Elaborazione dati real-time su architetture embedded many-core e FPGA

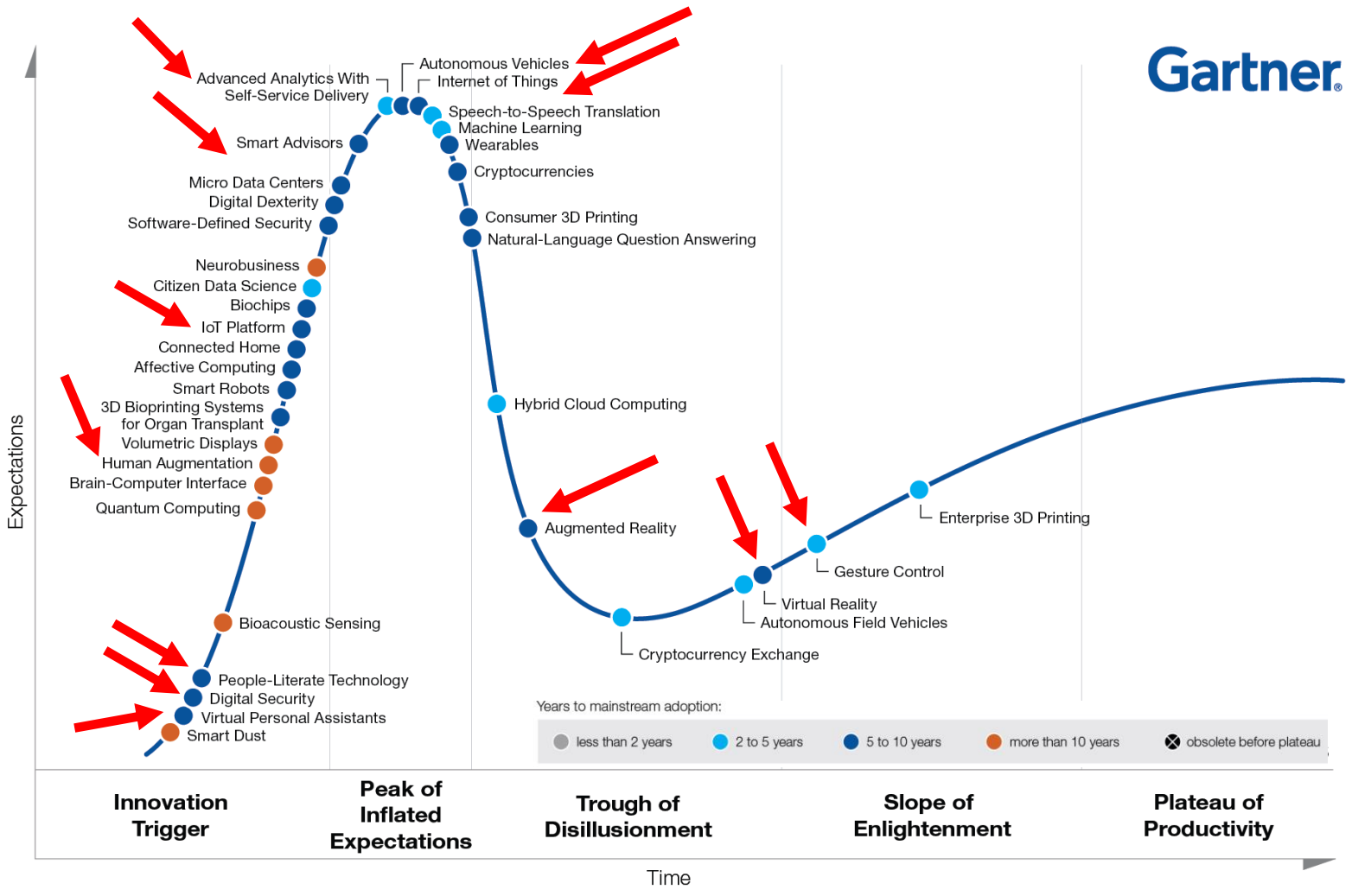
DAVIDE ROSSI

ALESSANDRO CAPOTONDI
GIUSEPPE TAGLIAVINI
ANDREA MARONGIU

CIRI-ICT Università Di Bologna

OPEN-NEXT

Heterogeneous Many-Core Architectures



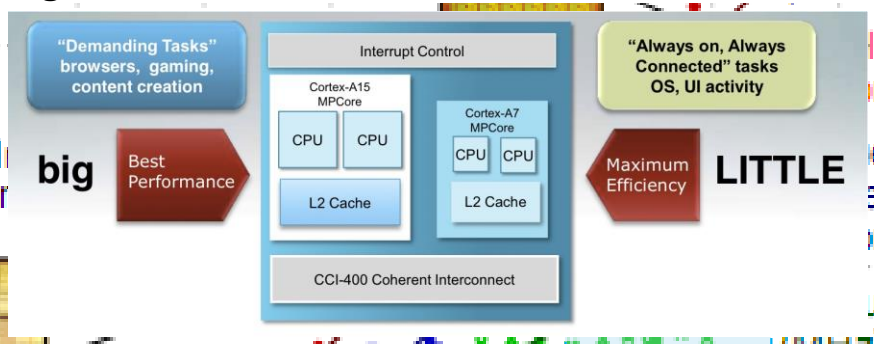
Heterogeneous Many-Core Architectures

Embedded systems need
process workloads usuall
workstations or

Multi-Processor Systems-on-Chip (MPSoCs)
- **computing units embedded** in the same die
- designed to **deliver high performance** at low **power consumption = high energy efficiency (GOPS/Watt)**

Various design schemes are available:

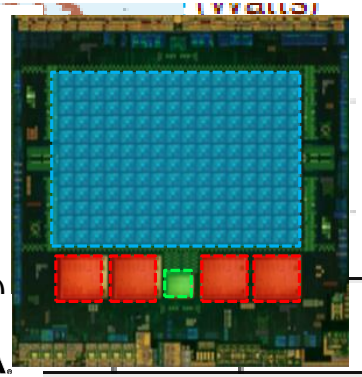
- Targeting **ADAPTIVITY: heterogeneity** and **specialization** for efficient computing. Es. ARM Big.Little

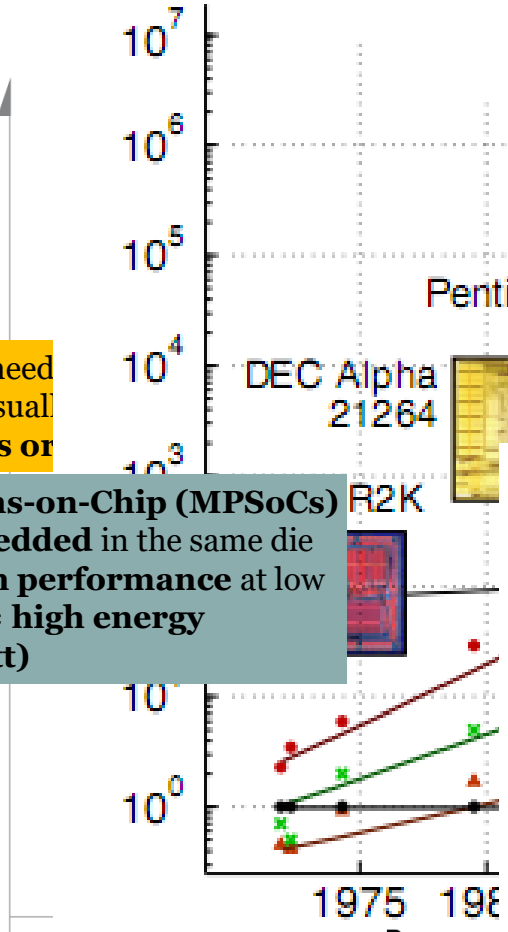


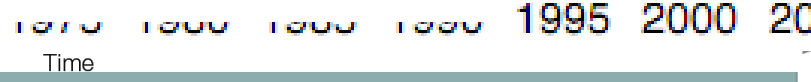
- Targeting **PARALLELISM**: massively parallel many-core accelerators, to maximize **GOPS/Watt** (i.e. GPUs, GPGPUs, PMCA).

