



Institute of Computing Technology
Chinese Academy of Sciences



High-Efficient Architecture of Godson-T Many-Core Processor

Dongrui Fan, Hao Zhang, Da Wang, Xiaochun Ye,
Fenglong Song, Junchao Zhang, Lingjun Fan

Advanced Micro-System Group
National Key Laboratory of Computer Architecture,
Institute of Computing Technology, Chinese Academy of Sciences



Overview



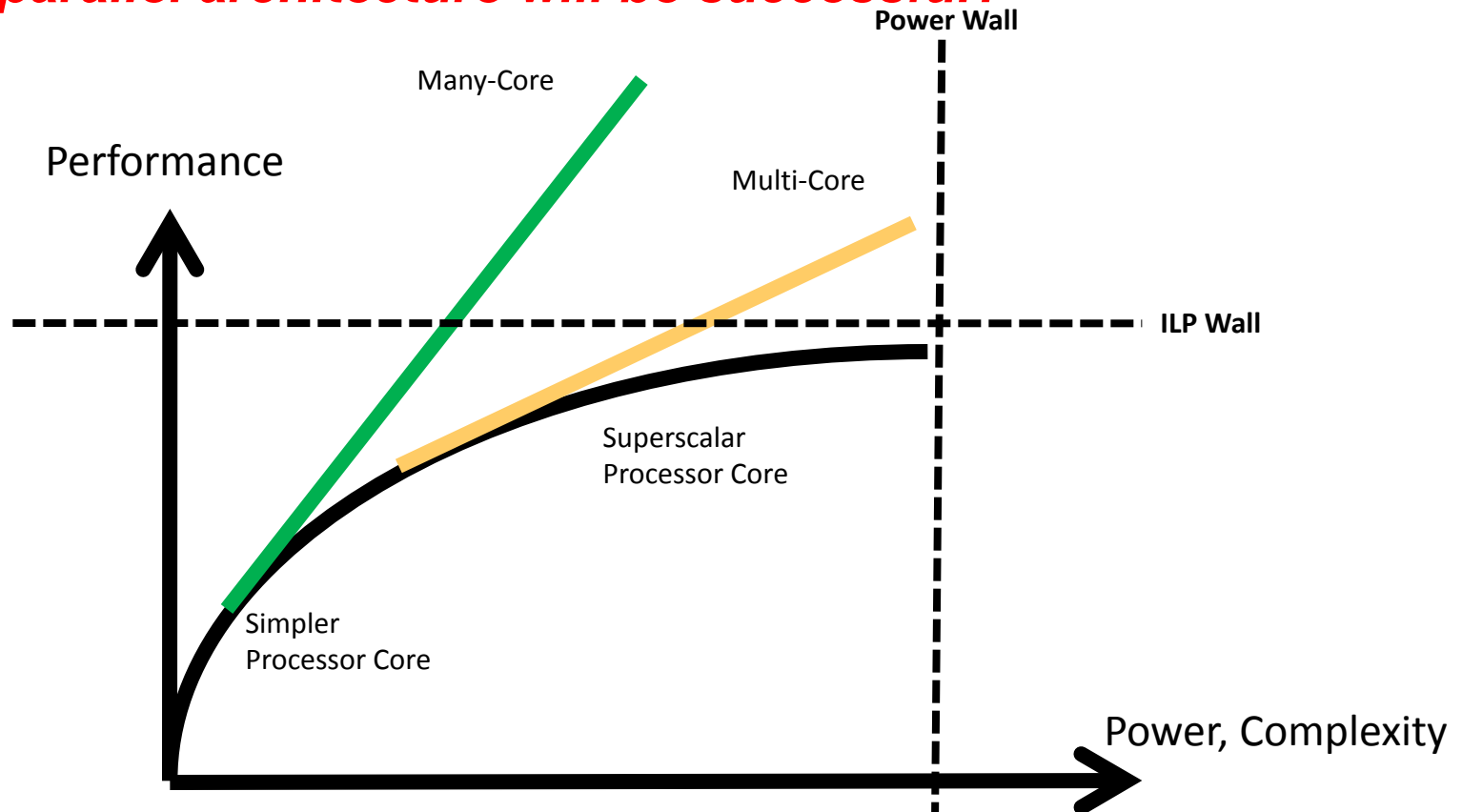
- **Microprocessor Architecture Challenges**
- **Godson-T Features for Parallel Program**
- **Godson-T Design and Implementation**

Microprocessor Architecture Revolution



- Memory wall + Power wall + ILP wall = Brick wall !

What parallel architecture will be successful?

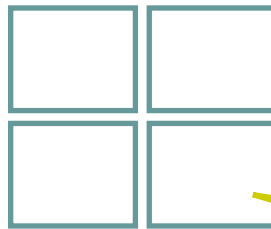


- Parallel architectures come to rescue, superscalar processor is being replaced by on-chip multi-core / many-core processor

Exploiting Polymorphic Parallelism



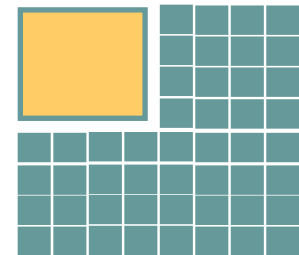
**Instruction-Level
Parallelism**



Godson-3



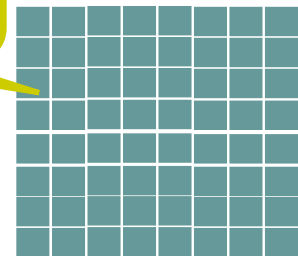
**With Powerful
Streaming Unit**



**Heterogenous
Many-Core**

**With moderate
DLP and strong
TLP exploitation**

Godson-T



Thread-Level Parallelism

Many-Core Challenges: The 5 P's



- **Power efficiency challenge**
 - Performance per watt is the new metric – Dark Silicon will appear in 2020 when facing 11nm process technology
- **Performance challenge**
 - How to scale from 1 to 1000 cores – the number of cores is the new Megahertz
- **Programming challenge**
 - How to provide a converged many core solution in a standard programming environment
- **Parallelism challenge**
 - How to exploit parallelism through software and hardware co-design
- **Platform challenge**
 - How to maximize the usability, such as, runtime system (OS), compiler & library, many-core debug...

What we could help.....



Productivity Side:

- Most sequential programmers are not ready to switch into parallel programming
 - conservative programming model and make incremental improvement?
- Unique programming model cannot solve all problems efficiently;
 - support multiple programming features efficiently?
- Locks are messy
 - new way to eliminate deadlock?

.....

Performance Side: ***Take this carefully, because it may affect productivity !***

- On-chip synchronization is fast;
 - handle all synchronizations on-chip?
- Flops are cheap, memory communications are expensive;
 - trade flops for communication latency?
- Fine-grained parallelism should be taken into consideration;
 - enable data-driven thread execution on chip?

.....



