• 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

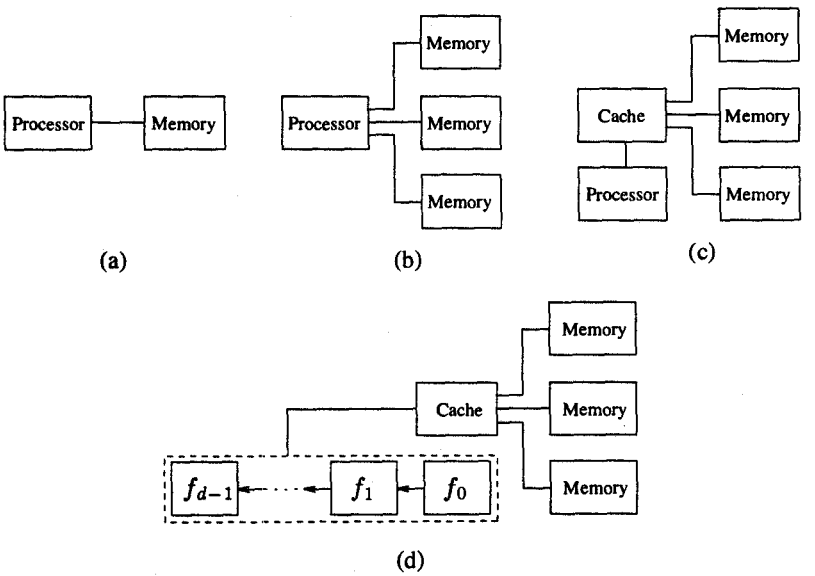


Figure 2.1 The evolution of a typical sequential computer: (a) a simple sequential computer; (b) a sequential computer with memory interleaving; (c) a sequential computer with memory interleaving and cache; and (d) a pipelined processor with d stages.

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

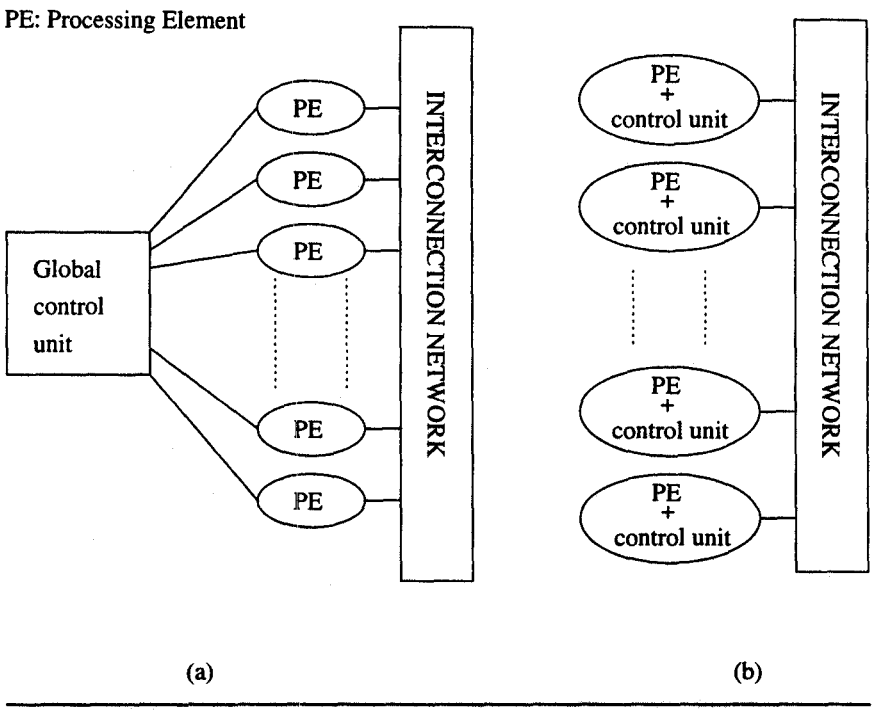


Figure 2.2 A typical SIMD architecture (a) and a typical MIMD architecture (b).

