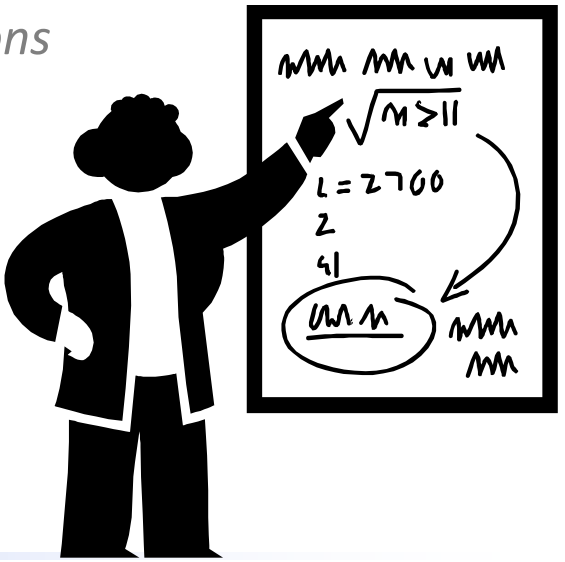
Flot de conception d'applications parallèles sur plateforme reconfigurable dynamiquement

Clément Foucher, Fabrice Muller et Alain Giulieri

Université de Nice-Sophia Antipolis (UNS), (LEAT/ CNRS)
{Clement.Foucher ; Fabrice.Muller ; Alain.Giulieri}@unice.fr

Plan

- Parallel and reconfigurable computing today
  - Context
  - *Systems limitations*
- *Our proposal: the SPORE system*
- *Implementing SPORE*
  - *Application design*
  - *The hardware platform and its implementations*
- *Conclusion & future works*

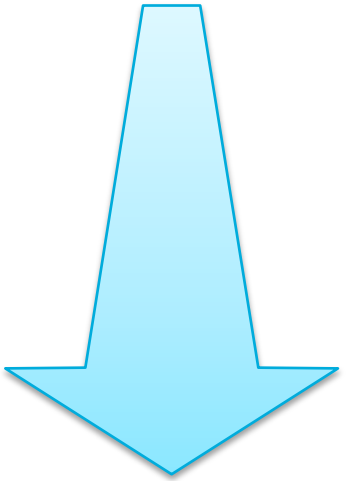


Parallel systems

- Personal computers
  - Low amount of software cores

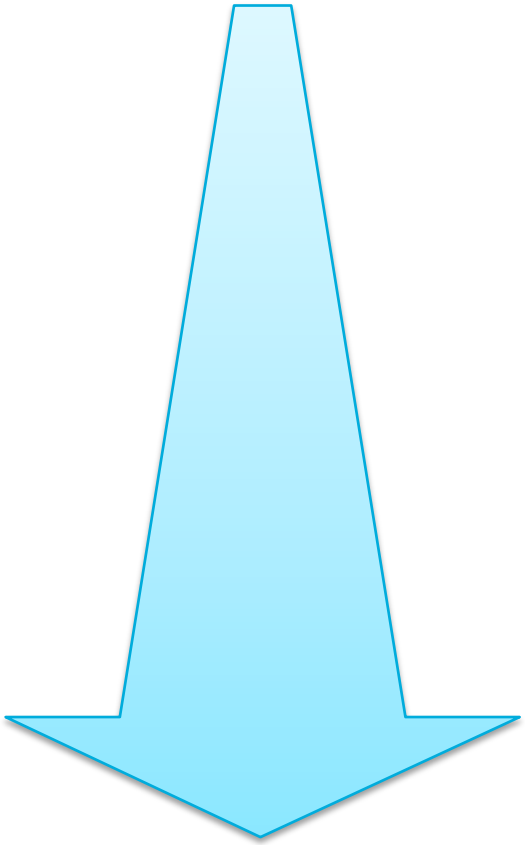
Parallel systems

- Personal computers
  - Low amount of software cores
- Manycore systems
  - Still rising



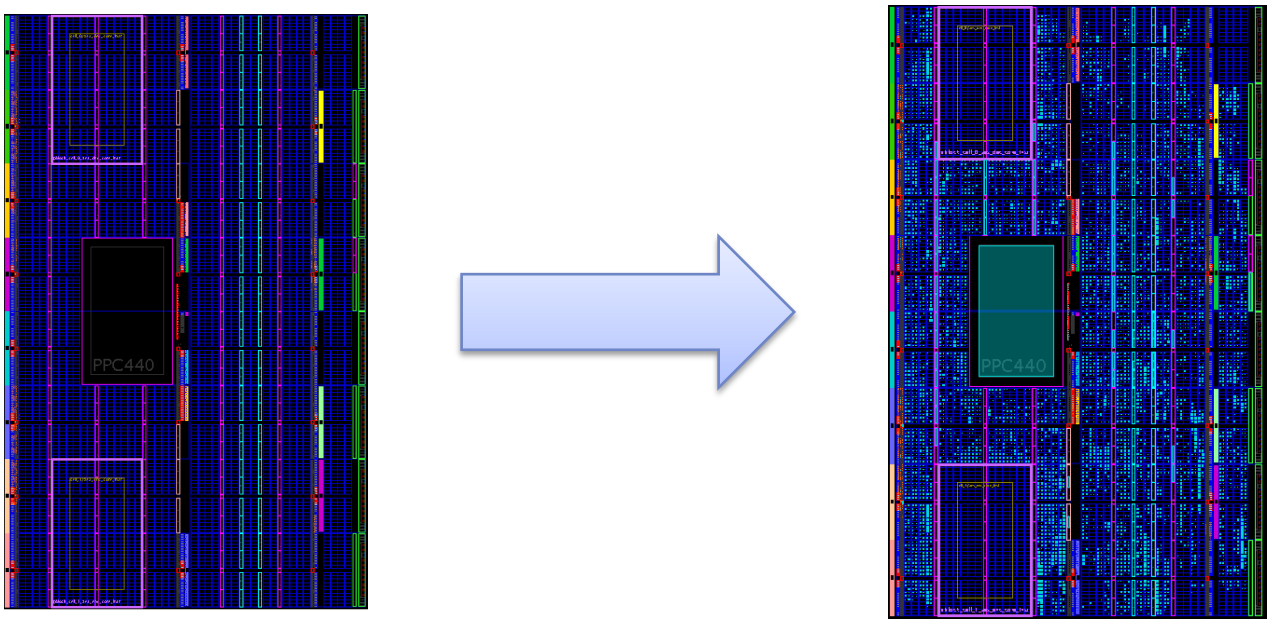
Parallel systems

- Personal computers
  - Low amount of software cores
- Manycore systems
  - Still rising
- High performance computers
  - Massively parallel software systems



