Digital Logic

W. Bolton, "Mechatronics --- Electronic control systems in mechanical and electrical engineering," 5th edition, Pearson Education Limited 2012, Chap 5
J. Edward Carryer, R. Matthew Ohline, Thomas W. Kenny, "Introduction to Mechatronic Design," Prentice Hall 2011, Chap 18
線上學習網站：https://www.electronics-tutorials.ws
PowerPoint 中部分圖片擷取和修改自教科書和網路圖片

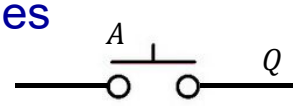
林沛群
國立台灣大學
機械工程學系

Definitions

- Digital logic
  - The manipulation of binary values that uses circuits and logic gates to construct the implementation of computer operations
- Logic gates
  - The building blocks for digital electronic circuits
  - Five main components
    - AND
    - OR
    - XOR
    - NAND
    - NOR

Logic Gates -1

- Yes



NO, Normally open

Boolean equation: $Q = A$

- NOT

NC, Normally closed

Boolean equation: $Q = \bar{A}$

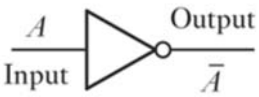
Represented by switches

Input	Output
A	$Q = \bar{A}$
0	1
1	0

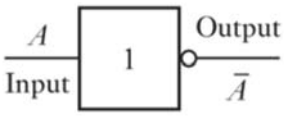
Truth table

Venn diagram:

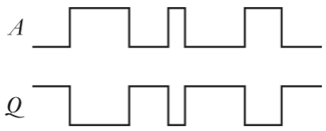
A diagram that shows all possible logical relations between a finite collection of different sets



US symbols



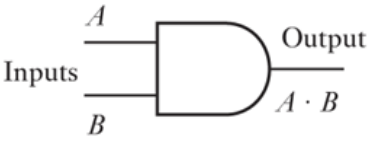
International standard (IEEE/ANSI)



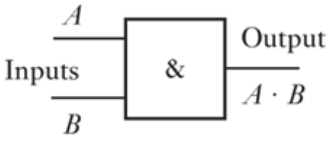
Logic Gates -2

- AND

Represented by switches
"in series"

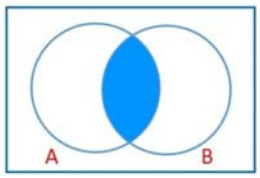


US symbols

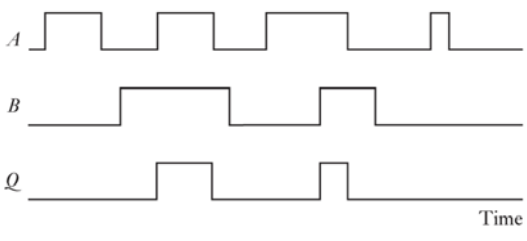


International standard
(IEEE/ANSI)

Boolean equation: $Q = A \cdot B$



Venn diagram

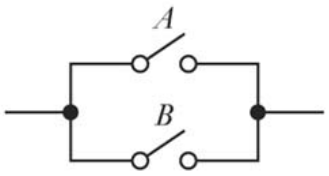


Input		Output
A	B	$Q = A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

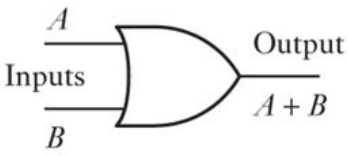
Truth table

Logic Gates -3

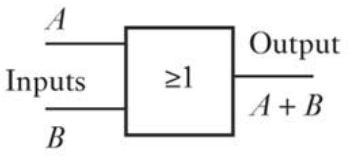
- OR



Represented by switches
"in parallel"

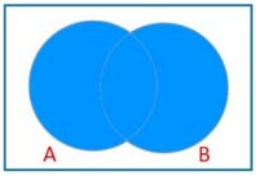


US symbols

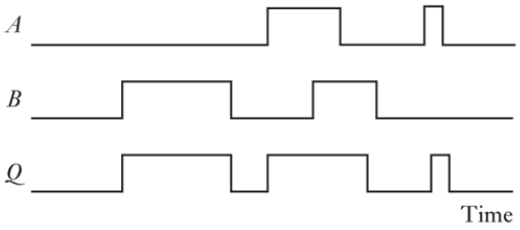


International standard
(IEEE/ANSI)

Boolean equation: $Q = A + B$



Venn diagram



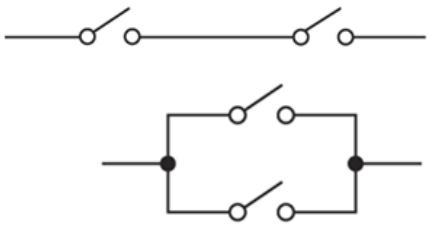
Input		Output
A	B	$Q = A + B$
0	0	0
0	1	1
1	0	1
1	1	1

Truth table

Boolean Algebra -1

- Operations: $0,\ 1,\ \cdot (AND), +(OR)$

$0 \cdot 0 = 0$
$0 + 0 = 0$
$1 \cdot 1 = 1$
$1 + 1 = 1$ ⇦ Different from arithmetic operations
$1 \cdot 0 = 0 \cdot 1 = 0$
$1 + 0 = 0 + 1 = 1$



- Commutative property

$X \cdot Y = Y \cdot X$
$X + Y = Y + X$

When $Y = 0$
$X \cdot 0 = 0$
$X + 0 = X$

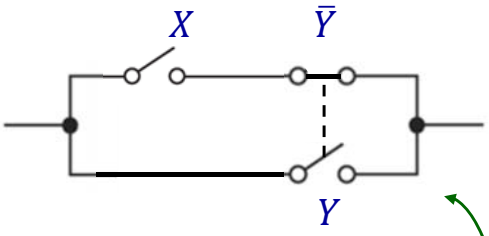
When $Y = 1$
$X \cdot 1 = X$
$X + 1 = 1$

When $Y = X$
$X \cdot X = X$
$X + X = X$

When $Y = \bar{X}$
$X \cdot \bar{X} = 0$
$X + \bar{X} = 1$

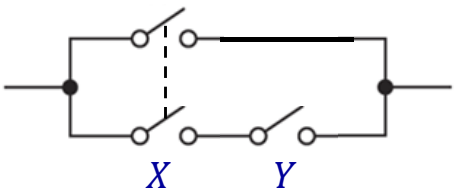
Boolean Algebra -2

- Sequence of operations
  - Negation (inversion)
  - Multiplication
  - Addition



- Distribution property

$$X \cdot (Y + Z) = X \cdot Y + X \cdot Z$$



Absorption laws

$$X + X \cdot Y = X(1 + Y) = X$$
$$X \cdot (X + Y) = X \cdot X + X \cdot Y = X$$
$$(X + \bar{Y}) \cdot Y = XY$$
$$X \cdot \bar{Y} + Y = X + Y$$
$$X \cdot Y + \bar{Y} = X + \bar{Y}$$

Boolean Algebra -3

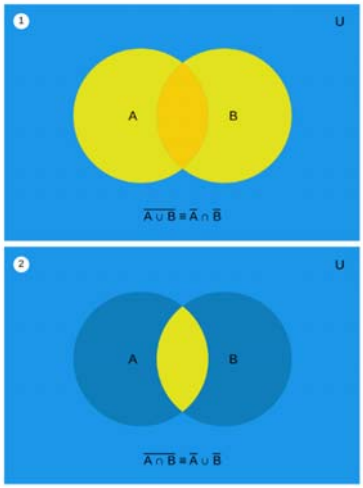
- De Morgan's Laws

$$\overline{(X + Y + Z + \cdots)} = \bar{X} \cdot \bar{Y} \cdot \bar{Z} \cdot \cdots$$
$$\overline{(X \cdot Y \cdot Z \cdot \cdots)} = \bar{X} + \bar{Y} + \bar{Z} + \cdots$$

  - Two variables

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

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



Boolean Algebra -4

- Multiple variables

" · " is omitted

$X + Y + Z = (X + Y) + Z = X + (Y + Z)$

$XYZ = (XY)Z = X(YZ)$

$(X + Y)(X + Z) = XX + XY + XZ + YZ = X(1 + Y + Z) + YZ = X + YZ$

$(X + Y)(X + Z)(Z + \bar{X}) = (X + Y)\,(Z + \bar{X})$

$(X + Y)\,(Z + \bar{X}) = XZ + \bar{X}Y$

$XY + YZ + Z\bar{X} = XY + Z\bar{X}$

⇧

How to prove? Use this as the example

Boolean Algebra -5

- Algebraic proof

$XY + YZ + Z\bar{X} = XY + 1YZ + Z\bar{X} = XY + (X + \bar{X})YZ + Z\bar{X}$

$= XY + XYZ + \bar{X}YZ + Z\bar{X} = XY(1 + Z) + (Y + 1)Z\bar{X}$

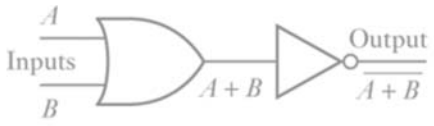
$= XY + Z\bar{X}$

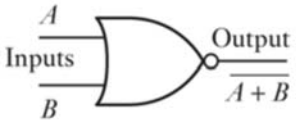
- Brute force

X	Y	Z	XY	YZ	$Z\bar{X}$	$XY + YZ + Z\bar{X}$	$XY + Z\bar{X}$
0	0	0	0	0	0	0	0
0	0	1	0	0	1	1	1
0	1	0	0	0	0	0	0
0	1	1	0	1	1	1	1
1	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	1	1	0	1	1

