# Analyzing the MPSoC Design Space: the MPARM environment

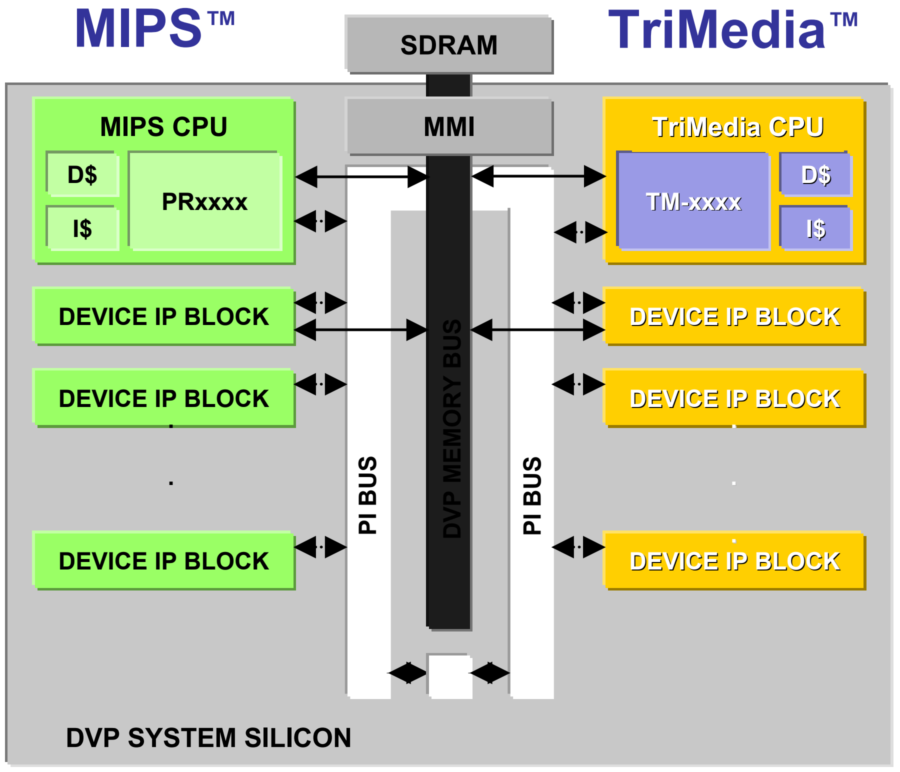
Davide Bertozzi
Luca Benini
DEIS University of Bologna
{dbertozzi-lbenini}@deis.unibo.it

# A SoC Platform: Nexperia™ DVP

**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**

**…and more**



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

**Nexperia™ System Buses**
- **32-128 bit**

- Does it look familiar?
  - Think about TI, Motorola…

## MP-SoC DESIGN

- Plenty of cores can be integrated
  - Current silicon technology allows ~100s
  - Near future allows ~1000s
- Emergence of heterogeneous embedded processors
- IP design reuse to increase SoC productivity
- Orthogonalization of communication versus computation
  - Standard interface sockets (OCP, VSI)
- Communication bottleneck
  - Need for system interconnect scalability and modularity

# Interconnect delay problem

| **Operation** | **Delay** (0.13um) | (0.05um) |
|---|---|---|
| 32b ALU Operation | 650ps | 250ps |
| 32b Register Read | 325ps | 125ps |
| Read 32b from 8KB RAM | 780ps | 300ps |
| Transfer 32b across chip (10mm) | 1400ps | 2300ps |
| Transfer 32b across chip (20mm) | 2800ps | 4600ps |

Taken from W.J. Dally presentation: Computer architecture is all about interconnect (it is now and it will be more so in 2010) HPCA Panel February 4, 2002

- Global on-chip comm. to operation delay 2:1, will be 9:1 in 2010
- Latency for across chip communication: 6-10 clock cycles (50 nm)
- Fraction of chip area reachable in 1 clock cycle: 0.4-1.4%

***Effect on the performance of the overall design***

## Outline

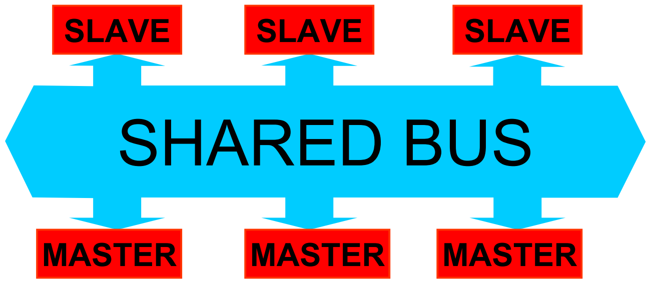
- MPSoC design challenges
  - ➡ Hardware
  - Software
- Introduction to MPARM
- MPARM at work

## On-Chip communication: design objectives

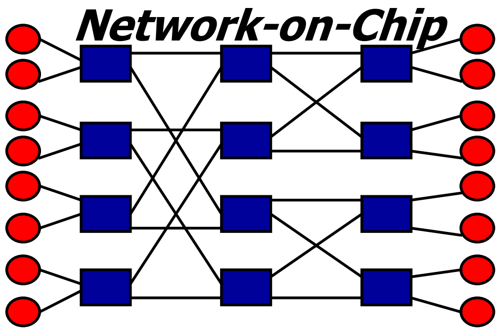
- Overcomes technology hurdles and become key differentiating factor
  - Low communication latency
  - High communication bandwidth
  - Low energy consumption
- Support applications
  - Sustainable scalability
  - High system-level reliability
  - High system behavior predictability

# Communication architectures

- **SHARED BUS** (broadcast)
  - Low area
  - Poor scalability
  - High energy consumption



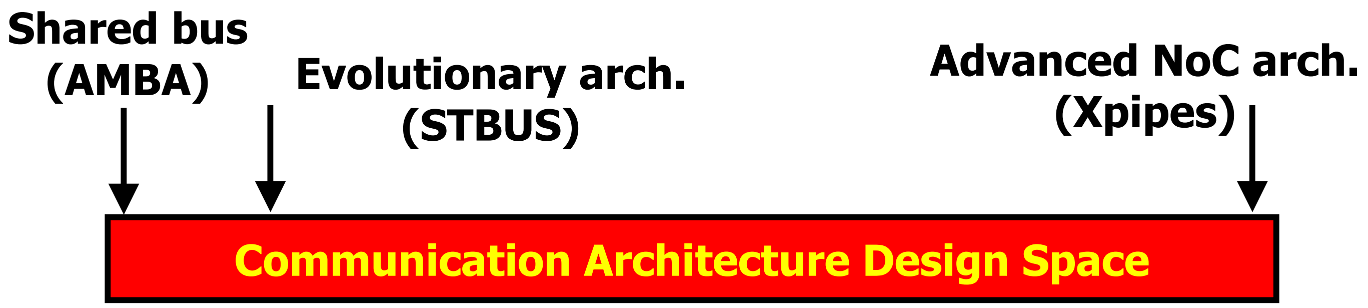
- **NETWORK on CHIP** (reduce sharing)
  - Scalability and modularity
  - Low energy consumption
  - Increase of design complexity



**These are extreme points of a wide design space**

# Approach to design space analysis

- Realization of meaningful points in the design space
  - Modelling accuracy emphasized


Shared bus (AMBA)
Evolutionary arch. (STBUS)
Advanced NoC arch. (Xpipes)
Communication Architecture Design Space

- Multiprocessor SoC simulation environment (MPARM)
  - Porting of the different SoC interconnects

- Performance analysis of on-chip interconnects for different:
  - Classes of applications *(computation vs communication dominated)*
    - Software architectures *(stand-alone vs OS supported appln)*
      - System configurations *(cache size, memory latency, ..)*

# Proposed NoC architectures

***From shared busses to network-on-chip architectures***

- Evolutionary communication architectures
  - *Sonics Micronetwork*
  - *STBus* Interconnect

- More radical solutions required in the long term
  - Early work: *Maia* heterogeneous architecture
  - *Tile-based* architecture (Dally and Lacy)
  - *Nostrum*
  - *HiNoC*
  - *Linkoeping SoCBUS*
  - *SPIN*
  - *Star-connected on-chip network*
  - *Aethereal*
  - *Proteo*

## XPIPES: Features

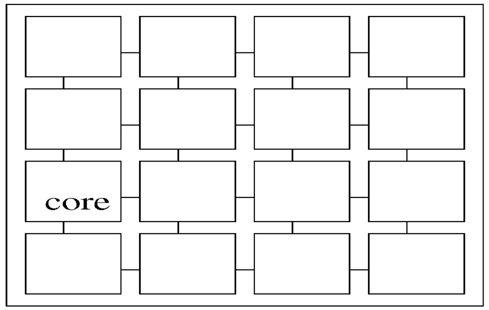
- Parameterizable network building blocks
  - Plug-and-play composable for arbitrary network topology
  - Design time tunable
- Pipelined links
- Source based routing
  - Street sign routing
  - Very high performance switch design
- Wormhole switching
  - Minimize buffering area while reducing latency
- Standard OCP interface

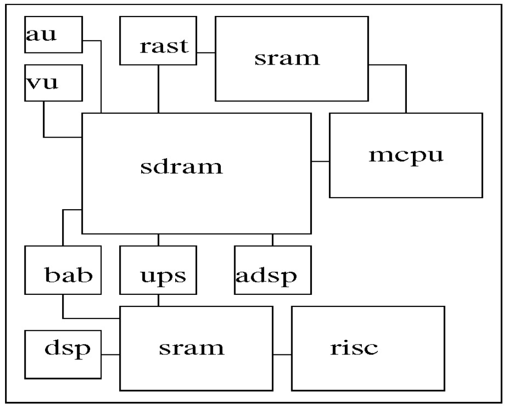
**Written in synthesizable SystemC at the cycle accurate level**

# 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

## Soft macros

***LIBRARY OF DESIGN TIME***
***TUNABLE AND COMPOSABLE SOFT MACROS***

GLOBAL PARAMETERS

- ✓ *Flit size*
- ✓ *Degree of redundancy in error detecting codes*
- ✓ *Address space of cores*
- ✓ *Number of bits for end-to-end flow control*
- ✓ *Number of flit types*
- ✓ *Max no. of hops between any two nodes*
- ✓ *Number of packet types*

## Soft macros

***LIBRARY OF DESIGN TIME***
***TUNABLE AND COMPOSABLE SOFT MACROS***

BLOCK-SPECIFIC PARAMETERS

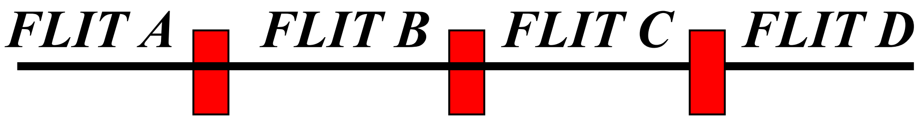
NETWORK INTERFACE
- ✓ *Interface parameters (nr of data/address lines, max. burst length)*
- ✓ *Type of interface (master, slave, or both)*
- ✓ *Flit buffer size in the output port*
- ✓ *Content of routing table*

SWITCH
- ✓ *Nr of I/O ports*
- ✓ *Nr of virtual channels*
- ✓ *Link buffer size*

LINK
- ✓ *Nr of pipeline stages*

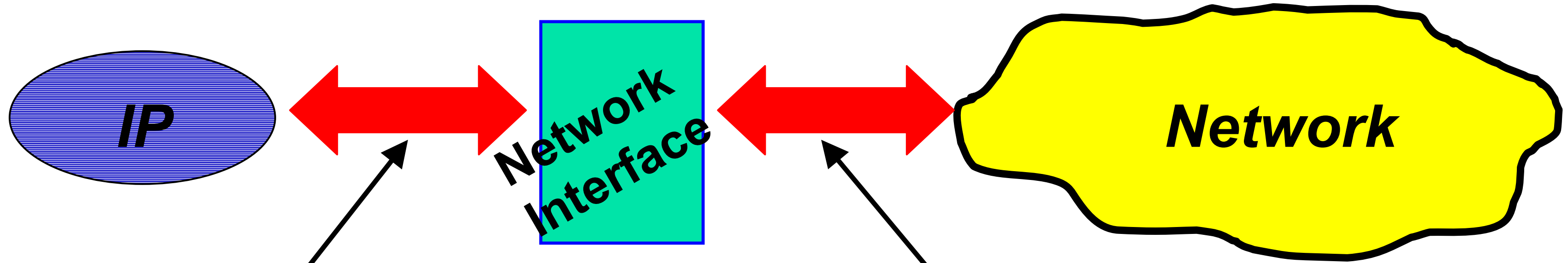
# Link delay bottleneck

- Wire delay is serious concern for NoC Links
  - If NoC “beat” is determined by worst case link delay, performance can be severely limited
- Pipelined links
  - Delay is transformed in Latency
  - Data introduction rate is not bound by link delay any more



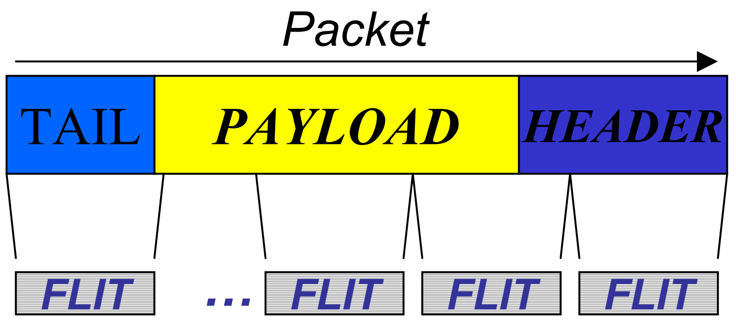
  - Switch operation must be latency-insensitive
    - ✓ Buffering requirements

