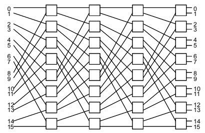
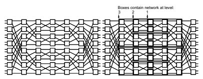
Multistage-Network Overview

Major Types

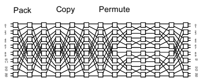
- *Banyan* and Banyan-Like.
- Clos and Beneš
- Batcher Sorters
- Miscellaneous



Banyan (Omega)

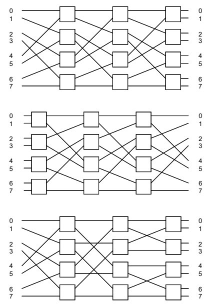


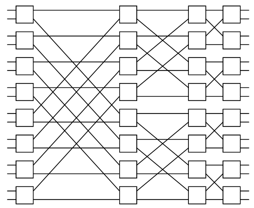
Benes

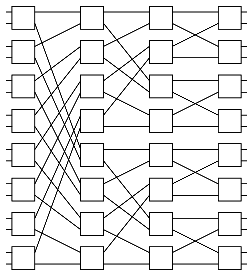


Ofman Gen. Con.

- *Banyan* and Banyan-Like.







Includes omega network.

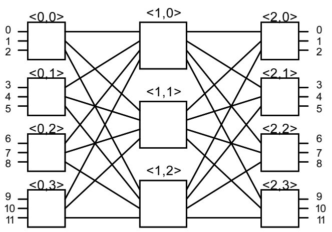
Provide efficient interconnections.

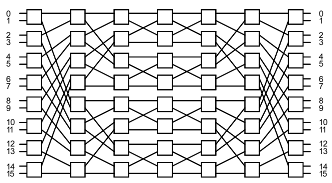
Easy to route.

Cannot support all connection assignments.

Most commonly used MIN type.

- Clos and Beneš





Cost roughly twice as much as Banyans.

Difficult to route.

Can support all permutation connection assignments.

Used in a small number of research computers.

- Batcher Sorters

Cost roughly $\log N$ times more than N-input Banyan.

Easy to route.

Can support all permutation connection assignments.

- Miscellaneous

Fanout/Concentrate Family

Composite Networks

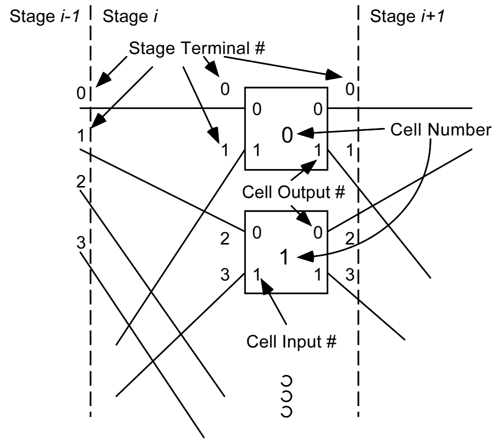
Of non-practical-theoretical and practical interest.

Multistage-Network Terminology

A MIN consists of cells and links.

Cells arranged in stages.

Links connect cells in adjacent stages.



Multistage-Network Terminology

The following numbering will be used:

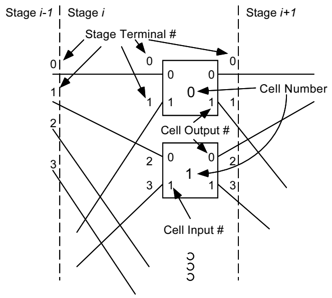
Stages, consecutively from 0 (input) to $s-1$ or $n-1$.

Cells within a stage, consecutively from 0.

Cell inputs and outputs, consecutively from 0 for each cell.

Cell inputs and outputs also given *stage terminal* numbers.

Terminal numbers consecutively from 0, for all cells.



Link Patterns

A *link pattern* is an arrangement of links that could connect adjacent stages in a MIN.

More Formally:

Let T_1 and T_2 be equal-cardinality sets of terminal numbers.

A *link pattern* is a bijection from T_1 to T_2.

Example:

Let $T_1 = \{0, 1, 2\}$ and $T_2 = \{0, 1, 2\}$.

Let $\pi(0) = 1$, $\pi(1) = 0$, and $\pi(2) = 2$.

