EIE130

Digital Circuits

Course Outline

Information of Instructor

- Name : **Dr. KinTak U**
- Tel: **858-8897-2249**
- E-mail: **ktu@must.edu.mo**
- Office: **Room C403**
- Office Hours: **Tuesday : 16:30~19:30**
  Wednesday: 15:30~18:30
  Thursday : 14:30~15:30;18:30~19:30
  Friday : 14:30~15:30;18:30~19:30

Required Textbook & Reference books

Textbook

1. **M. Morris Mano and Charles R. Kime. Logic and Computer Design Fundamentals, 4th/5th/6th Edition, Pearson, 2008. ISBN: 9780132067119.**
   http://www.writphotec.com/mano4/

Reference

1. **M. Morris. Mano and M. D. Ciletti. Digital Design: With an Introduction to the Verilog HDL, Fifth Edition, Pearson, 2012. ISBN: 9780273764526**
2. **J. F. Wakerly. Digitial Design: Principles and Practices, Fourth Edition, Pearson, 2006.**
   ISBN: 9789810677916

Objectives

1. **EIE130** is designed to provide an introduction to digital systems for undergraduates.
2. The course will cover **digital systems** and information, **combinational logic circuits**, **combinational logic design**, **arithmetic functions**, **sequential circuits.**
3. It also aims to acquaint the student with the basic principles of digital systems **with some exercise and assignments**.

Grading Scheme

- Exercises: **6%** (in each class)
- Written assignments: **20%** (totally 12 assignments)
- **Midterm Exam 24%**
- **Final Exam**: **50%**

Lecture Schedule

Index	Topic	Hours	Teaching Method
1	Digital System and Information (Number System and its operations)	5	Lecture
2	Combinational Logic Circuits Part I: Gate Circuits and Boolean Equations	5	Lecture
3	Combinational Logic CircuitsPart II: Circuit Optimization	5	Lecture
4	Combinational Logic CircuitsPart III: Additional Gates and Circuits	5	Lecture
5	Combinational Logic DesignPart I: Design Procedure. Part II: Combinational Logic	5	Lecture
6	Review for midterm exam.	2	Lecture
7	Mid-term Exam.	3	
8	Arithmetic Functions	5	Lecture
9	Sequential CircuitsPart I: Storage Elements and Analysis Part II: Sequential Circuit Design	5	Lecture
10	Review for final exam.	5	Lecture

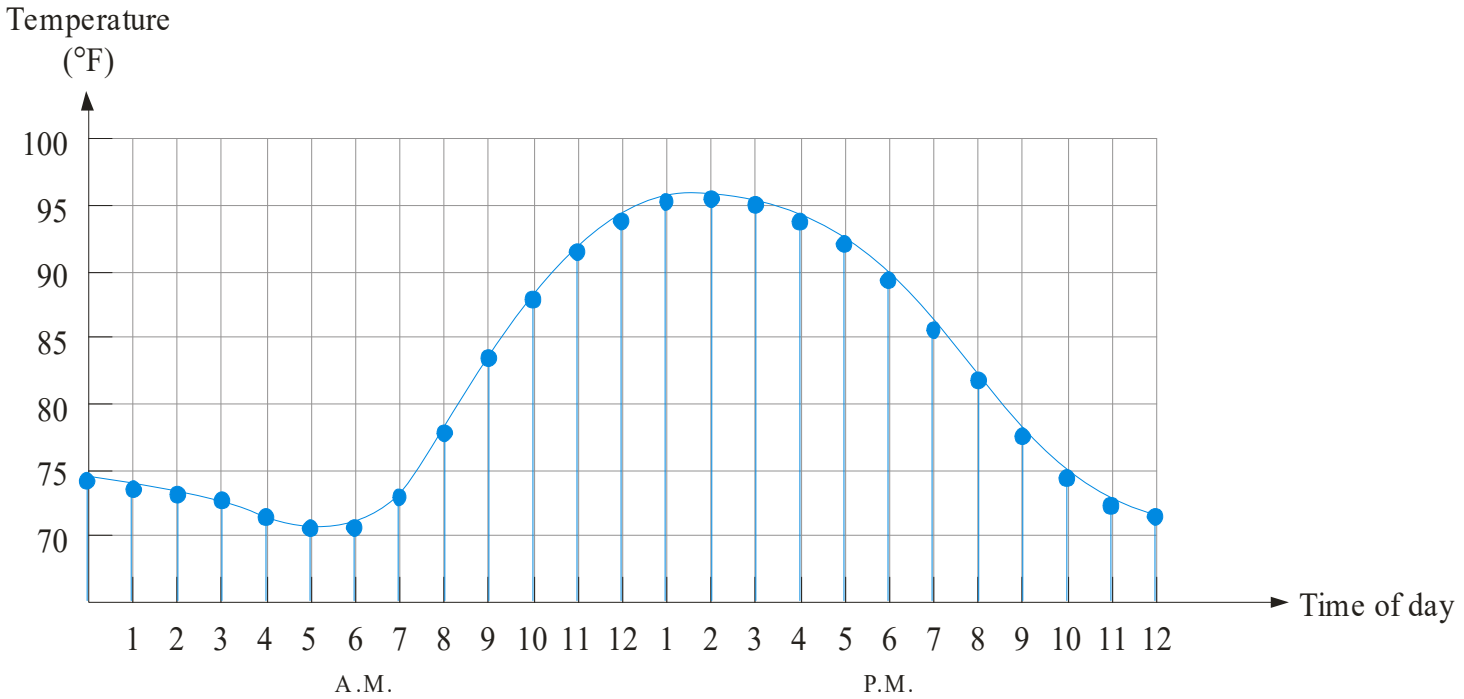
EIE130 Digital Circuits

Chapter 1

Digital Systems and Information

Analog Quantities

Most natural quantities that we see are **analog** and vary continuously. Analog systems can generally handle **higher power** than digital systems.



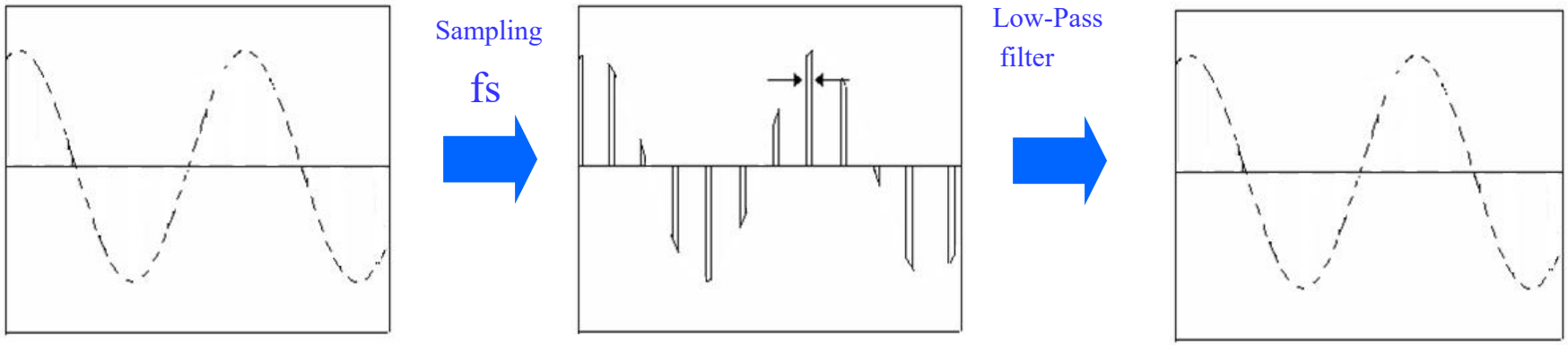
Digital systems can **process, store, and transmit data** ***more efficiently*** but can only **assign discrete values** to each point.

Why Discrete Signal can represent Analog Signal

To store analog signal will spend many space.

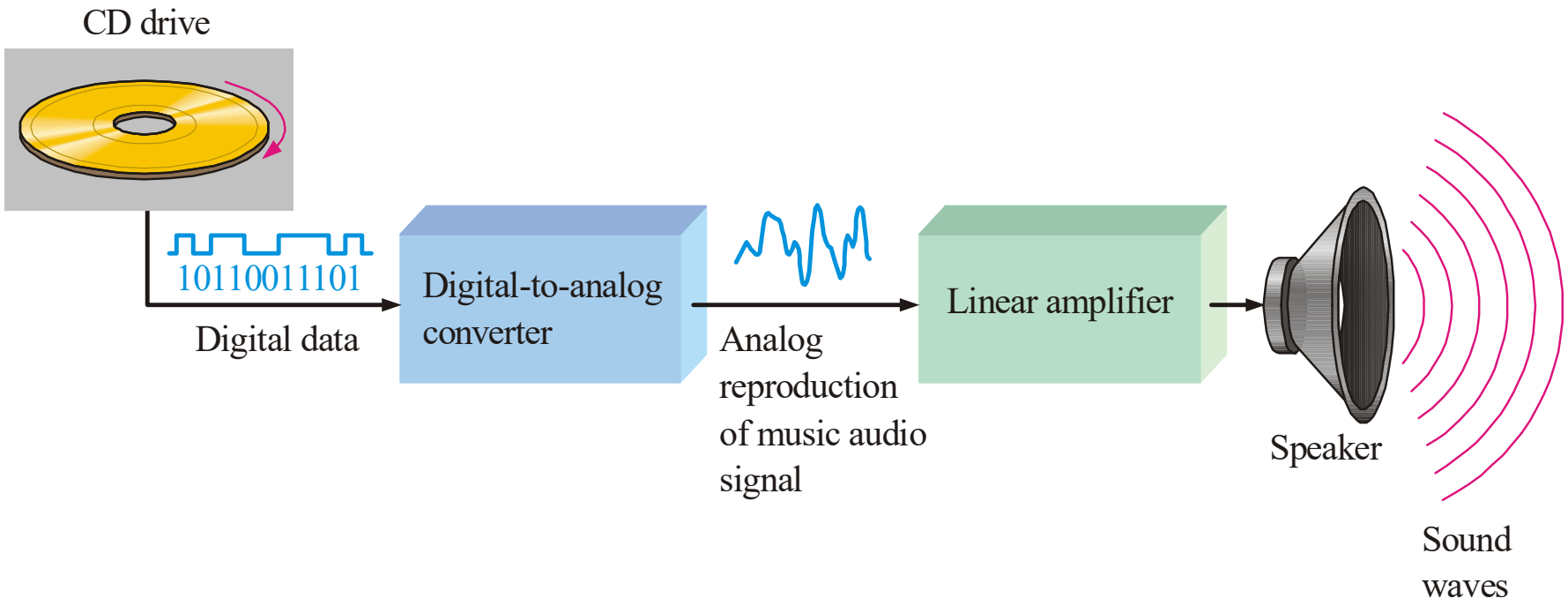
Discrete Signal is obtained by sampling the analog signal at a sampling frequency fs

By the Sampling Theorem, if the fs is greater than two times of the maximum frequency (fmax) of the analog signal, then the obtained discrete sampled signal can fully be recovered to its original analog signal by a low-pass filter.



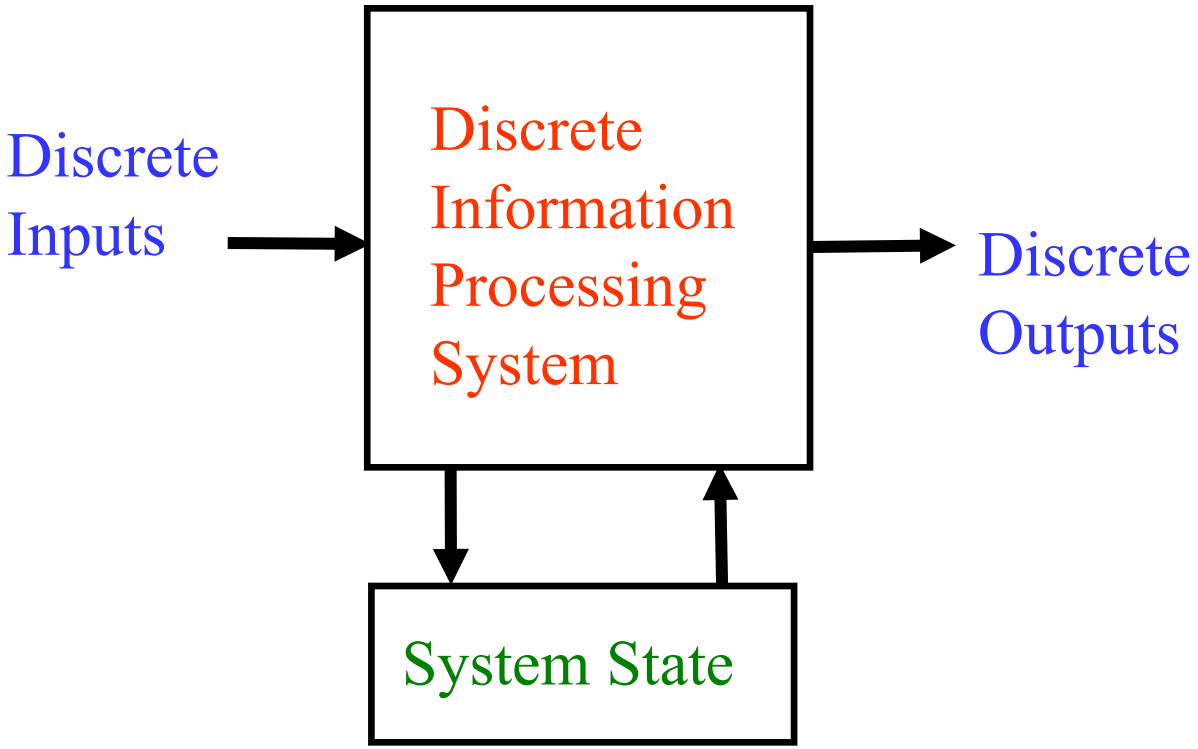
Analog and Digital Systems

Many systems use a mix of analog and digital electronics to take advantage of each technology. A typical CD player accepts digital data from the CD drive and converts it to an analog signal for amplification.



