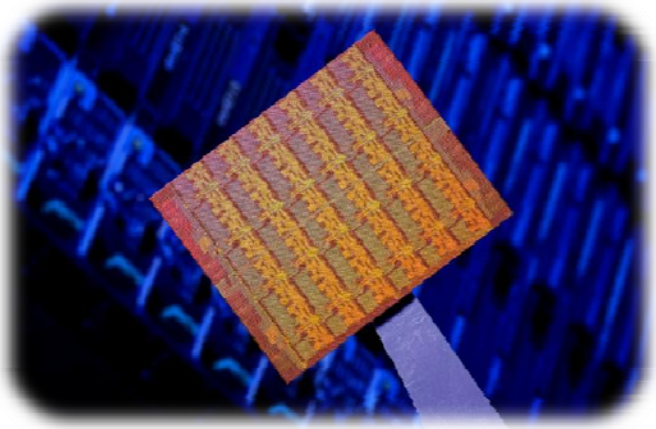
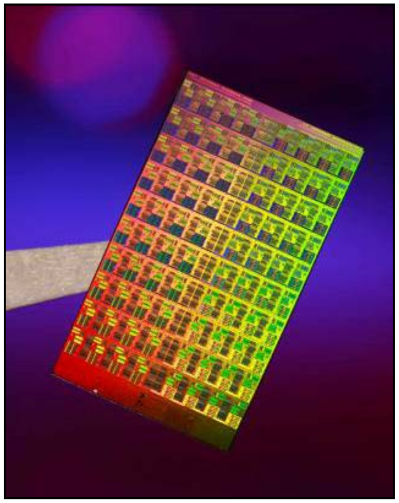


The Future of Many Core Computing:

A tale of two processors



Tim Mattson

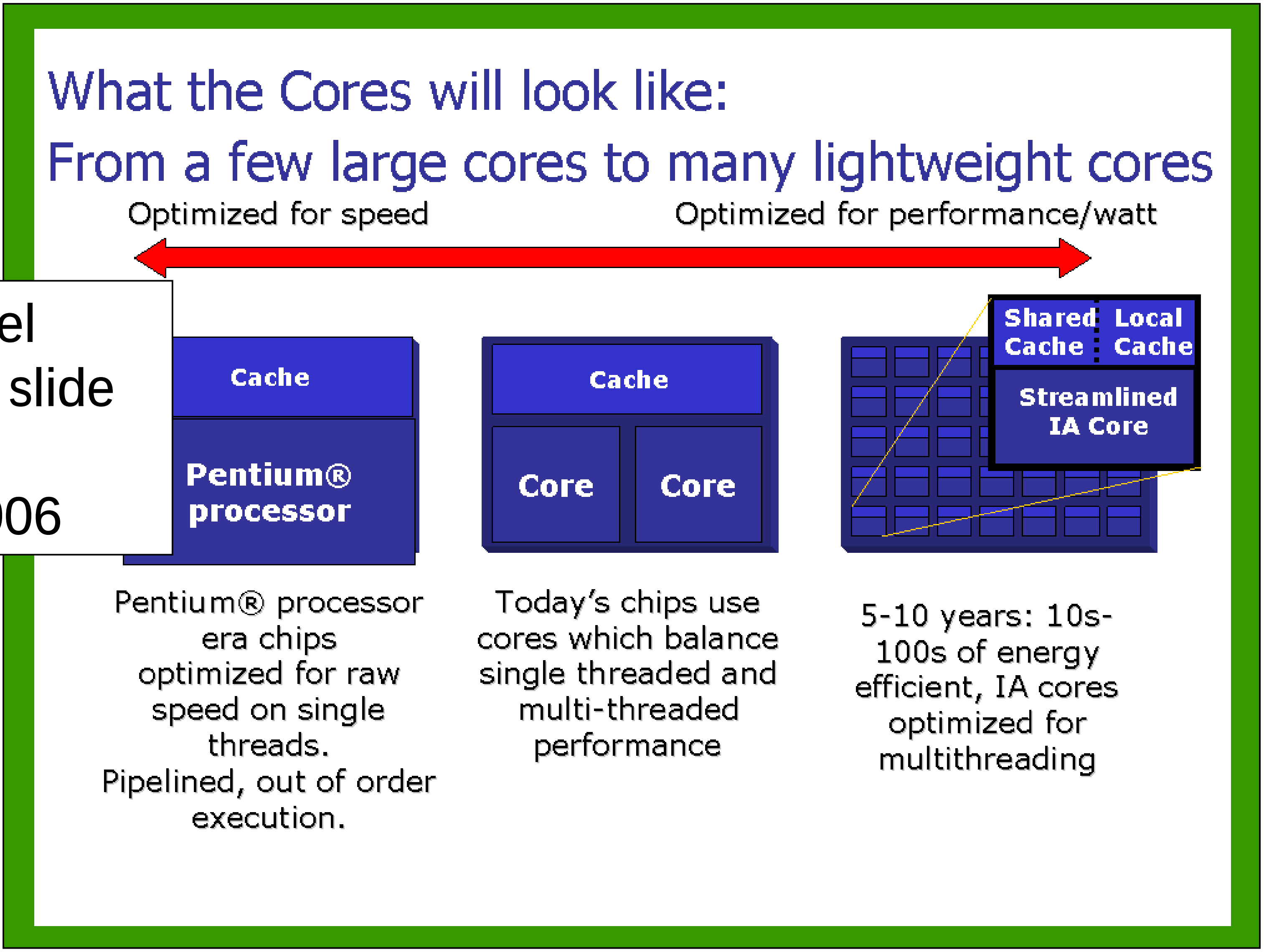
Intel Labs

Disclosure

- The views expressed in this talk are those of the speaker and not his employer.
- I am in a research group and know nothing about Intel products. So anything I say about them is highly suspect.
- This was a team effort, but if I say anything really stupid, it’s all my fault … don’t blame my collaborators.

A common view of many-core chips

An Intel Exec's slide from IDF'2006



What the Cores will look like:

From a few large cores to many lightweight cores

Optimized for speed

Optimized for performance/watt

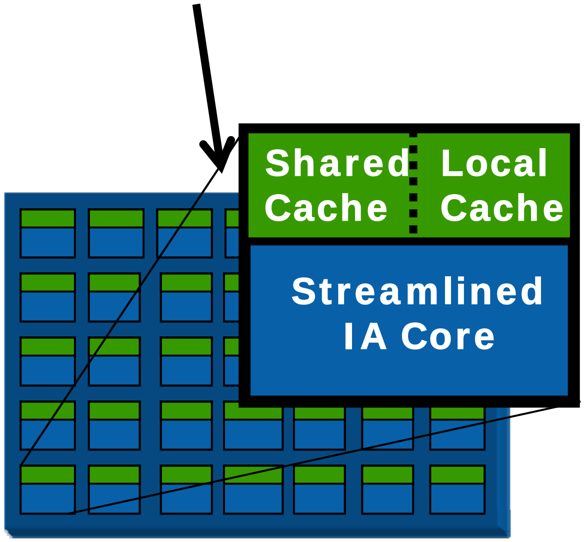
Pentium® processor era chips optimized for raw speed on single threads. Pipelined, out of order execution.

Today's chips use cores which balance single threaded and multi-threaded performance

5-10 years: 10s-100s of energy efficient, IA cores optimized for multithreading

Challenging the sacred cows

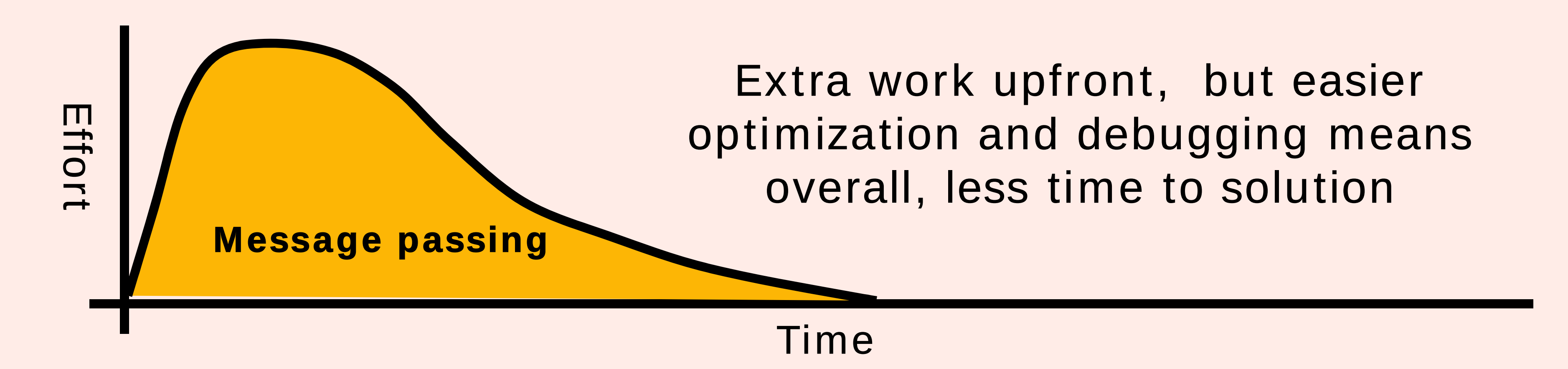
Assumes cache coherent shared address space!



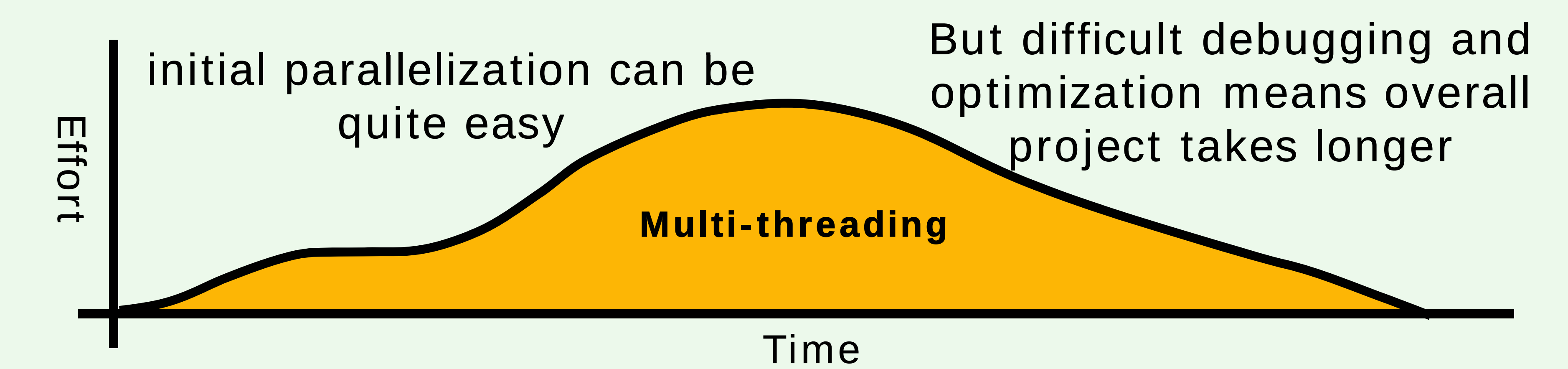
... IA cores optimized for multithreading

- Is that the right choice?
  - Most expert programmers do not fully understand relaxed consistency memory models required to make cache coherent architectures work.
  - The only programming models proven to scale non-trivial apps to 100’s to 1000’s of cores all based on distributed memory.
  - Coherence incurs additional architectural overhead

Isn't shared memory programming easier? Not necessarily.



Extra work upfront, but easier optimization and debugging means overall, less time to solution



initial parallelization can be quite easy

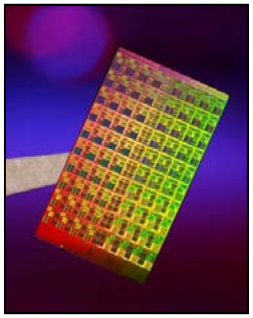
But difficult debugging and optimization means overall project takes longer

Proving that a shared address space program using semaphores is race free is an NP-complete problem*

*P. N. Klein, H. Lu, and R. H. B. Netzer, Detecting Race Conditions in Parallel Programs that Use Semaphores, Algorithmica, vol. 35 pp. 321–345, 2003

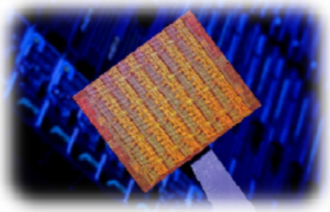
The many core design challenge

- **Scalable architecture**:
  - How should we connect the cores so we can scale as far as we need (O(100's to 1000) should be enough)?
- **Software**:
  - Can "general purpose programmers" write software that takes advantage of the cores?
  - Will ISV's actually write scalable software?
- **Manufacturability**:
  - Validation costs grow steeply as the number of transistors grows. Can we use tiled architectures to address this problem?
    - For an N transistor budget … Validate a tile (M transistors/tile) and the connections between tiles. Drops validation costs from KO(N) to K'O(M) (warning, K, K' can be very large).



80 core Research processor

Intel's "TeraScale" processor research program is addressing these questions with a series of Test chips … two so far.

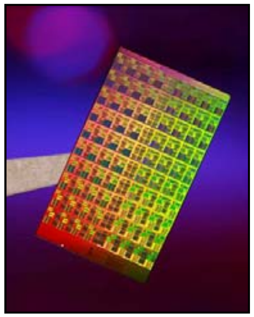


48 core SCC processor

Agenda

- The 80 core Research Processor
  - Max FLOPS/Watt in a tiled architecture
- The 48 core SCC processor
  - Scalable IA cores for software/platform research

Agenda



- The 80 core Research Processor
  - Max FLOPS/Watt in a tiled architecture
- The 48 core SCC processor
  - Scalable IA cores for software/platform research

Sriram R. Vangal, Jason Howard, Gregory Ruhl, Member, Saurabh Dighe, Howard Wilson, James Tschanz, David Finan, Arvind Singh, Member, Tiju Jacob, Shailendra Jain, Vasantha Erraguntla, Clark Roberts, Yatin Hoskote, Nitin Borkar, and S. Borkar,
"**An 80-Tile Sub-100-W TeraFLOPS Processor in 65-nm CMOS**,"
IEEE Journal of Solid-State Circuits, Vol. 43, No. 1, Jan 2008.

