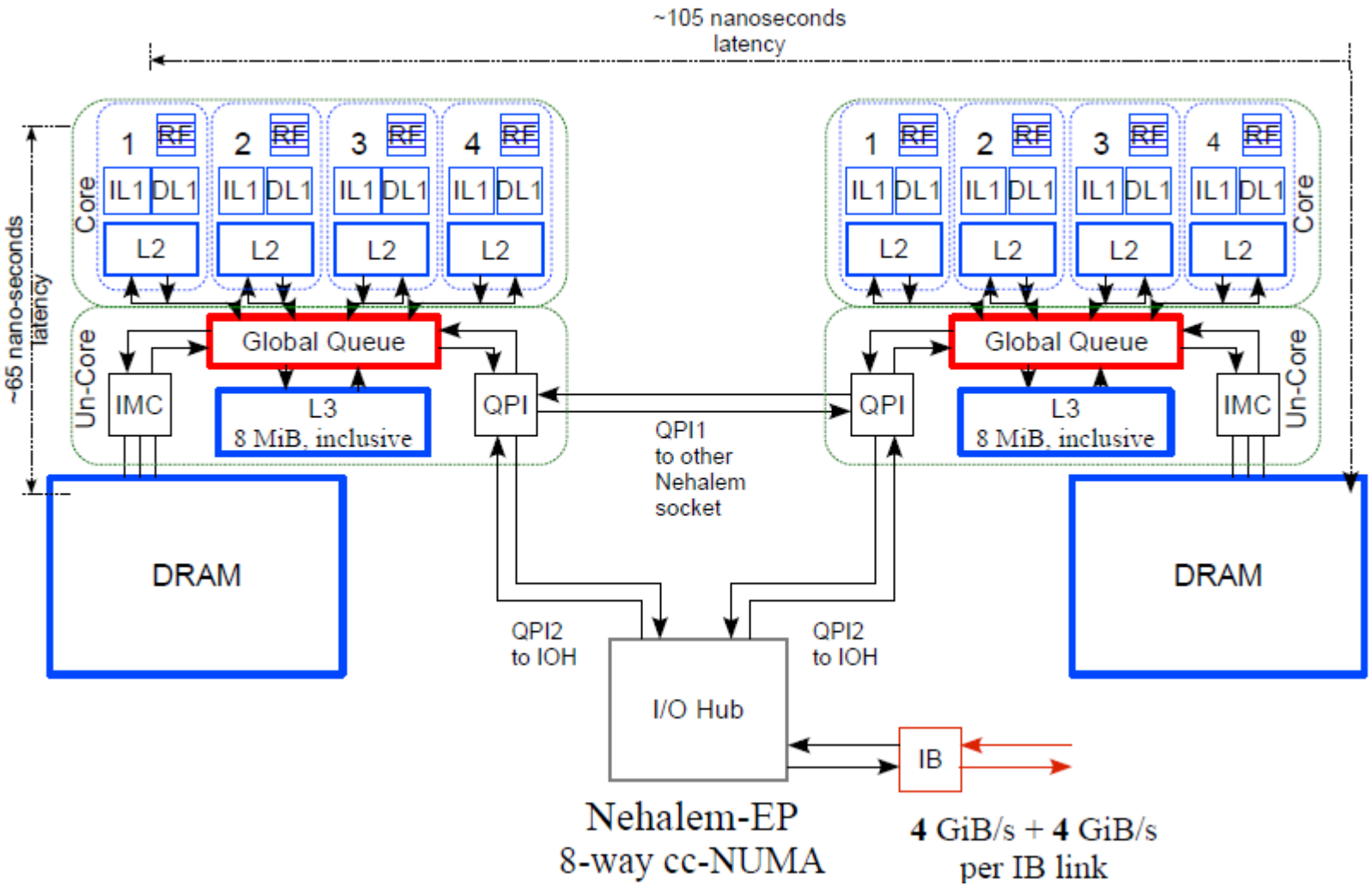
Advanced Computer Architectures

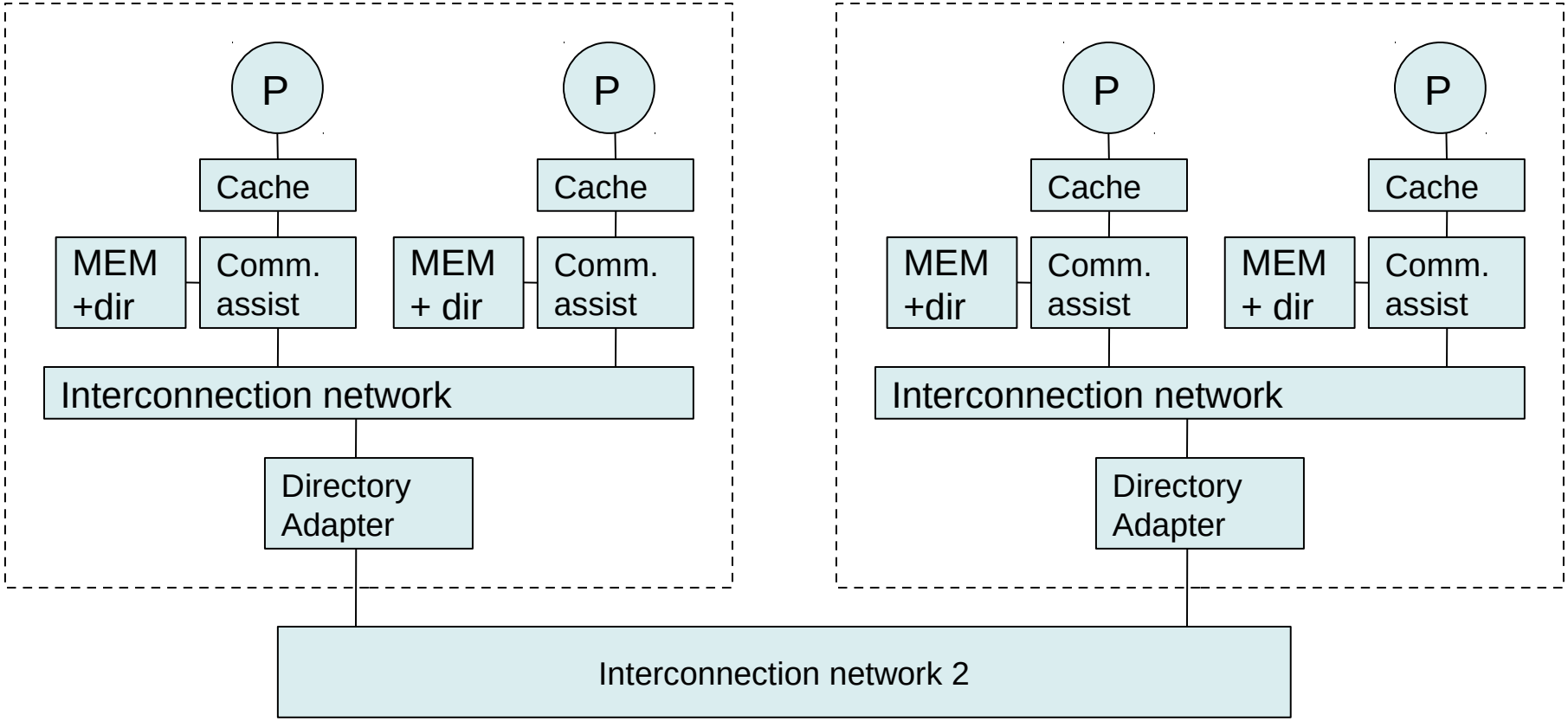
Interconnection networks

Czech Technical University in Prague, Faculty of Electrical Engineering
Slides authors: Michal Štepanovský, update Pavel Píša

Motivation: Interconnection of two Intel processors by QPI



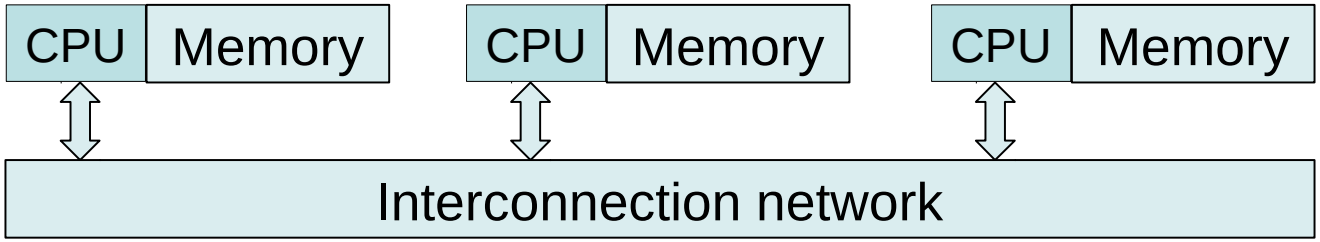
Two-level cache coherent system



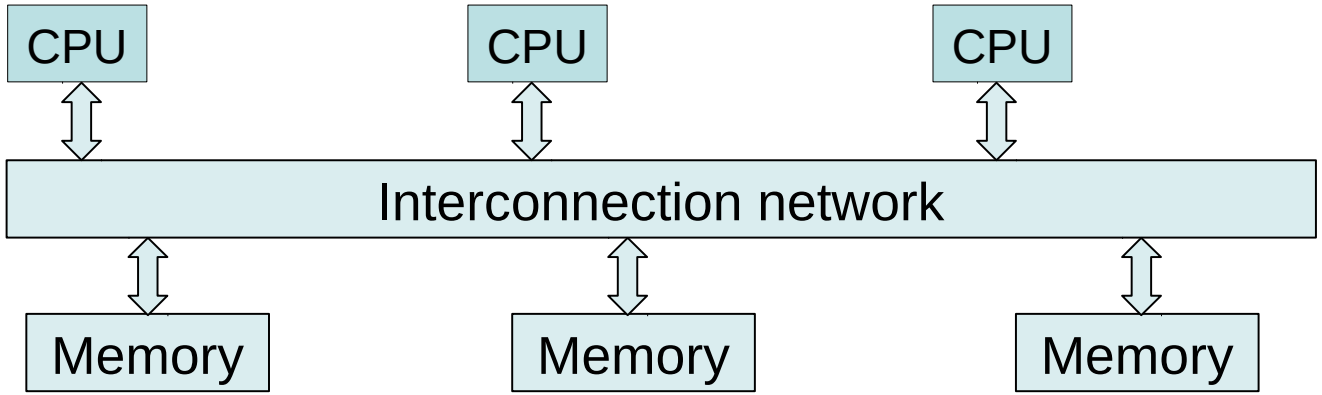
- Directory-directory
- Alternatives: Snooping-Snooping, Snooping-Directory, Directory-Snooping

Parallel computers' memory architectures

- **Loosely coupled** – memory is connected to distributed nodes and each node can access memory of any other node
  Remark: If cross memory access is not possible, iy is not shared memory system.



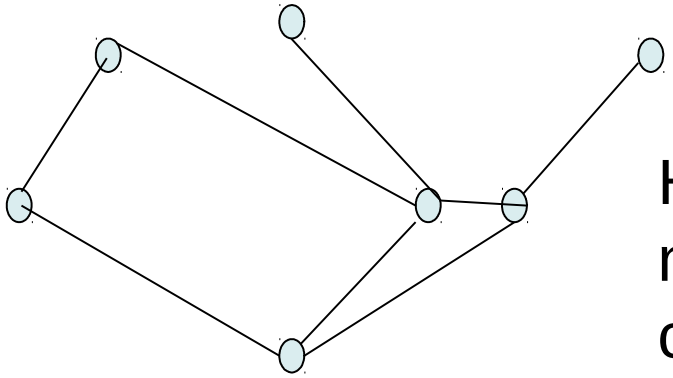
- **Tightly coupled** – memory is located centrally. Processor can be equipped by local memory/cache.



SW view of interconnection network

Network must/should support:

- One-to-one communication
- One-to-all broadcast and All-to-one reduction
- All-to-all broadcast and All-to-all reduction
- Scatter a Gather



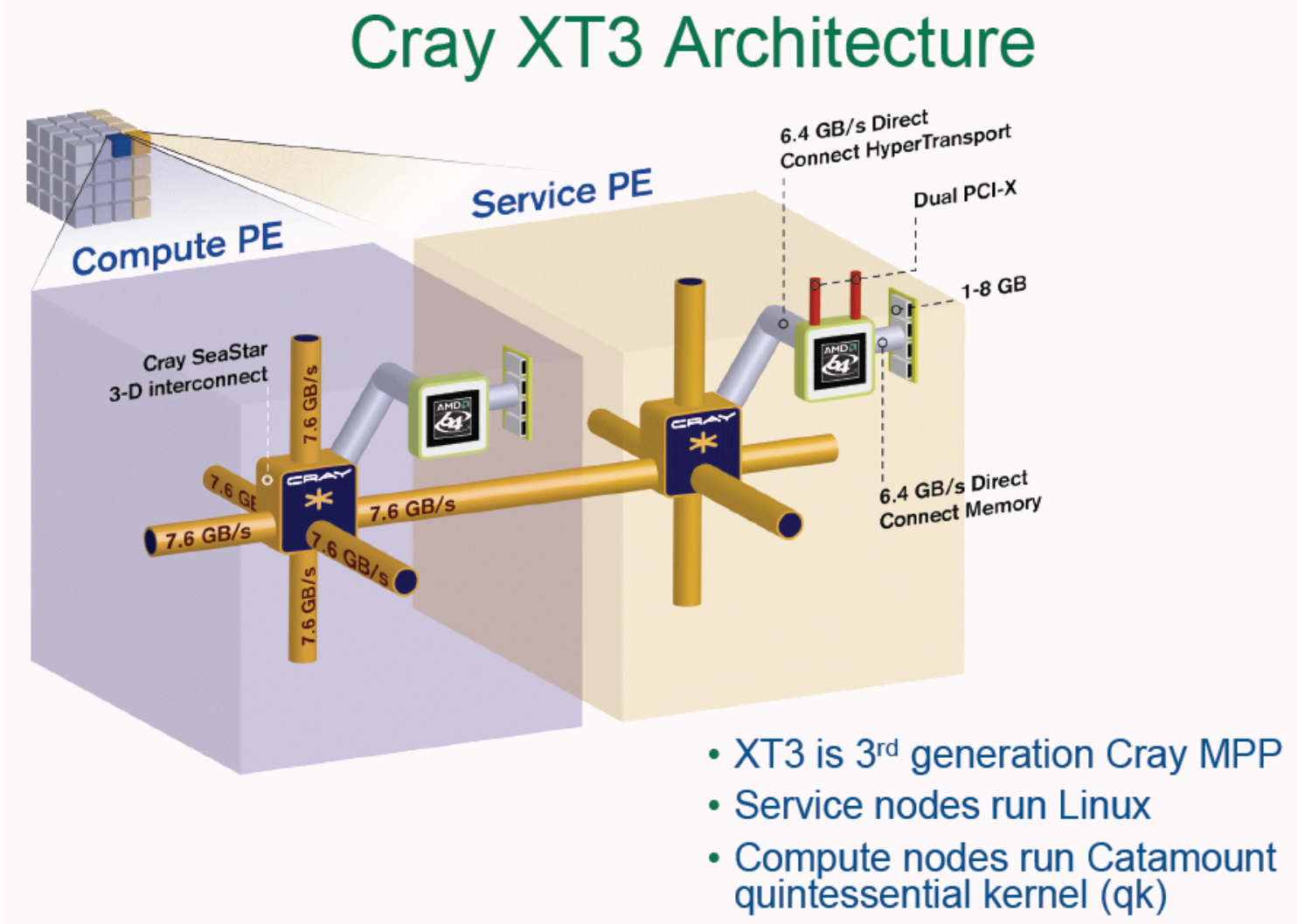
How does an optimal network realizing this communication look like?

Motivation – Cray XT3

Cray XT3 (20.5 TFlops) v Oak Ridge National Laboratory:

- 56 cabinets, 5212 computation processor elements (PEs), 82 service processor elements
- Computation element – build from 4 CPU (2GB/PE)
- CPU: 64-bit 2.4GHz AMD Opteron

Motivation – Cray XT3



Topologies

Interconnection networks topology

Static networks

- linear array
- ring
- chordal ring
- binary tree
- fat tree
- 2D, 3D mesh
- 2D, 3D torus
- hypercube
- Cube Connected Cycles (CCC)

Dynamic networks

- Bus network
- Single stage interconnect network (crossbar switches, …)
- Multistage interconnection networks (omega, Banyan, Cantor, Clos, …)

Interconnection network topology

Direct networks

Each node includes switch and vice versa

Indirect networks

Some switches are not attached to nodes but only route traffic to another switches

Static networks

Interconnection networks topology

Symmetric networks **Asymmetric networks**

Parameters

- **Network size N: number of nodes in a network,**
- **Node degree d: the number of edges entering or leaving the node,**
- **Bisection width B: minimal number of edges which have to be cut when the network is divided into two halves of the same size.**
- **Network diameter D: number of edges of shortest distance between the two most distant nodes in the network – corresponds to longest communication in a network**
- **Cost C: count of communication links (edges) in the network**

Linear array

Communication costs in parallel systems

Message-Passing Systems

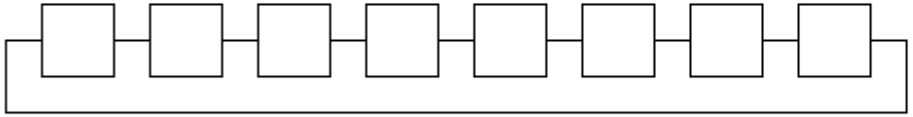
The communication cost of a data-transfer

- operation depends on:
  - start-up time: t_s
    - add headers/trailer, error-correction, execute the routing algorithm, establish the connection between source & destination
  - per-hop time: t_h
    - time to travel between two directly connected nodes.
      - node latency
  - per-word transfer time: t_w
    - 1/channel-width

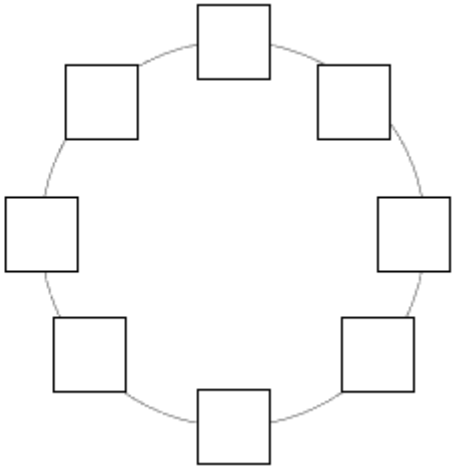
Static networks

Linear array

d: 1 (for terminal nodes), 2 (for others)
D: N-1
B: 1
C: N-1



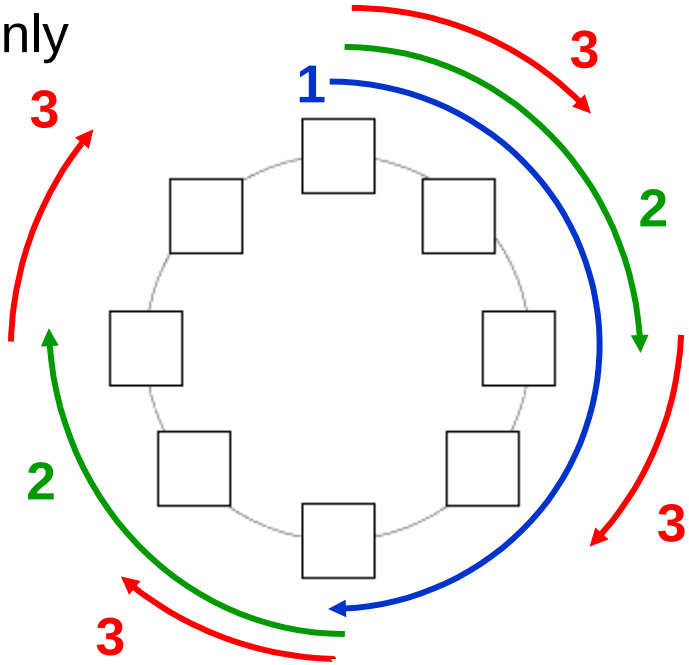
Ring



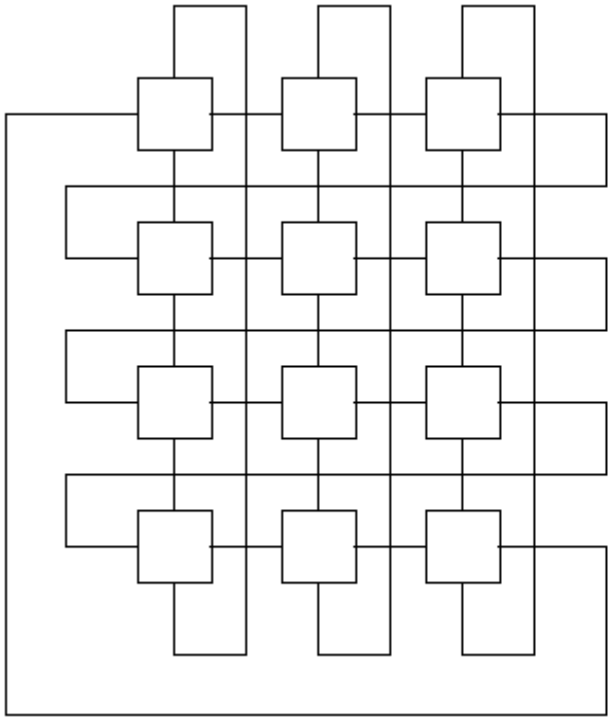
d: 2
D: N/2
B: 2
C: N

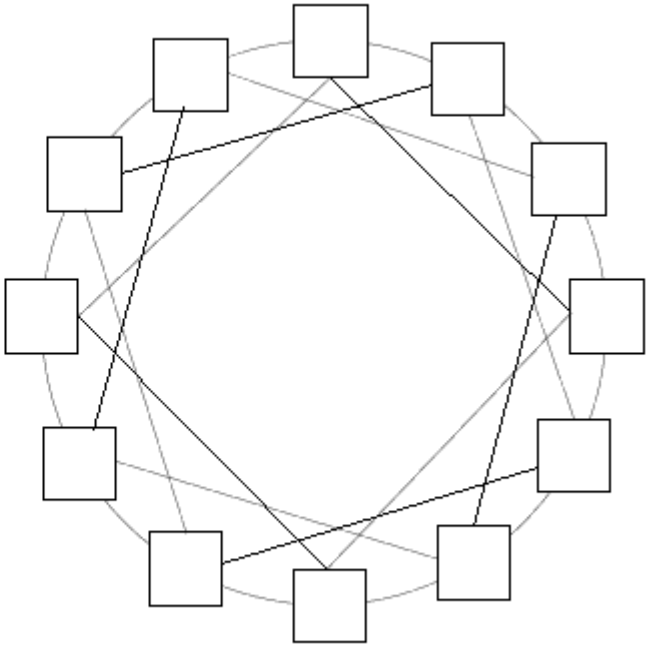
One-to-all broadcast and All-to-one reduce

- Sending N-1 messages from source to other N-1 nodes is the simplest solution. But it is not efficient…
- More efficient with less ring edge load is **recursive doubling** algorithm – source sends the message to a selected node in the middle, then both send two messages to quarters etc.
- Another option is to send the message only to the neighboring node. Even less edges in communication, but usually slower if resend requires some processing time on the node.
- Reduce – Inverted recursive doubling. But some processing required before each resend (for example addition of two results)



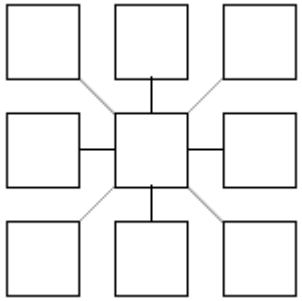
Static networks





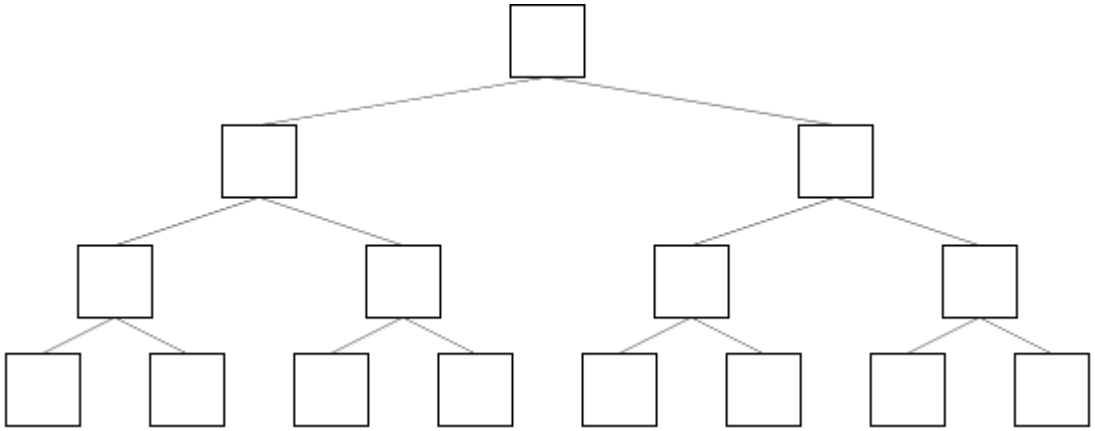
Chordal ring

Static networks



Star

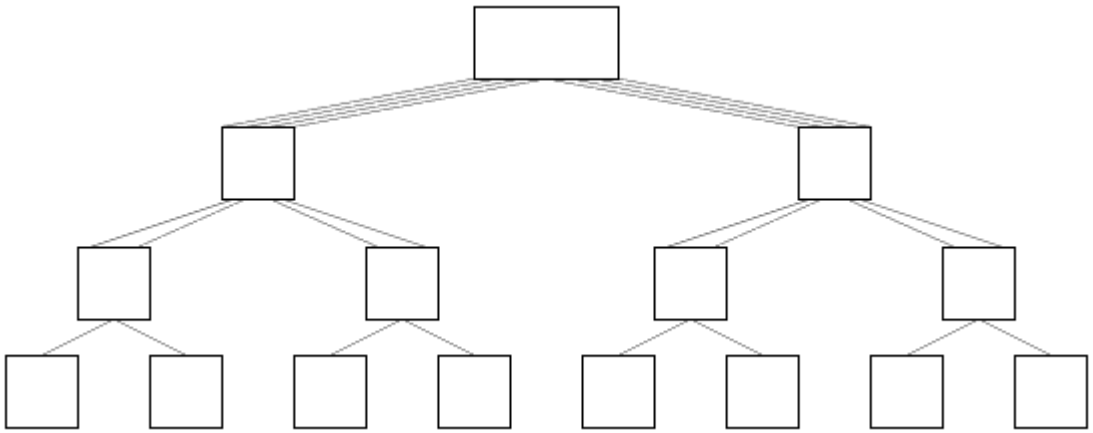
d: 1 , N-1
D: 2
B: 1
C: N-1



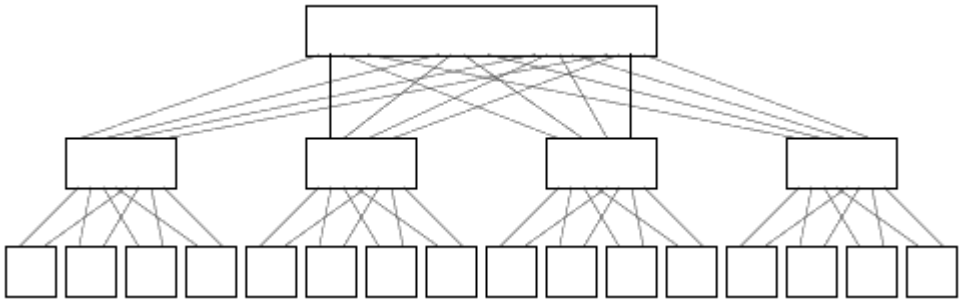
Binary tree

d: 1 (leaf nodes), 2 (root), 3 (others)
D: 2log((N+1)/2)
B: 1
C: N-1

Static networks



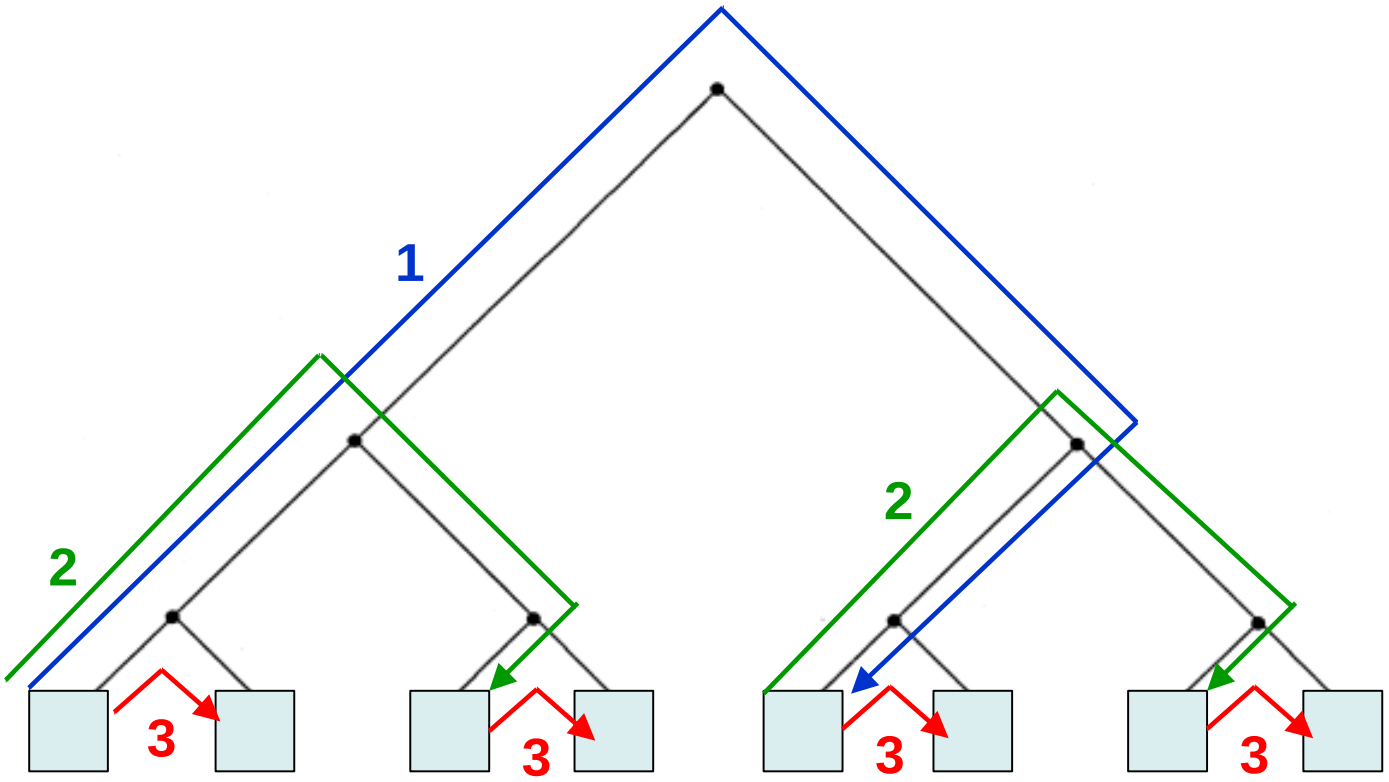
Binary fat tree



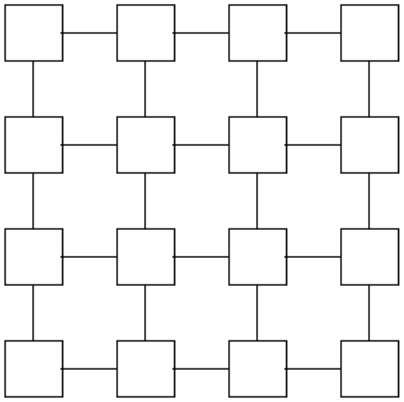
4-ary fat tree

Balanced Indirect Binary Tree – communication example

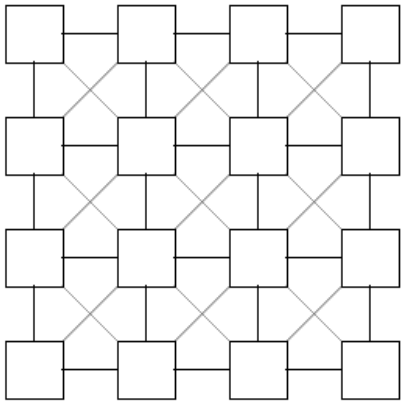
- Processing nodes are at the leaves of the indirect binary tree, and internal nodes are used only for routing.
- One-to-all broadcast and All-to-one reduce



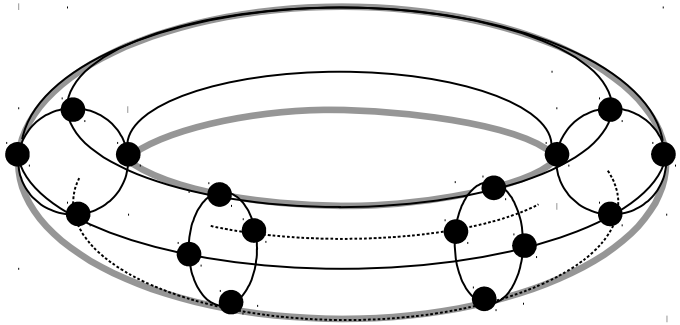
Static networks

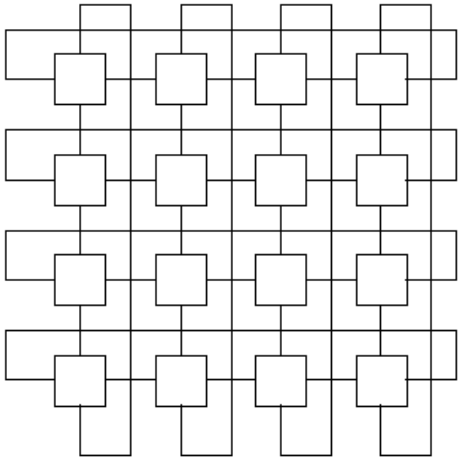


4-connected 2D mesh



8-connected 2D mesh



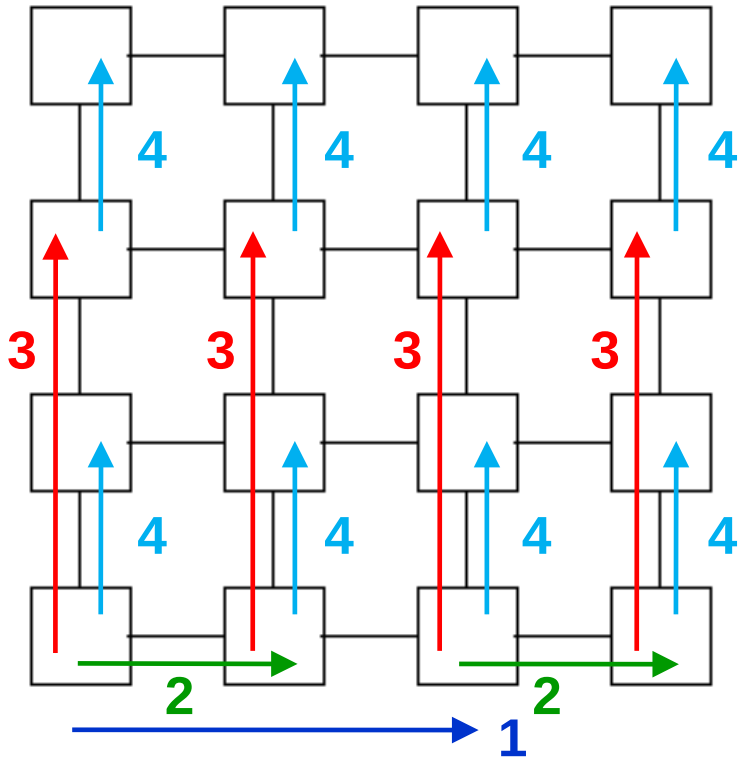


d: 4
D: $N^{1/2}$
B: $2N^{1/2}$
C: $2N$

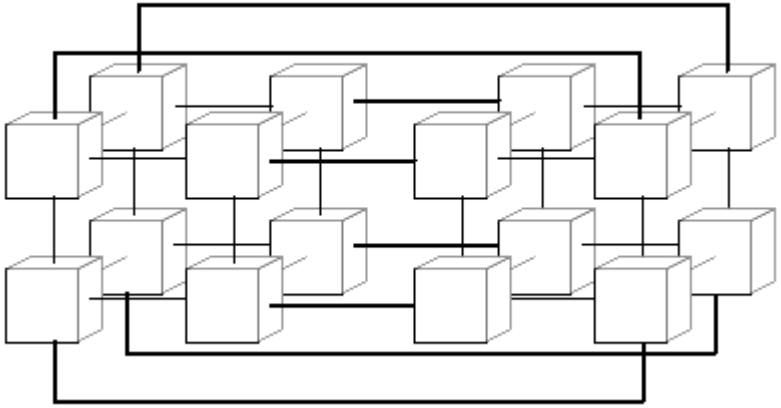
2D Torus (anuloid)