Digital System

- **Takes a set of discrete information inputs and discrete internal information (system state) and generates a set of discrete information outputs.**



Types of Digital Systems

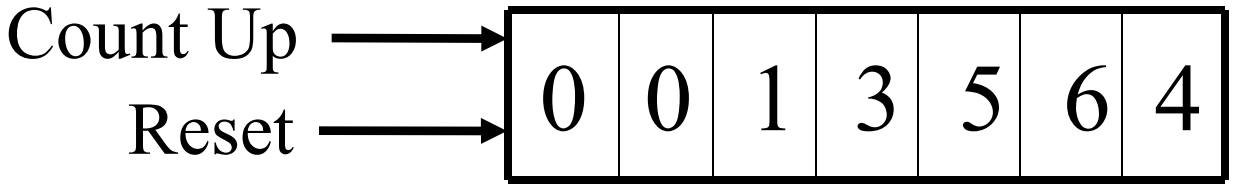
- **No state present**
  - **Combinational Logic System**
  - **Output = Function(Input)**
- **State present**
  - **State updated at discrete times (Clock)**

    => Synchronous Sequential System
  - **State updated at any time**

    =>Asynchronous Sequential System
  - **State = Function (State, Input)**
  - **Output = Function (State) or Function (State, Input)**

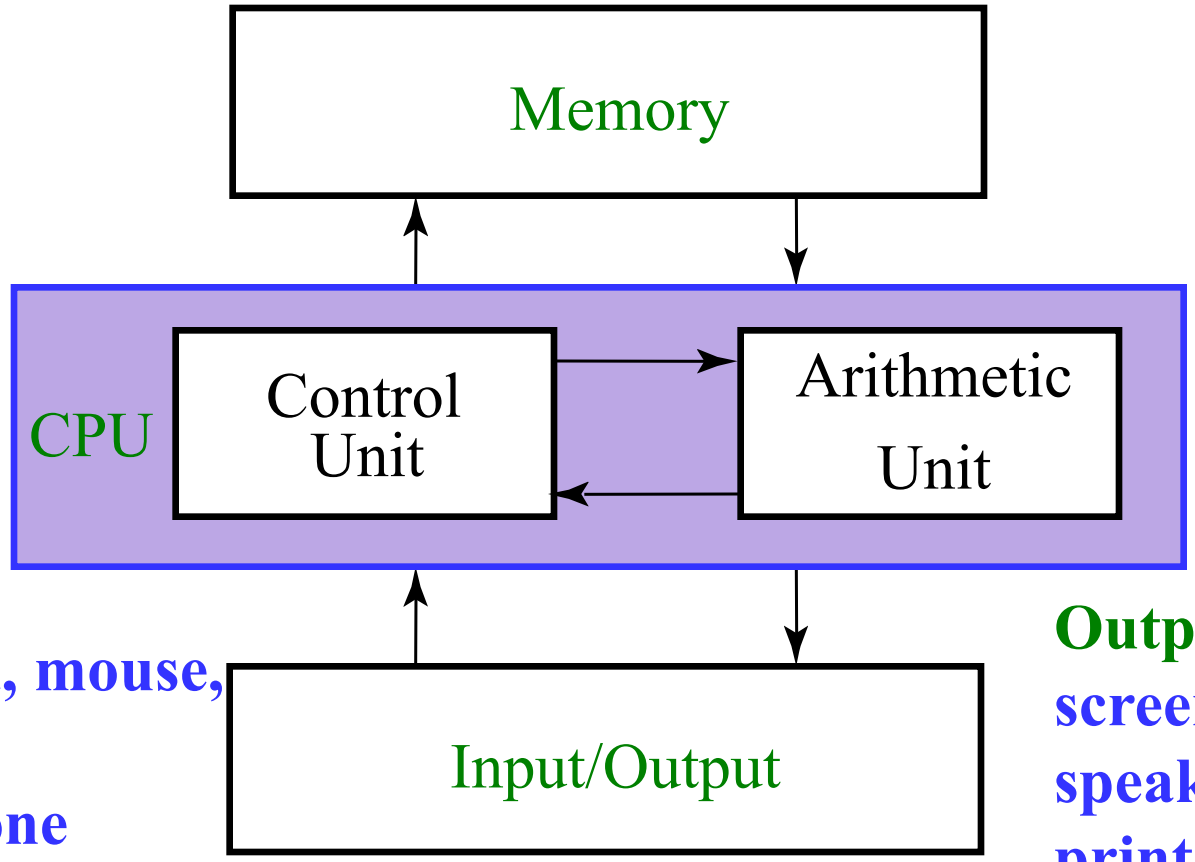
Digital System Example:

A Digital Counter (e. g., odometer):



Inputs: Count Up, Reset
Outputs: Visual Display
State: "Value" of stored digits

Digital Computer Example

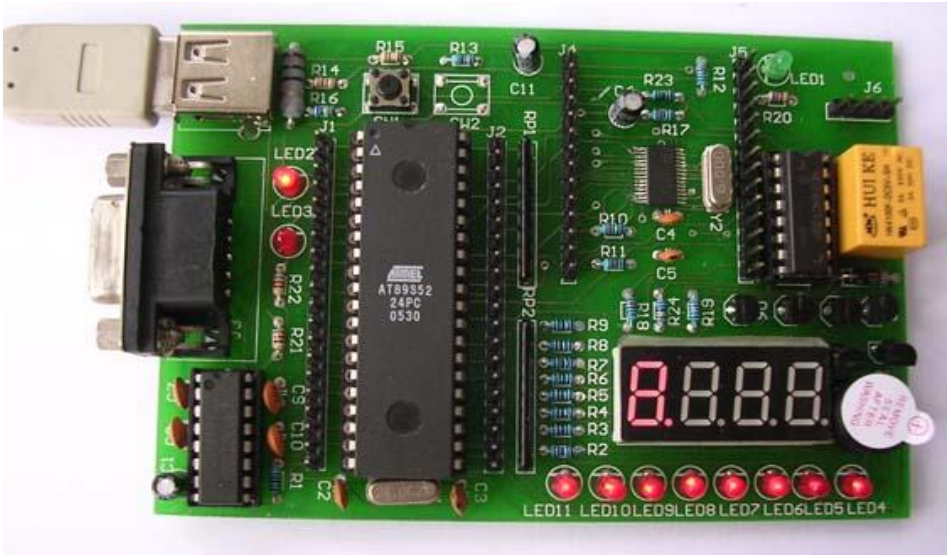


Inputs:
keyboard, mouse, wireless, microphone

Outputs: LCD screen, wireless, speakers, printer

Embedded Computers

- Computers as integral parts of other products
- Examples of embedded computers
  - Micro Controller Unit (MCU)
  - Single Chip Microcomputers (SCM)
  - Digital Signal Processors (DSP)



Embedded Systems Applications

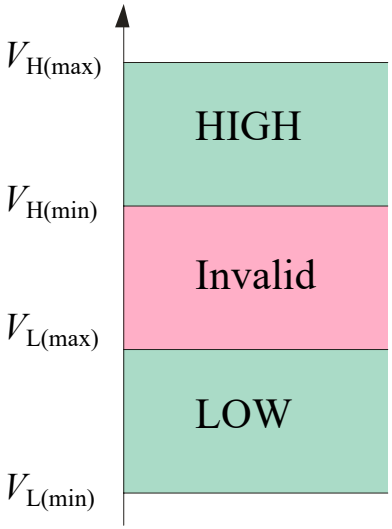
Examples of Embedded Systems Applications

- Cell phones (J2ME, Android, Objective-C)
- Automobiles (C, Assembly)
- Video games
- Copiers
- Dishwashers
- Flat Panel TVs
- Global Positioning Systems (GPS)

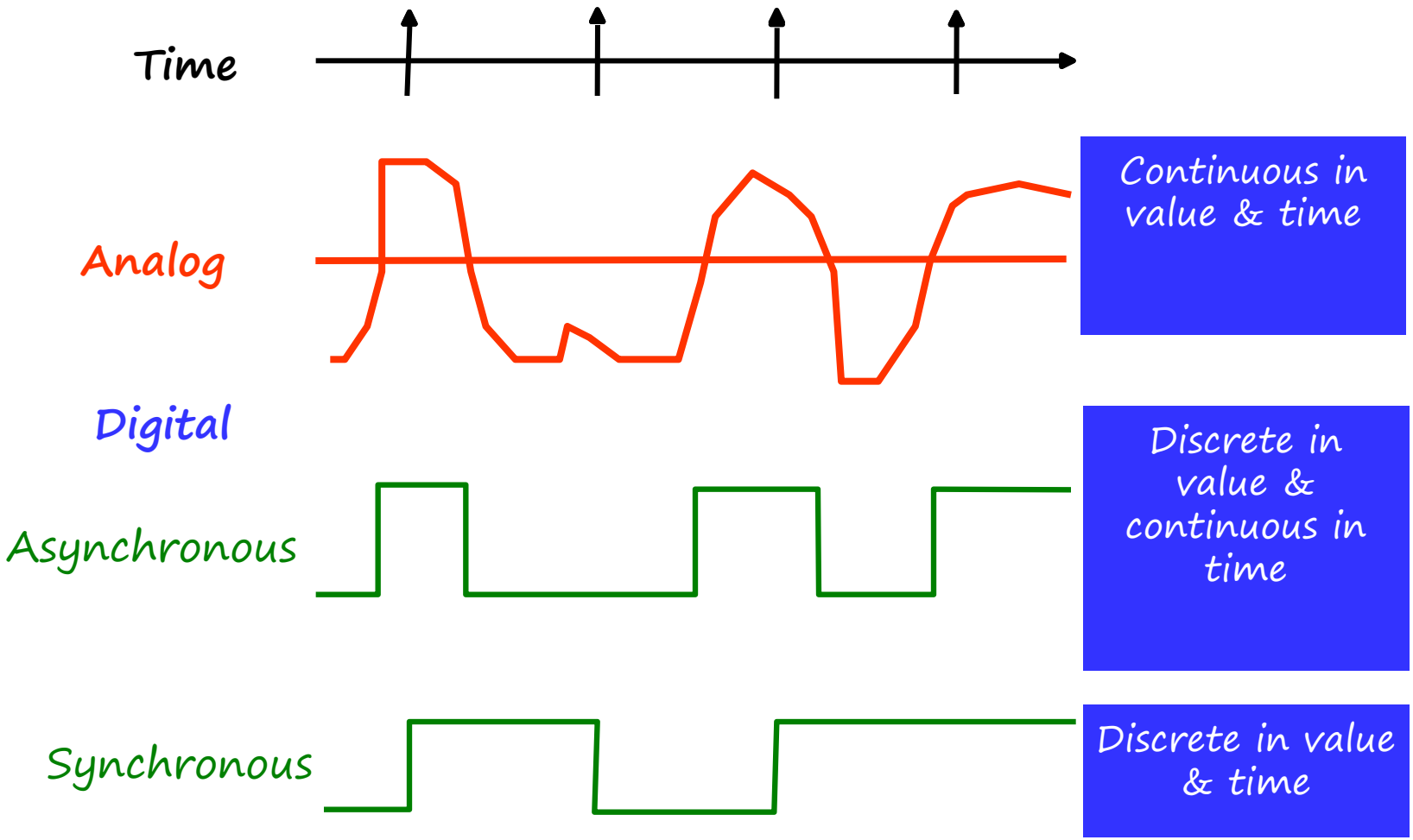
Information Representation - Signals

- **Information variables represented by physical quantities.**
- **For digital systems, the variables take on discrete values.**
- **Two level, or binary values are the most prevalent values in digital systems.**
- **Binary values are represented abstractly by:**
  - **digits 0 and 1 (bit = binary digit),**

    1 byte = 8 bits
  - **words (symbols) False (F) and True (T)**
  - **words (symbols) Low (L) and High (H)**
  - **and words Off and On.**

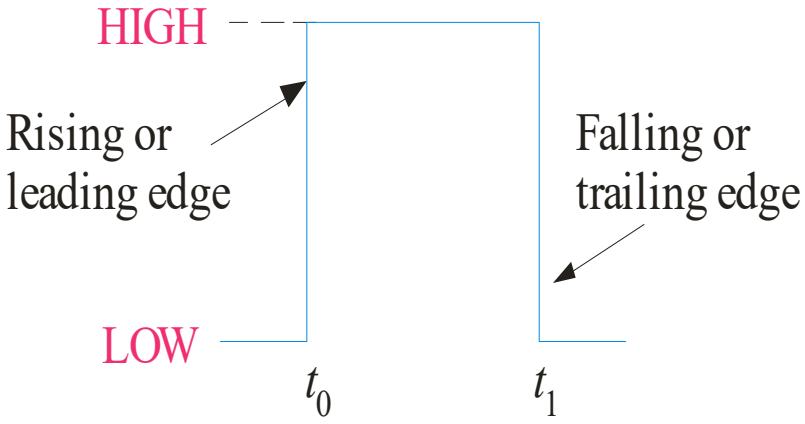


Signal Examples Over Time

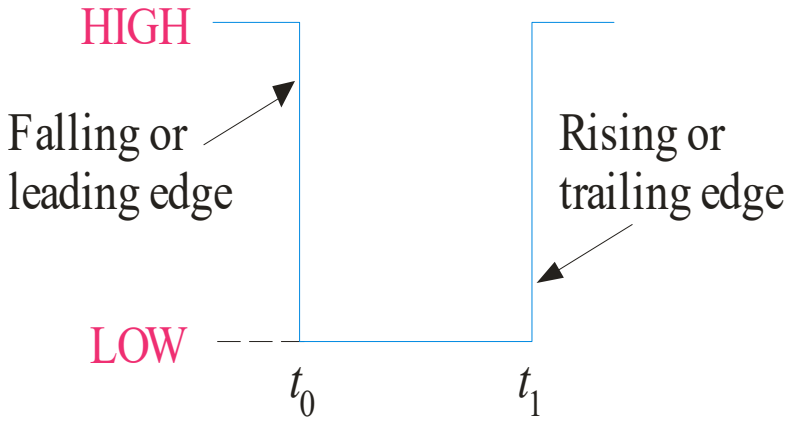


Digital Waveforms

Digital waveforms change between the LOW and HIGH levels. A positive going pulse is one that goes from a normally LOW logic level to a HIGH level and then back again. Digital waveforms are made up of a series of pulses.



