Digital Systems and Information

Hsi-Pin Ma 馬席彬

https://eeclass.nthu.edu.tw/course/3452

Department of Electrical Engineering

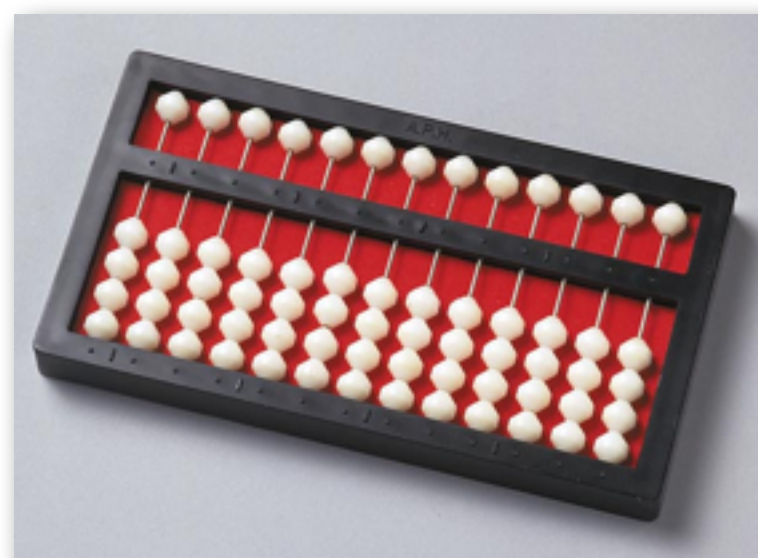
National Tsing Hua University

Outline

- Digital Systems
- Digital Signals
- Data Representation
  - Number Systems
    - Arithmetic Addition and Subtraction
  - Codes

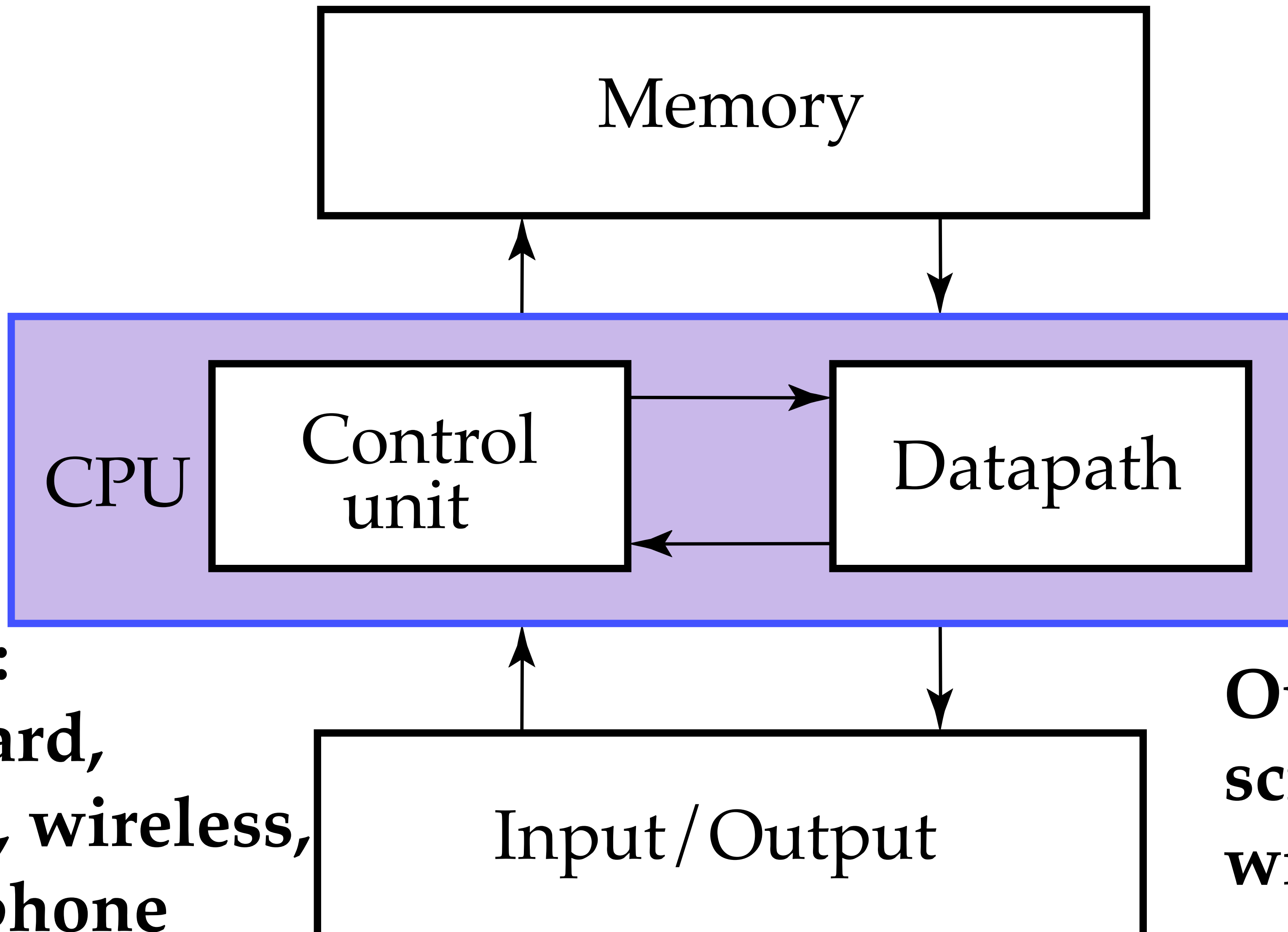
System

- A group of interacting, interrelated, or interdependent elements forming a complex whole
  – The American Heritage Dictionary



Digital Systems

A Digital Computer



Inputs: keyboard, mouse, wireless, microphone

Outputs: LCD screen, wireless, speakers

Synchronous or Asynchronous?

