Imperial College
London

Lecture 1: Overview & Data Respresentations

Aero 2 Signals & Systems (Part 2)
Digital Circuits

Professor Peter Cheung
Department of EEE, Imperial College London

(Slides based on Floyd & Tocci)
(Floyd 2.1-2.4, 2.8, 2.10-2.11)
(Tocci 2.1 – 2.8)

Imperial College
London

Aim

- To give a first course in **digital electronics** providing you with both the knowledge and skills required to design simple digital circuits and understand about the digital world.

Imperial College
London

Objectives

- to impart to you a **formalism of logic** enabling you to **analyse logical processes**
- to enable you to implement **simple logical operations** using combinational logic circuits
- to enable you to understand common forms of **number representation** in digital electronic circuits and to be able to convert between different representations
- to enable you to understand the logical operation of simple **arithmetic** and other **MSI circuits (Medium Scale Integrated Circuits)**
- to impart to you the concepts of **sequential circuits** enabling you to analyse sequential systems in terms of **state machines**
- to enable you to **implement synchronous state machines** using flip-flops

Imperial College
London

Course Content

- 10 Lectures
  1. Overview & Data Representation
  2. Gates & Boolean Logic
  3. Boolean Algebra
  4. Karnaugh Map
  5. More Gates
  6. Arithmetic
  7. ROMs & PLDs
  8. Flip-Flops
  9. Counters
  10. Finite State machines

Tutorial Questions

- accompany each lecture
- a chance to practice the techniques studied
- graded according to difficulty:
  - * easy, only a little interesting
  - ** harder, more interesting
  - *** challenge, very interesting
- completion of all * and ** questions is essential to your success
- completion of *** questions indicates a very good understanding
- answers given out shortly after questions
- not assessed

Lectures

- Ten lectures of about 50 minutes each
  - copies of the overhead slides given out
  - some blanks in the slides for you to fill in, for example:
  - the truth table for an AND gate is:

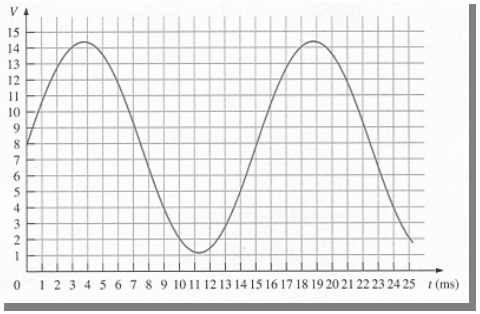
X	Y	Z=X.Y
0	0	0
0	1	0
1	0	0
1	1	1

- references given to the course book as we go along
- you are expected to read the relevant sections of the book as “homework” just after the lecture

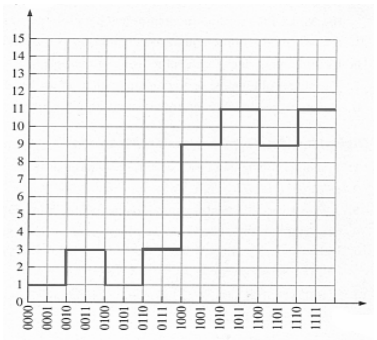
Text Books

- “Digital Systems – Principles and Applications”, 9th Ed, R. J. Tocci, N. S. Widmer, G. Moss, Pearson, 2004 (~£45)
- “Digital Fundamentals with PLD Programming”, T.L. Floyd, Prentice Hall, June 2005 (~£45)

Digital and Analog Quantities



Analog quantities have continuous values



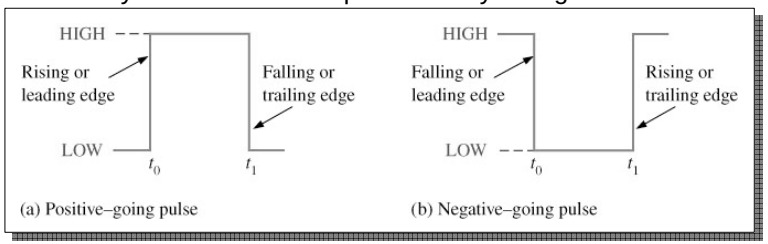
Digital quantities have discrete sets of values

Binary Digits, Logic Levels, and Digital Waveforms

- The conventional numbering system uses ten digits: 0,1,2,3,4,5,6,7,8, and 9.
- The binary numbering system uses just two digits: **0** and **1.**
- The two binary digits are designated **0** and **1**
- They can also be called LOW and HIGH, where **LOW = 0** and **HIGH = 1**

