Parallel Prefix Algorithms

Initial Condition: A vector which holds the number of items in each set in a sequence of sets.

Result:A vector with the number of items preceding each set in the sequence of sets.

Parallel Matrix Multiply

Initial Condition: Two conforming arrays holding the elements of the matrices.

Result: An array holding the result of matrix multiplication.

Parallel Prefix Algorithm

Sequential Algorithm

For i = 2, step 1 until N

v[i] = v[i-1] + v[i];

Even-Odd Parallel Prefix - Diagram

Figure 2-6 Odd/even construction for parallel prefix.

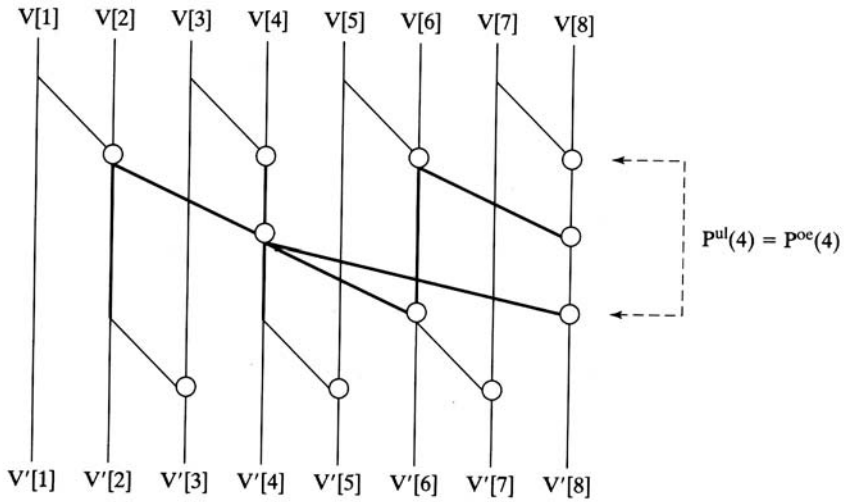


Figure 2-7 Recursive odd/even division for $N = 8$, $P^{oe}(8)$.

Sequential Program for Even/Odd Algorithm

```
level := 2;
while level <= N
        begin
        for i := level step level until N
                V[i] = V[i] + V[i-level/2];
        level := level * 2;
        end
If level = N then level := level/2;
while level > 1
        begin
        for i = level + level/2 step level until N
                V[i] = V[i] + V[i-level/2];
        level := level/2;
        end;
```

Parallel Algorithm for Odd/Even Prefix

```
Private level, j, i;
Shared N, NP, V[1:N];
level := 2;
while level <= N
        begin
        for i := level (j-i)*level step level*NP until N
                V[i] = V[i] + V[i-level/2];
        barrier
        level := level * 2;
        end
If level = N then level := level/2;
while level > 1
        begin
        for i = level + level/2 +(j-i)*level step level*NP until N
                V[i] = V[i] + V[i-level/2];
        barrier
        level := level/2;
        end;
```

SIMD/Vector Code

```
for i := step 1 until N-1
        begin
        C[i,j] :=0; (o<=j<=N-1)
        for k := 0 step 1 until N-1
        C[i,j] = C[i,j] + A[i,k]*B[k,j], (0<=j <= N-1);
end
```

```
private i, j, k;
shared A[N,N], B[N,N], C[N,N], N;
For j := 0 step 1 until N-2 fork DOCOL;
j:= N-1;
For i := 0 step 1 until N-1
begin
        C[i,j] := 0;
        for k := 0 step until N-1
        C[i,j] = := C[i,j] + A[i,k]* B[k,j];
end
```

More Parallel Matrix Multiply Algorithms

Scalar Formulation

$$a(i, j) = \sum_{k=1}^{n} b(i, k) * c(k, j) \text{ , for } i = 1 - - - n \text{ and } j = 1 - - - n$$

