Banyan-based switches

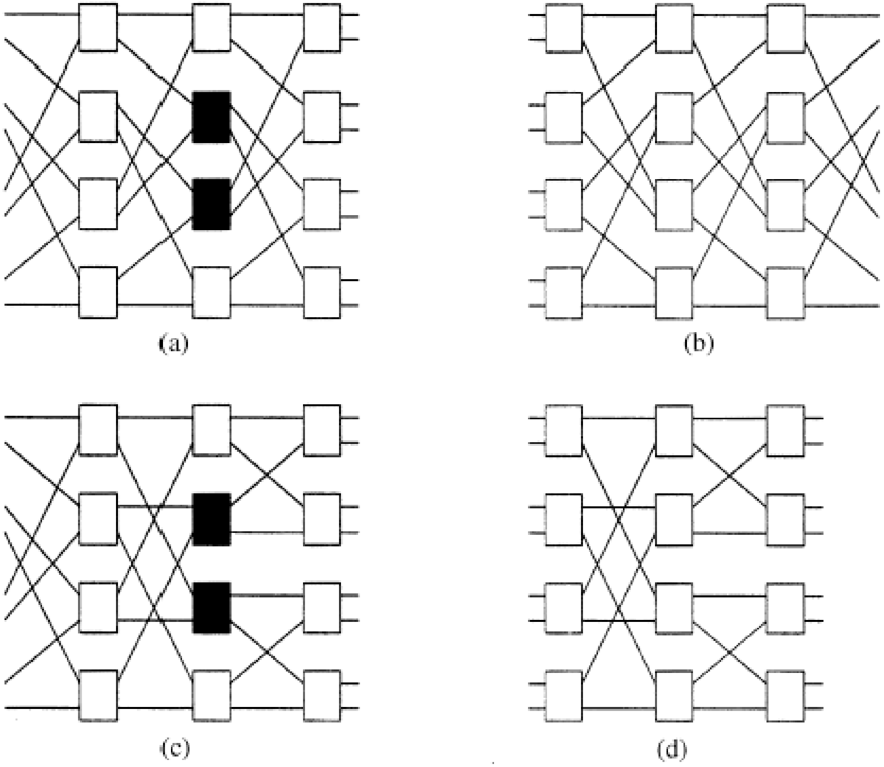
Masoud Sabaei

Associate professor

Department of Computer Engineering,
Amirkabir University of Technology

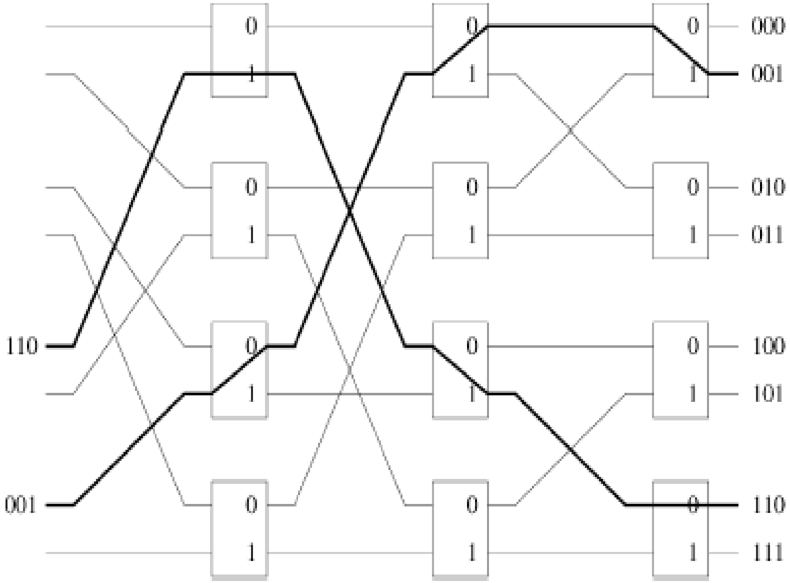
Introduction

- Multistage Network
  - First in circuit switched telephone networks
  - To aim nonblocking with less crosspoints than a crossbar
- Banyan network
  - Exactly one path from any input to any output
  - 4 classes
    - a) Shuffle-exchange (Omega)
    - b) Reverse shuffle-exchange
    - c) Narrow-sense banyan
    - d) Baseline



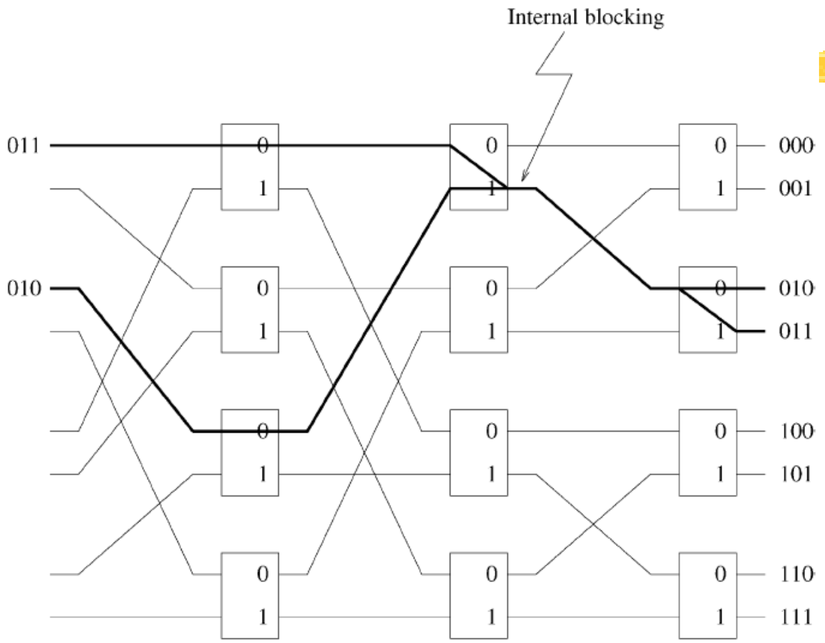
Introduction

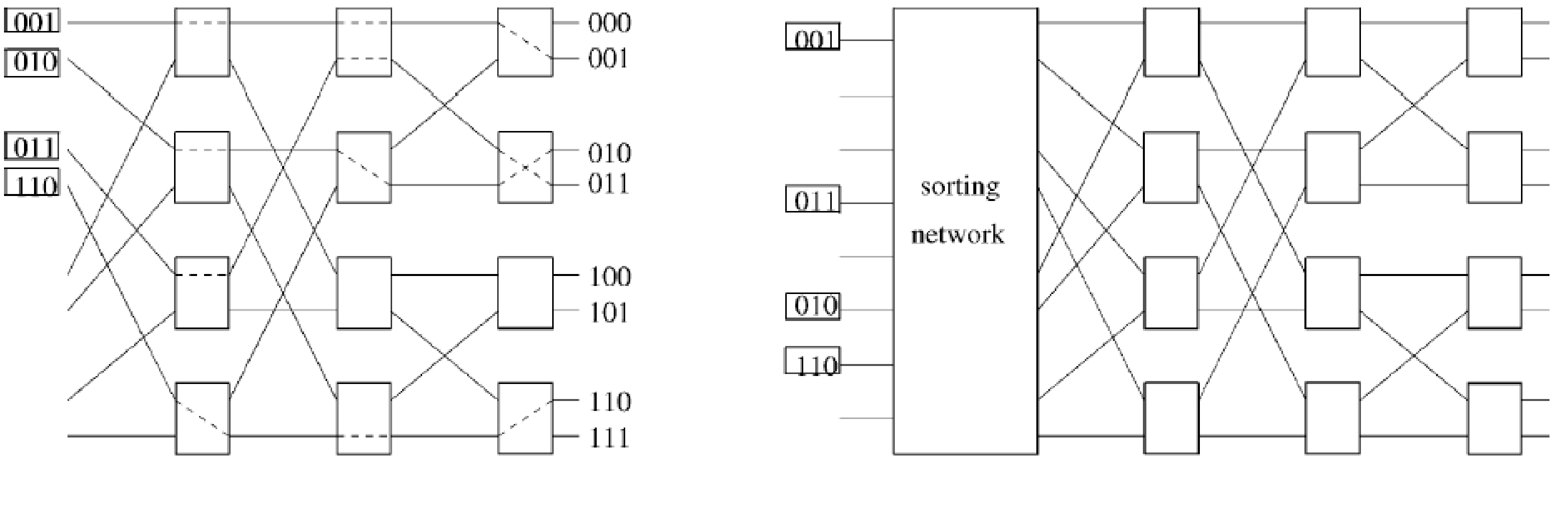
- Banyan network principal properties
  - Stages = n = $\text{Log}_2 N$, Nodes per stage = N/2
  - Self routing: one bit check in each step →
  - Regularity: Attractive for VLSI implementation



Internal blocking

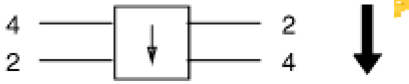
- Internal blocking
  - Cell is lost due to the contention on a link inside the network
  - Can not occur in banyan networks if
    - There is no idle input between any two active inputs
    - Destination address of cells are sorted
  - The need for sorting network

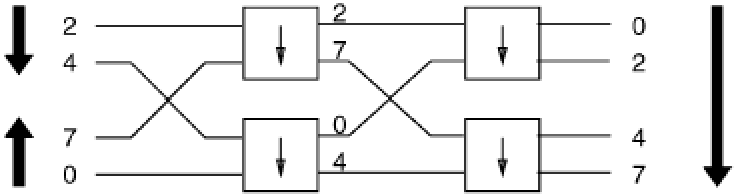


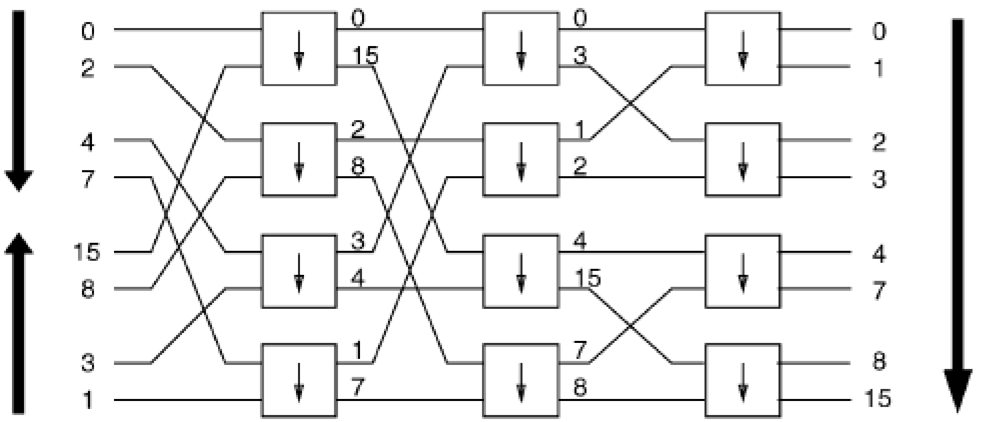


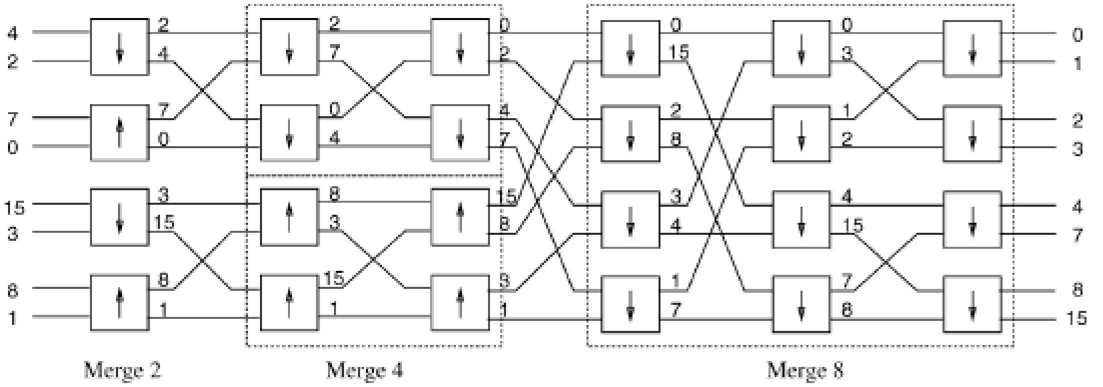
Batcher sorting network

- Merge network
  - Consists of 2*2 sorting elements
  - Partial sort
  - Merge2
  - Merge4
  - Merge8
  - MergeN contains $(N\mathrm{Log}_2N)/2$ SEs
- Batcher sorting network
  - A series of merge networks
  - 8*8: merge2 → merge4 → merge8
  - N*N batcher network
    - $(N\mathrm{Log}_2N)(\mathrm{Log}_2N+1)/4$ SEs
    - $1+2+\ldots+\mathrm{Log}_2N = (\mathrm{Log}_2N)(\mathrm{Log}_2N+1)/2$ Stages



