INTERNATIONAL ATOMIC ENERGY AGENCY
UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION
INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS
I.C.T.P., P.O. BOX 586, 34100 TRIESTE, ITALY, CABLE: CENTRATOM TRIESTE

UNITED NATIONS INDUSTRIAL DEVELOPMENT ORGANIZATION
INTERNATIONAL CENTRE FOR SCIENCE AND HIGH TECHNOLOGY
c/o INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS 34100 TRIESTE (ITALY) VIA GRIGNANO, 9 (ADRIATICO PALACE) P.O. BOX 586 TELEPHONE 040-224572 TELEFAX 040-224575 TELEX 460449 APH I

SMR/474 - 16

COLLEGE ON
"THE DESIGN OF REAL-TIME CONTROL SYSTEMS"
1 - 26 October

PARALLEL PROCESSING

I. WILLERS
Advanced Computing Group
CERN
CH-1211 Geneva 23
Switzerland

Parallel Processing

Ian Willers

College on
The Design of Real-Time
Control Systems

Parallel Computing

Ian Willers

Why is it interesting?

Classification of Architectures;

Common Practical Models;

Three concrete examples.

Why is it interesting?

A Super Computer

CRAY X-MP/48 – 4 processors, shared memory

Intel iPSC/860 – 128 i860 processors distributed memory

Connection Machine – 65,535 simple processors

Is a Super Computer cost effective?

Compare with –

Apollo DN10000 – 4x3 units (Cray 4x8)

Silicon Graphics SGI 240 – 4xMIPS 2000 processors

We do need all this power.

FERMILAB:

High Energy Physicists have always wanted more computer power than they could afford to buy in the commercial market place.

CERN: Computing in the 90's:

Total needs in 1991–92 = 600 units

150 in CERN computer center
300 from outside laboratories
150 by parallel farms

Present computer center has:

Cray X/MP 48 = 32 units
IBM 3090–600E = 39 units

What is the success of vectorization and parallelization of algorithms?

How can we benefit from automatic tools such as:

compilers

process schedulers

session schedulers

FARMING: Identical code on many processors each receiving a single physics event.

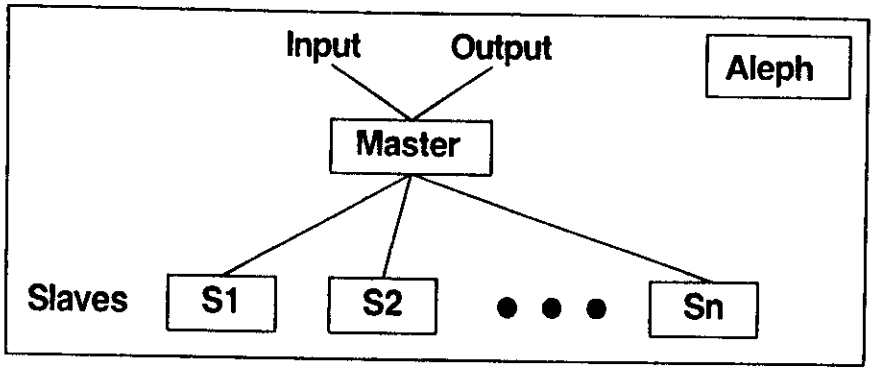
The IBM emulators (the original)

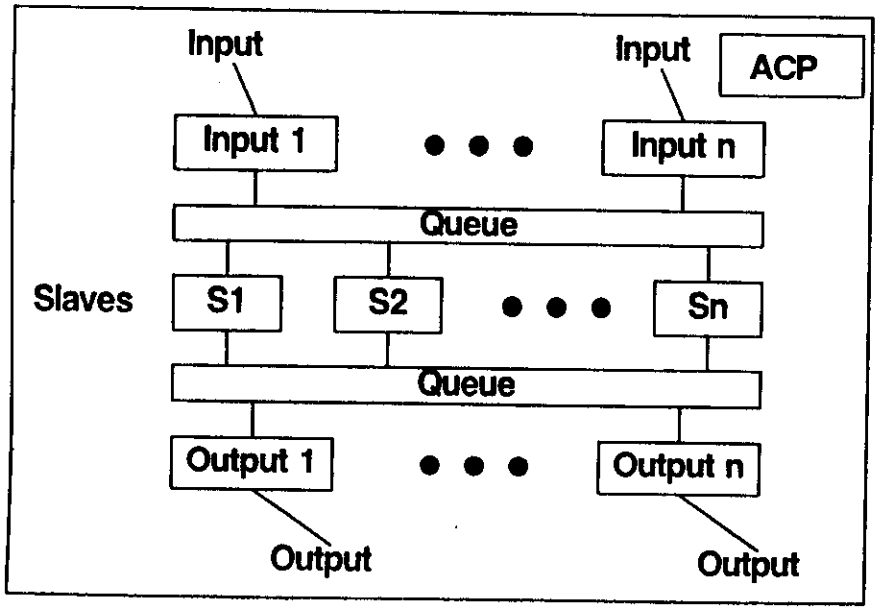
FERMILAB ACP (the ultimate)

L3 Lepics (compromise)

Aleph FALCON (system building)

FARMING: Identical code on many processors each receiving a single physics event.





On-line Applications

IBM Parallel Processing Computer Server (PPCS)

RISC Sparc on FASTBUS

MEIKO Transputers (+ I860's)

HP–Apollo, Cresco–Data (+DSP's)

ASPEX

AMT DAP

TRANSPUTERS

Used in the experiments:

Jetset

OPAL Jets

UA6

ZEUS

Parallel Computing

Ian Willers

Why is it interesting?

Classification of Architectures;

Common Practical Models;

Three concrete examples.

Definition

Parallel Architecture:

A parallel architecture provides an explicit, high-level framework for the development of parallel programming solutions by providing multiple processors, whether simple or complex, that cooperate to solve problems through concurrent execution.

Flynn's taxonomy

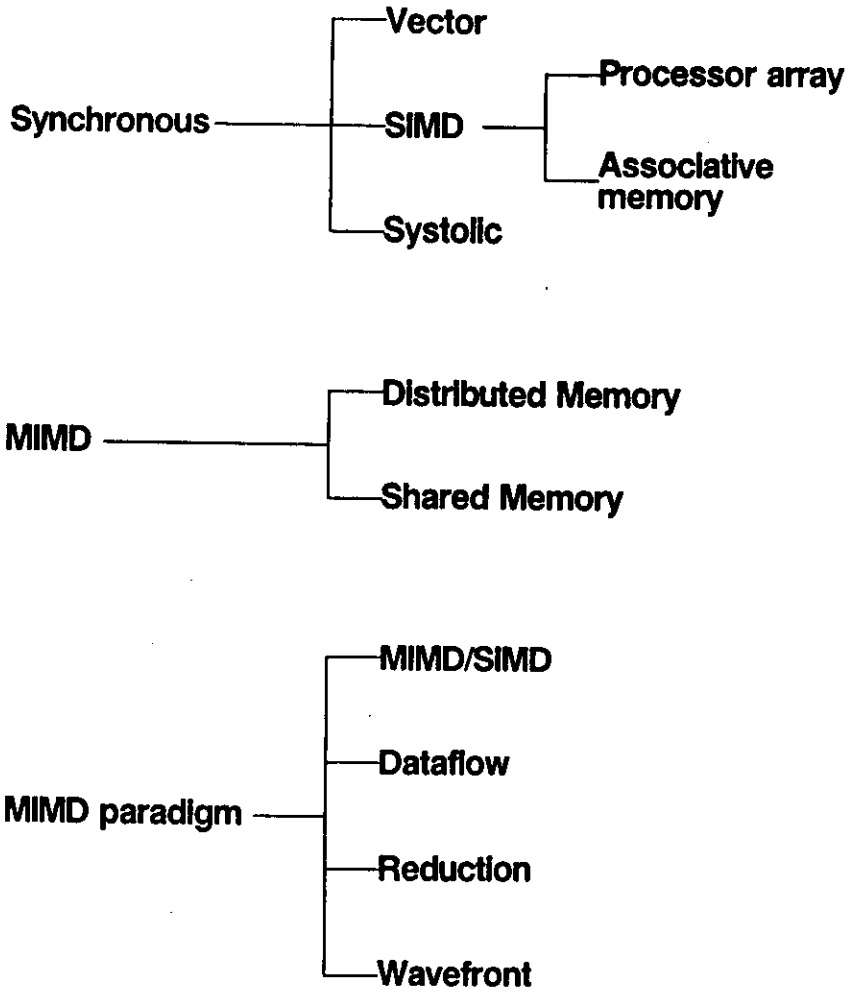
SISD – Single Instruction, Single Data stream
Defines serial computers

MISD – Multiple Instruction, Single Data stream
Multiple processors applying different instructions to a single data stream.
Possible but impractical.

