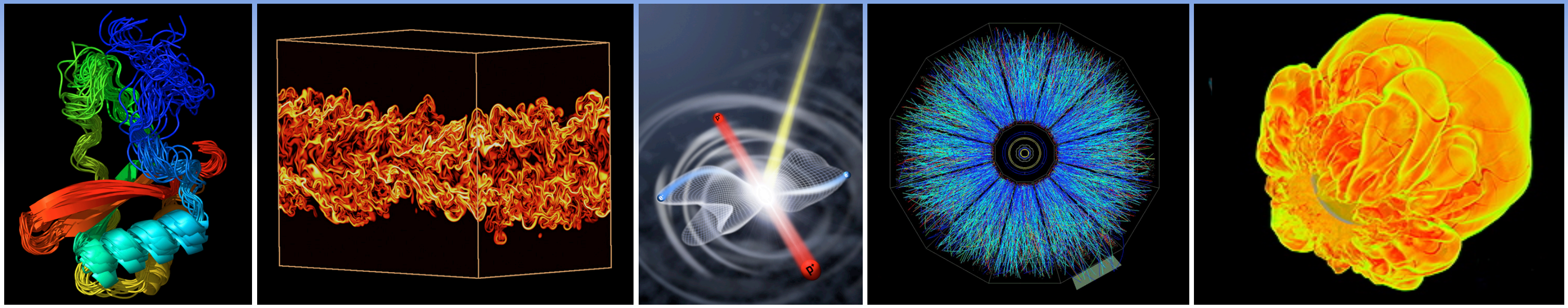


Future Hardware Challenges for Scientific Computing

John Shalf

National Energy Research Supercomputing Center
Lawrence Berkeley National Laboratory

1stNeuroinformatics Congress

Stockholm Sweden, September 7, 2008

Office of Science
U.S. DEPARTMENT OF ENERGY

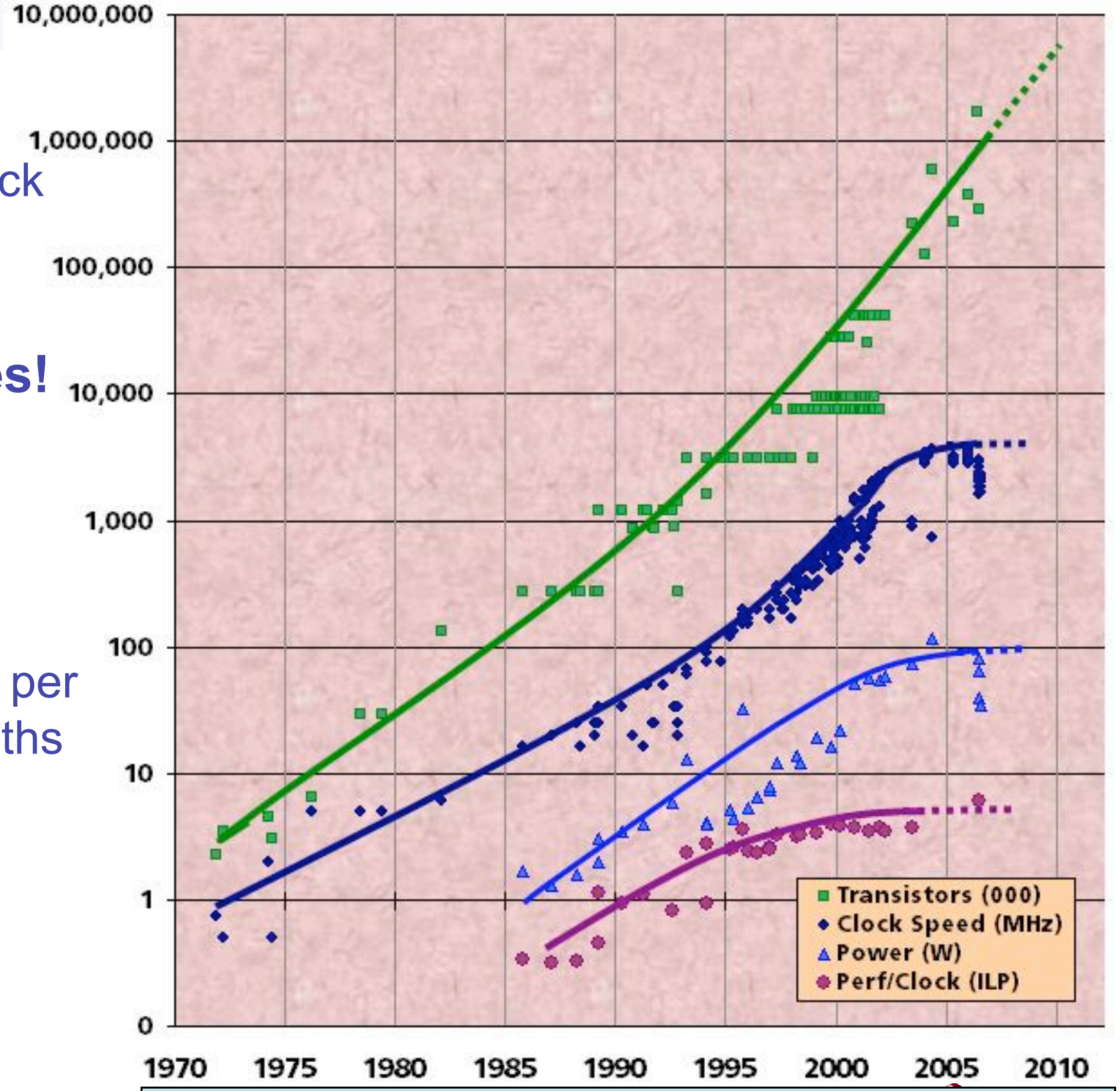
NERSC

BERKELEY LAB

Traditional Sources of Performance Improvement are Flat-Lining

- **New Constraints**
  - 15 years of *exponential* clock rate growth has ended

- **But Moore's Law continues!**
  - How do we use all of those transistors to keep performance increasing at historical rates?
  - Industry Response: #cores per chip doubles every 18 months *instead* of clock frequency!



1 Figure courtesy of Kunle Olukotun, Lance Hammond, Herb Sutter, and Burton Smith

Traditional Sources of Performance Improvement are Flat-Lining

- **New Constraints**
  - 15 years of *exponential* clock rate growth has ended
- **But Moore's Law continues!**
  - How do we use all of those transistors to keep performance increasing at historical rates?
  - Industry Response: #cores per chip doubles every 18 months *instead* of clock frequency!
  - *Is this a good idea, or is it completely brain-dead?*

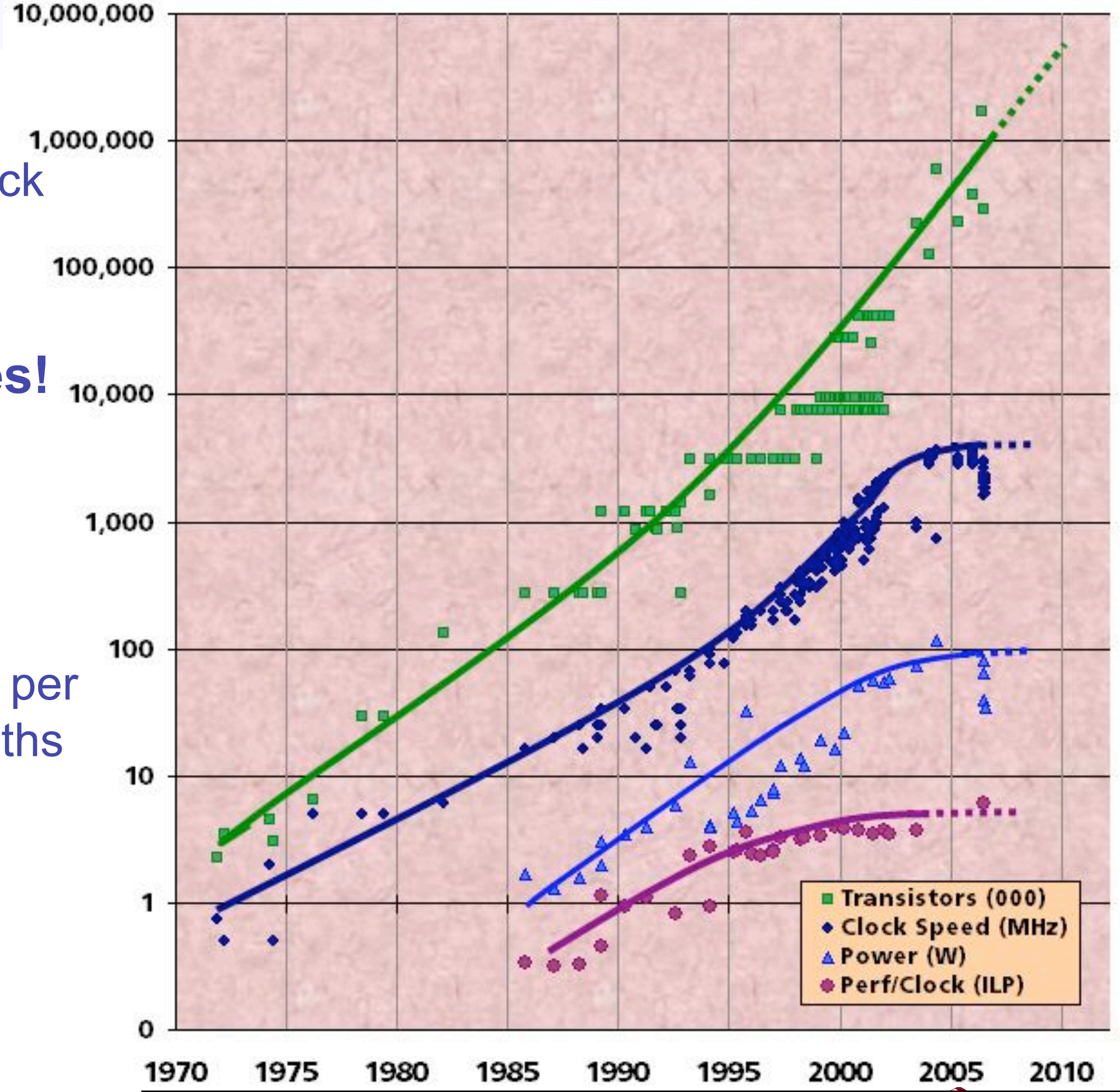


Figure courtesy of Kunle Olukotun, Lance Hammond, Herb Sutter, and Burton Smith

NERSC
NATIONAL ENERGY RESEARCH
SCIENTIFIC COMPUTING CENTER

Traditional Sources of Performance Improvement are Flat-Lining

- **New Constraints**
  - 15 years of *exponential* clock rate growth has ended
- **But Moore's Law continues!**
  - How do we use all of those transistors to keep performance increasing at historical rates?
  - Industry Response: #cores per chip doubles every 18 months *instead* of clock frequency!
  - *Is this a good idea, or is it completely brain-dead?*
  - *Has industry run out of ideas?*

