

# Tema 1. Rendiment i Consum

Joan Manuel Parcerisa



UNIVERSITAT POLITÈCNICA DE CATALUNYA  
BARCELONATECH  
Facultat d'Informàtica de Barcelona



# Rendiment i Consum

- Mesures de rendiment
- Llei d'Amdahl
- Mesures de consum

# Definició de rendiment

Com sabem si un computador A ofereix millor *rendiment* que un computador B?

- ***Temps d'execució***

- "Temps transcorregut entre l'inici i el final d'una única tasca"
- Sinònim del *temps de resposta* d'una aplicació
- Rendiment és la inversa del temps d'execució

# Definició de rendiment

Com sabem si un computador A ofereix millor *rendiment* que un computador B?

- ***Temps d'execució***

- "Temps transcorregut entre l'inici i el final d'una única tasca"
- Sinònim del *temps de resposta* d'una aplicació
- Rendiment és la inversa del temps d'execució

- ***Productivitat (throughput)***

- "Nombre de tasques completades per unitat de temps"
- Mesura usada per a servidors web, centres de càlcul, bases de dades, etc.

# Relació entre temps d'execució i productivitat

- Exemple

- Un computador A té 1 CPU, que tarda  $10\mu\text{s}$  en completar una tasca. Un computador B té 200 CPUs, que tarden cadascun  $20\mu\text{s}$  en completar la mateixa tasca. Quin té major rendiment?

# Relació entre temps d'execució i productivitat

- Exemple
  - Un computador A té 1 CPU, que tarda  $10\mu\text{s}$  en completar una tasca. Un computador B té 200 CPUs, que tarden cadascun  $20\mu\text{s}$  en completar la mateixa tasca. Quin té major rendiment?
- A menor temps d'execució, major productivitat
- A major productivitat es redueix el temps d'execució?
  - Sols en cas de congestió, ja que es redueix el temps d'espera en cues.

# Definició de rendiment en EC

- En EC mesurarem **rendiment** amb el **temps d'execució**

$$\text{rendiment} = \frac{1}{t_{exe}}$$

# Definició de rendiment en EC

- En EC mesurarem **rendiment** amb el **temps d'execució**

$$\text{rendiment} = \frac{1}{t_{exe}}$$

- Components del temps d'execució
  - **Temps de CPU** ( $t_{cpu}$ )
  - Temps d'espera d'E/S ( $t_{E/S}$ )
  - Temps d'espera mentre s'executen altres tasques, en execució concurrent o *time-sharing* ( $t_{altres}$ )

$$t_{exe} = t_{cpu} + t_{E/S} + t_{altres}$$



# Definició de rendiment en EC

- En EC mesurarem **rendiment** amb el **temps d'execució**

$$\text{rendiment} = \frac{1}{t_{exe}}$$

- Components del temps d'execució
  - **Temps de CPU** ( $t_{cpu}$ )
  - Temps d'espera d'E/S ( $t_{E/S}$ )
  - Temps d'espera mentre s'executen altres tasques, en execució concurrent o *time-sharing* ( $t_{altres}$ )

$$t_{exe} = t_{cpu} + t_{E/S} + t_{altres}$$

- Per simplificar, en EC sols considerarem temps de CPU

$$t_{exe} = t_{cpu}$$

# Guany de rendiment (*speedup*)

- **Guany de Rendiment** (o *speedup*):
  - Expressa quants cops més ràpida s'executa una tasca en introduir una millora

$$\textit{Speedup} = \frac{\textit{rendiment}_{\textit{millorat}}}{\textit{rendiment}_{\textit{original}}} = \frac{t_{\textit{original}}}{t_{\textit{millorat}}}$$

- Les millores poden ser a molts nivells
  - En el programa (algorismes, estructures de dades, etc.)

# Guany de rendiment (*speedup*)

- **Guany de Rendiment** (o *speedup*):
  - Expressa quants cops més ràpida s'executa una tasca en introduir una millora

$$\text{Speedup} = \frac{\text{rendiment}_{\text{millorat}}}{\text{rendiment}_{\text{original}}} = \frac{t_{\text{original}}}{t_{\text{millorat}}}$$

- Les millores poden ser a molts nivells
  - En el programa (algorismes, estructures de dades, etc.)
  - En el compilador (optimització de bucles, de crides, etc.)

# Guany de rendiment (*speedup*)

- **Guany de Rendiment** (o *speedup*):

- Expressa quants cops més ràpida s'executa una tasca en introduir una millora

$$\text{Speedup} = \frac{\text{rendiment}_{\text{millorat}}}{\text{rendiment}_{\text{original}}} = \frac{t_{\text{original}}}{t_{\text{millorat}}}$$

- Les millores poden ser a molts nivells

- En el programa (algorismes, estructures de dades, etc.)
- En el compilador (optimització de bucles, de crides, etc.)
- En la microarquitectura (ALUs, pipeline, especulació, etc.)

