



- Prepare Request
- Search Requests
- Generate Reports
- Approvals
- Help
- Wizard

- Search Requests
- [New Search](#)
- [Refine Search](#)
- [Search Results](#)
- [Clone Request](#)
- [Edit Request](#)
- [Cancel Request](#)

## Search Detail

### Submittal Details

#### Document Info

**Title :** Issues for the Future of Supercomputing: Impact of Moore's Law and Architecture on Application Performance

**Document Number :** 5246357

**SAND Number :** 2006-5989 P

**Review Type :** Electronic

**Status :** Approved

**Sandia Contact :** [DEBENEDICTIS,ERIK P.](#)

**Submittal Type :** Viewgraph/Presentation

**Requestor :** [DEBENEDICTIS,ERIK P.](#)

**Submit Date :** 09/26/2006

**Comments :** This is a tutorial at SC 06 performed in conjunction with Peter Kogge and David Keyes

**Peer Reviewed? :** N

#### Author(s)

**DEBENEDICTIS,ERIK P.**

#### Event (Conference/Journal/Book) Info

**Name :** Supercomputing 2006 Tutorial M06

**City :** Tampa

**State :** FL

**Country :** USA

**Start Date :** 11/13/2006

**End Date :** 11/13/2006

#### Partnership Info

**Partnership Involved :** No

**Partner Approval :**

**Agreement Number :**

#### Patent Info

**Scientific or Technical in Content :** Yes

**Technical Advance :** No

**TA Form Filed :** No

**SD Number :**

#### Classification and Sensitivity Info

**Title :** Unclassified-Unlimited

**Abstract :**

**Document :** Unclassified-Unlimited

**Additional Limited Release Info :** None.

**DUSA :** DIS-CS

### Routing Details

Role	Routed To	Approved By	Approval Date
Manager Approver	<a href="#">PUNDIT,NEIL D.</a>	<a href="#">PUNDIT,NEIL D.</a>	09/27/2006
Conditions:			
Administrator Approver	<a href="#">LUCERO,ARLENE M.</a>		

Created by WebCo Problems? Contact CCHD: **by email** or at 845-CCHD (2243).

For Review and Approval process questions please contact the **Application Process Owner**





---

# Tutorial M06

David E. Keyes



# Taking on the ITER Challenge, Scientists Look to Innovative Algorithms, Petascale Computers

By Michelle Sipics

The promise of fusion as a clean, self-sustaining and essentially limitless energy source has become a mantra for the age, held out by many scientists as a possible solution to the world's energy crisis and a way to reduce the amounts of greenhouse gases released into the atmosphere by more conventional sources of energy. If self-sustaining fusion reactions can be realized and maintained long enough to produce electricity, the technology could potentially revolutionize energy generation and use.

ITER, initially short for International Thermonuclear Experimental Reactor, is now the official, non-acronymic name (meaning “the way” in Latin) of what is undoubtedly the largest undertaking of its kind. Started as a collaboration between four major parties in 1985, ITER has evolved into a seven-party project that finally found a physical home last year, when it was announced that the ITER fusion reactor would be built in Cadarache, in southern France. (The participants are the European Union, Russia, Japan, China, India, South Korea, and the United States.) In May, the seven initialed an agreement documenting the negotiated terms for the construction, operation, and decommissioning of the ITER tokamak, signifying another milestone for both the project itself and its eventual goal of using fusion to facilitate large-scale energy generation for the world.

Problems remain, however—notably the years, and perhaps decades, of progress needed to attain such a goal. In fact, even *simulating* the proposed ITER tokamak is currently out of reach. But according to David Keyes, a computational mathematician at Columbia University and acting director of the Institute for Scientific Computing Research (ISCR) at Lawrence Livermore National Laboratory, the ability to perform such simulations may be drawing closer.

## Hardware 3, Software 9

“Fusion scientists have been making useful characterizations about plasma fusion devices, physics, operating regimes and the like for over 50 years,” Keyes says. “However, to simulate the dynamics of ITER for a typical experimental ‘shot’ over scales of interest with today’s most commonly used algorithmic technologies would require approximately  $10^{24}$  floating-point operations.” That sounds bleak, given the 280.6 Tflop/s ( $10^{12}$  flops/s) benchmark performance of the IBM BlueGene/L at Lawrence Livermore National Laboratory—as of June the fastest supercomputer in the world. But Keyes is optimistic: “We expect that with proper algorithmic ingenuity, we can reduce this to  $10^{15}$  flops.”

Optimizing the algorithms used, in other words, could lower the computing power required for some ITER simulations by an astounding nine orders of magnitude. Even more exciting, those newly feasible simulations would be at the petascale—ready to run on the petaflop/s supercomputers widely expected within a few years.

The ingenuity envisioned by Keyes even has a roadmap. Together with Stephen Jardin of the Princeton Plasma Physics Laboratory, Keyes developed a breakdown that explains where as many as 12 orders of magnitude of speedup will come from over the next decade: 1.5 from increased parallelism, 1.5 from greater processor speed and efficiency, four from adaptive gridding, one from higher-order elements, one from field-line following coordinates, and three from implicit algorithms.





# Presentation Features

---

- Briefly reflect on recent progress in high-end scientific computing, as illustrated on Bell Prize-winning applications – why?
  - Bell has attracted high-end attention thru two decades of architectures
  - Winners document performance issues beyond details found in other computational science papers, which instead emphasize science
  - PDE-based simulations are the dominant type of Bell submission
  - Performance-orientation exposes an interesting fallacy for our discussion 😊
- Look generically at PDE-based simulation and the basis of continued optimism for its growth – capability-wise, looking at real applications
- Look at some specific hurdles to PDE-based simulation posed by high-end architecture
- Study in detail an unstructured Bell Prize entry to note architectural stresses





# Technical aspects of presentation

---

- **Introduce a parameterized highly tunable class of algorithms for parallel implicit solution of PDEs**
  - **understand the source of their “weak scalability”**
  - **understand their lack of “strong scalability”**
  - **understand why explicit algorithms generally do not scale, even weakly, in the high spatial resolution limit**
- **Note some algorithmic “adaptations” to architectural stresses**





# Gordon Bell Prize “peak performance”

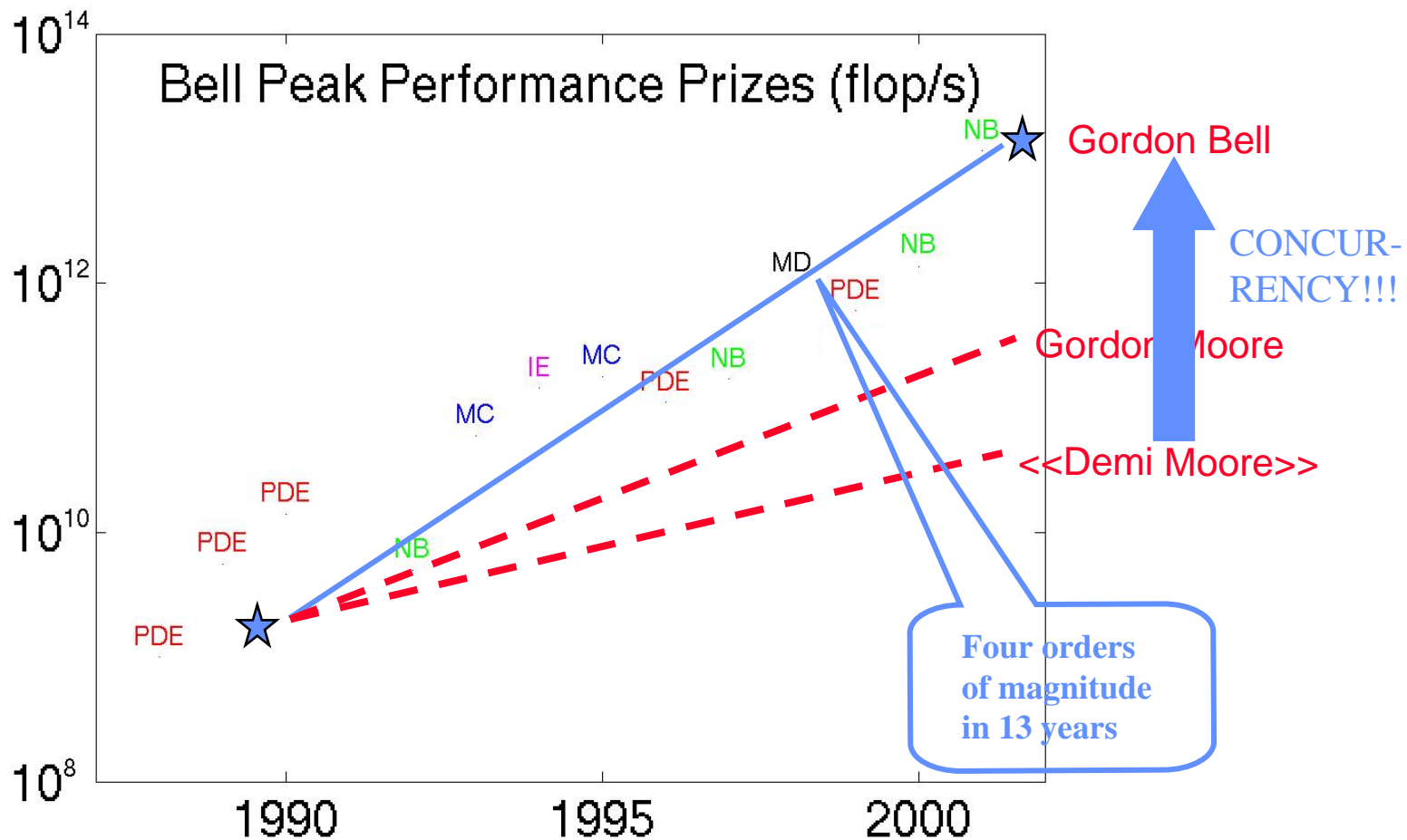
<i>Year</i>	<i>Type</i>	<i>Application</i>	<i>No. Procs</i>	<i>System</i>	<i>Gflop/s</i>
1988	PDE	Structures	8	Cray Y-MP	1.0
1989	PDE	Seismic	2,048	CM-2	5.6
1990	PDE	Seismic	2,048	CM-2	14
1992	NB	Gravitation	512	Delta	5.4
1993	MC	Boltzmann	1,024	CM-5	60
1994	IE	Structures	1,904	Paragon	143
1995	MC	QCD	128	NWT	179
1996	PDE	CFD	160	NWT	111
1997	NB	Gravitation	4,096	ASCI Red	170
1998	MD	Magnetism	1,536	T3E-1200	1,020
1999	PDE	CFD	5,832	ASCI BluePac	627
2000	NB	Gravitation	96	GRAPE-6	1,349
2001	NB	Gravitation	1,024	GRAPE-6	11,550
2002	PDE	Climate	5,120	Earth Sim	26,500
2003	PDE	Seismic	1,944	Earth Sim	5,000
2004	PDE	CFD	4,096	Earth Sim	15,200
2005	MD	Solidification	131,072	BG/L	101,700

**Five orders  
of magnitude  
in 17 years**





# Gordon Bell Prize outpaces Moore's Law







# IBM's BlueGene/L: 65536 dual procs, 360 Tflop/s

System  
(64 cabinets, 64x32x32)

Cabinet  
(32 Node boards, 8x8x16)

Node Board  
(32 chips, 4x4x2)  
16 Compute Cards

Compute Card  
(2 chips, 2x1x1)

Chip  
(2 processors)

2.8/5.6 GF/s  
4 MB

5.6/11.2 GF/s  
0.5 GB DDR

90/180 GF/s  
8 GB DDR

2.9/5.7 TF/s  
256 GB DDR

Present offer from IBM

Single cabinet  
5.7 TFlop/s peak  
\$2M in acad. consortium







# **Tally of Peak Prize formulations and apps**

---

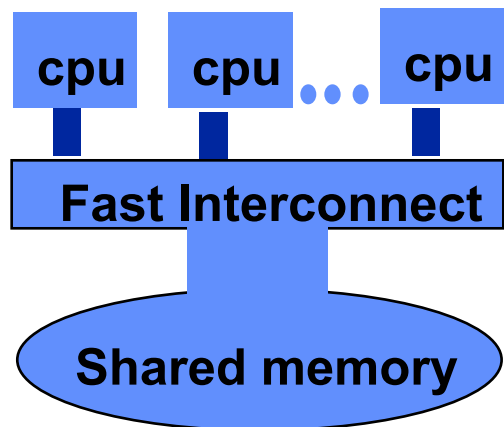
- **8 × Partial differential equations**
  - Climate, fluids, seismology, structures
- **4 × N-body dynamics**
  - Gravitation
- **3 × Molecular dynamics**
  - Electronic structure, magnetism, solidification
- **2 × Monte Carlo methods**
  - Boltzmann, Quantum Chromodynamics
- **1 × Integral equations**
  - Structures





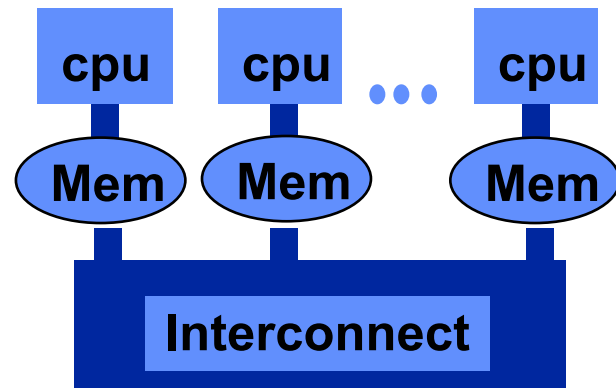
# Tally of Gordon Bell Peak Prize hardware

## Symmetric Multi-Processor (SMP)



- two to hundreds of processors
- **shared memory**
- **global addressing**
- **4 prizes, last in 1993**

## Massively Parallel Processor (MPP)



- thousands to hundreds of thousands of processors
- **distributed memory**
- **local addressing**
- **13 prizes, including last 12**





# The “other” Bell prizes

---

- “Peak” is only one of several types of Gordon Bell Prizes that have been awarded over the years
  - The only one awarded *each* time there have been Bell Prizes
- “Price-performance” has been recognized 12 times, but not since 2001, when it stagnated at about 25 cents per delivered Mflop/s
  - A *few* of these have been for implementations of PDEs
- “Special” was first awarded in 1999 and has sometimes inspired multiple awards per year
  - *Most* of these have gone to implementation of PDEs
- “Compiler-derived parallelism” has been awarded three times, most recently in 2002 for HPF
  - Two of these have gone to implementations of PDEs
- “Speedup” (strong, that is) was explicitly recognized once, in 1992
  - For an implementation of a PDE





# Gordon Bell Prize: “price performance”

<i>Year</i>	<i>Application</i>	<i>System</i>	<i>\$ per Mflops</i>
1989	Reservoir modeling	CM-2	2,500
1990	Electronic structure	IPSC	1,250
1992	Polymer dynamics	cluster	1,000
1993	Image analysis	custom	154
1994	Quant molecular dyn	cluster	333
1995	Comp fluid dynamics	cluster	278
1996	Electronic structure	SGI	159
1997	Gravitation	cluster	56
1998	Quant chromodynamics	custom	12.5
1999	Gravitation	custom	6.9
2000	Comp fluid dynamics	cluster	1.9
2001	Structural analysis	cluster	0.24

**Four orders  
of magnitude  
in 12 years**





# Whimsical remarks on Bell, 1988-2005

---

- If similar improvements in **speed** ( $10^5$ ) had been realized in the airline industry, a 3-hour flight would require one-tenth of a second today
- If similar reductions in **cost** ( $10^4$ ) had been realized in higher education, tuition room and board would cost about \$2 per year
- If similar improvements in **storage** ( $10^4$ ) had been realized in the publishing industry, our office bookcases could hold the book portion of the collection of the Library of Congress (~18M volumes)





## How to use a jar of peanut butter with a rapidly dropping price?

---

- In 2006, at \$3.19: make sandwiches
- By 2009, at \$0.80: make recipe substitutions
- By 2012, at \$0.20: use as feedstock for biopolymers, plastics, etc.
- By 2115, at \$0.05: heat homes
- By 2118, at \$0.012: pave roads ☺

The cost of computing has been on a curve like this for two decades. Can we count on another decade? If so, like everyone else, scientists & engineers should plan increasing uses for it. If *not* ...





# Performance vs. time-to-solution

---

- **Gordon Bell peak prizes cannot, by definition, go to thread-nonuniform, flop-bare simulations**
- **Prizes tend to concentrate in regular, Cartesian index space, flop-rich computations**
- **There is a conflict between what the peak prize measures and**
  - **what is good for the computational science community, in terms of getting its work done**
  - **what is good for the computational mathematics community, in terms of identifying interesting problems**
- **The “special” prize attempts to remedy this shortcoming of the traditional prize, and is often the most interesting category**





# Gordon Bell Prize: “special”

---

<i>Year</i>	<i>Application</i>	<i>Discretization</i>	<i>System</i>
1999	Aerodynamics	unstructured	Intel ASCI Red
1999	Stellar physics	spectral	Intel ASCI Red
2000	Reactive flow	Cartesian AMR	Intel ASCI Red
2001	Relativistic fields	structured	cluster
2002	Structural dynamics	unstructured	IBM ASCI White
2002	DNS	structured	Earth Simulator
2002	Biomolecular dynamics	—	
2003	Seismic inversion	unstructured	HP LeMieux
2004	Bone mechanics	unstructured	IBM ASCI White

*8 of 9 “special” awards have gone to PDE simulations*

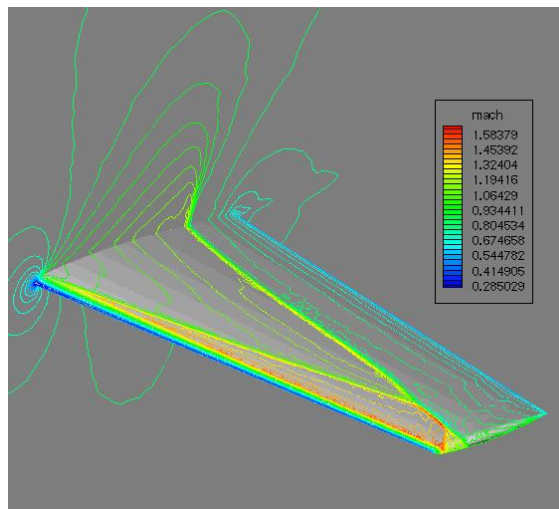
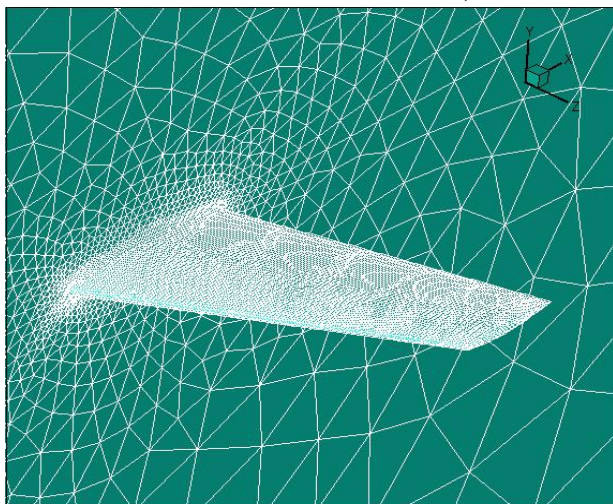




# 1999 Gordon Bell “special” prize

- 1999 Bell Prize in “special category” went to implicit, unstructured grid aerodynamics problems
  - 0.23 Tflop/s sustained on 3 thousand processors of Intel’s ASCI Red
  - 11 million degrees of freedom
  - incompressible and compressible Euler flow
  - employed in NASA analysis/design missions

Transonic “Lambda” Shock, Mach contours on surfaces

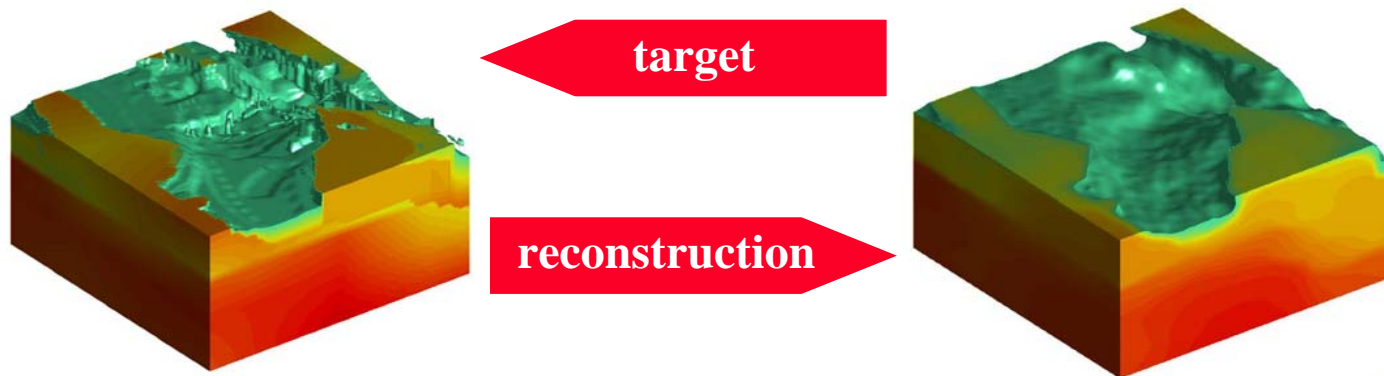






# 2003 Gordon Bell “special” prize

- 2003 Bell Prize in “special category” went to unstructured grid geological parameter estimation problem
  - 1 Tflop/s sustained on 2 thousand processors of HP’s “Lemieux
  - each explicit forward PDE solve: 17 million degrees of freedom
  - seismic inverse problem: 70 billion degrees of freedom
  - employed in NSF seismic research at CMU

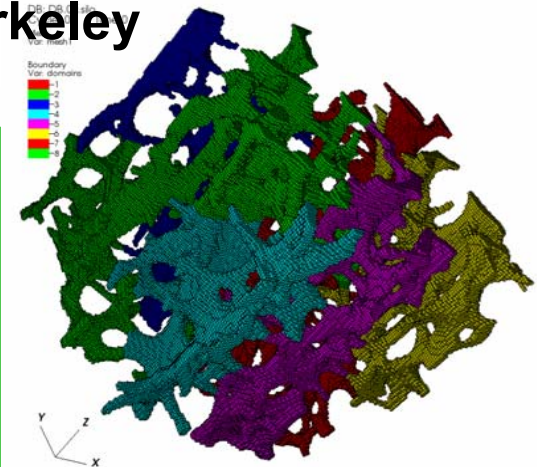
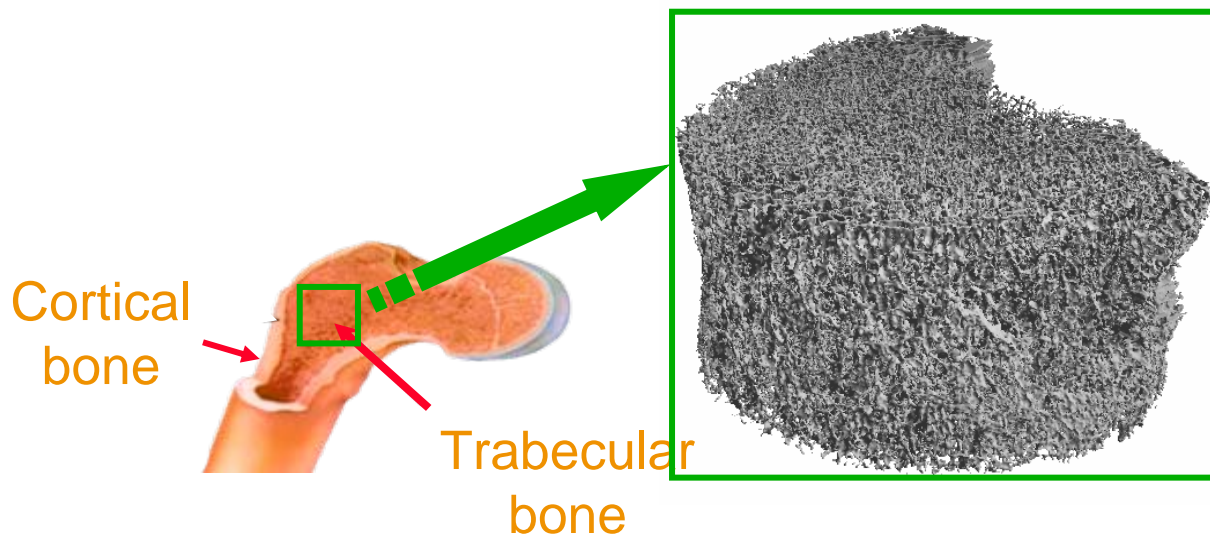






# 2004 Gordon Bell “special” prize

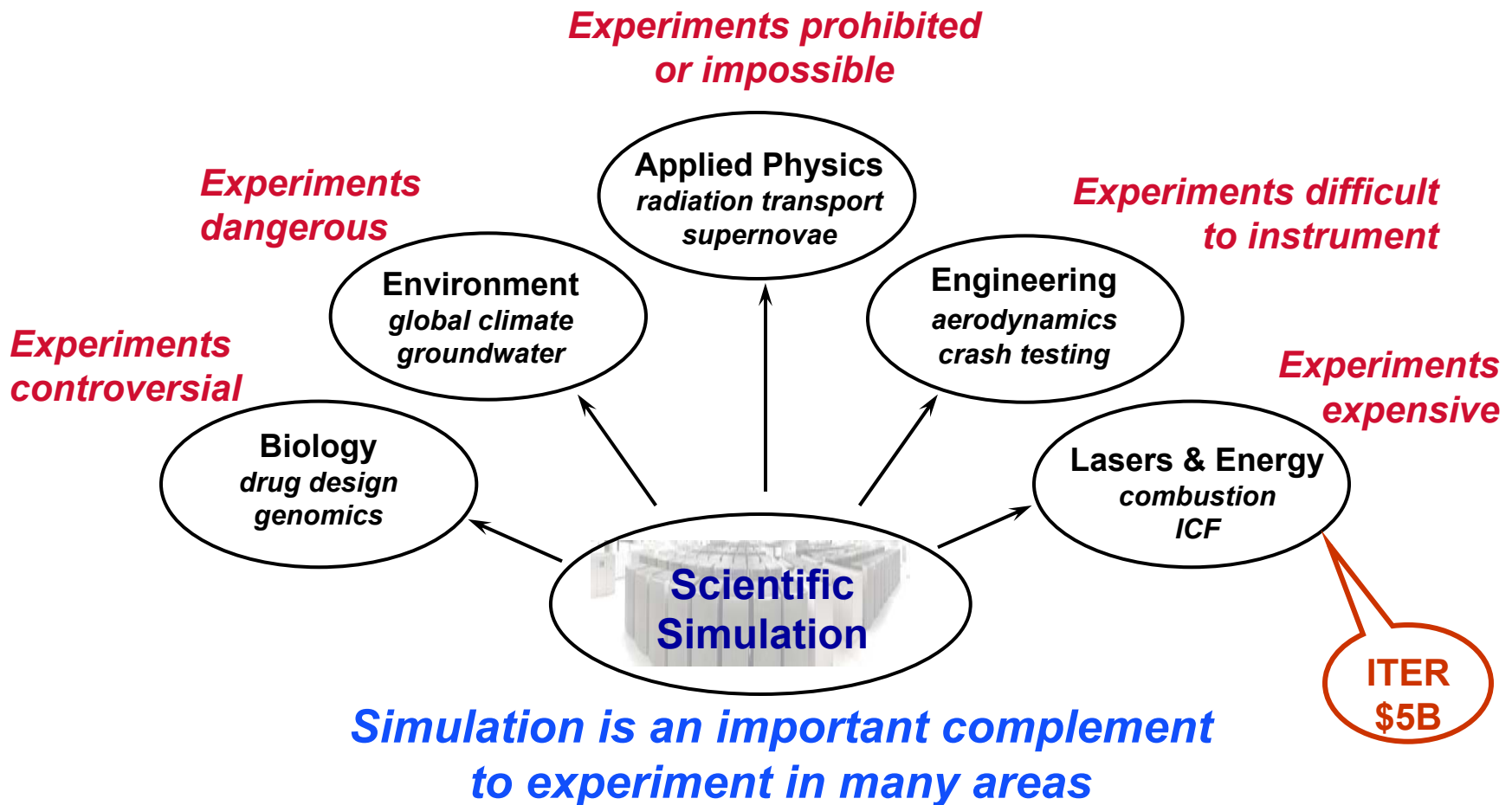
- 2004 Bell Prize in “special category” went to an implicit, unstructured grid bone mechanics simulation
  - 0.5 Tflop/s sustained on 4 thousand procs of IBM’s ASCI White
  - 0.5 billion degrees of freedom
  - large-deformation analysis
  - employed in NIH bone research at Berkeley







# Terascale simulation is pitched as an alternative to (some) experimentation







# Context: many recent reports promote high-end simulation

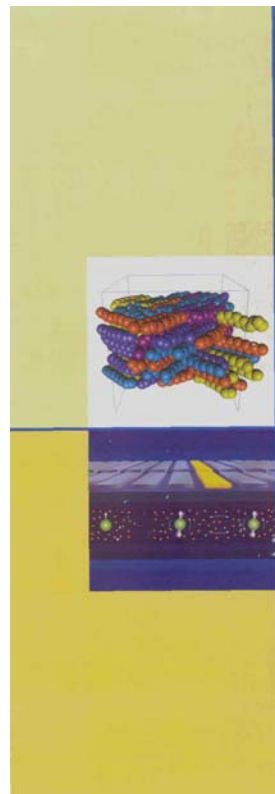
---

- **Cyberinfrastructure (NSF, 2003)**
  - new research environments through cyberinfrastructure
- **Facilities for the Future of Science (DOE, 2003)**
  - “ultrascale simulation facility” ranked #2 behind ITER only
- **High End Computing Revitalization Task Force (Interagency, 2004)**
  - strategic planning on platforms
- **Future of Supercomputing (NAS, 2005)**
  - broad discussion of the future of supercomputing
- **PITAC (Interagency, 2005)**
  - challenges in software and in interdisciplinary training
- **Simulation-based Engineering Science (NSF, 2006)**
  - opportunities in dynamic, data-driven simulation and engineering design
- **Advanced Nuclear Energy Simulations (DOE, 2006)**
- **SCaLeS report, Vol 1 (DOE, 2003) & Vol 2 (DOE, 2004)**
  - implications of large-scale simulation for basic scientific research
- **Capability Computing Needs (DOE, 2004)**
  - profiles of leading edge DOE codes in 11 application domains

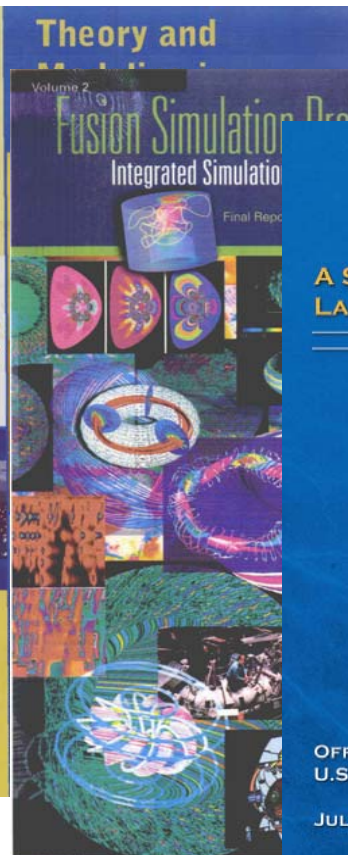




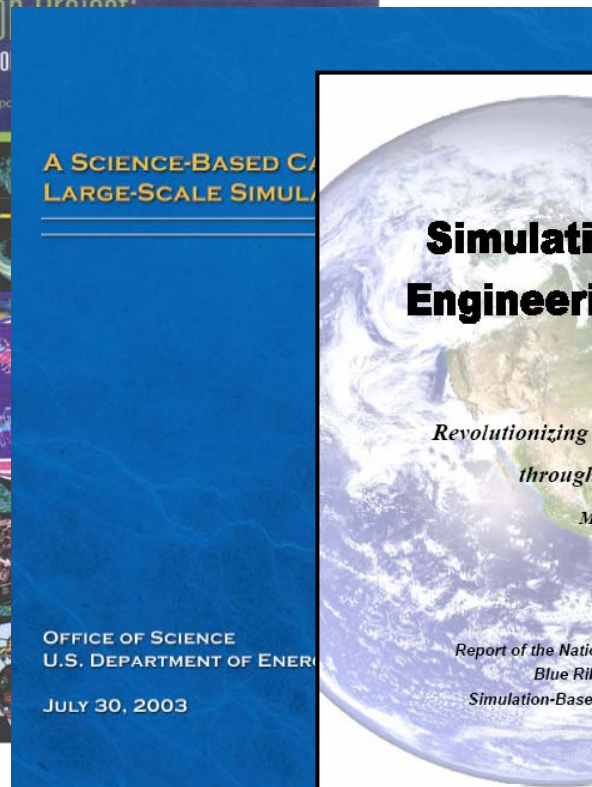
# Diverse applications, common algorithmic and architectural infrastructure



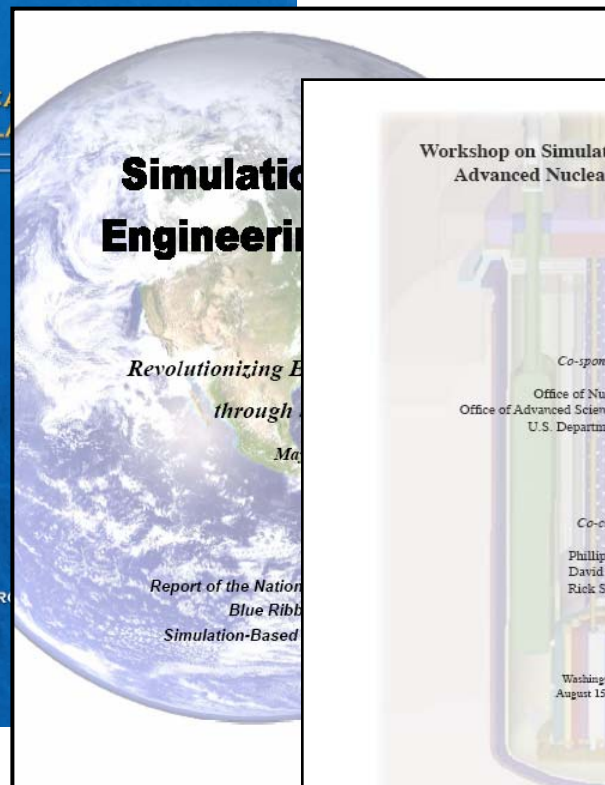
2002



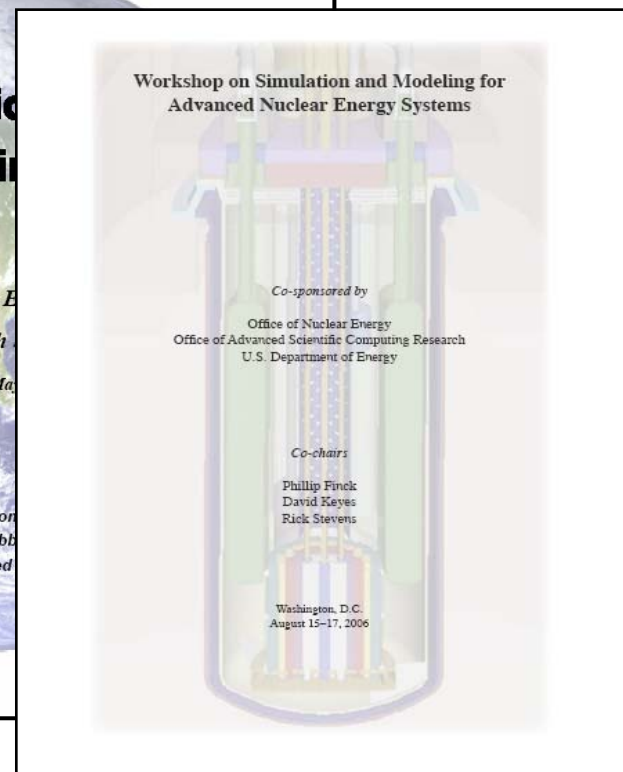
2003



2003-2004 (vol 2)



2006

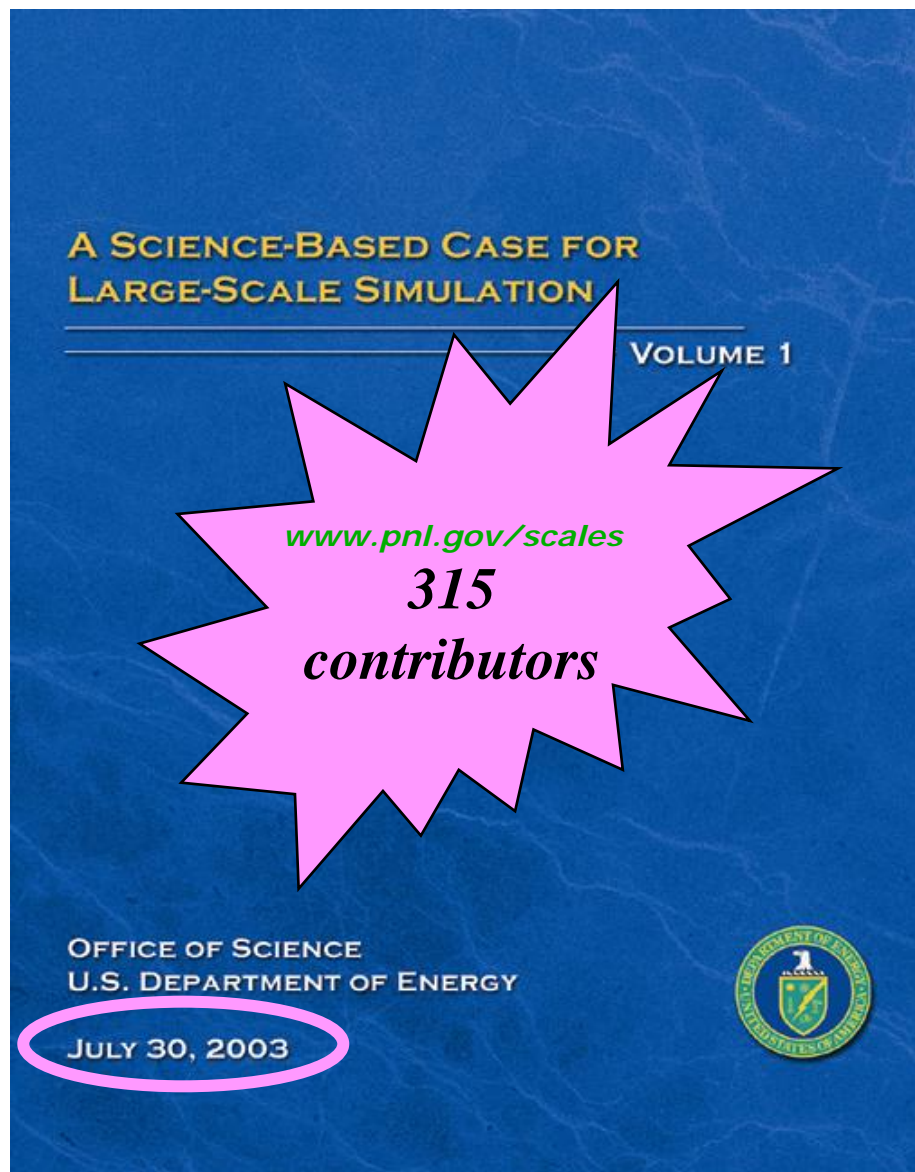


2006





# A primary source: SCaLeS



- Chapter 1. *Introduction*
  - Chapter 2. *Scientific Discovery through Advanced Computing: a Successful Pilot Program*
  - Chapter 3. *Anatomy of a Large-scale Simulation*
  - Chapter 4. *Opportunities at the Scientific Horizon*
  - Chapter 5. *Enabling Mathematics and Computer Science Tools*
  - Chapter 6. *Recommendations and Discussion*
- Volume 2 (2004):**
- 11 chapters on applications
  - 8 chapters on mathematical methods
  - 8 chapters on computer science and infrastructure





# **“What would scientists do with 100-1000x?” (SCaLeS)**

---



- **Predict future climates**
- **Probe structure of particles**
- **Design accelerators**
- **Design and control tokamaks**
- **Control combustion**
- **Probe supernovae**





# What would scientists do with 100-1000x?

## Example: predict future climates

---



- **Resolution**
  - refine horizontal in atmosphere from 160 to 40 km
  - refine horizontal in ocean from 105 to 15km
- **New “physics”**
  - atmospheric chemistry
  - carbon cycle
  - dynamic terrestrial vegetation (nitrogen and sulfur cycles and land-use and land-cover changes)
- **Improved representation of subgrid processes**
  - clouds
  - atmospheric radiative transfer

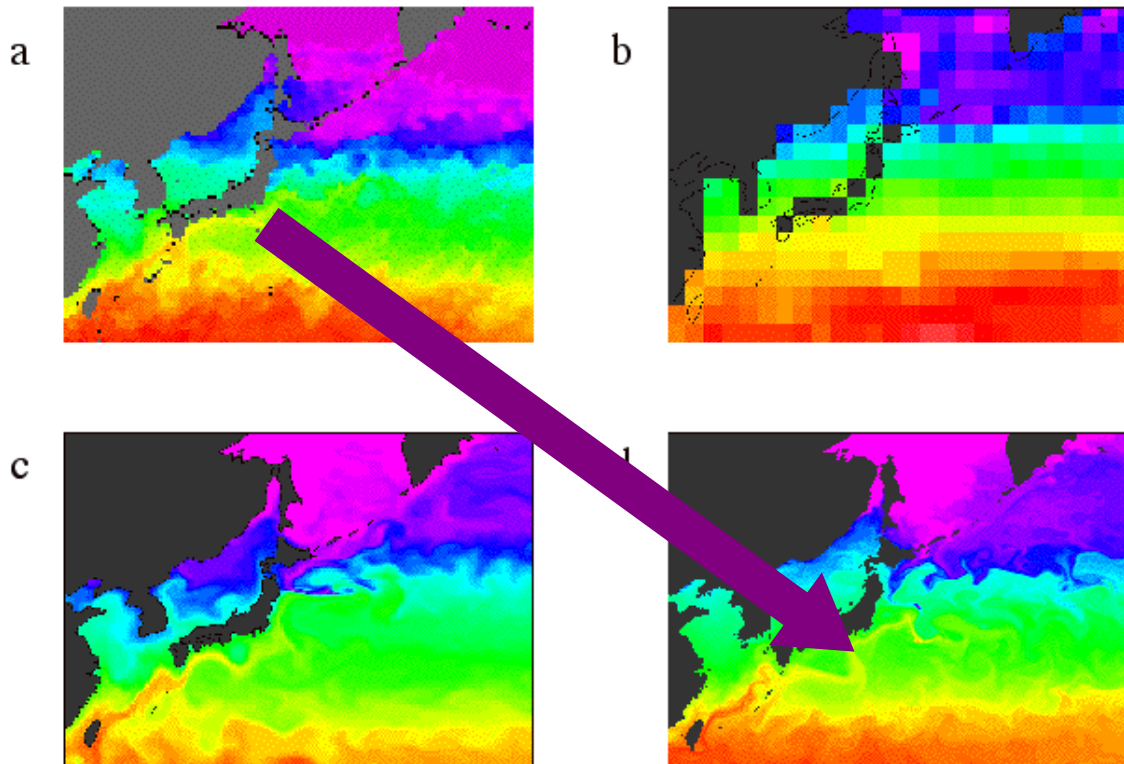




# What would scientists do with 100-1000x?

## Example: predict future climates

**Resolution of Kuroshio Current:** Simulations at various resolutions have demonstrated that, because equatorial meso-scale eddies have diameters  $\sim 10$ -200 km, the grid spacing must be  $< 10$  km to adequately resolve the eddy spectrum. This is illustrated in four images of the sea-surface temperature. Figure (a) shows a snapshot from satellite observations, while the three other figures are snapshots from simulations at resolutions of (b)  $2^\circ$ , (c)  $0.28^\circ$ , and (d)  $0.1^\circ$ .







What would scientists do with 100-1000x?



## Example: probe structure of particles

---

- **Resolution**

- take current 4D models from  $32 \times 32 \times 32 \times 16$  to  $128 \times 128 \times 128 \times 64$

- **New physics**

- “unquench” the lattice approximation: enable study of the gluon structure of the nucleon, in addition to its quark structure
  - obtain chiral symmetry by solving on a 5D lattice in the domain wall Fermion formulation
  - allow precision calculation of the spectroscopy of strongly interacting particles with unconventional quantum numbers, guiding experimental searches for states with novel quark and gluon structure

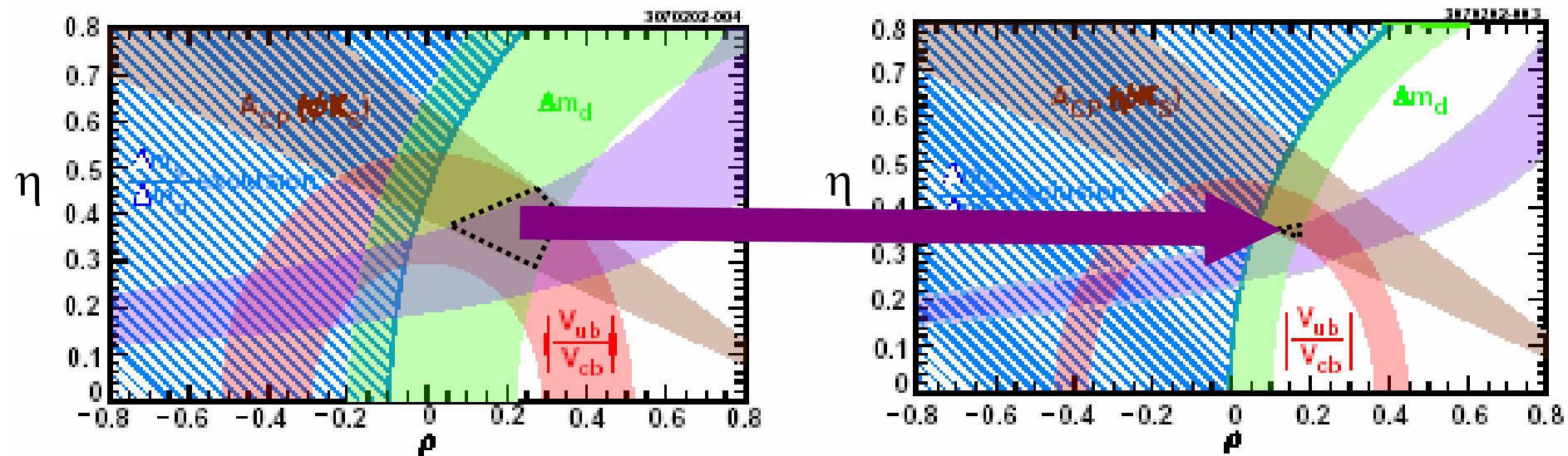




What would scientists do with 100-1000x?

## Example: probe structure of particles

*Constraints on the Standard Model parameters  $\rho$  and  $\eta$ .* For the Standard Model to be correct, these parameters from the Cabibbo-Kobayashi-Maskawa (CKM) matrix must be restricted to the region of overlap of the solidly colored bands. The figure on the left shows the constraints as they exist today. The figure on the right shows the constraints as they would exist with no improvement in the experimental errors, but with lattice gauge theory uncertainties reduced to 3%.







# What would scientists do with 100-1000x?

## Example: design accelerators

---



- **Resolution**

- complex geometry (long assemblies of damped detuned structure (DDS) cells, each one slightly different than its axial neighbor) requires unstructured meshes with hundreds of millions of degrees of freedom
- Maxwell eigensystems for interior elements of the spectrum must be solved in the complex cavity formed by the union of the DDS cells

- **Novel capability**

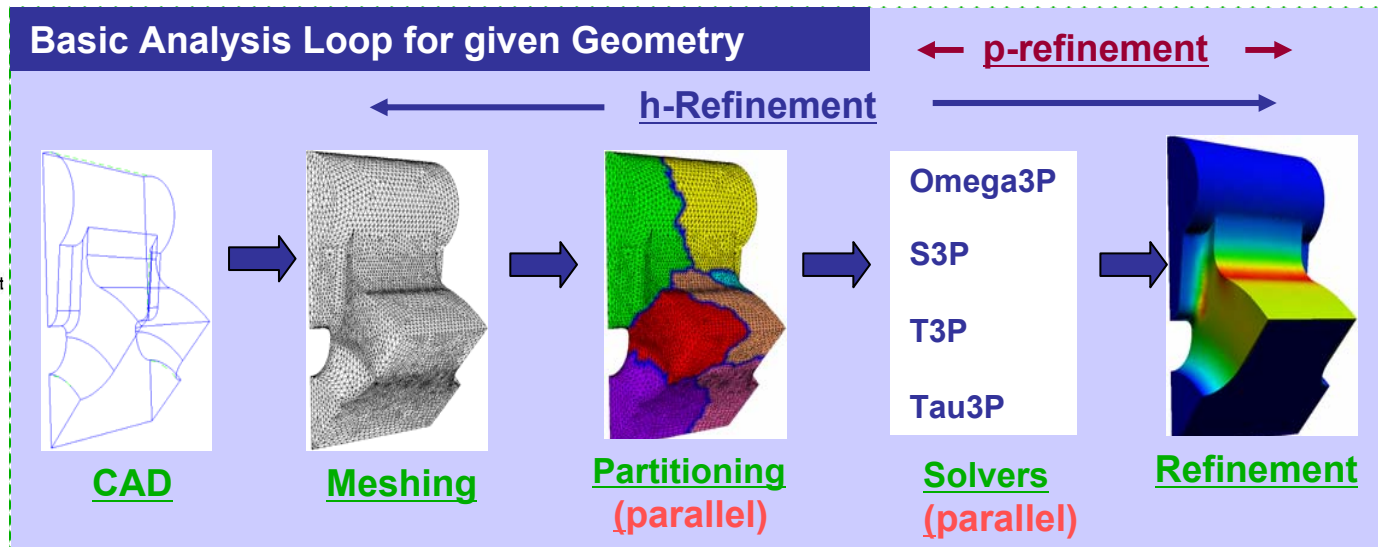
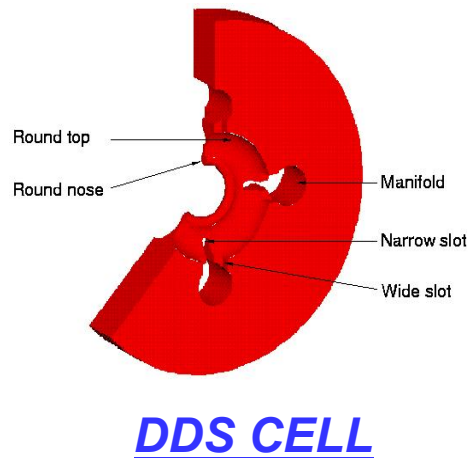
- PDE-based mathematical optimization will replace expensive and slow trial and error prototyping approach
- each inner loop of optimization requires numerous eigensystem analyses





# What would scientists do with 100-1000x?

## Example: design accelerators



*Next generation accelerators have complex cavities.* Shape optimization is required to improve performance and reduce operating cost.



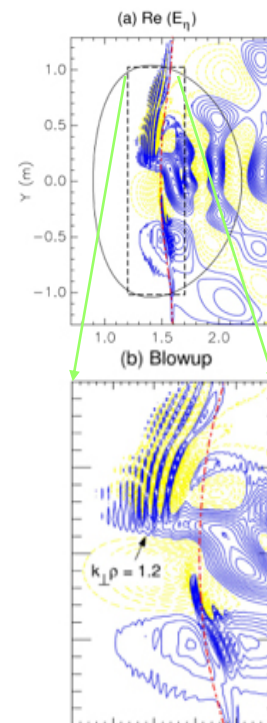
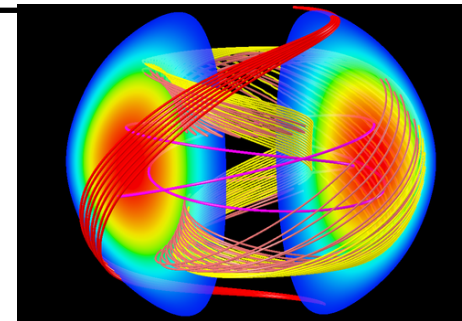




What would scientists do with 100-1000x?

## Example: design and control tokamaks

- Resolution
  - refine meshes and approach physical Lundquist numbers
- Multiphysics
  - combine MHD, PIC, and RF codes in a single, consistent simulation
  - resolve plasma edge
- Design and control
  - optimize performance of experimental reactor ITER and follow-on production devices
  - detect onset of instabilities and modify before catastrophic energy releases from the magnetic field

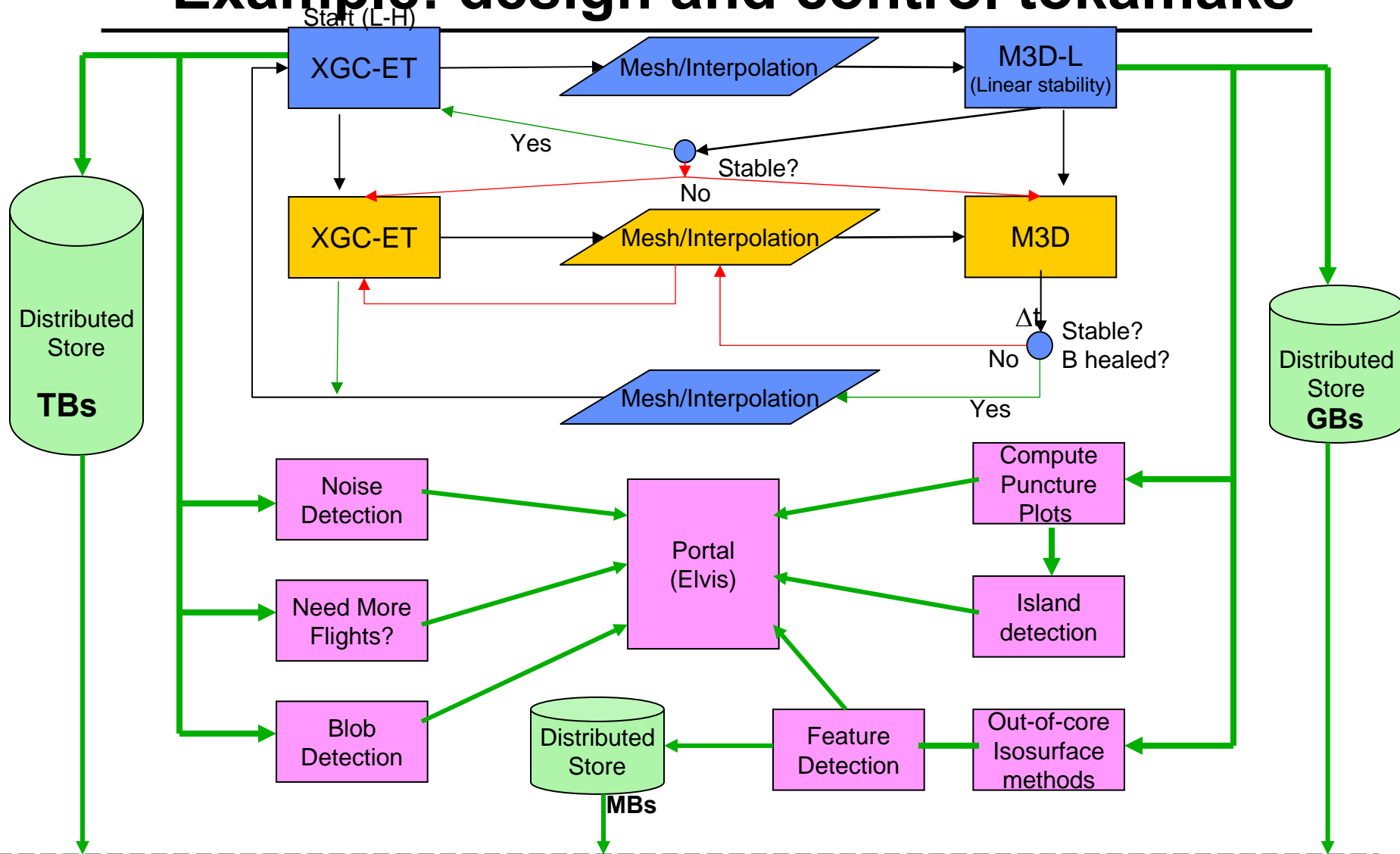






What would scientists do with 100-1000x?

# Example: design and control tokamaks







# What would scientists do with 100-1000x?

## Example: control combustion

---

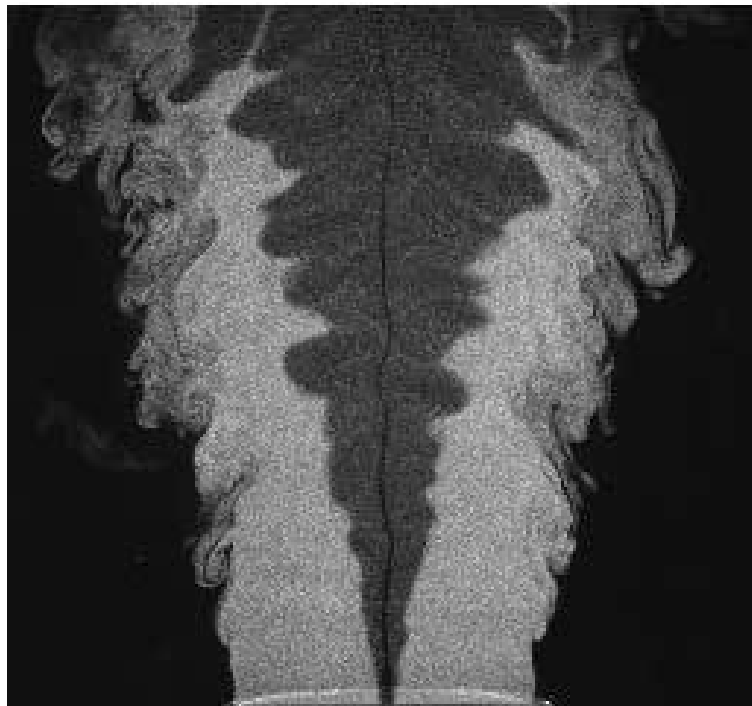
- **Resolution**
  - evolve 3D time-dependent large-eddy simulation (LES) codes to direct Navier-Stokes (DNS)
  - multi-billions of mesh zones required
- **New “physics”**
  - explore coupling between chemistry and acoustics (currently filtered out)
  - explore sooting mechanisms to capture radiation effects
  - capture autoignition with realistic fuels
- **Integrate with experiments**
  - pioneer simulation-controlled experiments to look for predicted effects in the laboratory





# What would scientists do with 100-1000x?

## Example: control combustion



Experimental PIV measurement

Instantaneous flame front imaged by density of inert marker



Simulation

Instantaneous flame front imaged by fuel concentration

Images c/o R. Cheng (left), J. Bell (right), LBNL, and NERSC  
**2003 SIAM/ACM Prize in CS&E (J. Bell & P. Colella)**





What would scientists do with 100-1000x?



## Example: probe supernovae

---

- **Resolution**

- current Boltzmann neutrino transport models are vastly under-resolved
- need at least  $512^3$  spatially, at least 8 polar and 8 azimuthal, and at least 24 energy groups per each of six neutrino types
- to discriminate between competing mechanisms, must conserve energy to within 0.1% over millions of time steps

- **Full dimensionality**

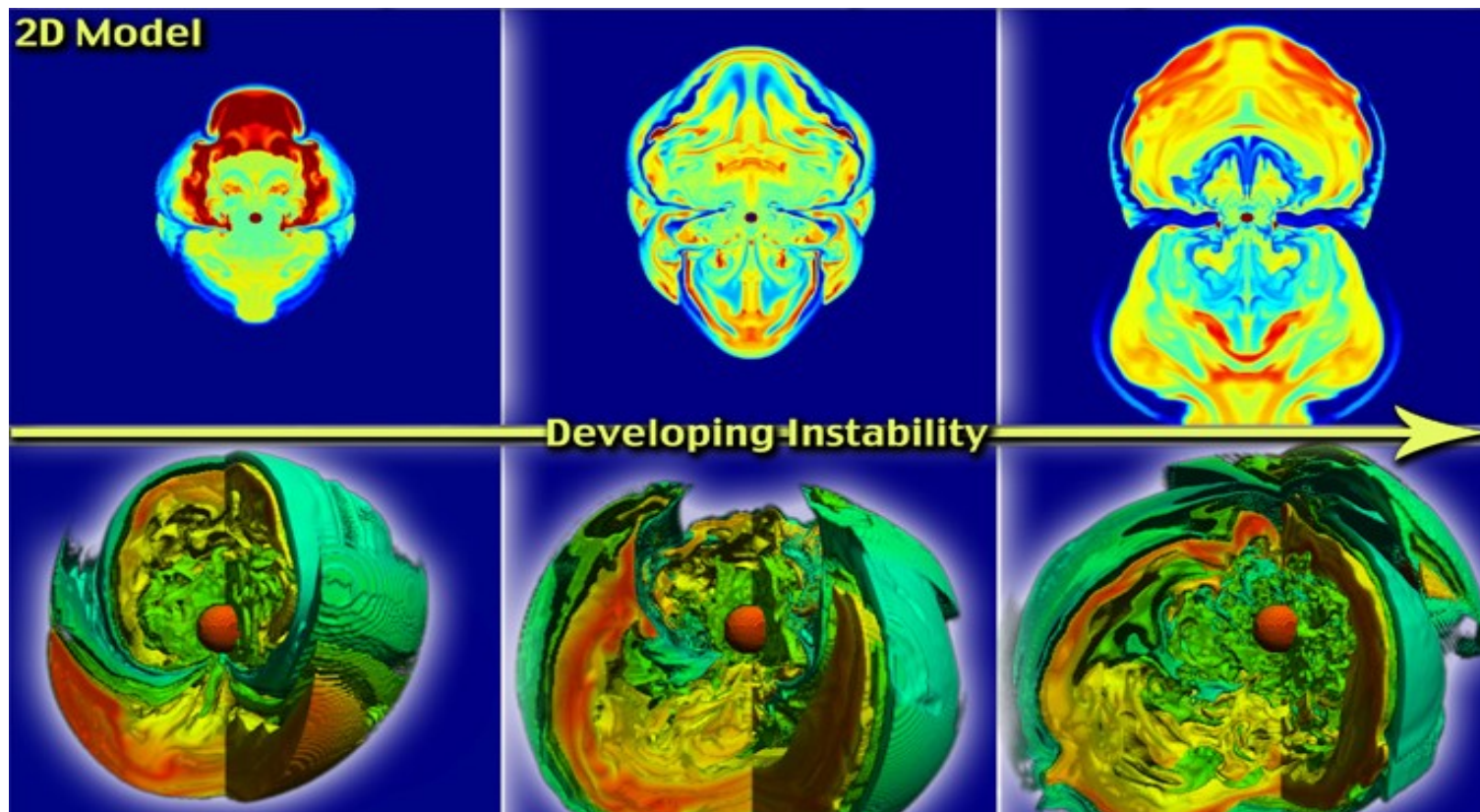
- current models capable of multigroup neutrino radiation are lower-dimensional; full 3D models are required





# What would scientists do with 100-1000x?

## Example: probe supernovae



*Stationary accretion shock instability defines shape of supernovae and direction of emitted radiation.* Lower dimensional models produce insight; full dimensional models are ultimately capable of providing radiation signatures that can be compared with observations.





# Progress in scaling PDE applications

---

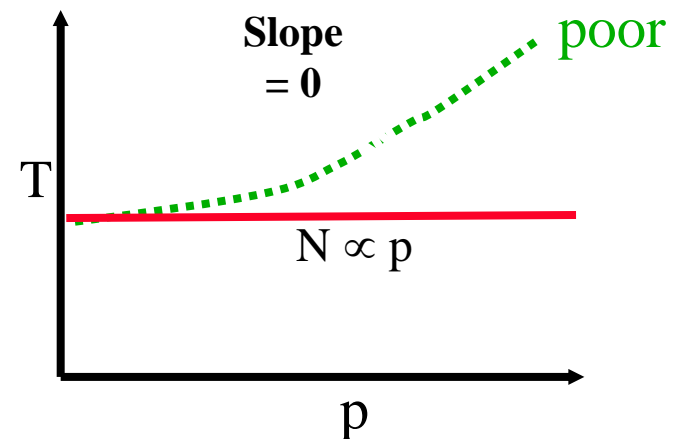
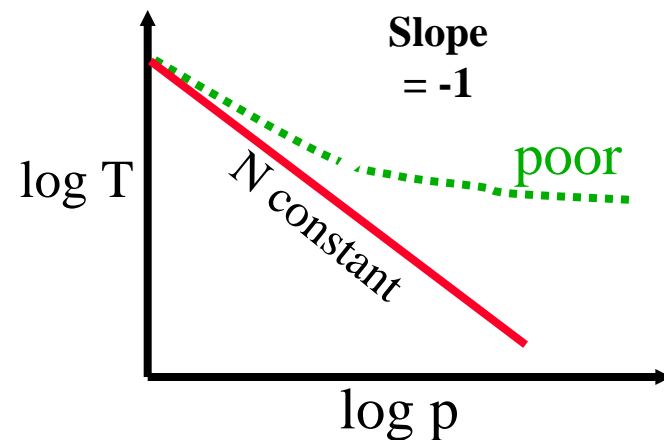
- Both structured and unstructured grids
- Both explicit and implicit methods
- Multiple decades of spatial “resolution”
- Many-thousand-fold concurrency
- Strong scaling within modest ranges
- Weak scaling (also called “scaled speedup”) without obvious limits





# Review: two definitions of scalability

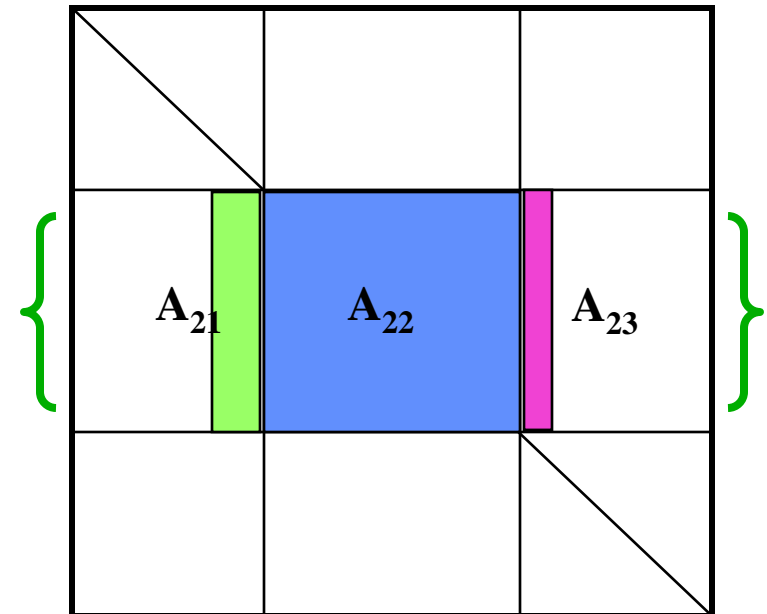
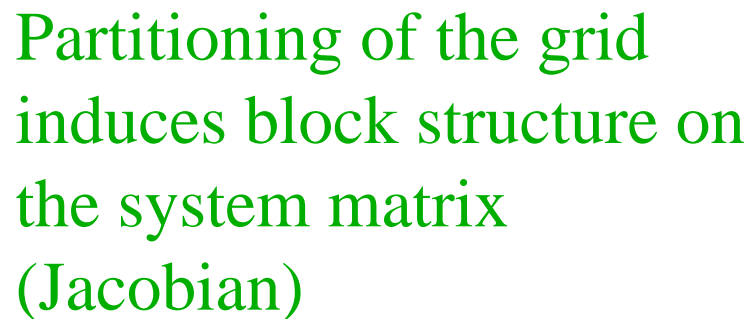
- “Strong scaling”
  - execution time decreases in inverse proportion to the number of processors
  - *fixed size problem overall*
  - often instead graphed as reciprocal, “speedup”
- “Weak scaling”
  - execution time remains constant, as problem size and processor number are increased in proportion
  - *fixed size problem per processor*
  - also known as “Gustafson scaling”







(volume) work to (surface)  
communication is preserved  
under weak scaling





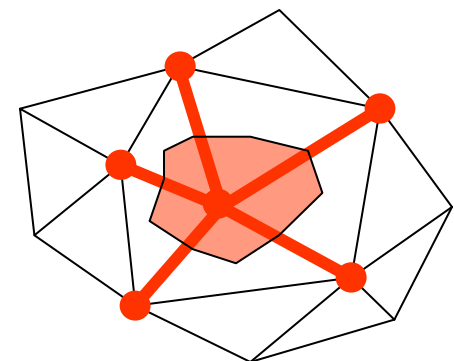
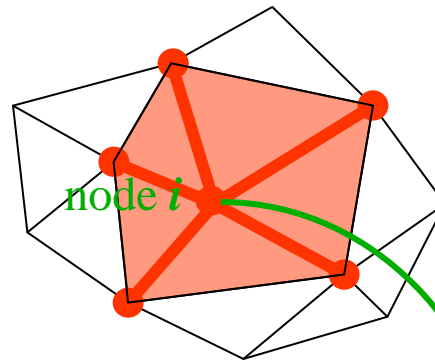
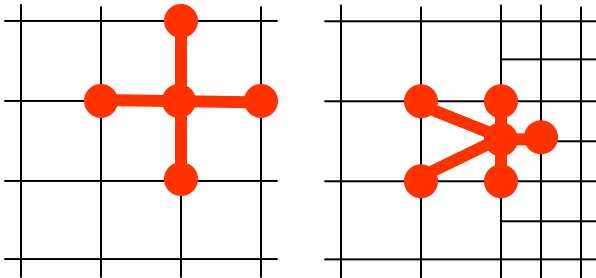


# DD relevant to any local stencil formulation

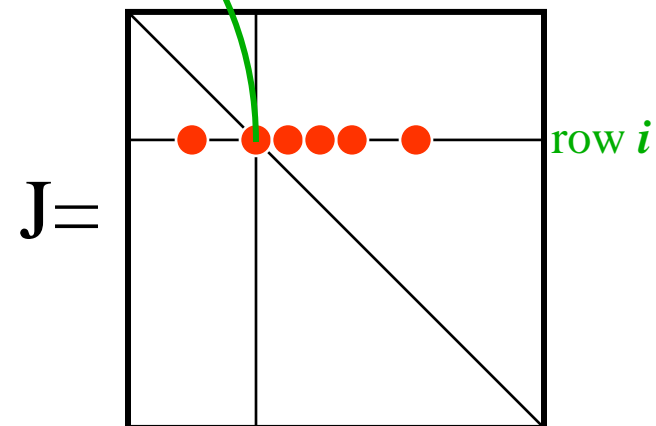
finite differences

finite elements

finite volumes



- All lead to sparse Jacobian matrices
- However, the inverses are generally dense; even the factors suffer unacceptable fill-in in 3D
- Want to solve in subdomains only, and use to precondition full sparse problem







# An algorithm for PDE simulation: Newton-Krylov-Schwarz

---

nonlinear residual  
evaluations, inner  
products, DAXPYs

sparse MATVECs,  
inner products,  
DAXPYs

local solves,  
small global  
solves



Newton

nonlinear solver

*asymptotically quadratic*



Krylov

accelerator

*spectrally adaptive*



Schwarz

preconditioner

*parallelizable*





# Krylov-Schwarz parallelization is simple!

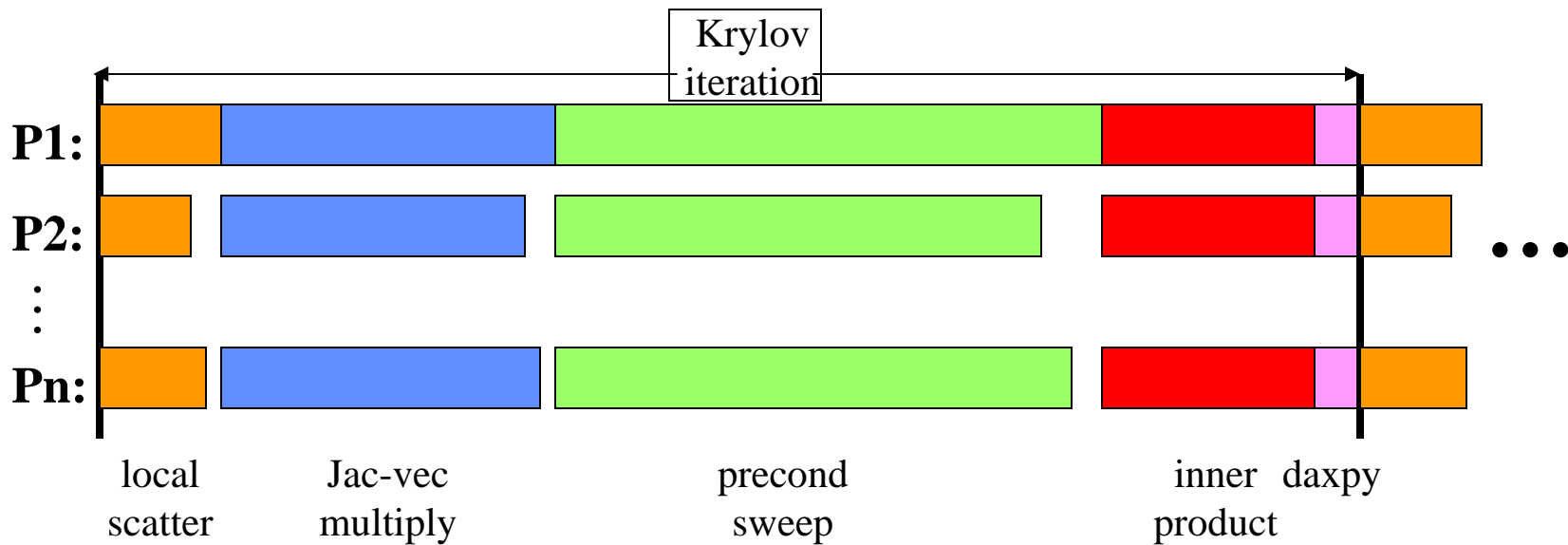
---

- **Decomposition into concurrent tasks**
  - by domain
- **Assignment of tasks to processes**
  - typically one subdomain per process
- **Orchestration of communication between processes**
  - to perform sparse matvec – near neighbor communication
  - to perform subdomain solve – nothing
  - to build Krylov basis – global inner products
  - to construct best fit solution – global sparse solve (redundantly)
- **Mapping of processes to processors**
  - typically one process per processor

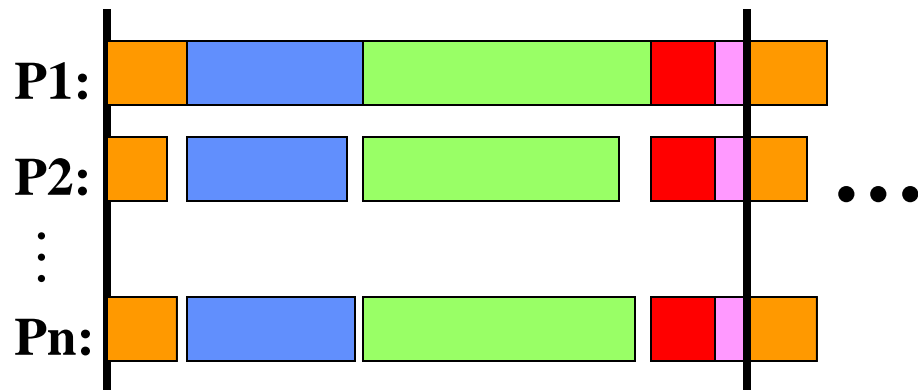




# Inner Krylov-Schwarz kernel: a Bulk Synchronous Process (BSP)



What happens if, for instance, in this (schematicized) iteration, arithmetic speed is *doubled*, scalar all-gather is *quartered*, and local scatter is *cut by one-third*? Each phase is considered separately. Answer is to the right.







