Computer System Architecture

M. Morris Mano

정보통신공학과　이 명 의(A-405)
melee@kut.ac.kr

Class Overview

- Contents
  - Chap. 1 Digital Logic Circuits
    - The fundamental knowledge needed for the design of digital systems constructed with individual gates and flip-flops.
  - Chap. 2 Digital Components
    - The logical operation of the most common standard digital components(Decoders, Multiplexers, Registers, Counters, and Memories).
    - These digital components are used as building blocks for the design of larger units(Mano Machine).
  - Chap. 3 Data Representation
    - Various data types found in digital computers are represented in binary form in computer registers.
  - Chap. 4 Register Transfer and Microoperations
    - A register transfer language is used to express microoperations in symbolic form.
    - Symbols are defined for arithmetic, logic, and shift microoperations.
    - To show the hardware design of the most common microoperations, a composite arithmetic logic shift unit is developed.
  - Chap. 5 Basic Computer Organization and Design
    - The organization and design of a basic digital computer(Mano Machine).
    - Register transfer language is used to describe the internal operation of the computer.
    - By going through the detailed steps of the design presented in this chapter, the student will be able to understand the inner workings of digital computers.

Class Overview

- Chap. 6 Programming the Basic Computer
  - The 25 instructions of the basic computer to illustrate techniques used in assembly language programming.
  - Programming examples are presented for a number of data processing tasks.
  - The basic operations of an assembler are presented to show the translation from symbolic code to an equivalent binary program.
- Chap. 7 Microprogrammed Control
  - Introduction to the concept of microprogramming.
  - A specific microprogrammed control unit is developed to show by example how to write microcode for a typical set of instructions.
  - The design of the control unit is carried-out in detail.
- Chap. 8 Central Processing Unit
  - CPU as seen by the user(ISA).
  - General register organization, the operation of memory stack, variety of addressing modes, instruction format.
  - The Reduced Instruction Set Computer(RISC) concept.
- Chap. 9 Pipeline and Vector Processing
  - The concept of pipelining is explained(Pipeline can speed-up processing).
  - Both arithmetic and instruction pipeline is considered
  - Vector processing is introduced(Example: Floating-point operations using pipeline procedures)

Class Overview

- Chap. 10 Computer Arithmetic
  - Arithmetic algorithms for digital hardware implementation(addition, subtraction, multiplication, and division).
- Chap. 11 Input-Output Organization
  - Computer communication with input and output devices.
  - I/O interface units are presented to show the way that the processor interacts with external peripherals.
  - 4 modes of transfer : Programmed I/O, Interrupt initiated transfer, direct memory access, and IOP.
- Chap. 12 Memory Organization
  - The concept of memory hierarchy : cache memory, main memory, auxiliary memory
  - The organization and operation of associative memories is explained in detail.
  - Memory Management Unit : physical address and logical address mapping
- Chap. 13 Multiprocessors
  - A multiprocessor system is an interconnection of two or more CPUs.
  - Various interconnection structures are presented : Time-shared common bus, Multiport Memory, Crossbar Switch, Multistage Switching Network, Hypercube Interconnection
  - Interprocessor Arbitration : System bus, Serial Arbitration Procedure, Parallel Arbitration Logic, Dynamic arbitration Algorithms.
  - Interprocessor Communication and Synchronization : Mutual Exclusion with a Semaphore
  - Cache coherence

Class Overview

- All 3 subjects associated with computer hardware in this book
  - Computer Organization(Chap 1, 2, 3, 4)
    - H/W components operation/connection.
    - Various digital components used in the organization and design of digital computer.
  - Computer Design(Chap 5, 6, 7)
    - H/W Design/Implementation.
    - The steps that a designer must go through to design and program an elementary digital computer(Chap. 6 : program = ISA)
  - Computer Architecture(Chap 6, 8, 9, 11, 12)
    - Structure and behavior of the computer as seen by the user
      - » Information format, Instruction set, memory addressing : S/W = ISA
      - » CPU, I/O, Memory : H/W
    - Chapter in detail
      - » Chap. 6 : ISA
      - » Chap. 8 and 9 : CPU
      - » Chap. 11 : I/O
      - » Chap. 12 : Memory

Class Overview

- What is “Computer Architecture”?

  - Hennessy and Patterson, *Computer Organization and Design(1990)*

  - Computer Architecture
    - Instruction Set Architecture (ISA) : S/W
    - Machine Organization : H/W and Design

- “ISA(Instruction Set Architecture)”?
  - the attributes of a system as seen by the programmer, i.e., the conceptual structure and functional behavior, as distinct from the organization of the data flows and controls, the logic design, and the physical implementation.

    - Amdahl, Blaaw, and Brooks(1964)

  - Instructions, Addressing modes, Instruction and data formats, Register

- “Machine Organization”?
  - CPU(Control & Data path), Memory, Input/Output

Class Overview

- First Course in Computer Hardware
- Learn how a computer actually works
- Build the “Mano Machine”
- Learn one computer in detail, others are mastered easily.
- Homework:
  - Solve the even number of problems
  - Due at the beginning of the next class
- Optional “Mano Machine” Design Report
- Grade:
  - Homework(20%)
  - Optional Report(10%)
  - Mid/Final Exam(each 30%)
  - Class Participation(10%)
- Lecture Notes: http://microcom.kut.ac.kr

