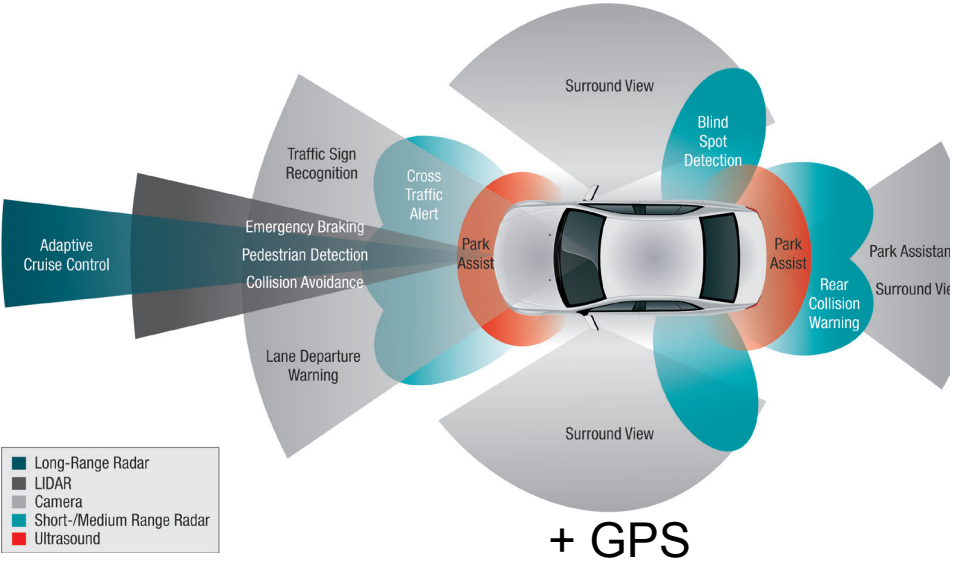
Manycore Embedded processors for portable, optimized and power efficient processing of methods for vision algorithms

Jean-François NEZAN

IETR, Rennes, France

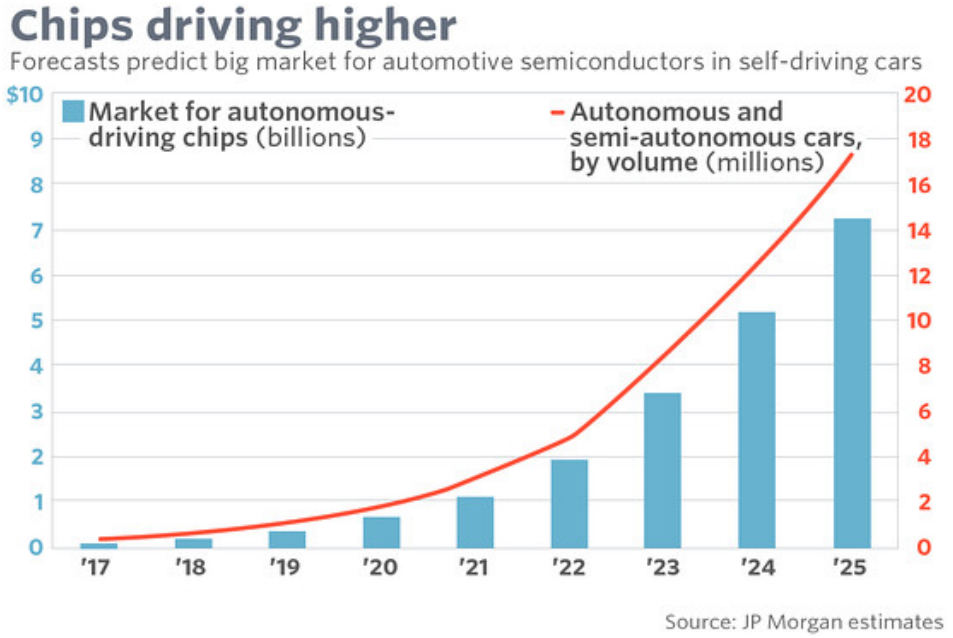
jnezan@insa-rennes.fr

- From Advanced Driver-Assistance Systems (ADAS) to Autonomous Cars
  - >16 Sensors and several Embedded systems
  - Performance, Energy Efficiency, Robustness, Reliability and Cost



+ GPS
+ Odometry

[1]



[2]

1. Introduction

- Low latency data transfers between sensors
  - Camera, Lidar, Radar : Ethernet and LVDS
- Low power / High performance computing
  - Computer vision, Object recognition (Machine Learning), Fusion of data, Image Processing …
- Dependability requirements (new ISO 26262 standard)
  - Hardware example : Detection, isolation of errors that occur in digital circuits, Transient electrical problems, soft errors, physical damage …
- Cyber-security
  - Root of Trust, Software IP protection

1. Introduction

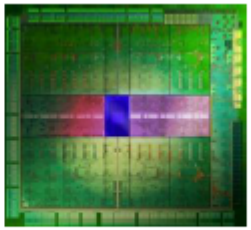
- Autonomous cars are « Safety-critical systems »
  - **Typical Hardware/Software Codesign Challenge + Safety**
  - Examples in Medecine, Nuclear engineering, Aviation, Spaceflight …
- Time-critical requirements
  - Time constraints associated with information manipulation
  - Execution time determinism, predictability, composability
  - Certification of worst-case response time by static analysis
- How to select the best computer vision algorithm in that context ?

1. Introduction

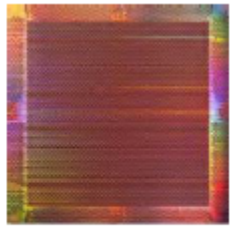
- Field-Programmable Gate Arrays (FPGA)
  - Most effective on bit-level computations
  - Require HDL programming
- Digital Signal Processors (DSP)
  - Most effective on fixed-point arithmetic
  - Require low-level programming
- Graphics Processing Units (GPU)
  - Most effective on regular computations
  - Require CUDA or OpenCL programming
- Intel Many Integrated Core (MIC)
  - Require multicore programming + exploitation of SIMD instructions (AVX)

CPUs

DSPs

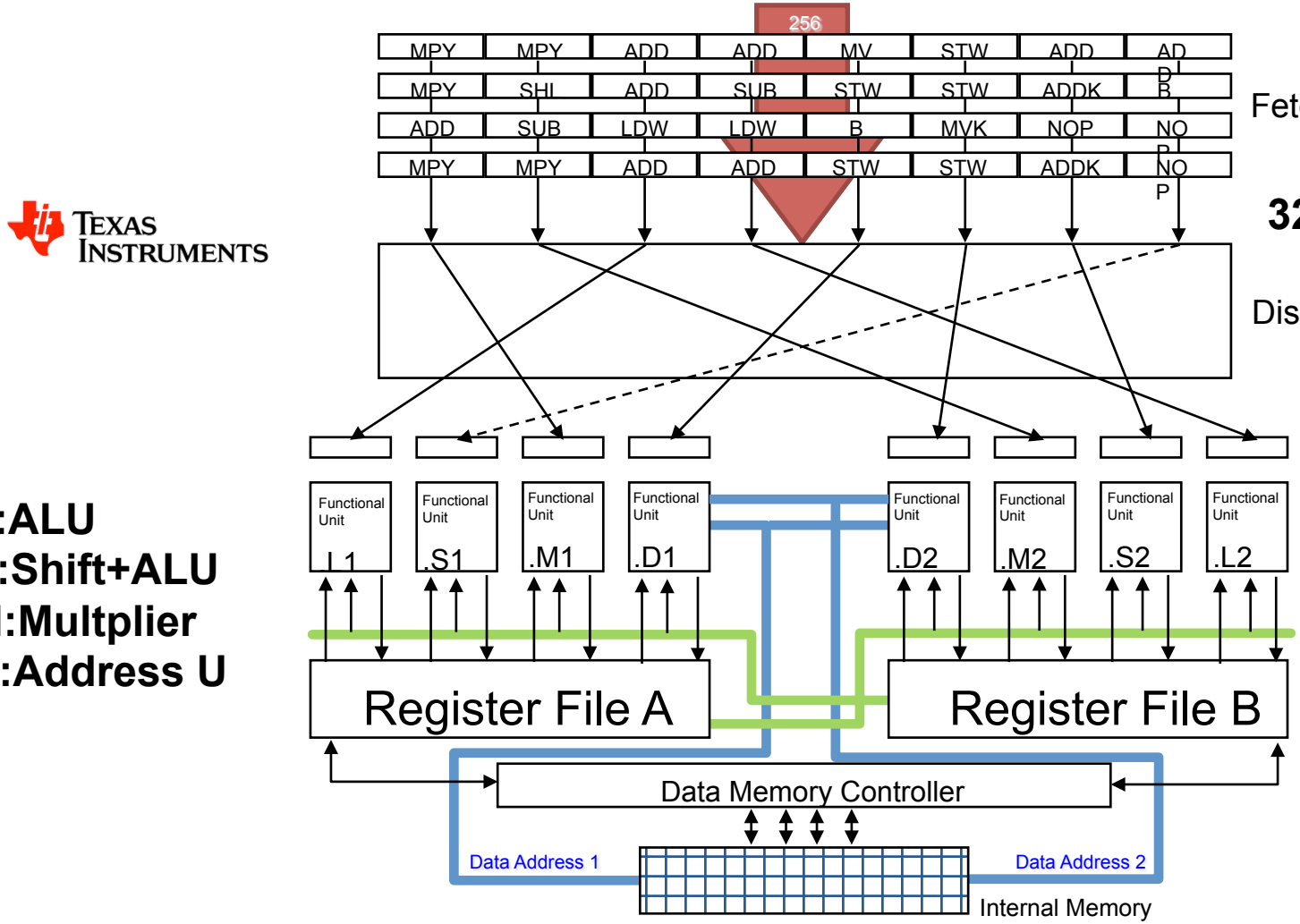


GPUs



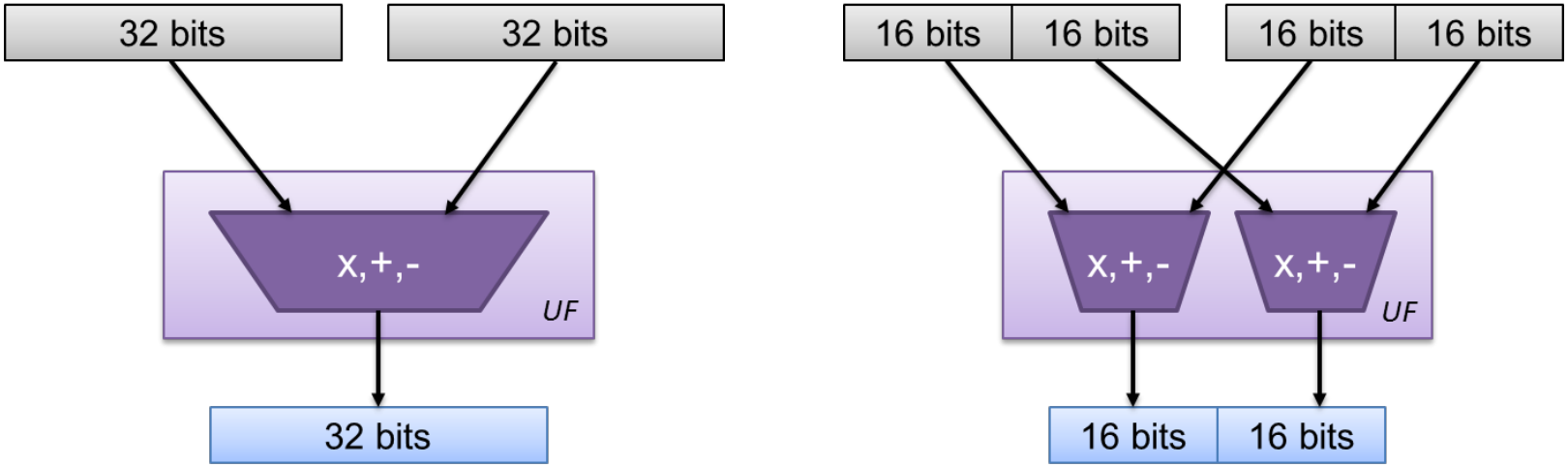
FPGAs

- DSP Cores : VLIW Architectures



L:ALU
S:Shift+ALU
M:Multplier
D:Address U

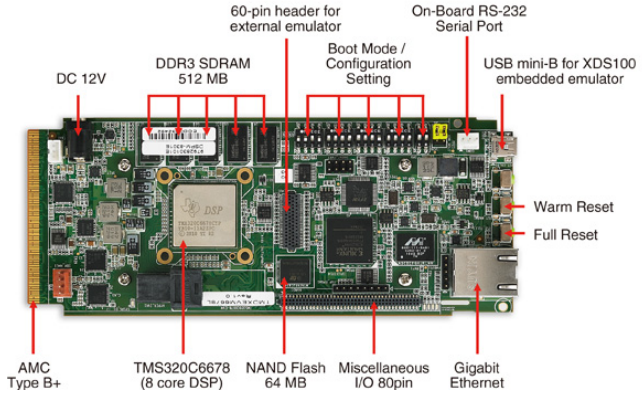
- DSP Cores : SIMD
  - Single Instruction Multiple Data
  - Splitting Functional Units into subparts
  - Example: **2** 16-bit additions on **1** 32-bit adder



