SUNMAP / xpipes: A NoC Synthesis Flow

Federico Angiolini
fangiolini@deis.unibo.it
Universita' di Bologna

Srinivasan Murali
smurali@stanford.edu
Stanford University

Outline

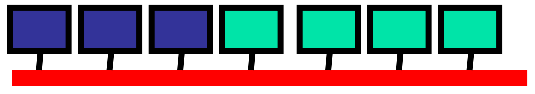
- The need for NoCs
- The xpipes NoC
- The SUNMAP flow
- xpipes simulation in MPARM
- xpipes synthesis results

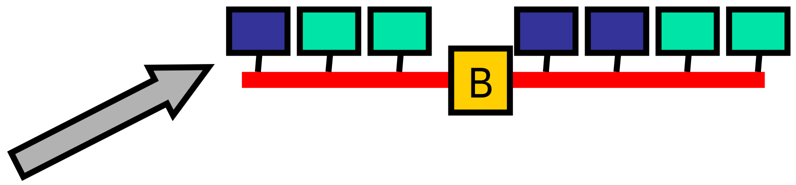
What’s happening in SoCs?

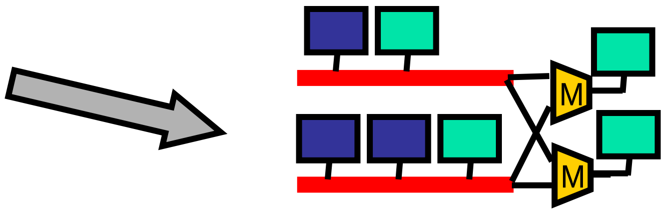
- Technology: *no slow-down in sight!*
  - Faster and smaller transistors: 90→65→45 nm
  - ... but slower wires, lower voltage, more noise!
- Design complexity: *from 2 to 10 to 100 cores!*
  - Design reuse is essential
  - ...but differentiation/innovation is key for winning on the market!
  - Design space exploration? Validation?
- Performance and power: *GOPS for MWs!*
  - Performance requirements keep going up
  - ...but power budgets don’t!

Topology

- Single shared bus is clearly non-scalable
- Evolutionary path
  - "Patch" bus topology
- Two approaches
  - Clustering & Bridging
  - Multi-layer/Multibus



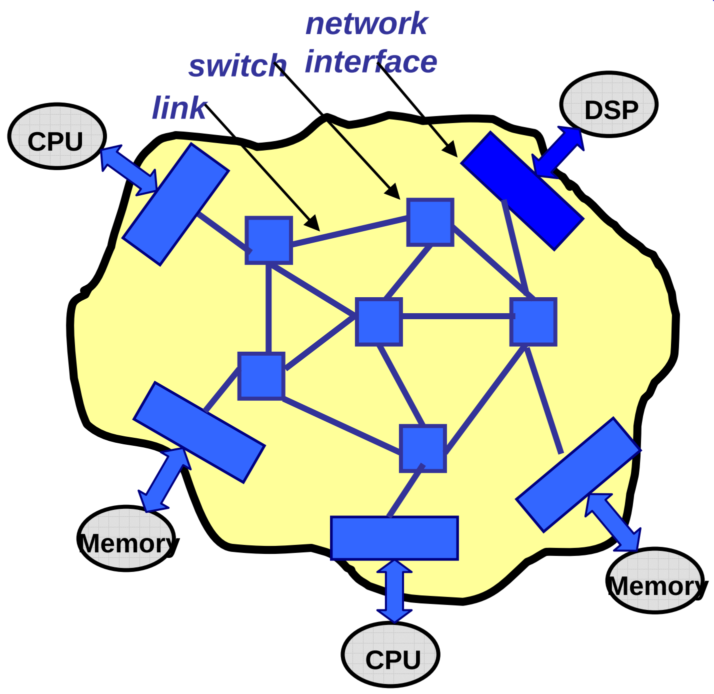




Crossbar: critical analysis

- No bandwidth reduction
- Scales poorly
  - N^2 area and delay
  - A lot of wires and a lot of gates in a bus-based crossbar
    - e.g. Area_cell_4x4/Area_cell_bus ~2 for STBus
- No locality
- Does not scale beyond 10x10!

NoCs



- More radical solutions in the long term
  - *Nostrum*
  - *HiNoC*
  - *Linkoeping SoCBUS*
  - *SPIN*
  - *Star-connected on-chip network*
  - *Aethereal*
  - *aSoC*
  - *Spidergon*
  - *Mango*
  - *Proteo*
  - *xpipes*
  - ...

The “power of NoCs”

Design methodology

- Clean separation at the **session layer**:
  1. Define end-to-end transactions
  2. Define quality of service requirements
  3. Design transport, network, link, physical
- Modularity at the HW level: only 2 building blocks
  1. Network interface
  2. Switch (router)
- Physical design aware (floorplan global routing)

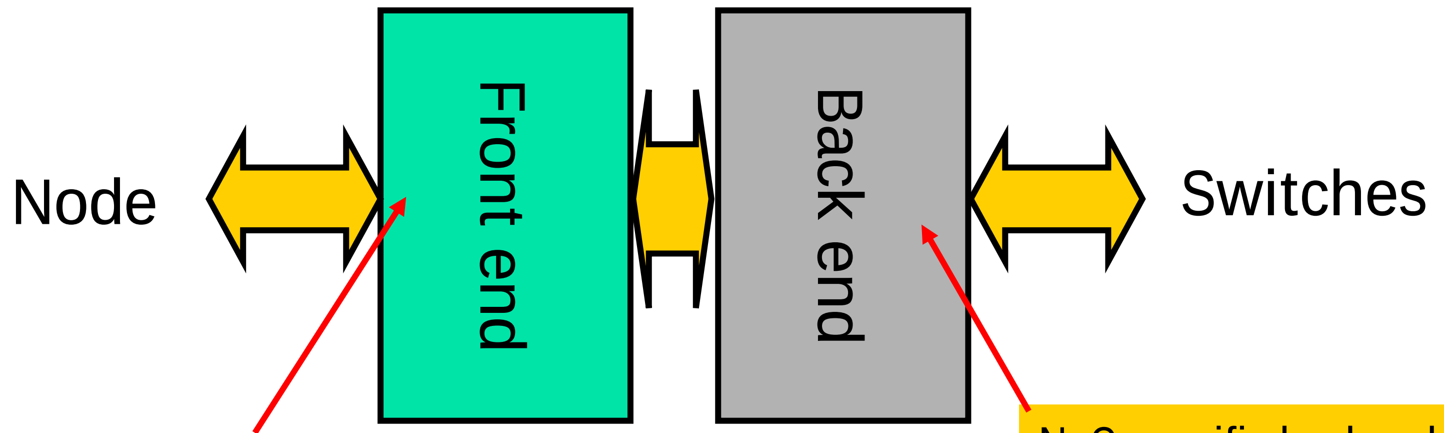
Scalability is supported from the ground up
(e.g. no centralized control structures)

NoCs vs. Busses

- Packet-based
  - No distinction address/data, only packets (but of many types)
  - Complete separation between end-to-end transactions and data delivery protocols
- Distributed vs. centralized
  - No global control bottleneck
  - Better link with placement and routing
- Bandwidth scalability, of course!