Reconfigurable systems

- Generic hardware
  - Arrays of reconfigurable elements linked by a configurable network
    - Configurable into particular systems

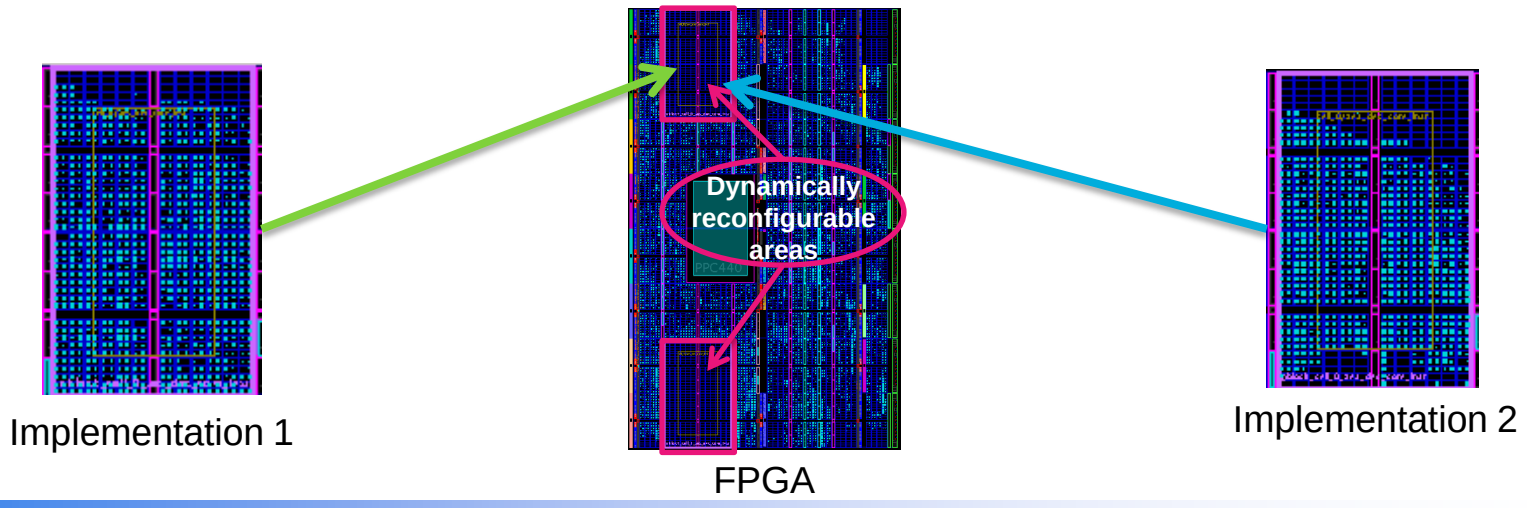


Blank FPGA

Routed FPGA

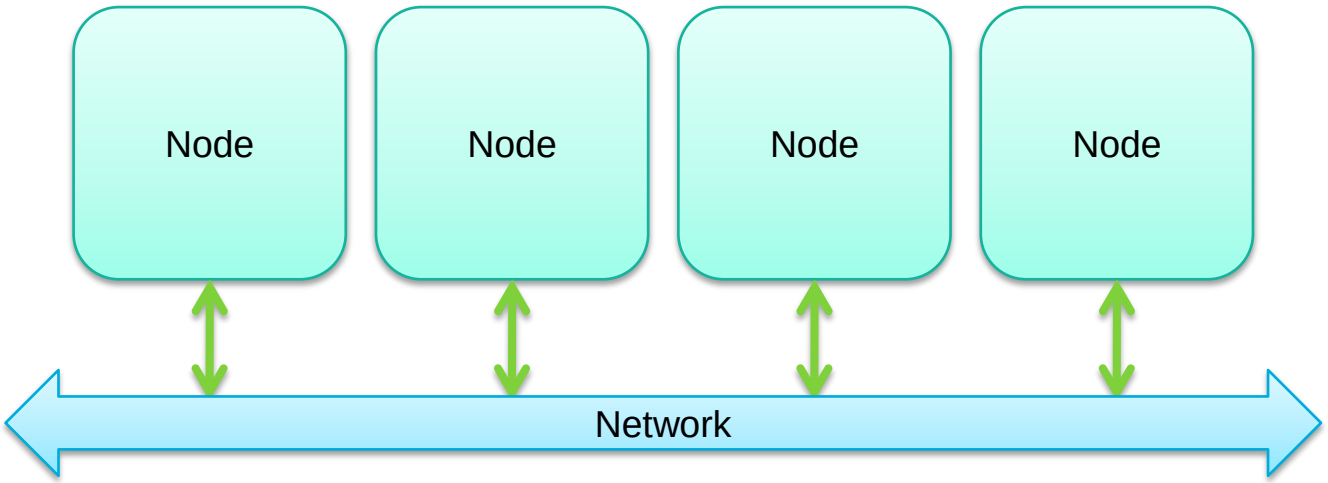
Reconfigurable systems

- Generic hardware
  - Arrays of reconfigurable elements linked by a configurable network
    - Configurable into particular systems
- Evolution: partial dynamic reconfiguration
  - Change only a part of the device on the fly



Reconfigurable parallel systems

- High Performance Reconfigurable Computers (HPRC)
  - Introduce generic hardware
    - Accelerate specific portions of code ("application kernels") by devolving computations to hardware accelerators

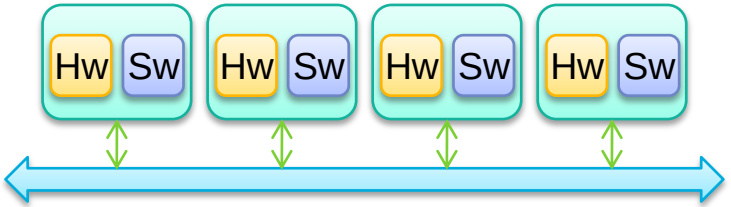


Standard HPC structure

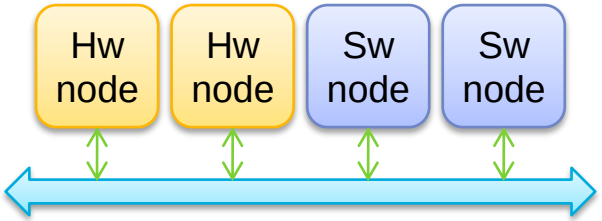
Reconfigurable parallel systems

- High Performance Reconfigurable Computers (HPRC)
  - Introduce generic hardware
    - Accelerate specific portions of code ("application kernels") by devolving computations to hardware accelerators