Nvidia Tegra-K1

Two levels of heterogeneity:
- Host Processor
  (4 powerful cores + 1 energy efficient core)
- Parallel many-core co-processor
  (192 cores accelerator: NvidiaKeplerGPU)





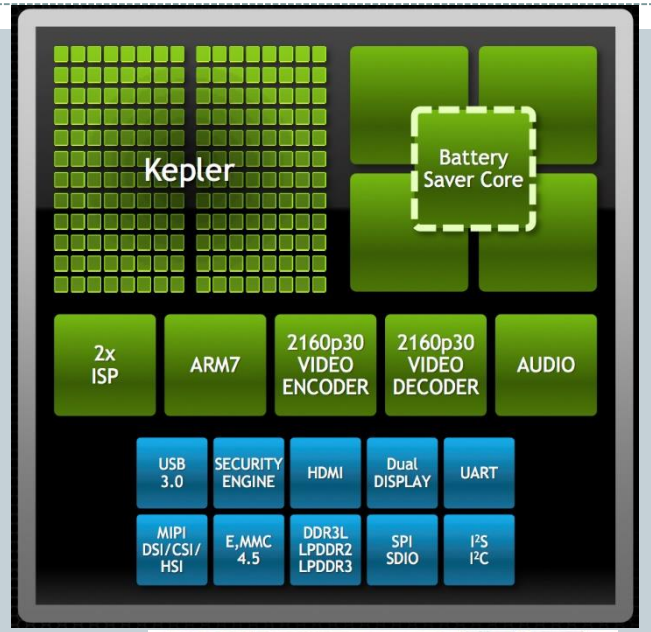


Nvidia **K1** (jetson)

Hardware Features
- Dimensions: 5" x 5" (127mm x 127mm) board
- Tegra K1 SOC (**1 to 5 Watts**):
  - **NVIDIA Kepler GPU** with 192 SM (**326 GFLOPS**)
  - **NVIDIA "4-Plus-1" 2.32GHz** ARM quad-core Cortex-A15
- DRAM: 2GB DDR3L 933MHz

IO Features
- mini-PCIe
- SD/MMC card
- USB 3.0/2.0
- HDMI
- RS232
- Gigabit Ethernet
- SATA
- JTAG
- UART
- 3x I2C
- 7 x GPIo

Software Features
- **CUDA** 6.0
- **OpenGL** 4.4
- **OpenMAX** IL multimedia codec including H.264
- **OpenCV4Tegra**

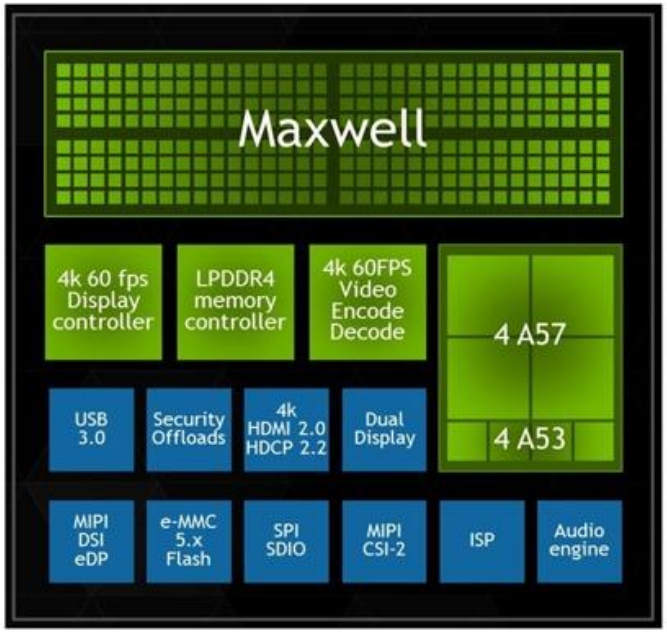


Nvidia **TX1**

Hardware Features
- Dimensions: 8" x 8” board
- Tegra TX1 SOC (**15 Watts**):
  - NVIDIA Maxwell GPU with 256 NVIDIA CUDA Cores(1 TFLOP/s)
  - Quad-core ARM Cortex-A57 MPCore Processor
- 4 GB LPDDR4 Memory

IO Features
- PCI-E x4
- 5MP CSI
- SD/MMC card
- USB 3.0/2.0
- HDMI
- RS232
- Gigabit Ethernet
- SATA
- JTAG
- UART
- 3x I2C
- 7 x GPIo

Software Features
- **CUDA** 6.0
- **OpenGL** 4.4
- **OpenMAX** IL multimedia codec including H.264
- **OpenCV4Tegra**

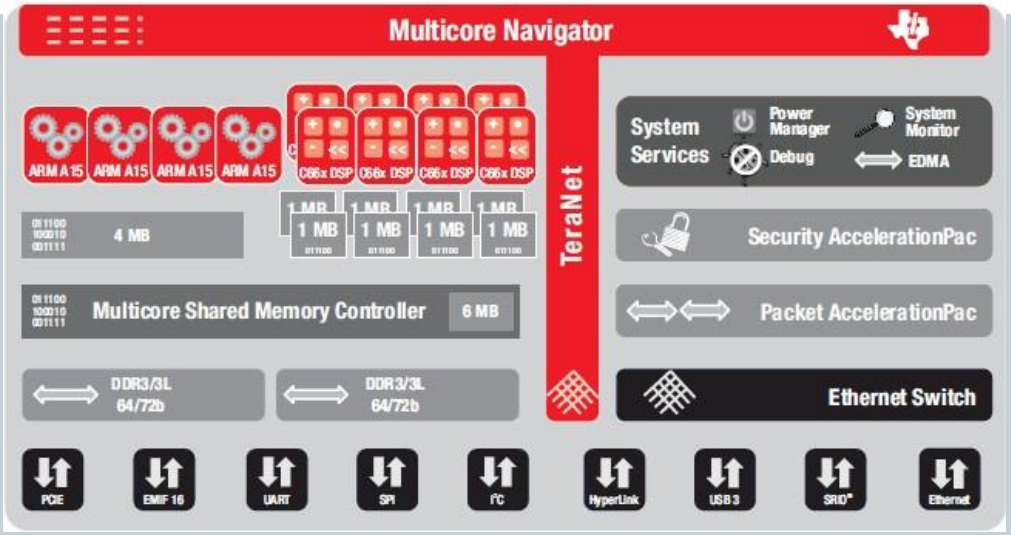


TI Keystone II

Hardware Features
- Dimensions: 8" x 8" board
- TI 66AK2H12 SOC (**14 Watts**):
  - 8x C6600 DSP 1.2 GHz (304 GMACs)
  - Quad-core ARM Cortex-A15 MPCore
- Up to 4 GB DDR3 Memory