Motivation



Target at
HPC

Godson-T
Many cores to
accelerate one program

Cost
Data
Communication

Method
Fine Grained
Thread division

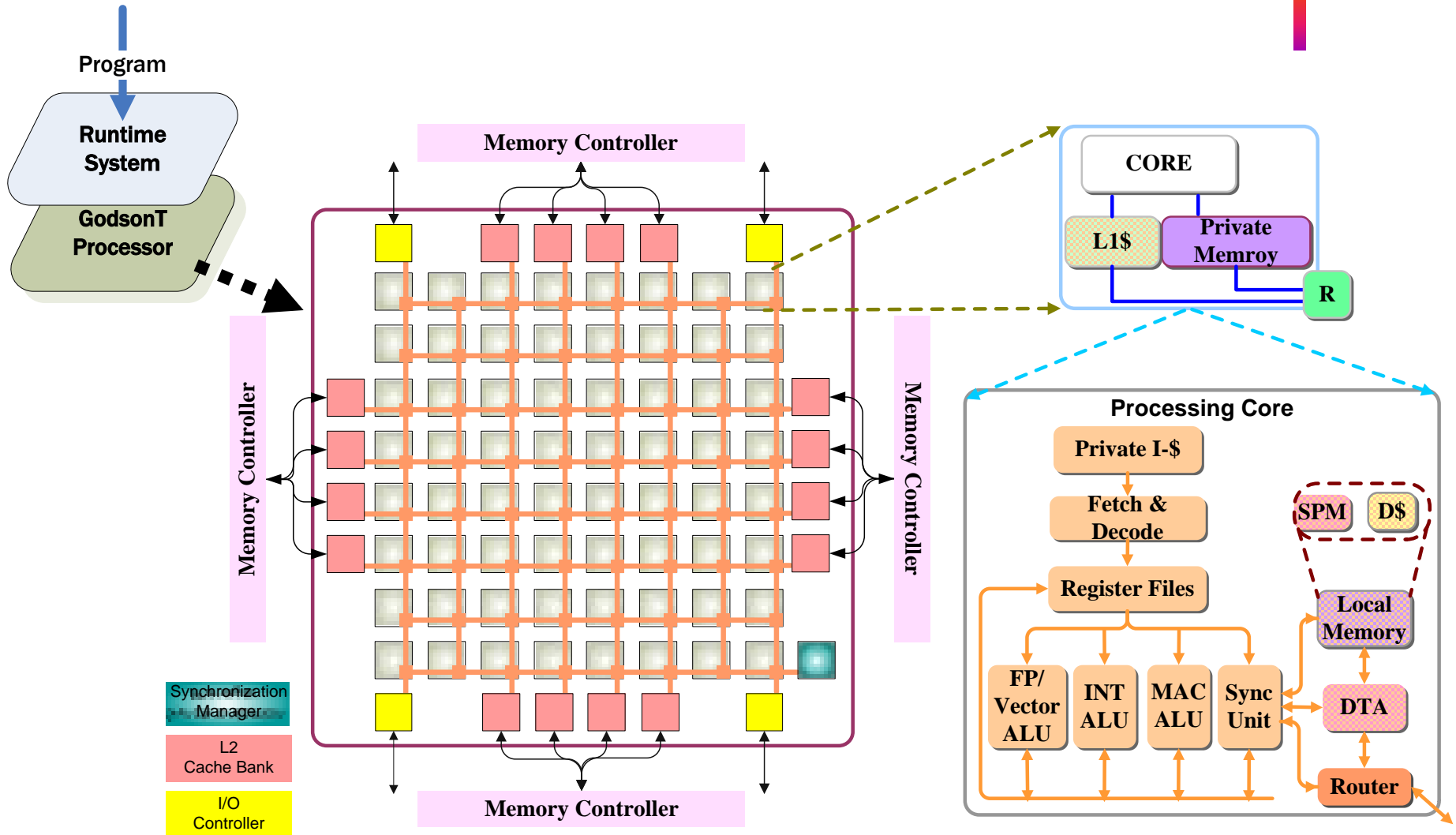
Cost
Thread
Synchronization

Overview

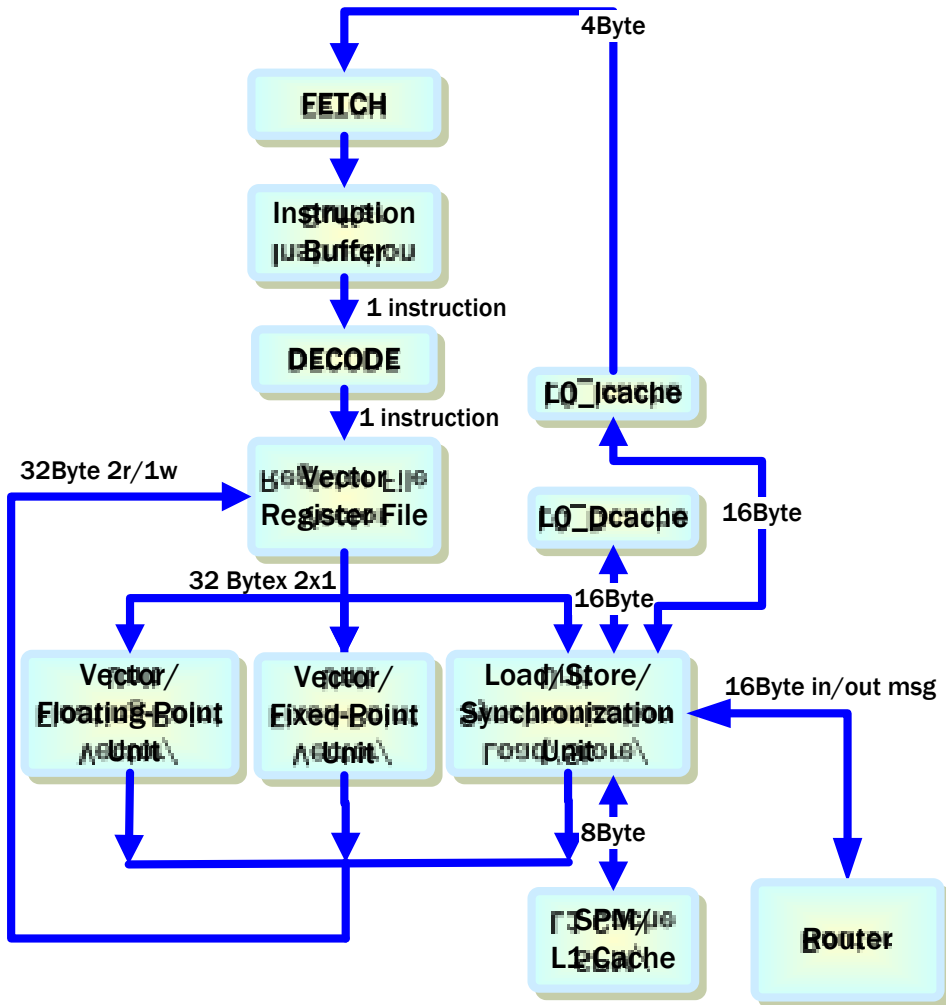


- ✓ **Microprocessor Architecture Challenges**
 - › Memory wall + Power wall + ILP wall
 - › The 5P's: Many-core Processor Challenges
- **Godson-T Features for Parallel Program**
 - › Godson-T architecture overview
 - › Architectural supports for multithreading
 - › Software runtime system
- **Godson-T Design and Implementation**
 - › Journey of design and implementation
 - › Software and hardware co-simulation
 - › Prototype

Architecture Overview of Godson-T



Processing Core

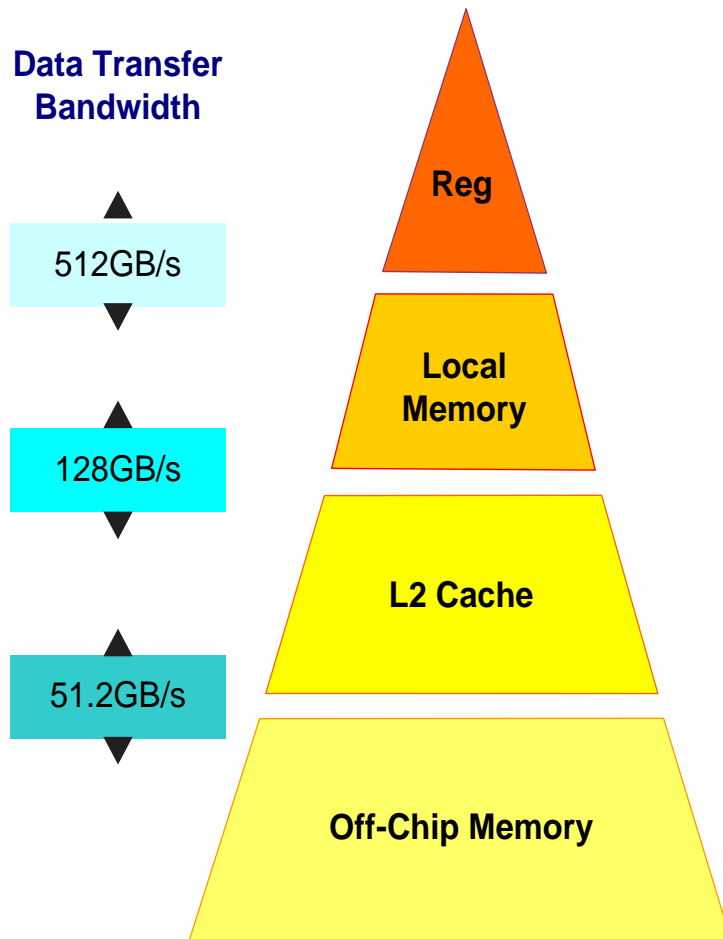


- ISA: MIPS (user), SIMD-ext., sync-ext.
- 8-stage pipeline
- Dual-issue per thread
- Expected SIMD
 - Load, Store, Data Move, Arithmetic
- Fast level-1 memory
- 16KB private memory
 - Automatically mapped into stack address space
 - Full/empty bit tagged on each 64-bit slot enables efficient producer-consumer style synchronization
- Communication with external modules through message packets

Interconnection

- Separated routers with each processing core
 - Static XY wormhole routing
 - Round-Robin arbitration
 - Two independent physical networks
 - Duplex 128bit link for each network
- Guaranteed in-order point-to-point communication
 - Deadlock-free & livelock-free
 - Scalable and power-efficient for MESH topology
 - Low latency core-to-core communication
 - Separated networks allowing traffic segregation and tolerating burst DMA transfer

