CSE 260
Digital Logic Design

Dewan Ziaul Karim (DZK)

Teaching Staff:

Name: Dewan Ziaul Karim (DZK)
Room: UB80902
Email : **ziaul.karim@bracu.ac.bd**
Subject: [CSE260] [Sec – 09, 10, 14 & 15]

Teaching Assistant (to be appointed) will be there for your support

Consultation Hours:

\\tsr\spring2020\CSE\DZK\CSE260\Lecture Slides

Appointment by email !

Objective

- Distinguish between analog and digital system

- Understand the advantage and limitation of digital system

- Understand the meaning of digital logic

Analog vs. Digital

- Analog data can vary over a continuous range of values. Example: speedometer
- Digital quantities can take on only discrete values (0 and 1, high and low). Example: Digital Computer, Decimal Digits, Alphabets

Digital System

- A digital system is a combination of devices designed to manipulate physical quantities or information that are represented in digital form.

- “A discreet information processing system”
- Signals: Discreet information

Advantage of digital system

- Greater accuracy or precision
- Easier to design (generality)
- Easier information storage
- Programmability (instructions)
- Speed
- Economical

Limitation of digital technology

- The real world is mainly analog

Overcome the limitation

- Convert the real world analog input data into digital one
- Process this digital data
- Then again convert into analog form

Digital logic

- Design logic is a term used to denote the design and analysis of digital system

- Digital logic is concerned with the interconnection among digital components and modules

- Digital logic design is engineering and engineering means problem solving

Number systems and codes

Digital Systems are built from circuits that process binary digits. BUT very few real-life problems are based on binary numbers.

SO a digital system designer must establish some correspondence between the binary digits processed by digital circuits and real-life numbers, events and conditions.

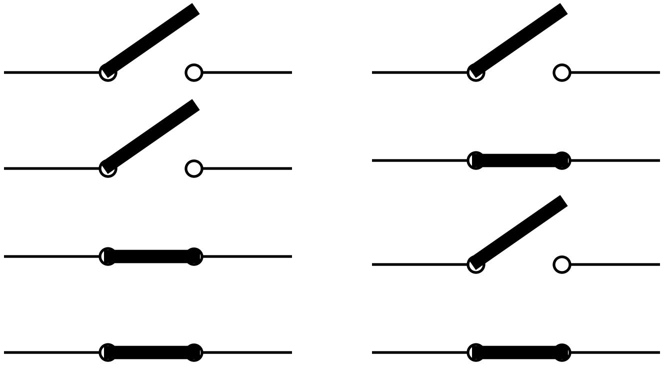
Information representation

- Human decisions tends to be binary i.e. Yes or No
- Elementary storage units inside computer are *electronic switches*. Each switch holds one of two states: *on* (1) or *off* (0).

- We use a *bit* (*bi*nary digi*t*), 0 or 1, to represent the state.

Information representation

- Storage units can be grouped together to cater for larger range of numbers. Example: 2 switches to represent 4 values.



0 (00)

1 (01)

2 (10)

3 (11)

Information representation

- In general, *N* bits can represent 2^N different values.
- For *M* values, $\lceil \log_2 M \rceil$ bits are needed.

1 bit → represents up to 2 values (0 or 1)
2 bits → rep. up to 4 values (00, 01, 10 or 11)
3 bits → rep. up to 8 values (000, 001, 010. ..., 110, 111)
4 bits → rep. up to 16 values (0000, 0001, 0010, ..., 1111)

32 values → requires 5 bits
64 values → requires 6 bits
1024 values → requires 10 bits
40 values → requires 6 bits
100 values → requires 7 bits

Positional Notations

- Decimal number system, symbols = { 0, 1, 2, 3, …, 9 }
- Position is important
- Example:$(7594)_{10} = (7 \times 10^3) + (5 \times 10^2) + (9 \times 10^1) + (4 \times 10^0)$
- In general, $(a_n a_{n-1} \dots a_0)_{10} = (a_n \times 10^n) + (a_{n-1} \times 10^{n-1}) + \dots + (a_0 \times 10^0)$
- $(2.75)_{10} = (2 \times 10^0) + (7 \times 10^{-1}) + (5 \times 10^{-2})$
- In general, $(a_n a_{n-1} \dots a_0 . f_1 f_2 \dots f_m)_{10} = (a_n \times 10^n) + (a_{n-1} \times 10^{n-1}) + \dots + (a_0 \times 10^0) + (f_1 \times 10^{-1}) + (f_2 \times 10^{-2}) + \dots + (f_m \times 10^{-m})$