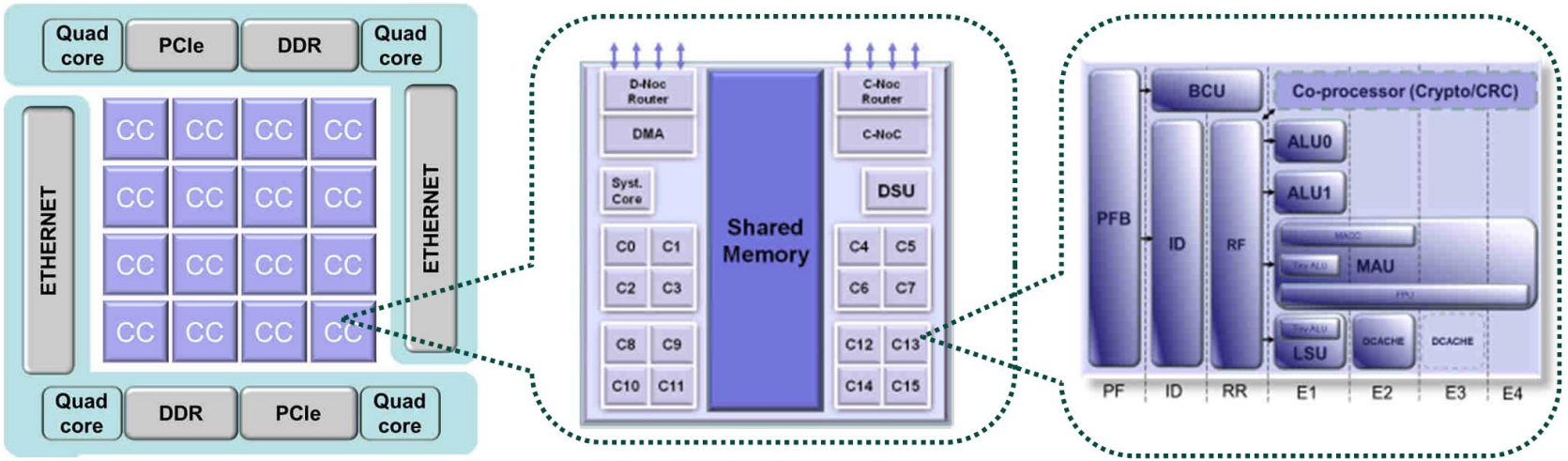
IO Features
- PCI-E
- SD/MMC card
- USB 3.0/2.0
- 2x Gigabit Ethernet
- SATA
- JTAG
- UART
- 3x I2C
- 38 x GPIo
- Hyperlink
- SRIO
- 20x 64bit Timers
- Security Accelerators

Software Features
- **OpenMP**
- **OpenCL**



MPPA Kalray

MPPA®-256 Bostan Processor Architecture



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 crpto co-processor
- 2.4 GFLOPS SP per core @600Mhz

Xilinx Zynq All Programmable SoC

Hardware Features

- Optimized for quickly prototyping embedded applications using Zynq-7000 SoCs
- Hardware, design tools, IP, and pre-verified reference designs
- Demonstrates a embedded design, targeting video pipeline

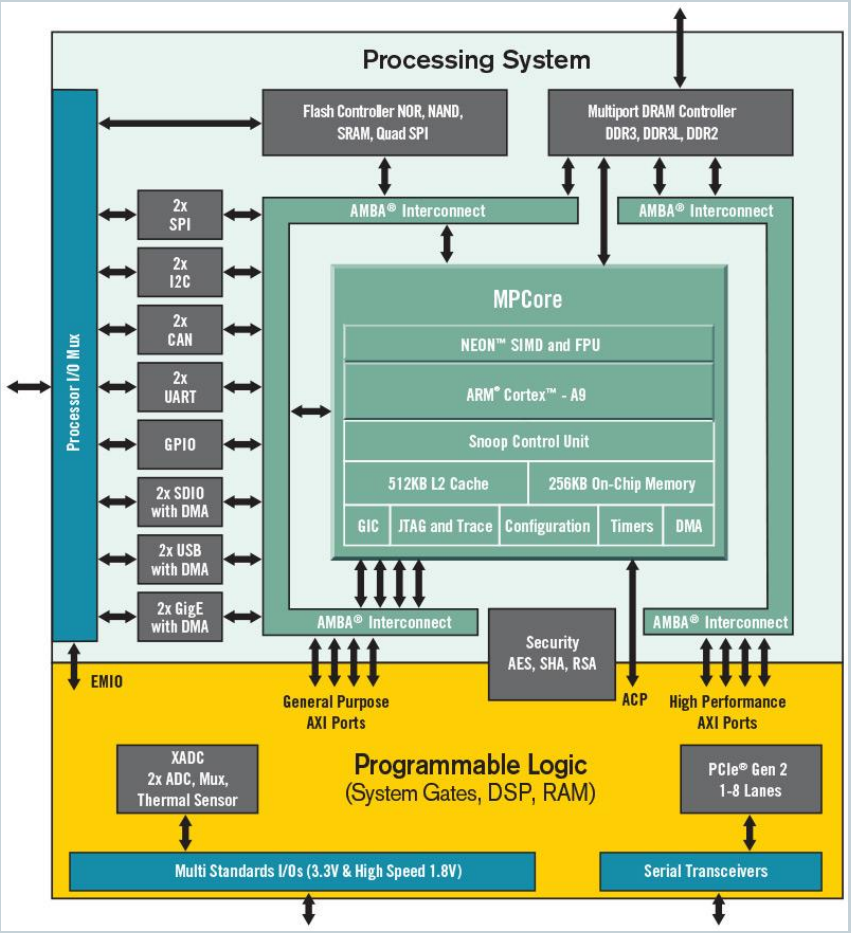
- Advanced memory interface with 1GB DDR3 Component Memory 1GB DDR3 SODIM Memory

Connectivity Features

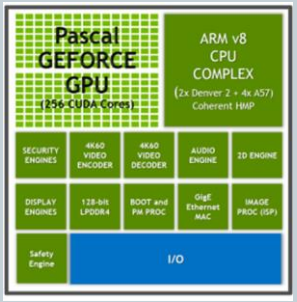
- Enabling serial connectivity with PCIe Gen2x4, SFP+ and SMA Pairs, USB OTG, UART, IIC
- Supports embedded processing with Dual ARM Cortex-A9 core processors
- Develop networking applications with 10-100-1000 Mbps Ethernet (RGMII)
- Implement Video display applications with HDMI out
- Expand I/O with the FPGA Mezzanine Card (FMC) interface

