ENGG2020 DIGITAL LOGIC AND SYSTEMS

CHAPTER 1: DIGITAL SYSTEMS AND BINARY NUMBERS

By Dr. Anthony Sum
Department of Computer Science and Engineering
The Chinese University of Hong Kong

CONTENTS

- Analog vs. Digital
- Number Systems and their conversions
- Binary Addition and Subtraction
- Floating Point Representation
- BCD and Gray Codes
- Error Detection and Correction

3

ANALOG VS DIGITAL

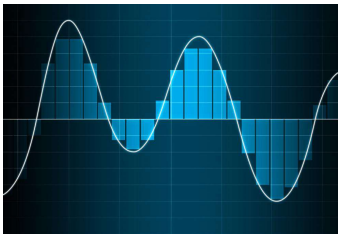
ANALOG VS. DIGITAL

- Anything seems become "digital" nowadays, why is that?
- Is it good? Or bad?
- If it is not, why not stay with "analog"?
- What is "analog"?

4

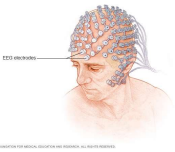
INTRODUCTION TO ANALOG WORLD

- Nearly everything we can feel, hear, touch, and sense is analog signals
  - We hear voice or sound waves
  - We feel heat or temperature
  - We see image or light waves
- These signals are analog and are continuous functions of time
  - They have value at any point in time

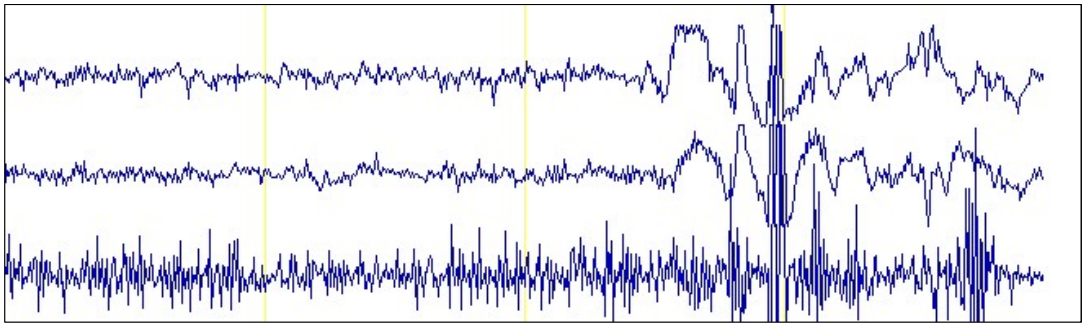


5

ANALOG SIGNAL



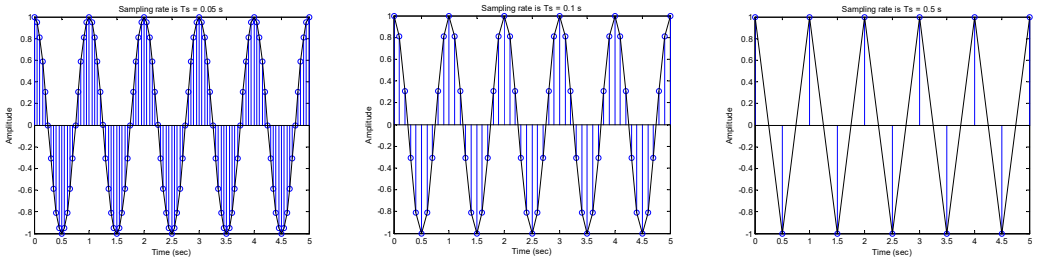
- For example, the electroencephalography (EEG) signal produced by the brain



6

DISCRETE TIME SAMPLES

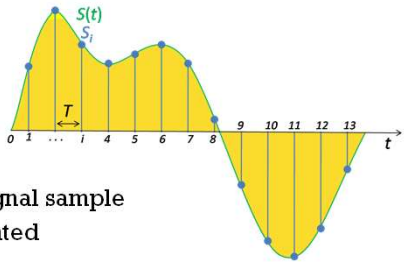
- Unlike analog signals, digital signals are actually discrete samples
  - They are not continuous
  - They are only available at a certain period of time, known as sampling period, Ts