Other Number Systems

- Binary (base 2): weights in powers-of-2. Binary digits (bits): ***0,1.***
- Octal (base 8): weights in powers-of-8. Octal digits: ***0,1,2,3,4,5,6,7***
- Hexadecimal (base 16): weights in powers-of-16. Hexadecimal digits: ***0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F***

Note: when base is r, coefficient values range from 0 to r-1.

Other Number System

- Binary (base 2): weights in powers-of-2. Binary digits (bits): ***0,1.***
- Octal (base 8): weights in powers-of-8. Octal digits: ***0,1,2,3,4,5,6,7***
- Hexadecimal (base 16): weights in powers-of-16. Hexadecimal digits: ***0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F***

Binary	Octal	Decimal	Hexadecimal
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F

Base-R to Decimal Conversion

***Formula= $\sum$ digit * source_base ^{position}

- $(1101.101)_2 = 1\times2^3 + 1\times2^2 + 1\times2^0 + 1\times2^{-1} + 1\times2^{-3}$
  $= 8 + 4 + 1 + 0.5 + 0.125$
  $= (13.625)_{10}$

- $(572.6)_8 = 5\times8^2 + 7\times8^1 + 2\times8^0 + 6\times8^{-1} = 320 + 56 + 2 + 0.75 = (378.75)_{10}$

- $(2A.8)_{16} = 2\times16^1 + 10\times16^0 + 8\times16^{-1} = 32 + 10 + 0.5 = (42.5)_{10}$

- $(341.24)_5 = 3\times5^2 + 4\times5^1 + 1\times5^0 + 2\times5^{-1} + 4\times5^{-2}$
  $= 75 + 20 + 1 + 0.4 + 0.16 = (96.56)_{10}$

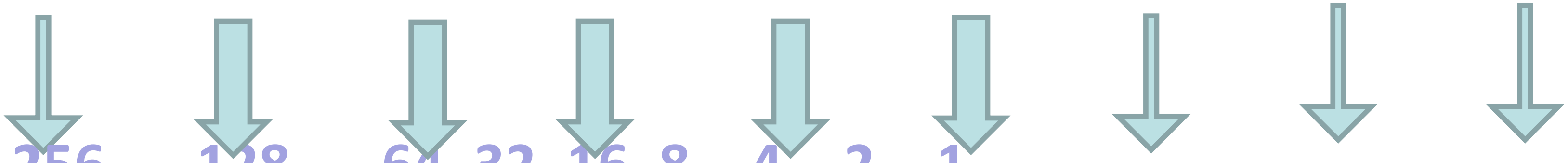
Decimal-to-Binary Conversion

Works for binary

- Method 1: *Sum-of-Weights Method*

Note: Remembering the below value helps

$2^8, 2^7, 2^6, 2^5, 2^4, 2^3, 2^2, 2^1, 2^0$



256, 128, 64, 32, 16, 8, 4, 2, 1

Method 2:

Works for all base

- *Repeated Division-by-2 Method* (for whole numbers)
- *Repeated Multiplication-by-2 Method* (for fractions)

Sum-of-Weights Method

- Determine the set of binary weights whose sum is equal to the decimal number.

$(9)_{10} = 8 + 1 = 2^3 + 2^0 = (1001)_2$

$(18)_{10} = 16 + 2 = 2^4 + 2^1 = (10010)_2$

$(58)_{10} = 32 + 16 + 8 + 2 = 2^5 + 2^4 + 2^3 + 2^1$

$= (111010)_2$

$(0.625)_{10} = 0.5 + 0.125 = 2^{-1} + 2^{-3}$

$= (0.101)_2$

Conversion between Decimal and other Bases

- Decimal to base-R
  - whole numbers: repeated division-by-R
  - fractions: repeated multiplication-by-R

Repeated Division-by-2 Method

- To convert a whole number to binary, use successive division by 2 until the quotient is 0. The remainders form the answer, with the first remainder as the *least significant bit (LSB)* and the last as the *most significant bit (MSB)*.

