

DDM-CMP: Data Driven Multithreading on a Chip Multiprocessor

Kyriakos Stavrou
Paraskevas Evripidou
Pedro Trancoso

CASPER group
Department of Computer Science
University Of Cyprus



The *CASPER* group: *Computer Architecture System Performance Evaluation Research*

Outline



- **Motivation**
- Data Driven Multithreading (DDM) model of execution
- DDM-CMP architecture
- Proof of concept
- Experimental Setup
- Experimental Results
- Thermal Aware Scheduling
- Work in progress
- Conclusions



Motivation



- Significant levels of **thread level parallelism** exist in programs
 - Significant speedup potentials
 - Memory wall
 - Synchronization latencies
 - Communication latencies
- Current trend is **increasing the complexity** to improve **performance**
 - Longer Validation and Design Phases
 - Increased power/energy consumption
 - Marginal **performance improvement potential**
- Higher **Temperatures**
 - Decreased lifetime
 - Decreased performance
 - Increased power consumption

Potential Solution:
Data Driven Multithreading (DDM)

Potential Solution:
Chip Multiprocessors (CMP)

Potential Solution:
Thermal Aware Scheduling (TSIC)

Motivation



- DDM-CMP **targets** all the three aspects mentioned in the previous slide
 - Decreases **complexity**
 - Benefits from the available **thread level parallelism**
 - Offers significant potential for **thermal behavior improvement**
- Our preliminary experimental results have shown that DDM-CMP achieves **speedup** ranging **from 5.2 to 14.9** compared to an **equal hardware budget** high-end state-of-the-art single chip microprocessor for the SPLASH2 kernels studied



Outline



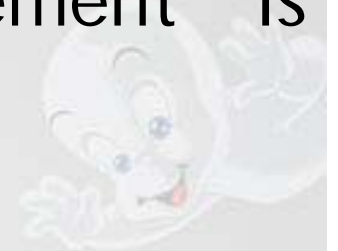
- Motivation
- **Data Driven Multithreading (DDM) model of execution**
- DDM-CMP architecture
- Proof of concept
- Experimental Setup
- Experimental Results
- Thermal Aware Scheduling
- Work in progress
- Conclusions



Data Flow Model



- **Data Driven Multithreading (DDM)** is the **evolution** of the **Data-Flow** model of execution
- **The Data Flow model of execution**
 - Proposed in the **70s** (Jack Dennis - MIT)
 - Execution of *instructions* is driven by data availability
 - An instruction is enabled only when *all* operands are available
 - An instruction can be fired (i.e. executed) only after it becomes enabled
- In the **Data-Flow** model the “basic element” the **instruction** whereas in **DDM** the “basic element” is the **thread**.



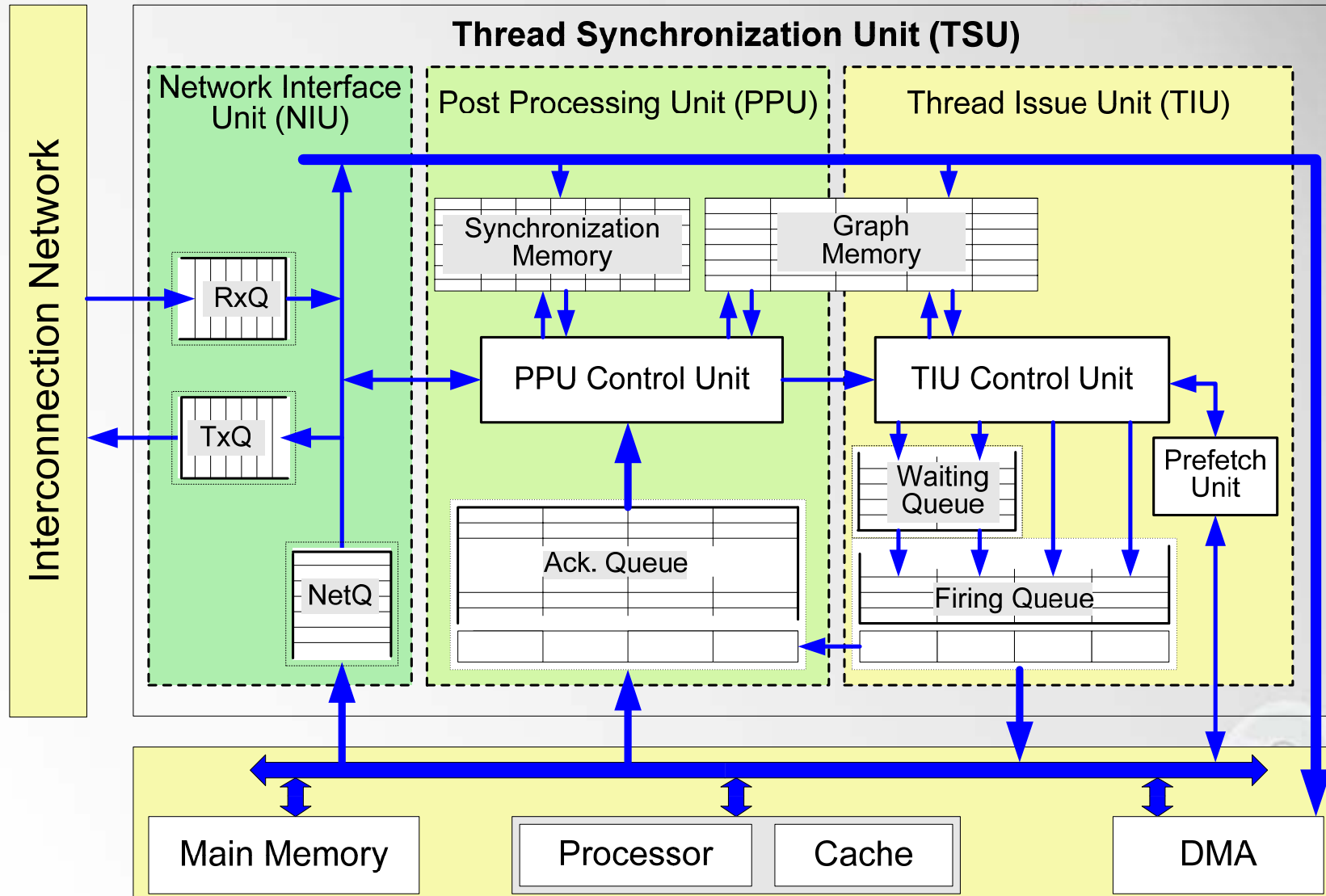
Data Driven Multithreading



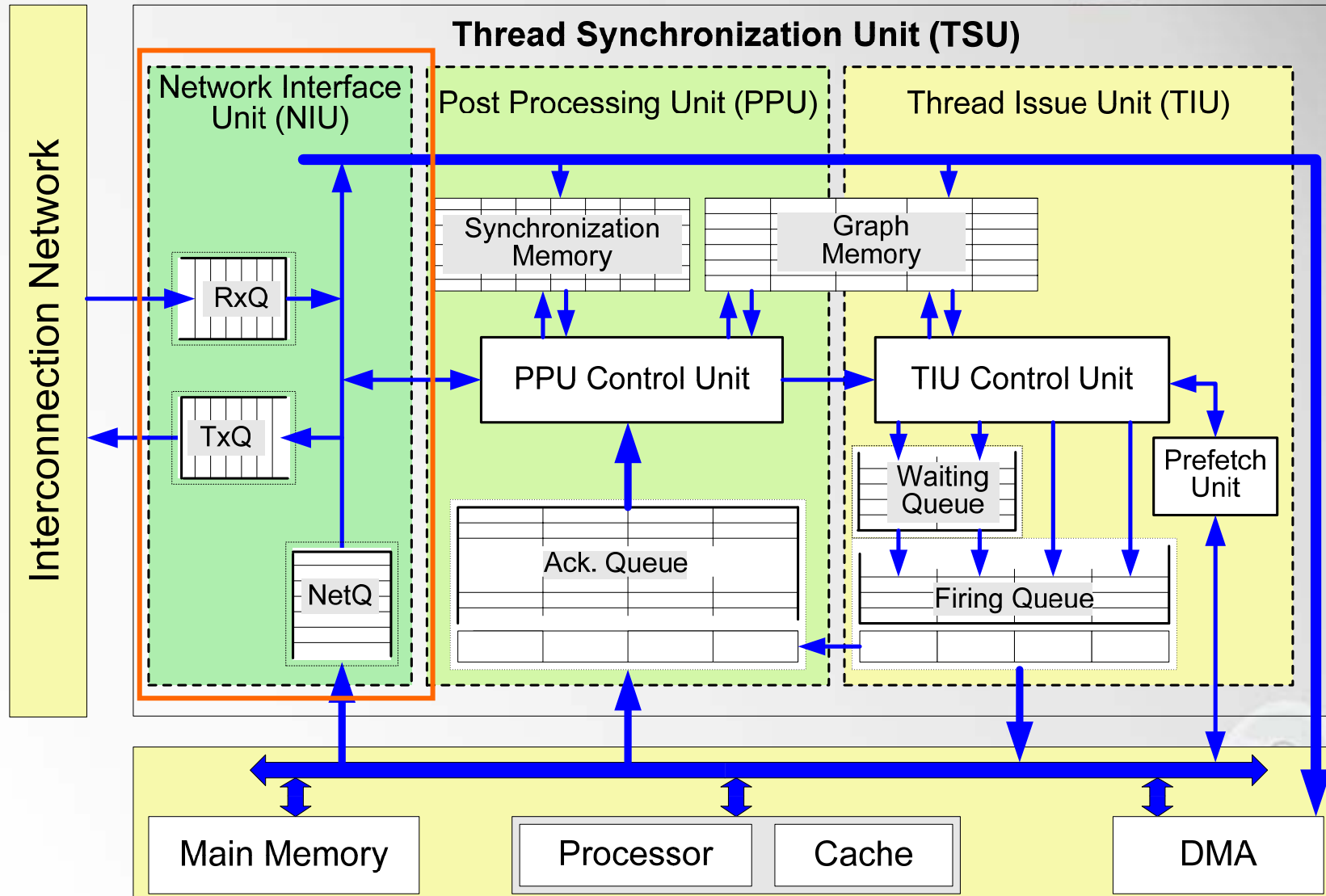
- **Synchronization** part of the program is **separated** from the **computation** part
- **Tolerates synchronization** and **communication latencies** by allowing the computation processor produce **useful work while a long latency event is in progress**
- Efficient **data prefetching** through **Cache-Flow** policies
- Data Driven Multithreading can be **implemented** using **commodity** microprocessors [Kyriacou 05]
 - The only additional requirement is a **small hardware structure**, the **TSU** (Thread Synchronization Unit)



