CHW 261: Logic Design

Instructors:
Prof. Hala Zayed http://www.bu.edu.eg/staff/halazayed14
Dr. Ahmed Shalaby http://bu.edu.eg/staff/ahmedshalaby14#

bu.edu.eg/staff/ahmedshalaby14#

Benha University

Staff Search: Go

Login

- Benha University
- Home
- النسخة العربية
- My C.V.
- About
- Courses
- Publications
- Reports
- Published books
- Workshops / Conferences
- Supervised PhD
- Supervised MSc
- Supervised Projects
- Education
- Language skills
- Academic Positions
- Administrative Positions
- Memberships and awards

You are in:Home

Dr. Ahmed Shalaby

Academic Position: Lecturer

Current Administrative Position:

Ex-Administrative Position:

Faculty: Computers and Informatics

Department: Computer Science

Edu-Mail: ahmed.shalaby@fci.bu.edu.eg

Alternative Email: a.shalaby@ieee.org

Mobile:

Scientific Name:

Publications [Titles(4) :: Papers(2) :: Abstracts(4) **]**

Inlinks(6) :: Courses Files(4) | Total points :49

News

IOT Project Update [2016-11-14]
Smart Water Management Irrigation System (SWMIS) project moves to field test phase on November 2016. more

Research Interests

System on Chip, Network on Chip, VLSI, Embedded System, High Efficiency Video Coding (HEVC)

Benha University
Home
النسخة العربية
My C.V.
About
Courses
Publications
Theses
Reports
Published books
Workshops / Conferences
Supervised PhD
Supervised MSc
Supervised Projects
Education
Language skills
Academic Positions
Administrative Positions
Memberships and awards
Committees
Scientific Activities
Experience
External Links

Dr. Ahmed Shalaby :: External Links:

This page provides the service of adding favorite links of faculty member in the various fields. This feature assists in facilitating the access to such links.

URL
Google Code Jam !
2017 Code Jam World Finals in Dublin, Ireland - Highlight
Valeo Challenge !
IEEE Spectrum Magazine
MIT Technology Review
PROJECTION MAPPING
Participate: Open Source Projects
Practice coding. Compete. Find jobs.
How do they make Silicon Wafers and Computer Chips?
The Development Channel : FPGA and Embedded System
awesome Tech : Michi Yamamoto Channel
Online Courses
Digilent Design Contest
Xilinx Open Hardware Contest
Intel-FPGA Design Contests - Altera

Digital Fundamentals

CHAPTER 2
Number Systems, Operations, and Codes

Number Systems

Decimal Numbers

The position of each digit in a weighted number system is assigned a weight based on the **base** or **radix** of the system. The radix of decimal numbers is ten, because only ten symbols (0 through 9) are used to represent any number.

The column weights of decimal numbers are powers of ten that increase from right to left beginning with $10^0 = 1$:

$\ldots 10^5\ 10^4\ 10^3\ 10^2\ 10^1\ 10^0$.

For fractional decimal numbers, the column weights are negative powers of ten that decrease from left to right:

$10^2\ 10^1\ 10^0.\ 10^{-1}\ 10^{-2}\ 10^{-3}\ 10^{-4}\ \ldots$

Decimal Numbers

Decimal numbers can be expressed as the sum of the products of each digit times the column value for that digit. Thus, the number 9240 can be expressed as

$$(9 \times 10^3) + (2 \times 10^2) + (4 \times 10^1) + (0 \times 10^0)$$

or

$$9 \times 1{,}000 + 2 \times 100 + 4 \times 10 + 0 \times 1$$

Example

Express the number 480.52 as the sum of values of each digit.

Solution

$$480.52 = (4 \times 10^2) + (8 \times 10^1) + (0 \times 10^0) + (5 \times 10^{-1}) + (2 \times 10^{-2})$$

Binary Numbers

For digital systems, the binary number system is used. Binary has a radix of two and uses the digits 0 and 1 to represent quantities.

The column weights of binary numbers are powers of two that increase from right to left beginning with $2^0 = 1$:

$$\ldots 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0.$$

For fractional binary numbers, the column weights are negative powers of two that decrease from left to right:

$$2^2\ 2^1\ 2^0.\ 2^{-1}\ 2^{-2}\ 2^{-3}\ 2^{-4}\ \ldots$$

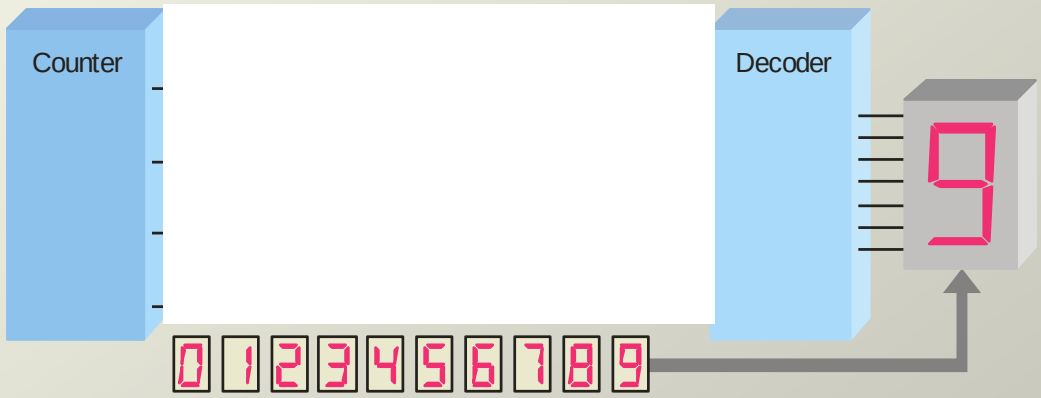
Decimal-to-Binary Conversion

Decimal Vs. Binary

A binary counting sequence for numbers from zero to fifteen is shown.

Notice the pattern of zeros and ones in each column.

Digital counters frequently have this same pattern of digits:



Decimal Number	Binary Number
0	0 0 0 0
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

Decimal-to-Binary Conversion

- Sum-of-weights method
- Repeated division-by-2 method
- Conversion of decimal fractions to binary

Decimal-to-Binary Conversion

Sum-of-weights method

Decimal numbers can be expressed as the sum of the products of each digit times the column value for that digit. Thus, the number 9240 can be expressed as

$$(9 \times 10^3) + (2 \times 10^2) + (4 \times 10^1) + (0 \times 10^0)$$

or

$$9 \times 1{,}000 + 2 \times 100 + 4 \times 10 + 0 \times 1$$

Decimal-to-Binary Conversion

Sum-of-weights method

You can convert a decimal whole number to binary by reversing the procedure. Write the decimal weight of each column and place 1's in the columns that sum to the decimal number.

Example

Convert the decimal number 49 to binary.

Solution

The column weights double in each position to the right. Write down column weights until the last number is larger than the one you want to convert.

2^6	2^5	2^4	2^3	2^2	2^1	2^0.
64	32	16	8	4	2	1.
0	1	1	0	0	0	1.

Decimal-to-Binary Conversion

Sum-of-weights method

Example Convert the binary number 100101.01 to decimal.

Solution Start by writing the column weights; then add the weights that correspond to each 1 in the number.

2^5 2^4 2^3 2^2 2^1 2^0. 2^{-1} 2^{-2}

32 16 8 4 2 1 . ½ ¼

1 0 0 1 0 1. 0 1 37¼

32 +4 +1 +¼ =

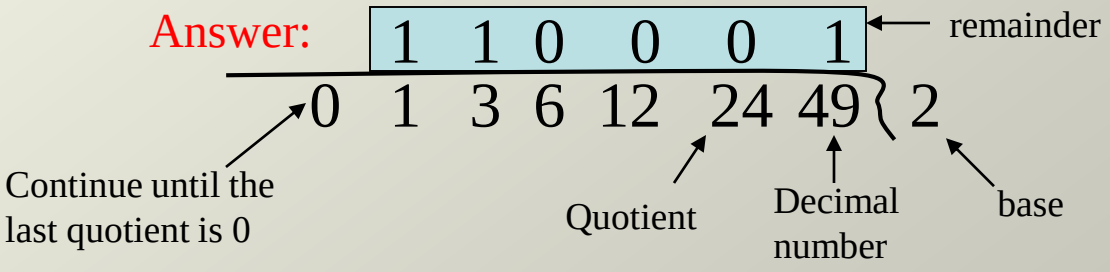
Decimal-to-Binary Conversion

Repeated division-by-2 method

You can convert decimal to any other base by repeatedly dividing by the base. For binary, repeatedly divide by 2:

Example Convert the decimal number 49 to binary by repeatedly dividing by 2.

Solution You can do this by "reverse division" and the answer will read from left to right. Put quotients to the left and remainders on top.



Decimal-to-Binary Conversion

Conversion of decimal fractions to binary

You can convert a decimal fraction to binary by repeatedly multiplying the fractional results of successive multiplications by 2. The carries form the binary number.

Example

Convert the decimal fraction 0.188 to binary by repeatedly multiplying the fractional results by 2.

Solution

$0.188 \times 2 = 0.376$ carry = 0
$0.376 \times 2 = 0.752$ carry = 0
$0.752 \times 2 = 1.504$ carry = 1
$0.504 \times 2 = 1.008$ carry = 1
$0.008 \times 2 = 0.016$ carry = 0

MSB

Answer = .00110 (for five significant digits)

Binary Arithmetic

Binary Arithmetic

- Binary addition
- Binary subtraction
- Binary multiplication
- Binary division

Binary Arithmetic

Binary Addition

The rules for binary addition are

0 + 0 = 0	Sum = 0, carry = 0
0 + 1 = 0	Sum = 1, carry = 0
1 + 0 = 0	Sum = 1, carry = 0
1 + 1 = 10	Sum = 0, carry = 1

When an input carry = 1 due to a previous result, the rules are

1 + 0 + 0 = 01	Sum = 1, carry = 0
1 + 0 + 1 = 10	Sum = 0, carry = 1
1 + 1 + 0 = 10	Sum = 0, carry = 1
1 + 1 + 1 = 10	Sum = 1, carry = 1

Binary Arithmetic

Binary Addition

Example Add the binary numbers 00111 and 10101 and show the equivalent decimal addition.