2	43			
2	21	rem	1	← LSB
2	10	rem	1	
2	5	rem	0	
2	2	rem	1	
2	1	rem	0	
	0	rem	1	← MSB

$(43)_{10} = (101011)_2$

Repeated Multiplication-by-2 Method

- To convert decimal fractions to binary, repeated multiplication by 2 is used, until the fractional product is 0 (or until the desired number of decimal places). The carried digits, or *carries*, produce the answer, with the first carry as the MSB, and the last as the LSB.

$(0.3125)_{10} = (.0101)_2$

	Carry	
$0.3125 \times 2 = 0.625$	0	←MSB
$0.625 \times 2 = 1.25$	1	
$0.25 \times 2 = 0.50$	0	
$0.5 \times 2 = 1.00$	1	←LSB

Note: difference between conversion of integer and fraction

Integer

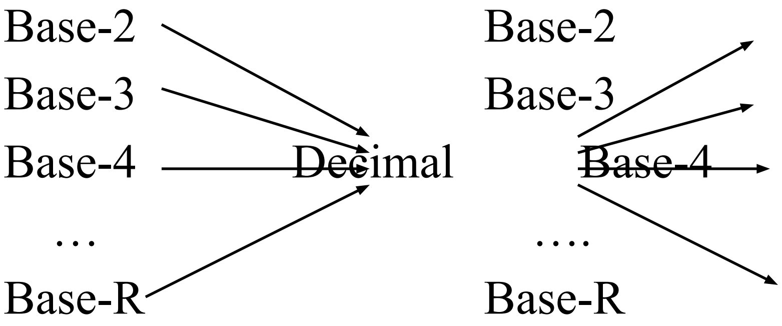
- Division by base of target no. system
- Remainders are accumulated
- By division we obtain LSB to MSB

Fraction

- Multiplication by base of target no. system
- Integers are accumulated
- By multiplication we obtain MSB to LSB

Conversion between Bases

- In general, conversion between bases can be done via decimal:



Binary-Octal/Hexadecimal Conversion

- Binary → Octal: Partition in groups of 3

  $(10\ 111\ 011\ 001\ .\ 101\ 110)_2 = (2731.56)_8$

- Octal → Binary: reverse

  $(2731.56)_8 = (10\ 111\ 011\ 001\ .\ 101\ 110)_2$

- Binary → Hexadecimal: Partition in groups of 4

  $(101\ 1101\ 1001\ .\ 1011\ 1000)_2 = (5D9.B8)_{16}$

- Hexadecimal → Binary: reverse

  $(5D9.B8)_{16} = (101\ 1101\ 1001\ .\ 1011\ 1000)_2$

Binary-Octal/Hexadecimal Conversion

- $(1100100.10)_2 = (144.4)_8$
  $= (64.8)_{16}$

Exercise:

(1)Try converting this to

$(101110001101011.111100000110)_2$

a)octal b) hexadecimal

(2) Try converting these to binary

a) $(673.124)_8$

b) $(306.D)_{16}$

Answers:

(1) a) $(26153.7406)_8$

(1) b) $(2C6B.F06)_{16}$

(2) a) $(110\ 111\ 011\ .\ 001\ 010\ 100)_2$

(2) b) $(0011\ 0000\ 0110\ .\ 1101)_2$

Binary operations: Addition: Addition Rules w/Carries

For 2 bit

- 0 + 0 = 0 0 (0 with a 0 carry)
- 0 + 1 = 0 1 (1 with a 0 carry)
- 1 + 0 = 0 1 (1 with a 0 carry)
- 1 + 1 = 1 0 (0 with a 1 carry)