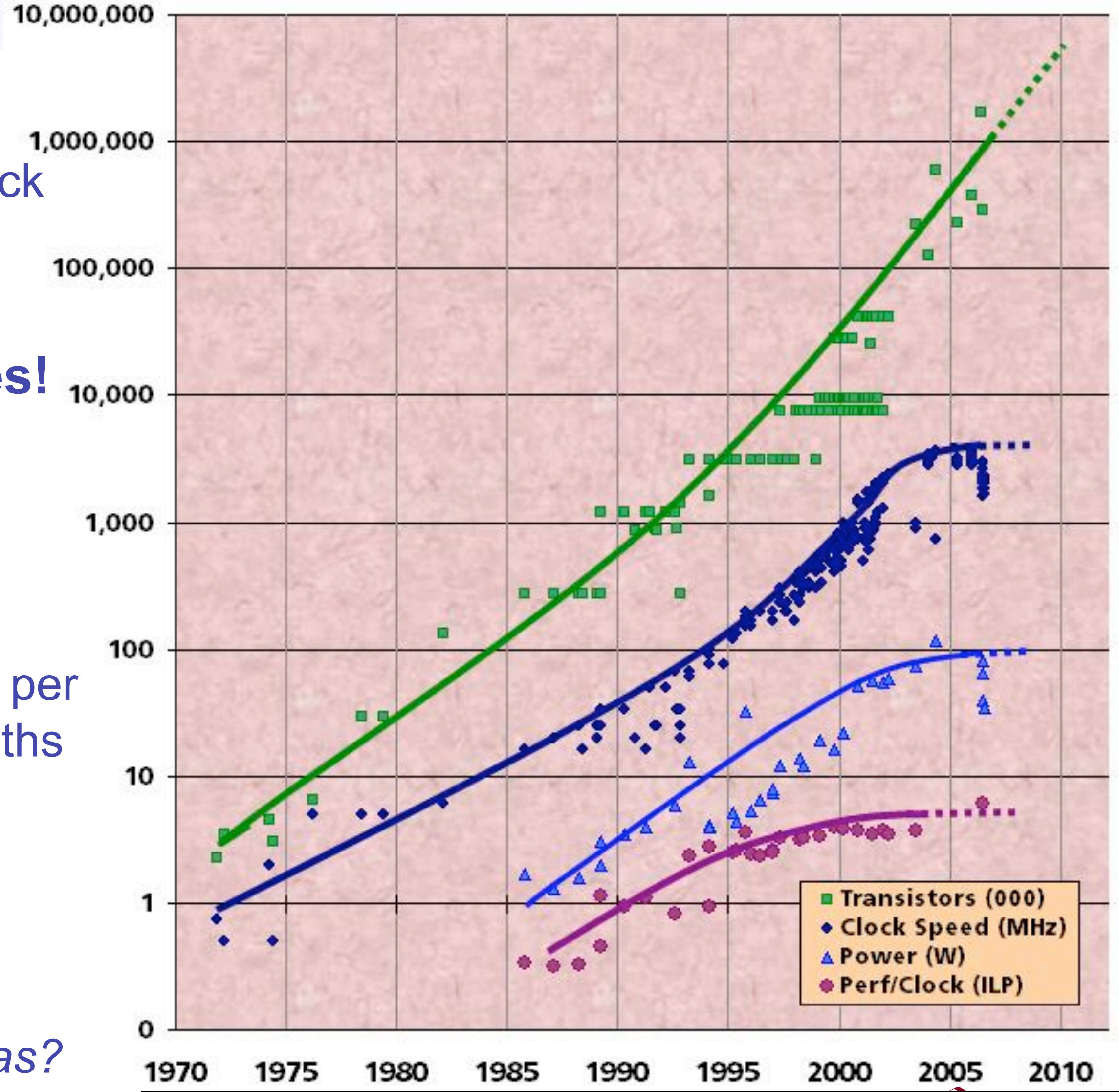


Figure courtesy of Kunle Olukotun, Lance Hammond, Herb Sutter, and Burton Smith

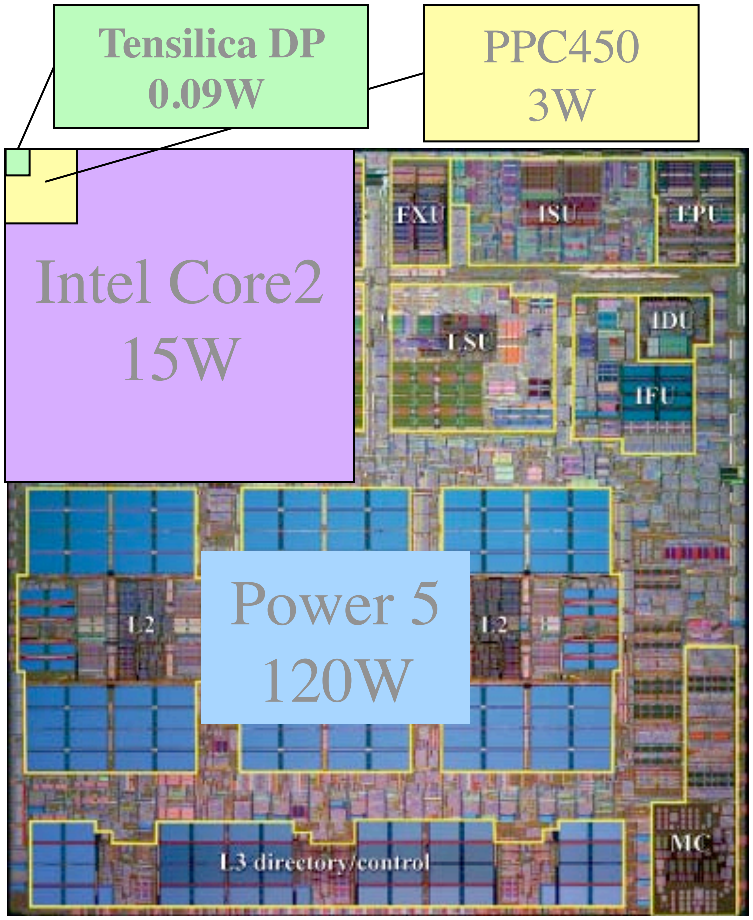
Office of Science
U.S. DEPARTMENT OF ENERGY

Hardware: What are the problems?

- **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

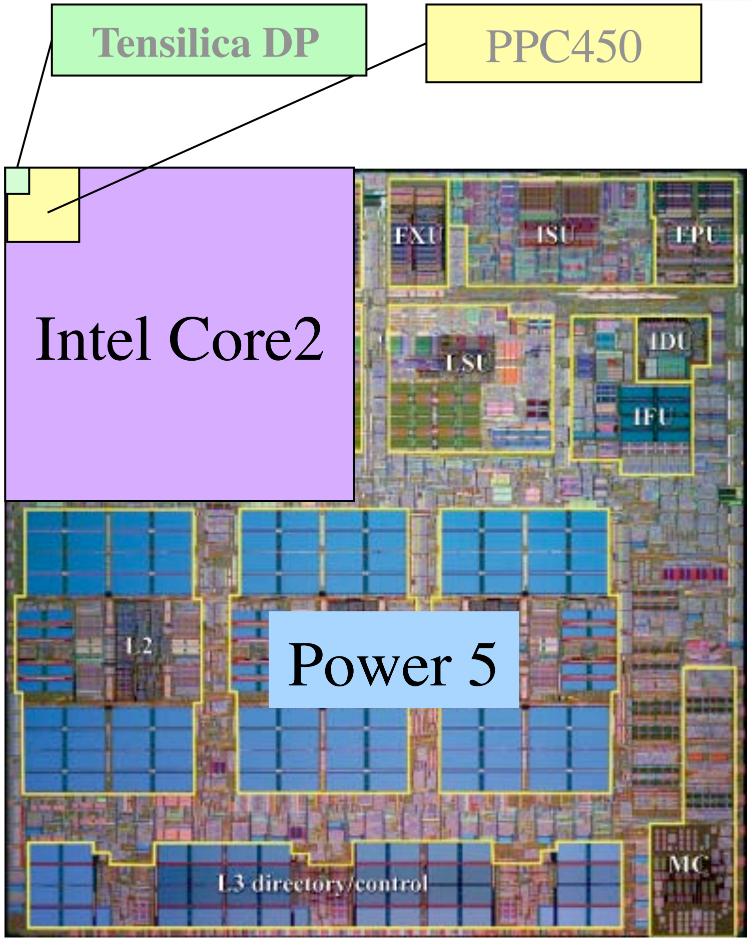
Elements of a Power-Efficient Processor Core Design



- Cubic power improvement with lower clock rate due to V^2F
- Slower clock rates reduce pipeline stages (fewer transistors for same functionality)
- Simpler in-order cores use less area (lower leakage) and reduce cost
- Tailor design to application reduce waste

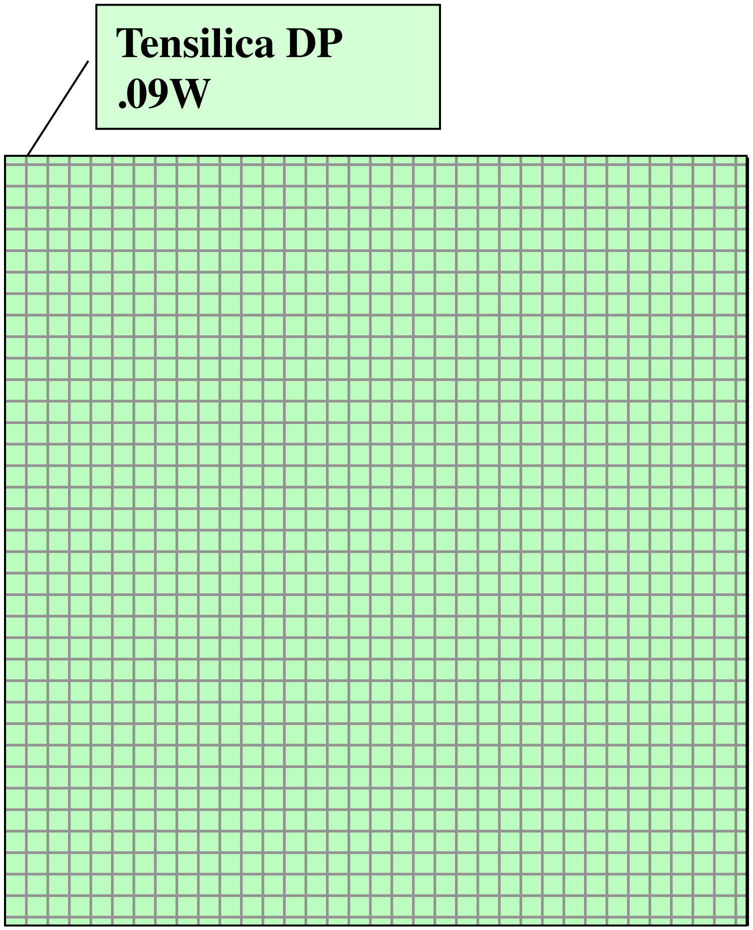
This is how consumer electronics devices such as iPhones and MP3 players are designed to maximize battery life and minimize cost