7

RESOLUTION

- Number of Samples
  - The original signal can be reconstructed by interpolation between consecutive discrete samples
  - The more samples we have, the better signal can be restored

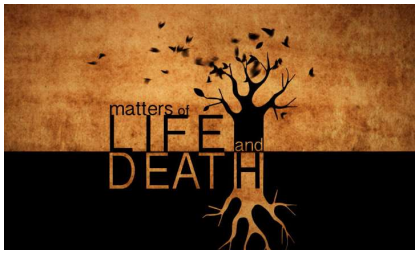


- Resolution
  - Resolution refers to the number of bits used to represent a signal sample
  - The higher resolution, the more accurate value can be presented

8

FIDELITY

- For example, if a medical device can display only 0.2mV difference in a vital signal, and gives a reading of 0mV…
- Does it really mean 0mV? OR anything between -0.2mV to 0.2mV?
- It could distinguish between life and death of a patient



- Should we go for digital then…?
- Should I drop the course…?

9

THE BRIGHT SIDE OF DIGITAL WORLD

- Digital circuits are relatively easy to design
- Digital circuits have higher accuracy, and programmability
- Digital signals can be stored easily
- Digital circuit is comparatively more immune to error and noise, error detection and error correction can be implemented
- Digital signal transmission will not be degraded over a long distance
- Digital signal is either high or low, hence there is less chance or confusion
- Digital circuits have higher flexibility, we can change the functionality by software
- Digital circuits are more reliable, analog signal can be distorted by thermal noise

10

11

NUMBER SYSTEMS AND BASE CONVERSIONS

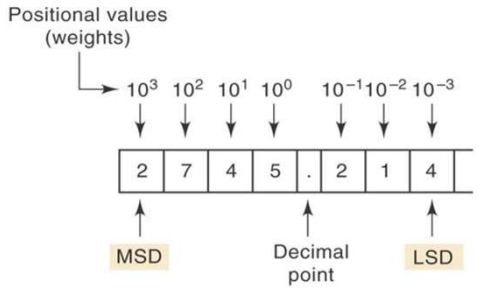
COMMON NUMBER SYSTEMS

- Decimal in base 10
- Binary in base 2
- Hexadecimal in base 16
- Octal in base 8

12

DECIMAL

- Decimal digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Most Significant Digit (MSD)
- Least Significant Digit (LSD)



13

DECIMAL NUMBERS

- Decimal digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

- What does a decimal number 1234_{10} means?
- $1234_{10} = 1 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$

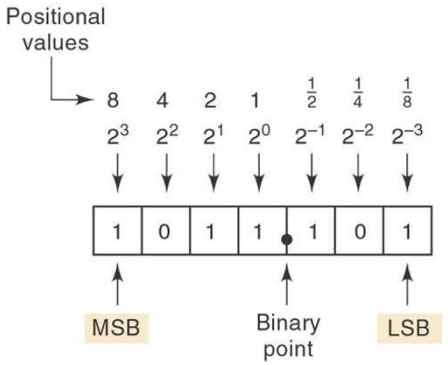
- What is 56.78_{10}?
- $56.78_{10} = 5 \times 10^1 + 6 \times 10^0 + 7 \times 10^{-1} + 8 \times 10^{-2}$

14

BINARY NUMBERS

- Binary digits: 0, 1
- Most Significant Bit (MSB)
- Least Significant Bit (LSB)

- Binary to Decimal:
- $1011_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
- $1.01_2 = 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2}$



15

DECIMAL TO BINARY

- By reversing the process…

$$45_{10} = 32 + 8 + 4 + 1 = 2^5 + 0 + 2^3 + 2^2 + 0 + 2^0$$
$$= 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1_2$$

$$76_{10} = 64 + 8 + 4 = 2^6 + 0 + 0 + 2^3 + 2^2 + 0 + 0$$
$$= 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0_2$$

