Active Power Management Technology Challenges and Implications for Programming Models

John Shalf
Department Head for Computer Science
CTO: National Energy Supercomputing Center
Lawrence Berkeley National Laboratory

Teratec Forum
June 26, 2013
Ecole Polytechnique Palaiseau - France

BERKELEY LAB

U.S. DEPARTMENT OF ENERGY | Office of Science

Ter@tec

Active Power Management Technology Challenges and Implications for Programming Models

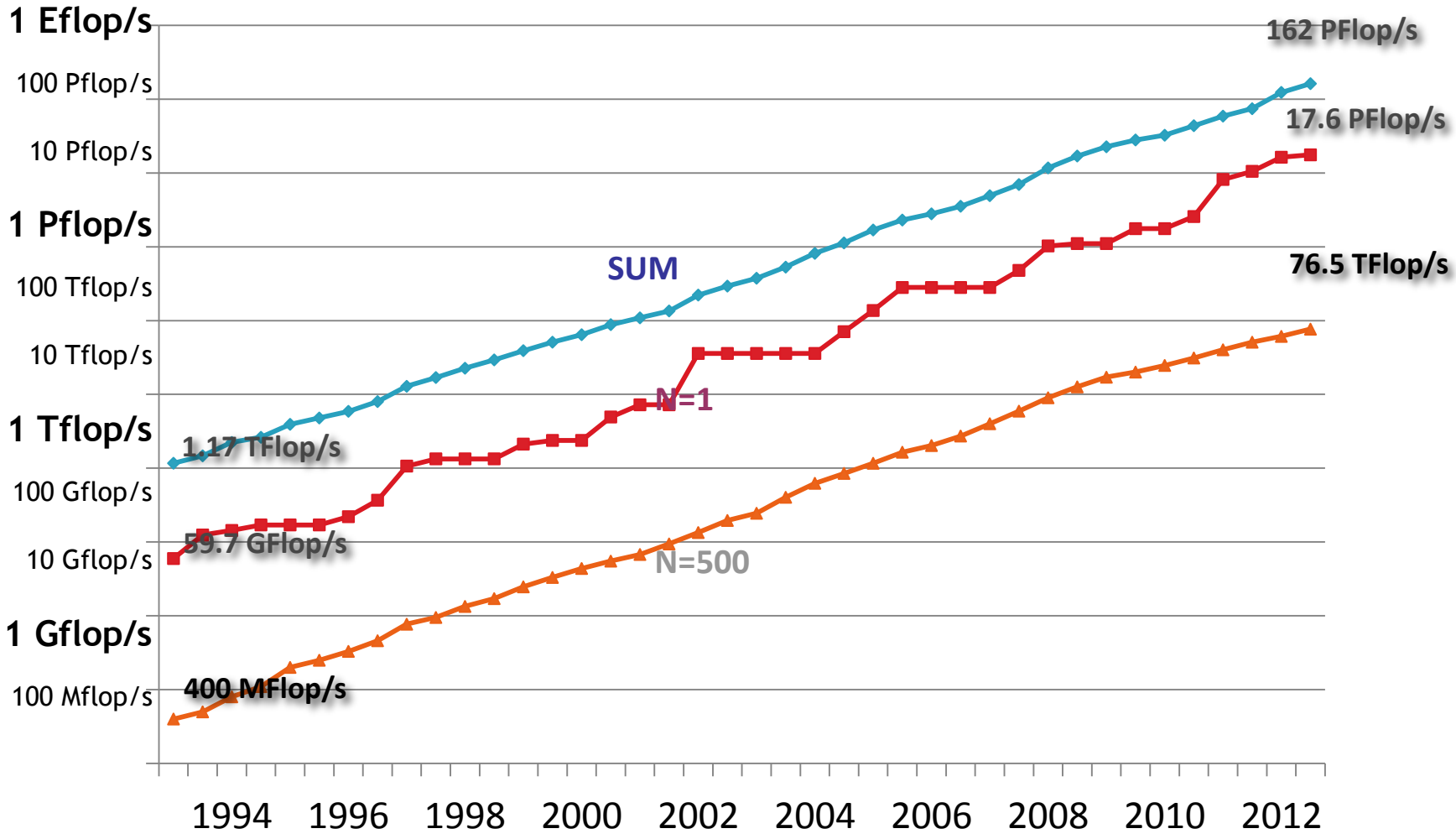
- **Active Power Management Technology Challenges and Implications for Programming Models**

- Dynamic Voltage and Clock Frequency scaling has dominated the discussion of active power management and power-aware algorithm design. However, there are many finer grained energy savings mechanisms that have yet to be fully exploited in server chip design. This talk will provide a survey of contemporary power management mechanisms incorporated into modern server chip designs as well as the many more aggressive mechanisms employed by mobile and embedded devices. For example, embedded and mobile devices make aggressive use of dark silicon, subthreshold logic design, and even opportunities for using software recovery mechanisms to enable a trade-off of soft error rates to achieve substantial power savings. However, HPC integrators and software designers face daunting challenges of coordinating mechanisms used for local optimal power management into large scale systems. Although these more aggressive techniques could enable enormous energy savings, these methods have a huge impact on the intrinsic performance inhomogeneity of our programming environment. Such changes fundamentally unravel the bulk-synchronous/SPMD programming paradigm that underpins the majority of our current HPC applications. Systemwide coordinated power management control loop cannot operate at the timescale that these local decisions are made. Such dramatic changes drive the study of alternative execution models to overcome the challenges of extreme performance heterogeneity and software-based resilience.

- **This talk will discuss these emerging technologies for more aggressive local power management and the implications for our programming environment. I will describe recent research into alternative execution models, and describe results from example implementations of these alternative models for computation.**

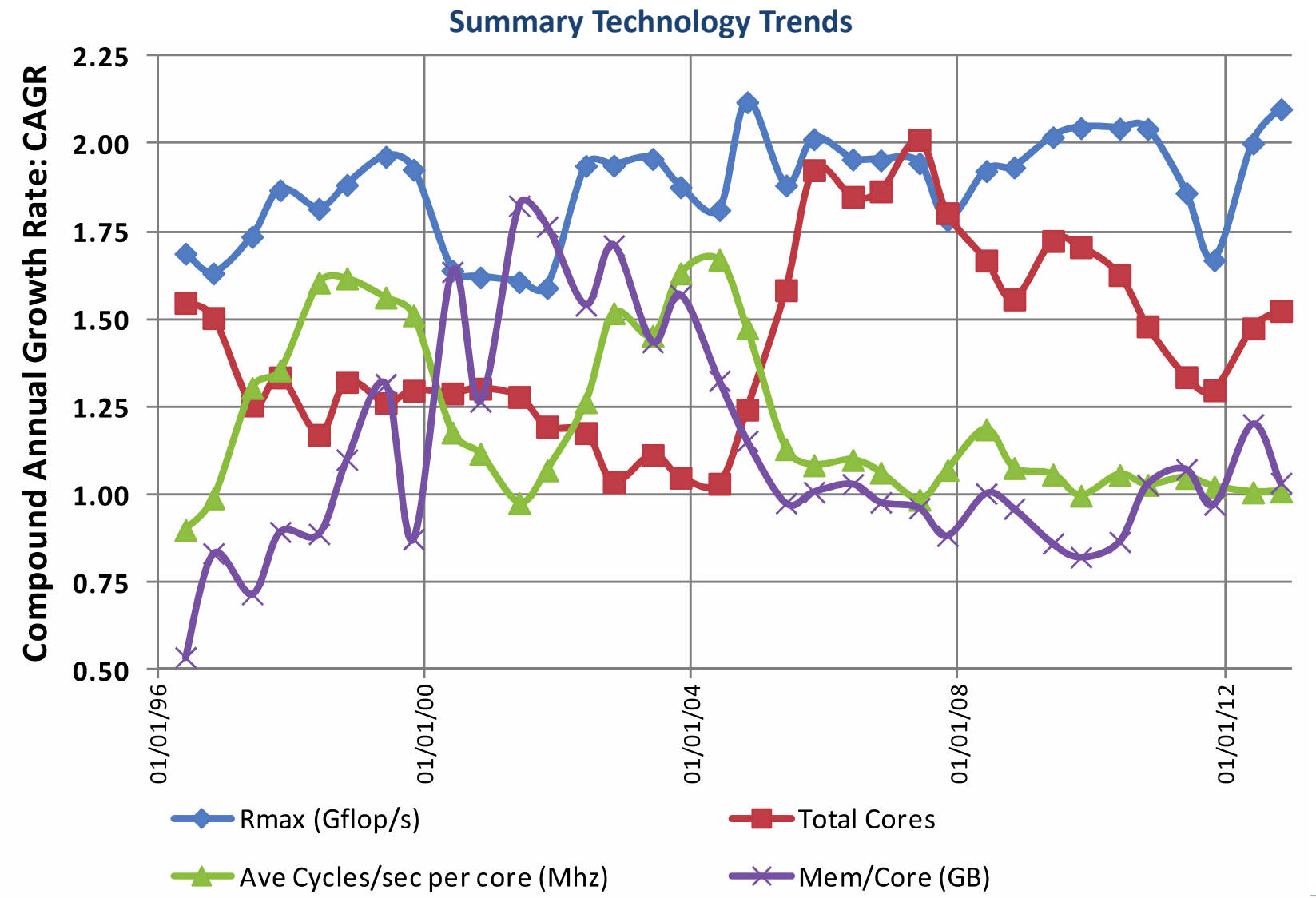
Context

Performance Development over 3 Decades



Source: TOP500 November 2012

It's the End of the World as We Know It!



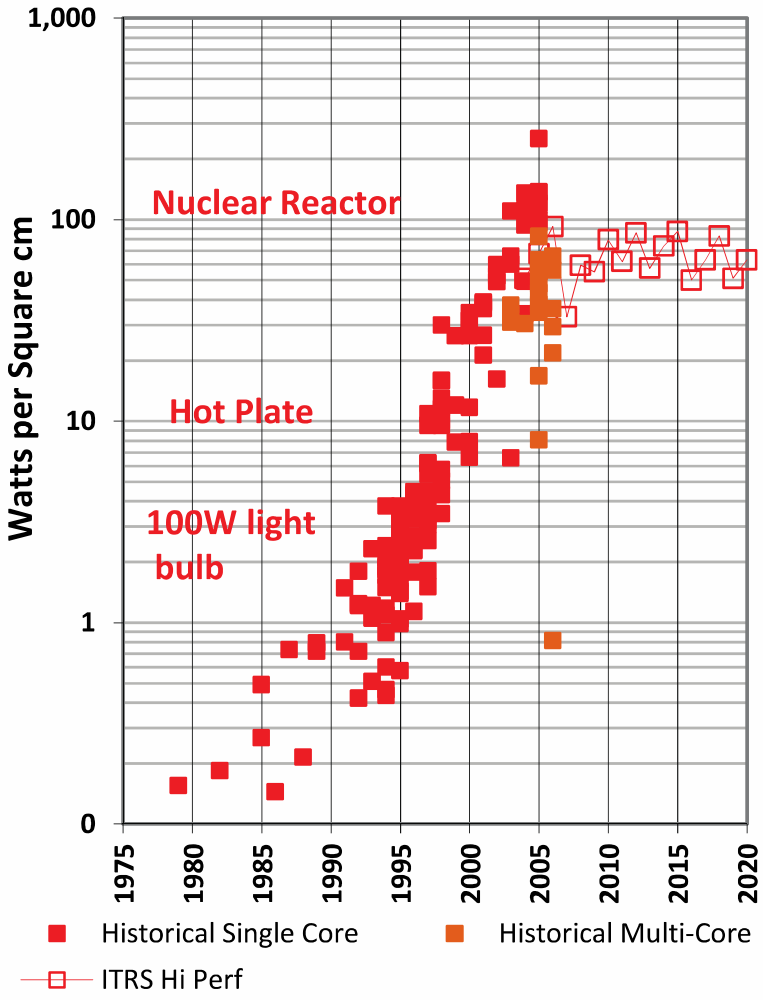
Source: Kogge and Shalf, IEEE CISE

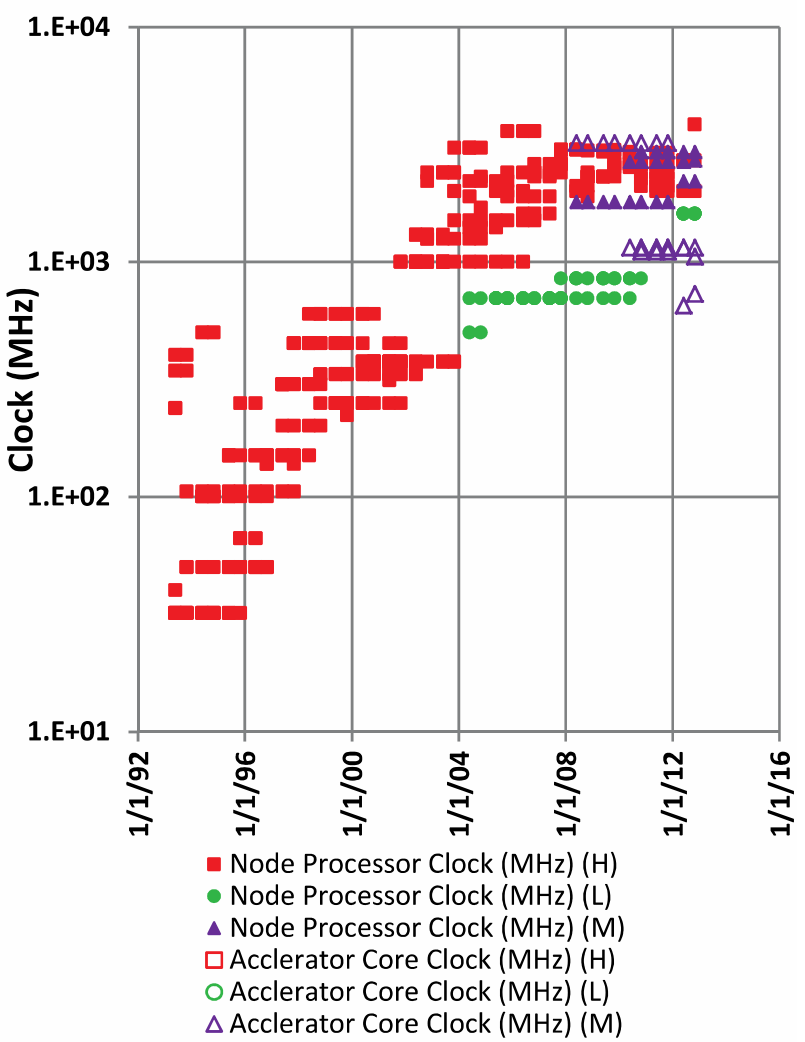
BERKELEY LAB

Ter@tec

The Power and Clock Inflection Point in 2004

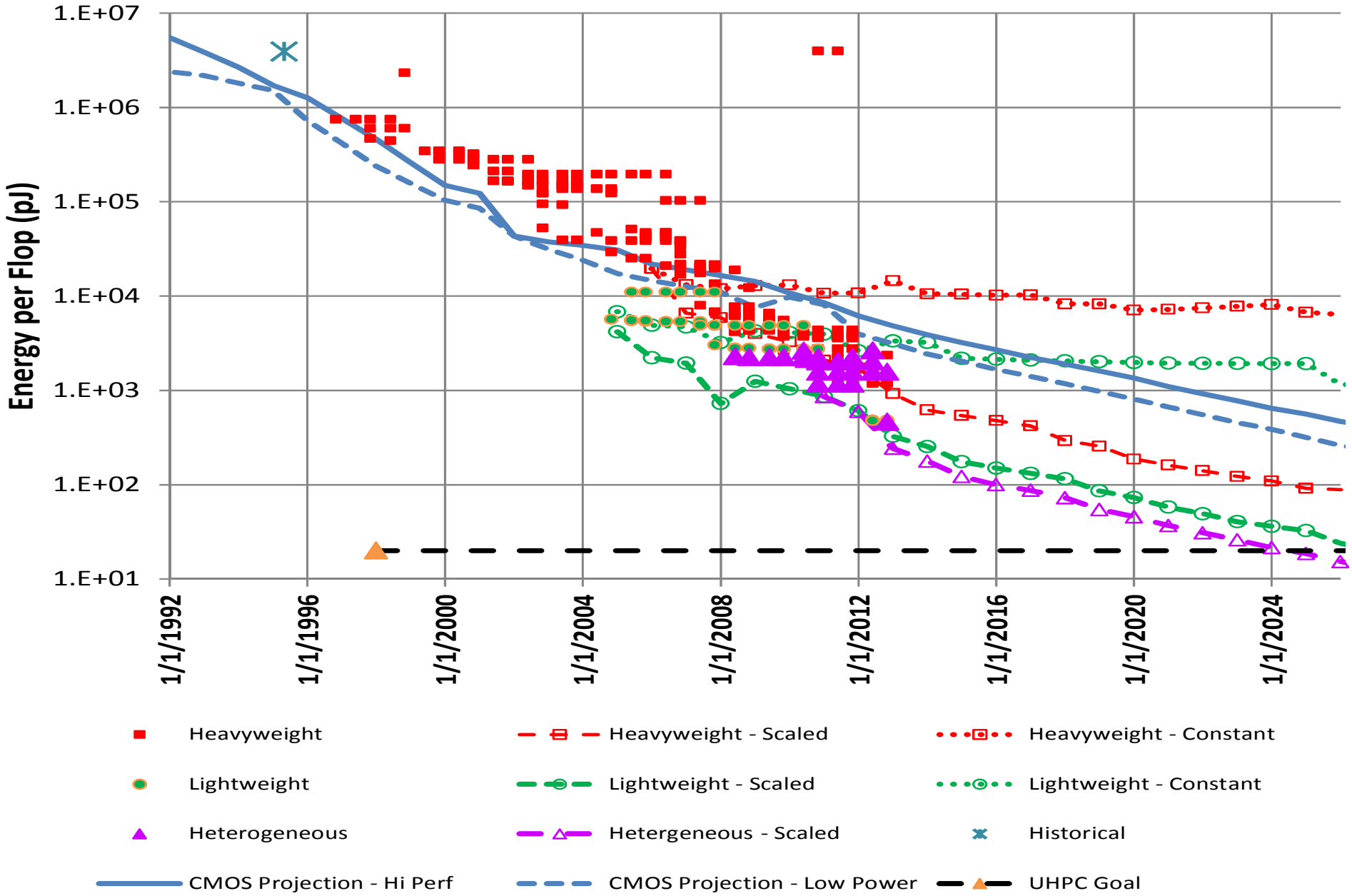
(the only path forward is to reduce power!!!)





Source: Kogge and Shalf, IEEE CISE

Stretching Towards Exaflop in 2024



Where Can We Find the Power Savings by 2018?

Technology Area	Current Status	Margin for Improvement % (factor)
Technology Scaling (*Moore's law + whats left of Dennard Scaling*)	With aggressive NTV, can lower supply voltages a bit more	200% (~2x)
Power Distribution	Huge improvements in distribution 480v-3 phase operating at 70% efficiency end-to-end	20% (1.2-1.3x)
Cooling Technology (*primary opportunity is increased density*)	Typical PUE's of 1.3, and can push down 1 (< 1 with cogeneration)	30% (1.3-1.4x)
Processor/ASIC Architecture	More SOC integration, Hybrid cores, Near threshold voltage	400% (4x) *(3x in circuits Dally ISC13)*
Memory	DDR is 35pj/bit (HMC Gen2 at 10pj/bit and moving to 7pj/bit)	400% (4x)
Dynamic Power Management	Finer grained power management using embedded voltage regulation (leakage limits margin)	200% (2x)

Where Can We Find the Power Savings?

Technology Area	Current Status	Margin for improvement % (factor)
Technology Scaling *(Moore's law + whats left of Dennard Scaling)*	With aggressive methods, can lower supply voltages a bit more	200% (~2x)
Power Distribution	Huge improvements in distribution 480v-3 phase operating at 70% efficiency end-to-end	20% (1.2-1.3x)
Cooling Technology (*primary opportunity is increased density*)	Typical PUE's of 1.3, and can push down 1 (< 1 with cogeneration)	30% (1.3-1.4x)
Processor/ASIC Architecture	More SOC integration, Hybrid cores, Near threshold voltage	400% (4x) *(3x in circuits Dally ISC13)*
Memory	DDR is 35pj/bit (HMC Gen2 at 10pj/bit and moving to 7pj/bit)	400% (4x)
Dynamic Power Management	Finer grained power management using embedded voltage regulation (leakage limits margin)	200% (2x)

Observations on Energy Efficient Computer Architecture

Lessons Learned from Green Flash (2006-2009) and Green Wave (2009-present)

Primary Observations from Green Flash/Wave

- **Technology**
  - Use small energy efficient cores
  - Hybrid: specialize many cores for work and fat cores for OS & drivers
  - Converging with Embedded technology
  - SoC to Minimize costs
- **Architecture (ISA and Chip-level Fabric)**
  - Include only what you need
  - Extend to manage data movement
- **Methodology**
  - Rapid prototyping with embedded tools
  - Rapid software tuning using Auto-tuning
  - Put it together, and we have accelerated codesign process
- **Some quick examples**
  - Climate
  - Seismic imaging

Governing Design Principle: Reduce Waste!

