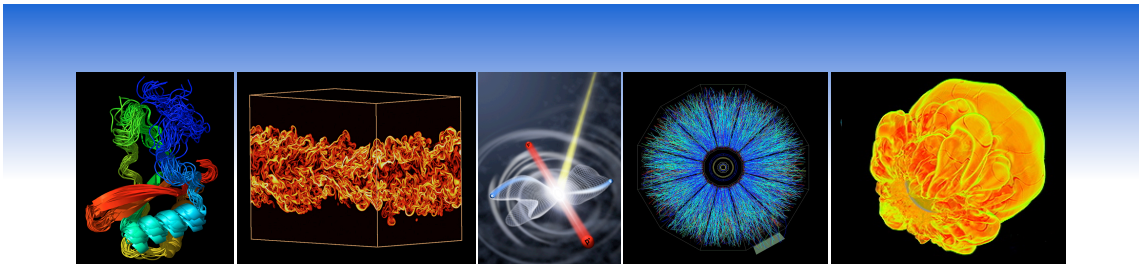


Green Flash: Exascale Computing on a Petascale Power Budget

John Shalf

David Donofrio, Leonid Oliker, Michael Wehner

National Energy Research Supercomputing Center

Lawrence Berkeley National Laboratory

iWAPT Workshop

Tokyo Japan, October 1, 2009

Office of Science U.S. DEPARTMENT OF ENERGY

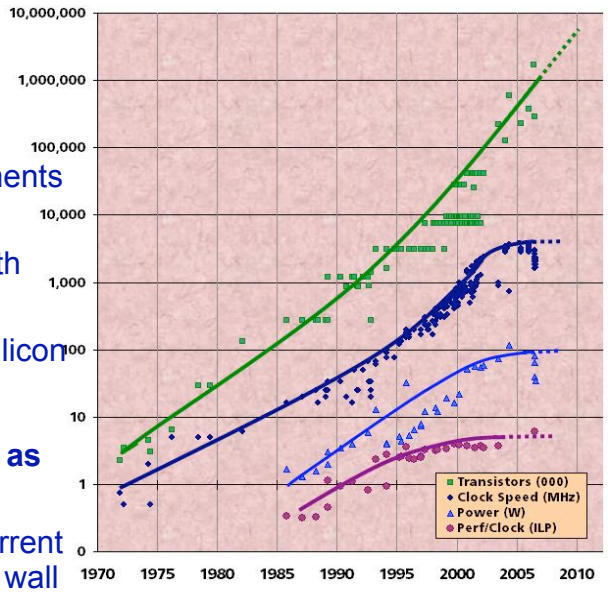
NERSC

BERKELEY LAB

New Design Constraint: *POWER*

NERSC NATIONAL ENERGY RESEARCH SCIENTIFIC COMPUTING CENTER

- **Transistors still getting smaller**
  - Moore's Law is alive and well
- **But Dennard scaling is dead!**
  - No power efficiency improvements with smaller transistors
  - No clock frequency scaling with smaller transistors
  - All "magical improvement of silicon goodness" has ended
- **Cannot continue with business as usual**
  - DARPA study extrapolated current design trends and found brick wall at end of exponential curves

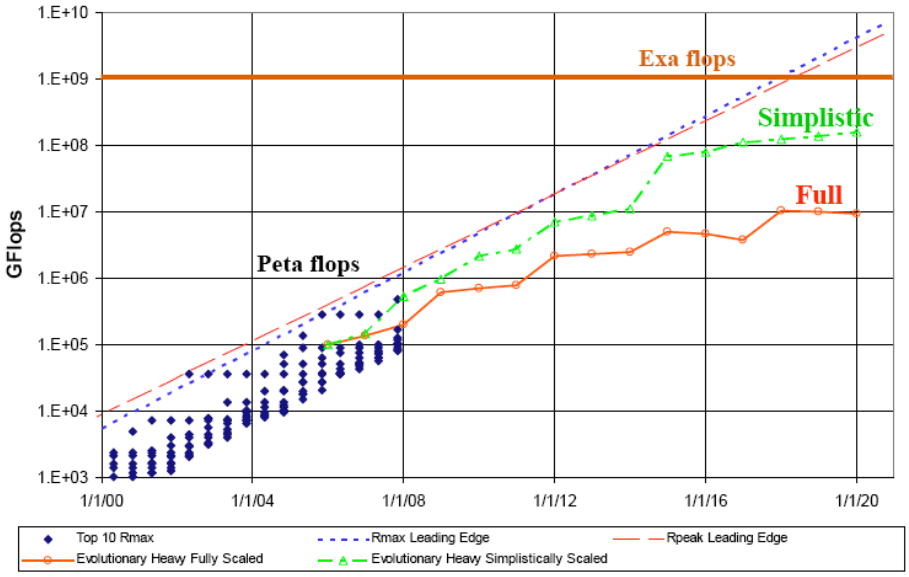


Olukotun et. al.

Office of Science U.S. DEPARTMENT OF ENERGY

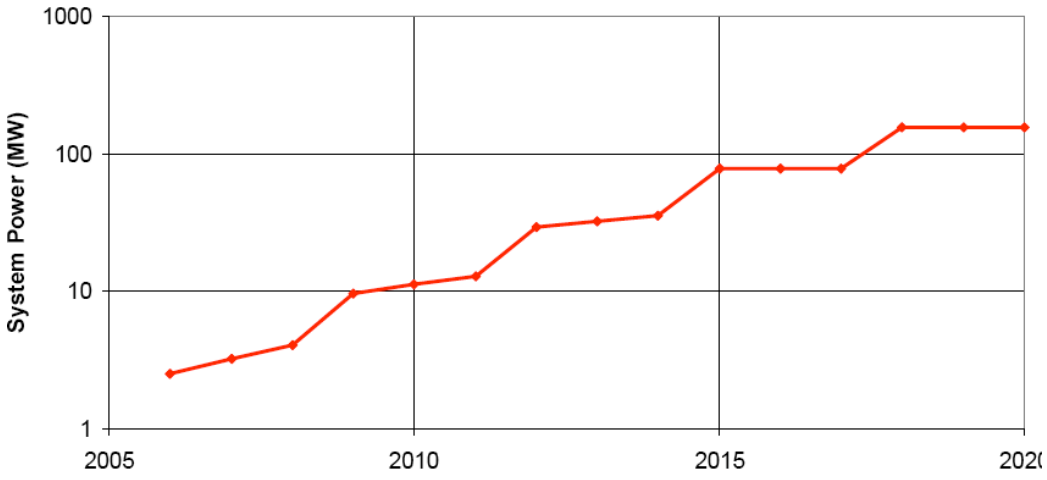
BERKELEY LAB

We won't reach Exaflops with the current approach



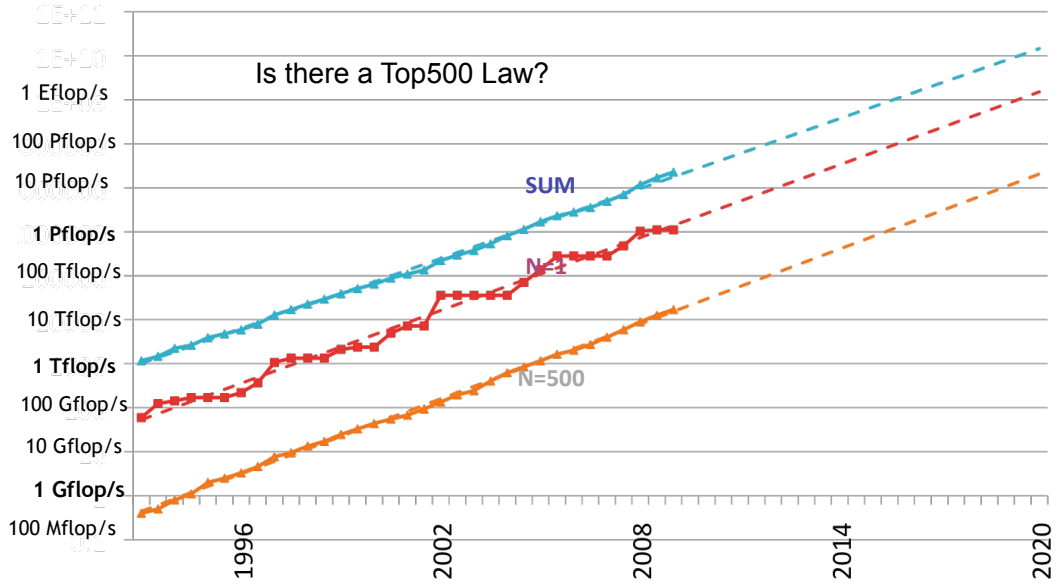
From Peter Kogge, DARPA Exascale Study

… and the power costs will still be staggering



From Peter Kogge, DARPA Exascale Study

Is Exascale a Sure Thing?



Office of Science
U.S. DEPARTMENT OF ENERGY

BERKELEY LAB

The Challenge

How to get 1000x performance without building a nuclear power plant next to my HPC center?

