Negative Num.

1's : 3 = 0011 , −3 = 1100. (+): end-around-carry

2's : 3 = 0011 , −3 = 1101 (+): ignore carry.

overflow detection : (+) + (+) = (−) / (−) + (−) = (+)

XOR

XNOR.

$X \oplus Y = XY' + X'Y.$

$X \equiv Y = X'Y' + XY.$

X Y	X ⊕ Y
0 0	0
0 1	1
1 0	1
1 1	0

$X \oplus 0 = X$

$X \oplus 1 = X'$

$X \oplus X = 0$

$X \oplus X' = 1$

$X \oplus Y = Y \oplus X$

$(X \oplus Y) \oplus Z = X \oplus (Y \oplus Z)$

$X(Y \oplus Z) = XY \oplus XZ$

$(X \oplus Y)' = X'Y' \oplus XY.$

8 basic forms for 2-level circuits.

- AND-OR ↔ NAND-NAND ↔ OR-NAND ↔ NOR-OR.
- OR-AND ↔ NOR-NOR ↔ AND-NOR ↔ NAND-AND.

Implicant : product term. (a circle).

Prime Implicant : maximal circle.

Essential Prime Implicant : if a minterm is covered by only one PI, the PI is essential.

AND. OR.

minterm 交集 聯集.

maxterm. 聯集. 交集.

base conversion.

minimum SOP : min # terms, min # literals

→ K-map 圈圈最少且最大.

(10) → (r)

$$
\begin{array}{r|l}
2 & 53 \\
2 & 26 \quad \cdots 1 \\
2 & 13 \quad \cdots 0 \\
2 & 6 \quad \cdots 1 \\
2 & 3 \quad \cdots 0 \\
2 & 1 \quad \cdots 1 \\
 & 0 \quad \cdots 1
\end{array}
$$

$53_{10} = 110101_2$

$0.75_{10} \rightarrow (7)$

$$
\begin{array}{r}
0.75 \\
\times \quad 7 \\
\hline
(5).25 \\
7 \\
\hline
(1).75 \\
7 \\
\hline
(5).25
\end{array}
$$

repeat.

$.75_{10} = .51_7.$

Simplification Theorems.

Uniting.

$XY + XY' = X.$

$(X+Y)(X+Y') = X.$

Absorption.

$X + XY = X.$

$X(X+Y) = X.$

Elimination.

$X + X'Y = X + Y.$

$X(X'+Y) = XY.$

Consensus.

$XY + X'Z + YZ = XY + X'Z$

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

$(X+Y)(X'+Z) = XZ + X'Y.$

4 logic values

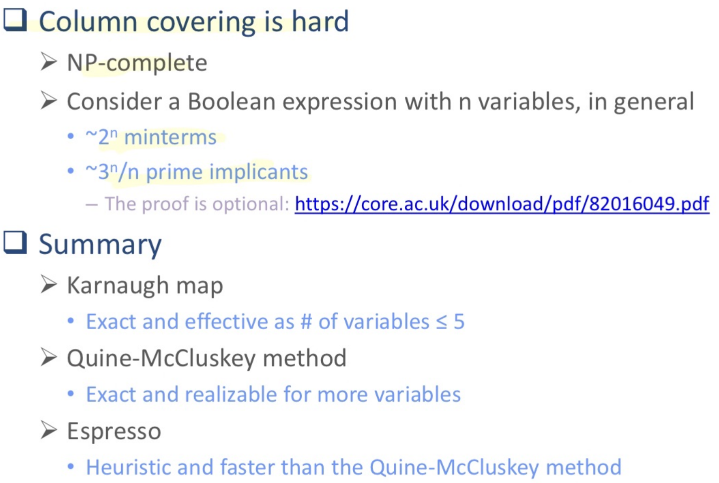
unknown. 沒接通.