# Estimating scalability of stencil computations

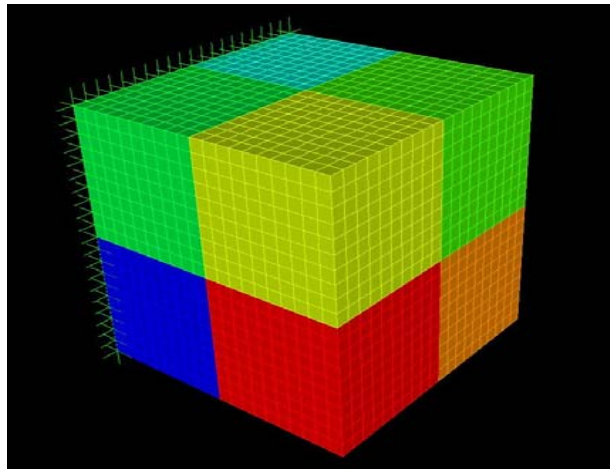
---

- **Given complexity estimates of the leading terms of:**
  - the concurrent computation (per iteration phase)
  - the concurrent communication
  - the synchronization frequency
- **And a bulk synchronous model of the architecture including:**
  - internode communication (network topology and protocol reflecting horizontal memory structure)
  - on-node computation (effective performance parameters including vertical memory structure)
- **One can estimate optimal concurrency and optimal execution time**
  - on per-iteration basis, or overall (by taking into account any granularity-dependent convergence rate)
  - simply differentiate time estimate in terms of  $(N, P)$  with respect to  $P$ , equate to zero and solve for  $P$  in terms of  $N$





# Estimating 3D stencil costs (per iteration)



- grid points in each direction  $n$ , total work  $N=O(n^3)$
- processors in each direction  $p$ , total procs  $P=O(p^3)$
- memory per node requirements  $O(N/P)$
- concurrent execution time per iteration  $A n^3/p^3$
- grid points on side of each processor subdomain  $n/p$
- Concurrent neighbor commun. time per iteration  $B n^2/p^2$
- cost of global reductions in each iteration  $C \log p$  or  $C p^{(1/d)}$ 
  - $C$  includes synchronization frequency
- same dimensionless units for measuring  $A, B, C$ 
  - e.g., cost of scalar floating point multiply-add





# 3D stencil computation illustration

Rich local network, tree-based global reductions

- **total wall-clock time per iteration**

$$T(n, p) = A \frac{n^3}{p^3} + B \frac{n^2}{p^2} + C \log p$$

- **for optimal  $p$ ,  $\frac{\partial T}{\partial p} = 0$  , ~~or~~  $3A \frac{n^3}{p^4} - 2B \frac{n^2}{p^3} + \frac{C}{p} = 0$ ,**

**or (with  $\theta \equiv \frac{32 B^3}{243 A^2 C}$  ),**

$$p_{opt} = \left( \frac{3A}{2C} \right)^{1/3} \left( \left[ 1 + (1 - \sqrt{\theta}) \right]^{1/3} + \left[ 1 - (1 - \sqrt{\theta}) \right]^{1/3} \right) \cdot n$$

- **without “speeddown,”  $p$  can grow with  $n$**
- **in the limit as  $B/C \rightarrow 0$**

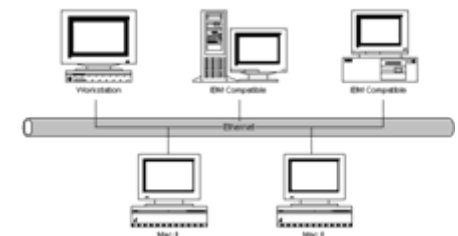
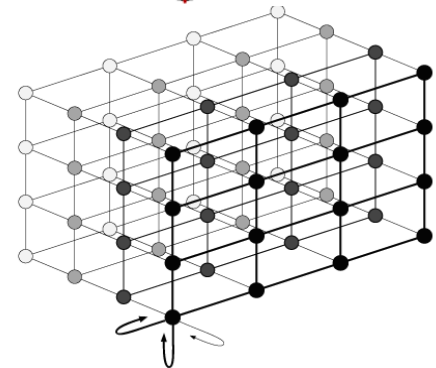
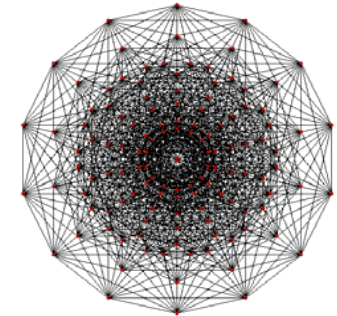
$$p_{opt} = \left( \frac{3A}{C} \right)^{1/3} \cdot n$$





# Scalability results for DD stencil computations

- With tree-based (logarithmic) global reductions and scalable nearest neighbor hardware:
  - optimal number of processors scales *linearly* with problem size
- With 3D torus-based global reductions and scalable nearest neighbor hardware:
  - optimal number of processors scales as *three-fourths* power of problem size (almost “scalable”)
- With common network bus (heavy contention):
  - optimal number of processors scales as *one-fourth* power of problem size (not “scalable”)







# What's under the rug?

---

- **This generic weak scaling type of argument has been made for ten years**
  - **in Petaflops Workshop series (1995 onward)**
  - **in “all-hands” group meetings of SciDAC users (2001 onward)**
- **Why aren't PDEs “humming” on BG/L?**
  - **Of six announced finalists for Bell in 2006, only one is based on PDE simulation, and it achieves only 0.5 Tflop/s on 4K nodes of BG/L**
  - **This compares with 200 Tflop/s on 64K nodes for MD on BG/L – a factor of 25 better in flop/s per node on 16 times more nodes for 400 × performance**





# Contraindications of scalability

---

- **Fixed problem size**
  - **Amdahl-type constraints**
    - “fully resolved” discrete problems (protein folding, network problems)
    - “sufficiently resolved” problems from the continuum
- **Scalable problem size**
  - **Resolution-limited progress in “long time” integration**
    - explicit schemes for time-dependent PDEs
    - suboptimal iterative relaxations schemes for equilibrium PDEs
  - **Nonuniformity of threads**
    - adaptive schemes
    - multiphase computations (e.g, particle and field)





# Amdahl's Law (1967)



- Fundamental limit to strong scaling due to small overheads
  - Independent of number of processors available
  - Analyze by binning code segments by degree of exploitable concurrency and dividing by available processors, up to limit
  - Illustration for just two bins:
    - fraction  $f_1$  of work that is purely sequential
    - fraction  $(1-f_1)$  of work that is arbitrarily concurrent
  - Wall clock time for  $p$  processors
  - Speedup  $\propto f_1 + (1 - f_1) / p$ 
    - for  $f_1=0.01$
  - Applies to any performance enhancement, not just parallelism
- $$= 1 / [f_1 + (1 - f_1) / p]$$

$p$	1	10	100	1000	10000
S	1.0	9.2	50.3	91.0	99.0





# Resolution-limited progress (weak scaling)

- Illustrate for CFL-limited explicit time stepping

- Parallel wall clock time

$$\propto T S^{1+\alpha/d} P^{\alpha/d}$$

- Example: explicit wave problem in 3D ( $\alpha=1$ ,  $d=3$ )

Domain	$10^3 \times 10^3 \times 10^3$	$10^4 \times 10^4 \times 10^4$	$10^5 \times 10^5 \times 10^5$
Exe. time	1 day	10 days	3 months

- Example: explicit diffusion problem in 2D ( $\alpha=2$ ,  $d=2$ )

Domain	$10^3 \times 10^3$	$10^4 \times 10^4$	$10^5 \times 10^5$
Exe. time	1 day	3 months	27 years

$d$ -dimensional domain, length scale  $L$

$d+1$ -dimensional space-time, time scale  $T$

$h$  mesh cell size

$\tau$  time step size

$\tau = O(h^\alpha)$  bound on time step

$n = L/h$  number of mesh cells in each dim

$N = n^d$  number of mesh cells overall

$M = T/\tau$  number of time steps overall

$O(N)$  total work to perform one time step

$O(MN)$  total work to solve problem

$P$  number of processors

$S$  storage per processor

$PS$  total storage on all processors ( $=N$ )

$O(MN/P)$  parallel wall clock time

$$\propto (T/\tau)(PS)/P \propto T S^{1+\alpha/d} P^{\alpha/d}$$

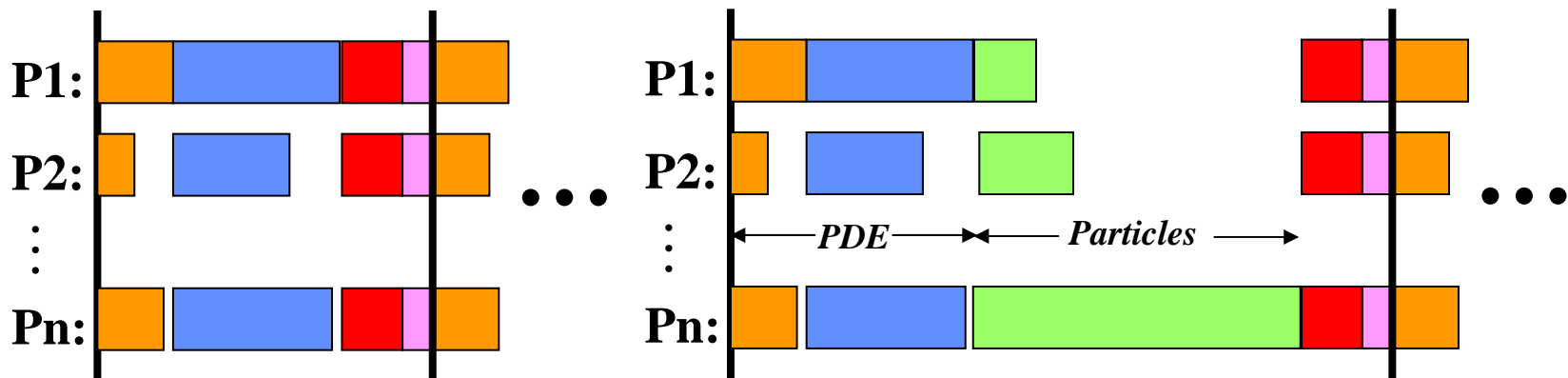
(since  $\tau \propto h^\alpha \propto 1/n^\alpha = 1/N^{\alpha/d} = 1/(PS)^{\alpha/d}$ )





# Thread nonuniformity

- Evolving state of the simulation can spoil load balance
  - adaptive scheme
    - local mesh refinement
    - local time adaptivity
  - state-dependent work complexity
    - complex constitutive or reaction terms
    - nonlinear inner loops with variable convergence rates
  - multiphase simulation
    - bulk synchronous alternation between different phases with different work distributions







# Algorithmic adaptation

---

- No computer system is well balanced for *all* computational tasks, or even for all phases of a *single* well-defined task, like solving nonlinear systems arising from discretized differential equations
- Given the need for high performance in the solution of these and related systems, one should be aware of which computational phases are limited by which aspect of hardware or software.
- With this knowledge, one can design algorithms to “play to” the strengths of a machine of given architecture, or one can intelligently select or evolve architectures for preferred algorithms.





## Four potential limiters on scalability in large-scale parallel scientific codes

---

- **Insufficient localized concurrency**
- **Load imbalance at synchronization points**
- **Interprocessor message latency**
- **Interprocessor message bandwidth**

“horizontal aspects”





# Four potential limiters on arithmetic performance

---

- **Memory latency**
  - Failure to predict which data items are needed
- **Memory bandwidth**
  - Failure to deliver data at consumption rate of processor
- **Load/store instruction issue rate**
  - Failure of processor to issue enough loads/stores per cycle
- **Floating point instruction issue rate**
  - Low percentage of floating point operations among all operations

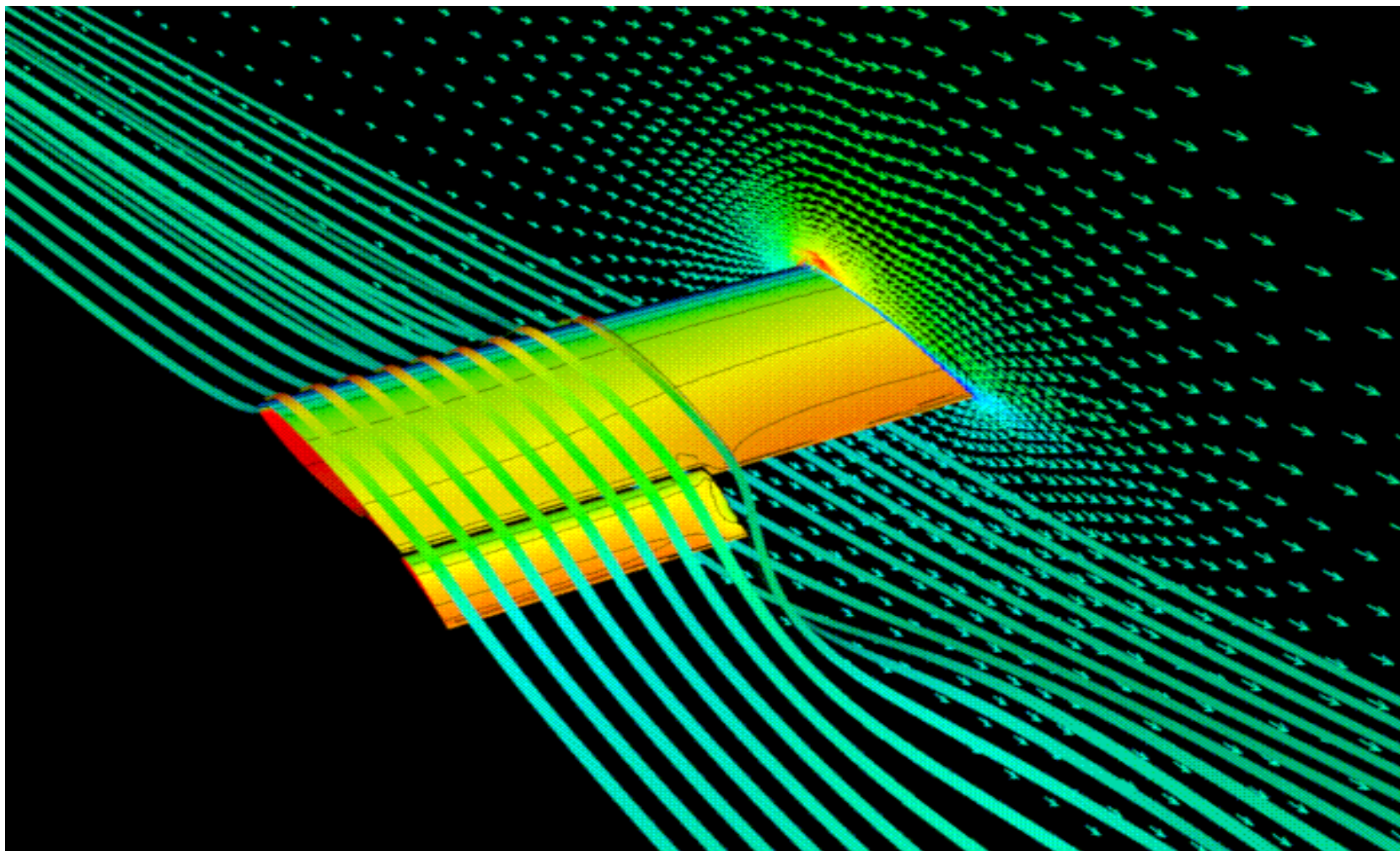
“vertical aspects”





# Application Domain: Computational Aerodynamics

---







# Euler Simulation

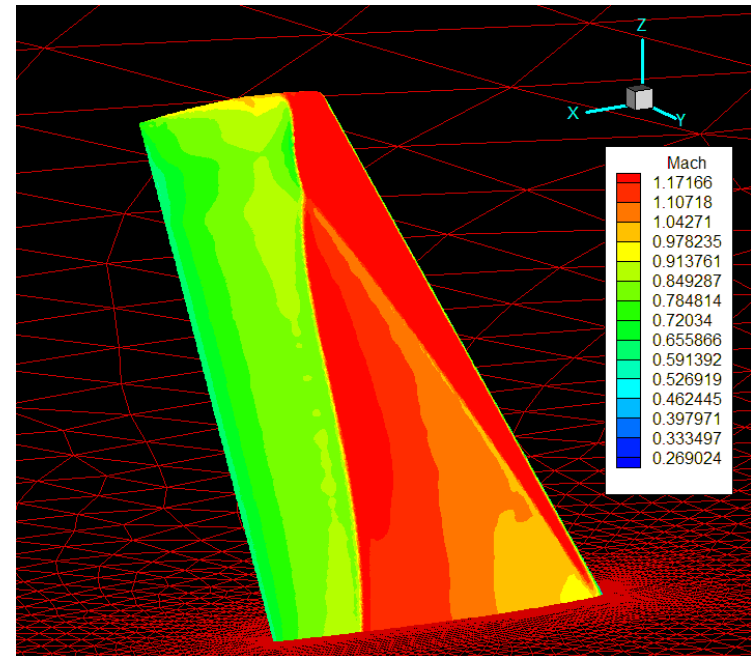
- 3D transonic flow over ONERA M6 wing, at  $3.06^\circ$  angle of attack (exhibits  $\lambda$ -shock at  $M = 0.839$ )

• Solve 
$$\frac{\partial Q}{\partial t} + \frac{1}{V} \oint_{\Omega} (\vec{F} \cdot \hat{n}) d\Omega = 0$$
 where

$$Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \end{bmatrix} \quad \vec{F} \cdot \hat{n} = \begin{bmatrix} \rho U \\ \rho U u + \hat{n}_x p \\ \rho U v + \hat{n}_y p \\ \rho U w + \hat{n}_z p \\ (E + p)U \end{bmatrix}$$

$$U = \hat{n}_x u + \hat{n}_y v + \hat{n}_z w$$

$$p = (\gamma - 1) \left[ E - \rho \frac{(u^2 + v^2 + w^2)}{2} \right]$$



$\rho$  = density,  $\mathbf{u}$  = velocity,  $p$  = pressure  
 $E$  = energy density





# Background of FUN3D Application

---

- **Tetrahedral vertex-centered unstructured grid code developed by W. K. Anderson (NASA) for steady compressible and incompressible Euler and Navier-Stokes**
- **Used in airplane, automobile, and submarine applications for analysis and design**
- **Standard discretization is second-order Roe scheme for convection and Galerkin for diffusion**
- **Newton-Krylov solver with global point-block-ILU preconditioning, with false timestepping for nonlinear continuation towards steady state; competitive with FAS multigrid in practice**
- **Legacy implementation/ordering is vector-oriented**





# Features of FUN3D Application

---

- Based on “legacy” (but contemporary) CFD application with significant F77 code reuse
- Portable, message-passing library-based parallelization, run on NT boxes through Tflop/s ASCII platforms
- Simple multithreaded extension between processors sharing memory physically
- Sparse, unstructured data, implying memory indirection with only modest reuse
- Wide applicability to other implicitly discretized multiple-scale PDE workloads
- Extensive profiling has led to follow-on algorithmic research





# Merits of NKS Algorithm/Implementation

---

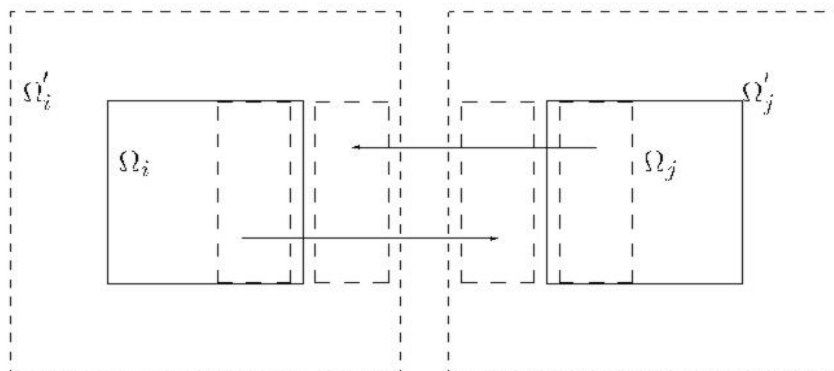
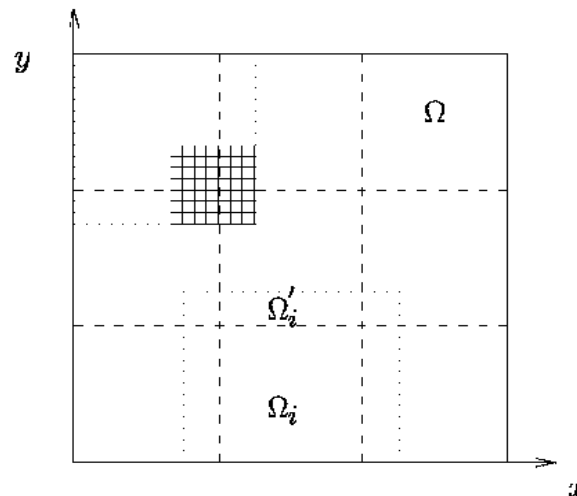
- **Relative characteristics: the “exponents” are *naturally* good**
  - **Convergence scalability**
    - weak (or no) degradation in problem size and parallel granularity (with use of small global problems in Schwarz preconditioner)
  - **Implementation scalability**
    - no degradation in ratio of surface communication to volume work (in problem-scaled limit)
    - only modest degradation from global operations (for sufficiently richly connected networks)
- **Absolute characteristics: the “constants” can be *made* good**
  - **Operation count complexity**
    - residual reductions of  $10^{-9}$  in  $10^3$  “work units”
  - **Per-processor performance**
    - up to 25% of theoretical peak
- **Overall, machine-epsilon solutions require as little as 15 microseconds per degree of freedom!**





# Additive Schwarz Preconditioning for $Au=f$ in $\Omega$

- Form preconditioner  $B$  out of (approximate) local solves on (overlapping) subdomains
- Let  $R_i$  and  $R_i^T$  be Boolean gather and scatter operations, mapping between a global vector and its  $i^{th}$  subdomain support



$$A_i = R_i A R_i^T$$

$$B_i = R_i^T \tilde{A}_i^{-1} R_i$$

$$B = \sum_i B_i$$





# Iteration Count Estimates from the Schwarz Theory

[ref: Smith, Bjorstad & Gropp, 1996, Camb. Univ. Pr.]

- Krylov-Schwarz iterative methods typically converge in a number of iterations that scales as the square-root of the condition number of the Schwarz-preconditioned system
- In terms of  $N$  and  $P$ , where for  $d$ -dimensional isotropic problems,  $N=h^{-d}$  and  $P=H^{-d}$ , for mesh parameter  $h$  and subdomain diameter  $H$ , iteration counts may be estimated as follows:

Preconditioning Type	in 2D	in 3D
Point Jacobi	$O(N^{1/2})$	$O(N^{1/3})$
Domain Jacobi	$O((NP)^{1/4})$	$O((NP)^{1/6})$
1-level Additive Schwarz	$O(P^{1/3})$	$O(P^{1/3})$
2-level Additive Schwarz	$O(1)$	$O(1)$





# Time-Implicit Newton-Krylov-Schwarz Method

For nonlinear robustness, NKS iteration is wrapped in time-stepping.

```
for (l = 0; l < n_time; l++) {  
  select time step  
  for (k = 0; k < n_Newton; k++) {  
    compute nonlinear residual and Jacobian  
    for (j = 0; j < n_Krylov; j++) {  
      forall (i = 0; i < n_Precon ; i++) {  
        solve subdomain problems concurrently  
      }  
      perform preconditioned Jacobian-vector product  
      enforce Krylov basis conditions  
      update optimal coefficients  
      check linear convergence  
    }  
    perform DAXPY update  
    check nonlinear convergence  
  }  
}
```

Steps in red involve global communication.





# Key Features of Implementation Strategy

---

- Subdomain partitioning by one of the MeTiS graph algorithms
- SPMD “owner computes” PETSc implementation under the dual objectives of minimizing the number of messages and overlapping communication with computation
- Each processor “ghosts” its stencil dependences in its neighbors
- Ghost nodes ordered after contiguous owned nodes
- Domain mapped from (user) global ordering into local orderings
- Scatter/gather operations created between *local sequential* vectors and *global distributed* vectors, based on runtime connectivity patterns
- Newton-Krylov-Schwarz operations translated into local tasks and communication tasks
- Profiling used to help eliminate performance bugs in communication and memory hierarchy





# Background of PETSc

---

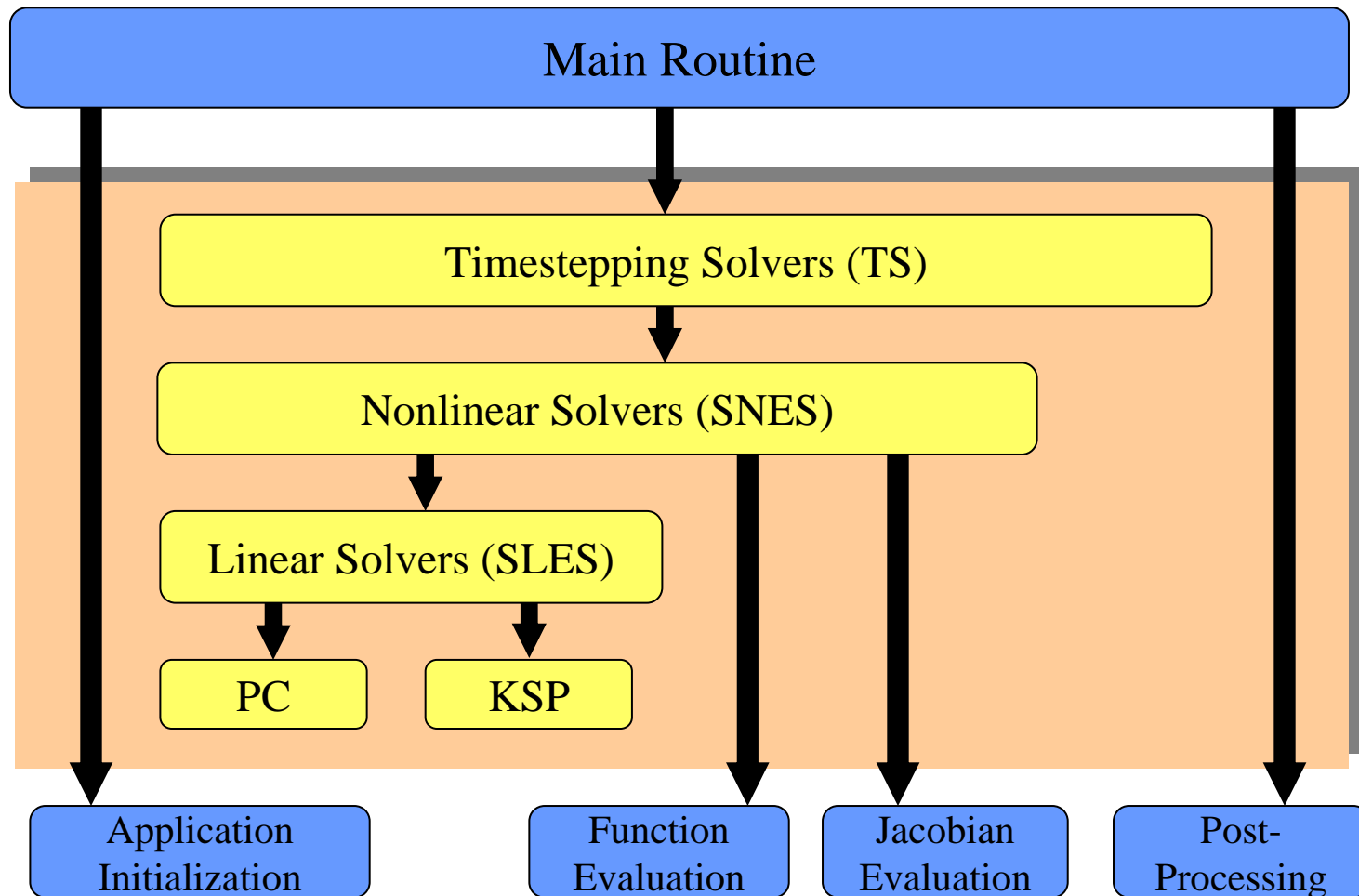
- Developed by Gropp, Smith, McInnes & Balay (ANL) to support research, prototyping, and production parallel solutions of operator equations in message-passing environments
- Distributed data structures as fundamental objects - index sets, vectors/gridfunctions, and matrices/arrays
- Iterative linear and nonlinear solvers, combinable modularly and recursively, and extensibly
- Portable, and callable from C, C++, Fortran
- Uniform high-level API, with multi-layered entry
- Aggressively optimized: copies minimized, communication aggregated and overlapped, caches and registers reused, memory chunks preallocated, inspector-executor model for repetitive tasks (e.g., gather/scatter)
- Now part of the Terascale Optimal PDE Simulations project (DOE SciDAC)

See <http://www.mcs.anl.gov/petsc>





# Separation of Concerns between User Code and PETSc Library



◆ User code

◆ PETSc code





# Outline for PDE Performance Study

---

- **General characterization of PDE requirements**
- **Identification of common algorithmic building blocks**
- **Simple complexity characterizations (computational work, interprocessor communication, intraprocessor data motion)**
- **Identification and illustration of bottlenecks on some of today's important platforms**
- **Experiments with a high-performance port of a NASA aerodynamic design code and with a sparse unstructured matrix-vector kernel**
- **Speculation on useful algorithmic research directions**





# Variety and Complexity of PDEs

---

- Varieties of PDEs
  - evolution (time hyperbolic, time parabolic)
  - equilibrium (elliptic, spatially hyperbolic or parabolic)
  - mixed, varying by region
  - mixed, of multiple type (e.g., parabolic with elliptic constraint)
- Complexity parameterized by:
  - spatial grid points,  $N_x$
  - temporal grid points,  $N_t$
  - components per point,  $N_c$
  - auxiliary storage per point,  $N_a$
  - grid points in stencil,  $N_s$
- Memory:  $M \approx N_x \bullet (N_c + N_a + N_c \bullet N_c \bullet N_s)$
- Work:  $W \approx N_x \bullet N_t \bullet (N_c + N_a + N_c \bullet N_c \bullet N_s)$





# Explicit Solvers

---

$$u^l = u^{l-1} - \Delta t^l \cdot f(u^{l-1})$$

- **Concurrency is pointwise,  $O(N)$**
- **Comm.-to-Comp. ratio is surface-to-volume,  $O((N/P)^{1/3})$**
- **Communication range is nearest-neighbor, except for time-step computation**
- **Synchronization frequency is once per step,  $O((N/P)^{-1})$**
- **Storage per point is low**
- **Load balance is straightforward for static quasi-uniform grids**
- **Grid adaptivity (together with temporal stability limitation) makes load balance nontrivial**





# Domain-decomposed Implicit Solvers

---

$$\frac{u^l}{\Delta t^l} + f(u^l) = \frac{u^{l-1}}{\Delta t^l}, \Delta t^l \rightarrow \infty$$

- Concurrency is pointwise,  $O(N)$ , or subdomainwise,  $O(P)$
- Comm.-to-Comp. ratio still *mainly* surface-to-volume,  $O((N/P)^{-1/3})$
- Communication still *mainly* nearest-neighbor, but nonlocal communication arises from conjugation, norms, coarse grid problems
- Synchronization frequency *often more* than once per grid-sweep, up to Krylov dimension,  $O(K(N/P)^{-1})$
- Storage per point is higher, by factor of  $O(K)$
- Load balance issues the same as for explicit





# Resource Scaling for PDEs

---

- For 3D problems, work is proportional to four-thirds power of memory, because
  - For equilibrium problems, work scales with problem size times number of iteration steps -- proportional to resolution in single spatial dimension
  - For evolutionary problems, work scales with problems size times number of time steps -- CFL arguments place latter on order of spatial resolution, as well
- Proportionality constant can be adjusted over a very wide range by both discretization (high-order implies more work per point and per memory transfer) and by algorithmic tuning
- If frequent time frames are to be captured, other resources -- disk capacity and I/O rates -- must both scale linearly with work, more stringently than for memory.





# Primary PDE Solution Kernels

(assumes vertex-based; dual statements for cell-based)

---

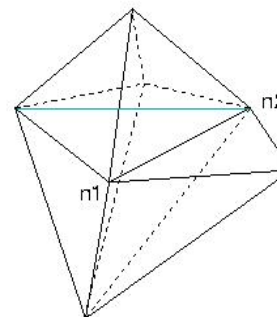
- **Vertex-based loops**
  - state vector and auxiliary vector updates
- **Edge-based “stencil op” loops**
  - residual evaluation
  - approximate Jacobian evaluation
  - Jacobian-vector product (often replaced with matrix-free form, involving residual evaluation)
  - intergrid transfer (coarse/fine)
- **Sparse, narrow-band recurrences**
  - approximate factorization and back substitution
  - smoothing
- **Vector inner products and norms**
  - orthogonalization/conjugation
  - convergence progress and stability checks





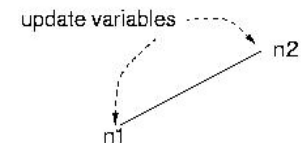
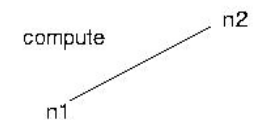
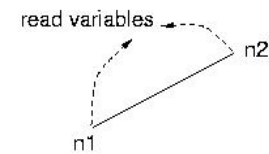
# Illustration of Edge-based Loop

- Vertex-centered grid
- Traverse by edges
  - load vertex values
  - compute intensively
    - e.g., for compressible flows, solve 5x5 eigen-problem for characteristic directions and speeds of each wave
  - store flux contributions at vertices
- Each vertex appears in approximately 15 flux computations



Variables at each node:  
density,  
momentum  $(x,y,z)$ ,  
energy,  
pressure

Variables at edge:  
identity of nodes,  
orientation  $(x,y,z)$   
normal area







# Complexities of PDE Kernels

---

- **Vertex-based loops**
  - work and data closely proportional
  - pointwise concurrency, no communication
- **Edge-based “stencil op” loops**
  - large ratio of work to data
  - colored edge concurrency; local communication
- **Sparse, narrow-band recurrences**
  - work and data closely proportional
  - frontal concurrency; no, local, or global communication
- **Vector inner products and norms**
  - work and data closely proportional
  - pointwise concurrency; global communication





# Candidate stresspoints of PDE kernels

---

- **Vertex-based loops**
  - memory bandwidth
- **Edge-based “stencil op” loops**
  - load/store (register-cache) bandwidth
  - internode bandwidth
- **Sparse, narrow-band recurrences**
  - memory bandwidth
  - internode bandwidth, internode latency, network diameter
- **Inner products and norms**
  - memory bandwidth
  - internode latency, network diameter





## Observation #1:

# Processor scalability no problem, in principle

---

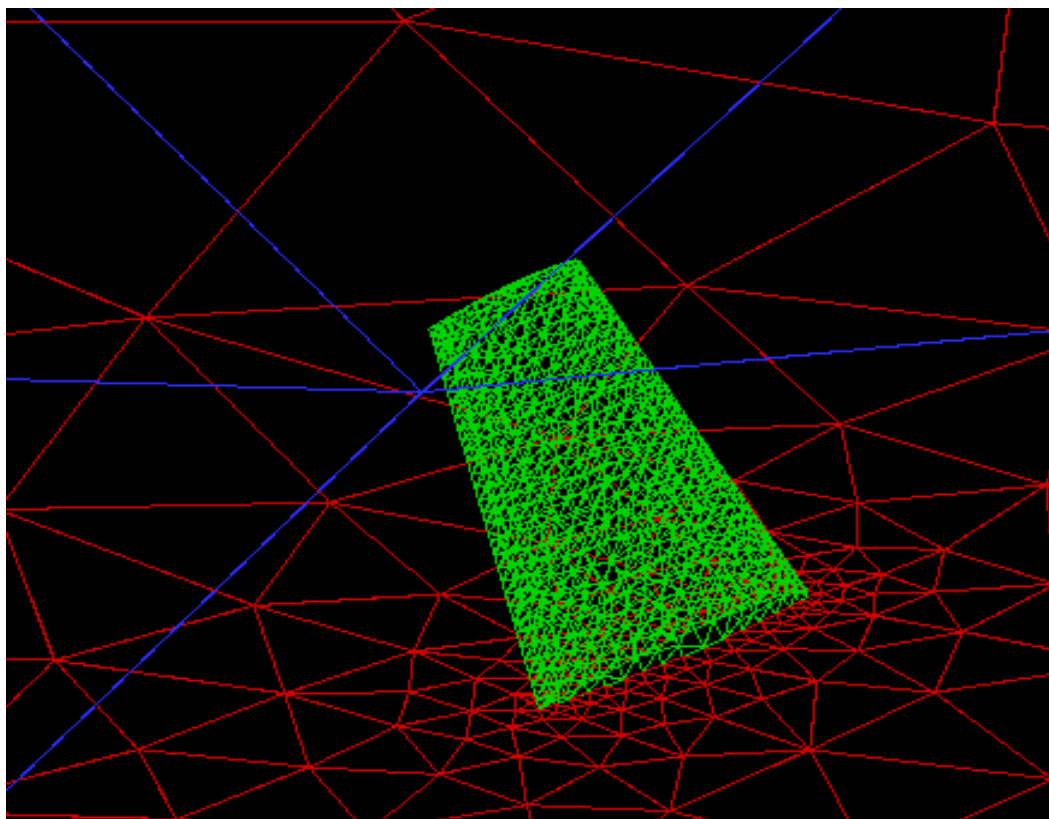
- As popularized with the 1986 Karp Prize paper of Benner, Gustafson & Montry, Amdahl's law can be defeated if serial (or bounded concurrency) sections make up a decreasing fraction of total work as problem size and processor count scale --- true for most iterative implicit nonlinear PDE solvers
- Simple, back-of-envelope parallel complexity analyses show that processors can be increased as fast, or almost as fast, as problem size, assuming load is perfectly balanced
- Caveat: the processor network must also be scalable (applies to protocols as well as to hardware); machines based on common bus networks will not scale





# Surface Visualization of Test Domain for Euler Flow over an ONERA M6 Wing

- Wing surface outlined in **green triangles**, **farfield blue**, **symmetry plane red**
- 2.8 M vertices in the actual computational domain (9K in image below)







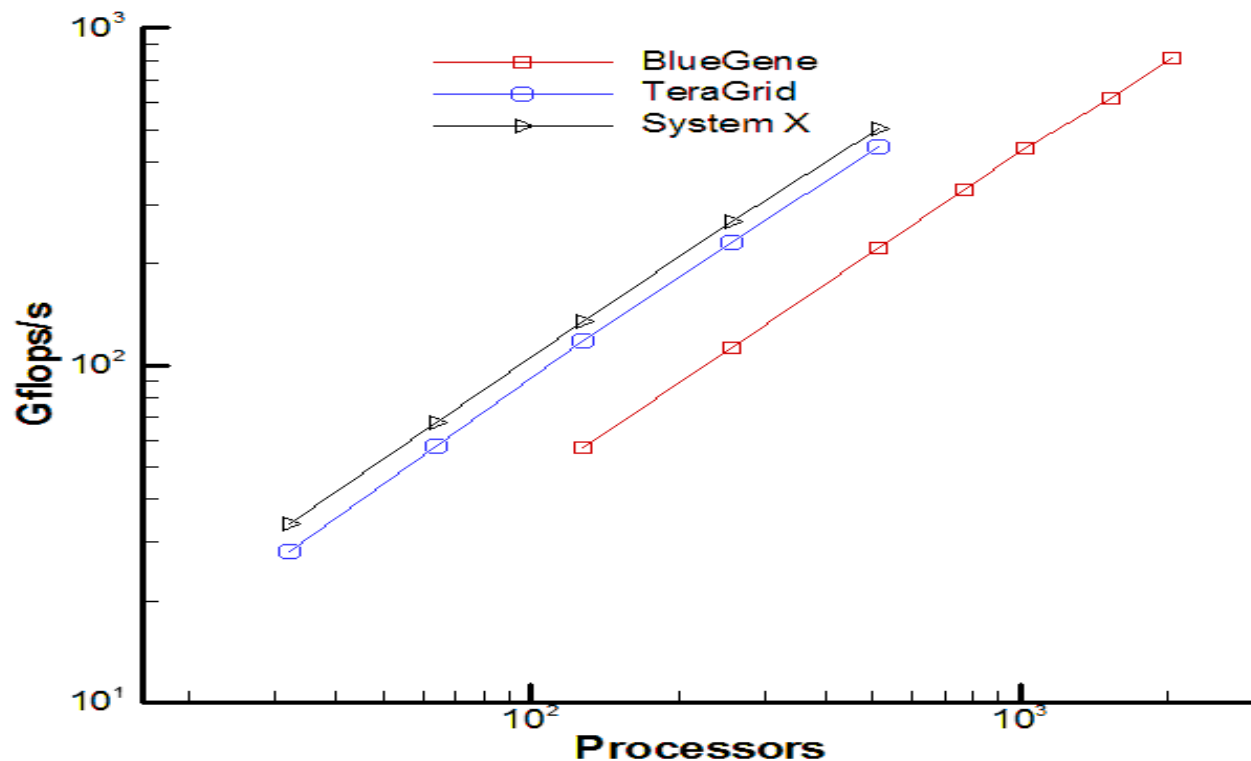
# Parallel Performance of PETSc-FUN3D

3D Mesh: 2,761,774 Vertices and 18,945,809 Edges

TeraGrid: Dual 1.5 GHz Intel Madison Processors with 4 MB L2 Cache

BlueGene: Dual 700 MHz IBM Processors with 4 MB L3 Cache

System X: Dual 2.3 GHz PowerPC 970FX processors with 0.5 MB L2 Cache

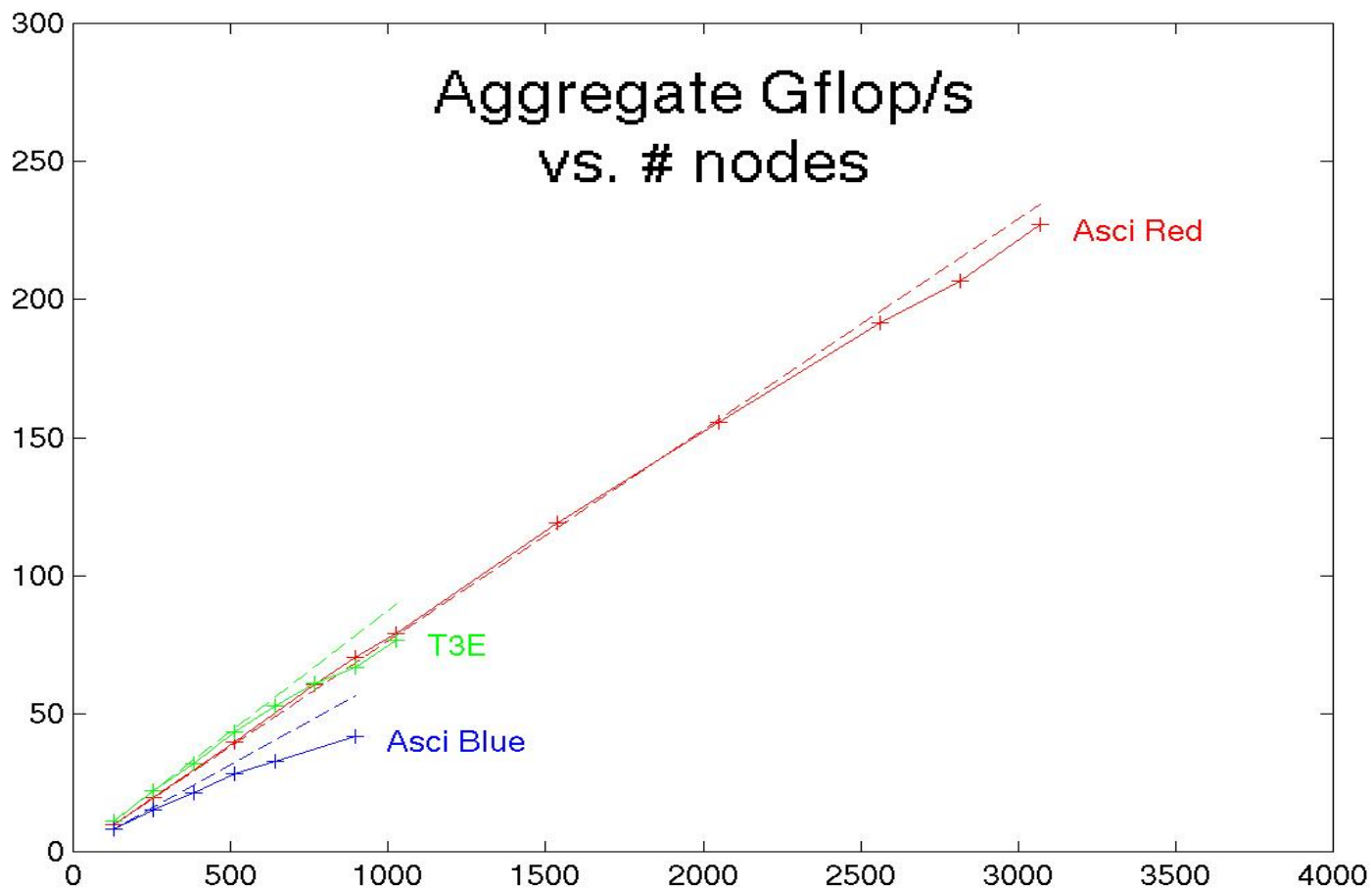






# Fixed-size Parallel Scaling Results (Flop/s)

Results on older generation of machines







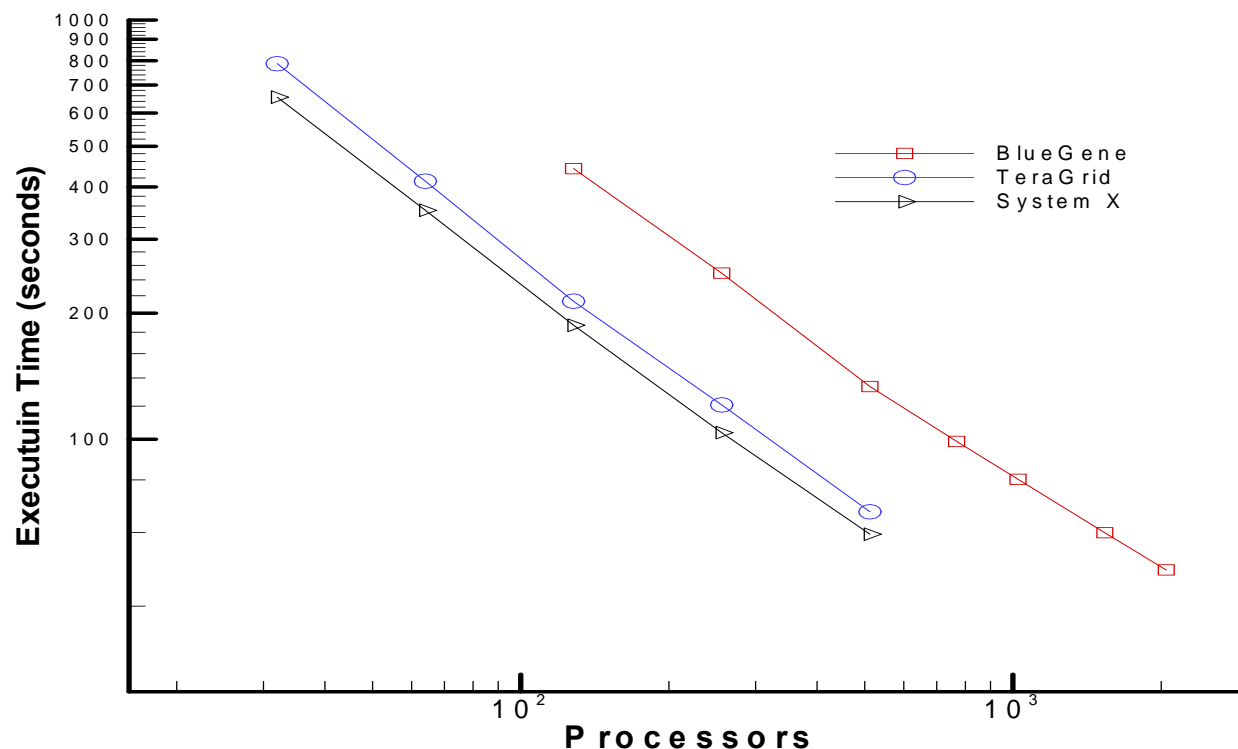
# Parallel Performance of PETSc-FUN3D

3D Mesh: 2,761,774 Vertices and 18,945,809 Edges

TeraGrid: Dual 1.5 GHz Intel Madison Processors with 4 MB L2 Cache

BlueGene: Dual 700 MHz IBM Processors with 4 MB L3 Cache

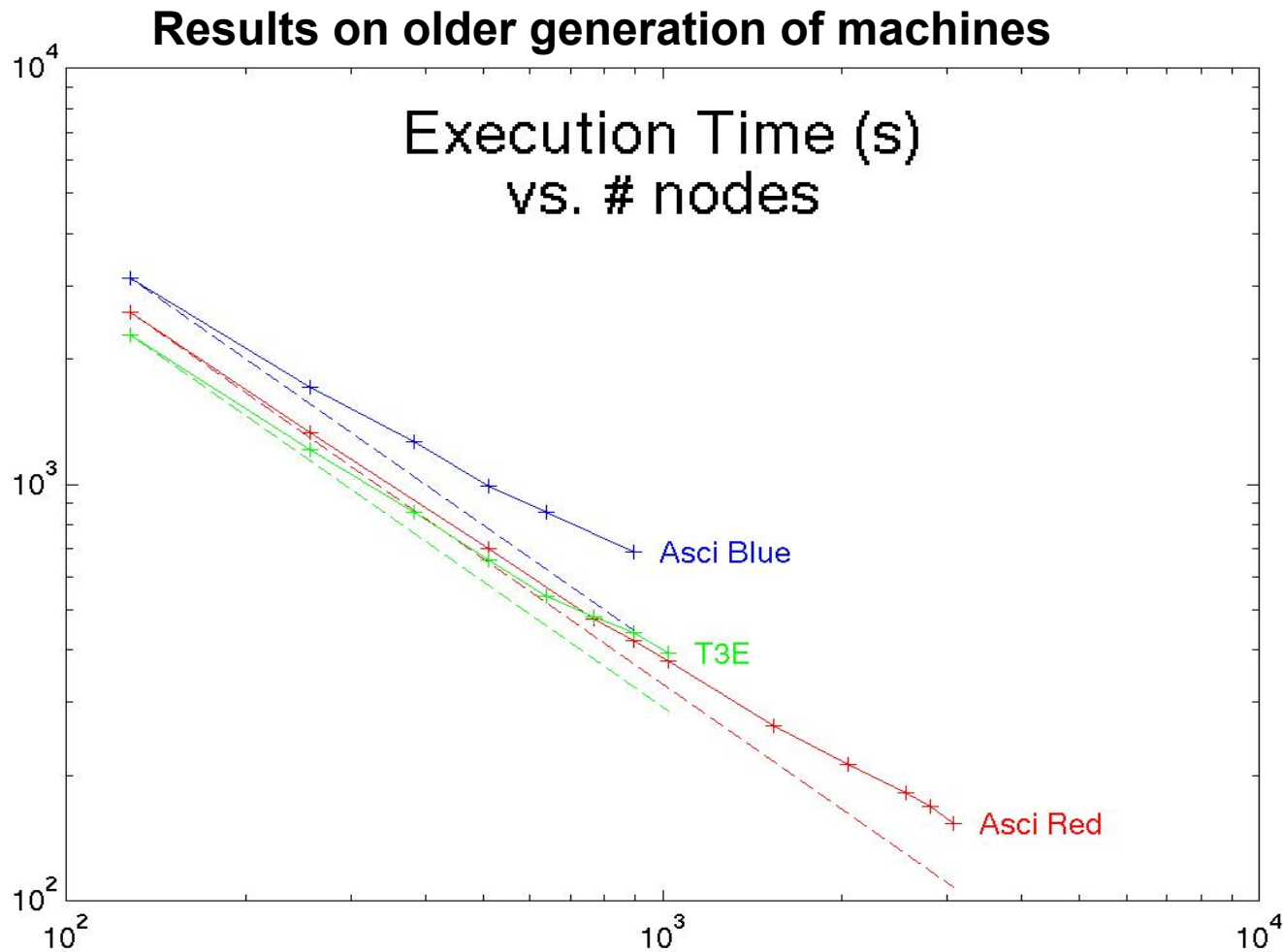
System X: Dual 2.3 GHz PowerPC 970FX processors with 0.5 MB L2 Cache







# Fixed-size Parallel Scaling Results (seconds)

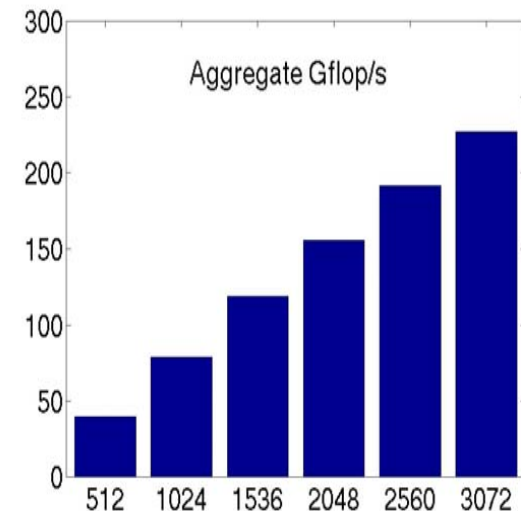
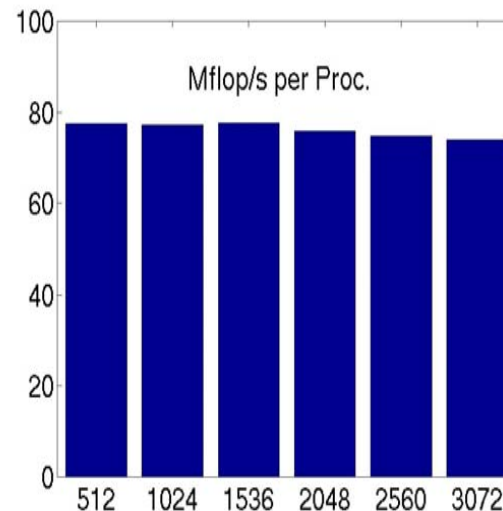
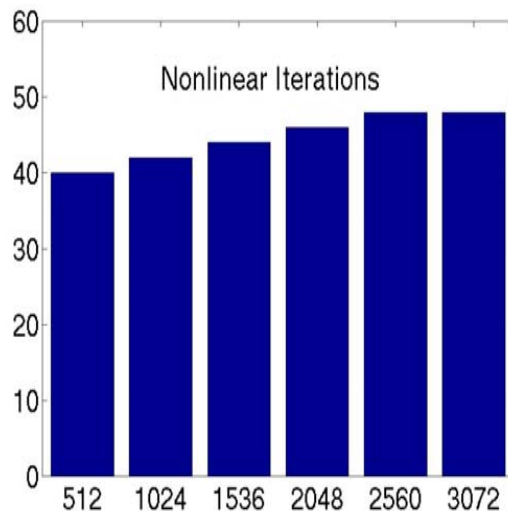
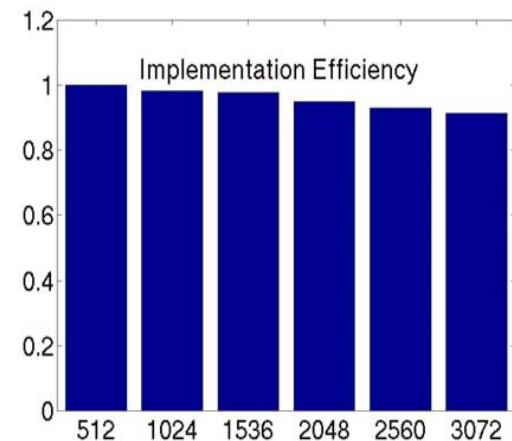
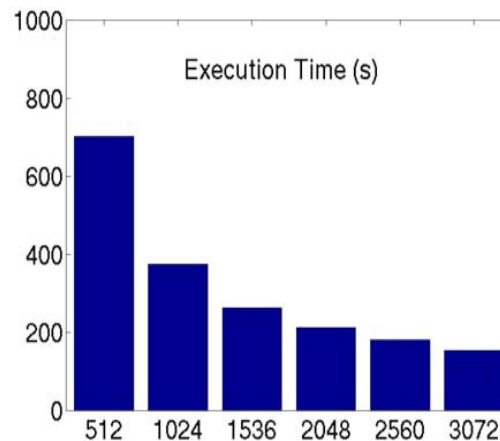
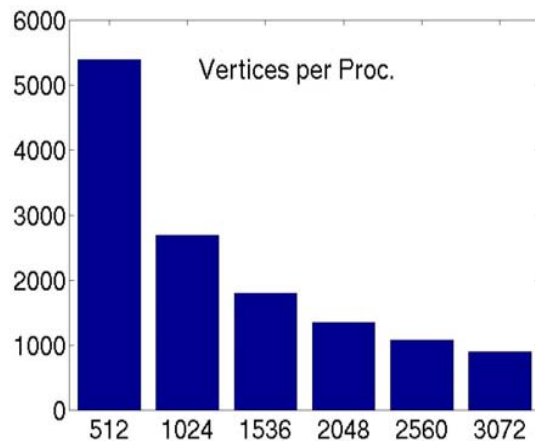






# Inside Parallel Scaling Results on ASCI Red

ONERA M6 Wing Test Case, Tetrahedral grid of 2.8 million vertices (about 11 million unknowns) on up to 3072 ASCI Red nodes (each with dual Pentium Pro 333 MHz processors)







## Observation #2 (for Fixed-Size Problems): Synchronization eventually a bottleneck

---

- Percentage of time spent in communication phases on ASCI Red for NKS unstructured Euler simulation
- Principal nonscaling feature is synchronization at global inner products and norms, while cost of halo exchange grows slowly even for fixed-size problem with deteriorating surface-to-volume

Number of Processors	Global reductions	Synchronizations	Halo Exchanges
128	5%	4%	3%
256	3%	6%	4%
512	3%	7%	5%
768	3%	8%	5%
1024	3%	10%	6%





## Observation #3:

### Memory latency no problem, in principle

---

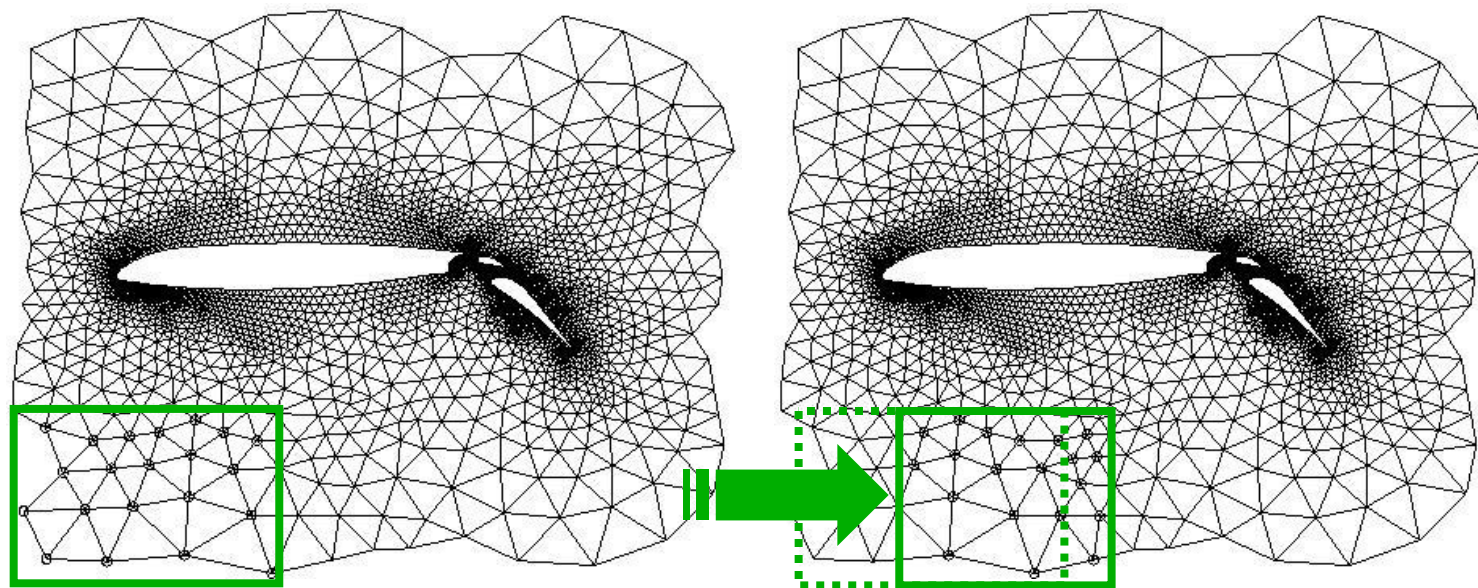
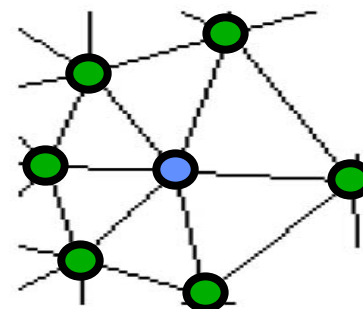
- Regularity of reference in static grid-based computations can be exploited through memory-assist features to cover latency
- PDEs have simple, periodic workingset structure that permits effective use of prefetch/dispatch directives, and lots of slackness (process concurrency in excess of hardware concurrency)
- Combined with coming processors-in-memory (PIM) technology for gather/scatter into densely used block transfers and multithreading for latency that cannot be amortized by sufficiently large block transfers, the solution of PDEs can approach zero stall conditions
- Caveat: high bandwidth is critical to covering latency





# Workingset Characterization of Memory Traffic

- **Smallest:** data for single stencil
- **Largest:** data for entire subdomain
- **Intermediate:** data for a neighborhood collection of stencils, reused as many times as possible

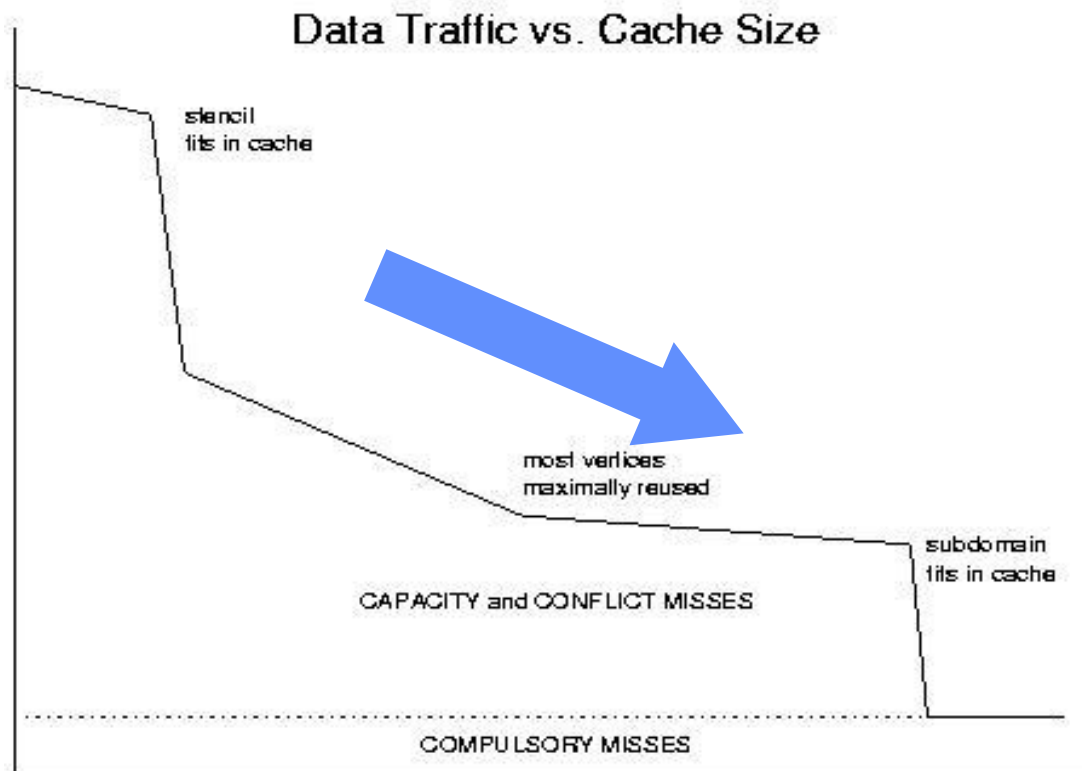






# Thought Experiment: Cache Traffic for PDEs

- As successive workingsets ``drop'' into a level of memory, capacity (and with effort conflict) misses disappear, leaving only compulsory, reducing demand on main memory bandwidth







# **BW-stretching Strategies Based on Workingsets**

- **No performance value in memory levels larger than subdomain**
- **Little performance value in memory levels smaller than subdomain but larger than required to permit full reuse of most data within each subdomain subtraversal (middle knee, prev. slide)**
- **After providing L1 large enough for smallest workingset (and multiple independent copies up to desired level of multithreading, if necessary all additional resources should be invested in large L2**
- **Tables describing grid connectivity are built (after each grid rebalancing) and stored in PIM --- used to pack/unpack dense-use cache lines during subdomain traversal**





# Three Types of Locality Enhancements

---

- ***Edge-reordering*** for maximal vertex reuse
- ***Field interlacing*** for maximal cache-line reuse
  - use  $U1, V1, W1, U2, V2, W2, \dots, Un, Vn, Wn$
  - rather than  $U1, U2, \dots, Un, V1, V2, \dots, Vn, W1, W2, \dots, Wn$
- ***Sparse Jacobian blocking*** for minimal integer metadata in manipulating a given amount of floating point physical data





# Improvements Resulting from Locality Reordering

Processor	Clock MHz	Peak Mflop/s	Opt. % of Peak	Opt. Mflop/s	Reord. Only Mflop/s	Interl. only Mflop/s	Orig. Mflop/s	Orig. % of Peak
R10000	250	500	25.4	127	74	59	26	5.2
P3	200	800	20.3	163	87	68	32	4.0
P2SC (2 card)	120	480	21.4	101	51	35	13	2.7
P2SC (4 card)	120	480	24.3	117	59	40	15	3.1
604e	332	664	9.9	66	43	31	15	2.3
Alpha 21164	450	900	8.3	75	39	32	14	1.6
Alpha 21164	600	1200	7.6	91	47	37	16	1.3
Ultra II	300	600	12.5	75	42	35	18	3.0
Ultra II	360	720	13.0	94	54	47	25	3.5
Ultra II/HPC	400	800	8.9	71	47	36	20	2.5
Pent. II/LIN	400	400	20.8	83	52	47	33	8.3
Pent. II/NT	400	400	19.5	78	49	49	31	7.8
Pent. Pro	200	200	21.0	42	27	26	16	8.0
Pent. Pro	333	333	18.8	60	40	36	21	6.3





## Observation #4: Memory bandwidth a major bottleneck

---

Execution times for NKS Euler Simulation on Origin 2000:  
(standard) **double** precision matrices versus **single** precision

Number of Processors	Computational Phase			
	Linear Solve		Overall	
	Double	Single	Double	Single
16	223s	136s	746s	657s
32	117s	67s	373s	331s
64	60s	34s	205s	181s
120	31s	16s	122s	106s

Note that times are nearly halved, along with precision, for the BW-limited linear solve phase, indicating that the BW can be at least doubled before hitting the next bottleneck!





# ASCI Memory Bandwidth Bottleneck

---

- Per-processor memory bandwidth versus rate of work
  - approximately 10-15 flops per word transferred from memory
  - fairly constant across machines, and fairly poor without extensive reuse

	<b>Peak (MF/s)</b>	<b>BW/proc (MW/s)</b>	<b>(MF/s)/ (MW/s)</b>
<b>White</b>	<b>1500</b>	<b>125.0</b>	<b>12.0</b>
<b>Blue Mtn</b>	<b>500</b>	<b>48.8</b>	<b>10.2</b>
<b>Blue Pac</b>	<b>666</b>	<b>45.0</b>	<b>14.8</b>
<b>Red</b>	<b>333</b>	<b>33.3</b>	<b>10.0</b>





# Implications of Bandwidth Limitations in Shared Memory Systems



- The processors on a node compete for the available memory bandwidth
- The computational phases that are memory bandwidth limited will not scale and may even run slower due to arbitration
- Stream Benchmark on ASCI Red MB/s for the Triad Operation

Vector Size	1 Thread	2 Threads
1E04	666	1296
5E04	137	238
1E05	140	144
1E06	145	141
1E07	157	152

Larger vectors in last three rows do not fit into cache and are bandwidth-limited





# **BW-stretching Strategies**

## **Based on Multivectors in Sparse Matvecs**

---

- **The sparse matrix-vector multiply (matvec) is one of the most common kernels in scientific computing**
  - **Same data access considerations as stencil-op kernel in explicit methods for PDEs**
  - **Same as Krylov kernel and similar to preconditioner application kernel in implicit methods for PDEs**
- **When multiplying a single vector, each element of the sparse matrix is used exactly once per matvec**
- **If the matrix is large, none of its elements will remain in the cache from one matvec to the next**
- **If multiple vectors, say  $N$ , are multiplied at once, each element of the matrix is reused  $N$  times**
- **A simple complexity model for the sparse matrix-vector product illustrates the issues**





# Matrix-vector Multiplication for a Single Vector

---



```
do i=1, n
  fetch ia(i+1)
  sum = 0
  ! loop over the non-zeros of the row
  do j = ia(i), ia(i+1)-1 {
    fetch ja(j), a(j), x(ja(j))
    sum = sum + a(j) * x(ja(j))
  }
  enddo
  Store sum into y(i)
enddo
```

This version performs  $A \times x$





# Matrix-Vector Multiplication for $N$ Independent Vectors

---



```
do i = 1, n
  fetch ia(i+1)
  ! loop over the non-zeros of the row
  do j = ia(i), ia(i+1) - 1
    fetch ja(j), a(j), x1(ja(j)), .....xN(ja(j))
    do N fmadd (floating multiply add)
  enddo
  Store y1(i) .....yN(i)
enddo
```

A red arrow points from the  $i+1$  in the `fetch ia(i+1)` line to the  $i+1$  in the `do j = ia(i), ia(i+1) - 1` line.

