Networks on chip: Evolution or Revolution?

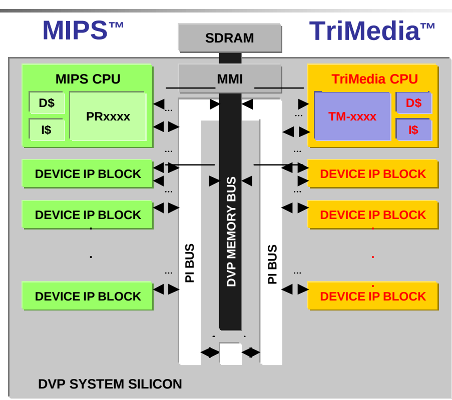
Luca Benini
lbenini@deis.unibo.it
DEIS-Universita' di Bologna

The evolution of SoC platforms

General-purpose Scalable RISC Processor
- 50 to 300+ MHz
- 32-bit or 64-bit

Library of Device IP Blocks
- Image coprocessors
- DSPs
- UART
- 1394
- USB

...



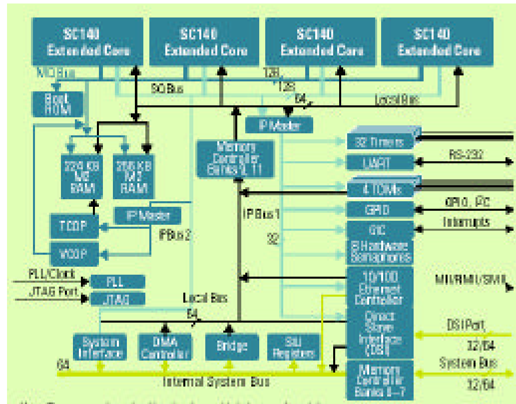
Scalable VLIW Media Processor:
- 100 to 300+ MHz
- 32-bit or 64-bit

Nexperia™ System Buses
- 32-128 bit

- **2 Cores**: Philips' Nexperia PNX8850 SoC platform for High-end digital video (2001)

Running forward…

- Four 350/400 MHz StarCore SC140 DSP extended cores
- 16 ALUs: 5600/6400 MMACS
- 1436 KB of internal SRAM & multi-level memory hierarchy
- Internal DMA controller supports 16 TDM unidirectional channels,
- Two internal coprocesssors (TCOP and VCOP) to provide special-purpose processing capability in parallel with the core processors



- **6 Cores**: Motorola's MSC8126 SoC platform for 3G base stations (late 2003)

What's happening in SoCs?

- Technology: no slow-down in sight!
  - Faster and smaller transistors
  - … 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!
- Performance and power: GOPS for MWs!
  - Performance requirements keep going up
  - …but power budgets don't!

…and on-chip communication?

- Starting point: the "on chip bus"
  - Advances in protocols
  - Advances in topologies
- Revolutionary approaches
  - Networks on chip
- Things are moving FAST
  - …but it's evolution or revolution?

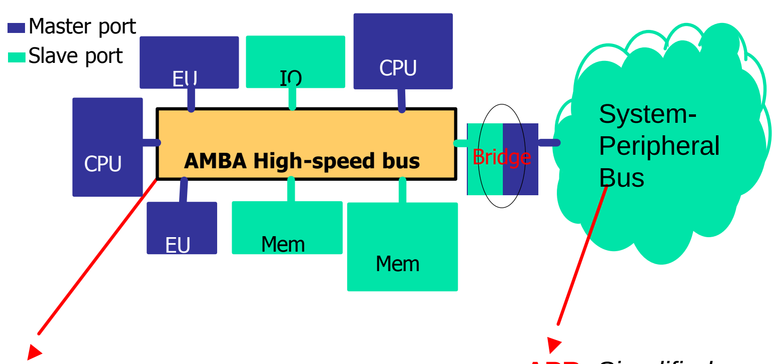
Outline

- Introduction and motivation
- On-chip networking
- The HW-SW interface

On-chip bus Architecture

- Many alternatives
  - Large semiconductor firms (e.g. IBM Coreconnect, STMicro STBus)
  - Core vendors (e.g. ARM AMBA)
  - Interconnect IP vendors (e.g. SiliconBackplane)
- Same topology, different protocols