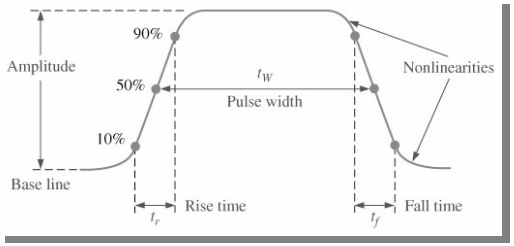
Binary Digits, Logic Levels, and Digital Waveforms

Binary values are also represented by voltage levels



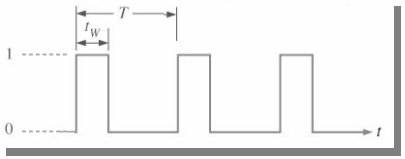
Major parts of a digital pulse

- Base line
- Amplitude
- Rise time (t_r)
- Pulse width (t_w)
- Fall time (t_f)



Binary Digits, Logic Levels, and Digital Waveforms

- t_w = pulse width
- T = period of the waveform
- f = frequency of the waveform



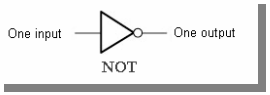
$$\mathbf{f = \frac{1}{T}}$$

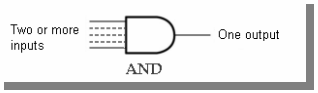
The duty cycle of a binary waveform is defined as:

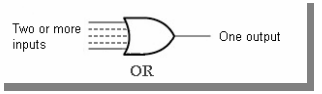
$$\mathbf{Duty\ cycle = \left(\frac{t_W}{T}\right)100\%}$$

Basic Logic Operations

There are only three basic logic operations:

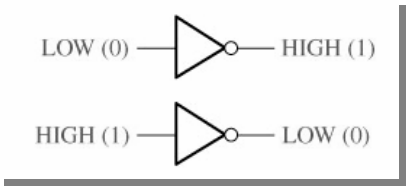






Basic Logic Operations

The NOT operation

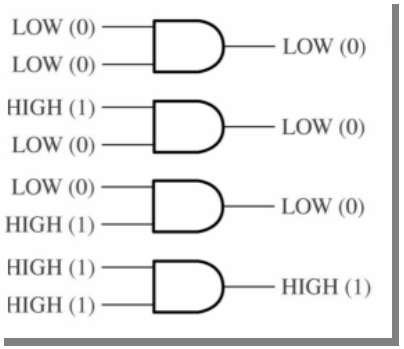


- When the input is LOW, the output is HIGH
- When the input is HIGH, the output is LOW

The output logic level is always opposite the input logic level.

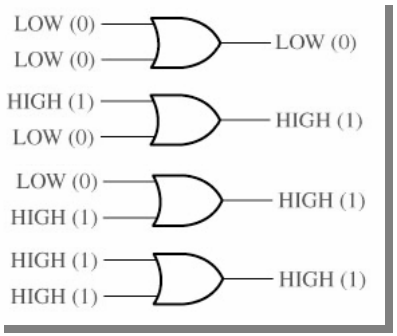
Basic Logic Operations

- The AND operation
  - When any input is LOW, the output is LOW
  - When both inputs are HIGH, the output is HIGH



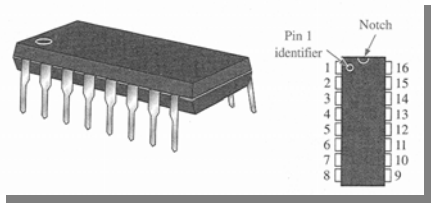
Basic Logic Operations

- The OR operation
  - When any input is HIGH, the output is HIGH
  - When both inputs are LOW, the output is LOW

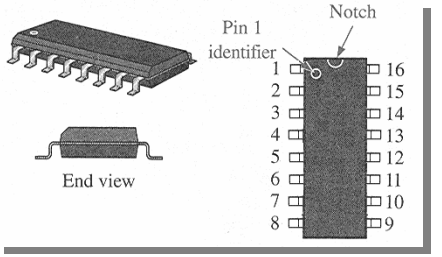


Fixed-Function Integrated Circuits

- Dual in-line package (DIP)

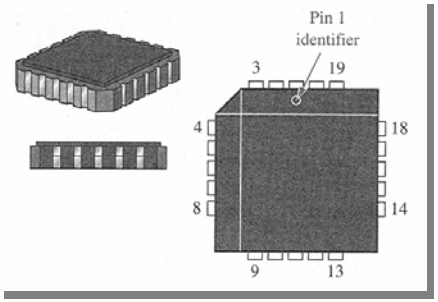


- Small-outline IC (SOIC)