Solution

$$\begin{array}{rr} {\scriptstyle 0\ 1\ 1\ 1} & \\ 00111 & 7 \\ 10101 & 21 \\ \hline 11100 & = 28 \end{array}$$

Binary Arithmetic

Binary Subtraction

The rules for binary subtraction are

$$0 - 0 = 0$$
$$1 - 1 = 0$$
$$1 - 0 = 1$$
$$10 - 1 = 1 \text{ with a borrow of 1}$$

Example Subtract the binary number 00111 from 10101 and show the equivalent decimal subtraction.

Solution

$$\begin{array}{r} {\scriptstyle 1\ 1\ 1} \\ 10101 \\ 00111 \\ \hline 01110 \end{array} \quad \begin{array}{r} 21 \\ 7 \\ \hline = 14 \end{array}$$

Binary Arithmetic

Binary Multiplication

Partial products formed by multiplying a single digit of the multiplier with multiplicand.
Shifted partial products **summed** to form result.

Decimal		Binary
230	multiplicand	0101
x 42	multiplier	x 0111
460	partial products	0101
+ 920		0101
		0101
		+ 0000
9660	result	0100011

230 x 42 = 9660

5 x 7 = 35

Complements of Binary Numbers

Complements of Binary Numbers

- 1’s complements
- 2’s complements

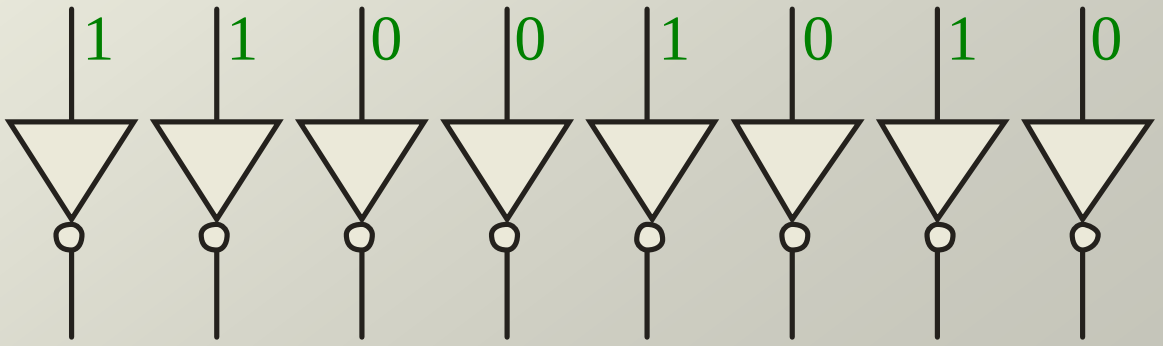
Complements of Binary Numbers

1's Complement

The 1's complement of a binary number is just the inverse of the digits. To form the 1's complement, change all 0's to 1's and all 1's to 0's.

For example, the 1's complement of 11001010 is 00110101

In digital circuits, the 1's complement is formed by using inverters:



Complements of Binary Numbers

2's Complement

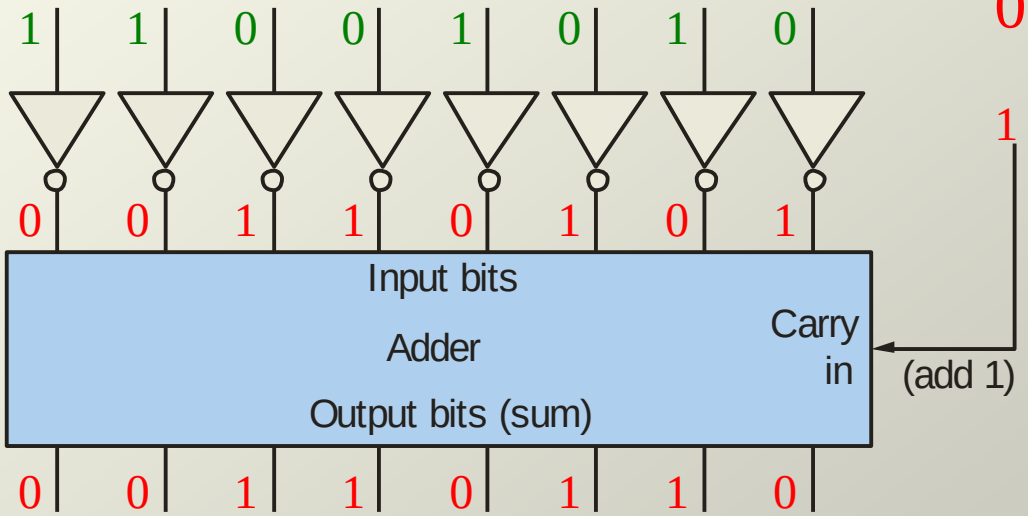
The 2's complement of a binary number is found by adding 1 to the LSB of the 1's complement.