AMBA bus

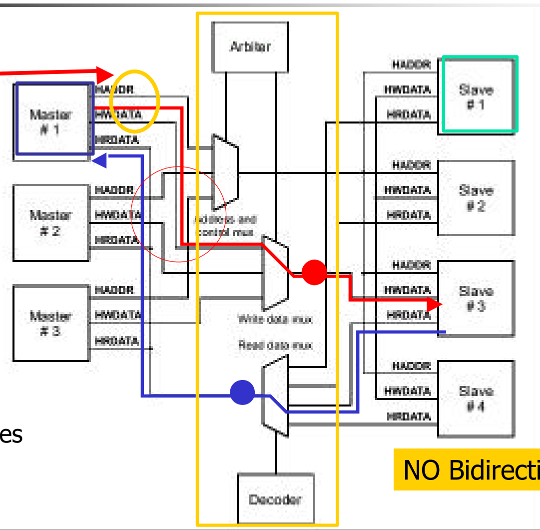


AHB: *high-speed high-bandwidth multi-master bus*

APB: *Simplified processor for general purpose peripherals*

AHB Bus architecture

Dedicated wires



Different wires

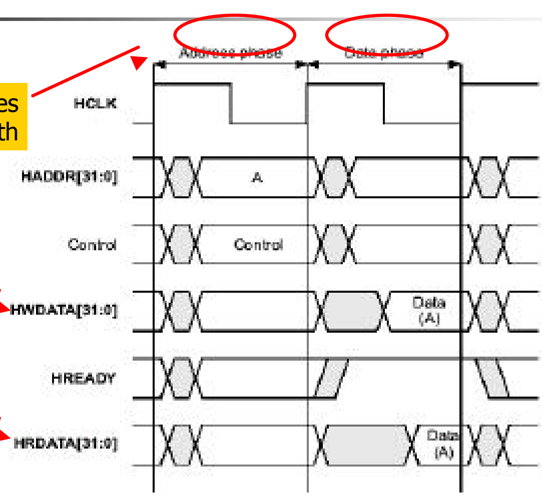
NO Bidirectional wires

AMBA basic transfer

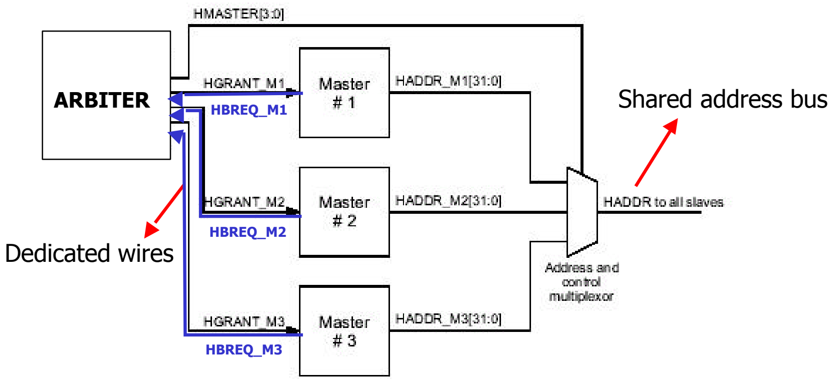
Pipelining increases Bus bandwidth

For a write

For a read

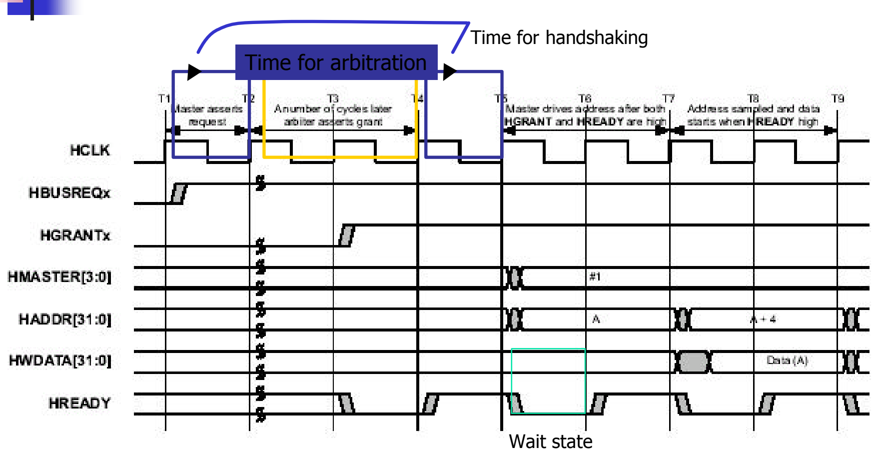


Bus arbitraton



Arbitration Protocol is defined, but Arbitration Policy is not

The price for arbitration



Burst transfers

- Burst transfers amortize arbitration cost
  - Grant bus control for a number of cycles
  - Help with DMA and block transfers
  - Help hiding arbitration latency
- Requires safeguards against starvation
  - Split and error

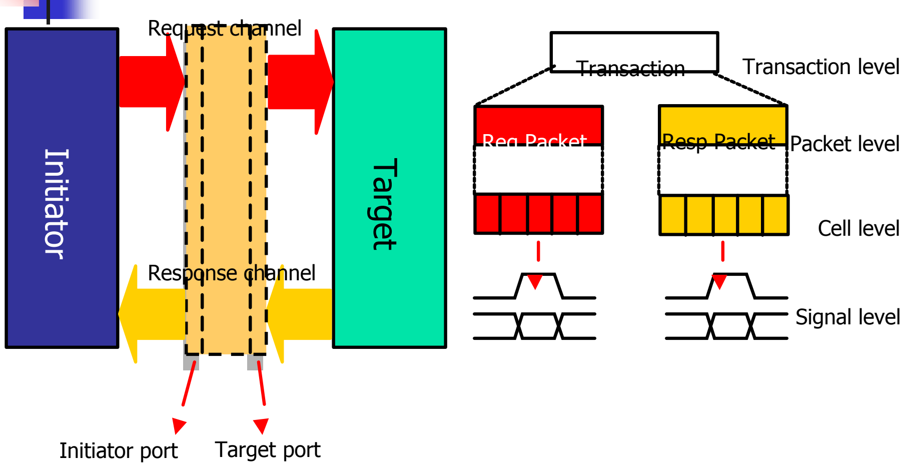
Critical analysis: bottlenecks

- Protocol
  - Lacks parallelism
    - In order completion
    - No multiple outstanding transactions: cannot hide slave wait states
  - High arbitration overhead (on single-transfers)
  - Bus-centric vs. transaction-centric
    - Initiators and targets are exposed to bus architecture (e.g. arbiter)
- Topology
  - Scalability limitation of shared bus solution!

