A Systems Perspective on End of Silicon

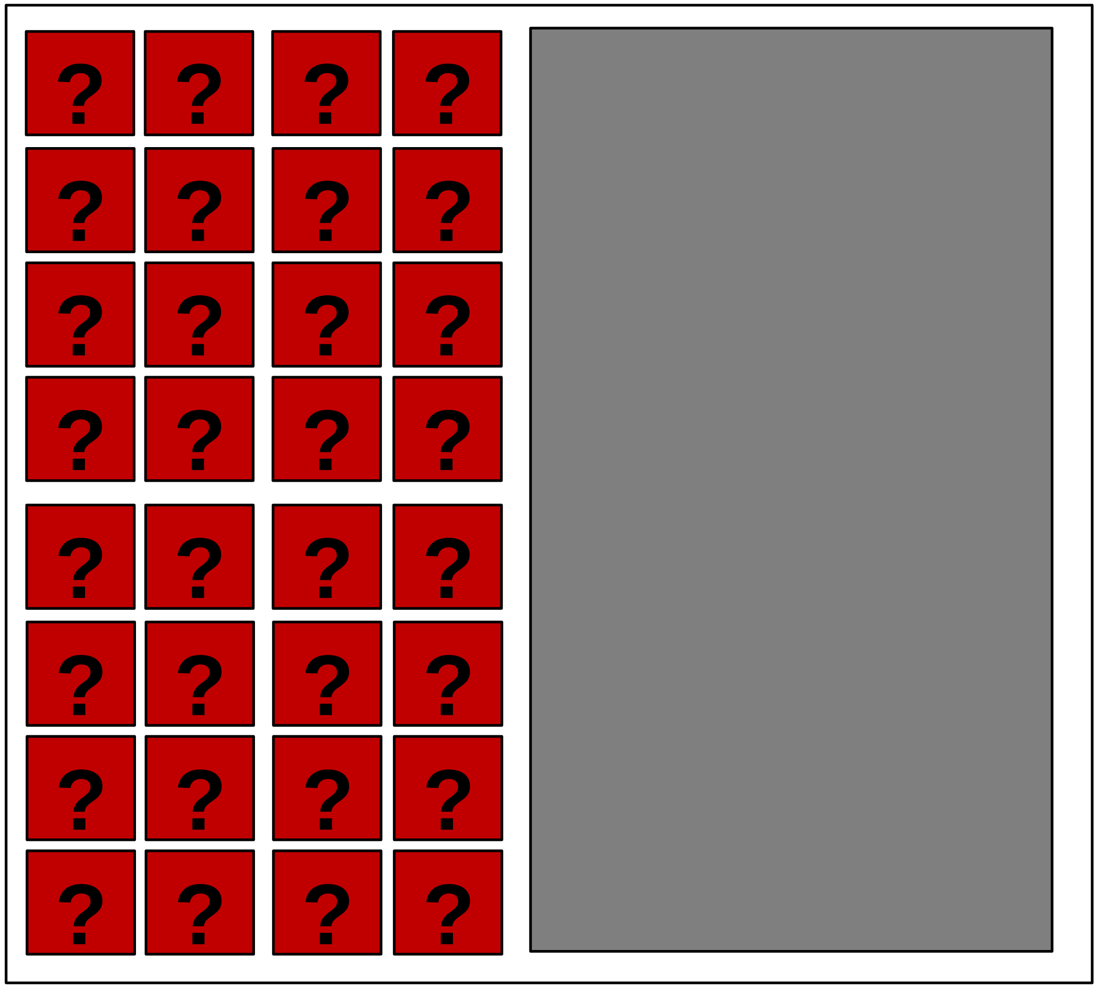
Frontiers of Extreme Computing
October 24, 2007

Karu Sankaralingam
University of Wisconsin-Madison
The University of Texas at Austin

Acknowledgment

- PIs:
  - Steve Keckler, Doug Burger, Kathryn McKinley – UT-Austin
- TRIPS Hardware Team
  - Raj Desikan, Saurabh Drolia, Madhu Sibi Govindan, Divya Gulati, Paul Gratz, Heather Hanson, Changkyu Kim, Haiming Liu, Robert McDonald, Ramdas Nagarajan, Nitya Ranganathan, Simha Sethumadhavan, Premkishore Shivakumar
- TRIPS Software Team
  - Kathryn McKinley, Jim Burrill, Xia Chen, Sundeep Kushwaha, Bert Maher, Nick Nethercote, Suriya Narayanan, Sadia Sharif, Aaron Smith, Bill Yoder
- IBM Microelectronics Austin ASIC Group
- TRIPS Sponsors
  - DARPA Polymorphous Computing Architectures
  - Air Force Research Laboratories
  - National Science Foundation
  - IBM, Intel, Sun Microsystems

