CSTE-2105

DIGITAL LOGIC DESIGN

Lesson Plan (as per week):				
Week	**Course Contents**	**Outcome** (at the end of the lesson, student should be able)	**Teaching Learning Strategy** (activities directed to achieve outcomes)	**Assessment Strategy** (How they are developed)
1.	**Introduction:** Digital and analog systems, The introductory concept of number systems and codes. Digital representation, Digital circuit, and Logic circuit. **Logic Gates, Boolean Algebra and Minimization**: Boolean constants and variables, truth tables. Basic Logic gates.	➢ To differentiate between digital and analog systems. ➢ To learn how to convert a number to different number systems. ➢ To familiar with logic gates, truth table, and logical algebra.	Lecture and discussion on detailed information about the course, including the objectives, course outcomes, examinations. Lecture delivery on the basics of digital and analog systems, number systems and logic gates.	Answering basic questions, quizzes, Homework etc.
2.	**Logic Gates, Boolean Algebra and Minimization**: Universality of NAND and NOR gates, Describing logic circuits algebraically, Evaluating logic circuit outputs, Boolean theorems, DeMorgan's theorems. Implementing logic circuits from boolean expressions, Alternate logic-gate representations.	➢ To learn how to implement logic gates and logical expression using NAND and NOR gates. ➢ To describe logic circuits algebraically. ➢ To implement a logic circuit from Boolean expression.	Lecture and discussion on the universality of NAND and NOR gates and the implementation of logic circuits. Exercise sample problems on logic circuit implementation.	Answering basic questions, quizzes, Homework etc.
3.	**Combinational Logic Circuits Design**: Sum-of-product and product-of-sum forms, Simplifying logic circuits, algebraic	➢ To learn how to express a logical expression in SOP and POS form.	Lecture and discussion on SOP and POS logical expression, logic simplification	Answering basic questions, quizzes, Homework etc.

OUTLINE

- Digital and analog systems
- Number systems and codes

NUMERICAL REPRESENTATIONS OF QUANTITIES

It is important when dealing with various quantities that we be able to represent their values efficiently and accurately.
Basic two ways of representing the numerical value of quantities: analog and digital

NUMERICAL REPRESENTATIONS OF QUANTITIES

Analog: Continuous

Digital: Discrete (step by step)

Example: Thermometer. Temperature with a mercury is analog quantity. Nearest line selection over a continuous range is digital

NUMERICAL REPRESENTATIONS OF QUANTITIES EXAMPLE:

1. Ten position switch (D)
2. Current flowing from an electrical outlet (A)
3. Temperature of a room (A)
4. Sand grains on the beach (D)
5. Automobile fuel gauge (A/D)

DIGITAL AND ANALOG DETAILS

Book reference: Digital Systems by Ronald J. Tocci

DIGITAL AND ANALOG DETAILS (HOME WORK)

Questions:

- Advantages of digital techniques over analog.
- Limitations of digital technique.
- Conversion steps from analog-digital-analog.

NUMBER SYSTEM

Decimal number system is the most common number system. Other popular number systems include binary number system, octal number system, hexadecimal number system, etc.

DECIMAL NUMBER SYSTEM

Decimal number system is a **base 10** number system having 10 digits from 0 to 9. This means that any numerical quantity can be represented using these 10 digits. Decimal number system is also a **positional value system**. This means that the value of digits will depend on its position.

Say we have three numbers – 734, 971 and 207. The value of 7 in all three numbers is different−

- In 734, value of 7 is 7 hundreds or 700 or 7×100 or 7×10^2
- In 971, value of 7 is 7 tens or 70 or 7×10 or 7×10^1
- In 207, value 0f 7 is 7 units or 7 or 7×1 or 7×10^0

DECIMAL NUMBER SYSTEM

The weightage of each position can be represented as follows −

10^5	10^4	10^3	10^2	10^1	10^0

DECIMAL NUMBER SYSTEM

In digital systems, instructions are given through electric signals; variation is done by varying the voltage of the signal. Having 10 different voltages to implement decimal number system in digital equipment is difficult. So, many number systems that are easier to implement digitally have been developed.