Tim Mattson, Rob van der Wijngaart, Michael Frumkin,
“**Programming Intel's 80 core terascale processor**,”
Proc. of the 2008 ACM/IEEE conference on Supercomputing , SC08, Austin Texas, Nov. 2008

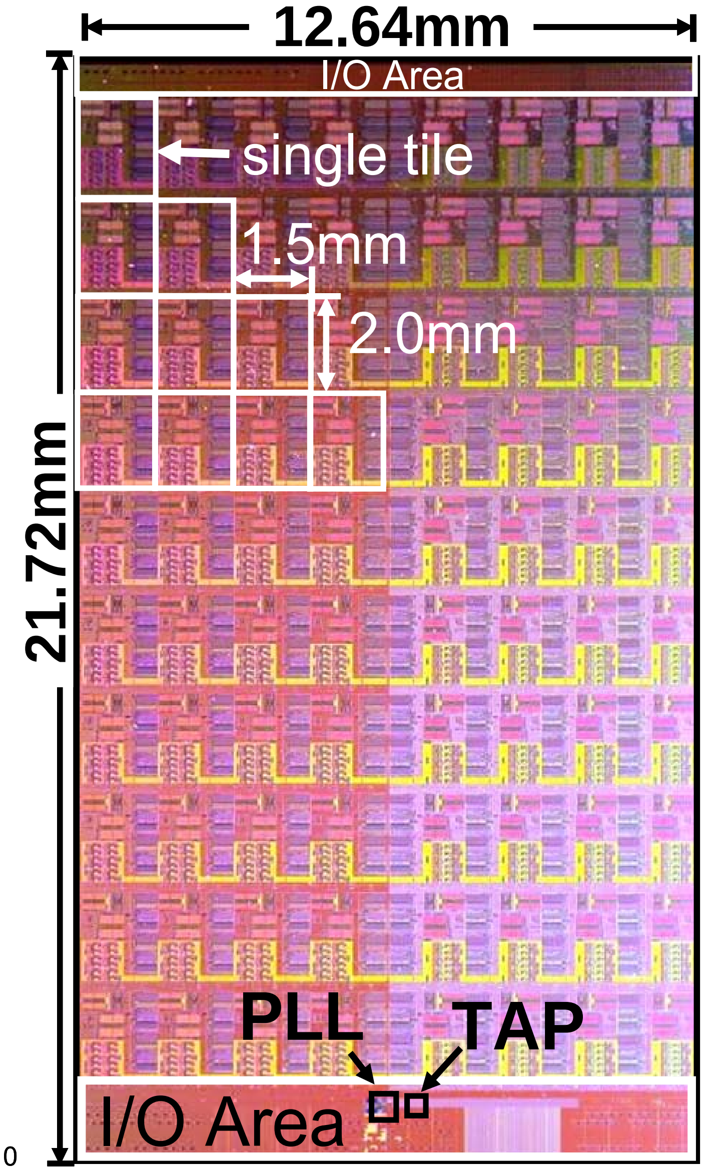
Acknowledgements

- Implementation
  - Circuit Research Lab Advanced Prototyping team (Hillsboro, OR and Bangalore, India)
- PLL design
  - Logic Technology Development (Hillsboro, OR)
- Package design
  - Assembly Technology Development (Chandler, AZ)
- The software team
  - Tim Mattson, Rob van der Wijngaart (Intel)
  - Michael Frumkin (then at Intel, now at Google)

A special thanks to our “optimizing compiler” … Yatin Hoskote, Jason Howard, and Saurabh Dighe of Intel’s Microprocessor Technology Laboratory.

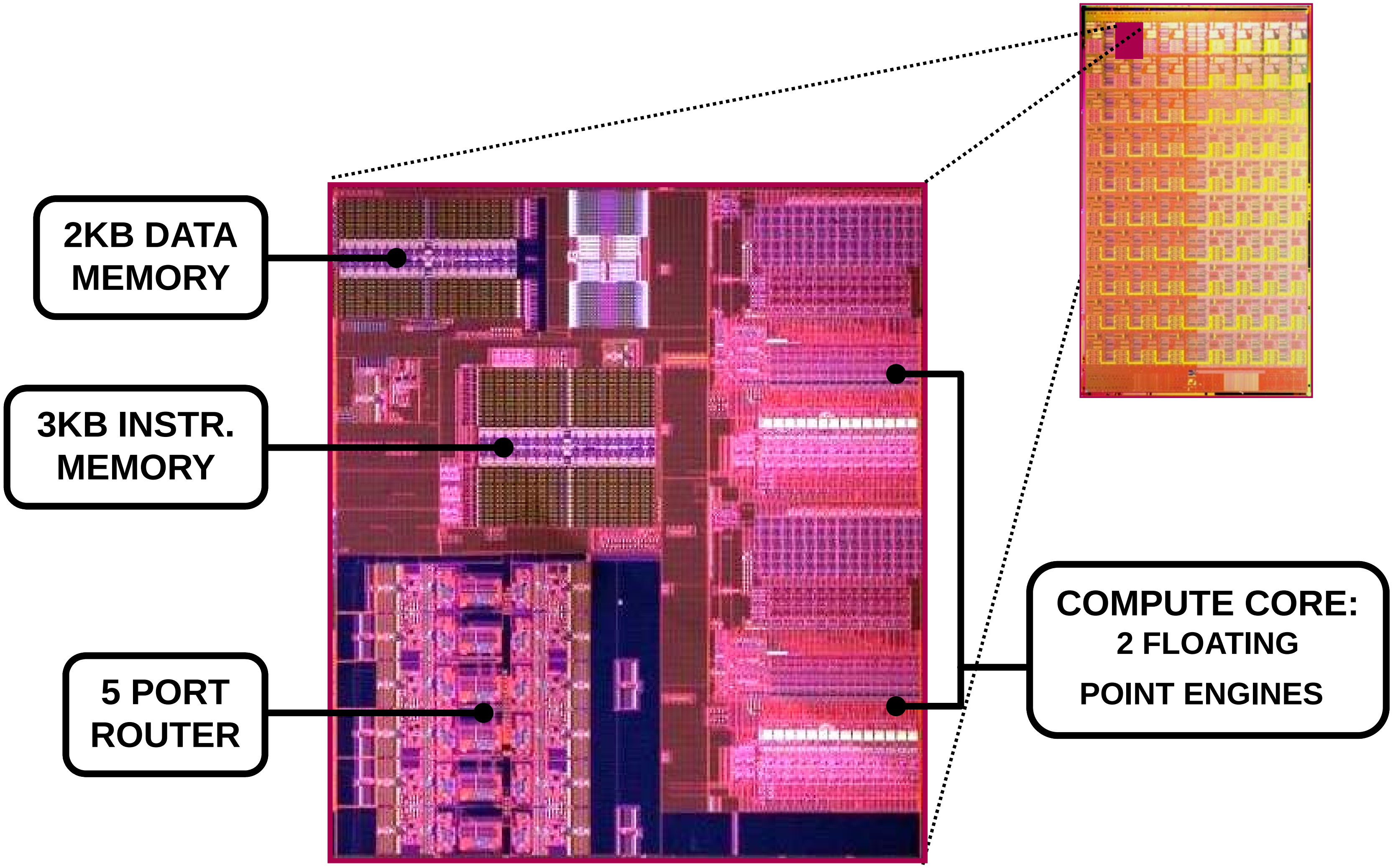
Intel's 80 core terascale processor

Die Photo and Chip Details



- Basic statistics:
  - 65 nm CMOS process
  - 100 Million transistors in 275 mm^2
  - 8x10 tiles, 3mm^2/tile
  - Mesosynchronous clock
  - 1.6 SP TFLOP @ 5 Ghz and 1.2 V
  - 320 GB/s bisection bandwidth
  - Variable voltage and multiple sleep states for explicit power management

We’ve made good progress with the hardware: Intel’s 80 core test chip (2006)



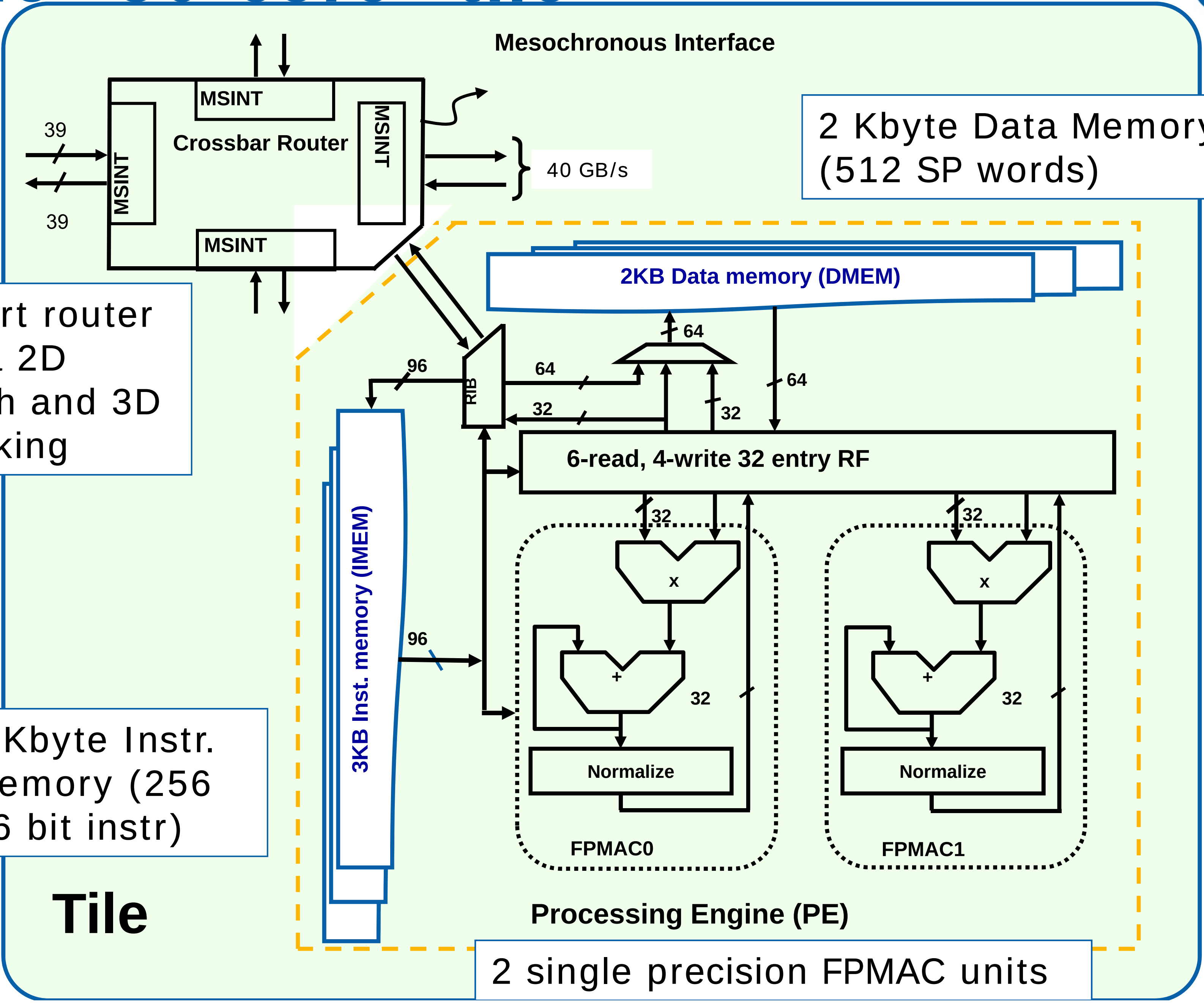
2KB DATA MEMORY

3KB INSTR. MEMORY

5 PORT ROUTER

COMPUTE CORE: 2 FLOATING POINT ENGINES

The “80-core” tile



2 Kbyte Data Memory (512 SP words)

5 port router for a 2D mesh and 3D stacking

3 Kbyte Instr. Memory (256 96 bit instr)

2 single precision FPMAC units

Programmer's perspective

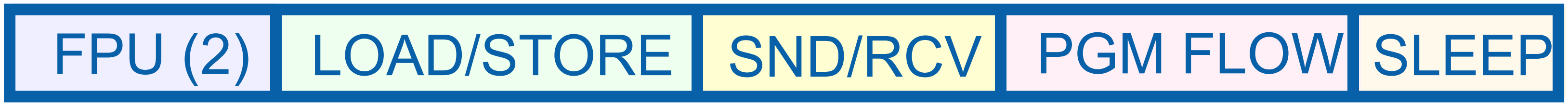
- **8x10 mesh of 80 cores**
- **All memory on-chip**
  - 256 instructions operating
  - 512 floating point numbers.
  - 32 SP registers, two loads per cycle per tile
- **Compute engine**
  - 2 SP FMAC units per tile → 4 FLOP/cycle/tile
  - 9-stage pipeline
- **Communication**
  - One sided anonymous message passing into instruction or data memory
- **Limitations:**
  - No division
  - No general branch, single branch-on-zero (single loop)
  - No wimps allowed! ... i.e. No compiler, Debugger, OS, I/O ...

SP = single precision, FMAC = floating point multiply accumulate, FLOP = floating point operations

Full Instruction Set

MULT	Multiply operands	FPU
ACCUM	Accumulate with previous result	
LOAD, STORE	Move a pair of floats between register file & data memory.	Load/Store
LOADO, STOREO, OFFSET	Move a pair of floats between the register file and data memory at address plus OFFSET.	
SENDI[H|A|D|T]	Send instr. header, address, data, and tail	SND/RCV
SENDD[H|A|D|T]	Send Data header, address, data, and tail	
WFD	Stall while waiting for data from any tile.	
STALL	Stall program counter (PC), waiting for a new PC.	
BRNE, INDEX	INDEX sets a register for loop count. BRNE branches while the index register is greater than zero	Program flow
JUMP	Jump to the specified program counter address	
NAP	Put FPUs to sleep	Sleep
WAKE	Wake FPUs from sleep	

Instruction word and latencies



FPU (2) | LOAD/STORE | SND/RCV | PGM FLOW | SLEEP

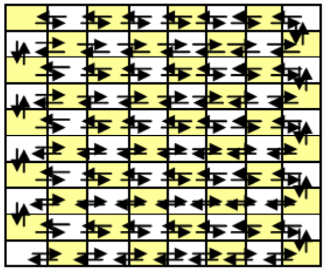
- 96-bit instruction word, up to 8 operations/cycle

Instruction Type	Latency (cycles)
FPU	9
LOAD/STORE	2
SEND/RECEIVE	2
JUMP/BRANCH	1
NAP/WAKE	1

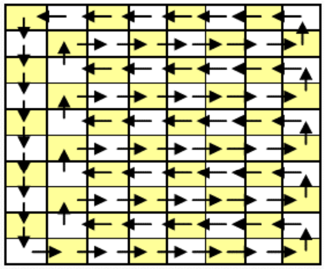
What did we do with the chip?