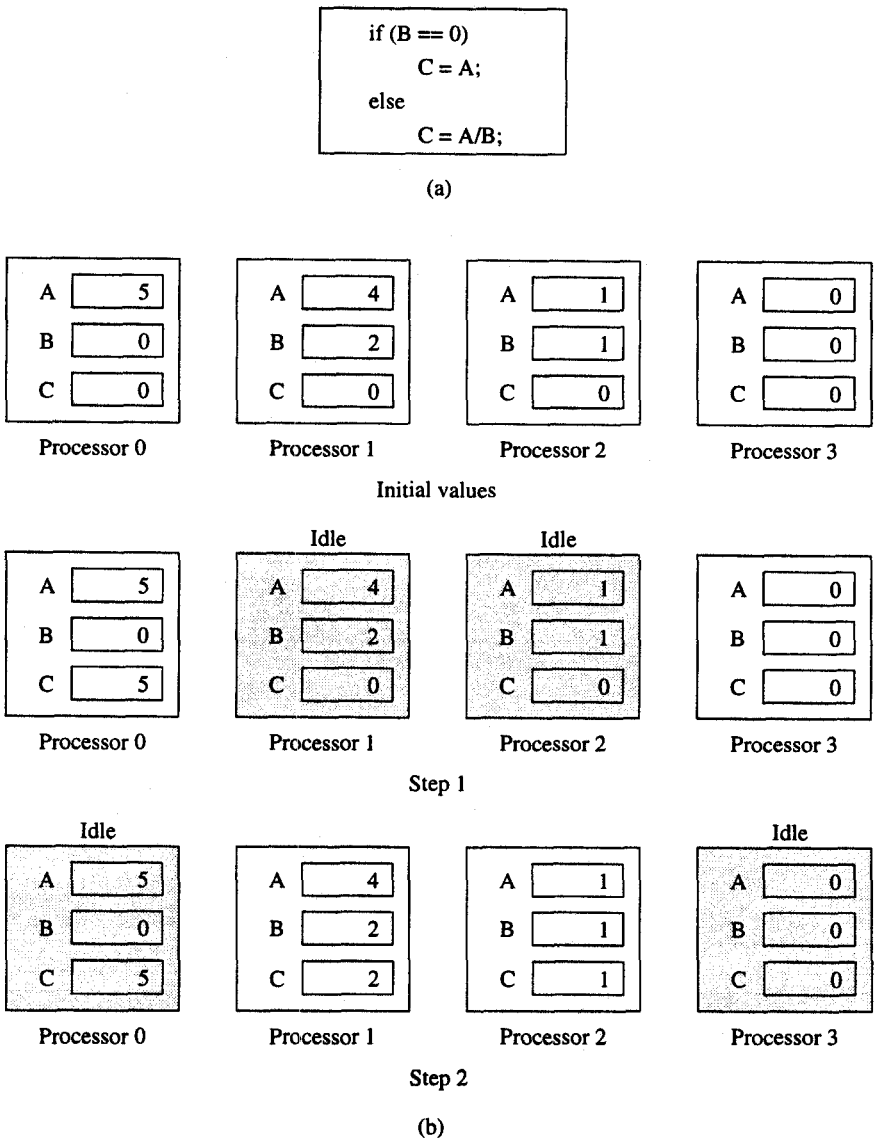


Figure 2.3 Executing a conditional statement on an SIMD computer with four processors: (a) The conditional statement; (b) The execution of the statement in two steps.

different processors cannot execute different instrs in the same clock cycle

distributed memory or private memory architecture

→ NUMA like

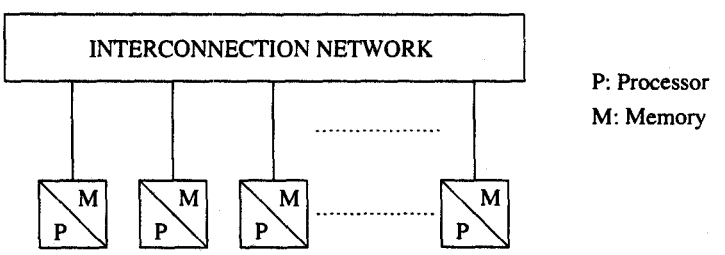


Figure 2.4 A typical message-passing architecture.

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

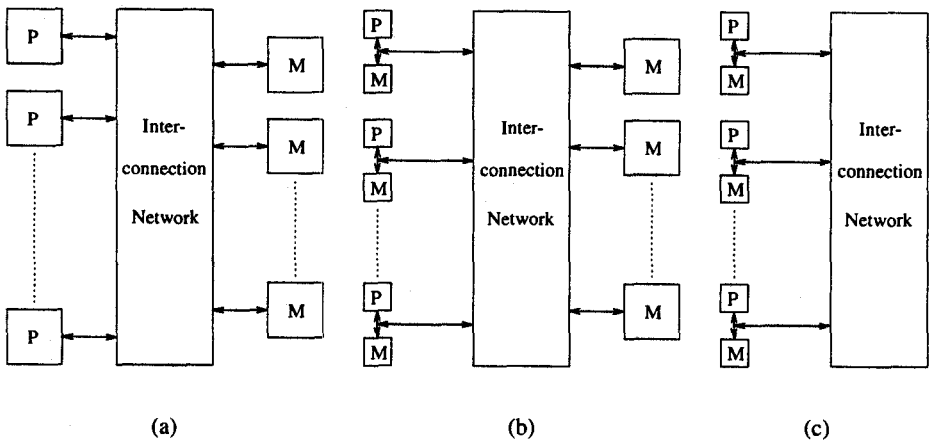


Figure 2.5 Typical shared-address-space architectures: (a) Uniform-memory-access shared-address-space computer; (b) Non-uniform-memory-access shared-address-space computer with local and global memories; (c) Non-uniform-memory-access shared-address-space computer with local memory only.

- 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
- $b \geq p$

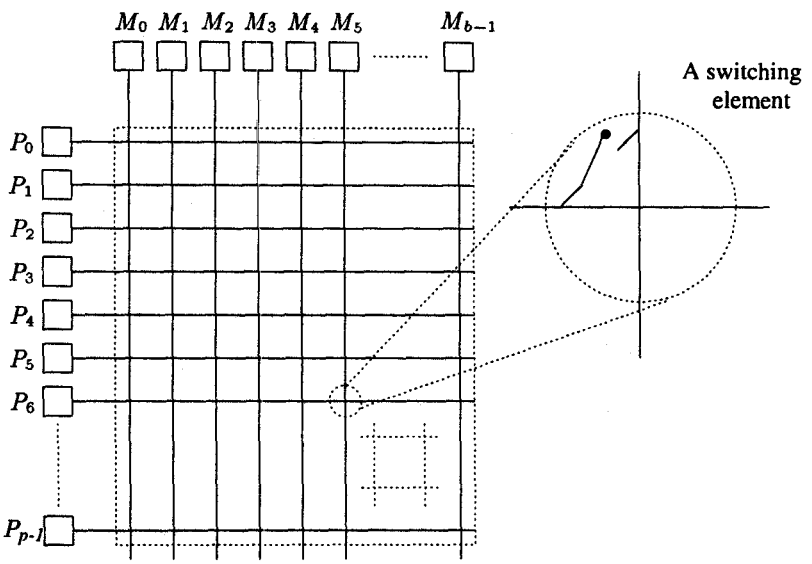


Figure 2.6 A completely nonblocking crossbar switch connecting p processors to b memory banks.