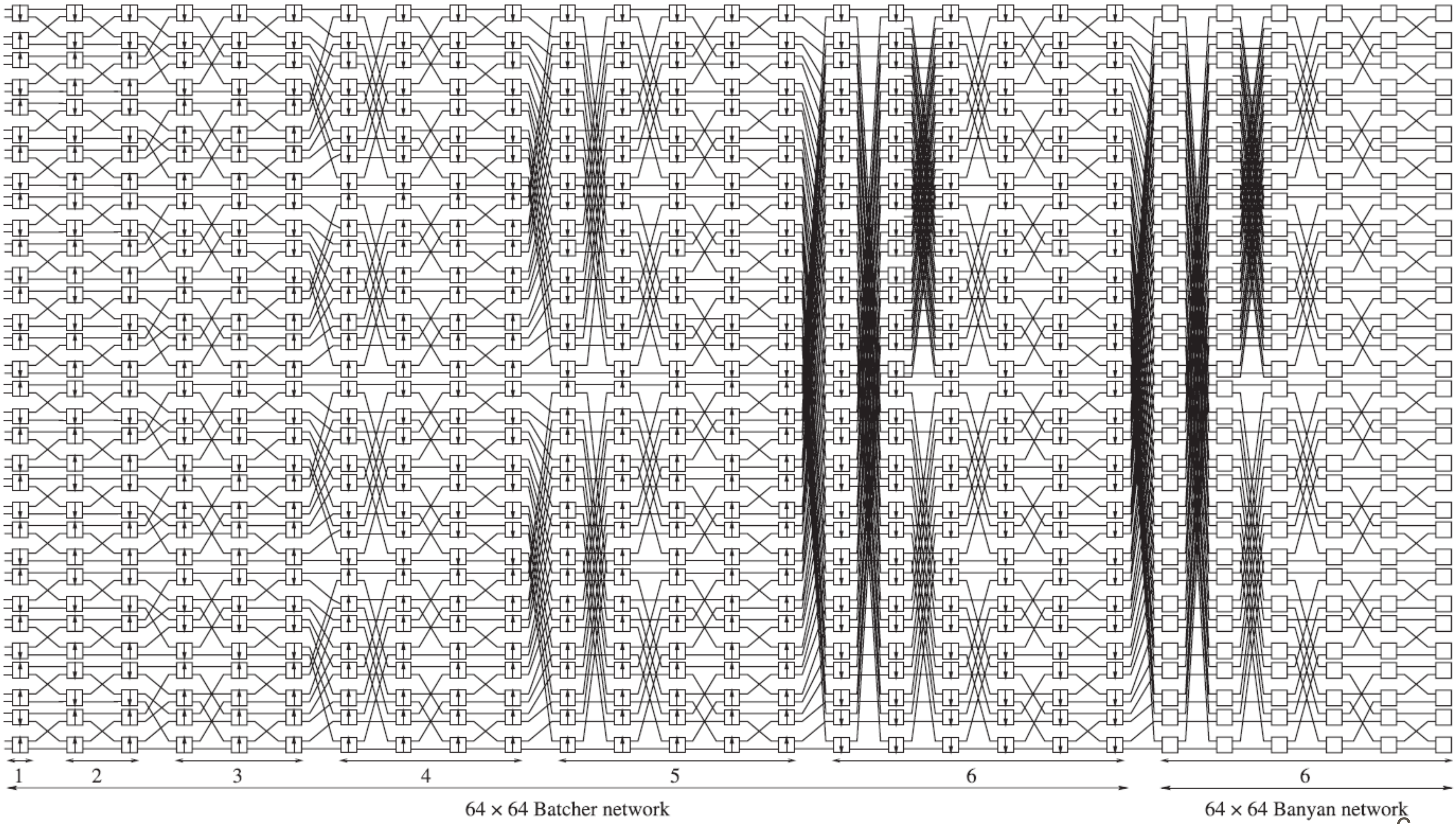






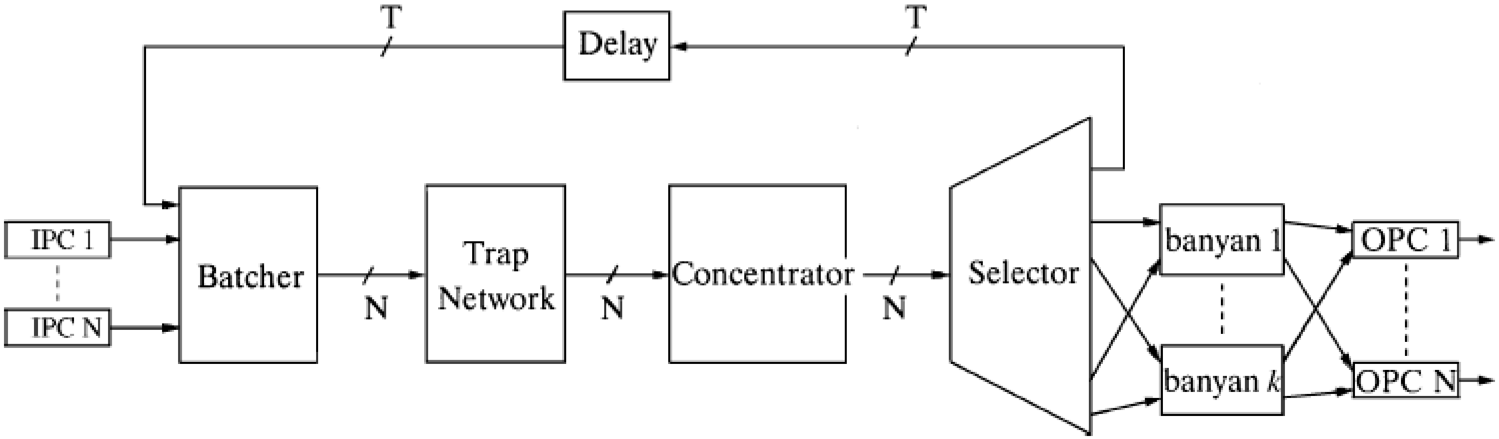
Batcher sorting network

- 64*64 batcher-banyan network!



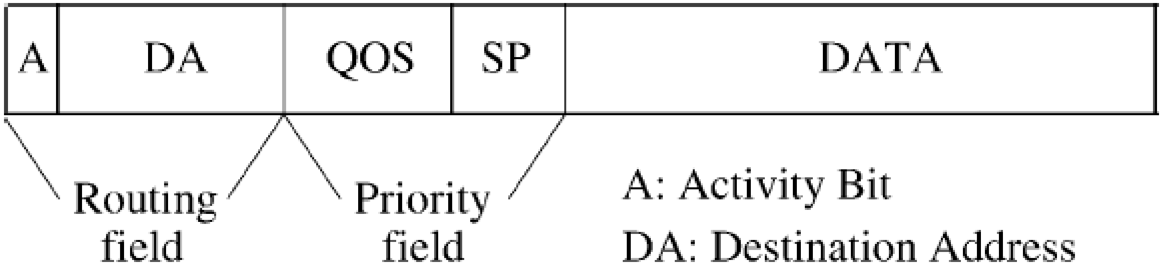
The sunshine switch

- The Sunshine switch
  - A batcher sorting network
  - k banyan networks in parallel
  - More than one path to each destination (k paths)
  - Recirculating queue
    - T paths in the figure
    - If more than k cells have the same output port
    - Excess cells are recirculated with a delay
  - Batcher network
    - Sorts in the order of port & priority
    - The highest priority cell for each port is selected by trap network



The sunshine switch

- The sunshine switch
  - Control Header
    - Is added to each cell by IPC
    - Two parts
      - Routing part
        - A: cell activity (non-emptiness)
        - DA: destination address
      - Priority part
        - QoS: quality of service (priority of cell)
        - SP: switch priority (assigned by switch to compensate the recirculation delay)



A: Activity Bit

DA: Destination Address

QOS: Quality of Service

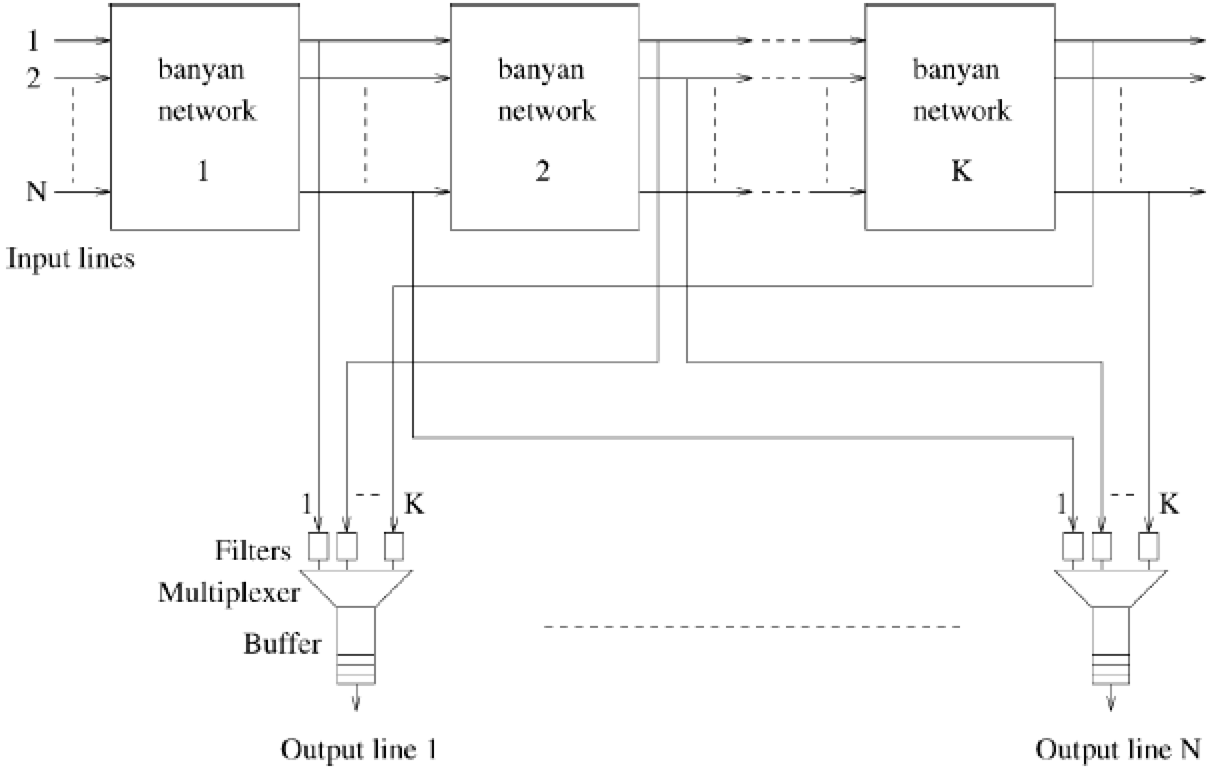
SP: Internal Switch Priority

Deflection routing

- Deflection
  - Two cells contend at a node
  - One of them will be routed incorrectly
- Works on deflection routing
  - Tandem banyan switch
  - Shuffle-exchange network with deflection routing
  - Dual shuffle exchange network with error-correcting routing