- Two kinds [1]
  - Nonuniform Node, Uniform System (NNUS)

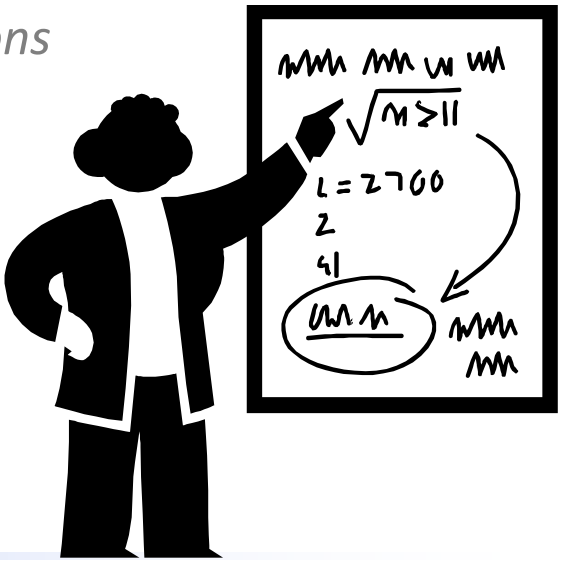


  - Uniform Node, Nonuniform System (UNNS)



Plan

- Parallel and reconfigurable computing today
  - *Context*
  - Systems limitations
- *Our proposal: the SPORE system*
- *Implementing SPORE*
  - *Application design*
  - *The hardware platform and its implementations*
- *Conclusion & future works*



Software nature of applications

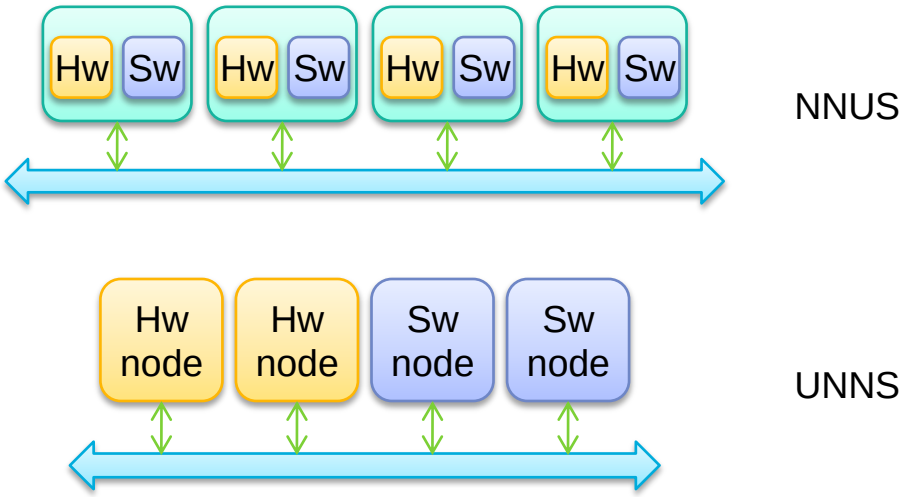
- Historically, systems were software only
  - By the time, hardware resources are added
    - Hardware is static while software is flexible

Applications are still mainly software, even if some particular computations are hardware accelerated

- Reconfigurable systems
  - Add more flexibility to hardware elements
    - But applications conception did not change: software-based, with some computations devolved to hardware

Applications linked to execution platform

- HPRCs
  - Ability to use hardware resources depends on the ratio hardware vs. software resources
    - UNNSs are more flexible than NNUSs
  - Communication performances between software and hardware depends on underlying buses
    - Platform change can lead to performances collapse

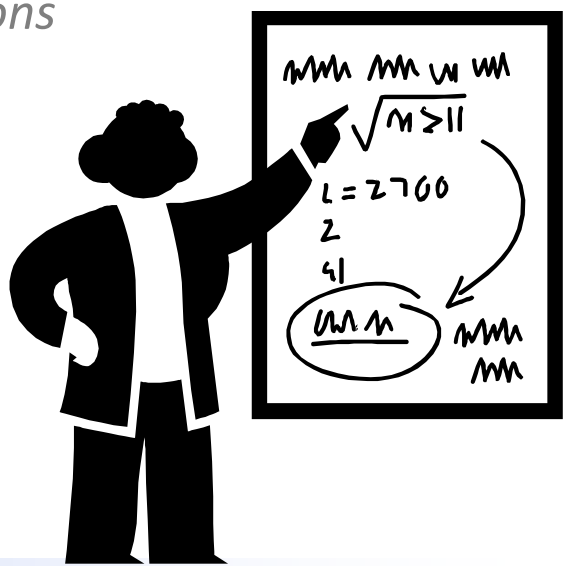


Applications linked to execution platform

- HPRCs
  - Ability to use hardware resources depends on the ratio hardware vs. software resources
    - UNNSs are more flexible than NNUSs
  - Communication performances between software and hardware depends on underlying buses
    - Platform change can lead to performances collapse
- Applications are thus deeply linked to the underlying hardware
  - Changing the execution platform of legacy application can force partial application re-write
    - To maintain performances
    - Or even to make the application compatible