STBUS

- On-chip interconnect solution by ST
  - Level 1-3: increasing complexity (and performance)
- Features
  - Higher parallelism: 2 channels (M-S and S-M)
  - Multiple outstanding transactions with out-of order completion
  - Supports deep pipelining
  - Supports Packets (request and response) for multiple data transfers
  - Support for protection, caches, locking
- Deployed in a number of large-scale SoCs in STM

STBUS Protocol (Type 3)



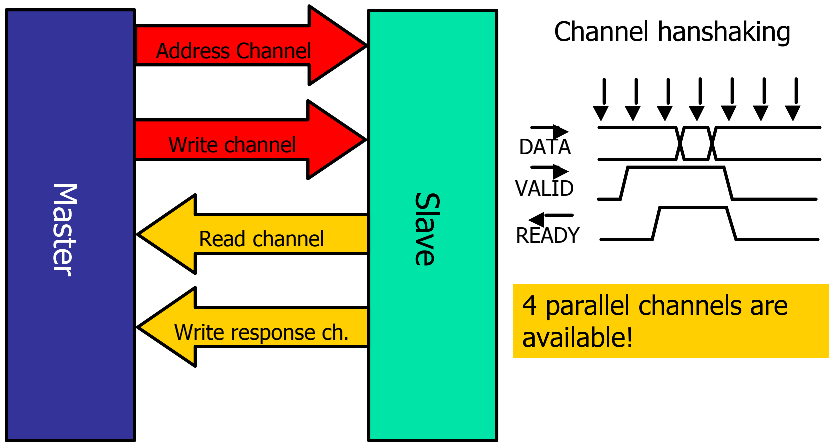
STBUS bottlenecks

- Protocol is not fully transaction-centric
  - Cannot connect initiator to target (e.g. initiator does not have control flow on the response channel)
- Packets are atomic on the interconnect
  - Cannot initiate nor receive multiple packets at the same time
  - Large data transfers may starve other initiators

AMBA AXI

- Latest (2003) evolution of AMBA
  - Advanced eXtensible Interface
- Features
  - Fully transaction centric: can connect M to S with nothing in between
  - Higher parallelism: multiple channels
  - Supports bus-based power management
  - Support for protection, caches, locking
- Deployment: ??

Multi-channel M-S interface

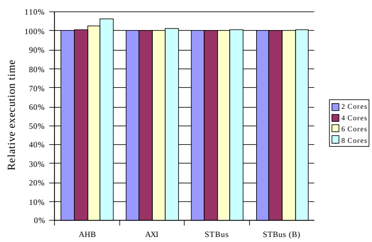


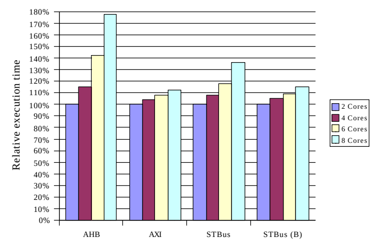
Multiple outstanding transactions

- A transaction implies activity on multiple channels
  - E.g Read uses the Address and Read channel
- Channels are fully decoupled in time
  - Each transaction is labeled when it is started (Address channel)
  - Labels, not signals, are used to track transaction opening and closing
  - Out of order completion is supported (tracking logic in master), but master can request in order delivery
- Burst support
  - Single-address burst transactions (multiple data channel slots)
  - Bursts are not atomic!
- Atomicity is tricky
  - Exclusive access better than locked access

Scalability: Execution Time

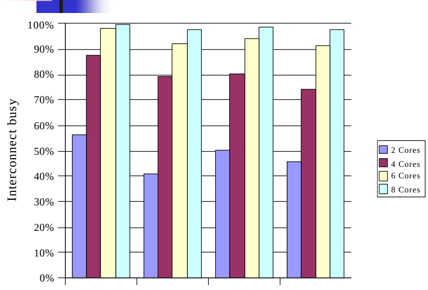
- Highly parallel benchmark (no slave bottlenecks)

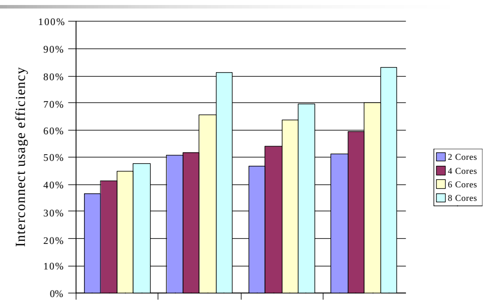




- 1 kB cache (low bus traffic)
- 256 B cache (high bus traffic)