How Small is “Small”



- IBM Power5 (server)
  - 120W@1900MHz
  - Baseline
- Intel Core2 sc (laptop) :
  - 15W@1000MHz
  - *4x more FLOPs/watt than baseline*
- IBM PPC 450 (automobiles - BG/P)
  - 0.625W@800MHz
  - 90x more
- TensilicaXTensa(Moto Razor) :
  - 0.09W@600MHz
  - 400x more

How Small is “Small”



- IBM Power5 (server)
  - 120W@1900MHz
  - Baseline
- Intel Core2 sc (laptop) :
  - 15W@1000MHz
  - *4x more FLOPs/watt than baseline*
- IBM PPC 450 (automobiles - BG/P)
  - 0.625W@800MHz
  - 90x more
- TensilicaXTensa(Moto Razor) :
  - 0.09W@600MHz
  - 400x more

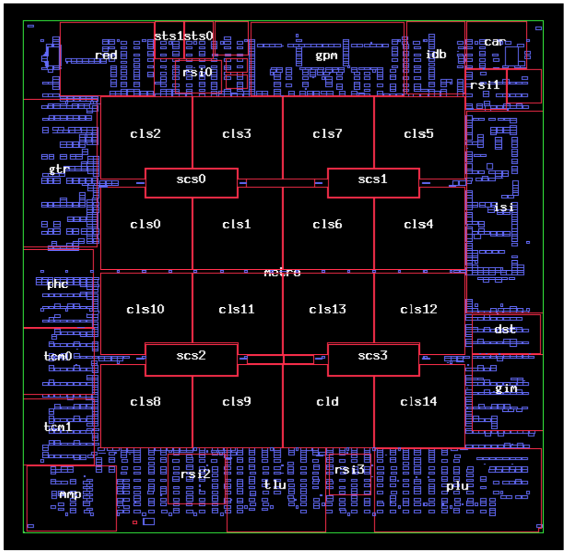
Even if each core operates at 1/3 to 1/10th efficiency of largest chip, you can pack 100s more cores onto a chip and consume 1/20 the power

Multicore vs. Manycore

- **Multicore: current trajectory**
  - Stay with current fastest core design
  - Replicate every 18 months (2, 4, 8 . . . Etc…)
  - Advantage: Do not alienate serial workload
  - Example: AMD X2 (2 core), Intel Core2 Duo (2 cores), Madison (2 cores), AMD Barcelona (4 cores), Intel Tigerton (4 cores)

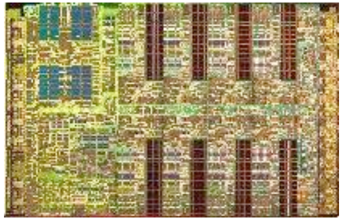
- **Manycore: converging in this direction**
  - Simplify cores (shorter pipelines, lower clock frequencies, in-order processing)
  - Start at 100s of cores and replicate every 18 months
  - Advantage: easier verification, defect tolerance, highest compute/surface-area, best power efficiency
  - Examples: Cell SPE (8 cores), Nvidia G80 (128 cores), Intel Polaris (80 cores), Cisco/Tensilica Metro (188 cores)

- **Convergence: Ultimately toward Manycore**
  - Manycore: *if we can figure out how to program it!*
  - Hedge: Heterogenous Multicore (still must run PPT)



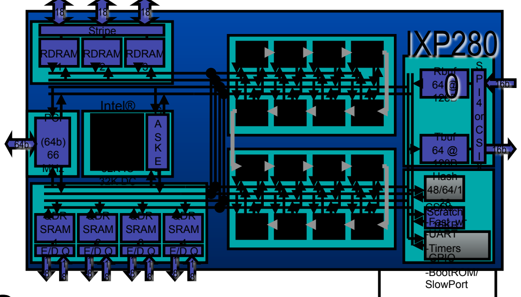
Convergence of Platforms

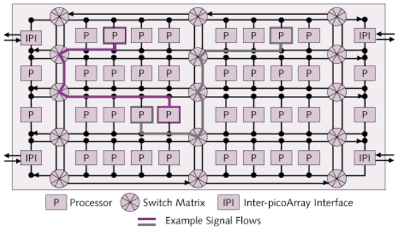
– **Multiple parallel general-purpose processors (GPPs)**
– **Multiple application-specific processors (ASPs)**



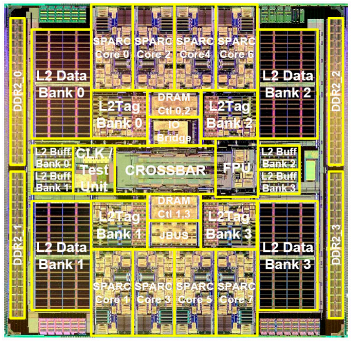
IBM Cell
1 GPP (2 threads)
8 ASPs

Intel Network Processor
1 GPP Core
16 ASPs (128 threads)

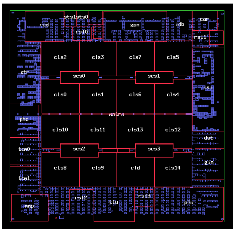




Picochip DSP
1 GPP core
248 ASPs

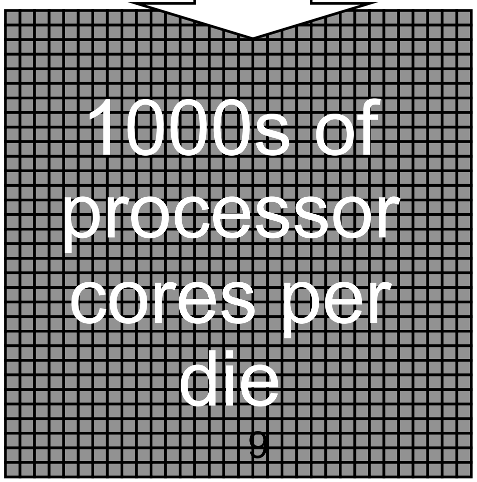


Sun Niagara
8 GPP cores (32 threads)



Cisco CRS-1
188 Tensilica GPPs

Intel 4004 (1971):
4-bit processor,
2312 transistors,
~100 KIPS,
10 micron PMOS,
11 mm^2 chip



1000s of processor cores per die

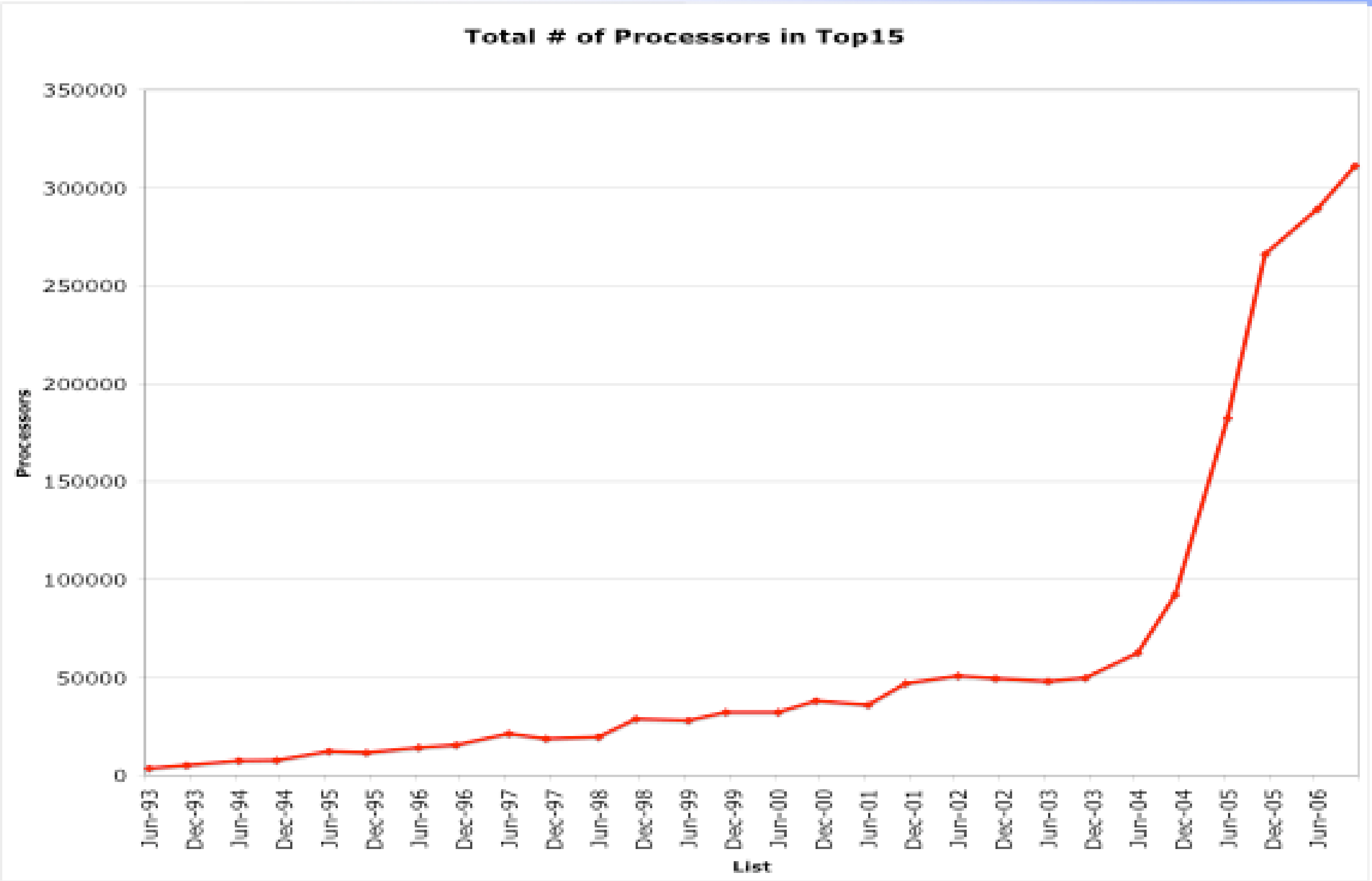
"The Processor is the new Transistor"

[Chris Rowen]

Ramifications of Massive Parallelism

(fear and loathing)

The Future of HPC System Concurrency



Must ride exponential wave of increasing concurrency for forseeable future! You will hit 1M cores sooner than you think!