For 3 bit

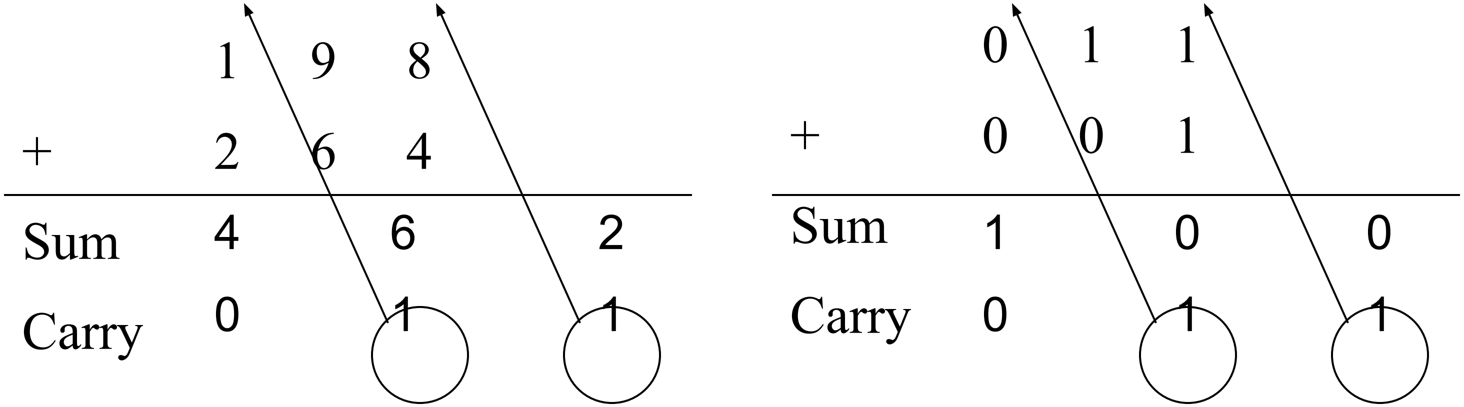
- 0+0+0 = 0 0 (0 WITH 0 CARRY)
- 0+0+1 = 0 1 (1 WITH 0 CARRY)
- 0+1+1 = 1 0 (0 WITH 1 CARRY)
- 1+1+1 = 1 1 (1 WITH 1 CARRY)

Adding Binary Numbers

$$\begin{array}{rcr} & & 0\;1\;1\;1\;0\;0\;0 \\ 28 & \longrightarrow & 00011100 \\ +\;\;43 & \longrightarrow & +\;\;00101011 \\ \hline 71 & & 01000111 \end{array}$$

Binary operations: Addition

- From right to left, we add each pair of digits
- We write the sum, and add the carry to the next column on the left



	1	9	8
+	2	6	4
Sum	4	6	2
Carry	0	1	1

	0	1	1
+	0	0	1
Sum	1	0	0
Carry	0	1	1

Exercise

3(a) Add $(101101)_2$ with $(100111)_2$

Solution

- 3(a) $(1010100)_2$

Working:

Augend:	101101
Addend:	+100111
Sum:	1010100

Addition of base-r

- Example:

  $(34)_5 + (41)_5 + (24)_5$

1+3+4+2= 10

10%5=0

10/5=2 (carry)

$$
\begin{array}{r}
21 \\
(34)_5 \\
(41)_5 \\
+\ (24)_5 \\
\hline
(2\ 0\ 4)_5
\end{array}
$$

4+1+4= 9

9%5=4

9/5=1 (carry)

Exercise

- Try:
  2 (a) $(FF)_{16}+(F1)_{16}$
  2 (b) $(66)_7 + (55)_7$

Solution

- 2(a) $(1F0)_{16}$
- 2(b) $(154)_{7}$

Binary Multiplication

- The multiplication of two binary numbers can be carried out in the same manner as the decimal multiplication.
- Unlike decimal multiplication, only two values are generated as the outcome of multiplying the multiplication bit by 0 or 1 in the binary multiplication. These values are either 0 or 1.
- The binary multiplication can also be considered as repeated binary addition. Therefore, the binary multiplication is performed in conjunction with the binary addition operation.