AND	0	1	X	Z
0	0	0	0	0
1	0	1	X	X
X	0	X	X	X
Z	0	X	X	X

OR	0	1	X	Z
0	0	1	X	X
1	1	1	1	1
X	X	1	X	X
Z	X	1	X	X

$F = \sum m(3,4,5,6,7) = \prod M(0,1,2).$

$F' = \sum m(0,1,2) = \prod M(3,4,5,6,7)$



❑ Column covering is hard

- ➢ NP-complete
- ➢ Consider a Boolean expression with n variables, in general
  - ~2^n minterms
  - ~$3^n/n$ prime implicants
    - – The proof is optional: https://core.ac.uk/download/pdf/82016049.pdf

❑ Summary

- ➢ Karnaugh map
  - Exact and effective as # of variables ≤ 5
- ➢ Quine-McCluskey method
  - Exact and realizable for more variables
- ➢ Espresso
  - Heuristic and faster than the Quine-McCluskey method

Distributive Laws.

- $X(Y+Z) = XY + XZ.$
- $X + YZ = (X+Y)(X+Z).$

BCD : ex. 93.2 → 1001 0011. 0010

$XY = XZ \not\Rightarrow Y = Z.$

$X + Y = X + Z \not\Rightarrow Y = Z.$

$X \oplus Y = X \oplus Z \Rightarrow Y = Z.$

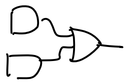
Functionally Complete Set.

✓ { AND. NOT }. { OR. NOT }. { NAND }. { NOR }.

x { 沒有 OR. 或 AND 的 }.

Gate level.

level 2 level 1



化簡 の laws.

Operations with 0/1

$X + 0 = X$

$X \cdot 1 = X$

$X + 1 = 1$

$X \cdot 0 = 0.$

Idempotent laws.

$X + X = X$

$X \cdot X = X.$

Laws of complementarity

$X + X' = 1.$

$X \cdot X' = 0.$

Distributive Laws.