16

DECIMAL TO BINARY

- By repeated divisions…
  - Divide the decimal number by 2
  - Write down the remainder after each division
  - Until a quotient of zero is obtained
- The first remainder is the LSB
- The last remainder is the MSB
- For example, $25_{10} = 11001_2$

$\frac{25}{2} = 12$ + remainder of 1 — LSB

$\frac{12}{2} = 6$ + remainder of 0

$\frac{6}{2} = 3$ + remainder of 0

$\frac{3}{2} = 1$ + remainder of 1

$\frac{1}{2} = 0$ + remainder of 1 — MSB

$25_{10} = 1\ 1\ 0\ 0\ 1_2$

17

DECIMAL TO BINARY

- For example, $37_{10} = 100101_2$

$\frac{37}{2} = 18.5 \longrightarrow$ remainder of 1 (LSB)

$\frac{18}{2} = 9.0 \longrightarrow$ 0

$\frac{9}{2} = 4.5 \longrightarrow$ 1

$\frac{4}{2} = 2.0 \longrightarrow$ 0

$\frac{2}{2} = 1.0 \longrightarrow$ 0

$\frac{1}{2} = 0.5 \longrightarrow$ 1 (MSB)

18

HEXADECIMAL

- Hexadecimal digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
- Useful in handling long binary strings
- By dividing them into 4-bit groups

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

4 bits

Additional to Decimal

19

HEXADECIMAL TO DECIMAL

- Convert 356_{16} to Decimal

$$\begin{aligned} 356_{16} &= 3 \times 16^2 + 5 \times 16^1 + 6 \times 16^0 \\ &= 768 + 80 + 6 \\ &= 854_{10} \end{aligned}$$

- Convert $2AF_{16}$ to Decimal, where A=10, and F=15

$$\begin{aligned} 2AF_{16} &= 2 \times 16^2 + 10 \times 16^1 + 15 \times 16^0 \\ &= 512 + 160 + 15 \\ &= 687_{10} \end{aligned}$$

20

DECIMAL TO HEXADECIMAL

- Convert 423_{16} to Hexadecimal

$$\frac{423}{16} = 26 + \text{remainder of } 7$$

$$\frac{26}{16} = 1 + \text{remainder of } 10$$

$$\frac{1}{16} = 0 + \text{remainder of } 1$$

$$423_{10} = 1A7_{16}$$

21

DECIMAL TO HEXADECIMAL

- Convert 214_{10} to HEX

$$\frac{214}{16} = 13 + \text{remainder of } 6$$

$$\frac{13}{16} = 0 + \text{remainder of } 13$$

$$214_{10} = D6_{16}$$

22

HEXADECIMAL TO BINARY

- Convert $9F2_{16}$ to Binary
- Each HEX digit gives 4 Binary bits
- Therefore, $9F2_{16} = 1001\ \ 1111\ \ 0010_2$

Hex	Binary
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

23

BINARY TO HEXADECIMAL

- Convert 1110100110_2 to HEX
- Group the binary number into 4-bit groups from LSB
- Leading 0 can be added to the left of the MSB

$$1\,1\,1\,0\,1\,0\,0\,1\,1\,0_2 = \underbrace{0\,0\,1\,1}_{3}\ \underbrace{1\,0\,1\,0}_{A}\ \underbrace{0\,1\,1\,0}_{6}$$
$$= 3A6_{16}$$

Hex	Binary
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

24

OCTAL

- Octal digits: 0, 1, 2, 3, 4, 5, 6, 7
- Octal to/from DEC/HEX
  - $173_8 = 123_{10} = 78_{16}$
- Octal to Binary
  - Each Octal digit gives 3 Binary bits
  - $173_8 = 001\ \ 111\ \ 011_2$

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

25

26

BINARY ADDITION AND SUBTRACTION

BINARY ADDITION

- Bitwise addition from right to left
- Example: $0110_2 + 1001_2 = 01111_2$ with Carry bit = 0
- Example: $0110_2 + 1101_2 = 10011_2$ with Carry bit = 1

What will happen if the system can only support 4-bit number?

