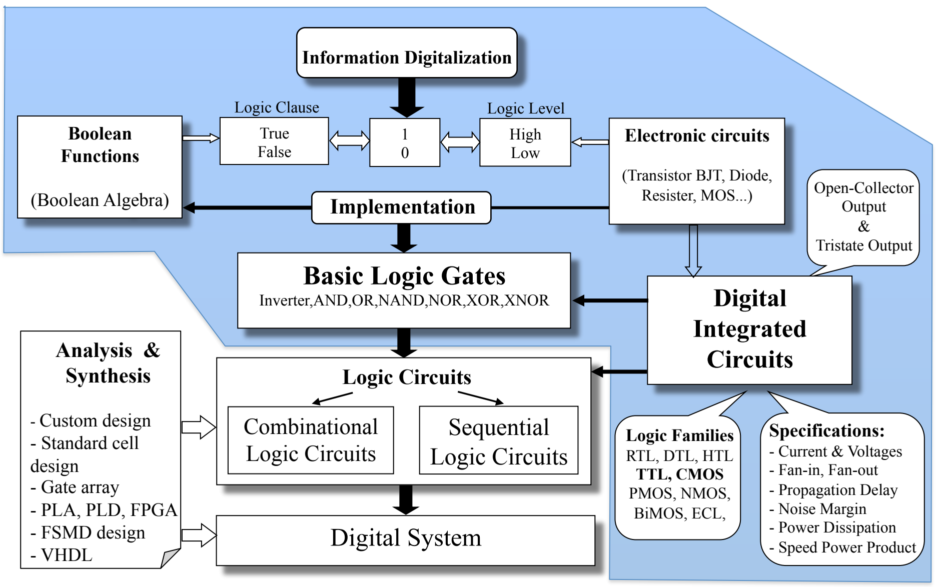
Digital Electronics

- Part I: Digital Principle -

Dr. Lê Dũng

Department of Electronics and Computer System (C9-401)
School of Electronics and Telecommunications
Hanoi University of Science and Technology
Email: ledung-fet@mail.hut.edu.vn

Part I: Digital Principles - Overview



Part I: Digital Principles - Contents

Chapter 1
Binary system and Binary Codes

1.1 Binary System

➢ Decimal System

+ 10 digits = {0,1,2,3,4,5,6,7,8,9} → radix = 10 (Decimal)

+ A number

$D = \mathbf{1974.28}_{10} = 1 \cdot 10^3 + 9 \cdot 10^2 + 7 \cdot 10^1 + 4 \cdot 10^0 + 2 \cdot 10^{-1} + 8 \cdot 10^{-2}$

r (radix) = 10 and i (weighted position) runs from -2 to 3

1.1 Binary System

➢ Number System

+ An ordered set of symbols

+ A number = Positional Notation

$$N = (a_{n-1}a_{n-2}\ldots a_1 a_0 \;.\; a_{-1}a_{-2}\ldots a_{-m})_r$$

where

$.$ = radix point separating the integer and fractional digits
r = radix or base of the number system being used
n = number of integer digits to the left of the radix point
m = number of fractional digits to the right of the radix point
a_i = integer digit i when $n-1 \geq i \geq 0$
a_i = fractional digit i when $-1 \geq i \geq -m$
a_{n-1} = most significant digit
a_{-m} = least significant digit

+ Polynomial Notation $N = \sum_{i=-m}^{n-1} a_i \bullet r^i$ $\quad r-1 \geq a_i \geq 0$

(with r- radix and i-weighted position)

1.1 Binary System

➢ Counting in Decimal System

+ Based on the order {0→1→ 2→3→4→5→6→7→8→9}

+ When 9 return 0 at the weighted position (i)
→ a change at the weighted position (i+1)

For example: 00 → 01 → 02 → … → 09
10 → 11 → 12 → … → 19
20 → 21 → 22 → … → 29
……………………..→ 099 → 100