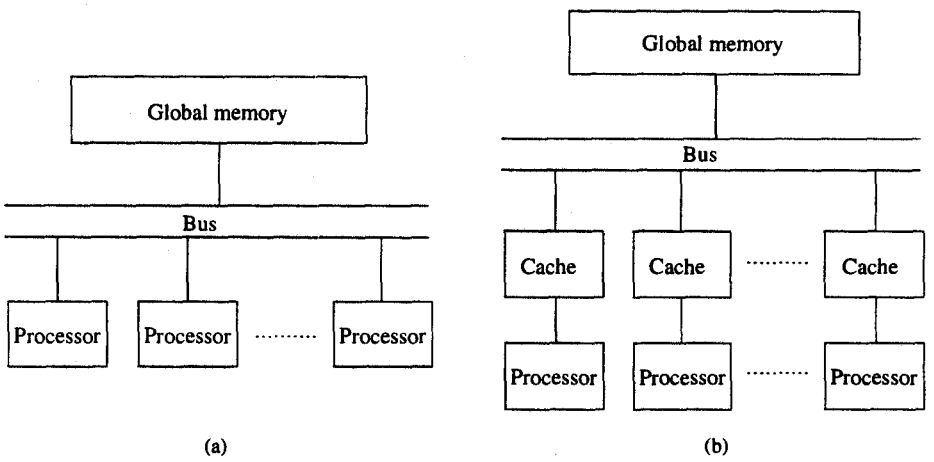


Figure 2.7 A typical bus-based architecture with no cache (a) and with cache memory at each processor (b).

- Crossbar : scalable w.r.t. performance ; not scalable w.r.t. cost
- Shared bus : NOT scalable " " ; scalable w.r.t cost

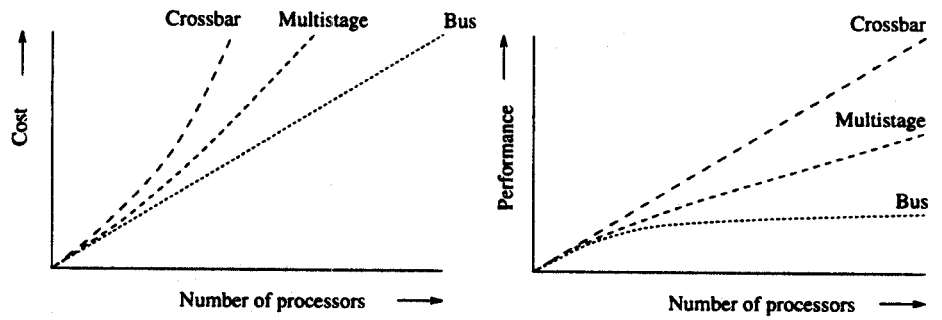


Figure 2.9 (a) Cost versus number of processors for interconnection networks based on bus, multistage, and crossbar connected networks; (b) Performance versus number of processors for the three networks.

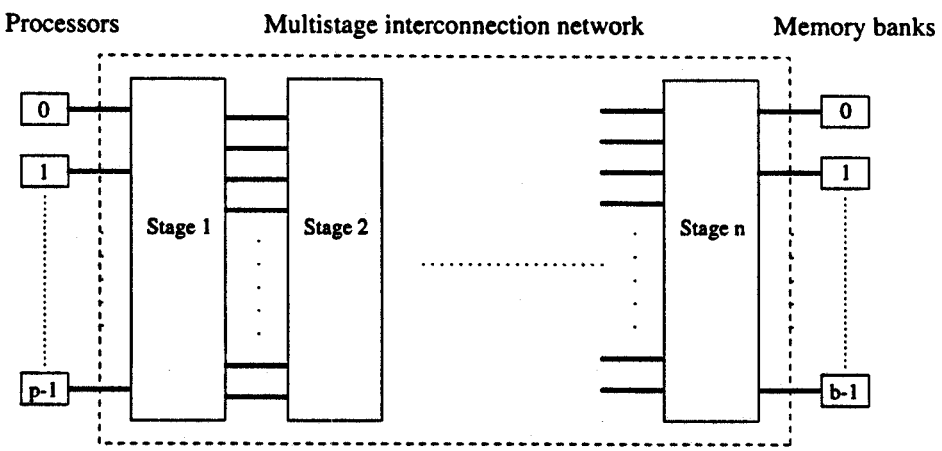


Figure 2.8 The schematic of a typical multistage interconnection network.

