Gli Array Processor

prima parte

di Giuseppe Cardinale Ciccotti

Un Array Processor consiste di un certo numero di processori (PE) sotto la supervisione di un'Unità di Controllo (CU). In generale un Array Processor può assumere una delle due configurazioni mostrate in Figura 1. Come potete notare, le due architetture sono abbastanza simili tranne per la maniera in cui i moduli di memoria sono connessi ai processori. Lo schema I è strutturato con n PE sincronizzati, sotto il controllo di una CU. Ciascun PE è essenzialmente una unità di esecuzione, per esempio una ALU (Arithmetic Logic Unit), con un certo numero di registri e una memoria locale PEM_i per la memorizzazione dei dati distribuiti. La memoria connessa alla CU è necessaria alla memorizzazione del programma. La CU decodifica ciascuna istruzione e determina quali PE devono eseguirla. Le operazioni scalari e di controllo sono eseguite dalla CU stessa. La maggior efficienza si avrà nel caso in cui si debba eseguire un'operazione su un vettore di lunghezza pari al numero dei PE dell'Array Processor. In tal caso la CU attiva tutti i PE che contemporaneamente eseguono l'operazione specificata. Si ottiene così un parallelismo spaziale completo. Gli operandi andranno ovviamente caricati nelle PEM_i attraverso il data-bus prima di eseguire il programma. In una struttura di questo tipo tutti i PE funzionano in modo sincrono sotto il comando della CU; questo implica che la CU deve poter abilitare separatamente i vari PE per eseguire le operazioni necessarie. Ad ogni ciclo di istruzione viene eseguita un'operazione di «mascheramento» dei PE non partecipanti all'esecuzione dell'istruzione stessa. La CU controlla anche gli scambi di dati tra i vari PE, che avvengono attraverso la rete di interconnessione. Un'altra architettura possibile per un Array Processor è riportata in figura 1 nello schema II. Due aspetti differenziano questa configurazione dalla precedente. Il primo consiste nella maniera in cui le memorie locali PEM_i sono rimpiazzate da moduli di memoria condivisi dai PE attraverso una rete di «allineamento». Il secondo è che i moduli di memoria possono essere in numero diverso dai PE. La rete di allineamento consiste in un sistema di selezione di cammini tra i PE e i moduli di memoria.

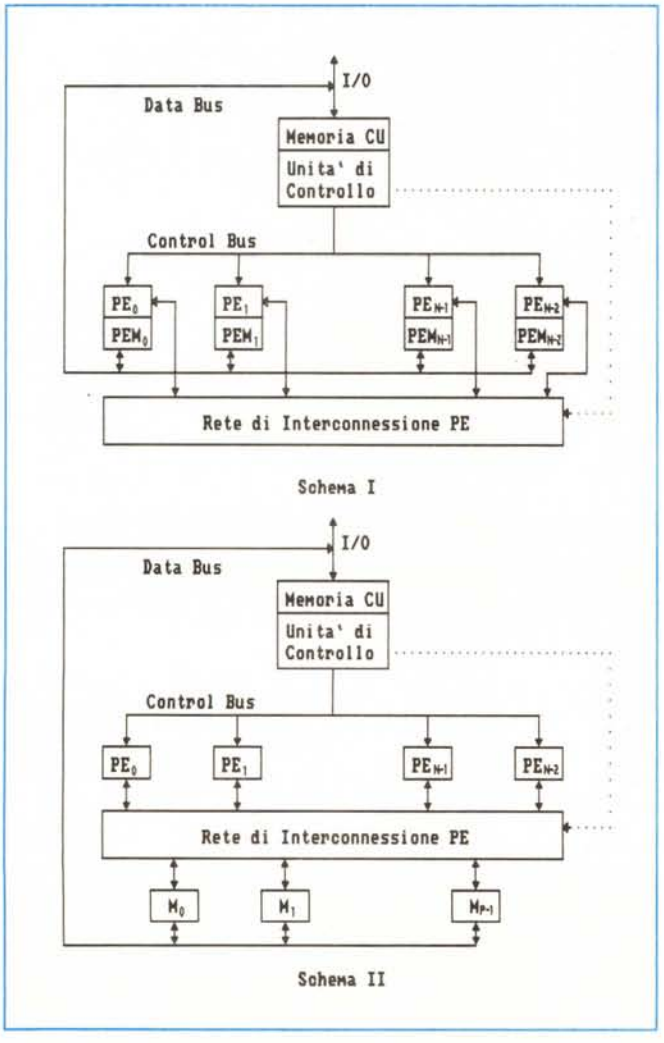


Figura 1 Architetture fondamentali di un Array Processor.