## Network interface



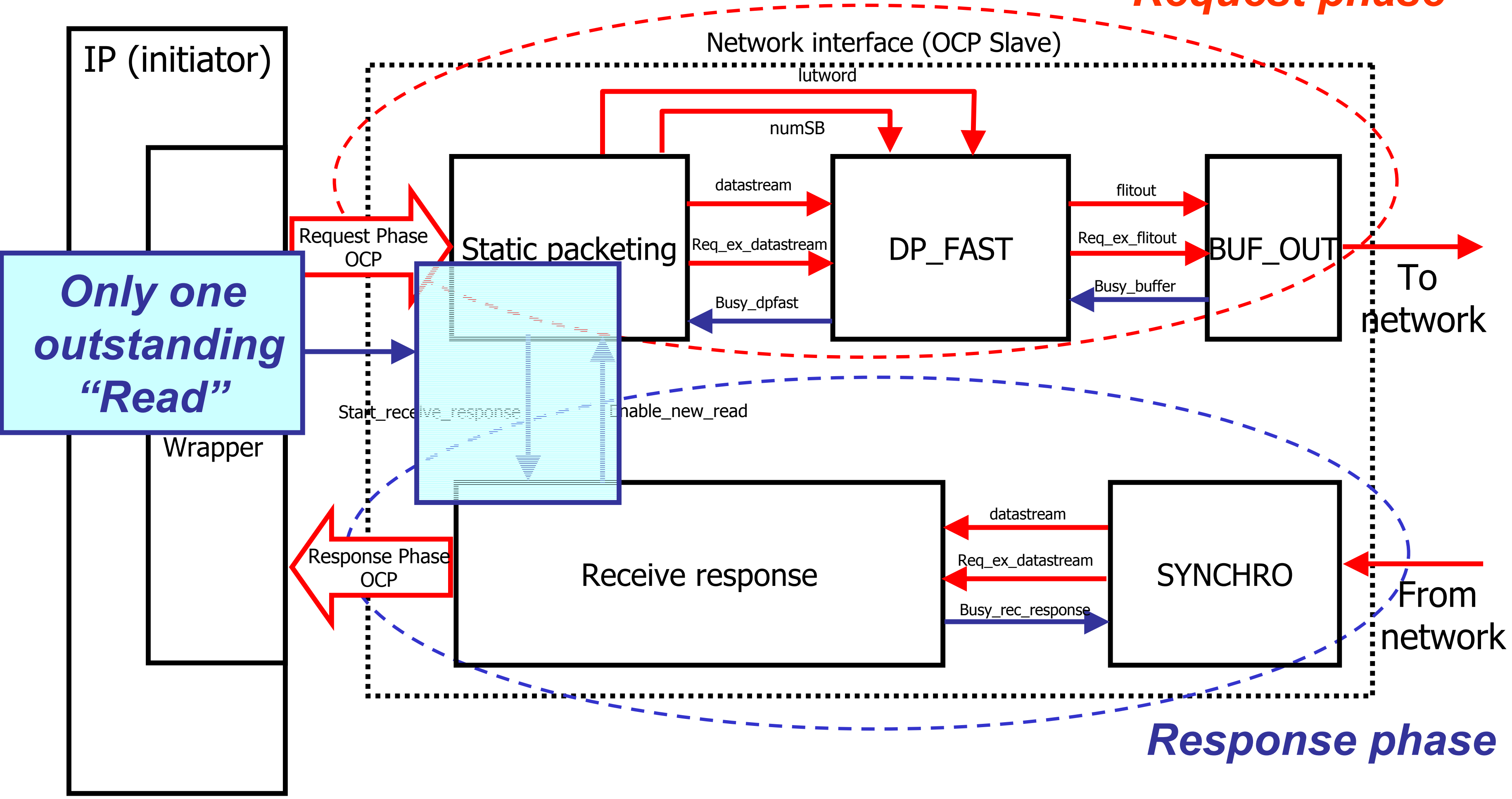
**Open Core Protocol (OCP)**
*End-to-end communication protocol*
- *pipelining*
- *independence of request/response phase*

**Network protocol**



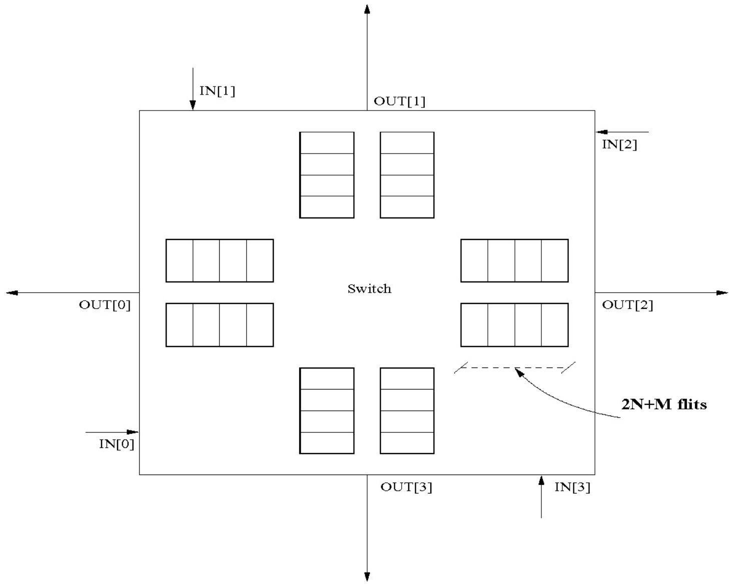
Header includes:
✓Path across the network
✓Source
✓Destination
✓Command type
✓Burst ID (MBurst)
✓Packet identifier within message (ID-PACKET)
✓ Local target IP address (IP_ADDR)

# NI Architecture



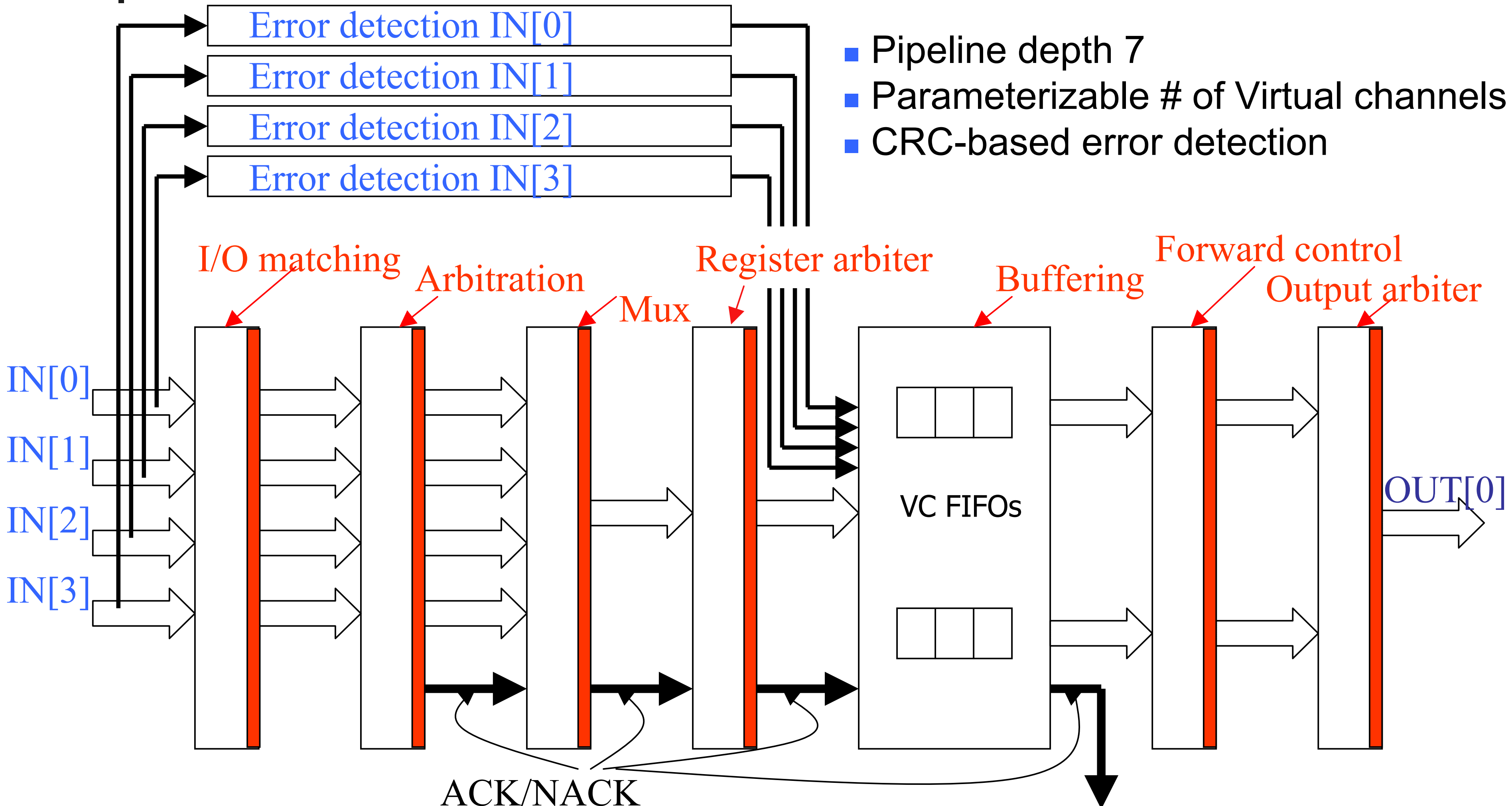
# Switch architecture

- Highly parameterized
- Buffering at the outputs with virtual channel support
- Deeply pipelined architecture
- Forward control flow (ACK/NACK protocol)
- Distributed error detection logic



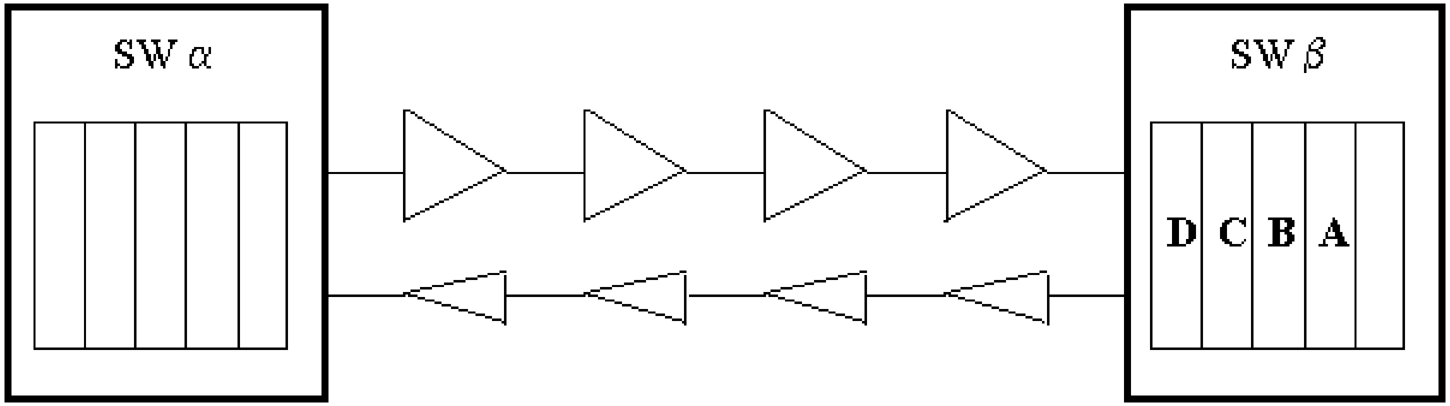
- Aggregate bandwidth:
  64 Gbit/s (32 bit links, 500 MHz)
- Estimated switch area (0.10 um)
  0.33 mm2

# Output port architecture



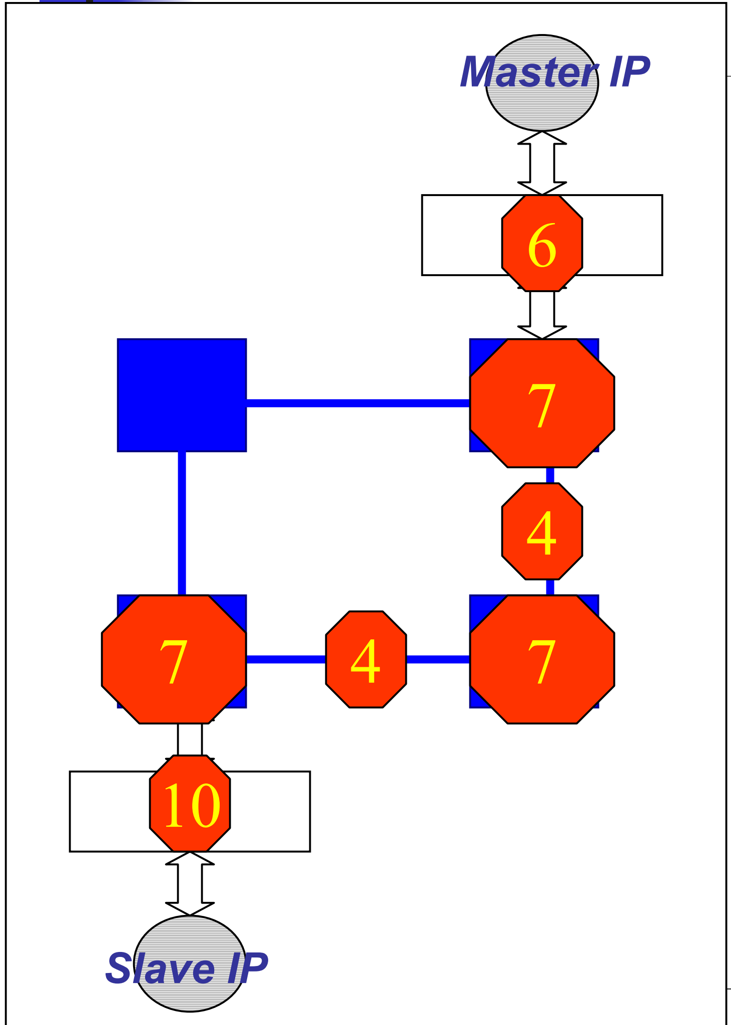
- Pipeline depth 7
- Parameterizable # of Virtual channels
- CRC-based error detection