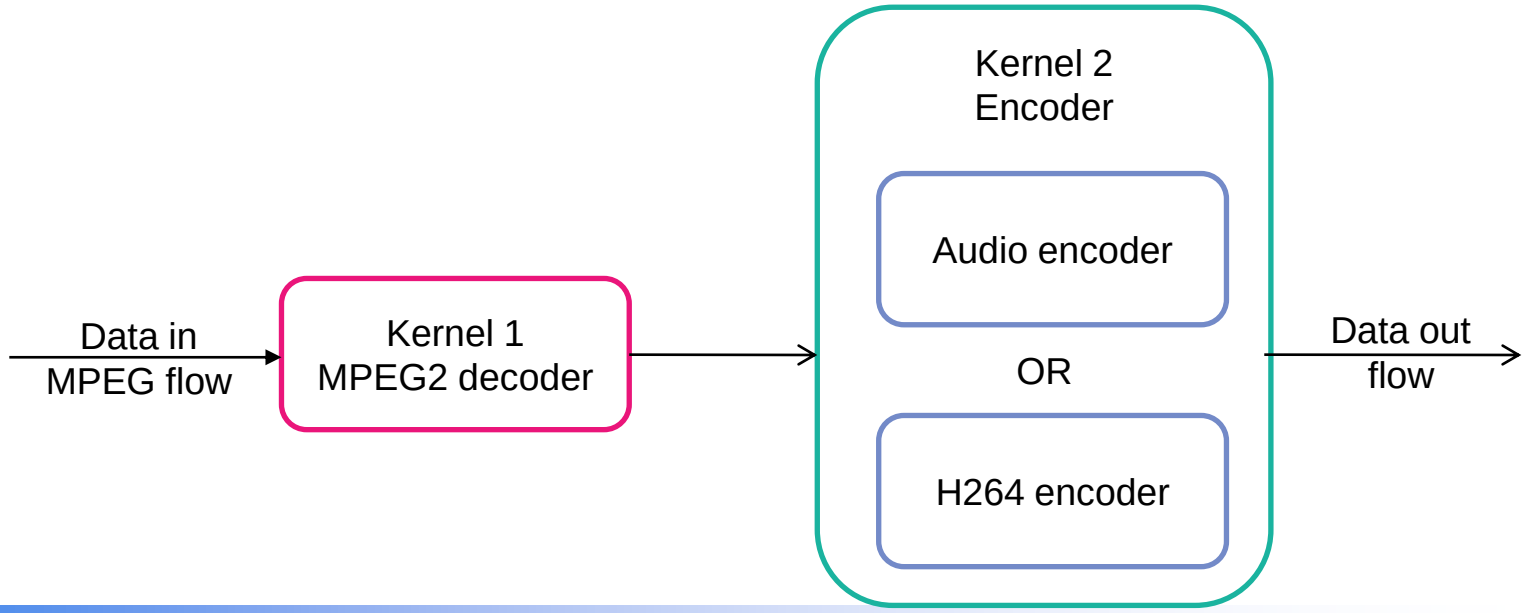
Plan

- *Parallel and reconfigurable computing today*
  - *Context*
  - *Systems limitations*
- Our proposal: the SPORE system
- *Implementing SPORE*
  - *Application design*
  - *The hardware platform and its implementations*
- *Conclusion & future works*



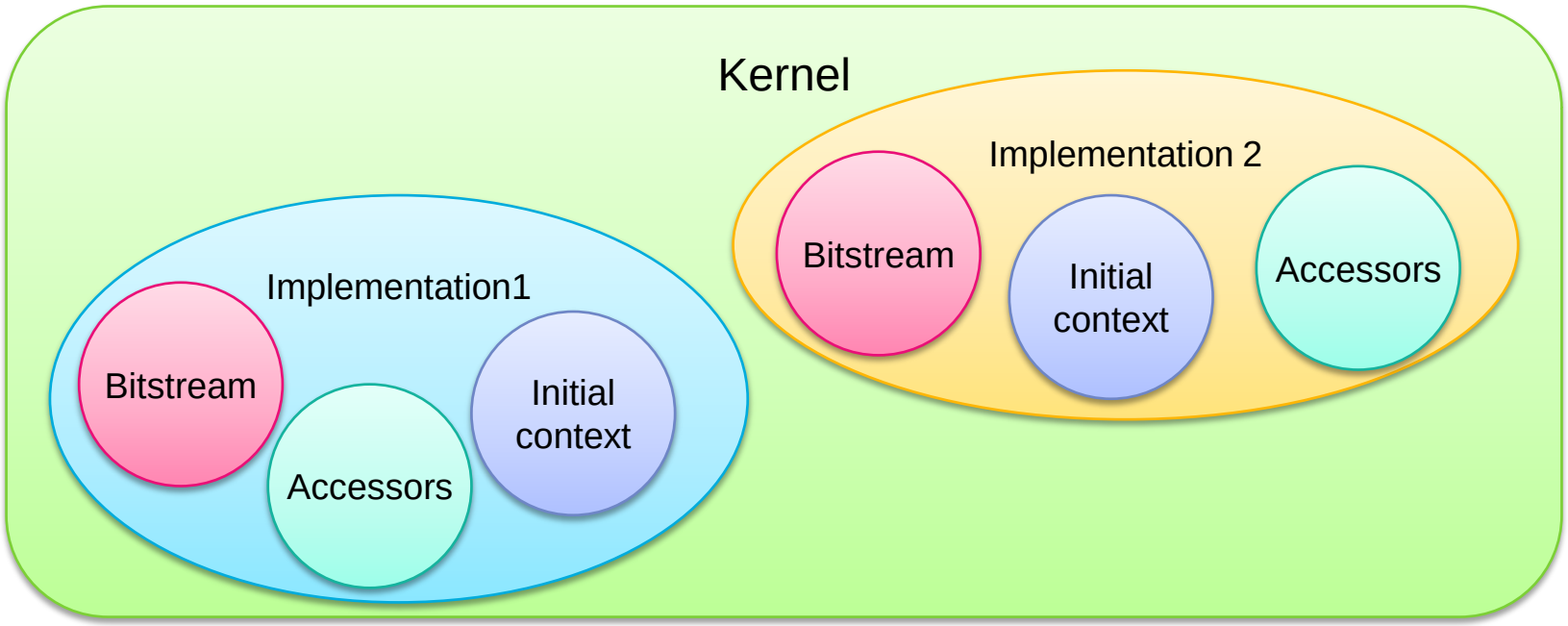
Our proposal – Applications

- Build applications as sets of *kernels*
- Kernels linked by data flows
- A kernel is *what* to do without knowing *how* to do



Our proposal – Kernels

- Kernel implementation is handled independently from the application
  - Each kernel can have various implementations, hardware ones and/or software ones



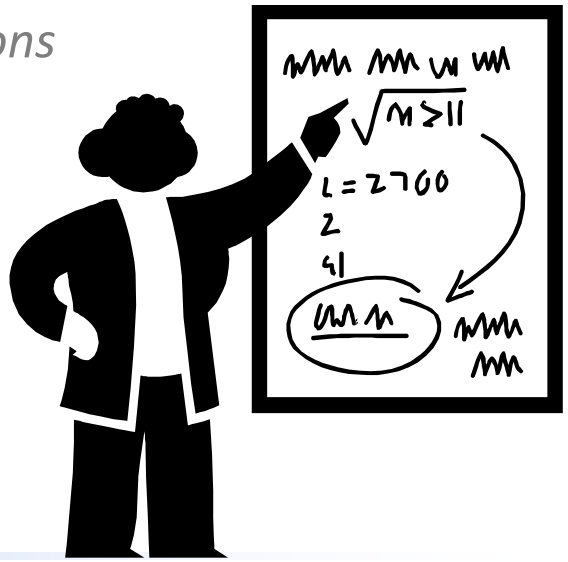
Our proposal – Execution platform

- The platform
  - Various nodes connected through a network
- The nodes
  - A host cell, in charge of inter-node communication
  - Various computing cells
- The computing cells
  - Reconfigurable