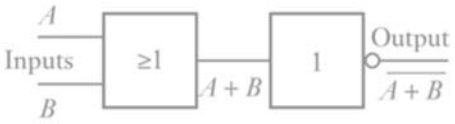
Logic Gates -4

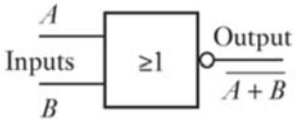
- NOR = OR + NOT





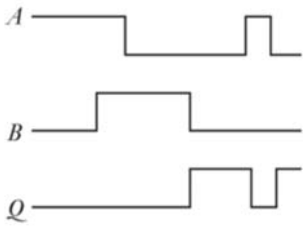
US symbols





International standard (IEEE/ANSI)

Boolean equation: $Q = \overline{A + B} = \bar{A} \cdot \bar{B}$

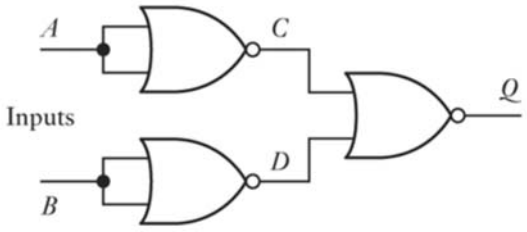


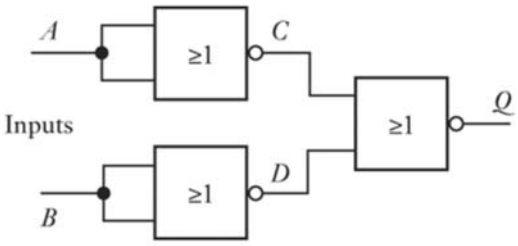
Input		Output
A	B	$Q = \overline{A + B}$
0	0	1
0	1	0
1	0	0
1	1	0

Truth table

Logic Gates -5

- NOR: a universal gate
  - AND: Using three NOR gates





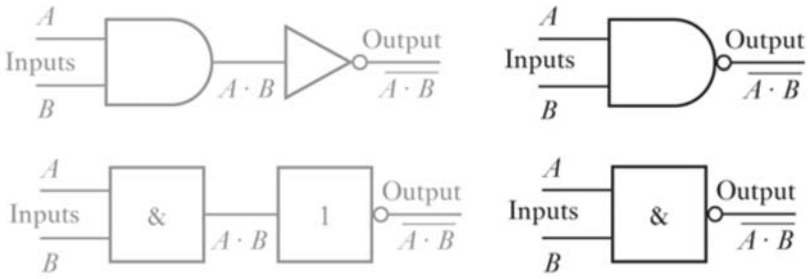
Input				Output
A	B	C	D	$Q = A \cdot B$
0	0	1	1	0
0	1	1	0	0
1	0	0	1	0
1	1	0	0	1

$$Q = \overline{C + D}$$
$$= \bar{C} \cdot \bar{D}$$
$$= \overline{\overline{A + A}} \cdot \overline{\overline{B + B}}$$
$$= A \cdot B$$
= AND

Using NOR to make other gates such as NOT, OR, NAND, XOR…?

Logic Gates -6

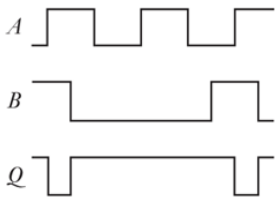
- NAND = AND + NOT



US symbols

International standard (IEEE/ANSI)

Boolean equation: $Q = \overline{A \cdot B} = \bar{A} + \bar{B}$

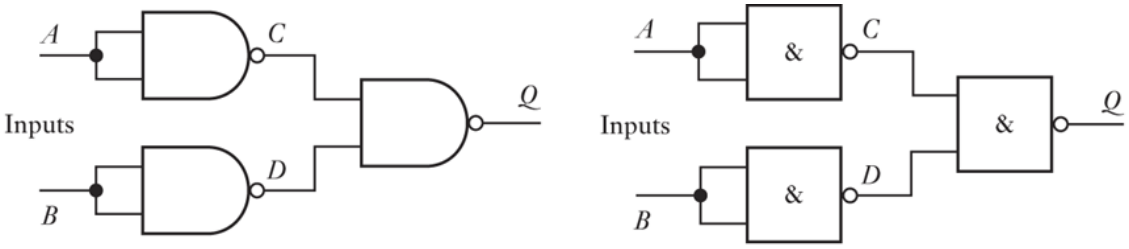


Input		Output
A	B	$Q = \overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

Truth table

Logic Gates -7

- NAND: a universal gate
  - Duality: NOR & NAND
  - OR: Using three NAND gates

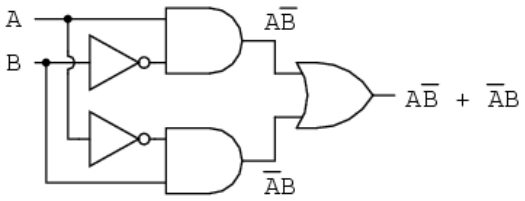


Input				Output
A	B	C	D	$Q = A + B$
0	0	1	1	0
0	1	1	0	1
1	0	0	1	1
1	1	0	0	1

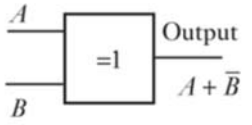
$$Q = \overline{C \cdot D}$$
$$= \bar{C} + \bar{D}$$
$$= \overline{A \cdot A} + \overline{B \cdot B}$$
$$= A + B$$
$$= \text{OR}$$

Logic Gates -8

- XOR

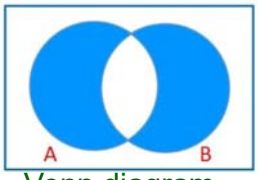


US symbols

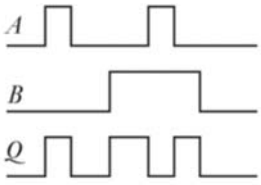


International standard (IEEE/ANSI)

Boolean equation: $Q = A \oplus B = A \cdot \bar{B} + \bar{A} \cdot B$



Venn diagram

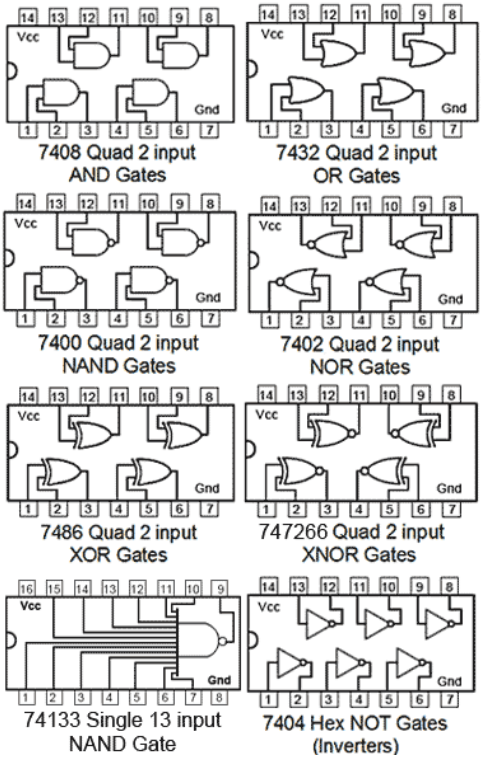


Input		Output
A	B	$Q = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Truth table

Logic Families and Integrated Circuits -1

- Families
  - Transistor-transistor logic (TTL)
  - Complementary metal0oxide semiconductor (CMOS)
  - Emitter-coupled logic (ECL)
- Integrated circuits
  - 74XX series (TTL)
    - 74YYXX, YY=(none), LS, F…
  - 4000 series (CMOS)
  - 74CXX, 74HCXX (CMOS)

