3. Time Switching, Multi-Queue Memories, Shared Buffers, Output Queueing Family

Manolis Katevenis

CS-534 – Univ. of Crete and FORTH, Greece

http://archvlsi.ics.forth.gr/~kateveni/534

3.1 TDM, Time Switching, Cut-Through

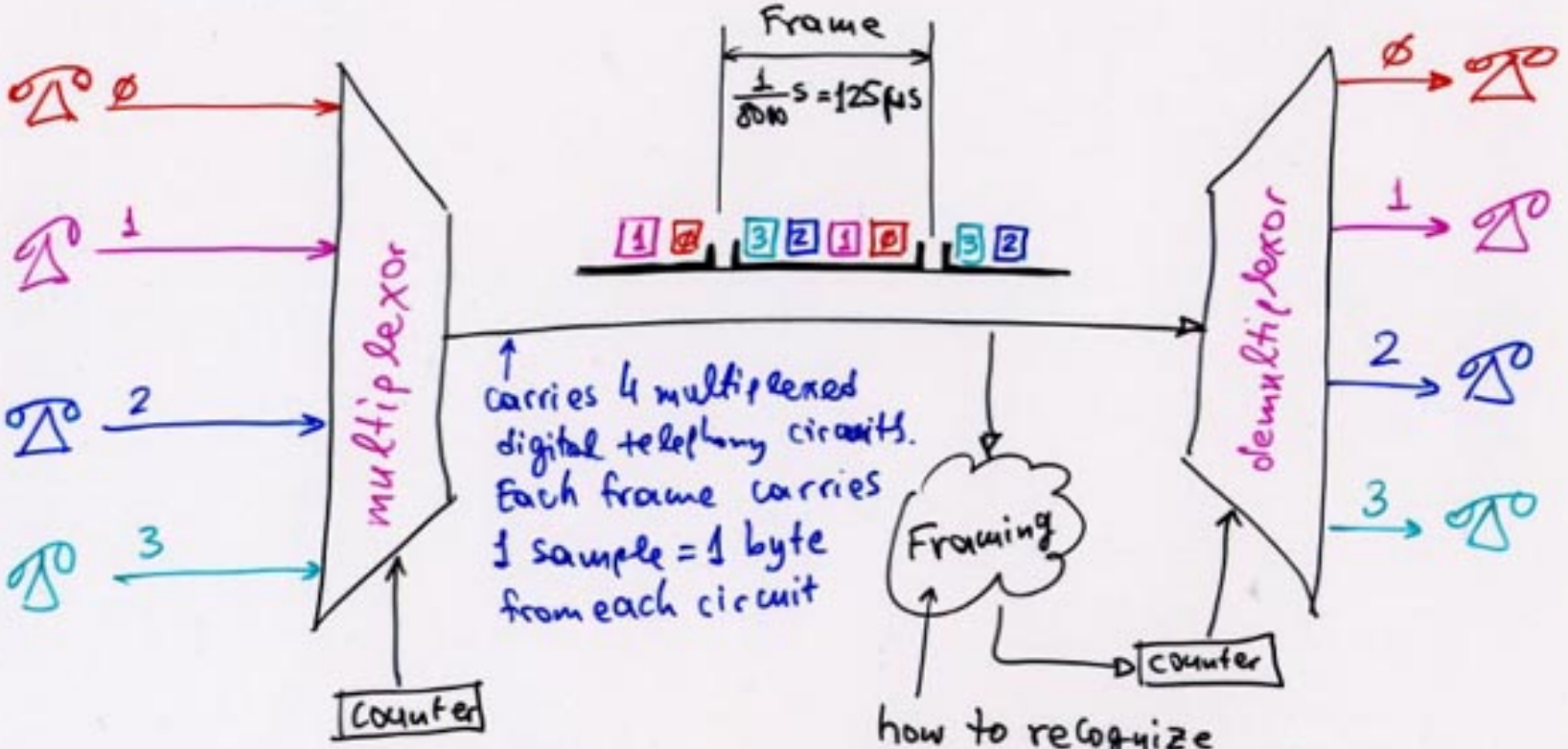
Table of Contents:

- Digital Telephony, Time-Division Multiplexing (TDM)
- Time Switching, Time-Slot Interchanges (TSI)
- “First Generation” Switches
  - general-purpose computers, using a shared I/O bus
- “Second Generation” Switches
  - using DMA from input to output line cards
- Store-and-Forward versus Cut-Through

