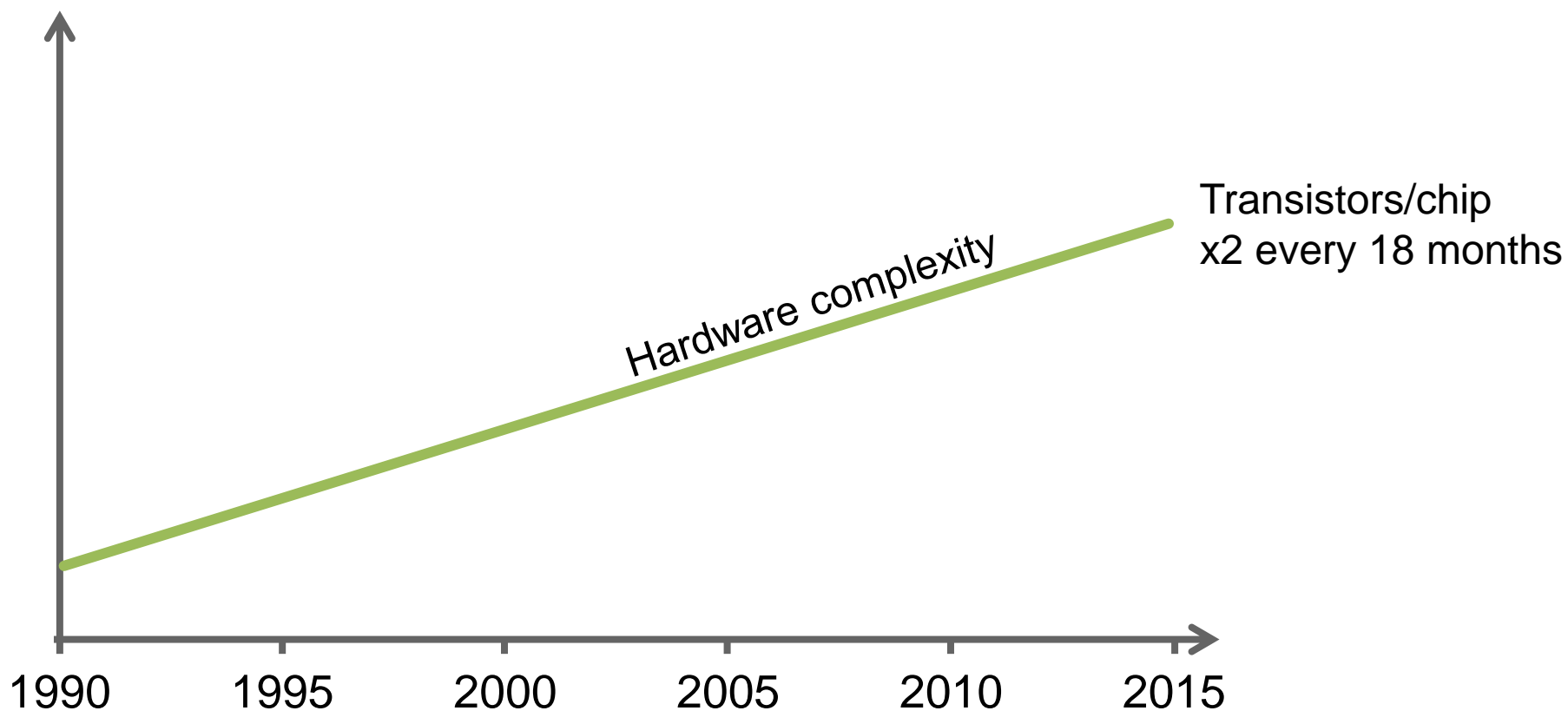


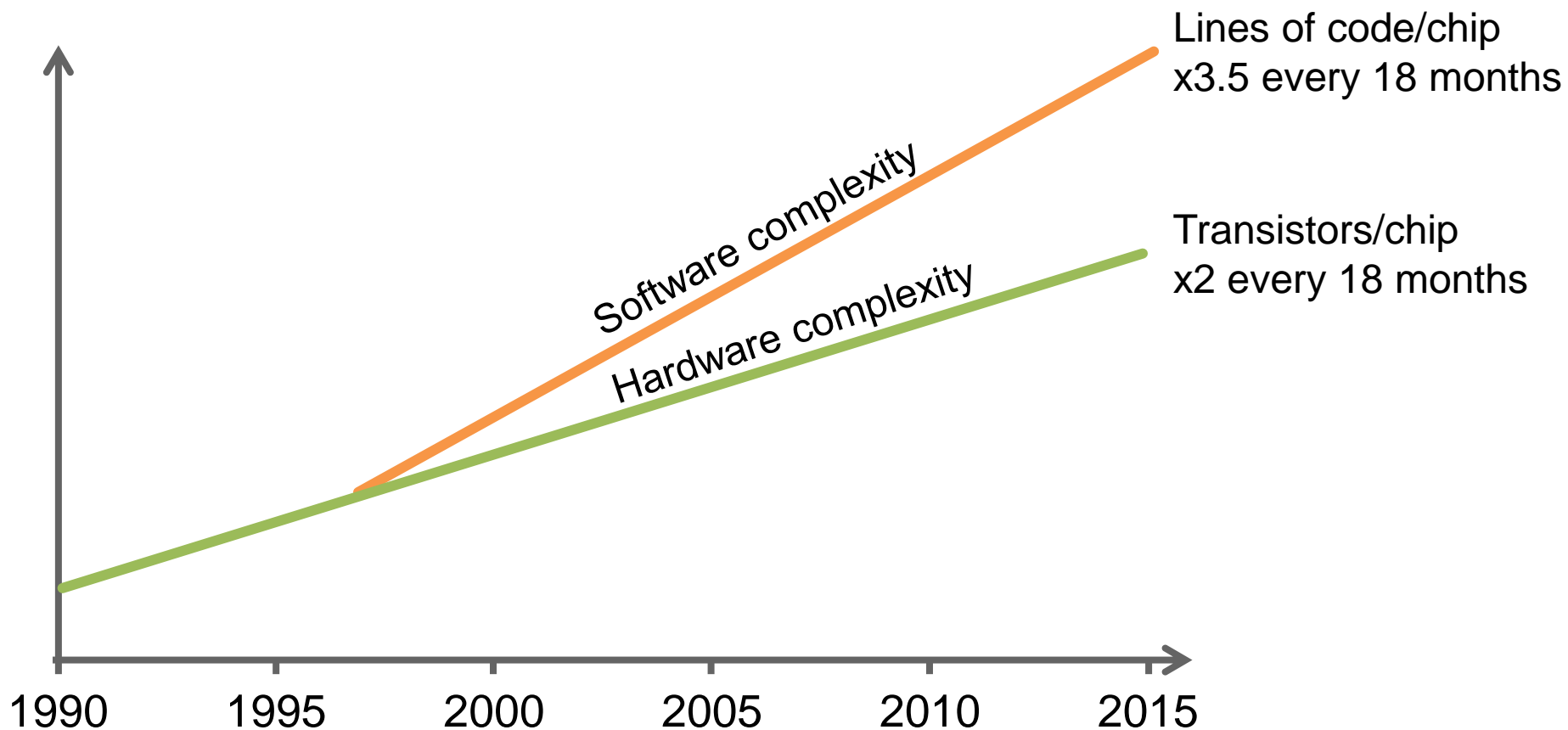
# **PREESM: A Dataflow-Based Rapid Prototyping Framework for Simplifying Multicore DSP Programming**