- Impact on
  - Quality/accuracy of the result
  - Memory requirements
  - Computing time

Texas Instruments Keystone II

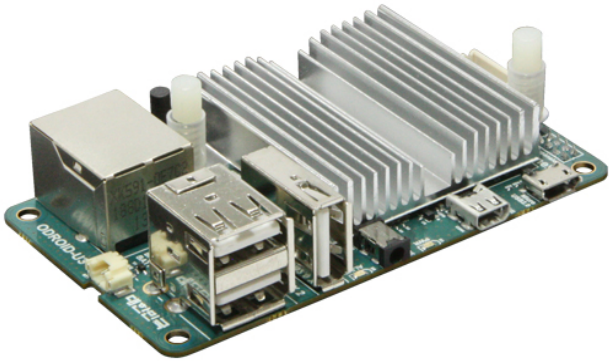
8x DSP cores @ 1.2 GHz – 4x ARM Cortex A15 @ 1.4 GHz – 6 MB memory + DDR – 650$ - 28nm – 16 Watts

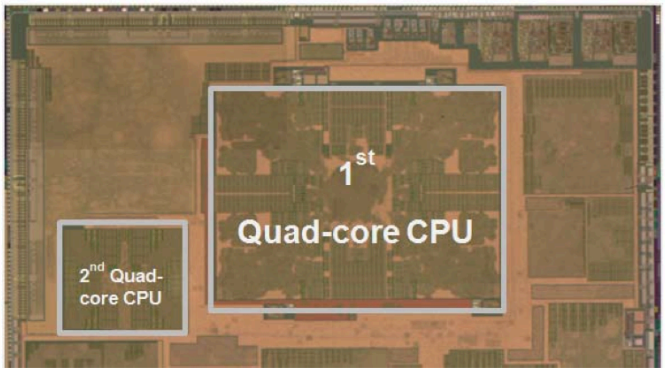


Die Photo

Odroid with Samsung Exynos 5

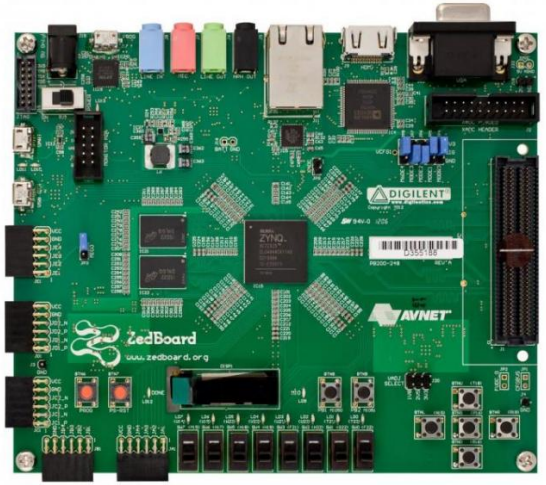
4x Cortex A15 @ 1.8 GHz– 4x Cortex A7 @ 1.2 GHz
+ PowerVR GPU – 50$ - 28 nm – 1Watt to 6 Watts





Zboard with Xilinx Zynq

2x ARM Cortex A9 @ 1 GHz – 444 Logic Cells
512 MB memory + DDR – 400$ - 28nm – 3 or 4 Watts ?

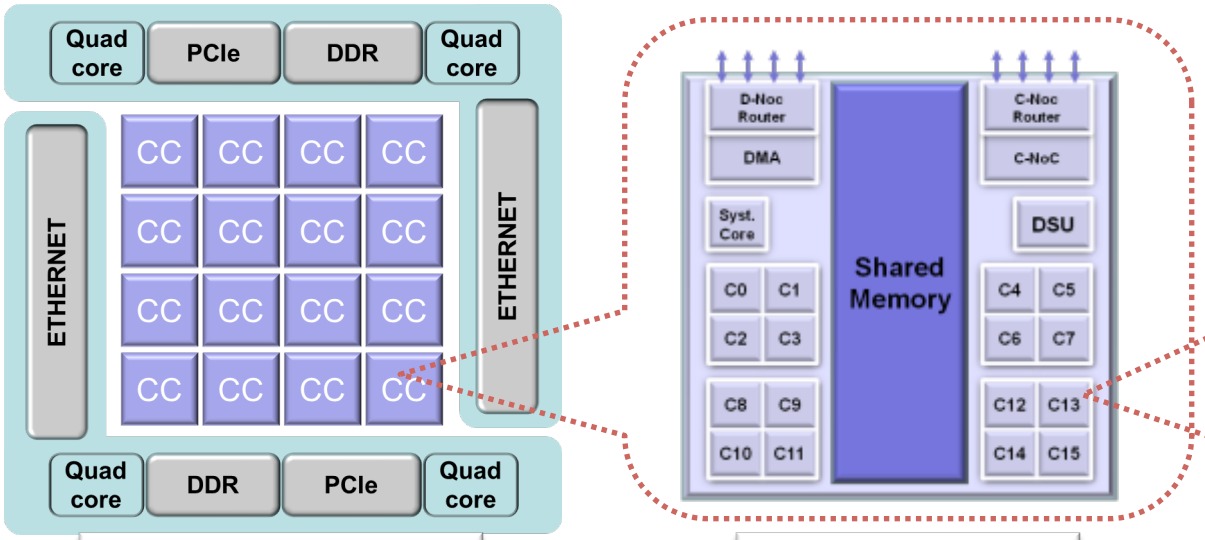


Nvidia GeForce GTX Titan X

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

KALRAY

MPPA



Manycore Processor

- 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

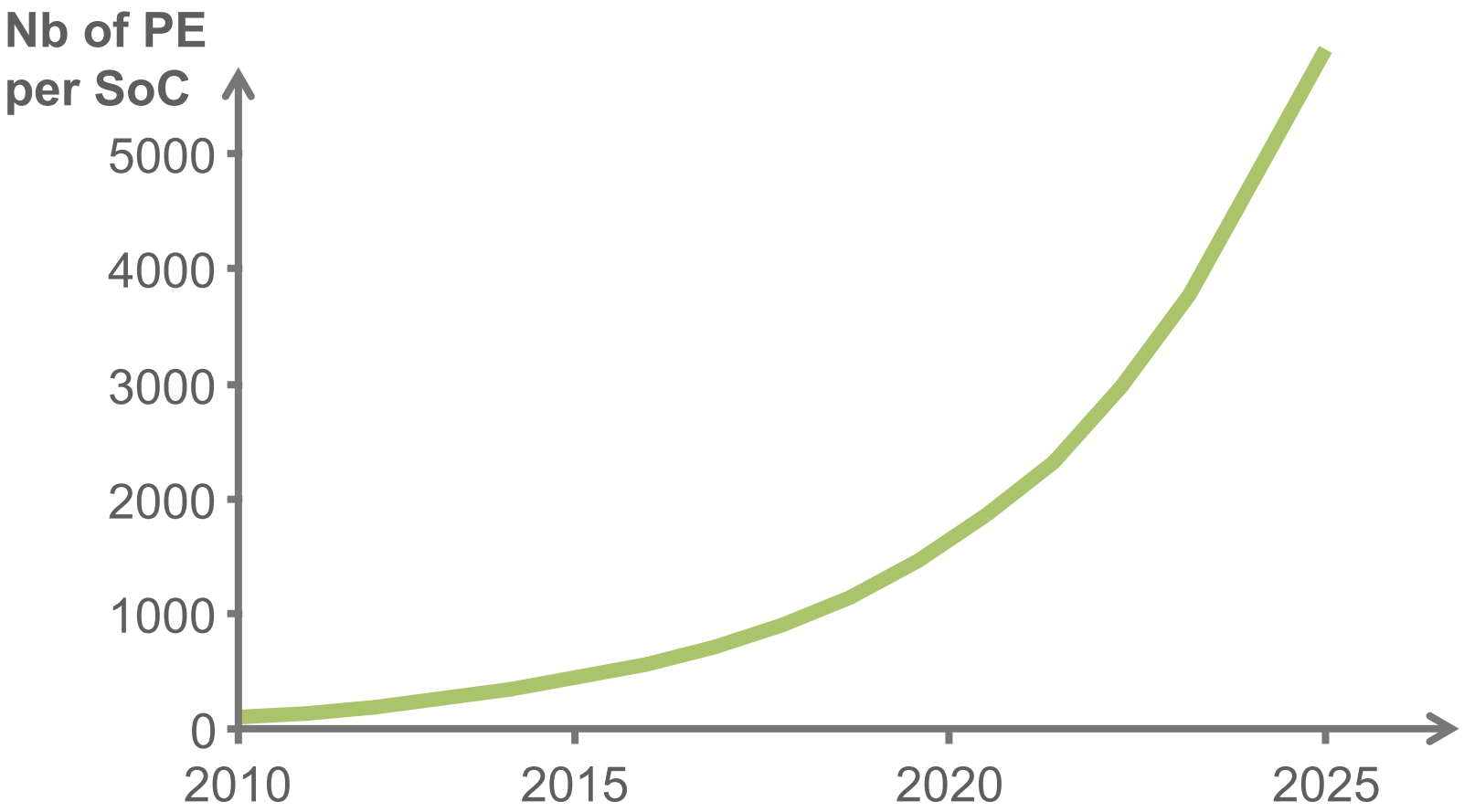
Compute Cluster

- 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

VLIW Core

- 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

- Hardware trend towards heterogeneous manycore [3]



Source: ITRS System Drivers 2011

1. Introduction

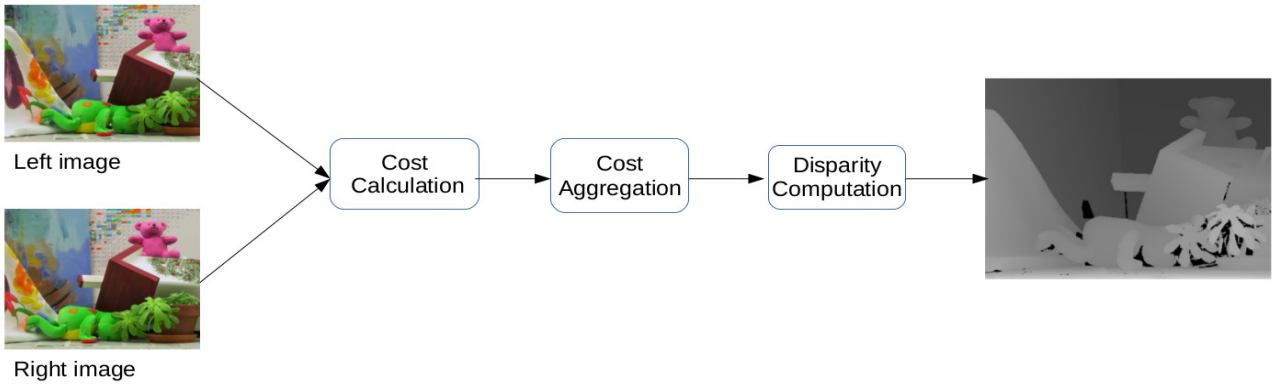
- Hardware trend towards heterogeneous manycore
  - Different cores
    - Control, Signal processing, Co-processors …
    - Number of cores
  - Different memories
    - Shared/distributed/caches/NoC …
    - Memory sizes
  - Different costs
    - New Hardware are designed for high volumes (Killer application)
    - For specific systems either
      - Expensive Specific hardware
      - Sub-optimal solutions based on low cost hardware
- How to optimize one algorithm on a given hardware ?
- What is the best hardware for one algorithm/application ?