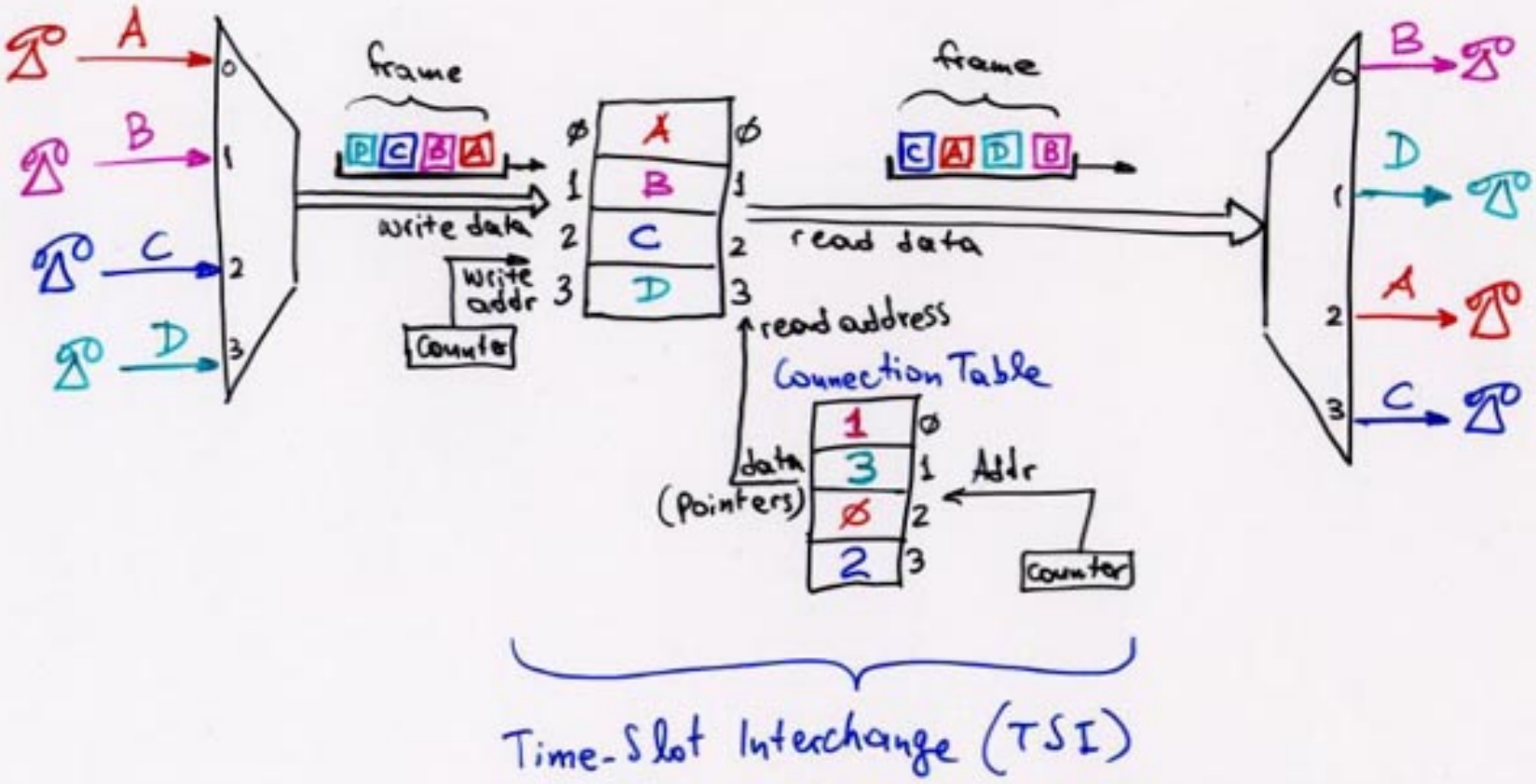
Digital Telephony

telephone quality voice $\lesssim$ 3.5 KHz $\Rightarrow$ $\approx$ 8000 $\frac{\text{samples}}{\text{second}}$