**Maxime Pelcat, Karol Desnos, Julien Heulot  
Clément Guy, Jean-François Nezan, Slaheddine Aridhi**

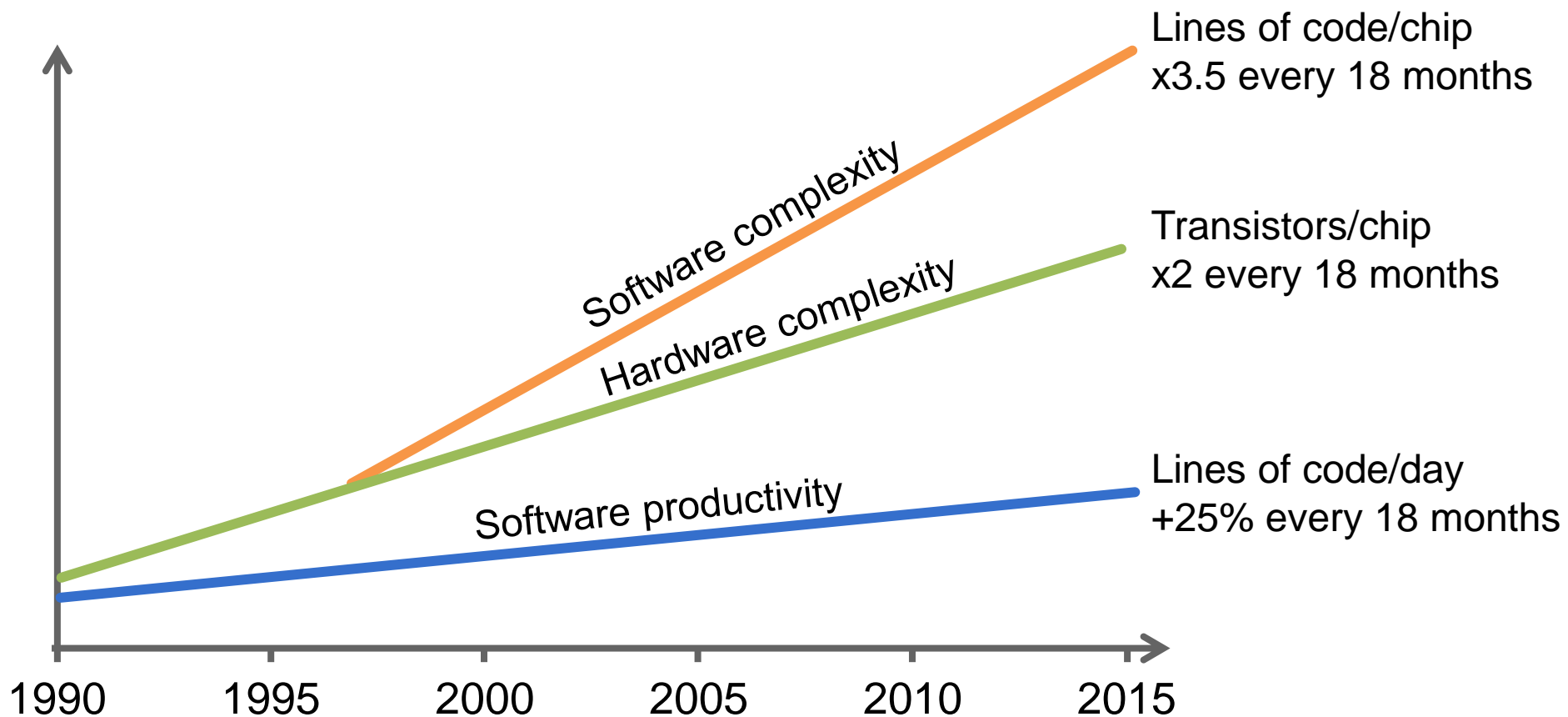
EDERC 2014 Conference, Milan, September 11<sup>th</sup>



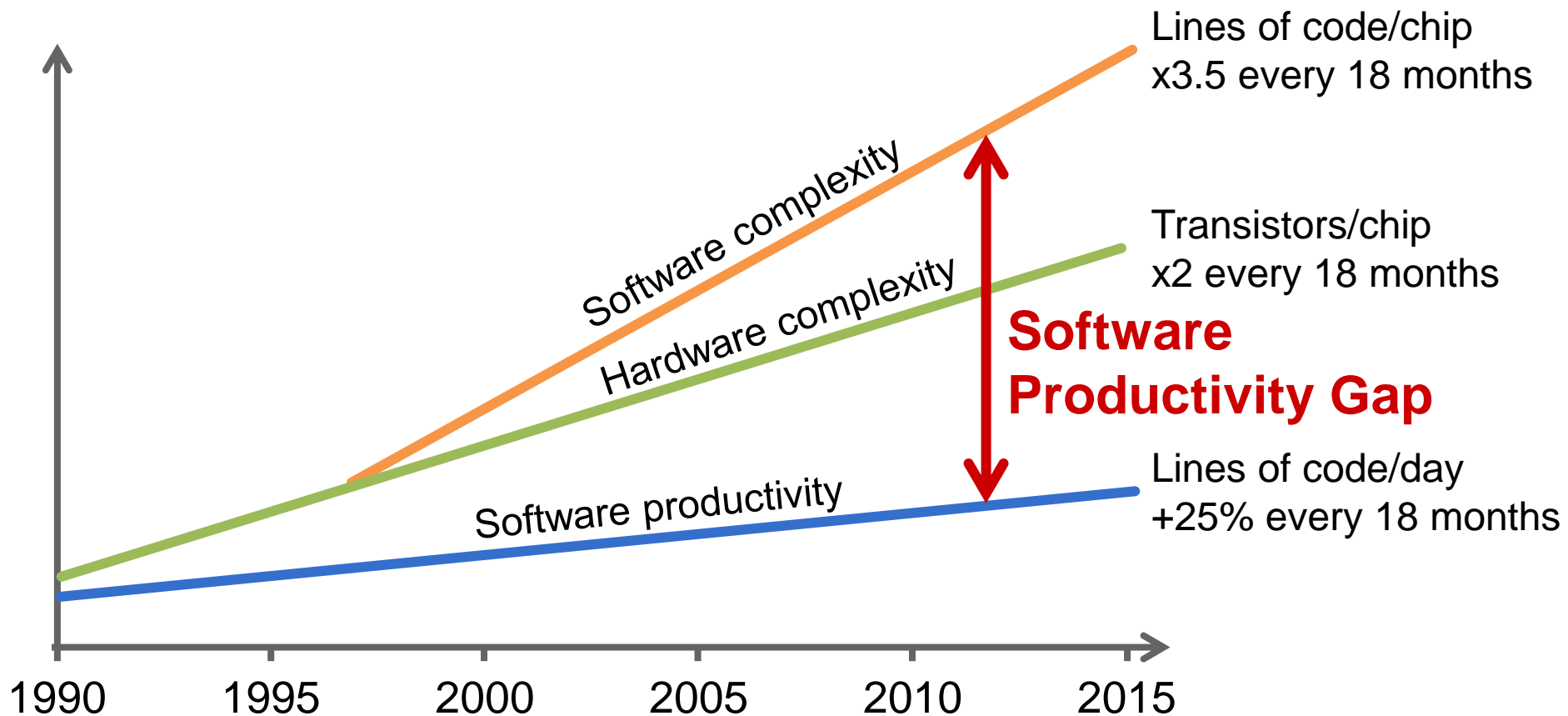
Source: “Hardware-dependent Software”, Ecker, et. al