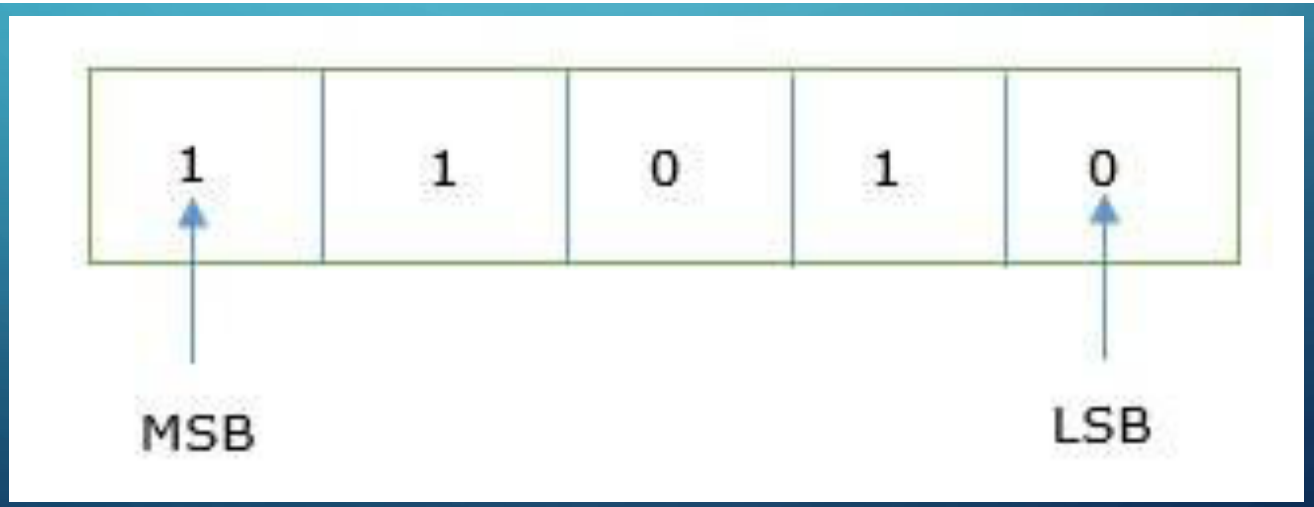
BINARY NUMBER SYSTEM

The easiest way to vary instructions through electric signals is two-state system – on and off. On is represented as 1 and off as 0, though 0 is not actually no signal but signal at a lower voltage. The number system having just these two digits : 0 and 1 is called **binary number system**.

Each binary digit is also called a **bit**. Binary number system is also positional value system, where each digit has a value expressed in powers of 2.

BINARY NUMBER SYSTEM

In any binary number, the rightmost digit is called **least significant bit (LSB)** and leftmost digit is called **most significant bit (MSB).**



BINARY NUMBER SYSTEM

...and decimal equivalent of this number is sum of product of each digit with its positional value.

$11010_2 = 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$

$= 16 + 8 + 0 + 2 + 0$

$= 26_{10}$

Computer memory is measured in terms of how many bits it can store. Here is a chart for memory capacity conversion.

- 1 byte (B) = 8 bits
- 1 Kilobytes (KB) = 1024 bytes
- 1 Megabyte (MB) = 1024 KB
- 1 Gigabyte (GB) = 1024 MB
- 1 Terabyte (TB) = 1024 GB
- 1 Exabyte (EB) = 1024 TB
- 1 Zettabyte = 1024 EB
- 1 Yottabyte (YB) = 1024 ZB

OCTAL NUMBER SYSTEM

Octal number system has eight digits – 0, 1, 2, 3, 4, 5, 6 and 7. Octal number system is also a positional value system with where each digit has its value expressed in powers of 8, as shown here −Decimal equivalent of any octal number is sum of product of each digit with its positional value.

$$726_8 = 7\times8^2 + 2\times8^1 + 6\times8^0$$
$$= 448 + 16 + 6$$
$$= 470_{10}$$

HEXADECIMAL NUMBER SYSTEM

Hexadecimal number system has 16 symbols – 0 to 9 and A to F where A is equal to 10, B is equal to 11 and so on till F. Hexadecimal number system is also a positional value system with where each digit has its value expressed in powers of 16

16^5	16^4	16^3	16^2	16^1	16^0

$27FA_{16} = 2\times16^3 + 7\times16^2 + 15\times16^1 + 10\times16^0$
$= 8192 + 1792 + 240 + 10$
$= 10234_{10}$

RELATIONSHIP AMONG NUMBER SYSTEMS

The following table depicts the relationship between decimal, binary, octal and hexadecimal number systems.

