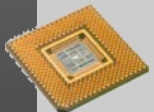
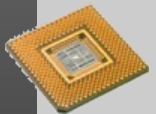


# Measuring Memory Access Latency for Software Objects in a NUMA System-on-Chip Architecture

ReCoSoC 2013  
Darmstadt, Germany  
Daniela Genius



- 1- Introduction
- 2- Related Work
- 3- The Memory Spy Module
- 4 - Example Application
- 5 - Experiments
- 6 - Conclusion and Future Work



# 1 – Introduction

Complex streaming applications

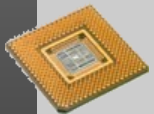
- Parallelism (pipeline, task farm)
- Data transferred between tasks via channels or shared memory
- Software objects stored in on-chip memory

Performance : Network on chip

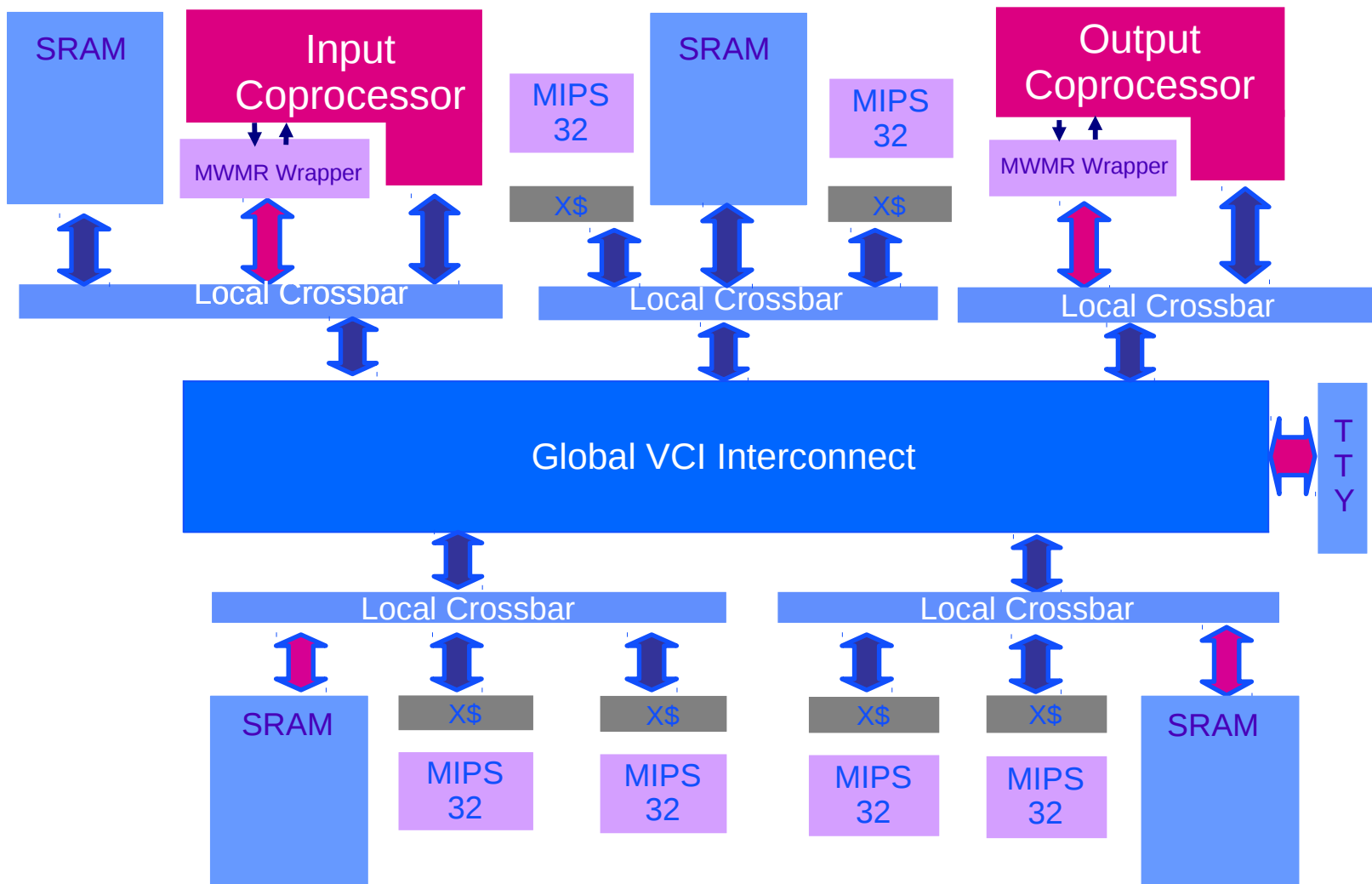
NUMA architectures : varying latencies for memory accesses

SoCLib/DSX (*Design Space Explorer*)

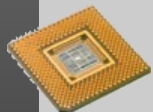
- Components written in SystemC
- *Mutek* micro kernel
- VCI/OCB shared address space paradigm
- Task and Communications Graph (TCG)
- Explicit and fine-grained mapping of software objects



# 1 -Introduction



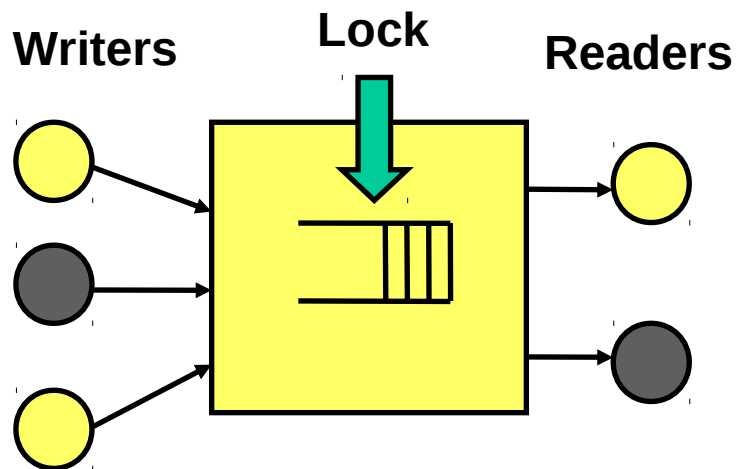
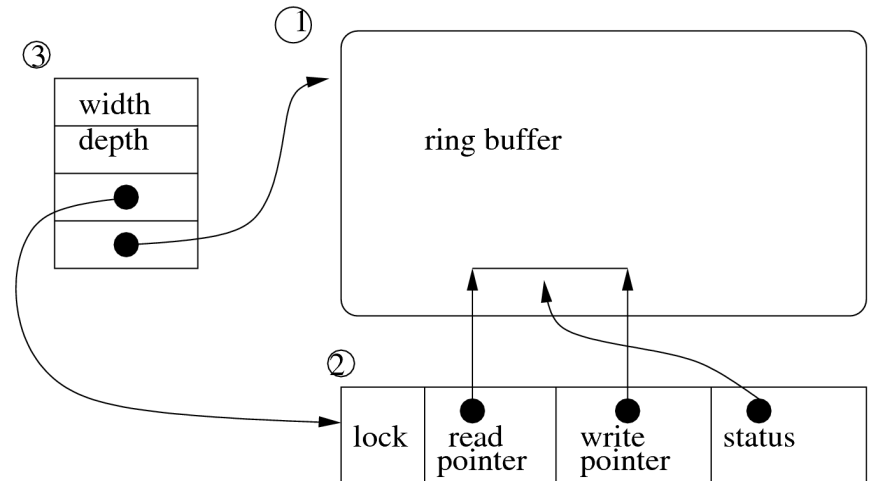
NUMA Architecture built from SoCLib components



# 1 – Introduction

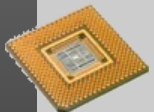
Multi Writer Multi Reader channels  
[Faure/Greiner 2006]

Shared memory  
memspaces in on-chip  
memory



get lock (READ)  
test MWMR status (READ)  
transfer burst (READ/WRITE)  
update status (WRITE)  
release lock (WRITE)

→ Increased traffic !