Memory Hierarchy



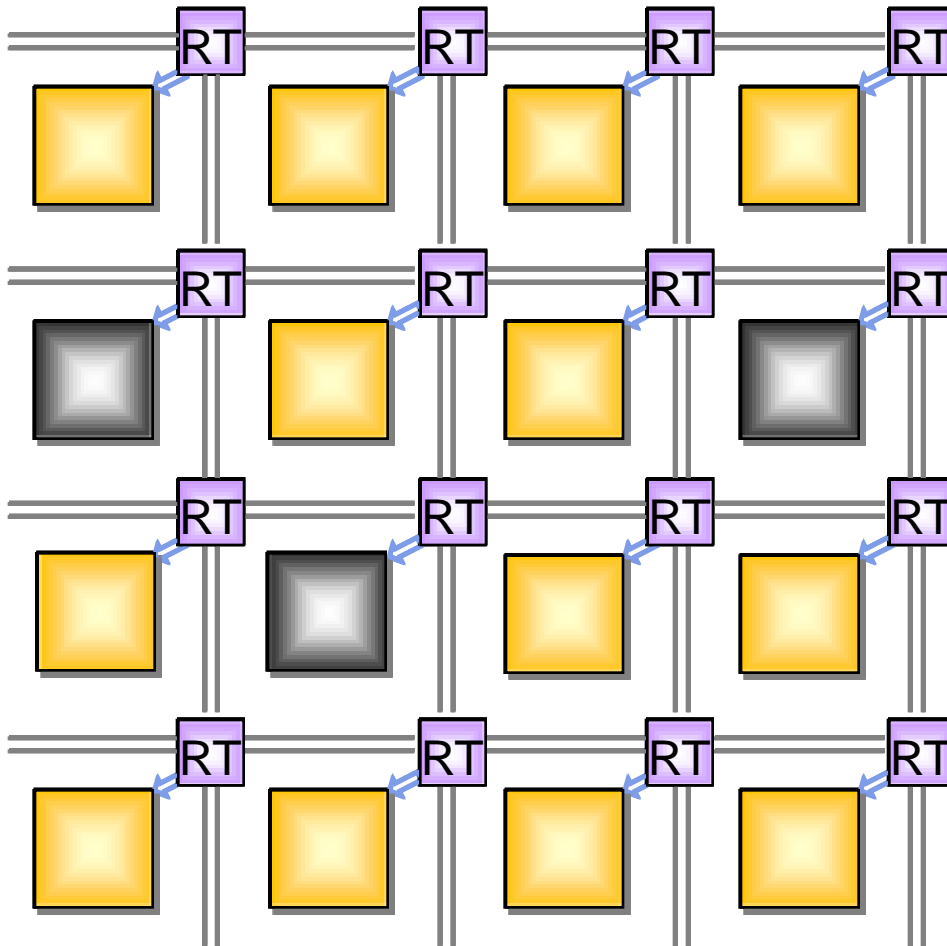
Each processing core with 32 fixed-point and 32 floating-point registers

32KB local memory, including L1\$D, SPM

16 address-interleaved L2 cache banks, 128KB each

4 DDR3-1600 memory controllers

Power Management



- Monitor status of each core at program level
- Shut down or turn on cores separately
- Reassign tasks

Overview



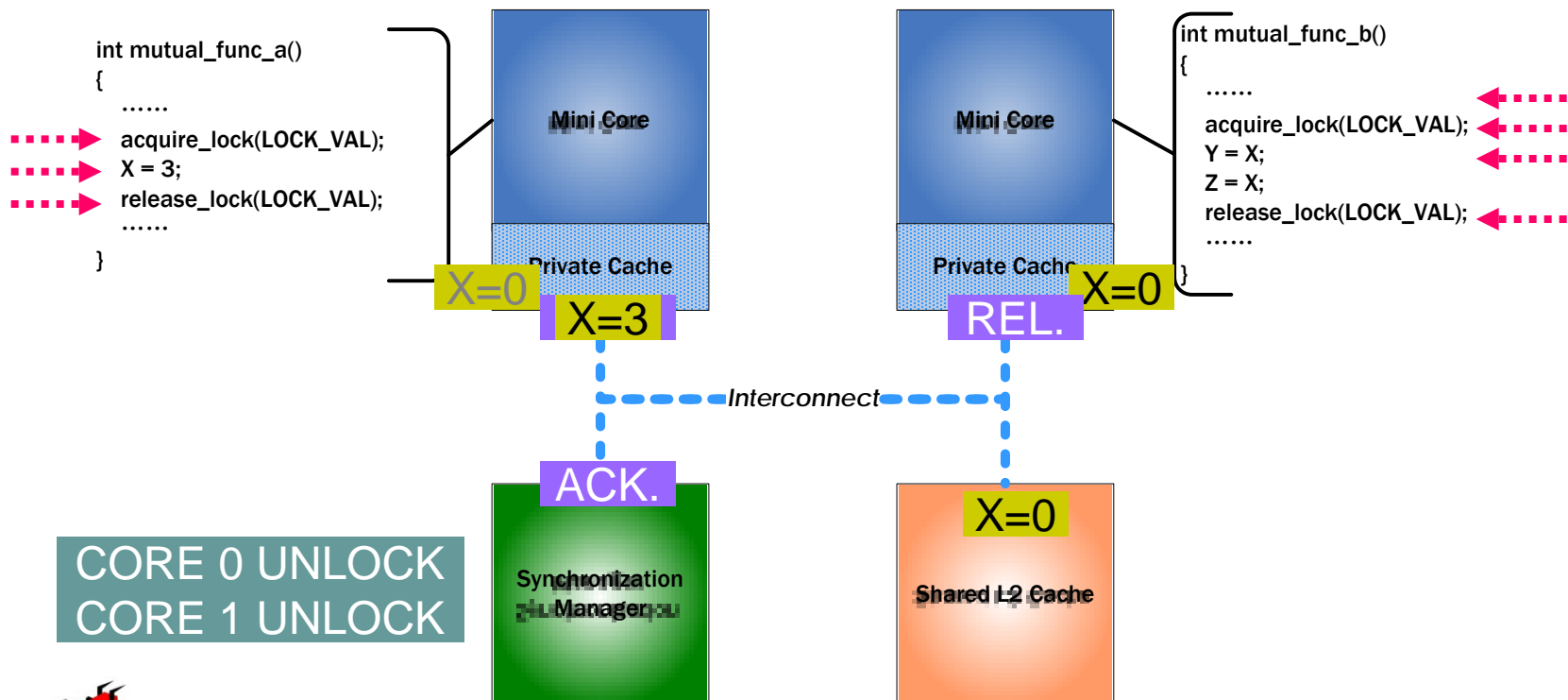
- ✓ **Microprocessor Architecture Challenges**
 - › Memory wall + Power wall + ILP wall
 - › The 5P's: Many-core Processor Challenges
- **Godson-T Features for Parallel Program**
 - › Godson-T architecture overview
 - › Architectural supports for multithreading
 - › Software runtime system
- **Godson-T Design and Implementation**
 - › Journey of design and implementation
 - › Software and hardware co-simulation
 - › Prototype

Thread Communication & Synchronization

Lock-Based Cache Coherent Protocol

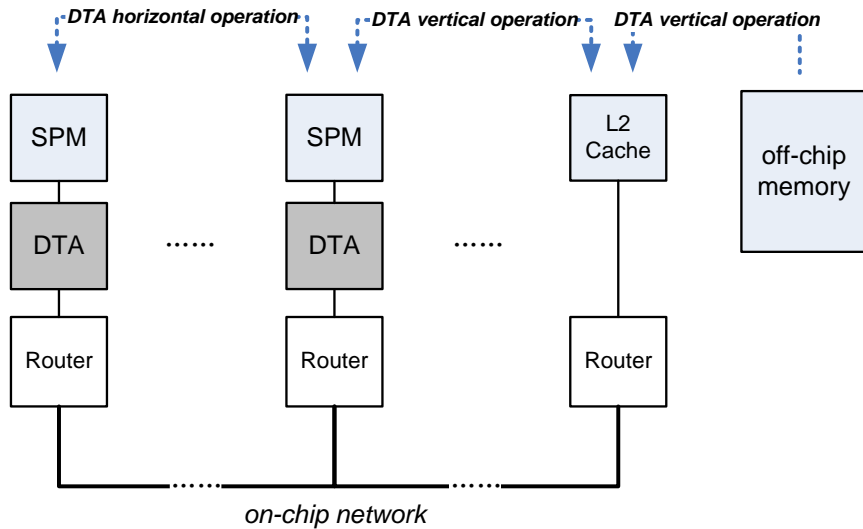


- **Lazy cache coherent protocol**
 - Pure mutual-exclusion synchronization instruction **without memory accesses**;
 - Eliminate busy-waiting for locks;
 - More scalable than bus-snoopy and directory-based cache protocol;
 - Enable hardware deadlock detecting.

