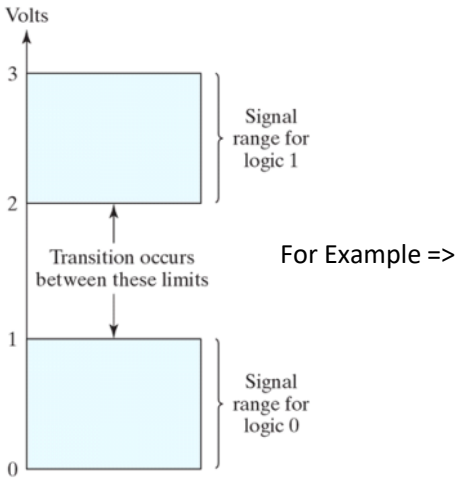
MSE 352: Digital Logic and Microcontrollers

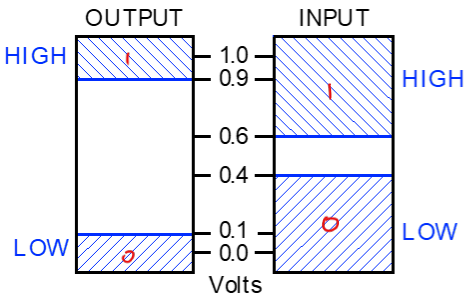
Chapter 1
Digital Systems and Binary Numbers

Mohammad Narimani
School of Mechatronic Systems Engineering
Simon Fraser University

SFU SIMON FRASER UNIVERSITY SURREY MSE 352 Digital Logic and Microcontrollers

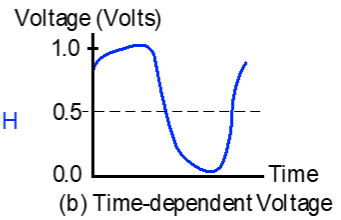


For Example =>

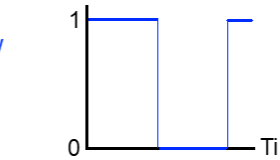


(a) Example voltage ranges

RANGES USED AS FILTERS



(b) Time-dependent Voltage



(c) Binary model of time-dependent voltage

Number Systems

$$(\text{Number})_r = \sum_{i=0}^{i=n-1} A_i \cdot r^i + \sum_{j=-m}^{j=-1} A_j \cdot r^j$$

(Integer Portion) + (Fraction Portion)

		General	Decimal	Binary
Radix (Base)		**r**	**10**	**2**
Digits		**0 => r - 1**	**0 => 9**	**0 => 1**
Powers of Radix	**0**	r^0	**1**	**1**
	1	r^1	**10**	**2**
	2	r^2	**100**	**4**
	3	r^3	**1000**	**8**
	4	r^4	**10,000**	**16**
	5	r^5	**100,000**	**32**
	-1	r^{-1}	**0.1**	**0.5**
	-2	r^{-2}	**0.01**	**0.25**
	-3	r^{-3}	**0.001**	**0.125**
	-4	r^{-4}	**0.0001**	**0.0625**
	-5	r^{-5}	**0.00001**	**0.03125**

$(149)_{10} = 9 \times 10^0 + 4 \times 10^1 + 1 \times 10^2$

Digits $0 \rightarrow 9$ or $0-(r-1)$ r = base value

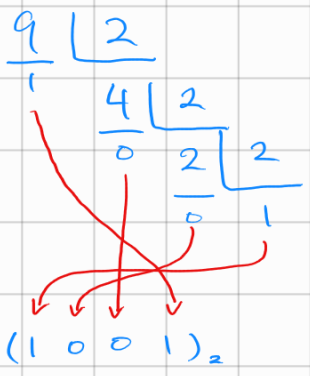
MSB ↘ ↙ LSB

$(101101)_2 = 1\times2^0 + 0\times2^1 + 1\times2^2 + 1\times2^3 + 0\times2^4 + 1\times2^5$

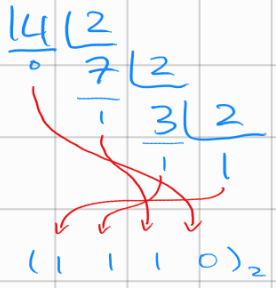
$= 1 + 4 + 8 + 32 = (45)_{10}$

$(101 \cdot 1001)_2 = 1\times2^0 + 0\times2^1 + 1\times2^2 + 1\times2^{-1} + 0\times2^{-2} + 0\times2^{-3} + 1\times2^{-4}$

$9_{10} \rightarrow 1001_2$



$14_{10} \rightarrow (1110)_2$



$(12.56)_{10} = (1100 \cdot 100011)$ accuracy of 6

$12 + 0.56$

$0.56 \times 2 = 1.12$ $\quad 0.24 \times 2 = 0.48$ $\quad 0.96 \times 2 = 1.92$

$.12 \times 2 = 0.24$ $\quad 0.48 \times 2 = 0.96$ $\quad 0.92 \times 2 = 1.84$

Accurately equivalent value

$(1.25)_{10} = (1.01)_2$

$0.25 \times 2 = 0.5$

$0.5 \times 2 = 1.0$

$0 \times 2 = 0$ and so on...