	0	1	1	0
+	1	0	0	1
Carry	0	0	0	0
Sum	1	1	1	1

	0	1	1	0
+	1	1	0	1
Carry	1	1	0	0
Sum	0	0	1	1

27

BINARY SUBTRACTION

- Bitwise subtraction from right to left
- Example (13 - 1 = 12): 1110 - 0001 = 1101
- Example (6 - 9 = -3) : 0110 - 1001 = 1101

For a 4-bit system, are they correct?

	1	1	1	0
-	0	0	0	1
Borrow	0	0	0	1
Ans	1	1	0	1

	0	1	1	0
-	1	0	0	1
Borrow	1	0	0	1
Ans	1	1	0	1

28

SUBTRACTION WITH COMPLEMENTS

- In digital hardware, the borrow concept in subtraction is less efficient in implementation
- Therefore, the subtraction of two binary unsigned number (e.g. M – N) can be done by addition of M and the 2's complement of N
- If M ≥ N, the sum will produce an end Carry, which can be discarded
- If M < N, the sum does not produce an end Carry. Take the 2's complement of the sum and place a negative sign in front.

29

1'S AND 2'S COMPLEMENTS

- A binary number X = 1001
- 1's complement of X = 0110 (by changing 0 to 1, and 1 to 0)
- 2's complement of X = 0111 (by adding 1 to the 1's complement)

30

EXAMPLE

- For a 4-bit **unsigned** system, calculate 1011 – 0011 (i.e. 11 – 3 = 8)

- 1's complement of 0011 = 1100
- 2's complement of 0011 = 1101

- Adding 1011 with the 2's complement of 0011
- = 1011 + 1101
- = 11000 with Carry bit =1

- Since M>N, the Carry bit can be discarded, and the answer is 1000

31

EXAMPLE

- For a 4-bit **unsigned** system, calculate 0011 – 1011 (i.e. 3 – 11 = -8)

- 1's complement of 1011 = 0100
- 2's complement of 1011 = 0101

- Adding 0011 with the 2's complement of 1011
- = 0011 + 0101
- = 1000 with Carry bit = 0

- Since M<N, the answer is the negative 2's complement of the sum 1000 = -1000

32

SIGNED AND UNSIGNED BINARY NUMBERS

Unsigned Binary	Unsigned Decimal		Signed 1's Complement	Signed Decimal		Signed 2's Complement	Signed Decimal
000	0		111	-0		000	0
001	1		110	-1		111	-1
010	2		101	-2		110	-2
011	3		100	-3		101	-3
100	4		011	+3		100	-4
101	5		010	+2		011	+3
110	6		001	+1		010	+2
111	7		000	+0		001	+1

- Ranges of different n bits binary representations:
- Unsigned number, from 0 to $+(2^n-1)$
- Signed 1's complement, from $-(2^{(n-1)}-1)$ to $+(2^{(n-1)}-1)$, with 2 zeros
- Signed 2's complement, from $-(2^{(n-1)})$ to $+(2^{(n-1)}-1)$

33

EXAMPLE

- For a 3-bit **signed** 2's complement system, calculate 011 – 100 (i.e. 3 – 4 = -1)

- 1's complement of 100 = 011
- 2's complement of 100 = 100

- Adding 011 with the 2's complement of 100
- = 011 + 100
- = 111_2 (-1_{10})

34

OVERFLOW

- Overflow can occur only when two positive or two negative numbers are being added
  - If two 2's complement numbers are added, and they both have the same sign,
  - Then overflow occurs if and only if the result has the opposite sign
- Overflow never occurs when adding operands with different signs

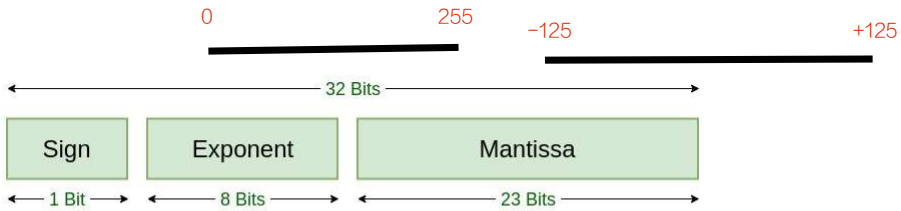
35

36

FLOATING POINT REPRESENTATION

FLOATING POINT TO DECIMAL

- There are 3 elements in a 32-bit floating point representation
  - Sign
  - Exponent
  - Mantissa



