How does an SIMD computer work?

- A Host computer is necessary to do the I/O operations.
- The user program is loaded into the control memory.
- The data is distributed to all the memory modules.
- The control unit decodes the instn and executes it if it is a scalar instn. If it is a vector instn, it broadcasts the control signals to the PEs to do the executions.
- Before broadcasting the control signals, the CU broadcasts an enable vector which will enable the PE's.

Masking and Data Routing Mechanisms

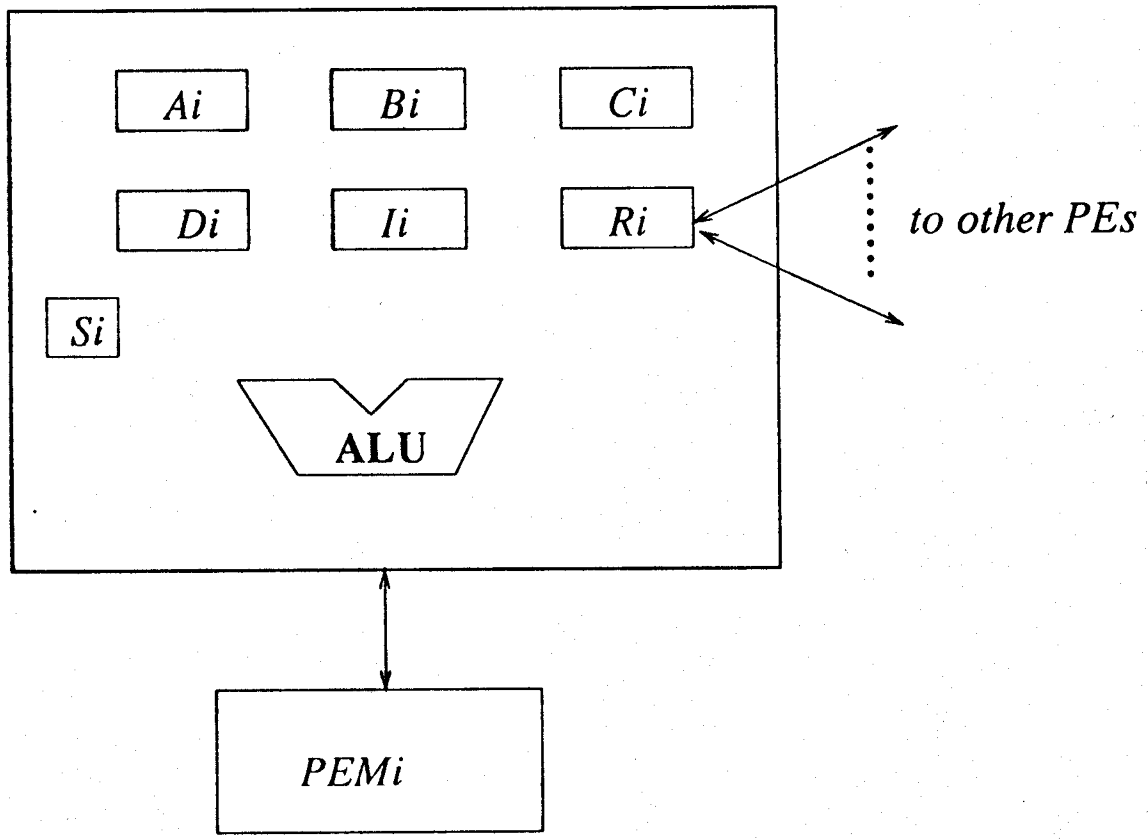
A,B,C -- working registers

Si = status (1 active, 0 inactive)