- **Biggest win was in what we do NOT include in an HPC Design (CoDesign for energy optimization)**

- 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."*

Design Methodology: Co-Design

(overview of Green Flash and Green Wave)

- **Research effort: study feasibility of designing an application-targeted supercomputer and share insight w/community**

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

- **What is *(was)* 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

Embedded Design Automation

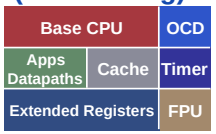
(Using FPGA emulation to do rapid prototyping)

Processor configuration

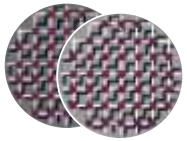
1. Select from menu
2. Automatic instruction discovery (XPRES Compiler)
3. Explicit instruction description (TIE)

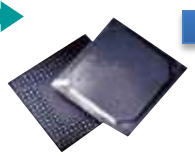
Processor Generator (*Tensilica*)

Application-optimized processor implementation (RTL/Verilog)



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

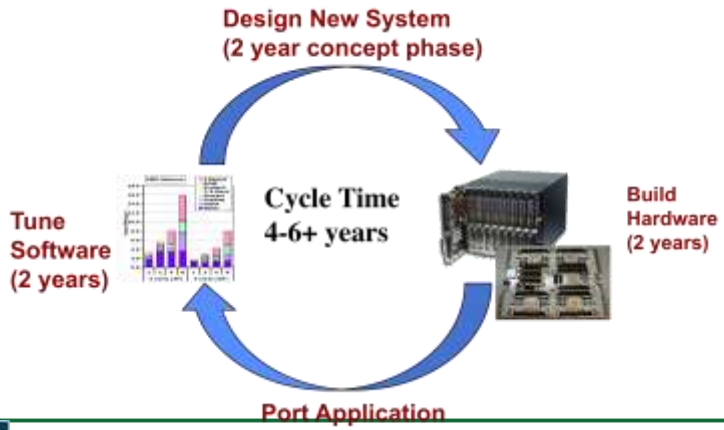




Build with any process in any fab

Or "tape out"
To FPGA

RAMP FPGA-accelerated Emulation of ASIC

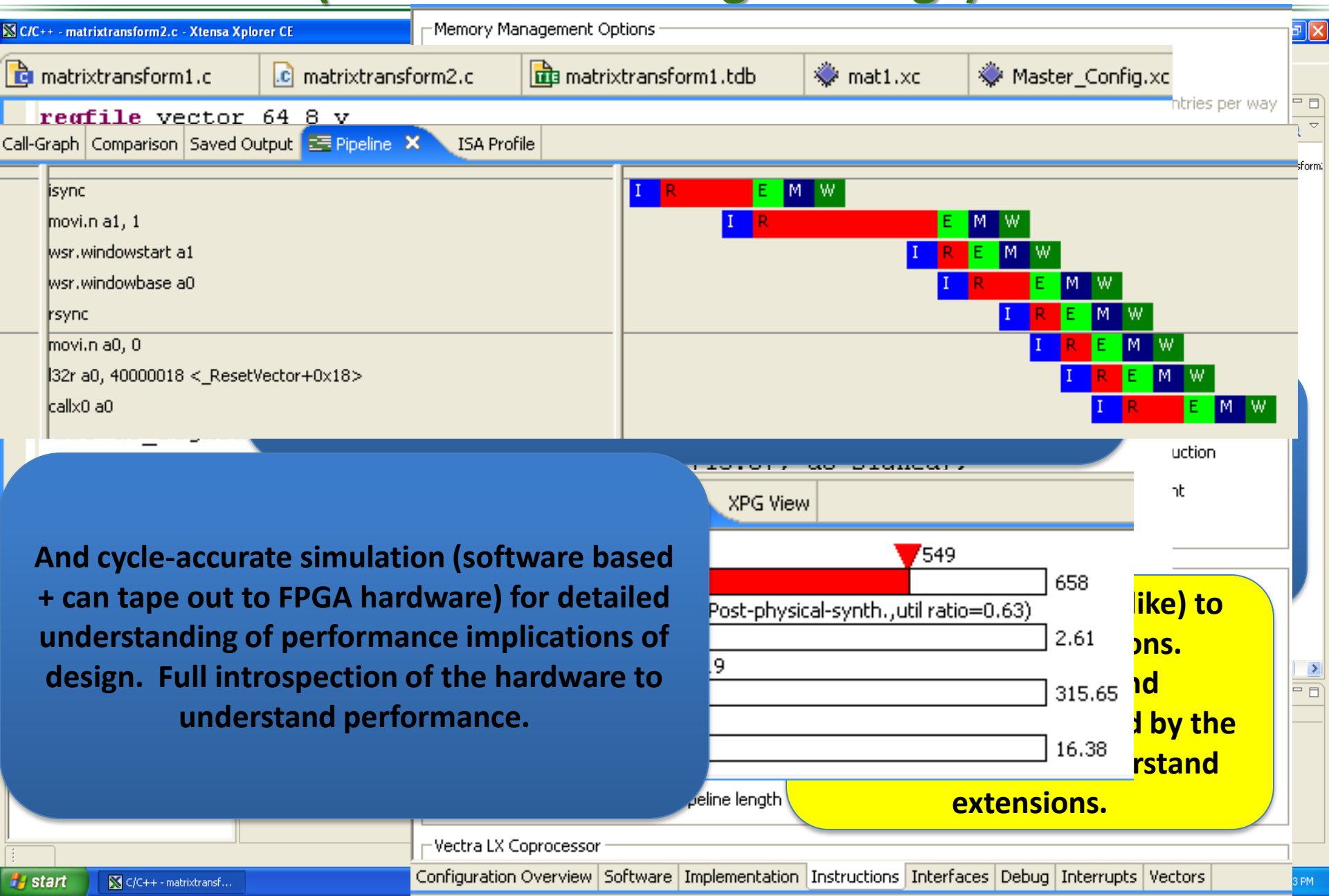


This is central to CoDesign (and it ain't new)

BERKELEY LAB

Ter@tec

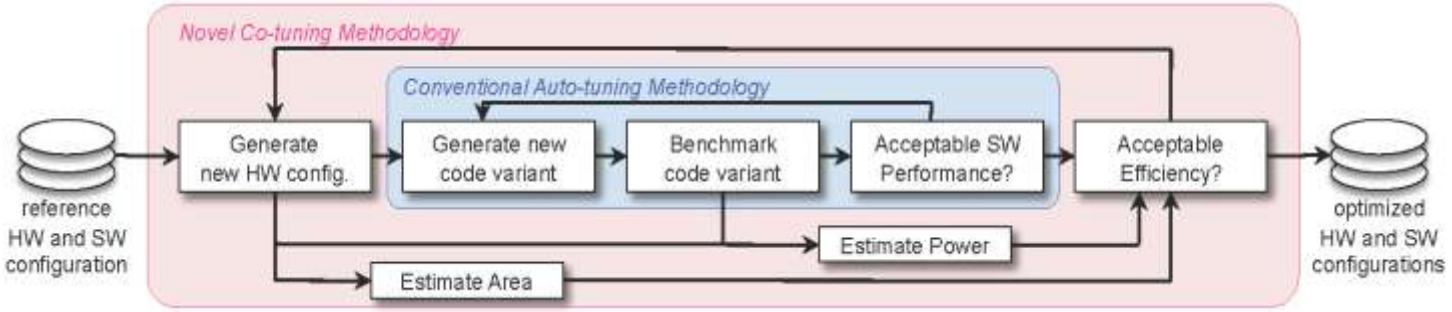
A tour of the Processor Generator (software modeling for triage)

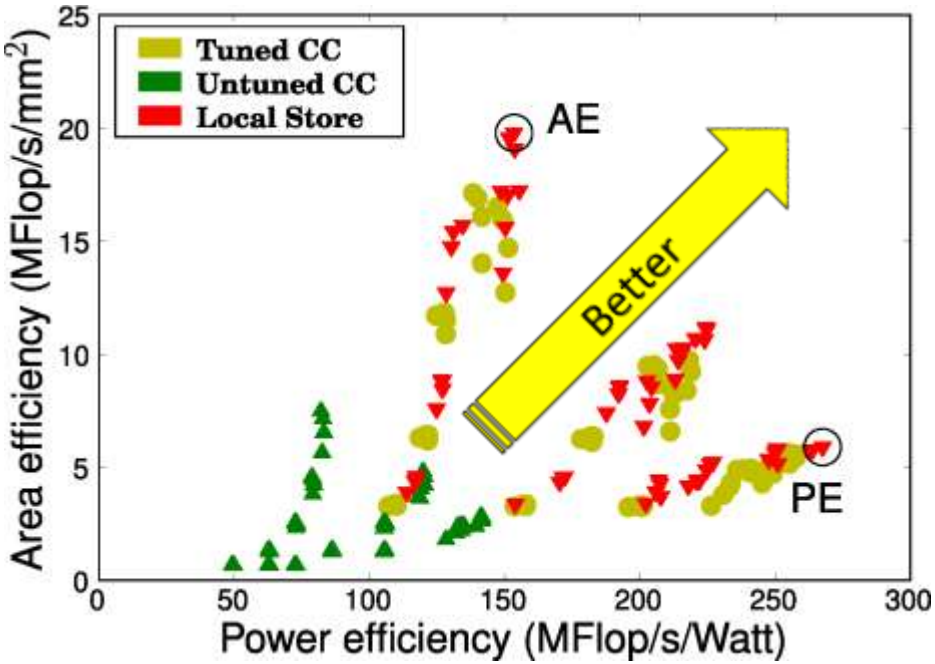


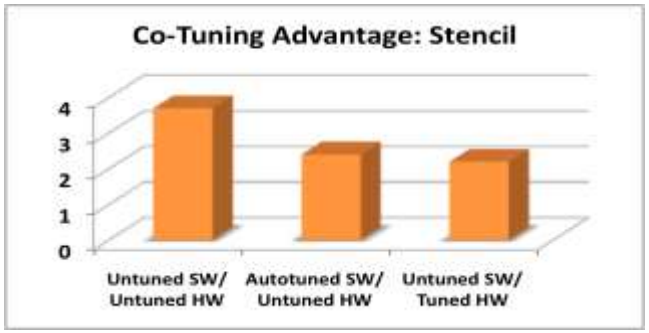
And cycle-accurate simulation (software based + can tape out to FPGA hardware) for detailed understanding of performance implications of design. Full introspection of the hardware to understand performance.

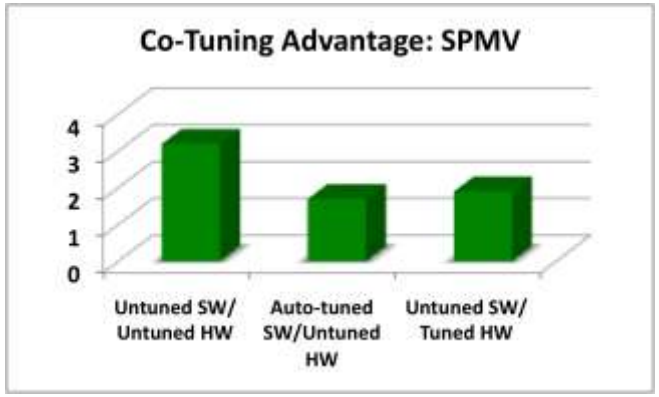
extensions.

Hardware/Software Co-Tuning for Energy Efficiency



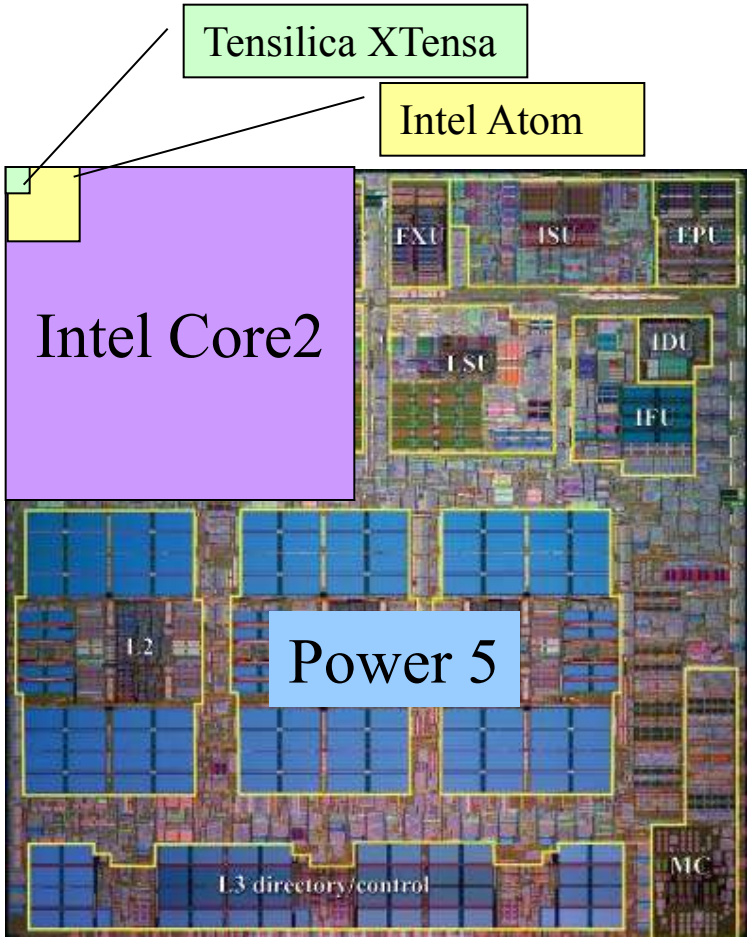






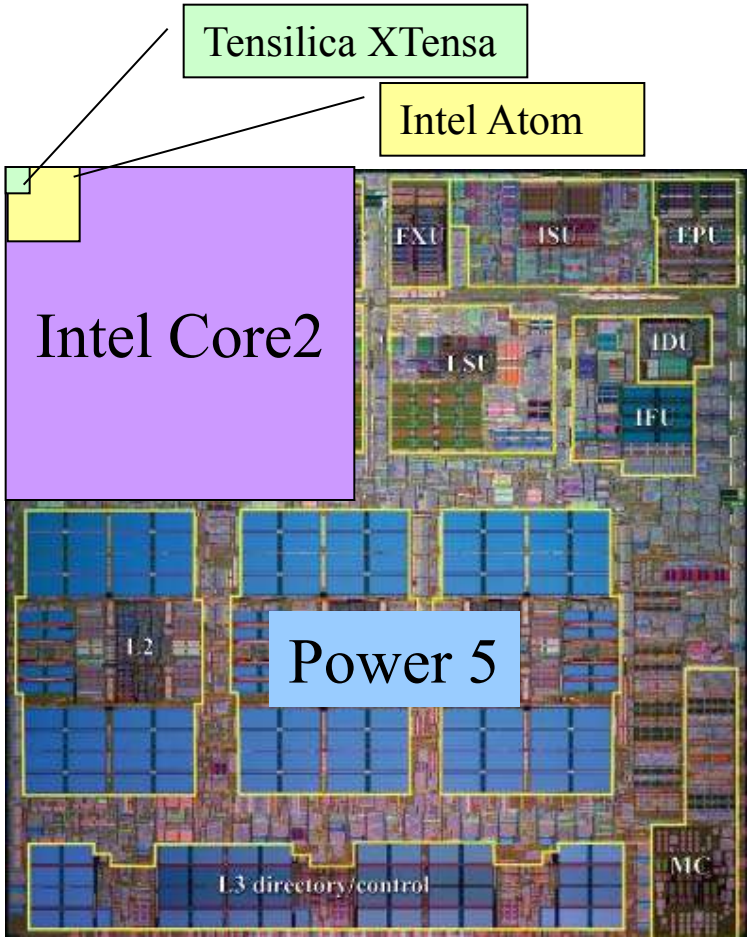
Co-Tuning can improve power-efficiency and area-efficiency by ~4x

Low-Power Design Principles for Core



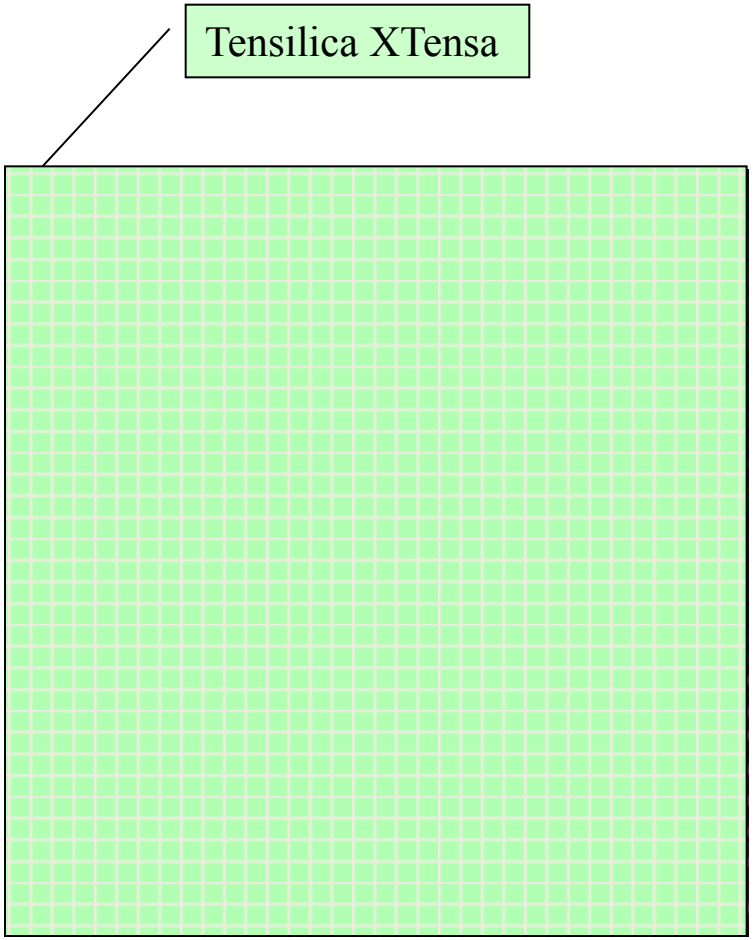
- **Cubic power improvement with lower clock rate due to V^2F**
- **Slower clock rates enable use of simpler cores**
- **Simpler cores use less area (lower leakage) and reduce cost**
- **Tailor design to application to REDUCE WASTE**

Low-Power Design Principles for Core



- **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 (Moto Razor) :**
  - 0.09W@600MHz
  - 400x more *(80x-120x sustained)*

Low Power Design Principles for Core



- **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 more power efficient.

System on Chip (SoC)

Embrace Embedded Technology

Use SoC to reduce energy and design complexity

(back to “include only what you need”)

Design Principle: SoC from IP Logic Blocks

Increased integration reduces power and reduces costs!

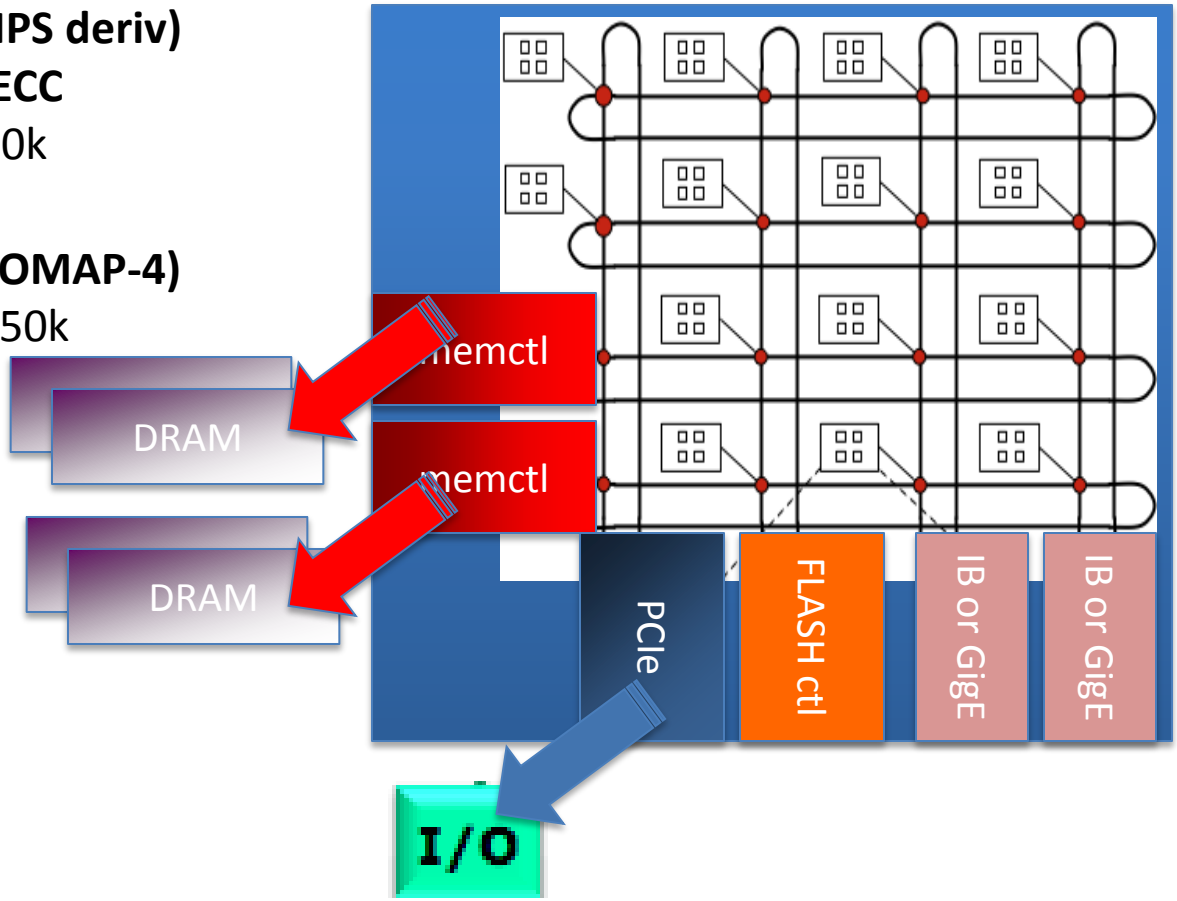
Processor Core (ARM, Tensilica, MIPS deriv)
With extra "options" like DP FPU, ECC
IP license cost $150k-$500k

NoC Fabric: (Arteris, Denali, other OMAP-4)
IP License cost: $200k-$350k

HMC or DDR memory controller
(Denali / Cadence, SiCreations)
+ Phy and Programmable PLL
IP License: $250-$350k

