CpE2210
Introduction to Digital Logic

Dr. Minsu Choi

CH 1,2. Introduction to Concepts in
Digital Systems & Computer Arithmetic

What is a Digital System?

- A digital system is an electronic network that processes information using only digits (numbers) to implement calculations and operations.
- Binary number system is used for digital systems in general.

Binary number system

- The mostly used number system for a digital system.
- Base-2 number system (e.g., base-10 - decimal, base-8 - octal, base-16 - hexadecimal, etc)
- Each digit can be either a value of 0 or 1. The number themselves (0 or 1) are called bits (e.g., 0101 is 4-digit binary number)

Digital system tasks

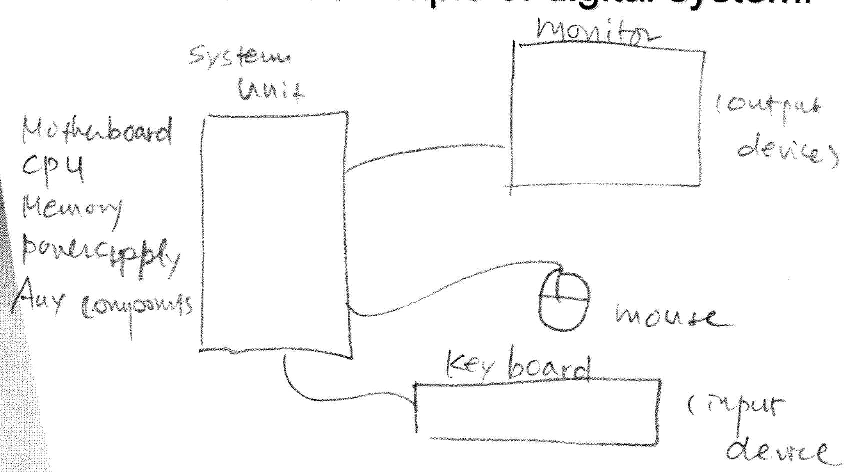
- Input translation: translates information from our world into a binary "language" that can be understood by the digital network.
- Data processing: performs the required calculations and operations using only the binary digits 0 and 1.
- Output: returns an answer to our world in a form that ~~can we~~ we can understand.

Views of a Digital System

- A digital network can be viewed in different ways:
  - Hierarchies - primitive units -> more complex units -> even larger units.
  - Logic networks - digital network is based on the behavior of Binary numbers. It is possible to describe any digital network in terms of the fact that groups of binary variable can be used to represent virtually any set of data.
  - Electrical circuits - the physical realization of a digital network is accomplished by using electronic components that control the flow of electric current in a manner that implements logic operations.
  - Formal description - it is possible to describe the behavior of a digital system by using only descriptive phrases that are defined within a context of a "language" HDLs (hardware description languages) can be used.

The Personal Computer (PC)

- An excellent example of digital system.



Binary Number System

- A Binary variable can store either 0 or 1.
- ex) Bin variable A ⇒ $A = \emptyset$ or $A = 1$.
- Unary operation NOT (also called inversion).

$NOT(A) = \overline{A} = A'$ (our textbook)

called complement of A.

such that

$NOT(\emptyset) = 1$

$NOT(1) = \emptyset$

How to describe more complex situations?

- We can use groups of bits!
- Ex) Four digit data representation

data = $a_3 a_2 a_1 a_0$ where a_i can represent either $\emptyset$ or 1.

⇒ data = 0000, 0001, 0010, 0011
0100, 0101, 0110, 0111
1000, 1001, 1010, 1011
1100, 1101, 1110, 1111

$= 2^4 = 16$ possible permutations
$= 16$ possible bit patterns.

- A group of bits -> word.

data = 0101 means

$a_3 = 0, a_2 = 1$

$a_1 = 0, a_0 = 1$

# of bit patterns.

Bit & Byte

- Suffix b – bit and suffix B – byte such that 1B = 8b (e.g., 1KB = 8Kb = 8 x 1024 bits).

word size	# of values	abbreviation for value
8b	$2^8 = 256$	
10b	$2^{10} = 1024$	1K~~b (kilobit)~~
16b	65,536	64KB
20b	2^{20}	1M~~b (megabit)~~
28b	2^{28}	256MB
30b	2^{30}	1G~~b (gigabit)~~
40b	2^{40}	1T~~b (terabit)~~

mem size	abbreviation
1024b	1Kb
2^{20}b	1Mb
2^{30}b	1Gb
2^{40}b	1Tb

Data Representations

- Binary numbers can be used to represent anything that we want by designing them in an appropriate manner.
- Ex) 4 directions (left, right, forward, backward)

4 directions → So, 2 bit binary word is required, since $2^2 = 4$.

