Supercomputing 2007, www.top500.org

#1 : IBM Blue Gene
500 TFlops/sec] Main memory → 74 GB

213,000 processors.

→ Vendors : IBM 46%
HP 33%

→ Multicore processors (Intel Clovertown quad-core most popular)

→ 71% Intel

16% AMD Opteron

12% IBM power processor

→ 81% cluster architecture

→ Interconnects : Gigabit Ethernet, Infiband

→ LINPACK / HPC benchmarks

(HPL)	→ LINPACK, linear system of eqns
(DGEMM)	Floating pt rate for double precision matrix-matrix mult
(STREAM)	sustainable memory BW for simple vector kernel
(PTRANS)	Parallel matrix transpose
(Random Access)	integer random updates of memory
(FFT)	FFT
(B_EFF)	latency & BW of simultaneous communication patterns

- CPU speed limitations → instruction execution rate
  CPU ⟷ memory rate

→ memory interleaving, cache
→ instr and execution pipelining
→ superscalar execution: data/resource/branch dependencies
→ VLIW processors & IA64
"instrs that can be concurrently executed are packed

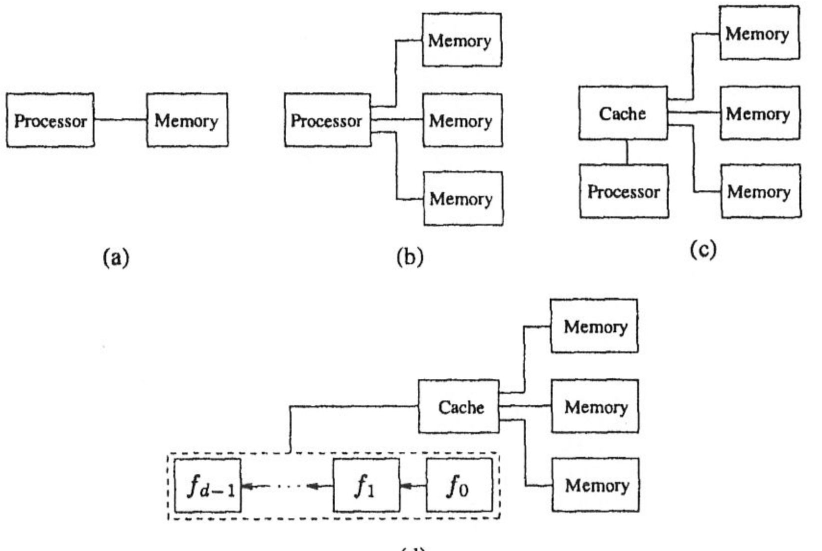


Figure 2.1 The evolution of a typical sequential computer: (a) a simple sequential computer; (b) a sequential computer with memory interleaving; (c) a sequential computer with memory interleaving and cache; and (d) a pipelined processor with *d* stages.
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

in a single long instruction word & executed on multiple functional units at same time"
requires extensive compiler support
loop unrolling, branch prediction, speculative execution

→ VLIW & superscalar processors:
exploit implicit parallelism
small scale of concurrency

SIMD eg

Illiac IV, MPP, DAP,
CM-2, MasPar
MP-1 & MP-2

MIMD eg

Cosmic Cube, nCube
iPSC, Symmetry,
FX-series
TC-2000, CM-5
KSR-1, Paragon XP/S

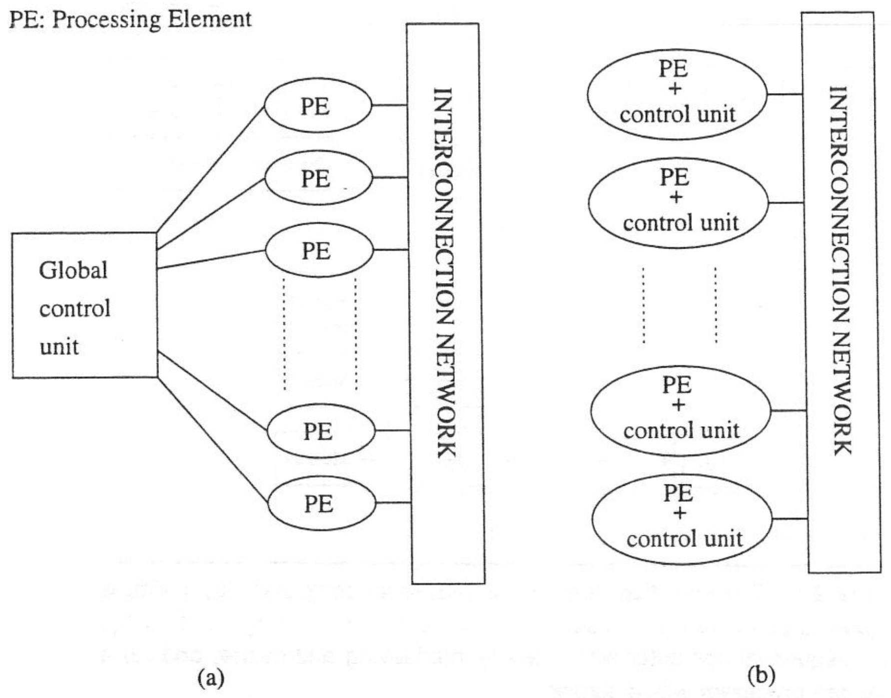


Figure 2.2 A typical SIMD architecture (a) and a typical MIMD architecture (b).
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

SIMD : Single Instruction stream Multiple Data stream

MIMD : Multiple '' '' '' '' ''

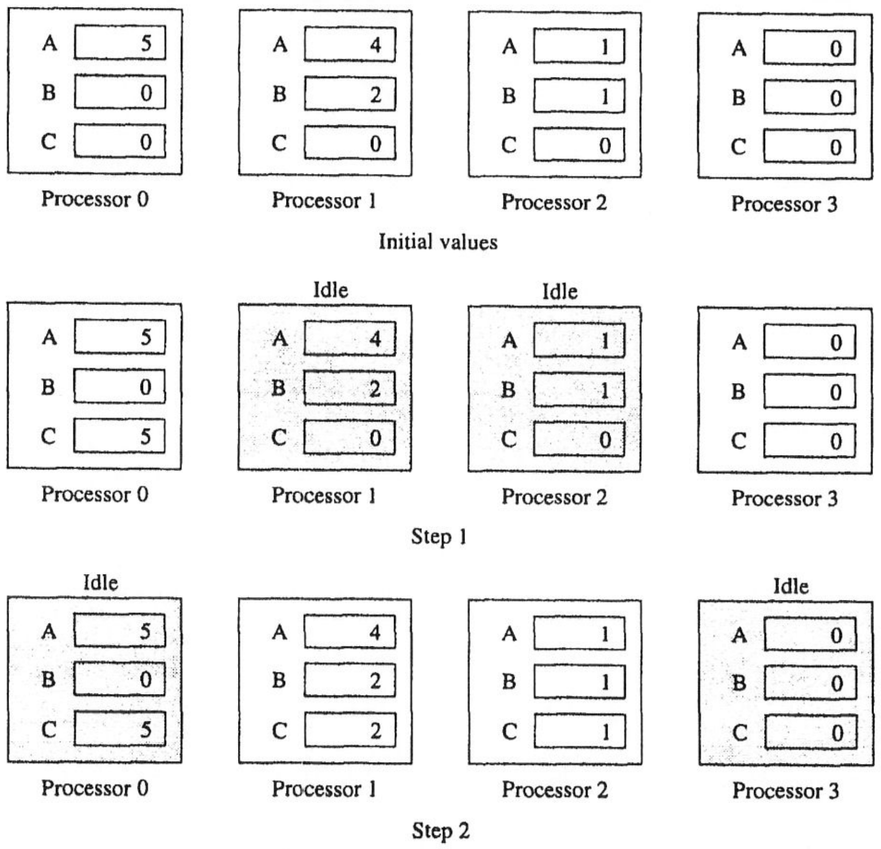
SISD : Single '' '' Single '' ''

MISD : Multiple '' '' ~~Sing~~ '' '' ''

FLYNN's taxonomy.

```
if (B == 0)
        C = A;
else
        C = A/B;
```

(a)



(b)