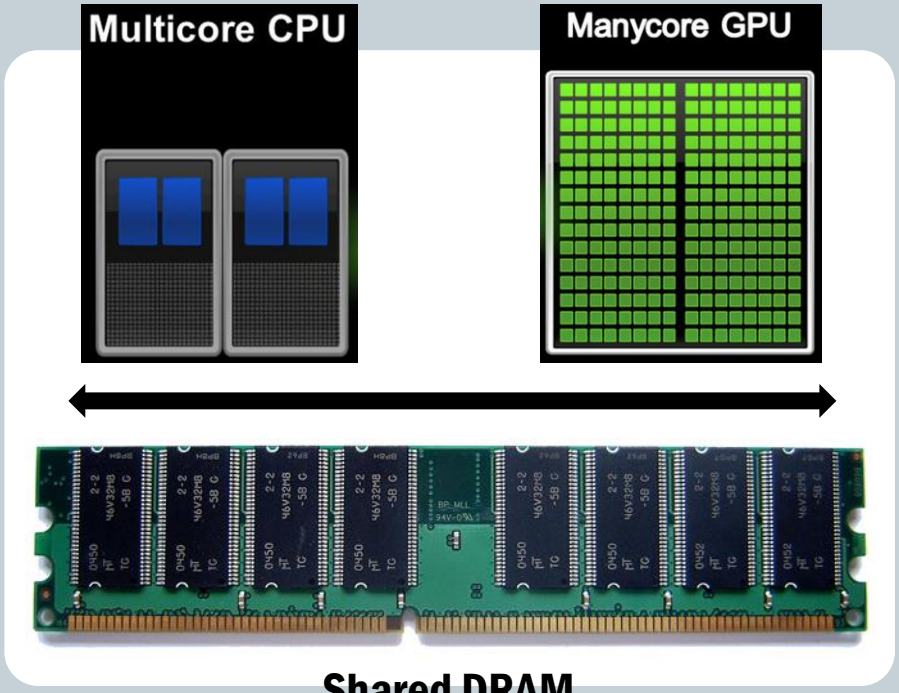
Programming Features

- **C/C++**
- **VHDL/Verilog**
- **SDSoC / OpenCL flows**



Heterogeneous SoCs Memory Model





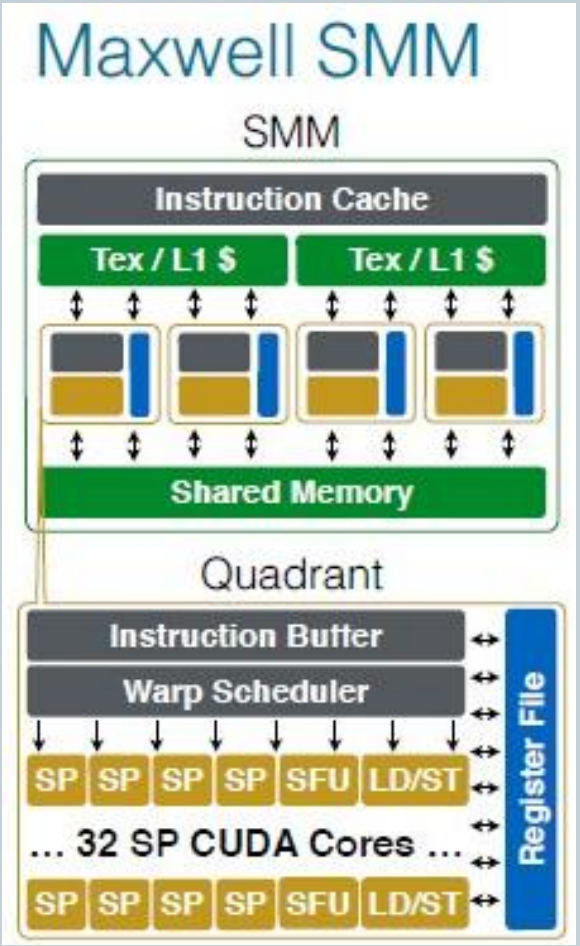
Strong push for unified memory model in heterogeneous SoCs

- **Good for programmability**
- **Optimized to reduce performance loss**
- **How about predictability?**

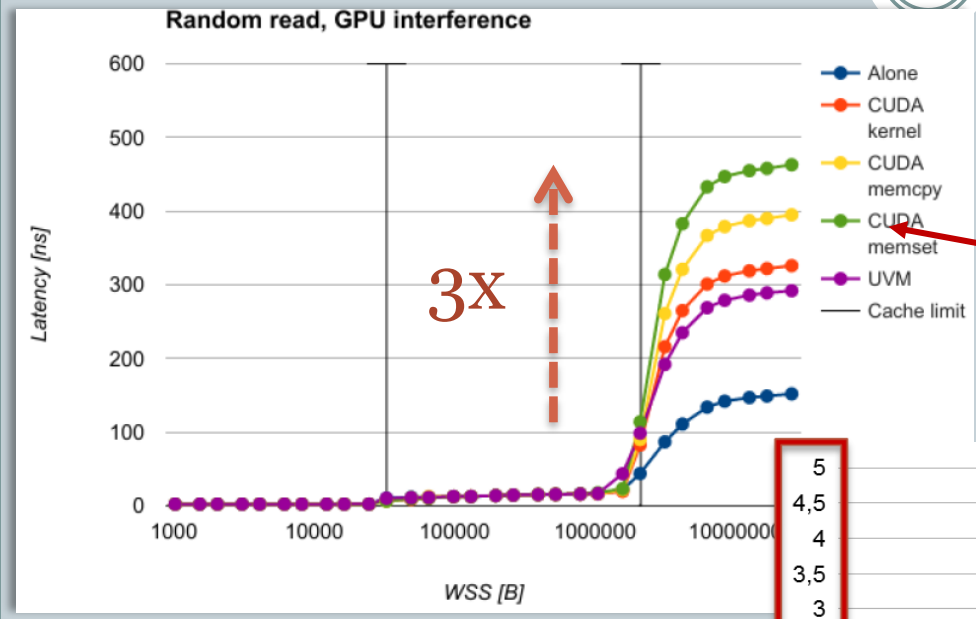
Maxwell GPGPU Architecture (Tegra X1)

MEMORY HIERARCHY

- **On-Chip Shared Memory**
  - Explicitly managed data accesses
  - No interference between host and accelerator
- **Intruction/Data Cache**
  - Refill on System Memory
  - Can cause interference between host and accelerator



How large can the interference in execution time among the two subsystems be?

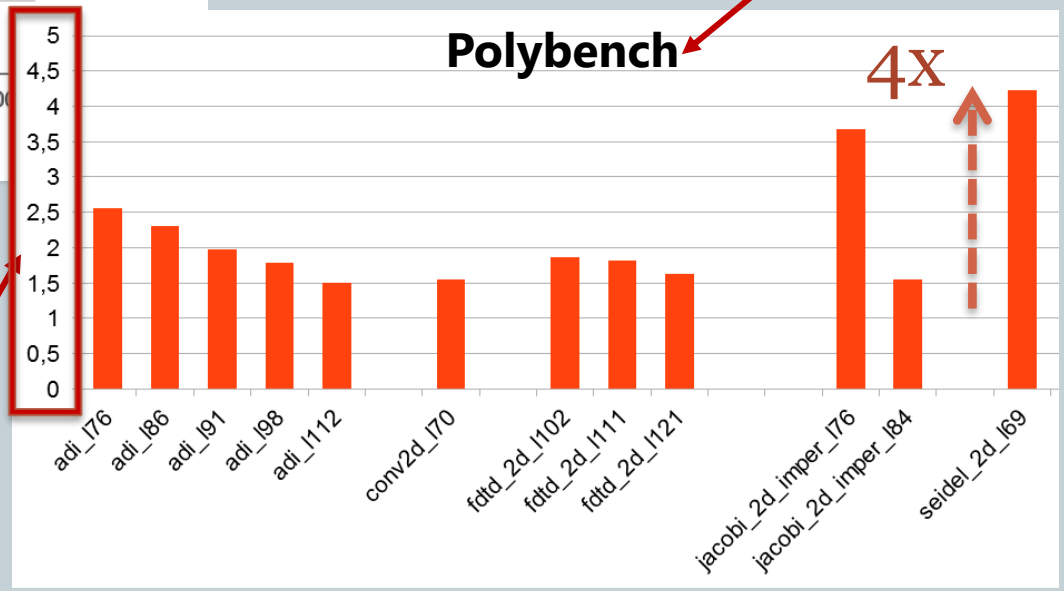


NVIDIA Tegra X1

GPU execution

A mix of synthetic workload and real bechmarks

Polybench



CPU execution

Synthetic workload to model high-interference

slowdown VS
GPU execution in isolation

How large can the interference in execution time among the two subsystems be?

HOW ABOUT REAL WORKLOADS?

- **Rodinia benchmarks executing on both the GPU and the CPU (on all 4 A57, one is observed)**
- **Up to 2.5x slower execution under mutual interference**

