디지털 논리회로

1장

1-2

1, Analog representation : continuously variable, proportional indicator
digital " : discrete

1-5

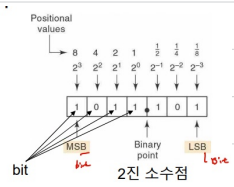
1, advantages of digital techniques
ⅰ, ease of design
ⅱ, well suited for storing information
ⅲ, Accuracy and precision(정밀도)
ⅳ, Programmable operation
ⅴ, less affected by noise
ⅵ, ease of fabrication on IC chips.

2, limitations of digital techniques
ⅰ, takes time

1.6 Digital number systems

ⅰ, decimal : 수값이 위치에 따라 결정되는 위치 값 체계 : positional value system
위치값은 10의 거듭제곱으로 표현
· Most significant digit (MSD) : 최상위숫, 가장 큰 가중치를 가지는 맨 왼쪽의 숫자
· Least " (LSD) : 최하위숫 작은 이것저

ⅱ, binary : position value system



· 자릿수 → bit라 부르기
· Most significant bit (MSB) : 가장 왼쪽 비트
least (LSB) : 오른쪽
· 2진법의 계수 : 주어진 비트(n)으로 표현할 수 있는 수의 갯수 : 2^n-1
· representing binary quantities : light bulb, diode, relay(switch), transistor, photocell
CD, oscilloscope, logic analyzer

1-8

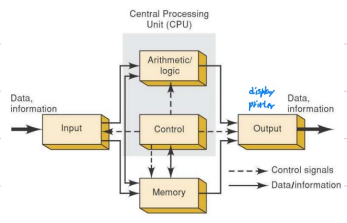
1, parallel transmission : simultaneously, fast, requires many data lines.
2, serial : 하나의 선으로 하나의 비트씩 보냄, LSB가 먼저 보내짐

1-9 Memory

1, temporarily : RAM, permanent : ROM
2, method
ⅰ, magnetic method : hard disc
ⅱ, optical(레이저) method : CD, DVD
ⅲ, electronic method : SRAM

1-10 digital Computer

1,



2장

121171398 신건희

2-1 ~ 2-3

Conversion (decimal, binary, hexadecimal)

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

1, Hexadecimal number system
allows convenient handling of long binary strings, using groups of 4bits

16^4	16^3	16^2	16^1	16^0	16^{-1}	16^{-2}	16^{-3}	16^{-4}

Hexadecimal point

2, binary to
ⅰ, decimal : double-double method
자릿수 곱하고 더하기 hex → dec 와 같음

Given: $1\ 1\ 0\ 1\ 1_2$
Results:
$1 \times 2 = 2$
$+1$
$3 \times 2 = 6$
$+0$
$6 \times 2 = 12$
$+1$
$13 \times 2 = 26$
$+1$
27_{10}

ⅱ, hex : grouping bits in four starting with LSB → convert to hex equivalent

$1110100110_2 = \underbrace{0\ 0\ 1\ 1}_{3}\ \underbrace{1\ 0\ 1\ 0}_{A}\ \underbrace{0\ 1\ 1\ 0}_{6}$
$= 3A6_{16}$

3, decimal to
ⅰ, binary : repeat division until quotient is zero

$\frac{25}{2} = 12$ + remainder of 1 (LSB)
$\frac{12}{2} = 6$ + remainder of 0
$\frac{6}{2} = 3$ + remainder of 0
$\frac{3}{2} = 1$ + remainder of 1
$\frac{1}{2} = 0$ + remainder of 1 (MSB)
$25_{10} = 1\ 1\ 0\ 0\ 1_2$

ⅱ, hex : repeat division until quotient is zero

$\frac{423}{16} = 26$ + remainder of 7
$\frac{26}{16} = 1$ + remainder of 10
$\frac{1}{16} = 0$ + remainder of 1
$423_{10} = 1A7_{16}$

4, hex to
ⅰ, binary : $9F2_{16} = 9\ F\ 2$
$= 1\ 0\ 0\ 1\ \ 1\ 1\ 1\ 1\ \ 0\ 0\ 1\ 0$
$= 100111110010_2$

ⅱ, decimal : multiplying each hex digit by its positional weight.

$$2AF_{16} = 2 \times 16^2 + 10 \times 16^1 + 15 \times 16^0$$
$$= 512 + 160 + 15$$
$$= 687_{10}$$

2-4 Binary coded decimal (BCD, 2진화 10진코드)

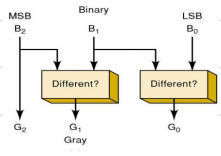
· it is decimal number with each digit encoded to its binary equivalent
ⅰ, decimal → BCD
874₁₀ to BCD : 8 → 1000, 7 → 0111, 4 → 0100
ⅱ, BCD → decimal
0110 1000 0011 1001 to decimal : 0110 1000 0011 1001 → 6 8 3 9

2-5 the gray code

· used in applications where numbers change rapidly
· only one bit changes from each value to the next

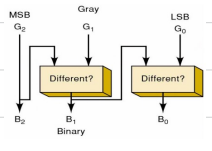
B_2	B_1	B_0	G_2	G_1	G_0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

ⅰ, binary → gray



다르면 1
같으면 0

ⅱ, gray → binary



〃

2-7 byte, nibble, word

i) byte : most microcomputer handle any storage information in groups of eight digit

8bits = 1byte

ii) nibble : binary number are often broken into groups of four bit.

4bits = a nibble

iii) word : group of bits, depends on the size of the data pathway.

2-8 Alphanumeric Codes : represents characters and functions found on a computer keyboard

AsCII : 가장 널리 사용, 7비트 ⇒ 2^7=128 possible code groups

코드는 8bit로 저장되므로 AscII 코드의 왼쪽에 0 이추가

2-9 Parity method for error detection ; requires the addition of an extra bit to a code group

only detection. no Correction

- Parity bit ; it can be either a 0 or 1, depending on the number of 1s in the code group

i) even method : 1의 개수가 짝수 → 0
" 홀수 → 1
⇒ 1의 개수를 짝수로 만들어 줌

ii) odd method : 1의 개수가 짝수 → 1
" 홀수 → 0

2-10 asynchronous (비동기식) data communication

- ASCII character must be framed so the receiver knows where the data begins and ends
  - i) first bit ; start bit (logic 0)
  - ii) LSB → MSB
  - iii) after MSB, parity bit is appended to check for transmission errors
  - iv) Last bit ; stop bit (logic 1)

Exercise (c) (c,d)
- ch2 : 12, 16, 28, 37
- ch3 : 6, 12, 19, 24

prove 3-10 [15(a) and 15(b)] using the Boolean Algebra

3강

1, Boolean algebra

i,

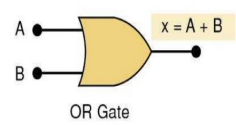
Logic 0	Logic 1
False	True
Off	On
LOW	HIGH
No	Yes
Open switch	Closed switch

ii) 3 basic logic operations : or, AND, NOT

2, Operations

i) or : X = A + B
either 1 → 1

A	B	X = A + B
0	0	0
0	1	1
1	0	1
1	1	1

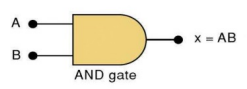


output will go high when any input is high

ii) AND : X = A·B
X is truth when A and B are true

A	B	X = A · B
0	0	0
0	1	0
1	0	0
1	1	1

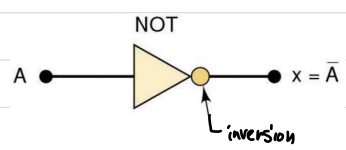
Truth table —



Gate symbol.

output will go high when all inputs are high

iii) NOT : X = $\bar{A}$
- commonly call inverter
- only a single input

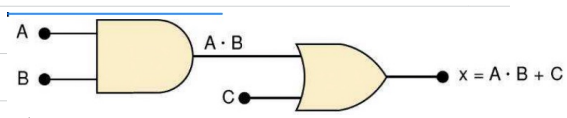


A	X = $\bar{A}$
0	1
1	0

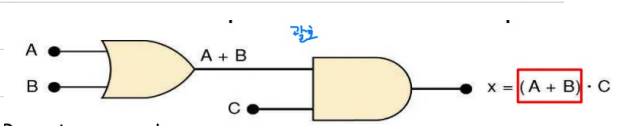
opposite to the logic level of this input

3, 계산

i) AND operation will be performed first



ii) unless there is a parenthesis in the expression



4. evaluating logic circuit outputs

i) Rules : 1. (괄호를 뺀) 변수의 모든 역변환을 수행
2. 괄호내의 연산 수행
3. 괄호가 없다면 AND 연산 수행 후 OR 연산 수행
4. 연산식에 바가 있으면 바부 연산 수행 후 결과를 역변환 해줌

$A = 0, B = 1, C = 1, \text{and } D = 1$

$x = AB\overline{C}(\overline{A+D})$

$x = 0 \cdot 1 \cdot 1 \cdot \overline{(0+1)}$

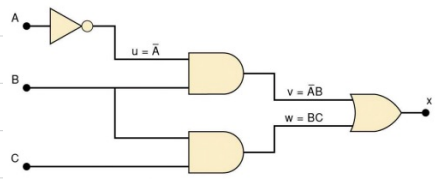
$x = 1 \cdot 1 \cdot 1 \cdot \overline{(0+1)}$

$x = 1 \cdot 1 \cdot 1 \cdot (\bar{1})$

$x = 1 \cdot 1 \cdot 1 \cdot 0$

$x = 0$

ii) truth table



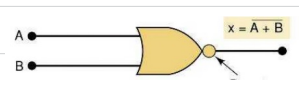
⇒

A	B	C	u = $\bar{A}$	v = $\bar{A}$B	w = BC	X = v+w
0	0	0	1	0	0	0
0	0	1	1	0	0	0
0	1	0	1	1	0	1
0	1	1	1	1	1	1
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	0	0	0	0
1	1	1	0	0	1	1

5. NOR, NAND Gates

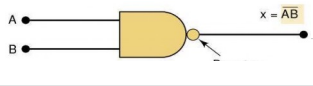
simplifying the writing of Boolean expressions

NOR :



A	B	A + B	$\overline{A+B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

NAND



A	B	AB (AND)	$\overline{AB}$ (NAND)
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

6. Boolean Theorems

- help us to simplify logic expressions and logic circuits

i) AND Gates : $X \cdot 0 = 0$ | $X \cdot 1 = X$ | $X \cdot X = X$ | $X \cdot \bar{X} = 0$

ii) OR Gates : $X + 0 = X$ | $X + 1 = 1$ | $X + X = X$ | $X + \bar{X} = 1$

iii) Commutative laws (교환법칙) : $x + y = y + x$, $x \cdot y = y \cdot x$

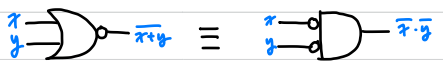
iv) Associative laws (결합법칙) : $x + (y + z) = (x + y) + z = x + y + z$, $x(yz) = (xy)z = xyz$

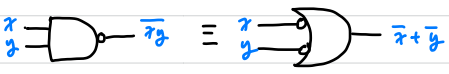
v) Distribute laws (분배법칙) : $x(y + z) = xy + xz$, $(w + x)(y + z) = wy + xy + wz + xz$

vi) multivariable theorems : $x + xy = x$, $x + \bar{x}y = x + y$, $\bar{x} + xy = \bar{x} + y$

