Introduction to CMOS VLSI Design

Lecture 1: Circuits & Layout

Outline

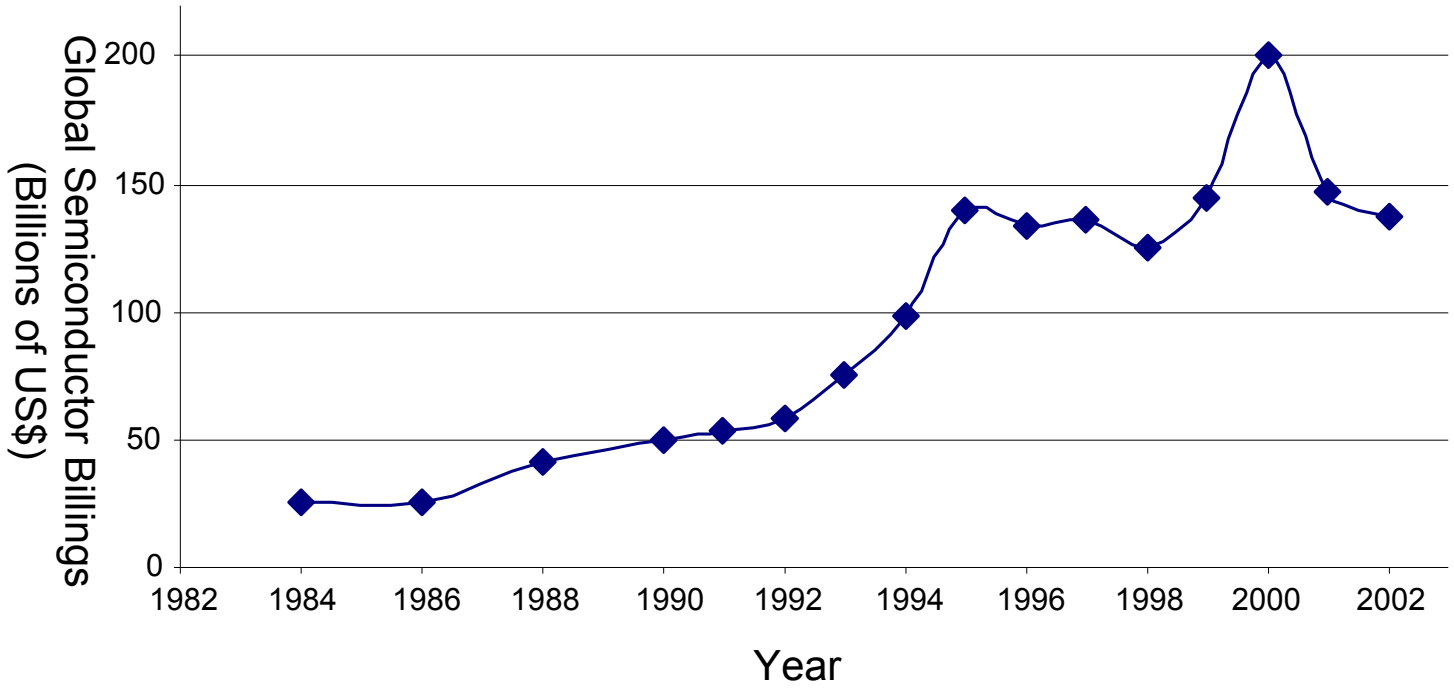
- A Brief History
- CMOS Gate Design
- Pass Transistors
- CMOS Latches & Flip-Flops
- Standard Cell Layouts
- Stick Diagrams

A Brief History

- 1958: First integrated circuit
  - Flip-flop using two transistors
  - Built by Jack Kilby at Texas Instruments
- 2003
  - Intel Pentium 4 μprocessor (55 million transistors)
  - 512 Mbit DRAM (> 0.5 billion transistors)
- 53% compound annual growth rate over 45 years
  - No other technology has grown so fast so long
- Driven by miniaturization of transistors
  - Smaller is cheaper, faster, lower in power!
  - Revolutionary effects on society

Annual Sales

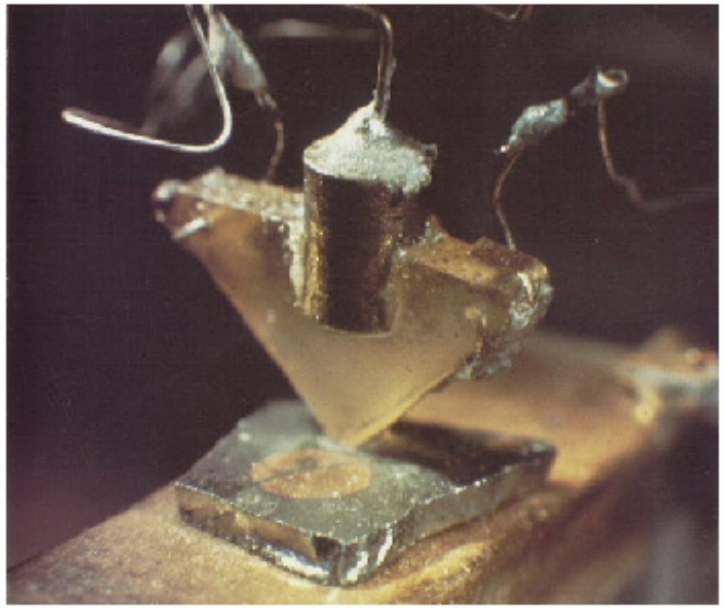
- 10^{18} transistors manufactured in 2003
  - 100 million for every human on the planet



Invention of the Transistor

電晶體 原本是 真空管。但太耗電

- Vacuum tubes ruled in first half of 20th century
  Large, expensive, power-hungry, unreliable
- 1947: first point contact transistor
  - John Bardeen and Walter Brattain at Bell Labs
  - Read *Crystal Fire*
    by Riordan, Hoddeson



Transistor Types

兩極 的 電晶體 二極體

是用电流

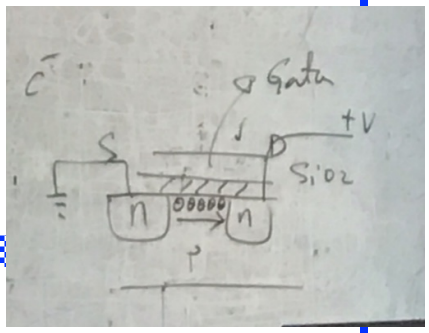
- Bipolar transistors
  - npn or pnp silicon structure
  - Small current into very thin base layer controls large currents between emitter and collector
  - Base currents limit integration density
- Metal Oxide Semiconductor Field Effect Transistors
  - nMOS and pMOS MOSFETS

是用 电場 感应

  - Voltage applied to insulated gate controls current between source and drain

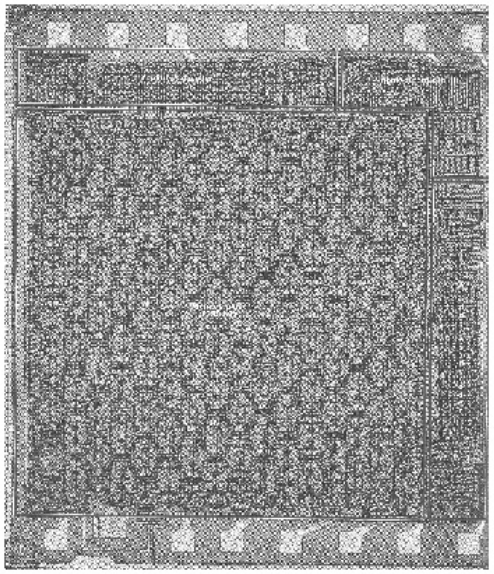
可以做到很小
也就可以低功率

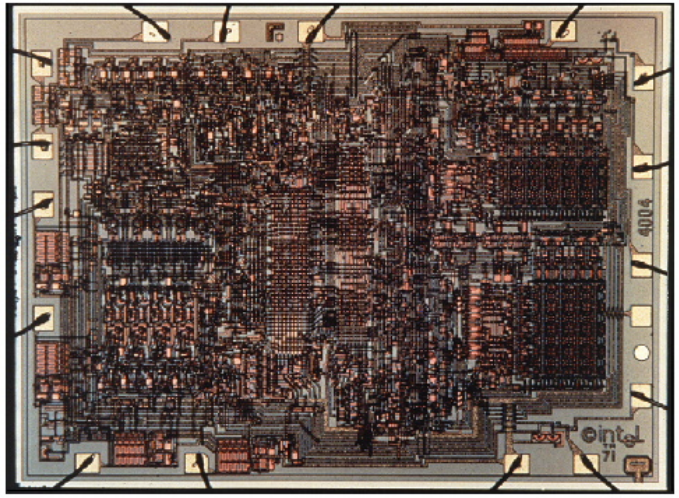
  - Low power allows very high integration



MOS Integrated Circuits

- 1970's processes usually had only nMOS transistors
  - Inexpensive, but consume power while idle



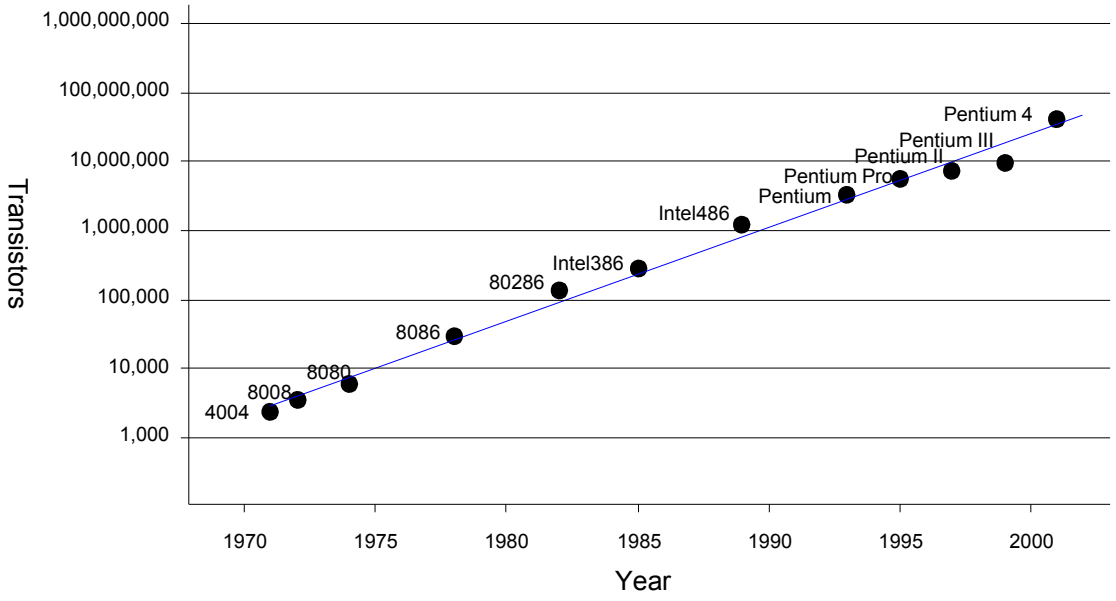


Intel 1101 256-bit SRAM Intel 4004 4-bit μProc

- 1980s-present: CMOS processes for low idle power

Moore's Law

- 1965: Gordon Moore plotted transistor on each chip
  - Fit straight line on semilog scale
  - Transistor counts have doubled every 26 months



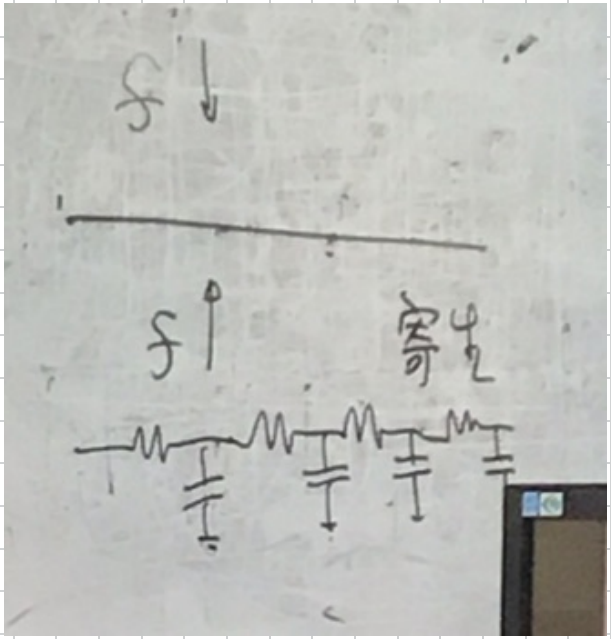
Integration Levels

SSI: 10 gates

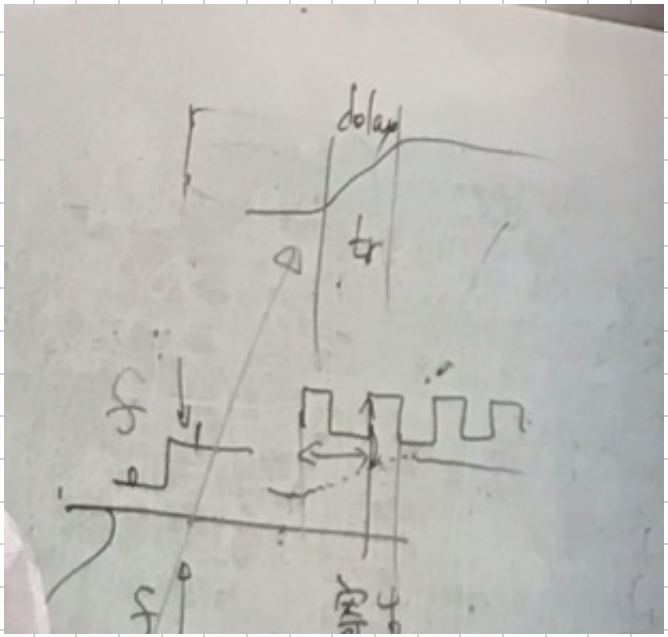
MSI: 1000 gates

LSI: 10,000 gates

VLSI: > 10k gates



導線 在頻率高的時候
就可以看成一串 電阻 和 電容

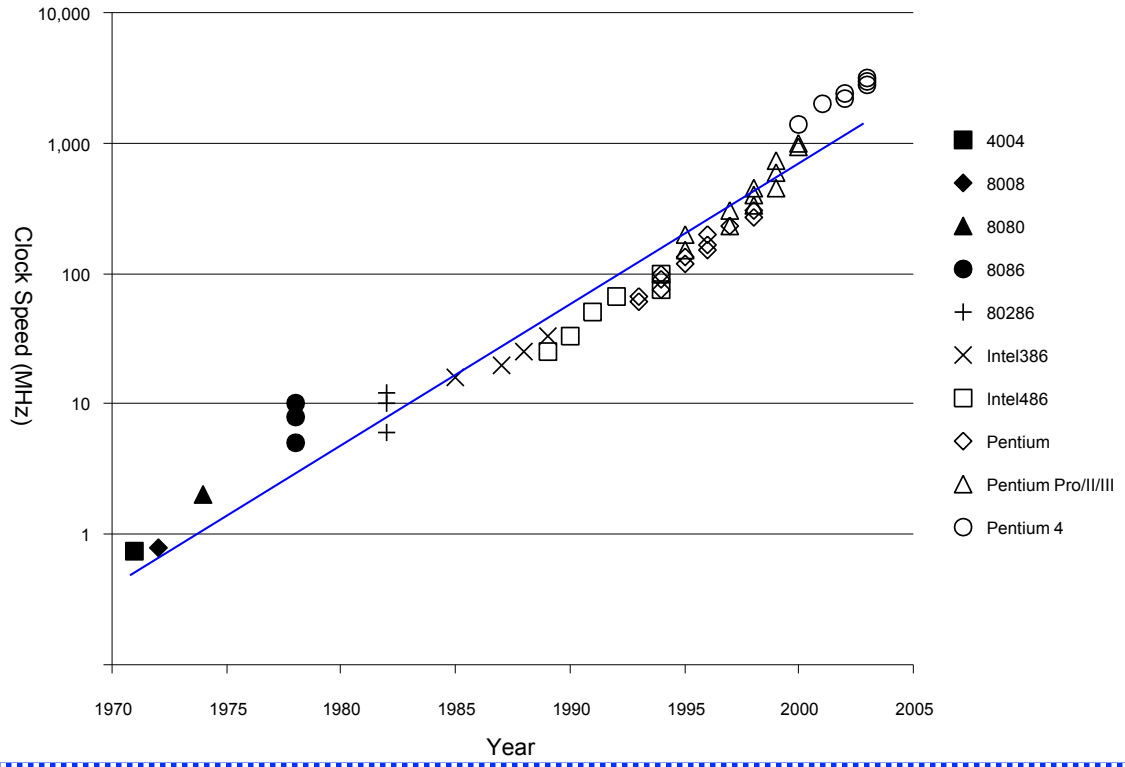


在快速電路 就會有可能
不夠
這個電壓 上升的時間

所以才會希望電路越短越好
才能越做越快

Corollaries

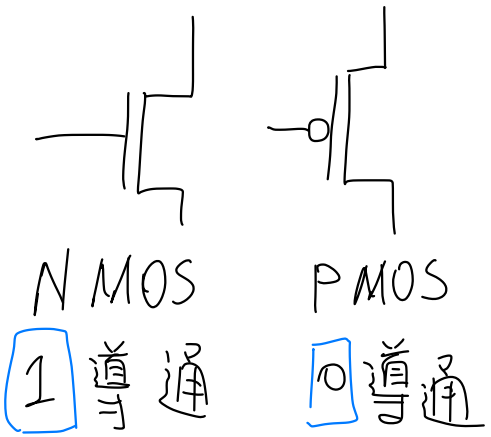
- Many other factors grow exponentially
  - Ex: clock frequency, processor performance