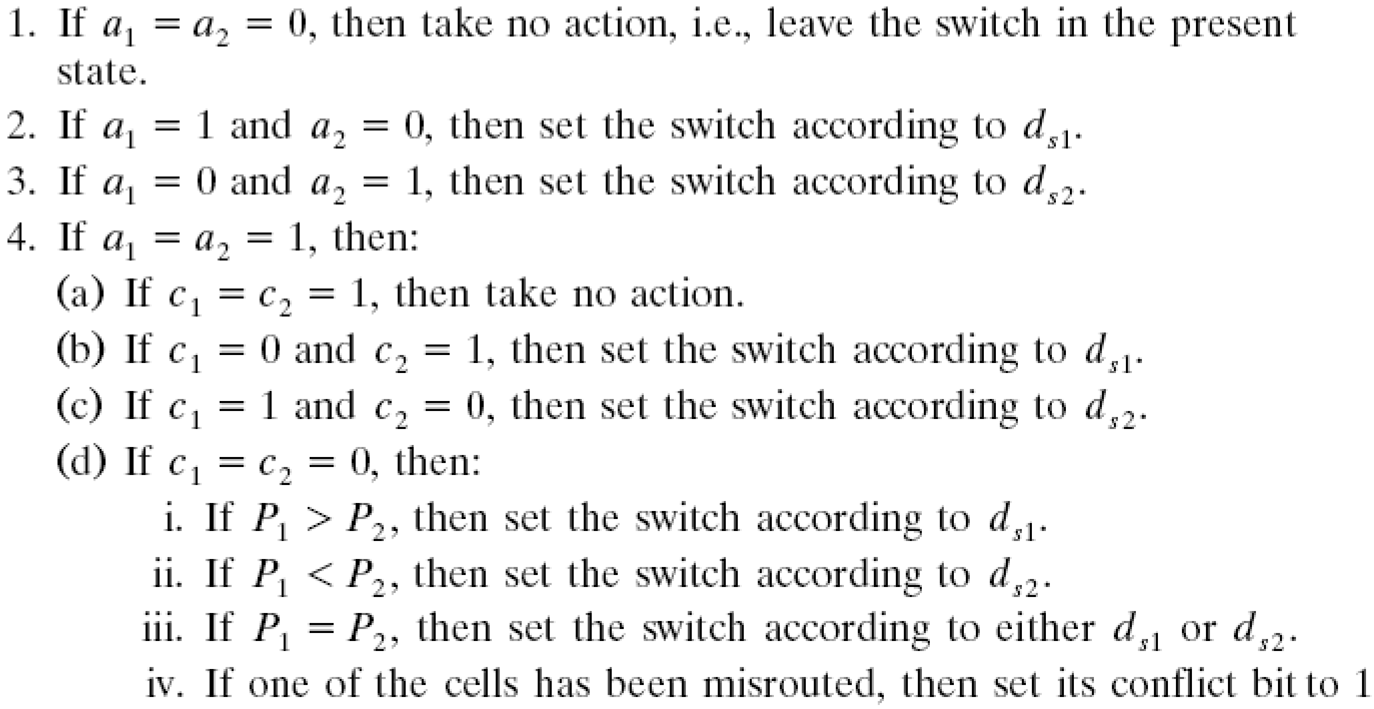
Tandem banyan switch

- Tandem banyan switch
  - Chain of K banyan networks
  - Deflected cells continue into the next banyan network
  - Correctly routed cells go to output buffers
  - One added banyan network → one order of magnitude reduction in deflections



Tandem banyan switch

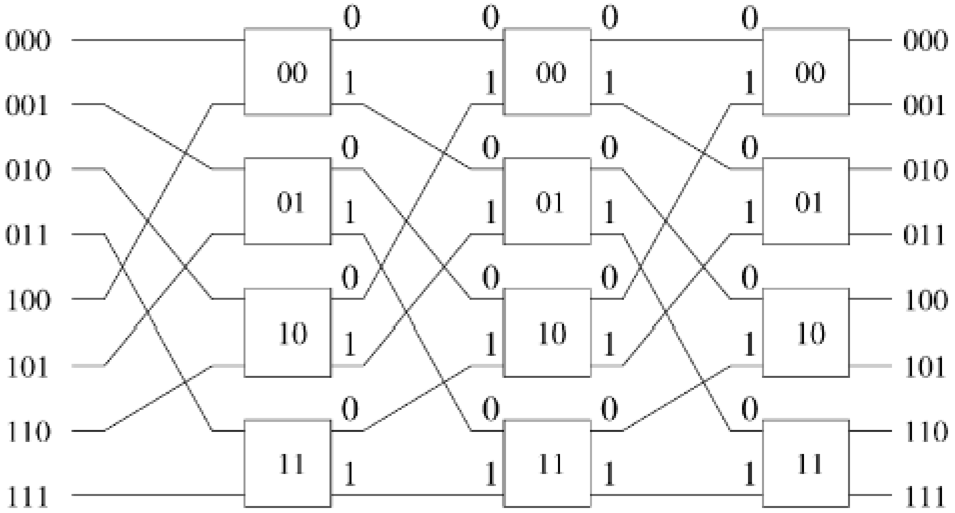
- Tandem banyan switch
  - Switching header
    - Activity bit: a
    - Conflict bit: c
    - Priority field: P
    - Address field: D
  - For two cells 1, 2 in the same stage s:

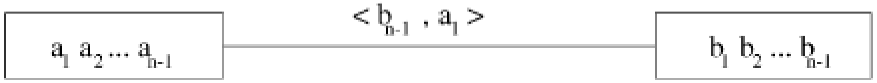


1. If $a_1 = a_2 = 0$, then take no action, i.e., leave the switch in the present state.
2. If $a_1 = 1$ and $a_2 = 0$, then set the switch according to d_{s1}.
3. If $a_1 = 0$ and $a_2 = 1$, then set the switch according to d_{s2}.
4. If $a_1 = a_2 = 1$, then:
   (a) If $c_1 = c_2 = 1$, then take no action.
   (b) If $c_1 = 0$ and $c_2 = 1$, then set the switch according to d_{s1}.
   (c) If $c_1 = 1$ and $c_2 = 0$, then set the switch according to d_{s2}.
   (d) If $c_1 = c_2 = 0$, then:
      i. If $P_1 > P_2$, then set the switch according to d_{s1}.
      ii. If $P_1 < P_2$, then set the switch according to d_{s2}.
      iii. If $P_1 = P_2$, then set the switch according to either d_{s1} or d_{s2}.
      iv. If one of the cells has been misrouted, then set its conflict bit to 1

Shuffle-exchange network with deflection routing

- N*N shuffle-exchange network (SN)
  - Stages = n = $\log_2 N$
  - SEs per stage = N/2
  - SE labels: numbers with n-1 bits length
  - SE input (output) labels: 1 bit numbers
  - How SE forwards cells
    - The cell with a 0 in i'th destination address bit goes to output 0
    - The cell with a 1 in i'th destination address bit goes to output 1
  - Network connections:
    - Consider binary lebels represented in form ($a_1 a_2 \ldots$)
    - Output a_n of node X=($a_1 a_2 \ldots a_{n-1}$) → input a_1 of node Y=($a_2 a_3 \ldots a_n$)
    - This link is represented as $<a_n, a_1>$





$a_2\ a_3 \ldots a_{n-1} = b_1\ b_2 \ldots b_{n-2}$

Shuffle exchange network with deflection routing

- The path from input to output
  - Is determined by:
    - Source address $s_1s_2\ldots s_n$
    - Destination address $d_1d_2\ldots d_n$

$$\begin{aligned}
S = s_1 \cdots s_n &\xRightarrow{\langle -,\, s_1\rangle} (s_2 \ldots s_n) \xRightarrow{\langle d_1,\, s_2\rangle} (s_3 \ldots s_n\, d_1) \\
&\xRightarrow{\langle d_2,\, s_3\rangle} \cdots \xRightarrow{\langle d_{i-1},\, s_i\rangle} (s_{i+1} \ldots s_n d_1 \ldots d_{i-1}) \\
&\xRightarrow{\langle d_i,\, s_{i+1}\rangle} \cdots \xRightarrow{\langle d_{n-1},\, s_n\rangle} (d_1 \cdots d_{n-1}) \\
&\xRightarrow{\langle d_n,\, 0\rangle} d_1 \ldots d_n = D.
\end{aligned}$$

    - An example: S=001 and D=101

$$001 \xrightarrow{<-,0>} 01 \xrightarrow{<1,0>} 11 \xrightarrow{<0,1>} 10 \xrightarrow{<1,0>} 101$$

$$S \xrightarrow{<-,s_1>} s_2s_3 \xrightarrow{<d_1,s_2>} s_3d_1 \xrightarrow{<d_2,s_3>} d_1d_2 \xrightarrow{<d_3,0>} D$$

    - SEs the cell passes
      - An string $s_2\ldots s_nd_1\ldots d_{n-1}$
      - An (n-1) bits window shifting one bit in each stage from left to right

Shuffle exchange network with deflection routing

- State of the traveling cell
  - Pair (R,X)
  - R: Current routing tag
  - X: Label of SE which cell resides
  - First state: ($d_n \ldots d_1$, $s_2 \ldots s_n$)
  - State transition (self routing algorithm):

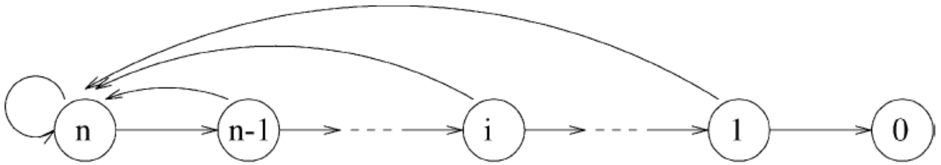
$$\underset{\text{input label } x_n}{(r_1 \ldots r_k, x_1 x_2 \ldots x_{n-1})} \overset{\text{exchange}}{\Longrightarrow} \underset{\text{output label } r_k}{(r_1 \ldots r_{k-1}, x_1 x_2 \ldots x_{n-1})}$$

$$\underset{\langle r_k, x_1 \rangle}{\overset{\text{shuffle}}{\Longrightarrow}} (r_1 \ldots r_{k-1}, x_2 \ldots x_{n-1} r_k).$$

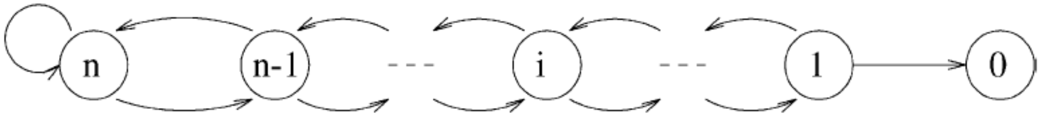
  - Routing bit is removed after each transition
  - Final state: ($d_1 \ldots d_{n-1}$)
- Deflected cells
  - Routing tag is reset to $d_n \ldots d_1$
  - Network is extended to have more than n stages