PCIe Gen3 Root complex
IP License: $250k

Integrated FLASH Controller
IP License: $150k



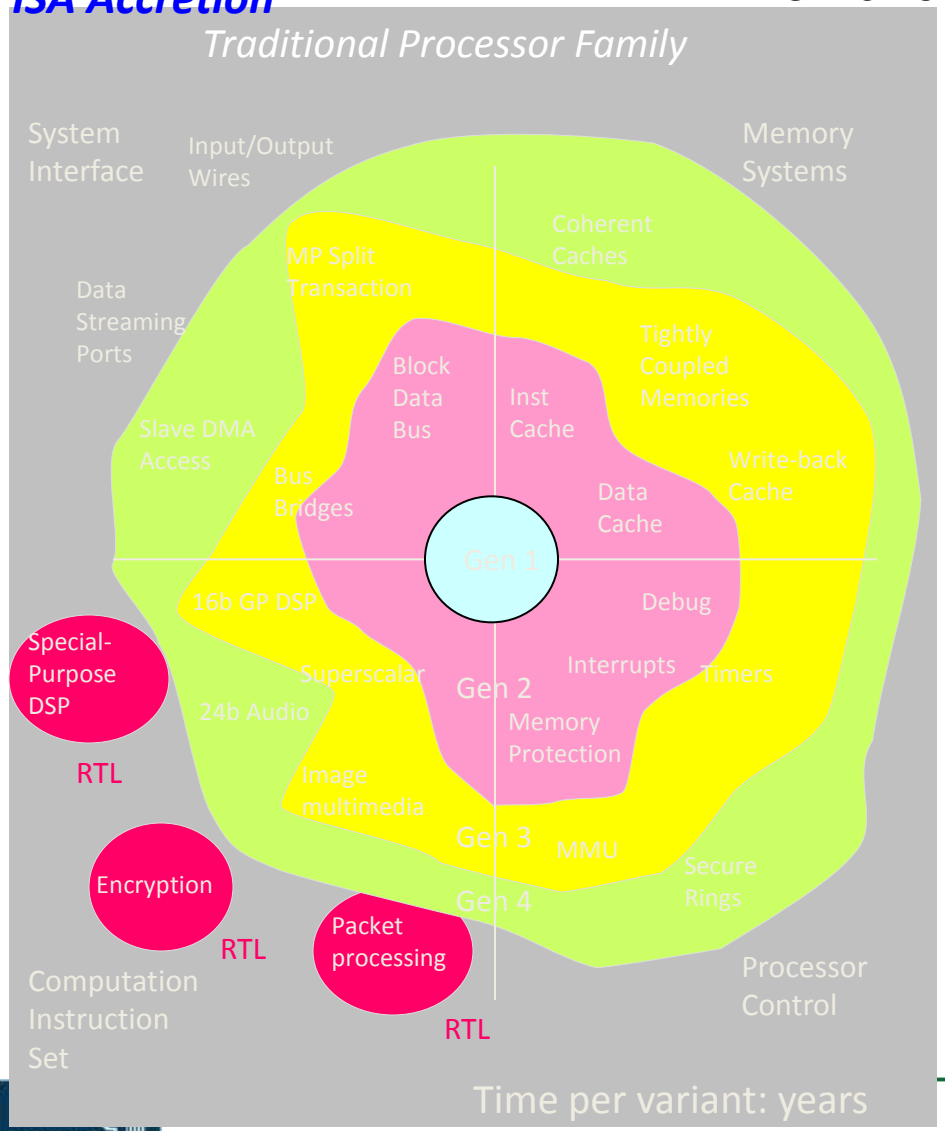
10GigE or IB DDR 4x Channel
IP License: $150k-$250k

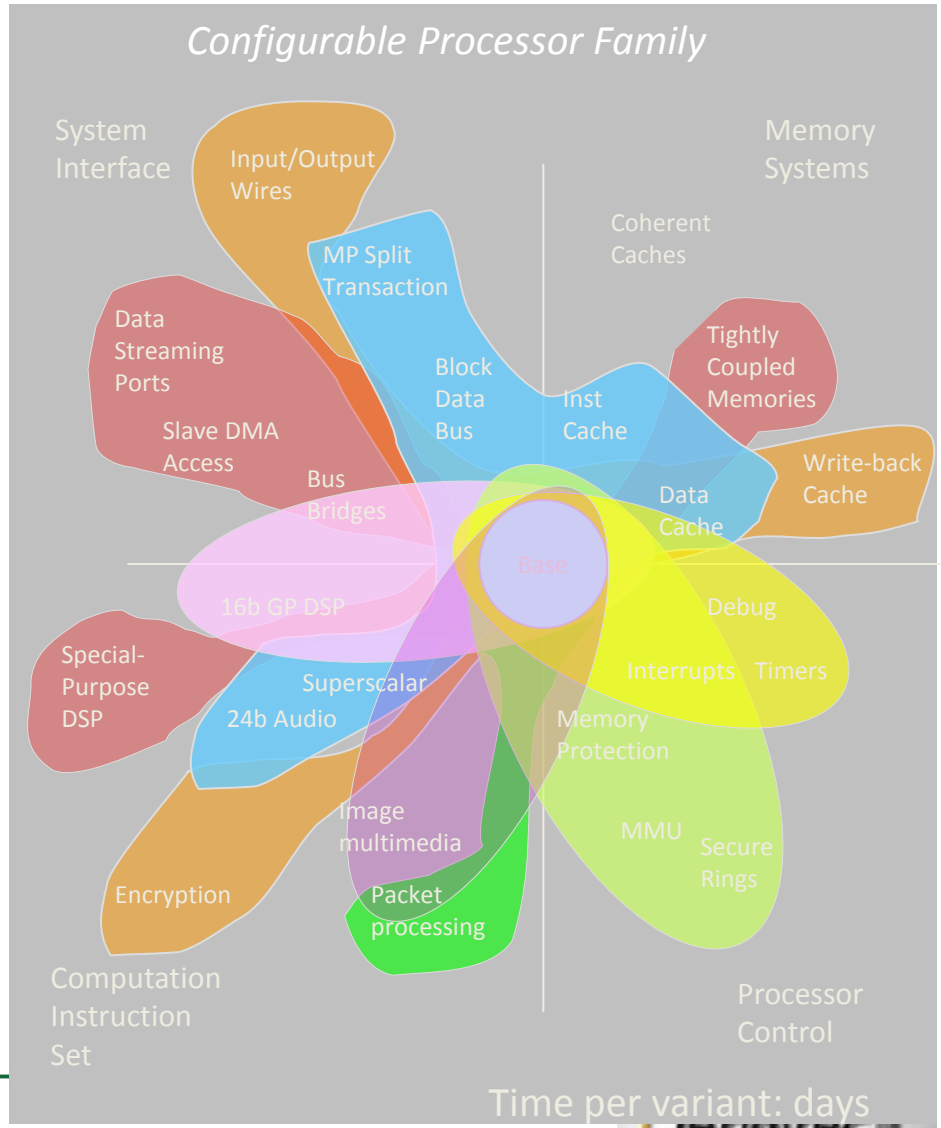
ISA Design Principles

Current Commoditization Strategy Is NOT Aligned with Low Power Design Principles

Chris Rowen: Tensilica

ISA Accretion





Area = silicon cost and power

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

mnemonic	op1	op2	op3	op4	iext	pf	0F	po	so	o	proc	st	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....a.c		ASCII Adjust AX Before Division
AAM	AL	AH						D4	0A								o..szapc	...sz.p.	o....a.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	zAX	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	zAX	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	P3+										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	P3+										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....a.c		Adjust AX Before Division
ALTER						64					P4+	u[1]									Alternating branch prefix (used only with Jcc instructions)
AMX	AL	AH	imm8					D4									o..szapc	...sz.p.	o....a.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	zAX	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	03+				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	P3+										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	P3+										Bitwise Logical AND of Packed Single-FP Values

More Unused Opcodes

ARPL	r/m16	r16	
BOUND	r16/32	m16/32&16/32	eFlags
BSF	r16/32/64	r/m16/32/64	
BSR	r16/32/64	r/m16/32/64	
BSWAP	r16/32/64		
BT	r/m16/32/64	r16/32/64	
BT	r/m16/32/64	imm8	
BTC	r/m16/32/64	imm8	
BTC	r/m16/32/64	r16/32/64	
BTR	r/m16/32/64	r16/32/64	
BTR	r/m16/32/64	imm8	
BTS	r/m16/32/64	r16/32/64	
BTS	r/m16/32/64	imm8	
CALL	rel16/32		
CALL	rel32		
CALL	r/m16/32		
CALL	r/m64		
CALLF	ptr16:16/32		
CALLF	m16:16/		
CBW	*AX*		
CBW	*AX*		
CWDE	*EAX*		
CDQE	*RAX*		
CDQ	*EDX*		
CLC			
CLD			
CLFLUSH	m8		
CLI			
CLTS	*CR0*		
CMC			
CMOVB	r16/32/		
CMOVNAE	r16/32/		
CMOVC	r16/32/		
CMOVBE	r16/32/		
CMOVNA	r16/32/		
CMOVL	r16/32/		
CMOVNGE	r16/32/		
CMOVLE	r16/32/		
CMOVNG	r16/32/		
CMOVNB	r16/32/		
CMOVAE	r16/32/		
CMOVNC	r16/32/		
CMOVNBE	r16/32/		
CMOVA	r16/32/		
CMOVNL	r16/32/		
CMOVGE	r16/32/		
CMOVNLE	r16/32/		
CMOVG	r16/32/		

CVTPS2PD	xmm	xmm/m128	
CVTPS2PI	mm	xmm/m64	
CVTSD2SI	r32/64	xmm/m64	
CVTSD2SS	xmm	xmm/m64	
CVTSI2SD	xmm	r/m32/64	
CVTSI2SS	xmm	r/m32/64	
CVTSS2SD	xmm	xmm/m32	
CVTSS2SI	r32/64	xmm/m32	
CVTTPD2DQ	xmm	xmm/m128	
CVTTPD2PI	mm	xmm/m128	
CVTTPS2DQ	xmm	xmm/m128	
CVTTPS2PI	mm	xmm/m64	
CVTTSD2SI	r32/64	xmm/m64	
CVTTSS2SI	r32/64	xmm/m32	
CWD	*DX*	*AX*	
CWD	*DX*	*AX*	
CDQ	*EDX*	*EAX*	
CQO	*RDX*	*RAX*	
CWDE	*EAX*	*AX*	
DAA	AL		
DAS	AL		

r16/32/64	r/m16/32/64	
r16/32/64	r/m16/32/64	
r16/32/64	r/m16/32/64	
r/m8	r8	
r/m16/32/64	r16/32/64	
r8	r/m8	
r16/32/64	r/m16/32/64	
AL	imm8	
rAX	imm16/32	
r/m8	imm8	
r/m16/32/64	imm16/32	
r/m8	imm8	
r/m16/32/64	imm8	
xmm	xmm/m128	imm8
xmm	xmm/m128	imm8
m8	m8	
m8	*m8*	
m16	m16	

FXCH4	*ST*	STi
FXCH4	*ST*	STi
FXCH7	*ST*	STi
FXCH7	*ST*	STi
FXRSTOR	*ST*	*STi*
FXRSTOR	*ST*	*STi*
FXSAVE	m512	*ST*
FXSAVE	m512	*ST*
FXTRACT	*ST*	
FYL2X	*STi*	*ST*
FYL2XP1	*STi*	*ST*
GS	*GS*	
HADDPD	xmm	xmm/m128
HADDPS	xmm	xmm/m128
HLT		
HSUBPD	xmm	xmm/m128

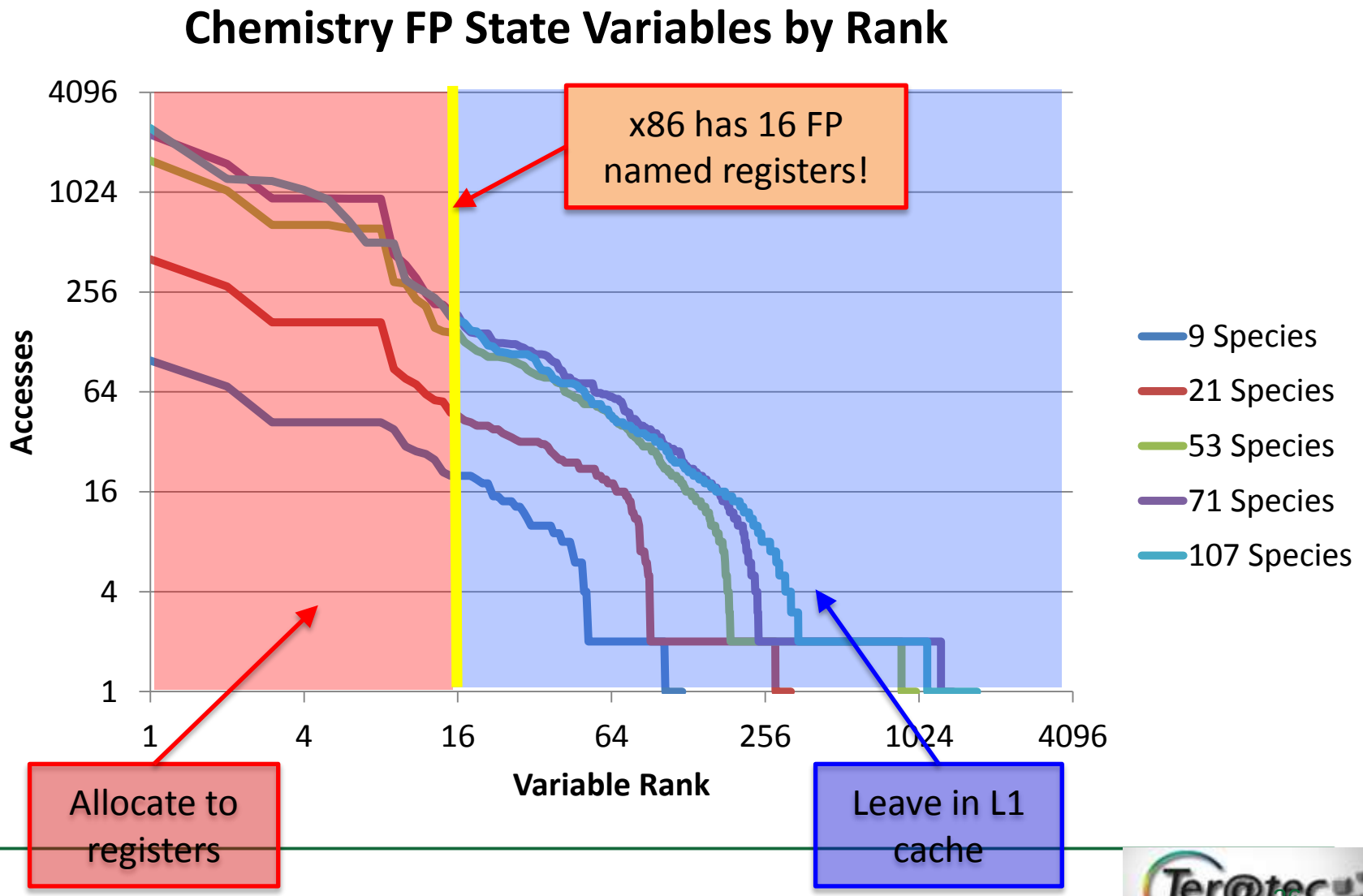
INTO	eFlags	
INVD		
INVLPG	m	

- **We only need 80 out of the nearly 300 ASM instructions from 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)

BERKELEY LAB

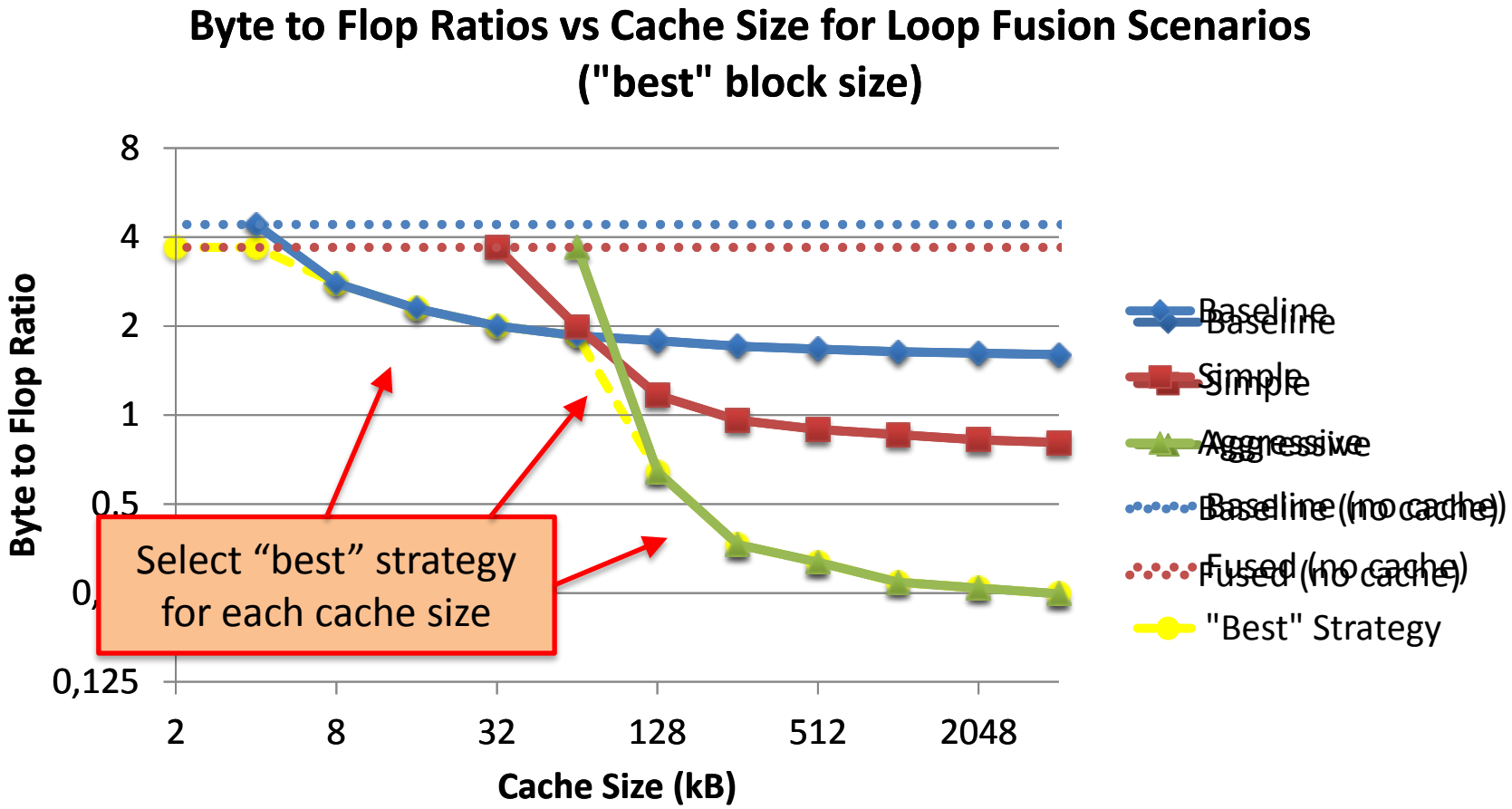
Typical Processors Underprovisioned for Registers



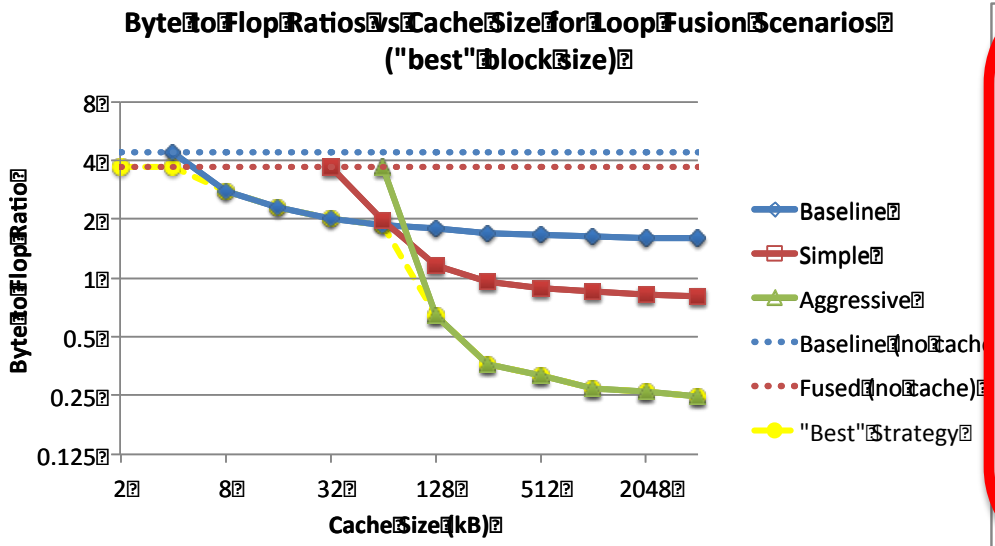
Typical Processors Underprovisioned for L1 Cache

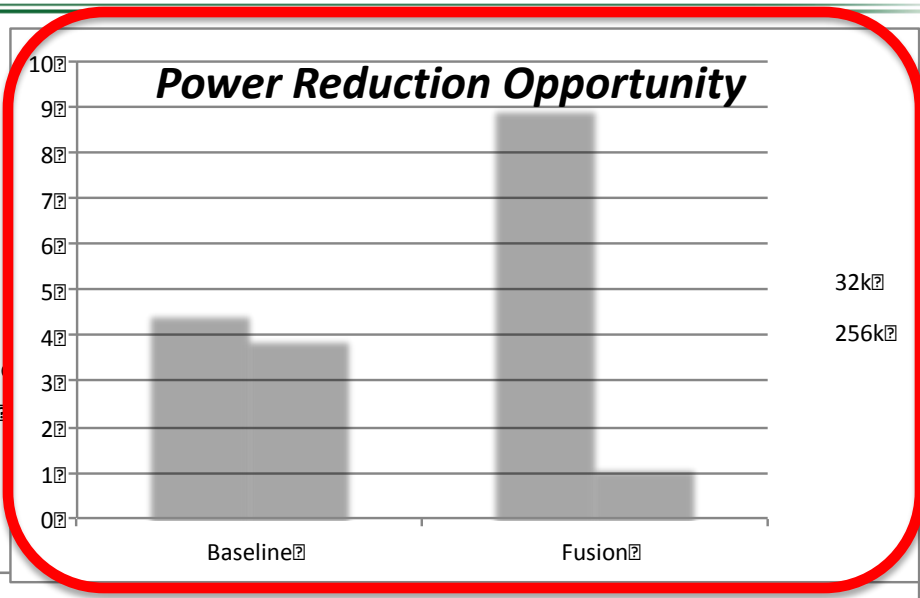
Huge opportunity to reduce memory bandwidth requirements!!