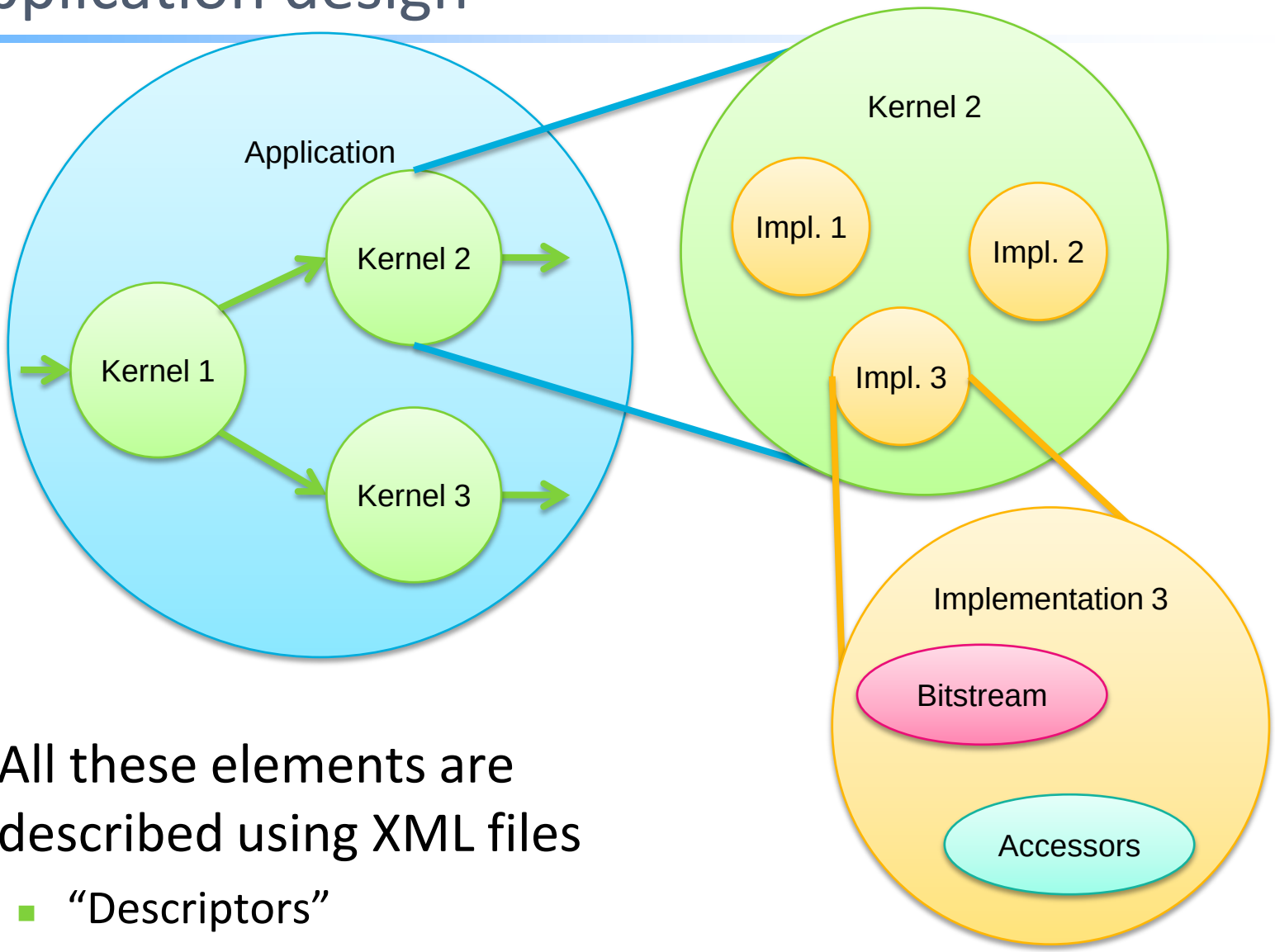
This is the
Simple Parallel platfOrm for Reconfigurable Environment
(SPORE)

Plan

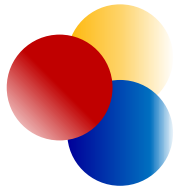
- *Parallel and reconfigurable computing today*
  - *Context*
  - *Systems limitations*
- *Our proposal: the SPORE system*
- Implementing SPORE
  - Application design
  - *The hardware platform and its implementations*
- *Conclusion & future works*



Application design



- All these elements are described using XML files
  - "Descriptors"

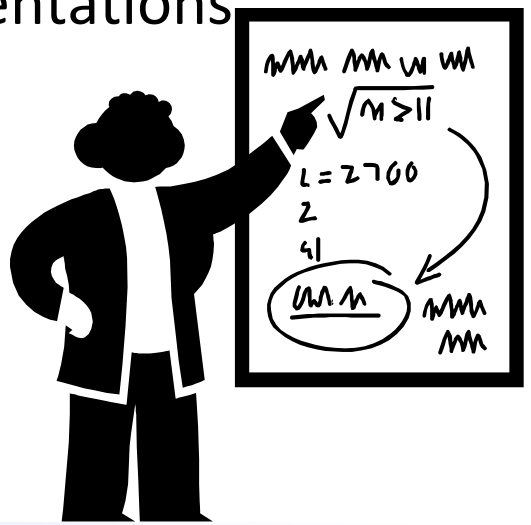


Accessors

- Different actions are needed by kernels
  - Context
  - Passing input data
  - Retrieving results
- Same kernel, various implementations
  - The same elements are needed
  - But the way to provide them can differ
    - E.g. to start a computation
      - Implementation 1 requires writing the value "0x00000001" in register 2
      - Implementation 2 requires writing the value "0x10000000" in register 4
- Accessors
  - Sets of specific interactions to execute to realize a particular action
    - Read / Write
    - Registers / Memory range / FIFO
    - …