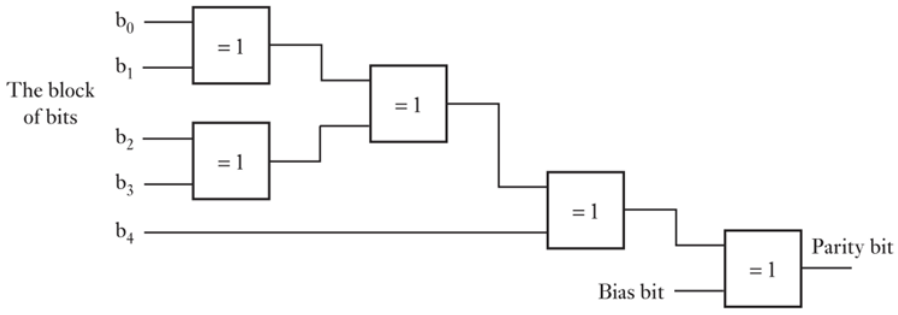


Logic Families and Integrated Circuits -2

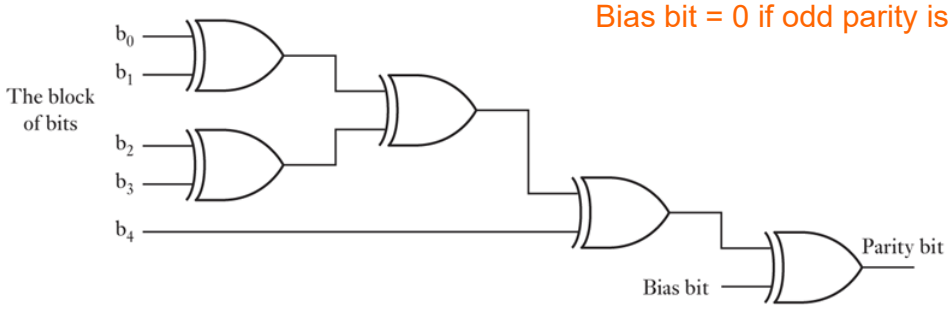
- Parameters
  - Logic level
  - Noise immunity
  - Fan-out: The number of gate inputs that can be driven by a standard gate output while maintaining the desired LOW or HIGH levels
  - Current-sourcing or current-sinking action
  - Propagation delay time
  - Power consumption

Applications of Logic Gates -1

- Parity generators
  - Using XOR: output 0 if all the inputs are 0 or all are 1; otherwise output 1

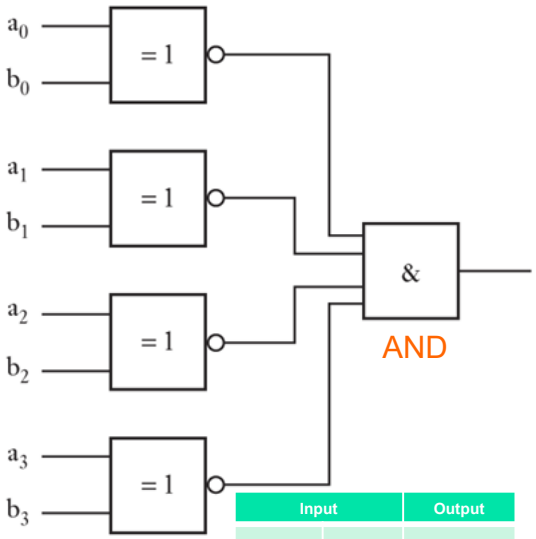


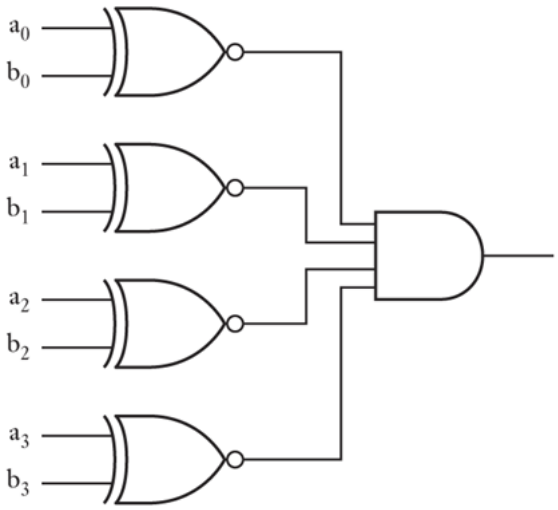
Bias bit = 0 if odd parity is required



Applications of Logic Gates -2

- Digital comparator
  - Comparing two digital words to determine if they are exactly equal