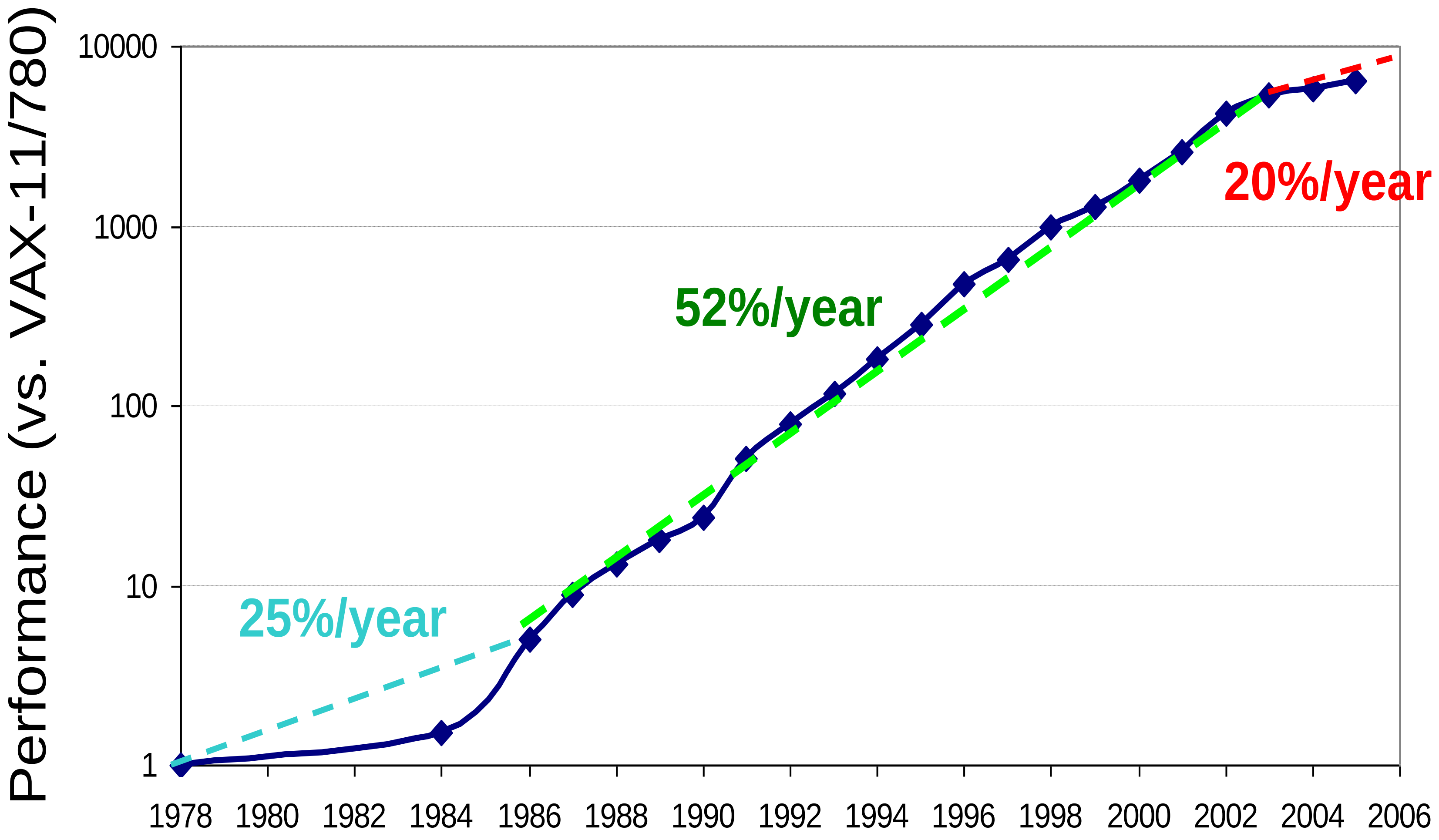
Processor Architecture



Where are HPC Systems Going?

- Scaling of uniprocessor performance has been historical driver
  - 50-55% per year for a significant period
  - Systems with a constant number of processors benefit
- Transistor scaling may continue to the end of the roadmap
  - However, system scaling must change considerably
  - The "last classical computer" will look very different from today's systems
- Outline of driving factors and views
  - Exploitation of concurrency - are more threads the only answer?
    - We are driving to a domain where tens to hundreds of thousands of processors are the sole answer for HPC systems
  - How will power affect system and architecture design?
  - How to provide the programmability, flexibility, efficiency, and performance future systems need?

Shift in Uniprocessor Performance



Slide by Dave Patterson

Historical Sources of Performance

- Four factors
  - Device speed (17%/year)
  - Pipelining (reduced FO4) - ~18%/year from 1990-2004
  - Improved CPI
  - Number of processors/chip - n/a
- Device speed will continue for some time
- Deeper pipelining is effectively finished
  - Due to both power and diminishing returns
  - Ends the era of 40%/year clock improvements
- CPI is actually increasing
  - Effect of deeper pipelines, slower memories
  - On-chip delays
  - Simpler cores due to power
- Number of processors/chip starting to grow
  - “Passing the buck” to the programmer
  - Have heard multiple takes on this from HPC researchers

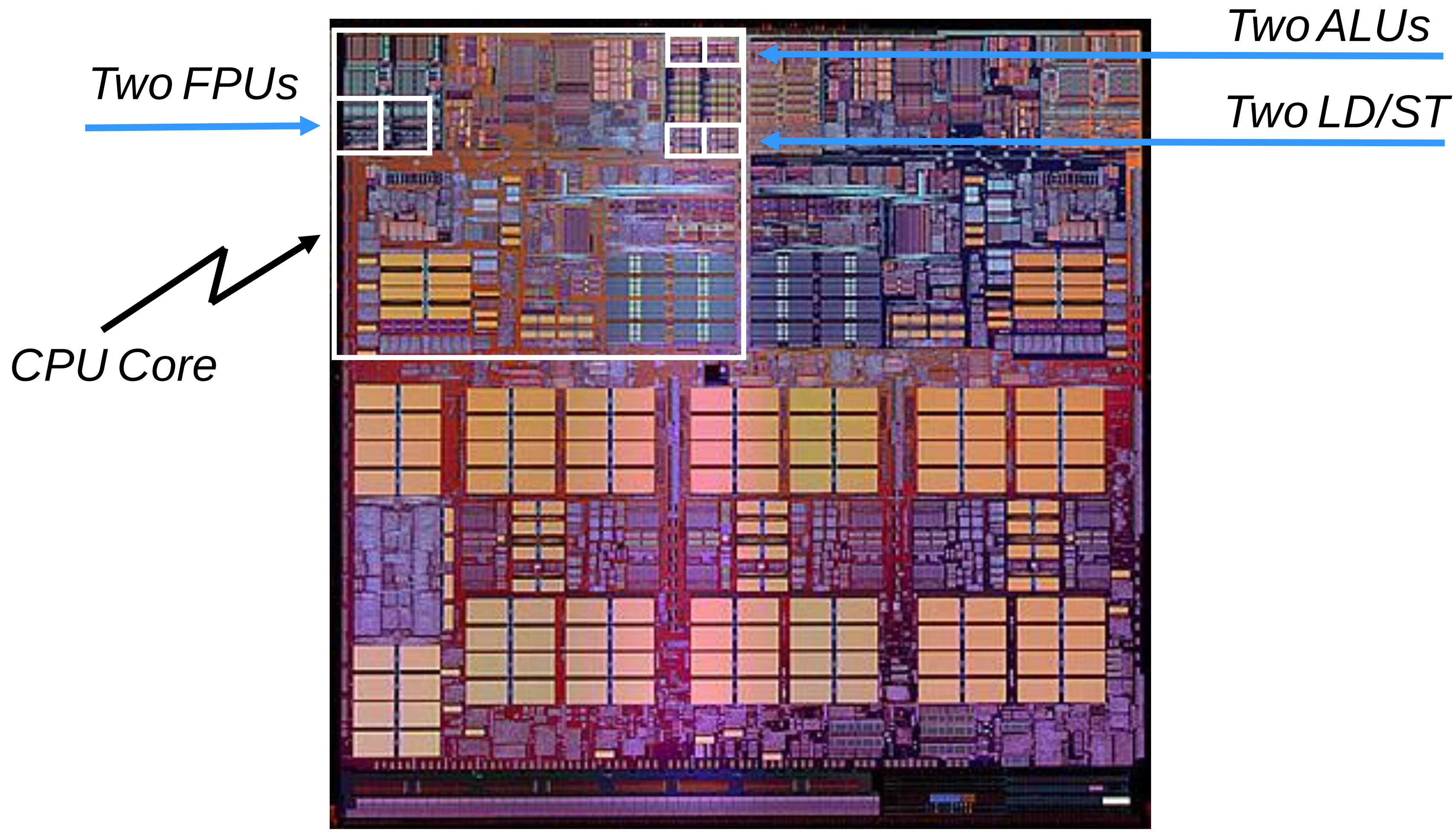
Opportunity to End of Si Roadmap

- How much performance growth between now and 2020 per unit area of silicon?
  - 17% device scaling gives 10x performance boost
  - 50x increase in device count provides what level of performance?
  - Linear growth in performance: 500x performance boost
- What have we gotten historically?
  - 500x performance boost over that same period
  - However, a large fraction of that is increased frequency
  - Without that, historical boost would be 50X
  - The extra 10x needs to come from concurrency
- Opportunity
  - **Many simpler processors per unit area provide more FLOP/transistor efficiency**
  - **May be efficiency issues (communication, load balancing)**
  - **May be programmability issues**
- $64K question: how can we get that efficiency while circumventing the above problems?