How do you achieve this in 10 years with a finite development budget?

How do you make it "programmable?"

Green Flash: Overview

We present an alternative approach to developing systems to serve the needs of scientific computing

- Choose our science target first to drive design decisions
- Leverage new technologies driven by consumer market
- ***Auto-tune software*** for performance, productivity, and portability
- Use hardware-accelerated architectural emulation to rapidly prototype designs (***auto-tune the hardware too!***)

- **Requires a holistic approach: Must innovate algorithm/software/hardware together (Co-tuning)**

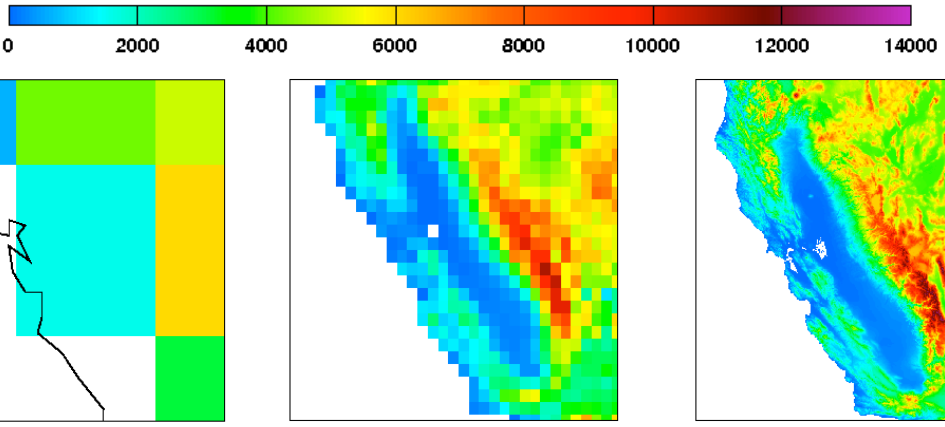
Achieve 100x energy efficiency improvement over mainstream HPC approach

An Application Driver:
Global Cloud Resolving Climate Model

Identify Target First!

(Global Cloud Resolving Climate Model)

Surface Altitude (feet)



200km
Typical resolution of IPCC AR4 models

25km
Upper limit of climate models with cloud parameterizations

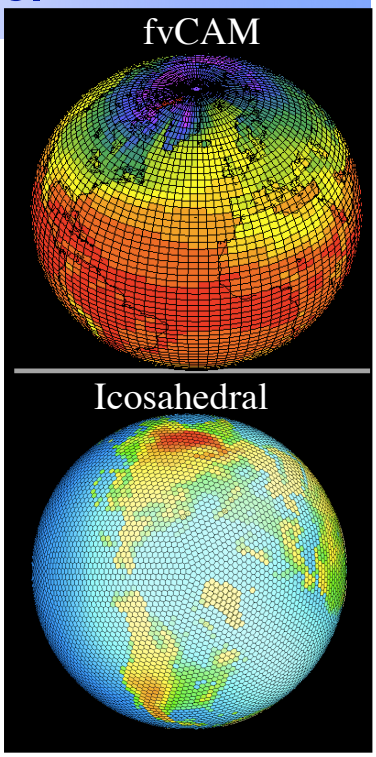
1km
Cloud system resolving models are a transformational change

Computational Requirements for 1km Climate Model

Must maintain 1000x faster than real time for practical climate simulation

- **~2 million horizontal subdomains**
- **100 Terabytes of Memory**
  - **5MB memory per subdomain**
- **~20 million total subdomains**
  - **20 PF sustained *(200PF peak)***
  - **Nearest-neighbor communication**

- ***New discretization for climate model***
  - ***CSU Icosahedral Code***



Energy Efficient Hardware Building Blocks

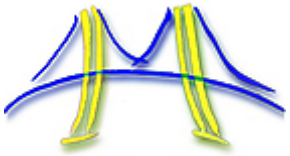
Mark Horowitz 2007: *"Years of research in low-power embedded computing have shown only one design technique to reduce power:* ***reduce waste****."*

Seymour Cray 1977: *"Don't put anything in to a supercomputer that isn't necessary."*

Hardware: What are the problems?

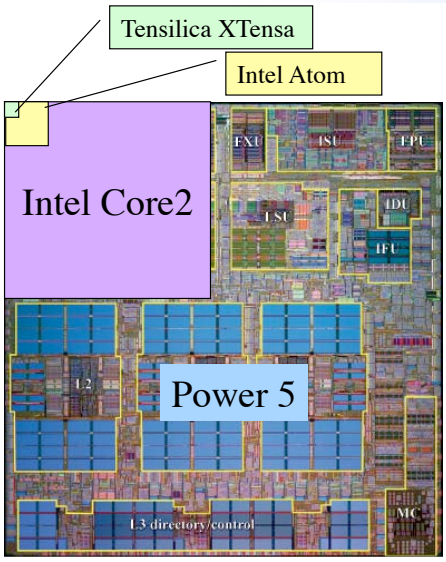
(Lessons from the Berkeley View)

- **Current Hardware/Lithography Constraints**
  - **Power limits leading edge chip designs**
    - Intel Tejas Pentium 4 cancelled due to power issues
  - **Yield on leading edge processes dropping dramatically**
    - IBM quotes yields of 10 – 20% on 8-processor Cell
  - **Design/validation leading edge chip is becoming unmanageable**
    - Verification teams > design teams on leading edge processors
- **Solution: Small Is Beautiful**



  - **Simpler (5- to 9-stage pipelined) CPU cores**
    - Small cores not much slower than large cores
  - **Parallel is energy efficient path to performance:CV^2F**
    - Lower threshold and supply voltages lowers energy per op
  - **Redundant processors can improve chip yield**
    - Cisco Metro 188 CPUs + 4 spares; Sun Niagara sells 6 or 8 CPUs
  - **Small, regular processing elements easier to verify**

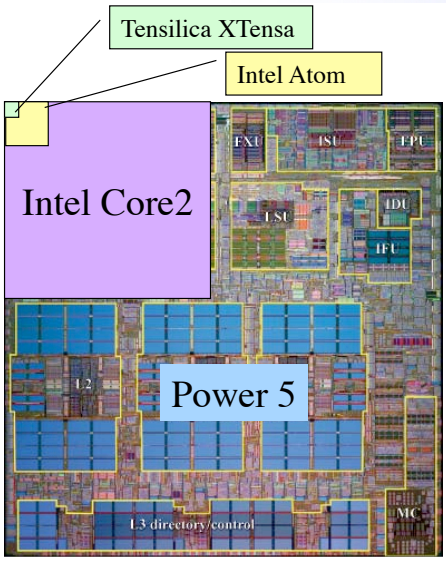
Low-Power Design Principles



- **Cubic power improvement with lower clock rate due to V^2F**
- **Slower clock rates enable use of simpler cores**
- **Simpler cores use less area (lower leakage) and reduce cost**
- **Tailor design to application to REDUCE WASTE**

This is how iPhones and MP3 players are designed to maximize battery life and minimize cost

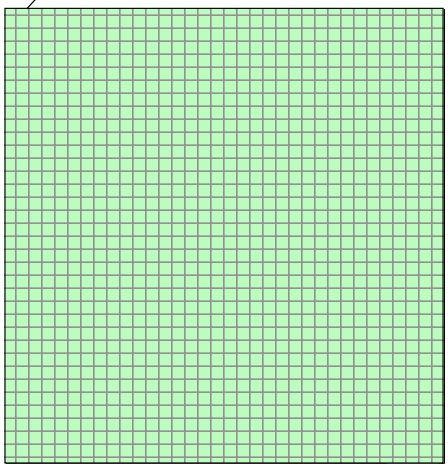
Low-Power Design Principles



- **Power5 (server)**
  - **120W@1900MHz**
  - **Baseline**
- **Intel Core2 sc (laptop) :**
  - **15W@1000MHz**
  - ***4x more FLOPs/watt than baseline***
- **Intel Atom (handhelds)**
  - **0.625W@800MHz**
  - **80x more**
- **Tensilica XTensa DP (Moto Razor) :**
  - **0.09W@600MHz**
  - **400x more *(80x-120x sustained)***

Office of Science U.S. DEPARTMENT OF ENERGY
BERKELEY LAB

Low Power Design Principles



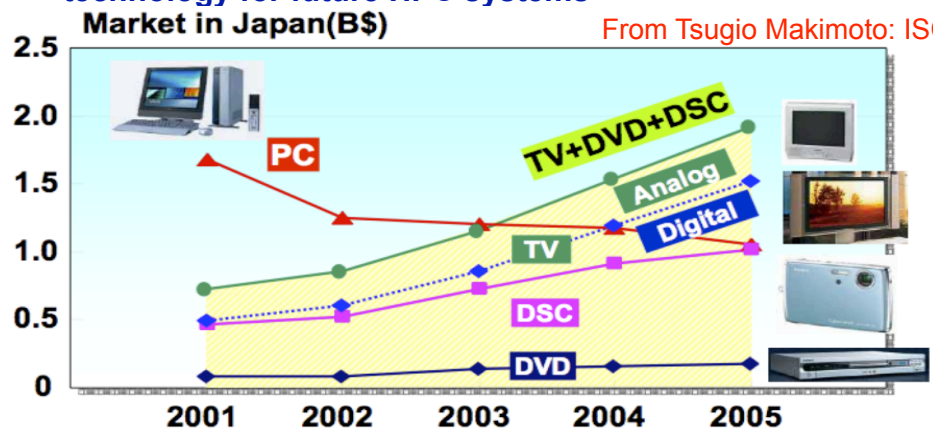
- Power5 (server)
  - 120W@1900MHz
  - Baseline
- Intel Core2 sc (laptop) :
  - 15W@1000MHz
  - *4x more FLOPs/watt than baseline*
- Intel Atom (handhelds)
  - 0.625W@800MHz
  - 80x more
- Tensilica XTensa DP (Moto Razor) :
  - 0.09W@600MHz
  - 400x more (80x-100x sustained)

Even if each simple core is 1/4th as computationally efficient as complex core, you can fit hundreds of them on a single chip and still be 100x more power efficient.

