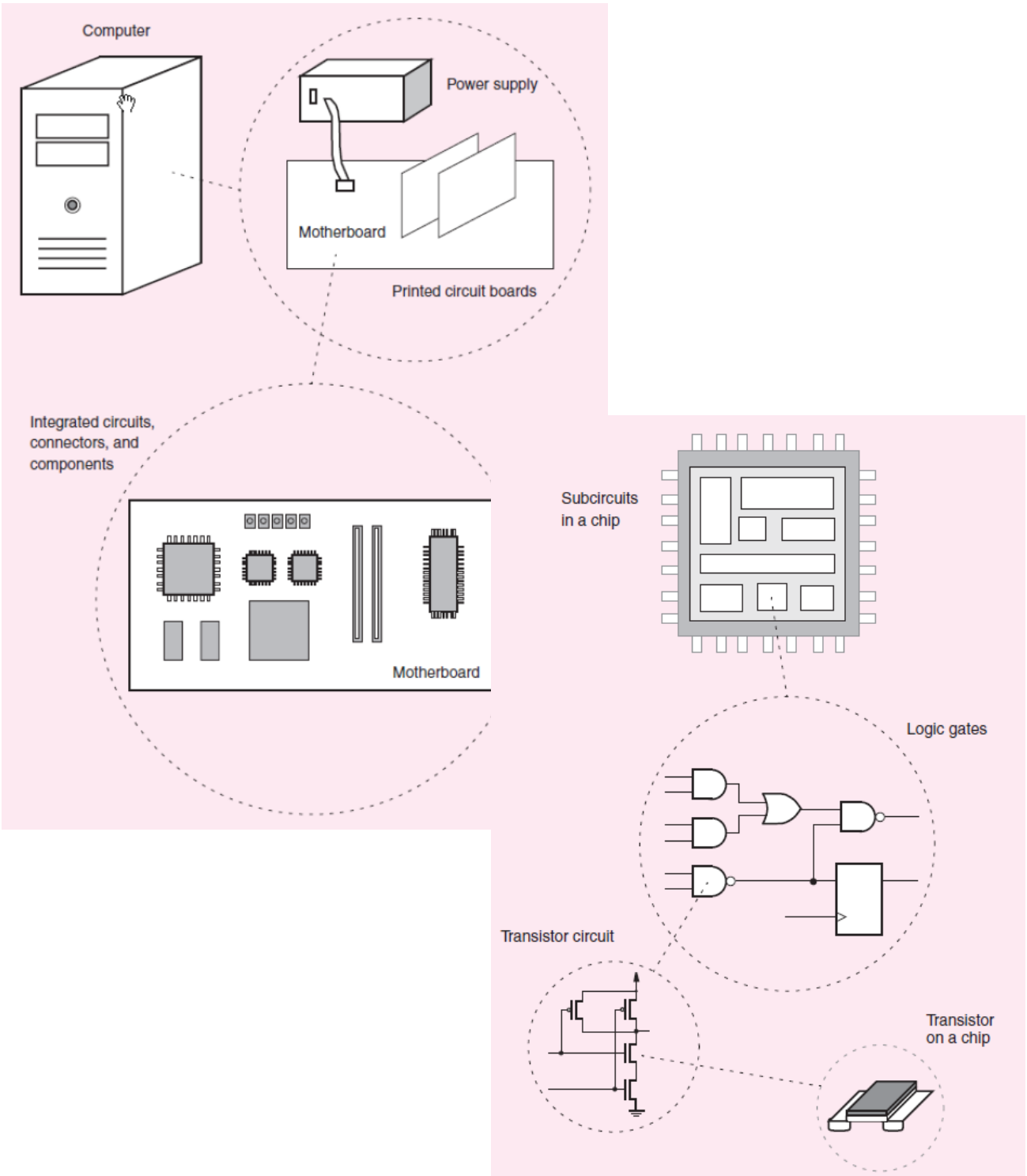
Lecture 1: Introduction and Numbers

- In this lecture, we will briefly introduce the scope of the class, concept of layered abstractions, and the number systems.

Introduction

- Digital signals
  - Signals with discrete values.
- Analog signals
  - Signals varying continuously over a specified range.
- Quantization
  - A process to quantify an analog value into a digital value.
  - Limits of accuracy imposed by the digital number system. (Length of bits used)

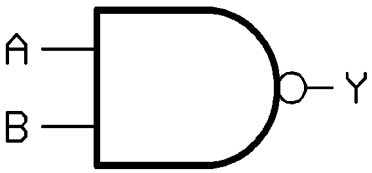
Level of Abstractions



Abstraction: lower level

- Abstraction (from the Latin *abs*, meaning *away from* and *trahere*, meaning *to draw*) is the process of taking away or removing characteristics from something in order to reduce it to a set of essential characteristics.