Source: “Hardware-dependent Software”, Ecker, et. al

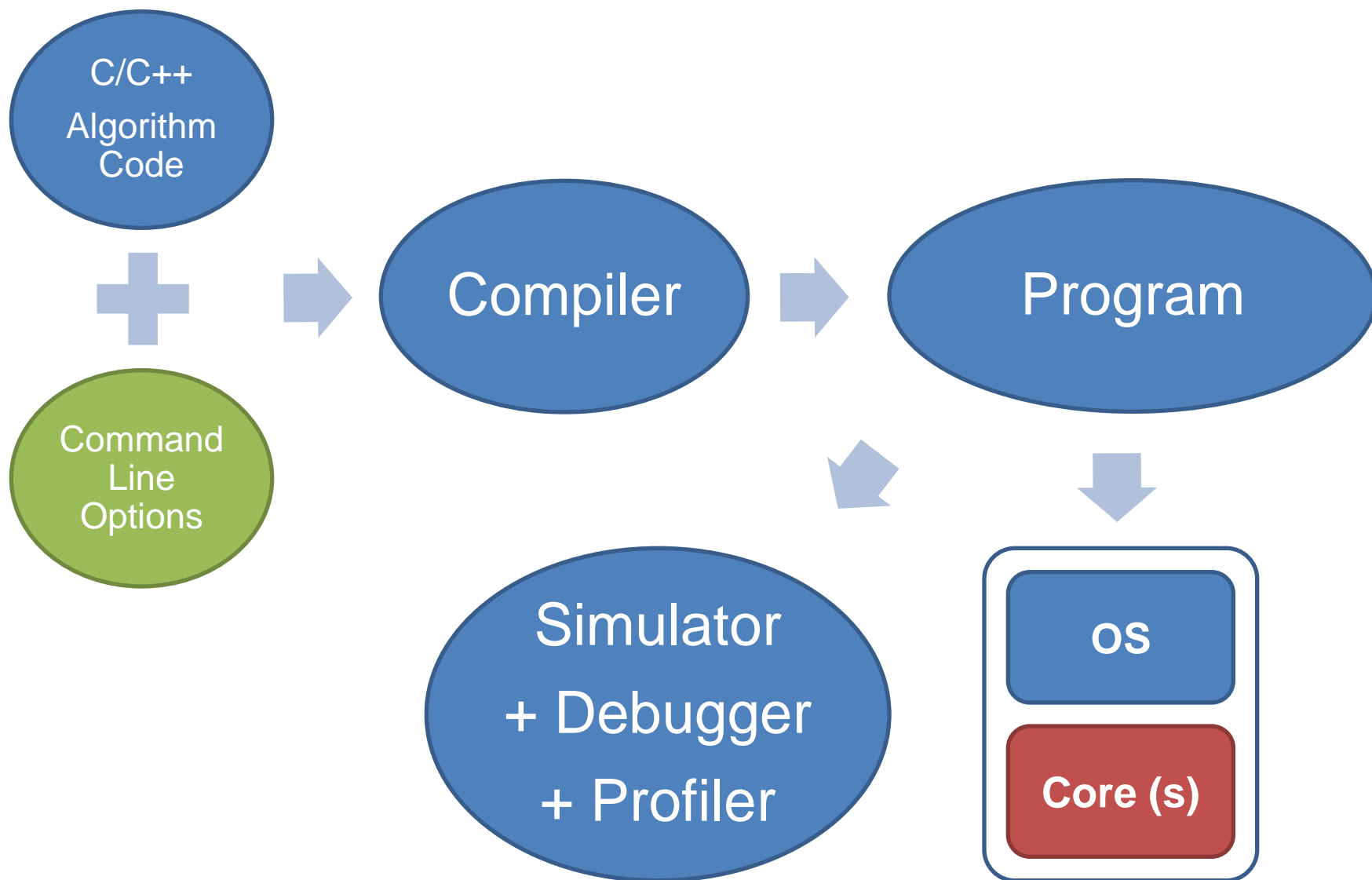


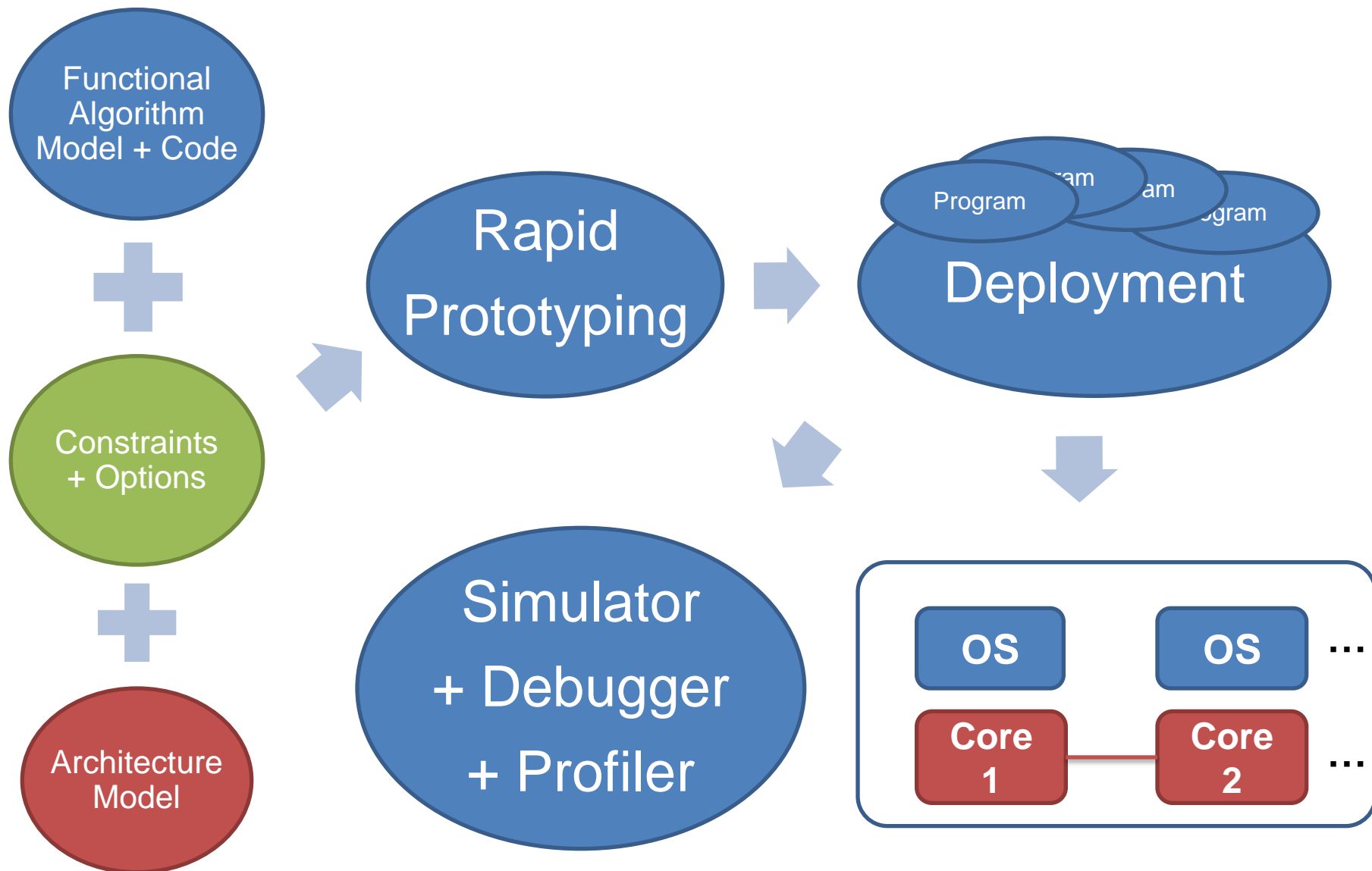
Source: “Hardware-dependent Software”, Ecker, et. al