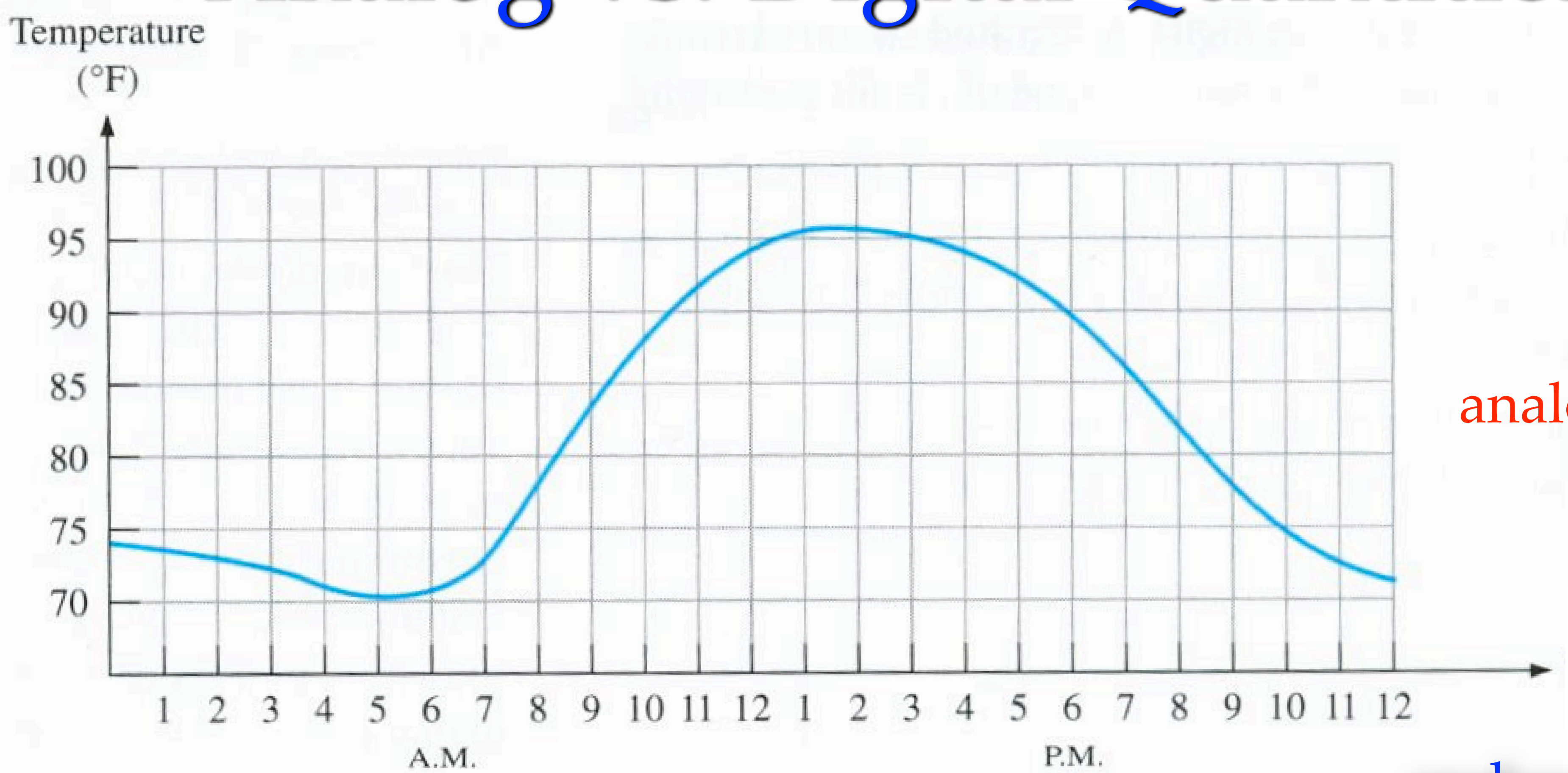
von Neumann architecture

Digital Logic Functions

- Information
  - represented as digital signals
- Logic function
  - computed by digital logic circuits
- Digital logic circuits
  - Combinational logics
    - output depends only on the current inputs
  - Sequential logics
    - output depends not only on the current inputs, but also in the internal states

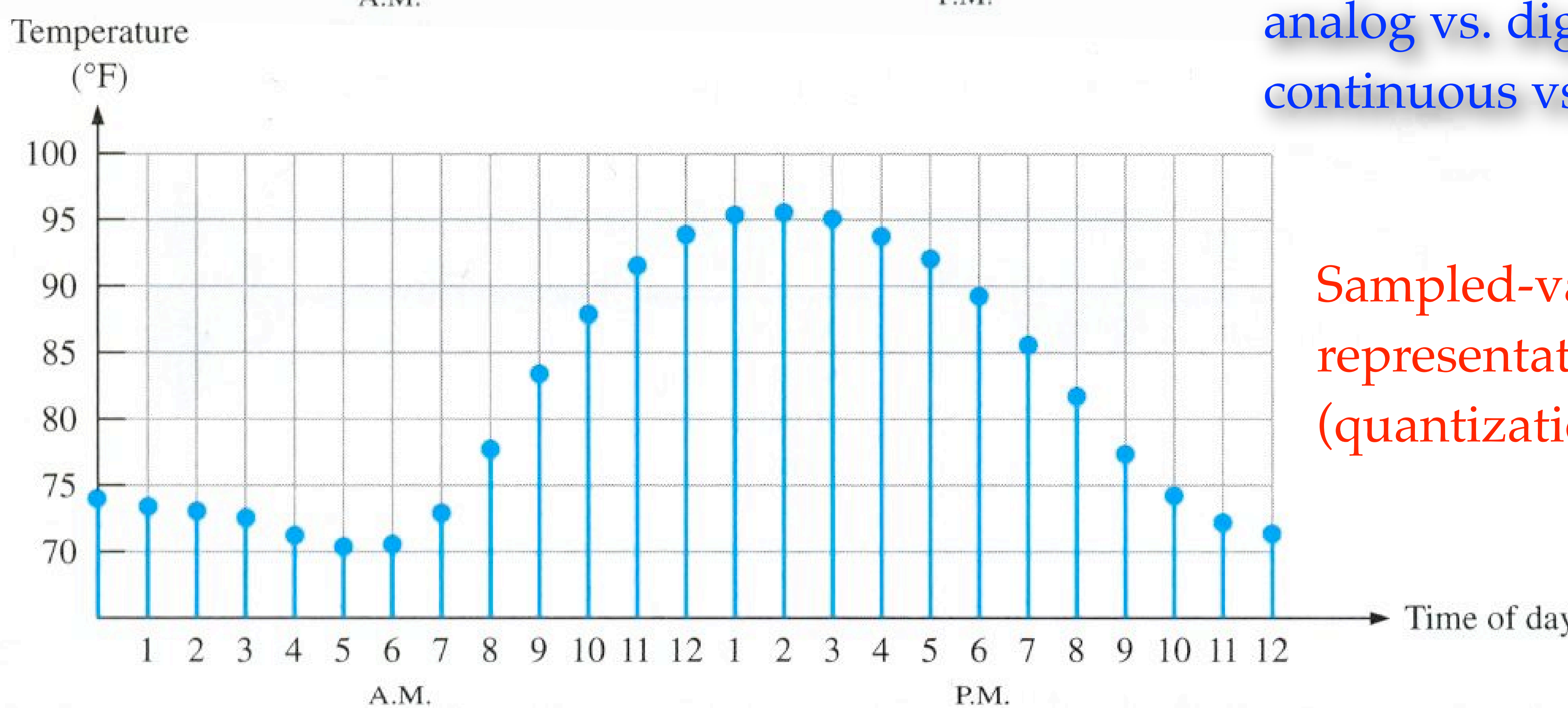
Digital Signals

Analog vs. Digital Quantities



analog quantity

analog vs. digital
continuous vs. discrete



Sampled-value representation (quantization)

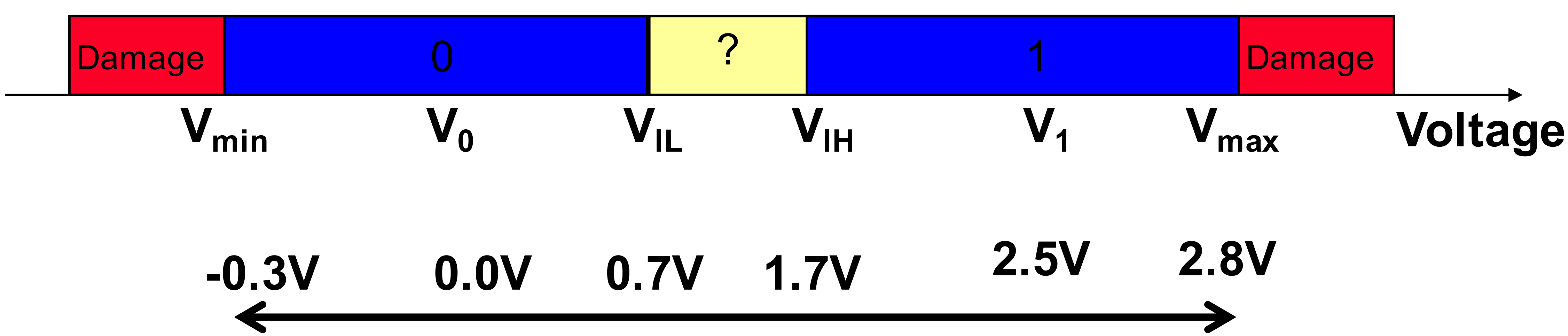
Digital Image

512x512

64x64

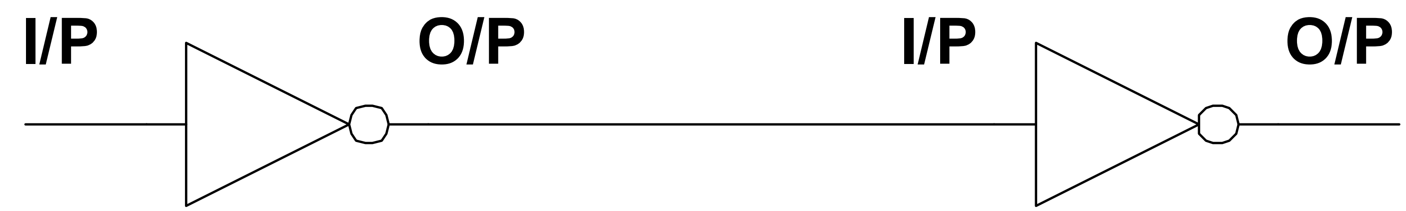
Encoding of Binary Signals for 2.5 V LVCMOS Logic