Binary Multiplication

A	B	A×B
0	0	0
0	1	0
1	0	0
1	1	1

Example : Perform the binary multiplication of the decimal numbers 12 and 10.

The equivalent binary representation of the decimal number 12 is 1100.
The equivalent binary representation of the decimal number 10 is 1010.

$$\begin{array}{ccccccc}
 & & & 1 & 1 & 0 & 0 \\
 & & \times & 1 & 0 & 1 & 0 \\
\cline{3-7}
 & & & 0 & 0 & 0 & 0 \\
 & & 1 & 1 & 0 & 0 & \\
 & 0 & 0 & 0 & 0 & & \\
1 & 1 & 0 & 0 & & & \\
\hline
1 & 1 & 1 & 1 & 0 & 0 & 0
\end{array}$$

Exercise

6 (a) Multiply 1011 with 101

Solution

Multiplicand	1011
Multiplier	× 101
Partial Products	1011
	0000 -
	1011 - -
Product	110111

Multiplication with base-r

```
•   2A3C
     xB7
 ------------
   127A4
 1D094
 ------------
 1E30E4
```

Working

7*C = 7*12 = 84 =(write 4, carry 5)
7*3+5 = 26 = 1A (write A, carry 1)
7*A+1 = 71 = 0x47 (write 7, carry 4)
7*2+4 = 18 = 0x12
This completes the 7*2A3C = 127A4 partial product.

B*C = 11*12 = 132 = (write 4, carry 8)
B*3+8 = 11*3+8 = 41 = (write 9, carry 2)
B*A+2 = 11*10+2 = 112 = (write 0, carry 7)
B*2+7 = 11*2+7 = 29 = 1D
This completes the B[0]*2A3C = 1D094[0] partial product, where I'm noting the [0] digits to remind us this is in the 16s column.

Adding the partial products: 127A4 + 1D0940
4+0 = 4
A+4 = E
7+9 = 16 = 0x10 (write 0, carry 1)
2+0+1 = 3
1+D = E
1 = 1

Exercise

7 (a) Multiply $(34)_5$ with $(42)_5$
7 (b) Multiply $(25)_9$ with $(36)_9$

Solution

7 (a) $(3133)_5$
7 (b) $(1033)_9$

Binary operations: Subtraction Rules w/Carries

For 2 bit

- 0 - 0 = 0 0 (0 with a 0 carry)
- 1 - 1 = 0 0 (0 with a 0 carry)
- 1 - 0 = 0 1 (1 with a 0 carry)
- 0 - 1 = ? (???)

Subtracting Binary Numbers: approach 1

$$\begin{array}{rcr} 2 & \longrightarrow & \overset{0}{\cancel{1}}\overset{2}{0} \\ -\;1 & \longrightarrow & -\;0\cancel{1} \\ \hline 1 & & 01 \end{array}$$

Subtracting Binary Numbers: approach 2

$$\begin{array}{rcr}
 & & \cancel{1}\ 1 \\
 & & 0\ 1\ \cancel{1} \\
4 & \longrightarrow & \cancel{1}00 \\
-\ 1 & \longrightarrow & -\ 00\cancel{1} \\
\hline
3 & & 011
\end{array}$$

Exercise

5(b) Subtract $(100111)_2$ from $(101101)_2$

Solution

5 (b) $(000110)_2$

Working:

5(b)Subtraction

Minuend:	101101
Subtrahend:	−100111
Difference:	000110

Subtraction of Base-r

$$\begin{array}{r} {}^{3}\quad{}^{16}\qquad \\ (\cancel{4}\ A\ 6)_{16} \\ -(1\ B\ 3)_{16} \\ \hline (2\ F\ 3)_{16} \end{array}$$

$$\begin{array}{r} 4\ 6\quad \\ (\cancel{5}4)_{6} \\ -\ (35)_{6} \\ \hline (15)_{6} \end{array}$$

Exercise

4 (a) $(71)_8$-$(56)_8$

4(b) $(21)_3$-$(12)_3$

Solution

- 4 a) $(13)_8$
- 4 b) $(2)_3$

Number Conversion Website: https://coderstoolbox.net/number/