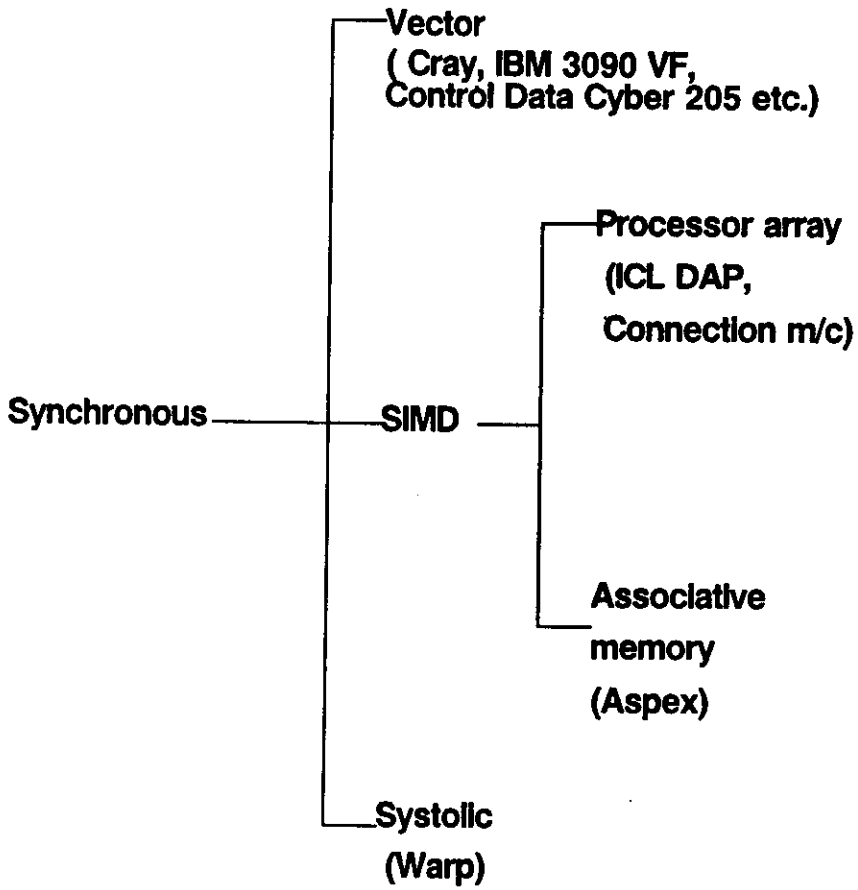
SIMD – Single Instruction, Multiple Data streams
Multiple processors simultaneously executing the same instruction on different data.

MIMD – Multiple Instruction, Multiple Data streams
Multiple processors autonomously executing diverse instructions on diverse data.