Ramifications of Massive Parallelism

- **Programming Model**
- **Algorithms**
- **Compiler Technology**
- **Software Engineering**

- **Green Flash:** *Design a machine to fit the application!*

Programming Model

targeting million-way parallelism leads to uncertainty regarding future programming model

Multicore is NOT a Familiar Programming Target

- **What about Message Passing on a chip?**
  - MPI buffers &datastructures growing O(N) to O(N^2) problem for limited memory
  - Redundant use of memory for shared variables and program image
  - *Flat view of parallelism doesn't make sense given hierarchical nature of multicore systems!!! (worry about depth of parallelism rather than breadth)*

- **What about SMP on a chip?**
  - Hybrid Model (MPI+OpenMP) : *Long and mostly unsuccessful history*
  - But it is NOT an SMP on a chip
    - 10-100x higher bandwidth on chip
    - 10-100x lower latency on chip
  - SMP model ignores potential for much tighter coupling of cores
  - *Failure to exploit hierarchical machine architecture will drastically inhibit ability to efficiently exploit concurrency! (requires code structure changes)*

- **Entering transition period for programming models**

Looking Beyond the SMP

(focus on parallelism Depth instead of Breadth)

#sockets relatively constant (# cores is doubling)

What can we do ON-CHIP that is different than off-chip?

- **Cache Coherency:** *necessary but not sufficient (and not efficient for manycore!)*
  - Fine-grained language elements difficult to build on top of CC protocol
  - Hardware Support for Fine-grained hardware synchronization
- **Message Queues:** direct hardware support for messages
- **Transactions:** Protect against incorrect reasoning about concurrency
  - If there is an inter-loop dependency, transactions “roll back” and run slower (but still get the same result as serial case)
  - Allows more aggressive use of auto-parallelization technology
  - Still many “semantic” issues to work out (this is not a panacea)

Compiler Technology

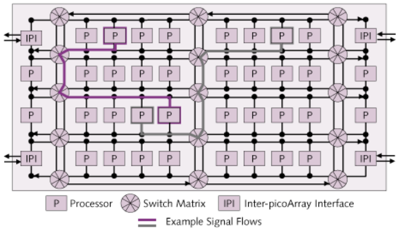
Faced with increased architectural diversity

Auto-parallelizing compilers are not going to magically solve our problems

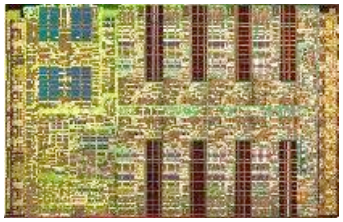
Performance Portability

- **Diverse set of architectural options == Daunting tuning requirements**
  - **What is a compiler to do?**
- **Performance portability was bad enough**
  - Diversity makes performance portability tough
  - In many cases, basic portability is lost
  - Need new approaches such as multi-target languages, auto-tuning and/or code generators

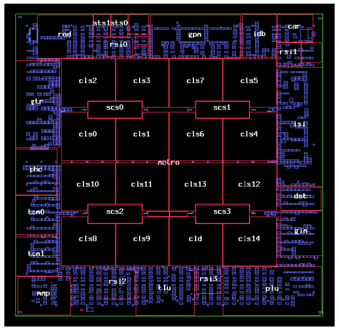
Picochip DSP
1 GPP core
248 ASPs

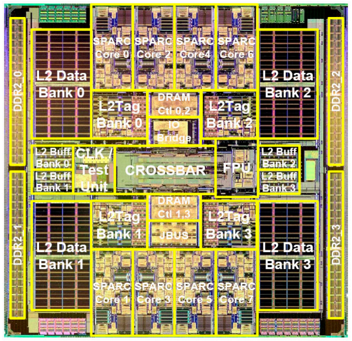


STI Cell
8 ASPs, 1GPP

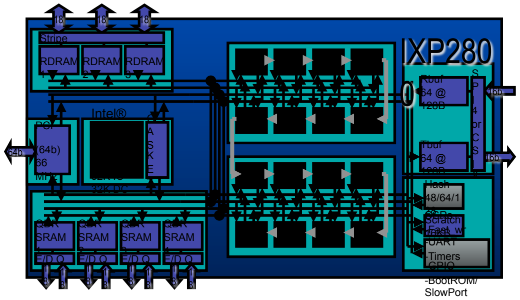


Cisco CRS-1
188 Tensilica GPPs





Sun Niagara
8 GPP cores (32 threads)

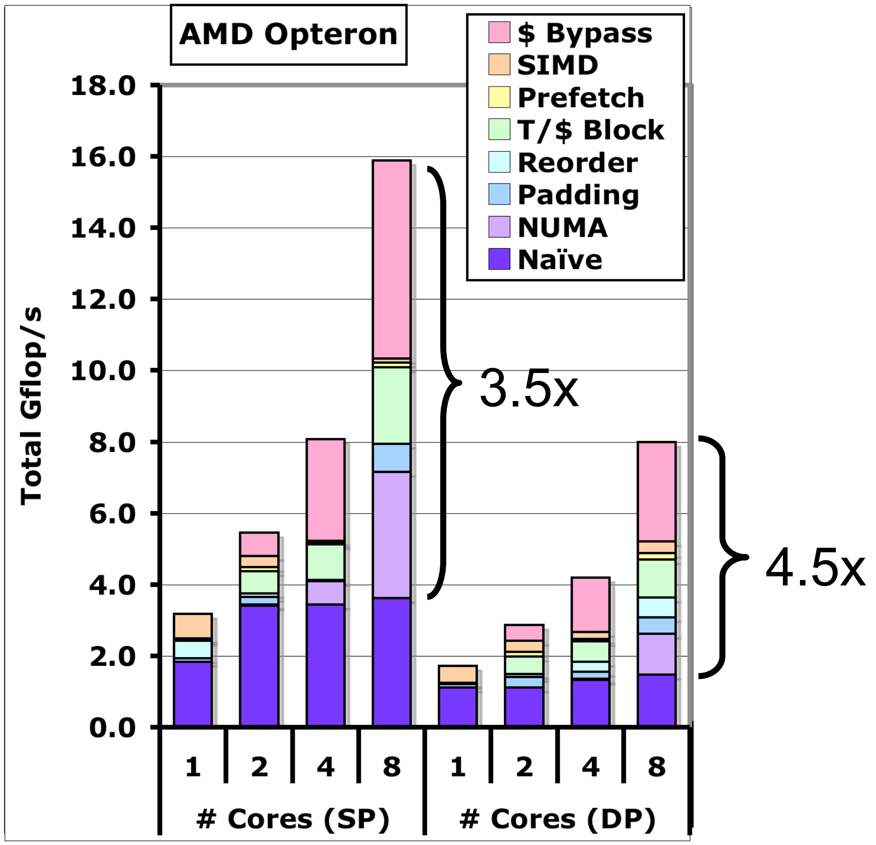


Intel Network Processor
1 GPP Core
16 ASPs (128 threads)

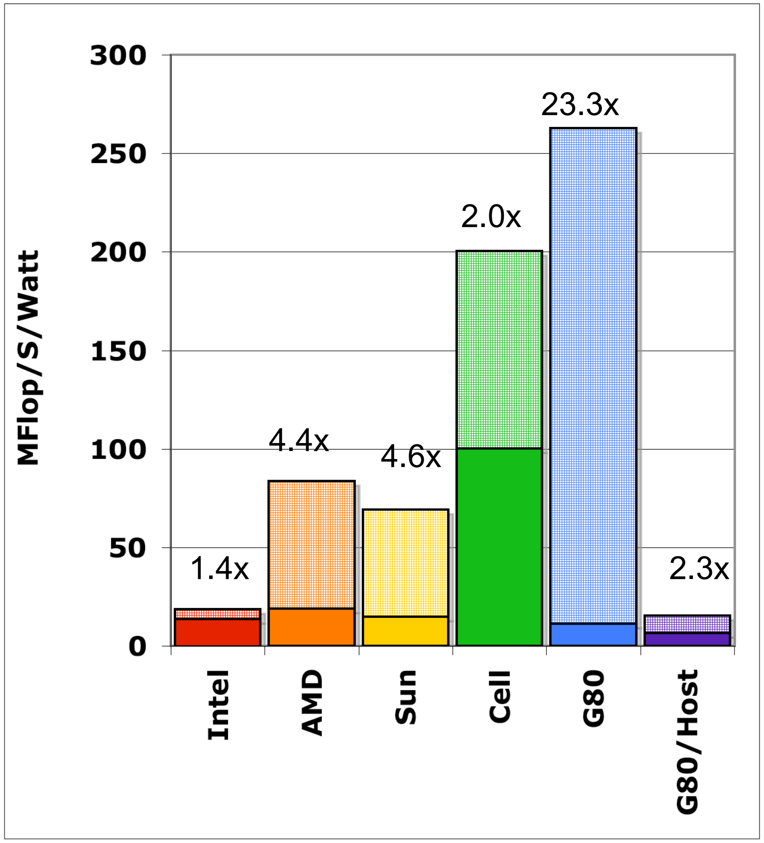
Multiprocessor Performance

(auto-tuned stencil kernel)