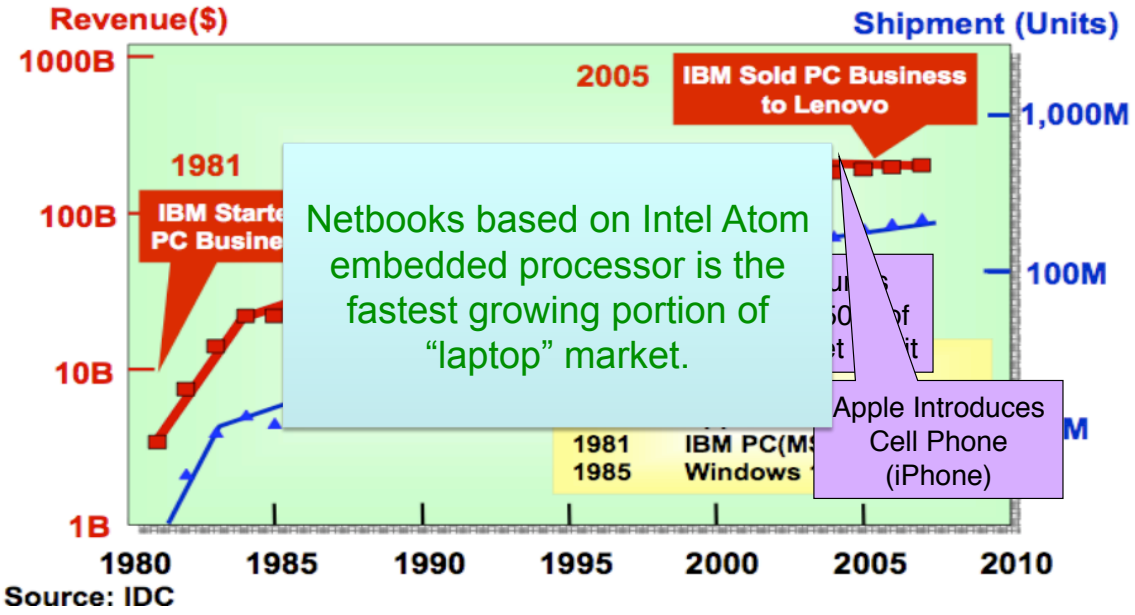
Technology Investment Trends

- 1990s - R&D computing hardware dominated by desktop/COTS
  - Had to learn how to use COTS technology for HPC
- 2010 - R&D investments moving rapidly to consumer electronics/ embedded processing
  - Must learn how to leverage embedded processor technology for future HPC systems

From Tsugio Makimoto: ISC2006



Consumer Electronics has Replaced PCs as the Dominant Market Force in CPU Design!!



From Tsugio Makimoto: ISC2006

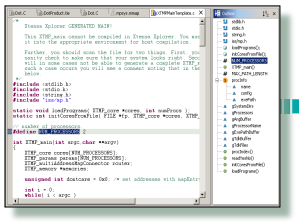
Embracing the Embedded Market

- **Have all of the IP and experience with for low-power technology**
- **Have sophisticated tools for rapid turn-around of designs**
- **Vibrant commodity market in IP components**

- **Convergence with HPC requirements**
  - Need better computational efficiency and lower power
  - Now we both must face parallelism

Embedded Design Automation

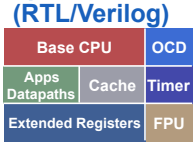
(Example from Existing Tensilica Design Flow)

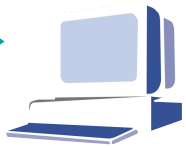


Processor configuration
1. Select from menu
2. Automatic instruction discovery (XPRES Compiler)
3. Explicit instruction description (TIE)

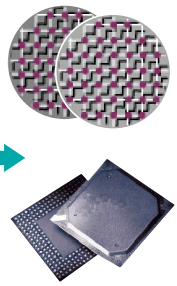
Processor Generator (*Tensilica*)

Application-optimized processor implementation (RTL/Verilog)





Tailored SW Tools: Compiler, debugger, simulators, Linux, other OS Ports *(Automatically generated together with the Core)*



Build with any process in any fab

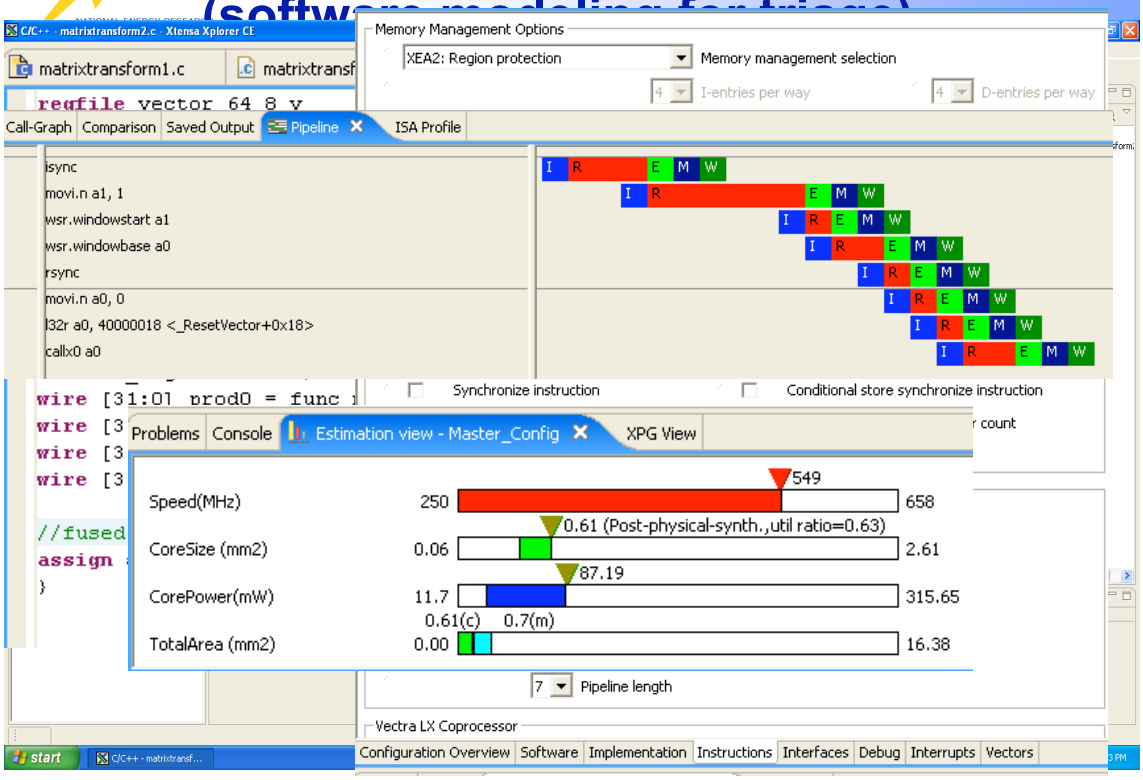
Bringing Autotuning to the Hardware

- **Software Design Space Exploration: "auto-tuning"**
  - Auto-search through parameter space of code optimizations
  - Tune to diverse & complex hardware
- **Hardware Design Space Exploration:**
  - What if hardware configuration was also parameterized?
  - Search through diverse space of hardware configurations
- **What if you could do both together?**
  - Auto-tune software for hardware
  - Auto-tune hardware for software
  - Repeat?
- **Hardware/Software co-design**
  - Demonstrate how to apply to HPC
  - Enable Energy Efficient computing for Extreme Scale Science