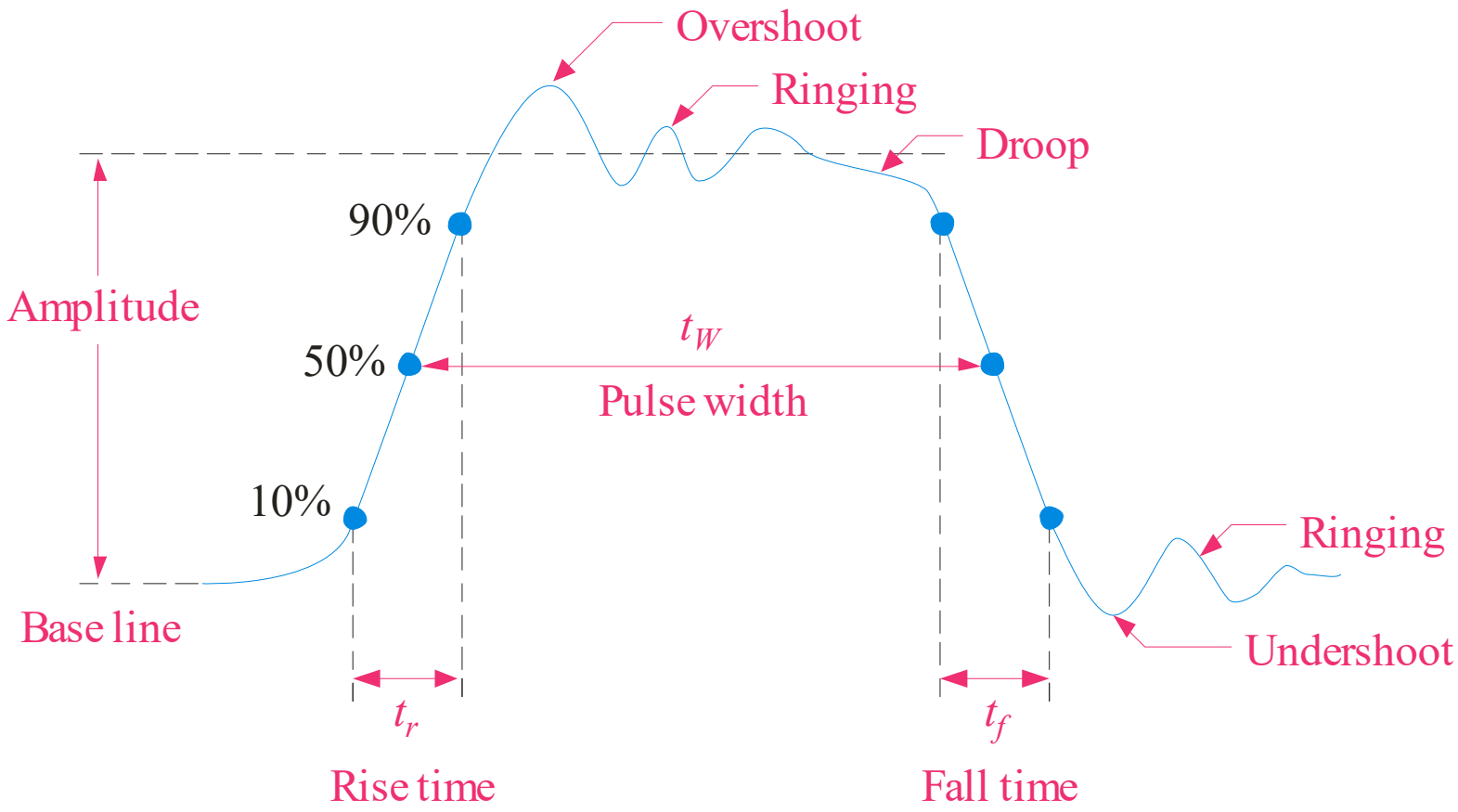
(a) Positive–going pulse



(b) Negative–going pulse

Pulse Definitions

Actual pulses are not ideal but are described by the rise time, fall time, amplitude, and other characteristics.



Periodic Pulse Waveforms

Periodic pulse waveforms are composed of pulses that repeats in a fixed interval called the **period (second)**. The **frequency** is the rate it repeats and is measured in hertz (Hz).

$$f = \frac{1}{T} \qquad T = \frac{1}{f}$$

The **clock** is a basic timing signal that is an example of a periodic wave.

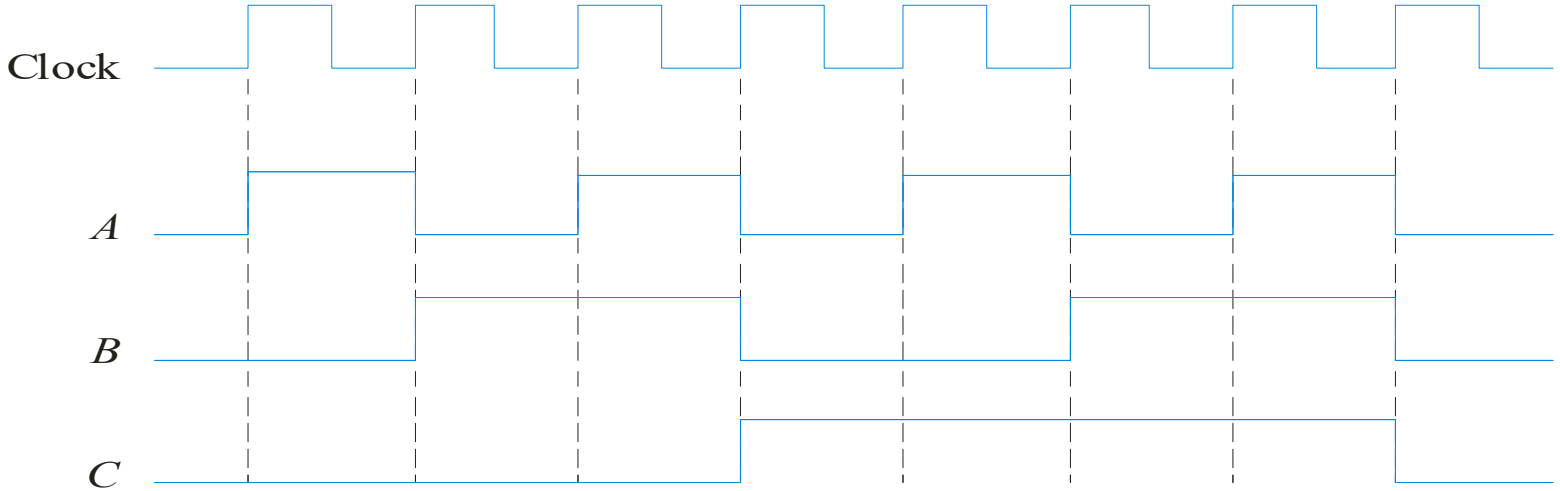
Example What is the period of a repetitive wave if $f = 3.2$ GHz?

Solution $T = \frac{1}{f} = \frac{1}{3.2\,\text{GHz}} =$ 313 ps

T	G	M	K	m	μ	n	p
10^{12}	10^{9}	10^{6}	10^{3}	10^{-3}	10^{-6}	10^{-9}	10^{-12}

Timing Diagrams

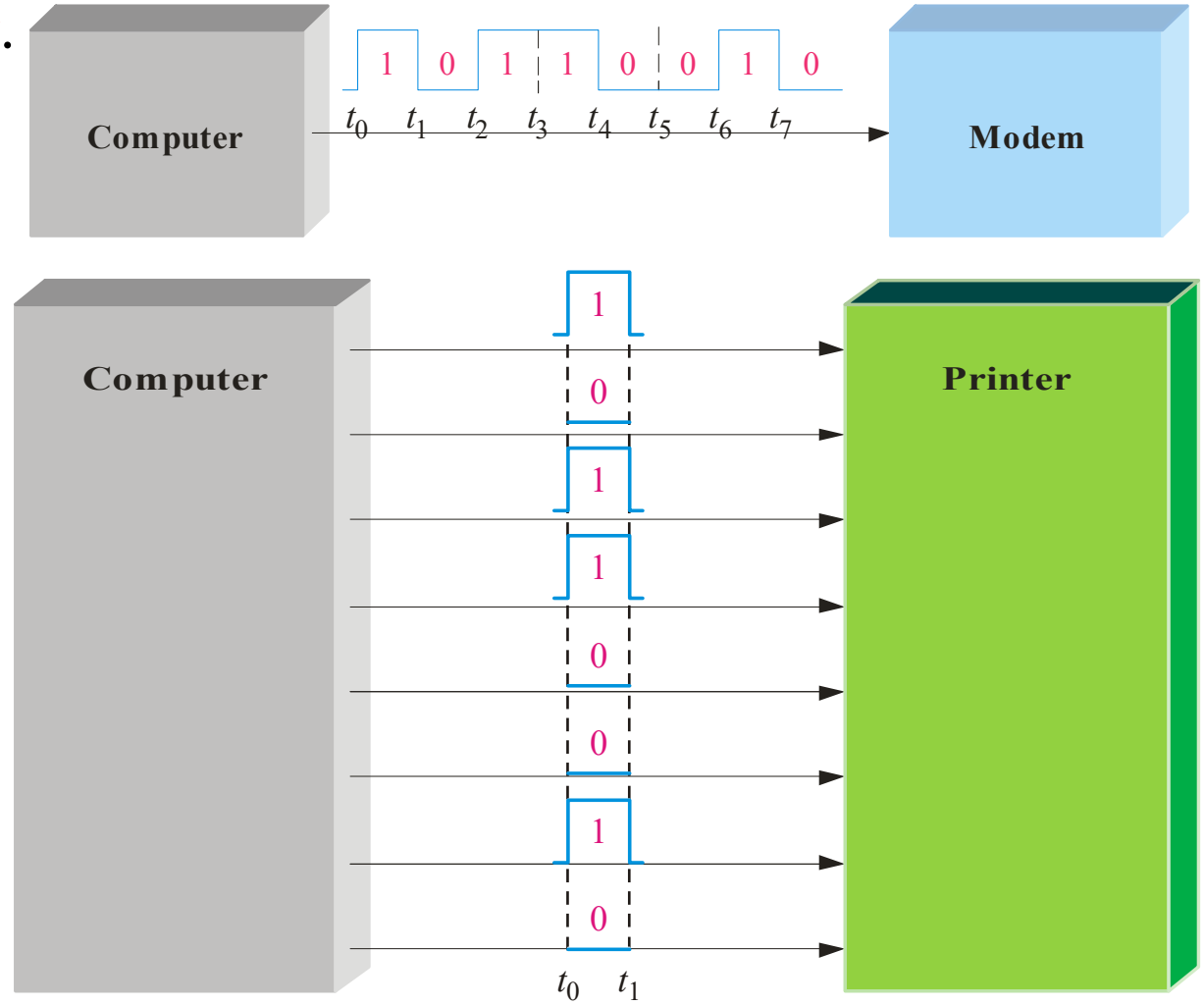
A timing diagram is used to show the relationship between two or more digital waveforms,



A diagram like this can be observed directly on a logic analyzer.

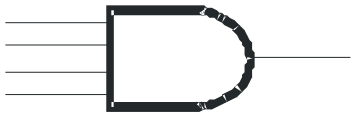
Serial and Parallel Data

Data can be transmitted by either serial transfer or parallel transfer.



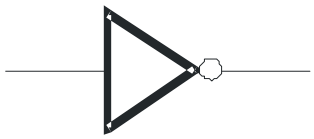
Basic Logic Functions

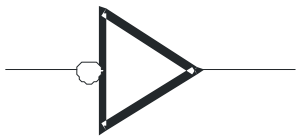
AND True only if ***all*** input conditions are true.



OR True only if ***one or more*** input conditions are true.

NOT Indicates the ***opposite*** condition.

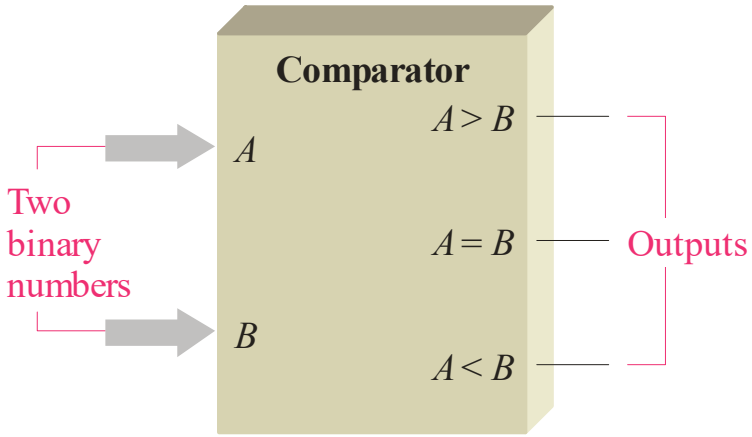




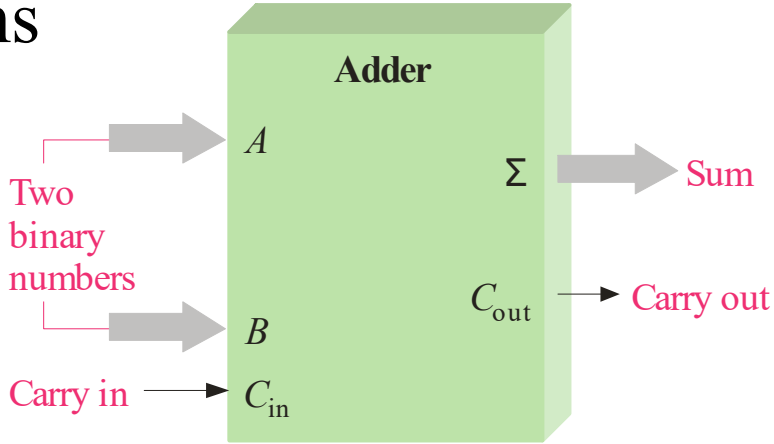
Basic System Functions

And, **or**, and **not** elements can be combined to form various logic functions. A few examples are:

The comparison function

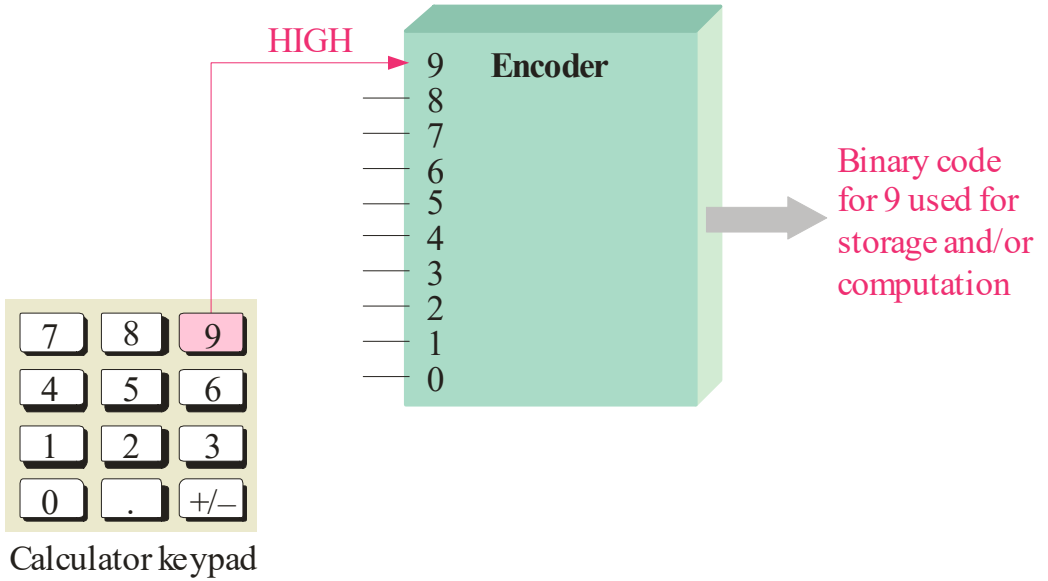


Basic arithmetic functions

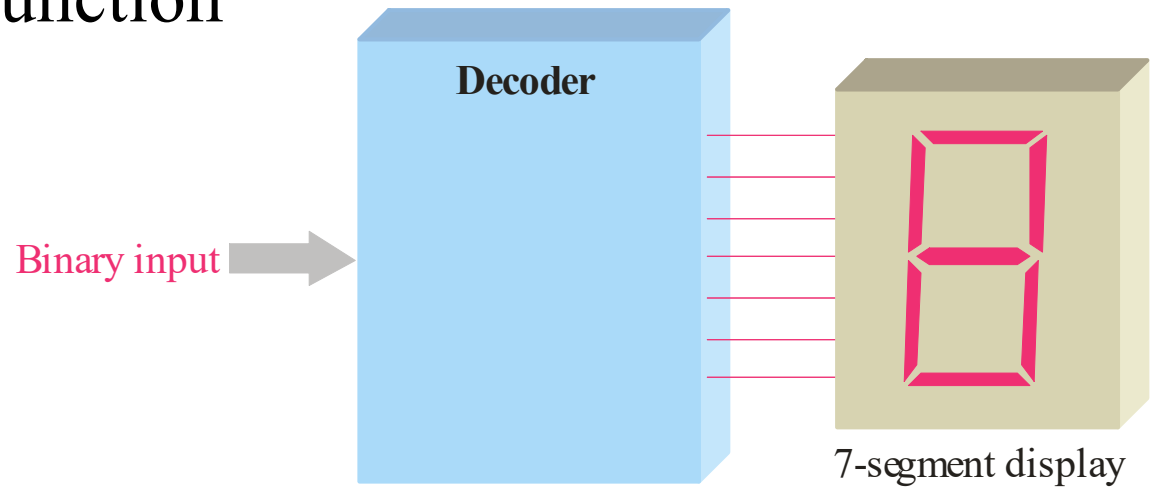


Basic System Functions

The encoding function

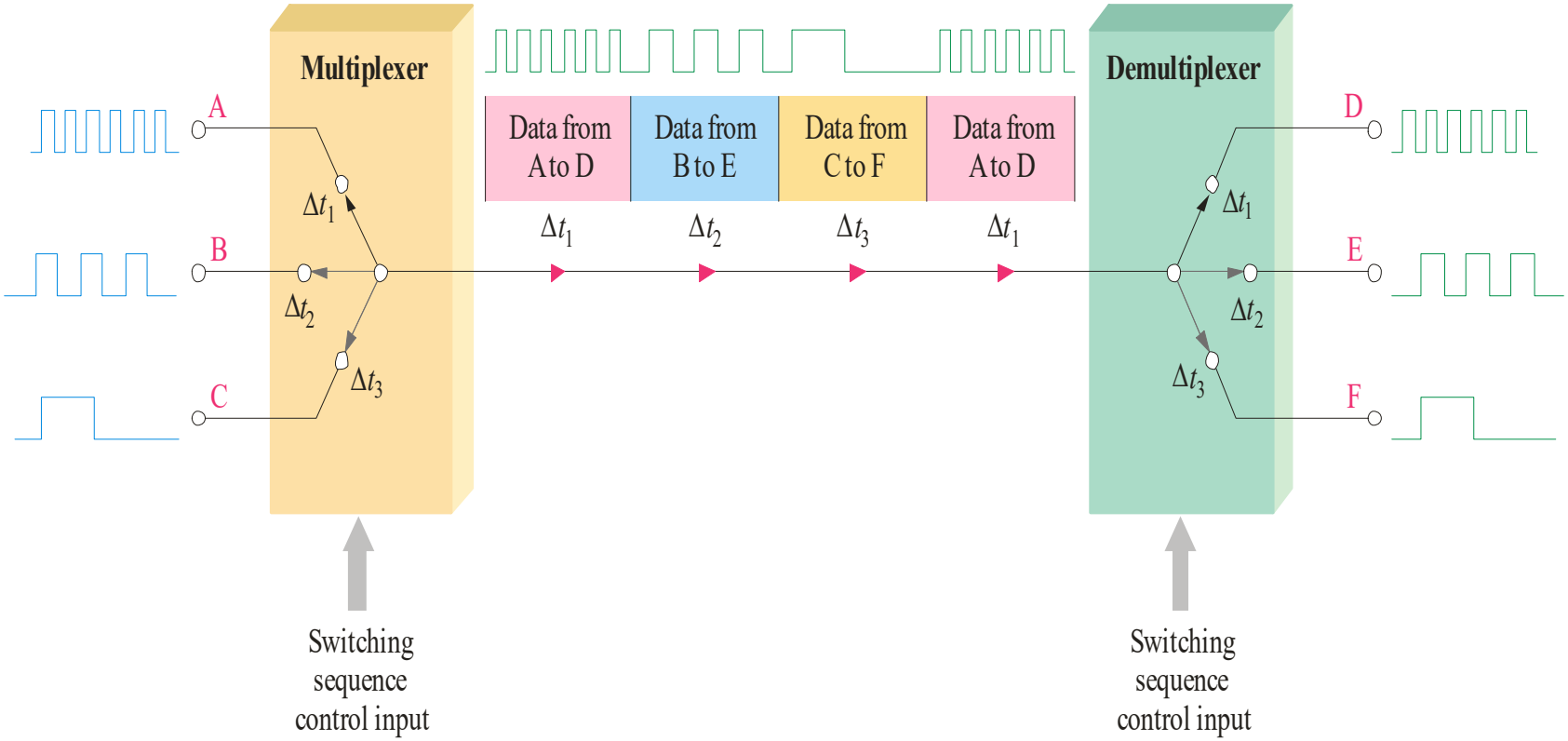


The decoding function



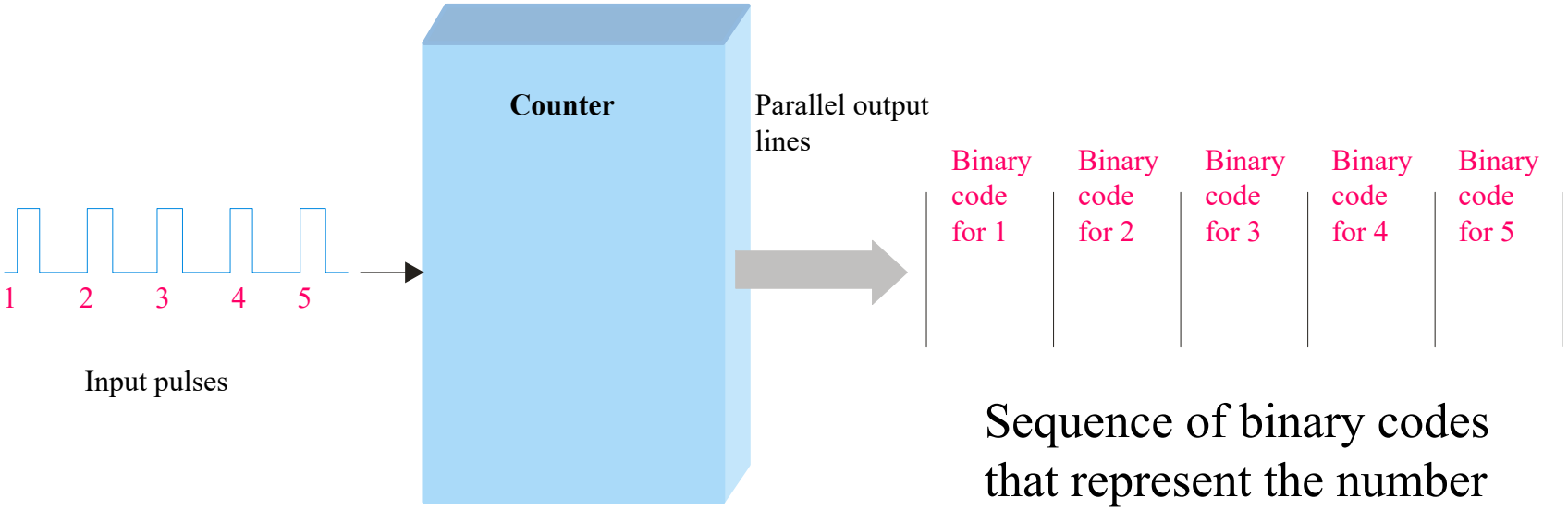
Basic System Functions

The data selection function



Basic System Functions

The counting function

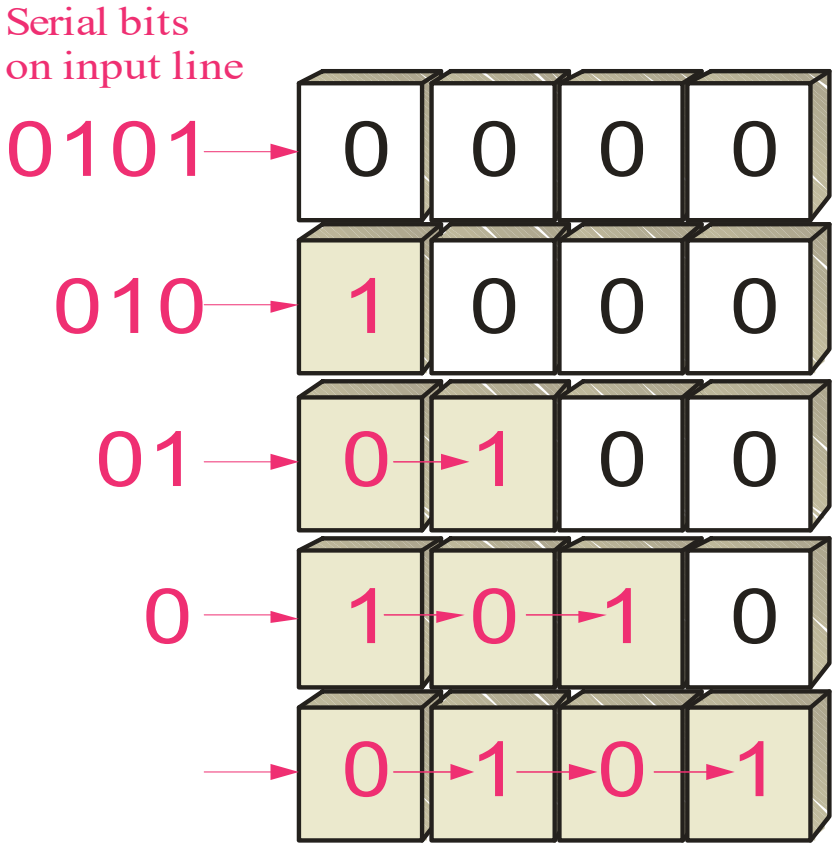


Sequence of binary codes that represent the number of input pulses counted.

…and other functions such as code conversion and storage.

Basic System Functions

One type of storage function is the shift register, that moves and stores data each time it is clocked.



Initially the register contains only invalid data or all zeros as shown here.

First bit (1) is shifted serially into the register.

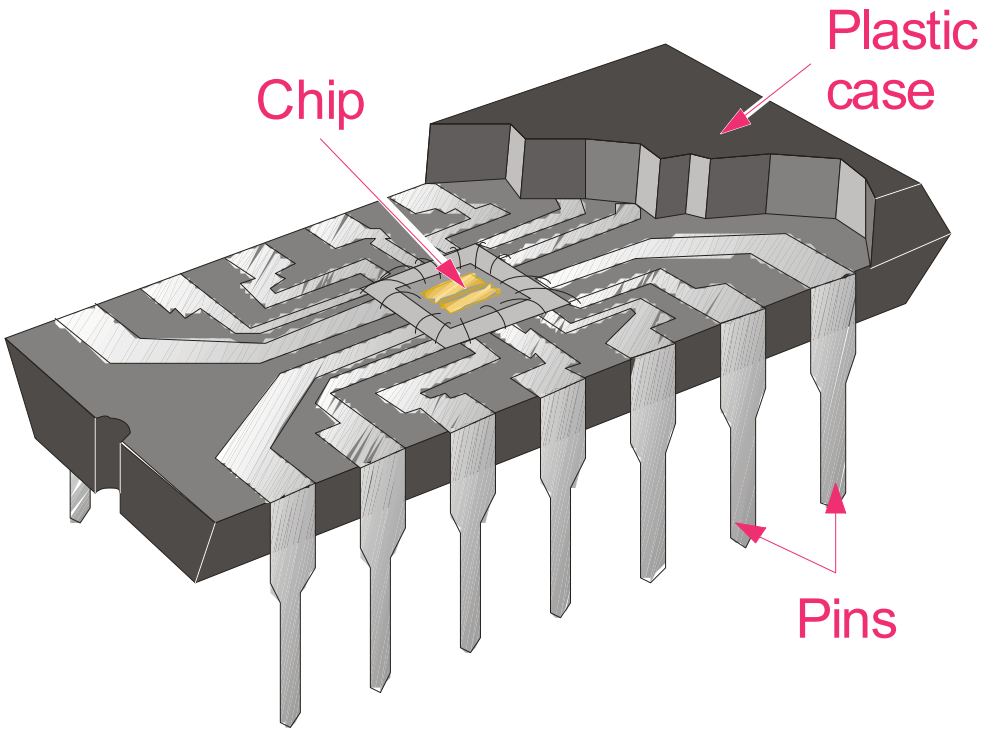
Second bit (0) is shifted serially into register and first bit is shifted right.