Ri -- Data routing register

Di -- holds address

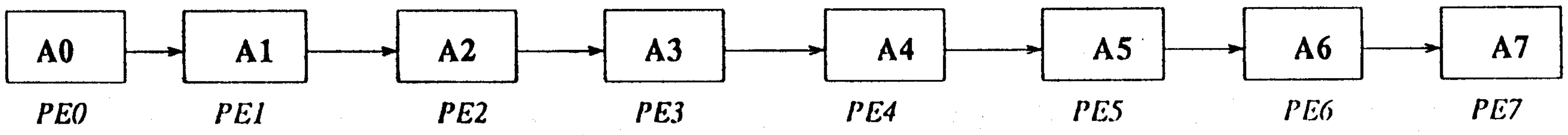
Ii -- Index register



Example: To perform S(k) = Summation(Ai) where i goes from 0 to k

k = 8, number of processing elements = 8

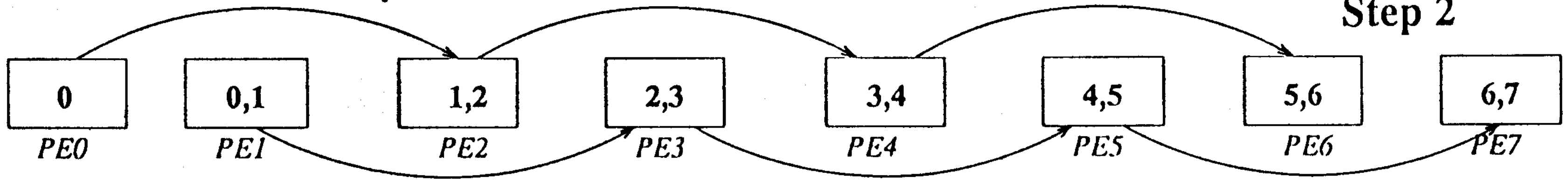
Step 1



R(i+1) <- R(i) for 0 <= PE i <= 6

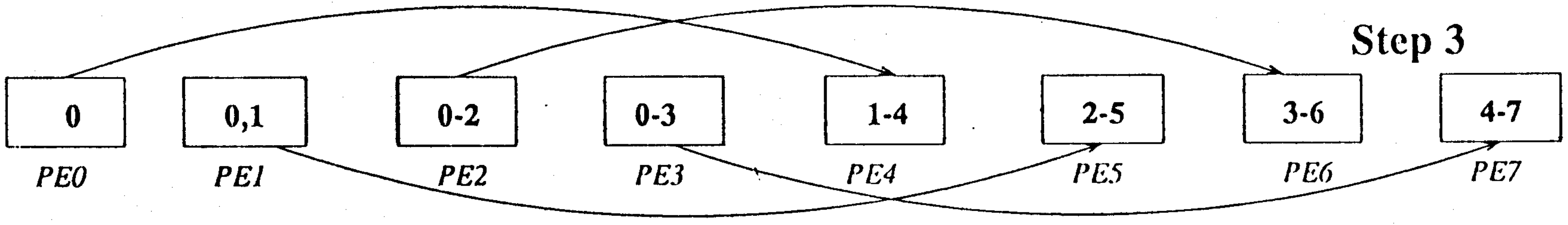
Ai <- [A(i)] + R(i) for 1 <= i <= 7

Step 2



R(i) <- A(i)   R(i+2) <- R(i) 0 <= i <= 5   Ai <- Ai + Ri 2 <= i <= 7

Step 3



R(i) <- A(i)   R(i+4) <- R(i) 0 <= i <= 3   Ai <- Ai + Ri 4 <= i <= 7

Matrix Multiplication

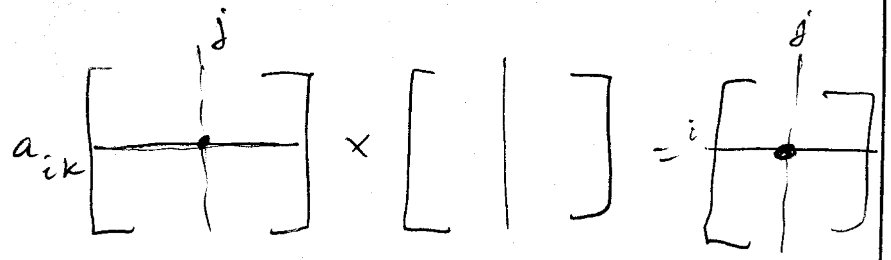
Example 5.3: An $O(n^3)$ algorithm for SISD matrix multiplication

```
For $i = 1$ to $n$ Do
  For $j = 1$ to $n$ Do
    $c_{ij} = 0$ (initialization)