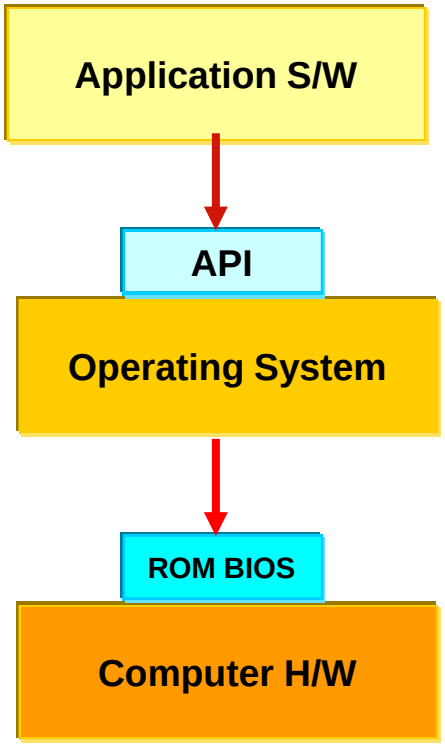
8 Student Types

- Insecure: 25 %
- Silent: 20 %
- Independent: 12 %
- Friendly: 11 %
- Obedient: 10 %
- Heroic: 9 %
- Critic: 9 %
- Unmotivated: 4 %

- Michigan State University

1-1 Digital Computers

- Digital Computer = H/W + S/W
  - Digital
    - implies that the information in the computer is represented by variables that take a limited number of discrete values.
    - the decimal digits 0, 1, 2,….,9, provide 10 discrete values, but digital computers function more reliably if only two states are used.
    - because of the physical restriction of components, and because human logic tends to be binary(true/false, yes/no), digital component are further constrained to take only two values and are said to be binary.
  - Bit = binary digit : 0/1
  - Program(S/W)
    - A sequence of instruction
    - S/W = Program + Data
      - » The data that are manipulated by the program constitute the data base
    - Application S/W = DB, word processor, Spread Sheet
    - System S/W = OS, Firmware, Compiler, Device Driver



1-1 Digital Computers

continued

- Computer Hardware(H/W)
  - CPU
  - Memory
    - Program Memory(ROM)
    - Data Memory(RAM)
  - I/O Device
    - Interface: 8251 SIO, 8255 PIO, 6845 CRTC, 8272 FDC, 8237 DMAC, 8279 KDI
    - Input Device: Keyboard, Mouse, Scanner
    - Output Device: Printer, Plotter, Display
    - Storage Device(I/O): FDD, HDD, MOD

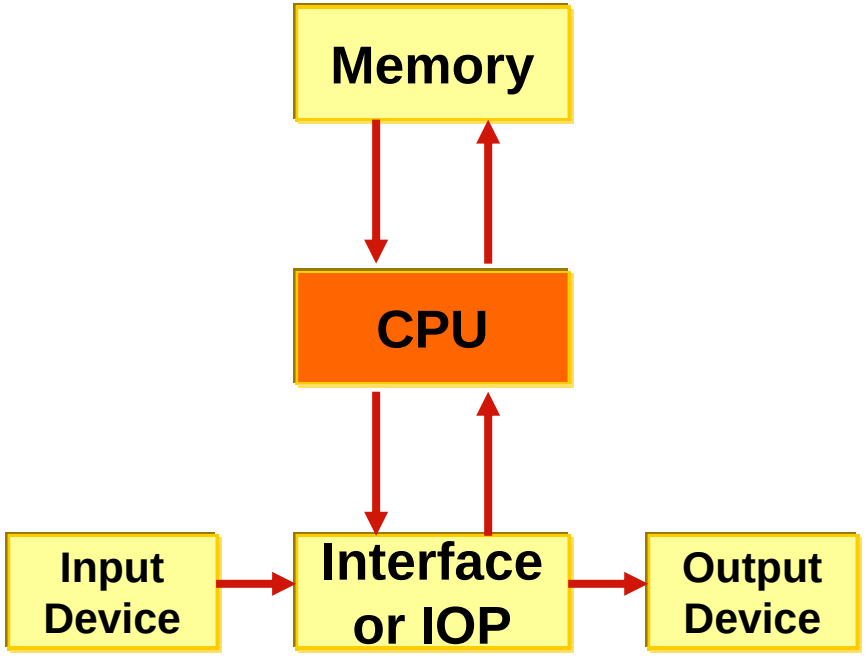
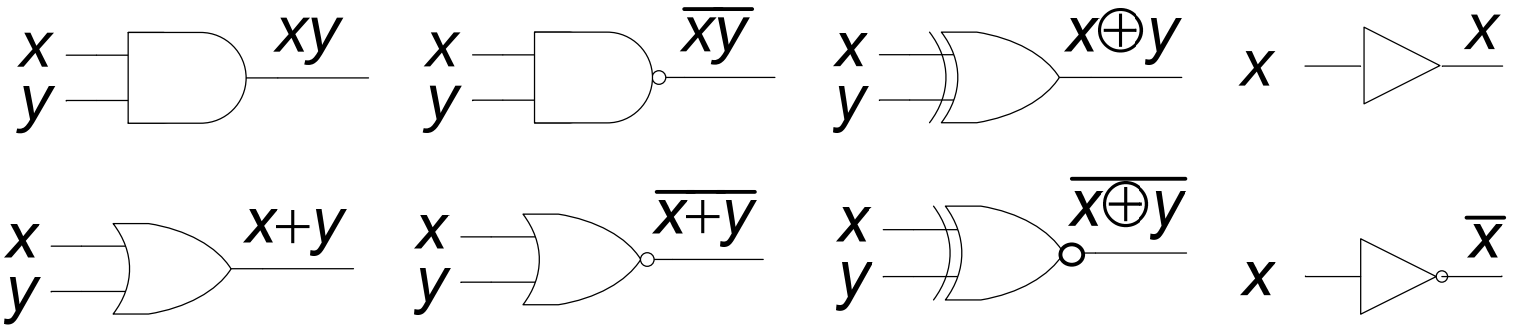