외워~

7. DeMorgan's Theorems

i) $\overline{(x+y)} = \bar{x} \cdot \bar{y}$ 

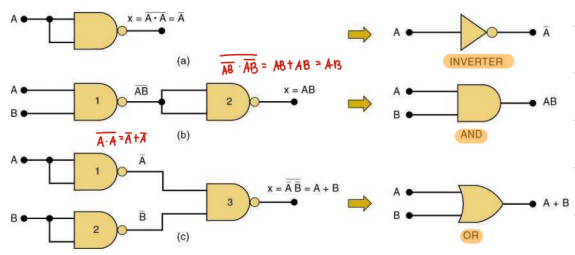
ii) $\overline{(x \cdot y)} = \bar{x} + \bar{y}$ 

cf, $\overline{A \cdot B \cdot C} = \bar{A} + \bar{B} + \bar{C}$, $\overline{x+y+z} = \bar{x} \cdot \bar{y} \cdot \bar{z}$

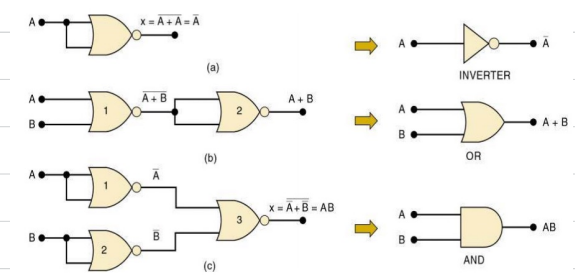
8. Universality of NAND and NOR gates

NAND or NOR gate $\xrightarrow{\text{create}}$ OR, AND, INVERT

i) using only NAND gates

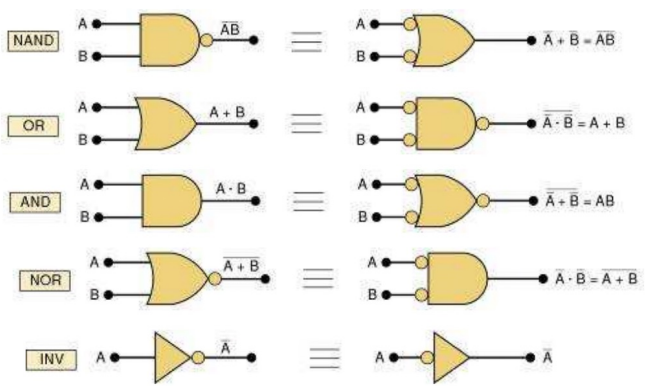


ii) using only NOR gates



9. Alternate Logig-Gate (대안 논리 회로)

i) 방법 : 1, 표준 기호의 입출력을 역변환 시킴
2, 연산기호 변환 AND ⇄ or (인버터는 X)



ii) NAND and NOR : inverting gates (역연산 게이트)
표준, 대안 기호 모두 입력 또는 출력 한쪽에만 버블 존재

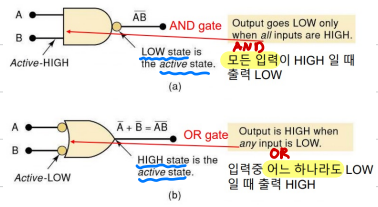
AND and OR : noninverting gates (역연산 없는 게이트)
대안기호는 입력과 출력에 모두 버블이 있음

iii) active logic level (활성 논리 값)

1, active-High : 버블X] 입,출력 따로 봄 값이 high 일때 활성
2, active-low : 버블O] '' low ''

활성과 High-low 헷갈리지 말기

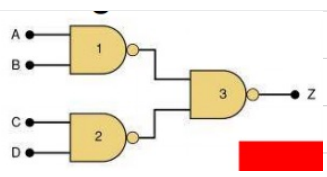
3. NAND Gate



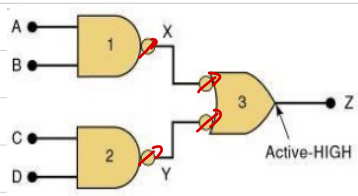
문제에서 출력의 상태가 Low 일때 게이가 작동한다하면 출력에 bubble이 있게 회로를 표현한다 (active Low)

10. Which gate representation to use 어떤 게이트 표현을 사용할 것인가?

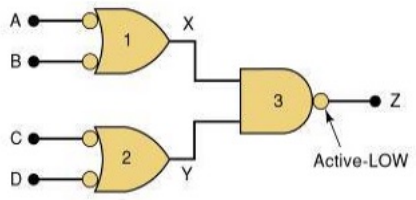
i)



proper use of alternate gate symbols in the circuit diagram can make circuit operation much clear.



Output Z will go high whenever either A=B=1 or C=D=1

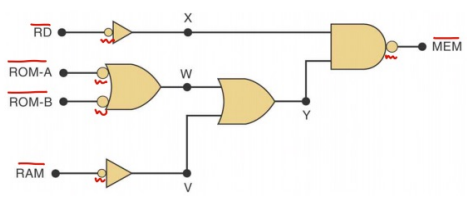


Output Z will go Low only when A or B is low and C or D is Low

ii) When possible, choose gate symbols ┌ bubble outputs - bubble input
└ nonbuble ouputs - nonbuble input

iii) Asseted levels (주장하는값) ! ┌ asserted = active
└ unasserted = inactive

iv) Labeling : overbar to activelow



4장 Combination Logic Circuits

1, Sum-of-products form

ⅰ, sum-of-products (SOP) : 곱의합, AND항이 서로 OR로 묶여 있다.

ex, $ABC + \bar{A}B\bar{C}$, $AB + \bar{A}B\bar{C} + \bar{C}\bar{D} + D$

역기호는 각 항의 하나의 변수 범위 넘으면 안됨 ($\overline{ABC}$, $\overline{RST}$ X)

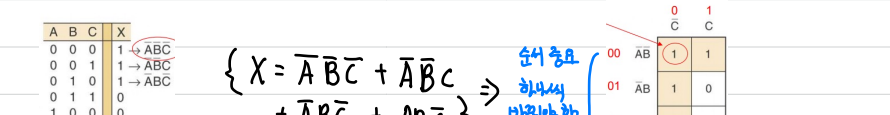
ⅱ, product of sums (POS) : 합의 곱, 2개 이상의 OR항이 AND로 묶여있다.

ex, $(A+\bar{B}+C)\cdot(A+C)$, $(A+\bar{B})(\bar{C}+D)F$

2, Simplify Logic Circuits

DeMorgan's theorems -> SOP -> 공통인수 찾아서 항 줄이기

by DeMorgan's theorems 역기호가 하나의 변수 범위를 넘지 않게 설정



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

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

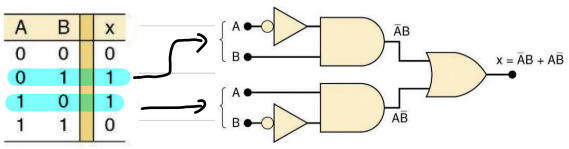
3, design Combinational Logic circuits

truth table (진리표) 작성 -> 출력이 1인것을 고르고 AND항으로 변환 -> SOP -> 간략화

A	B	x
0	0	0
0	1	1
1	0	1
1	1	0

$\bar{A}\cdot B$ ⇐ 0 1 1

$A\cdot\bar{B}$ ⇐ 1 0 1



A=1, $\bar{A}$=0 이라 생각

$X = \bar{A}\cdot B + A\cdot\bar{B}$ (SOP)

(1)

$\overline{X} = \bar{A}\cdot\bar{B} + A\cdot B$

(0)

4, Karnaugh Map (K map)

논리식을 간략화 하거나 진리표를 상응하는 논리회로로 변환하는데 사용

ⅰ, 2 variables

A	B	X
0	0	1 → $\bar{A}\bar{B}$
0	1	0
1	0	0
1	1	1 → AB

$\{X = \bar{A}\bar{B} + AB\}$ ⇒

	$\bar{B}$ (0)	B (1)
$\bar{A}$ (0)	1	0
A (1)	0	1

ⅱ, 3variables

A	B	C	X
0	0	0	1 → $\bar{A}\bar{B}\bar{C}$
0	0	1	1 → $\bar{A}\bar{B}C$
0	1	0	1 → $\bar{A}B\bar{C}$
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1 → $AB\bar{C}$
1	1	1	0

$\{X = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + AB\bar{C}\}$ ⇒

순서 중요 하나씩 바뀌어야 함

	$\bar{C}$ (00)	C (01)
$\bar{A}\bar{B}$ (00)	1	1
$\bar{A}B$ (01)	1	0
AB (11)	1	0
$A\bar{B}$ (10)	0	0

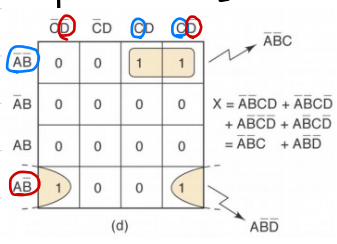
Horizontally adjacent Squares differ only in one variable.

Vertically adjacent Squares differ only in one variable.

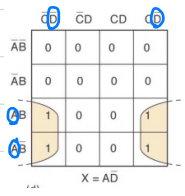
ⅲ, Looping : · 큰 그룹으로 묶었을 때 단지 공통변수만 최종식에 나타난다.

· 역형태와 역이 아닌형태를 모두 취하고 있는 변수를 제거

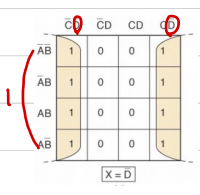
1, two pairs : 변수 하나를 소거할 수 있음



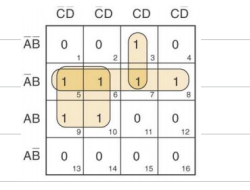
2, Quads : 2개의 변수 소거



3, Octets : 3개의 변수 소거

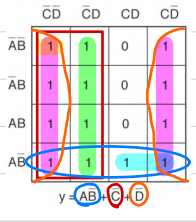


ⅳ, 1, truth table -> equation



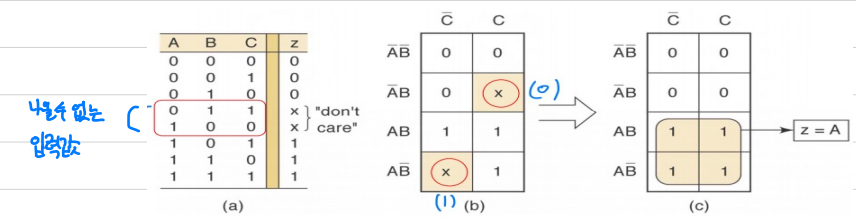
2, equation -> truth table

$y = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{C}\bar{D} + A\bar{B}\bar{C} + \bar{D}$



ⅴ, don't care condition

· 특정한 입력조건이 일어나지 않는 경우에는 이 입력조건하에 대해서 출력값 정의X

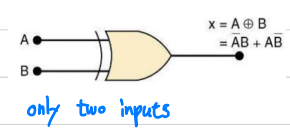


5. exclusive OR, NOR

ⅰ, XOR : high output whenever two inputs opposite level

A	B	x
0	0	0
0	1	1
1	0	1
1	1	0

$\bar{x} = \bar{A}B + A\bar{B}$ ⇒

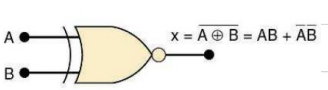


only two inputs

ⅱ, XNOR : high - at same level.