- 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

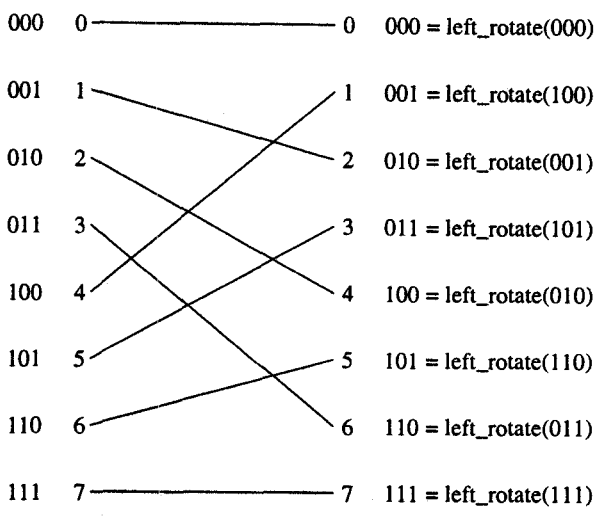


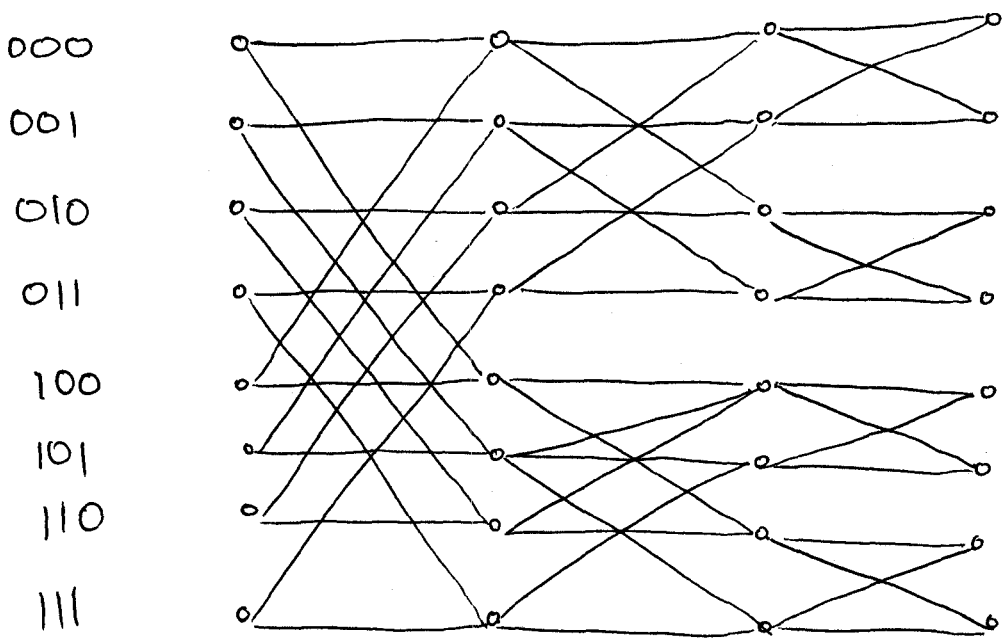
Figure 2.10 A perfect shuffle interconnection for eight inputs and outputs.

- $S_i \rightarrow S_j$ iff $j = i$
  at level l $j = i \oplus (2^{\log p - l})$

  "Butterfly network"
- Other networks
  → banyan, Beneš,

(a) (b)

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



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

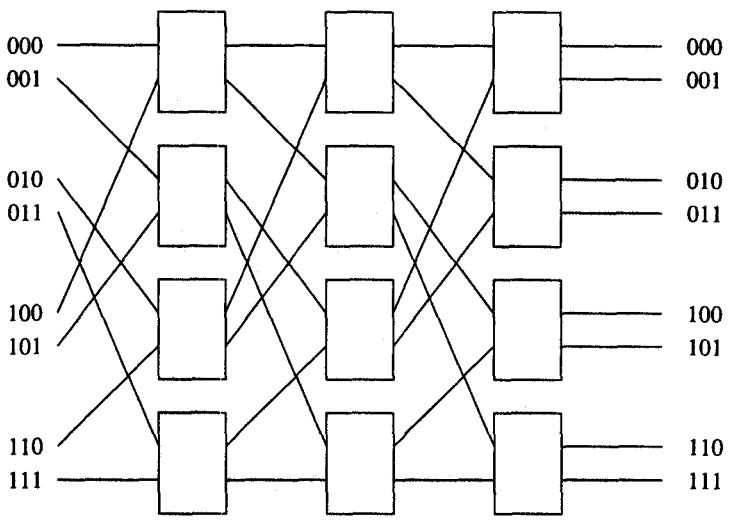


Figure 2.12 A complete omega network connecting eight inputs and eight outputs .

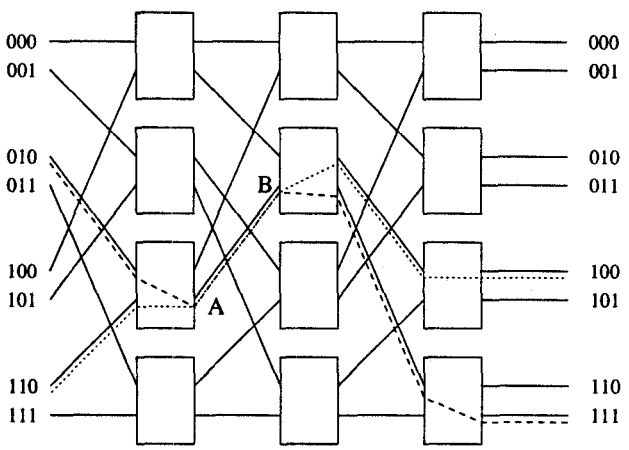


Figure 2.13 An example of blocking in omega network: one of the messages (010 to 111 or 110 to 100) is blocked at link AB.

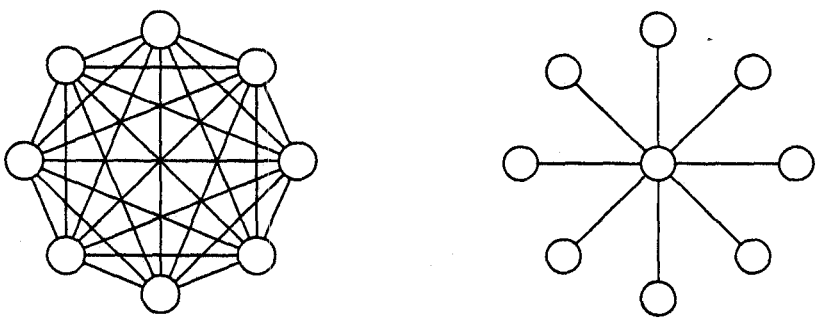


Figure 2.14 A completely-connected network of eight processors (a), and a star-connected network of nine processors (b).