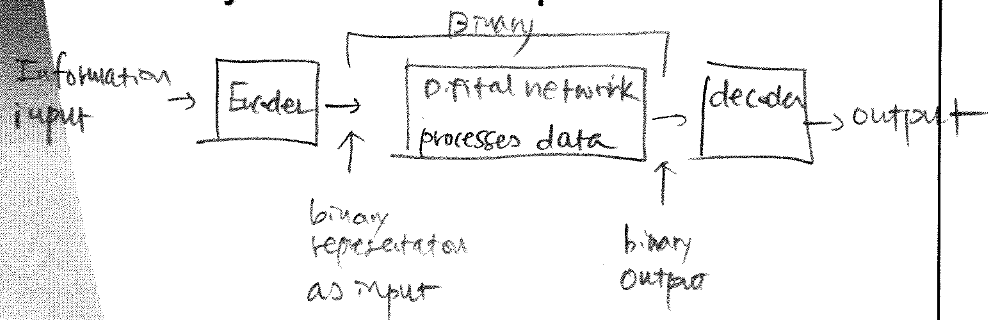
Let's define direction word $D = D_1 D_0$

$D = 00$ ⇒ left
$D = 01$ ⇒ right
$D = 10$ ⇒ forward
$D = 11$ ⇒ backward

note that these assignments are completely arbitrary.

Encoding & Decoding

- Encoding: The process of giving meaning to a group of bits.
- Decoding: The reverse process where a binary number is interpreted for our use.



Binary and Decimal Numbers

- We live in a world where there are ten symbols ~~digits~~ (0-9) -> Decimal number system.
- Digital systems use binary number system in general.
- Dec -> Bin and Bin -> Dec translations are discussed in this section.

Number Theory

- Each number system has base (or radix).
- Decimal number system's base is 10.
- Each digit can be either one of 10 symbols from 0 to 9 in decimal.
- Base or radix r for a number system.
- r=2 -> Binary (Base-2)
- r=8 -> Octal (Base-8)
- r=10 -> Decimal (Base-10)
- r=16 -> Hexadecimal (Base-16)

Binary to decimal conversion

- To represent 0-9 (decimal) using binary words, 4 Binary digit word is required, since 2^3 = 8 < 10 < 2^4 = 16.
- Let us construct a 4-bit Binary word. $N = N_3N_2N_1N_0$

Decimal representation is...

$= N_3 \times 2^3 + N_2 \times 2^2 + N_1 \times 2^1 + N_0 \times 2^0$

$= N_3 \times 8 + N_2 \times 4 + N_1 \times 2 + N_0 \times 1$ #

- This shows that a binary digit in N_j has a base-10 weighting of 2^j.

ex) $N = 110_{(2)} = 1\times2^2 + 1\times2^1 + 0\times2^0$

$= 6_{(10)}$

means binary # (base 2)

means decimal # (base 10)

For j-digit binary # $N = N_{j-1} N_{j-2} \cdots N_0$

$$\boxed{\text{decimal \#} = \sum_{i=j-1}^{0} N_i \times 2^i}$$

Examples

ex) $X = 0101\ 1100_2$ (8-bit bin#)

$= 0\times2^7 + 1\times2^6 + 0\times2^5 + 1\times2^4$

$+ 1\times2^3 + 1\times2^2 + 0\times2^1 + 0\times2^0$

$= 64 + 16 + 8 + 4 = 92_{10}$

Bin → Dec conversion can be used to decode binary # output to human-friendly decimal # output.

Decimal to binary conversion: successive division algorithm

- Ex) Convert 19_{10} into binary number.

Successively divide 19 by 2 & keep track of remainders.

$2 \,\underline{|\,19}$ ... 1 → remainder $R_0 = 1$ → Least Significant Bit (LSB)

$2 \,\underline{|\,9}$ ... 1 → " $R_1 = 1$

$2 \,\underline{|\,4}$ ... 0 → " $R_2 = 0$

$2 \,\underline{|\,2}$ ... 0 → " $R_3 = 0$

$2 \,\underline{|\,1}$ ... 1 → " $R_4 = 1$ → Most Significant Bit (MSB)

0

$\Rightarrow 19_{10} = \underbrace{1}_{R_4 \text{ (MSB)}}\ \underset{R_3}{0}\ \underset{R_2}{0}\ \underset{R_1}{1}\ \underbrace{1}_{R_0 \text{ (LSB)}}{}_2$

The first remainder is LSB.