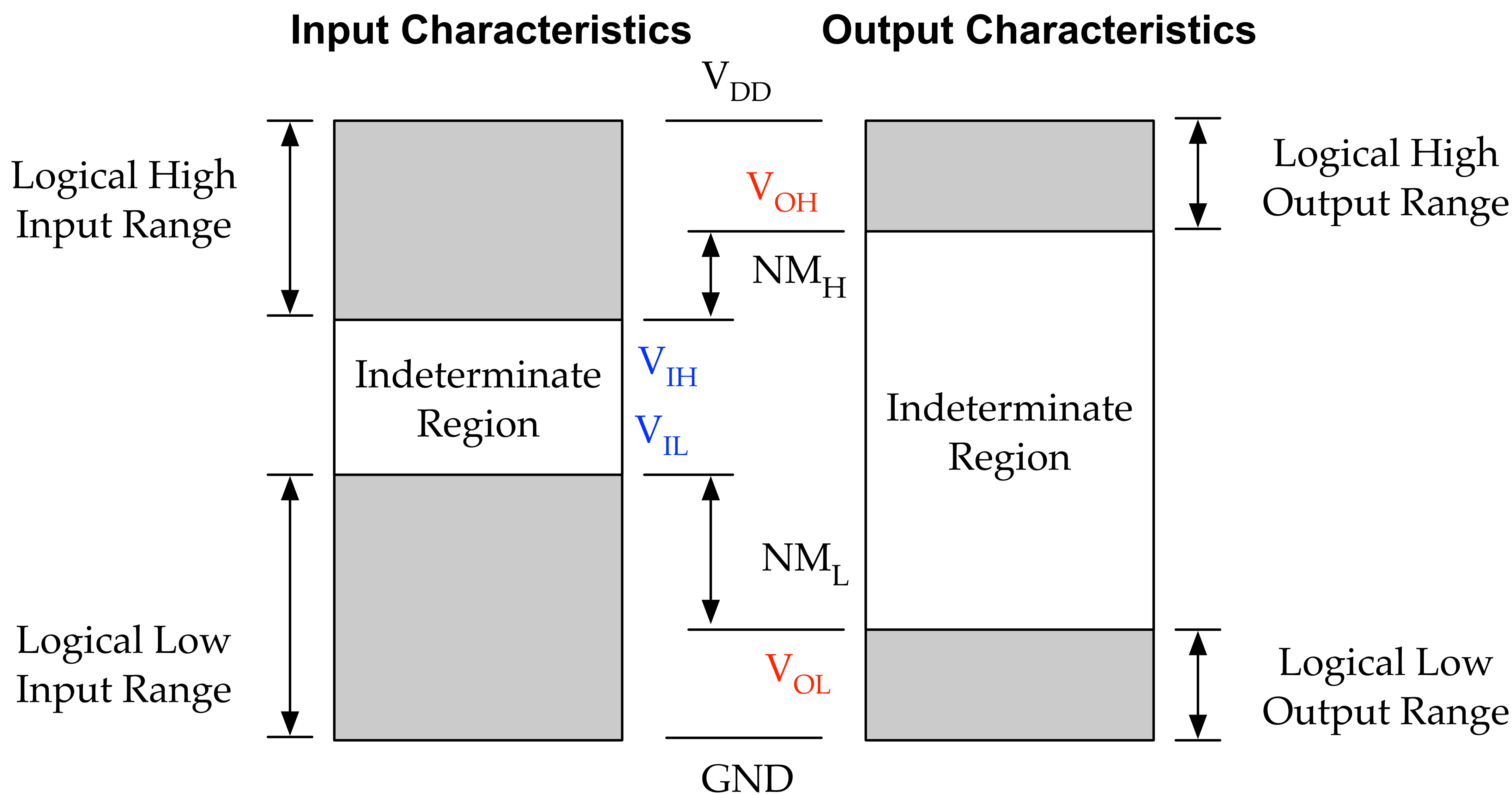
Parameter	Value	Description
V_{min}	-0.3 V	absolute minimum voltage below which damage occurs
V_0	0.0 V	nominal voltage representing logic "0"
V_{OL}	0.2 V	maximum output voltage representing logic "0"
V_{IL}	0.7 V	maximum voltage considered to be a logic "0" by a module input
V_{IH}	1.7 V	minimum voltage considered to be a logic "1" by a module input
V_{OH}	2.1 V	minimum output voltage representing logic "1"
V_1	2.5 V	nominal voltage representing logic "1"
V_{max}	2.8 V	absolute maximum voltage above which damage occurs