Mesh – communication example

- One-to-all broadcast and All-to-one reduce

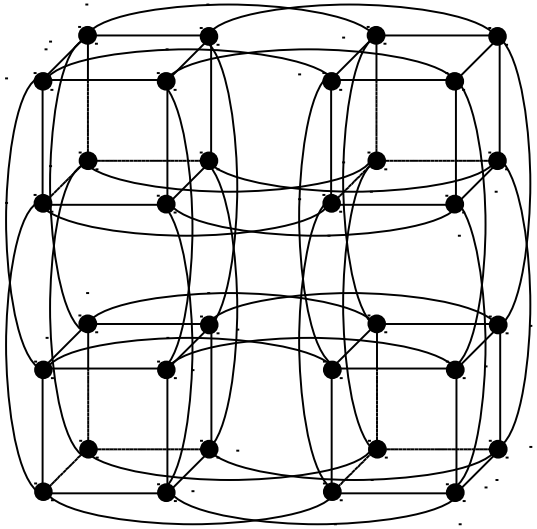


Static networks

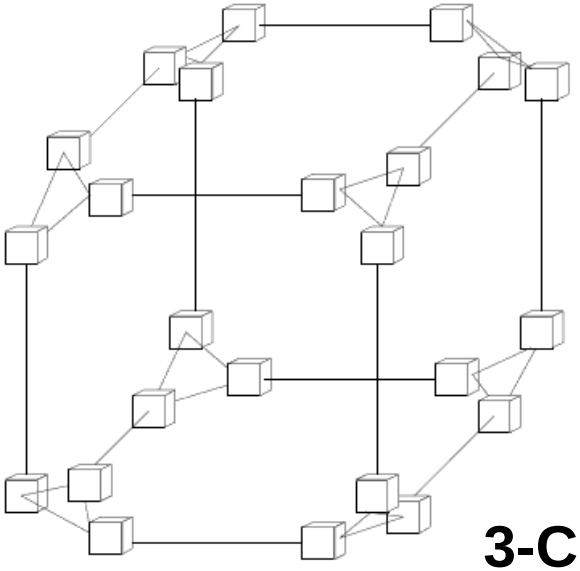


N=4

Hypercube



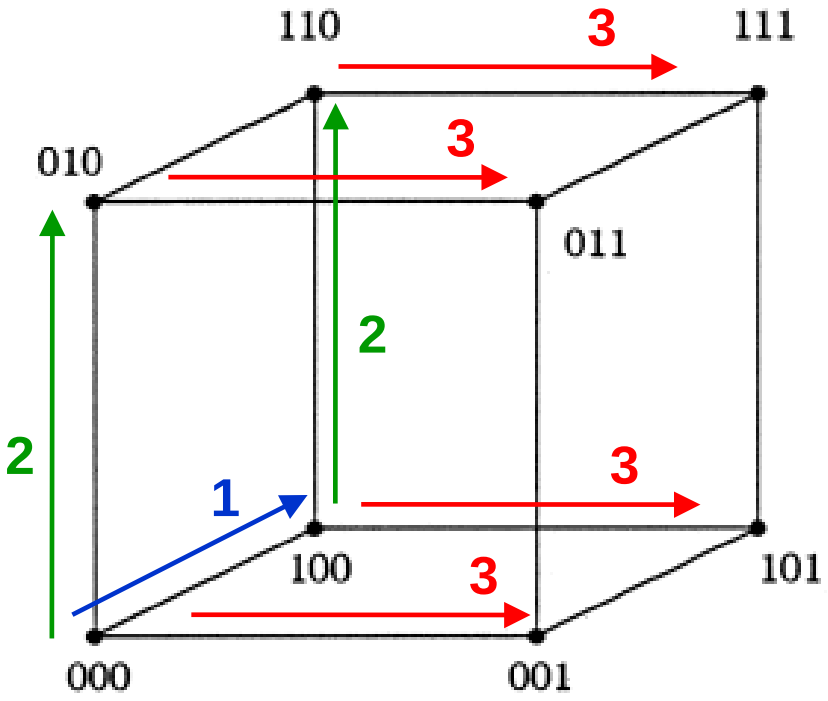
N=5



3-Cube Connected Cycles

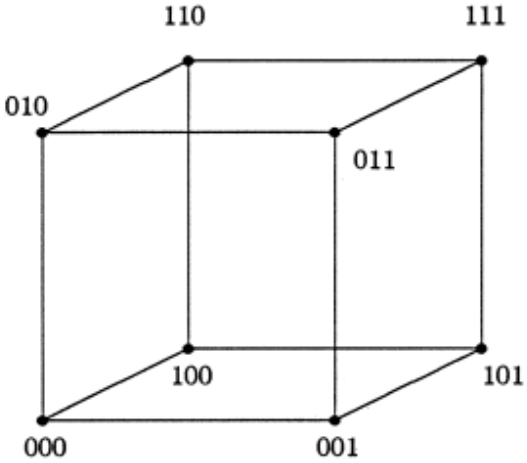
Hypercube – communication example

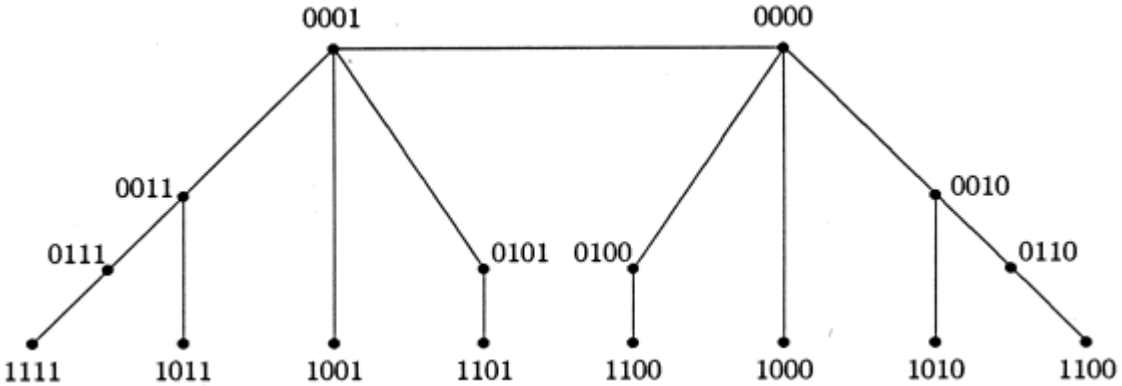
- One-to-all broadcast a All-to-one reduce



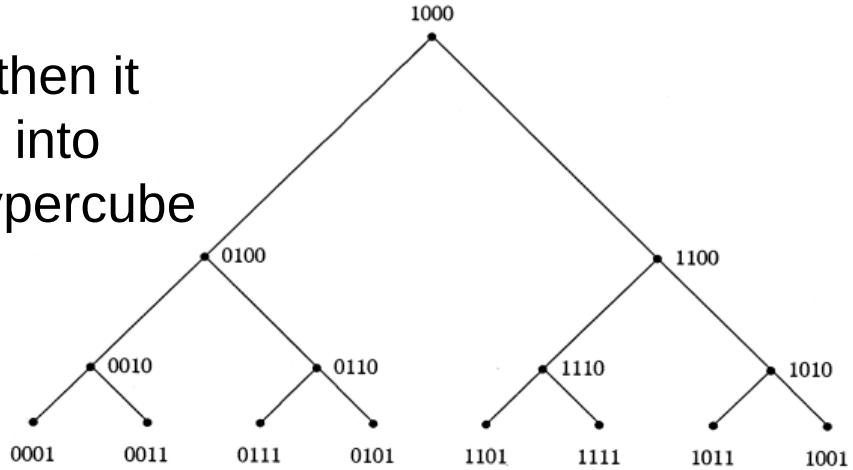
Hypercube – relation to other topologies

Tree structure of 4-dimensional hypercube:



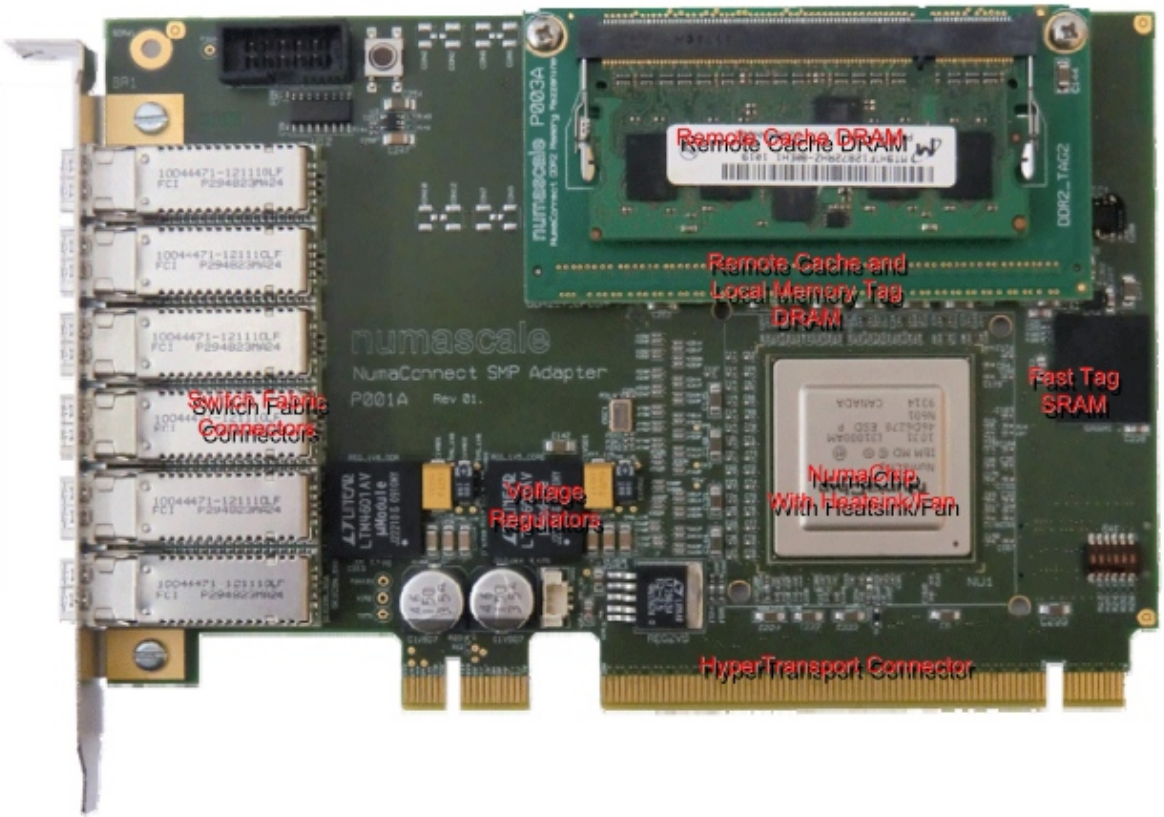


When the number of dimensions n>=3, then it is possible to map an n-level binary tree into a subgraph which is derived from the hypercube when one node/vertex is removed.
The tree edge then represents either a hypercube edge or a path of length 2.



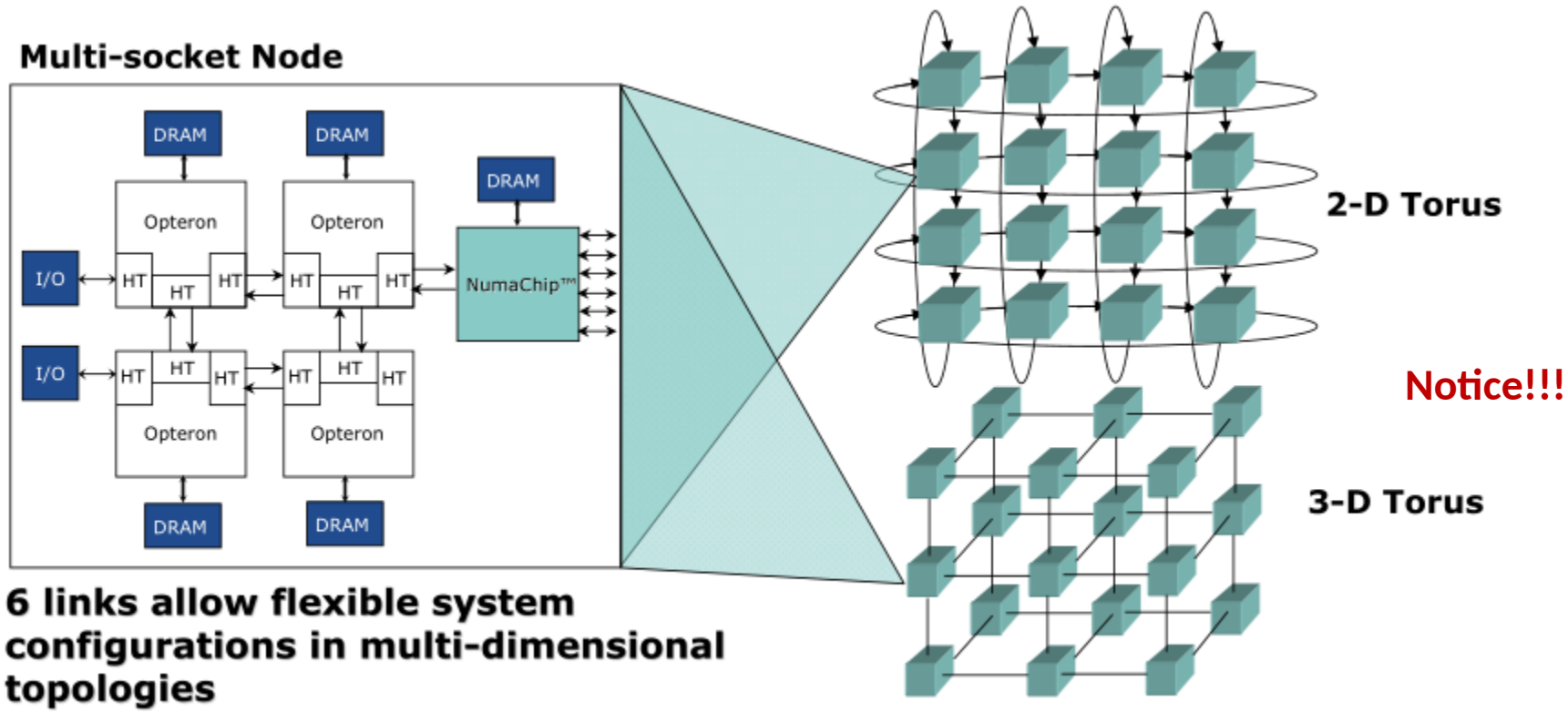
Example of scalable solution with many nodes...

numascale



- Scalable directory-based cache coherence in hardware
- Support for 4096 Nodes
- 4GB Cache 8 GB Tag (supports 240 GB Local Node RAM)

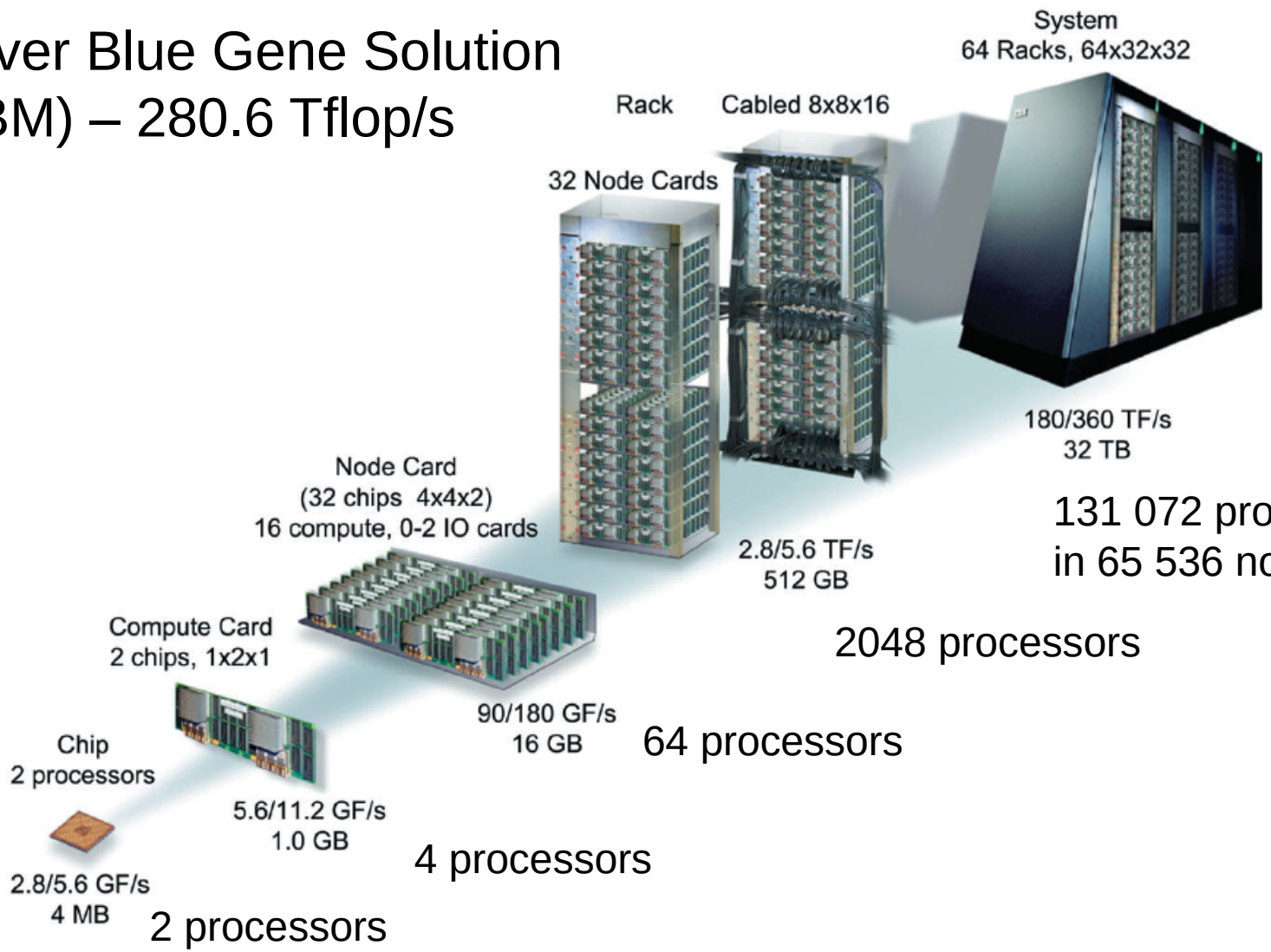
Example of scalable solution with many nodes...



- Use any Programming Model Available for the Node on the whole System (OpenMP, MPI, Threads, ...)
- NO Application Changes Required!

Motivation - Blue Gene Solution

eServer Blue Gene Solution (IBM) – 280.6 Tflop/s



131 072 processors in 65 536 nodes

2048 processors

64 processors

4 processors

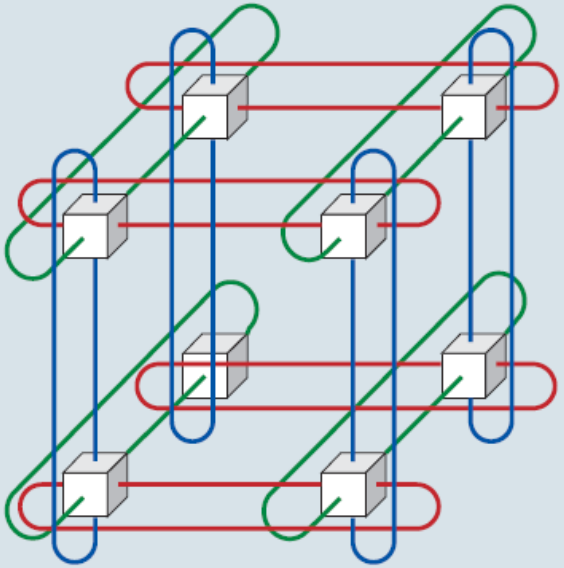
2 processors

Motivation - Blue Gene Solution

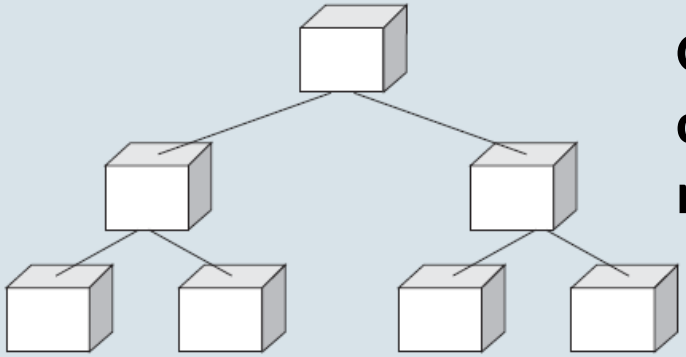
5 different networks for node interconnect are used

- **3D torus for point-to-point communication between the nodes (175 MBps in each direction),**
- **Global collective network – 350 MBps, 1.5µs latency (data from one node can be sent to all nodes – broadcast, or selected group; used for reduce as well),**
- **global barrier and interrupt network,**
- **control network (system boot, debug, monitoring temperature states, fans control and monitoring, …)**
- **gigabit Ethernet network for control and I/O operations**

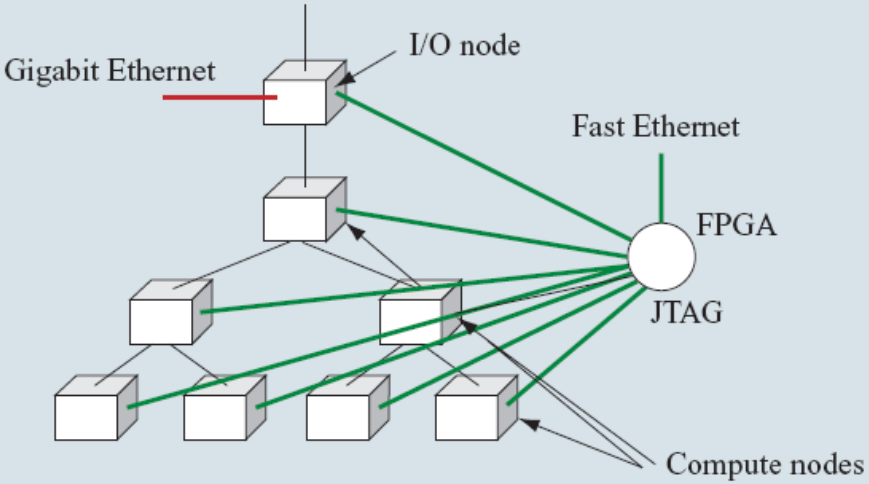
Motivation - Blue Gene Solution



3D Torus



Global collective network



Blue Gene/L control system network and Gigabit Ethernet networks

Topologies

Topologies of interconnect networks

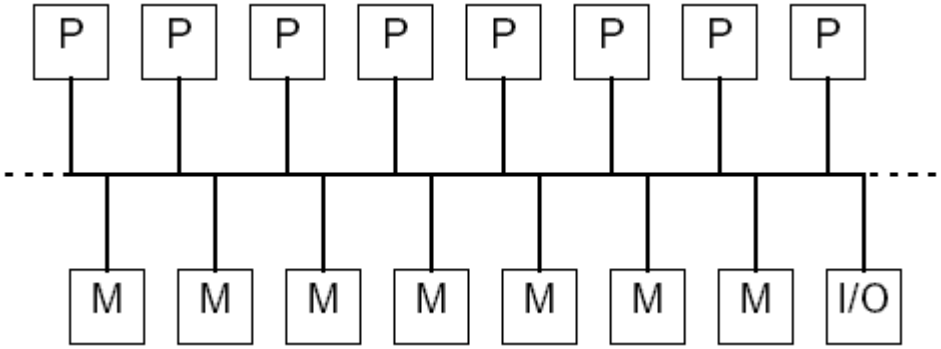
Static networks

- **linear array**
- **ring**
- **Chordal ring**
- **binary tree**
- **fat tree**
- **2D, 3D mesh**
- **2D, 3D torus**
- **hypercube**
- **Cube Connected Cycles (CCC)**

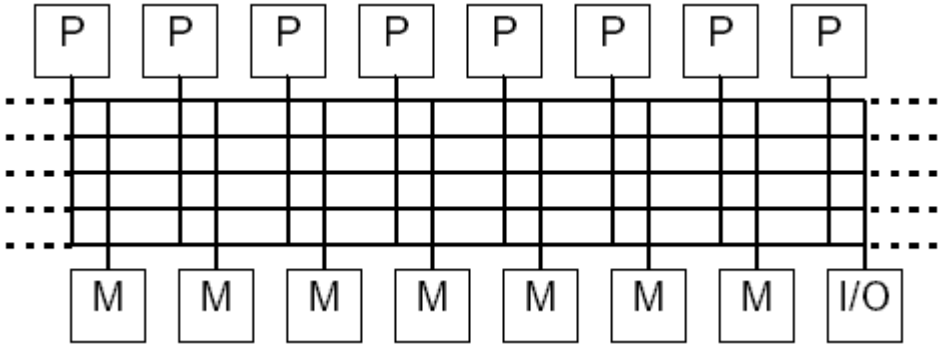
Dynamic networks

- **Bus network**
- **Single stage interconnect network (crossbar switches, …)**
- **Multistage interconnection networks (omega, Banyan, Cantor, Clos, …)**

Dynamic networks

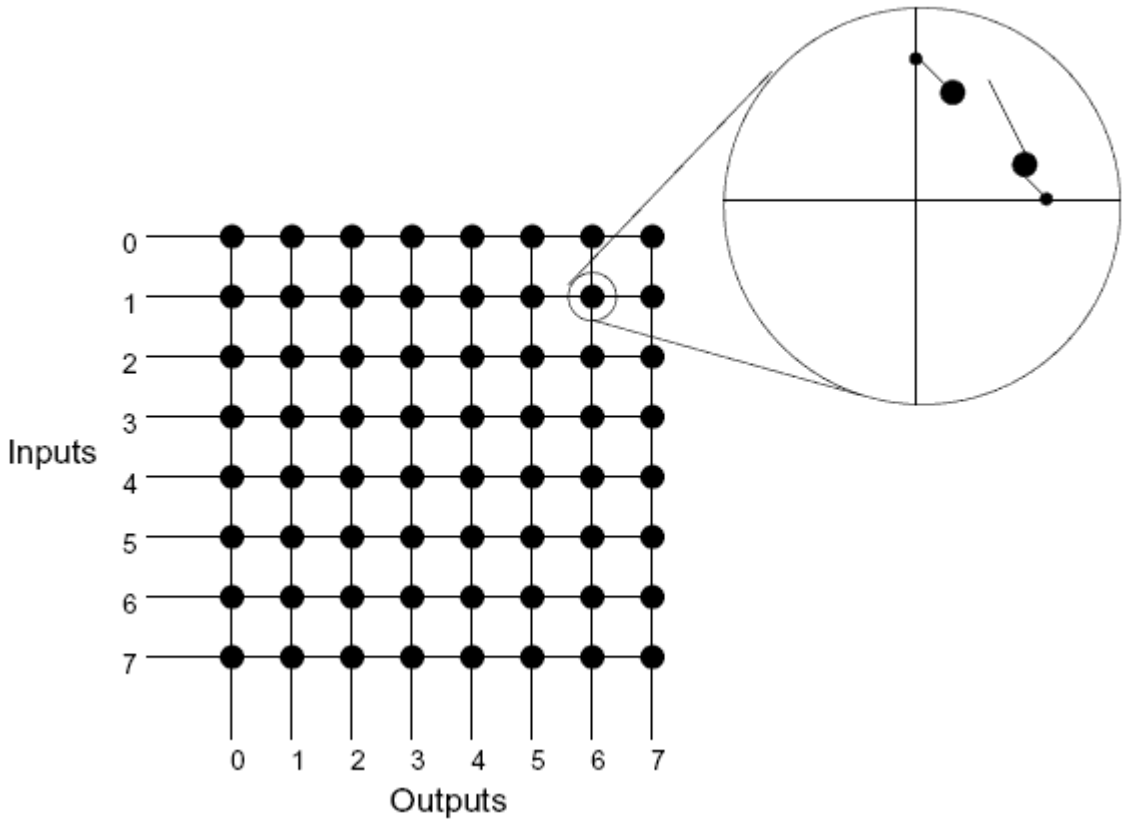


Bus-based multiprocessor system



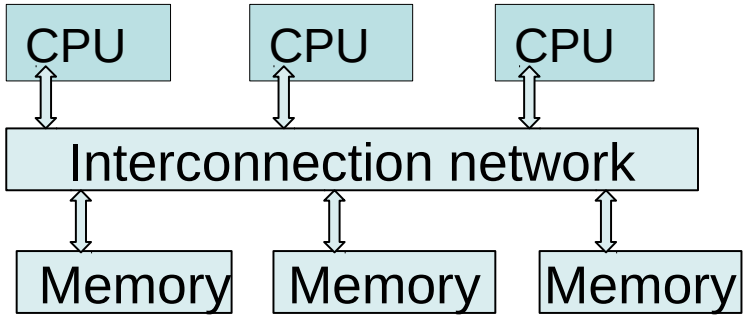
Multiple bus architecture

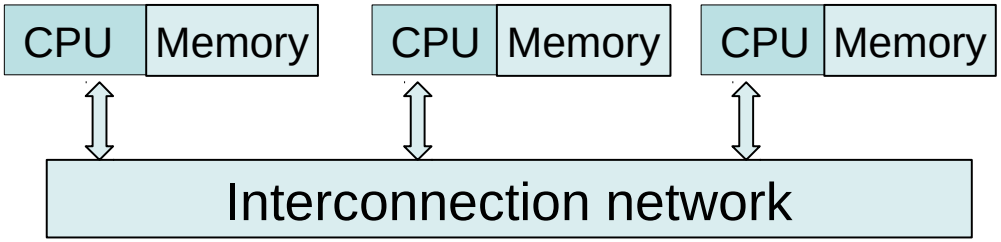
Dynamic networks – Single stage network

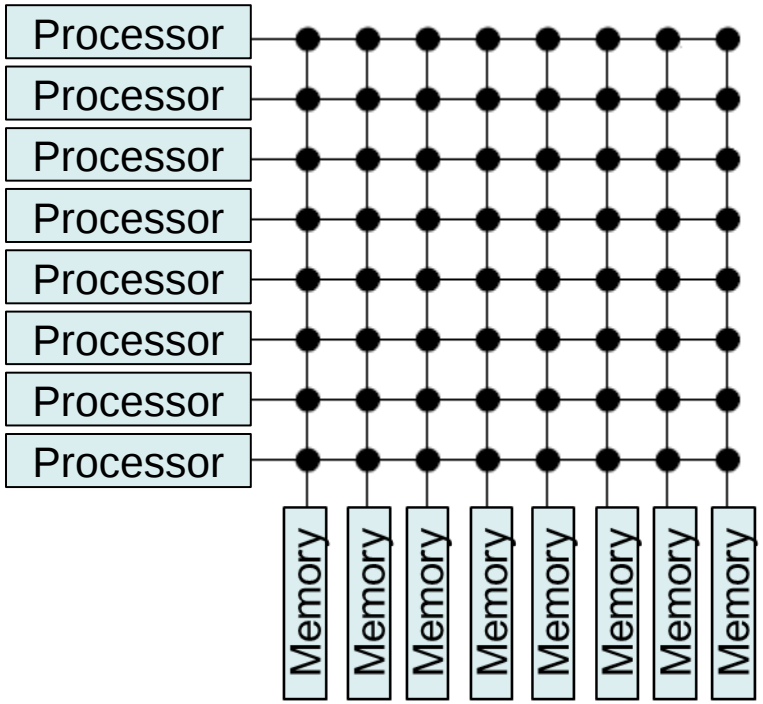


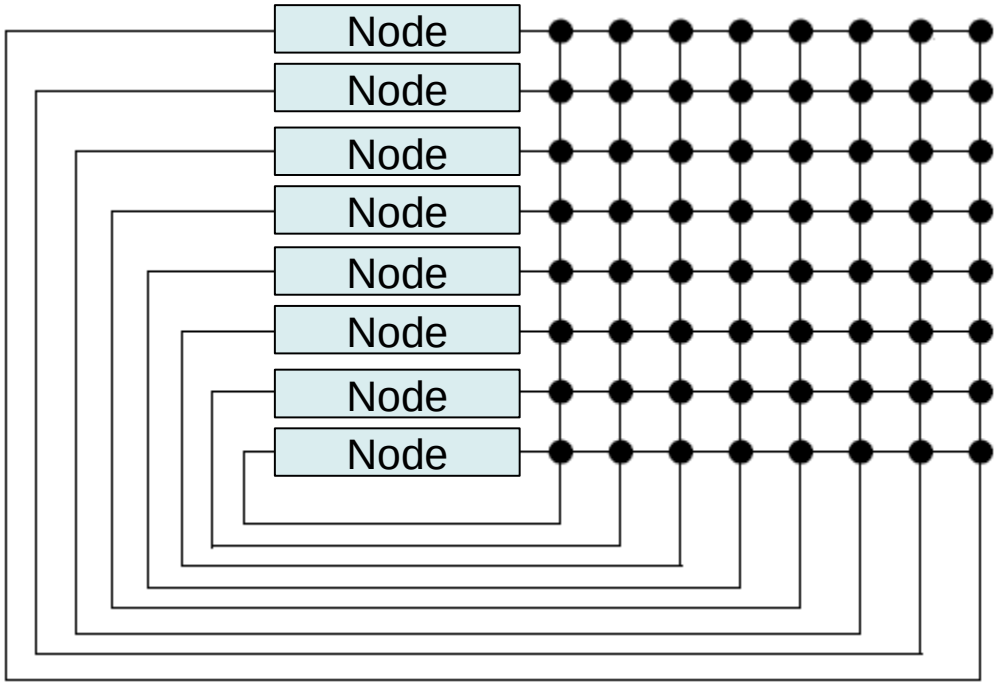
Crossbar switch – Cost N^2

Dynamic networks – Single stage network









Dynamic networks – Multistage interconnection networks

Multistage Interconnection Networks (MIN)

Intercon. type

- unicast MIN
- multicast MIN
- broadcast MIN

Blocking

- with blocking
- internal non-blocking
- external non-blocking
- rearrangeable non-blocking

Transport

- single - path
- multi - path

Buffer

- without buffers
- externally buffered
- internally buffered

Control

- centralized
- decentralized (selfrouting)

Dynamic networks – Multistage interconnection networks

- Unicast MIN (one-to-one, point-to-point) – in a given instant of time, an input may be connected to exactly one output.
- Multicast MIN (multipoint) – in a given instant of time, an input may be connected to multiple outputs.
- Broadcast MIN – in a given instant of time, an input is connected to all the outputs. Broadcast MIN is a special case of Multicast MIN.