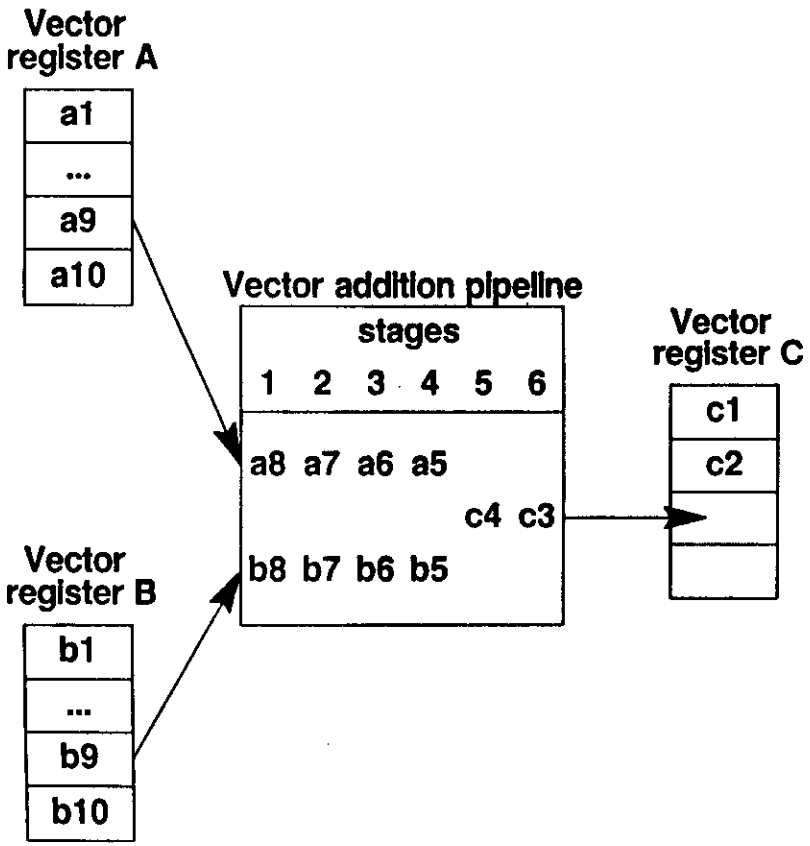
Duncan's taxonomy



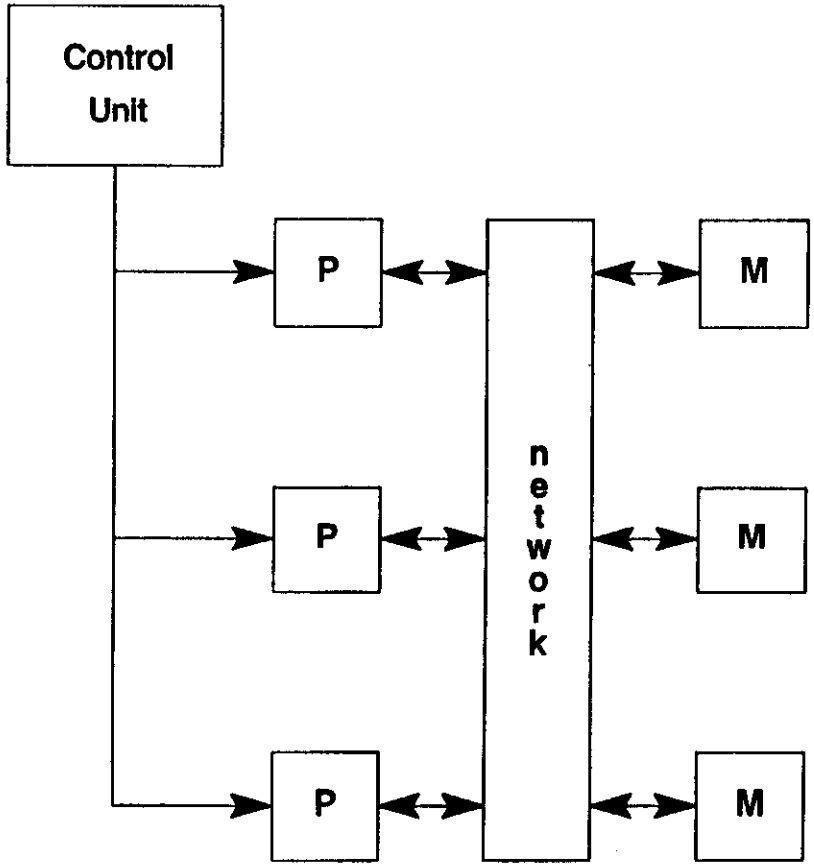
Duncan's taxonomy



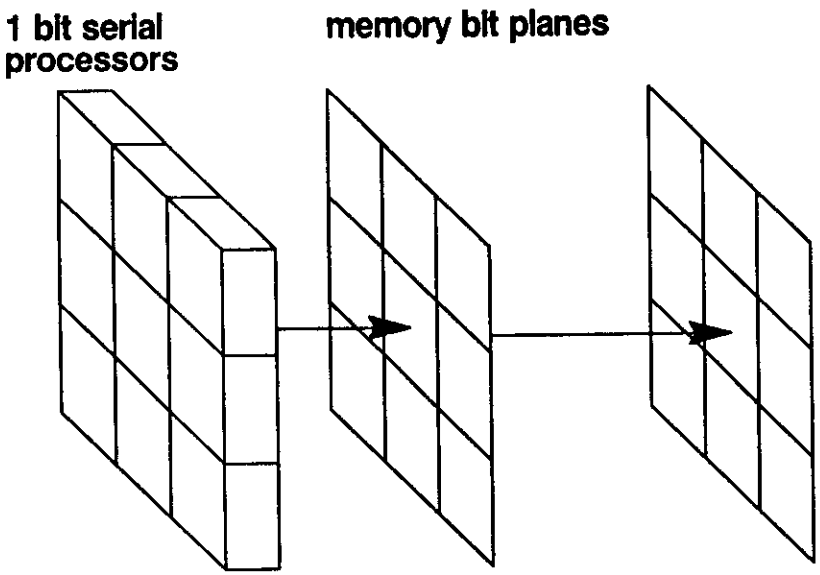
Pipelined Vector Processors



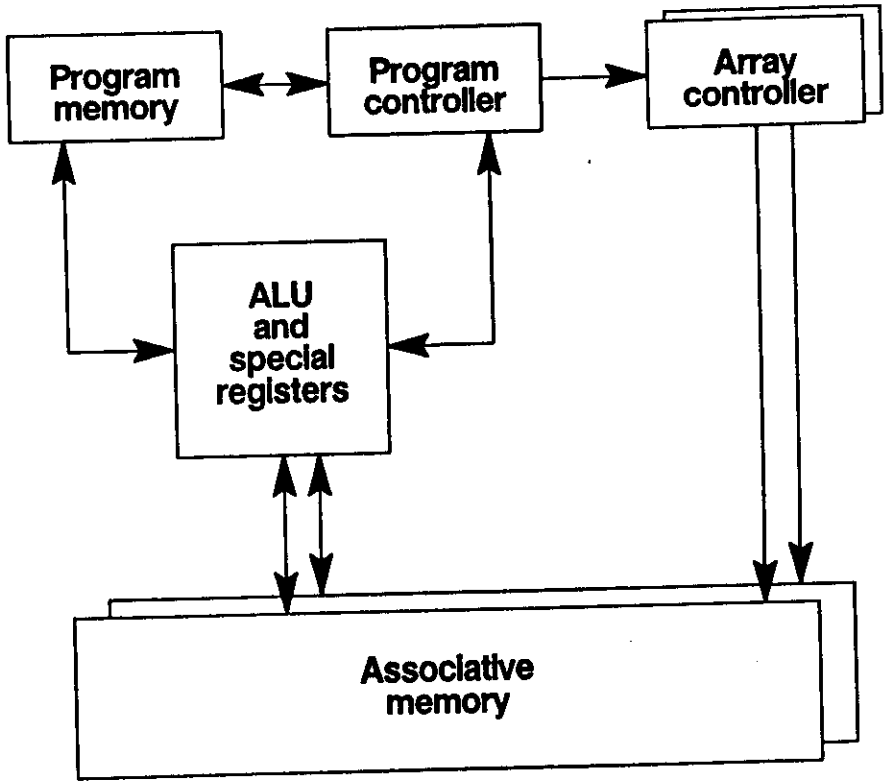
SIMD architectures



Bit-plane array processing



Associative memory processing



Associative memory processing

Comparison register

1	0	0	1	1	0	1	0

Mask register

1	1	1	1	0	0	0	0

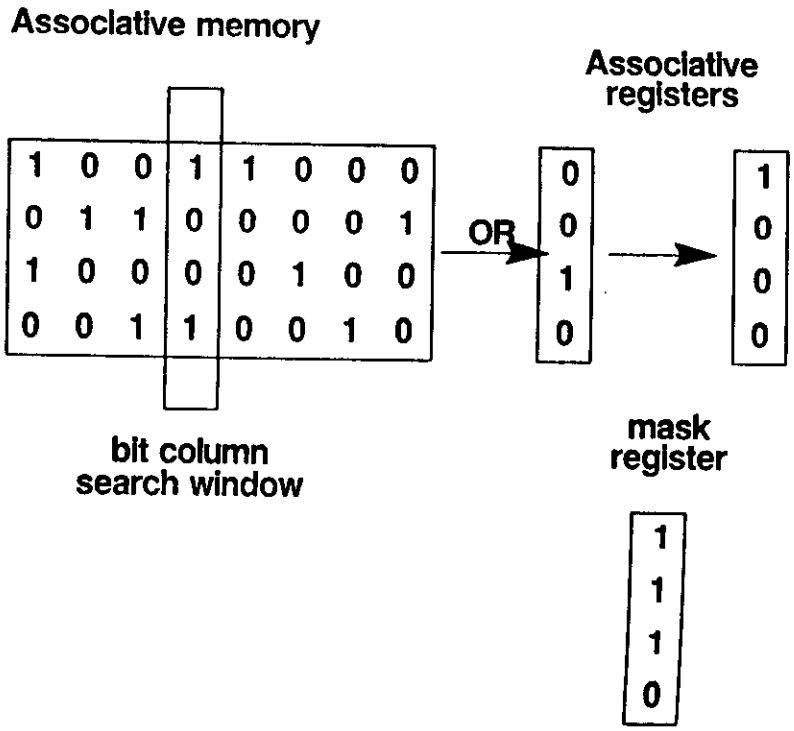
Associative memory

1	0	0	1	1	0	0	0
0	1	1	0	0	0	0	1
1	0	0	0	0	1	0	0
0	0	1	1	0	0	1	0

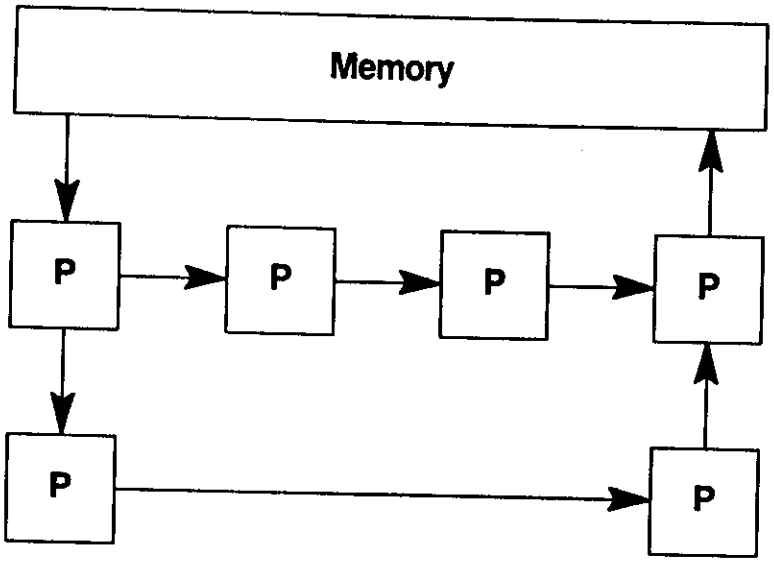
Associative register

1
0
0
0

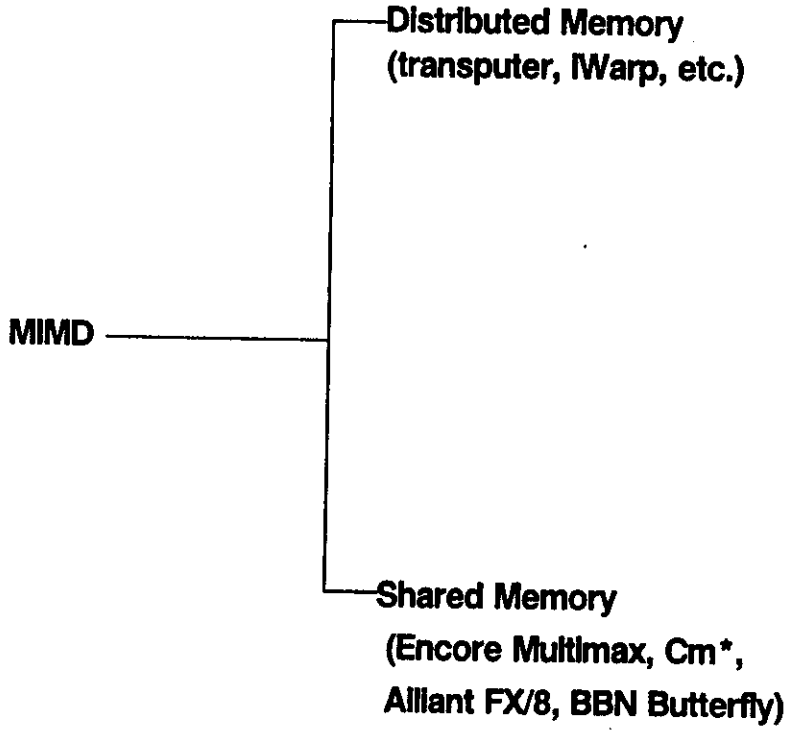
Associative memory processing