Figure 2.3 Executing a conditional statement on an SIMD computer with four processors: (a) The conditional statement; (b) The execution of the statement in two steps.
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

different processors cannot execute different instrs
in the same clock cycle

distributed memory or private memory architecture

→ NUMA like

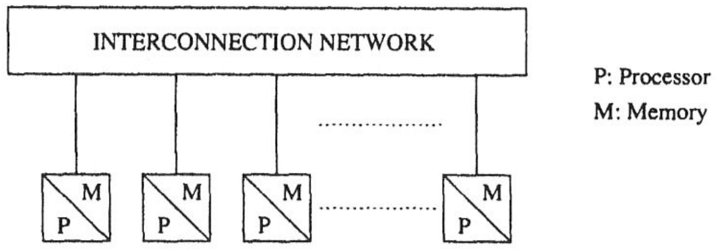


Figure 2.4 A typical message-passing architecture.
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

- BW of IC network must be substantial
  - → Conflicts
  - → multiple stages of IC
- UMA vs NUMA
- cache coherence

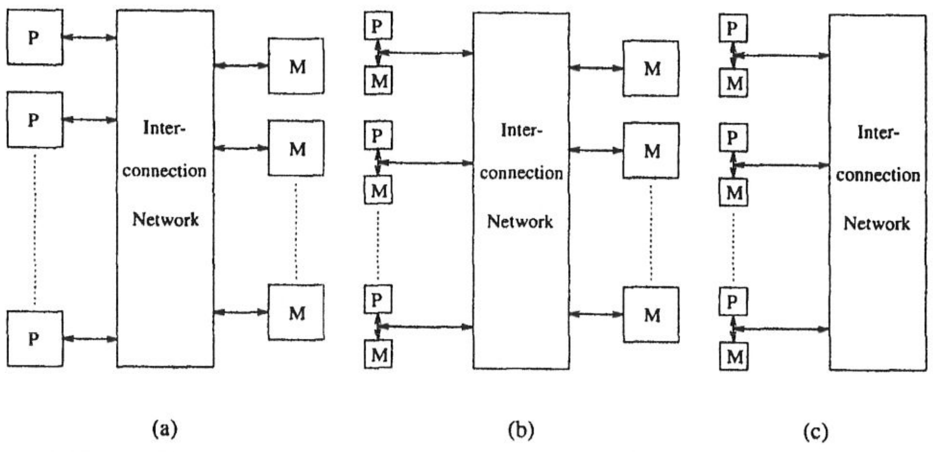


Figure 2.5 Typical shared-address-space architectures: (a) Uniform-memory-access shared-address-space computer; (b) Non-uniform-memory-access shared-address-space computer with local and global memories; (c) Non-uniform-memory-access shared-address-space computer with local memory only.
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

- shared memory NUMA vs msg-passing
  - → NUMA provides HW support for R/W to remote memories
    msg-passing : remote access emulated by explicit msg-passing
- easy to emulate msg-passing arch. by shared-memory arch.
- reverse is difficult

- nonblocking network
- $b \geq p$

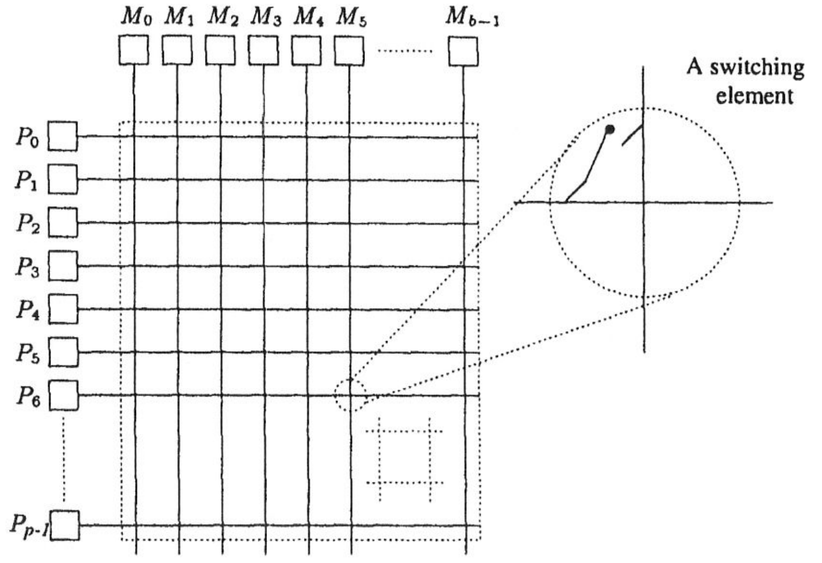


Figure 2.6 A completely nonblocking crossbar switch connecting p processors to b memory banks.
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

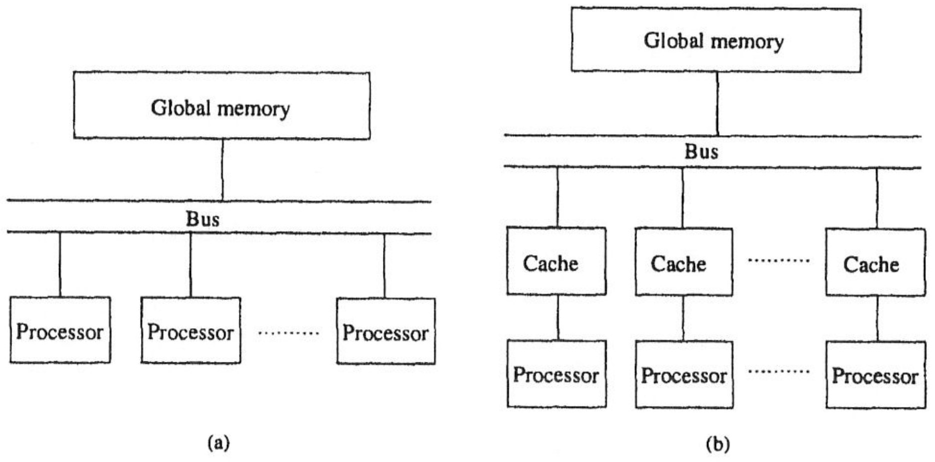


Figure 2.7 A typical bus-based architecture with no cache (a) and with cache memory at each processor (b).

• Crossbar : scalable w.r.t. performance ; not scalable w.r.t. cost

• Shared bus : NOT scalable " " ; scalable w.r.t. cost

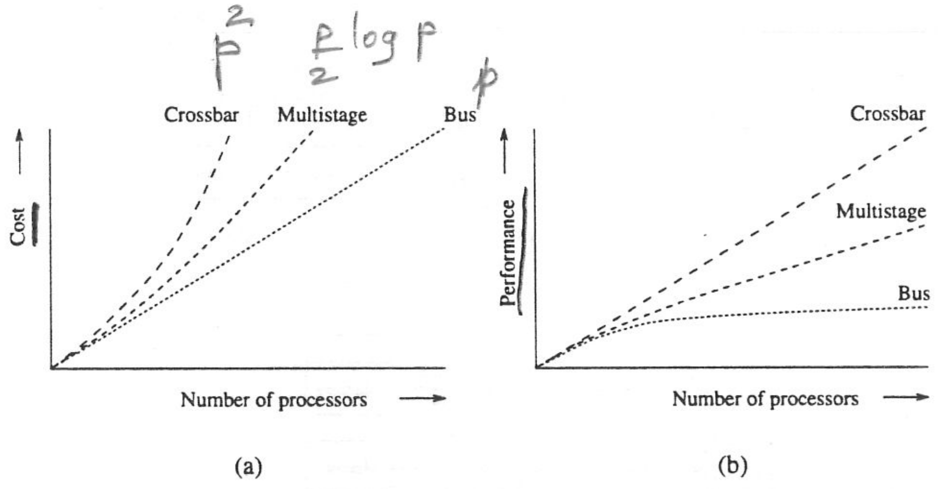


Figure 2.9 (a) Cost versus number of processors for interconnection networks based on bus, multistage, and crossbar connected networks; (b) Performance versus number of processors for the three networks.
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