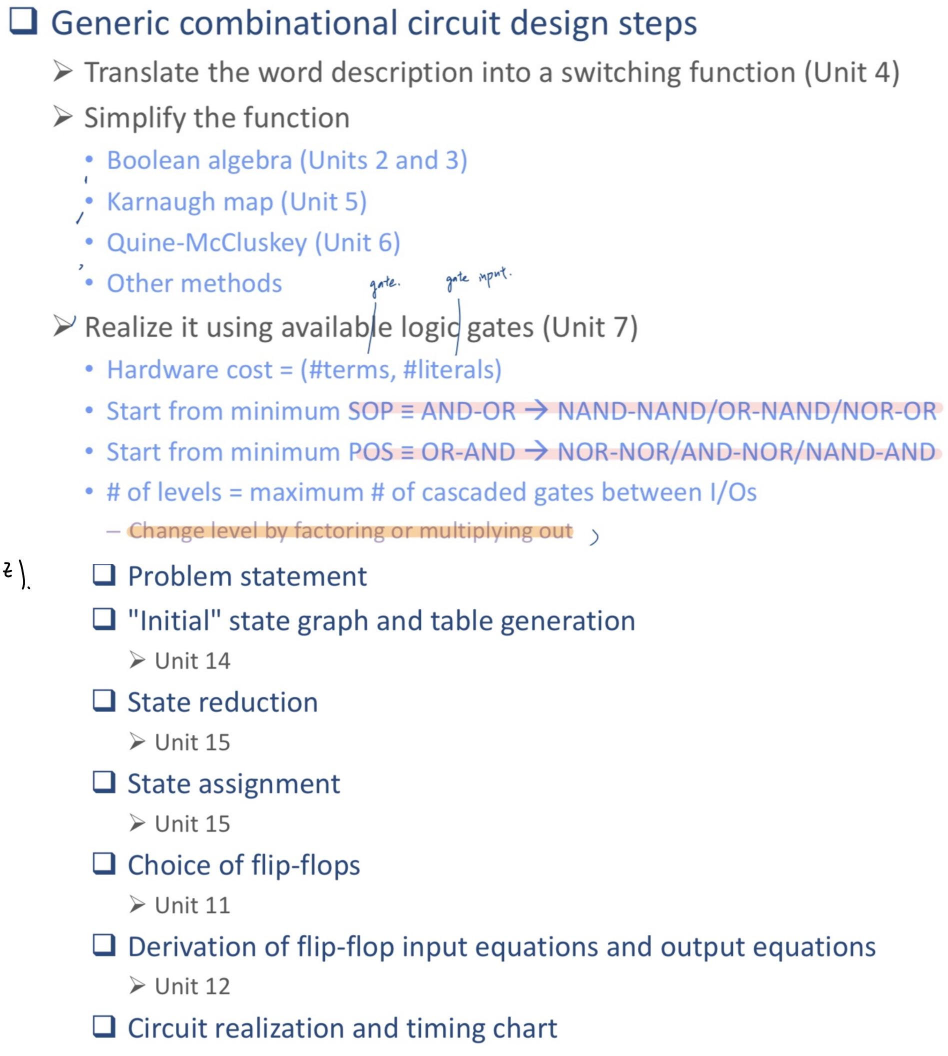
$X(Y+Z) = XY + XZ.$

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

DeMorgan's Laws.

$(X+Y)' = X'Y'$

$(XY)' = X' + Y'$



❑ Generic combinational circuit design steps

- ➢ Translate the word description into a switching function (Unit 4)
- ➢ Simplify the function
  - Boolean algebra (Units 2 and 3)
  - Karnaugh map (Unit 5)
  - Quine-McCluskey (Unit 6)
  - Other methods
- ➢ Realize it using available logic gates (Unit 7)

gate. gate input.

  - Hardware cost = (#terms, #literals)
  - Start from minimum SOP ≡ AND-OR → NAND-NAND/OR-NAND/NOR-OR
  - Start from minimum POS ≡ OR-AND → NOR-NOR/AND-NOR/NAND-AND
  - # of levels = maximum # of cascaded gates between I/Os
    - – ~~Change level by factoring or multiplying out~~

❑ Problem statement

❑ "Initial" state graph and table generation

- ➢ Unit 14

❑ State reduction

- ➢ Unit 15

❑ State assignment

- ➢ Unit 15

❑ Choice of flip-flops

- ➢ Unit 11

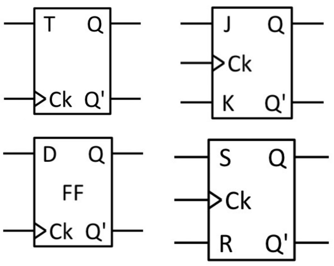
❑ Derivation of flip-flop input equations and output equations

- ➢ Unit 12

❑ Circuit realization and timing chart

Type of FF	Input	Q = 0		Q = 1		Rules for forming input map from next state map	
		$Q^+ = 0$	$Q^+ = 1$	$Q^+ = 0$	$Q^+ = 1$	Q = 0 Half of Map	Q = 1 Half of Map
D	D	0	1	0	1	No change	No change
T	T	0	1	1	0	No change	Complement
S-R	S	0	1	0	X	No change	Replace 1's with X's
	R	X	0	1	0	Replace 0's with X's Replace 1's with 0's	Complement
J-K	J	0	1	X	X	No change	Fill in with X's
	K	X	X	1	0	Fill in with X's	Complement

Type	Q^+
D Flip-Flop	D
S-R Flip-Flop	S + R'Q
J-K Flip-Flop	JQ' + K'Q
T Flip-Flop	TQ' + T'Q
D-CE Flip-Flop	D(CE) + Q(CE)'



Programmable Logic Arrays

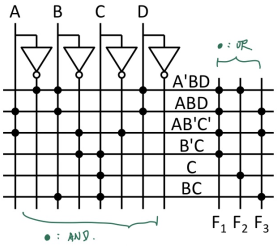
❑ Example

- ➢ $F_1(A,B,C,D) = \sum m(2,3,5,7,8,9,10,11,13,15)$
- ➢ $F_2(A,B,C,D) = \sum m(2,3,5,6,7,10,11,14,15)$
- ➢ $F_3(A,B,C,D) = \sum m(6,7,8,9,13,14,15)$