# Guany de rendiment (*speedup*)

- **Guany de Rendiment** (o *speedup*):

- Expressa quants cops més ràpida s'executa una tasca en introduir una millora

$$\text{Speedup} = \frac{\text{rendiment}_{\text{millorat}}}{\text{rendiment}_{\text{original}}} = \frac{t_{\text{original}}}{t_{\text{millorat}}}$$

- Les millores poden ser a molts nivells

- En el programa (algorismes, estructures de dades, etc.)
- En el compilador (optimització de bucles, de crides, etc.)
- En la microarquitectura (ALUs, pipeline, especulació, etc.)
- En la tecnologia (freqüència de rellotge, ample de transistors)

- Exemple

- $T_{\text{original}} = 3\text{s}$
- $T_{\text{millorat}} = 2\text{s}$
- $\text{Speedup} = 3/2 = 1,5$

# Factors que influeixen en $t_{exe}$

- Podem expressar el temps d'execució

$$t_{exe} = n_{cicles} \times t_c = n_{cicles} / f_{clock}$$

$n_{cicles}$  = Número total de cicles de rellotge que tarda l'execució

$t_c$  = Temps de cicle o període de rellotge

$f_{clock}$  = Freqüència de rellotge

- Dues maneres de reduir el temps d'execució
  - Reduir el número de cicles
  - Augmentar la freqüència de rellotge (reduir temps de cicle)

# Reduir $n_{\text{cicles}}$

- Nombre de cicles

$$n_{\text{cicles}} = n_{\text{ins}} \times CPI$$

$n_{\text{ins}}$  = Número d'instruccions executades

$CPI$  = Promig de *cicles per instrucció*

# Reduir $n_{\text{cicles}}$

- Nombre de cicles

$$n_{\text{cicles}} = n_{\text{ins}} \times CPI$$

$n_{\text{ins}}$  = Número d'instruccions executades

$CPI$  = Promig de *cicles per instrucció*

- Cada tipus d'instrucció  $i$  té un  $CPI_i$  diferent

$$n_{\text{cicles}} = \sum_{i=1}^m (CPI_i \times n_i)$$

$CPI_i$  = Cicles que tarda una instrucció de tipus  $i$

$n_i$  = Nombre total d'instruccions de tipus  $i$



# Reduir $n_{\text{cicles}}$

- Nombre de cicles

$$n_{\text{cicles}} = n_{\text{ins}} \times CPI$$

$n_{\text{ins}}$  = Número d'instruccions executades

$CPI$  = Promig de *cicles per instrucció*

- Cada tipus d'instrucció  $i$  té un  $CPI_i$  diferent

$$n_{\text{cicles}} = \sum_{i=1}^m (CPI_i \times n_i)$$

$CPI_i$  = Cicles que tarda una instrucció de tipus  $i$

$n_i$  = Nombre total d'instruccions de tipus  $i$

- Reduir  $n_{\text{cicles}}$ 
  - Reduint el nombre d'instruccions  $n_{\text{ins}}$ : Millorar el compilador
  - Reduint el  $CPI$ 
    - Millorar la microarquitectura
    - Substituir instruccions costoses per altres de ràpides (`mult` per `sll`)

# Reduir $n_{\text{cicles}}$

- Exemple

- a) Comparar el rendiment de 2 versions P1 i P2 d'un mateix programa en un computador amb 3 tipus d'instruccions A, B, C

tipus	CPI	$n_i$	
		P1	P2
A	1	$2 \cdot 10^9$	$4 \cdot 10^9$
B	2	$1 \cdot 10^9$	$1 \cdot 10^9$
C	3	$2 \cdot 10^9$	$1 \cdot 10^9$

$$\begin{aligned} n_{\text{ciclesP1}} &= CPI_A \cdot n_A + CPI_B \cdot n_B + CPI_C \cdot n_C \\ &= 1 \cdot 2 \cdot 10^9 + 2 \cdot 1 \cdot 10^9 + 3 \cdot 2 \cdot 10^9 = \mathbf{10 \cdot 10^9 \text{ cicles}} \end{aligned}$$

$$\begin{aligned} n_{\text{ciclesP2}} &= CPI_A \cdot n_A + CPI_B \cdot n_B + CPI_C \cdot n_C \\ &= 1 \cdot 4 \cdot 10^9 + 2 \cdot 1 \cdot 10^9 + 3 \cdot 1 \cdot 10^9 = \mathbf{9 \cdot 10^9 \text{ cicles}} \quad (\rightarrow \text{+ràpid!}) \end{aligned}$$