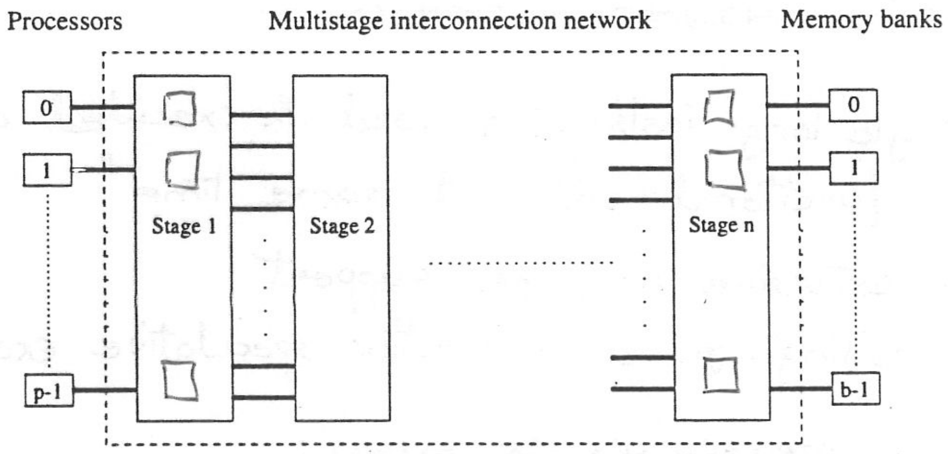


Figure 2.8 The schematic of a typical multistage interconnection network.
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

2X2 crossbars; $p/2$ such

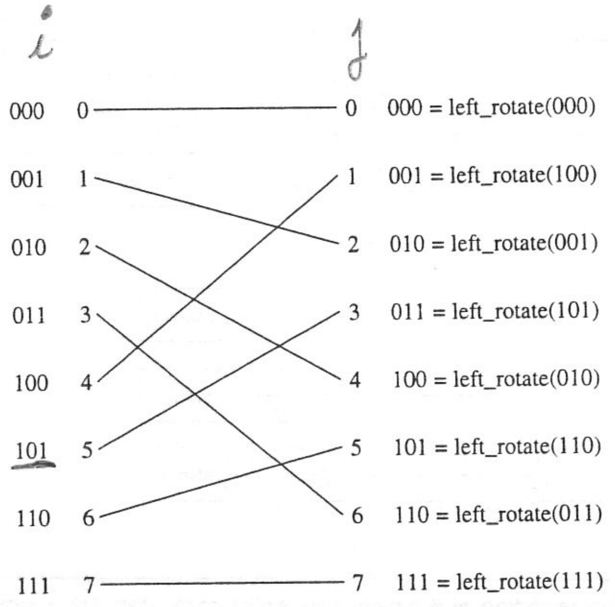
$n = \log p \ (= m)$

- Input i, output j

$$j = \begin{cases} 2i & 0 \le i \le p/2 - 1 \\ 2i + 1 - p & p/2 \le i \le p-1 \end{cases}$$

Left-rotation on binary representation of i

- Omega network : $\frac{p}{2} \times \log p$ switching elements



i
j
000 0 — 0 000 = left_rotate(000)
001 1 — 1 001 = left_rotate(100)
010 2 — 2 010 = left_rotate(001)
011 3 — 3 011 = left_rotate(101)
100 4 — 4 100 = left_rotate(010)
101 5 — 5 101 = left_rotate(110)
110 6 — 6 110 = left_rotate(011)
111 7 — 7 111 = left_rotate(111)

Figure 2.10 A perfect shuffle interconnection for eight inputs and outputs.
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

→ Omega, butterfly, baseline

- $s_i \rightarrow s_j$ iff $j = i$
  at level l $\quad j = i \oplus (2^{\log p - l})$
  "Butterfly network" // p = # processors
- Other networks
  → banyan, Benes,

$\underline{l=1}$
$j = i \oplus 2^{3-1}$
$= i \oplus 100$

$\underline{l=2}$
$j = i \oplus 010$

$l=3$
$j = i \oplus 001$

$M = n/2$ switches/stage
switch $\langle x, s \rangle$, where $x \in [0, M-1]$, stage $s \in [0, \log_2 n - 1]$

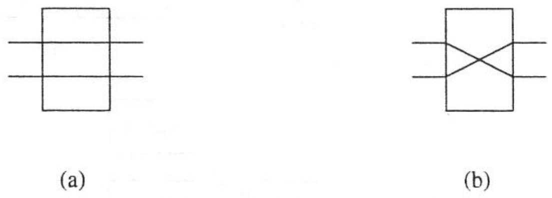
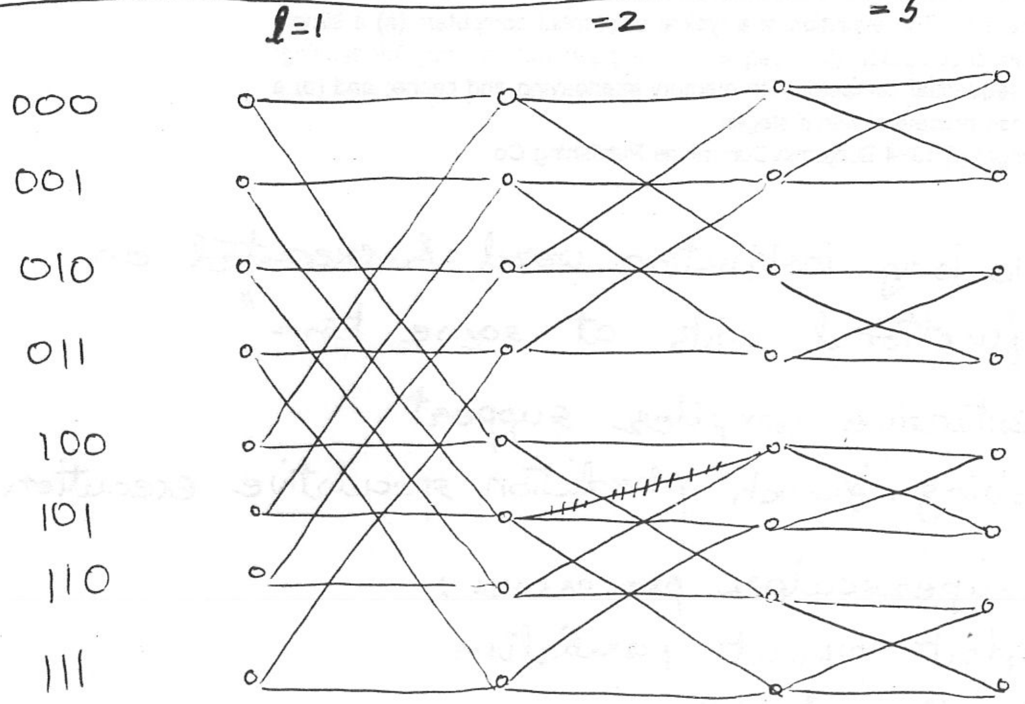


Figure 2.11 Two switching configurations of the 2×2 switch: (a) Pass-through; (b) Cross-over. Copyright (r) 1994 Benjamin/Cummings Publishing Co.

- Benes



Edge from $\langle x, s \rangle$ to $\langle y, s+1 \rangle$ if
i) $x = y$, or
ii) $x \oplus y$ has only one 1 bit, which is in $(s+1)^{th}$ MSB.

For stage s, apply above rule to $M/2^s$ switches.

eg. 101
$\oplus$ 010

- Routing in stage i uses i^{th} bit of destination
- Ω network used in BBN Butterfly, IBM RP-3, NYU Ultracomputer

eg: 001 to 010

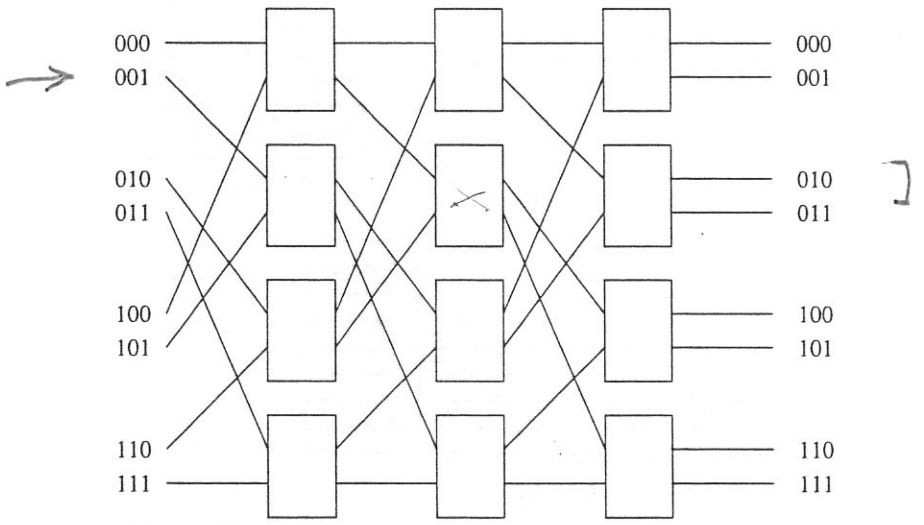
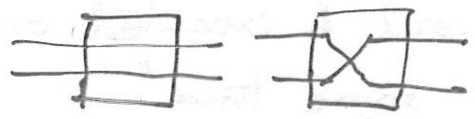


Figure 2.12 A complete omega network connecting eight inputs and eight outputs .
Copyright (r) 1994 Benjamin/Cummings Publishing Co.



