Trasportabilità
Lezione n°2

Algoritmi Avanzati

a.a.2011/2012

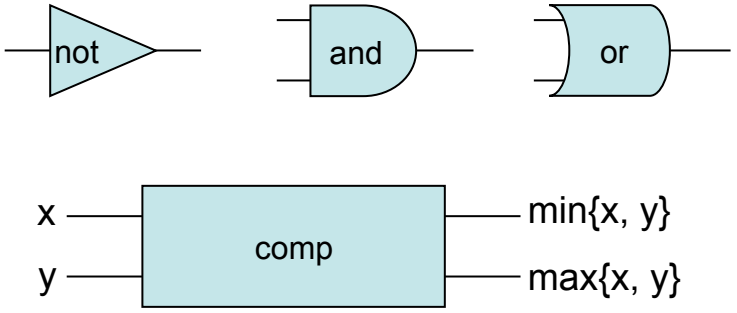
Prof.ssa Rossella Petreschi

Elemento combinatorio

Def: Un **elemento combinatorio** è un qualunque elemento di circuito che abbia un numero costante di input e output e che esegua una funzione ben precisa.

Esempi:

- Porte logiche
- Comparatore
- ecc…



Più elementi combinatori possono essere collegati tra di loro formando una **rete combinatoria** dove l'output di un elemento può essere l'input di uno o più altri elementi.

Rete combinatoria

Una rete combinatoria C può essere vista come un grafo diretto aciclico G_C avente un nodo per ciascun elemento combinatorio c e un arco diretto (c',c") se l'output di c' è input di c".
La **dimensione** di una rete combinatoria è il numero di nodi del grafo e la **profondità** è il diametro del grafo.

Il **fan-in** di un elemento c è il grado entrante del nodo c in G_C e corrisponde al numero di input dell'elemento.

Il **fan-out** di un elemento c è il grado uscente del nodo c in G_C.
È da notare che non necessariamente il numero di output di un elemento combinatorio è uguale al suo fan-out, infatti un elemento combinatorio con un solo filo di output può "servire" un numero qualunque di altri elementi.

Teorema di Brent

Teorema (CREW): *ogni algoritmo che lavora su una rete combinatoria di profondità d e dimensione n che sia a fan-in limitato, può essere simulato da un algoritmo che lavora su una PRAM CREW con p processori in O(n/p+d) tempo.*

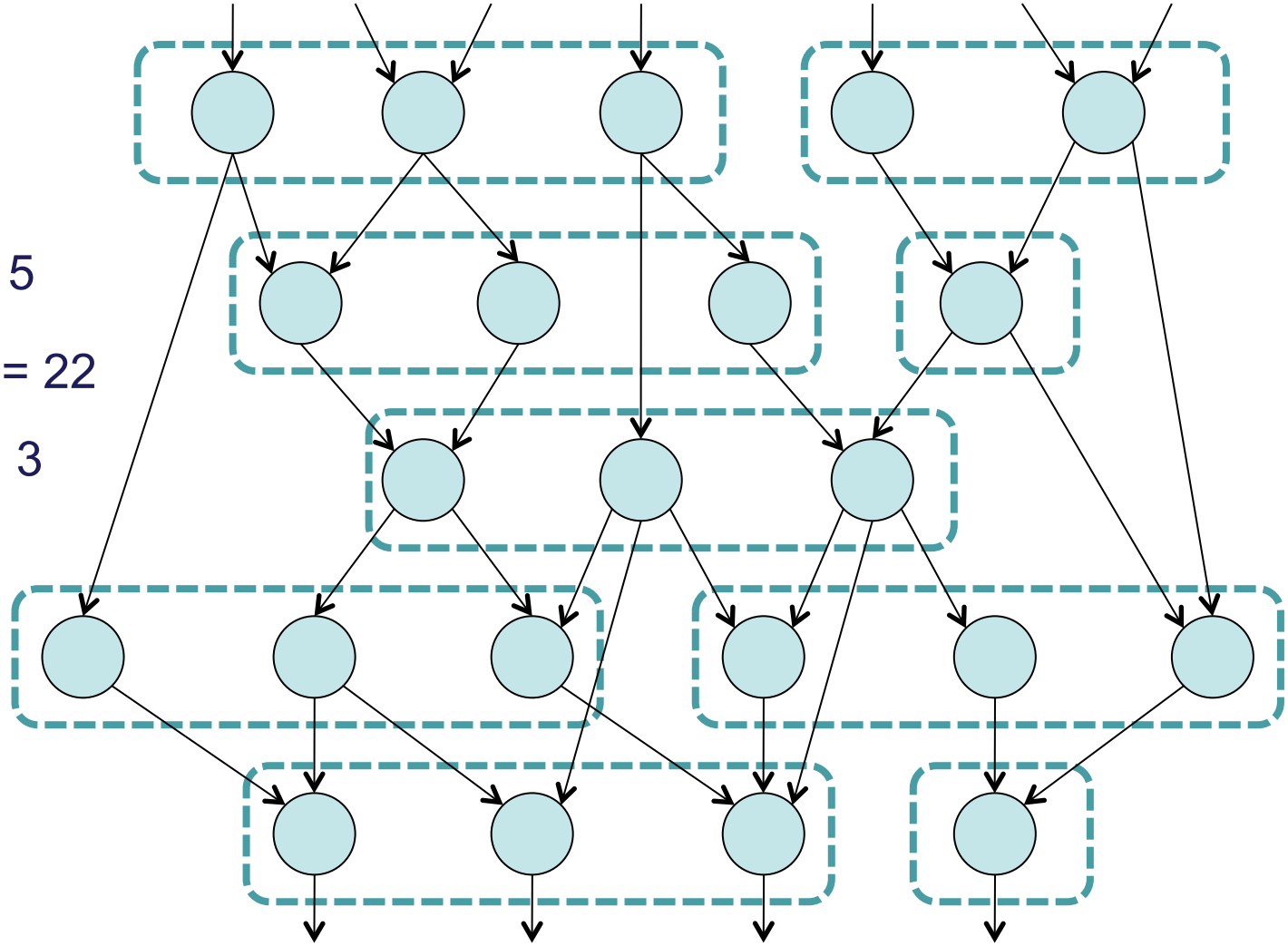
Teorema (EREW)*: ogni algoritmo che lavora su una rete combinatoria di profondità d e dimensione n che sia a fan-in e fan-out limitato, può essere simulato da un algoritmo che lavora su una PRAM EREW con p processori in O(n/p+d) tempo.*