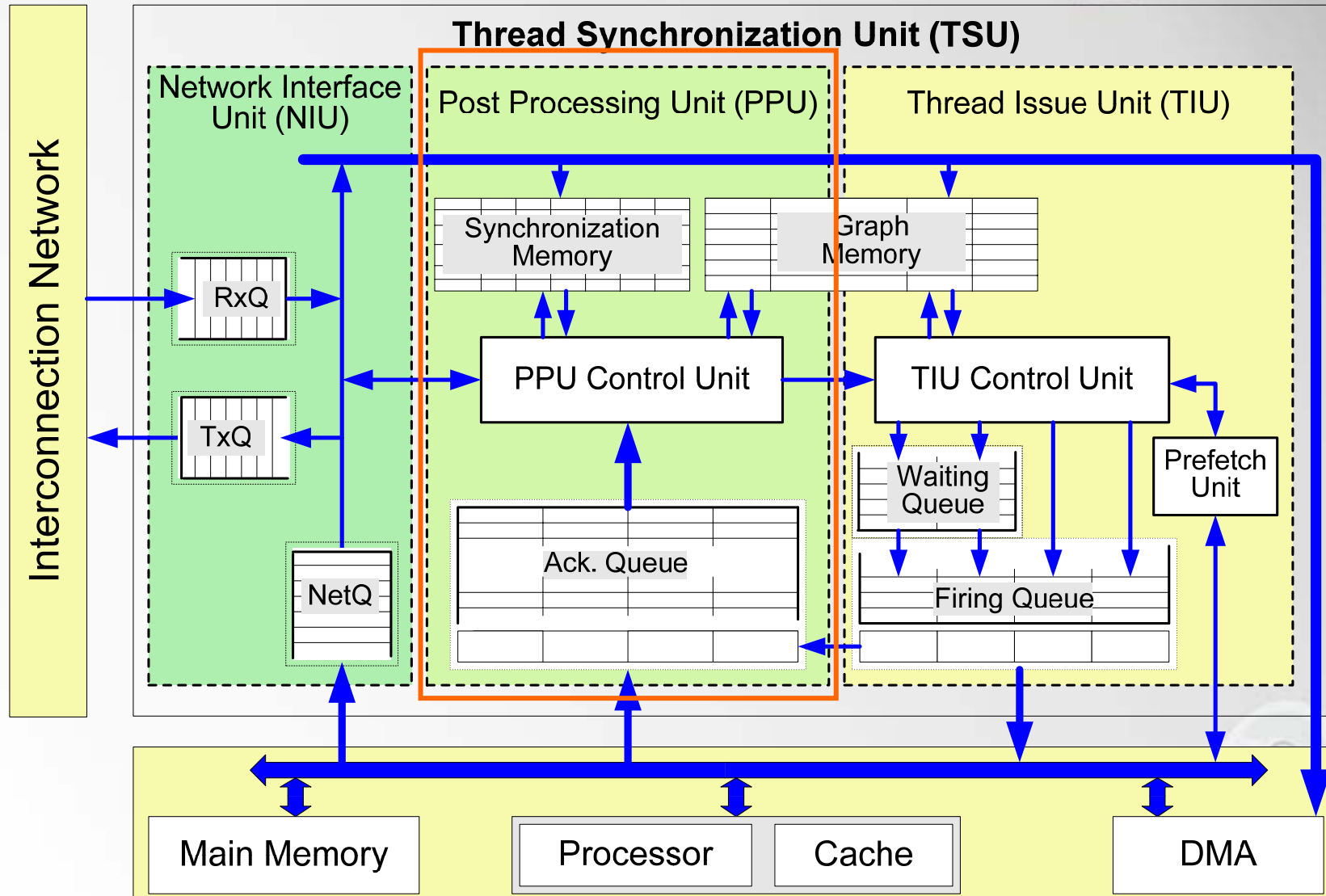
TSU: Hardware support for Data Driven Multithreading



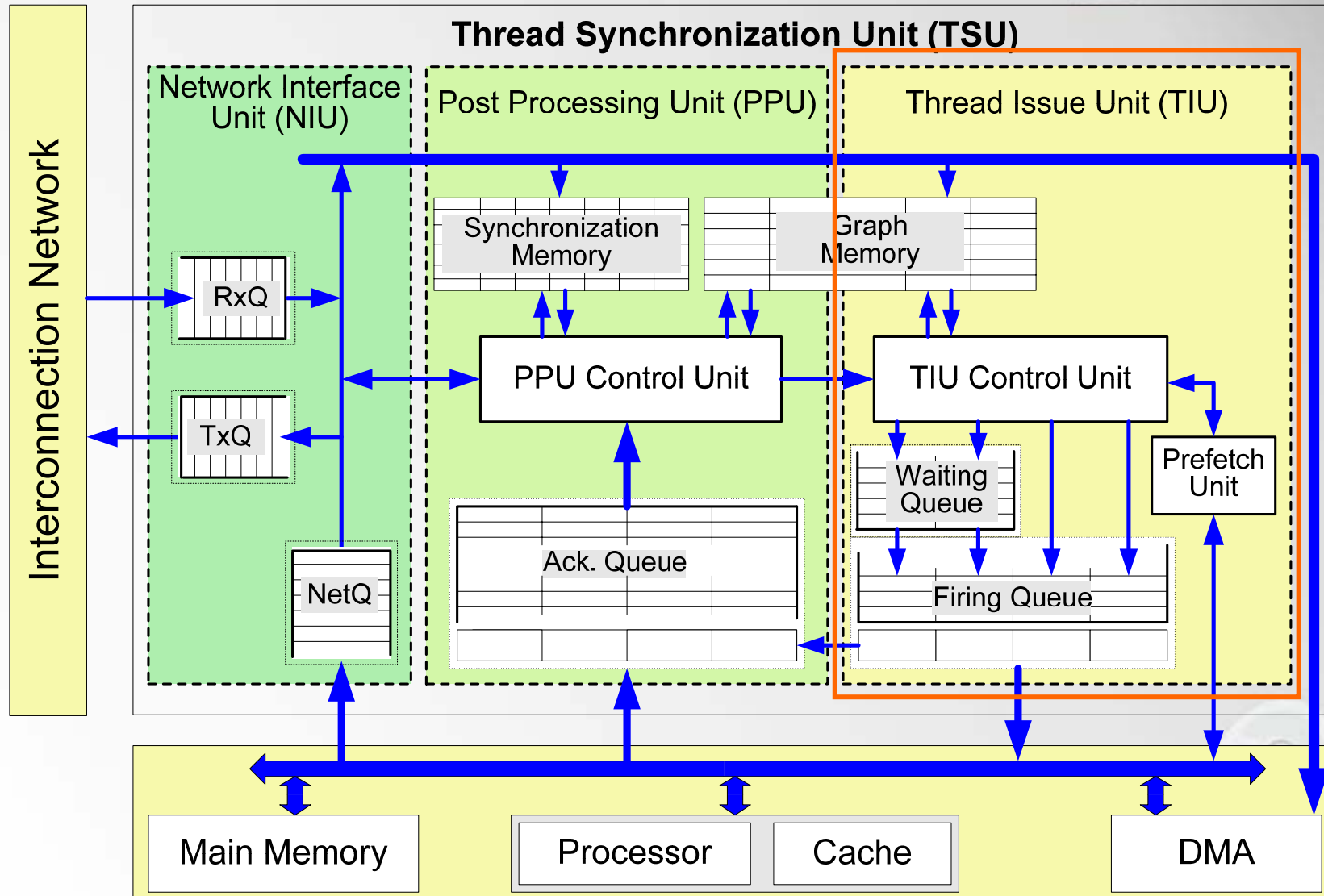
TSU: Hardware support for Data Driven Multithreading



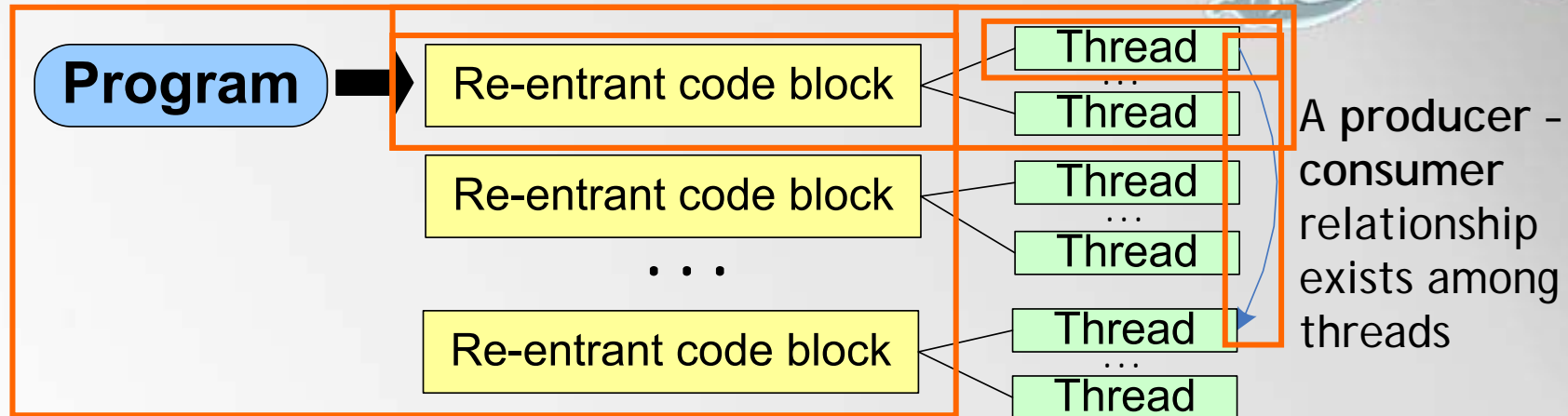
TSU: Hardware support for Data Driven Multithreading