NVIDIA Tegra X1

CPU/GPU	euler3d	sc_gpu	lud_cuda	srad_v1	srad_v2	heartwall	leukocyte	needle	backprop	kmeans	particlefilter_float	pathfinder	lavaMD
srad_v1	1,7	1,5	1,2	1,4	1,4	1,2	1,2	1,2	1,4	1,6	1,2	1,2	1,2
srad_v2	1,2	1,4	1,1	1,3	1,2	1,1	1,1	1,1	1,3	1,5	1,1	1,2	1,1
needle	1,6	2,0	1,3	1,9	1,9	1,3	1,3	1,3	1,9	2,5	1,3	1,7	1,3
euler3d_cpu	1,2	1,4	1,1	1,1	1,3	1,1	1,1	1,1	1,3	1,5	1,1	1,2	1,1
streamcluster	1,5	1,9	1,3	1,3	1,8	1,3	1,3	1,3	2,0	2,0	1,3	1,4	1,3
lud_omp	1,1	1,3	1,0	1,2	1,2	1,1	1,0	1,0	1,2	1,4	1,0	1,1	1,0
heartwall	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,1	1,0
particle_filter	1,2	1,2	1,2	1,3	1,1	1,1	1,2	1,1	1,1	1,1	1,1	1,3	1,1
mummergpu	1,6	2,1	1,3	1,4	1,9	1,4	1,4	1,3	2,0	2,5	1,4	1,7	1,4
bfs	2,1	1,7	1,6	1,9	1,8	1,7	1,5	1,7	1,6	1,6	1,5	1,8	1,7
backprop	1,9	2,4	1,8	2,0	2,4	1,5	1,6	1,5	2,3	2,5	1,7	2,0	1,6
nn	1,2	1,2	1,2	1,2	1,2	1,2	1,2	1,2	1,2	1,2	1,2	1,2	1,2
hotspot	1,5	1,5	1,5	1,5	1,7	1,5	1,7	1,7	1,5	1,8	1,6	1,5	1,5
hotspot3D	1,5	2,0	1,5	1,6	1,9	1,4	1,5	1,9	2,4	1,9	1,6	1,8	1,6
kmeans	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,0
lavaMD	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
myocyte.out	1,1	1,1	1,0	1,1	1,1	1,0	1,1	1,0	1,1	1,2	1,1	1,1	1,1
pathfinder	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1
b+tree	1,0	1,1	1,0	1,1	1,1	1,0	1,0	1,0	1,1	1,1	1,0	1,1	1,0
leukocyte	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1
hotspot	1,7	1,6	1,6	1,6	1,6	1,7	1,6	1,7	1,6	1,6	1,7	1,5	1,7

How large can the interference in execution time among the two subsystems be?

HOW ABOUT REAL WORKLOADS?

- **Rodinia benchmarks executing on both the GPU (observed values) and the CPU (on all 4 A57)**
- **Up to 33x slower execution under mutual interference**

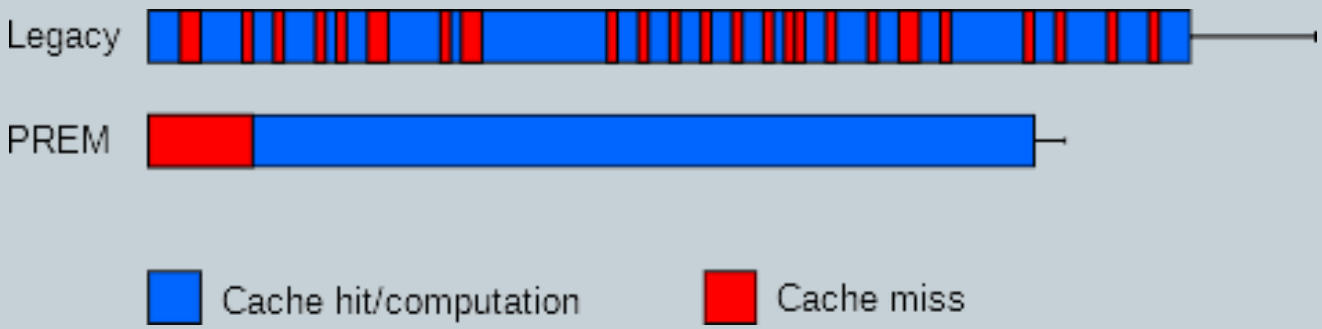
NVIDIA Tegra X1

GPU / CPU	srad_v1	srad_v2	needle	euler3d_cpu	sc_omp	lud_omp	heartwall	particle_filter	mummergpu	bfs	backprop	filelist_512k	hotspot	3D	kmeans_serial	lavaMD	myocyte.out	pathfinder	b+tree.out	leukocyte	hotspot	classification
euler3d	1,2	1,2	1,3	1,1	1,2	1,2	1,0	1,0	1,2	1,1	1,4	1,0	1,0	1,1	1,1	1,0	1,1	1,1	1,0	1,0	1,0	1,2
lud_cuda	1,5	1,5	1,3	1,1	1,2	1,1	1,0	1,0	2,2	1,1	1,6	1,0	1,0	1,2	1,1	1,0	1,2	1,1	1,0	1,0	1,0	2,4
srad_v1	1,2	5,2	1,3	1,3	1,4	1,2	1,1	1,3	1,4	1,6	1,5	1,2	1,2	1,2	1,2	1,1	1,1	1,2	1,1	1,1	1,1	3,1
srad_v2	1,4	8,9	1,5	1,1	1,1	1,9	1,0	6,5	1,3	3,6	5,6	1,2	1,8	1,1	1,1	1,0	1,2	1,5	1,3	1,0	2,9	3,6
heartwall	1,1	1,4	1,2	1,0	1,1	1,1	1,0	1,1	1,8	1,0	1,4	1,0	1,0	1,1	1,0	1,0	1,2	1,1	1,1	1,0	1,1	1,2
leukocyte	1,4	1,5	1,6	1,0	1,0	1,0	1,0	1,6	2,6	1,2	1,6	1,3	1,2	1,1	1,0	1,1	1,3	1,1	1,1	1,0	1,4	2,3
needle	2,8	4,5	2,9	2,6	2,7	2,4	1,7	13,0	4,6	5,2	2,9	3,8	6,4	2,6	2,4	2,4	3,2	2,6	2,5	1,9	4,1	14,2
backprop	3,6	32,7	2,6	2,6	2,6	16,3	2,0	3,6	9,2	2,1	3,8	7,7	2,0	2,2	2,6	2,0	3,6	2,2	2,1	2,0	2,1	2,5
kmeans	19,3	21,0	14,4	1,2	2,2	4,4	1,0	29,8	4,2	28,9	22,7	9,5	19,9	2,9	4,4	1,0	1,9	15,7	5,4	1,2	8,3	3,8
particlefilter_float	1,2	1,3	1,3	1,1	1,1	1,1	1,0	1,3	1,8	1,1	1,2	1,1	1,5	1,1	1,1	1,0	1,2	1,1	1,0	1,1	1,0	1,1
pathfinder	8,0	19,3	8,2	1,1	7,5	10,5	6,4	9,3	4,1	27,2	9,2	12,1	5,4	9,6	3,4	1,6	7,4	10,8	7,8	1,1	8,2	27,7
lavaMD	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,2	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,2

The predictable execution model (PREM)

- **Predictable interval**
  - Memory prefetching in the first phase
  - No cache misses in the execution phase
  - Non-preemptive execution

Requires compiler support for code re-structuring

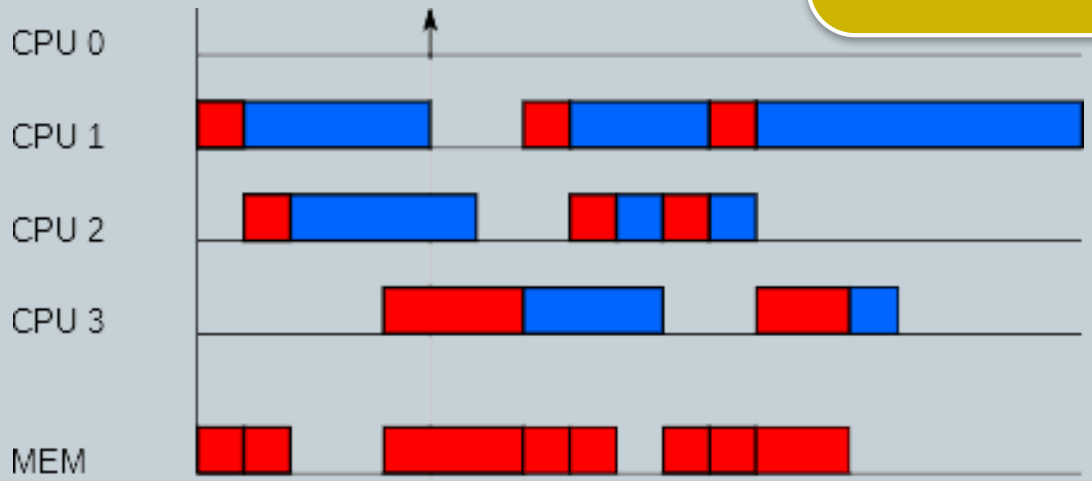


Originally proposed for (multi-core) CPU.
We study the applicability of this idea to heterogeneous SoCs