8000 samples/s x 8 bits/sample = 64,000 bits/s ... one digital telephony circuit

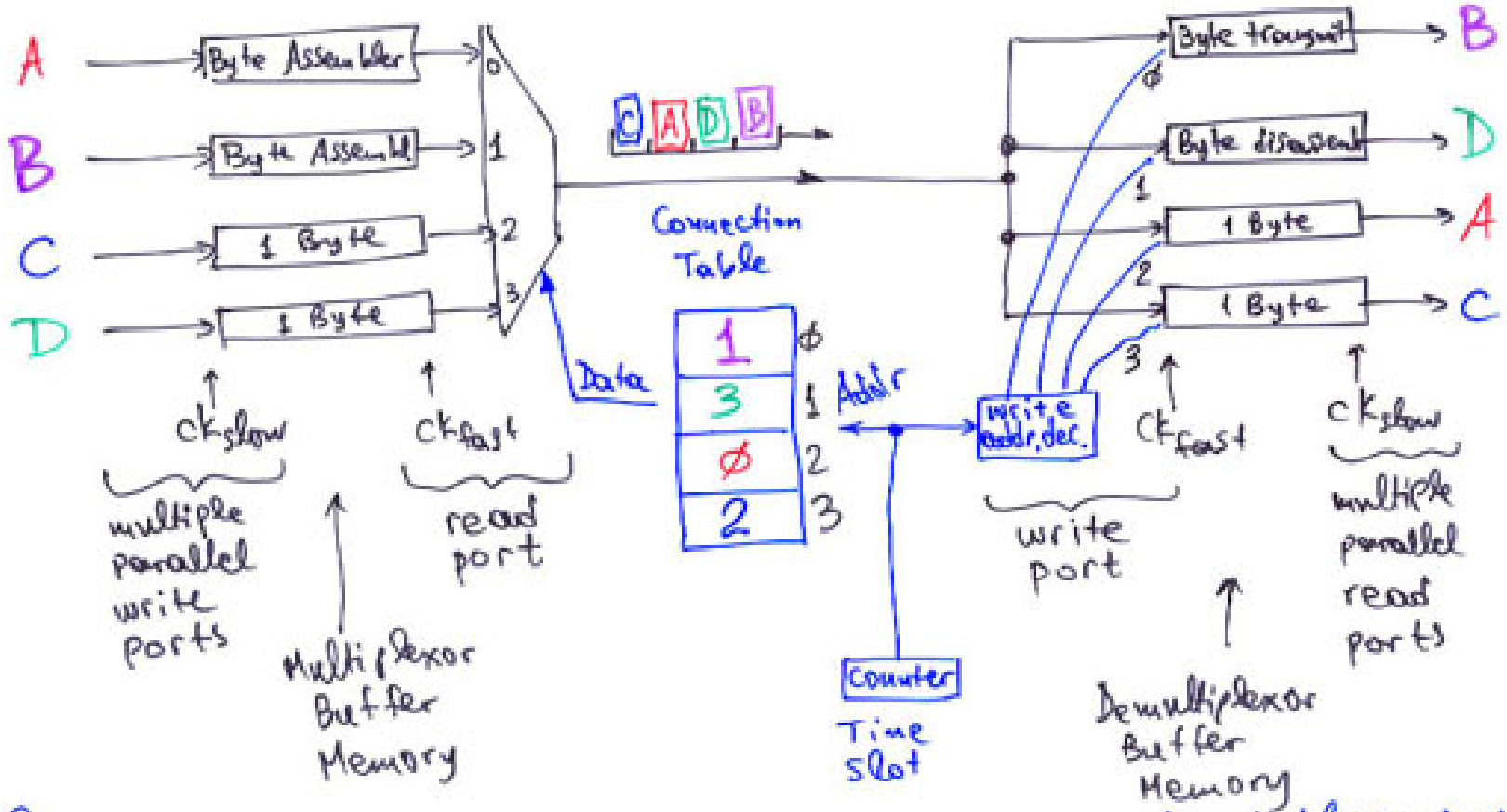


Time-Division Switching



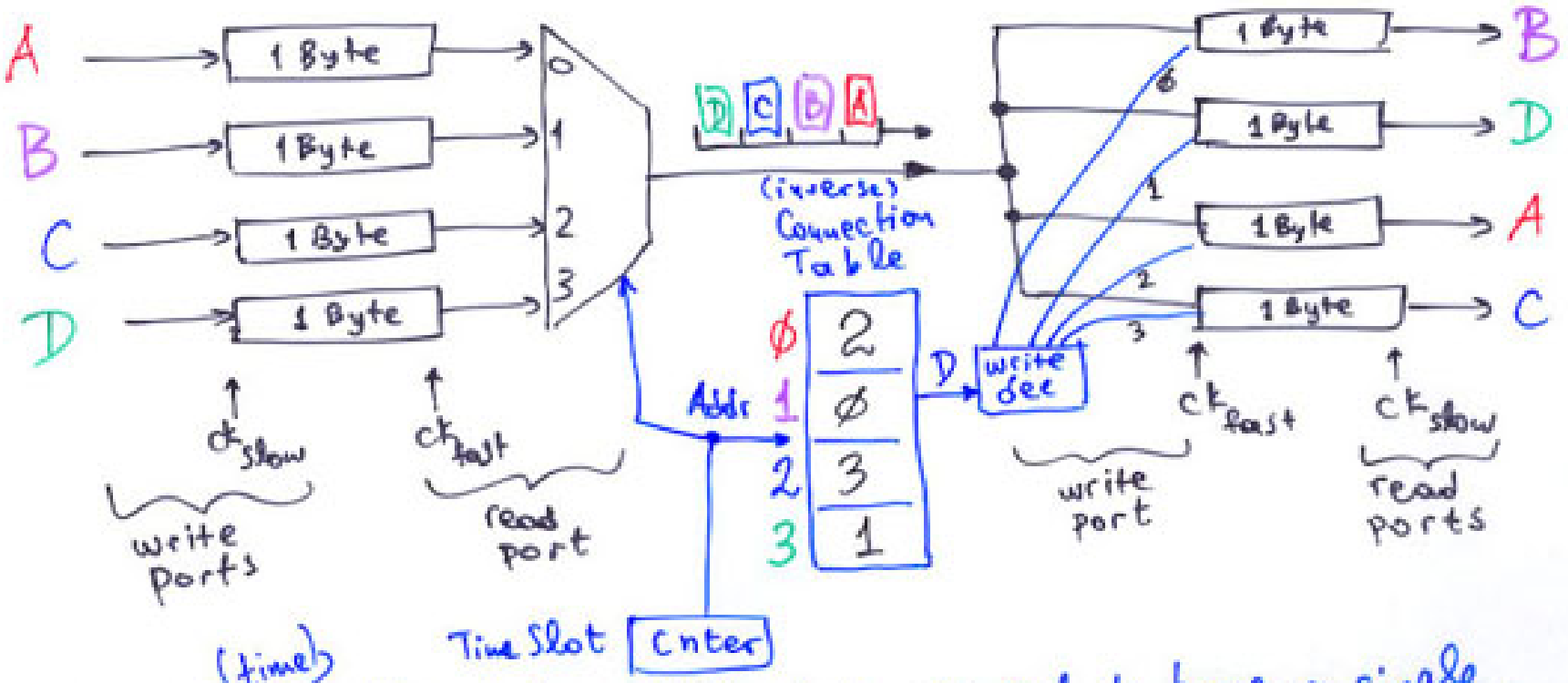
Time-Division Switching:

When each external link carries a single connection, the time-slot interchange can be merged with the multiplexor (or with...



(Could also use TDM at bit granularity, with the connection table running at 8 times higher clock rate)

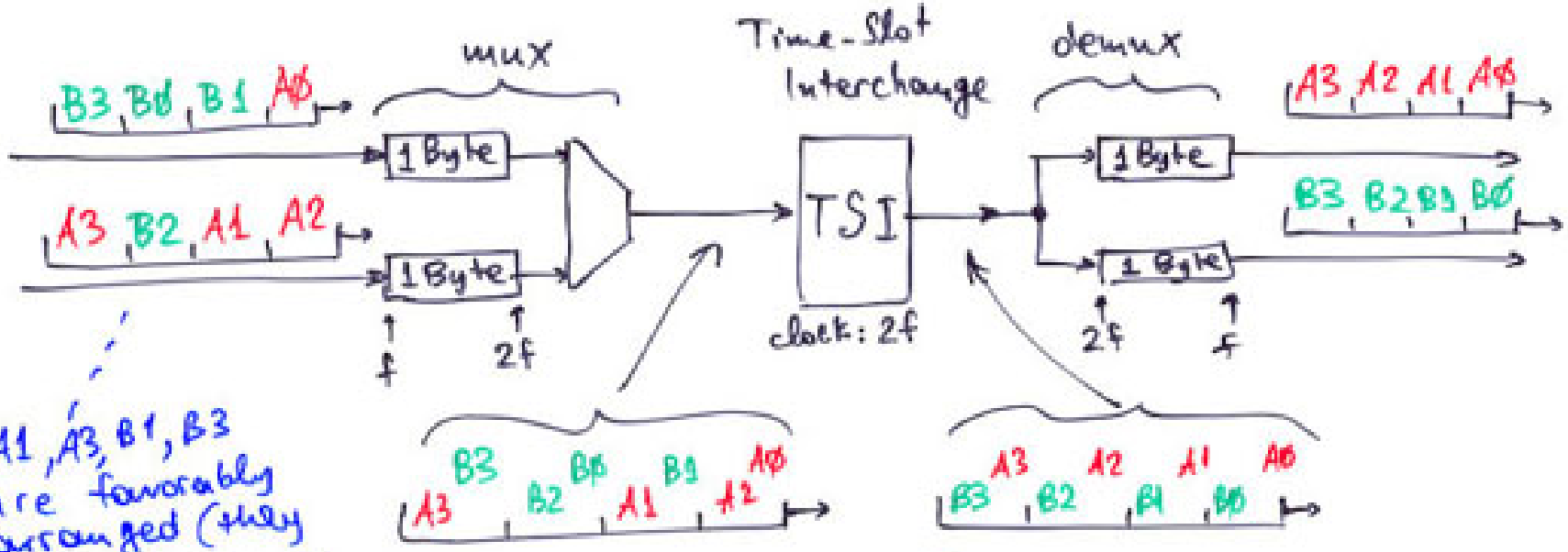
Time-Division Switching with external links carrying a single connection each:
Alternatively, the time-slot interchange can be merged with the demultiplexer



Note: for (time) switching to be possible, it is essential to have a single, shared line going from all inputs to all outputs.
Every single quantum of information goes through that single common line. The throughput of that line = Σ inp. throughp. = Σ out. throughp.
In this case, $f_{fast\,clock} = n \cdot f_{slow\,clock}$ (n = # ext. lines).

Time-Division Switching: more complex case: multiple connections per external line

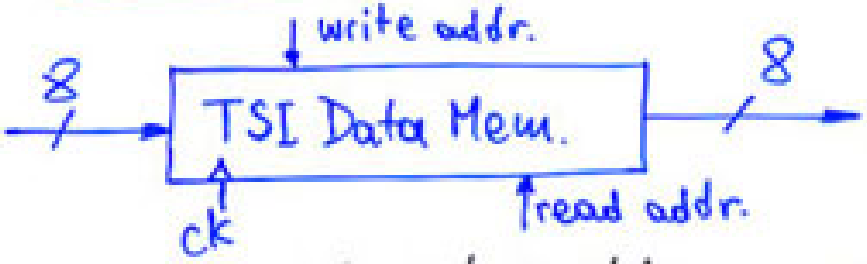
- Mux and Demux need less buffer memory than full frame
- Internal TSI needed with 1 full frame of buf. memory
- Internal TSI cannot be merged w. mux or demux
- Worst-case delay = 1 frame time, again.