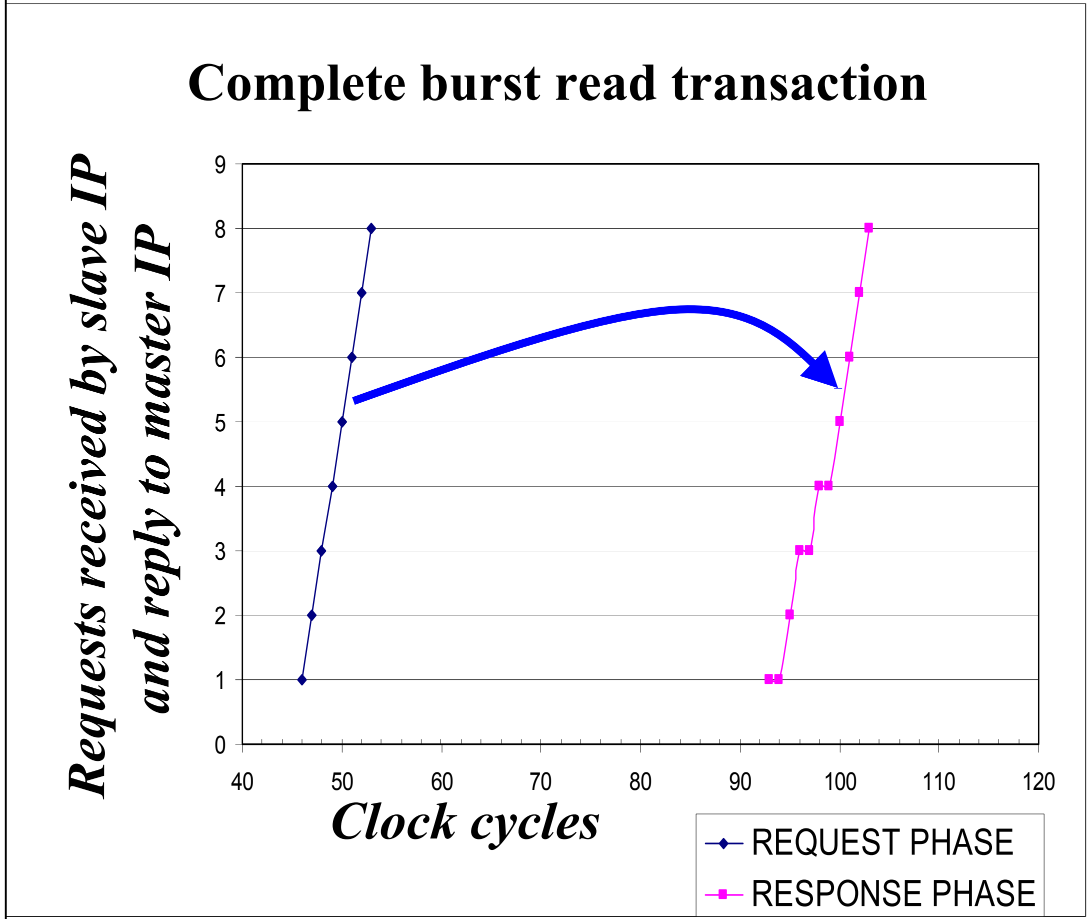
## Flow control



- Transmission
- ACK and buffering
- NACK
- ACK/NACK propagation
- Memory deallocation
- Retransmission
- Go-back-N

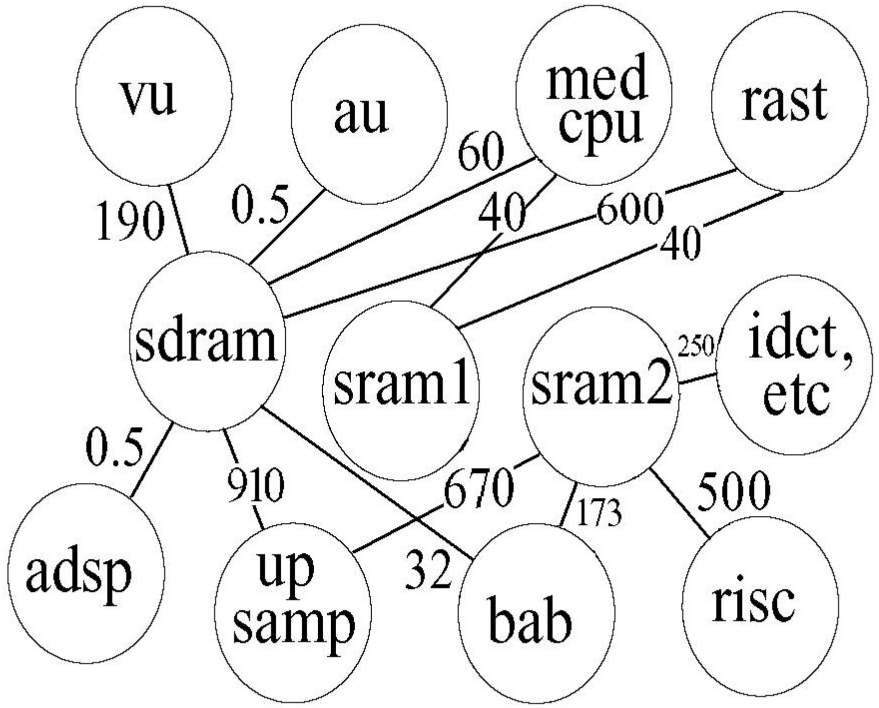
# Network throughput and latency



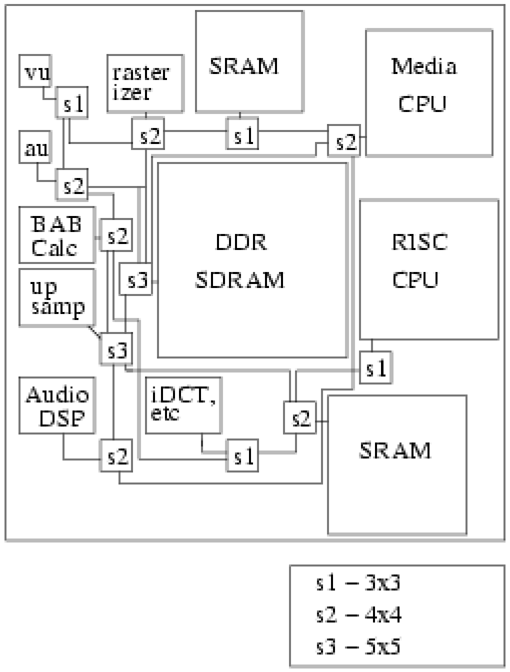


# Example: MPEG4 decoder

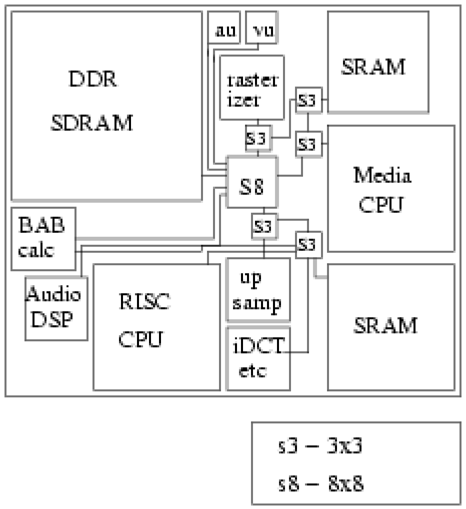
- Core graph representation with annotated average communication requirements



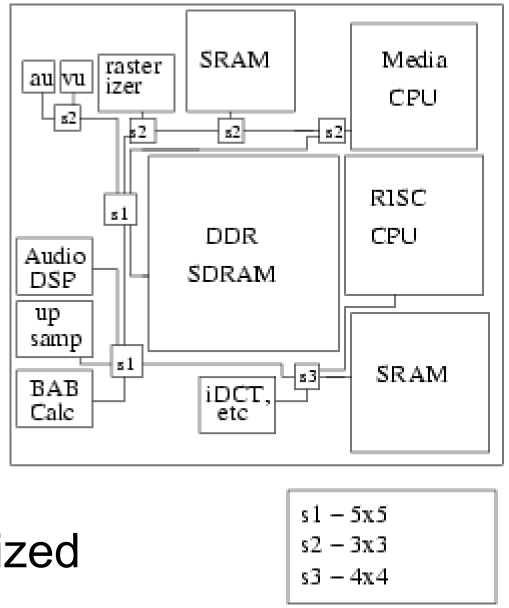
# NoC Floorplans



- General purpose: mesh

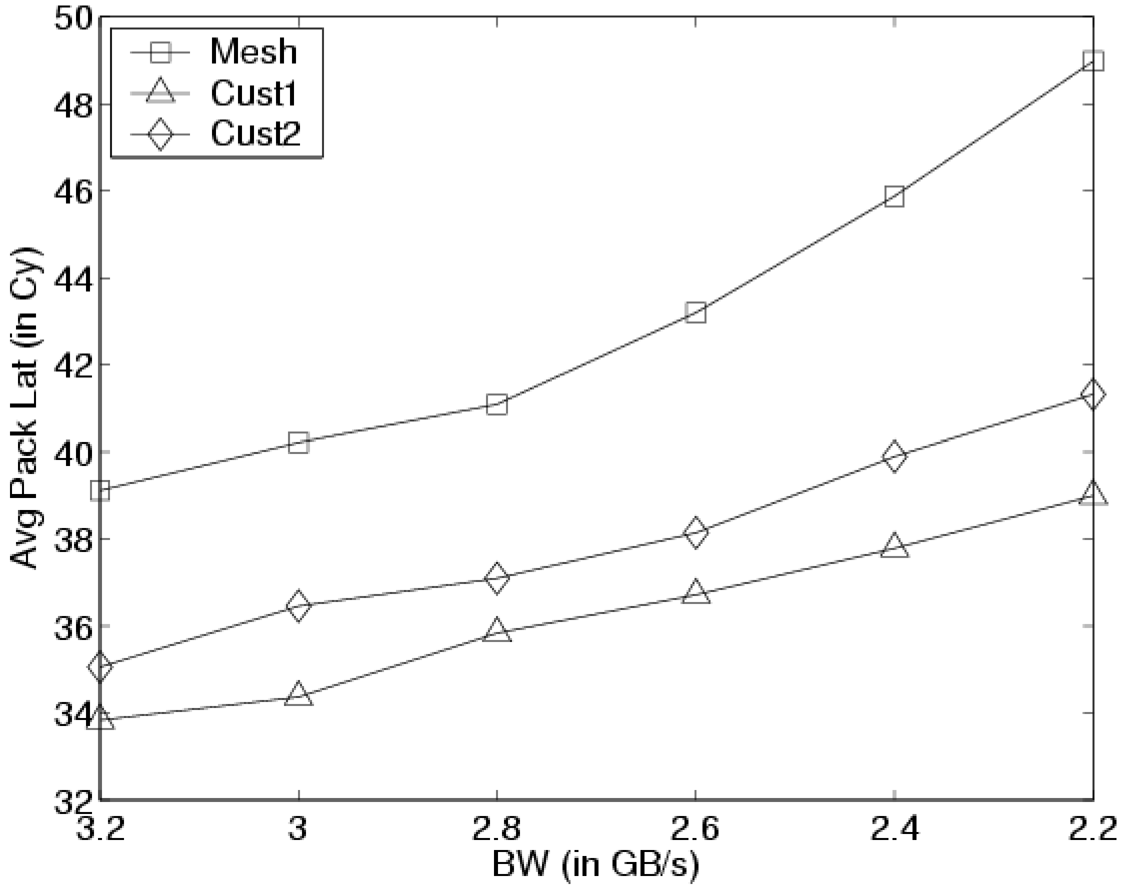


- Application specific: centralized



- Application specific: distributed

## Performance, area and power



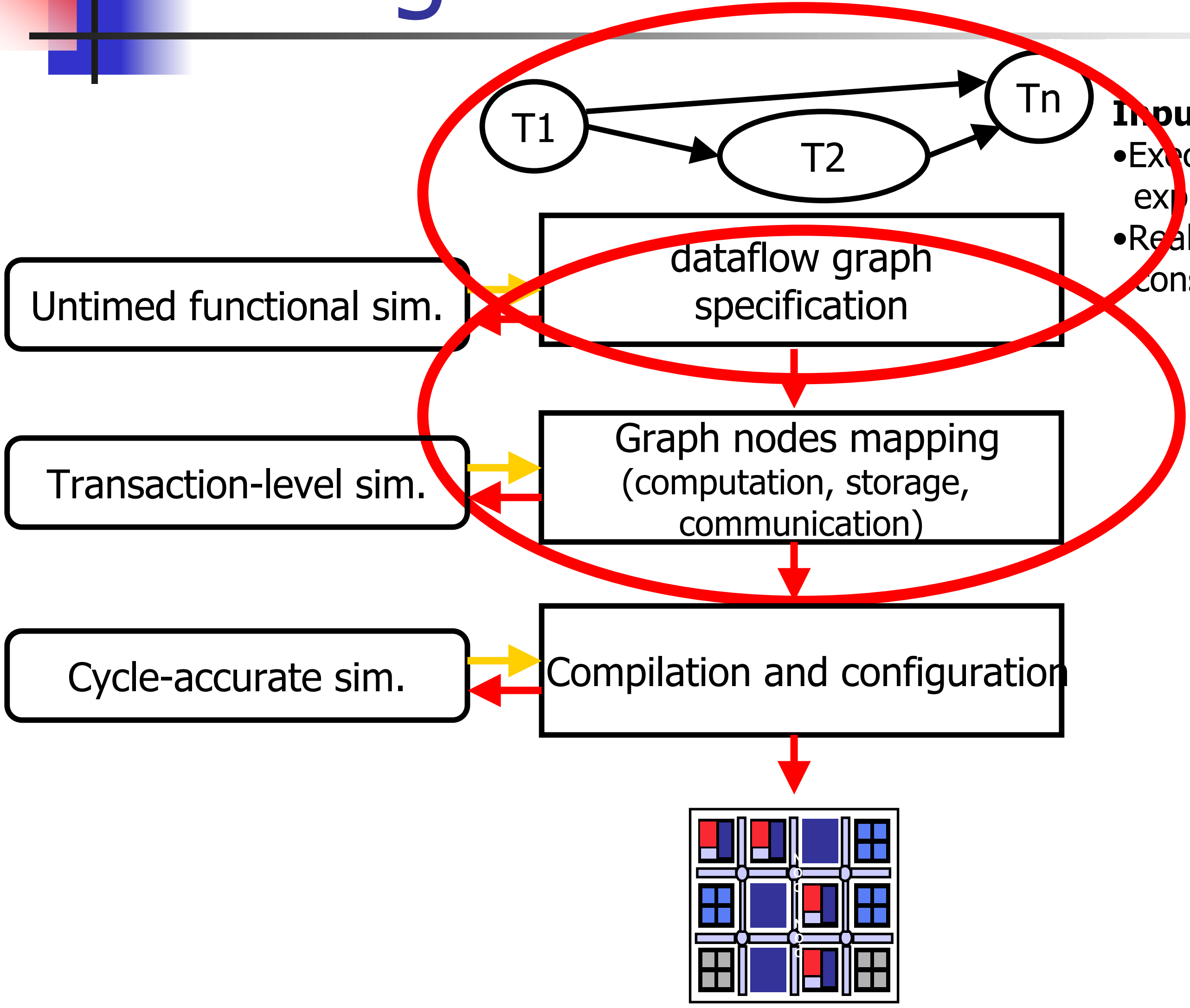
Less latency and better Scalability of custom NoCs

