Parallel Random Access Machine (PRAM)

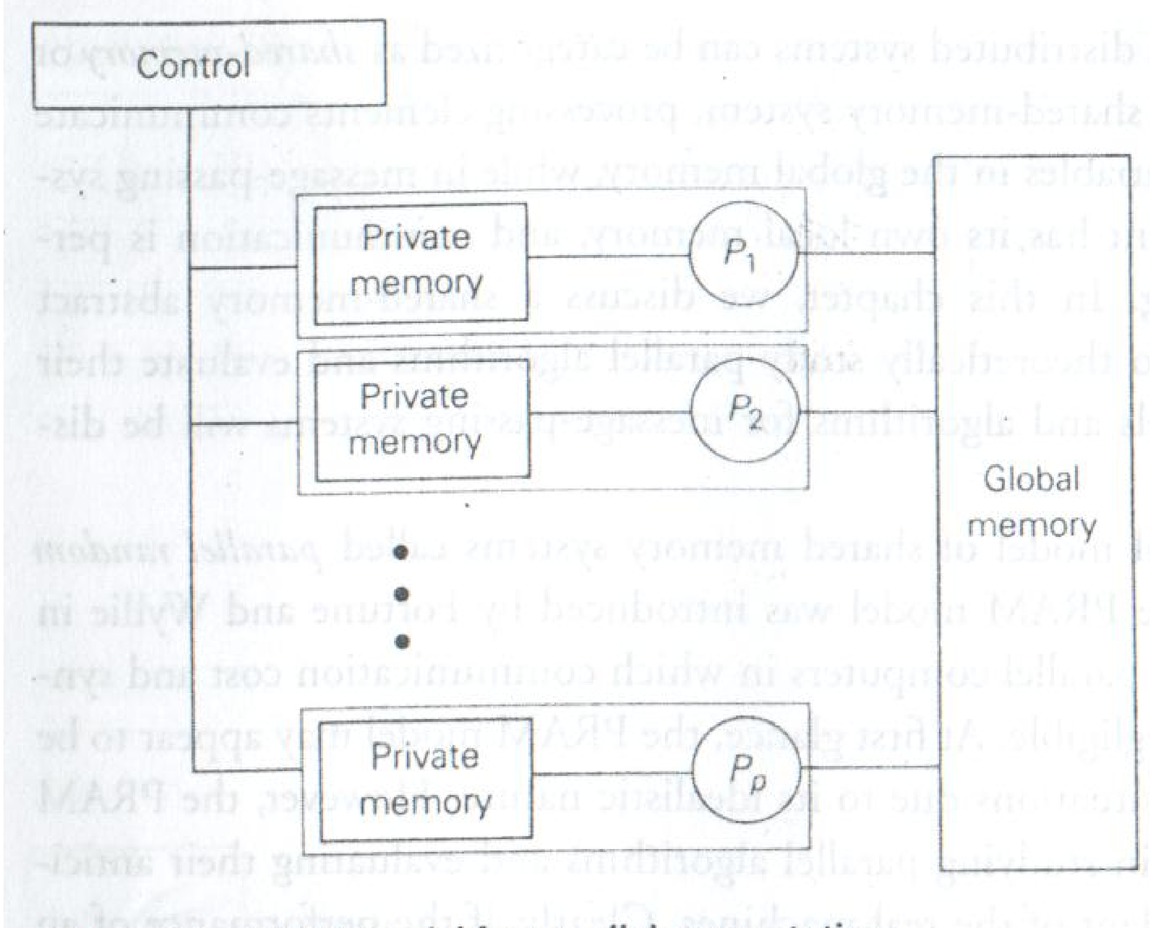


Figure 4.1 PRAM model for parallel computations

Read/Write Modes

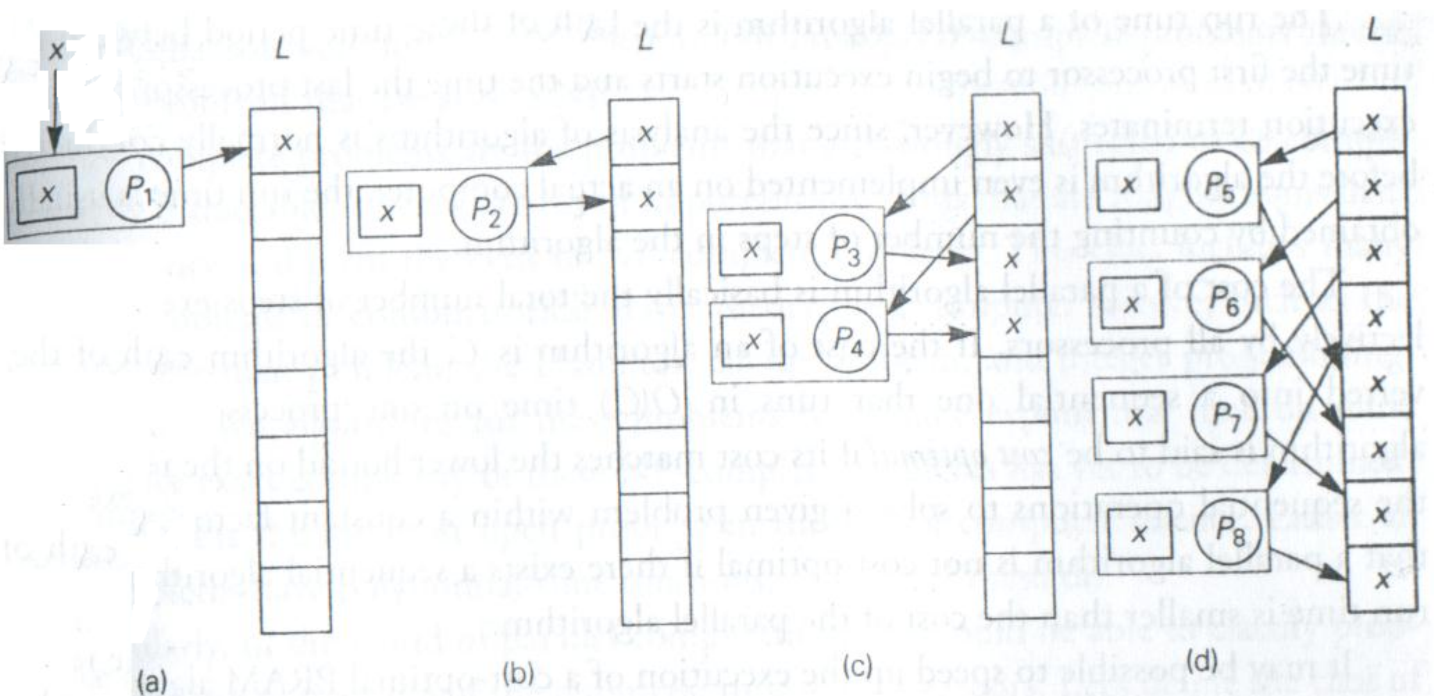