1.1 Binary System

➢ Binary System

+ Two ordered symbols (2 bits) = {0,1} → radix=2 (Binary)

+ Binary number

$B = 1011.101_2 = 1•2^3 + 0•2^2 + 1•2^1 + 1•2^0 + 1•2^{-1} + 0•2^{-2} + 1•2^{-3}$
$= 11.625_{10}$

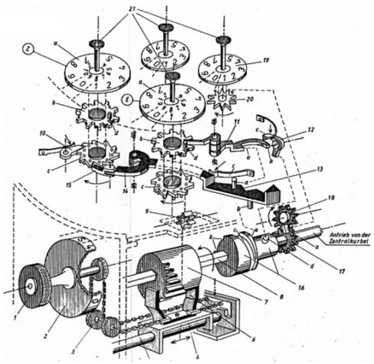
r (radix) = 2, a_i = digit ($0 \le a_i \le 1$)

$$B = \sum_{i=-n}^{m-1} a_i \bullet 2^i$$

+ Binary counting {0 → 1}
{00 → 01 → 10 → 11}
{000 → 001 → ….→111}
{0000 → 0001 → … → 1111}

1.1 Binary System

➢ Why do we use the binary system ?



A drawing of the calculating machine of Müller (Author Werner Lange)

Calculating machine (Müller 1784)
with decimal system

Because: Two bits {0, 1} can be represented more easily by:

+ Two positions of an electrical switch.
+ Two distinct voltage or current levels allowed by a circuit.
+ Two distinct levels of light intensity
+ Two directions of magnetization or polarization
+ ….

1.1 Binary System

➢ Disadvantage of Binary System ?

- Not easy to read and remember → Hexadecimal system

➢ Hexadecimal System

+ 16 symbols = {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,}

+ Hexadecimal Number

$2DC.1E_{16} = 2 \cdot 16^2 + 13 \cdot 16^1 + 12 \cdot 16^0 + 1 \cdot 16^{-1} + 14 \cdot 16^{-2}$

radix = 16 (Hexadecimal system) → Why ?.

1.1 Binary System

➢ Base Conversions

Name	Decimal	Binary	Hexadecimal
Radix	10	2	16
Digits	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	0, 1	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *A*, *B*, *C*, *D*, *E*, *F*
First	0	0	0
seventeen	1	1	1
positive	2	10	2
integers	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	10000	10

❑ Convert to base 10

→ use the polynomial notation with radix and weighted positions

❑ Convert to base 2

→ use radix divide method for the integer part (remainders and quotient)

→ use radix multiply method for the fraction part.

❑ Convert between base 2 and 16

→ 4 bits ←→ 1 hexadecimal digit

1.2 Binary Arithmetic

➢ Addition

Binary addition table ➡ Add two binary numbers

+	0	1
0	0	1
1	1	10

$1 + 1 = 0$ carry $1 = 10_2$

	1	*1*	*1*	*1*	*1*	*1*		*Carries*
		1	1	1	1	0	1	Augend
+			1	0	1	1	1	Addend
	1	0	1	0	1	0	0	Sum

1.2 Binary Arithmetic

➢ Subtraction

1 - 1 = 0 - 0 = 0
1 - 0 = 0
0 - 1 = 1 borrow 1

A (Minuend)		1 1 1 0 1
	-	
B (Subtrahend)		1 1 1 1
borrow		1 1 1 0
difference		0 1 1 1 0

Note: A – B = A + (-B) that means Sub → Add

1.2 Binary Arithmetic

➢ Multiplication

Binary multiplication table → Multiply two binary numbers

×	0	1
0	0	0
1	0	1

				1	0	1	1	1	Multiplicand
×					1	0	1	0	Multiplier
				0	0	0	0	0	
			1	0	1	1	1		
		0	0	0	0	0			
	1	0	1	1	1				
	1	1	1	0	0	1	1	0	Product