Building blocks: NI

- Session-layer interface with nodes
- Back-end manages interface with switches



Standardized node interface @ session layer.
Initiator vs. target distinction is blurred

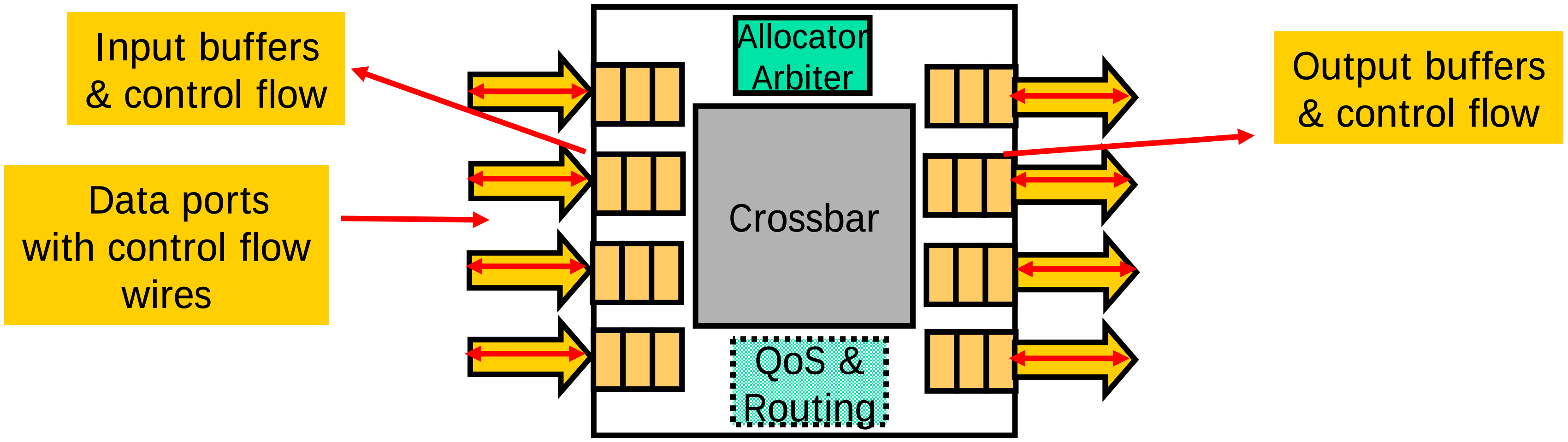
1. Supported transactions (e.g. QoSread...)
2. Degree of parallelism
3. Session prot. control flow & negotiation

NoC specific backend (layers 1-4)

1. Physical channel interface
2. Link-level protocol
3. Network-layer (packetization)
4. Transport layer (routing)

Building blocks: Switch

- Router: receives and forwards packets
  - NOTE: Packet-based does not mean datagram!


Input buffers & control flow
Data ports with control flow wires
Allocator Arbiter
Crossbar
QoS & Routing
Output buffers & control flow

Design options:

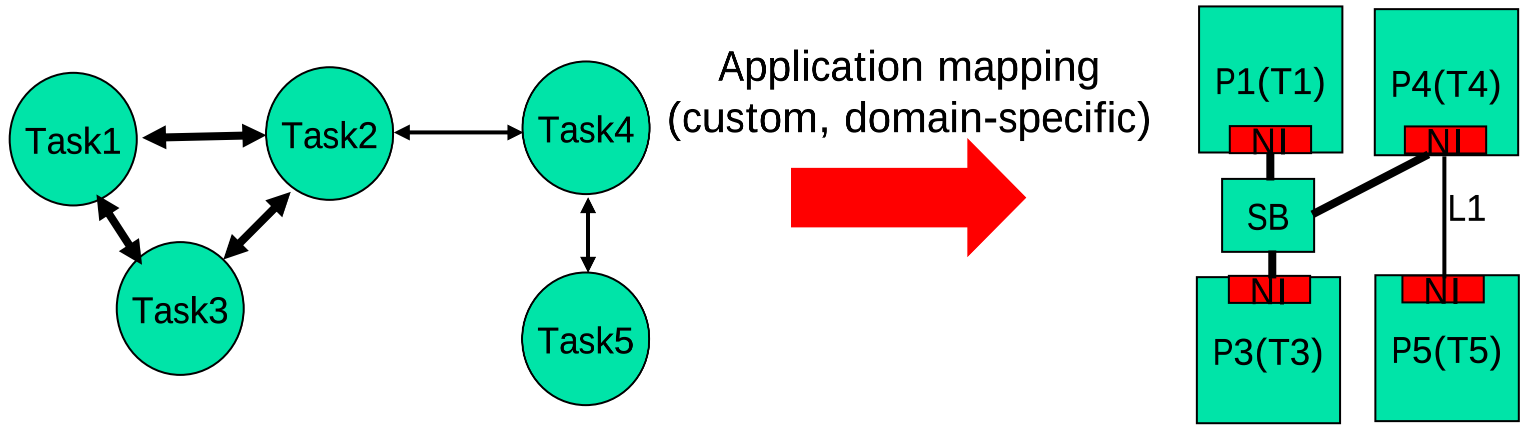
- Buffering (input, output, virtual channels)
- Switching technique (store and forward, virtual cut-through, wormhole)
- Routing (source-based, destination-based: deterministic, adaptive, randomized)
- Flow control (ACK-NACK, ON-OFF, credit-based)
- Arbitration policy (RR, iSLIP, TDMA, priority)
- Quality of Service (circuit switching, best-effort, priorities)

Outline

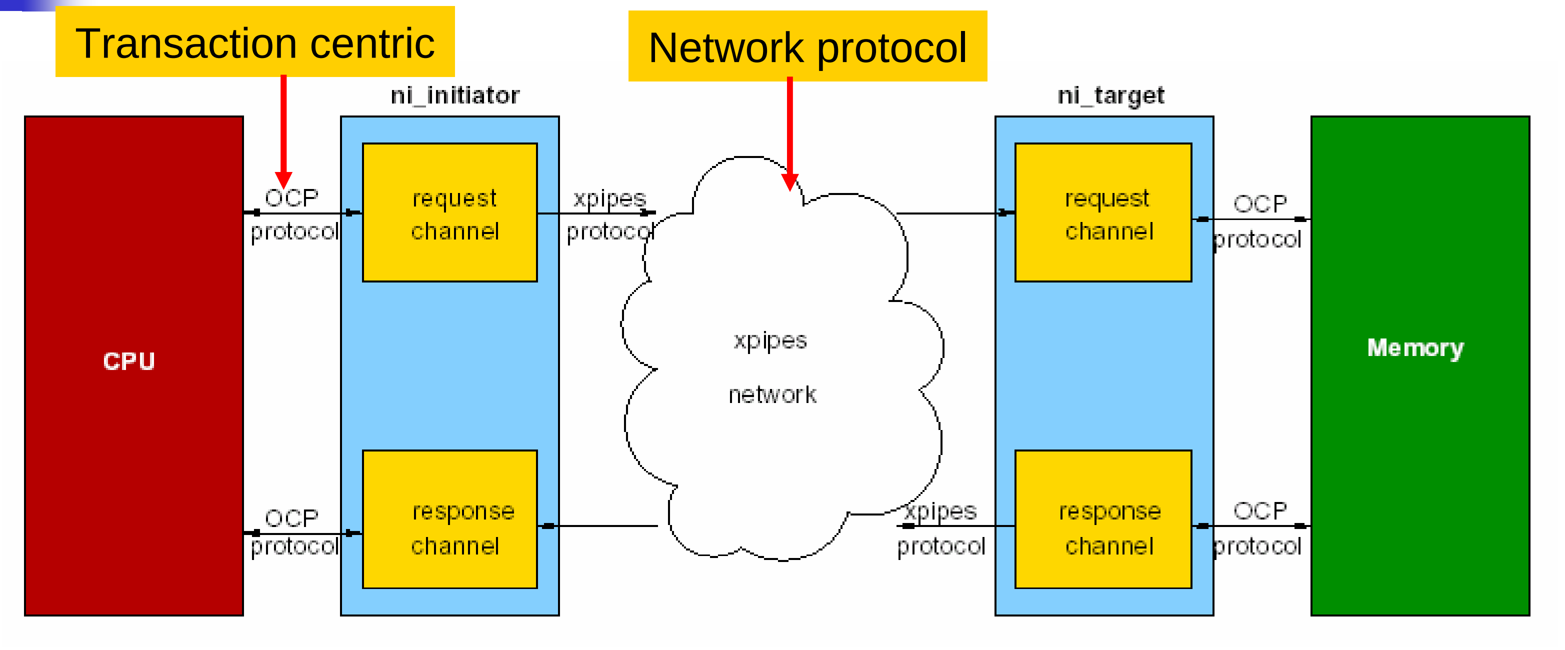
- The need for NoCs
- The xpipes NoC
- The SUNMAP flow
- xpipes simulation in MPARM
- xpipes synthesis results