1. Exclusive Read (ER)
2. Exclusive Write (EW)
3. Concurrent Read (CR)
4. Concurrent Write (CR)

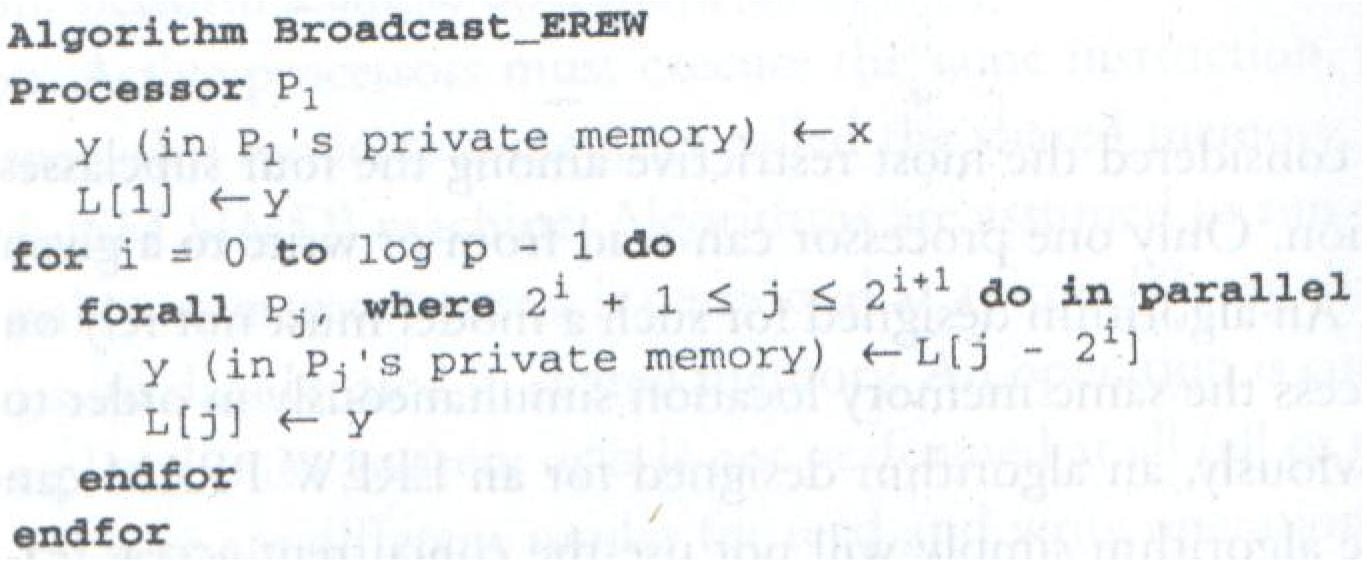
   Common, Arbitrary, Minimum (P-index), Reduction (sum/min/max)