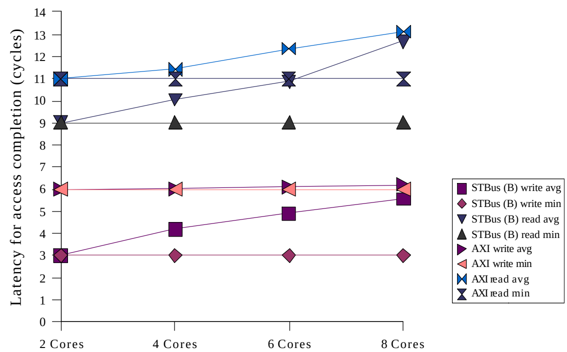
Scalability: Protocol Efficiency





- Increasing contention: AXI, STBus show 80%+ efficiency, AHB < 50%

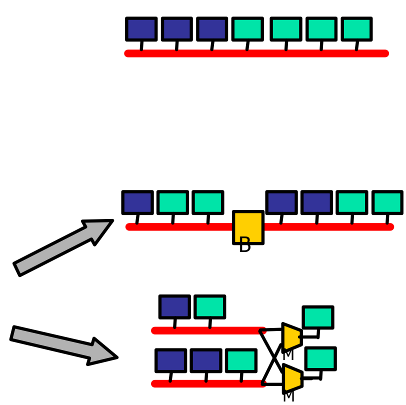
Scalability: latency



- STBus management has less arbitration latency overhead, especially noticeable in low-contention conditions

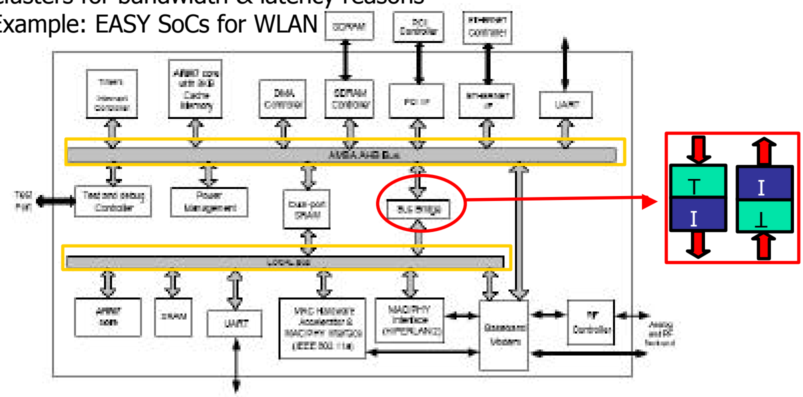
Topology

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



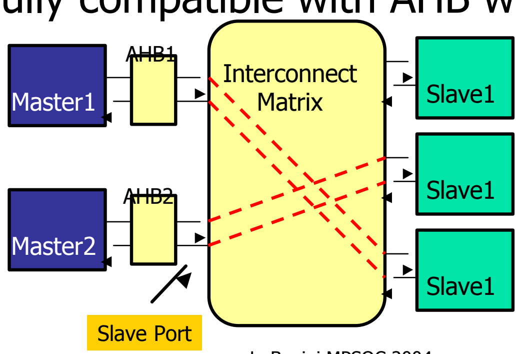
Clustering and bridging

- Heterogeneous architectures with asymmetric traffic
  - Cost for going across a bridge is HIGH
- Bus clusters for bandwidth & latency reasons
  - Example: EASY SoCs for WLAN



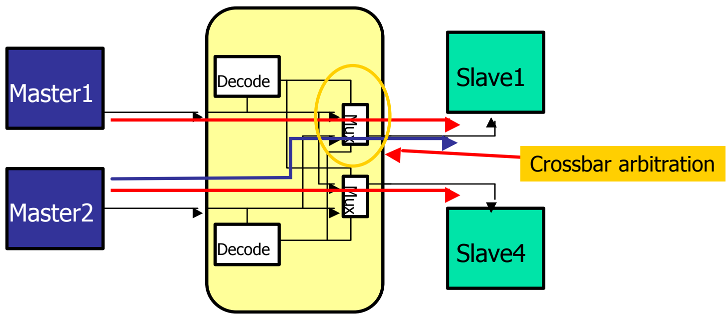
AMBA Multi-layer AHB

- Enables parallel access paths between multiple masters and slaves
- Fully compatible with AHB wrappers

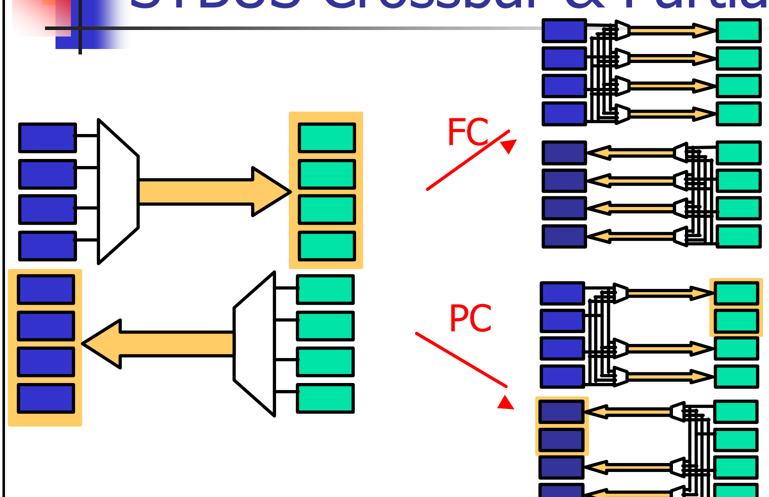


Multi-Layer AHB implementation

- The matrix is made of slave ports
  - No explicit arbitration of slaves
  - Variable latency in case of destination conflicts