Three Indices - Six Permutations

i j k
j i k
k i j
k j i
i k j
j k i

i j k AND j i k

1. Compute all intermediate products in parallel - n^3 parallelism.

2. Compute each final result with an invocation of a tree sum.

```
cobegin i : <1 : n>
    cobegin j : <1 : n>
        cobegin k : <1 : n>
                temp (i, j, k) := b(i, k) * c(k, j)
        coend
        a(i, j) := innersum (temp (i, j, 1 : n))
              innersum = cascade partial sum
                              (tree algorithm)
    coend
coend
```

k i j AND k j i

(1) Compute all first terms of the inner products;
(2) Compute all second terms of the inner products and accumulate results with the first terms;
(3) Continue

```
cobegin i : <1 : n>
    cobegin j : <1 : n>
            temp (i, j) := 0
            for k := 1 to n do
                temp (i, j) := temp (i, j) + b(i, k) x c(k, j)
            enddo
                a(i, j) := temp(i, j)
    coend
coend
```

i k j AND j k i

1. Use n^2 processes.

2. Use broadcast communication.

Let matrix_of_b(i,j) = b(i,k) be a broadcast operator where each element of matrix_of_b(i,j) receives value of b(i,k).

```
cobegin i : <1 : n>
    cobegin j : <1 : n>
            temp (i, j) := 0
            for k := 1 to n do
                matrix_of_b(i, j) := b(i, k)
                matrix_of_c(i, j) := c(k, j)
                temp (i, j) := temp(i, j) +
                                matrix_of_b(i, j) x matrix_of_c(i, j)
            enddo
            a(i, j) := temp(i, j)
    coend
coend
```

Parallel Matrix Multiply

$$c(i, j) = \sum_{k=1}^{n} a(i, k)\, b(k, j)$$

let n = 2 then

$$c(1, 1) = a(1, 1)\, b(1, 1) + a(1, 2)\, b(2, 1)$$
$$c(1, 2) = a(1, 1)\, b(1, 2) + a(1, 2)\, b(2, 2)$$
$$c(2, 1) = a(2, 1)\, b(1, 1) + a(2, 2)\, b(2, 1)$$
$$c(2, 2) = a(2, 1)\, b(1, 2) + a(2, 2)\, b(2, 2)$$

$$\begin{pmatrix} a(1, 1) & a(1, 2) \\ a(2, 2) & a(2, 1) \end{pmatrix} \otimes \begin{pmatrix} b(1, 1) & b(2, 2) \\ b(2, 1) & b(1, 2) \end{pmatrix}$$

$$c(1) = \begin{pmatrix} a(1, 1)\, b(1, 1) & a(1, 2)\, b(2, 2) \\ a(2, 2)\, b(2, 1) & a(2, 1)\, b(1, 2) \end{pmatrix}$$

$$\begin{pmatrix} a(1, 2) & a(1, 1) \\ a(2, 1) & a(2, 2) \end{pmatrix} \otimes \begin{pmatrix} b(2, 1) & b(1, 2) \\ b(1, 1) & b(2, 2) \end{pmatrix}$$

$$c(2) = \begin{pmatrix} a(1, 2)\, b(2, 1) & a(1, 1)\, b(1, 2) \\ a(2, 1)\, b(1, 1) & a(2, 2)\, b(2, 2) \end{pmatrix}$$