The last remainder is MSB

Continued,

- Ex) Convert 56_{10} to binary number.

$$
\begin{array}{l l}
2\,\underline{|\,56} \cdots 0 & \rightarrow LSB \\
2\,\underline{|\,28} \cdots 0 & \\
2\,\underline{|\,14} \cdots 0 & \\
2\,\underline{|\,7} \cdots 1 & \\
2\,\underline{|\,3} \cdots 1 & \\
2\,\underline{|\,1} \cdots 1 & \rightarrow MSB. \\
\quad 0 &
\end{array}
\qquad = 111000_2
$$

<u>verification</u> (reverse calculation)

$111000_2 = 1\times2^5 + 1\times2^4 + 1\times2^3$

$= 32 + 16 + 8 = 56_{10}.$

same!

Fractions

- Bin->Dec: a binary fraction can be written in the form...

$b = 0.\,b_{-1}\,b_{-2}\,b_{3}\,\ldots$

- Its corresponding decimal fraction is...

$F = b_{-1}\times2^{-1} + b_{-2}\times2^{-2} + b_{-3}\times2^{-3}\ldots$

- Ex) $b = 0.1011_2$ to its base-10 equivalent.

$F = 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} + 1\times2^{-4}$

$= 1\times0.5 + 0\times0.25 + 1\times0.125 + 1\times0.0625$

$= 0.5 + 0.125 + 0.0625$

$= 0.6875_{10}$

Continued,

- Dec->Bin: successive multiplication algorithm.
- Ex) $0.6875_{10} \rightarrow$ Bin

$0.6875 \times 2 = 1.375$ $\quad b_{-1} = 1$

$0.375 \times 2 = 0.75$ $\quad b_{-2} = 0$

$0.75 \times 2 = 1.50$ $\quad b_{-3} = 1$

$0.50 \times 2 = 1.00$ $\quad b_{-4} = 1$

since it is 0, stop!

So, $0.6875_{10} = 0.b_{-1}b_{-2}b_{-3}b_{-4} = 0.\boxed{1011}_2$ #

Round-off error

- The accuracy of the translation from decimal to binary depends on the number of bits that are used in the base-2 word.
- Ex) 4-bit binary fraction X.

$X = 0.X_{-1}X_{-2}X_{-3}X_{4}$

decimal $\Rightarrow X_{-1} \times 0.5 + X_{-2} \times 0.25 + X_{-3} \times 0.125 + X_{-4} \times 0.0625$

Let's consider two very similar binary fractions

$X_a = 0.1110_2$ & $X_b = 0.1111_2$

Continued,

substituting gives...

$X_a = 0.8750_{10}$

$X_b = 0.9375_{10}$

- The smallest resolution allowed is 0.0625. So, it is not possible to represent numbers between these decimal values using the 4-bit binary fraction.

Round-off error calculation

- Ex) 0.9270_{10} can't be represented exactly using the form $0.x_{-1}x_{-2}x_{-3}x_{-4}$

The closest one is $X_b = 0.1111_2 = 0.9375_{10}$

$\Rightarrow$ round-off error $= \dfrac{|0.9375 - 0.9270|}{0.9270} \times 100 = 1.13\%$

How to overcome this problem?

- Add more bits to the binary representation.
- Ex) 5 bit binary fraction $0.11101_2 = 0.90625_{10}$

  6 " $0.111011_2 = 0.921875_{10}$

  getting close to the actual value, but there still exists round-off error.

Hexadecimal (base-16, radix r=16) numbers

- It has 16 symbols for each digit.

  0 ... 9 A B C D E F → either upper or lower case

  $\downarrow$ A=10, B=11, C=12, D=13, E=14, F=15_{10}.

- Each digit is called "Hex" digit.
- Hex -> Bin and Bin -> Hex conversions are easy due to the fact that 4-bit binary word is equivalent to single hex digit.

Octal (base-8, radix r=8)

It has 8 symbols for each digit.

0 ~ 7.