Teorema di Brent - Esempio

Profondità: d = 5

Dimensione: n = 22

Processori: p = 3



Dimostrazione del teorema di Brent

•**Modelli di calcolo:**

- PRAM-EW con p processori $p_0, p_1, \ldots, p_{p-1}$.
- Rete combinatoria C di profondità d e dimensione $n = \sum n_i$, i=1..d, con n_i numero di elementi al livello i.

L'input del circuito combinatorio si considera disponibile nella memoria condivisa. Il fan-in limitato evita che la memoria condivisa non sia sufficiente a memorizzare i risultati intermedi del calcolo.

•**Idea della dimostrazione**: Ogni processore simula un componente del primo livello della rete e fornisce l'output che può essere immagazzinato nella memoria condivisa. Si ripete l'operazione per un numero di volte pari alla profondità del circuito utilizzando l'output di ogni livello come input del livello successivo. Nel caso in cui p sia minore del massimo numero di elementi combinatori presenti in uno stesso livello, si dovrà effettuare un numero di passi seriali proporzionale al rapporto tra il numero di elementi del livello i e p.

Modelli CREW ed EREW nel Teorema di Brent

• La complessità tiene conto della profondità del circuito e dei passi seriali necessari a simulare un livello.

• $T_p = \sum \lceil n_i / p \rceil \leq \sum (n_i / p + 1) = n / p + d$ con $i=1..d$

• Abbiamo detto che il processore che simula l'elemento combinatorio fornisce l'output che può essere immagazzinato nella memoria condivisa. Se più processori richiedono in input quello stesso valore si deve accettare che la P-RAM sia a lettura concorrente. Se si accetta l'ipotesi che anche il il fan-out sia limitato, si può considerare che in tempo costante l'output del circuito combinatorio sia direttamente copiato nell'input dei circuiti che lo richiedono. Questa limitazione permetterebbe la simulazione su una P-RAM a lettura esclusiva.

Trasportabilità fra P-RAM con diverso numero di processori

Teorema (p' < p)*: ogni algoritmo A che lavora in tempo parallelo O(t) su una PRAM con p processori può essere simulato da un algoritmo A' che lavora su una PRAM con p' processori (p' < p) in tempo O(t p / p').*

Dimostrazione*:* durante ognuno dei t passi dell'esecuzione di A, i p processori lavorano parallelamente in tempo O(1). Durante ogni passo della esecuzione di A', ciascuno dei p'<p processori eseguirà un blocco seriale di p/p' operazioni in tempo O(p/p'). Pertanto il tempo parallelo relativo alla esecuzione di A' sarà O(tp/p').

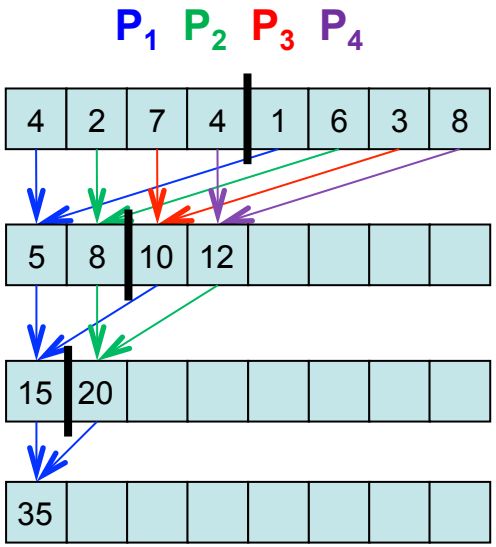
Il costo dei due algoritmi A e A' rimane pari a O(tp).