Sub Classes of PRAM

EREW PRAM
ERCW PRAM
CREW PRAM
CRCW PRAM

Multiple Accesses on EREW





```
Algorithm Broadcast_EREW
Processor $P_1$
  y (in $P_1$'s private memory) ← x
  L[1] ← y
for i = 0 to log p - 1 do
  forall $P_j$, where $2^i + 1 \le j \le 2^{i+1}$ do in parallel
    y (in $P_j$'s private memory) ← L[j - $2^i$]
    L[j] ← y
  endfor
endfor
```

Complexity Analysis

Run Time T(n)
Number of Processors P(n)
Cost C(n) = T(n) * P(n)

Computing Sum of an ARRAY on EREW PRAM

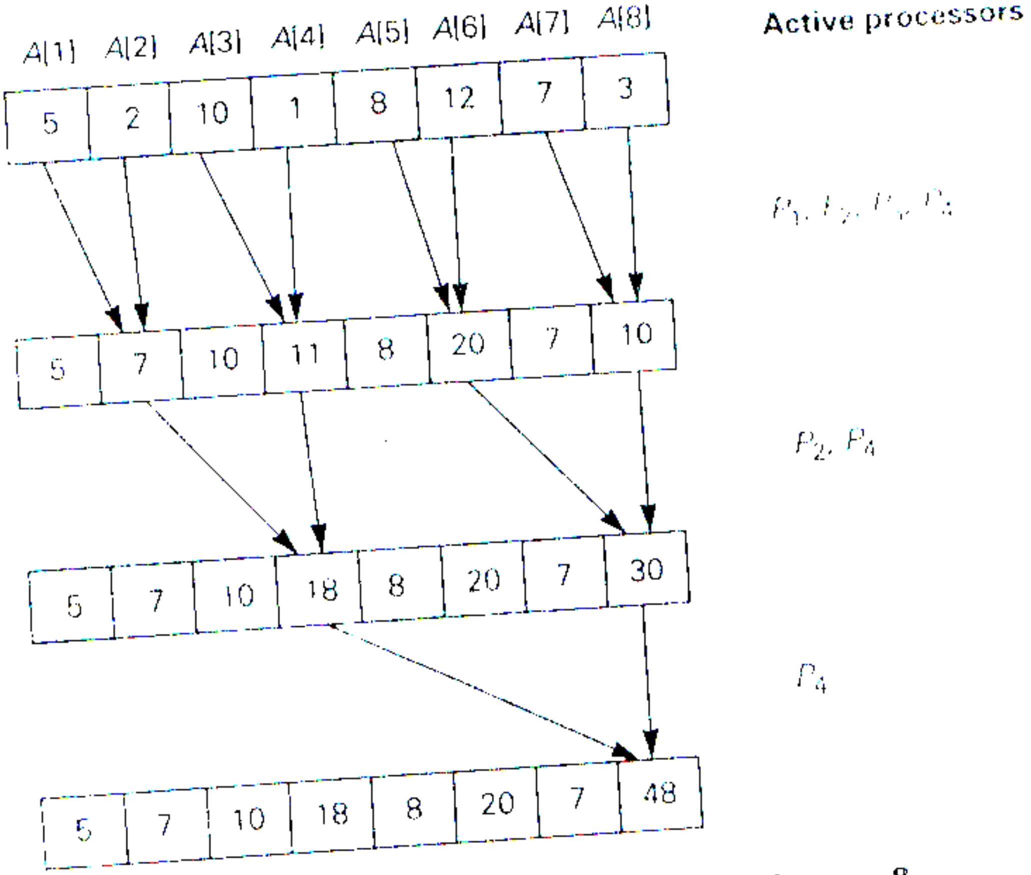
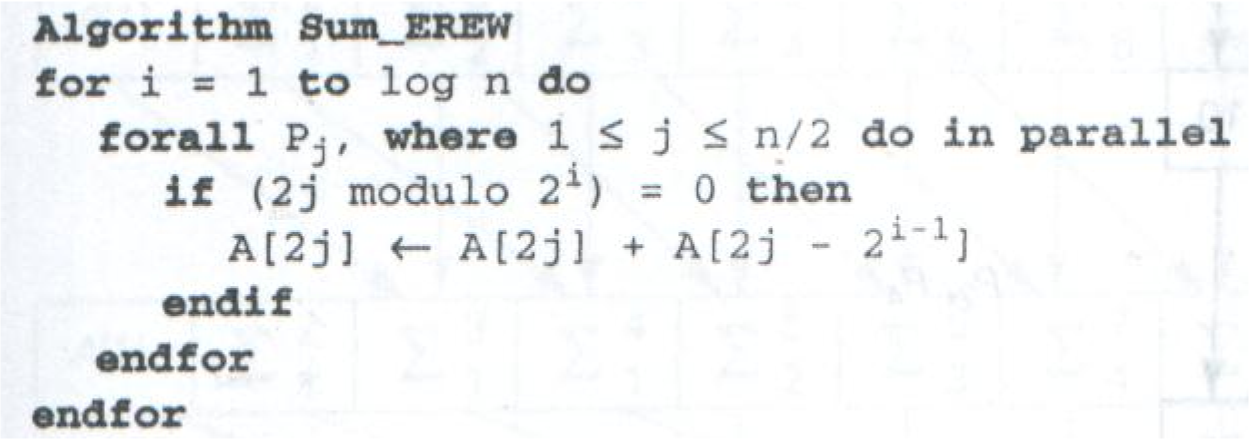