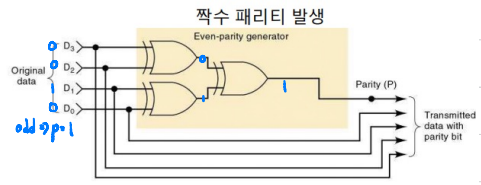
A	B	x
0	0	1
0	1	0
1	0	0
1	1	1

$x = AB + \bar{A}\bar{B}$ ⇒

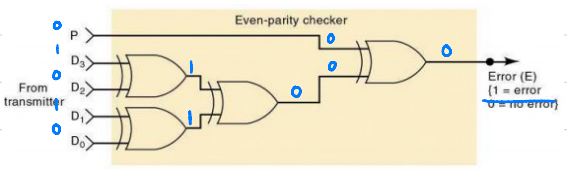


6. Parity generator and checker by using XOR

ⅰ,



ⅱ,



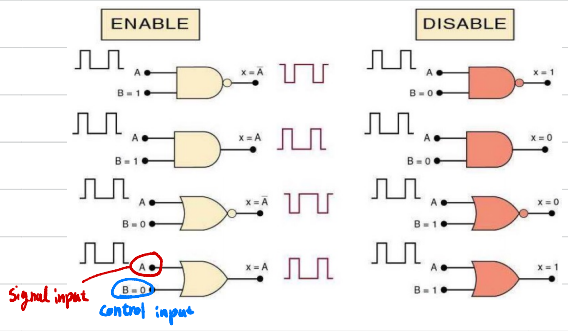
짝수 패리티 검사

7, Enable/Disable circuits

ⅰ, Enable : 출력이 입력의 형태를 보임

입력과 출력이 같다는 것은 아님

ⅱ, disable : 출력이 고정된 값으로 보임



8. Basic characteristics of digital ICs

ⅰ, Power and Ground : 1, V_{cc} : TTL, 5V

2, V_{dd} : CMOS, 3V ~ 18V

3, GND : Ground

ⅱ, floating input : 1, TTL : act like a logic 1, measure a dc level of between 1.4 ~ 1.8V (unconnected input) 2, CMOS : become overheated and destroy itself

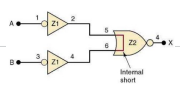
9. 내부 고장

ⅰ, malfunction (오동작) in the internal circuitry

ⅱ, Inputs or outputs shorted to ground or V_{cc}

ⅲ, " open-circuited

ⅳ, short between two pins.



10. 외부고장

ⅰ, 개방된 신호선

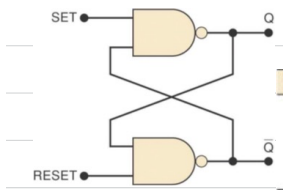
ⅱ, 단락된 신호선

ⅲ, 전원전압의 고장

ⅳ, 출력의 과부하

CH5. Flip-Flops

1. NAND Gate Latch : Active-Low 이어야 래치가 동작.

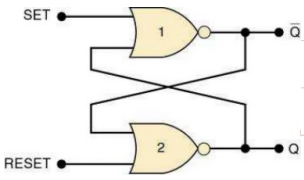


Set	Reset	Output
1	1	No change
0	1	Q = 1
1	0	Q = 0
0	0	Invalid*

- Output will change when the input is pulsed Low
- (set, reset) :
  - i, (1,1) : output depends on the previous input
  - ii, (0,1) : set 입력에 low를 인가하면, 항상 래치는 Q=1
  - iii, (1,0) : reset 〃 , Q=0

i, resting state ii, Setting latch iii, Clearing Latch

2. NOR Gate Latch : Active-high 이어야 동작



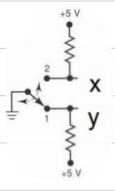
Set	Reset	Output
0	0	No change
1	0	Q = 1
0	1	Q = 0
1	1	Invalid*

- Output will change when the input is pulsed high
- (set, reset) :
  - i, (0,0) : output depends on the previous input
  - ii, : set 입력에 high를 인가하면, 항상 래치는 Q=1
  - iii, : reset 〃 , Q=0

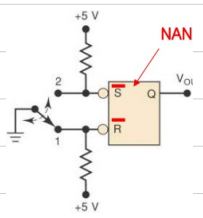
i, resting state ii, Setting latch iii, Clearing Latch

cf, bouncing : 스위치가 2에 완전히 안착하기 전까지 스위치의 바운스현상
(접점 2에서 여러번 붙었다 떨어지는 현상)

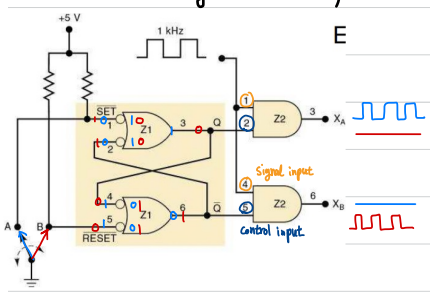
switching : switch on ⇒ '0', off ⇒ '1'
접지되어 땅으로 전류가 흐르면 자동 논 접촉이 안흐름



해결



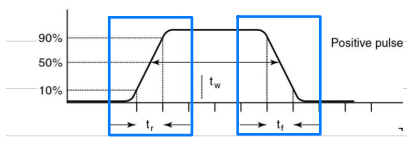
3. Troubleshooting Case Study



E

Switch position	X_A	X_B
A	pulses	LOW
B	LOW	pulses

4. Digital Pulses.



leading edge trailing edge

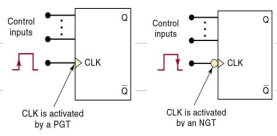
i, positive pulse : active high level, negative 〃 : active low level

ii, t_r : rise time, t_f : fall time, t_w : duration width

5. Clocked FFs

i, Synchronus system (동기) : ouput can change state only at a specific time in the clock cycle

- PGT : 0→1, NGT : 1→0
- FFs 는 Clock 신호의 엣지에서만 동작함
  제어입력은 transition(edge) 에서만 효과 있음

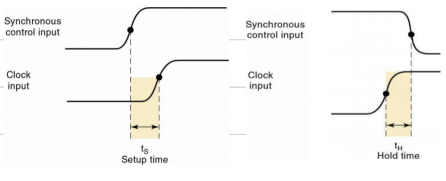


제어입력신호 변화는 일정한 시간을 두어야 한다.

ii, FF가 정확하게 동작하기 위해서는 클럭의 transition 이 일어났을 때 제어입력신호가 일정시간이상 유지되고 있어야한다.

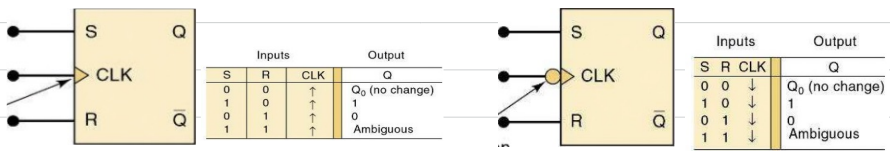
- Set up time (t_s) : prior to the clock transition
- hold up time (t_H) : after the clock transition

점이 transition을 표현



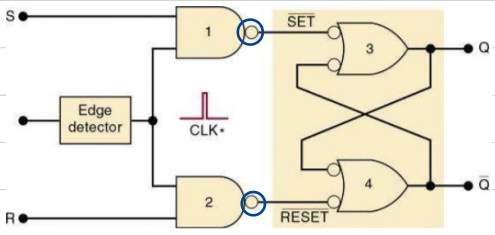
6. Clocked S-R FlipFlop

i, FF trigers on Positive / Negative transition



Inputs			Output
S	R	CLK	Q
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	Ambiguous

Inputs			Output
S	R	CLK	Q
0	0	↓	Q_0 (no change)
1	0	↓	1
0	1	↓	0
1	1	↓	Ambiguous



1,2 : 펄스 구동회로

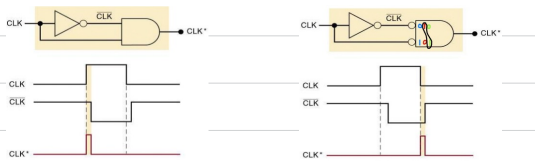
3,4 : basic NAND-gate

S=1, R=0, CLK^* ⇒ Q=1

S=0, R=1, CLK^* ⇒ Q=0

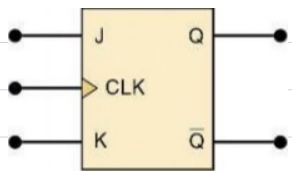
↑ NAND Gate Latch 에서와 입력에 버블표시가 더불어 동작은 NOR Gate Latch와 같다.

ii, Edge detector



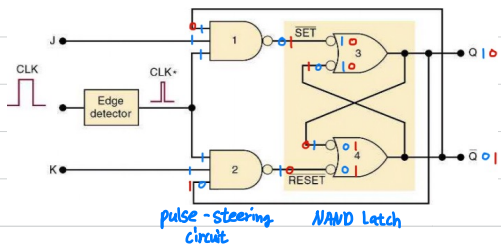
7. Clocked J-K Flip Flop

i, S-R FF 와의 차이 : J=k=1 ⇒ Q: toggle



J	K	CLK	Q
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	$\bar{Q}_0$ (toggles)

ii, internal circuitry of an edge triggered J-K flip-flop

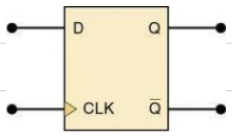


J=k=1, Q=0, $\bar{Q}$=1 ⇒ Q=1 (initial condition)

J=k=1, Q=1, $\bar{Q}$=0 ⇒ Q=0

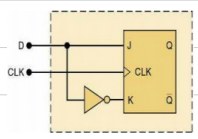
8. Clocked D Flip-Flop

i, CLK가 발생했을때 D의 입력상태와 같은값을 출력함



D	CLK	Q
0	↑	0
1	↑	1

ii, internal circuitry of an edge triggered D flip-flop

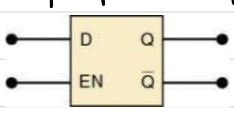


출력 Q는 정확히 정의된 시간에만 입력 D의 같은값을 갖음

⇒ Parallel Data Transfer.

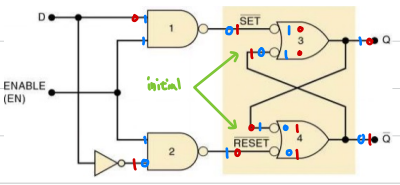
9. D Latch

i, D flip-flop 과 차이 : edge detector 사용 X ⇒ clk에 의존X



Inputs		Output
EN	D	Q
0	X	Q_0 (no change)
1	0	0
1	1	1

ii, internal circuitry of an edge triggered D Latch

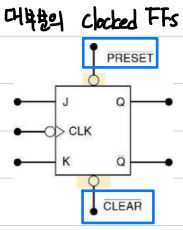


clock input → Enable input

10. Asynchronous inputs

i, S,R,J,k,D input : clk에 의존

대부분의 clocked FFs 는 비동기식 입력단이 존재 : $\overline{PRESET}$, $\overline{CLEAR}$: Active-low



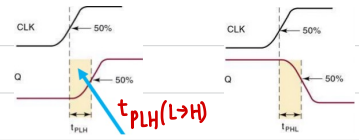
J	K	Clk	$\overline{PRE}$	$\overline{CLR}$	Q
0	0	↓	1	1	Q (no change)
0	1	↓	1	1	0 (Synch reset)
1	0	↓	1	1	1 (Synch set)
1	1	↓	1	1	$\bar{Q}$ (Synch toggle)
x	x	x	1	1	Q (no change)
x	x	x	1	0	0 (asynch clear)
x	x	x	0	1	1 (asynch preset)
x	x	x	0	0	(Invalid)

$\overline{CLEAR}$ 이 〃 , 〃 0

$\overline{PRESET}$ 이 활성화 되고, Q는 CLK와 무관하게 1

11. FF timing consideration - parameters

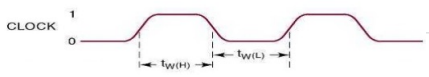
i, propagation delays (전파지연) : 신호가 인가되고 출력이 변화할때까지 걸리는 시간



t_{PLH} (L→H)

t_{PHL} (H→L)

