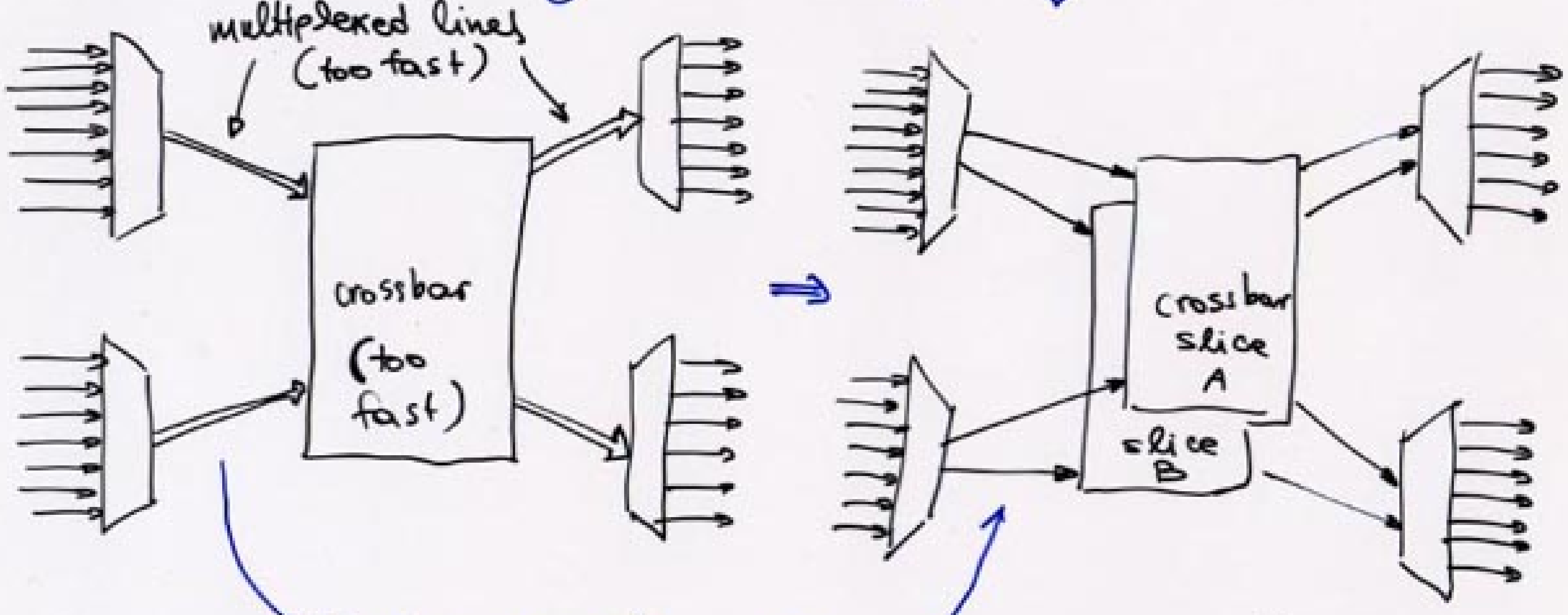
5.1 Byte-Slicing, Inverse Multiplexing

- How to increase the throughput of a Link?
  - wide link #1 – bit-slicing, byte-slicing:
    each packet passes through all wires – time switching
  - wide link #2 – packet (flow) slicing, Inverse Multiplexing:
    each packet passes through some (of the) wire(s) – space switching
- Duality of Time Switching and Space Switching
  - TSI changes the position of information in time, on one (wide) link
  - Crossbar changes the position in space, among a set of wires/links
  - TST switching is dual of 3-stage fabrics made of smaller crossbars

How to build VERY-VERY large Switches:

- increase crossbar size?... too expensive (N^2)
- increase multiplexing?... too fast, but