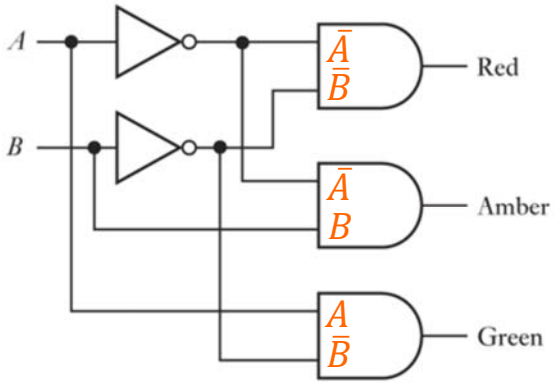


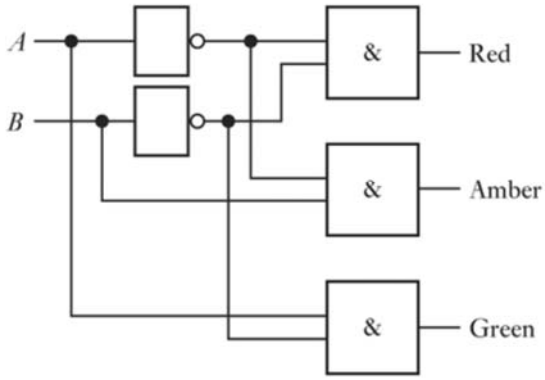
XNOR gate = XOR + NOT

Input		Output
A	B	$Q = A + \bar{B}$
0	0	1
0	1	0
1	0	0
1	1	1

Applications of Logic Gates -3

- Coder / decoder
  - Ex: Traffic lights

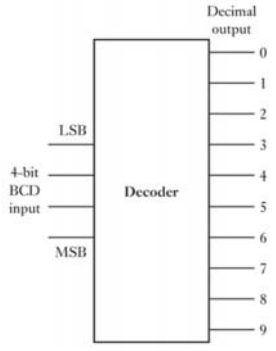


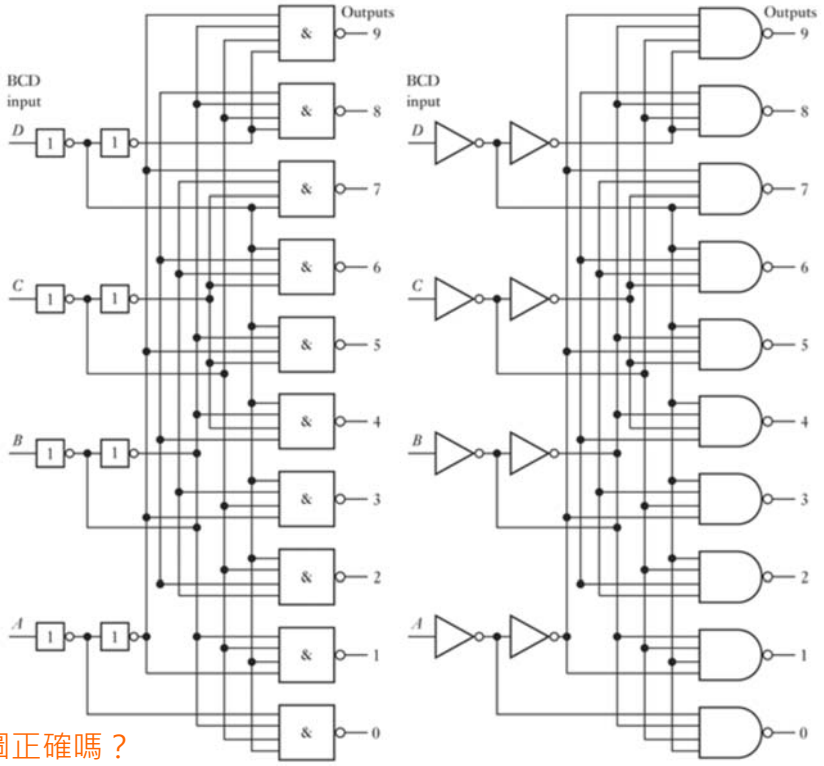


A	B	$\bar{A}$	$\bar{B}$	
0	0	1	1	Red
0	1	1	0	Amber
1	0	0	1	Green

Applications of Logic Gates -4

- Code converter
  - Ex: BCD-to-decimal





圖正確嗎？

  - Ex: 74LS244 --- 4-bit BCD to seven segments of a display

Applications of Logic Gates -5

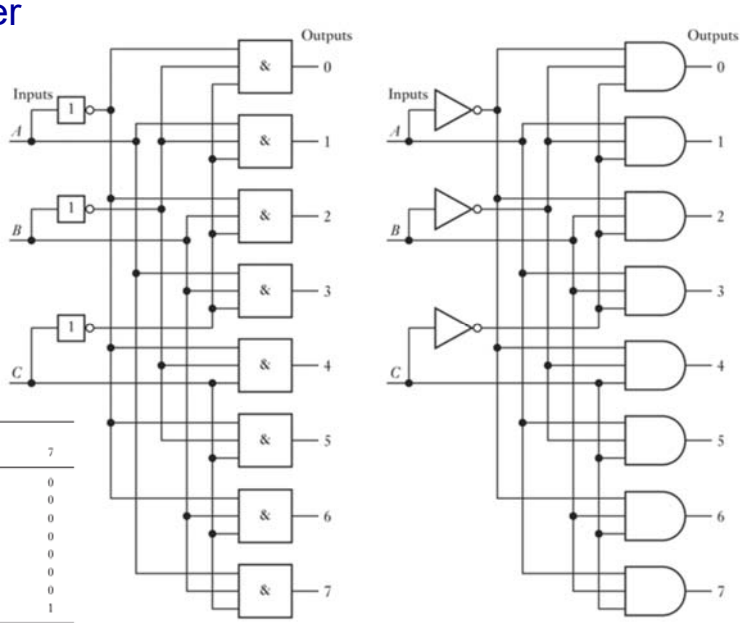
- Number systems

Binary-coded decimal system

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

Applications of Logic Gates -6

- Decoder
  - Ex: 3-line to 8-line decoder

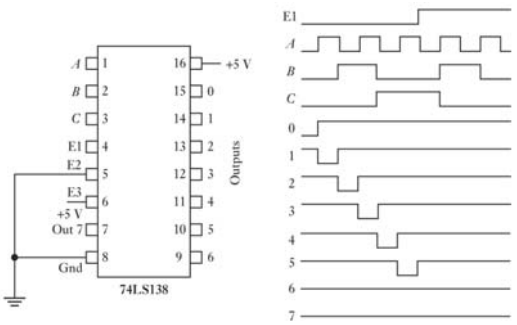


Inputs C	B	A	Outputs 0	1	2	3	4	5	6	7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

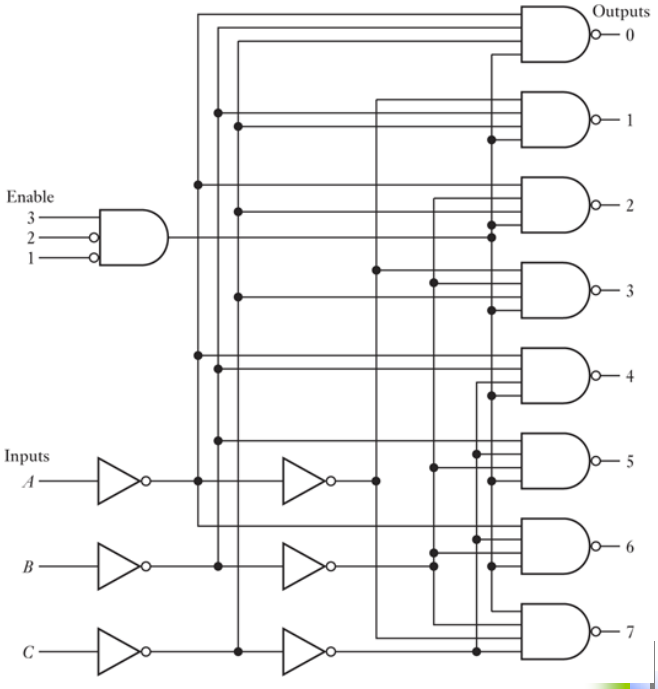
  - Ex: 74LS138

Applications of Logic Gates -7

- Decoder
  - Ex: 74LS138, 3-line to 8-line decoder with ENABLE inputs



Enable E1	E2	E3	Inputs C	B	A	Outputs 0	1	2	3	4	5	6	7
1	X	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X	X	1	1	1	1	1	1	1	1
X	X	0	X	X	X	1	1	1	1	1	1	1	1
0	0	1	0	0	0	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	0	1	1	1
0	0	1	1	0	1	1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	0



Simplification of Binary Functions -1

- Where do these complicate functions come from?
  - The result of translating the practical requirements of a given control problem into the logic language of Boolean algorithm
- Why do we want to simplify?
  - Save number of logic gates
  - Easier to understand the consequence

Simplification of Binary Functions -2

- Method 1 – Algebraic method
- Ex:

$$ABCD + AB\bar{C} + AB\bar{D}$$ 2 NOT, 3 AND, 1 OR

$$= AB(CD + \bar{C} + \bar{D})$$

$$= AB(\bar{C} + D + \bar{D})$$

$$= AB(\bar{C} + 1)$$

$$= AB1$$

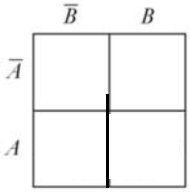
$$= AB$$ 1 AND

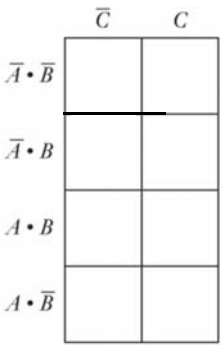
Simplification of Binary Functions -3

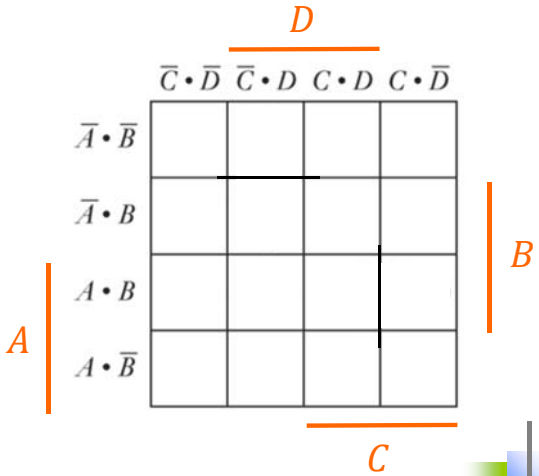
- Method 2 – Karnaugh Map
  - A graphical method that can be used to produce simplified Boolean expressions from sums of products obtained from truth tables
  - Advantages
    - No need tricks
    - Faster and simpler
    - Always produces the simplest possible result
    - Include “don’t care” conditions

Karnaugh Map -1

- Generation of map
  - Any two adjacent squares MUST differ in only one variables (top-bottom left-right edges also considered adjacent)



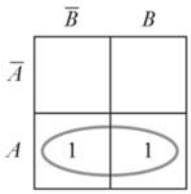




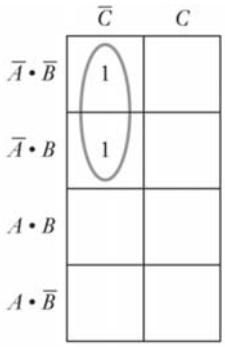
Karnaugh Map -2

- Procedure
  - Fill “1” into the map, then “0” for the rest
  - Loop adjacent “1” into cells of 2,4,8 cells
    - Each “1” must be covered by at least one cell
    - Cell number: as small as possible
    - Cell size: as large as possible

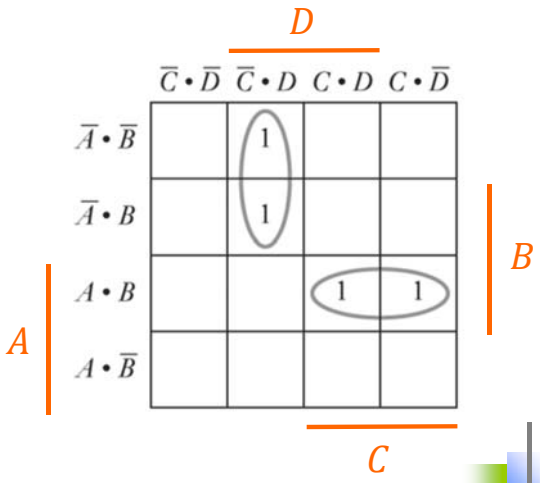
$$T = ABCD + ABC\bar{D} + \bar{A}B\bar{C}D + \bar{A}\bar{B}\bar{C}D$$



$$T = AB + A\bar{B}$$



$$T = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C}$$

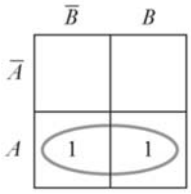


Karnaugh Map -3

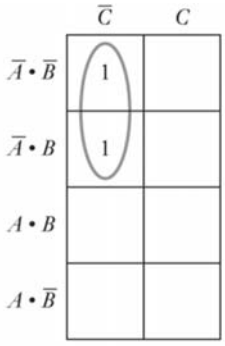
- Write new logic

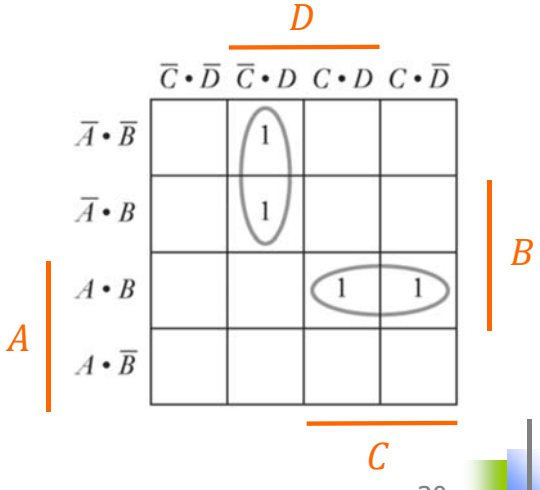
$$T = ABCD + ABC\bar{D} + \bar{A}B\bar{C}D + \bar{A}\bar{B}\bar{C}D$$
$$= \bar{A}\bar{C}D + ABC$$

$$T = AB + A\bar{B}$$
$$= A$$



$$T = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C}$$
$$= \bar{A}\bar{C}$$