- Single-Path MIN – all packets belonging to one virtual connection are using the same path.
- Multi-Path MIN – packets belonging to one virtual connection use multiple paths. Packets can be distributed randomly. Internal traffic in the network is independent from external traffic (that is, on services which are connected by network). Such networks require equipment for packet ordering on exit from network.

Dynamic networks – Multistage interconnection networks

- MIN with centralized control – control is performed by a central processor. The processor determines the path selection in the network according to connection requests.
- MIN with decentralized control – control is distributed among the switching elements (self-routing). A tag (additional information, address) has to be attached to each input packet and is used to determine the input/output combination in each switching element.

Dynamic networks – Multistage interconnection networks

- Networks with blocking – a particular input may not be able to connect to a particular output, even though no other input is connected there at the time. This may happen if an internal switching element on the path is currently accommodating another connection and there is no other path available. Blocking is associated with the loss of information or its delay. To ensure the quality of the connection, it is desirable to avoid blocking or to minimize it to a specified level.
- Internal non-blocking networks – any input may be connected to any output without canceling or reconfiguring any other internal path in the network. However, although the network is without any internal blocking, two or more inputs trying to connect to the same output at the same time still cause a conflict.
- External non-blocking networks – any input may connect to any output at any time. Information storage, i.e. Buffer memory, is required in the event of a conflict.
- Reconfigurable networks without blocking (Rearrangeable non-blocking) – (Rearrangeable non-blocking) – are a special case of blocking networks; however, they can always make connections from any input to any output. In the event of a conflict, the existing routes are restructured (reordered) and a new connection configuration is created.

Dynamic networks – Non-blocking MIN – Formal Definitions

- A network is **non-blocking** if it can change from satisfying A to satisfying B without tearing down paths in $A \cap B$, where A and B are any two connection assignments the network can realize.
- A network is **rearrangeably non-blocking** if when changing from satisfying A to satisfying B it may tear down and rebuild some paths in $A \cap B$, where A and B are any two connection assignments the network can realize.
- A network is **strictly non-blocking** if it can change from satisfying A to satisfying B without tearing down paths in $A \cap B$ for any routing of A, where A and B are any two connection assignments the network can realize.
- A network is **wide-sense non-blocking** if it can change from satisfying A to satisfying B without tearing down paths in $A \cap B$ if a proper routing procedure had been followed for A, where A and B are any two connection assignments the network can realize.

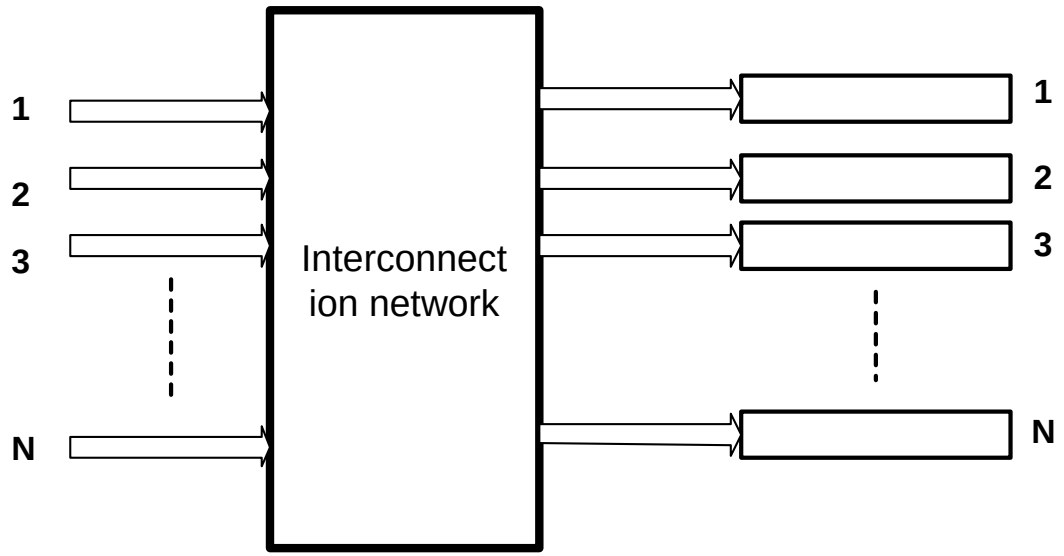
Dynamic networks – Multistage interconnection networks

- Externally Buffered MIN – buffers are located on network inputs, on network outputs, or both combined.
- Internally Buffered MIN – buffers are in the individual switching elements, i.e. inside the network. Even in the switching element, Input Buffering, Output Buffering, or Central Buffering may be used.

Main options to queue (buffer, store) packets:

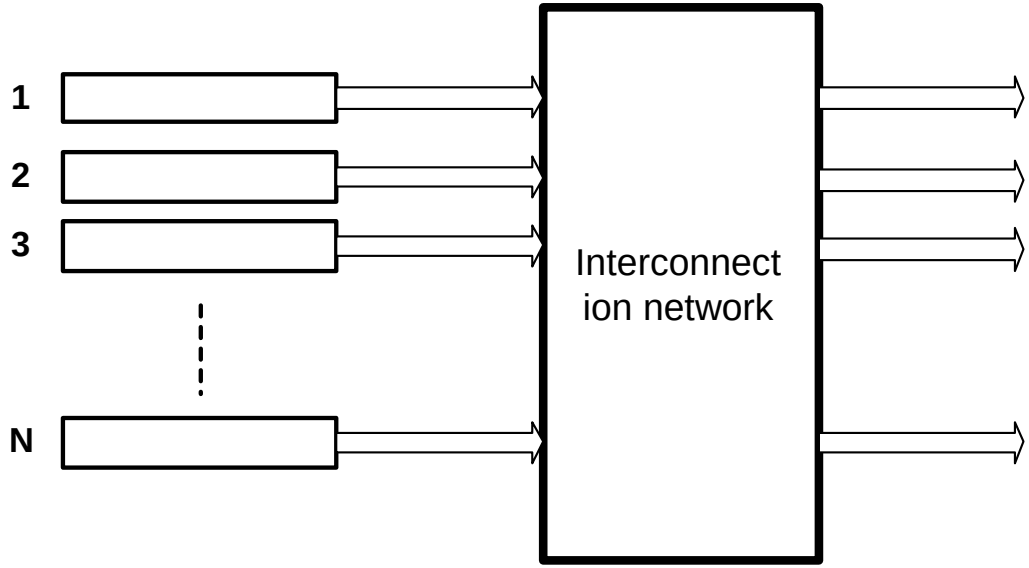
- Output Queuing/Buffering (OQ)
- Input Queuing/Buffering (IQ)
- Combined Input-Output Queuing/Buffering (CIOQ)
- Centralized Shared Queuing/Buffering (CSQ)
- Virtual Output Queuing/Buffering (VOQ)

OQ – output queuing



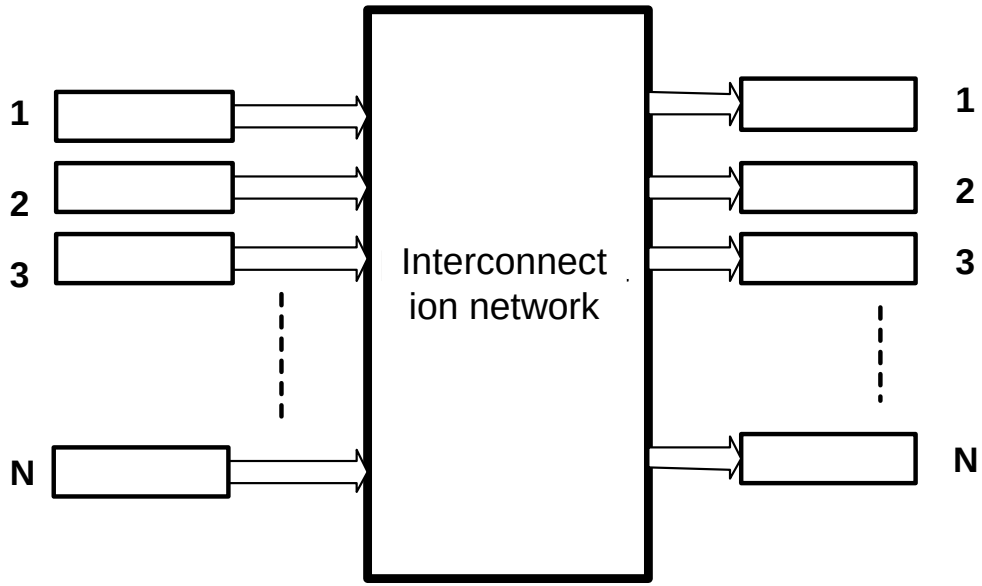
When the packet arrives at the input port, it is immediately (after passing through the network) stored in the buffer that is on the appropriate output port. Because packets destined for the same output port can come simultaneously from multiple input ports, the output buffer needs to sort the packets at a much faster rate than the input port. In the worst case scenario, it can be up to N times faster (where N is the number of input ports), if packets from all input ports are intended for one particular output port. However, the speed at which we can access the output buffer is limited.

IQ – input queuing



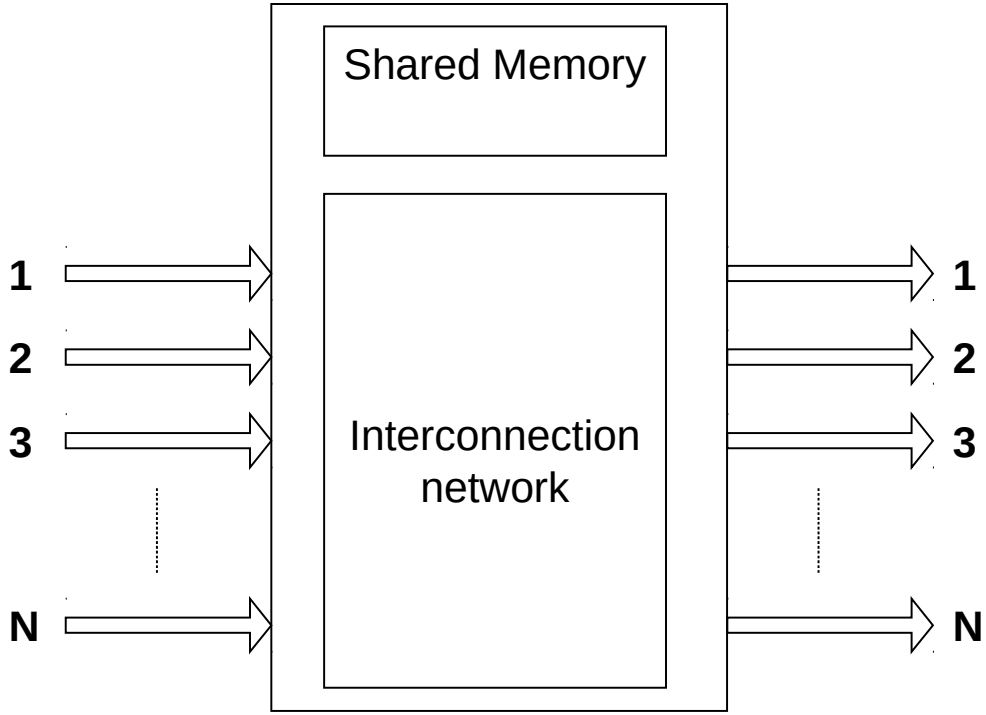
Input queuing does not have such limits as (for example) output queuing or centralized shared queuing. In this architecture, each input port has a FIFO of buffers, and only the first buffer in the queue is eligible for transmission over a given time period. The disadvantage of FIFO sorting is that when the buffer on the front of the queue blocks, all buffers behind it are blocked (for transmission), even when the corresponding output port is free. This is called Head-of-Line Blocking. Mathematical analysis and computer simulation has shown that HOL blocking limits the throughput of each input port to a maximum of 58.6 percent for random traffic density, and this value is still much lower in very dense traffic.

CIOQ – combined input-output queuing



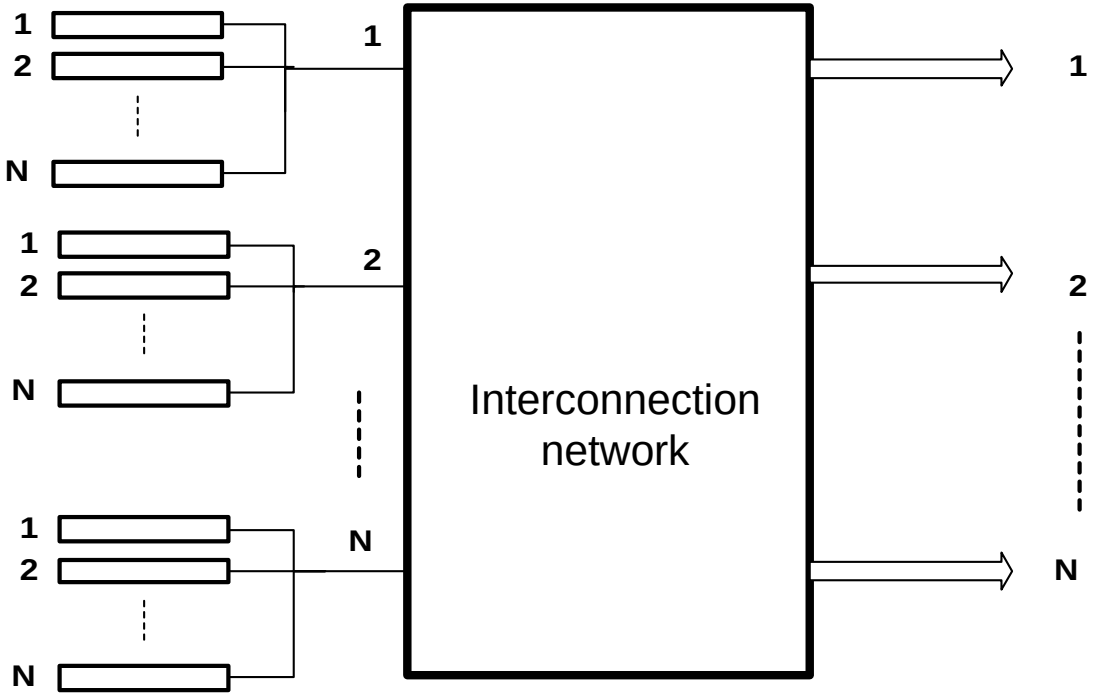
This queuing scheme is a combination of input and output queuing. It is a good compromise between performance and the demand to expand OQ and IQ switches. For given input switche, at most one packet can be delivered to the output port in one time slot. For the output a queue switch, up to N packets can be delivered to the output port for one time slot from different inputs. By using CIOQ, instead of these two extreme options, we can choose a compromise between them.

Centralized shared queuing



A buffer is shared by all the output ports of the switch and can be viewed as a shared memory unit supporting N concurrent write accesses for the N input ports and N concurrent read accesses for the N output ports. Because packets for the same output port can come from multiple input ports simultaneously, output ports must be able to read much faster than the input ports can write.

VO – virtual output queuing



This queuing scheme handles Head-of-Line blocking while maintaining its scalability advantage. With this method, each input port maintains an isolated queue for each output port. A key factor for achieving high performance using VOQ switches is the scheduling algorithm that is responsible for selecting the packets that should be transferred in each time unit from the input port to the output. Several such algorithms have already been proposed, such as PIM (parallel iterative matching), iSLIP (iterative round-robin matching with slip) and RPA (Reservation with Preemption and Acknowledgement). It has been shown that with less than four repetitions of the above-mentioned PIM control algorithm, the throughput of the switch exceeds 99 percent.

Basic types of multistage interconnection networks

Unicast blocking **single-path** interconnection networks

- Only **one-to-one** connections are provided by single-path unicast networks. Single-Path interconnection networks provide only one way (path) how to connect given input with given output.
  - Baseline network
  - Banyan network
  - Delta network
  - Omega network

Unicast blocking **multi-path** interconnection networks

- Multi-path networks provide alternative/multiple ways/paths between a given input and a given output while maintaining the self-routing properties of the network and only minimally contribute to its time complexity. The reliability and network throughput are improved.
  - Banyan network with split load
  - Data Manipulator – DM
  - Augmented Data Manipulator – ADM
  - Inverse Augmented Data Manipulator – IADM
  - gamma network

Basic types of multistage interconnection networks

Unicast non-blocking networks

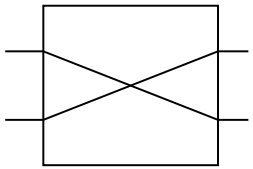
- All blocking network types require an action to suppress blocking. The most common is the placement of buffers in the network.
- The blocking problem can be solved when designing the network, creating a network without blocking. We distinguish two options. Either the networks are topologically the non-blocking, i.e. their architecture minimizes the probability of blocking, or non-blocking property is achieved by a network control mechanism which removes the blocking. The second case is also called reconfigurable networks:
  - Beneš network
  - Parallel baseline network
  - Clos network
  - Batcher network
  - Batcher-banyan network

Basic types of multistage interconnection networks

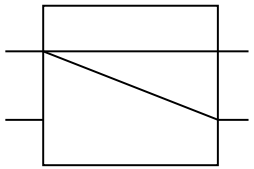
Multicast networks

- Multicast multistage interconnection networks are typically networks with N inputs and N outputs, where **any group of inputs can be connected to any group of outputs**. Each input port can be connected to more than one output, but each output port is usually connected at maximum to one input port.
- Multicast networks can make N^N of different connections, therefore they have more power than unicast networks that perform one-to-one permutation of inputs to outputs and realize a maximum of N! different connections.
- Basically, each unicast network can operate as a multicast network if its switching elements can connect their inputs to multiple outputs.

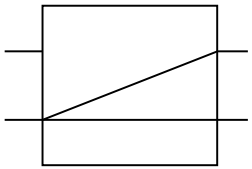
Straight-trough



Exchange

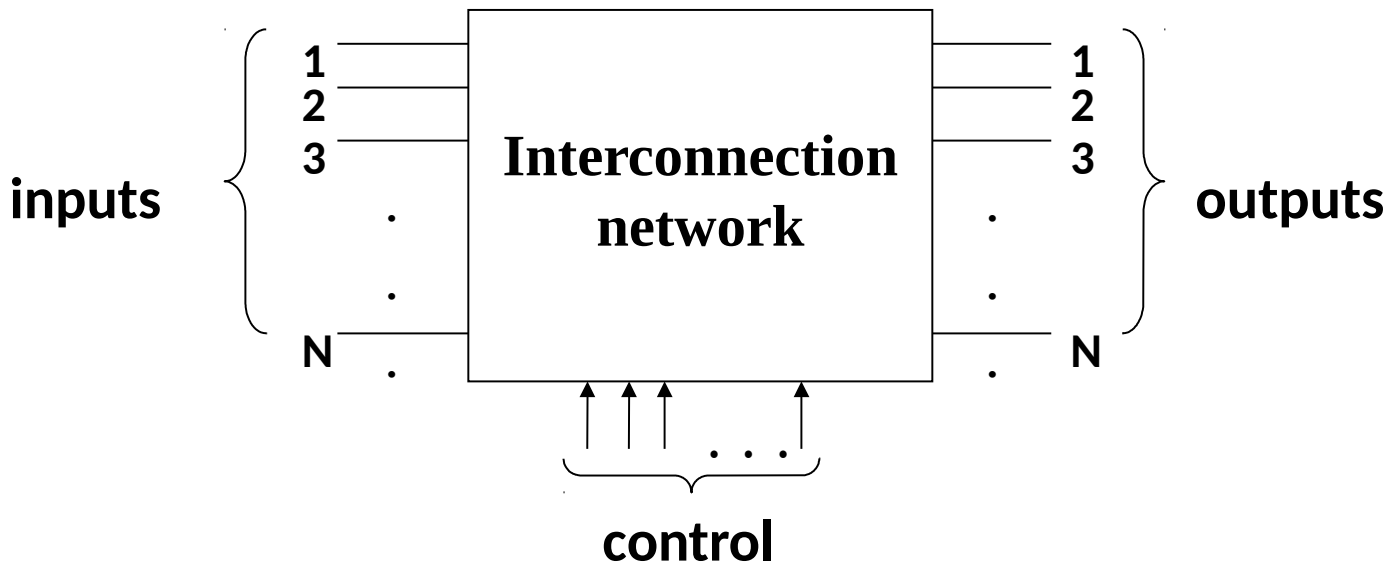


Upper-broadcast



Lower-broadcast

Block diagram interconnection network



- Interconnection network – IN [N x N] is device which allows to connect any of its N inputs (X0, X1, …, XN-1) with any of its N outputs (Y0, Y1, …, YN-1).
- The basis are interconnection networks which allows to realize only one-to-one assignment, only one output can be connected to one input -> **permutation networks**
- **For inputs A labeled from zero to N-1: A = {0, 1, 2, ... , N-1}**

  is result of permutation function defined as: f:A → A

Basic permutations

Basic permutations:

- **perfect shuffle permutation** (**σ**)
- **butterfly permutation** (**β**)
- **reversing permutation** (**ρ**)
- **exchange permutation** (**E**)

- Perfect shuffle permutation:

$$\sigma(j)=\left(2j+\left\lfloor\frac{2j}{N}\right\rfloor\right) mod\, N$$

- In general, the shuffle operation is defined in the form:

$$\sigma(j,K)=\left(Kj+\left\lfloor\frac{Kj}{N}\right\rfloor\right) mod\, N$$

- where K is "number of hands" required for shuffle for which it is valid $K = 2^k$, k = 0, 1, ..., n-1.

Perfect shuffle permutation

- If port number *x* is represented in the binary representation then the permutation of the perfect shuffle corresponds to the cyclic shift of the binary representation by one bit to the left:

$$\sigma(x) = \sigma([x_n \;\; x_{n-1} \;\; x_{n-2} \ldots x_2 \;\; x_1]) = [x_{n-1} \;\; x_{n-2} \ldots x_1 \;\; x_n]$$

- If only part of binary representation is cyclic shifted then k-th sub-shuffle $\sigma_{(k)}(x)$ – permutation is obtained which is defined for $1 \leq k \leq n$ by next pattern:

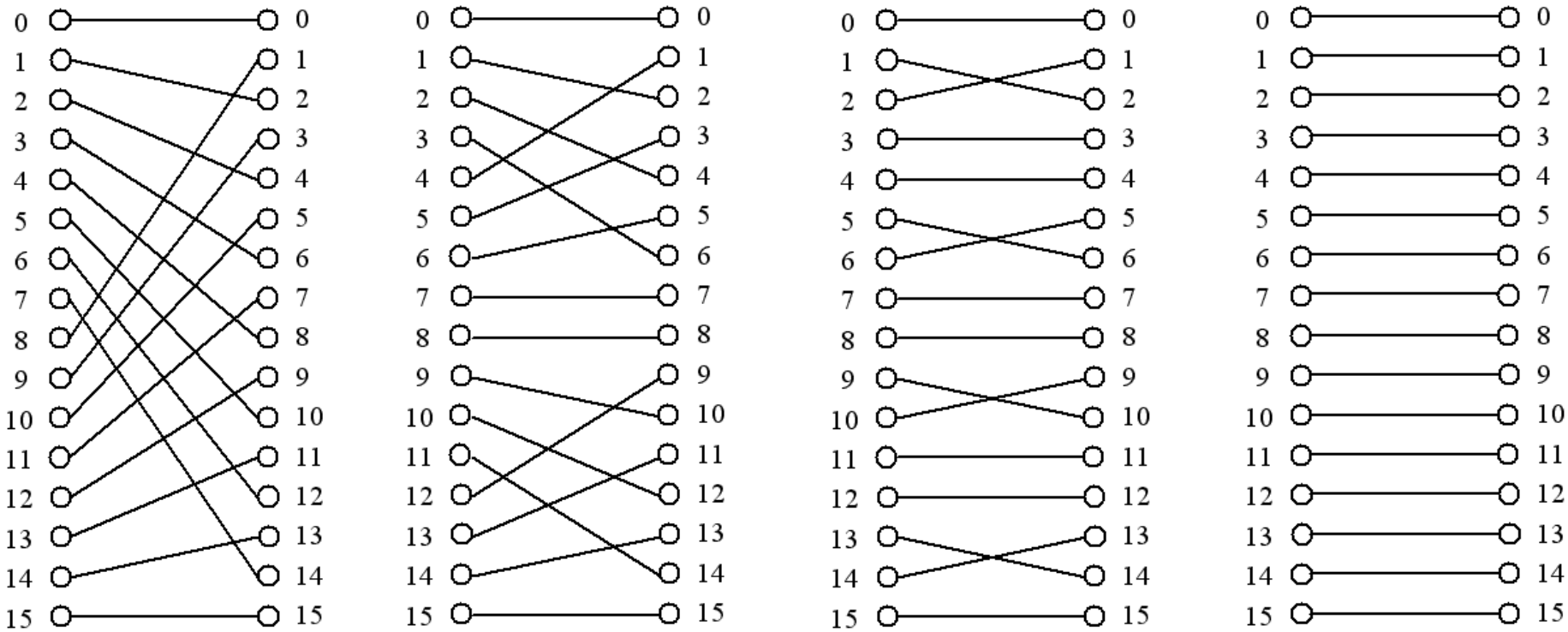
$$\sigma_{(k)}(x) = \sigma_{(k)}([x_n \;\; x_{n-1} \;\; x_{n-2} \ldots x_2 \;\; x_1]) = [x_n \;\; \ldots x_{k+1} \;\; x_{k-1} \ldots x_1 \; x_k]$$

- For this k-th sub-shuffle definition is valid:
  $\sigma_{(1)}(x) \equiv x, \;\; \sigma_{(n)}(x) \equiv \sigma(x)$

Perfect shuffle and k-th sub-shuffle permutation

Example for N=16, $n=\log_2 N=4$:

- $\sigma_{(4)}(0) = \sigma_{(4)}([0000]) = [0000] = 0$
- $\sigma_{(4)}(1) = \sigma_{(4)}([0001]) = [0010] = 2$
- $\sigma_{(4)}(2) = \sigma_{(4)}([0010]) = [0100] = 4$ etc.

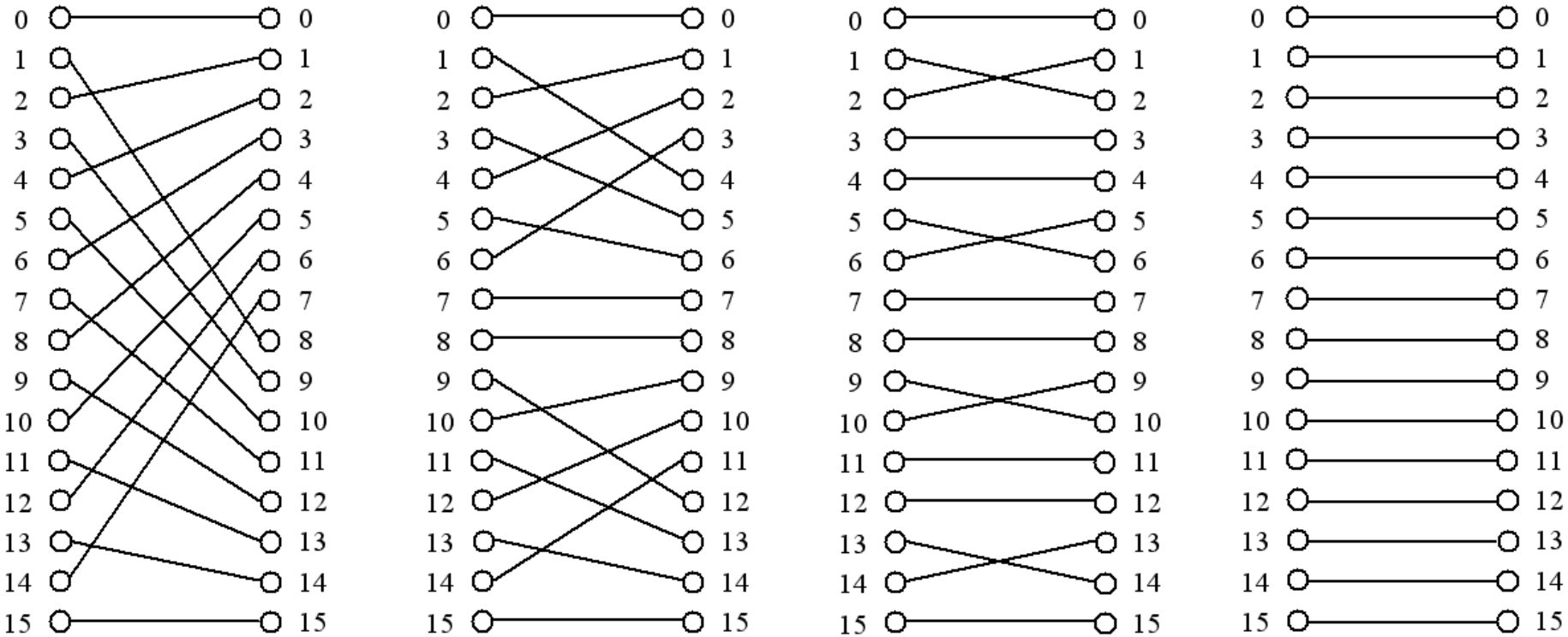


$\sigma_{(4)}(x) \equiv \sigma(x)$ $\sigma_{(3)}(x)$ $\sigma_{(2)}(x)$ $\sigma_{(1)}(x) \equiv x$

Inverse perfect shuffle and k-th sub-shuffle permutation

Inverse perfect shuffle corresponds to cyclic shift of binary representation by one bit right, similar for k-th inverse sub-shuffle:

$$\sigma^{-1}(x) = \sigma^{-1}([x_n \;\; x_{n-1} \;\; x_{n-2} \ldots x_2 \;\; x_1]) = [x_1 \;\; x_n \;\; x_{n-1} \;\; x_{n-2} \ldots x_2]$$



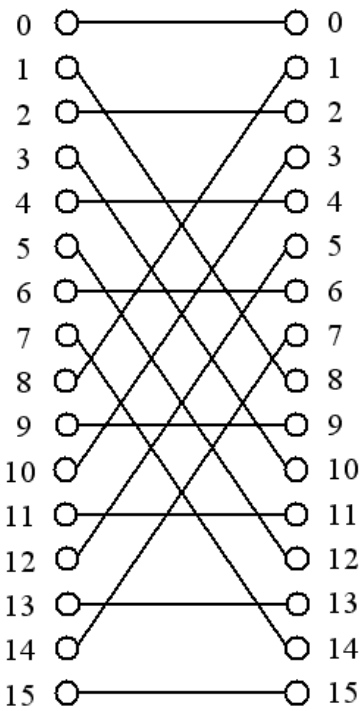
$\sigma_{(4)}^{-1}(x) \equiv \sigma^{-1}(x)$ $\sigma_{(3)}^{-1}(x)$ $\sigma_{(2)}^{-1}(x)$ $\sigma_{(1)}^{-1}(x) \equiv x$

Butterfly permutation

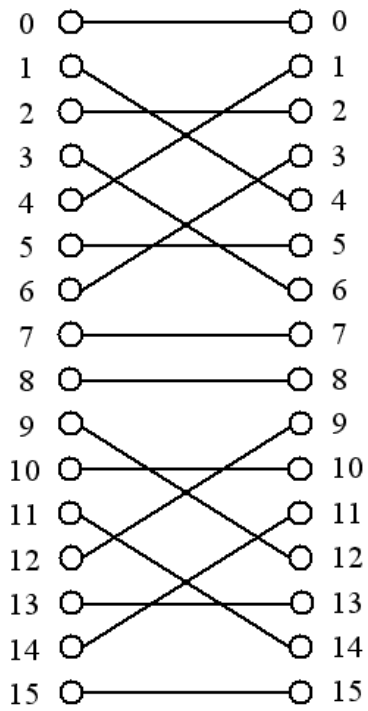
Butterfly permutation:

k-th butterfly permutation $\beta_{(k)}(x)$ for $1 \le k \le n$, $n = \log_2 N$ is defined as swap of the first and k-th bit:

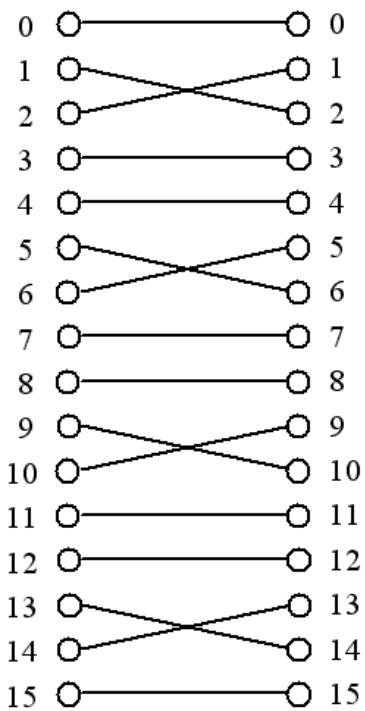
$$\beta_{(k)}(x) = \beta_{(k)}([x_n \;\; x_{n-1} \;\; x_{n-2} \ldots x_2 \;\; x_1]) = [x_n \;\; x_{n-1} \ldots x_{k+1} \;\; x_1 \;\; x_{k-1} \ldots x_2 \;\; x_k]$$



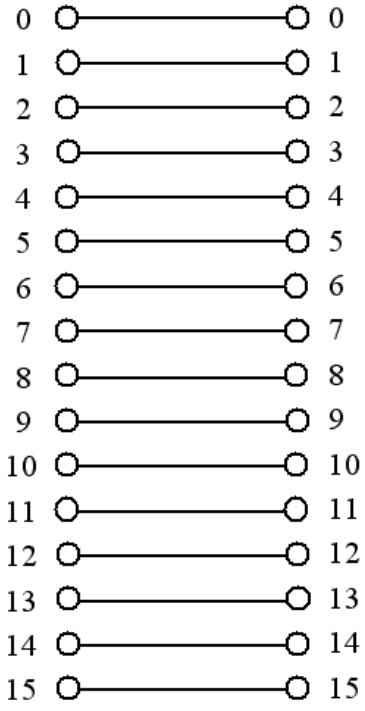
$\beta_{(4)}(x) \equiv \beta_{(4)}^{-1}(x)$