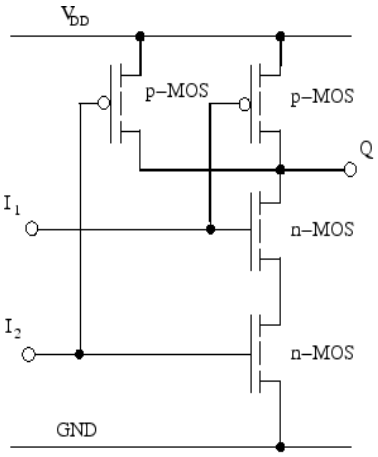
Logic Course

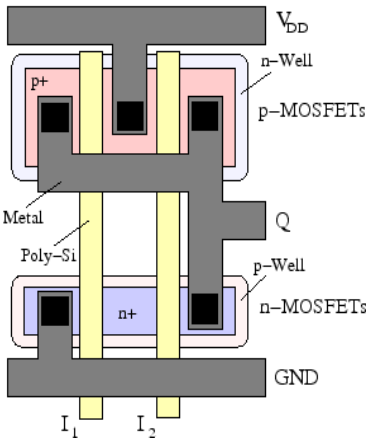


NAND

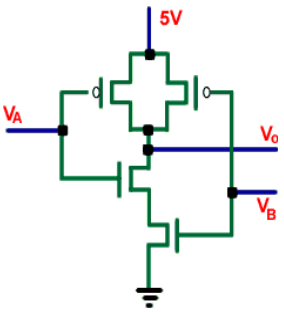
VLSI Layout

CMOS NAND layout





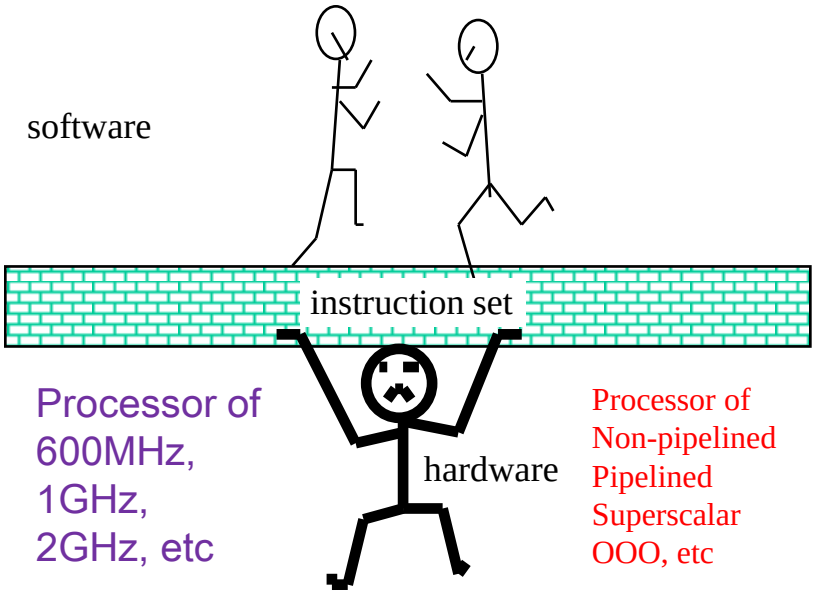
Microelectronics



CMOSNAND

Abstraction: somewhere in the middle

Computer organization
Operating system



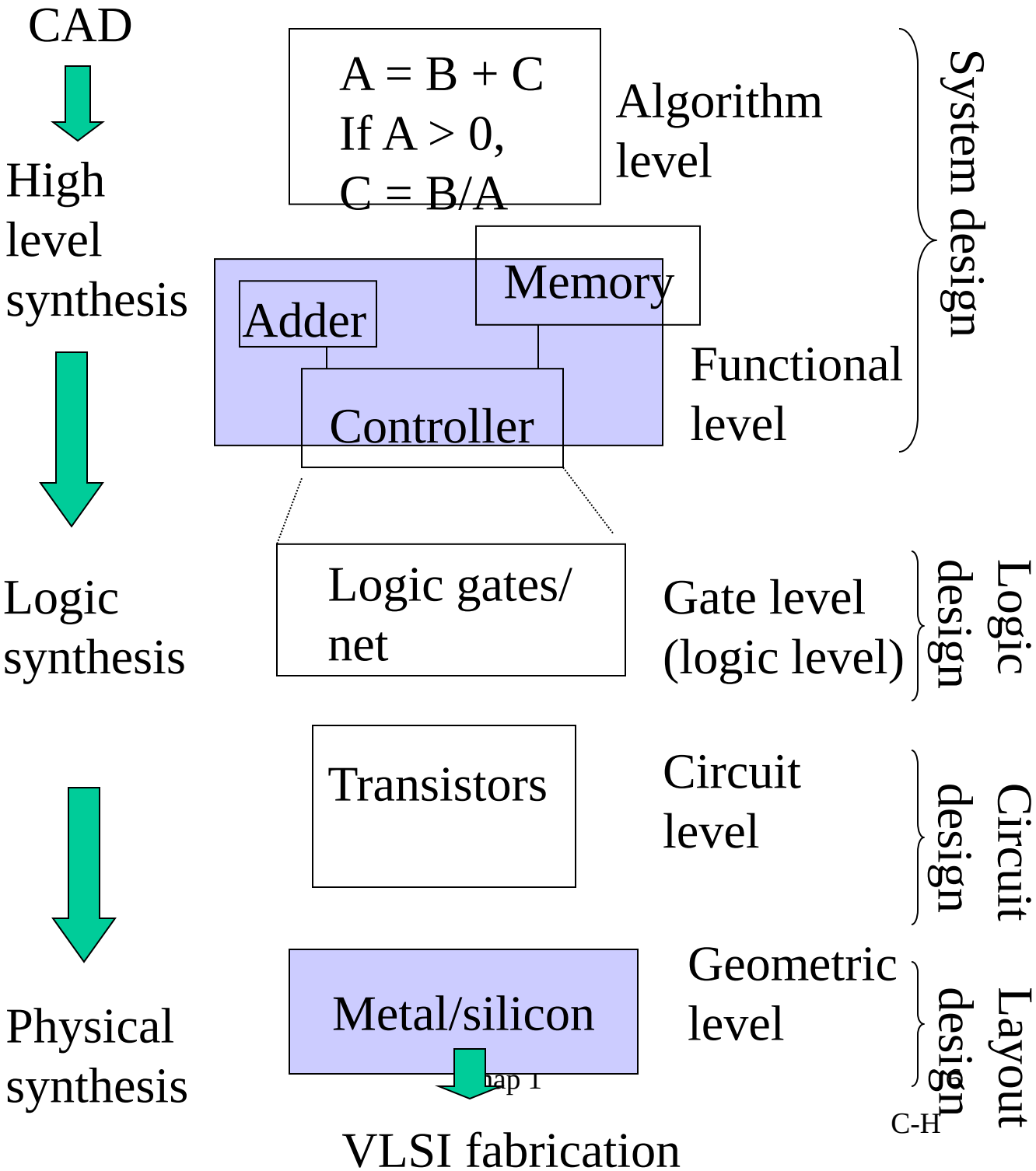