2.5V LVCMOS Logic

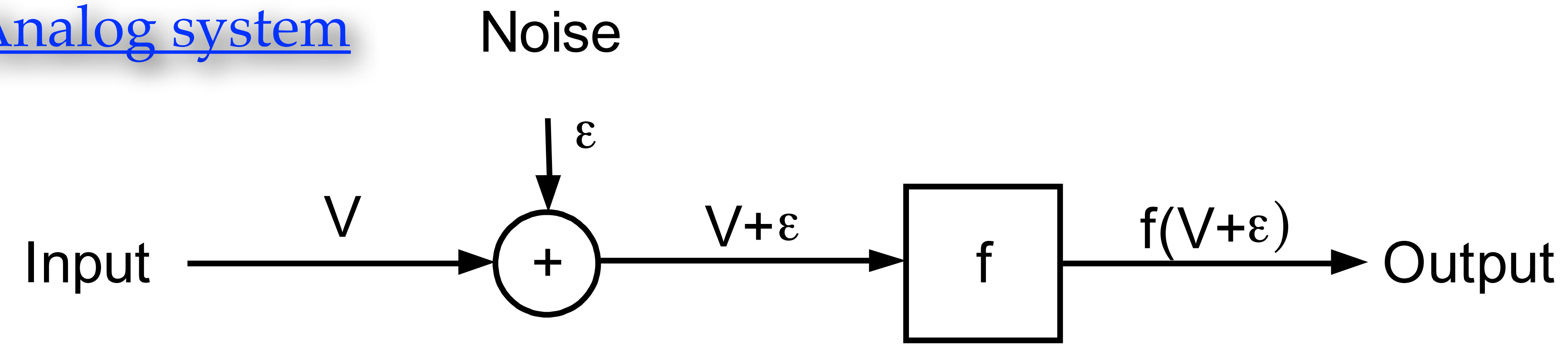
Noise Margins





Effects of Noise on Analog and Digital Signals

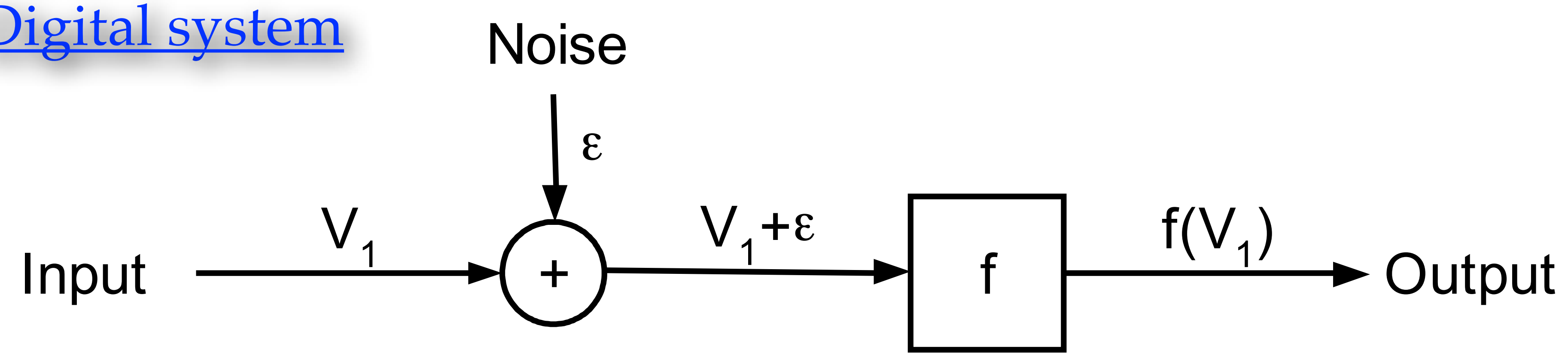
Analog system



Noise added to input

Error at output

Digital system



noise added to input < Noise Margin

Correct value at output

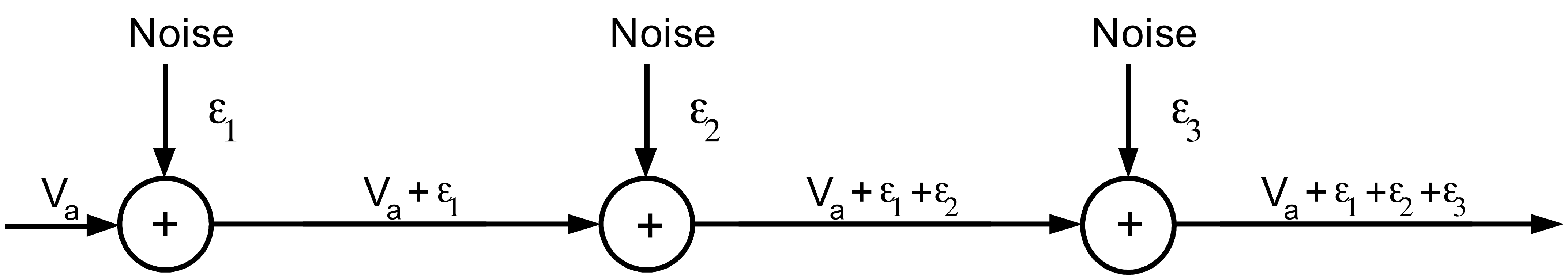
Digital Signals Tolerate Noise

Laboratory for Reliable Computing

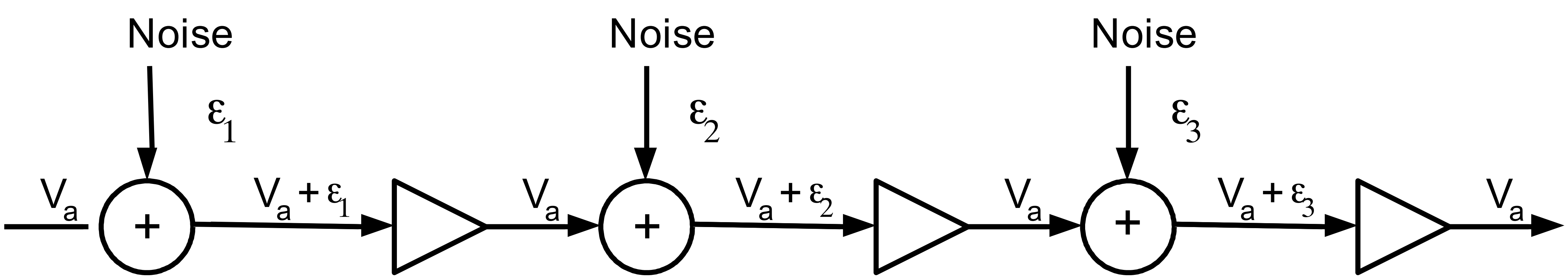
NTHU EE

Restoration of Digital Signals with Buffers

Noise Accumulation

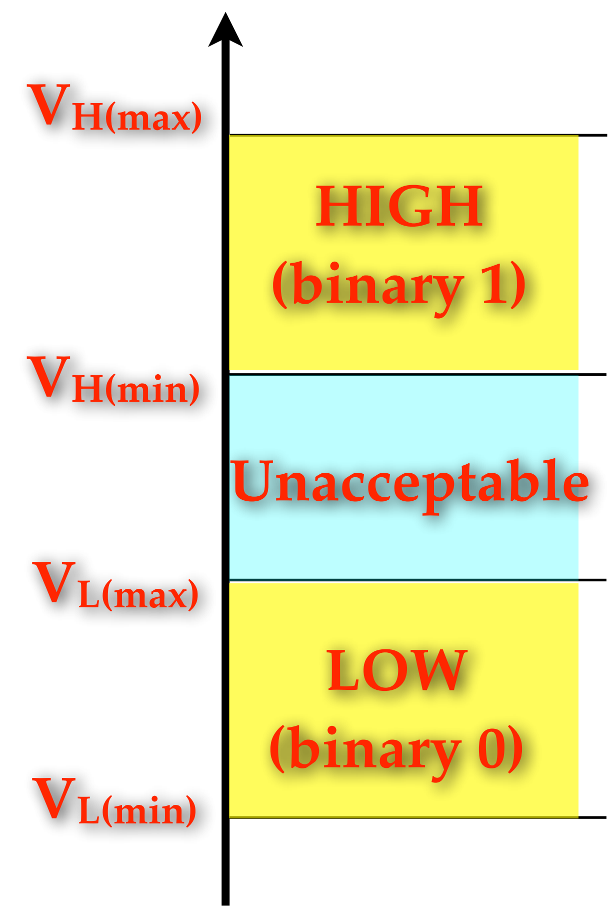


Signal Restoration



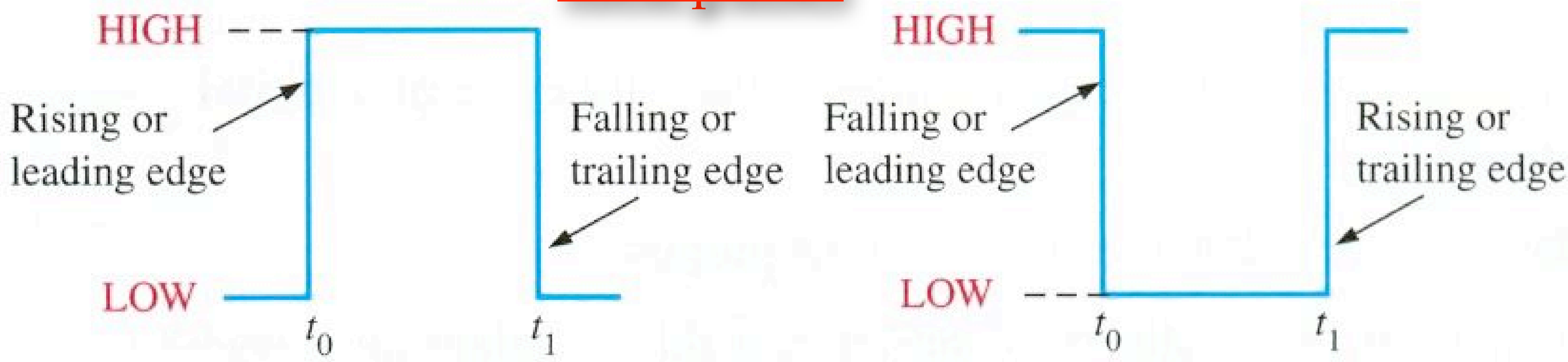
Binary Digits and Logic Levels

- Bit: ***bi***nary digi***t***
  - 1: HIGH (TRUE)
  - 0: LOW (FALSE)
- Codes: group of bits (combinations of 1s and 0s)
  - Used to represent numbers, letters, symbols, instructions, and anything else required in a given application.
- Logic levels

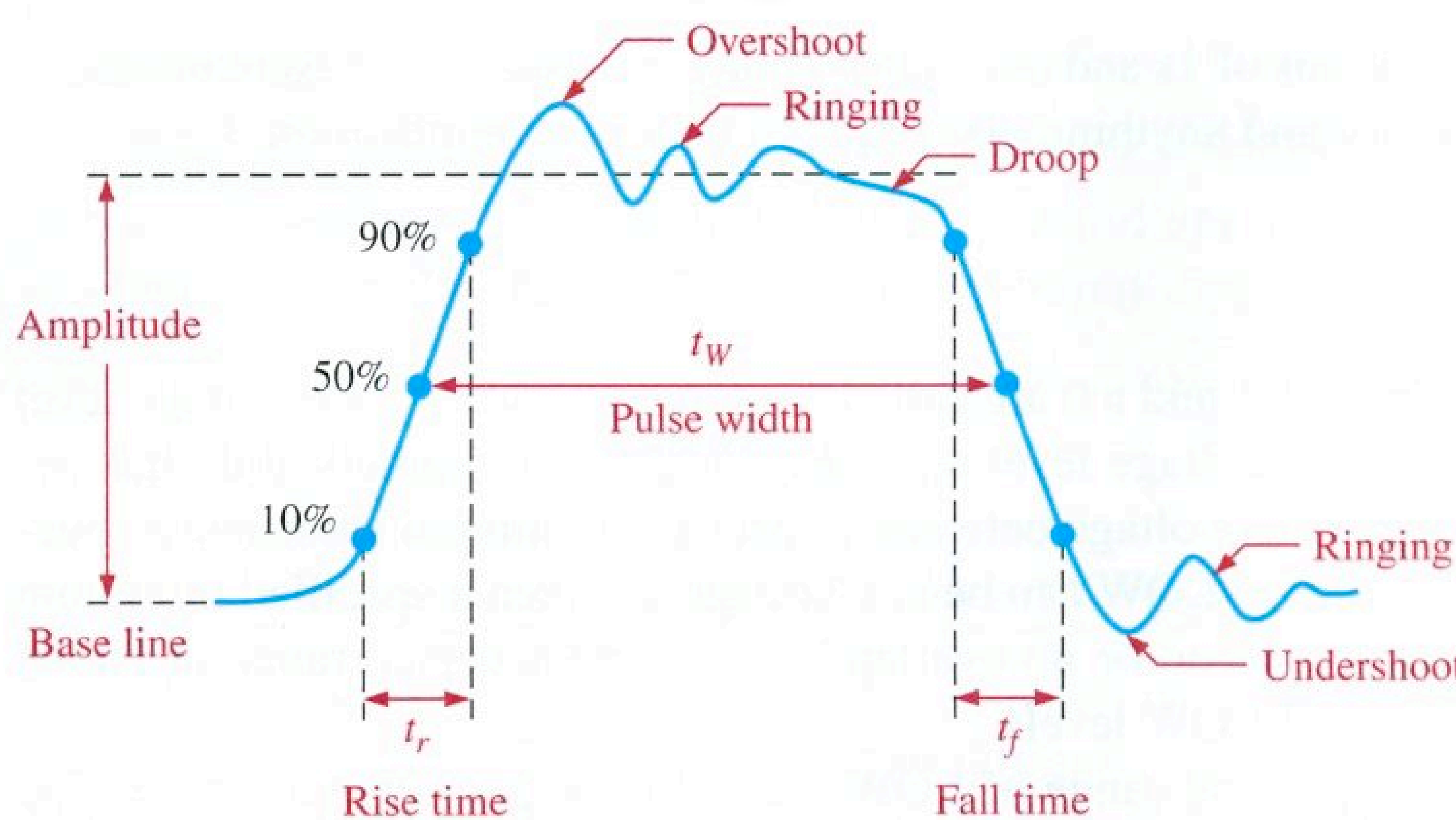


Digital Waveforms (1/2)