# Reduir $n_{\text{cicles}}$

- Exemple

- a) Comparar el rendiment de 2 versions P1 i P2 d'un mateix programa en un computador amb 3 tipus d'instruccions A, B, C

tipus	CPI	$n_i$	
		P1	P2
A	1	$2 \cdot 10^9$	$4 \cdot 10^9$
B	2	$1 \cdot 10^9$	$1 \cdot 10^9$
C	3	$2 \cdot 10^9$	$1 \cdot 10^9$

$$\begin{aligned} n_{\text{ciclesP1}} &= CPI_A \cdot n_A + CPI_B \cdot n_B + CPI_C \cdot n_C \\ &= 1 \cdot 2 \cdot 10^9 + 2 \cdot 1 \cdot 10^9 + 3 \cdot 2 \cdot 10^9 = \mathbf{10 \cdot 10^9 \text{ cicles}} \end{aligned}$$

$$\begin{aligned} n_{\text{ciclesP2}} &= CPI_A \cdot n_A + CPI_B \cdot n_B + CPI_C \cdot n_C \\ &= 1 \cdot 4 \cdot 10^9 + 2 \cdot 1 \cdot 10^9 + 3 \cdot 1 \cdot 10^9 = \mathbf{9 \cdot 10^9 \text{ cicles}} \quad (\rightarrow \text{+ràpid!}) \end{aligned}$$

- b) Suposant  $f_{\text{clock}} = 2\text{GHz}$ , calcular  $t_{\text{exe}}$  i el speedup de P2

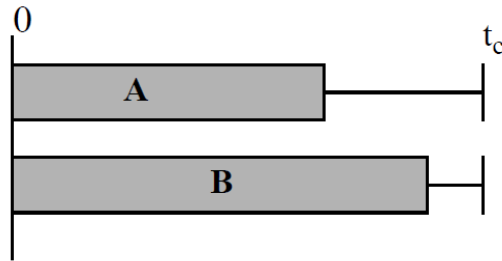
$$t_{\text{exeP1}} = n_{\text{ciclesP1}} / f_{\text{clock}} = 10 \cdot 10^9 / (2 \cdot 10^9) = \mathbf{5 \text{ s}}$$

$$t_{\text{exeP2}} = n_{\text{ciclesP2}} / f_{\text{clock}} = 9 \cdot 10^9 / (2 \cdot 10^9) = \mathbf{4,5 \text{ s}}$$

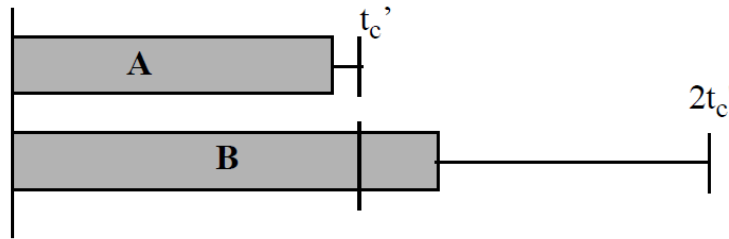
$$\text{Speedup}_{\text{P2}} = t_{\text{exeP1}} / t_{\text{exeP2}} = 5 / 4,5 = \mathbf{1,1}$$

# Augmentar $f_{\text{clock}}$ (reduir $t_c$ )

- Suposem els següents temps d'execució d'instruccions del tipus A i B, amb  $\text{CPI}=1$  i un temps de cicle  $t_c$



- Reduïm el temps de cicle de  $t_c$  a  $t'_c$



- Es redueix la latència de les instruccions de tipus A
- Les de tipus B ara requereixen 2 cicles en lloc d'1 ( $\text{CPI}_B=2$ )
- El benefici depèn del número d'instruccions de tipus A i B
- Augmentar  $f_{\text{clock}}$  no sempre millora el rendiment!**

# Augmentar $f_{\text{clock}}$ (reduir $t_c$ )

- Exemple

- Un processador A té un temps de cicle  $t_{cA} = 500$  ps. Amb un programa de test hem mesurat en promig un  $\text{CPI}_A = 2$
- El redissenem perquè usi un menor temps de cicle. El nou processador B té  $t_{cB} = 250$  ps. Però el canvi ha comportat un major nombre de cicles del programa de test, i  $\text{CPI}_B = 3$
- És més ràpid el nou disseny?

# Augmentar $f_{\text{clock}}$ (reduir $t_c$ )

- Exemple

- Un processador A té un temps de cicle  $t_{cA} = 500$  ps. Amb un programa de test hem mesurat en promig un  $\text{CPI}_A = 2$
- El redissenem perquè usi un menor temps de cicle. El nou processador B té  $t_{cB} = 250$  ps. Però el canvi ha comportat un major nombre de cicles del programa de test, i  $\text{CPI}_B = 3$
- És més ràpid el nou disseny?