HEXADECIMAL	DECIMAL	OCTAL	BINARY
0	0	0	0000
1	1	1	0001
2	2	2	0010
3	3	3	0011
4	4	4	0100
5	5	5	0101
6	6	6	0110
7	7	7	0111
8	8	10	1000
9	9	11	1001
A	10	12	1010
B	11	13	1011
C	12	14	1100
D	13	15	1101
E	14	16	1110
F	15	17	1111

NUMBER SYSTEM AND CODES

DECIMAL NUMBER SYSTEM

The weightage of each position for integer number:

10^5	10^4	10^3	10^2	10^1	10^0

For fractional number:

The decimal point is used to separate the integer and fractional parts of the number.

10^3	10^2	10^1	10^0	.	10^{-1}	10^{-2}	10^{-3}

EXAMPLE (SUM OF PRODUCT METHOD)

$$27.35_{10}$$
$$= 2\times10^{1} + 7\times10^{0} + 3\times0.1 + 5\times0.01$$

EXAMPLE
BINARY TO DECIMAL CONVERSION

1011.101_2

$= 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3}$

$= 8+0+2+1+0.5+0+0.125$

$= 11.625_{10}$

EXAMPLE
OCTAL TO DECIMAL CONVERSION

24.68_8

$= 2\times8^1 + 4\times8^0 + 6\times8^{-1} + 8\times8^{-2}$

$= 16+4+0.75+0.125$

$= 20.875_{10}$

EXAMPLE
HEX TO DECIMAL CONVERSION

$1F.01B_{16}$

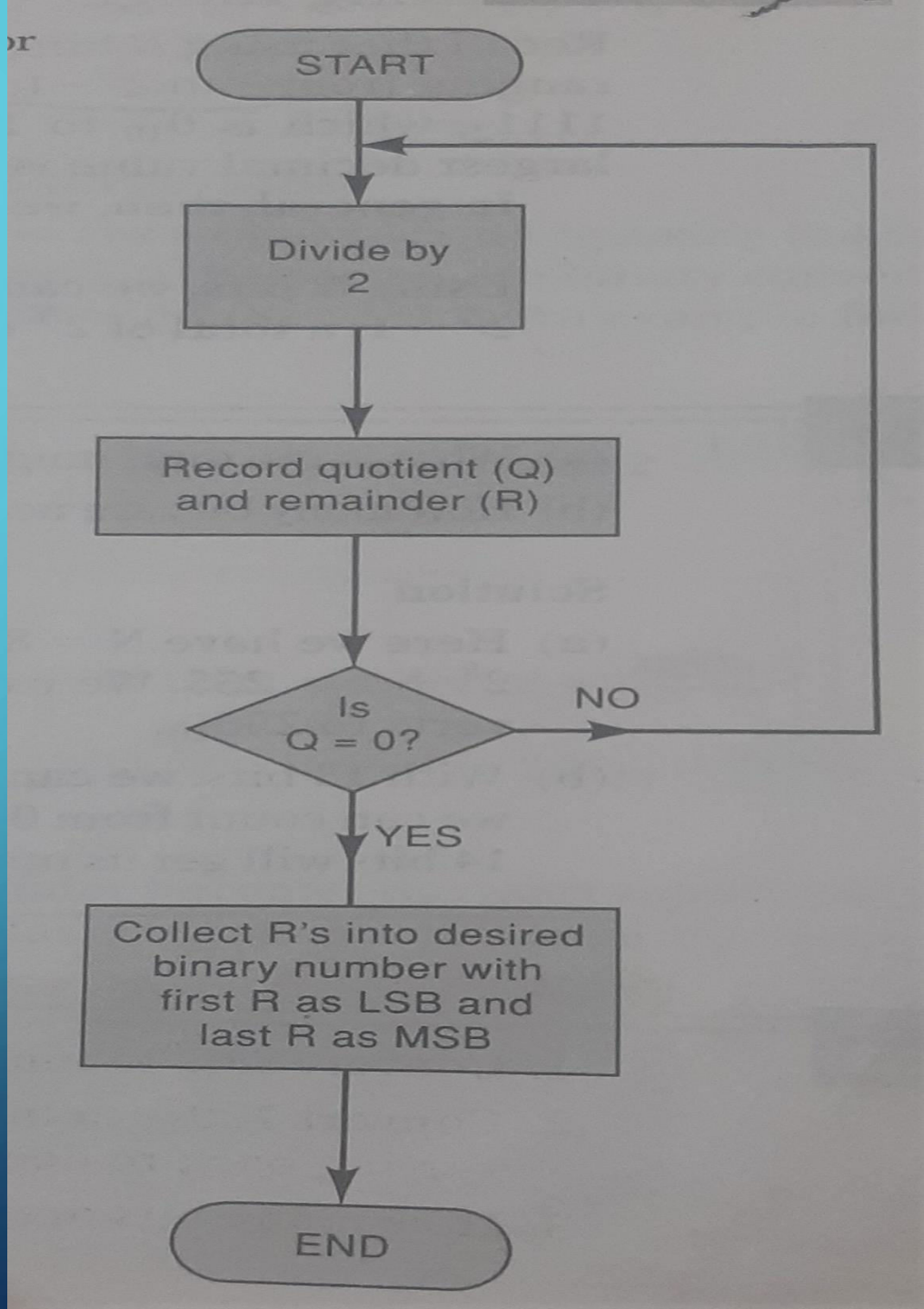
$= 1\times16^{1} + 15\times16^{0} + 0\times16^{-1} + 1\times16^{-2} + 11\times16^{-3}$