The predictable execution model (PREM)

- **System-wide co-scheduling of memory phases from multiple actors**

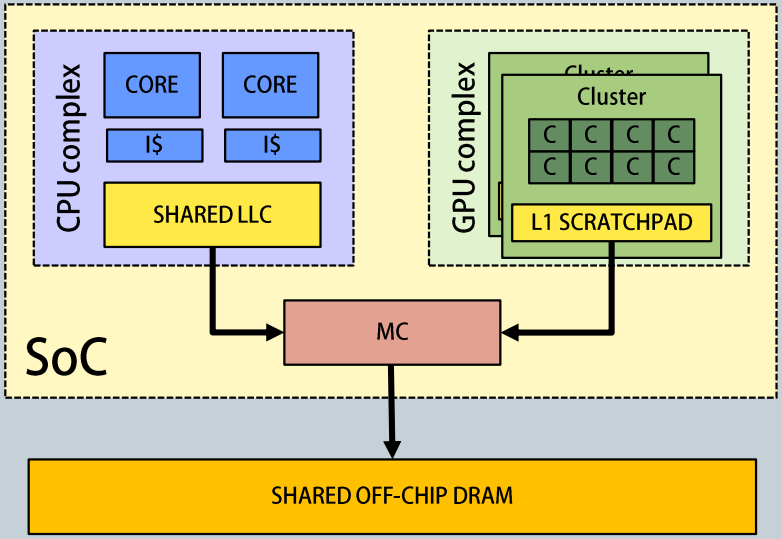
Requires runtime techniques for global memory arbitration

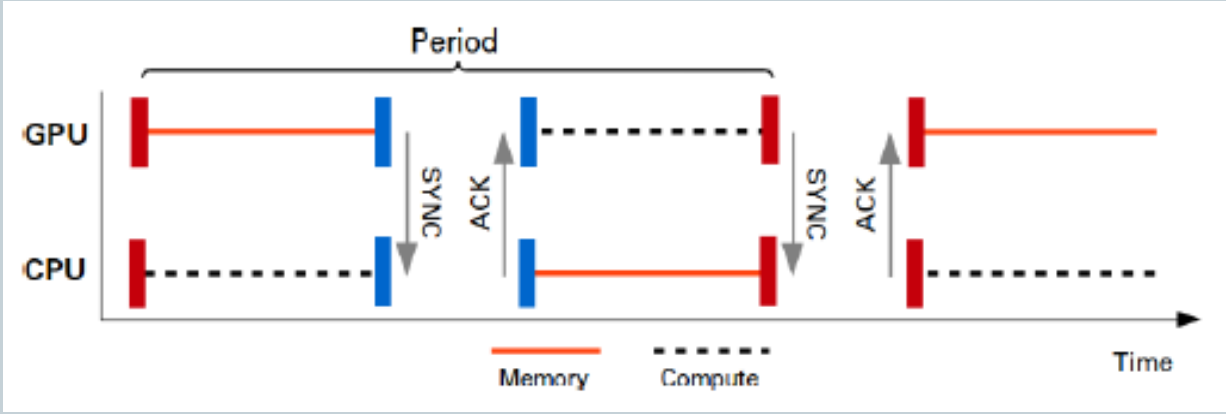


Originally proposed for (multi-core) CPU.
We study the applicability of this idea to heterogeneous SoCs

A heterogeneous variant of PREM

- **Current focus on GPU behavior**
  (way more severely affected by interference than CPU)
  - SPM as a predictable, local memory
  - Implement PREM phases within a single offload
    - **Arbitration of main memory accesses via timed interrupts+shmem**

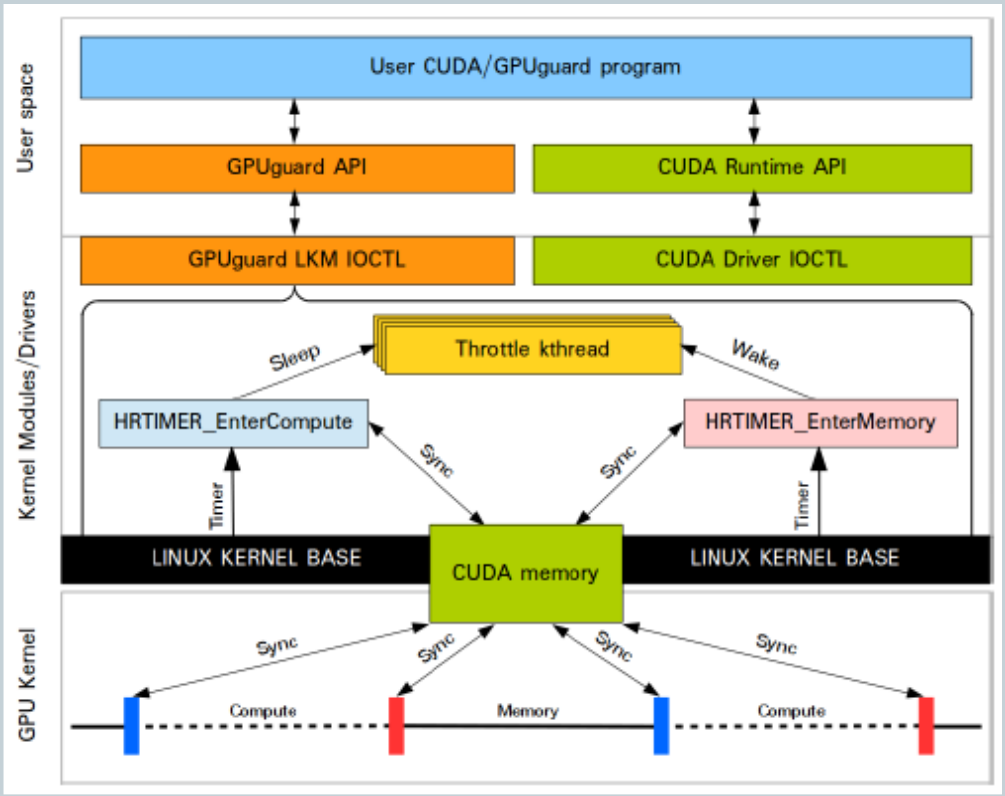




CPU/GPU synchronization

Basic mechanism in place to control GPU execution

- **CPU inactivity forced via the *throttle thread* approach (MEMguard)**
- **GPU inactivity forced via active polling on the on-chip shared memory (GPUguard)**



OpenMP offloading annotations

```
#pragma omp target teams distribute parallel for
for (i = LB; i < UB; i++)
      C[i] = A[i] + B[i]
```

The annotated loop is to be offloaded to the accelerator

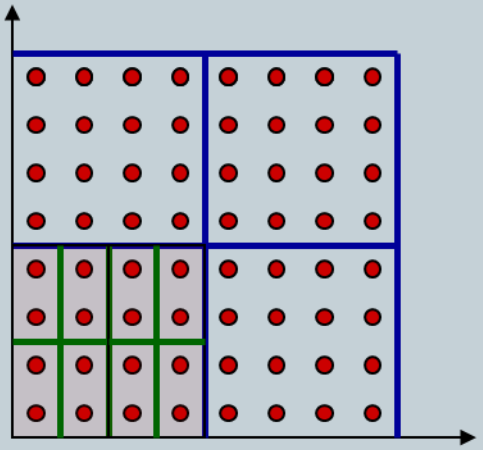
Divide the iteration space over the available execution groups (SMs in CUDA)

Execute loop iterations in parallel over the available threads

Loops

- Apply *tiling* to reorganize loops so as to operate in stripes of the original data structures whose footprint can be accommodated in the SPM
  - Can leverage well-established techniques to prepare loops to expose the most suitable access pattern for SPM reorganization