$\beta_{(3)}(x) \equiv \beta_{(3)}^{-1}(x)$



$\beta_{(2)}(x) \equiv \beta_{(2)}^{-1}(x)$



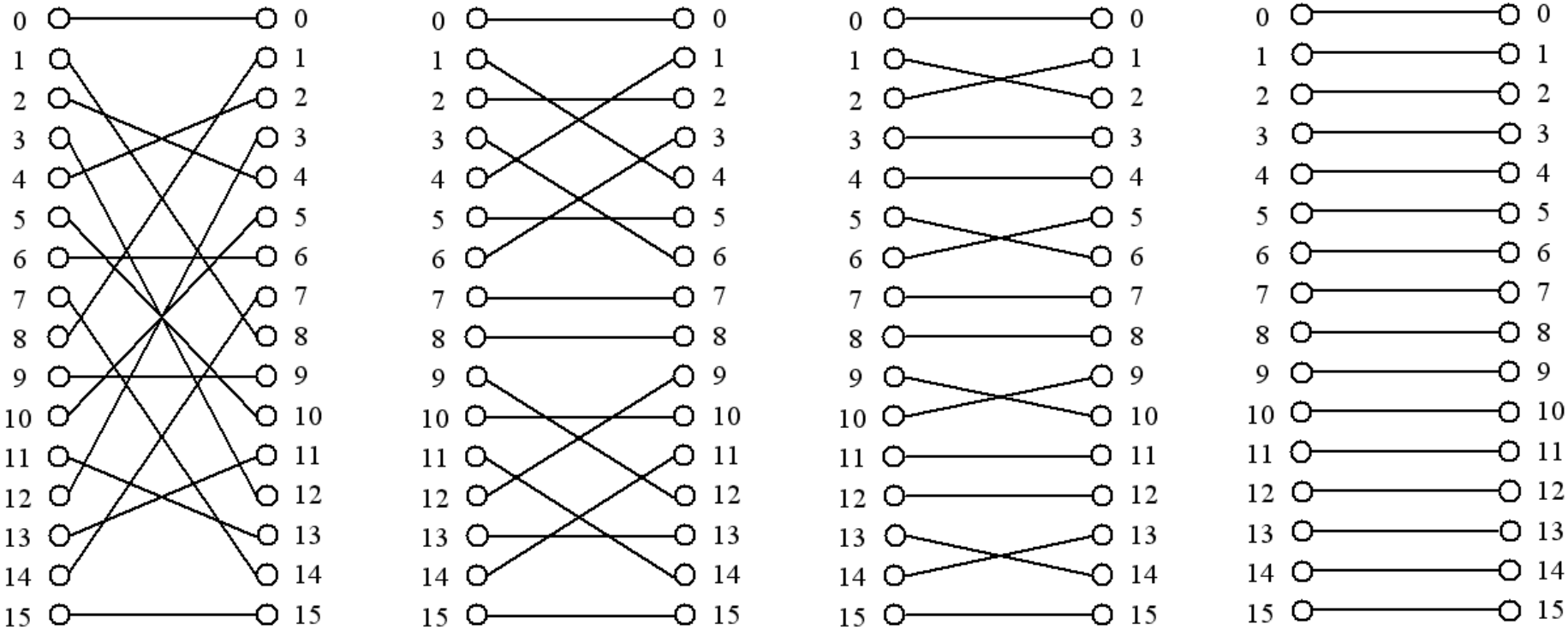
$\beta_{(1)}(x) \equiv \beta_{(1)}^{-1}(x)$

Reversing permutation

Reversing permutation:

is defined as swapping more significant bits with their less significant counterparts (mirroring):

$$\rho(x) = \rho([x_n \;\; x_{n-1} \;\; \ldots \; x_2 \;\; x_1]) = [x_1 \;\; x_2 \; \ldots \; x_{n-1} \;\; x_n]$$



$\rho = \rho_4$ ρ_3 ρ_2 $\rho_1 = x$

Exchange permutation or switch

Exchange permutation:

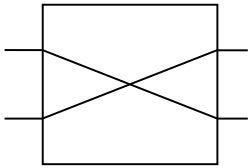
Id defined $\hat{x}$ as $\hat{x} = [x_n \;\; x_{n-1} \;\; \ldots \;\; x_2 \;\; \overline{x_1}]$, where $\overline{x_1}$ denotes negation of (least significant) bit. Two cases can be distinguished for exchange switch:

either $e(x) = x$ and $e(\hat{x}) = \hat{x}$

or $e(\hat{x}) = x$ and $e(x) = \hat{x}$

Straight-trough

(Identity)



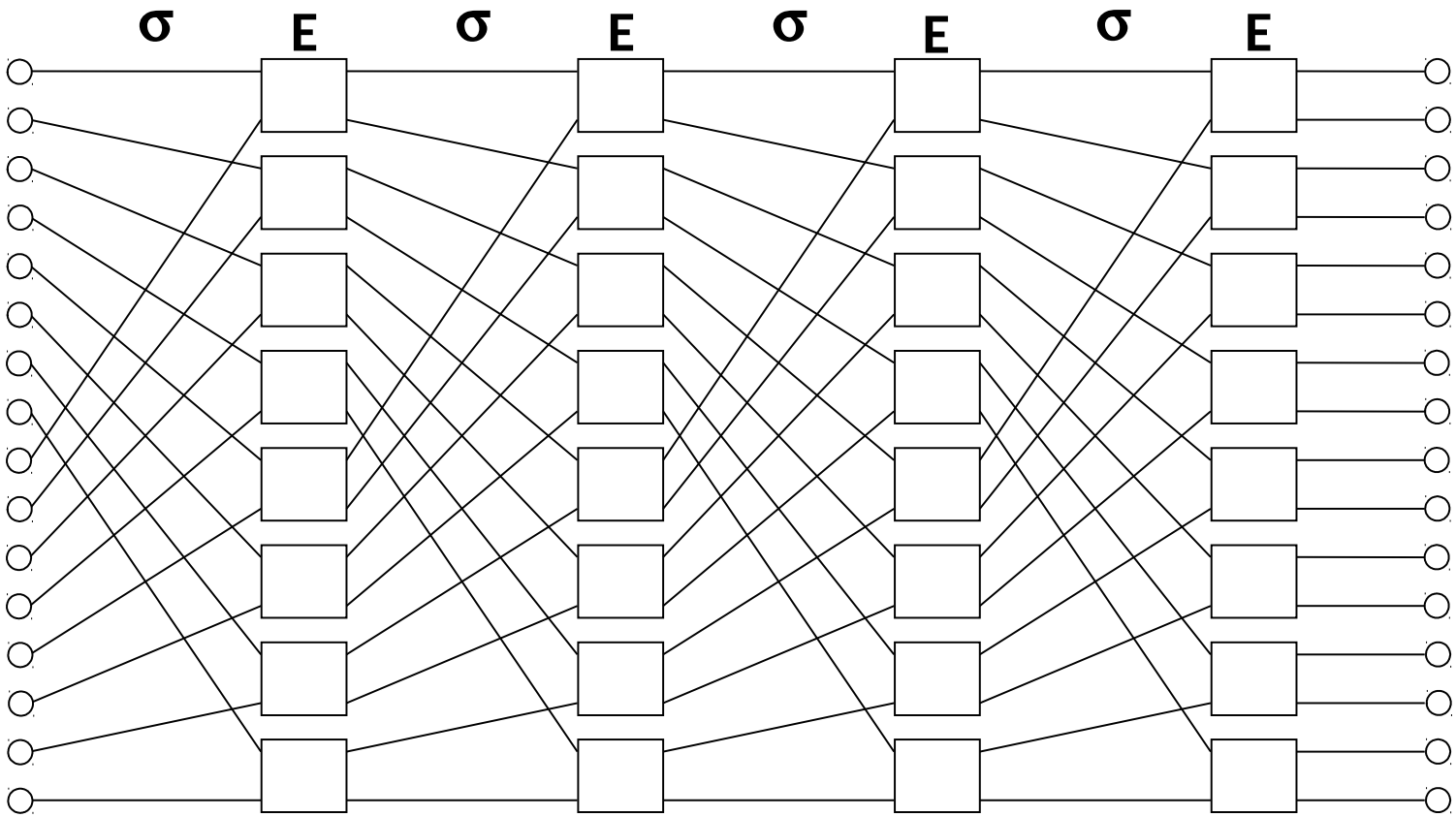
Exchange

(Exchange)

Omega network

$$\Omega_N = \sigma E \sigma E \ldots \sigma E = (\sigma E)^n \qquad n = \log_2 N$$

- Example for N=16, n=4: $\Omega_{16} = (\sigma E)^4$



Omega network

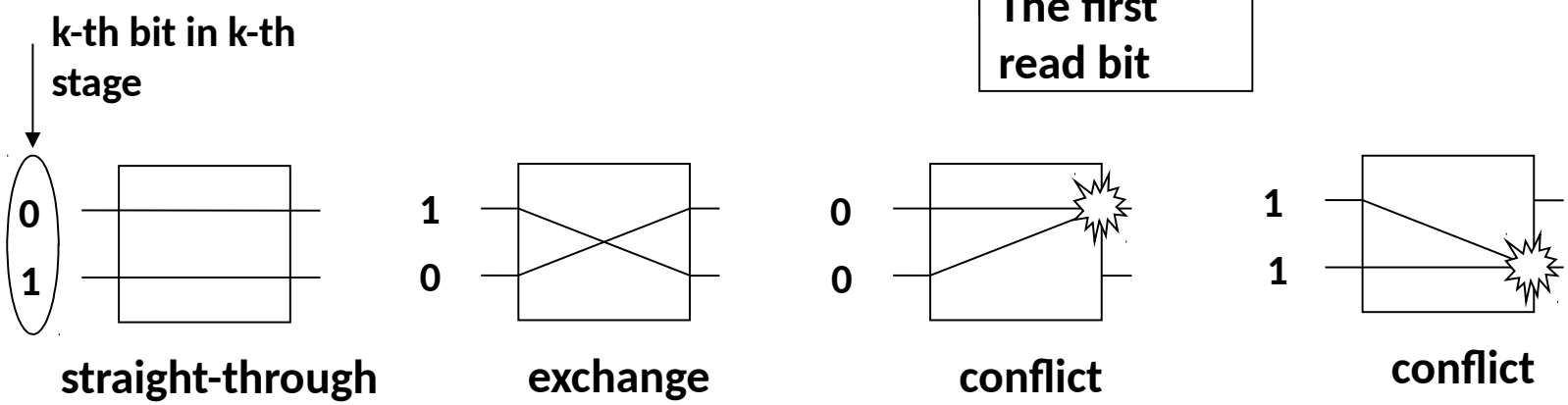
Self-routing algorithm:

- Following rule makes possible to control setup of individual switches in the network based directly on the destination address, that is binary representation of output port number (routing tag = destination port id). Switch in k-th stage reads k-th bit of routing tag (starting by MSB towards LSB) and if this bit value is:
  - 0 then select upper output
  - 1 then select lower output

