• CPU speed limitations → instruction execution rate
  CPU ← memory rate
→ memory interleaving, cache
→ instruction and execution pipelining
→ superscalar execution: data/resource/branch dependencies
→ VLIW processors & IA64
  Instructions 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 (c) 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:  
SLTAC IV, MPP, DAP,  
CM-2, MasPar,  
MP-1 & MP-2

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

PE: Processing Element

![Diagram](a) and (b) figure 2.2 A typical SIMD architecture (a) and a typical MIMD architecture (b).

Copyright (c) 1994 Benjamin/Cummings Publishing Co.
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 (c) 1994 Benjamin/Cummings Publishing Co.

different processors cannot execute different instructions 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 (c) 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).
Copyright (c) 1994 Benjamin/Cummings Publishing Co.
- Crossbar: scalable to performance; not scalable to cost
- Shared bus: Not scalable; scalable to 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.
Input $i$, output $j$

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

left-rotation on binary representation of $i$

• Omega network: $\frac{p}{2} \times \log_2 p$ switching elements

---

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

Copyright (c) 1994 Benjamin/Cummings Publishing Co.
- \( s_i \rightarrow s_j \) if \( j = i \) at level \( l \)
  \[ j = i \oplus (2^{\log p - l}) \]
  "Butterfly network"

- Other networks
  \( \rightarrow \) banyan, Benes,

---

**Figure 2.11** Two switching configurations of the 2 \( \times \) 2 switch: (a) Pass-through; (b) Cross-over.

Copyright (c) 1994 Benjamin/Cummings Publishing Co.
- Routing in stage $i$ uses $i^{th}$ bit
- JL network used in BBN Butterfly, IBM RP-3, NYU Ultracomputer.

![Diagram of an omega network](image)

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

Copyright (c) 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.

Copyright (r) 1994 Benjamin/Cummings Publishing Co.
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 mesh: DAP, Paragon XP/S
- 3-D mesh: Gray T3D, J-machine

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

Copyright (c) 1994 Benjamin/Cummings Publishing Co.
communication bottleneck at higher levels.

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

Figure 2.18 A fat tree network of 16 processors.
Copyright (c) 1994 Benjamin/Cummings Publishing Co.
(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 HC's? $2^k$

**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. Copyright (c) 1994 Benjamin/Cummings Publishing Co.

(5) # communic links in shortest path = Hamming distance

sat: route along dimensions where the corr. 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.

Copyright (c) 1994 Benjamin/Cummings Publishing Co.

\[ k \text{-ary } d \text{-cube networks} \]

- radix = \# processors along each dimension
- \( d \)-dim hypercube = binary \( d \)-cube
- ring of \( p \) processors = \( p \)-ary 1-cube
- 2D wraparound mesh of \( p \) processors = \( \sqrt{p} \)-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.

Copyright (r) 1994 Benjamin/Cummings Publishing Co.
Table 2.1: A summary of the characteristics of various static network topologies connecting \( p \) processors.

<table>
<thead>
<tr>
<th>Network</th>
<th>Diameter</th>
<th>Bisection Width</th>
<th>Arc Connectivity</th>
<th>Cost (No. of links)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completely-connected</td>
<td>1</td>
<td>( p^2/4 )</td>
<td>( p - 1 )</td>
<td>( p(p - 1)/2 )</td>
</tr>
<tr>
<td>Star</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>( p - 1 )</td>
</tr>
<tr>
<td>Complete binary tree</td>
<td>( 2 \log((p + 1)/2) )</td>
<td>1</td>
<td>1</td>
<td>( p - 1 )</td>
</tr>
<tr>
<td>Linear array</td>
<td>( p - 1 )</td>
<td>1</td>
<td>1</td>
<td>( p - 1 )</td>
</tr>
<tr>
<td>Ring</td>
<td>( [p/2] )</td>
<td>2</td>
<td>2</td>
<td>( p )</td>
</tr>
<tr>
<td>2-D mesh without wraparound</td>
<td>( 2(\sqrt{p} - 1) )</td>
<td>( \sqrt{p} )</td>
<td>2</td>
<td>( 2(p - \sqrt{p}) )</td>
</tr>
<tr>
<td>2-D wraparound mesh</td>
<td>( 2[\sqrt{p}/2] )</td>
<td>( 2\sqrt{p} )</td>
<td>4</td>
<td>( 2p )</td>
</tr>
<tr>
<td>Hypercube</td>
<td>( \log p )</td>
<td>( p/2 )</td>
<td>( \log p )</td>
<td>( (p \log p)/2 )</td>
</tr>
<tr>
<td>Wraparound k-ary d-cube</td>
<td>( d[k/2] )</td>
<td>( 2d )</td>
<td>( d )</td>
<td>( dp )</td>
</tr>
</tbody>
</table>