Third bit (1) is shifted into register and the first and second bits are shifted right.

Fourth bit (0) is shifted into register and the first, second, and third bits are shifted right. The register now stores all four bits and is full.

Integrated Circuits

- Cutaway view of DIP (Dual-In-line Pins)

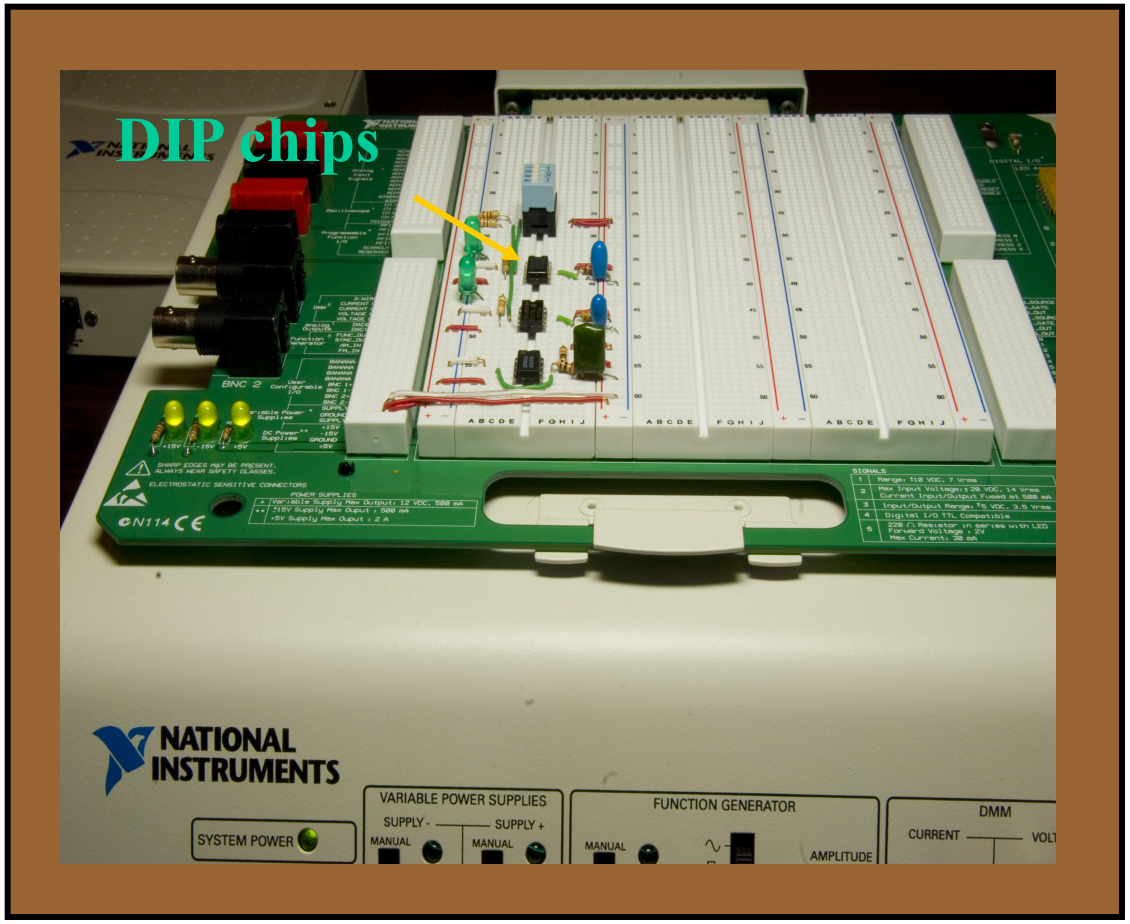


- The TTL (Transistor-Transistor Logic) series, available as DIPs are popular for laboratory experiments with logic. 5V=High, 0V=Low in TTL logic.

Integrated Circuits

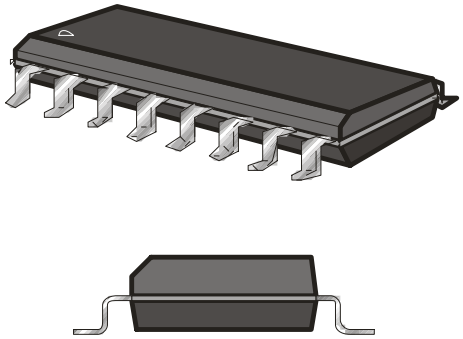
An example of laboratory prototyping is shown. The circuit is wired using DIP chips and tested.

In this case, testing can be done by a computer connected to the system.



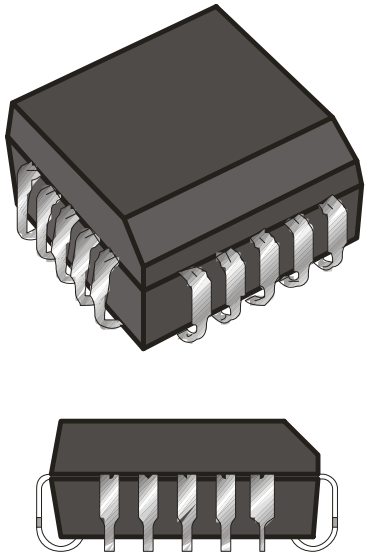
Integrated Circuits

Other surface mount packages:



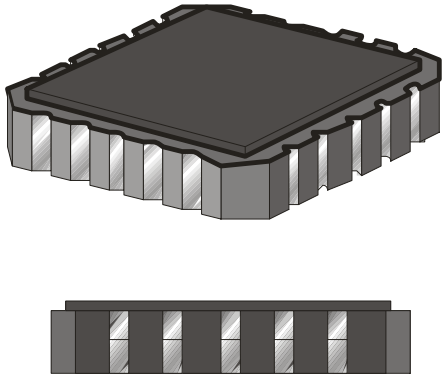
End view

SOIC



End view

PLCC

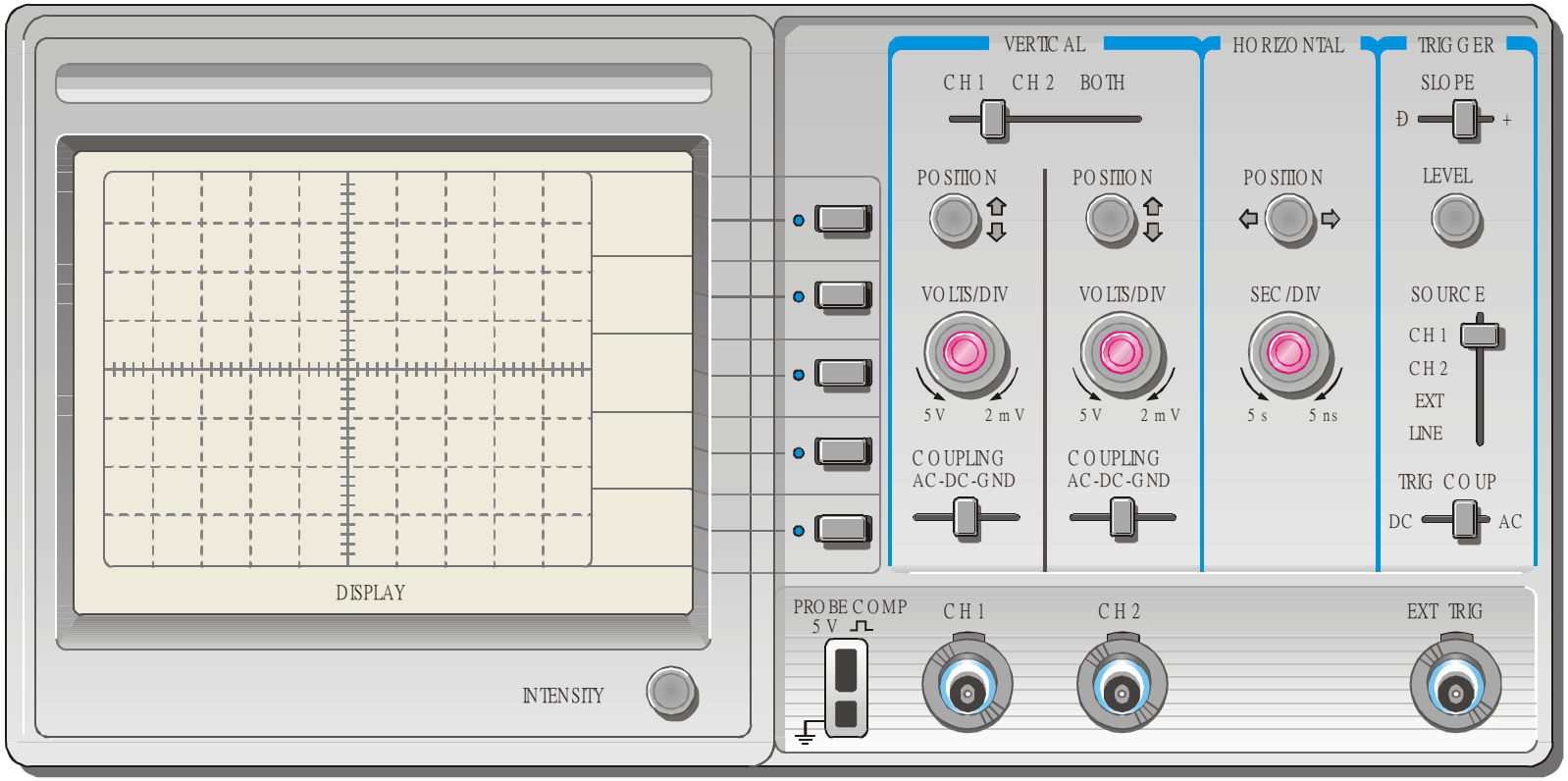


End view

LCCC

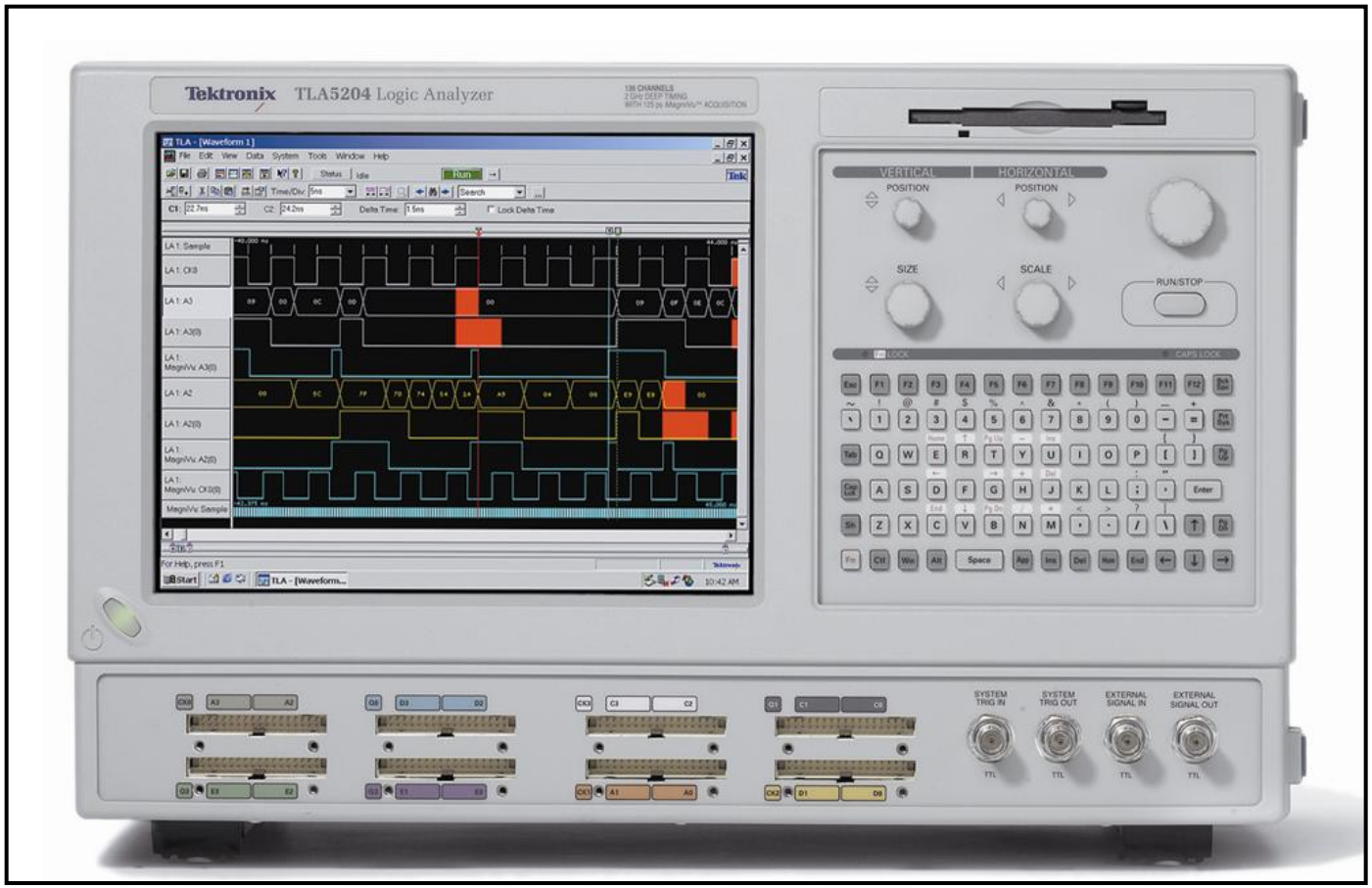
Test and Measurement Instruments

The front panel controls for a general-purpose oscilloscope can be divided into four major groups.