Source: “Hardware-dependent Software”, Ecker, et. al

# Typical Single DSP Environment



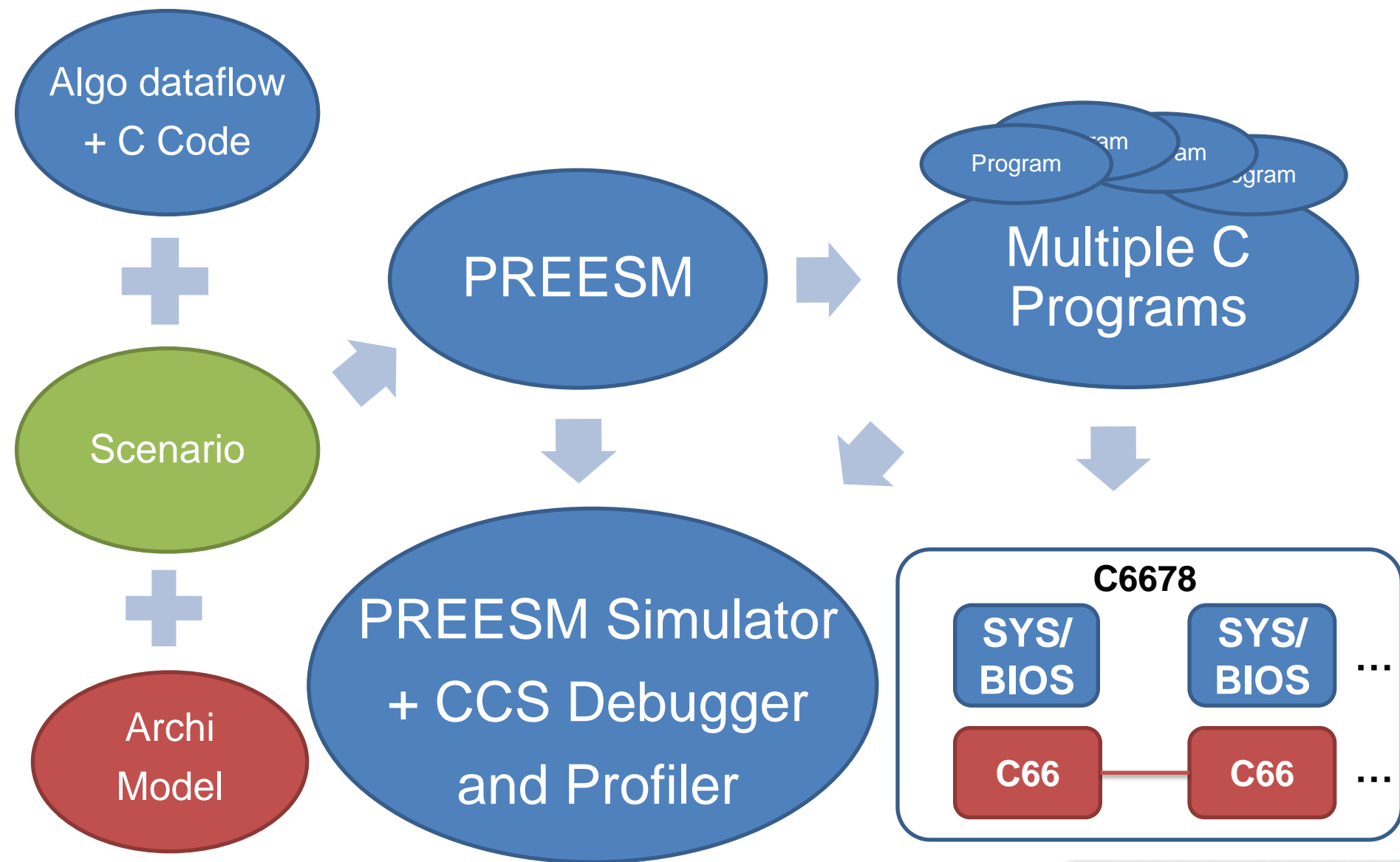


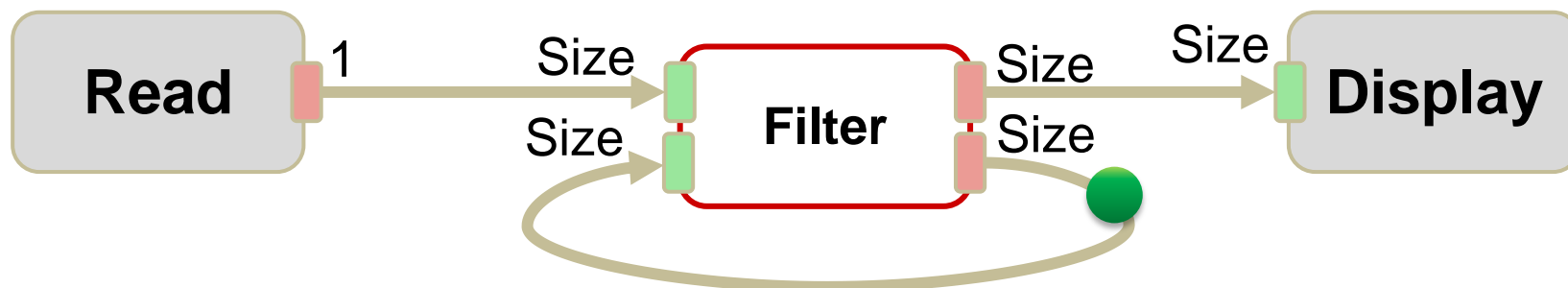
- In early design phases: Metrics
- Design parallel algorithms
  - Automatic mapping and scheduling
- Predictable time and memory
  - choose the right algorithm and hardware

