EEE283

Digital Logic Design

Course Instructor: A K M Anindya Alam

Topic 2:

Review of Number Systems

Number System: Binary

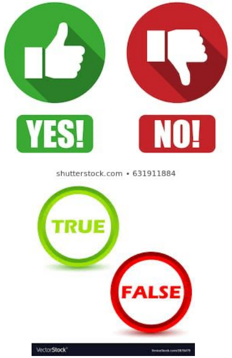
Binary: Bi (two)

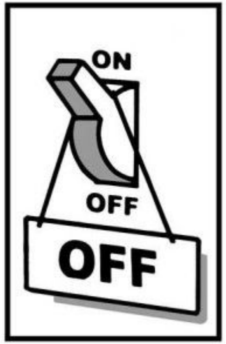
Each subsequent digit a multiplier of 2

Base: 2

Symbols:

01, TF





N/2	Q	R
13/2	6	**1**
6/2	3	**0**
3/2	1	**1**
1/2	0	**1**

$$\mathbf{13|_{10} = 1101|_{2}}$$

N-Base Number System

- The system uses n different symbols
- Each subsequent digit: n-times
- For this course we denote a number, N with it's base as $\mathbf{N}|_{\mathbf{n},}$ n is denoted in decimal
- For n<10, we use symbols as 0-9, and for n>10 we use symbols as ABCD…

Octal and Hexadecimal Systems

- Base-8
- Octal
- Symbols: 01234567

- Base-16
- Hexadecimal
- Symbols: 0123456789ABCDEF
- $F|_{16} = 15|_{10}$

Decimal and Binary Numbers

- **Base 10** (our everyday number system)

1000's column, 100's column, 10's column, 1's column

$$5374_{10} = \underbrace{5 \times 10^3}_{\text{five thousands}} + \underbrace{3 \times 10^2}_{\text{three hundreds}} + \underbrace{7 \times 10^1}_{\text{seven tens}} + \underbrace{4 \times 10^0}_{\text{four ones}}$$

- **Base 2**: Binary numbers

8's column, 4's column, 2's column, 1's column

$1101_2 =$

Base 2

Powers of 2

- $2^0 = 1$
- $2^1 = 2$
- $2^2 = 4$
- $2^3 = 8$
- $2^4 = 16$
- $2^5 = 32$
- $2^6 = 64$
- $2^7 = 128$
- $2^8 = 256$
- $2^9 = 512$
- $2^{10} = 1024$
- $2^{11} = 2048$
- $2^{12} = 4096$
- $2^{13} = 8192$
- $2^{14} = 16384$
- $2^{15} = 32768$

Handy to memorize up to 2^9

Decimal-Binary Interconversion

- Binary to decimal conversion:
  - Convert 10011_2 to decimal
  - $16\times1 + 8\times0 + 4\times0 + 2\times1 + 1\times1 = 19_{10}$

- Decimal to binary conversion:
  - Convert 47_{10} to binary
  - $32\times1 + 16\times0 + 8\times1 + 4\times1 + 2\times1 + 1\times1 = 101111_2$

Decimal to Binary Conversion

- Two methods:
  - **Method 1:** Find the **largest power of 2 that fits**, subtract and repeat.
  - **Method 2:** Repeatedly **divide by 2**, remainder goes in next most significant bit

Decimal to Binary Conversion

Method 1: Find the **largest power of 2 that fits**, subtract and repeat.

53_{10} 32×1

$53-32 = 21$ 16×1

$21-16 = 5$ 4×1

$5-4 = 1$ 1×1 $= 110101_2$

Method 2: Repeatedly **divide by 2**, remainder goes in next most significant bit.

$53_{10} =$ $53/2 = 26$ R1

$26/2 = 13$ R0

$13/2 = 6$ R1

$6/2 = 3$ R0

$3/2 = 1$ R1

$1/2 = 0$ R1 $= 110101_2$

Decimal to Binary Conversion

Another example: Convert 75_{10} to binary

$75_{10} = 64 + 8 + 2 + 1 = 1001011_2$

or

75/2 = 37 R1
37/2 = 18 R1
18/2 = 9R0
9/2= 4R1
4/2= 2R0
2/2= 1R0
1/2= 0R1

Binary Values and Range

- ***N*-digit decimal number**
  - How many values? **10^N**
  - Range? **$[0, 10^N - 1]$**
  - Example: 3-digit decimal number:
    - **$10^3 = 1000$ possible values**
    - **Range: [0, 999]**