Recall that the 1's complement of 11001010 is 00110101 (1's complement)

To form the 2's complement, add 1:

$$\begin{array}{r} 00110101 \\ +\quad\;\; 1 \\ \hline 00110110 \end{array}$$

(2's complement)



Signed Numbers

Signed Numbers

Signed Binary Numbers

There are several ways to represent signed binary numbers. In all cases, the **MSB in a signed number is the sign bit**, that tells you if the number is positive or negative.

Computers use a modified 2's complement for signed numbers. Positive numbers are stored in *true* form (with a 0 for the sign bit) and negative numbers are stored in *complement* form (with a 1 for the sign bit).

For example, the positive number 58 is written using 8-bits as 00111010 (true form).

Sign bit

Magnitude bits

The sign bit is the left-most bit in a signed binary number

1's and 2's complement form

Signed Binary Numbers

Negative numbers are written as the 2's complement of the corresponding positive number.

The negative number –58 is written as:

–58 = 11000110 (complement form)

Sign bit — Magnitude bits

An easy way to read a signed number that uses this notation is to assign the sign bit a column weight of –128 (for an 8-bit number). Then add the column weights for the 1's.

Example Assuming that the sign bit = –128, show that 11000110 = –58 as a 2's complement signed number:

Solution Column weights: –128 64 32 16 8 4 2 1.

1 1 0 0 0 1 1 0

–128 +64 +4 +2 = –58

Quiz

- **Question**：Represent 7.5 using 4 integer bits and 4 fraction bits.

Arithmetic Operations with Signed Numbers

- Addition
- Subtraction
- Multiplication
- Division

Arithmetic Operations with Signed Numbers - Addition

Arithmetic Operations with Signed Numbers

Using the signed number notation with negative numbers in 2's complement form simplifies addition and subtraction of signed numbers.

Rules for **addition**: Add the two signed numbers. Discard any final carries. The result is in signed form.

Examples:

00011110 = +30	00001110 = +14	11111111 = –1
00001111 = +15	11101111 = –17	11111000 = –8
00101101 = +45	11111101 = –3	1 11110111 = –9

Discard carry

Arithmetic Operations with Signed Numbers - Overflow

Arithmetic Operations with Signed Numbers

Note that if the number of bits required for the answer is exceeded, overflow will occur. This occurs only if both numbers have the same sign. The overflow will be indicated by an incorrect sign bit.

Two examples are:

01000000 = +128
01000001 = +129
10000001 = –126

Wrong! The answer is incorrect and the sign bit has changed.

10000001 = –127
10000001 = –127
Discard carry → 100000010 = +2

Overflow – Problem – Ariane5 !

https://www.youtube.com/watch?v=gp_D8r-2hwk

Arithmetic Operations with Signed Numbers

Four conditions for adding numbers:

- Both numbers are positive.
- Both numbers are negative.
- A positive number that is larger than a negative number.
- A negative number that is larger than a positive number.

Arithmetic Operations with Signed Numbers

Signs for Addition

- When both numbers are positive, the sum is positive.
- When both numbers are negative, the sum is negative (2's complement form). The carry bit is discarded.
- When the larger number is positive and the smaller is negative, the sum is positive. The carry is discarded.
- When the larger number is negative and the smaller is positive, the sum is negative (2's complement form).

Arithmetic Operations with Signed Numbers- Subtraction

Subtraction

- The sign of a positive or negative binary number is changed by taking its 2's complement.
- To subtract two signed numbers, take the 2's complement of the subtrahend and add. Discard any final carry bit.

Arithmetic Operations with Signed Numbers

There are two methods for multiplication:

- Direct addition
- Partial products

Multiplication is equivalent to adding a number to itself a number of times equal to the multiplier.

The method of partial products is the most commonly used.

Arithmetic Operations with Signed Numbers

Multiplication of Signed Numbers

- If the signs are the same, the product is positive.
- If the signs are different, the product is negative.

Division is equivalent to subtracting the divisor from the dividend a number of times equal to the quotient.

Arithmetic Operations with Signed Numbers

Division of Signed Numbers

- If the signs are the same, the quotient is positive.
- If the signs are different, the quotient is negative.

Hexadecimal Numbers

Hexadecimal Numbers

Hexadecimal Numbers

Hexadecimal uses sixteen characters to represent numbers: the numbers 0 through 9 and the alphabetic characters A through F.