- Solució

- Calculem el guany de rendiment (*speedup*) de B

$$\begin{aligned} s &= t_{\text{exeA}} / t_{\text{exeB}} \\ &= (n_{\text{ins}} \cdot \text{CPI}_A \cdot t_{cA}) / (n_{\text{ins}} \cdot \text{CPI}_B \cdot t_{cB}) \\ &= (n_{\text{ins}} \cdot 2 \cdot 500 \cdot 10^{-12}) / (n_{\text{ins}} \cdot 3 \cdot 250 \cdot 10^{-12}) \\ &= 1000 / 750 = \mathbf{1,33} \rightarrow \mathbf{B \text{ és } 1,33 \text{ cops més ràpid que A}} \end{aligned}$$

# Llei de Amdahl

"El màxim speedup  $S_{max}$  que es pot aconseguir minimitzant el retard d'una part està limitat per la fracció  $P$  del temps d'execució que representa aquesta part sobre el temps total"

- Una tasca s'executa inicialment en un temps  $t_0$
- En millorem una part que comporta una fracció  $P$  del temps
- La fracció millorada obté un guany de rendiment  $S$

# Llei de Amdahl

"El màxim speedup  $S_{max}$  que es pot aconseguir minimitzant el retard d'una part està limitat per la fracció  $P$  del temps d'execució que representa aquesta part sobre el temps total"

- Una tasca s'executa inicialment en un temps  $t_0$
- En millorem una part que comporta una fracció  $P$  del temps
- La fracció millorada obté un guany de rendiment  $S$ 
  - Temps de la fracció no-millorada =  $(1 - P) \cdot t_0$
  - Temps de la fracció millorada =  $(P \cdot t_0) / S$
  - Temps final d'execució  $t_1 = (1 - P) \cdot t_0 + (P \cdot t_0) / S$
- El *speedup* total és  $S_t = t_0 / t_1$

$$S_t = \frac{t_0}{(1-P) \cdot t_0 + \frac{P \cdot t_0}{S}} = \frac{1}{(1-P) + \frac{P}{S}}$$



# Llei de Amdahl

"El màxim speedup  $S_{max}$  que es pot aconseguir minimitzant el retard d'una part està limitat per la fracció  $P$  del temps d'execució que representa aquesta part sobre el temps total"

- Una tasca s'executa inicialment en un temps  $t_0$
- En millorem una part que comporta una fracció  $P$  del temps
- La fracció millorada obté un guany de rendiment  $S$ 
  - Temps de la fracció no-millorada =  $(1 - P) \cdot t_0$
  - Temps de la fracció millorada =  $(P \cdot t_0) / S$
  - Temps final d'execució  $t_1 = (1 - P) \cdot t_0 + (P \cdot t_0) / S$
- El **speedup** total és  $S_t = t_0 / t_1$

$$S_t = \frac{t_0}{(1-P) \cdot t_0 + \frac{P \cdot t_0}{S}} = \frac{1}{(1-P) + \frac{P}{S}}$$

- El **màxim speedup** que es pot aconseguir (per a  $S \rightarrow \infty$ ) és

$$S_{max} = \frac{1}{1-P}$$

# Llei de Amdahl

"El màxim speedup  $S_{max}$  que es pot aconseguir minimitzant el retard d'una part està limitat per la fracció  $P$  del temps d'execució que representa aquesta part sobre el temps total"

- Una tasca s'executa inicialment en un temps  $t_0$
- En millorem una part que comporta una fracció  $P$  del temps
- La fracció millorada obté un guany de rendiment  $S$ 
  - Temps de la fracció no-millorada =  $(1 - P) \cdot t_0$
  - Temps de la fracció millorada =  $(P \cdot t_0) / S$
  - Temps final d'execució  $t_1 = (1 - P) \cdot t_0 + (P \cdot t_0) / S$
- El **speedup** total és  $S_t = t_0 / t_1$

$$S_t = \frac{t_0}{(1-P) \cdot t_0 + \frac{P \cdot t_0}{S}} = \frac{1}{(1-P) + \frac{P}{S}}$$

- El **màxim speedup** que es pot aconseguir (per a  $S \rightarrow \infty$ ) és

$$S_{max} = \frac{1}{1-P}$$

- Quin és el màxim speedup que podem obtenir si millorem un 80% del temps d'execució?  $\rightarrow s_{max} = 1/(1-0,80) = 5$