Granularity versus Number of Processors

- Historically, designers opted for improved CPI over number of processors
- Shifting due to lack of CPI improvements (finite core issue widths)
  - What will be granularity of CMPs?
  - What will be power dissipation curves?
- Small number of heavyweight cores versus many lightweight cores?
- Interested in HPC researchers' thoughts on granularity issue
  - Key question: is the ideal architecture as many lightweight cores as possible, with frequency/device speed scaled down to make power dissipation tractable?
- Amdahl's law
  - Need powerful uniprocessor for single-thread performance

Superscalar core



Only 12% of Non-Cache, Non-TLB Core Area is Execution Units

Out-of-Order Overheads

- A day in the life of a RISC/CISC instruction
  - ISA does not support out-of-order execution
  - Fetch a small number of instructions
  - Scan them for branches, predict
  - Rename all of them, looking for dependences
  - Load them into an associative issue window

- Interface is out-dated
- Microarchitecture overly burdened

- BUT: performance from in-order architectures hurt badly by cache misses
  - Unless working set fits precisely in the cache
  - Take a bit hit in CPI, need that many more processors!
- Programmable, good performance, but now poor efficiency
  - Can take C, magically gets 2X better every 2 years

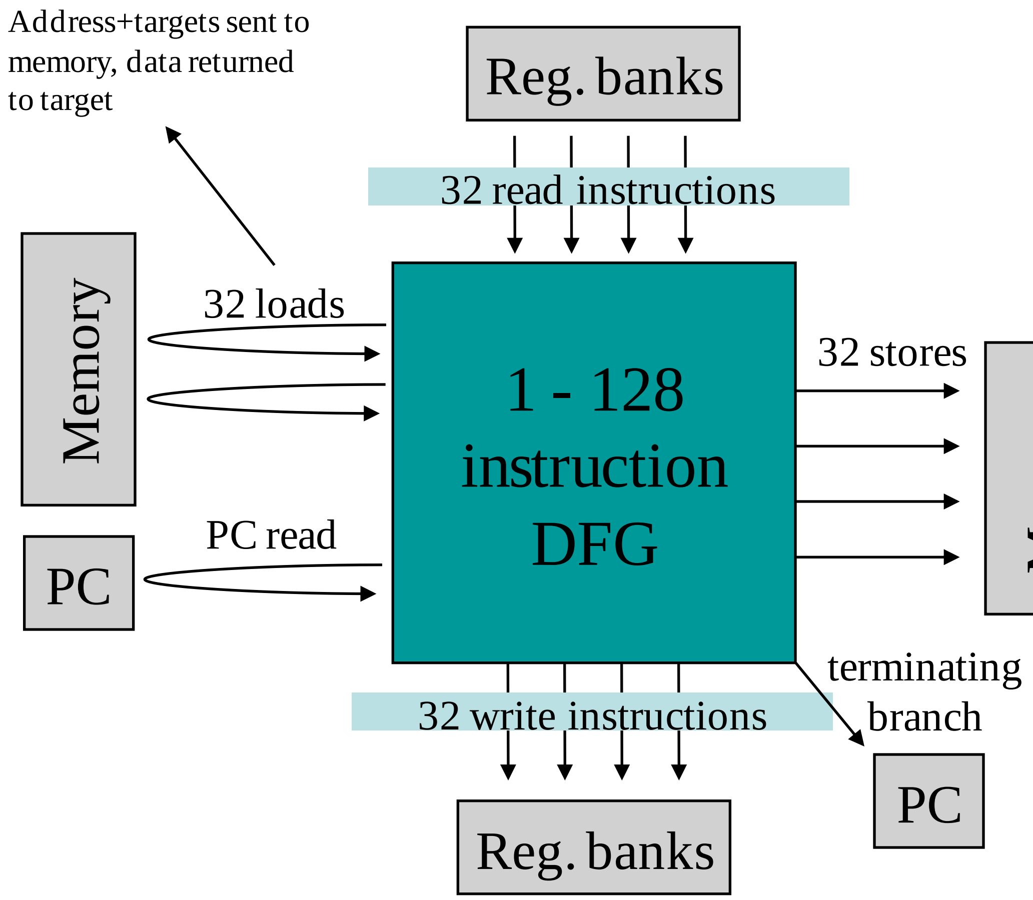
TRIPS Approach

- Renegotiate Compiler, ISA, Microarchitecture responsibilities
- This talk
  - EDGE ISA
  - TRIPS Microarchitecture
  - Prototype design

TRIPS Approach to Execution Efficiency

- EDGE (Explicit Data Graph Execution) architectures have two key features
  - Block-atomic execution
  - Direct instruction communication
- Form large blocks of instructions with no internal control flow transfer
  - We use hyperblocks with predication
  - Control flow transfers (branches) only happen on block boundaries
- Form dataflow graphs of instructions, map directly to 2-D substrate
  - Instructions communicate directly from ALU to ALU
  - Registers only read/written at begin/end of blocks
  - Static placement optimizations
    - Co-locate communicating instructions on same or nearby ALU
    - Place loads close to cache banks, etc.

Architectural Structure of a TRIPS Block



Block characteristics:

- Fixed size:
  - 128 instructions max
  - L1 and core expands empty 32-inst chunks to NOPs
- Load/store IDs:
  - Maximum of 32 loads+stores may be emitted, but blocks can hold more than 32
- Registers:
  - 8 *read insts.* max to reg. bank (4 banks = max of 32)
  - 8 *write insts.* max to reg bank (4 banks = max of 32)
- Control flow:
  - Exactly one branch emitted
  - Blocks may hold more

TRIPS ISA: Dataflow in the ISA

RISC ISA

```
$g1 ← 0
$g2 ← 1
loopbody:
add $g1  ← $g1, $g2
cmp $g0 ← $g2, 10
bz looptail
inc $g2 ← $g2
br loopbody
looptail:
…
```

TRIPS

```
.blockbegin init
      block's instructions
.blockend


.blockbegin loopbody
      block's instructions
.blockend


.blockbegin looptail
      block's instructions
.blockend
```