Dual shuffle-exchange network with error-correcting routing

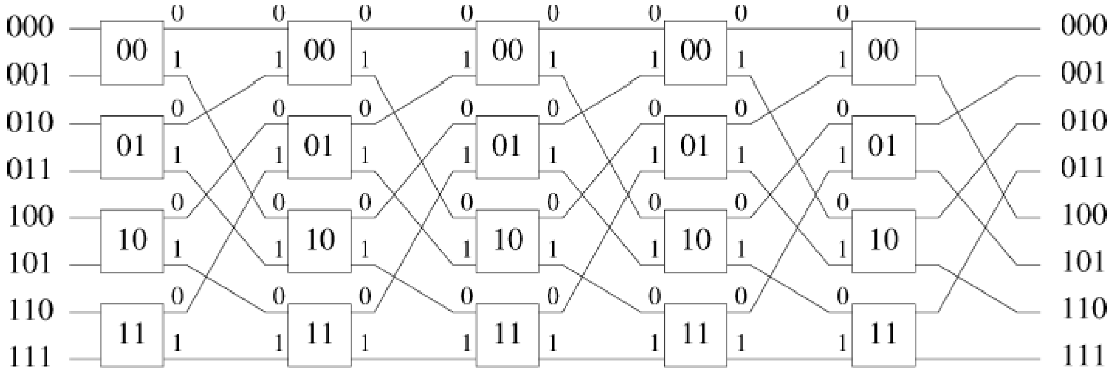
- SN with deflection routing (the previous scheme)
  - Highly inefficient: Routing must be restarted for deflected cell:



  - Desired network behavior:

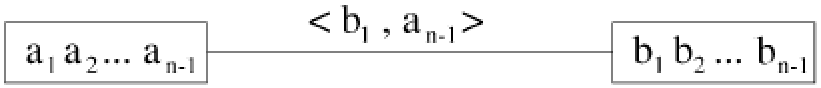


  - Dual shuffle-exchange network
    - A shuffle-exchange network (SN)
    - An unshuffle-exchange network (USN)
      - Mirror of SN network
      - An 8*8 example of USN:



Dual shuffle-exchange network with error-correcting routing

- USN: mirror image of SN
- Similar rules as SN:
  - SE connections:



$$a_1\,a_2 \ldots a_{n-2} = b_2\,b_3 \ldots b_{n-1}$$

  - Paths from inputs to outputs:

$$S = s_1 \ldots s_n$$

$$\xRightarrow{\langle - , s_n \rangle} (s_1 \ldots s_{n-1}) \xRightarrow{\langle d_n, s_{n-1} \rangle} (d_n\ s_1 \ldots s_{n-2})$$

$$\xRightarrow{\langle d_{n-1}, s_{n-2} \rangle} \ldots \xRightarrow{\langle d_{i+2}, s_{i+1} \rangle} (d_{i+2} \ldots d_n s_1 \ldots s_i)$$

$$\xRightarrow{\langle d_{i+1}, s_i \rangle} \ldots \xRightarrow{\langle d_2, s_1 \rangle} (d_2 \ldots d_n)$$

$$\xrightarrow{\langle d_1, 0 \rangle} d_1 \ldots d_n = D.$$

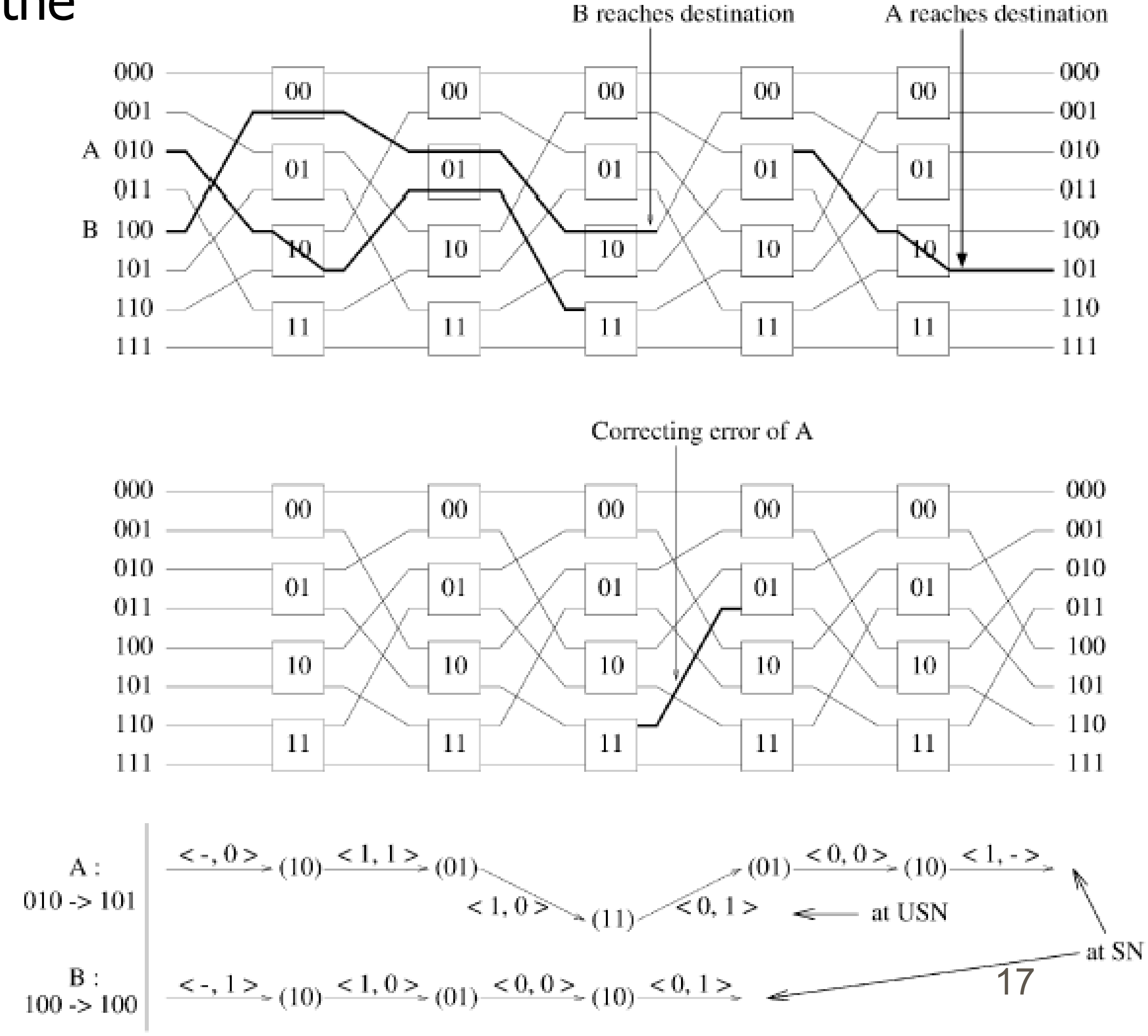
    - SEs the cell passes
      - An string $d_2 \ldots d_n s_1 \ldots s_{n-1}$
      - An (n-1) bits window shifting one bit in each stage from right to left
  - Traveling cell state diagram:
    - Initial state: $(d_1 \ldots d_n , s_1 \ldots s_{n-1})$
    - Final state: $(d_1 \ldots d_n)$
    - Transition:

$$\underset{\text{input label } x_n}{(r_1 \ldots r_k , x_1 x_2 \ldots x_{n-1})} \xRightarrow{\text{exchange}} \underset{\text{output label } r_k}{(r_1 \ldots r_{k-1} , x_1 x_2 \ldots x_{n-1})}$$

$$\xRightarrow[\langle r_k , x_{n-1} \rangle]{\text{unshuffle}} (r_1 \ldots r_{k-1} , r_k x_1 \ldots x_{n-2})$$

Dual shuffle-exchange network with error-correcting routing

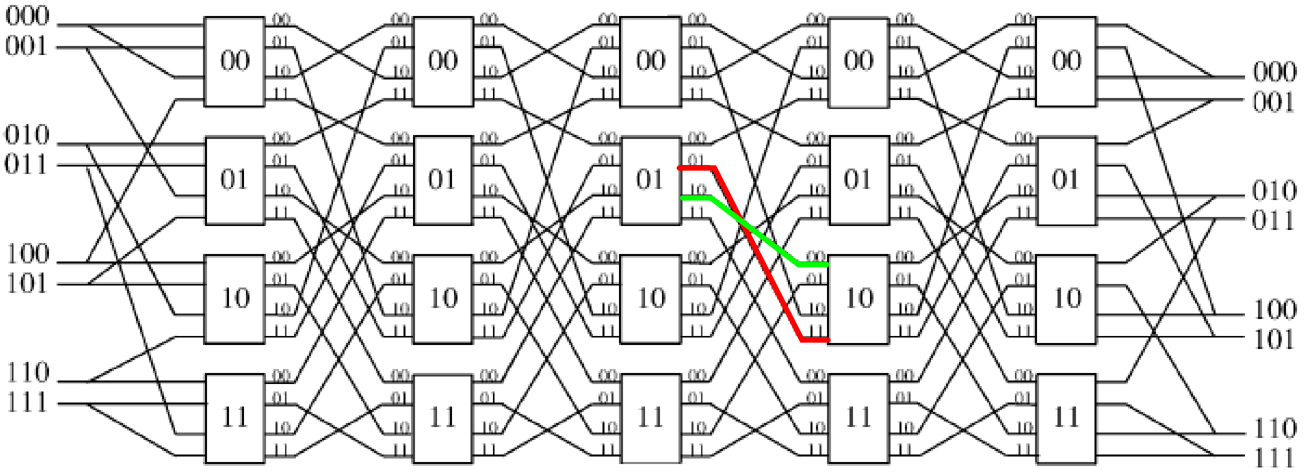
- Consider a USN overlaid on a SN
- USN can undo what SN performs
- Deflected cell can return to the state before deflection
- Example:



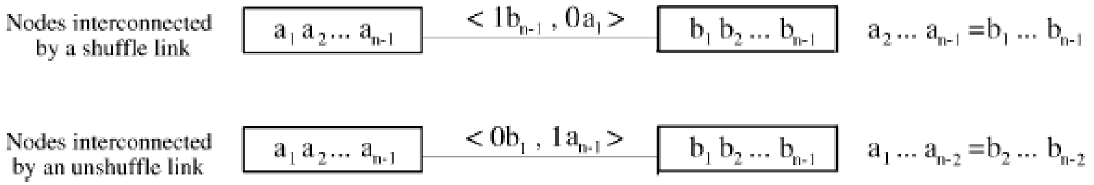
Dual shuffle-exchange network with error-correcting routing

- Error correction procedure
  - Cell state: $(r_1 \ldots r_k, x_1 \ldots x_{n-1})$
  - Cell should be sent via link $<r_k, x_1>$ to the next stage
  - It is deflected to link $<r'_k, x_1>$
  - It goes to node $(x_2 \ldots x_{n-1} r'_k)$ in the next stage
  - Error correction procedure does not remove the bit r_k
  - Instead, it attaches bit x_1 to the routing tag
  - New state: $(r_1 \ldots r_k x_1, x_2 \ldots x_{n-1} r'_k)$
  - Then the cell is moved to the USN
  - It will be sent via link $<x_1, r'_k>$
  - It will return to the state before deflection: $(r_1 \ldots r_k, x_1 \ldots x_{n-1})$
  - If cell is deflected in the USN, it can be corrected in SN in a similar way

Dual shuffle-exchange network with error-correcting routing



- Merging SN and USN
  - SN: 2*2 SEs
  - USN: 2*2 SEs
  - Merged (dual SN): 4*4 SEs