$= 31.0065918_{10}$

REVERSE CALCULATION

REPEATED DIVISION METHOD

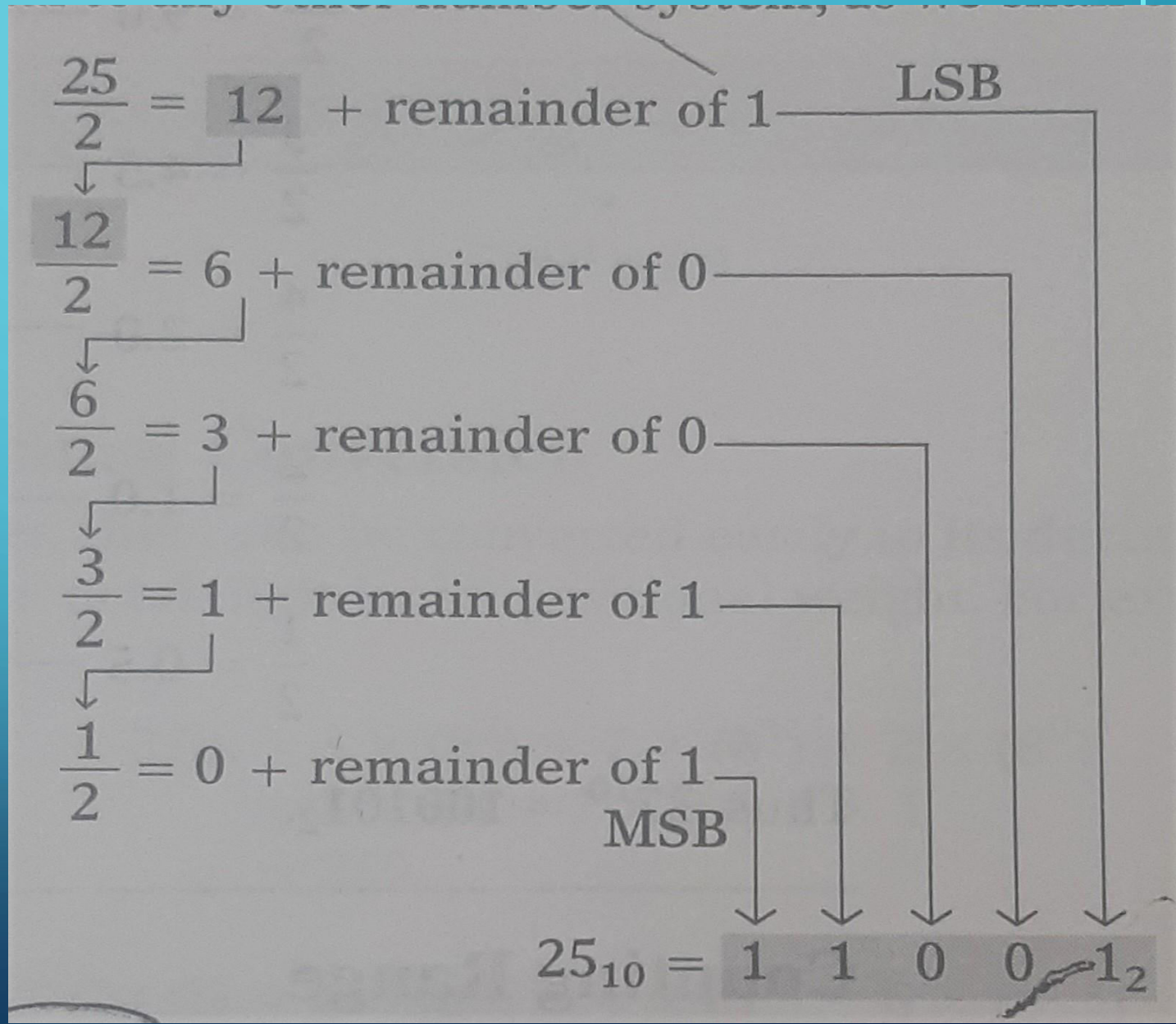
DECIMAL TO BINARY CONVERSION REPEATED DIVISION METHOD