Outline

1. Introduction
2. Stereomatching on Manycore Embedded Processors
3. Programming multicore embedded processors
4. Dataflow programming and Design Space Exploration

How to optimize one algorithm on a given hardware ?
What is the best hardware for one algorithm/application ?

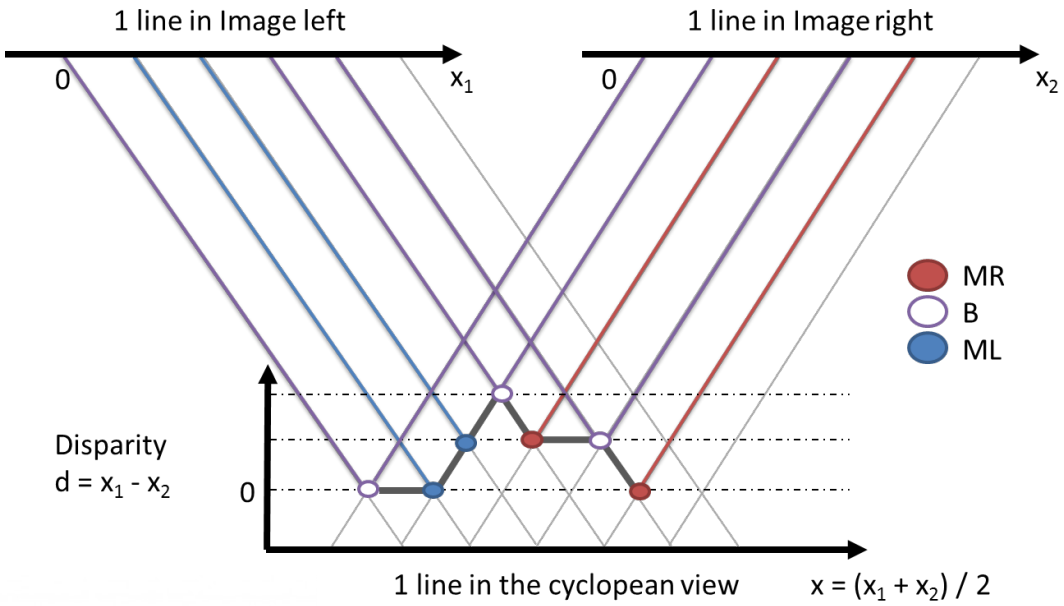
Middlebury Computer Vision Website



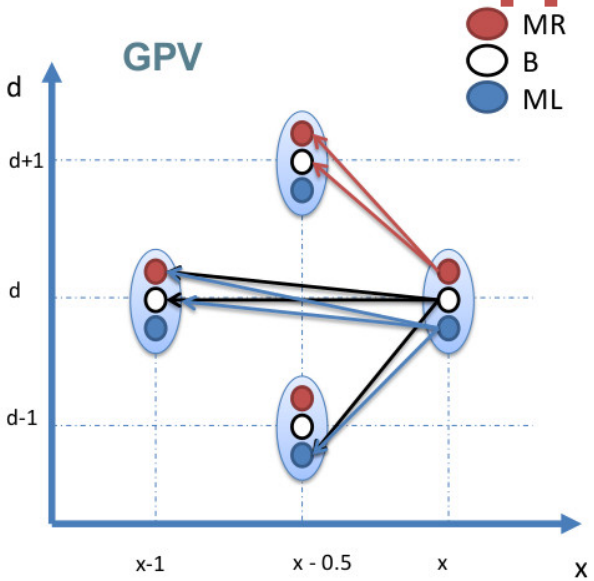
- Many algorithms
- Many Parameters
  - optimized for stereo pairs of the dataset
- Timing results
  - CPU/GPU oriented
- Based on an objective criteria (Average percent of bad pixels)
  - Need for subjective evaluation
  - Disparity map aligned with the left image (no cyclopean view)

- Optimization of BP-1D

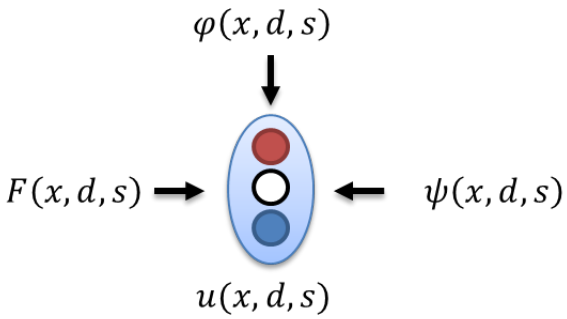
Graph of Profile Variants (GPV)



Possible transitions [4]

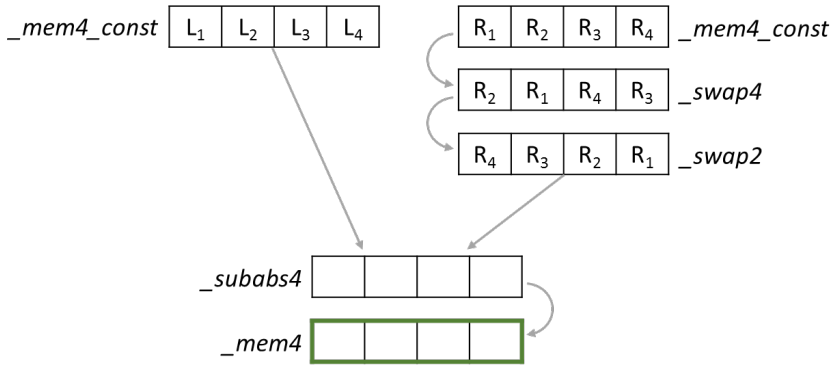


Messages for a node

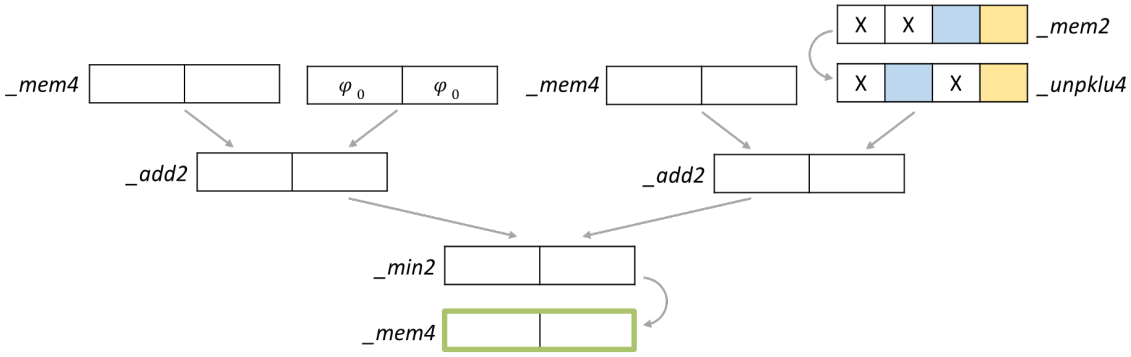


THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

- Optimization of BP-1D : single core code optimization (Keystone 1) [4]
  - Sum of Absolute Difference Cost Computation (SAD)
    - Work in grey scale input images (8 bits)
    - Use subabs4 SIMD instruction (4 absolute subtractions in 1 cycle)



  - Cost aggregation
    - Split the main loop in two sub-loops (disparity odd and even)
    - Compute 2 levels of disparity at the same time



[4]

- Optimization of BP-1D : results
  - Teddy dataset / 434x380 frame resolution / 18 disparity levels

[4]

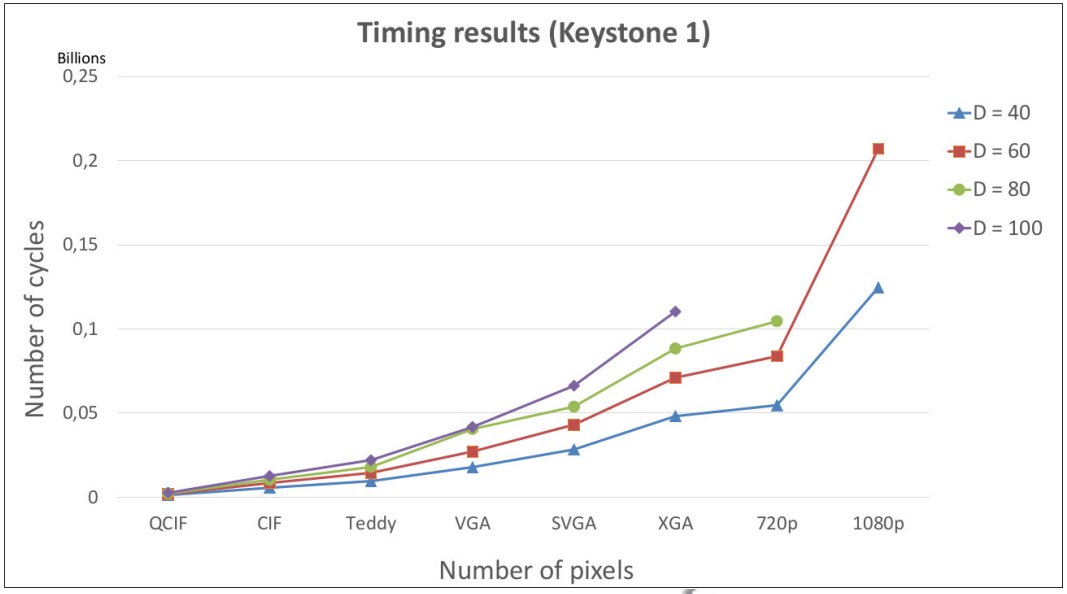
- Optimization of BP-1D : results
  - Teddy dataset / 434x380 frame resolution / 18 disparity levels

Monocore profiling

	Number of cycle in one core		
	Original version	Optimize version	**Speed-Up**
Cost computation	≈31M	≈3M	***8,13***
Forward pass of Cost aggregation	≈70M	≈44M	***1,58***
Backward pass of Cost aggregation + Disparity selection	≈154M	≈74M	***2,08***
Total	≈256M	≈118M	***2,17***

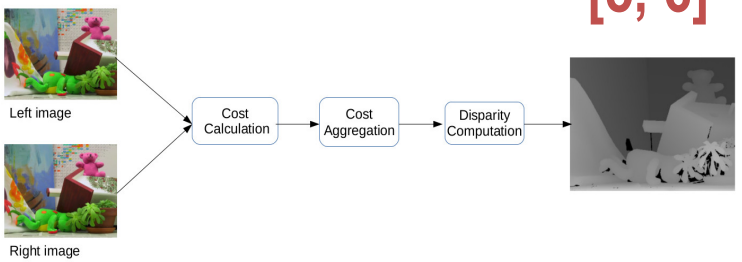
8-cores profiling

Speed-up factor of 8
245 fps for teddy dataset



[5, 6]

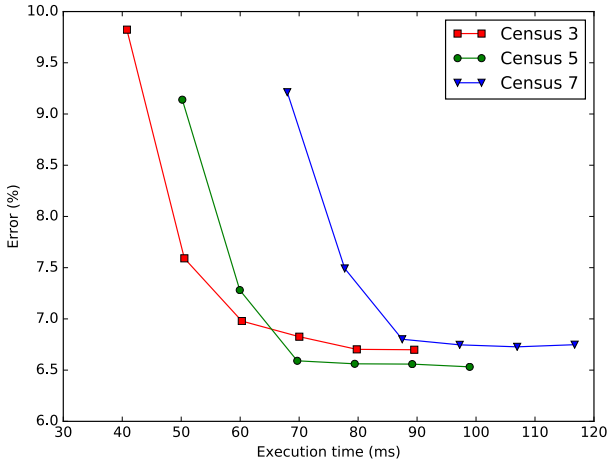
- Considered Algorithms :
  - Cost Calculation
    - Sum of Absolute Differences (SAD)
    - Census Cost
    - Mutual Information (MI)
  - Cost Aggregation
    - BFA (Bilateral Filtering Aggregation)
    - BP-1D (1 Dimension Belief Propagation)
    - SGM (Semi Global Matching)
  - Disparity Computation
    - Energy minimization (BP-1D Backward Pass)
    - WTA (Winner Takes All)
  - No Simulated Annealing (SA) and Graph Cut (GC)