TSU: Hardware support for Data Driven Multithreading



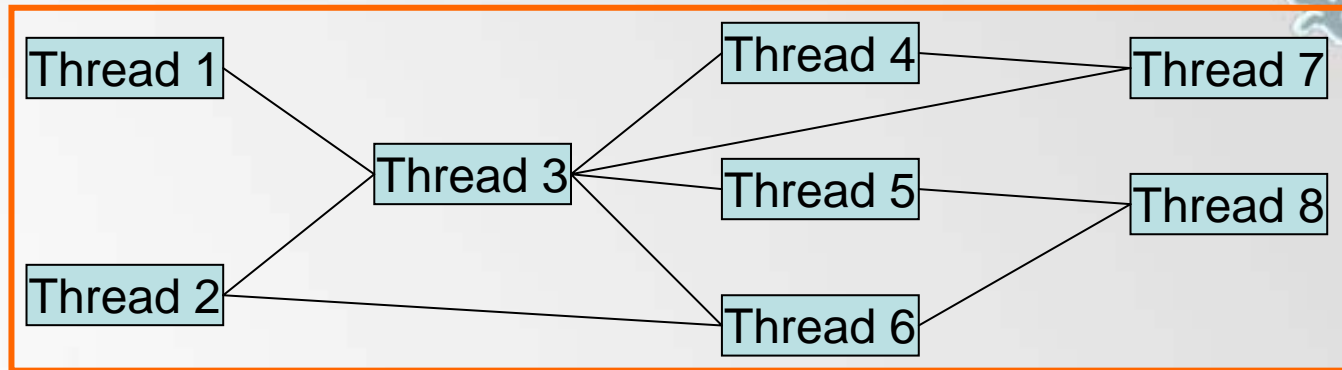
Model of execution



- A **program in DDM** is a **collection of re-entrant code blocks**
 - A **code-block** is **equivalent to a function** or a **loop body** in the high-level context
 - Each **code-block** comprises of **several threads**
 - A **Thread** is a sequence of instructions equivalent to a **basic block**
 - A **producer-consumer relationship** exists among **threads**
 - **Scheduling** of code-blocks and threads is done **dynamically** according to **data availability**



Model of execution

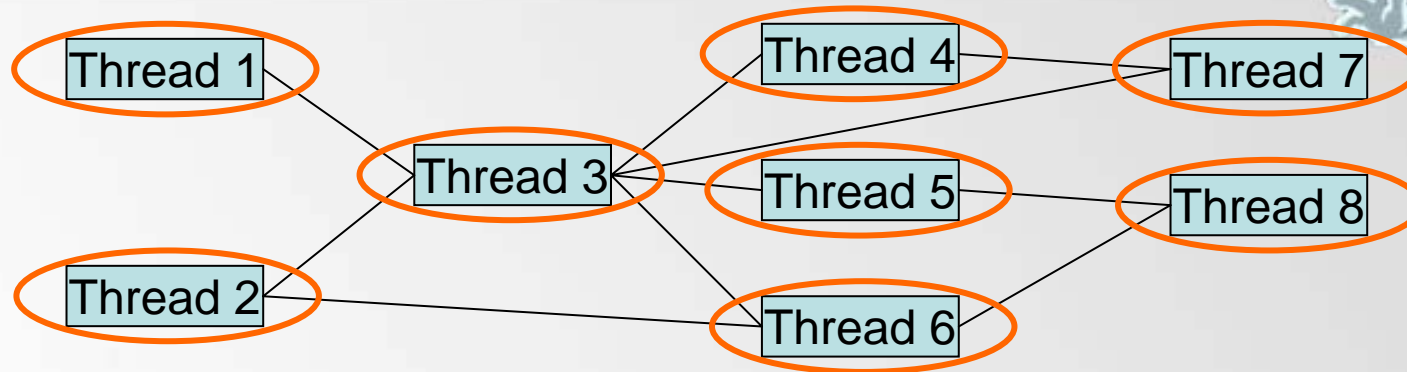


The DDM programs are represented as graphs

- A program in DDM is a **collection of re-entrant code blocks**
 - A **code-block** is equivalent to a **function** or a **loop body** in the high-level context
 - Each **code-block** comprises of **several threads**
 - A **Thread** is a sequence of instructions equivalent to a **basic block**
 - A **producer-consumer relationship** exists among threads
 - **Scheduling** of code-blocks and threads is done **dynamically** according to **data availability**



Model of execution

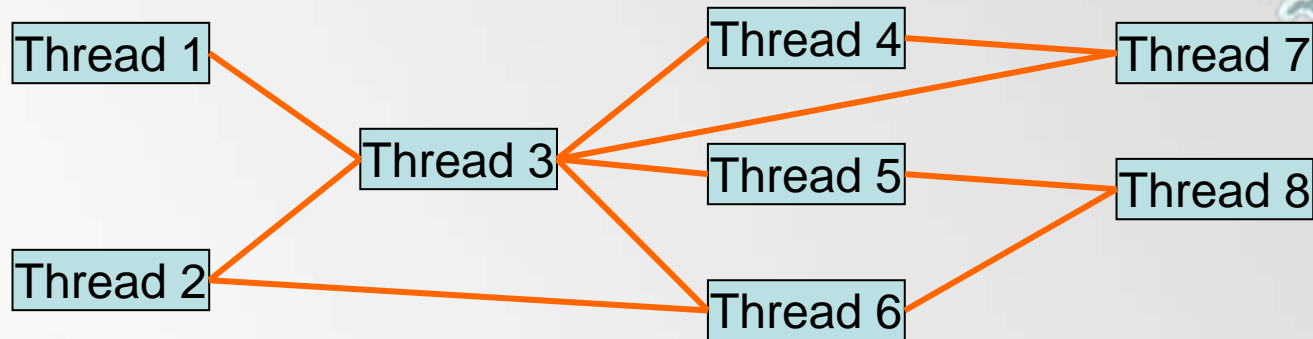


Nodes of the synchronization graph represent the threads

- A program in DDM is a **collection of re-entrant code blocks**
 - A **code-block** is equivalent to a **function** or a **loop body** in the high-level context
 - Each **code-block** comprises of **several threads**
 - A **Thread** is a sequence of instructions equivalent to a **basic block**
 - A **producer-consumer relationship** exists among threads
 - **Scheduling** of code-blocks and threads is done **dynamically** according to **data availability**



Model of execution

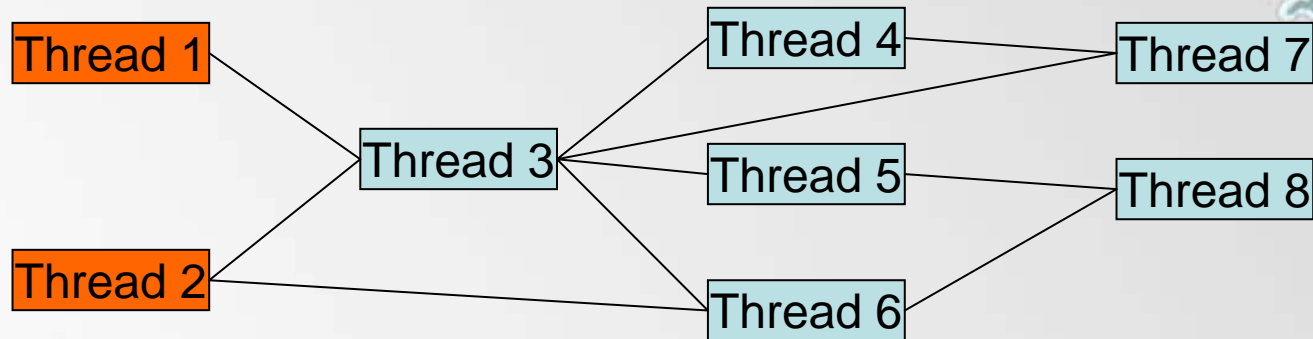


Arcs carry data from the producer to the consumer threads

- A program in DDM is a collection of re-entrant code blocks
 - A code-block is equivalent to a function or a loop body in the high-level context
 - Each code-block comprises of several threads
 - A Thread is a sequence of instructions equivalent to a basic block
 - A producer-consumer relationship exists among threads
 - Scheduling of code-blocks and threads is done dynamically according to data availability



Model of execution

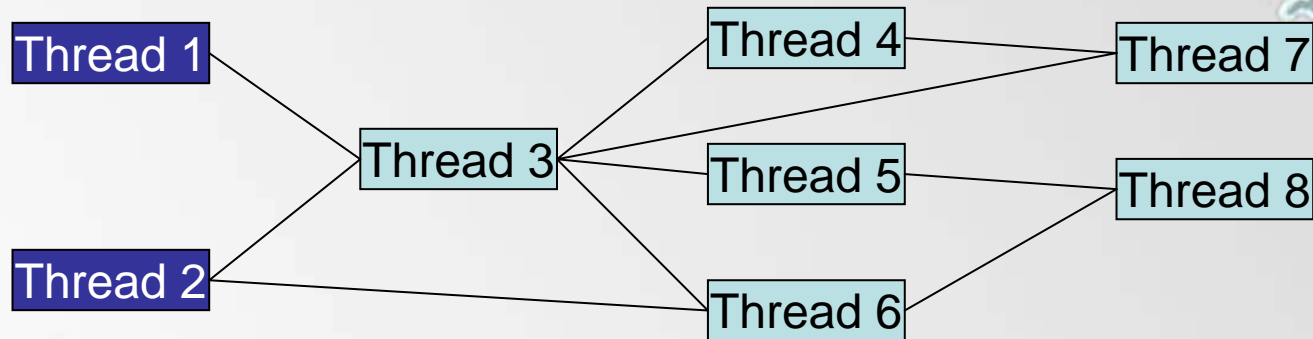


Threads 1 and 2 are executing in parallel

- A program in DDM is a **collection of re-entrant code blocks**
 - A **code-block** is equivalent to a **function** or a **loop body** in the high-level context
 - Each **code-block** comprises of **several threads**
 - A **Thread** is a sequence of instructions equivalent to a **basic block**
 - A **producer-consumer relationship** exists among threads
 - **Scheduling** of code-blocks and threads is done **dynamically** according to **data availability**