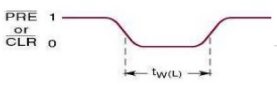
ii, Clock pulse high and low times



t_w(H) : low로 되돌아 가기전 high로 유지하는 최소시간

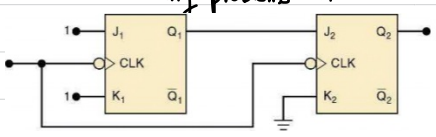
t_w(L) : high 〃 low 〃

iii, Asynchronus Active Pulse Width



t_w(L) : PRESET, CLEAR 입력이 활성상태로 유지해야 되는 최소시간

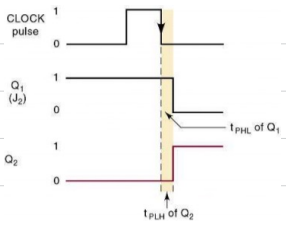
12. Potential timing problems in FF



- 클럭의 NGT에서 Q_1이 변하고, 따라서 Q_2의 입력 J_2가 같은 NGT 동안에 변하므로 타이밍 문제가 일어날수 있음

if, t_{PHL} of Q_1 > t_H of Q_2 ⇒ 문제 X

실제 대부분 t_H=0

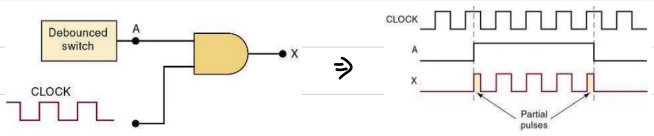


FF의 출력은 활성클럭 천이가 나타나기 직전에 동기 제어 입력에 인가되어 있는 논리 신호에 의해 결정

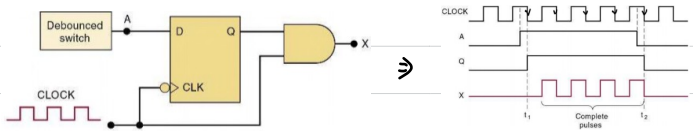
14. 동기화

1) 동기식(CLK) + 비동기식(CLK와 무관하게 사용자 입력 스위치를 조작)
   → 불규칙성으로 인한 문제 → 동기화 시켜야!

문제: 불규칙 pulse 발생

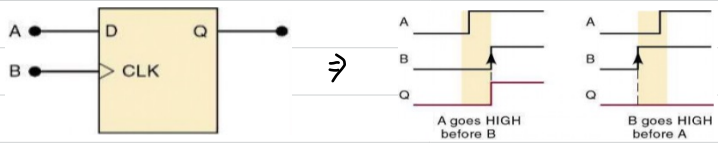


해결



한계: A의 불규칙적 high로인해 t_p(셋업딜레이) 어길수 있음

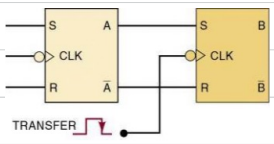
15. 입력 순차 검출 : 입력이 순차적으로 활성화 되었을때 출력 동작



17. Data storage and transfer

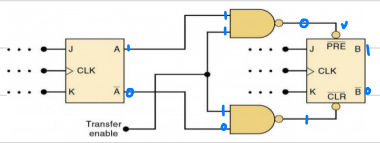
- 데이터를 FF (or Registers)에 저장하기 위해 수행하는 동작 : data transfer (데이터 전송)

i) 동기 전송



- (S,R) or (J,K) or D
- 동기식 신호와 CLK신호가 전송을 수행하는 데 이용
- NGT 이후 B의 출력은 A와 같아짐

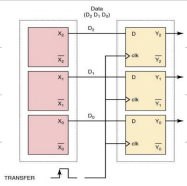
ii) 비동기 전송



- Transfer enable = 0 → PRE=1 (inactive), CLR=1 (inactive)
  no effect on FF outputs
- Transfer Enable = 1 and A=1 → PRE=0 (active) → B=1
- Transfer Enable = 1 and A=0 → CLR=0 (active) → B=0

T.E : Low → NAND출력 : '1' → 아무런 영향X but, high → NAND : A에 따라 Low → A=B

iii) 병렬 전송 (Simultaneously) fast but many lines

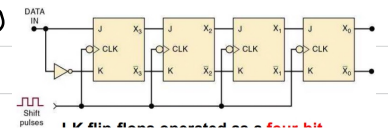


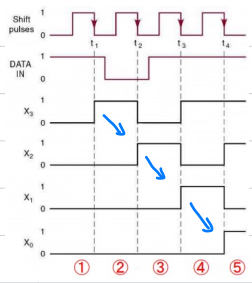
after transfer the contents of X are maintained

iv) 직렬 전송 - shift register slow but economic and simple

FF에 저장된 2진수를 매 클럭마다 하나의 FF에서 다른 FF로 이동 (as shift pulses are applied)

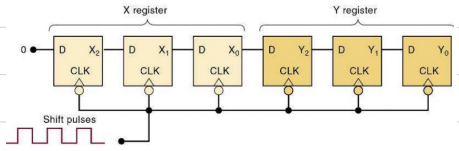
①





interval	X3	X2	X1	X0
①	0	0	0	0
②	1	0	0	0
③	0	1	0	0
④	1	0	1	0
⑤	1	1	0	1

② two connected

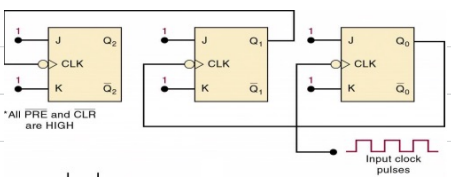


X_2	X_1	X_0	Y_2	Y_1	Y_0	
1	0	1	0	0	0	- before pulse applied
0	1	0	1	0	0	- first pulse
0	0	1	0	1	0	- second pulse
0	0	0	1	0	1	- third pulse

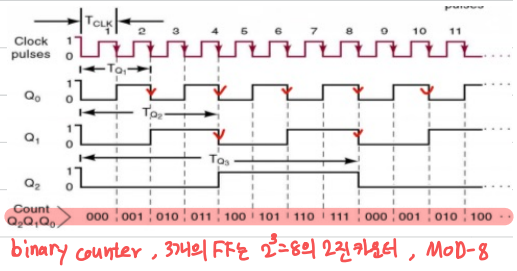
After transfer, the original data are lost

19. Frequency division and counting

n개의 FF를 사용하면 출력주파수는 입력주파수의 $1/2^n$로 분할



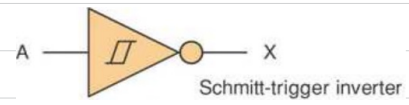
NGT 신경쓰기 , (1,1)일경우 toggle

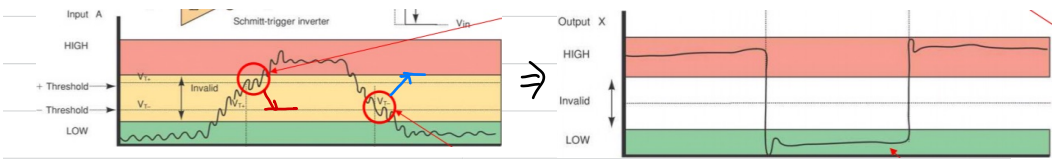


binary counter , 3개의 FF로 2^3=8의 2진카운터 , MOD-8

22. Schmitt-Trigger device

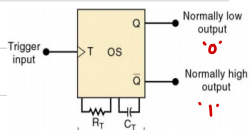
입력 전압이 $V_T = V_{T+}$를 넘으면 V_T는 V_{T-}로 바뀜
" $V_T = V_{T-}$를 넘으면(작아지면) V_T는 V_{T+}로 바뀜
⇒ 느리게 변하는 입력신호이 발생하지 않고 신뢰성있게 동작



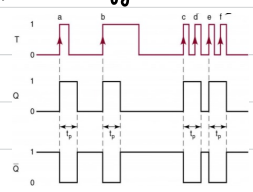


23. One-shot

i) 하나의 안정된 출력 (보통 Q=0, $\bar{Q}$=1) 입력 신호에 의해 트리거 되기 까지 그 상태 유지
트리거되면 OS 출력은 반대 상태 (Q=1, $\bar{Q}$=0, 준안정상태)로 t_p동안 변한다
$t_p = RC$ (외부에 연결된 소자)

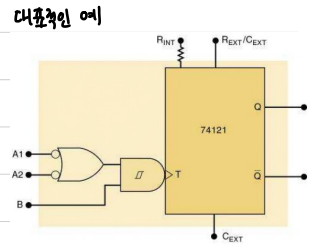


ii) NON retriggerable

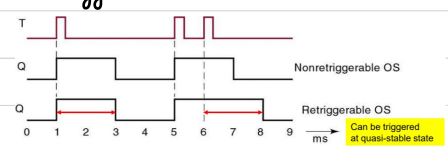


transition at d and f 효과 못줌, ∵ 이미 high

대표적인 예



iii) retriggerable



begin new t_p interval each time it receives trigger pulse

CH6. Digital Arithmetic (디지털 산술)

1. Binary Addition & Subtraction

both performed just like the decimal

2. Signed Numbers

i, extra sign bit of 0 : positive
" 1 : negative

ii, 2's complement system : 부호를 갖는 2진수를 표현하는 체계 (Negation : 부정)

주어진 binary bit에 1의 보수를 (0→1, 1→0) 취하고 LSB에 1을 더한다

ex, −9를 2진수로 표현 : 9_{10} : 01001_2 → 10110 ⇒ 10111 : -9_{10}

부호비트를 이용하여 5비트사용

decimal 이 주어진경우 (음수) → binary	:	양수의 binary (sign bit 포함) ⇒ 2의 보수
binary bit (MSB=1) → decimal	:	부호는 음수이고 크기는 2의 보수 취해서 구함
binary bit (MSB≠1), 2의보수 → 결과.	:	원래 binary decimal 값의 부호만 바뀜

iii, 부호확장

레지스터의 크기는 각 수를 저장할 2진 자릿수 결정

양수 → 0을 덧붙임, 음수 → 1을 덧붙임

iii, special case

$1000 = -2^3 = -8$
$10000 = -2^4 = -16$
$100000 = -2^5 = -32$

} 2의 보수를 하면 자기자신으로 (∵ 그 비트로 나타낼 수 있는 음수한계값)

N개의 크기 비트가 2의 보수체계에서 표현할 수 있는 전체 값의 범위

$(-2^N \sim 2^N-1)$

3. Addition in 2's complement system

※ 각 수의 부호비트가 크기 비트와 같은 방법으로 연산, 부호비트에서 자리올림은 무시

i, 두 양수

```
+9 → 0 1001  (augend, 피가산수)
+4 → 0 0100  (addend, 가수)
     0 1101  (sum=+13)
     └→ sign bits
```

ii, 양수와 크기가 작은 음수

```
 +9 → 0 1001  (augend)
 −4 → 1 1100  (addend)
 (1)  0 0101  (sum=+5)
```

자리올림 무시 ┘ ↑ sign bit

iii, 양수와 크기가 큰 음수

```
      sign bits
−9 → 1 0111  (augend)
+4 → 0 0100  (addend)
     1 1011  (sum=−5)
```

iv, 두 음수

```
        sign bits
 −9 → 1 0111  (augend)
 −4 → 1 1100  (addend)
 (1)  1 0011  (sum=−13)
```

v, 크기가 같고 부호 반대

```
        sign bits
 −9 → 1 0111  (augend)
 +9 → 0 1001  (addend)
 (1)  0 0000  (sum=0)
```

4. Subtraction in 2's complement system

i, how to : 감수에 부정을 취한다. → 이 것을 피감수에 더한다.