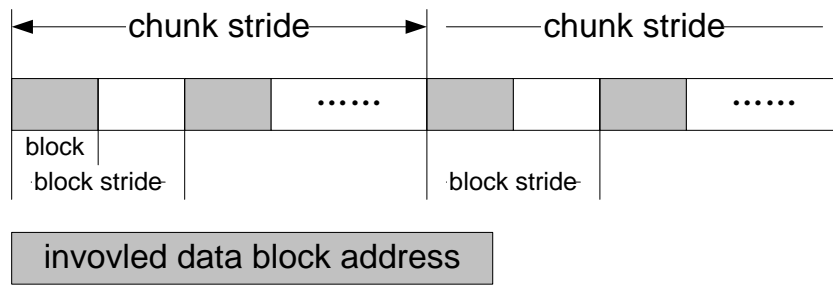


Thread Communication & Synchronization

Data Transfer Agent (DTA)



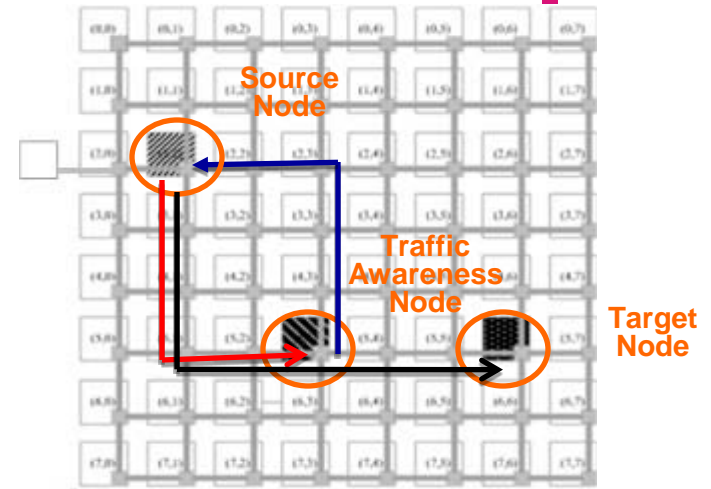
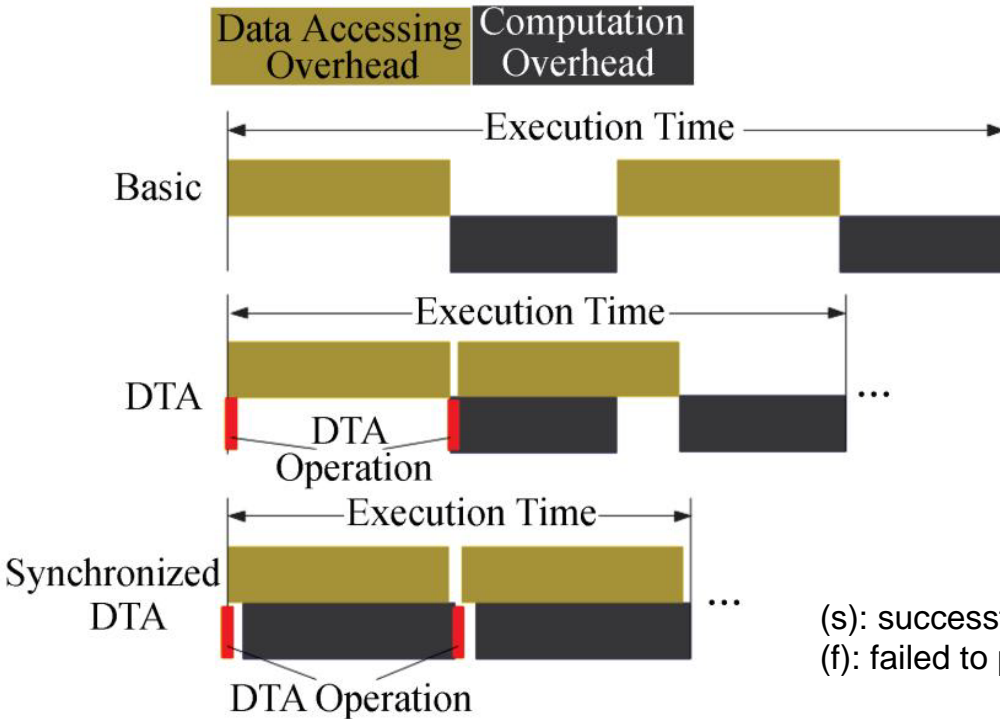
(a) vertical and horizontal DTA operations



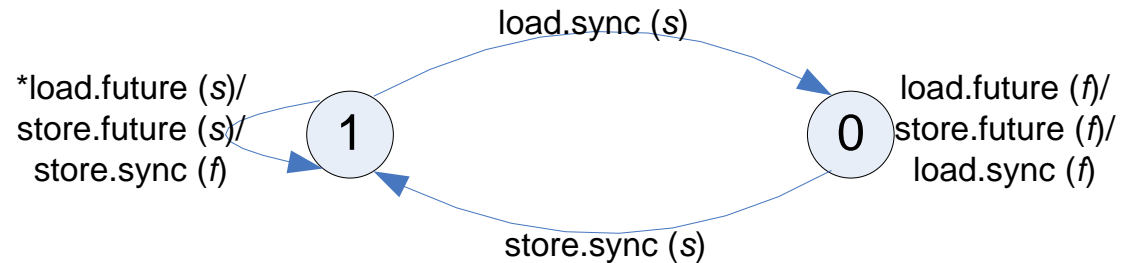
(b) 2D strided DTA operations

- **Programmable asynchronous data transfer agent**
 - Support vertical and horizontal DTA operations (such as prefetch)
 - Data transfers between multi-dimension addresses (such as matrix inversion)
 - Network load perception, automatically flow control (improve bandwidth-efficiency)
 - Support fine-grain synchronous operations

Synchronized DTA Operations

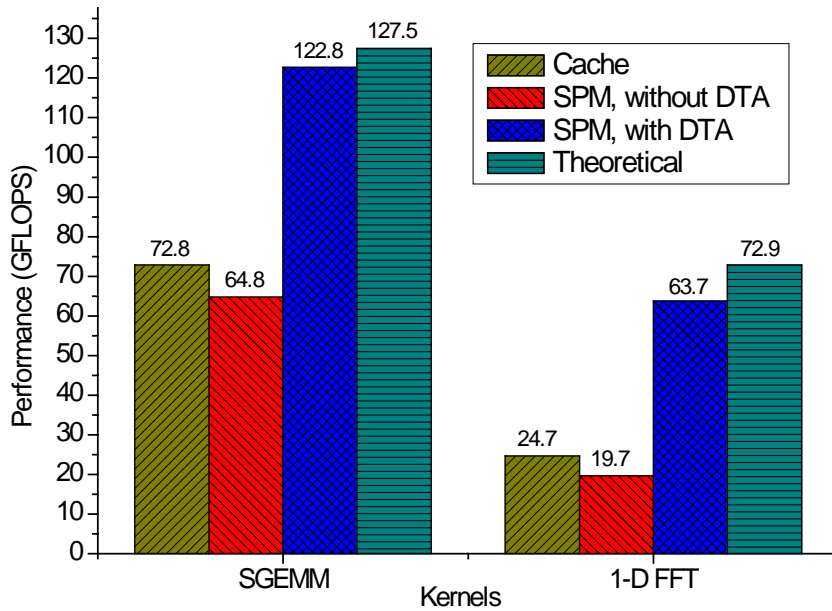


(s): successfully perform on the state of full/empty bit
 (f): failed to perform on the state of full/empty bit



Evaluating On-chip DTA

Benchmark	Processor	Godson-T	Cell	Cyclops-64	GTX8800
SGEMM	Efficiency	95.9% ¹	99.9% ²	43.4%	60.0%
	Performance	122.81	204.7	13.9	206.0
1-D FFT	Efficiency	33.2%	20.4%	25.8%	29.9%
	Performance	63.72	41.8	20.7	155.0

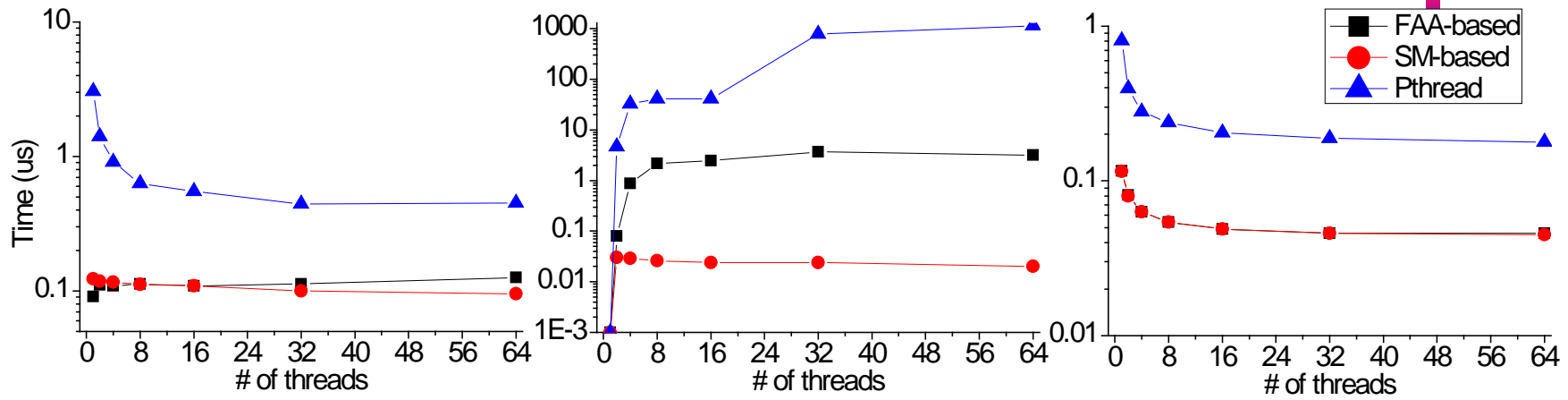


1 The SGEMM kernel contains only multiply-and-add operation, so that the ideal peak performance is measured by the multiply-and-add function unit, which is 128GFLOPS.