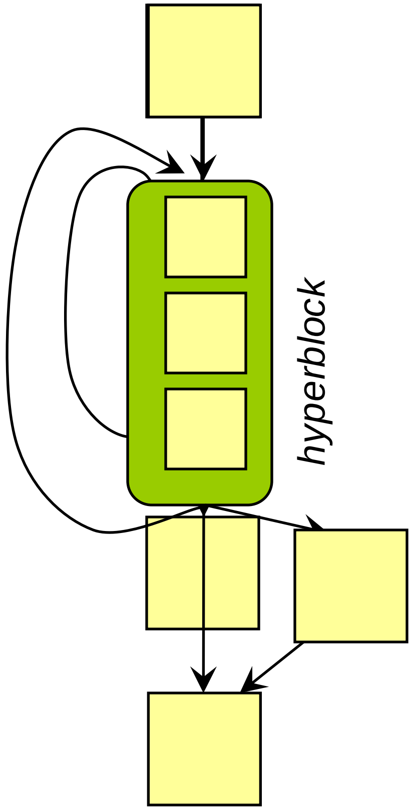
```
.blockbegin loopbody
N[0] read $g1 →N[2],N[3],N[6]
N[1] read $g2 →N[2]

N[2] inc        → N[7], N[4]
N[3] add        → N[8]
N[4] teqi 10    → N[5], N[6]
N[5] bro_f loopbody
N[6] bro_t looptail

N[7] write $g1
N[8] write $g2
.blockend
```

TRIPS Execution (1)

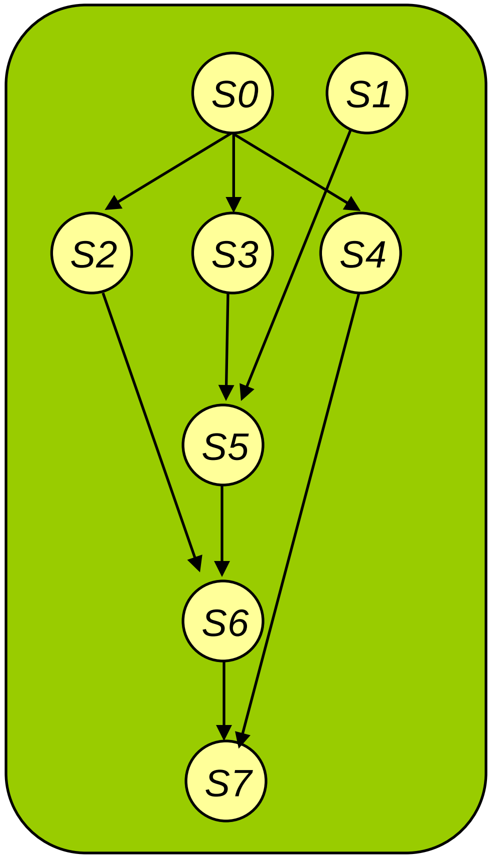
C Code

```
int main(void) {
  int z, i;
  z = 0;
  for (i = 1;
        i <= 10; i++) {
    z += i;
  }
  printf(“%d\n", z);
}
```

Control flow Graph

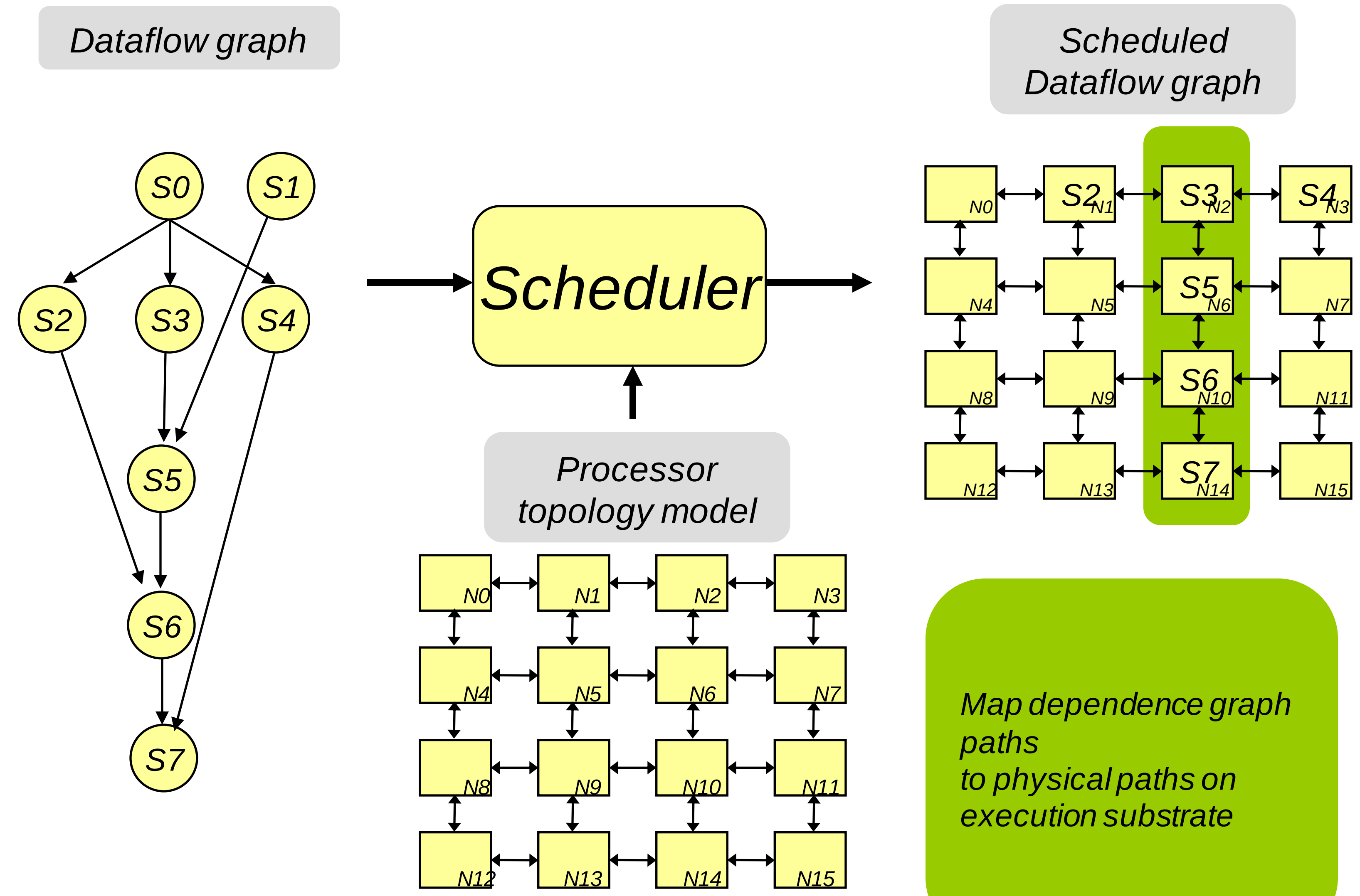


Dataflow Graph


$$\begin{tikzcd}
 & S0 \arrow[dl] \arrow[d] \arrow[dr] & S1 \arrow[dl] \\
S2 \arrow[dd] & S3 \arrow[d] & S4 \arrow[ddl] \\
 & S5 \arrow[d] & \\
 & S6 \arrow[d] & \\
 & S7 &
\end{tikzcd}$$