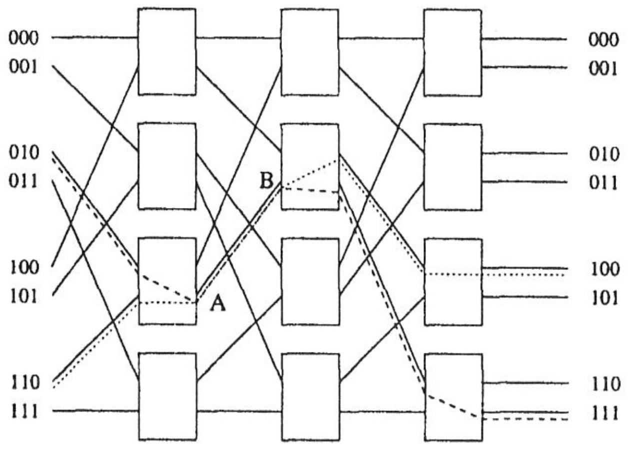
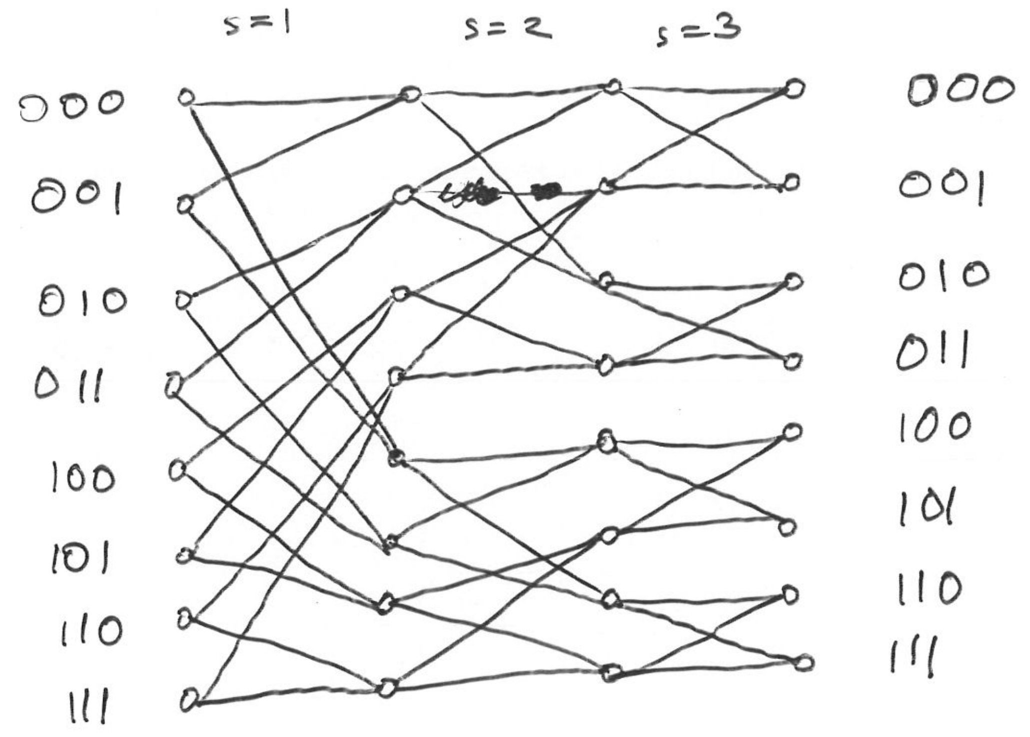


Figure 2.13 An example of blocking in omega network: one of the messages (010 to 111 or 110 to 100) is blocked at link AB.

Baseline Network



$$\cancel{j_1 = \left\lfloor \frac{i}{2} \right\rfloor}$$

$$\cancel{j_2 = \left\lfloor \frac{i}{2} \right\rfloor + 2^{\log p - s}}$$

$$K = \log p - (s-1)$$

$$J_1 = \left\lfloor \frac{i}{2^K} \right\rfloor \cdot 2^K + \left\lfloor \frac{i \bmod 2^K}{2} \right\rfloor = i \,\&\, \sim(2^K - 1) + (i \,\&\, (2^K-1)) >> 1$$

$$J_2 = J_1 + 2^{K-1}$$

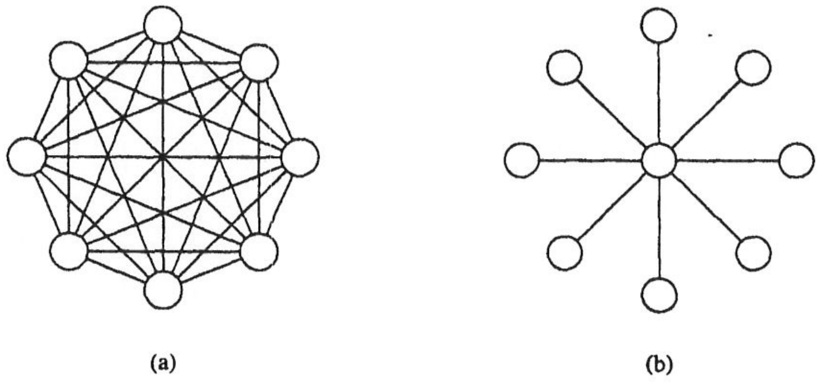


Figure 2.14 A completely-connected network of eight processors (a), and a star-connected network of nine processors (b).
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

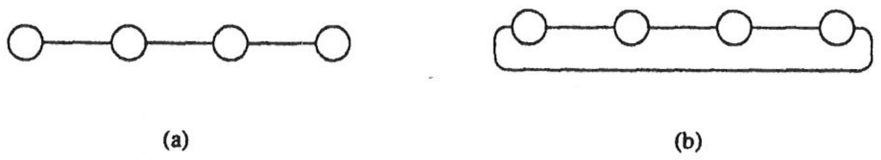


Figure 2.15 A four-processor linear array (a) and a four-processor ring (b).
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

- 2-D meshes : DAP, Paragon XP/S
- 3-D meshes : Cray T3D, J-machine

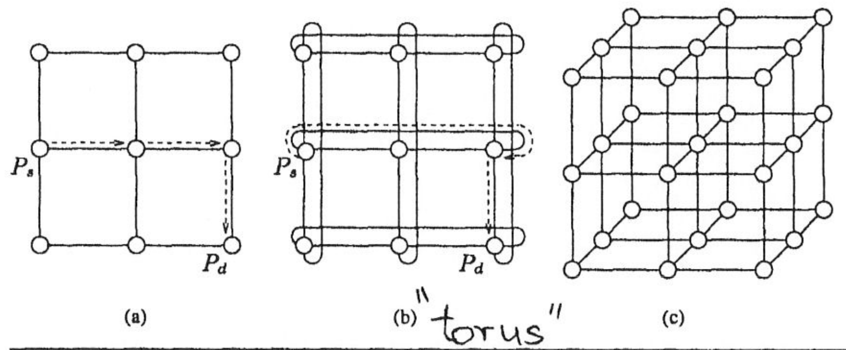


Figure 2.16 (a) A two-dimensional mesh with an illustration of routing a message from processor P_s to processor P_d; (b) a two-dimensional wraparound mesh with an illustration of routing a message from processor P_s to processor P_d; (c) a three-dimensional mesh.

communication bottleneck at higher levels.