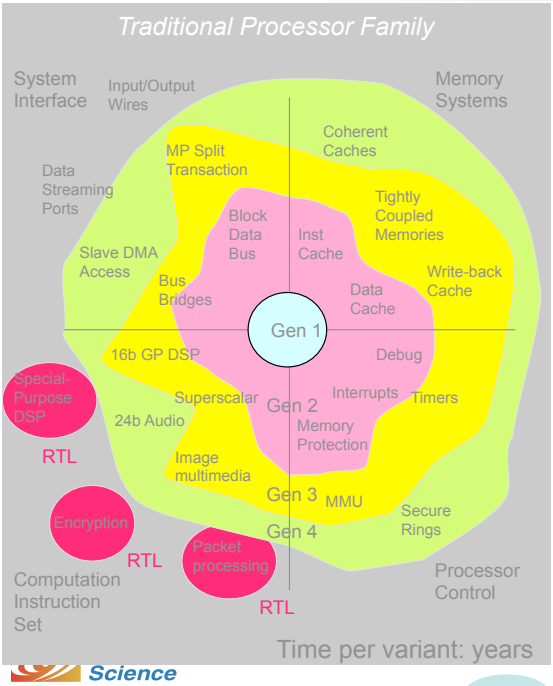
A Parameterized Core Design To Enable Hardware Auto-Tuning

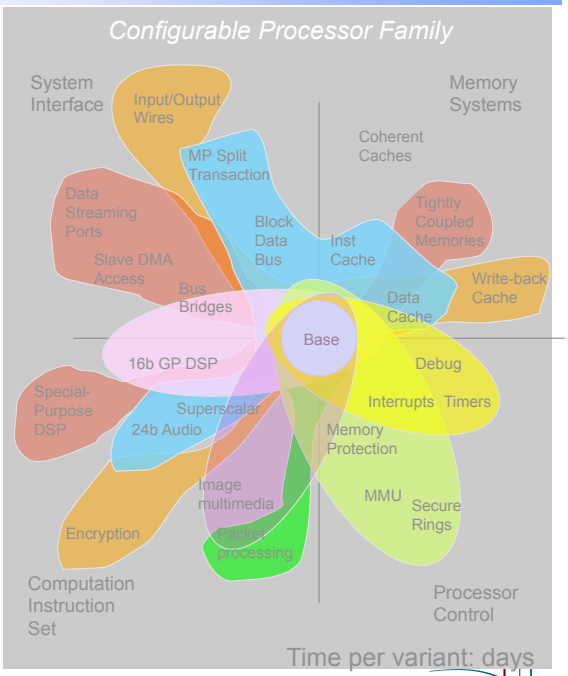
- **Highly configurable**
  - Custom VLIW support
  - Configure #registers, width, ISA
  - Configure memory subsystem, local-store, cache org
- **Verilog-like TIE language allows custom ISA extensions**
  - Functional and performance verification built in
  - Auto generated compiler intrinsics
  - 64-bit IEEE-DP floating point coded up in TIE and available
- **Inter-processor communication easily enabled through:**
  - TIE Ports
  - TIE Queues
    - Access to direct HW support for interprocessor communication
  - TIE Lookups
    - Allows interface to external ROMs or other RTL block

Processor Generator



Peel Back the Historical Growth of Instruction Sets *(accretion of junk!)*





Area = silicon cost and power

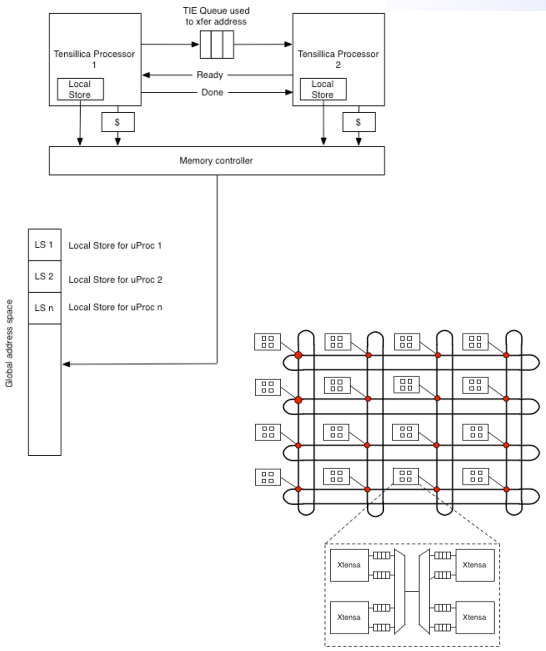
A Short List of x86 Opcodes that Science Applications Don't Need!

mnemonic	op1	op2	op3	op4	iext	mf	0f	op	es	s	proc	m	rl	x	tested f	modif f	def f	undef f	f values	description, notes
AAA	AL	AH						37							.....a..	o..szapc	.....a.c	o..sz.p.		ASCII Adjust After Addition
AAD	AL	AH						D5	0A							o..szapc	...sz.p.	o.....c		ASCII Adjust AX Before Division
AAM	AL	AH						D4	0A							o..szapc	...sz.p.	o.....c		ASCII Adjust AX After Multiply
AAS	AL	AH						3F							.....a..	o..szapc	.....a.c	o..sz.p.		ASCII Adjust AL After Subtraction
ADC	r/m8	r8						10		r				L	c	o..szapc	o..szapc			Add with Carry
ADC	r/m16/32/64	r16/32/64						11		r				L	c	o..szapc	o..szapc			Add with Carry
ADC	r8	r/m8						12		r					c	o..szapc	o..szapc			Add with Carry
ADC	r16/32/64	r/m16/32/64						13		r					c	o..szapc	o..szapc			Add with Carry
ADC	AL	imm8						14							c	o..szapc	o..szapc			Add with Carry
ADC	rAX	imm16/32						15							c	o..szapc	o..szapc			Add with Carry
ADC	r/m8	imm8						80		2				L	c	o..szapc	o..szapc			Add with Carry
ADC	r/m16/32/64	imm16/32						81		2				L	c	o..szapc	o..szapc			Add with Carry
ADC	r/m8	imm8						82		2				L	c	o..szapc	o..szapc			Add with Carry
ADC	r/m16/32/64	imm8						83		2				L	c	o..szapc	o..szapc			Add with Carry
ADD	r/m8	r8						00		r				L		o..szapc	o..szapc			Add
ADD	r/m16/32/64	r16/32/64						01		r				L		o..szapc	o..szapc			Add
ADD	r8	r/m8						02		r						o..szapc	o..szapc			Add
ADD	r16/32/64	r/m16/32/64						03		r						o..szapc	o..szapc			Add
ADD	AL	imm8						04								o..szapc	o..szapc			Add
ADD	rAX	imm16/32						05								o..szapc	o..szapc			Add
ADD	r/m8	imm8						80		0				L		o..szapc	o..szapc			Add
ADD	r/m16/32/64	imm16/32						81		0				L		o..szapc	o..szapc			Add
ADD	r/m8	imm8						82		0				L		o..szapc	o..szapc			Add
ADD	r/m16/32/64	imm8						83		0				L		o..szapc	o..szapc			Add
ADDPD	xmm	xmm/m128			sse2	66	0F	58		r	P4+									Add Packed Double-FP Values
ADDPS	xmm	xmm/m128			sse1		0F	58		r	P2+									Add Packed Single-FP Values
ADDSD	xmm	xmm/m64			sse2	F2	0F	58		r	P4+									Add Scalar Double-FP Values
ADDSS	xmm	xmm/m32			sse1	F3	0F	58		r	P2+									Add Scalar Single-FP Values
ADDSUBPD	xmm	xmm/m128			sse3	66	0F	D0		r	P4++									Packed Double-FP Add/Subtract
ADDSUBPS	xmm	xmm/m128			sse3	F2	0F	D0		r	P4++									Packed Single-FP Add/Subtract
ADX	AL	AH	imm8					D5								o..szapc	...sz.p.	o.....c		Adjust AX Before Division
ALTER						64					P4+	yt								Alternating branch prefix (used only with Jcc instructions)
AMX	AL	AH	imm8					D4								o..szapc	...sz.p.	o.....c		Adjust AX After Multiply
AND	r/m8	r8						20		r				L		o..szapc	o..sz.pc	.....a..	o.....c	Logical AND
AND	r/m16/32/64	r16/32/64						21		r				L		o..szapc	o..sz.pc	.....a..	o.....c	Logical AND
AND	r8	r/m8						22		r						o..szapc	o..sz.pc	.....a..	o.....c	Logical AND
AND	r16/32/64	r/m16/32/64						23		r						o..szapc	o..sz.pc	.....a..	o.....c	Logical AND
AND	AL	imm8						24								o..szapc	o..sz.pc	.....a..	o.....c	Logical AND
AND	rAX	imm16/32						25								o..szapc	o..sz.pc	.....a..	o.....c	Logical AND
AND	r/m8	imm8						80		4				L		o..szapc	o..sz.pc	.....a..	o.....c	Logical AND
AND	r/m16/32/64	imm16/32						81		4				L		o..szapc	o..sz.pc	.....a..	o.....c	Logical AND
AND	r/m8	imm8						82		4				L		o..szapc	o..sz.pc	.....a..	o.....c	Logical AND
AND	r/m16/32/64	imm8						83		4	02+			L		o..szapc	o..sz.pc	.....a..	o.....c	Logical AND
ANDNPD	xmm	xmm/m128			sse2	66	0F	55		r	P4+									Bitwise Logical AND NOT of Packed Double-FP Values
ANDNPS	xmm	xmm/m128			sse1		0F	55		r	P2+									Bitwise Logical AND NOT of Packed Single-FP Values
ANDPD	xmm	xmm/m128			sse2	66	0F	54		r	P4+									Bitwise Logical AND of Packed Double-FP Values
ANDPS	xmm	xmm/m128			sse1		0F	54		r	P2+									Bitwise Logical AND of Packed Single-FP Values