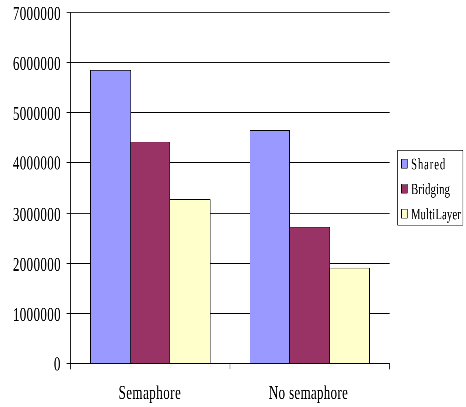


STBUS Crossbar & Partial CB



Topology speedup (AMBA AHB)

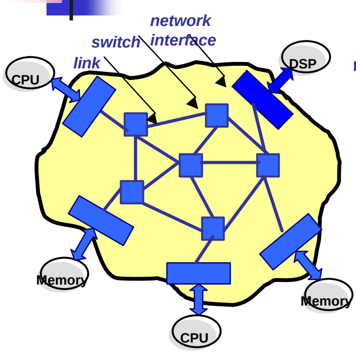
- Independent tasks (matrix multiply)
- With & without semaphore synchronization
- 8 processors (small cache)



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
  - Proteo
  - Xpipes
  - … (at least 15 groups)

NOCs vs. Busses

STBUS and AXI

- 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!

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

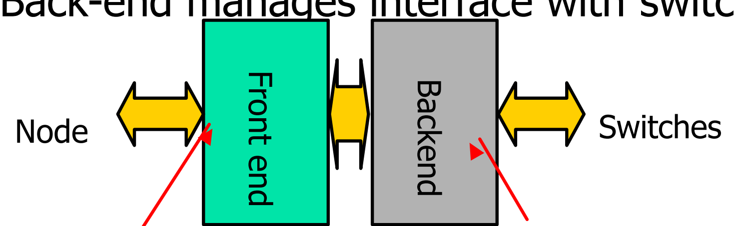
Modularity at the HW level: only 2 building blocks

1. Network interface
2. Switch (router)

Scalability is supported from the ground up (not as an afterthought)

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

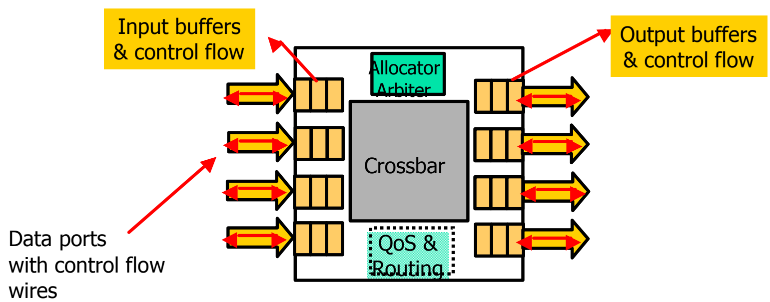
1. Supported transactions (e.g. QoSread…)
2. Degree of parallelism
3. Session prot. control flow & negotiation

NoC specific backend (layers 1-4)

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

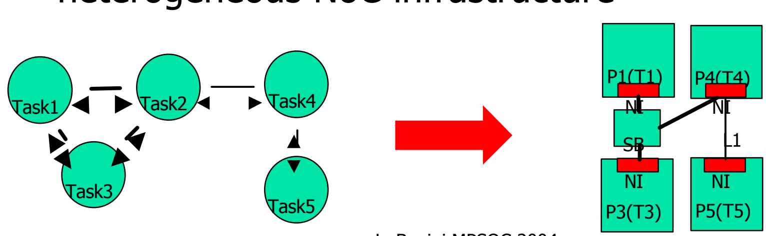
Building blocks: Switch

- Router: receives and forwards packets
  - NOTE: Packet-based does not mean datagram!
- Level 3 or Level 4 routing
  - No consensus, but generally L4 support is limited (e.g. simple routing)



Xpipes: context

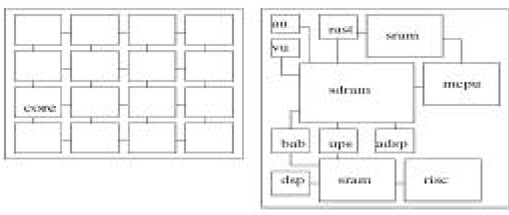
- Typical applications targeted by SoCs
  - Complex
  - Highly heterogeneous
  - Communication intensive
- Xpipes is a synthesizable, high performance, heterogeneous NoC infrastructure



Heterogeneous topology

SoC component specialization lead to the integration of heterogeneous cores

Ex. MPEG4 Decoder

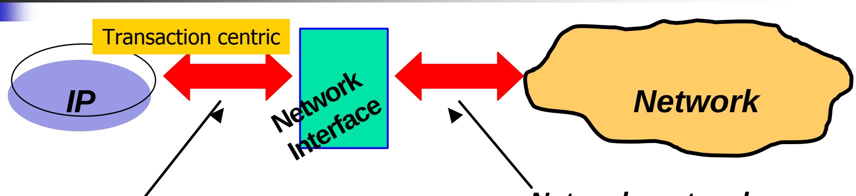


- Non-uniform block sizes
- SDRAM: communication bottleneck
- Many neighboring cores do not communicate

On a homogeneous fabric:

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

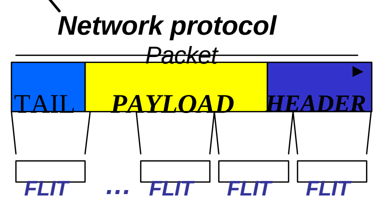
Network interface



Open Core Protocol (OCP)

End-to-end communication protocol

- *pipelining*
- *independence of request/response phase*

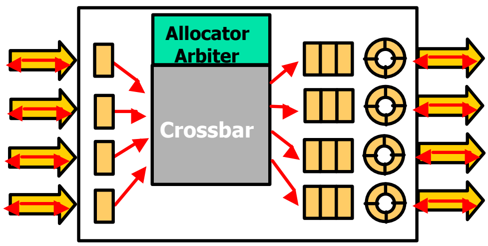


Header includes:

✓Path across the network
✓Source
✓Destination
✓Command type
✓Burst ID (MBurst)
✓Packet identifier within message (ID-PACKET)
✓ Local target IP address (IP_ADDR)

Switch (s-Xpipes)

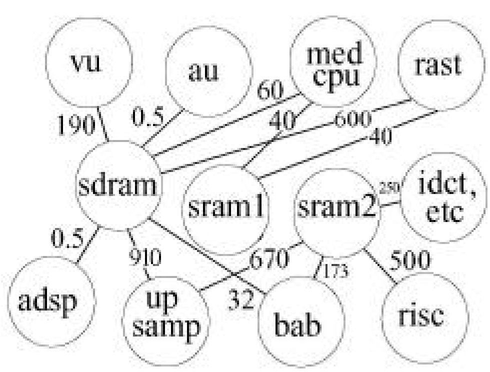
- Plain latching of inputs
- Buffering resources are on the output ports
  - FIFOs for performance (tunable area/speed tradeoff)
  - Circular buffers for ACK/NACK management (minimal size if directly attached to downstream component, can be larger for pipelined links)



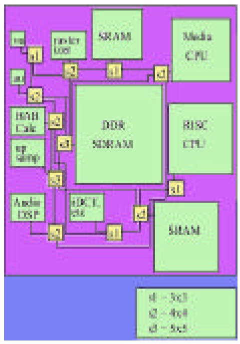
- ACK/NACK flow control
- 2-stage pipeline
- Tuned for high clock speeds

Example: MPEG4 decoder

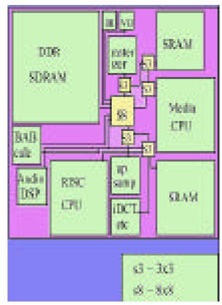
- Core graph representation with annotated average communication requirements



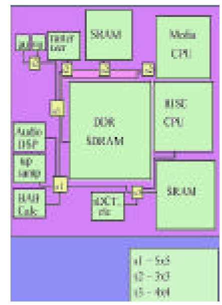
NoC Floorplans



General purpose: mesh

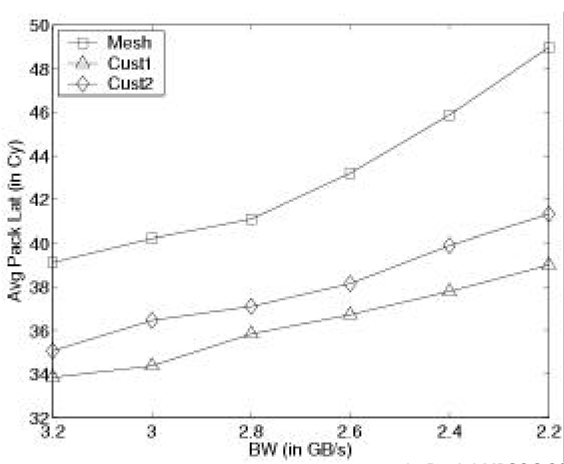


Application Specific
NoC1 (centralized)



Application Specific
NoC2 (distributed)

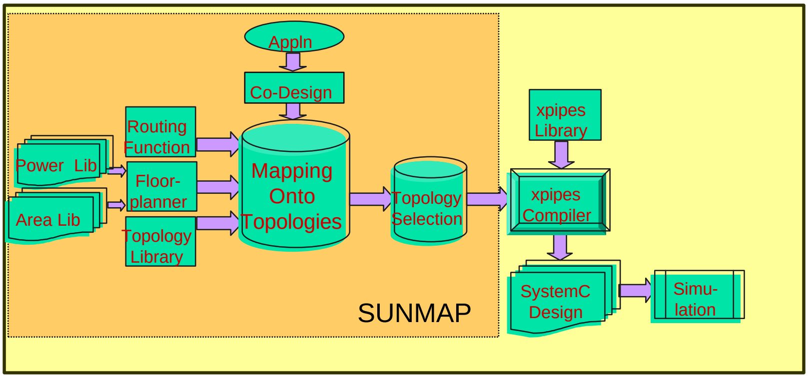
Performance, area and power



Less latency and better Scalability of custom NoCs

- Relative link utilization (customNoC/meshNoC): 1.5, 1.55
- Relative area (meshNoC/customNoC): 1.52, 1.85
- Relative power (meshNoC/customNoC): 1.03, 1.22

NoC synthesis flow

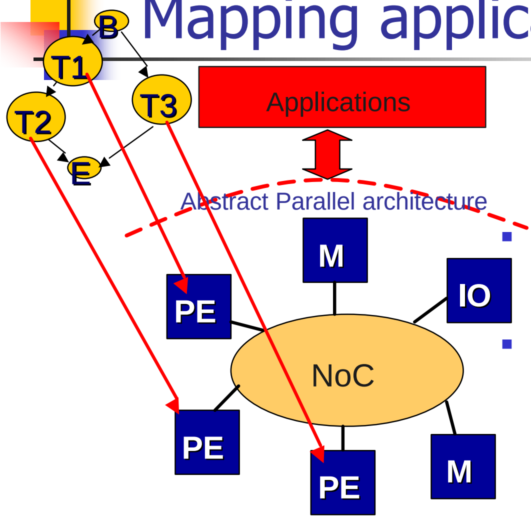


In cooperation with Stanford Univ.