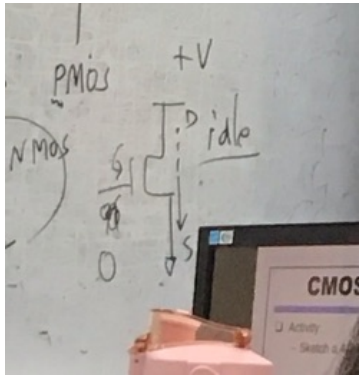
CMOS Gate Design

- Activity:
  - Sketch a 4-input CMOS NAND gate

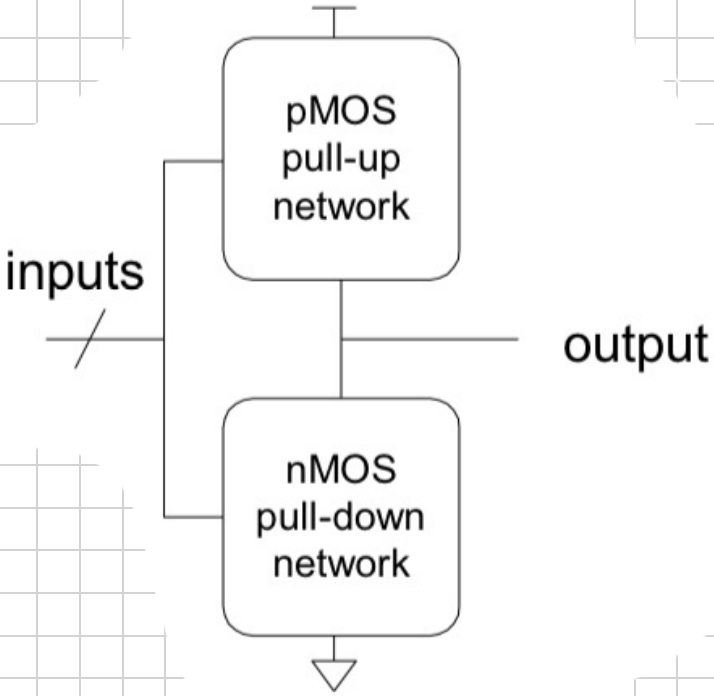
NMOSFET 就簡稱叫 nMOS
pMOS同理



pMOS 長得通常比較大



閒置的時候
也有可能有閒置電流
可能放著沒做事
也會有點熱熱的



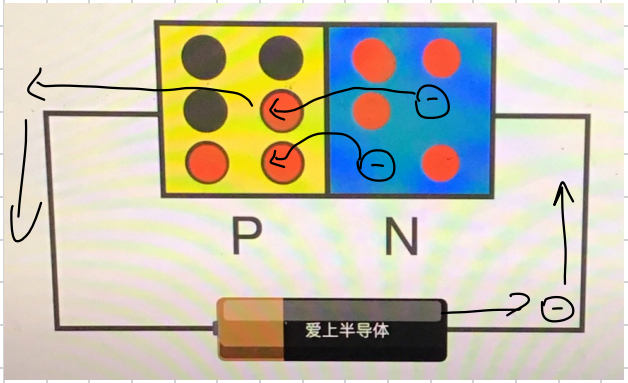
CMOS 主要目的 就是要省電
閒置的時候 不會一路導電到最下面 只會往右

只有在切換輸入的時候 會有一點點的導通

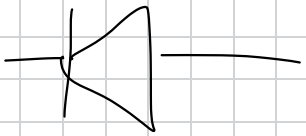
CMOS也是現在主要的製程

3價 金屬
形成 電洞

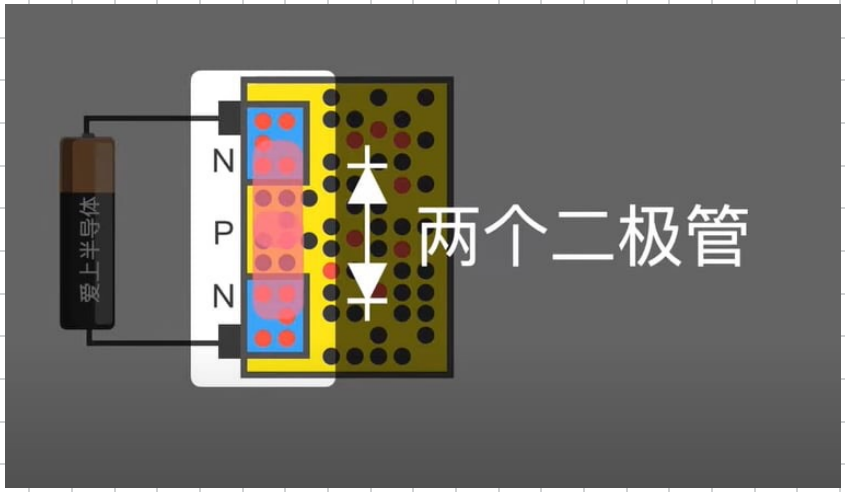
5價金屬
電子可以游離



2極体



這樣反向會走不了



這樣原本是都不能動的

剩下自己看

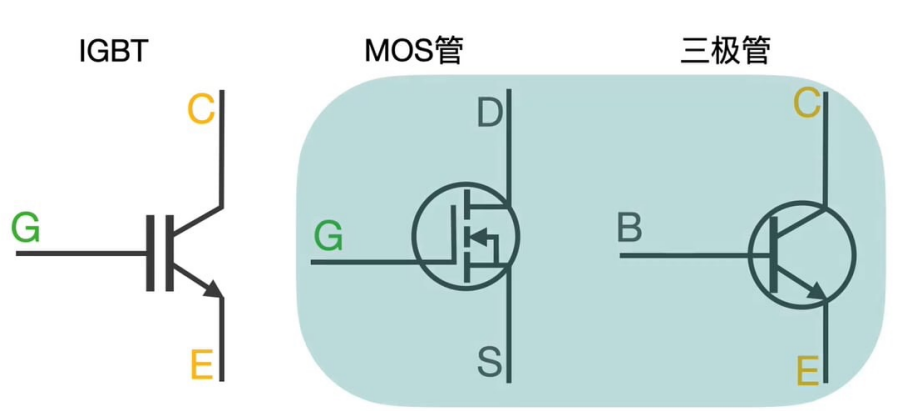
https://youtu.be/HtE6inJg2XQ

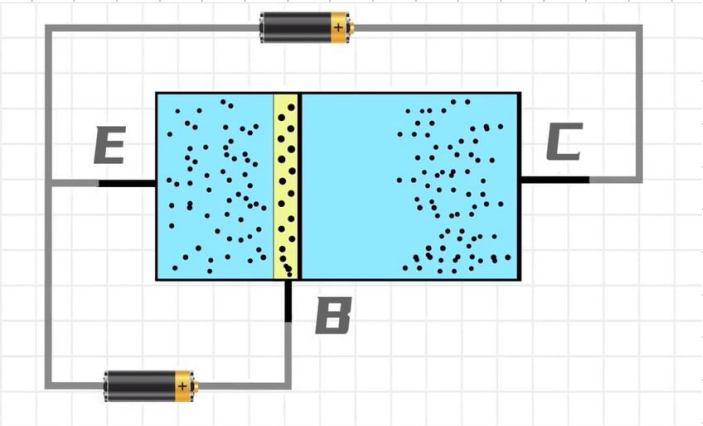
https://youtu.be/5BGM9zS32F0

這三個是不一樣的東西

IGBT 有 這兩個的優勢

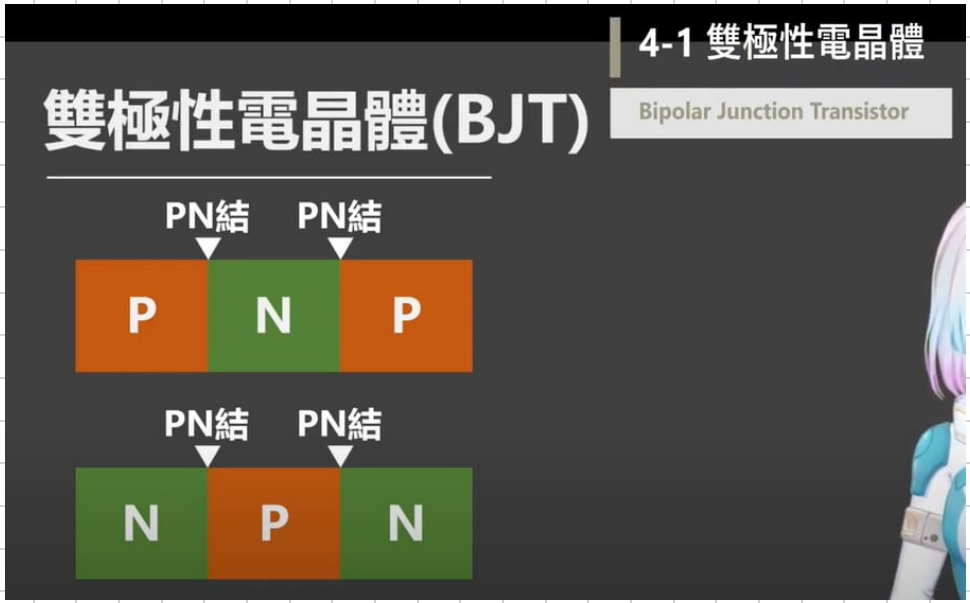
MOS 低驅動電流
三極管 低導通電阻





這是三極管 要先用下面的 把這個裝置給驅動
但是直接用上面的 直接硬通過去 他有可能嗎？

他是說 三極體還有放大的功能
Ic = Ie + Ib



BJT 就是 這三個組在一起的名字
又叫作 三極體

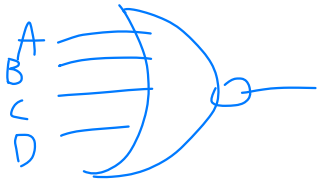
https://youtu.be/sVxlYAzKjxY
這是網路上人在教的

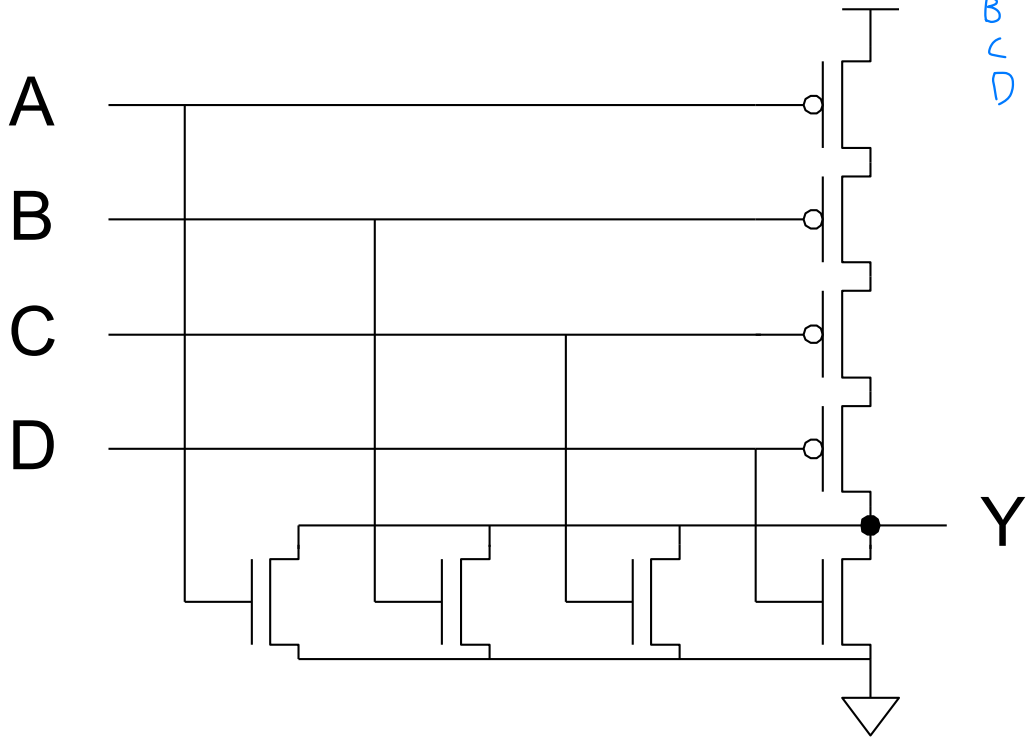
CMOS Gate Design

- Activity:

這個期中一定會考 給你一個邏輯閘 叫你畫出電路

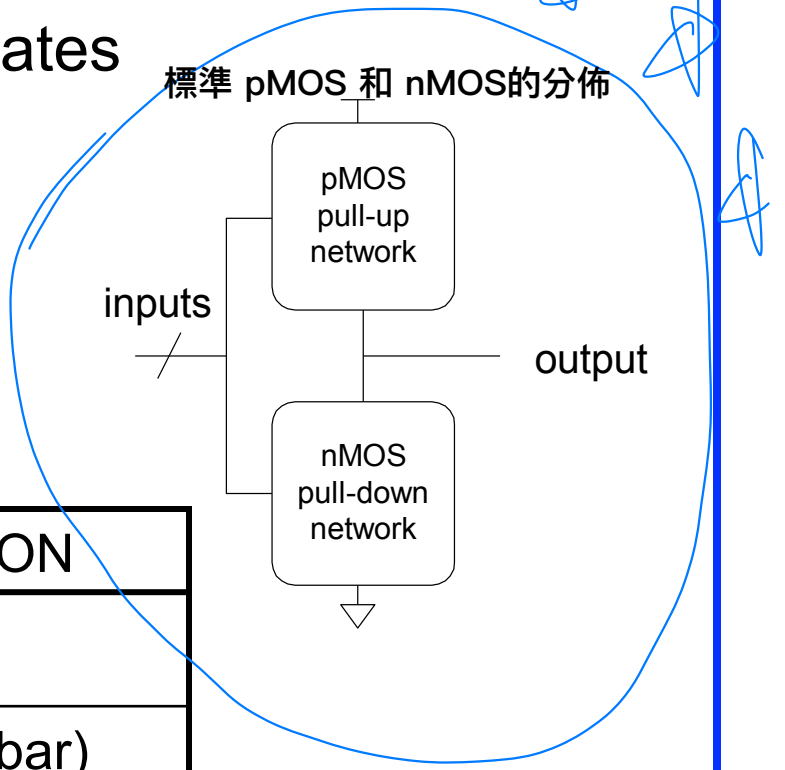
  - Sketch a 4-input CMOS NOR gate





Complementary CMOS

- Complementary CMOS logic gates
  - nMOS *pull-down network*
  - pMOS *pull-up network*
  - a.k.a. static CMOS



	Pull-up OFF	Pull-up ON
Pull-down OFF	Z (float)	1
Pull-down ON	0	X (crowbar)

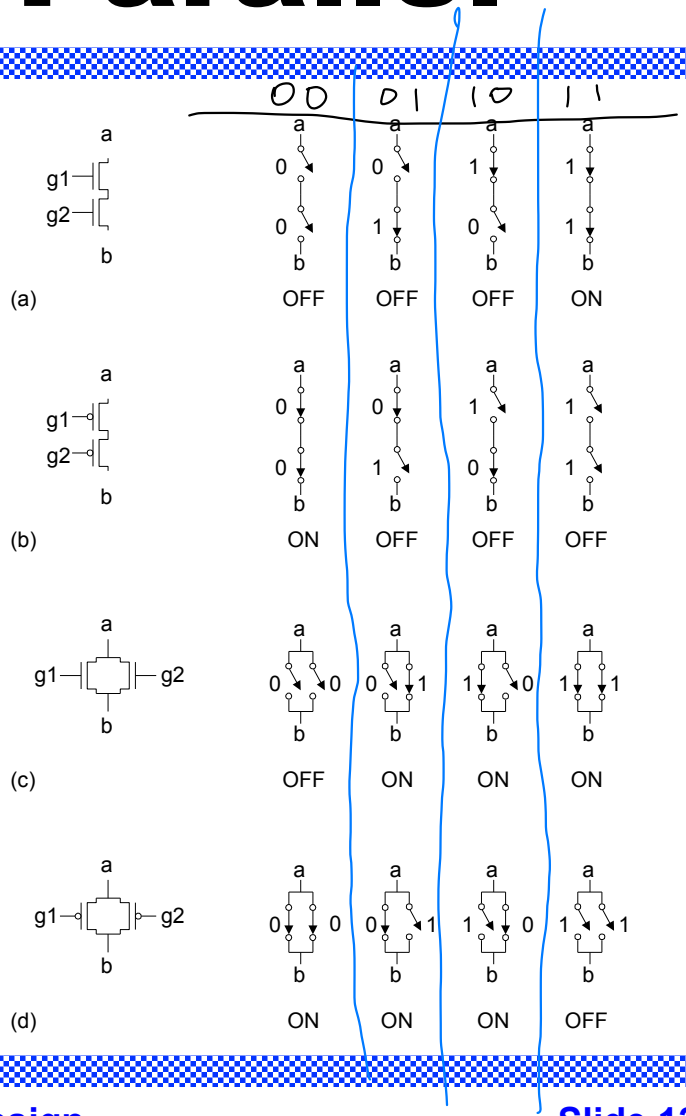
兩個pMOS nMOS
同時都是關掉的 就是

Float 浮接
正常電路最好是不要有 浮接
他會向天線一樣 有頻率的整數倍 就會共振

兩個同時都是開的
就是壞掉的

Series and Parallel

- nMOS: 1 = ON
- pMOS: 0 = ON
- *Series*: both must be ON
- *Parallel*: either can be ON

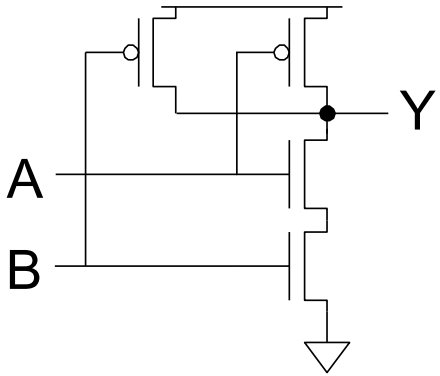


Conduction Complement

- Complementary CMOS gates always produce 0 or 1
- Ex: NAND gate

NAND NOR 兩個相反 電路接的方法剛好
並連串連的 pMOS nMOS剛好相反

  - Series nMOS: Y=0 when both inputs are 1
  - Thus Y=1 when either input is 0
  - Requires parallel pMOS

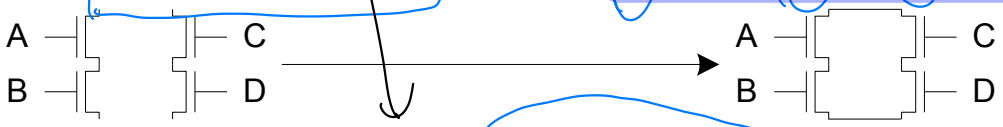


- Rule of *Conduction Complements*
  - Pull-up network is complement of pull-down
  - Parallel -> series, series -> parallel

Compound Gates

複合 邏輯閘

- *Compound gates* can do any inverting function 兩個兩個輸入
- Ex: $Y = \overline{A \cdot B + C \cdot D}$ (AND-AND-OR-INVERT, AOI22)



(a)

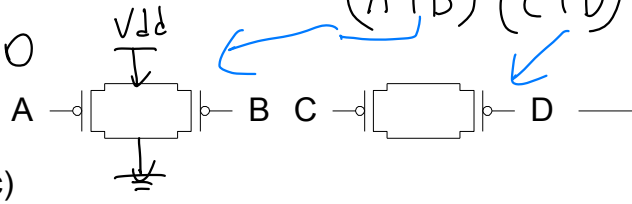
(b)

$\overline{A \cdot B + C \cdot D}$

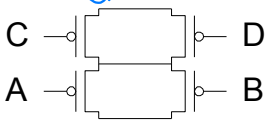
∵ A 或 B either 0

就是 Y=1

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



(c)



(d)

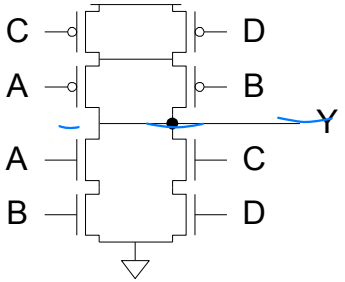
複習一下 卡諾圖

K map ?