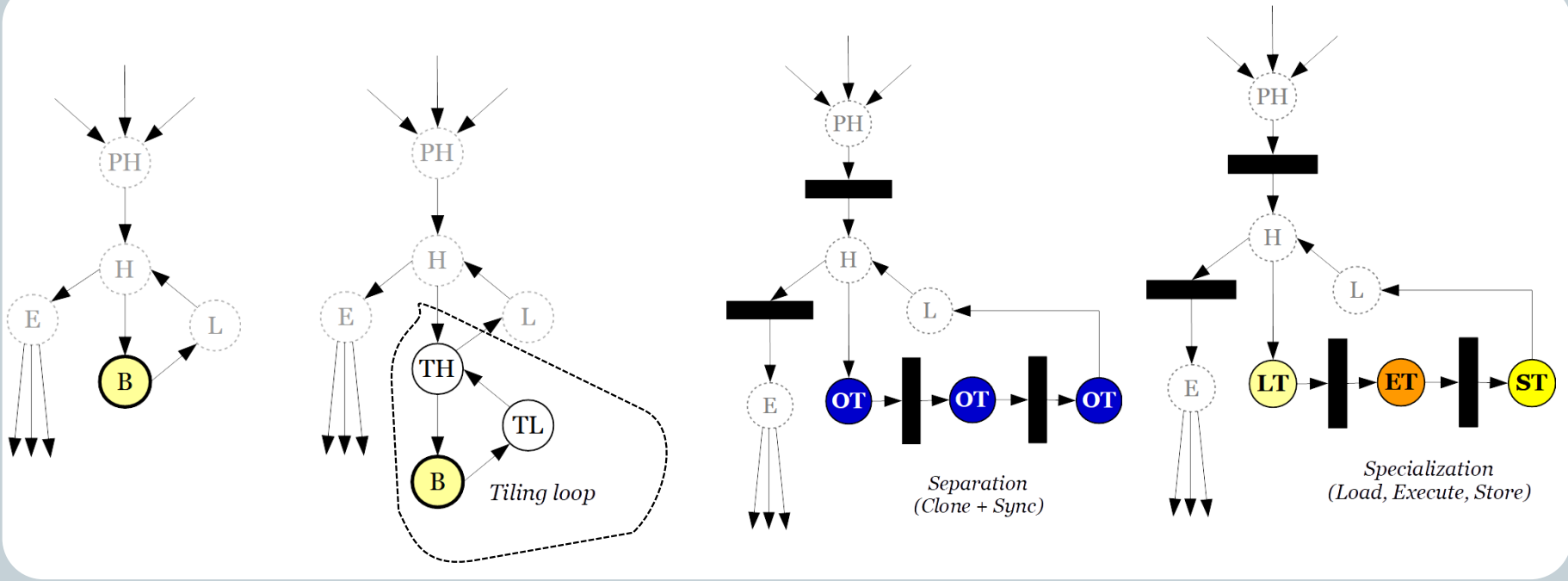
```
FORALL i = 1, Ni
 FORALL j = 1, Nj
  FOR k = 1, WS
   FOR l = 1, WS
    S1
   END FOR
  END FOR
 END FORALL
END FORALL
```



So far the main transformation applied

Clang OpenMP Compilation

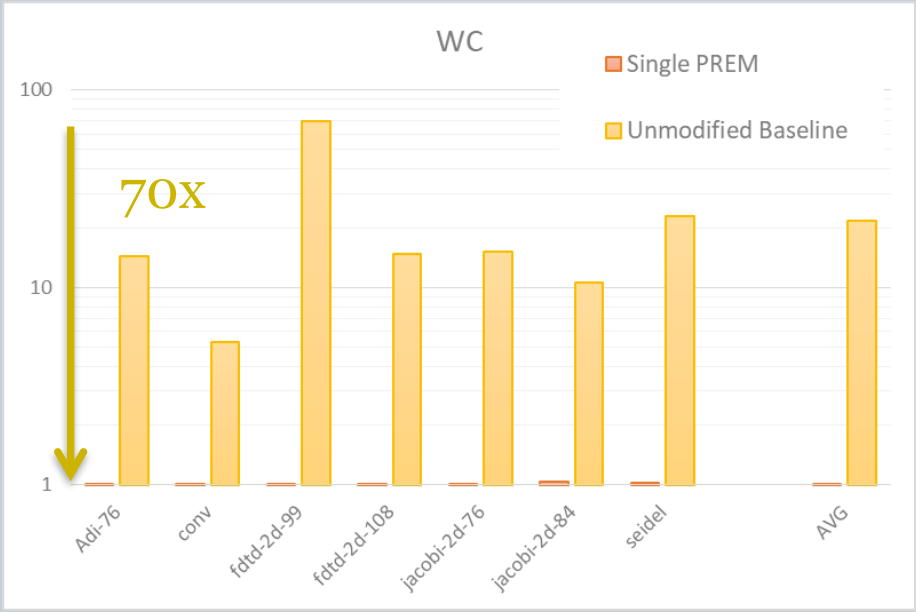
Loop tiling → **Loop outlining** → **Separation** → **Specialization**

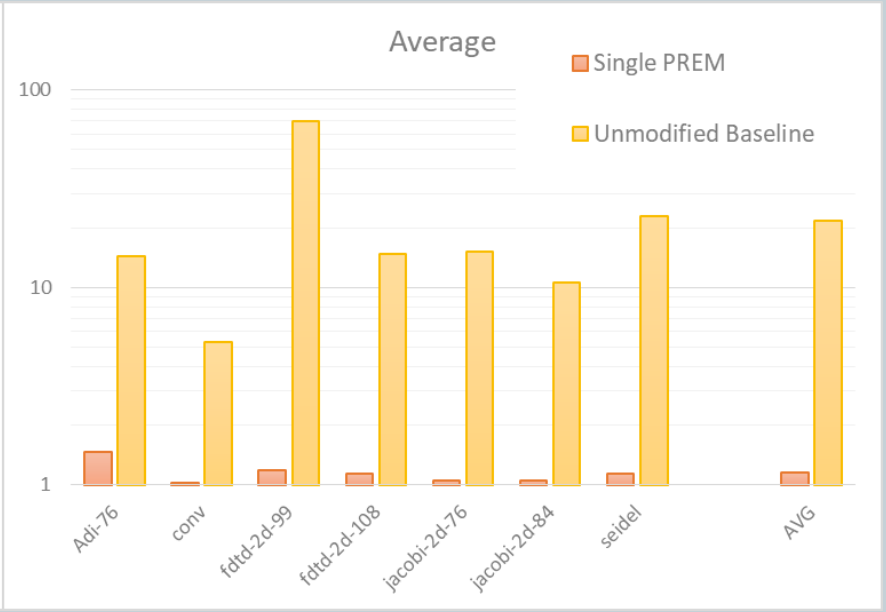


EVALUATION

PREDICTABILITY (PolyBench)

1. Near-zero variance when sizing PREM periods for the worst case
2. Up to 70x variance reduction w.r.t. baseline!!!
3. What's the performance drop?