More Wasted Opcodes

- **We only need 80 out of the nearly 300 ASM instructions in the x86 instruction set!**
- Still have all of the 8087 and 8088 instructions!
- Wide SIMD Doesn't Make Sense with Small Cores
- Neither does Cache Coherence
- Neither does HW Divide or Sqrt for loops
  - Creates pipeline bubbles
  - Better to unroll it across the loops (like IBM MASS libraries)
- Move TLB to memory interface because its still too huge (but still get precise exceptions from segmented protection on each core)

Science-Optimized Processor Design



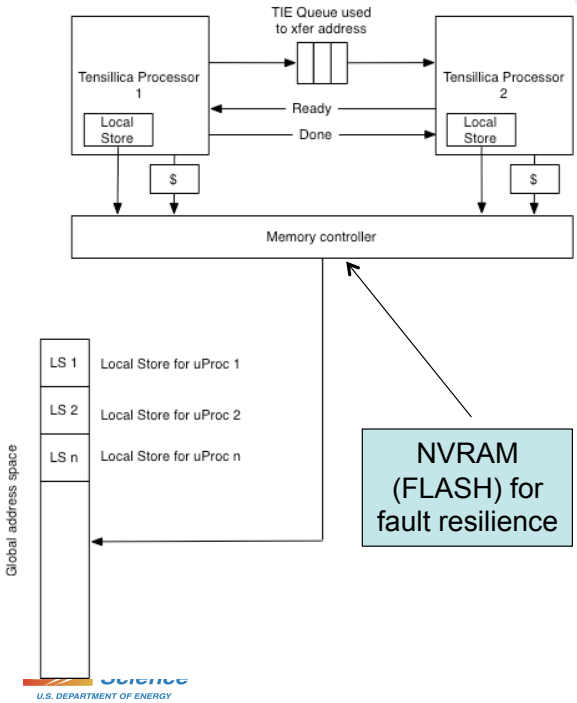
	Intel Core2 (Penryn)	Intel Atom core	Tensilica core w/ 64-bit FP
Die area (mm²)	53.5	25	5.32
Process	45 nm	45 nm	65 nm
Power	18W	0.625W	0.091W
Freq	2930 MHz	800MHz	500MHz
Flops / Watt	162	1280	***4065***

Novel Inter-processor Communication

Direct support for high-level language constructs

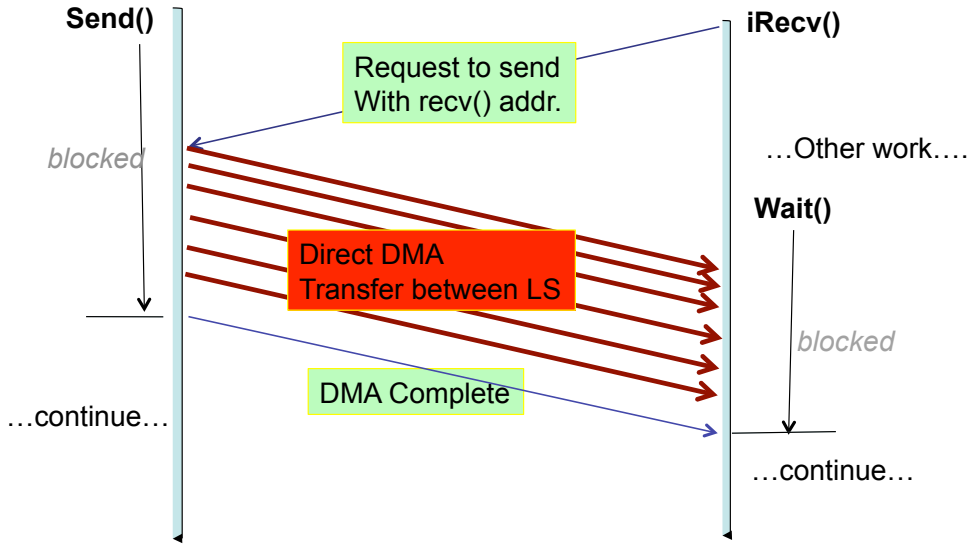
Architectural Support for IPC

Make hardware easier to program!



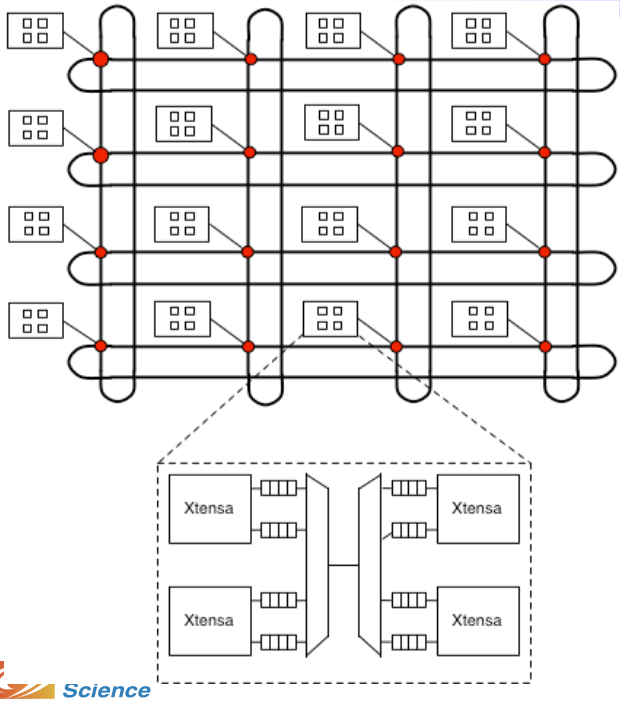
- **Logical topology is a full crossbar**
- **Each local store mapped to global address space**
- **To initiate a DMA transfer between processors:**
  - **Processors exchange starting addresses through TIE Queue interface**
    - **Optimized for small transfers**
  - **When ready, copy done directly from LS to LS**
  - **Copy will bypass cache hierarchy**

Example Timing Diagram For MPI-Like 2-sided Message



Green is TIE queue and red is Local Store DMA Engine

Network-on-Chip (NoC) Architecture



- Concentrated torus
  - Direct connect between 4 processors on a tile
  - Packet switched network connecting tiles
- Between 64 and 128 processors per die
- Silicon Photonics as option for NoC

Fault Resilience

If you have a 20Million Core System, you should expect some failures

Fault Resilience/Checkpointing

- **Industry has strong motivation to keep fault rates per node under control**
  - Historically target has been to make hardware slightly more reliable than dominant OS (MS Windows)
  - Hardening is done in circuit design (transparent to software)
- **However, HPC capability has historically grown faster than Moore's law!**
  - Top500 shows consistent 1000x increase in performance in 10 years *(Moore's law only gets you ~100x in same period)*
  - Therefore, number of nodes increasing (and hence failures)
  - Fault resilience creates load imbalance

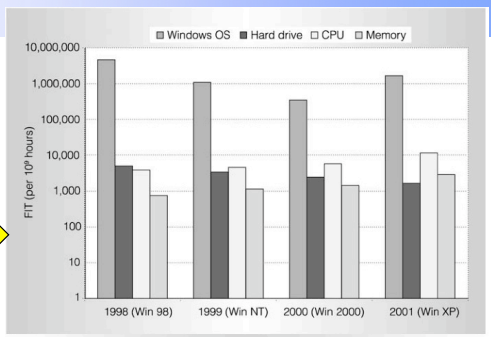
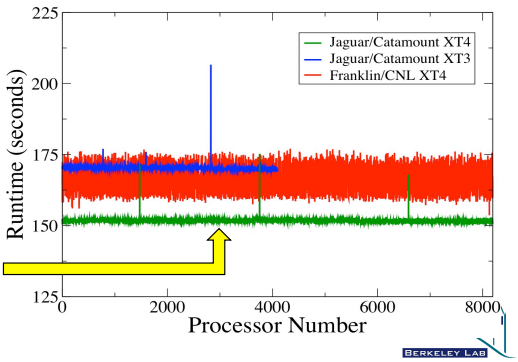
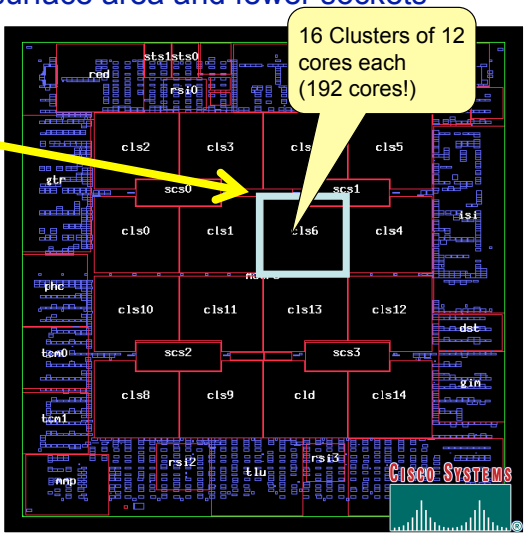