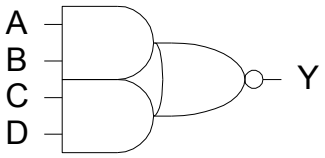
用來 精簡 電路

pMOS

nMOS



(e)



(f)

有invert
pMOS 就負責處理 Y = 1 正輸出 的狀況
nMOS 就負責處理 Y = 0 負輸出 的狀況
是這樣嗎

反正通常pMOS 都畫在上面 負責和Vdd 接著
nMOS 通常都在下面 負責和 接地 接著

這pMOS nMOS 的接法 就是剛好
並連 串連
相反

用成 ② ① POS $(\overline{A}+B)\cdot(C+\overline{D})$
sum
更好畫 pMOS 並連 串連

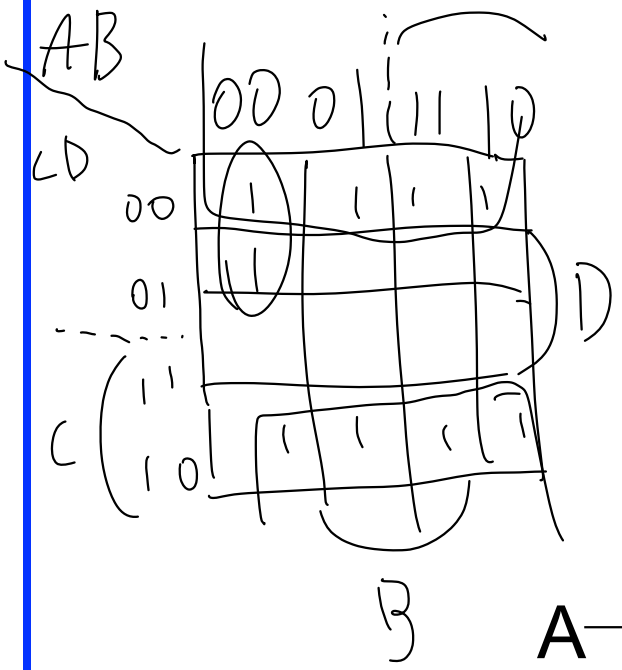
Example: O3AI

- $Y = \overline{\left(A + B + C\right) \bullet D}$

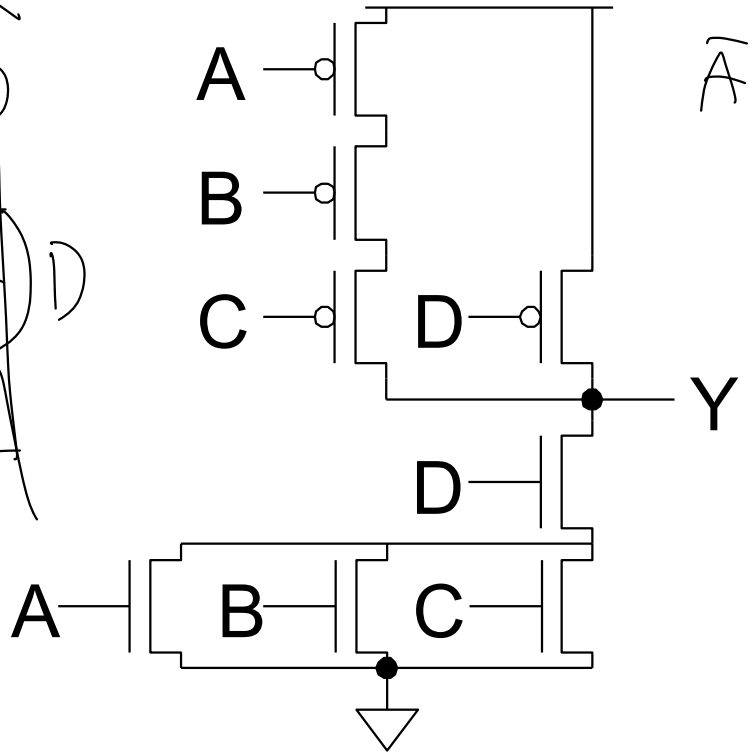
Example: O3AI

- $Y = \overline{(A + B + C) \cdot D}$

POS → SOP 再用 K map

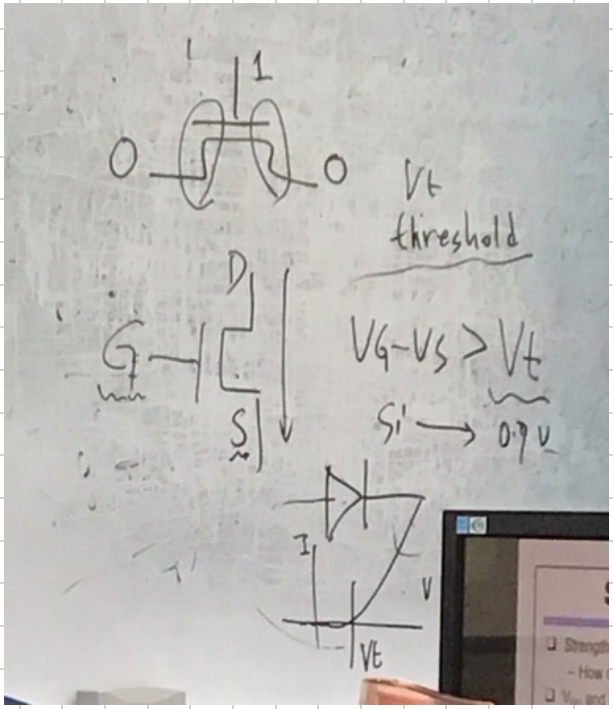


$\bar{A}\bar{B}\bar{C} + \bar{D}$

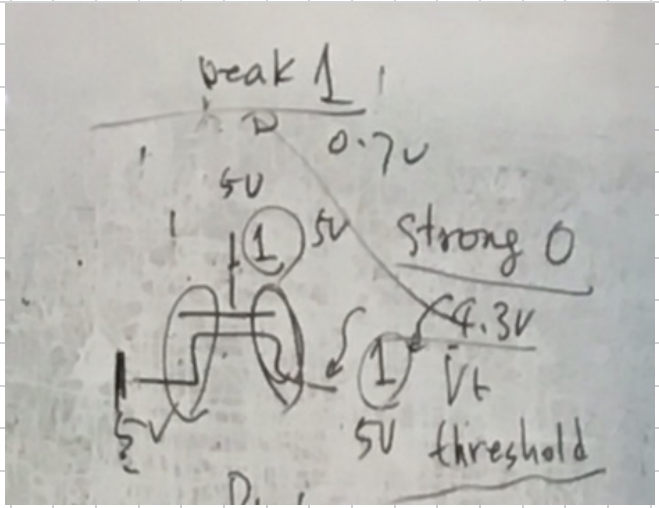


Signal Strength

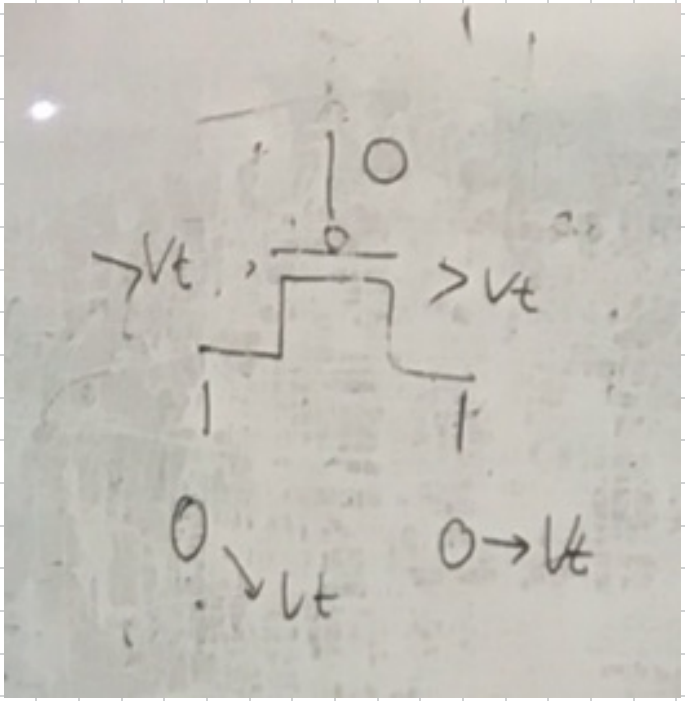
- *Strength* of signal
  - How close it approximates ideal voltage source
- V_{DD} and GND rails are strongest 1 and 0
- nMOS pass strong 0
  - But degraded or weak 1
- pMOS pass strong 1
  - But degraded or weak 0
- Thus nMOS are best for pull-down network



這個nMOS 要能夠導通 也要必須大於 threshold Vt
nMOS 可以讓 0 完整的通過 叫做 strong 0



上面這個5V 就像是要幫下面要通過的地方 拉開一個 通道
他必須要拉超過 Vt的高度 左邊的電流 才有可能可以留過去
但是留過去的最大量 不會超過 5V – 0.7 =4.3V
所以才會叫做 weak 1

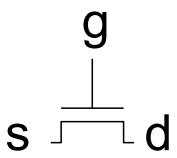


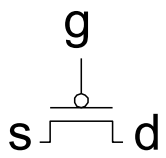
pMOS 0導通 比較奇特一點
他要通過也需要超過Vt
也就導致 0要通過 也會被拉高成 到Vt的強度
但他是strong 1 不太懂

所以 pMOS 才會通常用來負責 Y = 1
nMOS 才會通常用來負責 Y = 0

Pass Transistors

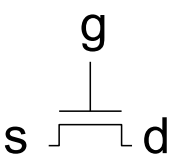
- Transistors can be used as switches

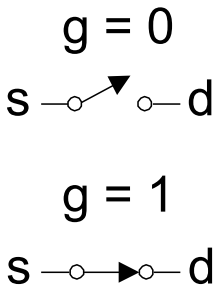


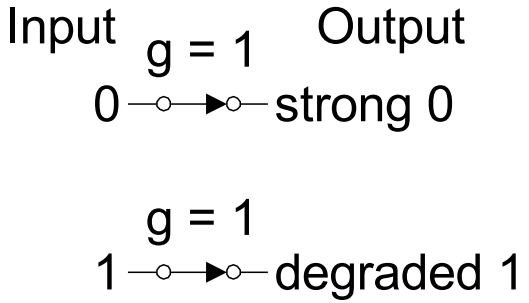


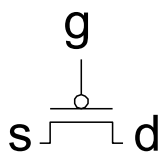
Pass Transistors

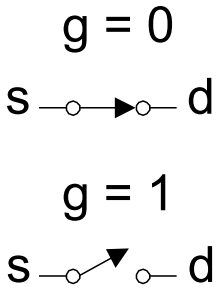
- Transistors can be used as switches

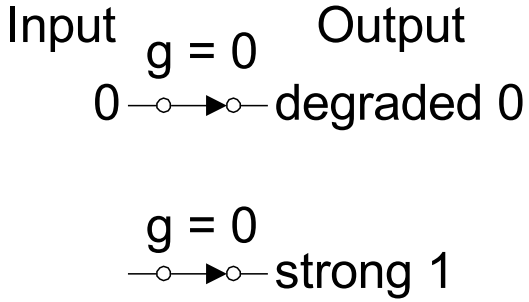








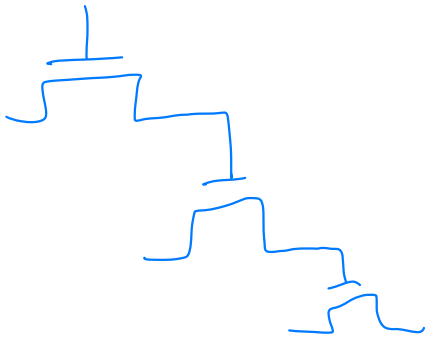




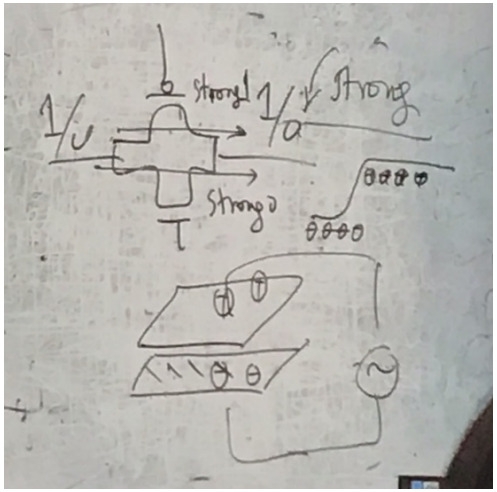
Transmission Gates

- Pass transistors produce degraded outputs
- *Transmission gates* pass both 0 and 1 well

但是他這樣 要串連 就有可能一直要減 Vt



所以就會把它弄成這樣子 並連
讓訊號不要衰減

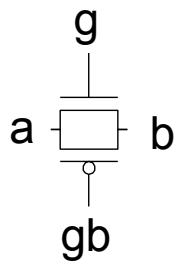


這是一個電容？

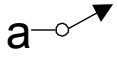
但弄成這樣 雜訊也會很容易通過

Transmission Gates

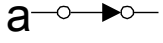
- Pass transistors produce degraded outputs
- *Transmission gates* pass both 0 and 1 well



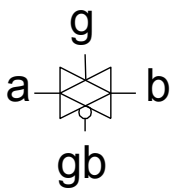
g = 0, gb = 1

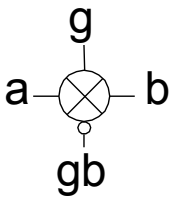


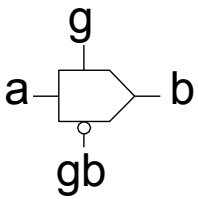
g = 1, gb = 0



Input	Output
g = 1, gb = 0 0	strong 0
g = 1, gb = 0 1	strong 1







Tristates

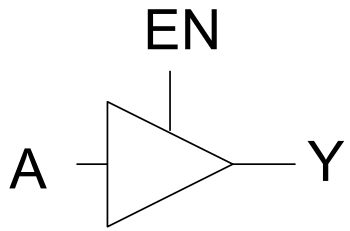
這是一個 緩衝器
不然電線也會消耗一些
就是中繼器

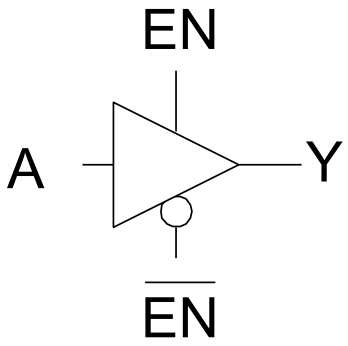
- *Tristate buffer* produces Z when not enabled

EN	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1

高阻態

High enbedays

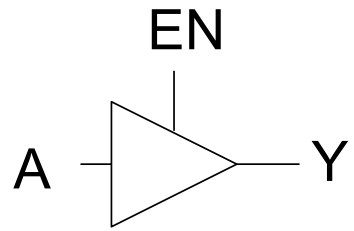


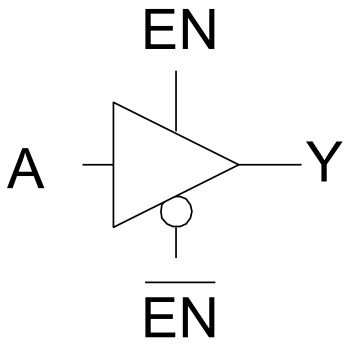


Tristates

- *Tristate buffer* produces Z when not enabled

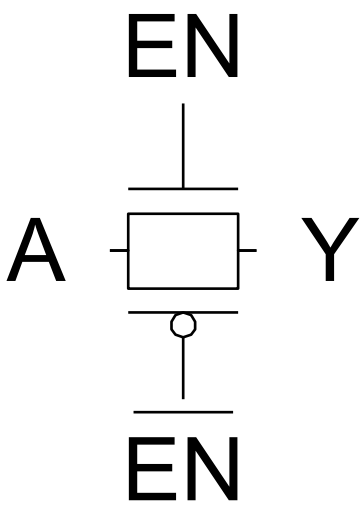
EN	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1





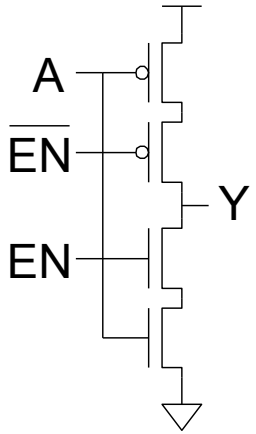
Nonrestoring Tristate

- Transmission gate acts as tristate buffer
  - Only two transistors
  - But *nonrestoring* 沒有辦法
    - Noise on A is passed on to Y