- 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

## Outline

- MPSoC design challenges
  - Hardware
  - ➡ Software
- Introduction to MPARM
- MPARM at work

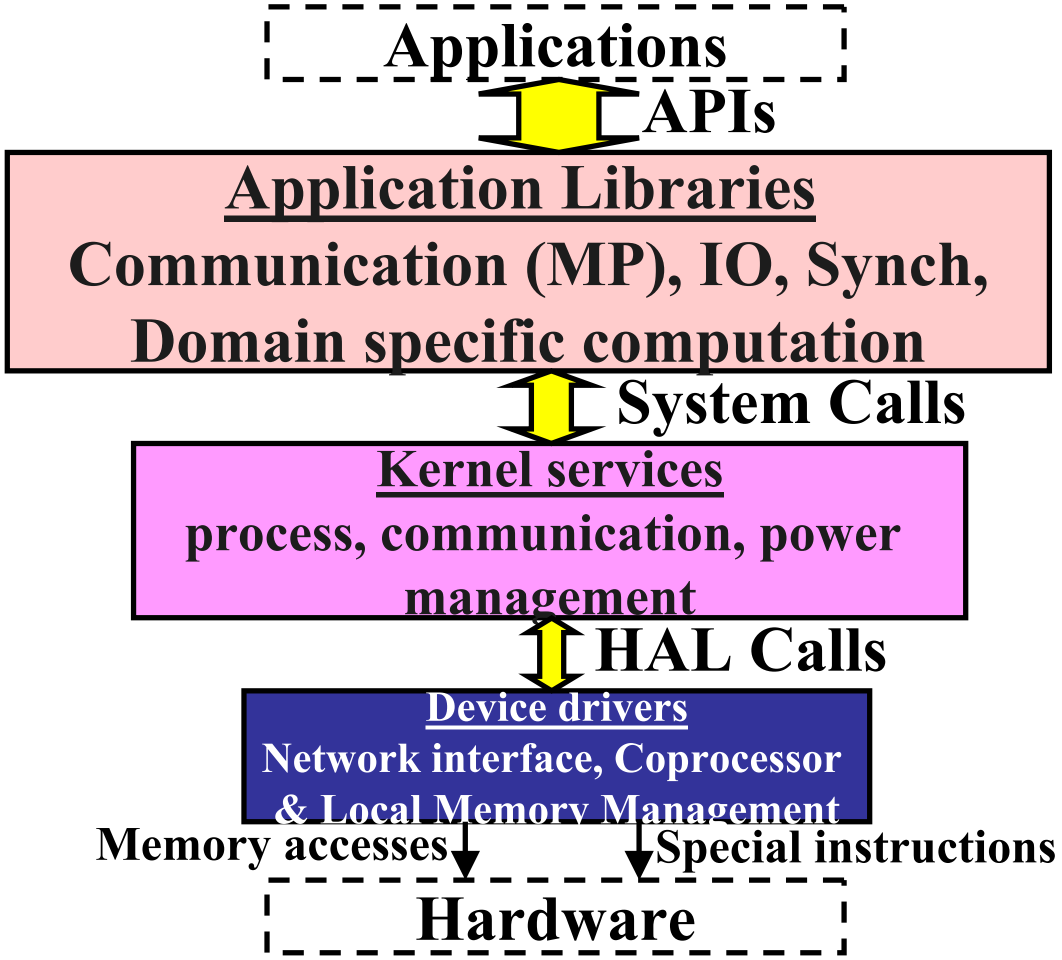
# Design flow and verification



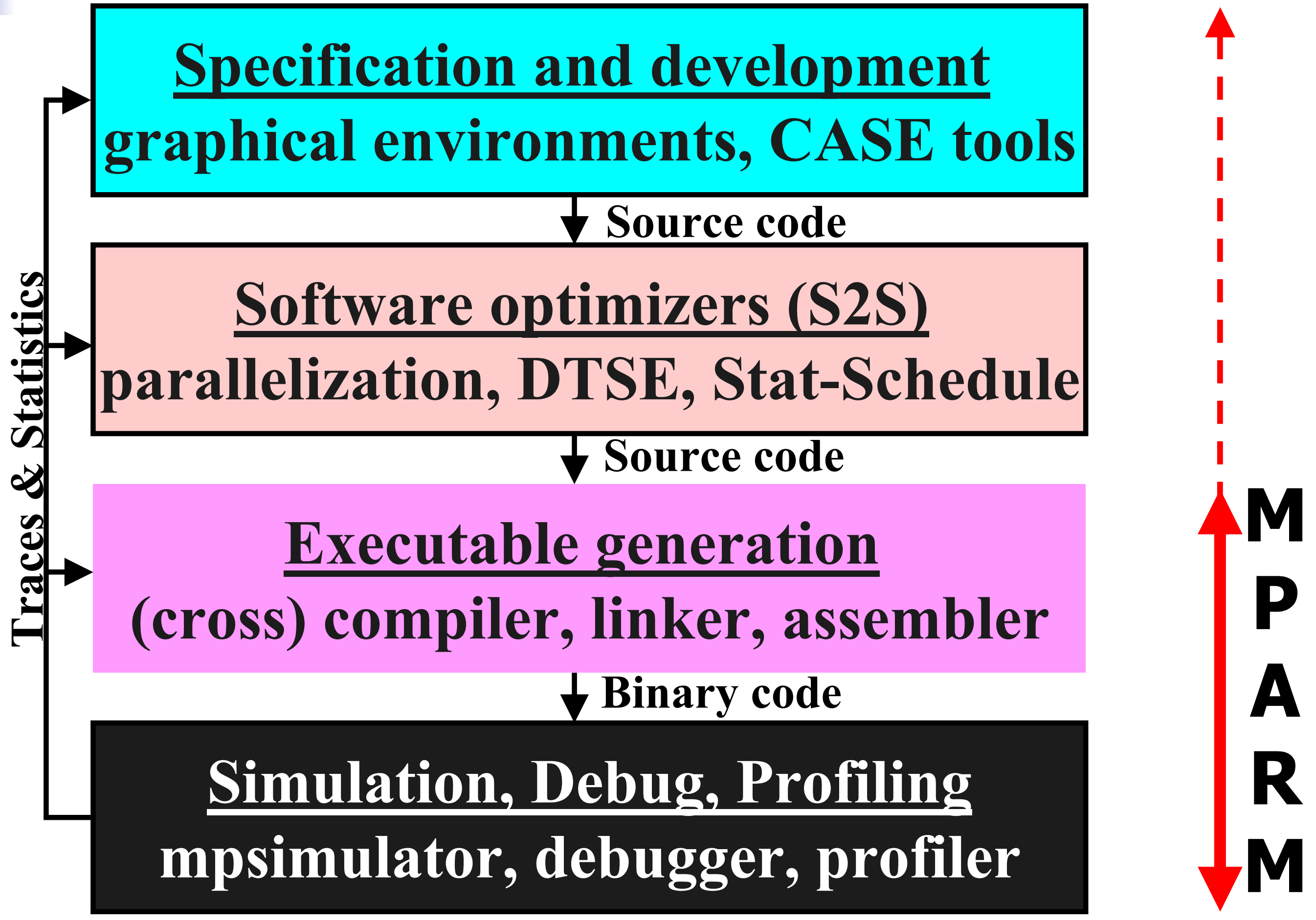
T1
T2
Tn
dataflow graph specification
Untimed functional sim.
Graph nodes mapping (computation, storage, communication)
Transaction-level sim.
Compilation and configuration
Cycle-accurate sim.

**Inputs:**
- Executable specification with explicit parallelism (e.g. task graph)
- Real-time (QoS), power constraints / objective function

# Middleware Architecture


Applications
APIs
Application Libraries
Communication (MP), IO, Synch,
Domain specific computation
System Calls
Kernel services
process, communication, power management
HAL Calls
Device drivers
Network interface, Coprocessor & Local Memory Management
Memory accesses
Special instructions
Hardware

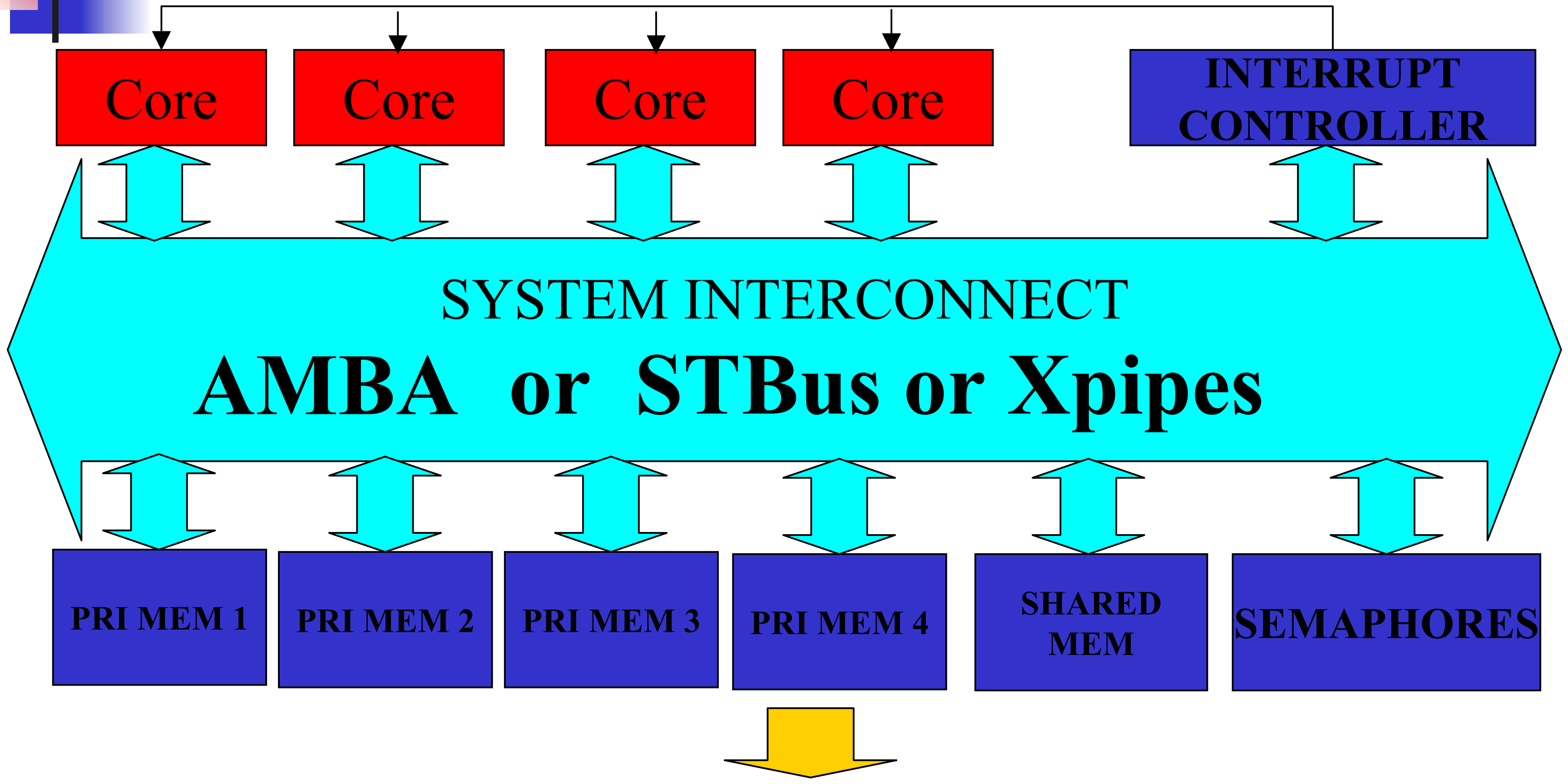
# Architecture of DD tools


Specification and development
graphical environments, CASE tools
Source code
Software optimizers (S2S)
parallelization, DTSE, Stat-Schedule
Source code
Executable generation
(cross) compiler, linker, assembler
Binary code
Simulation, Debug, Profiling
mpsimulator, debugger, profiler
Traces & Statistics
MPARM

## Outline

- MPSoC design challenges
  - Hardware
  - Software
- ➡ Introduction to MPARM
- MPARM at work

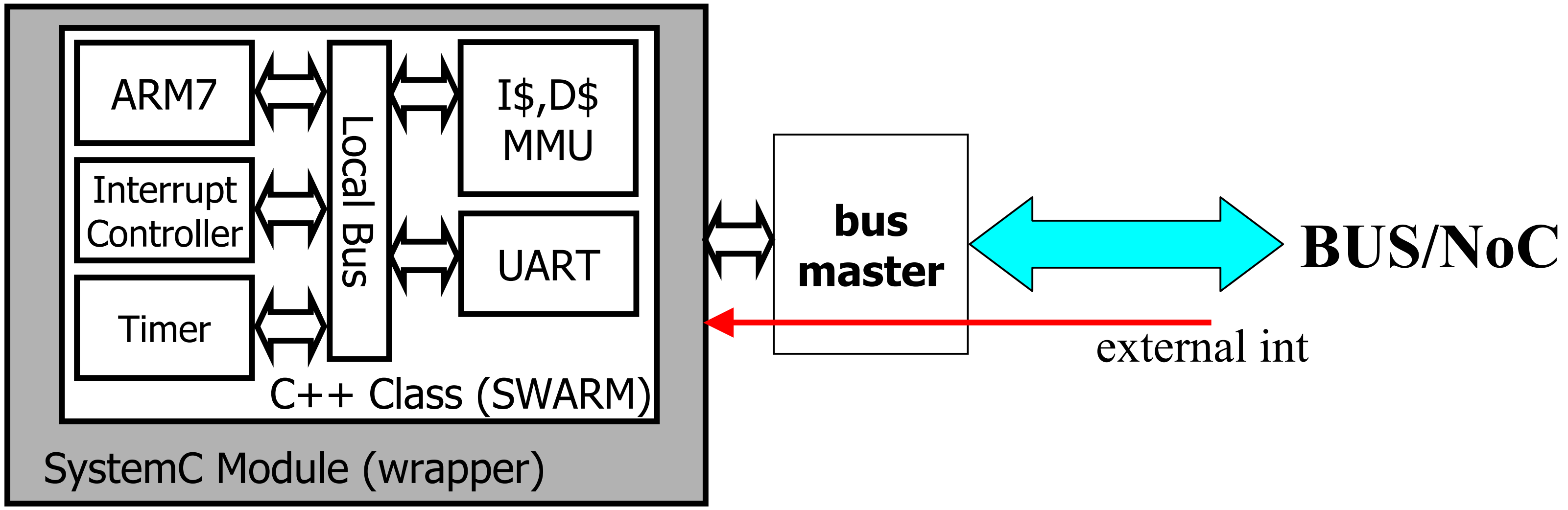
## Architecture



Simulation is cycle accurate

(~ 62-82 Kcycles/sec with 6 cores on a Pentium4 2.2 GHz)