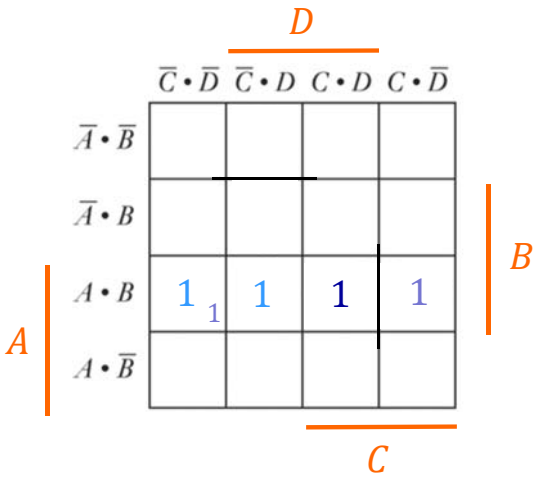




Karnaugh Map -4

- Ex: Revisit the same example

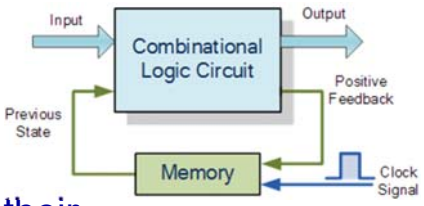
$$ABCD + AB\bar{C} + AB\bar{D}$$



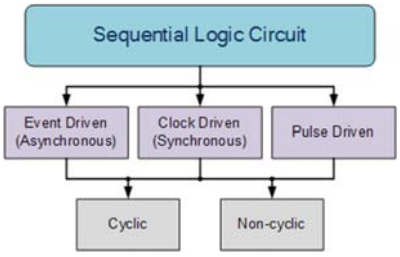
$\Rightarrow T = AB$

Sequential Logic -1

- A type of logic circuit whose output depends on
  - The present value of its input signals
  - The sequence of past inputs, “memory”
  - Generally a bistable device which can have their output or outputs set in one of two basic states



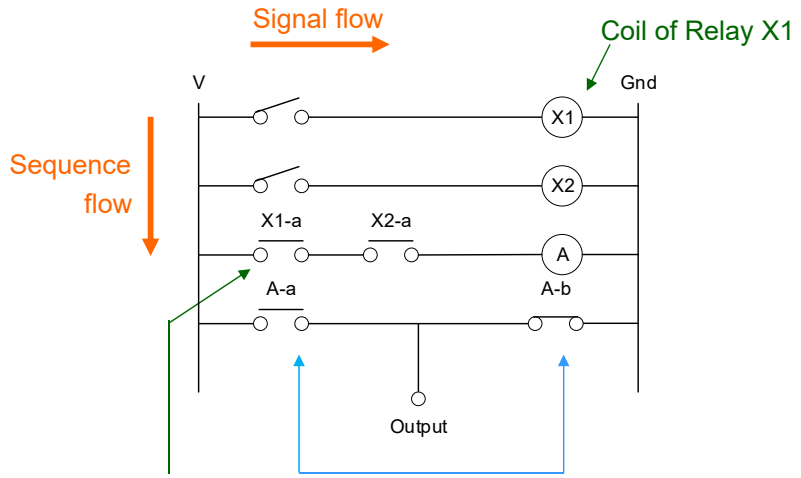
- Classification
  - Event driven, asynchronous circuits that change state immediately when enabled
  - Clock driven, synchronized to a specific clock signal
  - Pulse driven, a combination of the two

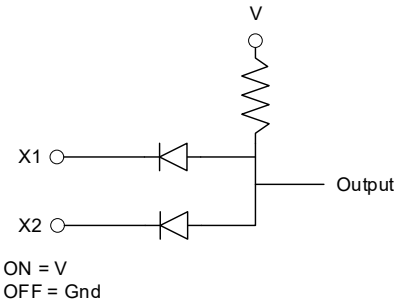


Sequential Logic -2

- Basic gates using relays
  - AND

Ladder diagram



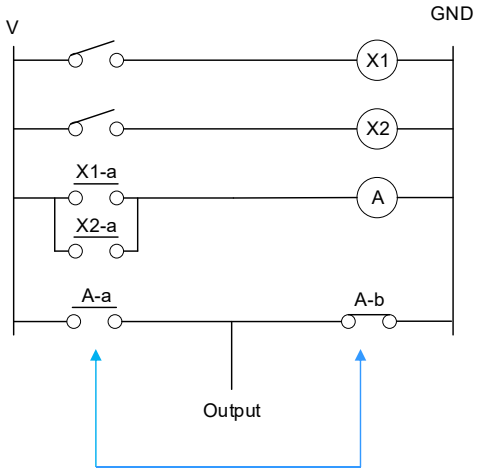


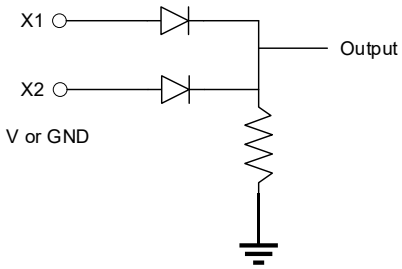
Switch of Relay X1
a: normally open
b: normally closed