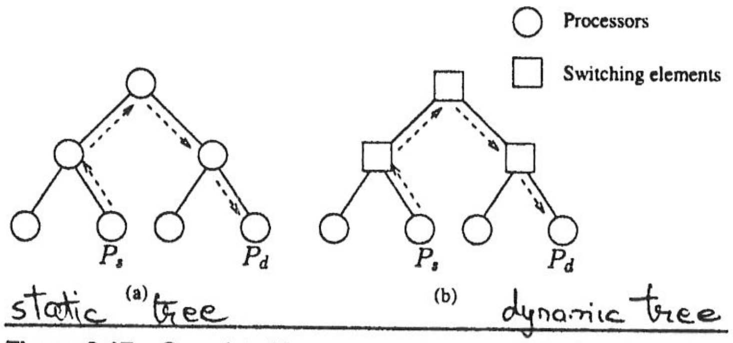


Figure 2.17 Complete binary tree networks and message routing in them.
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

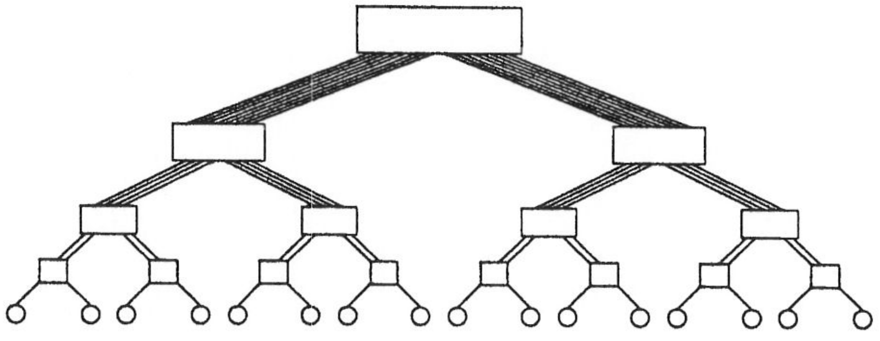


Figure 2.18 A fat tree network of 16 processors.
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

(1) $P_i \overset{edge}{—} P_j$ iff binary rep. differ in 1 bit position

(2) each proc. connected to d. other procs

(3) d-dim hypercube split into 2-(d-1)-dim hypercubes in d ways

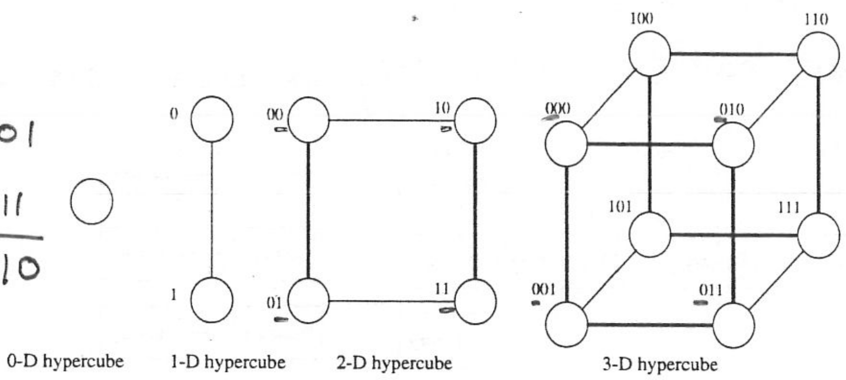
(4) By fixing k/d bits, we have (d-k)-dim HC
How many such HCs? 2^k

d dim HC
2^d proc.

d bit address.

2D cube

s = 01	s = 01
d = 10	d = 11
⊕ 11	⊕ 10



'd' links

⊕ 1110
0101 s

1011 d

R-to-L

W Z X

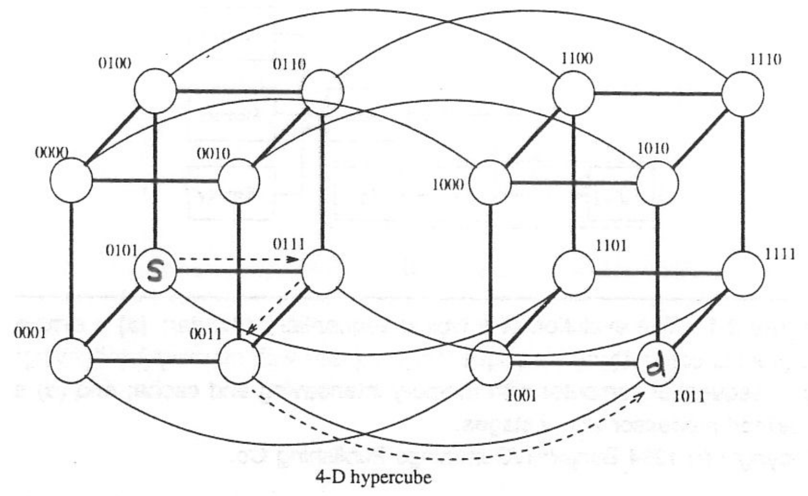


Figure 2.19 Hypercube-connected architectures of zero, one, two, three, and four dimensions. The figure also illustrates routing of a message from processor 0101 to processor 1011 in a four-dimensional hypercube.
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

mesh w/ 2 procs in each dimension

(5) # communic. links in shortest path ≡ Hamming distance
s ⊕ t : route along dimensions where the corresp. bit position = 1

cf nCUBE 2, Cosmic Cube, iPSC

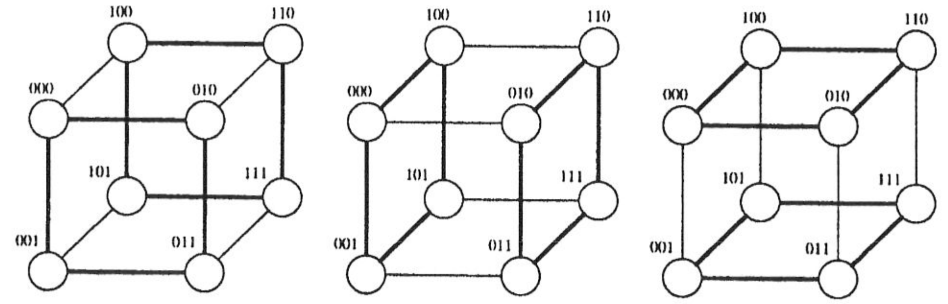


Figure 2.20 Three distinct partitions of a three-dimensional hypercube into two two-dimensional cubes. Links connecting processors within a partition are indicated by bold lines.

k-ary d-cube networks

k → radix

d → dimension

radix = # processors along each dimension

- d-dim hypercube ≡ binary d-cube
- ring of p processors ≡ p-ary 1-cube
- 2D wraparound mesh of p proce ≡ ~~k-ary~~ $\sqrt{p}$ ary 2-cube
- k-ary d-cube ≡ constructed from k k-ary (d-1) cubes by connecting the processors that occupy identical positions in the cubes into rings

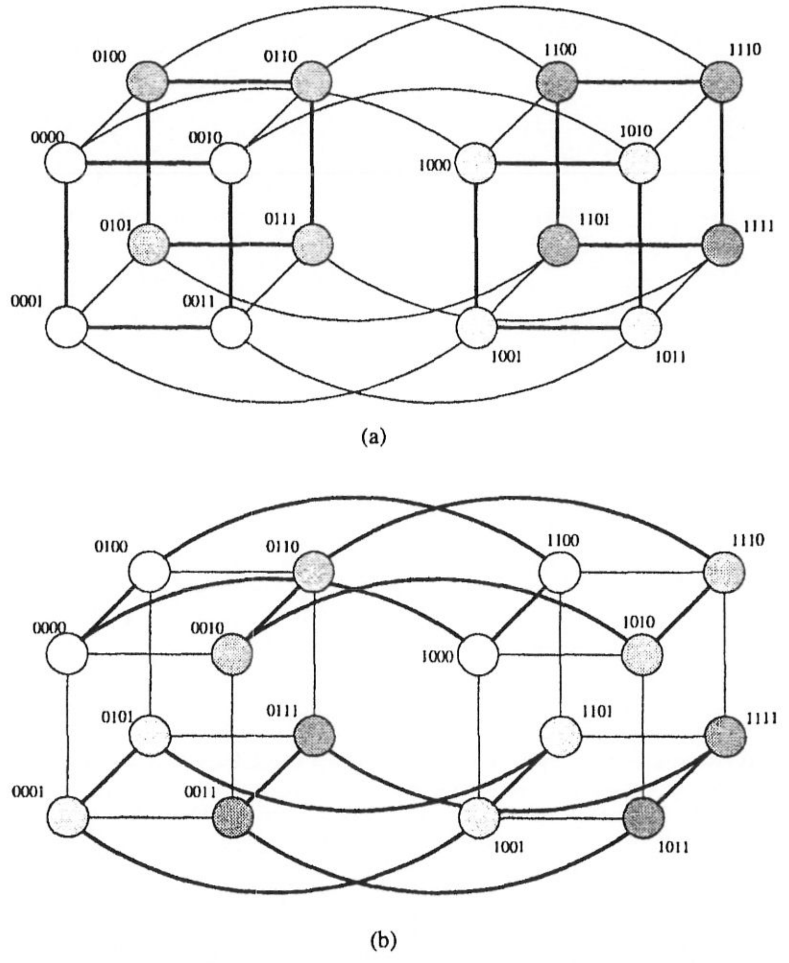


Figure 2.21 The two-dimensional subcubes of a four-dimensional hypercube formed by fixing the two most significant label bits (a) and the two least significant bits (b). Processors within a subcube are connected by bold lines.