2 Efficiency of SGEMM on Cell is slightly better than that on Godson-T, because 256KB SPM for each SPE on Cell makes the better utilization of data locality.

Performance comparisons of SGEMM and 1-D FFT

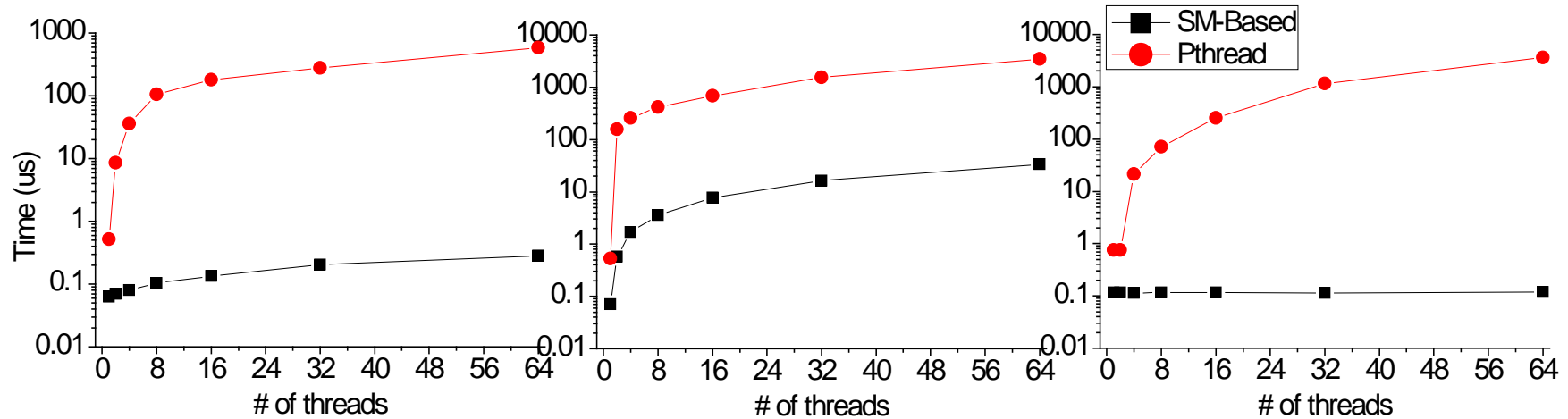
Evaluating Synchronization without Memory



(a) lock overhead without lock contention

(b) lock transferring overhead

(c) average time for each load

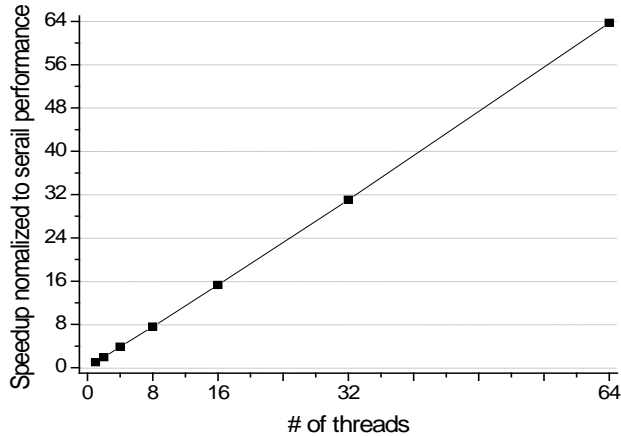


(a) barrier overhead without workload

(b) barrier overhead with load imbalance

(c) average time of each load

Evaluating Full-empty Bit Synchronization



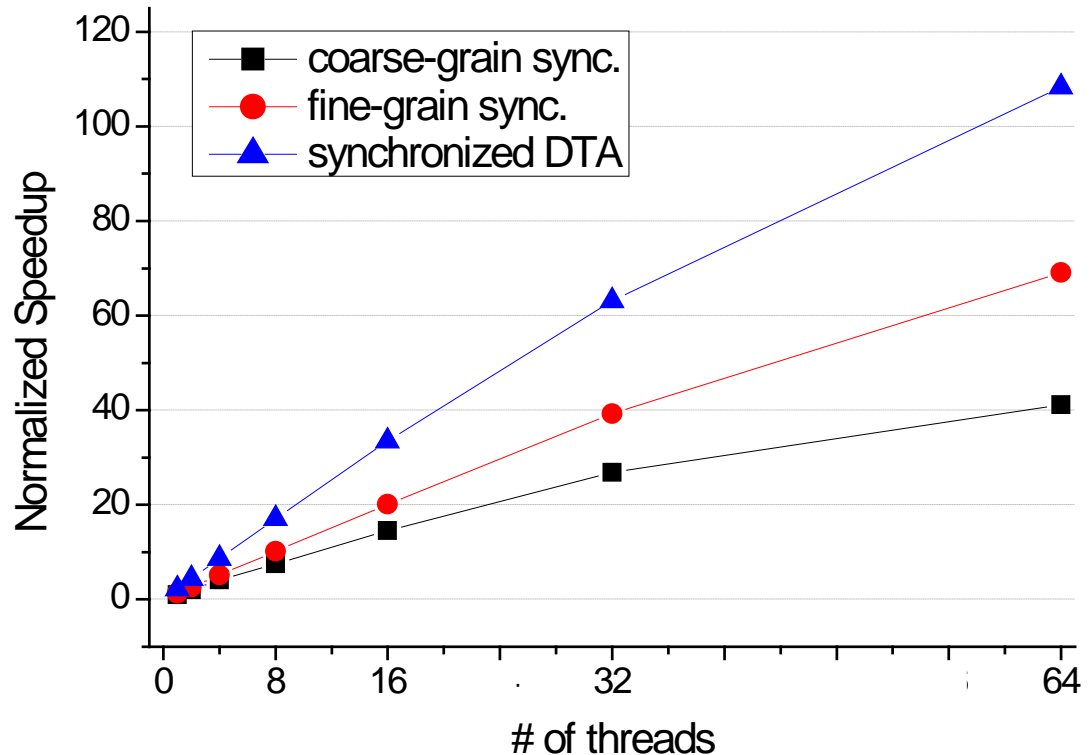
Speedup of 2-D Wavefront

Livermore loop 6

```

for ( i=1 ; i<n ; i++)
  for ( k=0 ; k<i ; k++)
    W[i] += B[k][i] * W[(i-k)-1];

```



Speedup of Livermore Loop 6

Overview



- ✓ **Microprocessor Architecture Challenges**
 - › Memory wall + Power wall + ILP wall
 - › The 5P's: Many-core Processor Challenges
- **Godson-T Features for Parallel Program**
 - › Godson-T architecture overview
 - › Architectural supports for multithreading
 - › Software runtime system
- **Godson-T Design and Implementation**
 - › Journey of design and implementation
 - › Software and hardware co-simulation
 - › Prototype