Execution time with interference/Execution time with no interference

EVALUATION

PREDICTABILITY vs PERFORMANCE (PolyBench)

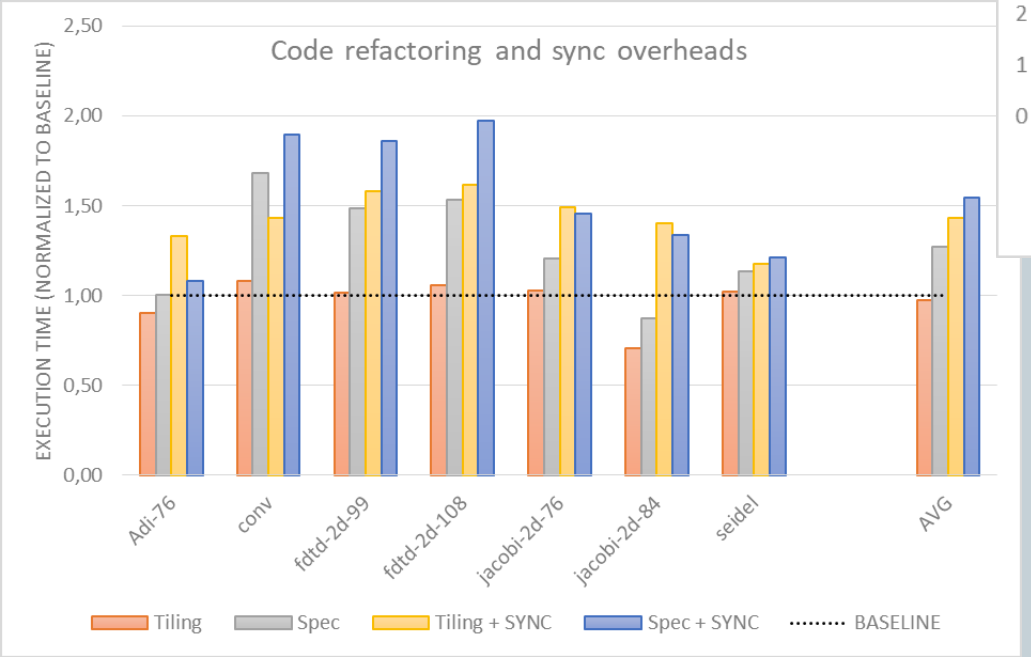
1. Up to 6x slowdown wrt unmodified program w/o interference
2. Up to 11x improvement wrt unmodified program w interference

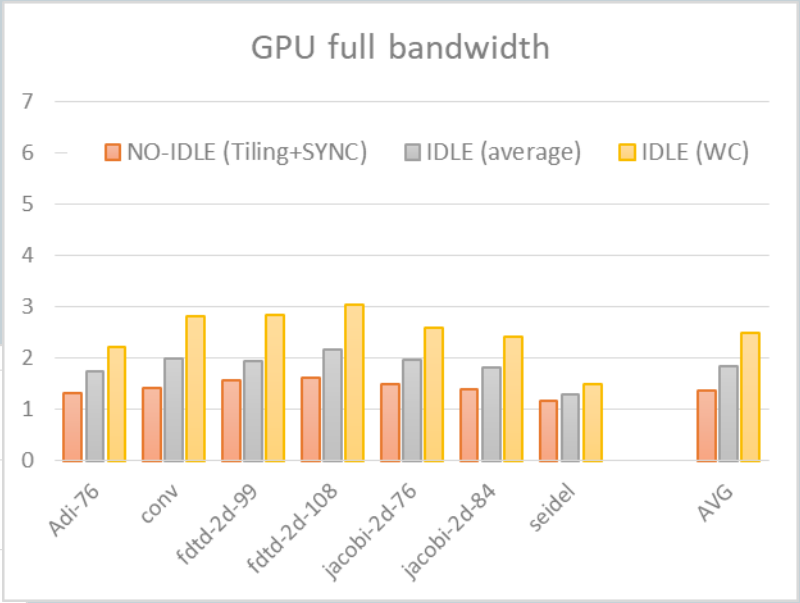
Execution Time normalized w.r.t. unmodified baseline execution w/o iterference

EVALUATION

OVERHEADS (1)

1. Code refactoring
2. Synchronization

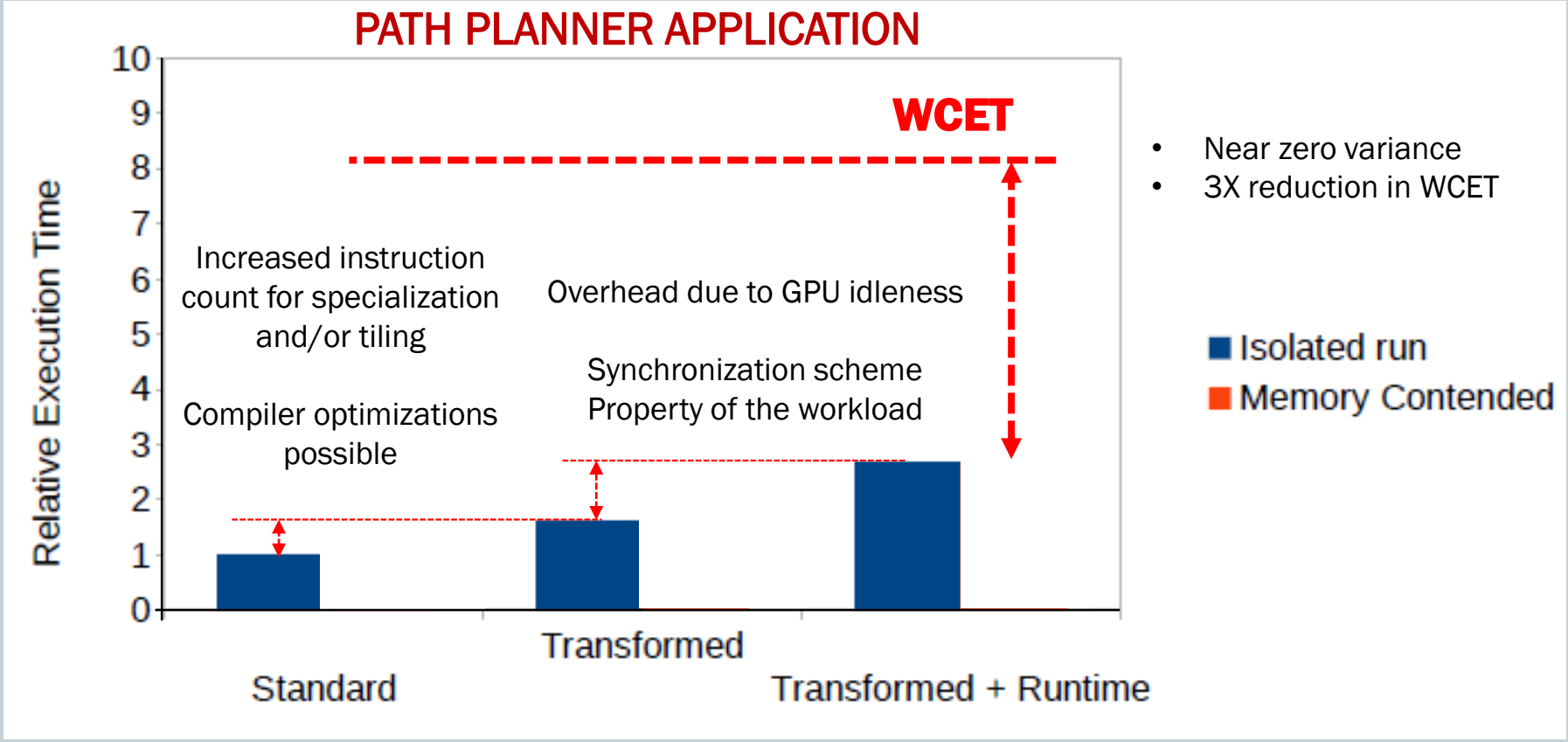




OVERHEADS (2)

1. GPU idleness

EVALUATION



- Near zero variance
- 3X reduction in WCET

Björn Forsberg, Daniele Palossi, Andrea Marongiu, Luca Benini:
GPU-Accelerated Real-Time Path Planning and the Predictable Execution Model. **ICCS 2017**

WORK IN PROGRESS…

- Understanding key functionality/performance issues
- Extensively characterizing performance and overheads on real benchmarks
- PREMization of CPU code
  - Extend scope of analysis to more control-oriented codes
- Extend the metodology to All Programmable SoC (CPU + FPGA)

END!

DAVIDE ROSSI

ALESSANDRO CAPOTONDI
GIUSEPPE TAGLIAVINI
ANDREA MARONGIU

Università Di Bologna

davide.rossi@unibo.it