Figure 2. Failures in billions of hours of operation.[2-5]



Green Flash: Fault Tolerance/Resilience

- ***Large scale applications must tolerate node failures***
- **Our design does not expose unique risks**
  - Faults proportional to sockets (not cores) & silicon surface area
  - Low-power manycore uses less surface area and fewer sockets
- **Hard Errors**
  - Spare cores in design (Cisco Metro: 188 cores + 8 spares)
  - SystemOnChip design (fewer components→fewer sockets)
- **Soft Errors**
  - ECC for memory and caches
  - On-board NVRAM controller for localized checkpoint

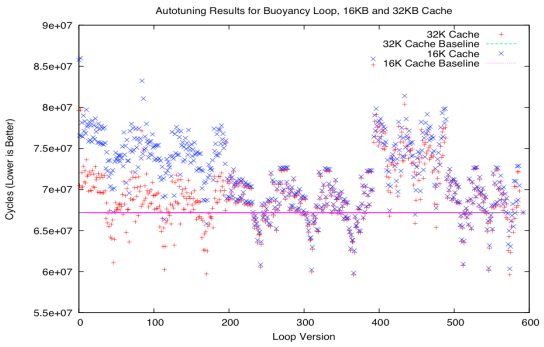


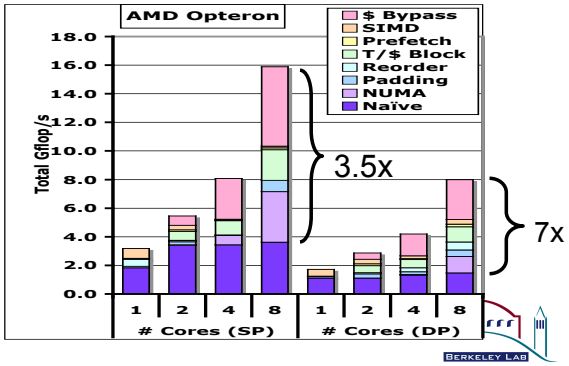
Software Performance

Software Auto-tuning: Don't depend on a human to do a machine's job.

Software Auto-tuning

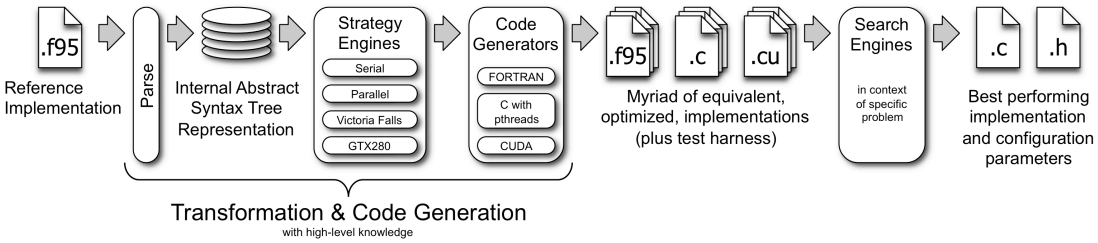
- **Problem: want to compare best potential performance of diverse architectures, avoiding**
  - Non-portable code
  - Labor-intensive user optimizations for each specific architecture
- **Our Solution: Auto-tuning**
  - **Automate search across a complex optimization space**
  - **Achieve performance far beyond current compilers**
  - **achieve performance portability for diverse architectures!**





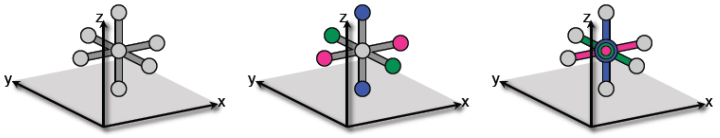
Generalized Stencil Auto-tuning Framework

- **Ability to tune many stencil-like kernels**
  - No need to write kernel-specific perl scripts
  - Uses semantic information from existing Fortran
- **Target multiple architectures**
  - Search over many optimizations for each architecture
  - Currently supports multi/manycore, GPUs
- **Better performance = Better energy efficiency**



Multi-Targeted Auto-Tuning

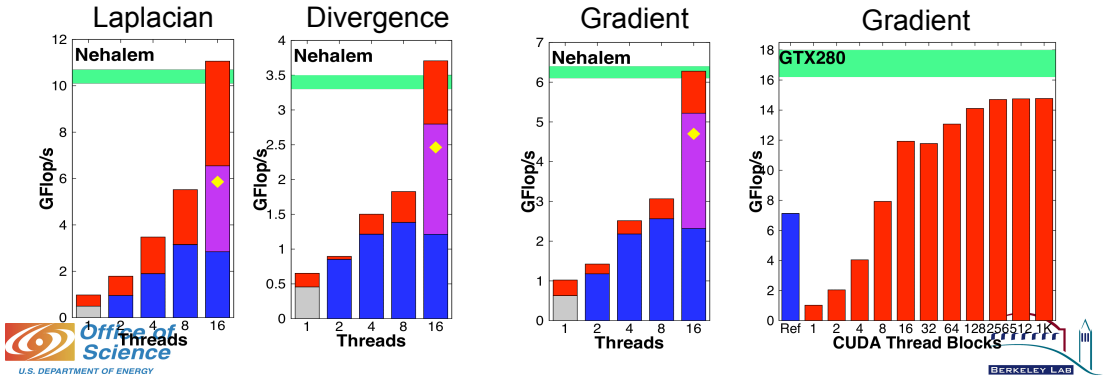
For Performance Portability



```
do k=2,nz-1,1
do j=2,ny-1,1
do i=2,nx-1,1

 uNext(i,j,k)=
  alpha*u(i,j,k)+
  beta*(u(i+1,j,k)+u(i-1,j,k)+
        u(i,j+1,k)+u(i,j-1,k)+
        u(i,j,k+1)+u(i,j,k-1)
       )
enddo
enddo
enddo
```

```
do k=2,nz-1,1
do j=2,ny-1,1
do i=2,nx-1,1

 u(i,j,k)=
  alpha*( x(i+1,j,k)-x(i-1,j,k) )+
   beta*( y(i,j+1,k)-y(i,j-1,k) )+
  gamma*( z(i,j,k+1)-z(i,j,k-1) )

enddo
enddo
enddo
```

```
do k=2,nz-1,1
do j=2,ny-1,1
do i=2,nx-1,1

  x(i,j,k)=alpha*( u(i+1,j,k)-u(i-1,j,k) )
  y(i,j,k)= beta*( u(i,j+1,k)-u(i,j-1,k) )
  z(i,j,k)=gamma*( u(i,j,k+1)-u(i,j,k-1) )

enddo
enddo
enddo
```

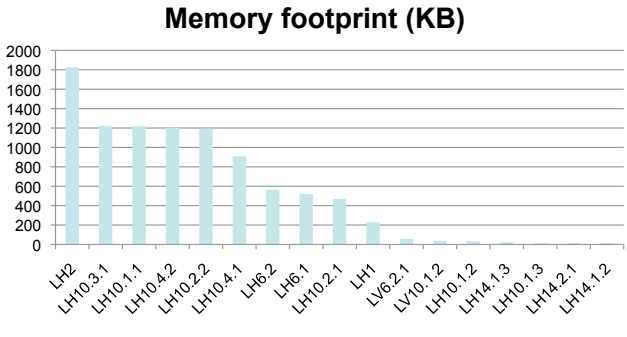


Office of Science
U.S. DEPARTMENT OF ENERGY

Analyze Climate Code Memory Movement

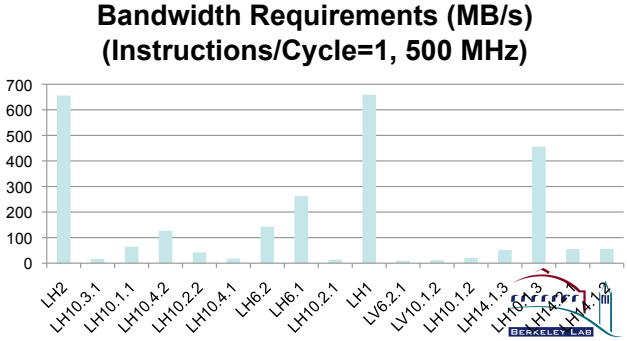
Optimized Data Movement: Huge Savings in Energy Efficiency and Cost

- **Analyzed Each Loop of Climate code Individually**
- **Trace analysis key to memory requirements**
  - **Actually running the code gives realistic values for memory footprint, temporal reuse, DRAM bandwidth requirements**