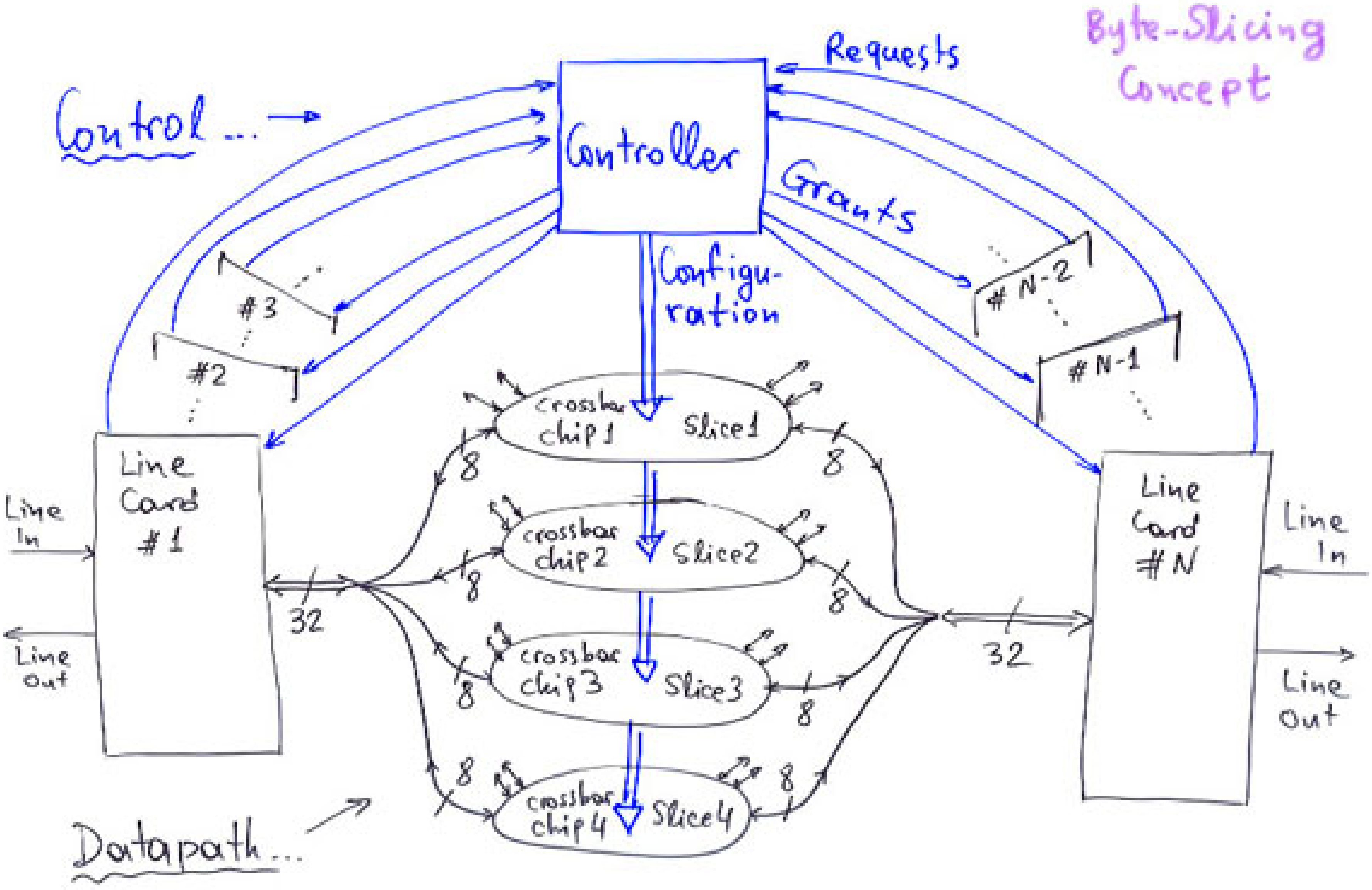


H/W terminology ... .. "bit slicing"

Switching terminology ... "inverse multiplexing"