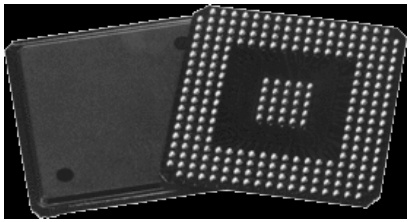


Fixed-Function Integrated Circuits

- Leadless-ceramic chip carrier (LCCC)



- Ball Grid Array (BGA)



History

- The first electronic logic was implemented using valves or relay as switches
  - slow by today's standards
  - large
  - got hot
  - relatively unreliable
- Transistor switches used now
  - many transistors can be "integrated" onto a single chip of silicon
  - fast (switch on and then off in around < 100 picosecond)
  - very small (order of 0.1 micron)
  - can get warm
  - very reliable

Example Applications

- Numeric
  - Calculators for addition, multiplication etc.
  - Aircraft navigation systems for calculating position, ETA etc.
  - Computers for averaging your exam marks
- Non-numeric
  - Parking meters for timing
  - Satellite TV encoding and decoding for revenue protection
  - Floppy-disk drives for controlling the rotation and head position

Design Example: Traffic Light Controller

- Specification
  - The traffic light points in 4 directions (N, S, E, W)
  - The lights on N and S are always the same, as are E and W
  - It cycles through the sequence green-yellow-red
  - N/S and E/W are never both green or yellow
  - Green lasts 45 seconds, yellow 15 seconds, red 60 seconds

- What are the outputs?
  - 12 (one for each light) but only 6 are unique
- What are the inputs?
  - start the controller (reset)
  - timing inputs (clocks)
- What about
  - performance
  - reliability
  - cost
  - power consumption
  - size, etc ?

- What about the logic?
  - IF N/S is green
  - AND E-W is red
  - AND 45 seconds has expired since the last light change
  - THEN the N/S lights should be changed from green to yellow
- What about the digital techniques to implement this?
  - It looks like a computer programme (that's logical!)
  - We need to form logical combinations of inputs
  - We need to conditionally set outputs according to the logical results

Points Addressed in this Lecture

- What do we mean by data?
- How can data be represented electronically?
- What number systems are often used and why?
- How do number systems of different bases work?
- How do you convert a number between binary and decimal?

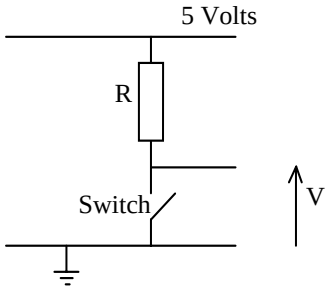
What do we mean by data?

  - Many definitions are possible depending on context

- We will say that:
  - data is a physical representation of information
- Data can be stored
  - e.g.: computer disk, cash till
- Data can be transmitted
  - e.g.: fax
- Data can be processed
  - e.g.: cash till

Electronic Representation of Data

- Information can be very complicated
  - e.g.:
    - Numbers Sounds
    - Pictures Codes
  - We need a simple electronic representation
- What can we do with electronics?
  - Set up voltages and currents
  - Change the voltages and currents
- A useful device is a switch
  - Switch Closed: V = 0 Volts
  - Switch Open: V = 5 Volts

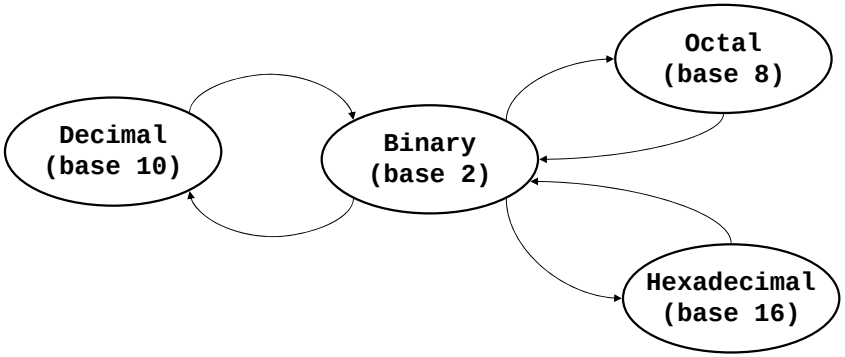


- Information can be represented by a voltage level
- The simplest information is TRUE/FALSE
  - This can be represented by two voltage levels:
    - 5 Volts for TRUE
    - 0 Volts for FALSE
- A voltage signal which has only two possibilities is a BIT
  - Bit stands for Binary Digit
- Binary means: only 2 possible values

FALSE	TRUE
(0)	(1)