Plan

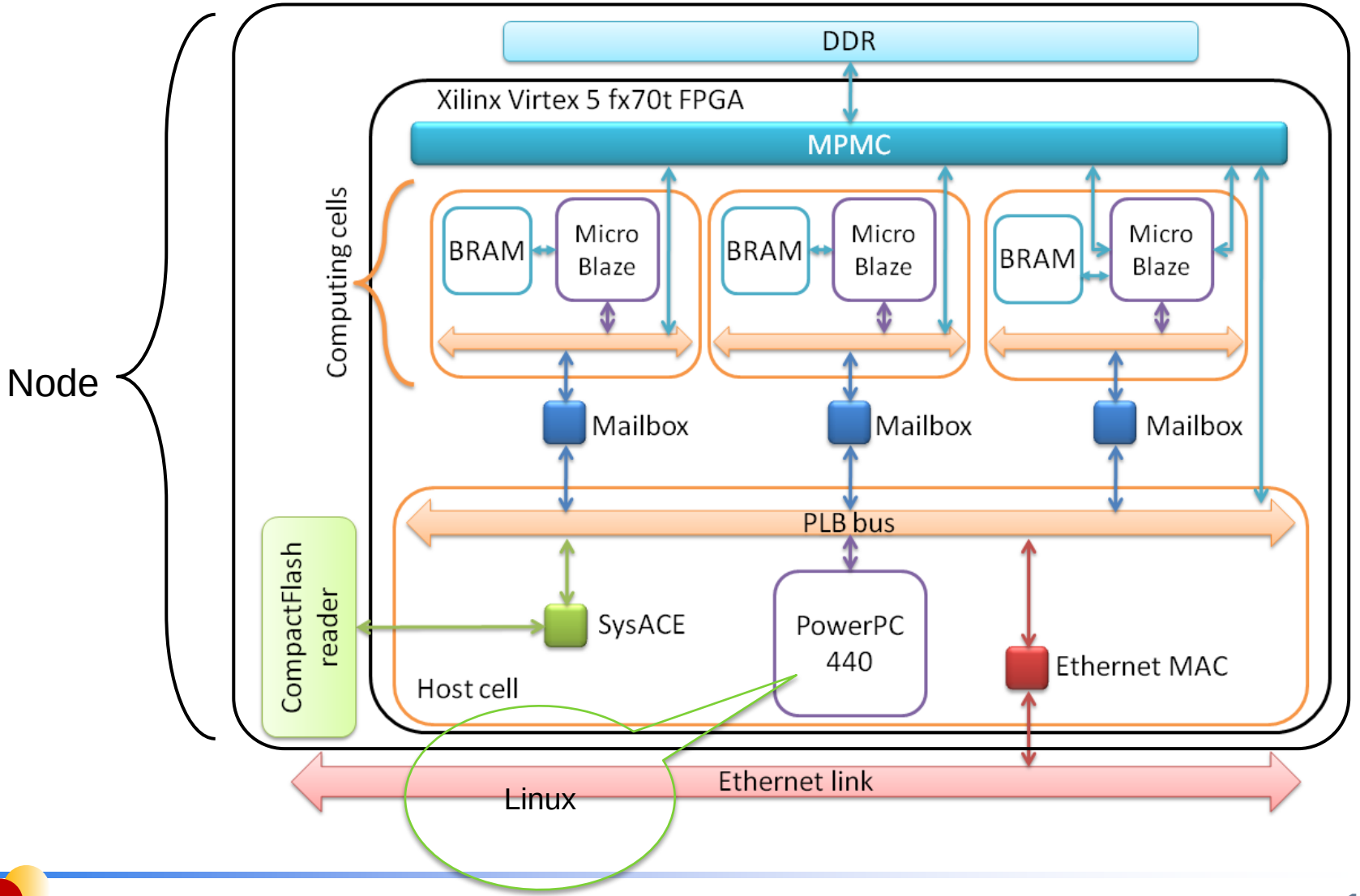
- *Parallel and reconfigurable computing today*
  - *Context*
  - *Systems limitations*
- *Our proposal: the SPORE system*
- Implementing SPORE
  - *Application design*
  - The hardware platform and its implementations
- *Conclusion & future works*



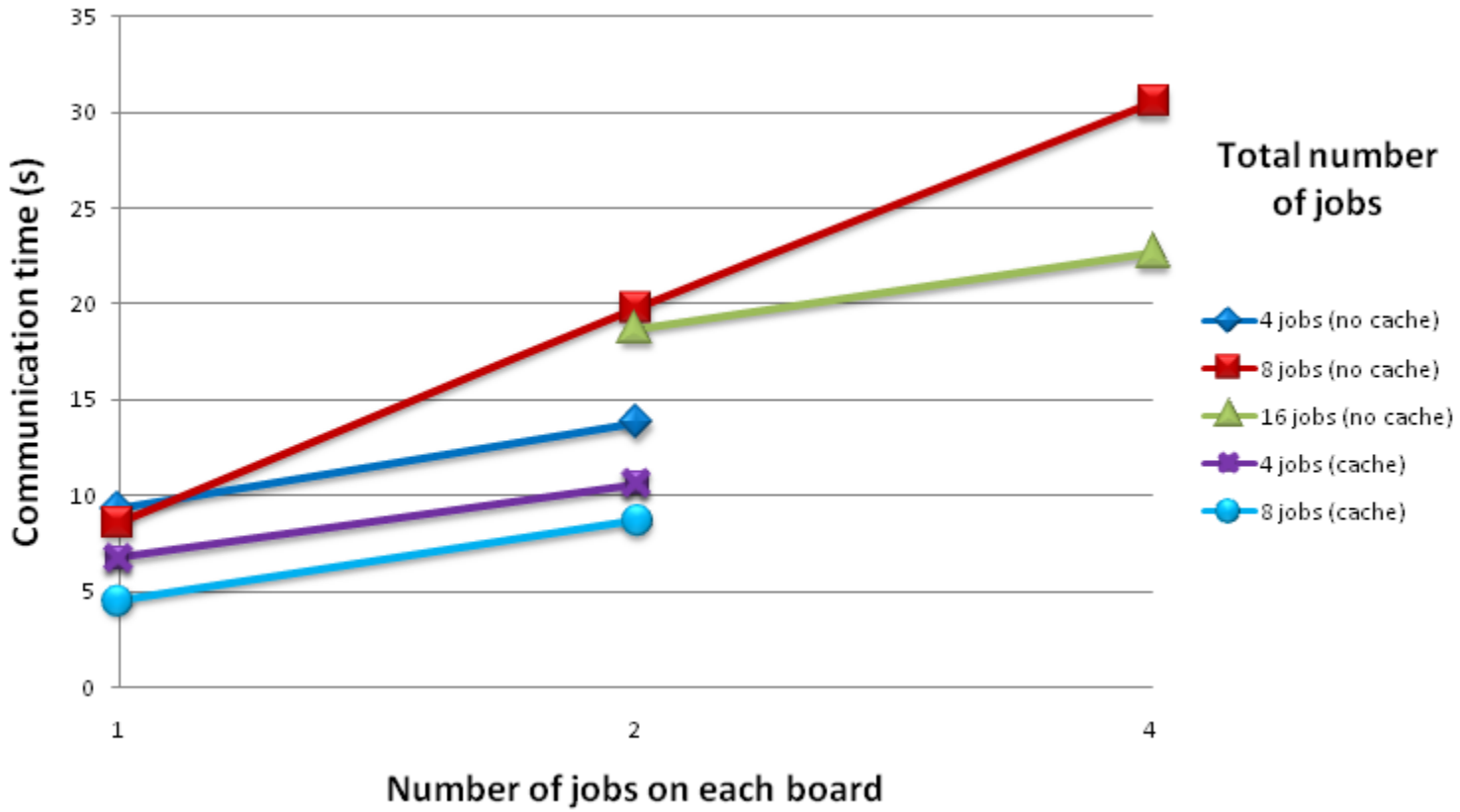
SPORE: 1st implementation [2]