(+9)−(+4) ⇒ 피감수 (+9) : 01001
감수 (+4) : 00100 →(부정) (−4) : 11100

⇒ 01001 (+9) − 00100 (+4) ; 01001 (+9) + 11100 (−4) = 무시 1 00101 (+5)

(−4)−(+9) ⇒ 피감수 (−4) : 11100
감수 (+9) : 01001 → (−9) : 10111

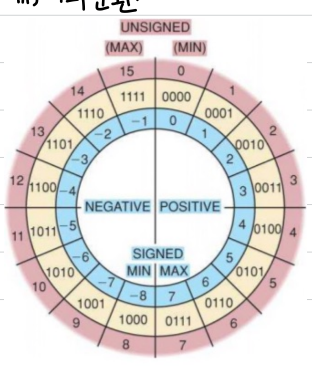
⇒ 11100 (−4) − 01001 (+9) = 11100 (−4) + 10111 (−9) = 무시 1 10011 (−13)

ii, overflow : 결과가 주어진 크기비트로 표현 못할때 오류가 발생함 ⇒ 넘침조건을 검출하는 회로존재

Addition : 같은 부호의 수
Subtraction : 다른 부호의 수

ex,
```
+9 → 0 1001
+8 → 0 1000
     1 0001
```
incorrect sign / incorrect magnitude

iii, 수의 순환



overflow

(1) 덧셈 : 피가수 값에서 시작하여 가수값만큼 시계방향으로 진행

넘침은 4비트 부호 있는 수 체계 한계를 넘어설때 : 7초과

(2) 뺄셈 : 가수 값의 크기만큼 반시계 진행

or 3+(−5) ; (−5)는 부호 없는 2진수로 인식되어 11을 나타냄. 3에서 시작

11만큼 시계방향 ⇒ 1110 (−2)

넘침은 서로 다른 부호를 가진 4비트 수 사이의 뺄셈 연산 결과가 7보다 크거나 −8보다 작으면 발생

3−(−6) : 3에서 6칸 시계방향으로 이동하면 부호있는 수 −7에 도달 : 틀린답

5. Multiplication

i, unsigned : similar to decimal

```
      1001   ← multiplicand = 9₁₀
      1011   ← multiplier = 11₁₀
      1001 ┐
     1001  │ partial products
    0000   │
   1001    ┘
   1100011   final product = 99₁₀
```

ii, signed : case1, two positive : same as unsigned
case2, two negative : 둘다 2의 보수 → 곱
case3, positive, negative : negative 2의 보수취함 → 곱 → 결과를 다시 2의 보수로

6. Division

i, unsigned : similar to decimal
ii, signed : Same as Multiplication

```
      0010.1
100 ) 1010.0
      100
       100
       100
         0
```

7. BCD Addition (뺄셈은 어려워서 고려X)

i, 합이 9 이하

```
  0101 ← BCD for 5
+ 0100 ← BCD for 4
  1001 ← BCD for 9
```

ii, 합이 9초과 : 6(0110)을 더해서 보정

```
 59           1
+38  ➡   0101 │ 1001 ← BCD for 59
+97    + 0011 │ 1000 ← BCD for 38
         1001 └ 0001 ← perform addition
                0110 ← add 6 to correct
        (1001) (0111) ← BCD for 97
          9      7
```

8. Hexadecimal Arithmetic

i, Addition : 가장 큰 16진 숫자는 9 대신 F라는 것만 알면 10진 덧셈과 똑같다.

(1) hex를 decimal 방식으로 더하고 9보다 큰 숫자는 decimal 등가삽입
(2) 합이 15 미만 → hex 로 표현
(3) 합이 16 이상 → 16을 빼고, 다음의 자리에 1 올림

```
carry → 1
        5 8
      + 4 B
        A 3
```

$8 + B = 19_{10} \geq 16$ → carry
19−16 = 3

ii, Subtraction : binary 와 동일한 방법 [16진 감수의 2의 보수를 취하고 피감수에 더한후 MSD에서의 자리올림은 무시]

hex 에서 2의 보수

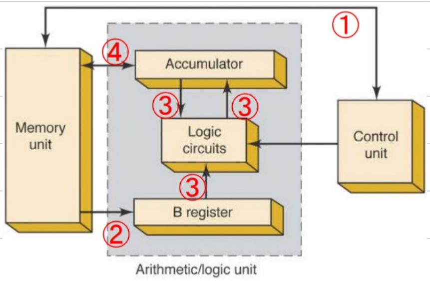
(1) binary 변환 → 2의보수 → hex 변환

```
      73A            ← hex number
0111 0011 1010       ← convert to binary
1000 1100 0110       ← take 2's complement
      8C6            ← convert back to hex
```

(2) Quicker procedure

```
 F   F   F
−7  −3  −A   } ← subtract each digit from F
 8   C   5
        +1     ← add 1
 8   C   6     ← hex equivalent of 2's complement
```

9. Arithmetic Circuits



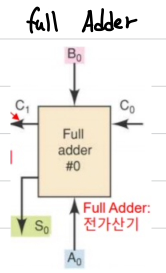
Accumulator, B register : FF

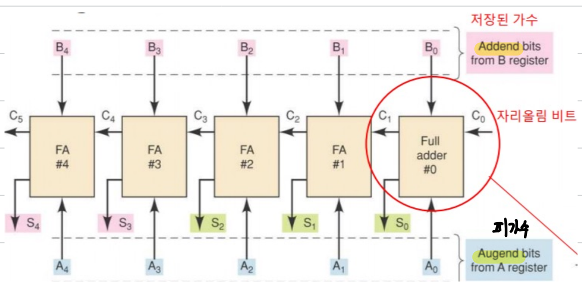
i, 제어장치는 특정 주소에 저장된 수가 누산기 레지스터에 현재 저장된 수에 더해져야 한다는 명령을 받음
ii, 더해지는 수는 메모리 → B register
iii, Logic circuits 에서 B register의 수와 Accumulator의 수와 더해지고 합은 Accumulator 에 저장을 위해 전송
iv, 새로운 수는 Accumulator에 남아있거나 저장을 위해 메모리로 전송

10. Parallel Binary Adder

full Adder

3 input
2 output

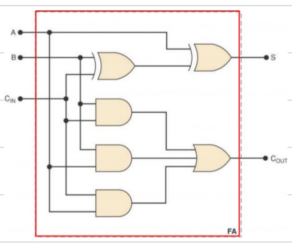




simultaneous
Full adder 한개당 1비트

11. Design of full Adder

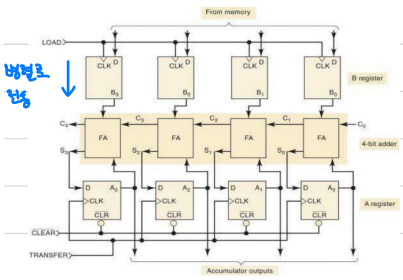
Augend bit input	Addend bit input	Carry bit input	Sum bit output	Carry bit output
A	B	C_{IN}	S	C_{OUT}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

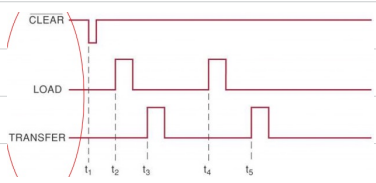


$$S = \bar{A}\bar{B}C_{IN} + \bar{A}B\bar{C}_{IN} + A\bar{B}\bar{C}_{IN} + ABC_{IN}$$
$$= \bar{A}(\bar{B}C_{IN} + B\bar{C}_{IN}) + A(\bar{B}\bar{C}_{IN} + BC_{IN})$$
$$= \bar{A}(B \oplus C_{IN}) + A(\overline{B \oplus C_{IN}})$$
$$= A \oplus (B \oplus C_{IN})$$

$$C_{OUT} = \bar{A}BC_{IN} + A\bar{B}C_{IN} + AB\bar{C}_{IN} + ABC_{IN} + ABC_{IN} + ABC_{IN}$$
$$= BC_{IN}(\bar{A} + A) + AC_{IN}(B + \bar{B})$$
$$+ AB(C_{IN} + \bar{C}_{IN})$$
$$= BC_{IN} + AC_{IN} + AB$$

12. Complete Parallel Adder with Registers





Sequence of operation for $1001_2 + 0101_2$

i, [A] = 0000 t_1 : $\overline{\text{CLEAR}}$ pulses is applied to $\overline{\text{CLR}}$

ii, [M] → [B] t_2 : 1001_2 는 Load pulse PGT에 register B로 적재

iii, [S] → [A] t_3 : FA는 1001를 출력 , Transfer pulse PGT에 A로전송

iv, [M] → [B] t_4 : 0101_2 는 두번째 Load PGT에 B로 적재

v, [S] → [A] t_5 : FA는 [B]=0101 과 [A]=1001 에서 [S]=1100 발생
두번째 Transfer PGT에 A로 전송

vi, [A] → [M] [A]는 메모리로 전송

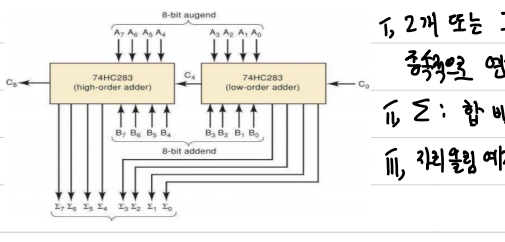
cf, [A] : 레지스터 A의 내용

13. Carry Propagation

MSB에서 발생한 합 비트가 LSB의 합에 의하여 발생한 자리올림에 의존

⇒ 자리올림예견으로 해결 : 높은 자리의 자리올림이 발생되는데 알아보기 위하여 저가속와 가속의 낮은 자리의 비트를 관찰
(look ahead carry)

14. Integrated Circuit Parallel Adder



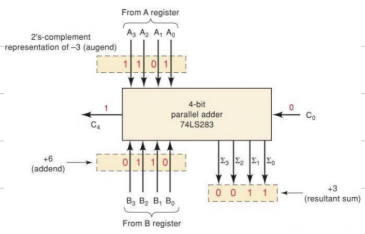
i, 2개 또는 그 이상의 IC 가산기는 더 큰 2진수의 덧셈을 허용하여 종속적으로 연결

ii, Σ : 합 비트에 대한 S 표기의 가장 보편적인 다른 표현

iii, 자리올림 예견가능은 C_4가 1일 때 생성된다.

15, 2's complement System.

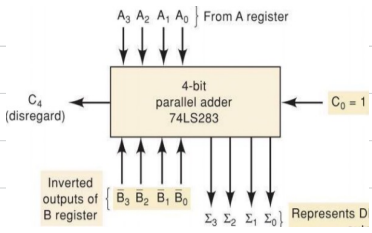
i, Addition



음수는 부호비트를 포함하는 2의 보수 형태로 표현하고 더함

C_4=1 은 2의 보수 방법에서는 무시

ii, Subtraction



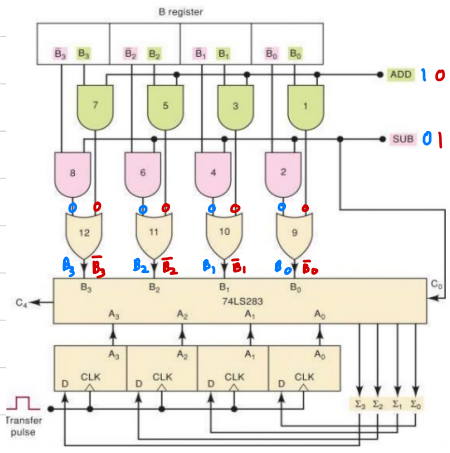
감수를 2의 보수로 바꾸어 피감수에 더한다.