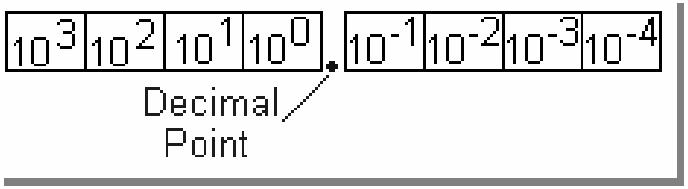
- Advantages of using binary representation
  - simple to implement in electronic hardware (switch)
  - good tolerance to noise

Number System Overview



Decimal Numbers

- The decimal number system has ten digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9
- The decimal numbering system has a base of 10 with each position weighted by a factor of 10:



Binary Numbers

- The binary number system has two digits: 0 and 1
- The binary numbering system has a base of 2 with each position weighted by a factor of 2:

POSITIVE POWERS OF TWO (WHOLE NUMBERS)									NEGATIVE POWERS OF TWO (FRACTIONAL NUMBER)					
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
256	128	64	32	16	8	4	2	1	1/2	1/4	1/8	1/16	1/32	1/64
									0.5	0.25	0.125	0.0625	0.03125	0.015625

Binary Number System

- Uses 2 symbols by our previous rule
  - 0 and 1
- Example: 10011 in binary is

$$1 \times 2^4 + 1 \times 2^1 + 1 \times 2^0 = 19$$

2^4	2^3	2^2	2^1	2^0
1	0	0	1	1

- Binary is the base 2 number system
- Most common in digital electronics

Integer and Fractional Parts

- Binary numbers can contain fractional parts as well as integer parts

2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}
1	0	0	1	1	0	1	1

Binary Point

$(19.375)_{10}$

- This 8-bit number is in Q3 format
  - 3 bits after the binary point
- How could 19.376 best be represented using an 8-bit binary number?
  - Quantization error

Conversion: decimal to binary (Method 1)

- The decimal number is simply expressed as a sum of powers of 2, and then 1s and 0s are written in the appropriate bit positions.

$$\begin{aligned} 50_{10} &= 32 + 18 \\ &= 32 + 16 + 2 \\ &= 1\times 2^5 + 1\times 2^4 + 1\times 2^1 \\ 50_{10} &= 110010_2 \end{aligned}$$

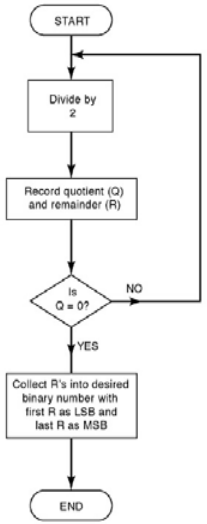
$$\begin{aligned} 346_{10} &= 256+90 \\ &= 256+64+26 \\ &= 256+64+16+10 \\ &= 256+64+16+8+2 \\ &= 1\times 2^8 + 1\times 2^6 + 1\times 2^4 + 1\times 2^3 + 1\times 2^1 \\ 346_{10} &= 101011010_2 \end{aligned}$$

Conversion: decimal to binary (method 2)

- Repeated division

	quotient	remainder	
50/2 =	25	0	LSB
25/2 =	12	1	
12/2 =	6	0	
6/2 =	3	0	
3/2 =	1	1	
1/2 =	0	1	MSB

$50_{10}=110010_2$



Conversion: binary to decimal

- The simplest way is to represent the binary number as

$$a_n \times 2^{n-1} + \ldots + a_2 \times 2^2 + a_1 \times 2^1 + a_0 \times 2^0$$

- The conversion can be done by substituting the a's with the given bits then multiplying and adding:
  - eg: Convert $(1101)_2$ into decimal
  - $1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = (13)_{10}$
- Other algorithms can be used as alternatives if you prefer

Binary Addition

- First recall decimal addition

	1	1	1	
A	1	2	3	4
+ B		9	8	7
Sum	2	2	2	1

- In binary addition we follow the same pattern but
  - 0 + 0 = 0 carry-out 0
  - 0 + 1 = 1 carry-out 0
  - 1 + 0 = 1 carry-out 0
  - 1 + 1 = 0 carry-out 1
  - 1 + 1 + carry-in = 1 carry-out 1

		1		
A	0	1	1	1
+ B	0	1	1	0
Sum	1	1	0	1

- Note that we need to consider 3 inputs per bit of binary number
  - A, B and carry-in
- Each bit of binary addition generates 2 outputs
  - sum and carry-out

Hexadecimal Numbers

- Decimal, binary, and hexadecimal numbers

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

Hexadecimal Numbers conversions

- **Binary-to-hexadecimal conversion**
  1. Break the binary number into 4-bit groups
  2. Replace each group with the hexadecimal equivalent

- **Hexadecimal-to-decimal conversion**
  1. Convert the hexadecimal to groups of 4-bit binary
  2. Convert the binary to decimal

- **Decimal-to-hexadecimal conversion**
  – Repeated division by 16

Binary Coded Decimal (BCD)

- Use 4-bit binary to represent one decimal digit
- Easy conversion
- Wasting bits (4-bits can represent 16 different values, but only 10 values are used)
- Used extensively in financial applications

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

Binary Coded Decimal (BCD)

- Convert 0110100000111001(BCD) to its decimal equivalent.