Table 2.1 A summary of the characteristics of various static network topologies connecting p processors.

Network	Diameter	Bisection Width	Arc Connectivity	Cost (No. of links)
Completely-connected	1	$p^2/4$	$p-1$	$p(p-1)/2$
Star	2	1	1	$p-1$
Complete binary tree	$2\log((p+1)/2)$	1	1	$p-1$
Linear array	$p-1$	1	1	$p-1$
Ring	$\lfloor p/2 \rfloor$	2	2	p
2-D mesh without wraparound	$2(\sqrt{p}-1)$	$\sqrt{p}$	2	$2(p-\sqrt{p})$
2-D wraparound mesh	$2\lfloor \sqrt{p}/2 \rfloor$	$2\sqrt{p}$	4	$2p$
Hypercube	$\log p$	$p/2$	$\log p$	$(p \log p)/2$
Wraparound k-ary d-cube	$d\lfloor k/2 \rfloor$	$2k^{d-1}$	$2d$	dp

Connectivity ≡ measure of multiplicity of paths between any 2 procs

Arc connectivity ≡ min. # arcs that must be removed to break network into 2 parts

Bisection width ≡ min. # links that have to be removed to partition network into 2 equal halves

• bisection ~~channel~~ BW = bisection ~~channel~~ width * channel BW

Diameter : $\max \left[\min\_length_{\forall i \forall j} (i,j) \right]$

<u>Embedding</u> networks into a hypercube

• without embedding, algorithm designed for a <u>specific graph</u> may have to be adapted to <u>another graph</u>

$(V,E) \rightarrow (V',E')$: max #. edges mapped to any edge in E' ≡ <u>congestion</u>

: max #. links in E' that any edge in E is mapped onto ≡ <u>dialation</u>

: $\frac{|V'|}{|V|}$ ≡ <u>expansion</u>

Embedding Linear Array into Hypercube

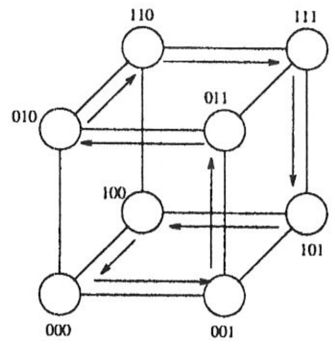
processor i of linear array $\longrightarrow$ processor $G(i,d)$ of HC

$$G(0,1) = 0$$
$$G(1,1) = 1$$
$$G(i,x+1) = \begin{cases} G(i,x) & i < 2^x \\ 2^x + G(2^{x+1}-1-i,\ x) & i \geq 2^x \end{cases}$$

1-bit Gray code	2-bit Gray code	3-bit Gray code	3-D hypercube	8-processor ring
0	0 0	0 0 0	0	0
1	0 1	0 0 1	1	1
	1 1	0 1 1	3	2
	1 0	0 1 0	2	3
		1 1 0	6	4
		1 1 1	7	5
		1 0 1	5	6
		1 0 0	4	7

Reflect along this line

(a)

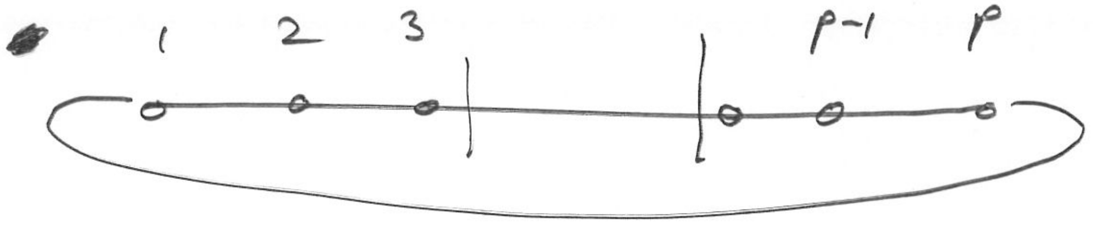


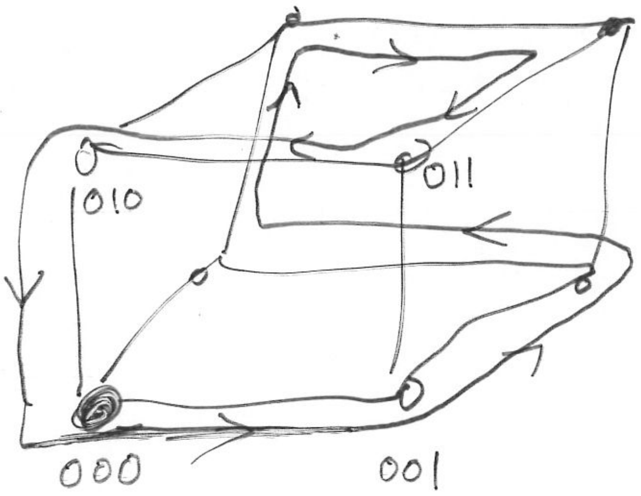
(b)

Figure 2.22 A three-bit reflected Gray code ring (a) and its embedding into a three-dimensional hypercube (b).
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

G ≡ binary reflected Gray code
$G(i,d)$ ≡ i^{th} entry in seq. of Gray codes of d bits

- mapping dialation = 1
  expansion = 1


1
2
3
p-1
p


010
011
000
001

$$\text{GRAY} = \text{BINARY} \oplus (\underset{\overset{\uparrow}{\text{LSR}}}{\text{BINARY}/2})$$

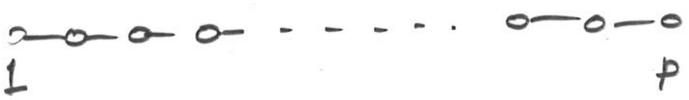
$$\text{BINARY} = 101 \quad \oplus$$

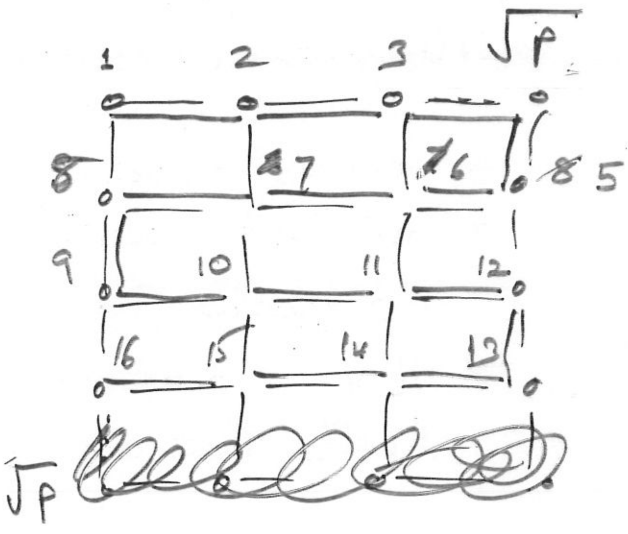
$$\begin{array}{r} 101 \\ \oplus\ 010 \\ \hline 111 \end{array}$$