    For $k = 1$ to $n$ Do
      $c_{ij} = c_{ij} + a_{ik} \cdot b_{kj}$ (scalar additive multiply)
    End of $k$ loop
  End of $j$ loop
End of $i$ loop
```

Example 5.4: An $O(n^2)$ algorithm for SIMD matrix multiplication

```
For $i = 1$ to $n$ Do
  Par for $k = 1$ to $n$ Do
    $c_{ik} = 0$ (vector load)
  For $j = 1$ to $n$ Do
  Broadcast $a_{ij}$
  Par for $k = 1$ to $n$ Do
    $c_{ik} = c_{ik} + a_{ij} \cdot b_{jk}$ (vector multiply)
  End of $j$ loop
End of $i$ loop
```

Note: $$c_{ij} = \sum_{k=1}^{n} a_{ik} \bullet b_{kj}$$



$c_{ij} = a_{i1} \cdot b_{1j} + a_{i2} \cdot b_{2j} + \dots + a_{in} \cdot b_{nj}$

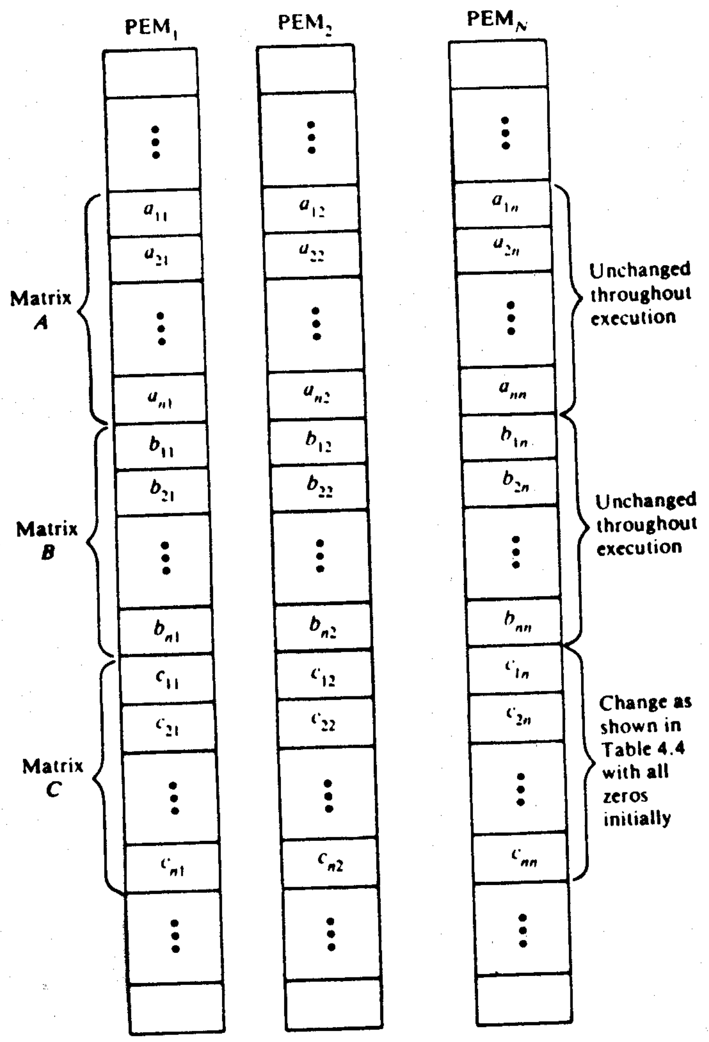


Figure 5.18 Memory allocation for SIMD matrix multiplication.