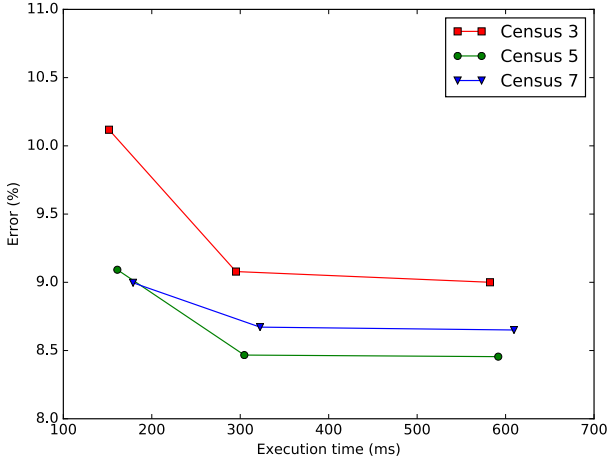
[5, 6]

- Impact of parameters on the monocore execution time
  - Census provides the best results
    - Sizes : 3x3, 5x5 and 7x7
  - BFA : Number of iteration [3 .. 8]
    - Distance of aggregation [1;NbIter^2]
  - SGM : Number of direction [2, 4, 8]

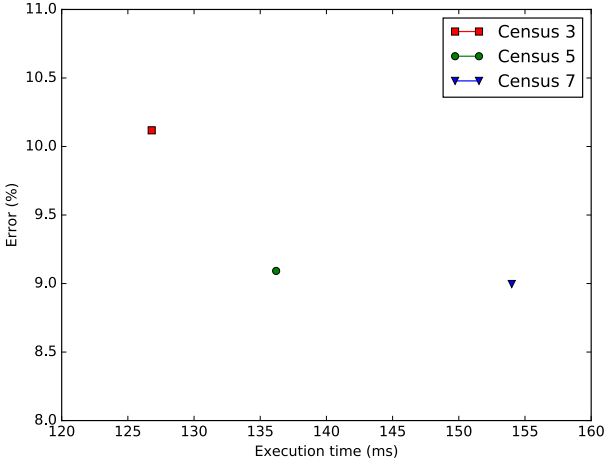
C1 : BFA + WTA



C2 : SGM

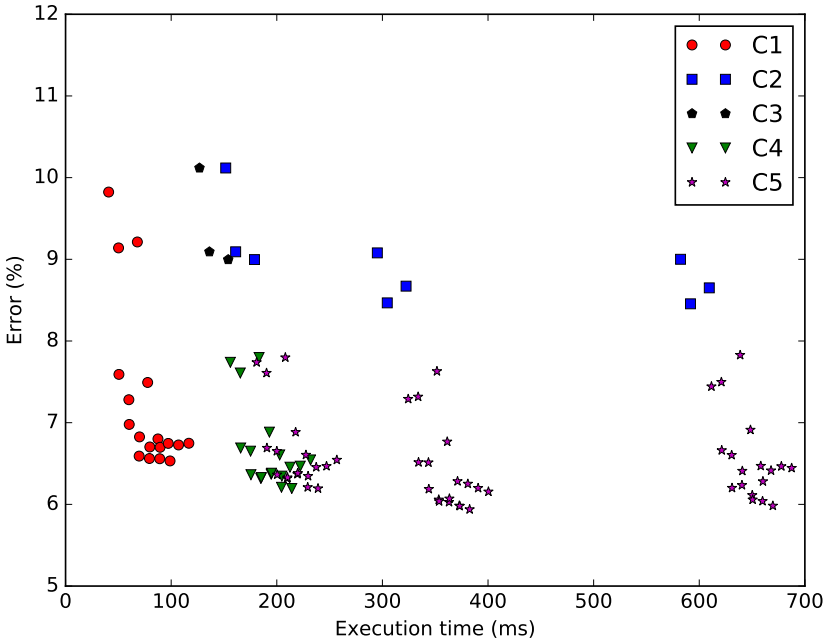


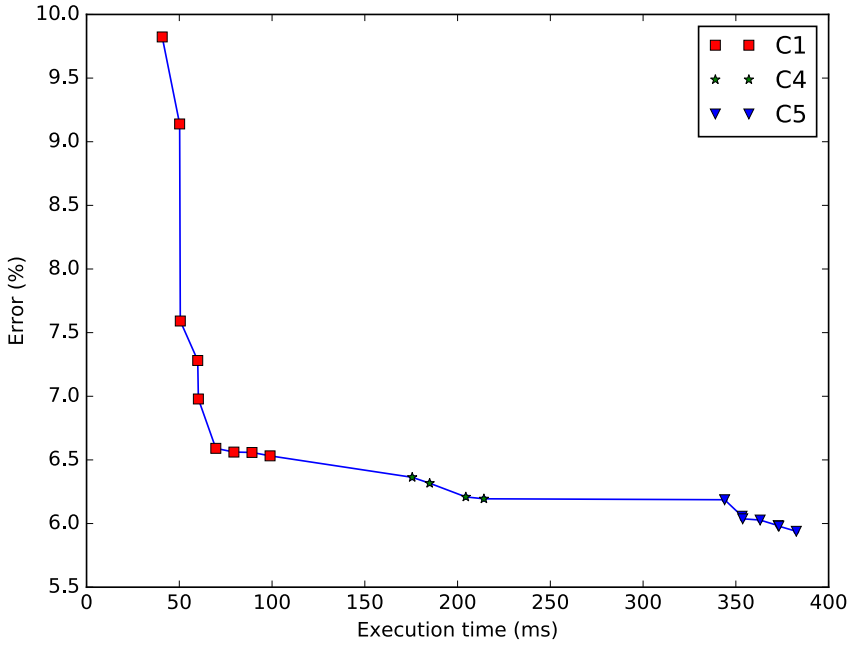
C3 : BP-1D



[5, 6]

- Impact of parameters on the monocore execution time
  - C1 : census + BFA + WTA
  - C2 : census + SGM
  - C3 : census + BP-1D
  - C4 : Census + BFA + BP1D
  - C5 : Census + BFA + SGM

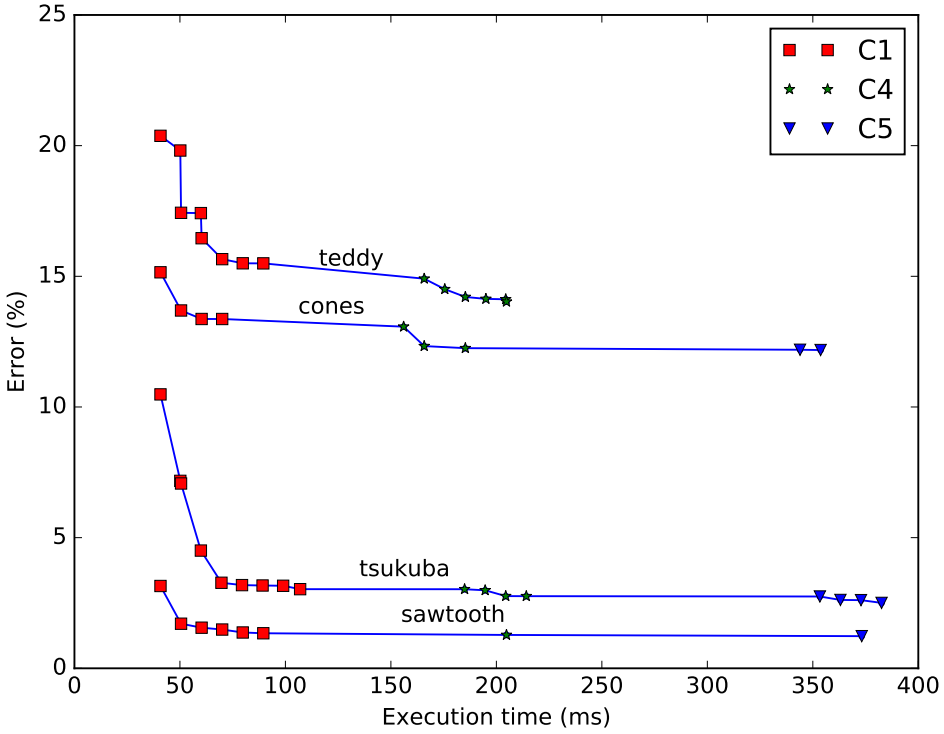




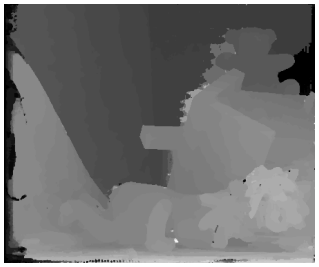
- Impact of input images
  - C1 : census + BFA + WTA
  - C2 : census + SGM
  - C3 : census + BP-1D
  - C4 : Census + BFA + BP1D
  - C5 : Census + BFA + SGM

[5, 6]

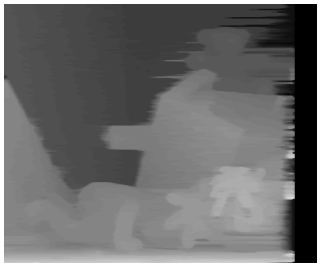
Name	Resolution	Number of disparities
Teddy	450x375	59
Cones	470x375	59
Tsukuba	384x288	15
Sawtooth	434x380	19



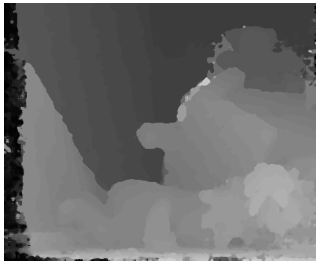
Teddy



BFA



BP1D



SGM

2. Stereovision on Manycore Embedded Processors

[5, 6]

- Multicore execution (Keystone)

	1 core	2 cores	4 cores	8 cores
BFA	52ms	29ms (x1.8)	16ms (x3.2)	11ms (x4.9)
BP1D	41ms	20ms (x2)	10ms (x4)	5ms (x8)
SGM	330ms	174ms (x1.9)	92ms (x3.6)	47ms (x7)

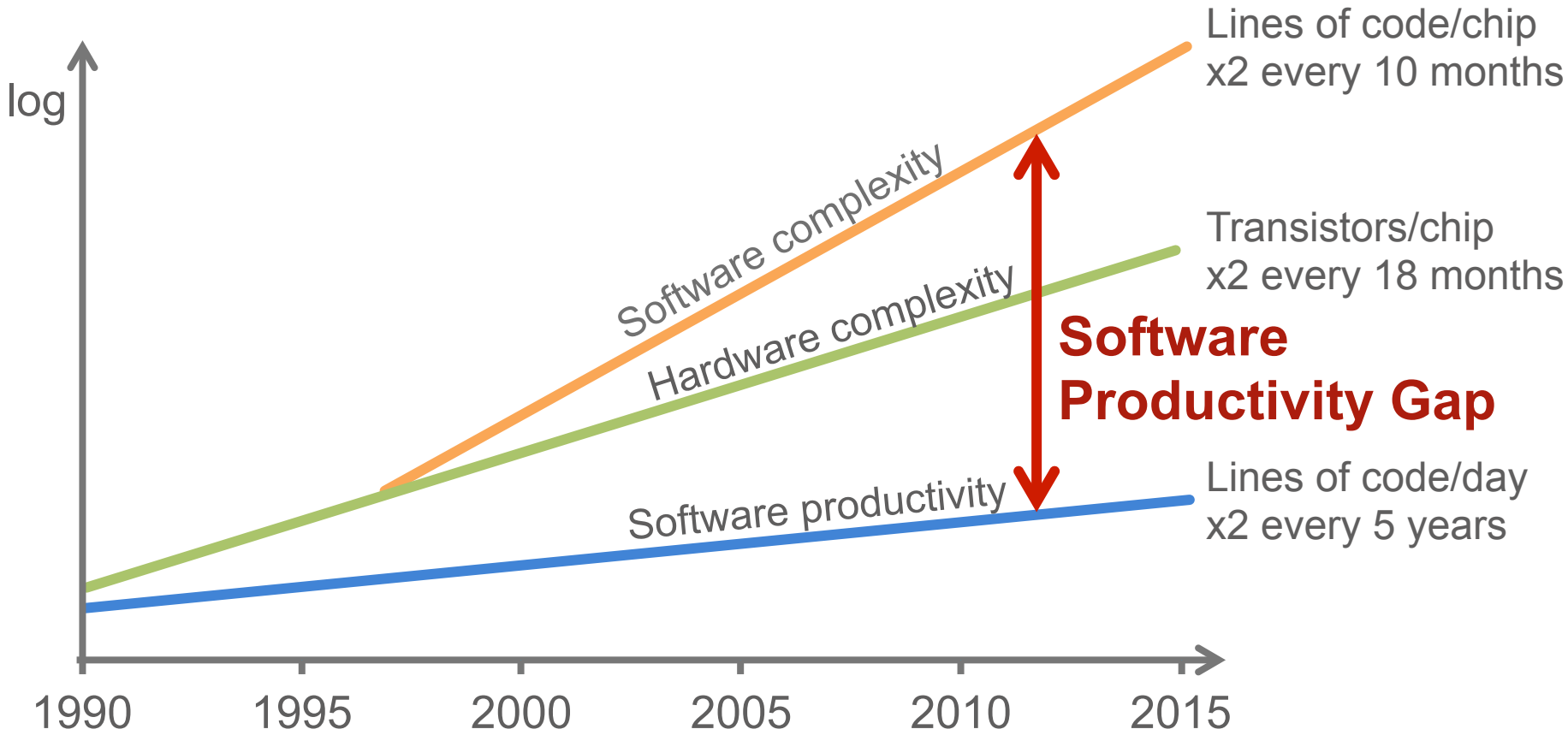
2. Stereovision on Manycore Embedded Processors