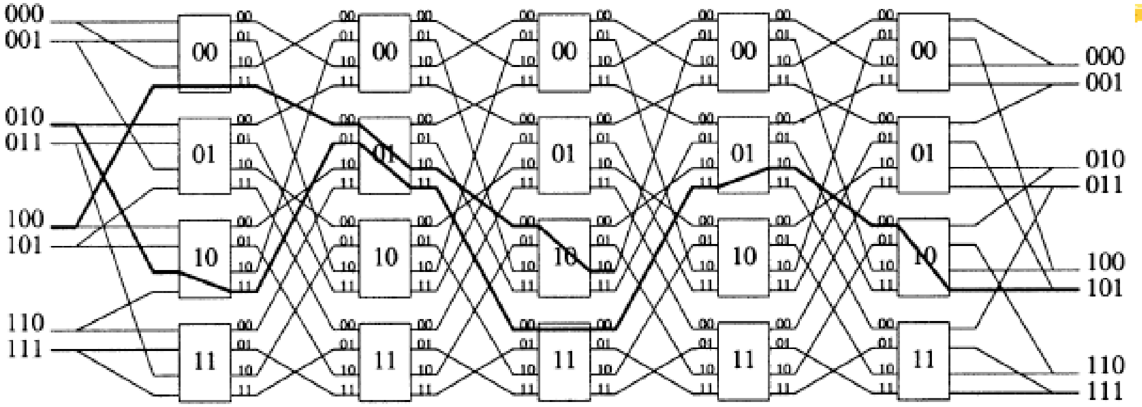


Nodes interconnected by a shuffle link
$a_1 a_2 \ldots a_{n-1}$
$< 1b_{n-1} , 0a_1 >$
$b_1 b_2 \ldots b_{n-1}$
$a_2 \ldots a_{n-1} = b_1 \ldots b_{n-1}$
Nodes interconnected by an unshuffle link
$a_1 a_2 \ldots a_{n-1}$
$< 0b_1 , 1a_{n-1} >$
$b_1 b_2 \ldots b_{n-1}$
$a_1 \ldots a_{n-2} = b_2 \ldots b_{n-2}$

- Labeling
  - Inputs: 00..11
  - Outputs: 00..11
  - Unshuffle links: <0a,1b>
    - Connect outputs 10 or 11 to inputs 00 or 01 of the next stage
  - Shuffle links: <1a,0b>
    - Connect outputs 00 or 01 to inputs 10 or 11 of the next stage
- Connections
  - Consider two nodes A=($a_1 \ldots a_{n-1}$) , B=($b_1 \ldots b_{n-1}$)
  - They are connected via unshuffle link <0b_1,1a_{n-1}> if $a_1 \ldots a_{n-2} = b_2 \ldots b_{n-1}$
  - They are connected via shuffle link <1b_{n-1},0a_1> if $a_2 \ldots a_{n-1} = b_1 \ldots b_{n-2}$

Dual shuffle-exchange network with error-correcting routing

- An example
  (The previous example)



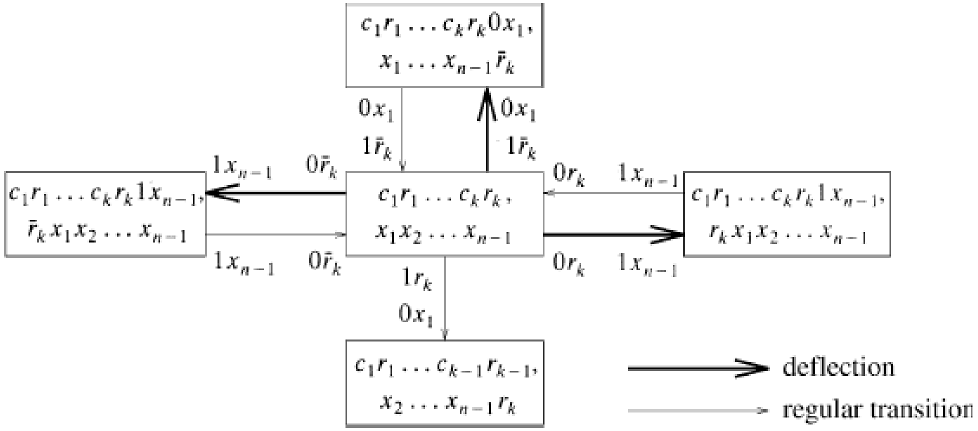
State transition of cells A and B

error error correction

A: 010 -> (111011, 10) -> (1110, 01) -> (111000, 11) -> (1110, 01) -> (11, 10) -> 101

B: 100 -> (101011, 00) -> (1010, 01) -> (10, 10) ->100

- State diagram



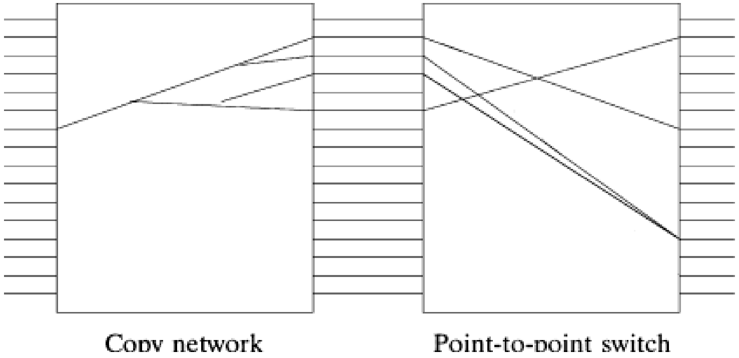
- Transitions
  for deflection

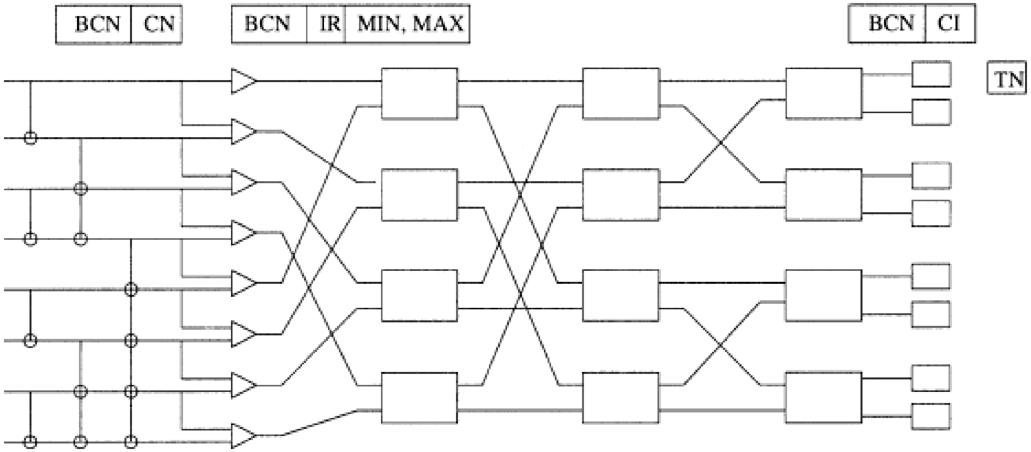
$$(c_1r_1 \dots c_kr_k, x_1 \cdots x_{n-1})$$

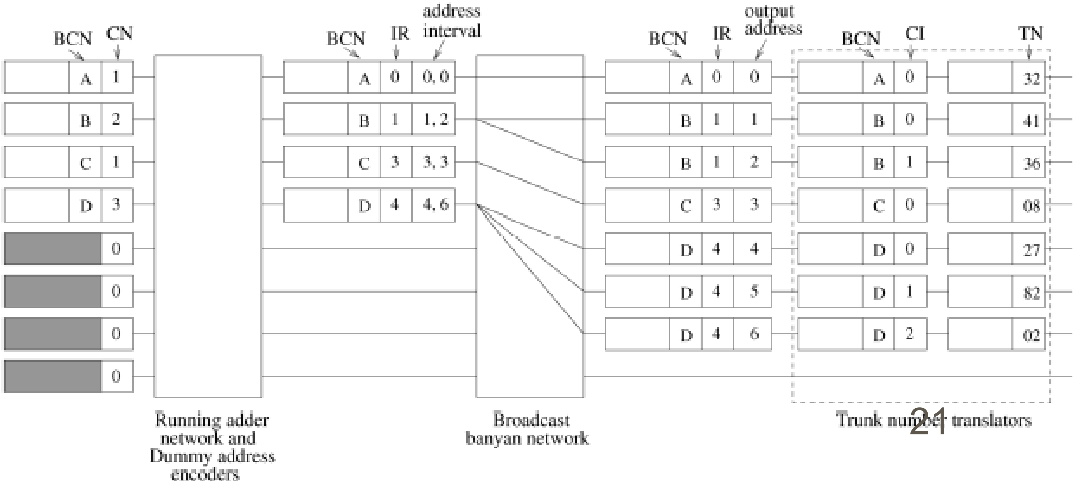
$$\begin{cases} \xRightarrow{\langle 0r, 1x_{n-1}\rangle} & (c_1r_1 \dots c_kr_k1x_{n-1}, rx_1 \dots x_{n-2}) \quad \text{if } c_kr_k \neq 0r, \\ \xRightarrow{\langle 1r, 0x_1\rangle} & (c_1r_1 \dots c_kr_k0x_1, x_2 \dots x_{n-1}r) \quad \text{if } c_kr_k \neq 1r. \end{cases}$$

Multicast copy networks

- Point-to-multipoint communications
  - A copy network (replicates cells)
  - A point-to-point switch
- The copy network components
  - Running address network (RAN)
  - Dummy address encoder (DAE)
  - Broadcast banyan network
  - Trunk number translator
- Acronyms in figure:
  - CN: Number of copies
  - IR: Index reference
  - CI: Copy index
  - BCN: Broadcast channel number
- An example







BCN	CN
A	1
B	2
C	1
D	3
	0
	0
	0
	0

Running adder network and Dummy address encoders

BCN	IR	address interval
A	0	0, 0
B	1	1, 2
C	3	3, 3
D	4	4, 6

Broadcast banyan network

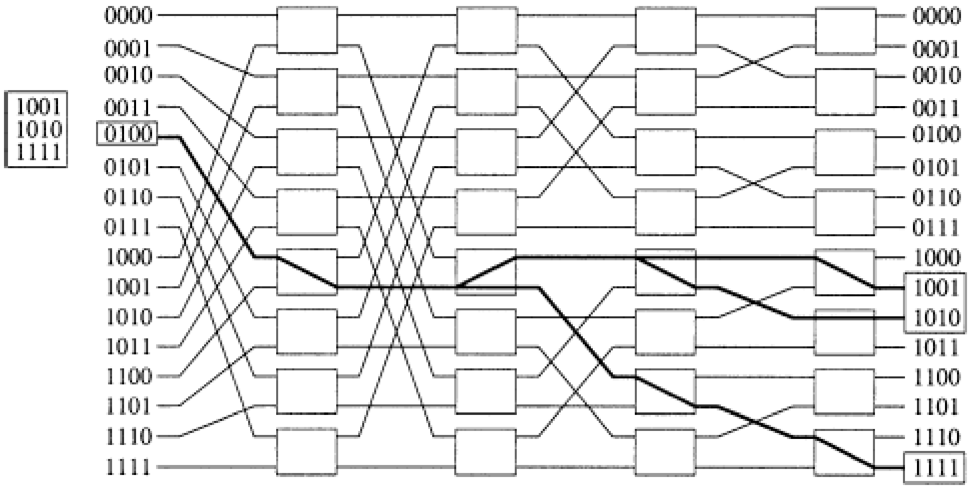
BCN	IR	output address
A	0	0
B	1	1
B	1	2
C	3	3
D	4	4
D	4	5
D	4	6