Ideal pulses

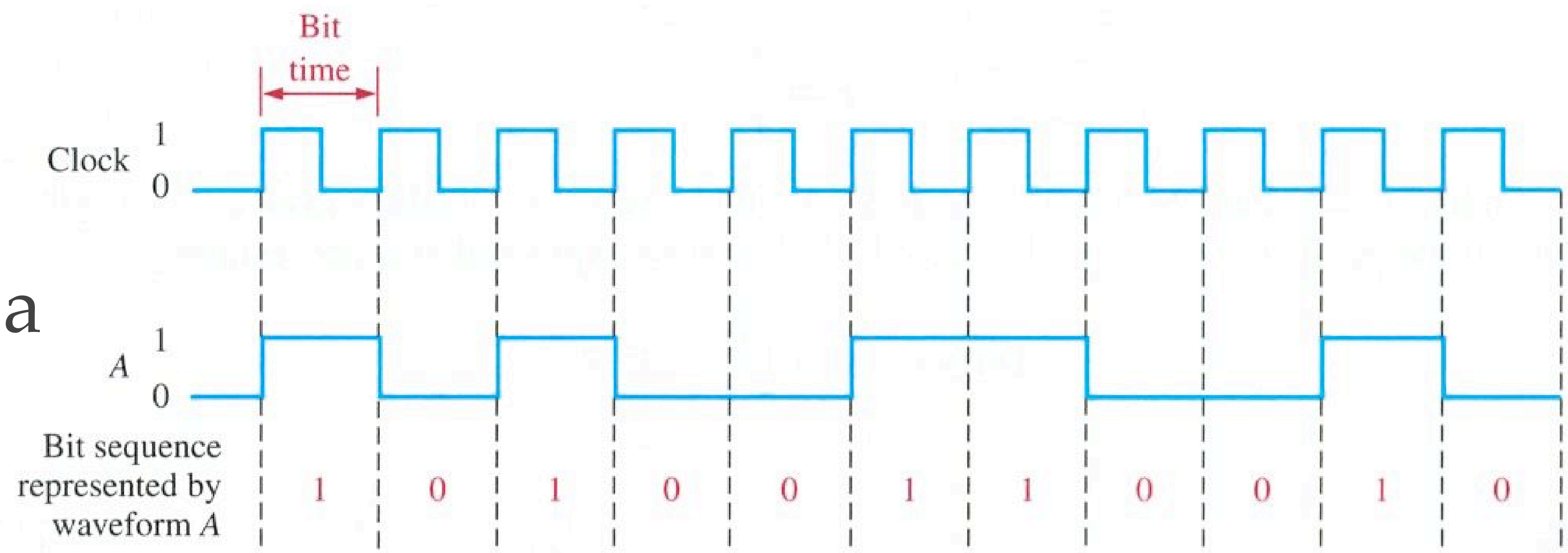


Nonideal pulses

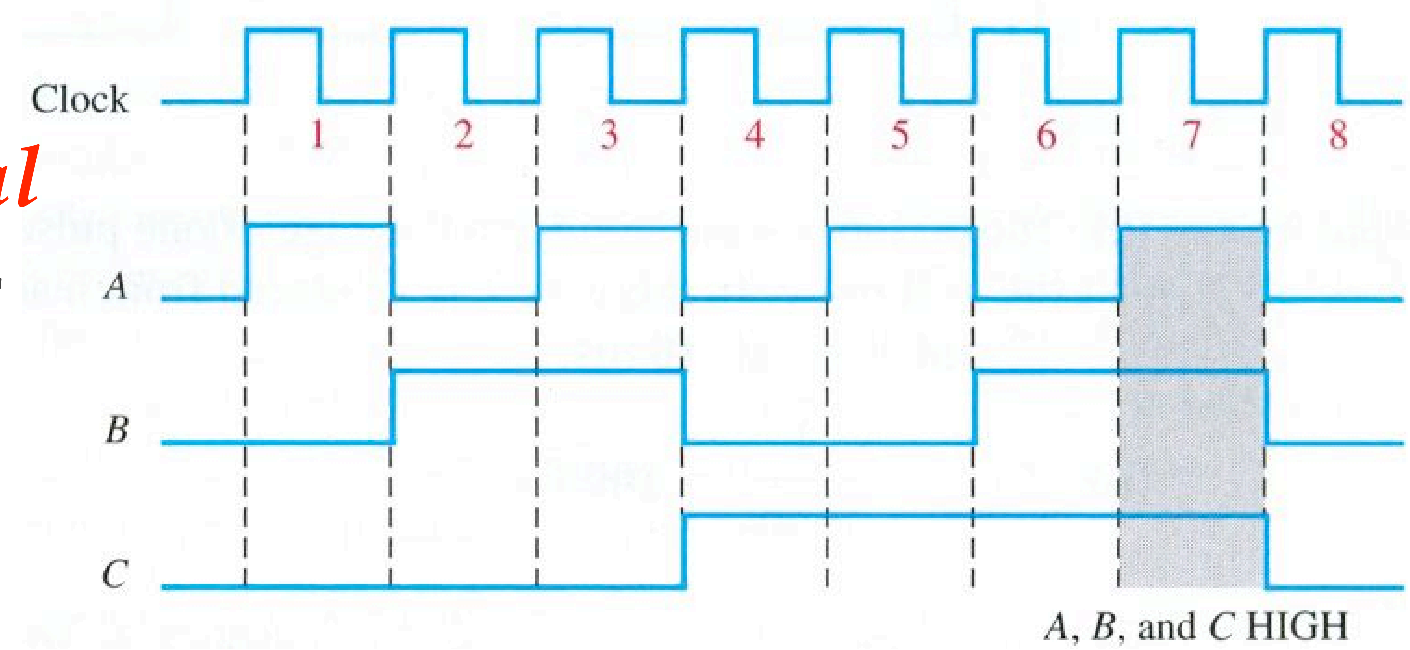


Digital Waveforms (2/2)

- Periodic vs. nonperiodic waveforms
  - frequency (f) vs. period (T) (f=1/T)
  - Duty cycle = (t_W/T) x 100%
- clock
  - All waveforms are synchronized with a basic timing waveform (***clock***).



- Timing diagram
  - A graph showing the ***actual time relationship*** of two or more waveforms and how each waveform ***changes in relation*** to others.



Data Representation

Data Representation (Data Types)

- Digital data can be categorized into
  - **Numbers**: used in arithmetic computation
  - **Letters of the alphabet**: used in data processing
  - **Discrete symbols**: used for variety of purposes
- All above are represented in binary-coded form
- Conversions between these data types and the binary code will be necessary

Number Systems

Positional Number Systems

- Let r be the radix (or base), then the $(n+m)$-digit number

$$D = d_{n-1}d_{n-2}\cdots d_1 d_0 . d_{-1}d_{-2}\cdots d_{-m} \qquad 0 \le d < r$$

radix point

 - has the value

$$D = d_{n-1}r^{n-1} + d_{n-2}r^{n-2} + \cdots + d_1 r + d_0 + d_{-1}r^{-1} + d_{-2}r^{-2} + \cdots + d_{-m}r^{-m}$$

Most-significant Digit (MSD)

Least-significant Digit (LSD)

$$D = \sum_{i=-m}^{n-1} d_i \cdot r^i$$

Positional Number Systems: Example

$$(7392)_{10} = 7\text{x}10^3 + 3\text{x}10^2 + 9\text{x}10^1 + 2\text{x}10^0$$

– Base (radix) r = 10

– Coefficients $D=(d_3,d_2,d_1,d_0) = (7,3,9,2)$

Binary Number System

- Let r=2, then the (n+m)-bit number

$$B = b_{n-1}b_{n-2}\cdots b_1 b_0.b_{-1}b_{-2}\cdots b_{-m}$$

  - has the value

$$B = b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \cdots + b_1 2 + b_0 + b_{-1}2^{-1} + b_{-2}2^{-2} + \cdots + b_{-m}2^{-m}$$

Most-significant Bit (MSB)

$$B = \sum_{i=-m}^{n-1} b_i \cdot 2^i$$

Least-significant Bit (LSB)

$$1010.101_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} = 10.625_{10}$$

Binary Special Unit

- byte: 8 bits
- word: processor architecture dependent (2, 4, 8 bytes or even more)
- 2^{10}: (1,024) Kilo, K
- 2^{20}: (1,048,576) Mega, M
- 2^{30}: (1,073,741,824) Giga, G
- 2^{40}: (1,099,511,627,776) Tera, T
- m, μ, n, f, ...