xpipes: context

- Typical applications targeted by SoCs
  - Complex
  - Highly heterogeneous (component specialization)
  - Communication intensive
- xpipes is a synthesizable, high performance, heterogeneous NoC infrastructure



xpipes Network Interface

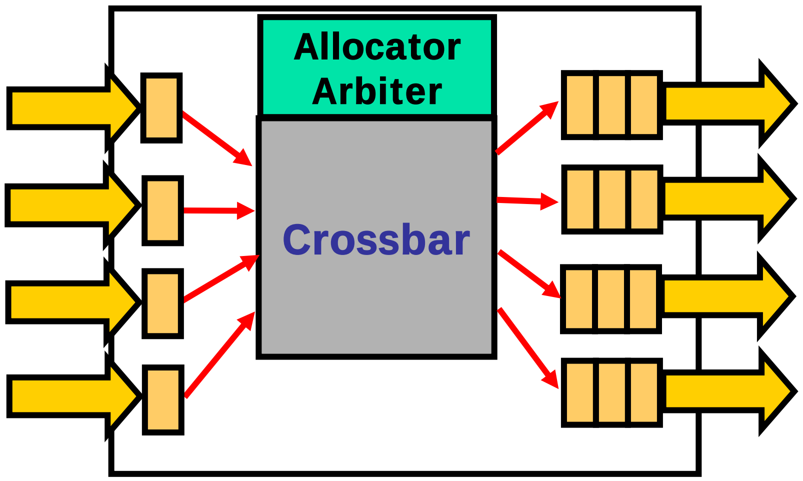


Open Core Protocol (OCP):

- End-to-end communication protocol
- Independence of request/response phases
- Can be tailored to core features
- Support for sideband signals (e.g., interrupts)
- Efficient burst handling
- Supports threading extensions

xpipes Switch

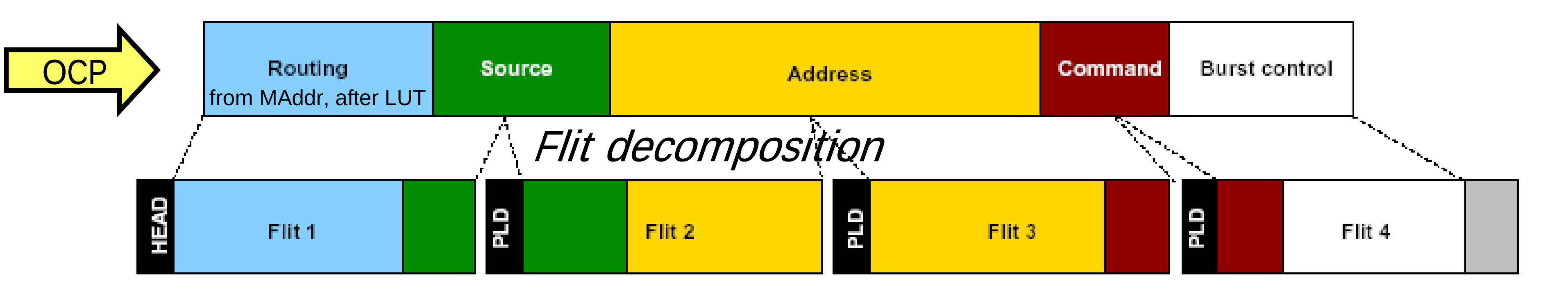
- Output Buffering
  - ✓ Dual ported memory bank, purposes:
    1. Buffering for performance (tunable area/speed tradeoff)
    2. Error recovery on NACK
- Tuned for pipelined unreliable links



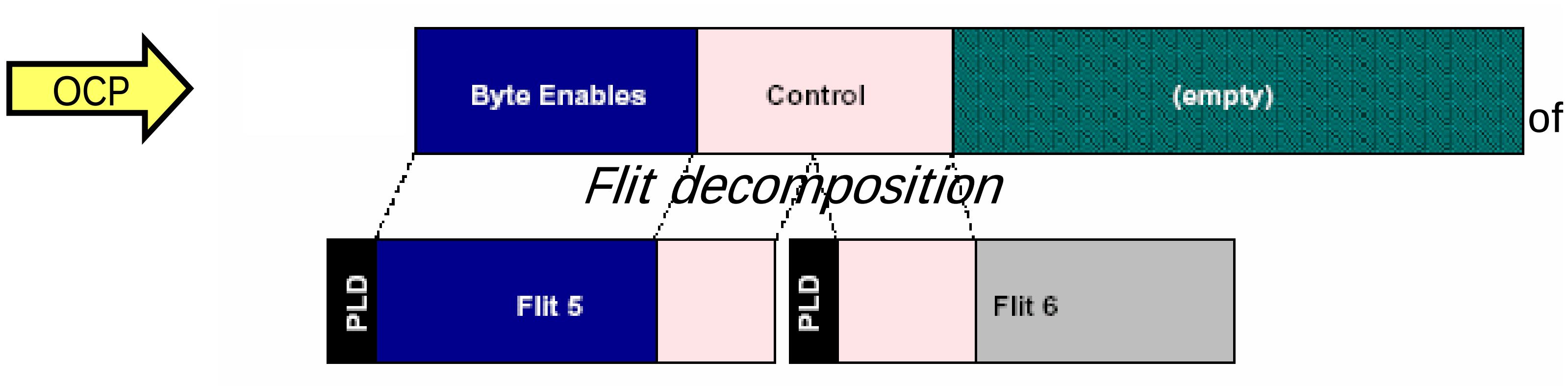
- Flow control: ACK/NACK, stall/go, T-Error
- 2-stages pipeline
- High speed (1GHz @ 130nm)
- Wormhole switching
- Arbitration: fixed priority, RR
- Source routing

xpipes Packeting Mechanism

Header register (about 50 bits): one for every transaction



Payload register: one for every burst beat



1st beat
of a burst read

xpipes Design Challenges

- The fight against latency: multi-hop topologies are at a disadvantage
  - Low number of stages per hop
  - Overclock the network
- Minimize the price for flexibility
  - Synthesis-aware design
  - Use specialized leaf cells

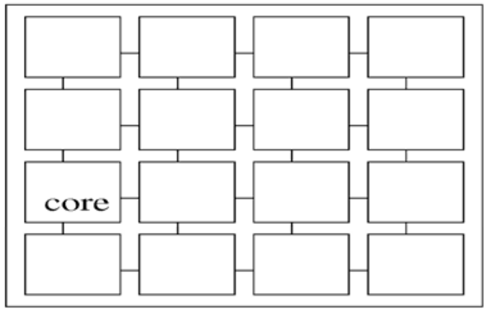
Outline

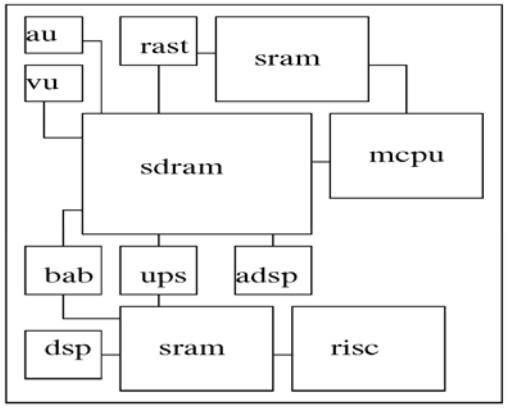
- The need for NoCs
- The xpipes NoC
- The SUNMAP flow
- xpipes simulation in MPARM
- xpipes synthesis results