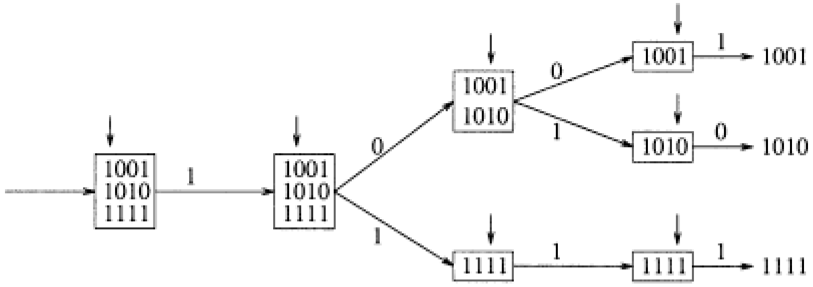
BCN	CI	TN
A	0	32
B	0	41
B	1	36
C	0	08
D	0	27
D	1	82
D	2	02

Trunk number translators

Multicast copy networks

- Broadcast banyan network
  - SEs can replicate cells
  - 3 possibilities (2 bits needed in cell header):
    - Cell goes to output 0
    - Cell goes to output 1
    - Cell is replicated to both outputs
  - Generalized self routing algorithm
    - Current bit of all destination addresses are checked at each stage
    - All zero → cell goes through output 0
    - All one → cell goes through output 1
    - Some zero, some one → cell is replicated
    - An example



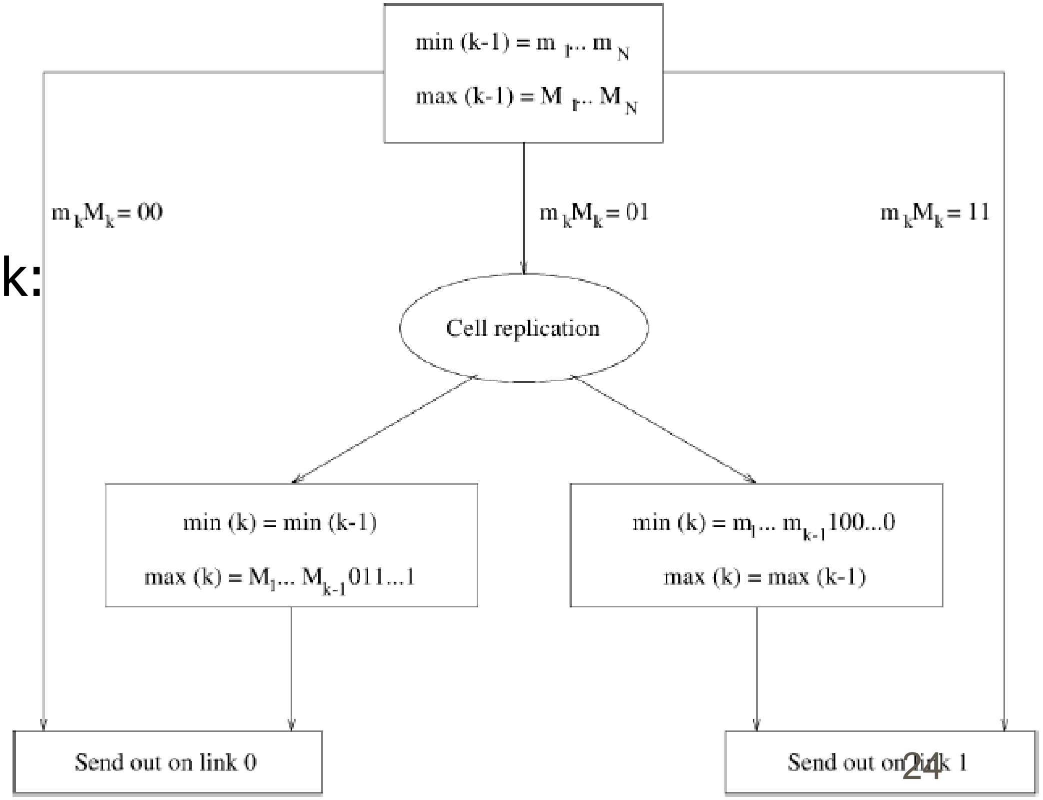


Multicast copy networks

- Broadcast banyan network
  - Generalized self routing algorithm
    - Problems
      - A variable number of destination addresses
        - Must be recorded in cell header
        - Must be checked at SEs at each stage
        - Need to be processed for cell header modifications
      - Cell path forms a tree → more blocking
    - Solution
      - Boolean interval splitting algorithm

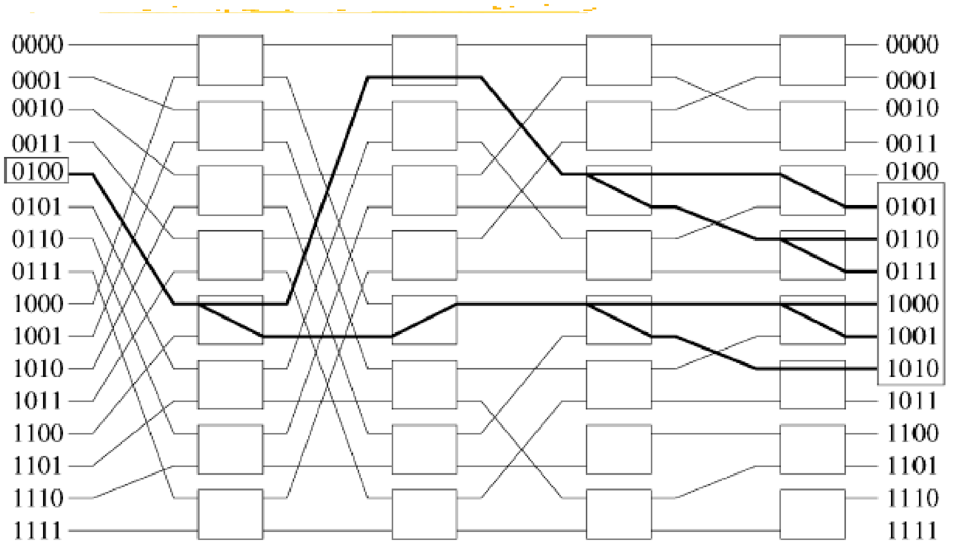
Multicast copy networks

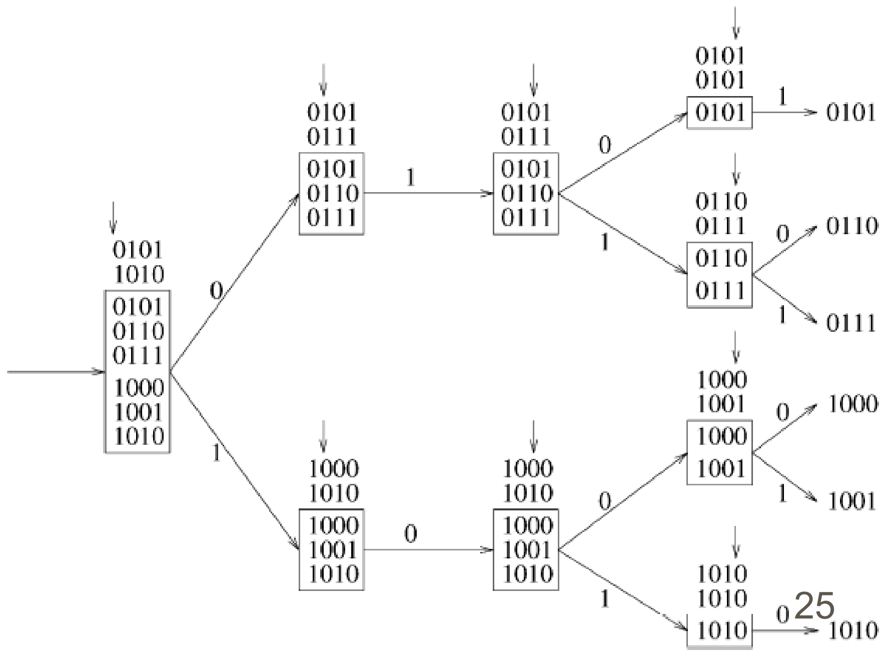
- Boolean interval splitting algorithm
  - Interval: a set of N-bits numbers between min and max
  - Interval represents a set of contiguous destination addresses
  - Stage k
    - min(k-1) = $m_1 \ldots m_k$
    - max(k-1) = $M_1 \ldots M_k$
    - Self routing at stage k:


min (k-1) = $m_1 \ldots m_N$
max (k-1) = $M_1 \ldots M_N$
$m_k M_k = 00$
$m_k M_k = 01$
$m_k M_k = 11$
Cell replication
min (k) = min (k-1)
max (k) = $M_1 \ldots M_{k-1}011\ldots1$
min (k) = $m_1 \ldots m_{k-1}100\ldots0$
max (k) = max (k-1)
Send out on link 0
Send out on link 1

Multicast copy networks

- Boolean interval splitting algorithm
  - An example





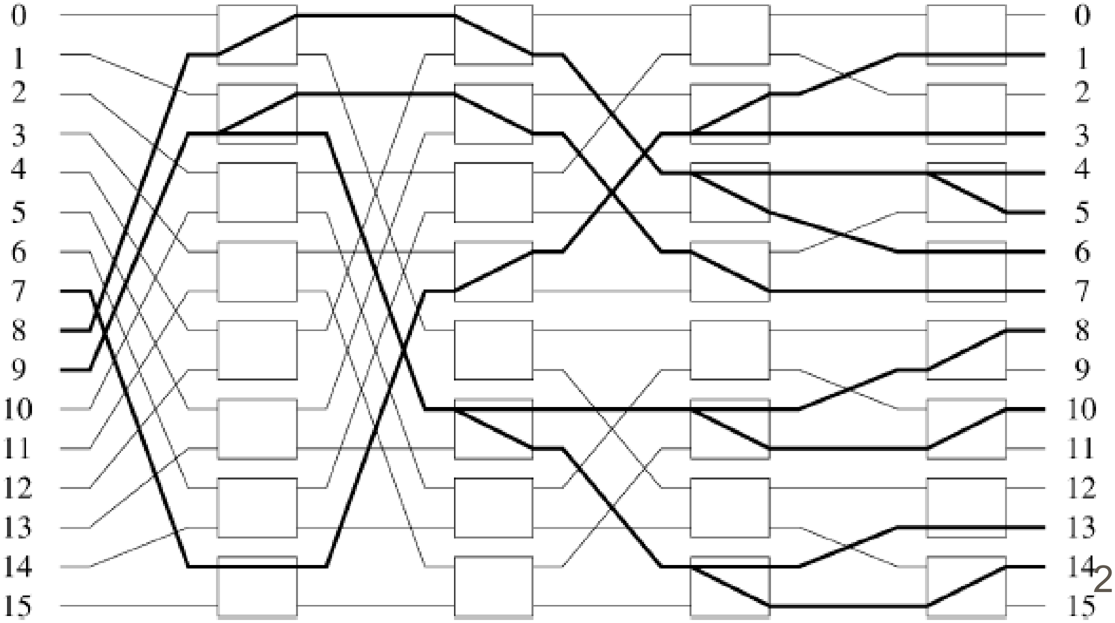
Multicast copy networks

- Broadcast banyan network is nonblocking when input cells are
  - Monotonic
    - Output port sets are sorted
  - Concentrated
    - No idle inputs exists between active outputs
  - An example

