

Physically-Aware Preemptive Virtual Channels for Deadlock-Free AXI Networks-on-Chip

Integrated Systems Laboratory (ETH
Zürich)

Lorenzo Leone

Luca Colagrande

Prof. Dr. Luca Benini

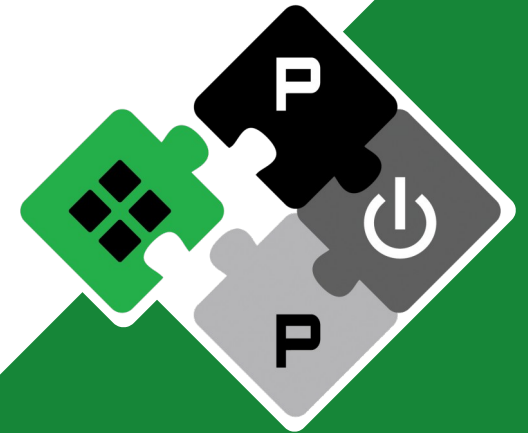
lleone@iis.ee.ethz.ch

colluca@iis.ee.ethz.ch

lbenini@iis.ee.ethz.ch

PULP Platform

Open Source Hardware, the way it should be!



@pulp_platfor



pulp-platform.org



youtube.com/

pulp_platform



Deadlock-Free AXI Networks-on-Chip

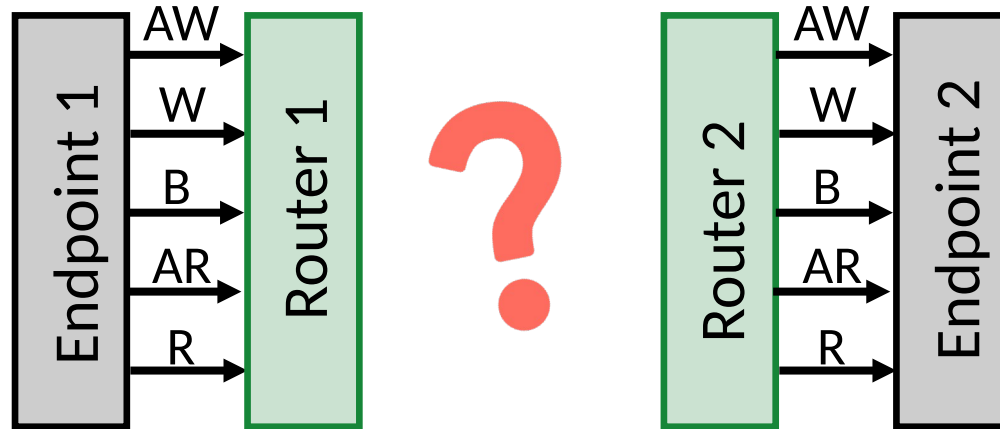


- AXI SoC interface standard:
 - Enables seamless IP integration
 - AXI Endpoints can create protocol dependencies invisible to the network
 - Read/Write channels are independent
 - circuit

arm

AMBA® AXI
Protocol Specification

How can the 5 AXI channels be mapped on the NoC while preserving deadlock freedom?