Heterogeneous topologies in SoCs

SoC *component specialization* leads to the integration of *heterogeneous cores*

Ex. MPEG4 Decoder



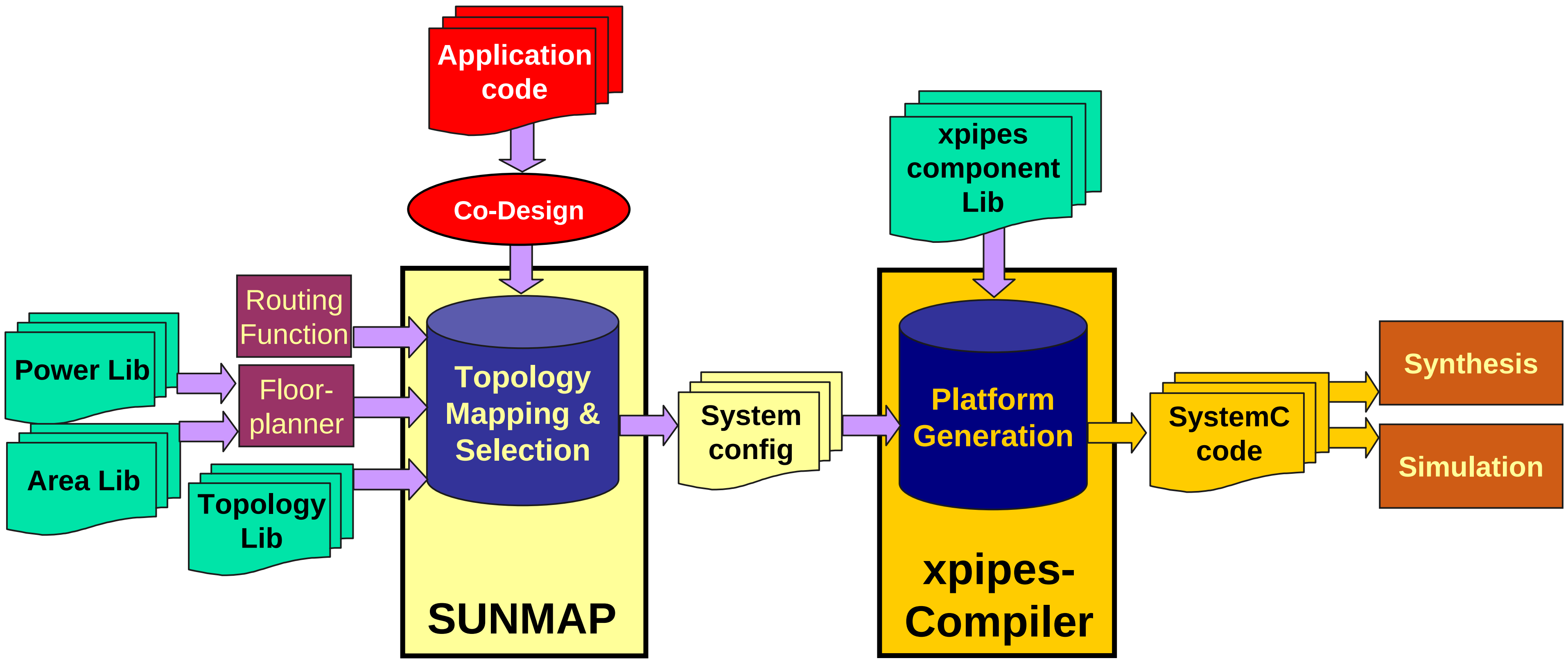


- Non-uniform block sizes
- SDRAM: communication bottleneck
- Many neighboring cores do not communicate

On a homogeneous fabric:

- Risk of under-utilizing many tiles and links
- Risk of localized congestion

NoC synthesis flow



SUNMAP: Topology Mapping

- Optimizes for area, power or delay within design constraints
- Uses heuristics to perform mapping onto topologies: mesh, torus, hypercube, clos and butterfly
- Built in floorplanner for area, power analysis
- Choice of different routing functions

SUNMAP: Topology Mapping 2

Heuristic approach with several phases:

Initial mapping using a greedy algorithm (from communication graph)

- Compute optimal routing (using flow formulation)

1. Floorplan solution
2. Check area and bandwidth constraints
3. Compute mapping cost

Iterative improvement loop (Tabu search)

Allows manual and interactive topology creation

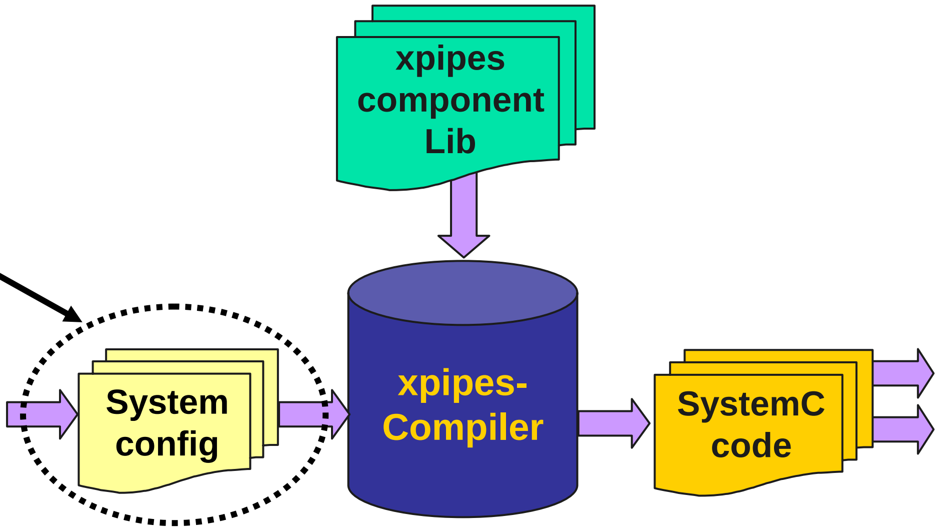
System configuration

```
// In this topology: 8 cores, 8 memories, 4x4 torus
// ---------------------------- IP cores
// name, switch number, clock divider, buffers, type
core(core_0, switch_0,  1, 6, initiator);
core(mem_8,   switch_11, 1, 6, target:0x00);
[…]
// ---------------------------- switches
// name, input ports, output ports, buffers
switch(switch_0,  5, 5, 6);
switch(switch_1,  5, 5, 6);
[…]
// ---------------------------- links
// name, source, destination
link(link0,  switch_0,  switch_1);
link(link1,  switch_1,  switch_0);
[…]
// ---------------------------- routes
// source, destination, hops
route(core_0, pm_8,  switches:0,1,5,6,7,11);
route(core_1, pm_9,   switches:1,5,9,8);
route(core_2, pm_10,  switches:2,6,5,9);
route(core_3, pm_11,  switches:3,2,6,10);
[…]
```

- Specifies
  - NIs (I/Os, clocks, buffers)
  - switches (I/Os, buffers)
  - links
  - routes

xpipesCompiler: Platform Generation

- Topology
- Routing tables
- Parameters (flit width, buffering, …)