Tale rete dovrebbe preservare gli accessi alla memoria dai conflitti tipici delle configurazioni a memoria condivisa su bus. Come già abbiamo avuto modo di puntualizzare nei precedenti articoli, il collo di bottiglia del sistema è costituito dal canale di comunicazione tra memoria e i PE. Nel caso degli Array Processor la memoria è distribuita fra i moduli di memoria presenti e il canale di comunicazione è l'insieme dei canali disponibili. Più grande è il numero di canali, minori sono le probabilità di bloccare i dati in attesa di comunicazione, di conseguenza i tempi di calcolo si abbreviano. Nel caso in cui le comunicazioni avvenissero su un unico bus, ci ritrove-

remmo con una architettura Von-Neumann, con tutte le conseguenze che abbiamo già analizzato. Le strutture finora introdotte vengono dette SIMD acrostico di «Single Instruction Multiple Data», proprio a caratterizzarne il modo di funzionamento.

Allocazione e scambio dei dati

Analizziamo ora il meccanismo di scambio dei dati tra i PE dell'Array Processor. Il tempo impiegato nella comunicazione è di cruciale importanza ai fini della valutazione delle performance totali della macchina, in quanto esso è un «overhead», cioè un sovraccarico non previsto rispetto all'algoritmo in forma seriale, che abbassa lo speed-up. Consideriamo quindi due situazioni limite: quelle del minimo overhead di comunicazione, cioè quando tutti gli operandi sono sui PE dove verrà eseguita l'istruzione che li implica, e quella invece di massimo overhead, vale a dire quando tutti gli operandi sono su un solo PE. Nel primo caso ovviamente, l'overhead di comunicazione sarà pari a 0, nel secondo sarà uguale a no dove no è il numero di operandi, infatti gli operandi dovranno essere trasmessi serialmente agli altri PE con no comunicazioni. Appare evidente perciò che l'allocazione dei dati è un problema fondamentale nel progetto di algoritmi per Array Processor; influenza infatti lo scheduling e quindi la topologia della rete di connessione, in definitiva il progetto hardware dell'Array Processor. Per comprendere appieno il problema dello scambio dei dati, facciamo un esempio, analizzando i dettagli dell'esecuzione della seguente istruzione vettoriale in un Array Processor composto da N PE. Supponiamo quindi di voler calcolare le somme S(k) delle prime k componenti di un vettore $A=(A_0, A_1,..., A_{n-1})$ di ordine n. Posto k=0,..., n-1, dobbiamo eseguire le seguenti n somme

$S(k)=\Sigma_k A_k$ k=0...n-1.

Questa somma può essere computata con un'assegnazione e n-1 ricorsioni:

$S(0)=A_0$ $S(k)=S(k-1)+A_k$ k=1...n-1.

In figura 2 potete esaminare lo scheduling dell'algoritmo nel caso di n=8 per un Array Processor di N=8 PE. Tenete conto che le frecce tra i PE indicano la trasmissione di una copia del risultato calcolato nel passo precedente. Ad ogni passo dell'algoritmo, i PE individuati dalle frecce entranti eseguono una somma.

La notazione i-j sta ad indicare la somma $A_{i+1} + A_j$. C'è da notare come per ogni passo alcuni PE non eseguano addizioni, in particolare P_0 non ne fa mai. Tutto ciò richiama il meccanismo di mascheramento introdotto in precedenza; per ogni istruzione è necessario essere disabilitati (o in alternativa eseguire una NOP). Questo mascheramento è ovviamente dinamico e dipende ancora una volta, dal modo in cui sono allocati i dati oltre che dall'istruzione da eseguire. L'algoritmo dell'esempio è computato in $\log_2(n) = \log_2(8) = 3$ passi, mentre la versione seriale richiede 7 passi. Per quanto riguarda lo speed-up, otteniamo

$$\text{speed-up} = \frac{7ta}{3ta+3tc} = \frac{7}{3}\left(\frac{ta}{ta+tc}\right)$$

