Paradigm Shift for EXASCALE Computing

Lets go Parallel

Skevos (Paraskevas) Evripidou skevos@cs.ucy.ac.cy
Dept. of Computer Science
University of Cyprus

Costas Kyriakou eng.kc@fit.ac.cy
Dept of Computer Science and Engineering
Frederic University, Cyprus

EASC 2015 Edinburgh, UK, 21st-23rd April 2015

Lets go Parallel !

- The end of the exponential growth (Dennard Scaling) of the sequential processors has facilitated the development of multicore systems.
- Thus, any growth in performance must come from Parallelism
- However, commercial microprocessor are still using sequential cores.
- A hybrid Processor based on Data-flow (Data-Driven) Scheduling and control-flow execution can be low power and low complexity.
  - Could use some of the savings to include more DRAM.

Paradigm shift for HPC/Exascale

- The current thinking for building High Performance Computing (HPC) Systems, is currently based on the sequential computing (von Neumann model) with the addition of some parallel constructs (MPI,OpenMP…).
  - Critical sections, Barriers, cache coherence etc
- The switch to Multi-core chips has brought the Parallel Processing into the mainstream.
- Does HPC also needs a more aggressive paradigm change?
- We are proposing a paradigm shift to a Novel Hybird system based on the Dynamic Data-Flow model of computation for Synchronization and the Control Flow for efficient execution.

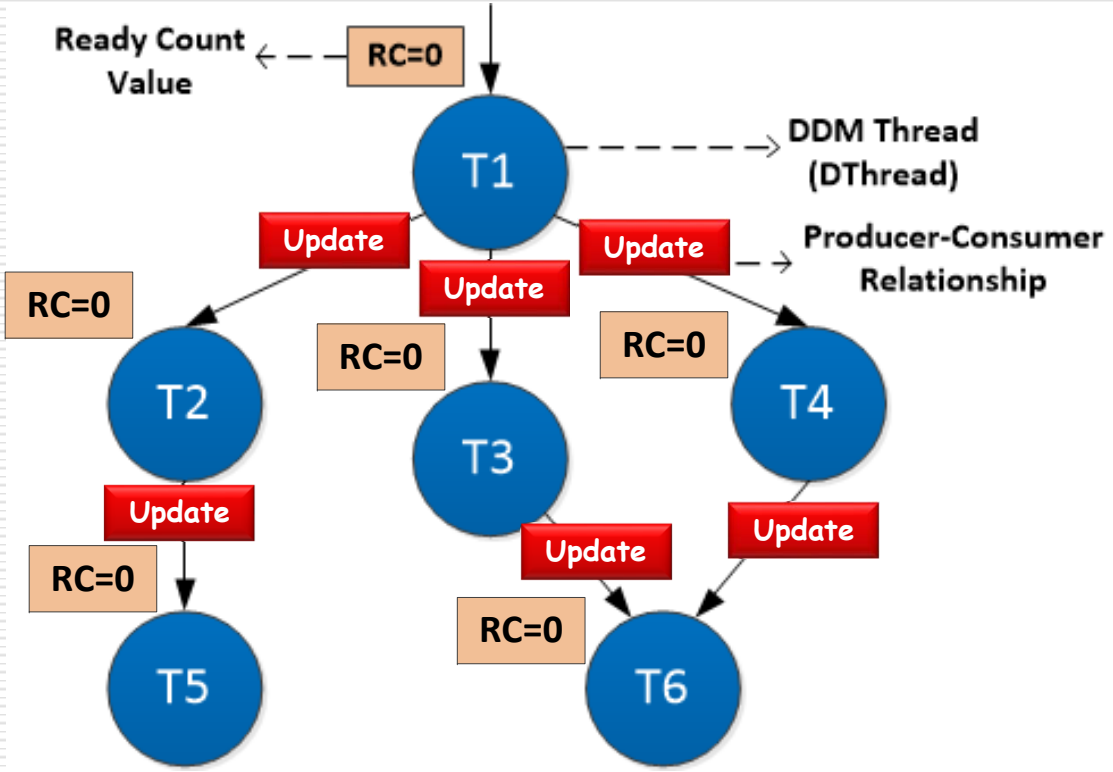
Data-Driven Scheduling (DDS)

- Data-driven scheduling enforces only a partial ordering as dictated by the true data-dependencies which is the minimum synchronization possible.
  - This is very beneficial for parallel processing because it enables it to **exploit the maximum possible parallelism**.
- DDS has been implemented in the Data-Driven Multithreading[1] in the form of Software systems with very good results.
- An eight-core hardware prototype, with Data-Driven Scheduling has been developed in the TERAFLUX project [2] (pp 18-23)

[1] Kyriacou et al. Data-Driven Multithreading Using Conventional Microprocessors 10.1109/TPDS.2006.136
[2] http://teraflux.eu/sites/default/files/TERAFLUX-D64-v10.pdf

Dependency Graph for Data Driven Scheduling

- Producer-Consumer relationships
- A thread can be scheduled for execution only when all of its producers have completed their execution (RC=0)