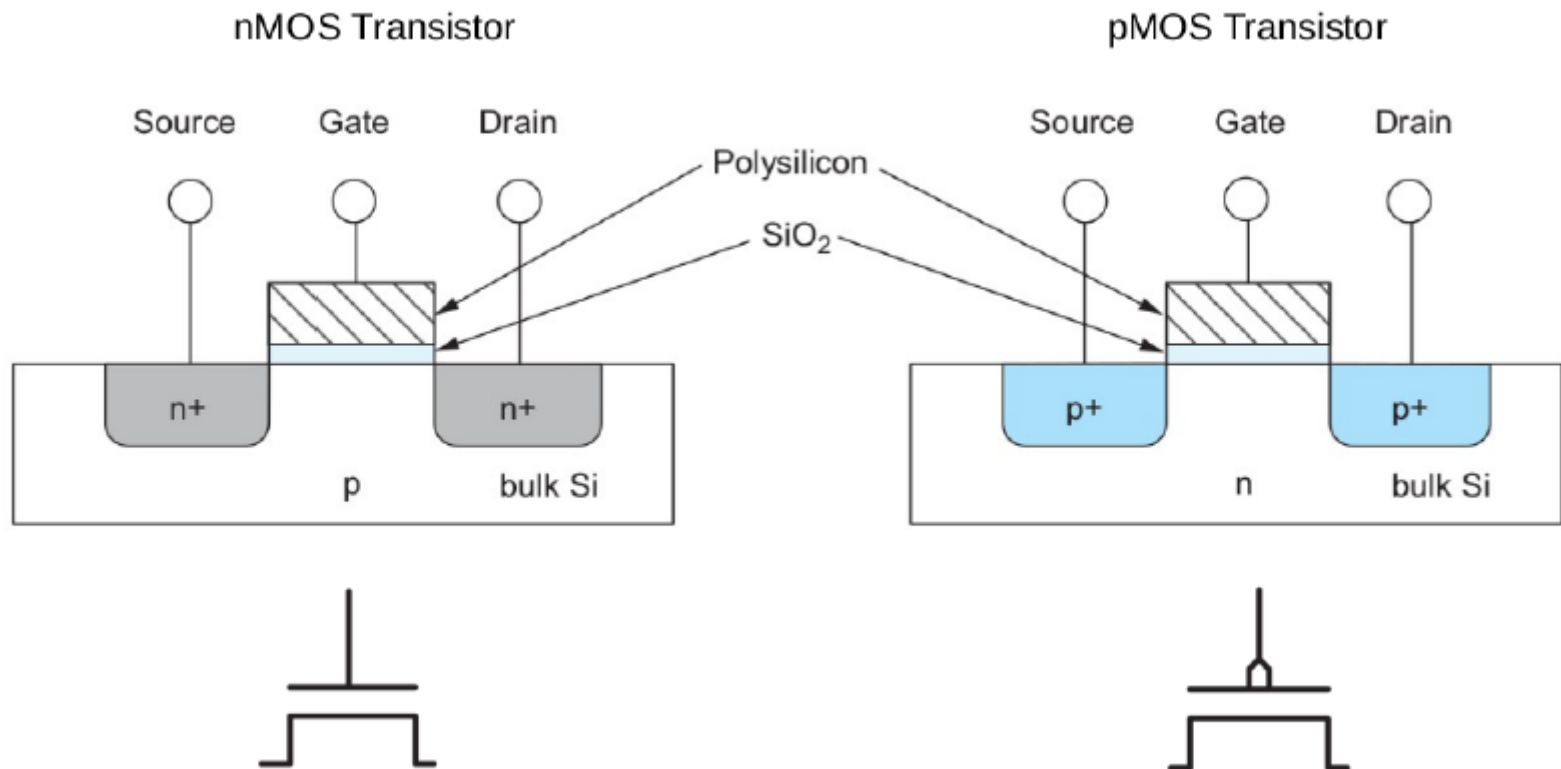
# Dissipació de potència

- **Potència (P)** = Energia dissipada per unitat de temps degut als corrents elèctrics del circuit
  - **Potència dinàmica ( $P_d$ )**: deguda a la càrrega i descàrrega de les capacitàncies dels transistors que conmuten
  - **Potència estàtica ( $P_s$ )**: deguda als corrents paràsits en els transistors, independentment de si conmuten o no

$$P = P_d + P_s$$

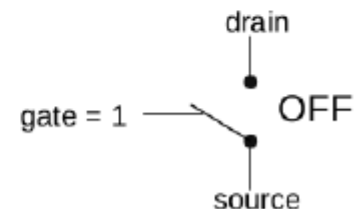
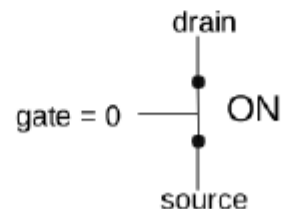
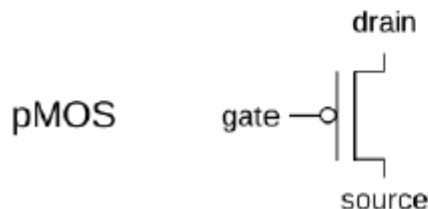
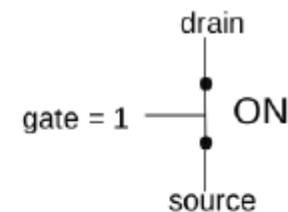
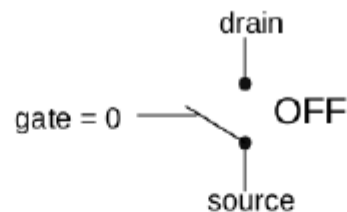
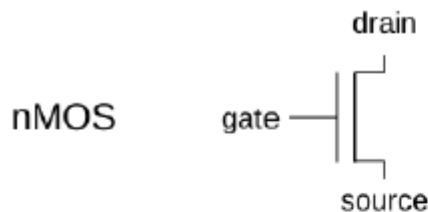
# Transistors *nmos* i *pmos*

- MOS (Metal Oxide Semiconductor)