## 2 – Related Work

Analysis of memory accesses

- Software-based

*valgrind* [Nethercote/Seward 2007]

In SoCLib-based simulation: distortion of results due to code augmentation

- Hardware-based

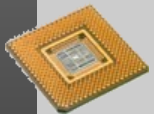
*MemTracker* [Venkataramani et al. 2007]

*Dynamic Instruction Stream Editing* [Corliss et al. 2003]

*Iwatcher* [Zhou et al. 2005]

*IBM Hardware Assisted Watchchpoints* [Allen et al. 2003]

Difficulty: determine which processor accesses which memory bank for software object at a given time!

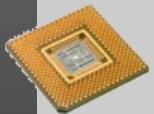


## 3 – The Memory Spy Module

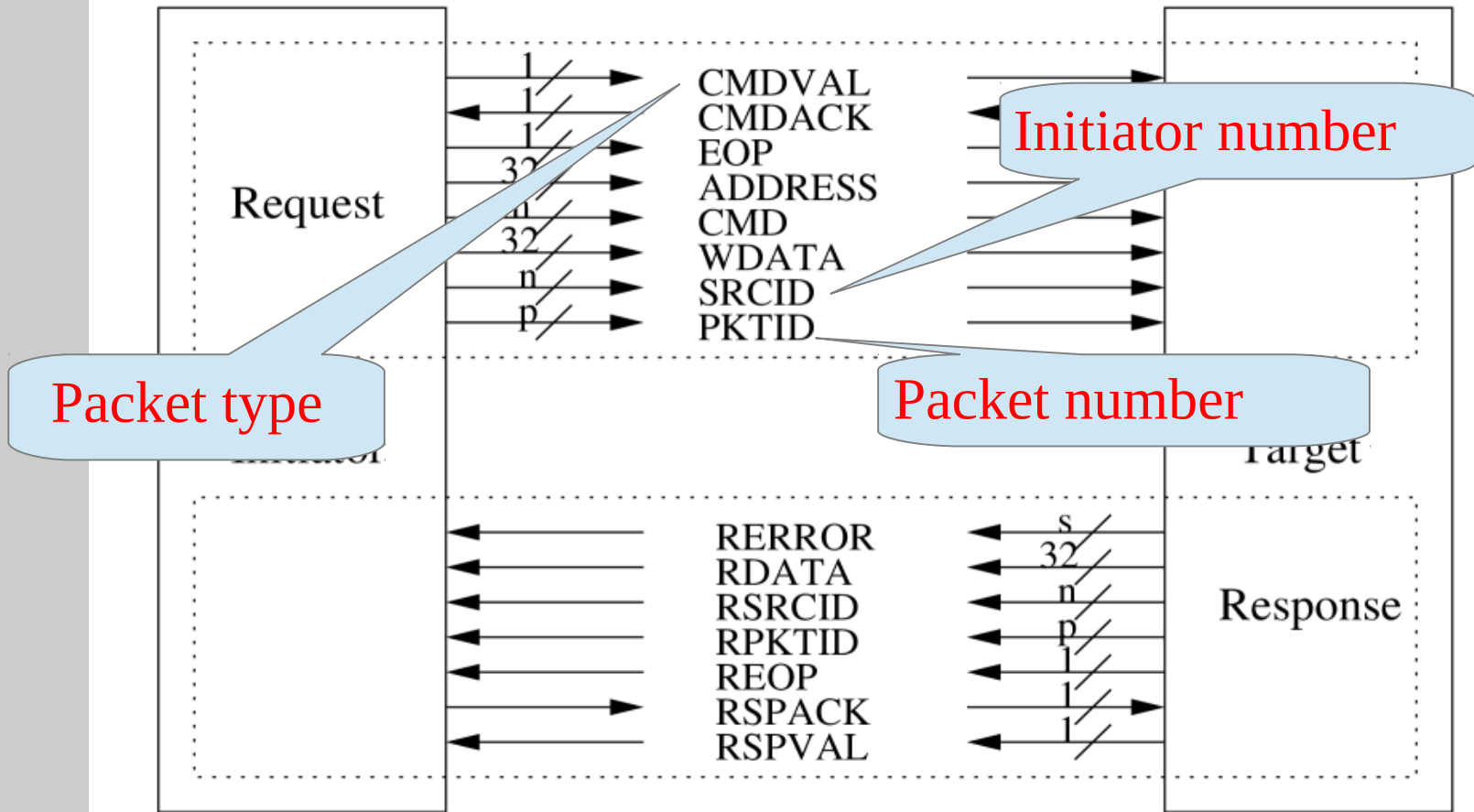
Measure latency of access to a specific software object in memory (e.g. buffer of MWMR channel, lock)

Information required and how to obtain it :

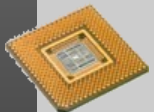
- *Packet type*: intercept VCI transfer
- *Initiator number*: intercept VCI transfer
- *Packet number*: intercept VCI transfer; problem of out-of-order delivery
- *Software object by name/address*: modify the loader and add loop over objects to the SystemC code of the VCI logger module
- *Simulation cycle*: add cycle counter to the SystemC code of the VCI logger module



# 3 – The Memory Spy Module



## Analyzing VCI Transfers





Modifications to the Loader

`Get_symbol_by_name()`

Identify relevant software objects

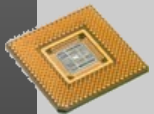
`channel8_status` - Address: `0x180000a0`