# Processor core

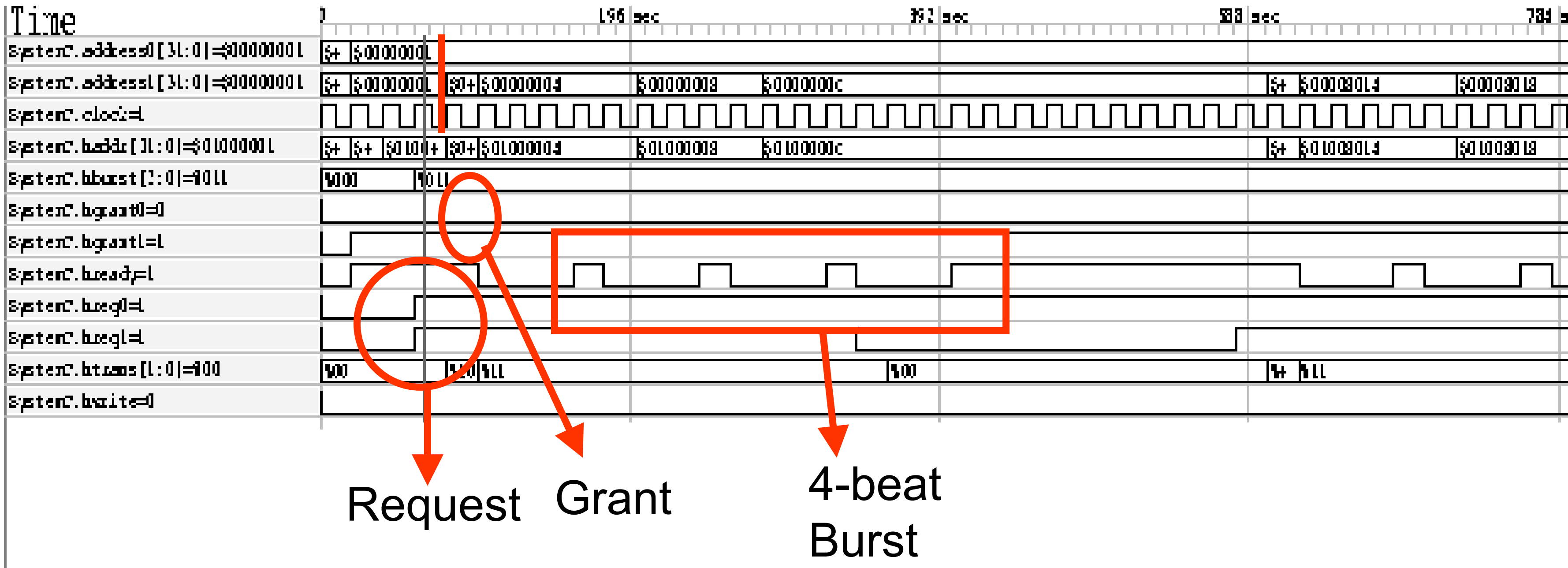


- I- and D-cache are modelled
- Hardware blocks for OS support: timers, IntCntrl, ..
- ISS instantiated as a C++ class

## Core wrapper

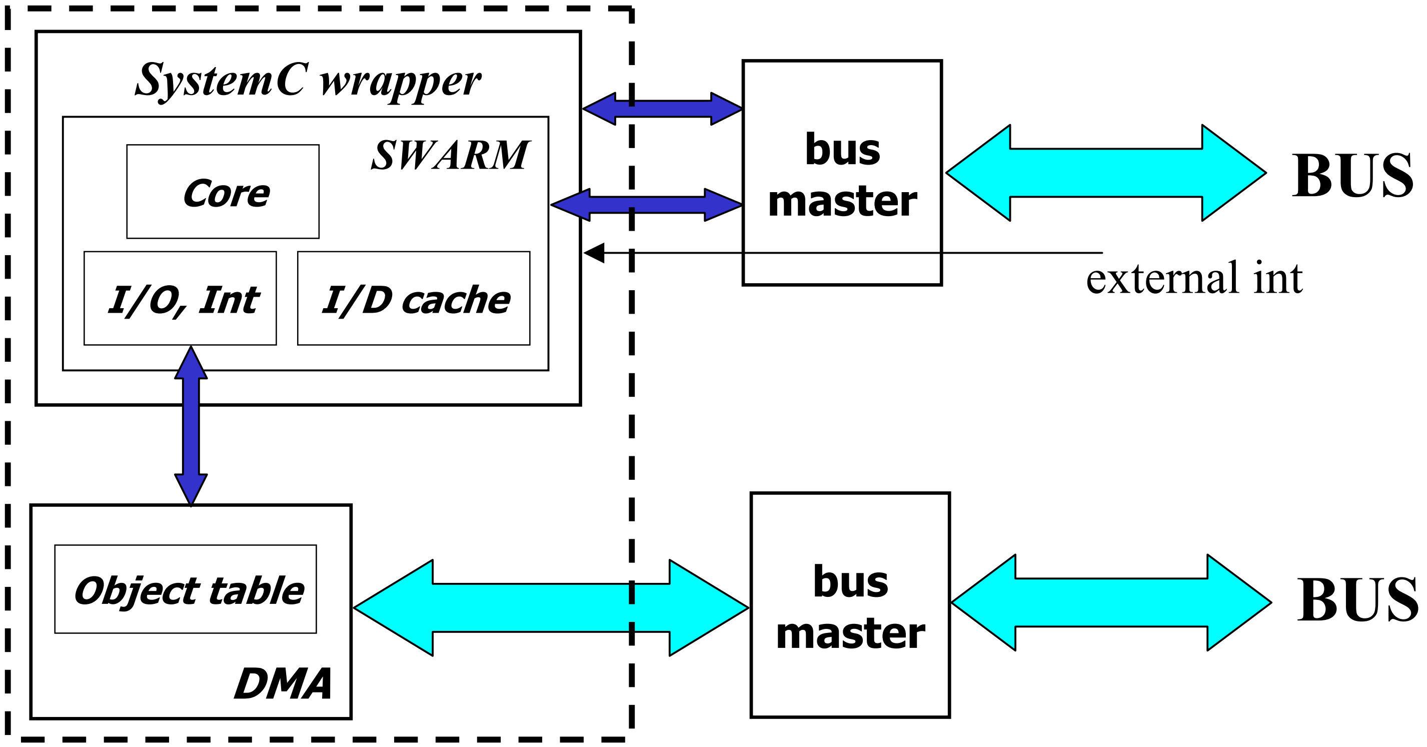
- Synchronizes the ISS with the system
  - Breaks the instruction execution loop on a cycle by-cycle basis
  - Checks and updates core boundary signals
  - Drives the bus interfaces
- It is the only core-specific simulation support block
  - Integration of new ISS in C++ is easy
  - More work for integrating C (e.g. simplescalar)

# Simulation Accuracy



- Cycle accurate and bus signal accurate simulation model
- Accuracy of core simulation depends on ISS

# Communication Assist in MPARM



- DMA engine programmed by writes to its address space
- Direct support to global objects and their application-controlled transfer

## Software Environment

- Requires programming abstractions and run-time support
- Porting of a parallel RTOS: RTEMS
  - Includes POSIX APIs
  - Multi-processor support
  - Multi-tasking support
  - Inter-processor synchronization and communication primitives

- Hardware support for message passing:
  - Shared memory provides hardware communication channel
  - Processor-dedicated interrupt slaves

Simulation time for booting a 5 processors system: 1 minute

# Analysis toolkit

- Simulation control
  - Profiling activation/deactivation/options
  - Based on pseudo-instruction (swi)
- Profiling support
  - In core (internal core events)
  - In bus master (on-chip communication)
  - New profiling capabilities can be easily programmed in C++
  - Waveform tracking is available
- Power modeling under way…

## Outline

- MPSoC design challenges
  - Hardware
  - Software
- Introduction to MPARM
- ➡ MPARM at work

## Traffic Modeling

*Traditionally used traffic models trade-off accuracy with complexity:*

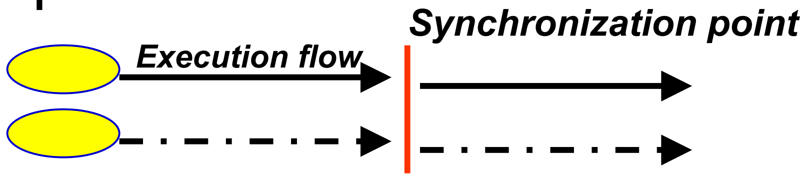
- ***Stochastic traffic models***
  - Analytical distributions
  - Easily parameterizable
- ***Trace-based models***
  - Higher accuracy
  - Does not consider dynamic traffic-dependent effects (e.g. inter-processor communication)
- ***Functional traffic***
  - Traffic directly generated by running applications
  - Requires OS support