Table 5.4 Successive contents of the C array in memory

Outer loop	Inner loop	Parallel SIMD operations on $k = 1, 2, \ldots, n$			
i	j	$c_{i1} \leftarrow c_{i1} + a_{ij} \times b_{j1}$	$c_{i2} \leftarrow c_{i2} + a_{ij} \times b_{j2}$	$\cdots$	$c_{in} \leftarrow c_{in} + a_{ij} \times b_{jn}$
1	1	$c_{11} \leftarrow c_{11} + a_{11} \times b_{11}$	$c_{12} \leftarrow c_{12} + a_{11} \times b_{12}$		$c_{1n} \leftarrow c_{1n} + a_{11} \times b_{1n}$
	2	$c_{11} \leftarrow c_{11} + a_{12} \times b_{21}$	$c_{12} \leftarrow c_{12} + a_{12} \times b^{22}$		$c_{1n} \leftarrow c_{2n} + a_{12} \times b_{2n}$
	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$
	n	$c_{11} \leftarrow c_{11} + a_{1n} \times b_{n1}$	$c_{12} \leftarrow c_{12} + a_{1n} \times b_{n2}$		$c_{1n} \leftarrow c_{1n} + a_{1n} \times b_{nn}$
2	1	$c_{21} \leftarrow c_{21} + a_{21} \times b_{11}$	$c_{22} \leftarrow c_{22} + a_{21} \times b_{12}$		$c_{2n} \leftarrow c_{2n} + a_{21} \times b_{1n}$
	2	$c_{21} \leftarrow c_{21} + a_{22} \times b_{21}$	$c_{22} \leftarrow c_{22} + a_{22} \times b_{22}$		$c_{2n} \leftarrow c_{2n} + a_{22} \times b_{2n}$
	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$
	n	$c_{21} \leftarrow c_{21} + a_{2n} \times b_{n1}$	$c_{22} \leftarrow c_{22} + a_{2n} \times b_{n2}$		$c_{2n} \leftarrow c_{2n} + a_{2n} \times b_{2n}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
n	1	$c_{n1} \leftarrow c_{n1} + a_{n1} \times b_{11}$	$c_{n2} \leftarrow c_{n2} + a_{n1} \times b_{12}$		$c_{nn} \leftarrow c_{nn} + a_{n1} \times b_{1n}$
	2	$c_{n1} \leftarrow c_{n1} + a_{n2} \times b_{21}$	$c_{n2} \leftarrow c_{n2} + a_{n2} \times b_{22}$		$c_{nn} \leftarrow c_{nn} + a_{n2} \times b_{2n}$
	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$
	n	$c_{nn} \leftarrow c_{nn} + a_{nn} \times b_{n1}$	$c_{n2} \leftarrow c_{n2} + a_{nn} \times b_{n2}$		$c_{nn} \leftarrow c_{nn} + a_{nn} \times b_{nn}$
Local memory		PEM$_1$	PEM$_2$	$\cdots$	PEM$_n$

n x n Mesh

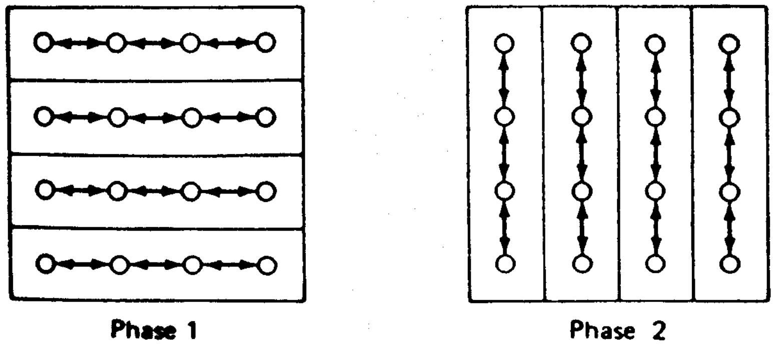


Figure 1.3.27 An $O(n)$ algorithm for multiplying two $n \times n$ matrices A and B with elements a_{ij} and b_{ij}, respectively, on an $n \times n$ mesh. Initially, processor (i, j) of the mesh holds elements a_{ij} and b_{ij}, and at the end of the algorithm, processor (i, j) will hold the ijth element $\sum_{m=1}^{n} a_{im} b_{mj}$ of the product AB.