Outline

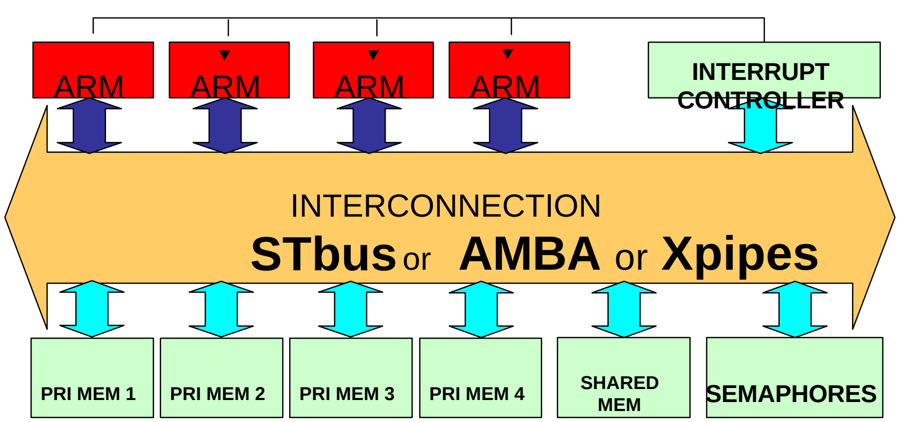
- Introduction and motivation
- On-chip networking
- The HW-SW interface
  - Session layer and above

Mapping applications

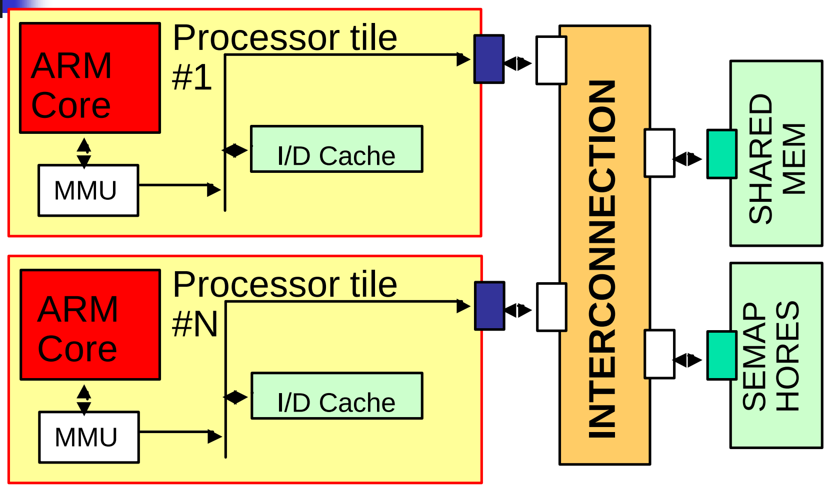


- Communication abstractions
  - Shared memory (UMA vs NUMA)
  - Message passing
- What hardware support to communication abstractions?