- On going work
  - Going to manycore architectures
    - 256 cores of the MPPA
    - Impact of the algorithms in terms of memory used
  - Automatic stereo vision configuration
    - Algorithm, Resolution, Number of disparities,
    - Available Hardware (number of cores, available memory)
    - Constraints (Quality, Power Consumption, Time)
- Open issues
  - New algorithms
  - Hardware dependence
  - Evaluation of the constraints (especially the required quality)

Outline

1. Introduction
2. Stereomatching on Manycore Embedded Processors
3. Programming multicore embedded processors
4. Dataflow programming and Design Space Exploration

3. Programming multicore embedded processors

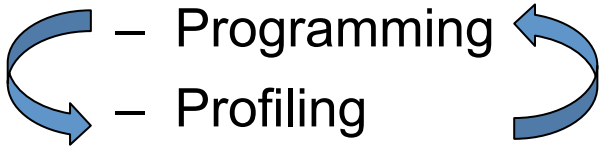


Source: ITRS & *Hardware-dependent Software*, Ecker et al., Springer

[3, 7]

3. Programming multicore embedded processors

- "How to optimize one algorithm on a given hardware ?"
  - Programming
  - Profiling



- "What is the best hardware for one algorithm/application ?"
  - Repeat the process on each candidate hardware
  - Extrapolations when Hardware is not available
- What do we need to improve the situation ?
  - 1 code fits all hardware
  - "Automatic" extrapolations = Design Space Exploration (DSE)

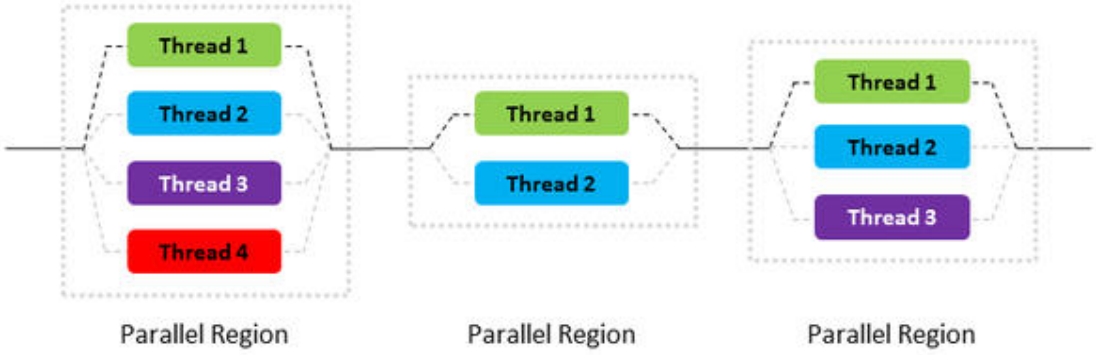
3. Programming multicore embedded processors

- OpenCV (Open Source Computer Vision Library)
  - Huge library mostly used for testing new algorithms
  - Different implementations of the API in C, C++, Python and Java
  - No Hardware specific implementations
  - No multicore support

- Multithreading programming
  - POSIX processes and threads
  - Pthreads features
    - Thread Management, Mutex, Shared variables …
    - No preemption mechanism between Threads
  - Usually associated with MPI (Message Passing Interface)

  - Drawbacks :
    - Low-level programming
    - Possibility to introduce bugs and locks difficult to debug
    - Shared Memory Model

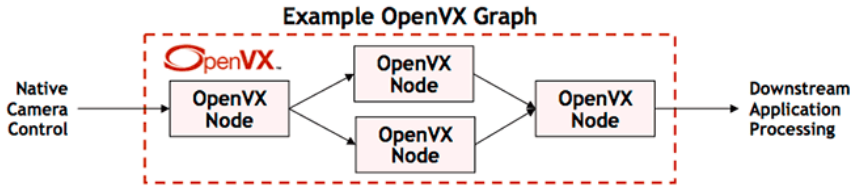
- OpenMP
  - A set of compiler directives, library and environment variables
  - Use of serial C Code + Compiler directives
  - Automatic generation of Threads
  - Drawbacks
    - Shared memory model
    - Possibility to introduce bugs and locks difficult to debug
    - Hardware provider dependent (compiler optimization – no GPU)

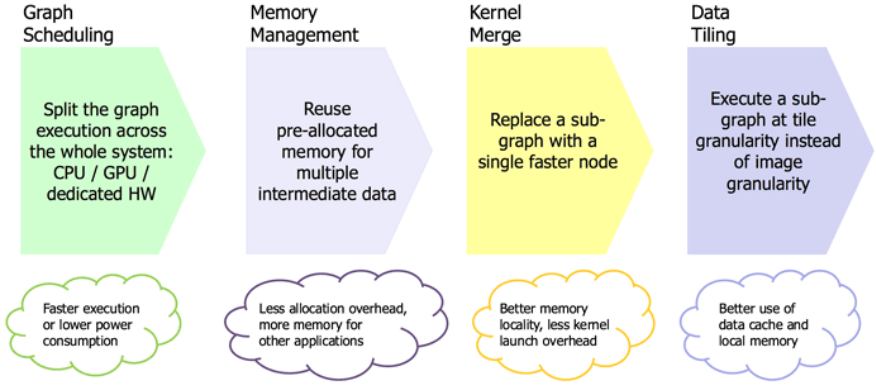


3. Programming multicore embedded processors

- OpenCL
  - Khronos Group (Altera, AMD, Apple, ARM, IBM, Intel, Nvidia …)
  - C based programming language (kernels) + APIs
  - Acceleration programming model (Host / Accelerators)
    - CPUs, GPUs, FPGAs, DSPs, Hardware accelerators
    - Kernels are offloaded on accelerators
  - 4 level memory hierarchy model
    - Global, read-only, local and private
  - Drawbacks
    - Hardware dependent code
    - Most of Hardware only support subparts of OpenCL

- OpenVX
  - Khronos Group (Intel, Texas Instruments, Cadence, Nvidia …)
  - Targets computer vision applications and embedded systems
    - + Neural Network Extension (CNN)
  - Acceleration programming model (Host / Accelerators)
  - OpenVX Dataflow Graphs





  - Drawbacks
    - Nodes are standard and optimized by the hardware provider
    - Application Dependent (Computer vision / CNN specific)
    - “Static” dataflow graphs

- To sum up
  - "Open" Standards
  - Multiple "Standards"
  - Standards in evolution (OpenMP4, OpenCL 2.2, …)
  - Next steps :
    - Multi-standard approaches (standards can be associated with a specific hardware)
    - Hardware independent specification + Hardware dependent code generation (based on existing standards)
  - On the road for "1 code fits all hardware"
    - C-based approaches (legacy code reuse)
    - Dataflow approaches (Communication between "functions" or "code blocks")

Outline

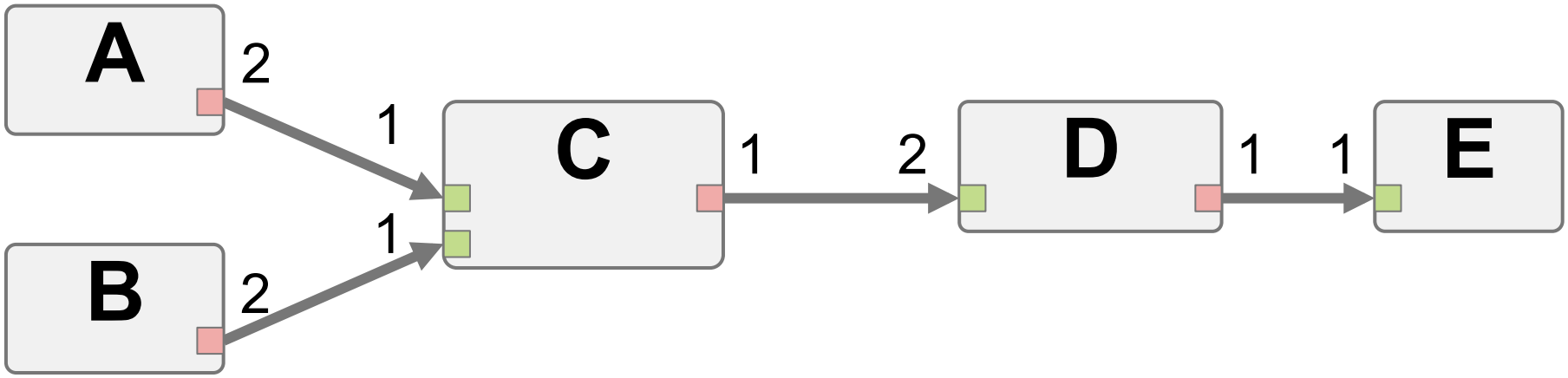
1. Introduction
2. Stereomatching on Manycore Embedded Processors
3. Programming multicore embedded processors
4. Dataflow programming and Design Space Exploration (DSE)

Dataflow programming aims

- To provide metrics
  - Design of parallel algorithms
  - Throughput/Latency evaluation
  - Predictable memory footprints
- To build a working prototype
  - Code generation for multicore architectures
  - Guaranteed deadlock-freeness
  - Inter-core communications
- For design-space exploration
  - Seamless porting to a new architecture
  - Legacy code reusability

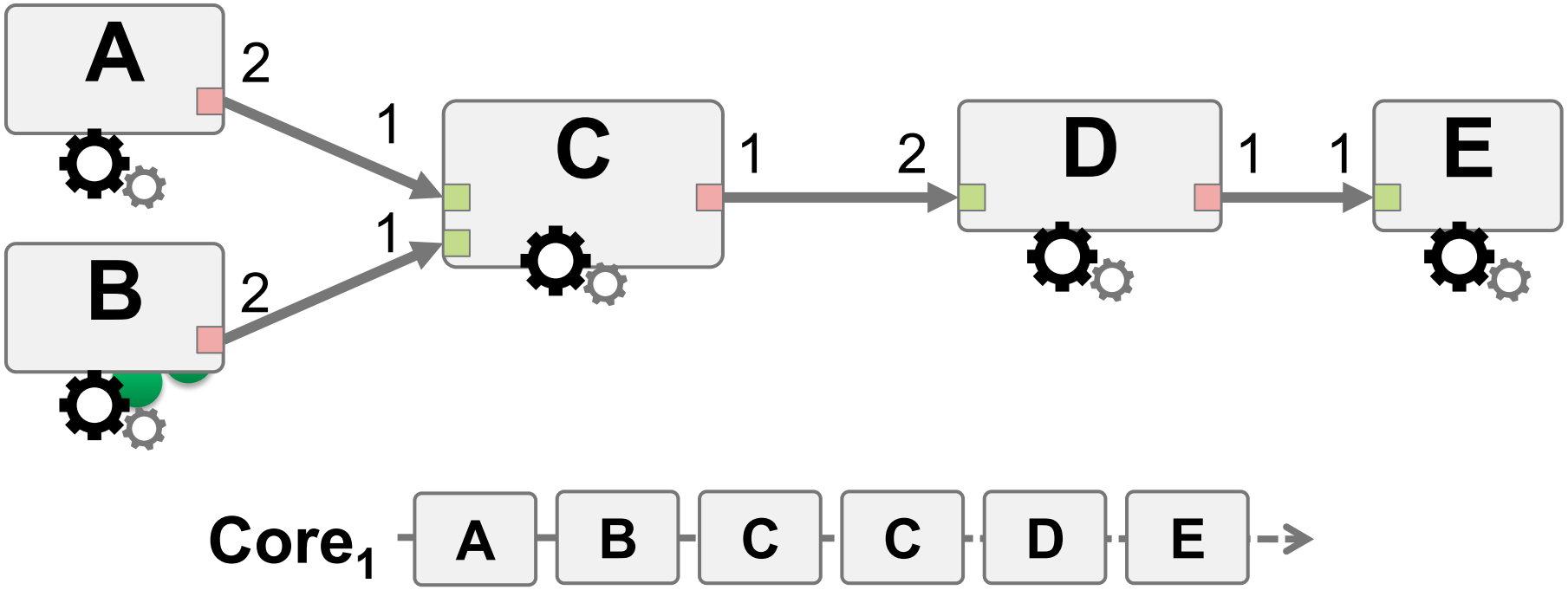
Algorithm descriptions using Dataflow Graphs [8]