***Complexity***
***Accuracy***

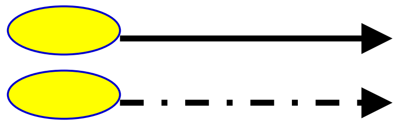
# Performance analysis

***We assessed performance in presence of 3 different communication patterns:***

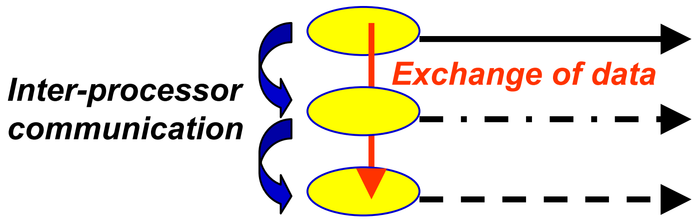
1. Mutually dependent tasks



2. Independent tasks

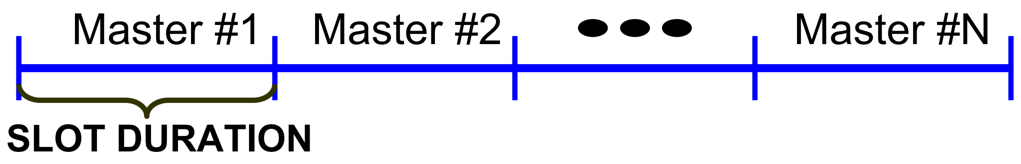


3. Pipelined tasks

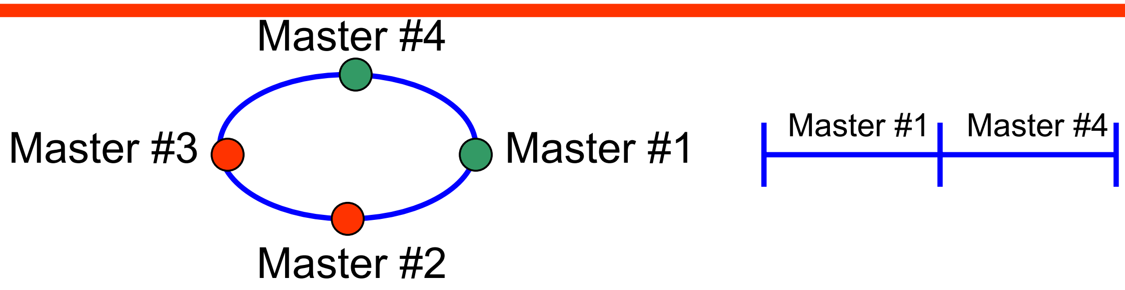


# Arbitration Policies

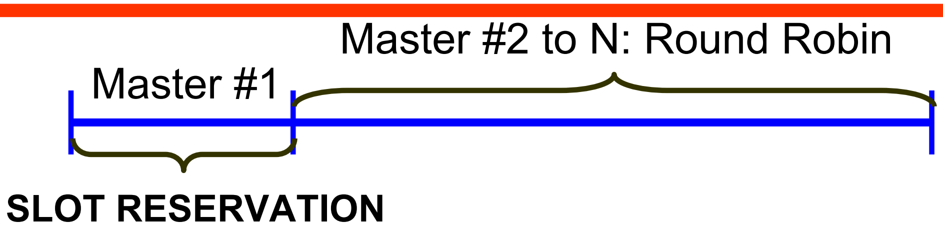
***TDMA***



***Round robin***



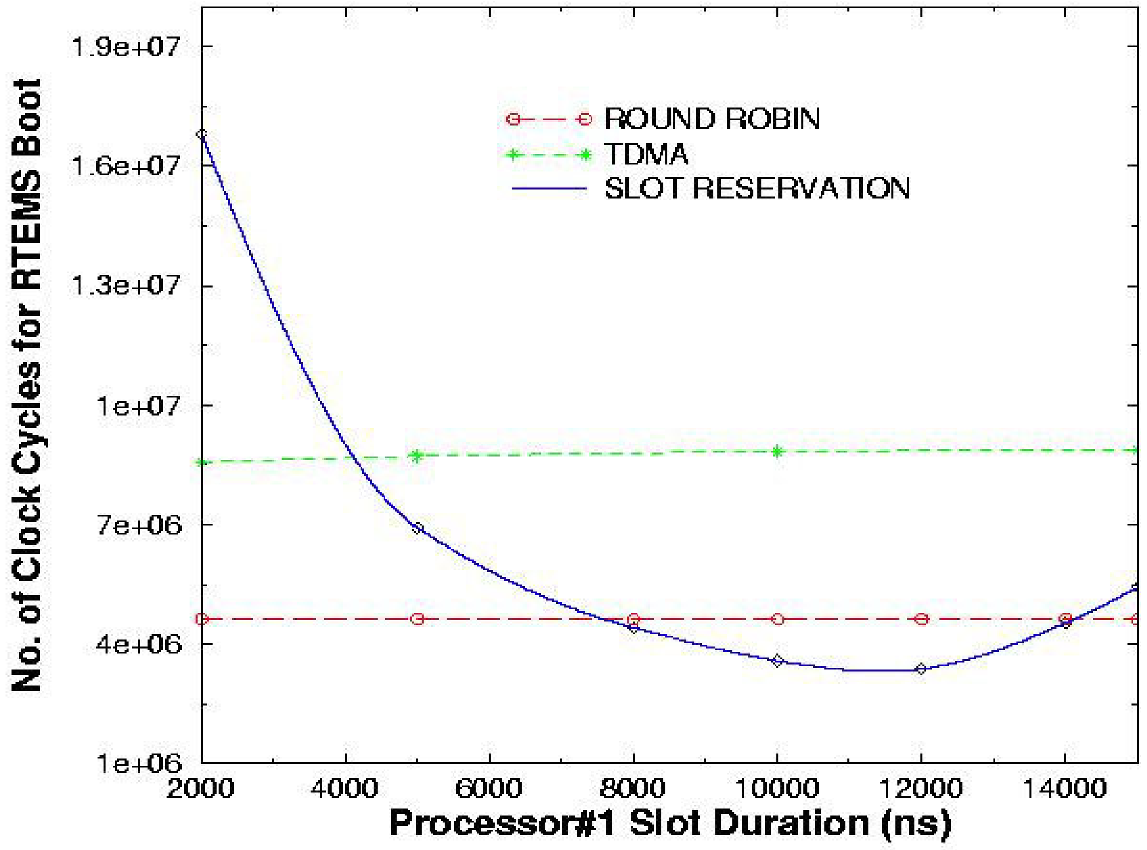
***Slot reservation***



**Goal**
performance investigation of the arbitration algorithms under different traffic patterns across the bus

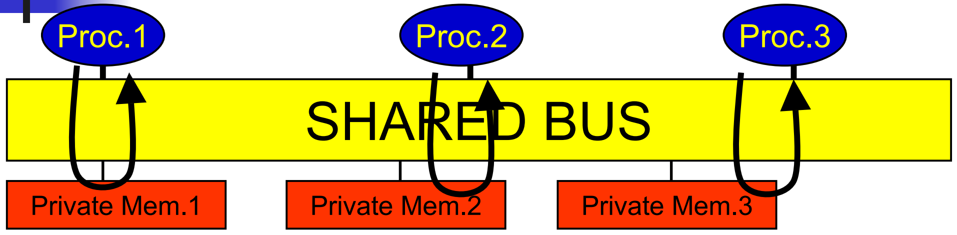
# Mutually dependent tasks

RTEMS bootstrap on 5 processors

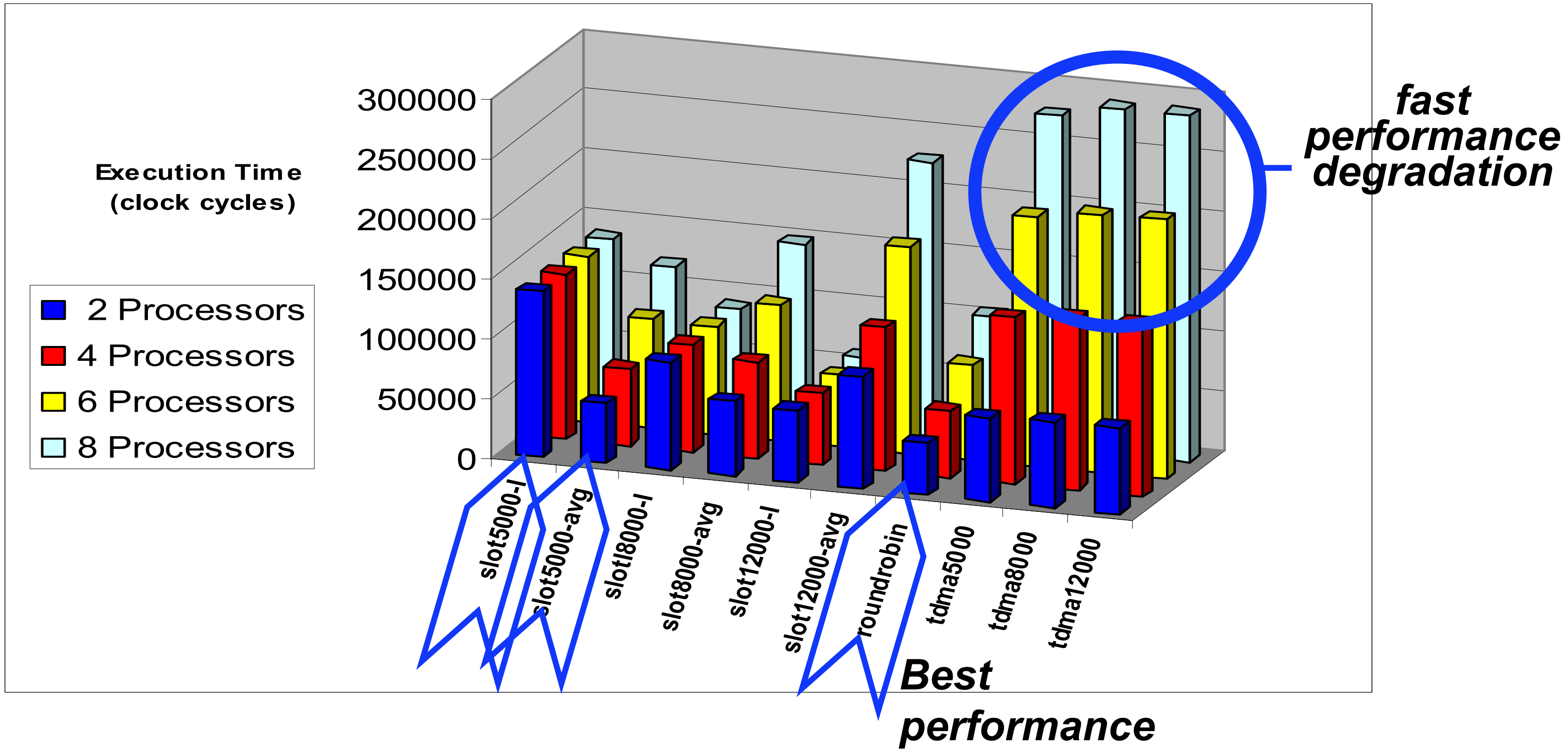


- Performance metric: ***Execution time***
- A synchronization point does exist
- Slot reservation makes up for the asymmetric workload of processors
- TDMA inefficiently allocates bandwidth

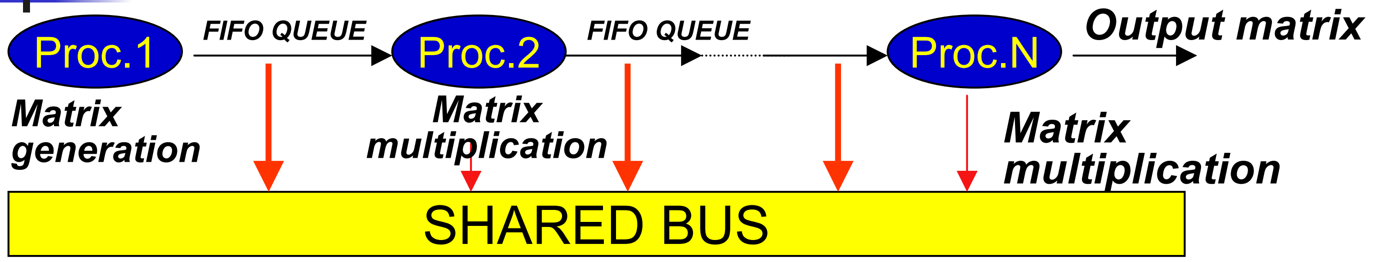
# Independent Tasks

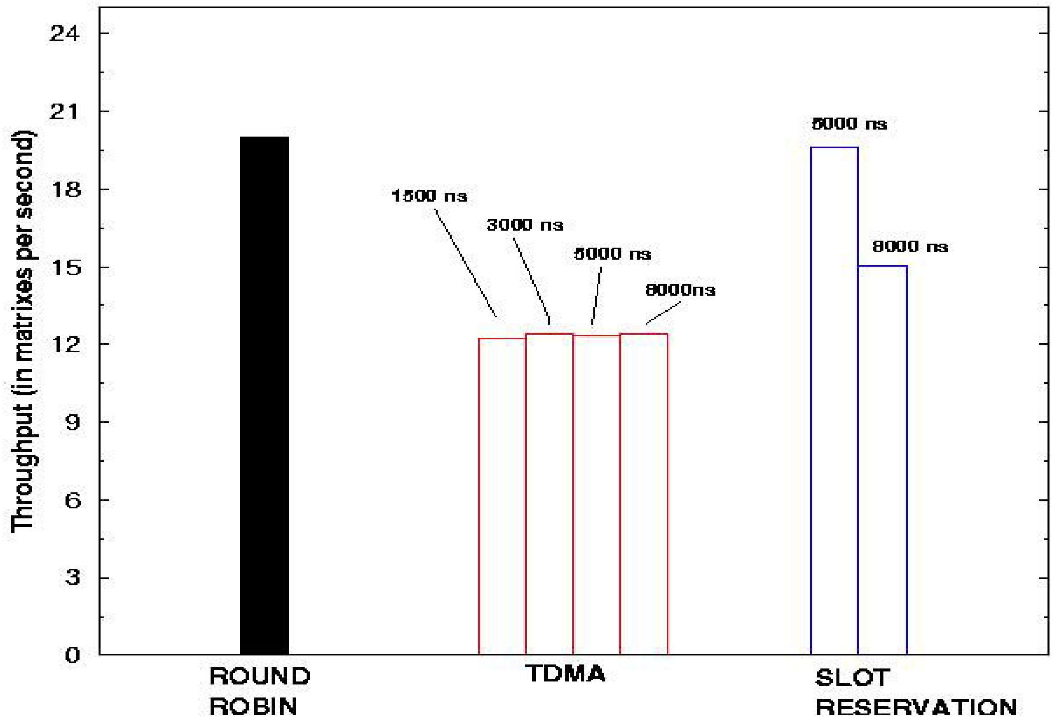


Performance metric:
<u>average execution time</u>



# Pipelined Tasks

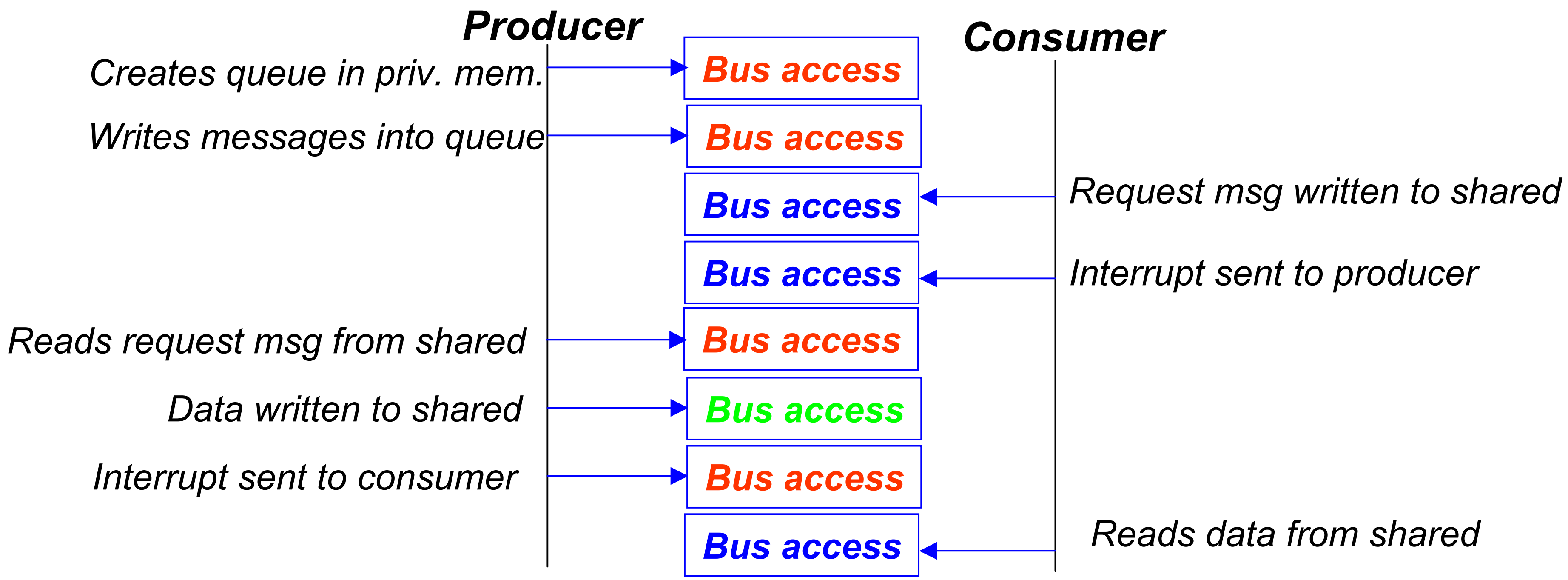




1. Performance metric: <u>throughput</u>
2. TDMA is surprisingly outperformed by round-robin

# TDMA Performance

***TDMA poor performance depends on hw/sw mismatch***


Producer
Consumer
Creates queue in priv. mem.
Bus access
Writes messages into queue
Bus access
Bus access
Request msg written to shared
Bus access
Interrupt sent to producer
Reads request msg from shared
Bus access
Data written to shared
Bus access
Interrupt sent to consumer
Bus access
Bus access
Reads data from shared