❑ Minimize using K-map

- ➢ F_1 = A'BD + ABD + AB'C' +B'C
- ➢ F_2 = C + A'BD
- ➢ F_3 = BC + AB'C' + ABD

	A	B	C	D	F_1	F_2	F_3
A'BD	0	1	–	1	1	1	0
ABD	1	1	–	1	1	0	1
AB'C'	1	0	0	–	1	0	1
B'C	–	0	1	–	1	0	0
C	–	–	1	–	0	1	1
BC	–	1	1	–	0	0	1



Mealy: Output depends on state + input.

Moore.. Output depends on state only.

Q	Q^+	D
0	0	0
0	1	1
1	0	0
1	1	1

D Flip-Flop

Q	Q^+	T
0	0	0
0	1	1
1	0	1
1	1	0

T Flip-Flop

Q	Q^+	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

S-R Flip-Flop

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

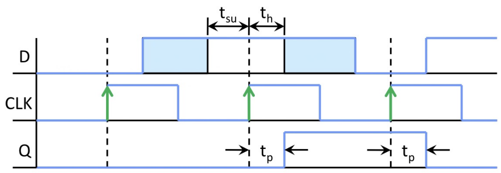
J-K Flip-Flop

❑ Setup time

- ➢ The amount of time that D must be stable before the active edge

❑ Hold time

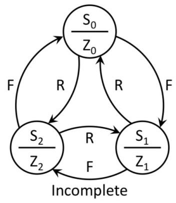
- ➢ The amount of time that D must hold the same value after the active edge



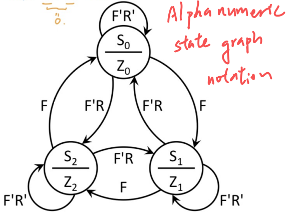
Completely Specified state graph.

- ➢ **OR** together all input labels on arcs emanating from a state, the result can reduce to 1
  - Cover all conditions: F + F'R +F'R' = F + F' = 1
- ➢ **AND** together any pair of input labels on arcs emanating from a state, the result can reduce to 0
  - Only one arc is valid: F·F'R = 0, F·F'R' = 0, F'R·F'R' = 0

0 0. 0.



Incomplete



Alphanumeric state graph notation

Equivalent States.

❑ Definition

- ➢ N_1, N_2: sequential circuits (not necessarily different)
- ➢ **X**: a sequence of inputs of arbitrary length
- ➢ Then, state p in N_1 ≡ state q in N_2 if and only if $\lambda_1(p,\mathbf{X}) = \lambda_2(q,\mathbf{X})$ for every possible input sequence X
  - λ: output
- ➢ Difficult to check the equivalence using this definition!
  - Infinite number of input sequences

❑ Theorem

- ➢ Two states p and q of a sequential circuit are equivalent if and only if for every single input X, the outputs are the same and the next states are equivalent, i.e., λ(p,X) = λ(q,X) and δ(p,X) ≡ δ(q,X)
  - δ: next state
  - Note that the next state do not have to be equal, just equivalent

Equivalent Sequential circuits

❑ Definition

- ➢ Two sequential circuits are equivalent: $N_1 \equiv N_2$ if
- ➢ For each state p in N_1, there is a state q in N_2 such that p ≡ q
- ➢ For each state s in N_2, there is a state t in N_1 such that s ≡ t

	A	B	C	D	E	F	G
B	~~D-F~~ ~~C-H~~						
C	X	X					
D	C-E	~~A-F~~ ~~E-H~~	X				
E	X	X	A-D	X			
F	X	X	~~E-F~~ ~~B-D~~	X	~~C-F~~ ~~A-B~~		
G	~~B-D~~ ~~C-H~~	~~B-F~~	X	~~A-B~~ ~~E-H~~	X	X	
H	X	X	~~C-E~~ ~~D-G~~	X	~~A-G~~	~~C-F~~ ~~B-G~~	X