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



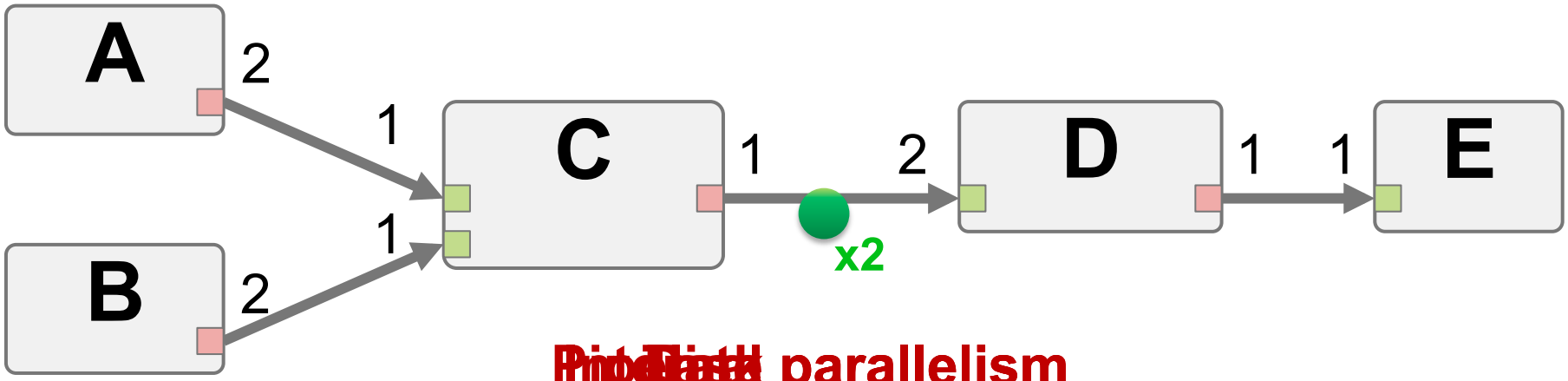
Algorithm descriptions using Dataflow Graphs [8]

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

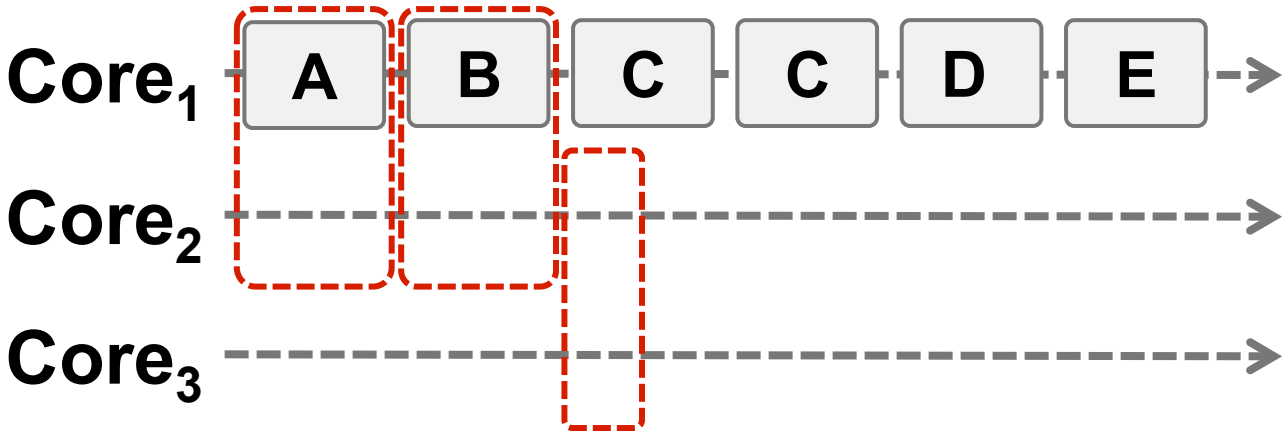


Algorithm descriptions using Dataflow Graphs [8]

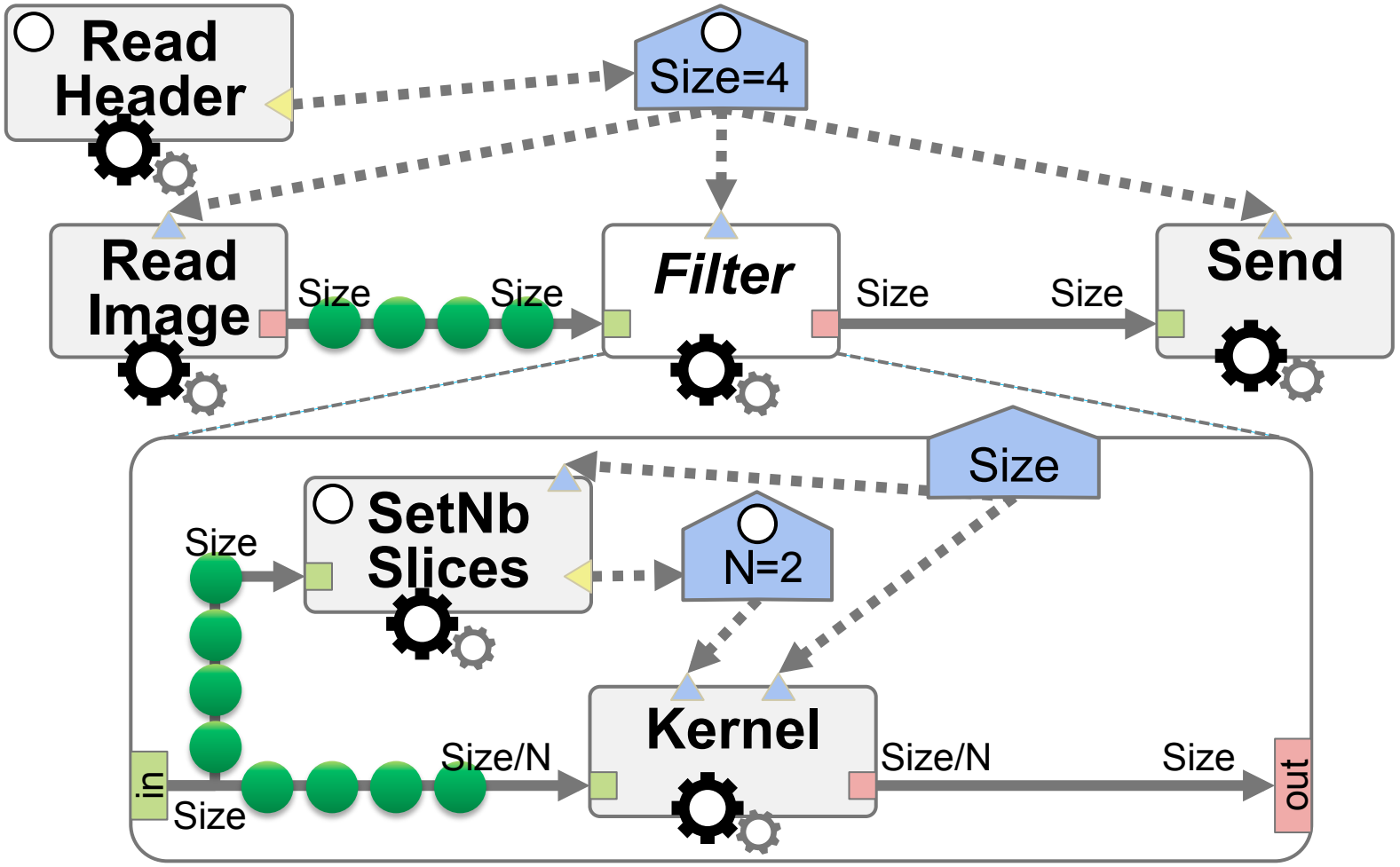
- Expression of parallelisms:Task/ Data /Pipeline/ Internal



Pipeline parallelism



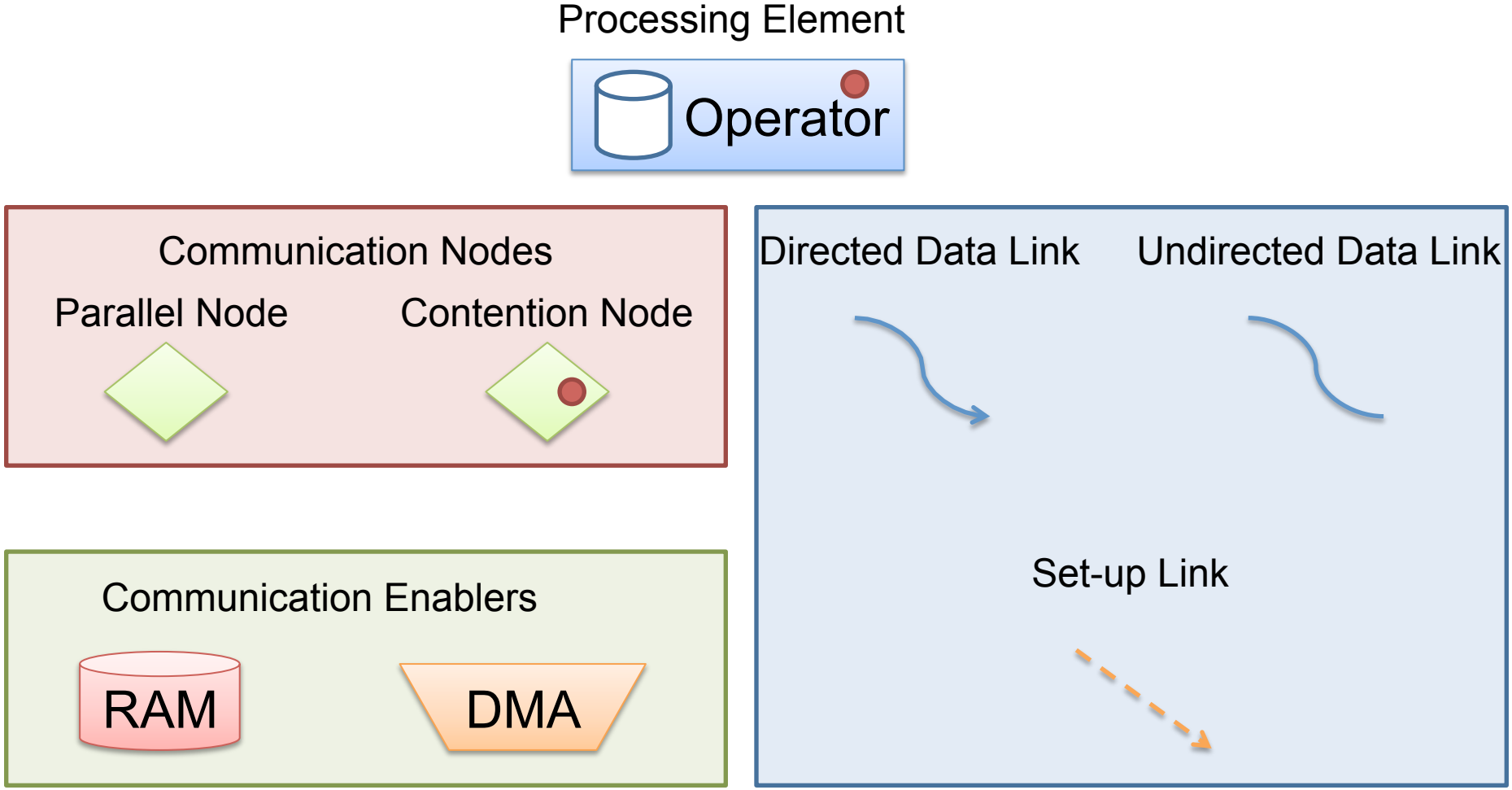
PiSDF (Parameterized and Interfaced Synchronous Dataflow) [9]