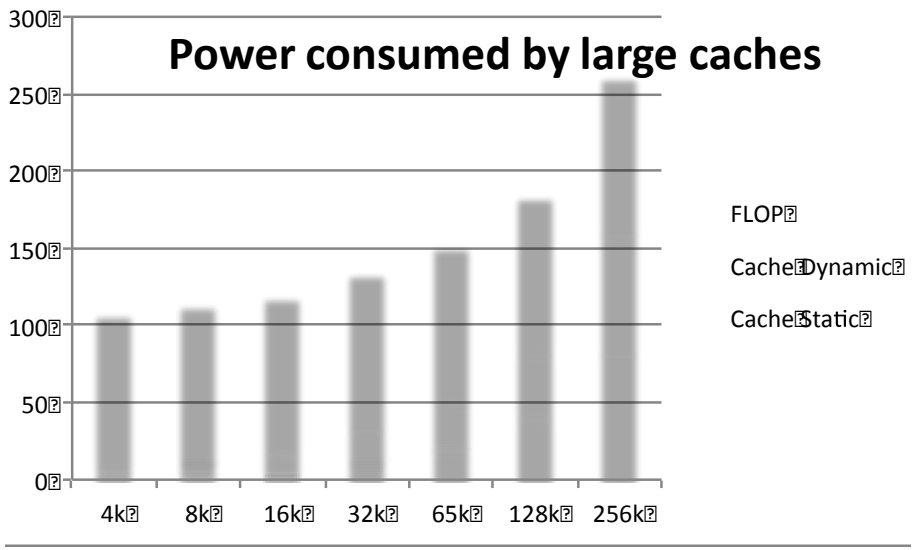
Current execution environments do not enable us to reason about this kind of fusion

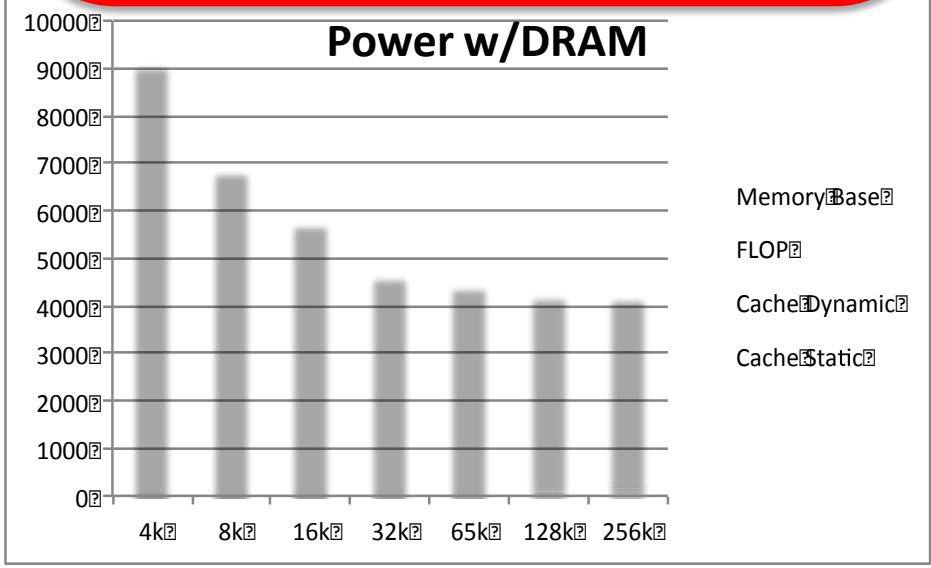


Power Consequences of Big L1 Scratchpads









Data Movement

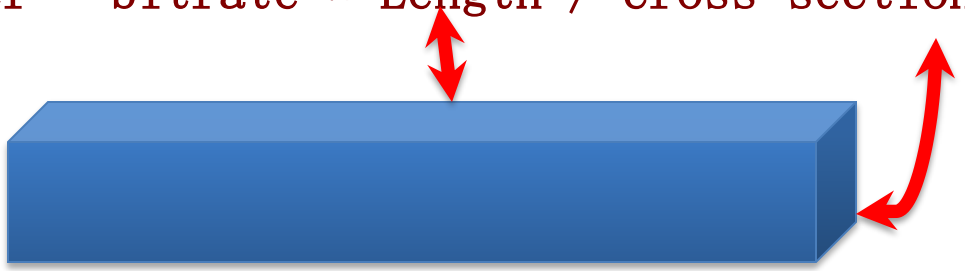
The “un-core”

Managing Data Movement

The problem with Wires:

Energy to move data proportional to distance

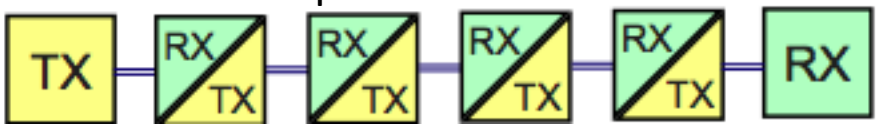
- **Cost to move a bit on copper wire:**
  - Power = bitrate * Length / cross-section-area



- **Wire data capacity constant as feature size shrinks**
- ***Cost to move bit proportional to distance***
- ***~1-5TByte/sec max feasible off-chip BW (10-20GHz/pin)***
- ***Photonics is a wildcard***

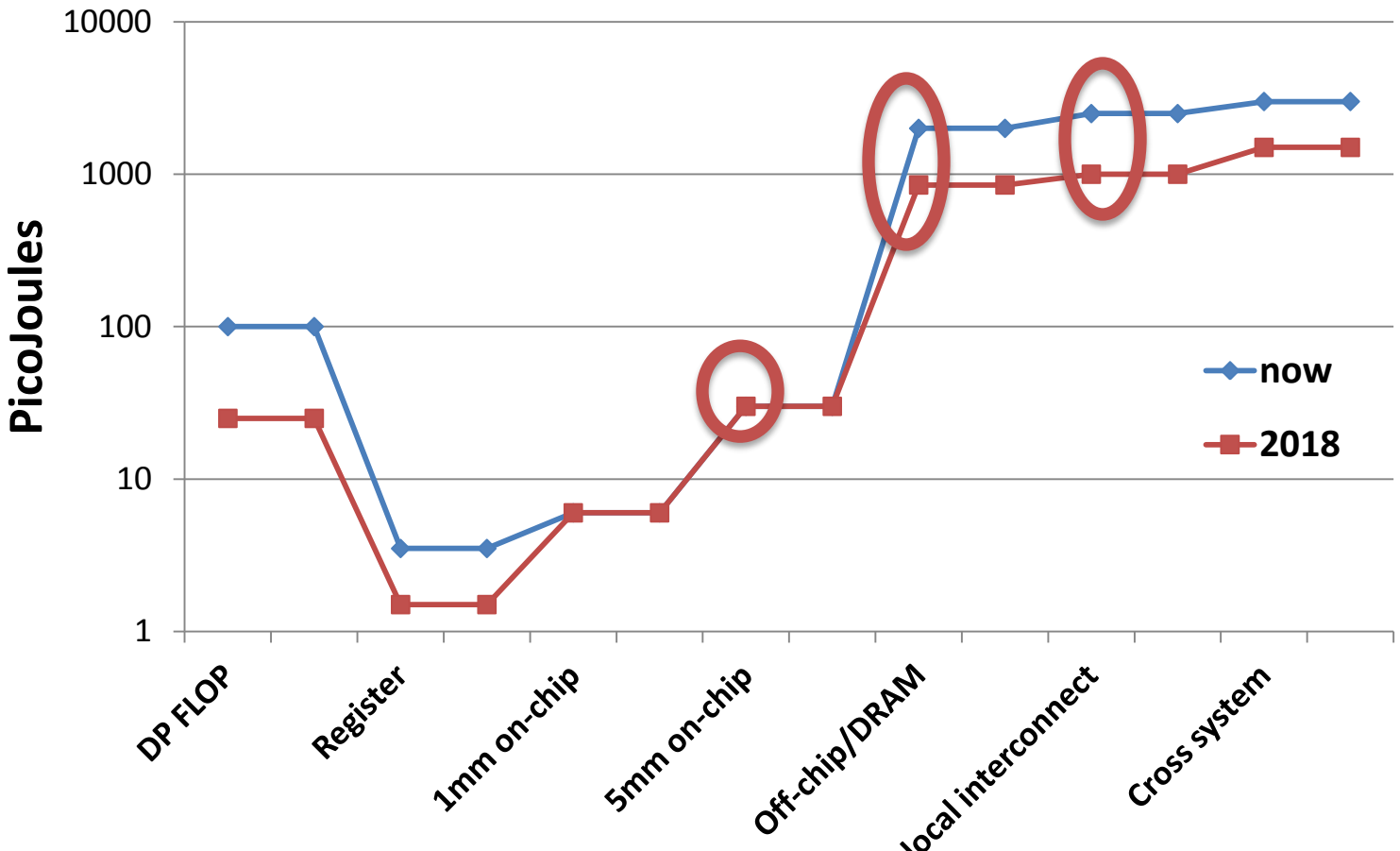
Photonics requires no redrive and passive switch little power

Copper requires to signal amplification even for on-chip connections



Data Movement Costs

Energy Efficiency will require careful management of data locality



Important to know when you are on-chip and when data is off-chip!

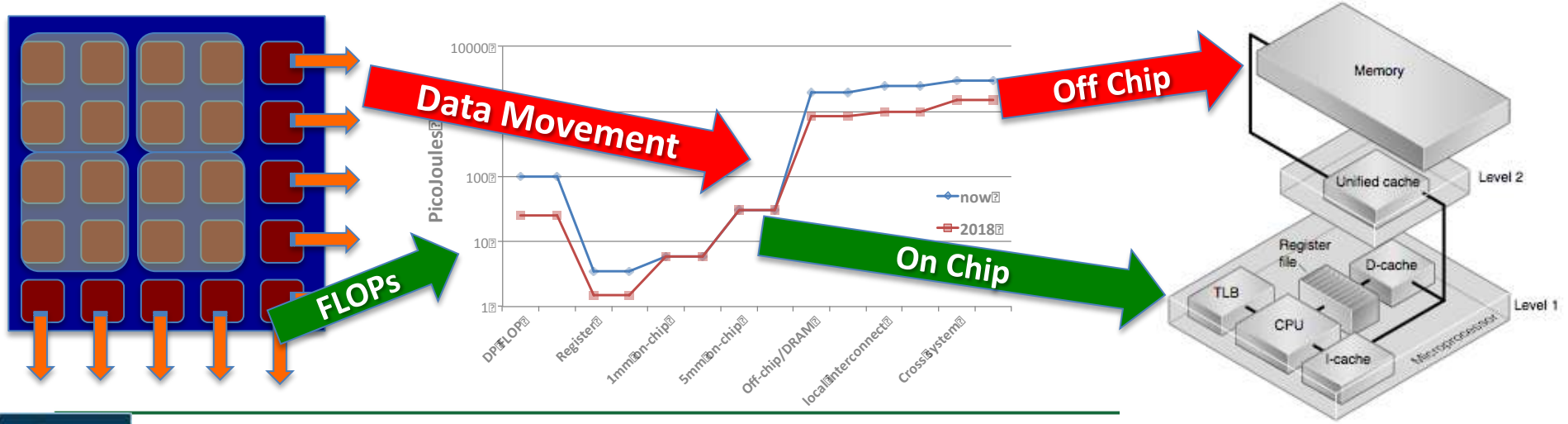
Consequences of Data Movement Costs

- **Current Programming Environments Over-Value FLOPS and Under-Value data movement**
  - Order of complexity is based on FLOPS (not data movement)
- **Programming environment virtualizes data locality or even ignores it!**
  - OpenMP assumes uniform costs between cores within node
  - MPI assumes uniform costs between nodes within system
- **We quantify the consequences due to virtualizing data locality!**
  - Assumptions that are increasingly diverging with hardware reality!!!

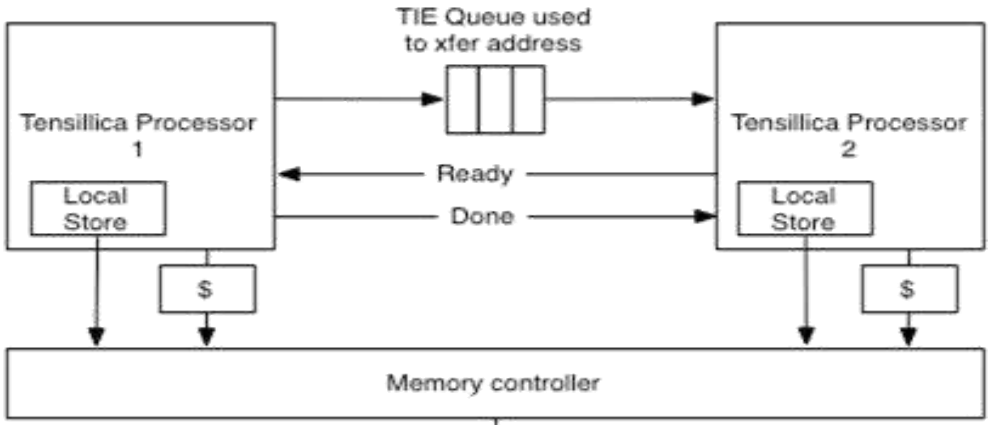
Horizontal Locality Management
(topological optimization)

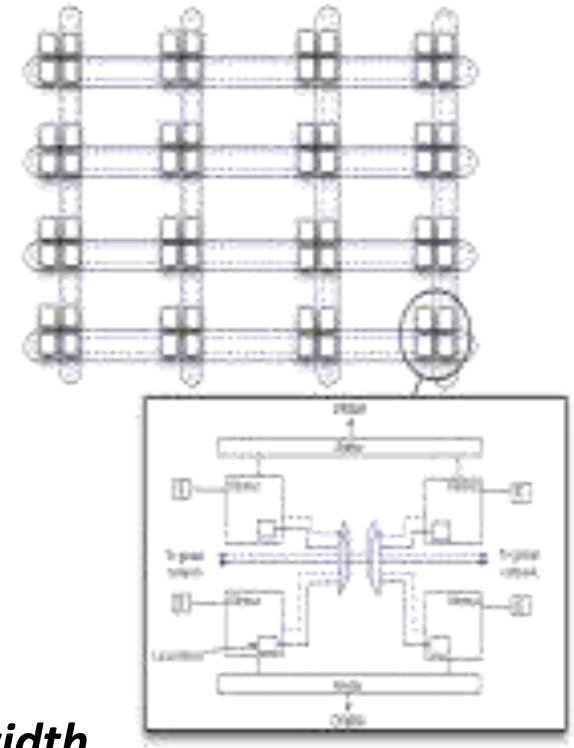
Energy Cost of Moving Data Exceeds FLOPs
(Must Conserve Data Movement – NOT FLOPs)

Vertical Locality Management
(spatio-temporal optimization)



Design Principle: Focus ISA on Data Movement





- **Lightweight energy efficient cores**
- **Better control of data movement**
  - *Direct message queues between cores*
  - *Local Store into the global address space*
- **Local-store for more efficient use of *memory bandwidth***
  - *Can put Local store **side-by-side** with conventional cache*
  - *Design library enables incremental porting to local store*
- **Hardware support for lightweight synchronization**
  - Enables direct inter processor communication for low-overhead synchronization
  - Maintain consistency between memory-mapped local stores

Design Methodology: CoDesign

Design application together with HPC systems to achieve better integrated and more efficient hardware/software solution

Design Methodology: Co-Design

(overview of Green Flash and Green Wave)

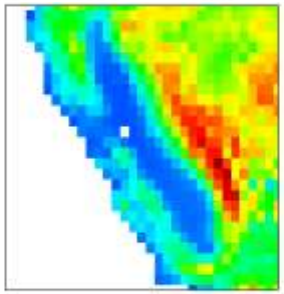
- **Choose the science target first**
  - *climate*
  - *seismic imaging*
- **Design systems for applications**
  - *Use rapid prototyping environments from embedded*
  - *Apply HW design principles discussed above*
- **Co-Design:** Design hardware, software, scientific algorithms together using
  - hardware emulation
  - auto-tuning

Example Design Study: *Global Cloud-Resolving Climate Models*

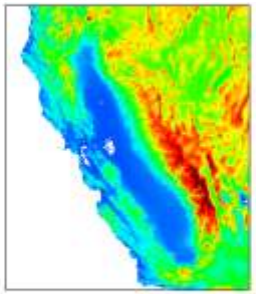
Lowest Energy To Solution Insufficient (need for speed)

http://www.lbl.gov/cs/html/greenflash.html

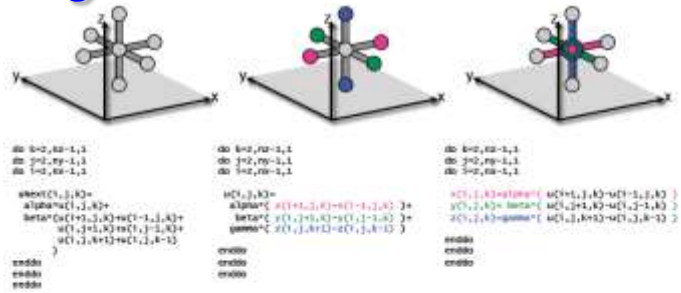
200km
Typical resolution of IPCC AR4 models

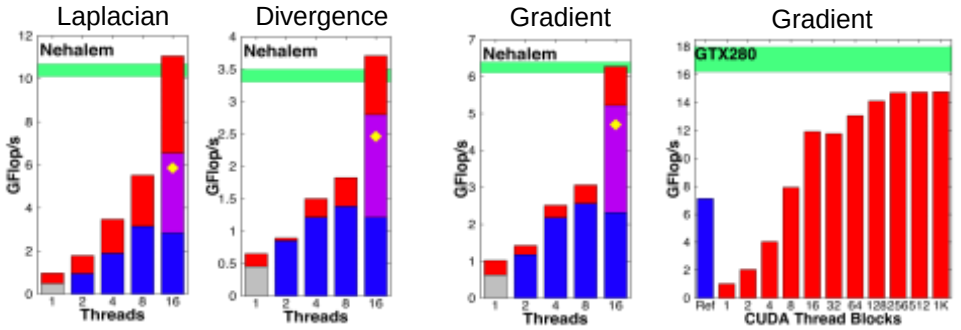


25km
Upper limit of climate models with cloud parameterizations



1km
Cloud system resolving models





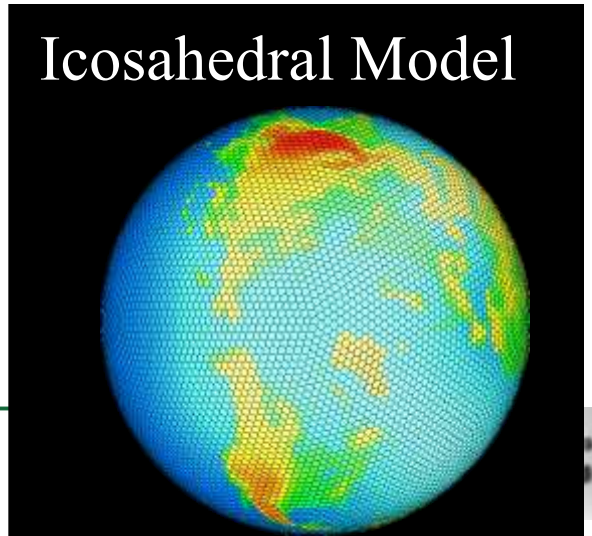
- Direct simulation of cloud systems replacing statistical parameterization.
  - This approach recently was called for by the 1st WMO Modeling Summit.