Communication Patterns

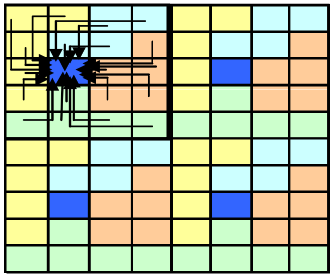
- 4 applications kernels
  - **Stencil**
    - 2D PDE solver (heat diffusion equation) using a Gauss Seidel algorithm



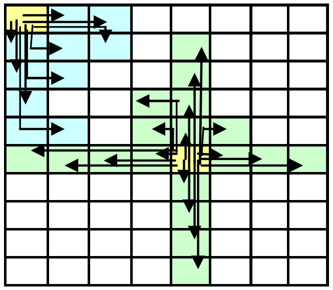
  - **SGEMM (Matrix Multiply)**
    - C = A*B with rectangular matrices



  - **Spreadsheet**
    - Synthetic benchmark … sum dense array of rows and columns (local sums in one D, reduction in the other D)

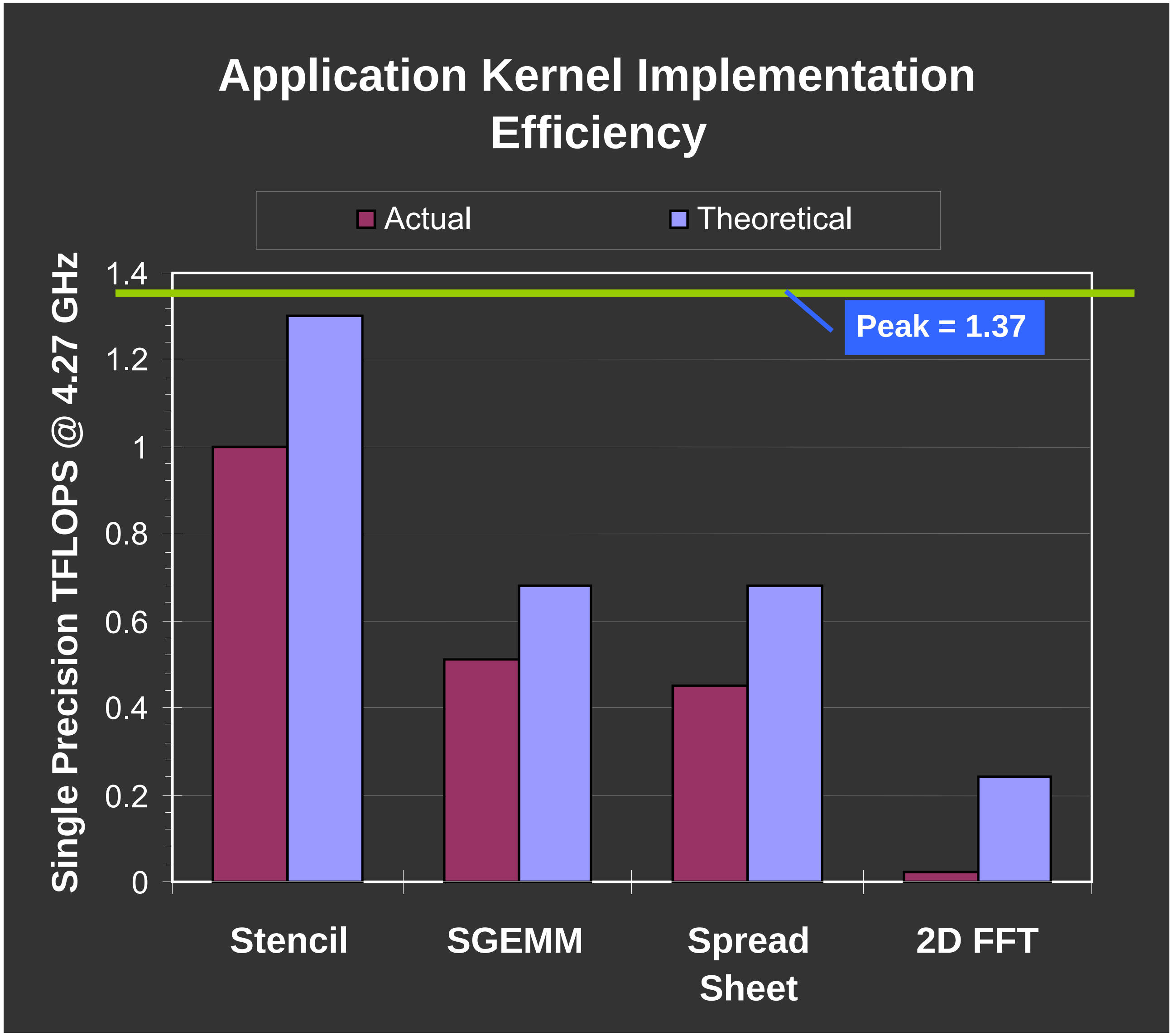


  - **2D FFT**
    - 2D FFT of dense array on an 8 by 8 subgrid.



These kernels were hand coded in assembly code and manually optimized. Data sets sized to fill data memory.

Programming Results

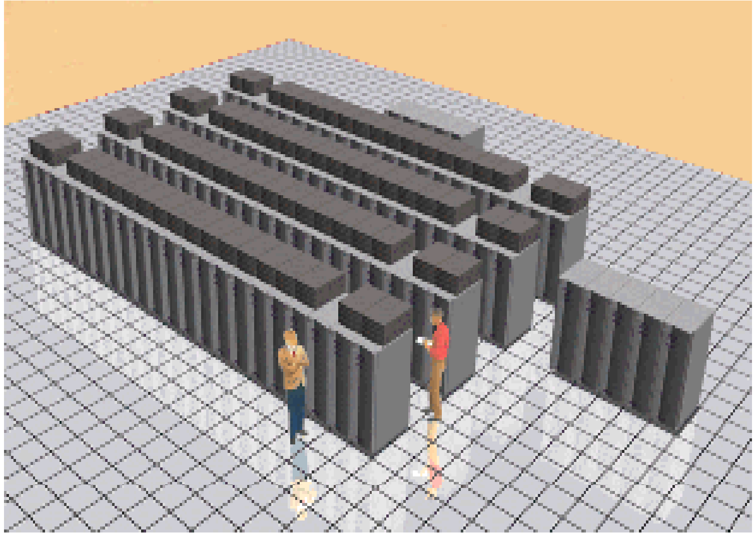


Theoretical numbers from operation / communication counts and from rate limiting bandwidths.

1.07V, 4.27GHz operation 80 C

Why this is so exciting!

First TeraScale* computer: 1997



Intel’s ASCI Option Red

Intel’s ASCI Red Supercomputer

9000 CPUs

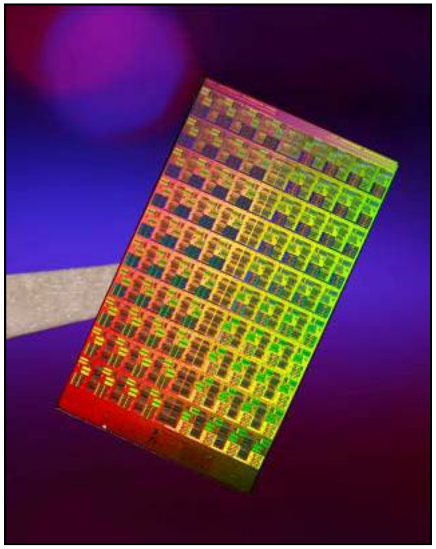
one megawatt of electricity.

1600 square feet of floor space.

*Double Precision TFLOPS running MP-Linpack

10 years later

First TeraScale% chip: 2007



Intel’s 80 core teraScale Chip

1 CPU

97 watt

275 mm2

%Single Precision TFLOPS running stencil

A TeraFLOP in 1996: The ASCI TeraFLOP Supercomputer, Proceedings of the International Parallel Processing Symposium (1996), T.G. Mattson, D. Scott and S. Wheat.

Source: Intel

Lessons: Part 1

- What should we do with our huge transistor counts
  - A fraction of the transistor budget should be used for on-die memory.
  - The 80-core Terascale Processor with its on-die memory has a 2 cycle latency for load/store operations ... this compares to ~100 nsec access to DRAM.
  - As core counts increase, the need for on-chip memory will grow!
  - For Power/Performance, specialized cores rule!
- What role should Caches play?
  - This NoC design lacked caches.
  - Cache coherence limits scalability:
    - Coherence traffic may collide with useful communication.
    - Increases overhead … Due to Amdahl's law, A chip with on the order of 100 cores would be severely impacted by even a small overhead ~1%

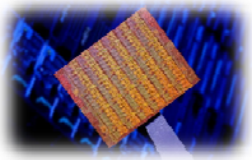
Lessons: Part 2

- Minimize message passing overhead.
  - Routers wrote directly into memory without interrupting computing … i.e. any core could write directly into the memory of any other core. This led to extremely small comm. latency on the order of 2 cycles.
- Programmers can assist in keeping power low if sleep/wake instructions are exposed and if switching latency is low (~ a couple cycles).

- Application programmers should help design chips
  - This chip was presented to us a completed package.
  - Small changes to the instruction set could have had a large impact on the programmability of the chip.
    - A simple computed jump statement would have allowed us to add nested loops.
    - A second offset parameter would have allowed us to program general 2D array computations.

Agenda

- The 80 core Research Processor
  - Max FLOPS/Watt in a tiled architecture
- The 48 core SCC processor
  - Scalable IA cores for software/platform research



J. Howard, S. Dighe, Y. Hoskote, S. Vangal, D. Finan, G. Ruhl, D. Jenkins, H. Wilson, N. Borkar, G. Schrom, F. Pailet, S. Jain, T. Jacob, S. Yada, S. Marella, P. Salihundam, V. Erraguntla, M. Konow, M. Riepen, G. Droege, J. Lindemann, M. Gries, T. Apel, K. Henriss, T. Lund-Larsen, S. Steibl, S. Borkar, V. De1, R. Van Der Wijngaart, T. Mattson,
"**A 48-Core IA-32 Message-Passing Processor with DVFS in 45nm CMOS**",
Proceedings of the International Solid-State Circuits Conference, Feb 2010

Timothy G. Mattson, Rob F. Van der Wijngaart, Michael Riepen, Thomas Lehnig, Paul Brett, Werner Haas, Patrick Kennedy, Jason Howard, Sriram Vangal, Nitin Borkar, Greg Ruhl, Saurabh Dighe
"**The 48 core SCC Processor: A programmers view**"
Submitted to Proc. of the 2010 ACM/IEEE Conference on Supercomputing

Acknowledgements

- SCC Application software:

RCCE library and apps and HW/SW co-design	Rob Van der Wijngaart Tim Mattson
Developer tools (icc and MKL)	Patrick Kennedy

- SCC System software:

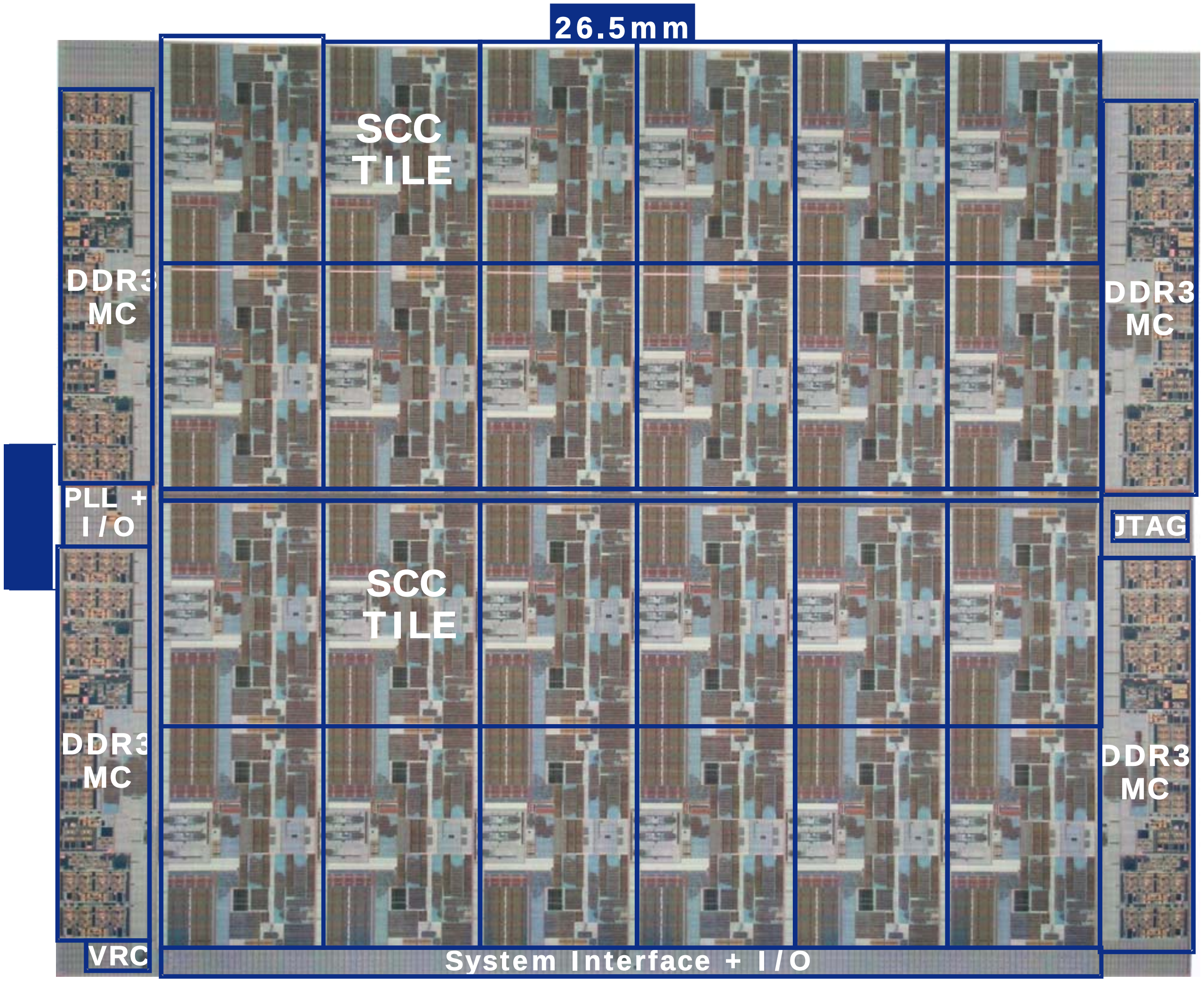
Management Console software and BareMetalC workflow	Michael Riepen
Linux for SCC	Thomas Lehnig Paul Brett
System Interface FPGA development	Matthias Steidl
TCP/IP network drivers	Werner Haas

- And the HW-team that worked closely with the SW group:

Jason Howard, Yatin Hoskote, Sriram Vangal, Nitin Borkar, Greg Ruhl

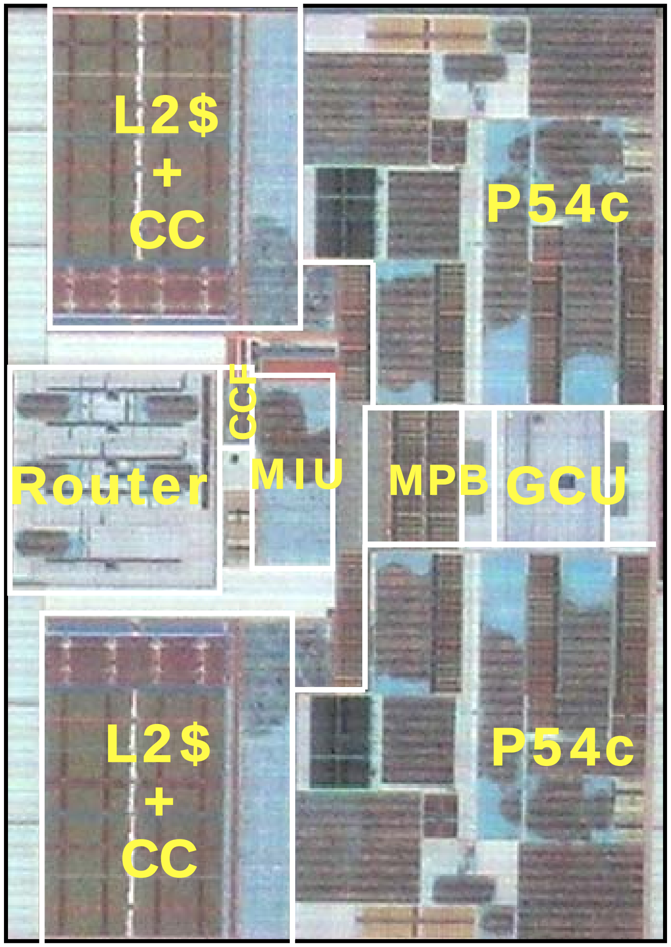
SCC full chip

- 24 tiles in 6x4 mesh with 2 cores per tile (48 cores total).



Technology	45nm Process
Interconnect	1 Poly, 9 Metal (Cu)
Transistors	Die: 1.3B, Tile: 48M
Tile Area	18.7mm^2
Die Area	567.1mm^2

SCC Dual-core Tile

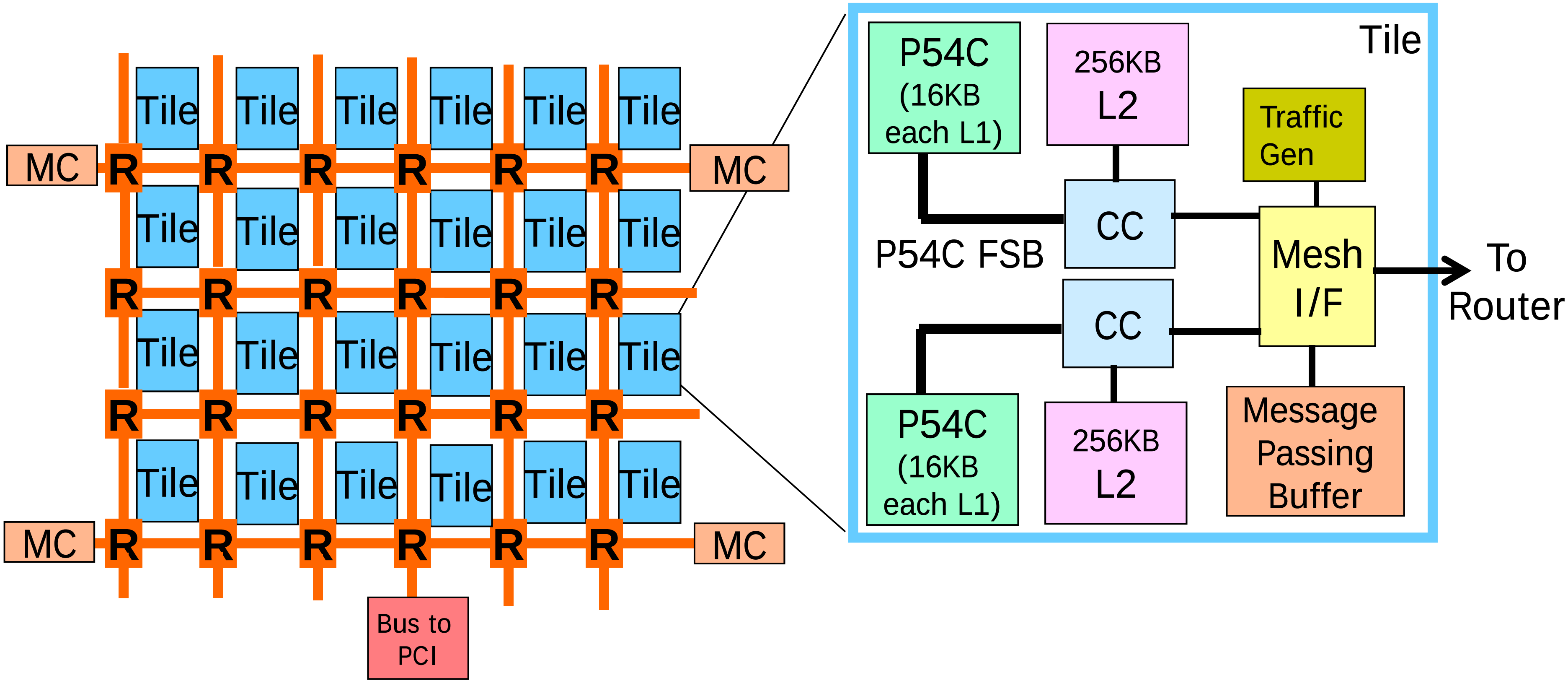


- 2 P54C cores (16K L1$/core)
- 256K L2$ per core
- 8K Message passing buffer
- Clock Crossing FIFOs b/w Mesh interface unit and Router

- Tile area 18.7mm^2
- Core are 3.9mm^2
- Cores and uncore units @1GHz
- Router @2GHz

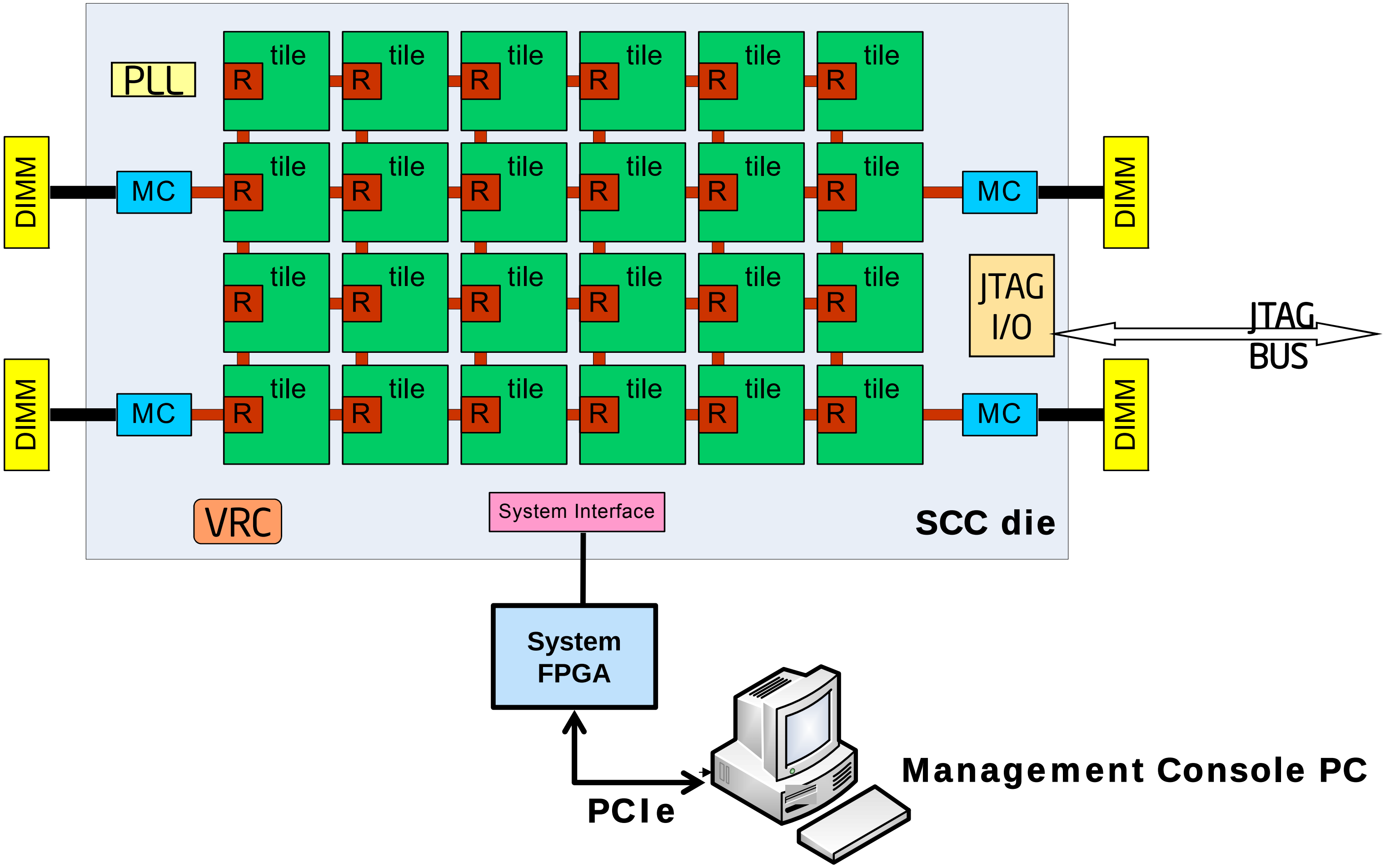
Hardware view of SCC

- 48 P54C cores in 6x4 mesh with 2 cores per tile
- 45 nm, 1.3 B transistors, 25 to 125 W
- 16 to 64 GB total main memory using 4 DDR3 MCs
- 2 Tb/s bisection bandwidth @ 2 Ghz



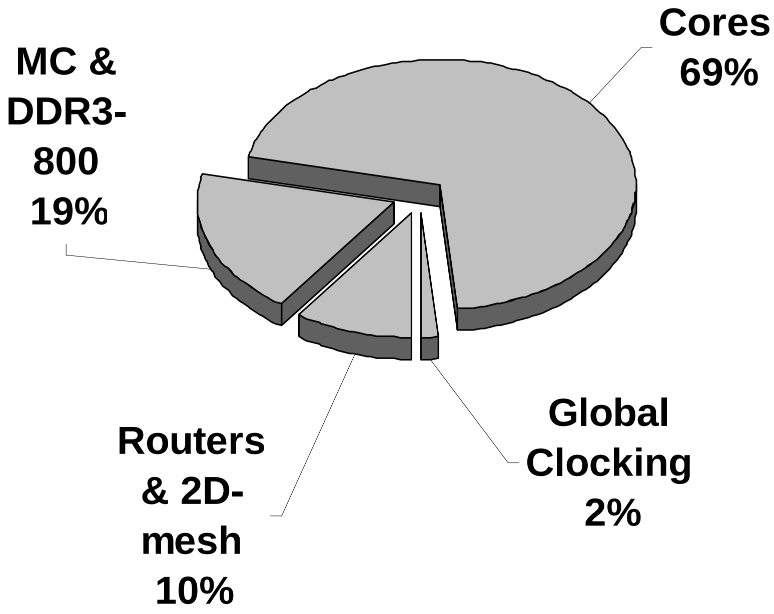
R = router, MC = Memory Controller, P54C = second generation Pentium core, CC = cache cntrl.