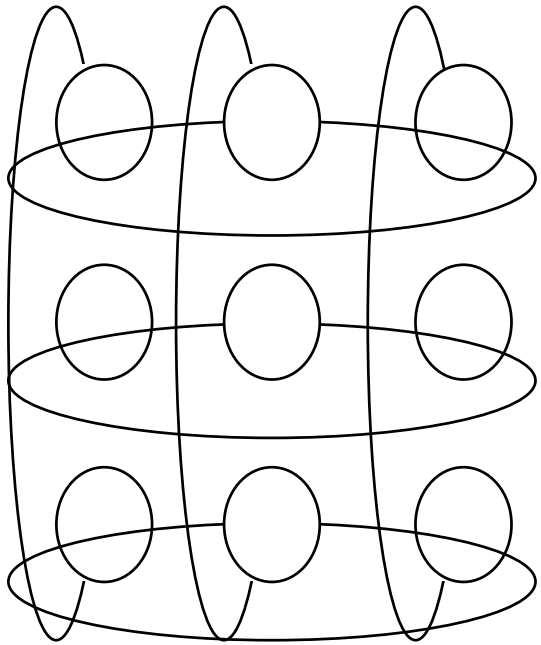
$$c = c(1) + c(2)$$

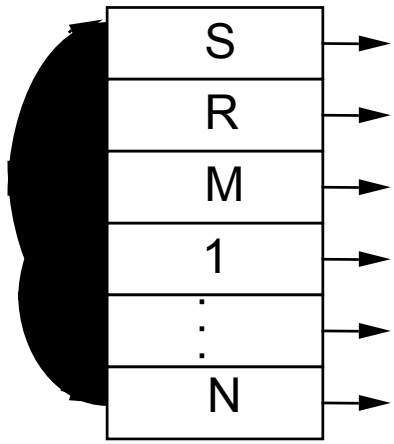
Matrix multiplication using element by element multiplication and matrix addition.

The operator $\otimes$ is element by element multiplication

Matrix Multiply on an array processor with end-around nearest neighbor connections.

(Cannon's Algorithm)





SHIFT PLANE

REGISTER PLANE

MEMORY PLANE

Memory planes Connected to S, R, M planes

Memory for each cell

Rotation Algorithm

```
for k := 1 to n do
    cobegin i : <1 : n>
        cobegin j : <1 : n>
            if i > k then b(i, j) := b(i, j Rot 1)
            if j > k then c(i, j) := c(i Rot 1, j)
        coend
    coend
enddo
```

Rot is a column (row) operation which shifts columns up (or rows to right)

Parallel Matrix Multiply

```
cobegin i : <1 : n>
    cobegin j : <1 : n>
        temp (i, j) := b(i, j) x c(i, j)
        for k := 1 to n do
        begin
            b(i, j) := b(i, j Rot 1)
            c(i, j) := c(i Rot 1, j)
            temp(i, j) := temp(i, j) + b(i, j) x c(i, j)
        enddo
        a(i, j) := temp(i, j)
    coend
coend
```

What have we done?

Pre-skewed a , pre-skewed b
By rows by cols

Rotate jth row rotate jth column
By j-1 positions by j-1 positions
To left to top

Then

1) **elementwise multiply**

2) **rotate a left by rows**
   Rotate b top by columns

3) **elementwise multiply and sum**

4) **repeat 2 and 3**

Block Matrix Multiply

$A = (N \times N)$, $B = (N \times N)$

$N = 2k$

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \qquad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

$A_{ij} = B_{ij} = (k \times k)$

$$C = A \times B = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

$$= \begin{pmatrix} A_{11} B_{11} + A_{12} B_{21} & A_{11} B_{12} + A_{12} B_{22} \\ A_{21} B_{11} + A_{22} B_{21} & A_{21} B_{12} + A_{22} B_{22} \end{pmatrix}$$