Test and Measurement Instruments

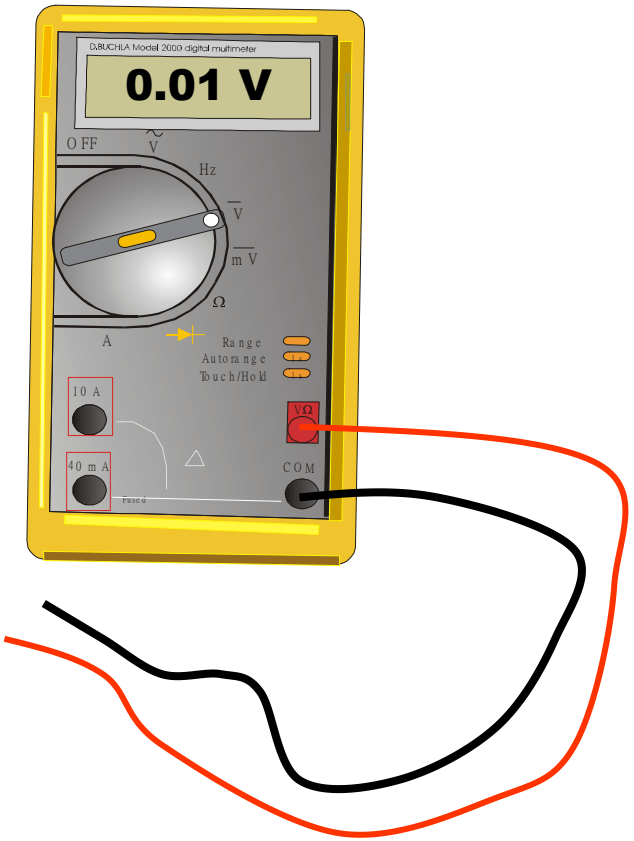
The logic analyzer can display multiple channels of digital information or show data in tabular form.



Test and Measurement Instruments

The DMM(Digital Multimeter) can make three basic electrical measurements.

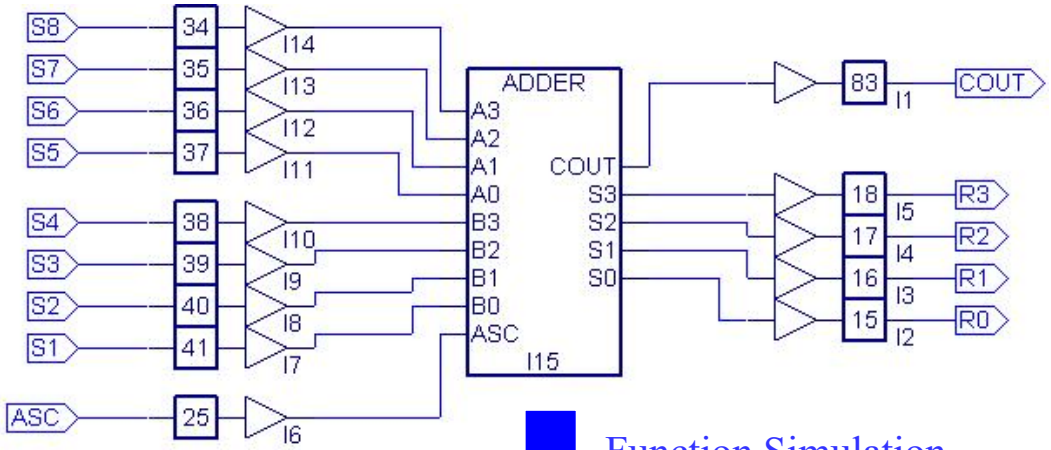
- Voltage
- Resistance
- Current



In digital work, DMMs are useful for checking power supply voltages, verifying resistors, testing continuity, and occasionally making other measurements.

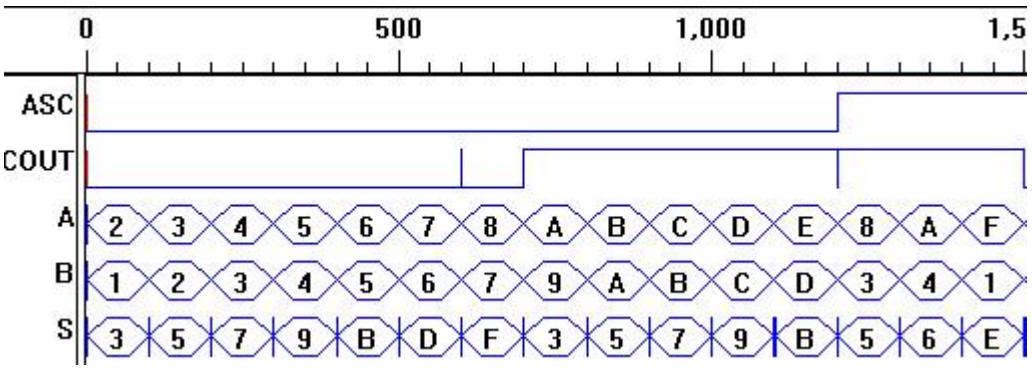
Programmable Logic Device(PLD)

Programmable logic devices can be programmed repeatedly for a specific purpose or proto-type at design level. CPLD, FPGA & ISP PLD are general PLD.



Function Simulation

Download



Traditional Digital System->*Irreversible*

- The traditional Digital System is ***irreversible*** because the **logic gates** inside **are irreversible**.
- That is, **some information** about the inputs **are erased** when a logic operation (function) is performed.
- Thus, we **cannot deduce the inputs from just knowing the outputs alone**.

AND		
X	Y	$Z = X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

Problems in the Traditional Digital System

- **Some heat** producing from computation was due to the destruction of information (eliminated bits of information) and not to the processing bits.
- According to the Landauer's principle, the **heat generated** in an irreversible logic computation is: **kTlog(2)**, for every bit of information that is lost.

  where k is Boltzmann's constant(1.3807×10^{-23} JK^{-1}) and T is the operation temperature.

  When T equals the room temperature (300K), this heat is small, i.e. **2.9×10^{-21} joule**, **but non-negligible**.
- **The heat dissipation in computing is the main power consumption in the whole digital system**.

A reversible digital system

- For any system to be reversible, it has to be able to ***operate in a backward direction***. This ability allows us to ***reproduce the inputs from the outputs*** of the system.
- In 1973, C. H. Bennett concluded that **no energy would dissipate from a (reversible) system ideally** as long as the system was able to return to its initial state from its final state regardless of what occurred in between.
- A reversible system can **save lots of power practically!**
- For example, a reversible full adder can **save 94% of the power.**

How to get a Reversible Digital System

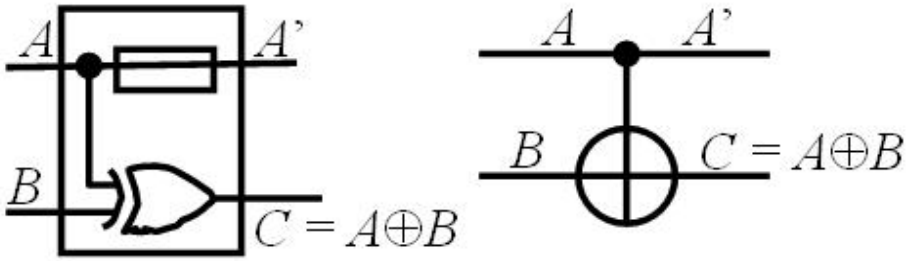
- Design a digital system **by using the reversible gates**, to **make it reversible**.
- **Use the Logic Synthesis Theory** to design a new reversible gate or reversible digital system based on some **basic reversible gates**.
- **Design some new Logic Synthesis Theories for the reversible logic.**
- Reversible logic gates **can be built physically** in CMOS technologies such as Quantum or Optical.

Overview of Basic Reversible Gates

- Any logic gate can be characterized by its number of inputs and outputs. A (m, n) gate has m inputs and n outputs.
- **For a reversible gate, m should be equals to n(m=n).**
- A trivial (1, 1) reversible gate is an input copying gate: the input simply passes though the gate unchanged. We call it a ***buffer***.
- Another example is the ***one-bit inverter***, with its output being the inverse of its input.

(2, 2) Reversible Logic Gates---Feynman gate

- A Feynman gate with input vector (A, B) and output vector (A', C) implements the logic functions: **A' = A and C =A xor B**.

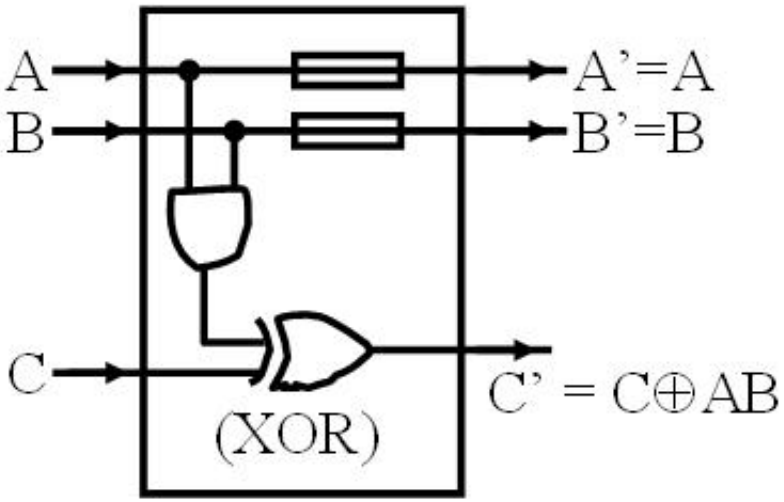


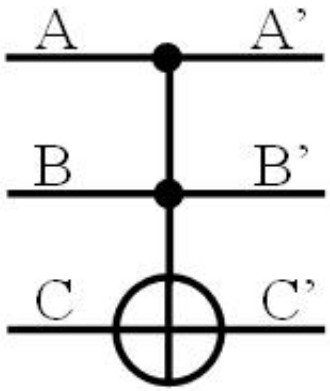
A	B→	A'	C
0	0→	0	0
0	1→	0	1
1	0→	1	1
1	1→	1	0

- If A = 0, then the output A' is simply duplicating the input *A*; if A = 1, then the output C =B⊕1 = ~B, an inverse of the input B.
- For this reason, the Feynman gate is also called the *controlled NOT* (1-CNOT) gate, or the *quantum XOR* gate

(3, 3) Reversible Logic Gates---Toffoli Gate (CCNOT)

- This gate has the input-output relations: **A' = A, B' = B, and C' = C ⊕ (A·B).** The Toffoli gate is a **2-CNOT gate** since **it passes the two inputs (A & B) unchanged and it inverts the input *C* when A=1 and B=1(C' = C⊕ (A · B)=C⊕1=~C).**

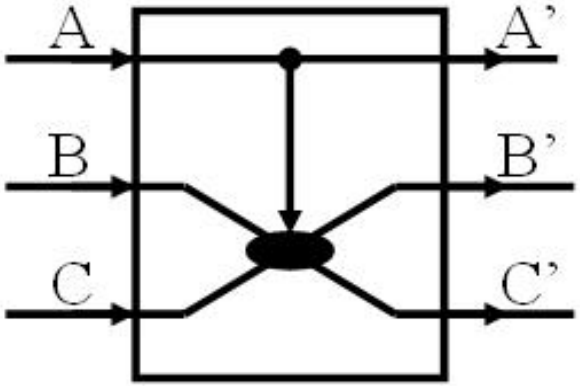




A	B	C	A'	B'	C'
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

(3, 3) Reversible Logic Gates---Fredkin Gate

- **B and C are swapped if A=1**, otherwise passed unchanged.



A	B	C	A'	B'	C'
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	1	1	1

- The first universal reversible logic gate to be discovered.
- It can be **"programmed" to become AND,OR and Not** gate.

Fredkin gate implementation of (a) AND (b) OR and (c) NOT gate

- **AND gate**:

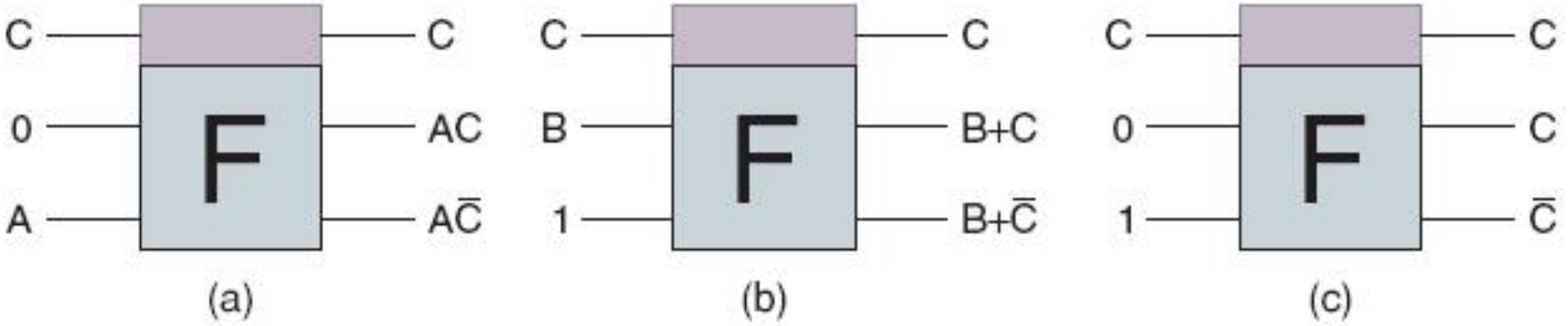
  If input ***B* = 0**, then *C'* = *C*, **B'=*AC***, and A'= *A(~C)*.

- **OR gate**:

  If input **A = 1**, then C' = C, **B' = *BC* + *C*= *B* + *C***, and A' = *B* +(~ *C*).

- **NOT gate**:

  If ***A* = 1 and *B* = 0**, then *C'* = *C*, *B'* = *C*, and ***A'* = ~*C*** .



Applications of the Reversible Digital System

- Low-power Processor Design
- Low-power VLSI
- Quantum computing
- Nanotechnology
- Digital Signal Processing
- Communication
- Computer Graphics
- Cryptography
- Computational Complexity Theory
- Parallel Programs

Number Systems–Representation

- Positive radix/base, positional number systems

- A number with *radix r* is represented by a string of digits:

$$A_{n-1}A_{n-2}\cdots A_1A_0 \cdot A_{-1}A_{-2}\cdots A_{-m+1}A_{-m}$$

  in which $\mathbf{0} \le A_i < r$ and . is the *radix point*.