- Instruction set architecture

  ADD r2, r3, r4

  Ripple adder
  Carry lookahead adder

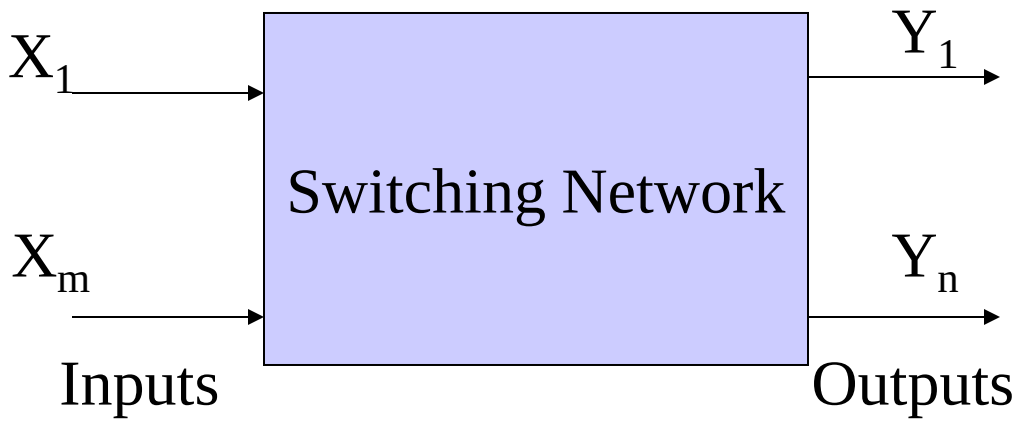
Layer of Digital System Design

- Level of abstraction



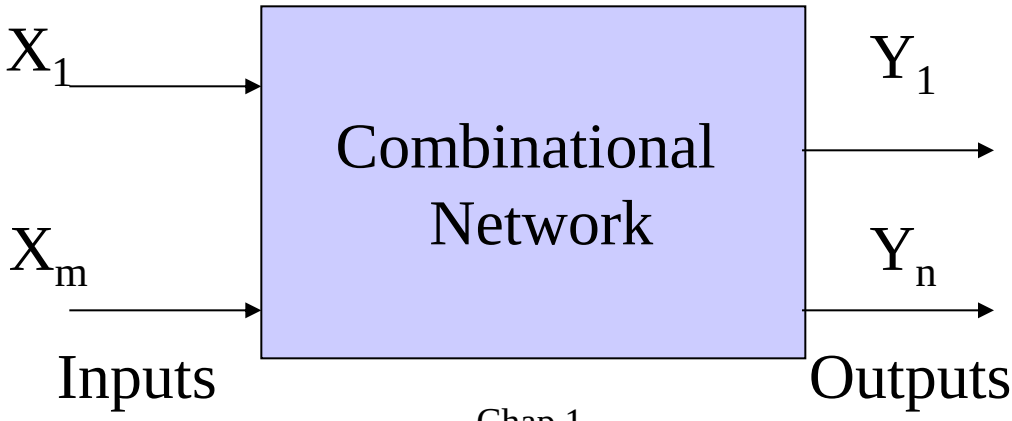
Switching Network

- Primary Inputs and Primary output
  - Combinational network
  - Sequential network