Control flow heuristics – loop unrolling, inlining, if-conversion…

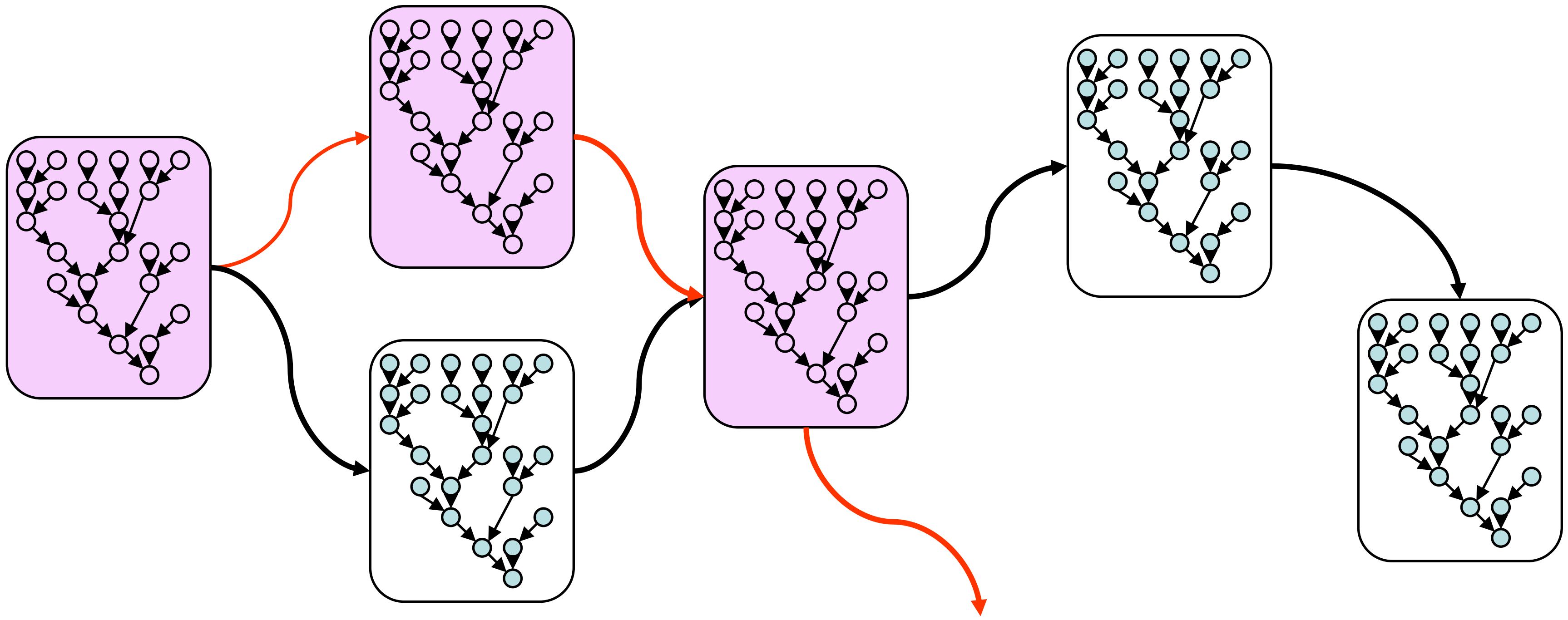
TRIPS Execution (2)



Map dependence graph paths to physical paths on execution substrate

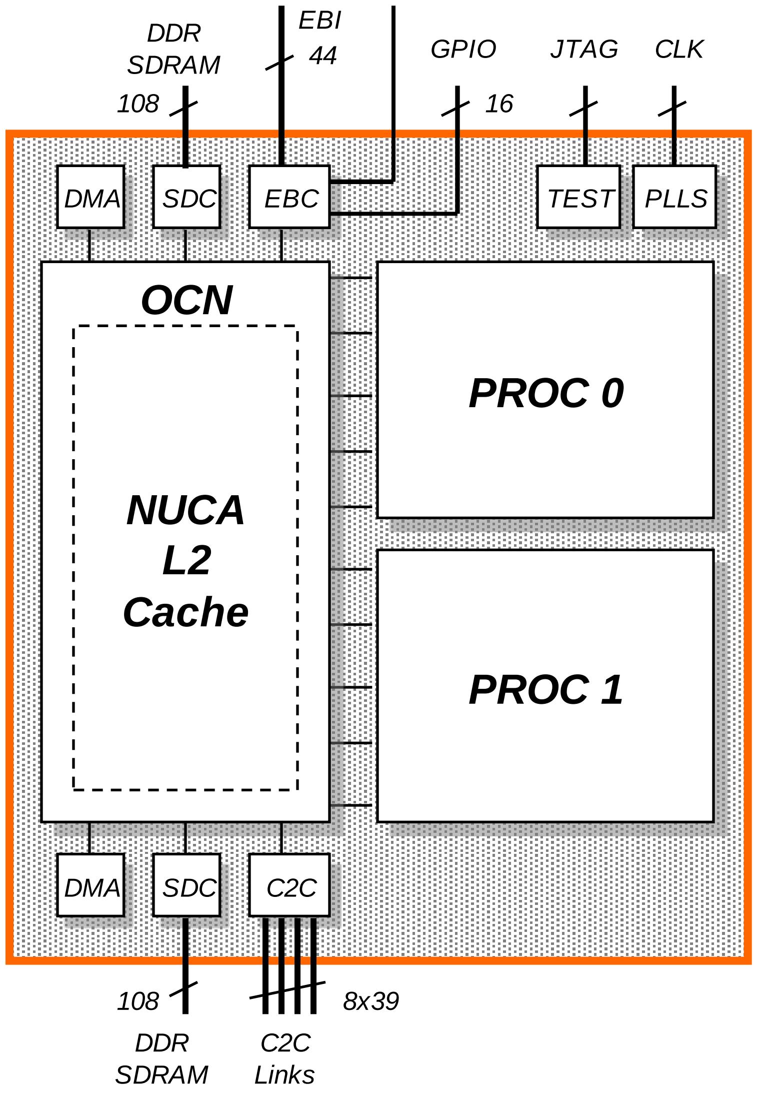
TRIPS Block Flow

- Compiler partitions program into “mini-graphs”
- Within each graph, instructions directly target others
- These mini-graphs execute like highly complex instructions
- Reduce per-instruction overheads, amortized over a block