Single Precision
IEEE 754 Floating-Point Standard

37

FLOATING POINT REPRESENTATION

- Sign bit = 0 for positive number, sign bit = 1 for negative number
- Exponent part is an exponent with a bias of 127
  - For an exponent = 3, the biased exponent = 127+3 = 130
  - For an exponent = -2, the biased exponent = 127-2 = 125
- Mantissa part is the rest of digits after the decimal point of the normalized binary number
  - For normalized number = 1.101, the mantissa is 10100000 00000000 00000000

because range:−125 to +125
127 out bound

Decimal	*Binary*	*Normalized*	*Sign*	*Biased Exponent*	*Mantissa*
+12	1100	1.1×2^3	0	10000010	10000000 00000000 00000000
−12	1100	1.1×2^3	1	10000010	10000000 00000000 00000000
+100	1100100	1.1001×2^6	0	10000101	10010000 00000000 00000000
−1.75	1.11	1.11×2^0	1	01111111	11000000 00000000 00000000
+0.25	0.01	1.0×2^{-2}	0	01111101	00000000 00000000 00000000
+0.0	0	0	0	00000000	00000000 00000000 00000000

38

39

OTHER USEFUL CODES

OTHER USEFUL CODES

- Binary Coded Decimal (BCD)
- Gray Code
- ASCII Code

40

BINARY CODED DECIMAL

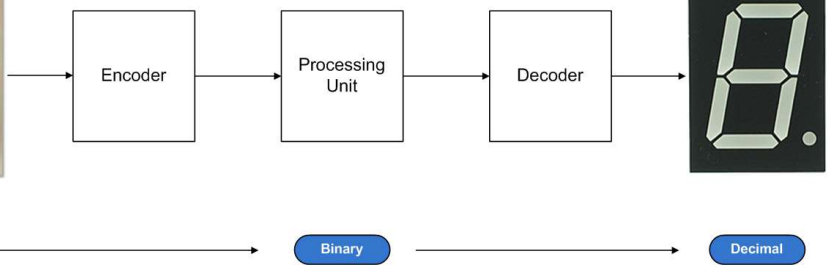
- Binary Coded Decimal (BCD) is a coding system that assigns a four-digit binary code to each decimal digit (i.e. 0 ~ 9)

DECIMAL	BINARY	BINARY CODE DECIMAL 8421
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111
8	1000	1000
9	1001	1001
10	1010	0001 0000
11	1011	0001 0001
12	1100	0001 0010
13	1101	0001 0011
14	1110	0001 0100
15	1111	0001 0101

41

APPLICATION OF BCD

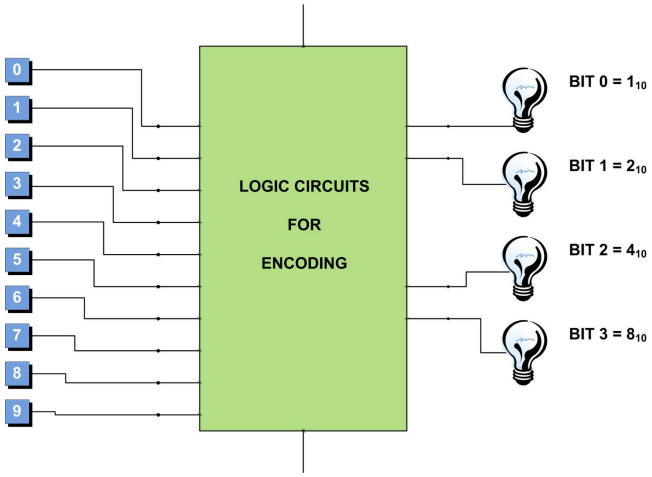
- BCD are commonly used in number display systems
- When you press a number on the keypad, the corresponding number is showed on a 7-segment LED display