NAND = AND + NOT
Switch A-a & A-b

Sequential Logic -3

- Basic gates using relays
  - OR

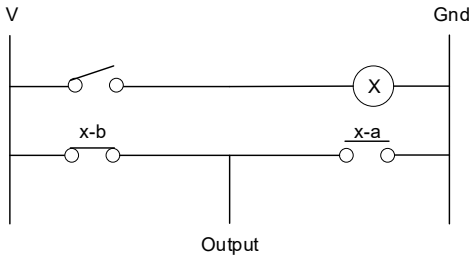


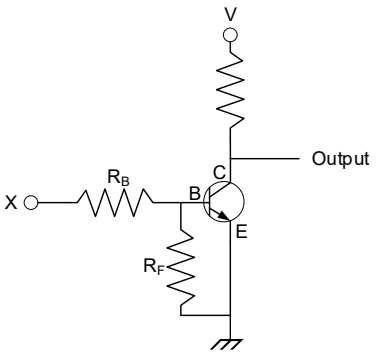


NOR = OR + NOT
Switch A-a & A-b

Sequential Logic -4

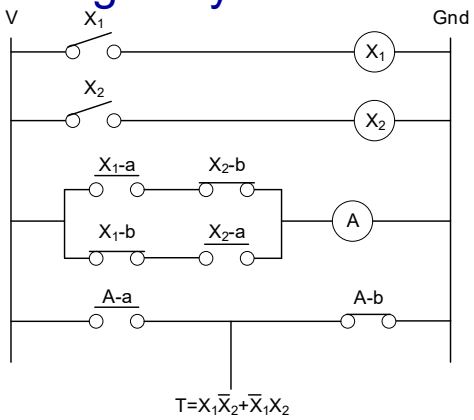
- Basic gates using relays
  - NOT

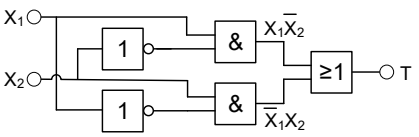




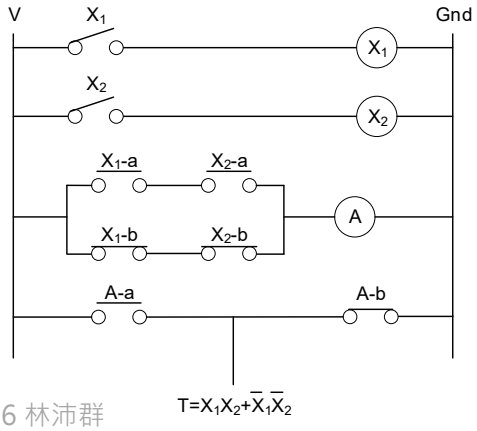
Sequential Logic -5

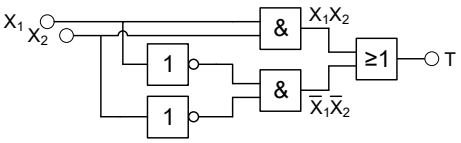
- Basic gates using relays
  - XOR





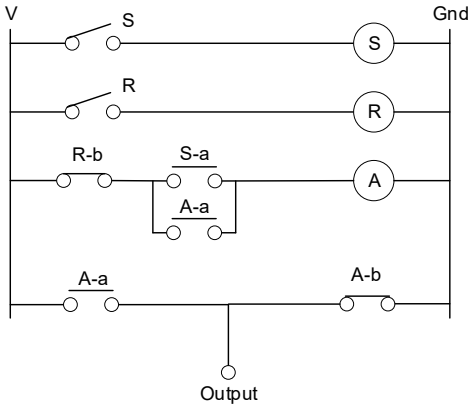
  - 一致電路

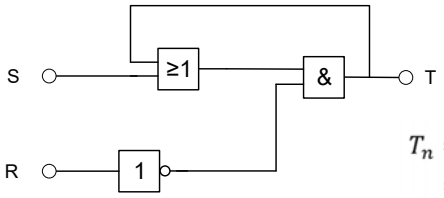




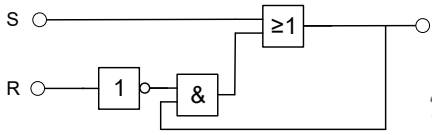
Sequential Logic -6

- Latching function 自我保持, a SR flip-flop circuit

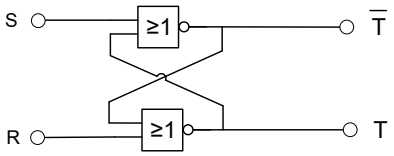




$$T_n = (S + T_{n-1}) \cdot \bar{R}$$
$$= S\bar{R} + T_{n-1}\bar{R}$$

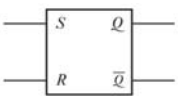


$$T_n = S + \bar{R} \cdot T_{n-1}$$

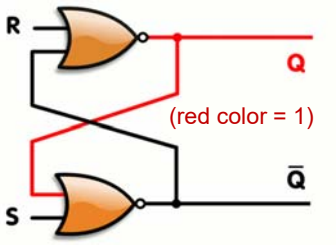


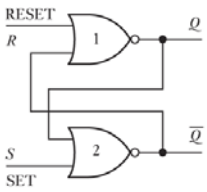
Sequential Logic -7

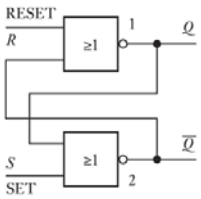
- SR (set-reset) flip-flop

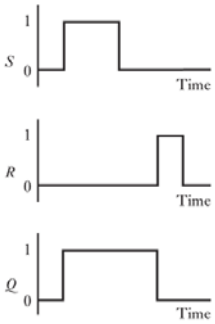


$$Q_{n+1} = \bar{R} \cdot (Q_n + S)$$









Q_n	S	R	Q_{n+1}	Action
0	0	--	0	Hold state
0	1	0	1	Set
1	0	1	0	Reset
1	--	0	1	Hold state

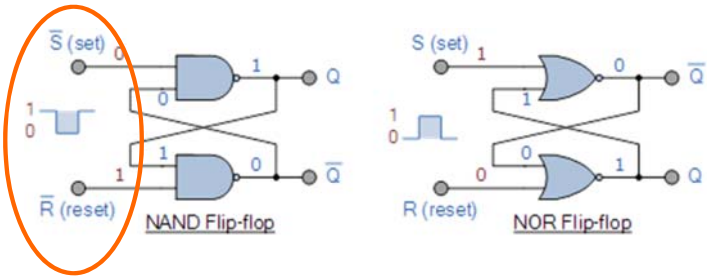
Excitation table (-- doesn't matter)

Q_n	S	R	Q_{n+1}
0	0	0	0
0	1	0	1
0	0	1	0
0	1	1	X
1	0	0	1
1	1	0	1
1	0	1	0
1	1	1	X

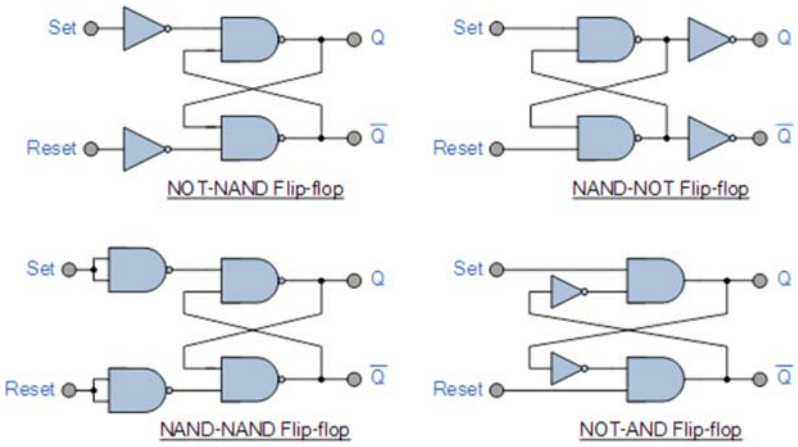
Truth table

Sequential Logic -8

- SR flip-flop
  - Basic forms

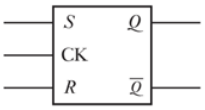


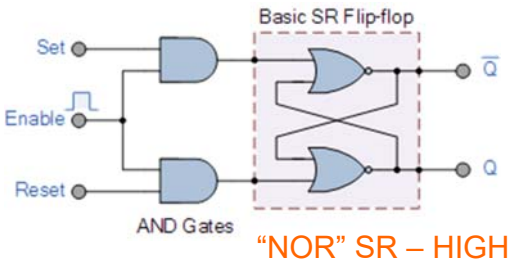
  - Active HIGH

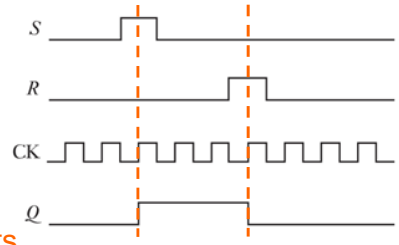


Sequential Logic -9

- Clocked (Gated) SR flip-slop
  - A synchronous system, to regulate the set/reset timings

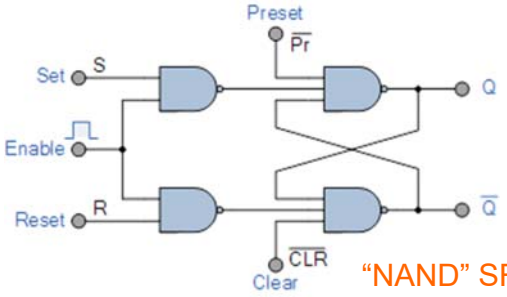






"NOR" SR – HIGH inputs

- SR Flip-flop with Preset and Clear Inputs

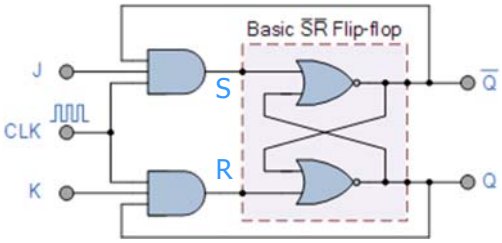


"NAND" SR – LOW inputs

Sequential Logic -10

- Clocked JK flip-flop
  - = Clocked RS + "flip" (when $J \cdot K = 1$)

$$Q_{n+1} = J \cdot \overline{Q_n} + \overline{K} \cdot Q_n$$



Q_n	J	K	Q_{n+1}	Action
Q_n	0	0	Q_n	Hold state
0	1	0	1	Set
1	0	1	0	Reset
Q_n	1	1	$\overline{Q_n}$	Toggle