- ***N*-bit binary number**
  - How many values? **2^N**
  - Range: **$[0, 2^N - 1]$**
  - Example: 3-digit binary number:
    - **$2^3 = 8$ possible values**
    - **Range: $[0, 7] = [000_2$ to $111_2]$**

Binary to Decimal Generalized

$$A: \{a_{N-1}, a_{N-2}, \ldots a_1, a_0\}$$

$$A = \sum_{i=0}^{N-1} a_i 2^i$$

Example:

$$\begin{aligned} 1101_2 &= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &= 8 + 4 + 0 + 1 \\ &= 13 \end{aligned}$$

Hexadecimal Number

- Base 16
- Shorthand for binary

Hex Digit	Decimal Equivalent	Binary Equivalent
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
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

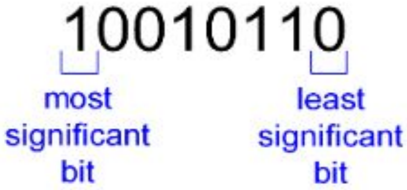
Hexadecimal Number

- Hexadecimal to binary conversion:
  - **Convert 4AF$_{16}$ (also written 0x4AF) to binary**
  - **0100 1010 1111$_2$**

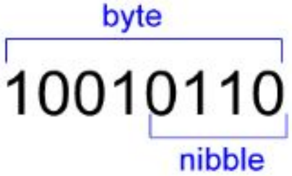
- Hexadecimal to decimal conversion:
  - **Convert 4AF$_{16}$ to decimal**
  - $4 \times 16^2 + A \times 16^1 + F \times 16^0$
  - $4 \times 16^2 + 10 \times 16^1 + 15 \times 16^0 = 1199_{10}$

Bits, Bytes, Nibbles

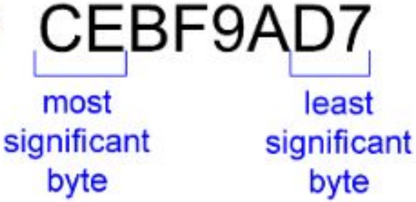
- **Bits**
  - **msb:** most significant bit
  - **lsb:** least significant bit



- **Bytes & Nibbles**



- **Bytes**
  - **MSB:** most significant byte
  - **LSB:** least significant byte
  - Each hex digit represents a nibble (4 bits)



Large Power of Twos

- 2^{10} = 1 kilo (**kibi**) ≈ 10^3 (1024)
- 2^{20} = 1 mega (**mebi**) ≈ 10^6 (1,048,576)
- 2^{30} = 1 giga (**gibi**) ≈ 10^9 (1,073,741,824)
- 2^{40} = 1 tera (**tebi**) ≈ 10^{12}
- 2^{50} = 1 peta (**pebi**) ≈ 10^{15}
- 2^{60} = 1 exa (**exbi**) ≈ 10^{18}

Large Power of Twos

- 2^{10} = 1 kilo ≈ 1000 (1024)

  kibibyte = **1 Ki**

  for example: 1 KiB = 1024 Bytes

  1 Kib = 1024 bits

- 2^{20} = 1 mega ≈ 1 million (1,048,576)

  mebibyte = **1 Mi**

  for example: 1 MiB, 1 Mib (1 megabit)

- 2^{30} = 1 giga ≈ 1 billion (1,073,741,824)

  gibibyte = **1 Gi**

  for example: 1 GiB, 1 Gib

Estimating Powers of 2

- What is the approximate value of 2^{24}?

  $2^4 \times 2^{20} \approx$ **16 million**

- Approximately how many values can a 32-bit variable represent?

  $2^2 \times 2^{30} \approx$ **4 billion**

First factor out the largest 2^{10x}. Then estimate.

Base Conversion from Decimal

$230|_{10}$

Octal

N/8	Q	R
230/8	28	3
28/8	3	1
3/8	0	3

$230|_{10} = 313|_{8}$

Hexadecimal

N/16	Q	R
230/16	14	3
14/16	0	14(**E**)

$230|_{10} = E3|_{16} =$ E3H

Base Conversion to Decimal

$230|_8$	**$1F4|_{16}$**
Octal	**Hexadecimal**
$0 \times 8^0 = 0$	$4 \times 16^0 = 4$
$3 \times 8^1 = 24$	$15 \times 16^1 = 240$
$2 \times 8^2 = 128$	$1 \times 16^2 = 256$
152	**500**