Large binary number can easily be converted to hexadecimal by grouping bits 4 at a time and writing the equivalent hexadecimal character.

Example

Express $1001\ 0110\ 0000\ 1110_2$ in hexadecimal:

Solution

Group the binary number by 4-bits starting from the right. Thus, 960E

Decimal	Hexadecimal	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
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Hexadecimal Numbers

Hexadecimal Numbers

Hexadecimal is a weighted number system. The column weights are powers of 16, which increase from right to left.

Column weights { 16^3 16^2 16^1 16^0 . 4096 256 16 1 .

Example Express $1A2F_{16}$ in decimal.

Solution Start by writing the column weights:

4096	256	16	1
1	A	2	F_{16}

$1(4096) + 10(256) + 2(16) + 15(1) = 6703_{10}$

Decimal	Hexadecimal	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
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Hexadecimal Numbers

- Binary-to-hexadecimal conversion
  - Break the binary number into 4-bit groups.
  - Replace each group with the hexadecimal equivalent.
- Hexadecimal-to-decimal conversion
  - Convert the hexadecimal to groups of 4-bit binary.
  - Convert the binary to decimal.
- Decimal-to-hexadecimal conversion
  - Repeated division by 16

Octal Numbers

Octal Numbers

Octal Numbers

Octal uses eight characters the numbers 0 through 7 to represent numbers. There is no 8 or 9 character in octal.

Binary number can easily be converted to octal by grouping bits 3 at a time and writing the equivalent octal character for each group.

Example

Express $1\ 001\ 011\ 000\ 001\ 110_2$ in octal:

Solution

Group the binary number by 3-bits starting from the right. Thus, 113016_8

Decimal	Octal	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	10	1000
9	11	1001
10	12	1010
11	13	1011
12	14	1100
13	15	1101
14	16	1110
15	17	1111

Octal Numbers

Octal Numbers

Octal is also a weighted number system. The column weights are powers of 8, which increase from right to left.

Column weights { 8^3 8^2 8^1 8^0 . }
512 64 8 1

Example Express 3702_8 in decimal.

Solution Start by writing the column weights:

512 64 8 1
3 7 0 2_8

$3(512) + 7(64) + 0(8) + 2(1) = 1986_{10}$

Decimal	Octal	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	10	1000
9	11	1001
10	12	1010
11	13	1011
12	14	1100
13	15	1101
14	16	1110
15	17	1111

Binary Coded Decimal (BCD)