Performance Scaling

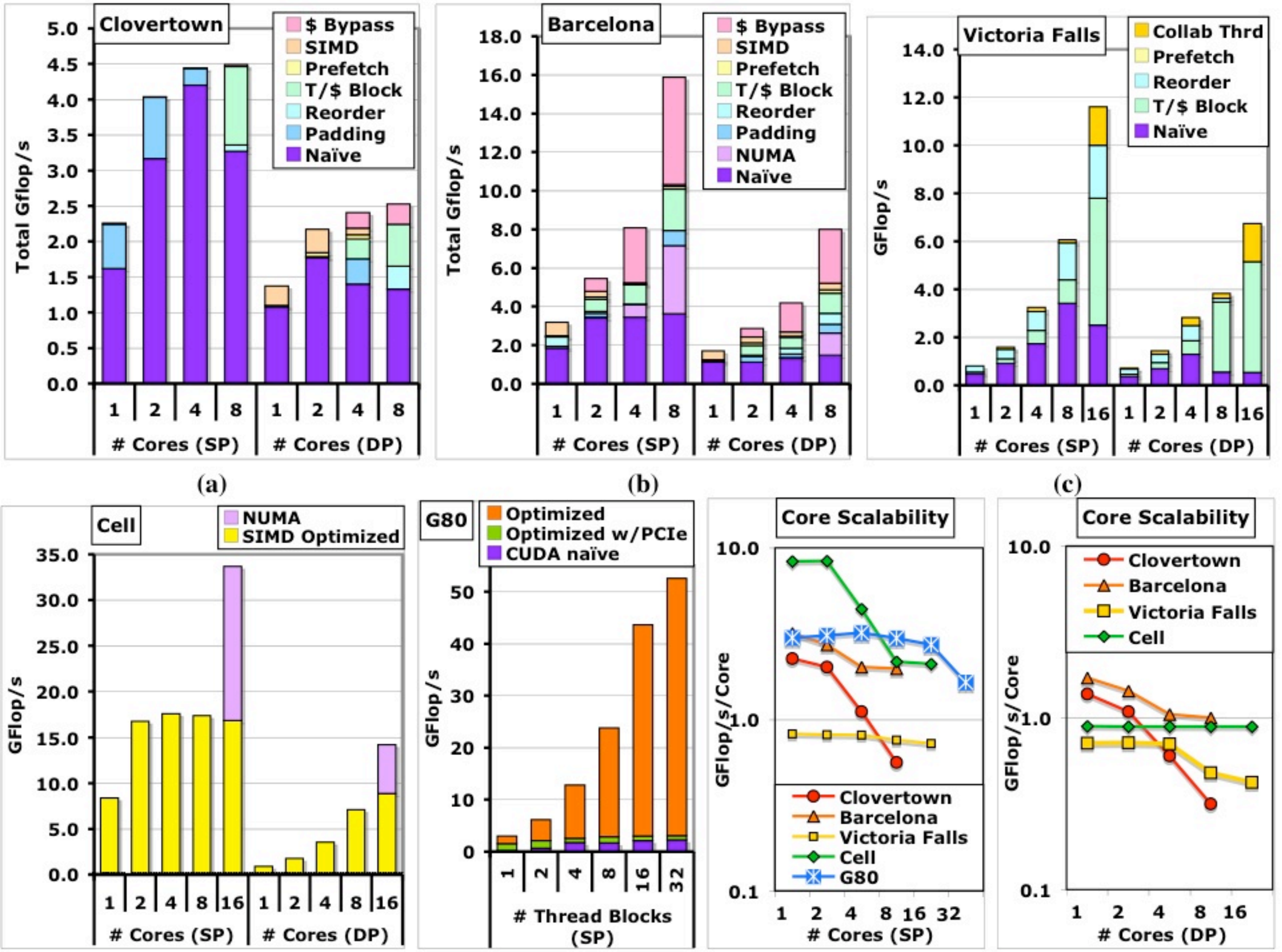


Power Efficiency



Compilers with maximum optimization are not delivering scalable performance

Performance Portability



(a) (b) (c)

(d) (e) (f) (g)

Algorithms

targeting million-way parallelism changes the selection of algorithms

Technology Trends are Affecting Algorithm Requirements

- **Parallel computing has thrived on weak-scaling for past 15 years**
- **Flat CPU performance increases emphasis on strong-scaling**
- **Algorithm Requirements will change accordingly**
  – Concurrency will increase proportional to system scale (every 18 months)
  – Timestepping algorithms will be increasingly driven towards implict or semi-implicit stepping schemes
  – Multiphysics/multiscale problems increasingly rely on spatially adaptive approaches such as Berger-Oliger AMR
  – Strong scaling will push applications towards smaller messages sizes – requiring lighter-weight messaging (weak point of MPI)

Where to Find 12 Orders in 10 years? (for simulations of ITER)

David Keyes, Columbia U.

Hardware: 3

- 1.5 orders: increased processor speed and efficiency
- 1.5 orders: increased concurrency

Software: 9

- 1 order: higher-order discretizations
  - Same accuracy can be achieved with many fewer elements
- 1 order: flux-surface following gridding
  - Less resolution required along than across field lines
- 4 orders: adaptive gridding
  - Zones requiring refinement are <1% of ITER volume and resolution requirements away from them are ~10^2 less severe
- 3 orders: implicit solvers
  - Mode growth time 9 orders longer than Alfven-limited CFL

Regarding Code & Model Complexity

Application Code Complexity

- **Application Complexity has Grown**
  - Big Science is a multi-disciplinary, multi-institutional, multi-national efforts!
  - Looking more like science on atom-smashers
  - Rapidly outstripping our ability to Verify & Validate our results against experiments!

- **Advanced Parallel Languages Necessary, but NOT Sufficient!**
  - Need higher-level organizing constructs for teams of programmers and scientists

Community Codes & Frameworks

- **Complexity of hardware is daunting**
- **Complexity of the model is even more daunting**
- **Both require adoption of more formalized practices of software engineering (frameworks, etc…)**
  - **Idiom for parallelism:** Externalize from specification of the mathematical operators
  - **Modular code:** unit-testing, algorithm comparisons
  - **Frameworks:** A social contract between computer scientists and model developers (no CS magic here)
  - **Verification & Validation:** Supported by modularity and standardization of software design practices

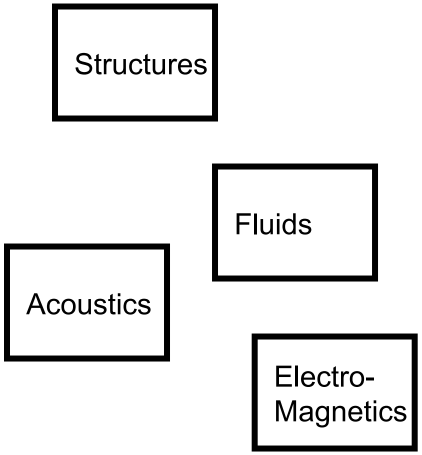
Community Codes & Frameworks

(hiding complexity using good SW engineering)

- **Community-grown frameworks (eg. Chombo, Cactus, SIERRA, UPIC, etc…)**
  - Clearly separate roles and responsibilities of expert programmers from that of the domain experts/scientist/users (productivity vs. performance layer)
  - Define *social* contract between expert programmers and domain scientists
  - Enforce and facilitate SW engineering style/discipline to ensure correctness
  - Hides complex domain-specific parallel abstractions from scientist/users to enable performance
  - Allow scientists/users to code nominally serial plug-ins that is invoked by a parallel "driver"
  - Modularity enables efficient UNIT TESTING of components for V&V
- **Properties of the "plug-ins" for successful frameworks (CSE07)**
  - Relinquish control of main(): framework decides when to invoke module!
  - Module must be stateless (so it can be invoked in any order)
  - Module only operates on the data it is given (well understood side-effects)

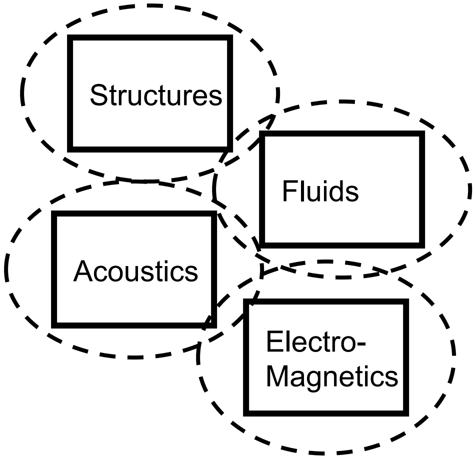
Framework Component Interoperability (from DARPA frameworks workshop)

Minimal Component Interoperability:



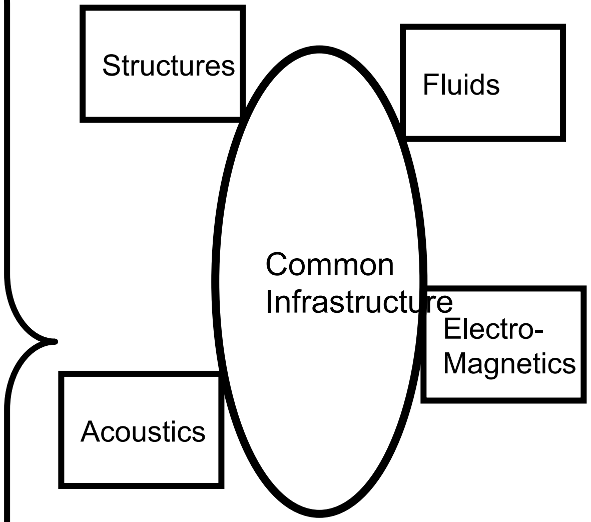
- Physics models are completely uncoupled.
- May exchange static datasets through flat files.

Shallow Component Interoperability:

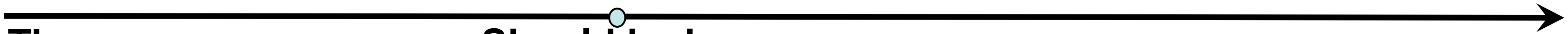


- Physics models are loosely coupled.
- Data management and parallelism is independent in each module.
- Exchange common data events via wrappers (web services, etc.).

Deep Component Interoperability:



- Physics models are tightly coupled.
- Data exchange across shared service infrastructure.



Time

Should be here

Green Flash:
Design a machine to fit the problem

Green Flash Overview

- **Research effort: study feasibility and share insight w/community**

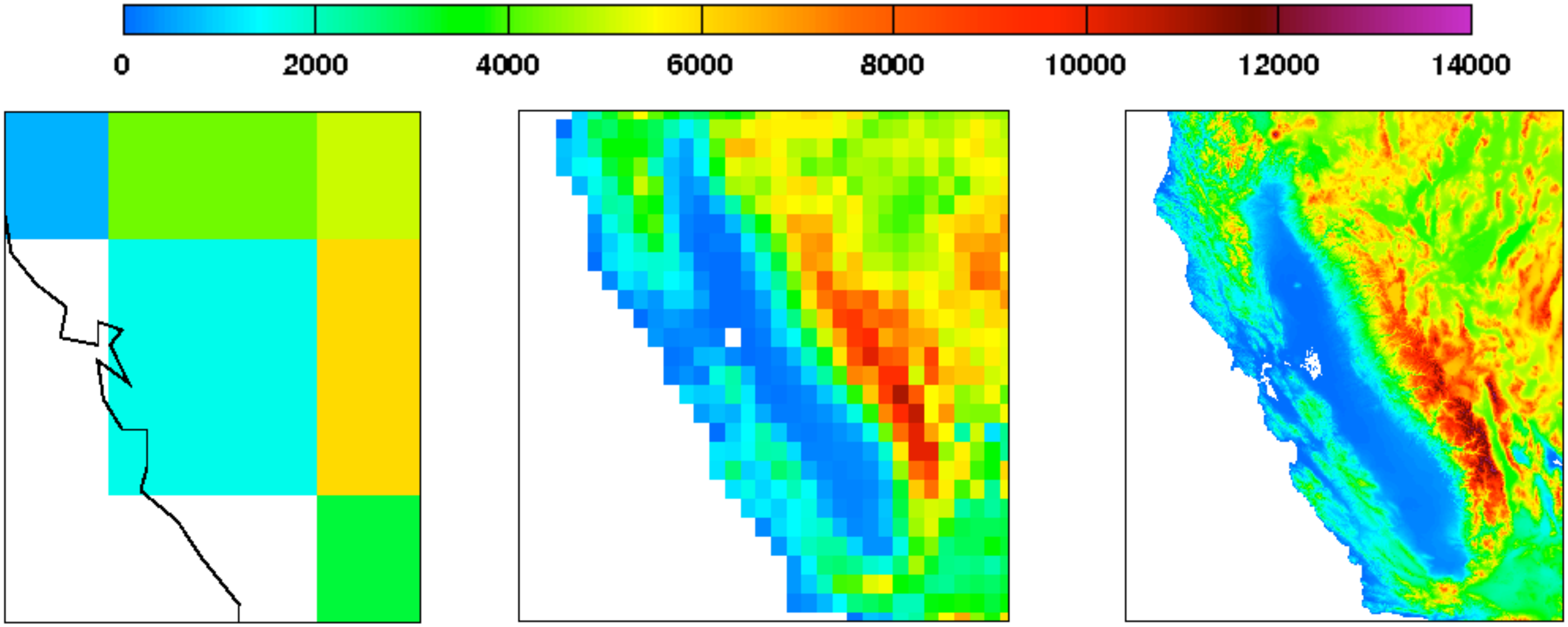
- **Elements of the approach**
  - **Choose the science target first** *(climate or neuroinformatics)*
  - **Design systems for applications** *(rather than the reverse)*
  - **Design hardware, software, scientific algorithms together using hardware emulation and auto-tuning**

- **What is NEW about this approach**
  - **Leverage commodity processes used to design power efficient embedded devices (redirect the tools to benefit scientific computing!)**
  - **Auto-tuning to automate mapping of algorithm to complex hardware**
  - **RAMP: Fast hardware-accelerated emulation of new chip designs**

Applicable to broad range of scientific computing applications

Global Cloud System Resolving Climate Models

Surface Altitude (feet)



200km
Typical resolution of
IPCC AR4 models

25km
Upper limit of climate models
with cloud parameterizations

1km
Cloud system resolving models

- Direct simulation of cloud systems replacing statistical parameterization.
  - This approach recently was called for by the 1st WMO Modeling Summit.