This version performs  $A \times \{x_1, \dots, x_N\}$





# **Estimating the Memory Bandwidth Limitation**

---

- **Assume ideal memory system apart from bandwidth**
  - **Perfect cache (only compulsory misses; no overhead)**
  - **No memory latency**
  - **Unlimited number of loads and stores per cycle**
- **Specify number of rows and nonzeros, sizes for integers and floats**
- **Assume matrix blocking factor and vector blocking factor**
- **Compute data volume associated with sparse matvec**
- **Compute number of floating-point multiply adds (fmadd)**
- **Bytes per floating multiply-add combined with memory bandwidth (bytes/second) give a bound on rate of execution of multiply-adds**





# Sparse Matvec Performance Summary

- On 250 MHz MIPS R10000
- Matrix size = 90,708; number of nonzero entries = 5,047,120, blocksize = 4
- Stream performance is 358 MB/sec (for triad vector operation)  
<http://www.cs.virginia.edu/stream>
- Number of Vectors is either 1 or a block of 4

Format	Number of Vectors	Bytes / fmadd	Bandwidth		MFlops	
			Required	Achieved	Ideal	Achieved
AIJ	1	12.36	3090	276	58	45
AIJ	4	3.31	827	221	216	120
BAIJ	1	9.31	2327		84	55
BAIJ	4	2.54	635	229	305	175

- Ratio of 2.7 for **AIJ** and 3.2 for **BAIJ** in going from 1 to 4





## Performance Summary on 2.4 GHz P4 Xeon

- Matrix size,  $n = 90,708$ ; number of nonzero entries,  $N_{nz} = 5,047,120$  (from computational aerodynamics,  $b=4$ )
- Stream performance is 1973 MB/sec (for triad vector operation, <http://www.cs.virginia.edu/stream>)
- Number of Vectors,  $N = 1$ , and 4

Format	Number of Vectors	Bytes / flop	Bandwidth (GB/s)		MFlops	
			Required	Measured	Ideal	Achieved
AIJ	1	6.18	14.83	1.97	319	274
AIJ	4	1.66	3.98	1.97	1188	615





# Comparison of Domain-Level Parallelism for MPI and OpenMP/MPI

---

- Table shows execution times of residual flux evaluation phase for W-cycle FAS Euler simulation on ASCI Red (2 processors per node)
- Thread management imposes an overhead of 5% up to more serious levels, depending upon the system
- In computational phases that are not memory bandwidth-limited, shared-memory multithreading can be more efficient than MPI-mediated domain-based multiprocessing

# Nodes	On each node	Sec./W-cycle
128	1 MPI process	14.01
128	2 MPI processes	7.98
128	2 OpenMP threads	7.56
256	1 MPI process	7.59





## Observation #5:

### Load-store functionality may be a bottleneck

---

- Table shows execution times of residual flux evaluation phase for NKS Euler simulation on ASCI Red (2 processors per node)
- In each paradigm, the second processor per node contributes another load/store unit while sharing fixed memory bandwidth
- Note that 1 thread is worse than 1 MPI process, but that 2-thread performance eventually surpass 2-process performance as subdomains become small

Nodes	MPI/OpenMP		MPI	
	1 Thr	2 Thr	1 Proc	2 Proc
256	483s	261s	456s	258s
2560	76s	39s	72s	45s
3072	66s	33s	62s	40s





# Quantifying the Load/Store Bottleneck

---

- Assume ideal memory system apart from load/store units
  - All data items are ready in cache
  - Each operation takes only one cycle to complete but multiple operations can graduate in one cycle
- If only one load or store can be issued in one cycle (as is the case on R10000 and many other processors), the best we can hope for is

$$\frac{\text{Number of floating point instructions}}{\text{Number of Loads and Stores}} * \text{Peak MFlops/s}$$

- Other restrictions (like primary cache latency, latency of floating point units etc.) need to be taken into account while creating the best schedule





## Observation #6: Fraction of Flops may be a Bottleneck

---

```

do i=1, m
  jrow = ia(i+1)           // 1Of, AT, Ld
  ncol = ia(i+1) -ia(i)    // 1 Iop
  Initialize, sum1 .....sumN // N Ld
  do j=1,ncol              // 1 Ld
    fetch ja(jrow), a(jrow), x1(ja(jrow)), .....xN(ja(jrow))
                                // 1 Of, N+2 AT N+2 Ld
    do N fmadd (floating multiply add) // 2N Flop
  enddo                      // 1 Iop, 1 Br
  Store sum1.....sumN in y1(i) .....yN(i) // 1 Of, N AT, and St
enddo                      // 1 Iop, 1 Br

```

**AT**:address transln; **Br**: branch; **Iop**: integer op; **Flop**: floating point op; **Of**: offset calculation; **Ld**: load; **St**: store

- Estimated number of floating point operations out of the total instructions (for the unstructured Euler Jacobian)
  - For  $N=1$ ,  $I_f = 0.18$
  - For  $N = 4$ ,  $I_f = 0.34$ ; this is one-third of peak





# Significance of Multivectors

---

- Using multivectors can improve the performance of sparse matrix-vector product significantly
- “Algorithmic headroom” is available for modest blocking
- Simple models predict the performance of sparse matrix-vector operations on a variety of platforms, including the effects of *memory bandwidth, and instruction issue rates*
  - achievable performance is a small fraction of stated peak for sparse matrix-vector kernels, independent of code quality
  - compiler improvements and intelligent prefetching can help but the problem is fundamentally an architecture-algorithm mismatch and needs an algorithmic solution

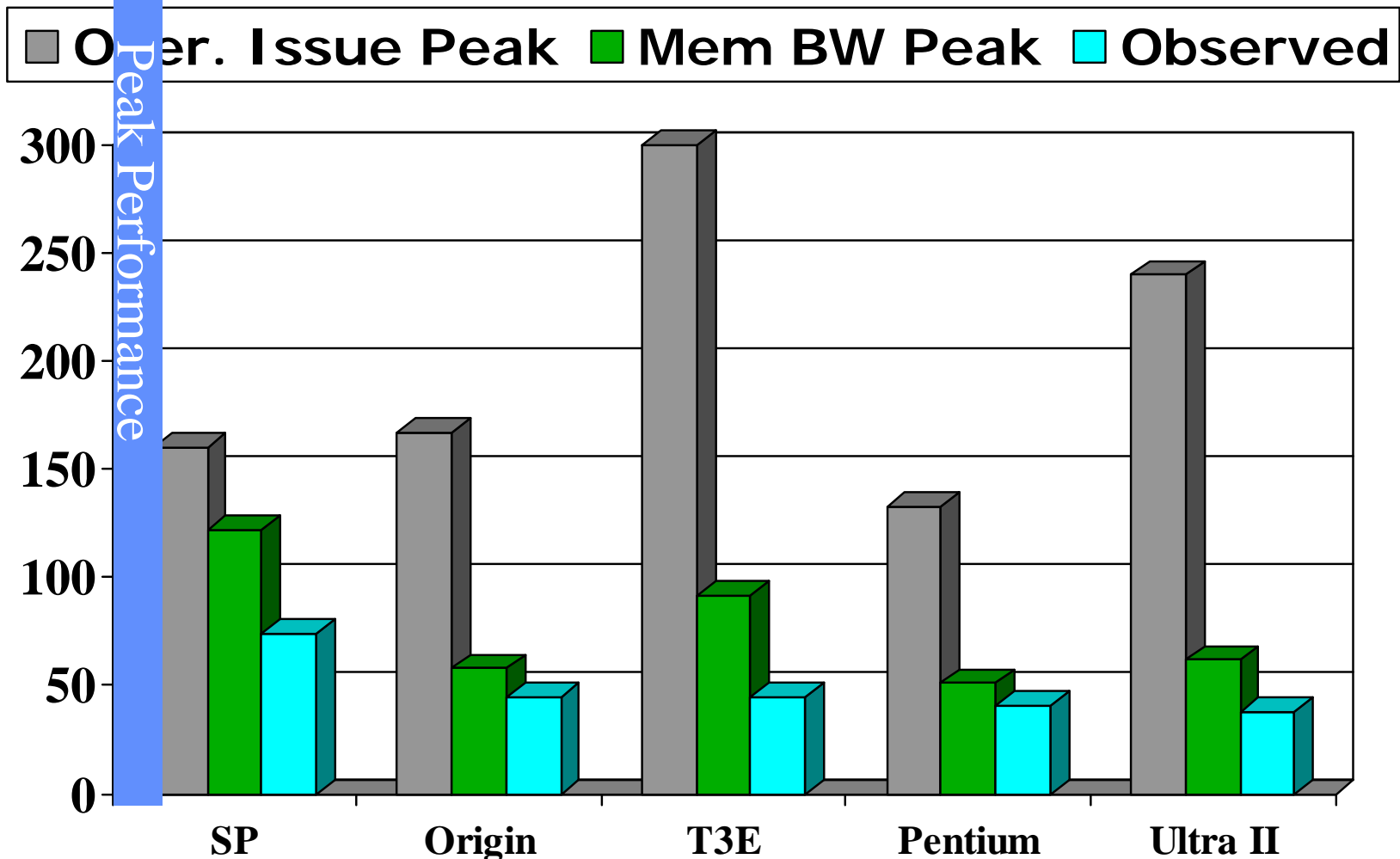




# Realistic Measures of Performance

Sparse Matrix Vector Product

one vector, matrix size = 90,708, nonzero entries = 5,047,120







# Summary of Observations for Simulation Codes

- Processor scalability is no problem, in principle
- Common bus-based network is a bottleneck
- For fixed-size problems, global synchronization is eventually a bottleneck
- Memory latency is no problem, in principle
- Memory bandwidth is a *major* bottleneck
- Load-Store functionality *may* be a bottleneck
- Frequency of floating point instructions *may* be a bottleneck





# **Lessons for High-end Simulation of PDEs**

---

- **Unstructured (static) grid codes can run well on distributed hierarchical memory machines, with attention to partitioning, vertex ordering, component ordering, blocking, and tuning**
- **Parallel solver libraries can give new life to the most valuable, discipline-specific modules of legacy PDE codes**
- **Parallel scalability is easy, but attaining high per-processor performance for sparse problems gets more challenging with each machine generation**
- **The NKS family of algorithms can be and must be tuned to an application-architecture combination; profiling is critical**
- ***Some* gains from hybrid parallel programming models (message passing and multithreading together) require little work; squeezing the last drop is likely much more difficult**





# Weighing in at the Bottom Line

---

- **Characterization of a 1 Teraflop/s computer of today**
  - about 1,000 processors of 1 Gflop/s (peak) each
  - due to inefficiencies within the processors, more practically characterized as about 4,000 processors of 250 Mflop/s each
- **How do we want to get to 1 Petaflop/s?**
  - 1,000,000 processors of 1 Gflop/s each (only wider)?
  - 10,000 processors of 100 Gflop/s each (mainly deeper)?
- **From the point of view of PDE simulations on quasi-static Eulerian grids**
  - **Either!**
- **Caveat: dynamic grid simulations are not directly covered in this discussion**
  - but see work 2003 SIAM/ACM Prize





# Some noteworthy algorithmic adaptations to distributed memory architecture

---



- **Restricted Schwarz in elliptic problems (Cai & Sarkis)**
  - omit every other local communication (actually leads to *better* convergence, now proved)
- **Extrapolated Schwarz in parabolic problems (Garbey & Tromeur-Dervout)**
  - hide interprocessor latency by extrapolating messages received in time integration, with rollback if actual messages have discrepancies in lower Fourier modes (higher mode discrepancies decay anyway)
- **Nonlinear Schwarz in elliptic problems (Cai & Keyes)**
  - reduce global Krylov-Schwarz synchronizations by applying NKS within well-connected subdomains and performing *few* global outer Newton iterations
- **Aggressive coarsening in linear AMG (Falgout, Yang, et al.)**
  - reduce size of coarse problems to trade-off cost per iteration with number of iterations (and many other such preconditioner quality ideas)





# **Four Sources of Performance Improvement**

---

- **Expanded number of processors**
  - arbitrarily large factor, through extremely careful attention to load balancing and synchronization
- **More efficient use of processor cycles, and faster processor/memory elements**
  - one to two orders of magnitude, through memory-assist language features, processors-in-memory, and multithreading
- **Algorithmic variants that are more architecture-friendly**
  - approximately an order of magnitude, through improved locality and relaxed synchronization
- **Algorithms that deliver more “science per flop”**
  - possibly large problem-dependent factor, through adaptivity
  - **This last does not contribute to raw flop/s!**





## Source #1: Expanded Number of Processors

---

- **Recall Observation #1 and “back-of-envelope estimates”: Scalability not a problem.**
- **Caveat: the processor network must also be scalable (applies to protocols as well as to hardware)**
- **Remaining four orders of magnitude could be met by hardware expansion (but this does *not* mean that fixed-size applications of today would run  $10^4$  times faster)**





## Source #2:

### More Efficient Use of Faster Processors

---

- Current low efficiencies of sparse codes can be improved if regularity of reference is exploited with memory-assist features
- Recall Observation #3: PDEs have exploitable periodic workingset structures that can overcome memory latency
- Caveat: high bandwidth is critical, since PDE algorithms do only  $O(N)$  work for  $O(N)$  gridpoints worth of loads and stores
- One to two orders of magnitude can be gained by catching up to the clock, and by following the clock into the few-GHz range





## Source #3:

# More “Architecture Friendly” Algorithms

---

- Algorithmic practice needs to catch up to architectural demands
  - several “one-time” gains remain to be contributed that could improve data locality or reduce synchronization frequency, while maintaining required concurrency and slackness
  - “One-time” refers to improvements by small constant factors, nothing that scales in  $N$  or  $P$  – complexities are already near information-theoretic lower bounds, and we reject increases in flop rates that derive from *less* efficient algorithms
  - Caveat: remaining algorithmic performance improvements may cost extra space or may bank on stability shortcuts that occasionally backfire, making performance modeling less predictable
- Perhaps an order of magnitude of performance remains here





# Raw Performance Improvement from Algorithms

---

- **Spatial reorderings that improve locality**
  - interlacing of all related grid-based data structures
  - ordering gridpoints and grid edges for L1/L2 reuse
- **Discretizations that improve locality**
  - higher-order methods (lead to larger denser blocks at each point than lower-order methods)
  - vertex-centering (for same tetrahedral grid, leads to denser blockrows than cell-centering)
- **Temporal reorderings that improve locality**
  - block vector algorithms (reuse cached matrix blocks; vectors in block are independent)
  - multi-step vector algorithms (reuse cached vector blocks; vectors have sequential dependence)





## **Raw Performance Improvement from Algorithms, cont.**

---

- **Temporal reorderings that reduce synchronization penalty**
  - **less stable algorithmic choices that reduce synchronization frequency (deferred orthogonalization, speculative step selection)**
  - **less global methods that reduce synchronization range by replacing a tightly coupled global process (e.g., Newton) with loosely coupled sets of tightly coupled local processes (e.g., Schwarz)**
- **Precision reductions that make bandwidth seem larger**
  - **lower precision representation of preconditioner matrix coefficients or poorly known coefficients (arithmetic is still performed on full precision extensions)**





## Source #4:

### Algorithms Packing More Science Per Flop

---

- **Some algorithmic improvements do not improve flop rate, but lead to the same scientific end in the same time at lower hardware cost (less memory, lower operation complexity)**
- **Caveat: such adaptive programs are more complicated and less thread-uniform than those they improve upon in quality/cost ratio**
- **Desirable that petaflop/s machines be general purpose enough to run the “best” algorithms**
- **Not daunting, conceptually, but puts an enormous premium on dynamic load balancing**
- **An order of magnitude or more can be gained here for many problems**





# Example of Adaptive Opportunities

---

- **Spatial Discretization-based adaptivity**
  - change discretization type and order to attain required approximation to the continuum everywhere without over-resolving in smooth, easily approximated regions
- **Fidelity-based adaptivity**
  - change continuous formulation to accommodate required phenomena everywhere without enriching in regions where nothing happens
- **Stiffness-based adaptivity**
  - change solution algorithm to provide more powerful, robust techniques in regions of space-time where discrete problem is linearly or nonlinearly stiff without extra work in nonstiff, locally well-conditioned regions





# Status and Prospects for Advanced Adaptivity

---

- Metrics and procedures well developed in only a few areas
  - method-of-lines ODEs for stiff IBVPs and DAEs, FEA for elliptic BVPs
- Multi-model methods used in *ad hoc* ways in production
  - Boeing TRANAIR code
- Poly-algorithmic solvers demonstrated in principle but rarely in the “hostile” environment of high-performance computing
- Requirements for progress
  - management of hierarchical levels of synchronization
  - user specification of hierarchical priorities of different threads





# Summary of Suggestions for High Performance

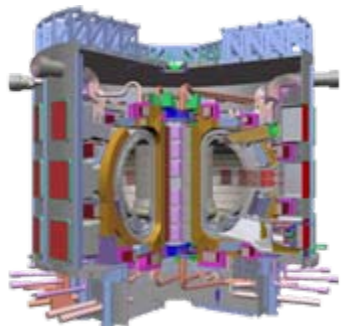
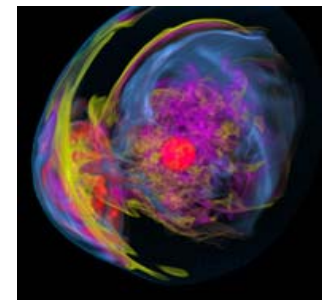
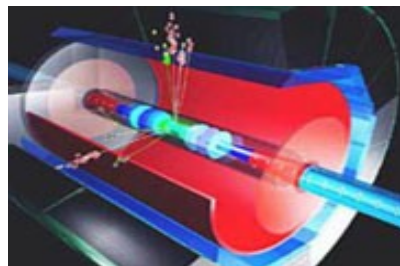
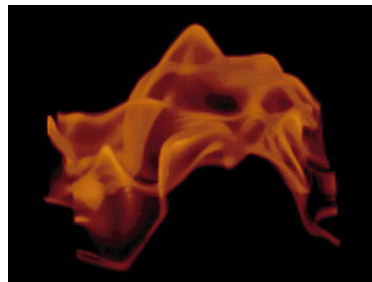
---

- Algorithms that deliver more “science per flop”
  - possibly large problem-dependent factor, through adaptivity (but we won't count this towards rate improvement)
- Algorithmic variants that are more architecture-friendly
  - expect *half* an order of magnitude, through improved locality and relaxed synchronization
- More efficient use of processor cycles, and faster processor/memory
  - expect *one-and-a-half* orders of magnitude, through memory-assist language features, PIM, and multithreading
- Expanded number of processors
  - expect *two* orders of magnitude, through dynamic balancing and extreme care in implementation

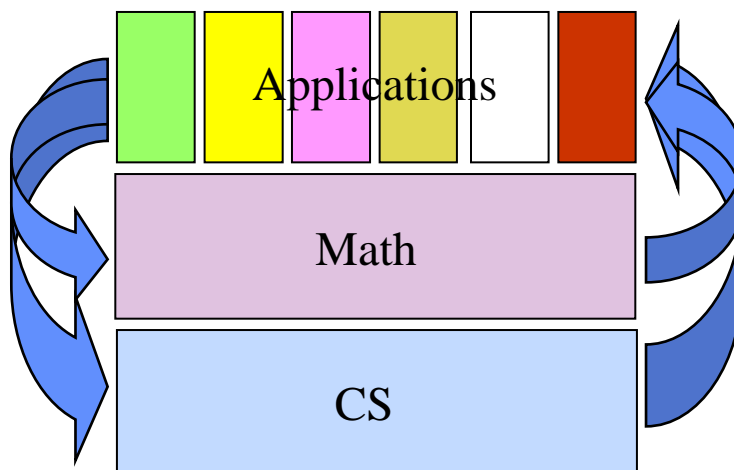




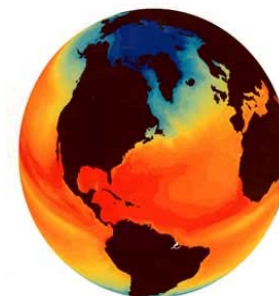
# It's *not* about the solver



Applications  
drive



Enabling  
technologies  
respond

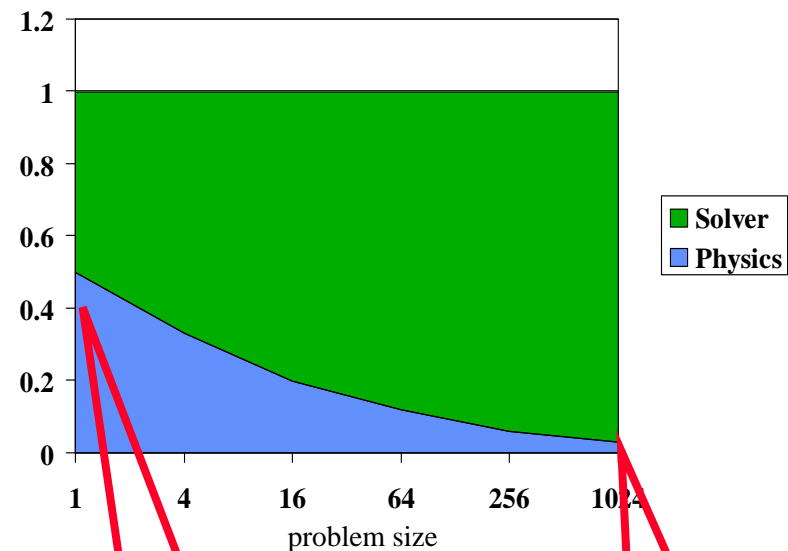






# It's *all* about the solver (at the terascale)

- Given, for example:
  - a “physics” phase that scales as  $O(N)$
  - a “solver” phase that scales as  $O(N^{3/2})$
  - computation is almost all solver after several doublings
- Most applications groups have not yet “felt” this curve in their gut
  - BG/L will change this
  - 64K-processor machine delivered in 2005



**Solver takes  
50% time  
on 64 procs**

**Solver takes  
97% time on  
64K procs**

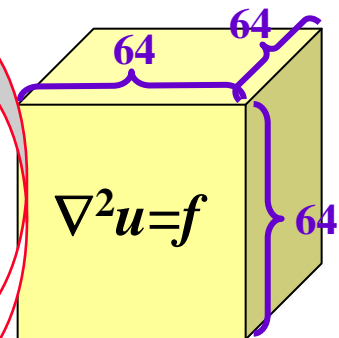




# The power of optimal algorithms

- Advances in algorithmic efficiency can rival advances in hardware architecture
- Consider Poisson's equation on a cube of size  $N=n^3$

Year	Method	Reference	Storage	Flops
1947	GE (banded)	Von Neumann & Goldstine	$n^5$	$n^7$
1950	Optimal SOR	Young	$n^3$	$n^4 \log n$
1971	CG	Reid	$n^3$	$n^{3.5} \log n$
1984	Full MG	Brandt	$n^3$	$n^3$



- If  $n=64$ , this implies an overall reduction in flops of ~16 million

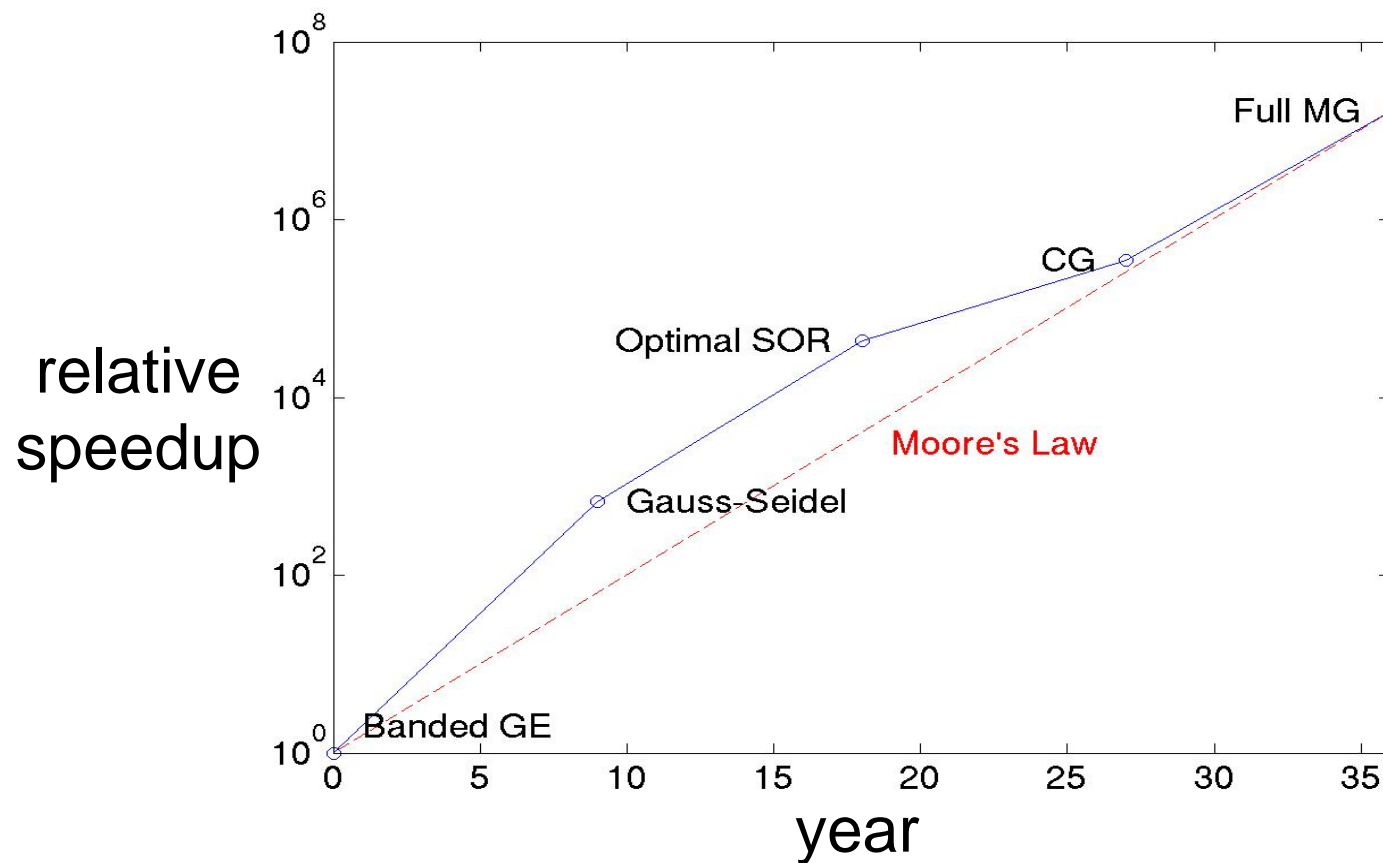
\*Six-months is reduced to 1 s





# Algorithms and Moore's Law

- This advance took place over a span of about 36 years, or 24 doubling times for Moore's Law
- $2^{24} \approx 16$  million  $\Rightarrow$  the same as the factor from algorithms alone!





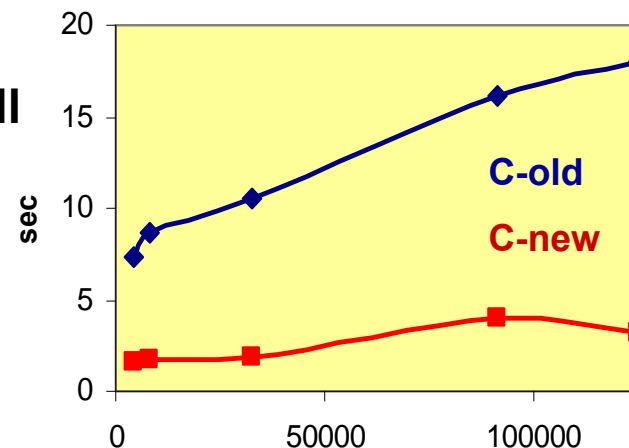


# Algebraic multigrid on BG/L

- Algebraic multigrid a key algorithmic technology
  - Discrete operator defined for finest grid by the application, itself, *and* for many recursively derived levels with successively fewer degrees of freedom, for solver purposes
  - Unlike geometric multigrid, AMG not restricted to problems with “natural” coarsenings derived from grid alone
- Optimality (cost per cycle) intimately tied to the ability to coarsen aggressively
- Convergence scalability (number of cycles) and parallel efficiency also sensitive to rate of coarsening

While much research and development remains, multigrid will clearly be practical at BG/L-scale concurrency

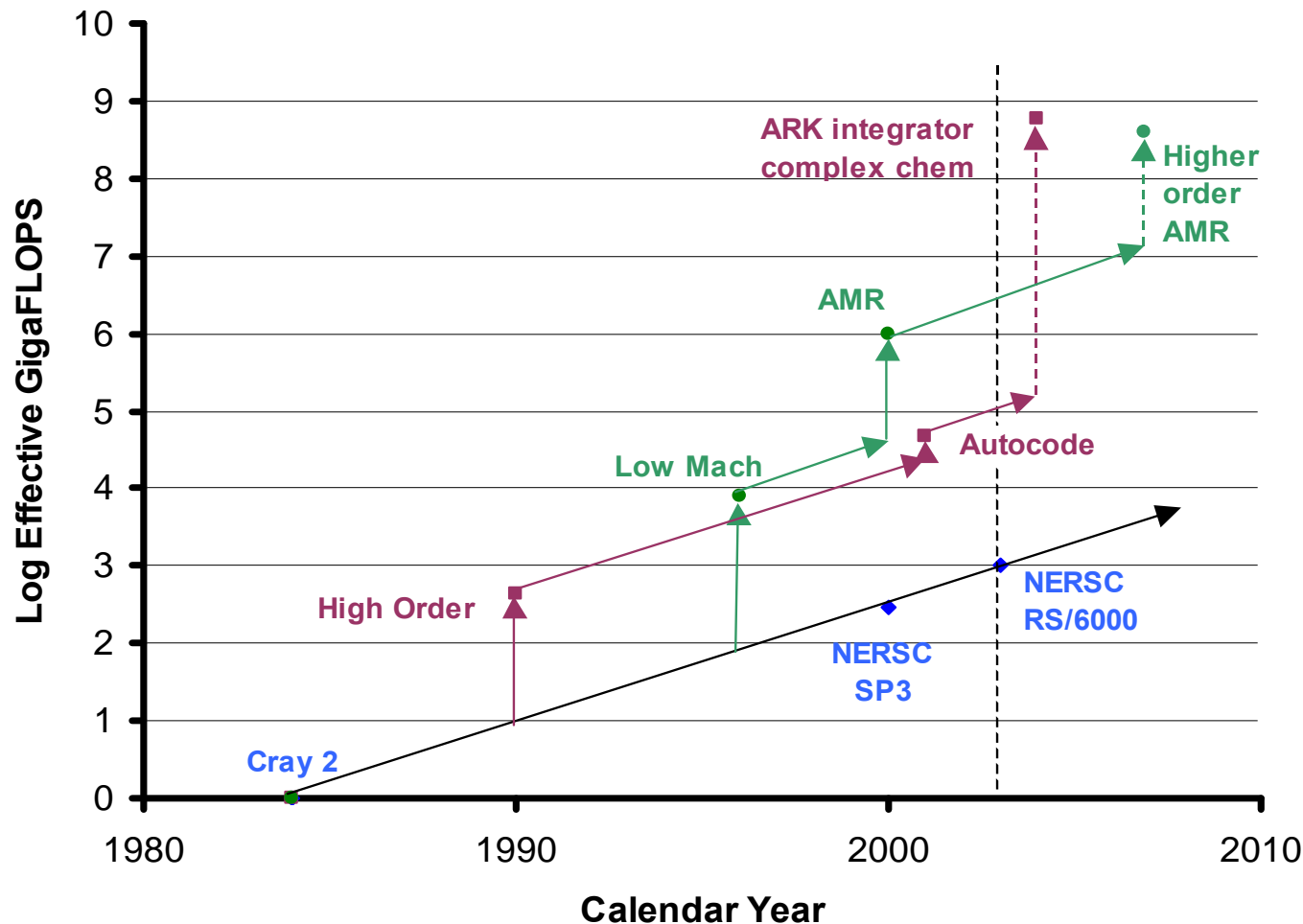
Figure shows weak scaling result for AMG out to 131,072 processors, with one  $25 \times 25 \times 25$  block per processor (from 15.6K dofs up to 2.05B dofs)







# “Moore’s Law” for combustion simulations

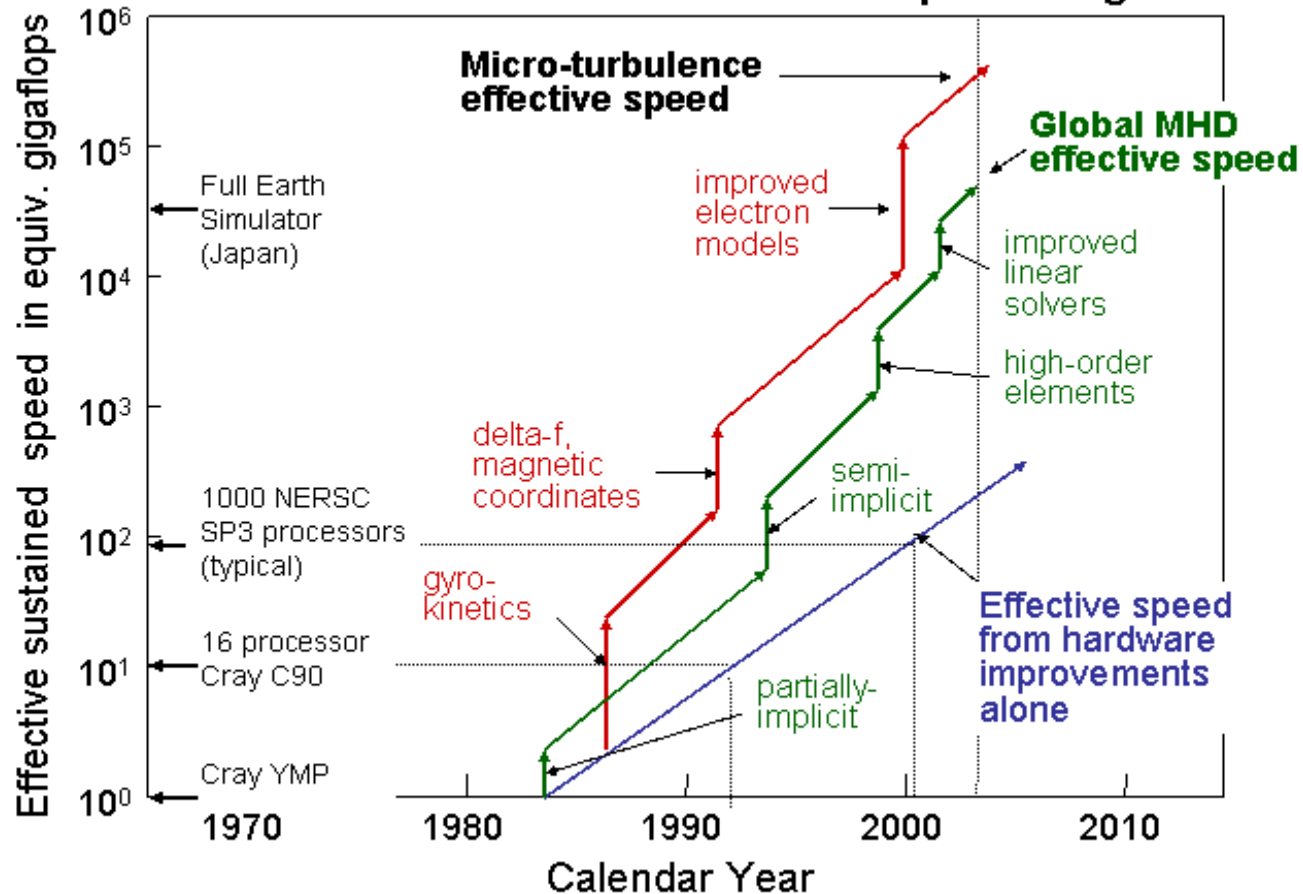






# “Moore’s Law” for MHD simulations

Magnetic Fusion Energy: “Effective speed” increases came from both faster hardware and improved algorithms



“Semi-implicit”:

All waves treated implicitly, but still stability-limited by transport

“Partially implicit”:

Fastest waves filtered, but still stability-limited by slower waves

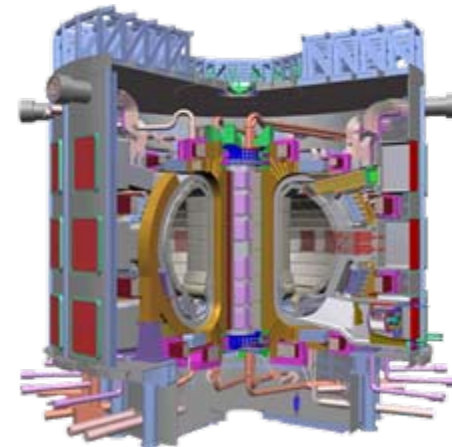




# Scaling fusion simulations up to ITER

name	symbol	units	CDX-U	DIII-D	ITER
Field	$B_0$	Tesla	0.22	1	5.3
Minor radius	$a$	meters	.22	.67	2
Temp.	$T_e$	keV	0.1	2.0	8.
Lundquist no.	$S$		$1 \times 10^4$	$7 \times 10^6$	$5 \times 10^8$
Mode growth time	$\tau_A S^{1/2}$	s	$2 \times 10^{-4}$	$9 \times 10^{-3}$	$7 \times 10^{-2}$
Layer thickness	$a S^{-1/2}$	m	$2 \times 10^{-3}$	$2 \times 10^{-4}$	$8 \times 10^{-5}$
zones	$N_R \times N_\theta \times N_\phi$		$3 \times 10^6$	$5 \times 10^{10}$	$3 \times 10^{13}$
CFL timestep	$\Delta X / V_A$ (Explicit)	s	$2 \times 10^{-9}$	$8 \times 10^{-11}$	$7 \times 10^{-12}$
Space-time pts			$6 \times 10^{12}$	$1 \times 10^{20}$	$6 \times 10^{24}$

10<sup>12</sup> needed



**International  
Thermonuclear  
Experimental  
Reactor**

**2017 – first  
experiments, in  
Cadaraches,  
France**





Hardware: 3

Software: 9

## Where to find 12 orders of magnitude in 10 years?

- 1.5 orders: increase processor speed and efficiency
- 1.5 orders: increase memory efficiency
- 1 order: increase resolution
  - Same number of elements, many fewer elements
- 1 order: increase number of field lines
- 4 orders: calculation down to petascale ( $10^{15}$ )!
- Zeroth order: ITER volume and resolution, 4th order: less severe
- 3 orders: implicit solve
  - Mode growth time 9 orders longer than Alfvén-limited CFL





# Summary

---

- PDEs continue to drive the highest-end computing, as they have since ca. 1945
- There appears to be no fundamental limit to solving PDEs on arbitrarily fine spatial meshes in fixed execution time with arbitrarily high numbers of processors provided...
  - one does not have to resolve timescales correspondingly finely in a CFL sense
  - one can do a very fine load balancing and amortize it over many steps
  - one has a near optimal linear implicit solver, like Krylov-MG
  - for nonlinear problems, one can use Newton in a resolution-independent asymptotic regime
- One should expect to have to *work!* to achieve such ends, and should start with good solver components as building blocks





# Reminder about the Source of Simulations

---

- Computational science and engineering is not about individual large-scale analyses, done fast and “thrown over the wall”
- Both “results” and their sensitivities are desired; often multiple operation points to be simulated are known *a priori*, rather than sequentially
- Sensitivities may be fed back into optimization process
- Full CFD analyses may also be inner iterations in a multidisciplinary computation
- In such contexts, “petaflop/s” may mean 1,000 analyses running somewhat asynchronously with respect to each other, each at 1 Tflop/s – clearly a less daunting challenge and one that has better synchronization properties for exploiting “The Grid” – than 1 analysis running at 1 Pflop/s





---

# **Tutorial M06**

**Erik P. DeBenedictis**





# Outline

---

- **Overview**
  - **Insight From a Dinner Conversation in DC**
  - **Super-Roadmap**
- **Limitations to Moore's Law**
  - **Transistor Scaling Limits per ITRS**
  - **Consequence to System Performance per Burger and Keckler Study**
- **What It Means and What To Do About It**
  - **Legacy C++/Fortran**
  - **Systolic Array Lessons**
  - **New Very Parallel Code**
  - **Special Purpose Assist**
  - **Analog/Neural Net**
- **Over the Horizon**
  - **Reversible Logic**
  - **Quantum Computing**





# Insight From A Dinner Conversation

---

- I have dinner with a physicist at a joint ITRS and electron device meeting in DC 12/2005
- The fellow tells me in hushed tones that he knows the future to Moore's Law
  - Is this trivial or profound?
- I ask what it is?
- Answer: More Parallelism.
  - I knew this: trivial
- I say there may not be enough parallelism in problems – and has he talked to programmers
- Answer: “no”
  - Oh boy, the future of Moore's Law depends on YOU programming smarter and you don't know this: profound





# Outline

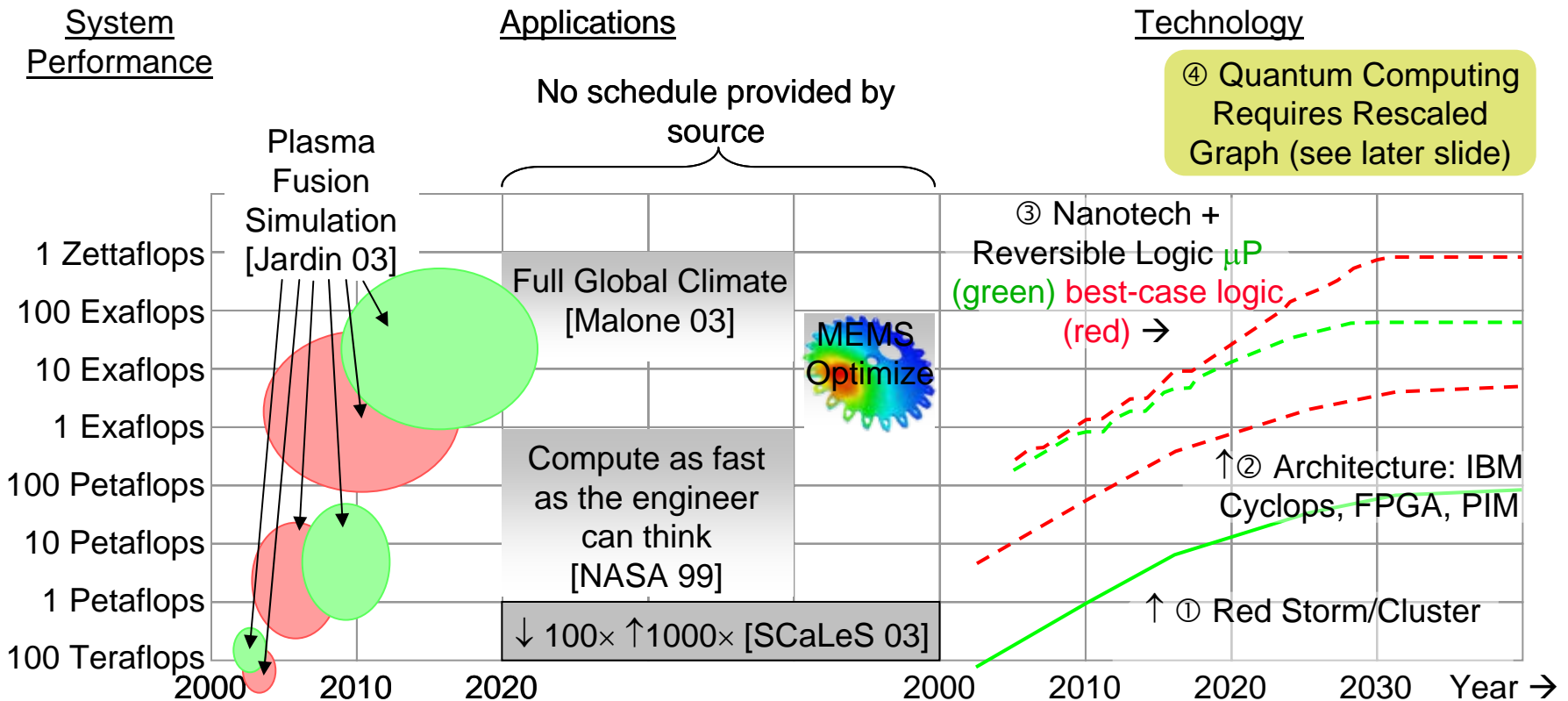
---

- **Overview**
  - Insight From a Dinner Conversation in DC
  - Super-Roadmap
- **Limitations to Moore's Law**
  - Transistor Scaling Limits per ITRS
  - Consequence to System Performance per Burger and Keckler Study
- **What It Means and What To Do About It**
  - Legacy C++/Fortran
  - Systolic Array Lessons
  - New Very Parallel Code
  - Special Purpose Assist
  - Analog/Neural Net
- **Over the Horizon**
  - Reversible Logic
  - Quantum Computing





# Applications and \$100M Supercomputers



[Jardin 03] S.C. Jardin, "Plasma Science Contribution to the SCaLeS Report," Princeton Plasma Physics Laboratory, PPPL-3879 UC-70, available on Internet.

[Malone 03] Robert C. Malone, John B. Drake, Philip W. Jones, Douglas A. Rotman, "High-End Computing in Climate Modeling," contribution to SCaLeS report.

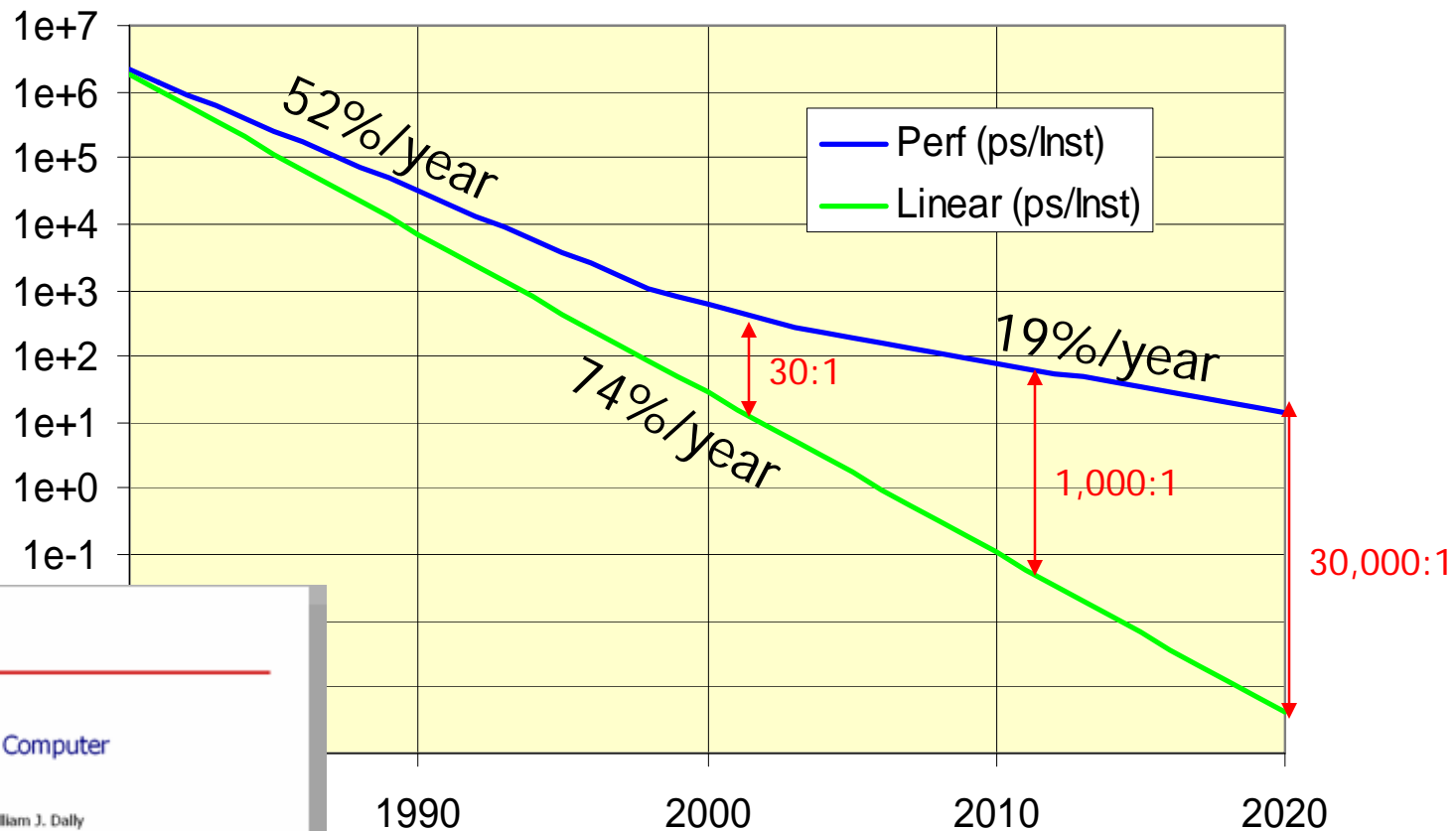
[NASA 99] R. T. Biedron, P. Mehrotra, M. L. Nelson, F. S. Preston, J. J. Rehder, J. L. Rogers, D. H. Rudy, J. Sobieski, and O. O. Storaasli, "Compute as Fast as the Engineers Can Think!" NASA/TM-1999-209715, available on Internet.

[SCaLeS 03] Workshop on the Science Case for Large-scale Simulation, June 24-25, proceedings on Internet a <http://www.pnl.gov/scales/>.

[DeBenedictis 04], Erik P. DeBenedictis, "Matching Supercomputing to Progress in Science," July 2004. Presentation at Lawrence Berkeley National Laboratory, also published as Sandia National Laboratories SAND report SAND2004-3333P. Sandia technical reports are available by going to <http://www.sandia.gov> and accessing the technical library.



# Future potential of novel architecture is large (1000 vs 30)



The Last Classical Computer  
ISAT Study

William J. Dally  
Professor of EE and CS, Stanford University

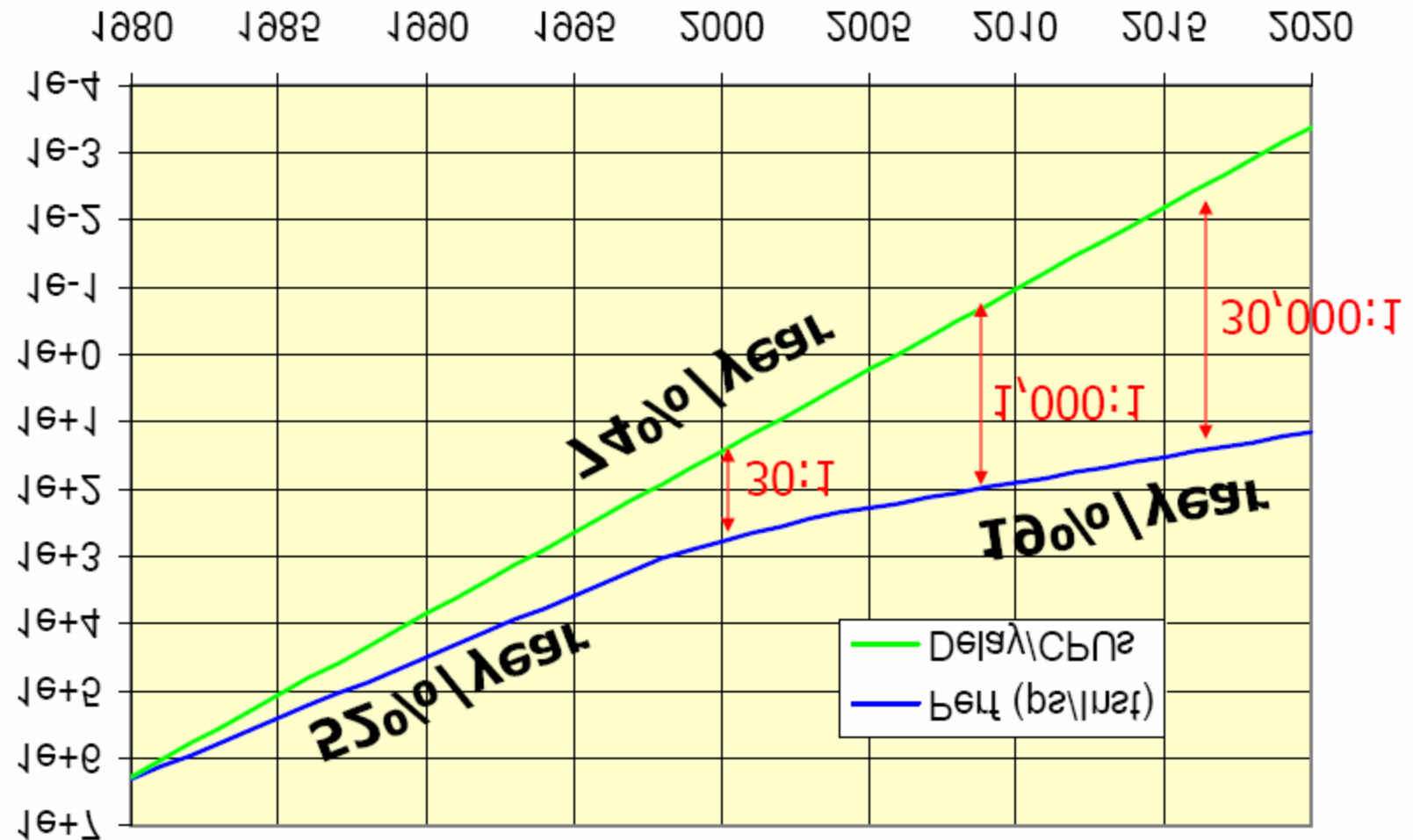
August 24, 2001

From here, reproduced with permission

August 24, 2001



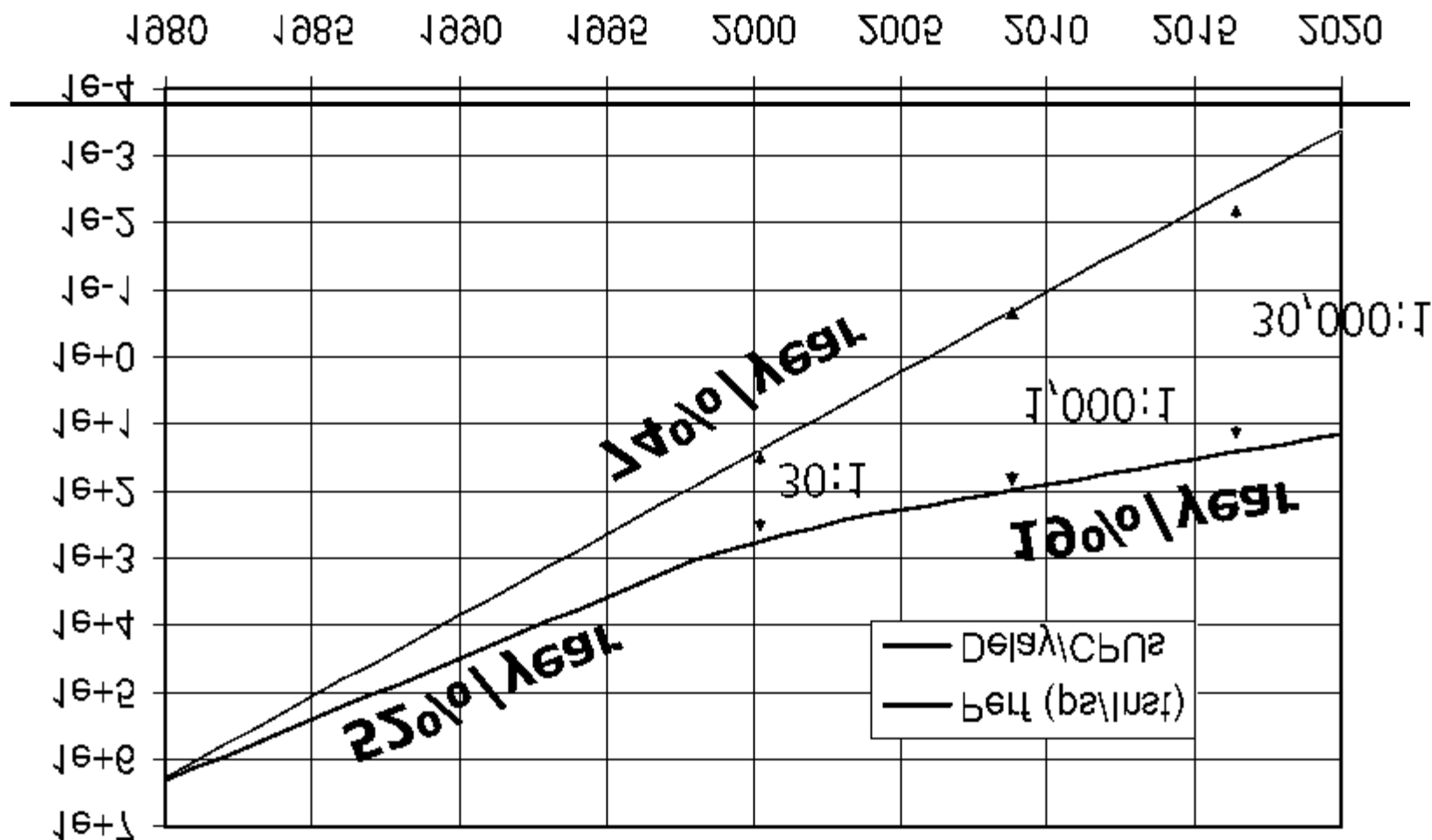
# Graph courtesy of Bill Dally



The capability gap







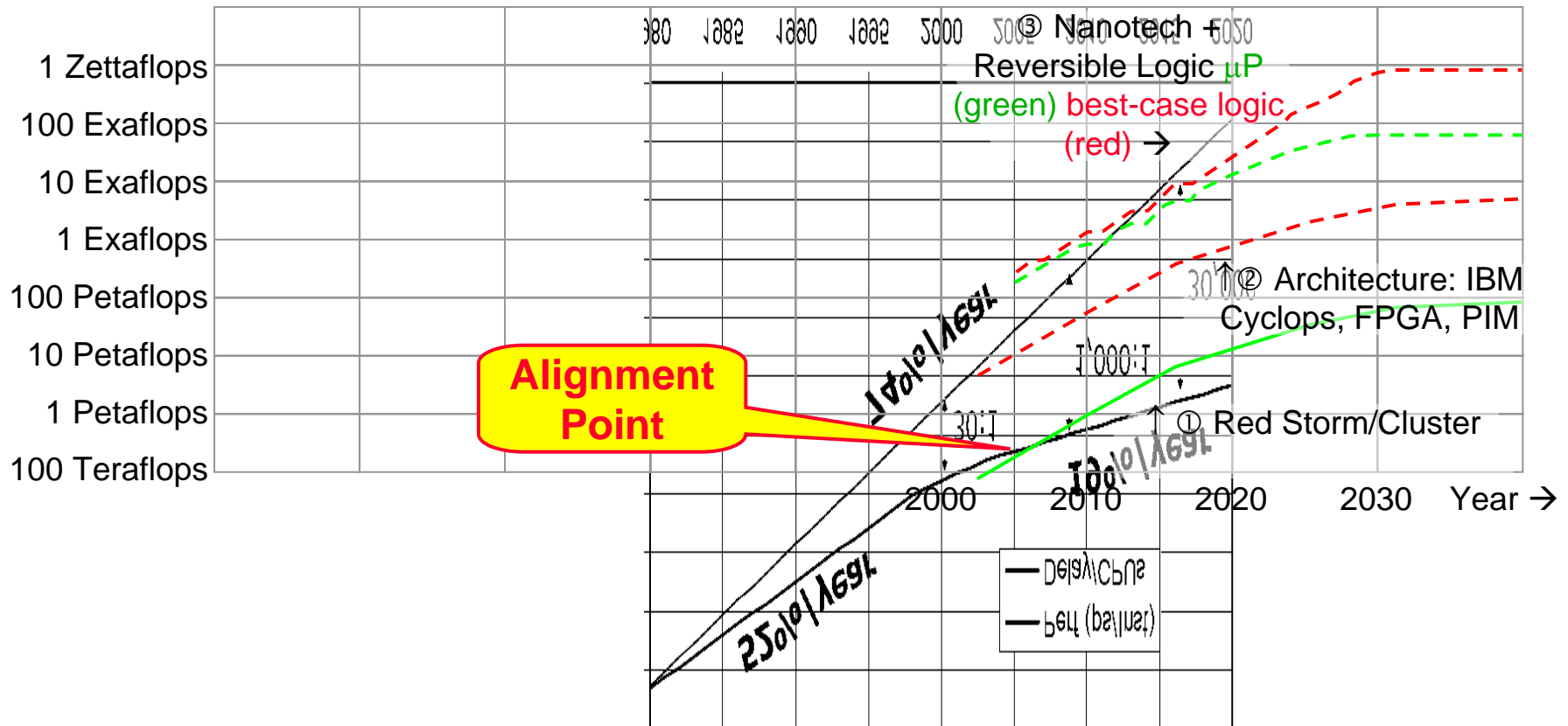




# Trends Align Pretty Well, But Mismatches are Instructive

System  
Performance

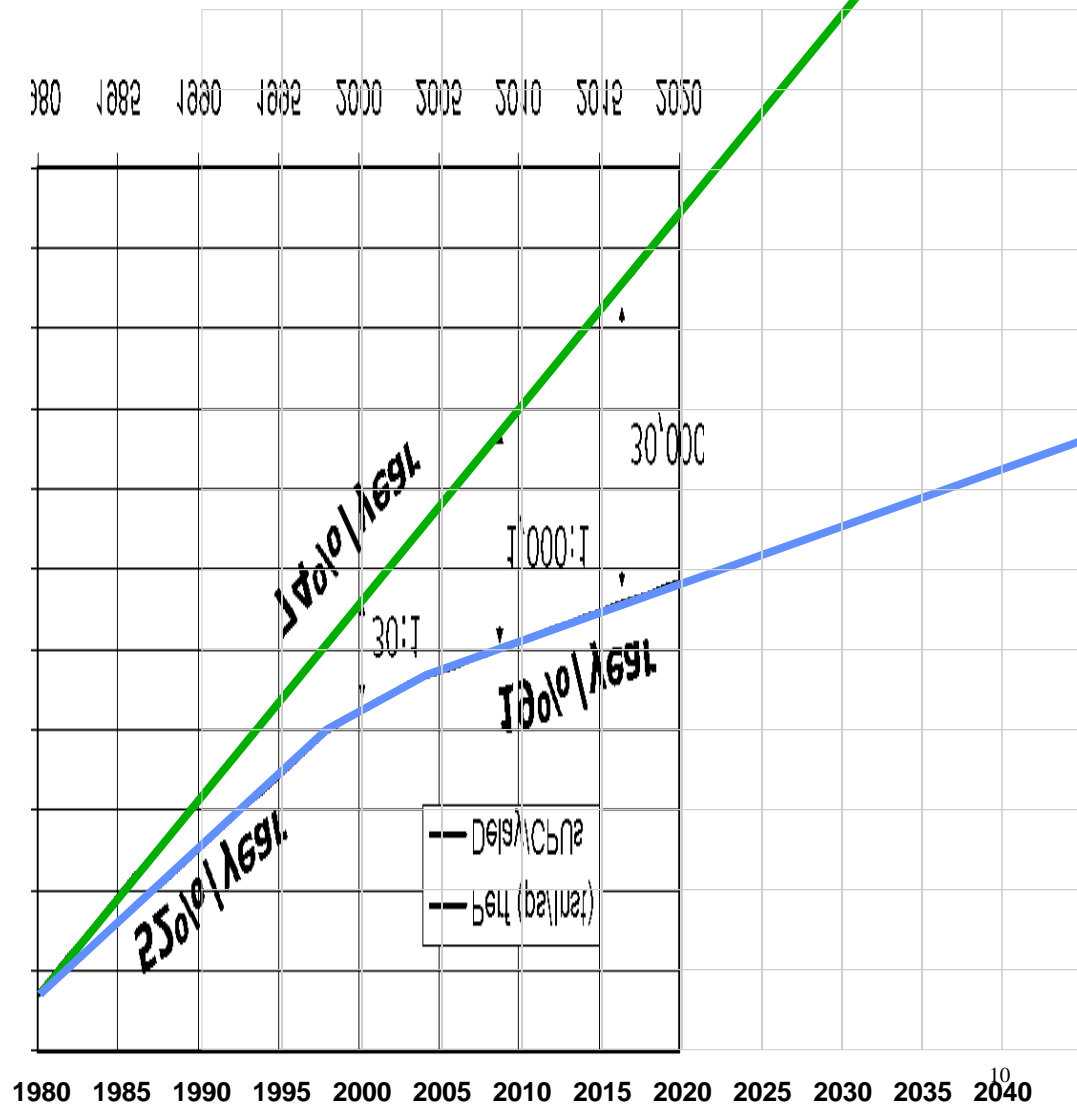
Technology







# Let's Build On The "Capability Gap"

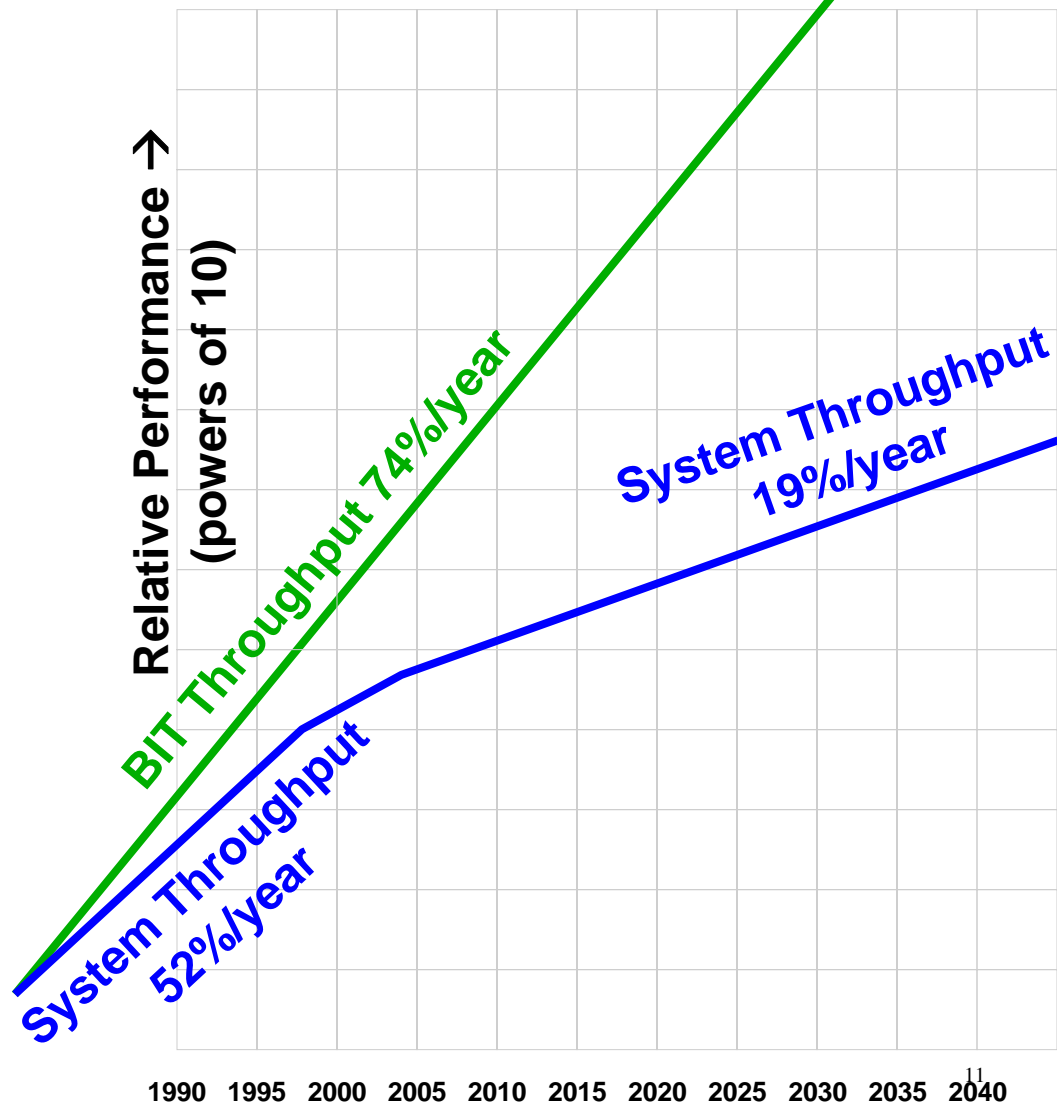






# What Path Will You Follow?

- Capability Gap reference

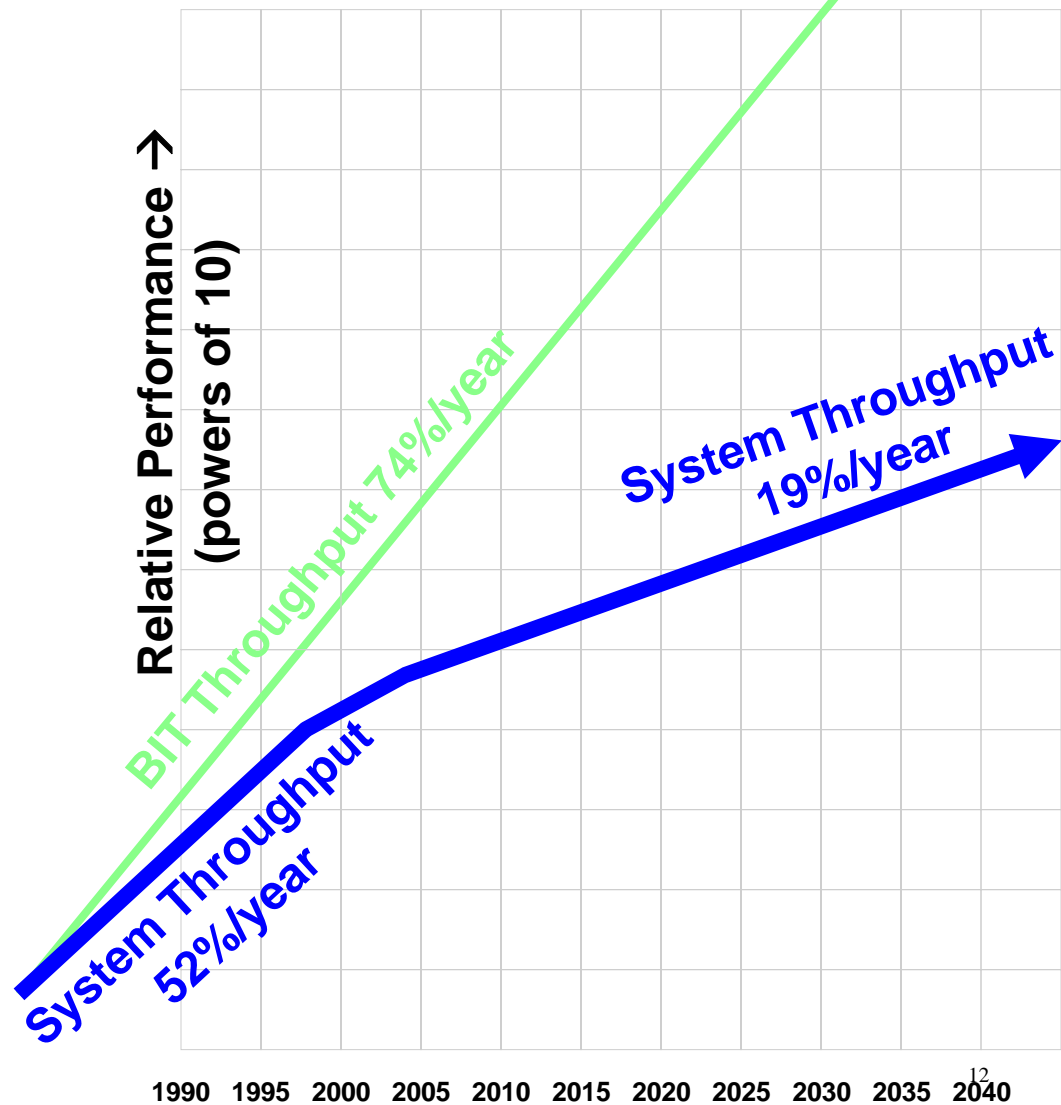






# Single Core Chips

- Although this is not the industry trend!