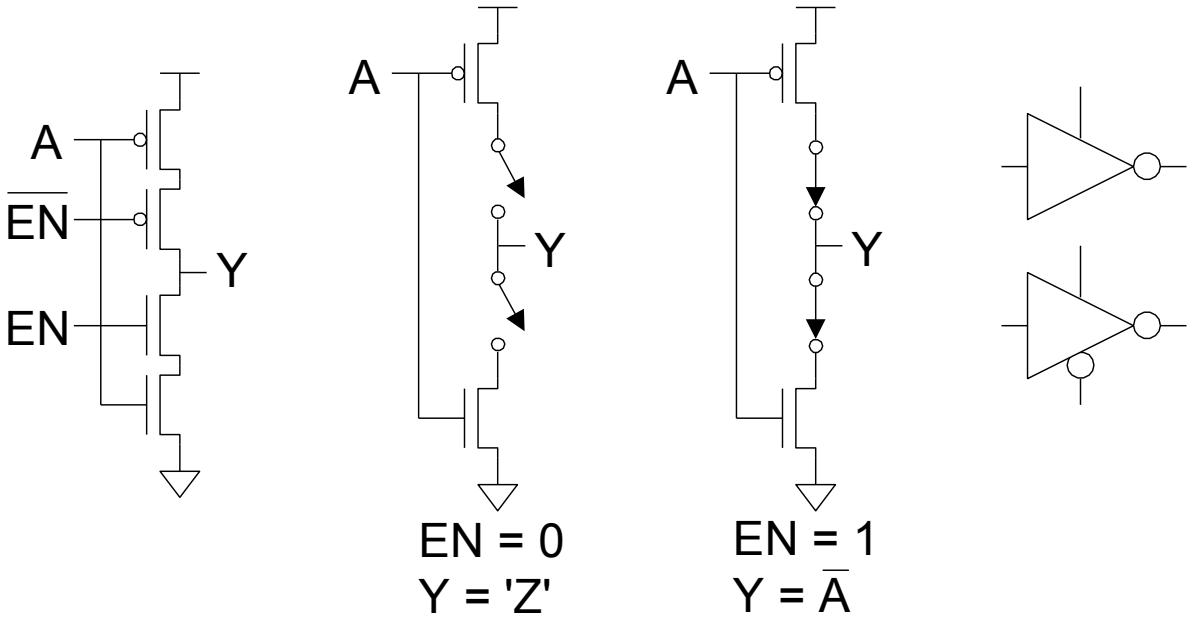
Tristate Inverter

- Tristate inverter produces restored output
  - Violates conduction complement rule
  - Because we want a Z output



Tristate Inverter

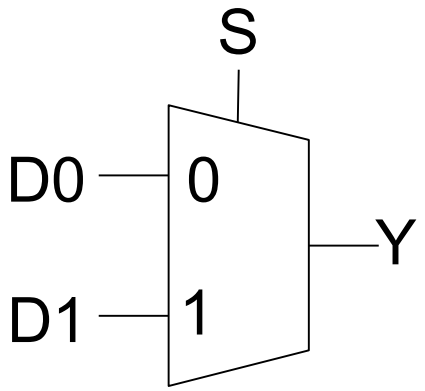
- Tristate inverter produces restored output
  - Violates conduction complement rule
  - Because we want a Z output



多工器 Multiplexers

- 2:1 *multiplexer* chooses between two inputs

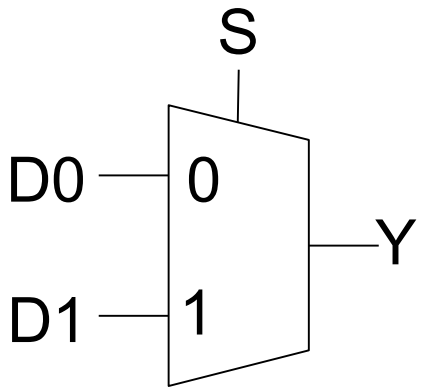
S	D1	D0	Y
0	X	0	
0	X	1	
1	0	X	
1	1	X	



Multiplexers

- 2:1 multiplexer chooses between two inputs

S	D1	D0	Y
0	X	0	0
0	X	1	1
1	0	X	0
1	1	X	1



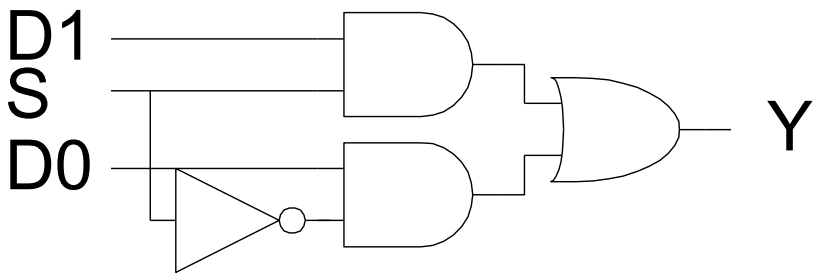
Gate-Level Mux Design

- $Y = SD_1 + \overline{S}D_0$ (too many transistors)
- How many transistors are needed?

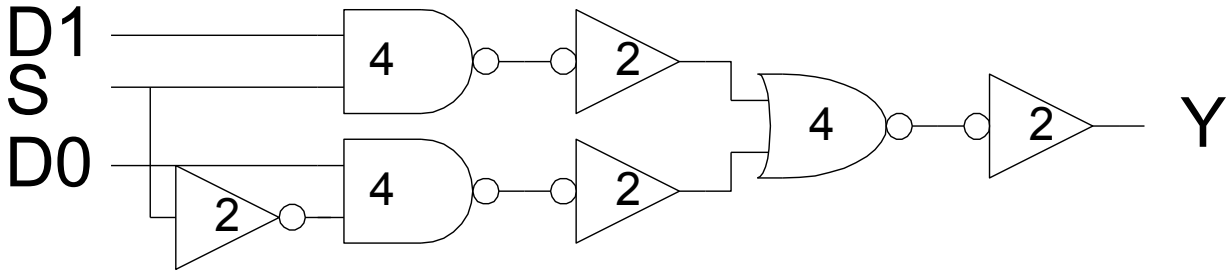
Gate-Level Mux Design

- $Y = SD_1 + \overline{S}D_0$ (too many transistors)
- How many transistors are needed? 20

這樣會需要太多電晶體



但不知道為什麼要加這麼多not



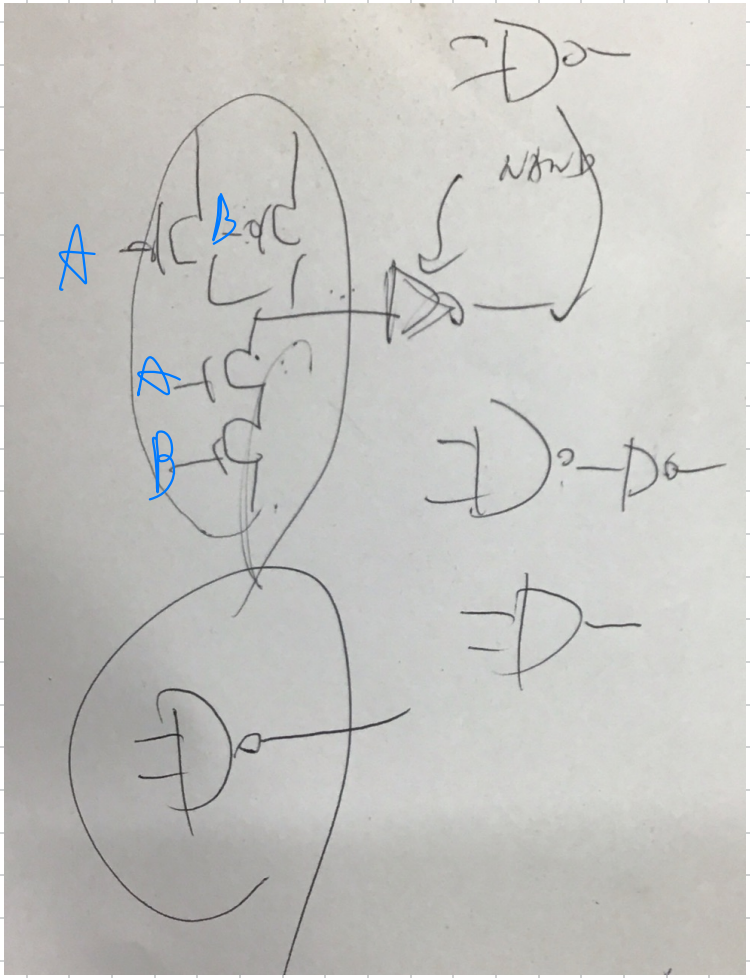
因為要組成邏輯閘
就是要用這些電路組成
這些電路 組成nand nor 更方便
要組成and 就是要再加上 not

NAND = $\overline{AB}$ = $\overline{A}+\overline{B}$

0	0	1
0	1	1
1	0	1
1	1	0

(SOP 畫 PMOS)

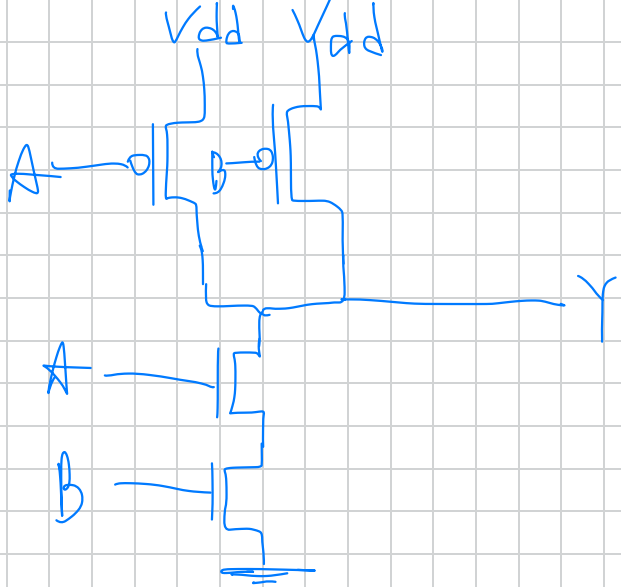
(POS 畫 nMOS)



NAND = $\overline{A \cdot B}$ = $\bar{A} + \bar{B}$ pMOS

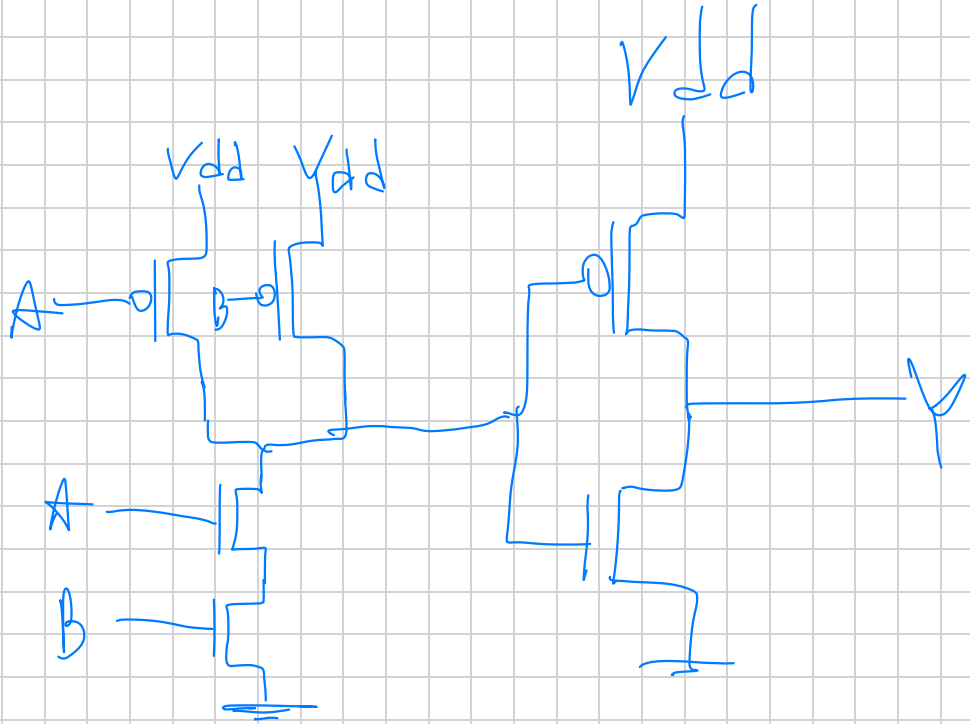
0 0	1
0 1	1
1 0	1
1 1	0

SOP → nMOS



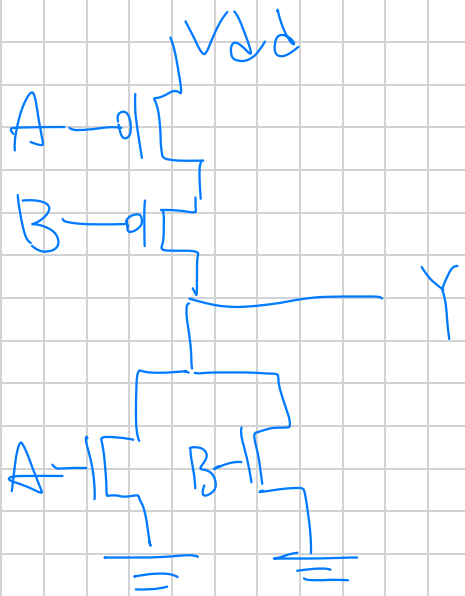
AND = $\overline{(\overline{A \cdot B})} = \overline{(\bar{A} + \bar{B})}$

0 0	0
0 1	0
1 0	0
1 1	1



NOR $\Rightarrow \overline{A+B} = \overline{A} \cdot \overline{B}$

0 0	1	POS
0 1	0	
1 0	0	SOP
1 1	0	



OR = $A+B = \overline{\overline{A+B}}$

0	$= \overline{(\overline{A} \cdot \overline{B})}$
1	
1	
1	

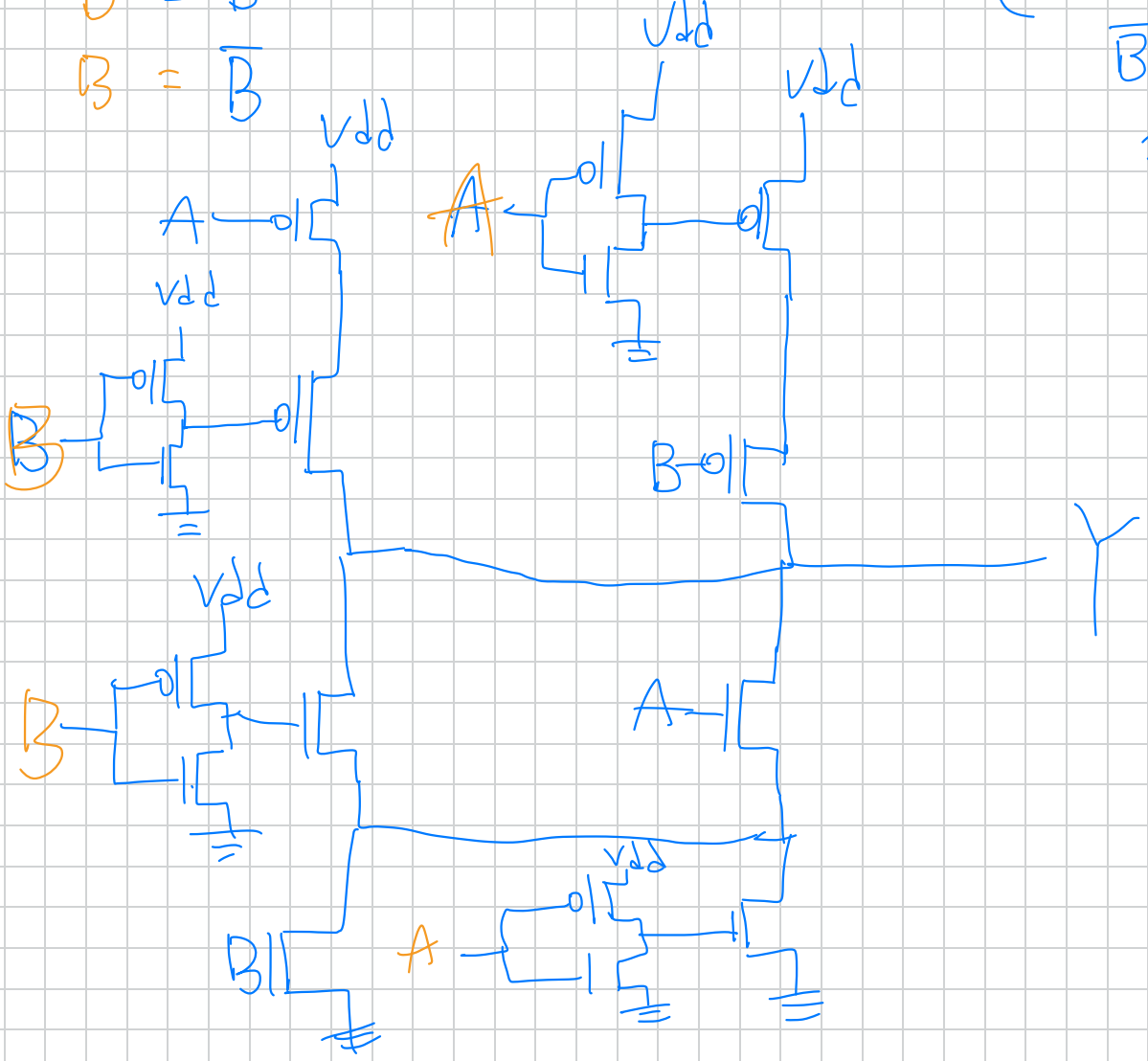
XOR = $A \oplus B = (\bar{A} \cdot B + A\bar{B})$ ⇒ $\bar{A}\bar{\bar{B}} + \bar{\bar{A}}\bar{B}$ → PDN