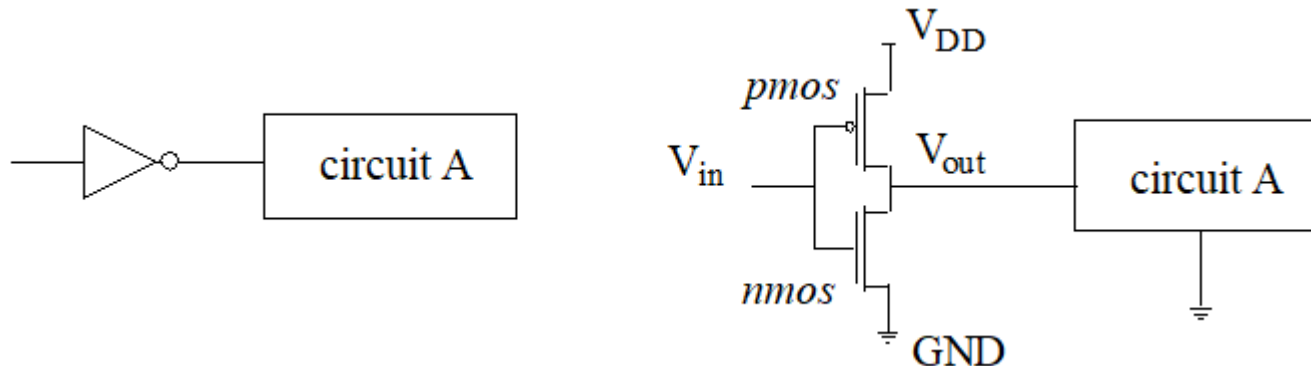
# Transistors *nmos* i *pmos*

- La tensió  $V_{in}$  d'entrada a la porta (gate) determina l'estat lògic del transistor, comparada amb la tensió llindar  $V_{th}$ 
  - Per ex., en el cas nMOS:
    - $V_{in} \leq V_{th} \rightarrow \text{estat lògic} = 0$
    - $V_{in} > V_{th} \rightarrow \text{estat lògic} = 1$
- Determina si està en circuit-obert (OFF) o curt-circuit (ON)



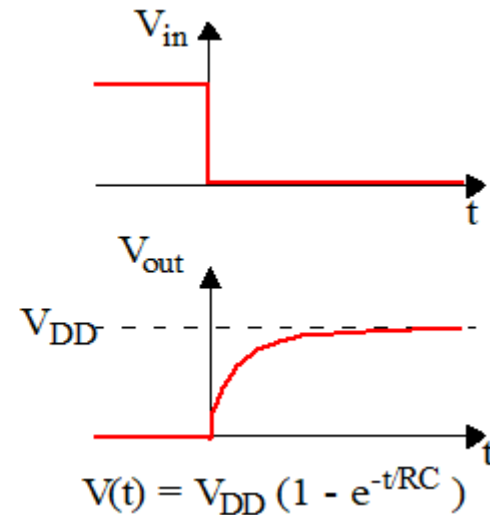
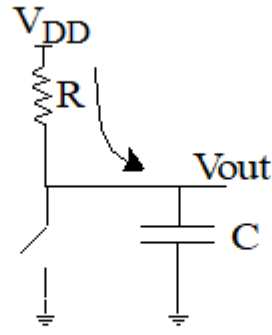
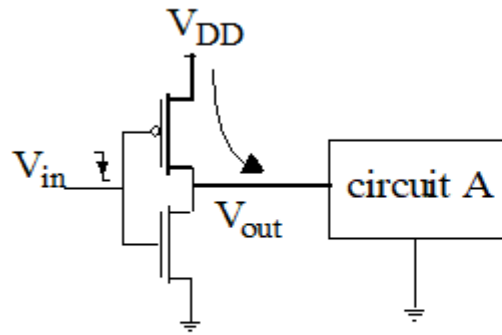
# Càrrega i descàrrega dels transistors

- Tecnologia CMOS (Complementary MOS)
  - Cada porta lògica inclou 2 subcircuitos: *pmos* i *nmos*
- Exemple: La porta NOT