1km-Scale Global Climate Model Requirements

Simulate climate 1000x faster than real time

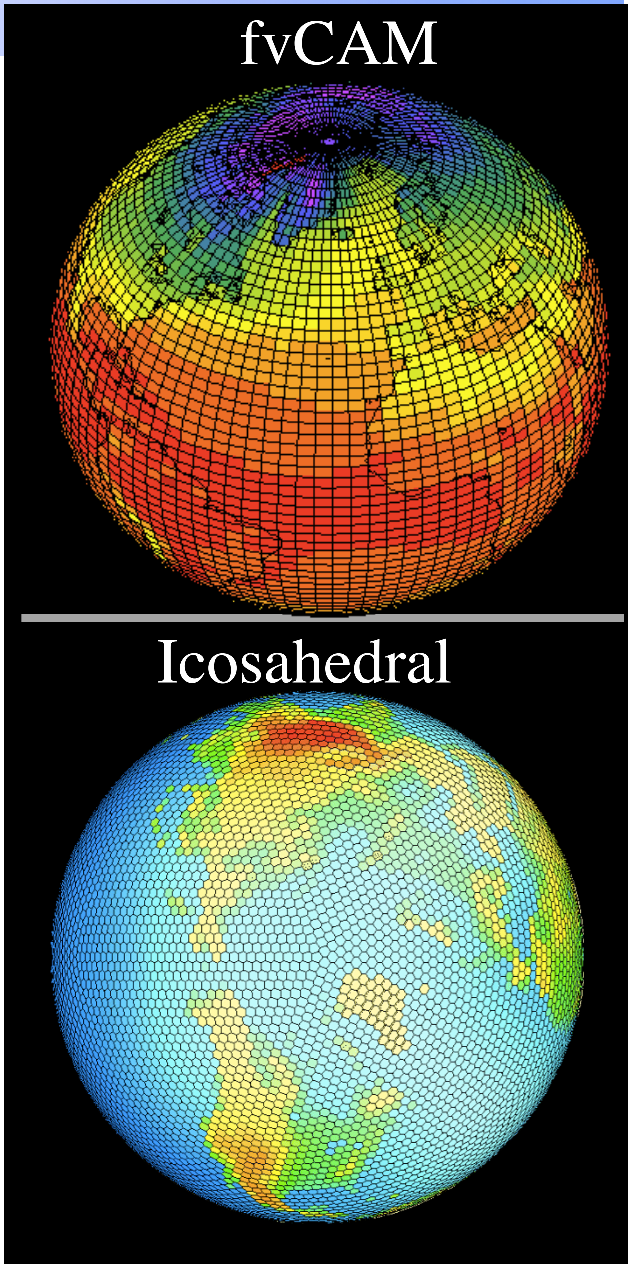
10 Petaflops sustained per simulation
(~200 Pflops peak)

10-100 simulations (~20 Exaflops peak)

Truly exascale!

Some specs:

- Advanced dynamics algorithms: icosahedral, cubed sphere, reduced mesh, etc.
- ~20 billion cells → Massive parallelism
- 100 Terabytes of Memory
- Can be decomposed into ~20 million total subdomains



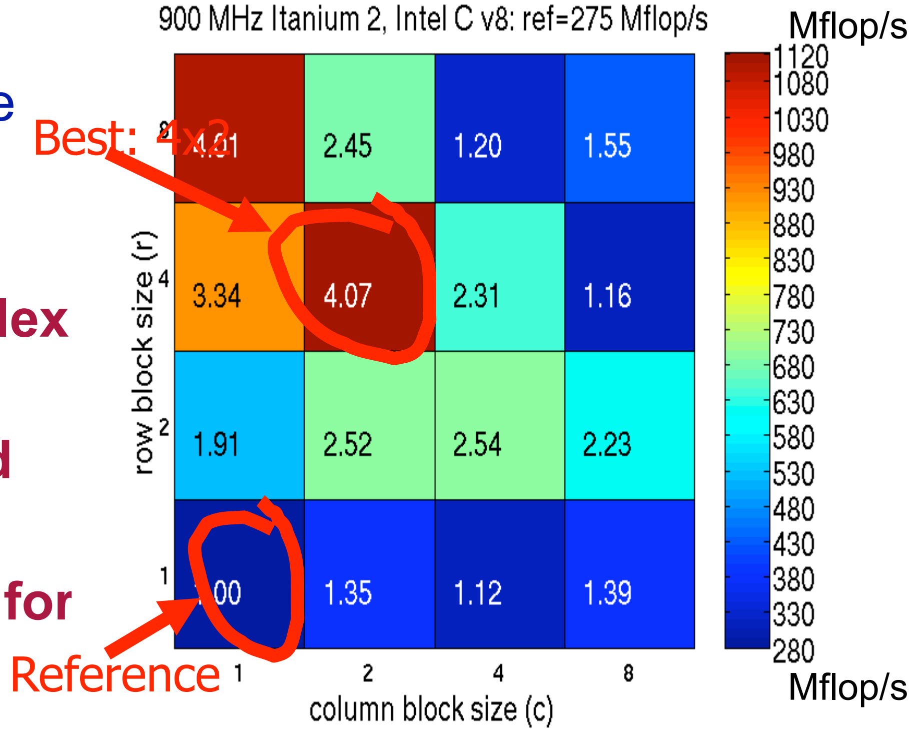
Requires New Algorithmic Approach to Achieve 20M-way concurrency

- Collaborating with CSU on Icosahedral Model

Auto-tuning

- **Problem: want performance on diverse architectures**
  - Code is non-portable
  - Optimizations are architecture-specific
  - To labor-intensive to hand-optimize for each system
- **A Solution: Auto-tuning**
  - **automate search across a complex optimization space**
  - **Achieve performance far beyond current compilers**
  - **achieve performance portability for diverse architectures!**

For finite element problem (BCSR)
[Im, Yelick, Vuduc, 2005]

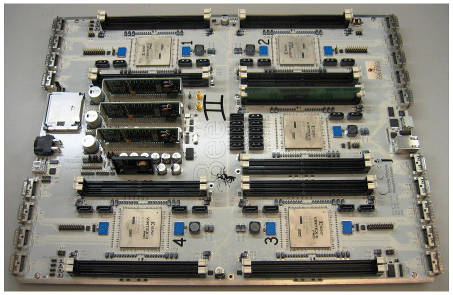


r \ c	1	2	4	8
8	4.01	2.45	1.20	1.55
4	3.34	4.07	2.31	1.16
2	1.91	2.52	2.54	2.23
1	1.00	1.35	1.12	1.39

Advanced Hardware Simulation (RAMP)

- Research Accelerator for Multi-Processors (RAMP)
  - Utilize FGPA boards to emulate large-scale multicore systems
  - Simulate hardware before it is built
  - Break slow feedback loop for system designs
  - Allows fast performance validation
  - Enables tightly coupled hardware/software/science co-design *(not possible using conventional approach)*

- Technology partners:
  - UC Berkeley: John Wawrzynek, David Patterson, Jim Demmel, Krste Asanovic, Dan Burke
  - Stanford University / Rambus Inc.: Mark Horowitz
  - Tensilica Inc.: Chris Rowen



Leveraging Commodity Hardware Design Flow

- **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

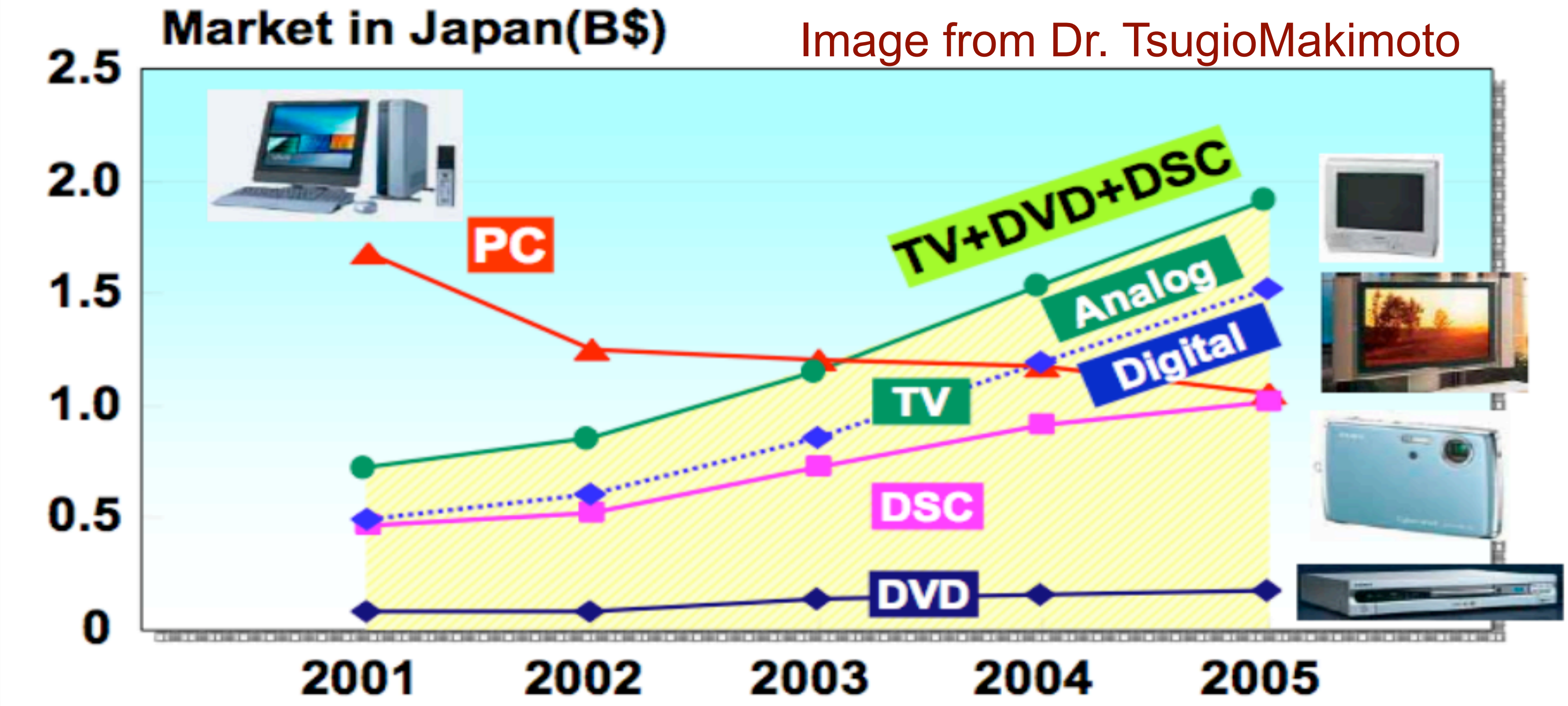
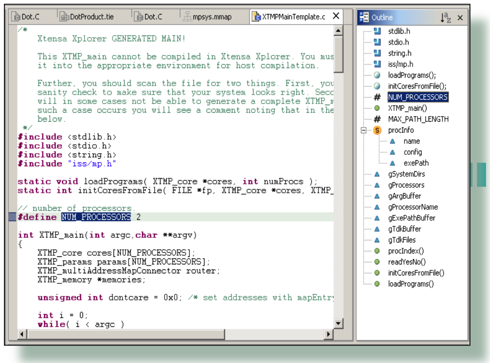