`channel9_status` - Address: `0x180000b0`

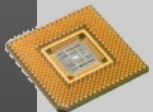
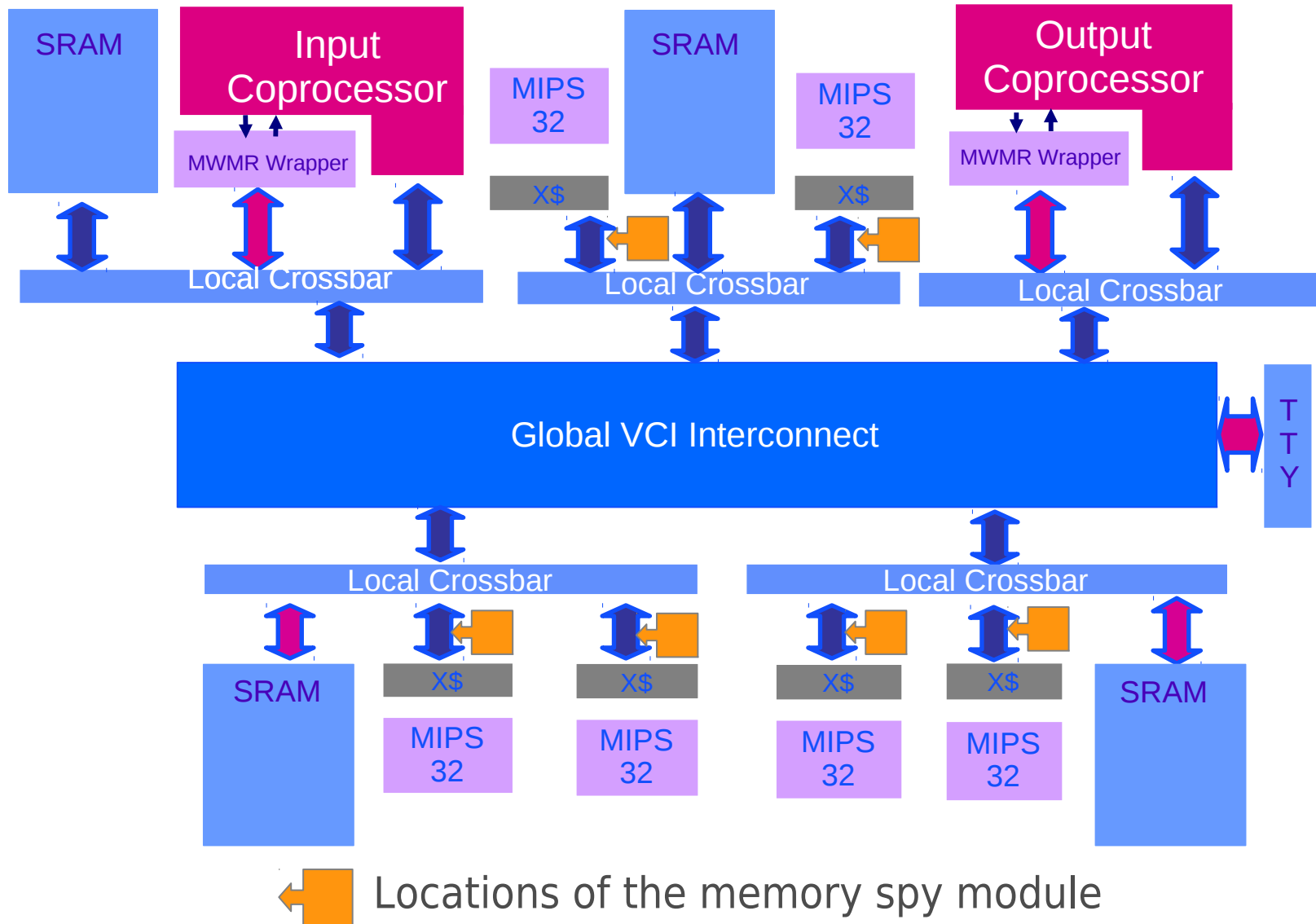
`mem1_buf` - Address: `0x18001138`

`mem2_buf` - Address: `0x18001538`

`barrier0_desc` - Address: `0x1804517c`

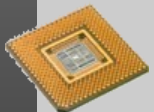


# 3 – The Memory Spy Module



```
at every cycle
  if address corresponds to address to spy then
    if CMD then
      save simulation cycle, initiator no., packet no.
    end if
  if RSP then
    if last packet then
      save simulation cycle
      look up corresponding command and cycle
      calculate difference
      store name, cycle object as sent, latency
    end if
  end if
end at every cycle
```

Algorithm to extract latency information from log file



## 4 – Example Application

Packet classification

Task farm type application

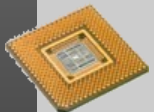
- Slots in shared memory (memspaces)
- Descriptors transmitted via MWMM channels

Addresses generated at bootstrap and sent to Input Coprocessor

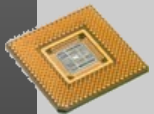
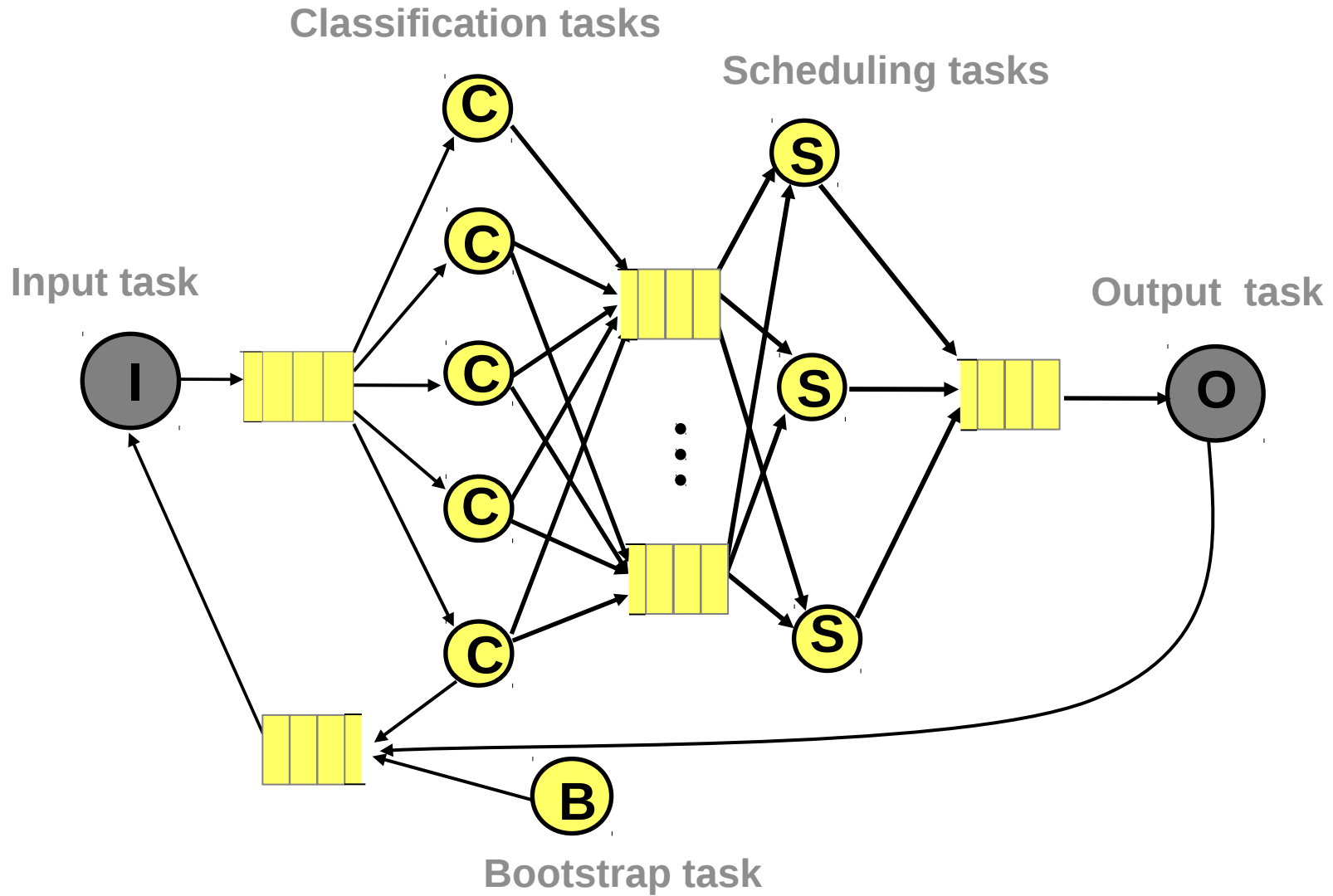
Feedback of addresses from Output Coprocessor