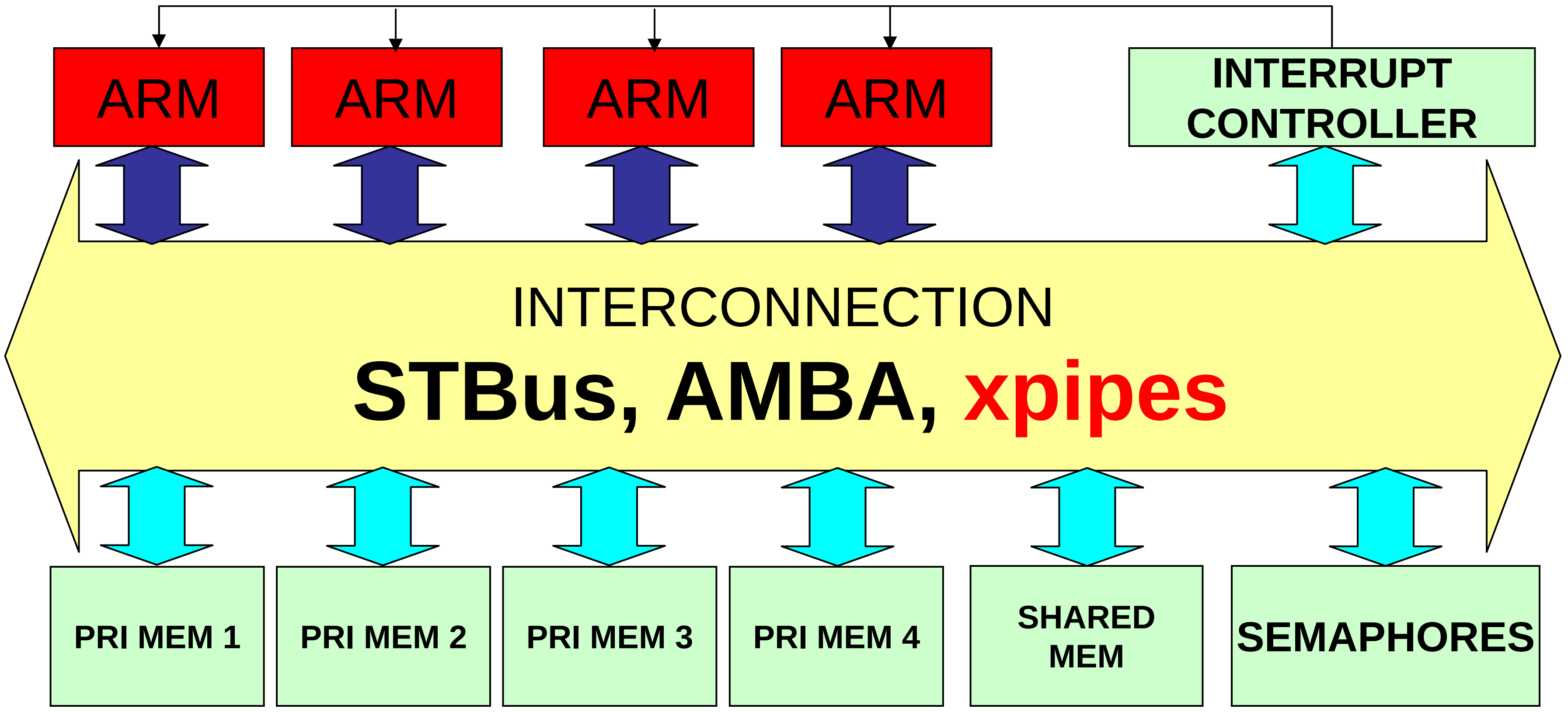


- Creation of a class template for each type of network component based upon component configuration (I/O ports, buffer sizing)
- Hierarchical instantiation of the platform in SystemC
  - Synthesis view
  - Simulation view

Outline

- The need for NoCs
- The xpipes NoC
- The SUNMAP flow
- xpipes simulation in MPARM
- xpipes synthesis results

MPARM Architecture

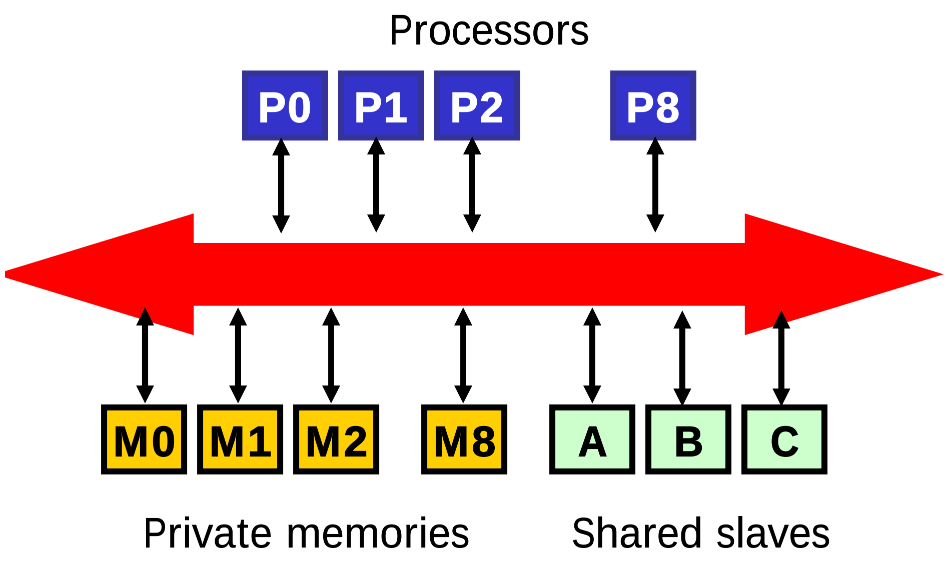


MPARM Features

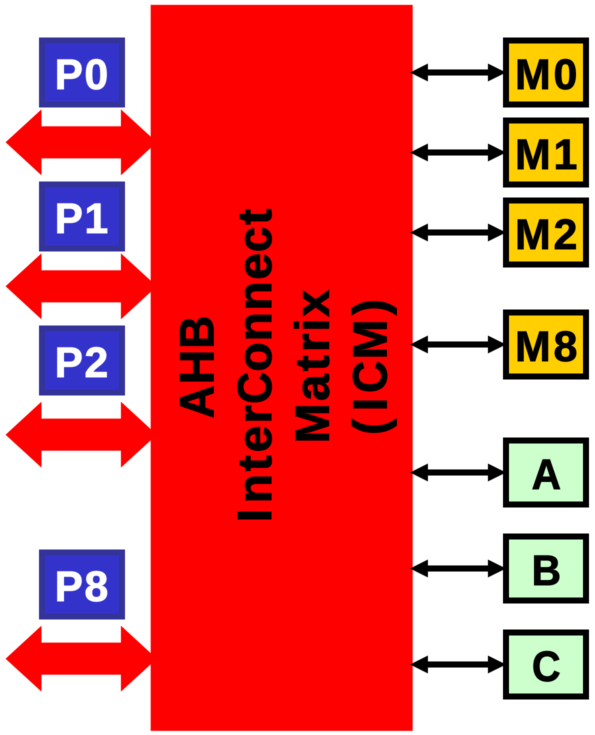
- Cycle-accurate environment
- Pluggable IP cores and interconnects
- Flexible memory hierarchy
- Power models for cores and memories
- Port of the RTEMS real-time embedded OS
- Growing benchmark suite (DES, FFT, JPEG, H.263, MPEG...)
- Actual functional traffic: real app on real OS on real IP core