EXAMPLE
DECIMAL TO BINARY CONVERSION
REPEATED DIVISION METHOD

Convert 25_{10} to binary number.

$25_{10} = 11001_2$



EXAMPLE
DECIMAL TO OCTAL CONVERSION
REPEATED DIVISION METHOD

Convert 266_{10} to octal number.

$266_{10} = 412_8$

EXAMPLE
DECIMAL TO HEX CONVERSION
REPEATED DIVISION METHOD

Convert 423_{10} to octal number.

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

MIXED CONVERSION

EXAMPLE
OCTAL TO BINARY CONVERSION

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

Convert 472_8 to binary number.

$472_8 = 100111010_2$

And

Convert 100111010_2 to octal number.

$100111010_2 = 472_8$

EXAMPLE
HEX TO BINARY CONVERSION

Convert $9F2_{16}$ to binary number.

$9F2_{16} = 100111110010_2$

And

Convert 100111110010_2 to hex number.

$100111110010_2 = 9F2_{16}$

CODE

CODE

When number, letters or words are represented by a special group of symbols , we say they are being encoded and the group of symbols is called a code. Example: Morse code

BCD CODE

Each digit of decimal number is represented by its binary equivalent- called binary coded decimal code.

874_{10}
$= 1000\ 0111\ 0100_{BCD}$
BCD code doesn't use A-F numbers.

BCD *VS* BINARY

137_{10}
$= 10001001_{2}$
$= 0001\ 0011\ 0111_{BCD}$

GRAY CODE

Only one bit ever changes between two successive numbers in the sequence.

BINARY-GRAY CONVERSION

B2	B1	B0	G2	G1	G0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

GRAY-BINARY CONVERSION

G2	G1	G0	B2	B1	B0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	1	0	1	0
0	1	0	0	1	1
1	1	0	1	0	0
1	1	1	1	0	1
1	0	1	1	1	0
1	0	0	1	1	1

