

Week 1- T1 2020

Introduction/Number Systems/Combinational Circuits

ELEC2141: Digital Circuit Design



Staff and consultation

Course coordinator:

Dr. Daniel Ssu-Han Chen

EE&T 346/329, ssu-han.chen@unsw.edu.au

Consultation hours:

Tuesday/Thursday 1am-2pm

Tutor:

Dr. Daniel Ssu-Han Chen

EE&T 346/329, ssu-han.chen@unsw.edu.au

Dr. Aron Michael

EE&T 316, a.michael@unsw.edu.au

Laboratory coordinator:

Yen Nee Ho, yennee.ho@unsw.edu.au



Course format

- **Formal face-to-face lectures (Mon/Thurs)**
 - Lecture recordings on Moodle
- **Tutorials (Week 2-10)**
 - No recordings
- **Fortnight online quizzes (Due in Week 3, 5, 7, 9)**
 - 10 Questions each. Two attempts, highest grade
- **Pre-lab quizzes (need to be completed prior to lab)**
 - 5 Questions each. Unlimited attempts, must score 100%
- **Laboratory sessions (Week 2-9)**
 - 8 Different labs. No Exemptions for repeating students.
 - Lab exam (Week 10/11)



Course format

- **Midterm exam (Week 5)**
 - Cover all materials up to week 4
- **Design Assignments (due in Week 6 and 9)**
 - Assignment I: Combinational logic circuit design
 - Assignment II: Sequential circuit design
- **Final exam**
 - Must pass the final exam to pass the course



Assessment

Fortnight online quizzes	5%
Pre-lab quizzes (must be completed)	0%
Laboratory practical experiments	15%
Lab examination	5%
Assignments (I & II)	20%
Midterm exam (1 hour)	15%
Final Exam (2 hours)	40%

Course resources

Textbook

- M. Mano, C. R. Kime and T. Martin, Logic and Computer Design Fundamentals, 5th Edition (Global Edition), Pearson, 2016. (e-book version available on UNSW library)

Online resources

- Lecture notes will be uploaded on Moodle after the lecture
- Tutorial solutions will be uploaded every Friday
- Digital version of lab manual available on Moodle

Others

- Lab manual can be purchased from the Bookshop from Week 1 (around Wednesday)
- Lab components can be obtained from the workshop (first one is free!)

Other matters

Academic Honestly and Plagiarism

- **DO NOT COPY ASSIGNMENTS!**

Workload

- 15 hours per week (4 hour lecture + 1 hour tutorial + 2 hour labs + 8 independent study)

Attendance

- Must attend 80% of the labs
- Attendance will not be checked for tutorials and lectures, but students are expected to show up to get the best learning experience

Other matters

Special Consideration and supplementary examinations

- Since 2019, processing of special consideration application was centralised. Fit to sit rule was introduced.
- Must submit the application prior to the start of the exam or before the assessment is submitted, except where illness or misadventure prevent you from doing so.
- <https://student.unsw.edu.au/special-consideration>

Course introduction

This course cover the analysis & design of digital circuits.

Digital circuits manipulate discrete or digital voltage instead of continuous or analog voltages.

They are used in personal computers, mobile phone, embedded control system, digital signal processing, telecommunications, speech analysis and recognition.

First half of the course: analysis and design of combinational and sequential logic circuits.

Second half of the course: VHSIC-Very High Speed Integrated Circuit Hardware Description Language (VHDL), arithmetic circuits, computer design fundamentals, CMOS and TTL technologies.

Overview

Information Representation

Introduction to digital circuits

Number systems

Binary numbers

Number conversion between bases

Binary codes

Logical gates

Boolean functions

Reading: Mano - Chapter 1, Chapter 2, 2.1-2.2



UNSW
SYDNEY

Information representation

Information represents physical parameters or man-made parameters

Known as **signals**

Most physical parameters are continuous- can take all possible values over a defined range

Example - temperature, humidity

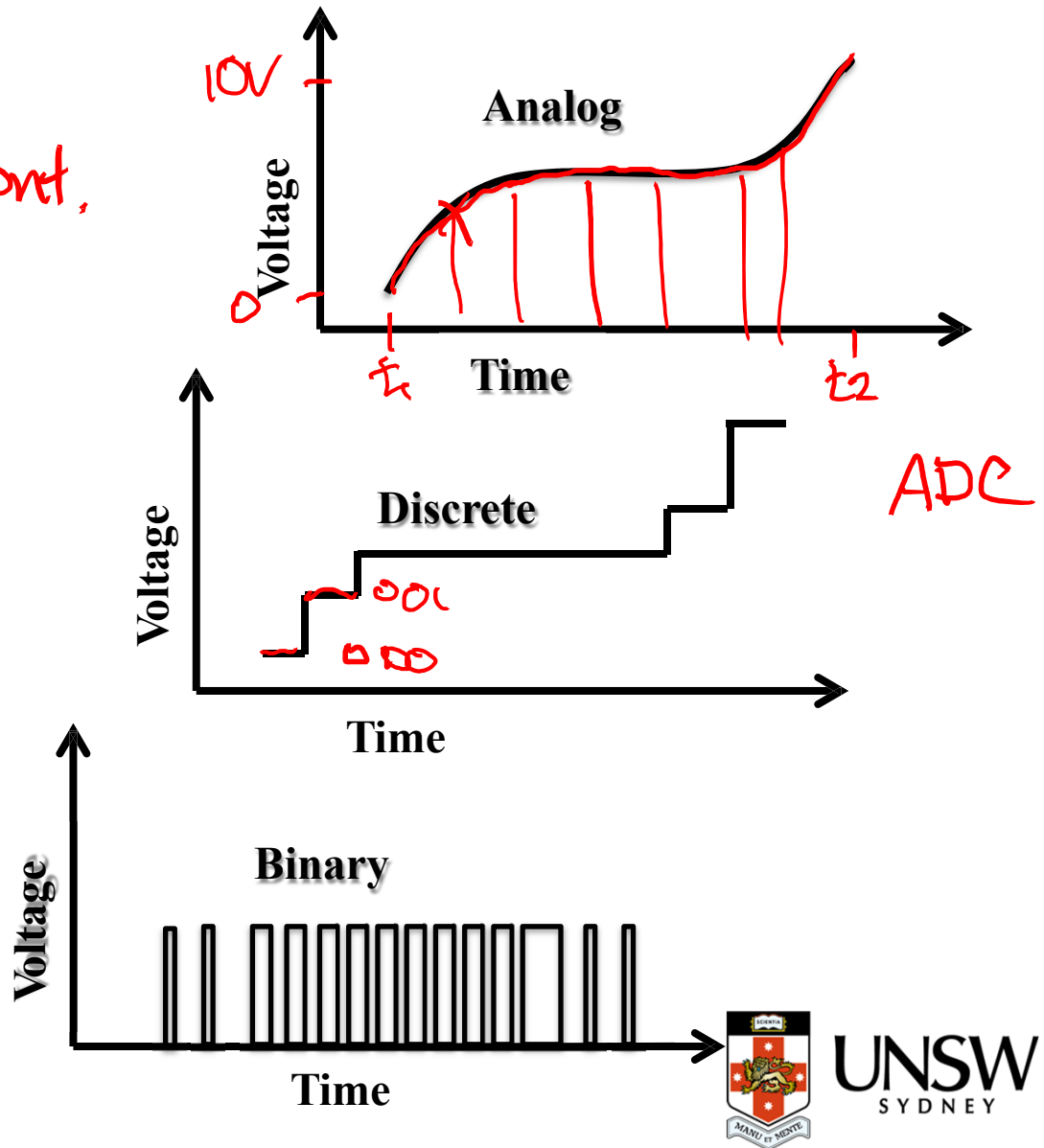
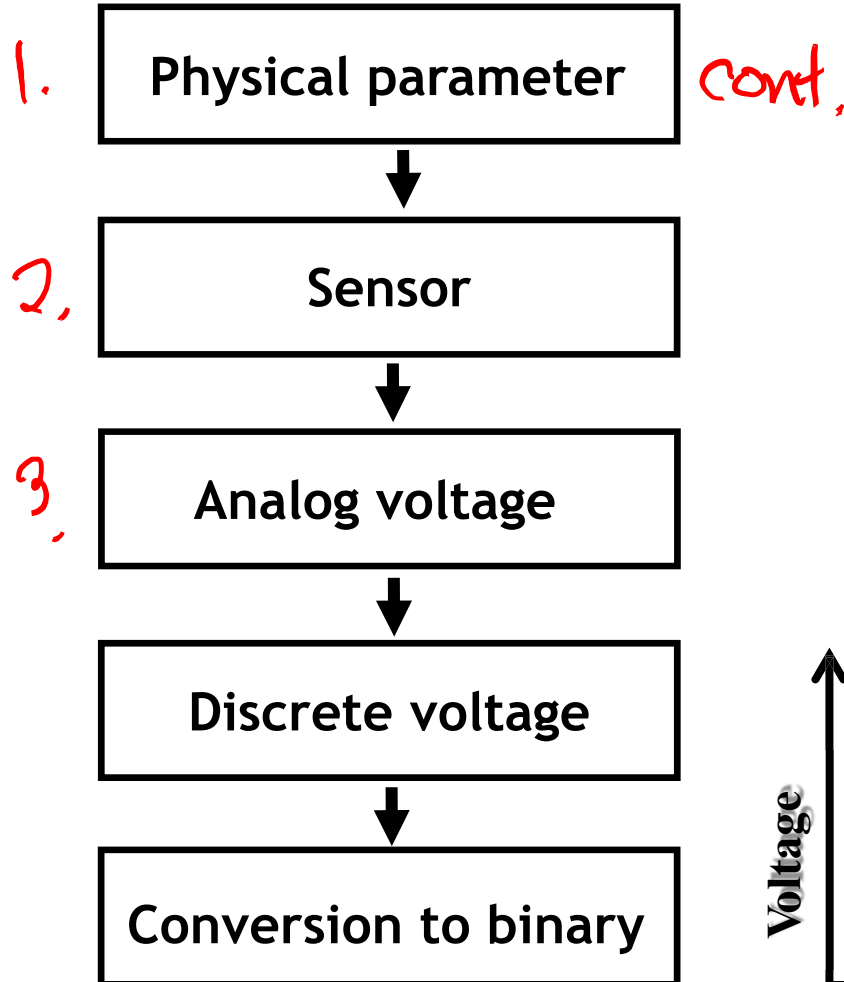
Man-made parameters are discrete - can take only finite values over a defined range