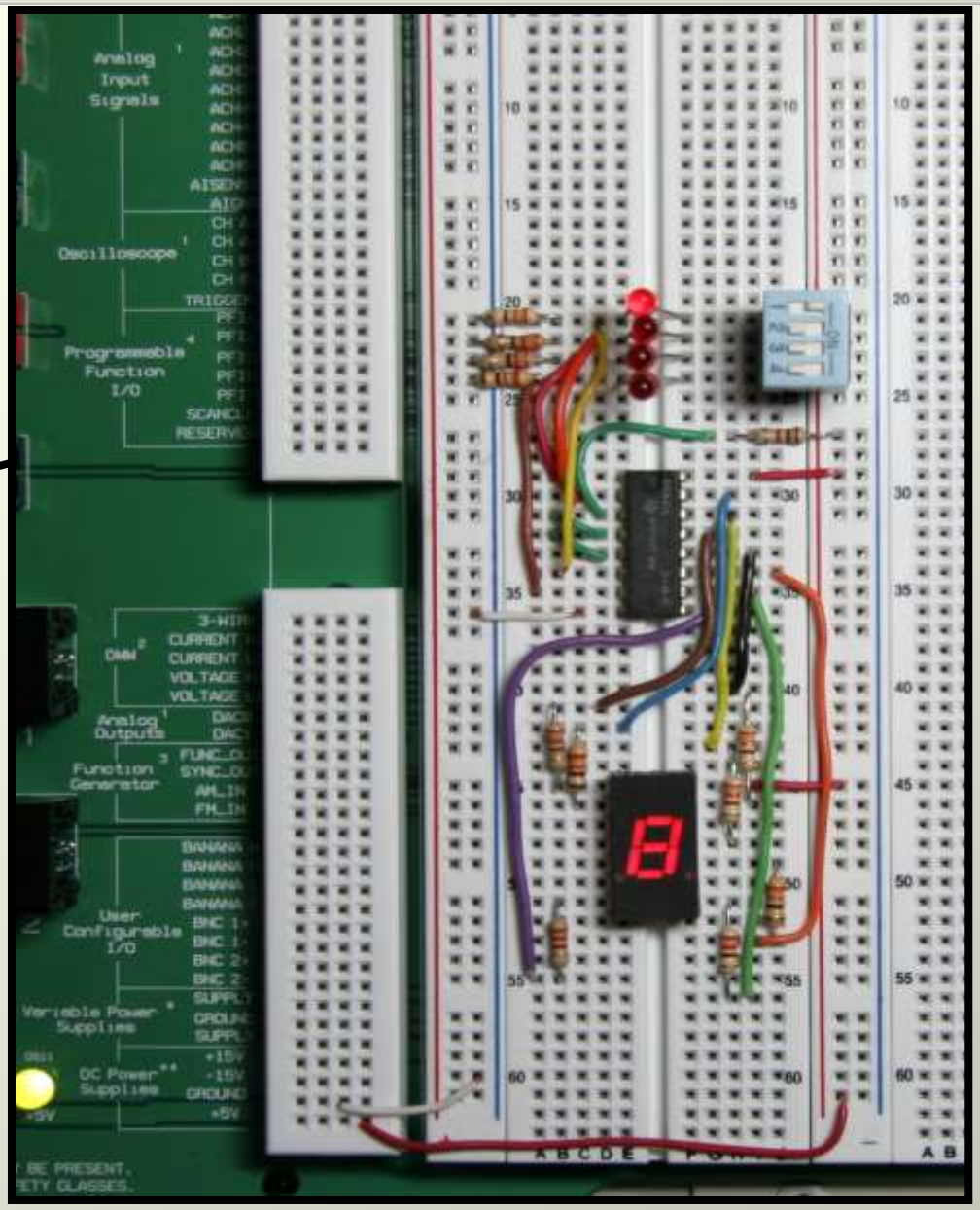
BCD Numbers

BCD

Binary coded decimal (BCD) is a weighted code that is commonly used in digital systems when it is necessary to show decimal numbers such as in clock displays.

The table illustrates the difference between straight binary and BCD. BCD represents each decimal digit with a 4-bit code. Notice that the codes 1010 through 1111 are not used in BCD.

Decimal	Binary	BCD
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	00010000
11	1011	00010001
12	1100	00010010
13	1101	00010011
14	1110	00010100
15	1111	00010101



Digital Codes

Digital Codes

- Gray code
- ASCII code

Digital Codes

Gray code

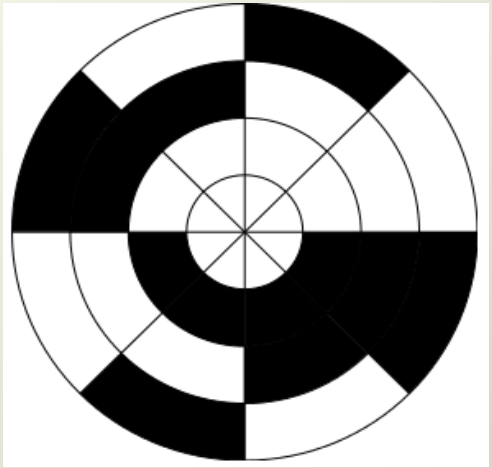
Gray code is an unweighted code that has a single bit change between one code word and the next in a sequence. Gray code is used to avoid problems in systems where an error can occur if more than one bit changes at a time.

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

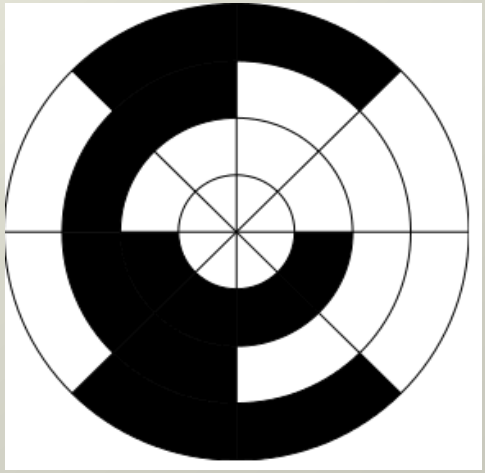
Digital Codes

Gray code

A shaft encoder is a typical application. Three IR emitter/detectors are used to encode the position of the shaft. The encoder on the left uses binary and can have three bits change together, creating a potential error. The encoder on the right uses gray code and only 1-bit changes, eliminating potential errors.



3-bit binary code: no good!



3-bit Gray code: better!

https://www.youtube.com/watch?v=cdeNxFkTwR0

Digital Codes

ASCII

ASCII is a code for alphanumeric characters and control characters. In its original form, ASCII encoded 128 characters and symbols using 7-bits. The first 32 characters are control characters, that are based on obsolete teletype requirements, so these characters are generally assigned to other functions in modern usage.

In 1981, IBM introduced extended ASCII, which is an 8-bit code and increased the character set to 256. Other extended sets (such as Unicode) have been introduced to handle characters in languages other than English.