$x_1 = 7 \quad Y_1 = \{1,3\}$

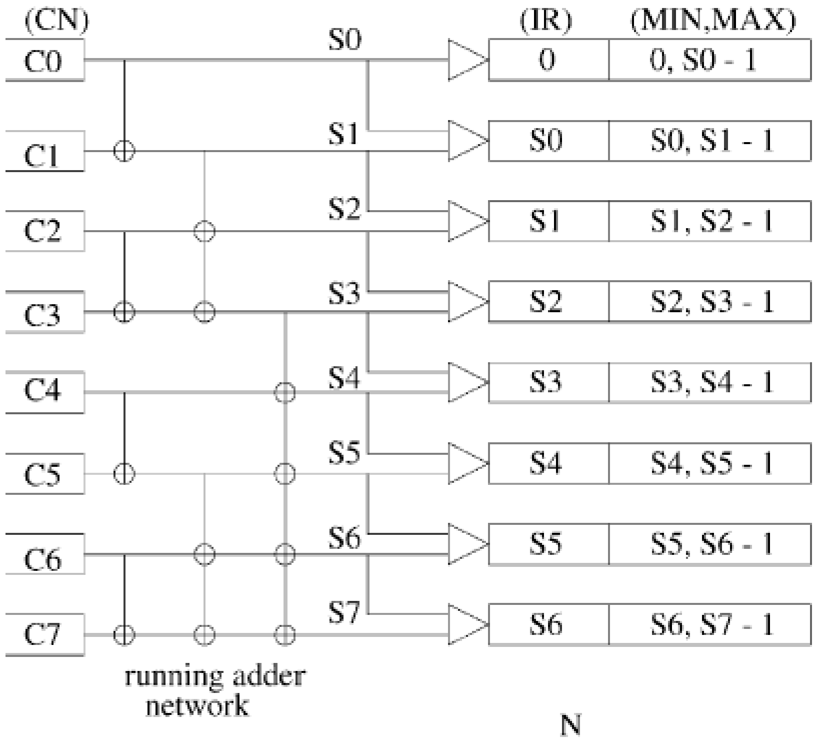
$x_2 = 8 \quad Y_2 = \{4,5,6\}$

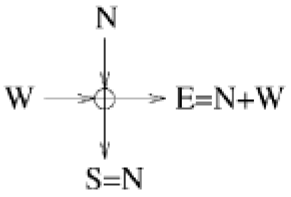
$x_3 = 9 \quad Y_3 = \{7,8,10,13,14\}$

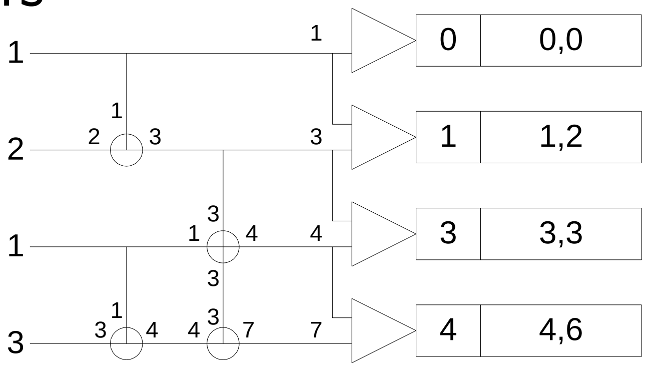


Multicast copy networks

- RAN, DAE
  - Encoding process
    - RAN, DAE transform copy numbers into a set of monotonic addresses
  - Decoding process
    - TNT determines the real destinations of copies
  - RAN, DAE
    - (N/2)Log$_2$N adders
    - Log$_2$N stages
  - An example

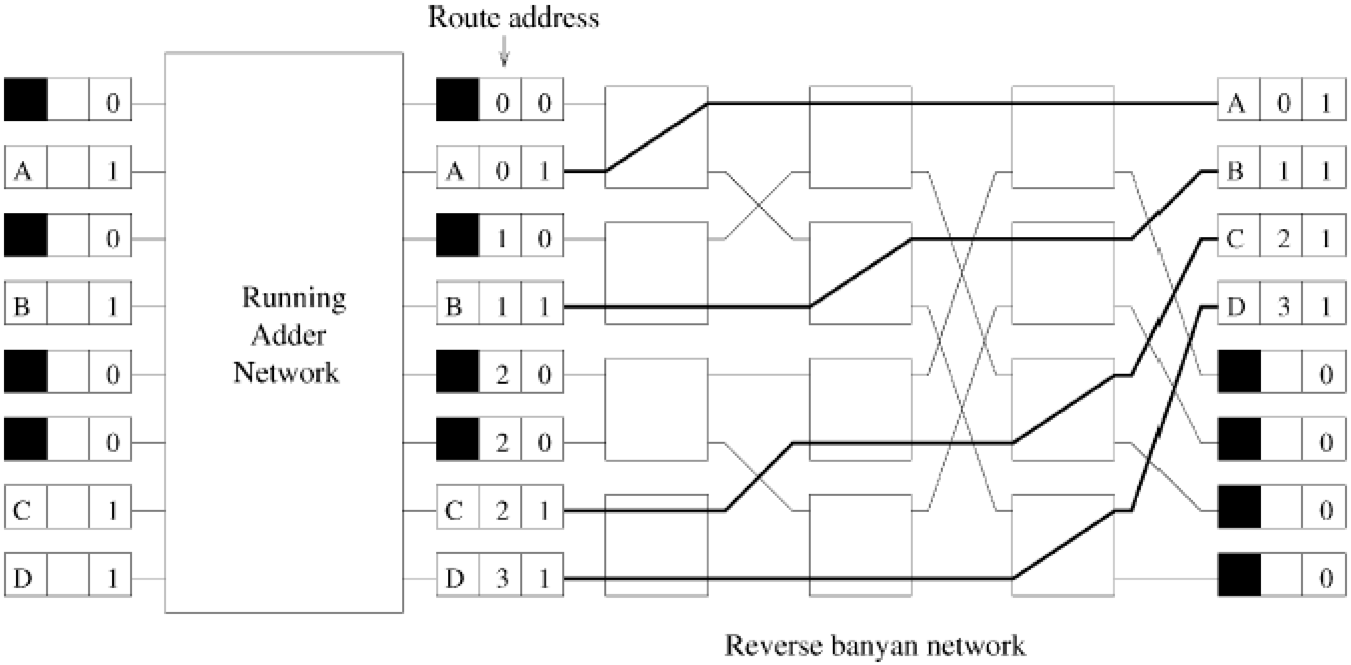






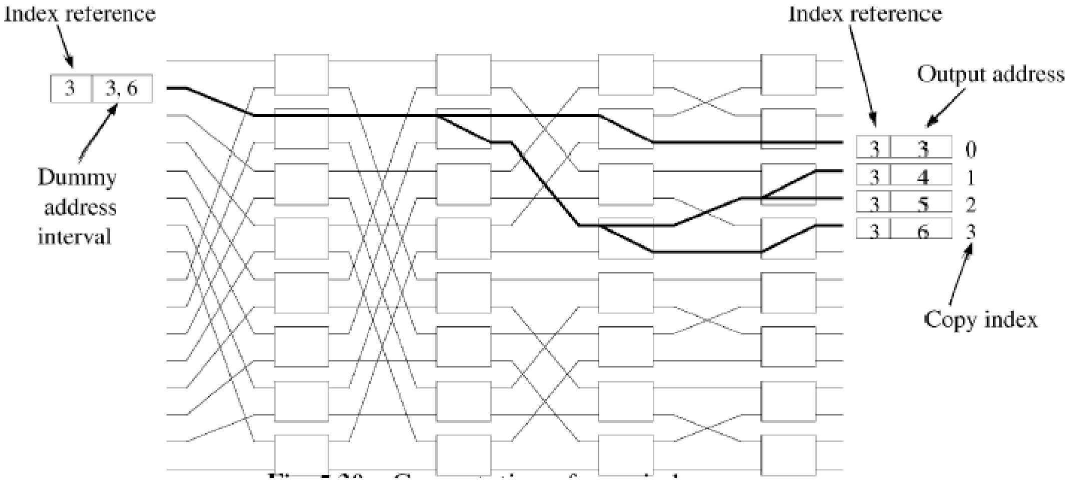
Multicast copy networks

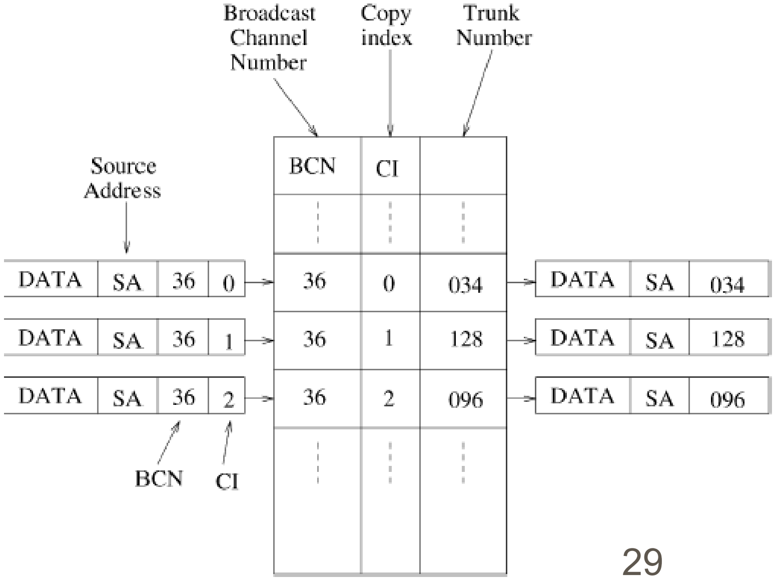
- Concentration
  - Eliminate idle inputs between active outputs
  - Must be done before broadcast banyan network
    - Berfor RAN, or
    - After DAE
  - A reverse banyan network (RBN) can be used



Multicast copy networks

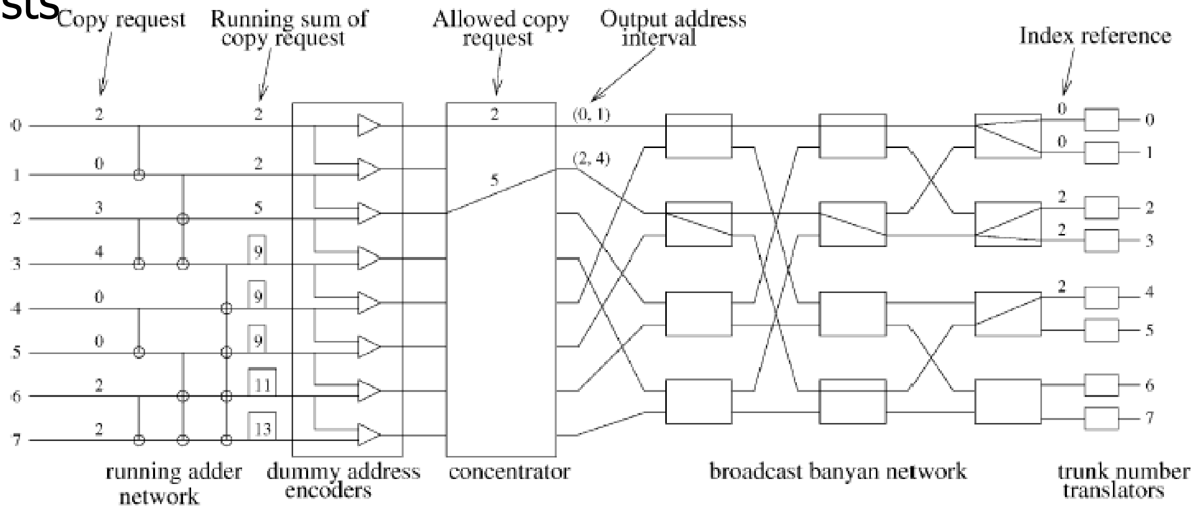
- Decoding process
  - When cell leaves broadcast banyan network,
    - The interval in the header is only one address
    - This address = min = max
    - Copy index = this address – index reference
    - TNT assigns the actual address to each copy
      - A simple table lookup
      - Search key: BCN and CI