Example - currency



UNSW
SYDNEY

Signal digitisation



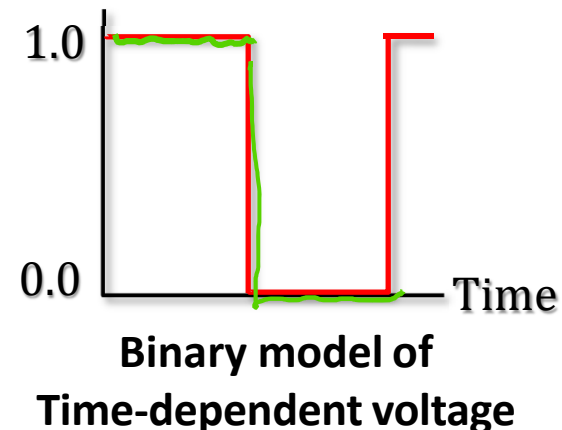
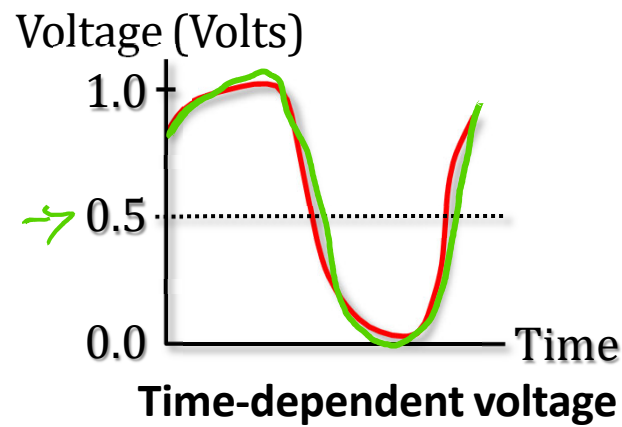
Digital circuits

Digital Circuits can understand (manipulate and store) signals as only binary

Signals in digital circuits are in fact analog, but interpreted as **binary**

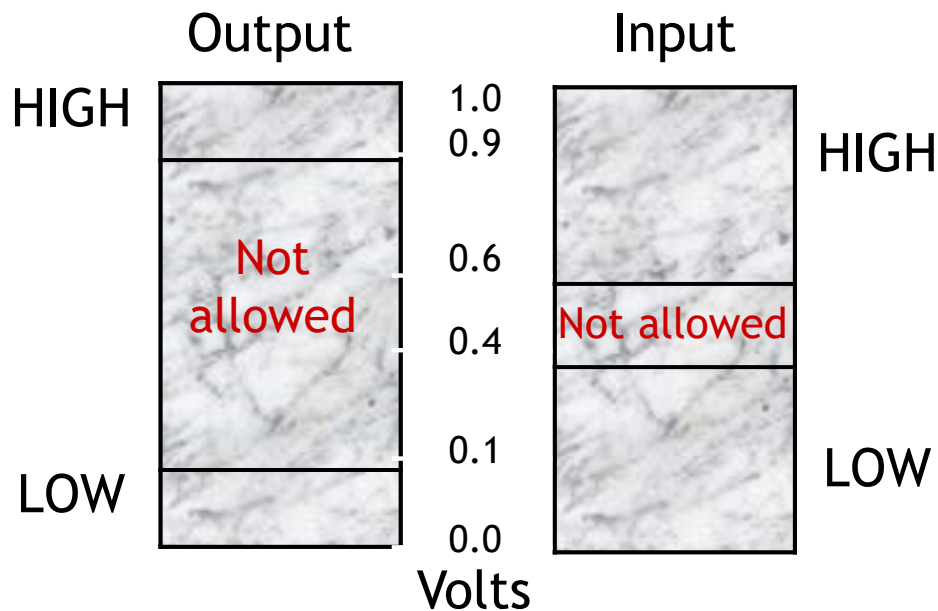
Binary signals can have amplitudes in only two ranges

The information associated with these ranges is represented by the voltage values of **HIGH/LOW**



Binary signals

Binary signals are identified as **HIGH/1/TRUE/On** or **LOW/0/FALSE/Off** based on their range of values



Output [0.9, 1.0] - HIGH

Output [0, 0.1] - LOW

Input [0.6, 1.0] - HIGH

Input [0, 0.4] - LOW

Example voltage ranges

Noise margin - difference in range of values for input and output



UNSW
SYDNEY

Positive vs negative logic

Usually, the higher amplitude represents 1 and the lower amplitude represents 0 (called **positive logic**)

In **negative logic**, the lower amplitude represents 1 and the higher amplitude represents 0

The logic value 1 is also called **high**, **true**, or **on**

The logic value 0 is also called **low**, **false**, or **off**

Digital voltage values

transistor-transistor logic

complementary metal oxide semiconductor

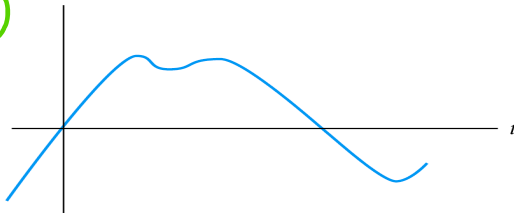
	TTL		<u>3.3 V</u> CMOS		<u>5V</u> CMOS	
	LOW	HIGH	LOW	HIGH	LOW	HIGH
INPUT	0 - 0.8	2 - 5	0 - 0.8	2 - 3.3	0 - 1.5	2.7 - 5
OUTPUT	0 - 0.5	2.7 - 5	0 - 0.4	2.4 - 3.3	0 - 0.5	3.5 - 5
Noise Margin	0.3	0.7	0.4	0.4	1.0	0.8

Advantages of digital signals

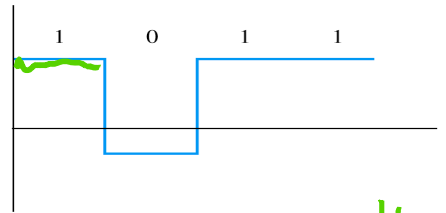
Even when noise in signal is large, logic values can still be determined in presence of noise unlike analog signals