dove ta=tempo di addizione e tc=tempo di comunicazione

Da questa relazione si vede come l'overhead di comunicazione influisca sullo speed-up; si dovrà perciò necessariamente tenere basso tc rispetto ai tempi di istruzione e predisporre un accurato e rapido sistema di commutazione dei canali nella rete inter-PE, se si vuole rendere l'Array Processor flessibile. Per completezza valutiamo il caso in cui la rete inter-PE fosse costituita da un bus: lo speed-up ottenuto sarebbe pari a

$$\text{spped-up} = \frac{7ta}{3ta+17tc}$$

Comunicazioni interprocessore

L'esempio precedente mette in evidenza quanto sia importante la rete di comunicazione inter-PE; la realizzazione e la gestione di tale sezione è un aspetto cruciale del progetto di un Array Processor, tanto che le diverse scelte implementative danno luogo a varie classi di strutture. Il progetto della rete è caratterizzato da 4 parametri di progetto: Modo operativo, Strategia di controllo, Metodologia di commutazione e Tipologia della rete.

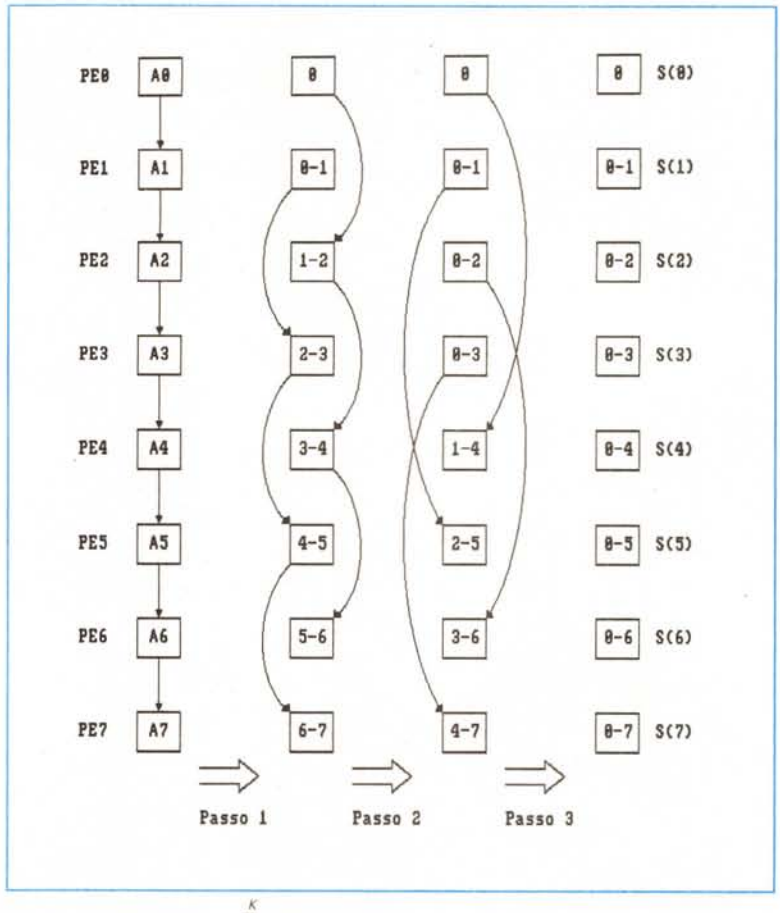


Figura 2 - Calcolo di $S(k) = \sum_{i:0}^{K} A_i$ *k=0,1,...,7 in un Array Processor.*