GRAY 111

$$\begin{array}{r} 011 \\ \oplus\ 001 \\ \hline 010 \end{array} \quad \oplus$$

Array $\longrightarrow$ 2-D grid





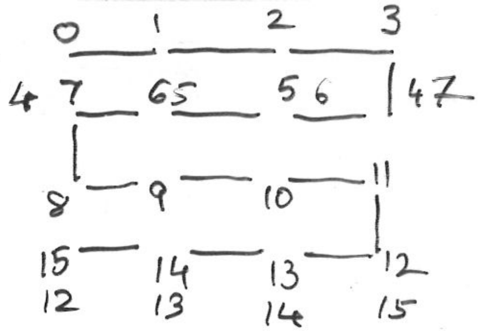
if $\left\lfloor \frac{i}{\sqrt{p}} \right\rfloor$ is even

$$i \longrightarrow i$$

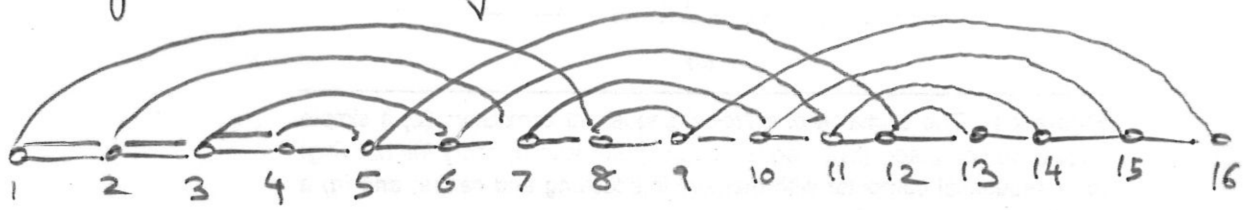
if $\left\lfloor \frac{i}{\sqrt{p}} \right\rfloor$ is odd.

[illegible] ?

$$i \longrightarrow \left\lfloor \frac{i}{\sqrt{p}} \right\rfloor \sqrt{p} + \left(\left\lfloor \frac{i}{\sqrt{p}} \right\rfloor + 1 \right) \sqrt{p} - i$$



(red edges) 2.D grid $\longrightarrow$ (blue edges) array.



Expansion $= 1$

Dialation $= 2\sqrt{p} - 1$

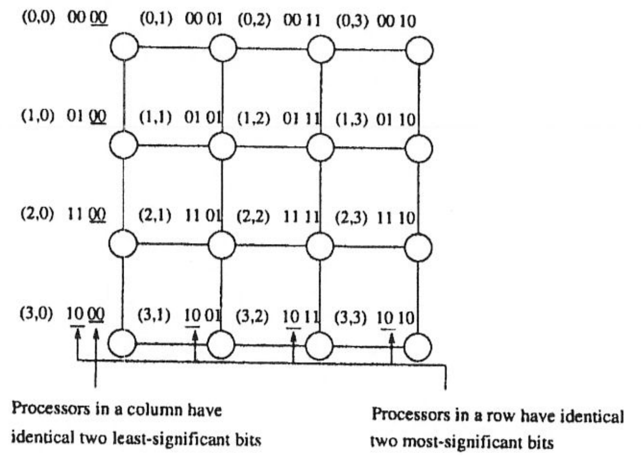
Congestion $= \sqrt{p} + 1$

Embedding a Mesh into a Hypercube [extension of ring embedding]

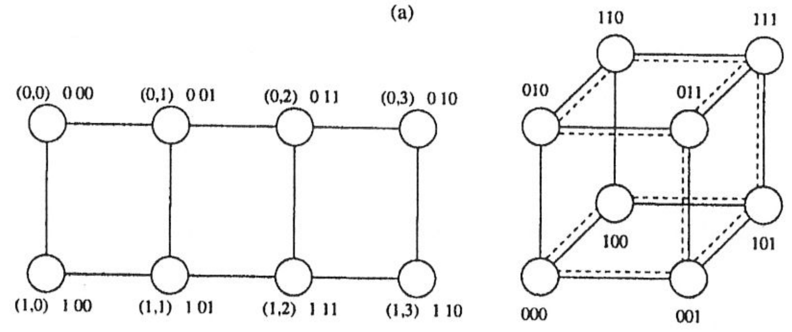
- $2^r \times 2^s$ wraparound mesh $\longrightarrow 2^{r+s}$ HC

  $(i,j)_{mesh} \longrightarrow G(i,r) \| G(j,s)_{HC}$

- dialation = 1, congestion = 1



(a)



(b)

Figure 2.23 (a) A 4 × 4 mesh illustrating the mapping of mesh processors to processors in a four-dimensional hypercube; and (b) a 2 × 4 processor mesh embedded into a three-dimensional hypercube. Copyright (r) 1994 Benjamin/Cummings Publishing Co.

- procs in same $row_{mesh} \longrightarrow procs_{HC}$ with labels having 'r' identical MSBs

  $row_{mesh} \longrightarrow$ distinct subcube

  $column_{mesh} \longrightarrow$ distinct subcube

Embedding Binary Tree into HC

- procs only at leaf nodes

(1) root $\longrightarrow$ any proc_{HC}

(2) $\forall$ node m at depth j

C_LEFT $(m) \longrightarrow$ same proc_{HC} to which m is mapped [let this be proc i]

C_RIGHT $(m) \longrightarrow$ proc_{HC} | we invert bit j of i

$(= \text{proc } i \oplus 2^{(j-1)})$

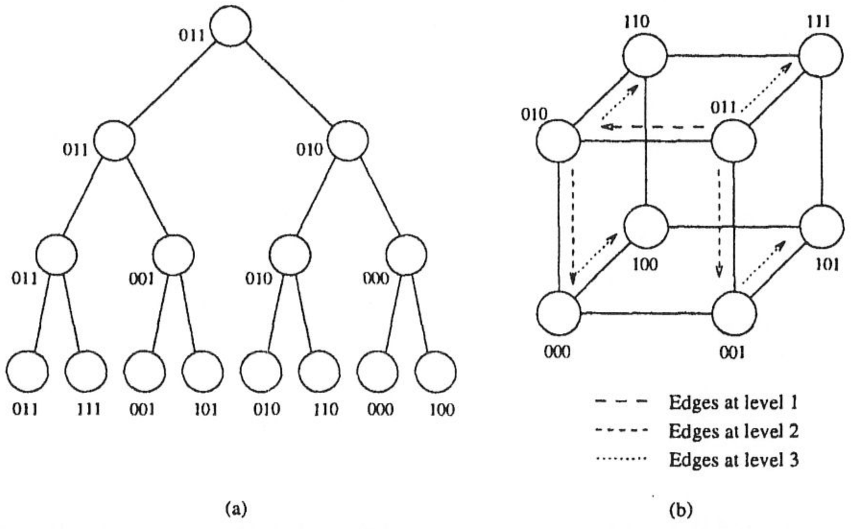


Figure 2.24 A tree rooted at processor 011 (=3) and embedded into a three-dimensional hypercube: (a) the organization of the tree rooted at processor 011, and (b) the tree embedded into a three-dimensional hypercube.
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

- In above mapping, expansion = 1

ROUTING MECHANISMS

- minimal vs non-minimal
- deterministic vs adaptive
- dimension-ordered routing eg XY routing, E-cube routing
- $P_s \longrightarrow P_d$.

  At any intermediate proc P_i:

  $\rightarrow$ fwd msg. along dimension corresp. least significant nonzero bit in $P_i \oplus P_d$

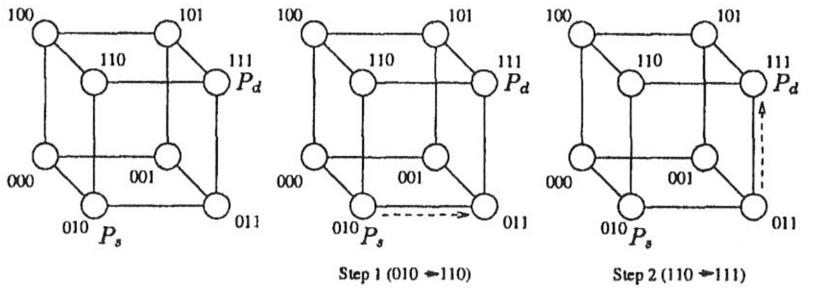


Figure 2.25 Routing a message from processor P_s (010) to processor P_d (111) in a three-dimensional hypercube using E-cube routing.
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

Communication Costs in Static I.N.