Also significantly more economical

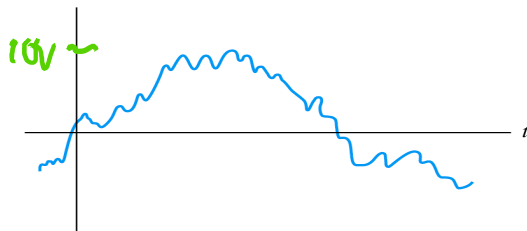
①



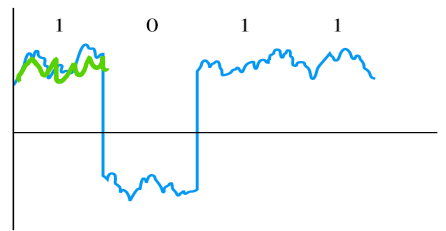
(a) Analog signal



(b) Digital signal

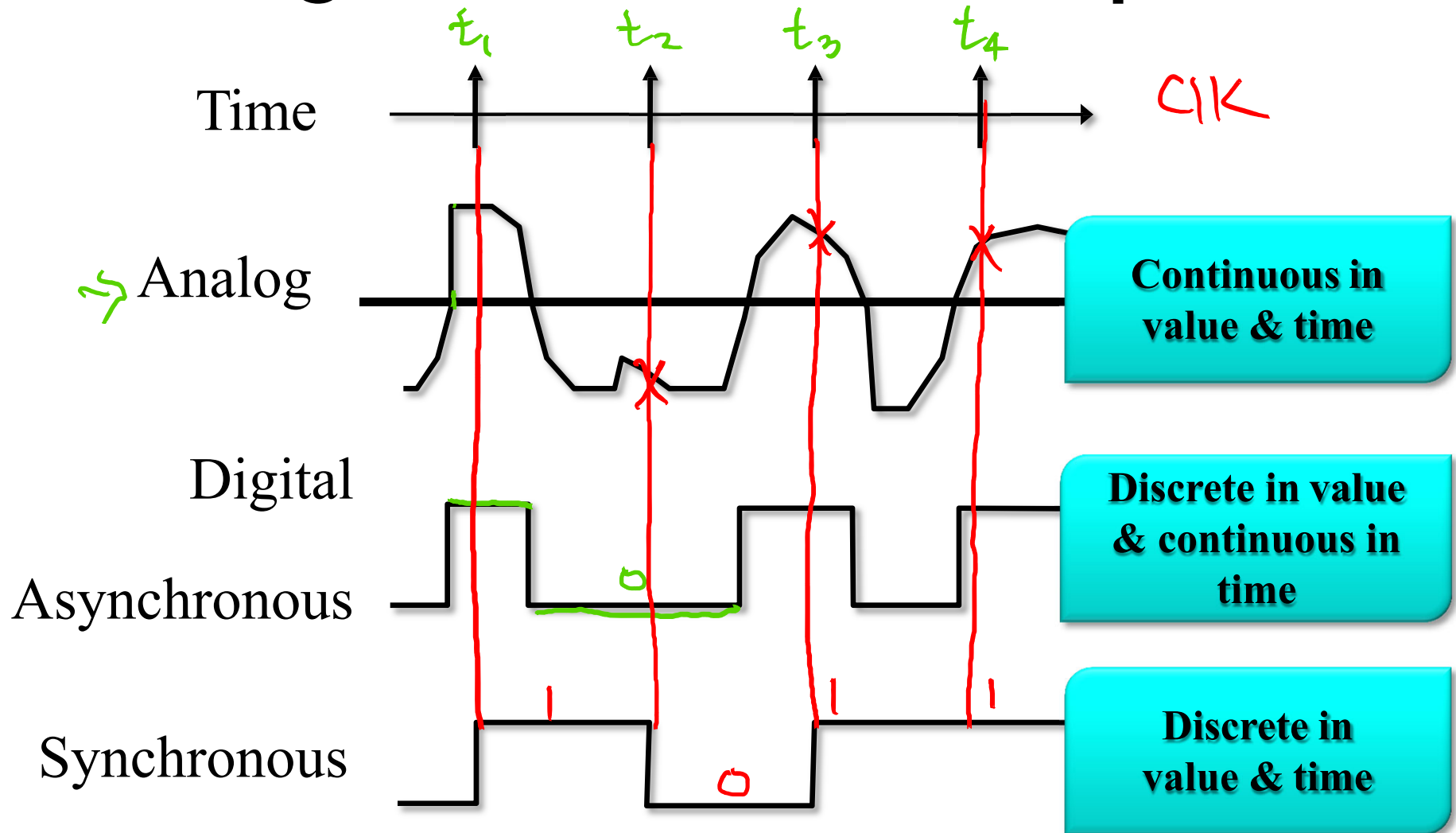


(c) Analog signal plus noise



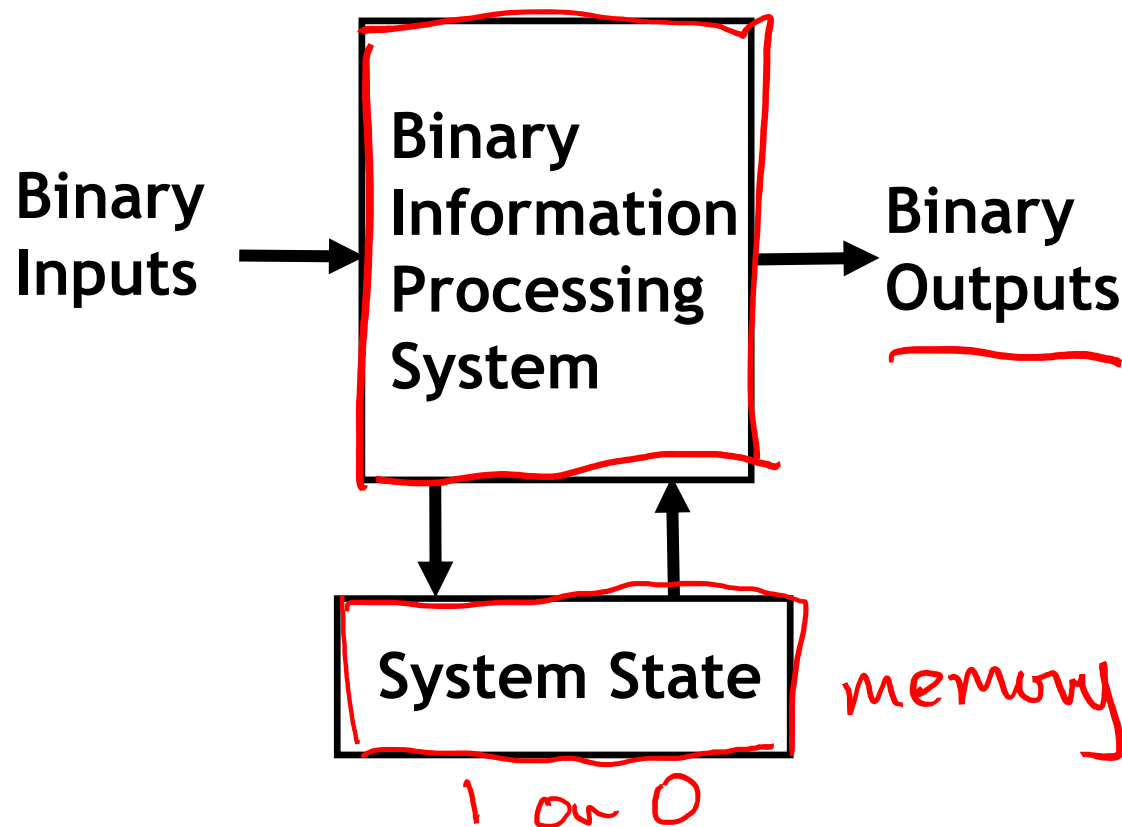
(d) Digital signal plus noise

Signal over time example



Digital circuits

Digital circuits have binary inputs, process binary signals, store binary signals (states), and generate binary outputs.



Types of digital circuits

Combinational logic circuits

No state present

Sequential logic circuits

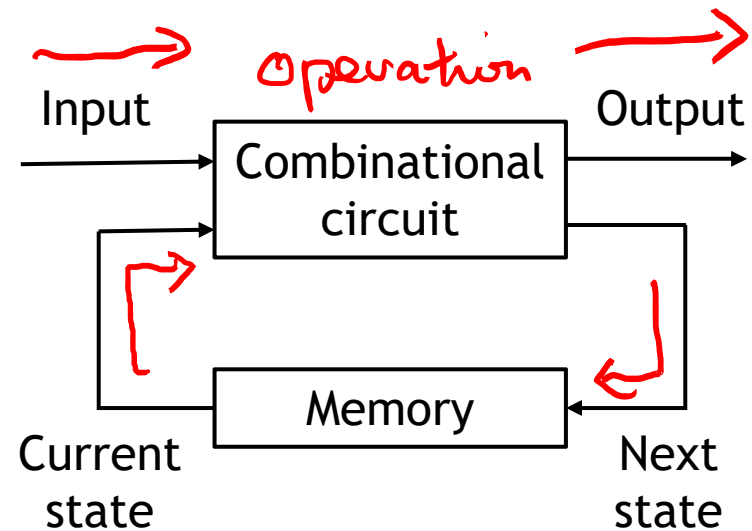
State present

State updated at discrete times

=> Synchronous Sequential circuits

State updated at any time

=> Asynchronous Sequential circuits



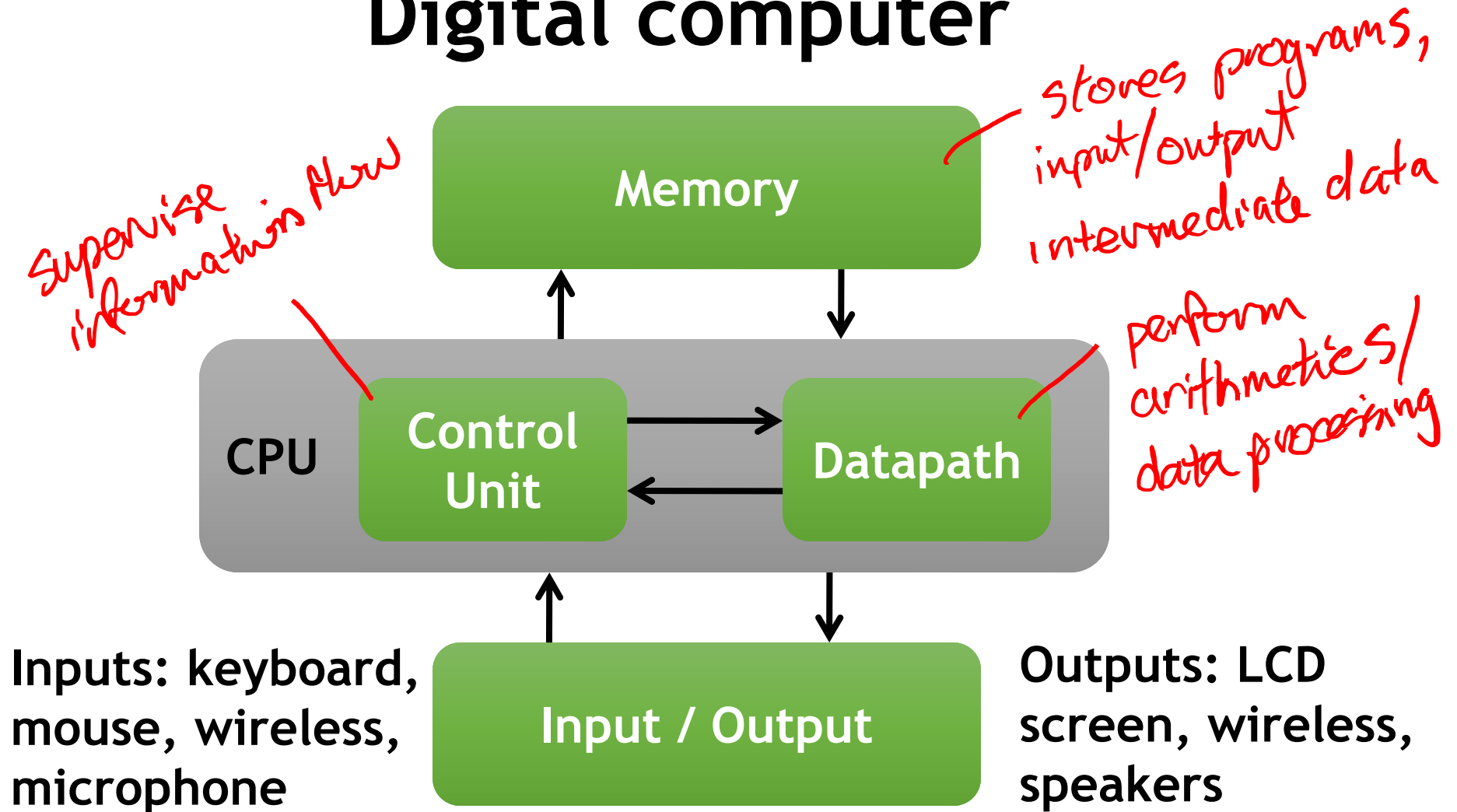
Digital system example

Digital Counter



Inputs: Count Up¹, Reset⁰
Outputs: Visual Display
State: “Value” of stored digits