Mapping π is a bijection from T_1 to T_2, and therefore a link pattern.

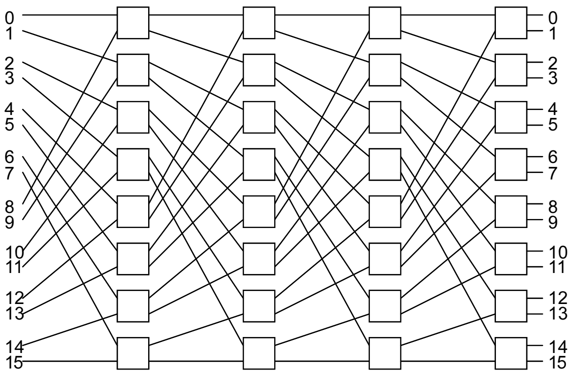
Let $\pi'(0) = 1$, $\pi'(1) = 1$, and $\pi'(2) = 2$.

Mapping π' is not a bijection, and so not a link pattern.

Omega Network Routing

To route request (u, v) in an n-stage 2×2 network ...

... at stage i take cell output $v_{(n-1-i)}$, for $i \in \langle n \rangle$.



Generalization of MIN Using Shuffles

Goal: Specify a family of networks which

- consists of stages of *square* cells
- with shuffles between stages chosen so that:

  outputs from an individual cell in one stage connect to consecutive cells in the next stage.

Each stage can have its own cell size.

These networks called a *flat square MIN* in class.

Generalization of MIN Using Shuffles

Notation: $(s, \vec{m}, \vec{k}, l, r)$, where

s is the number of stages;

$\vec{m} = (m_0, m_1, \ldots, m_s)$, m_i indicates the cell size in stage $i \in \langle s \rangle$;

$\vec{k} = \left(\frac{N}{m_0}, \frac{N}{m_1}, \ldots, \frac{N}{m_s} \right)$,

where $N = \prod_{i=0}^{s-1} m_i$, the number of inputs;

$\frac{N}{m_i}$ indicates the number of cells in stage i;

and the link patterns are as follows (on next slide):

Generalization of MIN Using Shuffles

- Stage 0:

  If $l = \mathrm{T}$ then $\langle \mathrm{I},\, u \rangle$ connects to $\langle 0,\, u \rangle$ for $u \in N$.

  That is, there is no shuffle in the first stage.

  If $l = \mathrm{S}$ then $\langle \mathrm{I},\, u \rangle$ connects to $\langle 0,\, \sigma_{m_0,k_0}(u) \rangle$ for $u \in N$.

  That is, there is an m_0, k_0 shuffle in the first stage.

Generalization of MIN Using Shuffles

- Stage $i \in \{1, 2, \ldots, s-1\}$:

  Stage terminal $\langle i-1,\, u \rangle$ connects to $\langle i,\, \sigma_{m_i,k_i}(u) \rangle$.

  That is, there is an m_i, k_i shuffle in all but the first stage.

- Following last stage:

  If $r = \mathrm{T}$ stage terminal $\langle s-1,\, u \rangle$ connects to output $\langle \mathrm{O},\, u \rangle$.

  That is, there is no shuffle following the last stage.

  If $r = \mathrm{S}$ stage terminal $\langle s-1,\, u \rangle$ connects to output $\langle \mathrm{O},\, \sigma_{m_s,k_s} u \rangle$.

  That is, there is a shuffle following the last stage.

  (The only purpose of m_s and k_s is to specify this shuffle.)

To route request (u, v):

- Convert u and v to radix $\vec{r}$ numbers, where $r_i = m_{n-1-i}$.
- At stage i take cell output v_{n-1-i}.

To convert the terminal numbers in each stage to decimal:

In stage j use radix $\vec{r}_j$ where

$$r_{j,i} = m_{n+j-i \bmod n}.$$

Then $x = x_0 + \sum_{i=1}^{n-1} x_i \prod_{l=0}^{i-1} r_{j,l}$.

Modified Omega Network

Consists of stages of cells connected by shuffles.

It's not an omega network (though it does look like one).

Described by (m, k), where $m > 1$ and $k \geq 1$.