Note: - Multiplication by repeated Add & Shift
- Can be implemented in a faster way

1.2 Binary Arithmetic

➢ Division

1 / 1 = 1

0 / 0 = 0 = 0 / 1

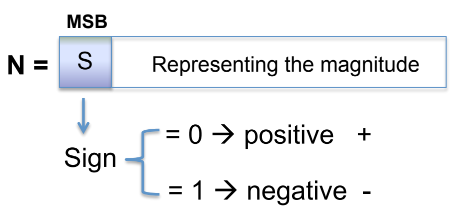
1 / 0 = undefined

```
Divisor 1 0 1 1 1 0 1 0 | 1 1 1 0   Dividend
       - 1 1 1 0          1 1 0 1   Quotient
         1 0 0 1 0 1 0
       -   1 1 1 0
             1 0 0 1 0
             0 0 0 0
             1 0 0 1 0
       -       1 1 1 0
                 1 0 0  Remainder
```

Note: - Division by repeated Sub & Shift

1.3 Sign Number Representation

➢ Sign Number Format



➢ Representing the magnitude

- ❑ Sign magnitude representation
- ❑ Two's complement system

1.3 Sign Number Representation

❑ Sign-Magnitude representation

MSB

N = | S | Magnitude = absolute value of N |

N - integer with *n* bits lies between $-(2^{n-1}-1)$ and $+(2^{n-1}-1)$

Decimal	Sign-magnitude
-8	-
-7	1111
-6	1110
-5	1101
-4	1100
-3	1011
-2	1010
-1	1001
0	1000 & 0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

```
         1010  - 2
  +      1100  - 4
Carry   10000
         0110  +6  → error
```

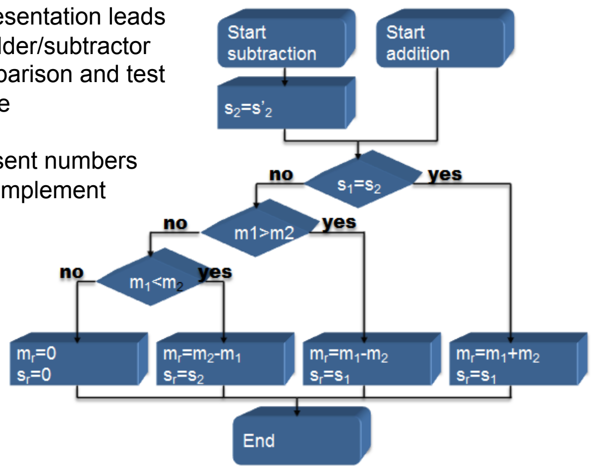
```
         0011  +3
  +      1011  -3
Carry    0110
         1110  -6  → error
```


1.3 Sign Number Representation

❑ Sign-Magnitude Numbers Addition and Subtraction

→ Sign-magnitude representation leads to slow, expensive adder/subtractor due to repeated comparison and test of sign and magnitude

→ This is why we represent numbers mostly using two's complement system



1.3 Sign Number Representation

- Two's Complement System

Radix-complement D* of a number D with n digits is

$D^* = r^n - D$ → $D^* + D = r^n$

Eg. The 2-complement of $D = 0011_2$ is

$D^* = 2^4 - 3 = 13 = 1101_2$

$$\begin{array}{rll} & 0011 & +3 \\ + & 1101 & (+3)_{2\text{-complement}} \Rightarrow \text{represents } (-3) \\ \text{Carry} & 11110 & \\ \hline & 0000 & 0 \rightarrow \text{Ok} \end{array}$$

→Two's Complement Calculation ?