$A + (\overline{B} + 1) = A - B$

iii, Combination

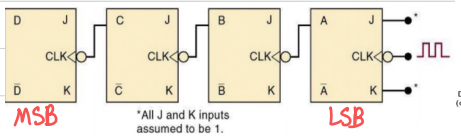
controlled by the two control signals ADD and SUB

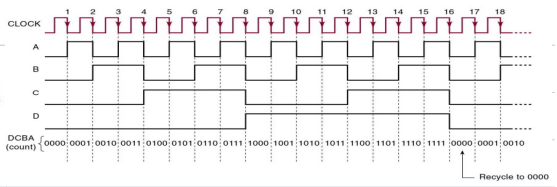
ADD - High : A+B , SUB - High : A-B



7. Counter & Resisters

1, Asynchronous (Ripple) Counters





i, FF(A)만 CLK에 동기되어 동작함 나머지는 CLK 동기X

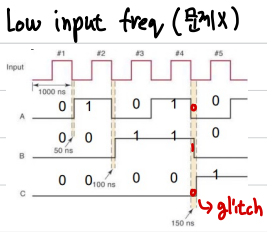
ii, 모든 J,K는 1로 동일

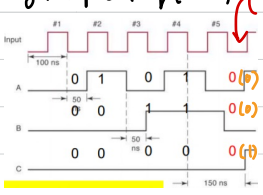
iii, MOD number: 카운터가 처음상태로 되돌아갈 때까지 한 주기를 이루는 상태의 개수, 2^N (N: FF의 수)

iv, 주파수 분할: MSB의 출력주파수는 입력 클럭 주파수의 MOD수로 나눈 값과 같다.

2, Propagation Delay (전파지연)

i, Low input freq (문제X) ii, high input freq (문제O) ← can't count properly

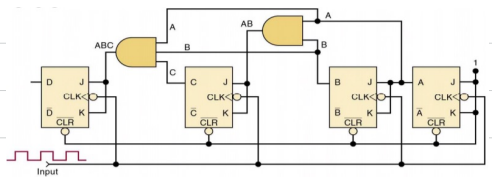




proper operation: $T_{clock} \geq N \times t_{pd}$

(사용가능한) Max freq: $1/N \times t_{pd}$

3. Sync Parallel counters (전파 지연 문제 개선)

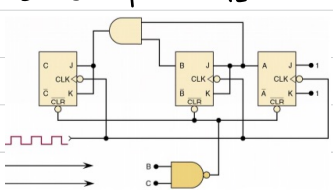


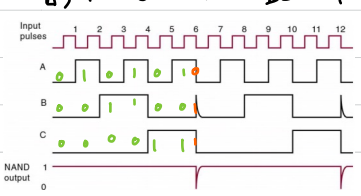
Count	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
0	0	0	0	0

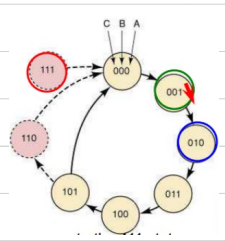
- A만 J,K 입력이 high 상태유지
- 하위단위의 출력이 모두 high 일때만 그 FF의 J,K입력이 High 상태가 되도록 연결.
- 총지연시간 = FF_{td} + AND 게이트 t_{pd}

4. MOD number $< 2^N$

기본 카운터에서 상태 일부를 무시하고 건너 뜀, 구현하는 모드수 보다 높은 모드수 회로에서 수정해야 한다.







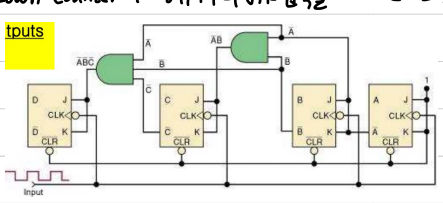
i, NAND - High ⇒ 영향 X, but NAND - Low ⇒ FF clear, go to 000

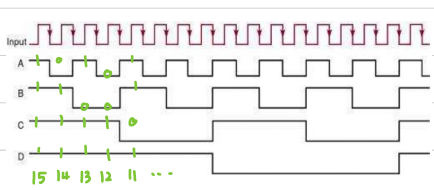
ii, B=1 이 될때 NAND-Low ⇒ 카운터 000 초기화

iii, 110 과 같은 순간적인 상태: 과도(transient) 상태, glitch 출력주파수는 f/MOD

5, Synchronous Down and Up/Down Counters

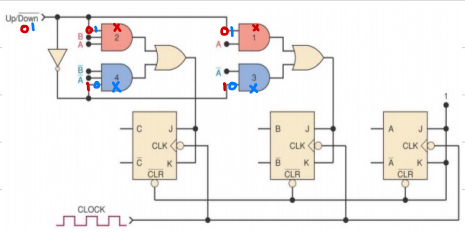
i, down Counter : 상위 FF의 J,K입력을 반전된 출력을 사용

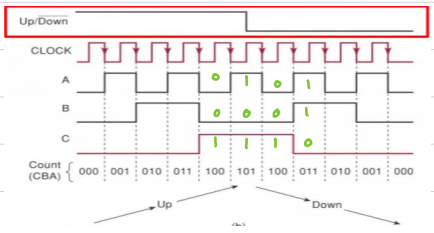




초기 ABCD: 1111_2, A=0($\bar{A}$=1)일때 NOT게 B변함, A=B=0($\bar{A}\cdot\bar{B}$=1)일때 NOT게 C변함

ii, UP/Down counter: FF의 J와K입력에 정상적인 출력 또는 반전된 출력중 어느 것이 입력될지 선택



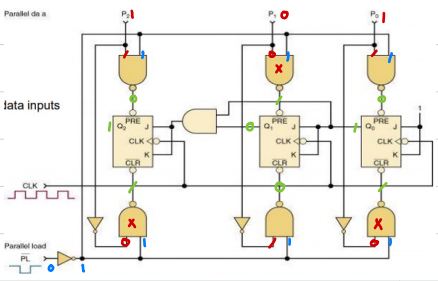


(1) UP/$\overline{Down}$ - High : A,B의 출력 → B,C의 입력

(2) - Low : $\bar{A}$,$\bar{B}$의 출력 → B,C의 입력

6, Presettable Counter

클럭신호에 비동기 또는 동기되어 원하는 초기계수값을 미리 정함



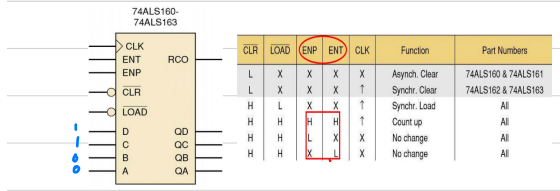
i, J,K,CLK 는 병렬 업카운트 회로, (PRESET, CLEAR) 입력은 비동기적으로 프리셋되도록 연결

ii, $\overline{PL}$이 Low 인가 ⇒ $(P_2P_1P_0)=(Q_2Q_1Q_0)$ 비동기적 (∵ 비동기 입력 중 하나를 가짐)

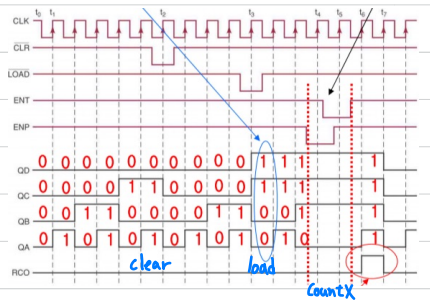
iii, $\overline{PL}$ - High ⇒ CLK에 따라 작동

7. IC Synchronous Counter

i, 74AL160 - 74ALS163 계열



CLR	LOAD	ENP	ENT	CLK	Function	Part Numbers
L	X	X	X	X	Asynch. Clear	74ALS160 & 74ALS161
L	X	X	X	↑	Synchr. Clear	74ALS162 & 74ALS163
H	L	X	X	↑	Synchr. Load	All
H	H	H	H	↑	Count up	All
H	H	L	X	X	No change	All
H	H	X	L	X	No change	All



우선순위

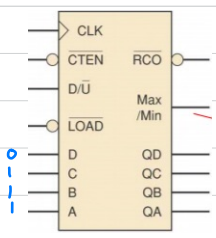
(1) $\overline{CLR}$ - Low : 카운터출력 0000으로 리셋 (동기식, 비동기식)

(2) $\overline{LOAD}$ - Low with $\overline{CLR}$ - High : 원하는 값 프리셋

(3) ENP, ENT - High with $\overline{CLR}$, $\overline{LOAD}$ - high : count 시작

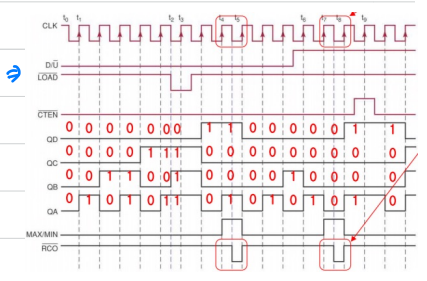
(4) RCO : 카운터의 마지막 가장 큰 값, 즉 터미널 상태를 검출하는 기능

ii, 74ALS190-191 / 74HC190-191 → UP/Down Counter



74HC190 : 제작 10 ⇒

LOAD	CTEN	D/U	CLK	Function
L	X	X	X	Asynch. Load
H	L	L	↑	Count up
H	L	H	↑	Count down
H	H	X	X	No change



(1) $\overline{LOAD}$ - Low : DCBA로 프리셋

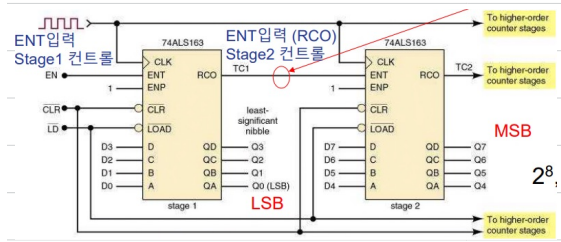
(2) CTEN - Low with $\overline{LOAD}$ - High : 카운터 동작, D/U로 카운트 방향 제어