Digital computer

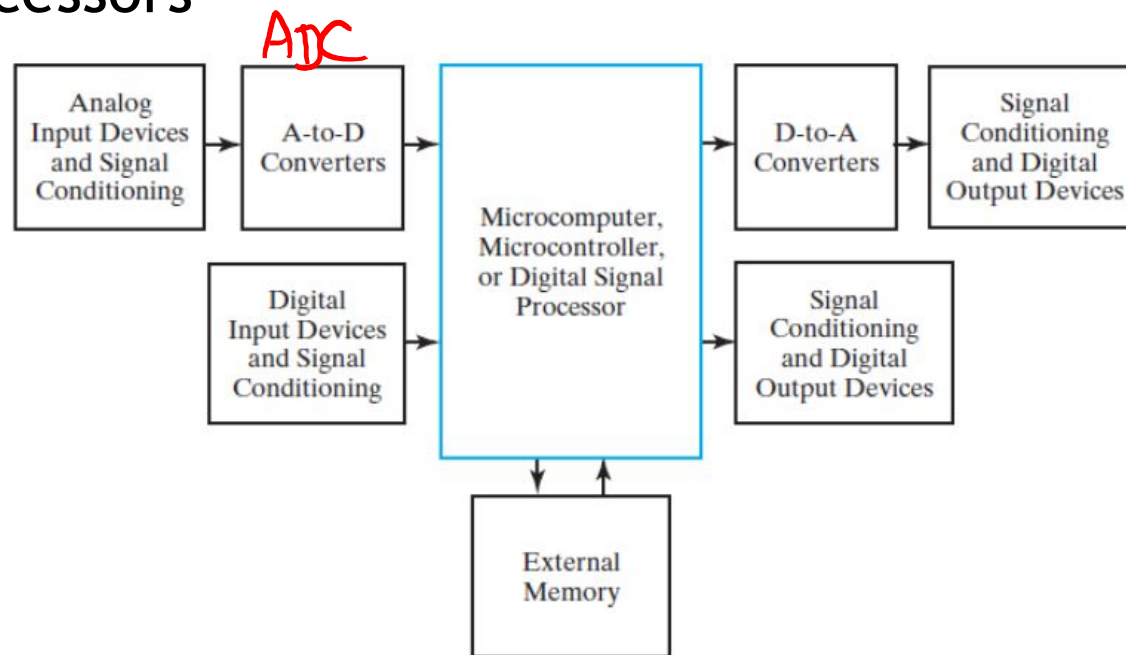


And beyond - embedded systems

Computers as integral parts of other products


Examples of embedded computers

Microcomputers, microcontrollers, digital signal processors

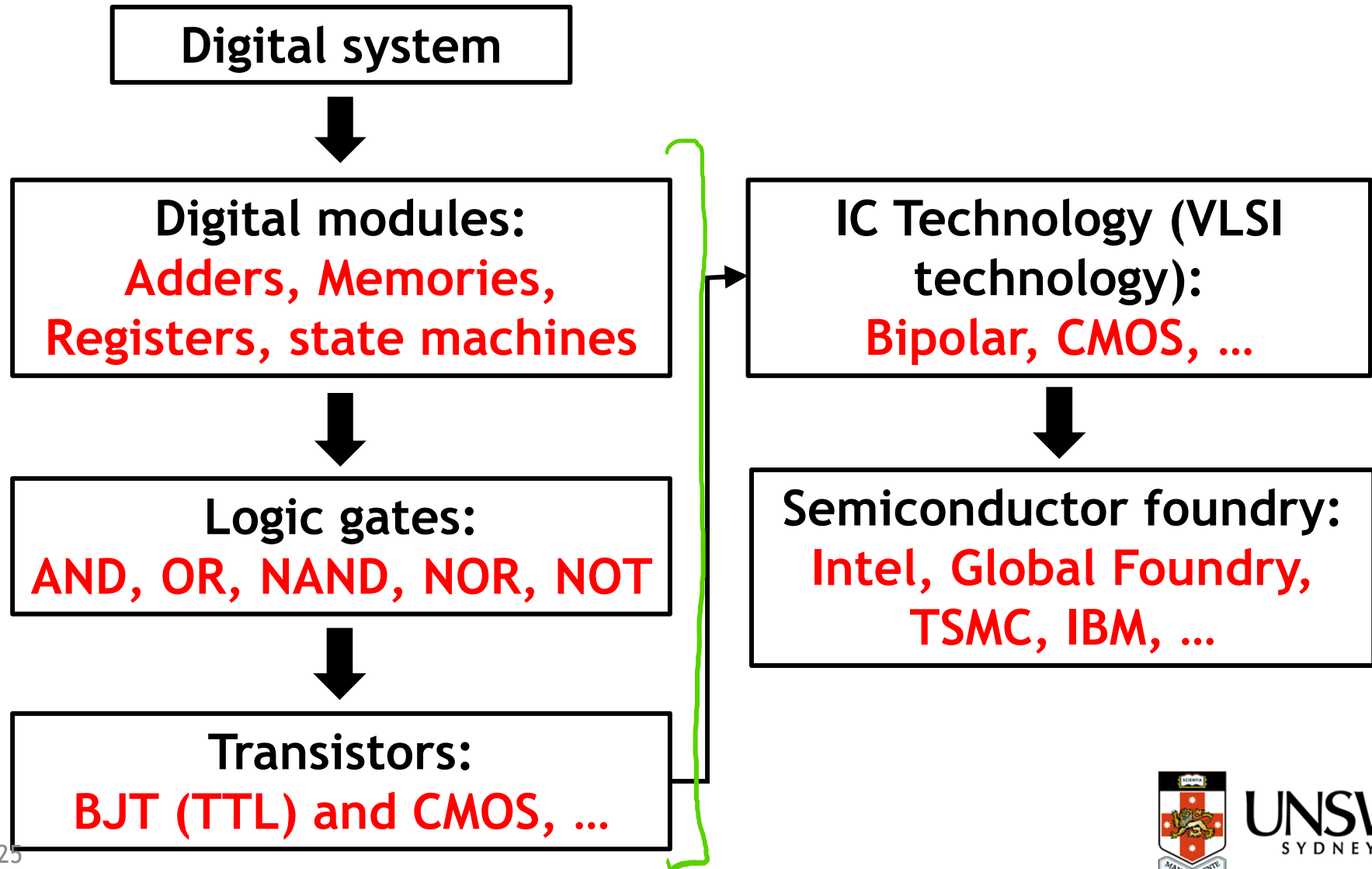


Binary physical quantities

Examples of some physical representations for binary 0 and 1 include

CPU:	<i>Voltage</i>	Hi / Lo
Floppy Disk:	<i>Magnetic Field Direction</i>	↑ ↓ ← →
CD:	<i>Surface Pits</i>	
Dynamic RAM:	<i>Electrical Charge</i>	

Hierarchy in a digital system



Number systems

A decimal number 724.5_{10} can be represented as

$$724.5_{10} = \underline{7 \times 10^2} + \underline{2 \times 10^1} + \underline{4 \times 10^0} + \underline{5 \times 10^{-1}}$$

A decimal number with n digits to the left of the decimal point and m digits to the right is represented as

$$A_{n-1}A_{n-2}\dots A_1A_0.A_{-1}A_{-2}\dots A_{-m+1}A_{-m} = A_{n-1}10^{n-1} + A_{n-2}10^{n-2} + \dots + A_010^0 + A_{-1}10^{-1} + \dots + A_{-m}10^{-m}$$

$A_i = 0, 1, 2, 3, 4, 5, 6, 7, 8$ or 9

A number in **base or radix** r with n -digits to the left of the radix point and m -digits to the right is represented by a string of coefficients and expressed as a power series in r

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

Binary numbers

Binary (base 2) numbers are widely used in digital systems

Base 2 system only has two possible digits:

0 and 1

Digits in binary numbers are called *bits* (Binary Digits)

The rightmost bit in a binary number is referred to as the *Least Significant Bit (LSB)*

Similarly, the leftmost bit is referred to as the *Most Significant Bit (MSB)*

1 0 0 1 1
MSB LSB

Binary to decimal example

Example:

$$\begin{array}{ccccccc} & 4 & 3 & 2 & 1 & 0 & \\ & \cdot & \cdot & \cdot & \cdot & \cdot & \\ 1 & 1 & 0 & 1 & 0 & & \\ & \cdot & \cdot & \cdot & \cdot & \cdot & \end{array} \quad \begin{array}{l} 11010_2 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \\ = 16 + 8 + 0 + 2 + 0 \\ = 26_{10} \end{array}$$

In practice, when calculating binary values, omit all 0-bits and add values for 1-bits

Example (with fractions):

$$\begin{array}{ccccccccccc} & 5 & 4 & 3 & 2 & 1 & 0 & -1 & -2 & & \\ & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & & \\ 1 & 1 & 0 & 1 & 0 & 1 & . & 1 & 1 & & \\ & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & & \end{array} \quad \begin{array}{l} 110101.11_2 = 1 \times 2^5 + 1 \times 2^4 + 0 + 1 \times 2^2 + 0 + 1 + \\ \quad 1 \times 2^{-1} + 1 \times 2^{-2} \\ = 32 + 16 + 4 + 1 + 0.5 + 0.25 \\ = 53.75 \end{array}$$



Problem

Determine the decimal values of

$$\begin{aligned} 0111011110_2 &= 0 \times 2^9 + 1 \times 2^8 + 1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 \\ &\quad + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \\ &= 478_{10} \end{aligned}$$

$$\begin{aligned} 100101.101_2 &= 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 \\ &\quad + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} \\ &= 37.625_{10} \end{aligned}$$

Positive powers of 2

Useful for base conversion

Important to remember!