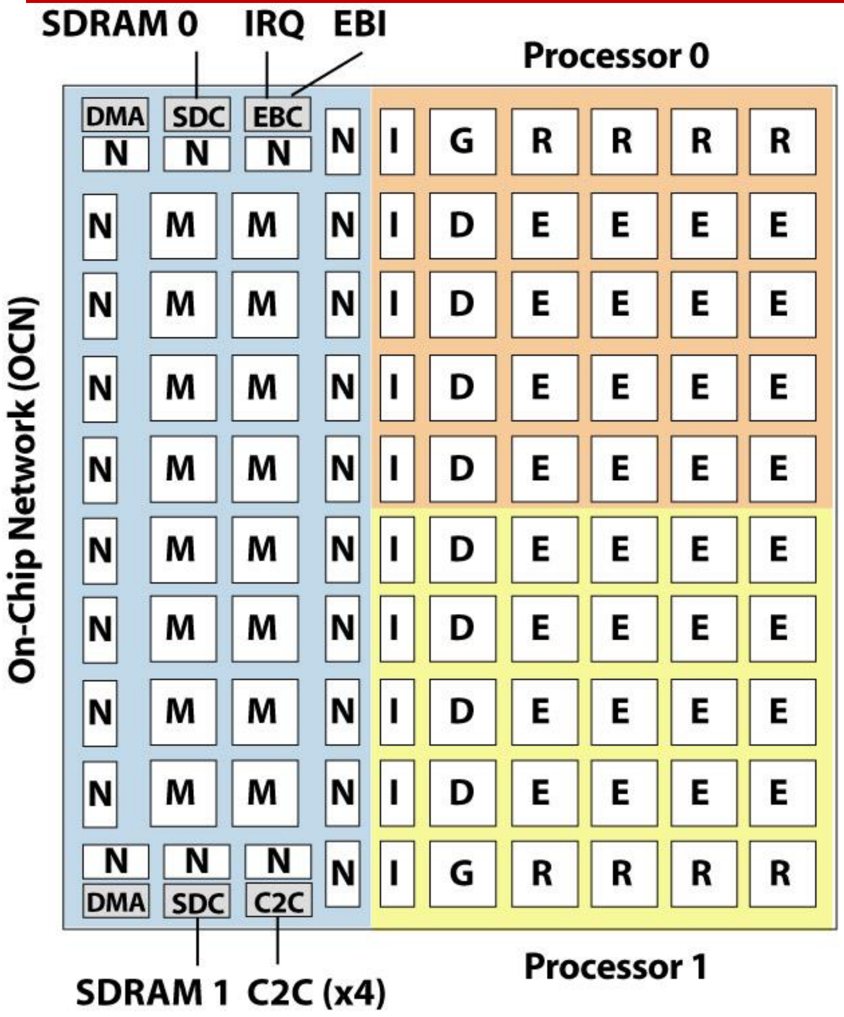


TRIPS Prototype Chip

- 2 TRIPS Processors
- NUCA L2 Cache
  - 1 MB, 16 banks
- On-Chip Network (OCN)
  - 2D mesh network
  - Replaces on-chip bus
- Controllers
  - 2 DDR SDRAM controllers
  - 2 DMA controllers
  - External Bus Controller (EBC)
    - Interfaces with PowerPC 440GP (control processor)
  - Chip-to-Chip (C2C) network controller
- Clocking
  - 2 PLLs
  - 4 Clock domains
    - 1x and 2x SDRAM
    - Main and C2C
  - Clock tree
    - Main domain has 4 quadrants to limit local skew



TRIPS Tile-level Microarchitecture



TRIPS Tiles

G: Processor control - TLB w/ variable size pages, dispatch, next block predict, commit