Oct → Bin & Bin → Oct conversions are easy since 3-bit binary # is equivalent to one octal digit.

ex) $01010\ 1101_2$ to Octal

$= 2\ 5\ 5_8$

Dec to Hex conversion ⇒ successive division

ex)

$16 \,|\underline{\ 17_{10}\ }$ ··· 1

$16 \,|\underline{\ 1\ }$ ··· 1

$\ \ \ \ \ 0$

↑ → 11_{16}

Dec to Oct conversion

ex)

$8 \,|\underline{\ 17_{10}\ }$ ·· 1

$8 \,|\underline{\ 2\ }$ ·· 2

$\ \ \ \ 0$

↑ → 21_8

Examples

$H = 304E_{16}$

$\xrightarrow{\text{decimal}}$ $3\times16^3 + 0\times16^2 + 4\times16^1 + E\times16^0$

Each Hex digit can represent 4-digit binary number

$0_{16} = 0000_2 \quad \cdots \quad F_{16} = 1111_2$

ex) $1001\ 1100\ 1110\ 0101_2$ to Hex

break it into individual 4-bit groups.

$\underbrace{1001}_{9}\ \underbrace{1100}_{C}\ \underbrace{1110}_{E}\ \underbrace{0101}_{5} = 9CE5_{16} = 0x9CE5$ or $9CE5h$

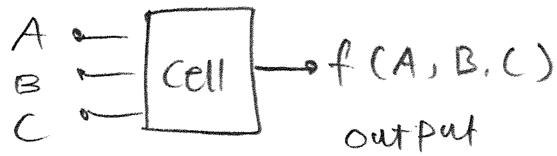
C-language convention

Assembly convention

Cells & Hierarchy

- Cells: fundamental building blocks to create a digital system.
- Logic diagrams: Graphical representations of digital networks can be used for both analysis and design.

Ex) 3 binary input variables and one output cell



output

f (A,B,C)

inputs

A, B & C

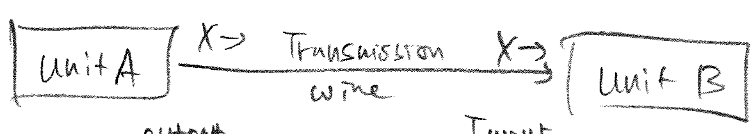
The input & output points of the cell are called ports. They allow the cell to be connected to their cells.

Ex) An example of a function

$$\begin{cases} f = 1 & \text{if } A \text{ or } B \text{ or } C = 1 \\ f = 0 & \text{otherwise.} \end{cases}$$

Creating a large system

- Use small cells to build larger cells with more complex functions.
- Signal flow paths (= transmission wires = interconnects) are used to interconnect cells.
- Ex) Serial data flow path connecting two units.

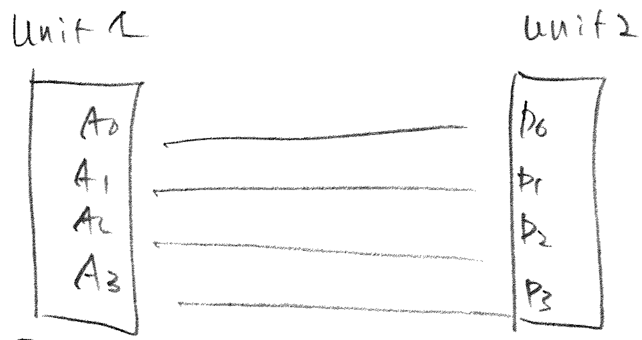


output from A

Input to B

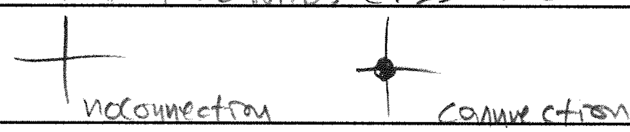
Continued,

- Ex) Parallel data flow connection.