42

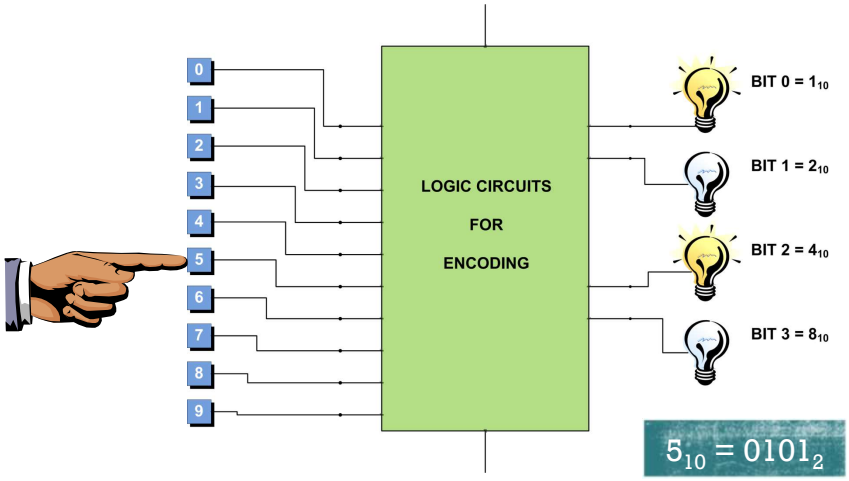
APPLICATION OF BCD

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



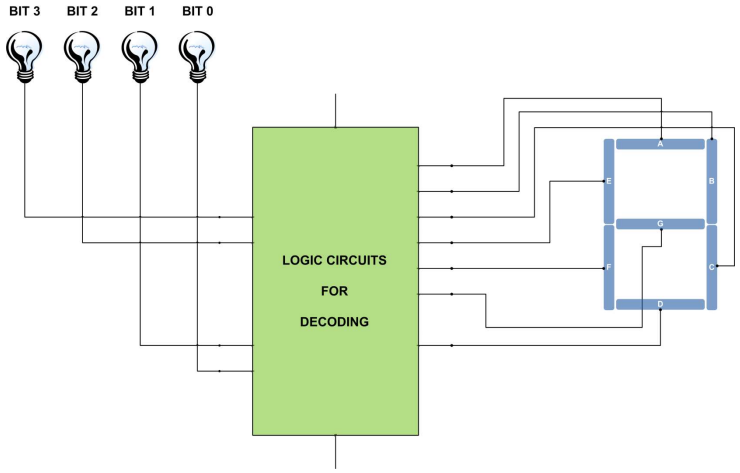
43

APPLICATION OF BCD



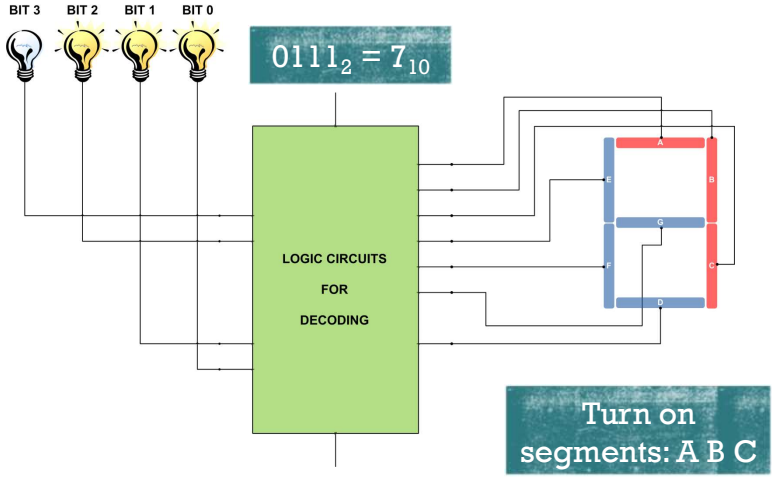
44

APPLICATION OF BCD



45

APPLICATION OF BCD

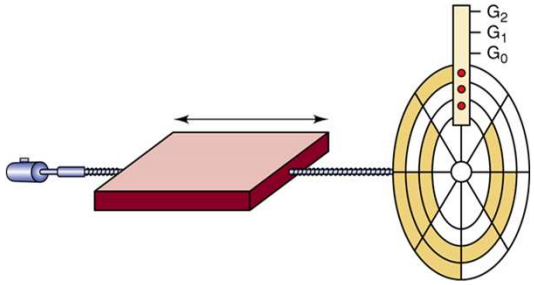


46

GRAY CODE

- Gray code is used in the applications which numbers change rapidly
- Only one bit changes from each value to the next

- 3-bit encoder for 8 different codes

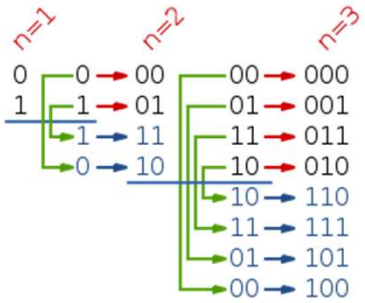


47

GRAY CODE

- Gray code is also called reflected binary number
- A binary numbering system which two successive values differ in only one bit

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



48

ASCII CODE

- ASCII is an abbreviation of American Standard Code for Information Interchange
- A 7-bit code is used to represent all the alphanumeric data used in computer

Dec	Hex	Code	Dec	Hex	Code	Dec	Hex	Code	Dec	Hex	Code
0	00	NUL	32	20	space	64	40	@	96	60	`
1	01	SOH	33	21	!	65	41	A	97	61	a
2	02	STX	34	22	"	66	42	B	98	62	b
3	03	ETX	35	23	#	67	43	C	99	63	c
4	04	EOT	36	24	&	68	44	D	100	64	d
5	05	ENQ	37	25	%	69	45	E	101	65	e