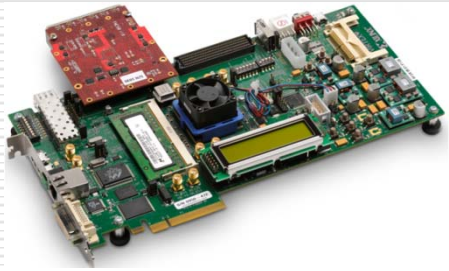


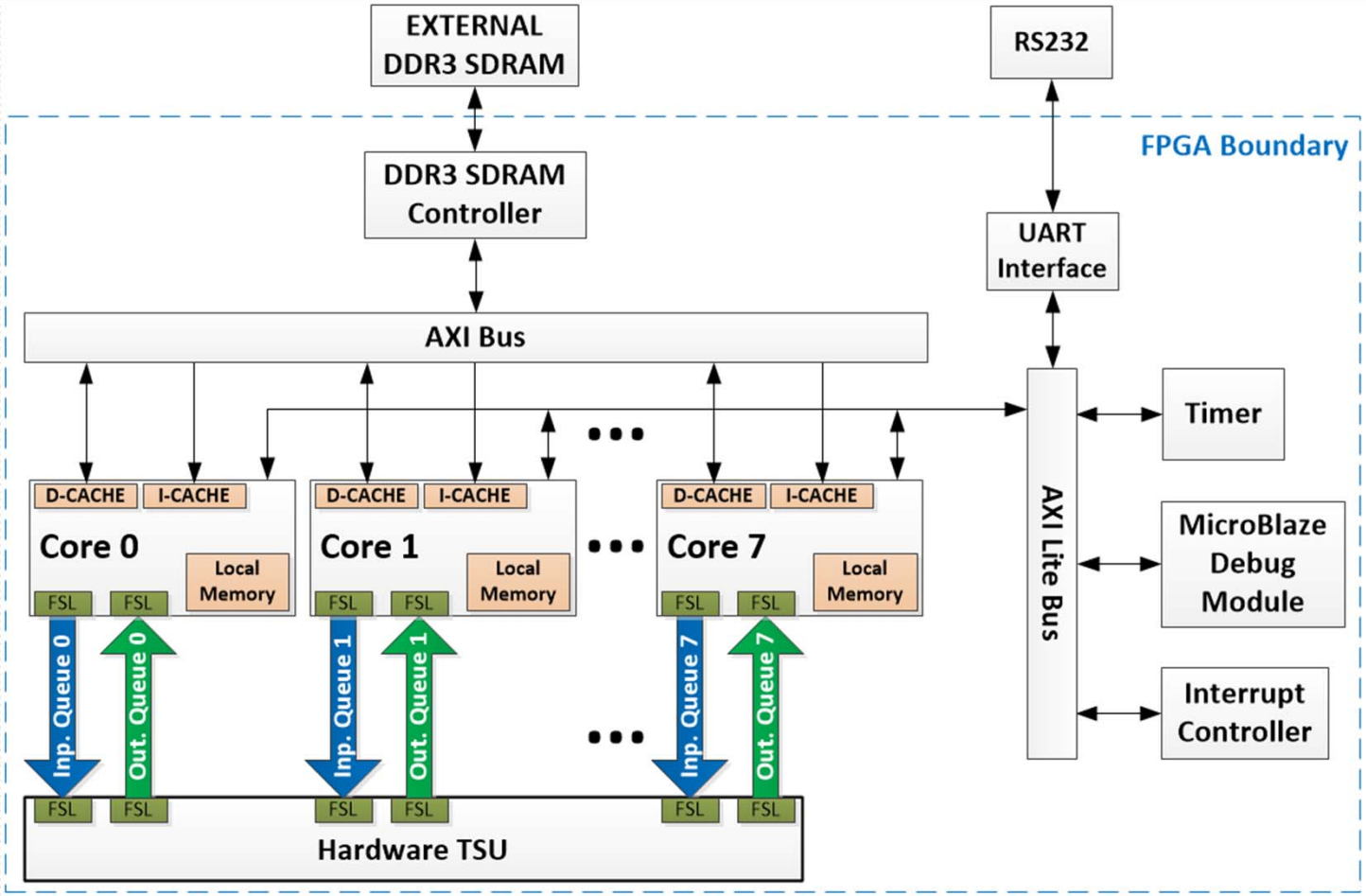
Processor with Data Driven Scheduling

- A thread can be scheduled for execution:
  1. Only after all its producers finish their execution and
  2. Move its input data.
  3. It is possible to have deterministic prefetching of the data in cache or Scratchpad Memory.
- Enables low power and low complexity system.
  - With DDS there no need for
    - Instruction level parallelism (ILP)
    - Out of Order execution (OOE)
    - Cache Coherency

Is this possible?

- In TERAFLUX project we have developed a Multicore system with 8-cores and a Thread Synchronization Unit that supports Data-Driven scheduling





Our Motivation: Quotations from Peter Kogge

Our Motivation comes from some of the observations of Peter Kogge, the leader of the DARPA/USA study group for exascale computing. We quote verbatim from his article Next Generation Supercomputers [3] some key observations:

1. *"The practical exaflops-class supercomputer DARPA was hoping for just wasn't going to be attainable by 2015. In fact, it might not be possible any-time in the foreseeable future and the biggest obstacle to that by far is power."*
2. *"So even our Sobering 67 MW power estimate was overly optimistic. A later study indicated that actual power would be more like 500 MW."*

[3]http://spectrum.ieee.org/computing/hardware/nextgeneration-supercomputers8

Quotations from Peter Kogge -2-

3. "*Realistic applications running on todays supercomputers typically use only 5 to 10 percent of the machines peak processing power at any given time. Most of the other processor cores are just treading water, perhaps waiting for data they need to perform their next calculation. It has been proven impossible to keep larger fraction of the processors working on calculations that are directly relevant to the applications.*
4. *And as the number of processor skyrockets, the fraction you can keep busy at any given can be expected to plummets."*

Data-Flow reply was given back in the future of 1980's!

Arvind and Iannucci, Data-Flow proponents, have been warning us since the 1980's about the two fundamental issues in Multiprocessing: *"long memory latencies and waits due to synchronization events"*. [4]

- In a series of articles they were warning us that the bigger the machine the bigger the problem with latencies.
- It seems that the observation of P. Koggee confirm that the communication and synchronization latency of the sequential model are getting out of hand for HPC/ Exascale machines.
- Data-Flow/Data-driven systems on the other hand have tolerance to communication and synchronization latencies.

[4] Arvind, R.A. Iannucci, Two Fundamental Issues in Multiprocessing. Proceedings of DFVLR - Conference 1987. June 25-26, 1987, Bonn-Bad Godesber.

Quotation from: *Michael Flynn in his keynote speech at PFL 2012*

" *We have multi-threaded, superscalar cores with limited ILP; worse yet, most of the die area (80%) is devoted to two or three levels of cache to support the illusion of sequential model. And the cache organization doesn't work for many data structures…"*

Architectural support for Data-Driven Scheduling (DDS)

We are addressing the challenges for Exascale with architectural and organization techniques the three main challenges:

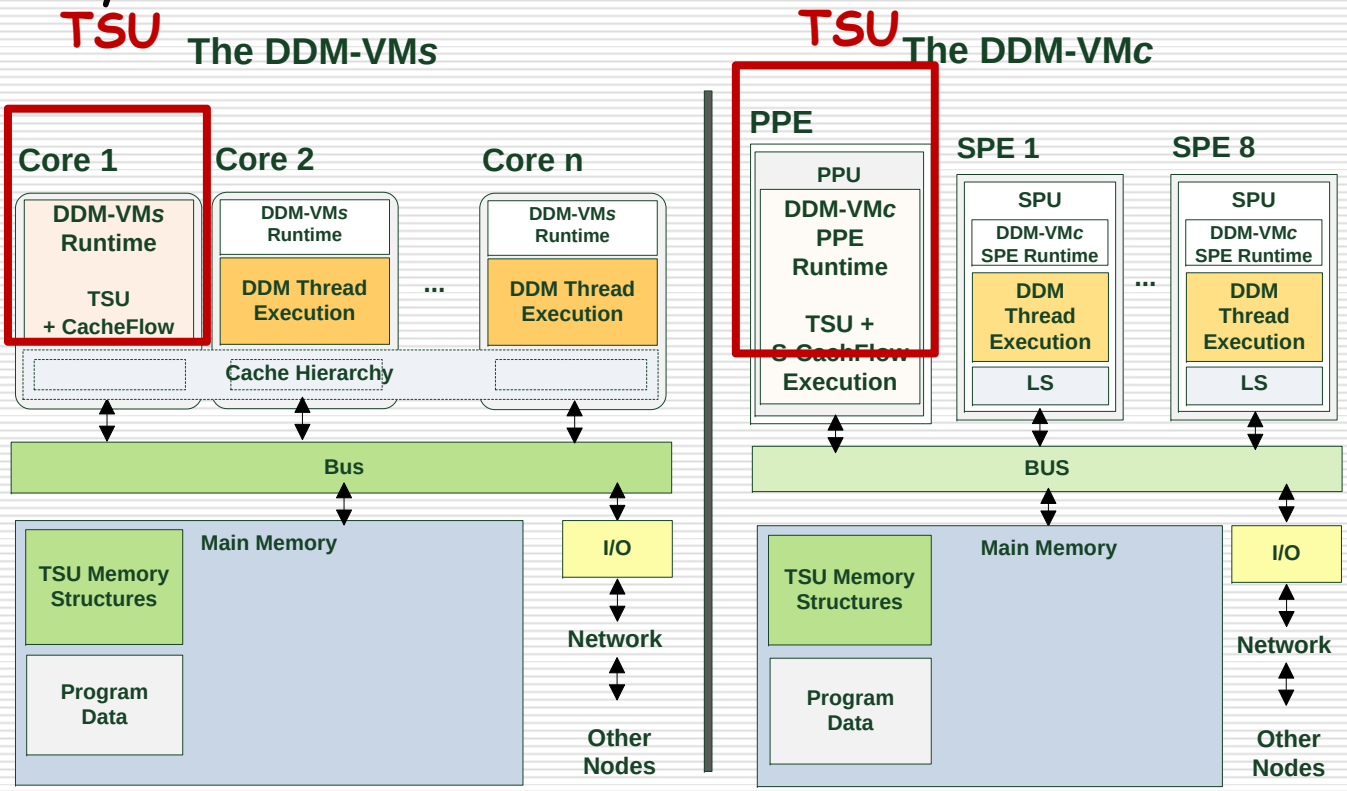
1. Synchronization latencies: The DDS semantics reduce the synchronization latencies to the bare minimum. Thus, increasing the utilization of the machine and at the same time save energy.
2. Memory: The Scratch-pad based lightweight memory hierarchy will reduce the amount of memory (SRAM) used and at the same time reduce power.
   - Furthermore, it will improve locality by scheduling threads for execution in the core that its input data is stored.
3. Power savings: (i) reduce the power by the removing cache hierarchy (up 8Mb) and replace it with Scratch-pad memory and (ii) by removing the unnecessary units like the ILP and OOE .

Data-Driven Mulithreading

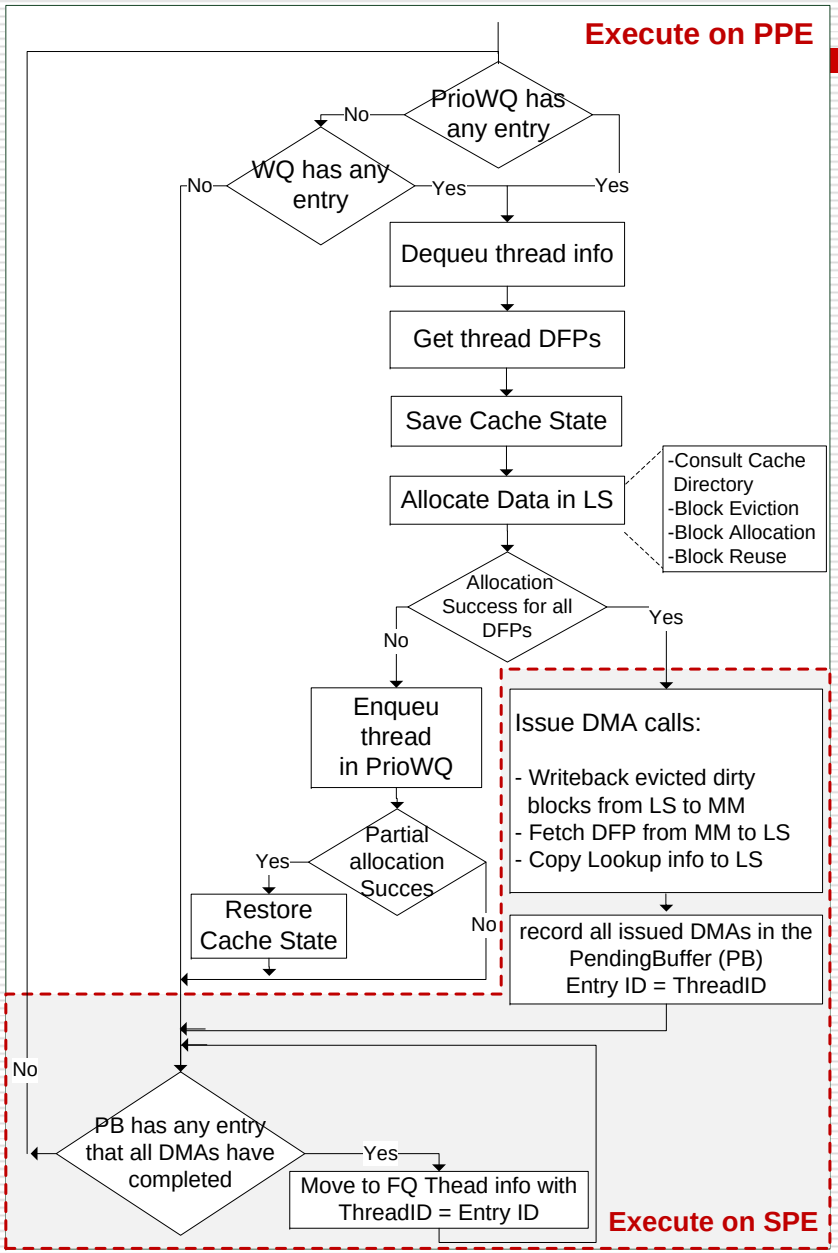
- The Data-Driven Multithreading (DDM) is a non-blocking multithreading model that schedules threads based on data availability on sequential processors.
- A DDM program consists of several threads of instructions, called DThreads, that have producer-consumer relationships.
- For each DThread, the Thread Synchronization Unit collects meta-data that enable the management of the dependencies among the DThreads and determine when a DThread can be scheduled for execution.
- A Dthread is scheduled for execution only when all it consumers have complete execution which guarantees that its input data are available
- The instructions within the DThreads are fetched and executed by the CPU sequentially in a control-flow manner.