Exponent	Value
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1,024

Exponent	Value
11	2,048
12	4,096
13	8,192
14	16,384
15	32,768
16	65,536
17	131,072
18	262,144
19	524,288
20	1,048,576
21	2,097,152

Memorize if you can (and have too much time on your hands)



UNSW
SYDNEY

Special powers of 2 *In Binary Sys.*

2^{10} (1,024) is Kilo, denoted “K” *Kibi*

$10^6 = 1000000$

2^{20} (1,048,576) is Mega, denoted “M” *Mebi*

2^{30} (1,073,741,824) is Giga, denoted “G” *Gibi*

2^{40} (1,099,511,627,776) is Tera, denoted “T” *Tebi*

Decimal to binary integer

Repeatedly divide integers by 2 to obtain quotient and remainder

Read remainders in reverse order

Example - convert 41_{10} to binary:

$41/2 = 20$	rem <u>1</u>	LSB
$20/2 = 10$	rem 0	
$10/2 = 5$	rem 0	
$5/2 = 2$	rem 1	
$2/2 = \underline{1}$	rem 0	
	MSB	

$(101001)_2$

Another conversion example

Can convert decimal to any other base

Example - Convert 388_{10} to base 7:

$$388 / 7 = 55 \quad \text{rem } 3$$

$$55 / 7 = 7 \quad \text{rem } 6$$

$$7 / 7 = 1 \quad \text{rem } 0$$

$$(1063)_7$$

Problems

Determine the binary value of 455_{10}

$$\begin{array}{ll} 455/2 = 227 \text{ r } \underline{1} \text{ LSB} & 28/2 = 14 \text{ r } 0 \\ 227/2 = 113 \text{ r } 1 & 14/2 = 7 \text{ r } 0 \\ 113/2 = 56 \text{ r } 1 & 7/2 = 3 \text{ r } 1 \\ 56/2 = 28 \text{ r } 0 & 3/2 = \underline{1} \text{ r } 1 \text{ MSB} \end{array}$$

$$(111000111)_2$$

Determine the octal (8) value of 641_{10}

$$(1201)_8$$

$$\begin{array}{ll} 641/8 = 80 \text{ r } 1 \\ 80/8 = 10 \text{ r } 0 \\ 10/8 = 1 \text{ r } 2 \end{array}$$

Decimal fractions to binary

Repeatedly multiply fractions by 2 to obtain integer product and fraction

Read integer products in order

Example - convert 0.6875_{10} to binary: 0.1011

$$\begin{array}{lcl} 0.6875 \times 2 = \underline{1.375} & 1 \\ 0.375 \times 2 = \underline{0.75} & 0 \\ 0.75 \times 2 = \underline{1.5} & 1 \\ 0.5 \times 2 = \underline{1.0} & 1 \end{array}$$

Check

$$\begin{aligned} & 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} \\ & + 1 \times 2^{-4} \\ & = 0.5 + 0.125 + 0.0625 \\ & = 0.6875 \end{aligned}$$

Additional issue in 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: Convert 0.65_{10} to binary

$$0.65 = 0.101001\overline{1001001}\dots \quad 0.101$$

The fractional part begins repeating every 4 steps, yielding repeating 1001 forever!

Solution: Specify number of bits to right of radix point and round/truncate to this number



UNSW
SYDNEY

Problem

~~0101~~

Convert 341.23_{10} to binary with 6 bits in fractional part

$$\begin{array}{r} \underline{341.23} \\ \div 2 \quad \times 2 \end{array} = 101010101.001110$$

$0.23 \times 2 = 0.46$	0
$0.46 \times 2 = 0.92$	0
$0.92 \times 2 = 1.84$	1
$0.84 \times 2 = 1.68$	1
$0.68 \times 2 = 1.36$	1
$0.36 \times 2 = 0.72$	0

$341 / 2 = 170$	r1	LSB
$170 / 2 = 85$	r0	
$85 / 2 = 42$	r1	
$42 / 2 = 21$	r0	
$21 / 2 = 10$	r1	
$10 / 2 = 5$	r0	
$5 / 2 = 2$	r1	
$2 / 2 = 1$	r0	MSB



UNSW
SYDNEY

Conversion between bases

In general, to convert between bases, you need to convert to decimal first, and then from decimal to the target base (exceptions follow)

So, to convert from one base to another:

- 1) Convert the integer part
- 2) Convert the fraction part
- 3) Join the two results with a radix point

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

A → 10
B → 11
C → 12
D → 13
E → 14
F → 15

The six letters **A**, **B**, **C**, **D**, **E**, and **F** in hexadecimal represent the digits for values 10, 11, 12, 13, 14, 15 respectively

Usually, the prefix **0x** is added to indicate a hexadecimal number - e.g. 0xA5

Numbers in different bases

4 bit

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

$$2^3 = 8$$

$$2^4 = 16$$



UNSW
SYDNEY

Hexadecimal to binary and back

Hexadecimal (or Octal) to Binary:

Restate each hexadecimal (octal) digit as the corresponding four (three) bits starting at the radix point and going both ways

Binary to Hexadecimal (or Octal):

Group the binary digits into four (three) bit groups starting at the radix point and going both ways, padding with zeros as needed in the fractional part

Convert each group of four (three) bits to a hexadecimal (octal) digit

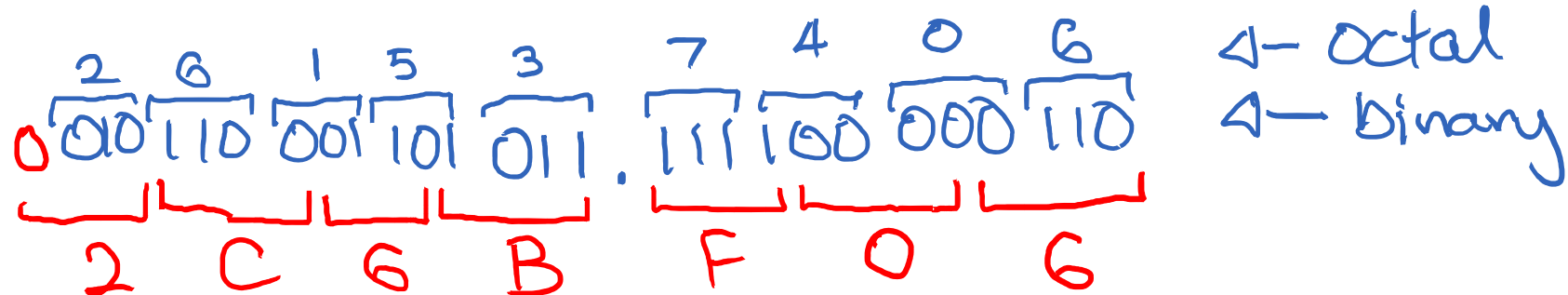
2	7	3	1
010	111	011	001
0101	1101	1001	
5	D	9	

2731 - octal

5D9 - HEXA

Octal to hexadecimal via binary

Example: Convert the octal number 26153.7406_8 to binary and therefore to hexadecimal



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

Problem

Convert $2B.5_{16}$ to binary and octal

Octal

Handwritten conversion process for $2B.5_{16}$ to binary and octal:

Hexadecimal: 2 B . 5

Binary: 000101011.010100

Octal: 0 5 3 . 2 4

The binary representation is grouped into octal digits: 000, 101, 011, 010, 100, which correspond to the octal digits 0, 5, 3, 2, and 4 respectively.

Binary coding

Flexibility of representation

Within the constraints below, can assign any binary combination (called a code word) to any data as long as data is uniquely encoded

Information Types

Numeric

Non-numeric

Non-numeric binary codes

Given n bits, a binary code is a mapping from a set of represented elements to a subset of the 2^n binary numbers

$2^n = \text{no. comb.}$

Example:

A binary code for the seven colors of the rainbow
Code 100 is not used

$2^3 = 8$

Color	Binary Code
Red	000
Orange	001
Yellow	010
Green	011
Blue	101
Indigo	110
Violet	111



UNSW
SYDNEY

Number of bits required

Given M elements to be represented by a binary code, the n minimum number of bits needed satisfies the following relationship:

$$2^{(n-1)} < M \leq 2^n \quad \text{or} \quad n \geq \lceil \log_2 M \rceil$$

Where this is the *ceiling function* - the integer greater than or equal to the argument

Example: How many bits are required to represent **decimal digits** with a binary code?

$$\begin{aligned} M &= 2^n \\ \log M &= \log 2^n \\ \log M &= n \log 2 \\ n &= \frac{\log M}{\log 2} \end{aligned}$$

$$\begin{aligned} n &= \frac{\log 7}{\log 2} \\ &= 2.8 \\ &\approx 3 \end{aligned}$$

$$3 \text{ bits} \Rightarrow 2^3 = 8$$

$$4 \text{ bits} \Rightarrow 2^4 = 16$$

Number of elements represented

Given n digits in radix r , there are r^n distinct elements that *can* be represented

But can represent less elements, m , such that:
 $m < r^n$

Examples:

You can represent 4 elements in radix $r = 2$ with
 $n = 2$ digits: 2 bits

$$2^2 = 4$$

You can represent 4 elements in radix $r = 2$ with
 $n = 4$ digits: 4 bits

$$2^4 = 16$$

Decimal codes - BCD and gray code

Two useful ways to code decimal digits into binary are *Binary Coded Decimal (BCD)* and *Gray Code*

Decimal	BCD	Gray Code
<u>0</u>	<u>0000</u>	0000
1	0001	0100
2	0010	0101
3	0011	0111
4	0100	0110
5	0101	0010
6	0110	0011
7	0111	0001
8	1000	1001
9	1001	1000

1 bit change



UNSW
SYDNEY

Binary coded decimal (BCD)

BCD 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

Every digit in a decimal number is encoded separately and then combined together

Example (BCD *coding*):

$$185_{10} = \underbrace{0001}_1 \underbrace{1000}_8 \underbrace{0101}_5$$

Compare with (binary *conversion*):

$$185_{10} = \underline{10111001}$$

Gray code

Gray code is a binary code where two successive values differ in only one bit change