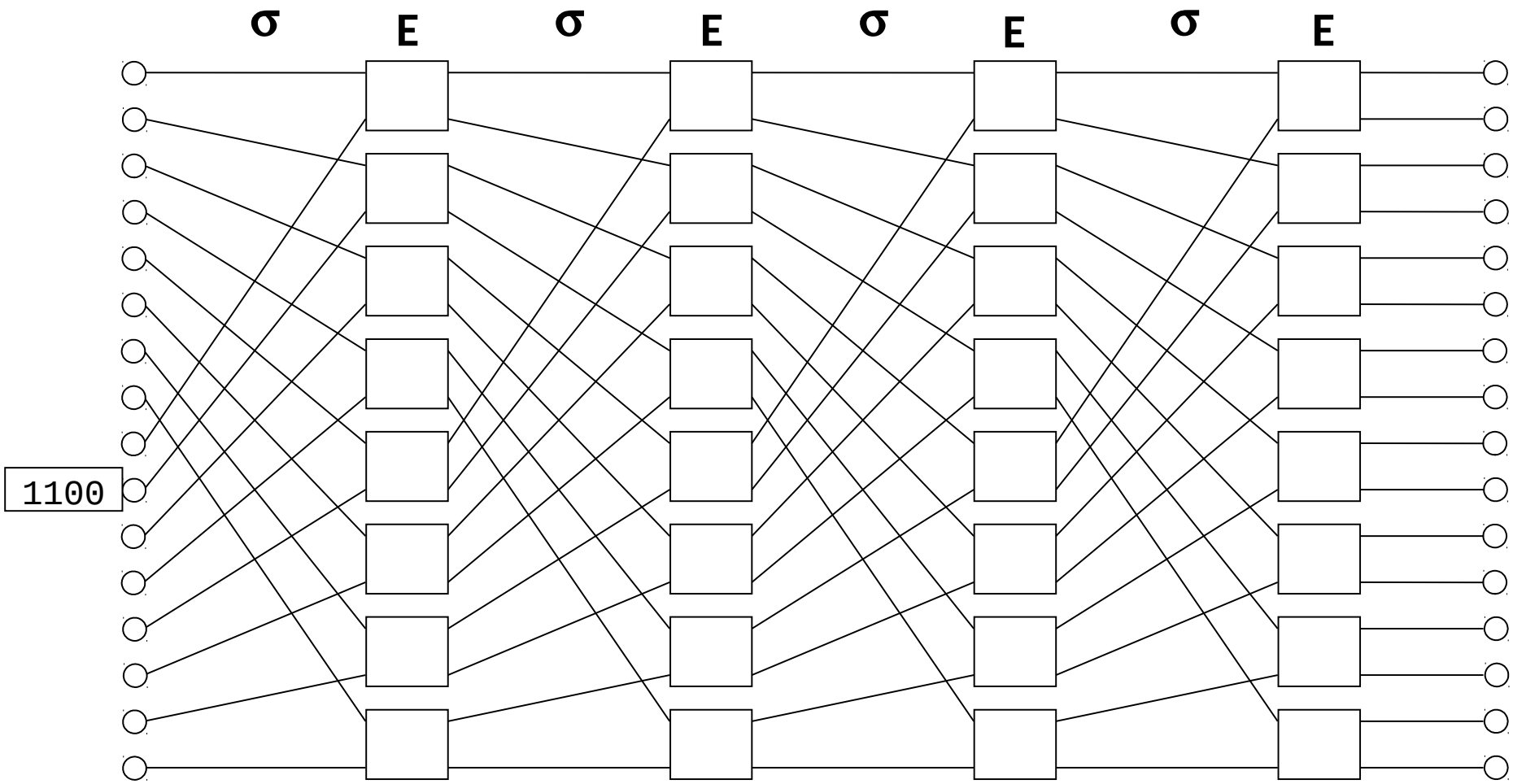
direction of bit read

$tag = [\ d_n\ d_{n-1}\ \ldots\ d_2\ d_1\ ]$

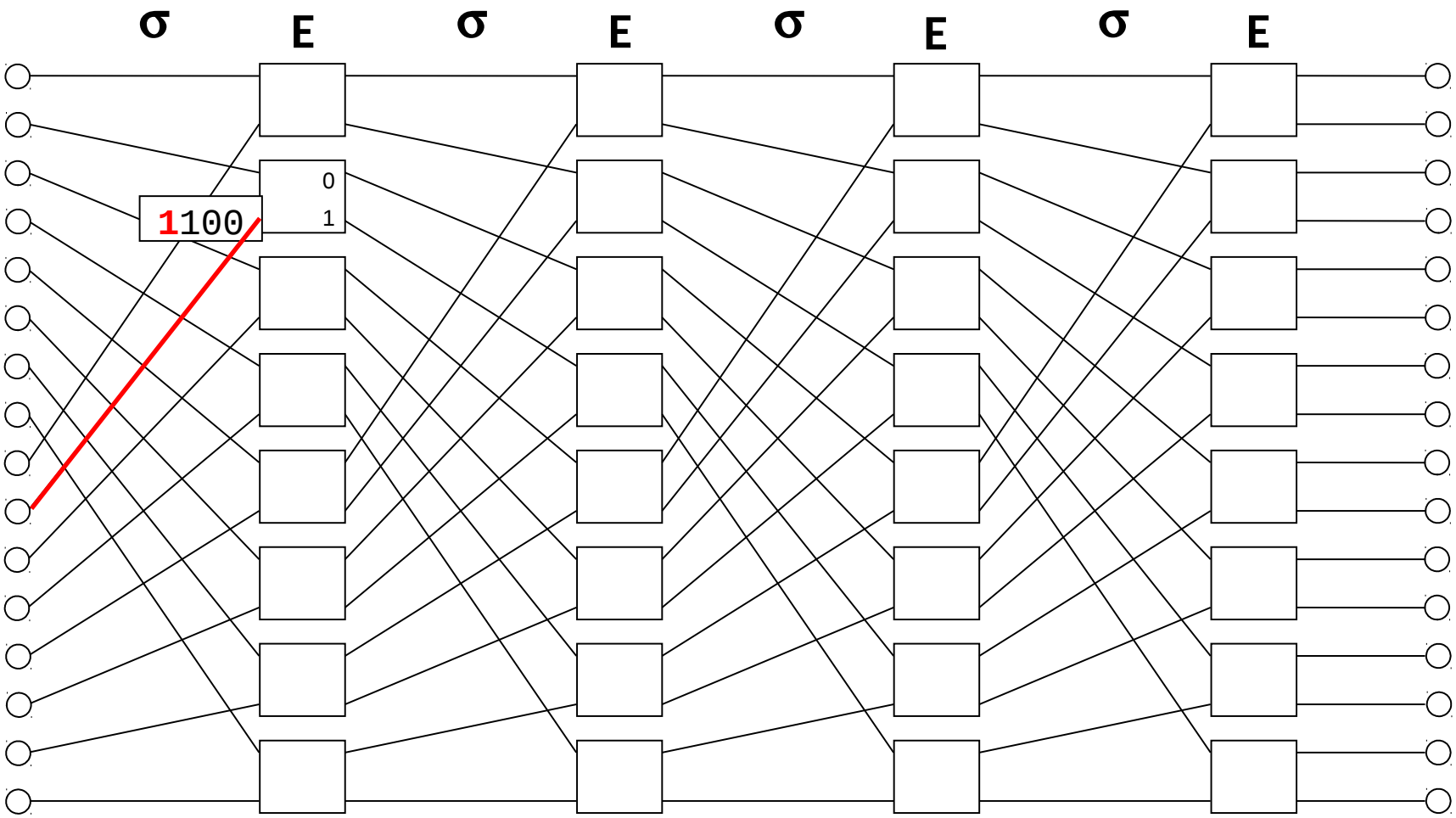
The first read bit



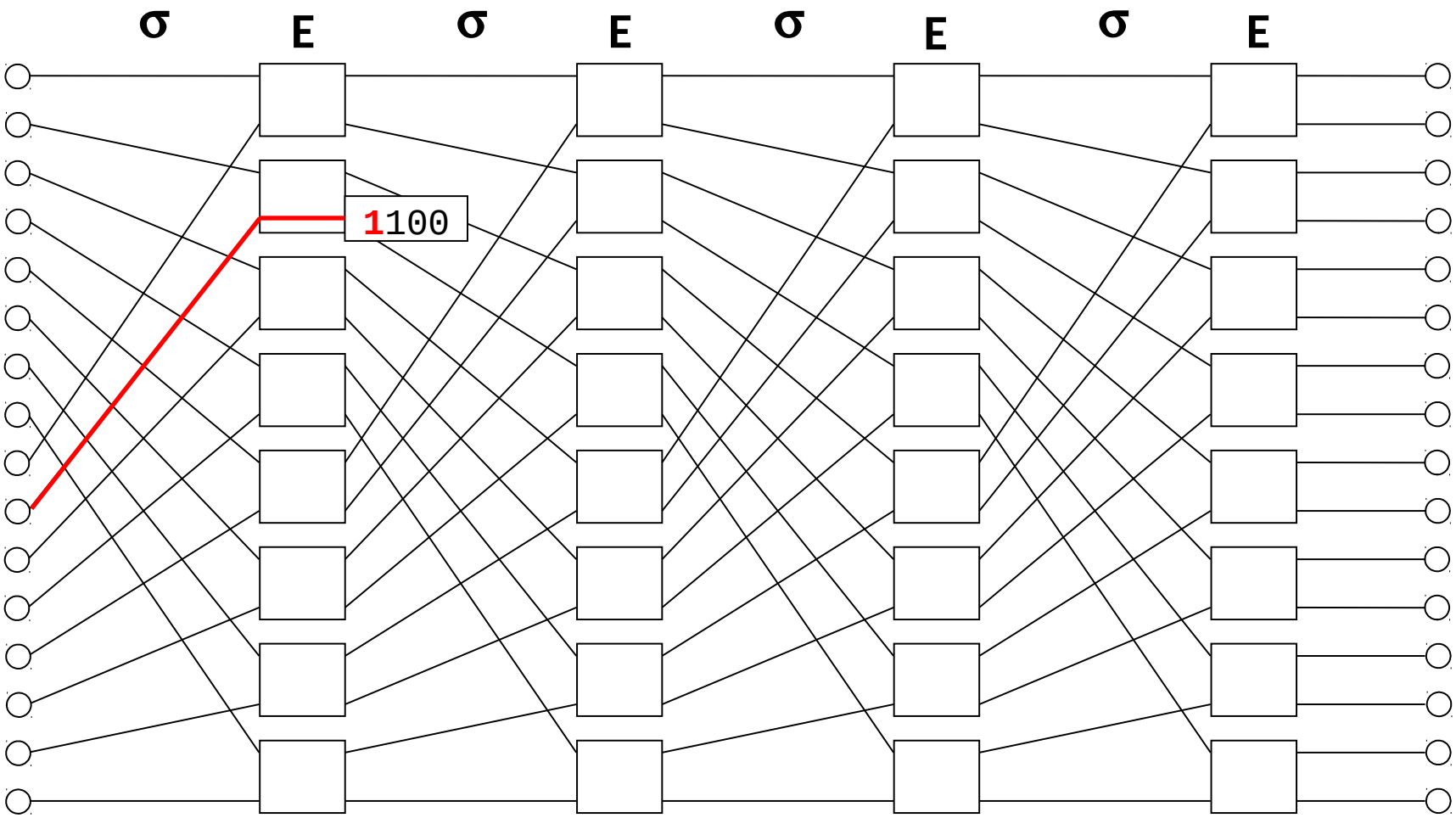
Omega network



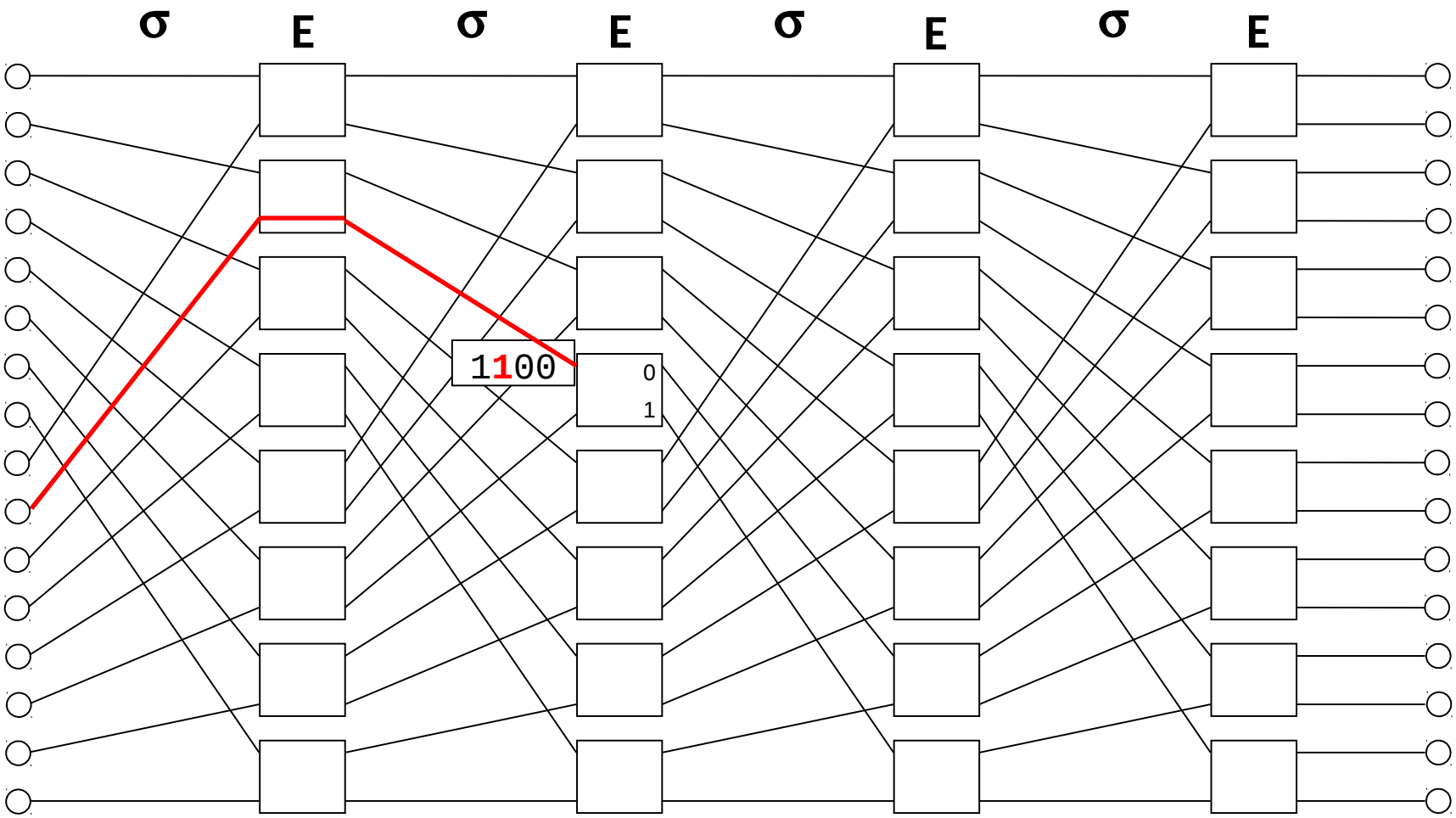
Omega network



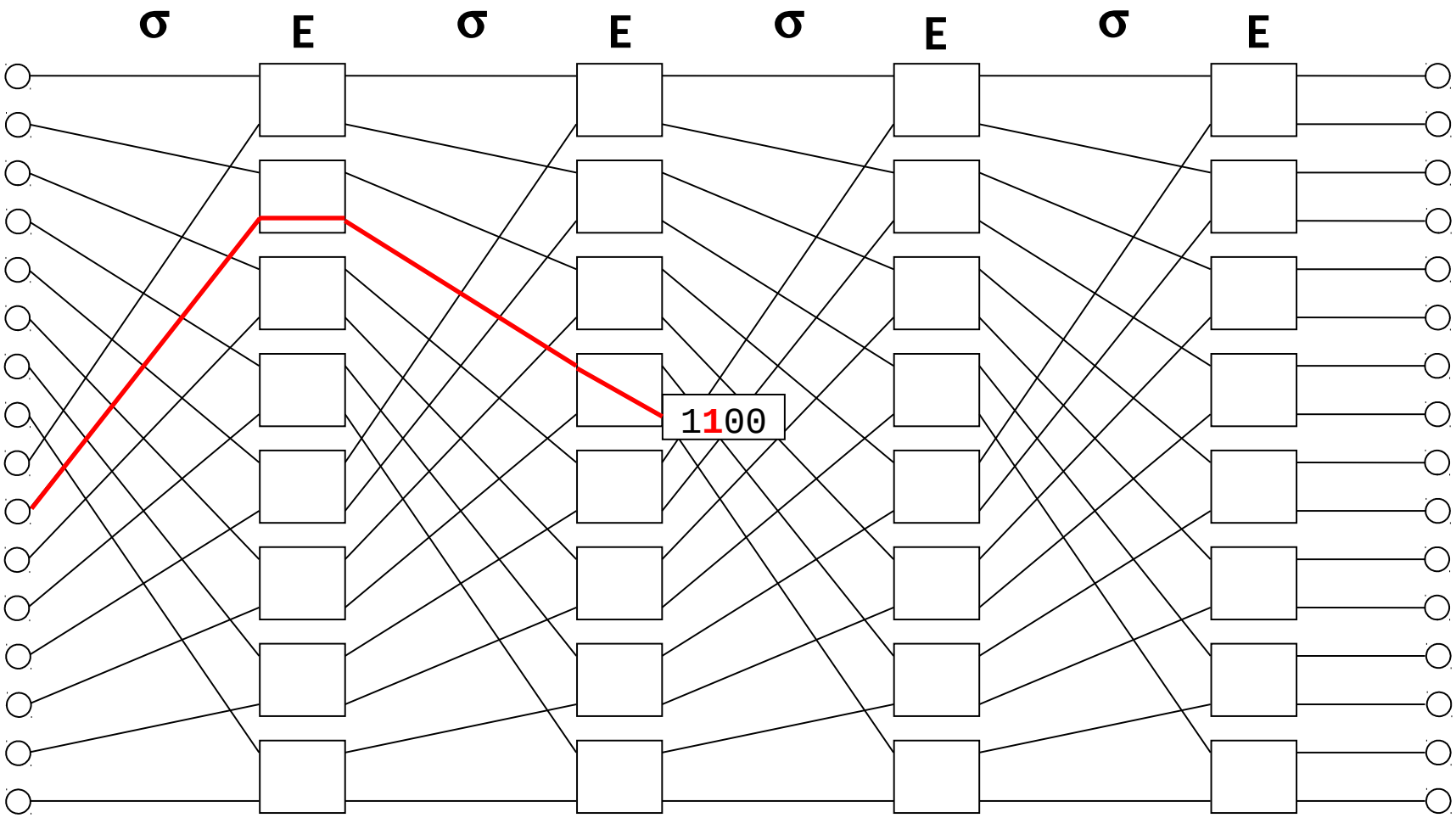
Omega network



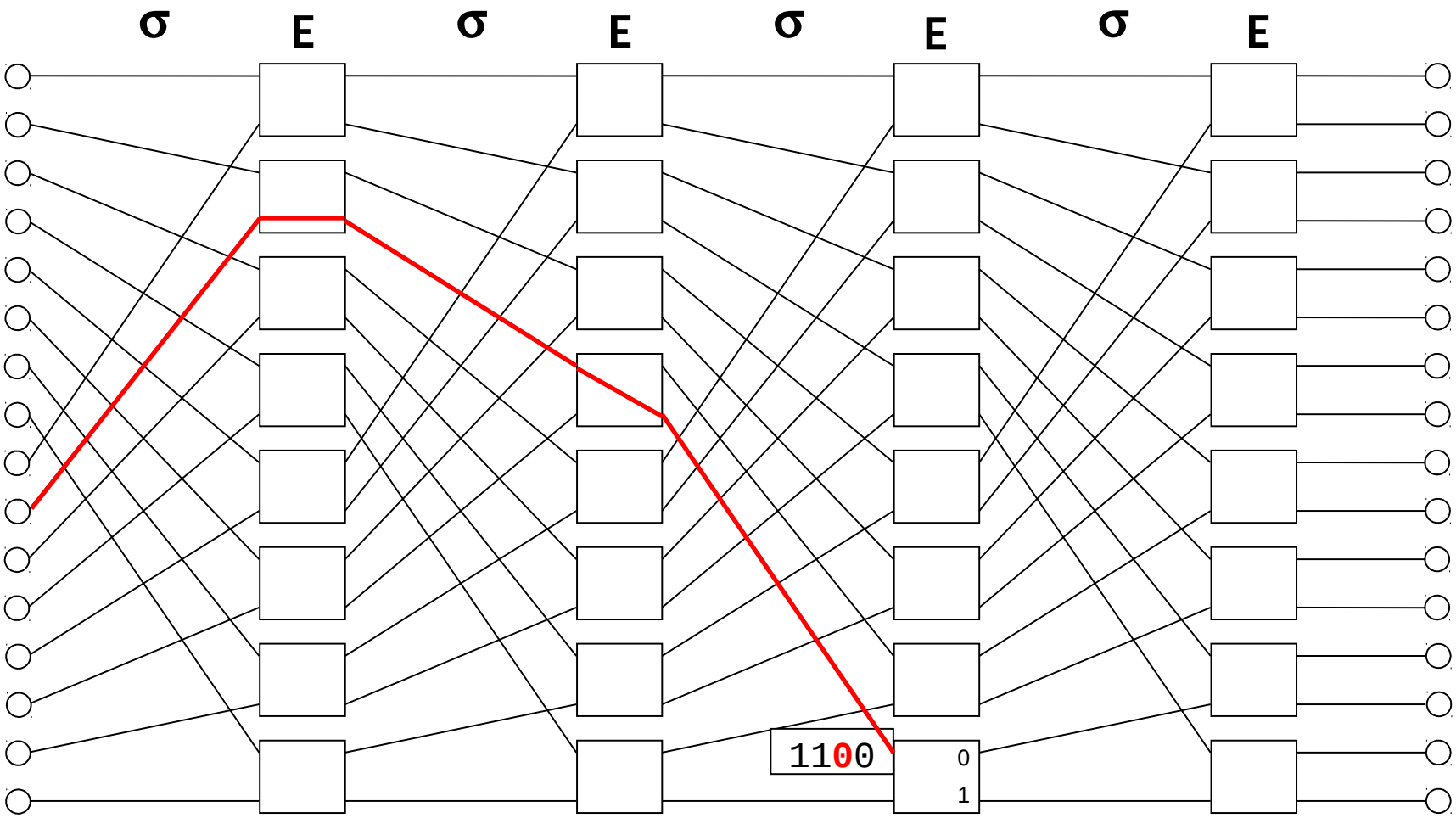
Omega network



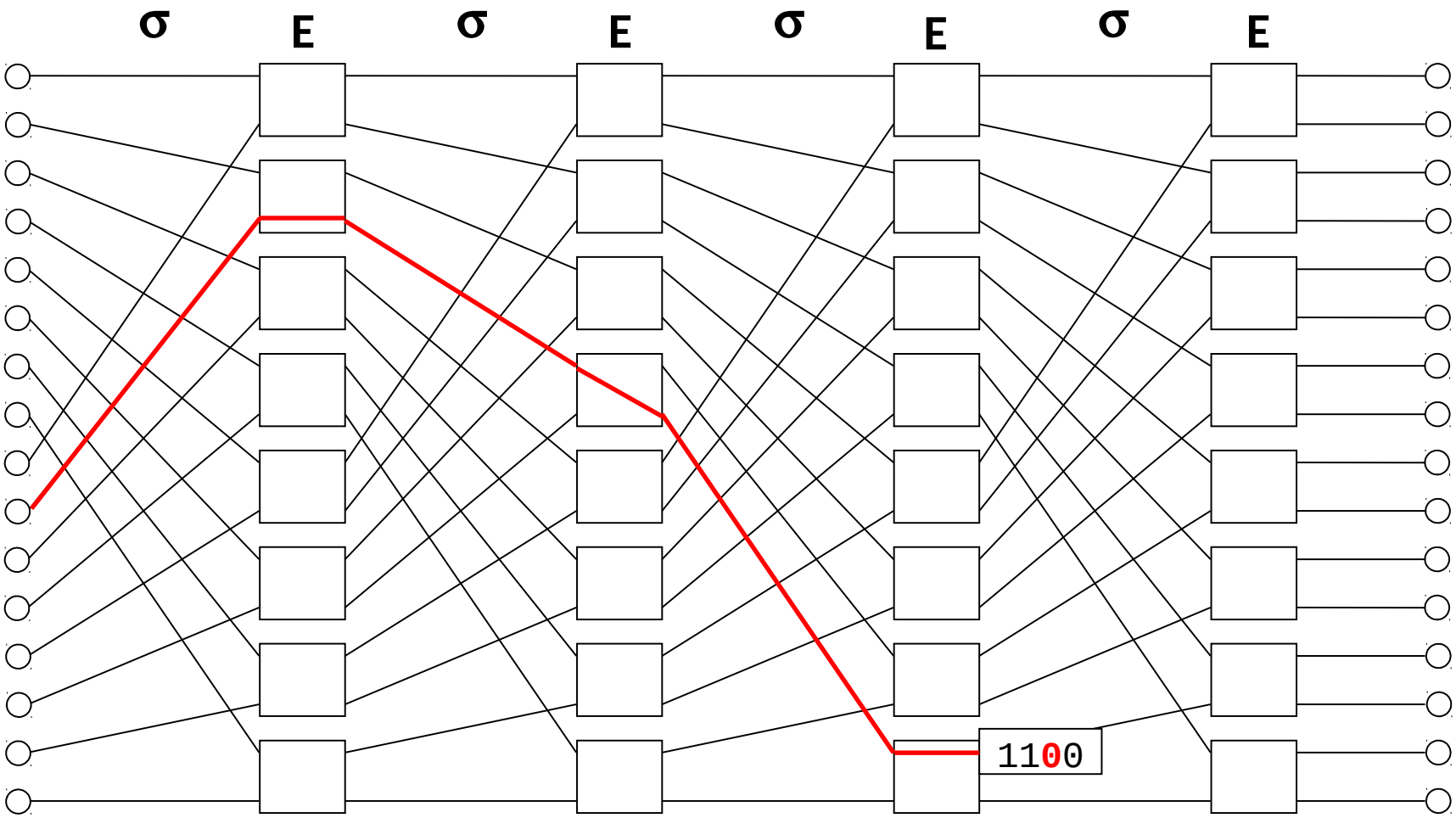
Omega network



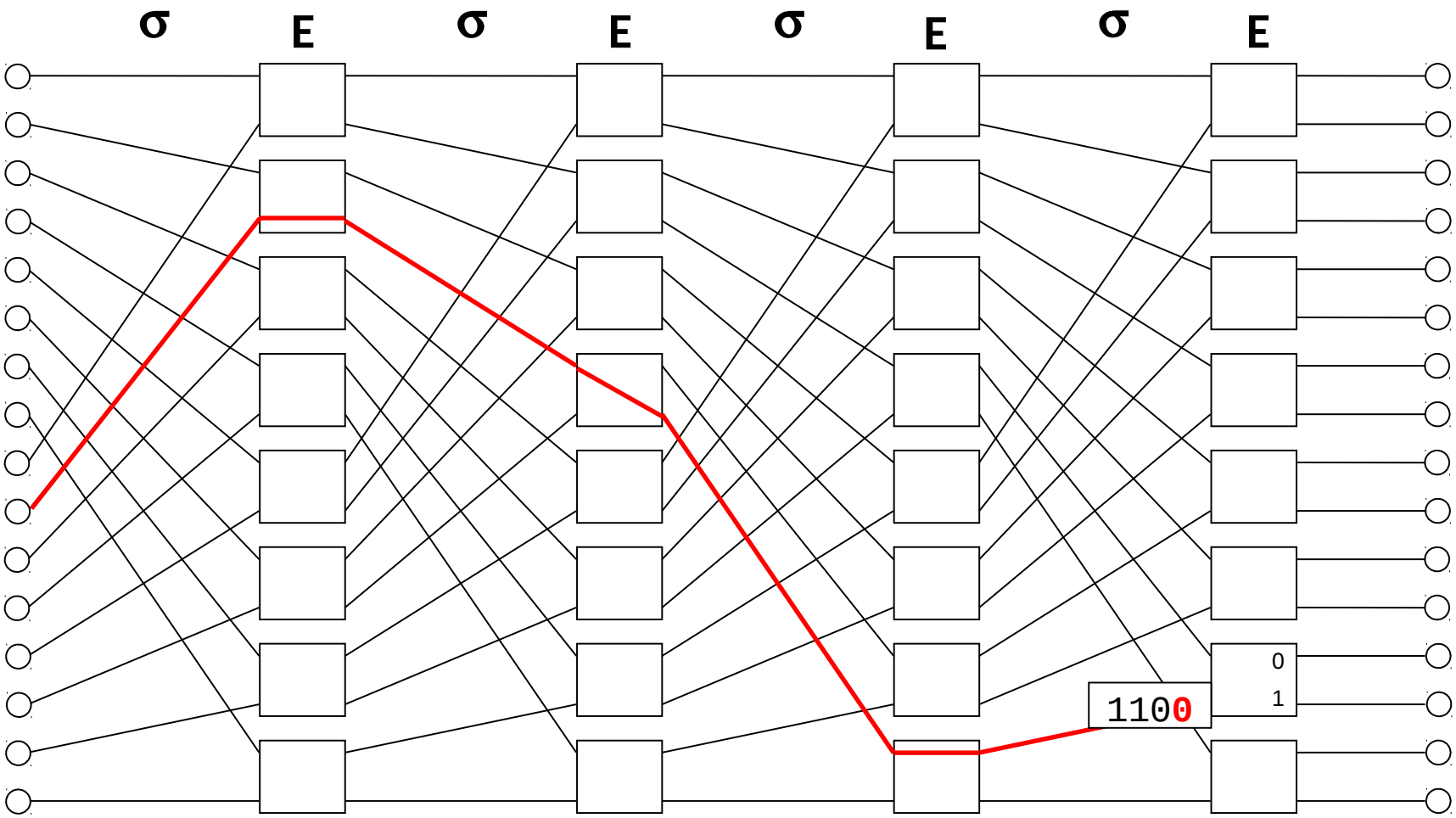
Omega network



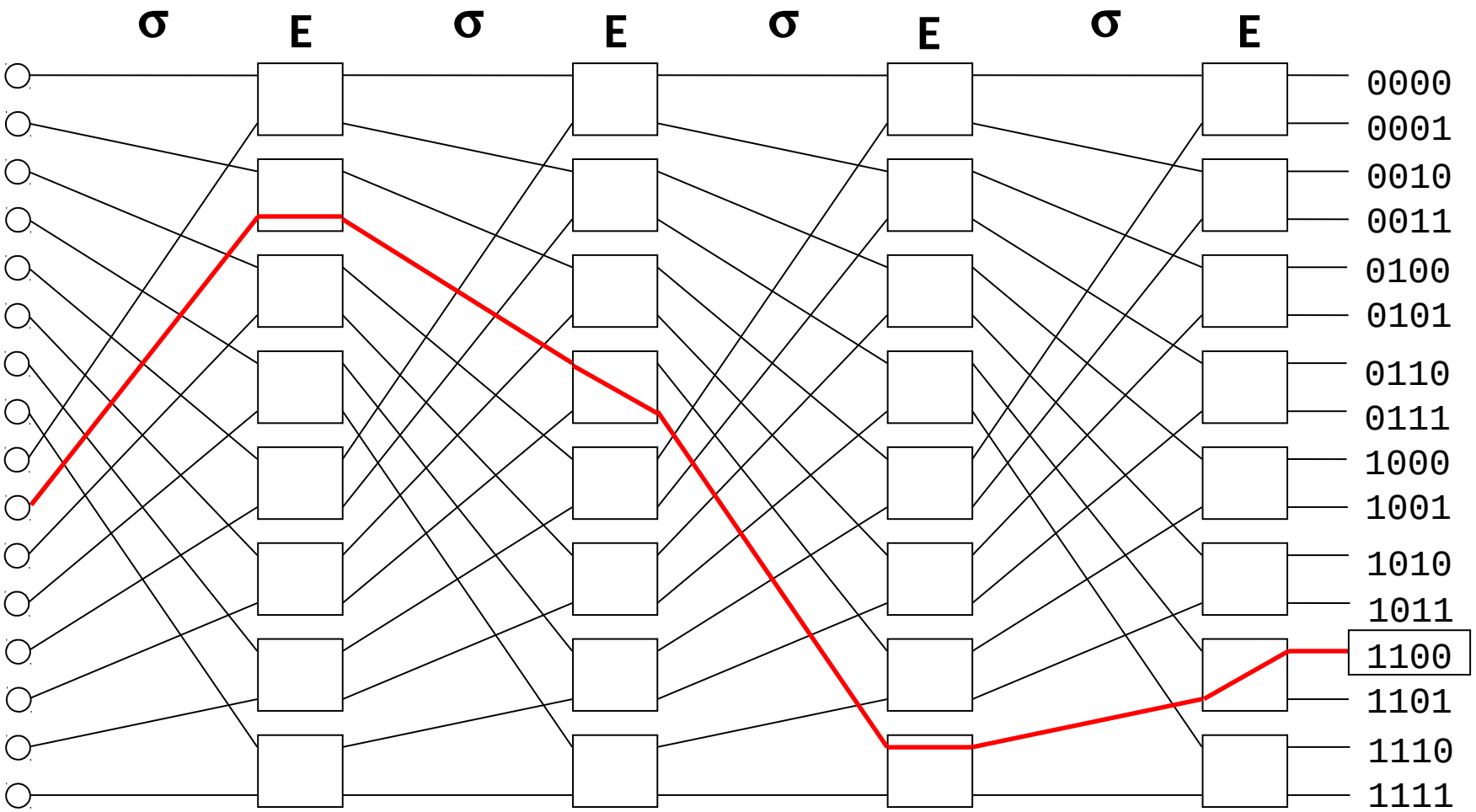
Omega network



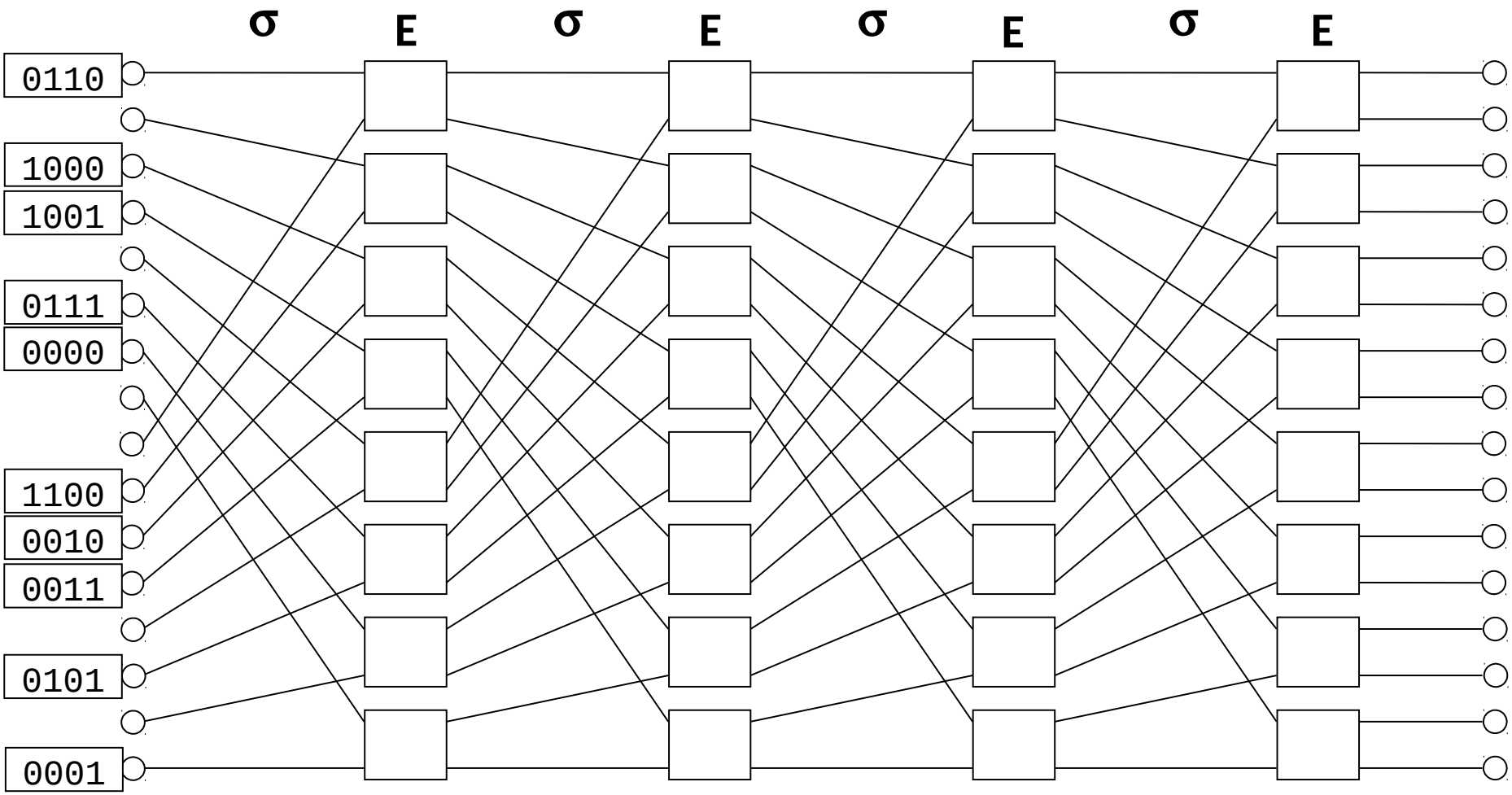
Omega network



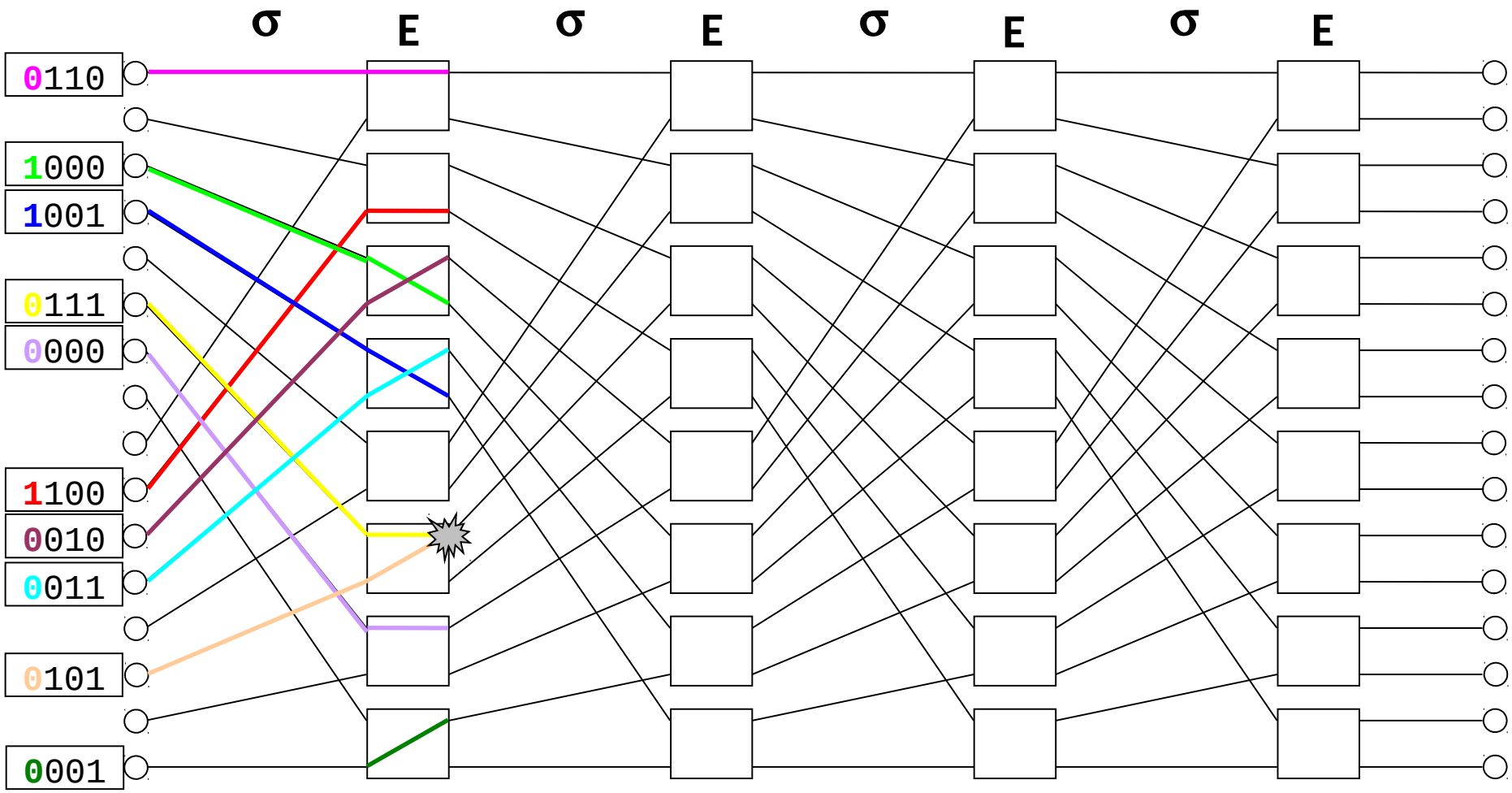
Omega network



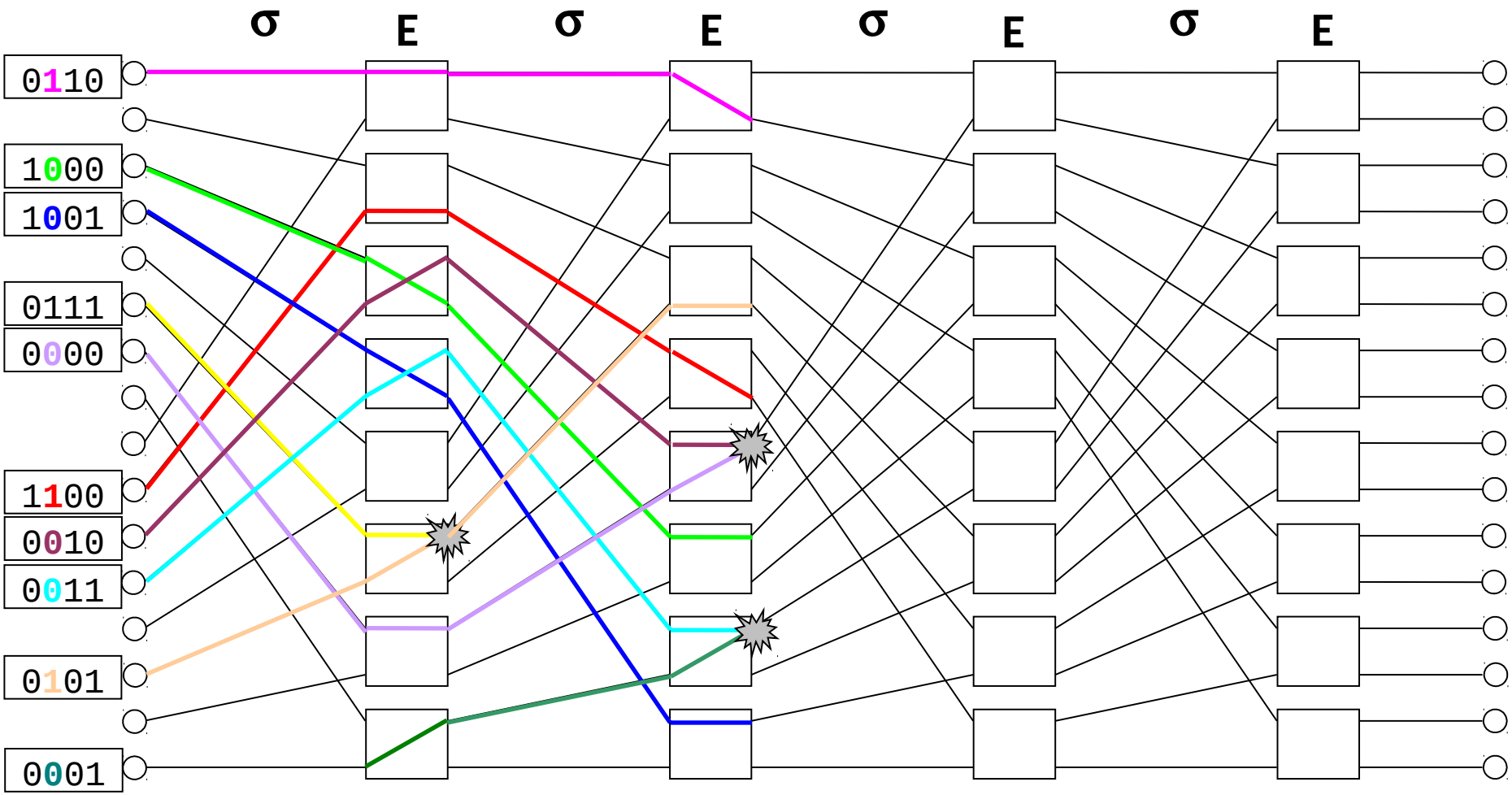
Omega network



Omega network

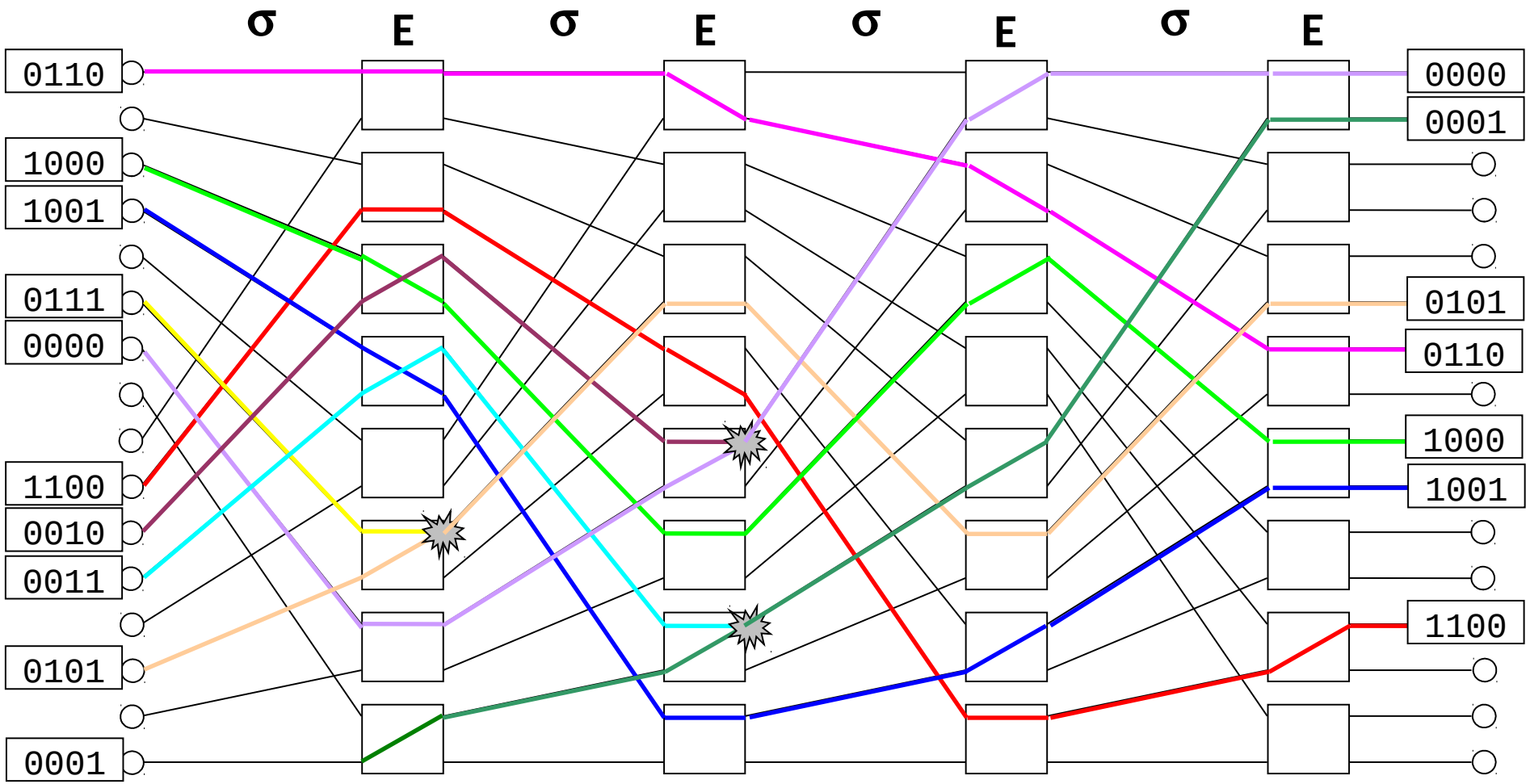


Omega network



Omega network

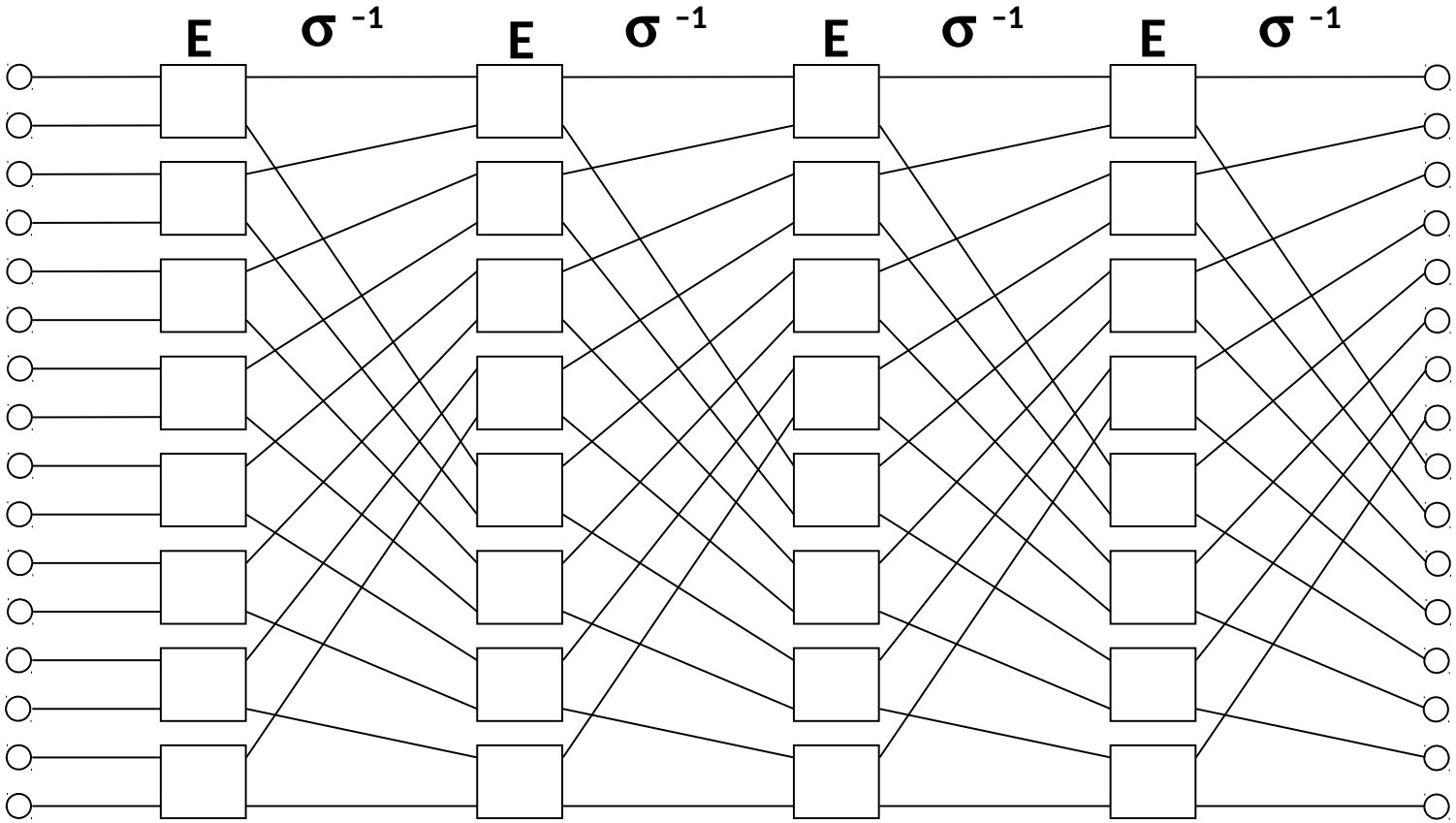
only 7 from 10 messages reached correct output port



Inverse omega network

$$\Omega_N^{-1} = (\Omega_N)^{-1} = ((\sigma E)^n)^{-1} = ((\sigma E)^{-1})^n = (E^{-1}\sigma^{-1})^n = (E\,\sigma^{-1})^n$$

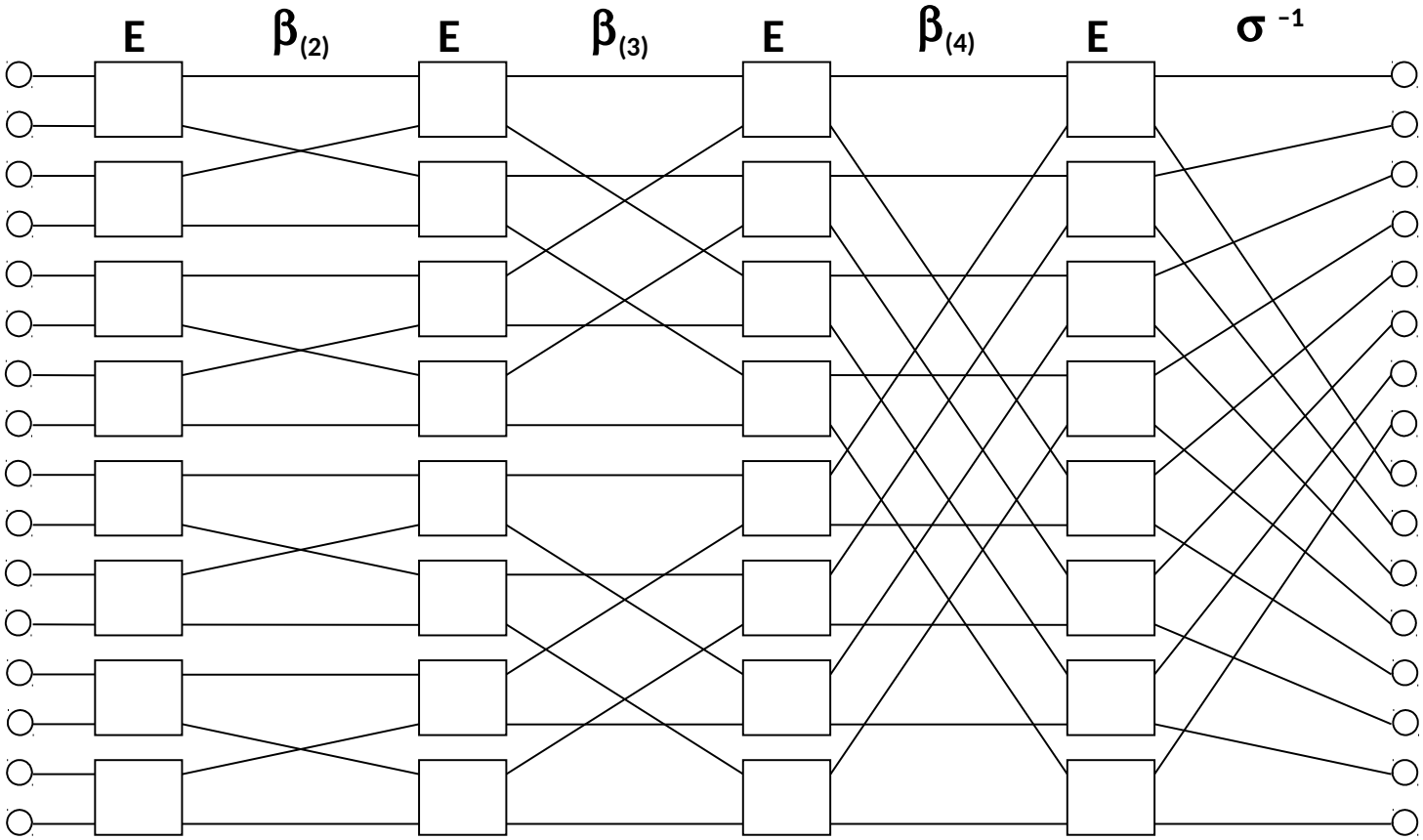
That is $\Omega_N^{-1} = (E\,\sigma^{-1})^n$



Indirect binary n-cube network

$$C_N = E\beta_{(2)}E\beta_{(3)}E\beta_{(4)} \ldots E\beta_{(n)}E\sigma^{-1}$$

Example for N=16, n=4 $\rightarrow C_{16} = E\beta_{(2)}\, E\beta_{(3)}\, E\beta_{(4)}\, E\sigma^{-1}$



Indirect binary n-cube network

Self-routing algorithm:

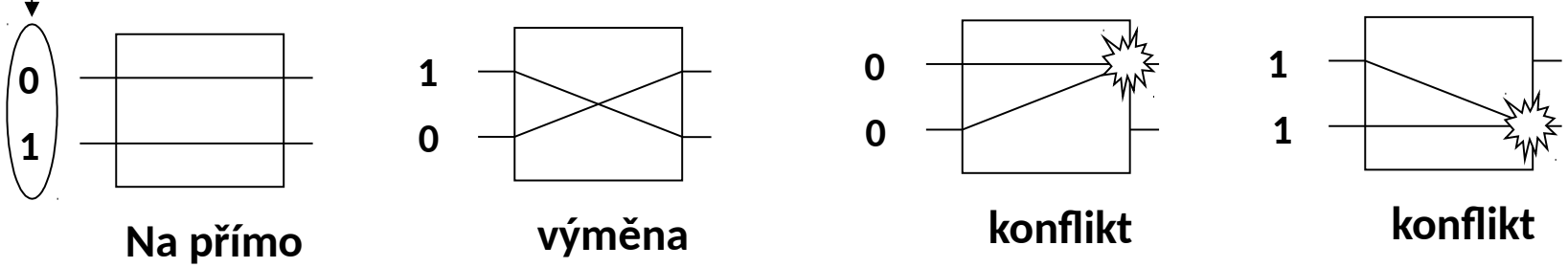
- Switch in k-th stage is reading k-th bit of routing label (starting from LSB towards MSB) and if the bit value is:
  - 0 then select upper output
  - 1 then select lower output

direction of bits read

tag = [d_n d_{n-1} … d_2 d_1]