- 2^{10} (1024) is Kilo, denoted "K"
- 2^{20} (1,048,576) is Mega, denoted "M"
- 2^{30} (1,073, 741,824) is Giga, denoted "G"
- 2^{40} (1,099,511,627,776) is Tera, denoted “T"

Number Systems

Decimal (base 10)	Binary (base 2)	Octal (base 8)	Hexadecimal (base 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Example: Convert $(A06F)$ to Binary

Example: Convert $(1100\ 1001\ 00110010)_2$ to Hexadecimal

Example: Convert $(1203)_{10}$ to Binary

Example: Convert $(1203)_{10}$ to Hexadecimal

Example: Convert $(1203)_{10}$ to Octal

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	—	—

Two's Complement -

If 2 Two's Complement numbers are added, and they both have the same sign (both positive or both negative), then overflow occurs if the result has the opposite sign.

- -39 + 92 = 53:

[1]	1		1	1				
	1	1	0	1	1	0	0	1
+	0	1	0	1	1	1	0	0
	0	0	1	1	0	1	0	1

Carryout without overflow. Sum is correct.

- -19 + -7 = -26:

[1]	1	1	1	1			1	
	1	1	1	0	1	1	0	1
+	1	1	1	1	1	0	0	1
	1	1	1	0	0	1	1	0

Carryout without overflow. Sum is correct.

- 44 + 45 = 89:

		1		1	1			
	0	0	1	0	1	1	0	0
+	0	0	1	0	1	1	0	1
	0	1	0	1	1	0	0	1

No overflow nor carryout.

- 104 + 45 = 149:

	1	1		1				
	0	1	1	0	1	0	0	0
+	0	0	1	0	1	1	0	1
	1	0	0	1	0	1	0	1

Overflow, no carryout. Sum is not correct.

- -75 + 59 = -16:

		1	1	1	1	1	1	
	1	0	1	1	0	1	0	1
+	0	0	1	1	1	0	1	1
	1	1	1	1	0	0	0	0

No overflow nor carryout.

- -103 + -69 = -172:

[1]		1	1	1		1	1	
	1	0	0	1	1	0	0	1
+	1	0	1	1	1	0	1	1
	0	1	0	1	0	1	0	0

Overflow, with incidental carryout. Sum is not correct.

- 10 + -3 = 7:

[1]	1	1	1	1				
	0	0	0	0	1	0	1	0
+	1	1	1	1	1	1	0	1
	0	0	0	0	0	1	1	1

Carryout without overflow. Sum is correct.

- 127 + 1 = 128:

	1	1	1	1	1	1	1	
	0	1	1	1	1	1	1	1
+	0	0	0	0	0	0	0	1
	1	0	0	0	0	0	0	0

Overflow, no carryout. Sum is not correct.

- -1 + 1 = 0:

[1]	1	1	1	1	1	1	1	
	1	1	1	1	1	1	1	1
+	0	0	0	0	0	0	0	1
	0	0	0	0	0	0	0	0

Carryout without overflow. Sum is correct.

- Overflow never occurs when adding operands with different signs.

Binary Codes- Binary-Coded Decimal (BCD)

This code is the simplest way to represent decimal digits by binary values.

Example: Show 2823 in a BCD code

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

BCD Arithmetic:

Given a BCD code, we use binary arithmetic to add the digits if the result is MORE THAN 9, the result must be represented by two digits. To correct the digit:

- add 6 (in binary) to the result

- add one to the next significant digit

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

$$\begin{array}{rrrrr} & 0001 & 1000 & 1001 & 0111 \\ + & \underline{0010} & \underline{1001} & \underline{0000} & \underline{0101} \\ & & & & \\ & \_\_\_\_ & \_\_\_\_ & \_\_\_\_ & \_\_\_\_ \end{array}$$

Binary Codes- Excess-3

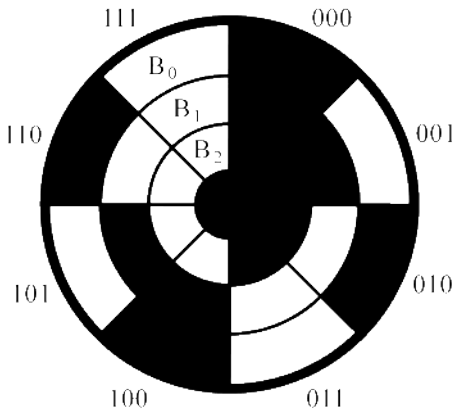
Codes have the property that the codes for 0 and 9, 1 and 8, etc. can be obtained from each other by replacing the 0's with the 1's and vice-versa in the code words

Decimal Digit	Excess-3
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

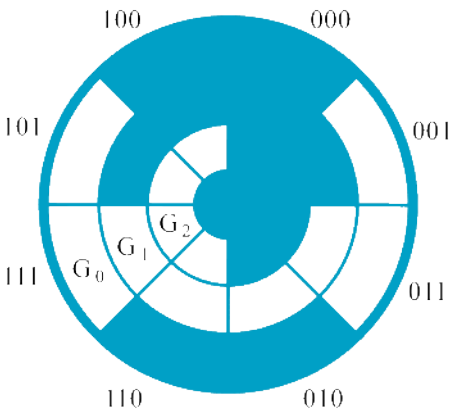
Binary Codes -Gray Code

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

Example: Optical Shaft Encoder



(a) Binary Code for Positions 0 through 7



(b) Gray Code for Positions 0 through 7

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)

PARITY BIT Error-Detection Codes

An extra bit appended onto the code word to make the number of 1's odd or even. 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.

Example: Fill in the even and odd parity bits:

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