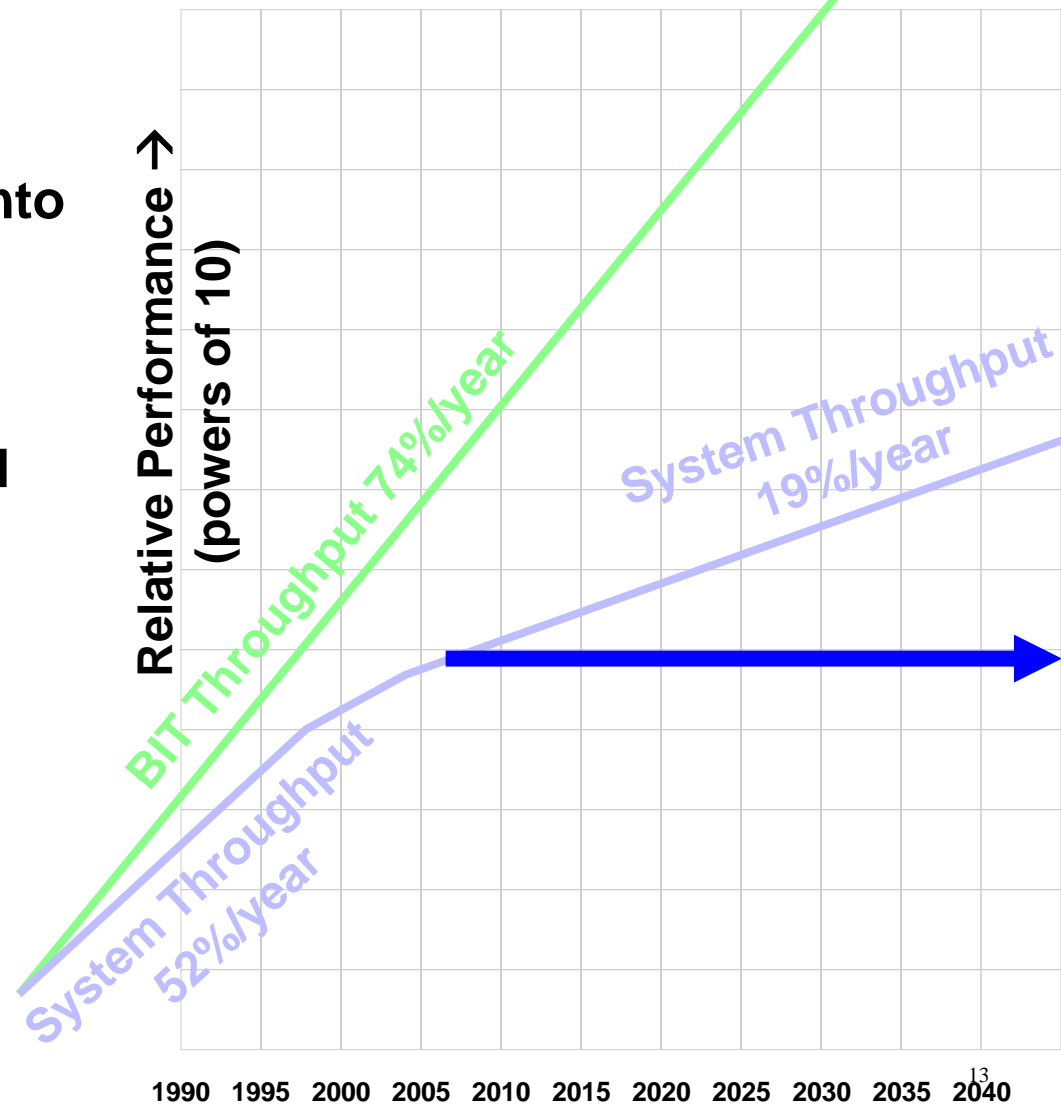






# One Core of a Multicore CPU

- Industry trend is to put benefit of Moore's Law into more cores in multicore  $\mu$ Ps.
- For code that uses one core, performance would be nearly flat

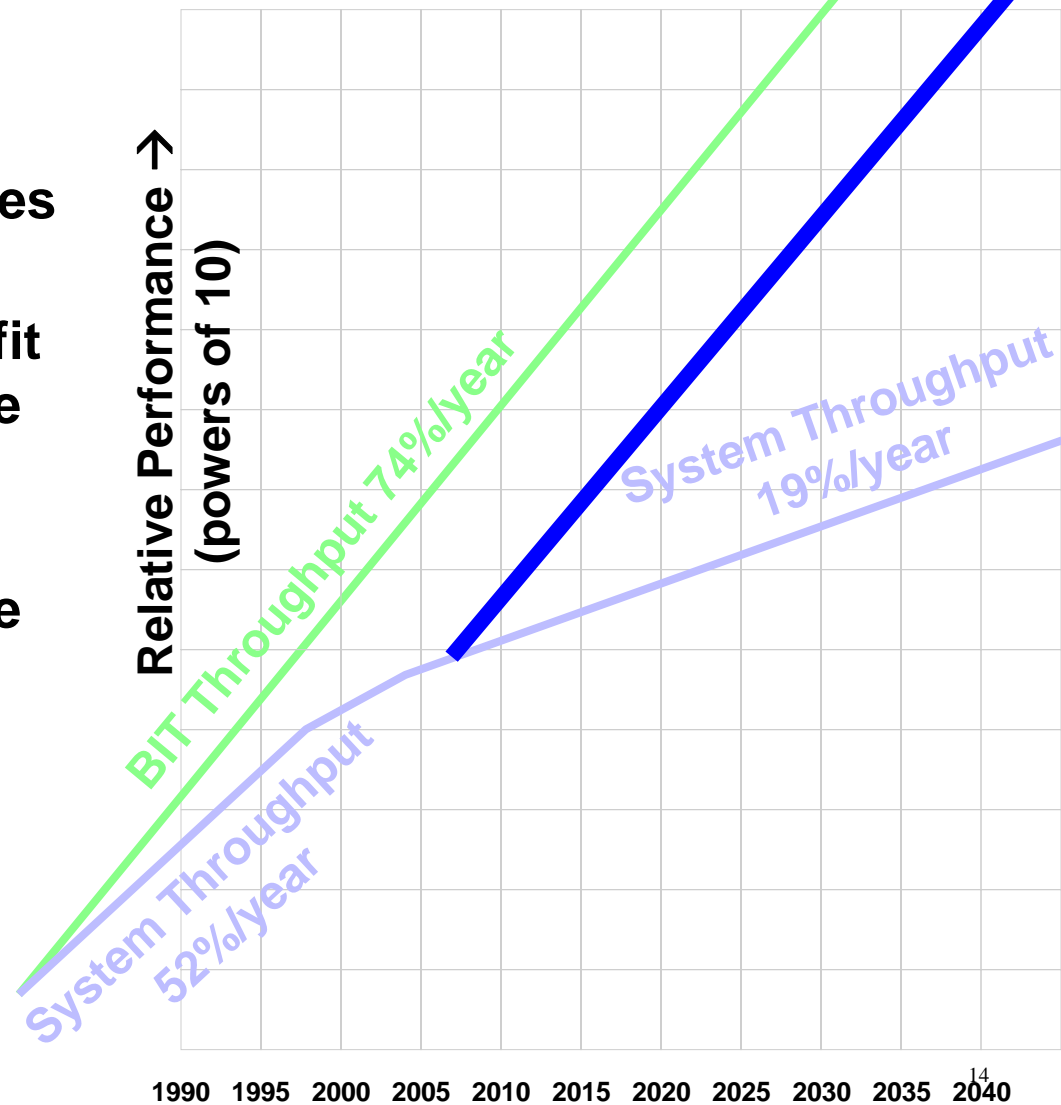






# Multicore CPU Programmed Efficiently

- If you could code to efficiently use all the cores on a multicore CPU AND
- Industry put all the benefit of Moore's Law into more cores THEN
- You would realize performance gains in line with BIT throughput



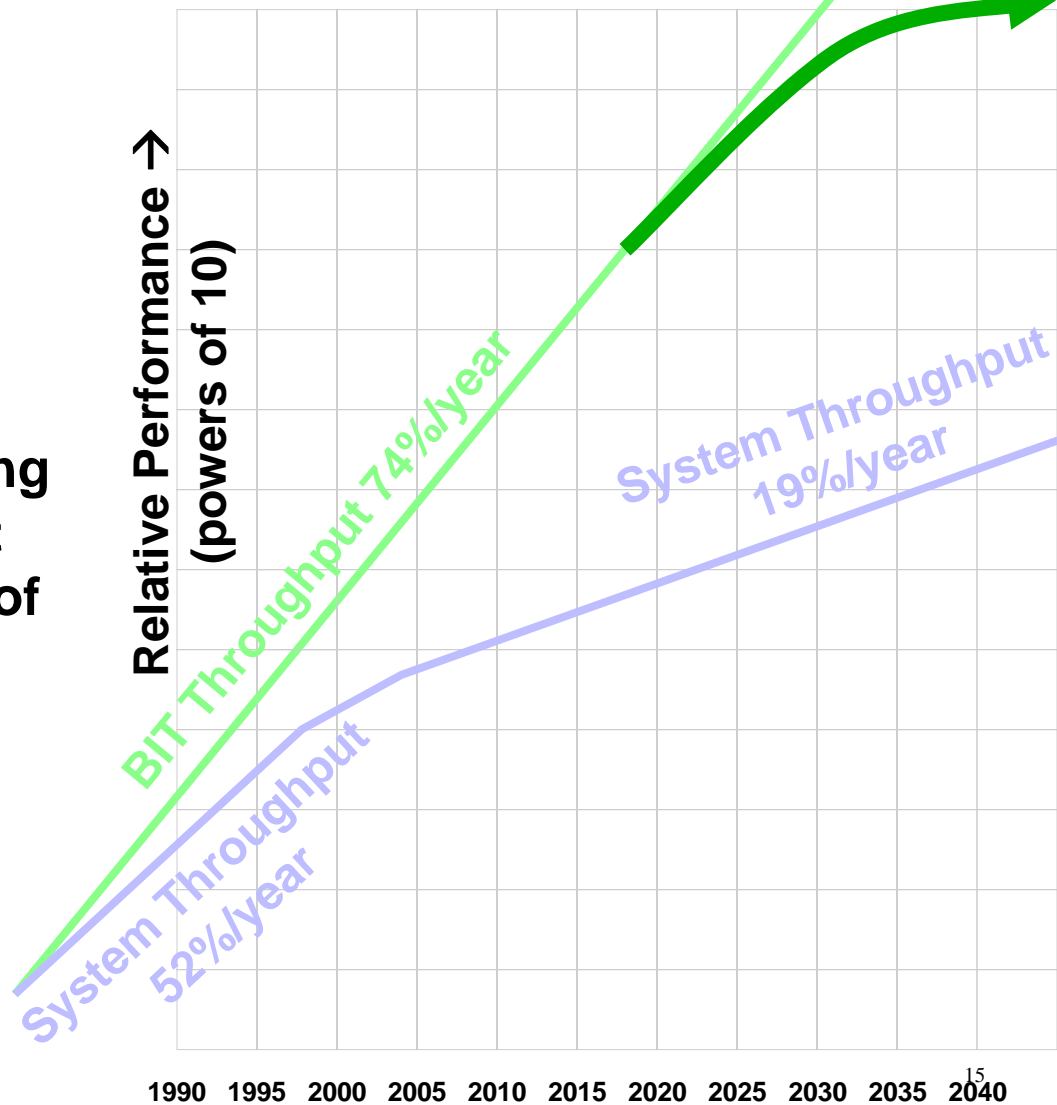




# Thermal Limit

$100k_B T$  Limit

- However, there is a limit for AND-OR-NOT logic beyond which heat production becomes a bottleneck
- Heat production is starting to be a problem now, but there are several orders of magnitude to go before reaching the real limit



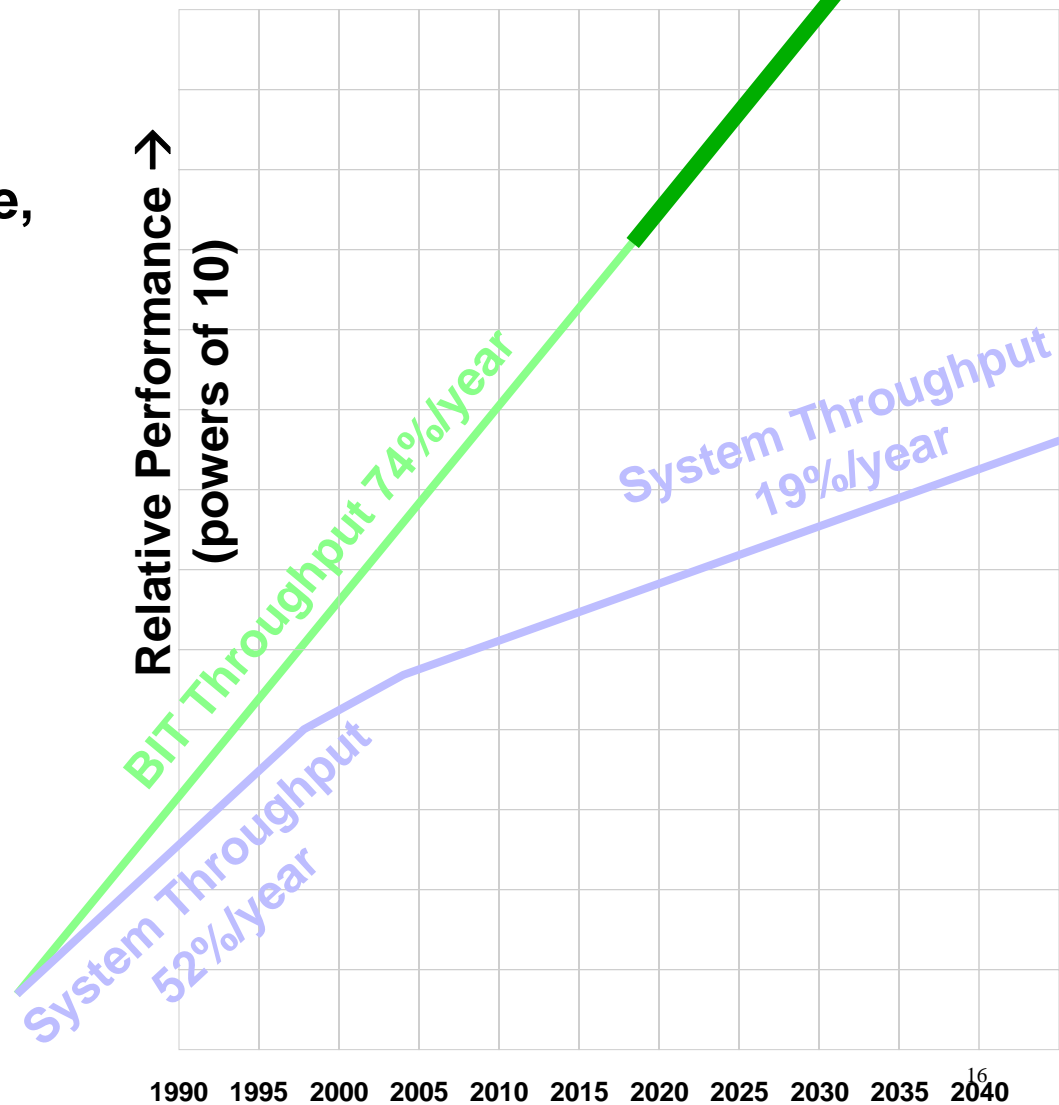




# Reversible Logic

100k<sub>B</sub>T Limit

- The thermal limit can be circumvented in principle, but you have to give up AND-OR-NOT logic





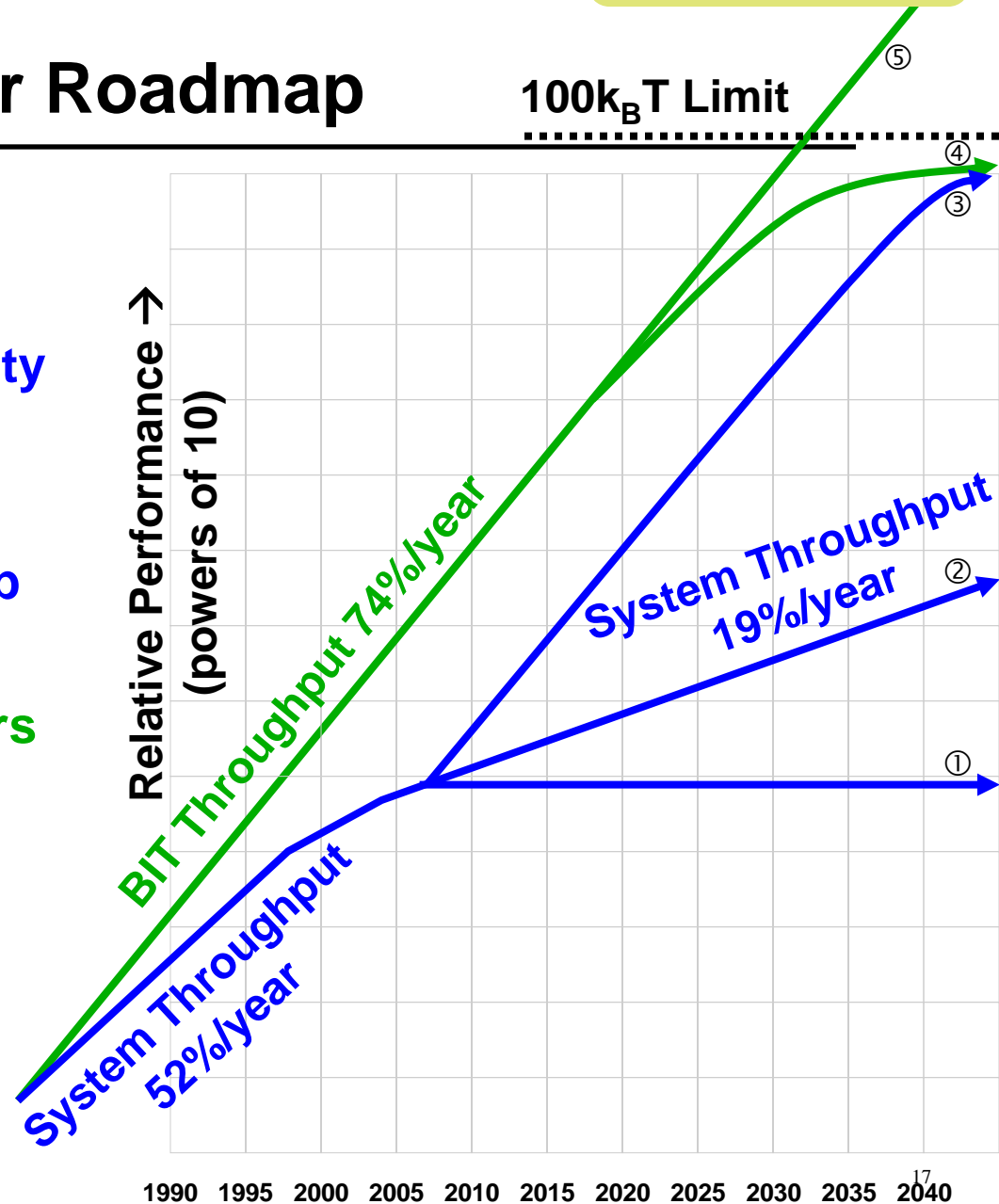


⑥ Quantum Computing  
Requires  
Rescaled Slide

# Super Roadmap

100k<sub>B</sub>T Limit

- ① Nearly flat
  - Single core, commodity
- ② Single core chip
  - C++, Fortran, etc.
- ③ Full benefit of speedup
  - More parallel code
- ④ Fully exploit transistors
  - Custom hardware
- ⑤ Full benefit of physics
  - Ditch AND-OR-NOT
- ⑥ Go beyond bits







- 
- **Remind audience that the last slide shows the impact of Moore's Law (horizontal axis) and architecture (multiple curves) on applications performance (vertical axis)**



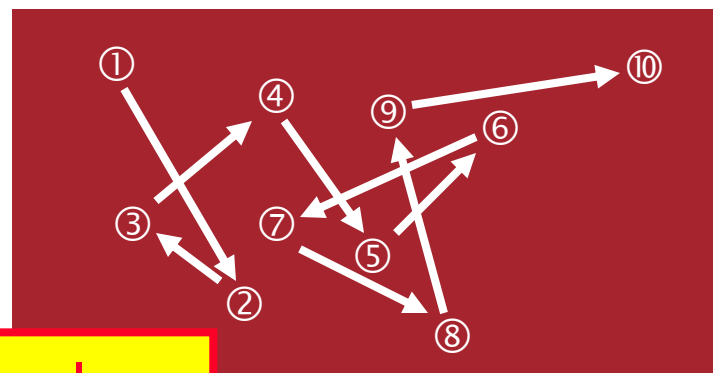


## \*\*\* THIS IS A PREVIEW \*\*\*

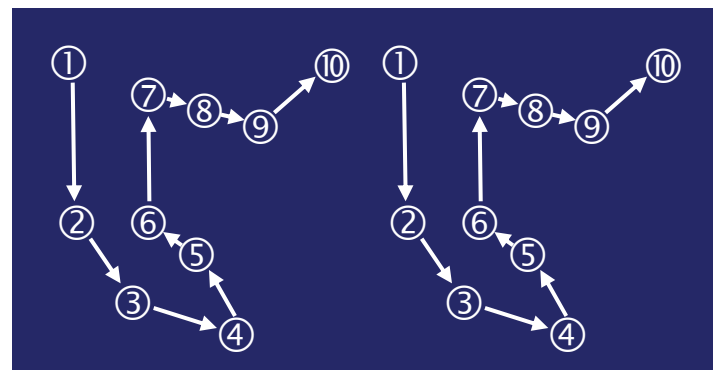
- High node visit rate
- Small size
- Fast propagation velocity
- Parallel
- Organize program graph for short distances
- Programming language must aid programmer in creating short, parallel graphs
- Programmer must use language effectively

### Bus Route Analogy

Bad



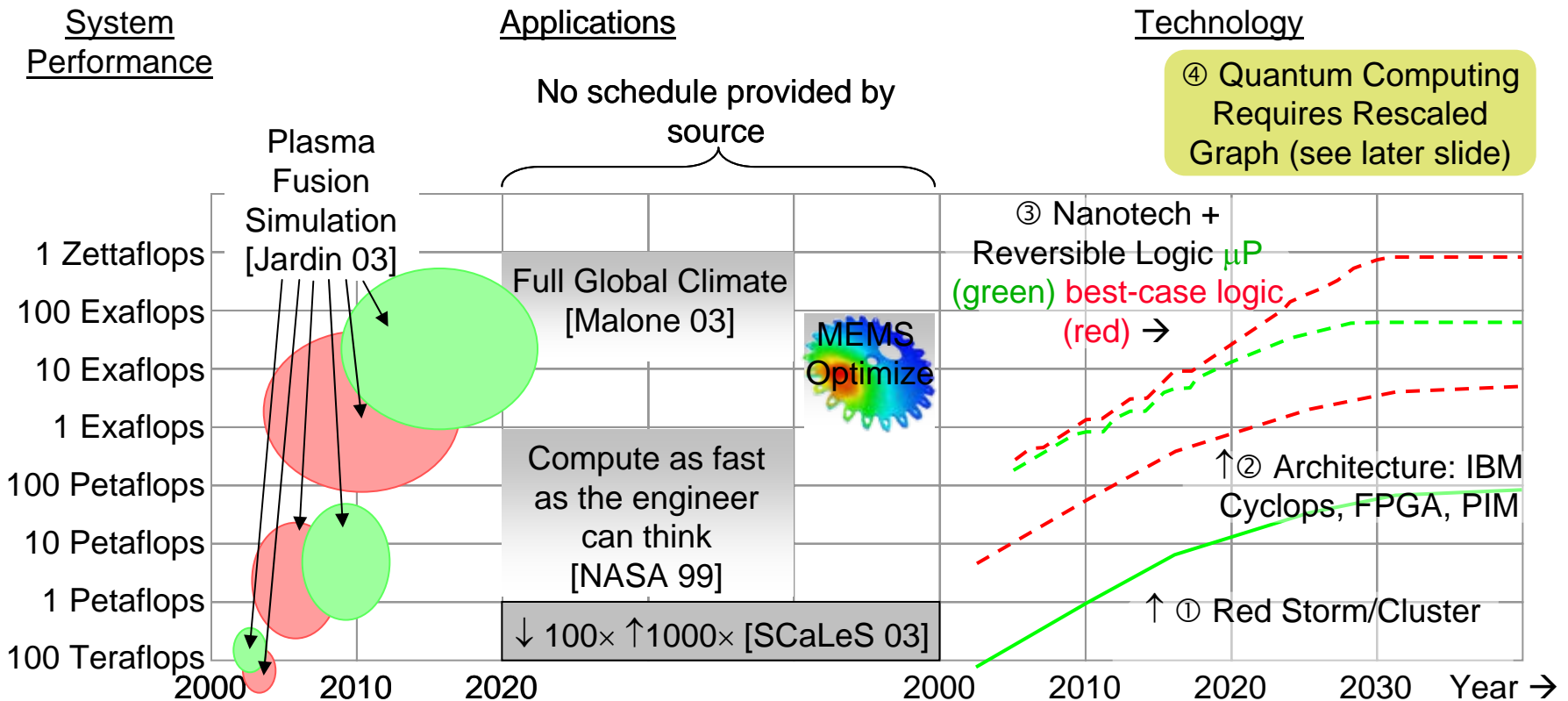
Better







# Quantum Computing (Starting Point)



[Jardin 03] S.C. Jardin, "Plasma Science Contribution to the SCaLeS Report," Princeton Plasma Physics Laboratory, PPPL-3879 UC-70, available on Internet.

[Malone 03] Robert C. Malone, John B. Drake, Philip W. Jones, Douglas A. Rotman, "High-End Computing in Climate Modeling," contribution to SCaLeS report.

[NASA 99] R. T. Biedron, P. Mehrotra, M. L. Nelson, F. S. Preston, J. J. Rehder, J. L. Rogers, D. H. Rudy, J. Sobieski, and O. O. Storaasli, "Compute as Fast as the Engineers Can Think!" NASA/TM-1999-209715, available on Internet.

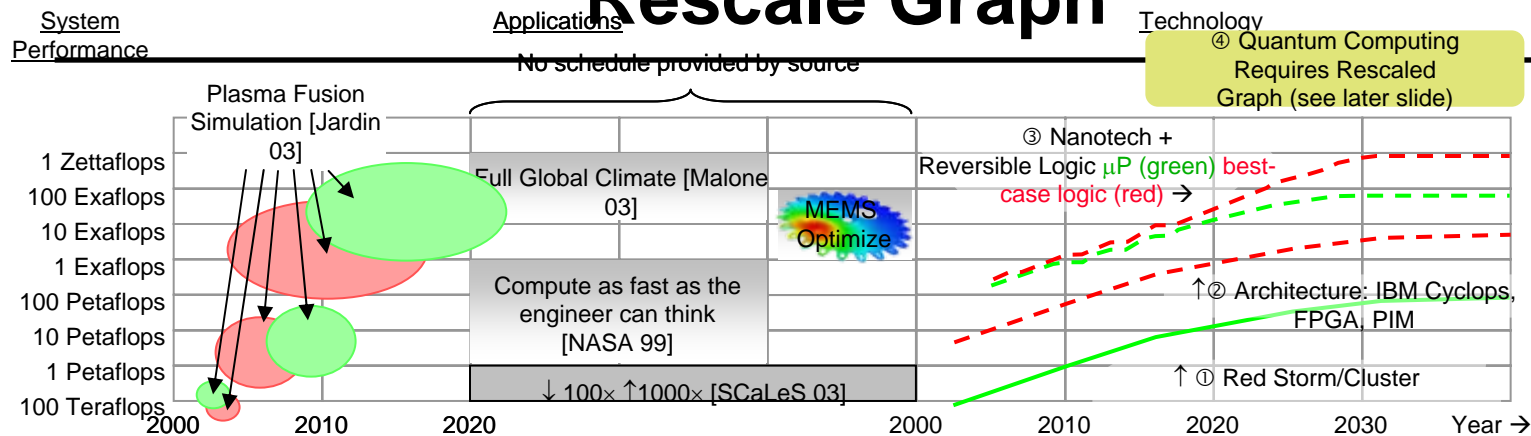
[SCaLeS 03] Workshop on the Science Case for Large-scale Simulation, June 24-25, proceedings on Internet a <http://www.pnl.gov/scales/>.

[DeBenedictis 04], Erik P. DeBenedictis, "Matching Supercomputing to Progress in Science," July 2004. Presentation at Lawrence Berkeley National Laboratory, also published as Sandia National Laboratories SAND report SAND2004-3333P. Sandia technical reports are available by going to <http://www.sandia.gov> and accessing the technical library.





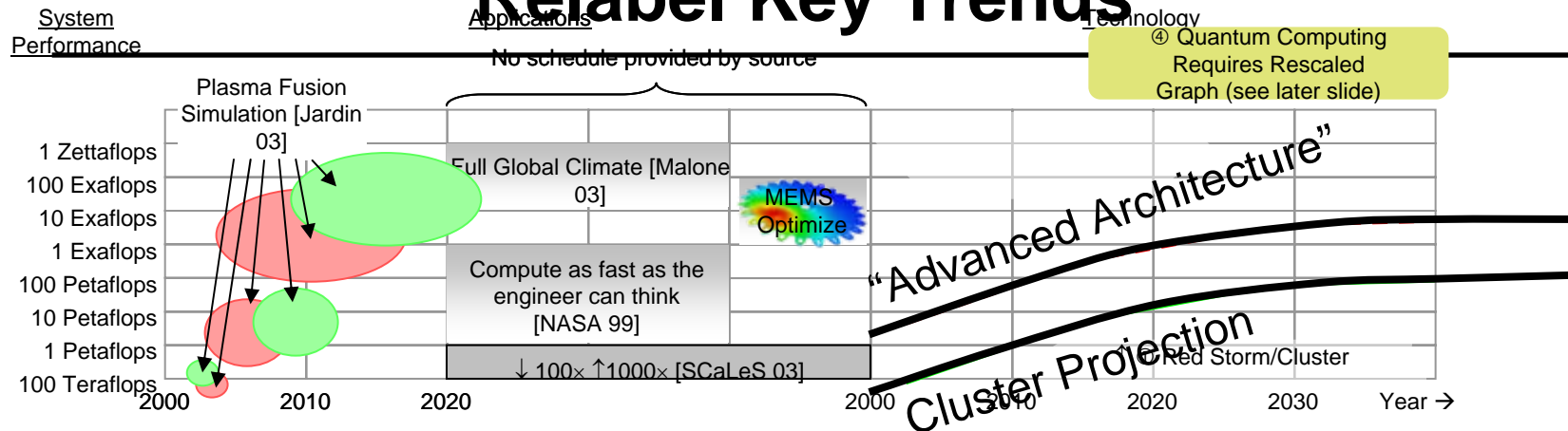
# Rescale Graph







# Relabel Key Trends

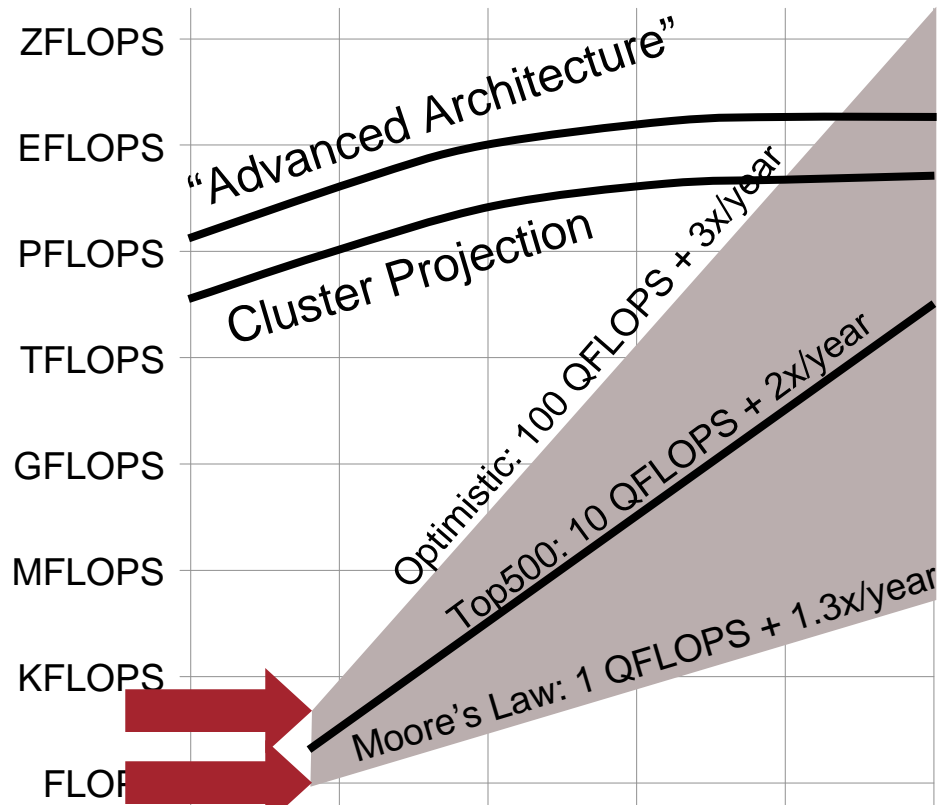






# Emergence of Quantum Computing

- There appears to be an engineering case for quantum computers of 1-100 Q-FLOPS
- One would expect an exponential growth rate for quantum computers similar to Moore's Law, but the rate constant is impossible to predict, so three possibilities have been graphed



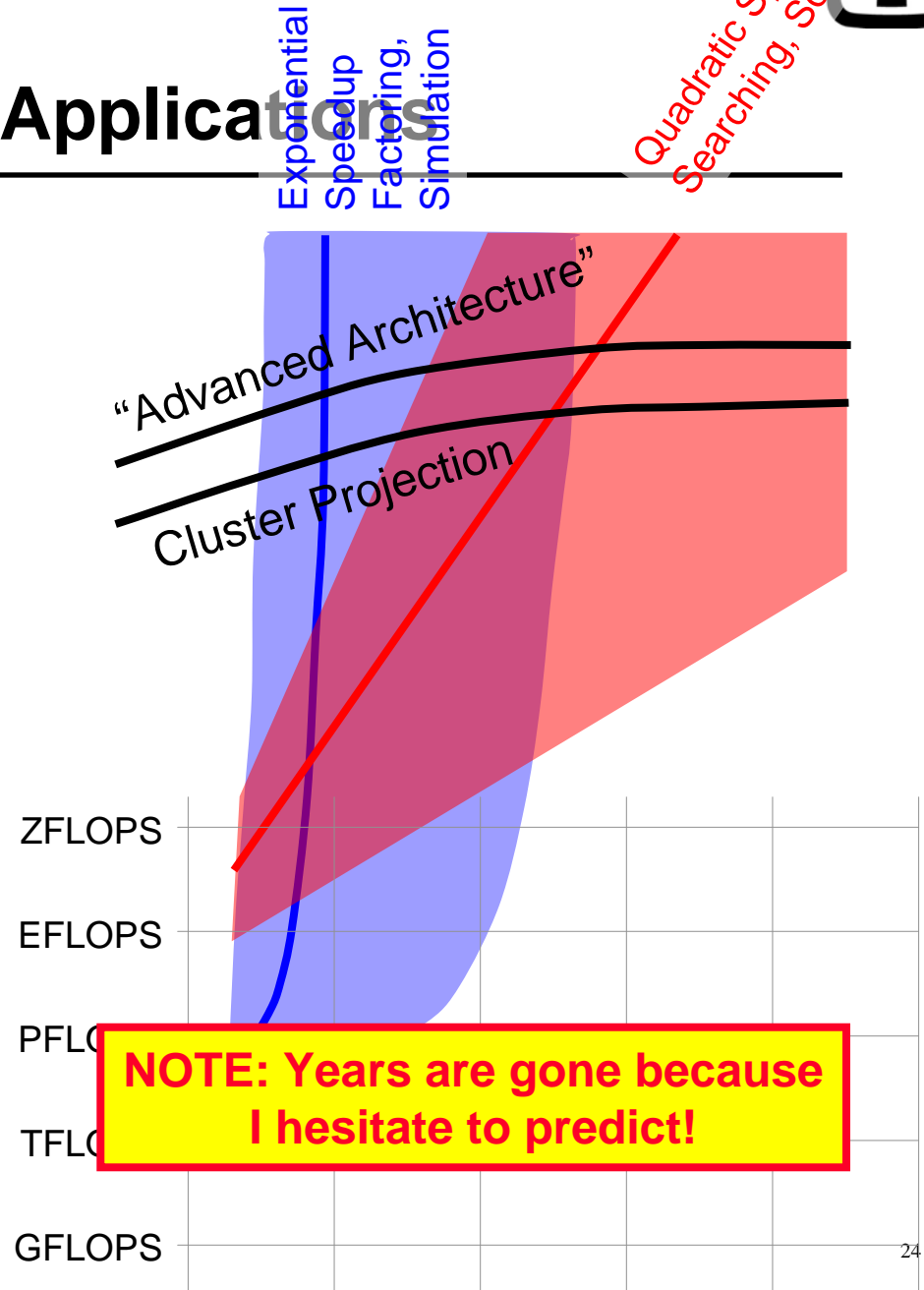
**NOTE: Years are gone because I hesitate to predict!**





# Quantum Applications

- Consider the classical computer equivalent to a Quantum Computer
- First use believed to be factoring in crypt-analysis, with exponential speedup over classical computers (blue)
- Second, a quantum computer can also be used for other applications (pink) with quadratic speedup (e. g. searching)







# Super Roadmap Summary

---

- **The Upside Potential for Innovative Computing is Growing**
- **The industry shift to multi-core just about freezes the performance of non-parallel C++, Fortran, ...**
- **However, there is not even a theoretical contemplated end to computer speed boost that could be termed Moore's Law**
- **However, many people will be disappointed...**





# Outline

---

- **Overview**
  - Insight From a Dinner Conversation in DC
  - Super-Roadmap
- **Limitations to Moore's Law**
  - Transistor Scaling Limits per ITRS
  - Consequence to System Performance per Burger and Keckler Study
- **What It Means and What To Do About It**
  - Legacy C++/Fortran
  - Systolic Array Lessons
  - New Very Parallel Code
  - Special Purpose Assist
  - Analog/Neural Net
- **Over the Horizon**
  - Reversible Logic
  - Quantum Computing





# End of the Roadmap

---

- **ITRS: Exponentials, Innovations, and Equations**
  - SPEC processor numbers and implications
  - The Big Spreadsheet
  - Total power and clock rate model
- **Review of Burger and Keckler Study**
  - Study of throughput under technology scaling
- **Implications**
  - Throughput scaling
  - Cache scaling
  - Bandwidth Scaling





# ITRS Construction Method and Limitations

---

- **ITRS Looks Perfectly Smooth**
  - Yes indeed, this is due to the concept of “targets”
    - $\sqrt{2}$  reduction in line width every 3 years
    - 17%/year increase in clock rate
  - Roadmap based on Excel spreadsheet with targets, inputs, and dependent variables
- **Limitations of ITRS Approach**
  - System performance involves dozens of interrelated variables
  - Smooth scaling is targeted for the dozen variables reported
  - By tying a dozen variables to a straight line, other variables become “dependent”





# Technology Model

- Two or three year interval between  $\sqrt{2}$  reductions in line width
  - Reducing line width by  $\sqrt{2}$  doubles the number of devices
- However, ability to predict the future is imperfect →

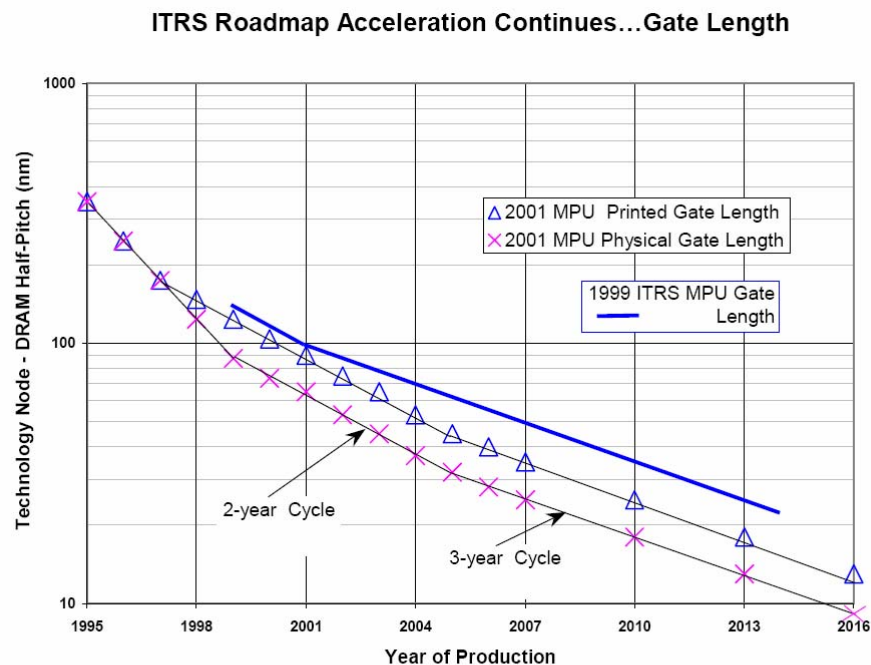


Figure 8 ITRS Roadmap Acceleration Continues—Gate Length Trends

ITRS 2001 edition Executive Summary





# End of the Roadmap

---

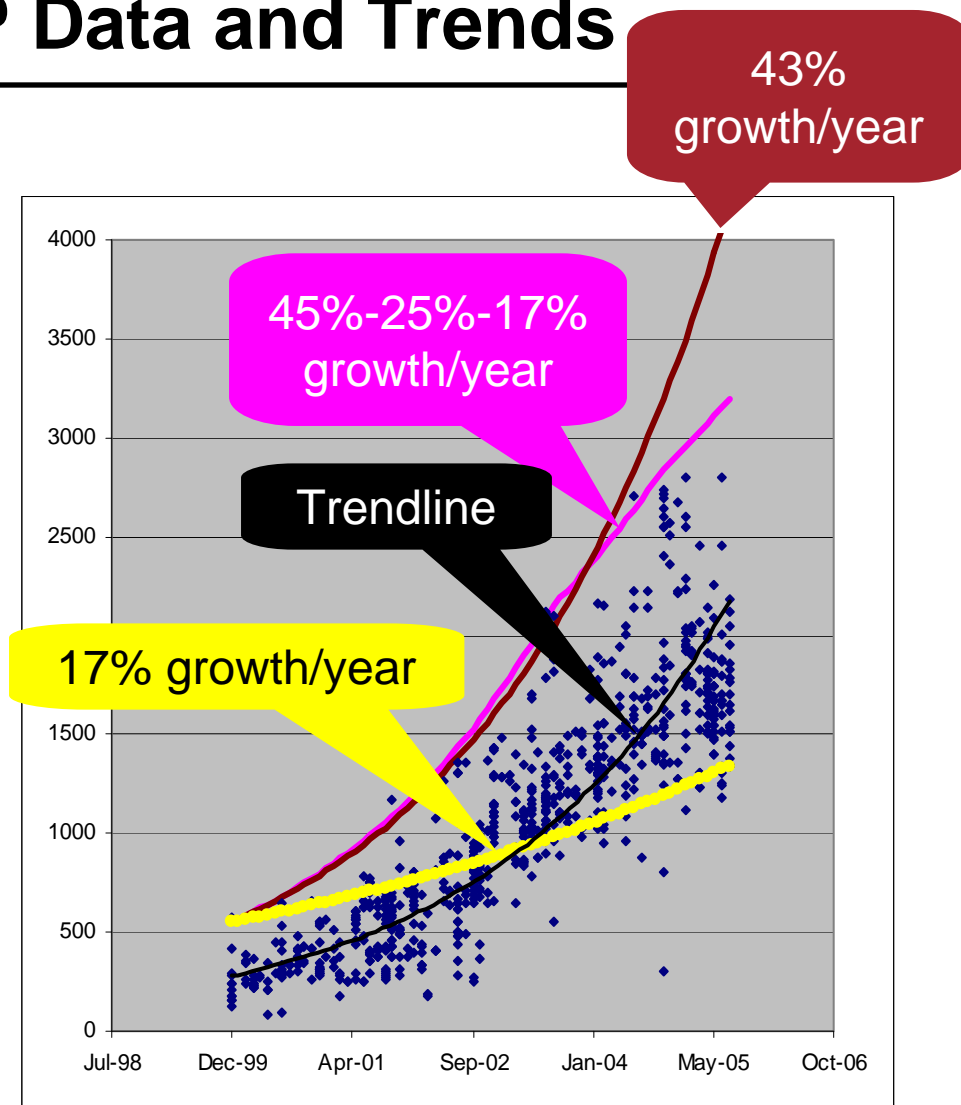
- **ITRS: Exponentials, Innovations, and Equations**
  - SPEC processor numbers and implications
  - The Big Spreadsheet
  - Total power and clock rate model
- **Review of Burger and Keckler Study**
  - Study of throughput under technology scaling
- **Implications**
  - Throughput scaling
  - Cache scaling
  - Bandwidth Scaling





# Per Core SpecFP Data and Trends

- Plot of 785 SpecFP submissions, considering only one core
- 43% per year is an important figure
  - ITRS projection
  - Excel's trendline
  - Erik's plot of "top of envelope"
- However, we are falling short of 43% growth



Data from Spec.org, per core numbers,  
entered into Excel spreadsheet for graphing





# End of the Roadmap

---

- **ITRS: Exponentials, Innovations, and Equations**
  - SPEC processor numbers and implications
  - **The Big Spreadsheet**
  - Total power and clock rate model
- **Review of Burger and Keckler Study**
  - Study of throughput under technology scaling
- **Implications**
  - Throughput scaling
  - Cache scaling
  - Bandwidth Scaling





# ITRS Spreadsheet

---

- Review spreadsheet interactively in Excel
- Points to make
  - Illustrate role and implementation of “targets”
    - Line width
    - Clock rate
  - Illustrate user inputs
    - Sub threshold adjustment factors rows 34 & 36
  - Illustrate rows derived by calculation
    - Illustrate iteration to target
    - Illustrate HP LOP LSTP
- Draw conclusions
  - Industry defines targets
  - Table preparer adds value by scheduling innovations to meet targets
  - Validity depends on innovations occurring on schedule
- Limited example next slide





# ITRS Spreadsheet Structure

Target is exponential  
in "Years in Future"

Line Width  
Scaling

Fprocessor is result of  
96 rows of targets,  
inputs, and iterative  
calculation

Result usually  
matches to one  
decimal place!

ITRS 2003  
supplementary  
material





# User Inputs

- Some factors will scale exponentially by definition, yet others will scale based on projections of engineers
- Supply voltage, doping levels, layer thicknesses, leakage, geometry, mobility, parasitic capacitance

J34 = 0.8

These values are typed-in, based on schedule in next slide

	A	B	C	E		J	K	L
32	<b>Off-State Current/Threshold-Voltage Parameters</b>							
33	Source/Drain Subthreshold Off-State Leakage Drain Current	uA/um	Idrain-off	0.03	0.05	0.05	0.05	0.07
34	Sub-threshold Slope Adjustment Factor (Full Depletion/Dual-Gate Effects) (0-1)		Param-Dual-Gate1	1.0	1.0	1.0	1.0	0.8
35	Sub-threshold Slope	mv/dec	SS	83	86	85	87	79
36	Threshold Voltage Adjustment Factor (Full Depletion/Dual-Gate Effects) (0-1)		Param-Dual-Gate2	1.0	1.0	1.0	1.0	0.8
	Drain Current Used for Vt Definition	uA/um	Idrain-Vt-def	100	110	100	100	100

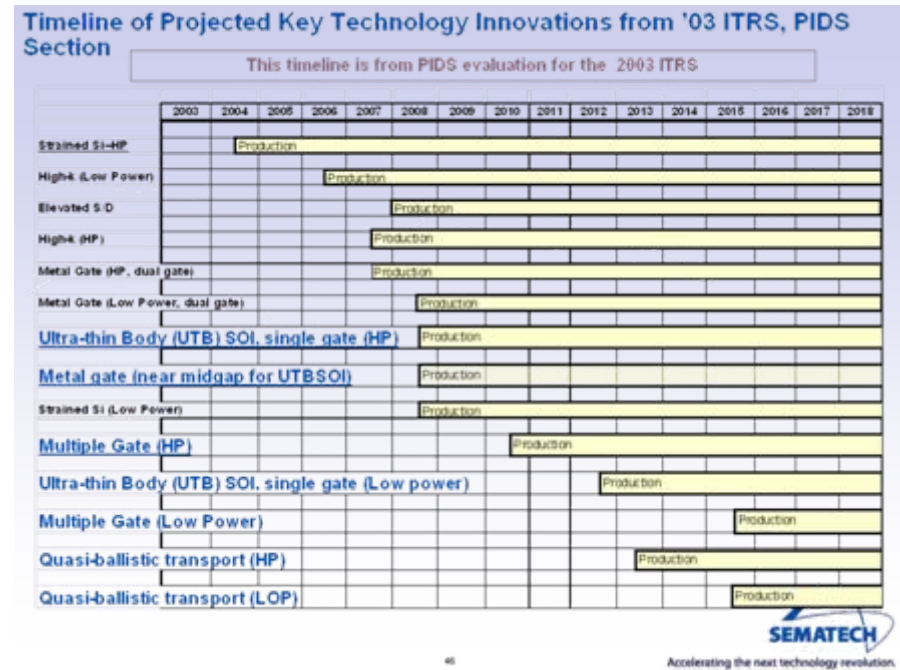
ITRS 2003-supplementary material





# Schedule of Innovations

- To make the calculations fit the projection of a smooth “Moore’s Law,” certain variables must be adjustable
- The independent variables are a “schedule of innovations,” or technology advances that must enter production on certain years

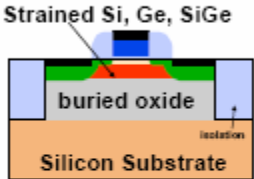
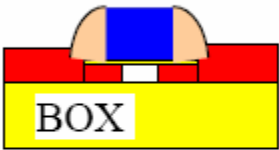
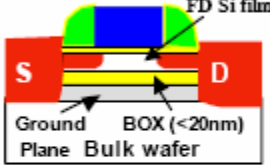
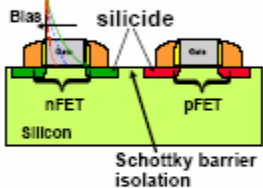
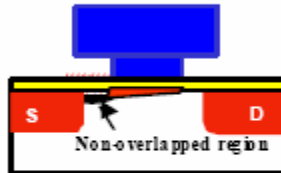


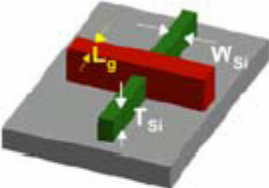
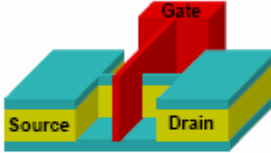
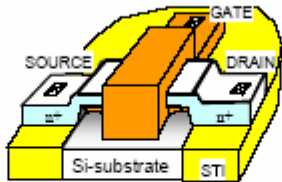


MOSFET Scaling Trends, Challenges, and Key Technology Innovations through the End of the Roadmap, Peter M. Zeitzoff





# ITRS Transistor Geometries

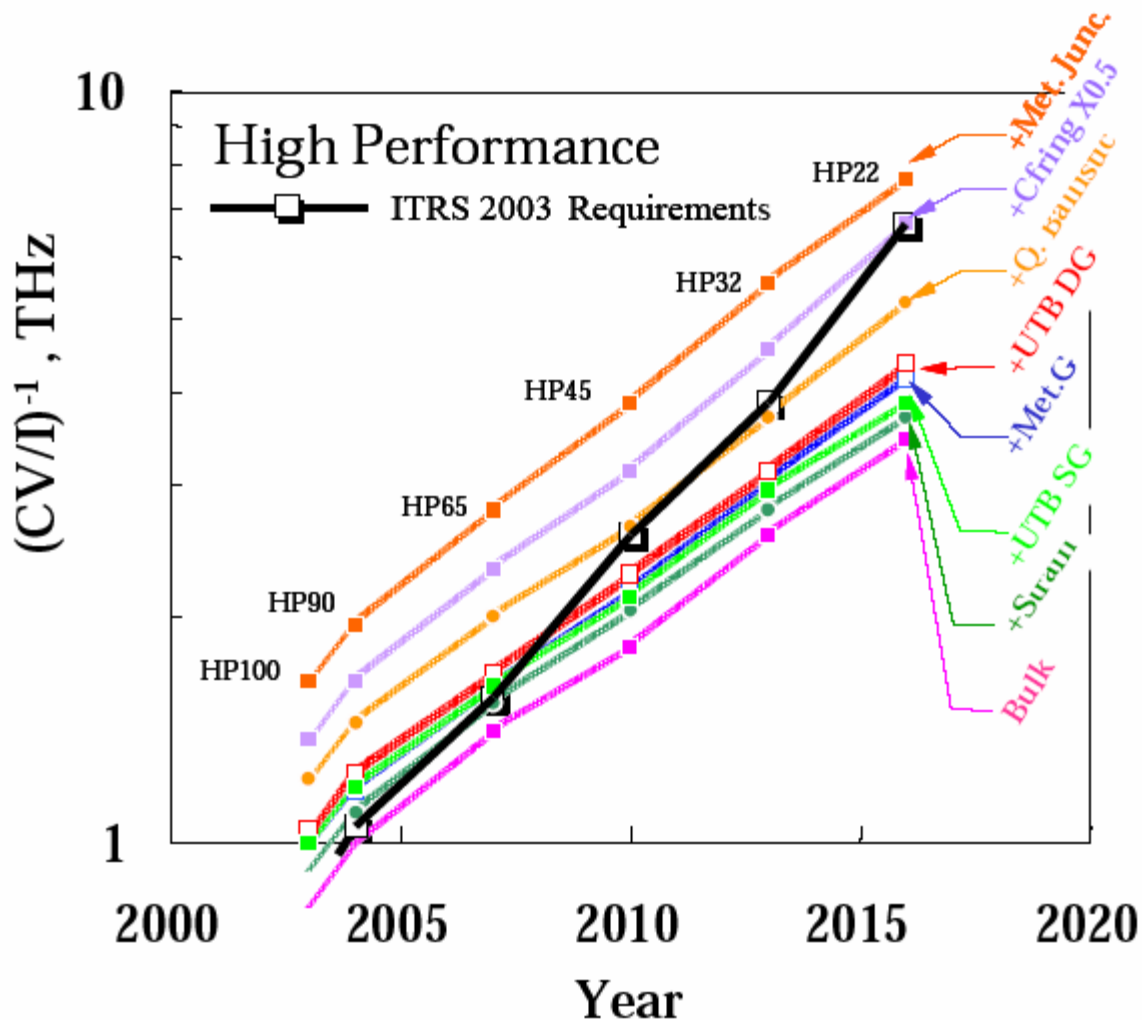
<i>Transport-enhanced FETs</i>	<i>Ultra-thin Body SOI FETs</i>		<i>Source/Drain Engineered FETs</i>	
				
Strained Si, Ge, SiGe, SiGeC or other semiconductor; on bulk or SOI	Fully depleted SOI with body thinner than 10 nm	Ultra-thin channel and localized ultra-thin BOX	Schottky source/drain	Non-overlapped S/D extensions on bulk, SOI, or DG devices

<i>N-Gate (<math>N &gt; 2</math>) FETs</i>	<i>Double-gate FETs</i>			
				
Tied gates (number of channels $> 2$ )	Tied gates, side-wall conduction	Tied gates planar conduction	Independently switched gates, planar conduction	Vertical conduction





# ITRS Technology Progression







# End of the Roadmap

---

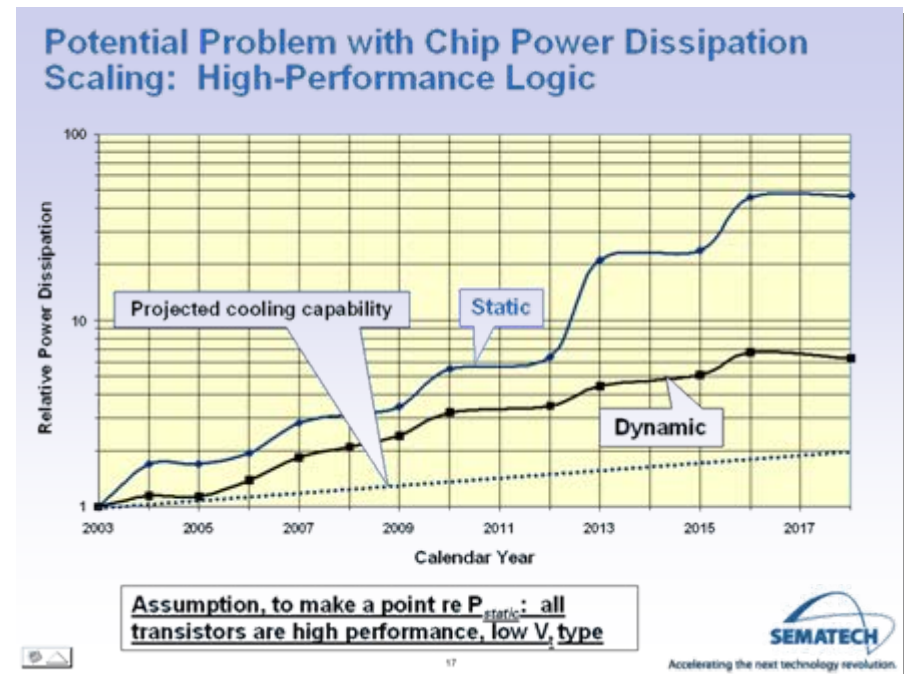
- **ITRS: Exponentials, Innovations, and Equations**
  - SPEC processor numbers and implications
  - The Big Spreadsheet
  - Total power and clock rate model
- **Review of Burger and Keckler Study**
  - Study of throughput under technology scaling
- **Implications**
  - Throughput scaling
  - Cache scaling
  - Bandwidth Scaling





# Power Dissipation

- By targeting a smooth exponential increase in performance over time, power dissipation becomes a dependent variable
- Power dissipation per  $\mu P$  chip is not a reported parameter
- Chart shows result



MOSFET Scaling Trends, Challenges, and Key Technology Innovations through the End of the Roadmap, Peter M. Zeitzoff





# Processor Clock Rate

- Processor operating frequency 10 gate delays with 30% latch overhead
- Gate delay assumes FO3, 2× parasitic capacitance
- Gate delay assumes CV<sup>2</sup> charging, hence supply voltage dependence
- However, these are gate level, not system level

SUM  $= (1 / (E94 * E92 * (1 + E95 / 100))) * 1000$  ITRS 2003 supplementary material

	A	B	C	D	E
1	HP PIDS Worksheet	Units	Variables	Parameter Equations	Spreadsheet Con
2	Version: Aug 04, 2003 -01			(All variables assumed to be of similar dimensional units)	Jim Chung (508)
3	General Parameters				Peter Zeitsoff (5
4	Year in Production		Year	Parameter from ORTC	2003
5	Years in Future		Delta-year	Delta-year = Year - 2003	0
6	Technology Generation		Node	Parameter from ORTC	
92	Nominal Gate Delay (NAND Gate)	ps	Tau-NAND	Tau-NAND = Tau-inverter * Param-NAND-log-eh * Param-NAND-ele-eh	30
93	Nominal Gate Delay Scaling Target	ps	Tau-NAND-target	Tau-NAND-target = Base-NAND / (1 + Yearly-rate) ^ Delta-year	30
94	Nominal Processor Gate Delays per Cycle		Param-gate-cycle	User-Specified Input Parameter	10
95	Latch Overhead Percentage of Cycle Time	%	Param-latch-overhead	User-Specified Input Parameter	30
96	Nominal HP Processor Operating Frequency	GHz	Fprocessor	Fprocessor = 1 / (Param-gate-cycle * Tau-NAND * (1 + Param-latch-overhead))	$= (1 / (E94 * E92 * (1 + E95 / 100))) * 1000$
97	Nominal HP Processor Operating Frequency Target	GHz	Fprocessor-target	Fprocessor-target = Base-freq * (1 + Yearly-rate) ^ Delta-year	2.5





# ITRS Scaling Conclusions

---

- **Optimism**

- Density doubles every three years
  - 26% per year
- Clock rate rises 17% per year
- Sum is 43%/year!
  - Reasonably close to the 41%/year of ideal scaling!

- **Limits of Applicability**

- Power dissipation partially covered
  - However, power dissipation per chip rises
  - Leakage power not covered
- Timing based on gates, not architecture
  - Wiring delay calculated, but not part of timing model





# End of the Roadmap

---

- **ITRS: Exponentials, Innovations, and Equations**
  - SPEC processor numbers and implications
  - The Big Spreadsheet
  - Total power and clock rate model
- **Review of Burger and Keckler Study**
  - Study of throughput under technology scaling
- **Implications**
  - Throughput scaling
  - Cache scaling
  - Bandwidth Scaling





# Outline

---

- **Overview**
  - Insight From a Dinner Conversation in DC
  - Super-Roadmap
- **Limitations to Moore's Law**
  - Transistor Scaling Limits per ITRS
  - Consequence to System Performance per Burger and Keckler Study
- **What It Means and What To Do About It**
  - Legacy C++/Fortran
  - Systolic Array Lessons
  - New Very Parallel Code
  - Special Purpose Assist
  - Analog/Neural Net
- **Over the Horizon**
  - Reversible Logic
  - Quantum Computing





# Scaling of Microprocessor Performance

---

- For a given design, performance proportional to clock rate
- However, designs change with technology
  - More transistors lead to architectures with more “instructions per clock”
  - Signal propagation (wire) delays lead to more pipelining
  - More pipelining leads to larger cache miss penalty
  - Cache miss penalty and desire to run larger programs (a. k. a. “code bloat”) leads to larger caches
- Question: What is the roadmap for microprocessor performance?





# How to Project Uniprocessor Performance

---

- Let's assume industry makes the innovations called for by the ITRS on schedule
- However, companies will not be constrained to do everything like the ITRS
  - Engineers can choose any power supply voltage they like
  - Doping levels can be changed

- Evaluate

$\max(\text{SpecFP})$   
engineering  
← choices,  
architecture

and report performance  
and architecture as a  
function of years into the  
future





# UT Austin Study (2000)

---

- **The Study**
  - **Clock Rate versus IPC:  
The End of the Road for  
Conventional  
Microarchitectures,  
Vikas Agarwal, M.S.  
Hrishikesh, Stephen W.  
Keckler, Doug Burger.  
27<sup>th</sup> Annual  
International  
Symposium on  
Computer Architecture**
- **Conclusions (to be  
Explained)**
  - **Modified ITRS roadmap  
predictions to be more  
friendly to architectures**
  - **Concluded there would  
be a 12%/year growth...**
  - **However, recent growth  
has been ~30%, with  
industry's maneuver to  
cheat the analysis  
instructive**





# Wire Delay Coverage in ITRS

- Wire delay added to ITRS 2002 edition

Table 62b MPU Interconnect Technology Requirements—Long-term

Year of Production		2010	2013	2016
DRAM $\frac{1}{2}$ Pitch (nm)		45	32	22
MPU/ASIC $\frac{1}{2}$ Pitch (nm)		45	32	22
MPU Printed Gate Length (nm)		25	18	13
MPU Physical Gate Length (nm)		18	13	9
Number of metal levels		10	11	11
Number of optional levels – ground planes/capacitors		4	4	4
Total interconnect length (m/cm <sup>2</sup> ) – active wiring only, excluding global levels [1]		16063	22695	33508
FITs/m length/cm <sup>2</sup> $\times 10^{-3}$ excluding global levels [2]		0.31	0.22	0.15
Jmax (A/cm <sup>2</sup> )—wire (at 105°C)		2.70E+06	3.30E+06	3.90E+06
Imax (mA)—via (at 105°C)		0.1	0.07	0.04
Local wiring pitch (nm)		105	75	50
Local A/R (for Cu)		1.8	1.9	2
Add	Interconnect RC delay 1 mm line (ps)	565	970	2008
Add	Line length where $\tau = RC$ delay (nm)	26	15	9
Cu thinning to minimum pitch due to etching (nm), 10% $\times$ height, 50% areal density, 500 $\mu$ m square array		5	4	3
Intermediate wiring pitch (nm)		135	95	65
Intermediate wiring dual Damascene A/R (Cu wire/via)		1.8/1.6	1.9/1.7	2.0/1.8
Add	Interconnect RC delay 1 mm line (ps)	348	614	1203
Add	Line length where $\tau = RC$ delay (nm)	33	19	11
Cu thinning to minimum intermediate pitch due to etching (nm), 10% $\times$ height, 50% areal density, 500 $\mu$ m square array		12	9	7
Minimum global wiring pitch (nm)		205	140	100
Add	Ratio range(global wiring pitch to intermediate wiring pitch)	1.5 - 10	1.5 - 13.0	1.5 - 16
Global wiring dual Damascene A/R (Cu wire/via)		2.3/2.4	2.4/2.2	2.6/2.3
Add	Interconnect RC delay 1 mm line (ps) at minimum pitch	131	246	452
Add	Line length where $\tau = RC$ delay (nm) minimum pitch	54	30	19
Deleted	Cu thinning global wiring due to etching and areal density (nm), 10% $\times$ height, 50% areal density, 1.5 mm wide wire	94	47	49
Add	Cu thinning of maximum width global wiring due to etching and areal density (nm), 10% $\times$ height, 50% areal density	166	146	130
Cu thinning global wiring due to etching (nm), 10% wide flexure		14	10	8
Connector effective resistivity ( $\mu\Omega$ -cm) Cu intermediate wiring		2.2	2.9	2.6
Etcher/etching thickness (for Cu intermediate wiring) (nm) [3]		5	3.8	2.6
Insulated metal insulator—effective dielectric constant ( $\epsilon$ )		2.1	1.9	1.8
Insulated metal insulator (minimum expected)—bulk dielectric constant ( $\epsilon$ )		<1.9	<1.7	<1.6





# Modeling Wire Delay

- For some year in the future
  - ITRS and other models project a clock rate
  - ITRS and other models project a signal propagation velocity
  - Divide the two figures to get  $d$ =distance traveled in one clock cycle
  - Chip area/ $d^2$  is plotted at right →

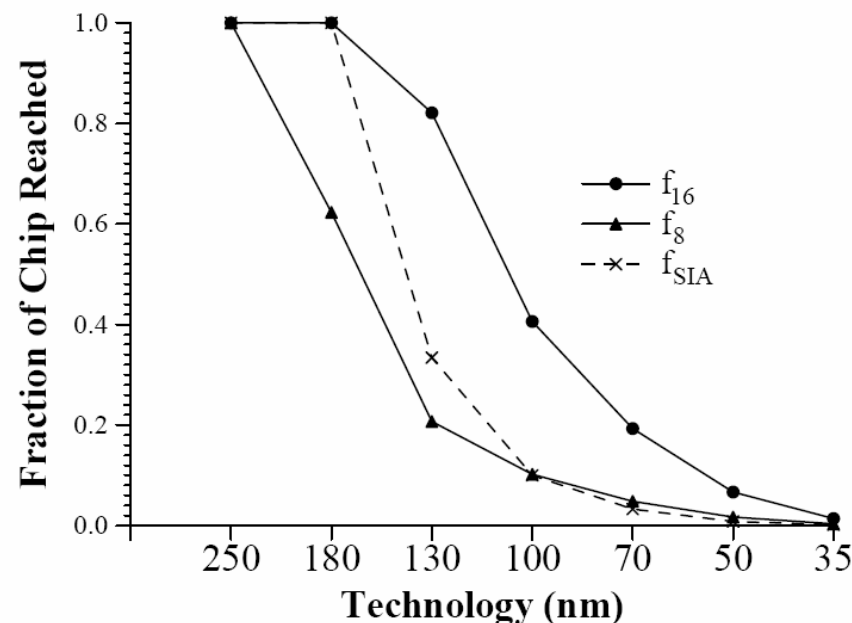


Figure 4: Fraction of total chip area reachable in one cycle.

- Figure 4 from “Clock Rate versus IPC: The End of the Road for Conventional Microarchitectures,” Vikas Agarwal, M.S. Hrishikesh, Stephen W. Keckler, and Doug Burger





# Cache Performance

- Authors used ECacti cache modeling tool
- ECacti lays out caches in terms of banks, associatively, etc.
- As technology progresses, size of cache accessible in 3 cycles decreases
- Remedy is obvious, but has consequences: increase depth of pipelining

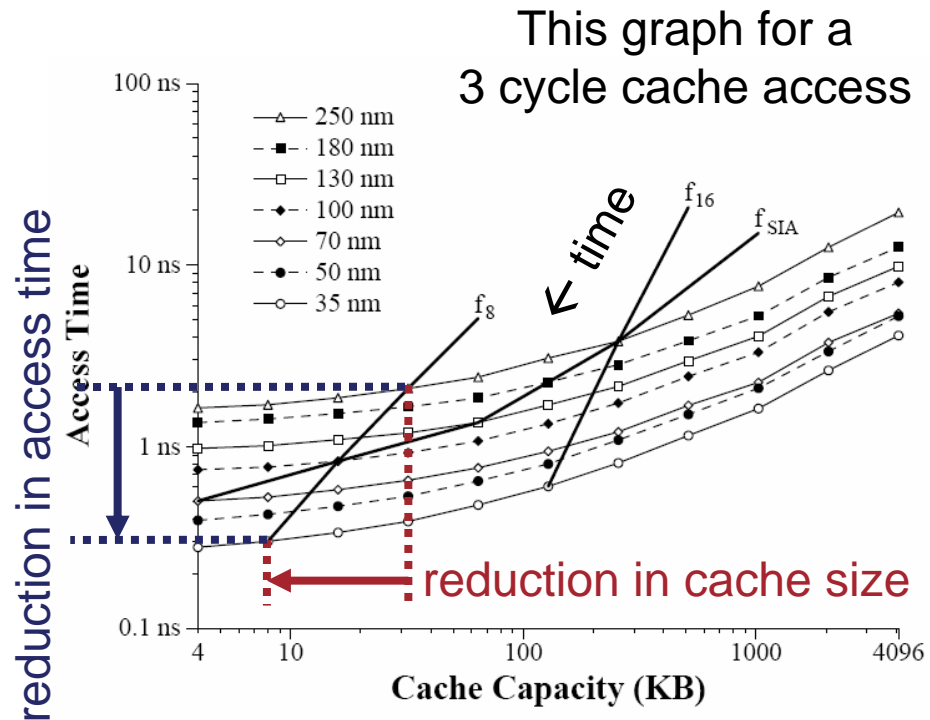


Figure 5: Access time for various L1 data cache capacities.

- Figure 5 from “Clock Rate versus IPC: The End of the Road for Conventional Microarchitectures” Vikas Agarwal, M.S. Hrishikesh, Stephen W. Keckler, and Doug Burger





# Modeling Pipelined $\mu$ P

---

- Authors used SimpleScalar, cycle accurate simulator of a DEC Alpha 21264
- However, actually models hypothetical future  $\mu$ Ps with parameterized
  - Cache parameters
  - Pipeline depth
  - Branch prediction
  - Technology (clock speed)
- Authors used SimpleScalar to model the 18 SPEC95 benchmarks for 500 million instructions each
  - Adjustments to avoid initialization
- Question to answer: What is the best architecture, and how well does it work?





# Simulation Results

- Results shown at right → are noted by author to be “remarkably consistent”
- If fact, the results are almost the same as the clock rate increase
- Conclusion: To first order, SPEC ratings will increase with speed of clock
  - Noting that this analysis is per  $\mu$ P core, and SPEC is for one core

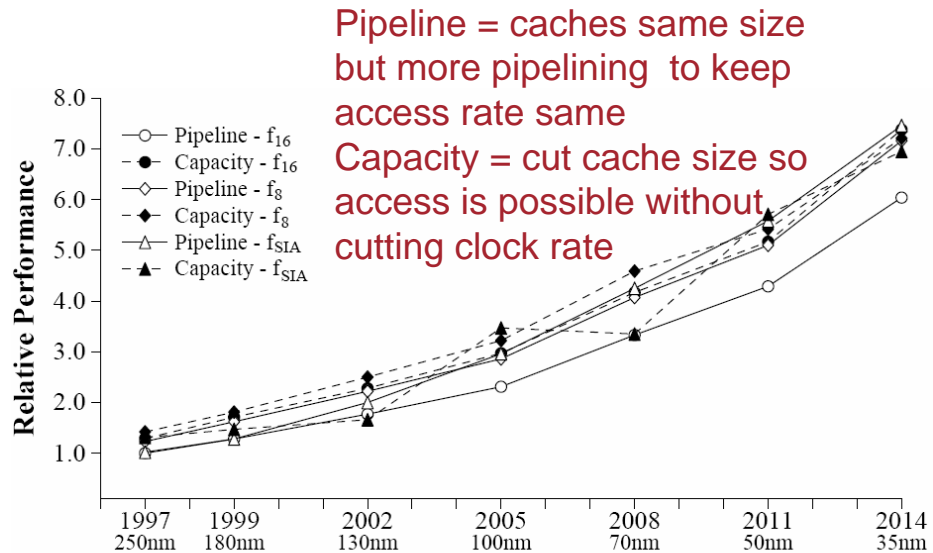


Figure 7: Performance increases for different scaling strategies.

- Figure 7 from “Clock Rate versus IPC: The End of the Road for Conventional Microarchitectures” Vikas Agarwal, M.S. Hrishikesh, Stephen W. Keckler, and Doug Burger





# Study Conclusions and Discussion

- UT Austin study concluded that  $\mu P$  performance should increase at about 12%/year
- However, it actually increased at 30%/year
- What is the discrepancy?
  - It is difficult to predict future
  - Vendors broke study assumptions by increasing power
  - Study was before its time (vendors went multicore this year)

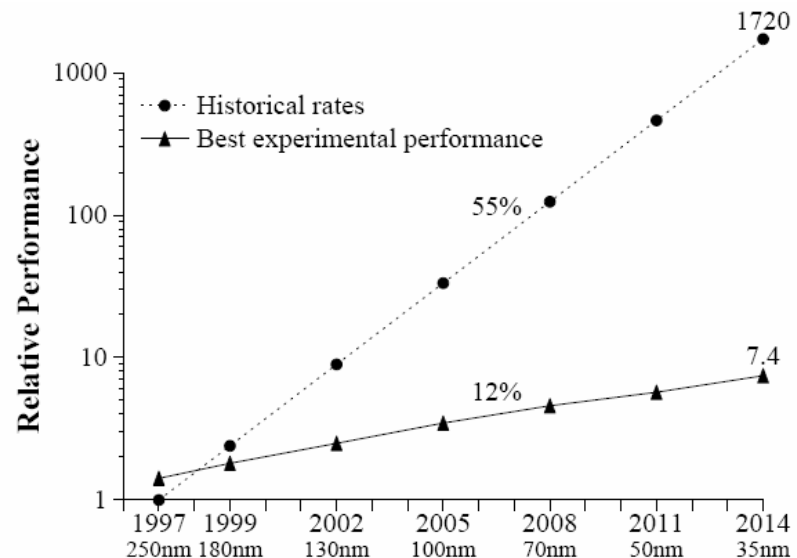


Figure 8: Projected performance scaling over a 17-year span for a conventional microarchitecture.