R: Register file - 32 registers x 4 threads, register forwarding

I: Instruction cache - 16KB storage per tile

D: Data cache - 8KB per tile, 256-entry load/store queue, TLB

E: Execution unit - Int/FP ALUs, 64 reservation stations

M: Memory - 64KB, configurable as L2 cache or scratchpad

N: OCN network interface - router, translation tables

DMA: Direct memory access controller

SDC: DDR SDRAM controller

EBC: External bus controller - interface to external PowerPC

C2C: Chip-to-chip network controller - 4 links to XY neighbors

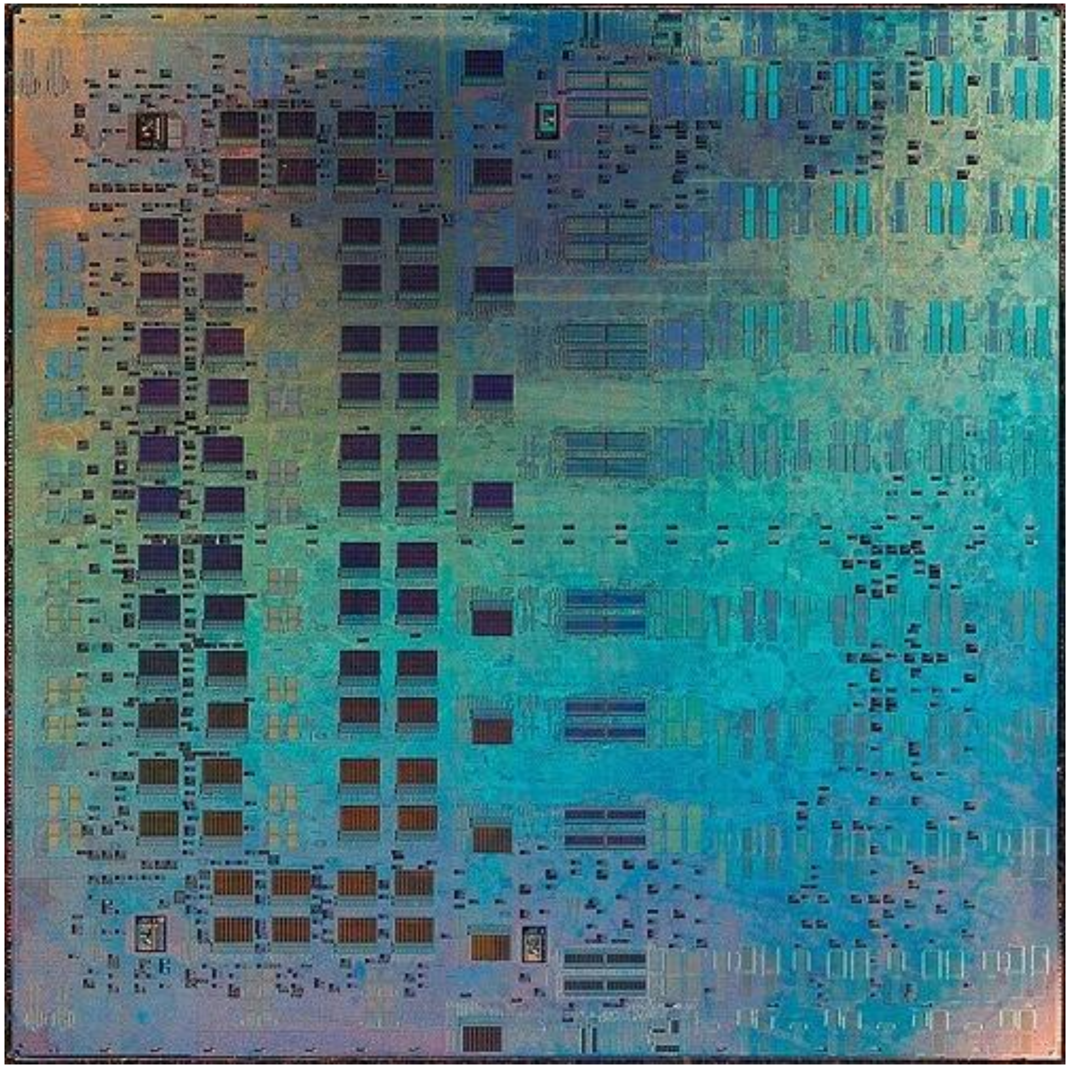
TRIPS Microarchitecture Principles

- Distributed and tiled architecture
  - Small and simple tiles (register file, data cache bank, etc.)
  - Short local wires
    - Tiles are small: 2-5 mm^2 per tile is typical
  - No centralized resources
- Networks connect the tiles
  - Networks implement distributed protocols (I-fetch, bypass, etc.)
    - Includes well-defined control and data networks
  - Networks connect only nearest neighbors
  - No global wires
- Design modularity and scalability
  - Design productivity by replicating tiles (design reuse)
  - Networks extensible, even late in design cycle

TRIPS Chip Implementation

Process Technology	130nm ASIC with 7 metal layers
Die Size	18.3mm x 18.37mm ($336\ mm^2$)
Package	47mm x 47mm BGA
Pin Count	626 signals, 352 Vdd, 348 GND
# of placed cells	6.1 million
Transistor count (est.)	170 million
# of routed nets	6.5 million
Total wire length	1.06 km
Power (measured)	36W at 366MHz (chip has no power mgt.)
Clock period	2.7ns (actual) 4.5ns (worse case sim)

Die photo

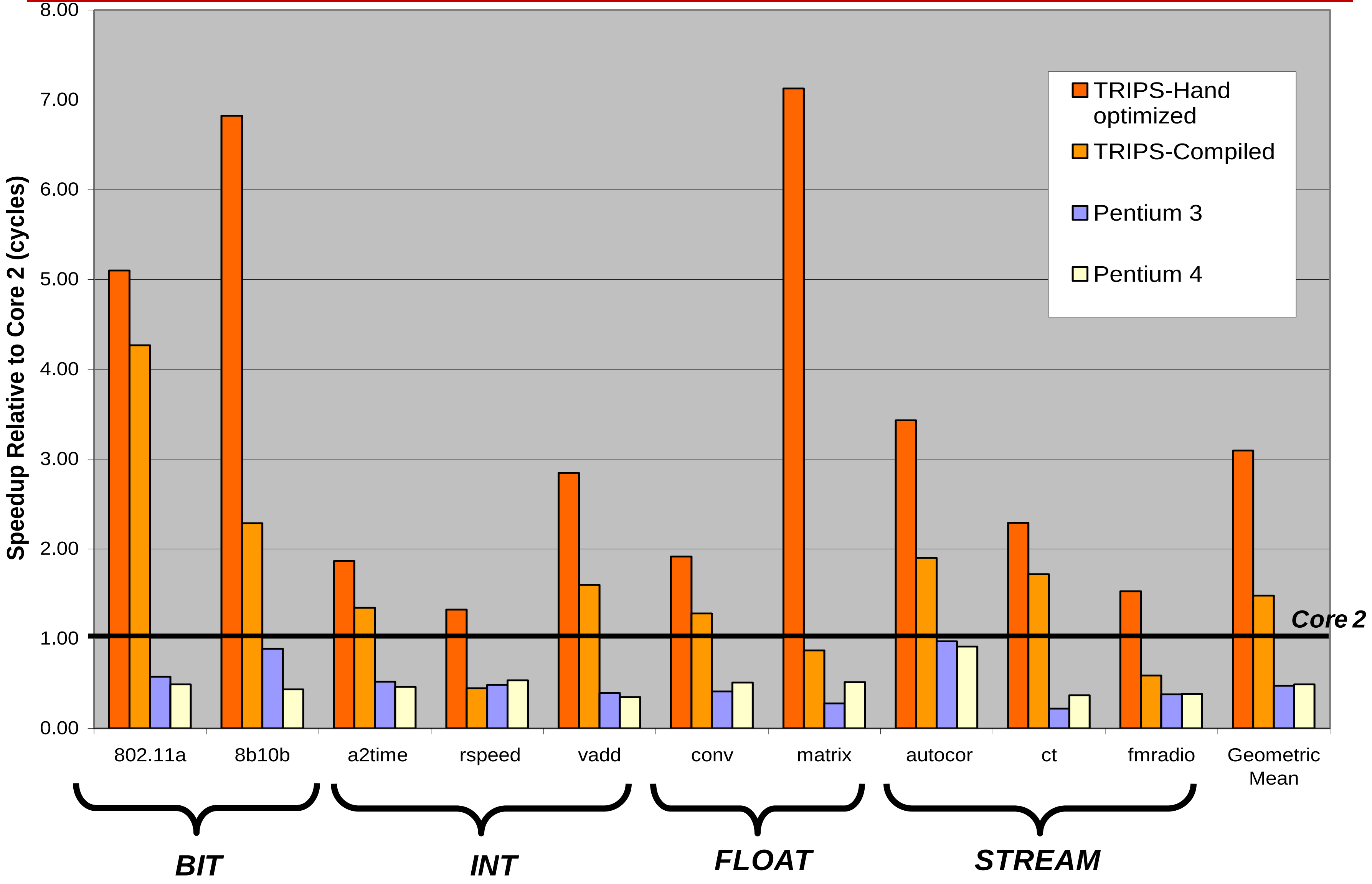


Preliminary Performance (HW)