... "TSSST" switch

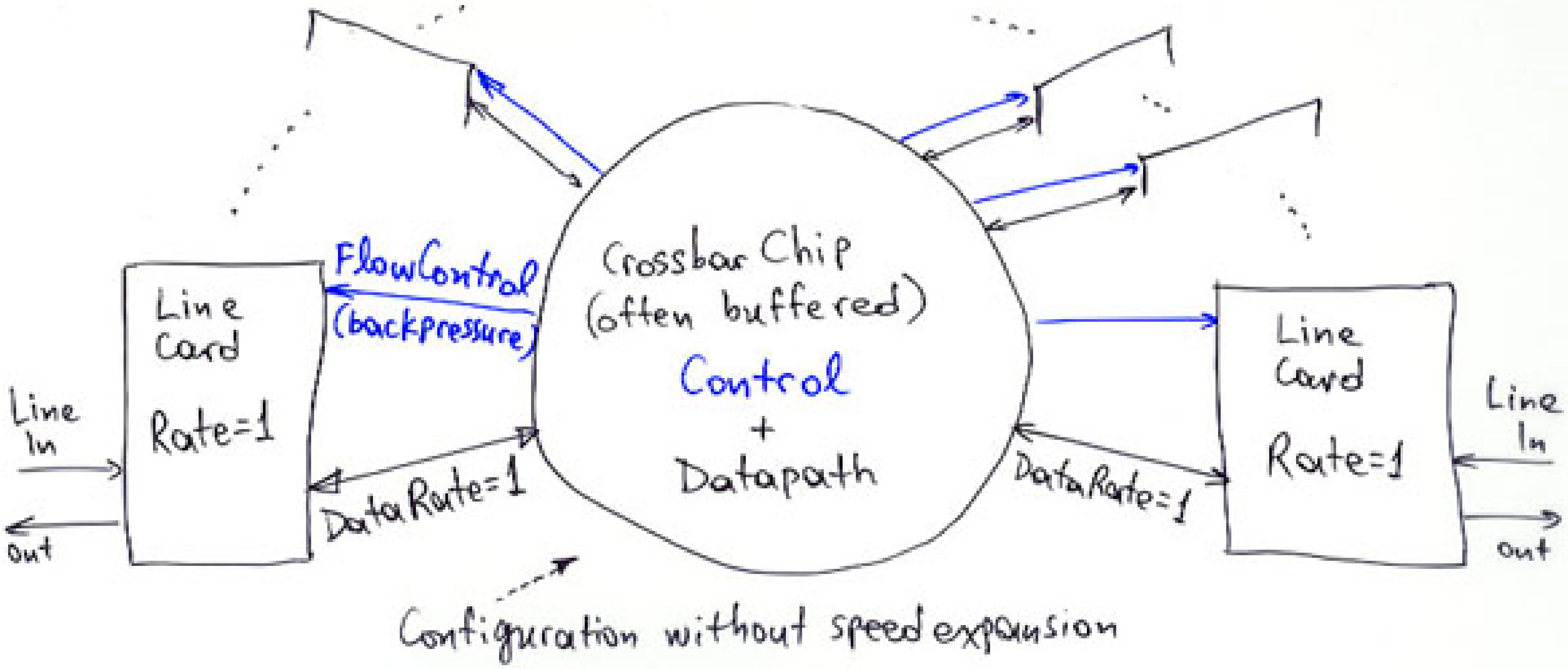
- bit slice?
- byte slice?
- packet slice?
- connection slice?
- flow slice?



