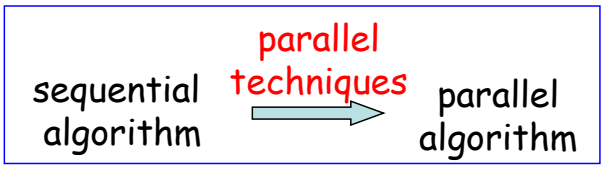
- 平行計算 (parallel computation) 是一門非常 精緻迷人的藝術
  - 開課目的不是要讓大家成為專家
  - 是為了讓大家 欣賞和感動

- prerequisite: algorithms
  (email your proofs in 1 week)



- 這是一門選修課
  - 希望上課聽懂就可以，沒有作業 *可能有加分作業 (自願)
  - 只有邏輯，~~沒有數學~~
    需要一些簡單的數學分析 (痛恨數學分析的，要趕快退選)
  - 有考試，需要花時間準備 (不喜歡考試的，要趕快退選)
  - 會花很多時間討論 (痛恨上課講話的，要趕快退選)
  - 缺課不得超過3次 (含請假) (不喜歡上課的，要趕快退選)
  - 不得使用手機或筆電 (每次扣5分，手機中毒的趕快退選)
  - 沒有預設進度，請踴躍發問

補充講義：eLearn 上有電子檔

algo 學分證明：

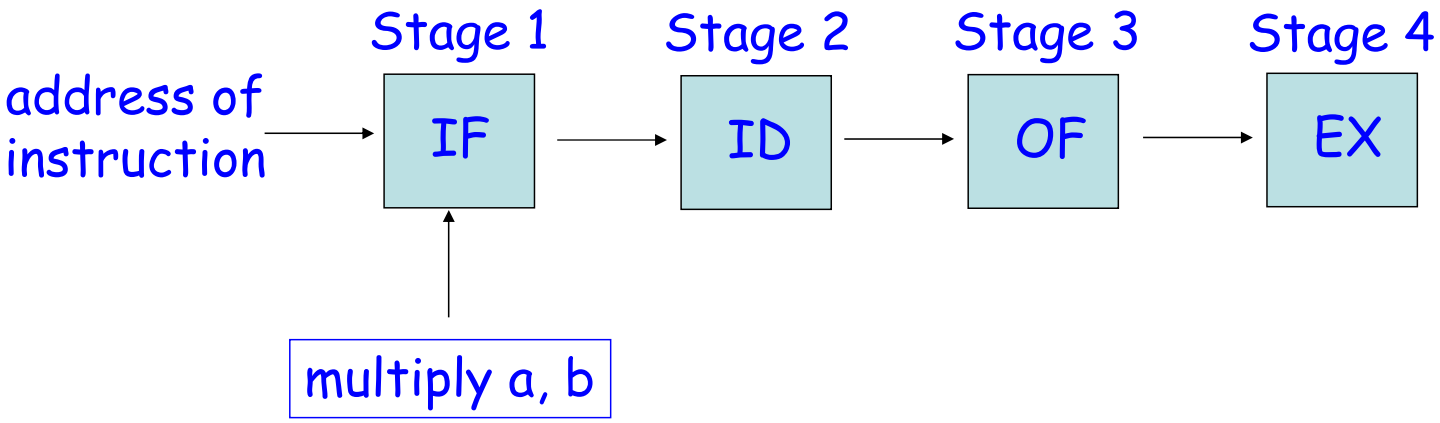
email (subject: 平行計算學分證明 王小明)
attachment (not a link; with a highlight)
no response (inform ones not sending)

加簽：

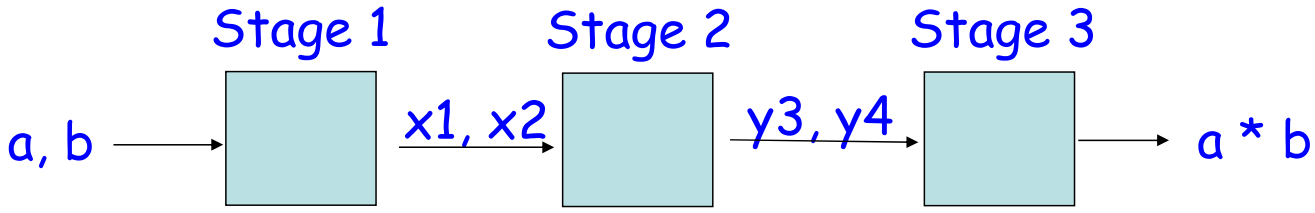
沒有加簽
(請透過校務資訊系統選課等待退選)