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

\[
\text{channel BW} = \frac{\text{bisection width} \times \text{channel BW}}{}
\]

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 \( E' \) = congestion

\[
\text{max. links in } E' \text{ that any edge in } E \text{ is mapped onto}
\]

\[
\# \frac{|V'|}{|V|} = \text{expansion}
\]
Embedding Linear Array into Hypercube

Processor \( i \) of linear array \( \rightarrow \) 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) 1-bit Gray code 2-bit Gray code 3-bit Gray code 3-D hypercube 8-processor ring

Reflect along this line

(b)

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

Copyright (c) 1994 Benjamin/Cummings Publishing Co.

\[ G = \text{binary reflected Gray code} \]

\[ G(i,d) = i^{th} \text{ entry in seq. of Gray codes of } d \text{ bits} \]

* mapping dilation = 1

* expansion = 1
Embedding a Mesh into a Hypercube

- \( 2^r \times 2^s \) wraparound mesh \( \rightarrow 2^{r+s} \) HC
  
  \[ (i,j)_{\text{mesh}} \rightarrow G(i,r) \parallel G(j,s)_{\text{HC}} \]
  
  \( \text{dilation} = 1, \text{ congestion} = 1 \)

- Processors in a column have identical two least-significant bits
- Processors in a row have identical two most-significant bits

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

Copyright (c) 1994 Benjamin/Cummings Publishing Co.

- Procs in same row \( \text{mesh} \) \( \rightarrow \) procs with labels having \( r \) identical MSBs

- Row \( \text{mesh} \) \( \rightarrow \) distinct subcube

- Column \( \text{mesh} \) \( \rightarrow \) distinct subcube
Embedding Binary Tree into HC

1. only at leaf nodes

\[ \text{root} \rightarrow \text{any } \text{proc} \]

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

\[ C_{\text{LEFT}}(m) \rightarrow \text{same proc}_{HC} \text{ to which } m \text{ is mapped} \]

\[ C_{\text{RIGHT}}(m) \rightarrow \text{proc}_{HC} \text{ we invert bit } j \text{ of } i \]

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

---

(a)

(b)

---

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 (c) 1994 Benjamin/Cummings Publishing Co.

In above mapping, expansion = 1
**Routing Mechanisms**

- Minimal vs. non-minimal
- Deterministic vs. adaptive
- Dimension-ordered routing vs. XY routing, E-cube routing

**Ps → Pd**

At any intermediate proc. \( P_i \):

- fwd msg. along dimension corresponding to least significant non-zero 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 (c) 1994 Benjamin/Cummings Publishing Co.

---

**Communication Costs in Static IN:**

1. \( t_s \) (startup time): once per msg.
2. \( t_h \) (per hop time) = node latency \( \times \) [latency in switch to determine which buffer/channel to use]
3. \( t_w \) (per word Xfer time) = \( \frac{1}{r} \), where \( r = \text{channel BW} \)
- Store-and-forward: \( t_{\text{comm}} = t_s + (mt_w + t_h)l = \Theta(ml) \)

- Cut-through routing: msg advanced from incoming \textbf{as it arrives} outgoing link

  eg wormhole routing

  flow-control digits (flits): are pipelined

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{diagram.png}
\caption{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. Copyright (c) 1994 Benjamin/Cummings Publishing Co.}
\end{figure}

- Susceptibility to deadlock
  
  \( \rightarrow \) How avoided?
  
  \( \rightarrow \) E-cube routing & XY routing are deadlock-free

- Cut-through: \( t_{\text{comm}} = t_s + lt_h + mt_w = \Theta(m+l) \)
Figure 2.27  An example of deadlock in a wormhole-routing network.
Copyright (c) 1994 Benjamin/Cummings Publishing Co.