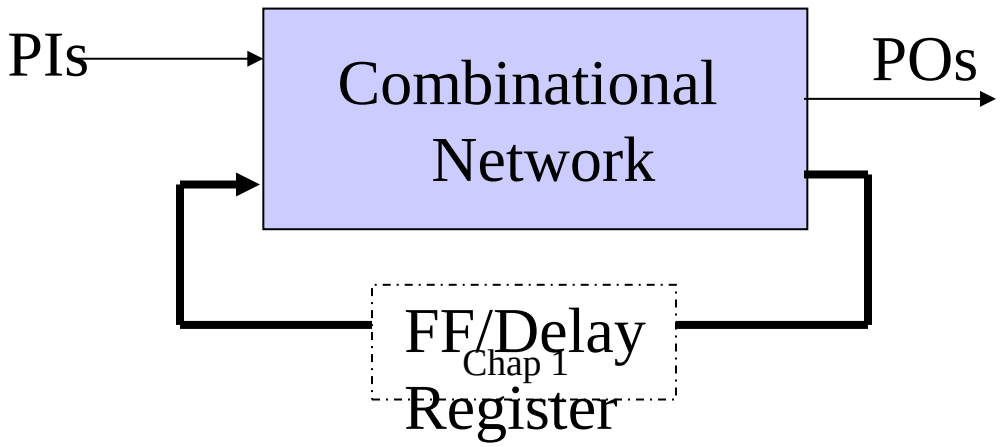
Combinational Network

- Output values depend on the present value of the inputs and not on the past values
  - Representation: Table, algebraic logic equations
  - Simplification methodology: Karnaugh map, Quine-McCluskey procedure, Boolean algebra
  - $Y_i = f(X_1,.. X_m)$,e.g., ALU



Sequential Network

- Output values depend on the present value of the inputs and on the past input values
  - Feedback signals
  - Synchrnous logic (Wait for CLK) v.s. asynchronous logic (Does not wait for CLK.)
  - Representation: timing diagram, state table, logic equations
  - Simplification: techniques used in COM, state minimization



Number System

- Decimal
  - A decimal number: $123.5 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 5 \times 10^{-1}$
- Binary
  - A binary number: $0101 = 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
- Weighted number system
  - W_i is the weight assigned to digit in the i^{th} position, X_i
  - W_i is the i^{th} power of a fixed integer R (R is the radix or base of the number system)

$$\sum_{i=0}^{n-1} X_i W_i$$

Number System: representation

- If R is the radix, then R digits (0, 1, 2, …R-1) are used.
- For example,

$$N = (d_4 d_3 d_2 d_1 d_0 . d_{-1} d_{-2})_R$$

$$= d_4 \times R^4 + d_3 \times R^3 + d_2 \times R^2 + d_1 \times R^1 + d_0 \times R^0 + d_{-1} \times R^{-1} + d_{-2} \times R^{-2},$$

where

d_i is the coeffieient of R^i and $0 =< d_i =< R-1$

Number System Examples

- $234.7_8 = 2 \times 8^2 + 3 \times 8^1 + 4 \times 8^0 + 7 \times 8^{-1}$
- Hexdecimal number
  - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.
- $AD05_{16} = A \times 16^3 + D \times 16^2 + 0 \times 16^1 + 5 \times 16^0$.

Typical Number System

- Binary, Decimal, Hexadecimal
  - Why bothers?

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

Number System Conversion

- Convert a decimal integer to base R?
- Example Convert 53_{10} to binary.
- $53_{10} = 110101_2$

53	1
26	0
13	1
6	0
3	1
1	1
0	

Number System Conversion

- Convert a decimal integer to base R?
- $N_{10} = (a_n a_{n-1} .. a_0)_R$
  $= a_n R^n + a_{n-1} R^{n-1} + \ldots + a_1 R^1 + a_0 R^0$

  $= \{\ldots\{a_n R + a_{n-1}\}R \ldots\ldots + a_3\}R + a_2\}R + a_1\}R + a_0$

- Therefore a_0 can be found by N/R since a_0 is the remainder. So does a_1 to a_n.

Number System Conversion cont.

- Convert a decimal fraction to base R?
- Example: Convert $.625_{10}$ to binary.

$.625_{10} = 0.101_2$

.625	.25	.5
x 2	x 2	x 2
1.250	0.5	1.0
$a_{-1} = 1$	$a_{-2} = 0$	$a_{-3} = 1$

Number System Conversion cont.

- Convert a decimal fraction to base R?
- $F_{10} = (.a_{-1}\, a_{-2} .. a_{-m})_R$
  $= a_{-1} R^{-1} + a_{-2} R^{-2} + \ldots + a_{-m} R^{-m}$

  $= R^{-1}\{a_{-1} + R^{-1}\{a_{-2} + R^{-1}\{a_{-3} + \ldots\}\}\}$
- Therefore a_{-1} can be found by $F \times R$. (Remember that the goal is to find the coefficients.)
- The conversion process does not always terminate!

Number System Conversion cont.

- Conversion between two bases other than decimal.
- Example 233.2_4 to base 7
  - Covert 233.2_4 to decimal first, then
  - Use division by 7 to find the integer part and use multiplication by 7 to find the fraction part.

Number System Conversion cont.

- Conversion between base 2 and base 8 (binary to octal) or binary and hexadecimal.
- Example
- $11000.11011_2 = 30.66_8$
- $11000.11011_2 = 18.D8_{16}$

Binary Arithmetic

- Addition
  - 0 + 0 = 0
  - 0 + 1 = 1
  - 1 + 0 = 1
  - 1 + 1 = 0 (and carry 1 to the next column)
- Example:

```
   1101
 + 1011
-------
 1 1000
```

Binary Arithmetic cont.