A pipelined computer

An instruction cycle
- instru. fetch
- instru. decoding
- operand fetch
- execution

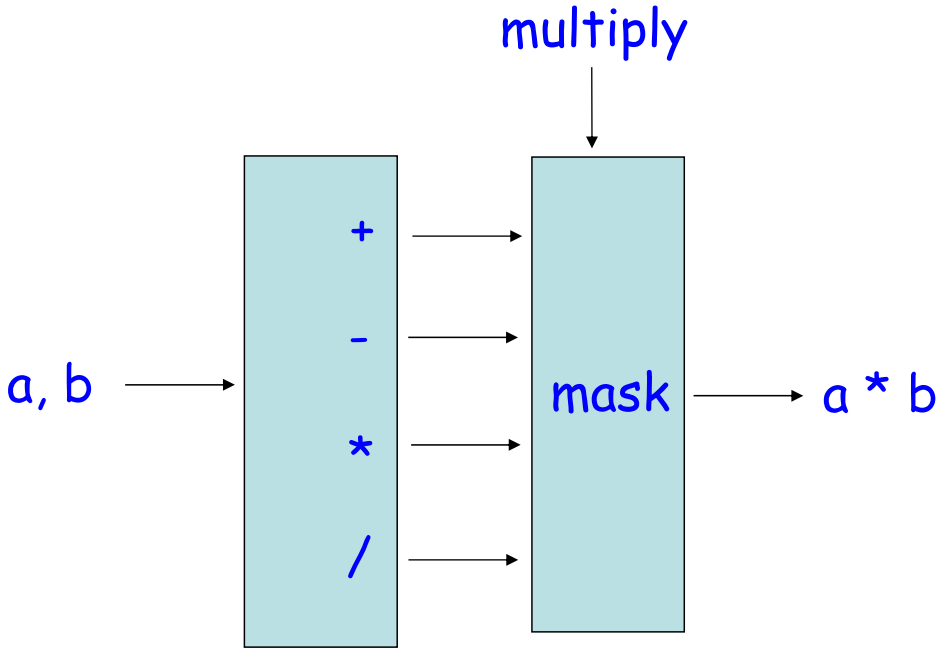


A pipelined multiplier (floating-point numbers)



Some ALU

Instruction: multiply a, b



Current PC

several CPUs: MIMD
(or several SISD)

DSP: vector + pipelining

Digital Signal Processor

SIMD

a technique

An Example of SIMD

Odd-Even Transposition Sort on a Linear Array

Odd iteration:

for each P_i, $1 \le i \le n$, **pardo**
if *i* is odd **then**
P_i Compare&Exchange with P_{i+1}

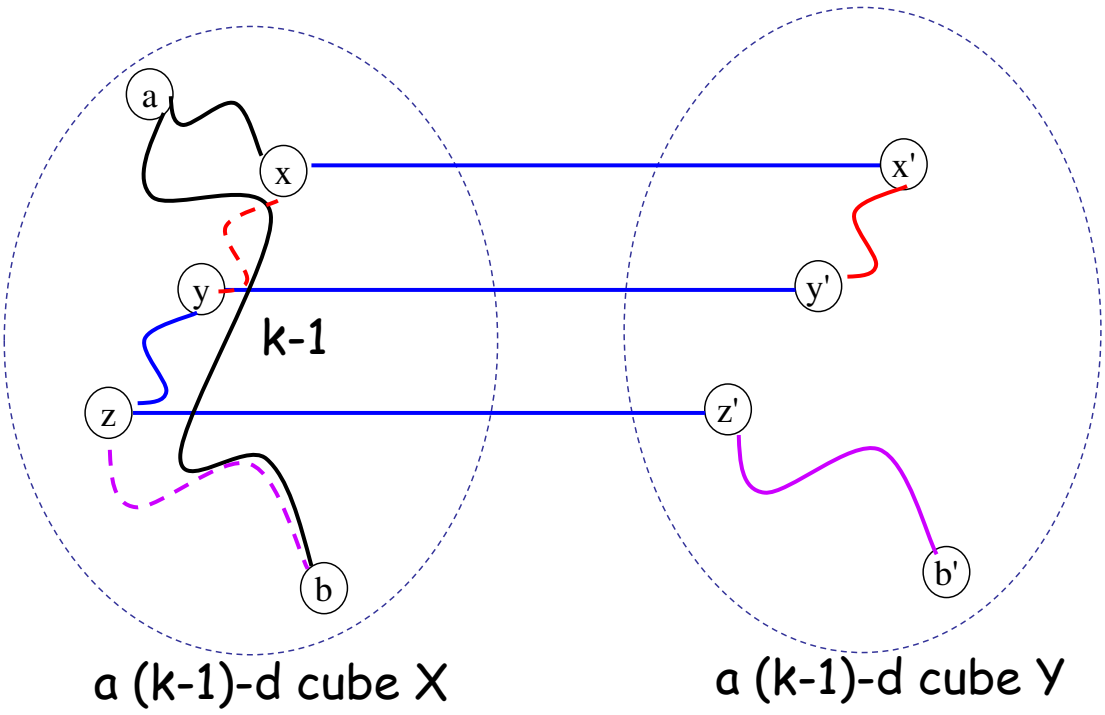
Even iteration:

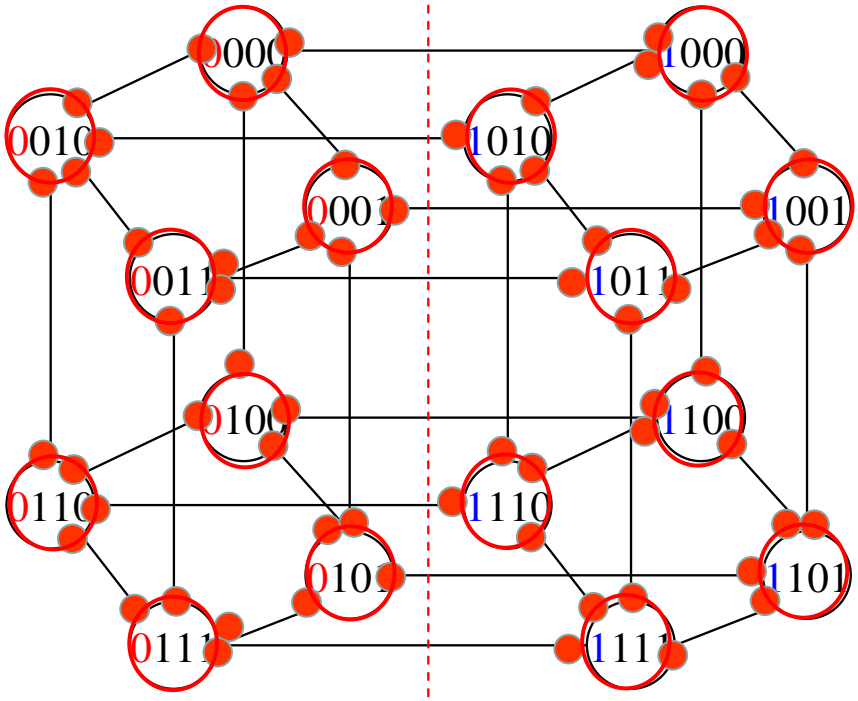
for each P_i, $1 \le i \le n$, **pardo**
if *i* is even **then**
P_i Compare&Exchange with P_{i+1}

Given d(a, b) in X is k-1, prove that d(a, b') = k
(assume that d(a, b') <= k is proved)

By contradiction, assume that d(a, b') < k

Then, in X, d(a, b) < k - 1, a contradiction.