Modo operativo

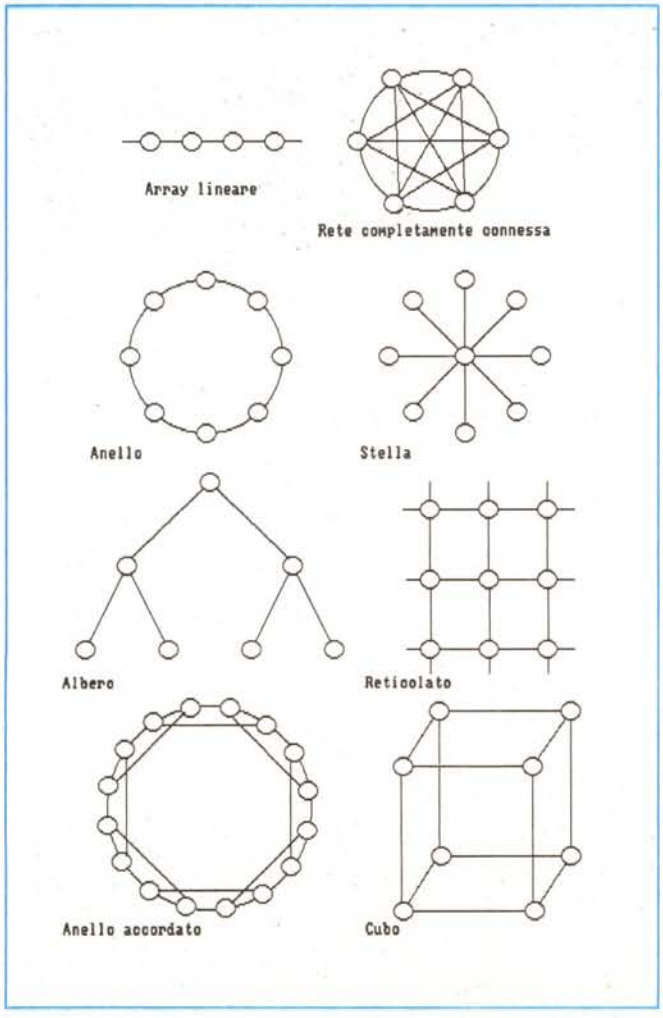
Il modo operativo identifica il tipo di funzionamento dell'Array Processor. Questo può essere sincrono, nel qual caso tutte le comunicazioni fra i PE avvengono negli stessi istanti, oppure può essere asincrono, se le comunicazioni inter-PE avvengono a richiesta del PE stesso. Tuttavia è sempre possibile che un Array Processor gestisca comunicazioni sia sincrone che asincrone.

Strategia di controllo

Abbiamo già osservato che per una maggior efficienza dell'Array Processor, la rete inter-PE deve poter essere riconfigurata, nel senso che un PE deve essere in qualche modo connesso a tutti gli altri PE nella rete. Ci sono due modi per soddisfare questo requisito: prevedere tutti i collegamenti possibili fra i PE oppure predisporre una serie di elementi di commutazione con cui connettere di volta in volta i PE coinvolti nella comunicazione. Il primo approccio è certo praticabile solo quando il numero dei PE sia piccolo oppure quando l'Array sia dedicato ed un'applicazione molto specializzata, quando sono determinati a priori tutti i collegamenti necessari. La seconda possibilità è l'unica possibile quando la rete è estesa e debba essere riconfigurata per eseguire diversi programmi. Gli elementi di commutazione, noti in letteratura come «switch», dovranno allora essere pilotati, come degli scambi di binari, dalla unità di controllo, perciò centralmente, oppure localmente dai PE stessi.