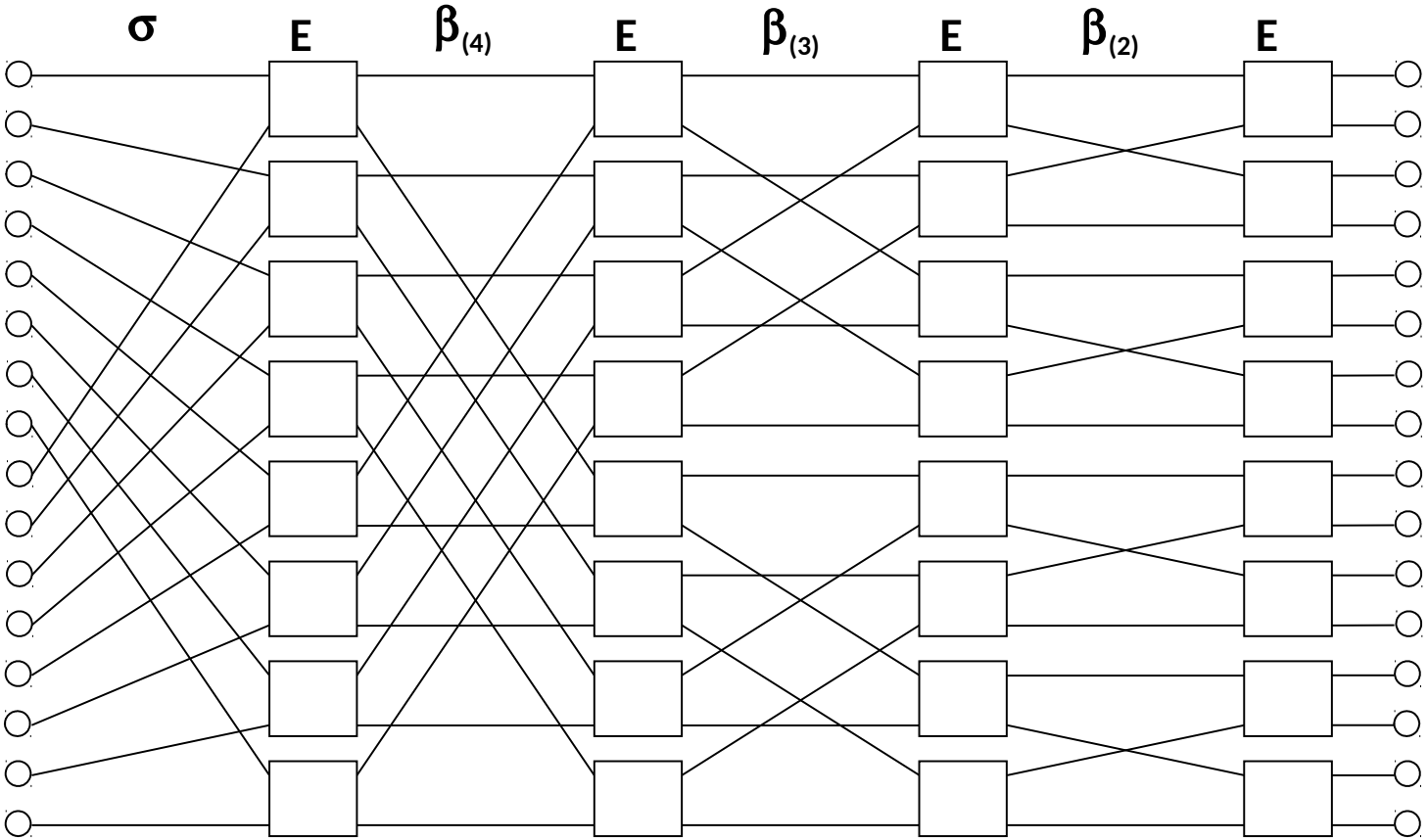
The first read bit

k-th bit in k-th stage



Indirect binary n-cube network

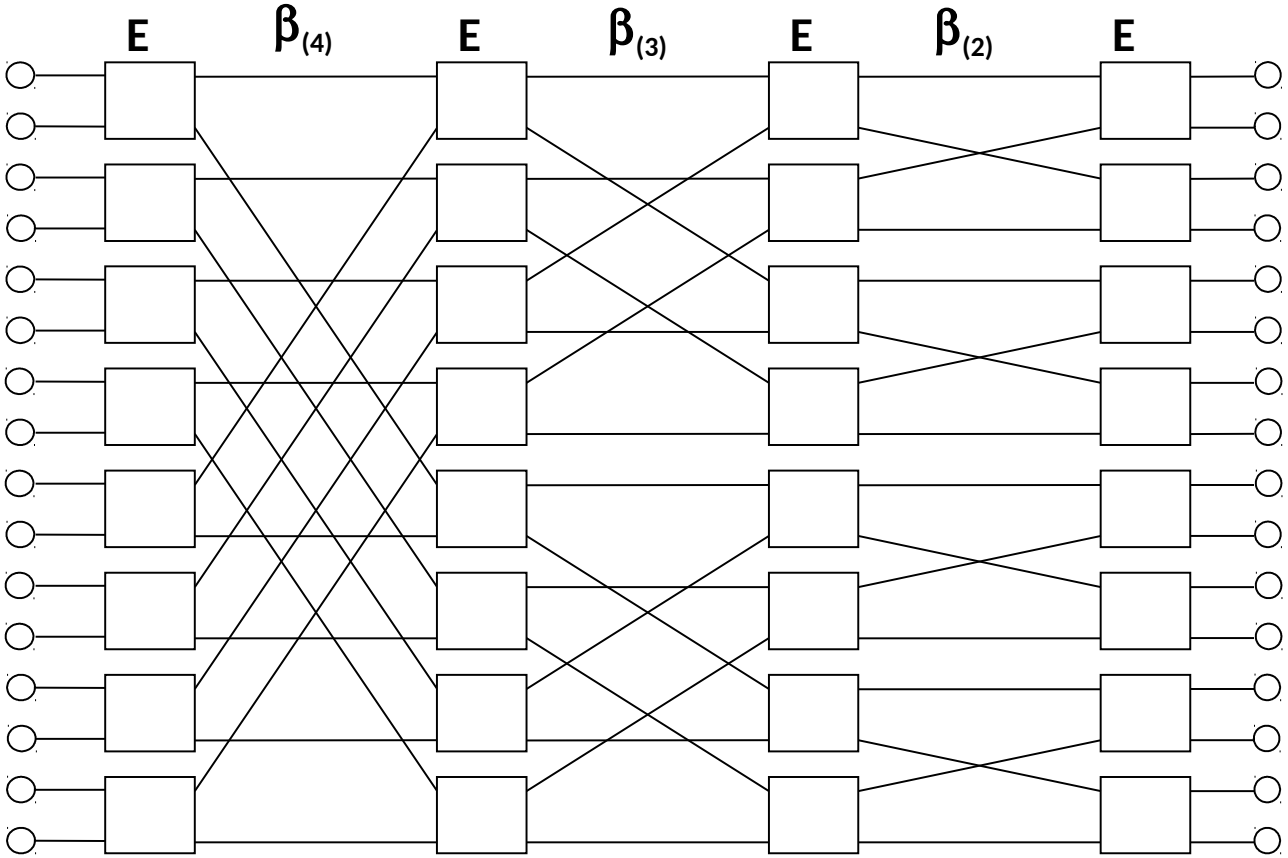
$$C_N^{-1} = (E\beta_{(2)}E\beta_{(3)}E\beta_{(4)} \ldots E\beta_{(n)}E\sigma^{-1})^{-1} = \sigma\, E\beta_{(n)} \ldots E\beta_{(4)}\, E\beta_{(3)}\, E\beta_{(2)}\, E$$



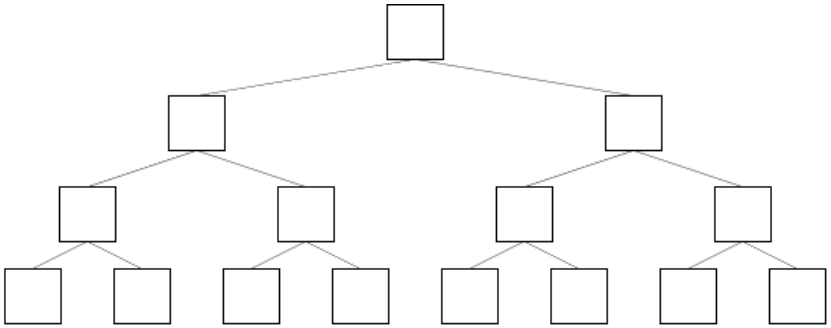
Butterfly network

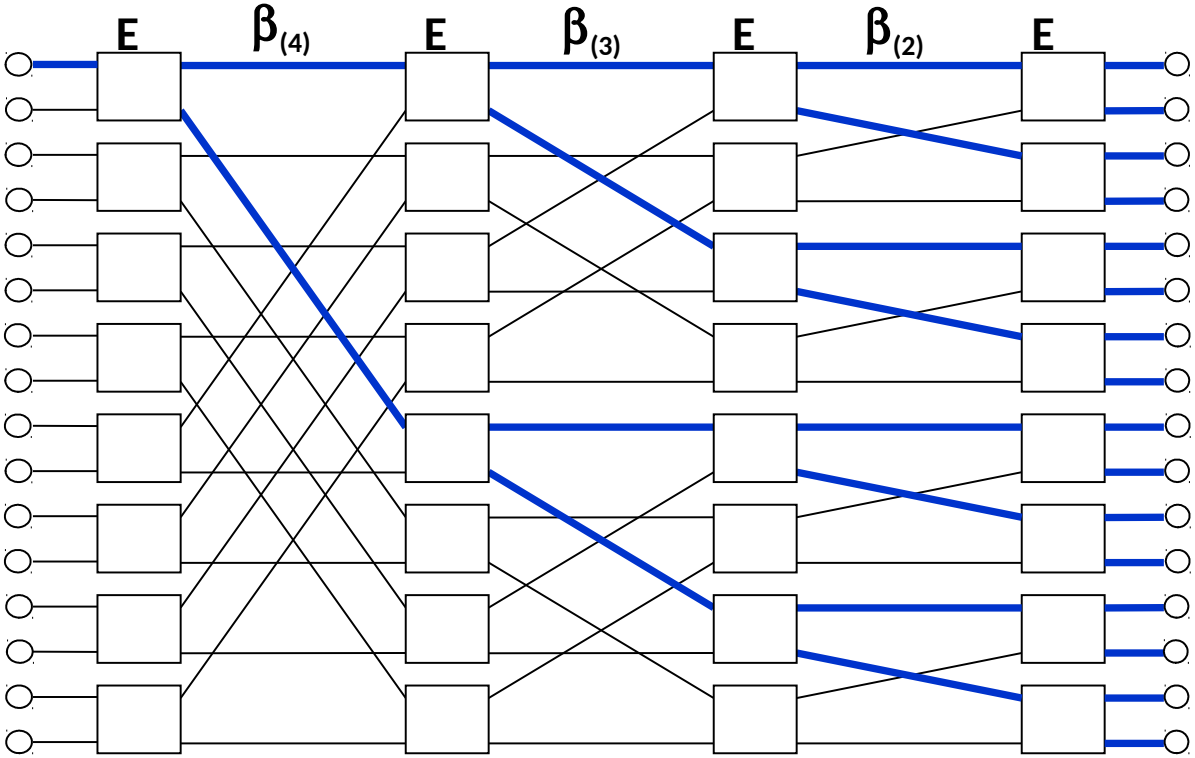
$$C_N^{-1} = (E\beta_{(2)}E\beta_{(3)}E\beta_{(4)} \ldots E\beta_{(n)}E\sigma^{-1})^{-1} = \cancel{\sigma}\, E\beta_{(n)} \ldots E\beta_{(4)}\, E\beta_{(3)}\, E\beta_{(2)}\, E$$

$$\mathbf{B = E\beta_{(n)} \ldots E\beta_{(4)}\, E\beta_{(3)}\, E\beta_{(2)}\, E}$$



Butterfly network

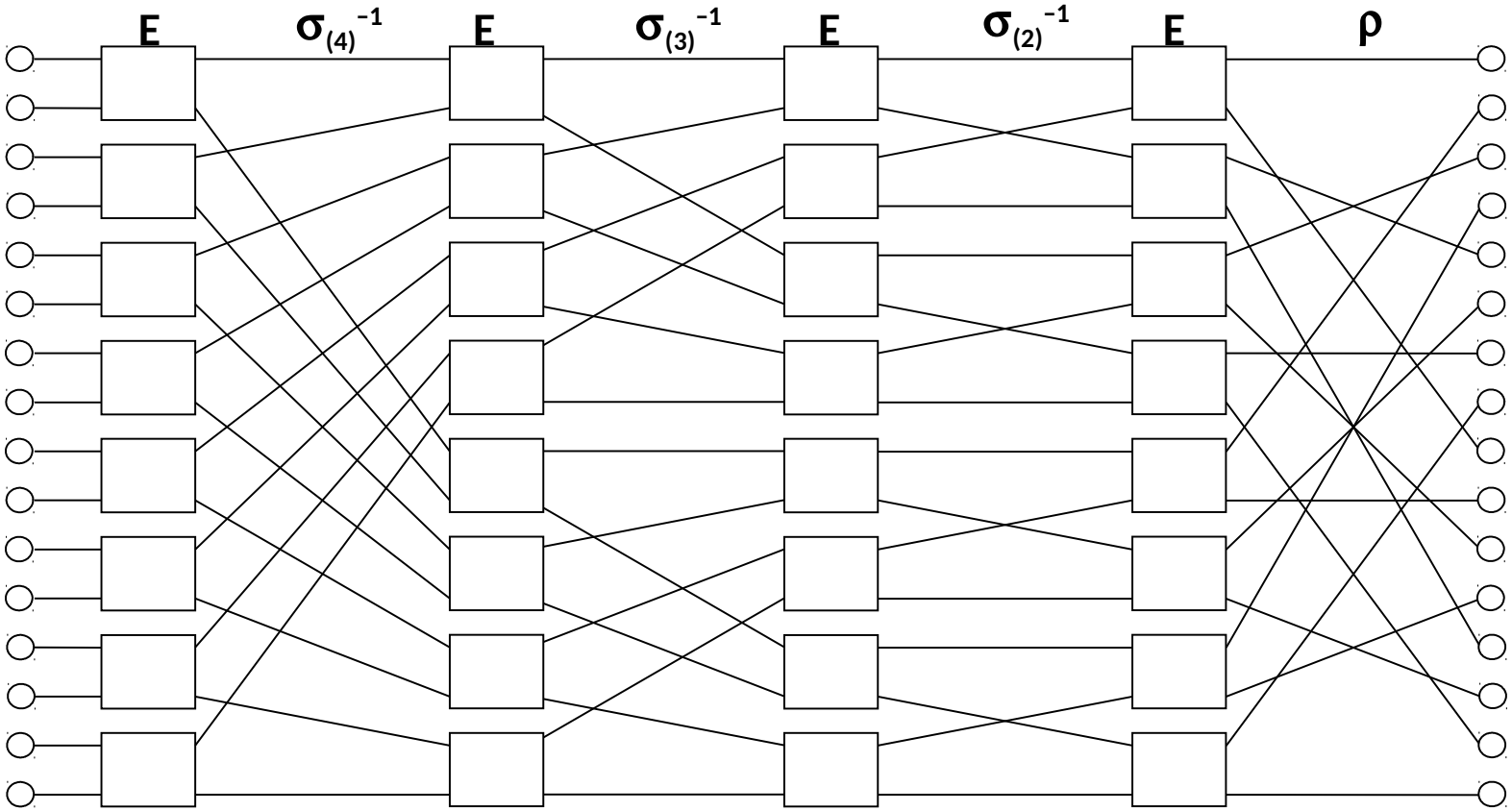




R-network

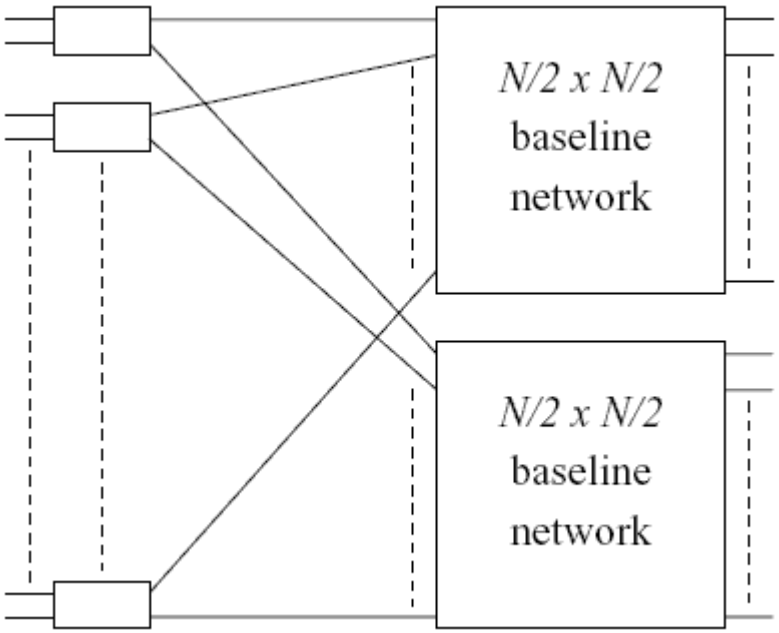
$$\mathbf{R_N = E\sigma_{(n)}^{-1} E\sigma_{(n-1)}^{-1} \ldots E\sigma_{(2)}^{-1} E\rho}$$

Example for N=16, n=4 → $R_{16} = E\sigma_{(4)}^{-1} E\sigma_{(3)}^{-1} E\sigma_{(2)}^{-1} E\rho$



Baseline network

- Baseline network topology is defined by recursive application of scheme shown in next picture:



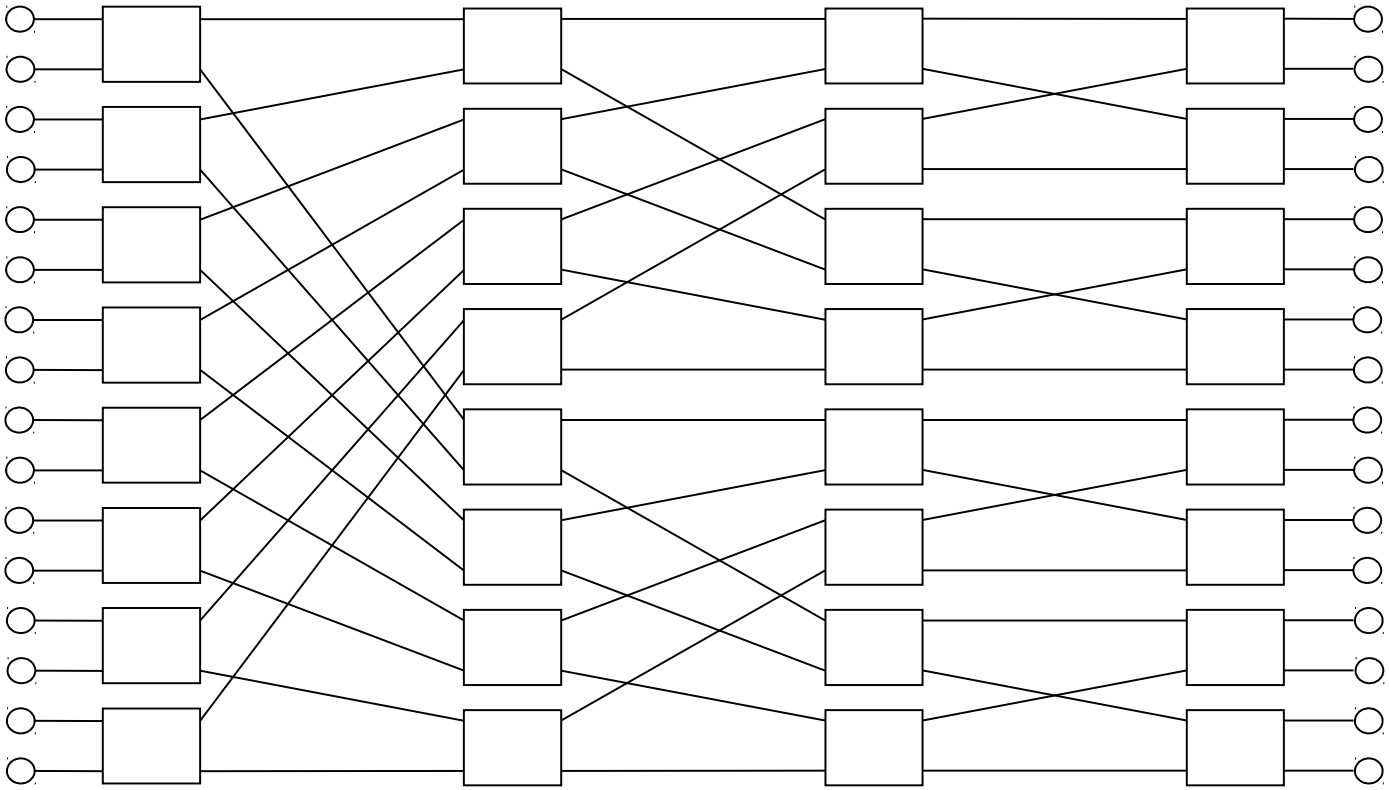
- The connection/permutation function between each stages is inverse shuffle **(σ^{-1}).**

Baseline network

- Recursive expansion is terminated when element size reaches 2x2 switch. After recursive expansion, the following specification is reached:

$$\mathbf{Baseline_N = E\sigma_{(n)}^{-1}\, E\sigma_{(n-1)}^{-1} \ldots E\sigma_{(2)}^{-1}\, E}, \quad \text{where } \mathbf{n=\log_2 N}$$

- Example for N = 16, n = 4:

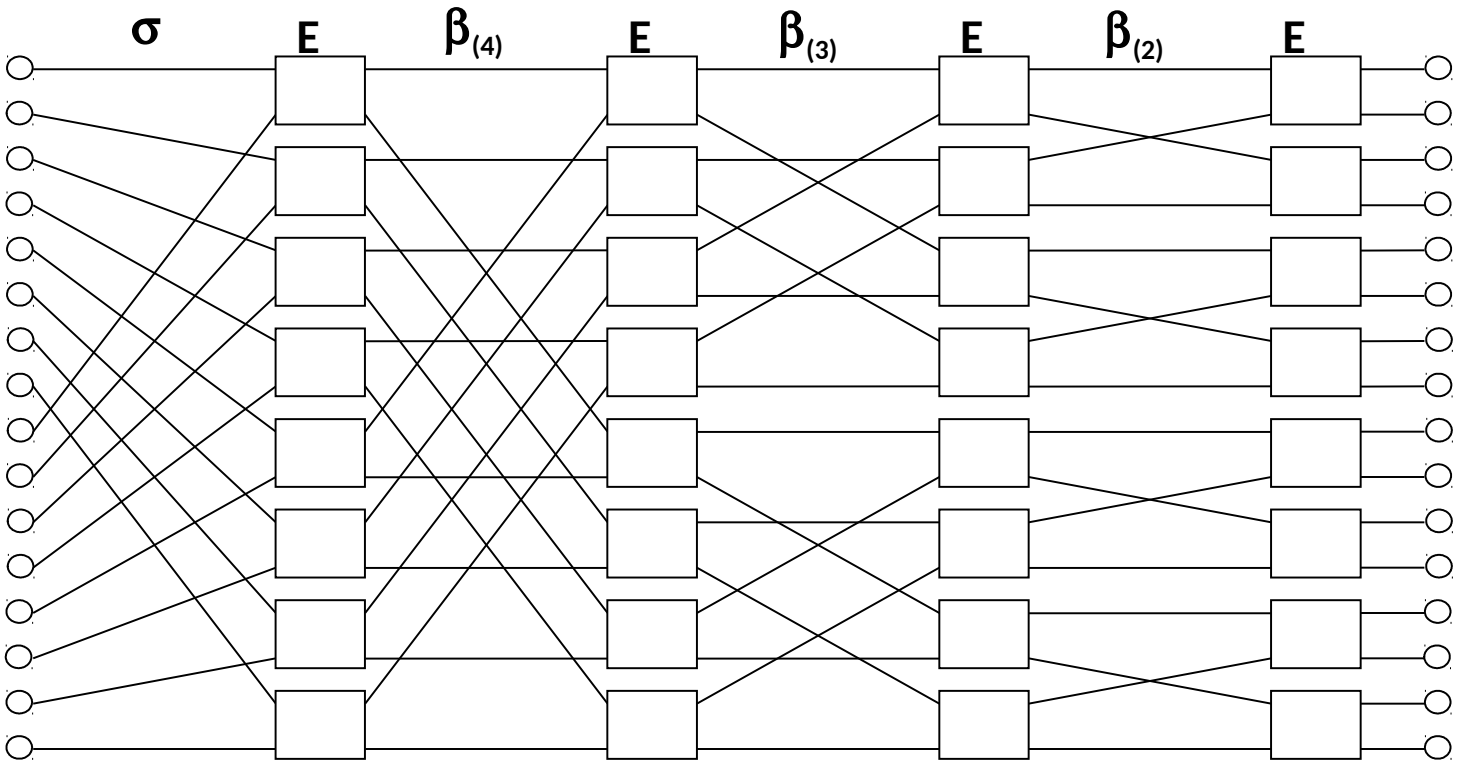


Banyan network

- Banyan network (sometimes described as generalized cube network) is a network defined by the formula:

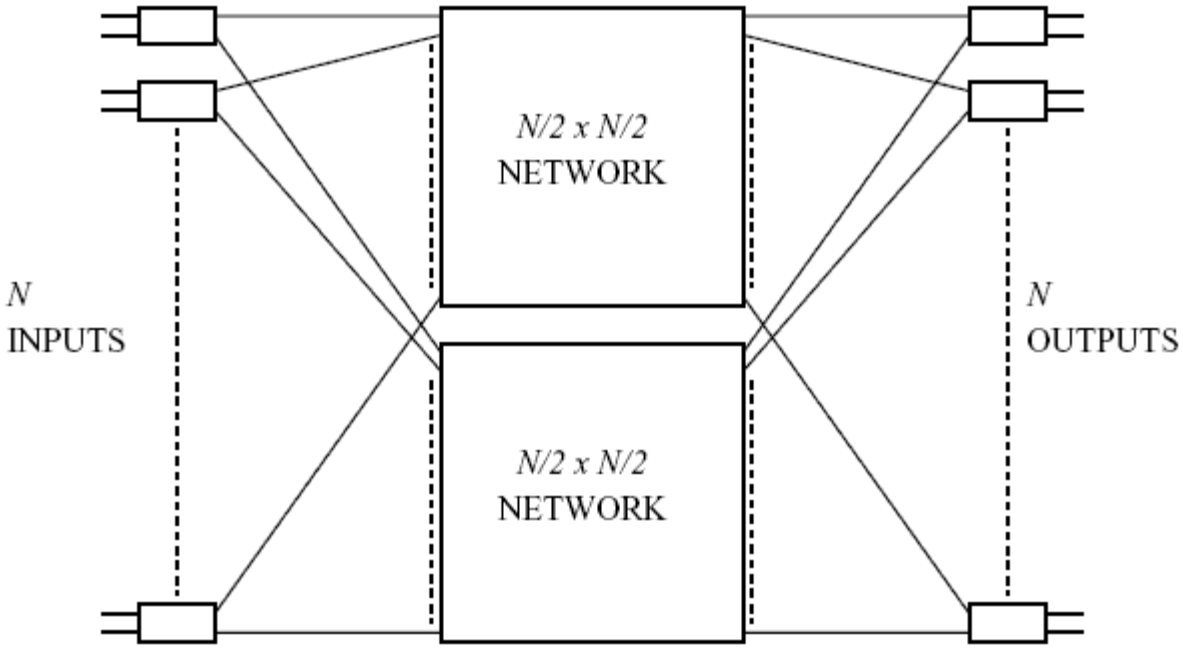
$$\mathbf{Banyan_N = \sigma\, E\beta_{(n)} \ldots E\beta_{(4)}\, E\beta_{(3)}\, E\beta_{(2)}\, E,} \quad \text{where } \mathbf{n=\log_2 N}$$

and so it is equivalent to inverse binary n-cube network.



Beneš network

- Recursive definition of Beneš network:

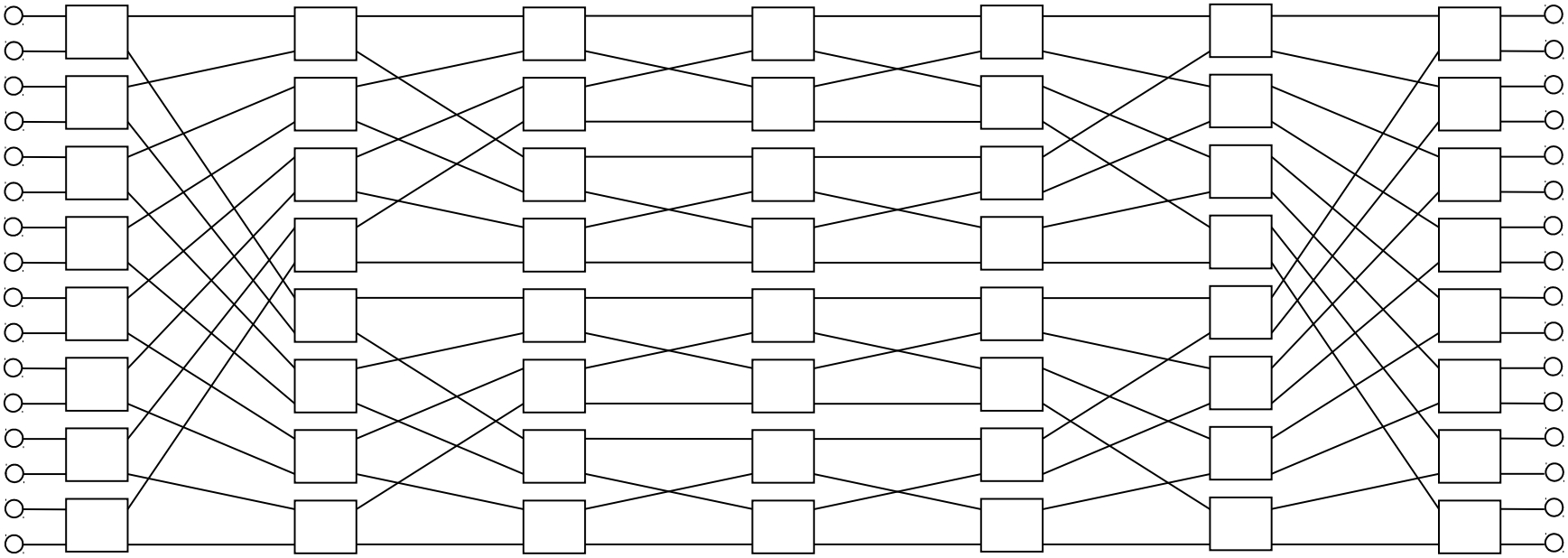


- When recursive expansion is finished then network is build from $2\log_2 N–1$ stages and in each stage are N/2 switching elements. Beneš network belongs **between rearrangeably non-blocking networks,** and that is why it requires an algorithm for the reconfiguration process – central control. There exists self-routing algorithms which suppress partially blocking in the network.

Beneš network

$$\text{Beneš}_N = E\sigma_{(k)}^{-1}\, E\sigma_{(k-1)}^{-1}\, E \ldots \sigma_{(2)}^{-1}\; E\; \sigma_{(2)} \ldots E\sigma_{(k-1)}\, E\sigma_{(k)}\, E,$$

- Example for N = 16, stages: $n = 2\log_2 N-1 = 7$:



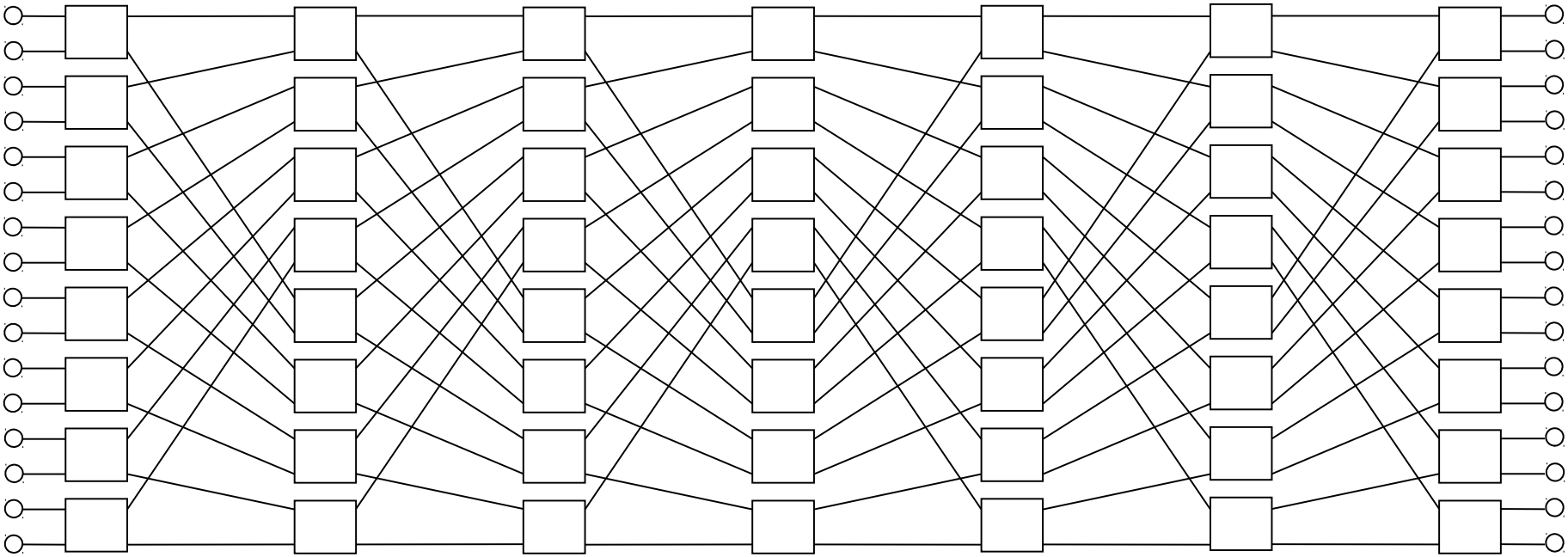
- Beneš network is equivalent to serial connection of two baseline networks (baseline network and inverse baseline network). That is why Beneš networks are sometimes called "serial baseline networks".