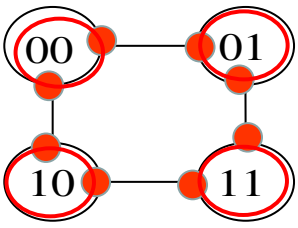


4-d CCC

4x2^4 processors

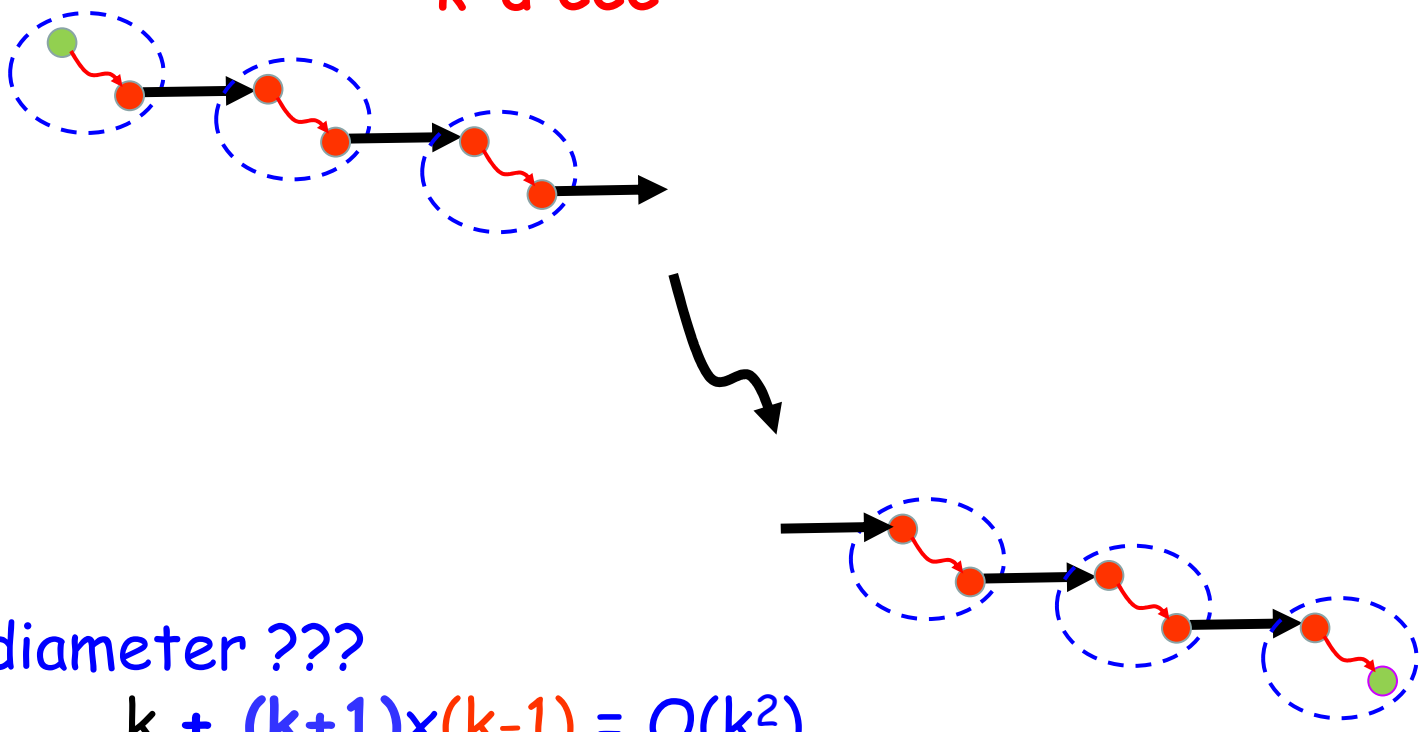
4-cube 2^4 processors

2-d CCC



2-cube

k-d CCC

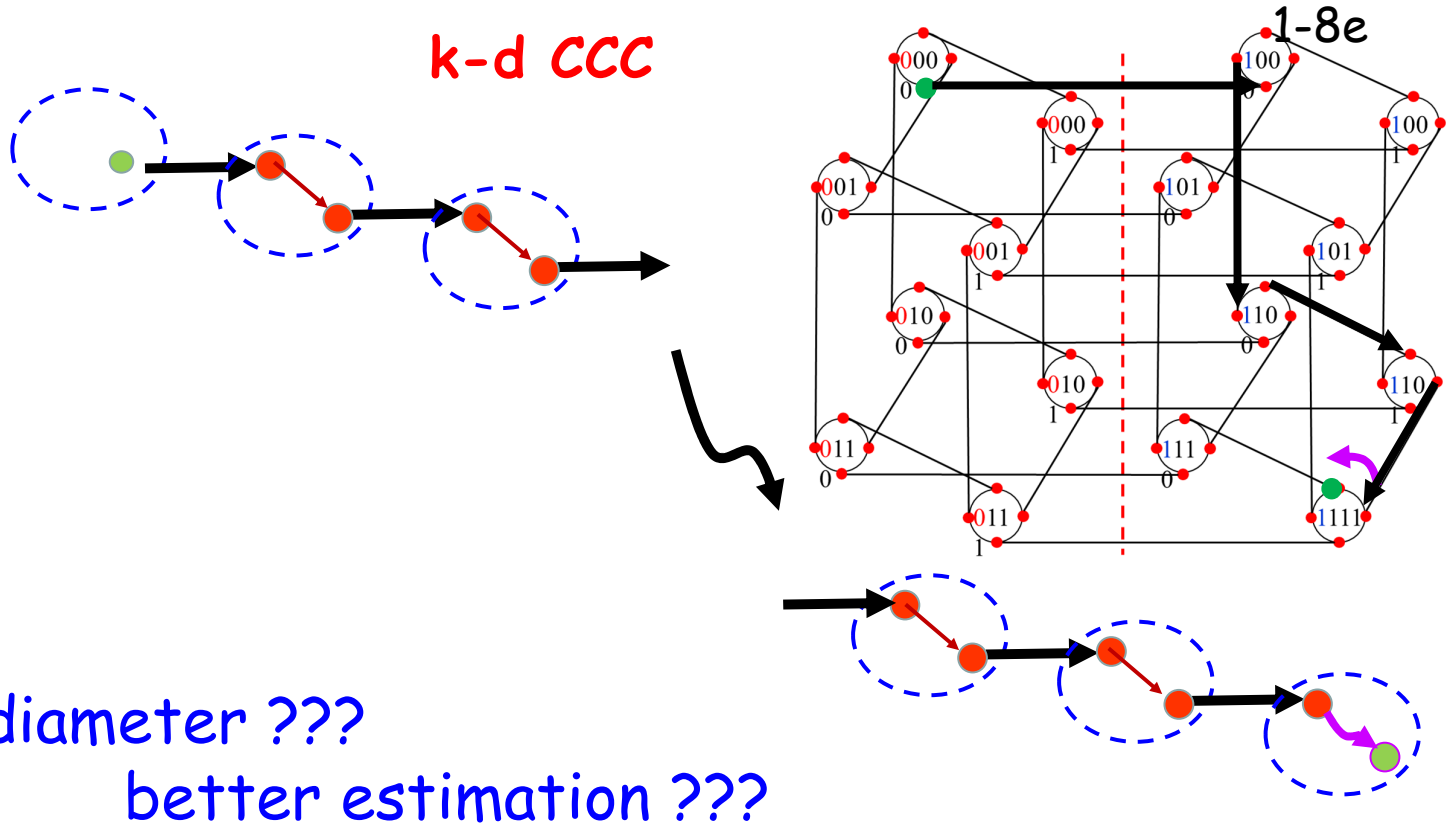


diameter ???

$k + (k+1)x(k-1) = O(k^2)$

(or $(k+1)x(k/2)$)

k-d CCC



diameter ???

better estimation ???

$k + (k-1) + \lfloor k/2 \rfloor$

(correct: $2k - 2 + \lfloor k/2 \rfloor$ for $k >= 4$)

Current PC:	several CPUs on a common bus MIMD (or several SISD) shared-memory
Super-Computer:	up to several 10^3 CPUs MIMD/SIMD intersection network

蓋 房 子 (每 位 工 人 每 天 工 錢 1 萬 元)

sequential:
p = 1, T_s = 100

parallel:
p = 5, T_p = 25

speedup = T_s / T_p = 100/25 = 4

cost = p * T_p = 5 * 25 = 125 (萬 元)

efficiency = T_s / cost = 100 / 125 = 0.8
= speedup / p = 4/5 = 0.8

蓋 房 子 (每 位 工 人 每 天 工 錢 1 萬 元)

sequential:
$p = 1, T_s = 100$

parallel:
$p = 10, T_p = 20$

$$\text{speedup} = T_s / T_p = 100/20 = 5$$

$$\text{cost} = p * T_p = 10 * 20 = 200 \text{ (萬 元)}$$

$$\text{efficiency} = T_s / \text{cost} = 100 / 200 = 0.5$$