Digital Codes

- ASCII code (graphic symbols 40h – 5Fh)

SYMBOL	DEC	BINARY	HEX	SYMBOL	DEC	BINARY	HEX
@	64	1000000	40	P	80	1010000	50
A	65	1000001	41	Q	81	1010001	51
B	66	1000010	42	R	82	1010010	52
C	67	1000011	43	S	83	1010011	53
D	68	1000100	44	T	84	1010100	54
E	69	1000101	45	U	85	1010101	55
F	70	1000110	46	V	86	1010110	56
G	71	1000111	47	W	87	1010111	57
H	72	1001000	48	X	88	1011000	58
I	73	1001001	49	Y	89	1011001	59
J	74	1001010	4A	Z	90	1011010	5A
K	75	1001011	4B	[	91	1011011	5B
L	76	1001100	4C	\	92	1011100	5C
M	77	1001101	4D	]	93	1011101	5D
N	78	1001110	4E	^	94	1011110	5E
O	79	1001111	4F	_	95	1011111	5F

Digital Codes

- ASCII code (graphic symbols 60h – 7Fh)

SYMBOL	DEC	BINARY	HEX	SYMBOL	DEC	BINARY	HEX
`	96	1100000	60	p	112	1110000	70
a	97	1100001	61	q	113	1110001	71
b	98	1100010	62	r	114	1110010	72
c	99	1100011	63	s	115	1110011	73
d	100	1100100	64	t	116	1110100	74
e	101	1100101	65	u	117	1110101	75
f	102	1100110	66	v	118	1110110	76
g	103	1100111	67	w	119	1110111	77
h	104	1101000	68	x	120	1111000	78
i	105	1101001	69	y	121	1111001	79
j	106	1101010	6A	z	122	1111010	7A
k	107	1101011	6B	{	123	1111011	7B
l	108	1101100	6C	|	124	1111100	7C
m	109	1101101	6D	}	125	1111101	7D
n	110	1101110	6E	~	126	1111110	7E
o	111	1101111	6F	Del	127	1111111	7F

Digital Codes

- ASCII code

Error Detection and Correction Codes

Error Detection and Correction Codes

- Parity error codes
- Hamming error codes

Parity error codes

Parity Method

The parity method is a method of error detection for simple transmission errors involving one bit (or an odd number of bits). A parity bit is an "extra" bit attached to a group of bits to force the number of 1's to be either even (even parity) or odd (odd parity).

Example The ASCII character for "a" is 1100001 and for "A" is 1000001. What is the correct bit to append to make both of these have odd parity?

Solution The ASCII "a" has an odd number of bits that are equal to 1; therefore the parity bit is 0. The ASCII "A" has an even number of bits that are equal to 1; therefore the parity bit is 1.

Parity error codes

The parity can detect up to One bit error and can't correct the error.

EVEN PARITY		ODD PARITY	
P	BCD	P	BCD
0	0000	1	0000
1	0001	0	0001
1	0010	0	0010
0	0011	1	0011
1	0100	0	0100
0	0101	1	0101
0	0110	1	0110
1	0111	0	0111
1	1000	0	1000
0	1001	1	1001

Hamming error codes

- Hamming code can detect up to 2 bits and correct 1 bit error.
- Hex equivalent of the data bits

0000000	0
0000111	1
0011011	2
0011110	3
0101010	4
0101101	5
0110011	6
0110100	7
1001011	8
1001100	9
1010010	A
1010101	B
1100001	C
1100110	D
1111000	E
1111111	F

Fixed-Point / Floating-Point Presentation

Signed Fixed-Point Numbers

- **Representations:**
  - Sign/magnitude
  - Two's complement
- **Example:** Represent -7.5_{10} using 4 integer and 4 fraction bits
  - **Sign/magnitude:**

    11111000
  - **2's complement:**

    1. +7.5: 01111000
    2. Invert bits: 10000111
    3. Add 1 to lsb: + 1

    10001000

Unsigned/Signed Numbers - Range of Values

Number System	Range
Unsigned	$[0, 2^N-1]$
Sign/Magnitude	$[-(2^{N-1}-1), 2^{N-1}-1]$
Two's Complement	$[-2^{N-1}, 2^{N-1}-1]$

Example Assuming that the N = 8, show the range of values for Unsigned/Signed/Two's Complement Numbering System:

Solution

Unsigned	[0, 255]
Sign/Magnitude	[-127,127]
Two's Complement	[-128,127]

Floating-Point Numbers

- Binary point floats to the right of the most significant 1
- Similar to decimal scientific notation

- For example, write 273_{10} in scientific notation:

$$\mathbf{273 = 2.73 \times 10^2}$$

- In general, a number is written in scientific notation as:

$$\mathbf{\pm M \times B^E}$$

  - **M** = mantissa
  - **B** = base
  - **E** = exponent
  - In the example, M = 2.73, B = 10, and E = 2