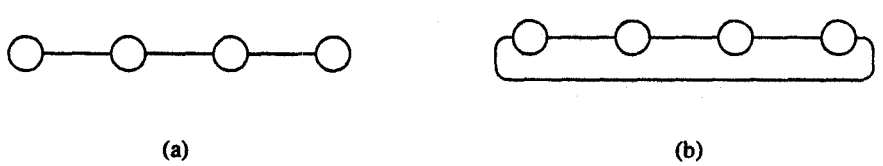


Figure 2.15 A four-processor linear array (a) and a four-processor ring (b).

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

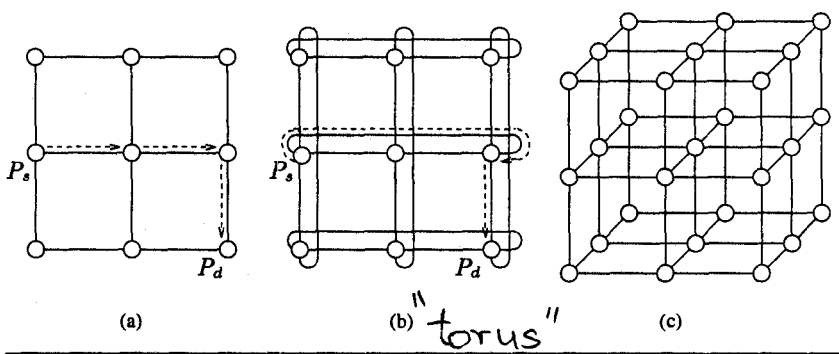


Figure 2.16 (a) A two-dimensional mesh with an illustration of routing a message from processor P_s to processor P_d; (b) a two-dimensional wraparound mesh with an illustration of routing a message from processor P_s to processor P_d; (c) a three-dimensional mesh.

communication bottleneck at higher levels.

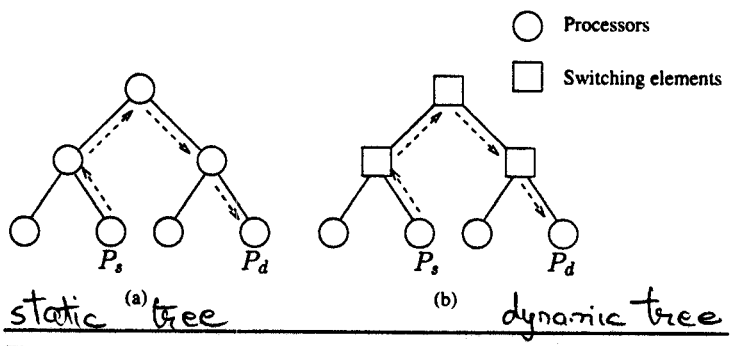


Figure 2.17 Complete binary tree networks and message routing in them.

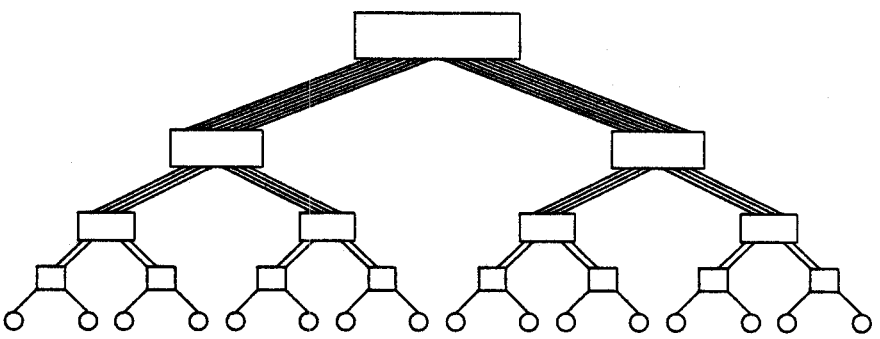


Figure 2.18 A fat tree network of 16 processors.