Five types of tasks :

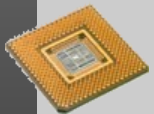
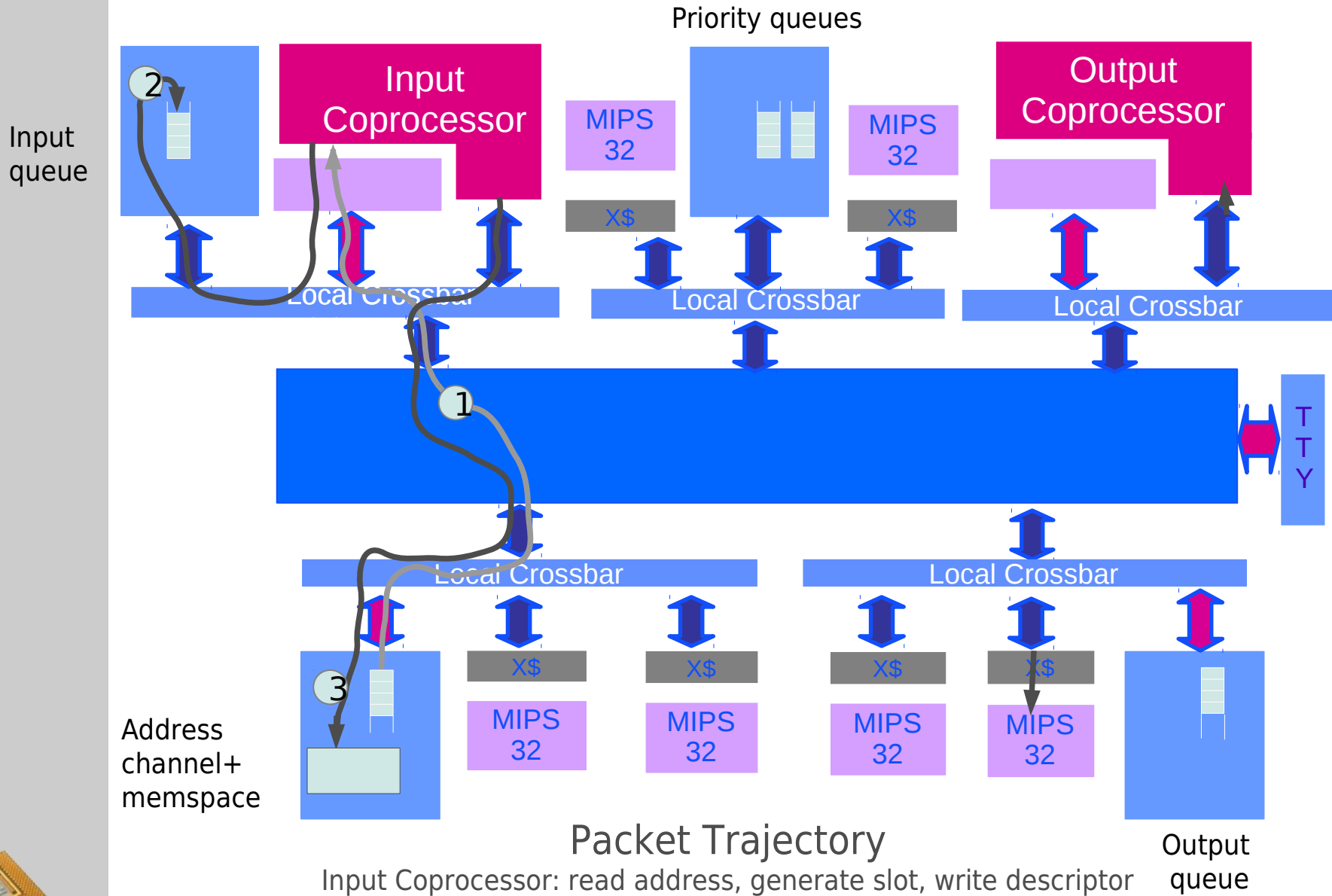
- Input Task: cuts packets into slots and generates descriptor
- Classification Task: reads descriptor, retrieves packet, analyzes header
- Scheduling Task: weighted round robin selection
- Output Task: reconstitutes packets
- Bootstrap Task: fills up internal and external address FIFOs with generated addresses in the beginning, starts coprocessors, then suspends