NoCs at work: cross-benchmarking

AMBA AHB Topologies



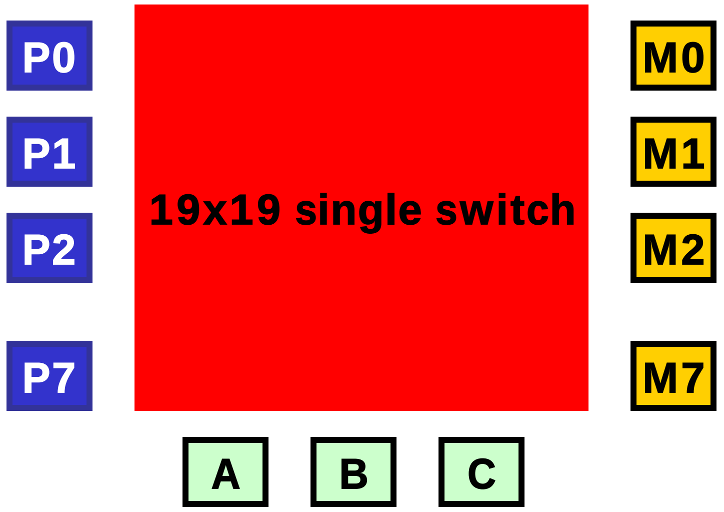
Shared bus



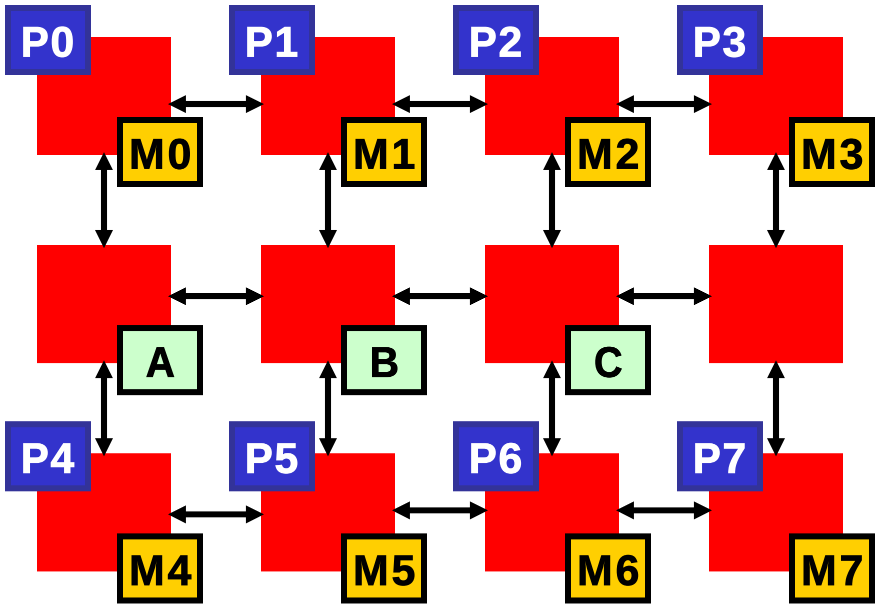
Multilayer (crossbar)