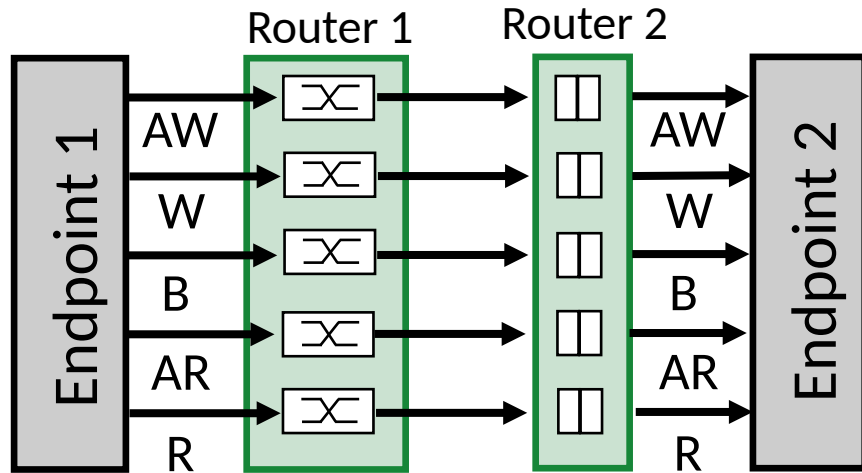


- A write request must be able to provide all write data for that transaction, without dependency on other transactions from that Manager.
 - When a Manager issues a write request and all write data, it must be able to accept all responses for that transaction, without dependency on other transactions from that Manager.
 - When a Manager has issued a read request, it must be able to accept all read data for that transaction, without dependency on other transactions from that Manager.
 - Note that a Manager can rely on read data returning in order from transactions with different IDs. the Manager only needs enough storage for read data from transactions with leading write
 - A Manager is permitted to wait for one transaction to complete before accepting another request, giving credits or sending transfers for another transaction.
 - A Subordinate must not block acceptance of data-less write requests due to transactions with leading write data.
- The protocol does not define any other relationship between the channels.
- The lack of relationship means, for example, that the write data can appear at an interface before the write request for the transaction. This can occur if the write request channel contains more register stages than the write data channel. Similarly, the write data might appear in the same cycle as the request.
- When the interconnect is required to determine the destination address space or Subordinate space, it must realign the request and write data. This realignment is required to assure that the write data is signaled as being valid only to the Subordinate for which it is destined.

The Wiring Complexity Trade-off

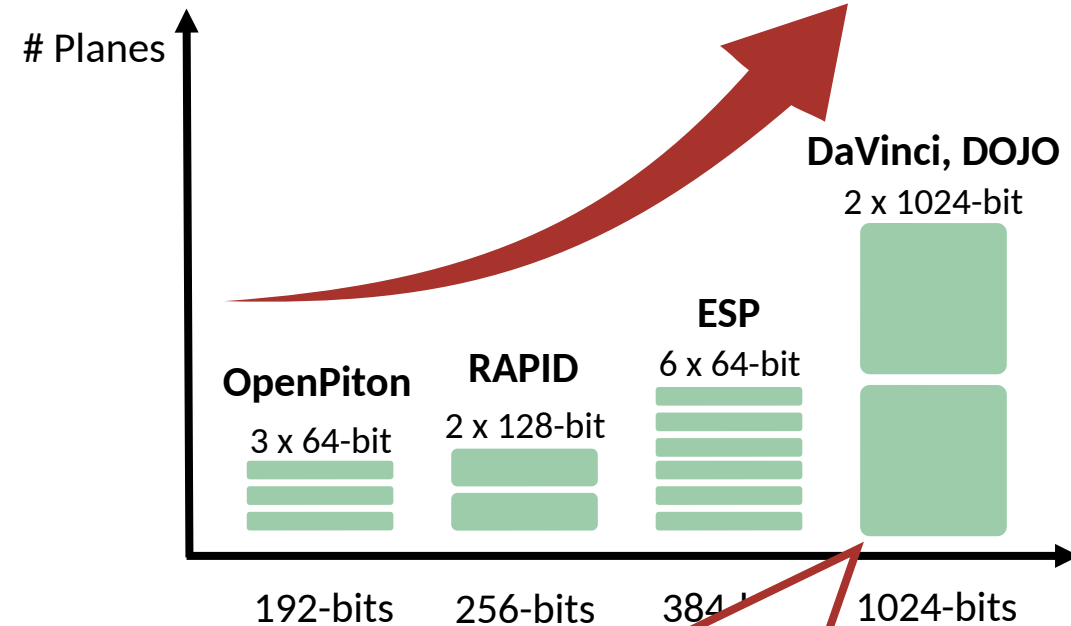


Multiplane



- Each AXI channel gets its own physical link
- Simple router microarchitecture
- **Cost: High wiring cost**