Such a network has:

- $N = mk$ inputs and outputs and
- $s = \lceil \log_m N \rceil$ stages.

Each stage consists of

- an $\sigma_{m,k}$ shuffle followed by
- k cells of size $m \times m$.

Important Difference

Uses integer shuffle function, not symbol shuffle function.

Significance: structure isn't related to radix-m link numbers.[1]

Routing more complex.

[1] Nor any mixed-radix form, whether or not the links are numbered differently.

Generalized Omega Network

A *generalized omega network* is a type of flat square min:

$$\left(s,\ (m_0, m_1, \ldots, m_s)\ ,\ \left(\frac{N}{m_0}, \frac{N}{m_1}, \cdots, \frac{N}{m_s}\right)\ , \mathrm{S}, \mathrm{T}\right), \text{ where}$$

$N = \prod_{i=0}^{s-1} m_i$, the number of inputs.

Path-length, $d = s$.

Generalized Omega Network

Mixed Radix Numbers (used in routing)

Let $\vec{r} = (r_{n-1}, r_{n-2}, \ldots, r_0)$ be a sequence of integers > 1.

To convert an integer x to radix $\vec{r}$:

LSD is $x_0 = x \bmod r_0$.

Second digit is $x_1 = \left\lfloor \frac{x}{r_0} \right\rfloor \bmod r_1$.

Digit $x_i = \left\lfloor \frac{x}{\prod_{j=0}^{i-1} r_j} \right\rfloor \bmod r_i$.

Examples:

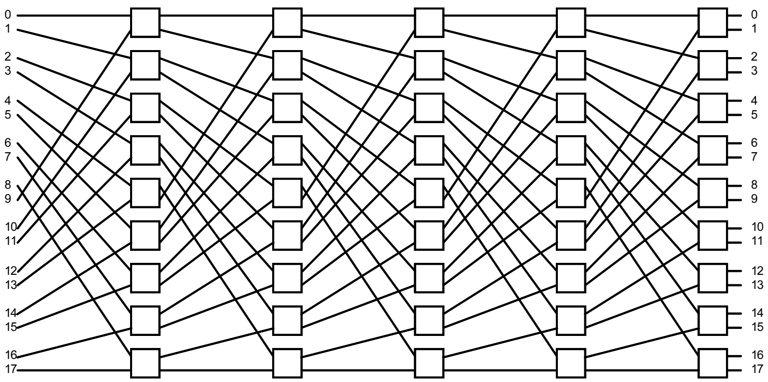
$\vec{r} = (2, 3, 4)$: $3 = (0, 0, 3)$, $4 = (0, 1, 0)$, $20 = (1, 2, 0)$.

$\vec{r} = (5, 4, 2)$: $3 = (0, 1, 1)$, $4 = (0, 2, 0)$, $20 = (2, 2, 0)$.

To convert $x_{(n-1)} x_{(n-2)} \cdots x_{(0)}$ in radix $\vec{r}$ to decimal:

$$x = x_0 + \sum_{i=1}^{n-1} x_i \prod_{l=0}^{i-1} r_l.$$

Modified Omega Network Example



Route for request $(1,3)$:

$t_1 = 3 + 1 \times 2 \times 2 = 7 = 00111$, t_1 can always be used.

$t_2 = 7 + 1 \times 18 = 25 = 11001 \quad < 32$, can be used.

$t_3 = 7 + 2 \times 18 = 43 \quad \not< 32$, can not be used.

Routing Modified Omega Networks

Unlike real omega, some requests have more than one path.

Will compute routing tags, denoted t_p, where p is a tag number.

Routing tag used as a destination in omega.
(Or, in an omega network, the destination *is* the routing tag.)

To route request (i,j) in an (m,k) modified omega network:

Let: $N = mk$, $s = \lceil \log_m N \rceil$, and $M = N - m^{s-1}$.

$t_1 = j + mMi$

$t_p = t_1 + (p-1)M$, valid if $t_p < m^s$.

Modified Omega Network Example

Network defined by $m = 2$, $k = 9$.

Then $N = mk = 18$ and $s = \lceil \log_2 18 \rceil = 5$.

Constant used in routing, $M = N - m^{s-1} = 18 - 16 = 2$.