- Purpose
  - Propose a HPC-like platform allowing evolving to reconfigurable architectures
  - Concept proof on actual implementation
- Architecture based on HPCs
  - Globally distributed – locally shared memory architecture
  - Implementing MPI for communication
  - Software only execution units
- Based on Xilinx ml507 board
  - Virtex 5 fx70t FPGA
    - Contains a PowerPC 440
  - 256 MiB DDR2
  - Ethernet interface
  - CompactFlash reader

SPORE: 1st implementation [2]



SPORE: 1st implementation results [2]

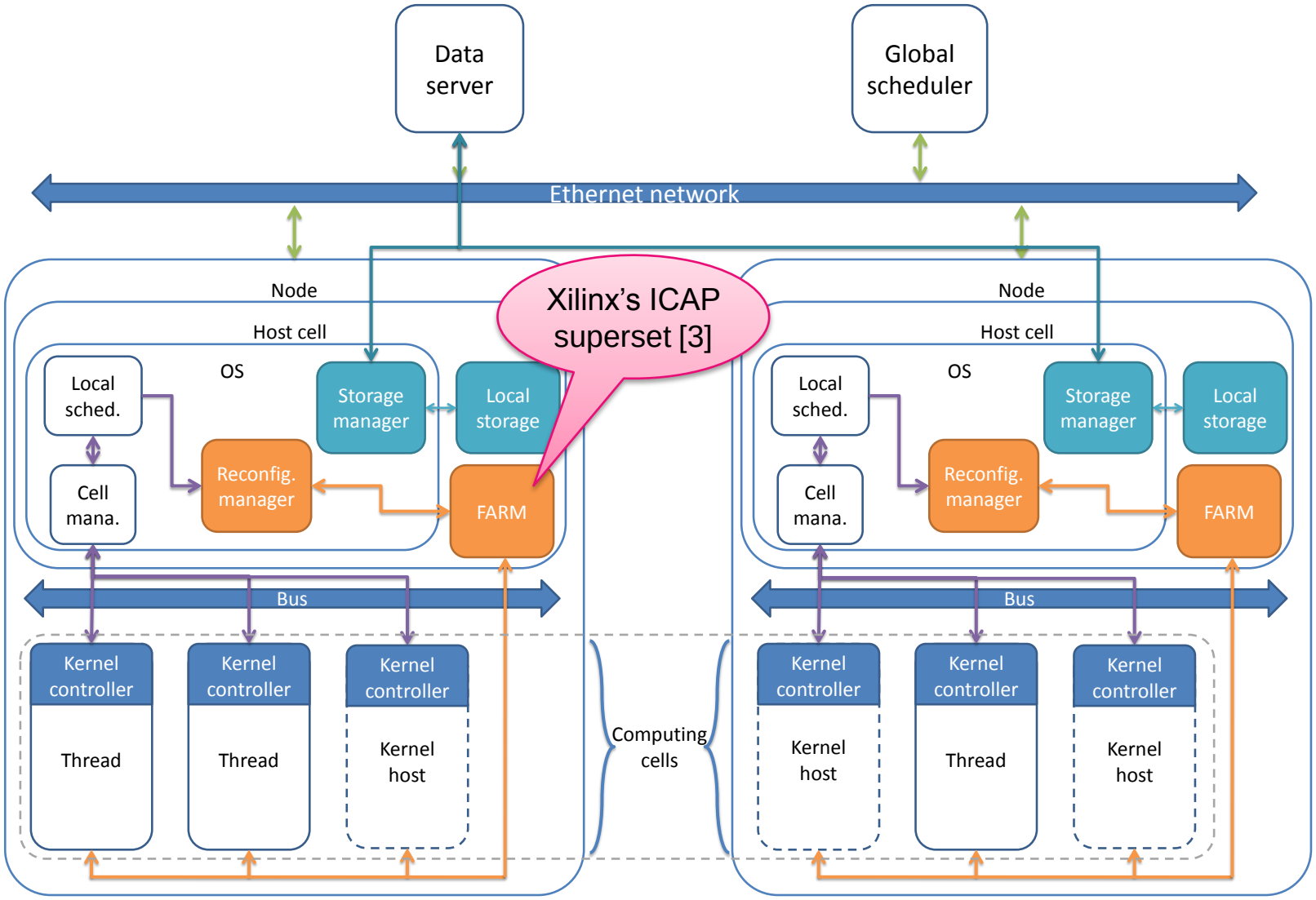


Communication time versus number of jobs/board

SPORE: 2nd implementation

- Application development flow concept proof
  - Includes reconfigurable hardware
  - Dynamic scheduling of kernels
  - Bus-based communication
    - Try to reduce memory issues
- No MPI communication