McKeown e.a.: “Tiny Tera: a Packet Switch Core”, IEEE Micro, Jan-Feb 1997

Version2: single-chip Control + Datapath with Speed Expansion Option

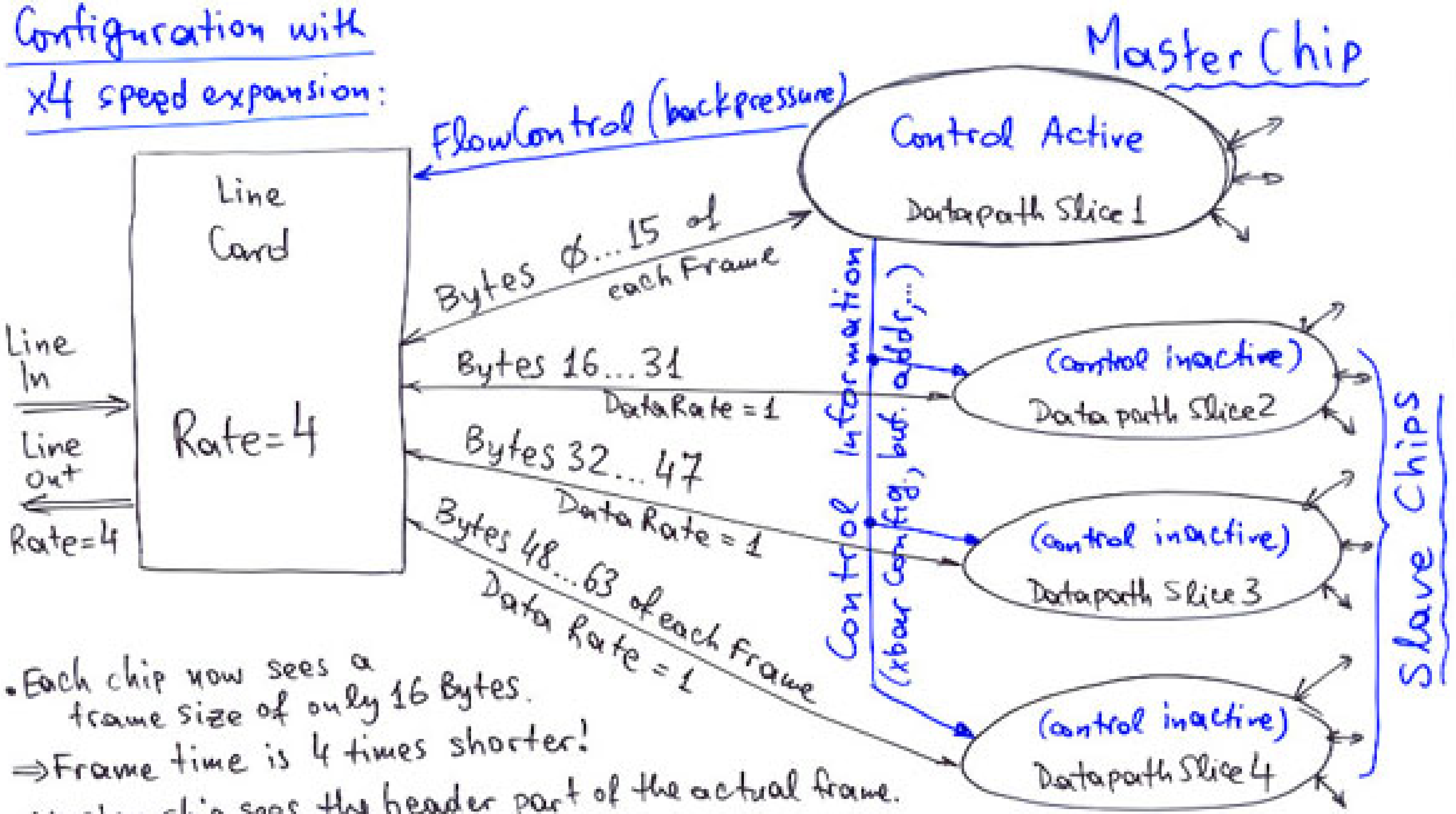
Popular style in commercial crossbar chip products.
Expansion factors usually offered: x2, x4 – rarely x8.