	A	B	C	D
S_0	X	$C\text{-}S_3$ $D\text{-}S_1$	~~$A\text{-}S_3$~~ ~~$C\text{-}S_1$~~	X
S_1	~~$B\text{-}S_3$~~ ~~$A\text{-}S_0$~~	X	X	$C\text{-}S_3$ $B\text{-}S_0$
S_2	$B\text{-}S_0$ ~~$A\text{-}S_2$~~	X	X	~~$C\text{-}S_0$~~ ~~$B\text{-}S_2$~~
S_3	X	~~$D\text{-}S_2$~~ ~~$D\text{-}S_3$~~	$A\text{-}S_2$ ~~$C\text{-}S_3$~~	X

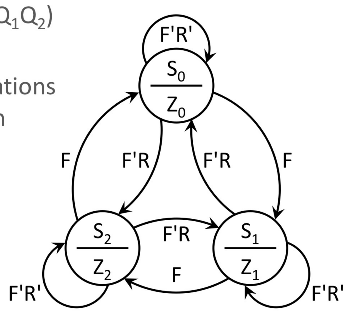
⇨ $A \equiv S_2$, $B \equiv S_0$, $C \equiv S_3$, $D \equiv S_1$ ⇨ $N_1 \equiv N_2$

❑ One-hot state assignment: One flip-flop for each state

- ➢ Example: 3 flip-flops for 3 states ($Q_0Q_1Q_2$)
  - S_0 = 100, S_1 = 010, S_2 = 001
- ➢ Write next-state and output (Z) equations directly by inspecting the state graph

從Q_0來 從Q_1來 從Q_2來.

  - $Q_0^+ = \underline{F'R'Q_0} + \underline{F'RQ_1} + \underline{FQ_2}$
  - $Q_1^+ = F'R'Q_1 + F'RQ_2 + FQ_0$
  - $Q_2^+ = F'R'Q_2 + F'RQ_0 + FQ_1$
  - $Z = Z_0Q_0 + Z_1Q_1 + Z_2Q_2$



Summary.

❑ First Half

- ➢ [1] Number Systems and Conversion
- ➢ [2] Boolean Algebra
- ➢ [3] Boolean Algebra (Continued)
- ➢ [4] Applications of Boolean Algebra
- ➢ [5] Karnaugh Maps
- ➢ [6] Quine-McCluskey Method
- ➢ [7] Multi-Level Gate Circuits
- ➢ [8] Combinational Circuit Design

❑ Second Half

- ➢ [9] Multiplexers, Decoders, and Programmable Logic Devices
- ➢ [11] Latches and Flip-Flops
- ➢ [12] Registers and Counters
- ➢ [13] Analysis of Clocked Sequential Circuits
- ➢ [14] Derivation of State Graphs and Tables
- ➢ [15] Reduction of State Tables
- ➢ [16] Sequential Circuit Design

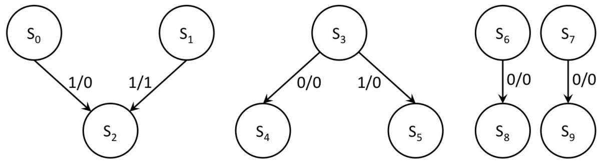
State Assignment

❑ Guidelines

- ➢ No guarantee a minimum solution
- ➢ Work for D & J-K flip-flops, not for T and S-R flip-flops

❑ Adjacent assignments

- ➢ For a given input, states with the same next state (S_0 and S_1)
- ➢ States which are next states of the same state (S_4 and S_5)
- ➢ States which have the same output (S_6 and S_7)
  - Place 1's together on the output maps


S_0
S_1
1/0
1/1
S_2
S_3
0/0
1/0
S_4
S_5
S_6
S_7
0/0
0/0
S_8
S_9