Figure 1-1 Block Diagram of a digital Computer

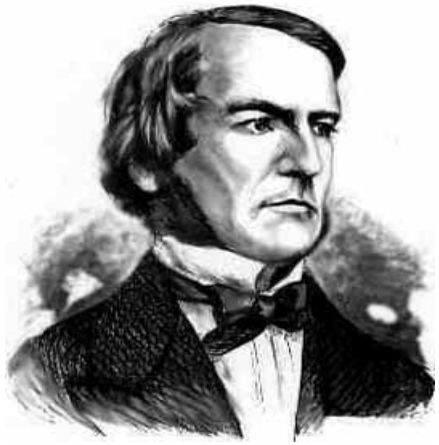
1-2 Logic Gates

- ADC(Analog to Digital Conversion)
  - Signal ⟶ Physical Quantity ⟶ Binary Information
    V, A, F, 거리 Discrete Value
    0 : 0.5v
    1 : 3v
- Gate
  - The manipulation of binary information is done by logic circuit called "gate".
  - Blocks of H/W that produce signals of binary 1 or 0 when input logic requirements are satisfied.
  - *Digital Logic Gates : Fig. 1-2*
    - AND, OR, INVERTER, BUFFER, NAND, NOR, XOR, XNOR



1-3 Boolean Algebra

- Boolean Algebra
  - Deals with binary variable (A, B, x, y: T/F or 1/0) + logic operation (AND, OR, NOT…)
- Boolean Function: *variable + operation*
  - F(x, y, z) = x + y'z
- George Boole
  - Born: 2 Nov 1815 in Lincoln, Lincolnshire, England
  - Died: 8 Dec 1864 in Ballintemple, County Cork, Ireland



1-3 Boolean Algebra

- Boolean Function: variable + operation
  - F(x, y, z) = x + y'z

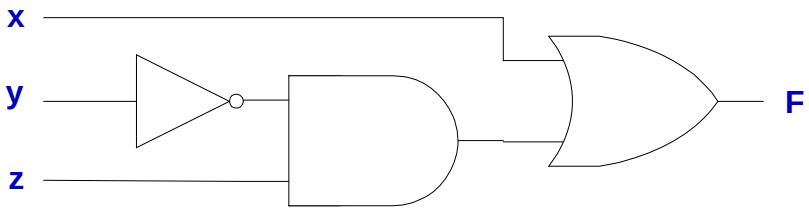
- Truth Table: *Fig. 1-3(a)*
  Relationship between a function and variable

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

2^n Combination

Variable n = 3

- Logic Diagram: *Fig. 1-3(b)*
  Algebraic Expression ⇒
  Logic Diagram(gates로 표현)



- Purpose of Boolean Algebra
  - To facilitate the analysis and design of digital circuit
- Boolean function = Algebraic form = convenient tool
  - Truth table (relationship between binary variables : *Fig 1-3a*) → Algebraic form
  - Logic diagram (input-output relationship : *Fig. 1-3b*) → Algebraic form
  - Find simpler circuits for the same function : *by using Boolean algebra rules*
- Boolean Algebra Rule : *Tab. 1-1*

- Operation with 0 and 1: **x + 0 = x , x + 1 = 1 , x • 1 = x , x • 0 = 0**

- Idempotent Law: **x + x =x , x • x = x**

- Complementary Law: **x + x' = 1 , x • x' = 0**

- Commutative Law: **x + y = y + x , x • y = y • x**

- Associative Law: **x + (y + z) = (x + y) + z , x • (y • z) = (x • y) • z**

- Distributive Law: **x • (y+ z) = (x • y) + (x • z) , x + (y • z) = (x + y) • (x + z)**

- DeMorgan's Law: **(x + y)' = x' • y' , (x • y)' = x' + y'**

$(A \cup B)^c = A^c \cap B^c$

General Form: $(x_1 + x_2 + x_3 + \dots x_n)' = x_1' \cdot x_2' \cdot x_3' \cdot \dots x_n'$

$(x_1 \cdot x_2 \cdot x_3 \cdot \dots x_n)' = x_1' + x_2' + x_3' + \dots x_n'$

- [예제]
  - F= AB' + C'D + AB' + C'D
    = x + x (let x= AB' + C'D)
    = x
    = AB' + C'D

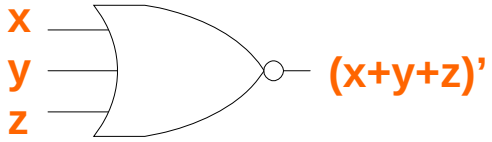
- [예제]
  - F= ABC + ABC' + A'C
    = AB(C + C') + A'C
    = AB + A'C

Fig. 1-6(a)

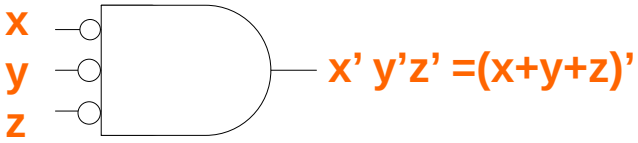
Fig. 1-6(b)

1 inverter, 1 AND gate 감소

- *Fig. 1-4 2 graphic symbols for NOR gate*

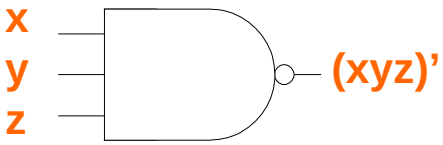


(a) OR-invert

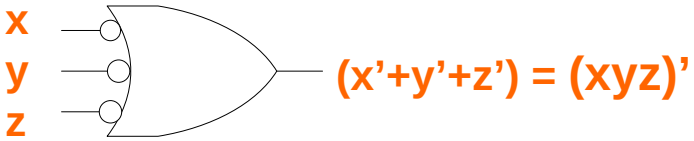


(b) invert-AND

- *Fig. 1-5 2 graphic symbols for NAND gate*



(a) AND-invert



(b) invert-OR

1-4 Map Simplification

- Karnaugh Map(K-Map)
  - Map method for simplifying Boolean expressions
- Minterm / Maxterm
  - Minterm : n variables ***product*** (x=1, x'=0)
  - Maxterm : n variables ***sum*** (x=0, x'=1)
- 2 variables example

x	y	Minterm		Maxterm	
0	0	x'y'	m_0	x + y	M_0
0	1	x'y	m_1	x + y'	M_1
1	0	x y'	m_2	x'+ y	M_2
1	1	x y	m_3	x'+ y'	M_3

