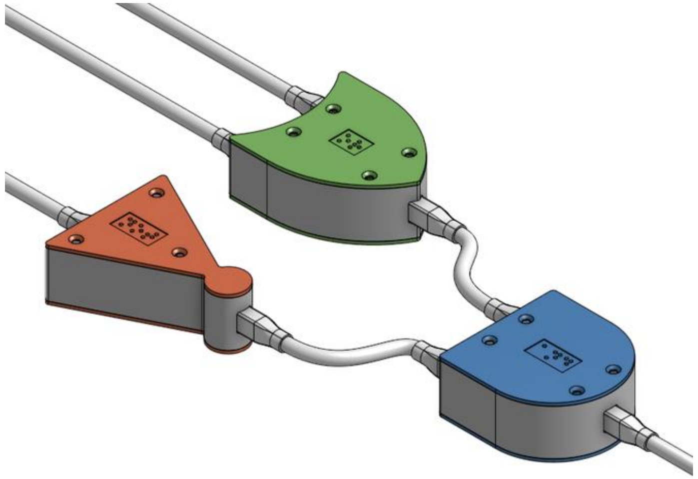
جامعة شقراء
Shaqra University



CS 211 - Digital Logic Design
211 عال - تصميم المنطق الرقمي

First Term - 1439/1440
Lecture #4

Dr. Hazem Ibrahim Shehata
Assistant Professor
College of Computing and Information Technology

Administrivia

- Assignment #1:
  - To be released on Sunday.
  - Work in groups of two.

Website: http://hshehata.github.io/courses/su/cs211

Floating-Point Numbers

- **Usage**: To represent very large/small integers and positive/negative real numbers with small number of bits!
- **Idea**: Use scientific notation ➔ number consists of three components: sign, mantissa (1+fraction), exponent.
- **Example**:
  - $101100100000000000_2 \rightarrow 1.011001 * 2^{10001}$
  - $-0.000000000001101_2 \rightarrow -1.101 * 2^{-1100}$

sign mantissa exponent

Single-Precision Floating-Point Numbers

- In IEEE standard 754, single-precision floating-point numbers are represented by 32 bits:

32 bits

Sign (S)	Exponent (E)	Fraction (F)
1 bit	8 bits	23 bits

- Sign (S): 0 ➔ positive, 1 ➔ negative
- Biased Exponent (E): Actual exponent plus 127_{10} (or 01111111_2).
- Fraction (F): Fractional part of mantissa.

Conversion: Decimal ➔ Single-Precision

- **Method**: Convert from decimal to binary, then rewrite binary number in the form: $\pm 1.xxx...xxx * 2^{\pm xxx...xxx}$
- **Example**: Represent -100.25_{10} as a single-precision floating-point number.
- **Solution**:
  - -100.25_{10} ➔ -1100100.01_2
  - ➔ $-1.10010001 * 2^{110}$
  - S=1, E = 110 + 01111111 = 10000101, F=10010001000...000
  - Number = 1 10000101 10010001000000000000000

Conversion: Single-Precision ➔ Decimal

- Method: Convert to binary using: $(-1)^S(1+F)(2^{E-127})$, then convert to decimal.
- Example: Convert the following single-precision FP number to decimal: 1 01111110 01000000000000000000000.
- Solution:
  - Number = $(-1)^1(1 + 0.010\ldots000)(2^{126-127})$
  - $= -1.01 * 2^{-1} = -0.101_2$
  - $= -(0.5 + 0.125) = -0.625$

Digital Codes

- Many specialized codes are used in digital systems:
  - Strictly numeric: represent numbers only.
    - Weighted - Arithmetic ➔ Example: Binary-Coded Decimal (BCD).
    - Unweighted – Non-arithmetic ➔ Example: Gray Code.
    - …
  - Alpha-numeric: represent numbers, symbols, letters, … etc.
    - Example: ASCII Code.
    - …

Binary-Coded Decimal (BCD)

- A way of representing decimal numbers using bits!!
- Each decimal digit is encoded using 4-bits:

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

- Note: the codes 1010 through 1111 are not used in BCD!!

Conversion: Decimal ➔ BCD

- Method: Replace each decimal digit with the appropriate 4-bit code.
- Example: Convert 4905_{10} to BCD.
- Solution:

4	9	0	5
↓	↓	↓	↓
0100	1001	0000	0101

Conversion: BCD ➔ Decimal

- Method: Group each 4 bits and replace with equivalent decimal digit.
- Example: Convert $1000000101110110 1001_{BCD}$ to Decimal.
- Solution:

1000	0001	0111	0110	1001
↓	↓	↓	↓	↓
8	1	7	6	9

BCD Addition

- Step 1: Add using rules for binary addition
- Step 2: If 4-bit sum <= 9, it is a valid BCD number.
- Step 3: If a 4-bit sum > 9, or if a carry out of 4-bit group generated, Add 6 (0110) to it ➔ correction.
- Example:

```
  0110 0101        0110 0101          1
 +0011 0001       +0001 0111        0110 1000
 ----------       ----------       +0001 1001
  1001 0110        0111 1100       ----------
                       +0110        1000 0001
                  ----------            +0110
                   1000 0010       ----------
                                    1000 0111
```

BCD Subtraction

- Step 1: Subtract using rules for binary addition
- Step 2: If 4-bit difference <= 9, it is a valid BCD number.
- Step 3: If a 4-bit sum > 9, or if a carry out of 4-bit group generated, Add 6 (0110) to it ➔ correction.
- Example:

$$
\begin{array}{r}
0110\ \ 0101 \\
-0011\ \ 0001 \\
\hline
0011\ \ 0100
\end{array}
\qquad
\begin{array}{r}
0110\ \ 0101 \\
-0001\ \ 0111 \\
\hline
0100\ \ \boxed{1110} \\
-0110 \\
\hline
1000\ \ 1000
\end{array}
\qquad
\begin{array}{r}
\boxed{1} \\
0110\ \ 0010 \\
-0001\ \ 1001 \\
\hline
0100\ \ 1001 \\
-0110 \\
\hline
0100\ \ 0011
\end{array}
$$

Gray Code

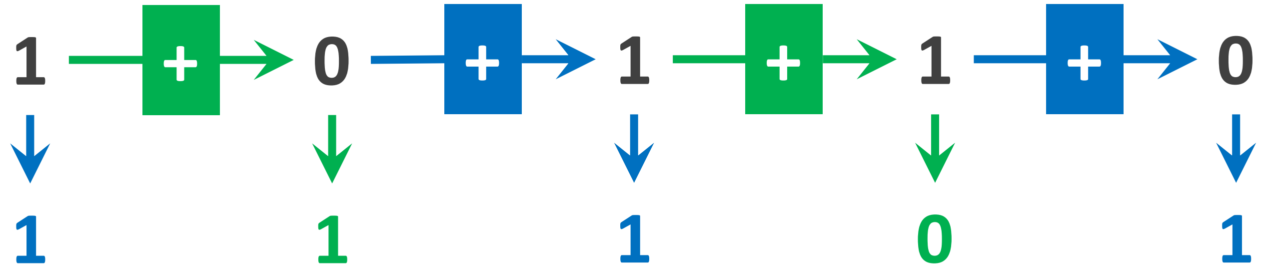
- Unweighted & Non-arithmetic code
  - No specific weights assigned to the bit positions!
- Most important feature:
  - Only 1-bit change from one code word to next in sequence!
- Has many applications:
  - Whenever error susceptibility increases with number of bit changes between adjacent numbers in a sequence!

Four-bit Gray Code

Decimal Number	Binary Number				Gray Code			
00	0	0	0	0	0	0	0	0
01	0	0	0	1	0	0	0	1
02	0	0	1	0	0	0	1	1
03	0	0	1	1	0	0	1	0
04	0	1	0	0	0	1	1	0
05	0	1	0	1	0	1	1	1
06	0	1	1	0	0	1	0	1
07	0	1	1	1	0	1	0	0
08	1	0	0	0	1	1	0	0
09	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

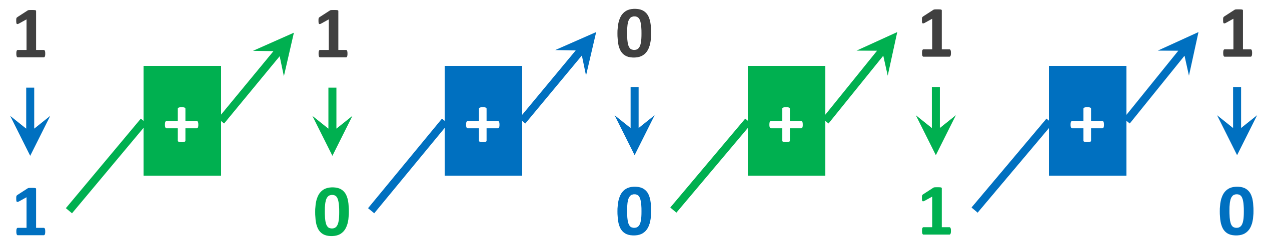
Conversion: Binary ➔ Gray Code

- **Method**:
  1. Most significant bit (MSB) in Gray code is same as corresponding MSB in binary.
  2. Going from left to right, add each adjacent pair of binary code bits to get next Gray code bit. Discard carries.
- **Example**: Convert 10110_2 to Gray code.
- **Solution**:



Conversion: Gray Code → Binary

- **Method**:
  1. Most significant bit in binary code is the same as corresponding bit in the Gray code.
  2. Add each binary code bit generated to Gray code bit in next adjacent position. Discard carries.
- **Example**: Convert Gray code 11011 to binary.
- **Solution**:



ASCII Code

- Stands for:
  - American Standard Code for Information Interchange
- Alphanumeric code used in most computers.
- When you enter a letter, a number, or control command on keyboard, corresponding ASCII code goes into computer.
- ASCII code consists of 7 bits ➔ Represents 128 characters.