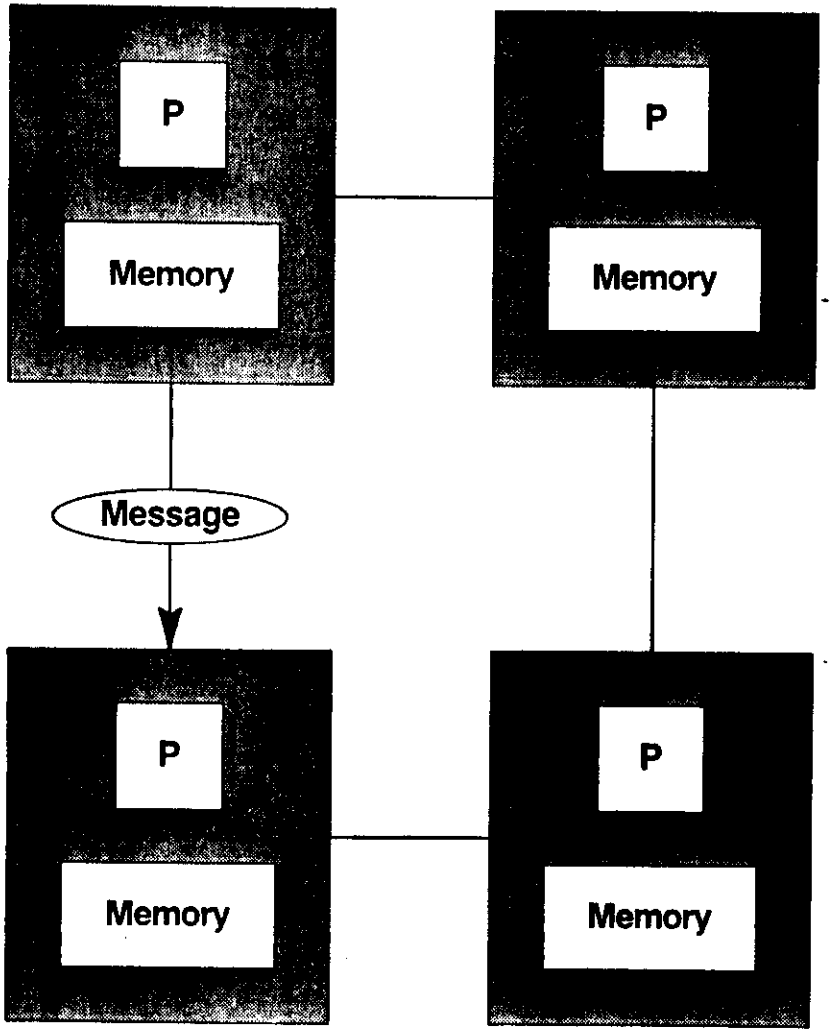
Systolic Arrays



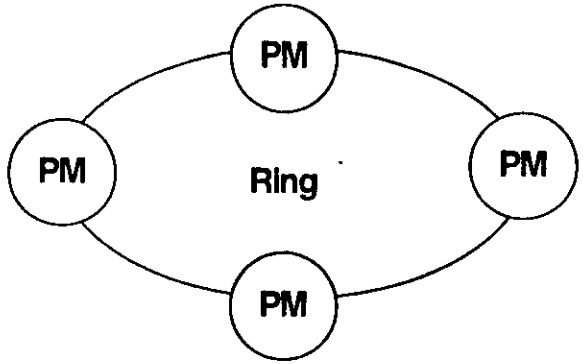
Duncan's taxonomy



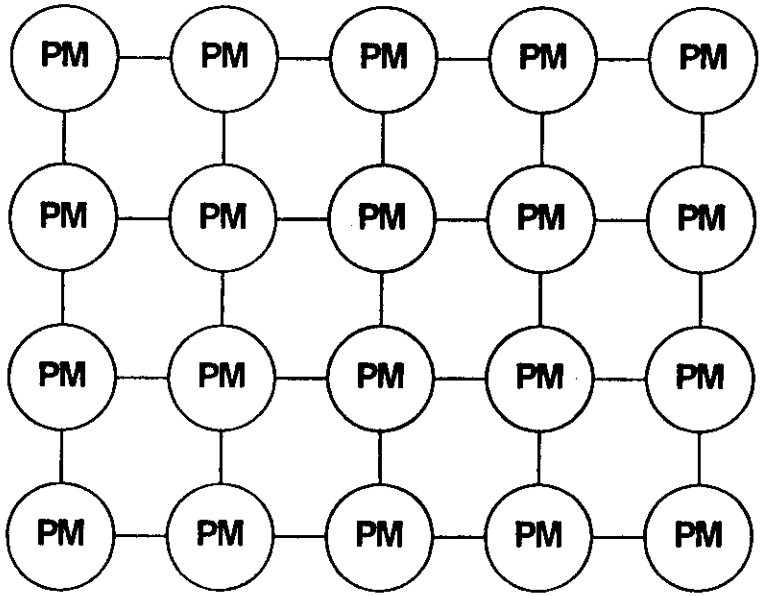
MIMD – distributed memory



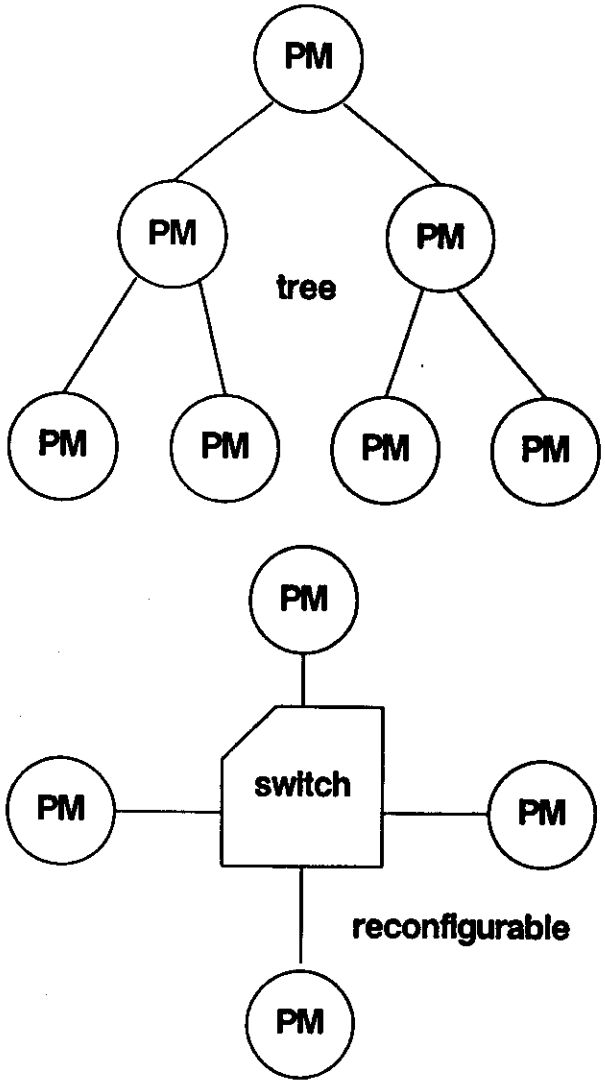
MIMD – distributed memory topologies



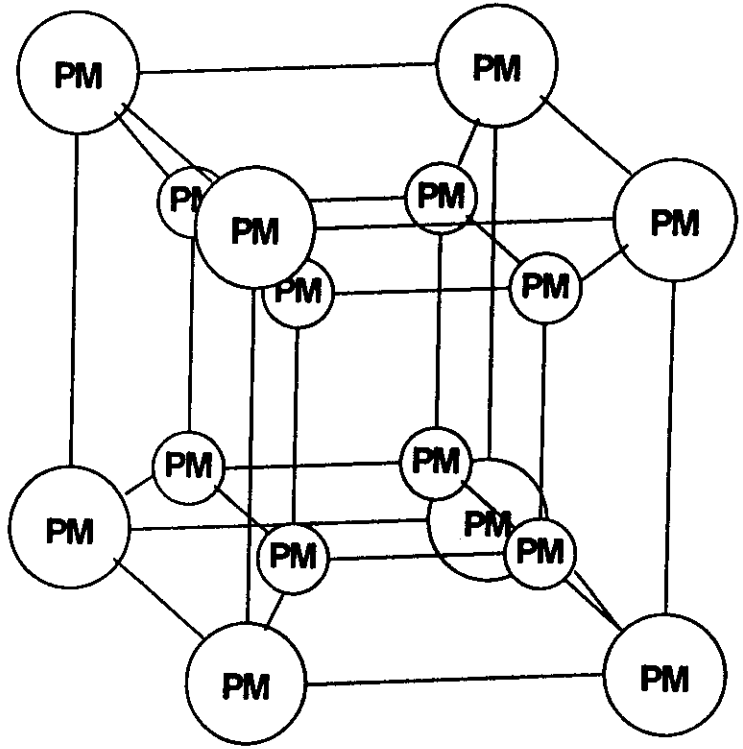
mesh



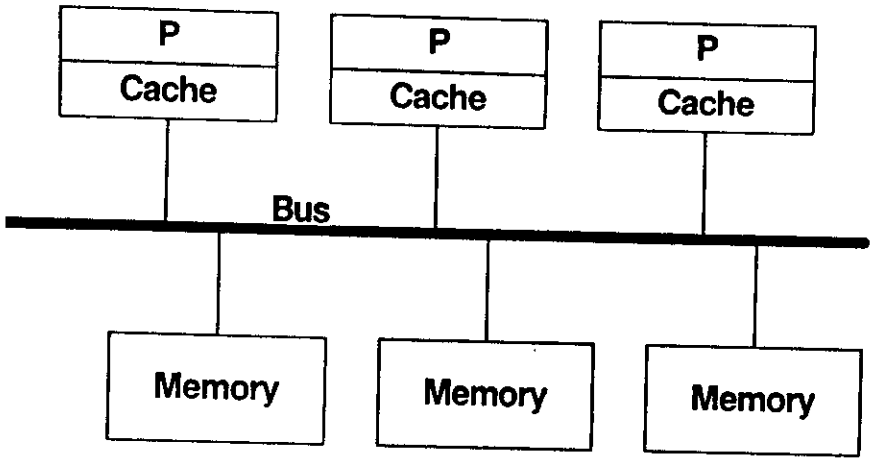
MIMD – distributed memory topologies



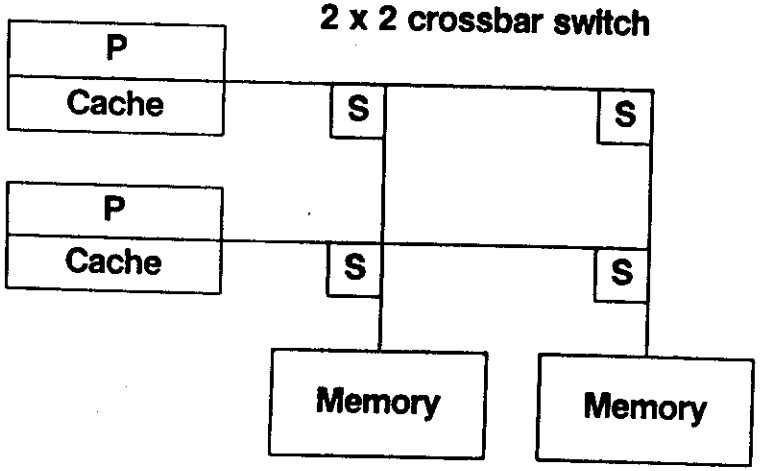
MIMD – distributed memory hypercube



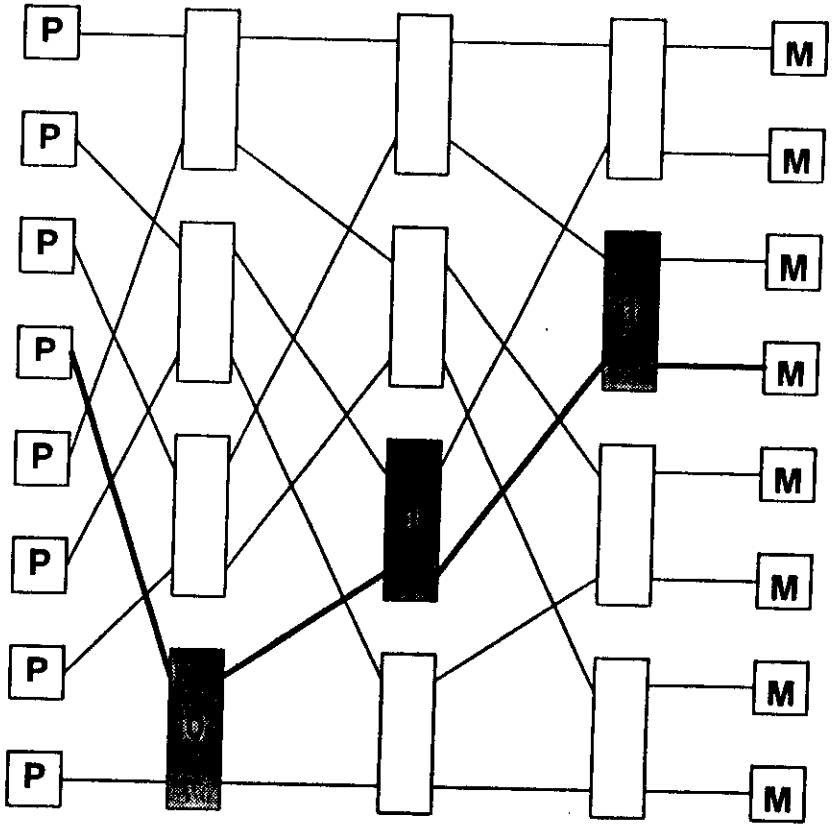
MIMD – shared memory



2 x 2 crossbar switch

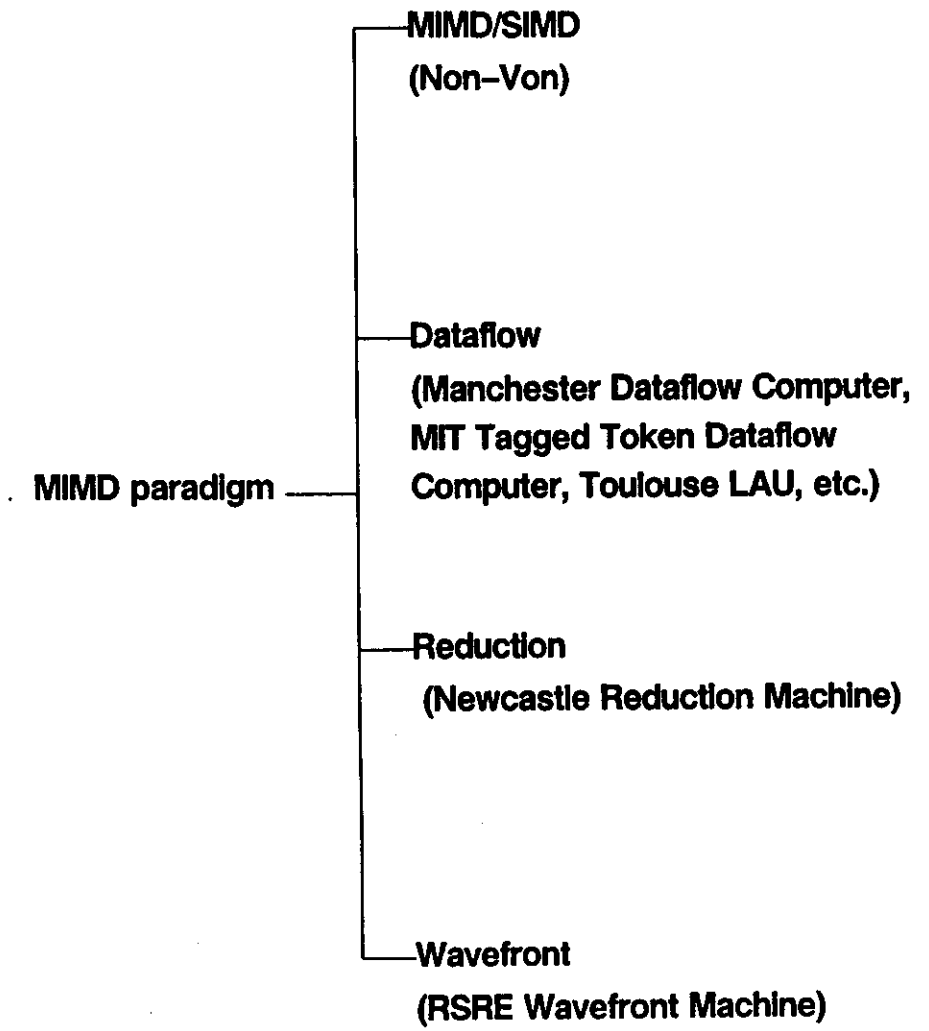


MIMD – shared memory

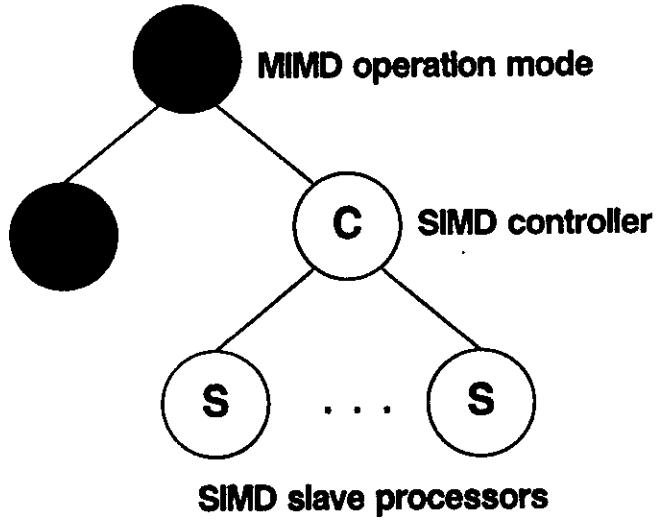


