Design Space Exploration in the context of SKA

Jean-François NEZAN

Computing for SKA

February 2016

With the courtesy of Kalray,

D. Ménard, M. Pelcat and K. Desnos

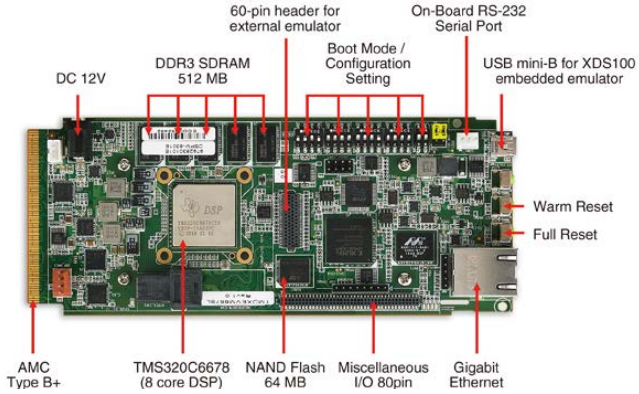
Introduction

- Many challenging applications
  - MPEG Codecs
    - MPEG4 Part2, AVC, SVC, HEVC, SHVC
  - Computer Vision and 3D Processing
    - Stereo Vision, SLAM
  - Cryptography
    - Chaotic-based Cryptography
  - Telecommunications
    - 3GPP LTE eNodeB
  - Square Kilometer Array project (SKA)
- Design Space Exploration : What is the best hardware ?

Introduction

Texas Instruments Keystone II

8x DSP cores @ 1.2 GHz – 4x ARM Cortex A15 @ 1.4 GHz – 6 MB memory + DDR – 650$ - 28nm – 16 Watts

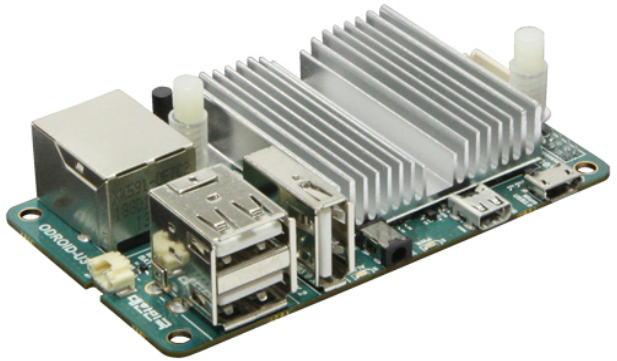


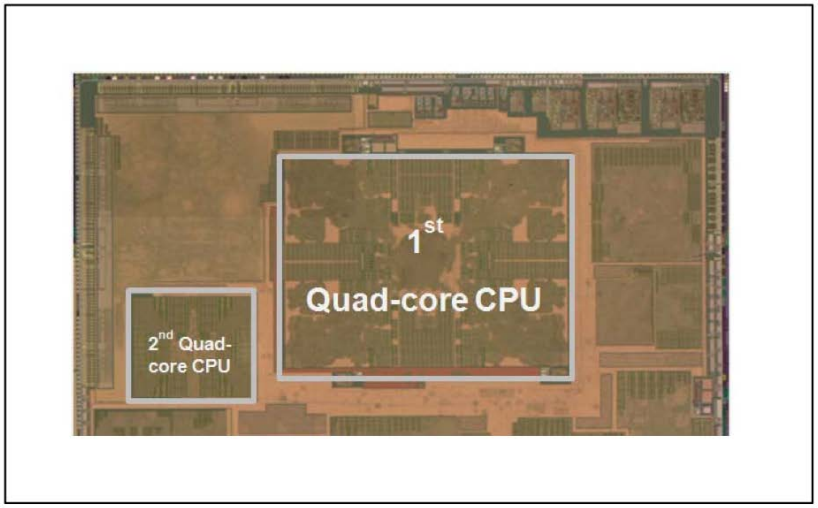
Die Photo

- Potential Candidates
  - Multicore

Odroid with Samsung Exynos 5

4x Cortex A15 @ 1.8 GHz– 4x Cortex A7 @ 1.2 GHz
+ PowerVR GPU – 50$ - 28 nm – 1Watt to 6 Watts





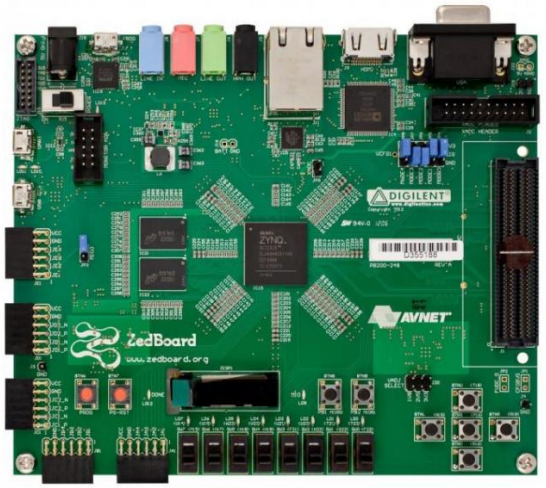
Introduction

- Potential Candidates

Zboard with Xilinx Zynq

2x ARM Cortex A9 @ 1 GHz – 444 Logic Cells

512 MB memory + DDR – 400$ - 28nm – 3 or 4 Watts ?



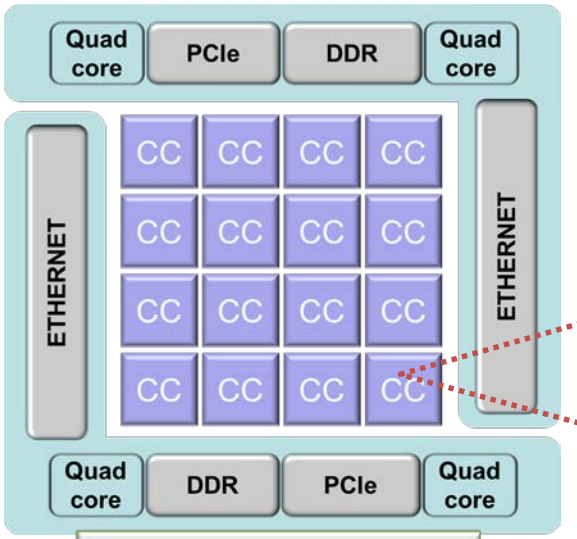
Nvidia GeForce GTX Titan X

3072 cores – 12 Go memory – 1200€ – 1 GHz – 28nm – 313 Watts

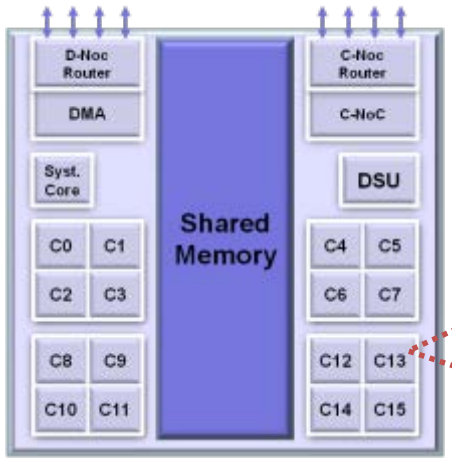
Introduction

- Potential Candidates
  - Manycore (NUMA architecture)

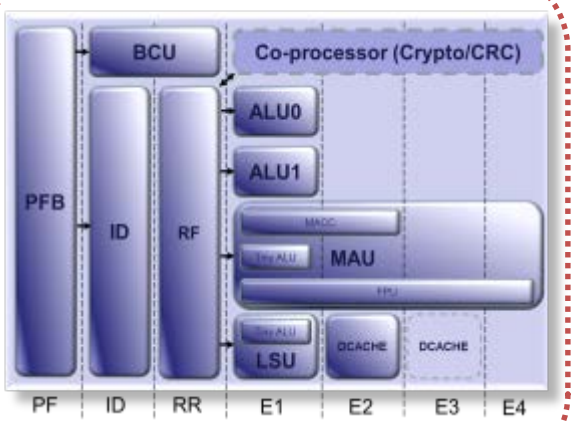
Kalray MPPA



Manycore Processor



Compute Cluster



VLIW Core

- 16 compute clusters
- 2 I/O clusters each with quad-core CPUs, DDR3, 4 Ethernet 10G and 8 PCIe Gen3
- Data and control networks-on-chip
- Distributed memory architecture
- 634 GFLOPS SP for 25W @ 600Mhz

- 16 user cores + 1 system core
- NoC Tx and Rx interfaces
- Debug & Support Unit (DSU)
- 2 MB multi-banked shared memory
- 77GB/s Shared Memory BW
- 16 cores SMP System

- 32-bit or 64-bit addresses
- 5-issue VLIW architecture
- MMU + I&D cache (8KB+8KB)
- 32-bit/64-bit IEEE 754-2008 FMA FPU
- Tightly coupled crypto co-processor
- 2.4 GFLOPS SP per core @600Mhz