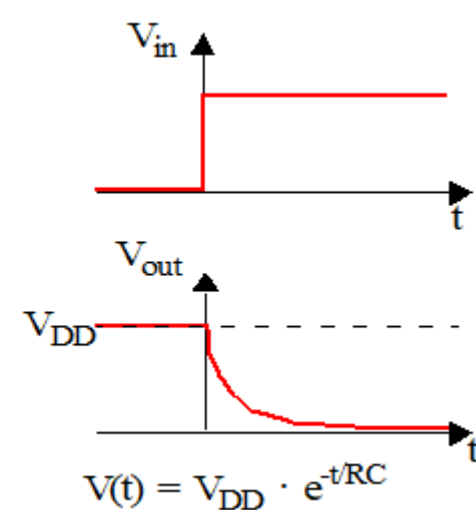
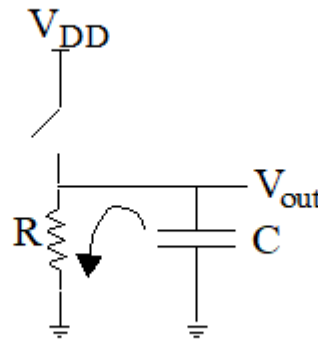
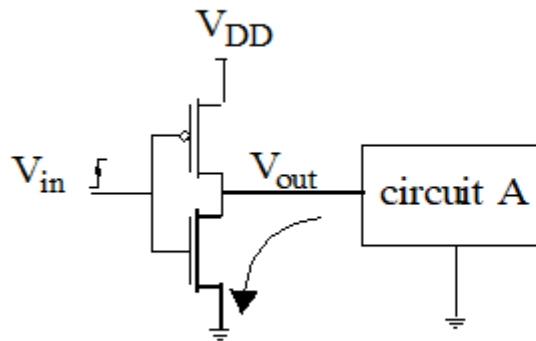
# Càrrega i descàrrega dels transistors

- **Càrrega:** quan l'entrada  $V_{in}$  conmuta de 1 a 0
  - El *pmos* està ON i el *cmos* està OFF
  - La capacitància equivalent (C) del circuit A es carrega



# Càrrega i descàrrega dels transistors

- **Descàrrega:** quan l'entrada  $V_{in}$  conmuta de 0 a 1
  - El *pmos* està OFF i el *cmos* està ON
  - La capacitància equivalent (C) del circuit A es descarrega





# Càlcul de la potència dinàmica

- Càrrega elèctrica  $q_i$  desplaçada en un cicle de càrrega i descàrrega, en 1 porta

$$q_i = C_i \cdot V$$

- $C_i$  = Capacitat equivalent del circuit A connectat a la sortida
- $V$  = Variació de potencial dels elements capacitius =  $V_{DD}$

# Càlcul de la potència dinàmica

- Càrrega elèctrica  $q_i$  desplaçada en un cicle de càrrega i descàrrega, en 1 porta

$$q_i = C_i \cdot V$$

- $C_i$  = Capacitat equivalent del circuit A connectat a la sortida
  - $V$  = Variació de potencial dels elements capacitius =  $V_{DD}$
- Corrent elèctric ( $I$ ) generat per totes les càrregues i descàrregues del circuit durant 1 cicle de rellotge ( $t_c$ )

$$I = \frac{\sum q_i}{t_c} = \sum (C_i \cdot V) \cdot f_{clock}$$

# Càlcul de la potència dinàmica

- Càrrega elèctrica  $q_i$  desplaçada en un cicle de càrrega i descàrrega, en 1 porta

$$q_i = C_i \cdot V$$

- $C_i$  = Capacitat equivalent del circuit A connectat a la sortida
  - $V$  = Variació de potencial dels elements capacitius =  $V_{DD}$
- Corrent elèctric ( $I$ ) generat per totes les càrregues i descàrregues del circuit durant 1 cicle de rellotge ( $t_c$ )

$$I = \frac{\sum q_i}{t_c} = \sum (C_i \cdot V) \cdot f_{clock}$$

- Però no totes les portes conmuten alhora
  - Sols una fracció  $\alpha$  (*factor d'activitat*)
  - Per tant, el sumatori  $\sum C_i$  és una fracció de la capacitància agregada ( $C$ ) de tot el circuit:  $\sum C_i = \alpha \cdot C$

# Càlcul de la potència dinàmica

- Càrrega elèctrica  $q_i$  desplaçada en un cicle de càrrega i descàrrega, en 1 porta

$$q_i = C_i \cdot V$$

- $C_i$  = Capacitat equivalent del circuit A connectat a la sortida
- $V$  = Variació de potencial dels elements capacitius =  $V_{DD}$
- Corrent elèctric ( $I$ ) generat per totes les càrregues i descàrregues del circuit durant 1 cicle de rellotge ( $t_c$ )

$$I = \frac{\sum q_i}{t_c} = \sum (C_i \cdot V) \cdot f_{clock}$$

- Però no totes les portes conmuten alhora
  - Sols una fracció  $\alpha$  (*factor d'activitat*)
  - Per tant, el sumatori  $\sum C_i$  és una fracció de la capacitància agregada ( $C$ ) de tot el circuit:  $\sum C_i = \alpha \cdot C$
- Considerant que  $P = I \cdot V$ , podem escriure

