011	0110	0101000	0001000000
4	0100	0100	0111	0110000	0000100000
5	0101	1011	1000	1000001	0000010000
6	0110	1100	1001	1000010	0000001000
7	0111	1101	1010	1000100	0000000100
8	1000	1110	1011	1001000	0000000010
9	1001	1111	1100	1010000	0000000001
<i>Unused code words</i>					
	1010	0101	0000	0000000	0000000000
	1011	0110	0001	0000001	0000000011
	1100	0111	0010	0000010	0000000101
	1101	1000	1101	0000011	0000000110
	1110	1001	1110	0000101	0000000111
	1111	1010	1111	. . .	. . .

We will cover some of them as examples

Binary coded decimal (BCD)

- To facilitate the easy conversion between binary and decimal, **binary-coded decimal (BCD)** is used.
- In **BCD number system**, each decimal digit is represented by **4 bits**.
- BCD encodes the digits 0 through 9 by their 4-bit unsigned binary representations, 0000 through 1001. The code words 1010 through 1111 are not used, i.e., six binary numbers not used in BCD.
- The weights for the BCD bits are 8, 4, 2 and 1, and for this reason it is sometimes called the **8421 code**.

Decimal to BCD conversion

0	0000
1	0001
2	0010
3	0011
4	0100

5	0101
6	0110
7	0111
8	1000
9	1001

Note: binary 1010 to 1111 are not used in BCD

e.g. 386_{10} is 0011 1000 0110 in BCD

e.g. 7589_{10} is 0111 0101 1000 1001 in BCD

BCD to Excess-3 code conversion

- **Excess-3** code is its BCD code plus 3.
- **Excess-3** code has the feature of self-complementing. i.e. the two BCD numbers with a sum of 9 have their **excess-3** codes complementing each other.

<i>BCD (8421)</i>				<i>Excess-3</i>			
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

Gray Code

- Binary numbers have more than 1 bits changing when increasing or decreasing continuously.
- e.g. 0011 to 0100 (3 to 4) ➡ 3 bits change
- In digital electronic circuits, sometimes it is desired to have only 1 bit changed.
- In this case, **Gray Code** can be used, where the adjacent numbers have only 1 bit different.
- Gray code is not weighted code.

4-bit Gray Code

Gray Code	Decimal
0000	0
0001	1
0011	2
0010	3
0110	4
0111	5
0101	6
0100	7

Gray Code	Decimal
1100	8
1101	9
1111	10
1110	11
1010	12
1011	13
1001	14
1000	15

Producing Gray Code

- Gray Code can be produced in this way:

1 bit:

0

1

2 bits:

0 0

0 1

1 1

1 0

Mirror →

Mirror
image

3 bits:

0 0 0

0 0 1

0 1 1

0 1 0

1 1 0

1 1 1

1 0 1

1 0 0

Mirror
image

Summary

- By now, we should know:
 - ❑ Terminology used in Number Systems
 - ❑ Number representations in Binary, Octal and Hexadecimal
 - ❑ How to convert numbers among Decimal, Binary, Octal and Hexadecimal
 - ❑ How to perform Binary Addition and Subtraction
 - ❑ Signed number representation with 2's Complement
 - ❑ How to convert between Decimal numbers and 2's Complement numbers
 - ❑ What BCD, 2421, excess-3, Gray and ASCII Codes are