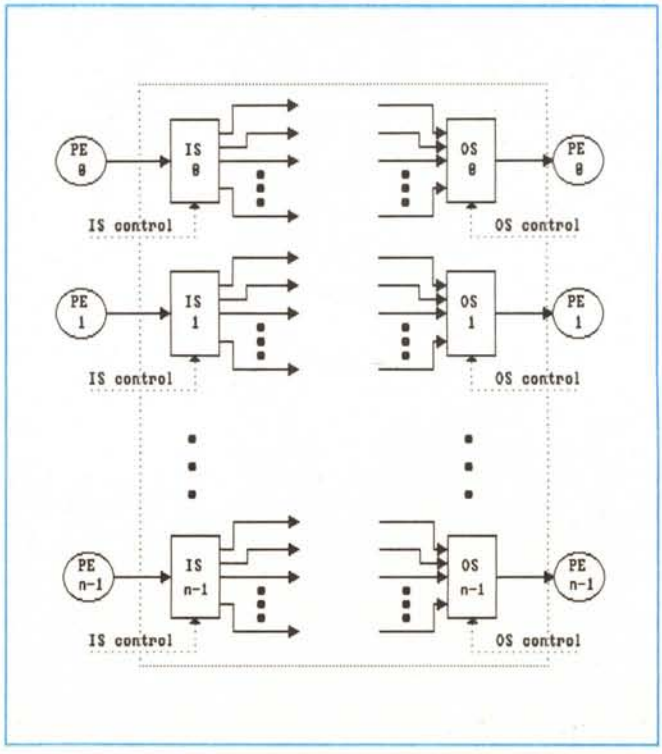
Metodologia di commutazione

Gli switch possono mettere in comunicazione due PE creando un circuito fisico o virtuale tra di essi. La comunicazione si dirà perciò a commutazione di circuito o a commutazione di pacchetto rispettivamente. Sono termini mutuati dalla telematica, perché da un punto di vista concettuale, la comunicazione fra due PE non differisce dal colloquio fra due sistemi. Nella commutazione di circuito, gli switch attivano i collegamenti in modo fisico, connettendo delle piste fra loro; viceversa nel secondo caso, il messaggio, sotto forma di «pacchetto» di bit da trasmettere, è instradato dallo switch sul percorso più opportuno verso il PE destinatario, attivando un solo collegamento switch-switch o switch-PE per volta, fino al PE ricevente, senza perciò stabilire un collegamento fisico tra i PE. In tale metodologia è necessario però, che il pacchetto rechi l'indirizzo del PE ricevente in modo tale che lo switch, detto «router» (instradatore), sappia dove mandarlo. Il vantaggio più evidente della comunicazione a commutazione di pacchetto, sta evidentemente nel fatto che la rete inter-PE può essere strutturata per connettere i PE in modo logico, e non fisico come invece richiede quella a commutazione di circuito. In quest'ultima situazione, un messaggio è l'unico proprietario del circuito fisico che connette i PE in comunicazione, quindi quante più connessioni della rete impegna il circuito, tanti meno circuiti possono essere contemporaneamente attivi e meno messaggi spediti. Per lo stesso motivo questa architettura di rete non è adatta quando le comunicazioni interprocessore sono lunghe. La comunicazione a commutazione di pac-



◄ *Figura 3 - Alcuni esempi di topologie in 1, 2 e 3 dimensioni.*

Figura 4 - Schema concettuale di una rete di interconnessione monostadio. Gli switch sono pilotati tramite il segnale di controllo. ▼



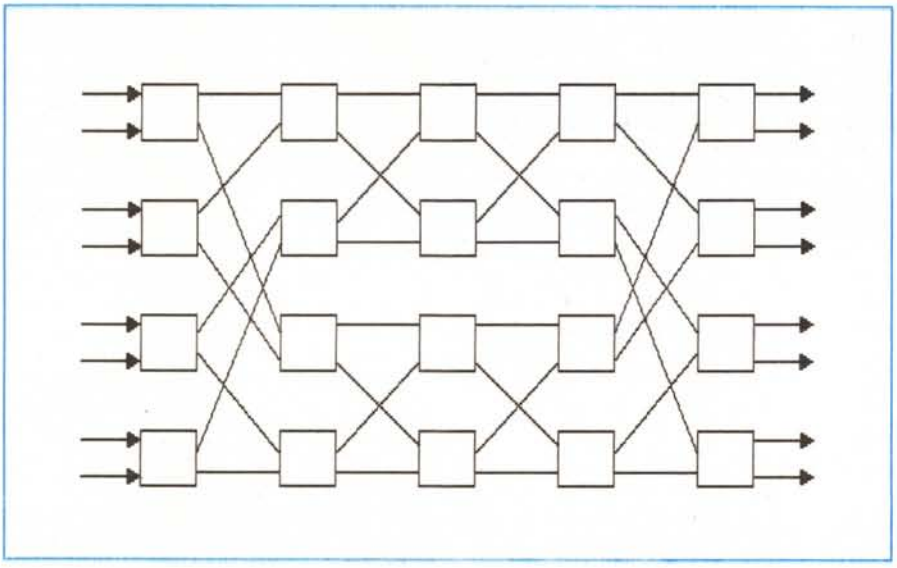


Figura 5 - Rete di Benes 8×8. Esempio di interconnessione multistadio.

chetto, viceversa si rivela efficiente proprio per comunicazioni lunghe e distanti; il pacchetto infatti, impegna solo un collegamento tra router e router per volta, ne consegue che sulla rete posso spedire, nel caso medio, più pacchetti. Per contro il router, a causa del lavoro di smistamento per cui è progettato, è in genere un circuito più complicato e costoso del semplice switch, soprattutto se si tiene conto che la sua efficienza di istradamento è determinante nel computo dell'efficienza totale dell'Array Processor. In casi limite (Connection Machine ad esempio), il router, un veloce processore dedicato corredato di buffer di memoria organizzati FIFO, è più complesso e costoso del PE.