$m_0 + m_1 + m_2 + m_3$ $\quad$ $M_0 \bullet M_1 \bullet M_2 \bullet M_3$

- F = $\underset{m_1}{x'y} + \underset{m_3}{xy}$

  $= \Sigma(1,3)$ ($m_1 + m_3$)

  $= \Pi(0,2)$ **(Complement =** $M_0 \bullet M_2$ **)**

Map

2 variables

A \ B	0	1
0	0	1
1	2	3

3 variables

A \ BC	00	01	11	10
0	0	1	3	2
1	4	5	7	6

4 variables

AB \ CD	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

5 variables

A \ BCDE	000	001	011	010	110	111	101	100
00	0	1	3	2	6	7	5	4
01	8	9	11	10	14	15	13	12
11	24	25	27	26	30	31	29	28
10	16	17	19	18	22	23	21	20

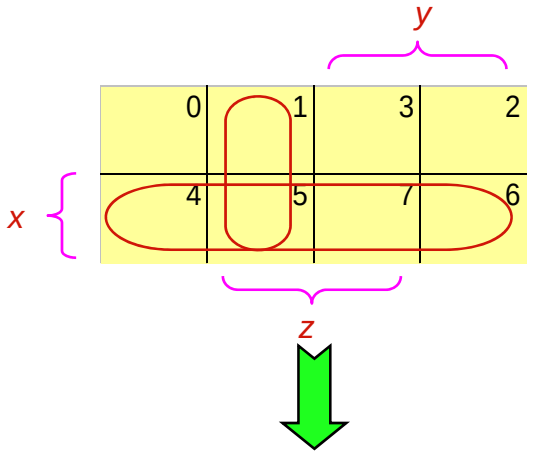
- [예제] F= x + y'z

(1) Truth Table

x	y	z	F	Minterm
0	0	0	0	m_0
0	0	1	1	m_1
0	1	0	0	m_2
0	1	1	0	m_3
1	0	0	1	m_4
1	0	1	1	m_5
1	1	0	1	m_6
1	1	1	1	m_7

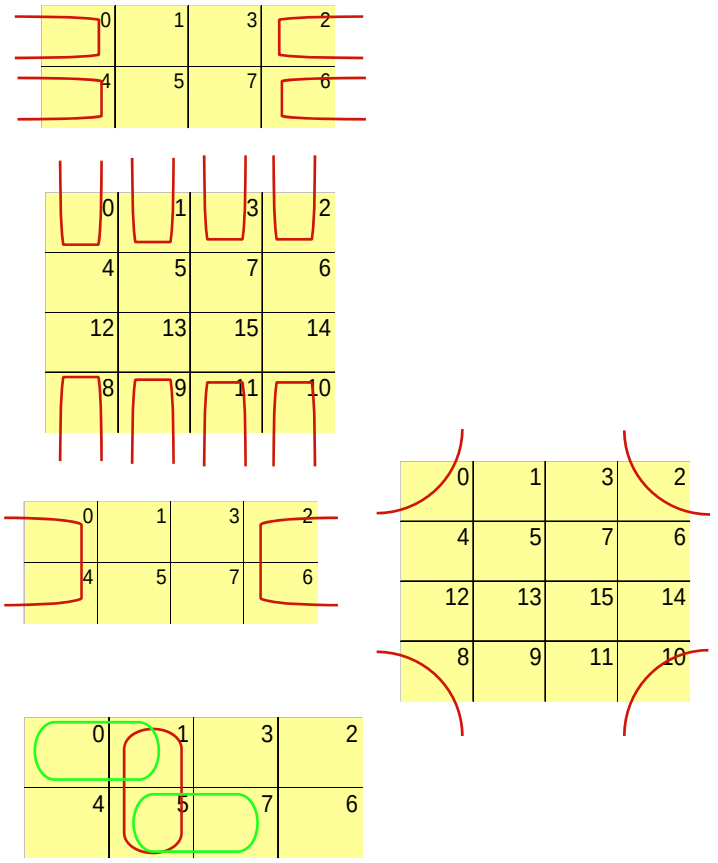
(2) $F(x, y, z) = \Sigma(1,4,5,6,7)$

(3)



F= x + y'z

- Adjacent Square
  - Number of square = 2^n (2, 4, 8, ….)
  - The squares at the extreme ends of the same horizontal row are to be considered adjacent
  - The same applies to the top and bottom squares of a column
  - The four corner squares of a map must be considered to be adjacent
  - Groups of combined adjacent squares may share one or more squares with one or more group



- [예제] $F(A,B,C)=\Sigma(3,4,6,7)$
  - F=AC' + BC

- [예제] $F(A,B,C)=\Sigma(0,2,4,5,6)$
  - F=C' + AB'

- [예제] $F(A,B,C,D)=\Sigma(0,1,2,6,8,9,10)$
  - F=B'D' + B'C' + A'CD'