- Subtraction
  - 0 - 0 = 0
  - 0 - 1 = 1 (borrow 1 from the next column)
  - 1 - 0 = 1
  - 1 - 1 = 0
- Example:

$$\begin{array}{r} 1101 \\ -\ 1011 \\ \hline 0010 \end{array}$$

Binary Arithmetic

- Multiplication
  - 0 x 0 = 0
  - 0 x 1 = 0
  - 1 x 0 = 0
  - 1 x 1 = 1

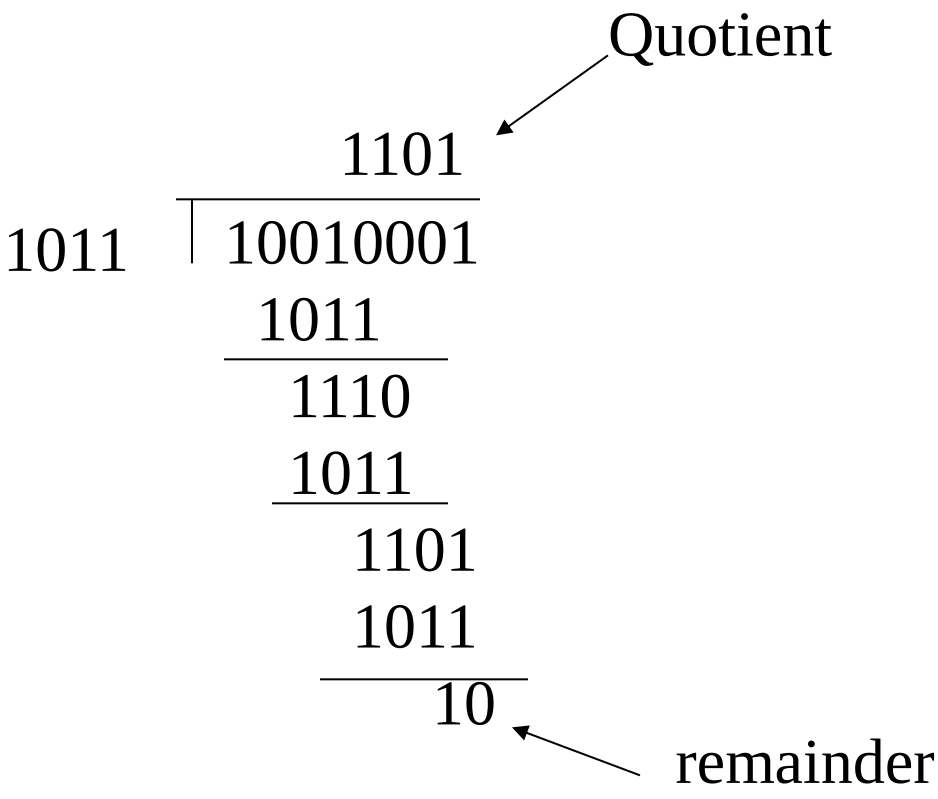
- Example: 13 x 11

13: multiplicand
11: multiplier

```
    1101
x   1011
------
    1101
   1101
  0000
 1101
------------
10001111        product
```

Binary Arithmetic

- Division
  - Example: 145/11(Dividend/divisor)



Binary Codes for Decimal Digits (Table1-1)

- These codes are different from that obtained by conversion.
  - Binary-coded-decimal (BCD, 8-4-2-1 BCD)
    - Weighted code
  - 6-3-1-1 code
    - Weighted code
  - Excess-3 code
    - (8-4-2-1) value + 3
  - 2-out-of-5 code
    - Exactly 2 out of the 5 bits are 1
  - Gray code
    - Successive decimal digits differs in one bit.

Binary Codes for Decimal Digits

- Why bothers?
- Implementation?

Binary Codes for Decimal Digits

Decimal Digit	8-4-2-1 Code (BCD)	6-3-1-1 Code	Excess-3 Code	2-out-of-5 Code	Gray Code
0	0000	0000	0011	00011	0000
1	0001	0001	0100	00101	0001
2	0010	0011	0101	00110	0011
3	0011	0100	0110	01001	0010
4	0100	0101	0111	01010	0110
5	0101	0111	1000	01100	1110
6	0110	1000	1001	10001	1010
7	0111	1001	1010	10010	1011
8	1000	1011	1011	10100	1001
9	1001	1100	1100	11000	1000

Mirror method

- Generate reflective Gray code

Decimal	1-bit reflective Gray code	Decimal	2-bit reflective Gray code	Decimal	3-bit reflective Gray code	Decimal	4-bit reflective Gray code
0	0	0	00	0	000	0	0000
1	1	1	01	1	001	1	0001
		2	11	2	011	2	0011
		3	10	3	010	3	0010
				4	110	4	0110
				5	111	5	0111
				6	101	6	0101
				7	100	7	0100
						8	1100
						9	1101
						10	1111
						11	1110
						12	1010
						13	1011
						14	1001
						15	1000