The Data-Driven Virtual Machine (DDM-VM)

- The DDM-VM is a virtual machine that supports DDM execution on **homogeneous** and **heterogeneous** multicores
- Virtualizes the **parallel resources** and handles thread scheduling, execution instantiation and data prefetching implicitly

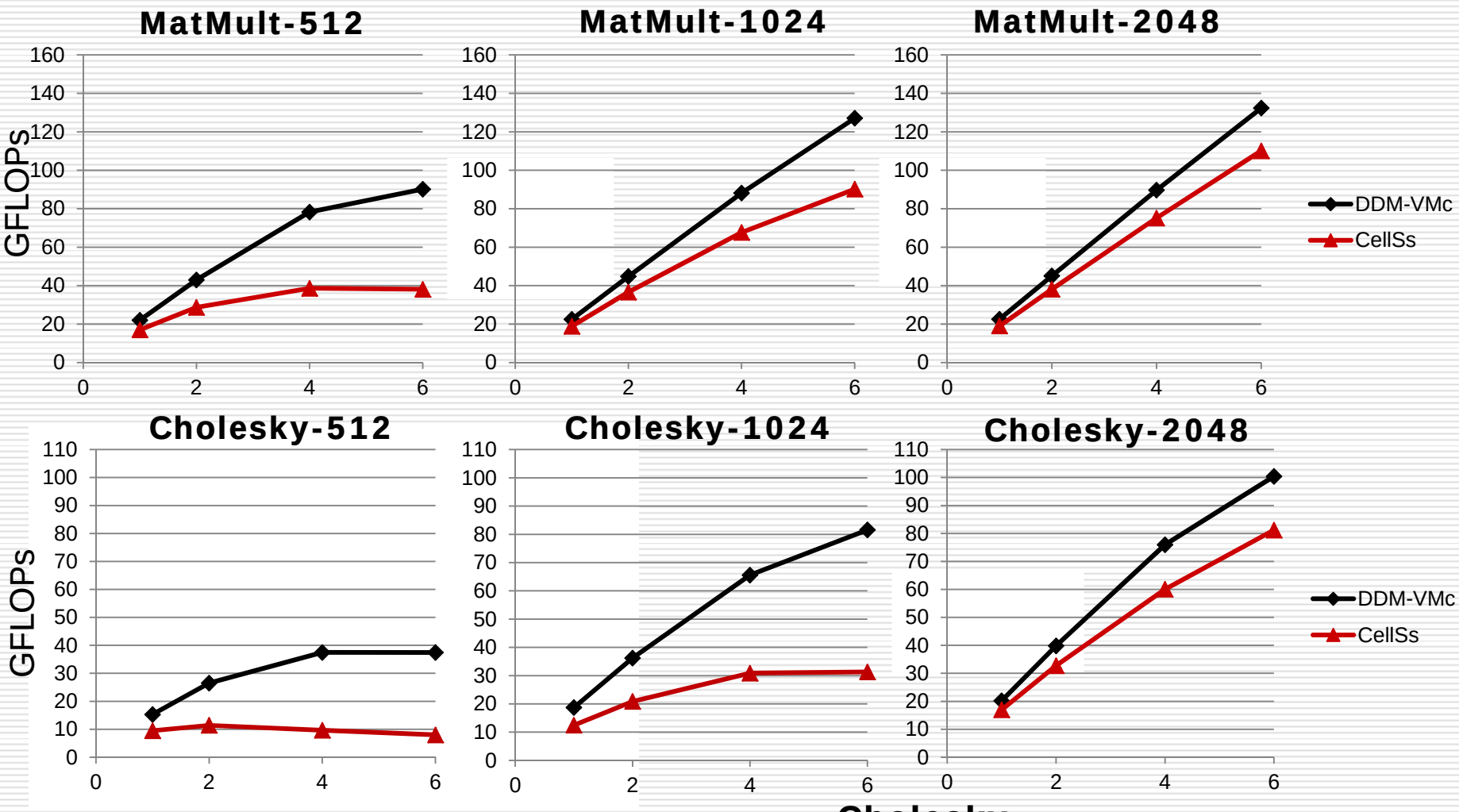


DDM Software CacheFlow for the CELL



- S-CacheFlow an automated and efficient memory management for the Cell
- A portion of the LS is pre-allocated and divided into cache blocks. A Cache Directory keeps the block's state
- S-CacheFlow maintains consistency with DF synchronization
- Overlaps Data transfers and management with thread execution to tolerate latencies and achieve multi-buffering
- Exploits explicit locality

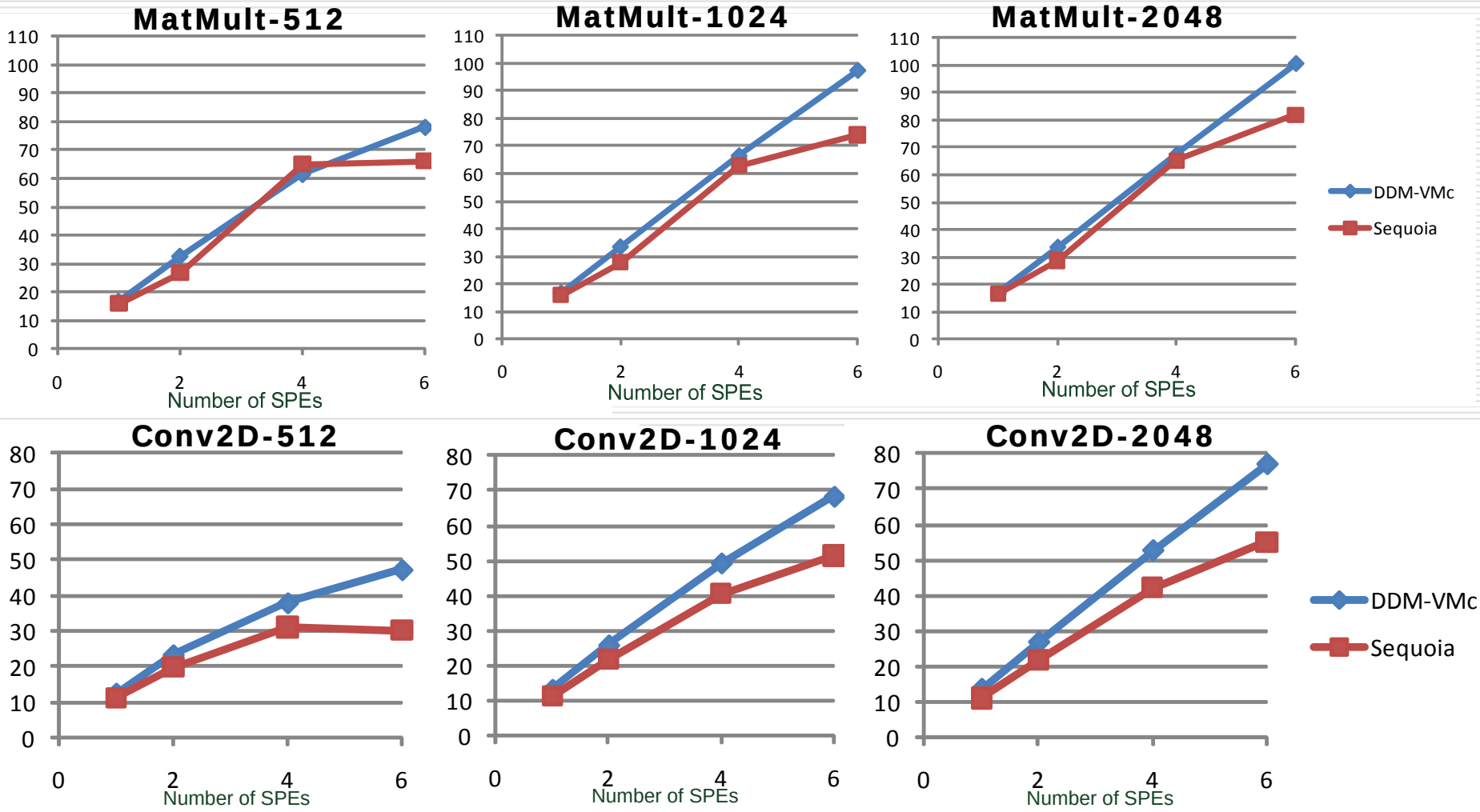
DDM-VMc Comparison with CellSs (BSC) [6]