Topologies under test

xpipes topologies

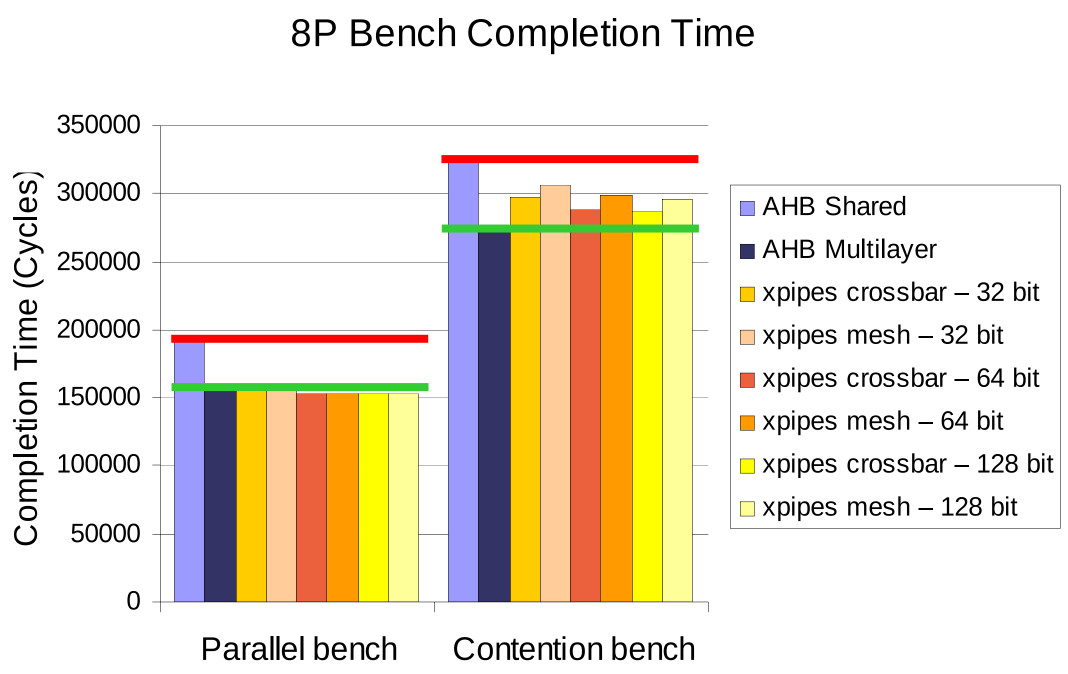


Crossbar-like



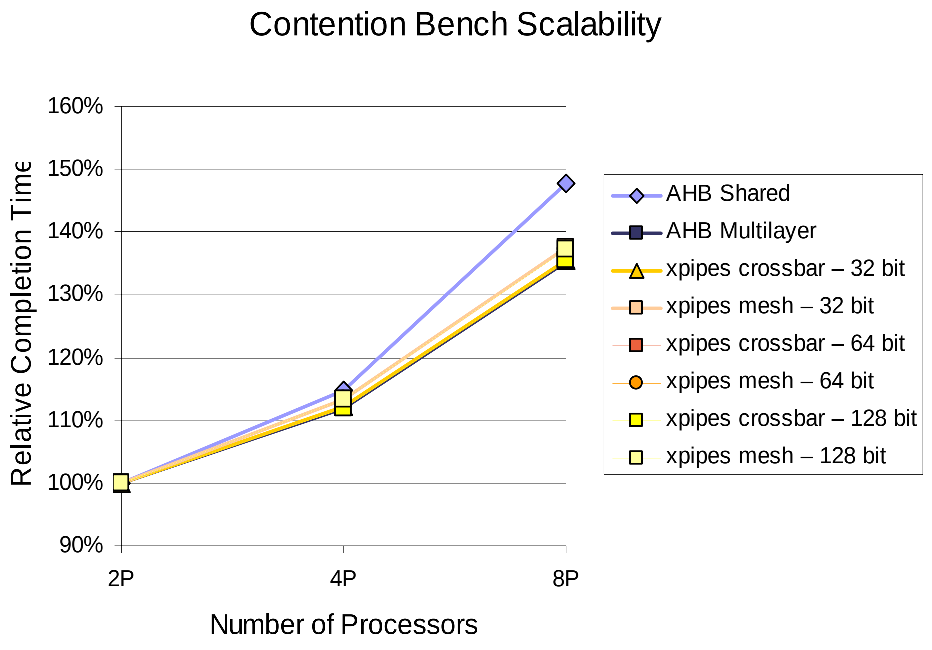
Mesh

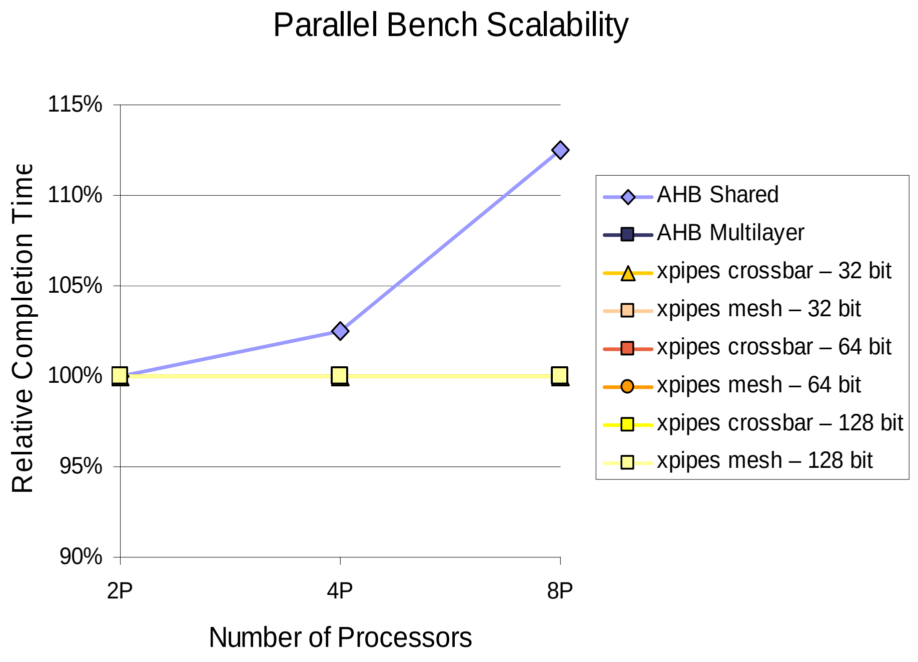
Benchmark execution time



- AHB Shared: saturated with 8P
- AHB ML: best case (full crossbar, no arbitration latency)
- xpipes: good performance due to available bandwidth, despite packeting latency penalty

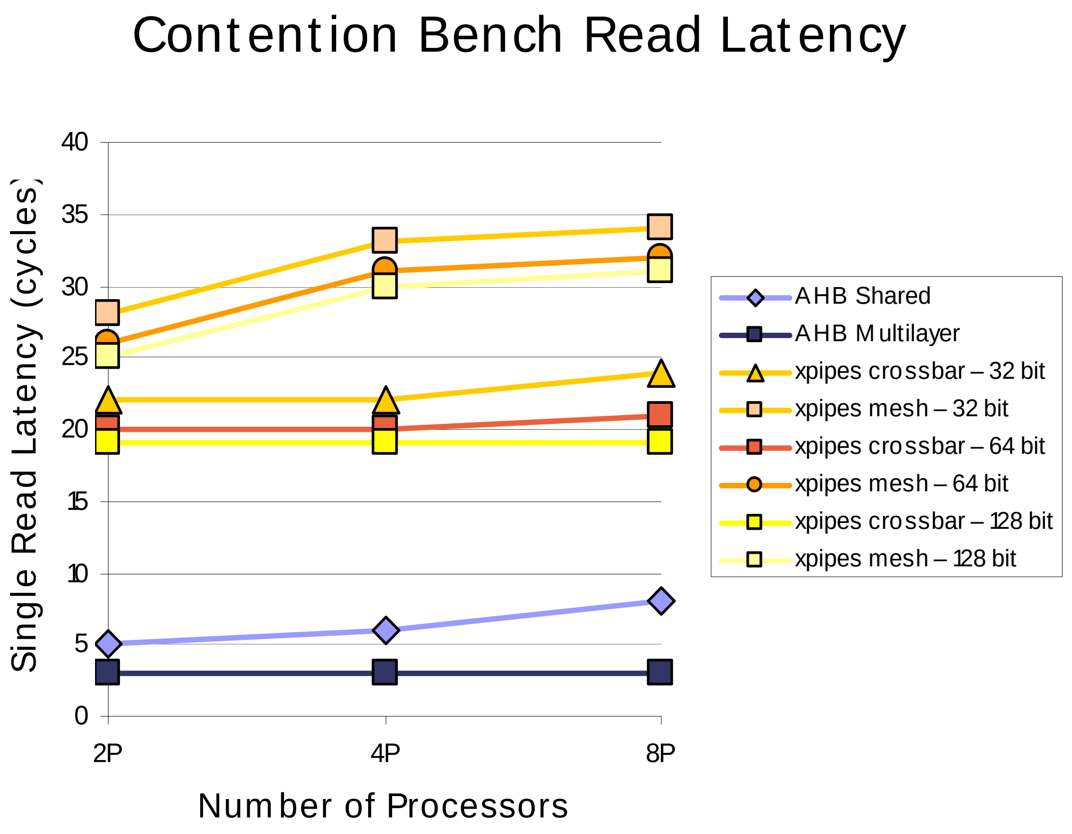
Scalability results





- xpipes crossbar scales as AHB ML
- xpipes mesh scales almost as well (yet, distributed topology!)

Latency analysis



- Reads to shared memory
- Mesh scales a bit worse than crossbar due to link congestion and more hops
- Latency is a target for optimization (but likely not only in xpipes!...)

Ongoing xpipes optimizations

- Improving latency by NI redesign:
  - latency-effective OCP handshaking
  - support for multiple clock domains in the NI (core 250 MHz / xpipes 1 GHz?), hiding packetization cycles
- Improving latency by flow control redesign:
  - under scrutiny: credit-based, pipelined start/stop, mixed credit-based + NACK

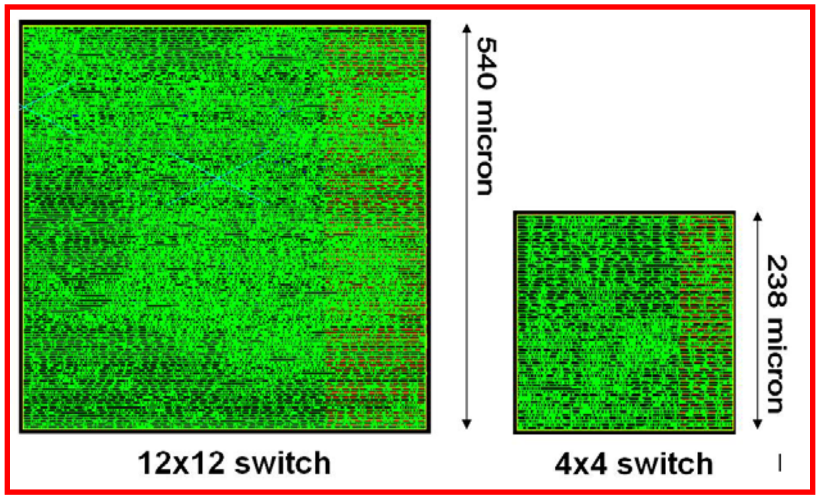
Outline

- The need for NoCs
- The xpipes NoC
- The SUNMAP flow
- xpipes simulation in MPARM
- xpipes synthesis results