***This interactive handshake mechanism is inefficiently accommodated by a TDMA based arbitration policy***

## HW-SW Mismatch

***Application layer***

***RTEMS:***
***High level communication primitives***

***Low level implementation***

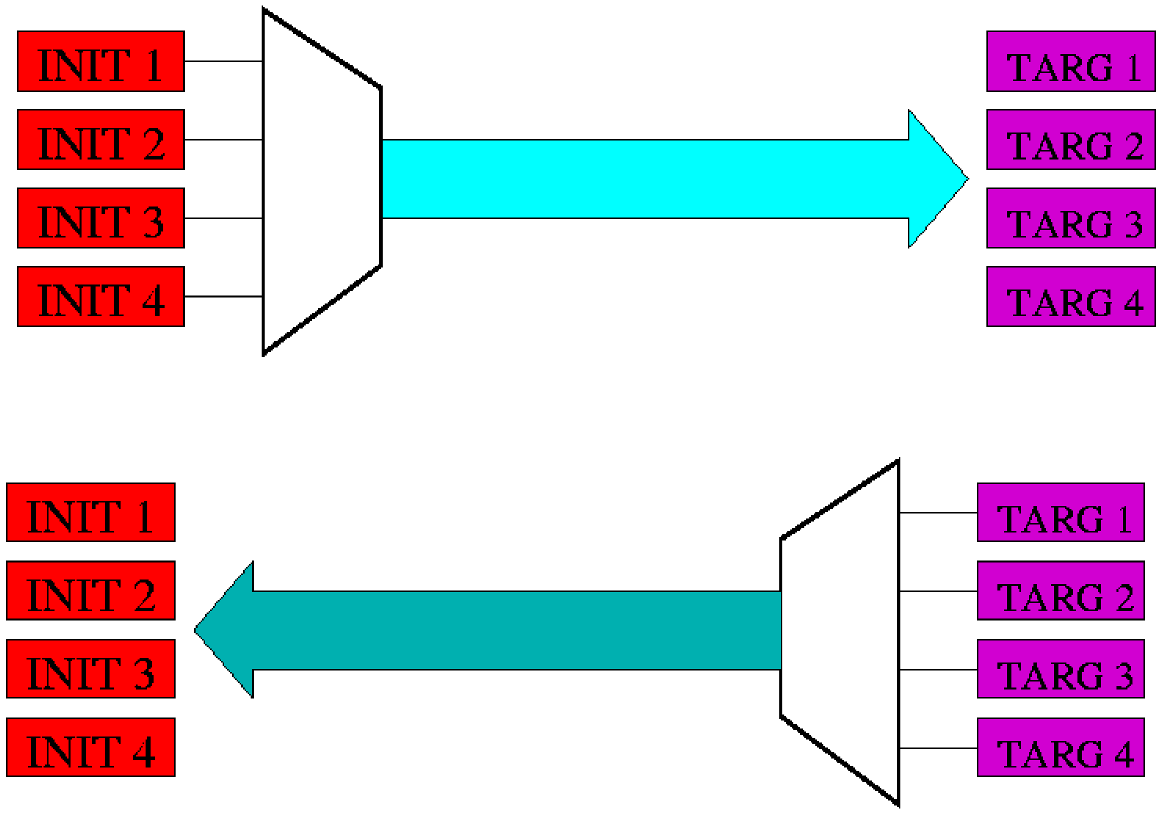
***PERFORMANCE PENALTY!!!!!***

***Software architecture should match the underlying hardware platform to preserve and maximize system performance***

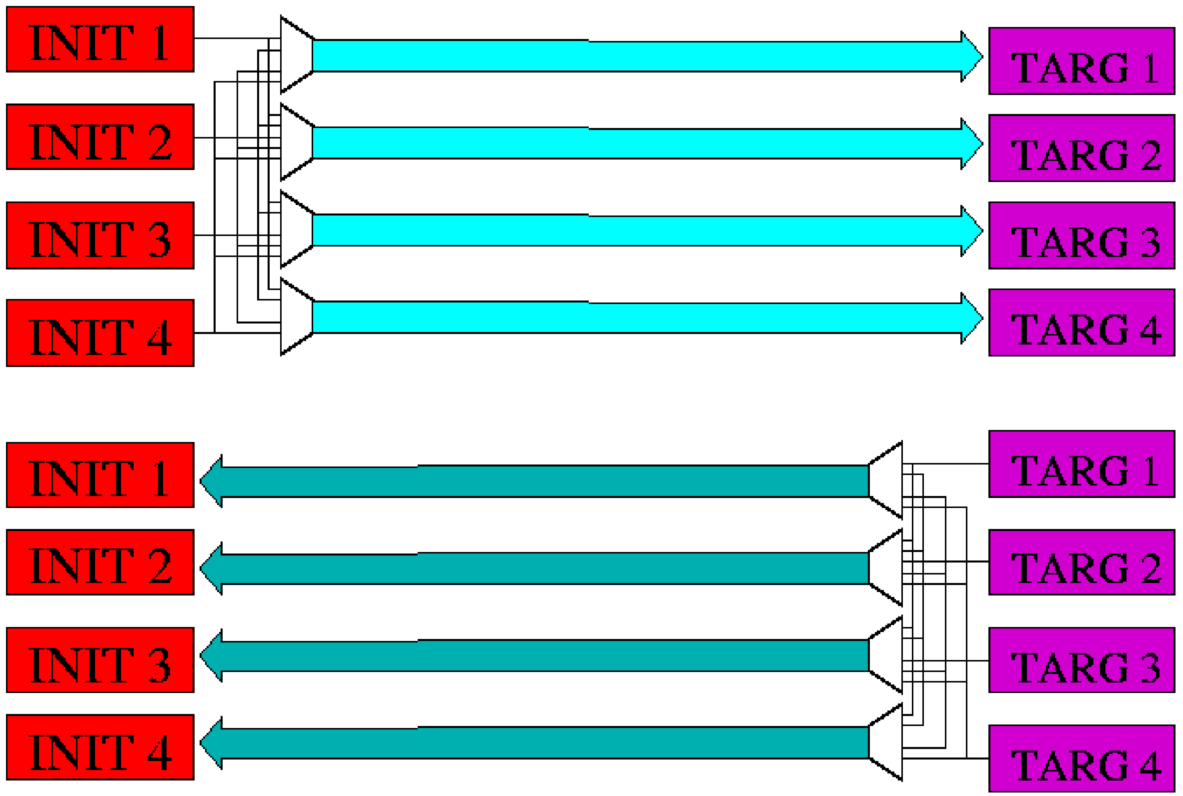
## Design space exploration

- Explore & compare different OCBs
- Identify key differences and performance bottlenecks
- Case study
  - AMBA vs. STBUS

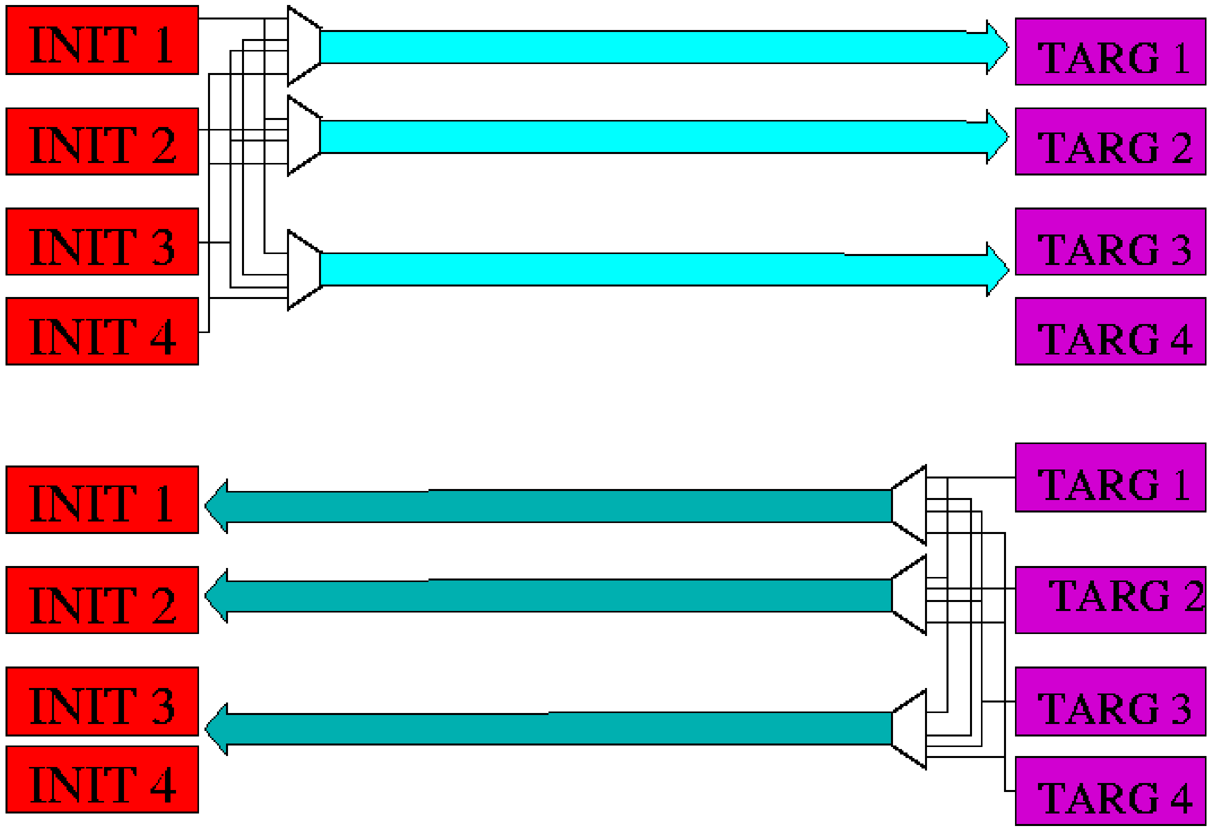
# STbus – Shared Bus



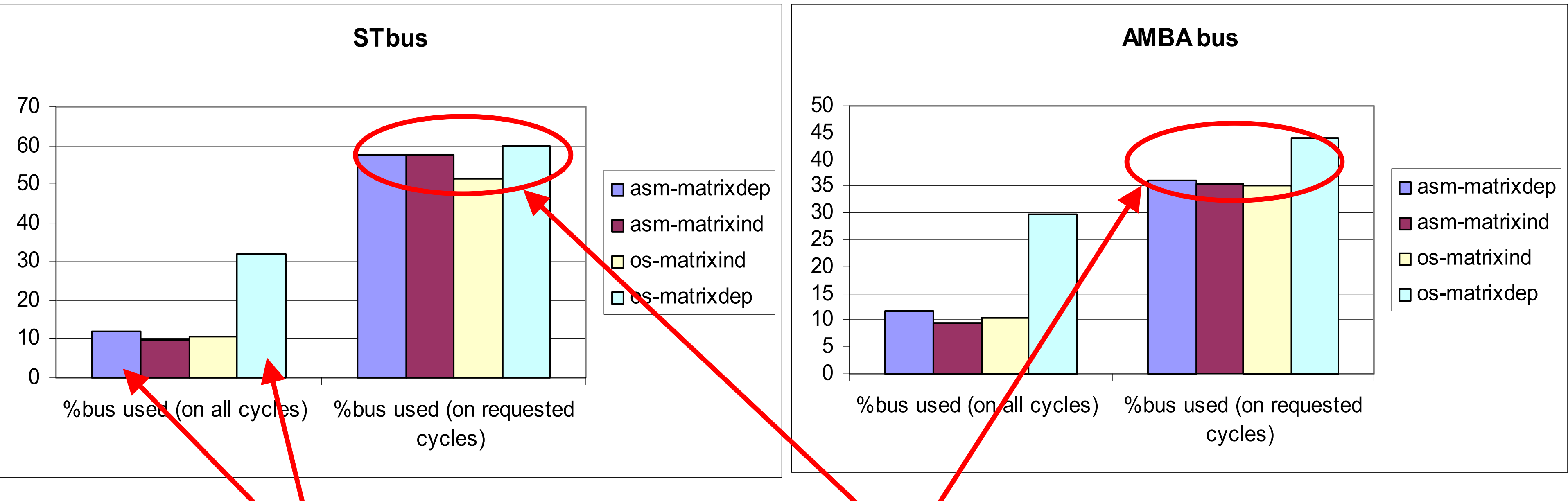
# STbus – Full Crossbar



# STbus – Partial Crossbar



# Comparison: bus traffic

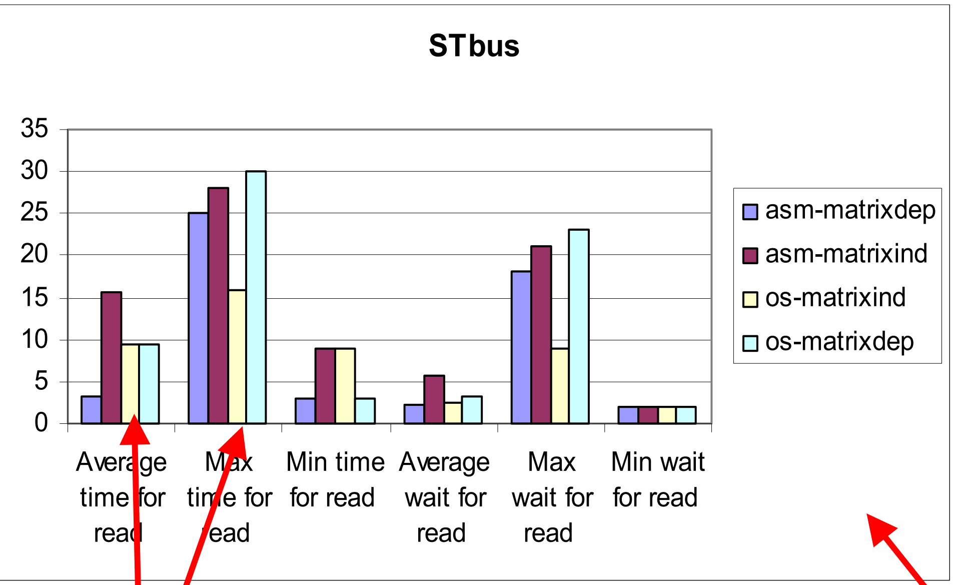


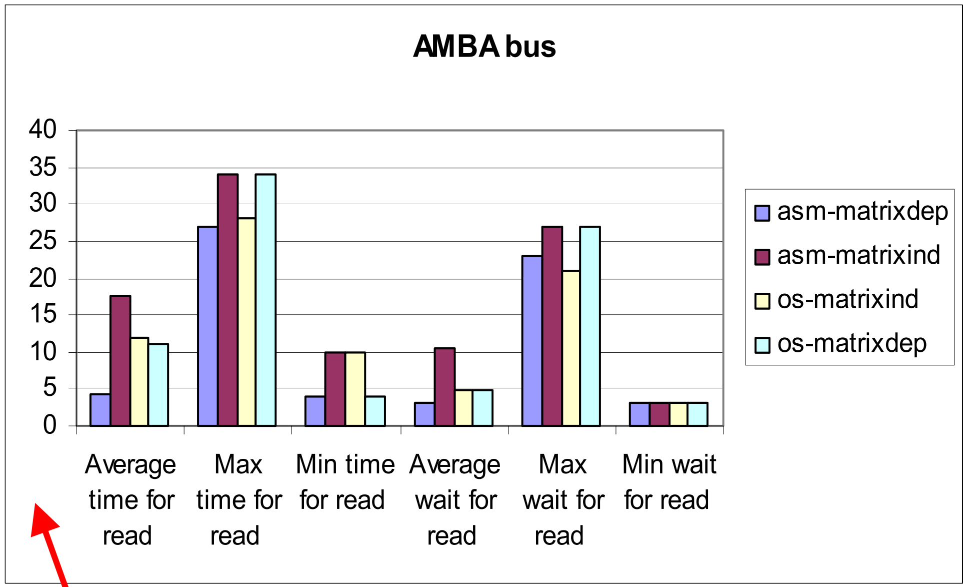
Communication support in OS increases significantly bus traffic (bus congestion)