A1, A3, B1, B3 are favorably arranged (they arrive on correct timeslot); however, not so for A0, A2, B0, B2, because A0, B0 arrive on same link ⇒ cannot be placed both in correct timeslot.

Discussion: Can I freely rearrange positions of connections inside inp. & output frames?

Byte-by-Byte Time Switching: Throughput Limit ?



Assume 300 MHz 2-port SRAM

$\Rightarrow$ Peak Throughput = $300 \frac{\text{MBytes}}{\text{s}}$ =

= 2.4 Gbit/s = 37,500 × 64Kb/s

(if making an 8×8 switch $\Rightarrow$ each link up to 300 Mb/s ... quite low @ today's standards)

Can we Increase Throughput by Widening the TSI Memory?

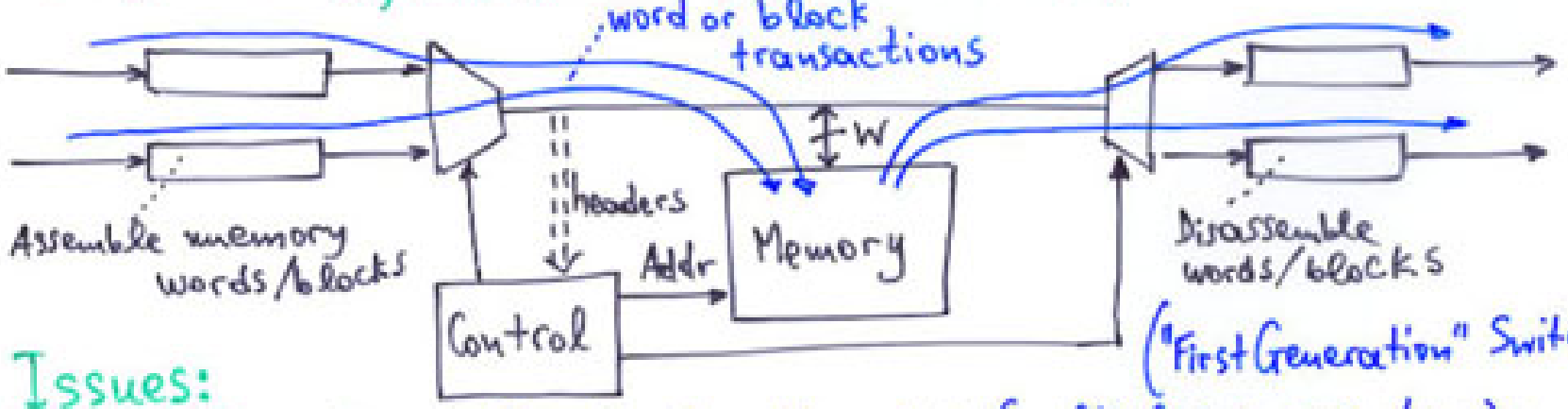
128 bits = 16 Bytes 128 b × 300 MHz = 38.4 Gbits/s

$\Rightarrow$ What is the Multiplexing Quantum ???

- 16 Bytes belonging to a same 64 kbps channel ?:
  - → must wait 16 frames = 16 × 125 μs = 2 ms to collect all these bytes!
  - → buffer size for collection = 2 ms × 38.4 Gb/s = 76.8 Mbits
- 16 Bytes belonging to 16 "adjacent" 64 kbps channels ?:
  - → must switch all 16 channels together:
    where one of them goes, all 16 of them must go!

Time Division Switching:
From Circuit Switching to Packet Switching

...from statically, off-line scheduled, fixed-throughput/channel
to dynamically, demand-driven, on-line scheduled, variable-throughput/flow



("First Generation" Switches)

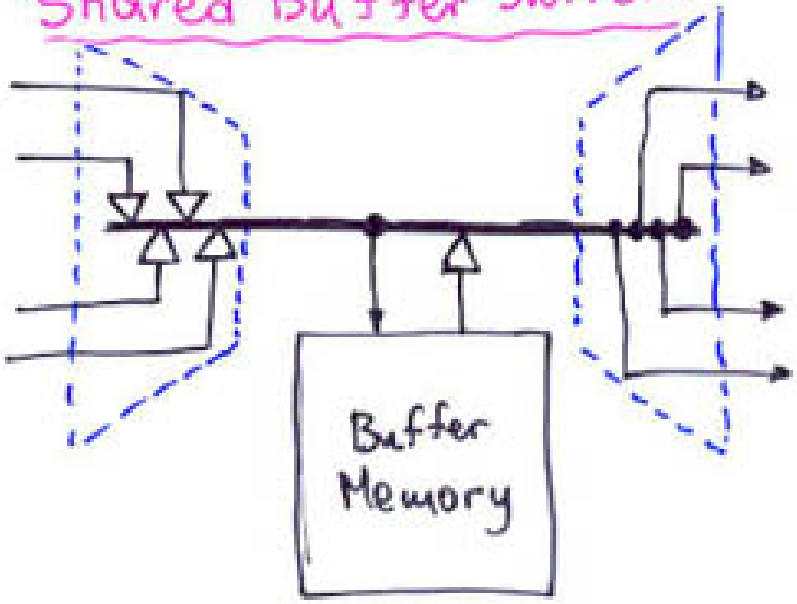
Issues:

- Granularity of transfers In→Mem, Mem→Out (multiplexing quantum):
  fine grain... (narrow word/block): ⊕ small buffers in I/O ⊕ small packets OK! ⊖ narrow mem ⇒ small throughput ⊖ access rate/bus turn-around bound
- Control structure & operation:
  - where to store the words or blocks of each packet? Contiguous? scattered?
    can I mix multiple packets in one block?
  - where to store the packets going to a certain output link?
    (or from a certain input link?) (or of a same QoS class?)

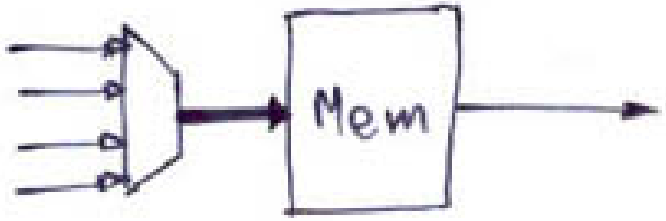
Time Switching:

Where in this memory to store packets? (which address?)

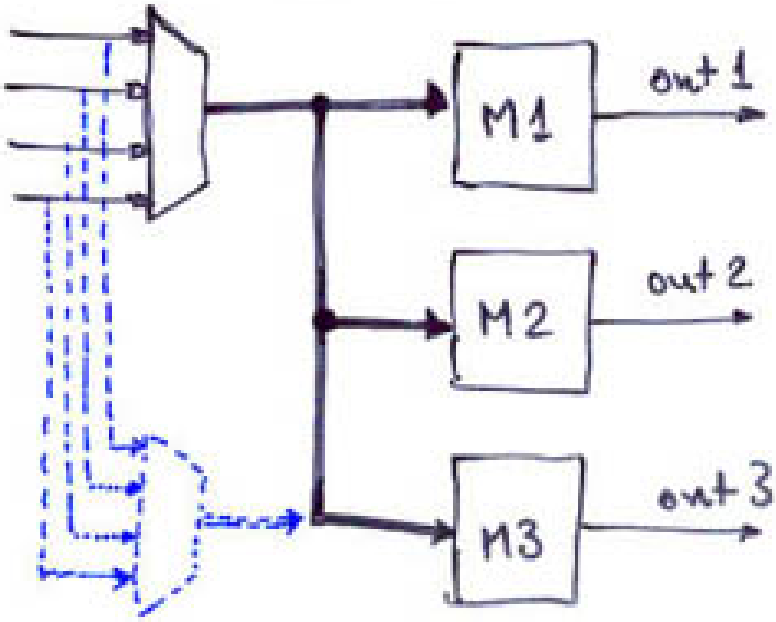
"Shared Buffer Switch":