Pthread-like Thread Execution using Private Memory

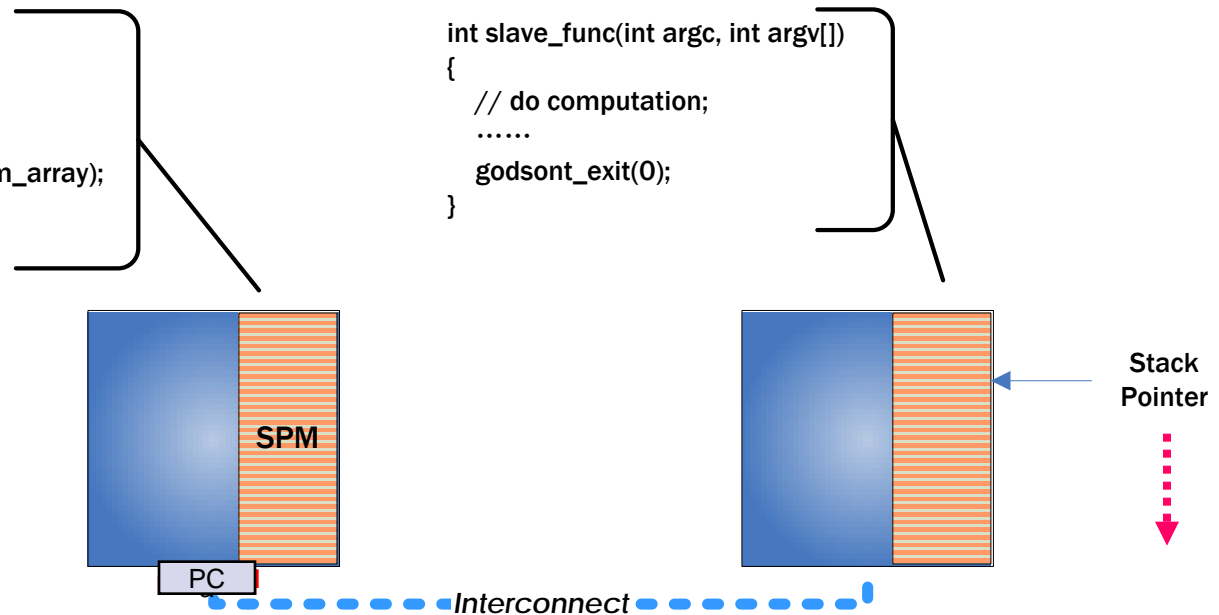


- Support master-slave style thread execution;
- Programming easily: C programming model + Pthread API;
- Compatible with large amount of conventional codes on shared memory system.

```
int mater_func()
{
    .....
    int thread_id =
        godsont_create_thread(slave_func, 2, param_array);
    .....
}
```

```
int slave_func(int argc, int argv[])
{
    // do computation;
    .....
    godsont_exit(0);
}
```

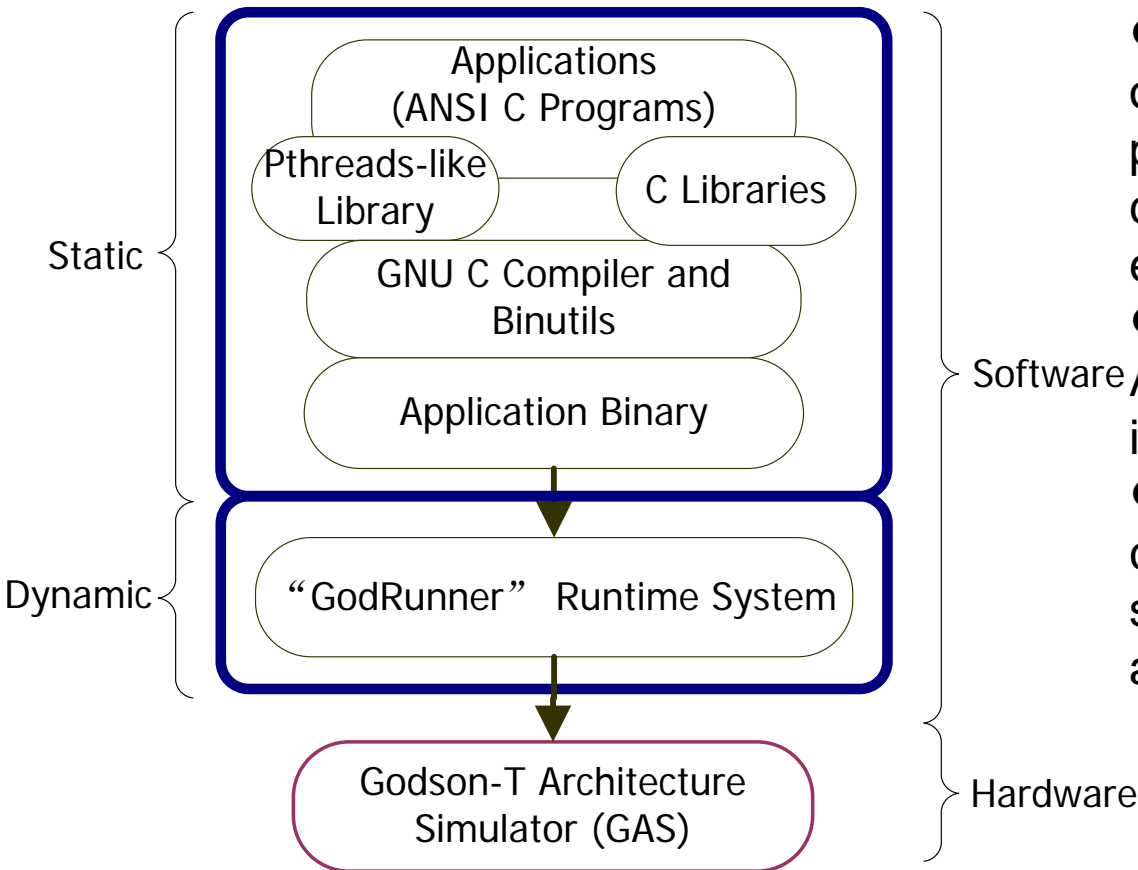
1. Master transmits parameters to slave;
2. Master setups PC and register context for slave;
3. Slave function begins execution, private memory acts as a local stack.



Stack v.s. L1-D\$ accesses ≈ 30 (8x8 matrix multiply, no opt.)

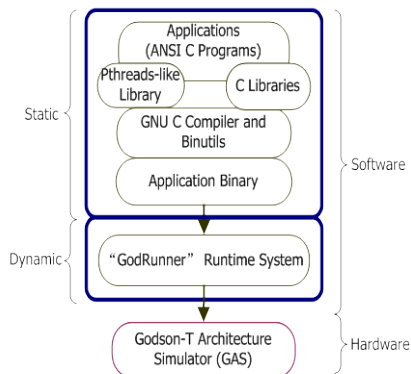
Tens of cycles to create or terminate a thread.

GodRunner Software Stack

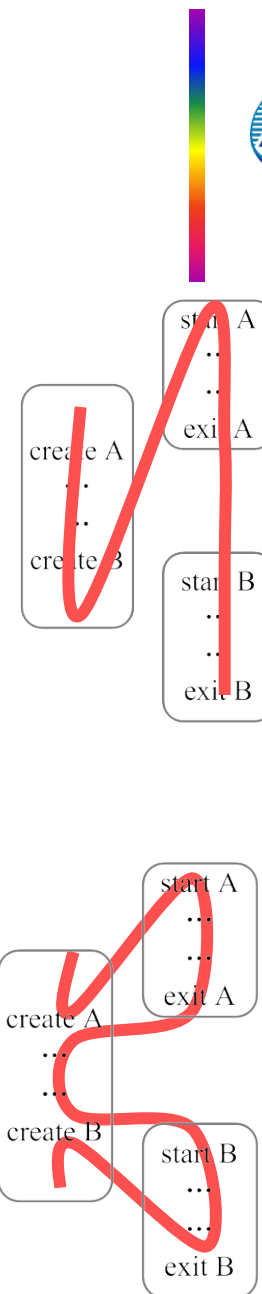
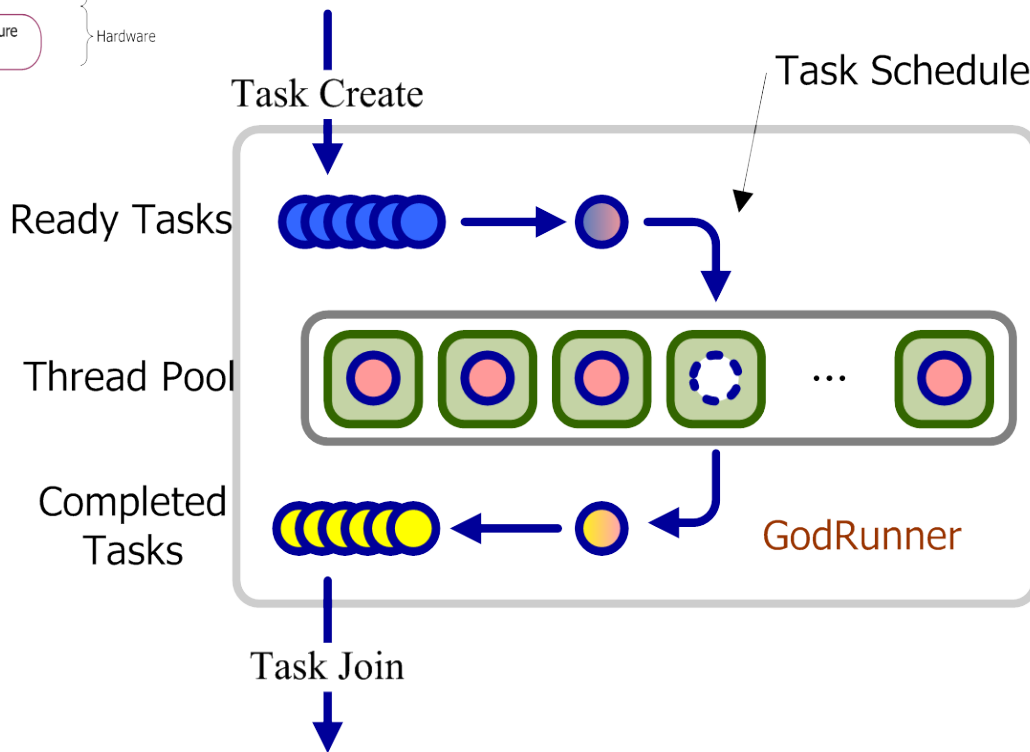


- The programmers majorly focus on expressing parallelism, While the processor and runtime system concentrate on efficient parallel executions.
- The library provides a rich set of APIs for task management, including batch thread management.
- Programmer is responsible for creating,terminating and synchronizing tasks by inserting appropriate Pthreads-like APIs.