Symbol	Prefix	SI Meaning
m	milli	$10^{-3}=1000^{-1}$
μ	micro	$10^{-6}=1000^{-2}$
n	nano	$10^{-9}=1000^{-3}$
p	pico	$10^{-12}=1000^{-4}$
f	femto	$10^{-15}=1000^{-5}$
a	atto	$10^{-18}=1000^{-6}$
z	zepto	$10^{-21}=1000^{-7}$

Symbol	Prefix	SI Meaning	Binary Meaning
K	kilo	$10^{3}=1000^{1}$	$2^{10}=1024^{1}$
M	mega	$10^{6}=1000^{2}$	$2^{20}=1024^{2}$
G	giga	$10^{9}=1000^{3}$	$2^{30}=1024^{3}$
T	tera	$10^{12}=1000^{4}$	$2^{40}=1024^{4}$
P	peta	$10^{15}=1000^{5}$	$2^{50}=1024^{5}$
E	exa	$10^{18}=1000^{6}$	$2^{60}=1024^{6}$
Z	zetta	$10^{21}=1000^{7}$	$2^{70}=1024^{7}$

Octal and Hexadecimal Numbers

- The octal (base-8) and hexadecimal (base-16) numbers are shorter forms for representing binary numbers.
  - powers of two bases
  - conversion from binary to octal (hexadecimal) is straightforward -- by 3-bit (4-bit) grouping
  - conversion from octal (hexadecimal) to binary is just the reverse of the above.

Numbers with Different Bases

Decimal (base 10)	Binary (base 2)	Octal (base 8)	Hexadecimal (base 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Number Ranges

- The range of numbers that can be represented is based on the ***number of bits*** available in the hardware structures that store and process information.
  - 16-bit unsigned integers: 0 ~ 2^{16}-1 (0 ~ 65535)
  - 16-bit unsigned fractions: 0 ~ (2^{16}-1)/2^{16} (0 ~ 0.9999847412)

Radix-*r* to Decimal Conversion

$$D = d_{n-1}r^{n-1} + d_{n-2}r^{n-2} + \cdots + d_1 r + d_0 + d_{-1}r^{-1} + d_{-2}r^{-2} + \cdots + d_{-m}r^{-m}$$

Most-significant Digit (MSD)

Least-significant Digit (LSD)

$$1010.101_2 = 1{\cdot}2^3 + 0{\cdot}2^2 + 1{\cdot}2^1 + 0{\cdot}2^0 + 1{\cdot}2^{-1} + 0{\cdot}2^{-2} + 1{\cdot}2^{-3} = 10.625_{10}$$

$$22.22_4 = 2{\cdot}4^1 + 2{\cdot}4^0 + 2{\cdot}4^{-1} + 2{\cdot}4^{-2} = 10.625_{10}$$

$$12.5_8 = 1{\cdot}8^1 + 2{\cdot}8^0 + 5{\cdot}8^{-1} = 10.625_{10}$$

$$A.A_{16} = 10{\cdot}16^0 + 10{\cdot}16^{-1} = 10.625_{10}$$

Decimal to Radix-r Conversion

- Integer part: Successive divisions by r and observe the remainders
- Fraction: Successive multiplications by r and observe the carries

$$A_r = D_{10} \quad \sum_{i=-m}^{n-1} a_i \cdot r^i = \sum_{j=-p}^{q-1} d_j \cdot 10^j \qquad 0 \le a_i < r \qquad 0 \le d_i < 10$$

$$D_{10} = D1_{10} + D2_{10} \qquad \sum_{j=-p}^{q-1} d_j \cdot 10^j = \underbrace{\sum_{j=0}^{q-1} d_j \cdot 10^j}_{D1} + \underbrace{\sum_{j=-p}^{-1} d_j \cdot 10^j}_{D2}$$

Integer part

$$\begin{aligned} D1 &= D1' \cdot r + a_0 \\ D1' &= D1'' \cdot r + a_1 \\ &\cdots \\ D1^{(n-2)} &= D1^{(n-1)} \cdot r + a_{n-2} \\ D1^{(n-1)} &= a_{n-1} \end{aligned}$$

Fractional part

$$\begin{aligned} D2 \cdot r &= a_{-1} . D2' \\ D2' \cdot r &= a_{-2} . D2'' \\ &\cdots \\ D2^{(m-1)} \cdot r &= a_{-m} . D2^{(m)} \end{aligned}$$

Arithmetic Addition and Subtraction

Addition & Subtraction of Binary Number

Addition

		512	256	128	64	32	16	8	4	2	1
x		1	1	1	1	0	1	1	0	1	1
y					1	1	1	1	0	1	1
Carries		1	1	1	1	1	1	0	1	1	
x+y	1	0	0	0	1	0	1	0	1	1	0
	s_{10}	s_9	s_8	s_7	s_6	s_5	s_4	s_3	s_2	s_1	s_0

Subtraction

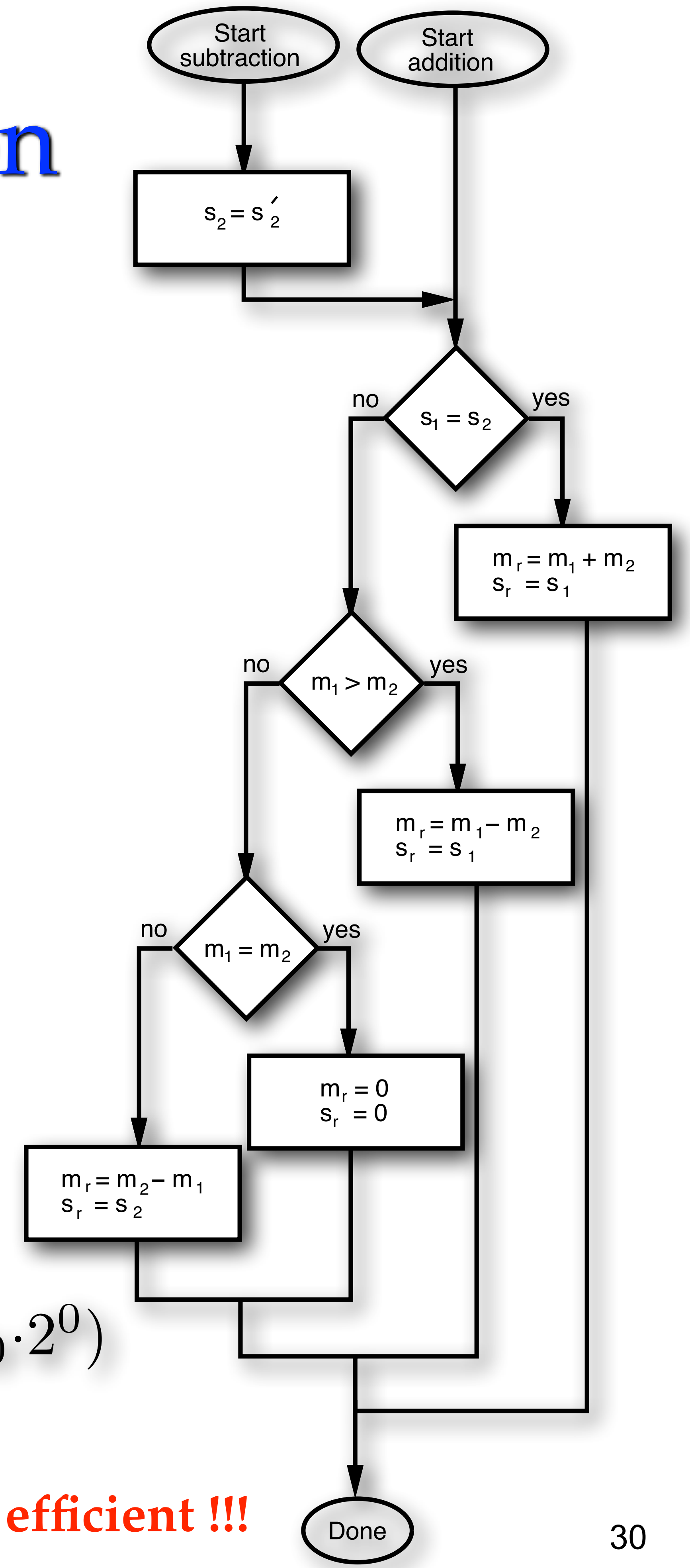
	512	256	128	64	32	16	8	4	2	1
x	1	1	1	1	0	1	1	0	1	1
y				1	1	1	1	0	1	1
Borrow	0	0	1	1	0	0	0	0	0	
x-y	1	1	0	1	1	0	0	0	0	0
	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0

Sign-Magnitude Representation

- D=<*s*,*m*>
  - *s*: sign, + (0) or - (1)
  - For an *n*-bit integer, *m* is an integer ranging from 0 to 2^{n-1}-1
- Assume we want to add / subtract D1 with D2
  - D1=<s_1,m_1>
  - D2=<s_2,m_2>

$$D = sm_{n-2}m_{n-3}...m_1m_0 = \pm(m_{n-2}\cdot 2^{n-2} + m_{n-3}\cdot 2^{n-3} + ... + m_1\cdot 2^1 + m_0\cdot 2^0)$$

  - 0111111 = +63, 1111111 = -63



Not efficient !!!