American Standard Code for Information Interchange (ASCII).

Control Characters				Graphic Symbols											
Name	**Dec**	**Binary**	**Hex**	**Symbol**	**Dec**	**Binary**	**Hex**	**Symbol**	**Dec**	**Binary**	**Hex**	**Symbol**	**Dec**	**Binary**	**Hex**
NUL	0	0000000	00	space	32	0100000	20	@	64	1000000	40	`	96	1100000	60
SOH	1	0000001	01	!	33	0100001	21	A	65	1000001	41	a	97	1100001	61
STX	2	0000010	02	"	34	0100010	22	B	66	1000010	42	b	98	1100010	62
ETX	3	0000011	03	#	35	0100011	23	C	67	1000011	43	c	99	1100011	63
EOT	4	0000100	04	$	36	0100100	24	D	68	1000100	44	d	100	1100100	64
ENQ	5	0000101	05	%	37	0100101	25	E	69	1000101	45	e	101	1100101	65
ACK	6	0000110	06	&	38	0100110	26	F	70	1000110	46	f	102	1100110	66
BEL	7	0000111	07	'	39	0100111	27	G	71	1000111	47	g	103	1100111	67
BS	8	0001000	08	(	40	0101000	28	H	72	1001000	48	h	104	1101000	68
HT	9	0001001	09	)	41	0101001	29	I	73	1001001	49	i	105	1101001	69
LF	10	0001010	0A	*	42	0101010	2A	J	74	1001010	4A	j	106	1101010	6A
VT	11	0001011	0B	+	43	0101011	2B	K	75	1001011	4B	k	107	1101011	6B
FF	12	0001100	0C	,	44	0101100	2C	L	76	1001100	4C	l	108	1101100	6C
CR	13	0001101	0D	−	45	0101101	2D	M	77	1001101	4D	m	109	1101101	6D
SO	14	0001110	0E	.	46	0101110	2E	N	78	1001110	4E	n	110	1101110	6E
SI	15	0001111	0F	/	47	0101111	2F	O	79	1001111	4F	o	111	1101111	6F
DLE	16	0010000	10	0	48	0110000	30	P	80	1010000	50	p	112	1110000	70
DC1	17	0010001	11	1	49	0110001	31	Q	81	1010001	51	q	113	1110001	71
DC2	18	0010010	12	2	50	0110010	32	R	82	1010010	52	r	114	1110010	72
DC3	19	0010011	13	3	51	0110011	33	S	83	1010011	53	s	115	1110011	73
DC4	20	0010100	14	4	52	0110100	34	T	84	1010100	54	t	116	1110100	74
NAK	21	0010101	15	5	53	0110101	35	U	85	1010101	55	u	117	1110101	75
SYN	22	0010110	16	6	54	0110110	36	V	86	1010110	56	v	118	1110110	76
ETB	23	0010111	17	7	55	0110111	37	W	87	1010111	57	w	119	1110111	77
CAN	24	0011000	18	8	56	0111000	38	X	88	1011000	58	x	120	1111000	78
EM	25	0011001	19	9	57	0111001	39	Y	89	1011001	59	y	121	1111001	79
SUB	26	0011010	1A	:	58	0111010	3A	Z	90	1011010	5A	z	122	1111010	7A
ESC	27	0011011	1B	;	59	0111011	3B	[	91	1011011	5B	{	123	1111011	7B
FS	28	0011100	1C	<	60	0111100	3C	\	92	1011100	5C	|	124	1111100	7C
GS	29	0011101	1D	=	61	0111101	3D	]	93	1011101	5D	}	125	1111101	7D
RS	30	0011110	1E	>	62	0111110	3E	^	94	1011110	5E	~	126	1111110	7E
US	31	0011111	1F	?	63	0111111	3F	_	95	1011111	5F	Del	127	1111111	7F

Error Codes

- Simplest bit-error detection → Parity method.
- Parity method can detect 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).

Error Codes

- **Example**: Add an even parity bit to the 7-bit ASCII code for the letter “K”. Then illustrate how single-bit transmission errors can be detected.
- **Solution**:
  - ASCII code for “K” = 1001011
  - Number of 1’s is even → parity bit = 0
  - Transmitter sends 8-bit: 01001011 (parity & ASCII code f “K”)
  - Suppose a single-bit error happens to 3rd bit → 010011̲11
  - Receiver will find an odd number of 1’s → error detected!!!

Reading Material

- Floyd, Chapter 2:
  - Pages 63 - 65
  - Pages 82 - 90
  - Pages 92 - 93