SCC system overview



Power breakdown

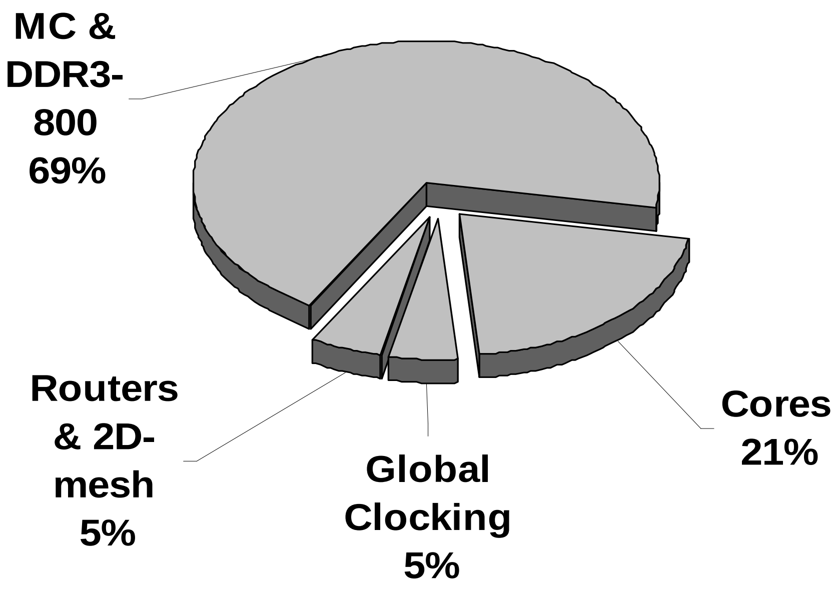
Full Power Breakdown Total -125.3W



Clocking: 1.9W Routers: 12.1W
Cores: 87.7W MCs: 23.6W

Cores-1GHz, Mesh-2GHz, 1.14V, 50°C

Low Power Breakdown Total - 24.7W



Clocking: 1.2W Routers: 1.2W
Cores: 5.1W MCs: 17.2W

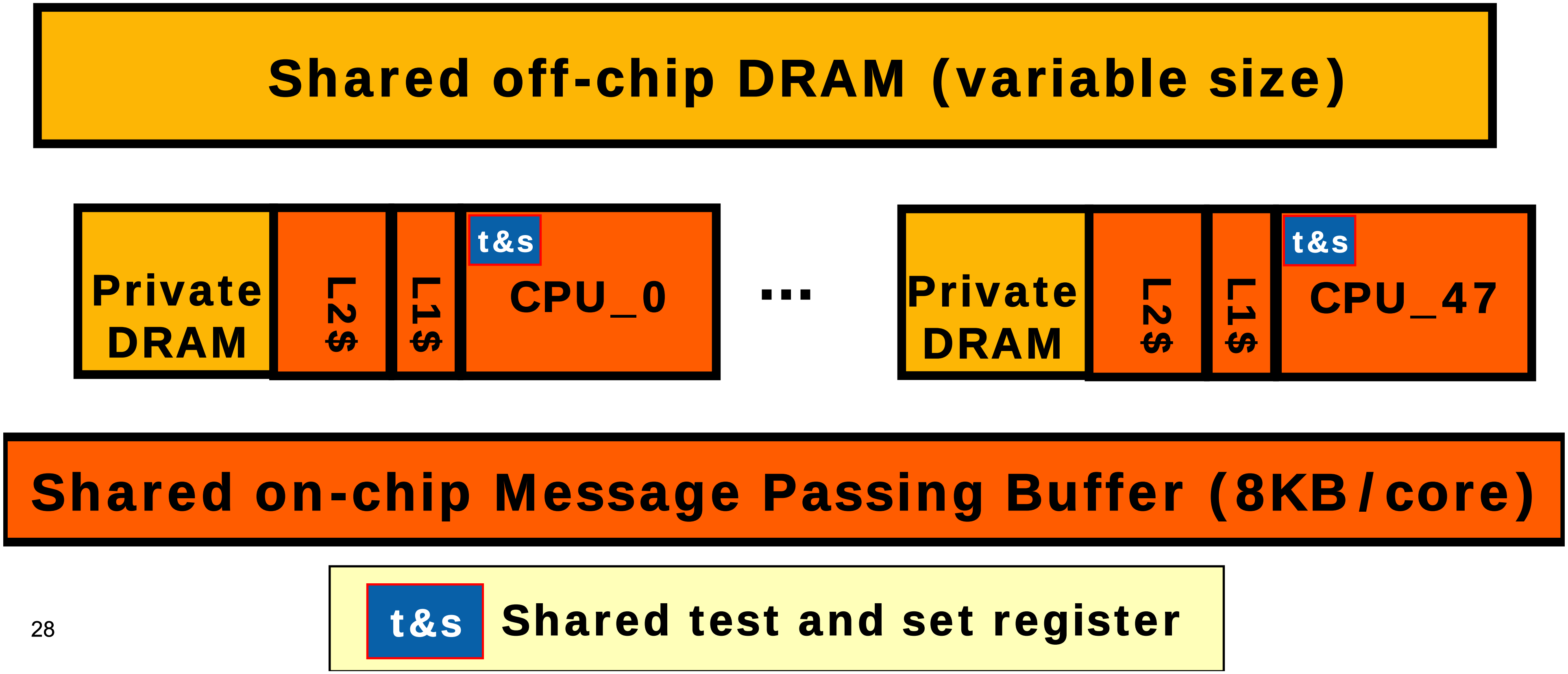
Cores-125MHz, Mesh-250MHz, 0.7V, 50°C

SCC Address spaces

- 48 x86 cores which use the x86 memory model for Private DRAM

🟧 On-chip
🟨 Off-chip
Memory

Where is the physical Memory



Shared off-chip DRAM (variable size)

Private DRAM | L2$ | L1$ | t&s | CPU_0

…

Private DRAM | L2$ | L1$ | t&s | CPU_47

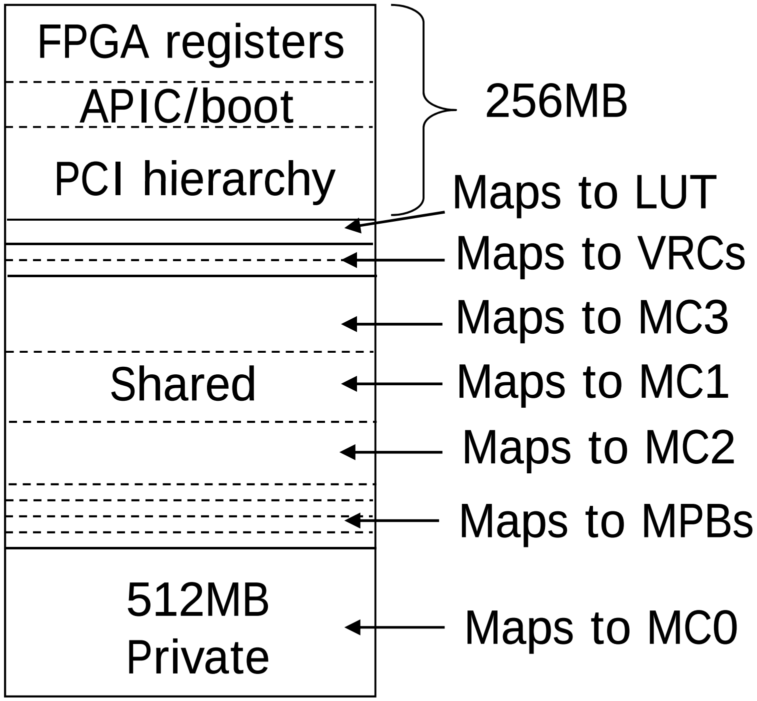
Shared on-chip Message Passing Buffer (8KB/core)

t&s Shared test and set register

SCC Memory Management

- Memory is managed through a Look-up Table (LUT) address translation … each core manages its own programmable LUT.

- LUT Table divides a core's 4 GByte memory space into 256 16MB pages:
  - Control registers including voltage Regulator control (VRC).
  - Shared space seen by all cores … through each memory controller (MC0 to MC3)
  - Private DRAM (MC0)
  - Message Passing Buffer (MPB)
- Private DRAM follows regular P54C memory model (L1$, L2$, DRAM).
- There is NEVER cache coherence between cores …



MC# = one of the 4 memory controllers, MPB = message passing buffer, VRC's = Voltage Regulator control

Impact of Core Position on Memory Performance

- Stream benchmark mapped to one core and MC channel
  - Position of core is varied
  - Report relative reduction in usable memory bandwidth

- **Tile 533 MHz, Router 800 MHz, Memory controller 800 MHz**
  - Up to 13% variation within quadrant of Memory controller (iMC)

	-13.0%	-15.8%	-19.8%	-23.0%	-25.5%	-28.7%	
iMC	-8.3%	-13.0%	-15.8%	-19.8%	-23.0%	-25.5%	iMC
	-5.3%	-8.3%	-13.0%	-15.8%	-19.8%	-23.0%	
iMC	0.0%	-5.3%	-8.4%	-13.0%	-15.8%	-19.8%	iMC

- **Tile 800 MHz, Router 1600 MHz, Memory controller 800 MHz**
  - Up to 8% variation within quadrant of Memory controller

	-8.0%	-8.9%	-11.6%	-14.9%	-15.6%	-18.0%	
iMC	-4.2%	-8.0%	-8.9%	-11.6%	-14.9%	-15.6%	iMC
	-1.0%	-4.2%	-8.0%	-8.8%	-11.6%	-14.9%	
iMC	0.0%	-1.0%	-4.2%	-8.0%	-8.8%	-11.6%	iMC

Source: Intel, SCC workshop, Germany March 16 2010

SCC’s message passing library: RCCE

- RCCE is a compact, lightweight communication environment.
  - SCC and RCCE were designed together side by side:
    - … a true HW/SW co-design project.
- RCCE is a research vehicle to understand how message passing APIs map onto many core chips.
- RCCE is for experienced parallel programmers willing to work close to the hardware.
- RCCE Execution Model:
  - Static SPMD:
    - identical UEs created together when a program starts (this is a standard approach familiar to message passing programmers)

UE: Unit of Execution … a software entity that advances a program counter (e.g. process of thread).

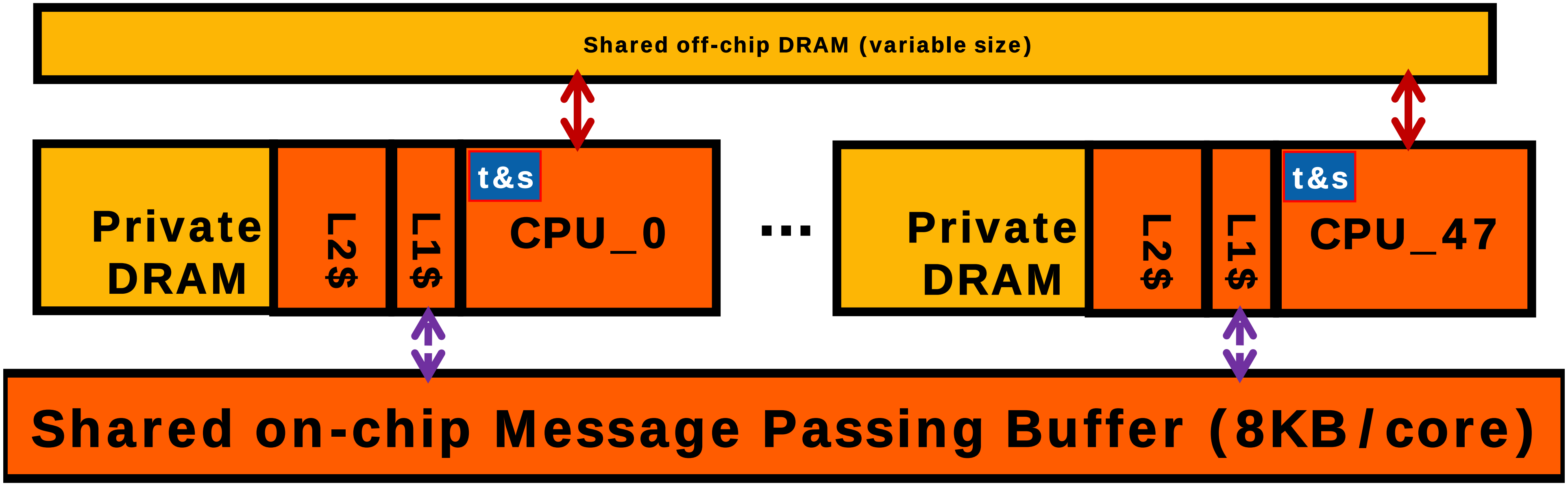
Msg. Pass. (RCCE) Configuration

- 48 x86 cores, the x86 memory model for Private DRAM

🟧 On-chip
🟨 Off-chip
Memory

⇠⇢ L1$, MPBT
⟷ Reg file, no$

- RCCE is a message passing library and thinks of the chip as a distributed memory platform … we have no use for shared DRAM and instead emphasize private DRAM.



Shared off-chip DRAM (variable size)

Private DRAM | L2$ | L1$ | t&s CPU_0 … Private DRAM | L2$ | L1$ | t&s CPU_47

Shared on-chip Message Passing Buffer (8KB/core)

t&s Shared test and set register

Msg. Pass. (RCCE) Configuration

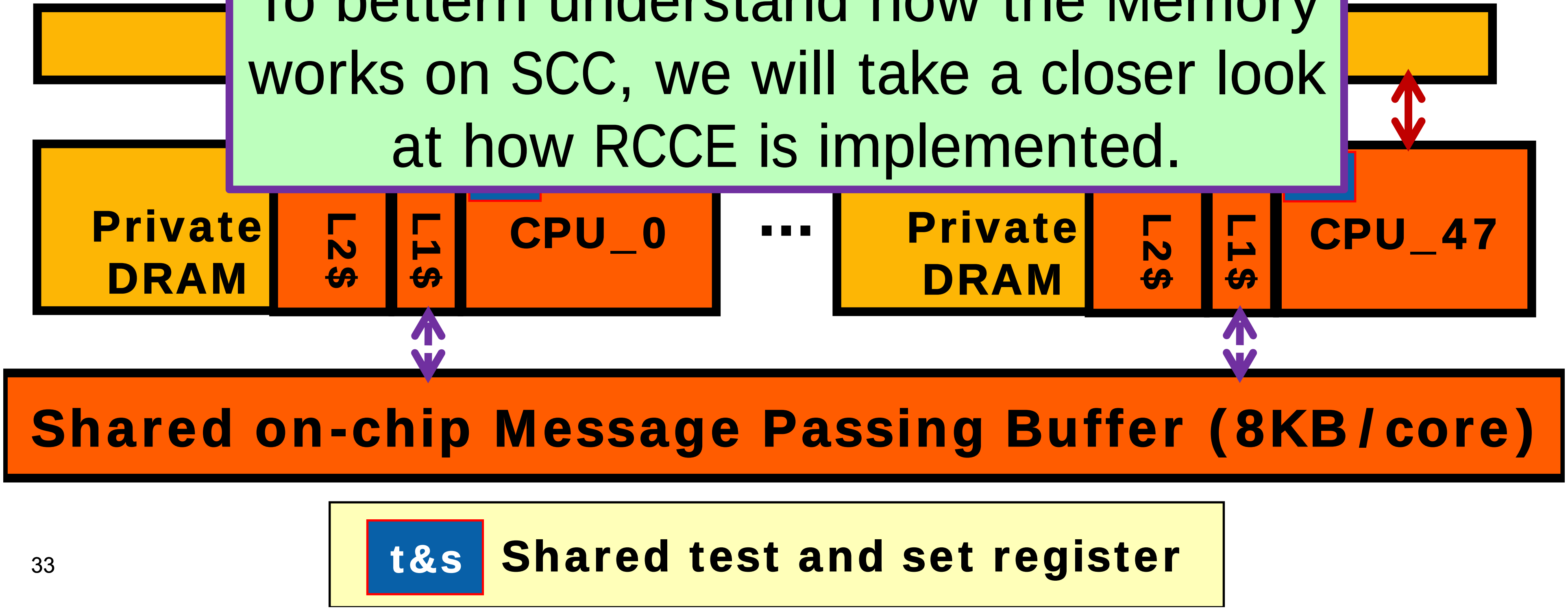
- 48 x86 cores, the x86 memory model for Private DRAM

🟧 On-chip Memory
🟨 Off-chip Memory

⇠⇢ L1$, MPBT
⟷ Reg file, no$

- RCCE is a message passing library and thinks of the chip as a distributed memory platform … we have no use for shared DRAM and instead emphasize private DRAM.

To bettern understand how the Memory works on SCC, we will take a closer look at how RCCE is implemented.