- Figure 8 from “Clock Rate versus IPC: The End of the Road for Conventional Microarchitectures” Vikas Agarwal, M.S. Hrishikesh, Stephen W. Keckler, and Doug Burger

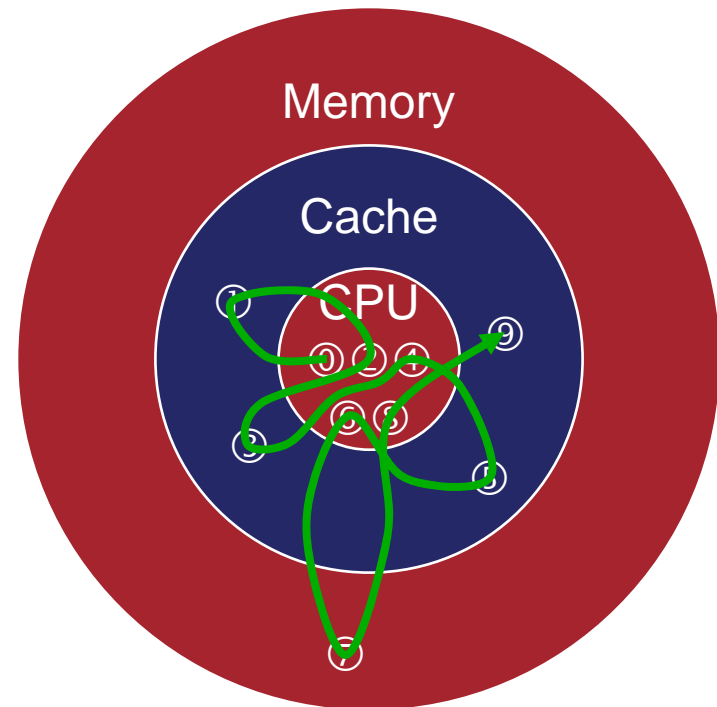




# Model of CPU Performance (Will Be Reused)

---

- Diagram's physical size corresponds to processor's physical size
- Program executes by visiting nodes  
①②③④⑤⑥⑦⑧⑨,  
moving at a propagation velocity  $\propto c$
- Evens at center due to Von Neumann architecture
- Performance is rate at which nodes are visited







# Projecting Applications Performance

---

- **Review of Issues**

- Thread speed & parallelism
- Inner loop memory requirements
- FLOPS/watt
- Devices per chip (multi-core scaling)
- Surface-to-area ratio
- Load imbalance revealed by synchronization overhead

- **Example**

- Instructor led example of projecting performance of a mesh algorithm





# Technology Scaling and Algorithms

---

- **Assumptions**
  - You have a fixed budget to buy and run computers
  - Technology scales according to ITRS
- **Question**
  - How will the performance of algorithms change as a function of time?
- **Solution Approach**
  - Find the scalability of an algorithm as a function of the “scaling” of the computer’s technology
- **Issues Generating Rules**
  - Thread speed & parallelism
  - Inner loop memory
  - FLOPS/watt
  - Devices per chip (or whatever)
  - Surface-to-area ratio
  - Load balance
    - App. Determined
    - Stability





# Projecting Applications Performance

---

- **Review of Issues**

- **Thread speed & parallelism**
- **Inner loop memory requirements**
- **FLOPS/watt**
- **Devices per chip (multi-core scaling)**
- **Surface-to-area ratio**
- **Load imbalance revealed by synchronization overhead**

- **Example**

- **Instructor led example of projecting performance of a mesh algorithm**





# Thread Speed and Parallelism

---

- **Runtime  $\geq$  sequential ops  $\div$  thread speed**
- **Single thread FLOPS rate determined by**
  - **Gate speed**
    - ITRS tell you this
  - **Architecture**
    - ~9 gate delays in a  $\mu$ P
    - Inflexible
  - **Communications speed**
    - Memory latency
- **The best algorithms have variable parallelism**
  - Each thread controls an array of cells
  - Size of the array can be cut, but not below 1 cell
- **Some algorithms have fixed parallelism**
  - Tough luck
- **Conclusion**
  - Optimization





# Projected Clock Rate Increases

- 2004 Update shows clock rates rising to 53 GHz by 2018
  - Not based on architecture

- The ITRS table projects clock rates based on inverter and latch delay, not accounting for system issues
- Recent historical information suggests much slower clock rate increases
  - Cancellation of certain microprocessors and shift to multi-core

Table 4d Performance and Package Chips: Frequency, On-chip Wiring Levels—Long-term Years  
UPDATED

	Year of Production	2010	2011	2012	2013	2014	2015	2016	2017	2018
	Technology Node	hp45			hp32			hp22		
WAS	DRAM ½ Pitch (nm)	45		35	32		25	22		18
IS	DRAM ½ Pitch (nm)	45	40	35	32	28	25	22	20	18
WAS	MPU/ASIC Metal 1 (M1) ½ Pitch (nm)	54		42	38		30	27		21
IS	MPU/ASIC Metal 1 (M1) ½ Pitch (nm)	54	48	42	38	34	30	27	24	21
WAS	MPU/ASIC ½ Pitch (nm) (Un-contacted Poly)	45	-	35	32	-	25	22	-	18
IS	MPU/ASIC ½ Pitch (nm) (Un-contacted Poly)	45	40	35	32	28	25	22	20	18
WAS	MPU Printed Gate Length (nm) ††	25		20	18		14	13		10
IS	MPU Printed Gate Length (nm) ††	25	22	20	18	16	14	13	11	10
WAS	MPU Physical Gate Length (nm)	18		14	13		10	9		7
IS	MPU Physical Gate Length (nm)	18	16	14	13	11	10	9	8	7
Chip Frequency (MHz)										
	On-chip local clock	15,079		20,065	22,980		33,403	39,683		53,207
	Chip-to-board (off-chip) speed (high-performance, for peripheral buses)[1]	9,536		14,901	18,626		29,103	36,379		56,843
WAS	Maximum number wiring levels—maximum	16		16	16		17	18		18
IS	Maximum number wiring levels—maximum	16	16	16	16	17	17	18	18	18
WAS	Maximum number wiring levels—minimum	12		12	12	-	13	14	-	14





# Projecting Applications Performance

---

- **Review of Issues**
  - Thread speed & parallelism
  - Inner loop memory requirements
  - FLOPS/watt
  - Devices per chip (multi-core scaling)
  - Surface-to-area ratio
  - Load imbalance revealed by synchronization overhead
- **Example**
  - Instructor led example of projecting performance of a mesh algorithm





# Inner Loop Working Set

---

- The application's inner loop will have a “cache working set” of storage
  - This working set will take up  $d \times d$  chip area
- Minimum access time will be  $2d \div v$ 
  - $v$  is signal propagation velocity
  - modulo constants
- Is this some hypothetical architectural thing?
  - Not necessarily, applies to existing  $\mu$ Ps where working set is in existing cache
- Implication to algorithm
  - Cutting working set size can cut running time
  - Physics supercedes complexity theory





# Implications of Inner Loop Working Set

---

- **Runs against Area-Volume Rule**
  - Fewer cells per CPU increases communications cost ☹️
  - At some point cutting cells per CPU lets all cells fit in cache, or other local memory 😊
- **Impacts tables**
  - Option A: compute  $f(x)$  when needed
  - Option B: precompute  $f(x)$ , store in a x Megabyte table
    - Option B may cut clock rate for everything else
      - No universal answer here
- **Allocate data structures to memories at different distances?**





# Projecting Applications Performance

---

- **Review of Issues**
  - Thread speed & parallelism
  - Inner loop memory requirements
  - **FLOPS/watt**
  - Devices per chip (multi-core scaling)
  - Surface-to-area ratio
  - Load imbalance revealed by synchronization overhead
- **Example**
  - Instructor led example of projecting performance of a mesh algorithm





# FLOPS/Watt

---

- **Thermodynamic limit at  $k_B T \log 2$** 
  - Currently operating at  $100,000 k_B T$
  - ITRS goes to about  $100 k_B T$
  - Unexplored gulf between  $100 k_B T$  and  $.7 k_B T$
- **Thermodynamic limit can be beat with reversible logic and Quantum**
- **Implications**
  - **Corollary: everything proportional to power**
    - Mfg cost
    - Operating cost
  - **Cost of running an algorithm depends on total FLOPS**
    - Cut FLOPS
    - Running time is a different story





# Projecting Applications Performance

---

- **Review of Issues**
  - Thread speed & parallelism
  - Inner loop memory requirements
  - FLOPS/watt
  - Devices per chip (multi-core scaling)
  - Surface-to-area ratio
  - Load imbalance revealed by synchronization overhead
- **Example**
  - Instructor led example of projecting performance of a mesh algorithm





# Device Density Scaling

---

- **Device density is projected to scale at 2× per three years**
- **There is a lot of innovation**
  - **Lithographic line width continues to shrink**
  - **DNA self assembly**
  - **Others**
- **We don't seem close to theoretical limits**





# Projecting Applications Performance

---

- **Review of Issues**
  - Thread speed & parallelism
  - Inner loop memory requirements
  - FLOPS/watt
  - Devices per chip (multi-core scaling)
  - Surface-to-area ratio
  - Load imbalance revealed by synchronization overhead
- **Example**
  - Instructor led example of projecting performance of a mesh algorithm





# Bandwidth Scaling

---

- **Overview: Bandwidth will continue to scale**
- **Theoretically, the limit on bandwidth is way out**
- **According to the ITRS Roadmap**
  - **Number of bonding pads on a chip becomes constant**
  - **Bandwidth per bonding pad equals internal clock rate (?)**
- **However, there are innovative solutions in the works**
  - **Optical interconnect**
  - **Capacitive interconnect**
- **For long haul communications**
  - **Optics has practically infinite bandwidth**





# Projecting Applications Performance

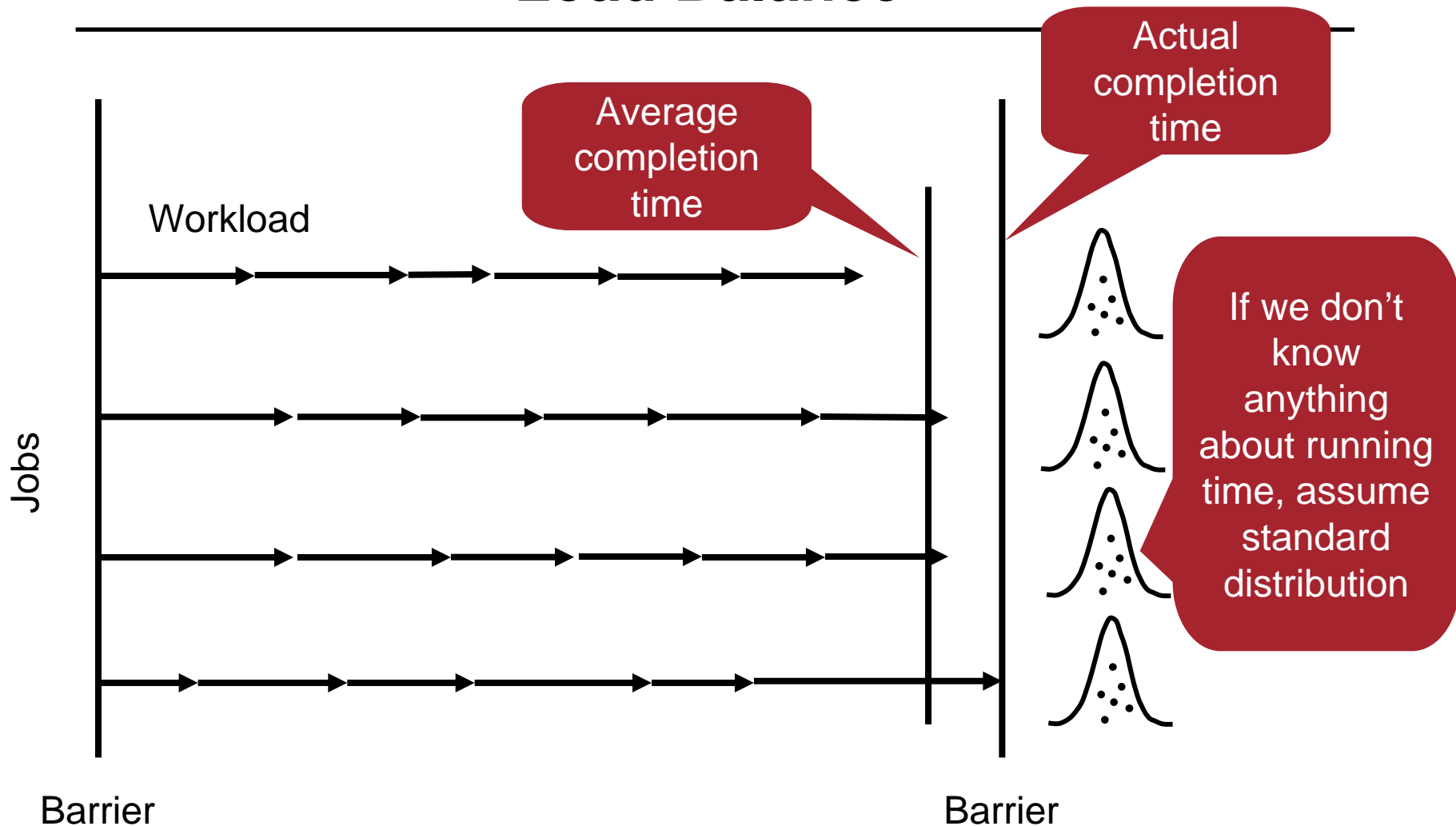
---

- **Review of Issues**
  - Thread speed & parallelism
  - Inner loop memory requirements
  - FLOPS/watt
  - Devices per chip (multi-core scaling)
  - Surface-to-area ratio
  - Load imbalance revealed by synchronization overhead
- **Example**
  - Instructor led example of projecting performance of a mesh algorithm





# Load Balance

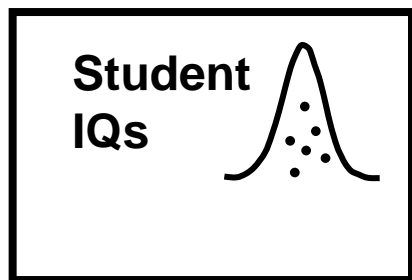






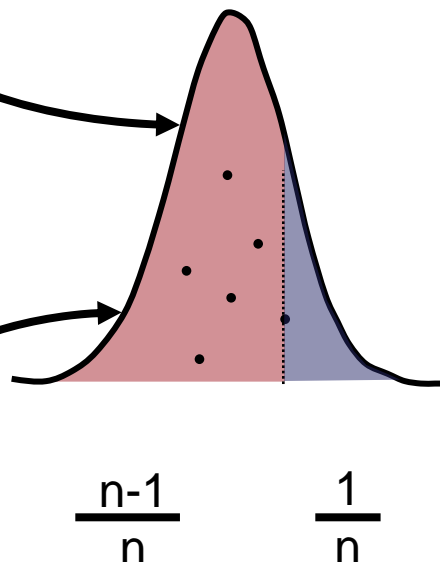
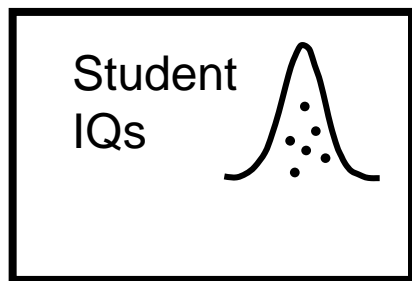
# Maximum IQ of a Class in Your Kids School

## Classroom 1



$\Sigma$  IQs will have bell curve as well

## Classroom n



- Each child has average IQ 100 and std of 15
  - Mean and std of task runtime
- Each class has total IQ of  $n \times 100$  and std of  $n^{1/2} \times 15$ 
  - Statistics of per node time between barriers
- Max average is inverse of cumulative normal distribution evaluated at  $n$

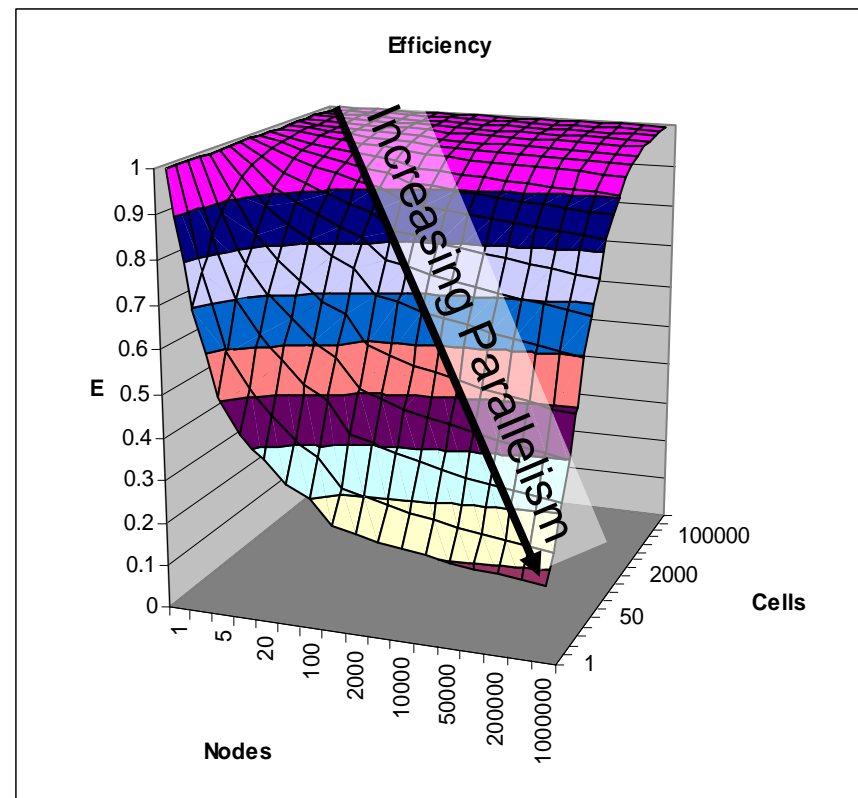




# Efficiency Loss Due To Load Balance

- Load imbalance becomes an issue when there are less than 10s to 100s of tasks per node
  - Presuming  $\text{mean} \approx \text{std}$
- Implications
  - This creates a ceiling to the amount of parallelism, unless
  - tasks can be shared

- Plot Mean=Std







# Projecting Applications Performance

---

- **Review of Issues**
  - Thread speed & parallelism
  - Inner loop memory requirements
  - FLOPS/watt
  - Devices per chip (multi-core scaling)
  - Surface-to-area ratio
  - Load imbalance revealed by synchronization overhead
- **Example**
  - Instructor led example of projecting performance of a mesh algorithm





## Example Problem: Future Mesh Problem

---

- We are given year 20XX
- 1. Outer Loop of Process:  
Pick Number of Cores
  - Processors are likely to be available with different numbers of cores – and there is no obligation to use all the cores on a chip
  - Repeat the following with 1, 2, 4... up to the max cores that will fit on a 20XX die
- 2. Look up 20XX in ITRS
  - Note device density
  - Note clock rate
- 3. Figure out how much cache you should have
  - Chip area goes to cores and cache
  - After taking out the area occupied by cores, the rest is cache
  - Track heat production (for use later)

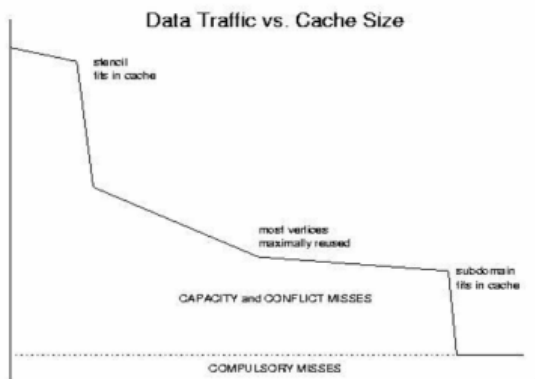




## Example, Part 2

- 4. Using algorithmic information and cache size, figure out at what tier the code will run, per discussion earlier. The level may strongly influence performance

As successive workingsets “drop” into a level of memory, capacity (and with effort conflict) misses disappear, leaving only compulsory, reducing demand on main memory bandwidth



- Levels are
  - Stencil in cache
  - Vertices in cache
  - Subdomain in cache
- 5. From level and “grind time,” figure out B:F ratio between CPU chip and main memory
- 6. Figure out likely memory bandwidth, either by using pins per ITRS specs or standard memory busses





## Example, Part 3

---

- **7. Calculate interchip communications rates**
  - This generally involves sending and receiving the “halo” from each node
  - Depending on architecture, could be from memory or CPU
  - Also in B:F ratios
- **8. Overall throughput will be minimum of**
  - FLOPS
  - Memory bandwidth divided by B:F ratio for memory
  - MPI bandwidth divided by B:F ratio for MPI
  - There has been some discussion of throttling chips due to excessive power





## Example, Part 4

---

- **Note: All rates should be adjusted for “percentage of peak.” If nothing else is known, use percentage of peak numbers for similar architectures**
- **9. Iterate to best solution, by going to step 1**
  - **varying the number of cores in a chip, devoting all area not occupied by cores with cache**
  - **turning off cores, sharing their cache**
  - **spreading problem over more or fewer nodes**





## Example, Part 5

---

- **10. Final step: The process just described is a mixture of analysis and design. The result will be meaningless if a vendor doesn't produce the required chip. For example, if your ideal design requires  $2\frac{1}{2}$  cores, you're probably out of luck.**





# Outline

---

- **Overview**
  - Insight From a Dinner Conversation in DC
  - Super-Roadmap
- **Limitations to Moore's Law**
  - Transistor Scaling Limits per ITRS
  - Consequence to System Performance per Burger and Keckler Study
- **What It Means and What To Do About It**
  - Legacy C++/Fortran
  - Systolic Array Lessons
  - New Very Parallel Code
  - Special Purpose Assist
  - Analog/Neural Net
- **Over the Horizon**
  - Reversible Logic
  - Quantum Computing





# Outline

---

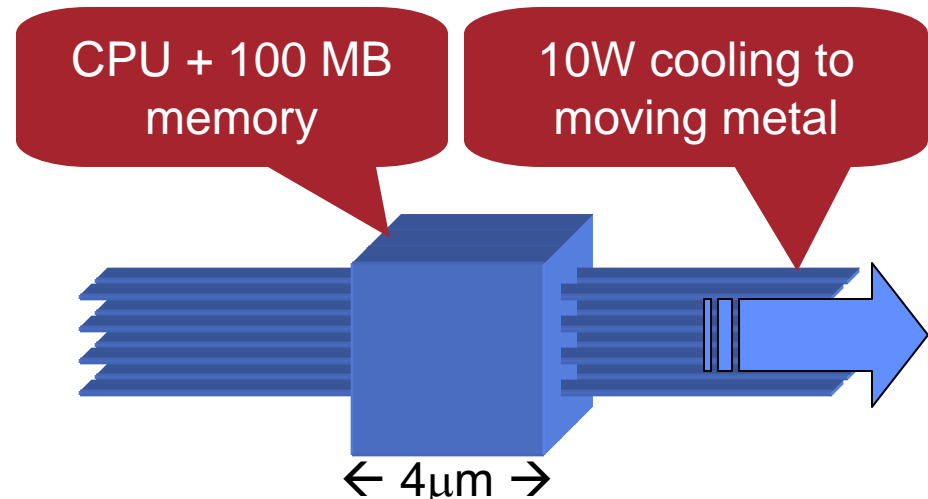
- **Overview**
  - Insight From a Dinner Conversation in DC
  - Super-Roadmap
- **Limitations to Moore's Law**
  - Transistor Scaling Limits per ITRS
  - Consequence to System Performance per Burger and Keckler Study
- **What It Means and What To Do About It**
  - **Legacy C++/Fortran**
  - Systolic Array Lessons
  - New Very Parallel Code
  - Special Purpose Assist
  - Analog/Neural Net
- **Over the Horizon**
  - Reversible Logic
  - Quantum Computing





# Fastest Possible C++ or Fortran Program

- How fast could a C++ or Fortran program ever run?
- Limited by memory access time to ~100 MBytes of data
- Ref. K. Eric Drexler, Nanosystems: Molecular Machinery, Manufacturing, and Computation
- Parameters for 100  
← megabytes memory



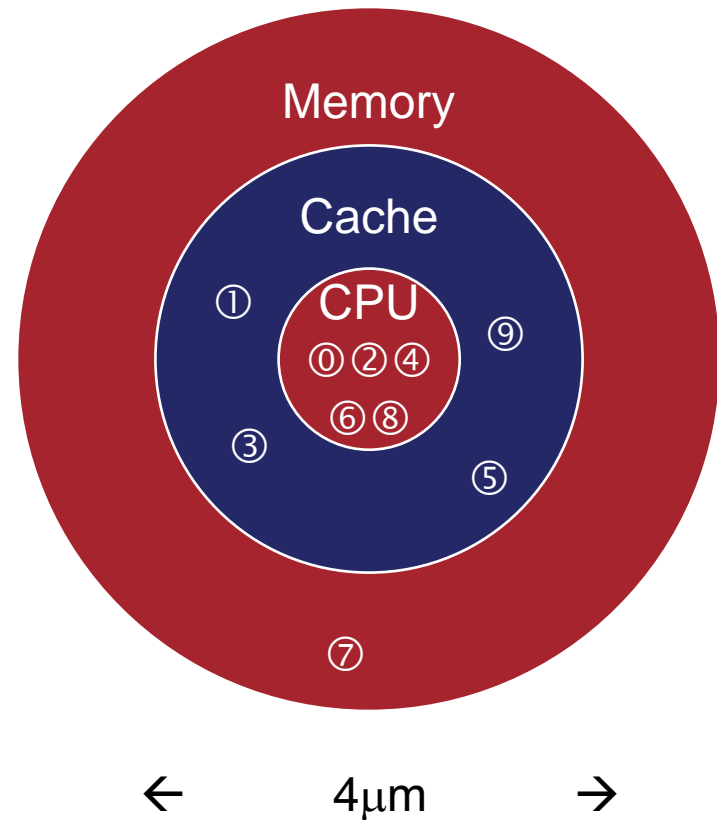
- Cooling method: ✓
- Back of envelope: 21 THz
- Conclusion: Faster than CMOS slower than Quantum Computer
- No research in this area





# Single CPU Performance

- Program executes by visiting nodes  
① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨
- To go fast
  - Raise speed of motion
  - Shrink physical size
  - Organize to put nodes closer to center
  - Predict order of access
- C++ and Fortran programs have little predictability and stochastic distribution





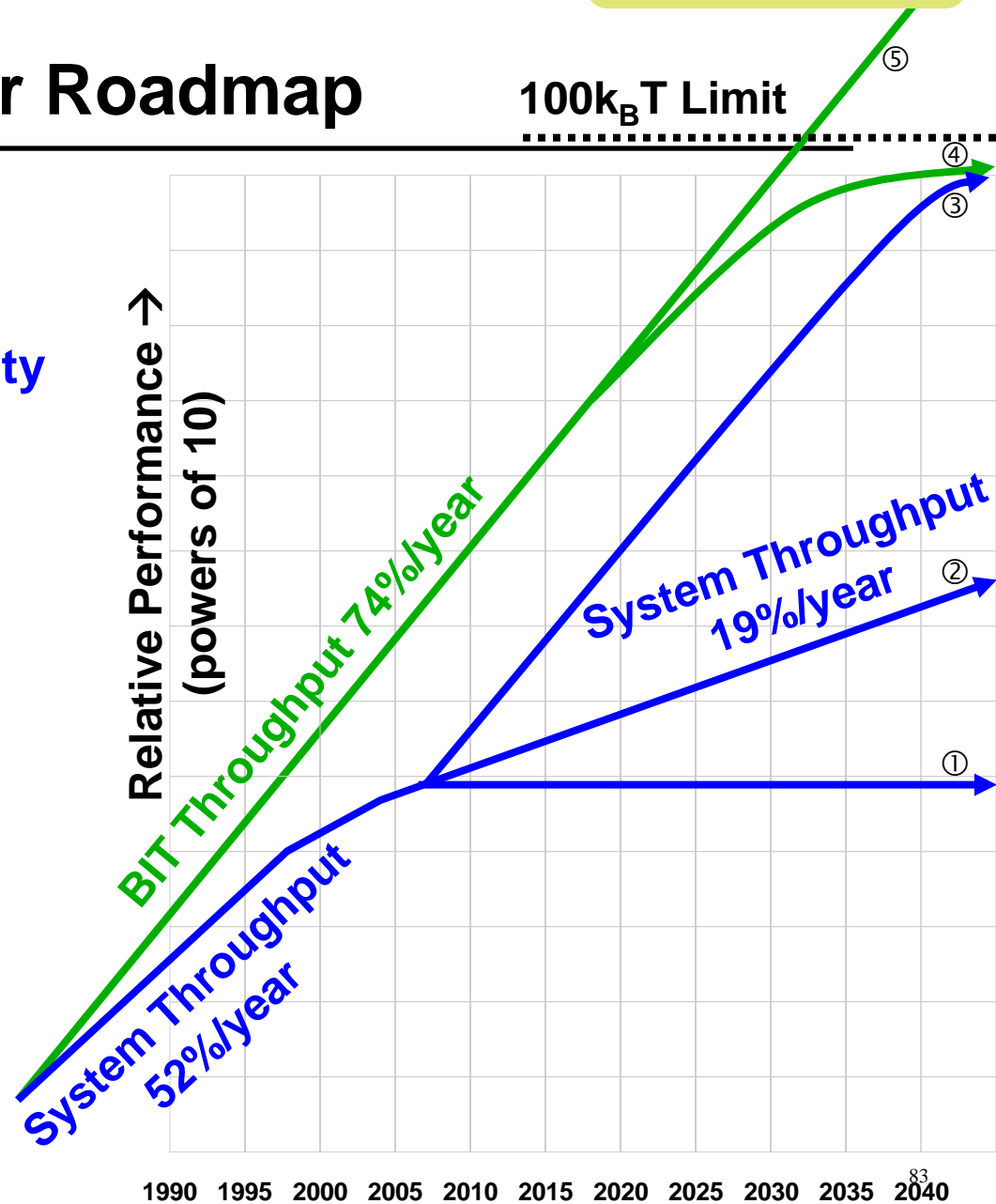


⑥ Quantum Computing  
Requires  
Rescaled Slide

# Super Roadmap

100k<sub>B</sub>T Limit

- ① Nearly flat
  - Single core, commodity
- ② Single core chip
  - C++, Fortran, etc.







# Outline

---

- **Overview**
  - Insight From a Dinner Conversation in DC
  - Super-Roadmap
- **Limitations to Moore's Law**
  - Transistor Scaling Limits per ITRS
  - Consequence to System Performance per Burger and Keckler Study
- **What It Means and What To Do About It**
  - Legacy C++/Fortran
  - **Systolic Array Lessons**
  - New Very Parallel Code
  - Special Purpose Assist
  - Analog/Neural Net
- **Over the Horizon**
  - Reversible Logic
  - Quantum Computing





# Systolic Architectures

---

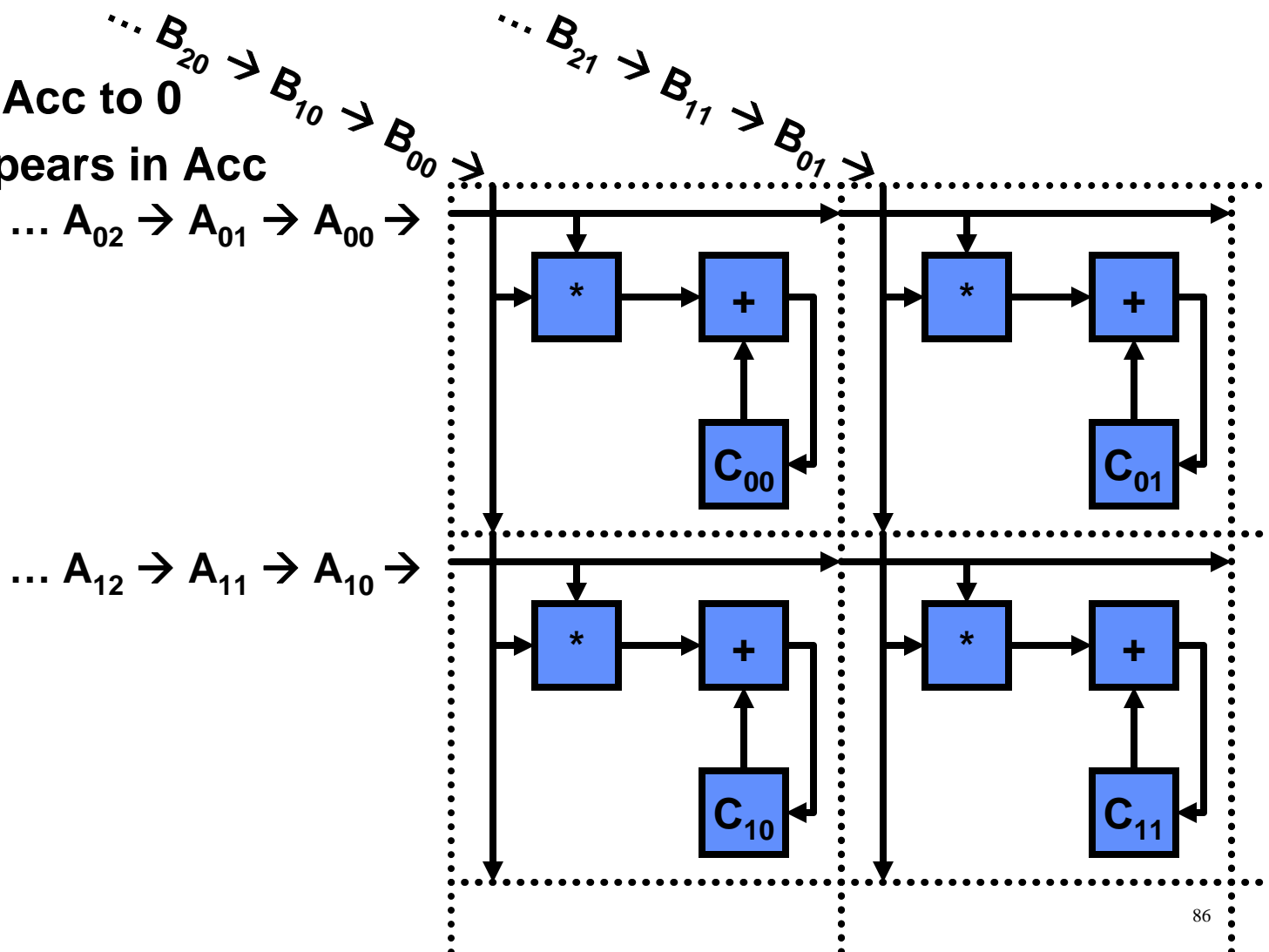
- **Overview**
  - “Special purpose hardware”
  - Efficient on all fronts
  - General, albeit not “programmed”
  - Leads to other things
- **Nodes comprise registers holding a few numbers**
- **Arcs convey numbers in lock-step communications**





# Systolic Array Matrix Multiply

- Initialize Acc to 0
- $A \times B$  appears in Acc







# Systolic Array Generality

---

- **Numerical**

- Filtering, convolution
- FFT
- Matrix-vector, matrix-matrix multiplication
- Matrix triangularization
- QR decomposition
- Linear systems solution
- Matrix inversion

- **Non-numeric**

- Searching, sorting
- Transitive closure, minimum spanning trees
- Regular expressions
- Dynamic programming
- Database operations





# Systolic Array Efficiency & Discussion

---

- The efficiency of a systolic array is just about obvious by inspection
  - The resource consuming components (space and energy) are drawn on the paper surface
  - Speed is one operation per clock
  - Not all cells are used every cycle
- Discussion
  - You get what you pay for
  - Programmer specifies data placement, data movement, and operations
  - Reward is full efficiency
  - This VLSI tool for non-complex operations, but the principles generalize (next)

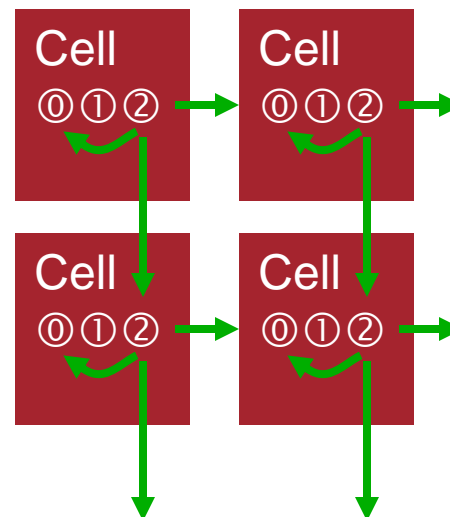




# Systolic Array Performance Model

---

- Program executes by visiting nodes  
0 1 2 0 1 2 0 1 2 ...
- Paths are regular, short, and predictable





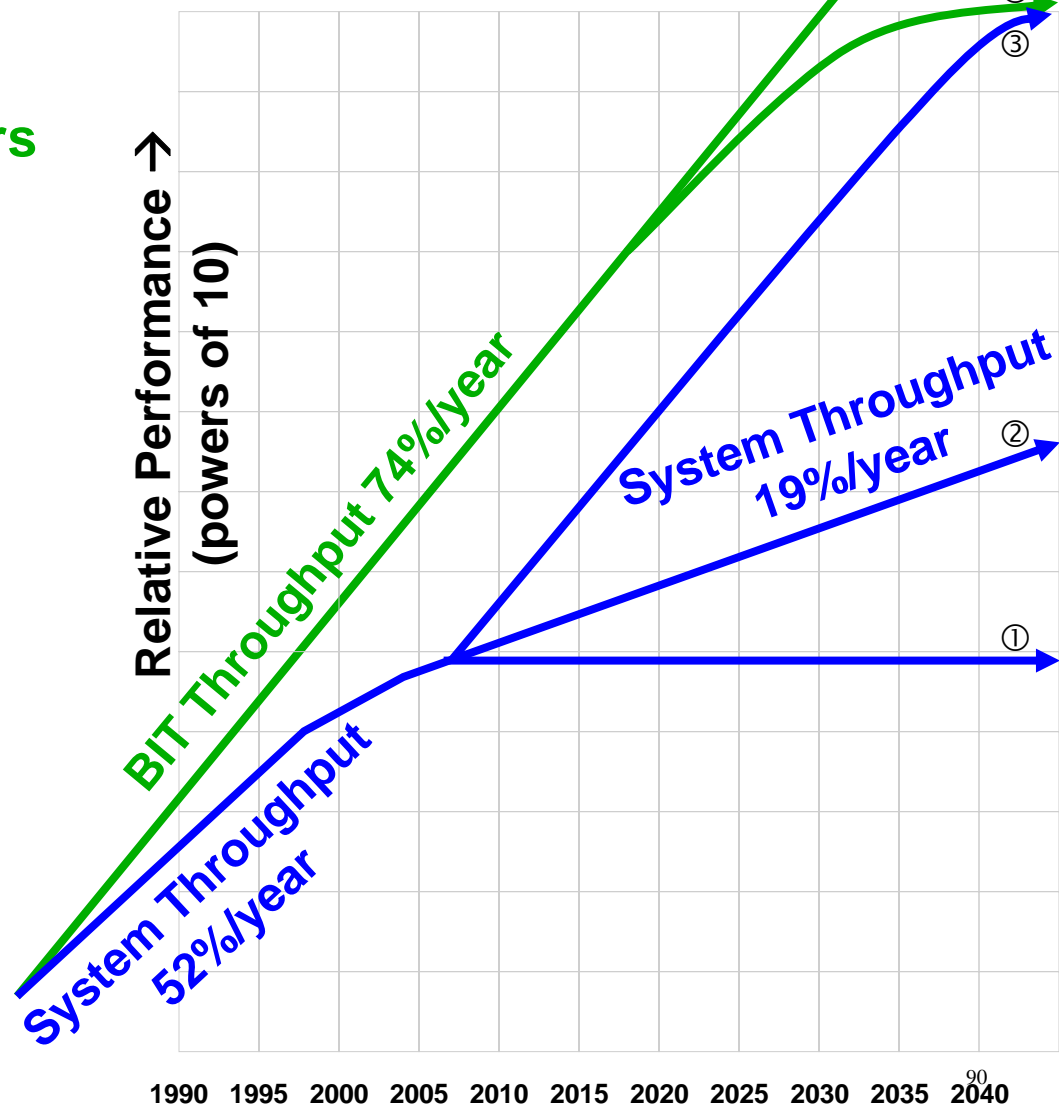


⑥ Quantum Computing  
Requires  
Rescaled Slide

# Super Roadmap

100k<sub>B</sub>T Limit

- ④ Fully exploit transistors  
– Custom hardware







# Outline

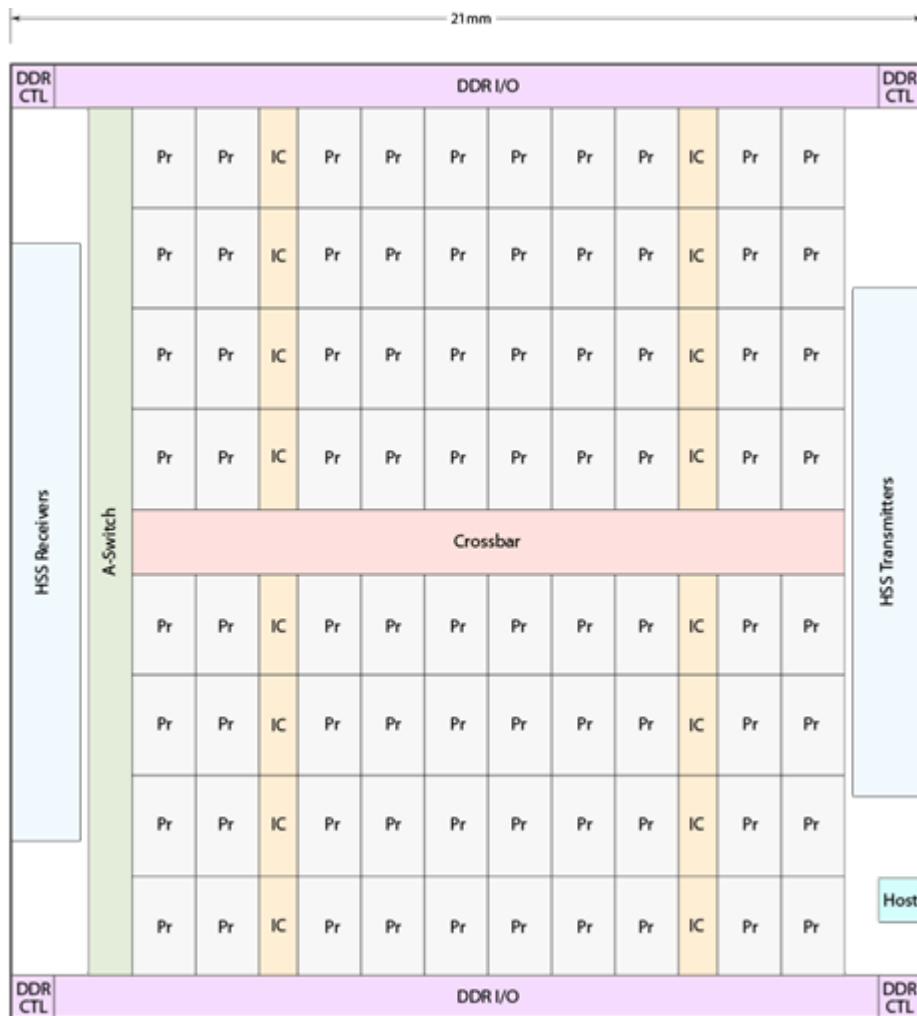
---

- **Overview**
  - Insight From a Dinner Conversation in DC
  - Super-Roadmap
- **Limitations to Moore's Law**
  - Transistor Scaling Limits per ITRS
  - Consequence to System Performance per Burger and Keckler Study
- **What It Means and What To Do About It**
  - Legacy C++/Fortran
  - Systolic Array Lessons
  - **New Very Parallel Code**
  - Special Purpose Assist
  - Analog/Neural Net
- **Over the Horizon**
  - Reversible Logic
  - Quantum Computing





# Components of Cyclops Chip



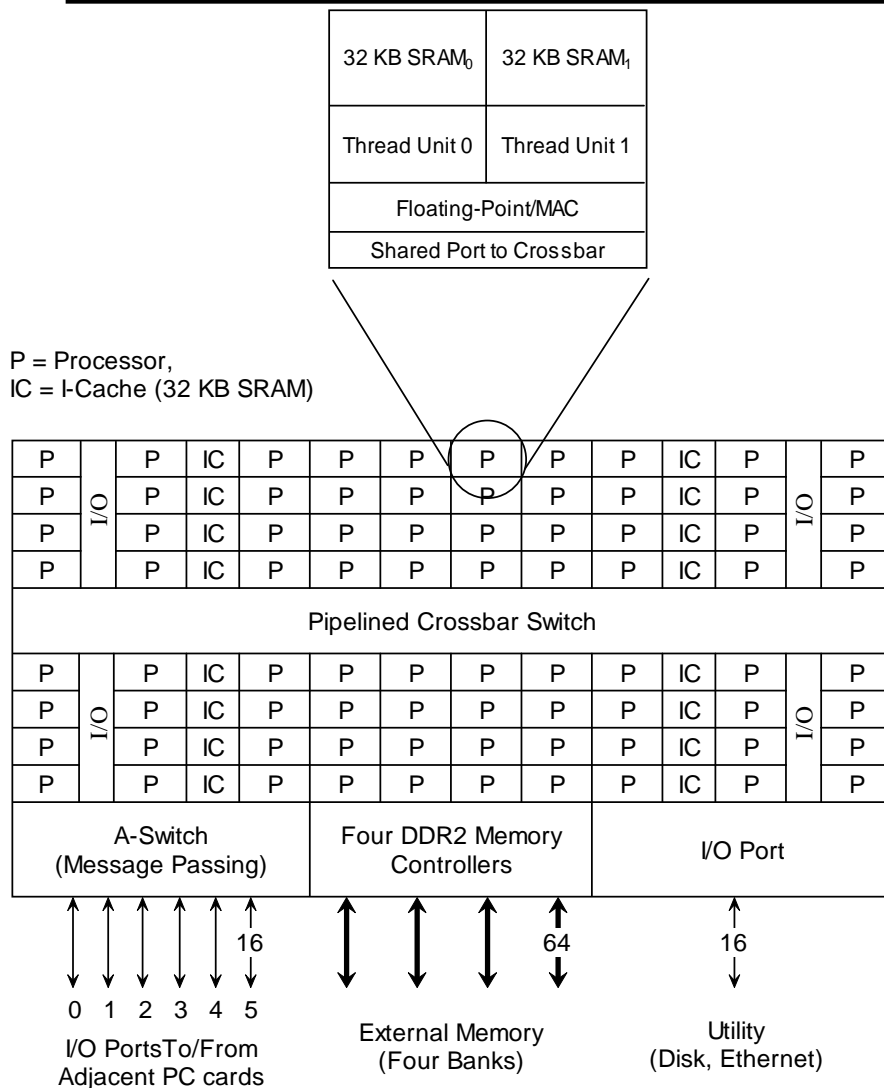
- 80 Float point processors
  - 40 KBytes scratch
- 160 Integer Processors
  - or 20 KBytes scratch
- Or on chip memory can fuse to to 3.2 MBytes
- External 1 GByte DRAM
  - 2 GBytes in a few years
- 3D Mesh Interconnect
  - 4 GBytes/sec IPC
- Disk per node I/O





# Processor Architecture

- Chip Architecture on Left
- System is 24×24×24 3D mesh

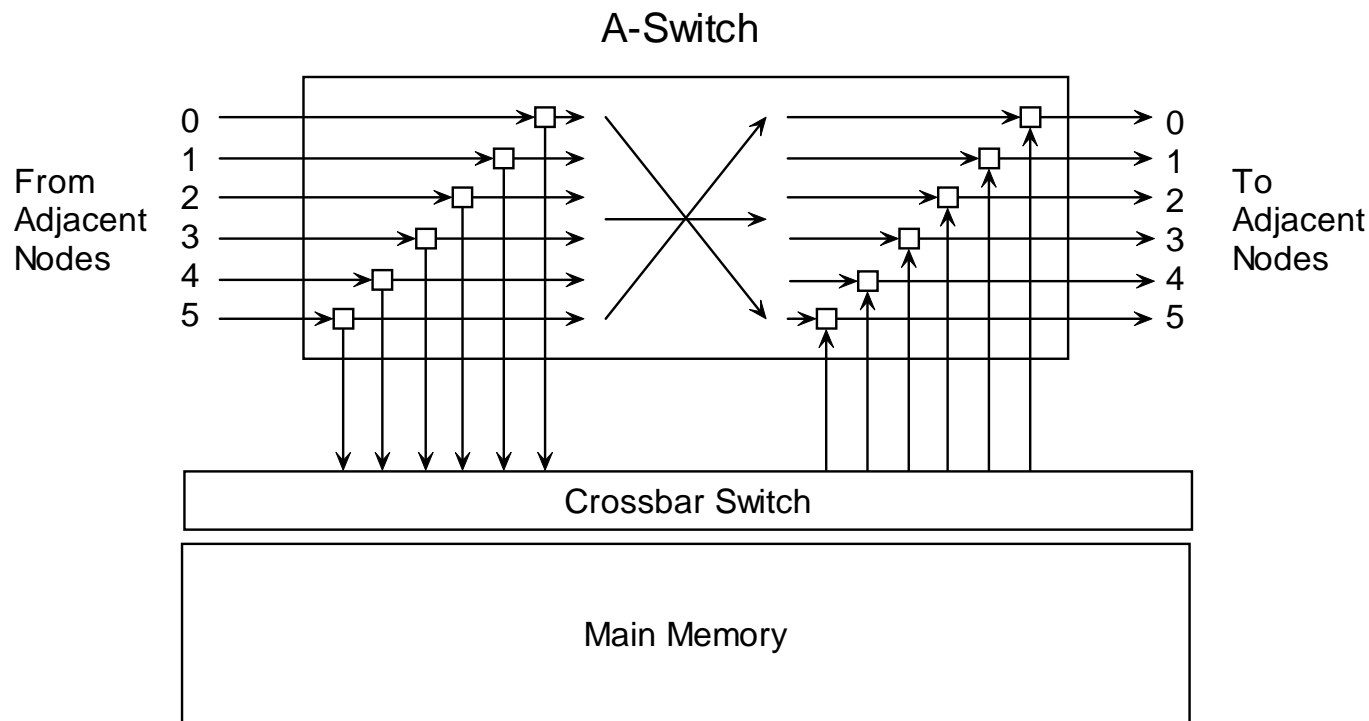






# Network

- Network is 3D mesh very much like Red Storm or Blue Gene



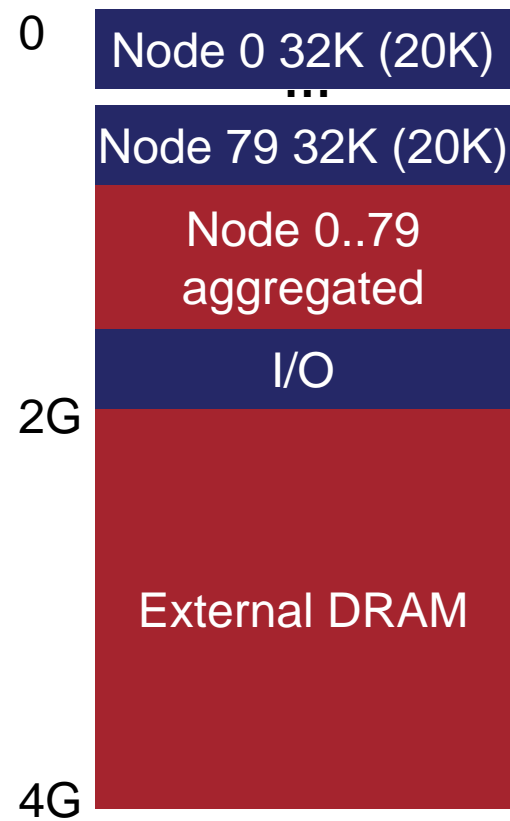




# Memory Map

---

- **Memory Hierarchy**
  - **Fastest: Your local memory (20K)**
  - **Another local node's local memory (80x20K)**
  - **On-chip aggregated memory (3.2 MB)**
  - **External memory (1 GB)**
- **User and supervisor mode**
- **Moveable barrier for aggregation**







# Cyclops Programming

---

- **Legacy Mode (my term)**
  - Run a legacy code, using internal processors and external memory, forget about on-chip memories
  - Bottleneck at external memory bus
  - Will run anything, but without advantage
- **Tuned Mode (my term)**
  - Rewrite “inner loop” to use local and aggregated on-chip memories by managing pointers
  - Use message passing, shared memory, or both
  - Run outer loop from external memory
  - Could work really well





# Cyclops Performance

- Cycle-accurate simulation of Cyclops shows promising speedup on scientific benchmarks

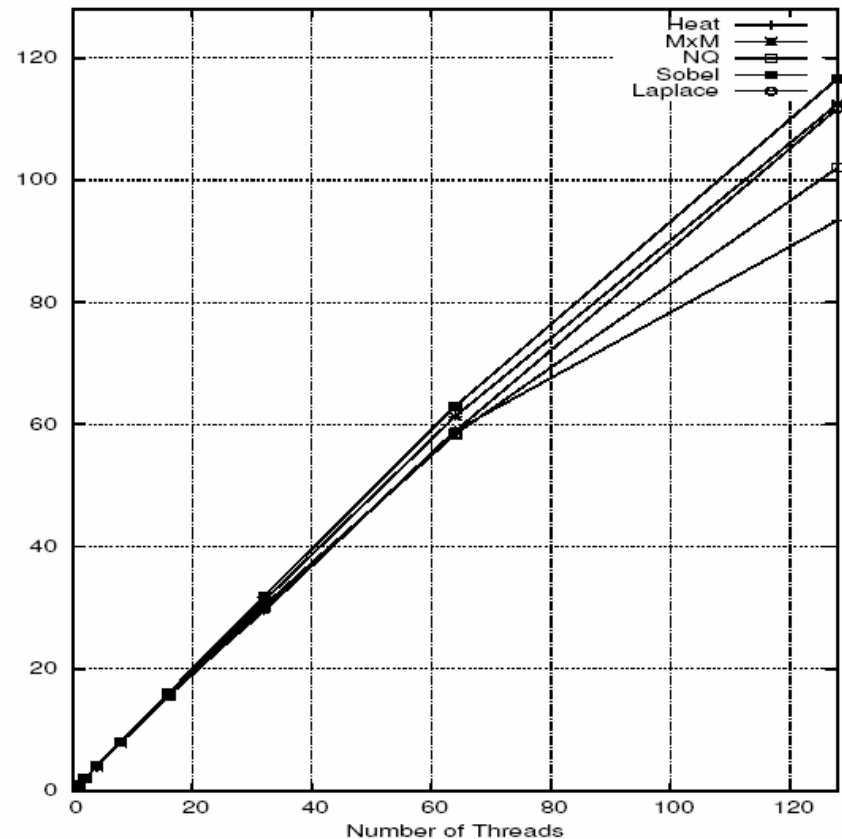
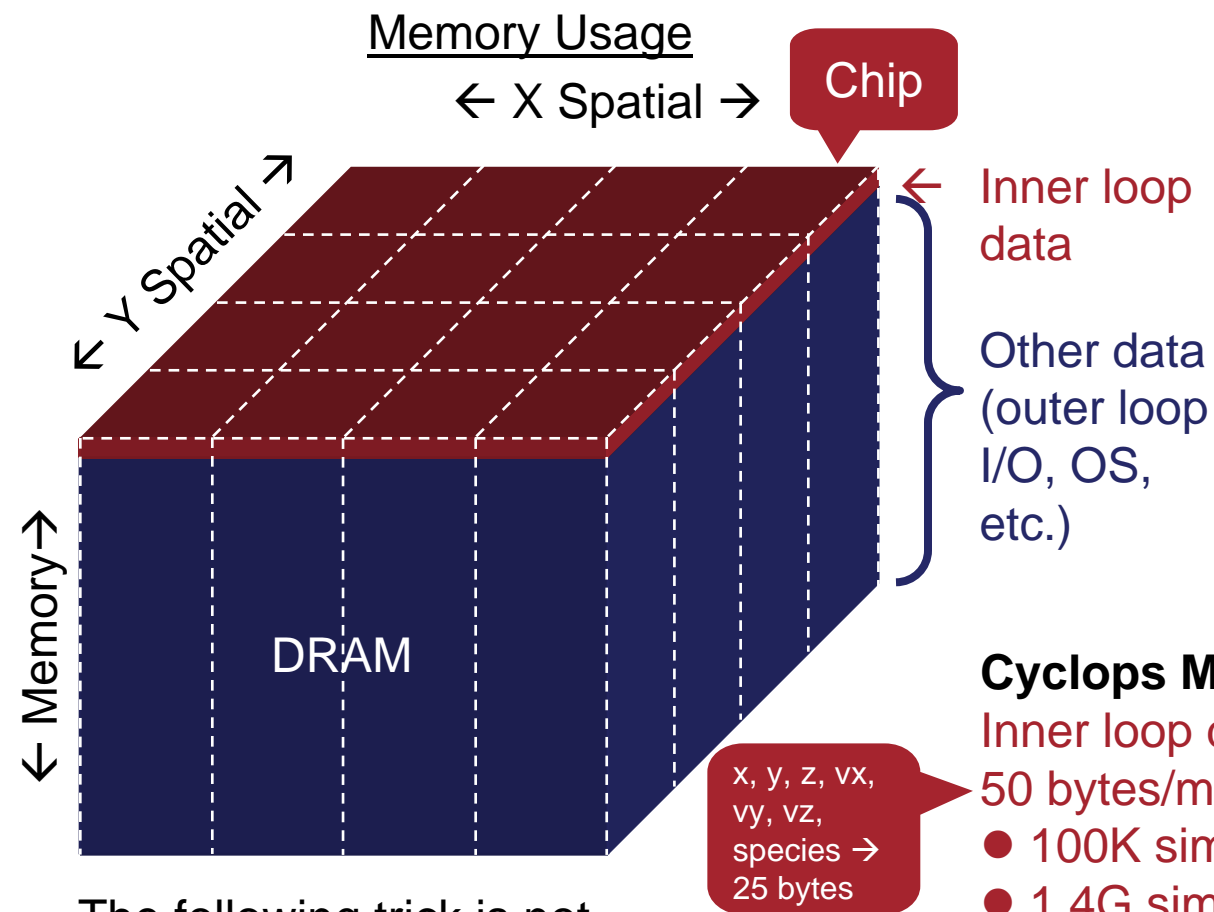


Figure 5. Assorted kernels: absolute speedup





# Cyclops Suitability Guide



The following trick is not available on Cyclops: you can't run a big problem on a small machine by adding DRAM and running longer!

## Suitability Rules:

1. Inner loop data should fit in  $80 \times 64K \approx 5.25$  MBytes/chip PIM high speed memory so inner loop runs at full speed
2. All other data goes in in per node DRAM of 1 or 2 GBytes and runs somewhat slower than a cluster – which is OK because if it is the outer loop, I/O, OS, etc.

## Cyclops Maximum DSMC Problem Size:

Inner loop data is molecular simulators at 50 bytes/molecular simulator

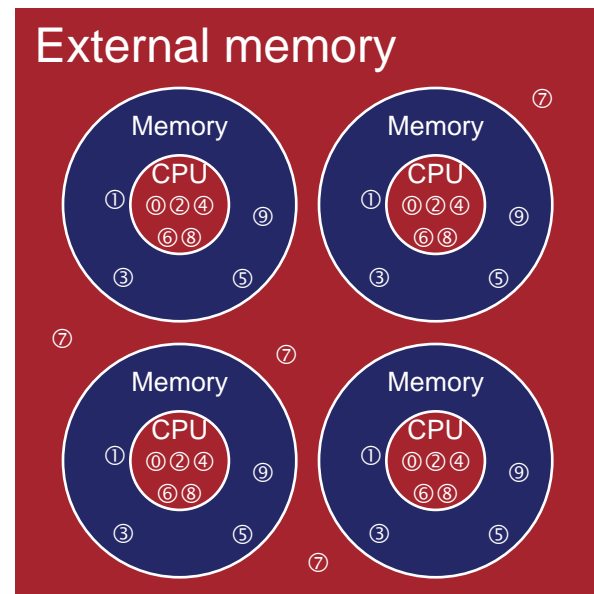
- 100K simulators/chip
- 1.4G simulators/1 Petaflops system
- 20M simulators/rack (goal is 100M simulators)





# Multi-Core Performance Model

- Compared to a single core chip, there are four threads visiting nodes rather than one
- Compared to a single core chip, the nodes are closer and the visit rate higher
- This doesn't tell you how to program your application, but tells you that if you can the machine will run fast





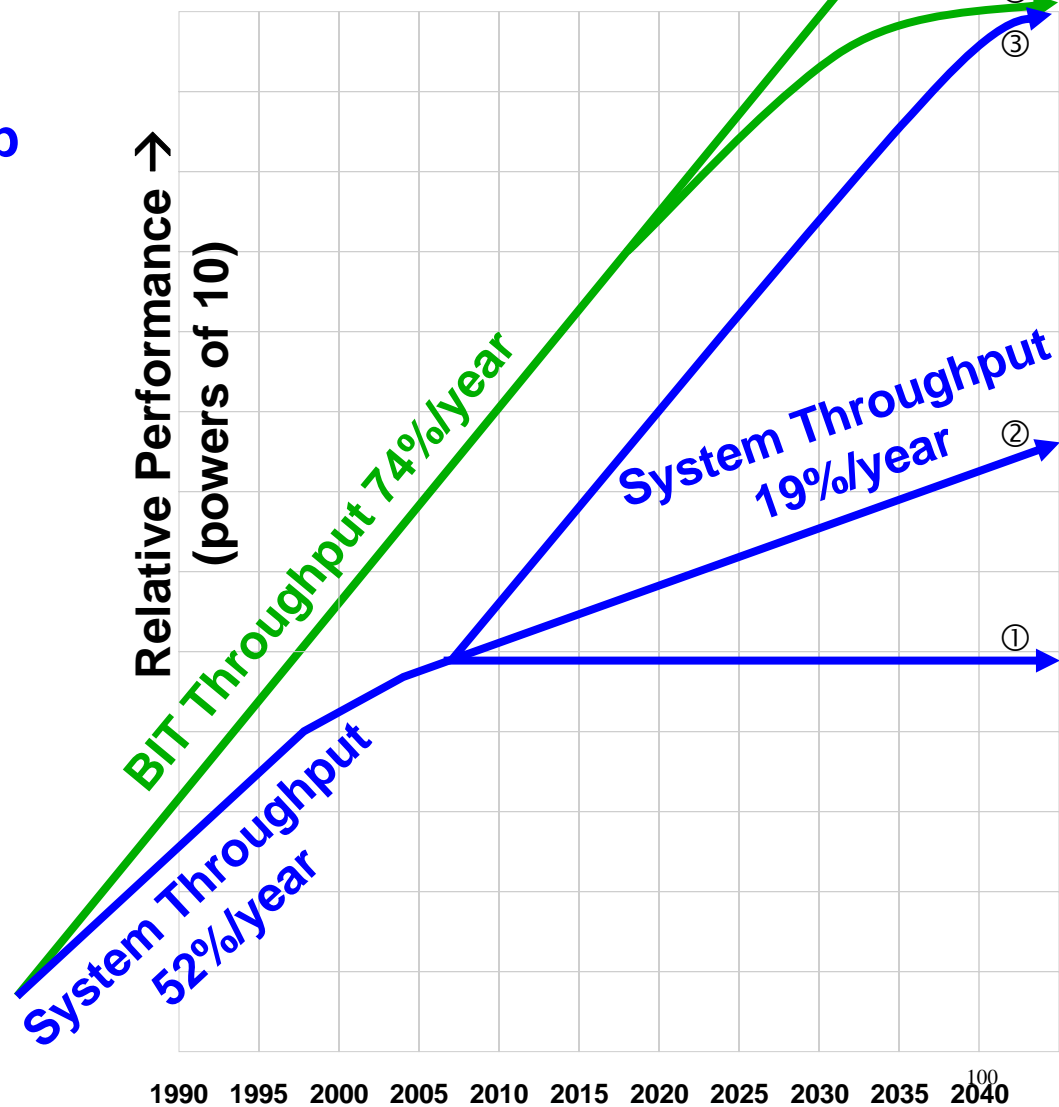


⑥ Quantum Computing  
Requires  
Rescaled Slide

# Super Roadmap

100k<sub>B</sub>T Limit

- ③ Full benefit of speedup  
– More parallel code







# Outline

---

- **Overview**
  - Insight From a Dinner Conversation in DC
  - Super-Roadmap
- **Limitations to Moore's Law**
  - Transistor Scaling Limits per ITRS
  - Consequence to System Performance per Burger and Keckler Study
- **What It Means and What To Do About It**
  - Legacy C++/Fortran
  - Systolic Array Lessons
  - New Very Parallel Code
  - **Special Purpose Assist**
  - Analog/Neural Net
- **Over the Horizon**
  - Reversible Logic
  - Quantum Computing

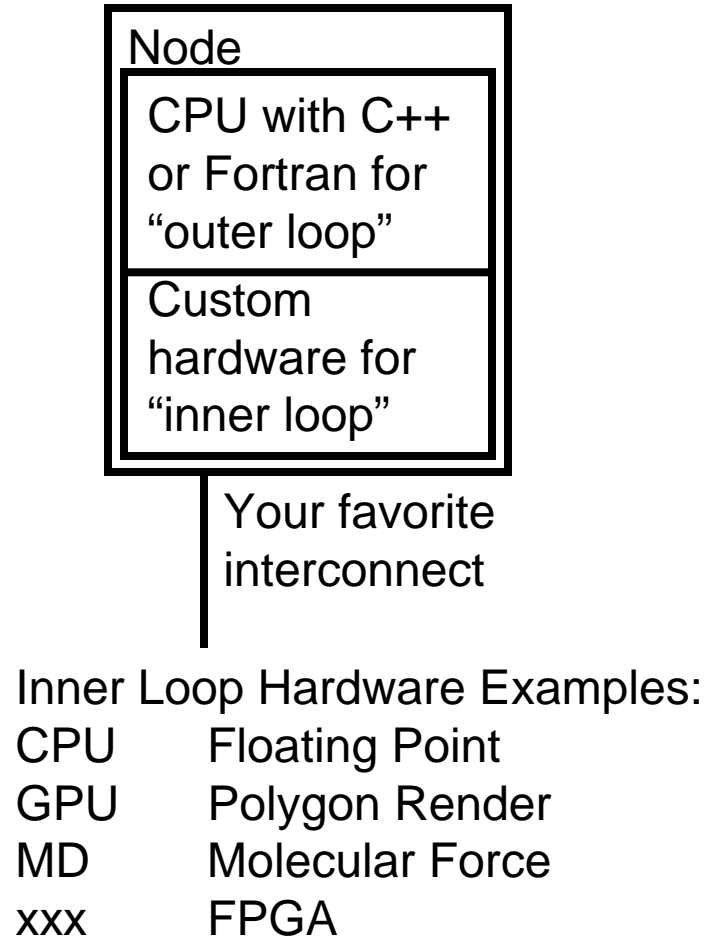




# Application-Specific Attached Processor

---

- **Idea**
  - **Develop custom hardware for main calculation in the “inner loop”**
  - **C++ or Fortran outer loop**
  - **Examples →**
- **In ideal case, runs with speed of full custom hardware with flexibility of C++ and Fortran**

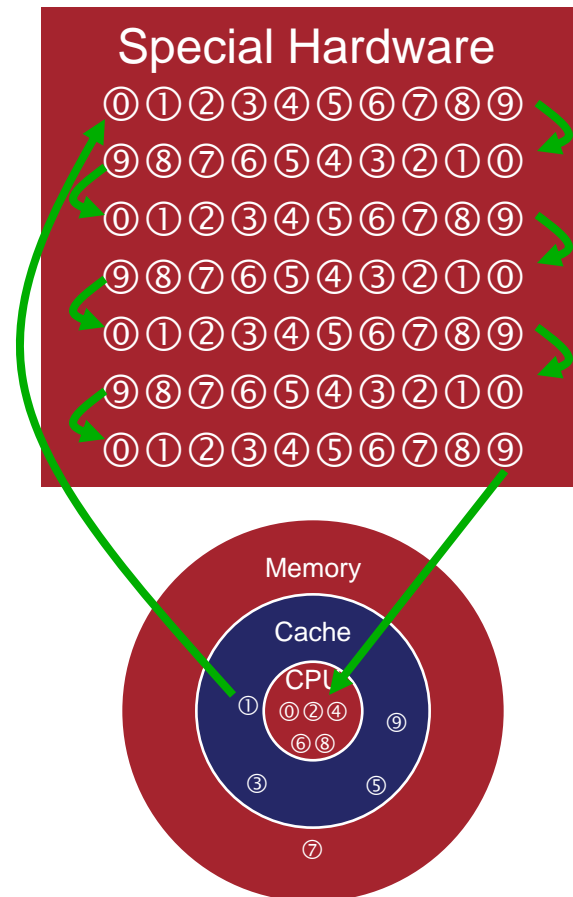






# Attached Hardware Performance Model

- Program executes by visiting nodes  
①②③④⑤⑥⑦⑧⑨...
- The special hardware is organized to execute a lot of nodes with short paths
- While a CPU exists, its contribution is diluted by the special hardware





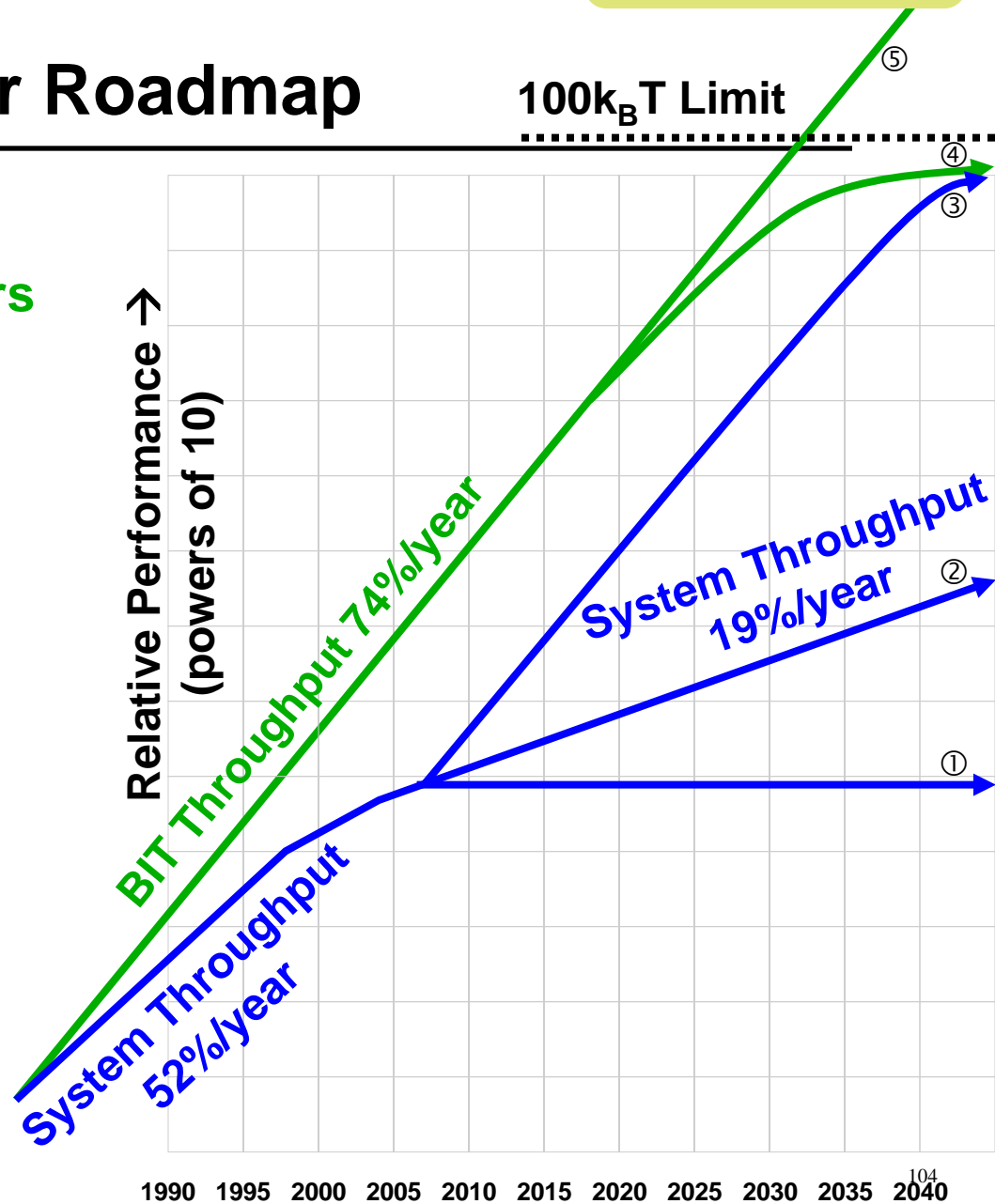


⑥ Quantum Computing  
Requires  
Rescaled Slide

# Super Roadmap

100k<sub>B</sub>T Limit

- ④ Fully exploit transistors  
– Custom hardware







# Outline

---

- **Overview**
  - **Insight From a Dinner Conversation in DC**
  - **Super-Roadmap**
- **Limitations to Moore's Law**
  - **Transistor Scaling Limits per ITRS**
  - **Consequence to System Performance per Burger and Keckler Study**
- **What It Means and What To Do About It**
  - **Legacy C++/Fortran**
  - **Systolic Array Lessons**
  - **New Very Parallel Code**
  - **Special Purpose Assist**
  - **Analog/Neural Net**
- **Over the Horizon**
  - **Reversible Logic**
  - **Quantum Computing**



# Going Beyond Moore's Law with Analog and Bio-inspired Processing

Rahul Sarpeshkar

Associate Professor

Electrical Engineering and Computer  
Science

MIT

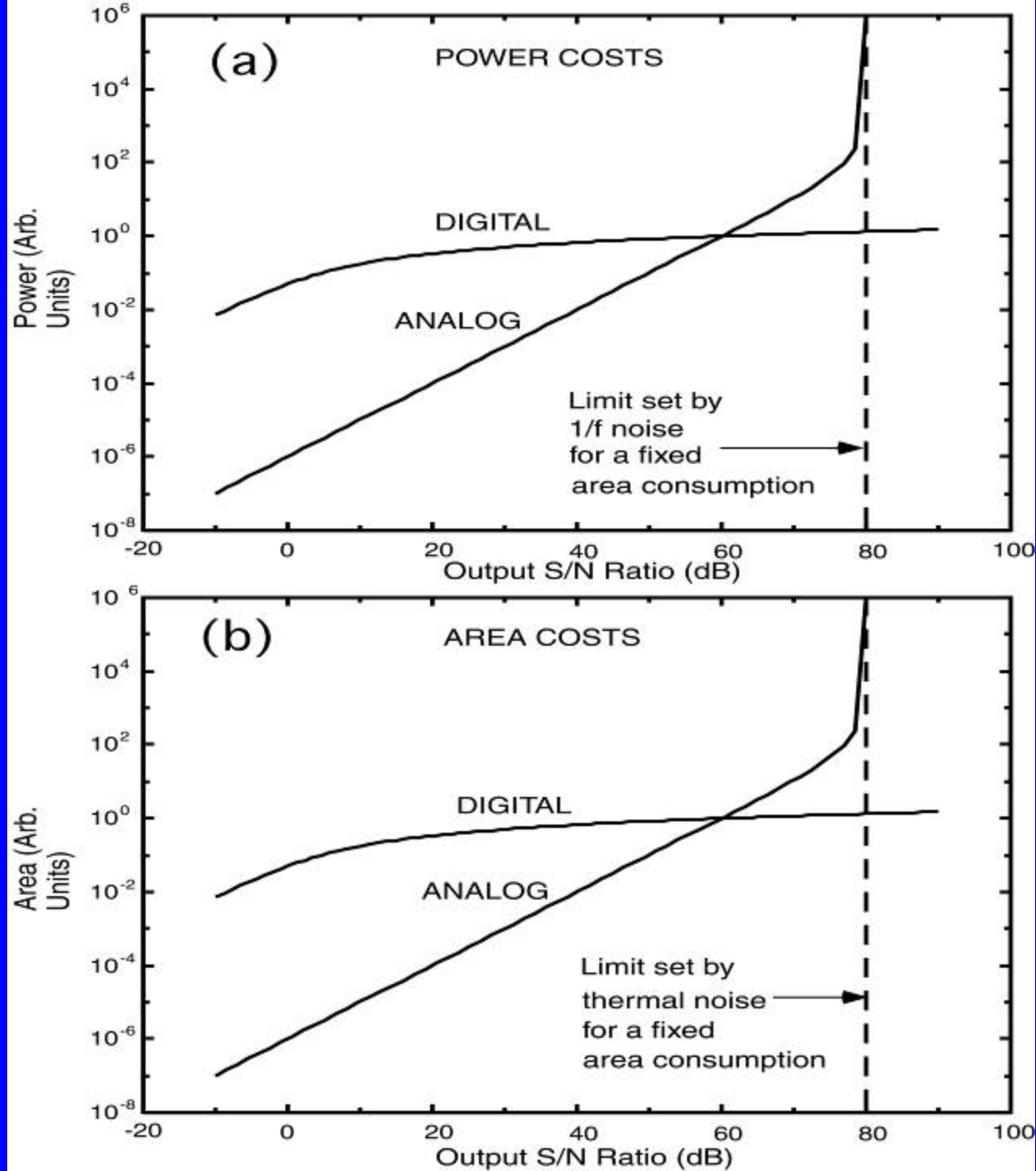
ITRS Talk  
July 9<sup>th</sup> 2006



ANALOG	DIGITAL
<ul style="list-style-type: none"> <li>○ Compute on a <b>continuous</b> set e.g. <math>\mathbb{R}</math> [0,1].</li> </ul>	<ul style="list-style-type: none"> <li>○ Compute on a <b>discrete</b> set e.g. <math>\{0,1\}</math>.</li> </ul>
<ul style="list-style-type: none"> <li>○ Primitives of computation arise from the <b>physics</b> of the computing devices: <b>Physical relations</b> of NFETs, PFETs, capacitors, resistors, floating-gate devices, KVL, KCL etc. The <b>amount of computation squeezed out of a single transistor is high</b>.</li> </ul>	<ul style="list-style-type: none"> <li>○ Primitives of computation arise from the <b>mathematics</b> of Boolean logic: <b>Logical relations</b> like AND, OR, NOT, NAND, XOR etc. The transistor is used as a switch, and the <b>amount of computation squeezed out of a single transistor is low</b>.</li> </ul>
<ul style="list-style-type: none"> <li>○ One wire represents <b>many</b> bits of information.</li> </ul>	<ul style="list-style-type: none"> <li>○ One wire represents <b>one</b> bit of information.</li> </ul>
<ul style="list-style-type: none"> <li>○ Computation is <b>offset-prone</b> since it's sensitive to the parameters of the physical devices.</li> </ul>	<ul style="list-style-type: none"> <li>○ Computation is <b>not offset-prone</b> since it's insensitive to the parameters of the physical devices.</li> </ul>
<ul style="list-style-type: none"> <li>○ Noise due to <b>thermal fluctuations</b> in physical devices.</li> </ul>	<ul style="list-style-type: none"> <li>○ Noise due to <b>roundoff error and temporal aliasing</b>.</li> </ul>
<ul style="list-style-type: none"> <li>○ Signal <b>not restored</b> at each stage of the computation.</li> </ul>	<ul style="list-style-type: none"> <li>○ Signal <b>restored</b> at each stage of the computation.</li> </ul>
<ul style="list-style-type: none"> <li>○ In a cascade of analog stages, noise starts to <b>accumulate</b> and build up.</li> </ul>	<ul style="list-style-type: none"> <li>○ Roundoff-error does <b>not accumulate</b> significantly for many computations.</li> </ul>
<ul style="list-style-type: none"> <li>○ <b>Not easily programmable.</b></li> </ul>	<ul style="list-style-type: none"> <li>○ <b>Easily programmable.</b></li> </ul>

From R. Sarpeshkar, "Analog Versus Digital: Extrapolating from Electronics to Neurobiology," Neural Computation, Vol. 10, pp. 1601-1638, 1998



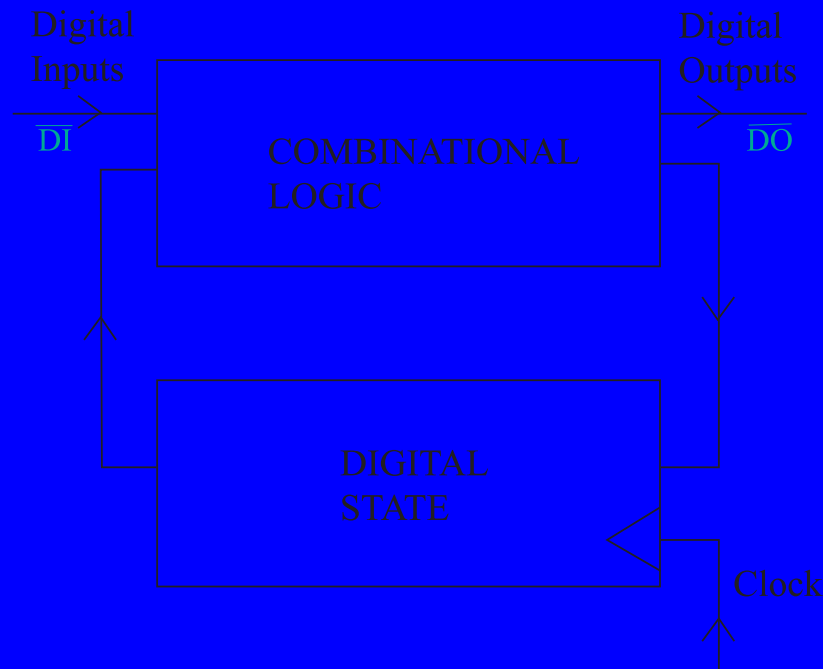


From R. Sarpeshkar, "Analog Versus Digital: Extrapolating from Electronics to Neurobiology," Neural Computation, Vol. 10, pp. 1601-1638, 1998

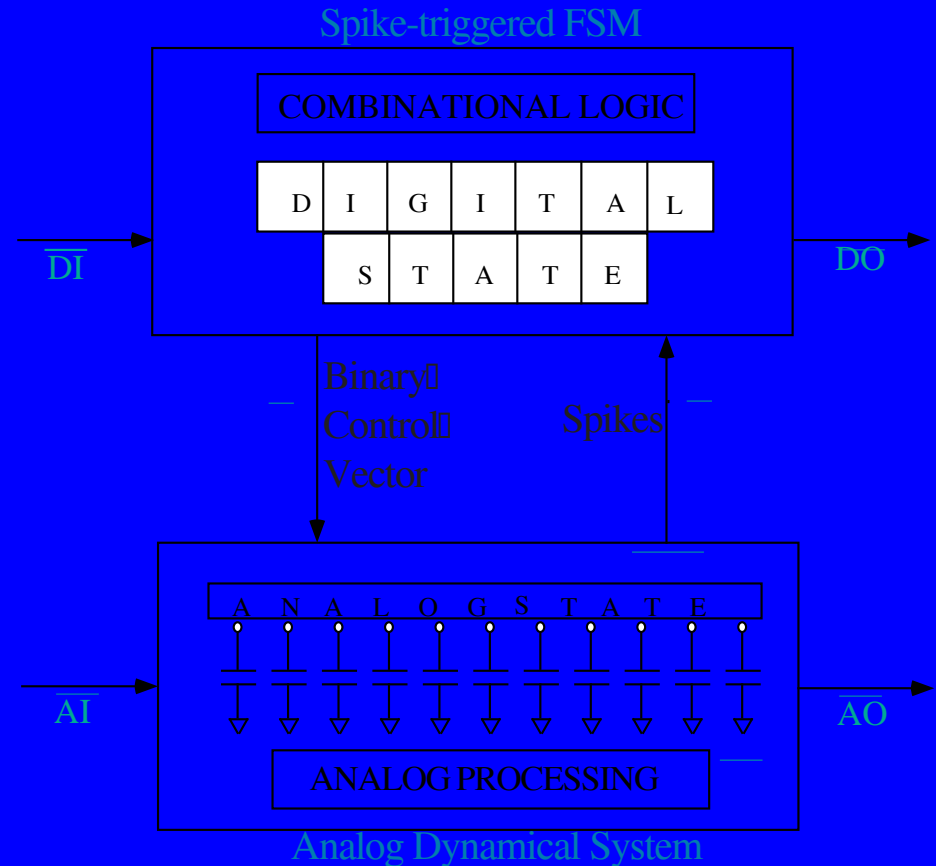


# HYBRID ANALOG-DIGITAL ARCHITECTURES

## FINITE STATE MACHINE



## HYBRID STATE MACHINE (HSM)



1. "Spike" = Pulse or Digital Event.
2. Each discrete state in the HSM is like a 'behavior' in which a rapidly reconfigurable analog dynamical system changes its parameters or topology.
3. Resulted in an extremely energy efficient time-based A/D converter with linear scaling in bit precision vs. exponential compared with other time-based converters.