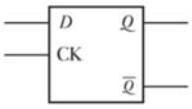
Excitation table

Q_n	J	K	Q_{n+1}
0	0	0	0
0	1	0	1
0	0	1	0
0	1	1	1
1	0	0	1
1	1	0	1
1	0	1	0
1	1	1	0

Truth table

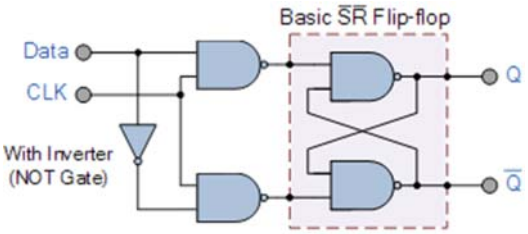
Sequential Logic -11

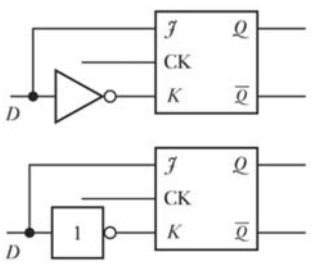
- D flip-flop

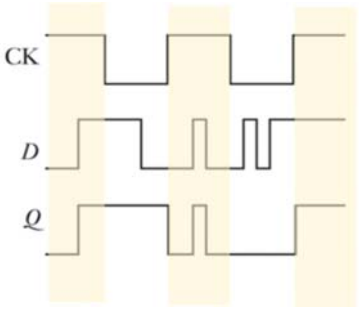


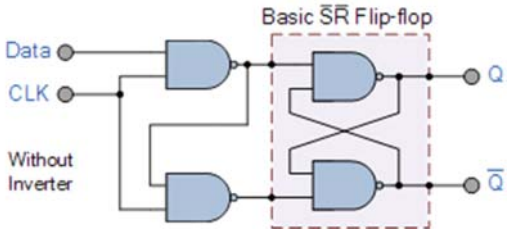
  - = SR/JK + NOT
  - One control input, forcing $S \cdot R = 0$ or $J \cdot K = 0$
  - $D = 1$ set (S); $D = 0$ reset (R)

$$Q_{n+1} = D$$





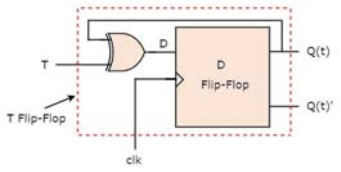


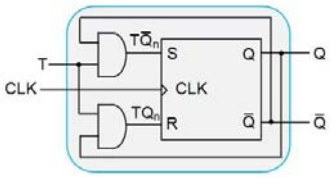


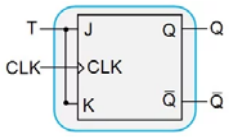
Sequential Logic -12

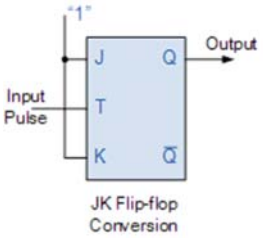
- T flip-flop

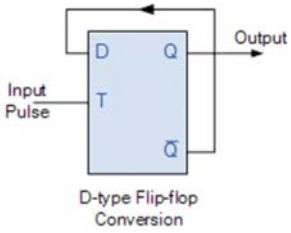
$$Q_{n+1} = T \oplus Q_n = T \cdot \overline{Q_n} + \overline{T} \cdot Q_n$$











$J = K = T$

"flip" when $T = 1$

$J \cdot K = 1$

"flip" when $T = 1$

Q_n	T	Q_{n+1}	Action
Q_n	0	Q_n	Hold state
Q_n	1	$\overline{Q_n}$	Toggle

Characteristic table

Q_n	T	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

Truth table

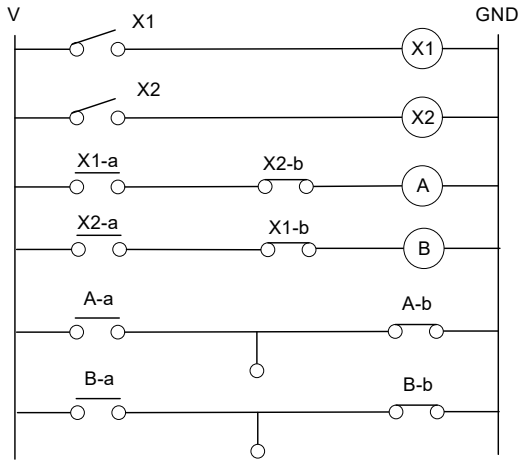
Sequential Logic -13

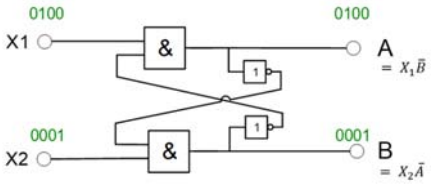
- Popular flip flop ICs

Device Number	Device Description
74LS73A	Dual Negative-edge Triggered JK Flip-flop with Clear
74LS74	Dual Positive-edge Triggered D-type with Preset and Clear
74LS75	Quad D-type Bistable Latch with Enable
74LS76	Dual Pulse Triggered JK Flip-flops with Preset and Clear
74LS107	Dual JK Flip-Flop with Clear
74LS111	Dual Master-slave JK Flip-flop with Clear
74LS175	Dual Positive-edge Triggered D-type with Clear
74LS279	Quad SR Latches with Active-LOW inputs

Sequential Logic -14

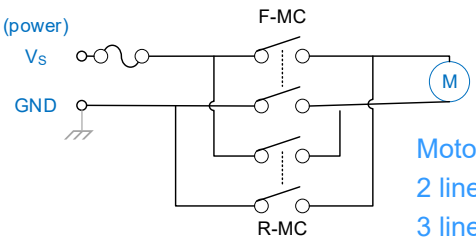
- Interlock function 互鎖電路



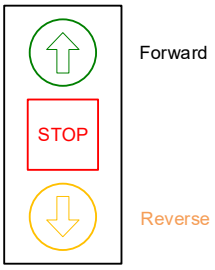


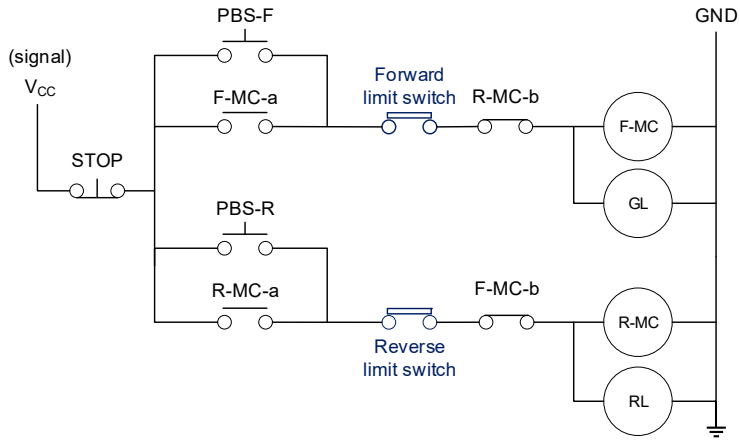
Sequential Logic -15

- Motor forward/reverse control using relays



Motor:
2 lines: DC, AC (110V – 1p, 220V – 1p)
3 lines: AC (220V 3p)





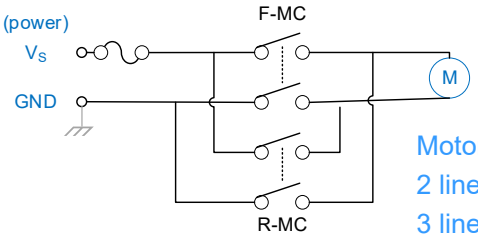
Features:
Power-Signal separation
Latching
Forward/reverse interlock
Auto-stop (limit switches)

Multiple sets of control switches?
(onsite & remote)
Forward/reverse – parallel
Stop - serial

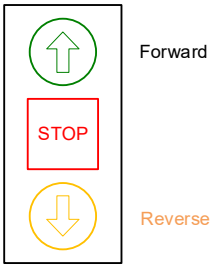
Sequential Logic -16

- Motor forward/reverse control using logic gates

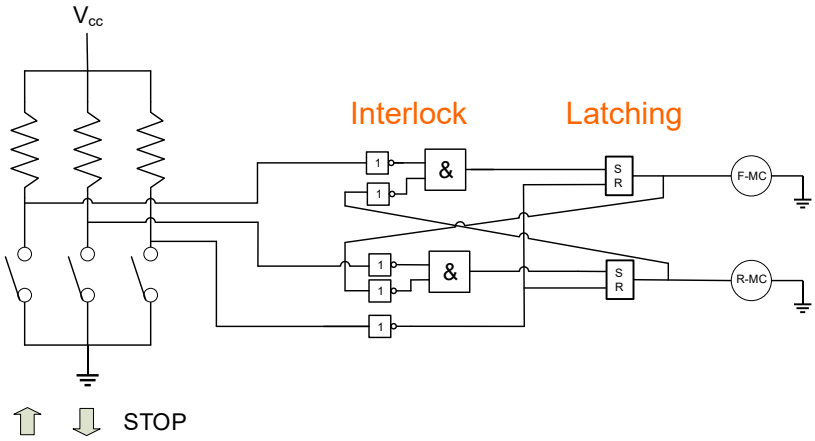
Driving circuit



Motor:
2 lines: DC, AC (110V – 1p, 220V – 1p)
3 lines: AC (220V 3p)