Model of execution

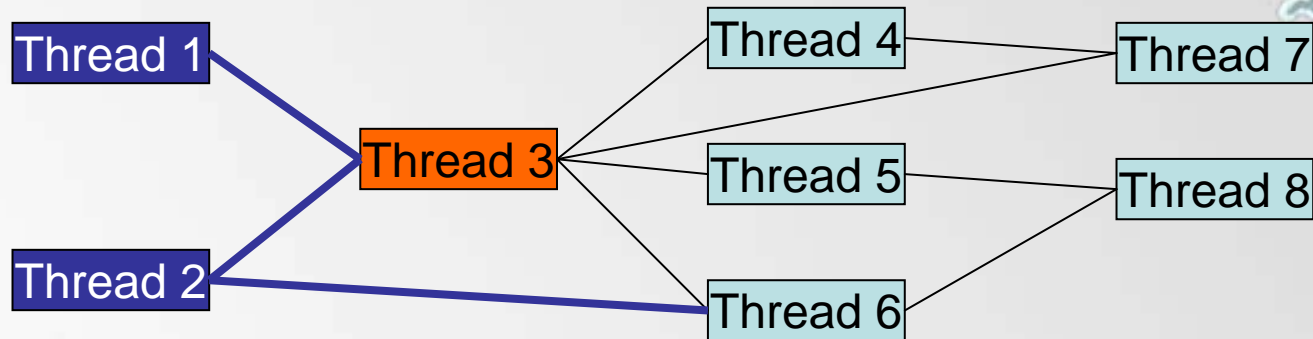


Threads 1 and 2 have completed their execution

- A program in DDM is a **collection of re-entrant code blocks**
 - A **code-block** is equivalent to a **function** or a **loop body** in the high-level context
 - Each **code-block** comprises of **several threads**
 - A **Thread** is a sequence of instructions equivalent to a **basic block**
 - A **producer-consumer relationship** exists among threads
 - **Scheduling** of code-blocks and threads is done **dynamically** according to **data availability**



Model of execution

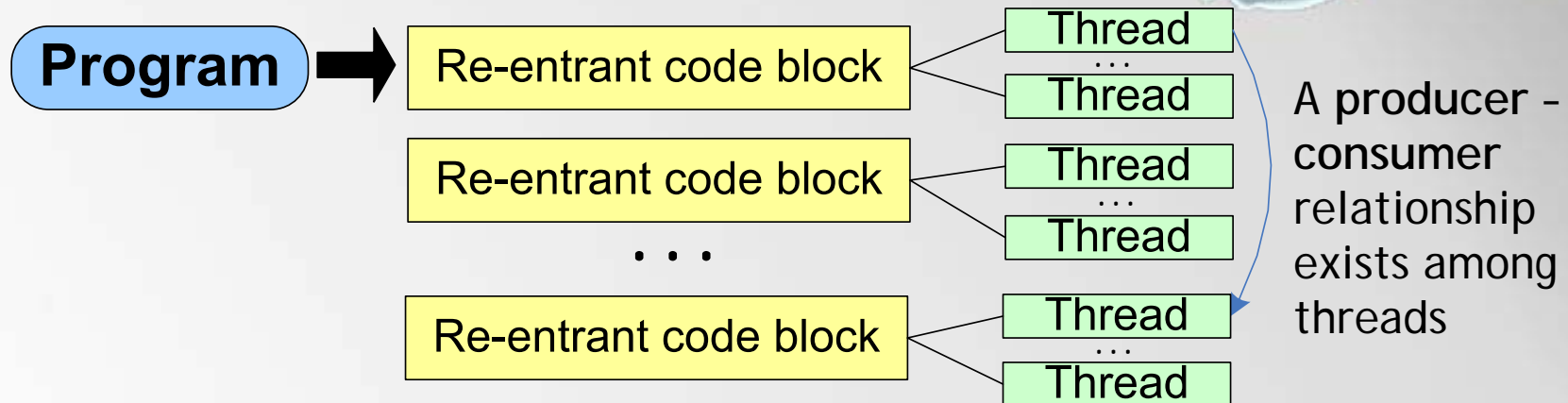


Results “propagation” – Thread 3 is ready to be executed

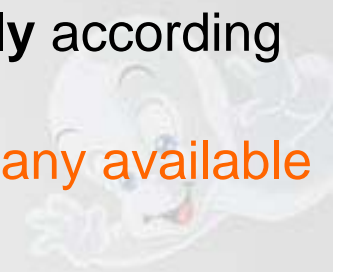
- A program in DDM is a **collection of re-entrant code blocks**
 - A **code-block** is equivalent to a **function** or a **loop body** in the high-level context
 - Each **code-block** comprises of **several threads**
 - A **Thread** is a sequence of instructions equivalent to a **basic block**
 - A **producer-consumer relationship** exists among threads
 - **Scheduling** of code-blocks and threads is done **dynamically** according to **data availability**



Model of execution



- A **program in DDM** is a **collection of re-entrant code blocks**
 - A **code-block** is **equivalent to a function** or a **loop body** in the high-level context
 - Each **code-block** comprises of **several threads**
 - A **Thread** is a sequence of instructions equivalent to a **basic block**
 - A **producer-consumer relationship** exists among **threads**
 - **Scheduling** of code-blocks and threads is done **dynamically** according to **data availability**
 - The CPU can execute the instructions within a thread using any available optimization



Data Driven Multithreading



- Why **Data Driven** multithreading and not any other multithreading model of execution?
 - DDM tolerates **communication** latencies
 - DDM tolerates **synchronization** latencies
 - DDM offers effective **data prefetching**
 - DDM can be implemented with **commodity microprocessors**