Stbus achieves better bus efficiency

# Comparison: bus latency

**Latency for read accesses**

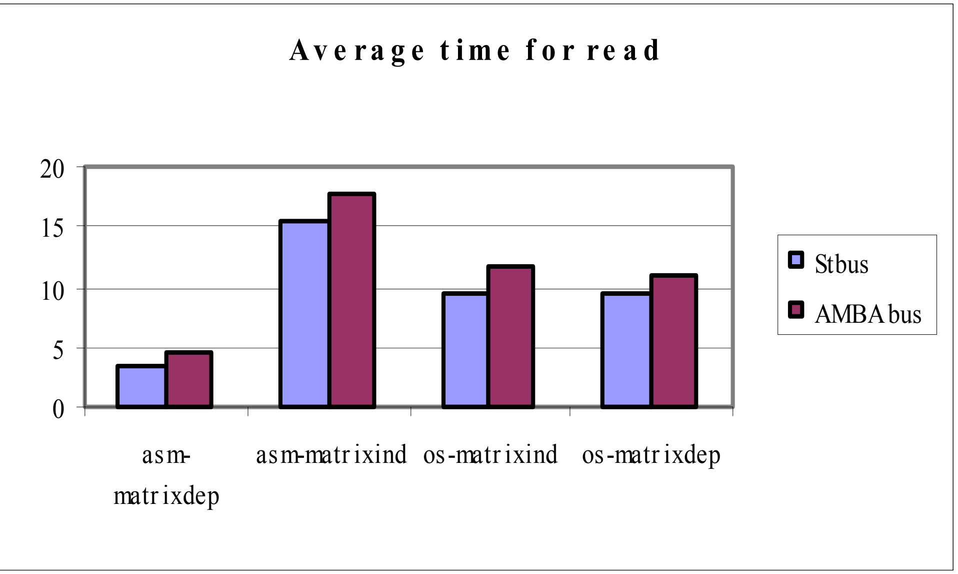


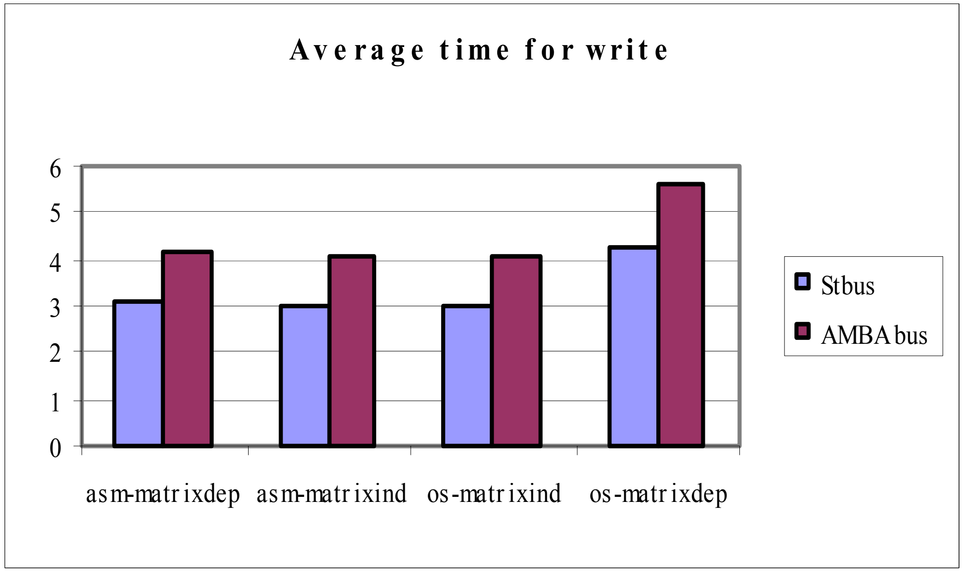


Max time and average time for read have different ratios with/without OS (due to different incidence of bursts)

Stbus has lower latency

# Comparison: performance of elementary transactions

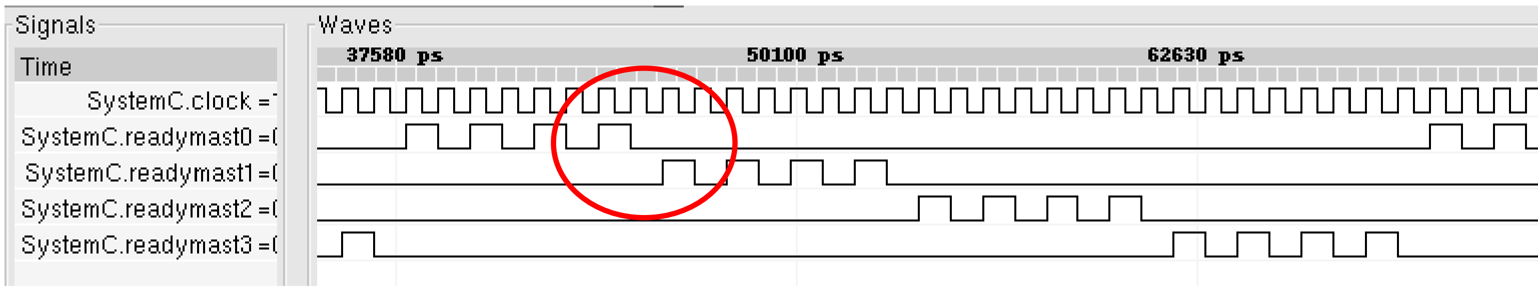




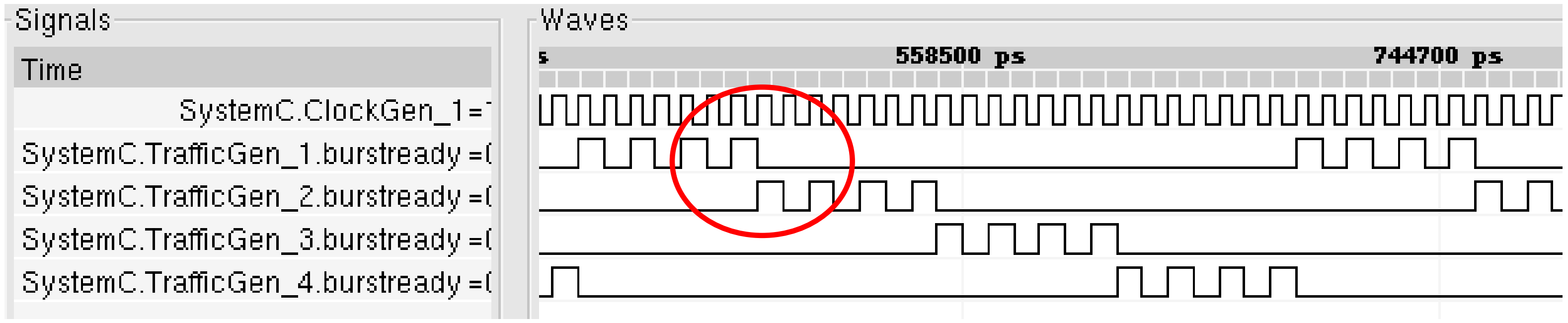
Stbus is consistently faster than AMBA on all benchmarks

# Analysis: Protocol differences

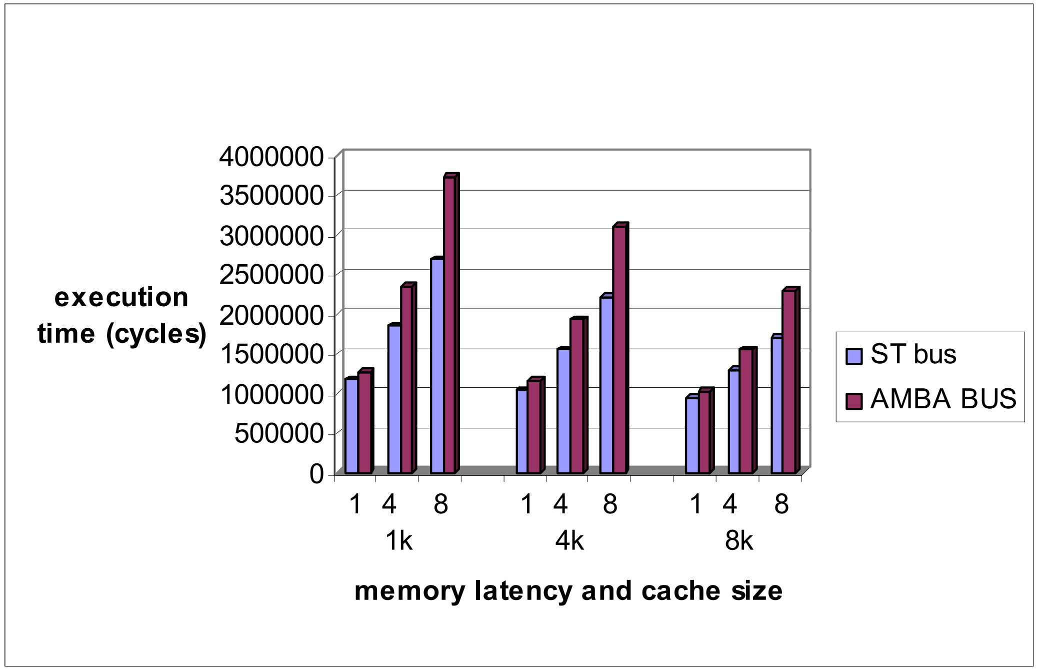
AMBA



STBUS

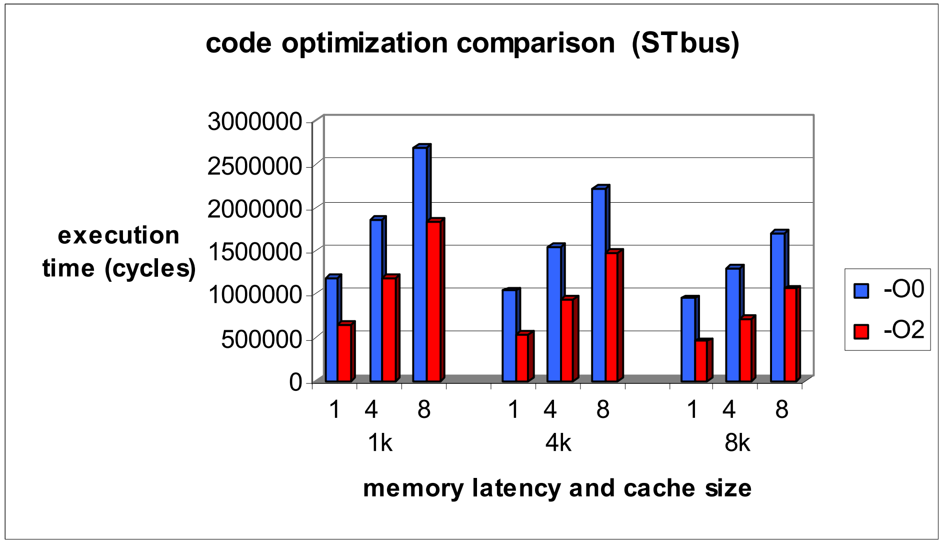

Signals
Waves
558500 ps
744700 ps
Time
SystemC.ClockGen_1 =
SystemC.TrafficGen_1.burstready =
SystemC.TrafficGen_2.burstready =
SystemC.TrafficGen_3.burstready =
SystemC.TrafficGen_4.burstready =

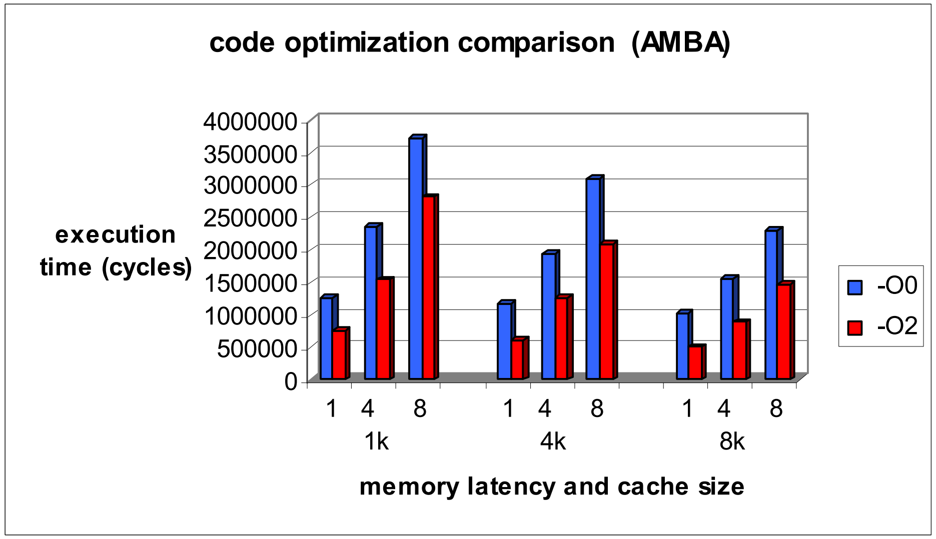
# Design space exploration: cache size & miss penalty



- STBUS cuts execution times by 9 to 35% with respect to AMBA
- A 4kB cache can bring 4 to 26% speed-ups wrt 1KB cache
- Increasing memory wait states from 1 to 4 slows down execution times from 35 to 104%

# Design space exploration: Code optimization strength





- In the STBus case, code optimization achieves 31 to 52% lower execution times

## Summary

- Paradigm shift towards MPSoC
- Designing MPSoCs
  - Huge design space
  - Communication is critical
  - Software-related effects are dominant
    - Hardware/software interface bottleneck
- MPARM provides a complete infrastructure for research in this area

## Outlook

- Simulation engine
  - TLM of communication links for speeding up simulation
  - Modeling GALS MPSoCs (first, multiple clocks)
- Hardware platform
  - OCP+Advanced interconnect (NoC): Xpipes
  - Advanced memory system (scratchpad)
  - Support for DPM (shutdown, DVS, complete power modeling)
  - New cores (PPC, StrongARM, LX, Reconfig)
- Software environment
  - Rethink communication support in OS
  - Integrate code optimizers (performance & power)
  - Bridge the gap to higher level of abstraction (e.g. support for dataflow programming)