Demonstrated during SC '08

Proof of concept

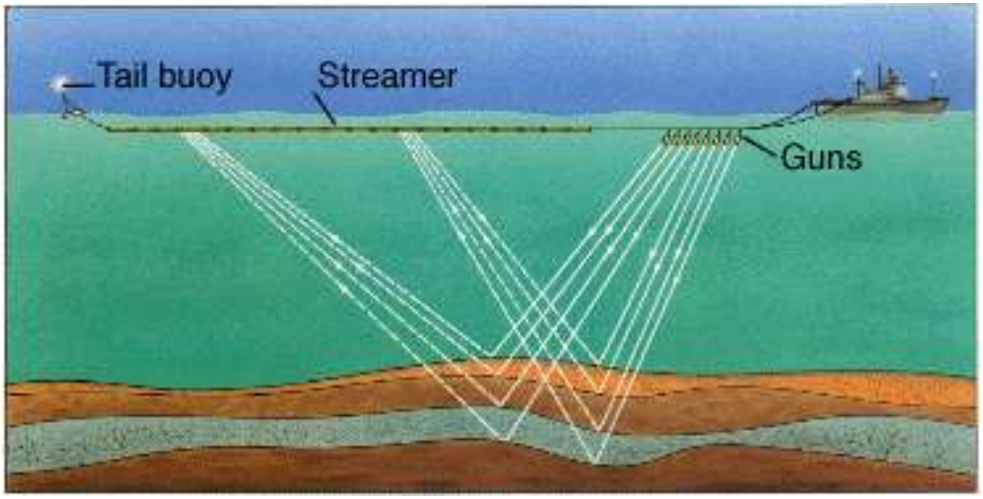
- CSU limited-area atmospheric model ported to Tensilica architecture
- Single Tensilica processor running atmospheric model at 50MHz

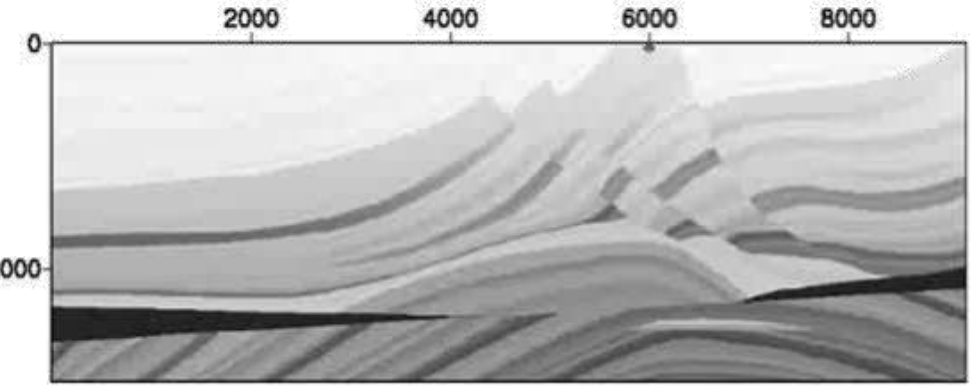
Actual code running - not representative benchmark



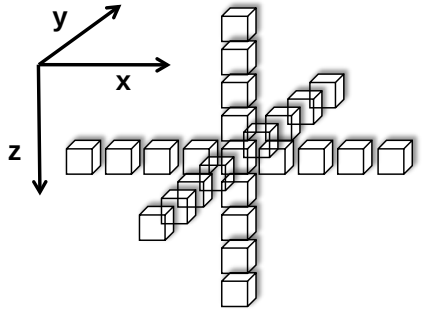
BERKELEY LAB

Application Driver:Seismic Imaging



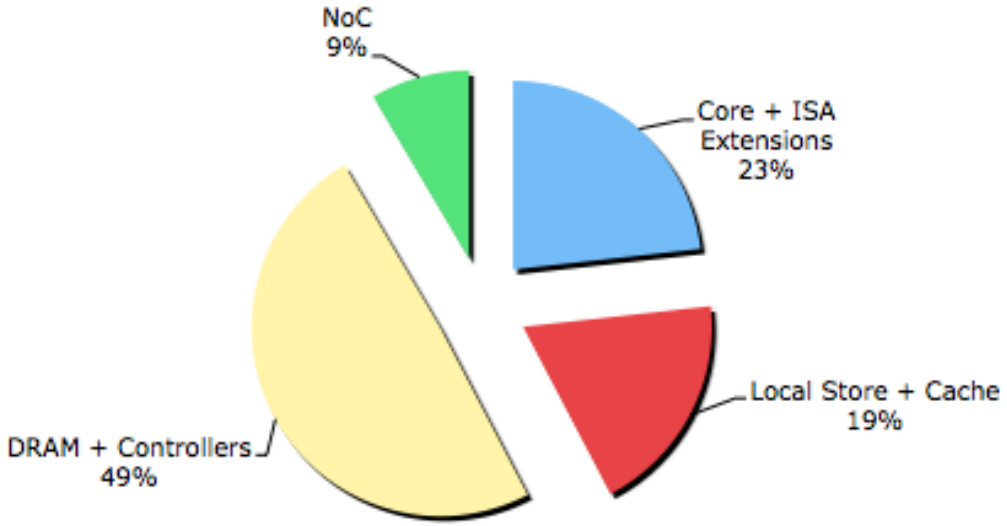


- **Seismic imaging used extensively by oil and gas industry**
  - Dominant method is RTM (Reverse Time Migration)
- **RTM models acoustic wave propagation through rock strata using explicit PDE solve for elastic equation in 3D**
  - High order (8th or more) stencils
  - High computational intensity
- **Typical survey requires months of computing on petascale-sized resources**

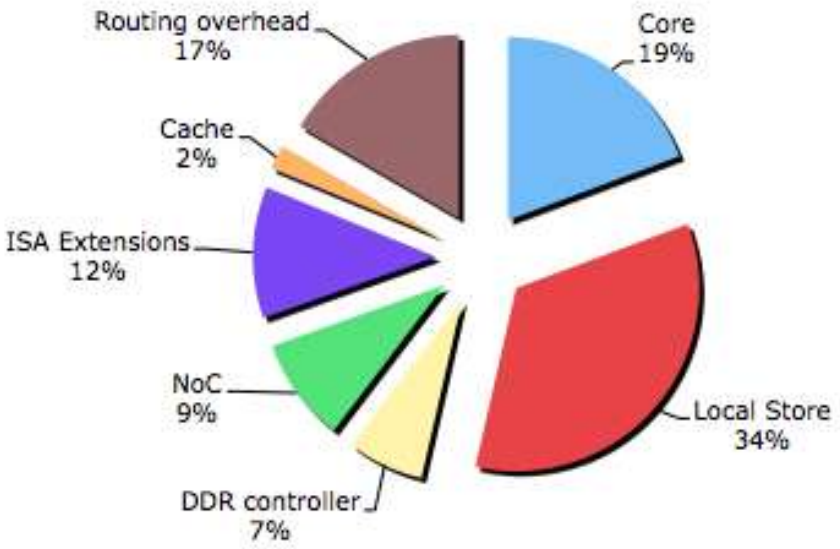


Green Wave ASIC Design
(power and area breakdown)

Power Breakdown
(70W total for SoC+ memory)



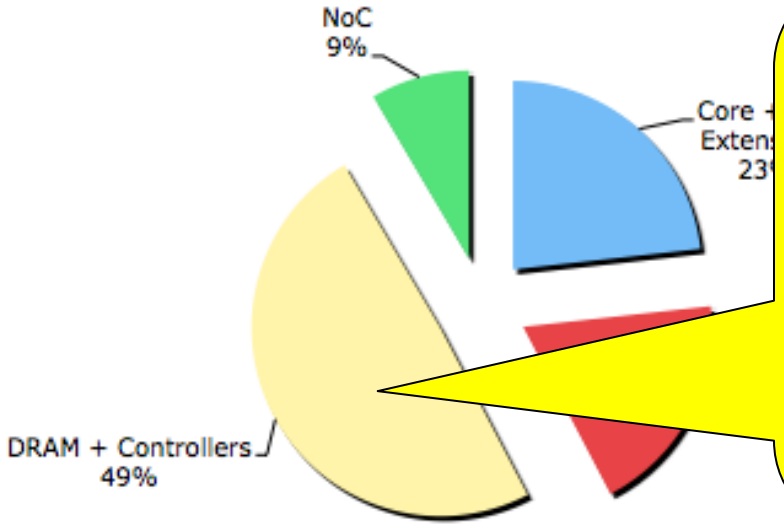
Area Breakdown
(240 mm² for SoC)



- **Developed RTL design for SoC in 45 nm technology using off-the-shelf embedded technology + simulated with RAMP FPGA platform**

Green Wave ASIC Design
(power and area breakdown)

Power Breakdown
(70W total for SoC+ memory)



Area Breakdown
(240 mm² for SoC)

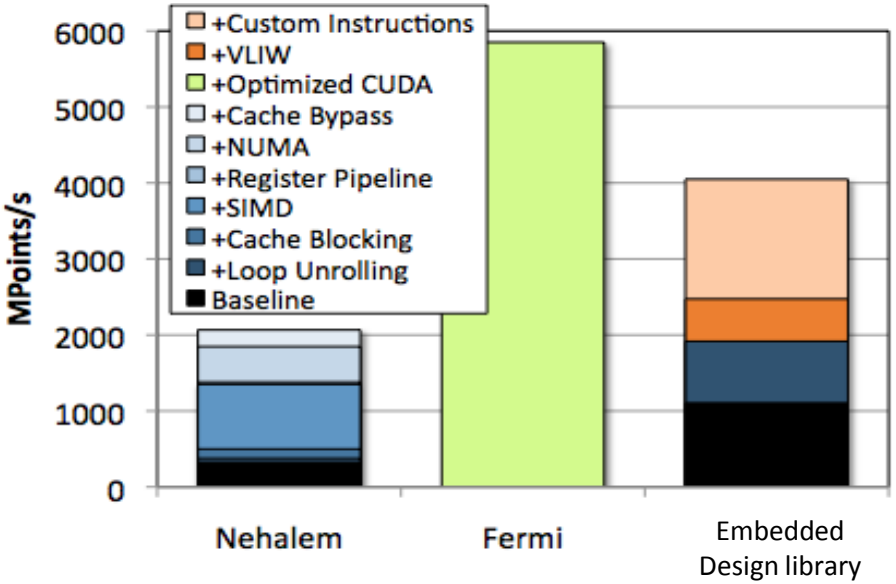
Can reduce this power fraction substantially using Micron Hybrid Memory Cube technology.

HMC-Gen2 = 15W device with 360+GB/s performance.

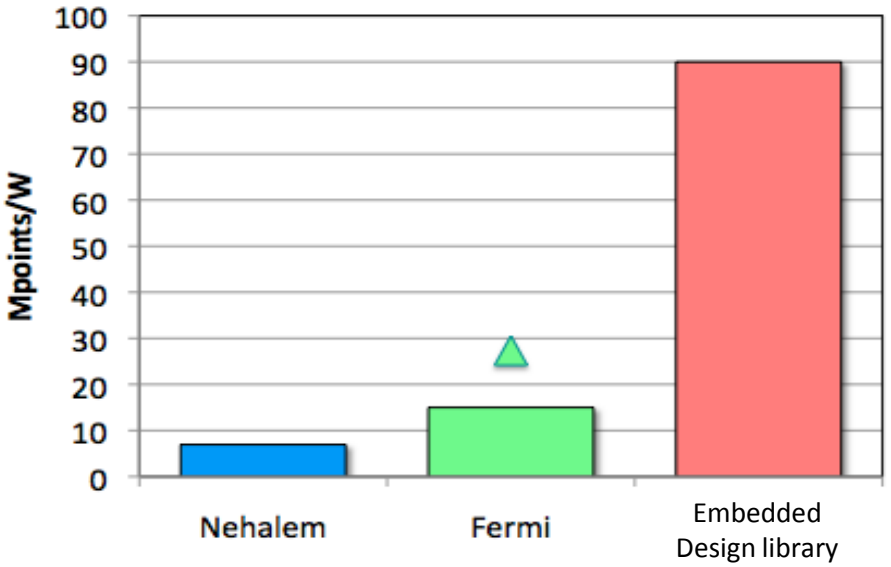
- **Developed RTL design for SoC in 45 nm technology using off-the-shelf embedded technology + simulated with RAMP FPGA platform**

Example Design Study Seismic Imaging

Performance



Energy Efficiency



We cannot touch an end-to-end engineered design?

but can get damned close.

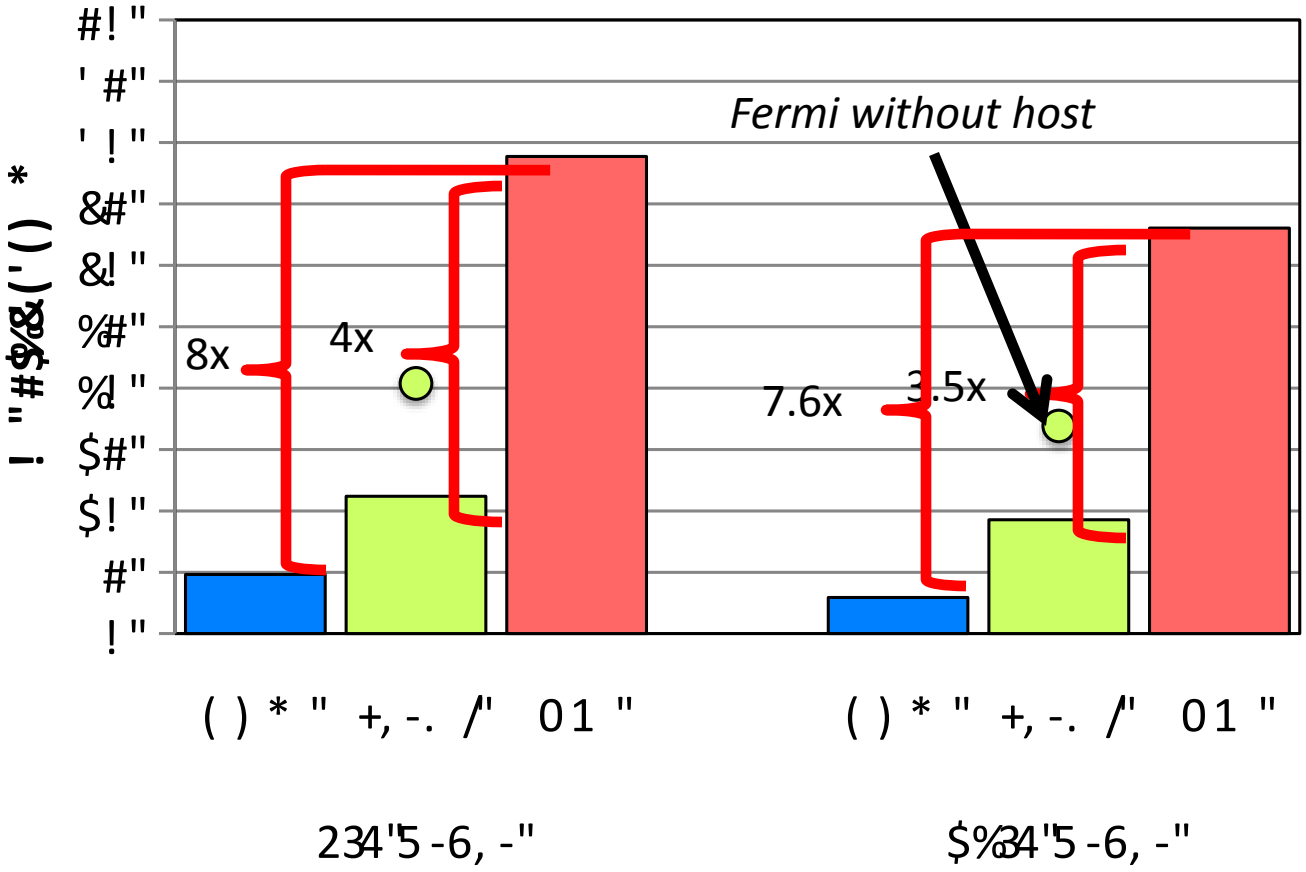
big win for efficiency from what is NOT included

Further improvements primarily constrained by the memory technology

BERKELEY LAB

Ter@tec

Green Wave Efficiency



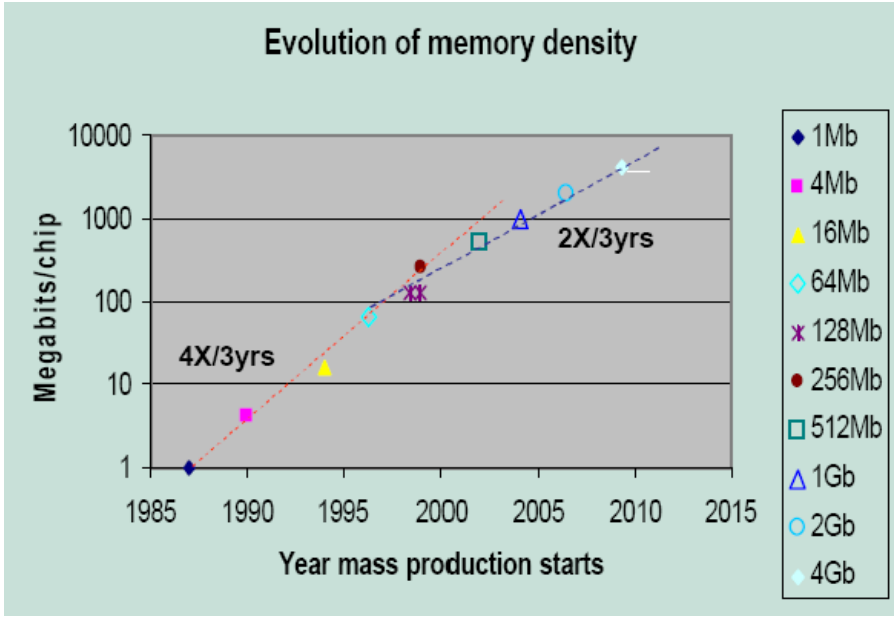
Take Home Message

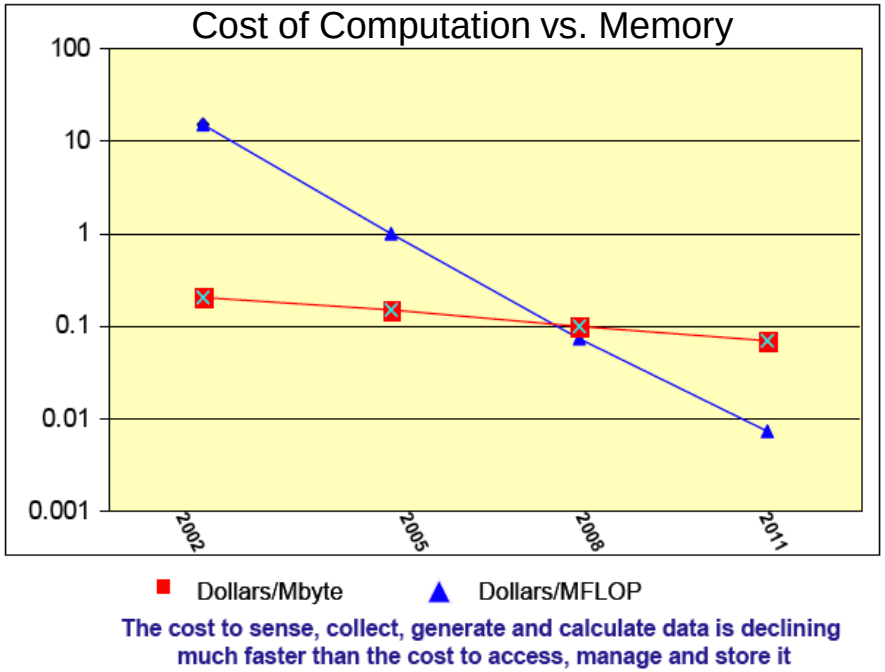
- **Primary Design Principle: Reduce waste**
  - **Biggest benefits were from what we did NOT include**
- **Focus on data movement**
  - needs hardware support that is lacking in current designs
- **Use design principles and technology**
  - **Low power cores**
  - **Rapid design prototyping tools**
  - **SoC**
  - **CoDesign**
- **CoDesign to get best Hardware/Software efficiency and integration**

Memory Technology

Projections of Memory Density Improvements

- **Memory density is doubling every three years; processor logic is every two**
  - Project 8Gigabit DIMMs in 2018
  - 16Gigabit if technology acceleration (or higher cost for early release)
- **Storage costs (dollars/Mbyte) are dropping gradually compared to logic costs**
  - Industry assumption: $1.80/memory chip is median commodity cost





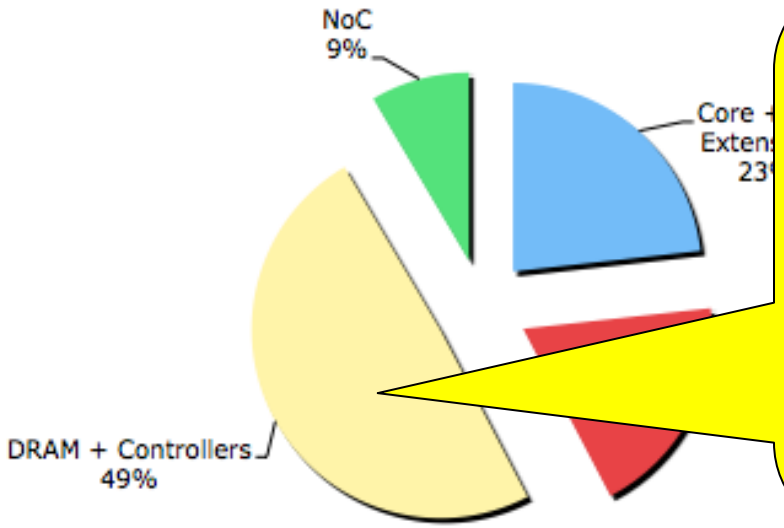
The cost to sense, collect, generate and calculate data is declining much faster than the cost to access, manage and store it

Source: David Turek, IBM

Memory Technology Bottleneck

Power Breakdown
(70W total for SoC+ memory)

Area Breakdown
(240 mm² for SoC)

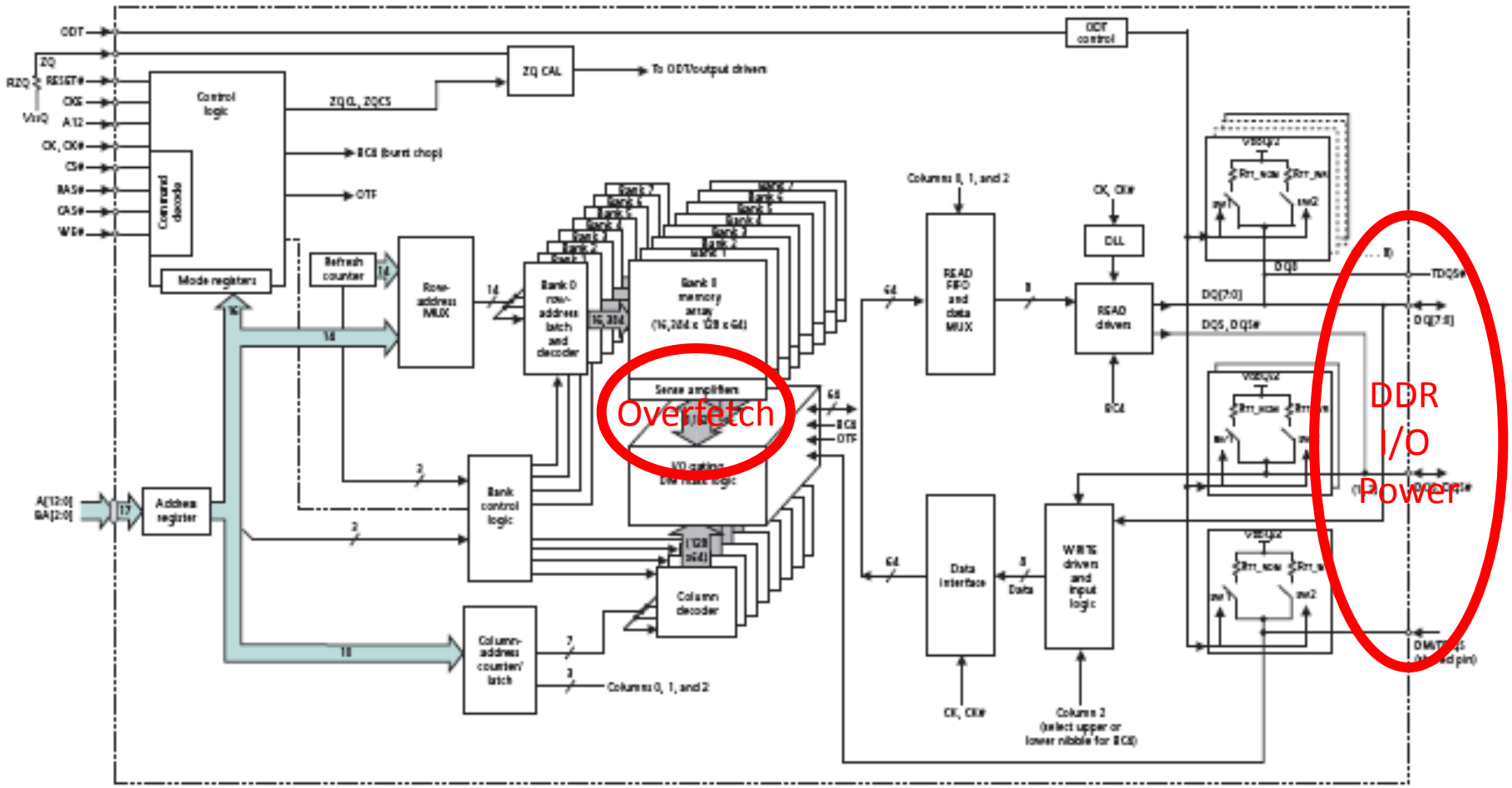


Can reduce this power fraction substantially using Micron Hybrid Memory Cube technology.