$230|_8 = 152|_{10}$

$1F4|_{16} = 500|_{10}$

Addition

- Decimal

$$\begin{array}{r} \color{blue}{11} \quad \leftarrow \text{carries} \\ 3734 \\ +\ 5168 \\ \hline 8902 \end{array}$$

- Binary

$$\begin{array}{r} \color{blue}{11} \quad \leftarrow \text{carries} \\ 1011 \\ +\ 0011 \\ \hline 1110 \end{array}$$

Binary Addition

- Add the following 4-bit binary numbers

$$
\begin{array}{r}
1 \\
1001 \\
+\ \ 0101 \\
\hline
1110
\end{array}
$$

- Add the following 4-bit binary numbers

$$
\begin{array}{r}
111 \\
1011 \\
+\ \ 0110 \\
\hline
10001
\end{array}
$$

Binary Addition

- Add the following 4-bit binary numbers

Any additional bits on the left are ignored (**overflow!**)

$$\begin{array}{r} 1 \\ 1001 \\ +\ \ 0101 \\ \hline 1110 \end{array}$$

- Generally, addition of two **n-bit** numbers gives an **n-bit** result.

$$\begin{array}{r} 111 \\ 1011 \\ +\ \ 0110 \\ \hline 10001 \end{array}$$

Overflow!

Overflow

- Digital systems operate on a **fixed number of bits**
- **Overflow:** when the result is **too big to fit** in the available number of bits
- See previous example of 11 + 6

$$\begin{array}{r} 111 \\ 1011 \\ +\ \ 0110 \\ \hline 10001 \end{array}$$

Overflow!

Signed Numbers

- A signed binary number consists of both sign and magnitude information.
- **Three formats:**
  - sign-magnitude
  - 1's complement
  - 2's complement.

Sign Magnitude Numbers

- 1 sign bit, *N*-1 magnitude bits
- Sign bit is the most significant (left-most) bit
  - Positive number: sign bit = 0
  - Negative number: sign bit = 1

$$A: \{a_{N-1}, a_{N-2}, \dots a_1, a_0)$$

$$A = (-1)^{a_{N-1}} \sum_{i=0}^{N-2} a_i 2^i$$

- Example, 4-bit sign/mag representations of ± 6:

  +6 = **0110**

  - 6 = **1110**

- Range of an *N*-bit sign/magnitude number:

  $[-(2^{N-1}-1), 2^{N-1}-1]$

Unsigned Binary Numbers

$$A: \{a_{N-1}, a_{N-2}, \dots a_1, a_0\}$$

$$A = \sum_{i=0}^{N-1} a_i 2^i$$

Example:

$$\begin{aligned} 1101_2 &= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &= 8 + 4 + 0 + 1 \\ &= 13 \end{aligned}$$

Sign Magnitude Binary Numbers

$$A: \{a_{N-1}, a_{N-2}, \dots a_1, a_0\}$$

$$A = (-1)^{a_{N-1}} \sum_{i=0}^{N-2} a_i 2^i$$

Example:

$$\begin{aligned} 1101_2 &= (-1)^1 \times (1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0) \\ &= -1 \times (4 + 0 + 1) \\ &= -5 \end{aligned}$$

A big problem with sign magnitude numbers

Problems:

- Addition doesn't work, for example -6 + 6:

$$\begin{array}{r} 1110 \\ +\ 0110 \\ \hline 10100 \end{array}$$ **(wrong!)**

- Two representations of 0 (± 0):

  1000

  0000

1's complement

Flip the bits.

- It makes a positive number negative.
- It makes a negative number positive.

Problem? Zero is not positive or negative. +0 is 0000 and -0 becomes 1111. Two representations of the same number!

2's complement

- **Method:**
  1. Invert the bits
  2. Add 1

- **Example:** Flip the sign of $3_{10} = 0011_2$
  1. 1100
  2. + 1

  $1101 = -3_{10}$

Resolves all the problems seen in signed representation and 1's complement!

2's complement

What is the decimal value of the two's complement number 1001_2?

- We know it's negative (msb = 1)
- Figure out magnitude by flipping the sign (i.e., "taking the two's complement")

1. 0110
2. + 1

$$\underline{0111}_2 = 7_{10}$$

- So, we know it's a negative number with magnitude 7.
- Thus, $1001_2 = -7_{10}$

Taking the two's complement is the second (and **recommended**) way of figuring out the value of a negative two's complement number.

