Digital Logic Design

Lecture No. 3

Overview

- Decimal Codes - Binary Codes for Decimal Digits
- Binary Coded Decimal (BCD)
- BCD Arithmetic
- Alphanumeric Codes - ASCII Character Codes
- Signed Unsigned Binary Numbers
- Signed Magnitude Representation
- 2’s Complement Representation
- 1’s Complement
- Subtraction using 2’s Complement

DECIMAL CODES - Binary Codes for Decimal Digits

There are over 8,000 ways that you can write 10 elements using binary numbers of 4 bits. Some of the common codes are given in table.

Decimal	8,4,2,1	Excess3
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Binary Coded Decimal (BCD)

- The BCD code is the 8,4,2,1 code.
- **8, 4, 2, and 1 are weights**
- BCD is a *weighted* code
- This code is the simplest, most intuitive binary code for decimal digits and uses the same powers of 2 as a binary number, but only encodes the first ten values from 0 to 9.
- **Example: 0110 (6) = 0110 (6)**
- **1010 (10) = 0001 (1) + 0000 (0)**
- How many “invalid” code words are there?
- What are the “invalid” code words?

Warning: Conversion or Coding?

- Do NOT mix up conversion of a decimal number to a binary number with coding a decimal number with a BINARY CODE.
- $13_{10} = 1101_2$ **(This is conversion)**
- 13 $\Leftrightarrow$ 0001|0011 **(This is coding)**

BCD Arithmetic

Given a BCD code, we use binary arithmetic to add the digits**:**

8	**1000**	**Eight**
+5	**+0101**	**Plus 5**
13	**1101**	**is 13 (> 9)**

Note that the result is MORE THAN 9, so must be represented by two digits!

To correct the digit, add 6:

8	1000	Eight
+5	+0101	Plus 5
13	1101	is 13 (> 9)
	+0110	so add 6
	carry = 1 0011	leaving 3 + cy
	0001 | 0011	**Final answer (two digits)**

BCD Addition Example

- Add 1897_{BCD} to 2905_{BCD} showing carries and digit corrections.

	1	1	1	
	0001	**1000**	**1001**	**0111**
+	**0010**	**1001**	**0000**	**0101**
	0100	1 0010	1010	1100
		+0110	+0110	+0110
=	0100	1000	0000	0010

ALPHANUMERIC CODES - ASCII Character Codes

- American Standard Code for Information Interchange (ASCII)
- This code is a popular code used to represent information sent as character-based data. It uses 7-bits to represent:
- The seven bits of the code are designated by $B1$ through $B7$, with $B7$ being the most significant bit.
- The most significant three bits of the code determine the column of the table and the least significant four bits the row of the table.
  - 94 Graphic printing characters.
  - 34 Non-printing characters
- Some non-printing characters are used for text format (e.g. BS = Backspace).
- Other non-printing characters are used for record marking and flow control (e.g. STX and ETX start and end text areas).

Contd…

- The letter A, for example, is represented in ASCII as 1000001 (column 100, row 0001).

American Standard Code for Information Interchange (ASCII)

	$B_7B_6B_5$							
$B_4B_3B_2B_1$	000	001	010	011	100	101	110	111
0000	NULL	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

PARITY BIT Error-Detection Codes

- To detect errors in data communication and processing, an additional bit is sometimes added to a binary code word to define its parity.
- A parity bit is the extra bit included to make the total number of 1s in the resulting code word either even or odd.
- Parity can detect all single-bit errors and some multiple-bit errors.
- A code word has even parity if the number of 1's in the code word is even.
- A code word has odd parity if the number of 1's in the code word is odd.

4-Bit Parity Code Example

- Fill in the even and odd parity bits:

Even Parity		Odd Parity	
Message -	Parity	Message -	Parity
000 -		000 -	
001 -		001 -	
010 -		010 -	
011 -		011 -	
100 -		100 -	
101 -		101 -	
110 -		110 -	
111 -		111 -	

- The codeword "1111" has <u>even parity</u> and the codeword "1110" has <u>odd parity</u>. Both can be used to represent 3-bit data.

Unsigned Binary Numbers

- Unsigned numbers don't have any sign, these can contain only magnitude of the number.
- Conventional Binary numbers
- Positive Binary numbers are unsigned.
- Examples:
  - 1101 = 13

  1101 is the 4 bit unsigned magnitude of the decimal number 13.
  - 10101 = 21

  10101 is the 5 bit unsigned magnitude of the decimal number 21.

Range of unsigned binary number

- Range of unsigned binary number is from

$$0 \text{ to } (2^n-1)$$

- Therefore, range of 6 bit unsigned binary number is *from* 0 to (2^6-1).

- Minimum value 0 (i.e., 000000) to maximum value 63 (i.e., 111111).