Line Card
Rate=1
Line In
Out
Flow Control (backpressure)
Data Rate=1
Crossbar Chip (often buffered)
Control + Datapath
Data Rate=1
Line Card
Rate=1
Line In
out
Configuration without speed expansion

Example: Frame Size (or Cell Size, or Segment Size) = 64 Bytes

Configuration with x4 speed expansion:

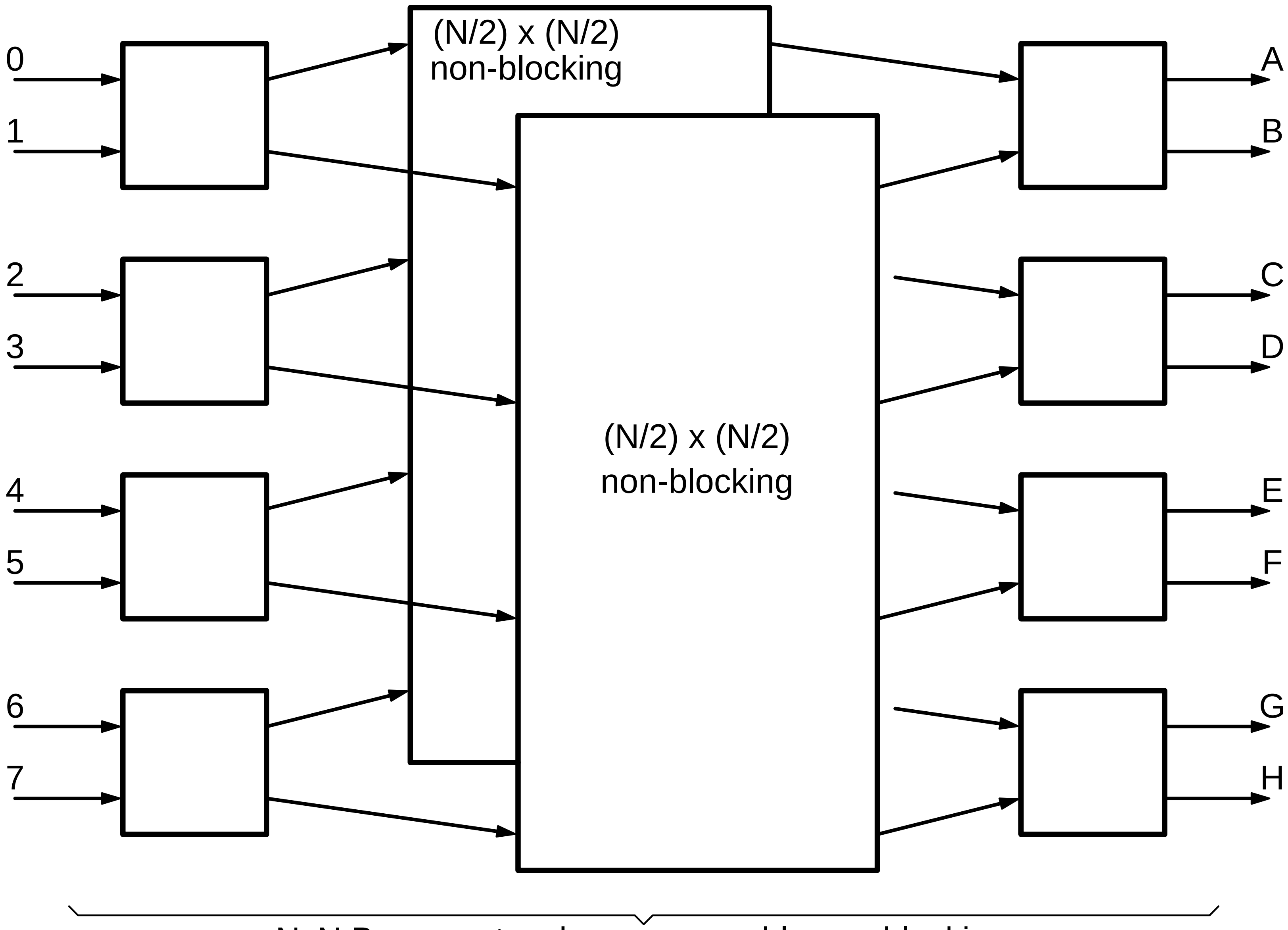


- Each chip now sees a frame size of only 16 Bytes.

⇒ Frame time is 4 times shorter!