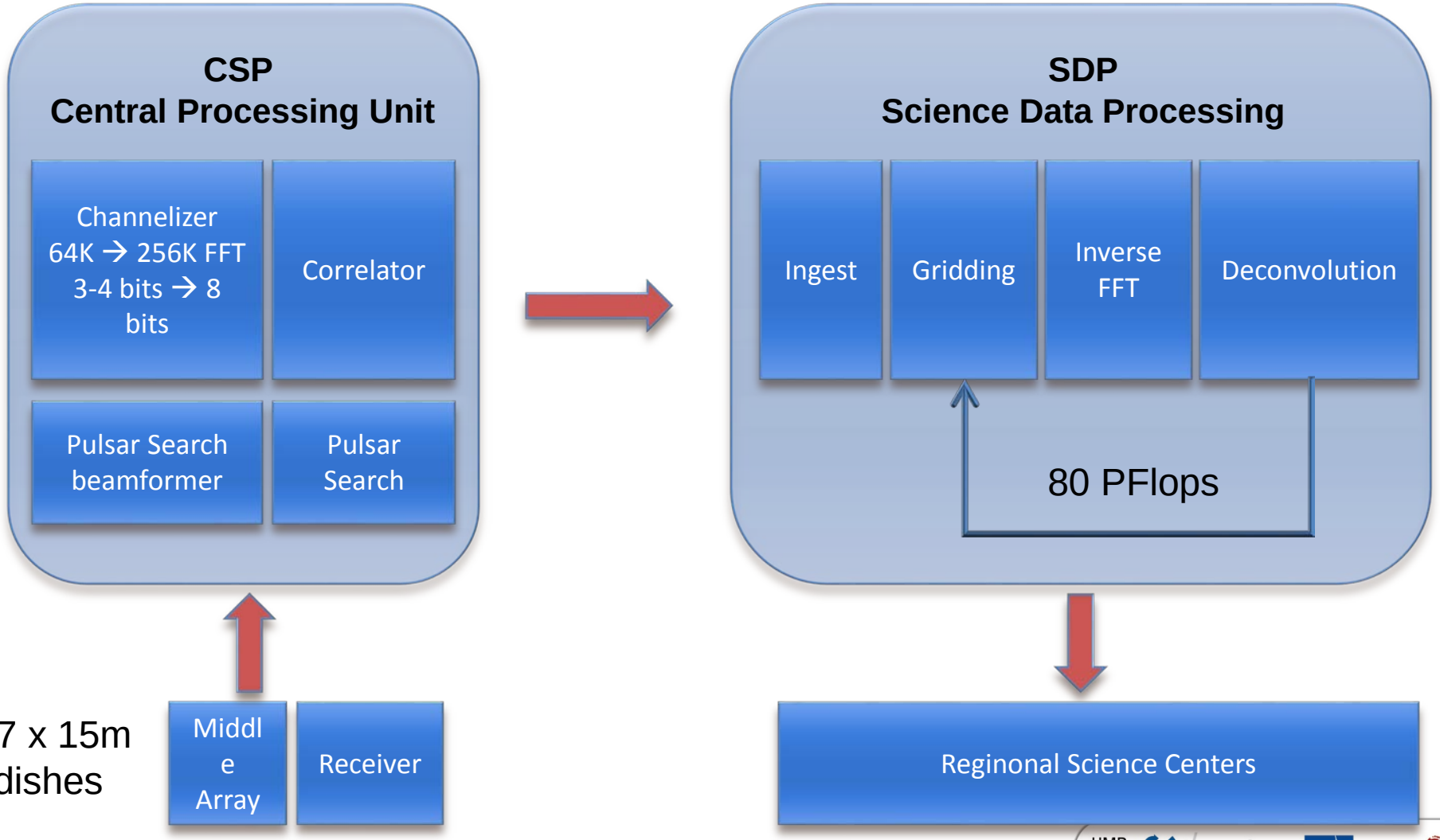
Outline

- Design Space Exploration : What is the best hardware ?

1. Introduction
2. Case study : SKA CSP Channelizer on MPPA
3. Challenges for DSE
4. PiSDF : a parallel Model of Computation for DSE
5. PREESM : a tool for DSE

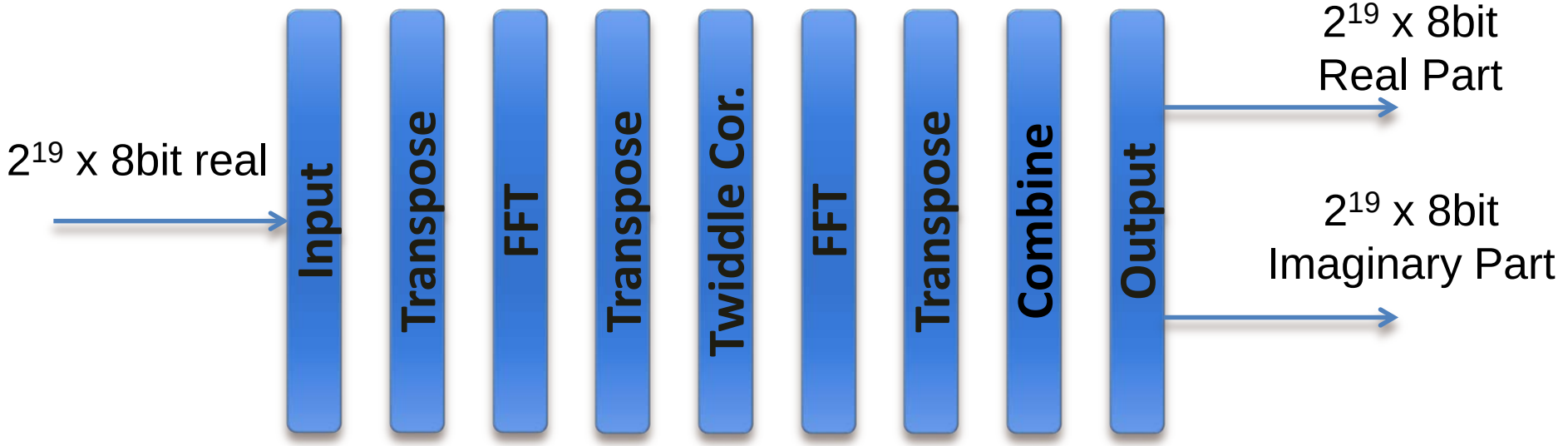
SKA CSP Channelizer on MPPA

- Square Kilometer Arr80ay Project in South Africa



SKA CSP Channelizer on MPPA

- CSP Channelizer application
  - 2^{19} real FFT in 524 µs
  - Main constraints : throughput and energy
  - Selection of both algorithm & architecture



SKA CSP Channelizer on MPPA

- Profiling of 2^{19} real FFT (requirement of 524 µs) including data transfers

Implementation	MPPA Compute Cluster Andey @ 400MHz (ms)	MPPA-256 Andey @ 400MHz (ms)	MPPA-256 Bostan @ 400MHz (ms)	MPPA-256 Bostan @ 600MHz (ms)
Fully Fixed-point – 6 steps	11,64	1,07	0,975	0.65
Mix Float Implementation Floating-point in FFT stages with fixed-point storage in the middle of the 6-step	13.70	1.17	1,05	0.70

Fully Fixed-point – 6 steps	MPPA-256 Andey @ 400MHz	MPPA-256 Bostan @ 400MHz	MPPA-256 Bostan @ 373 MHz	MPPA-256 Bostan @ 600MHz
Power consumption of 1 MPPA	8.7 W	8.7 W	8 W	17W
Number of MPPA	2 + 1 cluster	1 + 14 clusters	2	1 + 4 clusters
Power consumption	17,95 W	16,3 W	16 W	22 W

SKA CSP Channelizer on MPPA

MPPA® MANYCORE Processor Roadmap

2014	2015	2016	2017

MPPA®-1024

Volume

Samples Volume

Samples Volume

MPPA®-256

MPPA®-64

TSMC 28nm (HP)		28nm/16nm
Andey	**Bostan**	**Coolidge**
Kalray 1st generation	Kalray 2nd generation	Kalray 3rd generation
211 GFLOPS SP	634 GFLOPS SP	1056 GFLOPS SP
70 GFLOPS DP	316 GFLOPS DP	527 GFLOPS DP
32-bit addressing	64-bit addressing	64-bit addressing
400Mhz	600Mhz	1000Mhz

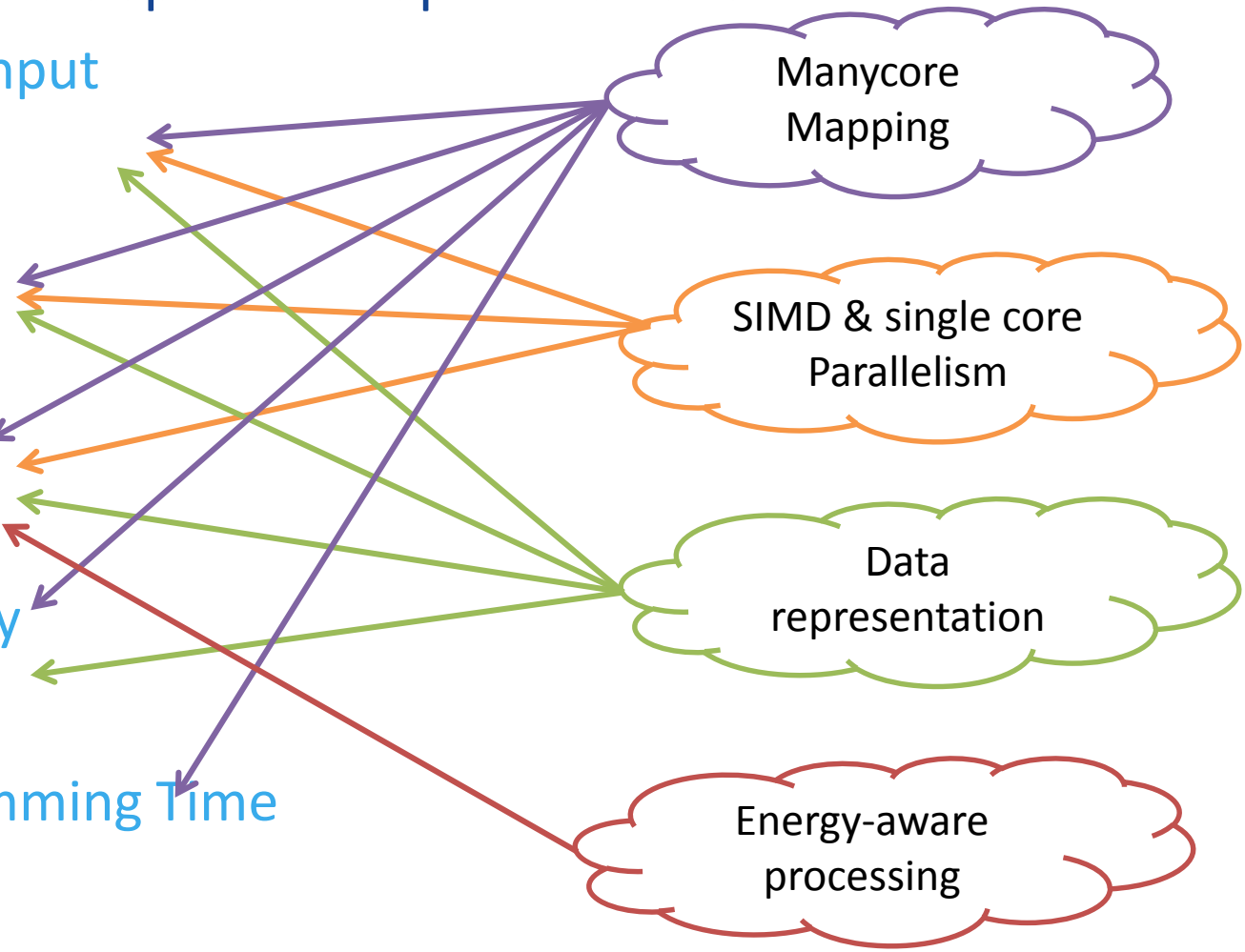
Production

Development

Under Specification

SKA CSP Channelizer on MPPA

- Architecture dependent optimizations
  - Throughput
  - Latency
  - Energy
  - Memory
  - Programming Time