GRAY CODE

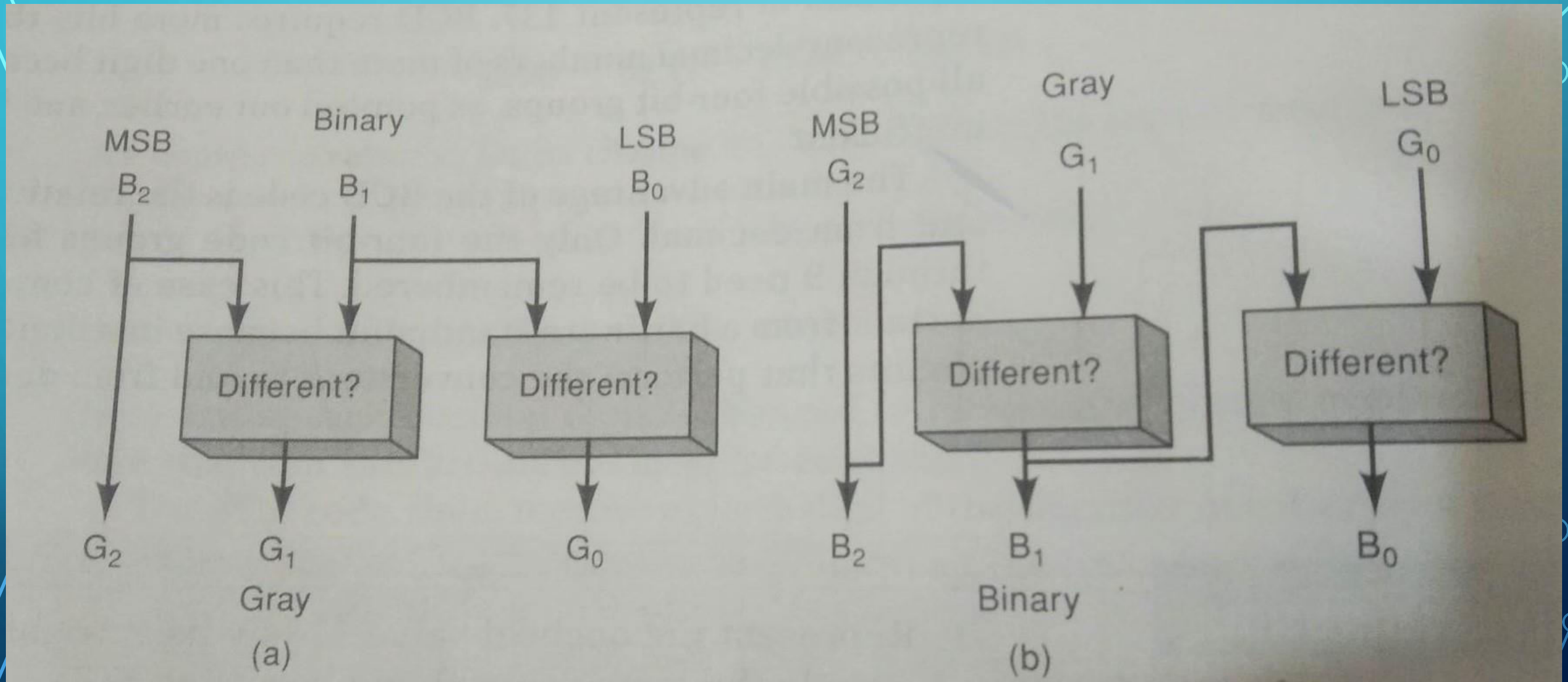


FIGURE 2-2 Converting (a) binary to Gray and (b) Gray to binary.

AT A GLANCE

Decimal	Binary	Octal	Hexadecimal	BCD	GRAY
0	0	0	0	0000	0000
1	1	1	1	0001	0001
2	10	2	2	0010	0011
3	11	3	3	0011	0010
4	100	4	4	0100	0110
5	101	5	5	0101	0111
6	110	6	6	0110	0101
7	111	7	7	0111	0100
8	1000	10	8	1000	1100
9	1001	11	9	1001	1101
10	1010	12	A	0001 0000	1111
11	1011	13	B	0001 0001	1110
12	1100	14	C	0001 0010	1010
13	1101	15	D	0001 0011	1011
14	1110	16	E	0001 0100	1001
15	1111	17	F	0001 0101	1000

Lesson Plan (as per week):				
Week	**Course Contents**	**Outcome** (at the end of the lesson, student should be able)	**Teaching Learning Strategy** (activities directed to achieve outcomes)	**Assessment Strategy** (How they are developed)
1.	**Introduction:** Digital and analog systems, The introductory concept of number systems and codes. Digital representation, Digital circuit, and Logic circuit. **Logic Gates, Boolean Algebra and Minimization**: Boolean constants and variables, truth tables. Basic Logic gates.	➢ To differentiate between digital and analog systems. ➢ To learn how to convert a number to different number systems. ➢ To familiar with logic gates, truth table, and logical algebra.	Lecture and discussion on detailed information about the course, including the objectives, course outcomes, examinations. Lecture delivery on the basics of digital and analog systems, number systems and logic gates.	Answering basic questions, quizzes, Homework etc.
2.	**Logic Gates, Boolean Algebra and Minimization**: Universality of NAND and NOR gates, Describing logic circuits algebraically, Evaluating logic circuit outputs, Boolean theorems, DeMorgan's theorems. Implementing logic circuits from boolean expressions, Alternate logic-gate representations.	➢ To learn how to implement logic gates and logical expression using NAND and NOR gates. ➢ To describe logic circuits algebraically. ➢ To implement a logic circuit from Boolean expression.	Lecture and discussion on the universality of NAND and NOR gates and the implementation of logic circuits. Exercise sample problems on logic circuit implementation.	Answering basic questions, quizzes, Homework etc.
3.	**Combinational Logic Circuits Design**: Sum-of-product and product-of-sum forms, Simplifying logic circuits, algebraic	➢ To learn how to express a logical expression in SOP and POS form.	Lecture and discussion on SOP and POS logical expression, logic simplification	Answering basic questions, quizzes, Homework etc.