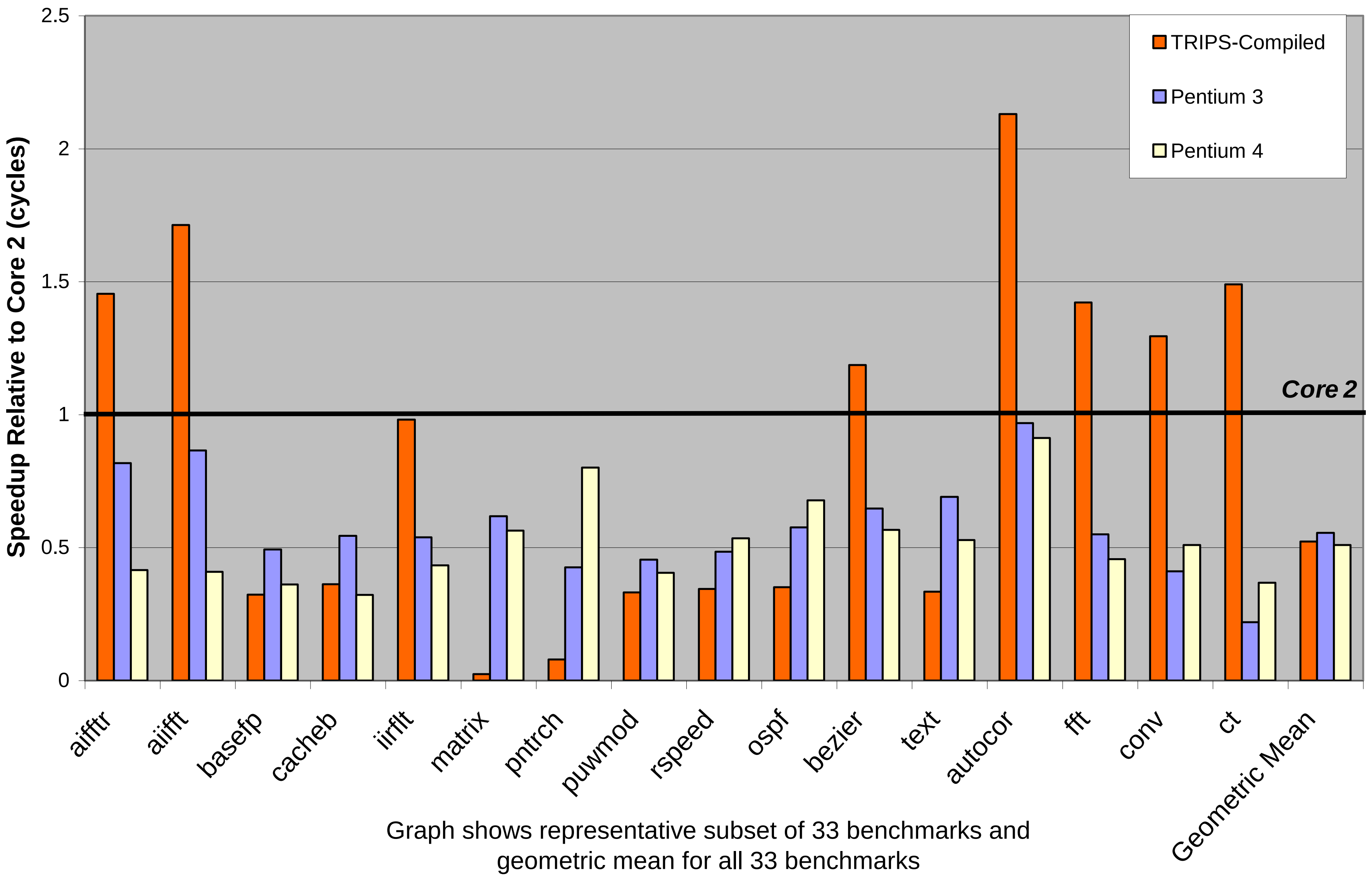
- Challenges
  - Different technology and ISAs
  - Different processor-to-memory clock ratio
  - TRIPS compiler fine-tuning in progress
- Cycle-to-cycle comparison on multiple HW platforms
  - TRIPS Performance counters
  - PAPI - Performance API on Linux systems for others
- Applications
  - Compiled + hand-optimized
    - Mix of kernels and full algorithms
  - Compiled only
    - The Embedded Microprocessor Benchmark Consortium (EEMBC)
    - Versabench (MIT)
  - SPEC benchmarks in progress

Processor	Clock Speed	Memory Speed	Process Technology
TRIPS	366 MHz	200 MHz DDR	130 nm
Core 2	1.6 GHz (underclocked)	800 MHz DDR2	65 nm
Pentium 4	3.6 GHz	533 MHz DDR2	90 nm
Pentium 3	450 MHz	100 MHz SDRAM	250 nm

TRIPS vs. Conventional Processors: Kernels



TRIPS vs. Conventional Processors EEMBC and signal processing (compiled)

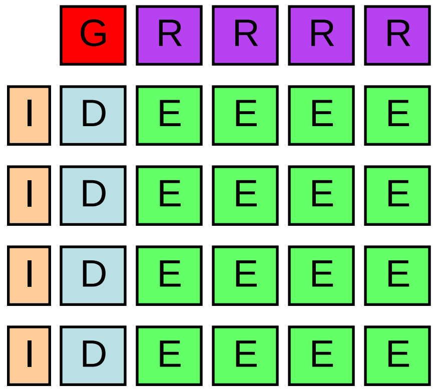


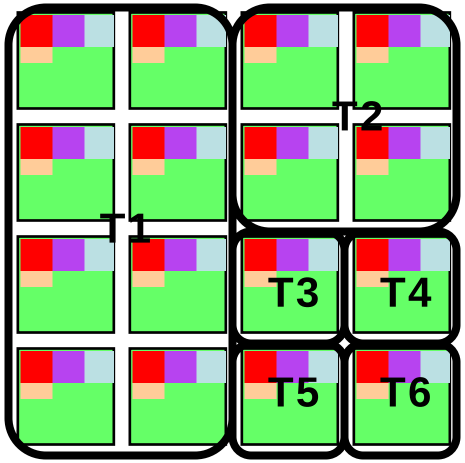
Graph shows representative subset of 33 benchmarks and geometric mean for all 33 benchmarks

Ongoing Work

- Performance tuning and analysis ongoing
  - Matrix multiply
    - 9 IPC
    - 5.8 FLOP/cycle
  - NAS Parallel benchmarks and other parallel apps
- Is granularity correct?
  - Tflex Microarchitecture and EDGE ISA
    - Fine-grain for parallel component
  - Group to form large uniprocessor for serial component
  - Fundamental architecture question?

Multigranular “Elastic” Threads





- Problems with TRIPS microarchitecture
  - Limited register/memory bandwidth
  - Number of tiles per core is fixed at design time
  - Multithreading is a hack to vary granularity
- Achievable by distributing all support tiles
  - Assume each tile can hold >= 1 block (128 insts.)
- Solutions being implemented to design challenges
  - Scalable cache capacity with number of tiles
  - Scalable memory bandwidth (at the processor interface)
- Does not address chip-level memory bandwidth

- **Config one: 1 thread, 16 blocks @ 8 insts/tile**
- **Config two: 2 threads, 1 block @ 128 insts/tile**
- **Config three: 6 threads, 1 thread on 8 tiles, 1 thread on 4 tiles, 4 threads on 1 tile each**

Looking Forward

- Area analysis shows by 2012
  - 256 tiles on chip
  - 32K instruction window on chip
  - Flexible partitioning of work
- Reliability of these PEs
  - Fine-grained redundancy
  - Make errors/failures first-class property
  - De-couple error detection and management

Conclusions

- ISA and microarchitecture can contribute
  - Don't just think more cores:
    - Uniprocessor important and provides opportunity
    - TRIPS: One programmability, performance, power tradeoff
  - Powerful and efficient uniprocessor is useful and *possible*
- Significant uncertainties remain
  - Device uncertainty
  - Heterogeneity? How many different designs will they support?
- Principles from application developers
  - Beyond desktop requirements
  - Fundamental application difference between COTS?
  - Or reconvergence?