Private DRAM

L2$

L1$

CPU_0

…

Private DRAM

L2$

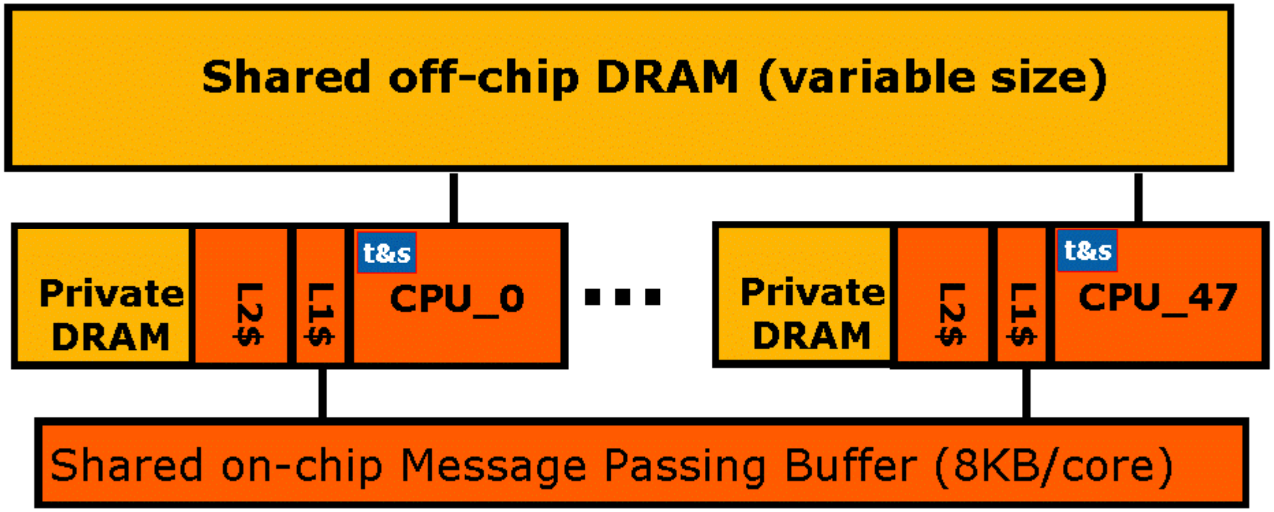
L1$

CPU_47

Shared on-chip Message Passing Buffer (8KB/core)

t&s Shared test and set register

How does RCCE work? Part 1


Shared off-chip DRAM (variable size)
Private DRAM
L2$
L1$
t&s
CPU_0
…
Private DRAM
L2$
L1$
t&s
CPU_47
Shared on-chip Message Passing Buffer (8KB/core)

Message passing buffer memory is special … of type MPBT

Cached in L1, L2 bypassed. Not coherent between cores

Data cached on read, not write. Single cycle op to invalidate all MPBT in L1 … Note this is not a flush

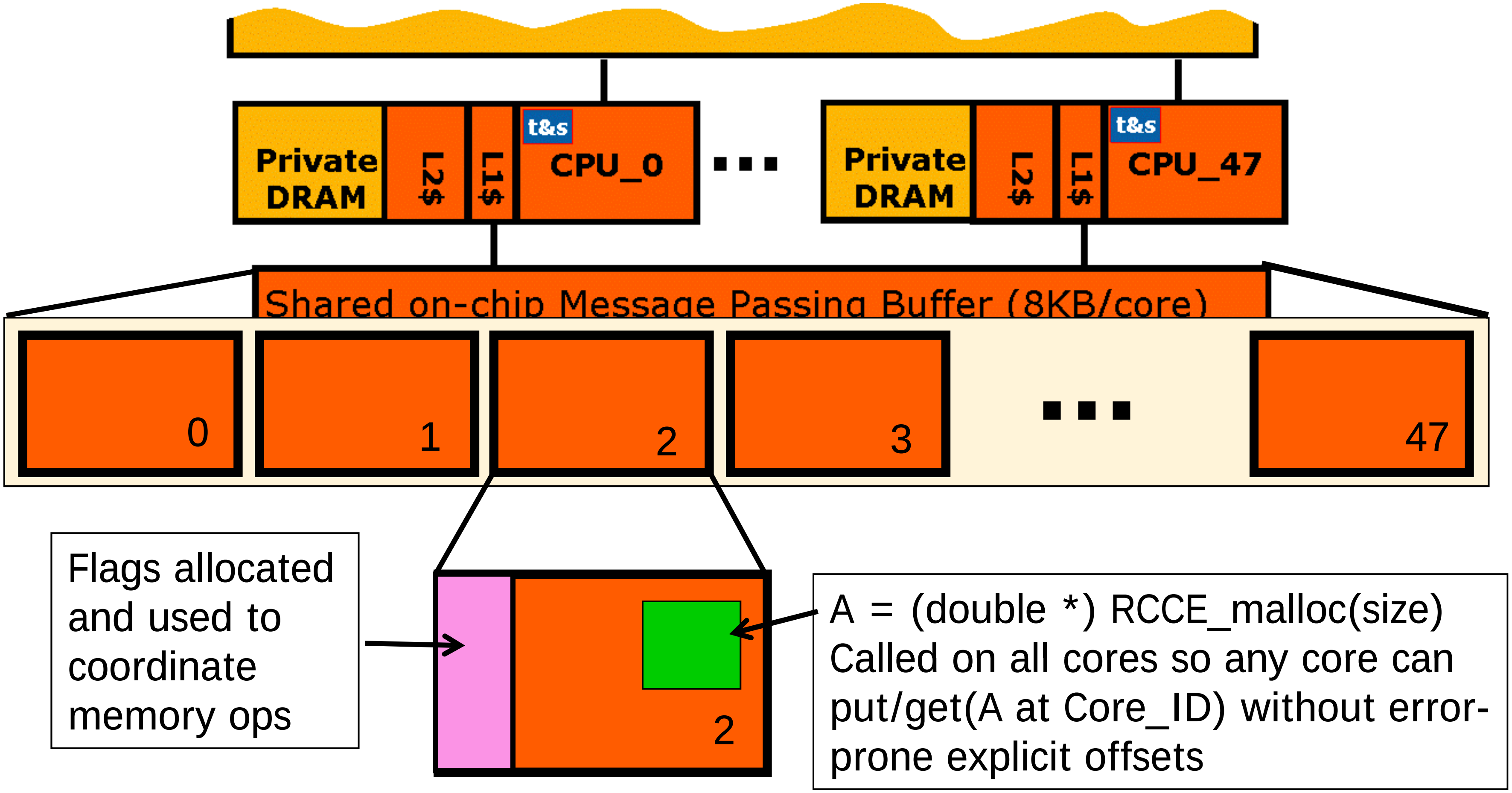
Consequences of MPBT properties:

- If data changed by another core and image still in L1, read returns stale data.
  - **Solution: Invalidate before read.**
- L1 has write-combining buffer; write incomplete line? expect trouble!
  - **Solution: don't. Always push whole cache lines**
- If image of line to be written already in L1, write will not go to memory.
  - **Solution: invalidate before write.**

Discourage user operations on data in MPB. Use only as a data movement area managed by RCCE … Invalidate early, invalidate often

How does RCCE work? Part 2

- Treat Msg Pass Buf (MPB) as 48 smaller buffers ... one per core.
- Symmetric name space ... Allocate memory as a collective op. Each core gets a variable with the given name at a fixed offset from the beginning of a core’s MPB.

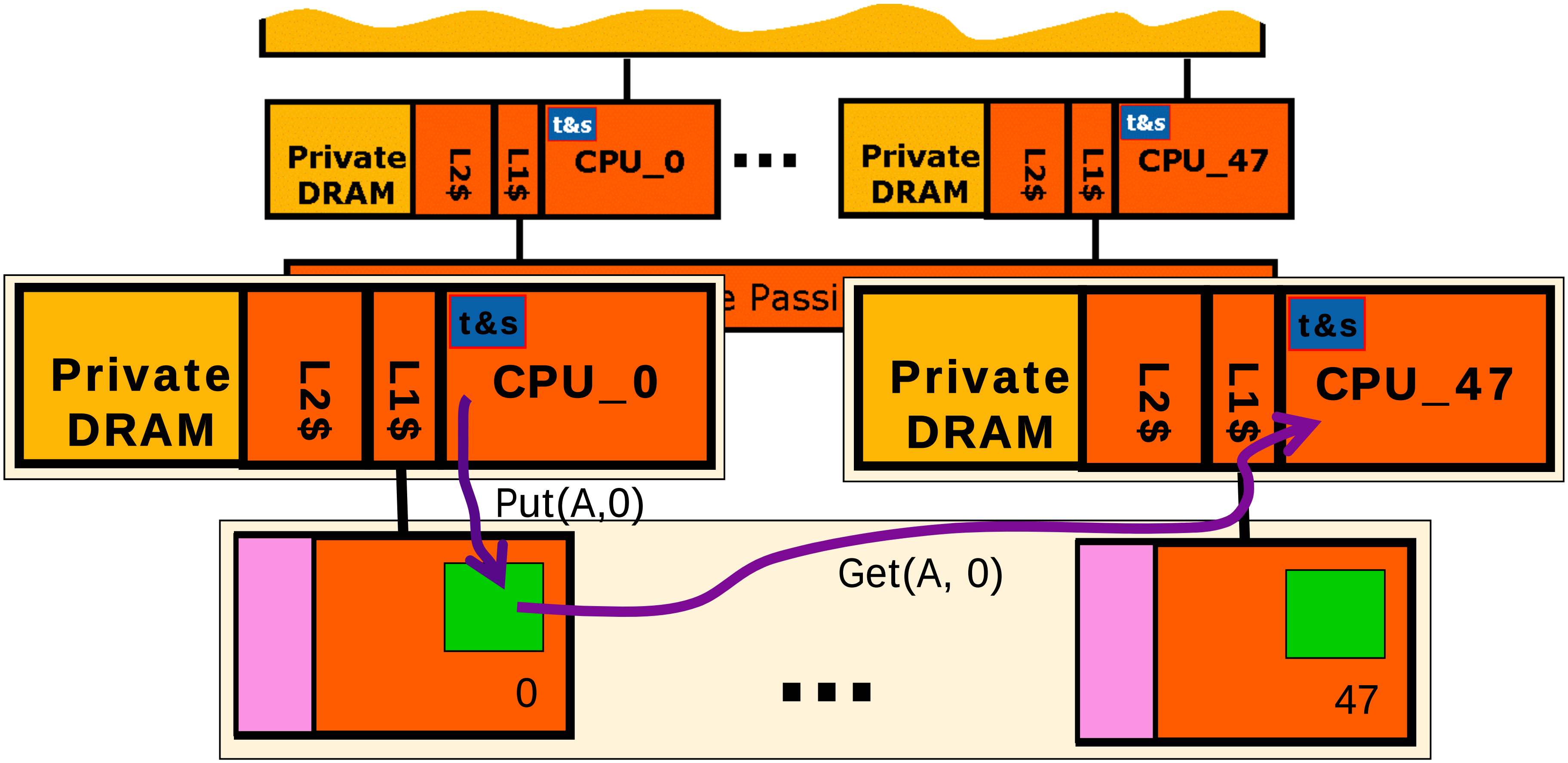


Flags allocated and used to coordinate memory ops

A = (double *) RCCE_malloc(size) Called on all cores so any core can put/get(A at Core_ID) without error-prone explicit offsets

How does RCCE work? Part 3

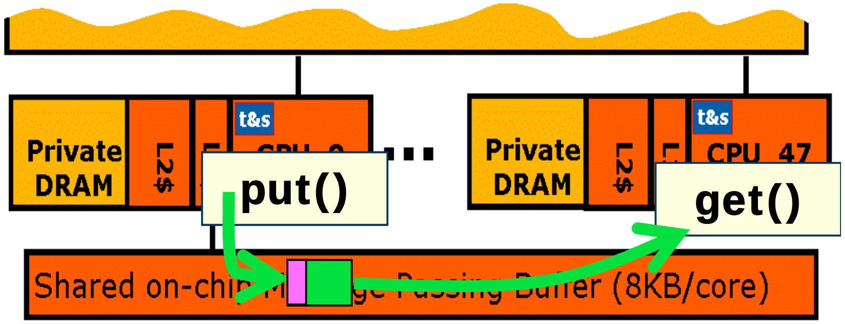
- The foundation of RCCE is a one-sided put/get interface.
- Symmetric name space … Allocate memory as a collective and put a variable with a given name into each core's MPB.



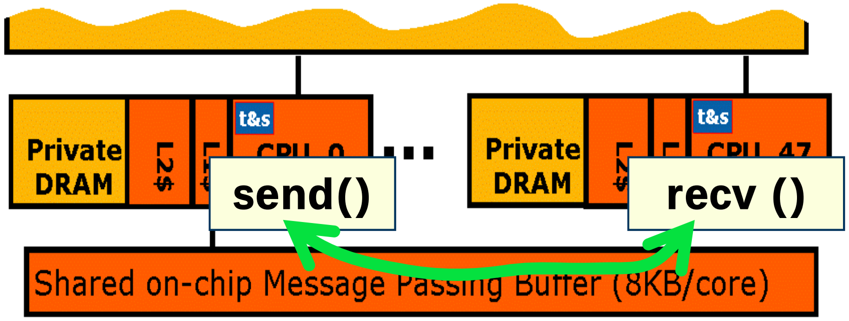
… and use flags to make the put's and get's "safe"

The RCCE library

- RCCE API provides the basic message passing functionality expected in a tiny communication library:
  - One + two sided interface (put/get + send/recv) with synchronization flags and MPB management exposed.
    - The “gory” interface for programmers who need the most detailed control over SCC

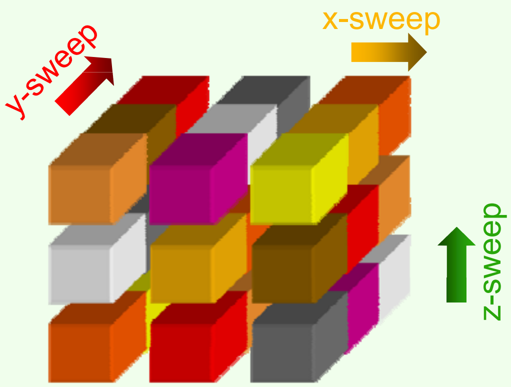


  - Two sided interface (send/recv) with most detail (flags and MPB management) hidden.
    - The “basic” interface for typical application programmers.