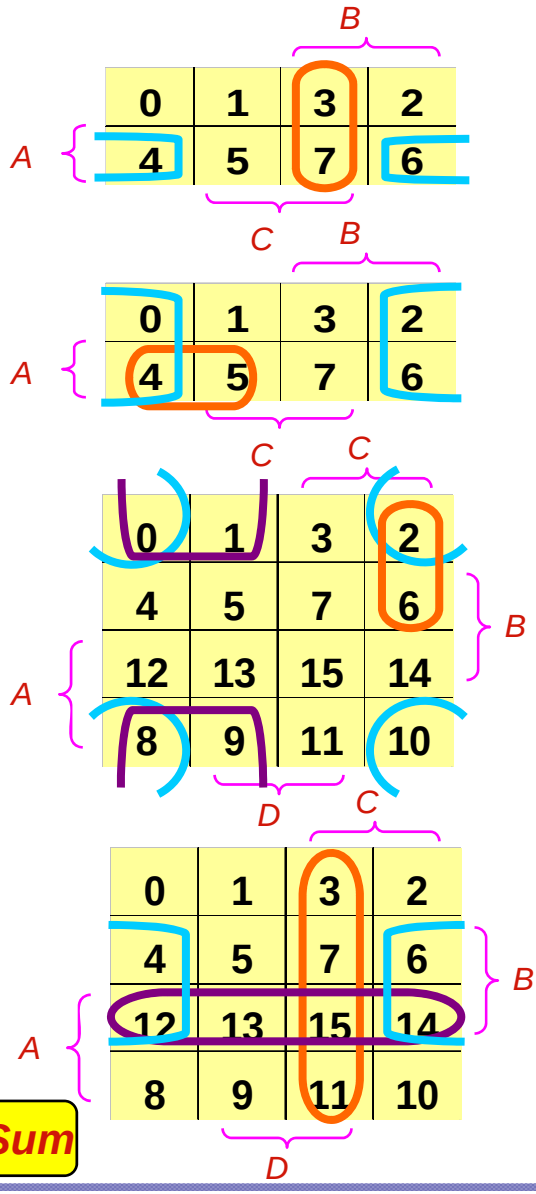
- Product-of-Sums Simplification

  $F(A,B,C,D)=\Sigma(0,1,2,5,8,9,10)$

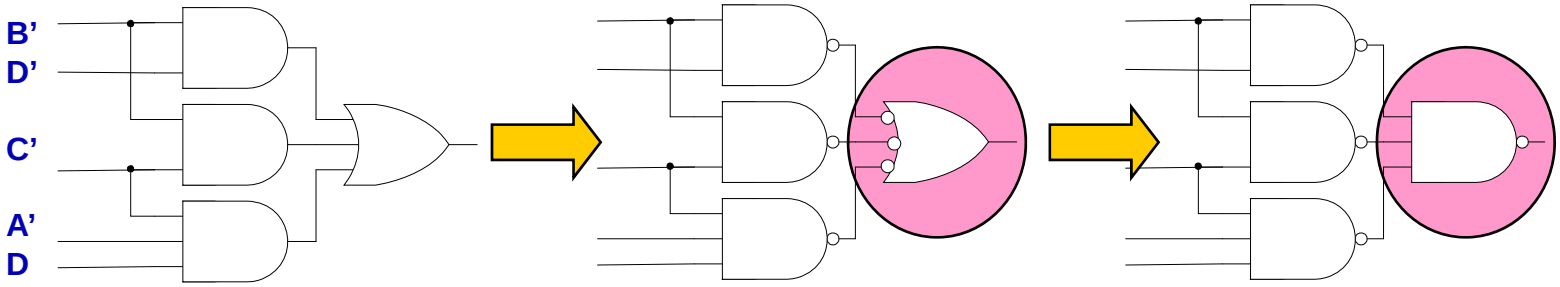
  F=B'D' + B'C' + A'C'D — ***Sum of product***

  F'=AB + CD + BD'(square marked 0's)

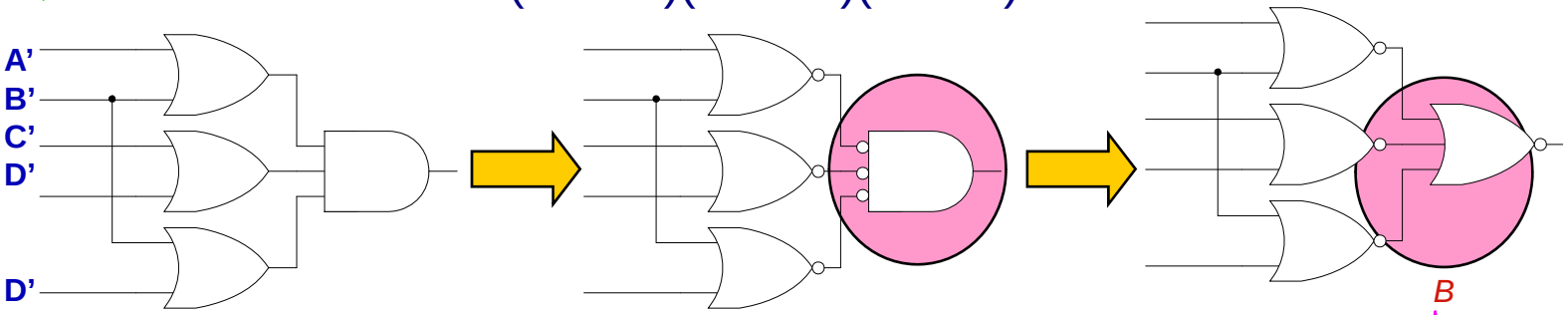
  F''(F)=(A' + B')(C' + D')(B' + D) 전개 — ***Product of Sum***



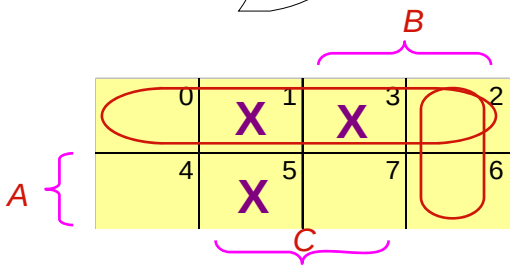
- NAND Implementation
  - Sum of Product : F=B'D' + B'C' + A'C'D



- NOR Implementation
  - Product of Sum : F=(A' + B')(C' + D')(B' + D)