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

$\bar{B} = \bar{\bar{B}}$

$B = \bar{\bar{B}}$

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



XNOR = $\bar{A} \cdot B + A \cdot \bar{B}$

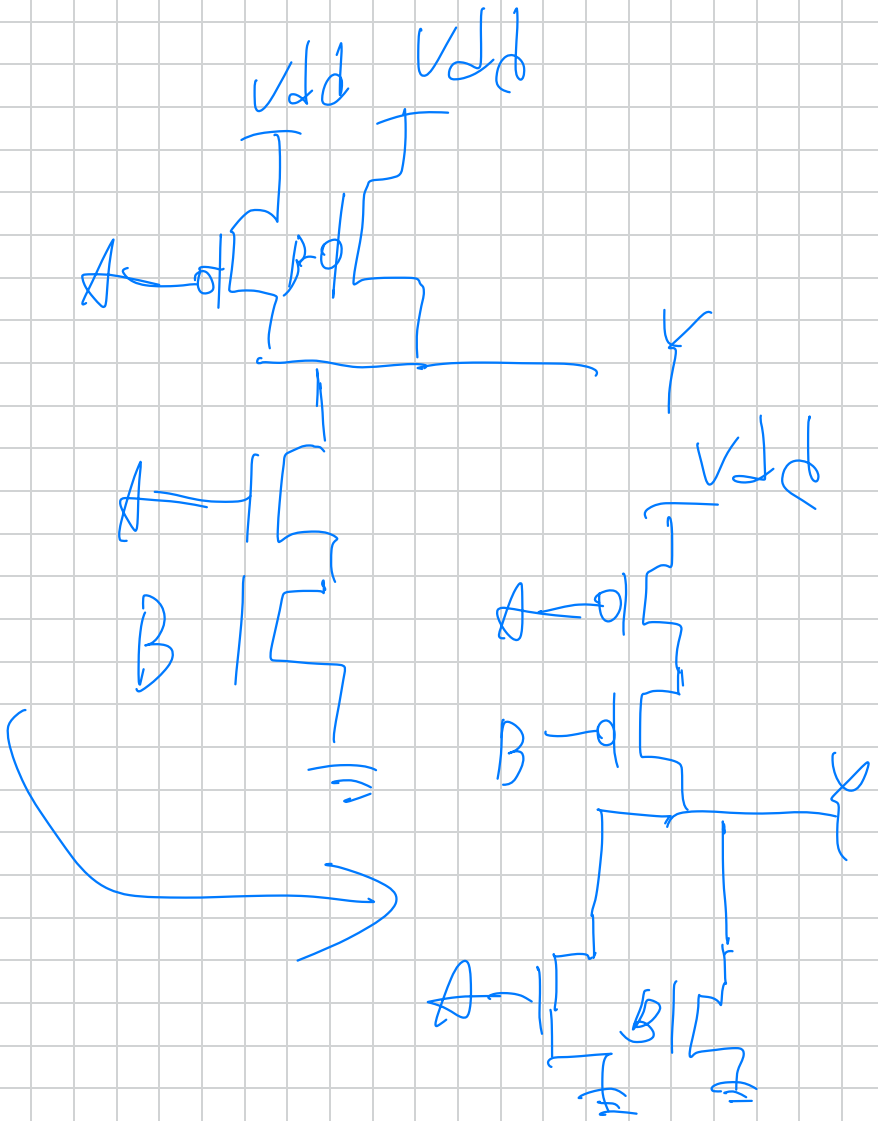
0	0	1
0	1	0
1	0	0
1	1	1

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

0	0	1
0	1	1
1	0	1
1	1	0

NOR = $\overline{A + B} = \bar{A} \cdot \bar{B}$

0	0	1
0	1	0
1	0	0
1	1	0



Transmission Gate Mux

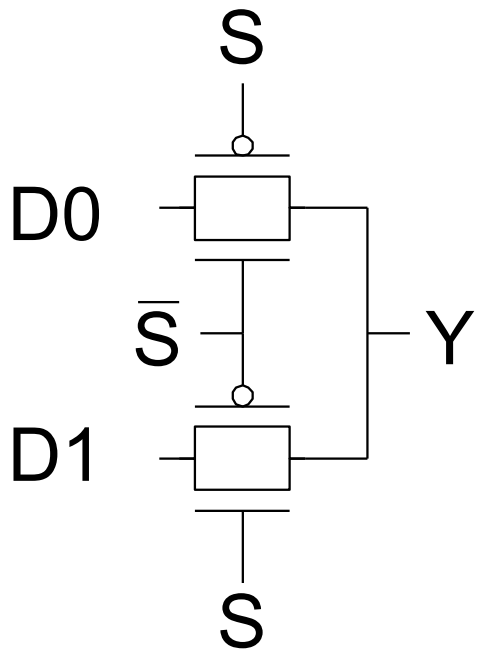
- Nonrestoring mux uses two transmission gates

Transmission Gate Mux

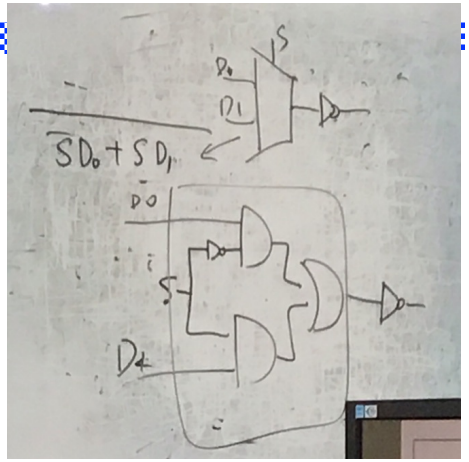
- Nonrestoring mux uses two transmission gates
  - Only 4 transistors

多工器2X1最後
可以變成這樣

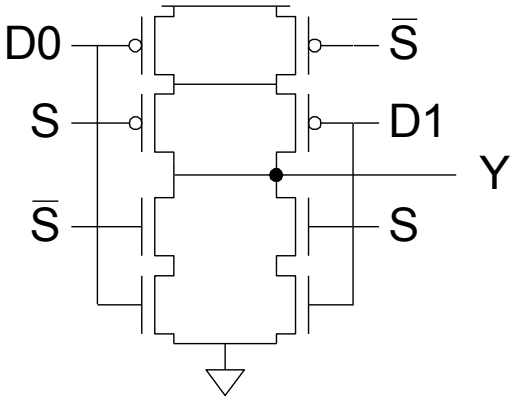
可以換成這樣
就只需要4個電晶體

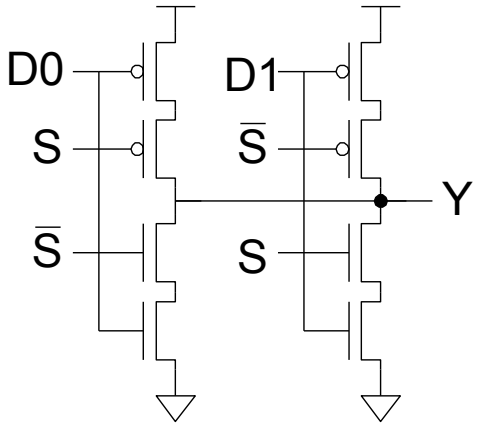


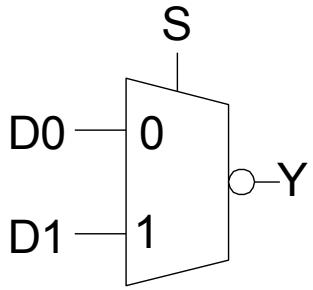
Inverting Mux



- Inverting multiplexer
  - Use compound AOI22
  - Or pair of tristate inverters
  - Essentially the same thing
- Noninverting multiplexer adds an inverter





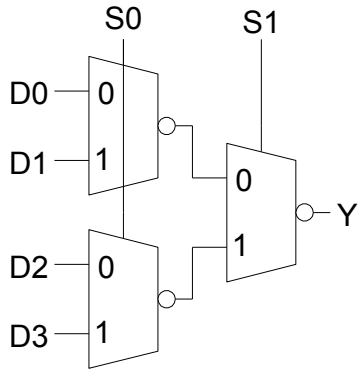


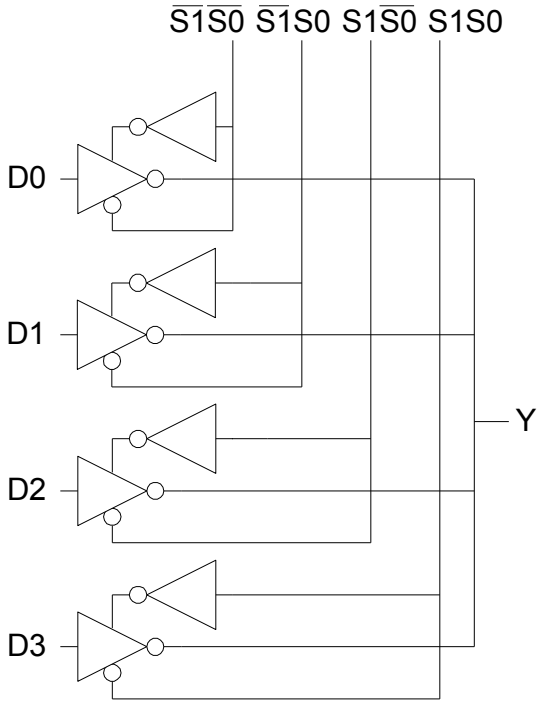
4:1 Multiplexer

- 4:1 mux chooses one of 4 inputs using two selects

4:1 Multiplexer

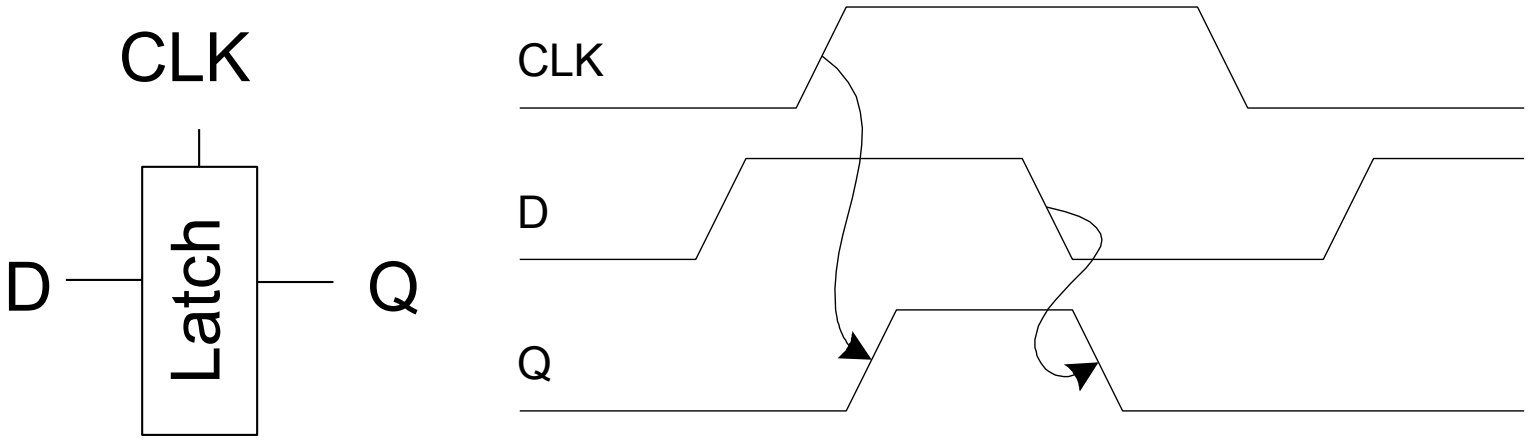
- 4:1 mux chooses one of 4 inputs using two selects
  - Two levels of 2:1 muxes
  - Or four tristates





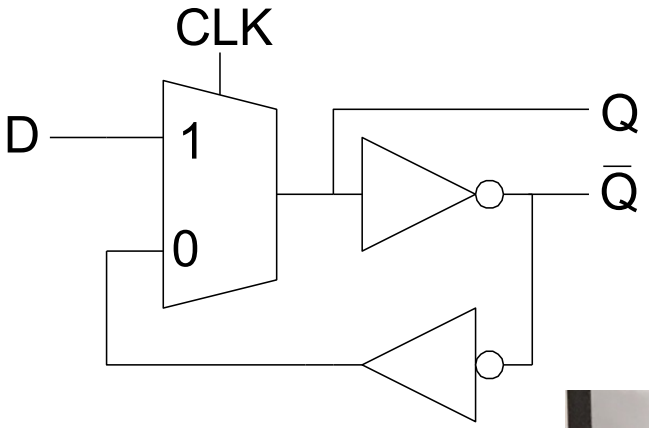
D Latch

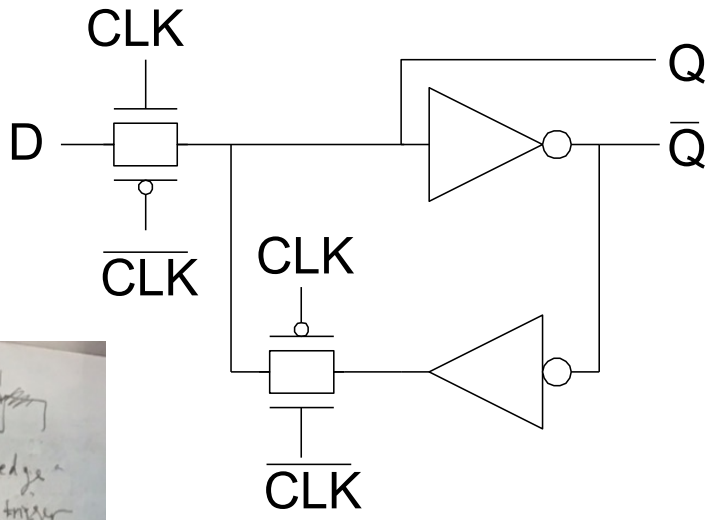
- When CLK = 1, latch is *transparent*
  - D flows through to Q like a buffer
- When CLK = 0, the latch is *opaque*
  - Q holds its old value independent of D
- a.k.a. *transparent latch* or *level-sensitive latch*

