Digital System Designs
(EOPC2006)

Module 1

- Number System
- Boolean Algebra
- Logic Gates

Number System

- Introduction to various number systems and their conversions
- Arithmetic Operations using 1's and 2's Complements
- Signed Binary and Floating-Point Number Representation
- Introduction to Binary codes and their applications

Introduction to various number systems and their conversions

- Decimal numbers
- Binary numbers
- Octal numbers
- Hexadecimal number
- Number with radix/ base *r*
- Signed Binary Numbers

Introduction to various number systems and their conversions

Number System	Base / Radix	Distinct Symbols	Examples
Binary	2	0, 1	10, 111, 1001
Octal	8	0, 1, 2, …, 6, 7	62, 217
Decimal	10	0, 1, 2, …., 8, 9	19, 340
Hexadecimal	16	0, 1, 2…8, 9, A, B, C, D, E, F	34, 3B, A89

Decimal Numbers Systems

Ten Digits in Decimal number system: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Weight of the places in Decimal number system are 10^n where, n is an integer. This n is negative after decimal point.

$57235.1597\,|_{10} = 5\times10^4 + 7\times10^3 + 2\times10^2 + 3\times10^1 + 5\times10^0 + 1\times10^{-1} + 5\times10^{-2} + 9\times10^{-3} + 7\times10^{-4}$

Decimal point

Return

Binary Numbers Systems and Its Conversion

Two Digits usually specified as two bits in Binary number system: 0 and 1

Weight of the places in this Binary number system are 2^n where, n is an integer. This weight is essential to convert a binary number to a decimal number. This n is negative after binary point.

$$110110.1001|_2 = 1\times2^5 + 1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 0\times2^{-3} + 1\times2^{-4}$$

$$= 32 + 16 + 0 + 4 + 2 + 0 + 0.5 + 0 + 0 + 0.0625 = 54.5625\,|_{10}$$

MSB LSB Binary point

Binary-to-Decimal Conversion

$$
\begin{aligned}
110110111.10101|_2 &= 1\times2^8 + 1\times2^7 + 0\times2^6 + 1\times2^5 + 1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0 \\
&\quad + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} + 0\times2^{-4} + 1\times2^{-5} \\
&= 256 + 128 + 0 + 32 + 16 + 0 + 4 + 2 + 1 + 0.5 + 0 + 0.125 + 0 + 0.03125 \\
&= 439.65625|_{10}
\end{aligned}
$$

Decimal-to-Binary Conversion

		Remainder
2	125	
2	62	1
2	31	0
2	15	1
2	7	1
2	3	1
2	1	1
2	0	1

$125|_{10} = 1111101|_2$

$0.125|_{10} = 0.0010|_2$

	Product	Carry
0.125×2	$= 0.25$	0
0.25×2	$= 0.5$	0
0.5×2	$= 1.0$	1
0×2	$= 0$	0

$125.125|_{10} = 1111101.001|_2$

Octal Number Systems and Its Conversion

Eight Digits in Octal number system: 0, 1, 2, 3, 4, 5, 6, 7

Weight of the places in this Octal number system are 8^n where, n is an integer. This weight is essential to convert an octal number to a decimal number. This n is negative after octal point.

$$7235.157\,|_8 = 7\times8^3 + 2\times8^2 + 3\times8^1 + 5\times8^0 + 1\times8^{-1} + 5\times8^{-2} + 7\times8^{-3}$$
$$= 7\times512 + 2\times64 + 24 + 5 + 0.125 + 5\times0.015625 + 7\times0.001953125$$
$$= 3584 + 128 + 29.125 + 0.078125 + 0.013671875$$
$$= 3741.216796875\,|_{10}$$

Octal point

Three binary bits are grouped

$$7235.157\,|_8 = 111010011101.001101111\,|_2$$

Hexadecimal Number Systems and Its Conversion

Sixteen Digits in Hexadecimal number system: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F

Weight of the places in this Hexadecimal number system are 16^n where, n is an integer. This weight is essential to convert an hexadecimal number to a decimal number. This n is negative after hexadecimal point.

$$
\begin{aligned}
12A5.1D\,|_{16} &= 1\times16^3 + 2\times16^2 + 10\times16^1 + 5\times16^0 + 1\times16^{-1} + 13\times16^{-2} + 7\times16^{-3} \\
&= 1\times4096 + 2\times256 + 160 + 5 + 0.0625 + 13\times0.00390625 \\
&= 4096 + 512 + 165.0625 + 0.05078125 \\
&= 4773.11328125\,|_{10}
\end{aligned}
$$

Hexadecimal point

Four binary bits are grouped

$$12A5.1D\,|_{16} = 0001001010100101.00011101|_2$$

Number with radix/ base r

Weight of the places in such number system are r^n where, n is an integer. This n is negative after decimal point.