SKA CSP Channelizer on MPPA

- Matlab Code translation
  - Time consuming
  - Error prone
- Evolution of the algorithm requirements
  - Discussions around accuracy
  - Future deployments (1st in 2018 and several ones until 2030)
- Evolution of the Hardware
  - Number of Cores
  - Core architecture
  - Technology

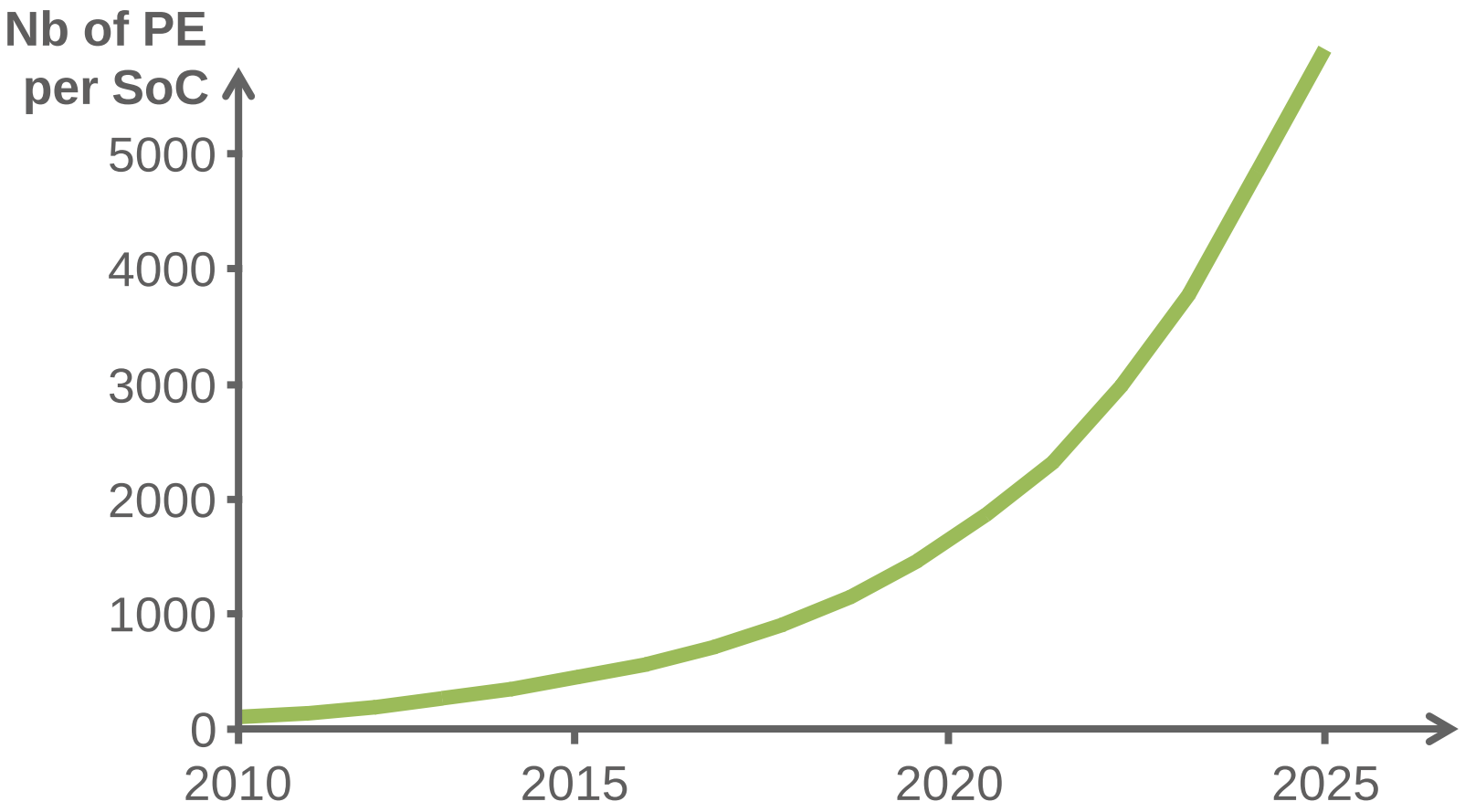
- Same work to be done on FPGA, GPU, Multicore ...

Outline

- Design Space Exploration : What is the best hardware ?

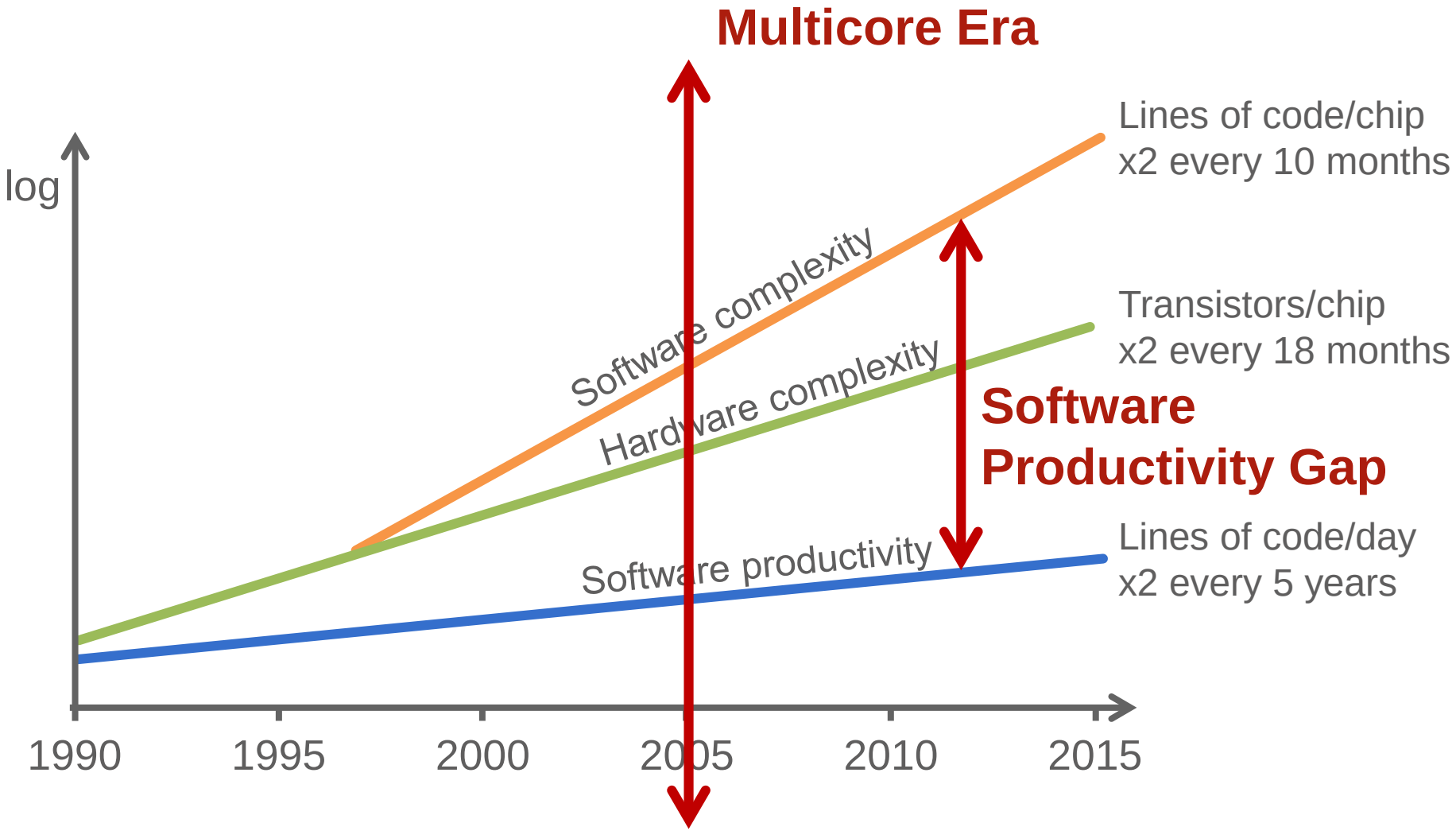
1. Introduction
2. Case study : SKA CSP Channelizer on MPPA
3. Challenges for DSE
4. PiSDF : a parallel Model of Computation for DSE
5. PREESM : a tool for DSE

Hardware Complexity



Source: ITRS System Drivers 2011

Challenges for Design Space Exploration



Source: ITRS & *Hardware-dependent Software*, Ecker et al., Springer

Challenges for Design Space Exploration

- Specification of parallel algorithms
  - Throughput/Latency evaluation
  - Predictable memory footprints
  - Legacy code reuse
- Working prototypes
  - Seamless porting to an existing platform (CPU, GPU, NUMA …)
  - Guaranteed deadlock-freeness
  - Inter-PE communications
- Virtual prototypes
  - Number of cores
  - Memory size and technology
  - Hierarchical topologies (memories and clusters of cores)