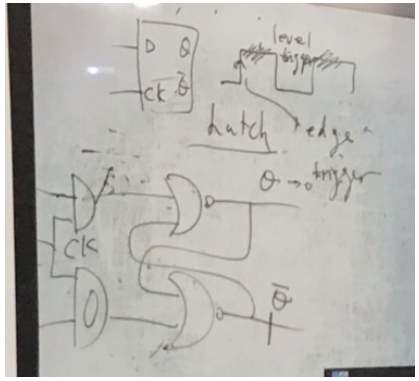


D Latch Design

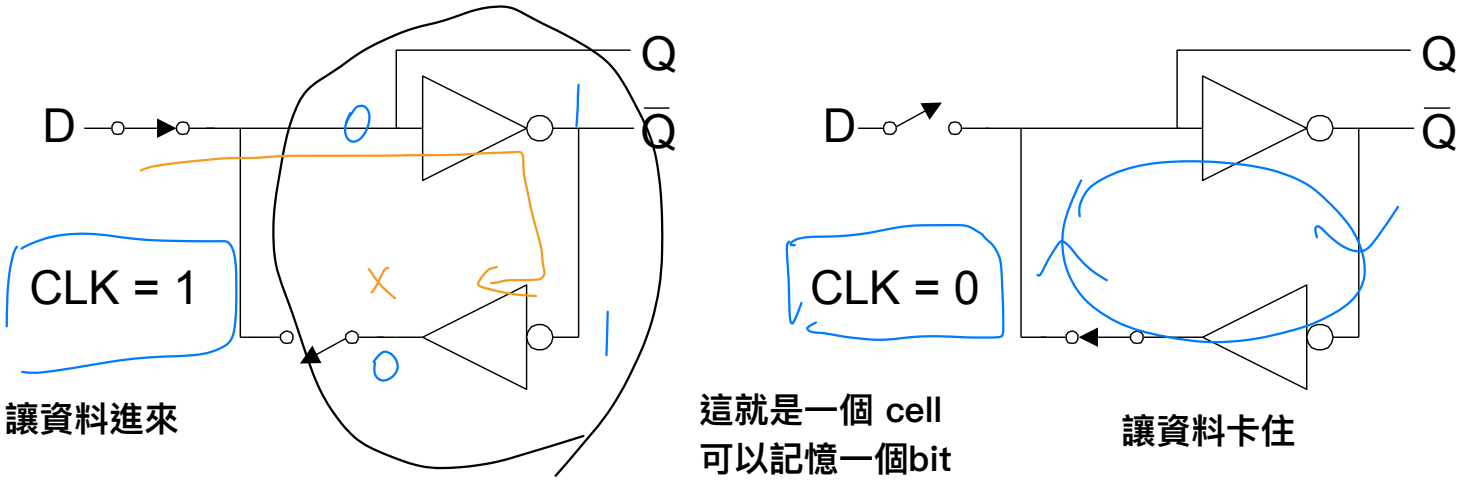
- Multiplexer chooses D or old Q

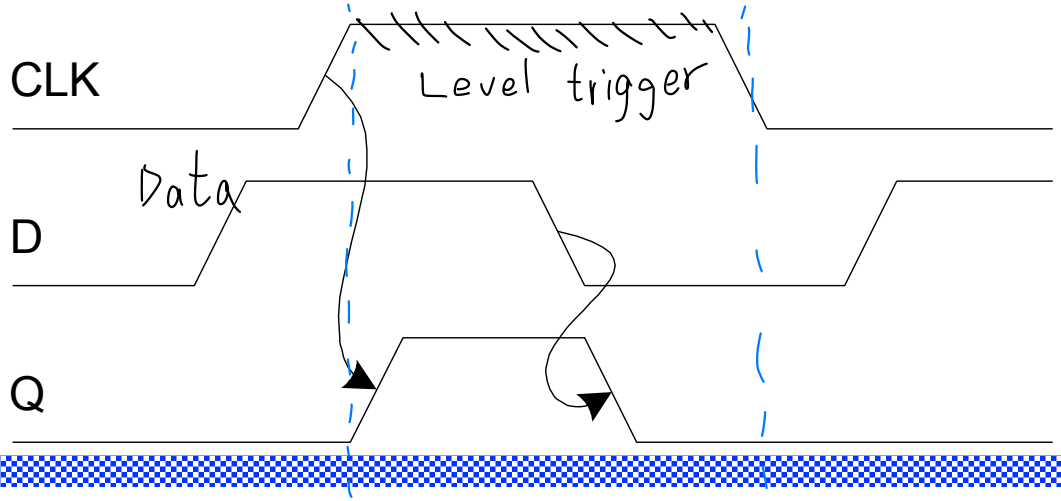






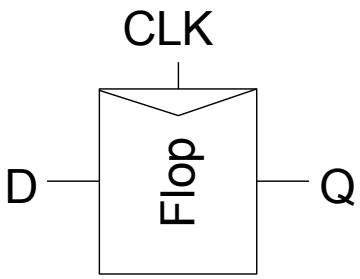
D Latch Operation

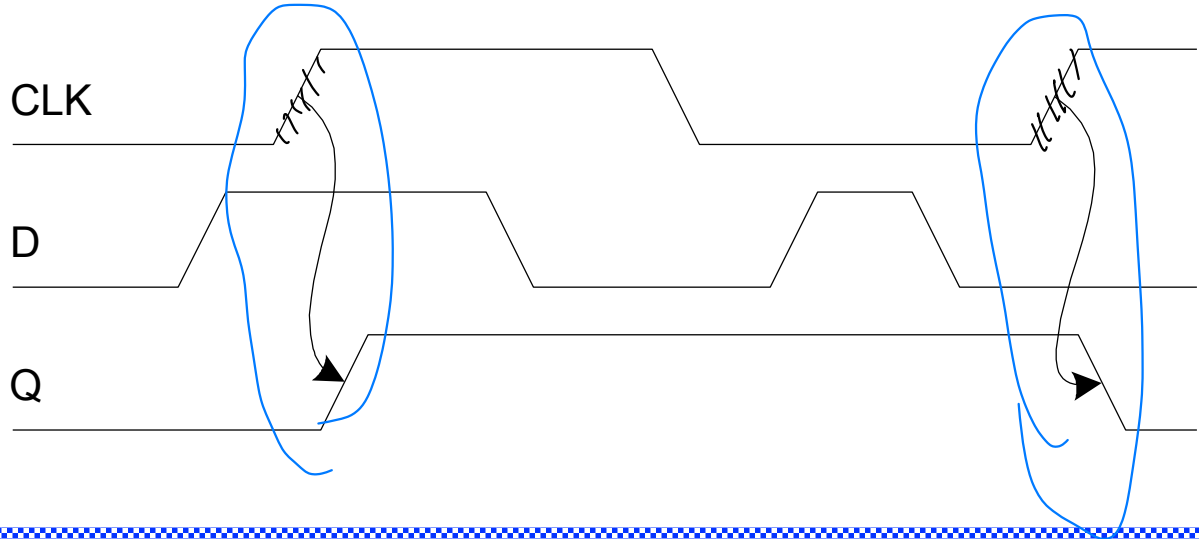


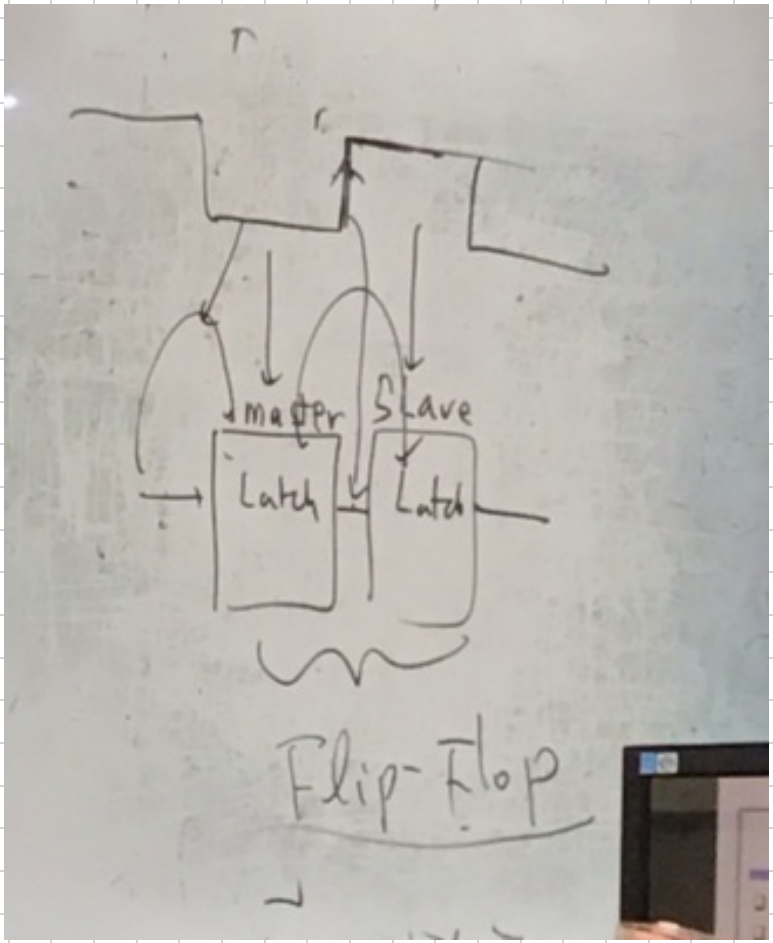


D Flip-flop

- When CLK rises, D is copied to Q
- At all other times, Q holds its value
- a.k.a. *positive edge-triggered flip-flop*, *master-slave flip-flop*



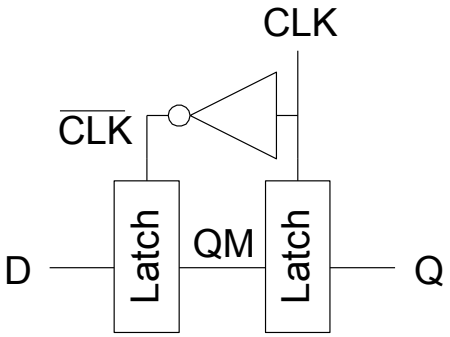


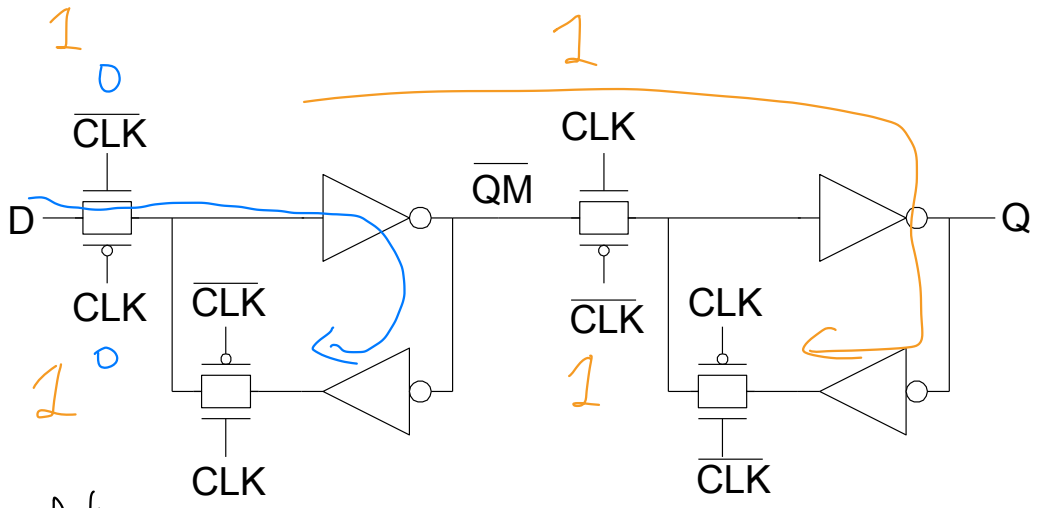


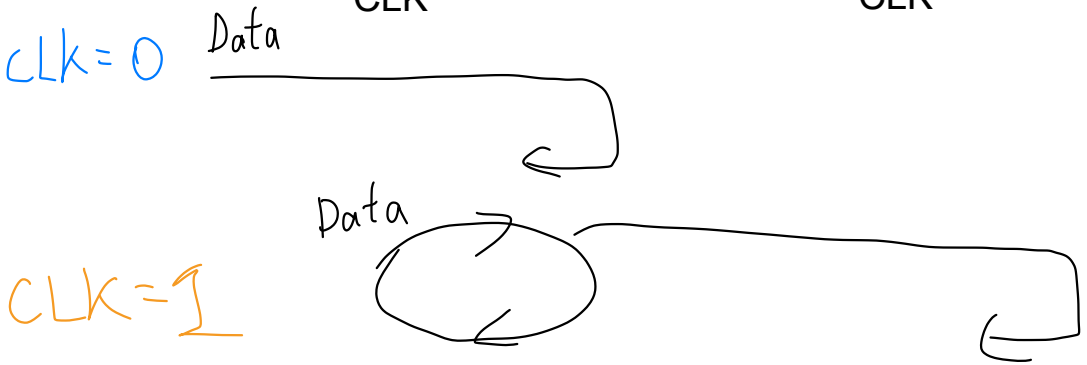
Flip Flop 是用兩個Latch 鬥出來的
主僕式 正反器

D Flip-flop Design

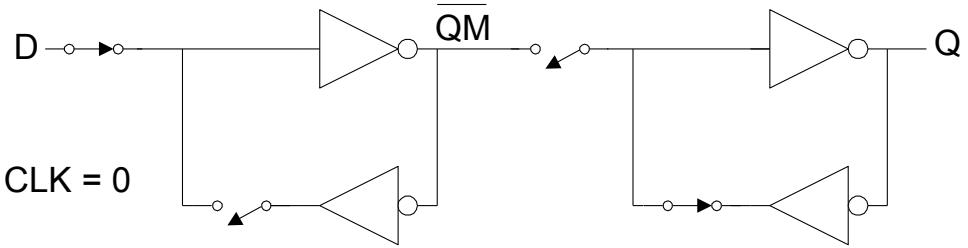
- Built from master and slave D latches

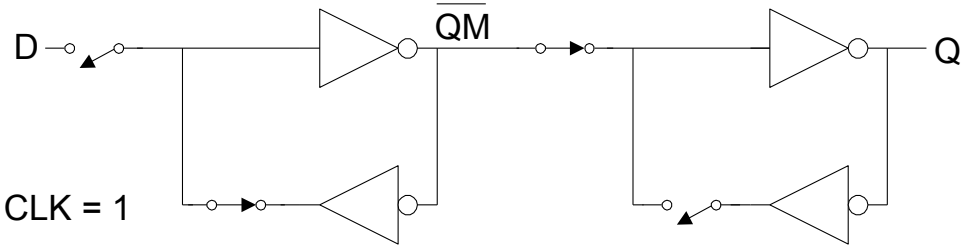


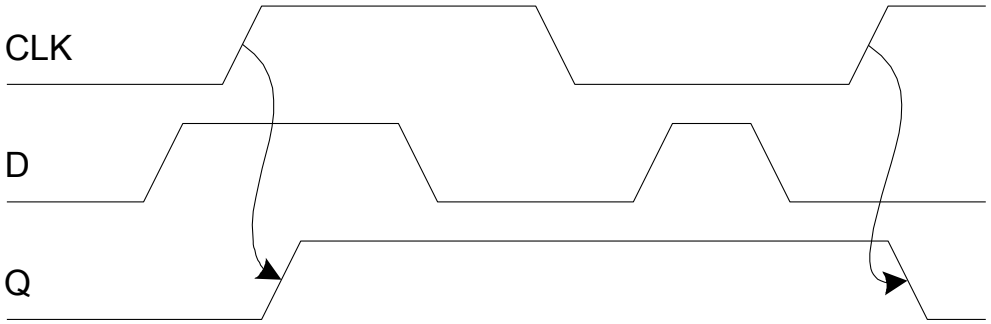




D Flip-flop Operation







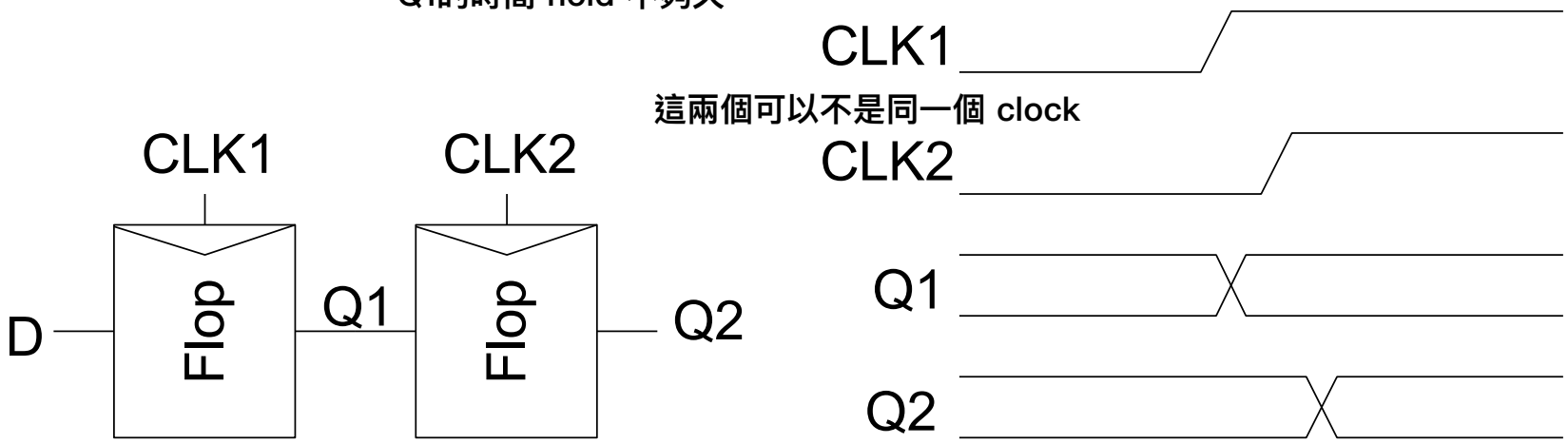
Race Condition

- Back-to-back flops can malfunction from clock skew
  - Second flip-flop fires late
  - Sees first flip-flop change and captures its result
  - Called *hold-time failure* or *race condition*

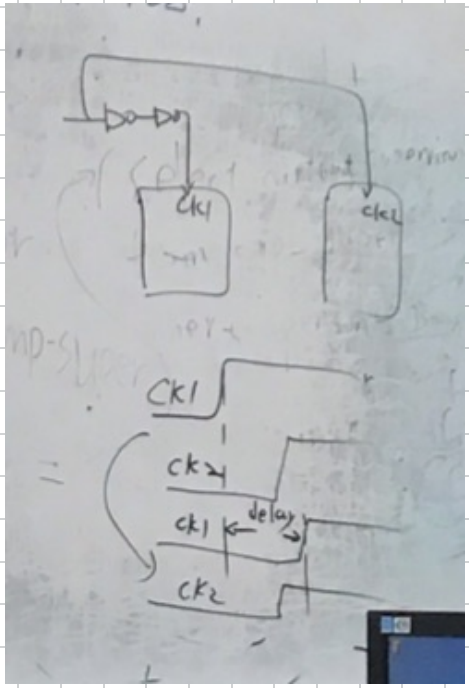
可能會提前把中間的資料 提前蓋掉了

Q1的時間 hold 不夠久

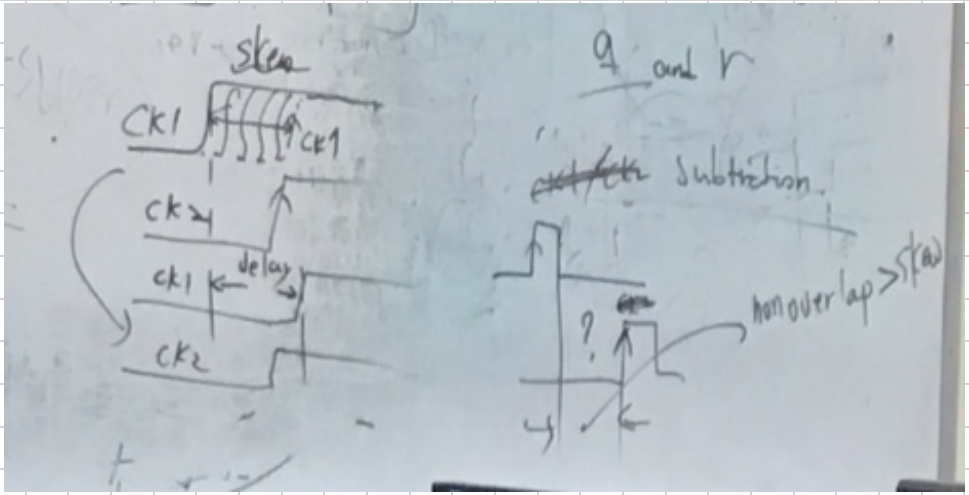
或叫做 爭跑



這兩個可以不是同一個 clock



也有一種方法 是用兩個not的延遲 來解決 爭跑問題

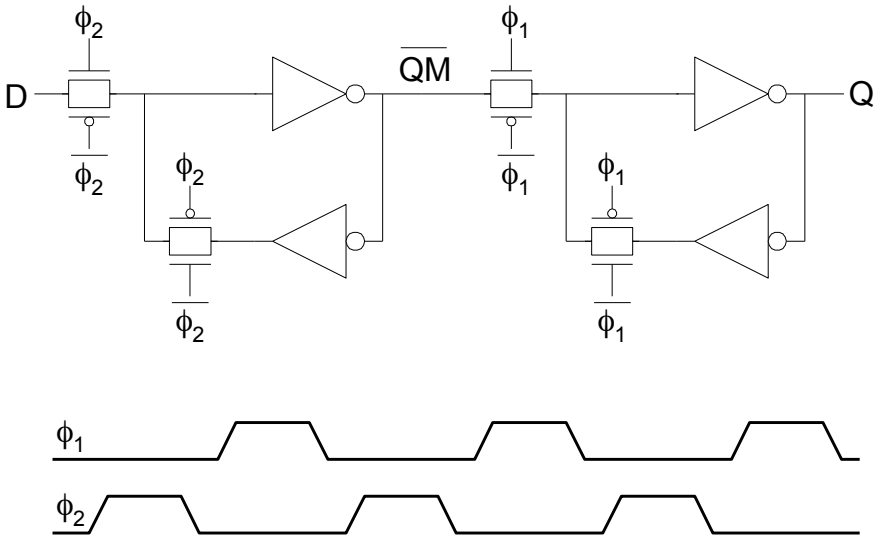


可能元件老化 就有可能造成 偏易skew的clock

所以希望資料傳輸間 的 間距時間
Nonoverlap > skew

Nonoverlapping Clocks

- Nonoverlapping clocks can prevent races
  - As long as nonoverlap exceeds clock skew
- We will use them in this class for safe design
  - Industry manages skew more carefully instead