(1) P_i — 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

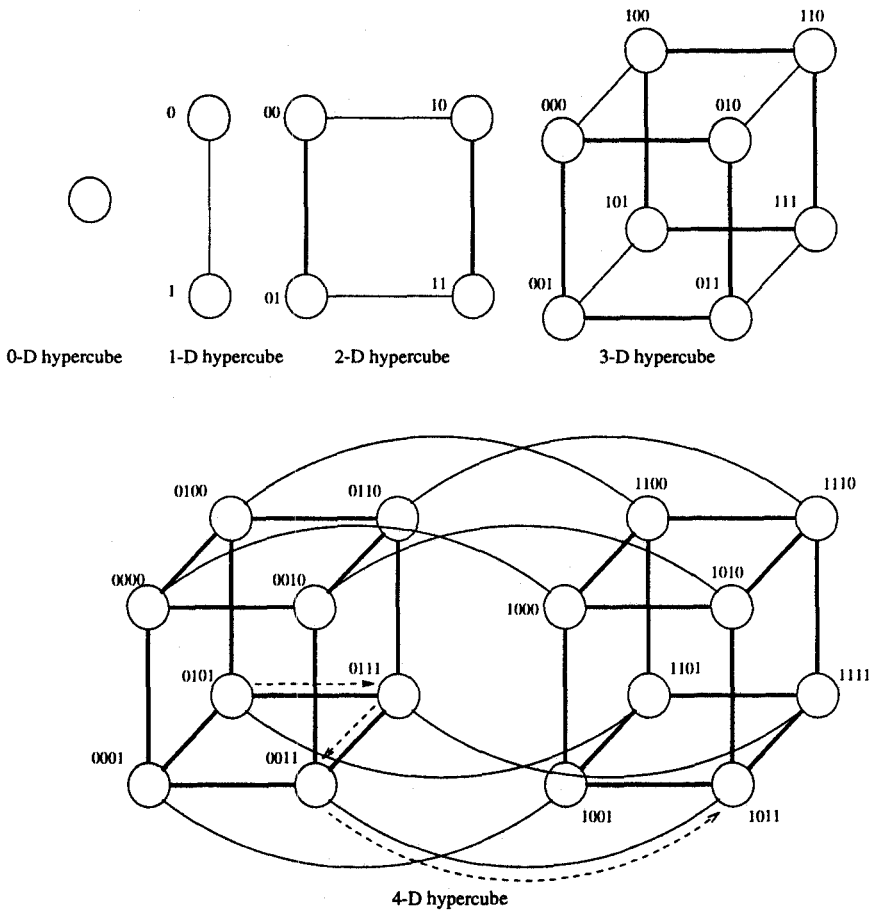


Figure 2.19 Hypercube-connected architectures of zero, one, two, three, and four dimensions. The figure also illustrates routing of a message from processor 0101 to processor 1011 in a four-dimensional hypercube.

mesh w/ 2 procs in each dimension

(5) # communic. links in shortest path ≡ Hamming distance

$s \oplus t$: route along dimensions where the corresp. bit position = 1

eg nCUBE 2, Cosmic Cube, iPSC

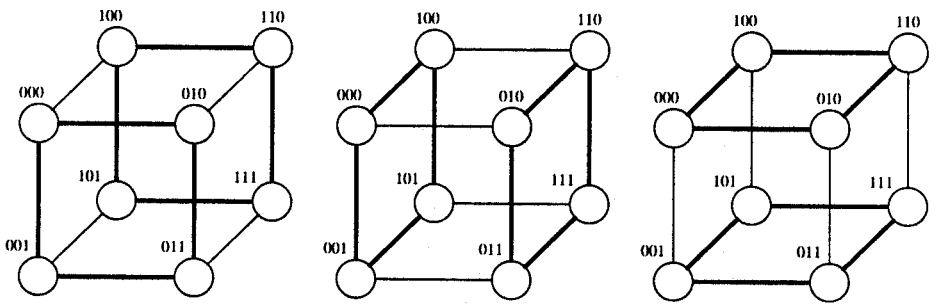


Figure 2.20 Three distinct partitions of a three-dimensional hypercube into two two-dimensional cubes. Links connecting processors within a partition are indicated by bold lines.

k-ary d-cube networks
- k → radix = # processors along each dimension
- d → 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

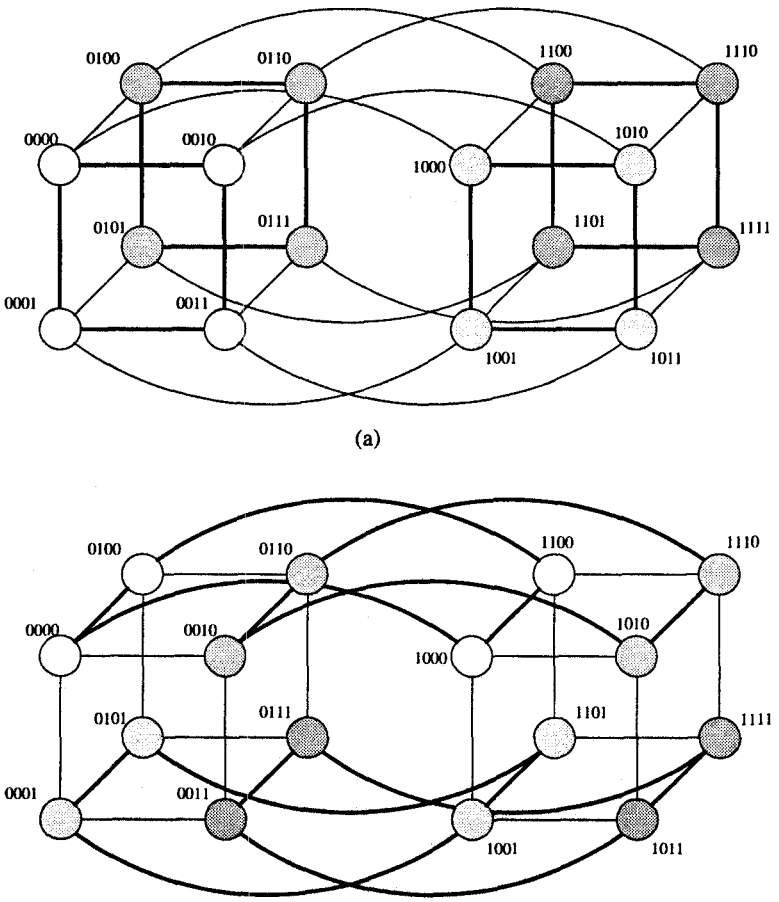


Figure 2.21 The two-dimensional subcubes of a four-dimensional hypercube formed by fixing the two most significant label bits (a) and the two least significant bits (b). Processors within a subcube are connected by bold lines.