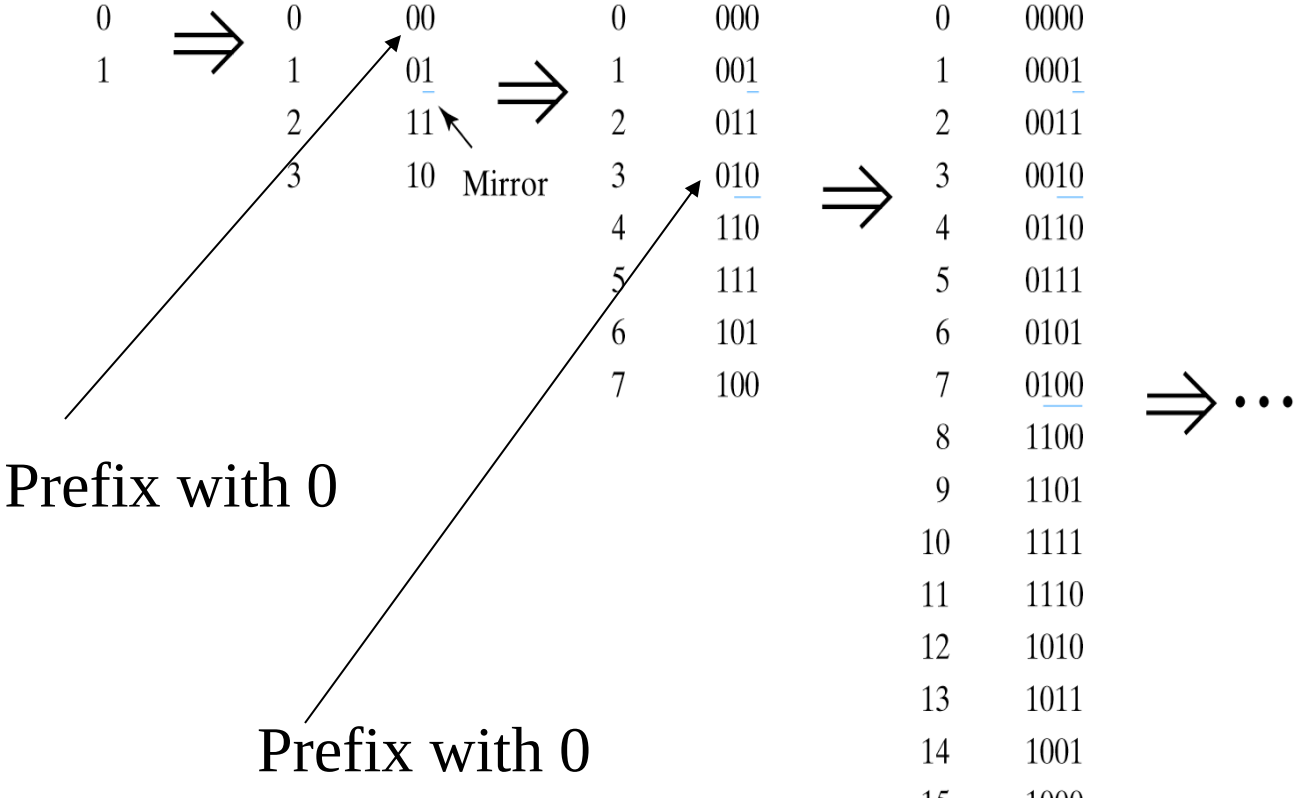


Figure 1.7.3 Decimal to reflective Gray-code conversion using the mirror method.

ASCII code

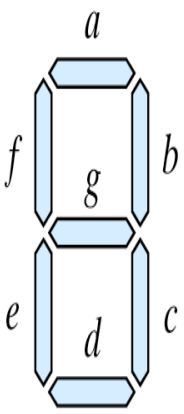
- America Standard Code for Information Interchange.

ASCII Code

Char-acter	ASCII Code A_6	A_5	A_4	A_3	A_2	A_1	A_0	Char-acter	ASCII Code A_6	A_5	A_4	A_3	A_2	A_1	A_0	Char-acter	ASCII Code A_6	A_5	A_4	A_3	A_2	A_1	A_0
space	0	1	0	0	0	0	0	@	1	0	0	0	0	0	0	`	1	1	0	0	0	0	0
!	0	1	0	0	0	0	1	A	1	0	0	0	0	0	1	a	1	1	0	0	0	0	1
"	0	1	0	0	0	1	0	B	1	0	0	0	0	1	0	b	1	1	0	0	0	1	0
#	0	1	0	0	0	1	1	C	1	0	0	0	0	1	1	c	1	1	0	0	0	1	1
$	0	1	0	0	1	0	0	D	1	0	0	0	1	0	0	d	1	1	0	0	1	0	0
%	0	1	0	0	1	0	1	E	1	0	0	0	1	0	1	e	1	1	0	0	1	0	1
&	0	1	0	0	1	1	0	F	1	0	0	0	1	1	0	f	1	1	0	0	1	1	0
'	0	1	0	0	1	1	1	G	1	0	0	0	1	1	1	g	1	1	0	0	1	1	1
(	0	1	0	1	0	0	0	H	1	0	0	1	0	0	0	h	1	1	0	1	0	0	0
)	0	1	0	1	0	0	1	I	1	0	0	1	0	0	1	i	1	1	0	1	0	0	1
*	0	1	0	1	0	1	0	J	1	0	0	1	0	1	0	j	1	1	0	1	0	1	0
+	0	1	0	1	0	1	1	K	1	0	0	1	0	1	1	k	1	1	0	1	0	1	1
,	0	1	0	1	1	0	0	L	1	0	0	1	1	0	0	l	1	1	0	1	1	0	0
-	0	1	0	1	1	0	1	M	1	0	0	1	1	0	1	m	1	1	0	1	1	0	1
.	0	1	0	1	1	1	0	N	1	0	0	1	1	1	0	n	1	1	0	1	1	1	0
/	0	1	0	1	1	1	1	O	1	0	0	1	1	1	1	o	1	1	0	1	1	1	1
0	0	1	1	0	0	0	0	P	1	0	1	0	0	0	0	p	1	1	1	0	0	0	0
1	0	1	1	0	0	0	1	Q	1	0	1	0	0	0	1	q	1	1	1	0	0	0	1
2	0	1	1	0	0	1	0	R	1	0	1	0	0	1	0	r	1	1	1	0	0	1	0
3	0	1	1	0	0	1	1	S	1	0	1	0	0	1	1	s	1	1	1	0	0	1	1
4	0	1	1	0	1	0	0	T	1	0	1	0	1	0	0	t	1	1	1	0	1	0	0
5	0	1	1	0	1	0	1	U	1	0	1	0	1	0	1	u	1	1	1	0	1	0	1
6	0	1	1	0	1	1	0	V	1	0	1	0	1	1	0	v	1	1	1	0	1	1	0
7	0	1	1	0	1	1	1	W	1	0	1	0	1	1	1	w	1	1	1	0	1	1	1
8	0	1	1	1	0	0	0	X	1	0	1	1	0	0	0	x	1	1	1	1	0	0	0
9	0	1	1	1	0	0	1	Y	1	0	1	1	0	0	1	y	1	1	1	1	0	0	1
:	0	1	1	1	0	1	0	Z	1	0	1	1	0	1	0	z	1	1	1	1	0	1	0
;	0	1	1	1	0	1	1	[	1	0	1	1	0	1	1	{	1	1	1	1	0	1	1
<	0	1	1	1	1	0	0	\	1	0	1	1	1	0	0	|	1	1	1	1	1	0	0
=	0	1	1	1	1	0	1	]	1	0	1	1	1	0	1	}	1	1	1	1	1	0	1
>	0	1	1	1	1	1	0	^	1	0	1	1	1	1	0	~	1	1	1	1	1	1	0
?	0	1	1	1	1	1	1	_	1	0	1	1	1	1	1	delete	1	1	1	1	1	1	1