Gate Layout

- Layout can be very time consuming
  - Design gates to fit together nicely
  - Build a library of standard cells
- Standard cell design methodology
  - V_{DD} and GND should abut (standard height)
  - Adjacent gates should satisfy design rules
  - nMOS at bottom and pMOS at top
  - All gates include well and substrate contacts

都運用標準元件設計方式 可以提升效率 和 正確率

nMOS pMOS 之間就是 迪摩根定理的轉換

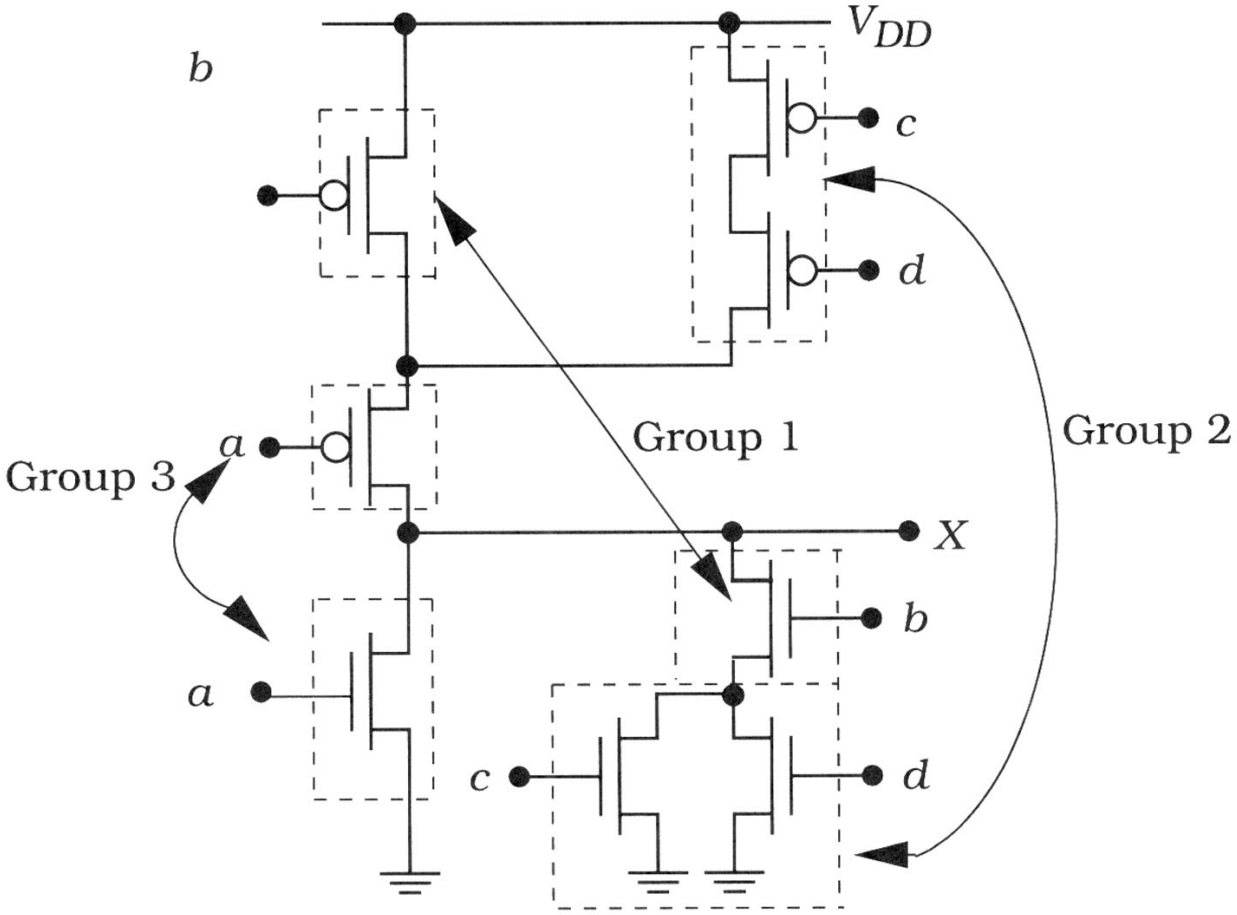
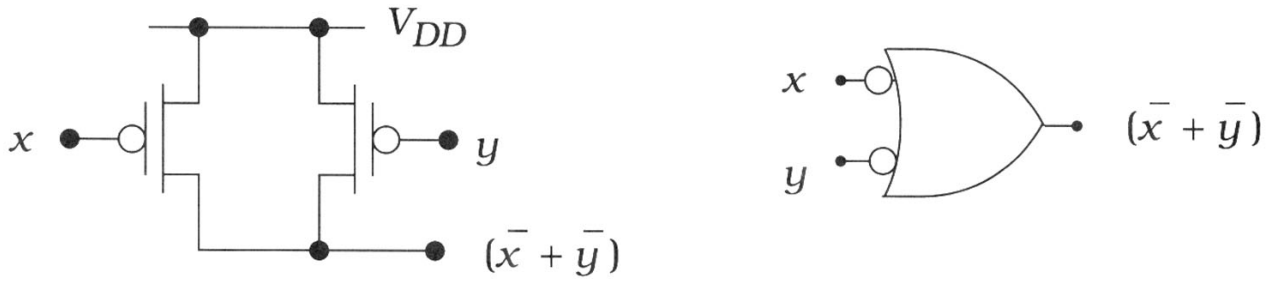
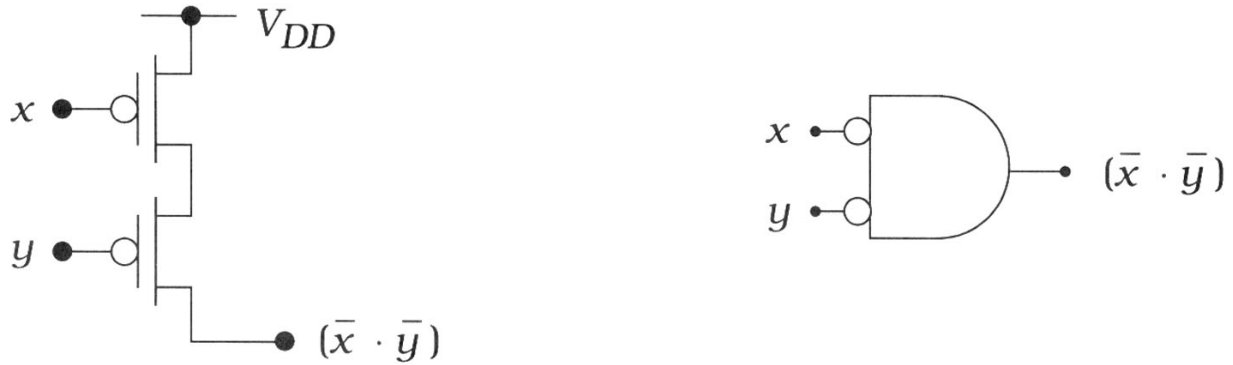


Figure 2.49 AOI circuit for Example 2.1



(a) Parallel-connected pFETs



(b) Series-connected pFETs

Figure 2.51 Assert-low models for pFETs

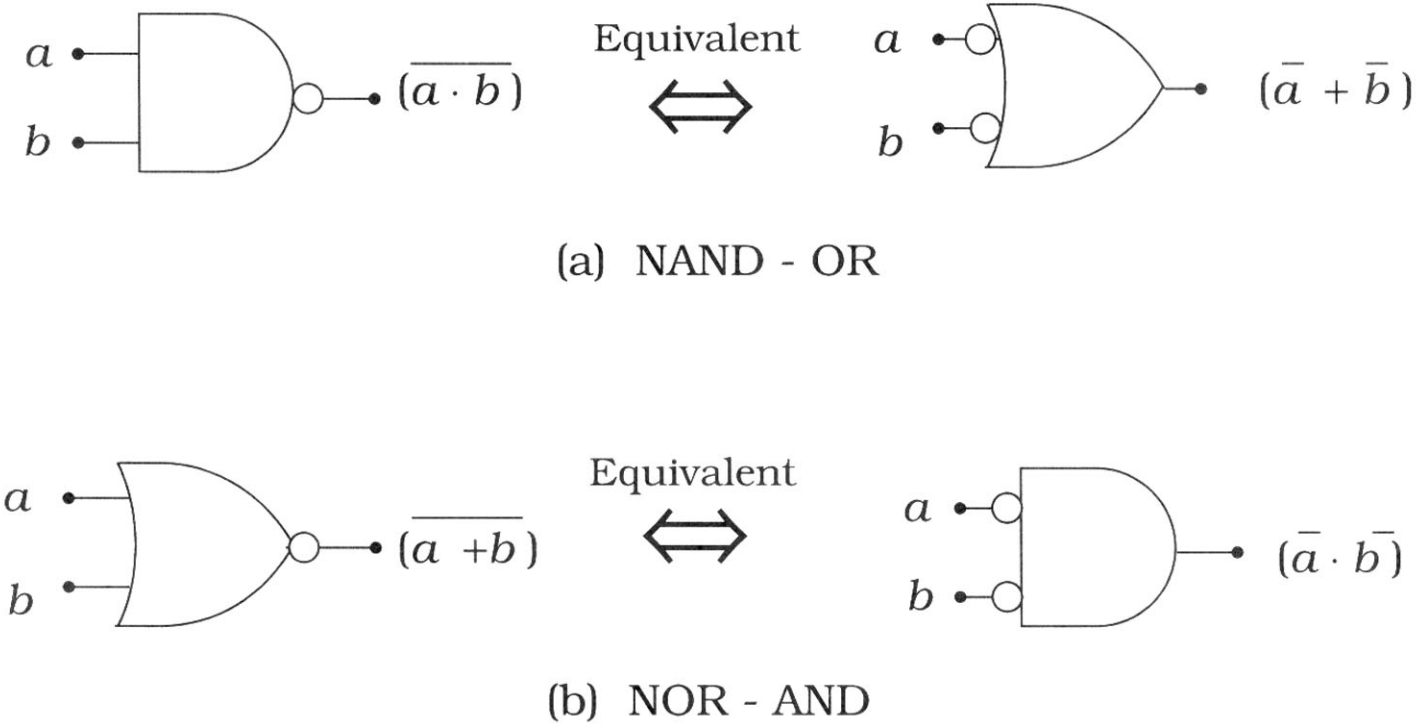


Figure 2.52 Bubble pushing using DeMorgan rules

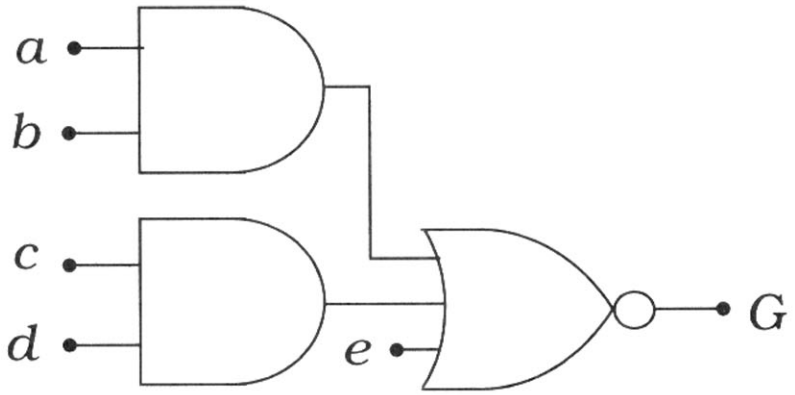
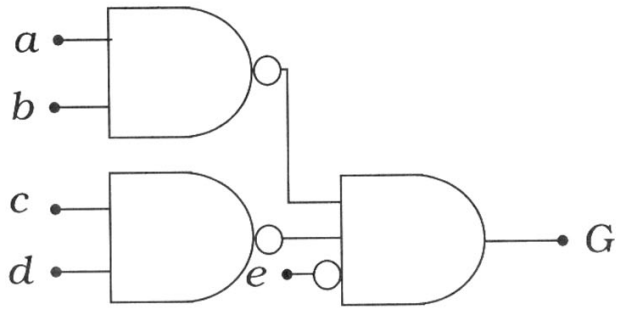
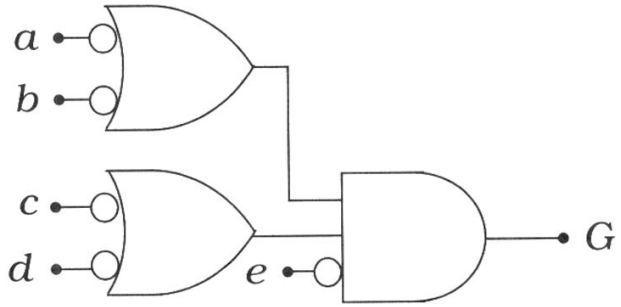


Figure 2.53 AOI logic diagram for bubble-pushing example



(a) First transformation



(b) Final form

Figure 2.54 Bubble pushing to obtain the topology of the pFET array

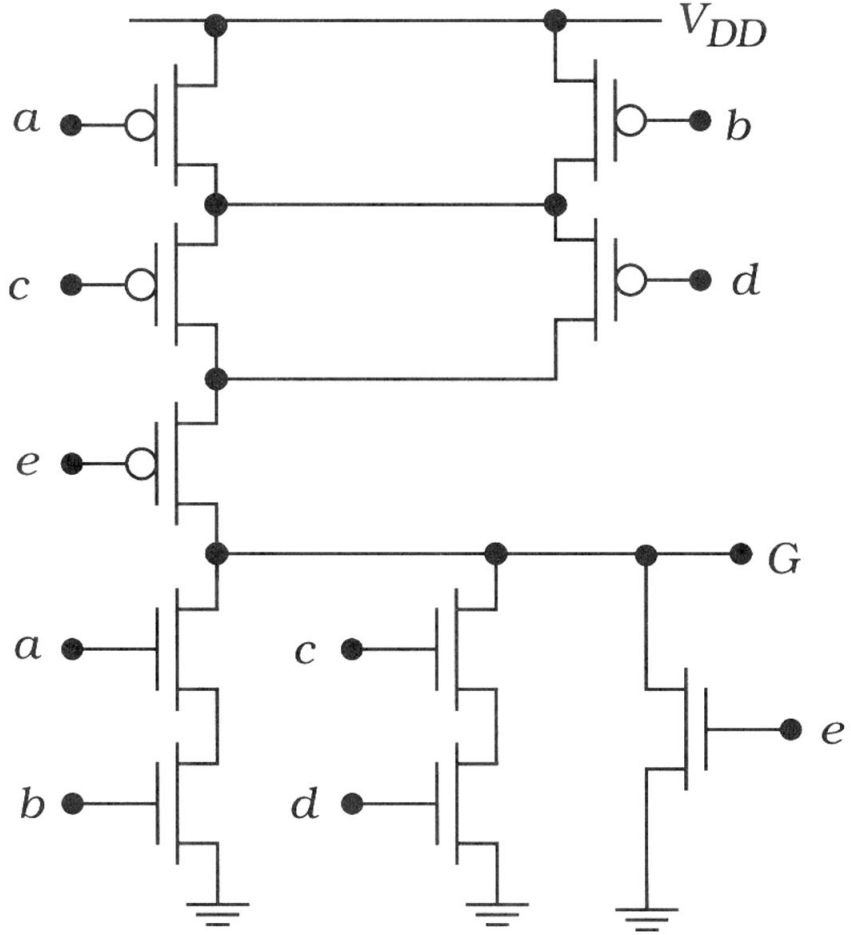
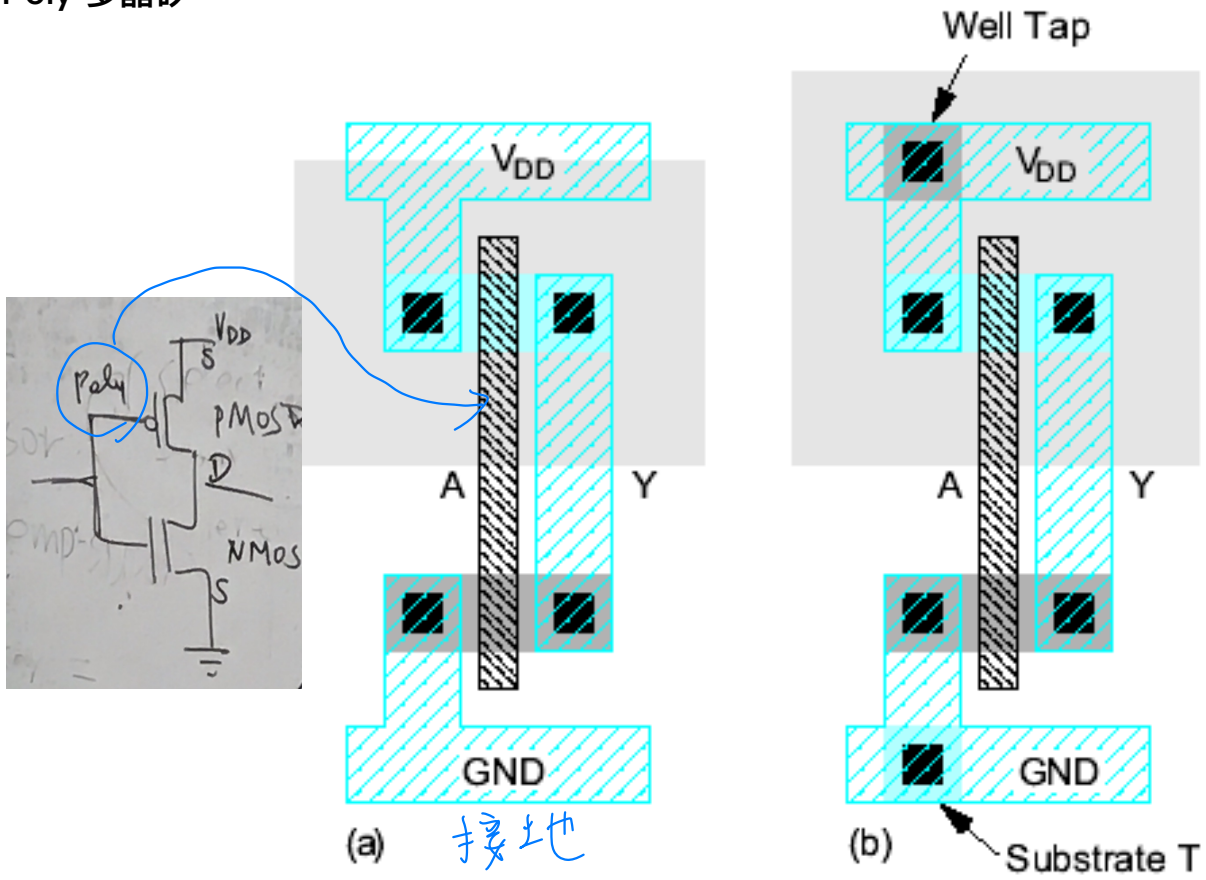