- The string of digits represents the power series:

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

(Integer Portion) + (Fraction Portion)

Commonly Occurring Bases

Name	Radix	Digits
Binary	2	0,1
Octal	8	0,1,2,3,4,5,6,7
Decimal	10	0,1,2,3,4,5,6,7,8,9
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

- The six letters A,B,C,D,E,F represent the digit values of 10,11,12,13,14,15 (in decimal) respectively

Decimal Numbers

The radix *r* of decimal numbers is **ten**, because only ten symbols (0 through 9) are used to represent any number.

The **column weights** of integral decimal numbers are powers of ten that increase from right to left beginning with 100 =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 $2^0 = 1$: $\ldots 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0.$

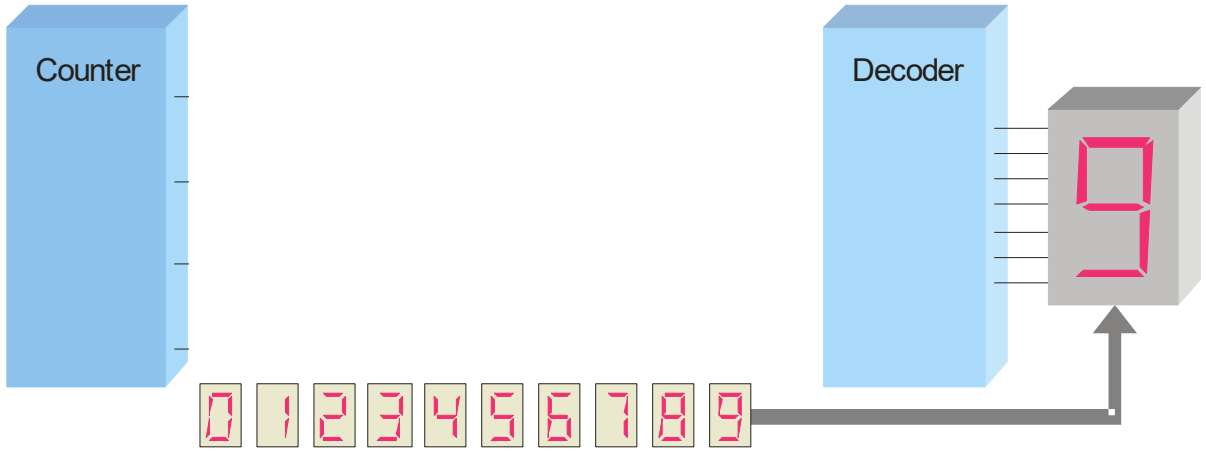
For fractional binary numbers, the **column weights** are negative powers of two that decrease from left to right: $2^2\ 2^1\ 2^0.\ 2^{-1}\ 2^{-2}\ 2^{-3}\ 2^{-4}\ \ldots$

Binary Numbers

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

Negative Positional Number Systems

If the $r<-1$, we say it is a Negative Positional Number Systems and we can use $0,\ldots,|r|-1$ symbols to represent the all the integer (in which $\mathbf{0} \le A_i < |r|$).

$$(number)_r = \sum_{i=0}^{n-1} A_i r^i$$

For example, we set $r = -2(0,|r|-1=1)$, then 5 and -3 in decimal can be expressed as

$$5 = 1\cdot(-2)^2 + 0\cdot(-2)^1 + 1\cdot(-2)^0 = (101)_{-2}$$

$$-3 = 1\cdot(-2)^3 + 1\cdot(-2)^2 + 0\cdot(-2)^1 + 1\cdot(-2)^0 = (1101)_{-2}$$

Negative Positional Number Systems

r=10	r=-2	r=10	r=-2
1	1	-1	11
2	110	-2	10
3	111	-3	1101
4	100	-4	1100
5	101	-5	1111
6	11010	-6	1110
7	11011	-7	1001
8	11000	-8	1000
9	11001	-9	1011
10	11110	-10	1010
11	11111	-11	110101
12	11100	-12	110100
13	11101	-13	110111
14	10010	-14	110110
15	10011	-15	110001

Gaussian Integer & Complex Radix

If x, y is real Integers, then $z = x + iy$ is called Gaussian Integer ($i = \sqrt{-1}$).

If r is the complex radix, then Gaussian Integer can be expressed as

$$Z = \sum_{j=0}^{k} A_j r^j$$

By experiments, we know that if the $r = -1+i$,

Then any Gaussian Integer can be expressed as

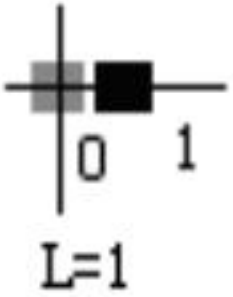
$$Z = \sum_{j=0}^{k} A_j (-1+i)^j$$

Gaussian Integer on Z plane when *r=-1+i*

Let A_i=(0,1) and L=length of Symbols

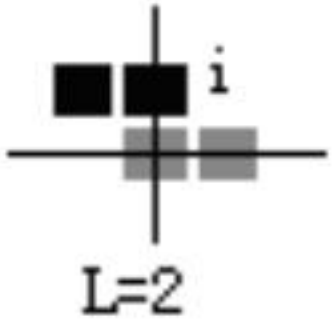
L=1 , $A_0 = 0, \quad Z = 0$

$A_0 = 1, \quad Z = 1$



L=2, $A_1A_0 = 10, \quad Z = 1 \times (i-1)^1 + 0 = -1 + i$

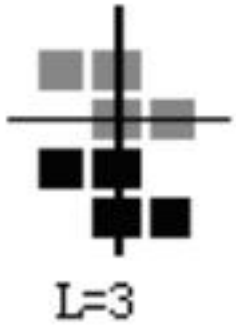
$A_1A_0 = 11, \quad Z = 1 \times (i-1)^1 + 1 = i$



Gaussian Integer on Z plane when *r=-1+i*

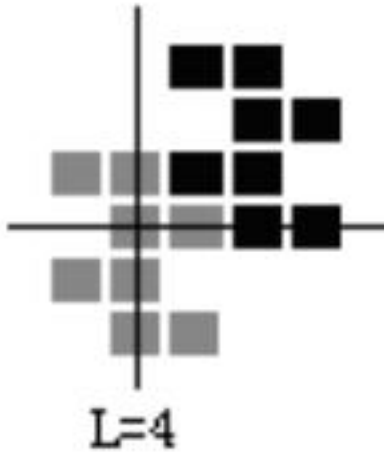
- L=3 ,

$$A_2A_1A_0 = 100, \quad N = 1\times(i-1)^2 + 0\times(i-1)^1 + 0 = -2i$$
$$A_2A_1A_0 = 101, \quad N = 1\times(i-1)^2 + 0\times(i-1)^1 + 1 = 1-2i$$
$$A_2A_1A_0 = 110, \quad N = 1\times(i-1)^2 + 1\times(i-1)^1 + 0 = -1-i$$
$$A_2A_1A_0 = 111, \quad N = 1\times(i-1)^2 + 1\times(i-1)^1 + 1 = -i$$



- L=4,

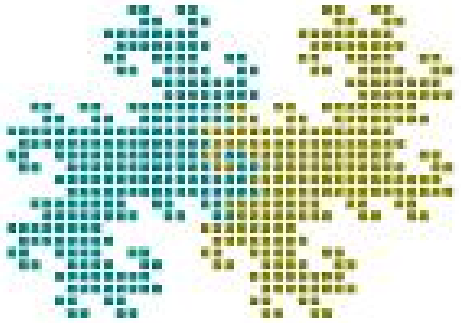
$$A_3A_2A_1A_0 = 1000, \quad N \to 1\times(i-1)^3 = 2+2i$$
$$A_3A_2A_1A_0 = 1001, \quad N \to 1\times(i-1)^3 + 1 = 3+2i$$
$$A_3A_2A_1A_0 = 1010, \quad N \to 1\times(i-1)^3 + 1\times(i-1)^1 = 1+3i$$
$$A_3A_2A_1A_0 = 1011, \quad N \to 1\times(i-1)^3 + 1\times(i-1)^1 + 1 = 2+3i$$
$$A_3A_2A_1A_0 = 1100, \quad N \to 1\times(i-1)^3 + 1\times(i-1)^2 = 2$$
$$A_3A_2A_1A_0 = 1101, \quad N \to 1\times(i-1)^3 + 1\times(i-1)^2 + 1 = 3$$
$$A_3A_2A_1A_0 = 1110, \quad N \to 1\times(i-1)^3 + 1\times(i-1)^2 + 1\times(i-1)^1 = 1+i$$
$$A_3A_2A_1A_0 = 1111, \quad N \to 1\times(i-1)^3 + 1\times(i-1)^2 + 1\times(i-1)^1 + 1 = 2+i$$



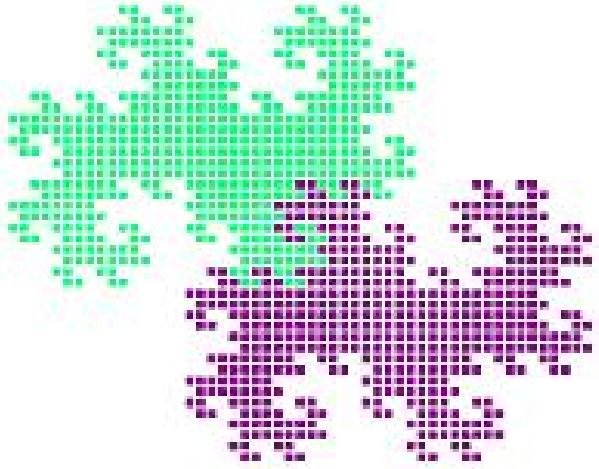
Beautiful Picture from Gaussian Integer on Z plane when *r = -1+i*

L=7

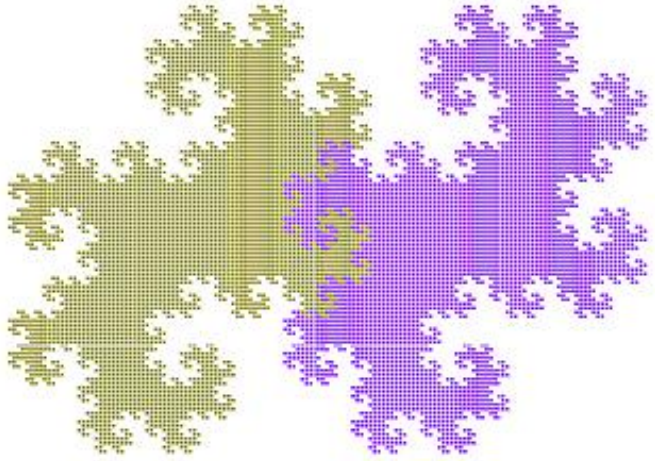
L=8



L=9



L=10



L=11

Conversion Between Bases

- **To Convert the Integral Part:**

  Repeatedly divide the number by the new radix and save the remainders. The digits for the new radix are the remainders in *reverse order* of their computation. If the new radix is > 10, then convert all remainders > 10 to digits A, B, …

- **To Convert the Fractional Part:**

  Repeatedly multiply the fraction by the new radix and save the integer digits that result. The digits for the new radix are the integer digits in *order* of their computation. If the new radix is > 10, then convert all integers > 10 to digits A, B, …

Example: Convert 46.6875_{10} To Base 2

Converting 46 as integral part:

46/2 = 23	remainder = 0
23/2 = 11	remainder = 1
11/2 = 5	remainder = 1
5/2 = 2	remainder = 1
2/2 = 1	remainder = 0
1/2 = 0	remainder = 1

Converting 0.6875 as fractional part:

0.6875 * 2 = 1.3750	int = 1
0.3750 * 2 = 0.7500	int = 0
0.7500 * 2 = 1.5000	int = 1
0.5000 * 2 = 1.0000	int = 1
0.0000	

Integral part: reading off in the reverse direction to get 101110_2

Fractional part: reading off in the forward direction to get 0.1011_2

Combining Integral and Fractional Parts: $101110.\ 1011_2$

Checking the Conversion

- To convert back, sum the digits times their respective powers of r.
- From the prior conversion of 46.6875_{10}

$$101110_2 = 1\cdot32 + 0\cdot16 + 1\cdot8 + 1\cdot4 + 1\cdot2 + 0\cdot1$$
$$= 32 + 8 + 4 + 2$$
$$= 46$$

$$0.1011_2 = 1/2 + 1/8 + 1/16$$
$$= 0.5000 + 0.1250 + 0.0625$$
$$= 0.6875$$

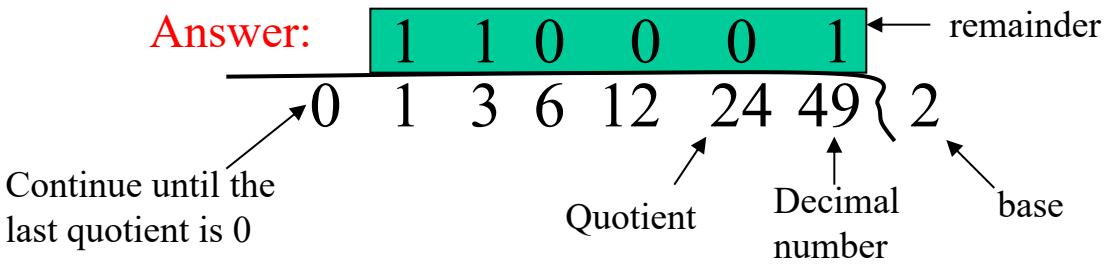
Additional Issue - Fractional Part

- Note that in this conversion, the fractional part can become 0 as a result of the repeated multiplications.
- In general, it may take many bits to get this to happen or it may never happen.
- Example Problem: Convert 0.65_{10} to N_2
  - 0.65 = 0.10100110011001 …
  - The fractional part begins repeating every 4 steps yielding repeating 1001 forever!
- Solution: Specify number of bits to right of radix point and round or truncate to this number.