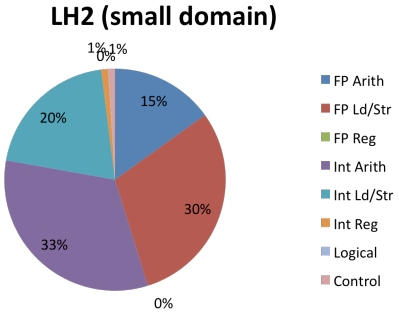
- **Measure DRAM bandwidth for each loop!**
  - **(instruction throughput) X (memory footprint)/ (instruction counts)**

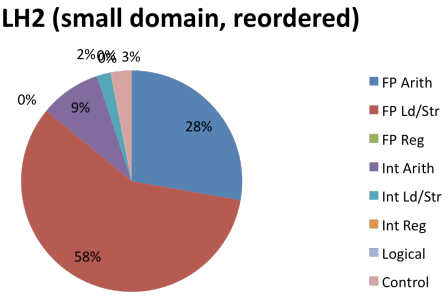
1-byte-per-FLOP could be reduced with local-store



Office of Science
U.S. DEPARTMENT OF ENERGY

Optimizing Instruction Mix





- Memory footprint: 160 KB
- Cache size requirement: 160 KB
- < 50% instructions are floating-point
  - Huge overhead for address generation
- Although code streams through data, loop ordering was bad ➔ cachelines reused although addresses were not

- Memory footprint: 160 KB
- Cache size requirement: 1 KB
- > 85% instructions are floating-point
  - Good ordering ➔ simpler addressing

160x reduction in cache size!
2x savings in execution time

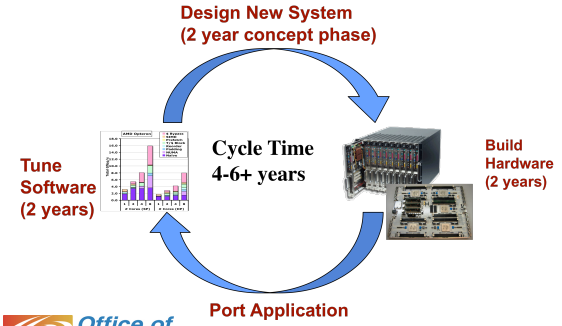
Rapid Prototyping of System Design

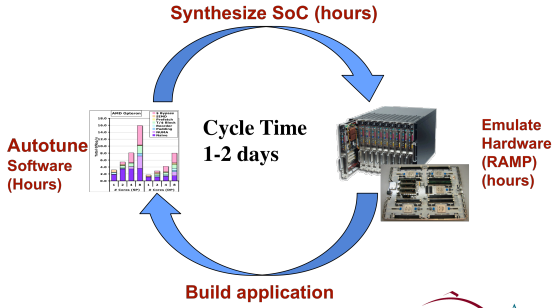
Using RAMP to Accelerate the hardware/software co-design cycle

Advanced Hardware Simulation (RAMP)

Enabling Hardware/Software/Science Co-Design

- **Research Accelerator for Multi-Processors (RAMP)**
  - **Simulate hardware *before* it is built!**
  - **Break slow feedback loop for system designs**
  - **Enables tightly coupled hardware/software/science co-design** *(not possible using conventional approach)*



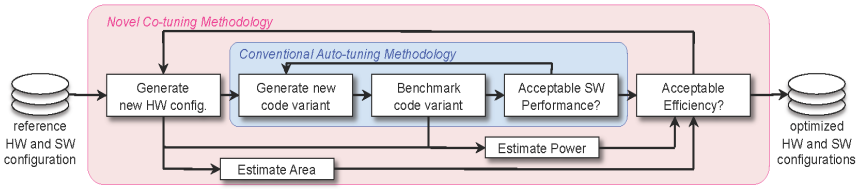


Office of Science
U.S. DEPARTMENT OF ENERGY

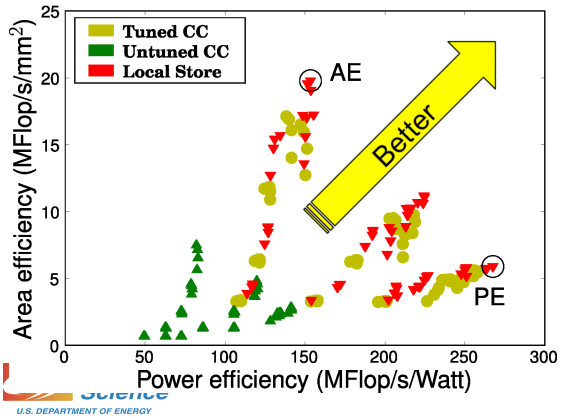
BERKELEY LAB

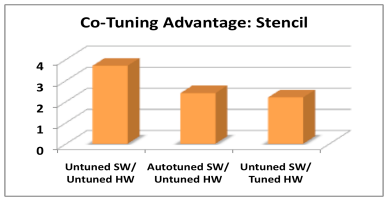
Hardware/Software Co-Tuning for Energy Efficiency

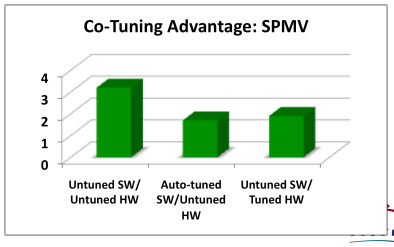
The approach: *Use auto-tuned code when evaluating architecture design points*



Co-Tuning can improve power-efficiency and area-efficiency by ~4x







Office of Science
U.S. DEPARTMENT OF ENERGY

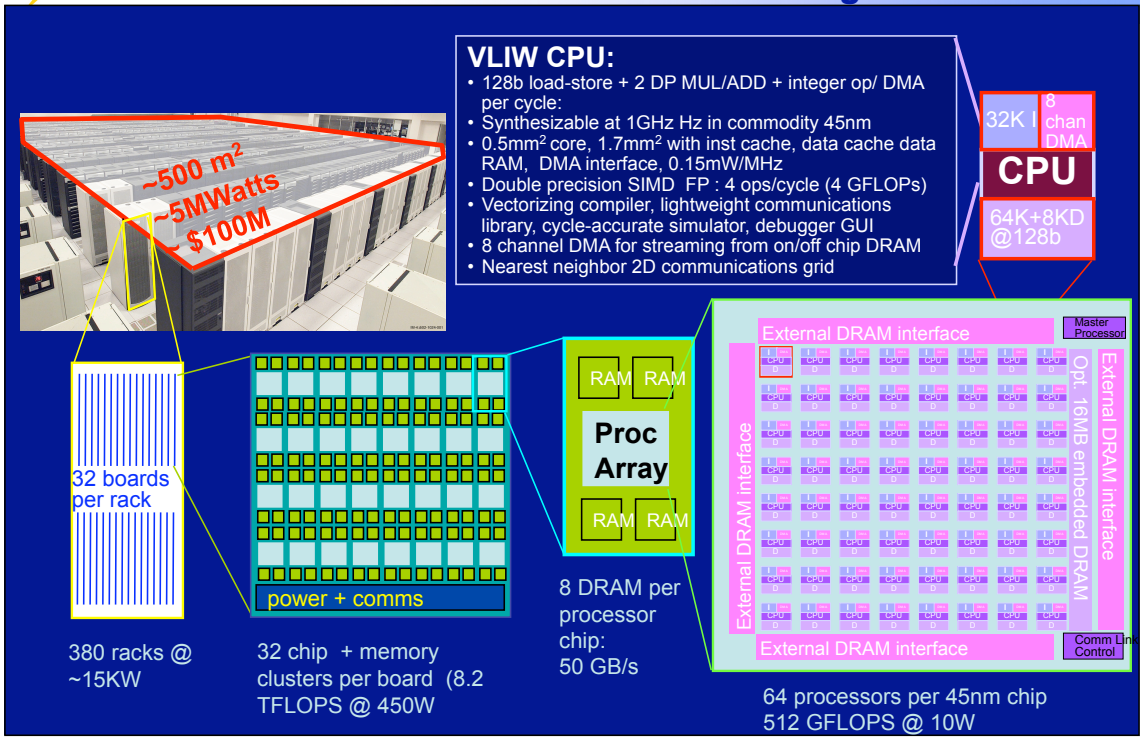
BERKELEY LAB

Lets Put it All Together!