Example - using the gray code assignments given earlier for decimal digits, only one bit changes on the transition from 3_{10} to 4_{10} :

$011\underline{1}_{GC} \rightarrow 011\underline{0}_{GC}$ 1 bit change

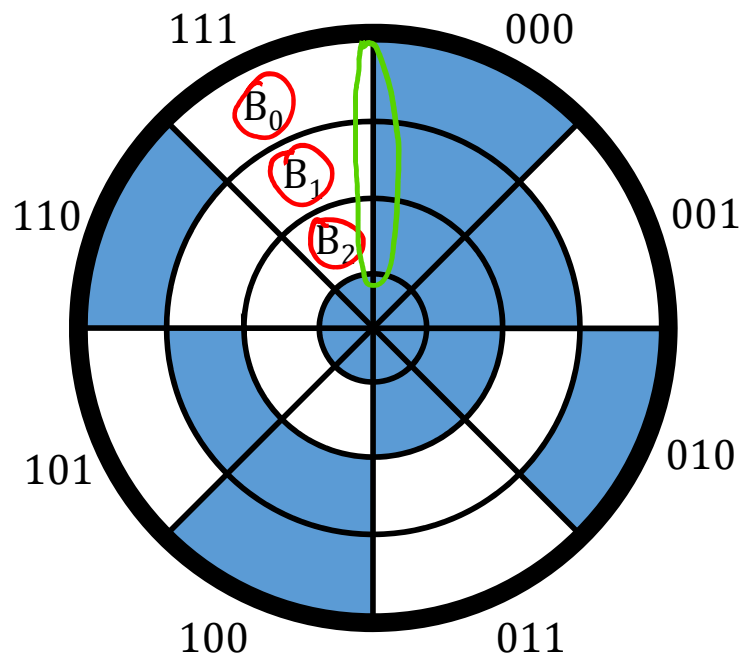
Compare with regular binary code, three bits change on the same transition:

011_2 100_2 3 bit change

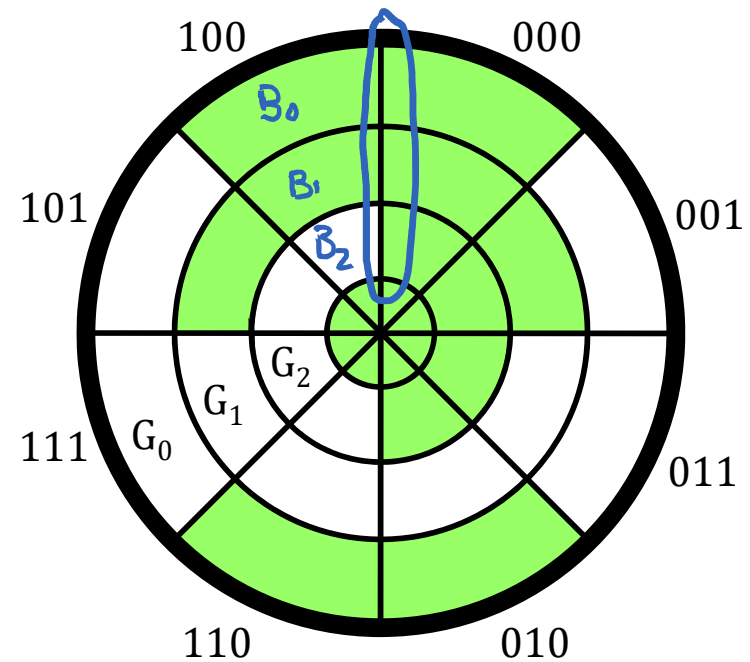
Optical shaft encoder using gray code

What is it good for?

An example: Optical Shaft Encoder



Binary Code



Gray Code



UNSW
SYDNEY

Alphanumeric codes

Digital systems need to handle both numeric and non-numeric (characters or symbols) data.

It is necessary to formulate a binary code for letters of the alphabets, numerals and special characters.

Alphanumeric character set is a set of elements that include the 10 decimal digits, the 26 letters of the alphabet (lower and upper case), and a number of special characters.

Alphanumeric codes - ASCII Code

64 to 128 elements in this character set.

Standard binary code for the alphanumeric characters is the American Standard Code for Information Interchange (ASCII), which uses seven bits to code 128 characters.

ASCII includes 10 numerals, 26 upper case letters, 26 lower case letters, 32 special printable characters such as %, @, and \$, and 34 non-printing characters.

instructions

Alphanumeric codes - ASCII Code

American Standard Code for Information Interchange (ASCII)

<u>B₄B₃B₂B₁</u>	<u>B₇B₆B₅</u>							
	000	<u>001</u>	010	<u>011</u>	<u>100</u>	101	110	111
<u>0000</u>	NULL	<u>DLE</u>	SP	0	@	P	`	p
<u>0001</u>	SOH	<u>DC1</u>	!	1	<u>A</u>	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	\$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(8	H	X	h	x
1001	HT	EM)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[k	{
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	-	=	M]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

Control Characters

NULL	NULL	DLE	Data link escape
SOH	Start of heading	DC1	Device control 1
STX	Start of text	DC2	Device control 2
ETX	End of text	DC3	Device control 3
EOT	End of transmission	DC4	Device control 4
ENQ	Enquiry	NAK	Negative acknowledge
ACK	Acknowledge	SYN	Synchronous idle
BEL	Bell	ETB	End of transmission block
BS	Backspace	CAN	Cancel
HT	Horizontal tab	EM	End of medium
LF	Line feed	SUB	Substitute
VT	Vertical tab	ESC	Escape
FF	Form feed	FS	File separator
CR	Carriage return	GS	Group separator
SO	Shift out	RS	Record separator
SI	Shift in	US	Unit separator
SP	Space	DEL	Delete



UNSW
SYDNEY

Binary logic and gates

Interconnected transistors form **logic gates** ^{2,4} transistors

Each gate has inputs and an output. It performs a **specific logical operation** on its binary inputs and provides a binary value at the output

Inputs and output of a logic gate are designated by alphabetical variables **A, B, C, ...**

These variables can assume only 1 or 0 values and are known as **binary variables**

Three basic logical operations : **AND, OR, and NOT**

AND gate

AND is represented by a dot or an absence of an operator

$$\underline{Z = X \cdot Y = XY = X \wedge Y}$$

Z is 1 if and only if $X = 1$ and $Y = 1$; otherwise $Z = 0$

input	
X	Y
0	0
0	1
1	0
1	1

$X \cdot Y$

$0 \cdot 0 = 0$

$0 \cdot 1 = 0$

$1 \cdot 0 = 0$

$1 \cdot 1 = 1$

Can be extended to more than two input binary variables

Also called **logical multiplication**

OR gate

OR is represented by a plus symbol or “v”

$$Z = X + Y = X \vee Y$$

Z is 1 if $X = 1$ or $Y = 1$; Z is 0 if $X = 0$ and $Y = 0$

$X + Y$	Z
$0 + 0 = 0$	<u>0</u>
$0 + 1 = 1$	<u>1</u>
$1 + 0 = 1$	<u>1</u>
$1 + 1 = 1$	<u>1</u>

OR logical operation can be extended to more than two input binary variables

Also called **logical addition**

NOT gate

NOT is represented by a bar over the variable

$$Z = \bar{X}$$

Z is 1 if $X = 0$ and Z is 0 if $X = 1$

X
$\bar{0} = 1$
$\bar{1} = 0$

Also called an **inverter**

Truth tables

A *Truth Table* is a tabular form that uniquely represents the relationship between the input variables of a function and its output

A function F that depends on n variables will have 2^n rows

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

OR		
X	Y	$X + Y$
0	0	0
0	1	1
1	0	1
1	1	1

NOT	
X	$Z = \bar{X}$
0	1
1	0

Problem

Building a Truth Table for 3 and 4 input variables

<i>X</i>	<i>Y</i>	<i>Z</i>	<i>F</i>
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

output

<i>W</i>	<i>X</i>	<i>Y</i>	<i>Z</i>	<i>F</i>
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

Logic gates

Logical gates are electrical circuits that implement logical operations

The input terminals accept voltage signals within allowable range (as a binary signal) and gives out at the output terminal a binary signal that also falls within the allowable range

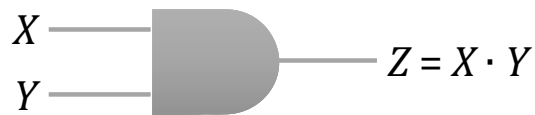
Intermediate values are crossed only during transitions from 0 to 1 or otherwise

0 \Rightarrow LO

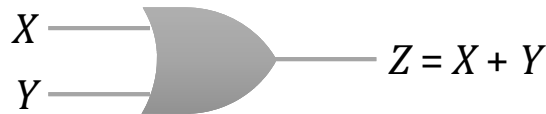
1 \Rightarrow HI

Graphical gate representation

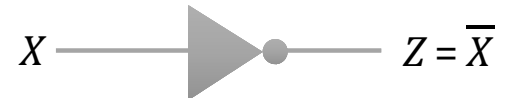
Graphical (symbolical) representations of the three types of gates: **AND**, **OR**, **NOT**



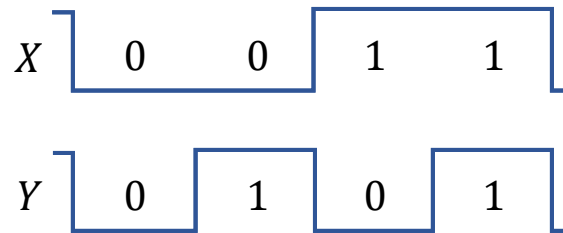
AND gate



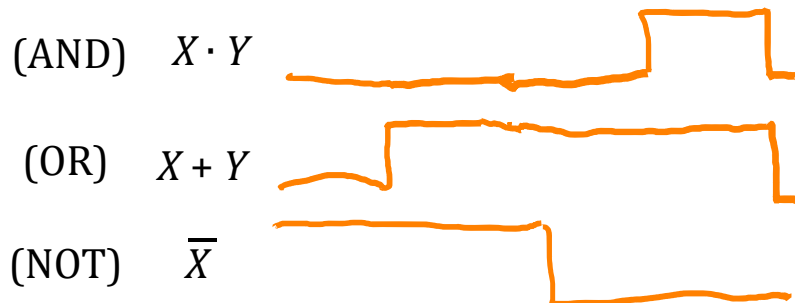
OR gate



NOT gate (inverter)



Time diagram shows waveform behavior



Boolean algebra

Boolean Algebra is an algebra dealing with binary variables and logic operations.