Figure 4.4 Example of Algorithm Sum_EREW when $n = 8$



```
Algorithm Sum_EREW
for i = 1 to log n do
  forall $P_j$, where $1 \le j \le n/2$ do in parallel
    if ($2j$ modulo $2^i$) = 0 then
      $A[2j] \leftarrow A[2j] + A[2j - 2^{i-1}]$
    endif
  endfor
endfor
```

Complexity Analysis

Run Time T(n)
Number of Processors P(n)
Cost C(n) = T(n) * P(n)

Computing all Partial Sum

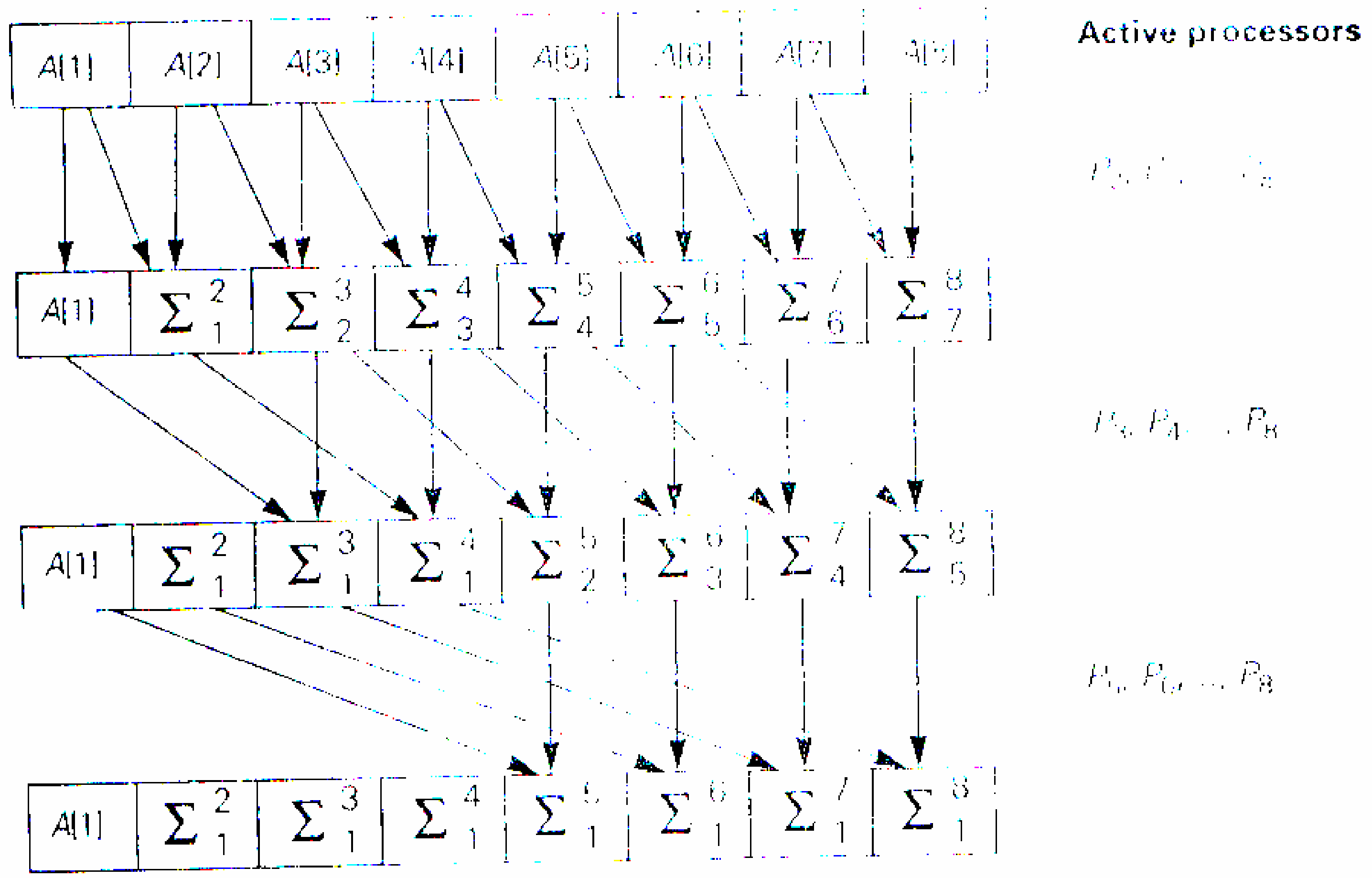
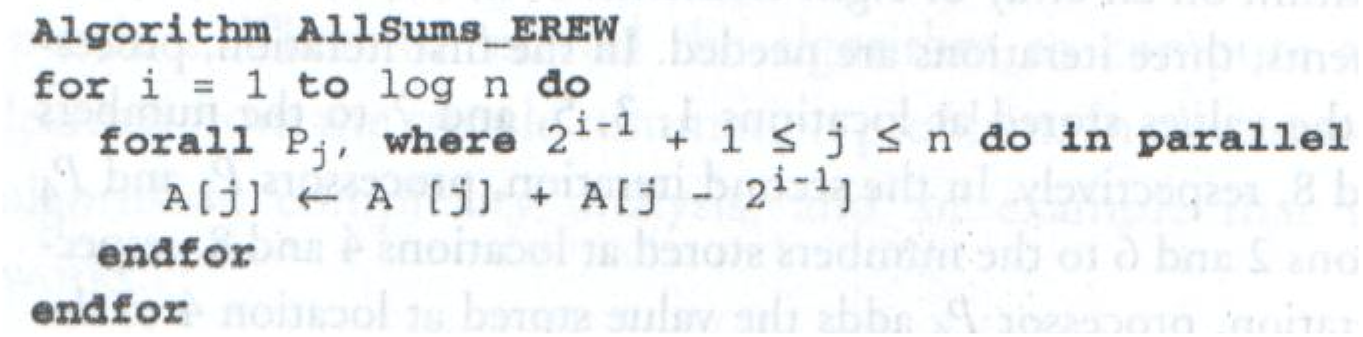

Active processors
A[1] A[2] A[3] A[4] A[5] A[6] A[7] A[8]
$P_2, P_3, \ldots, P_8$
A[1] Σ_1^2 Σ_2^3 Σ_3^4 Σ_4^5 Σ_5^6 Σ_6^7 Σ_7^8
$P_3, P_4, \ldots, P_8$
A[1] Σ_1^2 Σ_1^3 Σ_1^4 Σ_2^5 Σ_3^6 Σ_4^7 Σ_5^8
$P_5, P_6, \ldots, P_8$
A[1] Σ_1^2 Σ_1^3 Σ_1^4 Σ_1^5 Σ_1^6 Σ_1^7 Σ_1^8