$235.159\,|_{10} = 2\times r^{2} + 3\times r^{1} + 5\times r^{0} + 1\times r^{-1} + 5\times r^{-2} + 9\times r^{-3}$

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	—	—

Signed and unsigned Binary numbers in general

Decimal number	Equivalent Unsigned Binary number	Equivalent Signed Binary number
+ 10	1010	0,1010 or 01010
–10	1010 (Need to remember that this is to be deducted)	1,1010 or 11010
+ 25	00011001	0,00011001 or 000011001
–25	00011001 (Need to remember that this is to be deducted)	1,00011001 or 100011001

Signed Binary Numbers

- Signed Binary Numbers can be represented using three forms
    - Sign magnitude form
    - 1’s complement form
    - 2’s complement form
- In all the above three forms the positive numbers are same
- Out of N bits MSB is the sign bit and rest N-1 is used for magnitude
- Positive number has MSB 0 and negative number has MSB 1

Sign magnitude form

0	0	0	0	+ 0
0	0	0	1	+ 1
0	0	1	0	+ 2
0	0	1	1	+ 3
0	1	0	0	+ 4
0	1	0	1	+ 5
0	1	1	0	+ 6
0	1	1	1	+ 7
1	0	0	0	– 0
1	0	0	1	– 1
1	0	1	0	– 2
1	0	1	1	– 3
1	1	0	0	– 4
1	1	0	1	– 5
1	1	1	0	– 6
1	1	1	1	– 7

For an n bit binary number the range of numbers are $-(2^{n-1} - 1)$ to $+(2^{n-1} - 1)$

1’s complement form

0	0	0	0	+ 0
0	0	0	1	+ 1
0	0	1	0	+ 2
0	0	1	1	+ 3
0	1	0	0	+ 4
0	1	0	1	+ 5
0	1	1	0	+ 6
0	1	1	1	+ 7
1	0	0	0	– 7
1	0	0	1	– 6
1	0	1	0	– 5
1	0	1	1	– 4
1	1	0	0	– 3
1	1	0	1	– 2
1	1	1	0	– 1
1	1	1	1	– 0

For an n bit binary number the range of numbers are $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$

2's complement form

0	0	0	0	+ 0
0	0	0	1	+ 1
0	0	1	0	+ 2
0	0	1	1	+ 3
0	1	0	0	+ 4
0	1	0	1	+ 5
0	1	1	0	+ 6
0	1	1	1	+ 7
1	0	0	0	– 8
1	0	0	1	– 7
1	0	1	0	– 6
1	0	1	1	– 5
1	1	0	0	– 4
1	1	0	1	– 3
1	1	1	0	– 2
1	1	1	1	– 1

For an n bit binary number the range of numbers are -2^{n-1} to $+(2^{n-1}-1)$

Floating-Point Number Representation

What is floating point number?

A floating-point number is a way of representing real numbers (including fractions and very large or very small numbers) in computers. Unlike integers, which are whole numbers, floating point numbers can represent decimal values and scientific notation efficiently.

How is it represented?

A floating-point number is typically represented in scientific notation, which looks like:

$\text{Number} = \text{Sign} \times \text{Mantissa} \times 2^{\text{Exponent}}$

Where:

Sign (S): Determines whether the number is positive (0) or negative (1). Mantissa (M) or Significand: The significant digits of the number.

Exponent (E): Determines the scale of the number by raising 2 to this power.

Floating-Point Number Representation

Examples of floating-point numbers in base 10

- 5.341×10^{3}, 0.05341×10^{5}, -2.013×10^{-1}, -201.3×10^{-3}

Examples of floating-point numbers in base 2

- 1.00101×2^{23}, 0.0100101×2^{25}, -1.101101×2^{-3}, -1101.101×2^{-6}
- Exponents are kept in decimal for clarity
- The binary number $(1101.101)_2 = 2^3 + 2^2 + 2^0 + 2^{-1} + 2^{-3} = 13.625$

Floating-point numbers should be normalized

- Exactly **one non-zero digit** should appear before the point
    - In a decimal number, this digit can be from **1 to 9**
    - In a binary number, this digit should be **1**
- **Normalized FP Numbers:** 5.341×10^{3} and -1.101101×2^{-3}
- **NOT Normalized:** 0.05341×10^{5} and -1101.101×2^{-6}