1) t_s (startup time) : once per msg
2) t_h (per hop time) $\equiv$ node latency; α [latency in switch to determine which o/p buffer/channel to use]
3) t_w (per word Xfer time) $\equiv \frac{1}{r}$, where r = channel BW

- Store-&-fwd: $t_{comm} = t_s + (mt_w + t_h)\,l = \Theta(ml)$
- Cut-through routing: msg advanced from incoming $\rightarrow$ outgoing link as it arrives

eg wormhole ~~[illegible]~~ routing

flow-control digits (flits): are pipelined

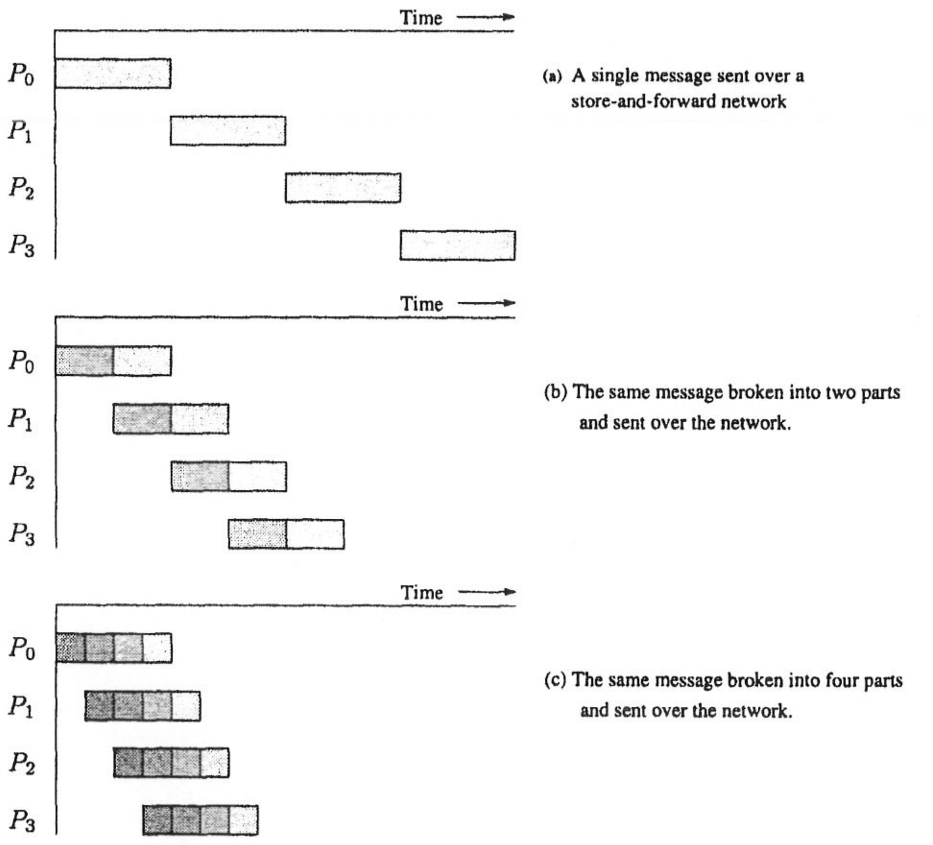

Time
P_0
P_1
P_2
P_3
(a) A single message sent over a store-and-forward network
(b) The same message broken into two parts and sent over the network.
(c) The same message broken into four parts and sent over the network.

Figure 2.26 Passing a message from processor P_0 to P_3 (a) through a store-and-forward communication network; (b) and (c) extending the concept to cut-through routing. The shaded regions represent the time that the message is in transit. The startup time associated with this message transfer is assumed to be zero.

- susceptibility to deadlock
  - $\rightarrow$ How avoided?
  - $\rightarrow$ E-cube routing & XY routing are deadlock-free

Cut-through: $t_{comm} = t_s + lt_h + mt_w = \Theta(m+l)$

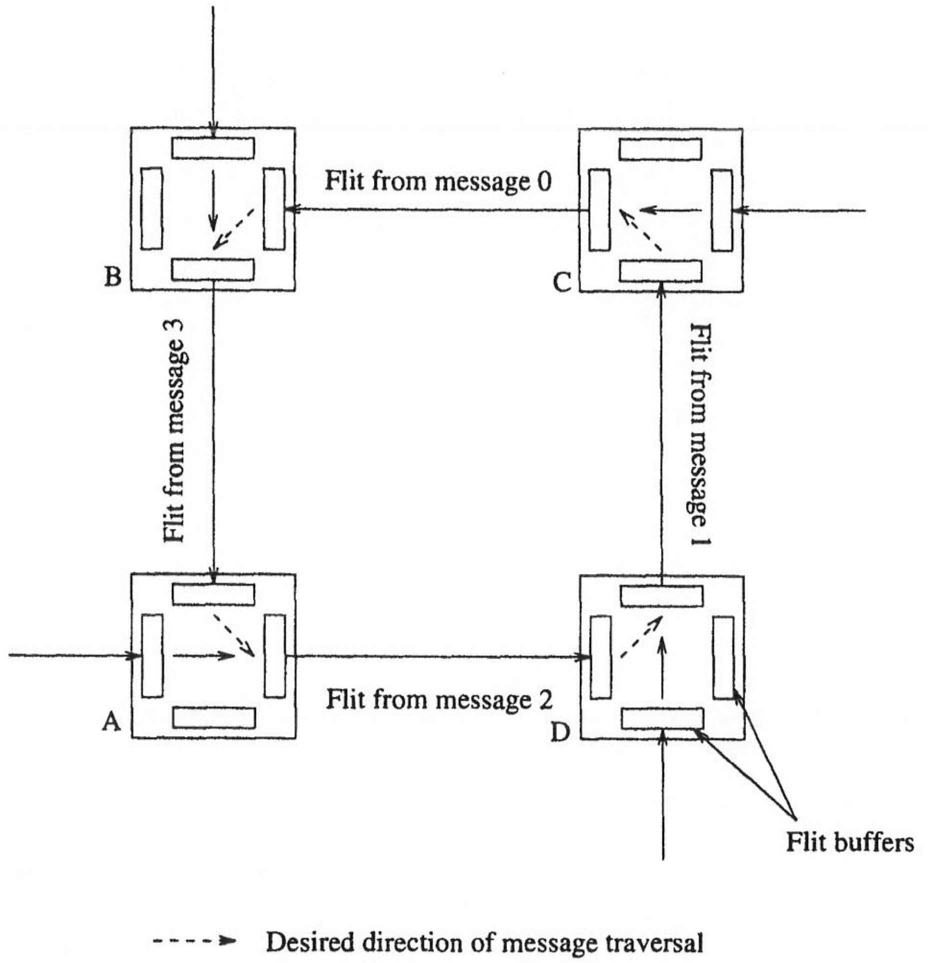


Figure 2.27 An example of deadlock in a wormhole-routing network.
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

Cost-Performance TradeOffs

- p-proc HC has $\frac{p\log p}{2}$ edges $\longrightarrow \frac{p\log p}{2}$ 'wires'
- p-proc wraparound mesh has $\frac{4p}{2}$ edges $\rightsquigarrow \left[\frac{\left(\frac{p\log p}{2}\right)}{\left(\frac{4p}{2}\right)}\right]$ ratio
- If cost $\propto$ # wires, mesh w/ $\left(\frac{\log p}{4}\right)$ wires/channel costs = HC w/ 1 wire/channel
- Avg dist : mesh $\cong \frac{\sqrt{p}}{2}$; HC $\cong \frac{\log p}{2}$
- With cut-through, [latency = $t_s + t_h l_{av} + t_w m$]

$$\text{Mesh} \cong t_s + t_h \frac{\sqrt{p}}{2} + \frac{4 m t_w}{\log p}$$

$$\text{HC} \cong t_s + t_h \frac{(\log p)}{2} + t_w m$$

- For light load, mesh is better ; heavy load, HC excels
  [ANALYSE]

- If cost $\propto$ bisection width, repeat above analysis

$$\text{mesh}: \; t_s + t_h \frac{\sqrt{p}}{2} + 4 m t_w / \sqrt{p}$$

$$\text{HC}: \; t_s + t_h \frac{(\log p)}{2} + t_w m$$

$\Rightarrow$ For $p > 16$ & sufficiently large message sizes
mesh outperforms HC $\longrightarrow$ at light loads
mesh comparable HC $\longrightarrow$ at high loads