The algorithm consists of three phases, each requiring $O(n)$ time. In the first phase, each processor (i, j) broadcasts a_{ij} to the processors in the ith row; these are n multinode broadcasts, one within each row, requiring $O(n)$ time. In the second phase, each processor (i, j) broadcasts b_{ij} to the processors in the jth column; these are n multinode broadcasts, one within each column, requiring $O(n)$ time. At the end of two phases, each processor (i, j) holds the values a_{im} and b_{mj} for $m = 1, 2, \ldots, n$, and can form the ijth element $\sum_{m=1}^{n} a_{im} b_{mj}$ of the product AB in time $O(n)$ (which is the third phase). Note that phases 1 and 2 can be done in parallel, assuming that simultaneous communication along all incident links is possible. Also, by appropriately interleaving additions, multiplications, and communications, the algorithm can be made more economical in terms of time and storage.

The Illiac IV Architecture

- Distributed memory architecture
- 64 PEs connected as an 8X8 2-D mesh with end around connection.

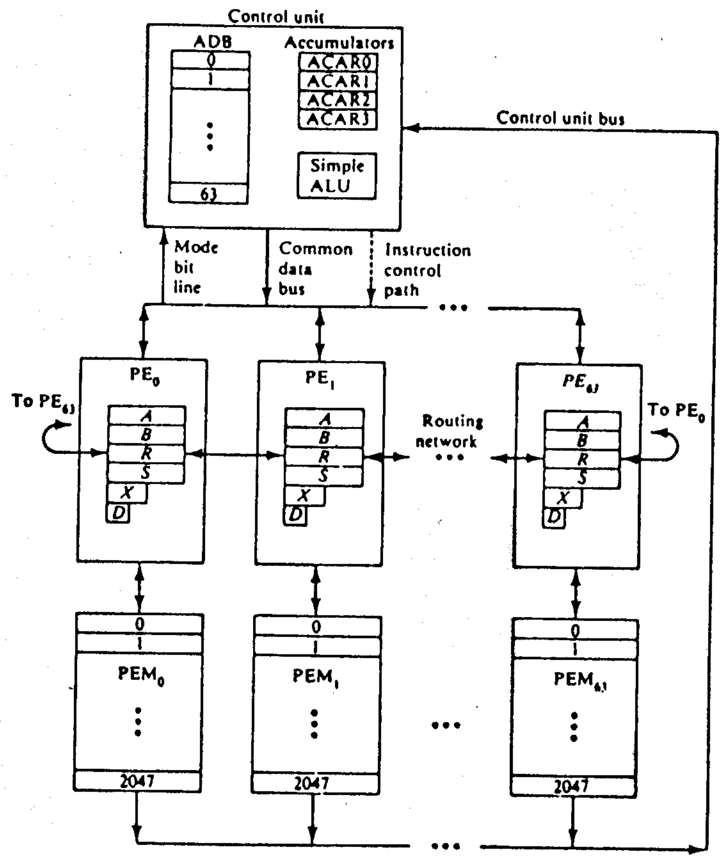


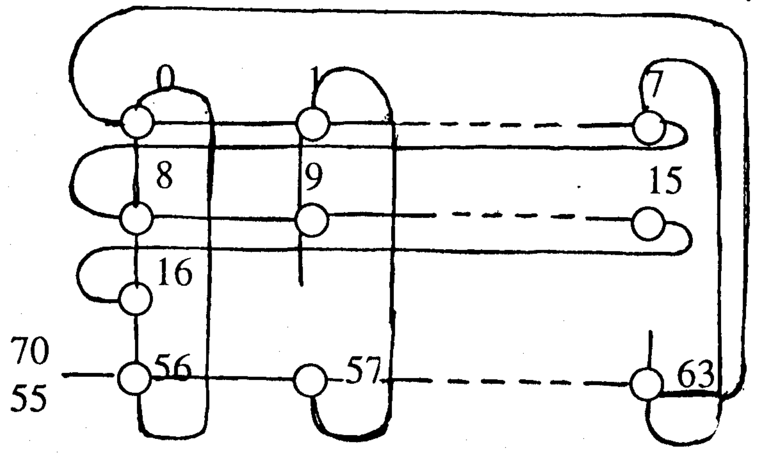
Figure 6.4 A 64-PE Illiac IV array. (Courtesy of *IEEE Proc.* Bouknight et al., April 1972.)

LDB: Local Data Buffer
64, 64-bits each

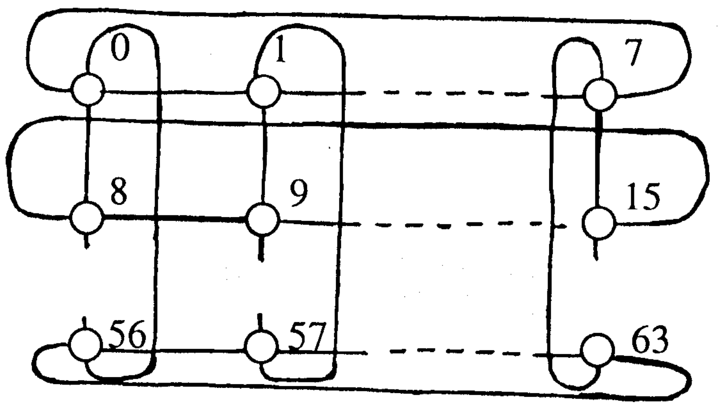
PEM$_i$: 2K X 64 bits memory

The Illiac IV Network

PE$_i$ is connected to PE $_{it}$, PE $_{i-7}$, PE $_{i+8}$, and PE $_{i-8}$ mod 64. (End around connection)

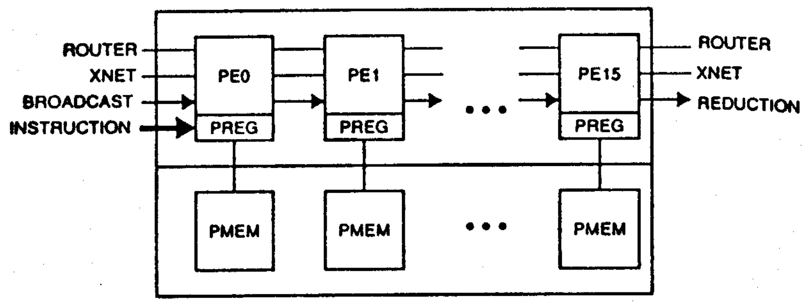


2-D Torus (wrap around)