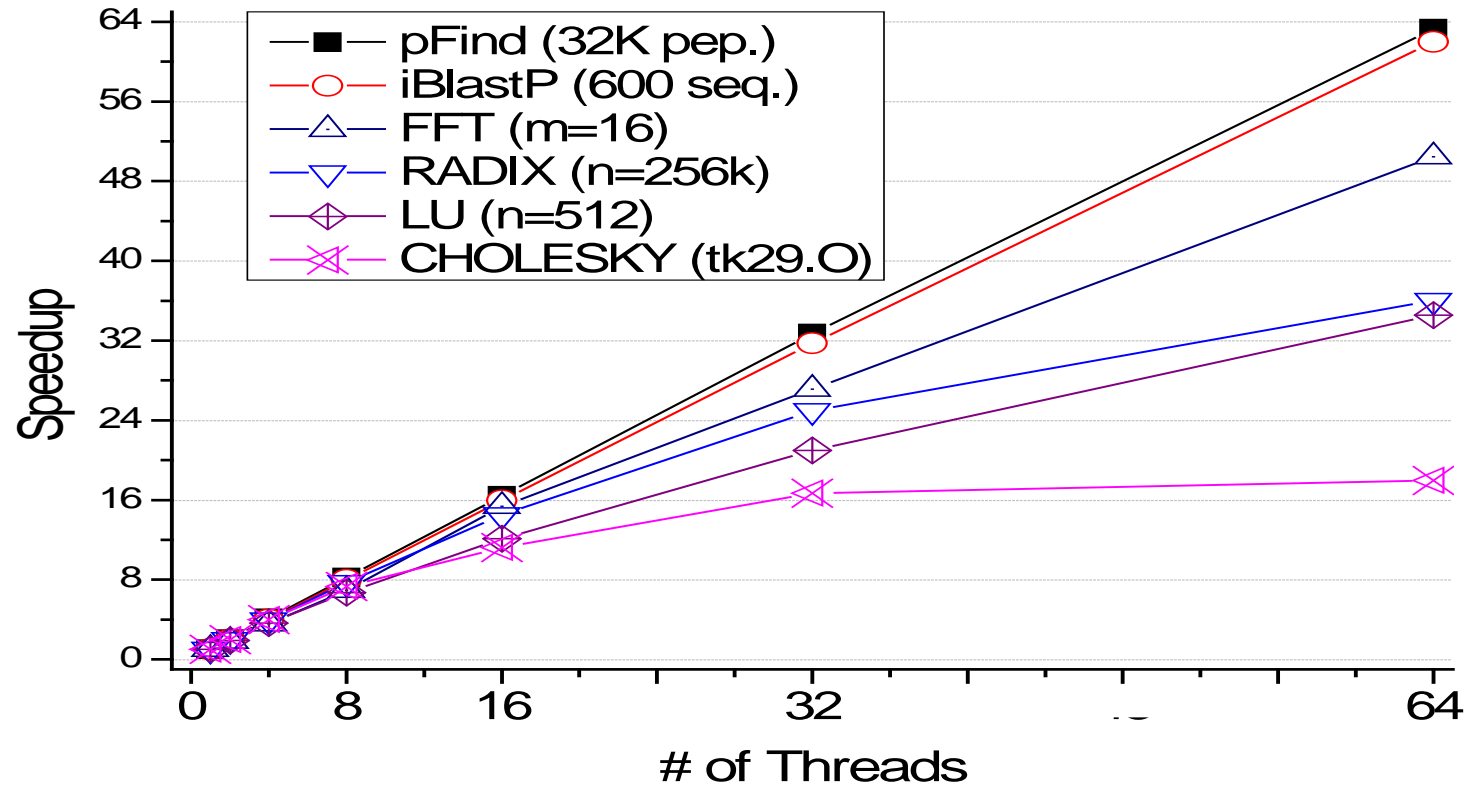
GodRunner Runtime System



● GodRunner permits programmers to create more tasks than hardware thread units, and transparently maps them to hardware thread units at runtime.



Evaluating the Scalability of Performance



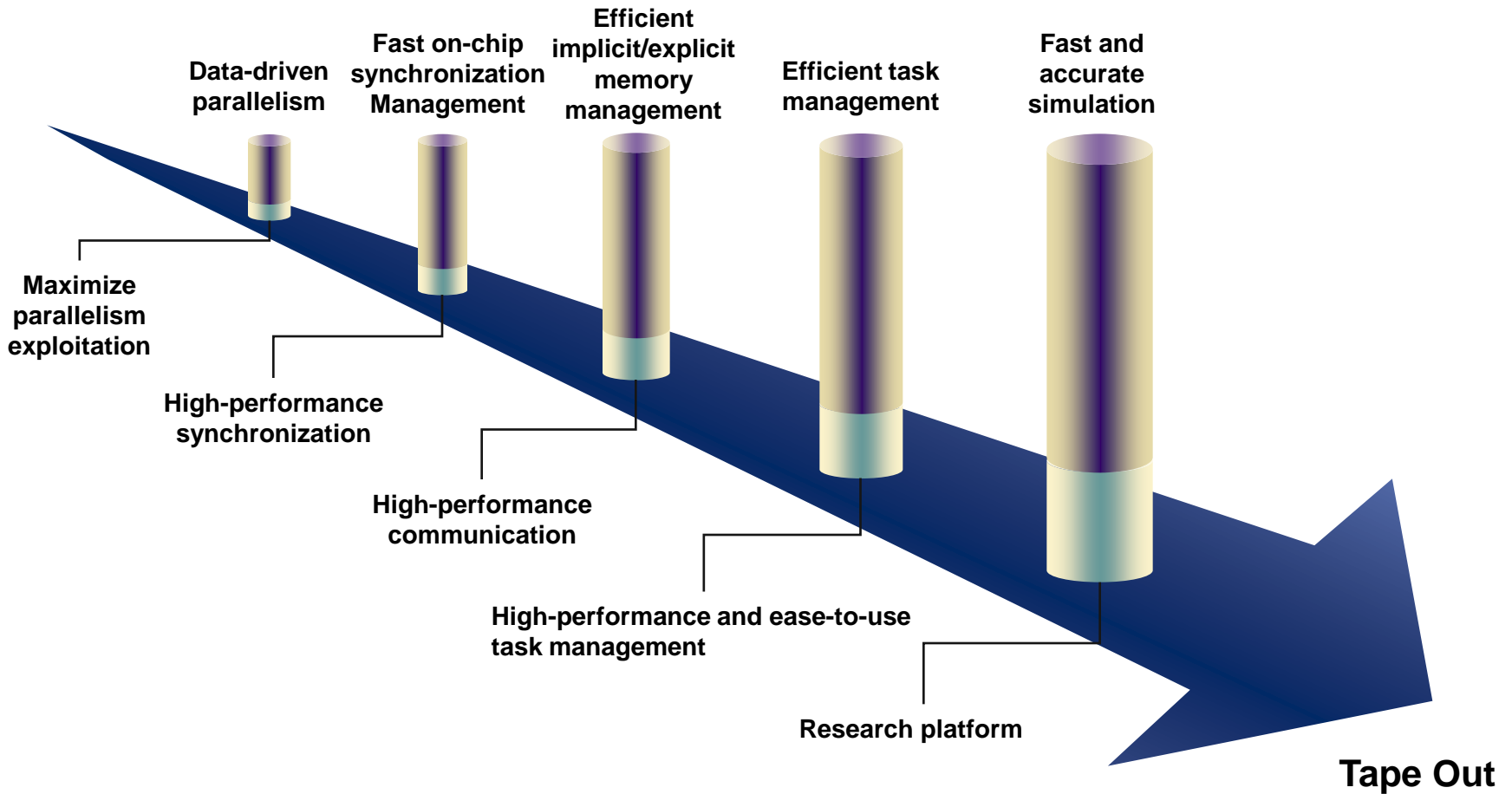
Speedup of the bioinformatics and SPLASH-2 benchmarks on Godson-T