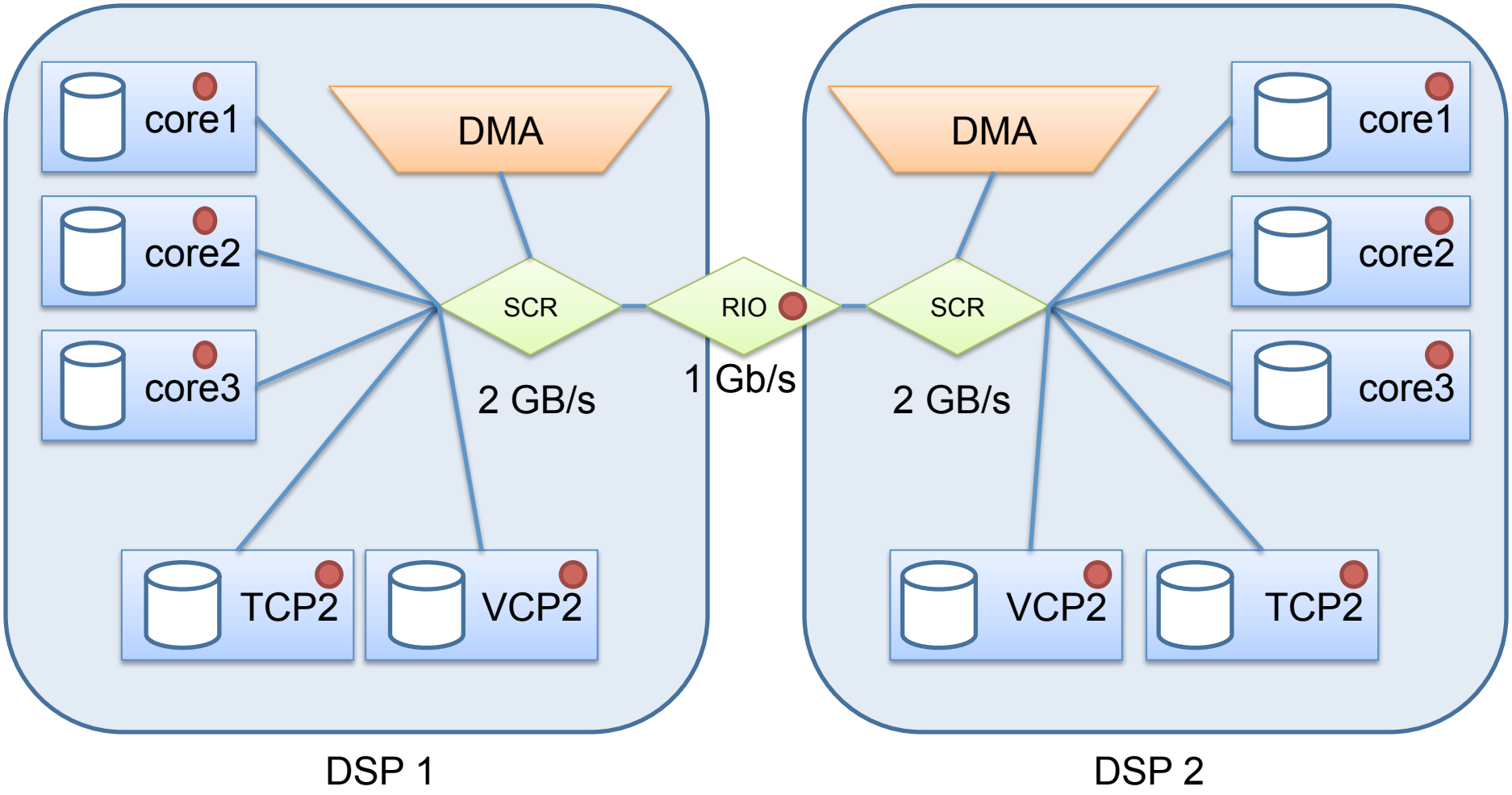
PiSDF (Parameterized and Interfaced Synchronous Dataflow)

- PiSDF is: [9]
  - Hierarchical & Compositional
  - Statically parameterizable
  - Dynamically reconfigurable
- PiSDF fosters:
  - Predictability
  - Parallelism
  - Lightweight runtime overhead
  - Developer-friendliness

S-LAM (System-Level Architecture Model) [10]



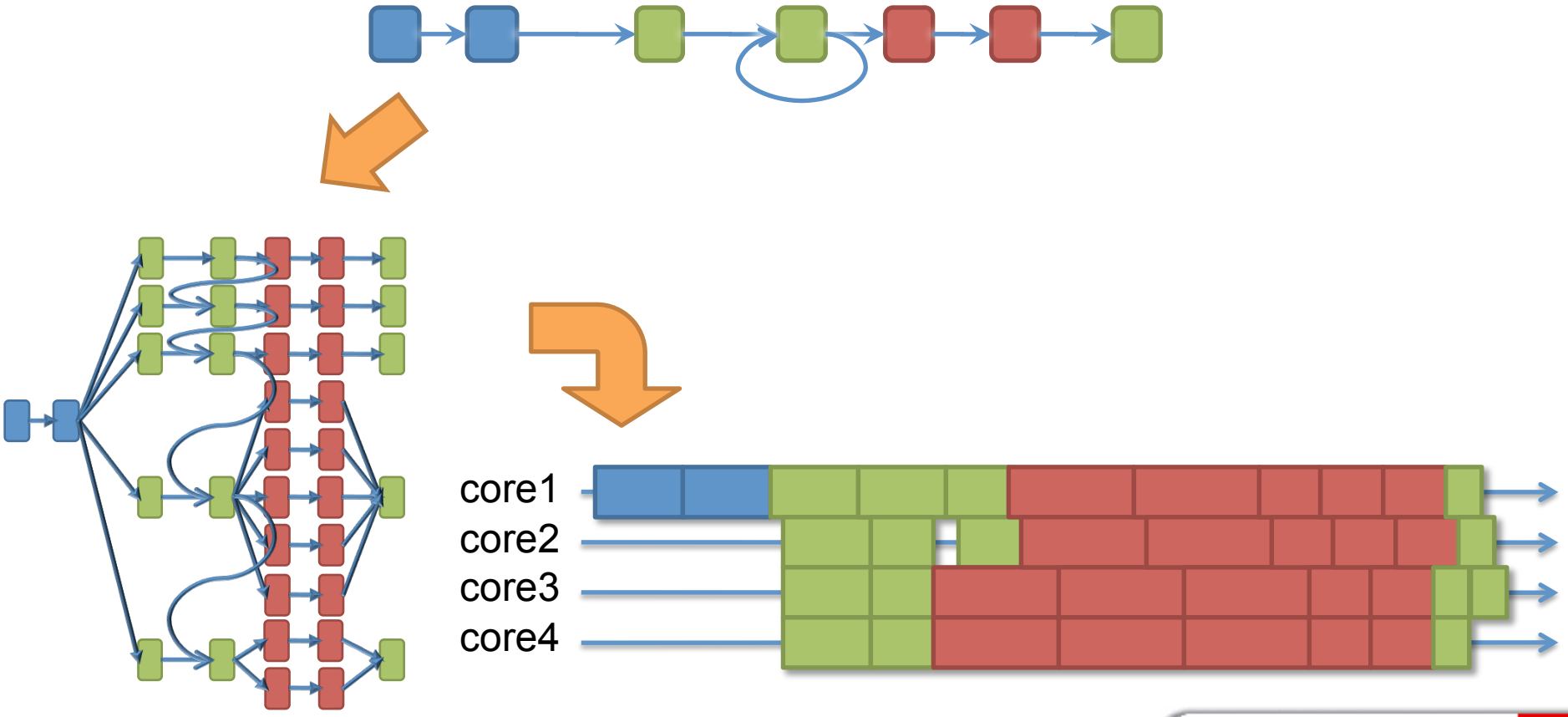
S-LAM (System-Level Architecture Model) [10]



Mapping/Scheduling for static graphs

[11]

- State-of-the-art algorithms (FAST, List, …)
- Latency and load balancing optimization



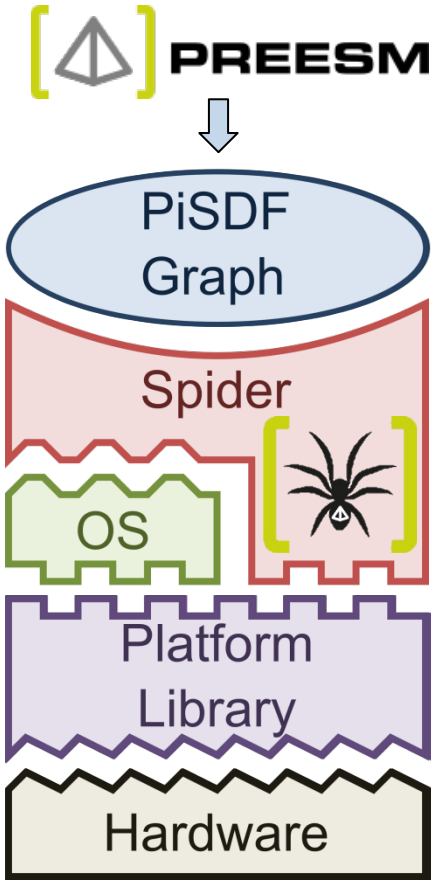
Code generation for multiple targets [12]

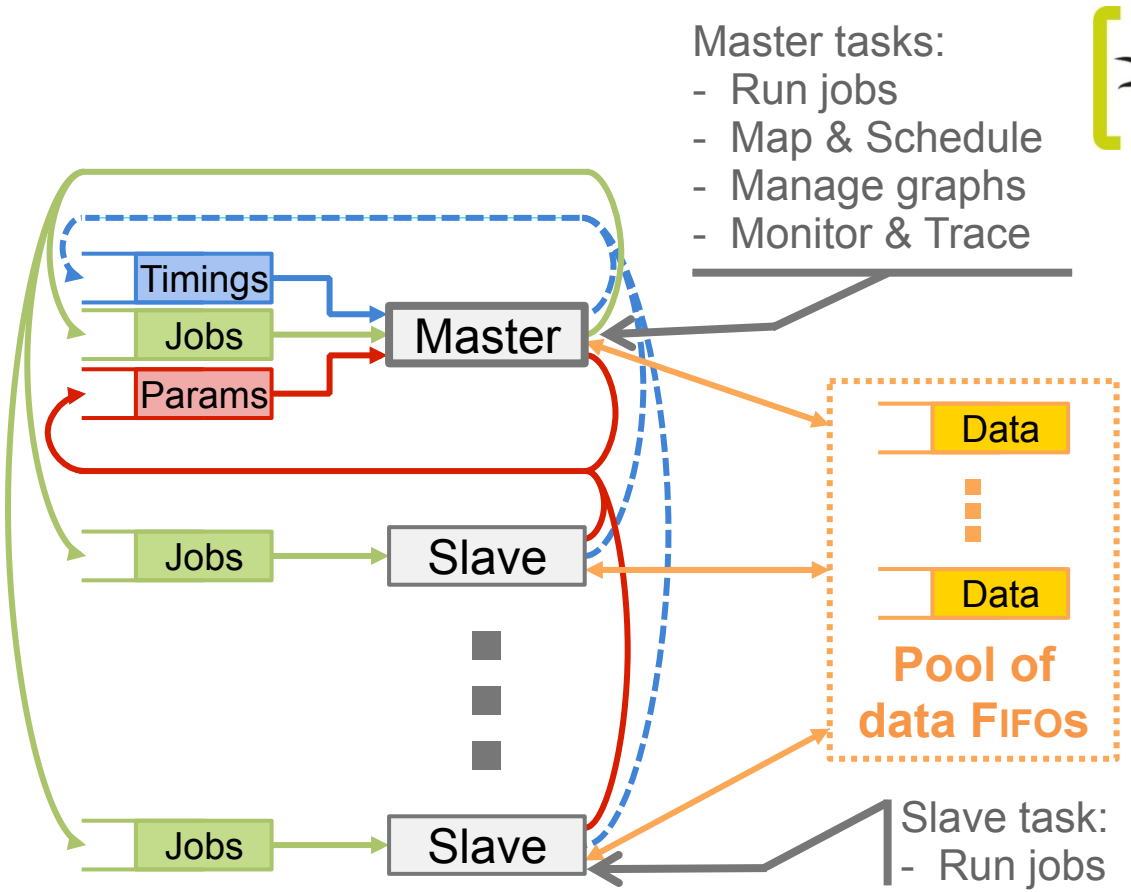
- Multi-C6X DSPs:
  - TMS320c6678 from Texas Instruments
  - Supports the activation of the DSP caches.
- Multi-x86 and multi-ARM CPUs:
  - Linux and Windows, pthread
- OMAP4 heterogeneous platform:
  - dual-core ARM Cortex-A9, 2 Cortex-M3, and a C64xT DSP.
- MPPA
  - Hardware specific library between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