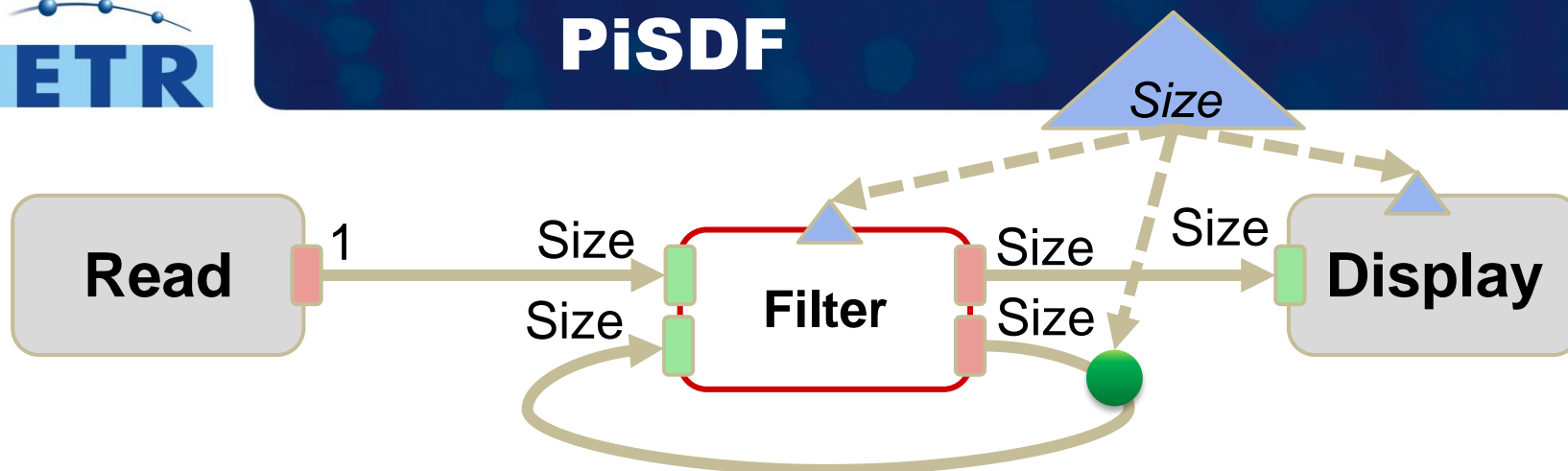


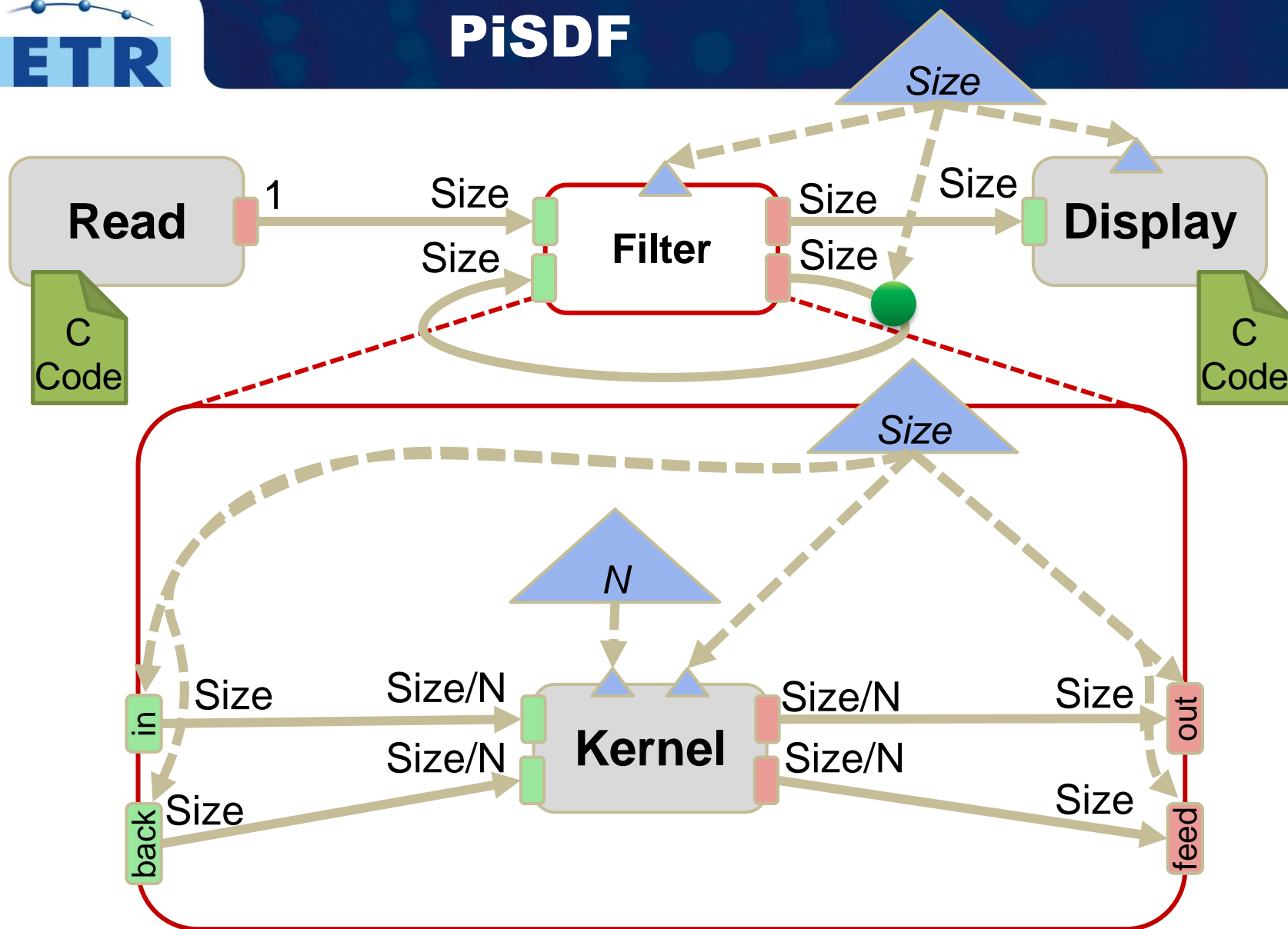
- In **late design phases**: Rapid Prototyping
- Automatic multi-core speedup
- Inter-core communication
- Guaranteed Deadlock-freeness

- For migration to a new hardware
  - Seamless porting to a new architecture
  - Legacy code reuseability
  - Portable performance
- Dataflow modelling can help









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



## PiSDF MoC is:

- ✓ Hierarchical & Compositional
- ✓ Statically parameterizable
- ✓ Dynamically reconfigurable

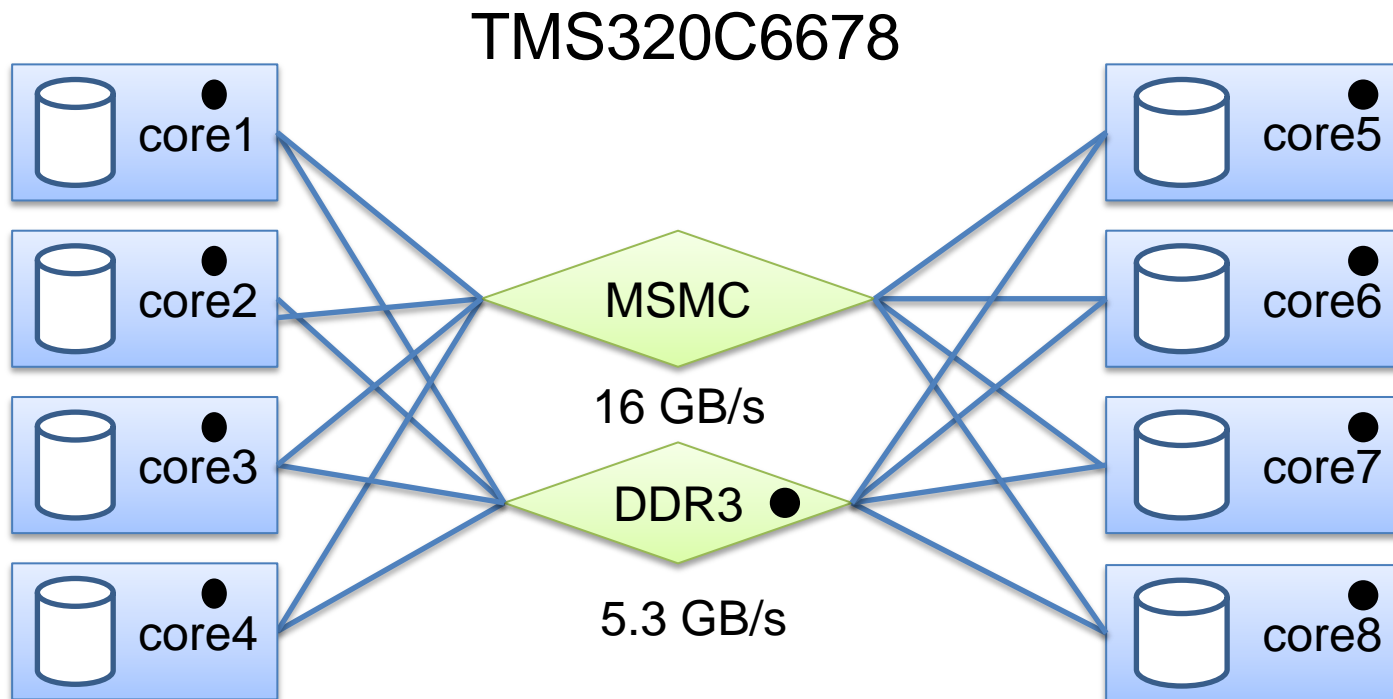
## PiSDF fosters:

- Predictability
- Parallelism
- Lightweight runtime overhead
- Developer-friendliness