A *Boolean expression* is an algebraic expression formed by using binary variables, the constants 0 and 1, the logic operation symbols, and parentheses

The order of evaluation in a Boolean expression:

(), NOT, AND, OR

Boolean functions

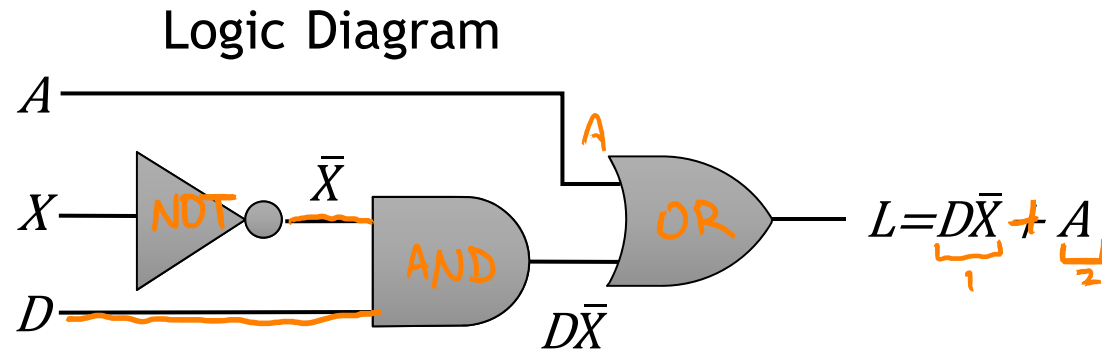
A **Boolean function** is a Boolean equation consisting of a binary variable identifying the function followed by an equal sign and Boolean expression

$$\underbrace{L(D, X, A)}_{\text{inputs}} = \overbrace{D\bar{X} + A}^{\text{Boolean expression}}$$

Boolean function or Boolean equation

A Boolean function can be transformed into a circuit diagram (logic diagram) composed of logic gates and interconnected by wires

Boolean functions



A Boolean function can be represented by a **truth table**

It uniquely represents the relationship between the input variables of a function and its output

A function F that depends on n variables will have 2^n rows

D	X	A	$L = D\bar{X} + A$	$D\bar{X}$
0	0	0	0	
0	0	1	1	
0	1	0	0	
0	1	1	1	
1	0	0	1	
1	0	1	1	
1	1	0	0	
1	1	1	1	



Boolean functions

The Boolean function, however, can be expressed by various Boolean equations, which are not the same but equivalent. e.g. L and F have the same function and truth table

$$L(D, X, A) = D\bar{X} + A$$

$$F(D, X, A) = D\bar{X} + AD + A\bar{D}$$

The Boolean equation dictates the interconnection of gates in the logic circuit diagram

Simpler expression reduces the number of gates and the number of inputs into the gates

Problem

Use a truth table to find F where

$$F(A, B, C) = A\bar{C} + C(A + B) + \overline{BC}$$

A	B	C	$A\bar{C}$	$A + B$	$C(A + B)$	BC	\overline{BC}	F
0	0	0	0	0	0	0	1	1
0	0	1	0	0	0	0	1	1
0	1	0	0	1	0	0	1	1
0	1	1	0	1	1	1	0	1
1	0	0	1	1	0	0	1	1
1	0	1	0	1	1	0	1	1
1	1	0	1	1	0	0	1	1
1	1	1	0	1	1	1	0	1

Combinational logic circuit

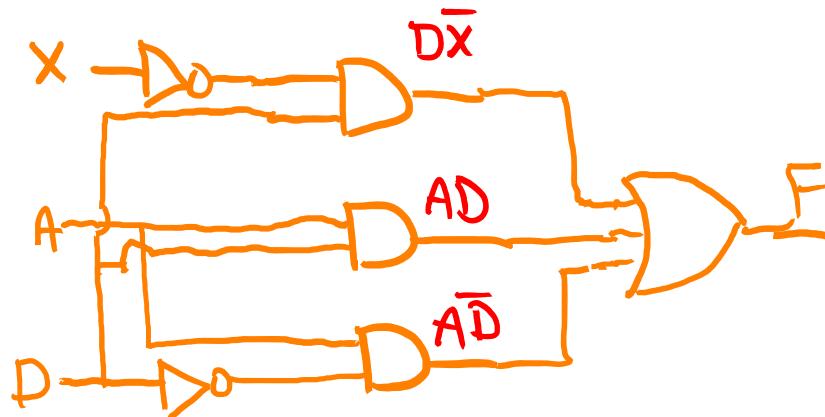
$$F(D, X, A) = \underline{D\bar{X}} + \underline{AD} + \underline{A\bar{D}}$$

A combinational logic circuit can be constructed to implement a Boolean function F , by appropriately connecting input signals and logic gates:

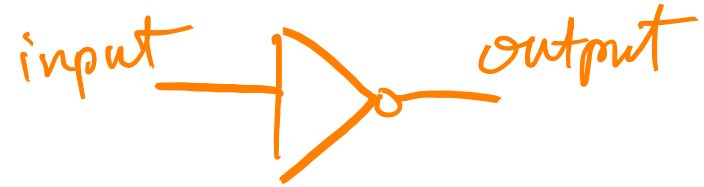
Circuit input signals \rightarrow from function variables (D, X, A)

Circuit output signal \rightarrow function output F

Logic gates \rightarrow from logic operations

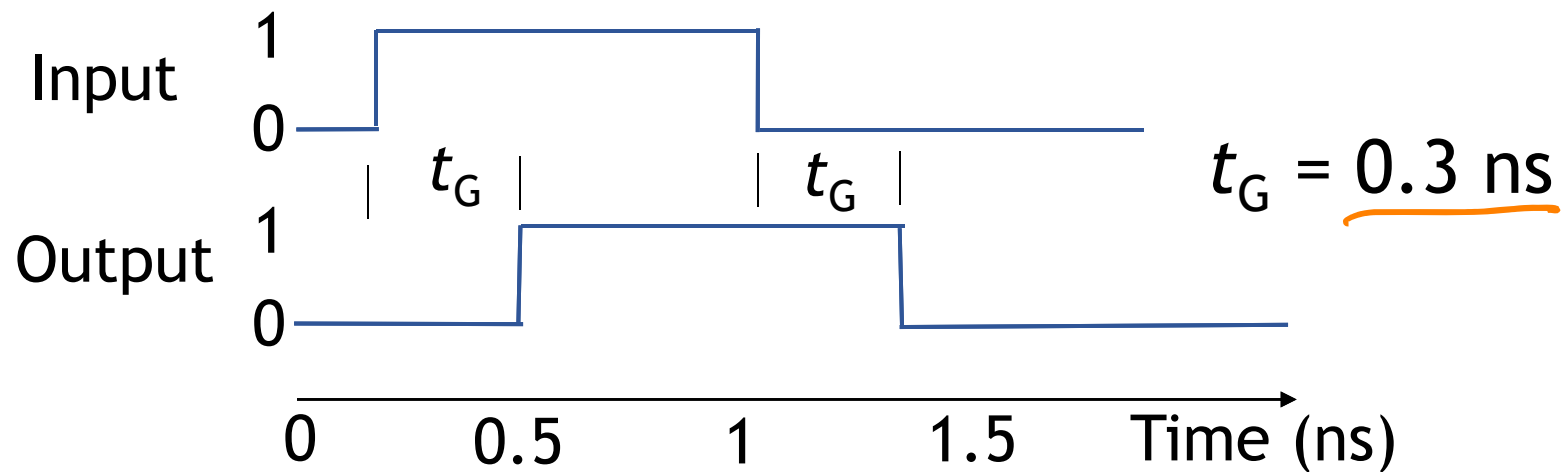


Gate delay



In reality, there is always a gate delay

Gate delay - the length of time it takes for an input changes to result in the corresponding output change



Gate delay is a function of **gate type**, **number of inputs**, **underlying technology**, and **circuit design of the gate**.