Multistage interconnection network

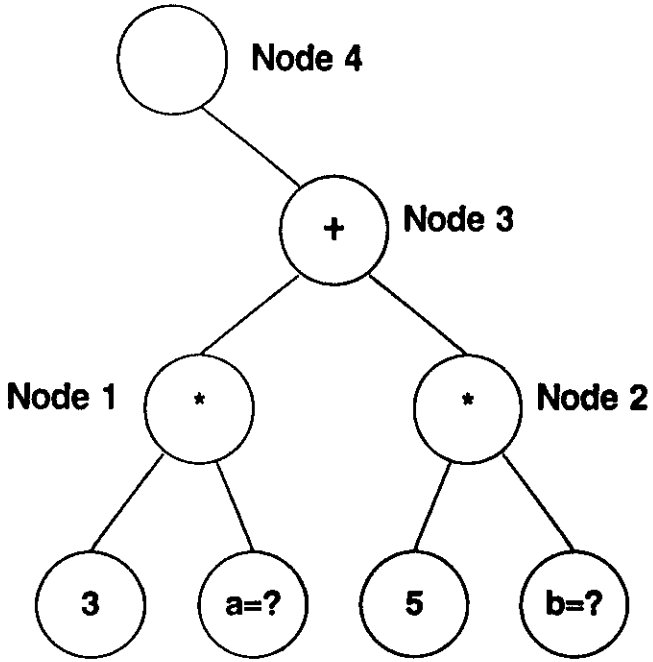
Duncan's taxonomy



MIMD/SIMD



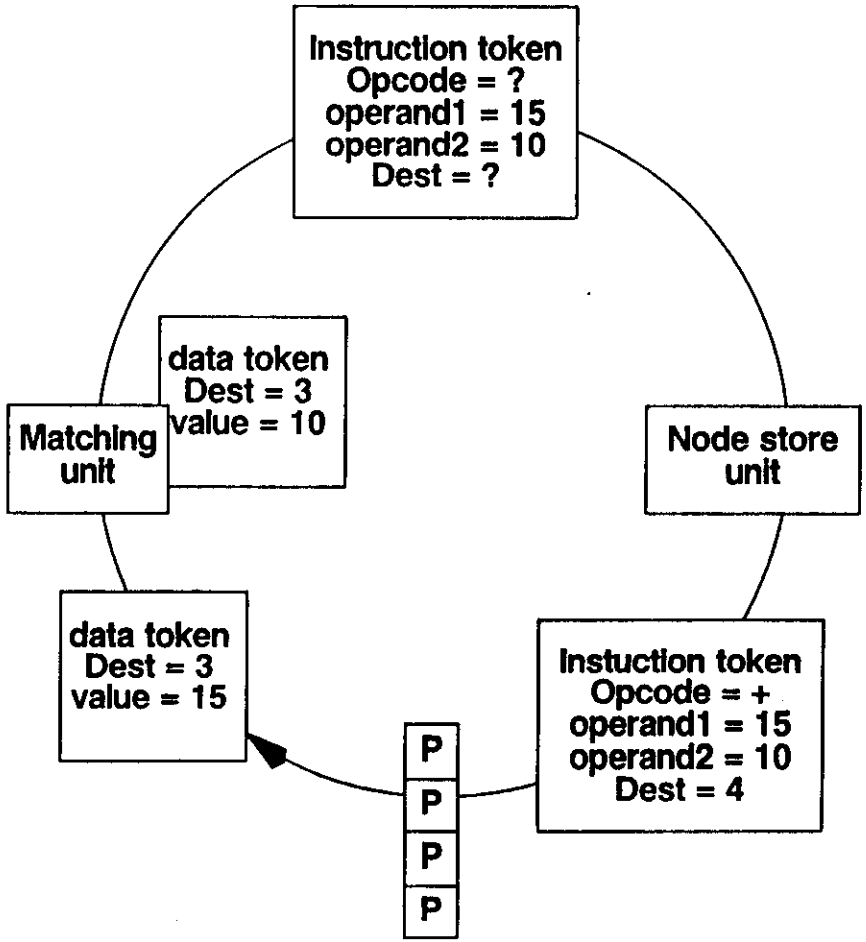
Dataflow



a=5

b=2

Dataflow using tokens



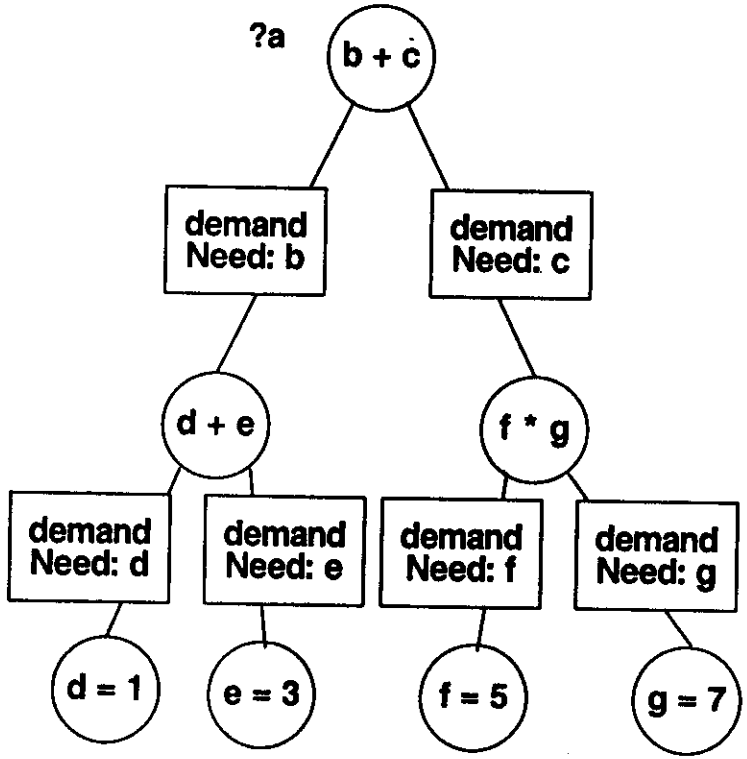
Reduction (or Demand Driven)

Like functional languages

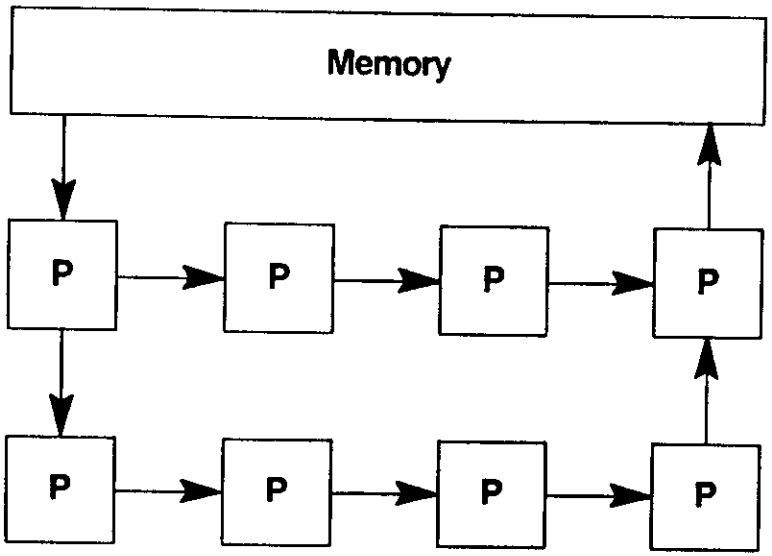
a = b + c
b = d + e
c = f * g
d = 1; e = 3; f = 5; g = 7

?a

Graph Reduction



Waveform Array



Dataflow in systolic arrays.

Parallel Computing

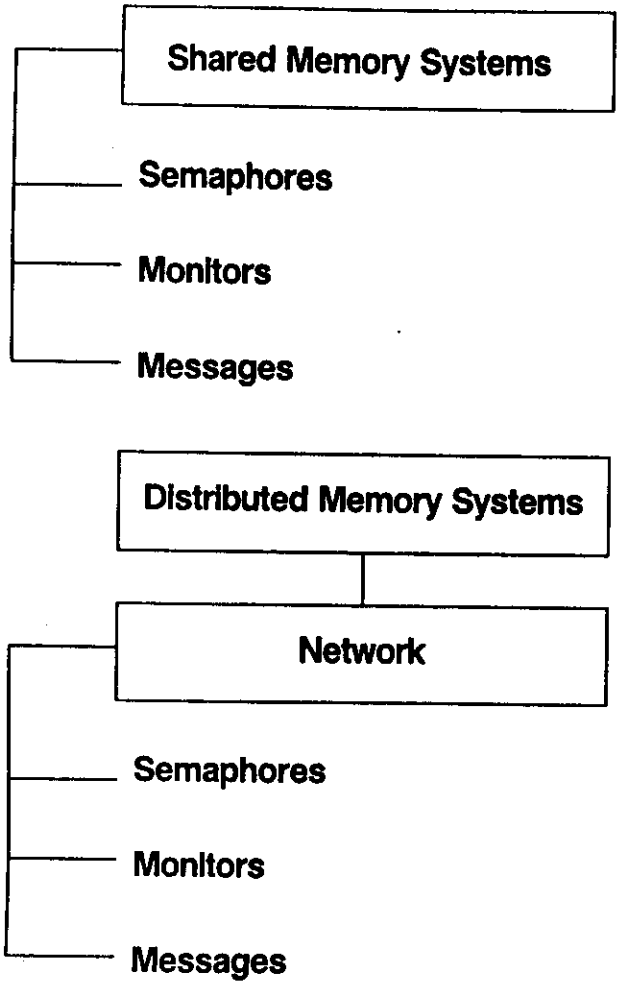
Ian Willers

Why is it interesting?

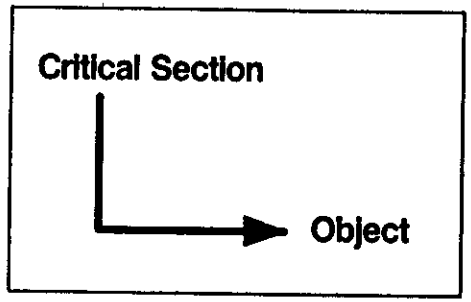
Classification of Architectures;

Common Practical Models;