Signed Binary Numbers

- Contain both positive and negative binary numbers.
- The most significant bit is the Sign bit.
- The sign bit represents sign of binary number.
- Sign bit is:
  - 0 for numbers >= 0 (positive numbers)
  - 1 for numbers < 0 (negative numbers)
- Examples:
  - 010010 -> positive binary number
  - 101100 -> negative binary number

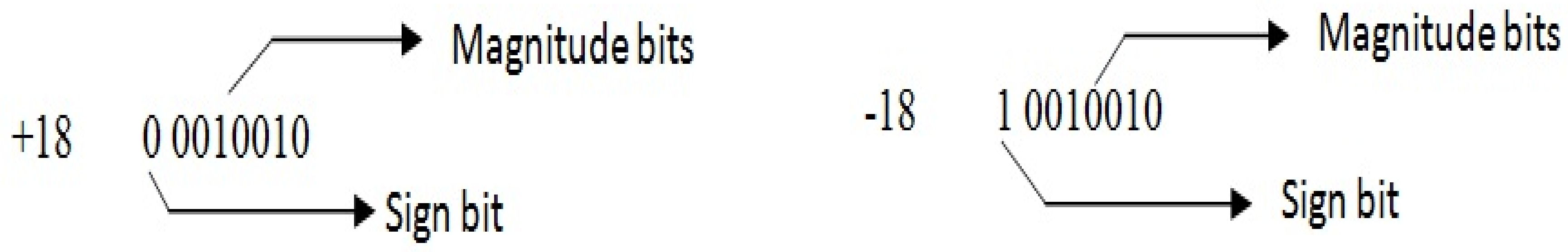
Range of signed binary number

- Range of n bit signed binary number is from $– 2^{n-1}+1$ to $2^{n-1}-1$
- Range of 6 bit Sign-Magnitude form binary number is from (-2^5+1) to (2^5-1)
- Minimum value -31 (i.e., 1 11111) to maximum value +31 (i.e., 0 11111).
- Zero (0) has two representation, -0 (i.e., 1 00000) and +0 (i.e., 0 00000). This is also a drawback.

Signed Magnitude

- To represent negative integers, we need a notation for negative values.

- Example: **Represent $(+18)_{10}$ $(-18)_{10}$ as Signed Binary Number**



One's Complement of Binary Numbers

- Is obtained by Complementing each bit.
- Replace each 0 by 1 and vice versa.
- Example:
  - The 1's complement of 10110000 is : 01001111
- If you add a number and its 1's complement, the answer is all ones.

$$
\begin{array}{r}
1\,0\,1\,1\,0\,0\,0\,0 \\
+\;0\,1\,0\,0\,1\,1\,1\,1 \\
\hline
1\,1\,1\,1\,1\,1\,1\,1
\end{array}
$$

Signed-1's-complement representation

- **Example:** Represent $(-5)_{10}$ as 1's-complement Signed Binary Number using 4-bits.

- **Step#1:** Write $(-5)_{10}$ as Signed Magnitude Number

  Signed Magnitude : 1101:

- **Step#2:** Convert Into 1`s-Complement leaving sign bit

  $(-5)_{10}$ as 1's-complement Signed Binary Number : 1010

Two’s Complement of Binary Numbers:

- Obtained in two steps:
  - Take one’s complement of a number.
  - Add 1 to the answer of one’s complement.
- Example: The two’s complement of 1100101 is:
  - One complement: 0011010
  - Add one: 0011010 + 0000001 = 0011011
- Signed numbers are stored in computers mostly in two’s complement form.

Signed-2's-complement representation

- **Example:** Represent $(-5)_{10}$ as 2's-complement Signed Binary Number using 4-bits.

- **Step#1:** Write $(-5)_{10}$ as Signed Magnitude Number

  Signed Magnitude : 1101:

- **Step#2:** Takes 2`s Complement ,leaving Sign bit

  2's-complement representation : 1011:

Signed Binary Numbers

Table 1.3
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	—	—

Addition of Signed Binary Numbers

$+6$	00000110
$+13$	00001101
$+19$	00010011

-6	11111010
$+13$	00001101
$+7$	00000111

$+6$	00000110
-13	11110011
-7	11111001

-6	11111010
-13	11110011
-19	11101101

Subtraction of Signed Binary Numbers

- Arithmetic Subtraction
  - In 2's-complement form:

1. Take the 2's complement of the subtrahend (including the sign bit) and add it to the minuend (including sign bit).
2. A carry out of sign-bit position is discarded.

$$(\pm A)-(+B)=(\pm A)+(-B)$$
$$(\pm A)-(-B)=(\pm A)+(+B)$$

- Example:

(– 6) – (– 13) → (11111010 – 11110011)

→ (11111010 + 00001101)

→ 00000111 (+ 7)

Subtraction by 1’s Complements

- Subtraction of unsigned numbers can also be done by means of the (r – 1)'s complement. Remember that the (r – 1) 's complement is one less then the r's complement.
- Example: Subtract by using 1's complement.

(a) $X - Y = 1010100 - 1000011$

$$
\begin{array}{rr}
X = & 1010100 \\
\text{1's complement of } Y = & +\,0111100 \\
\text{Sum} = & 10010000 \\
\text{End-around carry} = & \underline{+ \qquad\quad 1} \\
\text{Answer. } X - Y = & 0010001
\end{array}
$$

(b) $Y - X = 1000011 - 1010100$

$$
\begin{array}{rr}
Y = & 1000011 \\
\text{1's complement of } X = & \underline{+\,0101011} \\
\text{Sum} = & 1101110
\end{array}
$$

There is no end carry, Therefore, the answer is Y – X = – (1's complement of 1101110) = – 0010001.

Subtraction by 2’s Complement

- Example: Given the two binary numbers X = 1010100 and Y = 1000011, perform the subtraction (a) $X - Y$; and (b) $Y - X$, by using 2's complement.

(a)		
	X =	1010100
	2's complement of Y =	+0111101
	Sum =	10010001
	Discard end carry 2^7 =	–10000000
	Answer. X – Y =	0010001

(b)		
	Y =	1000011
	2's complement of X =	+ 0101100
	Sum =	1101111

There is no end carry. Therefore, the answer is Y – X = – (2's complement of 1101111) = – 0010001.

Any Questions?