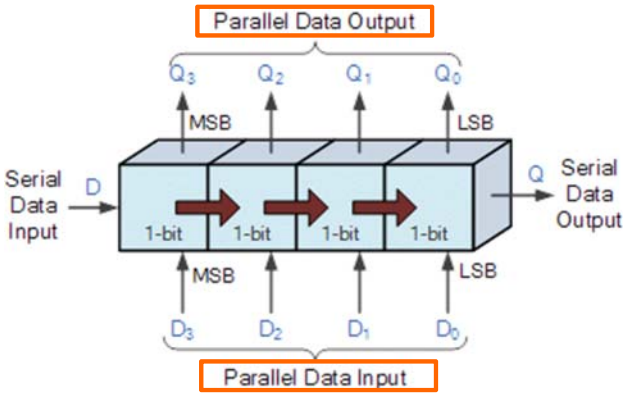


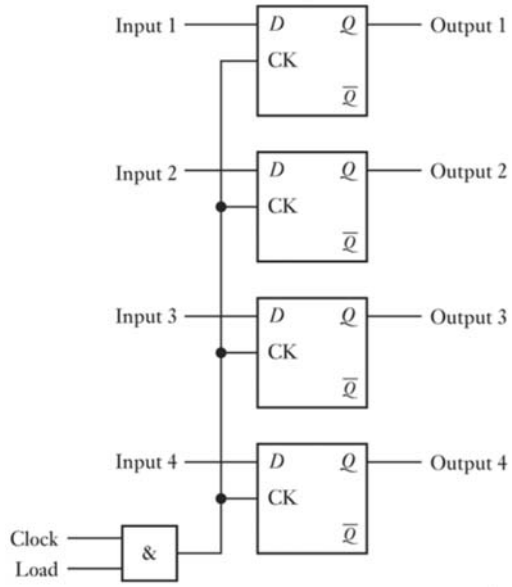
Control circuit



Registers -1

- Register
  - A set of memory elements which is used to hold information until it is needed



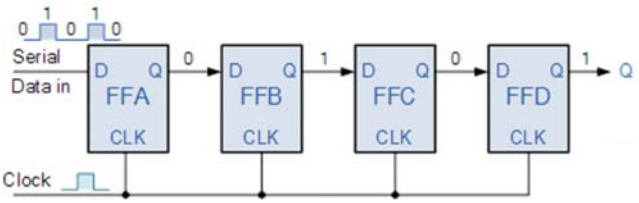


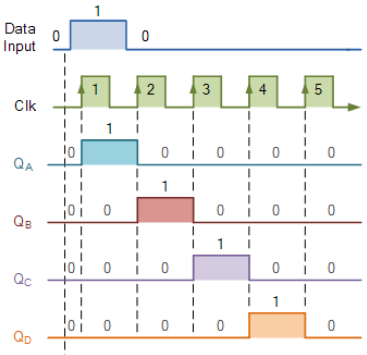
Use “load” to control update timing

Registers -2

- Shift register
  - A cascade of flip flops, sharing the same clock, in which the output of each flip-flop is connected to the "data" input of the next flip-flop in the chain, resulting in a circuit that shifts by one position the "bit array" stored in it

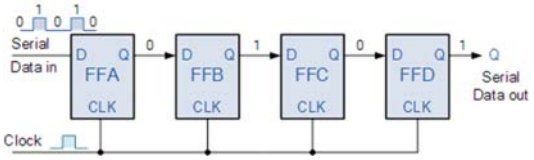
Clock Pulse No	QA	QB	QC	QD
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1
5	0	0	0	0



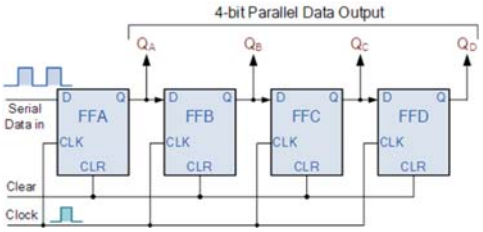


Registers -3

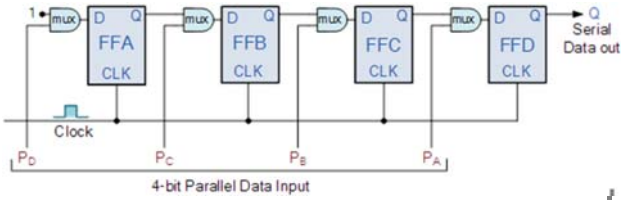
- Configurations
  - Serial-in serial-out (SISO)



  - Serial-in parallel-out (SIPO)
    - Conversion between serial and parallel interfaces

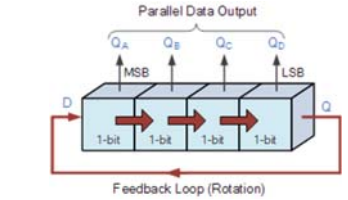


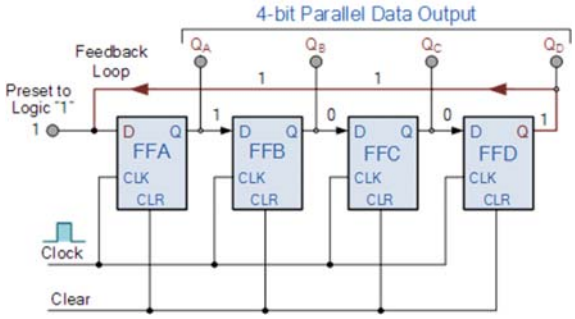
  - Parallel-in serial-out (PISO)

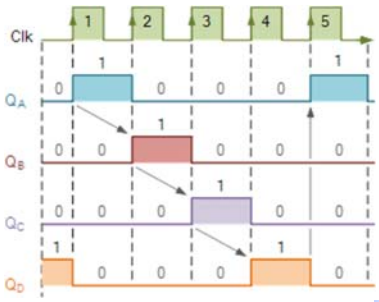


Ring Counter -1

- Ring counter
  - A type of counter composed of flip-flops connected into a shift register, with the output of the last flip-flop fed to the input of the first, making a "circular" or "ring" structure
  - Ex: A 4-bit ring counter, “mod-4”

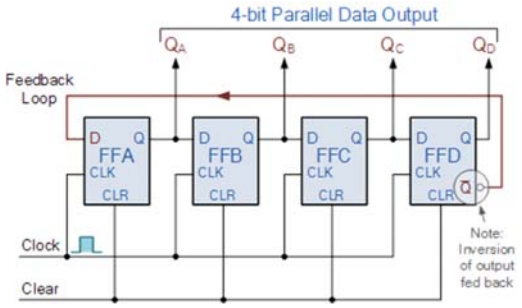






Ring Counter -2

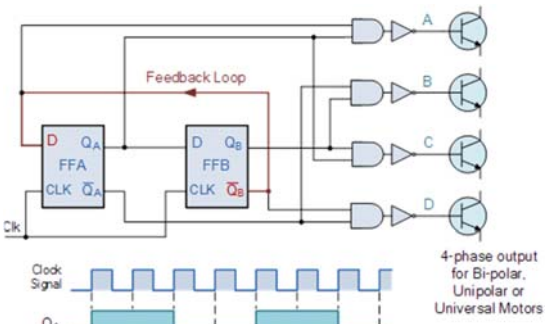
- Johnson (Twisted) ring counter
  - Connecting $\bar{Q}$ of the last flip-flop back to the input D of the first flip-flop
  - n -stage counter → $2n$ states
  - Ex: A 4-bit Johnson ring counter, “mod-8”
    - Four “0”s and four “1”s, 8-bit pattern

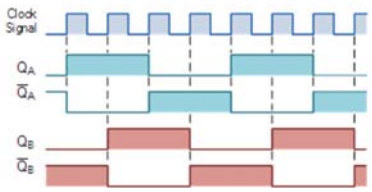


Clock Pulse No	FFA	FFB	FFC	FFD
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1

Ring Counter -3

- 2-bit quadrature generator



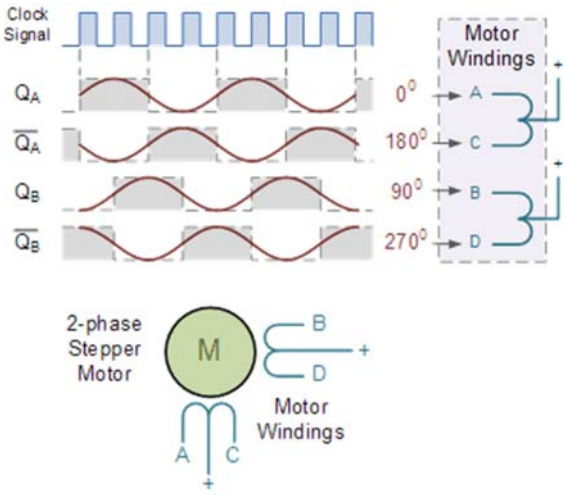


4-phase output for Bi-polar, Unipolar or Universal Motors

Output	A	B	C	D
$Q_A+\overline{Q}_B$	1	0	0	0
$\overline{Q}_A+Q_B$	0	1	0	0
Q_A+Q_B	0	0	1	0
$\overline{Q}_A+\overline{Q}_B$	0	0	0	1

2-bit Quadrature Oscillator, Count Sequence

- Stepper motor control



The End

- Questions?