(3) Max/Min : 카운터의 터미널 상태 검출 Down counter: (MIN) 0000, up counter: (MAX) 1001 or 1111

decimal counter MOD-16

iii, 다단계 연결

여러 개의 칩을 연결하여 더 큰 계수 범위를 갖는 회로 만듦



(1) $\overline{CLR}$ 로 동기식으로 클리어, $\overline{LD}$ 로 동기식으로 프리셋

(2) EN - 1단계 ENT, ENT와 RCO를 사용 → 다단연결 용이

(3) Q3,Q2,Q1,Q0 : 1111, EN - High → TC1 - High

→ 다음 PGT에 두카운터 모두 업카운트 (1단계→0000, 2단계→전단계에서 하나 증가) → TC1 - Low

→ EN=1 이면 1단계는 1111이될때에 증가 → 반복

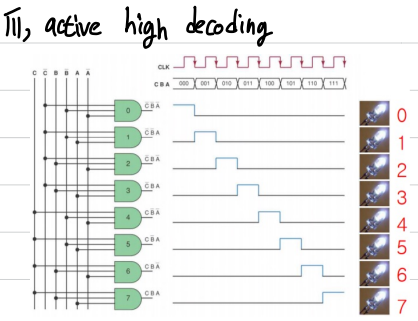
8. Decoding a Counter (display of counter state immediately recognized by human)

i, binary data : encoded data, not comfortable to human

decoded data : comfortable to human ex, decimal number

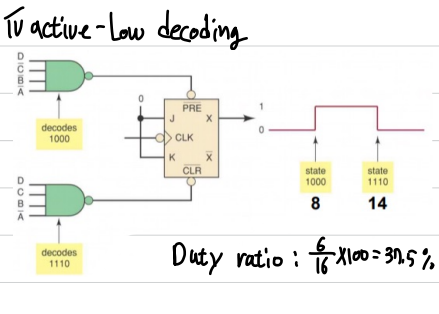
ii, electronic decoding: 많은 응용분야에 사람이 개입치 않고 자동으로 동작의 타이밍 또는 순서를 제어하는데 카운터많이 쓰임

iii, active high decoding



각 AND게이트는 카운터의 하나의 특정한 상태에 대해서 High 출력 발생

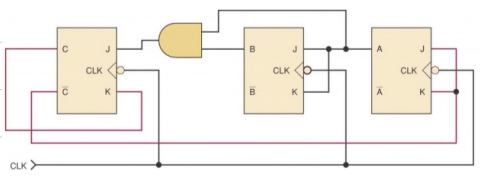
iv active-Low decoding



Duty ratio : $\frac{6}{16} \times 100 = 37.5\%$

9. Analyzing Synchronous Counters

i,



카운트 순차를 발생하려면 개별FF에 동기식 입력을 적절히 입력

step1, 입력값 작성 $\begin{cases} J_C = A\cdot B,\ K_C = C \\ J_B = K_B = A \\ J_A = K_A = \bar{C} \end{cases}$

PRESENT State C B A	Control Inputs J_C	K_C	J_B	K_B	J_A	K_A	NEXT State C B A
0 0 0	0	0	0	0	1	1	0 0 1
0 0 1	0	0	1	1	1	1	0 1 0
0 1 0	0	0	0	0	1	1	0 1 1
0 1 1	1	0	1	1	1	1	1 0 0
1 0 0	0	1	0	0	0	0	0 0 0
1 0 1	0	1	1	1	0	0	0 1 1
1 1 0	0	1	0	0	0	0	0 1 0
1 1 1	1	1	1	1	0	0	0 0 1

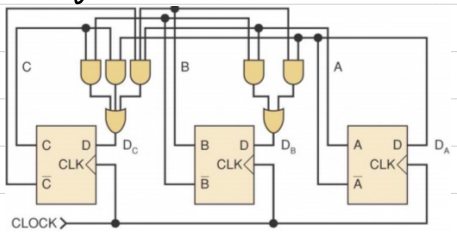
unused states

Step2, 카운터의 현재상태를 가정하고 카운터의 다음 상태 예측

"MOD-5 counter"

미사용 상태가 항상 정상 계수 순차로 돌아감

ii, using D-FF



PRESENT State C B A	Control Inputs D_C	D_B	D_A	NEXT State C B A
0 0 0	0	0	1	0 0 1
0 0 1	0	1	0	0 1 0
0 1 0	0	1	1	0 1 1
0 1 1	1	0	0	1 0 0
1 0 0	1	0	1	1 0 1
1 0 1	1	1	0	1 1 0
1 1 0	1	1	1	1 1 1
1 1 1	0	0	0	0 0 0

"MOD-8 binary counter"

$$D_C = C\bar{B} + C\bar{A} + \bar{C}BA = C(\bar{B}+\bar{A}) + \bar{C}BA = C\overline{BA} + \bar{C}BA = C \oplus (AB)$$

$$D_B = \bar{B}A + B\bar{A} = B \oplus A$$

$$D_A = \bar{A}$$

⇒ • MOD-16 binary counter $D_D = D \oplus (ABC)$, $D_C = C \oplus (AB)$, $D_B = B \oplus A$, $D_A = \bar{A}$

10. Sync Counter Design

※ 카운터의 여러 개의 상태를 디코딩하여 J.K 입력단에 적절한 논리값을 제공하는 논리회로를 설계

cf, J-K excitation table

Transition at FF Output	PRESENT State Q_n	NEXT State Q_{n+1}	Which case?	J	K
0→0	0	0	Clearing or no change	0	x
0→1	0	1	Setting or toggle	1	x
1→0	1	0	Clearing or toggle	x	1
1→1	1	1	Setting or no change	x	0

J	K	CLK	Q
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	$\overline{Q}_0$ (toggles)

Design Procedure

I, 비트수를 결정하고 계수 순서를 정한다.

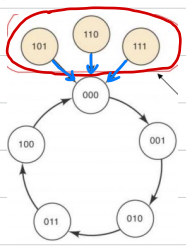
C	B	A
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
0	0	0
0	0	1

etc.

unused state

101, 110, 111

II, 모든 가능한 상태의 천이도를 그림



III, 현재-다음상태 표 작성

	PRESENT State C	B	A	NEXT State C	B	A
Line 1	0	0	0	0	0	1
2	0	0	1	0	1	0
3	0	1	0	0	1	1
4	0	1	1	1	0	0
5	1	0	0	0	0	0
6	1	0	1	0	0	0
7	1	1	0	0	0	0
8	1	1	1	0	0	0

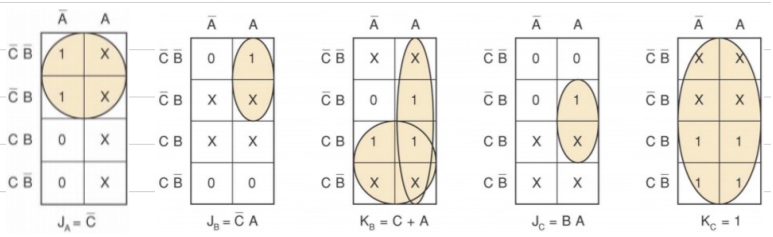
IV, J.K 입력 추가 (based on J-K excitation table)

	PRESENT State C	B	A	NEXT State C	B	A	J_C	K_C	J_B	K_B	J_A	K_A
Line 1	0	0	0	0	0	1	0	x	0	x	1	x
2	0	0	1	0	1	0	0	x	1	x	x	1
3	0	1	0	0	1	1	0	x	x	0	1	x
4	0	1	1	1	0	0	1	x	x	1	x	1
5	1	0	0	0	0	0	x	1	0	x	0	x
6	1	0	1	0	0	0	x	1	0	x	x	1
7	1	1	0	0	0	0	x	1	x	1	0	x
8	1	1	1	0	0	0	x	1	x	1	x	1

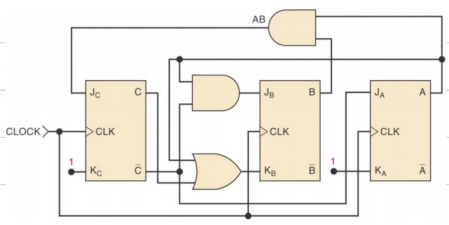
V, 각 J,K 입력에 필요한 원하는 값을 출력하는 논리회로 설계

PRESENT C	B	A	J_A
0	0	0	1
0	0	1	x
0	1	0	1
0	1	1	x
1	0	0	0
1	0	1	x
1	1	0	0
1	1	1	x

⇒

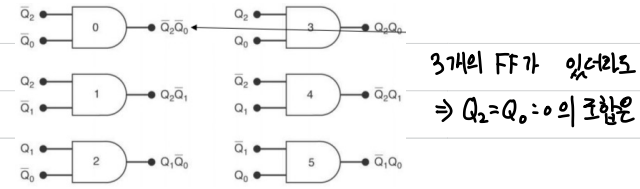


VI, 최종 표현식 구함



동기식 카운터에서 모든 FF는 동시에 클럭 입력

II. Shift Register Counter (Ring counter)



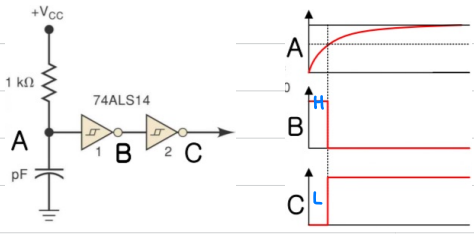
Q_3	Q_2	Q_1	Q_0	CLOCK pulse
1	0	0	0	0
0	1	0	0	1
0	0	1	0	2
0	0	0	1	3
1	0	0	0	4
0	1	0	0	5
0	0	1	0	6
0	0	0	1	7

"MOD-4 counter"

I, 2진 카운터 순서와 같이 진행하지는 않지만 각계속과 특정출력이 상응하기 때문에 카운터라고 본다.

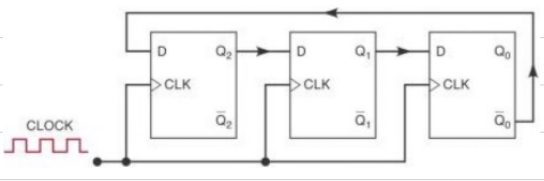
II, MOD-8 ring counter → 8FFs
" binary counter → 3FFs

III, ring counter 초기화 : operate properly → 1개의 FF 만 '1', 나머지 '0'

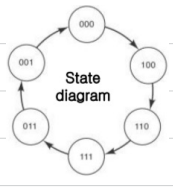


전원이 인가된 후 충분한 시간동안
Q_3의 $\overline{PRE}$ 입력과 Q_2, Q_1, Q_0의 $\overline{CLR}$ 입력에 Low 인가.

IV, Johnson counter (twisted counter)



Q_2	Q_1	Q_0	CLOCK pulse
0	0	0	0
1	0	0	1
1	1	0	2
1	1	1	3
0	1	1	4
0	0	1	5
0	0	0	6
1	0	0	7
1	1	0	8



"MOD-6 counter"

MOD 수 = FF x 2

decoding

3개의 FF가 있어도 디코딩 게이트에는 2개의 입력 필요
⇒ $Q_2=Q_0=0$의 조합은 0일때 한번 발생됨

Summary

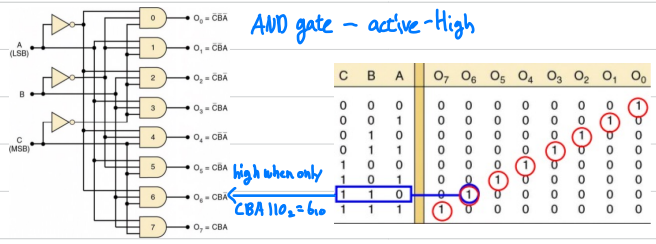
2의 보수체계의 연산 : 공통적지의 올림 무시, 2의 보수를 취하면 부호이 진리반 수목

밸셈을 고려하면 2의보수체계는 처절은 따라오는지 (binary, hex) BCDX

CH9. MSI (medium scale integration) 논리회로

1, Decoders

ⅰ,

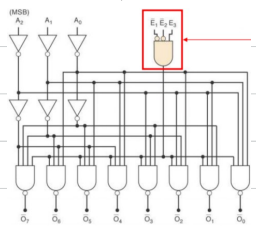


C	B	A	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

name
ⅰ, 3-line to 8-line decoder
ⅱ, binary to octal "
ⅲ, 1 of 8 "

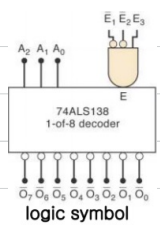
(only 1 of 8 ouputs is activated at one time)