Image from Dr. TsugioMakimoto

Embedded Design Automation

(Example from Existing Tensilica Design Flow)

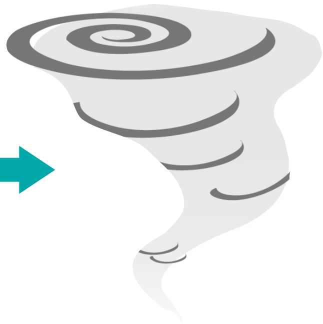


```
/*
    Xtensa Xplorer GENERATED MAIN!

    This XTMP_main cannot be compiled in Xtensa Xplorer. You mus
    it into the appropriate environment for host compilation.

    Further, you should scan the file for two things. First, you
    sanity check to make sure that your system looks right. Secc
    will in some cases not be able to generate a complete XTMP_m
    such a case occurs you will see a comment noting that in the
    below.
 */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "iss/mp.h"

static void loadPrograms( XTMP_core *cores, int numProcs );
static int initCoresFromFile( FILE *fp, XTMP_core *cores, XTMP_

// number of processors.
#define NUM_PROCESSORS 2

int XTMP_main(int argc,char **argv)
{
    XTMP_core cores[NUM_PROCESSORS];
    XTMP_params params[NUM_PROCESSORS];
    XTMP_multiAddressMapConnector router;
    XTMP_memory *memories;

    unsigned int dontcare = 0x0; /* set addresses with mapEntry

    int i = 0;
    while( i < argc )
```

```
stdlib.h
stdio.h
string.h
iss/mp.h
loadPrograms();
initCoresFromFile();
NUM_PROCESSORS
XTMP_main()
MAX_PATH_LENGTH
procInfo
    name
    config
    exePath
gSystemDirs
gProcessors
gArgBuffer
gProcessorName
gExePathBuffer
gTdkBuffer
gTdkFiles
procIndex()
readYesNo()
initCoresFromFile()
loadPrograms()
```

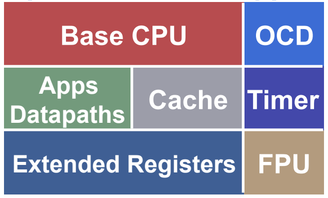
Processor configuration

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

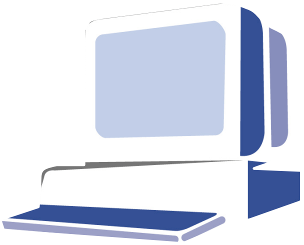


Processor Generator (*Tensilica*)

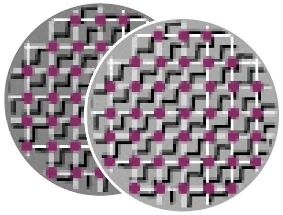
Application-optimized processor implementation (RTL/Verilog)

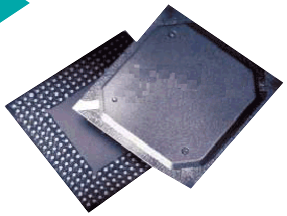


Base CPU | OCD
Apps Datapaths | Cache | Timer
Extended Registers | FPU



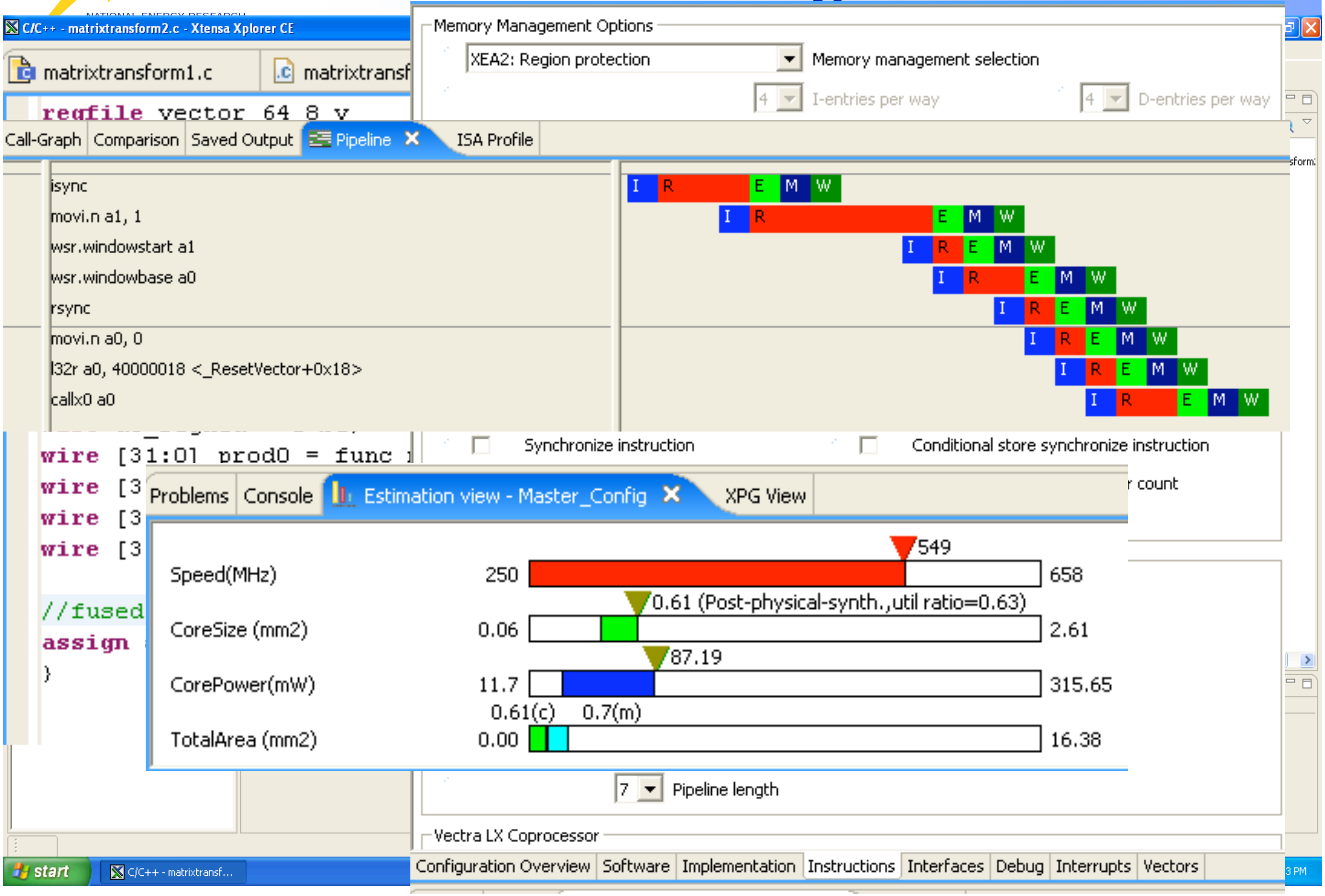
Tailored SW Tools: Compiler, debugger, simulators, Linux, other OS Ports *(Automatically generated together with the Core)*





Build with any process in any fab

Embedded Design Toolchain



C/C++ - matrixtransform2.c - Xtensa Xplorer CE

matrixtransform1.c | matrixtransf

```
regfile vector 64 8 v
wire [31:0] prod0 = func_i
wire [3
wire [3
wire [3

//fused
assign
}
```

Call-Graph | Comparison | Saved Output | Pipeline | ISA Profile

Instruction	Pipeline
isync	I R E M W
movi.n a1, 1	I R E M W
wsr.windowstart a1	I R E M W
wsr.windowbase a0	I R E M W
rsync	I R E M W
movi.n a0, 0	I R E M W
l32r a0, 40000018 <_ResetVector+0x18>	I R E M W
callx0 a0	I R E M W

Memory Management Options

XEA2: Region protection ▾ Memory management selection

4 ▾ I-entries per way 4 ▾ D-entries per way

□ Synchronize instruction □ Conditional store synchronize instruction

r count

Problems | Console | Estimation view - Master_Config | XPG View

	Min	Value	Max
Speed(MHz)	250	549	658
CoreSize (mm2)	0.06	0.61 (Post-physical-synth.,util ratio=0.63)	2.61
CorePower(mW)	11.7	87.19	315.65
TotalArea (mm2)	0.00	0.61(c) 0.7(m)	16.38