$$P_d = \alpha \cdot C \cdot V^2 \cdot f_{clock}$$

# Potència estàtica

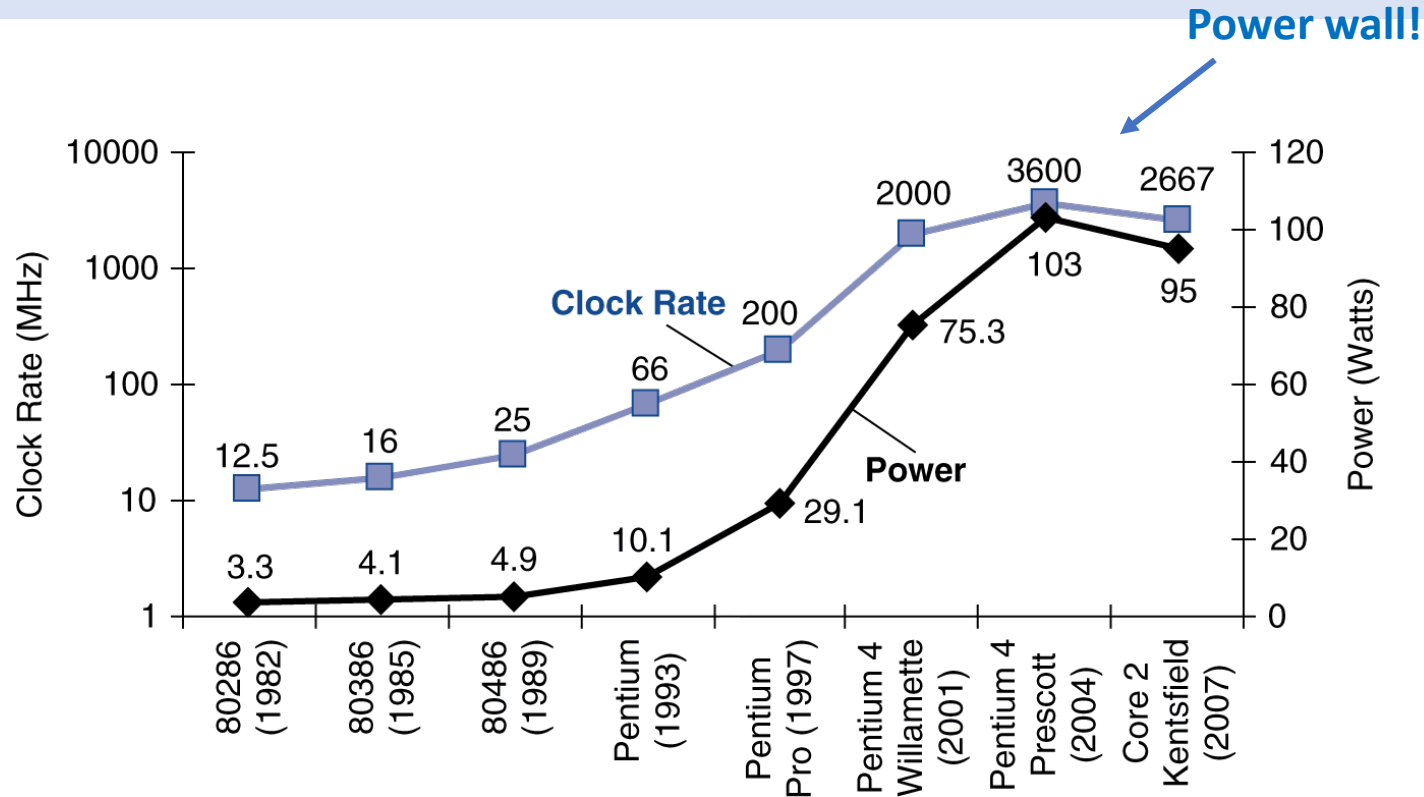
- La Potència estàtica és causada per corrents paràsits
  - Corrents de fuga ( $I_{leak}$ ) en els transistors en estat OFF
  - No depèn de les conmutacions: TOTS els transistors dissipen

$$P_s = I_{leak} \cdot V$$

- L'energia dissipada al cap d'un temps  $t$  és

$$E = P \cdot t$$

# Conseqüències de la miniaturització



$$P_d = \alpha \cdot C \cdot V^2 \cdot f_{clock}$$

x 30

5V → 1V

x 1000

Reduint  $V_{DD}$ , s'ha aconseguit augmentar  $f_{clock}$  x1000 amb un moderat augment x30 de  $P_d$

# Conseqüències de la miniaturització

- Reduint  $V_{DD}$ , s'ha aconseguit augmentar  $f_{clock}$  x1000 amb un moderat augment x30 de  $P_d$
- Problema (Power Wall):
  - La disminució de  $V_{DD}$ , comporta disminuir la tensió llindar  $V_{th}$ 
    - $V_{th}$  ha disminuït de 0,7V a 0,4V
  - La reducció de  $V_{th}$  causa un gran augment de  $I_{leak}$ , o sigui de  $P_s$
  - La miniaturització ja no permet millorar el rendiment augmentant  $f_{clock}$  sense que  $P$  rebassi l'umbral de tolerància tèrmica TDP (Thermal Design Power) del circuit (no podem reduir  $V_{DD}$ )
  - Suposa un límit a la millora de rendiment amb 1 sola CPU (veure gràfica)

# Com mitigar el power wall?

- Algunes estratègies
  - Clock gating
  - Power Gating
  - Dynamic Voltage and Frequency Scaling (DVFS)