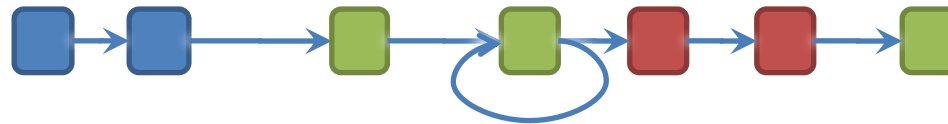
K. Desnos, M. Pelcat, J.-F. Nezan, S. S. Bhattacharyya, S. Aridhi “PiMM: Parameterized and Interfaced Dataflow Meta-Model for MPSoCs Runtime Reconfiguration”, SAMOS XIII



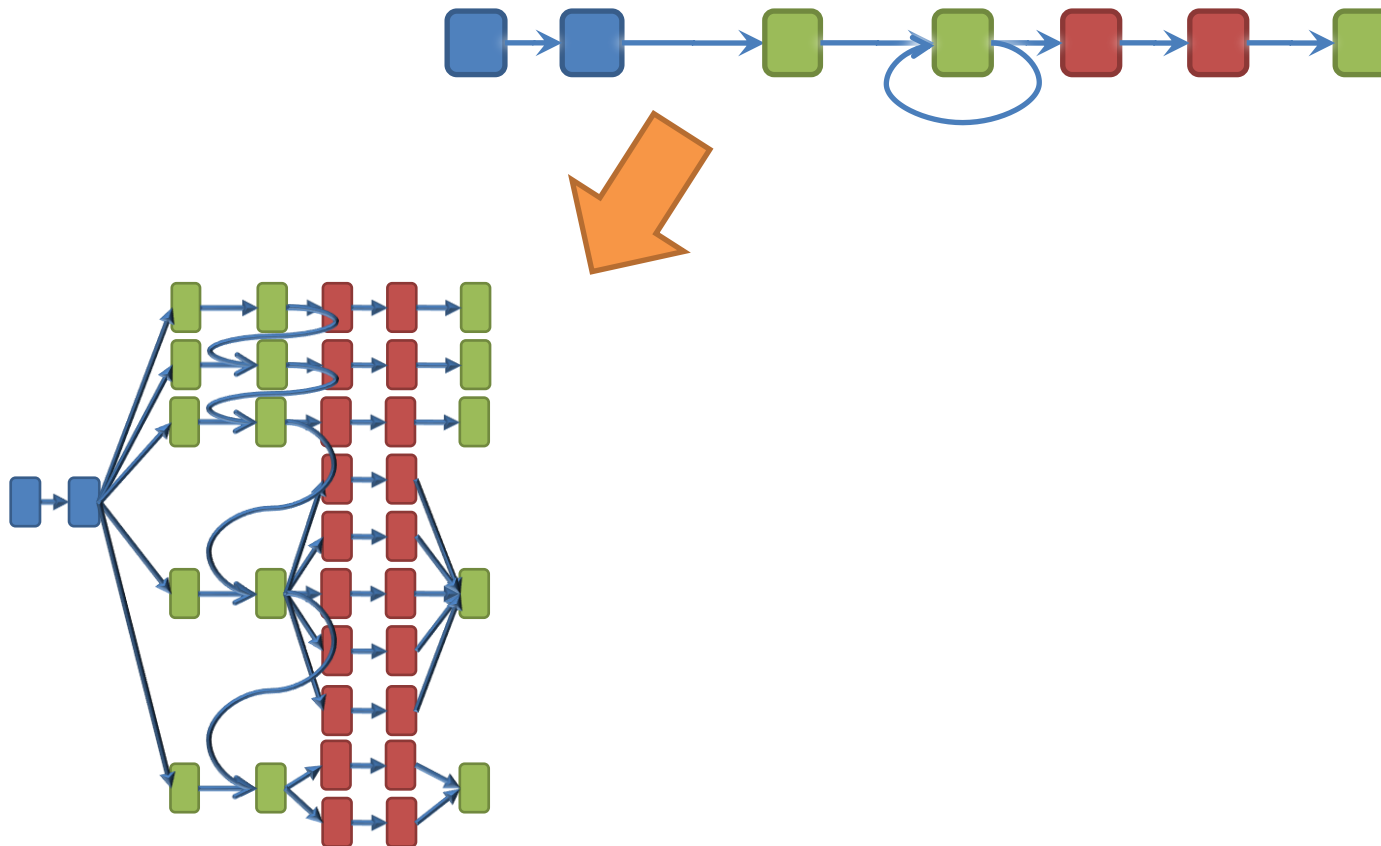
- Representing contentions as TDMA



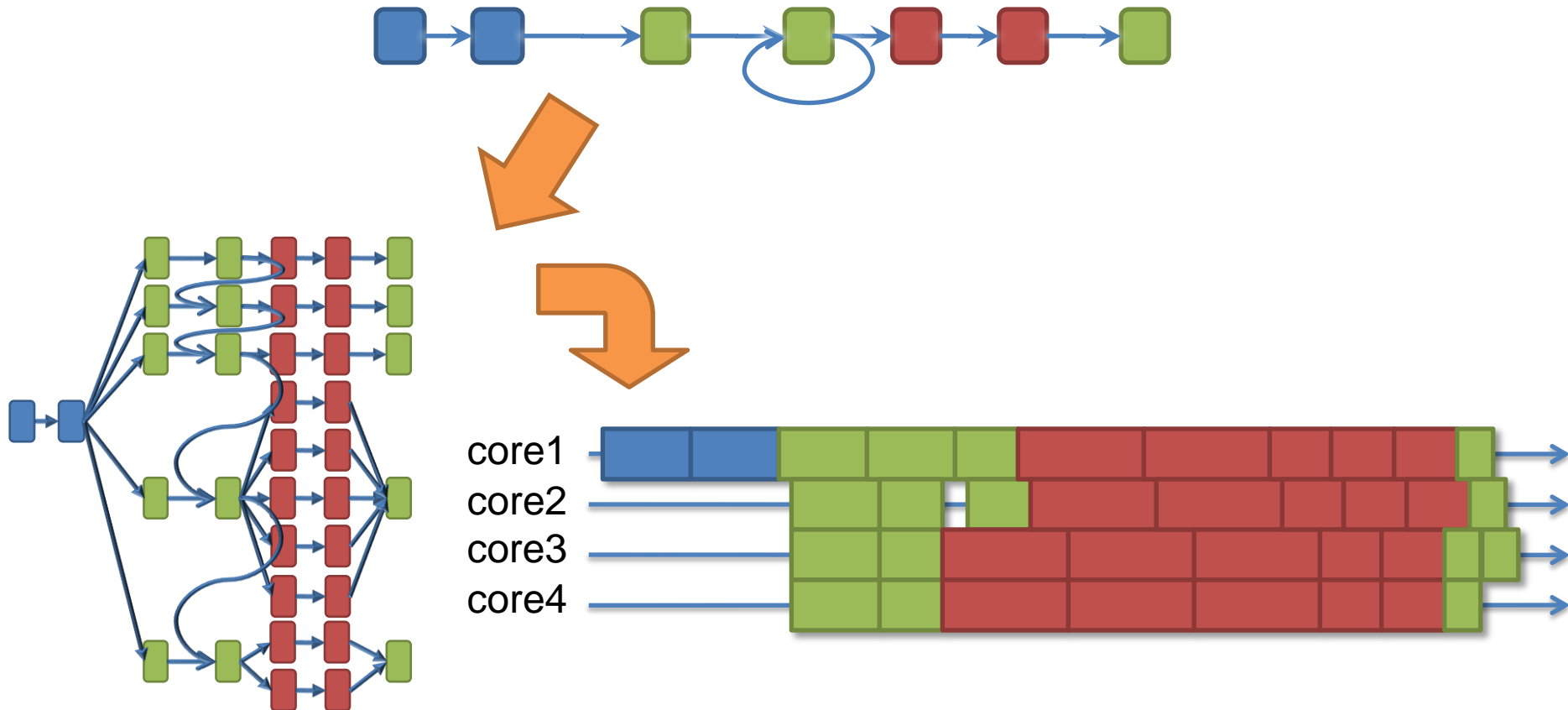
- Scheduling based on latency and load balancing