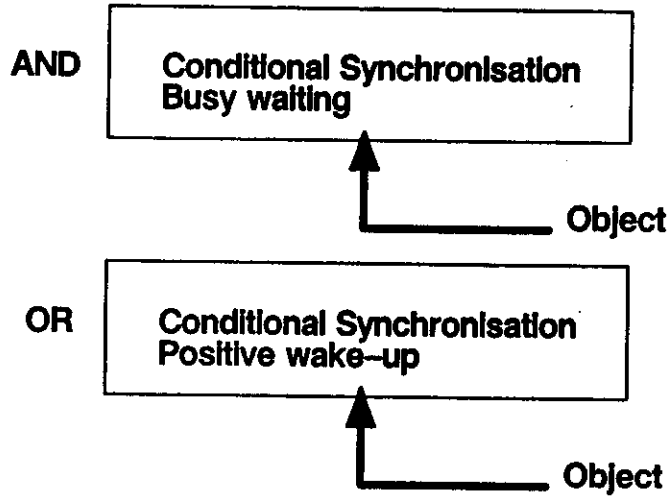
Three concrete examples.



Restrict Access to an Object



Like turning off interupts in a sequential computer



Danger: Deadlock

Semaphores

P(s) : Wait until s>0
s = s−1

V(s): s = s+1

Binary Semaphore restricts value to (0,1)

General Semaphore has any value >=0

Critical Section

P(s)

<critical section>

V(s)

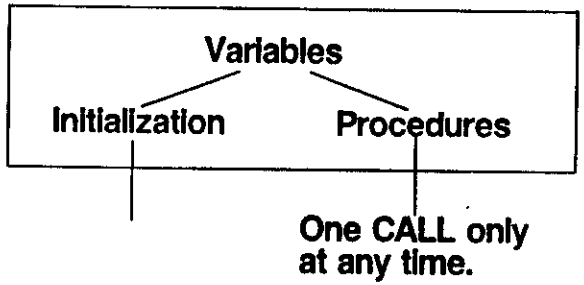
Conditional Synchronisation

When condition true
V(s)

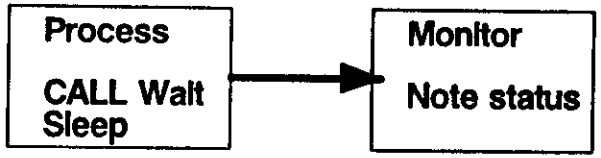
h/w: Fetch&Add or ReadTestWrite
s/w: Difficult to make deadlock free programs

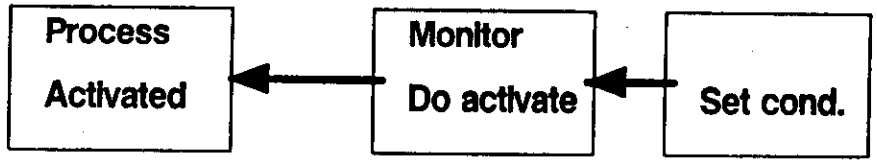
Monitors

Critical Section (based on Object Oriented Programming) called a monitor.