Addition using 2s complement

- Add 6 + (-6) using two's complement numbers

```
 111
   0110
+  1010
-------
  10000
```

- Add -2 + 3 using two's complement numbers

```
 111
   1110
+  0011
-------
  10001
```

Increasing Bit Width

Extend number from *N* to *M* bits (*M* > *N*) :

- Sign-extension
- Zero-extension

Sign Extension

- Sign bit copied to msb's
- Number value is same
- **Example 1:**
  - 4-bit representation of 3 = 0011
  - 8-bit sign-extended value: 00000011
- **Example 2:**
  - 4-bit representation of -5 = 1011
  - 8-bit sign-extended value: 11111011

Zero Extension

- Zeros copied to msb's
- Value changes for negative numbers
- **Example 1:**
  - 4-bit value = $0011 = 3_{10}$
  - 8-bit zero-extended value: $\mathbf{0000}0011 = 3_{10}$
- **Example 2:**
  - 4-bit value = $1011 = -5_{10}$
  - 8-bit zero-extended value: $\mathbf{0000}1011 = 11_{10}$

Number System Comparison

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

For example, 4-bit representation:

Binary Coded Decimal

- The 8421 BCD Code

The 8421 code is a type of **BCD** (binary coded decimal) code. Binary coded decimal means that each decimal digit, 0 through 9, is represented by a binary code of four bits. The designation 8421 indicates the binary weights of the four bits (23, 22, 21, 20).

TABLE 2–5

Decimal/BCD conversion.

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

The binary equivalent of a decimal number is different from its BCD representation!

Binary Coded Decimal

BCD Codes

Decimal	8 4 2 1 BCD	2 4 2 1 BCD	5 4 2 1 BCD
0	0 0 0 0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 0	0 0 1 0
3	0 0 1 1	0 0 1 1	0 0 1 1
4	0 1 0 0	0 1 0 0	0 1 0 0
5	0 1 0 1	1 0 1 1	1 0 0 0
6	0 1 1 0	1 1 0 0	1 0 0 1
7	0 1 1 1	1 1 0 1	1 0 1 0
8	1 0 0 0	1 1 1 0	1 0 1 1
9	1 0 0 1	1 1 1 1	1 1 0 0

For self study, visit this link.

Binary Coded Decimal

EXAMPLE 2–33

Convert each of the following decimal numbers to BCD:

(a) 35 **(b)** 98 **(c)** 170 **(d)** 2469

EXAMPLE 2–34

Convert each of the following BCD codes to decimal:

(a) 10000110 **(b)** 001101010001 **(c)** 1001010001110000

Solution

(a) $\underbrace{1000}_{\downarrow\ 8}\underbrace{0110}_{\downarrow\ 6}$ **(b)** $\underbrace{0011}_{\downarrow\ 3}\underbrace{0101}_{\downarrow\ 5}\underbrace{0001}_{\downarrow\ 1}$ **(c)** $\underbrace{1001}_{\downarrow\ 9}\underbrace{0100}_{\downarrow\ 4}\underbrace{0111}_{\downarrow\ 7}\underbrace{0000}_{\downarrow\ 0}$

Related Problem

Convert the BCD code 10000010001001110110 to decimal.

Gray Code

The **code** main characteristic is that between one combination of digits (0, 1) and the next one, whether upstream or downstream, there is only a difference of one digit. So it is also called Progressive Code. This progression also occurs between the last and the first combination. That is why it is also called Cyclic code. (see the table)

It is a non-weighted code.

Binary Code to Gray Code conversion method

To convert Binary into Gray, use the following method:

1. We add the binary number to another like it. The second number need to be moved one digit to the right. See the image.
2. We do a binary addition digit by digit, and we discard the carry.
3. We remove the last digit on the right side of the result on step 2 (we remove the zero, which is in red color). The resulting code is the GRAY code.

Gray Code

Gray code to Binary Conversion method

To convert a GRAY to a Binary, we use the following method:

1. The first digit of the Gray code will be the same in the binary number.
2. a) If the second digit is “0”, the second digit of the binary number is equal to the first digit.
   b) If the second digit is “1”, the second digit of the binary number is the inverse of the first digit.
3. If the third digit is “0”, the third digit of the binary number equals the second digit of the binary number.
   a) If the third digit is “1”, the third digit of the binary number is the inverse of the second binary number digit ….. And so on until finished.

Binary-Gray Code Interconversion

TABLE 44.8 Gray Code

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