7-segment display

- Seven-segment code



7-segment code *a b c d e f g*	7-segment display	7-segment code *a b c d e f g*	7-segment display
1 1 1 1 1 1 0	0	1 0 1 1 0 1 1	5
0 1 1 0 0 0 0	1	1 0 1 1 1 1 1	6
1 1 0 1 1 0 1	2	1 1 1 0 0 0 0	7
1 1 1 1 0 0 1	3	1 1 1 1 1 1 1	8
0 1 1 0 0 1 1	4	1 1 1 1 0 1 1	9

Figure 1.7.7 Seven-segment code for a 7-segment display.

Complement

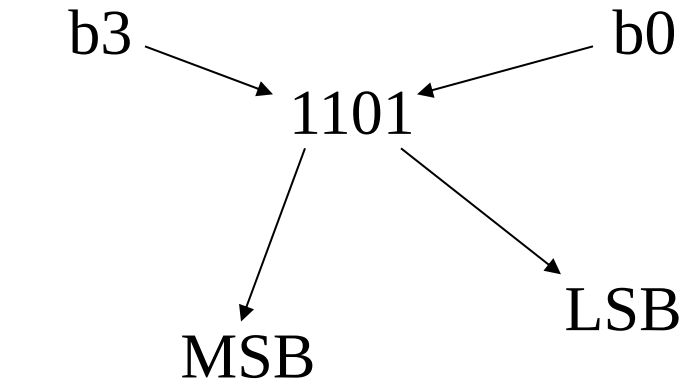
- What is 2's complement?
  - Definition
    - N* : 2's complement of N (a positive number of n bits)
    - $N^* = 2^n - N$
- 1's complement
    - $N^{\#} = (2^n - 1) - N$

$$N^* = 2^n - N = (2^n - 1 - N) + 1 = N^{\#} + 1$$

Negative Number

- 2’s complement for negative number
  - Example: Use 4 bits and 2’s complement to represent -3

  = 1101 why?



Representation of Signed Number

- Use 4 bits to represent a signed number.

Decimal	Signed bit	1's C	2's C
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
-0	1000	1111	0000
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1110	1001	1010
-7	1111	1000	1001
-8	NA	NA	1000

Representation of Signed Number cont.

- Three signed number representations:
  - Signed magnitude
  - Use a sign bit:
    - +22 = **0**10110, use 6-bit representation.
    - -22 = **1**10110
    - You have +0 and –0.
  - Radix complement representation (RC = 2'S)
  - Diminished radix complement (DRC = 1'S)
    - +0 = 000000
    - -0 = 111111

Overflow issues

- Unsigned overflow (usually called carry)
  - If numbers are unsigned, then overflow occurs when the sum of the two numbers being added exceeds the number range of the word size.
  - That is, if the most significant carry bit = 1, then overflow occurs.
- Signed overflow (2's complement)
  - C_{SBP}: carry bit into the sign bit position
  - C_{SBP+1}: carry bit out of the sign bit position.
  - If $C_{SBP} = C_{SBP+1}$, then no signed overflow
  - If C_{SBP} not equal to C_{SBP+1}, then signed overflow occurs.
  - Signed number is different from both (+) or both (-)

Additions

- Addition
  - 2's complement
    - Throw the last carry away.

```
-3   1101
-4   1100   +
-------------
   1 1001
```

  - 1's complement
    - End-around carry

```
-3    1100
-4    1011   +
    1 0111
-------------
        1
    1000
```