6	06	ACK	38	26	$	70	46	F	102	66	f
7	07	BEL	39	27	'	71	47	G	103	67	g
8	08	BS	40	28	(	72	48	H	104	68	h
9	09	HT	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	140	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

49

UNICODE

- ASCII is not sufficient for global intercommunication
- A character set including all languages is needed
- At most 4 bytes are required for a character
- UTF8 and UTF16 are different encodings of Unicode

Emoticons[1][2]

Official Unicode Consortium code chart (PDF)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+1F60x	😀	😁	😂	😃	😄	😅	😆	😇	😈	😉	😊	😋	😌	😍	😎	😏
U+1F61x	😐	😑	😒	😓	😔	😕	😖	😗	😘	😙	😚	😛	😜	😝	😞	😟
U+1F62x	😠	😡	😢	😣	😤	😥	😦	😧	😨	😩	😪	😫	😬	😭	😮	😯
U+1F63x	😰	😱	😲	😳	😴	😵	😶	😷	😸	😹	😺	😻	😼	😽	😾	😿
U+1F64x	🙀	□	□			🙍	🙎	🙏	🙐	🙑	🙒	🙓	🙔	🙕	🙖	🙗

Notes

1. ^ As of Unicode version 7.0
2. ^ Grey areas indicate non-assigned code points

50

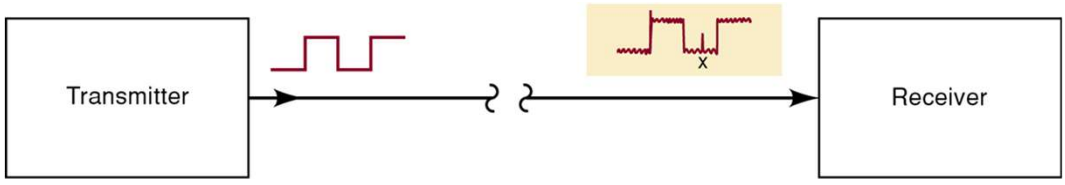
51

ERROR DETECTION AND CORRECTION

parallel and serial transmission :
Parallel connections have multiple wires running parallel to each other (hence the name), and can transmit data on all the wires simultaneously. Serial, on the other hand, uses a single wire to transfer the data bits one at a time.