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 BW = bisection width * channel BW

Embedding networks into a hypercube

- without embedding, algorithm designed for a specific graph may have to be adapted to another graph

$(V, E) \rightarrow (V', E')$: max #. edges mapped to any edge in $E' \equiv$ congestion

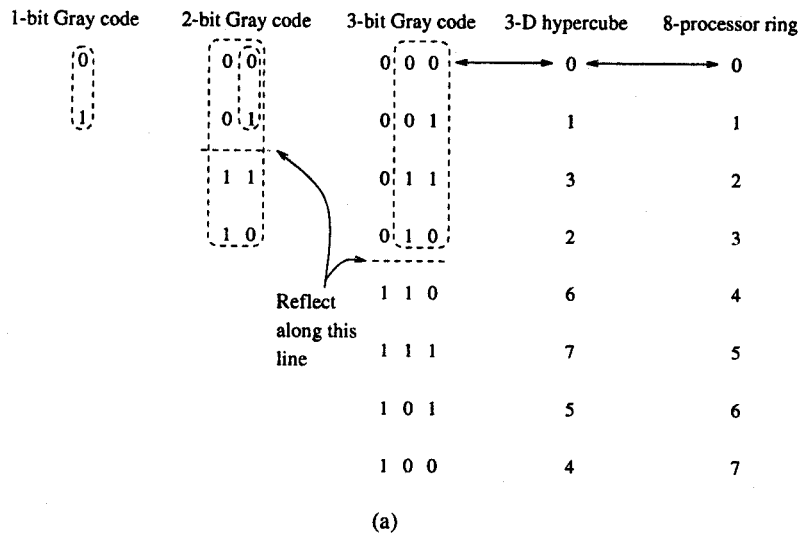
: max #. links in E' that any edge in E is mapped onto $\equiv$ dialation

: $\frac{|V'|}{|V|} \equiv$ expansion

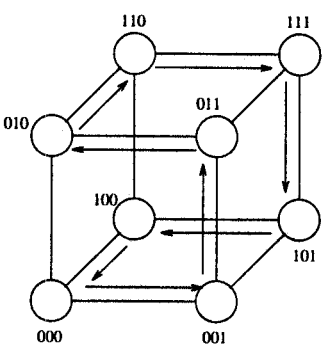
Embedding Linear Array into Hypercube

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

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



(a)



(b)

Figure 2.22 A three-bit reflected Gray code ring (a) and its embedding into a three-dimensional hypercube (b).

$G \equiv$ binary reflected Gray code

$G(i,d) \equiv i^{th}$ entry in seq. of Gray codes of d bits

- mapping dilation = 1
  expansion = 1

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

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

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

- dilation = 1, congestion = 1

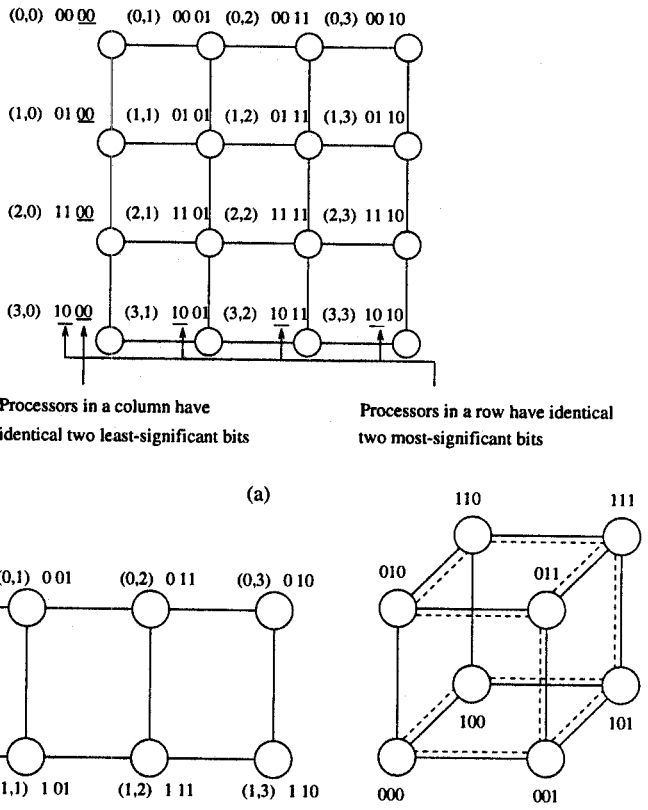


Figure 2.23 (a) A 4 × 4 mesh illustrating the mapping of mesh processors to processors in a four-dimensional hypercube; and (b) a 2 × 4 processor mesh embedded into a three-dimensional hypercube.

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

  row_{mesh} $\longrightarrow$ distinct subcube

  $column_{mesh}$ $\longrightarrow$ distinct subcube

Embedding Binary Tree into HC

- procs only at leaf nodes

(1) root → any $proc_{HC}$

(2) ∀ node m at depth j

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

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

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

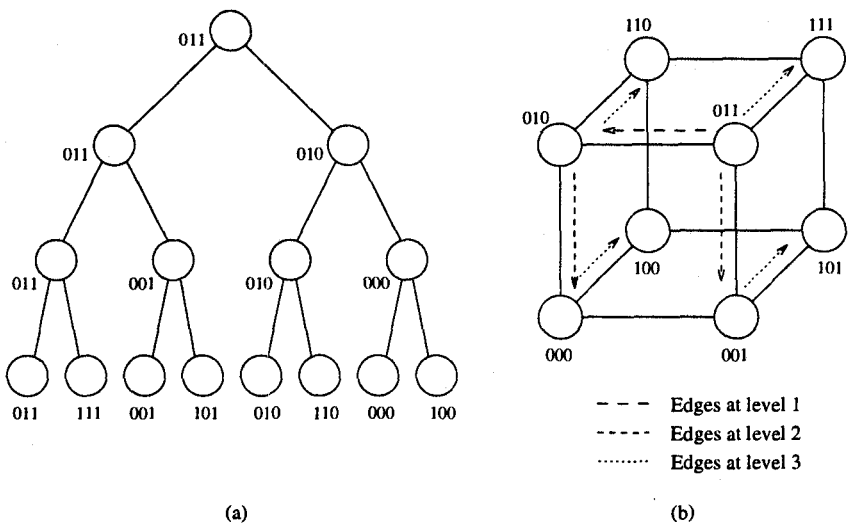


Figure 2.24 A tree rooted at processor 011 (=3) and embedded into a three-dimensional hypercube: (a) the organization of the tree rooted at processor 011, and (b) the tree embedded into a three-dimensional hypercube.