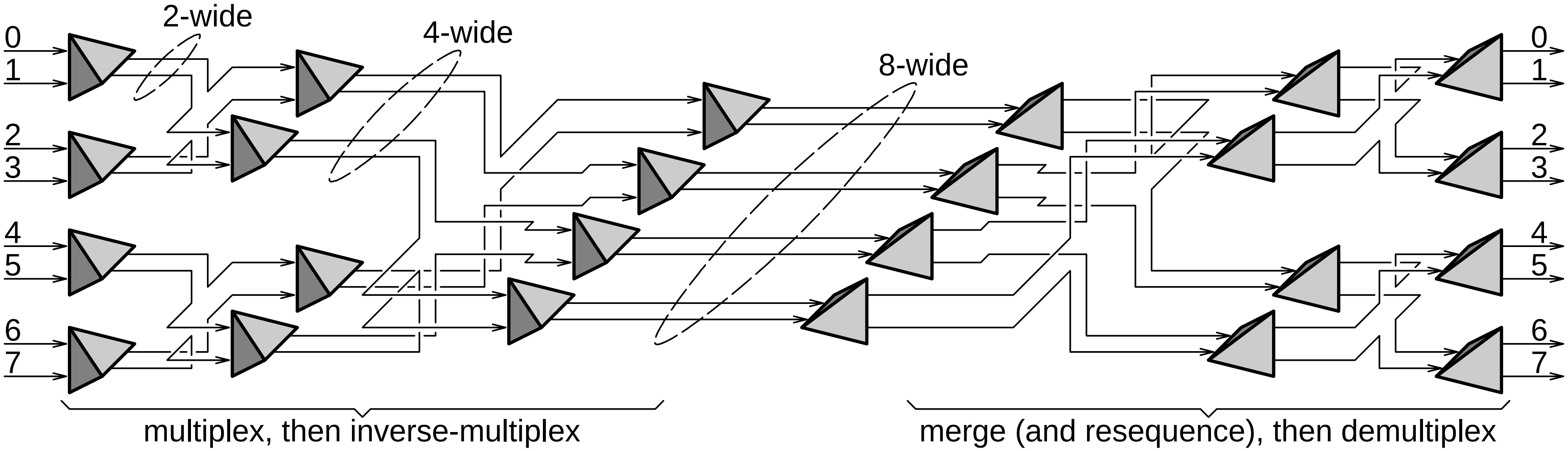
- Master chip sees the header part of the actual frame.
- Slave Frame transfer time and Slave Slice operation are offset in time by a fixed lag relative to the Master Frame transfer time and Master Slice operation - enough for the control information to reach the Slave Slices.