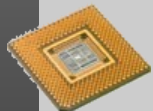
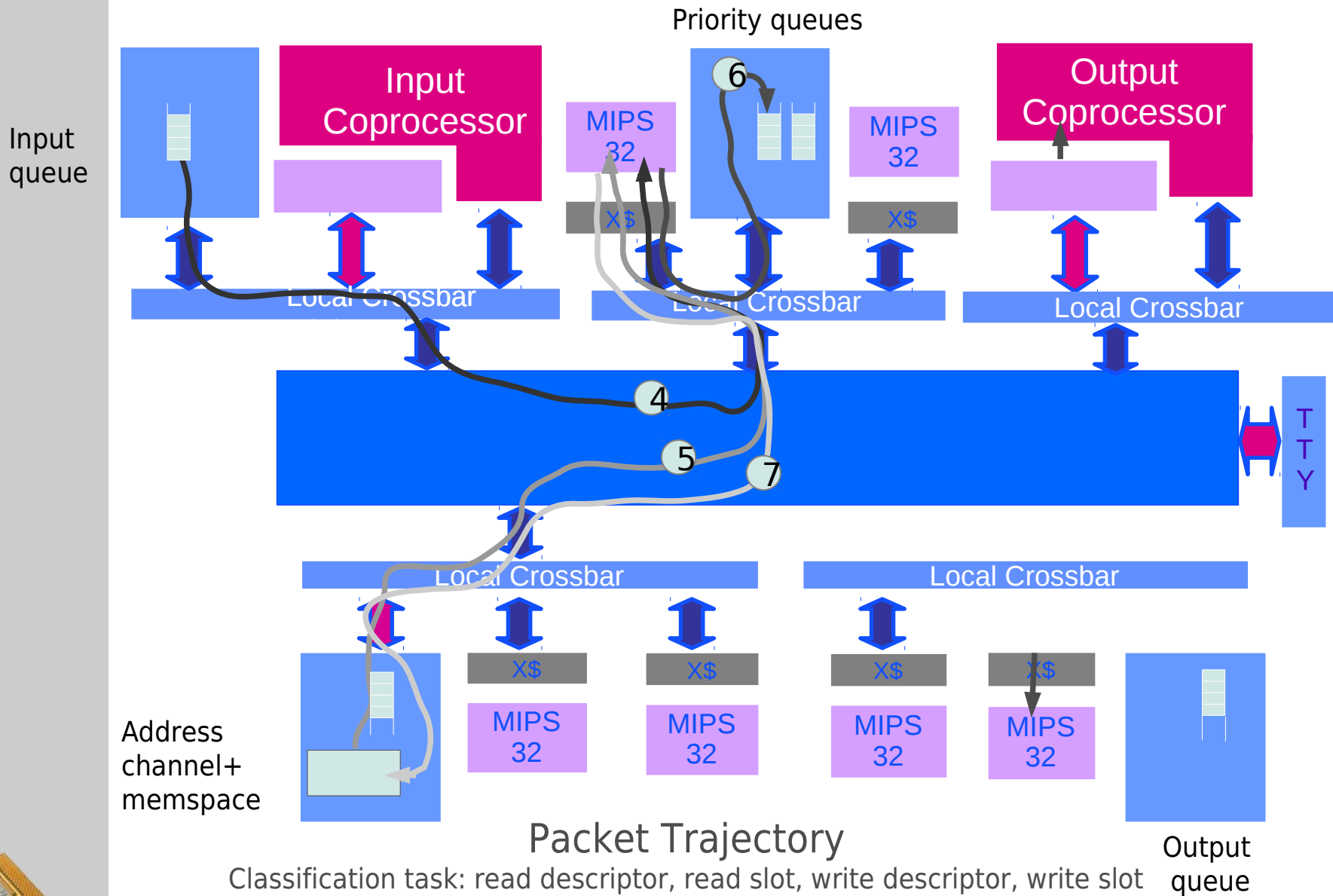
# 4 – Example Application



# 4 – Example Application

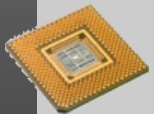
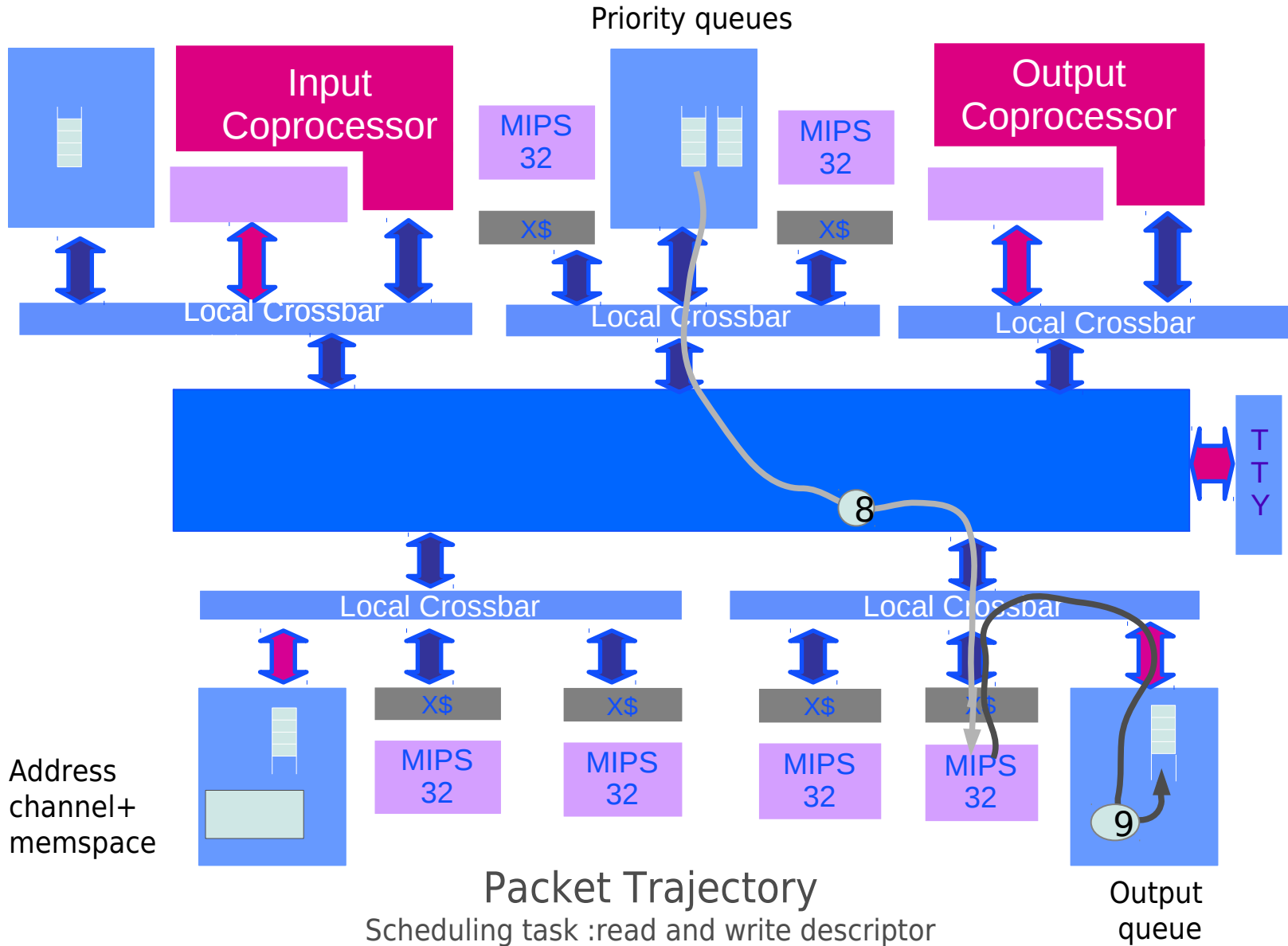