- Scheduling based on latency and load balancing



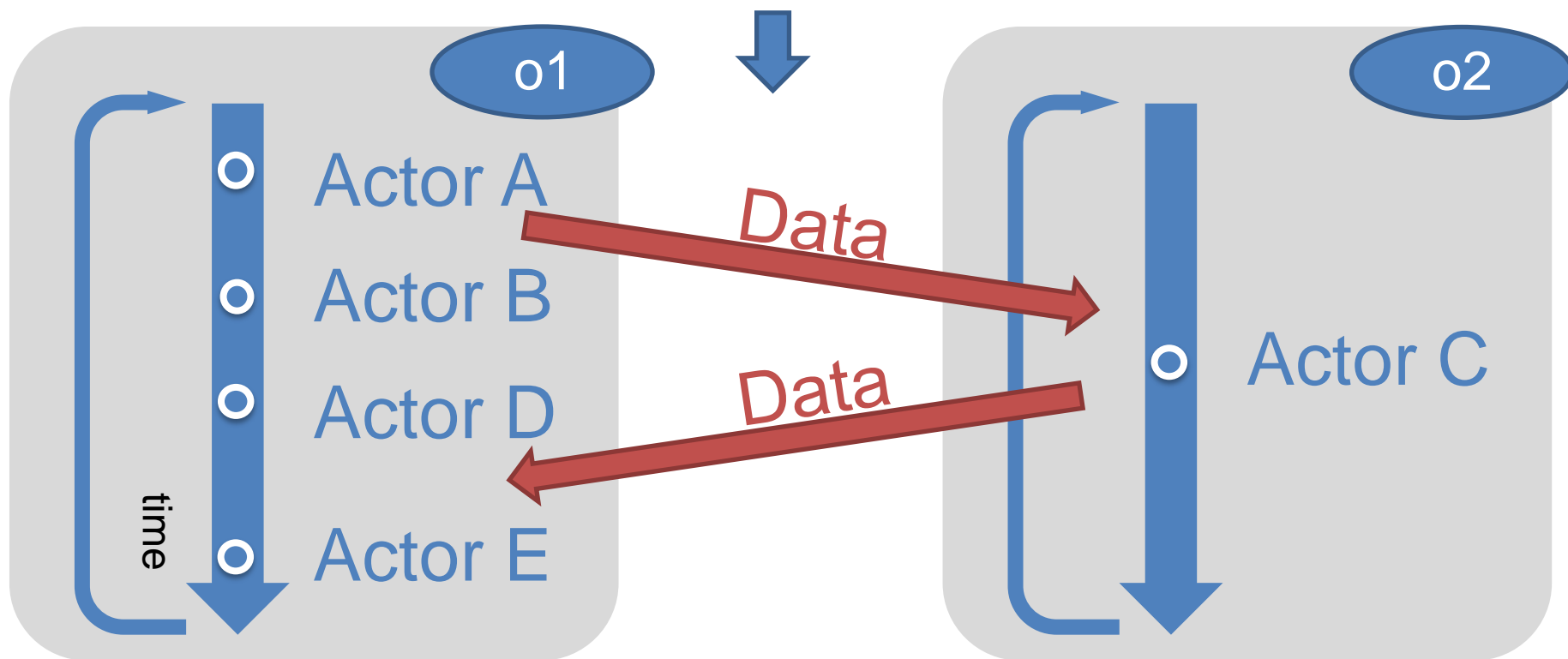
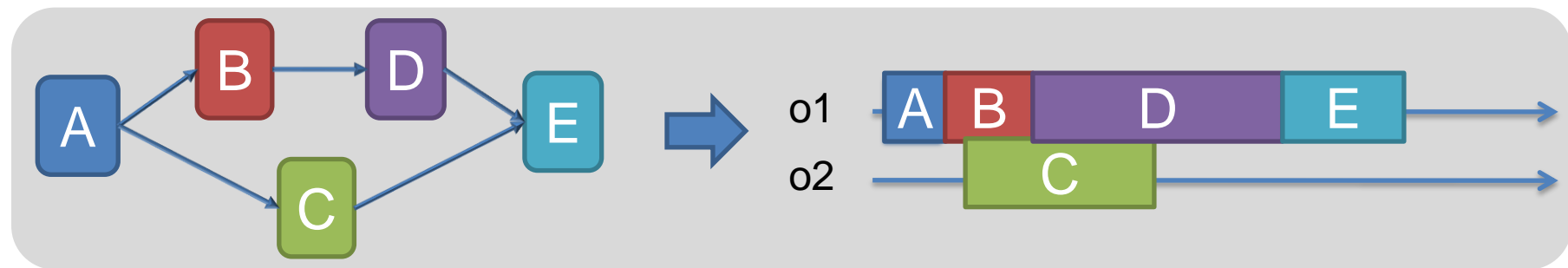
- Scheduling based on latency and load balancing



- **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





- 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>

- OpenMP, OpenEM
  - Adding Rapid Prototyping
- MAPS Compiler, Polycore Polymapper, SynDEx
  - Open-source code

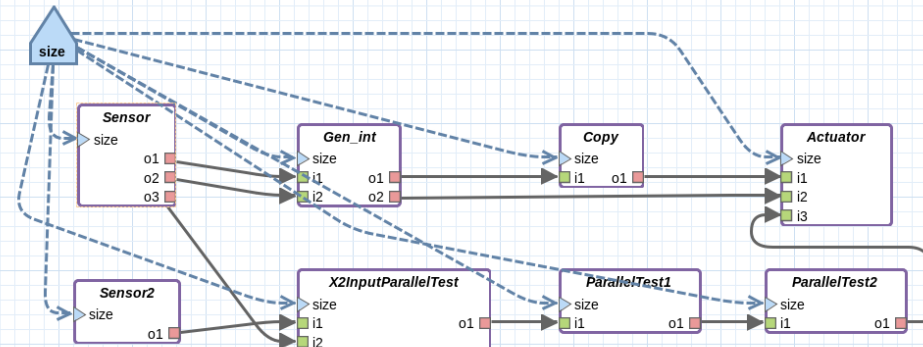


Java - org.ietr.preesm.tutorials.tutorial1/Code/generated/Core0.c - Eclipse Platform

Package Explorer

- org.ietr.preesm.tutorials.tutorial1 [preesm-apps master]
  - Algo
    - generated
      - TestCom.diagram
      - TestCom.pi
  - Archi
  - Code
    - generated
      - Core0.c
      - Core1.c
    - include
    - lib
    - src
      - CMakeCodeblock.bat
      - CMakeGCC.sh
      - CMakeLists.txt
      - CMakeVC2008.bat
  - DAG
  - Debug
  - Scenarios
  - Workflows
    - XTendCodegen.workflow
  - Readme.txt

TestCom



Palette

- Select
- Marquee
- Connections
  - Fifo
  - Dependency
- Objects
  - Actor
  - Parameter
  - Config. Input Interface
  - Config Output Interface
  - Data Input Interface
  - Data Output Interface
  - Broadcast Actor
  - Join Actor
  - Fork Actor
  - Round Buffer Actor