Complements

- Complements are used for simplifying the subtraction operation for easy manipulation of certain logical rules and events
  - Trade comparisons of sign and magnitude with complementation
  - Complementation can be performed very efficiently for binary numbers
- Two types for radix-*r* system
  - Radix complement (*r*'s-complement)
  - Digit complement (diminished radix complement and (*r*-1)'s-complement)

Two Types of Complements

- Radix complement
  - The r's-complement of an n-digit number D is defined as 0 if D=0, and else
    $$\bar{D} = D' + 1 = (r^n) - D$$
  - 10's-complement of 546700=1000000-546700=453300
  - 10's-complement of 012398=1000000-012398=987602
  - 2's-complement of 1011000=10000000-1011000=0101000
  - 2's-complement of 0101101=10000000-0101101=1010011
- Digit complement
  - The (*r*-1)'s-complement of an n-digit number D
    $$D' = (r^n - 1) - D$$
  - 9's-complement of 546700=999999-546700=453299
  - 9's-complement of 012398=999999-012398=987601
  - 1's-complement of 1011000=1111111-1011000=0100111
  - 1's-complement of 0101101=1111111-0101101=1010010

10's Complement Example

- Definition of sign
  - Positive number: MSD with 0
  - Negative number: MSD with 9
  - MSD with other numbers => illegal
- 9286-1801 (both unsigned decimal)
  - 10's complement of 1801 : 10000-1801=8199
  - 09286+98199=107485 (remove end carry) => 07485
- Still need minus operation in complement!!
- How to avoid??

2's Complement Example

- Sign definition
  - Positive number: MSB with 0
  - Negative number : MSB with 1
  - Leading bit with negative weight (provide half+/half-)
- 1111-1010 (both unsigned binary)
  - 2's complement of 1010: 100000-01010=10110
  - 01111+10110=100101 (leading 1 issue)
- Use 1's complement + 1 to remove the extra 'minus'

2's-Complement Representation

- Signed vs. Unsigned (*n*-bit binary number)

Unsigned binary representation

$$B = b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \cdots + b_1 2 + b_0$$

2's-complement binary representation

$$B = b_{n-1}(-2^{n-1}) + b_{n-2}2^{n-2} + \cdots + b_1 2 + b_0$$

- Example
  - 0111_2: 7 for unsigned and 2's complement number
  - 1111_2: 15 for unsigned number, -1 for 2's complement number (-1*8+1*4+1*2+1*1=-1)

Representation of Signed Binary Numbers

Decimal	Signed-2's Complement	Signed-1's Complement	Signed Magnitude
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
−0	—	1111	1000
−1	1111	1110	1001
−2	1110	1101	1010
−3	1101	1100	1011
−4	1100	1011	1100
−5	1011	1010	1101
−6	1010	1001	1110
−7	1001	1000	1111
−8	1000	—	—

- Range of an n-bit number in 2's-complement representation is $[-2^{n-1}, 2^{n-1}-1]$
- The *2's-complement representation* is by far the most popular.

Subtraction with Complements

- Replace subtraction with addition
- M_r-N_r
  - $M+(r^n-N) = M-N + r^n$
  - If M>=N, the ***end carry*** r^n is discarded, and the result is M-N
  - If M<N, there is no end carry, and the sum equals r^n - (N-M). Take its r's-complement we obtain N-M, i.e., -(M-N)

2’s-Complement Subtraction

- Let the 1’s-complement form of an n-bit number B be denoted as B’, then
  - $B + B' = 2^n - 1$; $B' + 1 = 2^n - B$
  - $-B = B' + 1 =$ 2’s-complement of B
- A-B=A+(2’s-complement of B)

2's-Complement Addition

- Adding two positive numbers generates correct results if there is no ***overflow***
  - 0010+0100=0110 (2+4=6)
- Adding two positive numbers generates incorrect results if there is overflow
  - 0110+0101=1011 (6+5=-5)
- Adding two negative numbers generates correct results if there is no ***underflow***
  - 1110+1100=1010 ((-2)+(-4)=(-6))
- Adding two negative numbers generates incorrect result if there is underflow
  - 1100+1011=0111((-4)+(-5)=7)
- Sign extension to avoid overflow or underflow

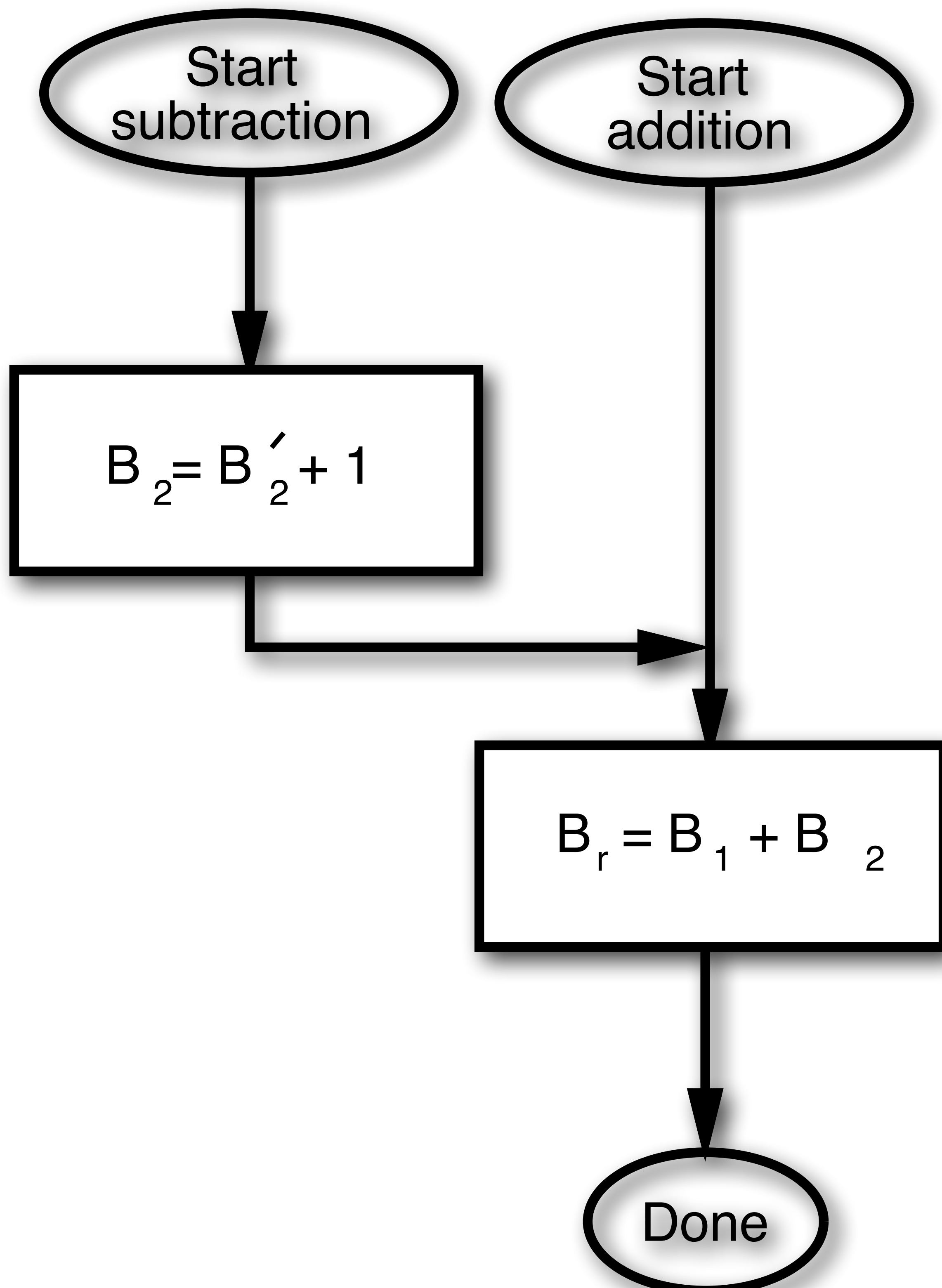
NTHU EE

Laboratory for Reliable Computing

Bit Insertion for Addition

- When doing A+B ($a_3a_2a_1a_0+b_3b_2b_1b_0$)
  - If A and B are unsigned numbers, add two bits to the beginning, then do summation.
    - One bit to convert unsigned number to signed number, and the other bit for sign extension
    - $00a_3a_2a_1a_0+00b_3b_2b_1b_0$
  - If A and B are signed numbers, only add one bit for sign extension to avoid overflow.
    - $a_3a_3a_2a_1a_0+b_3b_3b_2b_1b_0$