Mapping/Scheduling for reconfigurable PiSDF

[13]

- SPIDER: Synchronous Parameterized and Interfaced Dataflow Embedded Runtime

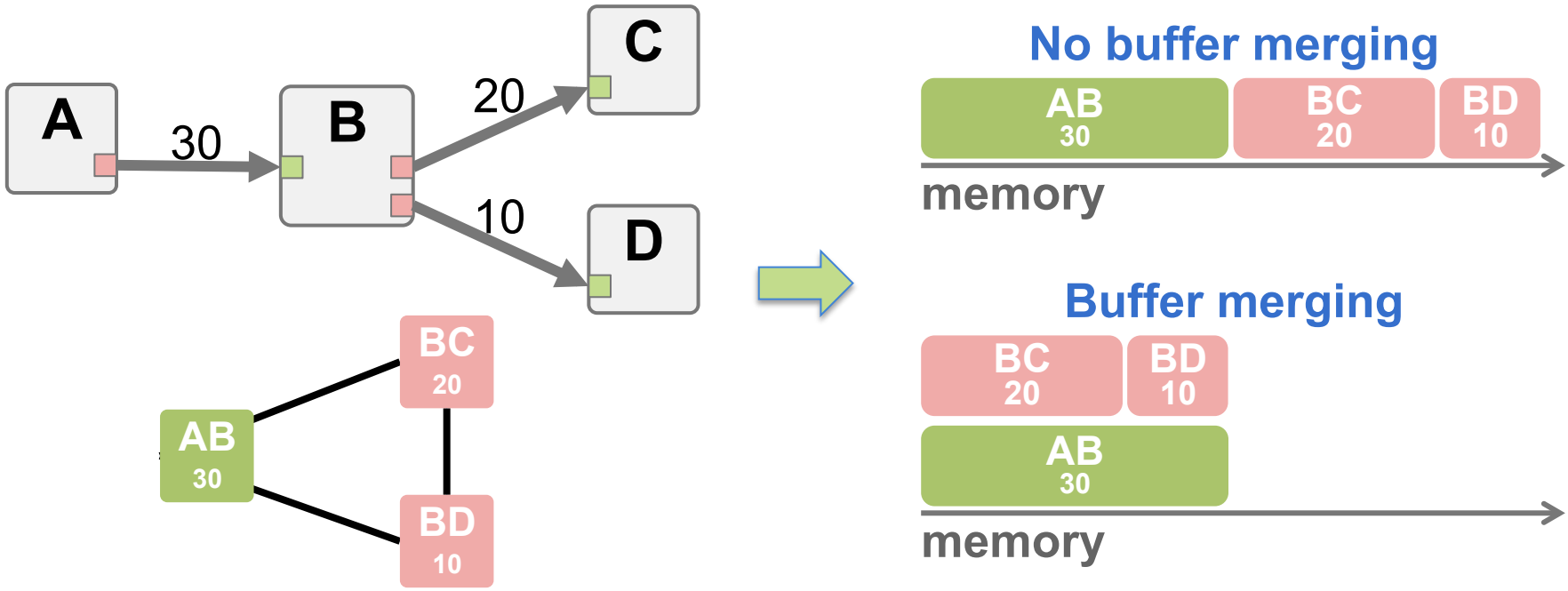




[14]

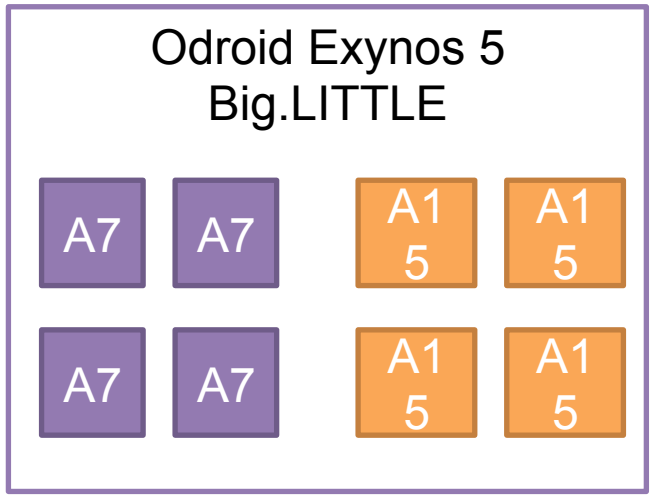
Memory optimizations for static graphs

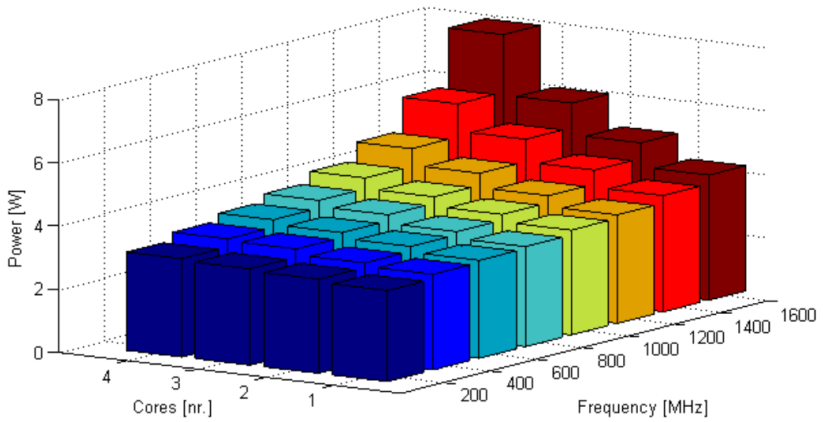
- Buffer merging technique for SDF graphs

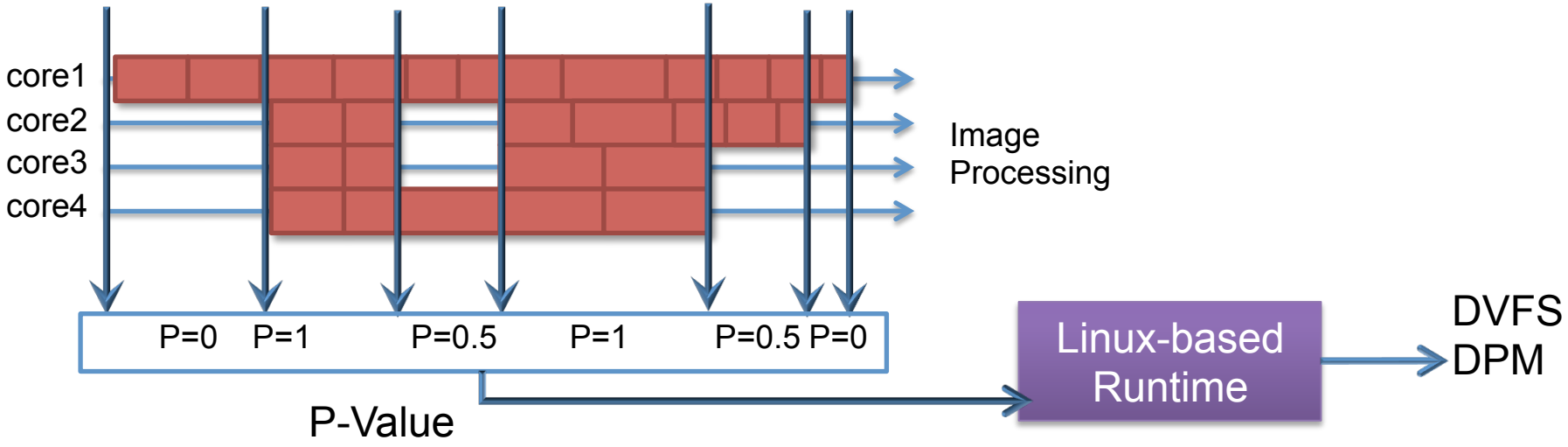


Energy optimization: platform Exynos 5

[15]



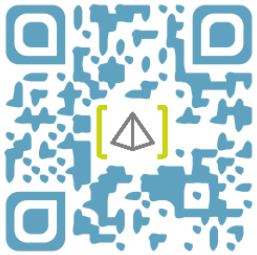




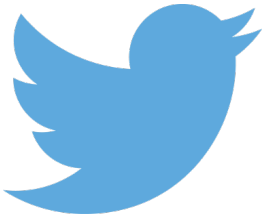
- Throughput evaluation of dataflow graphs for DSE [16]

IBSDF Graph			SrSDF	Exec. Time		
Name	Levels	Actors	Actors	ASAP	Periodic	Sched-Rep
Crypto	2	10	34	**4 ms**	8 ms	43ms
Large FFT	2	10	267	**29 ms**	48 ms	46 ms
LTE	4	18	250	**22 ms**	32 ms	47 ms
Stereo	2	41	1 604	3 676 ms	151 ms	**51 ms**
Graph 1	3	15	503	493 ms	67 ms	**47 ms**
Graph 2	5	20	17 727	>5 min	3 060 ms	**46 ms**
Graph 3	6	24	84 440	>5 min	14 600 ms	**43 ms**
Graph 4	5	150	653 289	>5 min	234 000ms	**69 ms**
Graph 5	8	240	39 E10	-	-	**70 ms**
Graph 6	10	100	31 E15	-	-	**73 ms**

Questions ?



http://preesm.sf.net



@PreesmProject

http://preesm.sourceforge.net/website/

[1] Z. Nikolic and al. "*TI Gives Sight to Vision-Enabled Automotive Technologies*". Texas Instruments White Paper. 2013
[2] http://www.marketwatch.com/story/want-to-invest-in-self-driving-cars-check-out-the-chips-2016-08-26
[3] International Technology Roadmap for Semiconductor "System Drivers". 2011 Edition
[4] Nezan and al "*Optimized Belief Propagation Algorithm onto Embedded Multi and Many-Core Systems for Stereo-Matching*". PDP16
[5] Menant and al "A comparison of cost construction methods onto a C6678 platform for stereo matching", DASIP 2016
[6] Menant and al "Comparison of stereo matching algorithms on multi-core digital signal processor platform ». 3DIPM 2017
[7] Ecker and al "*Hardware-dependant Software : Principles and Practice*" Springer 2009
[8] E. Lee and D. Messerschmitt, *"Synchronous data flow"*, Proceedings of the IEEE, 1987.
[9] Desnos and al "*PiMM: Parameterized and Interfaced Dataflow Meta-Model for MPSoCs Runtime Reconfiguration*", SAMOS XIII
[10] Pelcat and al, *"A System-Level Architecture Model for Rapid Prototyping of Heterogeneous Multicore Embedded Systems"*, 2009
[11] Pelcat and al "*Physical Layer Multicore Prototyping: A Dataflow-Based Approach for LTE eNodeB*". Springer. 2012
[12] Hascoet and al *"Hierarchical Dataflow Model for Efficient Programming of Clustered Manycore Processors*". ASAP2017
[13] Heulot and al "*SPIDER: A Synchronous Parameterized and Interfaced Dataflow-Based RTOS for Multicore DSPs*". EDERC 2014,
[14] Desnos and al "Buffer merging technique for minimizing memory footprints of Synchronous Dataflow specifications". ICASSP 2015
[15] Holmbacka and al "*Energy Efficiency and Performance Management of Parallel Dataflow Applications*". DASIP 2014
[16] Deroui and al "*Throughput Evaluation of DSP Applications based on Hierarchical Dataflow Models*". ISCAS 2017