Data Driven Multithreading overtakes the memory wall

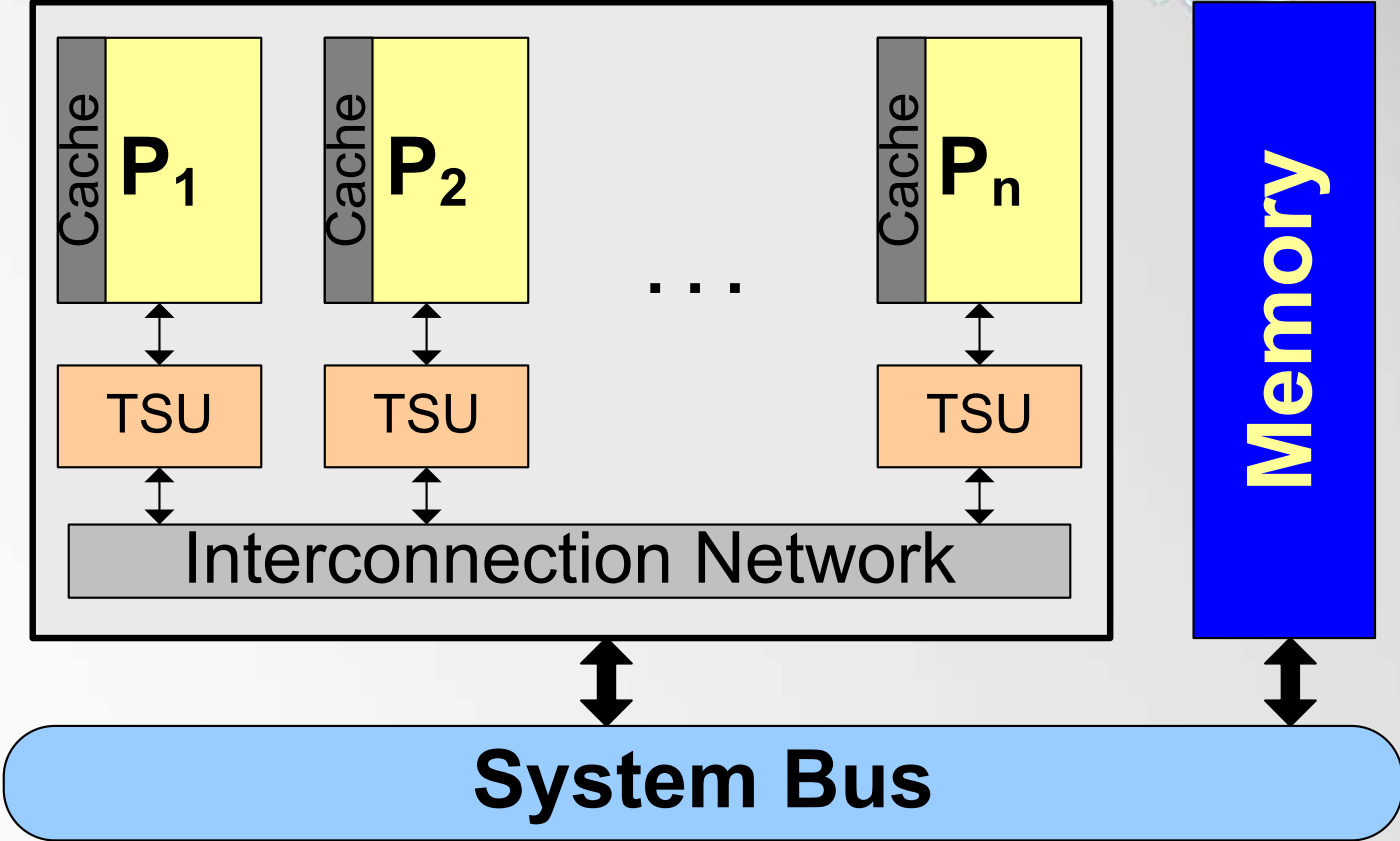
Outline



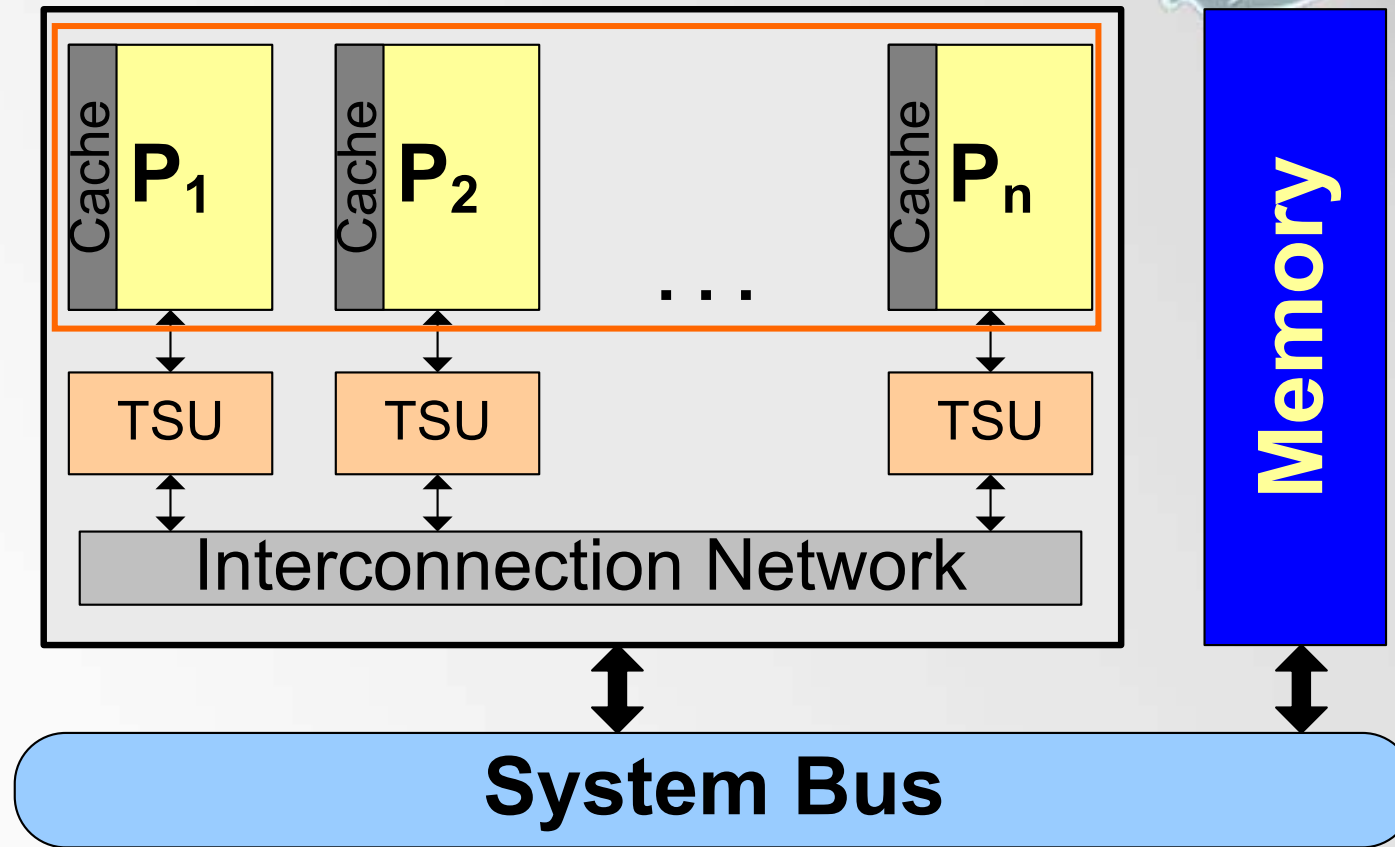
- Motivation
- Data Driven Multithreading (DDM) model of execution
- **DDM-CMP architecture**
- Proof of concept
- Experimental Setup
- Experimental Results
- Thermal Aware Scheduling
- Work in progress
- Conclusions