CODES

ASCII

Besides numerical data, computer must be able to handle alphabets, punctuation marks, mathematical operators, special symbols etc. that form the complete character set of English language. The complete set of characters or symbols are called alphanumeric codes.

- 26 upper case letters
- 26 lower case letters
- 10 numeric digits
- 7 punctuation marks
- 20 to 40 special characters

CODES
ASCII

A computer understands only numeric values, whatever the number system used. So all characters must have a numeric equivalent called the alphanumeric code. The most widely used alphanumeric code is American Standard Code for Information Interchange (ASCII). ASCII is a 7-bit code that has 128 (2^7) possible codes.

ASCII

TABLE 2-5 Standard ASCII codes.

Character	HEX	Decimal	Character	HEX	Decimal	Character	HEX	Decimal	Character	HEX	Decimal
NUL (null)	0	0	Space	20	32	@	40	64	`	60	96
Start Heading	1	1	!	21	33	A	41	65	a	61	97
Start Text	2	2	"	22	34	B	42	66	b	62	98
End Text	3	3	#	23	35	C	43	67	c	63	99
End Transmit.	4	4	$	24	36	D	44	68	d	64	100
Enquiry	5	5	%	25	37	E	45	69	e	65	101
Acknowlege	6	6	&	26	38	F	46	70	f	66	102
Bell	7	7	'	27	39	G	47	71	g	67	103
Backspace	8	8	(	28	40	H	48	72	h	68	104
Horiz. Tab	9	9	)	29	41	I	49	73	i	69	105
Line Feed	A	10	*	2A	42	J	4A	74	j	6A	106
Vert. Tab	B	11	+	2B	43	K	4B	75	k	6B	107
Form Feed	C	12	,	2C	44	L	4C	76	l	6C	108
Carriage Return	D	13	-	2D	45	M	4D	77	m	6D	109
Shift Out	E	14	.	2E	46	N	4E	78	n	6E	110
Shift In	F	15	/	2F	47	O	4F	79	o	6F	111
Data Link Esc	10	16	0	30	48	P	50	80	p	70	112
Direct Control 1	11	17	1	31	49	Q	51	81	q	71	113
Direct Control 2	12	18	2	32	50	R	52	82	r	72	114
Direct Control 3	13	19	3	33	51	S	53	83	s	73	115
Direct Control 4	14	20	4	34	52	T	54	84	t	74	116
Negative ACK	15	21	5	35	53	U	55	85	u	75	117
Synch Idle	16	22	6	36	54	V	56	86	v	76	118
End Trans Block	17	23	7	37	55	W	57	87	w	77	119
Cancel	18	24	8	38	56	X	58	88	x	78	120
End of Medium	19	25	9	39	57	Y	59	89	y	79	121
Substitue	1A	26	:	3A	58	Z	5A	90	z	7A	122
Escape	1B	27	;	3B	59	[	5B	91	{	7B	123
Form separator	1C	28	<	3C	60	\	5C	92	|	7C	124
Group separator	1D	29	=	3D	61	]	5D	93	}	7D	125
Record Separator	1E	30	>	3E	62	^	5E	94	~	7E	126
Unit Separator	1F	31	?	3F	63	_	5F	95	Delete	7F	127