ERROR DETECTION

- Electrical noise can cause errors during data transmission
- Fluctuations in voltage or current present in all electronic systems
- Error detection and correction is important



52

PARITY CHECK

one error bit detection

- Parity check is a simple error detection method which requires one extra bit to a set of codes
- The extra bit is called the parity bit
- There are two parity options, either even or odd
- For even/odd parity, the number of 1s in a data group must be a even/odd number respectively
- If the receiver find that the number of 1s is not matched the pre-set even/odd parity protocol, an error is detected

53

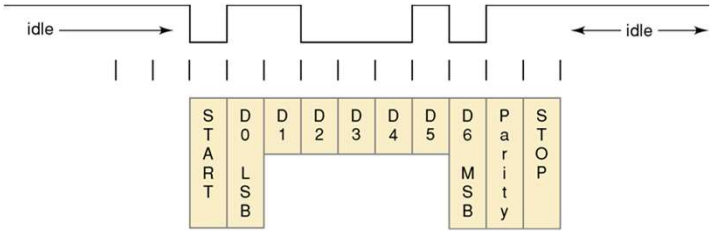
PARITY CHECK

- Assume we have a 8-bit original data 10111001
- For even parity, the transmitting data will become 110111001
- For odd parity, the transmitting data will become 010111001

54

USE OF PARITY IN ASCII

- An ASCII character must be framed, so that the receiver knows where the data begins and ends
- The first bit is a start bit which is logic 0
- Followed by 7-bit ASCII code D0 to D6
- Followed by a parity bit
- Ended with one/two stop bits which is/are logic 1



55

HAMMING CODE

- An encoding scheme that can correct any single bit error
- Contains many parity bits that are placed in between the data bits strategically
- When the position number of the encoded bit stream is a power of 2, that bit will be used as check bits
- Each check bit has a corresponding check equation that covers a portion of the original data

56

ENCODING USING HAMMING CODE

- 8-bit original data: 0100 1011 (D8~D1)
- 4 parity bits: P4, P3, P2, P1
- For even parity, we have the encoded data: 0100 1101 0110

12	11	10	9	8	7	6	5	4	3	2	1
D8	D7	D6	D5	P4	D4	D3	D2	P3	D1	P2	P1
0	1	0	0	?	1	0	1	?	1	?	?
0	1	0	0	?	1	0	1	?	1	?	0
0	1	0	0	?	1	0	1	?	1	1	
0	1	0	0	?	1	0	1	0	1		
0	1	0	0	1	1	0	1		1		
0	1	0	0	1	1	0	1	0	1	1	0

57

ERROR DETECTION AND CORRECTION

- 8-bit original data: 0100 1011 (D8~D1)
- 8-bit original data with error: 0101 1011 (D8~D1)
- 4 parity bits: P4, P3, P2, P1 = 1, 0, 1, 0
- For even parity, we have the received data: 0101 1101 0110

12	11	10	9	8	7	6	5	4	3	2	1
D8	D7	D6	D5	P4	D4	D3	D2	P3	D1	P2	P1
0	1	0	1	1	1	0	1	0	1	1	0
0	1	0	1	1	1	0	1	0	1	1	0 X
0	1	0	1	1	1	0	1	0	1	1 T	
0	1	0	1	1	1	0	1	0 T	1		
0	1	0	1	1 X	1	0	1		1		
0	1	0	1	1	1	0	1	0	1	1	0

D5 is 0, but face error->1 now

green=parity bits

check parity of original code (w/o error) to output the correct one

check yellowed even parity:

4 '1' s-> parity bit(p1)=0

check yellowed again, parity=3, p2=1

check again, 2, p3=0

check again=1, p4=1

By checking the data(X/T) we can locate the wrong data and correct it

combine p4p3p2p1, if error then report 1
Error code= error(p4p3p2p1=1001(2)=9
So position 9 is error

ANY QUESTIONS ?

59