SPORE: 2nd implementation



SPORE: 2nd implementation – Control

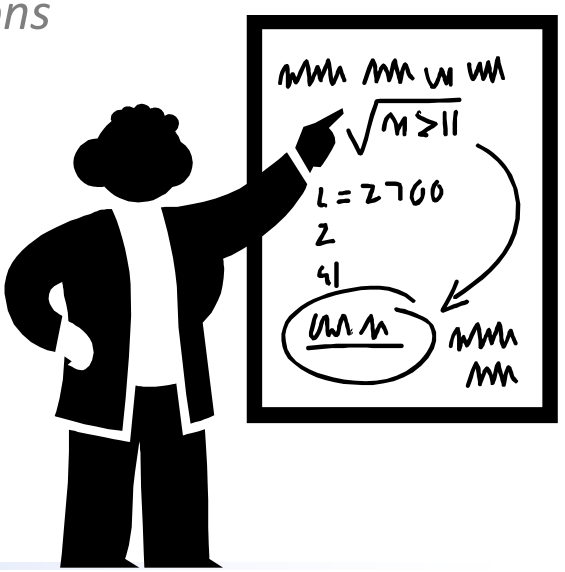
- Linux-based control
  - Scheduler
  - Data server communication
  - Dynamic reconfiguration
    - Linux driver for FARM
  - Cells management (accessors)
    - Linux driver for cells
  - XML parsing

SPORE: 2nd implementation results

- Only basic tests performed until now
  - Simple AES encrypt-then-decrypt application
    - 2 channels in parallel
- Fully functional for this basic test
  - XML-based application description
  - Hardware kernels reconfiguration
  - Kernels configuration and other accessors

Plan

- *Parallel and reconfigurable computing today*
  - *Context*
  - *Systems limitations*
- *Our proposal: the SPORE system*
- *Implementing SPORE*
  - *Application design*
  - *The hardware platform and its implementations*
- Conclusion & future works



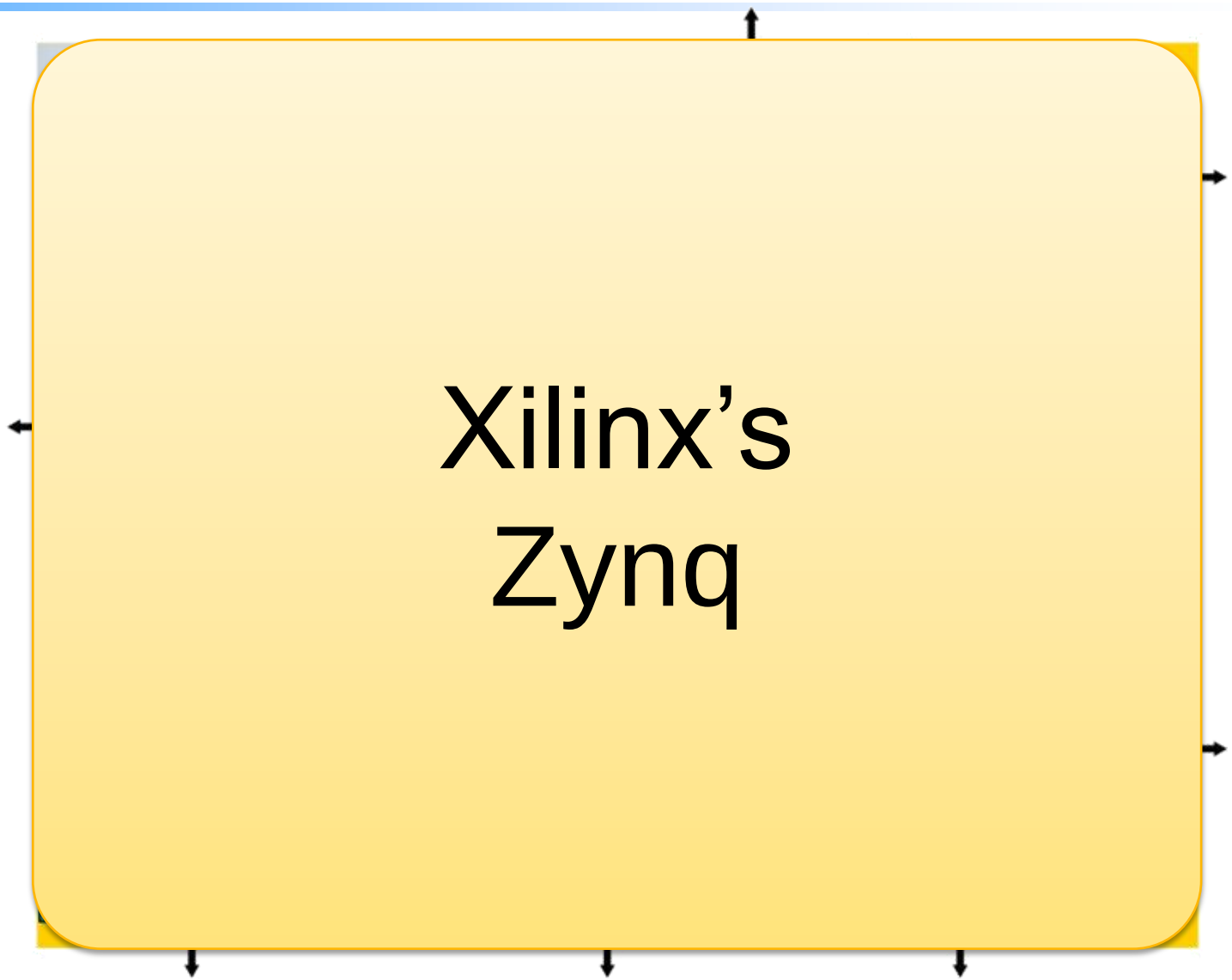
Conclusion

- SPORE is a platform virtualization tool
  - Can be adapted to any underlying reconfigurable hardware
- Preliminary SPORE implementations working
- General application flow validated
- Still need a complete SPORE implementation
  - Flow *and* HPC
  - Software *and* hardware

Future work

- 2nd platform
  - Perform further tests
    - Application containing more kernels
    - Video (H264)
- 3rd platform
  - Include elements from both previous
    - Software AND reconfigurable hardware
    - MPI communication
  - Improvements
    - NOC-based communication
- Scheduling
  - No real algorithm for now

And why not…



Questions

[Bibliography]

- [1]
  - Tarek El-Ghazawi, Esam El-Araby, Miaoqing Huang, Kris Gaj, Volodymyr Kindratenko, and Duncan Buell.
  - The promise of High-Performance Reconfigurable Computing.
  - Computer, 41 :69–76
  - February 2008
- [2]
  - Clément Foucher, Fabrice Muller, and Alain Giulieri.
  - Exploring FPGAs capability to host a HPC design.
  - 28th Norchip Conference (Norchip 2010), pages 1–4, Tampere Finland
  - November 2010
- [3]
  - François Duhem, Fabrice Muller, and Philippe Lorenzini.
  - FaRM: Fast reconfiguration manager for reducing reconfiguration time overhead on FPGA.
  - 7th International Symposium on Applied Reconfigurable Computing (ARC 2011), Belfast, United Kingdom
  - March 2011