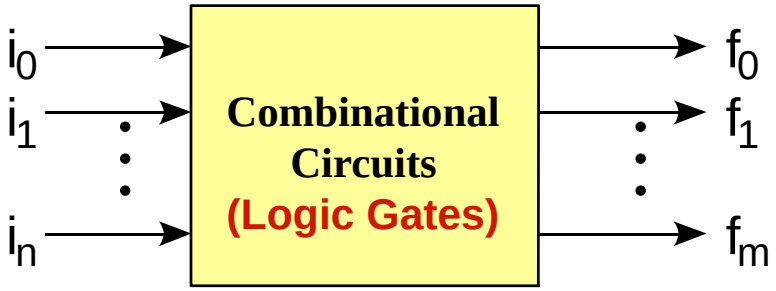


- Don't care conditions
  - F(A,B,C)=Σ(0, 2, 6), d(A,B,C)= Σ(1, 3, 5)
  - F=A' + BC'= Σ(0, 1, 2, 3, 6)



1-5 Combinational Circuits

- Combinational Circuits
  - A connected arrangement of *logic gates* with a set of inputs and outputs
  - *Fig. 1-15 Block diagram of a combinational circuit*

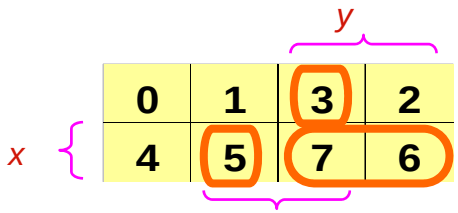


- Analysis
  - Logic circuits diagram → Boolean function or Truth table
- Design(Analysis의 반대) — Experience
  - 1. The Problem is stated
  - 2. I/O variables are assigned
  - 3. Truth table(I/O relation)
  - 4. Simplified Boolean Function(Map 과 Boolean 대수 이용)
  - 5. Logic circuit diagram

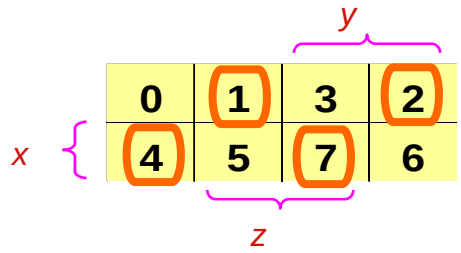
- Design Example : Full Adder
  - 1. Full adder is a combinational circuits that forms the arithmetic sum of three input bit *(Carry considered)*
  - 2. 3 Input(x, y, z), 2 Output(S: sum, C: carry)
  - 3. Truth Table

Input			Output	
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

  - 4. Simplification



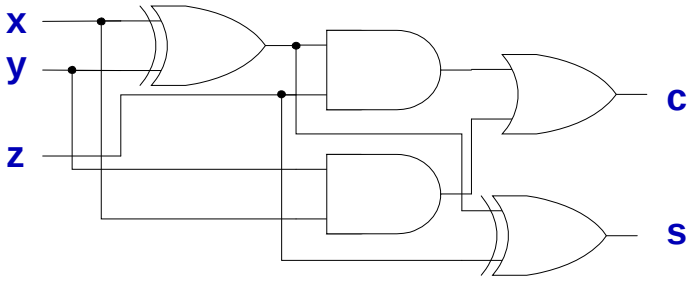
C= xy'z + x'yz + xy
=z(xy' + x'y) + xy
=z(x ⊕ y) + xy



S=xy'z' + x'y'z + xyz + x'yz'
= z'(xy' + x'y) + z(x'y' + xy)
= z'(x ⊕ y) + z(x ⊕ y)'
=a'b + ab' (let a=z, b=x ⊕ y)
=x ⊕ y ⊕ z

(x ⊕y)'=(xy'+x'y)'
=(x'+y)(x+y')
=x'x+x'y'+xy+yy'
=x'y'+xy

  - 5. Logic circuit diagram

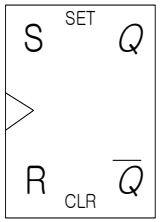


1-6 Flip-Flops

Combinational Circuit = Gate
Sequential Circuit = Gate + F/F

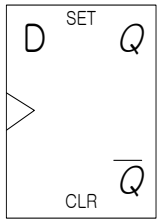
- Flip-Flop
  - The *storage elements* employed in clocked *sequential circuit*
  - A binary cell capable of storing one bit of information

- SR*(Set/Reset)* F/F



S	R	Q(t+1)	
0	0	Q(t)	no change
0	1	0	clear to 0
1	0	1	set to 1
1	1	?	Indeterminate

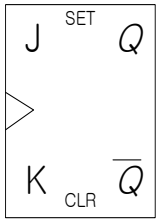
- D*(Data)* F/F



D	Q(t+1)	
0	0	clear to 0
1	1	set to 1

  - "no change" condition이 없다 : Q(t+1)=D
    - 해결방법 : 1) Disable Clock
      2) Feedback output into input p.52

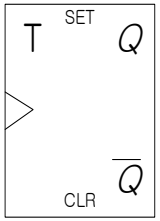
- JK*(Jack/King)* F/F



J	K	Q(t+1)	
0	0	Q(t)	no change
0	1	0	clear to 0
1	0	1	set to 1
1	1	Q(t)'	Complement

  - JK F/F is a refinement of the SR F/F
  - The indeterminate condition of the SR type is defined in complement

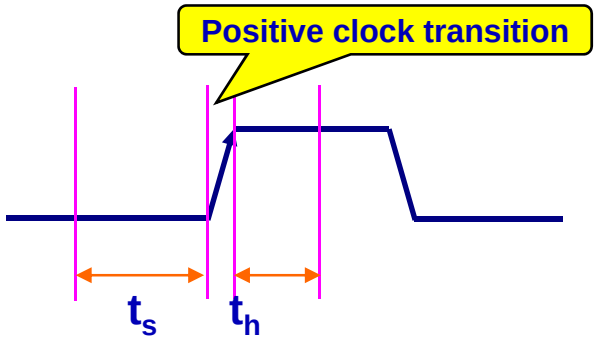
- T*(Toggle)* F/F



T	Q(t+1)	
0	Q(t)	no change
1	Q'(t)	Complement

  - T=1(J=K=1), T=0(J=K=0) 이면 JK F/F
  - 수식 표현 : Q(t+1)= Q(t) ⊕ T xor

- Edge-Triggered F/F
  - State Change : *Clock Pulse*
    - Rising Edge(positive-edge transition)
    - Falling Edge(negative-edge transition)