7 ▾ Pipeline length

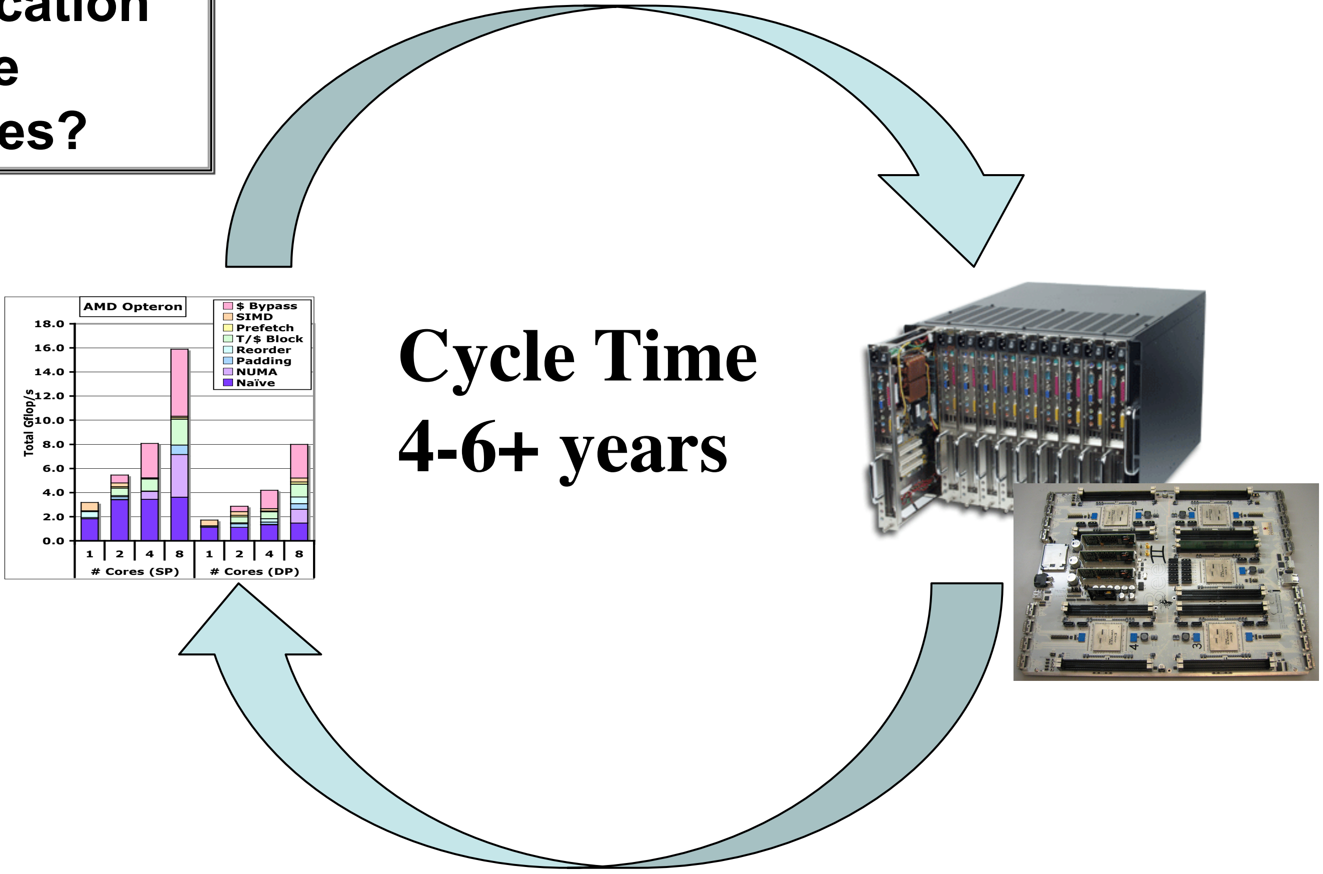
Vectra LX Coprocessor

Configuration Overview | Software | Implementation | Instructions | Interfaces | Debug | Interrupts | Vectors

start | C/C++ - matrixtransf...

Traditional New Architecture Hardware/Software Design

How long does it take for a full scale application to influence architectures?



Design New System
(2 year concept phase)

Cycle Time
4-6+ years

Build Hardware
(2 years)

Port Application

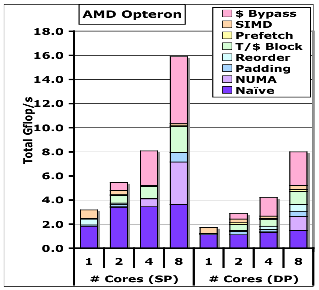
Tune Software
(2 years)

Proposed New Architecture Hardware/Software Co-Design

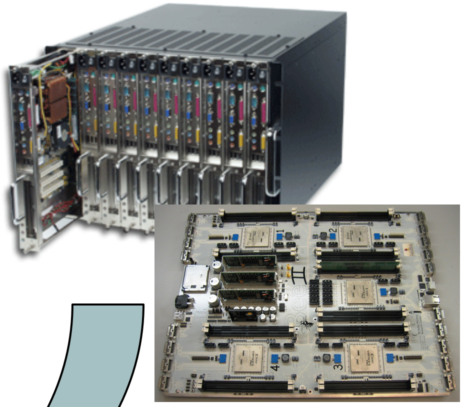
How long does it take for a full scale application to influence architectures?

Synthesize SoC (hours)

Autotune
Software
(Hours)



Cycle Time
1-2 days



Emulate
Hardware
(RAMP)
(hours)

Build application

Strawman System Design

We examined three different approaches (in 2008 technology)

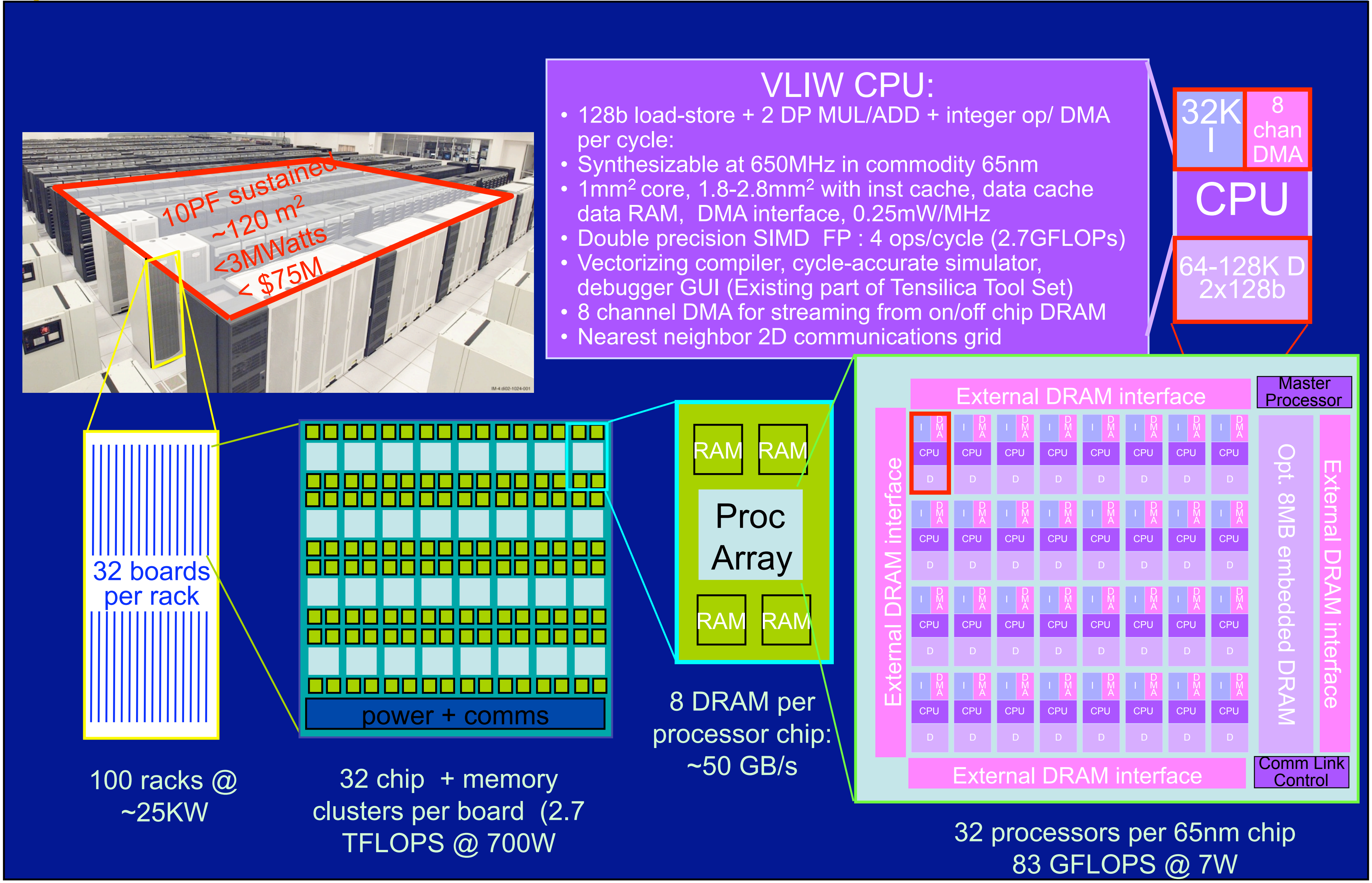
Computation .015°X.02°X100L: 10 PFlops sustained, ~200 PFlops peak

- **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) 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

Climate System Design Concept

Strawman Design Study



VLIW CPU:

- 128b load-store + 2 DP MUL/ADD + integer op/ DMA per cycle:
- Synthesizable at 650MHz in commodity 65nm
- 1mm^2 core, 1.8-2.8mm^2 with inst cache, data cache data RAM, DMA interface, 0.25mW/MHz
- Double precision SIMD FP : 4 ops/cycle (2.7GFLOPs)
- Vectorizing compiler, cycle-accurate simulator, debugger GUI (Existing part of Tensilica Tool Set)
- 8 channel DMA for streaming from on/off chip DRAM
- Nearest neighbor 2D communications grid

100 racks @ ~25KW

32 chip + memory clusters per board (2.7 TFLOPS @ 700W

8 DRAM per processor chip: ~50 GB/s

32 processors per 65nm chip 83 GFLOPS @ 7W

Summary

- Choose the science target first
- Design the supercomputer around application needs
- Design hardware, software, scientific algorithms together using hardware emulation and auto-tuning

- This approach is “fully programmable” and uses commodity design tools! (its not the same as full-custom design)

Could We Do this for Neuroinformatics Applications?

- **Problem contains massive amount of innate parallelism**
  - 10^{11} neurons, 10^{15} synaptic connections?
- **Add instructions for application**
  - Already have fast trapezoidal integration
  - Direct hardware support for sending events (synaptic connections) to neighbor processors
- **This is a fully programmable approach**

Conclusions

- **Enormous transition is underway that affects all sectors of computing industry**
  - Motivated by power limits
  - Proceeding before emergence of the parallel programming model
- **Will lead to new era of architectural exploration given uncertainties about programming and execution model *(and we MUST explore!)***
- **Need to get involved now**
  - 3-5 years for new hardware designs to emerge
  - 3-5 years lead for new software ideas necessary to support new hardware to emerge
  - 5+ MORE years to general adoption of new software

More Info

- The Berkeley View
  - http://view.eecs.berkeley.edu
- NERSC Science Driven System Architecture Group
  - http://www.nersc.gov/projects/SDSA