HMC-Gen2 = 15W device with 360+GB/s performance.

- **Developed RTL design for SoC in 45 nm technology using off-the-shelf embedded technology + simulated with RAMP FPGA platform**

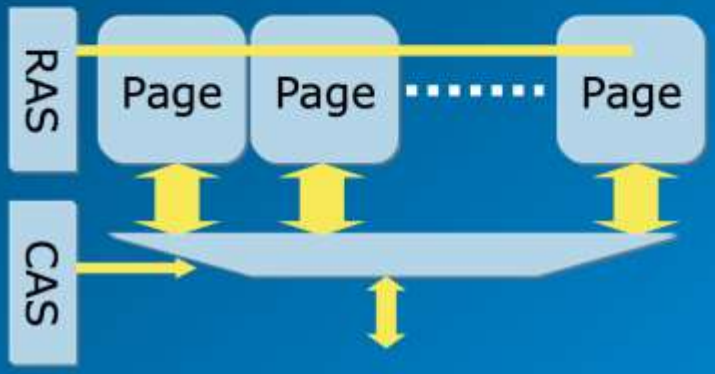
1Gbit DDR Memory Architecture



Slide from Dean Klein (Micron Technology)

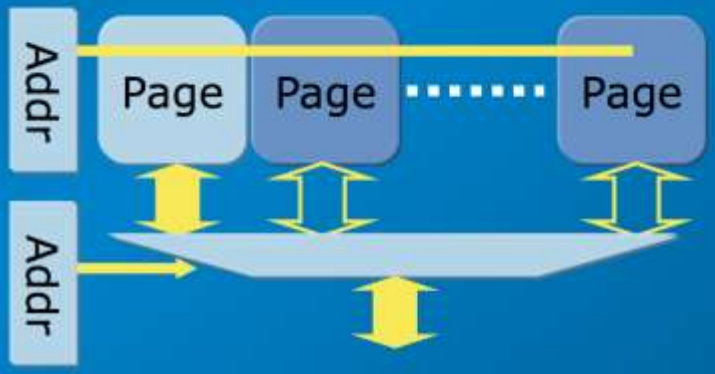
Revise DRAM Architecture

Traditional DRAM



Activates many pages
Lots of reads and writes (refresh)
Small amount of read data is used
Requires small number of pins

New DRAM architecture



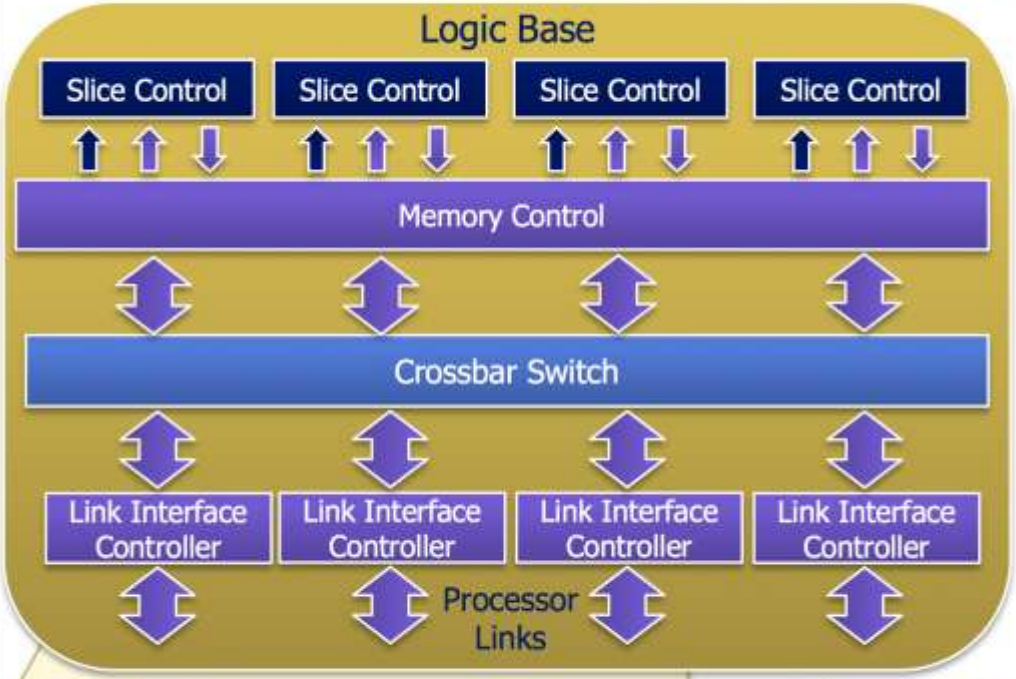
Activates few pages
Read and write (refresh) what is needed
All read data is used
Requires large number of IO's (3D)

150 pJ/bit
(including Memory Controller)

8-10 pJ/bit

Need to bring it down to 2 pJ/bit

HMC Architecture

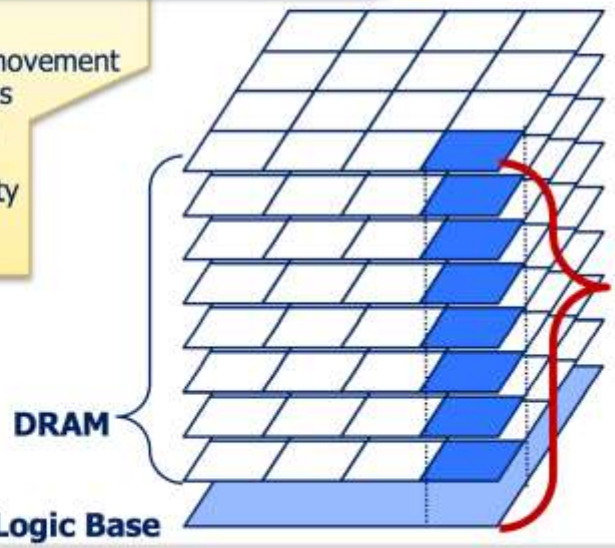


Logic Base

- Wide, high-speed local bus for data movement
- Advanced memory controller functions
- DRAM control at memory rather than distant host controller
- Reduced memory controller complexity and increased efficiency

Add sophisticated switching and optimized memory control…

And now we have a whole new set of capabilities

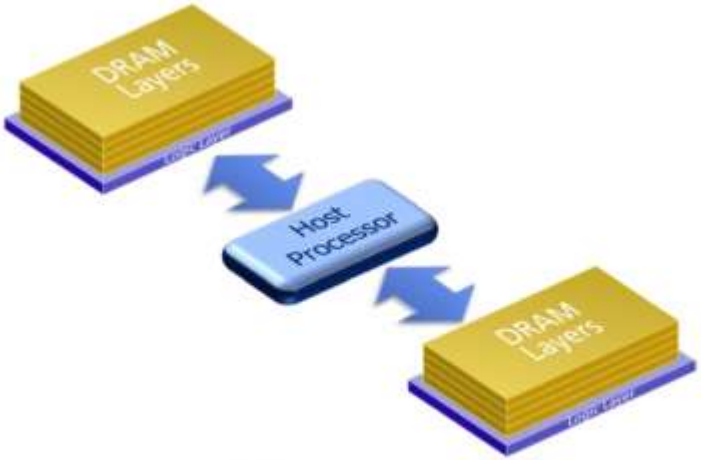


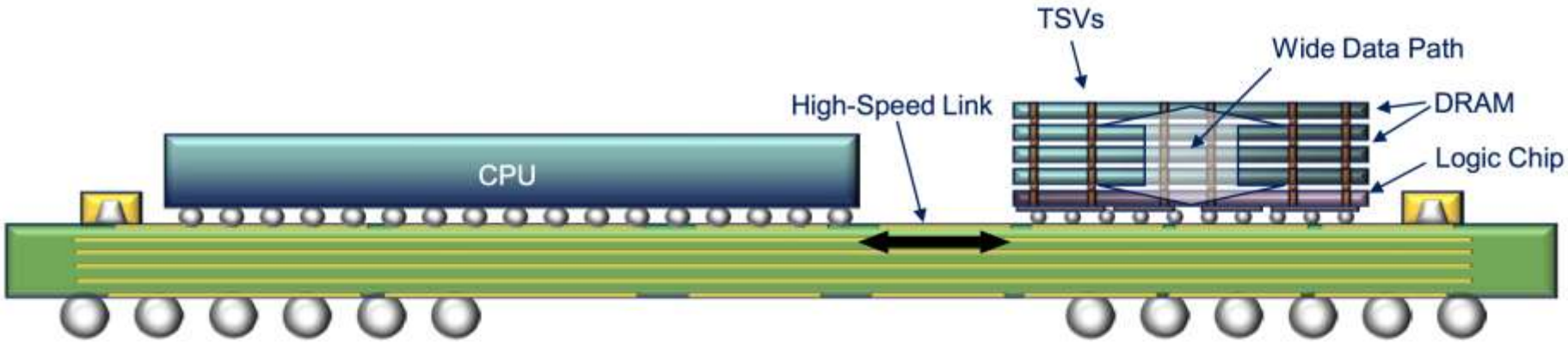
Vertical Slices are managed to maximize overall device availability

- Optimized management of energy and refresh
- Self test, error detection, correction, and repair in the logic base layer

Silicon Interposer

- All links are between host CPU and HMC logic layer
- Maximum bandwidth per GB capacity





Notes: MCM = multi-chip module
Illustrative purposes only; height is exaggerated

Memory Technology

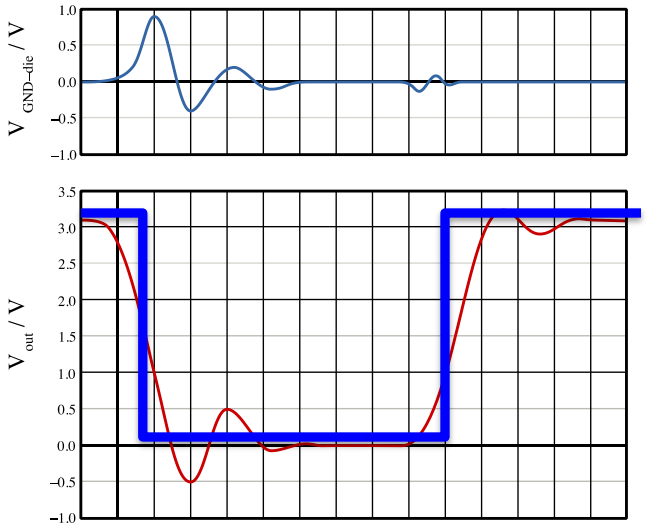
- **Overall, 4x+ improvements in efficiency are within our grasp**
- **Keeps pressure back on other elements of system design**

Active Power Management

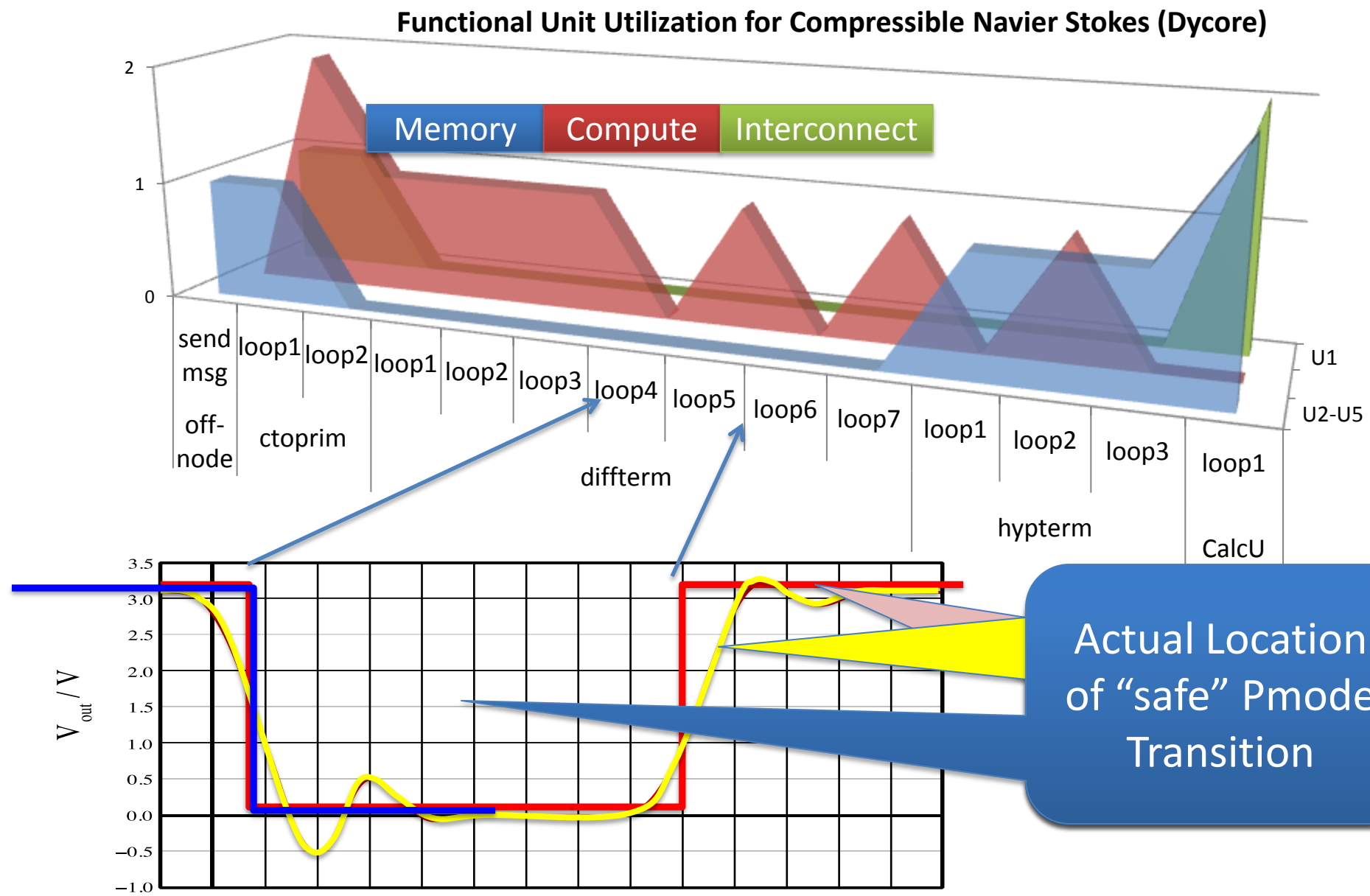
A few observations

Active Power Management

- **Principle of operation: Use DVFS to save power if CPU is underutilized**
  - Lower clock frequency
  - Then lower Vdd
  - Benefit is cubic ($V^2 * F$)
- **Why is it so hard**
  - Settling clock frequency (PLLs)
  - Voltage transients (ground bounce)
- **Results**
  - Yesterday: DVFS pstate change ~10k cycles
  - Today: DVFS pstate changes ~1k cycles
  - Future: developments could enable changes in ~100 cycles with finer granularity (this would be a huge win for active pwr savings)



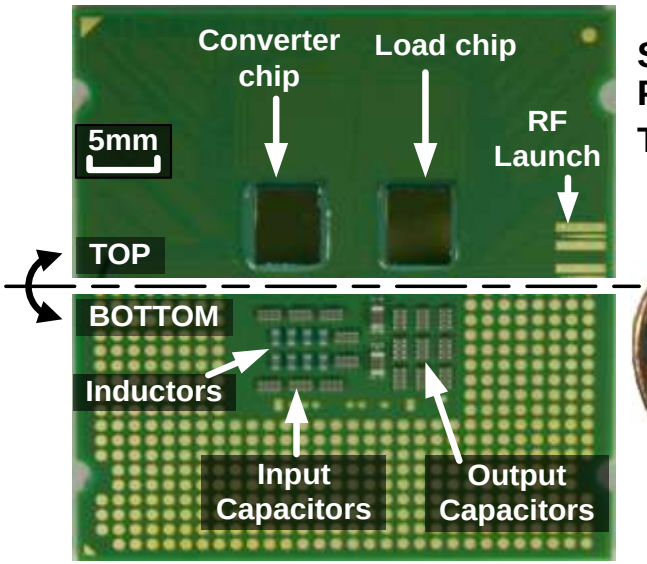
Diversity of FU Utilization is Opportunity for DVFS



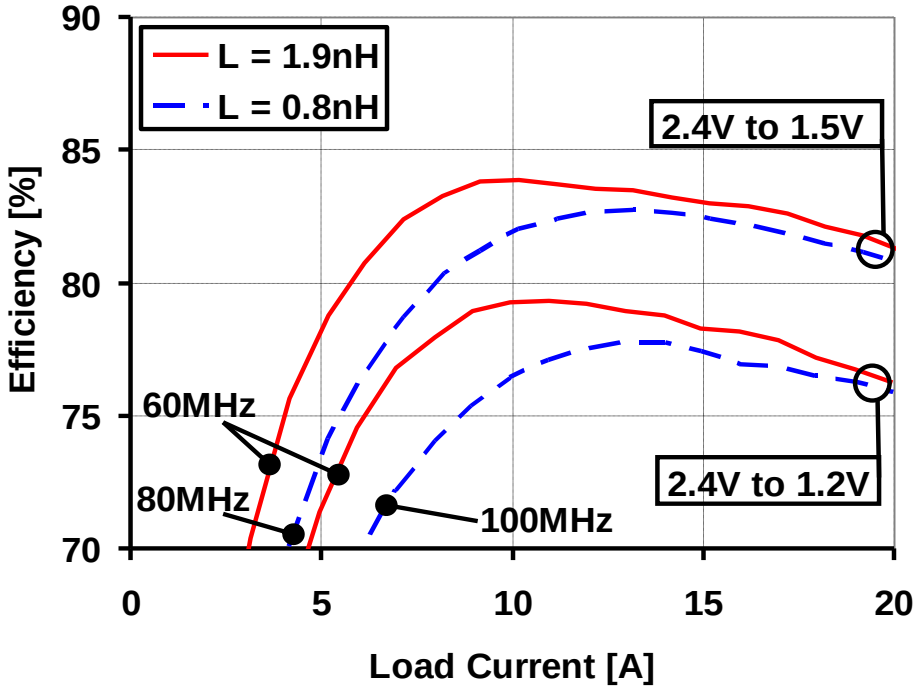
Integration of Power Delivery to reduce Ground Bounce Increase responsiveness and Efficiency