From R. Sarpeshkar and M. O'Halloran, "Scalable Hybrid Computation with Spikes," Neural Computation, Vol. 14, No. 9, pp. 2003-2024, September 2002

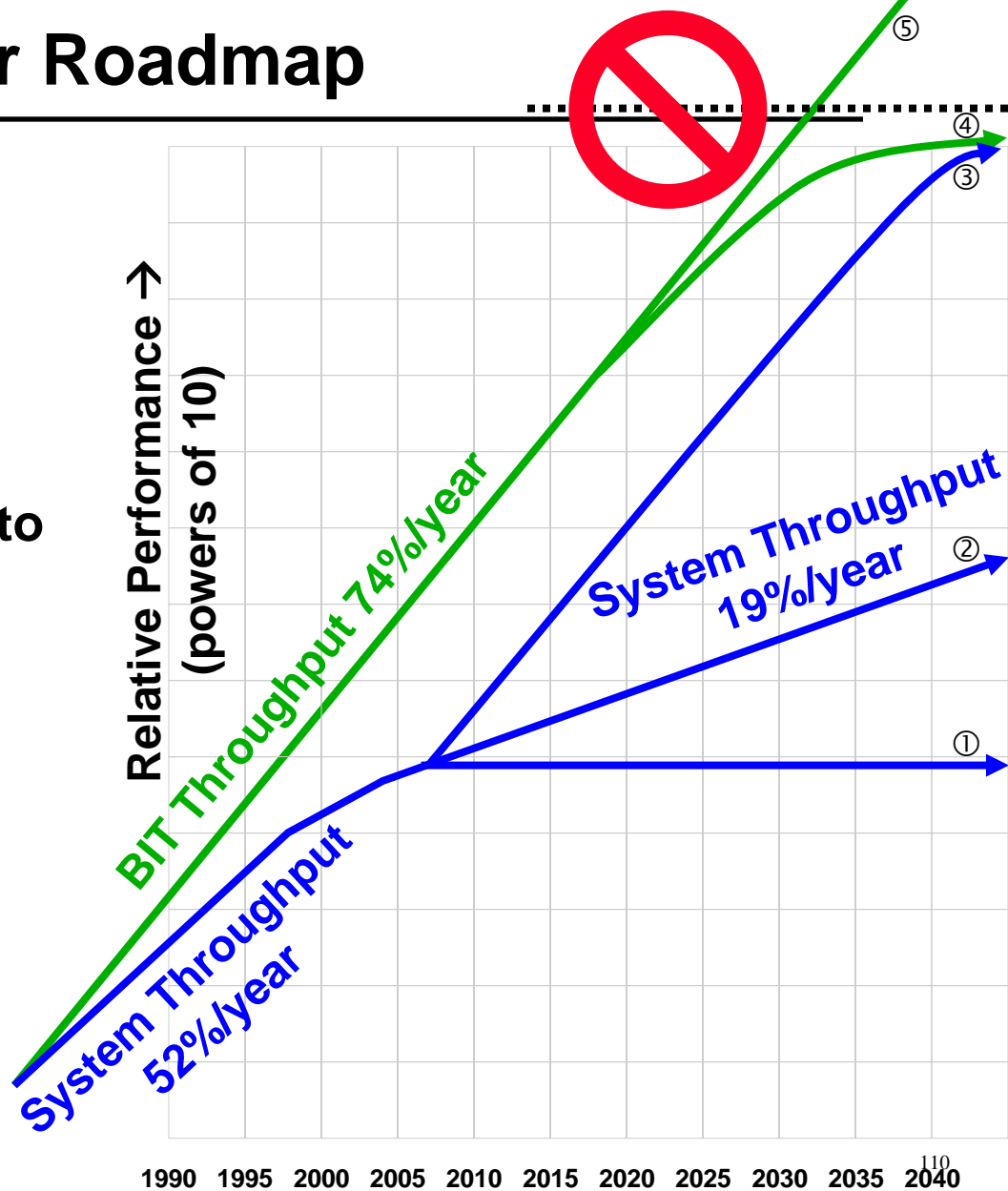




# Super Roadmap

⑥ Quantum Computing  
Requires  
Rescaled Slide

- This data point actually breaks the “super roadmap” as drawn.
- Analog computation is subject to a limit related to  $k_B T$  but the coefficient is different from digital electronics







# Outline

---

- **Overview**
  - Insight From a Dinner Conversation in DC
  - Super-Roadmap
- **Limitations to Moore's Law**
  - Transistor Scaling Limits per ITRS
  - Consequence to System Performance per Burger and Keckler Study
- **What It Means and What To Do About It**
  - Legacy C++/Fortran
  - Systolic Array Lessons
  - New Very Parallel Code
  - Special Purpose Assist
  - Analog/Neural Net
- **Over the Horizon**
  - Reversible Logic
  - Quantum Computing





# Beyond Transistors

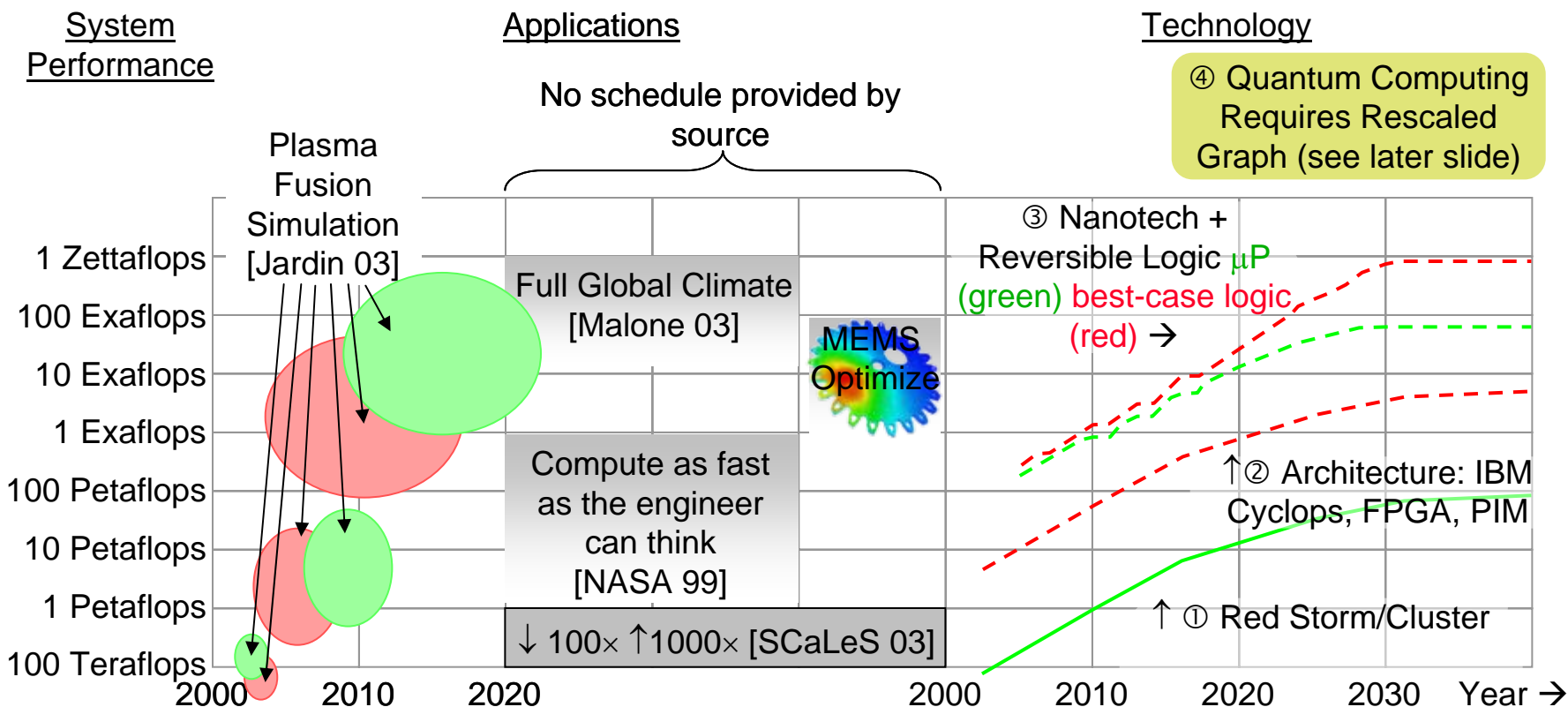
---

- **Applications Requirements**
- **Thermodynamic limits to total power**
  - Superconducting logic and Carnot cycle
- **Upside potential of advanced architectures/PIM**
- **Some nanotech technologies on the horizon**
- **Reversible logic may defeat thermodynamic limitations**
- **Upside potential of quantum computing**
  - Quantum speedup: none, quadratic, exponential
  - Algorithms numerical/cryptanalysis, simulation





# Applications and \$100M Supercomputers



[Jardin 03] S.C. Jardin, "Plasma Science Contribution to the SCaLeS Report," Princeton Plasma Physics Laboratory, PPPL-3879 UC-70, available on Internet.

[Malone 03] Robert C. Malone, John B. Drake, Philip W. Jones, Douglas A. Rotman, "High-End Computing in Climate Modeling," contribution to SCaLeS report.

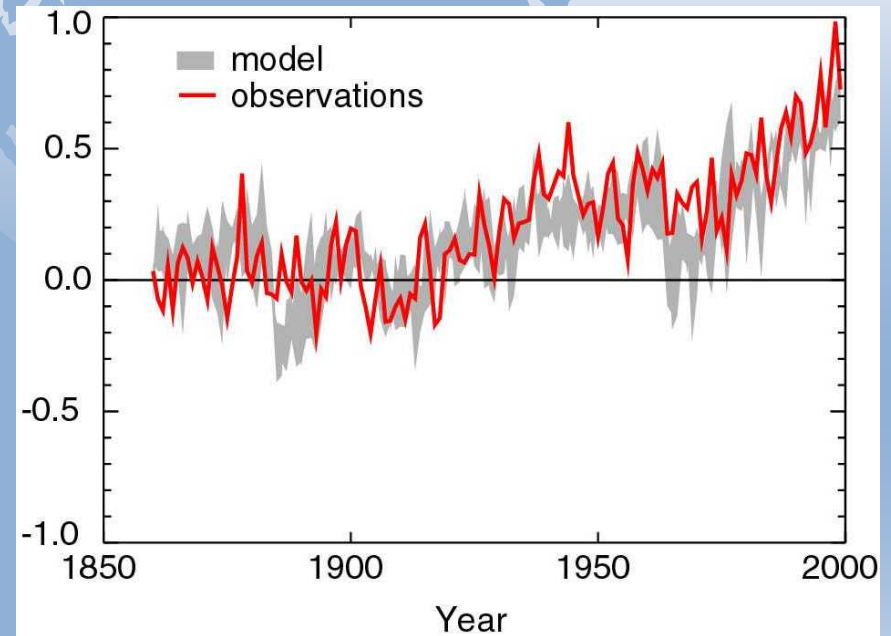
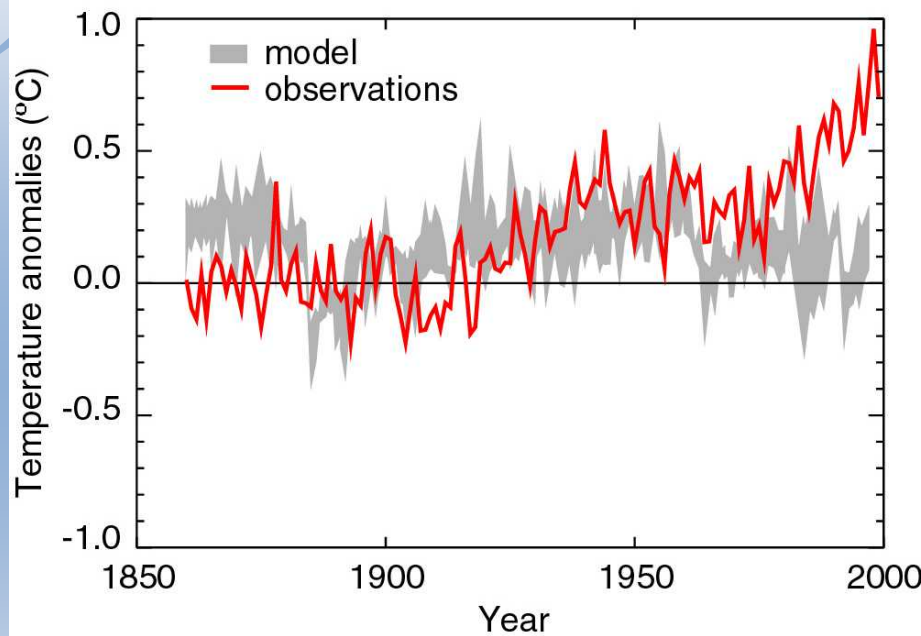
[NASA 99] R. T. Biedron, P. Mehrotra, M. L. Nelson, F. S. Preston, J. J. Rehder, J. L. Rogers, D. H. Rudy, J. Sobieski, and O. O. Storaasli, "Compute as Fast as the Engineers Can Think!" NASA/TM-1999-209715, available on Internet.

[SCaLeS 03] Workshop on the Science Case for Large-scale Simulation, June 24-25, proceedings on Internet a <http://www.pnl.gov/scales/>.

[DeBenedictis 04], Erik P. DeBenedictis, "Matching Supercomputing to Progress in Science," July 2004. Presentation at Lawrence Berkeley National Laboratory, also published as Sandia National Laboratories SAND report SAND2004-3333P. Sandia technical reports are available by going to <http://www.sandia.gov> and accessing the technical library.



# Simulation of Global Climate



**“Simulations of the response to natural forcings alone ... do not explain the warming in the second half of the century”**

Stott et al, Science 2000

**“..model estimates that take into account both greenhouse gases and sulphate aerosols are consistent with observations over this\*period” - IPCC 2001**





# FLOPS Increases for Global Climate

	Issue	Scaling
1 Zettaflops	Ensembles, scenarios 10×	Embarrassingly Parallel
100 Exaflops	Run length 100×	Longer Running Time
1 Exaflops	New parameterizations 100×	More Complex Physics
10 Petaflops	Model Completeness 100×	More Complex Physics
100 Teraflops	Spatial Resolution $10^4\times (10^3\times-10^5\times)$	Resolution
10 Gigaflops	Clusters Now In Use (100 nodes, 5% efficient)	

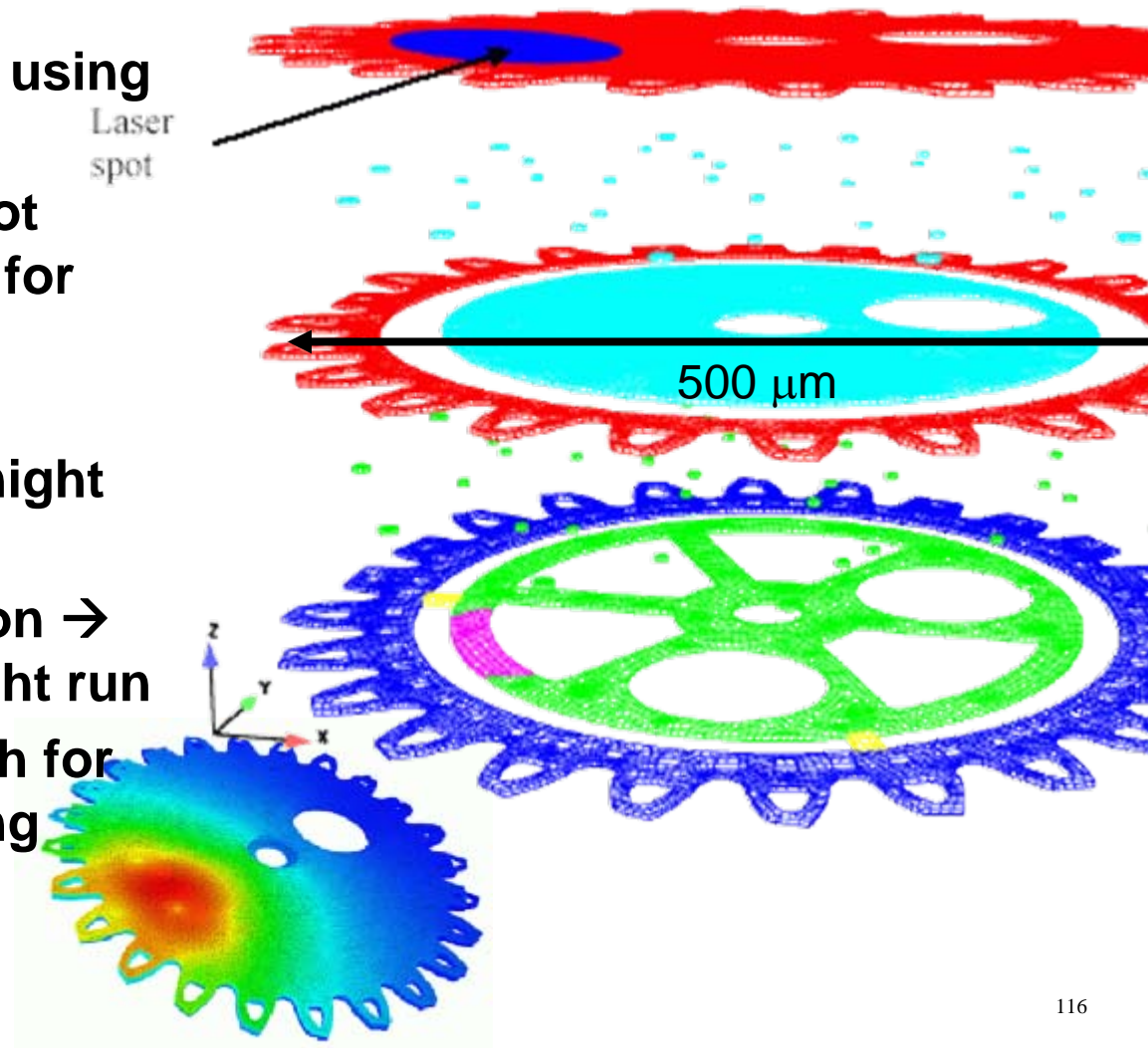
Ref. "High-End Computing in Climate Modeling," Robert C. Malone, LANL, John B. Drake, ORNL, Philip W. Jones, LANL, and Douglas A. Rotman, LLNL (2004)





# Exemplary Exa- and Zetta-Scale Simulations

- Sandia MESA facility using MEMS for weapons
- Heat flow in MEMS not diffusion; use DSMC for phonons
- Shutter needs 10 → Exaflops on an overnight run for steady state
- Geometry optimization → 100 Exaflops overnight run
  - Adjust spoke width for high b/w no melting







# FLOPS Increases for MEMS

	Issue	Scaling
100 Exaflops	Optimize 10×	Sequential
10 Exaflops	Run length 300×	Longer Running Time
30 Petaflops	Scale to $500\mu\text{m}^2 \times 12\mu\text{m}$ disk 50,000×	Size
600 Gigaflops	2D $\rightarrow$ 3D 120×	Size
5 Gigaflops	2 $\mu\text{m} \times .5\mu\text{m} \times 3\mu\text{s}$ 2D film 10 $\times$ 1.2 GHz PIII	





# Outline

---

- **Overview**
  - Insight From a Dinner Conversation in DC
  - Super-Roadmap
- **Limitations to Moore's Law**
  - Transistor Scaling Limits per ITRS
  - Consequence to System Performance per Burger and Keckler Study
- **What It Means and What To Do About It**
  - Legacy C++/Fortran
  - Systolic Array Lessons
  - New Very Parallel Code
  - Special Purpose Assist
  - Analog/Neural Net
- **Over the Horizon**
  - Reversible Logic
  - Quantum Computing





# Beyond Transistors

---

- Applications Requirements
- Thermodynamic limits to total power
  - Superconducting logic and Carnot cycle
- Upside potential of advanced architectures/PIM
- Some nanotech technologies on the horizon
- Reversible logic may defeat thermodynamic limitations
- Upside potential of quantum computing
  - Quantum speedup: none, quadratic, exponential
  - Algorithms numerical/cryptanalysis, simulation





# Beyond Transistors

---

- **Narrowing the Space**
  - We'll assume this audience is interested only in programmable digital computers
  - We'll assume this audience wants imperative programming, not AI
  - (I. e. ignore neural nets, analog computers , biochemical reactions, evolution of DNA, ...)
- **Options Within the Space**
  - **Thread Speed & Parallelism:** it looks like all paths to the future will require the programmer to expose more parallelism, but not equally
  - **Power and Heat:** Cost of electricity and danger of overheating become dominate issues





# Thermal Limit

---

- The probability of a “logic glitch” due to thermal noise is approximately  $e^{-N}$ , where  $N = E_{\text{sig}}/k_B T$
- To keep a multi Petaflops supercomputer running for several years without a glitch requires  $60 < N < 100$
- Current logic design styles thermalize all the signal energy at the output of every AND, OR, NOT gate
- Thus, it would be a reasonable “rule of thumb” that current design styles will have a hard barrier at 60-100  $k_B T$  energy per gate operation.
- ITRS predicts 30  $k_B T$ . While Erik thinks such devices might be manufacturable, redundancy in logic design should outweigh benefit
  - Also, MPF observation about information representation





# Metaphor: FM Radio on Trip to Orlando

---

- You drive to Orlando listening to FM radio
- Music clear for a while, but noise creeps in and then overtakes music
- Analogy: You live out the next dozen years buying PCs every couple years
- PCs keep getting faster
  - clock rate increases
  - fan gets bigger
  - won't go on forever
- Why...see next slide

Details: Erik DeBenedictis, "Taking ASCI Supercomputing to the End Game," SAND2004-0959

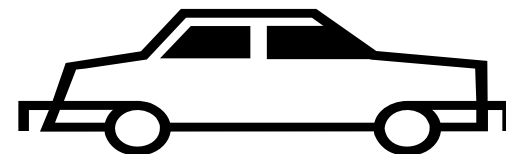
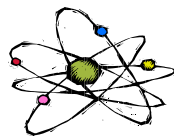




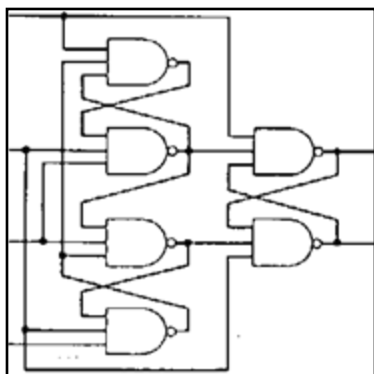
# FM Radio and End of Moore's Law



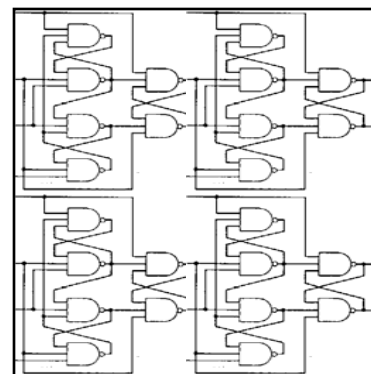
Distance



Driving away from FM transmitter → less signal  
Noise from electrons → no change



Shrink



Increasing numbers of gates → less signal power  
Noise from electrons → no change





## **Personal Observational Evidence**

---

- **Have radios become better able to receive distant terrestrial stations over the last few decades with a rate of improvement similar to Moore's Law?**
  - **XM is a different story**
- **You judge from your experience, but the answer should be that they have not.**
- **Therefore, electrical noise does not scale with Moore's Law.**





# Landauer's Arguments

---

- Landauer makes three arguments in his 1961 paper
  - Kinetics of a bistable well (next slide)
  - Entropy generation →

Sorry, I don't have a cute story (like the FM radio) for Landauer's argument

- Entropy of a system in statistical mechanics:

$$S = k_B \log_e(W)$$

$W$  is number of states

- Entropy of a mechanical system containing a flip flop in an unknown state:

$$S = k_B \log_e(2W)$$

- After clearing the flip flop:

$$S = k_B \log_e(W)$$

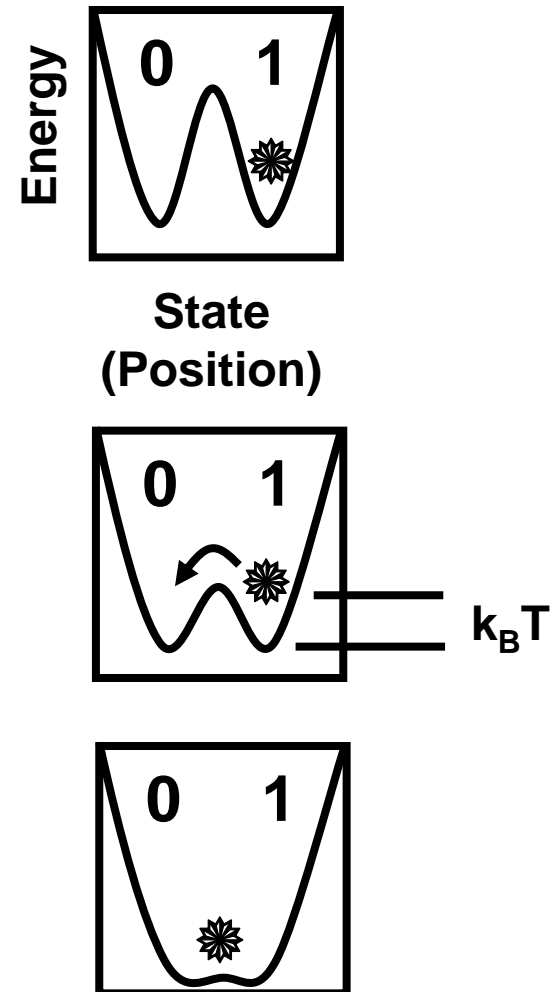
- Difference  $k_B \log_e(2)$





# Landauer's Limit

- The Landauer limit says you can reduce power dissipation for irreversible functions below  $100 k_B T$ , but not below  $k_B T \log_e 2$
- In the diagram on the right, when the energy barrier drops to below about  $k_B T$ , the state will spontaneously switch and dissipate remaining energy as heat







# Beyond Transistors

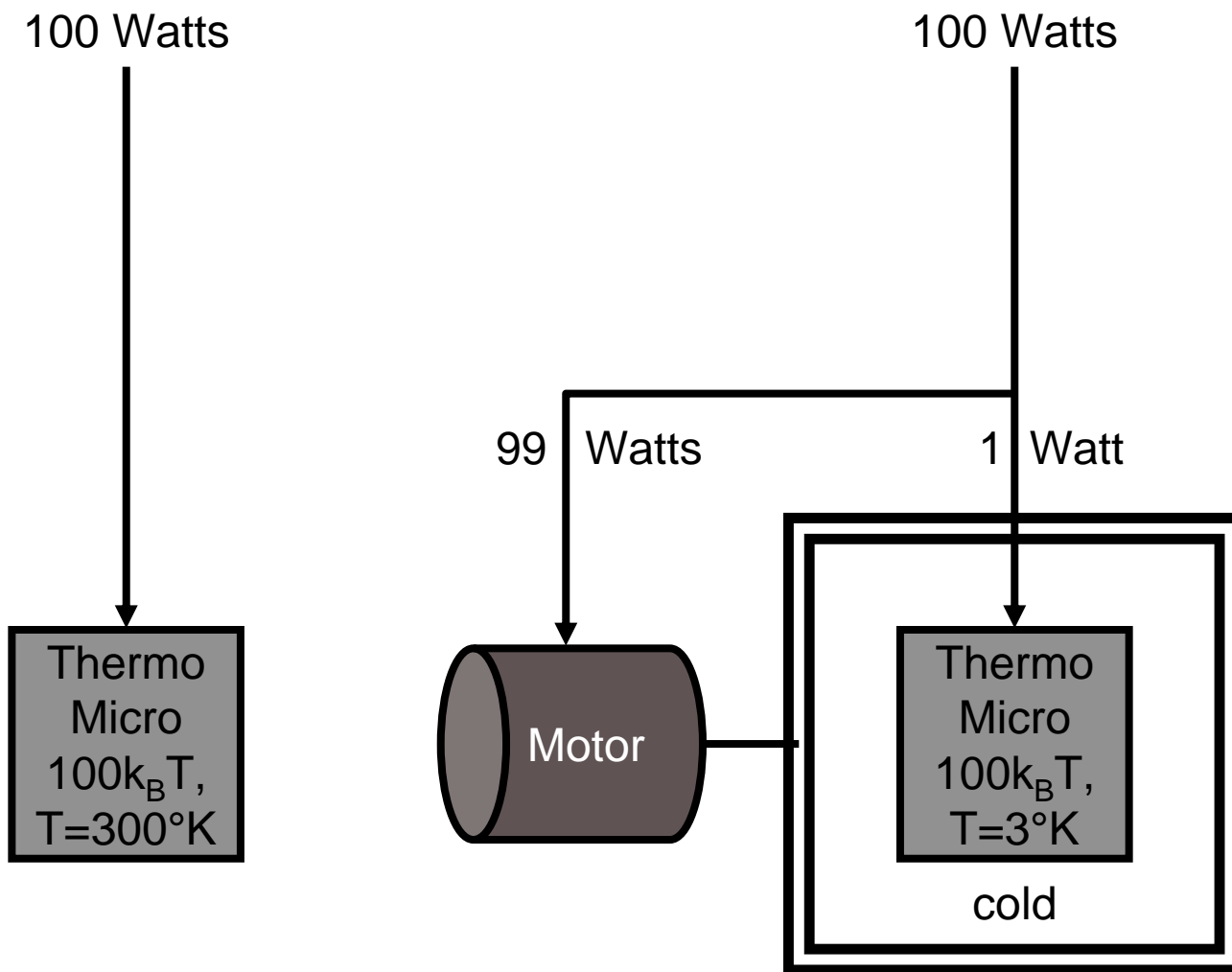
---

- Applications Requirements
- Thermodynamic limits to total power
  - Superconducting logic and Carnot cycle
- Upside potential of advanced architectures/PIM
- Some nanotech technologies on the horizon
- Reversible logic may defeat thermodynamic limitations
- Upside potential of quantum computing
  - Quantum speedup: none, quadratic, exponential
  - Algorithms numerical/cryptanalysis, simulation





# Cutting Temperature







# Cutting Temperature

---

$$\text{Carnot Efficiency } \eta_c = \frac{T_c}{T_h - T_c}$$

$$\text{Specific Power } 1/\eta_c = \frac{T_h - T_c}{T_c}$$

Specific power is watts input power required to remove one watt at the cooling temperature

Idea:

To cut computer power, let's cool the active devices to 3° K. This will cut minimum power per reliable operation from  $100k_B \times 300$  to  $100k_B \times 3$ , cutting device power by 100 fold!

$$\begin{aligned} \text{Specific Power } 1/\eta_c &= \frac{T_h - T_c}{T_c} \\ &= \frac{300 - 3}{3} \\ &= 99 \end{aligned}$$

Thus, we cut device power to 1% of original power at the price of a refrigerator consuming 99% of the original power, for resulting total power consumption of 100% of original power.

However, refrigerators are typically <20% efficient, so we're actually in the hole by 5× ...  
but it is cheaper to dissipate power in a big motor than an expensive chip.





# Beyond Transistors

---

- Applications Requirements
- Thermodynamic limits to total power
  - Superconducting logic and Carnot cycle
- Upside potential of advanced architectures/PIM
- Some nanotech technologies on the horizon
- Reversible logic may defeat thermodynamic limitations
- Upside potential of quantum computing
  - Quantum speedup: none, quadratic, exponential
  - Algorithms numerical/cryptanalysis, simulation





# Scientific Supercomputer Limits

Best-Case  
Logic

Microprocessor  
Architecture

Physical  
Factor

Source of  
Authority

$2 \times 10^{24}$  logic ops/s

Expert  
Opinion

100 Exaflops	800 Petaflops
← 125:1 →	

Estimate	25 Exaflops	200 Petaflops
	4 Exaflops	32 Petaflops

1 Exaflops	8 Petaflops
------------	-------------

Assumption: Supercomputer is size & cost of Red Storm: US\$100M budget; consumes 2 MW wall power; 750 KW to active components

80 Teraflops

40 Teraflops

Reliability limit  
 $750\text{KW}/(80k_B T)$

Esteemed physicists  
( $T=60^\circ\text{C}$  junction temperature)

Derate 20,000 convert  
logic ops to floating point

Floating point engineering  
(64 bit precision)

Derate for manufacturing  
margin (4×)

Estimate

Uncertainty (6×)

Gap in chart

Improved devices (4×)

Estimate

Projected ITRS  
improvement to 22 nm  
(100×)

ITRS committee of experts

Lower supply voltage  
(2×)

ITRS committee of experts

Red Storm

contract





# Beyond Transistors

---

- Applications Requirements
- Thermodynamic limits to total power
  - Superconducting logic and Carnot cycle
- Upside potential of advanced architectures/PIM
- Some nanotech technologies on the horizon
- Reversible logic may defeat thermodynamic limitations
- Upside potential of quantum computing
  - Quantum speedup: none, quadratic, exponential
  - Algorithms numerical/cryptanalysis, simulation





# Transistors vs. Other Irreversible Devices

---

- **Erik's View**

- My contacts on the ITRS staff tell me they believe transistors will get to the  $\sim 30 k_B T$  level. If this is so, transistors will be difficult to beat in this domain.
- At  $30 k_B T$ , logic would have a spontaneous error rate  $> e^{-30}$  (one error in a billion operations).
- I have no doubt that computing with a  $10^{-9}$  error rate is possible, but the overhead in error correction would consume more than a factor of 3. Remember Triple Modular Redundancy (TMR) consumes  $3\times$  hardware!





# Really Advanced Technology

- ITRS ERD [see below]
  - Influential over industrial and government funding
- International Technology Roadmap for Semiconductors (ITRS) Emerging Research Devices (ERD) architecture panel. All new devices are inadequate except CNFET

<div> <div>&gt; 20</div> <div>&gt; 16 - 18</div> <div>&gt; 18 - 20</div> <div>≤ 16</div> </div> <div>For each Technology Entry (e.g. 1D Structures, sum horizontally over the 8 Criteria) Max Sum = 24 Min Sum = 8</div>								
Evaluation of Emerging Research Logic Device Technologies against Technology Evaluation Criteria								
Logic Device Technologies	Scalability	Performance	Energy Efficiency	Gain	Operational Reliability	Room Temp. Operation ***	CMOS Compatibility **	CMOS Architectural Compatibility *
1D Structures	2.4	2.4	2.1	2.4	2.3	2.9	2.4	2.6
Resonant Tunneling Devices	1.4	2.0	1.9	1.7	1.7	2.9	2.1	2.1
SETs	1.9	1.0	2.5	1.3	1.2	1.9	2.4	2.0
Molecular Devices	1.9	1.1	2.0	1.1	1.3	2.6	1.9	1.6
Ferromagnetic Devices	1.5	1.2	1.8	1.5	1.8	2.2	1.5	1.8
Spin Transistor	1.7	1.7	2.2	1.5	2.0	2.2	1.7	1.8



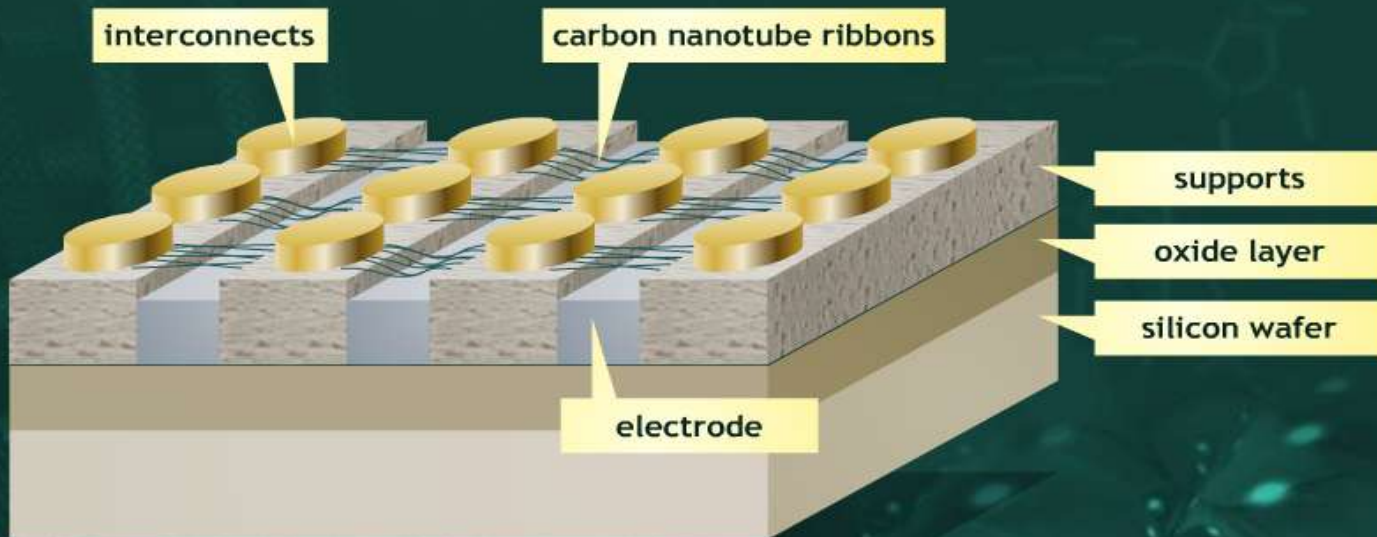


# ITRS Device Review 2016 + QDCA

Technology	Speed (min-max)	Dimension (min-max)	Energy per gate-op	Comparison
CMOS	30 ps-1 $\mu$ s	8 nm-5 $\mu$ m	4 aJ	
RSFQ	1 ps-50 ps	300 nm- 1 $\mu$ m	2 aJ	Larger
Molecular	10 ns-1 ms	1 nm- 5 nm	10 zJ	Slower
Plastic	100 $\mu$ s-1 ms	100 $\mu$ m-1 mm	4 aJ	Larger+Slower
Optical	100 as-1 ps	200 nm-2 $\mu$ m	1 pJ	Larger+Hotter
NEMS	100 ns-1 ms	10-100 nm	1 zJ	Slower+Larger
Biological	100 fs-100 $\mu$ s	6-50 $\mu$ m	.3 yJ	Slower+Larger
Quantum	100 as-1 fs	10-100 nm	1 zJ	Larger
QDCA	100 fs-10ps	1-10 nm	1 yJ	Smaller, faster, cooler

Data from ITRS ERD Section, data from Notre Dame





Subject: RE: May I show some Nantero Viewgraphs?

From: Suzanne <suzpr45@earthlink.net>

Date: 3:05 PM

To: 'Erik P. DeBenedictis' <epdeben@sandia.gov>

Hi Erik,

You may certainly have permission from Nantero. T

Best,

Suzanne

Suzanne Gibbons-Neff

CCNY Public Relations & Marketing

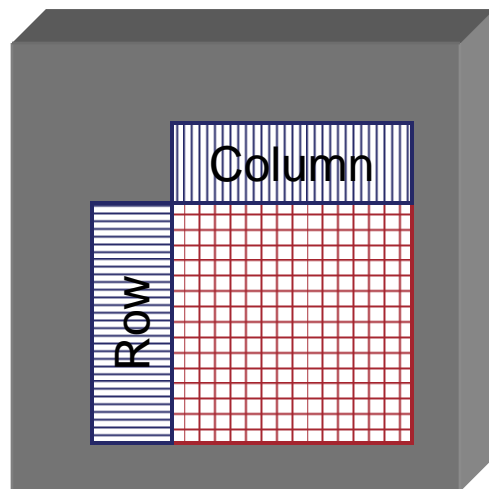




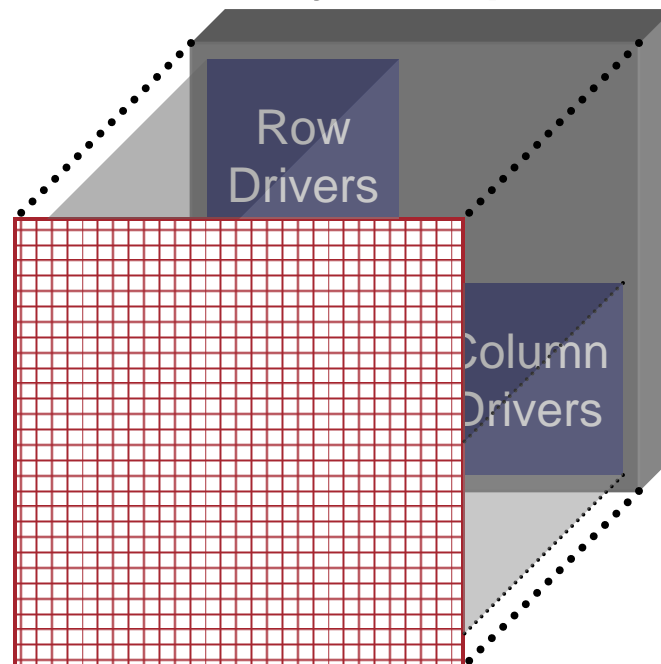
# Nanoarray Architecture

---

- Low Road
  - Planar, conventional architecture



- High Road
  - Fabricate nanotech array on top of chip

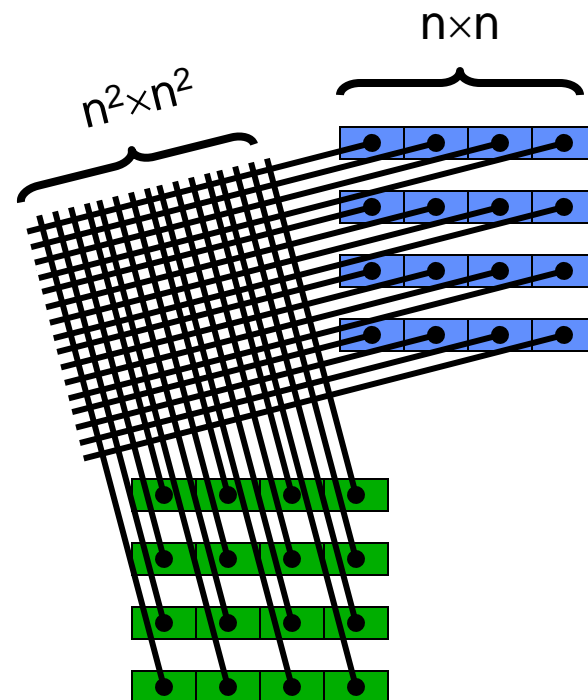






# Thought Experiment – Skewed Nanoarray

- Problem is that molecular scale mask alignment is very hard
- However, regular arrays of lines are more easily drawn →
- Diagram to right (from Likharev) uses  $2n^2$  drivers to drive  $n^4$  crosspoints



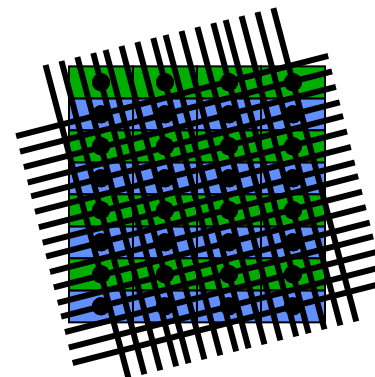




# Thought Experiment – Skewed Nanoarray

---

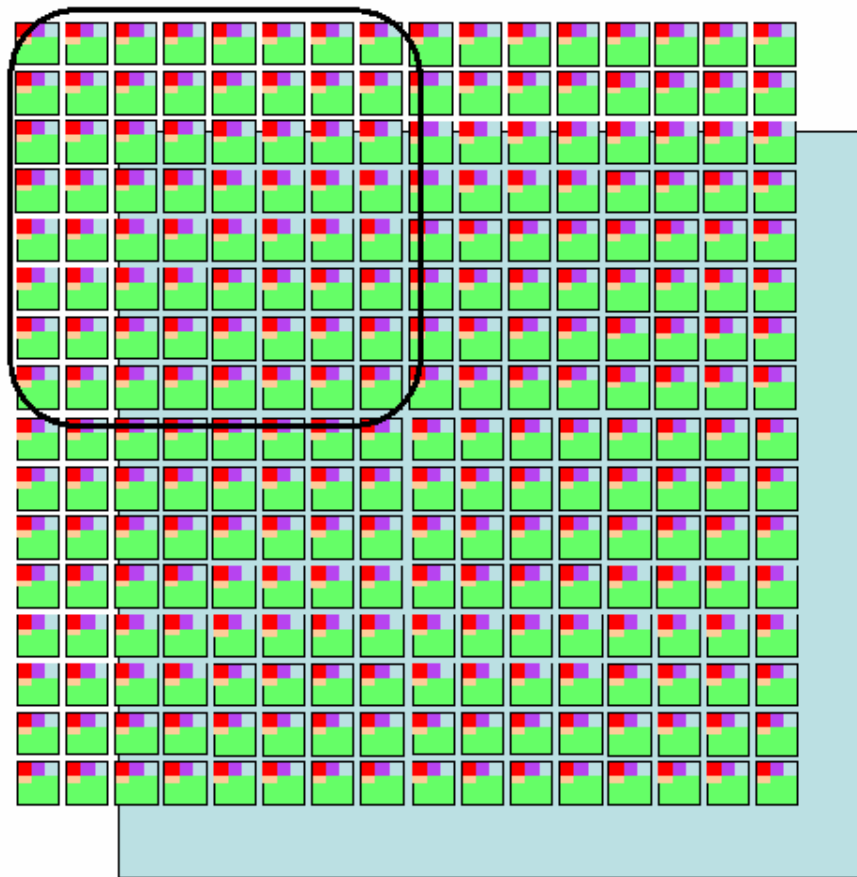
- Actual design  
superimposes row and  
column drivers with the  
crosspoint array





# Looking forward

*Map thread to PEs based on granularity,  
power, or cache working set*



*3-D integrated memory  
(stacked DRAM, MRAM, optical I/O)*

- 2012-era EDGE CMP
  - 8GHz at reasonable clock rate
  - 2 TFlops peak
  - 256 PEs
  - 32K instruction window
- Flexible mapping of threads to PEs
  - 256 small processors
  - Or, small number of large processors
  - Embedded network
- Need high-speed BW
- Ongoing analysis
  - What will be power dissipation?
  - How well does this design compare to fixed-granularity CMPs?
  - Can we exploit direct core-to-core communication without killing the programmer?





# Beyond Transistors

---

- Applications Requirements
- Thermodynamic limits to total power
  - Superconducting logic and Carnot cycle
- Upside potential of advanced architectures/PIM
- Some nanotech technologies on the horizon
- Reversible logic may defeat thermodynamic limitations
- Upside potential of quantum computing
  - Quantum speedup: none, quadratic, exponential
  - Algorithms numerical/cryptanalysis, simulation

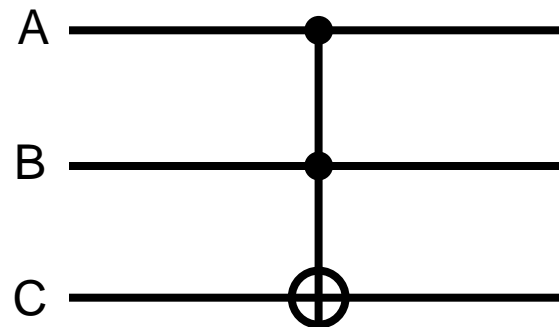




# Reversible Logic – Toffoli Gate

---

- The Toffoli gate is logically complete
- Reversible logic notation shown to right →
  - Bits shown as horizontal lines
  - Time nominally flows to right, but reverses naturally
- Function
  - If A and B true, invert C
- Note: self-inverse

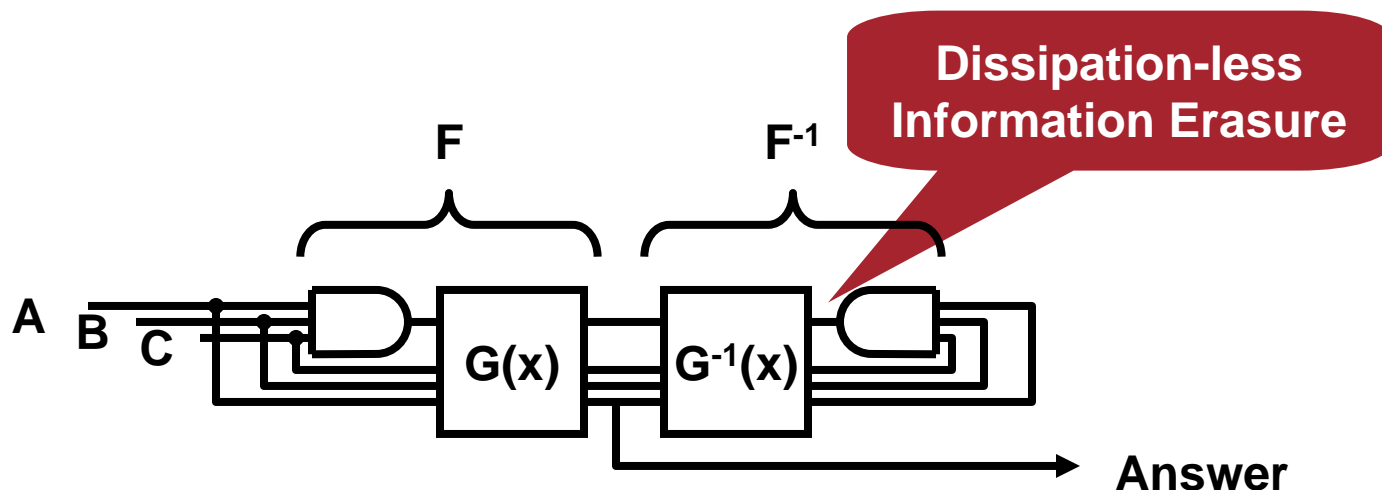






# Reversible Logic Can Beat Landauer's Limit

- Any function can be made reversible by saving its inputs
- Diagram below outlines an asymptotically zero-energy way to perform the AND function, in composition with other logical operations

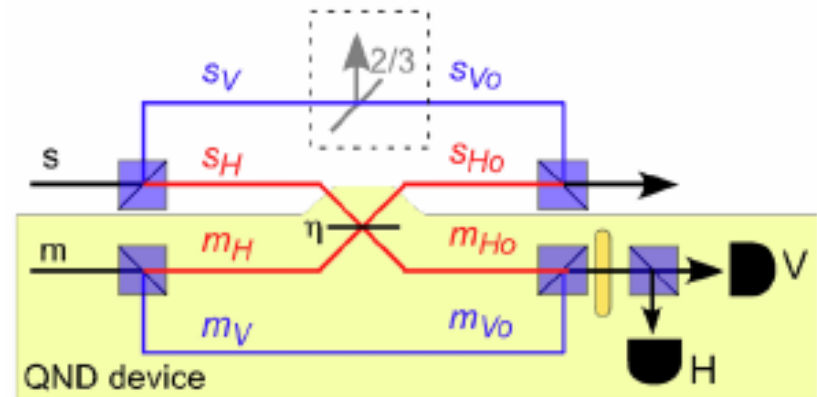






# Reversible Logic Example

- One photon headed to a glass plate goes through
- Two photons also go through, but phase shift each other a little bit
- By appropriate recombinations, a “controlled not” can be created
- A glass plate needs no power supply



- Measuring a Photonic Qubit without Destroying It. GJ Pryde, JL O'Brien, AG White, SD Bartlett, and TC Ralph. Centre for Quantum Computer Technology, ...

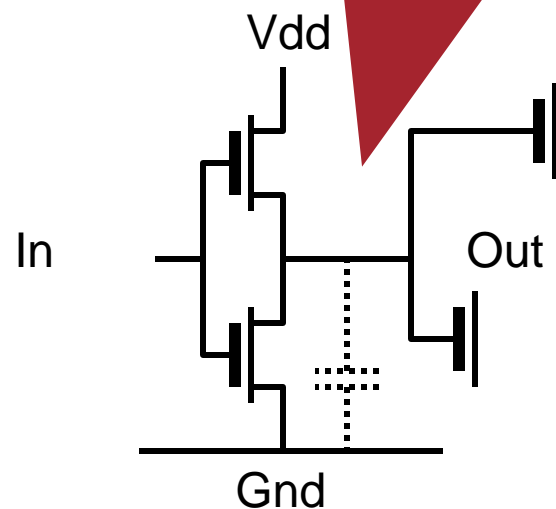




# Today's Universal Logic & Reliability Limit

- Today's logic operates on a simple principle
  - Create a "1" by taking charge from the positive supply
  - Create a "0" by sending charge to the negative supply
- Energy Consumption
  - Each gate switch generates  $E_{sw} = \frac{1}{2} CV^2 > \sim 100k_B T$  heat

Signal energy must be greater than  $\sim 100 k_B T$  to avoid spontaneous glitches. To change a bit, convert energy to heat.



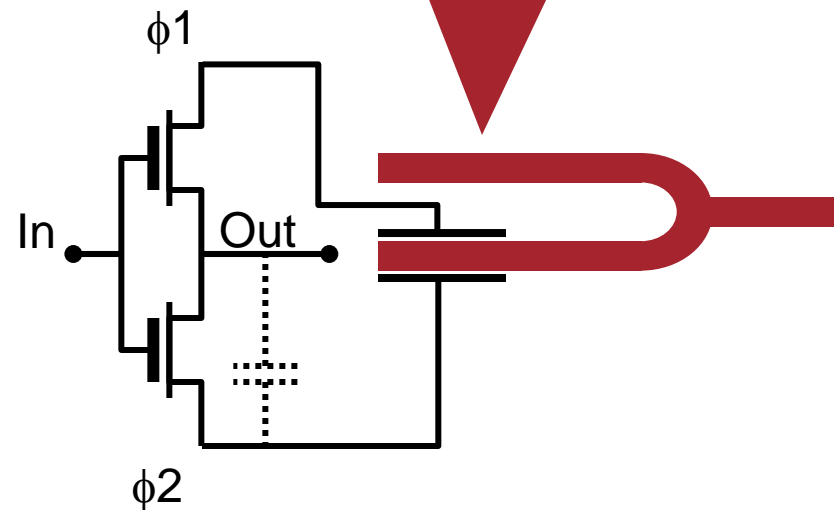




# “Recycling” Power

- The  $100k_B T$  limit appears unbeatable, but the energy can be “recycled”
- Diagram shows a “SCRL” circuit with regular transistors
- Power comes through a largely loss less resonant device (tuning fork)
- No apology offered for the mechanical device; this is the price of progress

Signal energy must be greater than  $\sim 100 k_B T$  to avoid spontaneous glitches. However, signal energy is recycled by tuning fork



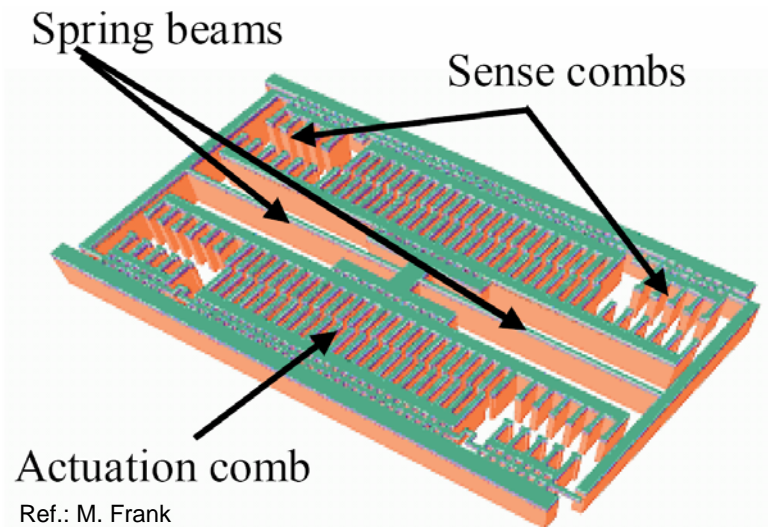


# Resonant Clocks

---

- A Resonant Clock is not perpetual motion, but instead reduces energy similarly to:
  - (a) lifting you child from the ground to the countertop 20 times
  - (b) giving your child a good push on a swingset and letting him/her go 20 cycles

- Tuning Fork
  - Nice idea but slow
- MEMs Resonator
  - Moderate speed and compatible with silicon fabrication







# Resonator Activity

- Nano resonators of appropriate frequency and 1 nW energy levels are available for cell phone filters.
- Frequency-Q products over  $10^{13}$
- However, power levels are too low
- For logic, engineer would like to design a non sinusoidal waveform

Table 1: Frequency vs. diameter for different materials

Diameter	16 $\mu\text{m}$		20 $\mu\text{m}$		24 $\mu\text{m}$	
Mode	1st	2nd	1st	2nd	1st	2nd
Si	318.2	868.9	254.5	695.1	212.1	579.3
SiC	449.1	1238.2	359.3	990.6	299.1	825.5
Diamond	698.8	1939.1	559.0	1551.3	465.8	1292.8

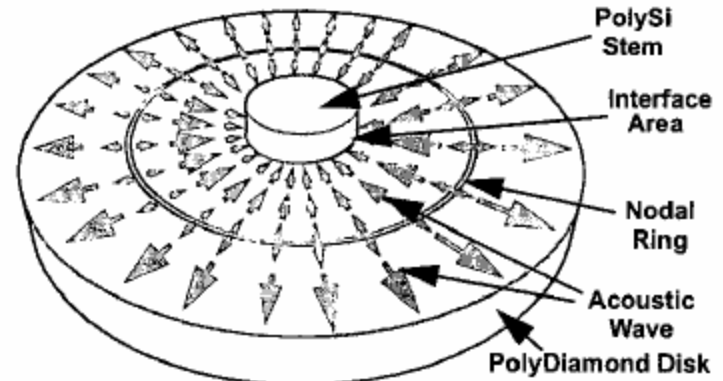


Fig. 2: Schematic of a polydiamond disk resonator with polysilicon stem illustrating acoustic wave propagation in its 2nd radial contour mode.

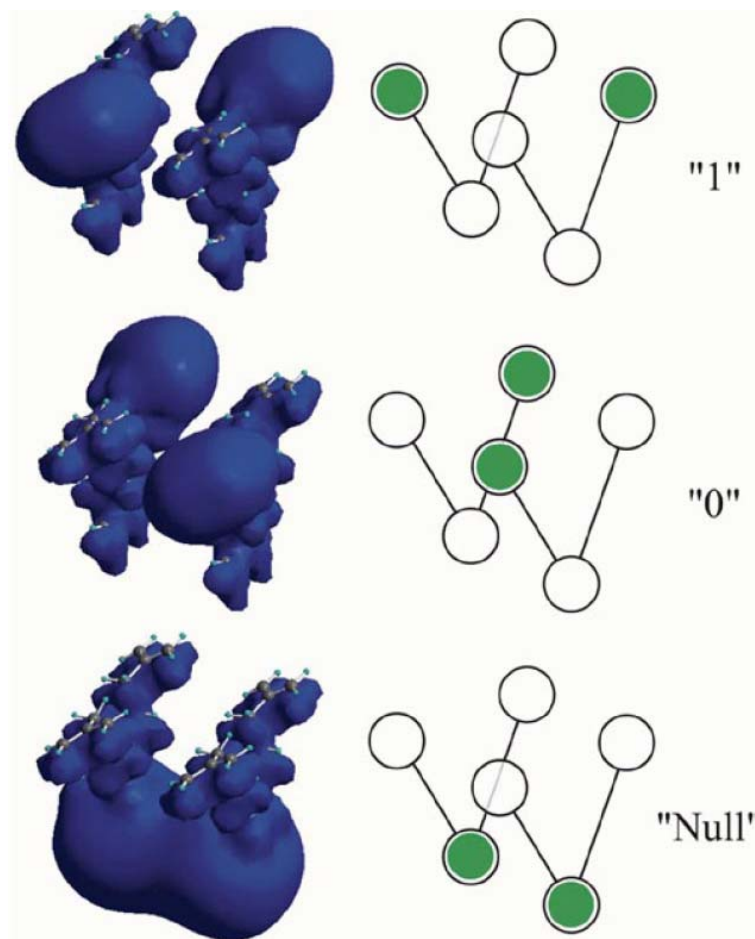
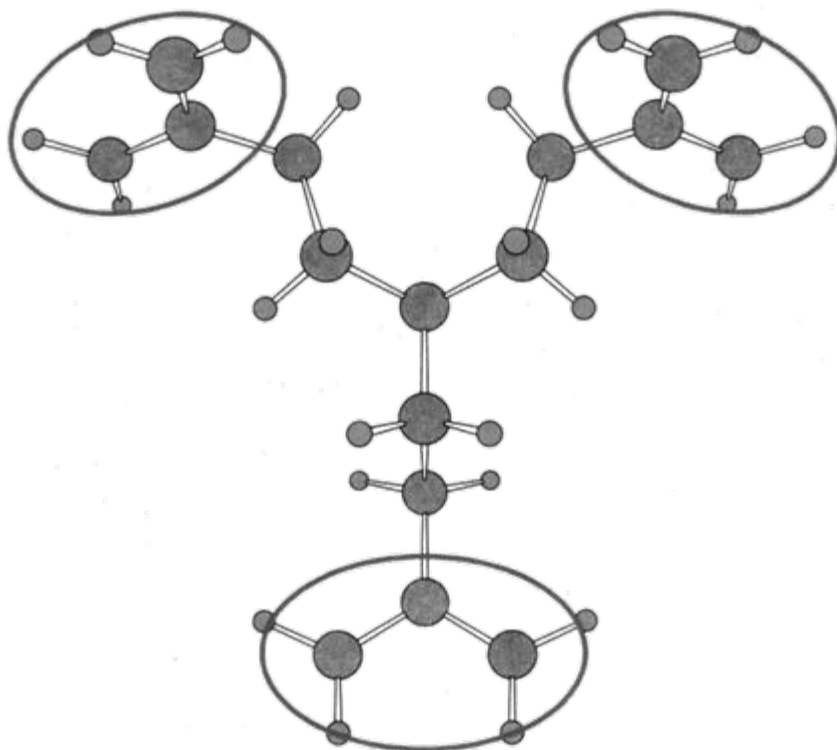
1.51-GHz nanocrystalline diamond micromechanical disk resonator with material-mismatched isolating support, J Wang, JE Butler, T Feygelson, CTC Nguyen - Tech. Dig., 17 th Int. IEEE Micro Electro Mech. Syst. Conf.





# A New Computing Device: Quantum Dots

- Pairs of molecules create a memory cell or a logic gate



Ref. "Clocked Molecular Quantum-Dot Cellular Automata," Craig S. Lent and Beth Isaksen  
IEEE TRANSACTIONS ON ELECTRON DEVICES, VOL. 50, NO. 9, SEPTEMBER 2003



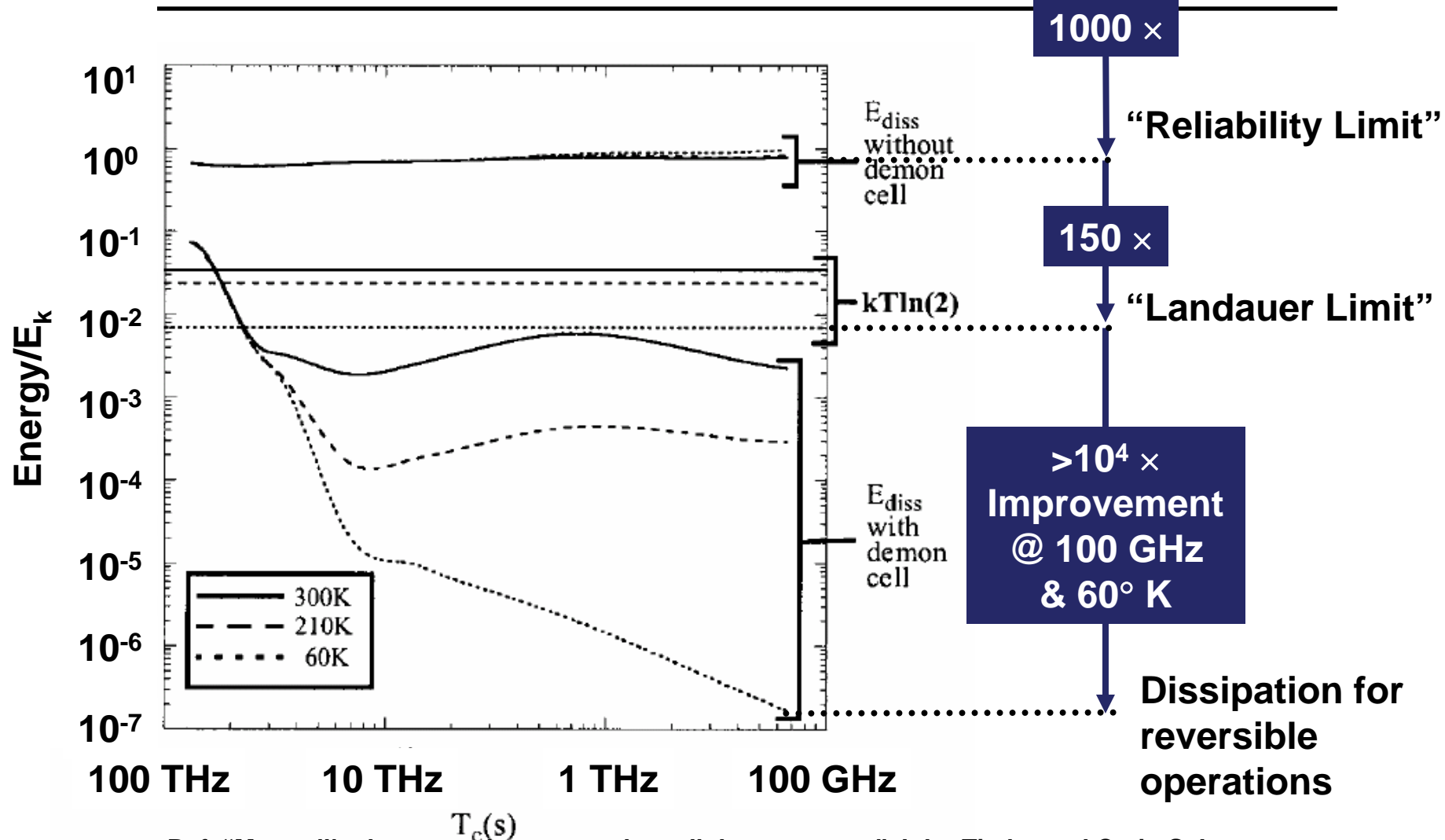






2004 Device Level  
.....

# Upside Potential of Quantum Dots



Ref. “Maxwell’s demon and quantum-dot cellular automata,” John Timler and Craig S. Lent,  
JOURNAL OF APPLIED PHYSICS 15 JULY 2003





# Reversible Multiplier Status

- **8×8 Multiplier Designed, Fabricated, and Tested by IBM & University of Michigan**
- **Power savings was up to 4:1**

## A True Single-Phase 8-bit Adiabatic Multiplier

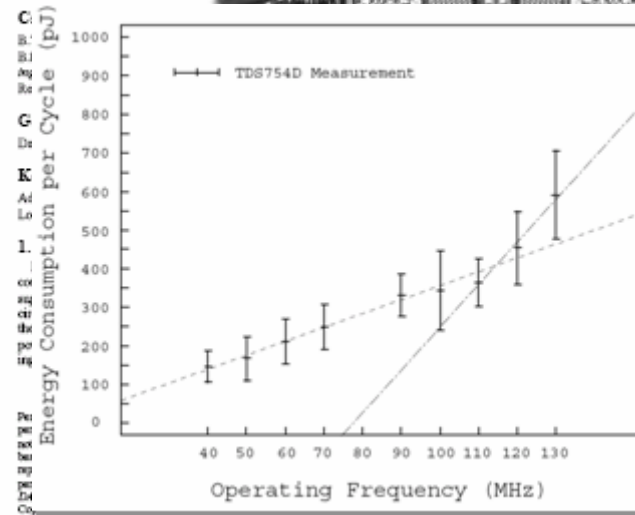
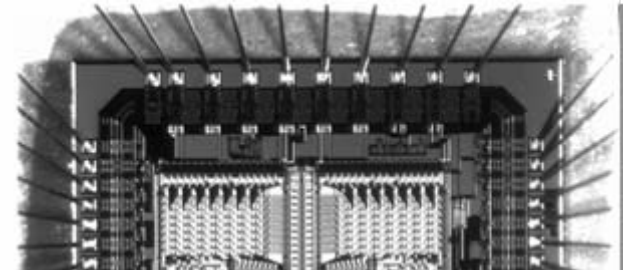
Suhwan Kim  
T. J. Watson Research Center  
IBM Research Division  
Yorktown Heights, NY 10598  
suhwan@us.ibm.com

Conrad H. Ziesler  
EECS Department  
University of Michigan  
Ann Arbor, MI 48109  
cziesler@eecs.umich.edu

Marios C. Papaefthymiou  
EECS Department  
University of Michigan  
Ann Arbor, MI 48109  
marios@eecs.umich.edu

### ABSTRACT

This paper presents the design of a multiplier. Both the multiplier and the multiplier have been designed using a true adiabatic energy recovery technique. Energy is supplied to the multiplier by a power-clock waveform that is generated with post-layout extraction at clock frequencies as high as 130 MHz per operation at 200 MHz. The multiplier has been fabricated in a 0.5 μm standard CMOS technology. Current day operating frequencies up to 130 MHz are measured. Measured dissipation is shown.



### test chip.

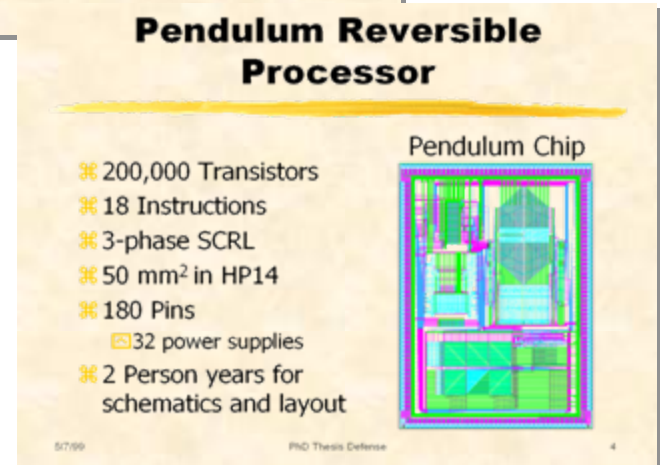
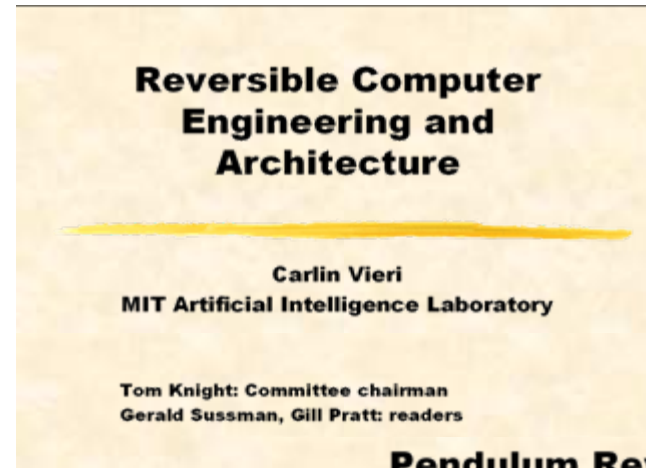
process a lower range energy than a voltage we designed for clock rate of 100MHz, only 91pJ per operation is roughly 4 times less, dissipating only. These efficiency measurement tools and device primarily aimed at substantial energy optimization. metal, 1-poly, 0.5 μm, experimentally validated size up to 130MHz.