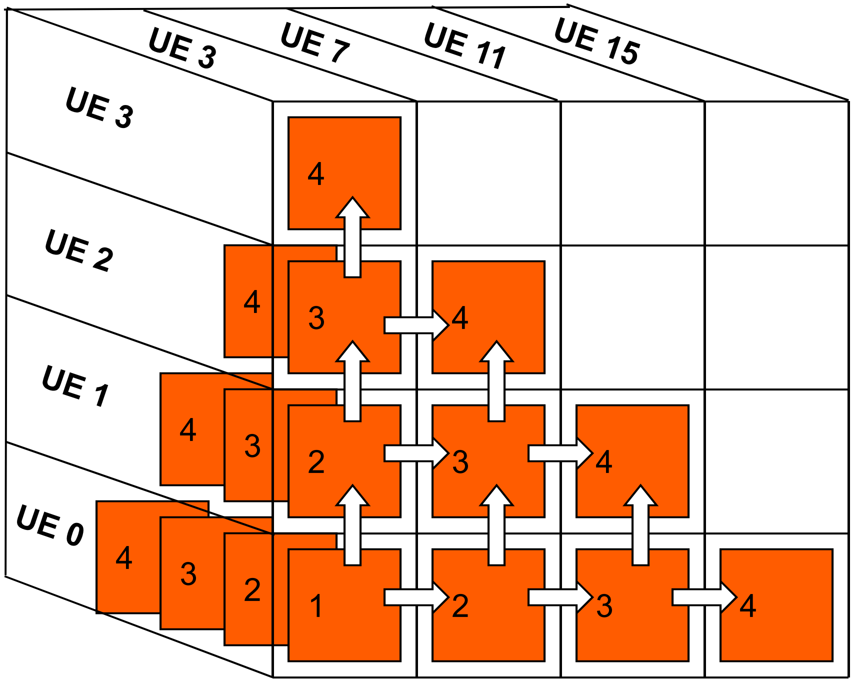
Linpack and NAS Parallel benchmarks

1. Linpack (HPL): solve dense system of linear equations
   - Synchronous comm. with “MPI wrappers” to simplify porting



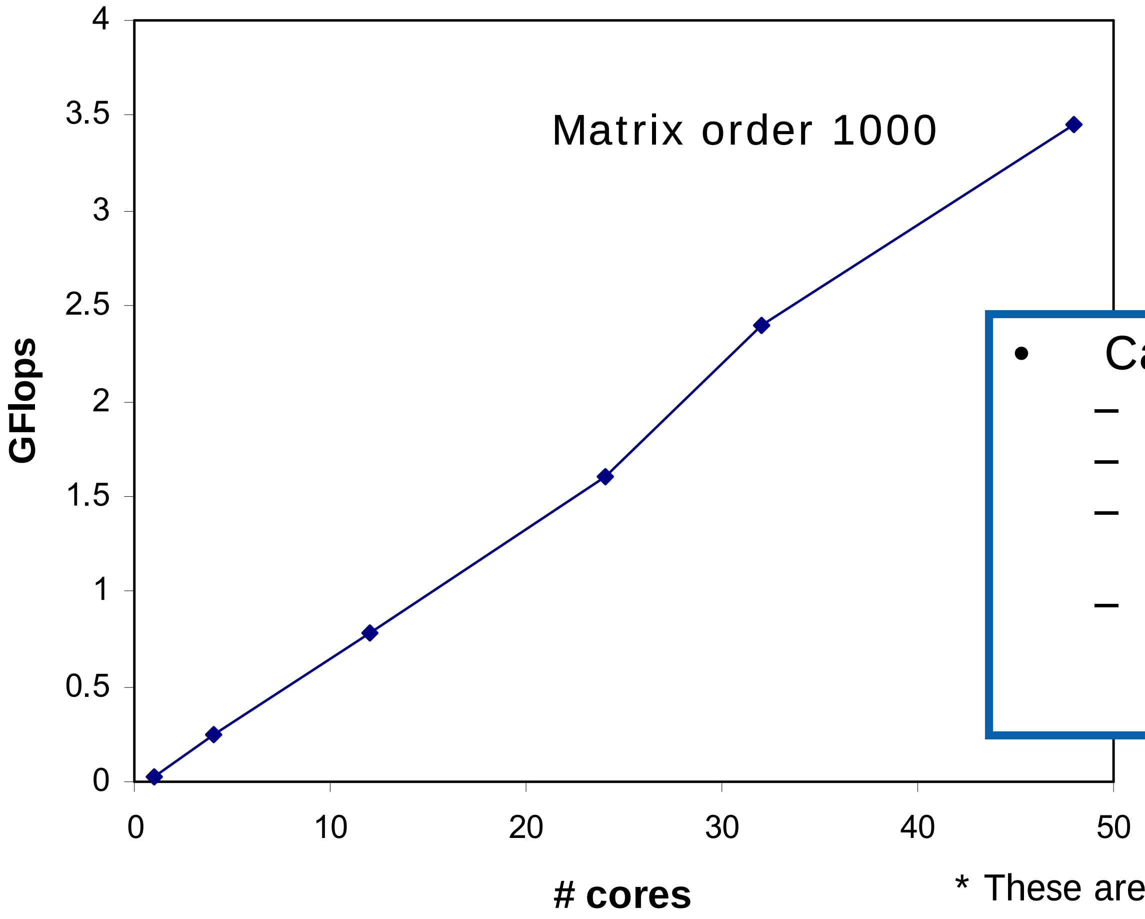
2. BT: Multipartition decomposition
   - Each core owns multiple blocks (3 in this case)
   - update all blocks in plane of 3x3 blocks
   - send data to neighbor blocks in next plane
   - update next plane of 3x3 blocks

3. LU: Pencil decomposition
   - Define 2D-pipeline process.
     - await data (bottom+left)
     - compute new tile
     - send data (top+right)



Linpack, on the Linux SCC platform

- Linpack (HPL)* strong scaling results:
  - GFLOPS vs. # of cores for a fixed size problem (1000).
  - This is a tough test … scaling is easier for large problems.



- Calculation Details:
  - Un-optimized C-BLAS
  - Un-optimized block size (4x4)
  - Used latency-optimized whole cache line flags
  - Performance dropped ~10% with memory optimized 1-bit flags

* These are not official LINPACK benchmark results.

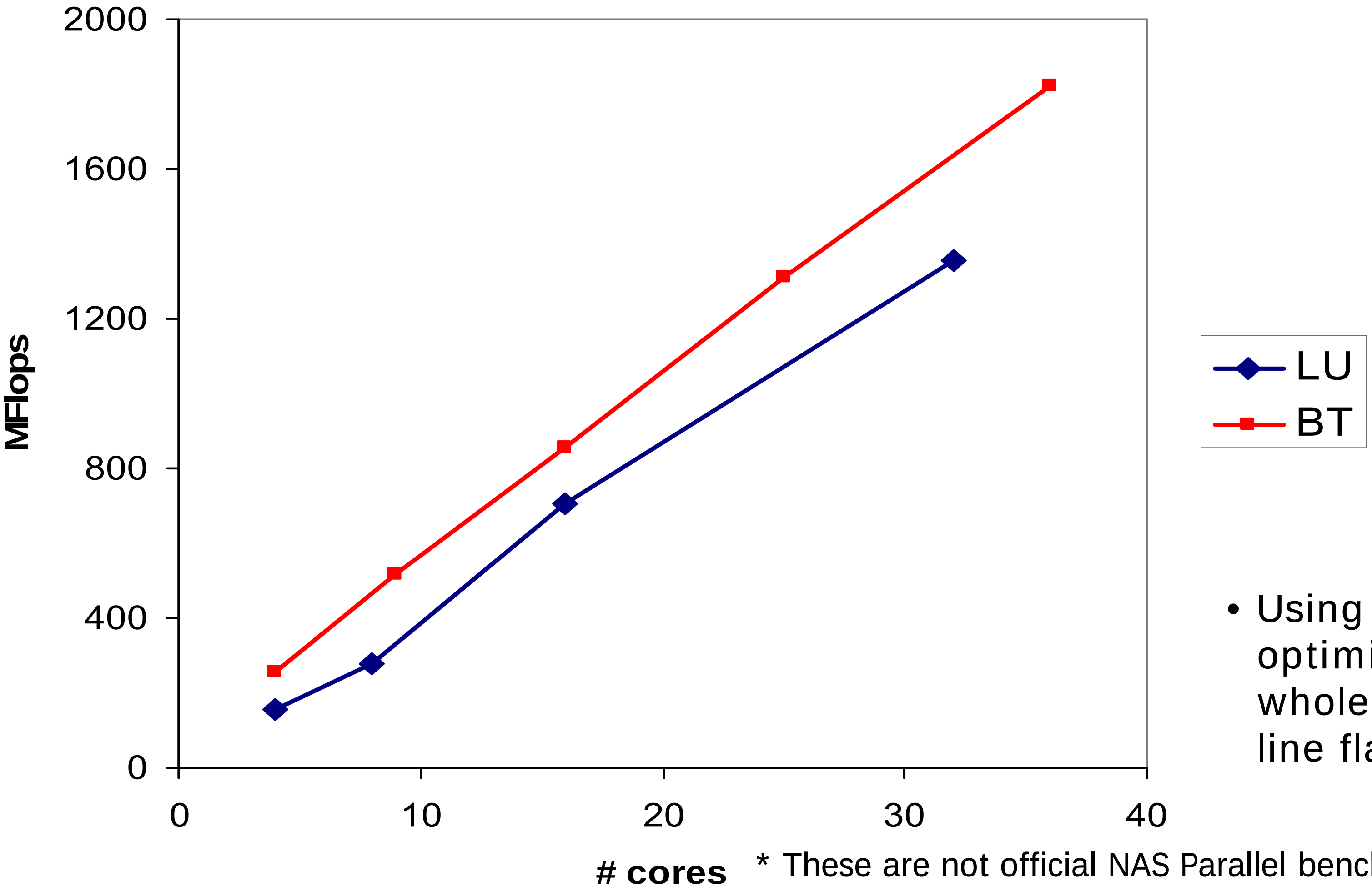
SCC processor 500MHz core, 1GHz routers, 25MHz system interface, and DDR3 memory at 800 MHz.

Third party names are the property of their owners.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, reference <http://www.intel.com/performance> or call (U.S.) 1-800-628-8686 or 1-916-356-3104.

LU/BT NAS Parallel Benchmarks, SCC

Problem size: Class A, 64 x 64 x 64 grid*



- Using latency optimized, whole cache line flags

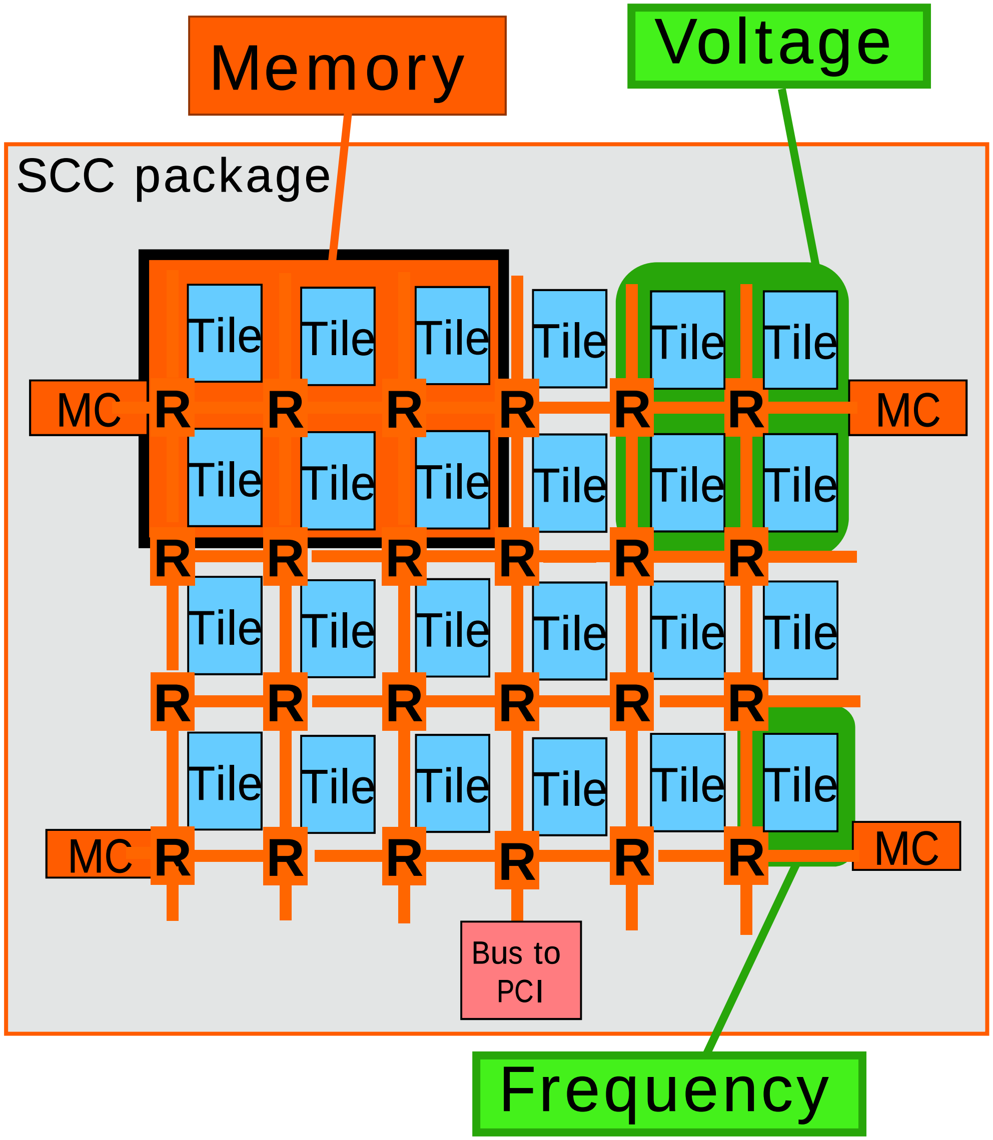
* These are not official NAS Parallel benchmark results.

SCC processor 500MHz core, 1GHz routers, 25MHz system interface, and DDR3 memory at 800 MHz.

Third party names are the property of their owners.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, reference <http://www.intel.com/performance> or call (U.S.) 1-800-628-8686 or 1-916-356-3104.