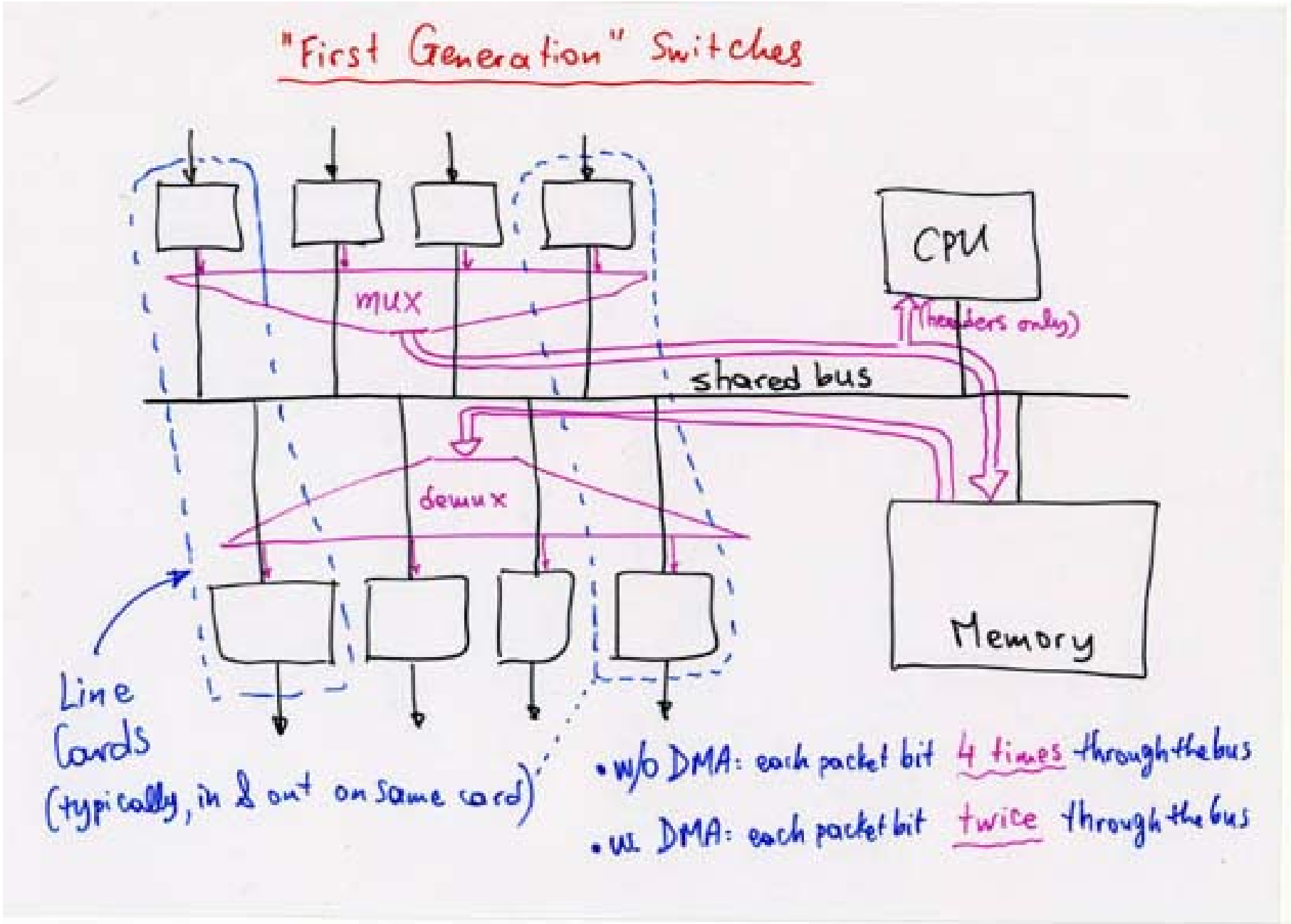
Simplification: n x 1 Switch "Output Queueing"



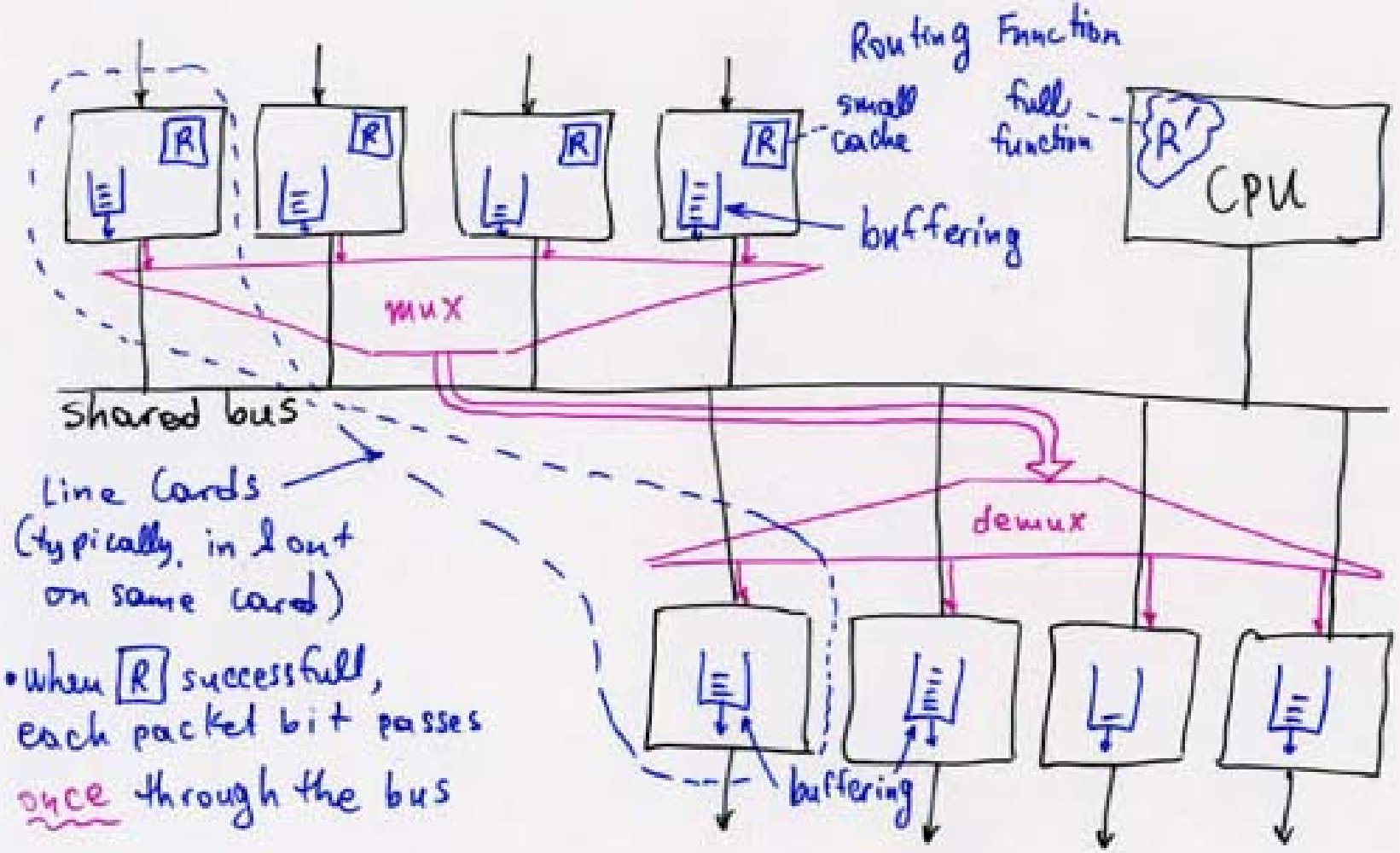
Note: can build the general case out of this:

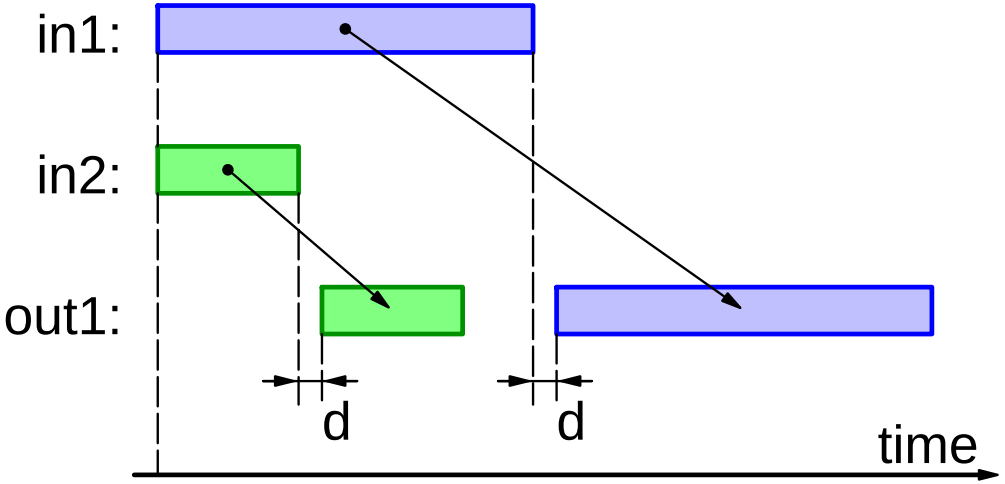


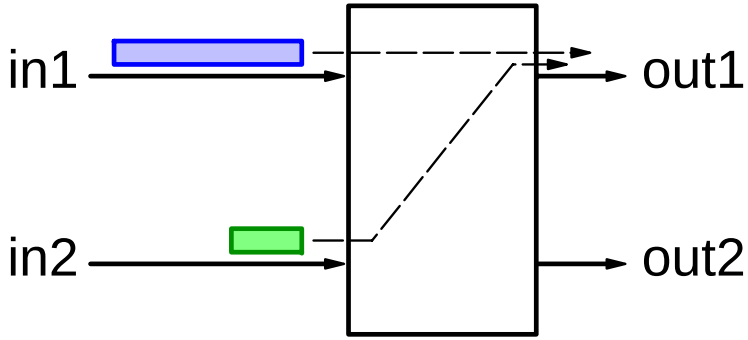
"First Generation" Switches



"Second Generation" Switches



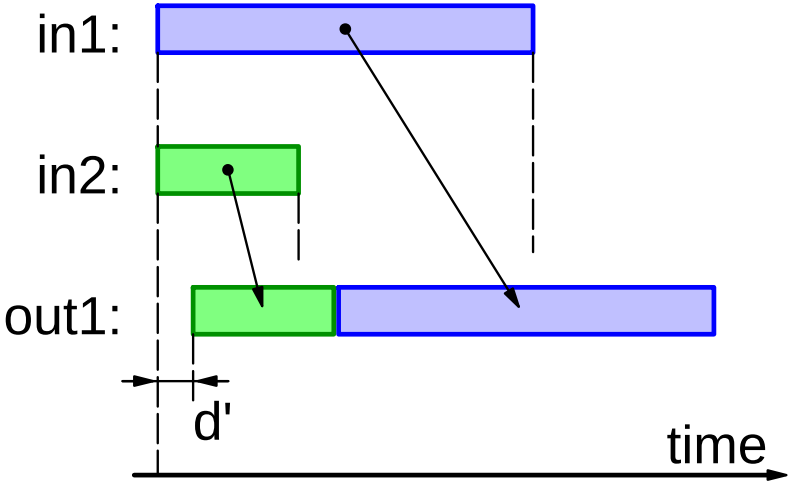




Store-and-Forward

versus

Cut-Through



Cut-through reduces delay.

Hiccup-less cut-through requires:

- hiccup-less incoming packets
- controlled rate difference between input and output