Binary Conversions Example

Example Convert the decimal number 49.188 to binary (for five significant digits at fraction)

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 =110001.00110 (for five significant digits at fraction)

Hexadecimal Numbers to Decimal

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

Column weights $\begin{matrix} 16^3 & 16^2 & 16^1 & 16^0. \\ 4096 & 256 & 16 & 1. \end{matrix}$

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

Octal Numbers to Decimal

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

Octal (Hexadecimal) to Binary and Back

- **Octal (Hexadecimal) to Binary:**
  - **Restate the octal (hexadecimal) as three (four) binary digits starting at the radix point and going both ways.**
- **Binary to Octal (Hexadecimal):**
  - **Group the binary digits into three (four) bit groups starting at the radix point and going both ways, padding with zeros as needed in the fractional part.**
  - **Convert each group of three bits to an octal (hexadecimal) digit.**

Octal to Hexadecimal via Binary

- **Convert octal to binary.**
- **Use groups of four bits and convert as above to hexadecimal digits.**
- **Example: Octal to Binary to Hexadecimal**

$$6\quad 3\quad 5.\quad 1\quad 7\quad 7_8$$

$$110\ 011\ 101\ .\ 001\ 111\ 111_2$$

- Regroup:

$$1\ \underline{1001}\ \underline{1101}\ .\ \underline{0011}\ \underline{1111}\ \underline{1(000)}_2$$

- Convert:

$$1\quad 9\quad D.\quad 3\quad F\quad 8_{16}$$

Exercise 1

1. Convert the following binary numbers to decimal: 1001101, 1010011.101
2. Convert the following decimal numbers to binary:193, 2007
3. Convert the following decimal numbers to the indicated bases **(round to four significant digits at fraction)**

   (a) 7562.45 to octal

   (b) 1938.257 to hexadecimal

Do it manually and fill the answers in EIE130_Chapter1_Exercise1 via 電子作業 on examcoo.com

Arithmetic Operations - Binary Arithmetic

- Single Bit Addition with Carry
- Multiple Bit Addition
- Single Bit Subtraction with Borrow
- Multiple Bit Subtraction
- Multiplication
- BCD Addition

Single Bit Binary Addition with Carry

Given two binary digits (X,Y), a carry in (Z) we get the following sum (S) and carry (C):

Carry in (Z) of 0:

Z	0	0	0	0
X	0	0	1	1
+ Y	+ 0	+ 1	+ 0	+ 1
C S	0 0	0 1	0 1	1 0

Carry in (Z) of 1:

Z	1	1	1	1
X	0	0	1	1
+ Y	+ 0	+ 1	+ 0	+ 1
C S	0 1	1 0	1 0	1 1

Multiple Bit Binary Addition

- **Extending this to two multiple bit examples:**

Carries	**0**	**0**
Augend	**01100**	**10110**
Addend	**+10001**	**+10111**
Sum	**11101**	**101101**

- **Note: The 0 is the default Carry-In to the least significant bit.**

Single Bit Binary Subtraction with Borrow

- **Given two binary digits (X,Y), a borrow in (Z) we get the following difference (S) and borrow (B):**
- **Borrow in (Z) of 0:**

Z	0	0	0	0
X	0	0	1	1
- Y	-0	-1	-0	-1
BS	0 0	1 1	0 1	0 0

- **Borrow in (Z) of 1:**

Z	1	1	1	1
X	0	0	1	1
- Y	-0	-1	-0	-1
BS	11	1 0	0 0	1 1

Multiple Bit Binary Subtraction

- **Extending this to two multiple bit examples:**

Borrows	**0**	**0**	
Minuend	**10110**	**10011**	**11110**
Subtrahend	**- 10010**	**- 11110**	**-10011**
Difference	**00100**	**00001**	**-01011**

- **Notes: If the Subtrahend > the Minuend, interchange them and append a minus sign "–" to the result.**

Binary Multiplication

The binary multiplication table is simple:

$$0 * 0 = 0 \mid 1 * 0 = 0 \mid 0 * 1 = 0 \mid 1 * 1 = 1$$

Extending multiplication to multiple digits:

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

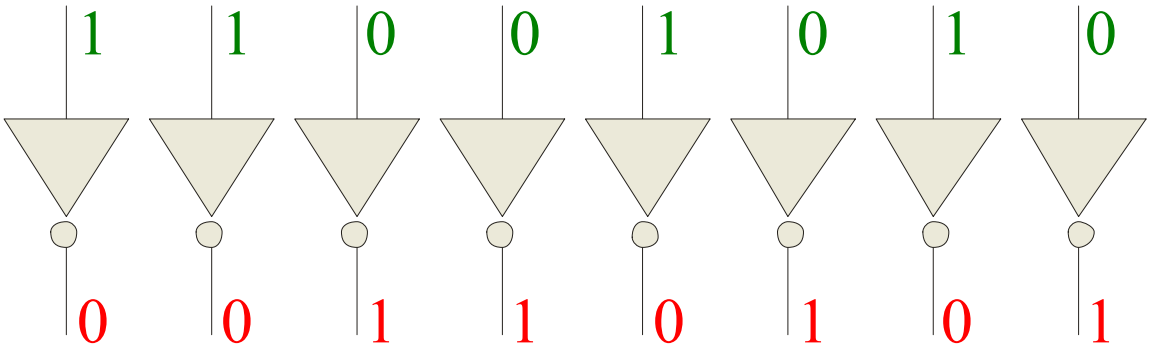
1's Complement

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:



2's Complement

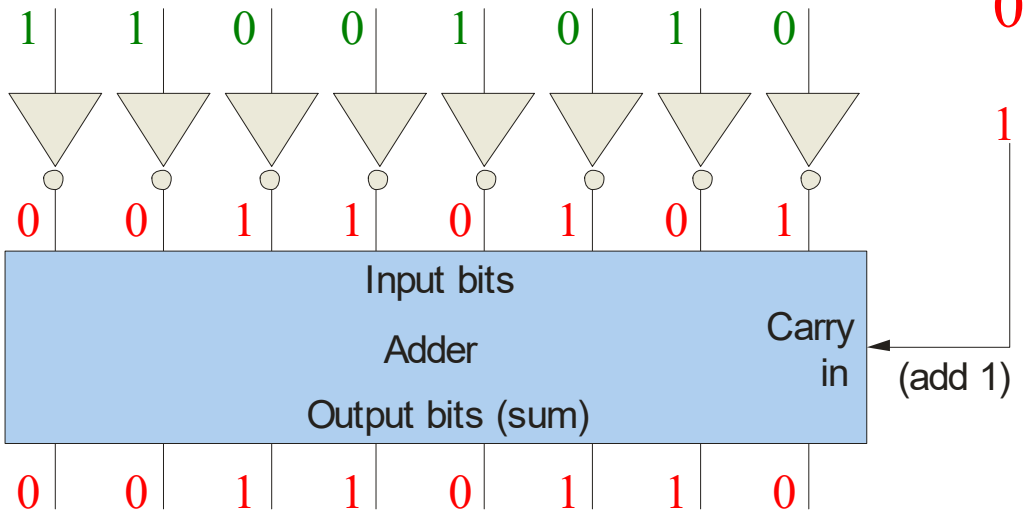
The 2's complement of a binary number is found by adding 1 to the LSB of the 1's complement.

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 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

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

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:

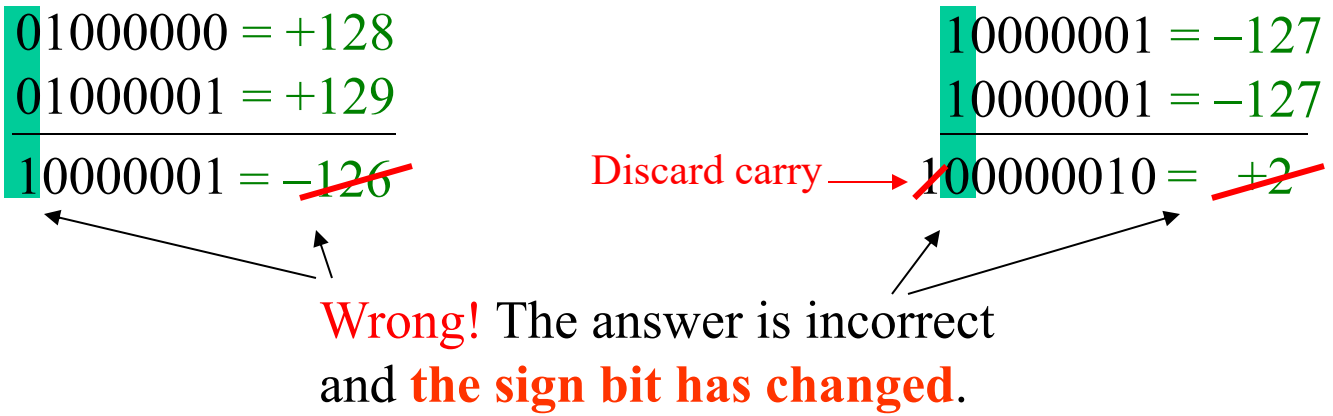
00011110 = +30	00001110 = +14	11111111 = –1
00001111 = +15	11101111 = –17	11111000 = –8
00101101 = +45	11111101 = –3	1 11110111 = –9

Discard carry

Arithmetic Operations with Signed Numbers

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:



Arithmetic Operations with Signed Numbers

Rules for **subtraction**: 2's complement the subtrahend and add the numbers. Discard any final carries. The result is in signed form.

Repeat the examples done previously, but subtract:

00011110	(+30)	00001110	(+14)	11111111	(–1)
– 00001111	–(+15)	– 11101111	–(–17)	– 11111000	–(–8)

2's complement subtrahend and add:

00011110 = +30	00001110 = +14	11111111 = –1
11110001 = –15	00010001 = +17	00001000 = +8
~~1~~00001111 = +15	00011111 = +31	~~1~~00000111 = +7

Discard carry

Discard carry

Binary Coded Decimal (BCD)

- **The BCD code is the 8,4,2,1 code.**
- **8, 4, 2, and 1 are weights**
- **BCD is a *weighted* code**
- **This code is the simplest, most intuitive binary code for decimal digits and uses the same powers of 2 as a binary number, but only encodes the first ten values from 0 to 9.**
- **Example: 1001 (9) = 1000 (8) + 0001 (1)**

Binary Coded Decimal (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	0001 0000
11	1011	0001 0001
12	1100	0001 0010
13	1101	0001 0011
14	1110	0001 0100
15	1111	0001 0101

BCD Arithmetic

- Given a BCD code, we use binary arithmetic to add the digits:

8	1000	Eight
+5	+0101	Plus 5
13	1101	is 13 (> 9)

- Note that the result is MORE THAN 9, so must be represented by two digits!
- To correct the digit, subtract 10 by adding 6 modulo 16.

8	1000	Eight
+5	+0101	Plus 5
13	1101	is 13 (> 9)
	+0110	so add 6
carry = 1	0011	leaving 3 + carry
	0001 | 0011	Final answer (two digits)

- If the digit sum is > 9, add one to the next significant digit

BCD Addition Example

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

	1	1	1	0
	0001	1000	1001	0111
+	0010	1001	0000	0101
	0100	10010	1010	1100
+	0000	0110	0110	0110
	0100	1000	0000	0010
	(4	8	0	$2)_{BCD}$

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

ASCII (American Standard Code for Information Interchange)

ASCII (American Standard Code for Information Interchange 0is 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.

ASCII (American Standard Code for Information Interchange)

ASCII value	Character	Control character	ASCII value	Character	ASCII value	Character	ASCII value	Character
000	(null)	NUL	032	(space)	064	@	096	
001	☺	SOH	033	!	065	A	097	a
002	☻	STX	034	"	066	B	098	b
003	♥	ETX	035	#	067	C	099	c
004	♦	EOT	036	$	068	D	100	d
005	♣	ENQ	037	%	069	E	101	e
006	♠	ACK	038	&	070	F	102	f
007	(beep)	BEL	039	'	071	G	103	g
008	◘	BS	040	(	072	H	104	h
009	(tab)	HT	041	)	073	I	105	i
010	(line feed)	LF	042	*	074	J	106	j
011	(home)	VT	043	+	075	K	107	k
012	(form feed)	FF	044	,	076	L	108	l
013	(carriage return)	CR	045	-	077	M	109	m
014	♫	SO	046	.	078	N	110	n
015	☼	SI	047	/	079	O	111	o
016	►	DLE	048	0	080	P	112	p
017	◄	DC1	049	1	081	Q	113	q
018	↕	DC2	050	2	082	R	114	r
019	‼	DC3	051	3	083	S	115	s
020	¶	DC4	052	4	084	T	116	t
021	§	NAK	053	5	085	U	117	u
022	▬	SYN	054	6	086	V	118	v
023	↨	ETB	055	7	087	W	119	w
024	↑	CAN	056	8	088	X	120	x
025	↓	EM	057	9	089	Y	121	y
026	→	SUB	058	:	090	Z	122	z
027	←	ESC	059	;	091	[	123	{
028	(cursor right)	FS	060	<	092	\	124	|
029	(cursor left)	GS	061	=	093	]	125	}
030	(cursor up)	RS	062	>	094	^	126	~
031	(cursor down)	US	063	?	095	_	127	⌂

Exercise 2

1. Perform the following binary multiplications:

   (a) 1101X1011 (b) 0101X1010

2. Find the binary representations for each of the following BCD numbers:

   (a) 0100 1000 0110 0111

   (b) 0011 0111 1000.0111 0101

Do it manually and fill the answers in EIE130_Chapter1_Exercise2 via 電子作業 on examcoo.com

EIE130 Assignment 1

1-7. Convert the following binary numbers to decimal:
1001101, 1010011.101 and 10101110.1001

1-8. Convert the following decimal numbers to **binary:193, 751, 2007 and 19450**

1-10. Convert the following decimal numbers to the indicated bases **(round to four significant digits at fraction)**
(a) 7562.45 to octal **(b) 1938.257 to hexadecimal**
(c) 175.175 to binary

1-12. Perform the following binary multiplications:
(a) 1101X1011 **(b) 0101X1010** **(c) 100111X011011**

1-18. Find the binary representations for each of the following BCD numbers:
(a) 0100 1000 0110 0111 **(b) 0011 0111 1000.0111 0101**

Do it manually and fill the answers in EIE130_Chapter1_Assignment1 via 班級考試 on examcoo.com

Due date: D1:Next Wednesday
D2:Next Friday