# 4 – Example Application



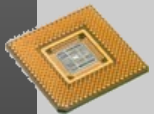
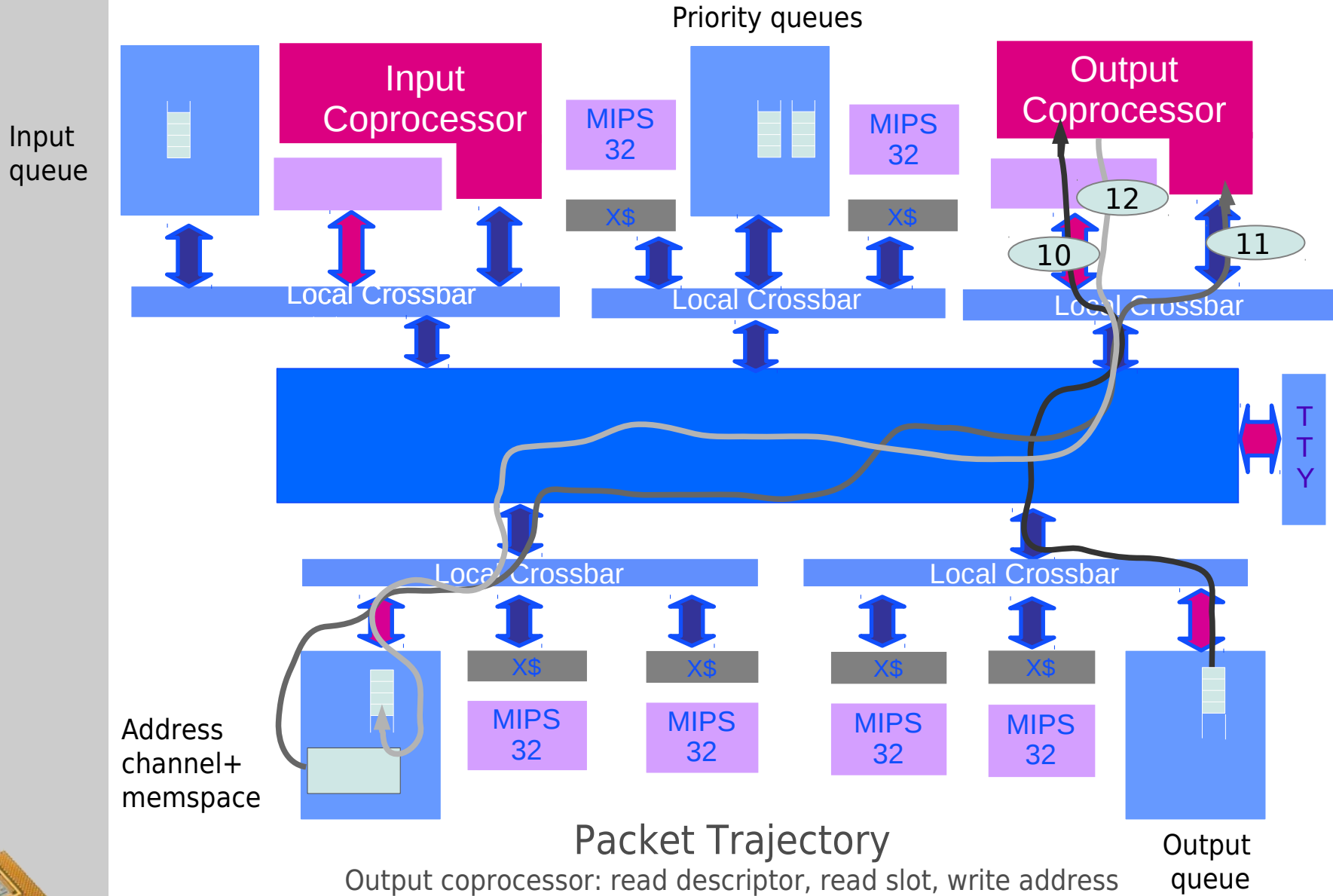
# 4 – Example Application