1.3 Sign Number Representation

- Two's Complement System

Two's Complement Calculation:

Algorithm 1: Complement bits then add 1

$$\begin{array}{rrl} N = & 01100101 & \\ & 10011010 & \text{Complement the bits} \\ & +1 & \text{Add 1} \\ [N]_2 = & (10011011)_2 & \end{array}$$

Algorithm 2: Copy from LSB to the first 1-bit then continue replace the bits with their complement until the MSB has been replaced

$$\begin{array}{rcccccccc} N = & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ & & & & & & \updownarrow & & \\ [N]_2 = & (0 & 0 & 1 & 0 & 1 & 1 & 0 & 0)_2 \end{array}$$

1.3 Sign Number Representation

- Two's Complement System

Decimal	2-complement
-8	1000
-7	1001
-6	1010
-5	1011
-4	1100
-3	1101
-2	1110
-1	1111
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

MSB

+N = | 0 | Magnitude = absolute value of N |

2-complement calculation

-N = | 1 | |

N - integer with *n* bits lies between $-(2^{n-1}-1)$ and $+(2^{n-1}-1)$

1.3 Sign Number Representation

- Add and Sub in Two's Complement System

Addition

0010 +2	0010 +2	1110 - 2
0100 +4	1100 - 4	1100 - 4
+ 00000	+ 00000	+ 11000
0110 +6	1110 - 2	1010 - 6

Subtraction
A+(B)'+1

0010 +2	0010 +2	1110 - 2
1011 (+4)'	0011 (- 4)'	0011 (- 4)'
+ 00111	+ 00111	+ 11111
1110 - 2	0110 +6	0010 +2

Overflow

0111 +7	1001 - 7
0110 +6	1010 - 6
+ 01100	+ 10000
1101 - 3	0011 +3

1.3 Sign Number Representation

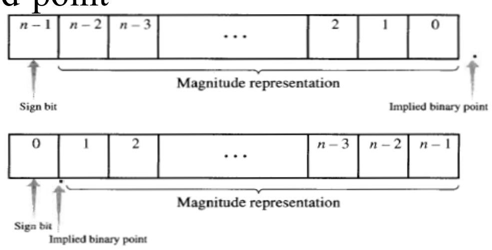
- Summary of Two's Complement Addition and Subtraction

Case*	Carry	Sign Bit	Condition	Overflow?
$B + C$	0	0	$B + C \le 2^{n-1} - 1$	No
	0	1	$B + C > 2^{n-1} - 1$	Yes
$B - C$	1	0	$B \le C$	No
	0	1	$B > C$	No
$-B - C$	1	1	$-(B + C) \ge -2^{n-1}$	No
	1	0	$-(B + C) < -2^{n-1}$	Yes

* B and C are positive numbers.

1.4 Real Number Code

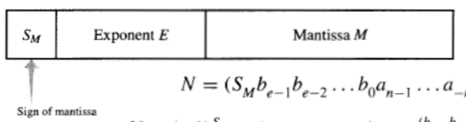
- Coding the position of the radix point
  - Fixed-point



  - Floating-point

Scientific notation

$N = M \times r^E$



$$N = (S_M b_{e-1} b_{e-2} \cdots b_0 a_{n-1} \cdots a_{-m})_r$$

$$N = (-1)^{S_M} \times (.a_{n-1} \cdots a_{-m})_r \times r^{(b_{e-1} b_{e-2} \ldots b_0) - 2^{e-1}}$$

1.4 Real Number Code

- Computer floating-point number

System/ Format	Total bits	Significand bits	Exponent bits	Exponent bias	Mantissa coding
IEEE Std. 754-1985:					Sign/Mag: (radix 2):
Single Precision	32	23 (+1)	8	127	$1 \le \|M\| < 2$
Double Precision	64	52 (+1)	11	1023	$1 \le \|M\| < 2$
IBM System/360:					Sign/Mag (radix 16):
Single Precision	32	24	7	64	$1/16 \le \|M\| < 1$
Double Precision	64	56	7	64	$1/16 \le \|M\| < 1$
DEC VAX 11/780:					Sign/Mag (radix 2):
F Format	32	23 (+1)	8	128	$1/2 \le \|M\| < 1$
D Format	64	55 (+1)	8	128	$1/2 \le \|M\| < 1$
G Format	64	52 (+1)	11	1024	$1/2 \le \|M\| < 1$
CDC Cyber 70:	60	48	11	1024	1's Complement (radix 2) $1 \le \|M\| < 2^{48}$

1.5 Binary Coded Decimal (BCD)

- Coding 10 decimal digits by 4 bits DCBA

Decimal digit	BCD DCBA
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Encode the decimal number $N = (9750)_{10}$ in BCD.