Figure 4.5 Computing partial sums of an array of eight elements

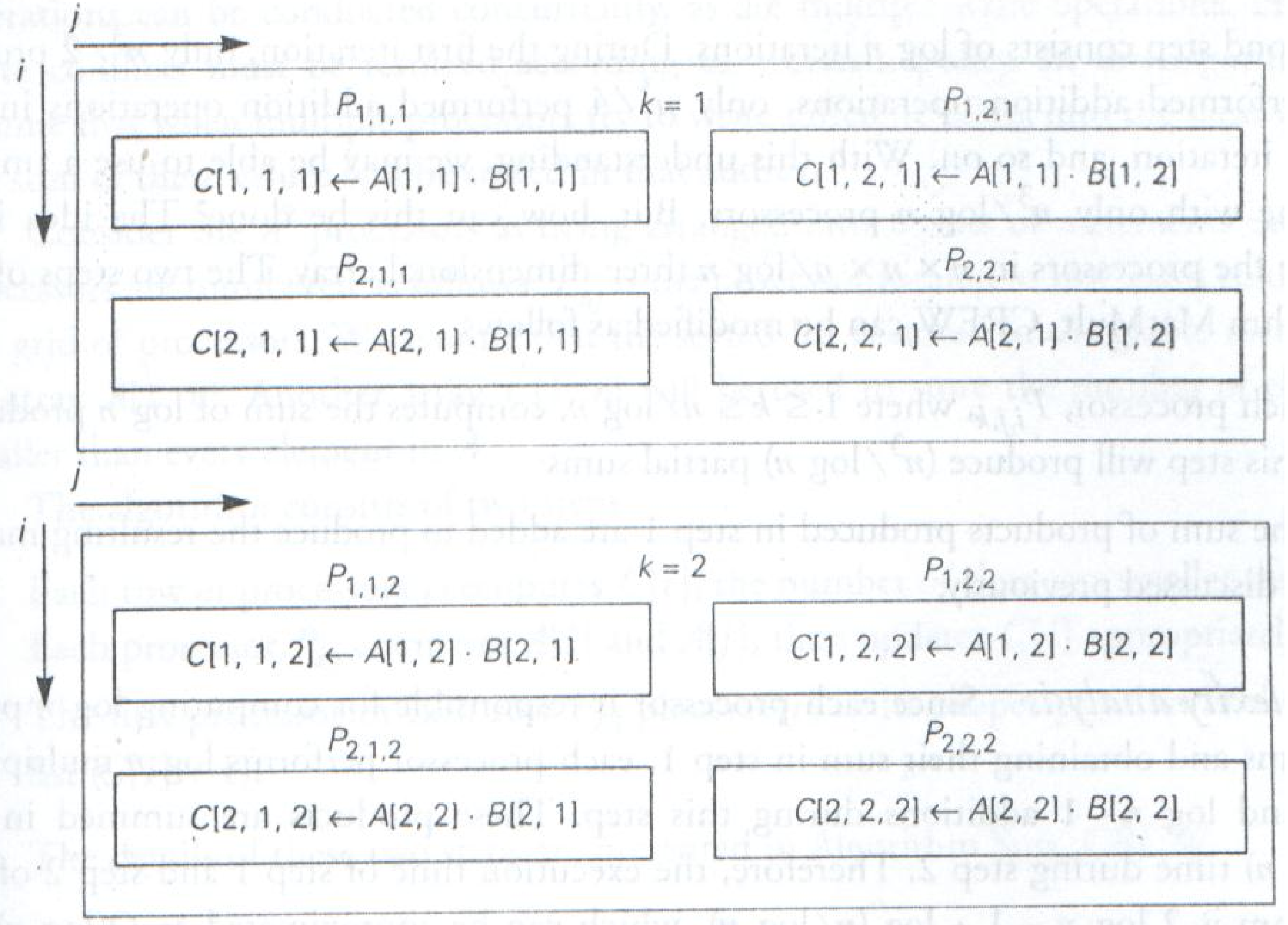


```
Algorithm AllSums_EREW
for i = 1 to log n do
    forall $P_j$, where $2^{i-1} + 1 \le j \le n$ do in parallel
        $A[j] \leftarrow A[j] + A[j - 2^{i-1}]$
    endfor
endfor
```