Challenges for Design Space Exploration

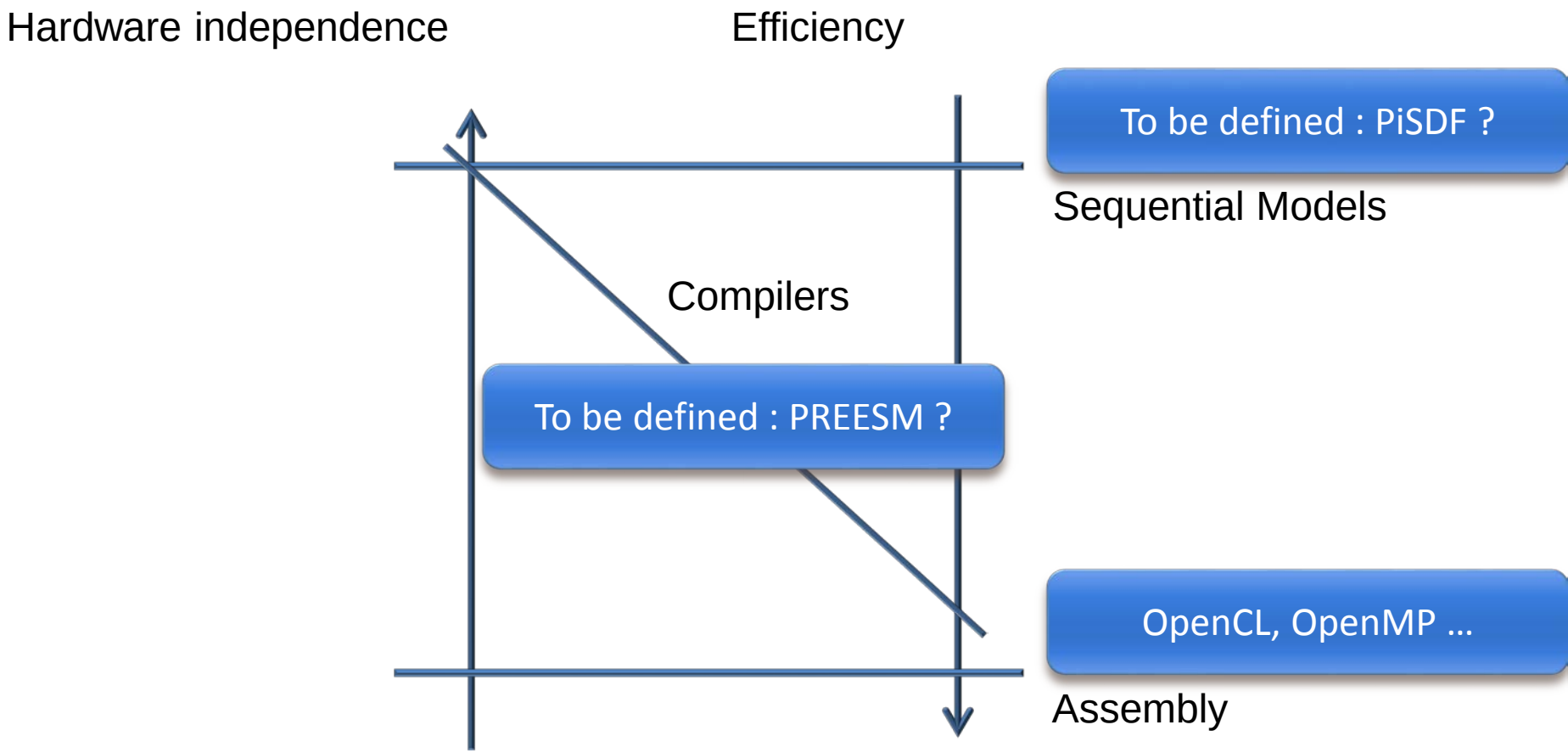
- From sequential to parallel Models of Computations (MoCs)

	Legacy code	Shared/Dist. memories	GPU	Mapping/scheduling	Heterogeneous MPSoC	Hardware dependance
VHDL						
Vivado HLS						
OpenCL						
OpenMP3						
OpenMP4						
PThread						

  - OpenACC, M-TAPI, Open Data Plane (ODP) …

Challenges for Design Space Exploration

- Analogy with sequential models, tools and methods



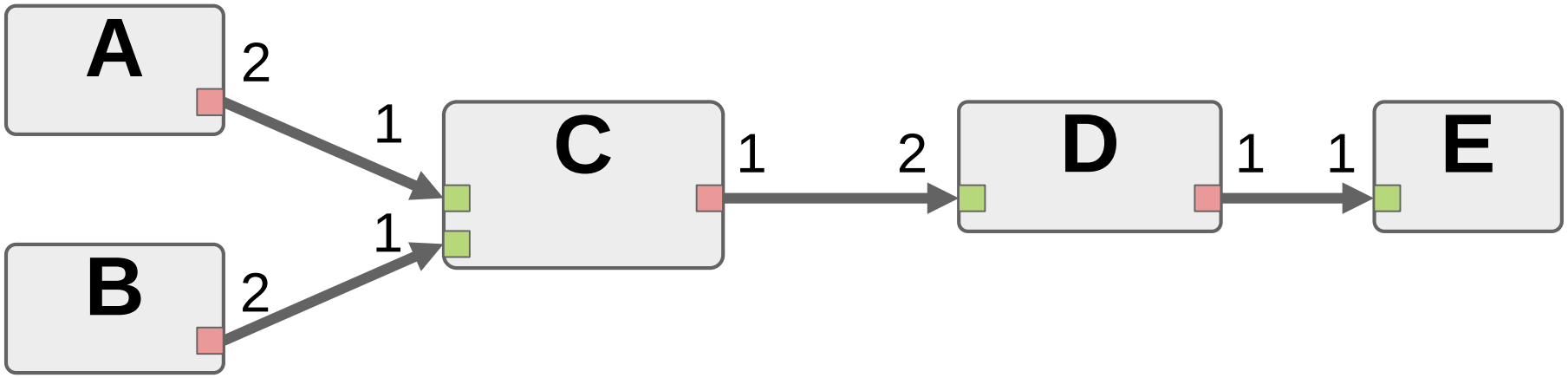
Outline

- Design Space Exploration : What is the best hardware ?

1. Introduction
2. Case study : SKA CSP Channelizer on MPPA
3. Challenges for DSE
4. PiSDF : a parallel Model of Computation for DSE
5. PREESM : a tool for DSE

Algorithm descriptions using Dataflow Graphs

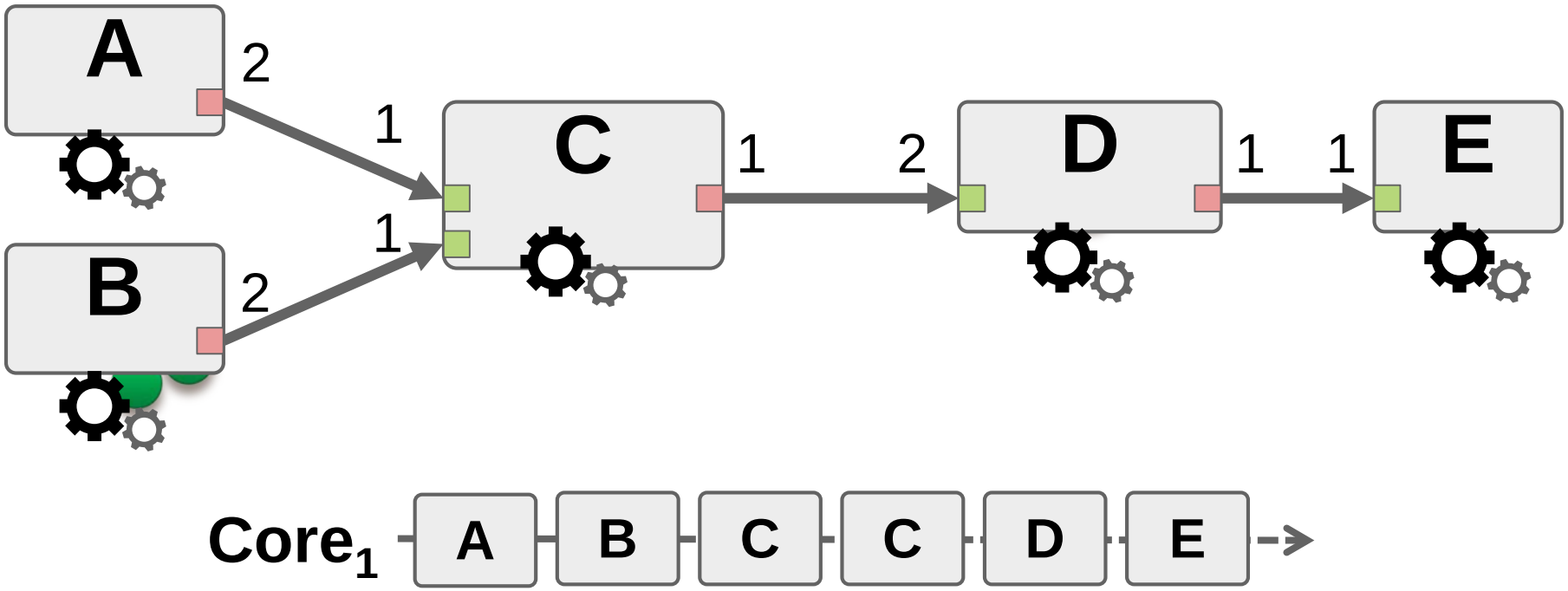
- Synchronous Dataflow (SDF)
  - Actors and Data ports
  - FIFO queues



E. Lee and D. Messerschmitt, *"Synchronous data flow"*, Proceedings of the IEEE, 1987.

Algorithm descriptions using Dataflow Graphs

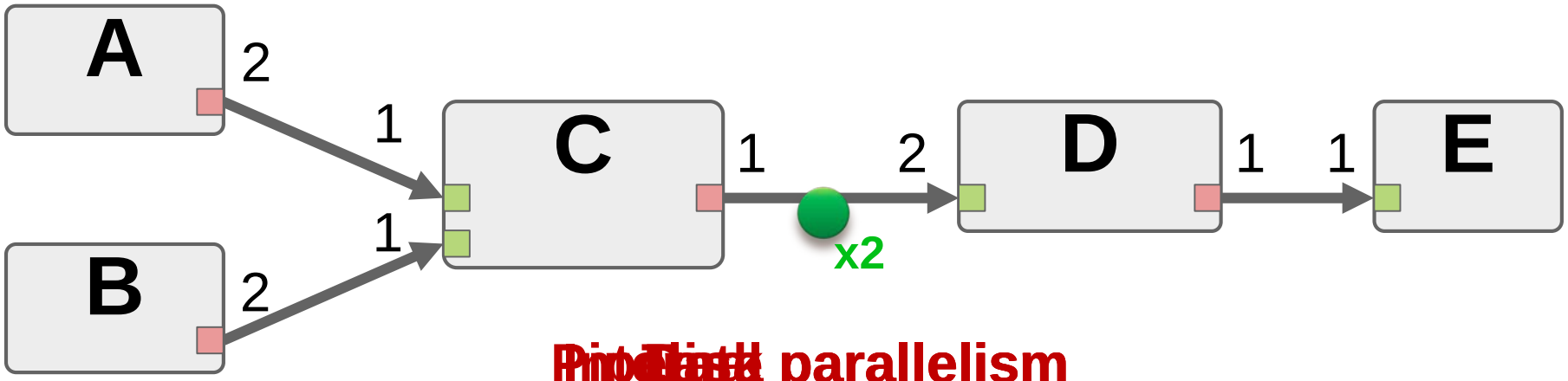
- Data-driven execution
  - An actor is fired when its input FIFOs contain enough data-tokens.