Topologia della rete

Con il termine topologia della rete si intende la geometria dei collegamenti fra i vari PE. Al fine di classificare le varie configurazioni di rete da un punto di vista dimensionale, assimileremo i nodi della rete a vertici e i collegamenti a spigoli. Le topologie tendono ad essere regolari e possono essere raggruppate in topologie statiche e topologie dinamiche. In strutture appartenenti alla prima classe, i collegamenti tra due processori sono fissi e le relative piste non possono essere modificate per connessioni dirette con altri PE; i collegamenti della classe dinamica possono invece essere riconfigurati attraverso gli elementi di commutazione della rete.

Bibliografia:
Hwang K., Briggs F.
«Computer architecture and parallel processing»
Mc Graw-Hill, 1988.

Reti di interconnessione statiche e dinamiche

Analizziamo ora più da vicino le proprietà delle reti statiche e delle reti dinamiche.

Reti statiche

Le topologie delle reti statiche possono essere ordinate in base alle dimensioni richieste per il loro sviluppo. In figura 3 sono mostrati alcuni esempi di topologia unidimensionale come l'Array lineare, generalizzazione di una struttura Pipeline, di topologie bidimensionali come l'Anello, la Stella, l'Albero, il Reticolato, e di topologie tridimensionali tra cui il Cubo e l'Anello accordato, su ogni nodo del quale incidono 3 collegamenti. Esistono anche topologie di ordine superiore e di queste le più comuni sono sicuramente gli Ipercubi e le strutture completamente connesse.

Reti dinamiche

Le reti dinamiche hanno la particolare proprietà di potersi riconfigurare secondo le necessità delle comunicazioni inter-PE. I collegamenti fisici tra i dispositivi non sono allocati esclusivamente alla comunicazione fra due PE ma vengono di volta in volta inseriti in circuiti che collegano PE diversi. Per attivare tali circuitii sono necessari elementi di commutazione. L'organizzazione e la natura degli switch nella rete è diversa in relazione al fatto di adottare uno schema di commutazione a singolo stadio oppure multistadio. Nella prima ipotesi, in figura 4, ci sono solo due schiere di switch: i selettori di input (IS) e i selettori di output (OS); possiamo schematizzare questi dispositivi con dei multiplexer e demultiplexer connessi tra loro e attivati in modo opportuno. Per stabilire un collegamento, ad esempio tra PE_i e PE_j, bisogna pilotare IS_i col codice j e OS_j col codice i. In generale gli IS e OS non hanno cardinalità massima cioè non è possibile collegare tutti i PE direttamente, la rete è detta allora ricircolante in quanto un messaggio attraversa diversi switch prima di giungere al PE destinatario. Nel caso in cui sia possibile il massimo numero di connessioni, gli IS e OS sono detti «Crossbar» e possono essere pensati come una matrice di interruttori sugli incroci dei canali di input e di output dei PE dell'Array Processor. È chiaro che maggiore è il numero di connessioni permesse, minore è il numero di ricircolazioni attraverso la rete. Le reti multistadio sono formate da molteplici stadi di elementi di commutazione. Questa caratteristica implica che più di un cammino possa connettere due PE. La rete di Benes in figura 5, evidenzia questa possibilità. Gli switch devono essere in grado di scegliere il percorso opportuno in base alle caratteristiche della rete e ad altri parametri come il traffico sulla rete, ordine di priorità, etc. Supponiamo che il PE_1 voglia trasmettere al PE_4, gli switch della rete allora si predisporranno in maniera da assicurare il cammino minimo tra i due PE; può tuttavia accadere che un collegamento tra quelli selezionati sia già assegnato ad un altro canale, lo switch sceglie perciò un collegamento alternativo per evitare l'attesa della disponibilità del collegamento occupato. La rete di Benes 8×8 in particolare permette 4 diversi canali di collegamento tra ciascuna coppia di PE. Il controllo di questi switch, può essere centralizzato oppure il dispositivo stesso può elaborare un algoritmo per scegliere localmente il collegamento migliore.

Negli schermi di comunicazione a commutazione di pacchetto, la scelta del collegamento è generalmente effettuata da ciascun dispositivo attraversato, basandosi sulla conoscenza dell'indirizzo del PE ricevente e sul carico di traffico sulla rete.

Concludiamo qui la prima parte di questo articolo dedicato agli Array Processor, nella seconda analizzeremo da vicino le caratteristiche delle macchine realizzate secondo i concetti fin qui esposti; daremo inoltre un esempio di programma per un Array Processor.