DDM-CMP Architecture



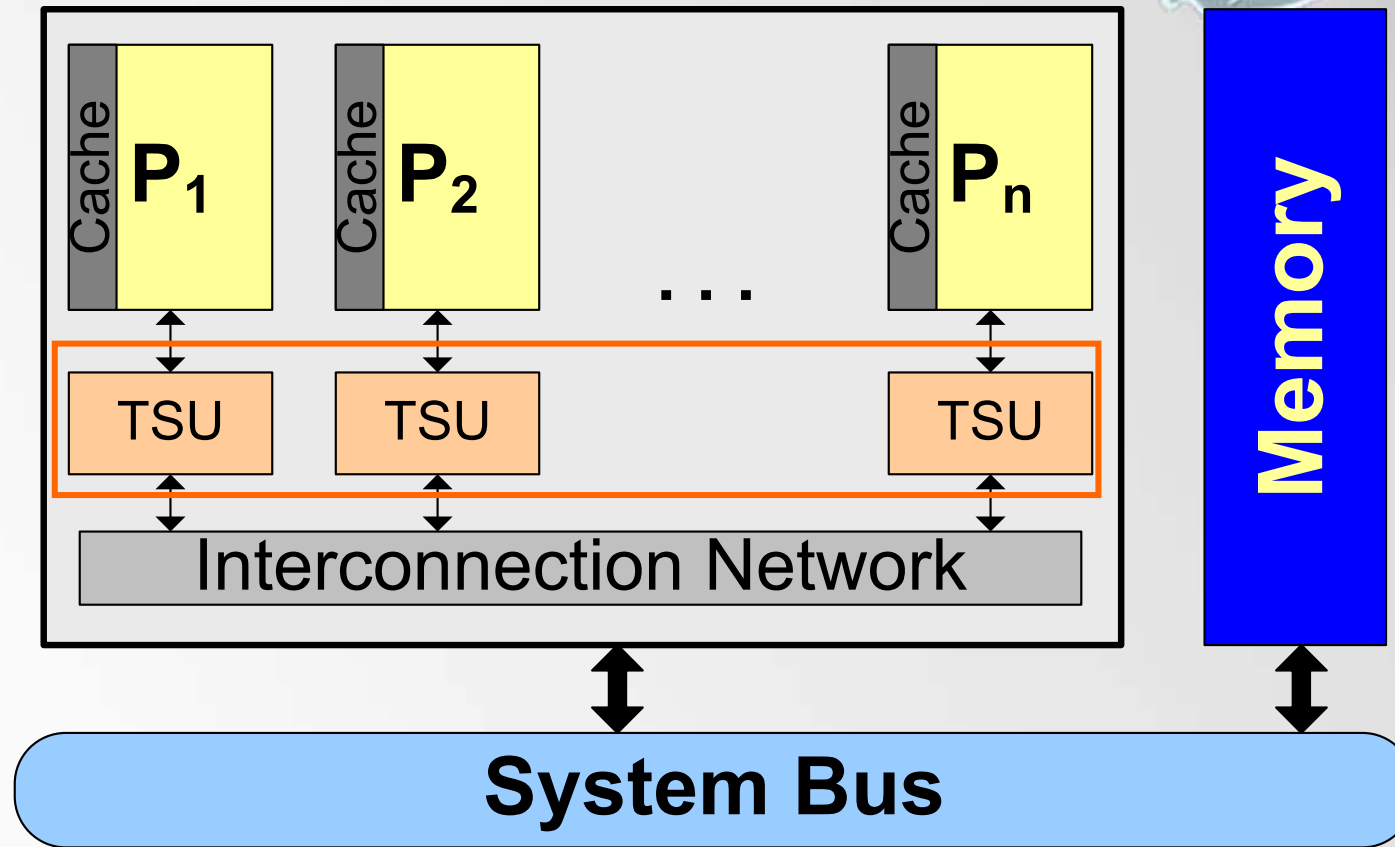
DDM-CMP Architecture



- The execution processors
- Cache is embedded in the execution processors



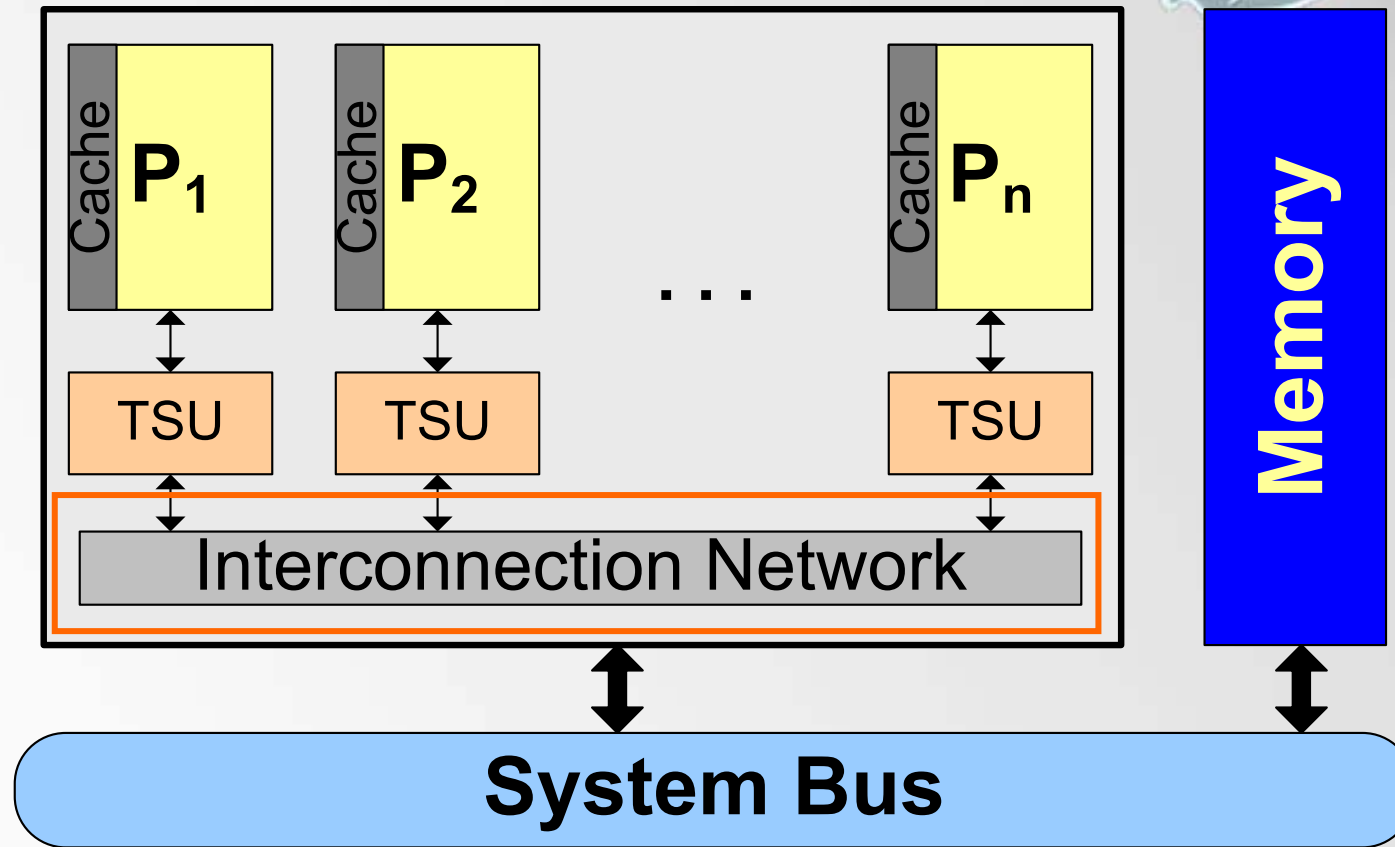
DDM-CMP Architecture



- The thread synchronization units (TSUs)
- One per processor here



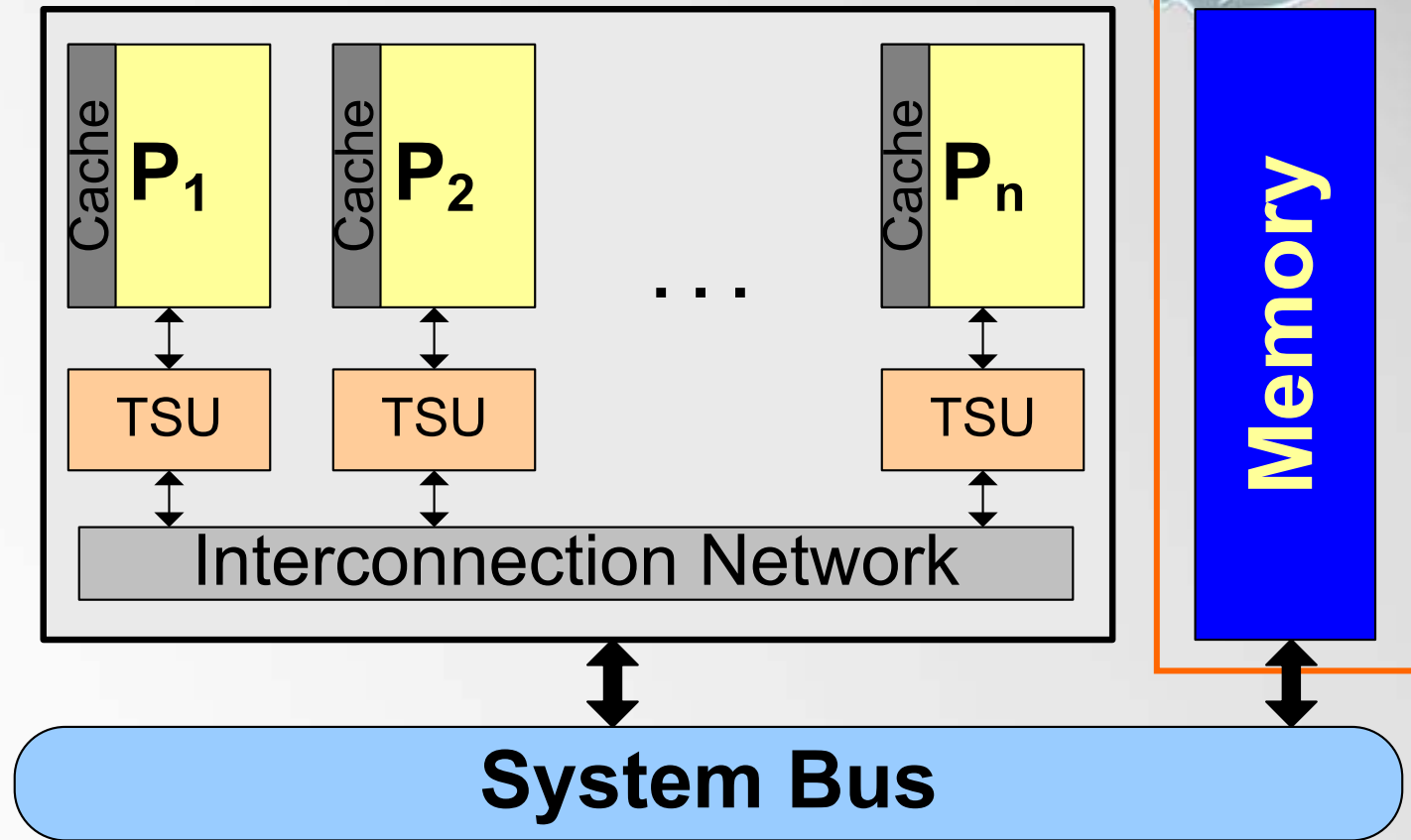
DDM-CMP Architecture



- The interconnection network
- TSUs communicate with each other through the interconnection network



DDM-CMP Architecture



- Off-chip memory
- Communication via the system bus



Outline



- Motivation
- Data Driven Multithreading (DDM) model of execution
- DDM-CMP architecture
- **Proof of concept**
- Experimental Setup
- Experimental Results
- Thermal Aware Scheduling
- Work in progress
- Conclusions



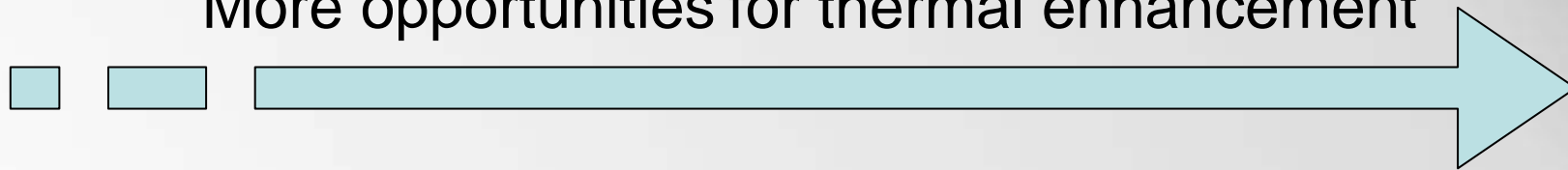
Proof of Concept



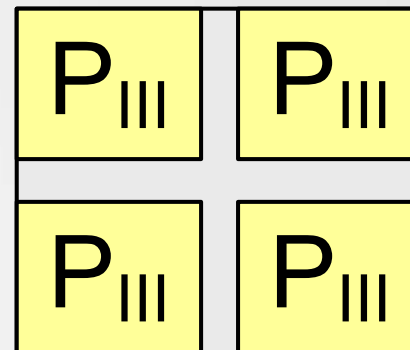
More Parallelism

Less Complex design

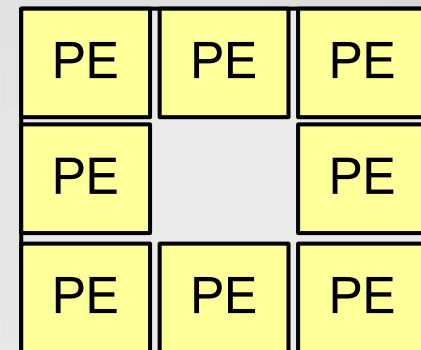
More opportunities for thermal enhancement



=



=

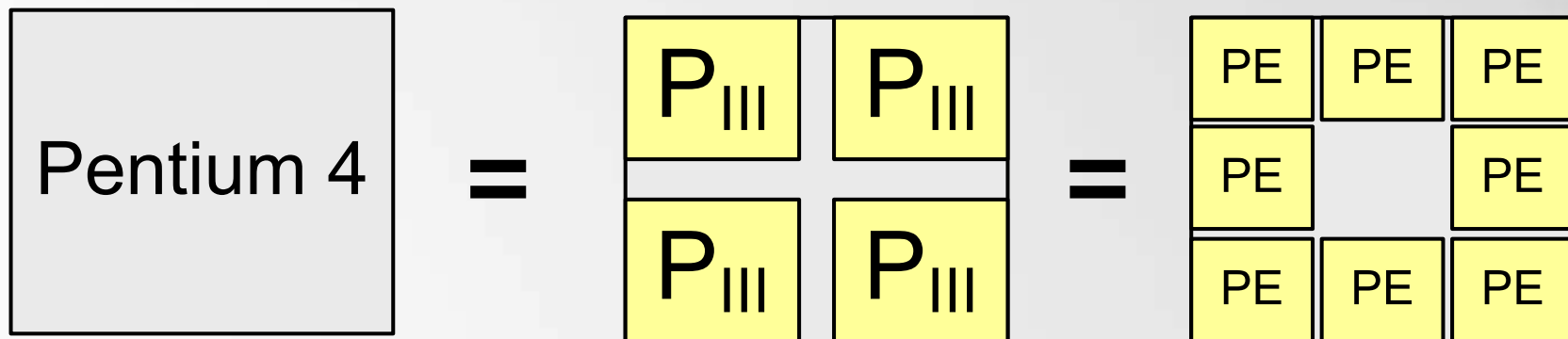


	Processor	Freq.	Tech.	L2 cache	L1 cache
P4	Pentium 4 - HT	3.2GHz	90nm	1MB	~28KB
P_III	Pentium III	800MHz	180nm	256KB	32KB
PE	Modified Pentium III	800MHz	180nm	32KB	16KB

Proof of Concept



- The number of transistors needed to build a Pentium 4 is enough for
 - 5.7 Pentium III processors
 - ~10 "modified Pentium III" processors.

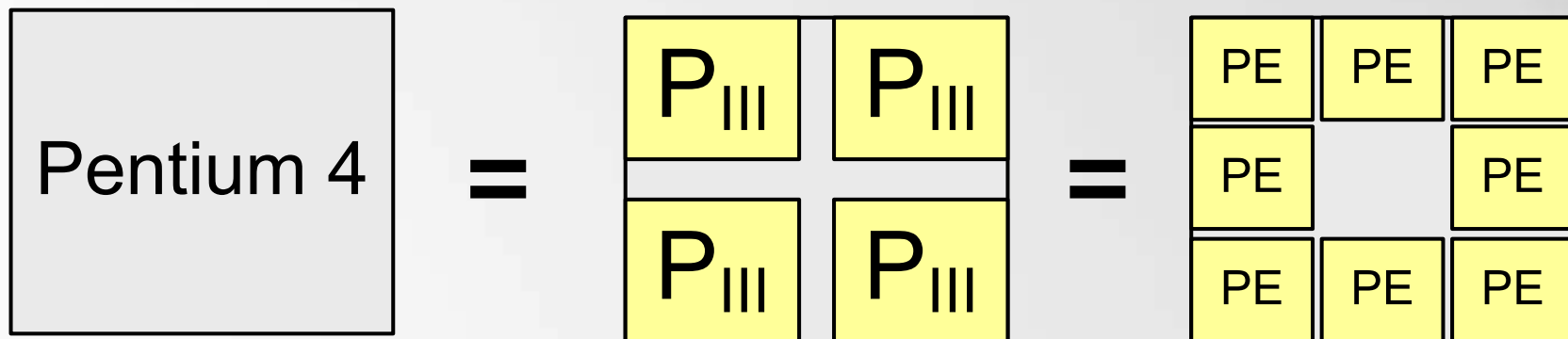


	Processor	Freq.	Tech.	L2 cache	L1 cache
P4	Pentium 4 - HT	3.2GHz	90nm	1MB	~28KB
P_III	Pentium III	800MHz	180nm	256KB	32KB
PE	Modified Pentium III	800MHz	180nm	32KB	16KB