Conditional Synchronisation

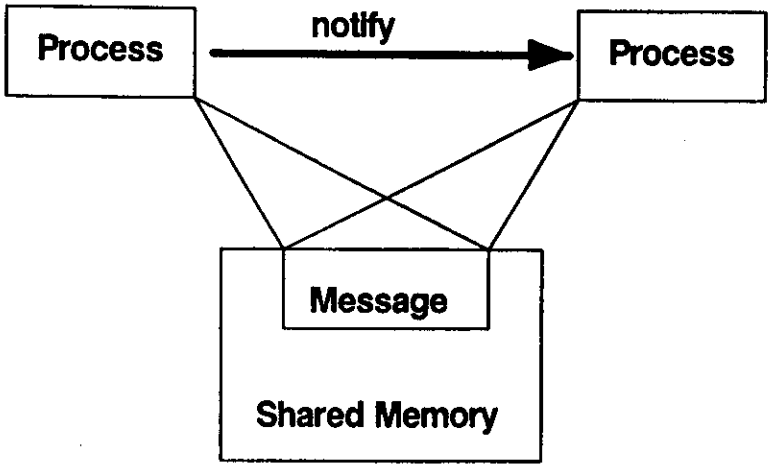




Message Passing (Examples)

Transputer (distributed memory)

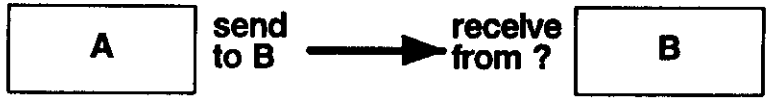
MACH operating system using shared memory



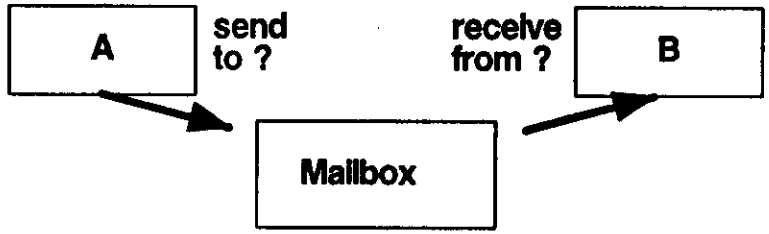
Message Passing (models for naming)

Direct Naming

Client/Server

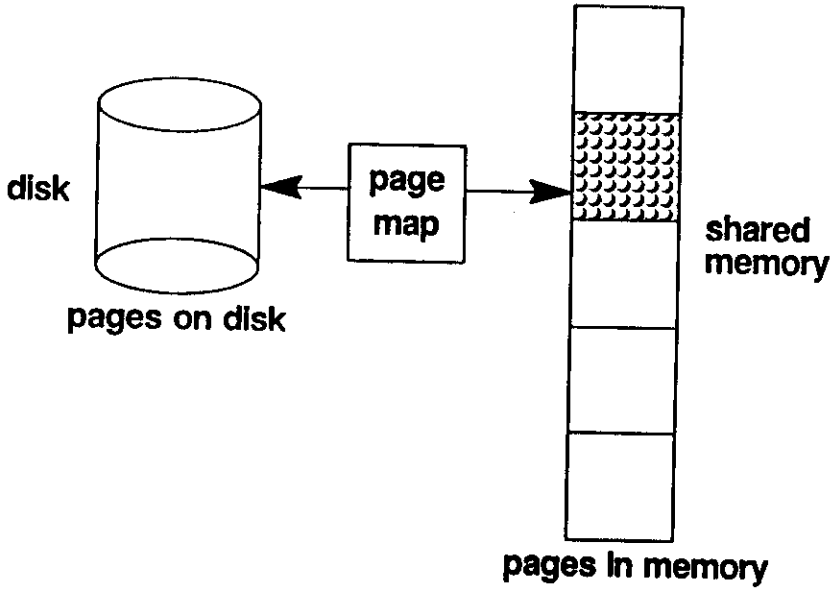


Global Naming



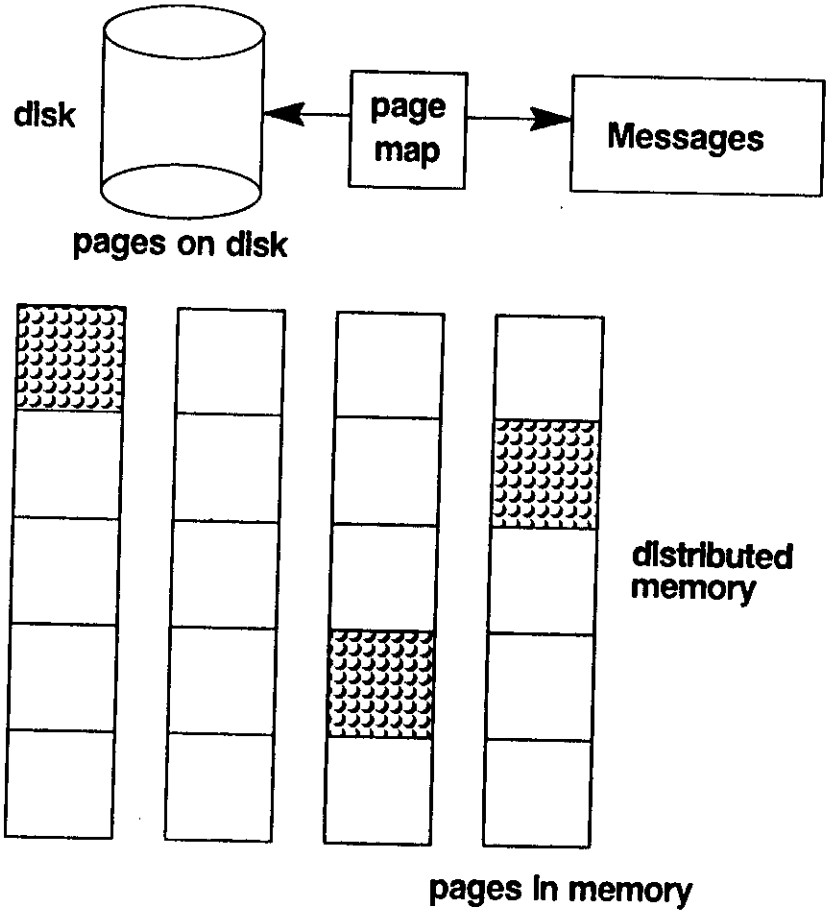
Models – shared and distributed memory

virtual memory with paging in shared memory model



Distributed memory looking like Shared memory

Leslie Vallent proved – with parallel slackness the communication can be hidden.



Parallel Computing

Ian Willers

Why is it interesting?

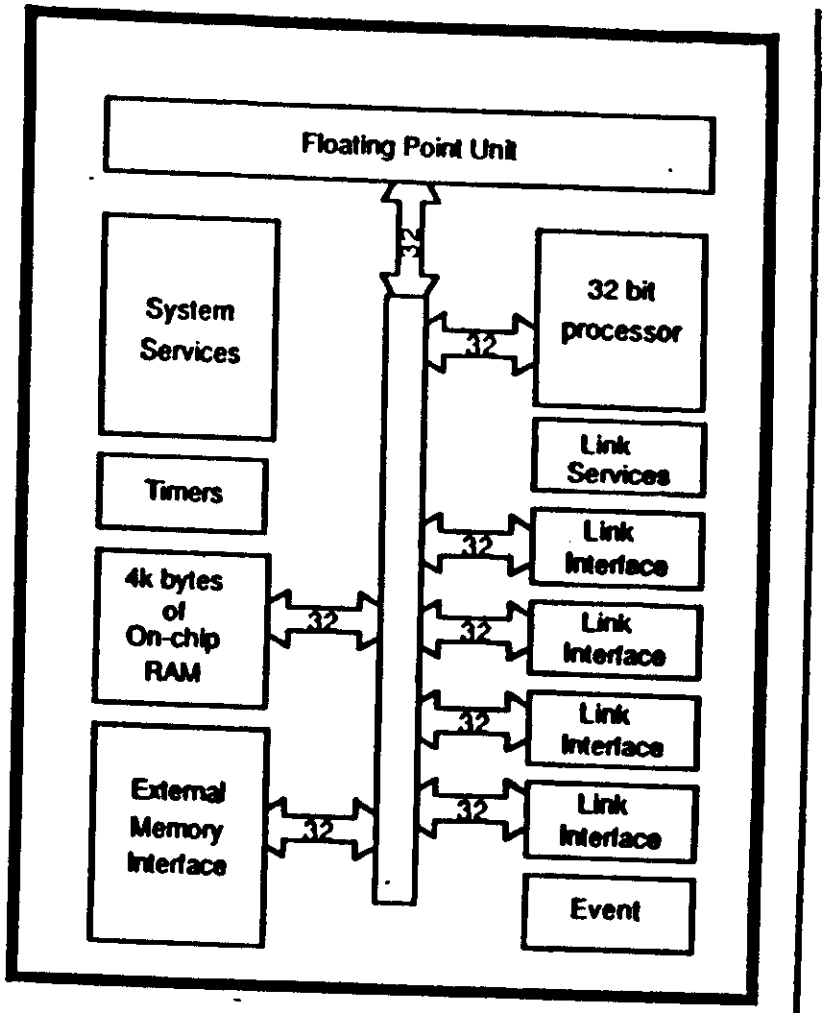
Classification of Architectures;

Common Practical Models;

Three concrete examples.