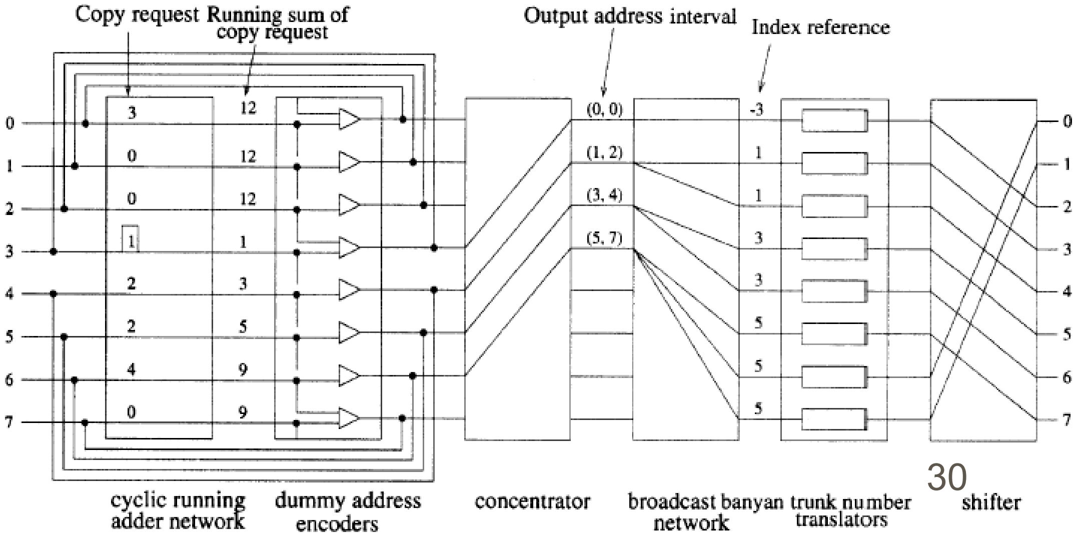


Multicast copy networks

- Overflow
  - Copy network may not be able to do all copy requests
  - An example
  - Overflow problems
    - Performance
    - Unfairness

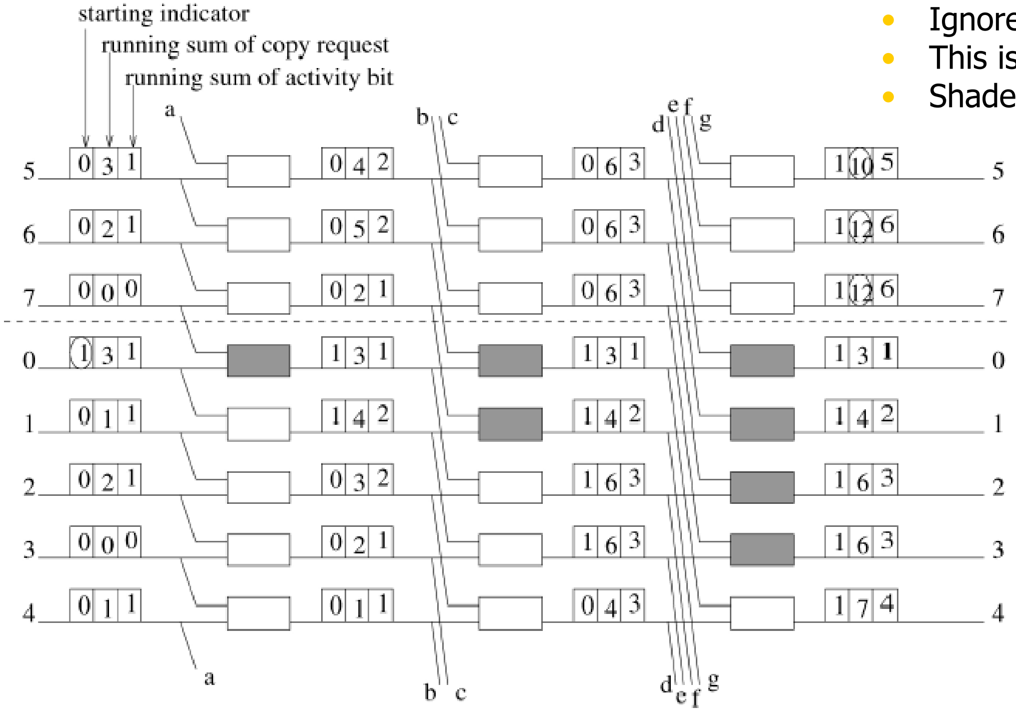


- Unfairness problem
  - Lower numbered inputs will have less overflow
  - Solution: CRAN instead of RAN
    - Adaptively changes RAN sum starting point



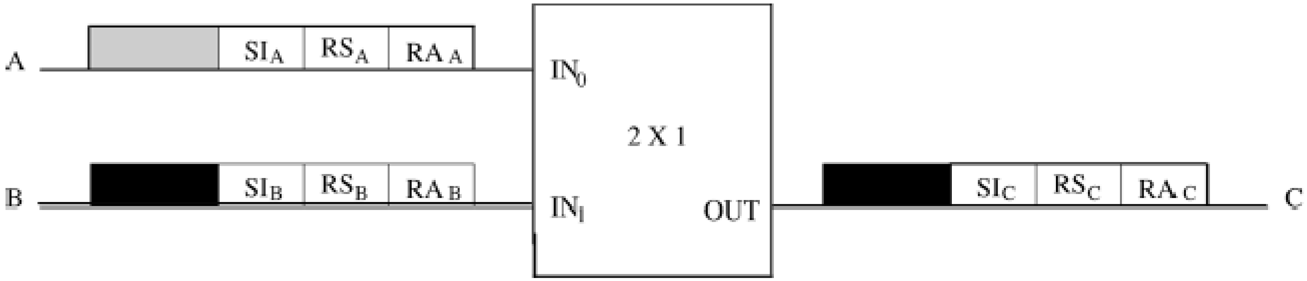
Multicast copy networks

- CRAN
  - Cell header fields
    - Starting indicator (SI)
    - Running sum (RS)
    - Routing address (RA)
  - Initial values
    - SI: nonzero only for the starting point
    - RS: the number of copies
    - RA: 1 if port is active, otherwise 0
  - At output, as the result:
    - RA has the running sum over activity bits
  - A node receiving SI=1
    - Ignores its links
    - This is propagated
    - Shaded node in the figure



Multicast copy networks

- CRAN
  - Header modification in a node



  - The next starting point
    - No overflow → same as the previous point
    - Overflow → the first port facing the overflow
  - RS updating

$$SI_0 = \begin{cases} 1 & \text{if } RS_{N-1} \leq N, \\ 0 & \text{otherwise.} \end{cases}$$

$$SI_i = \begin{cases} 1 & \text{if } RS_{i-1} \leq N \text{ and } RS_i > N, \\ 0 & \text{otherwise,,} \end{cases}$$

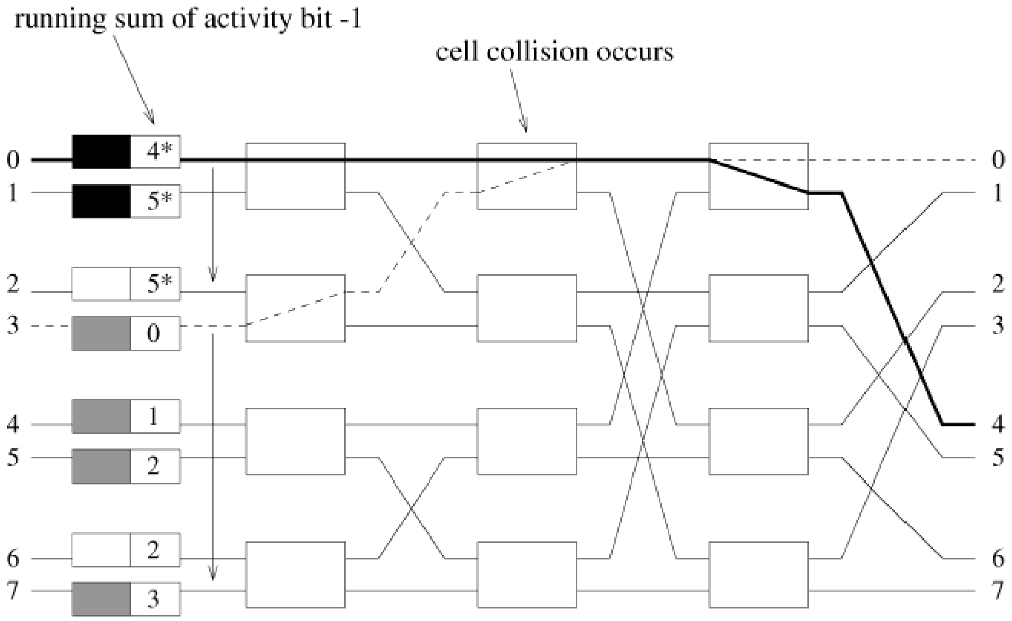
  - SCN: starting copy number
    - Is sent back via feedback paths to input ports
    - So they know how many copies to serve

$$SCN_0 = RS_0,$$

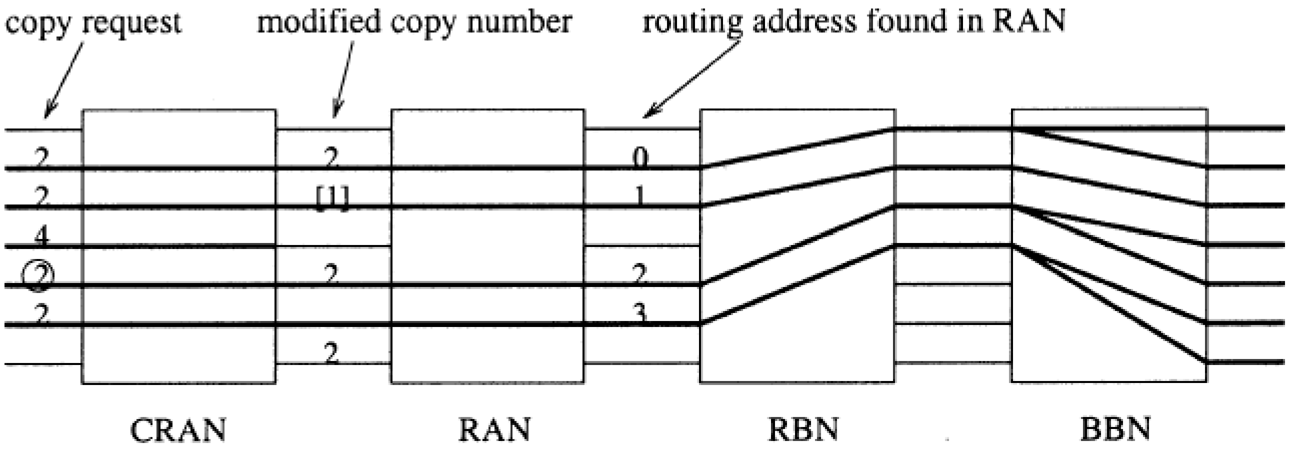
$$SCN_i = \begin{cases} \min(N - RS_{i-1}, RS_i - RS_{i-1}) & \text{if } RS_{i-1} < N \\ 0 & \text{otherwise} \end{cases}$$

Multicast copy networks

- Concentration problem
  - The starting point in CRAN may not be port 0
  - Internal collisions may occur in reverse banyan network (RBN)



  - Solution: an additional RAN before RBN



copy request
modified copy number
routing address found in RAN
2
2
4
2
2
2
[1]
2
2
0
1
2
3
CRAN
RAN
RBN
BBN