Basic identities of Boolean algebra

Dual →

1. $X + 0 = \underline{X}$	2. $X \cdot \underline{1} = \underline{X}$	Identity
3. $X + 1 = 1$	4. $X \cdot 0 = 0$	Null
5. $\overset{1/0}{X} + \overset{1/0}{X} = \overset{1/0}{X}$	6. $\overset{0/1}{X} \cdot \overset{0/1}{X} = \overset{0/1}{X}$	Idempotence
7. $X + \bar{X} = 1$	8. $X \cdot \bar{X} = 0$	Complementarity
9. $\bar{\bar{X}} = X$		Involution

OR

	X	Y	$X + Y$
→	0	0	0
	0	1	1
→	1	0	1
	1	1	1

AND

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



UNSW
SYDNEY

Basic identities of Boolean algebra

10. $Y + X = X + Y$	11. $Y \cdot X = X \cdot Y$	Commutative
12. $X + (Y + Z) = (X + Y) + Z$	13. $(XY)Z = X(YZ)$	Associative
14. $X(Y + Z) = XY + XZ$	15. $X + YZ = (X + Y)(X + Z)$	Distributive ✓
16. $\overline{X + Y} = \overline{X}\overline{Y}$	17. $\overline{XY} = \overline{X} + \overline{Y}$	DeMorgan's law

OR

X	Y	$X + Y$
0	0	0
0	1	1
1	0	1
1	1	1

AND

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



UNSW
SYDNEY

$$\begin{aligned}
 X + YZ &= X \cdot 1 + YZ \\
 &= X(1 + Y) + YZ \\
 &= X \cdot 1 + XY + YZ \\
 &= X(\underline{1} + Z) + XY + YZ
 \end{aligned}$$

$$X \cdot X = X \cdot 1 = X$$

$$X(X + Z) + Y(X + Z)$$

$$\underline{\underline{(X + Y)(X + Z)}}$$

Basic identities of Boolean algebra

Any expression can replace the variable X in all Boolean identities

e.g. $\underline{X} + 1 = 1$ then with $\underline{X = AB + C}$

$$\Rightarrow AB + C + 1 = 1$$

Identity 10 -14 are similar to ordinary algebra. However, Identity 15 does not hold in ordinary algebra. Identity 16 and 17 are the very important DeMorgan's rules

X	Y	$\bar{X} + \bar{Y}$	\overline{XY}	$\overline{X + Y}$	$\bar{X}\bar{Y}$
0	0	1	1	1	1
1	0	1	1	0	0
0	1	1	1	0	0
1	1	0	0	0	0

proof by
perfect induction

Basic identities of boolean algebra

DeMorgan's law can be extended to three or more variables. The general DeMorgan's theorem can be expressed as

$$\overline{X_1 + X_2 + \dots + X_n} = \overline{X_1} \overline{X_2} \dots \overline{X_n}$$
$$\overline{X_1 X_2 \dots X_n} = \overline{X_1} + \overline{X_2} + \dots + \overline{X_n}$$

Duality

The **dual** of an algebraic expression is obtained by interchanging + and · and interchanging the constants 0's and 1's

Any Boolean theorem that can be proven is thus also proven for its dual!

Previous identities appear in dual pairs

Example:

$$XY + \bar{X}Z + YZ = XY + \bar{X}Z$$

Dual $(X + Y)(\bar{X} + Z)(Y + Z) = (X + Y)(\bar{X} + Z)$

Problem

$$\overline{X+Y} = \bar{X}\bar{Y}$$

$$\overline{XY} = \bar{X} + \bar{Y}$$

Prove that the dual of

$$\underbrace{XY + \bar{X}Z + YZ}_F \text{ is } \underbrace{(X + Y)(\bar{X} + Z)(Y + Z)}_G$$

$$\begin{aligned}\bar{F} &= \overline{XY + \bar{X}Z + YZ} \\ &= (\overline{XY})(\overline{\bar{X}Z})(\overline{YZ}) \\ &= (\bar{X} + \bar{Y})(X + \bar{Z})(\bar{Y} + \bar{Z})\end{aligned}$$

①

$$\begin{aligned}\bar{G} &= \overline{(X+Y)(\bar{X}+Z)(Y+Z)} \\ &= (\overline{X+Y}) + (\overline{\bar{X}+Z}) + (\overline{Y+Z}) \\ &= \bar{X} \cdot \bar{Y} + X\bar{Z} + \bar{Y}\bar{Z}\end{aligned}$$

②

Useful theorems of Boolean algebra

Dual



$X + XY = X$	$X(X + Y) = X$	Absorption
$XY + X\bar{Y} = X$	$(X + Y)(X + \bar{Y}) = X$	Minimization
$X + \bar{X}Y = X + Y$	$X(\bar{X} + Y) = XY$	Simplification
$XY + \bar{X}Z + YZ = XY + \bar{X}Z$	$(X + Y)(\bar{X} + Z)(Y + Z) = (X + Y)(\bar{X} + Z)$	Consensus

Prove of absorption theorem

$$X + XY = X$$

$$X(X + Y) = X$$

$$\begin{aligned} X + XY &= X \cdot 1 + XY \\ &= X(1 + Y) \\ &= X \end{aligned}$$

$$\begin{aligned} X \cdot X + XY &= X + XY \\ &= X(1 + Y) \\ &= X \end{aligned}$$

$$[X \cdot X = X]$$

Prove of minimization theorem

$$XY + X\bar{Y} = X$$

$$(X + Y)(X + \bar{Y}) = X$$

$$X(Y + \bar{Y})$$

$$= X$$

$$XX + X\bar{Y} + YX + Y\bar{X}$$

$$= X + X\bar{Y} + YX$$

$$= X(1 + \bar{Y} + Y)$$

$$= X$$

Prove of simplification theorem

$$X + \bar{X}Y = X + Y$$

$$X(\bar{X} + Y) = XY$$

$$\begin{aligned} & X(X + \bar{X}) + \bar{X}Y \\ &= XX + \bar{X}X + \bar{X}Y \\ &= (X + \bar{X})(X + Y) \\ &= 1 \cdot (X + Y) \\ &= X + Y \end{aligned}$$

$$\begin{aligned} & \overset{0}{X\bar{X}} + XY \\ &= XY \end{aligned}$$

Prove of consensus theorem

$$XY + \bar{X}Z + YZ = \underline{XY + \bar{X}Z}$$

$$(X + Y)(\bar{X} + Z)(Y + Z) = (X + Y)(\bar{X} + Z)$$

$$XY + \bar{X}Z + YZ = XY + \bar{X}Z + YZ \cdot 1$$

$$= XY + \bar{X}Z + YZ(X + \bar{X})$$

$$= XY + \bar{X}Z + XYZ + \bar{X}YZ$$

$$= XY + \overset{\text{red arrow}}{XYZ} + \bar{X}Z + \overset{\text{red arrow}}{\bar{X}YZ}$$

$$= XY(1 + Z) + \bar{X}Z(1 + Y)$$

$$= \underline{XY + \bar{X}Z}$$

Expression simplification

A *literal* is a complemented or uncomplemented variable in a term

Simplify the following expression to contain the smallest number of literals:

10 literals

$$F = \underbrace{A\bar{B}C}_3 + \underbrace{ABC}_3 + (\underbrace{C+D}_2)(\underbrace{\bar{D}+E}_2)$$

$$= AC(\cancel{\bar{B}} + \cancel{B}) + C\bar{D} + CE + \cancel{D}\bar{D} + DE$$

$$= AC + \underbrace{C\bar{D} + CE + DE}_{-6} = AC + C\bar{D} + DE$$

$$= C(A + \bar{D}) + DE \quad -5 \text{ literals}$$

Consensus

$$\begin{array}{l} XY + \bar{X}Z + YZ = XY + \bar{X}Z \\ \downarrow \quad \downarrow \quad \downarrow \\ DE + \bar{D}C + EC = DE + \bar{D}C \end{array}$$

Expression Simplification

Simplify the following expression to contain the smallest number of literals:

15 literals

$$AB + ACD + \bar{A}BD + AC\bar{D} + \bar{A}BCD$$

$$= ACD + AC\bar{D} + \bar{A}BD + \bar{A}BCD + AB$$

$$= AC(\cancel{D} + \cancel{\bar{D}}) + \bar{A}BD(\cancel{1} + \cancel{C}) + AB$$

$$= AC + \bar{A}BD + AB$$

$$= AC + B(\underbrace{A + \bar{A}D})$$

$$= AC + B(A + D)$$

$$= AC + AB + BD$$

5 literals

$$\text{Simplification theorem}$$
$$X + \bar{X}Y = X + Y$$

Complement of a function

The complement of a function F , \bar{F} , can be obtained in two ways:

By applying DeMorgan's theorem

By taking the dual of the function and complementing each literal

$$\begin{aligned}\overline{X+Y} &= \bar{X}\bar{Y} \\ \overline{XY} &= \bar{X} + \bar{Y}\end{aligned}$$

Example: $\bar{F} = \overline{\bar{X}Y\bar{Z} + \bar{X}\bar{Y}Z}$

$$\begin{aligned}&= \overline{(\bar{X}Y\bar{Z})} \overline{(\bar{X}\bar{Y}Z)} \\ &= (X + \bar{Y} + Z)(X + Y + \bar{Z})\end{aligned}$$

De Morgan's

$$\bar{F} = (X + \bar{Y} + Z)(X + Y + \bar{Z}) \quad \text{Dual}$$

Problem

Find the complement of the function below using both DeMorgan's theorem and the dual of the function

$$D = AC + \bar{B}C$$

$$\begin{aligned}\bar{D} &= \overline{AC + \bar{B}C} \\ &= (\bar{A}\bar{C})(\overline{\bar{B}C}) \\ &= (\bar{A} + \bar{C})(B + \bar{C})\end{aligned} \quad \left. \vphantom{\begin{aligned}\bar{D} &= \overline{AC + \bar{B}C} \\ &= (\bar{A}\bar{C})(\overline{\bar{B}C}) \\ &= (\bar{A} + \bar{C})(B + \bar{C})\end{aligned}} \right\} \text{De Morgan's}$$

$$\bar{D} = (\bar{A} + \bar{C})(B + \bar{C}) \quad \text{Dual}$$

Problem

$$\overline{XY} = \overline{X} + \overline{Y}$$

$$\overline{X+Y} = \overline{X}\overline{Y}$$

Find the complement of the function below using both DeMorgan's theorem and the dual of the function

$$D = AC + \bar{B}C + \bar{A}(\bar{B} + BC)$$

DeMorgan's

$$\begin{aligned}\bar{D} &= \overline{AC + \bar{B}C + \bar{A}(\bar{B} + BC)} \\ &= (\overline{AC})(\overline{\bar{B}C})(\overline{\bar{A}(\bar{B} + BC)}) \\ &= (\bar{A} + \bar{C})(B + \bar{C})(A + \overline{\bar{B} + BC}) \\ &= (\bar{A} + \bar{C})(B + \bar{C})(A + B(\bar{B} + \bar{C}))\end{aligned}$$

Dual

$$\bar{D} = (\bar{A} + \bar{C})(B + \bar{C})(A + B(\bar{B} + \bar{C}))$$