General Case

$N = pk$

$$C_{ij} = \sum_{\ell=1}^{p} A_{i\ell} B_{\ell j}$$

Parallel Matrix Multiply

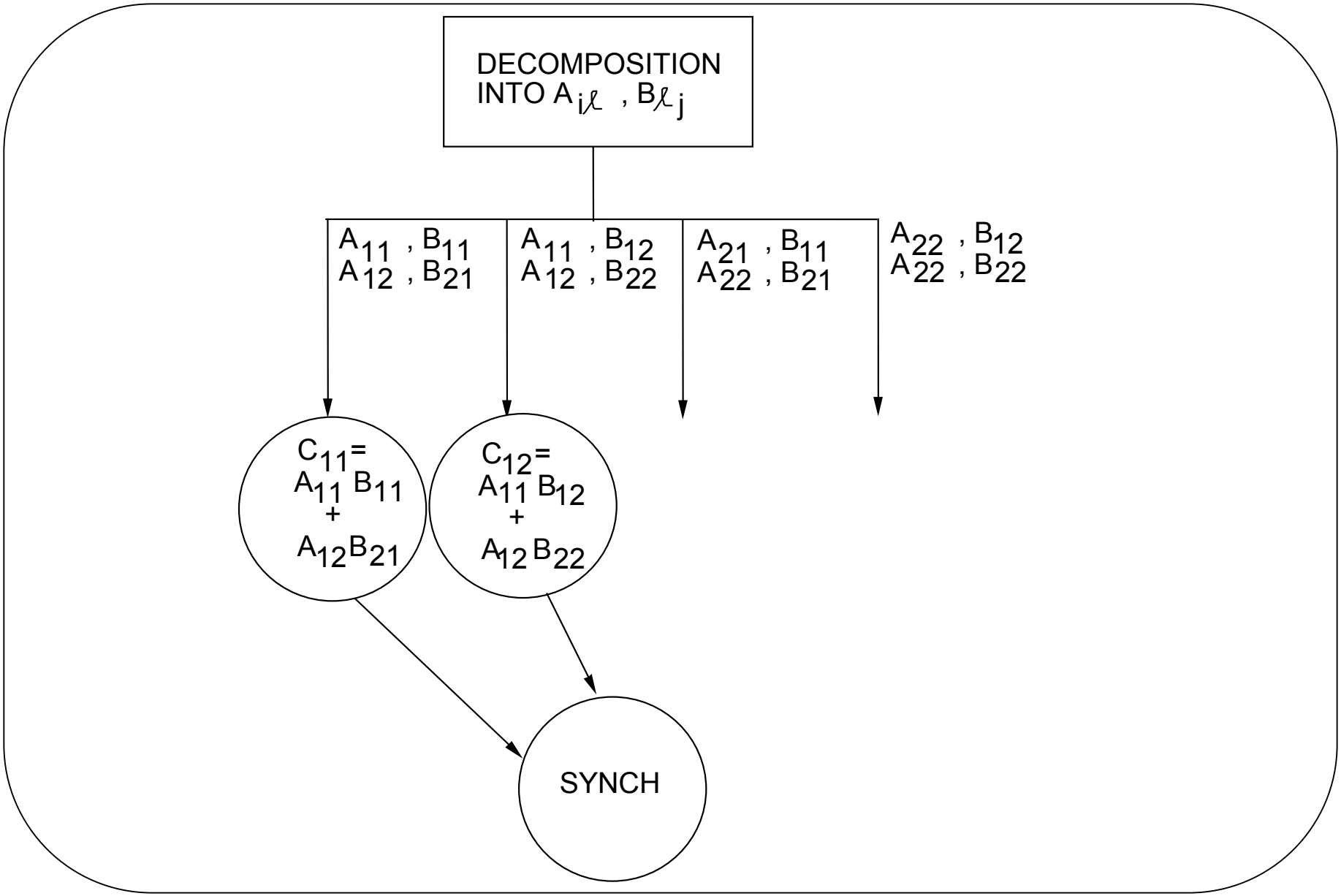
$$A_{11}B_{11} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

$$A_{12}B_{21} = \begin{pmatrix} a_{13} & a_{14} \\ a_{23} & a_{24} \end{pmatrix} \begin{pmatrix} b_{31} & b_{32} \\ b_{41} & b_{42} \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}$$

$$\begin{pmatrix} a_{13} & a_{14} \\ a_{23} & a_{24} \end{pmatrix} \begin{pmatrix} b_{31} & b_{32} \\ b_{41} & b_{42} \end{pmatrix} = \begin{pmatrix} a_{13}b_{31} + a_{14}b_{41} & a_{13}b_{32} + a_{14}b_{42} \\ a_{23}b_{31} + a_{24}b_{41} & a_{23}b_{32} + a_{24}b_{42} \end{pmatrix}$$

Parallel Matrix Multiply



Parallel Matrix Multiply