# Reversible Microprocessor Status

- **Status**
  - Subject of Ph. D. thesis
  - Chip laid out (no floating point)
  - RISC instruction set
  - C-like language
  - Compiler
  - Demonstrated on a PDE
  - However: really weird and not general to program with  $+=$ ,  $-=$ , etc. rather than  $=$







# Beyond Transistors

---

- Applications Requirements
- Thermodynamic limits to total power
  - Superconducting logic and Carnot cycle
- Upside potential of advanced architectures/PIM
- Some nanotech technologies on the horizon
- Reversible logic may defeat thermodynamic limitations
- Upside potential of quantum computing
  - Quantum speedup: none, quadratic, exponential
  - Algorithms numerical/cryptanalysis, simulation





# Outline

---

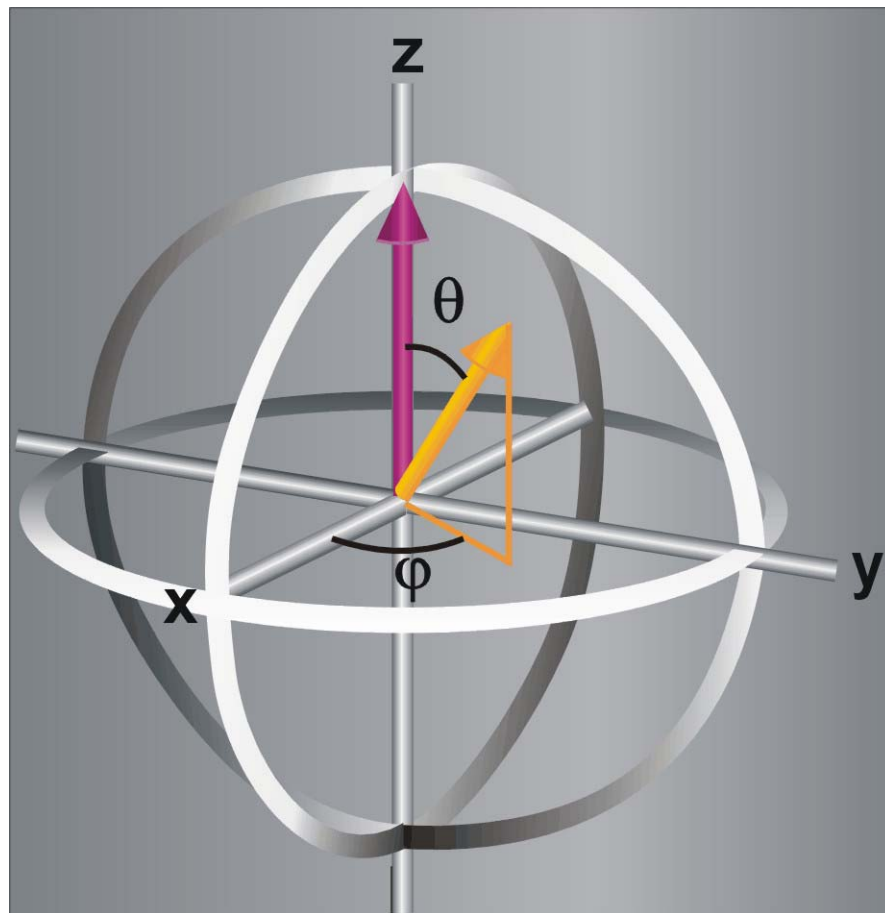
- **Overview**
  - Insight From a Dinner Conversation in DC
  - Super-Roadmap
- **Limitations to Moore's Law**
  - Transistor Scaling Limits per ITRS
  - Consequence to System Performance per Burger and Keckler Study
- **What It Means and What To Do About It**
  - Legacy C++/Fortran
  - Systolic Array Lessons
  - New Very Parallel Code
  - Special Purpose Assist
  - Analog/Neural Net
- **Over the Horizon**
  - Reversible Logic
  - Quantum Computing





# Why Quantum Computing is Interesting

- A Superset of Digital
  - Spin “up” is a 1
  - Spin “down” is a 0
  - Other spins
    - Sidewise
    - Entangled
    - Phase
  - Like wildcards
    - 1011??????
    - Up to  $2^N$  states  $\rightarrow$  in “quantum parallel”



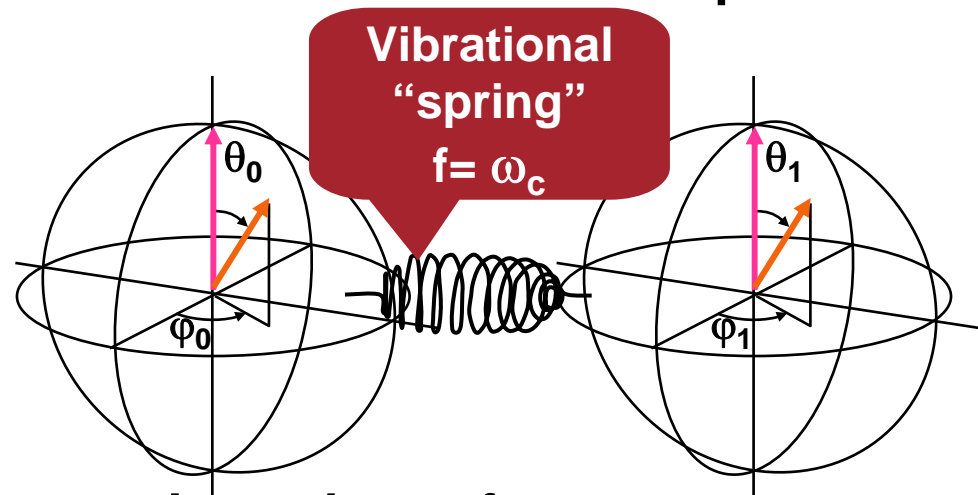




# Ion Trap Quantum Gates

- Hyperfine (internal qubit) frequencies are  $\omega_0$  and  $\omega_1$
- Vibrational center of mass frequency is  $\omega_c$
- Laser at frequency  $\omega_0 \pm \omega_c$  or  $\omega_1 \pm \omega_c$  couples qubit from hyperfine state to vibrational state and back
- Appropriate frequencies selectively move qubits based on data
- Works on superpositions

- Two ions in an ion trap



- Laser beam frequency  $\omega$

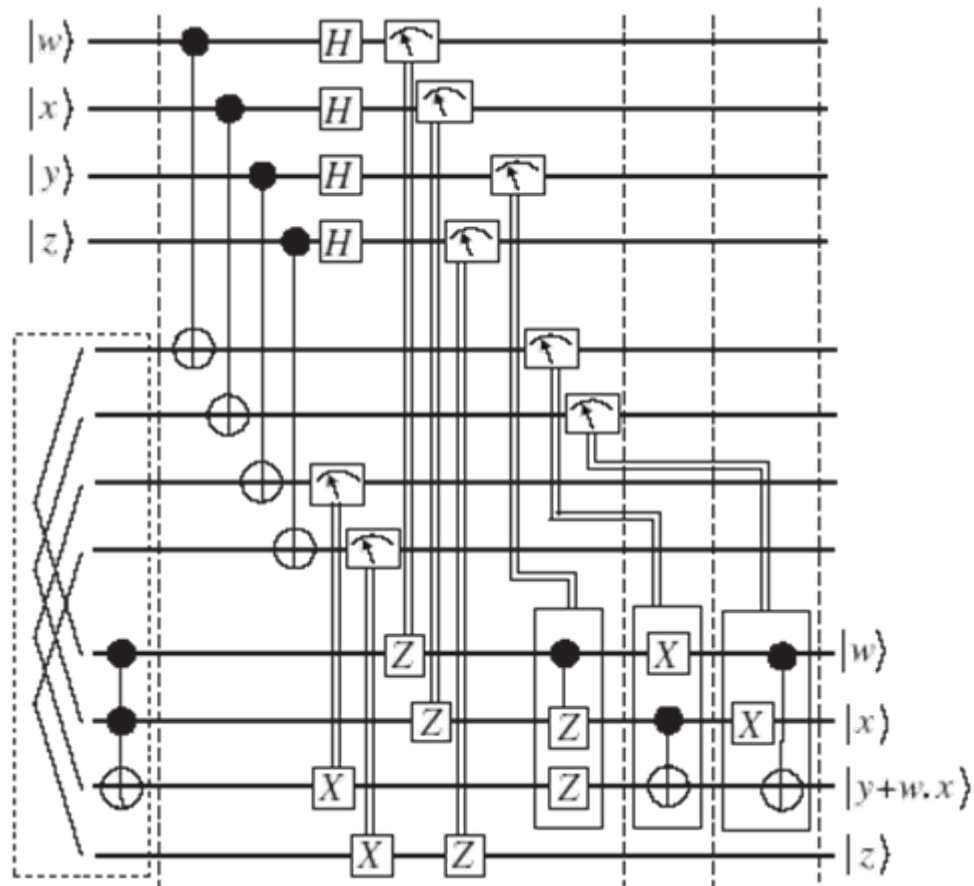




# Reliable Quantum Operations

- Microprocessors use ECC for memory and crash when logic errors occur
- QEC includes technology for error detection and correction on both memory and operations
- Example on right performs Toffoli operation on protected blocks, producing a protected block

## • Toffoli Gate



“Fault-Tolerant Logical Gate Networks for CSS Codes,” Steane, A, Iblinson, B, quant-ph/0311014





# Beyond Transistors

---

- Applications Requirements
- Thermodynamic limits to total power
  - Superconducting logic and Carnot cycle
- Upside potential of advanced architectures/PIM
- Some nanotech technologies on the horizon
- Reversible logic may defeat thermodynamic limitations
- Upside potential of quantum computing
  - Quantum speedup: none, quadratic, exponential
  - Algorithms numerical/cryptanalysis, simulation





# Quantum “Algorithms”

---

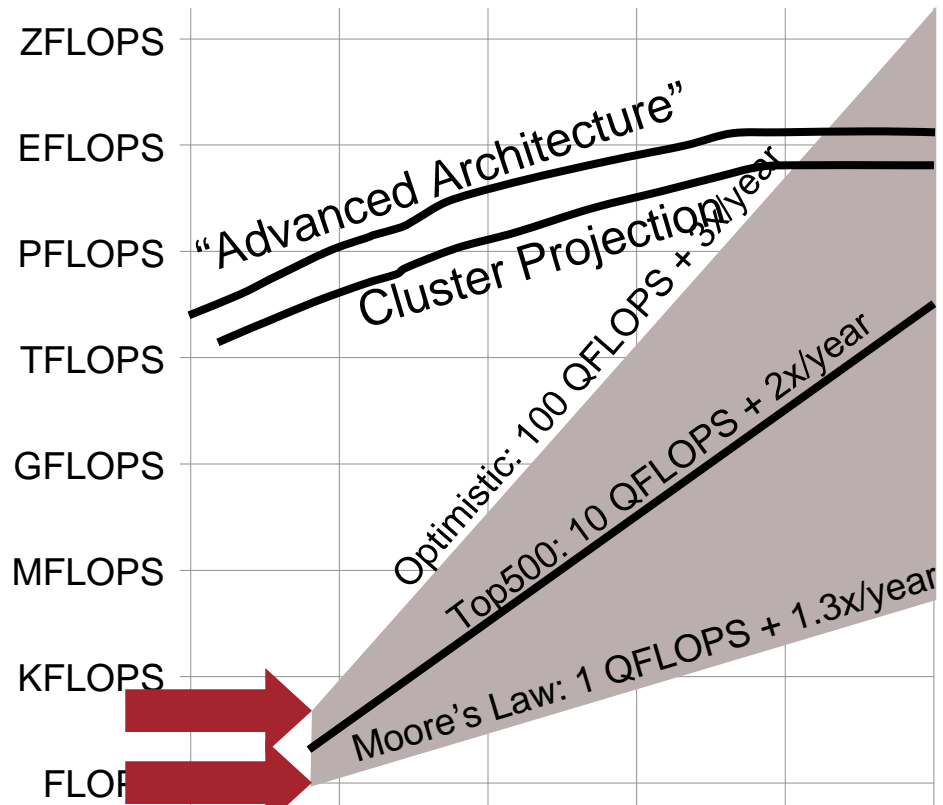
- **Category 1: No Speedup**
  - A quantum computer will be able to execute conventional computer logic – with no advantage
- **Category 2: Grover’s Algorithm with Quadratic Speedup**
  - Given an “Oracle” function, a QC can search, average, min, max, integrate, in  $n^{1/2}$  steps to same accuracy as a classical computer gets in  $n$  steps
- **Category 3: Shor’s Algorithm with Exponential Speedup**
  - There are a series of problems related to the “hidden subgroup problem” that can be solved with exponential speedup over a classical computer.
  - Includes code cracking and physics simulation





# Emergence of Quantum Computing

- There appears to be an engineering case for quantum computers of 1-100 Q-FLOPS
- One would expect an exponential growth rate for quantum computers similar to Moore's Law, but the rate constant is impossible to predict, so three possibilities have been graphed



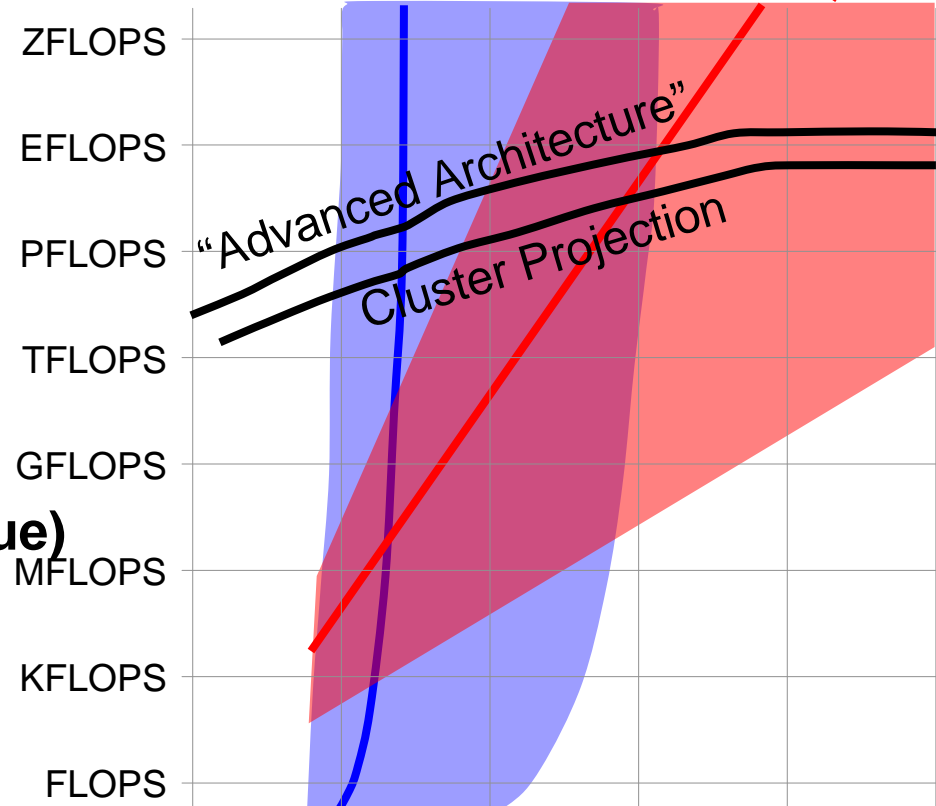
**NOTE: Years are gone because I hesitate to predict!**





# Quantum Applications

- Consider the classical computer equivalent to a Quantum Computer
- First use believed to be factoring in cryptanalysis, with exponential speedup over classical computers (blue)
- Second, a quantum computer can also be used for other applications (pink) with quadratic speedup (e. g. Actinide chemistry)



**NOTE: Years are gone because I hesitate to predict!**





# Beyond Transistors

---

- Applications Requirements
- Thermodynamic limits to total power
  - Superconducting logic and Carnot cycle
- Upside potential of advanced architectures/PIM
- Some nanotech technologies on the horizon
- Reversible logic may defeat thermodynamic limitations
- Upside potential of quantum computing
  - Quantum speedup: none, quadratic, exponential
  - Algorithms numerical/cryptanalysis, simulation





# One Slide Taxonomy of Quantum Algorithms

---

- **Exponential speedup for**
  - Period finding (see →)
  - Hidden subgroup problem
    - Factoring
    - Discrete logarithms
    - Algorithms for problems I never heard about except for QC
- **Quadratic speedup for**
  - Searching
  - Average, min, max
- **Feynman asserted that a QC could combat low efficiency of classical computer for simulating quantum problems**
  - This assertion has been repeatedly proven, but there are few concrete algorithms
  - This could be a “killer app” domain for supercomputing





# Overall Prescription for Fast Computing

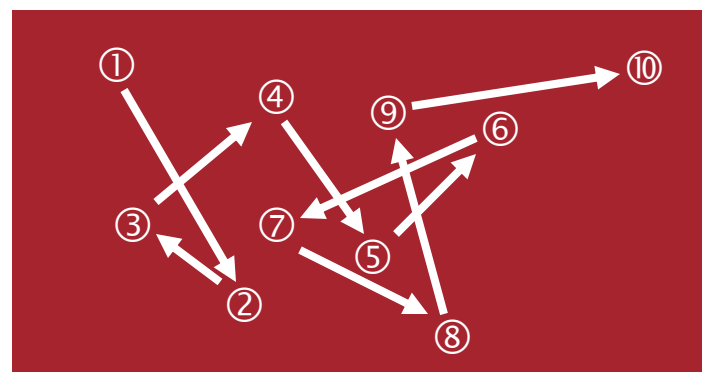
Devices

Architecture

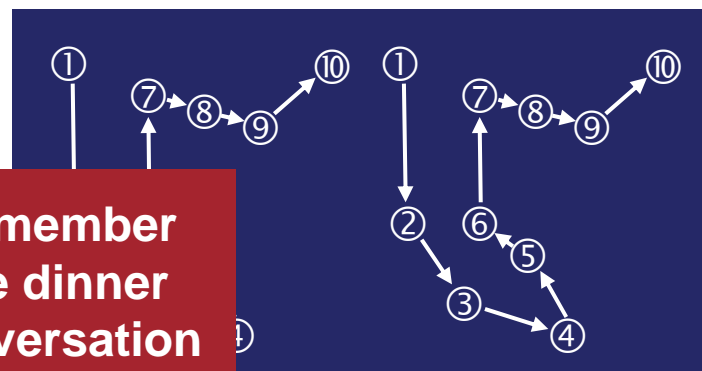
Programmer

- High node visit rate
- Small size
- Fast propagation velocity
- Parallel
- Organize program graph for short distances
- Programming language must aid programmer in creating short, parallel graphs
- Programmer must use language effectively

Bad



Better



Remember  
the dinner  
conversation





## Overall Summary

---

- **Find more parallelism.** While device technology will continue to improve exponentially for some time, exploiting these advances will require more parallelism in code.
- **There is parallelism to exploit for many supercomputing applications areas.**
- **Single-node C++, Fortran, etc. codes will not improve in speed very much at all.**
- **Innovative programming methods will be rewarded by higher performance for a very long time into the future.**

Please fill out the survey



---

# **The International Technology Roadmap for Semiconductors and Its Effect on Scalable High End Computing**

**Peter M. Kogge**

**McCourtney Prof. of CS & Engr, Concurrent Prof. of EE**

**Assoc. Dean for Research, University of Notre Dame**

**IBM Fellow (ret)**





# Why Is Supercomputing Hard In Silicon: Little's Tyranny

---

ILP: Getting tougher & tougher to increase

- Must extract from program
- Must support in very complex H/W

Concurrency

=

Throughput

Latency

*Much less than peak  
and degrading rapidly*

*Getting worse fast!!!!  
(The Memory Wall)*





# Technology Limits to Applications

(from NRC's "Getting Up to Speed")

	Stockpile	Intelligence	Defence	Climate	Plasma	Transportation	Bio-info	Health&Safety	Earthquakes	Geophysics	Astrophysics	Materials	Organ. Systems
Performance Flops			1	X	X						X		
Memory Capacity		X				3	2					X	
Memory Bandwidth		X		X							X	X	4
Memory Latency	X	X		X							X		4
Interconnect Bandwidth		X		X							X	X	4
Interconnect Latency	X	X		X							X		4

It's  
NOT  
Just  
Flops

- 1 Radar Cross section
- 2 Genomics
- 3 Automobile Noise
- 4 Biological Systems Modeling





# Why Look at Technology Scaling

---

- What are the basic units of memory & logic
  - In terms of *functionality* per sq. cm
- How will these change over time
- How with their individual performance characteristics change
- When do real-world limits come into play
  - Power and inter-chip bandwidth
- What's the likely best “*chip*” architectures





# What Seems to Be The Consensus

---

- Silicon will remain with us, but
  - Power becoming dominating concern
  - Individual CPU core complexity flattening
  - Clock rate increases flattening
  - Commodity memory bandwidths stagnant
  - Chip-to-chip growing in importance
- Impact on building-block chip architecture
  - Moore's Law converts to parallelism – within the chip
  - Line between “Logic” and “Memory” chips blurs
- We will increase “*threads per die*” not “*IPS/core*”





# Outline

---

- **Silicon Fundamentals**
- **Scaling**
- **ITRS Roadmap**
- **Limits on Classical Chips**
- **Multi-threading & Multi-core**
- **Processing in Memory**





---

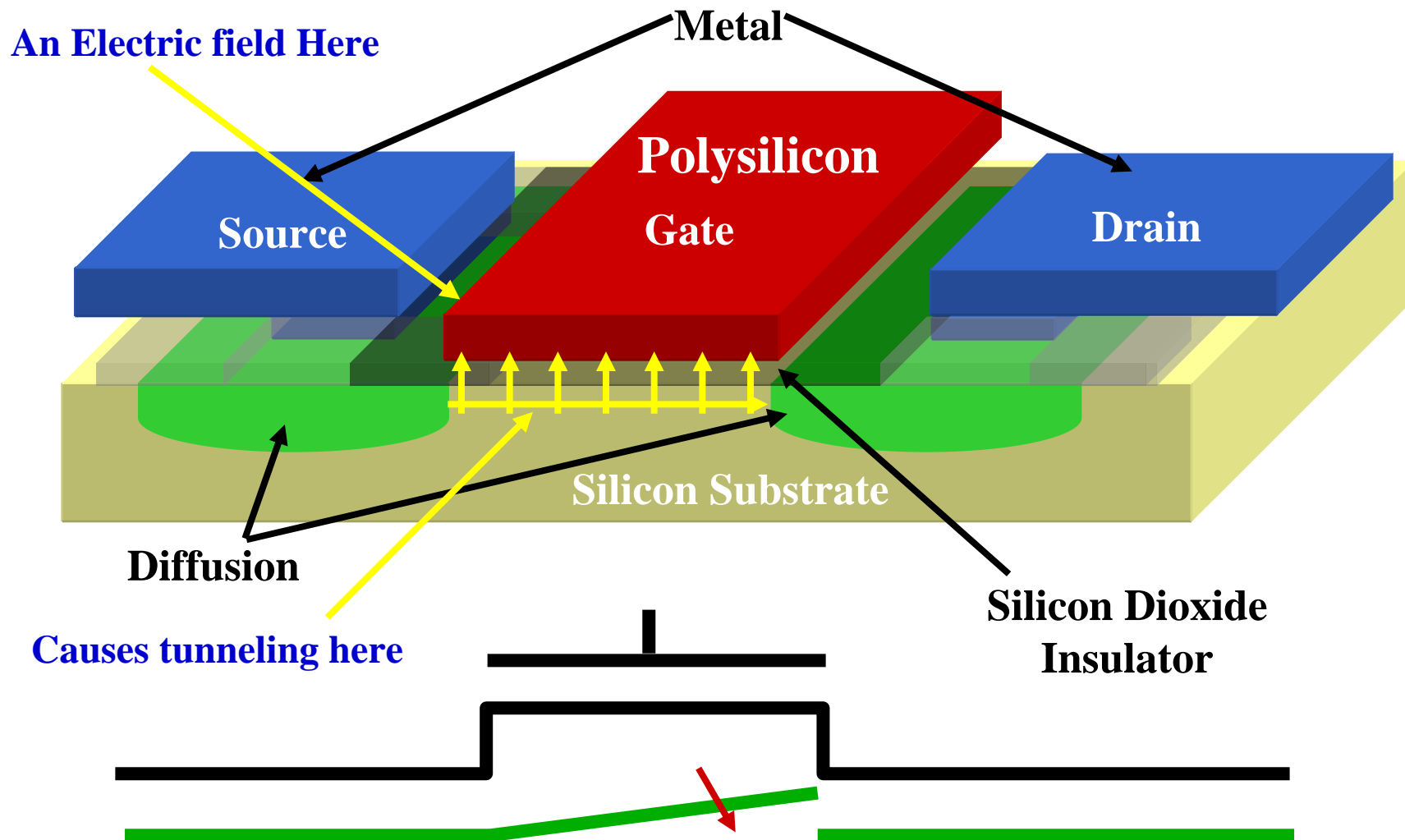
# Silicon Fundamentals

- MOSFET Transistor
- Simple Logic Circuits
- Variations of Memory
- Multiple Levels of Metal
- Off-Chip Interconnect





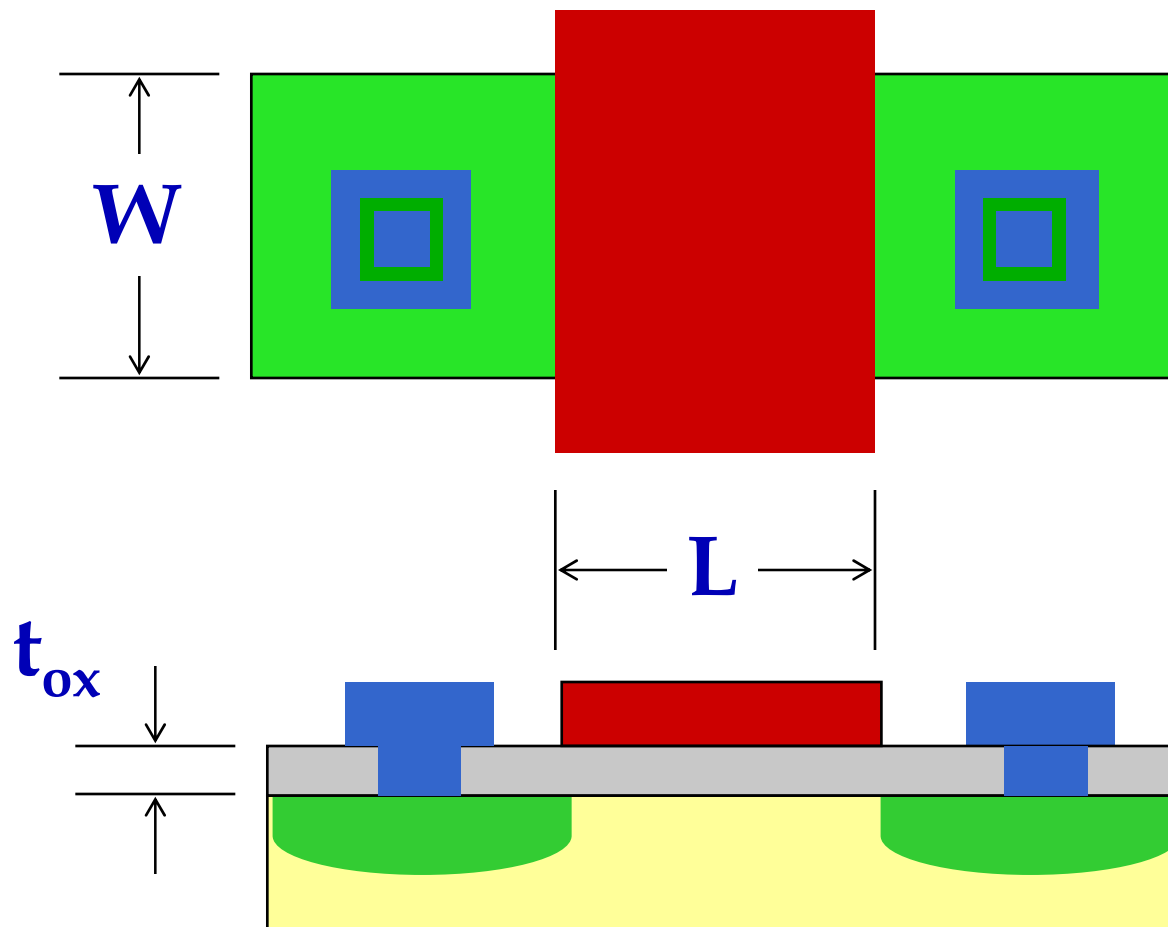
# A MOSFET Transistor







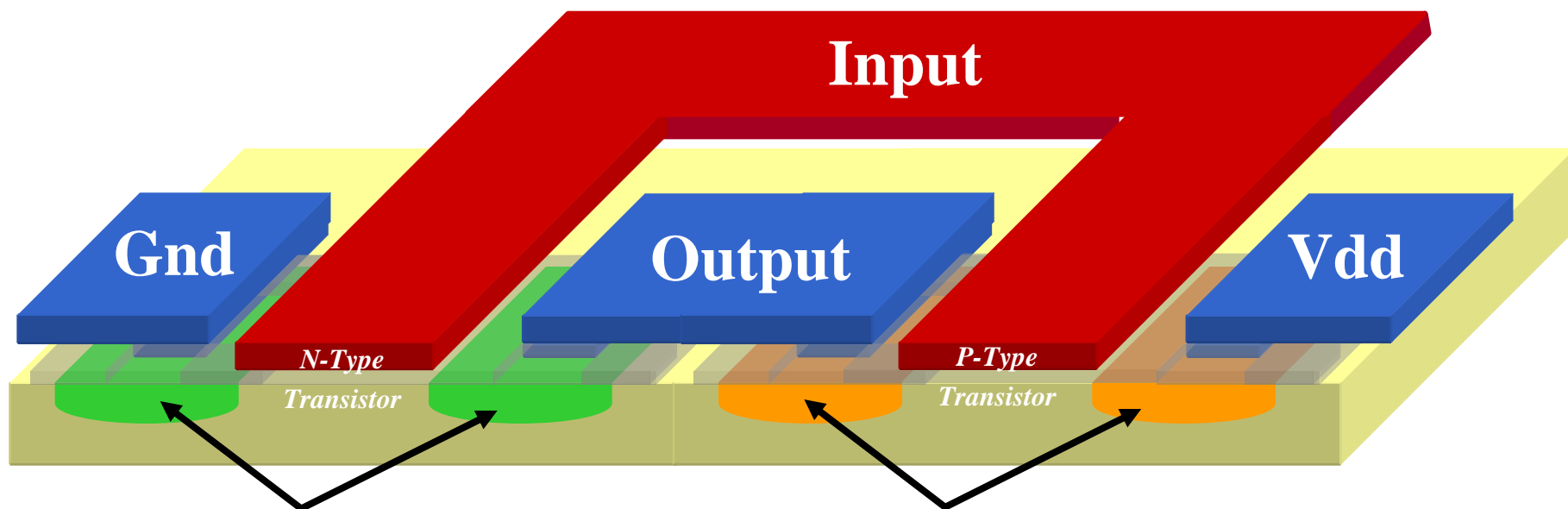
# Key Device Parameters







# A Logic Inverter

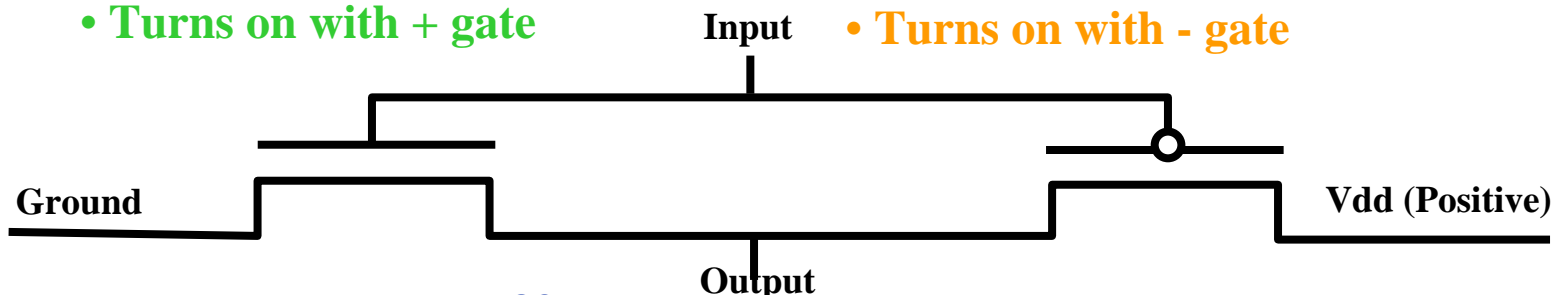


## N-Type Diffusion/Transistor

- electron rich
- Turns on with + gate

## P-Type Diffusion/Transistor

- electron poor
- Turns on with - gate





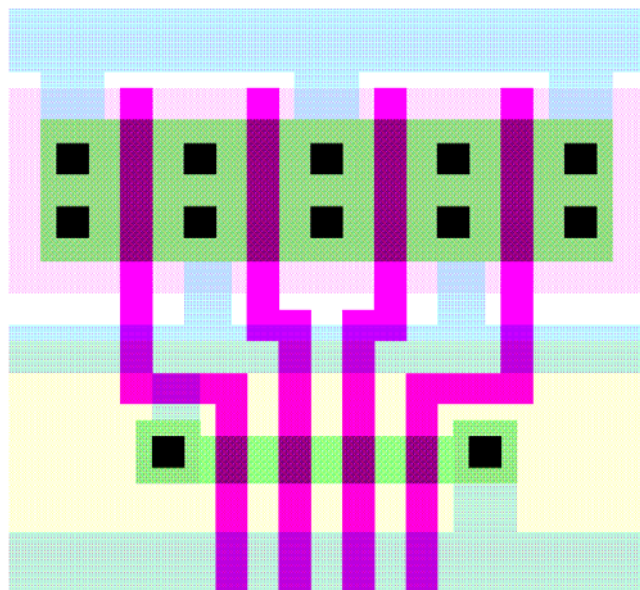


# Logic Examples

Vdd

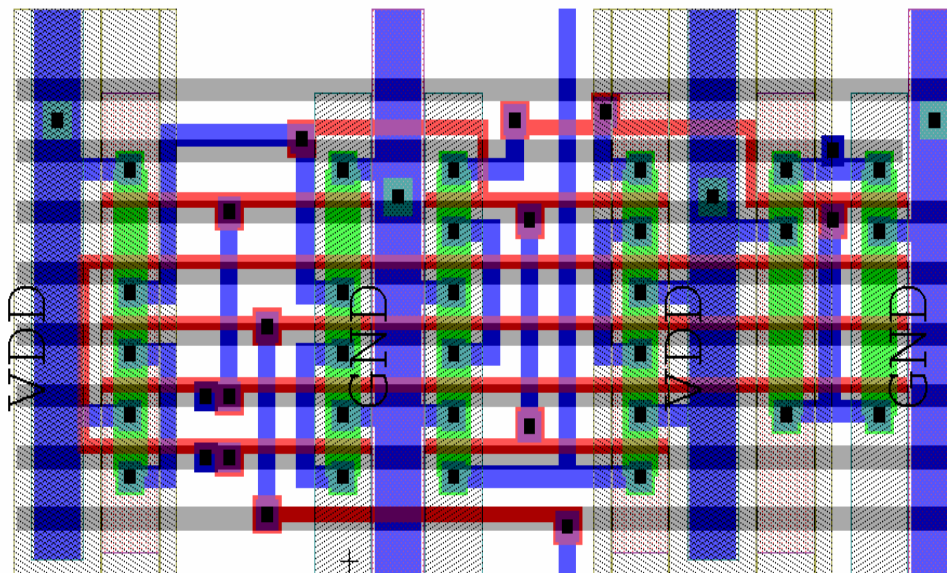
Out

GND



In1 In2 In3 In4

**4 Input NAND Gate**

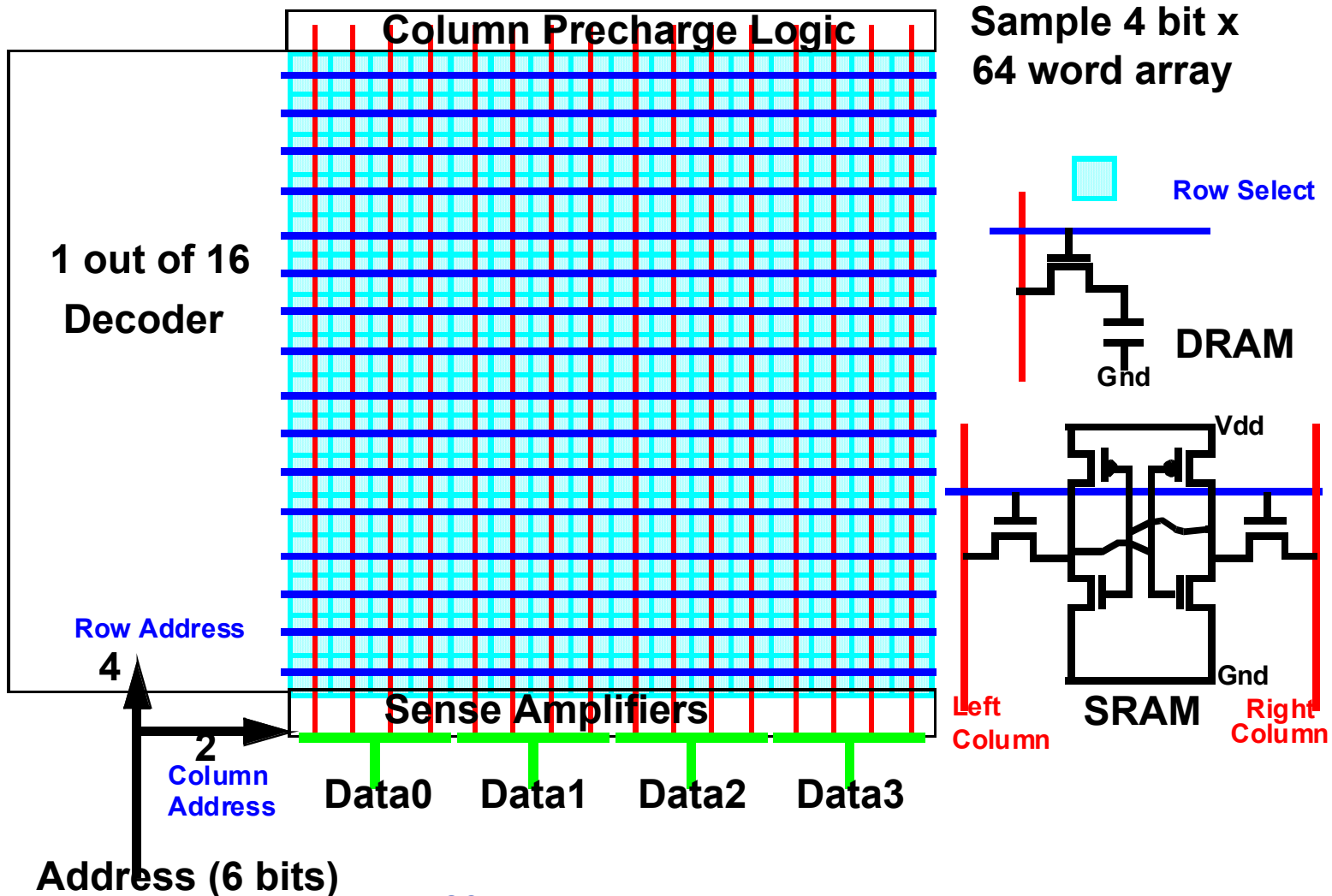


**Full Adder**





# Memory Arrays



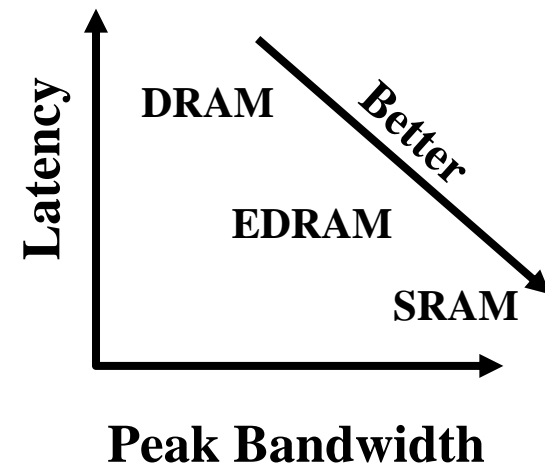
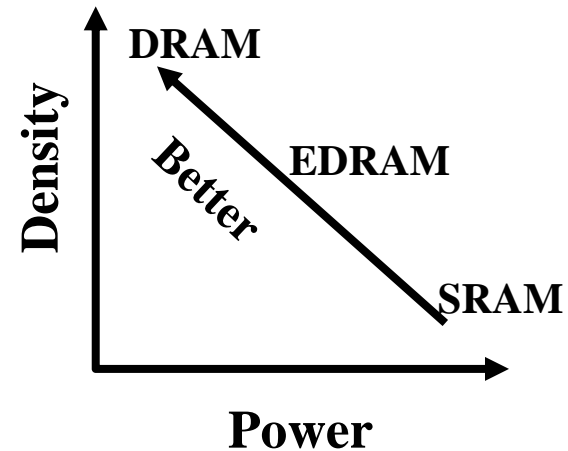




# Key Types of Memory Cells

- **Commodity DRAM**
- **Embedded DRAM**
- **SRAM**
- **Flash**
  - NAND Type
  - NOR Type

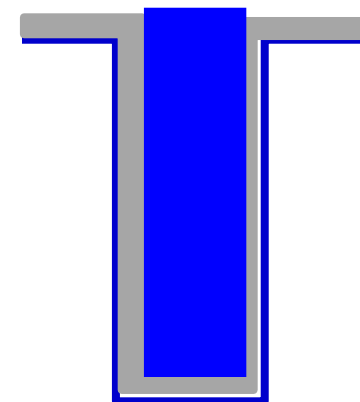
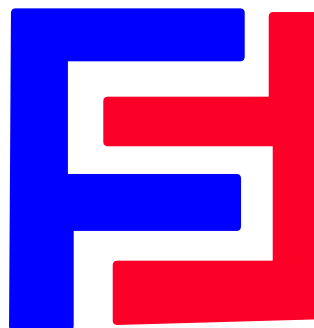
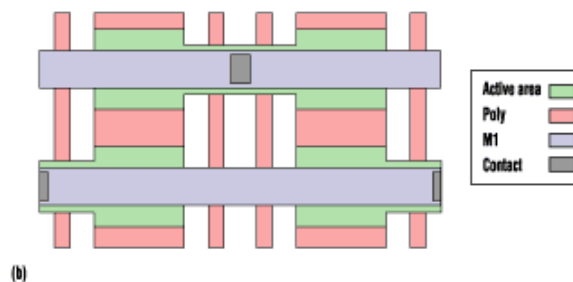
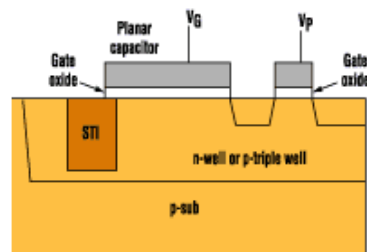
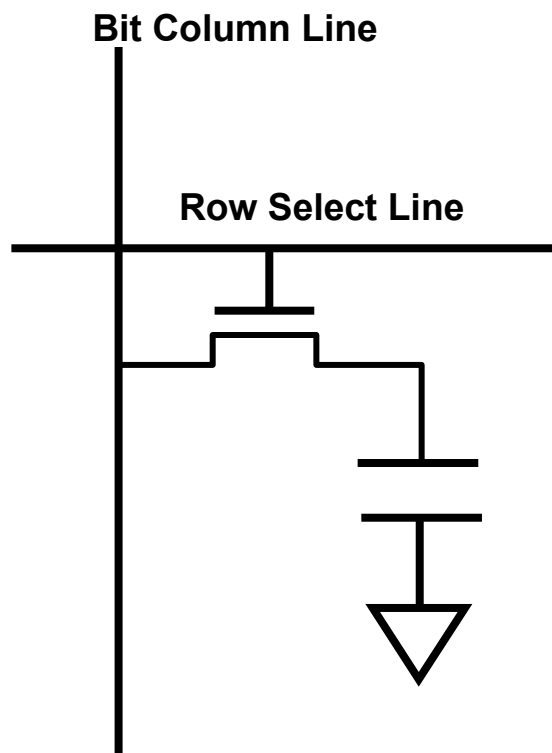
**No single optimal choice!**







# Compact DRAM Cells for Memory Arrays



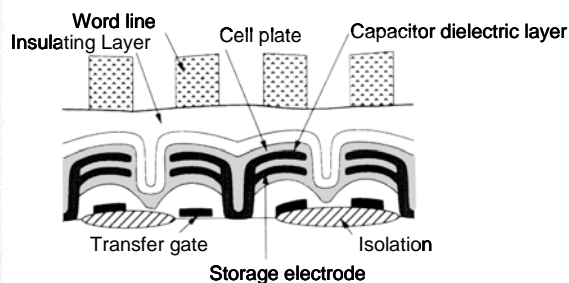
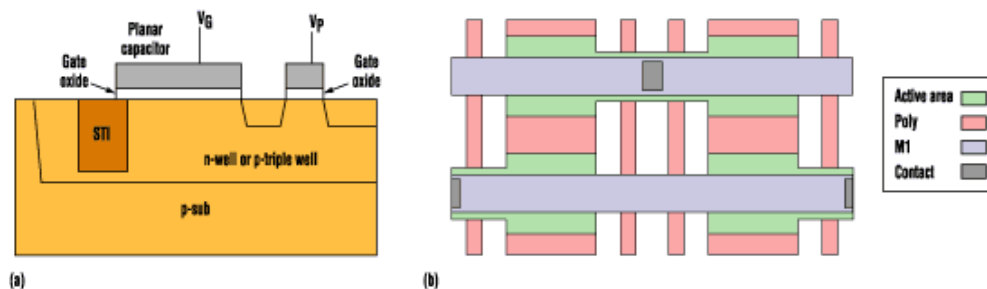
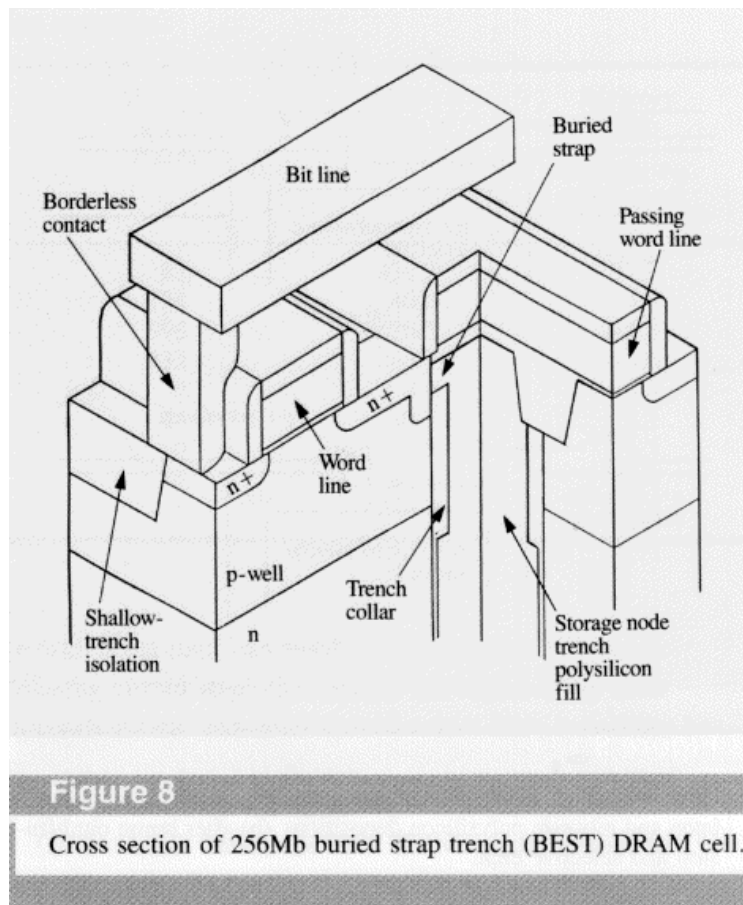
Stacked-capacitor Cell

Trench Cell

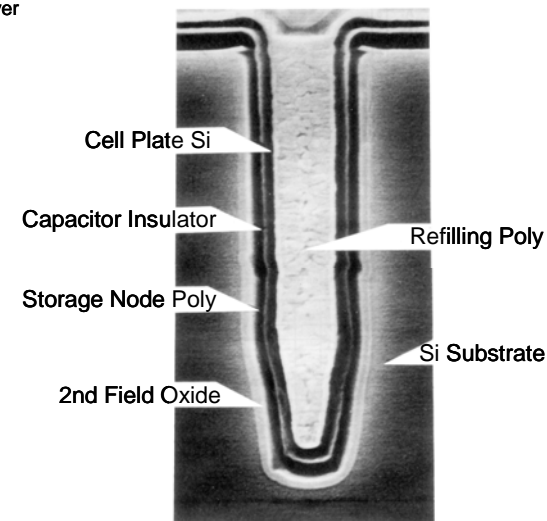
E. Adler, et al, " The evolution of IBM CMOS DRAM technology," IBM J. R&D, Vol. 39, No. 1/2, p.167, 1995.



# Compact DRAM Cells for Memory Arrays



Stacked-capacitor Cell



Trench Cell

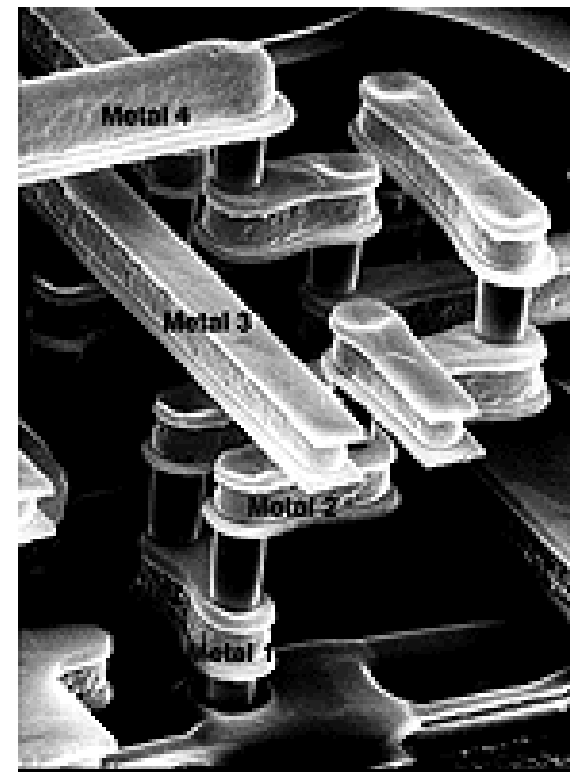
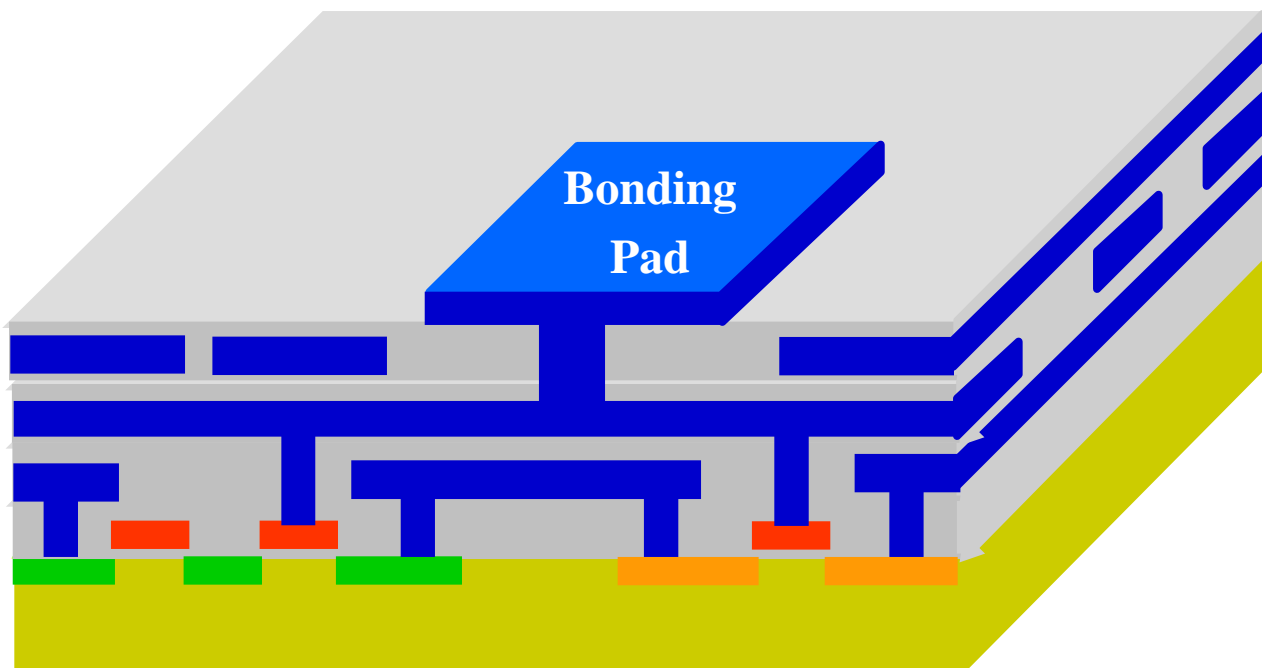
E. Adler, et al, "The evolution of IBM CMOS DRAM technology," IBM J. R&D, Vol. 39, No. 1/2, p.167, 1995.





# Multiple Levels of Metal

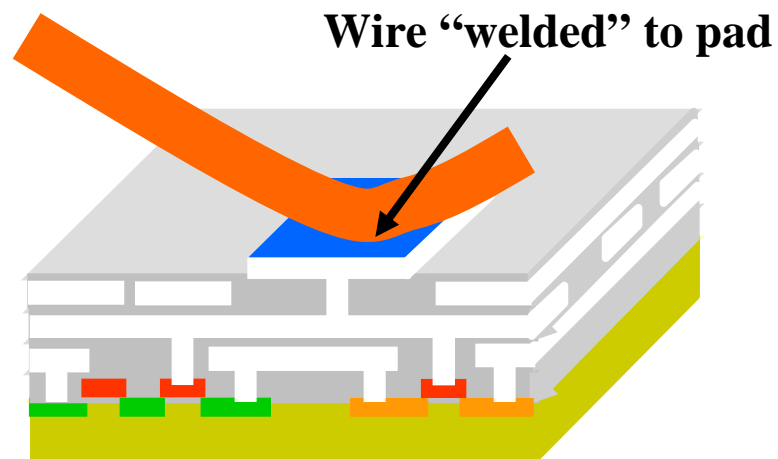
---



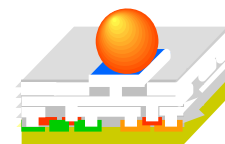
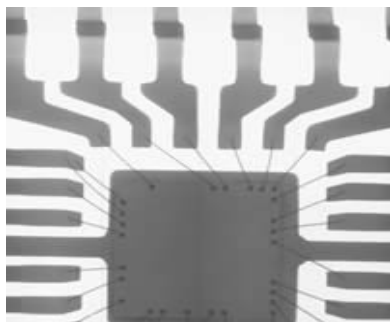




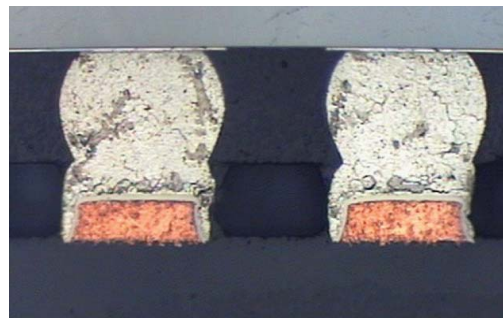
# Off-Chip Interconnect



**Wire Bond**



**C4 Solder Ball**

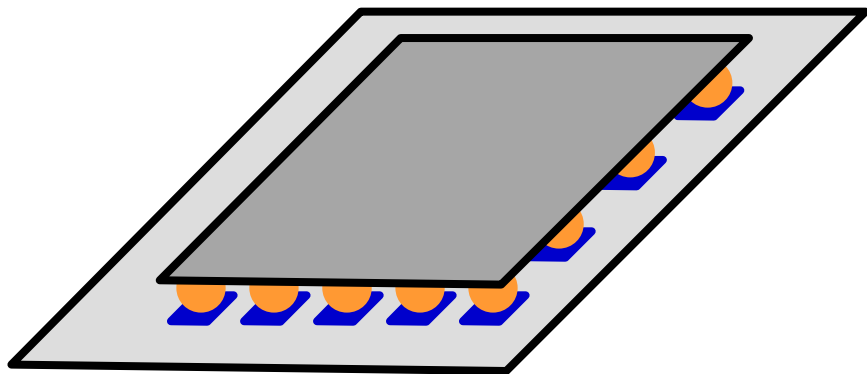




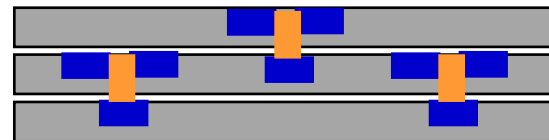


# 3D Chip Stacks

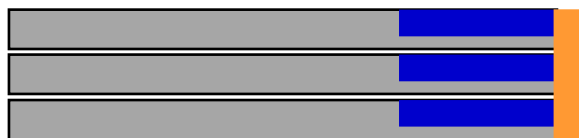
---



**Flip chip**



**Thru-Die Vias**



**Metal wires on side of Cube**





---

# Scaling & ITRS Roadmap

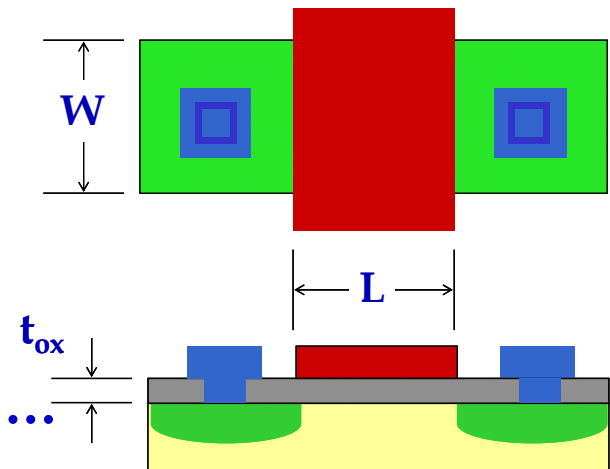




# Device Scaling

Key parameters: Gate length  $L$ , width  $W$

- “On” resistance  $\sim$  to  $L/W$
- “Delay”  $\sim LW/t_{ox}$
- Decreasing  $L$  thus a “good thing”
- Other “shrinkable” dimensions:
  - $t_{ox}$ , metal width, spacing between wires, ...



“Scaling:” shrink some feature by factor “ $S$ ” and:

- *Reduce* chip area to perform some function
- *Increase* frequency of operation
- *Reduce* operating voltage
- *Reduce* circuit power

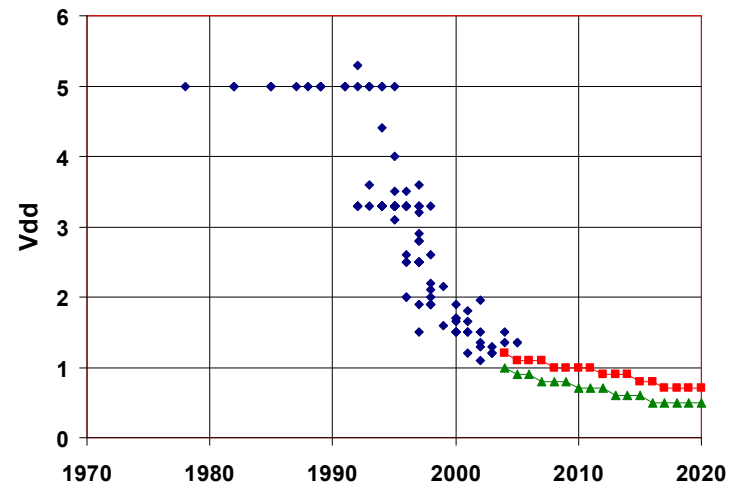
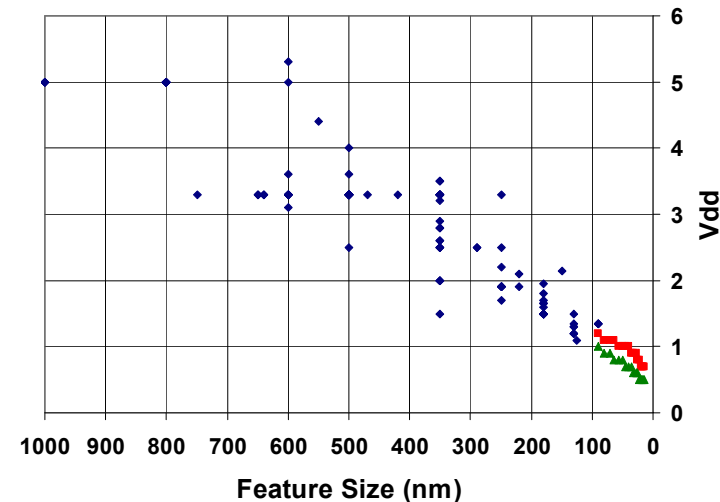
Key Metric: **Power density** = power per unit area





# Variations in Real World Scaling: Primarily Coupling with $V_{dd}$

- **Full scaling:** Ideal if possible
  - Keep gate capacitor E-field constant
  - Requires scaling  $L$ ,  $W$ ,  $t_{ox}$ ,  $V_{dd}$
  - Area shrinks, power drops, higher clock
- **Fixed  $V_{dd}$  Scaling:** Common until late 1990s
  - Scale only  $L$ ,  $W$
  - Keep  $V_{dd}$  constant
  - Same area shrink, very high clock, terrible power
- **General Scaling:** Typical today
  - Different scale factors for different parameters
  - $V_{dd}$  does not drop as fast (approaching another limit)
  - Lower peak clock, but better power & power density







# Approximate Scaling Relationships

Parameter	"Long Channel" Devices			"Short Channel" Devices		
	Full	Fixed V	General	Full	Fixed V	General
W, L	1/S	1/S	1/S	1/S	1/S	1/S
$t_{ox}$	1/S	1/S	1/S	1/S	1/S	1/S
V <sub>dd</sub>	1/S	1	1/U	1/S	1	1/U
Circuit Area	1/S <sup>2</sup>	1/S <sup>2</sup>	1/S <sup>2</sup>	1/S <sup>2</sup>	1/S <sup>2</sup>	1/S <sup>2</sup>
Clock	S	S <sup>2</sup>	S <sup>2</sup> /U	S	S	S
Circuit Power	1/S <sup>2</sup>	S	S/U <sup>3</sup>	1/S <sup>2</sup>	1	1/U <sup>2</sup>
Power Density	1	S <sup>3</sup>	S <sup>3</sup> /U <sup>3</sup>	1	S <sup>2</sup>	S <sup>2</sup> /U <sup>2</sup>

## The Original Moore's Law:

- 4X “*functionality*” every 3 years
- “Interpreted” as ~ S=2 every 3 years





# **International Technology Roadmap for Semiconductors**

---

- **Goal: predict scaling for next 15 years**
  - Convert “Moore’s Law” into detailed projections
  - Identify technical roadblocks
- **Result of a worldwide consensus**
  - U.S.A, Europe, Japan, Korea, and Taiwan
- **Dating back to 1994**
  - Initially every three years
  - But now significant yearly “updates”
- **This data from 2005 update (released Dec. 2005)**
  - <http://www.itrs.net/Links/2005ITRS/Home2005.htm>





# Trends And Challenges Addressed

---

## Trends Charted:

- ***Integration Level:*** Components/chip
- ***Cost:*** \$ per function
- ***Speed:*** Microprocessor clock rate, GHz
- ***Power:*** Laptop or cell phone battery life
- ***Compactness:*** Small and light-weight products
- ***Functionality:*** Nonvolatile memory, imager

## Challenges Identified:

- **System Drivers & Design**
- **Test & Test Equipment**
- **Process Integration, Devices, & Structures**
- **Front End Processes**
- **Lithography**
- **Interconnect**
- **Factory Integration**
- **Assembly & Packaging**
- **Environmental Safety & Health**
- **Yield Enhancement**
- **Metrology**
- **Modeling & Simulation**





# Types of Chip Technologies Discussed

---

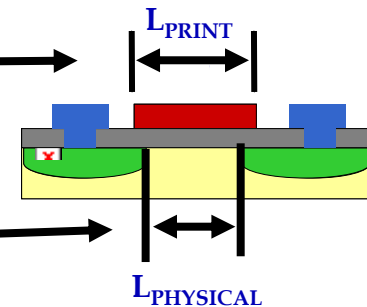
- **Logic:** high speed transistor, lots of metal layers
  - High Performance Microprocessors
  - Cost Performance Microprocessors
  - Low Power Microprocessors
  - ASICS (Application Specific ICs)
  - Also includes memory options: SRAM, Embedded DRAM
- **DRAM:** high threshold transistors, few metal, cheap fab processes
  - High Volume Commodity Dense memory part
  - Also includes Flash
- **Analog and Mixed Circuits**
- **Emerging Alternative Technologies**



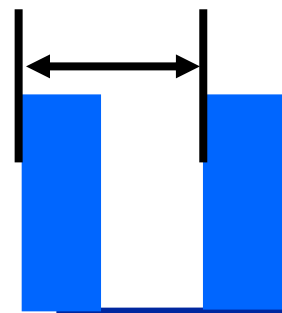
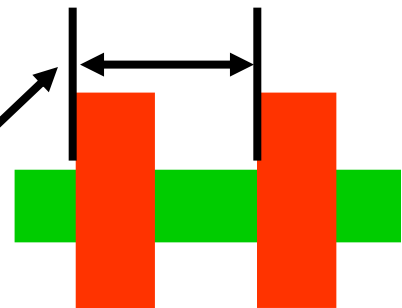


# Common Device Features to Track: (With values termed “Feature Sizes”)

- Gate length of a transistor gate “as printed”
- Gate length of a microprocessor transistor gate “as physically fabricated”



- $\frac{1}{2}$  of minimum pitch between two logic poly lines
- $\frac{1}{2}$  of minimum pitch between two DRAM metal lines







# Key Terms

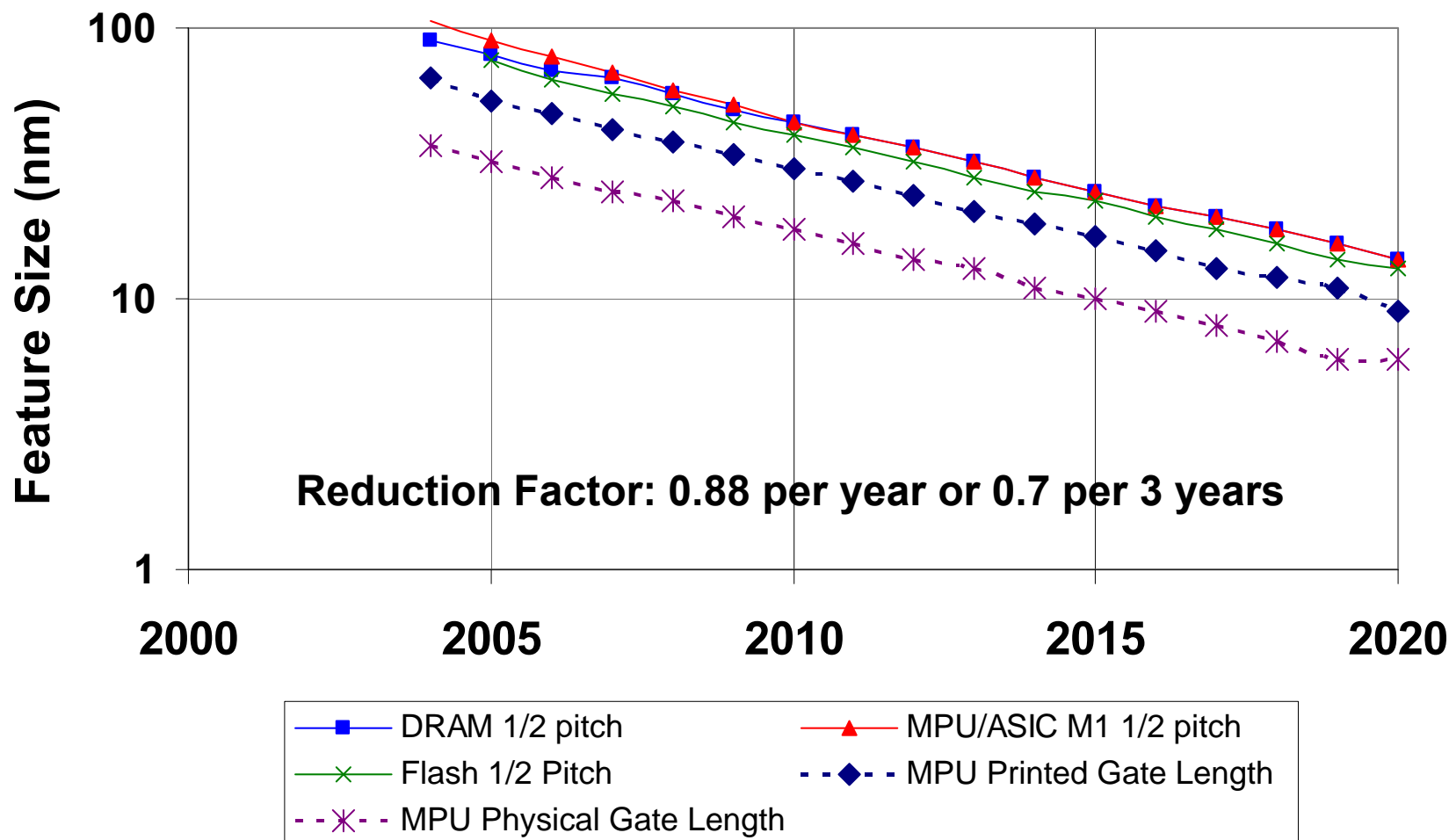
---

- **Technology Generation for Year X:**
  - Minimum feature size in any product in that year
- **Technology Node:**
  - Year in which *~4X growth* over prior Node
  - Typically tied to DRAM (usually smallest)
  - Based on *Year of Production*





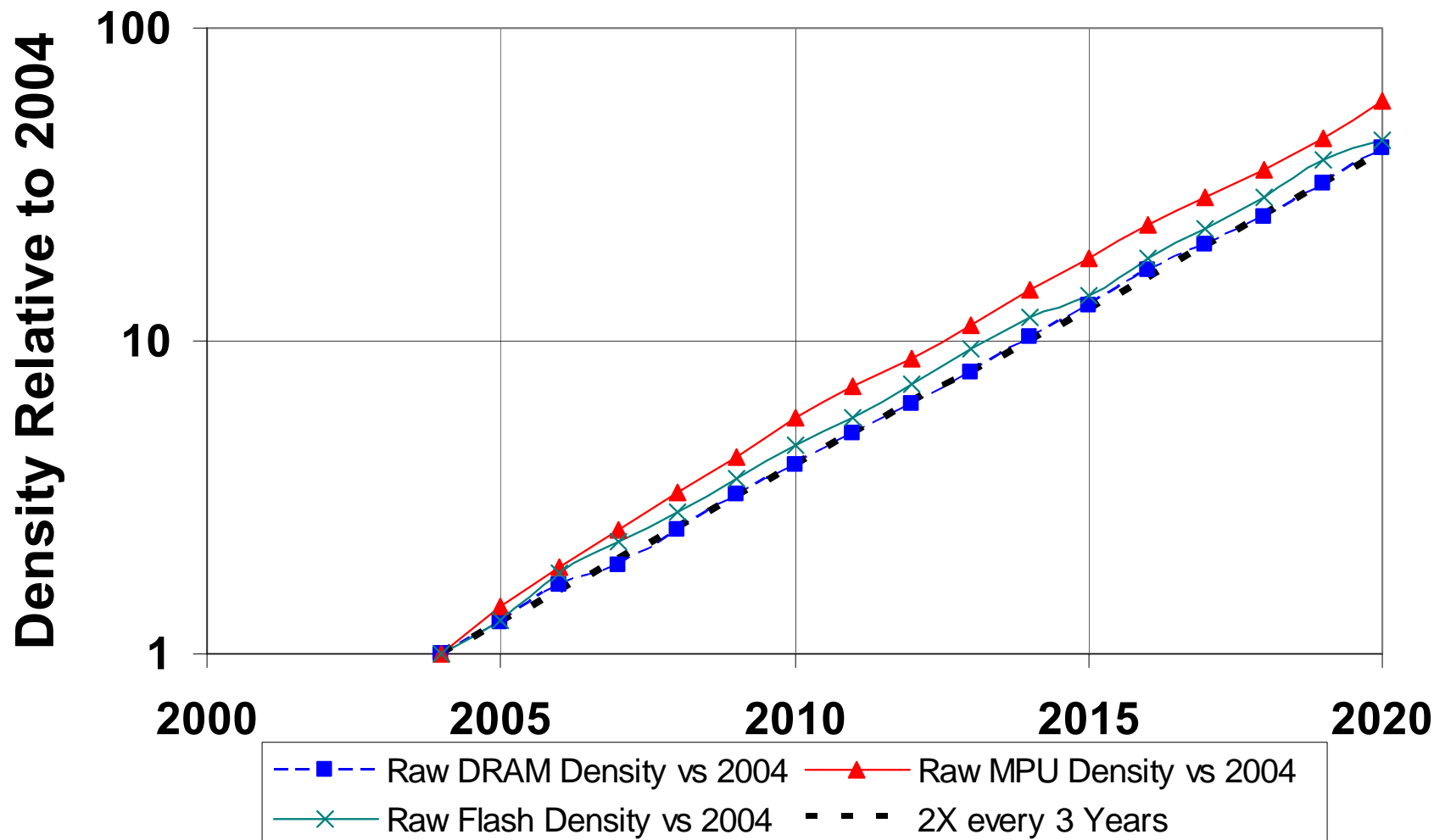
# Feature Size Projections







# Projected Density Growth ( $S^2$ )







# Comparison to Moore's Law

---

- **Moore's Law: ~4X functionality per 3 years**
- **But feature scaling provides only 2X**
- **Providing difference for microprocessors**
  - Clock frequency increase
  - More parallelism in CPU microarchitecture
- **Providing difference for DRAMs**
  - Denser cell design
  - Bigger die area
- **Both are reaching limits**