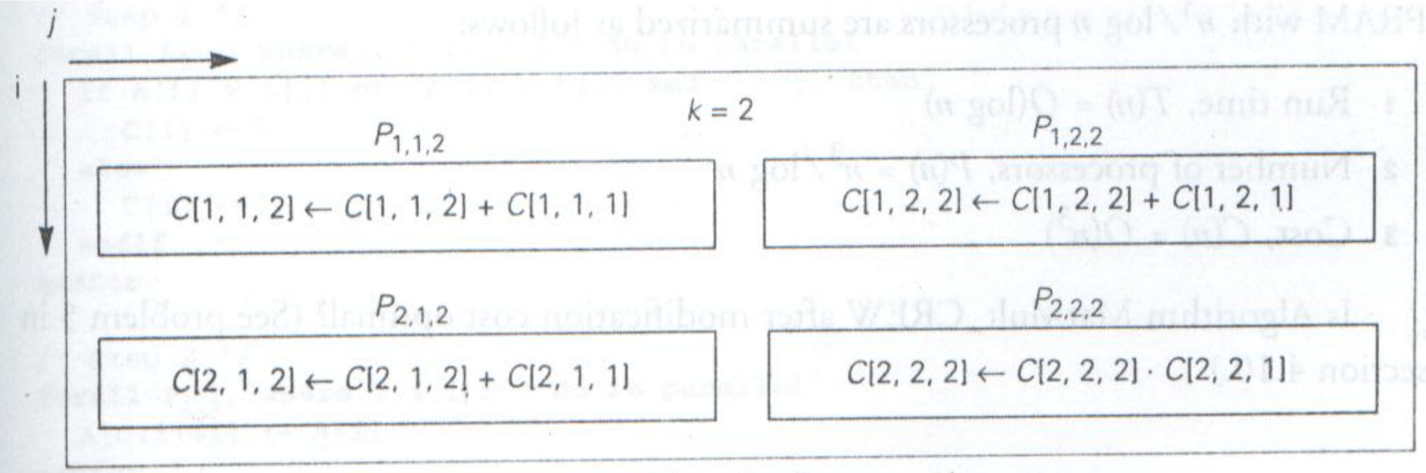
Complexity Analysis

Run Time T(n)
Number of Processors P(n)
Cost C(n) = T(n) * P(n)

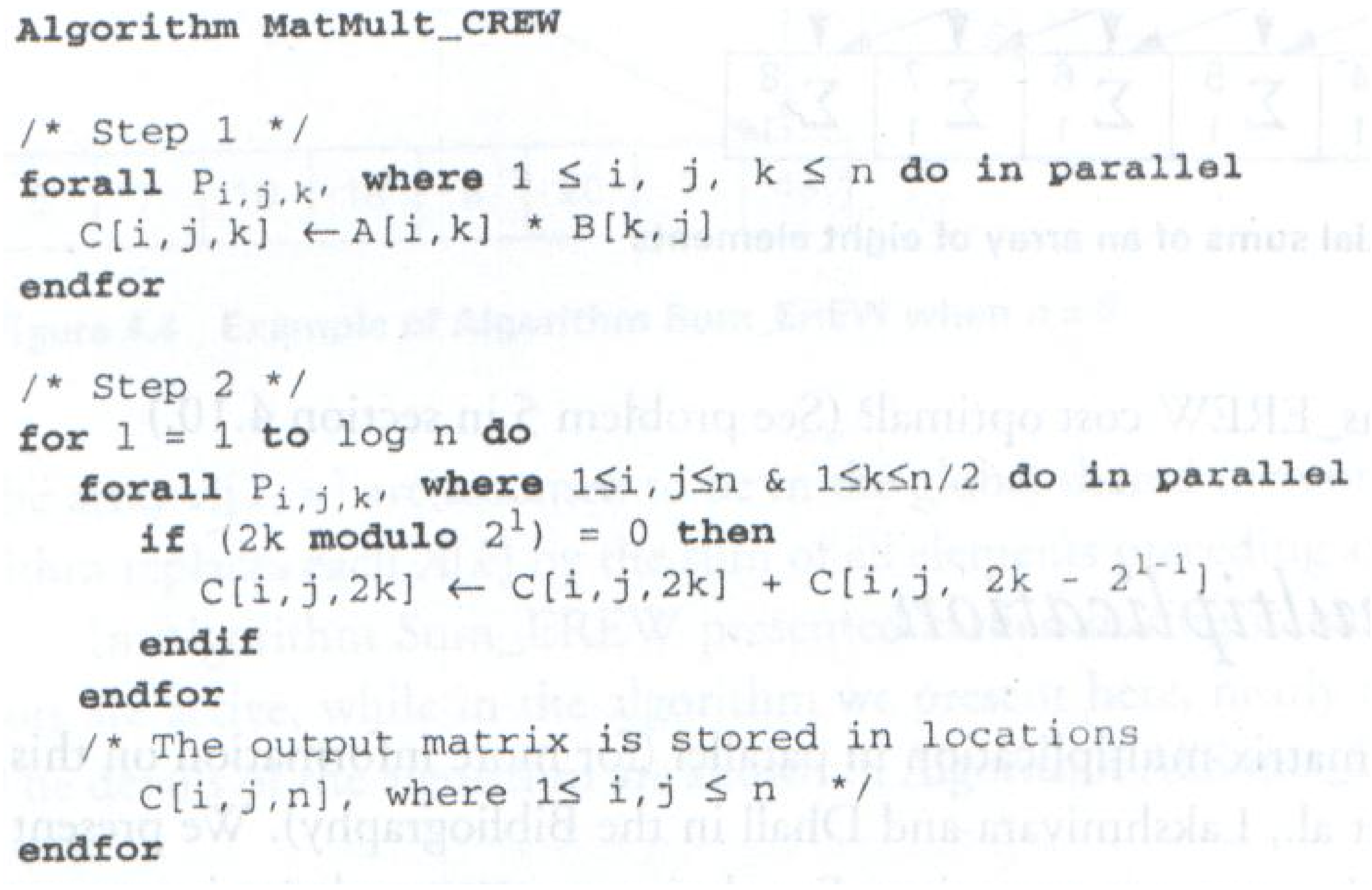
Matrix Multiplication (CREW PRAM)