- 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:

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

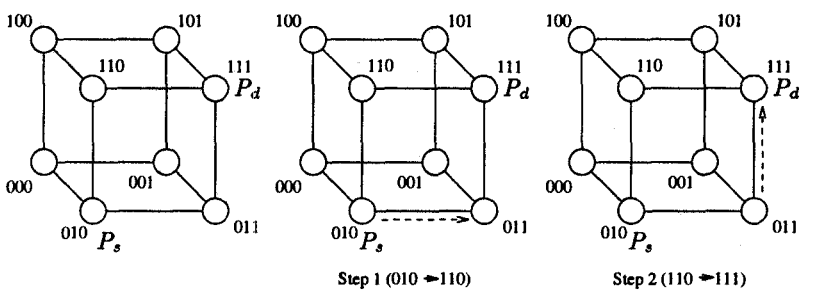


Figure 2.25 Routing a message from processor P_s (010) to processor P_d (111) in a three-dimensional hypercube using E-cube routing.

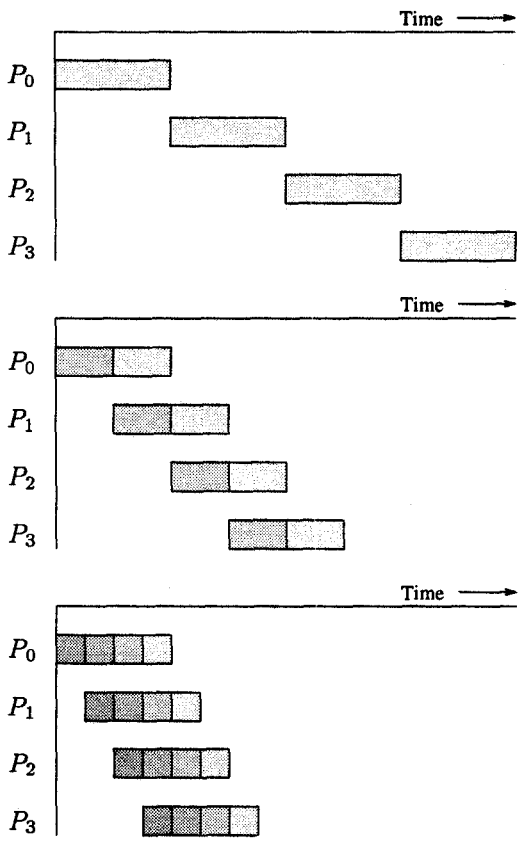
Communication Costs in Static I.N.

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

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

  eg wormhole routing

  flow-control digits (flits): are pipelined



(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
  → How avoided?
  → E-cube routing & XY routing are deadlock-free

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

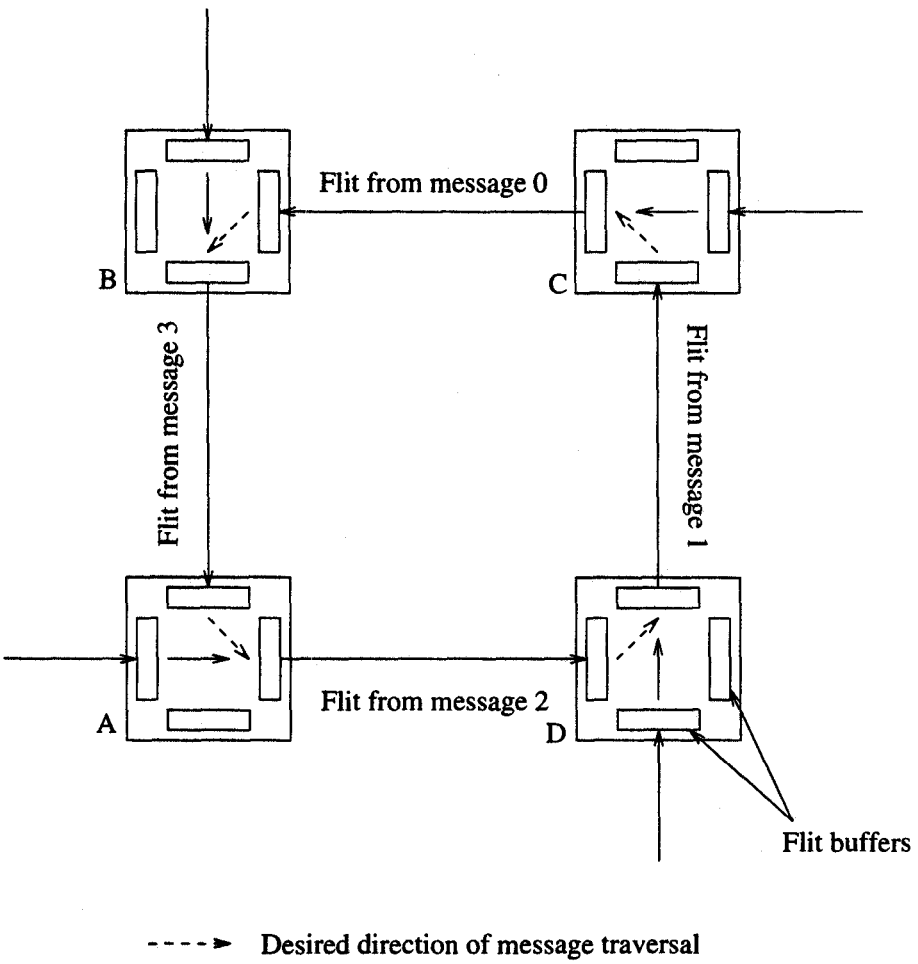


Figure 2.27 An example of deadlock in a wormhole-routing network.