Input queue

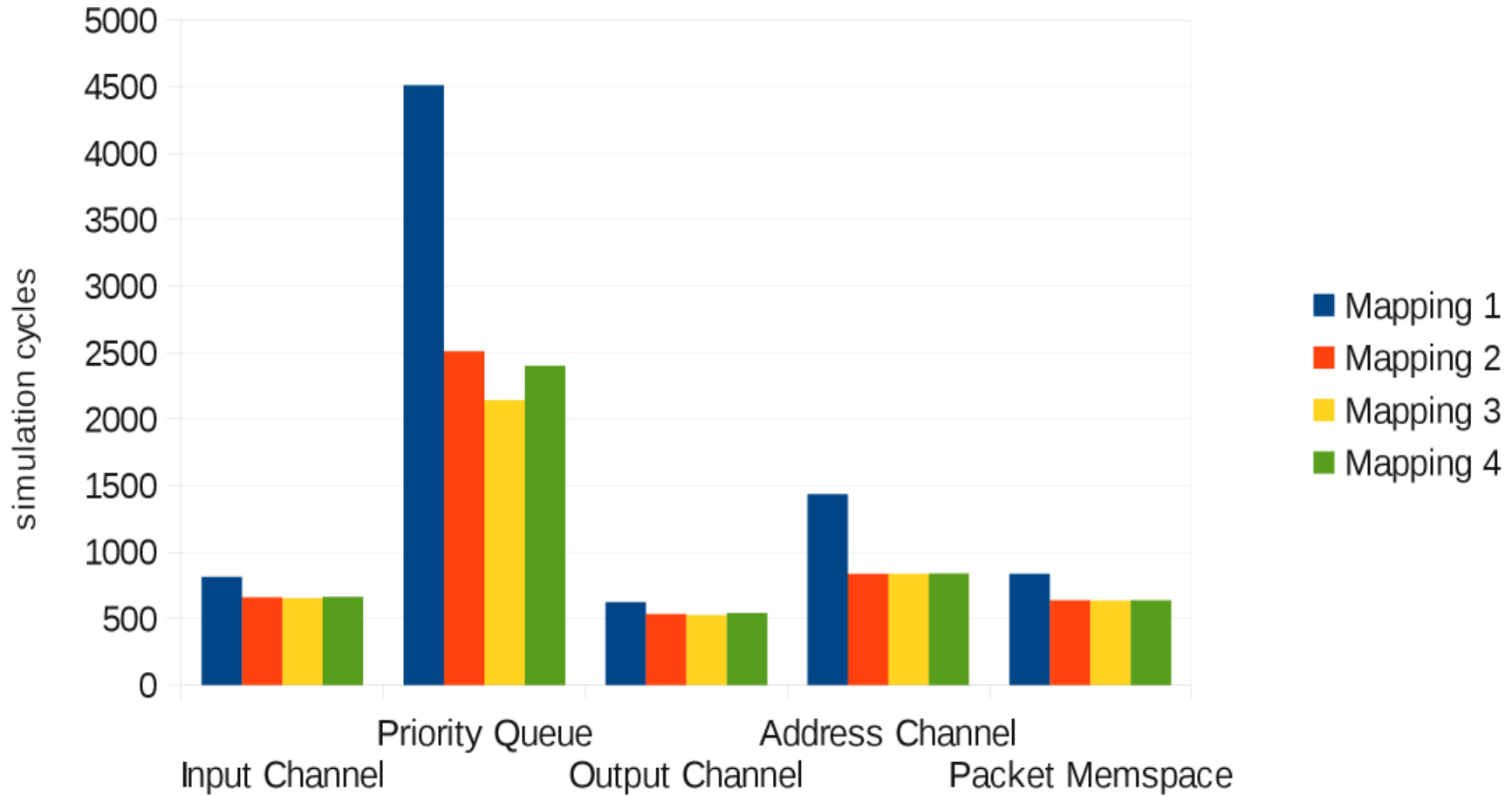




# 4 – Example Application



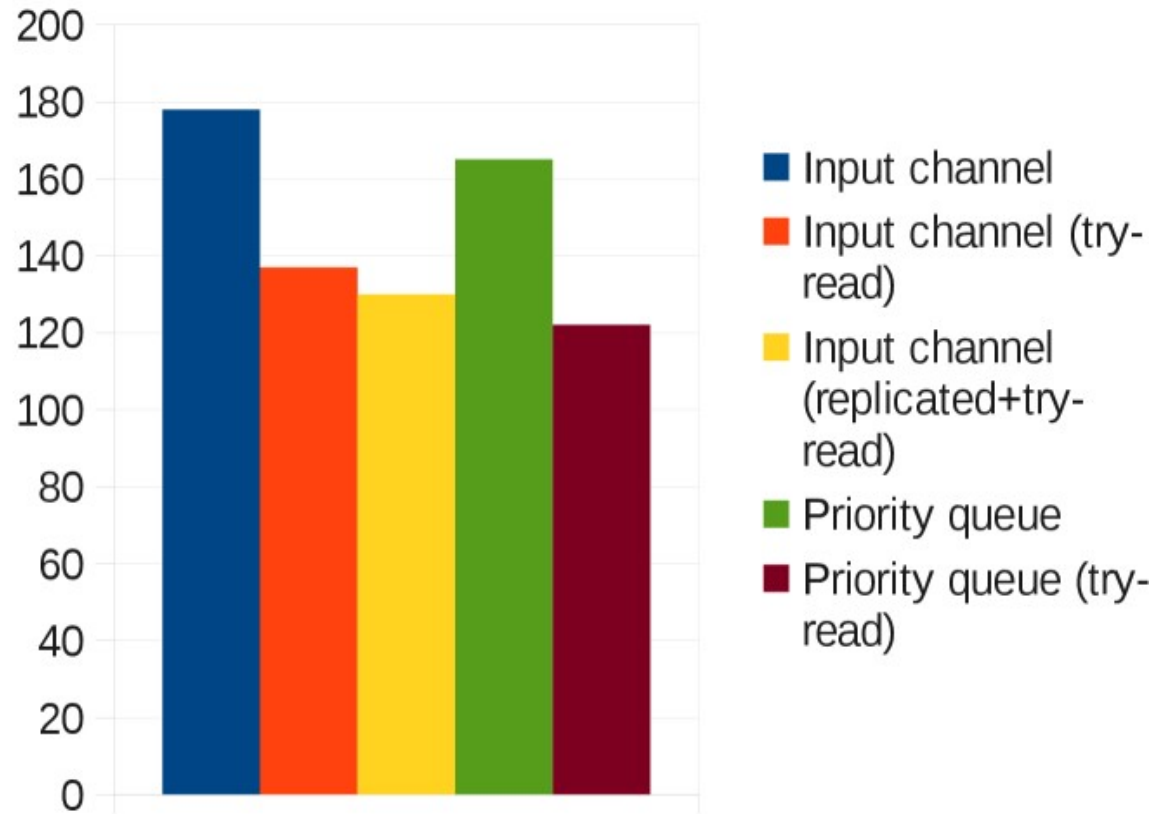
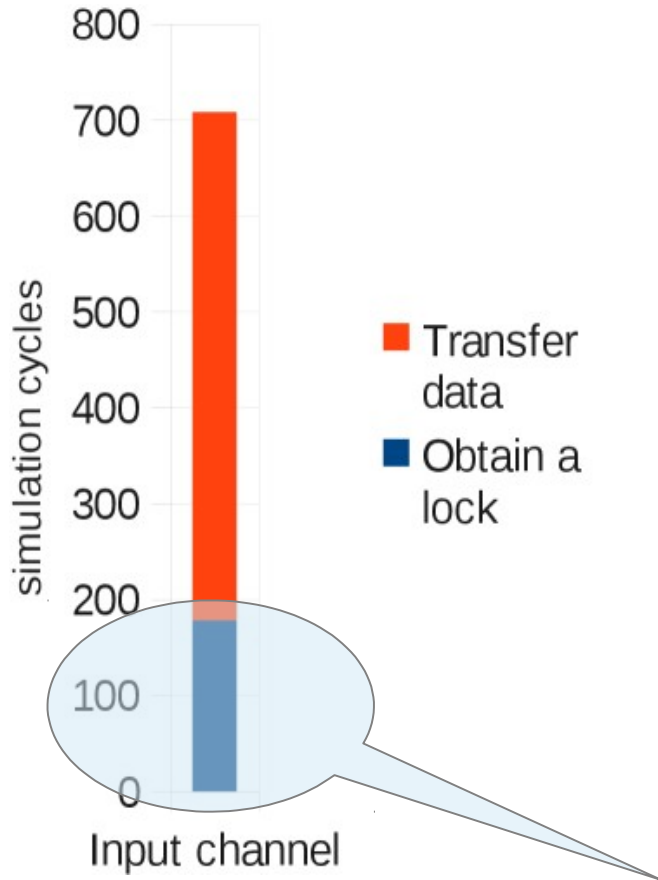
# 5 – Experimental Results



- All SW objects on input cluster
- Channels round robin, all others on input cluster
- Mapping of [Faure'07] : priority queues and output queue on scheduling cluster
- Round Robin+output queue on scheduling cluster