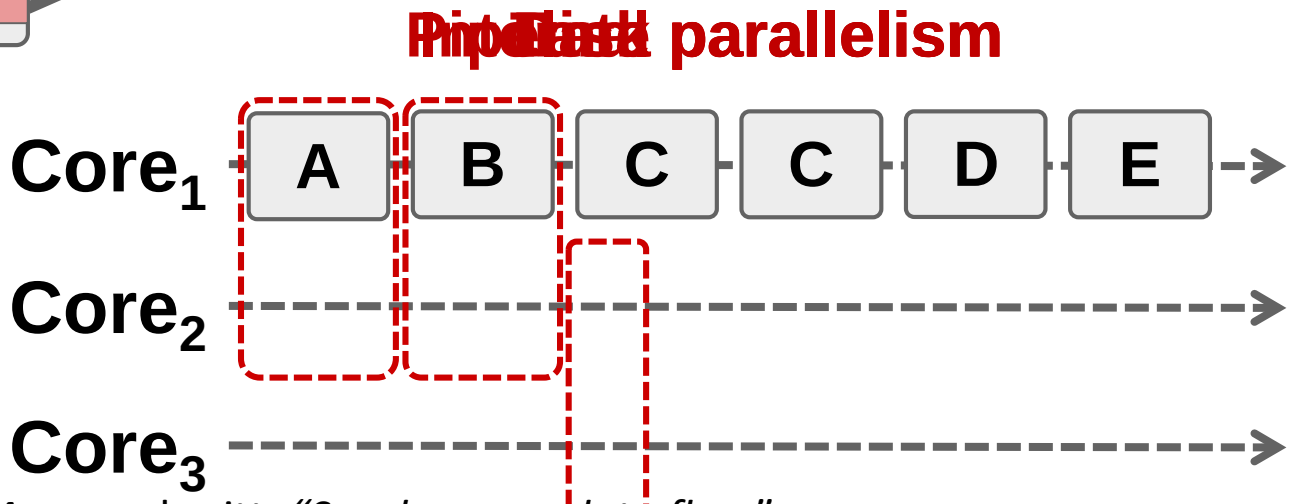


E. Lee and D. Messerschmitt, *"Synchronous data flow"*, Proceedings of the IEEE, 1987.

Algorithm descriptions using Dataflow Graphs

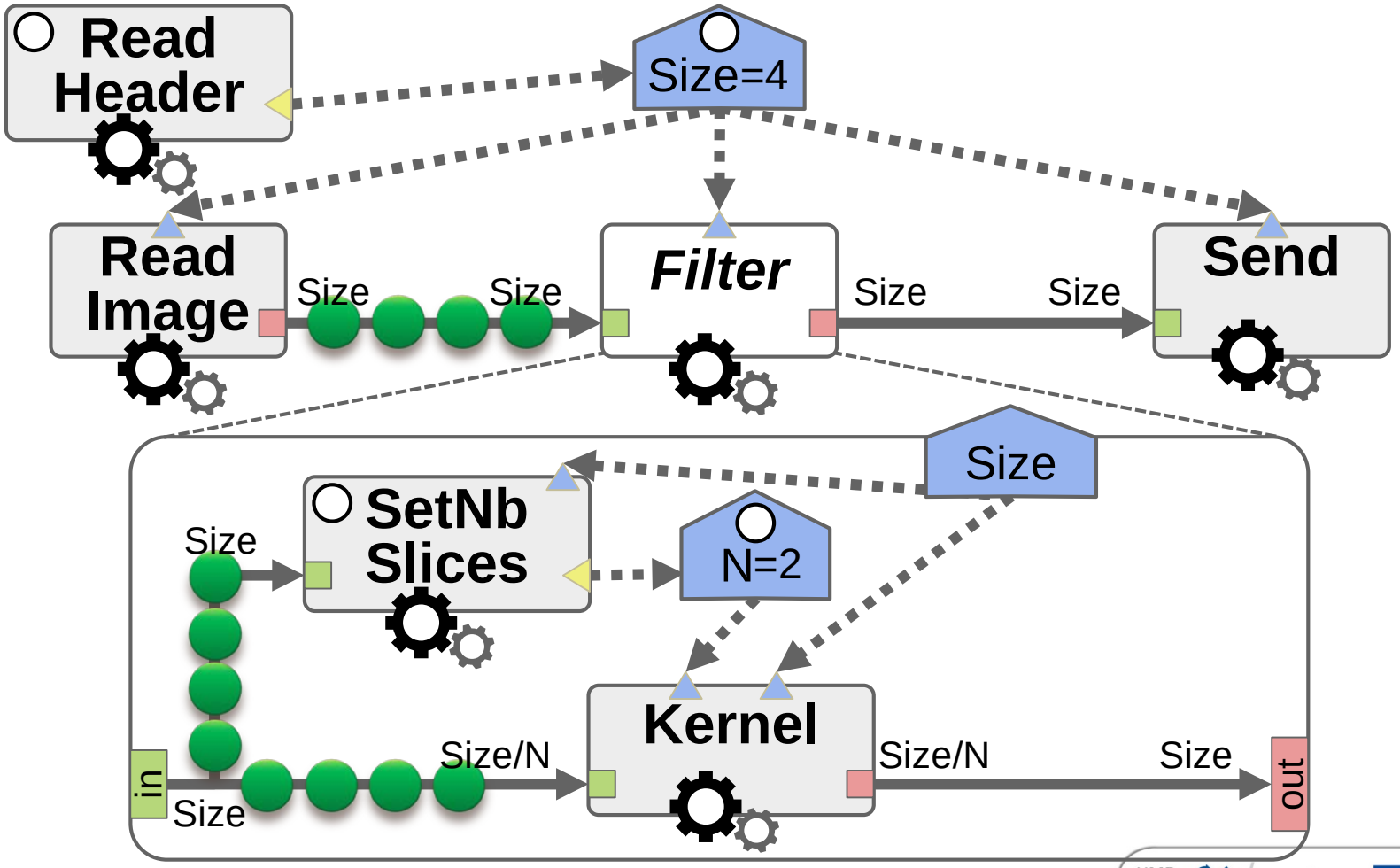
- Expression of parallelisms: Task / Data / Pipeline / Internal





E. Lee and D. Messerschmitt, *"Synchronous data flow"*, Proceedings of the IEEE, 1987.

PiSDF (Parameterized and Interfaced Synchronous Dataflow)



PiSDF (Parameterized and Interfaced Synchronous Dataflow)

- PiSDF is:
  - Hierarchical & Compositional
  - Statically parameterizable (for compile time analysis)
  - Dynamically reconfigurable (at runtime)
- PiSDF fosters:
  - Predictability
  - Parallelism
  - Lightweight runtime overhead
  - Developer-friendliness

K. Desnos, M. Pelcat, J.-F. Nezan, S. S. Bhattacharyya, S. Aridhi “PiMM: Parameterized and Interfaced Dataflow Meta-Model for MPSoCs Runtime Reconfiguration”, SAMOS XIII

Outline

- Design Space Exploration : What is the best hardware ?

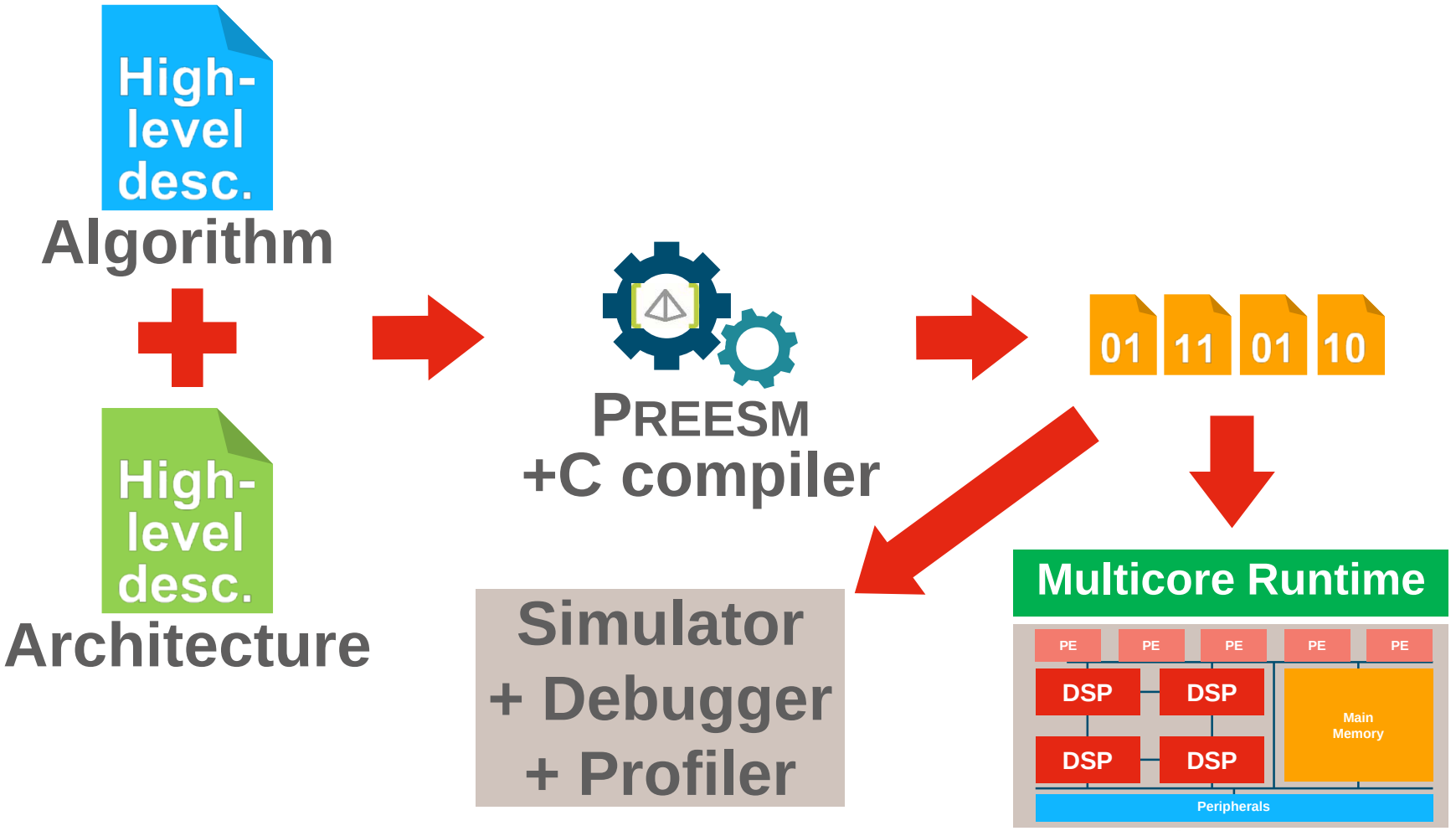
1. Introduction
2. Case study : SKA CSP Channelizer on MPPA
3. Challenges for DSE
4. PiSDF : a parallel Model of Computation for DSE
5. PREESM : a tool for DSE