j
i
k = 1
$P_{1,1,1}$
$C[1, 1, 1] \leftarrow A[1, 1] \cdot B[1, 1]$
$P_{1,2,1}$
$C[1, 2, 1] \leftarrow A[1, 1] \cdot B[1, 2]$
$P_{2,1,1}$
$C[2, 1, 1] \leftarrow A[2, 1] \cdot B[1, 1]$
$P_{2,2,1}$
$C[2, 2, 1] \leftarrow A[2, 1] \cdot B[1, 2]$
j
i
k = 2
$P_{1,1,2}$
$C[1, 1, 2] \leftarrow A[1, 2] \cdot B[2, 1]$
$P_{1,2,2}$
$C[1, 2, 2] \leftarrow A[1, 2] \cdot B[2, 2]$
$P_{2,1,2}$
$C[2, 1, 2] \leftarrow A[2, 2] \cdot B[2, 1]$
$P_{2,2,2}$
$C[2, 2, 2] \leftarrow A[2, 2] \cdot B[2, 2]$

After step 1


j
i
k = 2
$P_{1,1,2}$
$C[1, 1, 2] \leftarrow C[1, 1, 2] + C[1, 1, 1]$
$P_{1,2,2}$
$C[1, 2, 2] \leftarrow C[1, 2, 2] + C[1, 2, 1]$
$P_{2,1,2}$
$C[2, 1, 2] \leftarrow C[2, 1, 2] + C[2, 1, 1]$
$P_{2,2,2}$
$C[2, 2, 2] \leftarrow C[2, 2, 2] \cdot C[2, 2, 1]$

After step 2



```
Algorithm MatMult_CREW

/* Step 1 */
forall $P_{i,j,k}$, where $1 \le i, j, k \le n$ do in parallel
  C[i,j,k] ← A[i,k] * B[k,j]
endfor

/* Step 2 */
for l = 1 to log n do
  forall $P_{i,j,k}$, where $1 \le i,j \le n$ & $1 \le k \le n/2$ do in parallel
    if (2k modulo $2^l$) = 0 then
      C[i,j,2k] ← C[i,j,2k] + C[i,j, 2k - $2^{l-1}$]
    endif
  endfor
  /* The output matrix is stored in locations
    C[i,j,n], where $1 \le i,j \le n$ */
endfor
```

Complexity Analysis

Run Time T(n)
Number of Processors P(n)
Cost C(n) = T(n) * P(n)