Strawman Design

Climate Modeling System

Strawman 100PF Design



Green Flash Strawman System Design In 2008

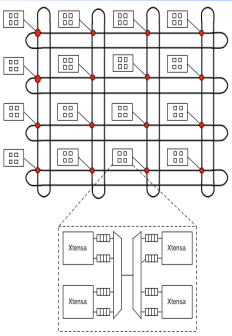
We examined three different approaches:

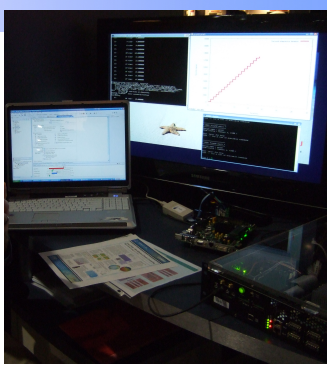
- **AMD Opteron:** Commodity approach, lower efficiency for scientific applications offset by cost efficiencies of mass market
- **BlueGene:** Generic embedded processor core and customize system-on-chip (SoC) services to improve power efficiency for scientific applications
- **Tensilica XTensa:** Customized embedded CPU w/SoC provides further power efficiency benefits but maintains programmability

Processor	Clock	Peak/ Core (Gflops)	Cores/ Socket	Sockets	Cores	Power	Cost 2008
AMD Opteron	2.8GHz	5.6	2	890K	1.7M	179 MW	$1B+
IBM BG/P	850MHz	3.4	4	740K	3.0M	20 MW	$1B+
Green Flash / Tensilica XTensa	650MHz	2.7	32	120K	4.0M	3 MW	$75M

SC08 Green Flash Hardware Demo

- **Demonstrated during SC '08**
- **Proof of concept**
  - CSU atmospheric model ported to Tensilica Architecture
  - Single Tensilica processor running atmospheric model at 50MHz
- **Emulation performance advantage**
  - Processor running at 50MHz vs. Functional model at 100 kHz
  - *500x Speedup*
- ***Actual code running - not representative benchmark***





If you Optimize the Processor Then Data Movement Will be the Problem

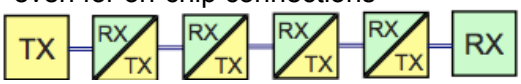
An "Amdahl's Law" for energy efficient hardware design

The problem with Wires: *Energy to move data proportional to distance*

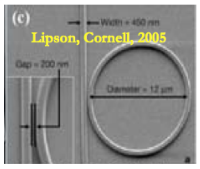
- **Wire cost to move a bit: (Telegraph Eqn.)**
  - **energy = bitrate * Length² / cross-section area**
  - **On-Chip (1cm): ~1pJ/bit, 100Tb/s**
  - **On-Module (5cm): ~2-5pJ/bit, 10Tb/s**
  - **On-Board (20cm): ~10pJ/bit, 1Tb/s**
  - **Intra-rack (1m): ~10-15pJ/bit, 1Tb/s**
  - **Inter-cabinet(2-50m): 15-30pJ/bit, 5-10Tb/s aggregate**
- **To move a bit with optics: target ~1-2pJ/bit for all distance scales**

Photonics requires no redrive and passive switch little power

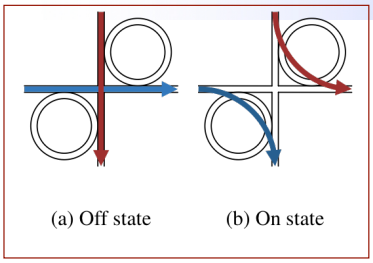
Copper requires to signal amplification even for on-chip connections

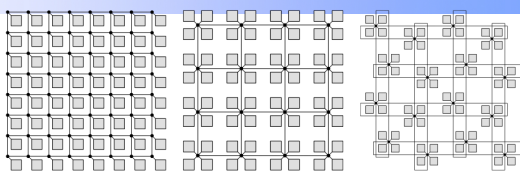


Silicon Photonics for Energy-Efficient Communication

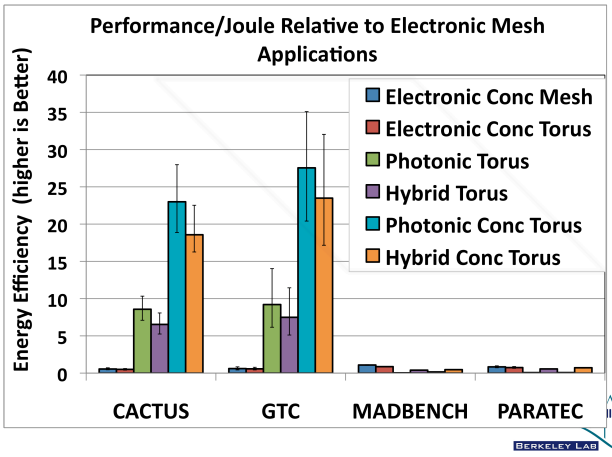


Silicon Photonic Ring Resonator

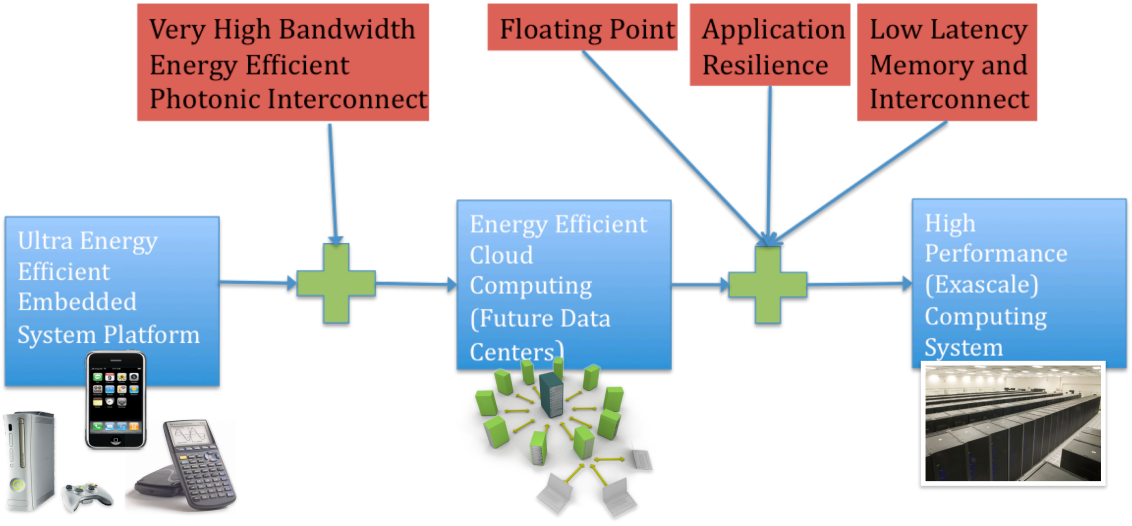

(a) Off state (b) On state


(a) Mesh (b) Concentrated Mesh (c) Concentrated Torus

- **Silicon photonics enables optics to be integrated with conventional CMOS**
- **Enables up to 27x improvement in communication energy efficiency!**



Technology Continuity for A Sustainable Hardware Ecosystem



Need building blocks for a compelling environment at all scales

Summary

- **Power is leading design constraint for future HPC**
  - **Future technology driven by handheld space**
  - **Notion of "commodity" moving on-chip**
- **Approach for Power Efficient HPC**
  - **Choose the science target first** *(climate in this case)*
  - **Design systems for applications** *(rather than the reverse)*
  - **Design hardware, software, scientific algorithms together using hardware emulation and auto-tuning**
  - ***This is the right way to design efficient HPC systems!***

Acknowledgements

Students

- Marghoob Mohiyuddin
- Shoaib Kamil
- Jens Krueger
- Kaushik Datta
- Kamesh Madurri
- Cy Chan

Staff

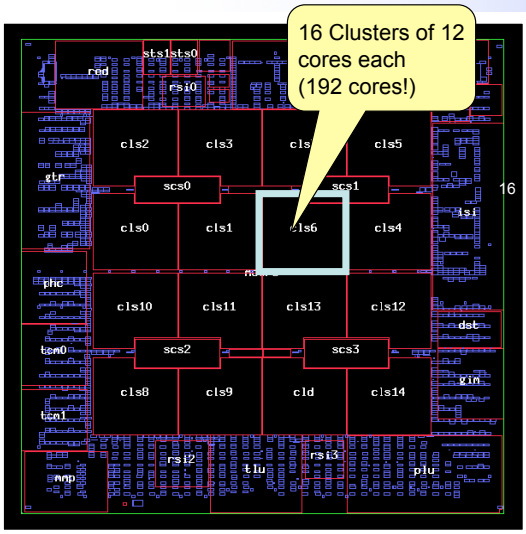
- David Donofrio
- Leonid Oliker
- Michael Wehner
- Tony Drummond
- Woo-Sun Yang
- Norman Miller
- Sam Williams

NERSC NATIONAL ENERGY RESEARCH SCIENTIFIC COMPUTING CENTER
Office of Science U.S. DEPARTMENT OF ENERGY
BERKELEY LAB

More Info

- Green Flash
  - http://www.lbl.gov/CS/html/greenflash.html
- NERSC System Architecture Group
  - http://www.nersc.gov/projects/SDSA
- The Berkeley View/Parlab
  - http://view.eecs.berkeley.edu
  - http://parlab.eecs.berkeley.edu/
- LBNL Future Technologies Group
  - http://crd.lbl.gov/ftg

Mitigating Cost of Verification and Defects:
Cisco CRS-1 Terabit Router



CISCO SYSTEMS

188+4 Xtensa general purpose processor cores per Silicon Packet Processor

Up to 400,000 processors per system

- (this is not just about HPC!!!)

Replaces ASIC using 188 GP cores!

Emulates ASIC at competitive power/performance

Better power/performance than FPGA!

New Definition for "Custom" in SoC