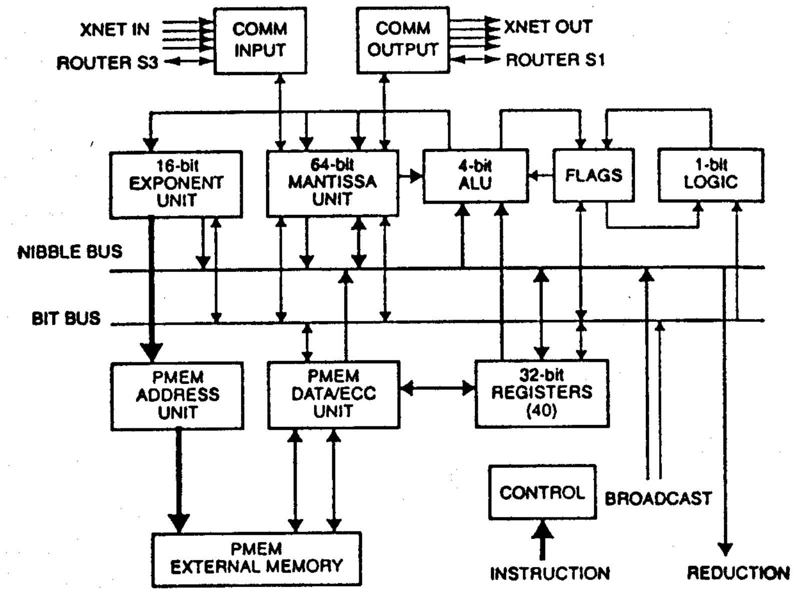


Maspar MP-1 Architecture

- Configuration with 1K-16K PE's are available
- Each PE has a 4-bit ALU, 1-bit logic unit, a 64-bit mantissa unit, a 16-bit exponent unit, communication input and output ports.
- Each PE has 40 32-bit registers available to the programmer.
- Each processor board has 1024 PE's arranged as 64 PE clusters (PEC's) with 16 PE's per cluster.
- Each PEC is a chip connected to 8 neighbors via an octagonal mesh.
- Another network, called Multistage Crossbar Network, with three router stages gives a function of 1024X1024 crossbar for routing from any PEC to another PEC.

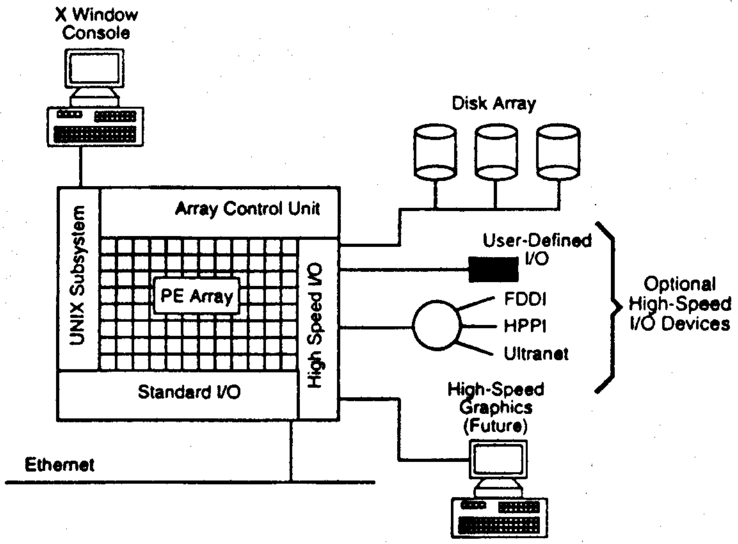


(a) A PE cluster

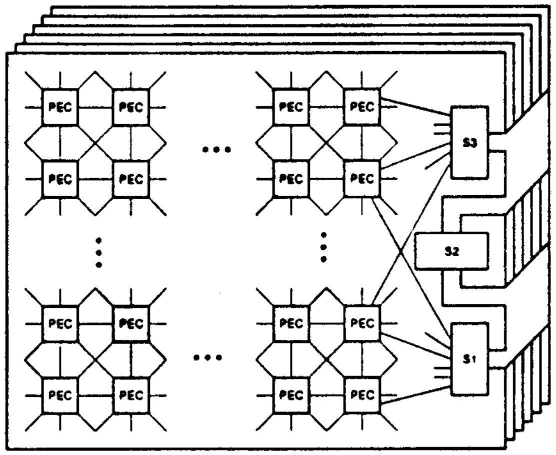


(b) Processor element and memory

Figure 8.26 Processing element and memory design in the MasPar MP-1. (Courtesy of MasPar Computer Corporation, 1990)



(a) MP-1 System Block Diagram



(b) Array of PE clusters

Figure 8.25 The MasPar MP-1 architecture. (Courtesy of MasPar Computer Corporation, 1990)