Radix-r Addition/Subtraction



Codes

Decimal Codes

- An *n*-bit binary code is a group of *n* bits that assume up to 2^n distinct combinations of 1s and 0s, with each combination representing one element of the set being coded.
  - Each element must be assigned a **unique** binary bit combination to **avoid ambiguity**
  - Example
    - 2-bit binary code: 00, 01, 10, 11
    - 3-bit binary code: 000, 001, 010, ..., 111
    - *n*-bit code: $0 \sim 2^n-1$
  - May have unassigned bit combinations

Binary-Coded Decimal (BCD)

- Represent the decimal system using binary number
  - 4 bits to represent 0-9 in the decimal system
  - A-F are discarded
  - $(185)_{10} = (0001\ 1000\ 0101)_{BCD}$
- seven-segment display

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

a nibble

Binary Codes for Decimal Numbers

Decimal Digit	BCD 8421	2421	Excess-3	8, 4, −2, −1
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010	0101	0110
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1011
6	0110	1100	1001	1010
7	0111	1101	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111
Unused bit combi-nations	1010	0101	0000	0001
	1011	0110	0001	0010
	1100	0111	0010	0011
	1101	1000	1101	1100
	1110	1001	1110	1101
	1111	1010	1111	1110

Binary Codes for Decimal Numbers

- Weighted codes
  - Each position is assigned a weighting factor to calculate the value of the number
  - BCD (8421), 2421, 84-2-1 codes
- Self-complementing codes
  - 9's complement of a decimal number is obtained directly by changing 1 to 0 or 0 to 1 in the code
  - 2421, excess-3 codes

Number of Bits Required to Represent a Binary Code

- Given M elements to be represented by a binary code, the minimum number of bits, ***n***, needed satisfies the following relationships
  - $2^{(n-1)} < M \leq 2^n \qquad n = \lceil \log_2 M \rceil$

Warning: Conversion vs. Coding

- Do NOT mix up conversion of a decimal number to a binary number with coding a decimal number with a BINARY CODE
  - $13_{10} = 1101_2$ (Conversion)
  - 13 <=> 0001 0011 (BCD Coding)

Alphanumeric Codes

- Represent numerals and special characters with binary codes in many other applications.
- Alphanumeric character set for English
  - Ten decimal digits
  - 26 letters of the alphabet
  - Several (more than three) special characters

ASCII Character Code

- American Standard Code for Information Interchange
  - The standard binary code for the alphanumeric characters
  - ASCII code is not enough for some languages, and 2-byte code is necessary, such as Chinese Big5 or Unicode

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Parity Bit

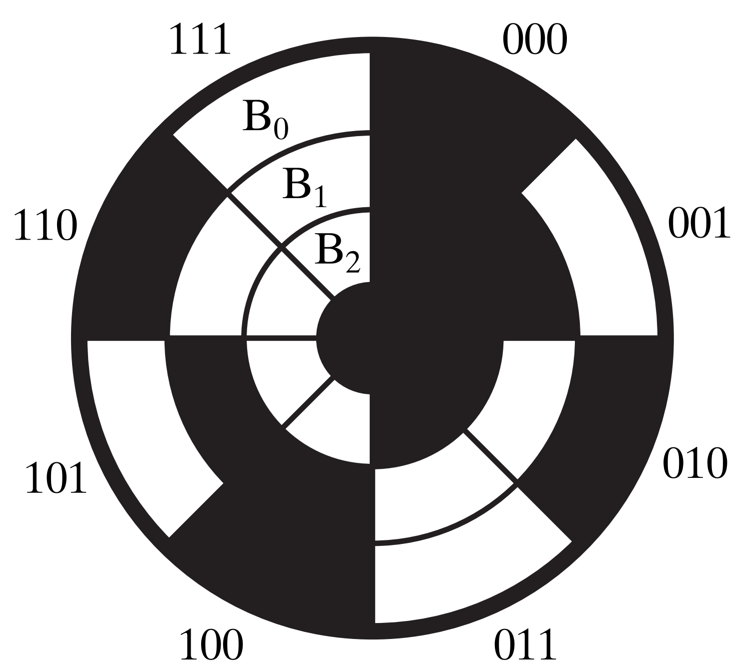
- Error detection
  - *Redundancy*, in the form of extra bits, can be incorporated into binary code words to detect and correct errors
  - *Parity* is an extra bit appended on to the codeword to make the number of 1s odd or even. Parity can detect all single-bit errors and some multiple-bit errors.
    - A code word has *even parity* if the number of 1s in the code word is even.
    - A code word has *odd parity* if the number of 1s in the code word is odd.

	With Even Parity	With Odd Parity
1000001	01000001	11000001
1010100	11010100	01010100

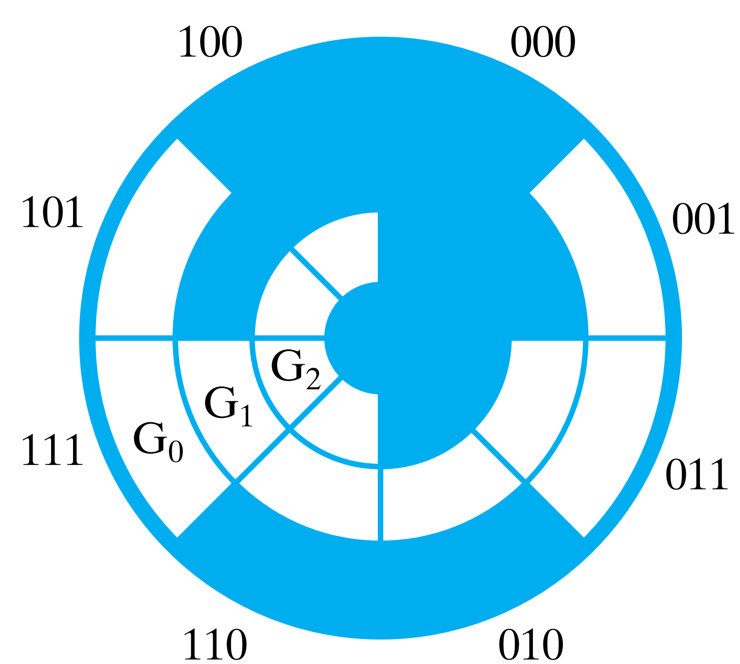
Gray Codes

- A binary code in which adjacent code words differ in only one bit position

Binary Code	Bit Changes	Gray Code	Bit Changes
000		000	
	1		1
001		001	
	2		1
010		011	
	1		1
011		010	
	3		1
100		110	
	1		1
101		111	
	2		1
110		101	
	1		1
111		100	
	3		1
000		000	



(a) Binary Code for Positions 0 through 7



(b) Gray Code for Positions 0 through 7

Gray Codes

1-bit GC	2-bit GC	3-bit GC	4-bit GC
0	00	000	0000
1	01	001	0001
	11	011	0011
	10	010	0010
		110	0110
		111	0111
		101	0101
		100	0100
			1100
			1101
			1111
			1110
			1010
			1011
			1001
			1000

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

Generation of Gray Codes

- Code number should be even (M=2k) number of code words

$$D = \sum_{i=0}^{n-1} d_i \cdot 2^i \longrightarrow G = g_{n-1}g_{n-2}g_{n-3}\cdots g_1 g_0$$

- For the first half $M/2$ codes
  - Let MSB=0
  - Replace each of the remaining bits with the even parity of the bit of the number and the bit to its left $g_i = d_{i+1} \oplus d_i, i = 0, 1, \ldots, n-2$
- For the rest half codes
  - Take the sequence of numbers formed for the first half and copy it in reverse order but with MSB=1