Power and memory-controller domains



Power ~ F V^2

- –Power Control domains (RPC):
  - –7 voltage domains … 6 4-tile blocks and one for on-die network.
  - –1 clock divider register per tile (i.e. 24 frequency domains)
  - –One RPC register so can process only one voltage request at a time; other requestors block

RCCE Power Management API

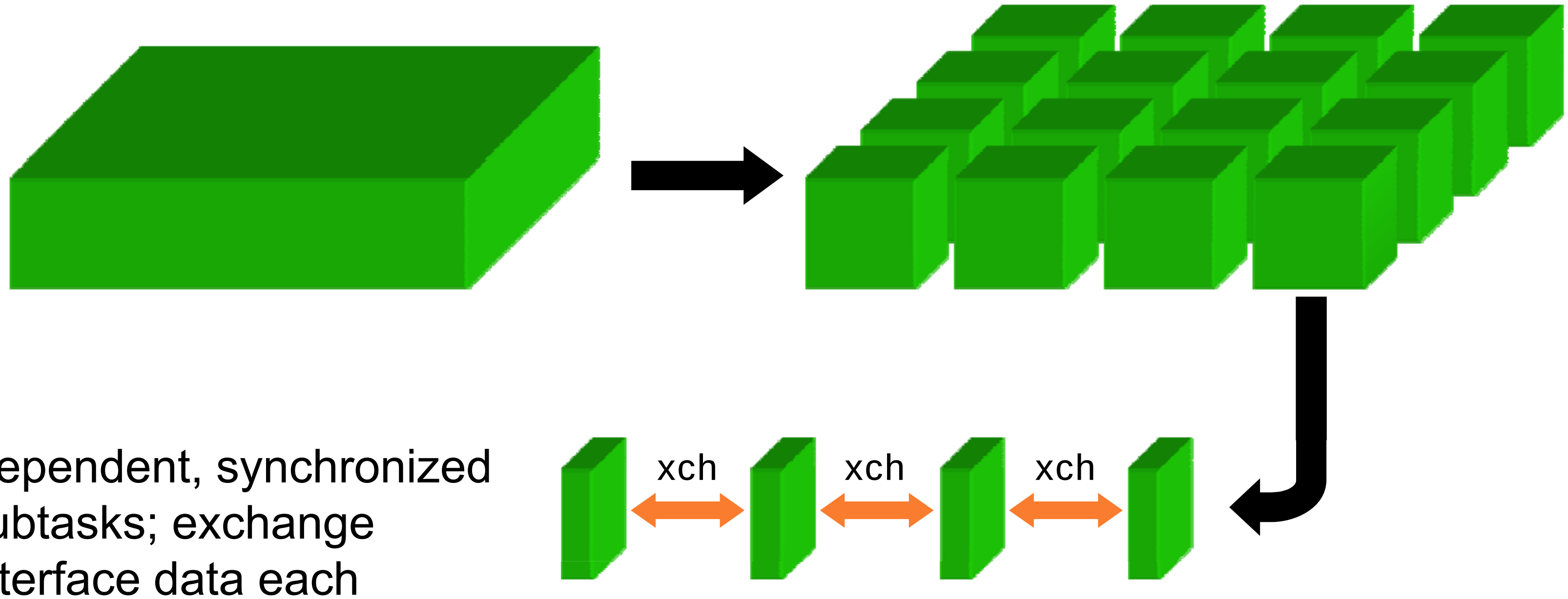
- RCCE power management emphasizes safe control: V/GHz changed together within each 4-tile (8-core) power domain.
  - A Master core sets V + GHz for all cores in domain.
    - RCCE_iset_power():
      - Input a frequency divisor (2 to 16) setting, and this will set the min voltage consistent with that frequency
    - RCCE_wait_power():
      - returns when power change is done
    - RCCE_set_frequency():
      - Set the frequency divisor (2 to 16)
- Power management latencies
  - V changes: Very high latency, *O*(Million) cycles.
  - GHz changes: Low latency, *O*(few) cycles.

Power management test

- A three-tier master-worker hierarchy,
  - one overall master, one team-lead per power domain, Team-members (cores) to do the work.
- Workload: A stencil computation to solve a PDE.

Overall data space

Independent tasks
(all different sizes)



Dependent, synchronized subtasks; exchange interface data each iteration

xch

xch

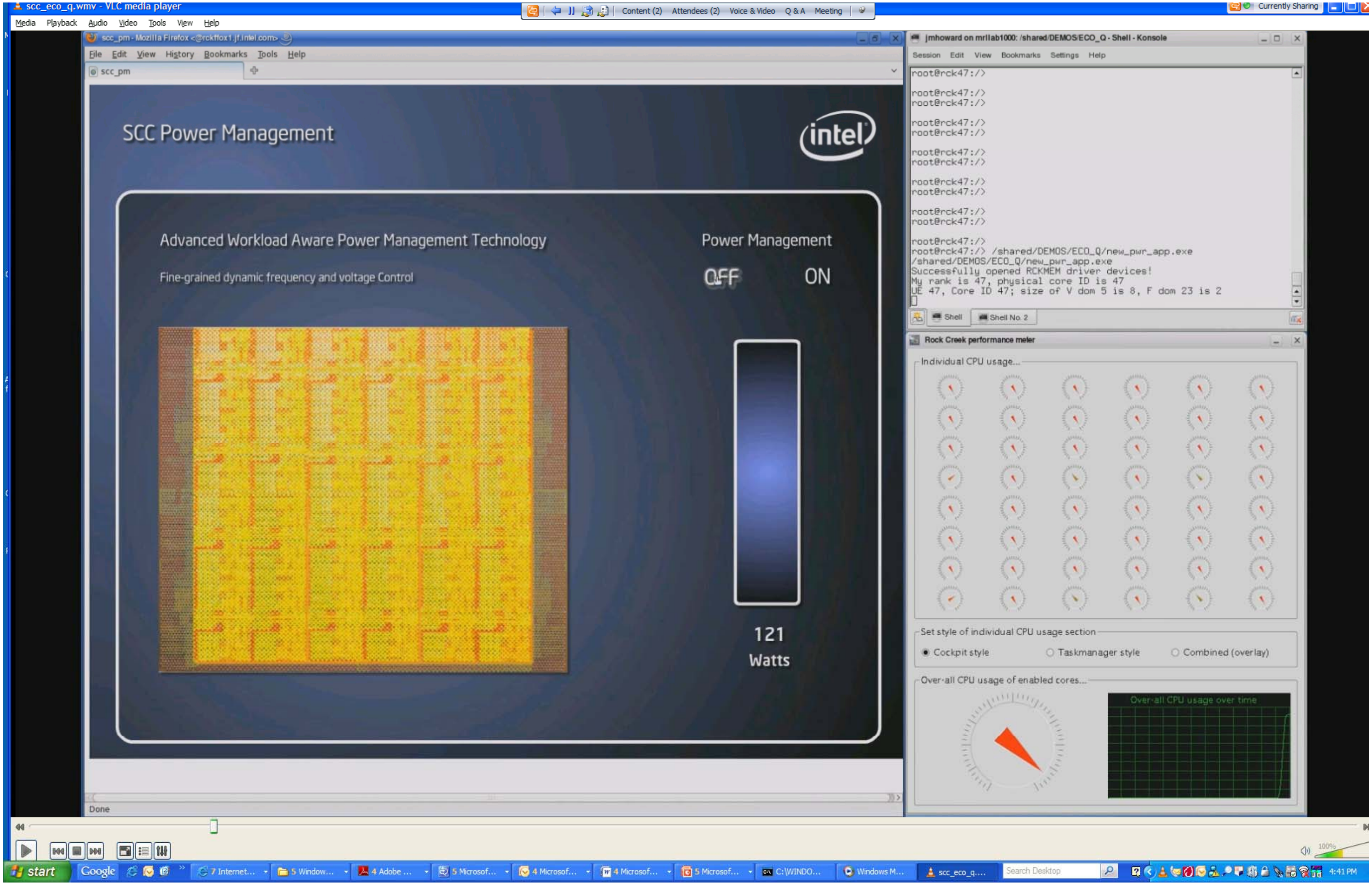
xch

Team member

Team member

Team member

Team lead


scc_eco_q.wmv - VLC media player
Media Playback Audio Video Tools View Help
Content (2) Attendees (2) Voice & Video Q & A Meeting
Currently Sharing
scc_pm - Mozilla Firefox <@rckffox1.jf.intel.com>
File Edit View History Bookmarks Tools Help
scc_pm
Done
jmhoward on mrllab1000: /shared/DEMOS/ECO_Q - Shell - Konsole
Session Edit View Bookmarks Settings Help
Shell
Shell No. 2
Rock Creek performance meter
Over-all CPU usage over time
start
Google
7 Internet...
5 Window...
4 Adobe ...
5 Microsof...
4 Microsof...
4 Microsof...
5 Microsof...
C:\WINDO...
Windows M...
scc_eco_q....
Search Desktop
4:41 PM

SCC Power Management

Advanced Workload Aware Power Management Technology

Fine-grained dynamic frequency and voltage Control

Power Management

OFF ON

121 Watts

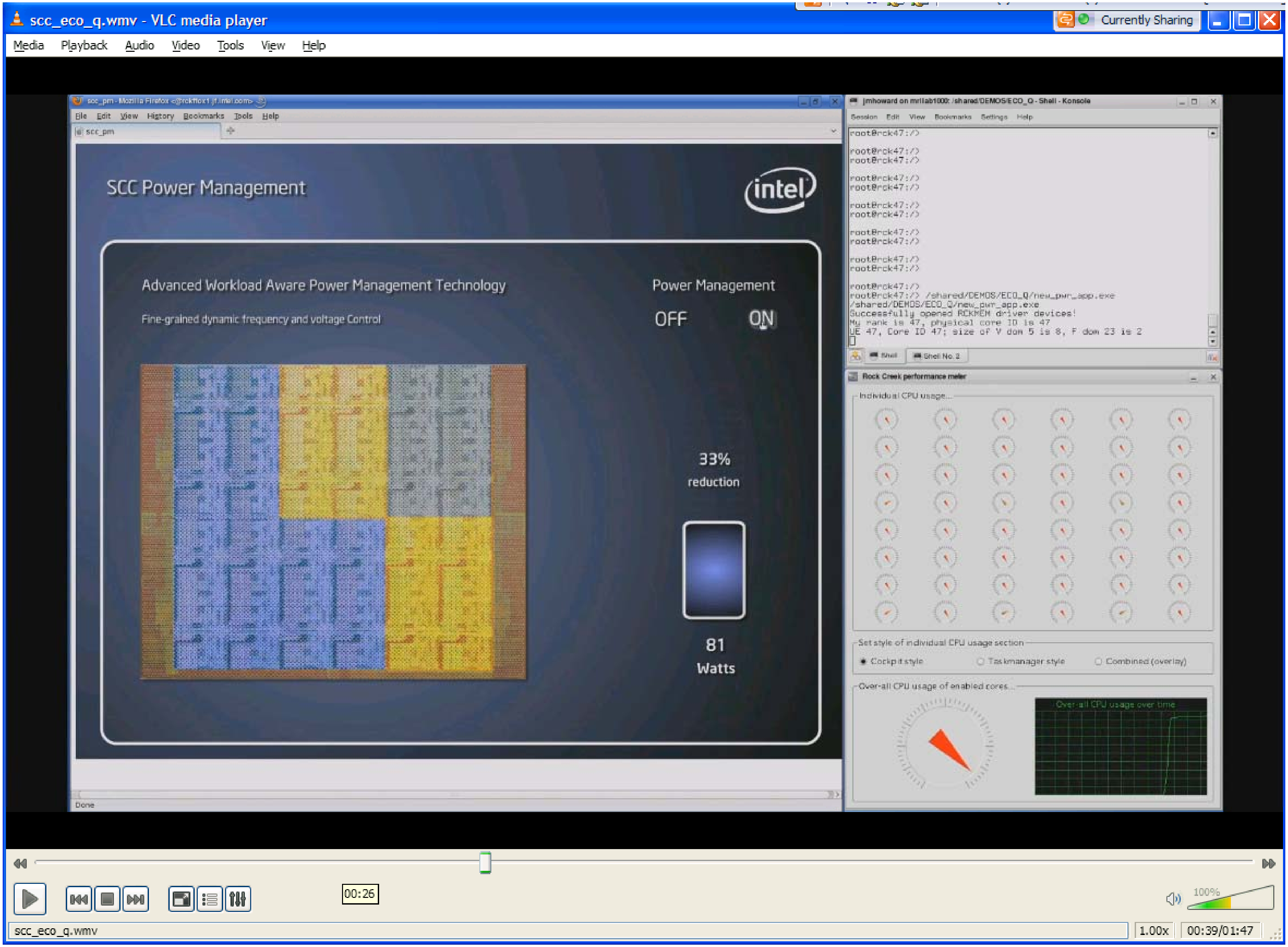
```
root@rck47:/>

root@rck47:/>
root@rck47:/>

root@rck47:/>
root@rck47:/>

root@rck47:/>
root@rck47:/>

root@rck47:/>
root@rck47:/>

root@rck47:/>
root@rck47:/>

root@rck47:/>
root@rck47:/> /shared/DEMOS/ECO_Q/new_pwr_app.exe
/shared/DEMOS/ECO_Q/new_pwr_app.exe
Successfully opened RCKMEM driver devices!
My rank is 47, physical core ID is 47
UE 47, Core ID 47; size of V dom 5 is 8, F dom 23 is 2
```

Individual CPU usage...

Set style of individual CPU usage section

● Cockpit style ○ Taskmanager style ○ Combined (overlay)

Over-all CPU usage of enabled cores...


scc_eco_q.wmv - VLC media player
Currently Sharing
Media
Playback
Audio
Video
Tools
View
Help
scc_pm - Mozilla Firefox <@rckffox1.jf.intel.com>
File
Edit
View
History
Bookmarks
Tools
Help
scc_pm
intel
Done
jmhoward on mrilab1000: /shared/DEMOS/ECO_Q - Shell - Konsole
Session
Edit
View
Bookmarks
Settings
Help
Shell
Shell No. 2
Rock Creek performance meter
Individual CPU usage...
Over-all CPU usage of enabled cores...
Over-all CPU usage over time
00:26
100%
1.00x
00:39/01:47
scc_eco_q.wmv

SCC Power Management

Advanced Workload Aware Power Management Technology

Fine-grained dynamic frequency and voltage Control

Power Management

OFF ON

33%
reduction

81
Watts

```
root@rck47:/>

root@rck47:/>
root@rck47:/>

root@rck47:/>
root@rck47:/>

root@rck47:/>
root@rck47:/>

root@rck47:/>
root@rck47:/>

root@rck47:/>
root@rck47:/>

root@rck47:/>
root@rck47:/> /shared/DEMOS/ECO_Q/new_pwr_app.exe
/shared/DEMOS/ECO_Q/new_pwr_app.exe
Successfully opened RCKMEM driver devices!
My rank is 47, physical core ID is 47
UE 47, Core ID 47; size of V dom 5 is 8, F dom 23 is 2
```

Set style of individual CPU usage section

● Cockpit style ○ Taskmanager style ○ Combined (overlay)

Conclusions

- RCCE software works
  - RCCE's restrictions (Symmetric MPB memory model and blocking communications) have not been a fundamental obstacle
  - Functional emulator is a useful development/debug device
- SCC architecture
  - The on-chip MPB was effective for scalable message passing applications
  - Software controlled power management works … but it's challenging to use because (1) granularity of 8 cores and (2) high latencies for voltage changes
  - The Test&set registers (only one per core) will be a bottleneck.
    - Sure wish we had asked for more!
- Future work
  - Add shmalloc() to expose shared off-chip DRAMM (in progress).
  - Move resource management into OS/drivers so multiple apps can work together safely.
  - We have only just begun to explore power management capabilities … we need to explore additional usage models.