For efficiency and management

Integrated Voltage Regulator Testchip



Standard OLGA Packaging Technology



Power delivery closer to the load for

1. **Improved efficiency**
2. **Fine grain power management**

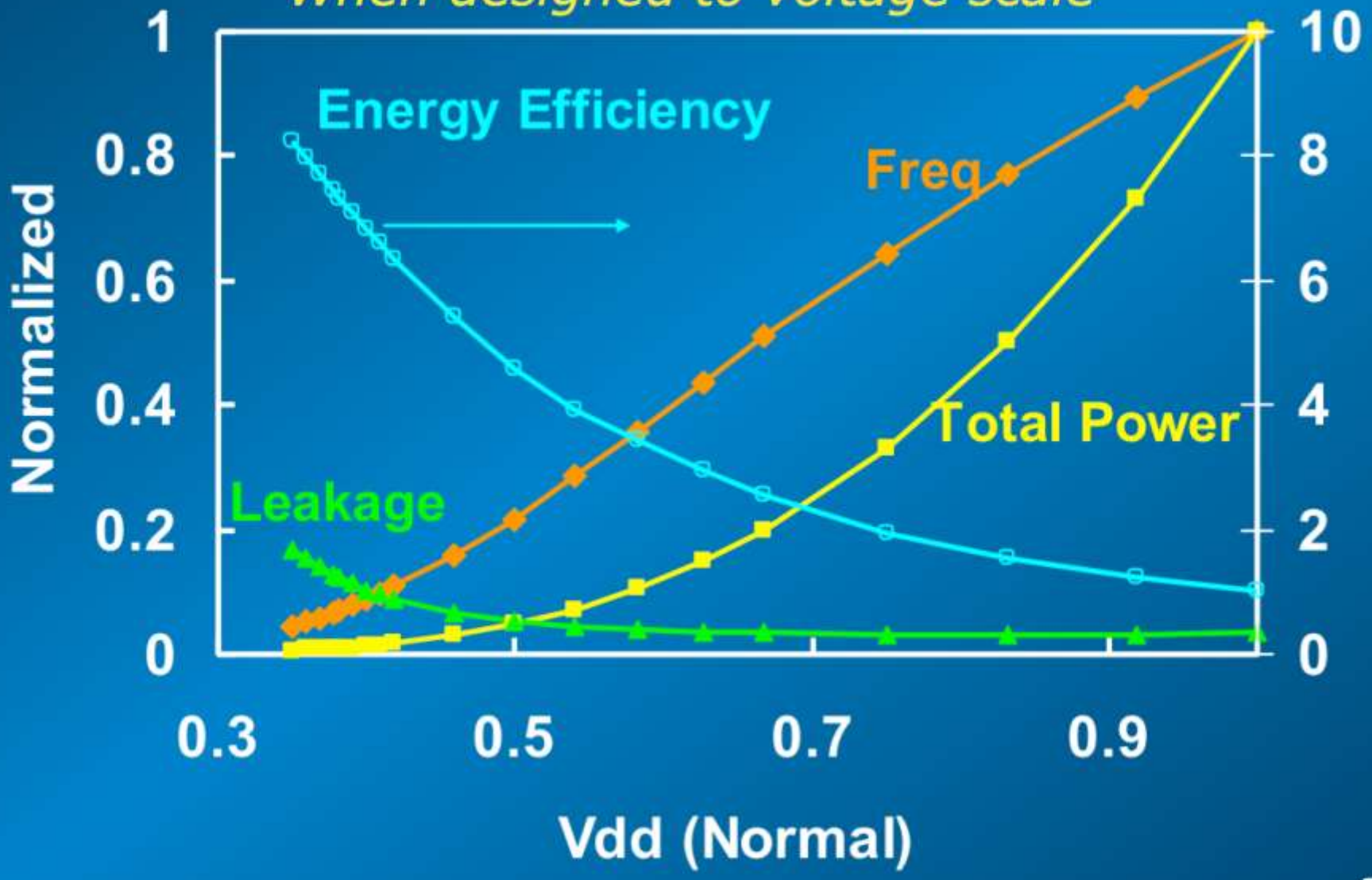
Design Consequences

- **Moving from 1000 cycles to Pstate change to 100s of cycles**
  - Finer grained
  - Faster Transitions (less hysteresis)
- **But for software control can we make reasonable "fine grained" decisions in < 100 clock cycles?**
  - **Optimal control theory** says you cannot have a control system that responds slower than item you want to control
  - Unstable system if software is too slow
  - Only thing fast enough is hardware
- **My viewpoint: Need to move towards policy-based mechanisms for power control**
  - **Today's Imperative mechanisms** allow you to query power counters and write to change states actively
  - **Policy-based:** Ask hardware to lower power state under some condition (need to get software out of the critical path)

Shekhar Borkar: Intel

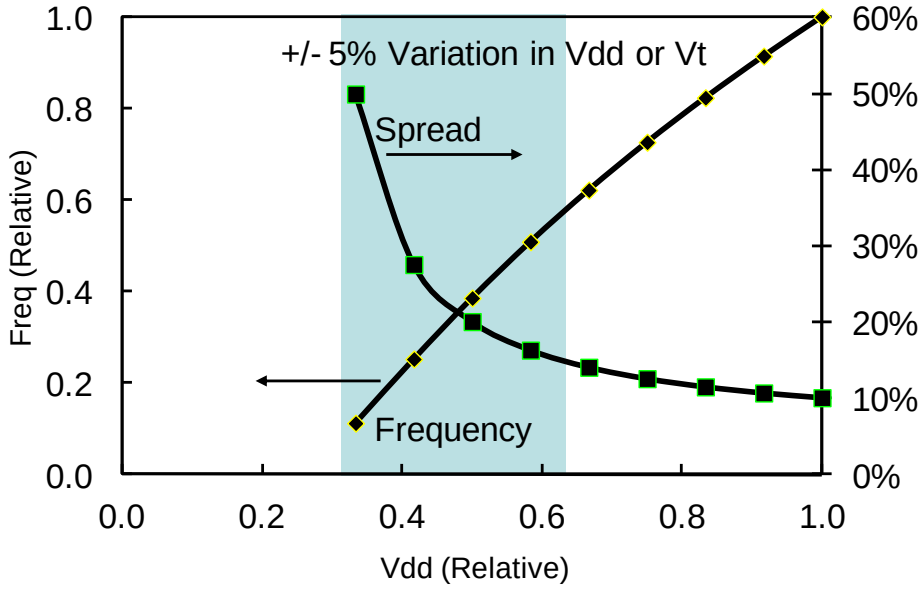
Voltage Scaling

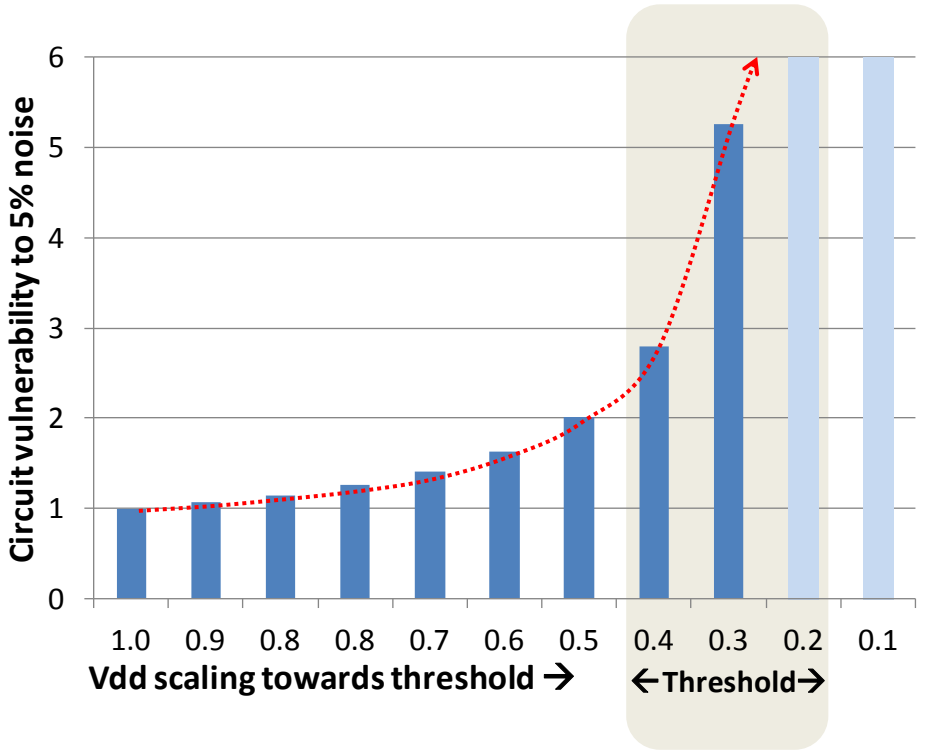
When designed to voltage scale



Impact of Variation on NTV

Shekhar Borkar: Intel





$$frequency \propto \frac{(V_{dd} - V_t)^{\alpha}}{V_{dd}}$$

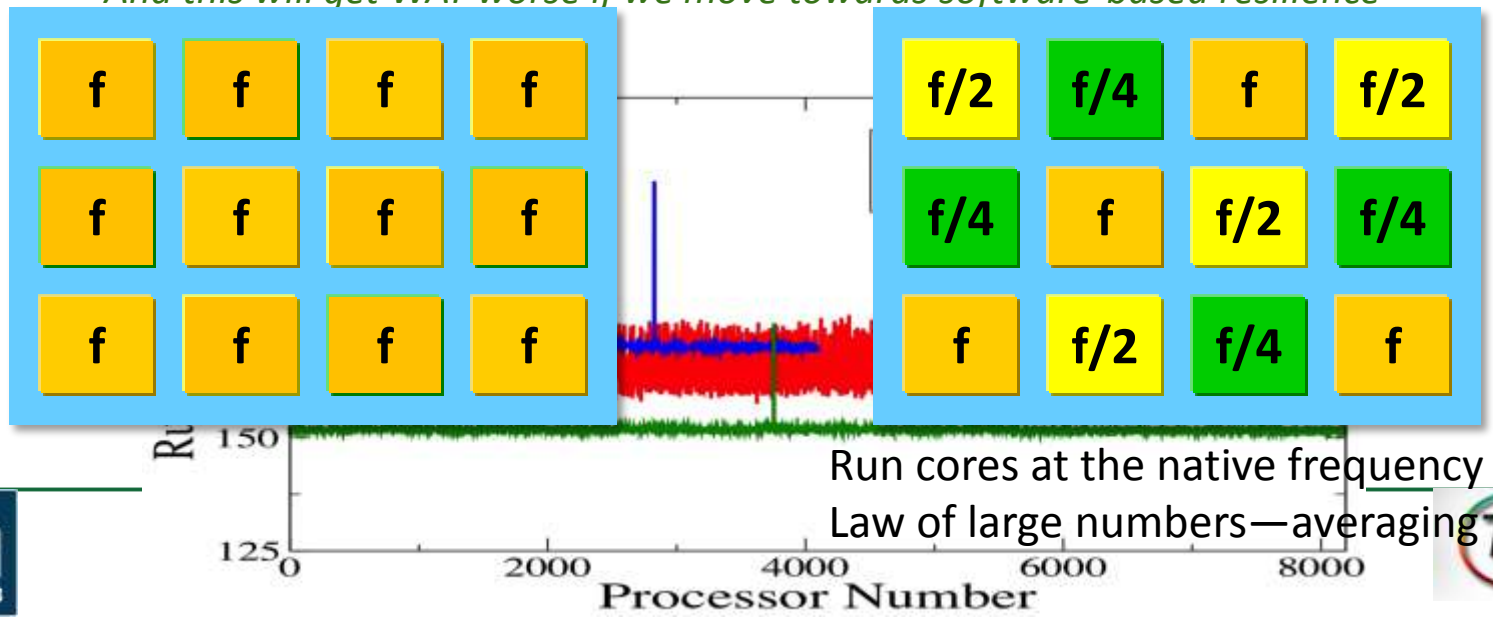
5% variation in Vt or Vdd results in 20 to 50% variation in circuit performance

Assumptions of Uniformity is Breaking

Power Management among many sources of speed nonuniformity

- **Heterogeneous compute engines (hybrid/GPU computing)**
- **Irregular algorithms**
- **Fine grained power mgmt. makes homogeneous cores look heterogeneous**
  - *thermal throttling on Sandybridge – no longer guarantee deterministic clock rate*
- **Nonuniformities in process technology creates non-uniform operating characteristics for cores on a CMP**
- **Fault resilience introduces inhomogeneity in execution rates**
  - *error correction is not instantaneous*
  - *And this will get WAY worse if we move towards software-based resilience*

Example: Many-core System



Run cores at the native frequency

Law of large numbers—averaging

Observations on Systemwide Power Management

- **Best opportunities for increasing efficiency of Power Management**
  - Fine grained (individual functional units)
  - Faster (100 cycles instead of 10,000)
  - Implies very tight control loop
- **Locally Optimal for HPC may be Globally Deficient**
  - Communicating control decisions at system level take 100k-1M cycles
  - **Solution 0 (baseline)**: don't use fine grained power management (unacceptable)
  - **Solution 1**: Use policy based mechanisms
  - **Solution 2**: Depart from bulk synchronous model for computation

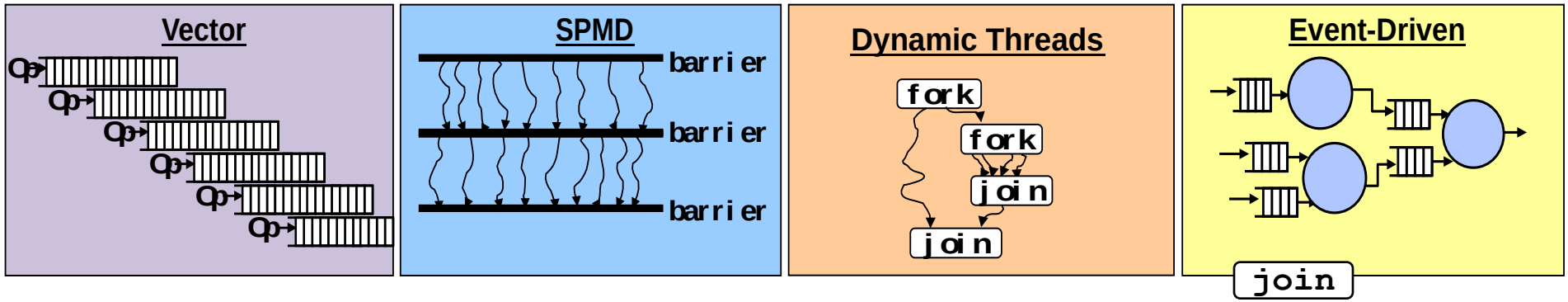
Where are We Today

- **Addicted to Bulk-Sync/SPMD Programming Model**
  - Low Cognitive Load
  - Everyone does the same thing at *appoximately* the same time
  - Data and control hazards are isolated to epochs of code execution *(not all possible interleavings of threads described in "The Problem with Threads")*

- **SPMD Models Have Demanding Requirements for Hardware/Software Ecosystem**
  - Demands homogeneous execution rates
    - Homogeneous performance per core (control OS "Noise" and code-your-own load balancing for adaptive algorithms)
  - Over-provision interconnect bandwidth for episodic/flood communication
  - Fast sync/collective operations (BG collective network)
    - Has similarity to instruction bcast for SIMD
  - Exhausting Sources of parallelism through domain decomposition
  - Gravitate towards bulk-sync communication
    - To make it easier to reason about control flow/messaging hazards
    - Creates episodic floods of interconnect traffic
    - *try to mitigate by getting overlap*

Re-Examining Execution Models

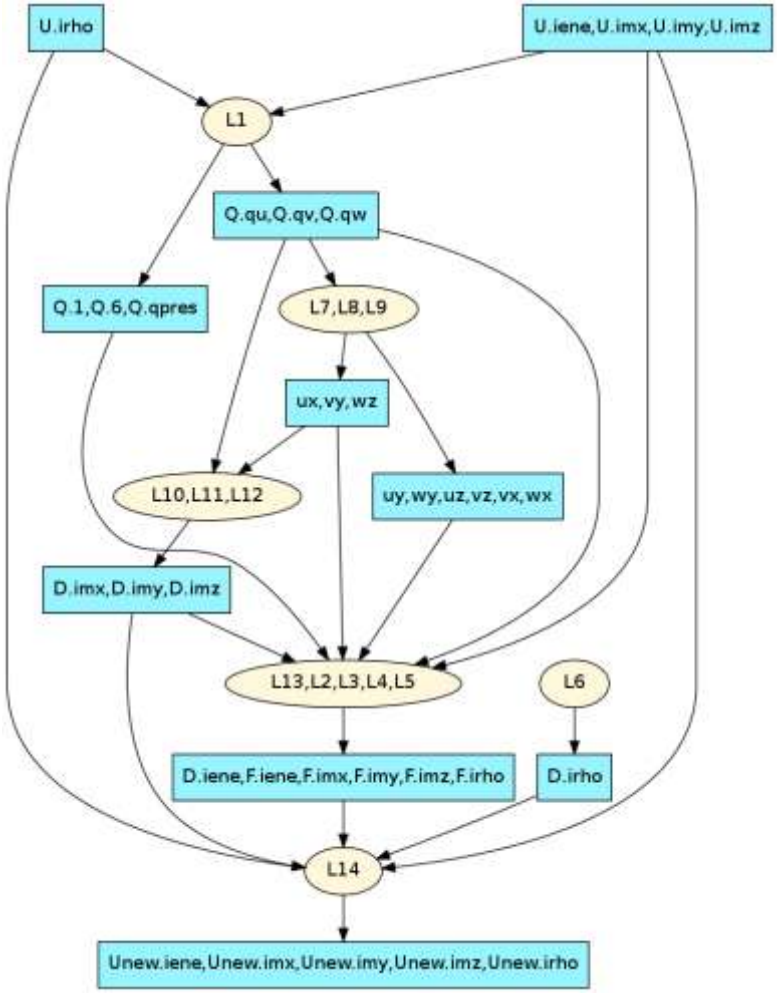
Examples of parallel execution models



- **What is the parallelism model for multicore *(exascale)*?**
  - Must balance *productivity* and *implementation efficiency*
- **Is the number of processors exposed in the model**
  - HPCS Language thrust: can we virtualize the processors?
- **How much can be hidden by compilers, libraries, tools?**
- **Re-examining old paradigms using modern methods**
  - ETI Swarm, HPX/ParalleX, Charm++, Intel Traleika Glacier

ParLab

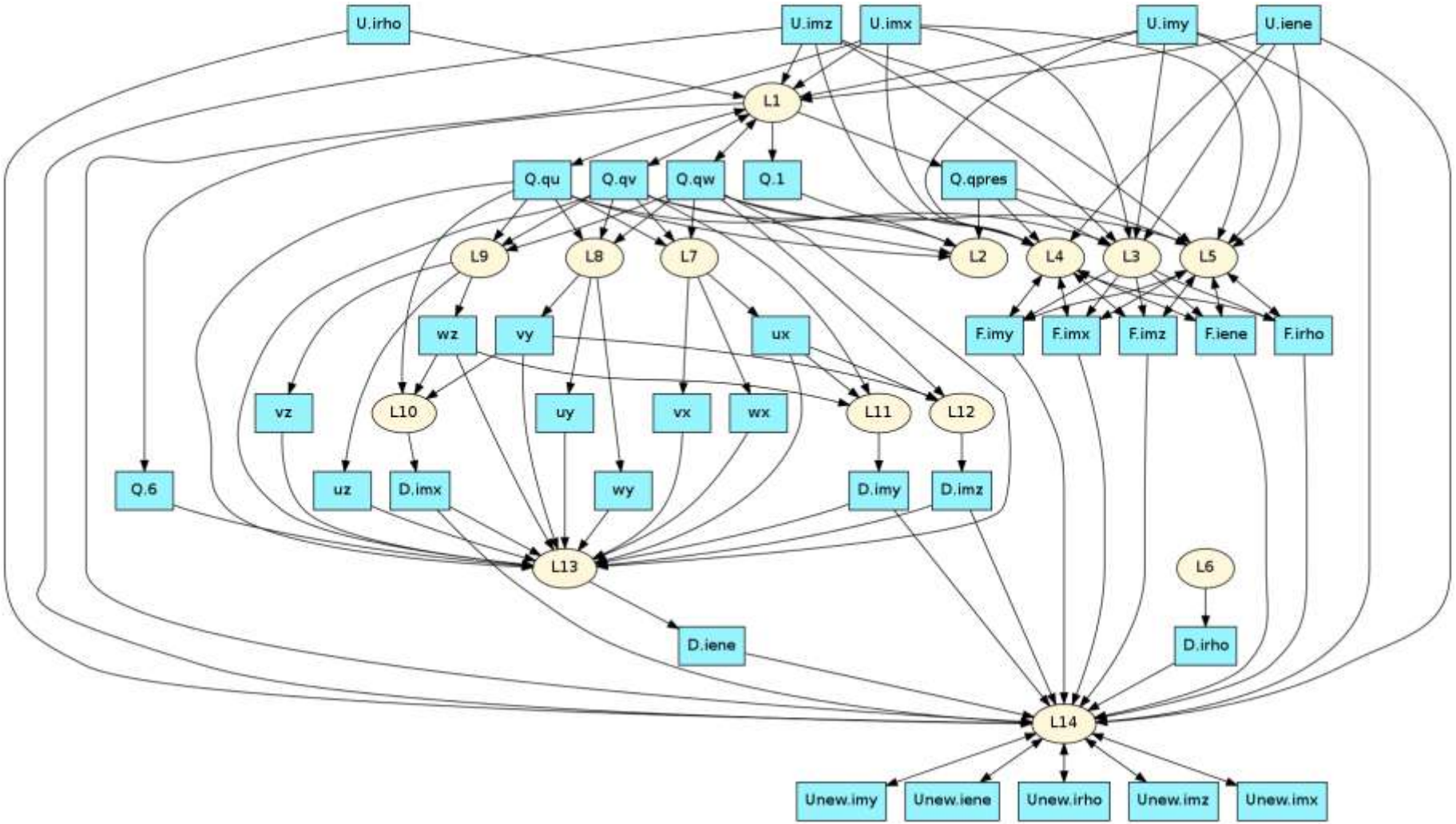
Dataflow Dependency Graph Analysis



Partially optimized graph

- Automated dependency graph manipulation
- Explore opportunities to extract extra concurrency by executing work items concurrently
- Dynamic Runtime to load balance among tasks and processors
- Semantics of CSP and C/Fortran base languages do NOT allow modern programming systems to automate these optimizations