Binary Addition

$$\begin{array}{rcr} 101010111|_2 & \rightarrow & 343|_{10} \\ +111110100|_2 & \rightarrow & +500|_{10} \\ \hline 1101001011|_2 & \rightarrow & 843|_{10} \end{array}$$

$$\begin{array}{rcr} 101010.111|_2 & \rightarrow & 42.875|_{10} \\ +111110.100|_2 & \rightarrow & +\ 62.500|_{10} \\ \hline 1101001.011|_2 & \rightarrow & 105.375|_{10} \end{array}$$

$$\begin{array}{rcccccccccccccc} & & & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & |_2 \rightarrow & 3\ 4\ 3 & |_{10} \\ & + & & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & |_2 \rightarrow & 5\ 0\ 0 & |_{10} \\ & + & & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & |_2 \rightarrow & 3\ 5\ 6 & |_{10} \\ & + & & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & |_2 \rightarrow & 4\ 5\ 3 & |_{10} \\ & + & & & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & |_2 \rightarrow & 1\ 7\ 0 & |_{10} \\ \hline 1 & 1 & & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & |_2 \rightarrow & 1\ 8\ 2\ 2 & |_{10} \end{array}$$

Binary Subtraction using usual and 2's complement

$$\begin{array}{rcr} 111110100|_2 & \rightarrow & 500|_{10} \\ -101010111|_2 & \rightarrow & -343|_{10} \\ \hline 010011101|_2 & \rightarrow & 157|_{10} \end{array}$$

$$\begin{array}{rcr} 111110.100|_2 & \rightarrow & 62.500|_{10} \\ -101010.111|_2 & \rightarrow & -\ 42.875|_{10} \\ \hline 010011.101|_2 & \rightarrow & 19.625|_{10} \end{array}$$

A Subtraction can be converted to an Addition process using 2's complement method

$$\begin{array}{rcrl} 111110100|_2 & \rightarrow & 111110100|_2 & \\ -101010111|_2 & \rightarrow & +010101001|_2 & \textbf{(2's complement of } -101010111\textbf{)} \\ \hline & & \cancel{1}010011101|_2 & \rightarrow \textbf{(Neglect carry, the sum is positive)} \end{array}$$

After completion of the addition process using 2's complement method, if the final sum has a carry at MSB, then the answer is positive and neglect the carry. If the final sum has no carry at MSB, then the answer is negative and in 2's complement form. So taking 2's complement of this answer will give the desired number with negative sign.

Binary Multiplication

$$
\begin{array}{ccccccccccclcr}
 & & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & |_2 & \rightarrow & 343\,|_{10} \\
 & & & & & & & \times & 1 & 0 & 1 & |_2 & \rightarrow & 5\,|_{10} \\
\hline
 & & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & |_2 & & 1715\,|_{10} \\
+ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & |_2 & & \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & & & |_2 & & \\
\hline
1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & |_2 & \rightarrow & 1715\,|_{10}
\end{array}
$$

Multiplication is a successive addition. Multiplying binary 10101 with binary 101 is same as adding 10101 five times. Else shifting technique using shift register can be used. As per this techniques first of all 10101 is placed, below that five zeros are placed after shifting it one place left, subsequently 10101 is placed by shifting it two places left. Then all bits are added column wise.

Binary Division

```
1 0 1 | 1 1 1 1 1 0 1 | 1 1 0 0 1
        1 0 1
        ---------
          1 0 1
          1 0 1
          ---------
              0 1
              0 0
              ---------
                1 0
                0 0
                ---------
                1 0 1
                1 0 1
                ---------
                    0
```

Binary division can be considered as successive subtraction

Binary Codes

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
	1010	0101	0000	0001
Unused	1011	0110	0001	0010
bit	1100	0111	0010	0011
combi-	1101	1000	1101	1100
nations	1110	1001	1110	1101
	1111	1010	1111	1110