Overview

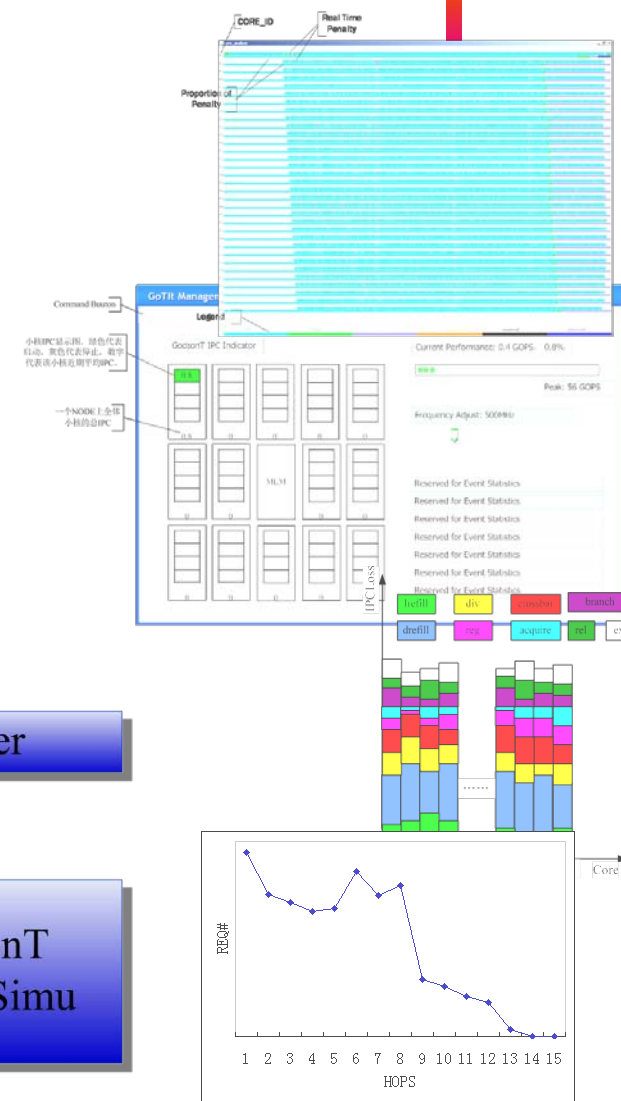
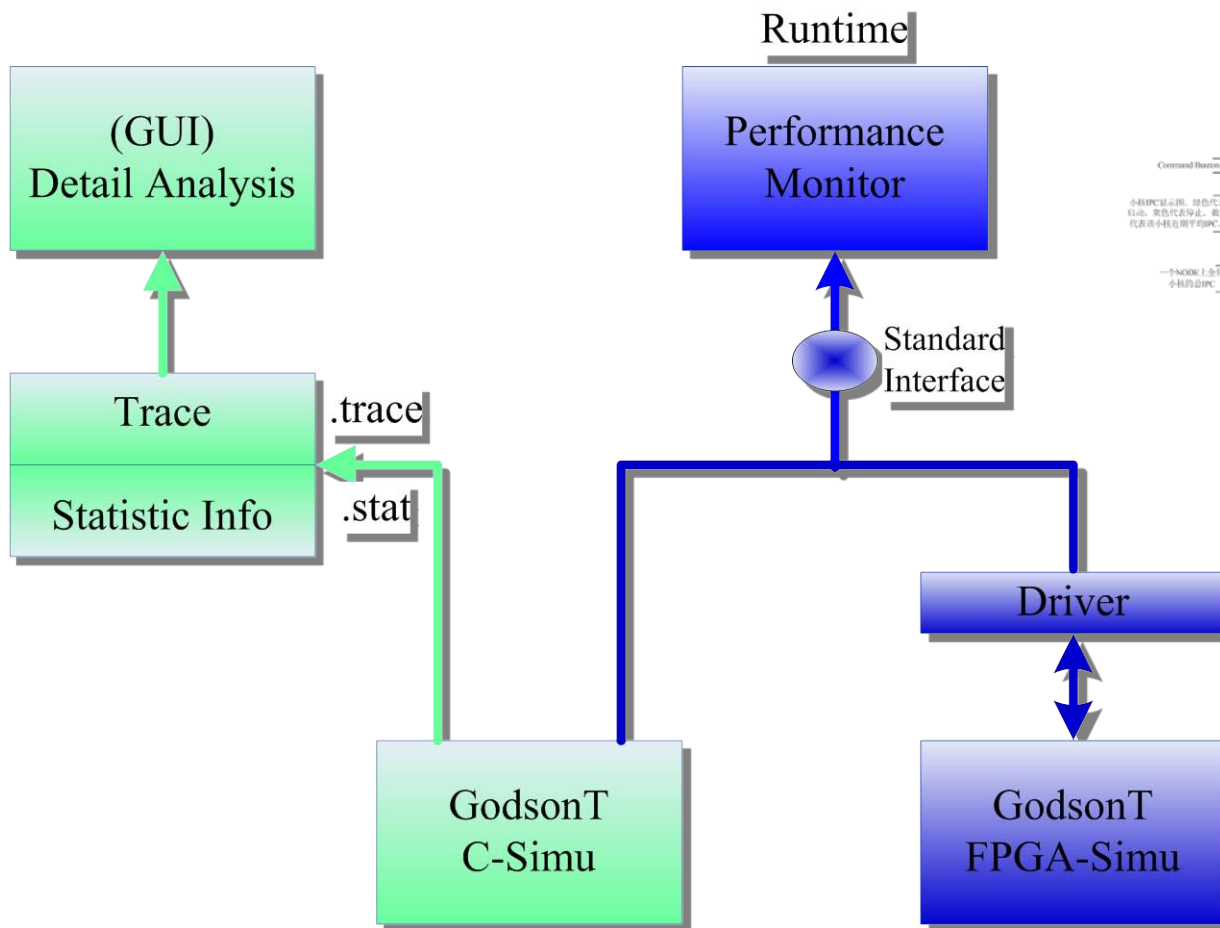


- ✓ **Microprocessor Architecture Challenges**
 - › Memory wall + Power wall + ILP wall
 - › The 5P's: Many-core Processor Challenges
- ✓ **Godson-T Features for Parallel Program**
 - › Godson-T architecture overview
 - › Architectural supports for multithreading
 - › Software runtime system
- **Godson-T Design and Implementation**
 - › Journey of design and implementation
 - › Software and hardware co-simulation
 - › Prototype

Journey of Godson-T

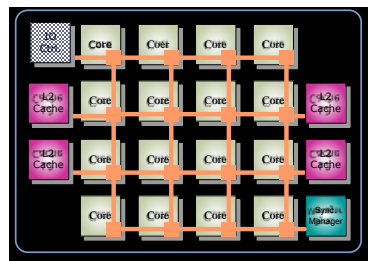


Monitor/Analysis/Evaluation/Platform

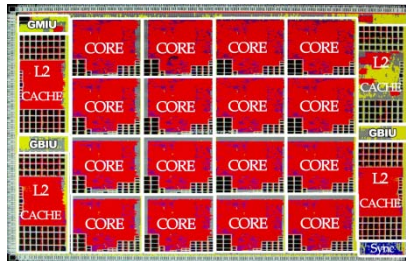


Godson-T Many-Core Prototype

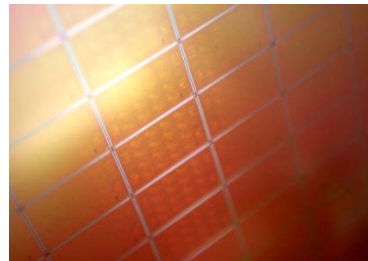
- 64 light-weight processing mini cores currently
- 16-core sample: 130nm, 230mm², SMIC
- Target to domain-specific parallel acceleration



Chip On Mind Concept
RTL



Chip on EDA
PHY Design



Chip on Silicon
Wafer



Chip in Package
Package



Chip on Board
System

Conclusions

- Exploiting parallelism on multiple level
 - Without disturbing DLP, Can merge with DLP processors
 - Exploiting fine-grained data-driven TLP
- A scalable many-core architecture research platform
 - Lock-based cache coherence protocol
 - Asynchronous DTA and hardware-supported synchronization mechanisms
 - Pthreads-like programming model, and versatile parallel libraries
 - Flexible to extend, easy to use, and simple to implement
- Keep balance between software and hardware to gain acceptable programmability and performance

Thank you!