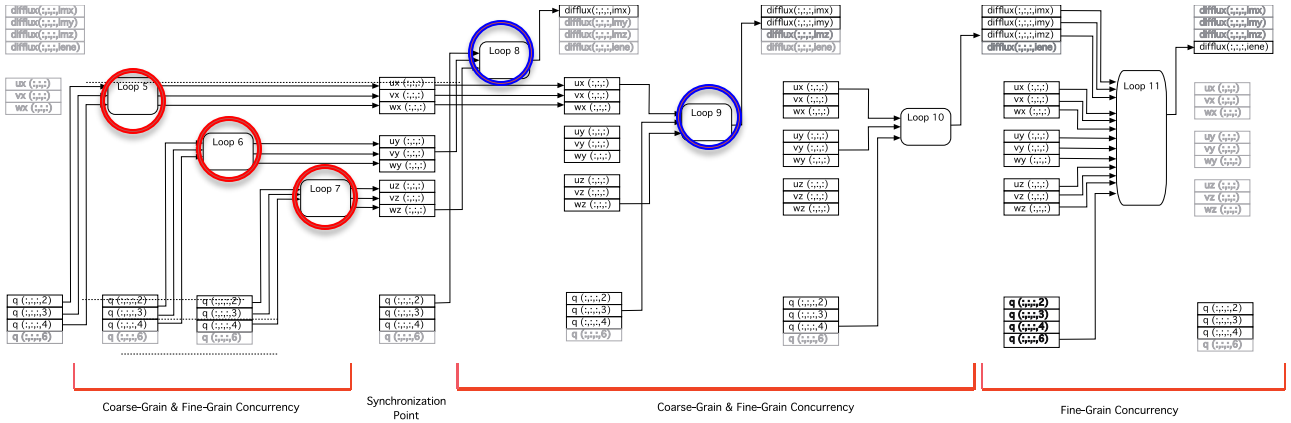
Raw Dependency Graph for CNS



Async Models to Reduce Energy to Solution

CNS Computational Kernels Serialized (*starvation*)

diffterm subroutine

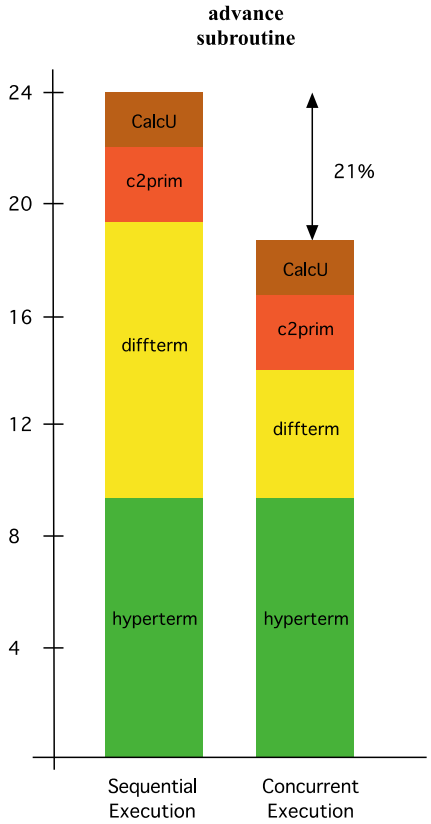


Code:

- Loop nests match “math”
- Sequence of “projections”
- Miss coarse-grain data dependences (see above)
  - 2.5x speedup of **diffterm**; 20% speedup for **advance** subroutine
  - Requires OpenMP task-level directives

Metrics:

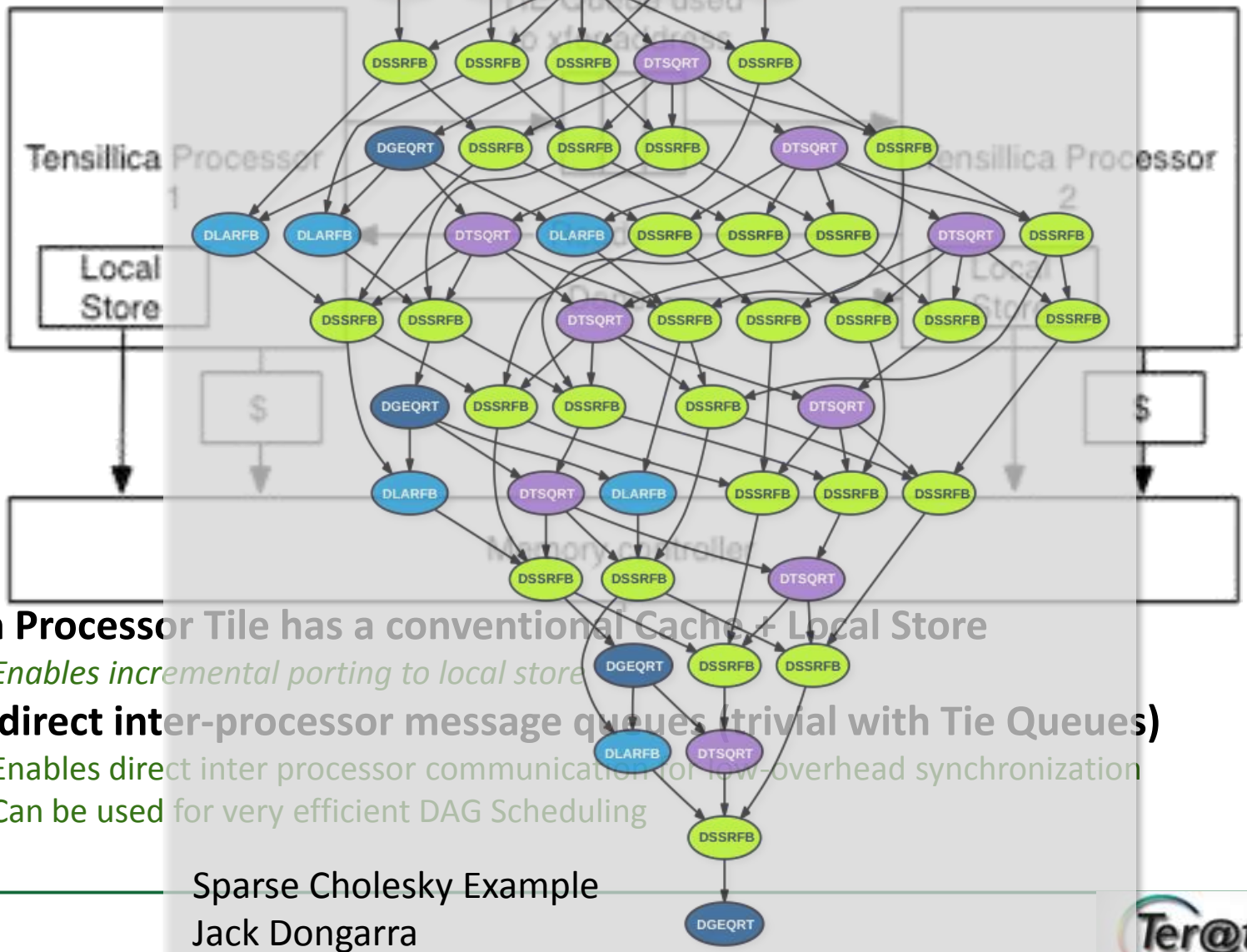
- Good cache behavior (≈98% L1 hit rate)
- Good FP/INT Instructions ratio: 2:1 to 3:1



BERKELEY LAB

Ter@tec

Providing Direct Support for DAG Scheduling and Data Locality Control



This is just utilizing off-the-shelf technology from embedded space!

- **Each Processor Tile has a conventional Cache + Local Store**
  - *Enables incremental porting to local store*
- **Has direct inter-processor message queues (trivial with Tie Queues)**
  - Enables direct inter processor communication for low-overhead synchronization
  - Can be used for very efficient DAG Scheduling

Sparse Cholesky Example
Jack Dongarra

Conclusions on Heterogeneity

- **Sources of performance heterogeneity increasing**
  - Heterogeneous architectures (accelerator)
  - Thermal throttling
  - Performance heterogeneity due to transient error recovery

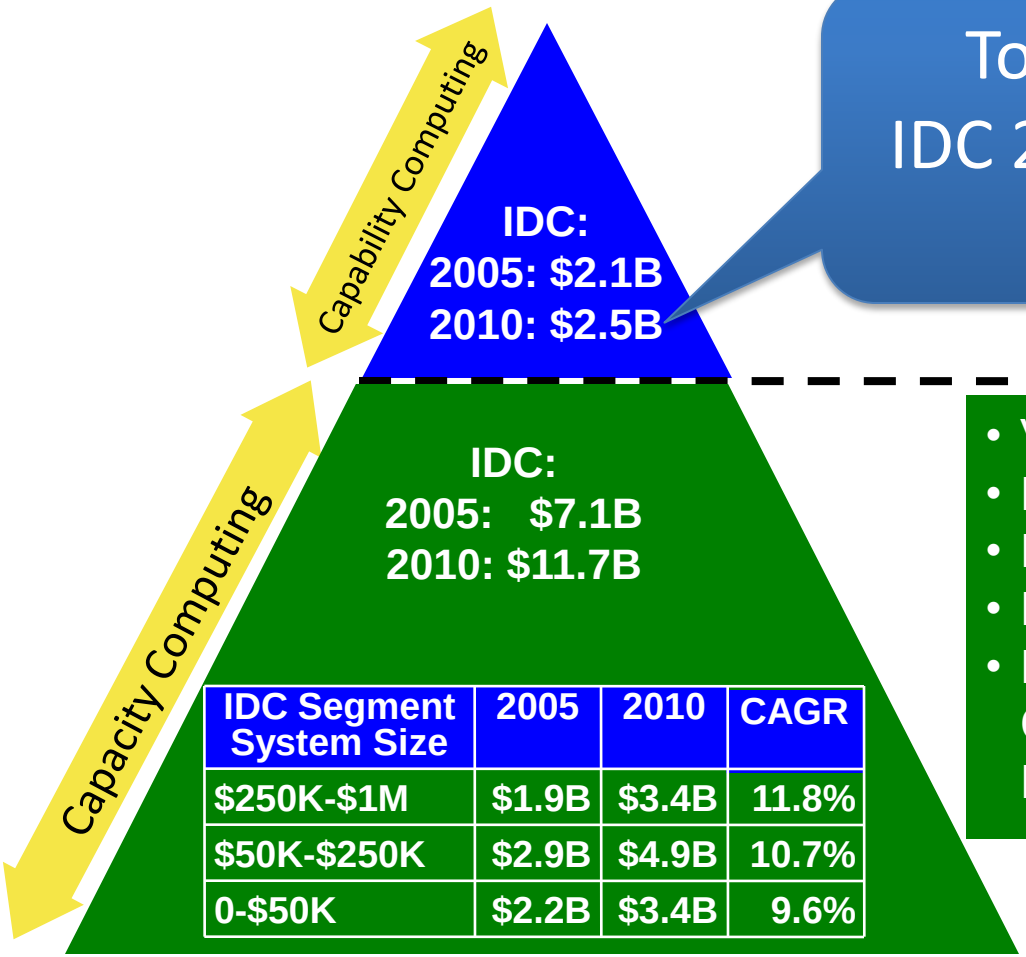
- **Current Bulk Synchronous Model not up to task**
  - Current focus is on removing sources of performance variation (jitter), is increasingly impractical
  - Huge costs in power/complexity/performance to extend the life of a purely bulk synchronous model

Embrace performance heterogeneity: Study use of asynchronous computational models (e.g. SWARM, HPX, Trailaika Glacier and other concepts from 1980s)

FIN!

HPC Market Overview

Mark Seager LLNL



IDC Segment System Size	2005	2010	CAGR
$250K-$1M	$1.9B	$3.4B	11.8%
$50K-$250K	$2.9B	$4.9B	10.7%
0-$50K	$2.2B	$3.4B	9.6%

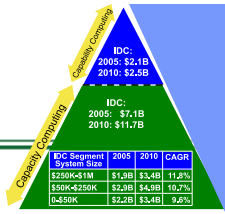
Totally Bogus Prediction
IDC 2010 puts HPC market at $10B

- Volume Market
- Mainly capacity; <~150 nodes
- Mostly clusters; >50% & growing
- Higher % of ISV apps
- Fast growth from commercial HPC; Oil &Gas, Financial services, Pharma, Aerospace, etc.

Total market >$10.0B in 2006
Forecast >$15.5B in 2011

HPC is built with of pyramid investment model

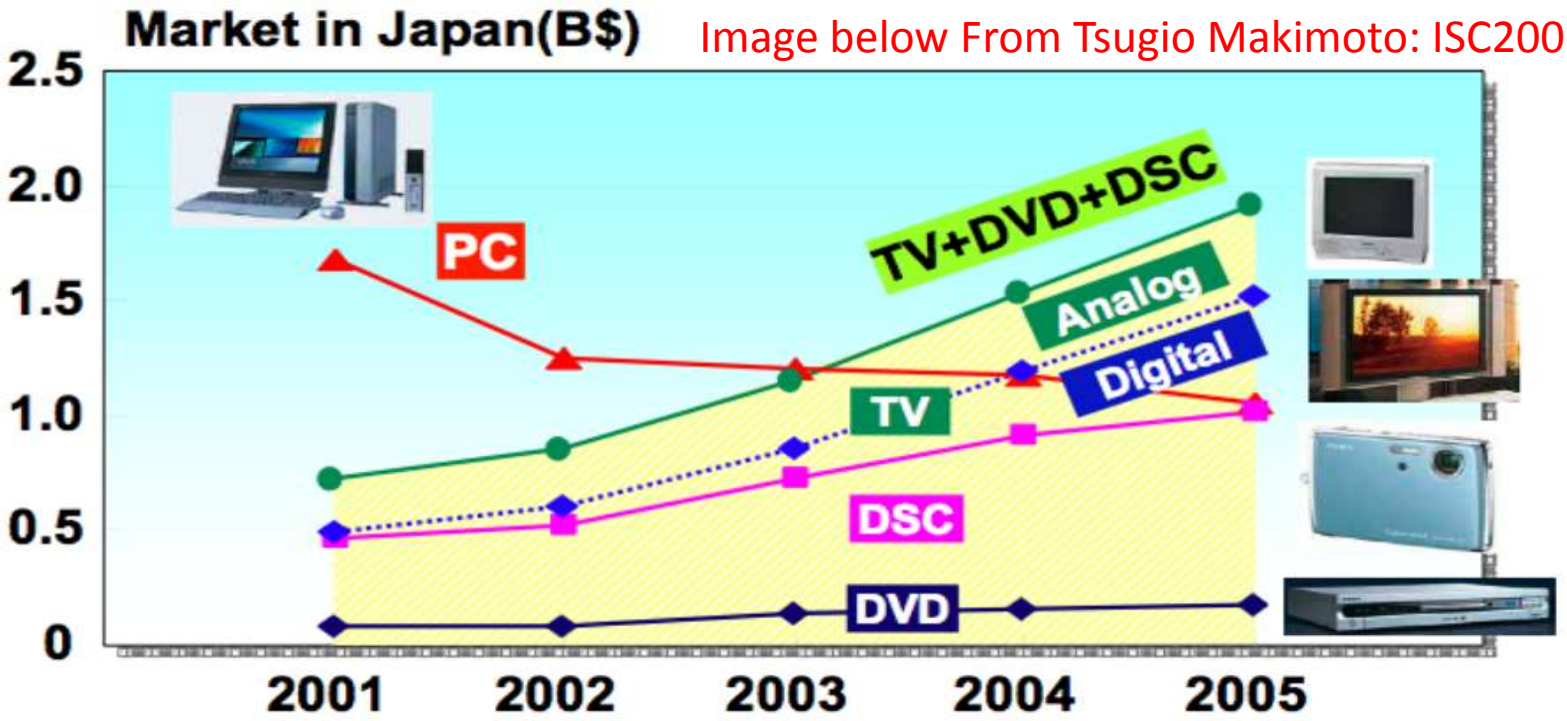
Technology Investment Trends



- **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/consumer processor technology for future HPC systems

Image below From Tsugio Makimoto: ISC2006



Redefining "commodity"

- **Must use "commodity" technology to build cost-effective design**
- **The primary cost of a chip is development of the intellectual property**
  - Mask and fab typically 10% of NRE in embedded
  - Design and verification dominate costs
  - SoC's for high perf. consumer electronics is vibrant market for IP/circuit-design (pre-verified, place & route)
  - Redefine your notion of "commodity"!

The 'chip' is not the commodity…

The stuff you put on the chip is the commodity

Embrace Embedded: Embedded / HPC Synergy

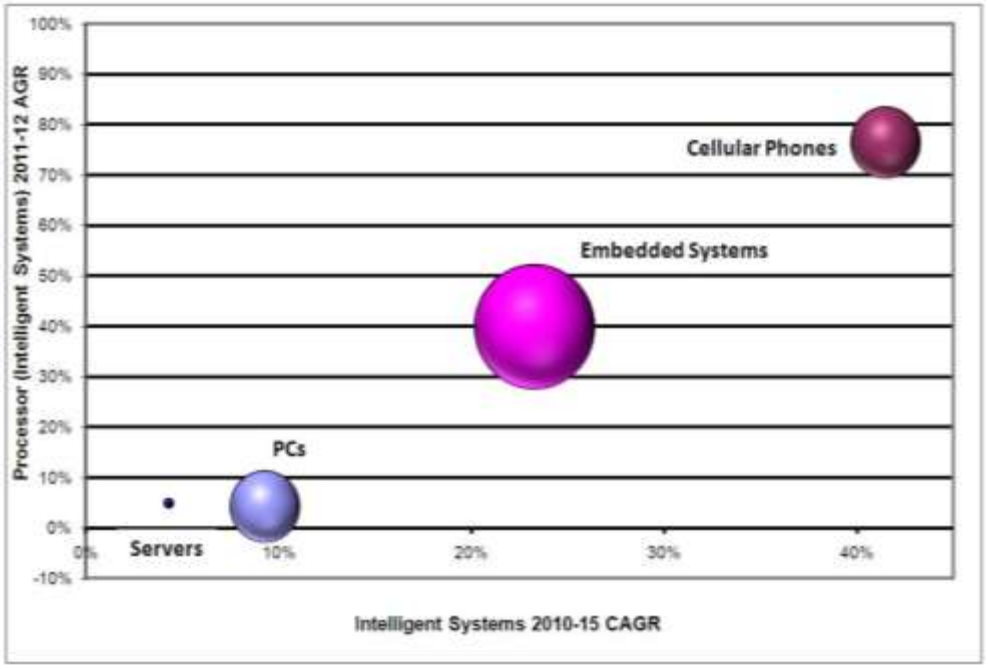
- **High Performance embedded is aligned with HPC**
  - HPC used to be performance without regard to power
  - Now HPC is power limited (max delivered performance/watt)
  - Embedded has always been driven by max performance/watt (max battery life) and minimizing cost ($1 cell phones)
  - Now HPC and embedded requirements are aligned

- **Your "smart phone" is driving technology development**
  - Desktops are no longer in the drivers seat
  - This is not a bad thing because high-performance embedded has longer track record of application-driven design
  - Hardware/Software co-design comes from embedded design
- **Changing notion of commodity (vibrant embedded IP market)**
  - ***Primary cost of chip is in IP blocks (not the mask and fab costs)***
  - ***The CHIP is not the commodity… the circuits ON the chip are the commodity***
  - ***IP blocks == silicon circuit board***

IDC 2010 Market Study

Embedded/Tiny Cores on SOC is aligned with Market Trends

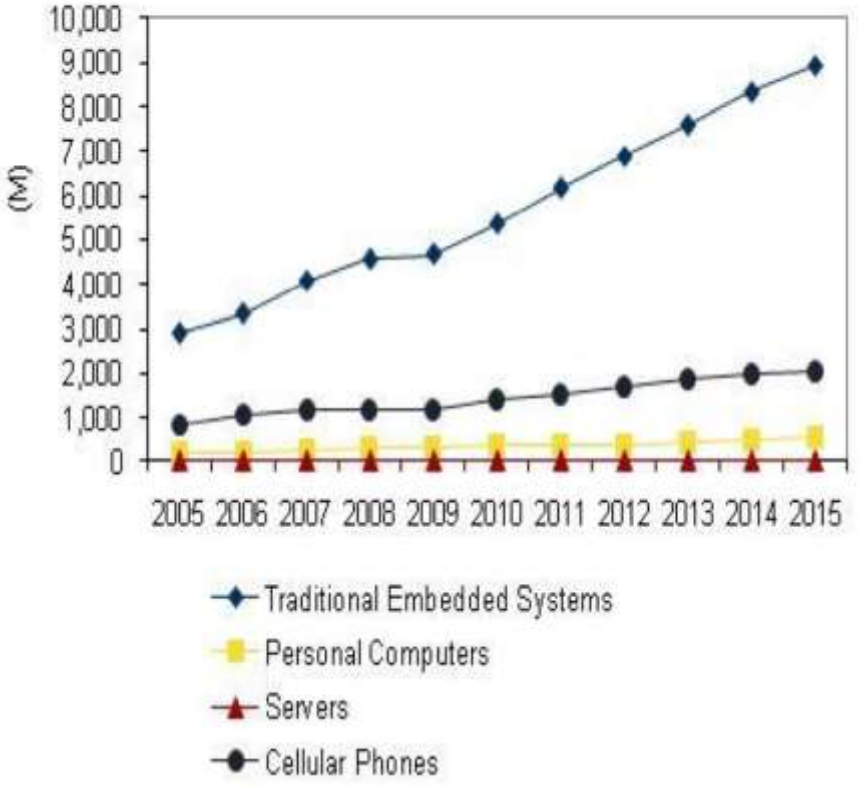
Worldwide Intelligent Systems Unit Shipments Comparison - Embedded Systems vs. Mainstream Systems 2011 Share and Growth



Notes:

Size of bubble equals 2011 share of system shipments. Growth of cell phone system shipments is driven by smartphones and multi core processor designs.

Worldwide Systems Unit Shipments - Traditional Embedded Systems vs. Mainstream Systems, 2005-2015 (Millions)



BERKELEY LAB

Ter@tec