First, the individual digits are encoded from Table 1.10.

$$9 \rightarrow 1001,\ 7 \rightarrow 0111,\ 5 \rightarrow 0101,\ \text{and } 0 \rightarrow 0000$$

Then the individual codes are concatenated to give

$$N = (1001011101010000)_{BCD}$$

Problem : Add two BCD codes ?

1.6 Character Codes

- American Standard Code for Information Interchange (ASCII 7-bit code)

b3b2b1b0	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	‘	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	V
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

- Unicode

1.7 Gray Code

Decimal	Binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

00 → 01 → 11 → 10
10 → 11 → 01 → 00

→ Two consecutive number differ in only 1 bit (distance = 1)

Why do we use the gray code ?.

1.8 Error Detection Code
Error Correction Code

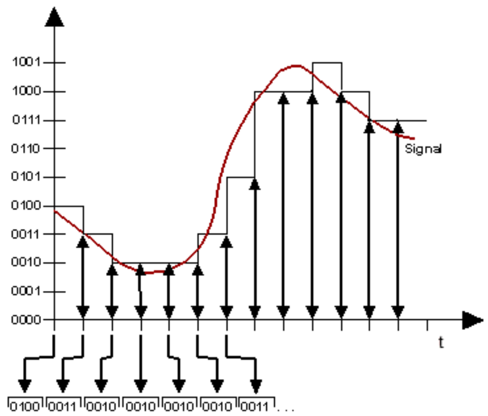
- Error ?
- Error Control: Error Detection and Error Correction
- Party Code
- Hamming Code
- Cyclic Redundancy Code (CRC-16, CRC-32)

1.9 Other Code

- Voice Encoding (Pulse Code Modulation)
- Image and Video Encoding (Pixels, Frames)
- Other information Encoding (ADC, DAC)

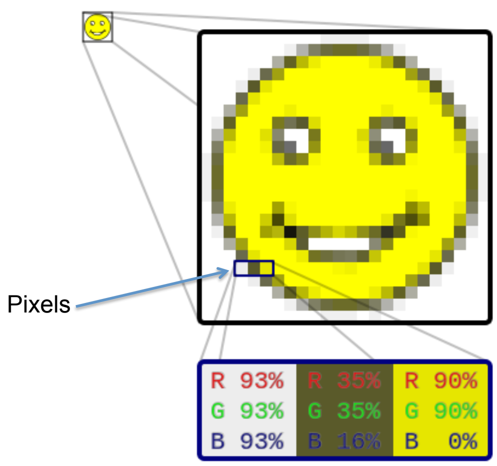
1.9 Other Code

- Voice Encoding (Pulse Code Modulation)



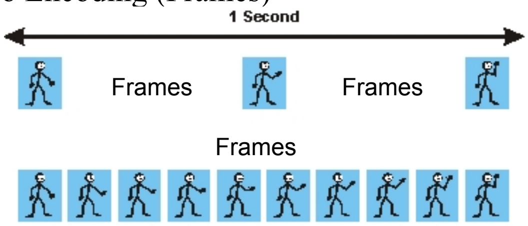
1.9 Other Code

- Image Encoding (Raster Image → Pixels)



1.9 Other Code

- Video Encoding (Frames)



Medium	fps
Film	24
European Video	25
American Video	30
European TV	50
American TV	60

1.9 Other Code

- ADC – Analog to Digital Converter