Gray Codes

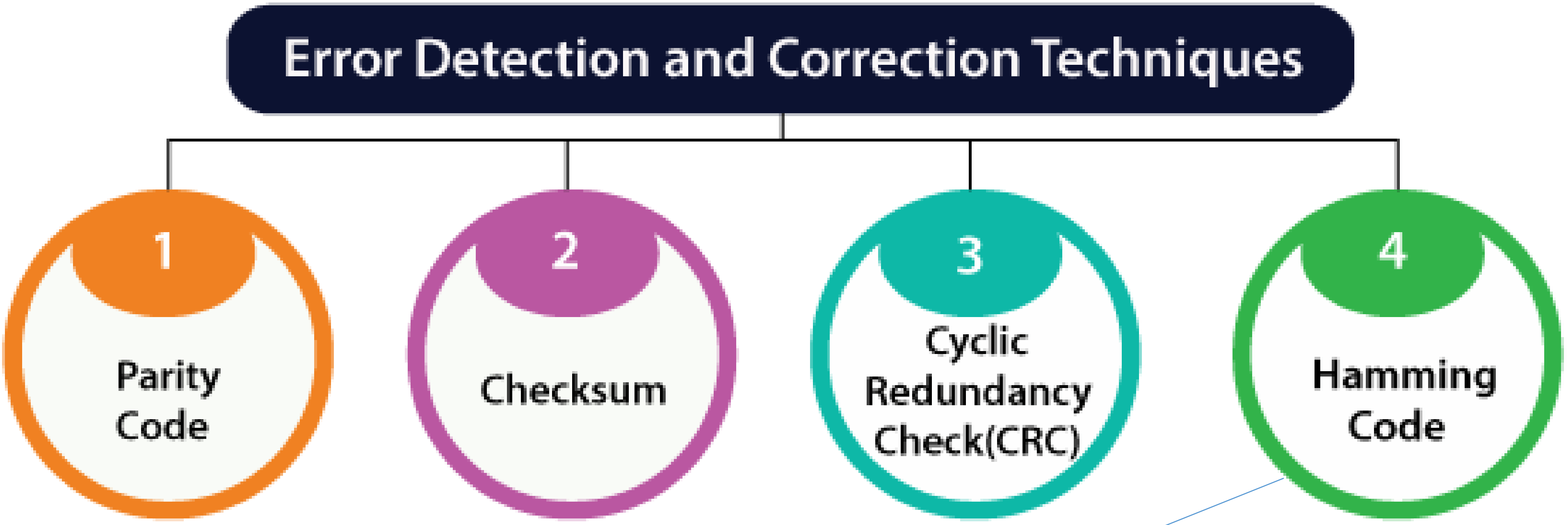
Gray Code	Decimal Equivalent
0000	0
0001	1
0011	2
0010	3
0110	4
0111	5
0101	6
0100	7
1100	8
1101	9
1111	10
1110	11
1010	12
1011	13
1001	14
1000	15

ASCII Code

	$b_7b_6b_5$							
$b_4b_3b_2b_1$	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

NUL	Null
SOH	Start of heading
STX	Start of text
ETX	End of text
EOT	End of transmission
ENQ	Enquiry
ACK	Acknowledge
BEL	Bell
BS	Backspace
HT	Horizontal tab
LF	Line feed
VT	Vertical tab
FF	Form feed
CR	Carriage return
SO	Shift out
SI	Shift in
SP	Space

Error-detecting and error-correcting codes



7 bit

Position	7	6	5	4	3	2	1
Bit	d_3	d_2	d_1	p_2	d_0	p_1	p_0

$$p_2 = d_3 \oplus d_2 \oplus d_1$$
$$p_1 = d_3 \oplus d_2 \oplus d_0$$
$$p_0 = d_3 \oplus d_1 \oplus d_0$$

15-bit Hamming code

Sl. No.	Parity Bits (n)	Data Bits ($2^n - n - 1$)	Total Bits ($2^n - 1$)
1	4	11	15
2	5	26	31

Bit Position:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Code	P1	P2	D1	P4	D2	D3	D4	P8	D5	D6	D7	D8	D9	D10	D11

Position of Parity bit Pi is at = 2^j, where j = 0, 1, 2, 3, …………

A Hamming Code with 4 Parity Bits: P1, P2, P4 and P8

Bit #	P8	P4	P2	P1
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Parity bit Pi checks all the bit positions, whose (Bit #) binary representation has 1’s in position i.

Examples:

P1 checks the bits 1, 3, 5, 7, 9, 11, 13, and 15

P2 checks the bits 2, 3, 6, 7, 10, 11, 14, and 15

P4 checks the bits 4, 5, 6, 7, 12, 13, 14, and 15

P8 checks the bits 8, 9, 10, 11, 12, 13, 14, and 15

$P1 = D1 \oplus D3 \oplus D5 \oplus D7 \oplus D9 \oplus D11 \oplus D13 \oplus D15$

$P2 = D2 \oplus D3 \oplus D6 \oplus D7 \oplus D10 \oplus D11 \oplus D14 \oplus D15$

$P4 = D4 \oplus D5 \oplus D6 \oplus D7 \oplus D12 \oplus D13 \oplus D14 \oplus D15$

$P8 = D8 \oplus D9 \oplus D10 \oplus D11 \oplus D12 \oplus D13 \oplus D14 \oplus D15$