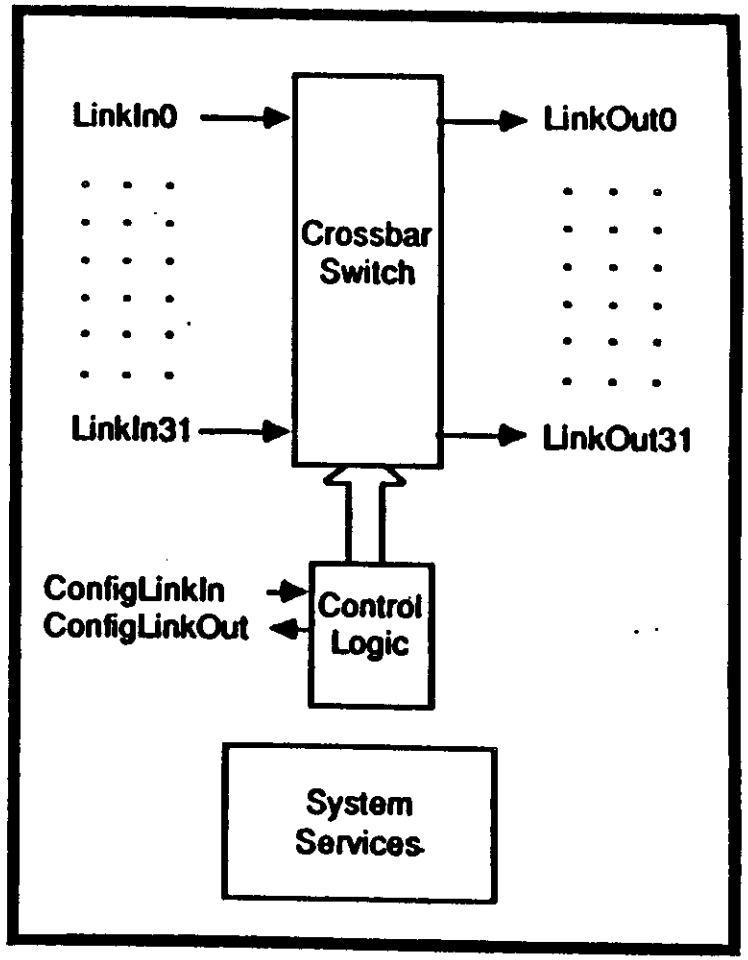
IMST801 transputer

- 32 bit architecture
- 30 MIPS (peak)
- On-chip IEEE 754 FPU
- 4.3 Mflops (peak)
- 630 ns interrupt response
- De-mulitplexed address and data bus
- 60 Mbytes/sec data rate to external memory
- 4 Kbytes on-chip SRAM
- 4 high speed serial links (5/10/20 Mbits/sec)
- 20 , 25 AND 30 MHz
- 100 pin PGA packages



IMS C004 programmable link switch

- Standard INMOS serial links
- 32 way crossbar switch
- Cascadable
- 10 or 20 Mbits/sec
- 84 pin PGA package



Inventing the future

TPC0045/89

occam Tony Hoare: Communicating Sequential Processes, CSP.

1) Sequential code:

```
SEQ
  a = 1
  b = 2
```

2) Parallel code:

```
PAR
  a = 1
  b = 2
```

3) Messages:

```
a ! 3
```

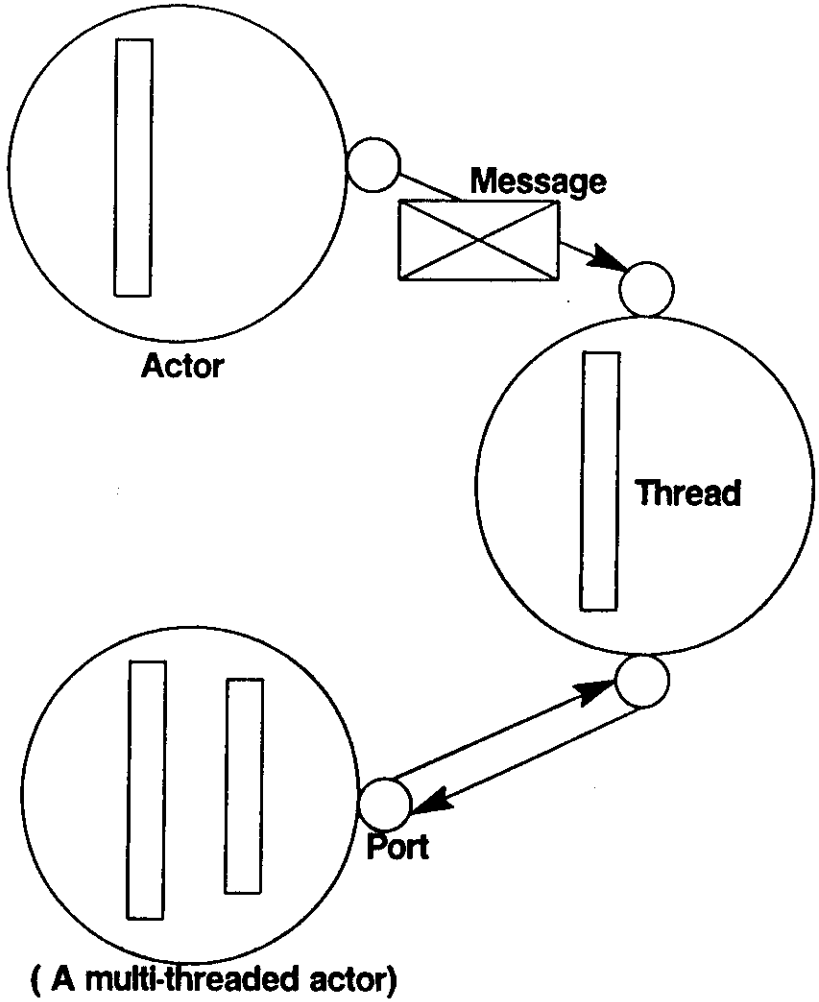
and

```
a ? value
```

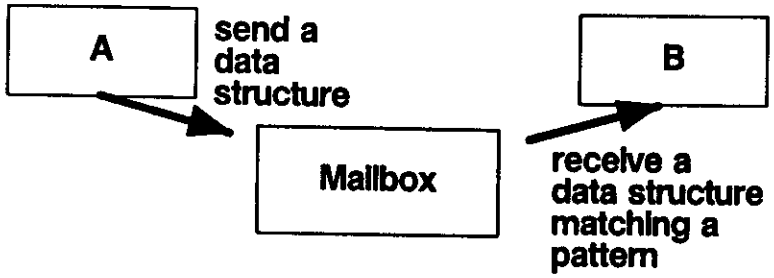
4) Aternative routes:

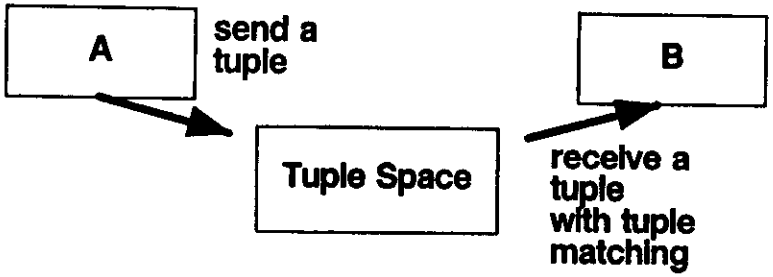
```
ALT
  ( d = 0 ) a ? value
                <code>
  ( d > 0 ) b ? value
                <code>
  ( d > 0 ) c ? value
                <code>
```

The Chorus Operating System



LINDA





LINDA

To put a tuple in tuple space:

```
I=2
out ( 1, 1.5, I )
```

To get a tuple from tuple space:

In (1, 1.5, 2) succeeds

In (1, 1.5, 3) blocks

In (1, f, 2) succeeds and sets f to 1.5

To read a tuple from tuple space:

rd (1, 1.5, 2) succeeds

etc.

Note: type and value matches are required

1) One can acheive high performance on essentially all scientific computations which are:

large (necessary condition)
loosely synchronous – MIMD or
synchronous – SIMD

2) Domain decomposition or data parallelism is a universal source of parallelism that scales to a large number of nodes

3) Greatest success has come from 1'000 to 10'000 line codes written from scratch for a particular machine. Usually the application scientist can specify parallelism from the natural geometrical structure of the problem.

4) Performance rules :–

Performance scales linearly in number of nodes at constant grain size (problem size proportional to machine size).

Fixed problem size does not scale by Amdahl's law.

5) A key question is: 'What is the appropiate productive standard programming environment for parallel machines?'

New languages and approaches (e.g. graphical techniques);

Compiler generated parallelism;

Application specific high-level environments;

Explicit user decomposition.

Parallel Computing

Ian Willers

Why is it interesting?

Classification of Architectures;

Common Practical Models;

Three concrete examples.

The End