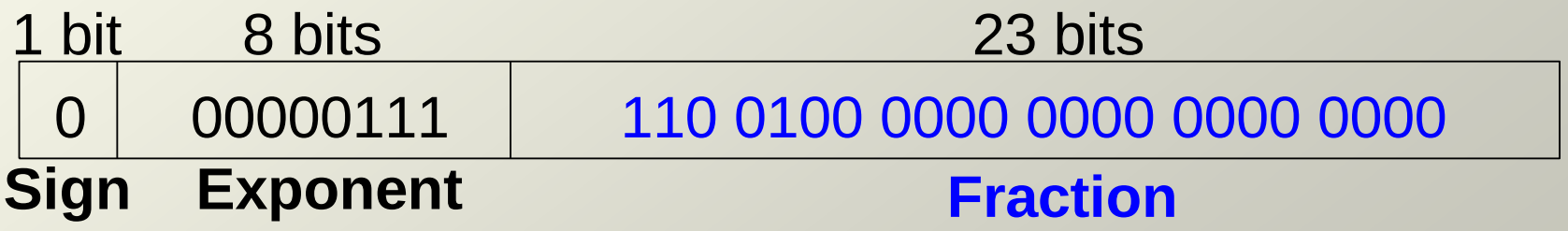
Floating-Point Numbers

- **Example:** represent the value 228_{10} using a 32-bit floating point representation

We show three versions –final version is called the **IEEE 754 floating-point standard**

Floating-Point Representation Example

- First bit of the mantissa is always 1:
  - $228_{10} = 11100100_2 = \mathbf{1}.11001 \times 2^7$
- So, no need to store it: *implicit leading 1*
- Store just fraction bits in 23-bit field



Floating-Point Representation Example ..Cont…

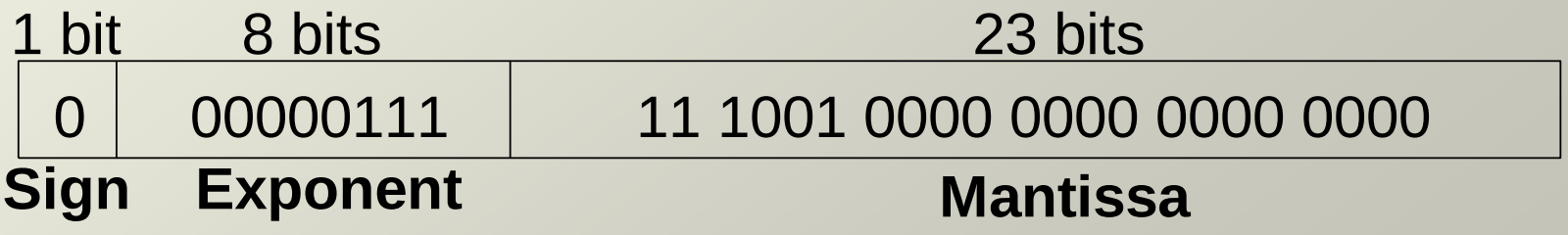
1. Convert decimal to binary (**don't reverse steps 1 & 2!**):

   $228_{10} = 11100100_2$

2. Write the number in "binary scientific notation":

   $11100100_2 = 1.11001_2 \times 2^7$

3. Fill in each field of the 32-bit floating point number:
   - The sign bit is positive (0)
   - The 8 exponent bits represent the value 7
   - The remaining 23 bits are the mantissa



Floating-Point Representation Example ..Cont…

- *Biased exponent*: bias = 127 (01111111_2)
  - Biased exponent = bias + exponent
  - Exponent of 7 is stored as:

$$127 + 7 = 134 = 0x10000110_2$$

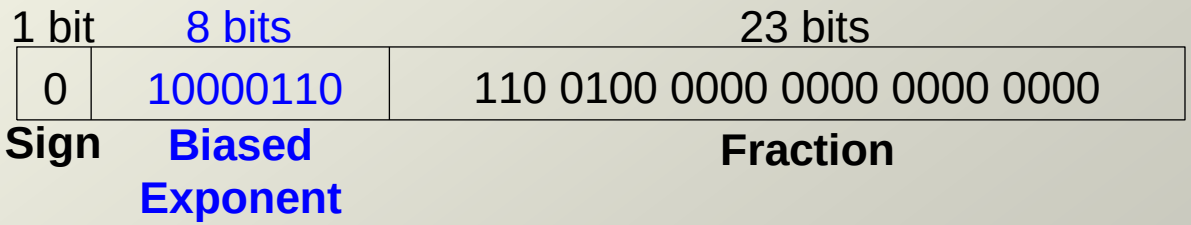
- The **IEEE 754 32-bit floating-point representation** of 228_{10}

1 bit	8 bits	23 bits
0	10000110	110 0100 0000 0000 0000 0000
Sign	**Biased Exponent**	**Fraction**

in hexadecimal: **0x43640000**

Floating-Point Precision

- **Single-Precision:**
  - 32-bit
  - 1 sign bit, 8 exponent bits, 23 fraction bits
  - bias = 127



- **Double-Precision:**
  - 64-bit
  - 1 sign bit, 11 exponent bits, 52 fraction bits
  - bias = 1023