Benes Network: Recursive Definition

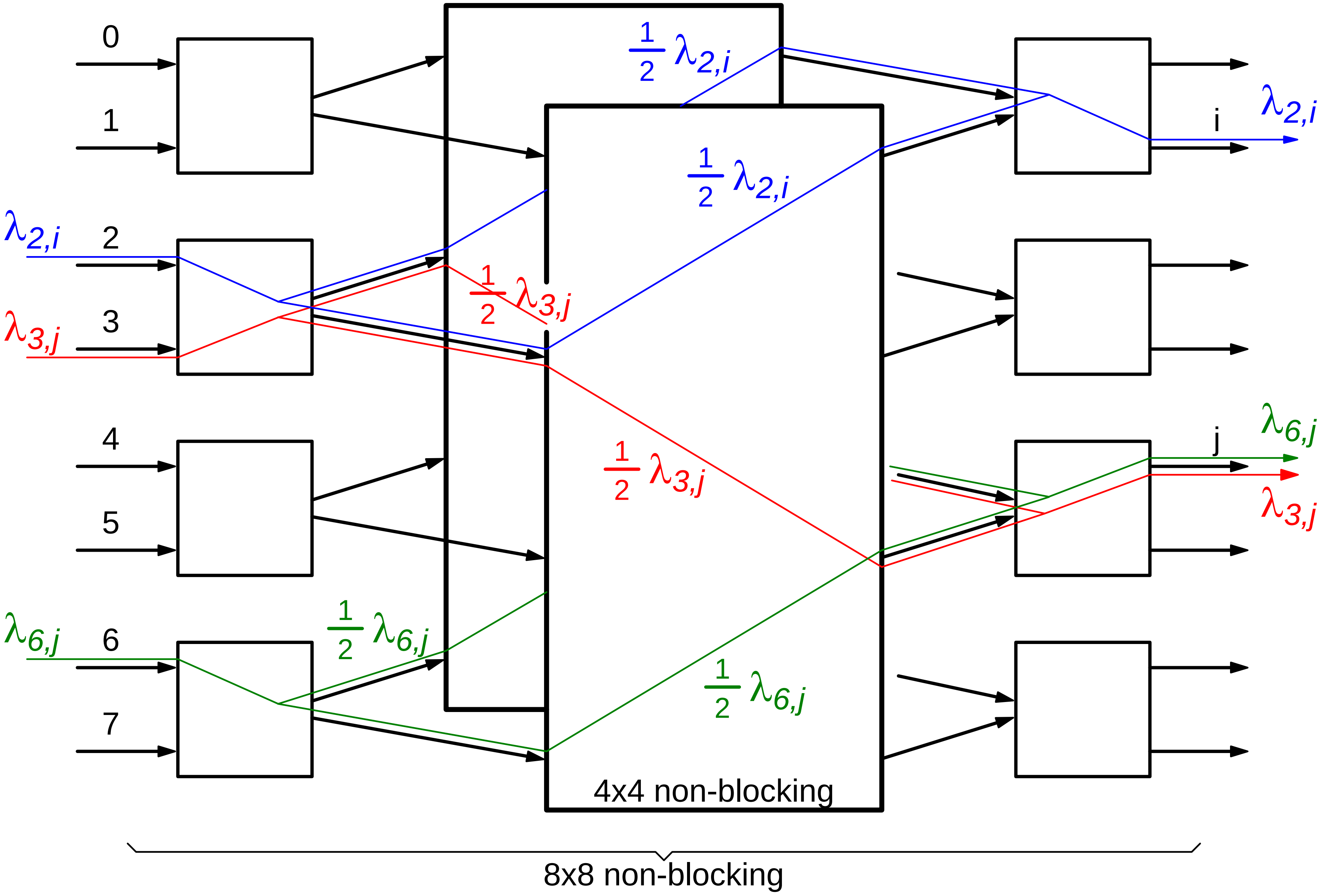


NxN Benes network: rearrangeably non-blocking

Conceptual View of 8x8 Benes using Inverse Multiplexing


2-wide
4-wide
8-wide
0
1
2
3
4
5
6
7
0
1
2
3
4
5
6
7
multiplex, then inverse-multiplex
merge (and resequence), then demultiplex

Benes Network with Inverse Multiplexing for Packet Switching



0
1
$\lambda_{2,i}$
2
$\lambda_{3,j}$
3
4
5
$\lambda_{6,j}$
6
7
$\frac{1}{2}\lambda_{2,i}$
$\frac{1}{2}\lambda_{2,i}$
$\frac{1}{2}\lambda_{3,j}$
$\frac{1}{2}\lambda_{3,j}$
$\frac{1}{2}\lambda_{6,j}$
$\frac{1}{2}\lambda_{6,j}$
i
$\lambda_{2,i}$
j
$\lambda_{6,j}$
$\lambda_{3,j}$
4x4 non-blocking
8x8 non-blocking

Inverse Multiplexing Implementations

- Per-Flow Splitting: poor man's approximation
  - path selection based on hash function of flow ID
  - all packets of a given flow through same path   in-order delivery
  - poor load balancing when small number of flows
- Per-Packet Splitting: full solution
  - distribute individual packets evenly among the multiple paths
  - potentially out-of-order delivery even for packets of a same flow

    ⇒ most applications require a resequencer
  - must ensure even distribution at outputs too –not just at inputs

    ⇒ distribute evenly per flow –flows defined as input-output pairs
  - some architectures perform "adaptive routing"…