DAC and ADC

- Digital to Analog Converter

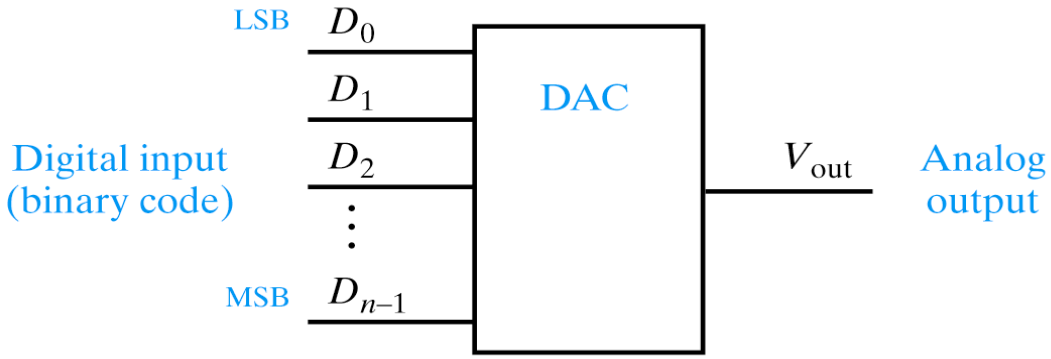


Figure P1.58

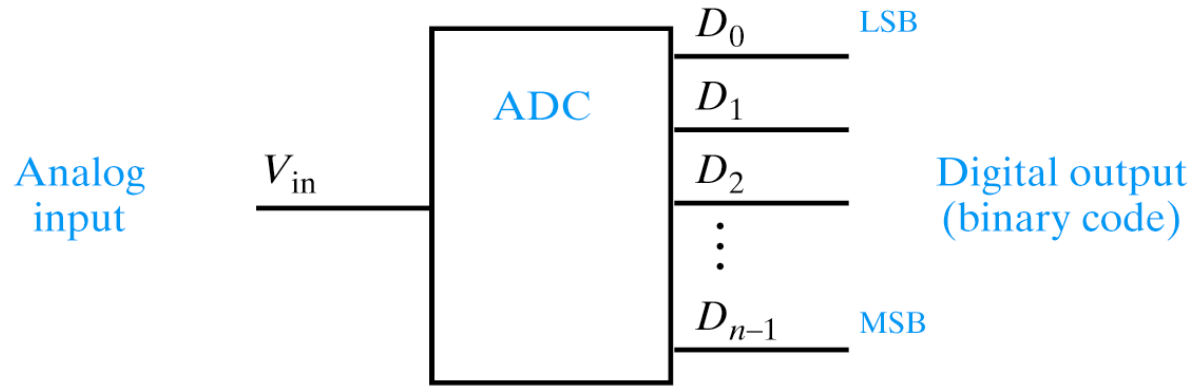
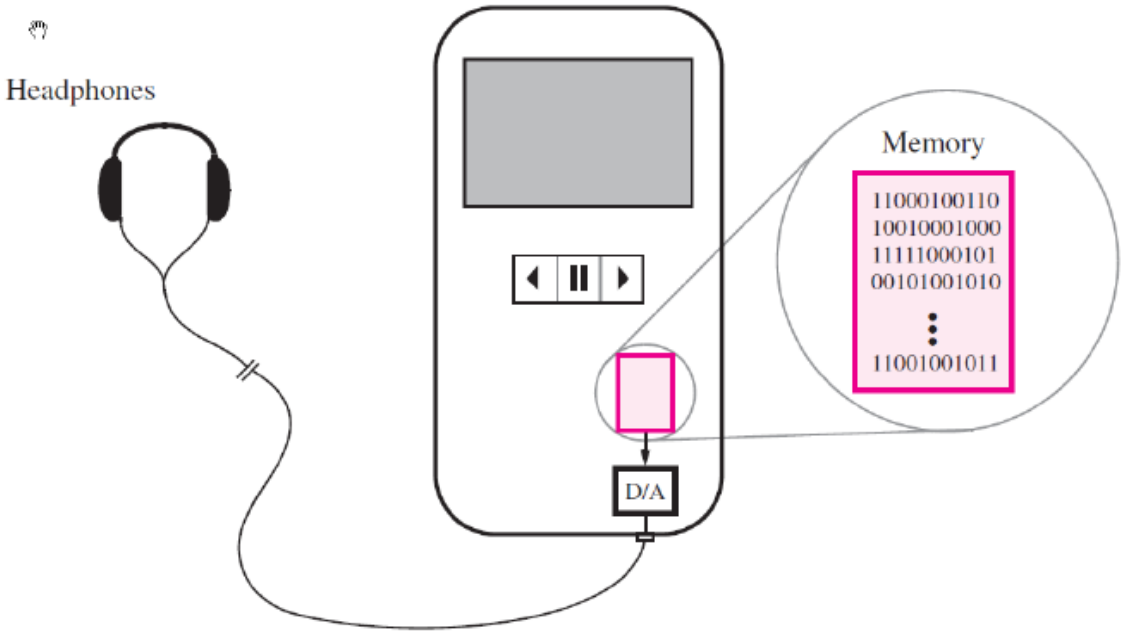


Figure P1.57

D2A Example: MP3



MP3, MPEG-1 or MPEG-2 Audio Layer III, is an encoding format for digital audio which uses a form of lossy data compression.