  - Setup time(20ns)
    - minimum time that D input must remain at constant value before the transition.
  - Hold time(5ns)
    - minimum time that D input must not change after the positive transition.
  - Propagation delay(max 50ns)
    - time between the clock input and the response in Q
    - 일반 논리 gate에서는 2-20 ns이며 setup 및 hold time은 F/F에서만 정의되며 일반 논리 gate에서는 정의되지 않음.
  - Master-Slave F/F
    - 2개의 F/F을 사용(Slave 와 Master F/F)하며 negative-edge transition 사용
    - 위와 같이 사용하는 이유: **Race 현상**을 방지

- Race 현상
  - 조건 - Hold time > Propagation delay
  - 증상 - 0 과 1을 반복하다가 Unstable한 상태가 된다
  - 해결책 - Edge triggered F/F *(with little or no hold time)* 또는 Master/Slave F/F 사용
  - 예제 : 7470(J-K Edge triggered F/F), 7471(J-K Master/Slave F/F)

- Excitation Table
  - Required input combinations for a given change of state
  - Present State 와 Next State로 표현

SR F/F

Q(t)	Q(t+1)	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

JK F/F

Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

D F/F

Q(t)	Q(t+1)	D
0	0	0
0	1	1
1	0	0
1	1	1

T F/F

Q(t)	Q(t+1)	T
0	0	0
0	1	1
1	0	1
1	1	0

Don't Care

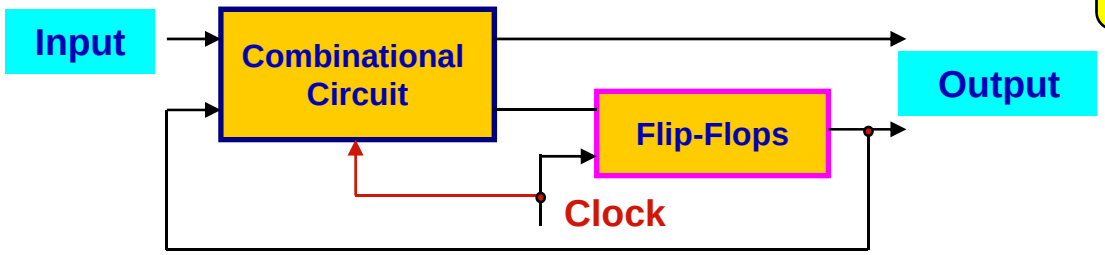
0 : Set to 1
1 : Complement

1 : Clear to 0
0 : No change

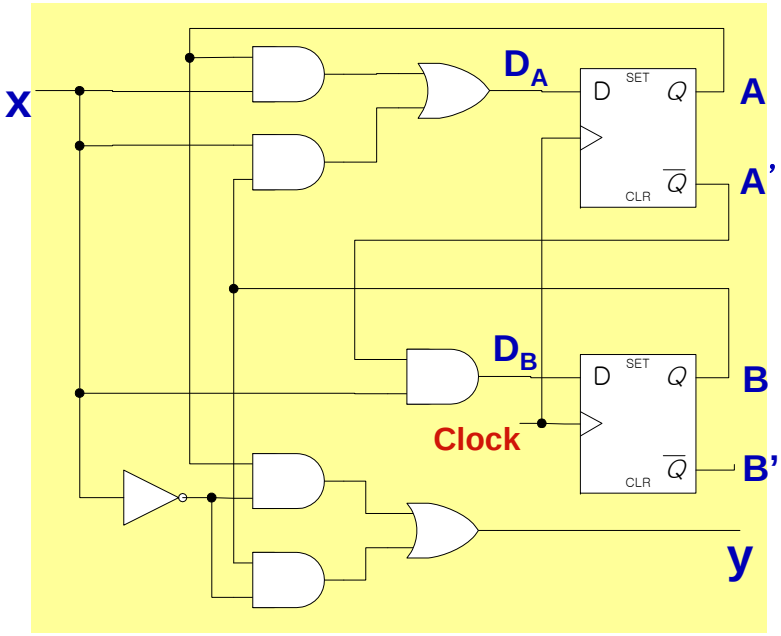
1-7 Sequential Circuits

- A sequential circuit is an interconnection of F/F and Gate
- Clocked synchronous sequential circuit

Combinational Circuit = Gate
Sequential Circuit = Gate + F/F



- Flip-Flop Input Equation
  - Boolean expression for F/F input
  - Input Equation 예제
    - $D_A = Ax + Bx$, $D_B = A'x$
  - Output Equation
    - $y = Ax' + Bx'$
  - *Fig. 1-25 Example of a sequential circuit*



State Table

- Present state, input, next state, output 표현

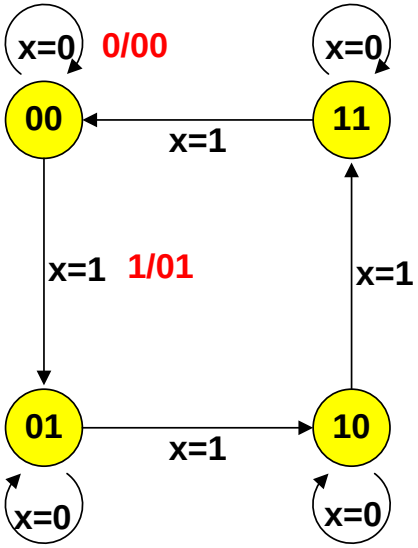
Input Equ. = Next State

Present State		Input			Input Equ.		Next State		Output
A	B	x	Ax	Bx	D_A	D_B	A	B	y
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	1	0
0	1	0	0	0	0	0	0	0	1
0	1	1	0	1	1	1	1	1	0
1	0	0	0	0	0	0	0	0	1
1	0	1	1	0	1	0	1	0	0
1	1	0	0	0	0	0	0	0	1
1	1	1	1	1	1	0	1	0	0