ⅱ, Enable input: control the operation of decoder

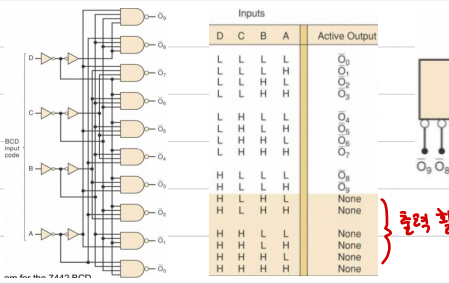


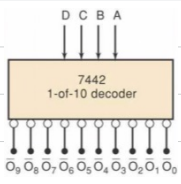
Truth table

$\bar{E}_1$	$\bar{E}_2$	E_3	Outputs
0	0	1	Respond to input code $A_2A_1A_0$
1	X	X	Disabled – all HIGH
X	1	X	Disabled – all HIGH
X	X	0	Disabled – all HIGH



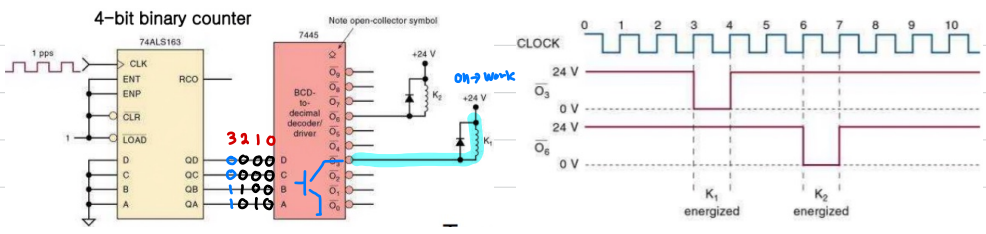
ⅲ, BCD-10진 디코더





{ 4 to 10 decoder
{ 1 of 10 decoder

ⅳ, BCD-10진 디코더/드라이버

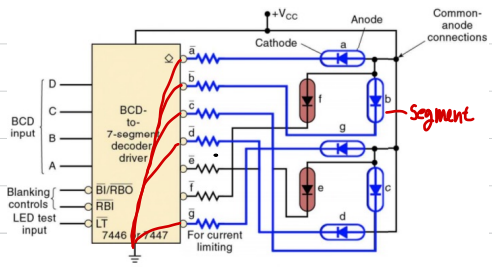


ⅴ, Application of Decoder

(1) Memory system이 널리 쓰이는데 특정 기억 장소를 활성화하기위해 중앙처리 장치에 의해 생성된 주소 코드에 따라 해당 장치 활성화

(2) 각각의 메모리 IC는 2진수를 저장할 레지스터로 구성, 다른 레지스터와 구분을 위해 각각의 주소 필요
디코더는 Memory Ic 회로 안에 장착 되어 있으며, 특정 저장레지스터가 해당입력 조합 주소로 적용될때 가동

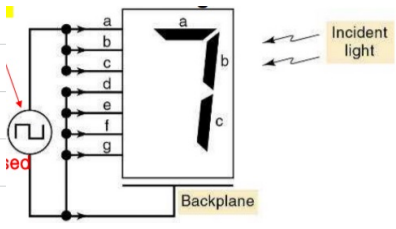
2. BCD to 7 segment decoder/driver [LED]



- 4비트 입력에 대한 출력을 10진수로 표시하기에 적당한 Segment의 점화 출력 (큰 전류 값필요)
- 디코더는 active-Low 출력을 갖고 있고 개방 콜렉터 드라이버
- Common-anode : 모든 세그먼트의 애노드들이 V_{CC}에 묶여있음

cf, Common-cathode : 캐소드들이 서로 묶여서 접지, 디코더는 active-High 출력을 가짐

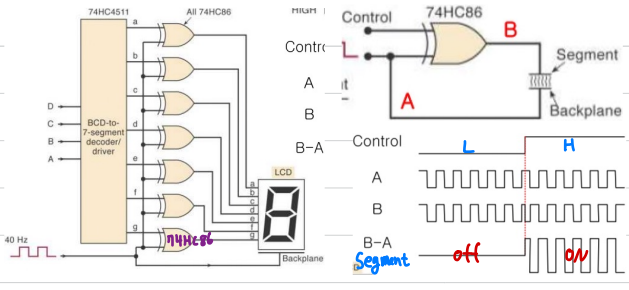
3. Liquid Crystal Display [LCD]



ⅰ, concept.

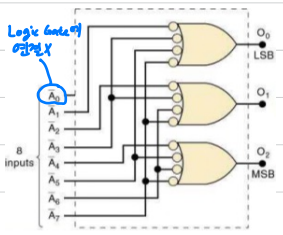
- 빛이 표시장치에 부딪혀서 반사를 이용하여 표시
- Segment를 동작하게 위해 교류전압이 공통인 후면판과 각 segment 사이에 인가.

ⅱ, driving



- 제어입력이 High이면 Segment 전압은 후면판에 비해 5V ~ -5V로 반복
  => "ON"

4, Encoder

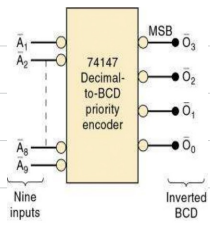


8 line to 3line encoder

Inputs								Outputs		
$\bar{A}_0$	$\bar{A}_1$	$\bar{A}_2$	$\bar{A}_3$	$\bar{A}_4$	$\bar{A}_5$	$\bar{A}_6$	$\bar{A}_7$	O_2	O_1	O_0
X	1	1	1	1	1	1	1	0	0	0
X	0	1	1	1	1	1	1	0	0	1
X	1	0	1	1	1	1	1	0	1	0
X	1	1	0	1	1	1	1	0	1	1
X	1	1	1	0	1	1	1	1	0	0
X	1	1	1	1	0	1	1	1	0	1
X	1	1	1	1	1	0	1	1	1	0
X	1	1	1	1	1	1	0	1	1	1

ⅰ, concept
여러개의 입력선중 하나가 시간에 동작하고 N비트의 출력코드가 생성

ⅱ, 한 번에 하나의 입력만 활성되어야 함 ⇒ 2개이상 입력이 동작할때 출력에 우선순위가 있어야 함



$\bar{A}_1$	$\bar{A}_2$	$\bar{A}_3$	$\bar{A}_4$	$\bar{A}_5$	$\bar{A}_6$	$\bar{A}_7$	$\bar{A}_8$	$\bar{A}_9$	$\bar{O}_3$	$\bar{O}_2$	$\bar{O}_1$	$\bar{O}_0$
1	1	1	1	1	1	1	1	1	1	1	1	1
X	X	X	X	X	X	X	X	0	0	1	1	0
X	X	X	X	X	X	X	0	1	0	1	1	1
X	X	X	X	X	X	0	1	1	1	0	0	0
X	X	X	X	X	0	1	1	1	1	0	0	1
X	X	X	X	0	1	1	1	1	1	0	1	0
X	X	X	0	1	1	1	1	1	1	0	1	1
X	X	0	1	1	1	1	1	1	1	1	0	0
X	0	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	0

(1001)
(1000)

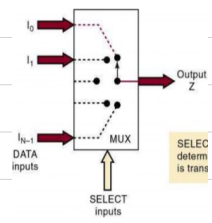
· active-Low 입력,출력

출력들은 각각을 인버터에 통과시켜서 정상적인 BCD로 변환 가능

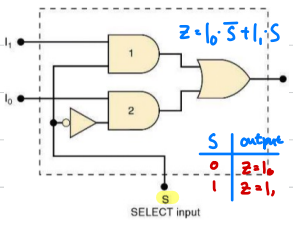
6. Multiplexer (Data selector)

ⅰ, concept

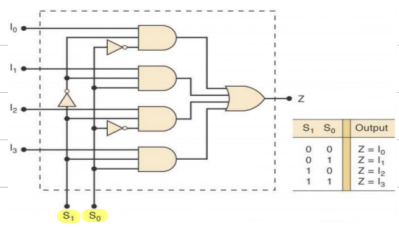
- 여러개의 입력을 받아 그 중 하나를 선택하여 출력
- 출력할 데이터의 입력 루트는 Select input에의해 제어 (흔히 ADDRESS 입력으로 지칭)



ⅱ, two input MUX

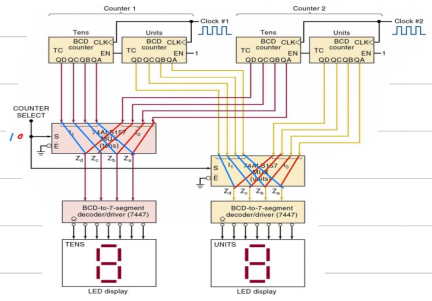


ⅲ, four-input MUX



7. MUX Application

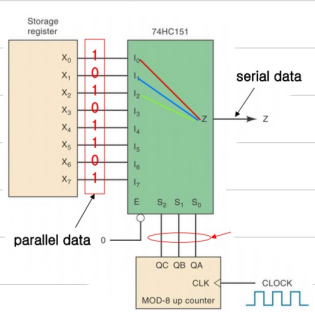
ⅰ, data routing : 2개의 BCD 카운터중 하나를 선택하고 표시

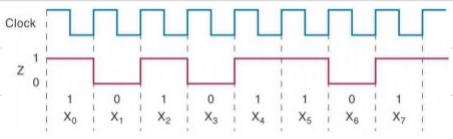


Counter select = Low
=> Counter 2 가 출력이 MUX을지남

Counter select = High
=> counter 1 출력 MUX를 지남

ⅱ, Parallel to serial conversion (먼 거리 전송하기 위해)

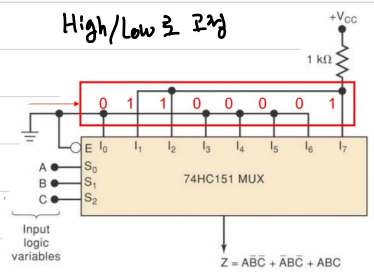




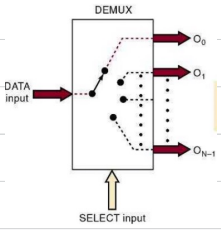
MOD-8 는 Select input $S_2S_1S_0$을 기준으로, 000 → 111로순환

ⅲ, Logic function generation : truth table에서 직접 논리 기능을 구현, 각 입력은 truth table에 따라 High/Low로 고정

C	B	A	Z
0	0	0	0 (I_0)
0	0	1	1 (I_1)
0	1	0	1 (I_2)
0	1	1	0 (I_3)
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1 (I_7)



8. Demultiplexers [DMUX] 분배

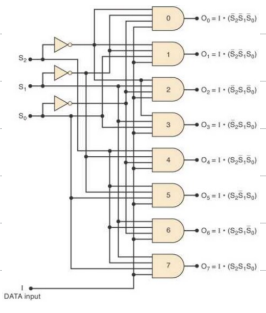


ⅰ, concept

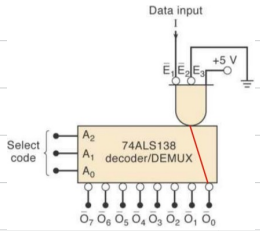
- 하나의 입력을 받아들여 여러개의 출력으로 분배

ⅱ, 1 line to 8 line DEMUX

하나의 입력선 I는 8개의 AND 게이트와 연결되지만, 이 중 하나만이 SELECT에 의해 enable



SELECT code			OUTPUTS							
S_2	S_1	S_0	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	0	0	0	0	0	0	0	I
0	0	1	0	0	0	0	0	0	I	0
0	1	0	0	0	0	0	0	I	0	0
0	1	1	0	0	0	0	I	0	0	0
1	0	0	0	0	0	I	0	0	0	0
1	0	1	0	0	I	0	0	0	0	0
1	1	0	0	I	0	0	0	0	0	0
1	1	1	I	0	0	0	0	0	0	0



디코더와 유사한데 2진코드 입력은 Select 입력으로 Enable 입력은 입력단자 I로 된다