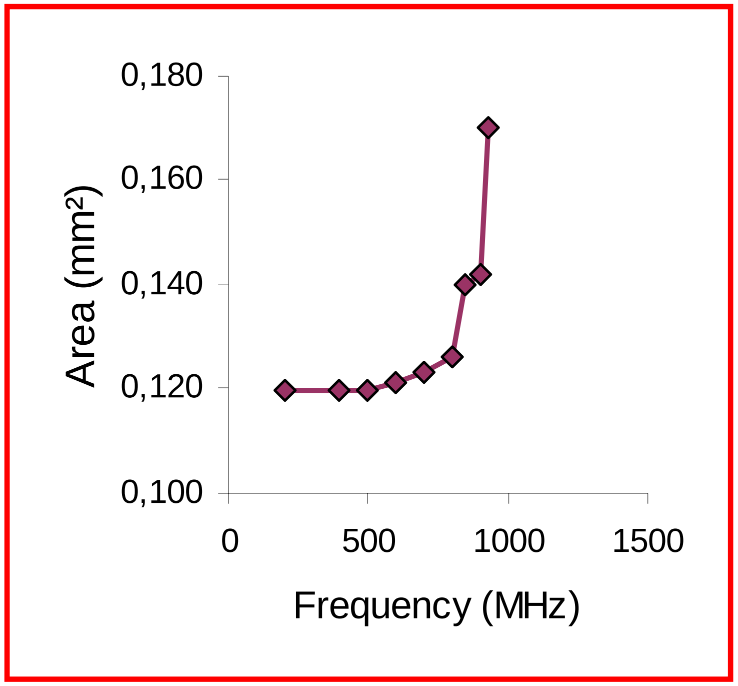
Synthesis back-end

- Fully synthesizable soft IPs, comparable with state-of-the art implementations

Switch layout

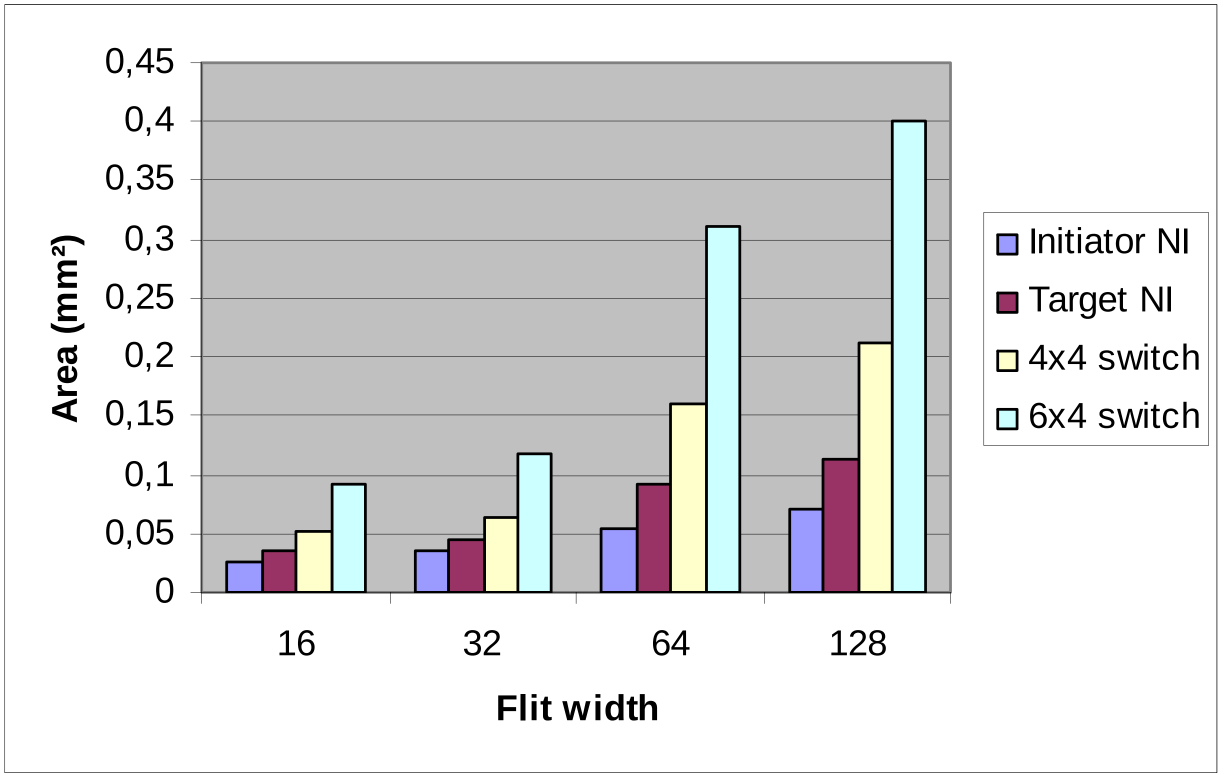


Area vs. frequency tradeoff



xpipes area/frequency

0.13 um technology, 4-flit output buffers



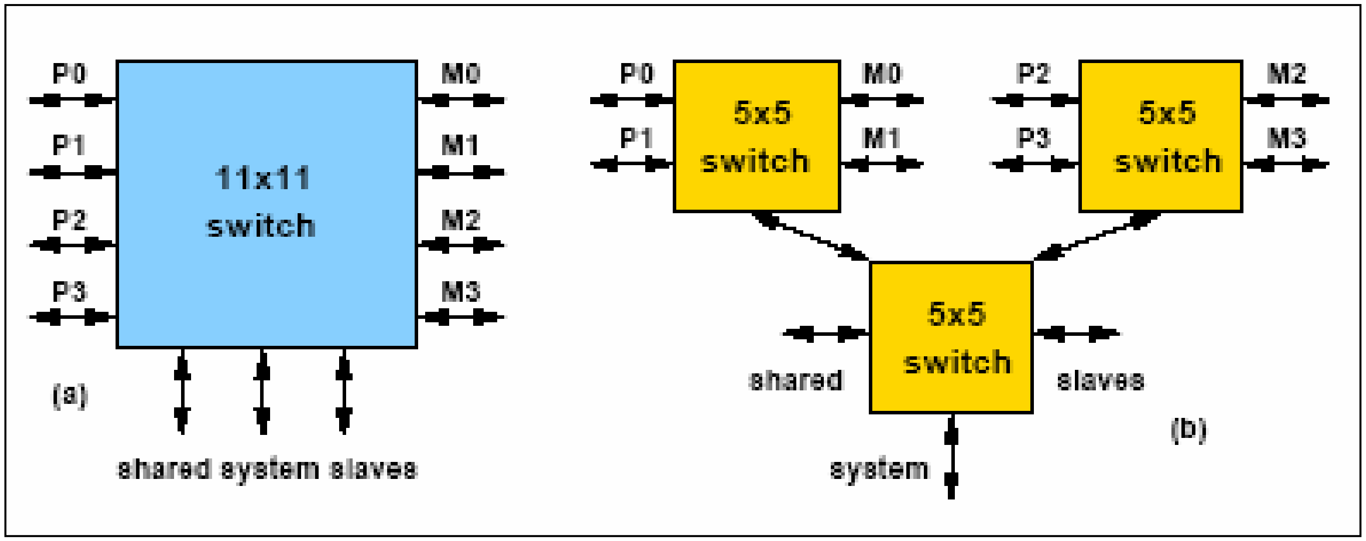
- Initiator NI: 1 GHz
- Target NI: 1 GHz
- 4x4 switch: 1 GHz
- 6x4 switch: 875-980 MHz

A 3x4 xpipes mesh with 8 processors and 11 slaves consumes ~2,6 mm²

xpipes validation

- ✓ Simulation with functional traffic (SystemC within MPARM)
- ✓ Pre-synthesis (Verilog)
- ✓ Post-synthesis (Verilog + synthesized blocks)

Design Space Exploration



- Functionally equivalent topologies
- (a): crossbar-like. Minimum latency. 0.48 mm², 780 MHz max
- (b): more latency, but more frequency headroom. Achieves performance parity at 850 MHz (0.42 mm²: -14% area) and a 10% performance boost at 925 MHz (0.51 mm²)