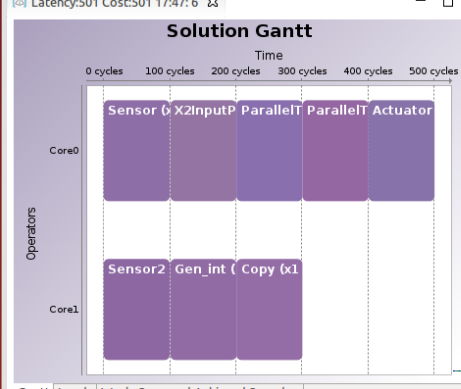
Problems Javadoc Declaration Console Properties Console Core0.c

```
// Initialisation(s)
sensor_init(o1_i1_4,o2_i2_0,o3_i2_0,100/*size*/); // Sensor
sem_init(&sem_0_SSRE,0,0/*init_val*/); // Sensor > Gen_int
circularg(o1_i1_3,o1_i1_5,100/*size*/); // X2InputParallelTest
parallel_init(o1_i1_5,o1_i1_2,100/*size*/); // ParallelTest1
parallel_init(o1_i1_2,o1_i1_3,0,100/*size*/); // ParallelTest2
actuator_init(o1_i1_0,o2_i2_1,o1_i1_3,0,100/*size*/); // Actuator

// Begin the execution loop
while(1){
  pthread_barrier_wait(&iter_barrier);
  if(stopThreads){
    pthread_exit(NULL);
  }
  sensor(o1_i1_4,o2_i2_0,o3_i2_0,100/*size*/); // Sensor
  sendStart(&sem_0_SSRE/*ID*/); // Core0 > Core1: Sensor_Gen_int_0
  sendEnd(); // Core0 > Core1: Sensor_Gen_int_0
  receiveStart(); // Core1 > Core0: Sensor2_X2InputParallelTest_0
  receiveEnd(&sem_1_SSRE/*ID*/); // Core1 > Core0: Sensor2_X2InputParallelTest_0
  receiveStart(); // Core0 > Core0: Gen_int_Actuator_0
  receiveEnd(&sem_2_SSRE/*ID*/); // Core1 > Core0: Gen_int_Actuator_0
  parallel2(o1_i1_3,o2_i2_0,o1_i1_5,100/*size*/); // X2InputParallelTest
  receiveStart(); // Core1 > Core0: Copy_Actuator_0
  receiveEnd(&sem_3_SSRE/*ID*/); // Core1 > Core0: Copy_Actuator_0
  parallel(o1_i1_5,o1_i1_2,100/*size*/); // ParallelTest1
  parallel(o1_i1_2,o1_i1_3,0,100/*size*/); // ParallelTest2
  actuator(o1_i1_0,o2_i2_1,o1_i1_3,0,100/*size*/); // Actuator
}
```

Latency:501 Cost:501 17:47: 6

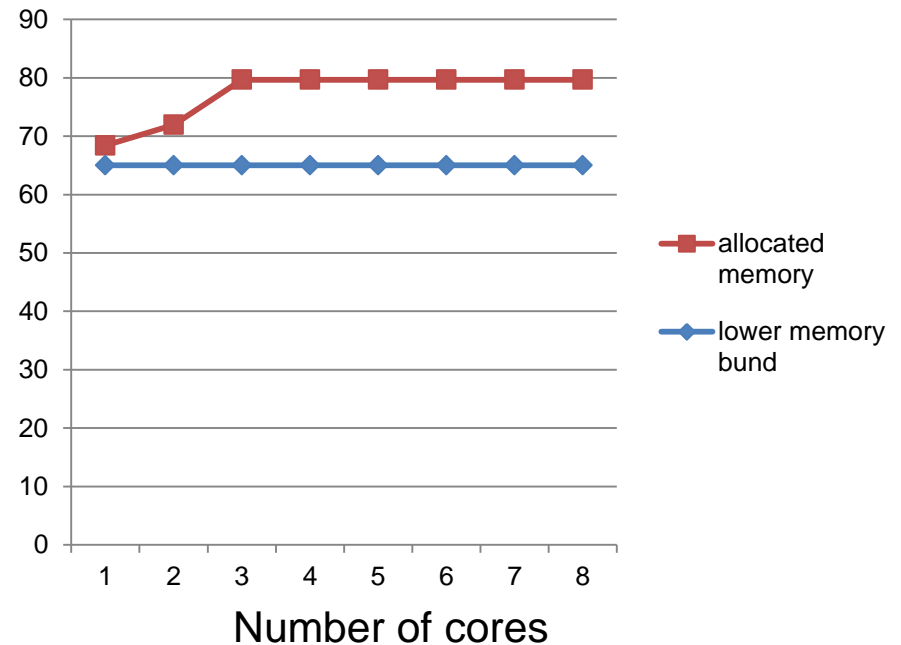
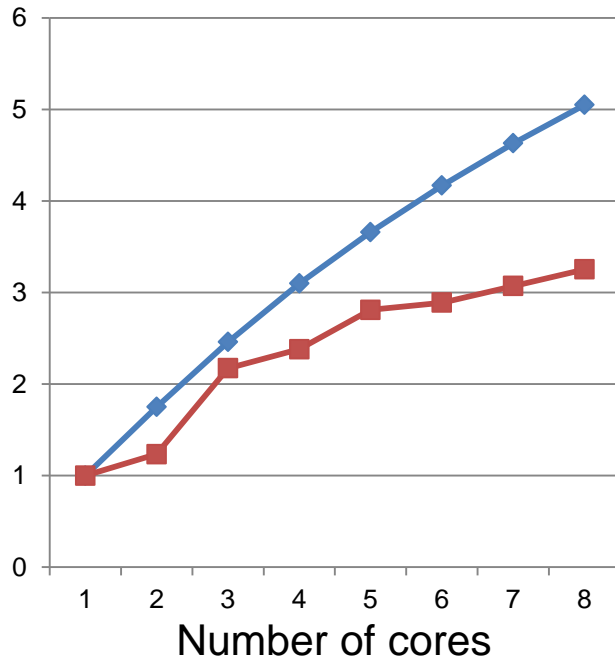
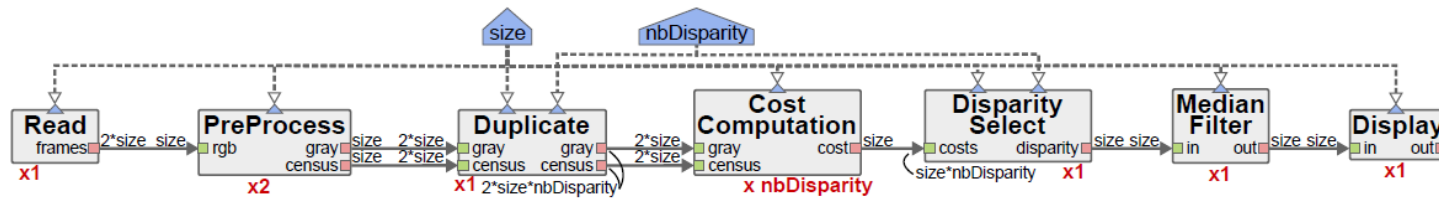
**Solution Gantt**



Gantt Loads Work, Span and Achieved Speedup

Writable Smart Insert 2:1

# Some Results on Stereo Matching



- Reduce Software Productivity Gap
  - Design space exploration
  - Rapid Prototyping
  - Extract coarse grain parallelism
  - Portable performance

**PREESM → Dataflow modelling can help!**

- Good decisions necessitate extensive information on both computation and data flow

# Thanks!

M. Pelcat, K. Desnos, J. Heulot, C. Guy, J.-F. Nezan, S. Aridhi, "**PREESM: A Dataflow-Based Rapid Prototyping Framework for Simplifying Multicore DSP Programming**" EDERC, 2014.

**PREESM Tutorial** – 16:00 – 17:00 - Room: Oro Plenaria

M. Pelcat, S. Aridhi, J. Piat, J.-F. Nezan, "**Physical Layer Multicore Prototyping: A Dataflow-Based Approach for LTE eNodeB**".  
Springer, 2012.