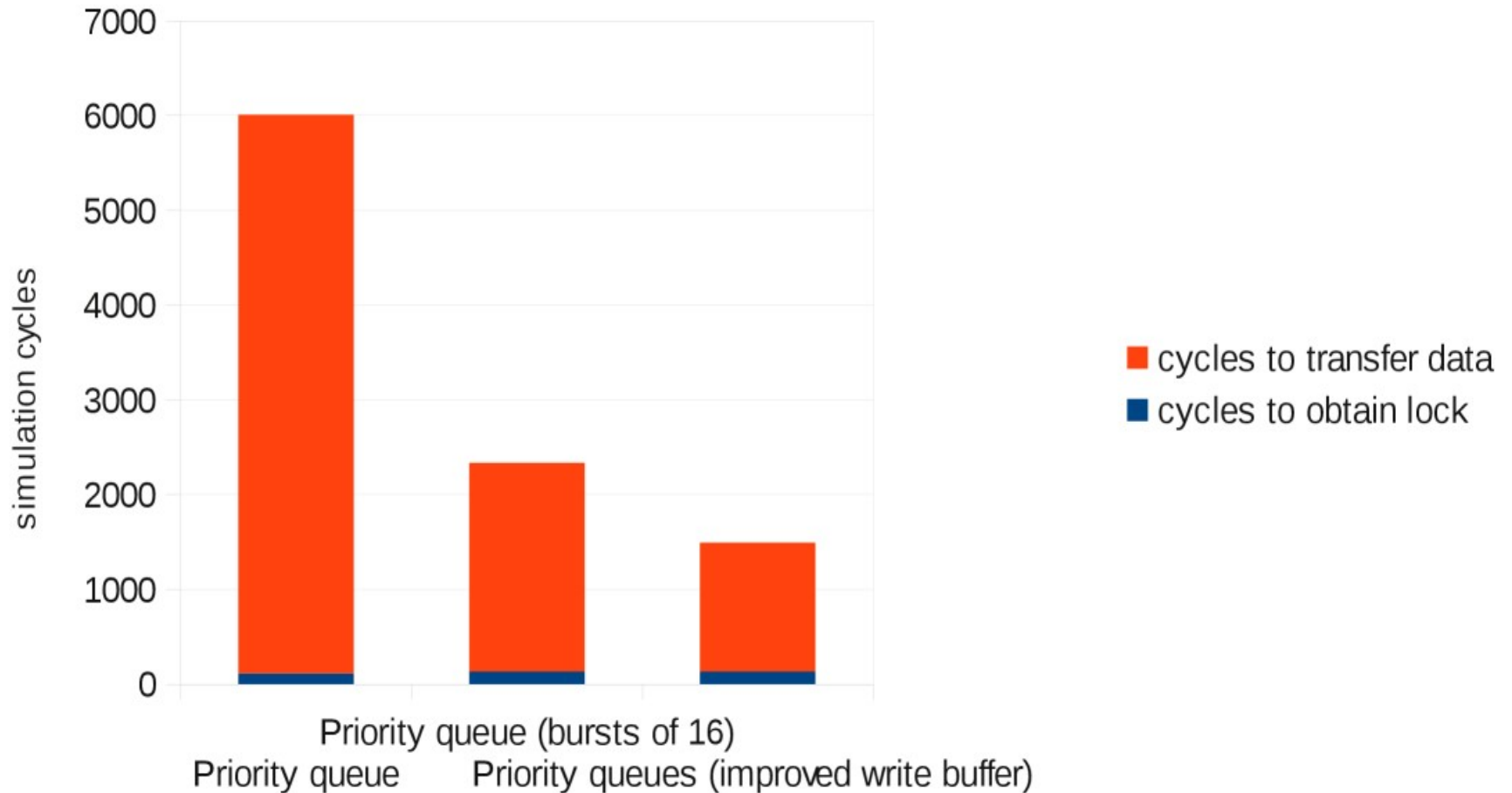
# 5 – Experimental Results



Reducing Latency to obtain a lock



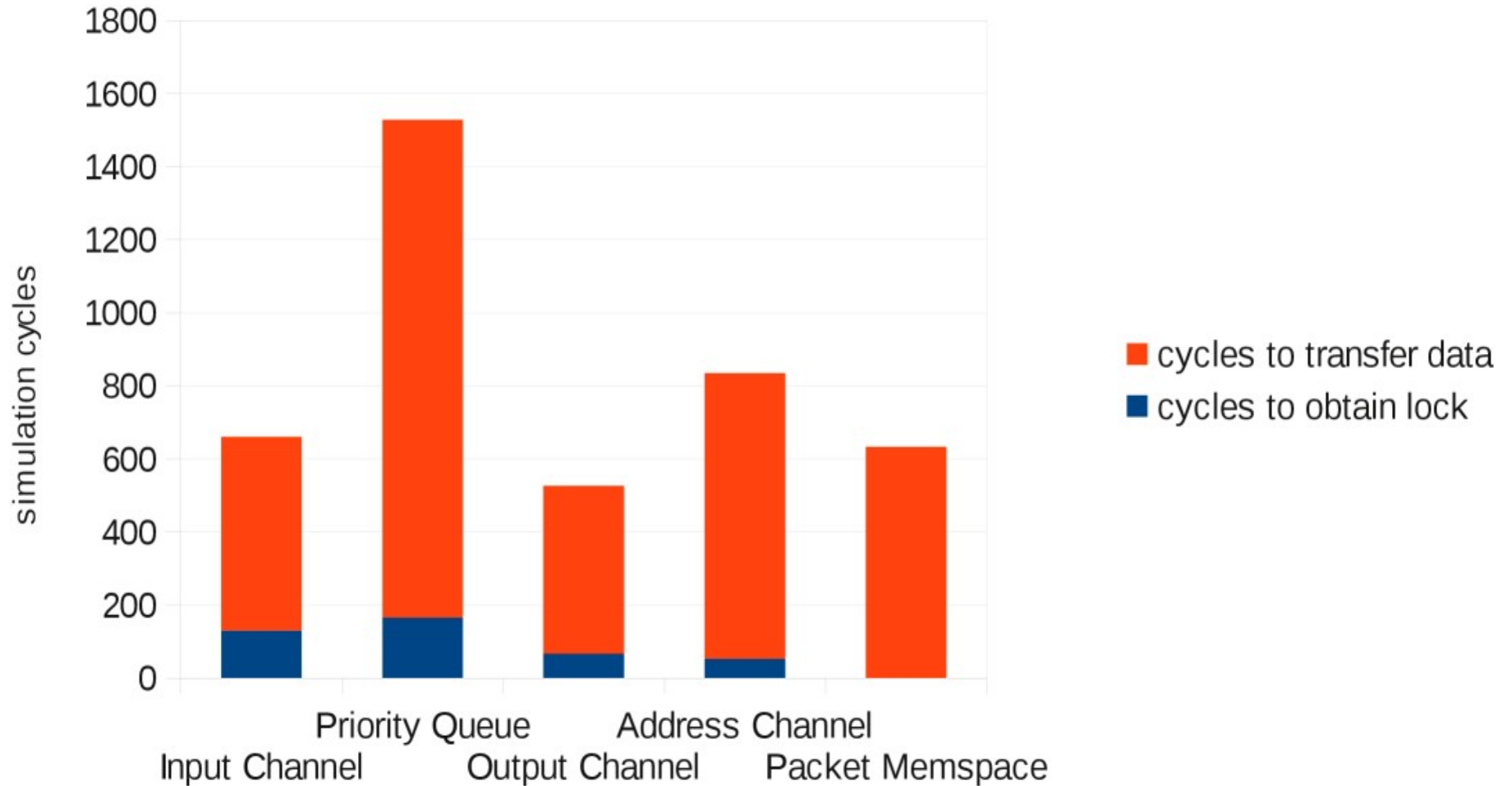
## 5 – Experimental Results



Obstacles : bursts, write buffer



# 5 – Experimental Results



Results after optimization

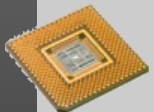


## Summary

- Irregular latencies : more detailed information to remap
- Latency to obtain a lock : break up latency into obtaining the lock and transfer itself (cf channel spy)
- Analysis of various problems with priority queues

18% simulation overhead

Traces 50 % bigger than those used in [Recosoc2011] as more information is required



Acceptable slowdown of the simulation

Detailed analysis helps to improve mapping

Benchmark of streaming applications with increasing complexity of memory accesses (MJPEG, H.264, packet classification)

Design Space Exploration

Combination with hardware counters (cache, ...)

Graphical representation (access conflicts, heavy usage of certain routes in the NoC, ...)

Extend application model by communication via shared memory?