Proof of Concept



- The **configuration of PE** was determined with simulations
 - According to our experimental results reducing cache size has minimal effect on performance (IPC) for the kernels studied



	Processor	Freq.	Tech.	L2 cache	L1 cache
P4	Pentium 4 - HT	3.2GHz	90nm	1MB	~28KB
P_III	Pentium III	800MHz	180nm	256KB	32KB
PE	Modified Pentium III	800MHz	180nm	32KB	16KB

Outline



- Motivation
- Data Driven Multithreading (DDM) model of execution
- DDM-CMP architecture
- Proof of concept
- **Experimental Setup**
- Experimental Results
- Thermal Aware Scheduling
- Work in progress
- Conclusions



Experimental Setup



- **Baseline**

- The baseline used for our experiments is a state-of-the-art single chip microprocessor, a **Hyper-threaded Pentium 4** clocked at 3.2GHz implemented with 90nm technology

- **Benchmarks**

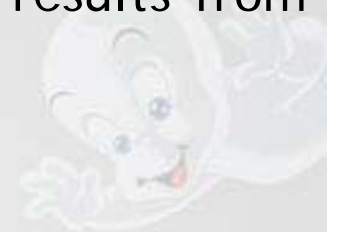
- To evaluate the proposed architecture we used **kernels** from **SPLASH2** benchmark suite



Experimental Setup



- **Pentium 4 (Baseline)**
 - The performance of Pentium 4 was quantified using **native execution** (hardware counters)
- **4 Cores CMP (Pentium III)**
 - The performance of a **single Pentium III** processor was quantified using **native execution** (hardware counters)
 - Performance of the 4-cores CMP was **extrapolated** using results of our previous work for the D²NOW (*Extrapolation is conservative*)
- **8 Cores CMP (PE: Modified Pentium III)**
 - The **performance loss** of modified compared to the **original Pentium III** was estimated via simulation (**SimpleScalar**)
 - Performance of the 8-cores CMP was **extrapolated** using results from **D²NOW**



Outline



- Motivation
- Data Driven Multithreading (DDM) model of execution
- DDM-CMP architecture
- Proof of concept
- Experimental Setup
- **Experimental Results**
- Thermal Aware Scheduling
- Work in progress
- Conclusions



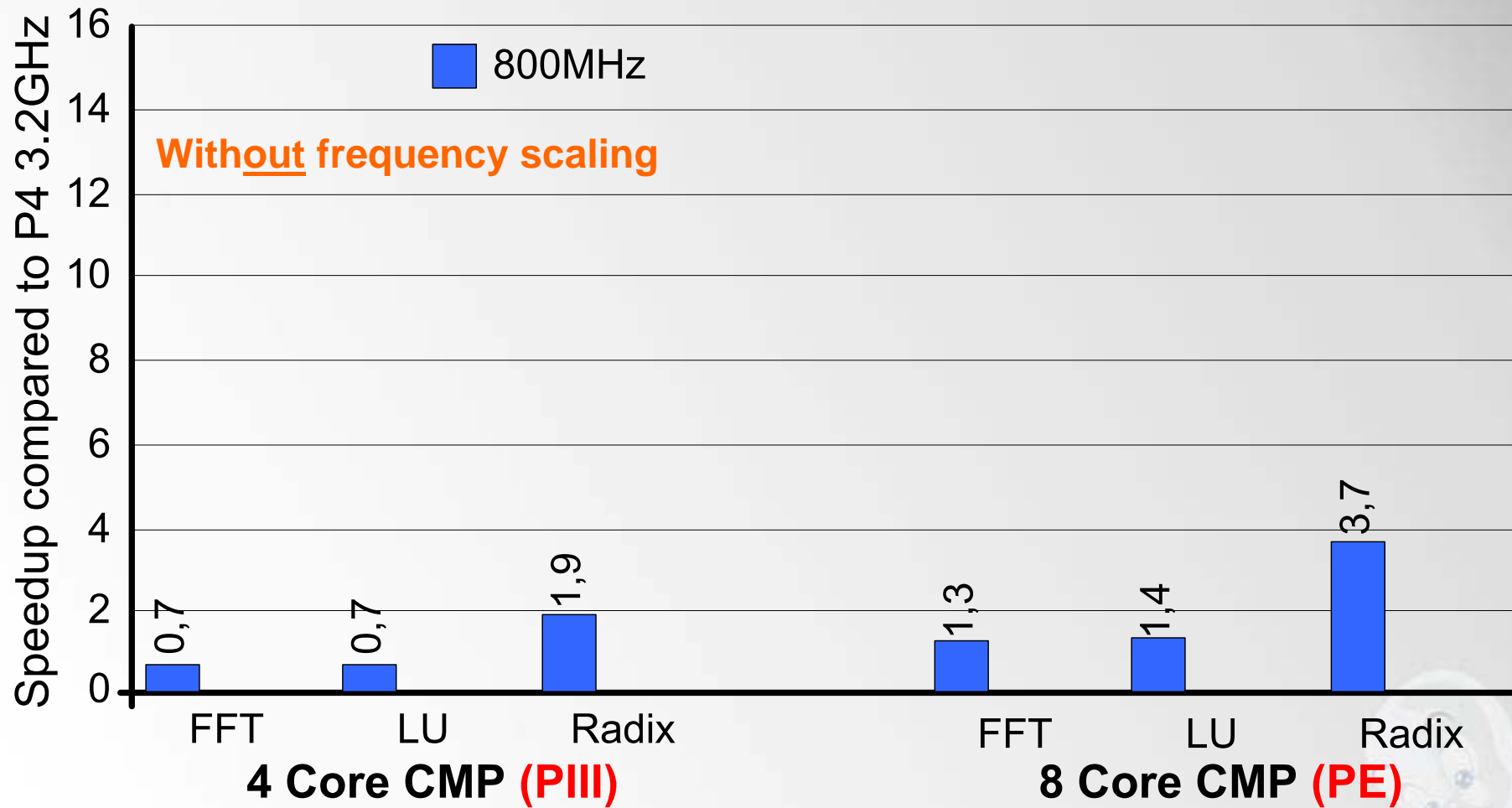
Frequency Scaling



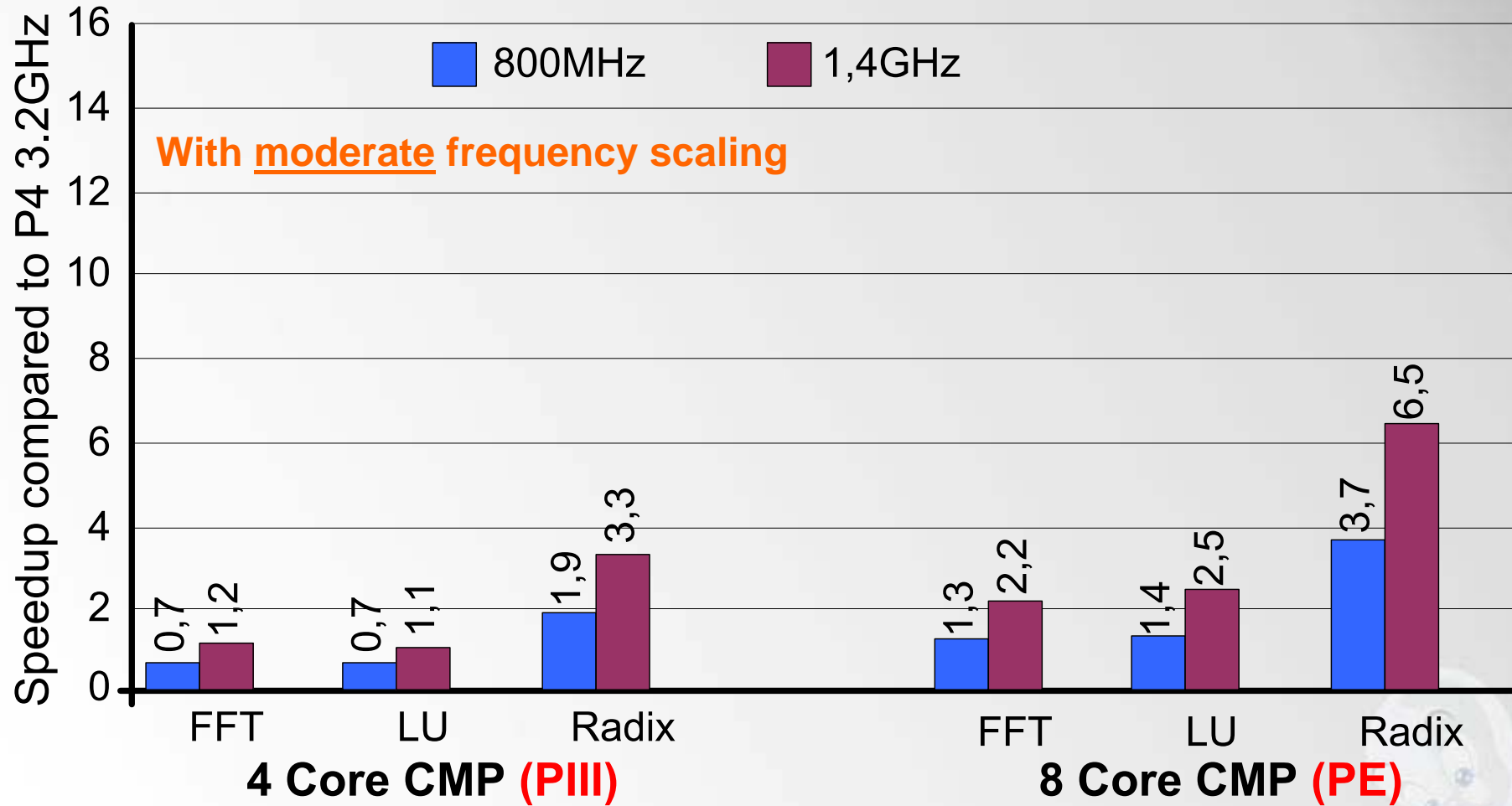
- For our experiments:
 - **Pentium 4** was clocked at **3.2GHz** with implementation technology of **90nm**
 - **The Pentium III** was clocked at **800MHz** with implementation technology **180nm**
- Today's implementation technology allows us to build a Pentium III with significantly **higher frequency than 800MHz**
- The "fastest" Pentium III ever was scaled at **1,4GHz (Tualatin)**