- **MatMult**
  - An **improvement** of 80%, 28% & 19% for the 512,1024 and 2048 sizes

- **Cholesky**
  - An **improvement** of 213%, 99% & 23% for the 512,1024 and 2048 sizes

❑ Comparison with Sequoia [Stanford] [8]



MatMult

- An **improvement** of 10%, 16% & 11% for the 512,1024 and 2048 sizes

Conv2D (9x9 convolution filter)

- An **improvement** of 27%, 22% & 27% for the 512,1024 and 2048 sizes

[8] Arandi, Evripidou: **DDM-VM$_c$: the data-driven multithreading virtual machine for the cell processor.** HiPEAC 2011

DDM-VMs vs MPI for Cholesky

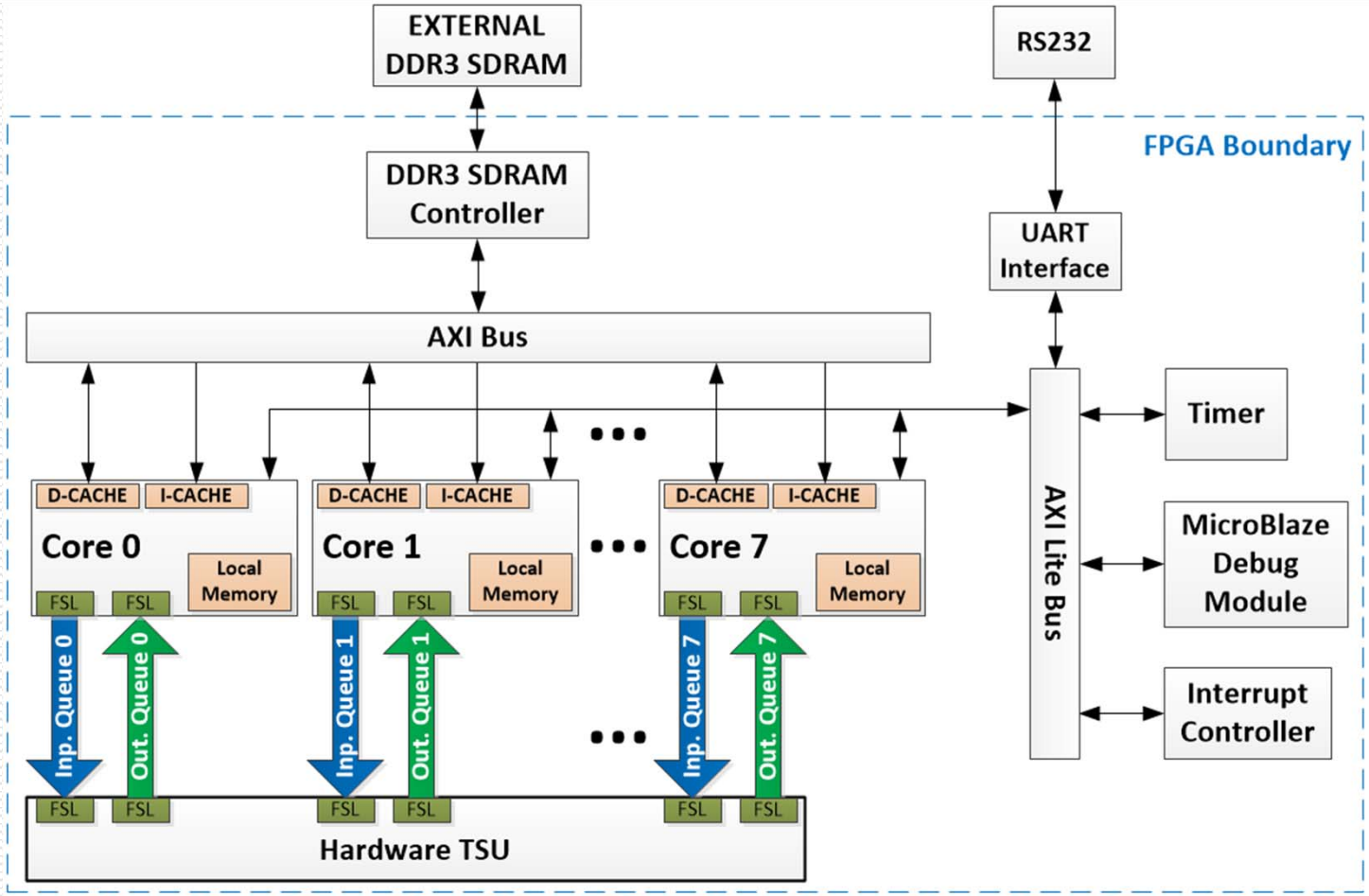
- For this comparison we have used 1, 2 and 4 AMD Opteron 6276 machines with a total of 128 cores
- For 32 cores DDM gets speedup close to 25 and MPI around 11.
- Overall DDM achieves speedup slightly above 115 where the MPI achieves only 4.
- We believe this due to the fact that the Cholesky algorithm has very complex data dependencies that cannot be handled well in the MPI implementation and the use of barriers.