Accelerator-centric systems

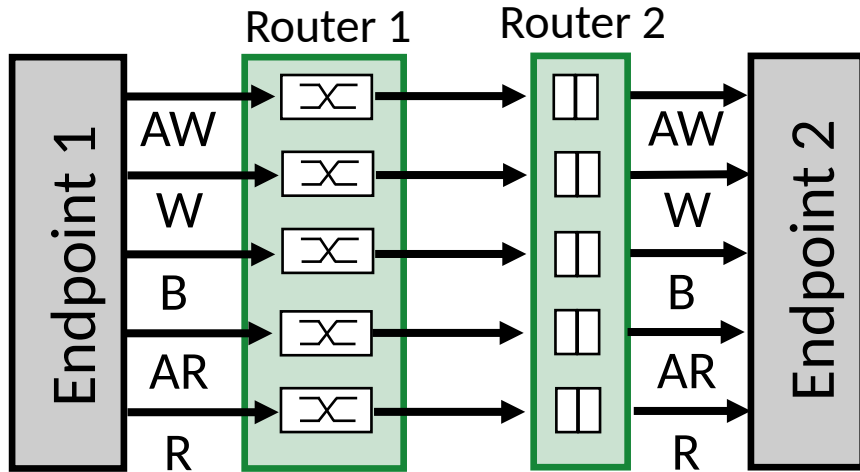


As link get **wider** physical duplication gets more and more **expensive**

The Wiring Complexity Trade-off

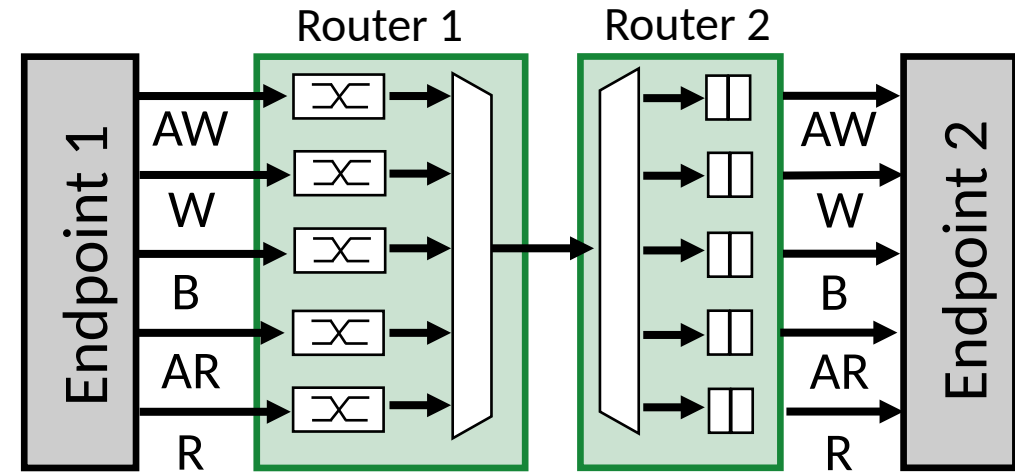


Multiplane



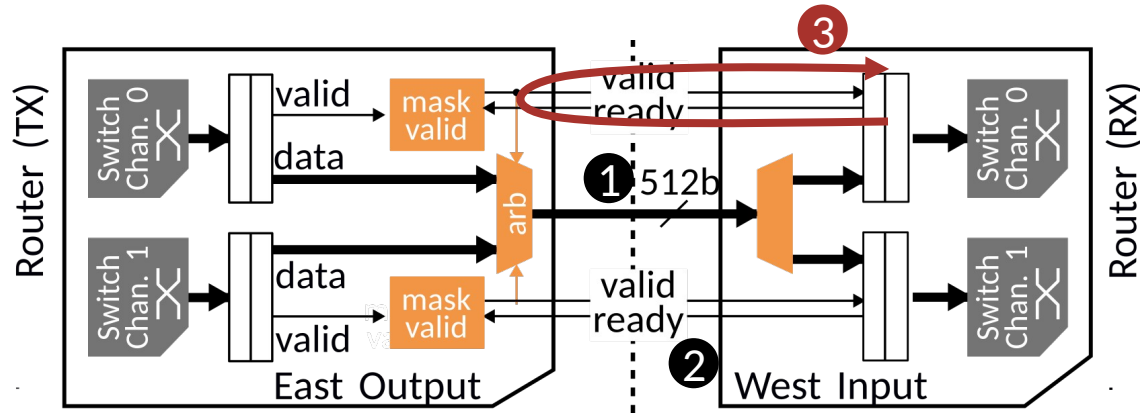
- Each AXI channel gets its own physical link
- Simple router microarchitecture
- **Cost: High wiring cost**

Lightweight Virtual Channels



- Same physical link is shared
- Traffic classes remain logically separated
- Fixed AXI traffic classes:
 - **No Complex VC allocation**
 - **Simple arbitration**