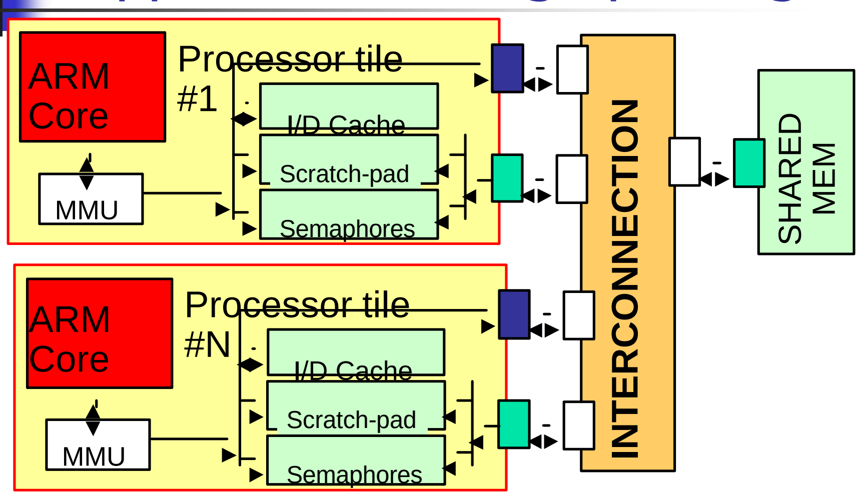
MPARM Architecture



Basic architecture

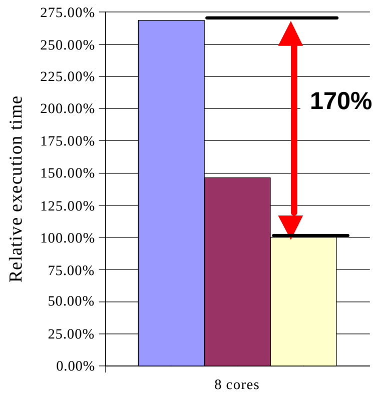


Support for message passing

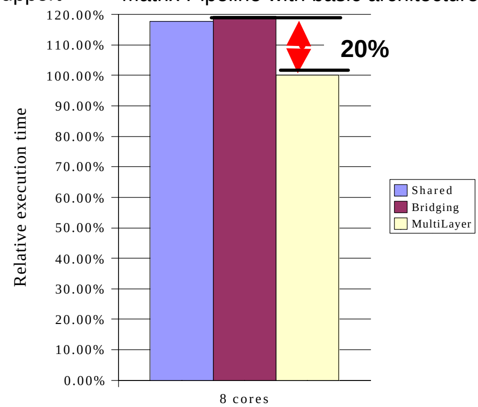


HW support for MP: results

Matrix Pipeline with message passing support

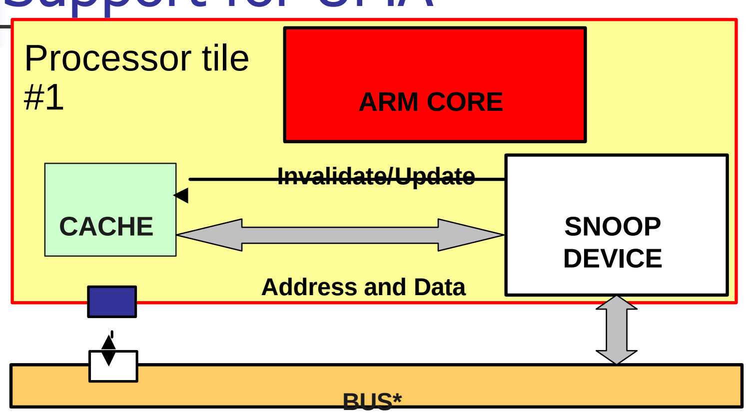


Matrix Pipeline with basic architecture



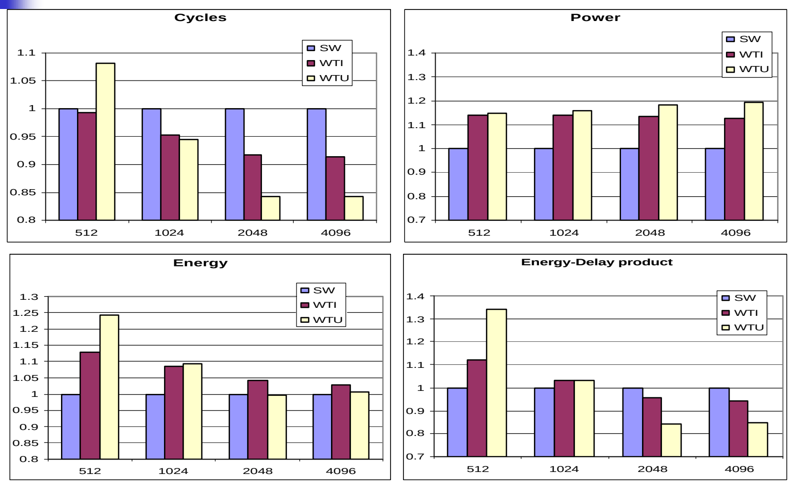
Send+Receive cost: 35KCycles (basic architecture) vs. 4KCycles (MP support)
Configuration: 4 Processors, Shared bus

Support for UMA

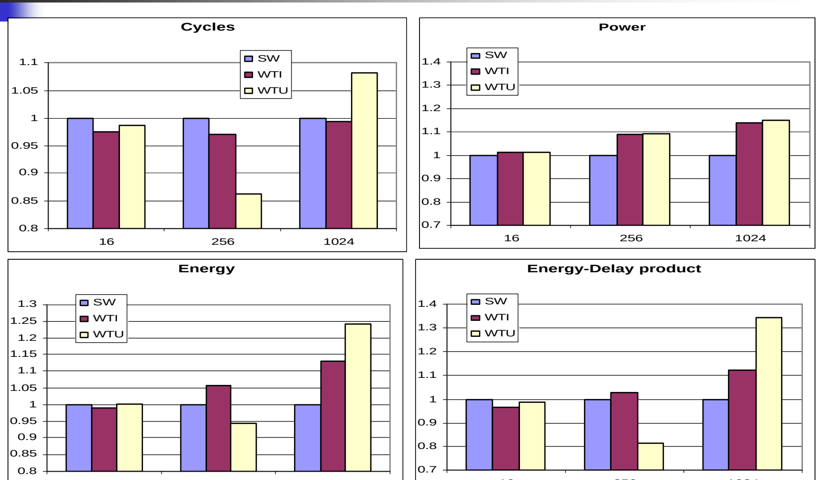


*cannot be a generic interconnect!

Readers-writers: varying cache size



Readers-writers: varying buffer size



Conclusions

- Evolutionary shift from bus-based interconnect to NoCs
  - Well underway (there's no stopping now)
  - Methodology/tooling is the main issue
- Platform challenges
  - Programming abstraction
  - HW/SW tradeoffs in session layer support