Configuration					*DDM*			*MPI*		
Matrix Size	*Block Size*	*Cores Per Node*	*# of Nodes*	*Total Cores*	*Serial Time*	*Parallel Time*	*Speedup*	*Serial Time*	*Parallel Time*	*Speedup*
4K	64	32	1	32	215.53	8.32	**25.89**	42.90	5.38	**7.97**
8K	64	32	1	32	1065.70	41.84	**25.47**	334.48	30.06	**11.13**
8K	64	32	2	64	971.74	24.56	**39.56**	335.25	306.93	**1.09**
16K	128	32	2	64	13595.94	197.66	**68.78**	2730.15	1360.47	**2.01**
32K	128	32	4	128	105386.21	911.48	**115.62**	21682.60	5432.17	**3.99**

[?] http://teraflux.eu/sites/default/files/TERAFLUX-D64-v10.pdf

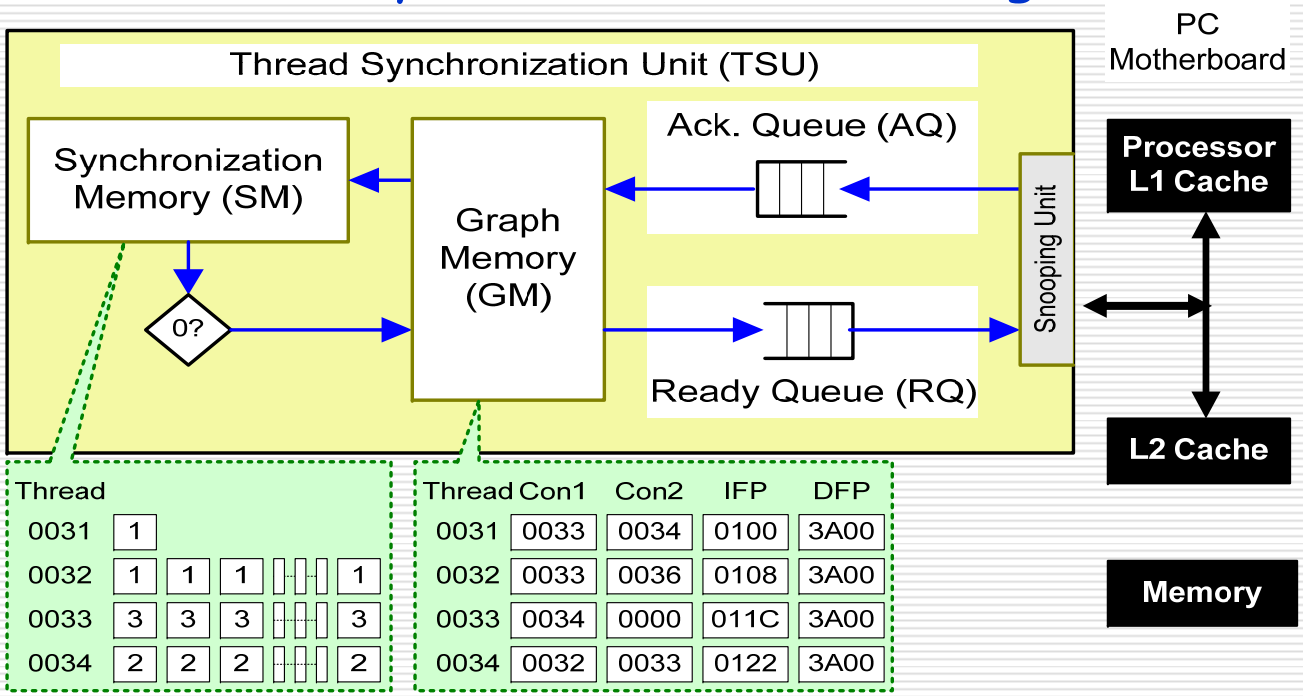
DDM Multi-core with TSU and caches

- In TERAFLUX project we have developed a Multicore system with 8-cores and a Thread Synchronization Unit that supports Data-Driven scheduling

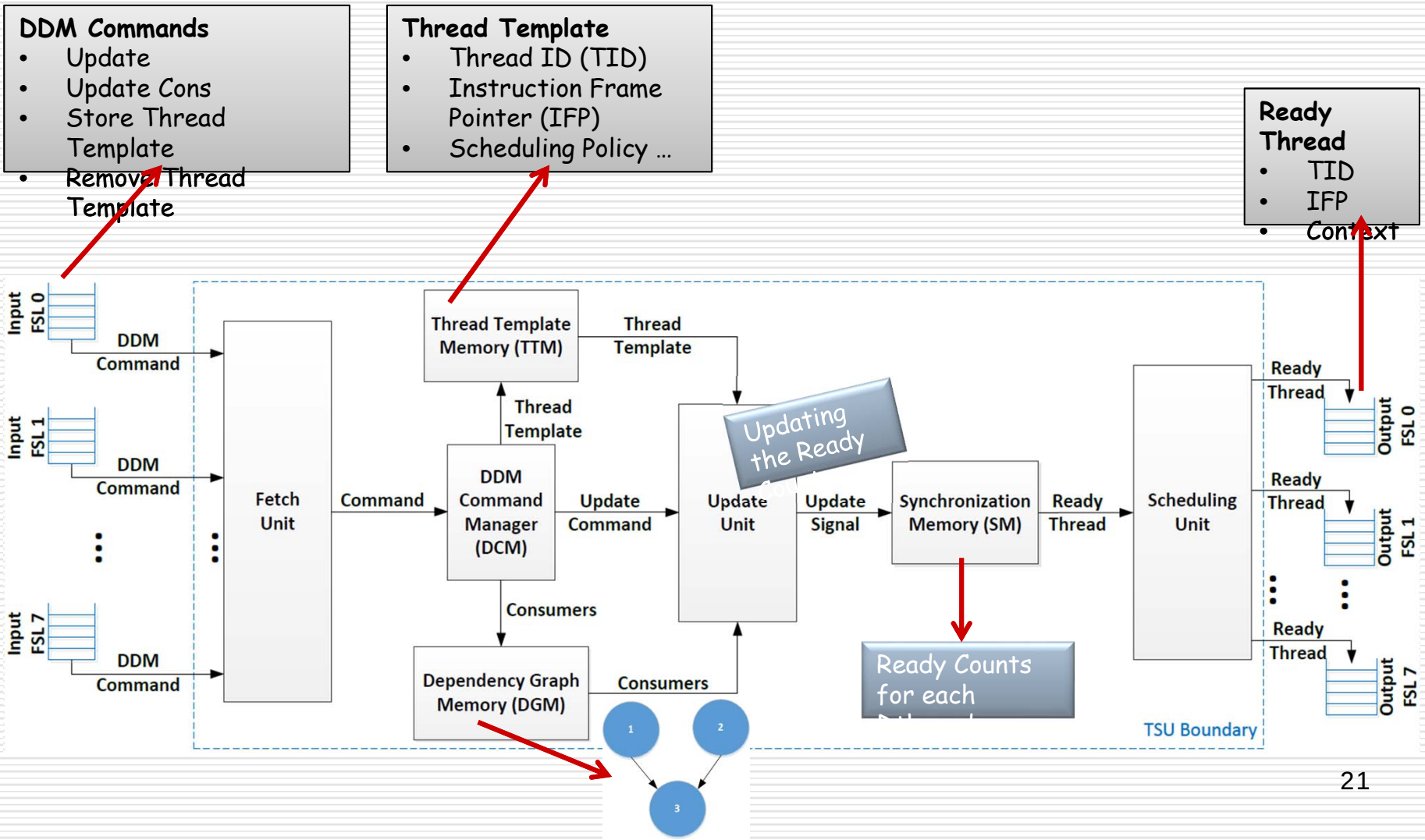


Thread Synchronization Unit (TSU)