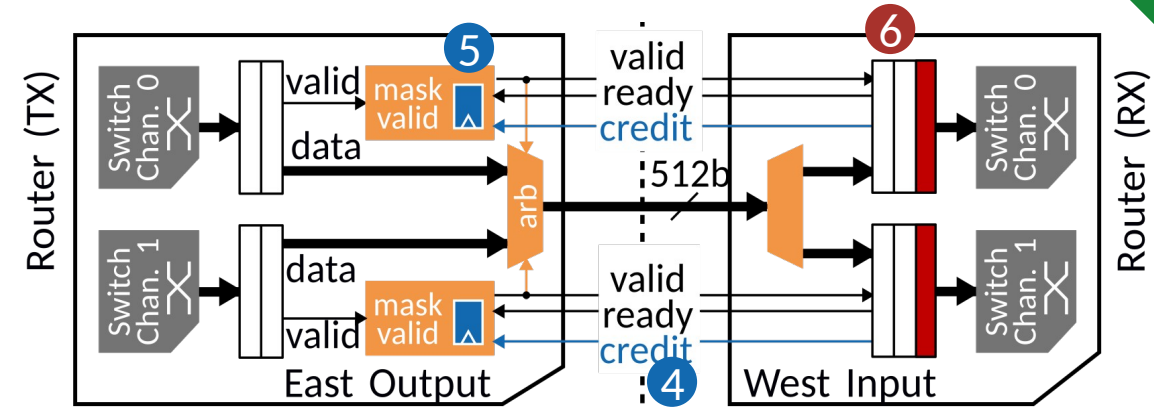
From Naïve VCs to Preemptive VCs



Naïve:

- ① Single 512b physical data link
- ② Ready-valid combinational dependency
- ③ Long tile-to-tile combinational path

Combinational flow control turns link delay into the **critical path**

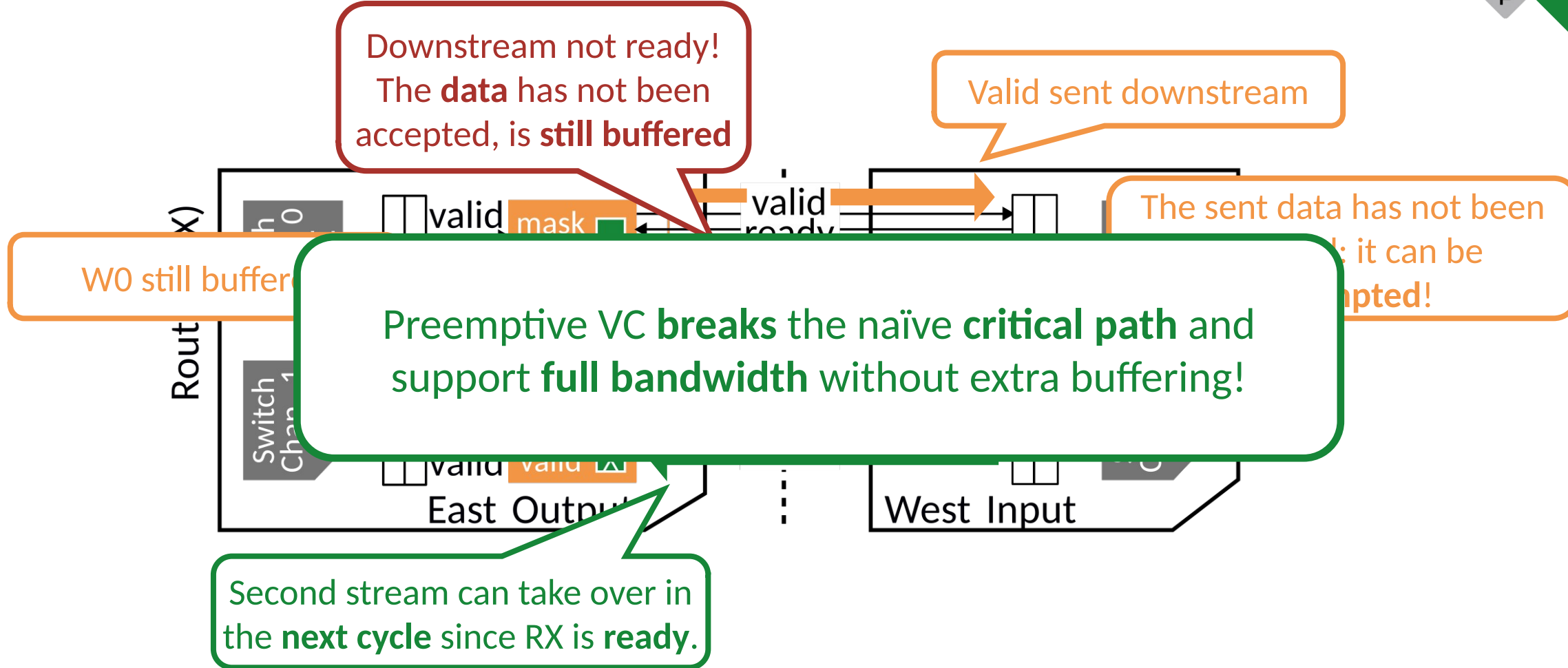


Credit Based:

- ④ Credit signal to indicate downstream flit consumption
- ⑤ Credit counter: cut the naïve critical path
- ⑥ Extra buffering for full bandwidth support

Credit round-trip **latency** requires **extra buffering** for full bandwidth

Preemptive Virtual Channels



Results



- **System level**

- Integrated into a 4x4 multicluster SoC

- **Area-Timing**

- Naïve: **15%** frequency increase over multiplane

- Credit-based: **28%** frequency increase over multiplane

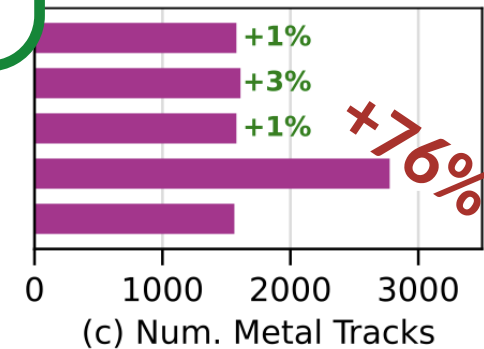
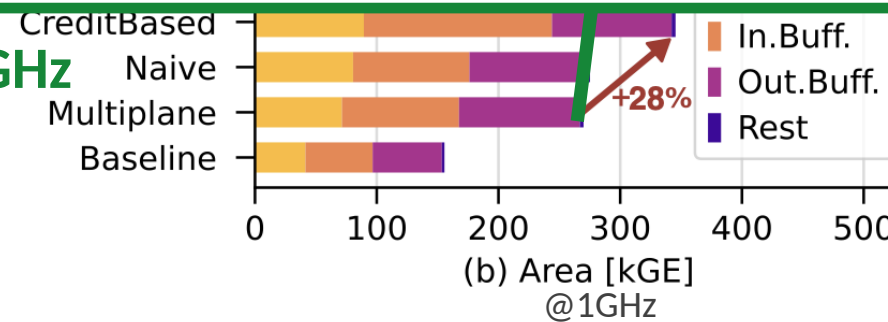
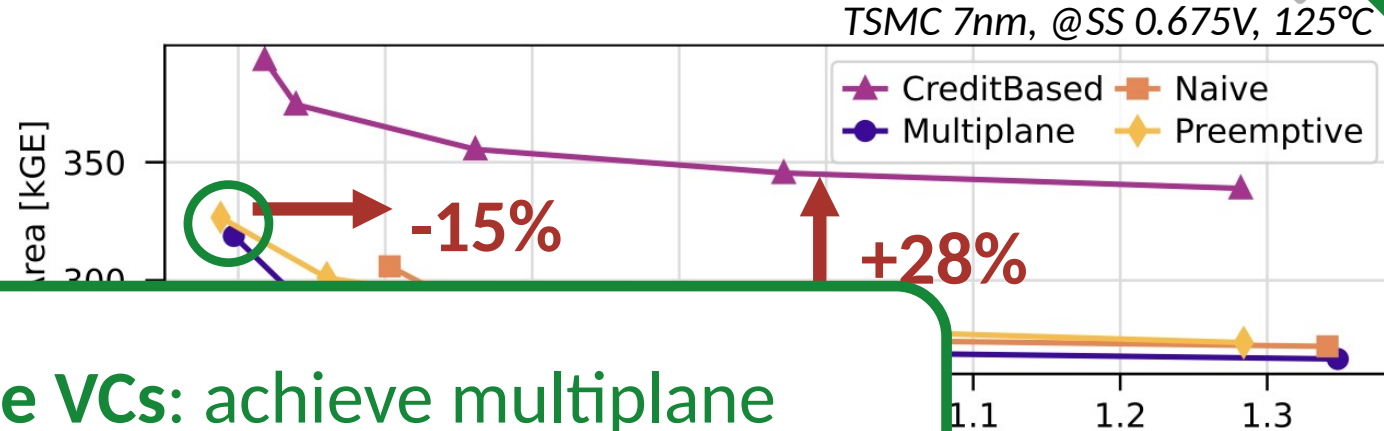
- **Preemptive:**

- Same frequency as multiplane: **1.7 GHz**

- Only **3%** area increase

- Save **76%** routing resources over multiplane

Preemptive VCs: achieve multiplane efficiency with 76% less wiring.



<https://github.com/pulp-platform/FlooNoC>