PREESM : a tool for Design Space Explorations

What is PREESM IS :

- Open Source Tool
  - Available on GitHub
- Research-Oriented Tool
  - New models, optimizations, scheduling
- Eclipse-based Integrated Tool
  - Several plug-ins, metamodels
- Extended Web Tutorials
  - http://preesm.sourceforge.net/website

PREESM : a tool for Design Space Explorations



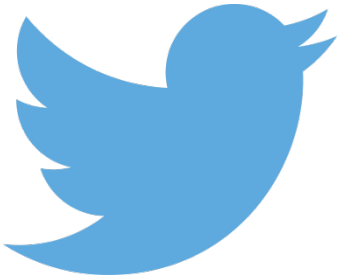
PREESM : a tool for Design Space Explorations

- PREESM Main features :
  - Automatic mapping / scheduling
    - Working and virtual prototypes
  - Memory analysis : bounding the memory needs
  - Memory optimisation : Buffer merging techniques
  - Power optimization techniques
  - Code generation
    - Bare metal (static parameters)
    - Specific runtime (dynamic parameters)

- Ask for a Demo !!

Questions ?

http://preesm.sf.net



@PreesmProject

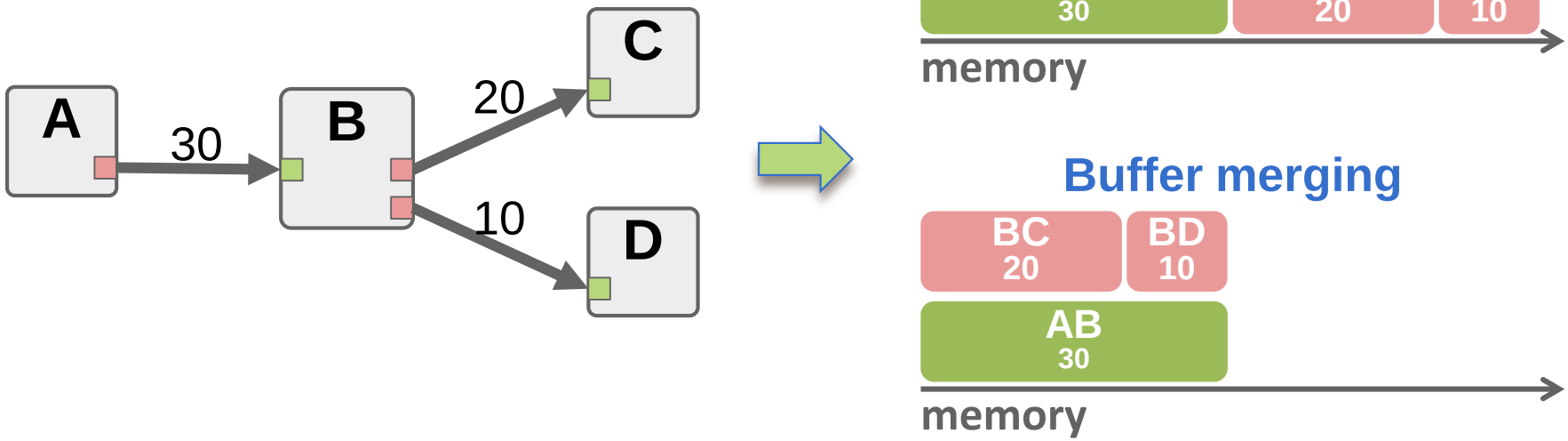
- Backup slides

Memory optimizations

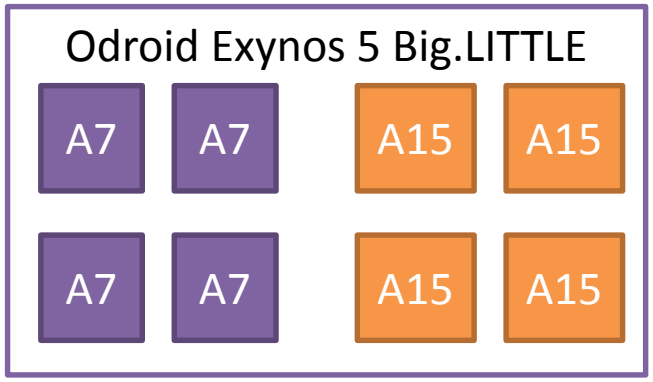
- Bounding the memory needs of an application graph to
  - Evaluate the memory requirements
  - Adjust the size of architecture memory
  - Assess the optimality of a memory allocation

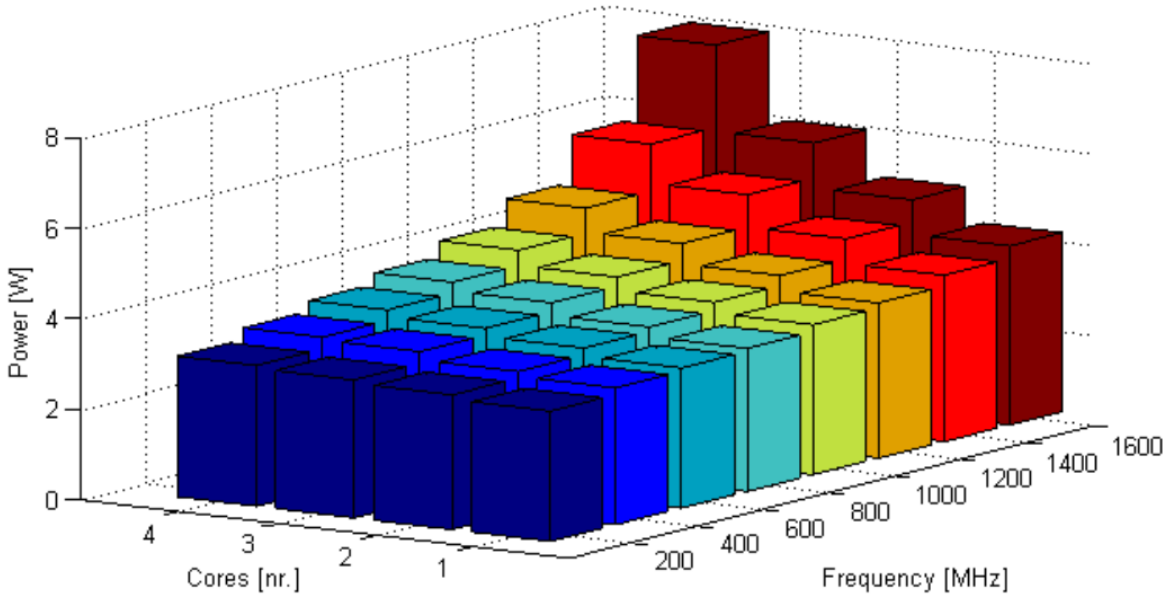
Memory optimizations

- Buffer merging technique for SDF graphs
  - 48% less memory than state-of-the-art techniques