Design Example: Binary Counter

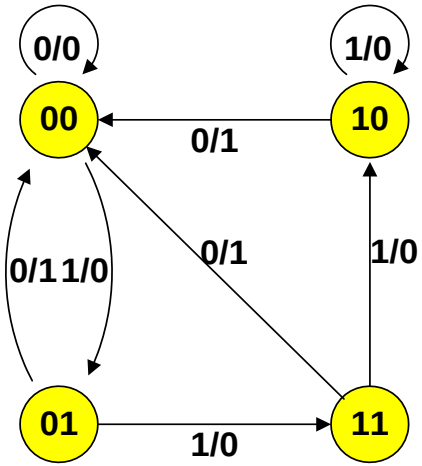
- x=1: 00, 01, 10, 11, 00, 01, …..
  x=0: no change
- State Diagram:

4 state(00, 01, 10, 11)



State Diagram

- Graphical representation of state table
  - Circle(state), Line(transition), I/O(input/output)



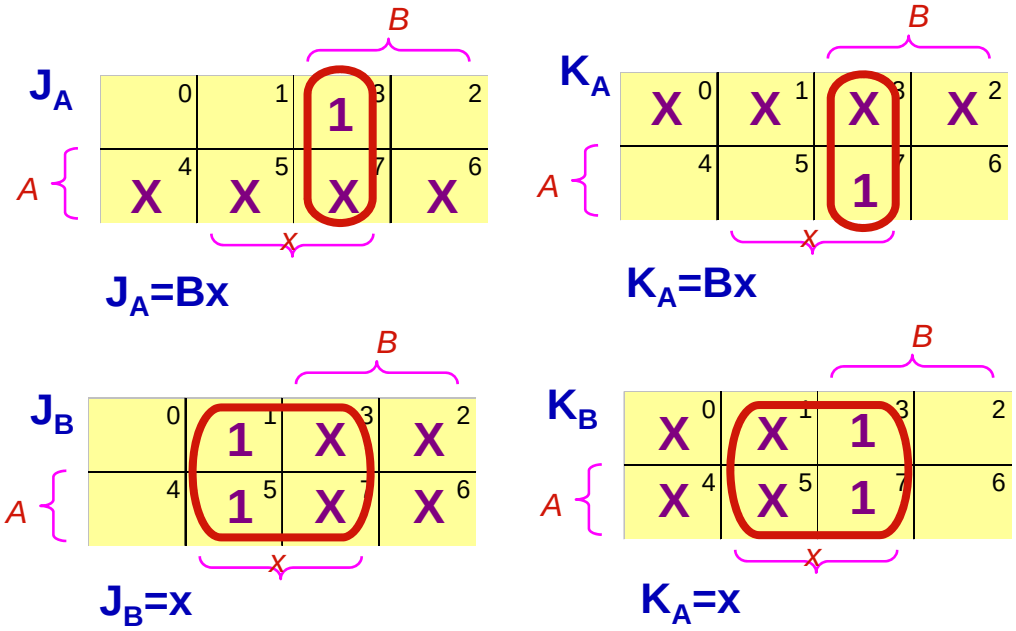
JK F/F			
Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Next State = Output

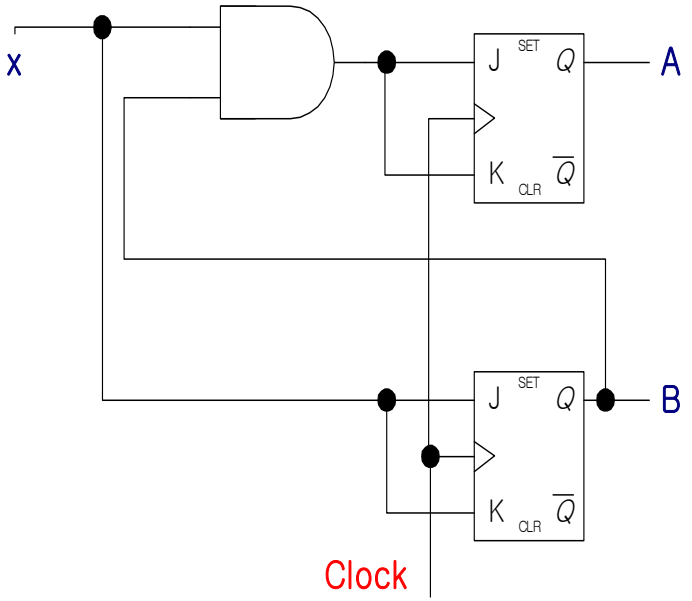
- Excitation Table(2 bit counter = 2 F/F)

Present State		Input	Next State		F/F Input			
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	0	0	x	0	x
0	0	1	0	1	0	x	1	x
0	1	0	0	1	0	x	x	0
0	1	1	1	0	1	x	x	1
1	0	0	1	0	x	0	0	x
1	0	1	1	1	x	0	1	x
1	1	0	1	1	x	0	x	0
1	1	1	0	0	x	1	x	1

- Map for simplification
  - Input variable: A, B, x



- Logic Diagram



- Sequential Circuit Design Procedure
  - 1-5 절 참고(Combinational Circuit Design)
  - Sequential Circuit은 절차 3에서 State diagram및 State table 이용
  - # of rows : 2^{m+n} (m - State 수, n - Input 수)

1. The Problem is stated
2. I/O variables are assigned
3. Truth table(I/O relation)
4. Simplified Boolean Function
5. Logic circuit diagram