- The DDM TSU provides the functionality of Dynamic Data-Flow concurrency on sequential Processors
  - Graph Memory (GM):Producer consumer Thread dependencies
  - Synchronization Memory (SM): Number of Producers
  - Ready (RQ) and Acknowledgment (AQ) Queues

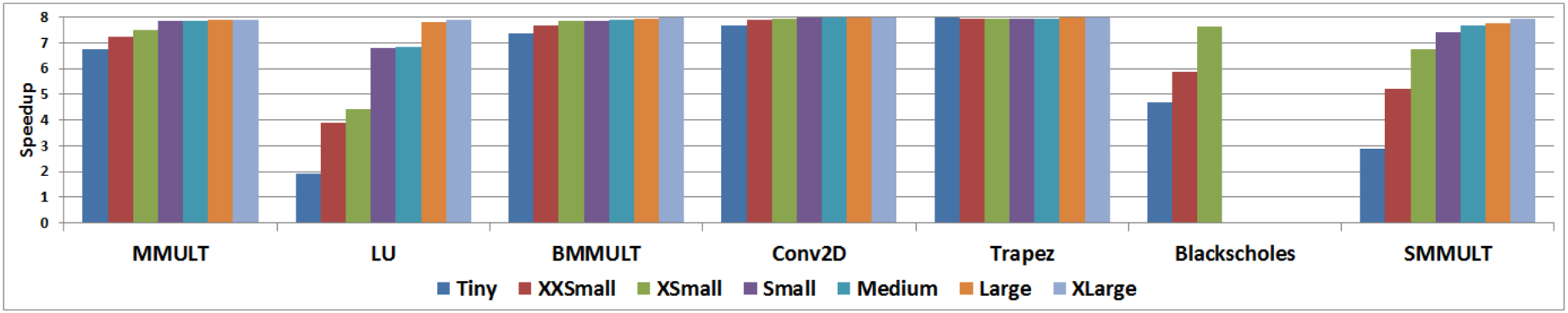


TSU: Dynamic Hardware Support for DDM



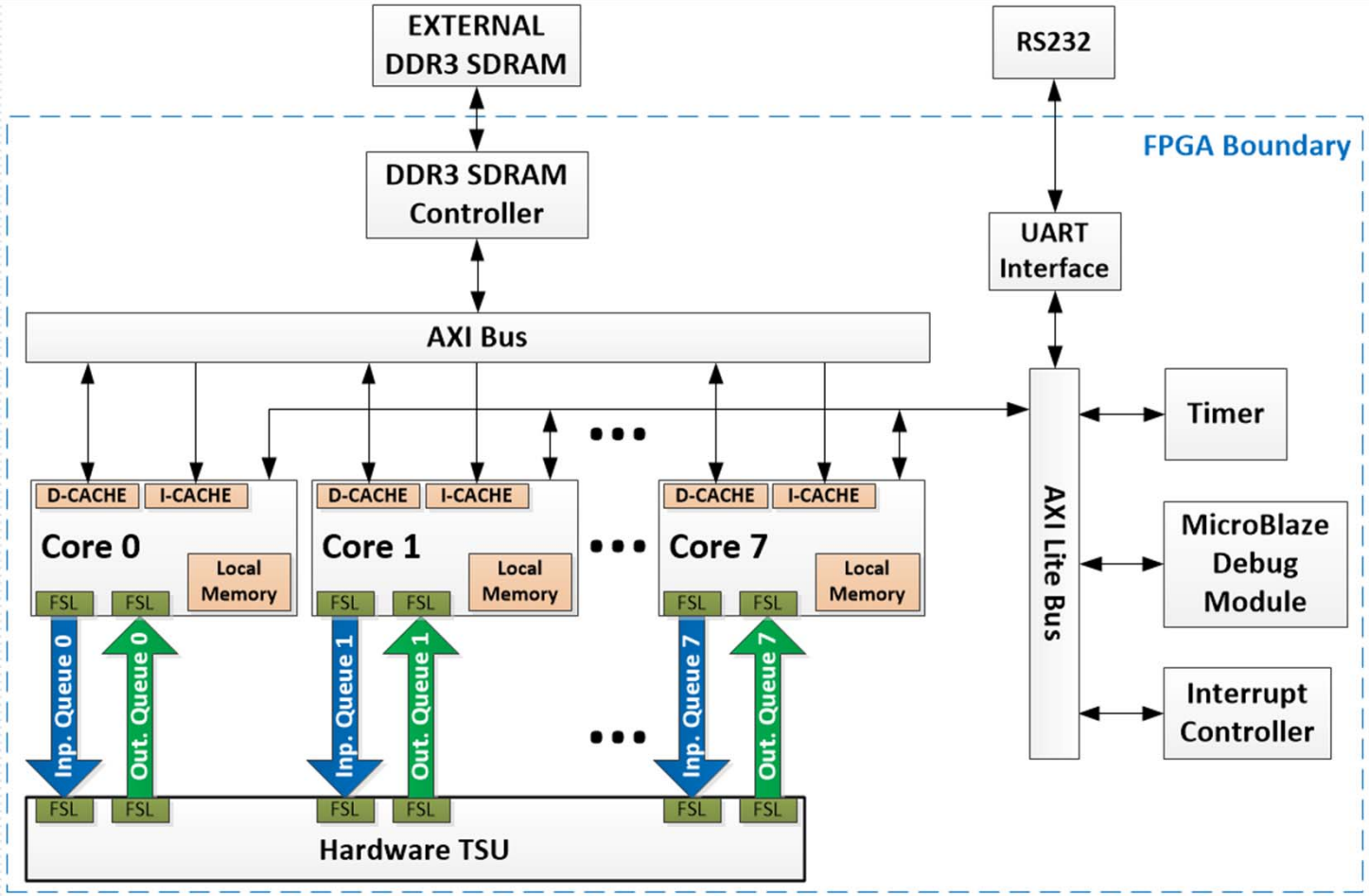
Performance Evaluation

Benchmark	Description	Source	Problem Size						
			Tiny	XXSmall	XSmall	Small	Medium	Large	XLarge
MMULT	Matrix Multiplication	Kernel	16x16	32x32	64x64	128x128	256x256	512x512	1024x1024
BMMULT	Blocked Matrix Multiplication	Kernel	16x16	32x32	64x64	128x128	256x256	512x512	1024x1024
LU	Blocked LU Decomposition	SPLASH-2	16x16	32x32	64x64	128x128	256x256	512x512	1024x1024
Conv2D	9x9 convolution filter	Kernel	16x16	32x32	64x64	128x128	256x256	512x512	1024x1024
Trapez	Trapezoidal rule for integration	Kernel	2^{16} steps	2^{18} steps	2^{19} steps	2^{20} steps	2^{21} steps	2^{23} steps	2^{25} steps
Blackscholes	Financial analysis application	PARSEC	8 options	12 options	16 options	-	-	-	-
SMMULT	Sparse Matrix Multiplication	Kernel	16x16	32x32	64x64	128x128	256x256	512x512	1024x1024



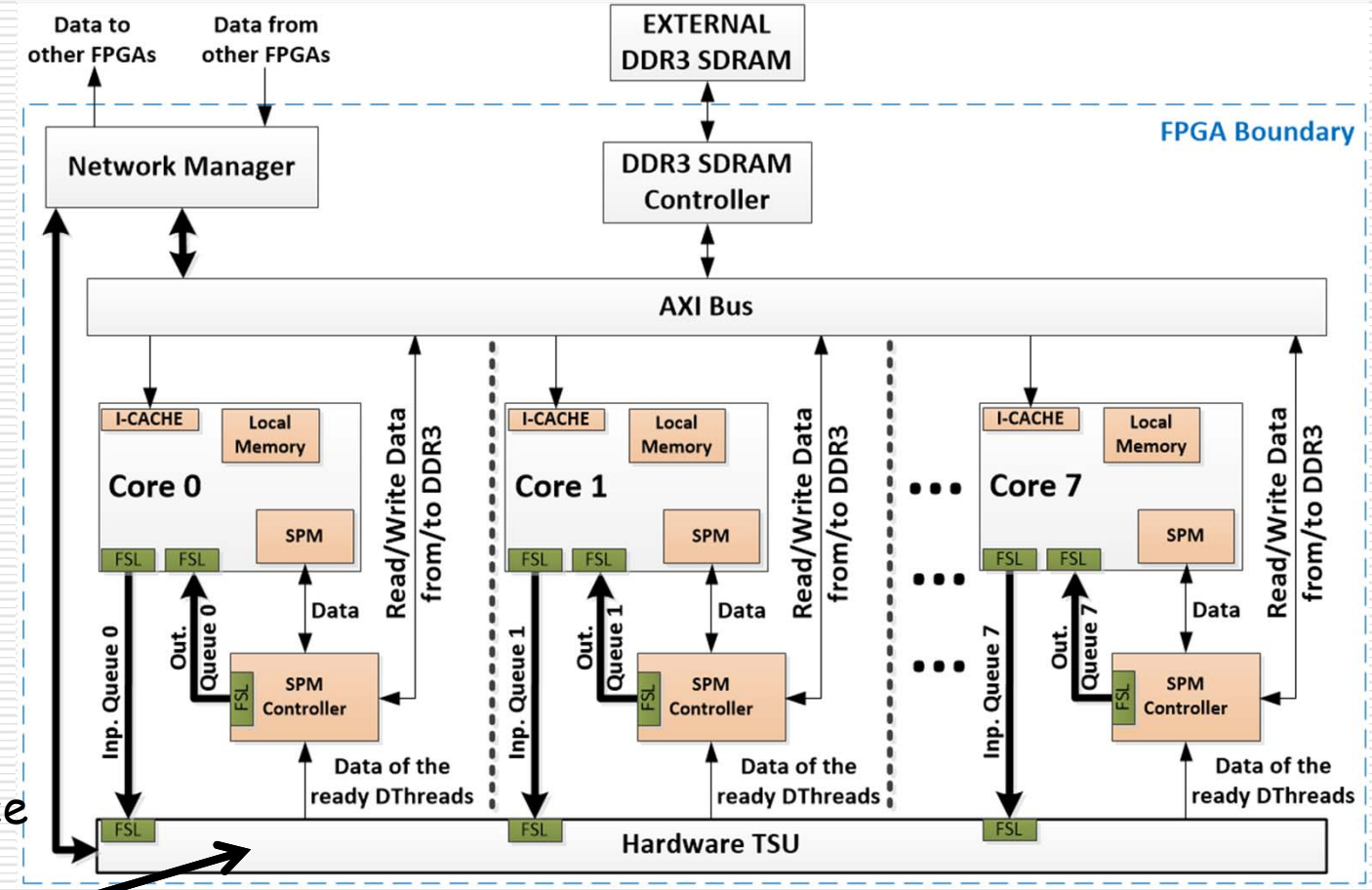
Multi-core with TSU and caches

- In TERAFLUX project we have developed a Multicore system with 8-cores and a Thread Synchronization Unit that supports Data-Driven scheduling



Replace the Cache hierarchy with an automated scratchpad memory