Beneš network

Alternative definition:

$$\textbf{Beneš}_N = E\sigma^{-1} E\sigma^{-1} E \ldots \sigma^{-1} E \sigma \ldots E \sigma E \sigma E = (E\sigma^{-1})^{k-1} E (\sigma E)^{k-1},$$

- where $k=\log_2 N$



Beneš network

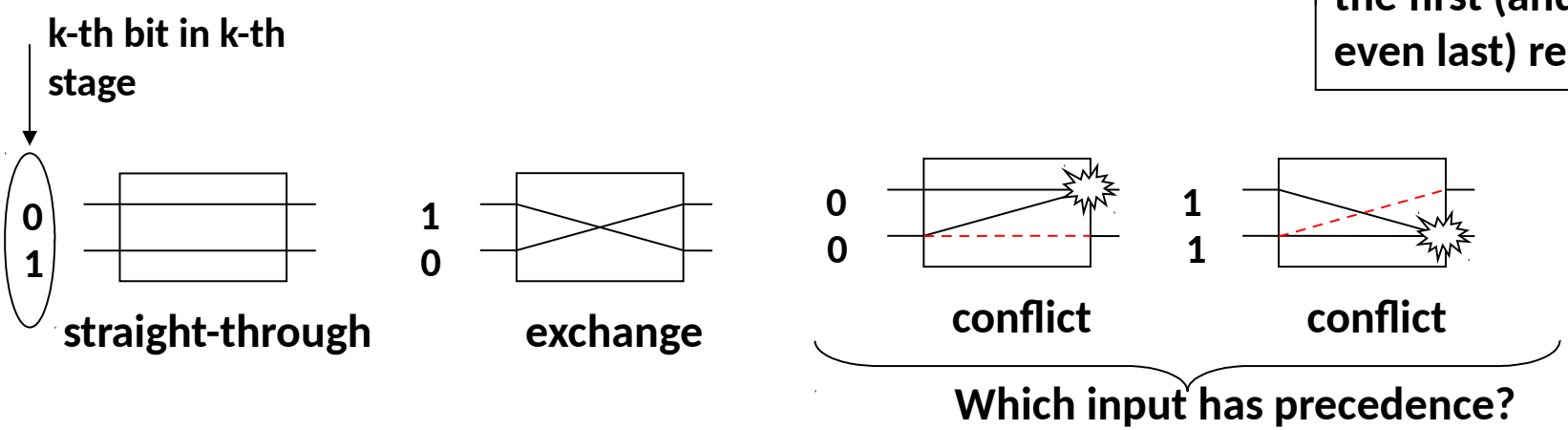
Self-routing algorithm:

- Switch in i-th, resp. j-th stage ($1 \le i \le k$, resp. $k < j \le 2k-1$, where $k = \log_2 N$) reads i-th bit, resp. (2k-j)-th bit of routing label (starting from LSB towards MSB) if value is equal to:
  - 0 then select upper output
  - 1 then selects lower output

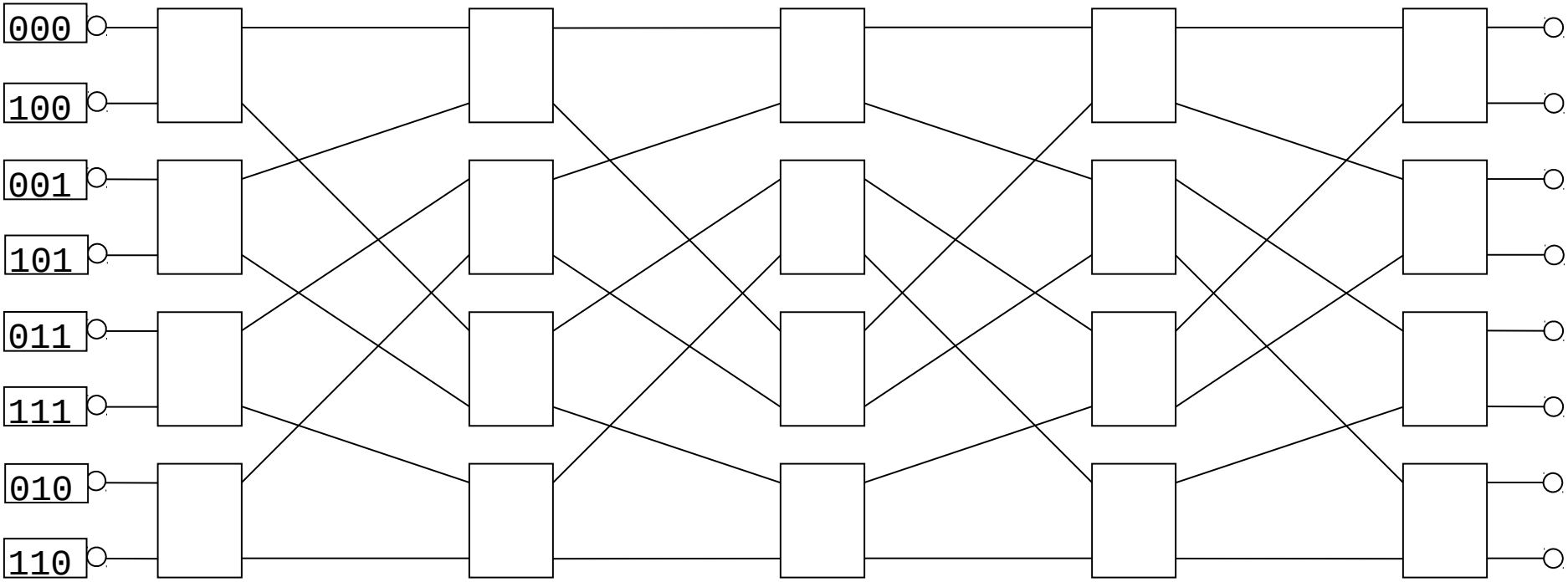
direction of bits read

$$\text{tag} = [\, d_n \;\; d_{n-1} \;\; \ldots \; d_2 \;\; d_1 \,]$$

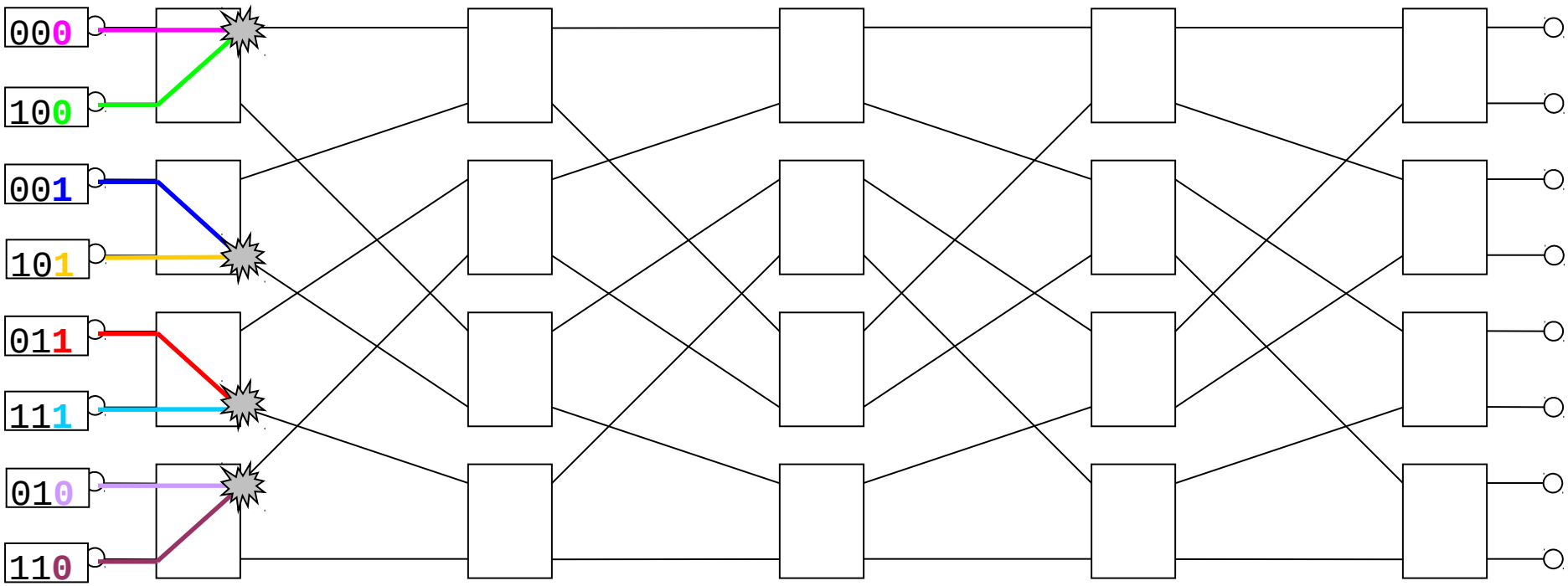
the first (and even last) read bit



Beneš network



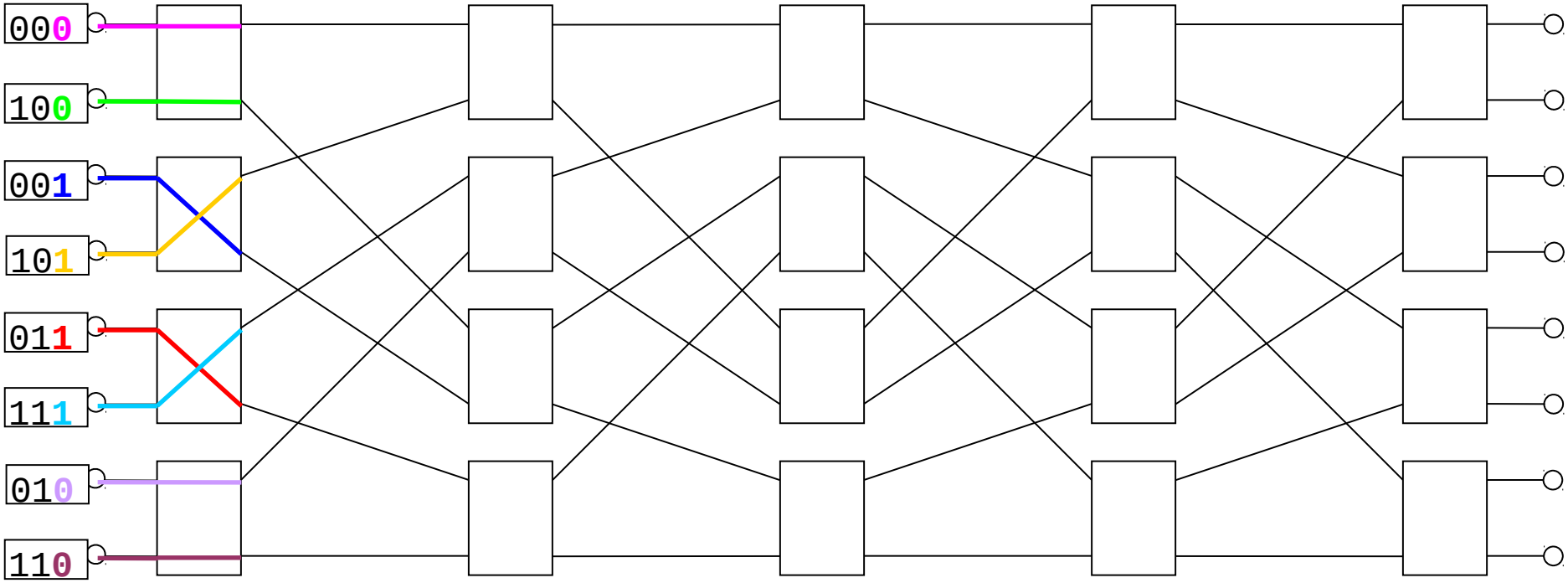
Beneš network



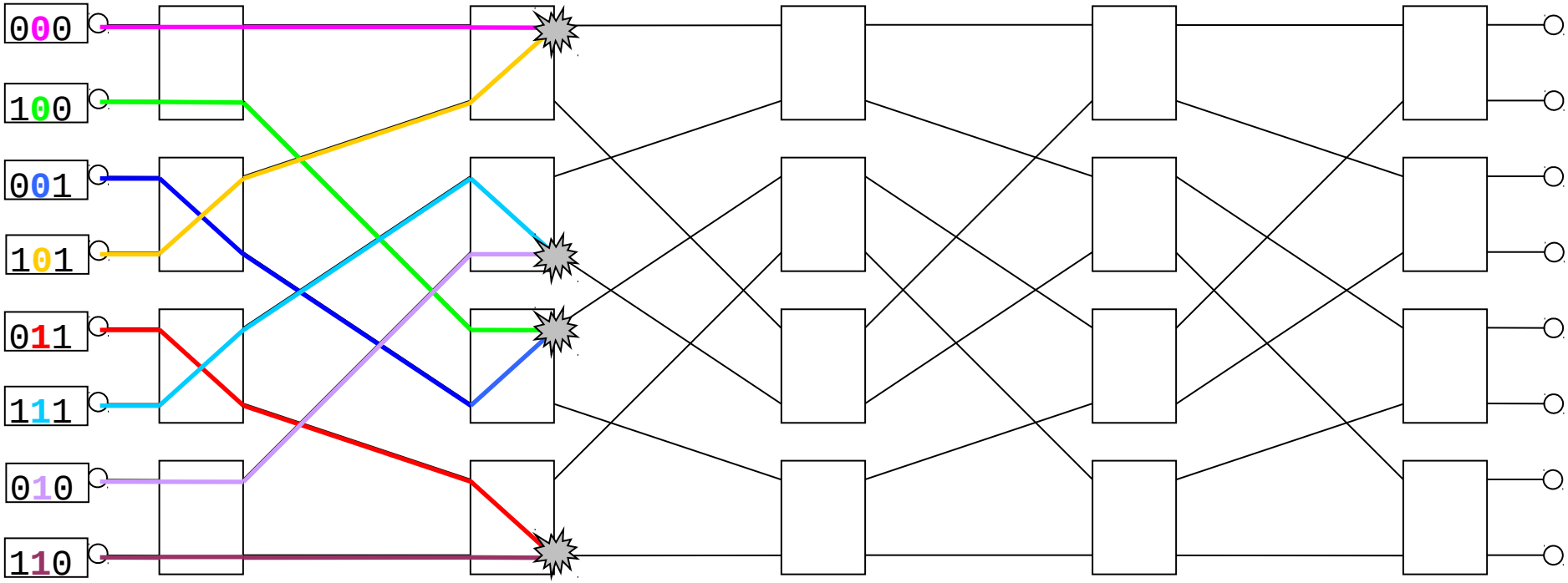
What to do in the conflict case ???

Example rule: Precedence for input which has lower value of routing label …

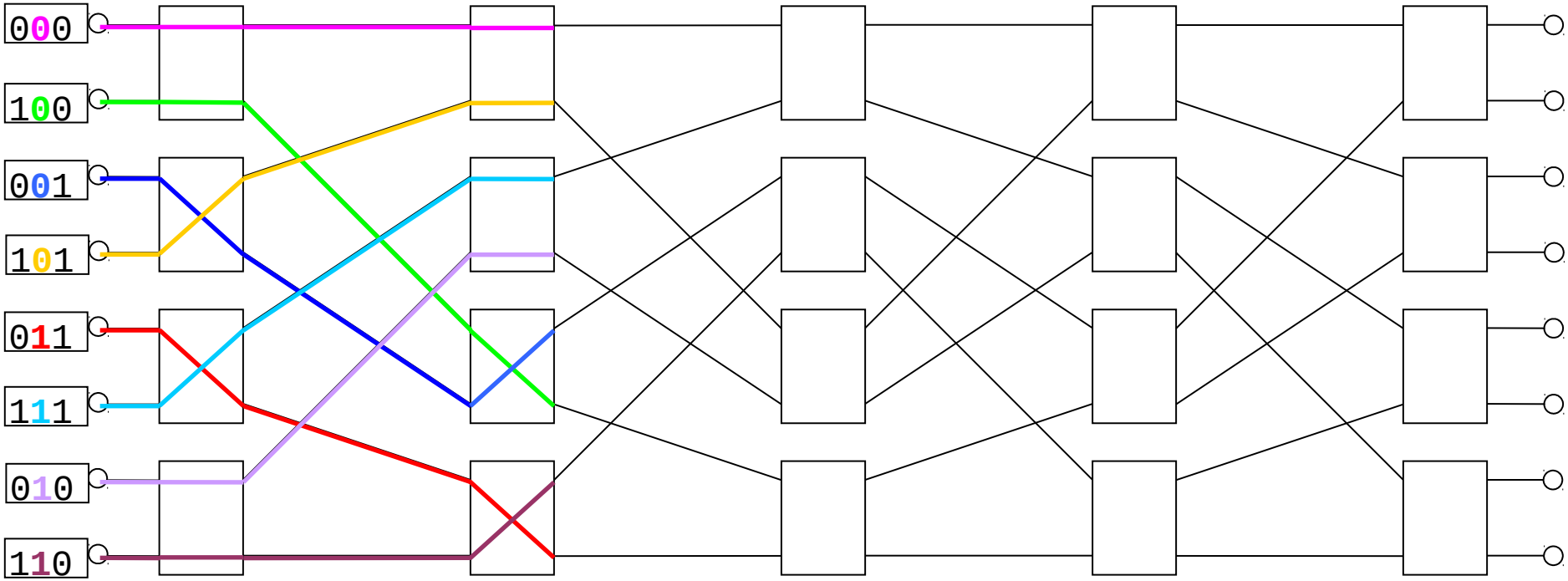
Beneš network



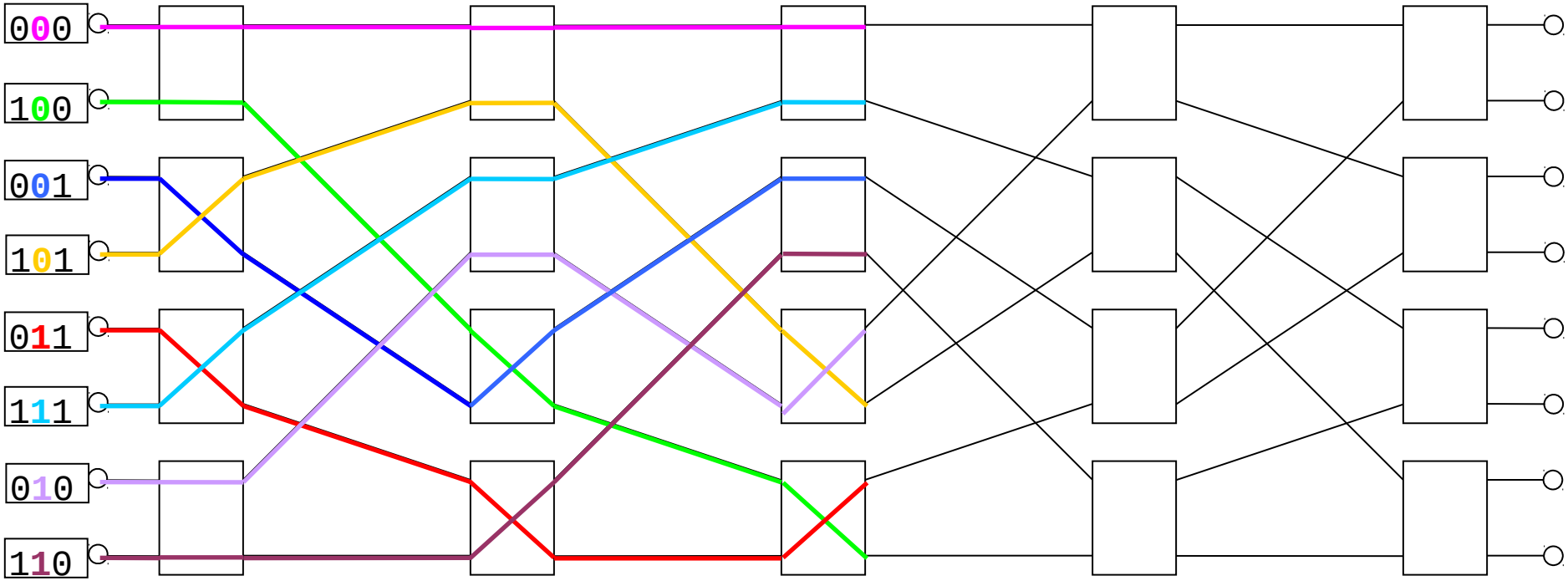
Beneš network



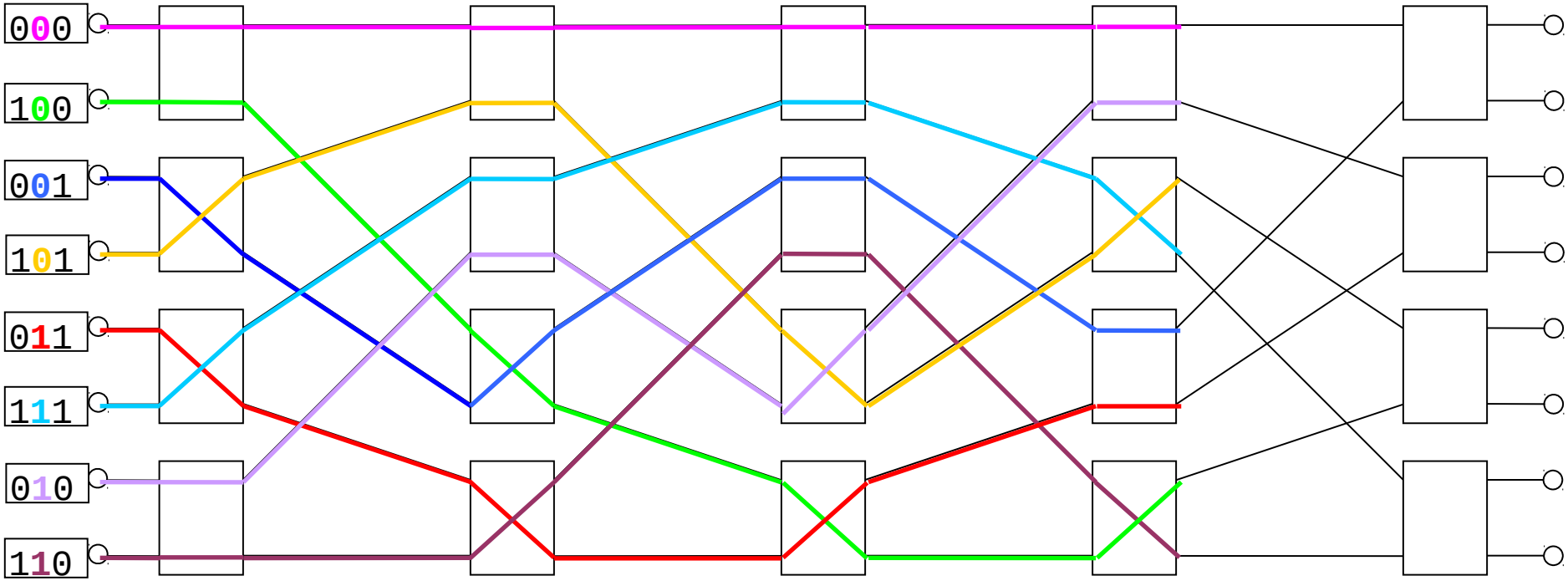
Beneš network



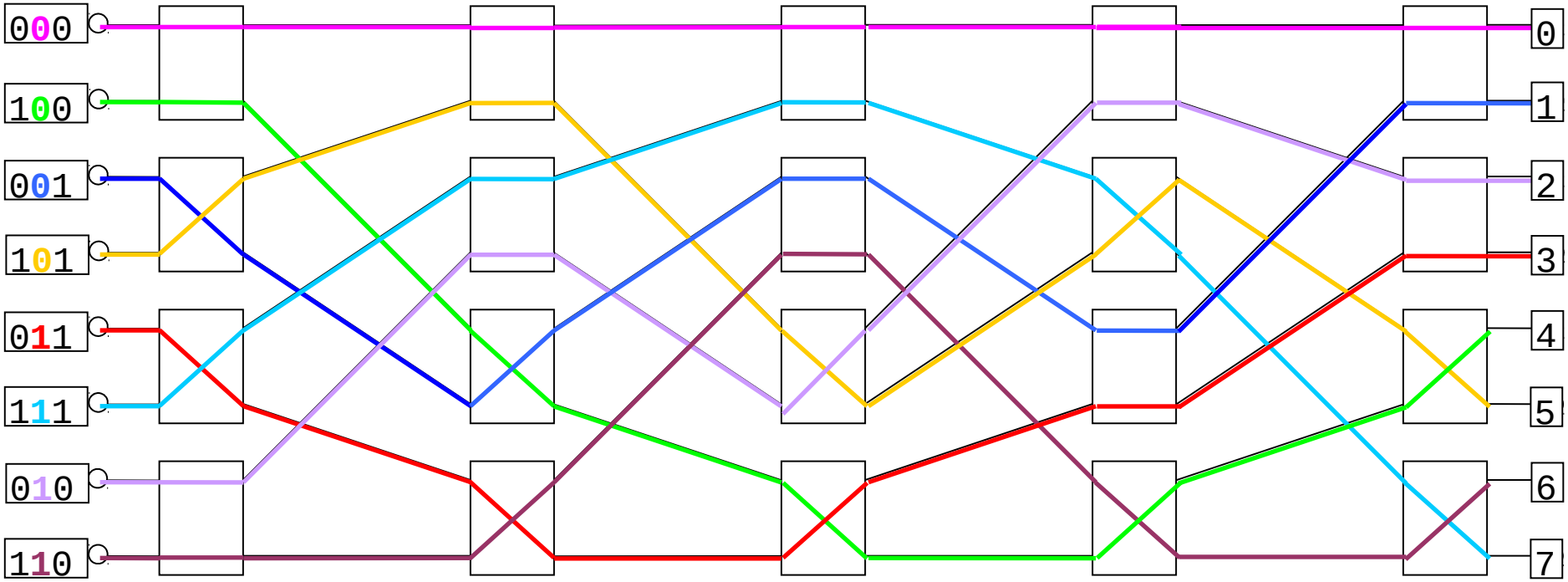
Beneš network



Beneš network

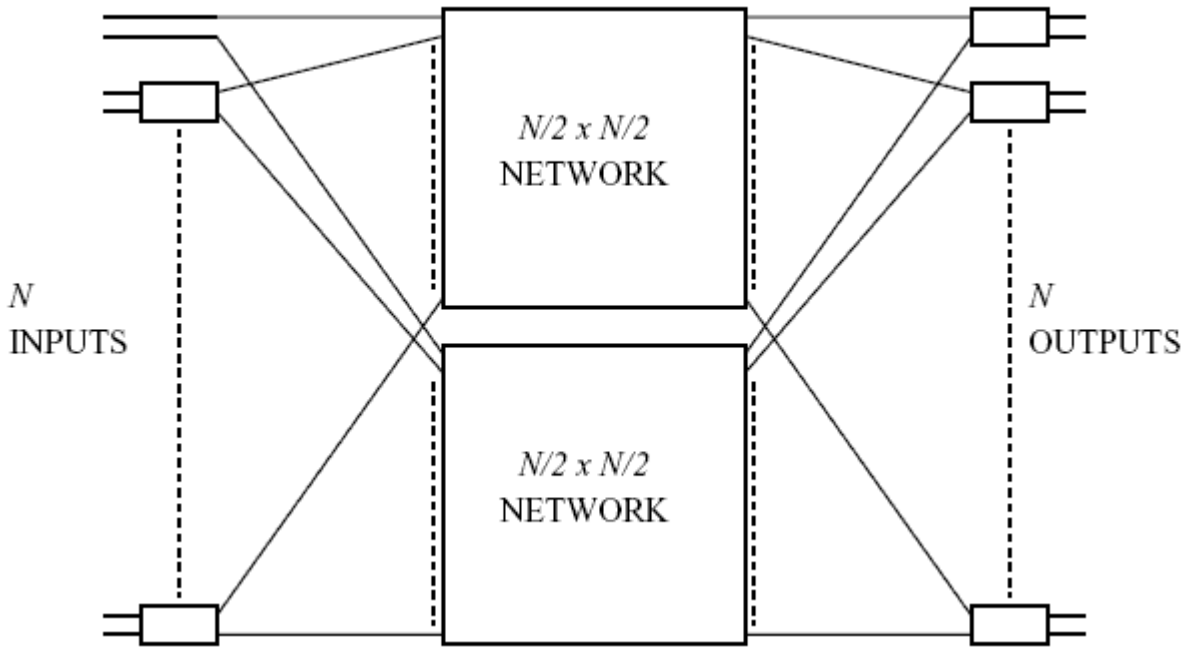


Beneš network



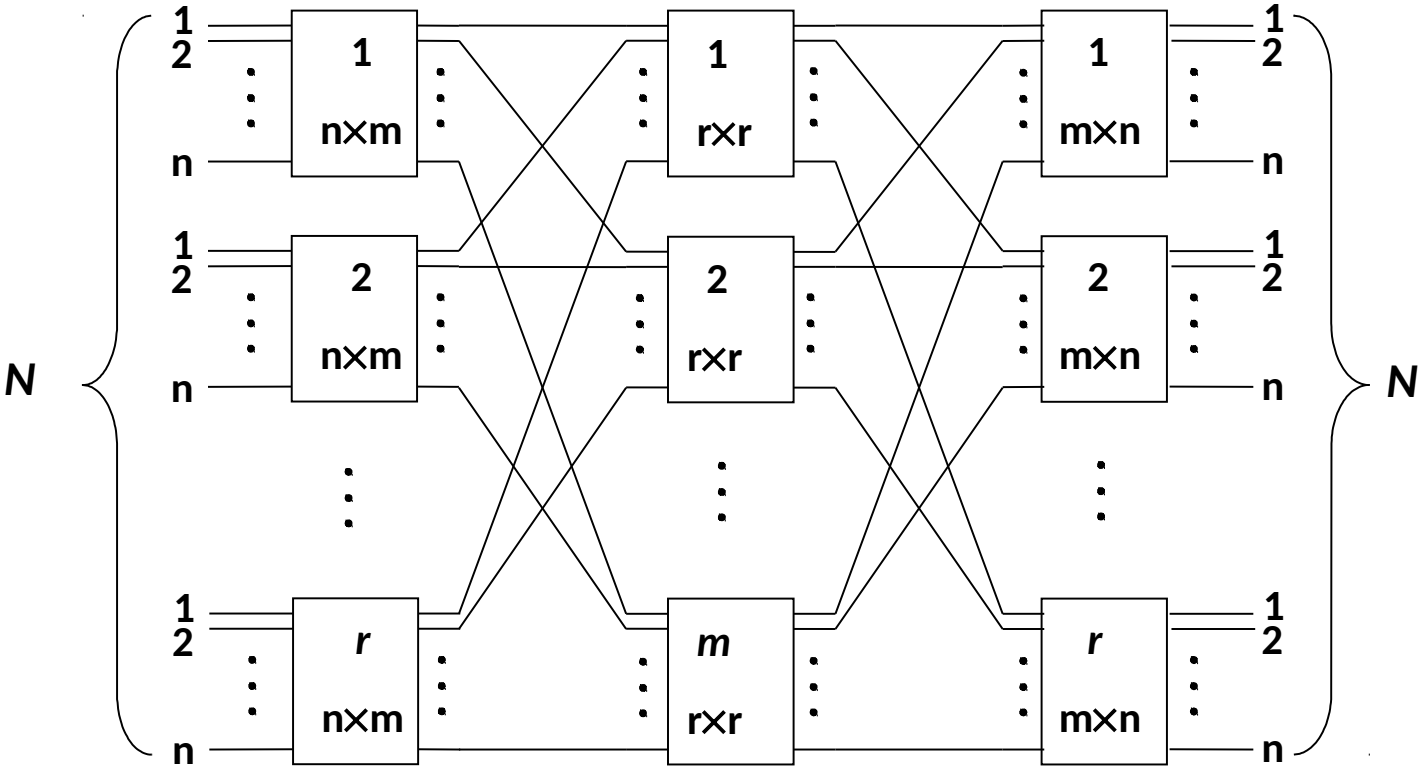
Waksman network

- When modified version of Slepian-Duguid teorem is used then Beneš network can be changed to Waksman network by leaving out one input or output switch element. Waksman network requires less switching elements than Beneš network.



Clos network

- Symmetric 3-stages Clos network with N input and N output ports uses r switching modules (crossbar switches) of size n×m in the first (ingress) stage, m switching modules of size r×r in middle stage and r witching modules of size m×n in the third (egress/output) stage. Such 3 stages network is symbolically denoted as C(m, n, r).

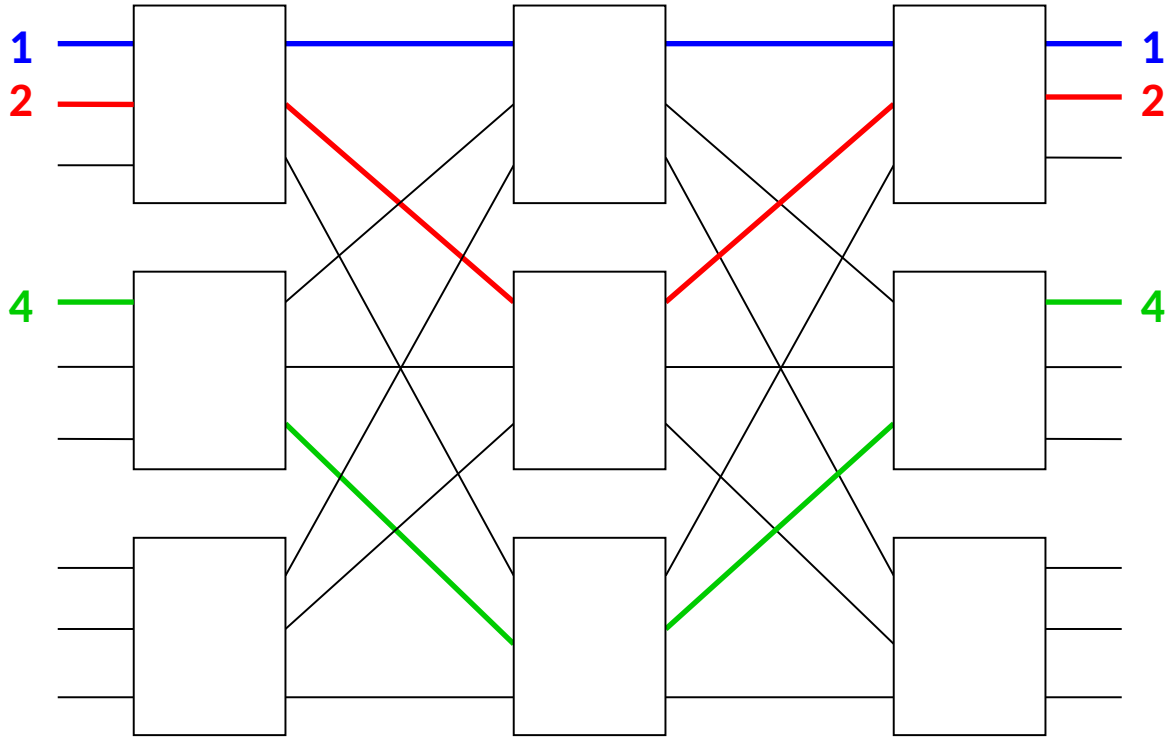


Clos network

Is Clos network for m = n non-blocking?