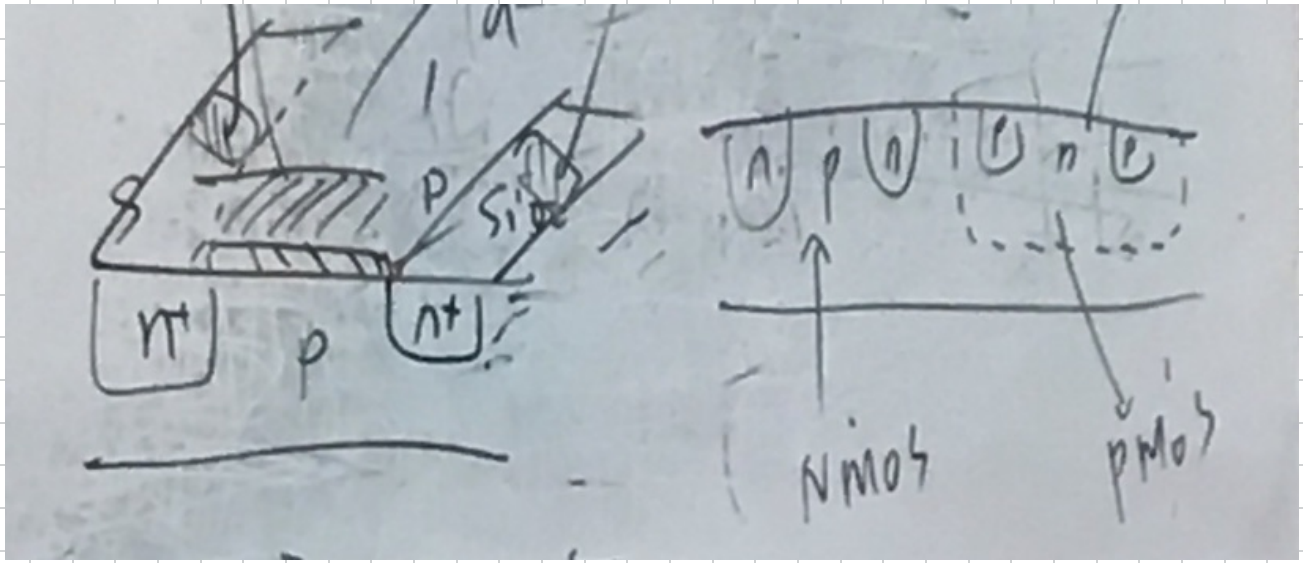
Figure 2.55 Final circuit for the bubble-pushing example

Example: Inverter

這個圖 叫 Layout

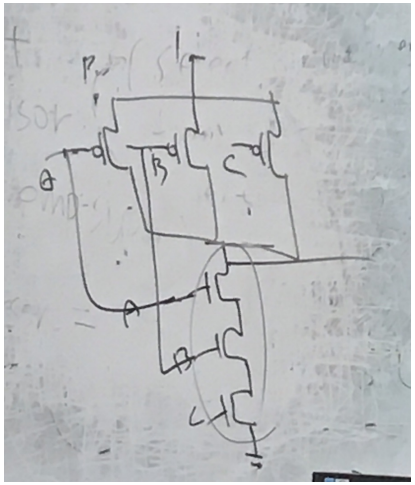
Poly 多晶矽




p
Si
n+
p
n
i
n
i
NMOS
PMOS

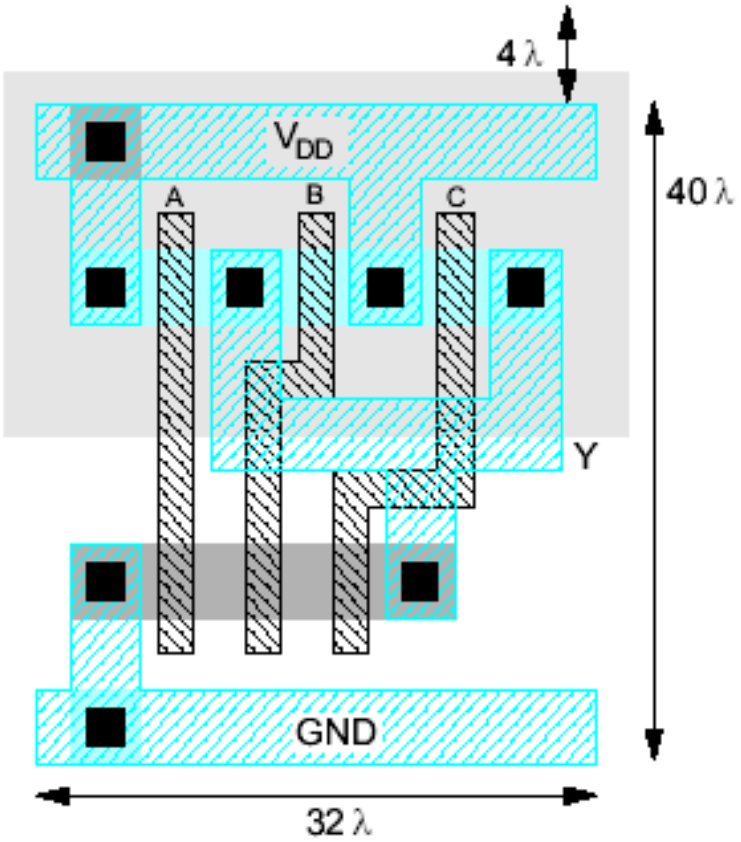
Example: NAND3

- Horizontal N-diffusion and p-diffusion strips
- Vertical polysilicon gates
- Metal1 V_{DD} rail at top
- Metal1 GND rail at bottom
- 32 λ by 40 λ



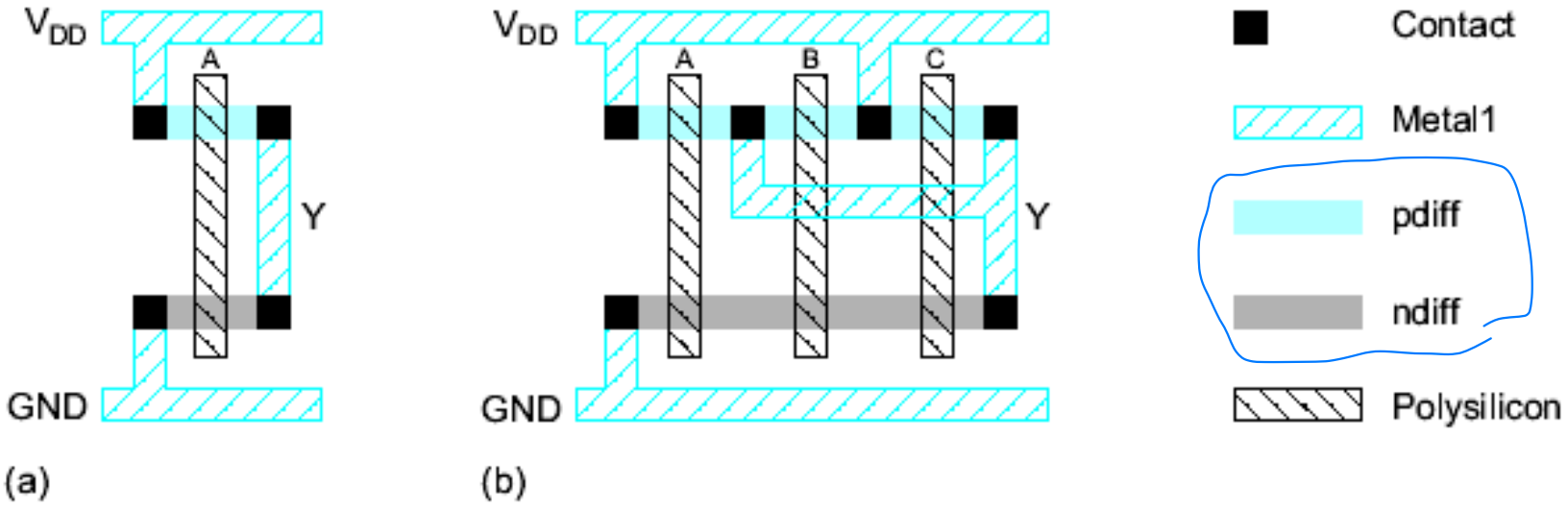
黑色正方形
就是要接出來
走電線的地方

淡藍色斜線
就是電線



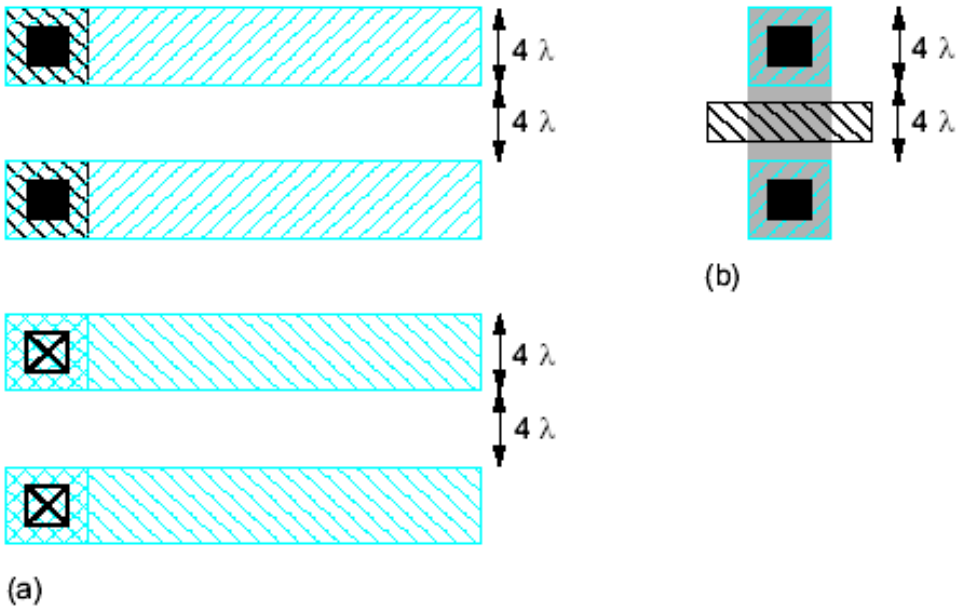
Stick Diagrams

- *Stick diagrams* help plan layout quickly
  - Need not be to scale
  - Draw with color pencils or dry-erase markers



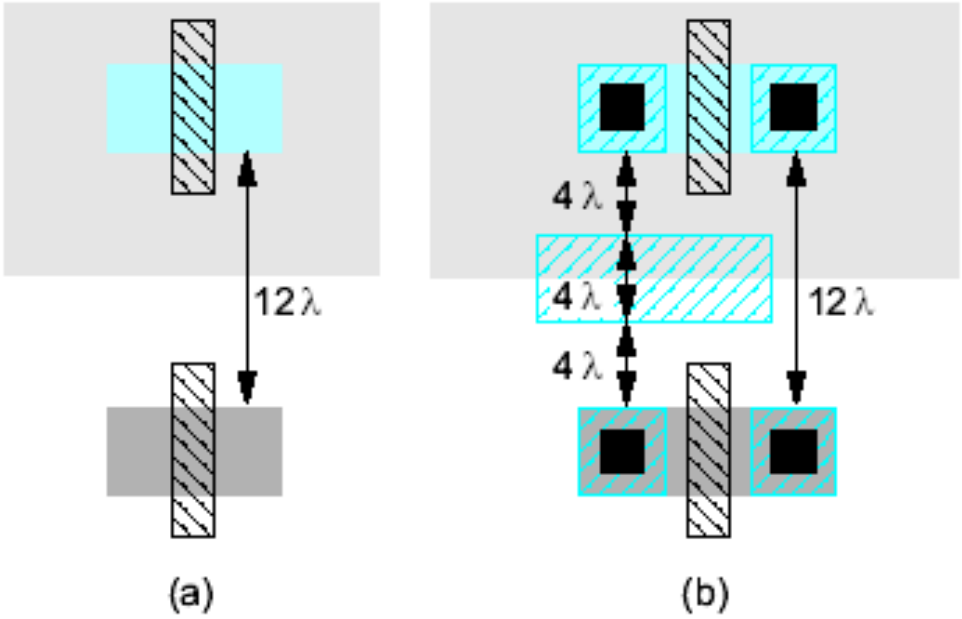
Wiring Tracks

- A *wiring track* is the space required for a wire
  - 4 λ width, 4 λ spacing from neighbor = 8 λ pitch
- Transistors also consume one wiring track



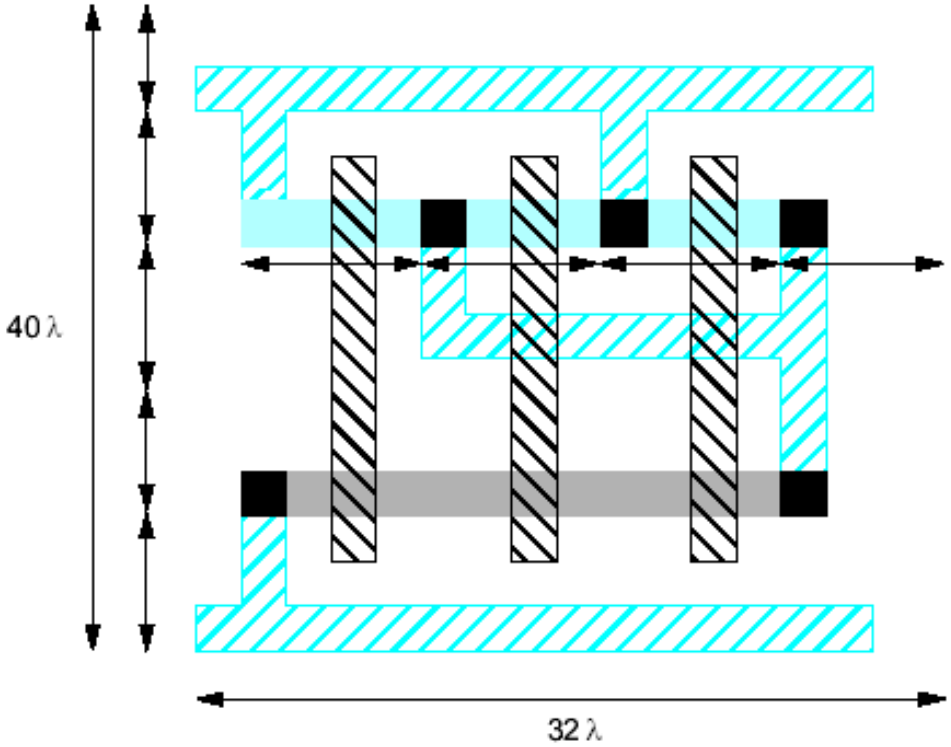
Well spacing

- Wells must surround transistors by 6 λ
  - Implies 12 λ between opposite transistor flavors
  - Leaves room for one wire track



Area Estimation

- Estimate area by counting wiring tracks
  - Multiply by 8 to express in λ

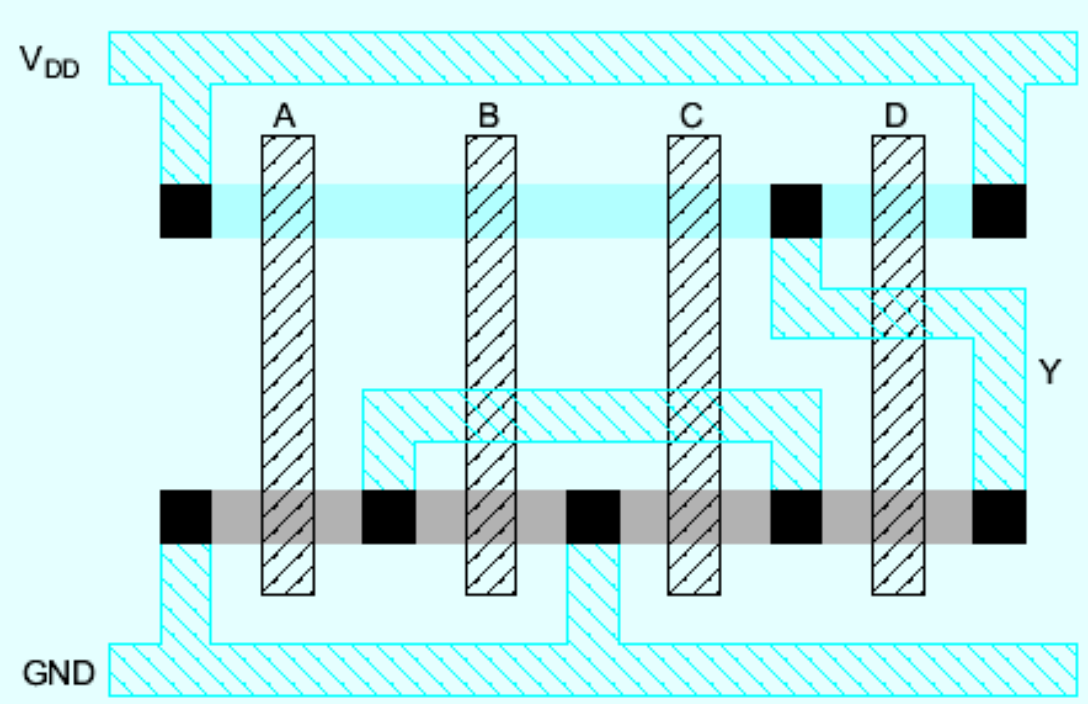


Example: O3AI

- Sketch a stick diagram for O3AI and estimate area
  - $\overline{Y = \left(A + B + C\right) \bullet D}$

Example: O3AI

- Sketch a stick diagram for O3AI and estimate area
  - $Y = \overline{(A + B + C) \cdot D}$



Example: O3AI

- Sketch a stick diagram for O3AI and estimate area
  - $Y = \overline{(A + B + C) \bullet D}$