- We have developed such a system in software for the CELL processor[8]



Network Interface Unit (NIU) will be added to the TSU

Conclusions

- The switch to Multi-core chips has brought the Parallelism in the mainstream of computing
  - Will the same happened with HPC?
- We are proposing a paradigm shift to a Novel Hybrid system based on the Dynamic Data-Flow model of computation for Synchronization and the Control Flow for efficient execution.
- A Multithreaded Processor with Data-Driven Scheduling provides a partial ordering of tasks as determine by the true data-dependencies
- DDS enables deterministic prefetching to Scratch-pad memories, thus reducing drastically the SRAM needs

Conclusions -2-

- A thread can be scheduled for execution:
  1. Only after all its producers finish their execution and
  2. Move its input data.
  3. It is possible to have deterministic prefetching of the data in cache or Scratchpad Memory.
- The Data-Driven semantics Enables low power and low complexity system.
  - With DDS there no need for
    - Instruction level parallelism (ILP)
    - Out of Order execution (OOE)
    - Cache Coherency

Conclusions -3-

- Do we need 160 million cores that are only fully utilized only 5-10% at the time and call it EXAFLOP?
  - P. Koggee expects this *to plummets* for EXAFLOP!
  - What if our design with 80 million cores with 10-20% fully utilized
  - What if our design with 40 million cores with 20-40% fully utilized
    - Will this be EXAFLOP?