  0110 1000 0011 1001
  6 8 3 9

- Convert the BCD number 011111000001 to its decimal equivalent.

  0111 1100 0001
  7 ↑ 1

  The forbidden code group indicated an error

Putting it together

Decimal	Binary	Octal	Hexadecimal	BCD
0	0	0	0	0000
1	1	1	1	0001
2	10	2	2	0010
3	11	3	3	0011
4	100	4	4	0100
5	101	5	5	0101
6	110	6	6	0110
7	111	7	7	0111
8	1000	10	8	1000
9	1001	11	9	1001
10	1010	12	A	0001 0000
11	1011	13	B	0001 0001
12	1100	14	C	0001 0010
13	1101	15	D	0001 0011
14	1110	16	E	0001 0100
15	1111	17	F	0001 0101

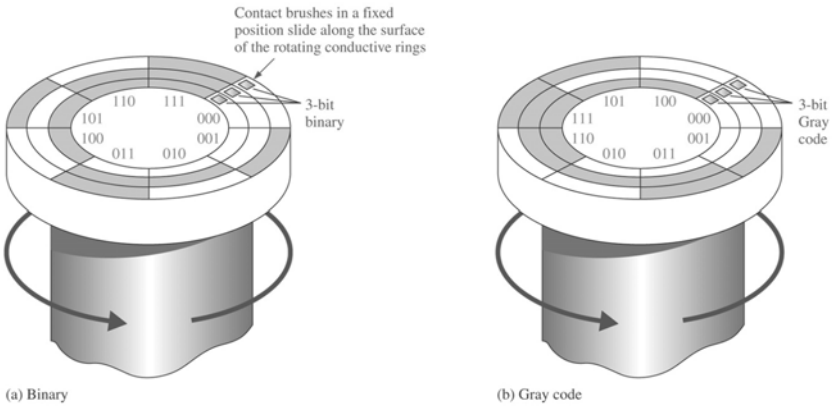
Gray Codes

- Only 1 bit changes in the count sequence
- Useful for industrial control

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

Gray Codes

- Binary code results in glitches
- Gray code avoids glitches



ASCII code

- Codes representing letters of the alphabet, punctuation marks, and other special characters as well as numbers are called *alphanumeric* codes.
- The most widely used alphanumeric code is the American Standard Code for Information Interchange(ASCII). The ASCII (pronounced “askee”) code is a seven-bit code.

Character	Seven-Bit ASCII	Octal	Hex	Character	Seven-Bit ASCII	Octal	Hex
A	100 0001	101	41	Y	101 1001	131	59
B	100 0010	102	42	Z	101 1010	132	5A
C	100 0011	103	43	0	011 0000	060	30
D	100 0100	104	44	1	011 0001	061	31
E	100 0101	105	45	2	011 0010	062	32
F	100 0110	106	46	3	011 0011	063	33
G	100 0111	107	47	4	011 0100	064	34
H	100 1000	110	48	5	011 0101	065	35
I	100 1001	111	49	6	011 0110	066	36
J	100 1010	112	4A	7	011 0111	067	37
K	100 1011	113	4B	8	011 1000	070	38
L	100 1100	114	4C	9	011 1001	071	39
M	100 1101	115	4D	blank	010 0000	040	20
N	100 1110	116	4E	.	010 1110	056	2E
O	100 1111	117	4F	(	010 1000	050	28
P	101 0000	120	50	+	010 1011	053	2B
Q	101 0001	121	51	$	010 0100	044	24
R	101 0010	122	52	*	010 1010	052	2A
S	101 0011	123	53	)	010 1001	051	29
T	101 0100	124	54	—	010 1101	055	2D
U	101 0101	125	55	/	010 1111	057	2F
V	101 0110	126	56	,	010 1100	054	2C
W	101 0111	127	57	=	011 1101	075	3D
X	101 1000	130	58	⟨RETURN⟩	000 1101	015	0D
				⟨LINEFEED⟩	000 1010	012	0A