$$= \text{speedup} / p = 5/10 = 0.5$$

Usage

	speedup T_s / T_p	cost $p * T_p$	efficiency T_s / cost speedup / p
experiment	√	√	√
algorithm	√ (θ or Ω)	√ (O or θ)	× (θ or Ω)

*experiment: in CPU time

*algorithms:

- usually, T_p first; then cost (or # of PEs)
- if speedup/efficiency are needed, in θ notation (suggested)

Odd-even transposition sort

sequential:
$p = 1$, $T_s = O(n \lg n)$

parallel:
$p = n$, $T_p = O(n)$

speedup = T_s / T_p = $\theta(\lg n)$

cost = $p * T_p$ = $\theta(n * n) = \theta(n^2)$

efficiency = T_s / cost = $\theta(n \lg n / n^2) = \theta(\lg n / n)$
= speedup / p = $\theta(\lg n / n)$

* Note: treat all O as θ

Another parallel sorting

sequential:
$p = 1$, $T_s = O(n \lg n)$

parallel:
$p = n^{0.5}$, $T_p = O(n^{0.5} \lg n)$

speedup = T_s / T_p = $\theta(n^{0.5})$ or $n^{0.5}$

cost = $p * T_p$ = $\theta(n \lg n)$ or $n \lg n$

efficiency = T_s / cost = $\theta(n \lg n / n \lg n) = \theta(1)$ or 1
= speedup / p = $\theta(n^{0.5}/n^{0.5}) = \theta(1)$ or 1

* Note: treat all O as θ (or simply remove the notation)