Numero di processori limitato (1)

Vogliamo sommare n numeri, con la tecnica della prima metà, avendo a disposizione un numero fissato a priori di p processori (p < n).

Ricordiamo come si lavora con n processori

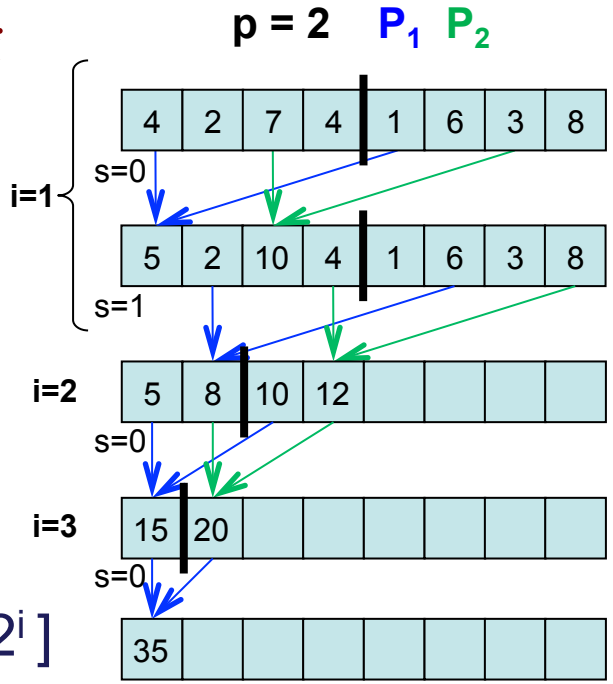
- **begin**
- **for** i = 1 **to** ⌈log n⌉ **do**
- **for** j = 0 **to** $n/2^i$ -1 **pardo**
- P_j: A[j] = A[j] + A[j+$n/2^i$]
- **return** A[0]
- **end**



Numero di processori limitato (2)

Vediamo cosa cambia con p <n processori

- **begin**
- **for** i = 1 **to** ⌈log n⌉ **do**
- **for** s = 0 **to** ⌈(n/2^i) / p⌉ -1 **do**
- **for** j = 0 **to** min(n/2^i, p) -1 **pardo**
- **if** (j+s*p < n/2^i) **then**
- P_j: A[j+s*p] = A[j+s*p] + A[j+s*p +n/2^i]
- **return** A[0]
- **end**



Trasportabilità degli algoritmi

Dati due differenti modelli di computazione M e M', si dice che un algoritmo A progettato per la computazione C sul modello M è trasportabile sul modello M' se l'algoritmo A' che computa C sul modello M' è ottenibile da A tramite l'applicazione di un insieme finito di regole fisse.

Il broadcast su PRAM è un primo esempio di trasportabilità di un algoritmo:
se su una PRAM CR tutti i processori vogliono leggere la stessa informazione, pagando un tempo O(logn) per il broadcast si può trasportare l'algoritmo su una PRAM ER.

Vediamo come affrontare la trasportabilità della scrittura concorrente da PRAM CW a PRAM EW.

Simulazione della scrittura concorrente: stessa informazione

Sia dato un algoritmo per PRAM CW con scrittura concorrente di valori identici. L'operazione compiuta da tutti i processori di scrivere nella stessa locazione R viene simulata su PRAM EW dal solo processore P_0, dopo aver controllato che tutti i valori a_i scritti dai rispettivi P_i siano uguali*. Questo controllo si effettua usando un vettore A con tutti i valori da scrivere e un vettore di booleani utilizzato per riportare le eventuali diversità nei valori di A:

```
      for j = 0 to n-1 pardo
Pj:       A[ j ] = aj; B[ j ] = true
      for i = 1 to log n do
          for j = 0 to (n/2^i -1) pardo
Pj:           if A[ j ] ≠ A[ j+ n/2^i ] or not B[ j ] or not B[ j+ n/2^i ] then
                  B[ j ] = false
P0:   if B[ 0 ] then R = A[ 0 ]
```

Il tempo parallelo richiesto è O(log n)

* Questo controllo su PRAM CW si suppone effettuato a livello hardware

Simulazione della scrittura concorrente: funzione dei valori

La simulazione di PRAM CW con scrittura del massimo valore (o della somma o di qualsiasi altra funzione commutativa f dei valori) viene eseguita su PRAM EW semplicemente utilizzando la tecnica della metà per il computo di f su un vettore A:

for j = 0 **to** n-1 **pardo**
P_j: A[j] = a_j
for i = 1 **to** log n **do**
for j = 0 **to** (n/2^i -1) **pardo**
P_j: A[j] = f (A[j], A[j+ n/2^i])
P_0: R = A[0]

Con lo stesso schema si può simulare anche la scrittura concorrente basata su priorità (ad esempio in funzione dell'indice del processore che vuole scrivere).

In entrambi i casi la simulazione richiede tempo parallelo logaritmico.