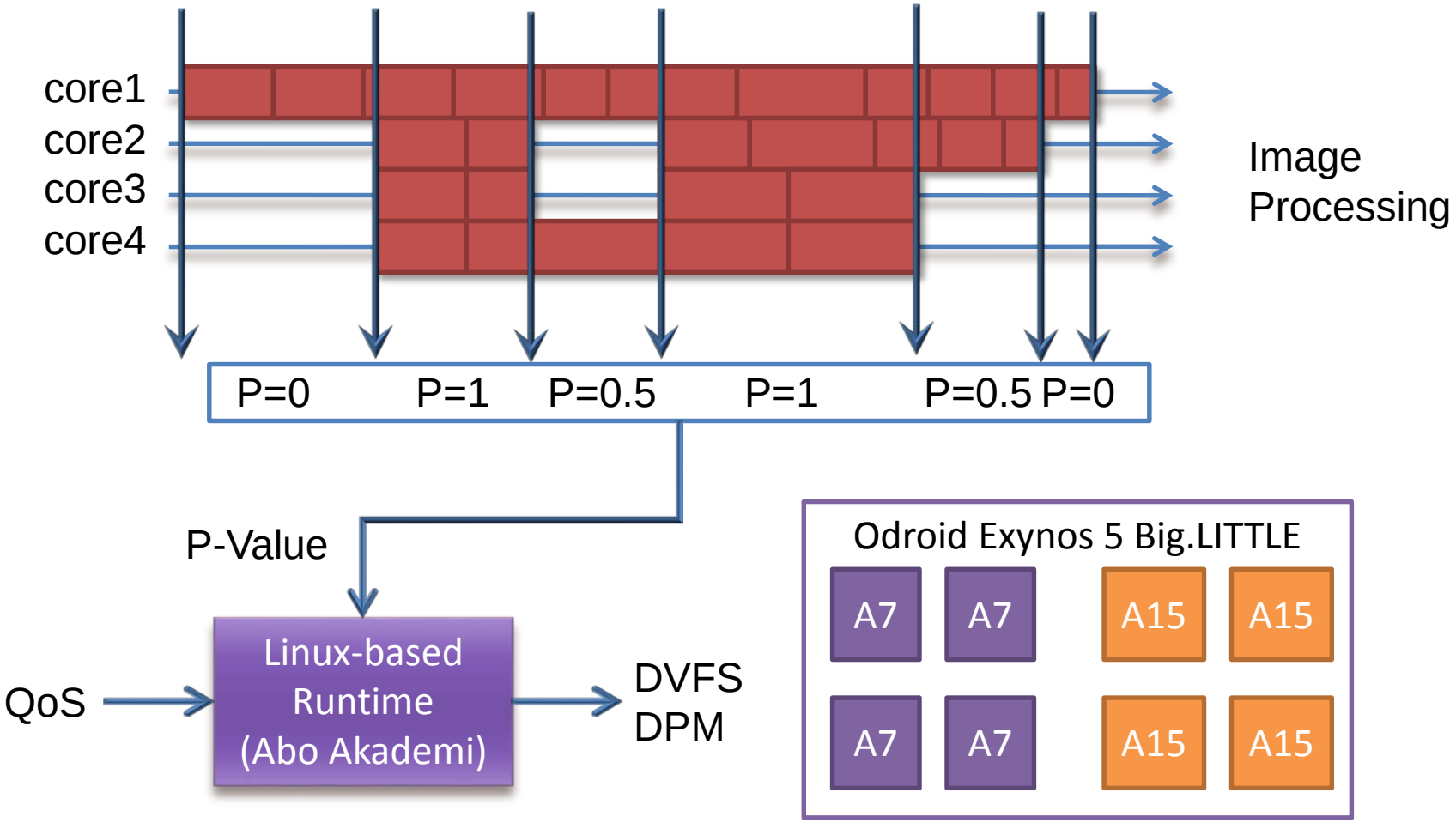


Energy optimization: platform Exynos 5





Energy optimization setup



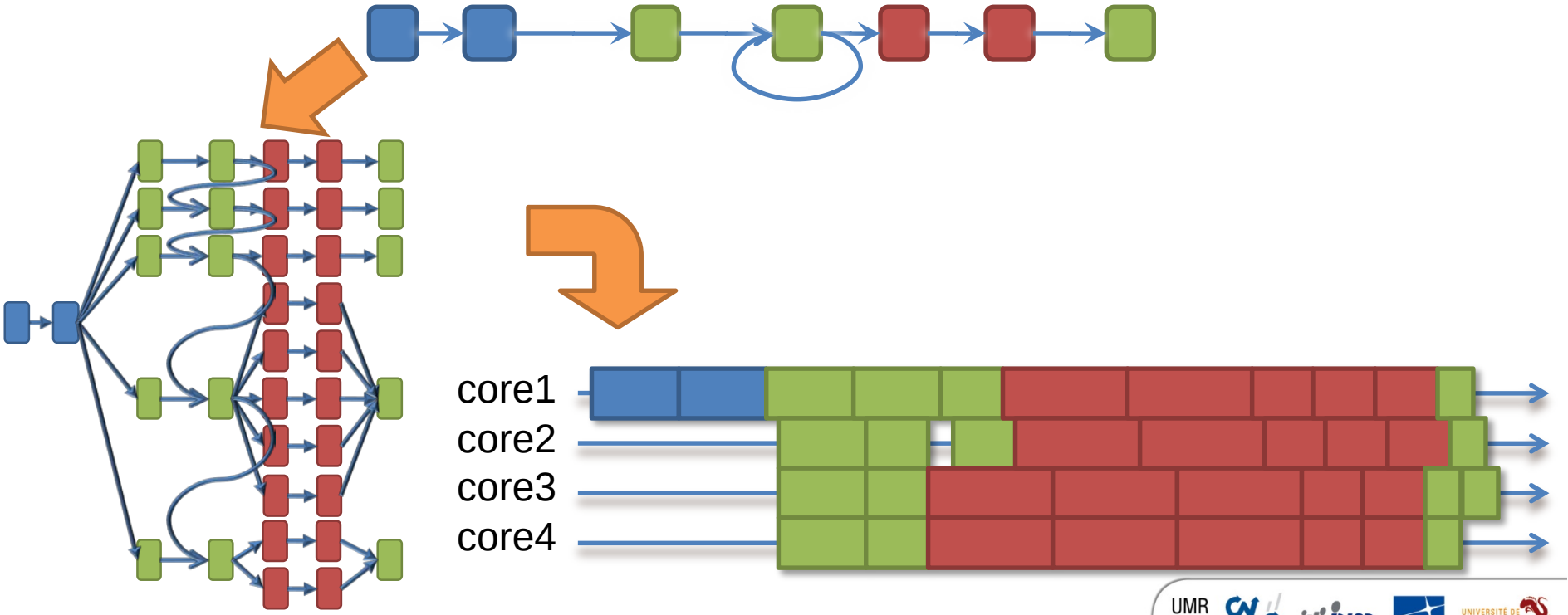
20% energy savings on a parallel Sobel + sequential postprocessing wrt. Linux completely fair scheduler and on-demand governor

Code generation for multiple targets

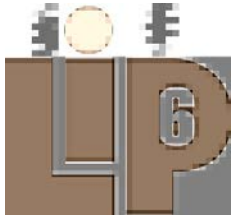
- Multi-C6X DSPs:
  - TMS320c6678 from Texas Instruments
  - Supports the activation of the DSP caches.
- Multi-x86 and multi-ARM CPUs:
  - Linux and Windows, pthread
- OMAP4 heterogeneous platform:
  - dual-core ARM Cortex-A9, 2 Cortex-M3, and a C64xT DSP

Mapping/Scheduling

- State-of-the-art algorithms (FAST, List, …)
- Latency and load balancing optimization
- Customizable accuracy (w.r.t. communications)



PREESM : academic partners



PREESM : industrial partners

PREESM Inputs

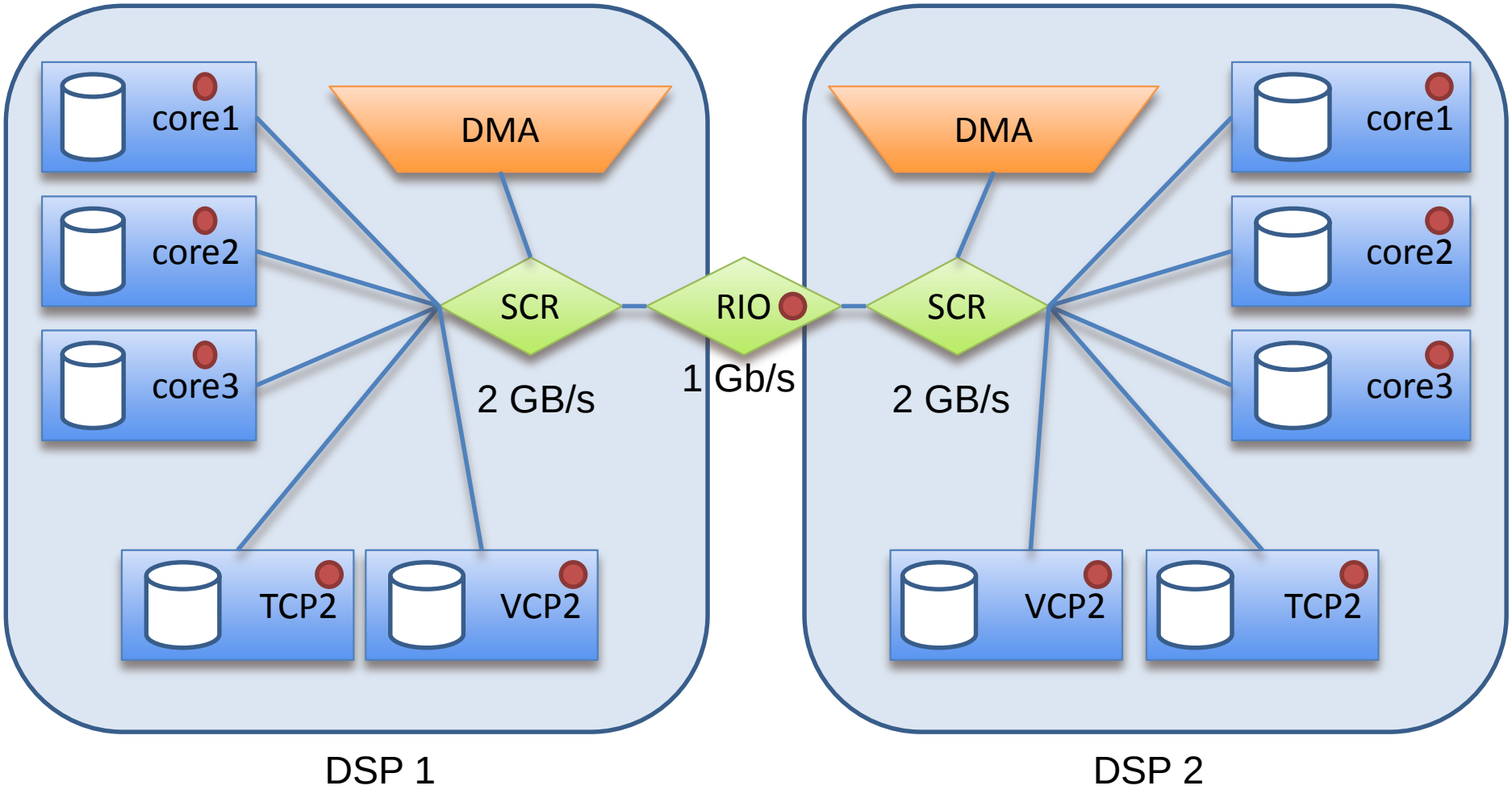
Algorithm/Architecture independence

- PiSDF graphs are architecture-independent
- S-LAM graphs are application-independent