Four separate wires are used to transmit all bits simultaneously

What if two wires cross one another? ⇒ use a black dot

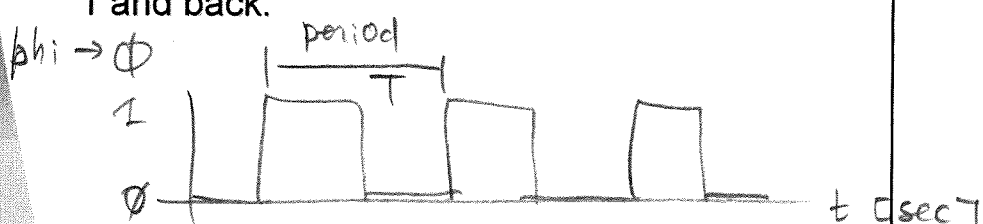


Hierarchical design

- The idea of using cells as building blocks is called "hierarchical design".
- This concept gives us a structured technique for analyzing and designing complex digital systems.
- Two approaches to designing a digital network:
  - Top-down: start with large-scale system specs then choose the cells that are needed.
  - Bottom-up: start with basic cells to build more complex cells.

System Primitives

- A system primitive: a basic function that is used several times to create the entire unit.
- Clocks: a periodic signal to synchronize operations. Always makes a transition from 0 to 1 and back.



Clock signal.

period (T) : the time for one complet cycle.

Frequency (f) : $f = 1/T$ = # of cycles/sec

usually given in units of Hertz (Hz)

Examples

ex) 1 Hz = 1cycle/sec

1GHz pentium processor's clock generates 1 G cycles/sec. Thus, the clock frequency can be used for measuring system speed.

ex) freq f = 50MHz calculate T.

$f = 1/T \Leftrightarrow T = 1/f$

$T = 1/f = \frac{1}{50 \times 10^6} = 2 \times 10^{-8}$ sec

$= 20 \times \boxed{10^{-9}}$ sec $= 20$ ns (nanosecond)

nano

Measurement units

Decimal	unit	Bin #s (for mem sizes)
10^3	K (kilo)	2^{10}
10^6	M (mega)	2^{20}
10^9	G (giga)	2^{30}
10^{-3}	m (mili)	NA
10^{-6}	μ (micro)	NA
10^{-7}	n (nano.	NA

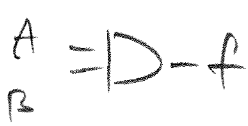
increased by 3

decreased by 3

increased by 10.

Logic gates

- Takes input bits and produces an output bit as defined by the logic operation.
- Ex) AND gate & OR gate



AND logic gate

A	B	f
0	0	0
1	0	0
0	0	0
1	1	1

Truth table

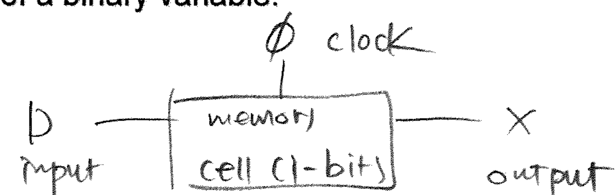
OR Gate

A	B	f
0	0	0
0	1	1
1	0	1
1	1	1

Continued,

- Ex) Memory cell: capable of capturing and holding the value of a binary variable.

ex)

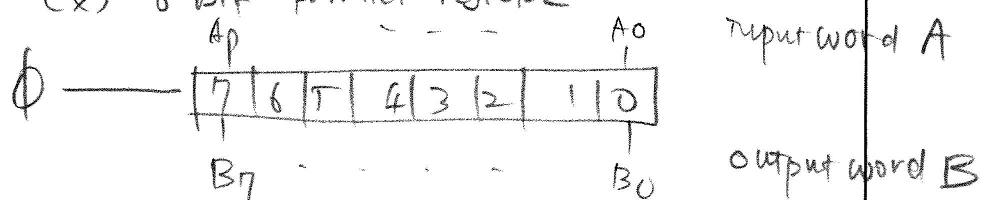


- Suppose input D = 1. Then, "1" is stored (= written or loaded) in the memory cell. Once a data bit is stored in the cell, it can be accessed (or read) at the output port X. The clock Φ is used to allow the operation to be synchronized with the rest of the system.

Continued,

- Registers: A register is a block of memory cells that can be used to store ~~words~~. a word

ex) 8-bit parallel register



- This allows for parallel loading and reading of an 8-bit word, synchronized by a clock signal Φ.

Design Metrics

- To compare different design solutions, we introduce the concepts of a metric (a unit of measurement).
  1. Temporal metric (time): ex) 1GHz vs 2GHz CPU. → printed circuit board
  2. Size: PCB based system vs. SoC. → system-on-chip.
  3. Electric power consumption: ex) cell phones (battery life 1hr vs. 2hr).

Program Completed

University of Missouri-Rolla

UMR UNIVERSITY OF MISSOURI-ROLLA
The Name. The Degree. The Difference.