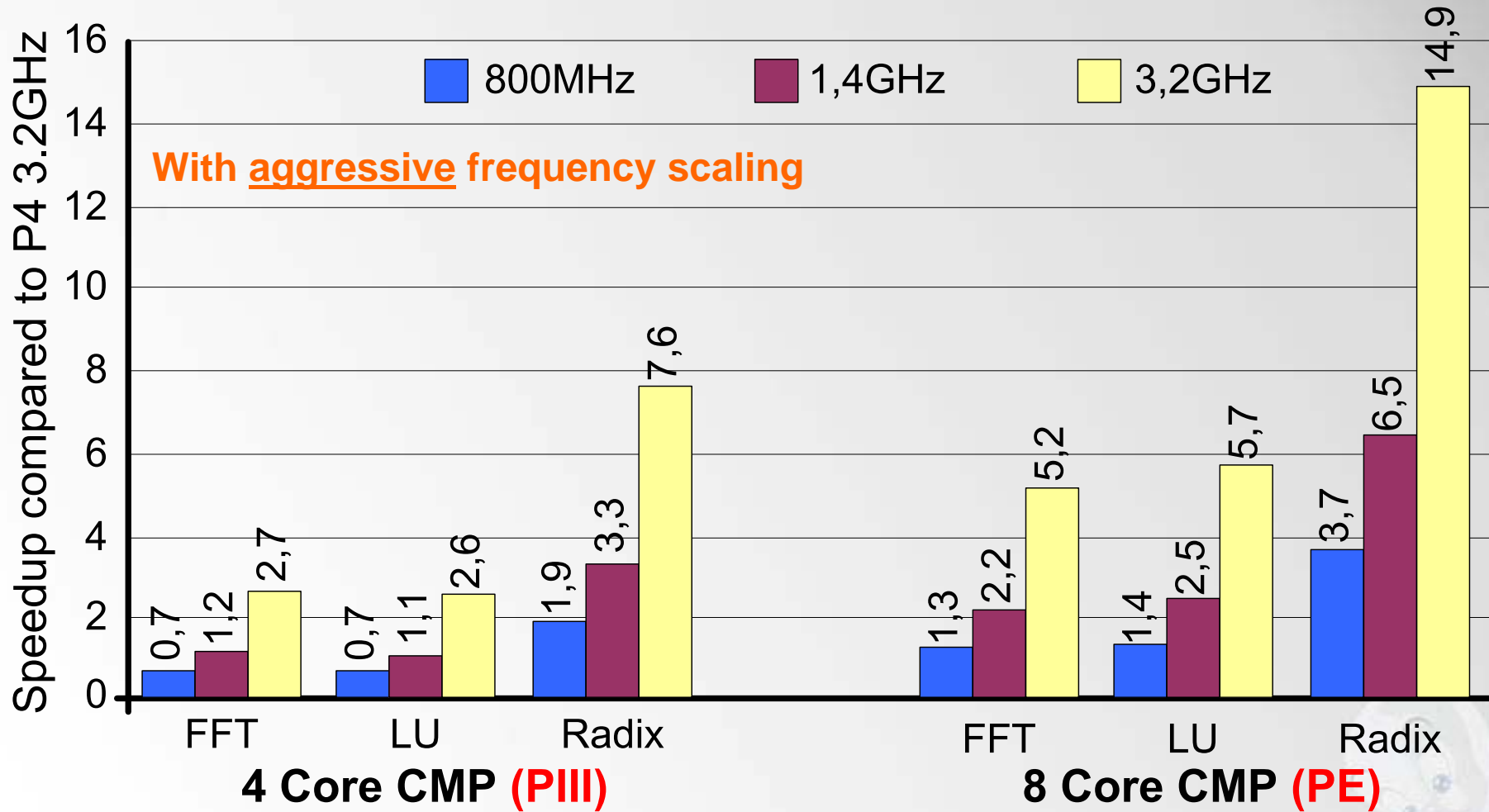
Experimental Results



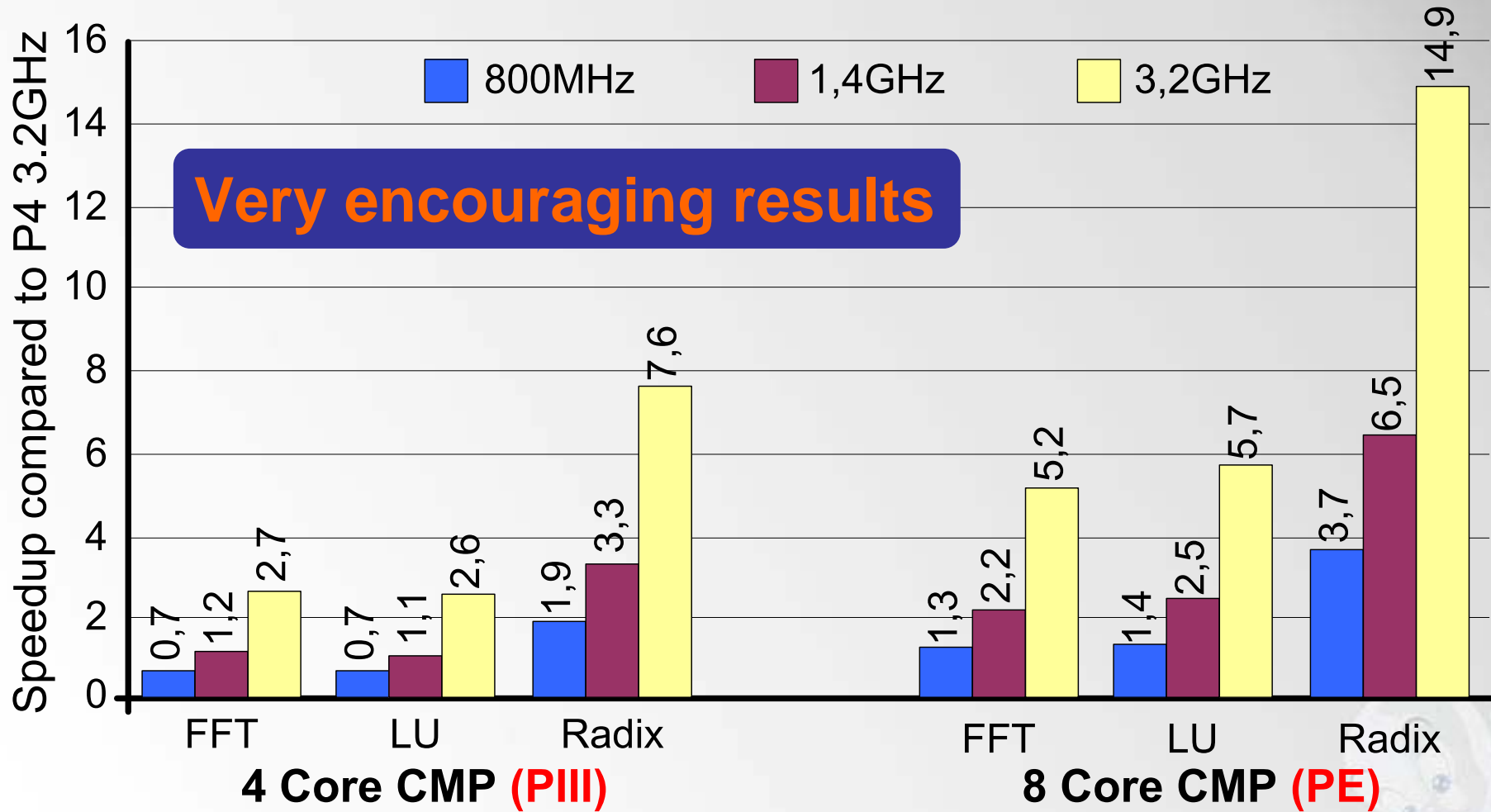
Experimental Results



Experimental Results



Experimental Results



Outline



- Motivation
- Data Driven Multithreading (DDM) model of execution
- DDM-CMP architecture
- Proof of concept
- Experimental Setup
- Experimental Results
- **Thermal Aware Scheduling**
- Work in progress
- Conclusions



Thermal Optimization Opportunities



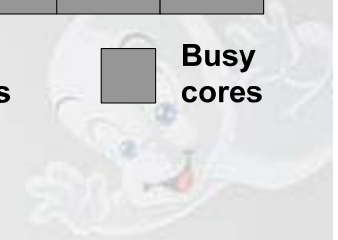
- In a multiprogramming environment some **cores** of the **CMP** will be **idle at any given time point**
- **Scheduling algorithms** will define where (on which idle core) a new process will run
- Examples of Scheduling Algorithms
 - **Always Coolest**
 - **Neighborhood aware**
- To evaluate the potential
 - **TSIC**: Thermal Simulator for Chip Multiprocessors

65	64	59	70
50	70	54	52
68	66	58	70
50	62	69	72

Idle
Cores

Busy
cores

*This work was accepted for publication in the
10th Panhellenic Conference on Informatics*



DDM-CMP **enhanced** thermal Optimization Opportunities



- DDM-CMP offers **enhanced** thermal optimization opportunities:
 - DDM-CMP is likely to be build out of a large number of small embedded microprocessors
 - Scheduling will be often in the DDM architecture
 - Thermal profiling information can be included during compilation



Outline



- Motivation
- Data Driven Multithreading (DDM) model of execution
- DDM-CMP architecture
- Proof of concept
- Experimental Setup
- Experimental Results
- Thermal Aware Scheduling
- **Work in progress**
- Conclusions



Work in progress



- A **hardware prototype** will be build using **Xilinx Virtex II Pro** chip
 - It includes
 - 2 embedded PowerPC processors
 - Programmable FPGA
 - We will execute **application threads** on the two **processors** and implement the **interconnection network** and the **TSUs** on the **FPGA**
- **DDM-CMP simulator**
 - Under implementation using the Liberty Simulation Environment (**LSE**)
- **DDM-CMP compiler**
 - Aims at automatically converting conventional code to binaries appropriate for the DDM model of execution



Conclusions



- **DDM-CMP**
 - Takes into **advantage thread level parallelism**
 - **Decreased** design and implementation **complexity**
 - **Improves** the **thermal** characteristics of the chip
- Achieves **speedup** ranging from **5.2 - 14.9** compared to a current high-end state-of-the-art single chip microprocessor
- A **hardware prototype** is under development



Questions



Thank you!

Questions?

The **CASPER** group:

www.cs.ucy.ac.cy/carch/casper/