- **Example of connection in the network:**

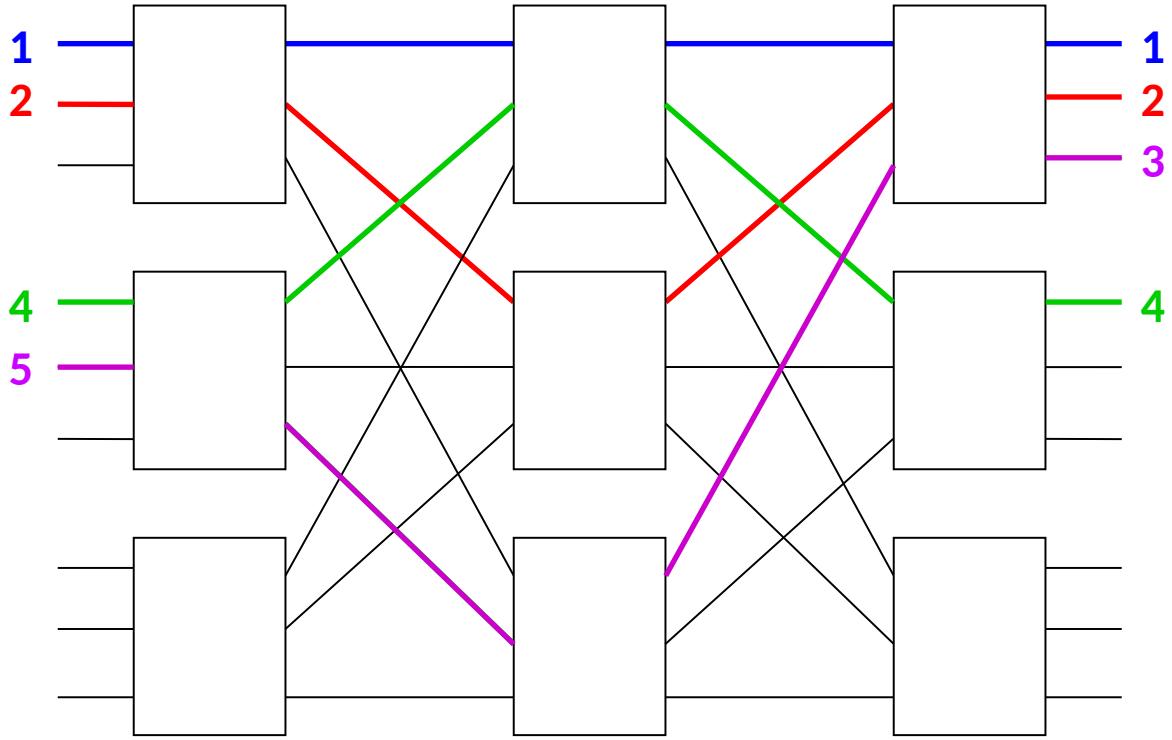
  (1,1), (2,2), (4,4), (5,3), ...



Clos network

Is Clos network for m = n non-blocking?

- **Example of connection in the network:**
- **(1,1), (2,2), (4,4), (5,3), ...**



Solution is to rearrange already existing connections in the network

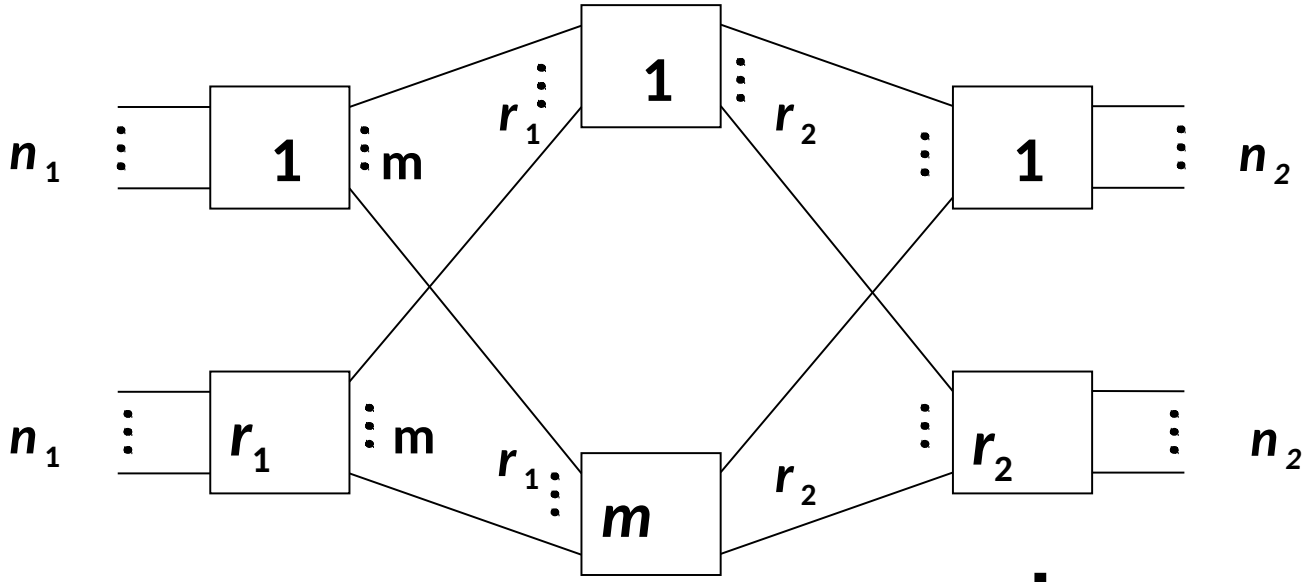
Clos network

The interconnection capability of the Clos network is dependent on the parameters n, r and m. For the given n, r and variable m there are many possibilities of interconnection. Non-blocking operations of the 3-stages network can be achieved in more than one way. We distinguish between 3 non-blocking Clos network modes:

- **strict-sense non-blocking network** (SNB), for **m≥2n–1**, unused input on an ingress switch can always be connected to an unused output on an egress switch without having to re-arrange existing calls
- **wide-Sense non-blocking** (WSN), if there exists an algorithm for initial and new route selection that grants all requests
- **rearrangeably non-blocking network** (RNB) for **m≥n**, unused input can always be connected to an unused output, but can require rearrange by assigning them to different central stage switches

The generalized three-stage Clos network C (n_1, r_1, m_2, n_2, r_2) is a three-stage network whose first stage consists of $n_1 \times m$ r_1 switches, third stage has r_2 switches dimension $m \times n_2$, $r_1 \times r_2$. If $n_1 = n_2$, then $r_1 = r_2$ we are talking about the symmetric Clos 3-stage network. C (n, r, m, n, r) denote C (m, n, r).

Clos network



For Clos network C (n_1, r_1, m, n_2, r_2), it is possible to specify required network state (defined by permutation P) into matrix with r_1 rows a r_2 columns where value specifies number of connections between given input and output switch

$$A = \begin{bmatrix} a_{1,1} & \cdots & a_{1,\mathrm{r2}} \\ \vdots & & \vdots \\ a_{\mathrm{r1},1} & \cdots & a_{\mathrm{r1},\mathrm{r2}} \end{bmatrix}$$

For matrix A is valid: $\sum_{i=1}^{r_1} a_{ij} = n_2 \, for \, \forall \, j = const.$ $\sum_{j=1}^{r_2} a_{ij} = n_1 \, for \, \forall \, i = const.$

Clos network

Consider permutation P and Clos network of size C (4,4,3):

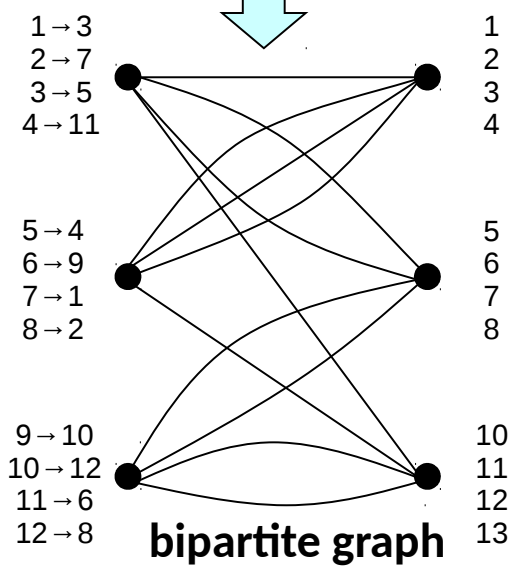
$$P=\left(\begin{array}{cccc|cccc|cccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ 3 & 7 & 5 & 11 & 4 & 9 & 1 & 2 & 10 & 12 & 6 & 8 \end{array}\right)$$

$$\begin{array}{c|ccc} & 1\text{-}4 & 5\text{-}8 & 9\text{-}12 \\ \hline 1\text{-}4 & 1 & 2 & 1 \\ 5\text{-}8 & 3 & 0 & 1 \\ 9\text{-}12 & 0 & 2 & 2 \end{array}$$

out switch

in switch

Interconnection matrix



bipartite graph

Clos network

The following algorithms are result from the chosen representation:

- interconnection matrix decomposition
- bipartite graph decomposition

Interconnection matrix decomposition:

- Decomposition of interconnection matrix A into partial matrices B_p , $p \in \{1, 2, \ldots, q\}$, such, that $A = B_1 + B_2 + \ldots + B_p + \ldots + B_q$ is valid and simultaneously $\forall B_p$ is valid

$b_{ij} \in \{0, 1\}$

$$\sum_{i=1}^{r_1} b_{ij} = 1 \quad \text{eventually} \quad \sum_{i=1}^{r_1} b_{ij} \leq 1$$

$$\sum_{j=1}^{r_2} b_{ij} = 1 \quad \text{eventually} \quad \sum_{j=1}^{r_2} b_{ij} \leq 1$$

we get configuration of the switches in the second stage of the network. For specific decomposition q! different switching settings in the second stage.

Clos network

- For example case:

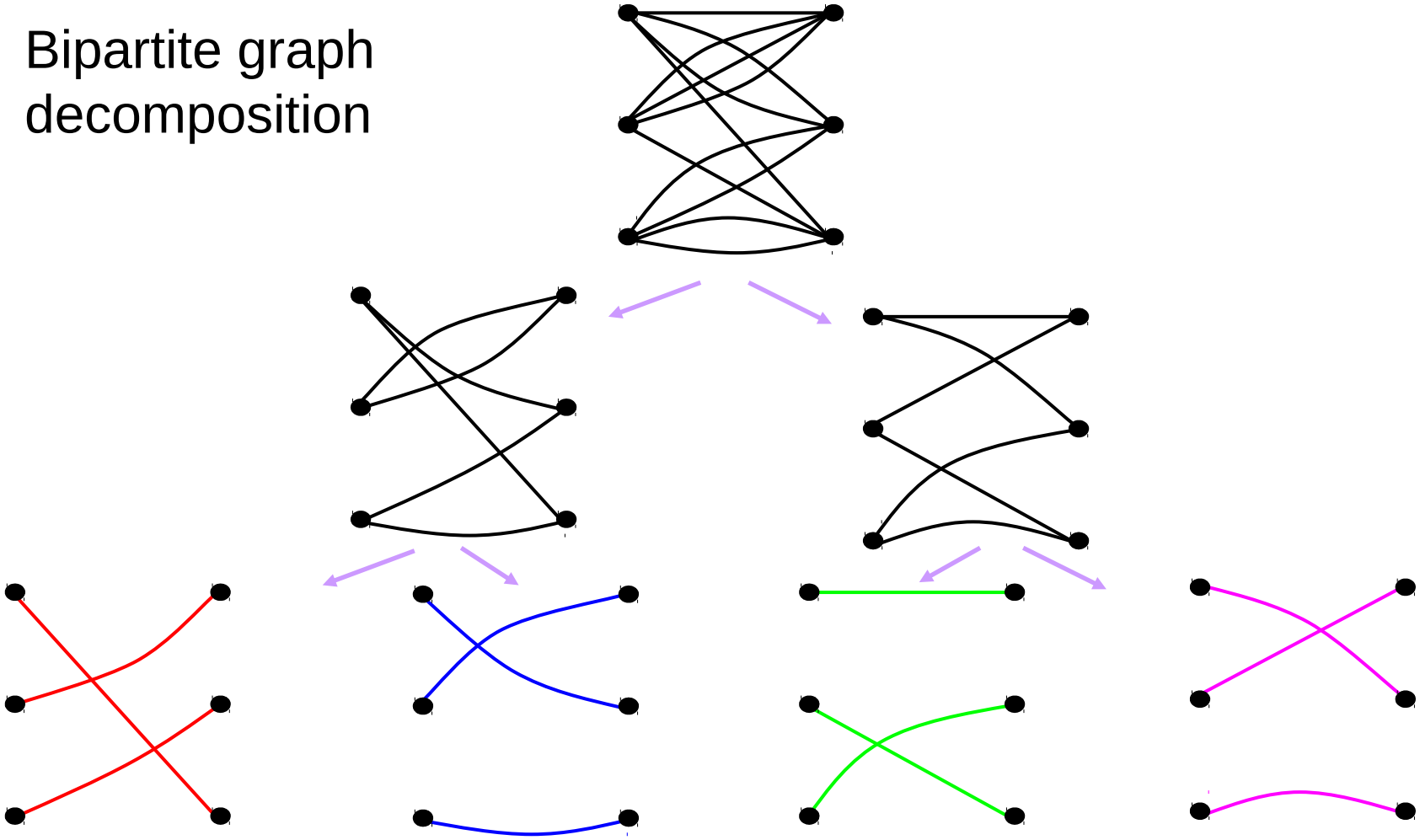
$$\begin{bmatrix} 1 & 2 & 1 \\ 3 & 0 & 1 \\ 0 & 2 & 2 \end{bmatrix}$$

$$S_1 = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad S_2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad S_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad S_4 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

A = S1 + S2 + S3 + S4

Clos network

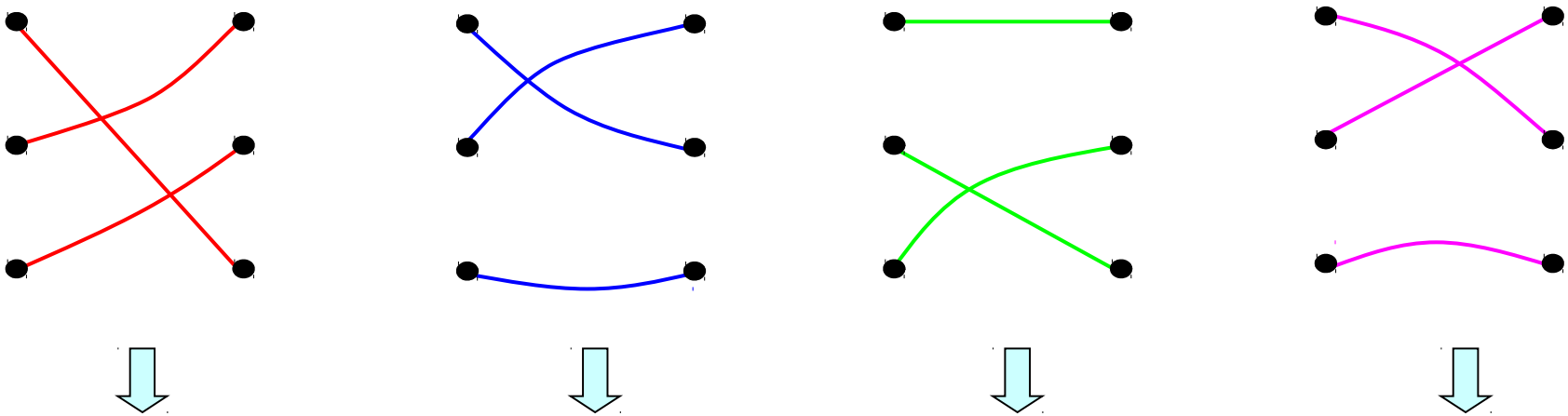
- Bipartite graph decomposition



Problem is transformed to coloring edges of graph such way that a given color appears in given vertex at most once. Each color represent one central switch.

Clos network

- Bipartite graph decomposition



$$S_1 = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad S_2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad S_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad S_4 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

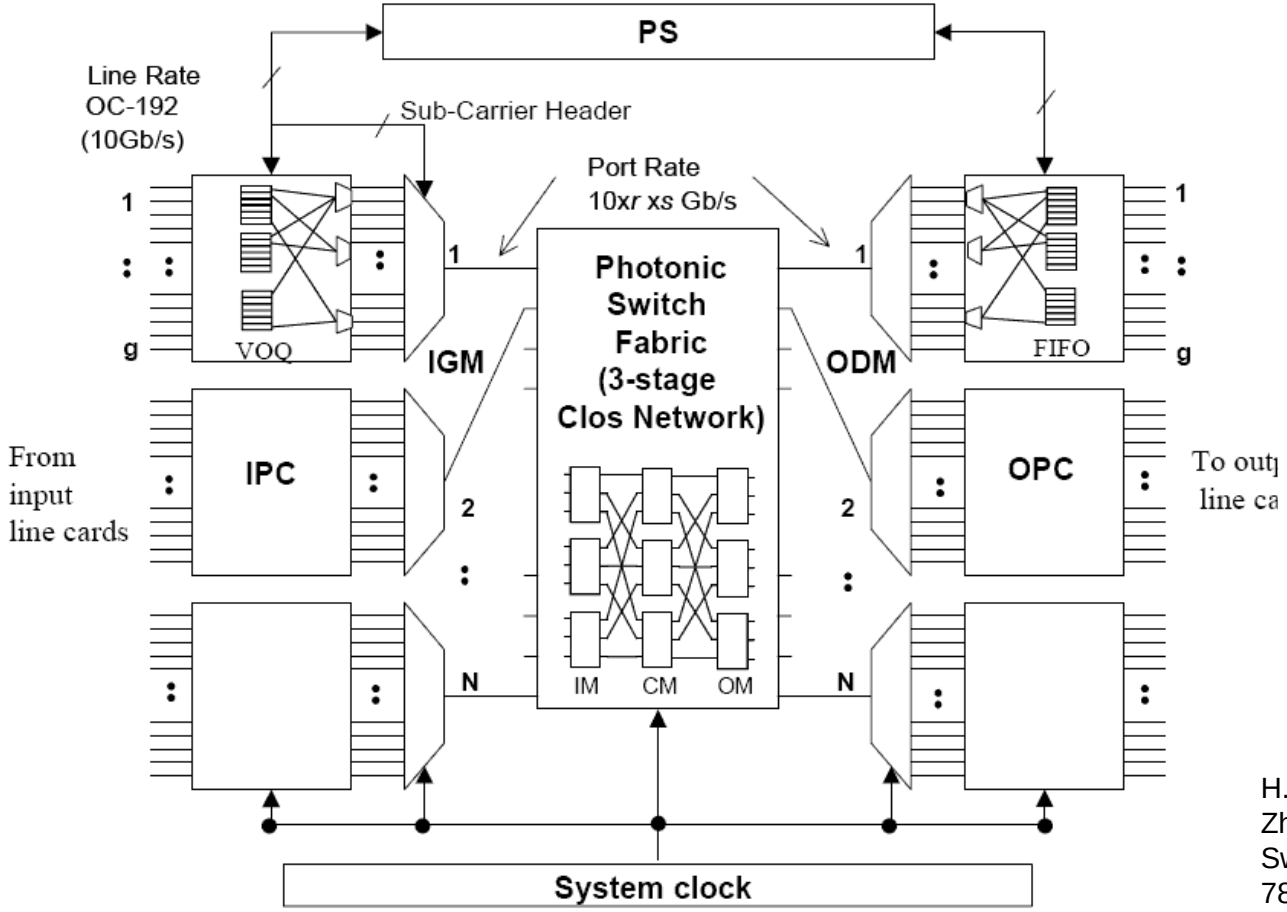
Clos network

- Matrices resulting from decomposition (whether graph or interconnection matrix) allow directly set the switches in the middle stage of the network.
- If setting of middle-class switches is known, it is not difficult to set the switches across the network.

- Remark: Beneš network is a special case of Clos network. Therefore, if we want to analyze the nonconflictability of the Beneš network, we can use the above mentioned algorithms for the Clos network.

Clos network

- Example: The architecture of the P3S peta byte packet optical switch of dimension the 6400x6400, is based on the Clos 3-stages network reaching a total of 1,024 petabit / s (160 Gbit / s for each port)

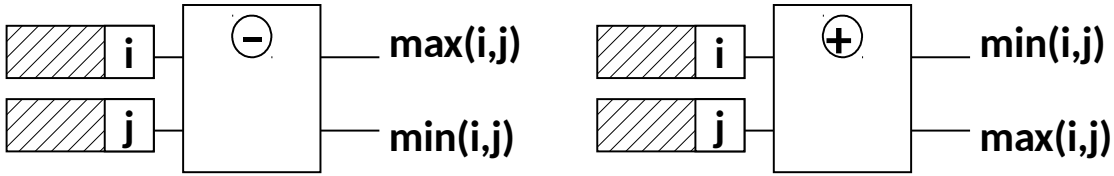


IPC: Input Port Interface Card
IGM: Input Grooming Module
ODM: Output Demultiplexng Module
OPC: Output Port Interface Card
PS: Packet Scheduler
VOQ: Virtual Output Queue
r: Cell Number / s: Speedup
g: Input Line Number
PSF: Photonic Switching Fabric
IM: Input Module
CM: Central Module
OM: Output Module

H. Jonathan Chao, Kung-Li Deng, and Zhigang Jing: A Petabit Photonic Packet Switch (P^3S), IEEE INFOCOM 2003, 0-7803-7753-2/03

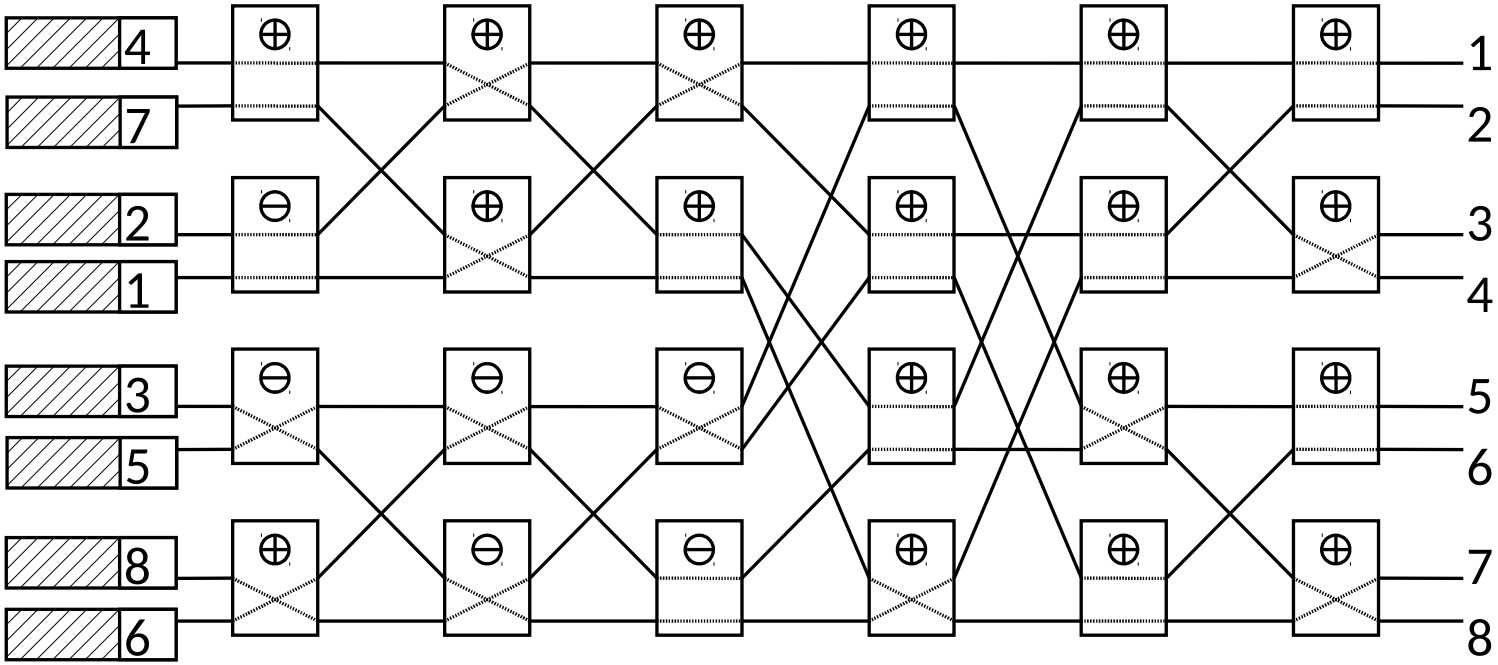
Batcher network

- Batcher or the sorting network sorts the input packets by their output address from the smallest addresses to the largest.
- The Batcher sorting network consists of the following elements:



- By appropriate line up of such elements, we get Batcher's network. The entire sorting network is composed of a sequence of **bitonic** sorters that sorts their outputs downward or ascending. In our case, they are 2 × 2 size sorters. If there is only one packet and only one address on the element input, it is considered to have a lower value.

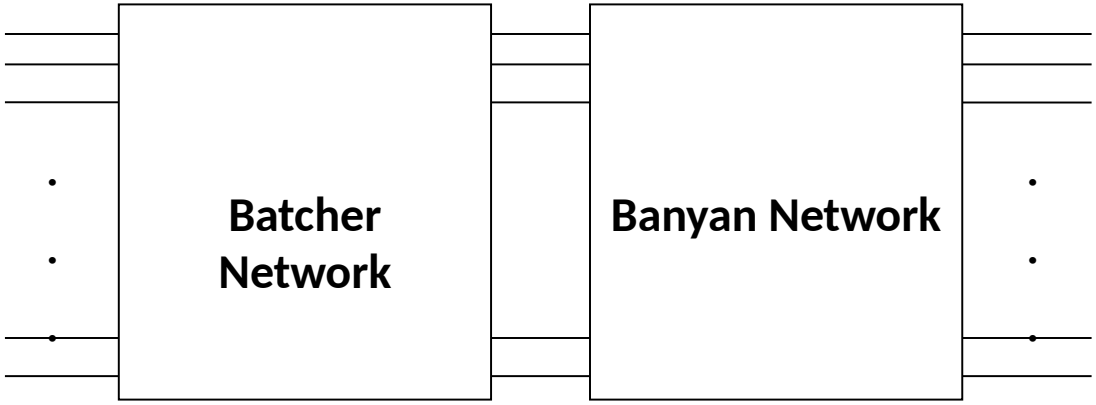
Batcher network



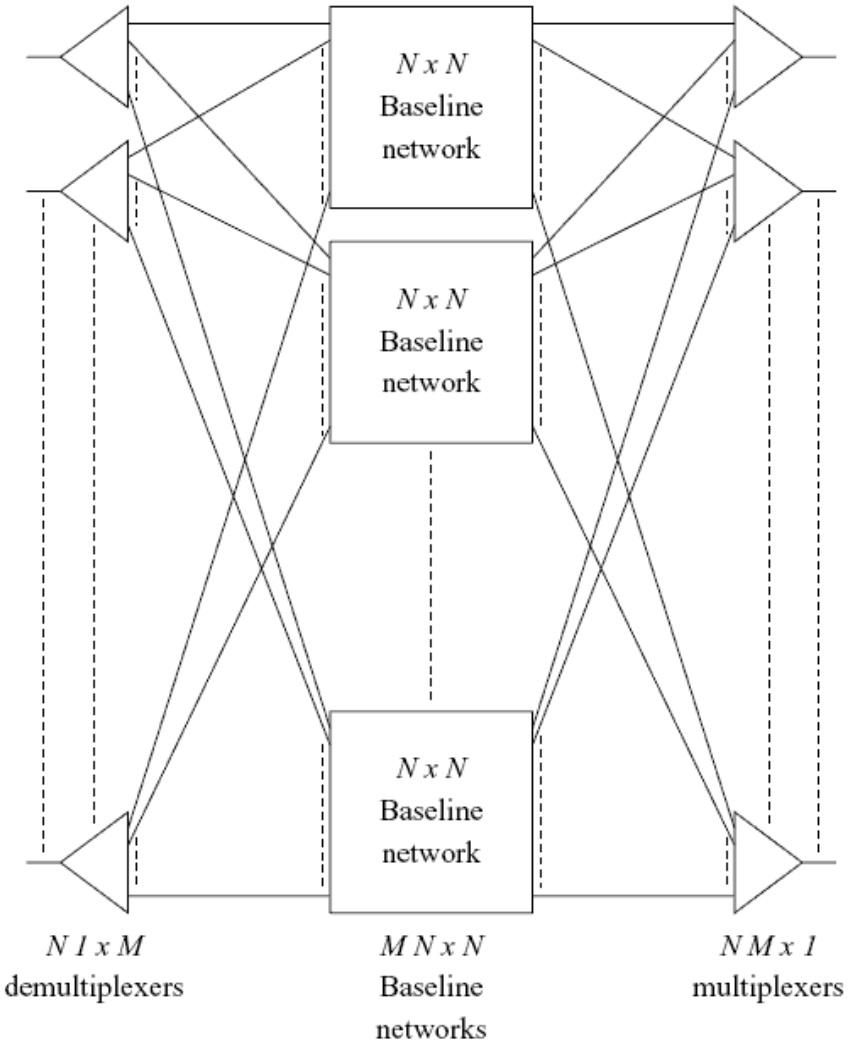
- At the Batcher network output, packets are sorted by ascending addresses, but they do not reach the correct output port by their destination address (unless all inputs are occupied). Therefore, Batcher network is further combined with banyan networks.

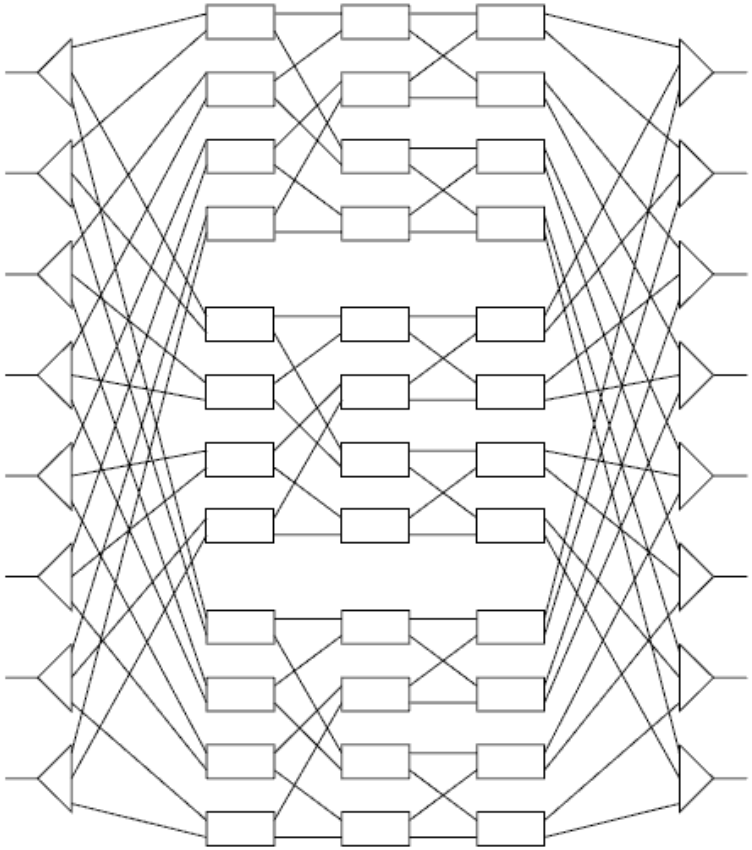
Batcher-Banyan network

- The Banyan network connected after the Batcher network has self-routing property to deliver packets to the right network output. Adding the sorting network in front of the banyan network eliminates HOL blocking if we assume that no two packets are routed to the same output. Because packets are sorted in the sorting network, there is no internal blocking in the banyan network. If there is a possibility of the same output addresses on the more network inputs, use the buffers is required.



Parallel Baseline network





Parallel Baseline network

- On both, input and output of baseline network (sometimes named as **Multi-Log$_2$N** network) is realized function of expansion and concentration. There are switching elements of size $1 \times m$ on the input which expands traffic and on the output are switching elements of size $m \times 1$ which concentrate traffic.

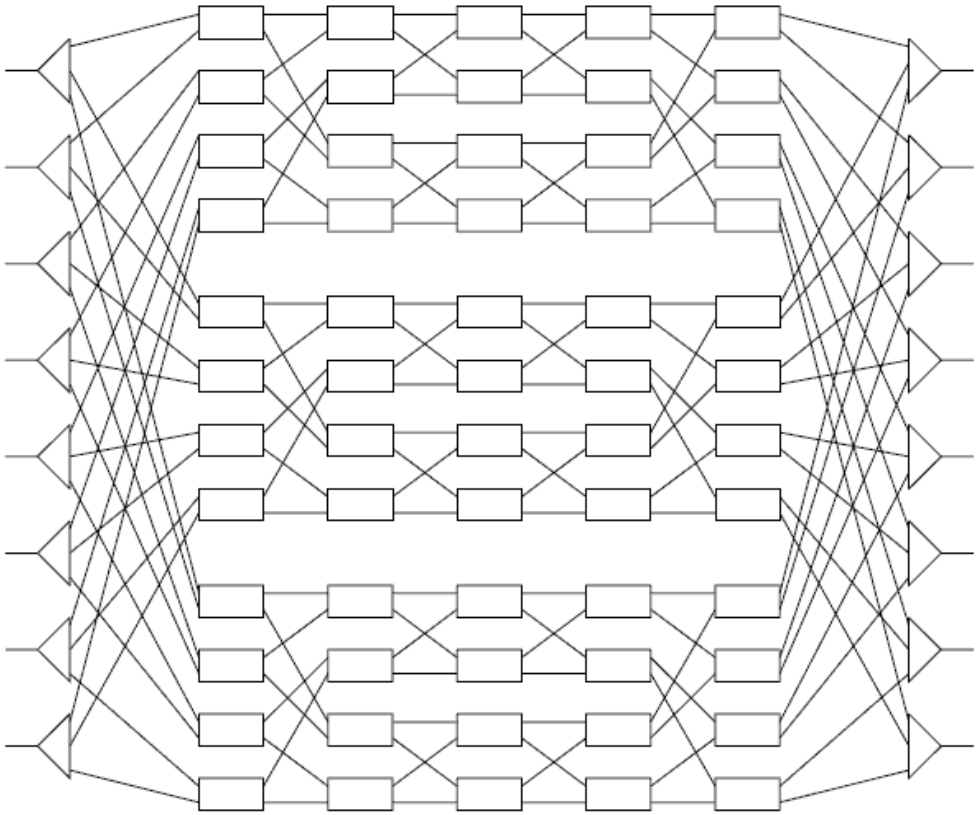
- If input and output stages (expansion and concentration) are not counted then number of network stages is **$n=\log_2 N$**. A graph theory allows to prove that if the number of parallel baseline sub-networks is

  $$m \geq 2^{(n/2)}$$

  Then parallel baseline network is reconfigurable without blocking. The consequence of the requirement is that optimal baseline network requires more switching elements than serial baseline network (Beneš network). On the other hand, the time of passing through parallel baseline networks is smaller than for the serial baseline network.

Cantor network

- Similarly as Clos network, even Cantorova network is classified as strictly non-blocking network . Cantor network $N \times N$ can be build from $\log_2 N$ Beneš networks, N demultiplexers and N multiplexers.



Literature and references

- Introduction to Parallel Computing, by V. Kumar, A. Grama, A. Gupta, G. Karypis, Benjamin-Cummins, 2nd edition, 2003. https://www-users.cs.umn.edu/~karypis/parbook/
- https://www.cs.uic.edu/~ajayk/c566/c566fa18.html
- http://homepages.inf.ed.ac.uk/rni/comp-arch/index.html
- http://www.ece.lsu.edu/ee7725/ln97.html
- Carpineli, J. D., and A. Yavuz Oruç. "A nonbacktracking matrix decomposition algorithm for routing on Clos networks." IEEE transactions on communications 41.8 (1993): 1245-1251.
- Ngo, Hung Q., and Van H. Vu. "Multirate rearrangeable clos networks and a generalized edge-coloring problem on bipartite graphs." SIAM Journal on Computing 32.4 (2003): 1040-1049.