ASCII

ASCII Code - Character to Binary

0	0011 0000	I	0100 1001	b	0110 0010	v	0111 0110
1	0011 0001	J	0100 1010	c	0110 0011	w	0111 0111
2	0011 0010	K	0100 1011	d	0110 0100	x	0111 1000
3	0011 0011	L	0100 1100	e	0110 0101	y	0111 1001
4	0011 0100	M	0100 1101	f	0110 0110	z	0111 1010
5	0011 0101	N	0100 1110	g	0110 0110		
6	0011 0110	O	0100 1111	h	0110 1000	:	0011 1010
7	0011 0110	P	0101 0000	i	0110 1001	;	0011 1011
8	0011 1000	Q	0101 0001	j	0110 1010	?	0011 1111
9	0011 1001	R	0101 0010	k	0110 1011	·	0010 1110
		S	0101 0011	l	0110 1100	'	0010 1111
A	0100 0001	T	0101 0100	m	0110 1101	!	0010 0001
B	0100 0010	U	0101 0101	n	0110 1110	'	0010 1100
C	0100 0011	V	0101 0110	o	0110 1111	"	0010 0010
D	0100 0100	W	0101 0111	p	0111 0000	(	0010 1000
E	0100 0101	X	0101 1000	q	0111 0001	)	0010 1001
F	0100 0110	Y	0101 1001	r	0111 0010	space	0010 0000
G	0100 0111	Z	0101 1010	s	0111 0011		
H	0100 1000			t	0111 0100		
		a	0110 0001	u	0111 0101		

OTHER CODES

We know that the bits 0 and 1 corresponding to two different range of analog voltages. So, during transmission of binary data from one system to the other, the noise may also be added. Due to this, there may be errors in the received data at other system.

That means a bit 0 may change to 1 or a bit 1 may change to 0. We can't avoid the interference of noise. But, we can get back the original data first by detecting whether any errors present and then correcting those errors. For this purpose, we can use the following codes

- Error detection codes
- Error correction codes

OTHER CODES

Error detection codes − are used to detect the errors present in the received data bit stream. These codes contain some bits, which are included appended to the original bit stream. These codes detect the error, if it is occurred during transmission of the original data bit stream.
Example − Parity code, Hamming code.

Error correction codes − are used to correct the errors present in the received data bit stream so that, we will get the original data. Error correction codes also use the similar strategy of error detection codes.
Example − Hamming code.

Therefore, to detect and correct the errors, additional bits are appended to the data bits at the time of transmission.

PARITY CODE

It is easy to include append one parity bit either to the left of MSB or to the right of LSB of original bit stream. There are two types of parity codes, namely even parity code and odd parity code based on the type of parity being chosen.

Even Parity Code:

The value of even parity bit should be zero, if even number of ones present in the binary code. Otherwise, it should be one. So that, even number of ones present in **even parity code**. Even parity code contains the data bits and even parity bit.

EVEN PARITY CODE

The following table shows the **even parity codes** corresponding to each 3-bit binary code. Here, the even parity bit is included to the right of LSB of binary code

Binary Code	Even Parity bit	Even Parity Code
000	0	0000
001	1	0011
010	1	0101
011	0	0110
100	1	1001
101	0	1010
110	0	1100
111	1	1111

EVEN PARITY CODE

Here, the number of bits present in the even parity codes is 4.

- If the other system receives one of these even parity codes, then there is no error in the received data.
- If the other system receives other than even parity codes, then there will be an error in the received data.

Limitation:

- In this case, we can't predict the original binary code because we don't know the bit positions of error.

Therefore, even parity bit is useful only for detection of error in the received parity code. But, it is not sufficient to correct the error.

ODD PARITY CODE

The value of odd parity bit should be zero, if odd number of ones present in the binary code. Otherwise, it should be one. So that, odd number of ones present in **odd parity code**. Odd parity code contains the data bits and odd parity bit.

ODD PARITY

- The following table shows the **odd parity codes** corresponding to each 3-bit binary code. Here, the odd parity bit is included to the right of LSB of binary code. If the other system receives one of these odd parity codes, then there is no error in the received data.

Binary Code	Odd Parity bit	Odd Parity Code
000	1	0001
001	0	0010
010	0	0100
011	1	0111
100	0	1000
101	1	1011
110	1	1101
111	0	1110

HAMMING CODE

Hamming code is useful for both detection and correction of error present in the received data. This code uses multiple parity bits.