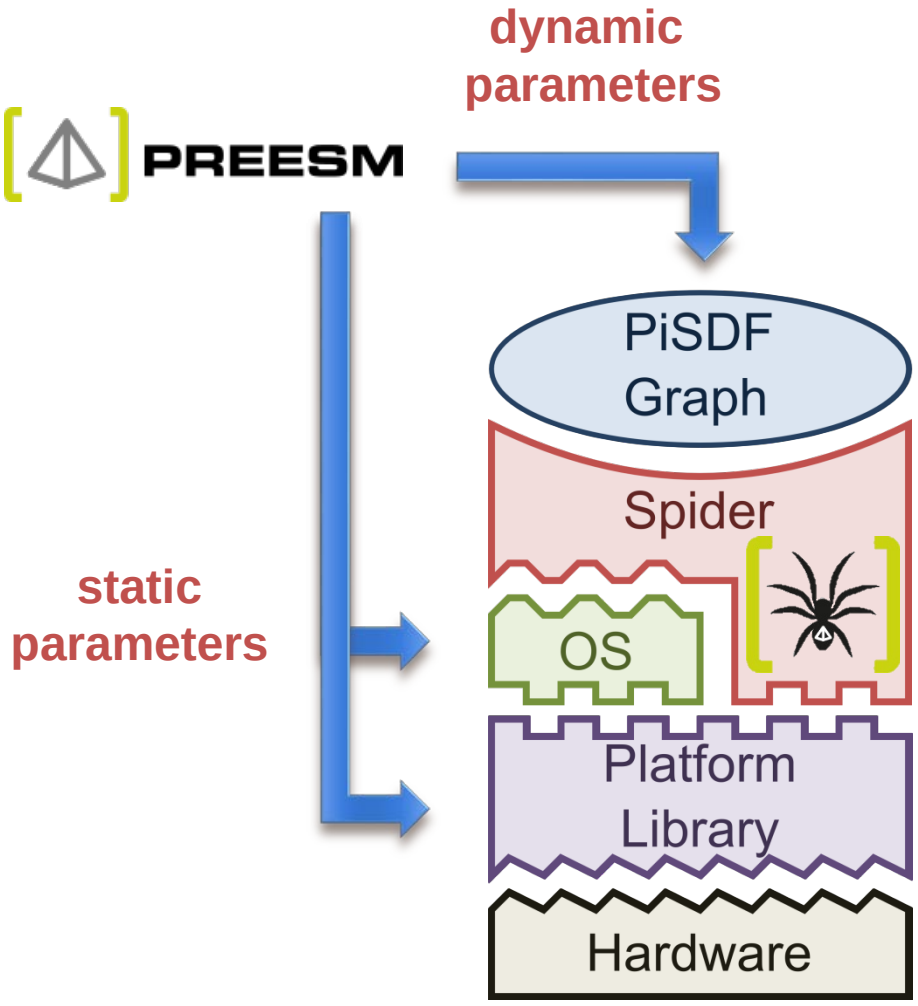
Scenario

- Define information/constraints for the deployment of a specific algorithm on a specific architecture
  - Mapping constraints
  - Heterogeneous timing constraints
  - …

S-LAM (System-Level Architecture Model)

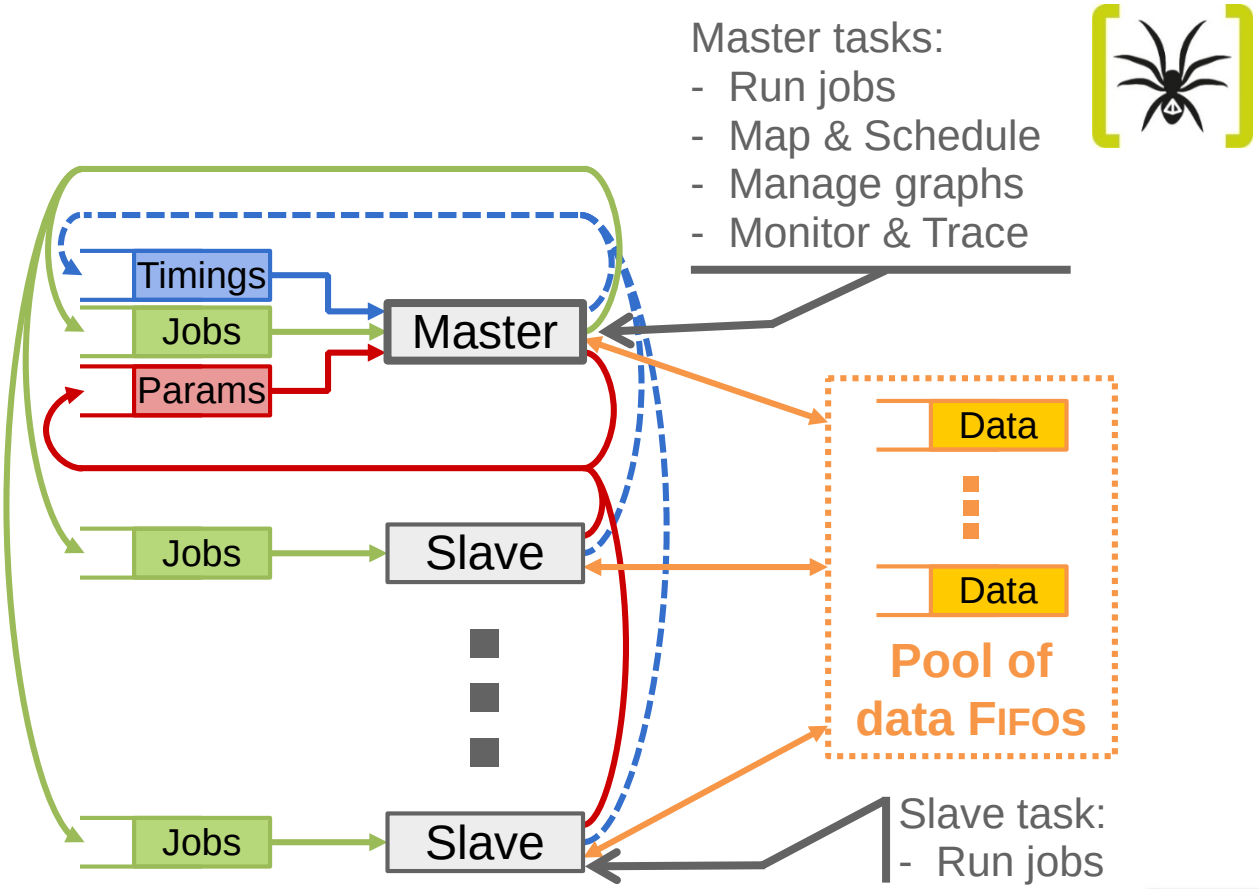


Code Generation

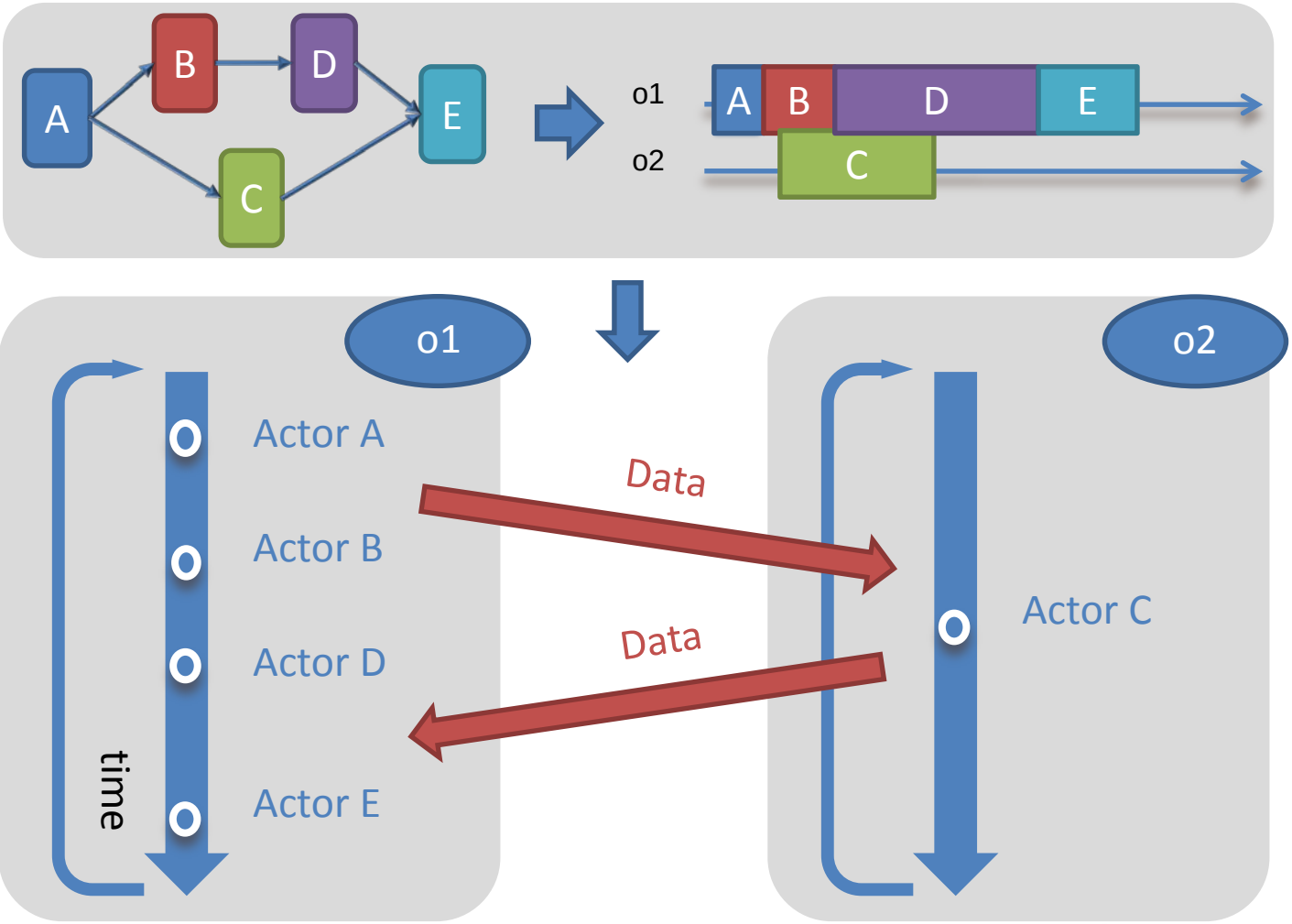


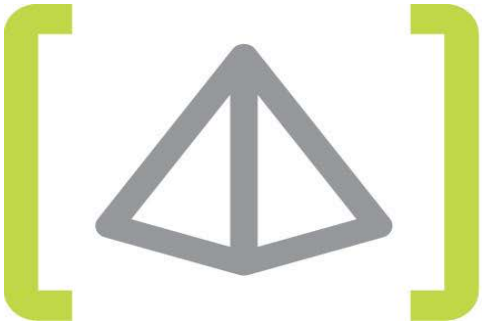
Mapping/Scheduling (dynamic parameters)

- SPIDER: Synchronous Parameterized and Interfaced Dataflow Embedded Runtime



Generation of self-timed multicore code (static parameters)





PREESM