# Commodity DRAM Capacity

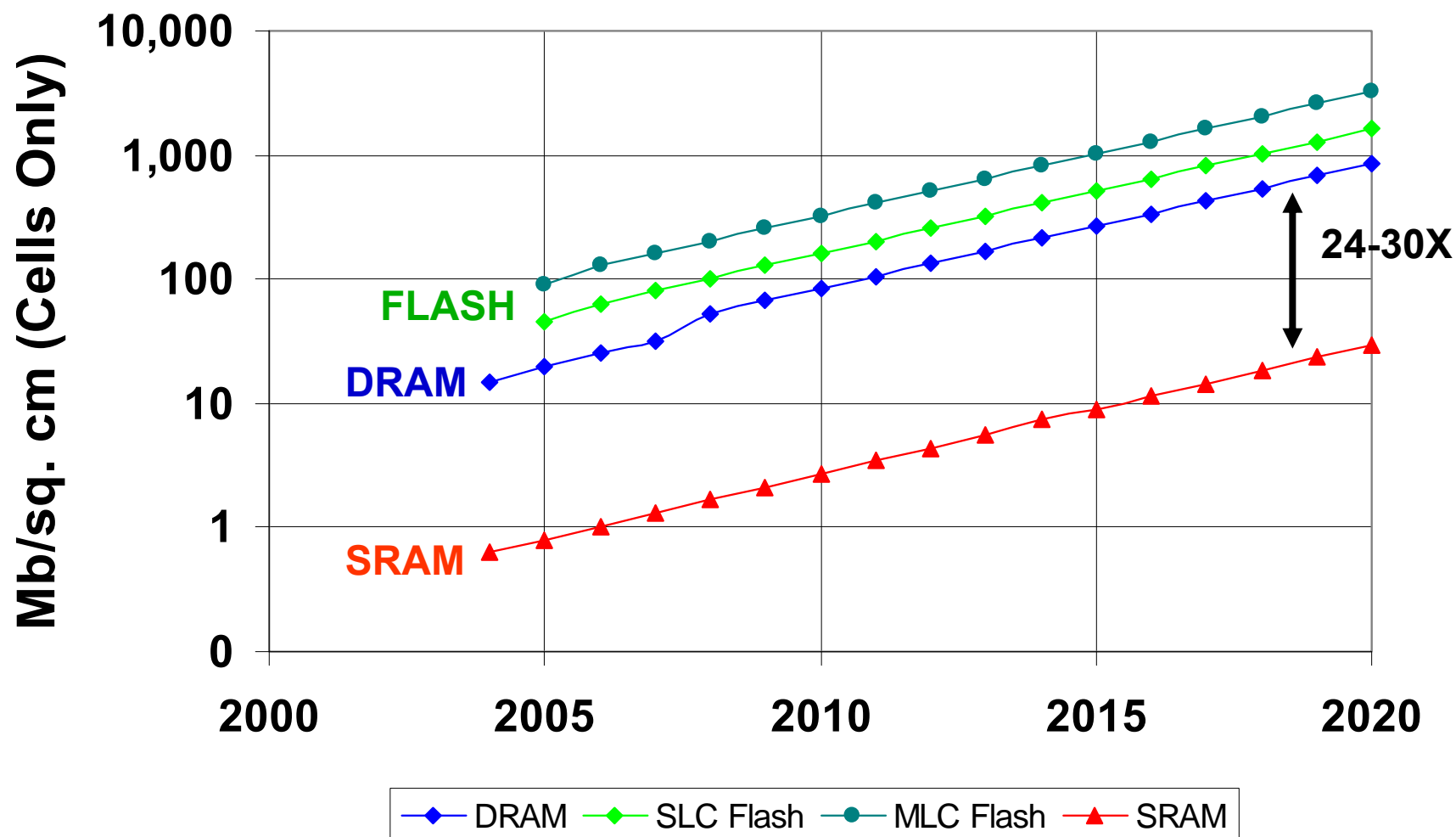
---

- **Cell Area:** area of one bit
  - Function of technology scaling & circuit features
- **Array area %:** % of chip that is cell
  - Constant at 63% in production
- **Chip Capacity:**
  - $(\text{Chip size} * \text{Array area \%}) / \text{Cell area}$
- **Chip Size:**
  - Initially increased to achieve Moore's Law
  - Now chosen to maximize yield





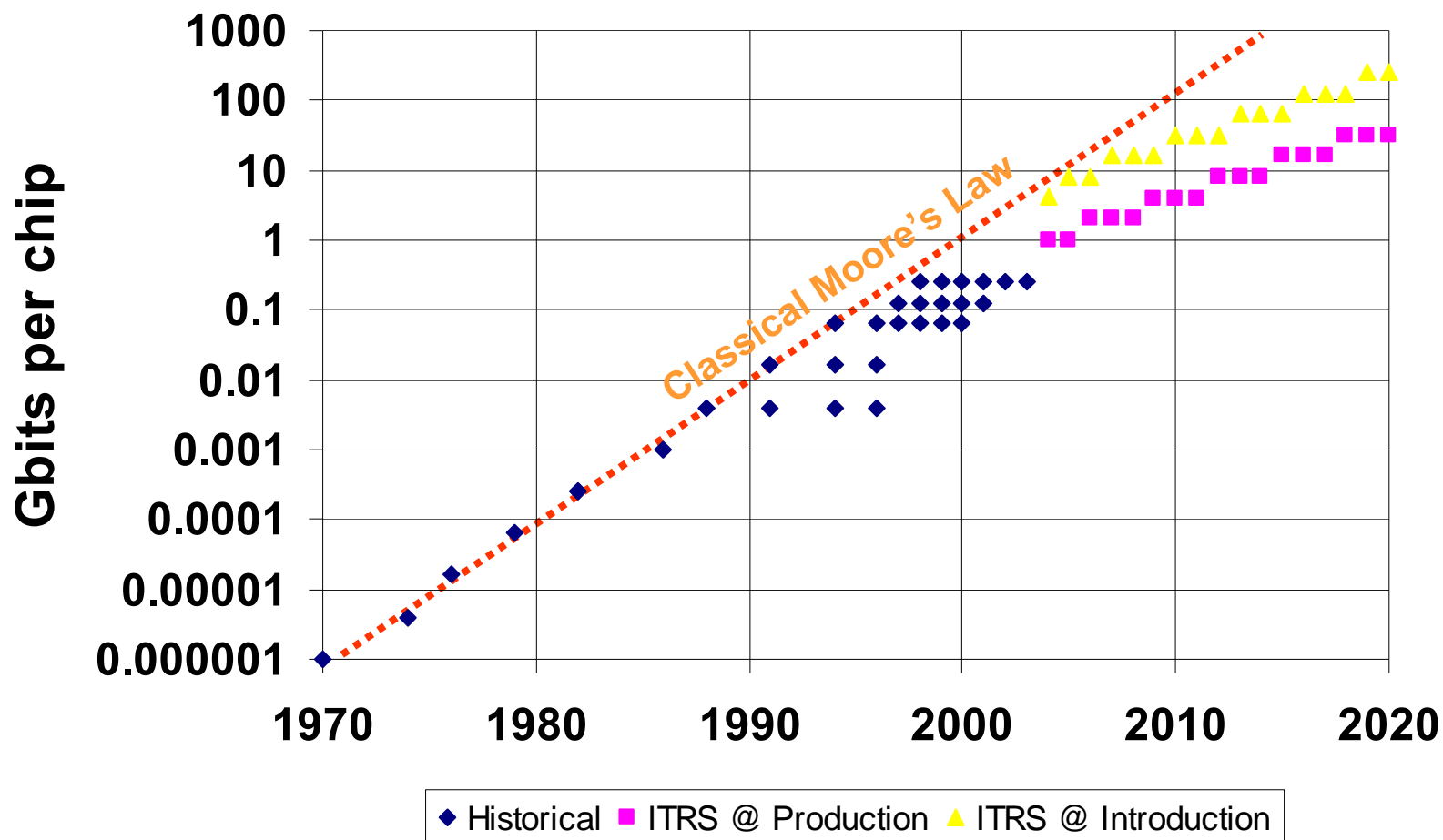
# Memory Density: Cells Only







# Chip Capacity

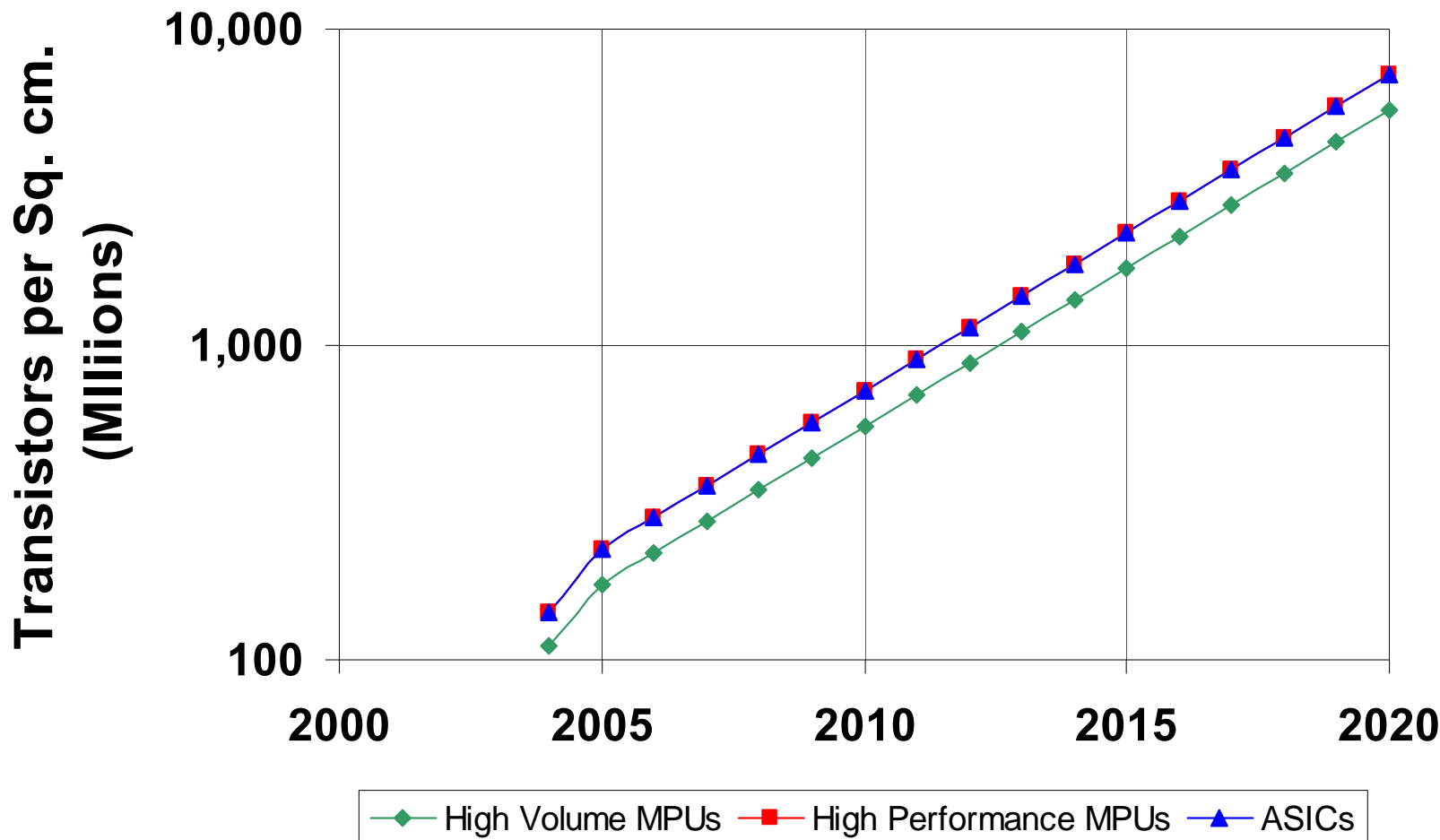


**Chip Capacity is No Longer Following Original Moore's Law**





# Logic Chip Density Scaling

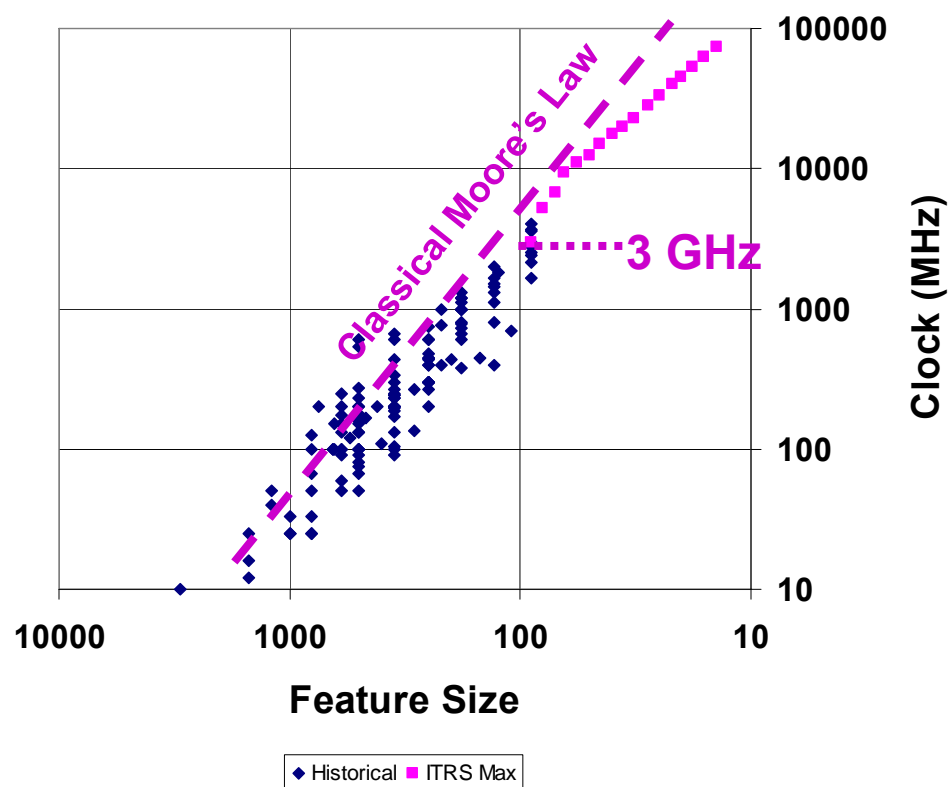
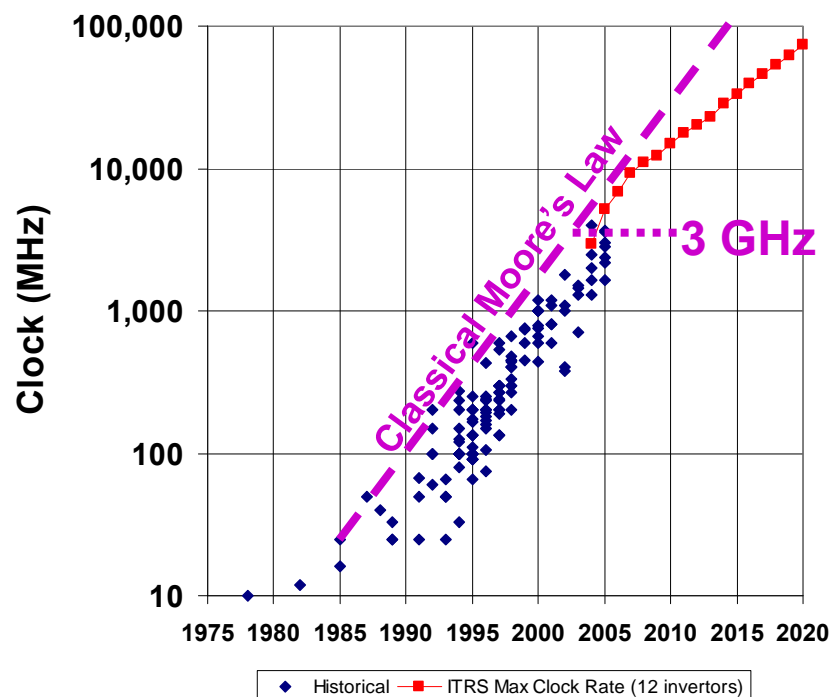


*Logic functions per unit area: ~2X every 3 years*





# Peak Logic Clock Rates

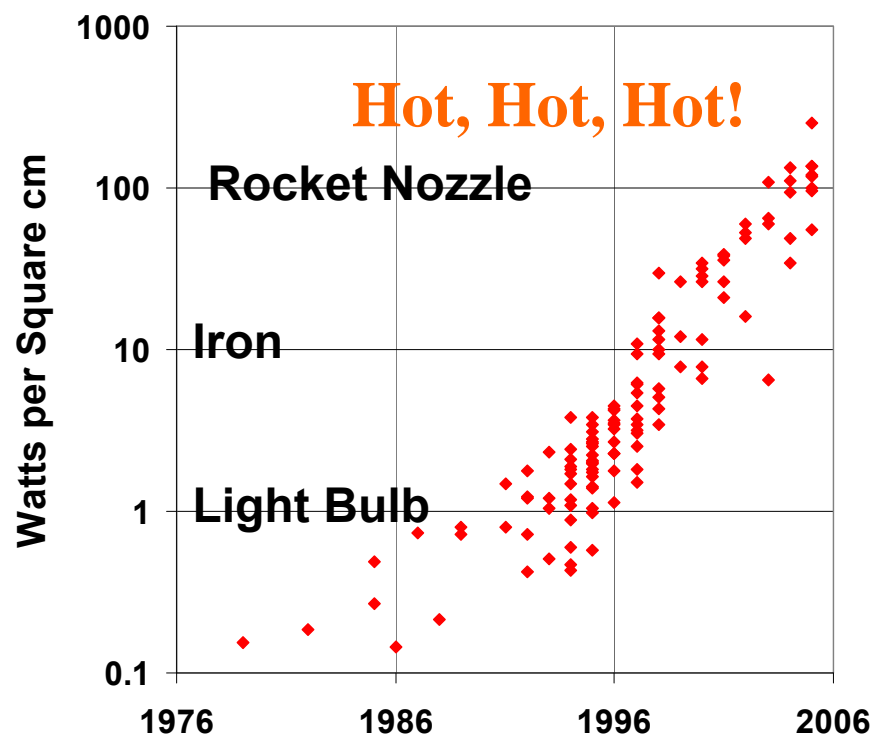
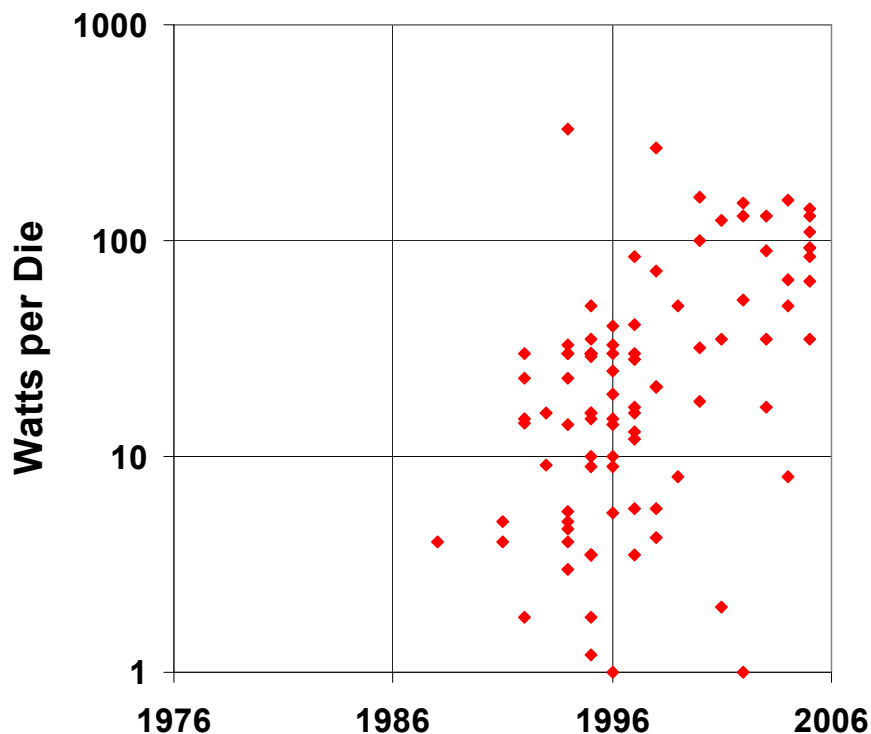


**2005 projection was for 5.2 GHz – and we didn't make it in production. Further, we're still stuck at 3+GHz in production.**





# Why the Clock Flattening? **POWER**







# The Power Equation

---

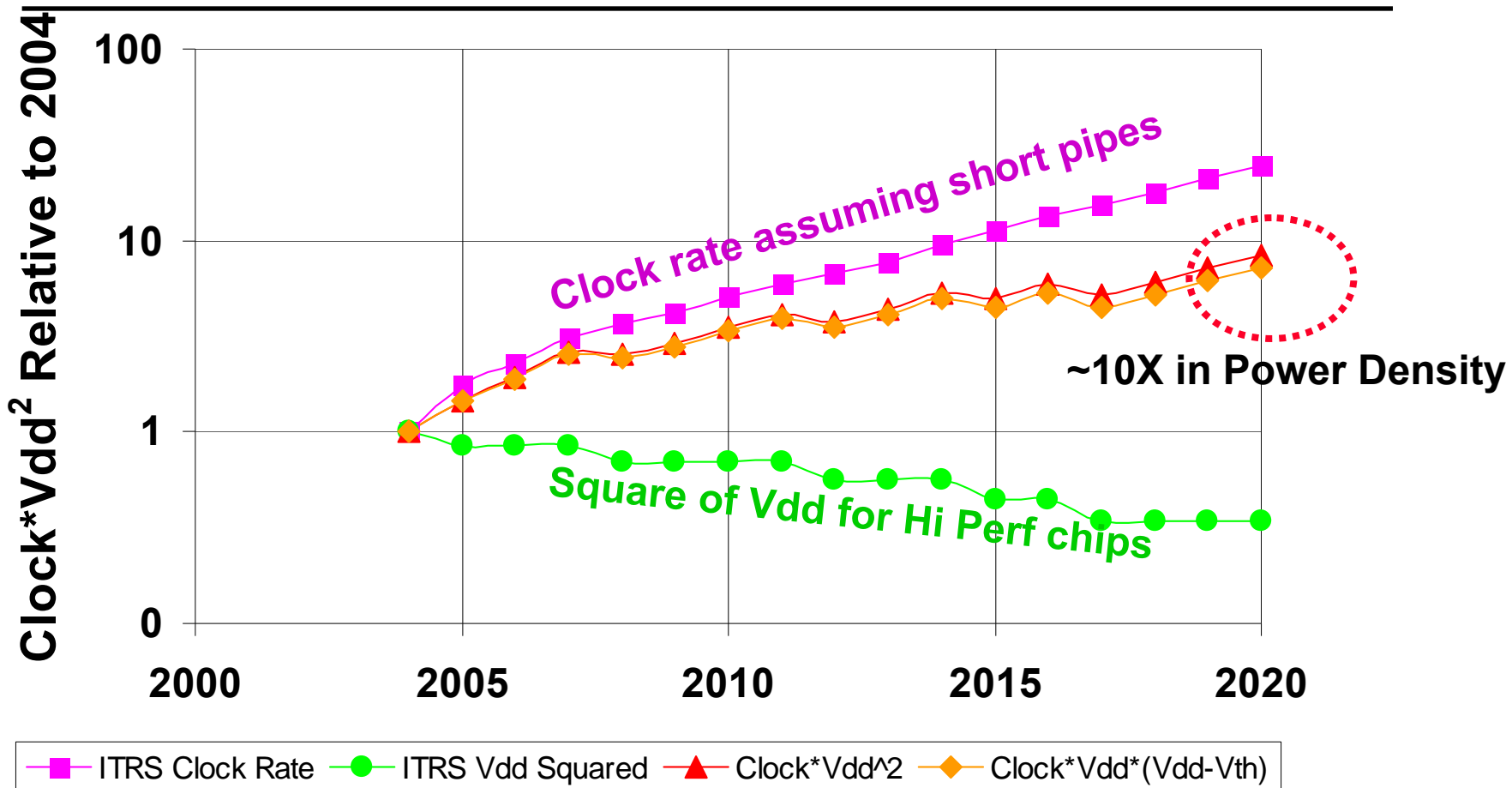
- Dissipated Power = Dynamic + Static
- Dynamic Power  $\sim CV^2FA$ 
  - $V = V_{dd}$
  - $F$  = Clock Rate
  - $A$  = Activity Rate = % of transistors that switch at each clock
  - $C$  = Effective capacitance switched at each clock
    - $\sim \underbrace{\text{\# of transistors switched}}_{\sim (1/S)^2 = \text{"Growing"}} \times \underbrace{\text{transistor gate capacitance}}_{\sim S^2 = \text{"Decreasing"}}$
- Static Power: leakage from each device
  - **GROWING** with # of devices

Approx.  
Constant





# ITRS-Based Power Density Increase

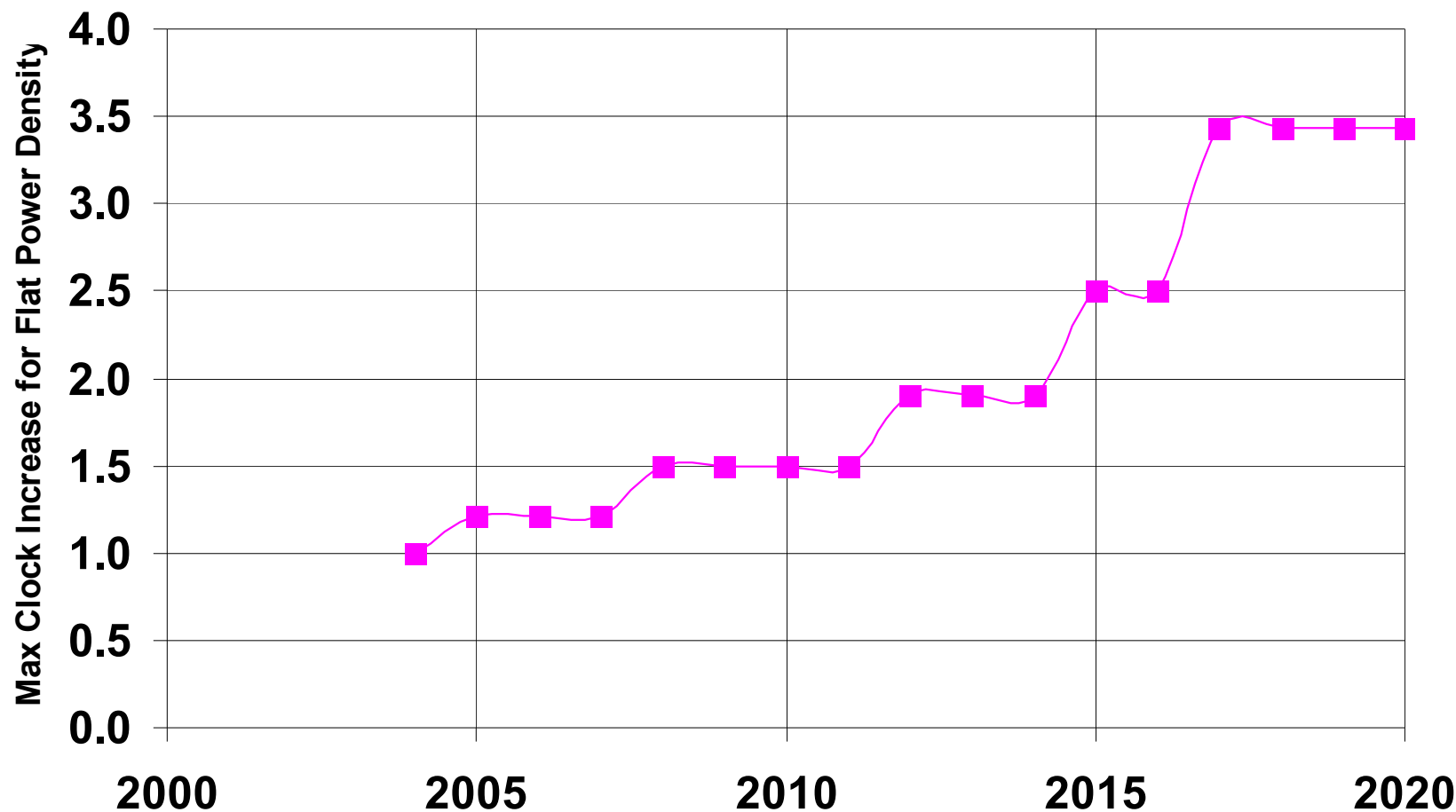


Transistors are getting faster faster than Vdd is declining





# Constraining Clock Rate for Flat Power Density



***And we haven't accounted for increase in static leakage power!!!***





# What Are Our Options?

---

- **Live with lower clock rates than technology allows?**
- **Use higher threshold transistors to lower static leakage power**
  - Still fits in lower clock regime
  - Possibly requires higher  $V_{dd}$
- **Decrease amount of “speculative execution”**
  - Eg. shorter pipes, less out-of-order
- **Lower the average number of transistors per unit area that switch per cycle**
  - Increase % of die that's memory





# Off Chip Bandwidth

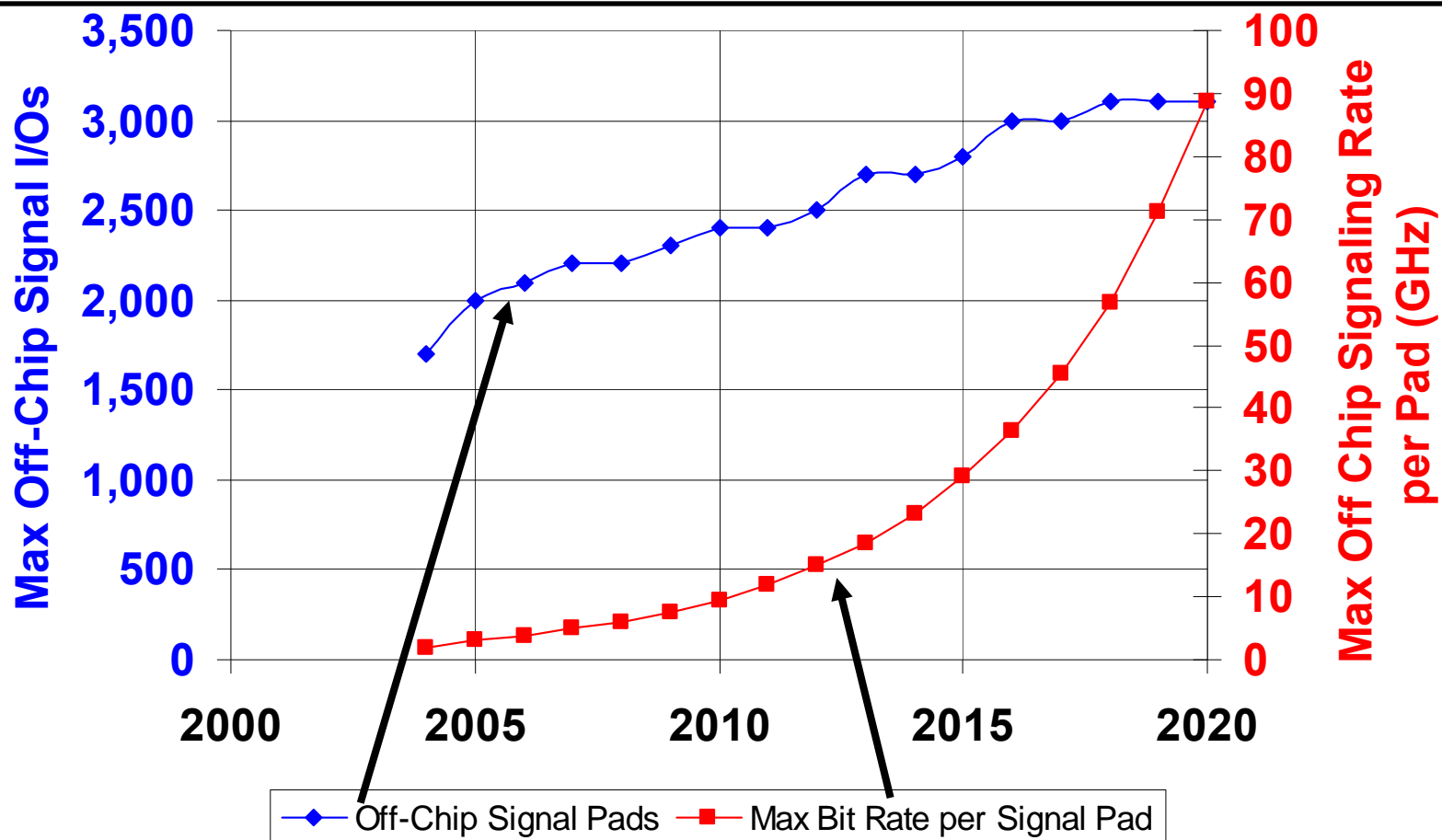
---

- **Today's Architectures: need to go off-chip for memory access**
  - And we don't have enough bandwidth today
- **Upper limit = product of:**
  - # of off-chip pins/contacts
  - % not used as power/ground
  - Max signaling rate per pin
- **Density & signal rate improve with time**
  - With 50% power/ground
  - But they don't match growth in performance potential





# Off-Chip Parameters

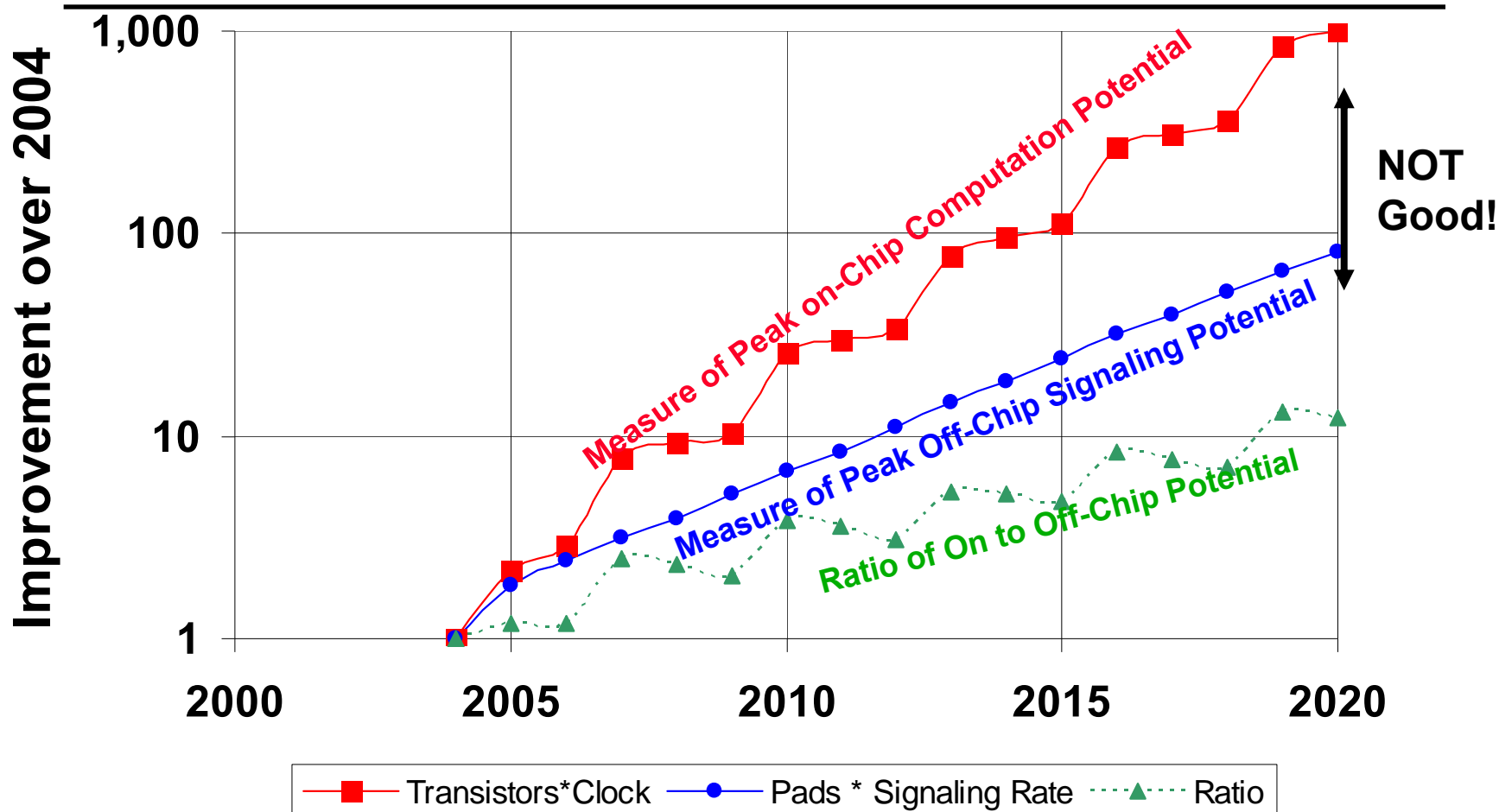


**Max off chip signaling rate is still exponential; but # of pads nearly flat!**





# Does Logic Performance Match Off-chip Bandwidth Potential?



Note: Today's Memory Chips cannot match today's Peak I/O Rates





# What Are Our Options for Bandwidth

---

- Place *much faster* interface logic on memory chips
  - And raise power and fabrication cost
- Add additional memory ports to MPU chips
  - And raise power and packaging
- Switch to narrow but very high speed memory channels
  - And require external memory controller chips



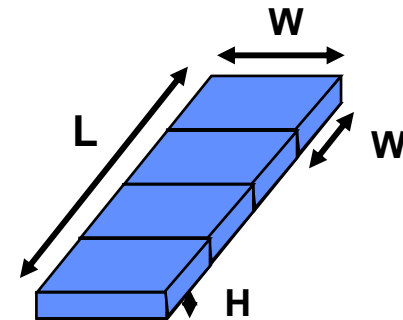


# On-Chip Wire Speed

## (Very Simplistic Approximation)

---

- $C \sim L \times W$
- $R \sim L / (W \times H)$
- Thus  $RC \sim L / H$
- **Scenario #1: L scales with technology**
  - Such as inside a core that shrinks in size
  - And so does W, H
  - Then  $RC \sim$  same (no scaling with clock)
- **Scenario #2: L is constant**
  - As in crossing a die of a fixed size
  - Then RC **goes up** as H shrinks
- **Conclusion: on-die interconnect getting slower!!!**
- ***AND THIS IS ONLY A SIMPLE APPROXIMATION!***



See for example: Banerjee, et al, "Interconnect Modeling and Analysis in the Nanometer Era: Cu and Beyond,"  
22nd Advanced Metallization Conference





---

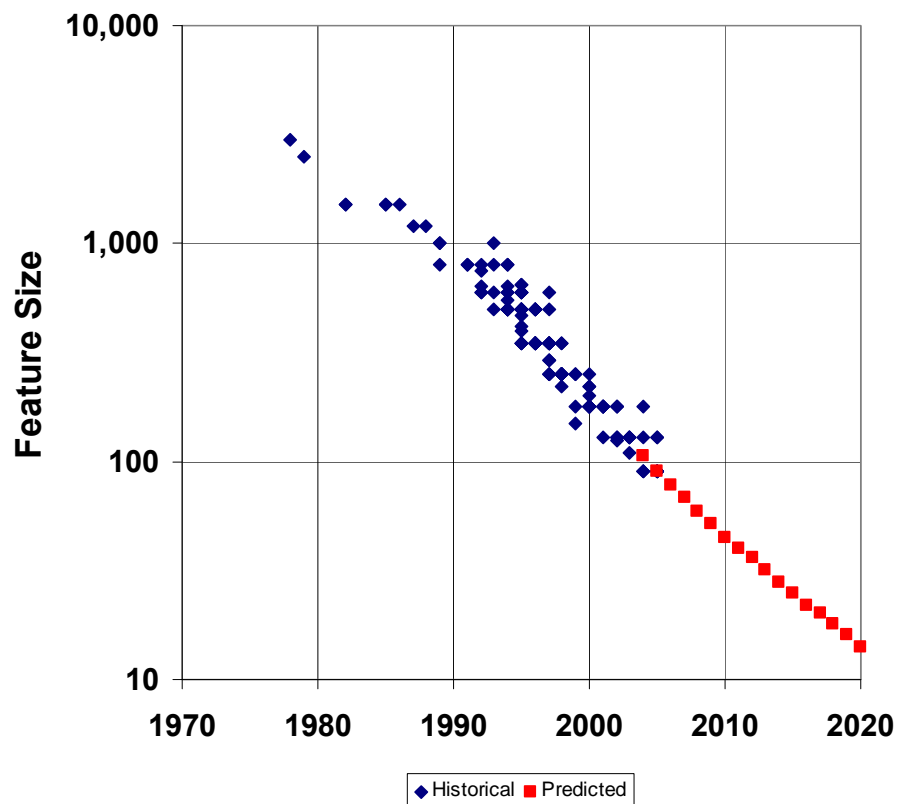
# **The Way We Were: A Brief Romp Thru Single Core Microprocessor Land**

- **Data from last 30 years of real chips**

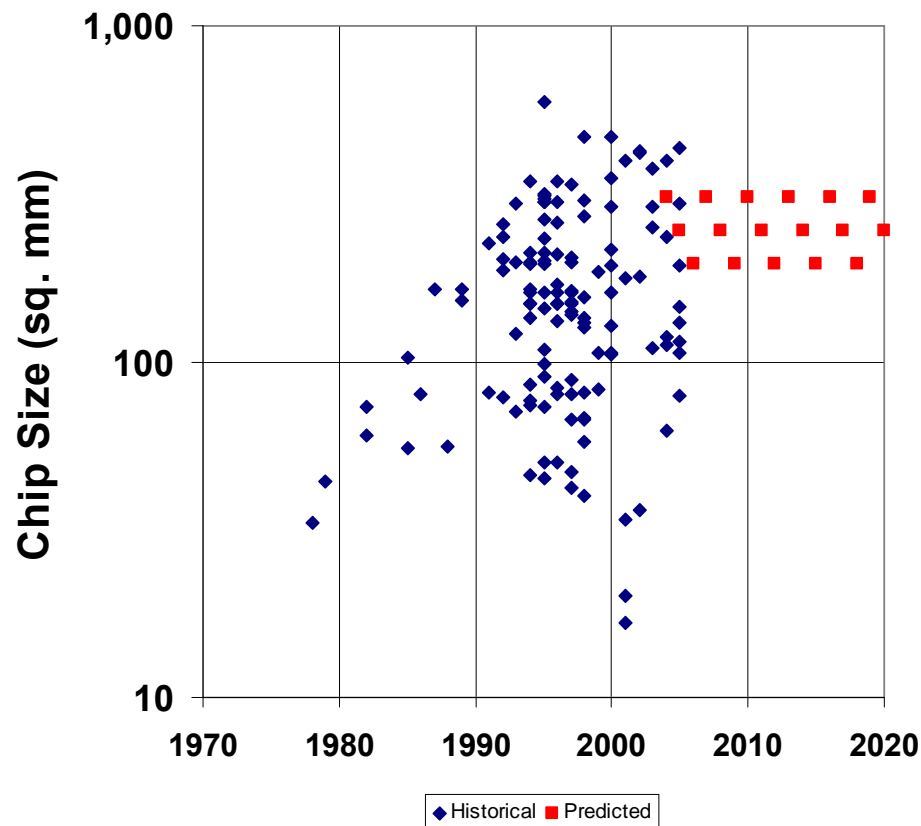




# Historical Changes in Single-Core MPU Parameters



**Moore's Law Advances in Feature Size**

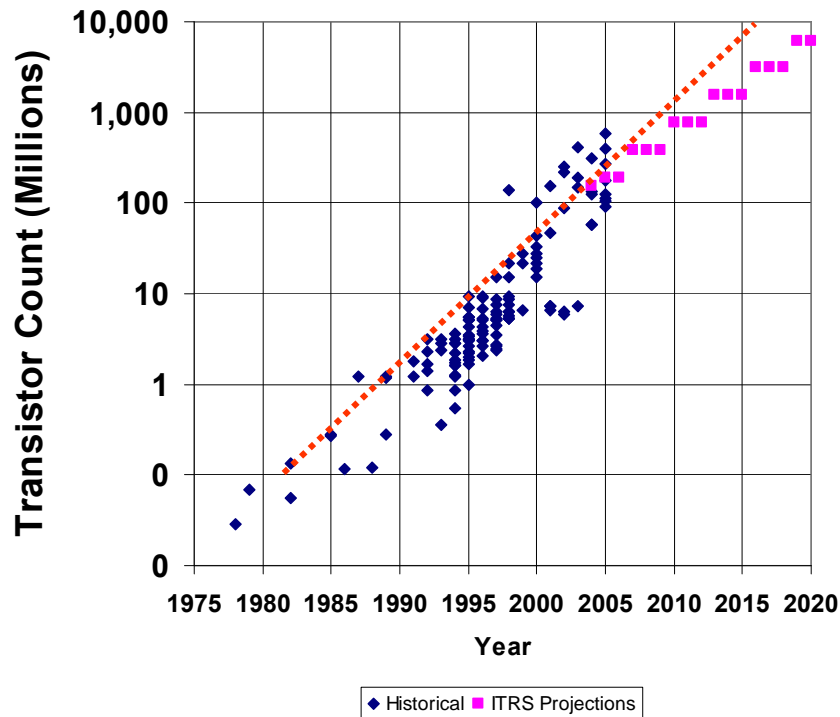


**But Chip Size is Flat**

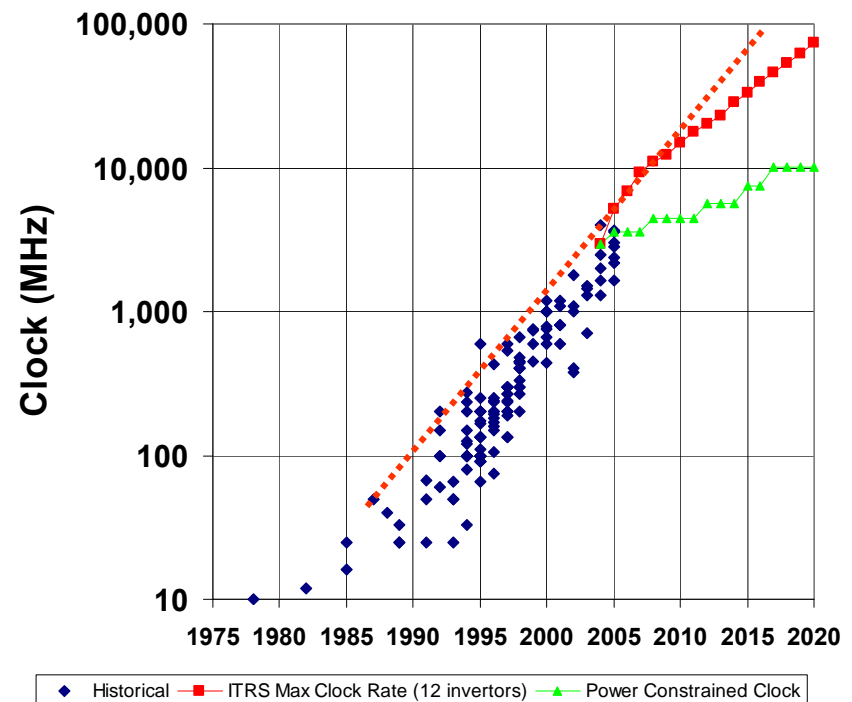




# Functionality



- ~ 4X per die every 3 years
- But: Most in cache
- And partially due to larger die
- And off-chip clock rates lagging

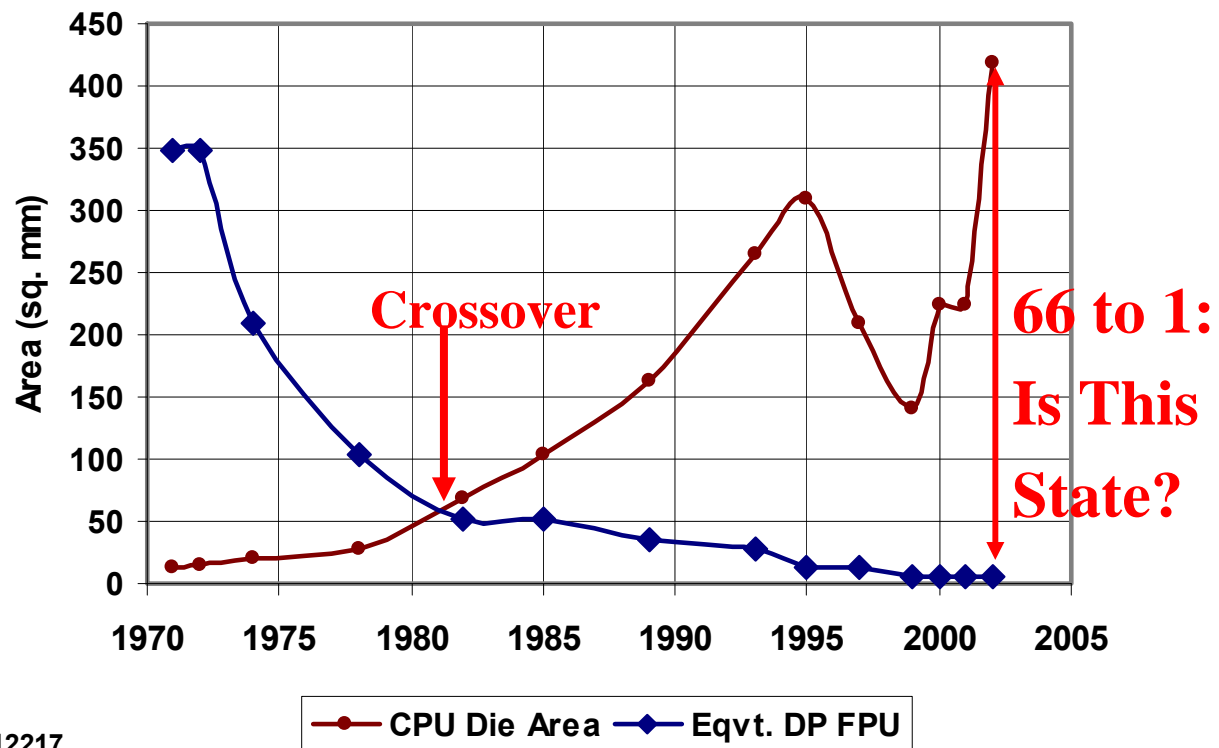
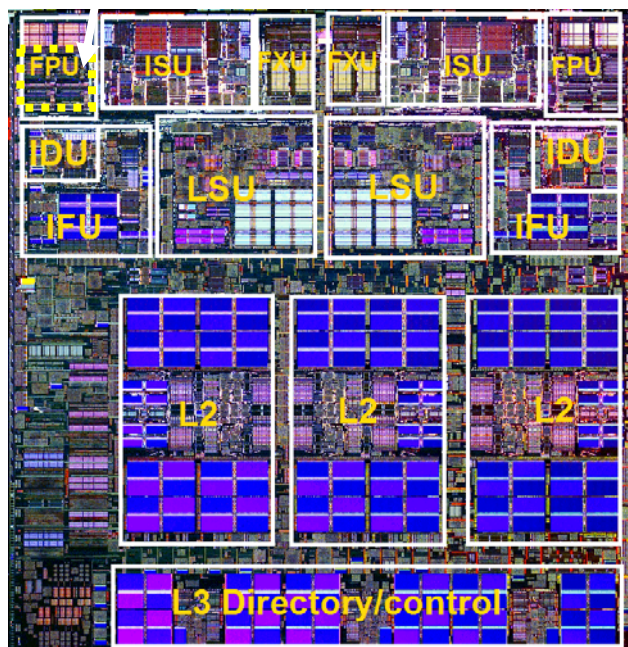


- Historically ~ 2.3X every 3 years
- But: increasing clock increases memory wall
- And clock rates stagnating





# How Are We Using These Transistors

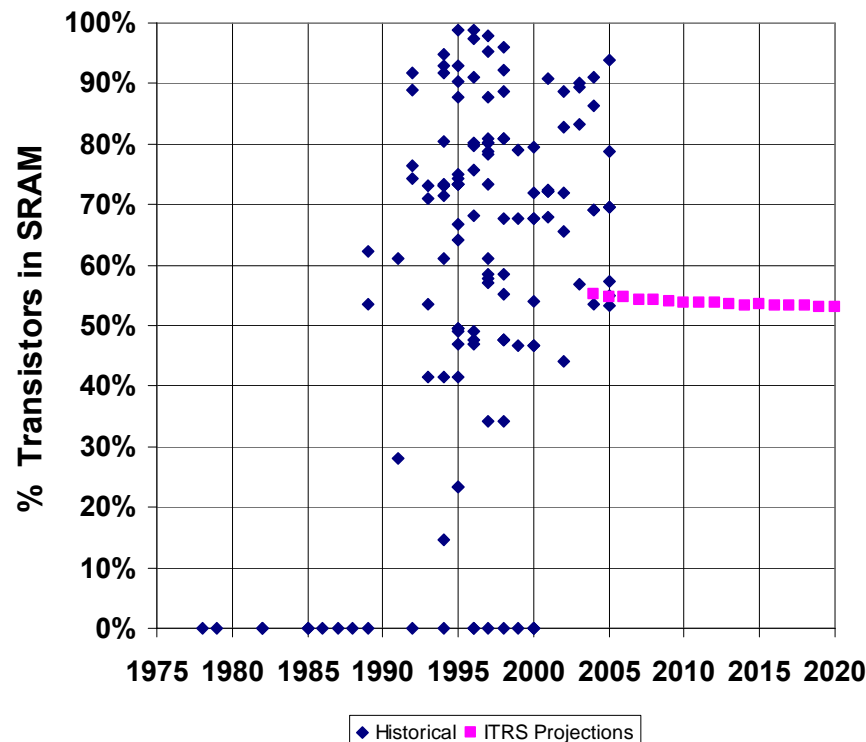
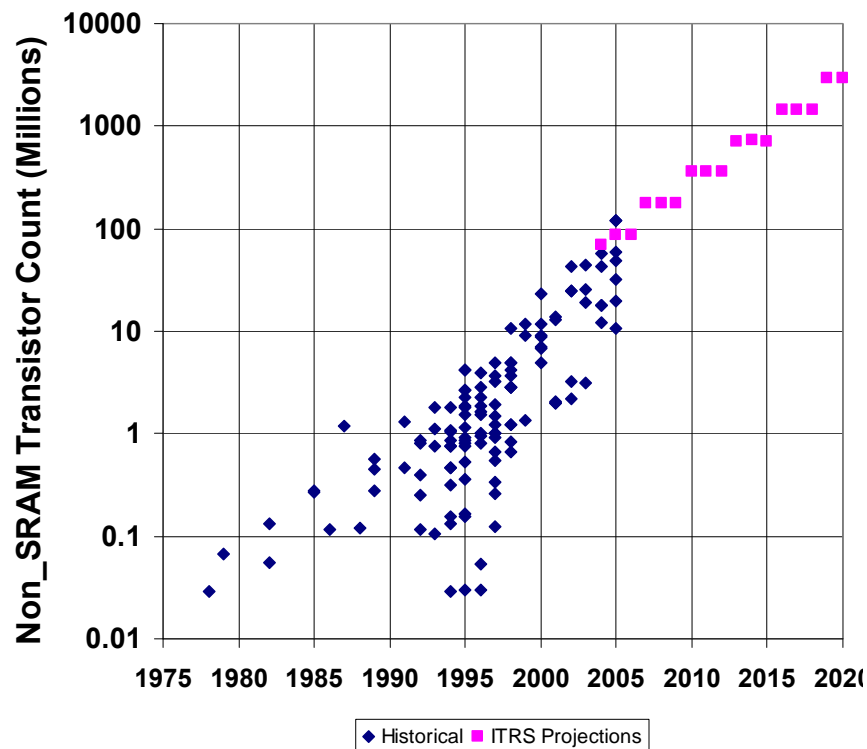


See <http://www.theinquirer.net/default.aspx?article=12217>





# Let's Look at Transistor Usage

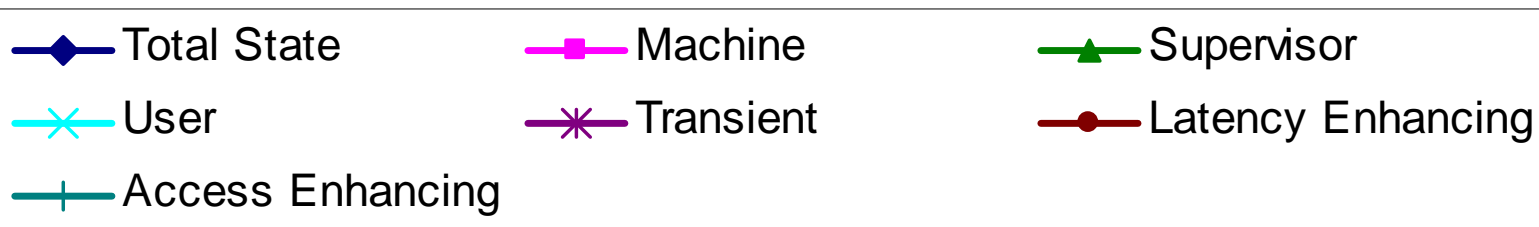
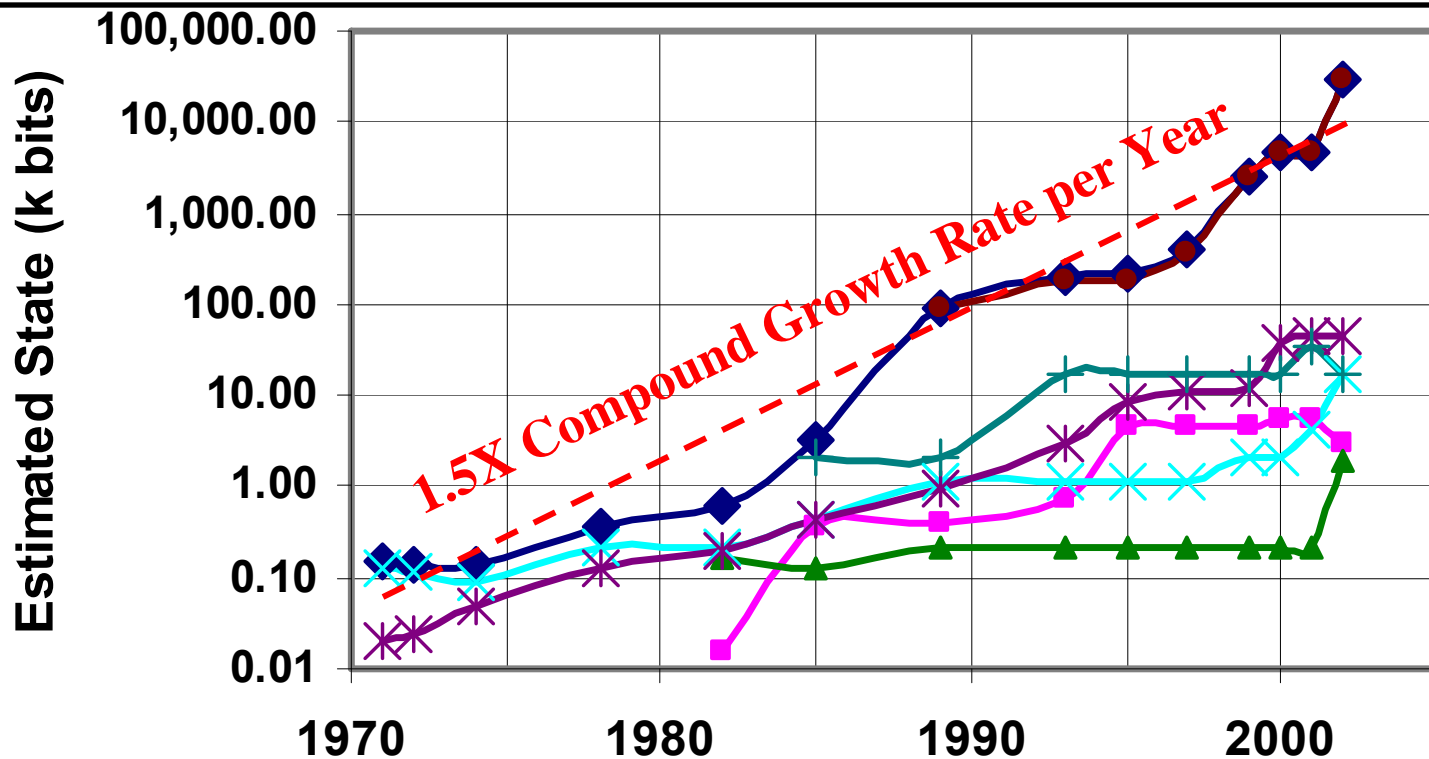


**Most of Microprocessor's Transistors are for cache  
– and forward estimates are below historical**





# Core CPU State vs Time







---

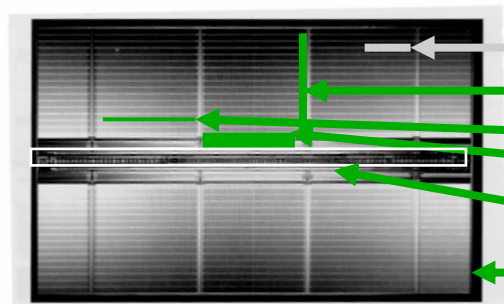
## **The Way We Were: A Brief Romp Thru Memory Land**

- **Data from last 30 years of real chips**

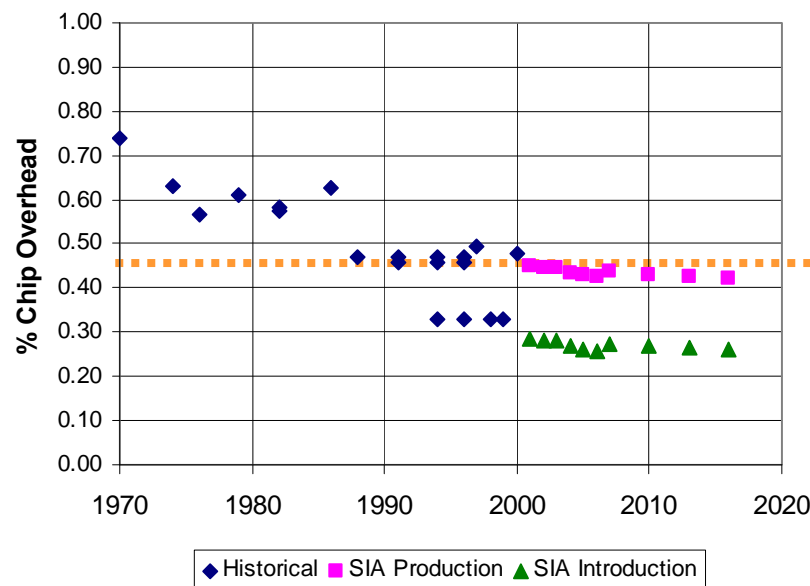
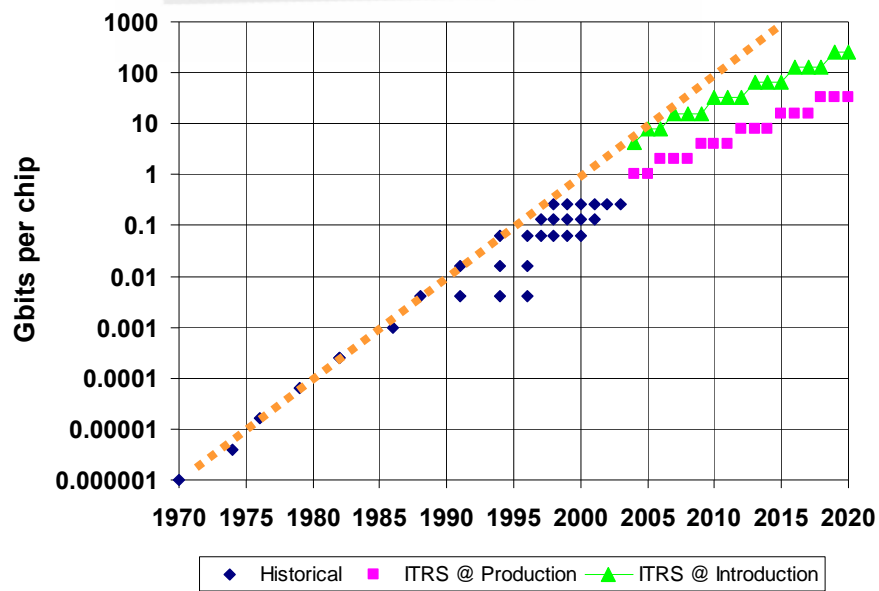




# Classical DRAM



- Row Decoders
- Primary Sense Amps
- Secondary sense amps & “page” multiplexing
- Timing, BIST, Interface
- Kerf



Density/Chip has dropped below 4X/3yrs

And 45% of Die is *Non-Memory*





# Basic Memory Operations

---

## Read:

- Send *Row Address* to chip
- Start row access in memory array
  - This results in up to 2048 bits read into “sense amps,” “row buffers,” ...
- Send *Column Address* to chip
  - This selects small (4, 8, 16) # of bits from previously read row for off-chip

Memory Latency

## Page Mode:

- Continue with multiple Column Addresses

## Refresh:

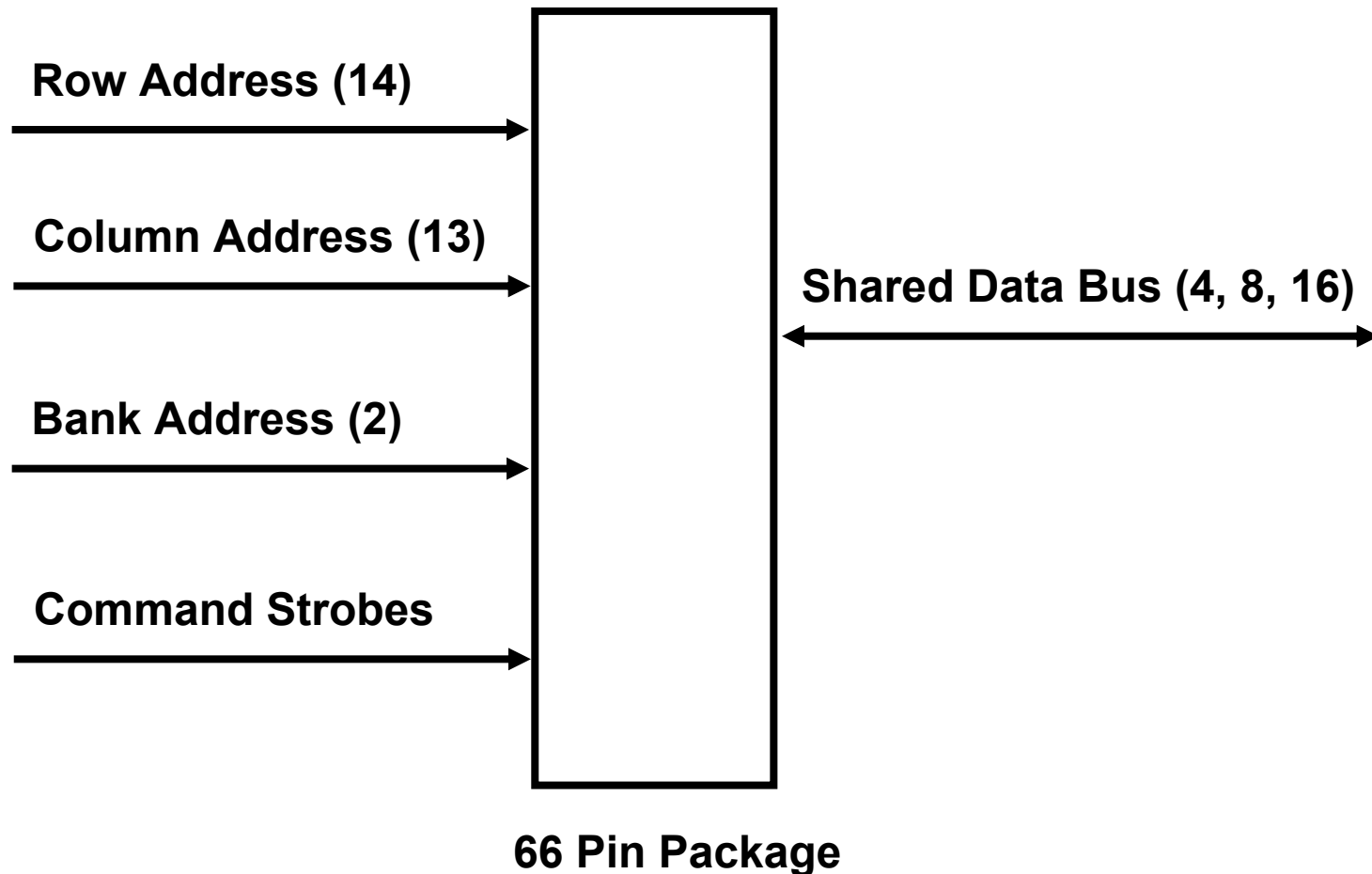
- Write current row back into memory array





# Conventional DRAM Part PinOut

---







# Chip-Level Memory Bandwidth

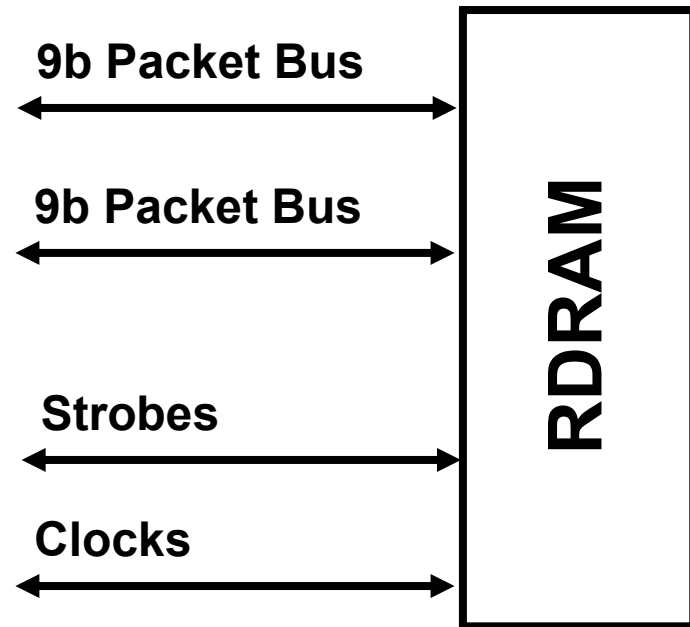
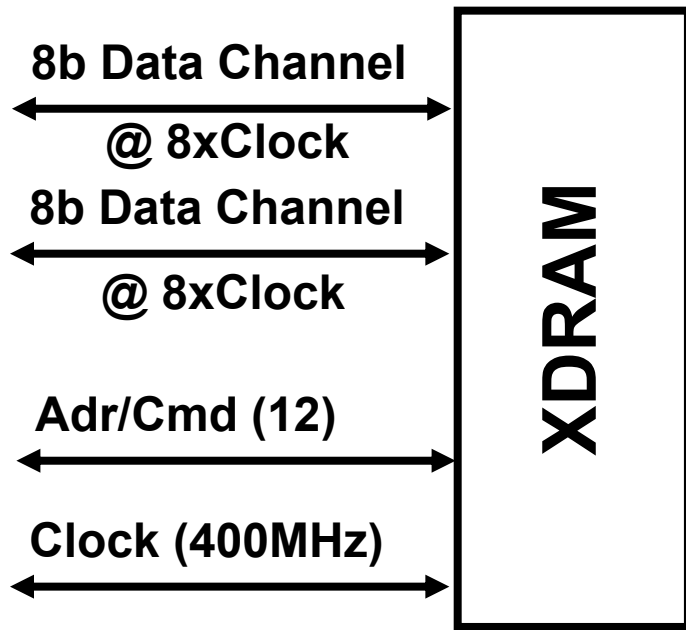
---

- **Memory Bandwidth:** Bits per second that move across chip
- **Early parts: *Unbuffered*:** One transfer per memory latency
  - Separate address/command/data pins
- **Improvements:**
  - Pipeline different accesses
    - Fast Page Mode, Synchronous
  - Include multiple independent banks within chip
    - DDR: up to 4; RDRAM: up to 32
  - Run interface at higher clock rate channel
  - Point to point synchronous data channels
    - XDR
  - Multiple data transfers per clock
    - Double Data Rate (DDR); XDRAM: 8 transfers per clock
  - Change from parallel to serial packet protocols
    - RDRAM
- **State of Art:**
  - DDR2-800: 2 transfers/clock x 400MHz x {4, 8, 16b}  $\leq 1.6\text{GB/s}$
  - XDR: 2 channels x 3.2GHz x 8b/channel  $\leq 6.4\text{GB/s}$
  - RDRAM: 2 x 8b x 1.6GHz  $\leq 3.2\text{GB/s}$





# Alternative Chip Interfaces

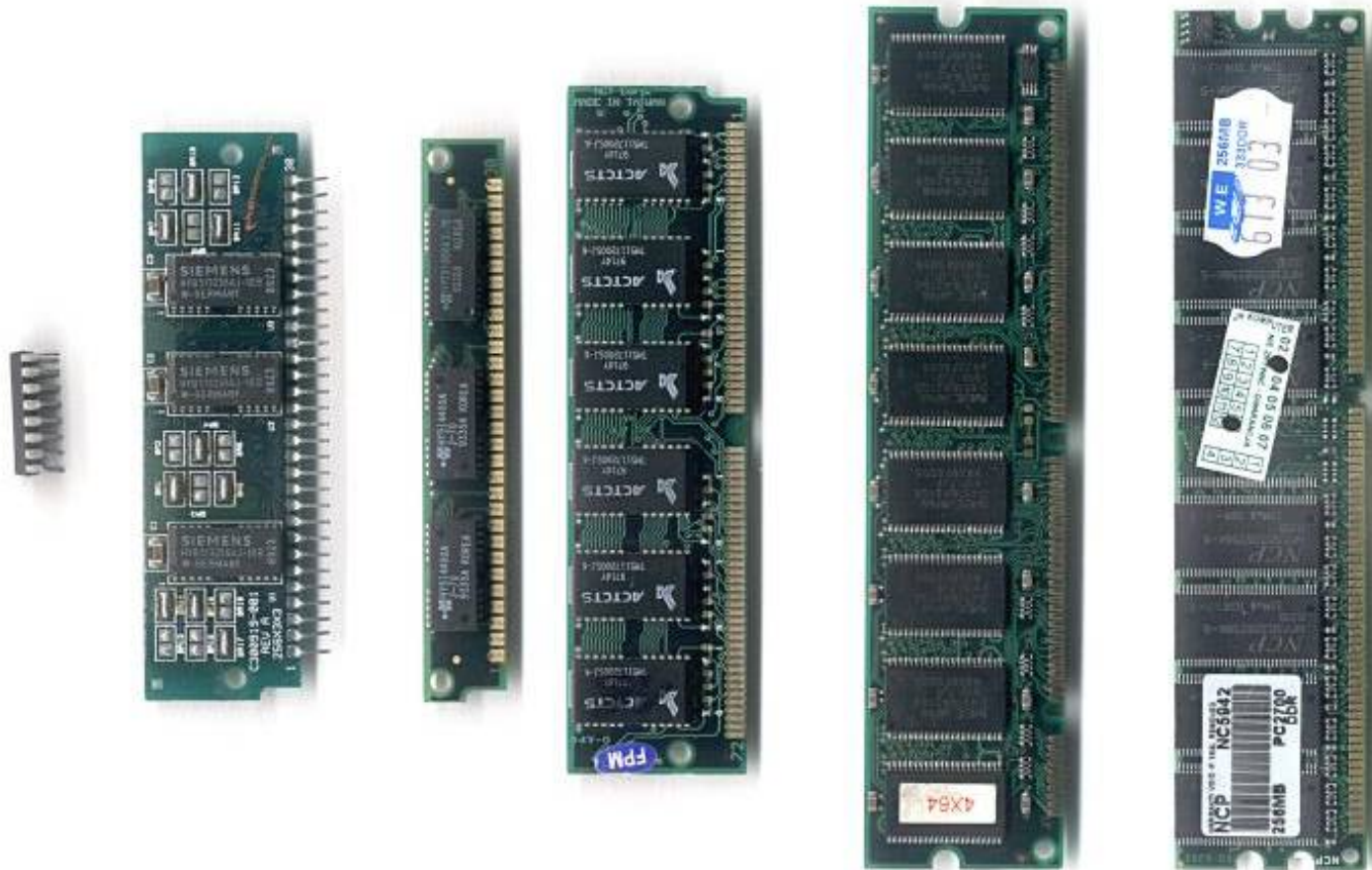


Commands & Addresses  
Serialized into Split Packets





# Packaging Multiple Memory Chips

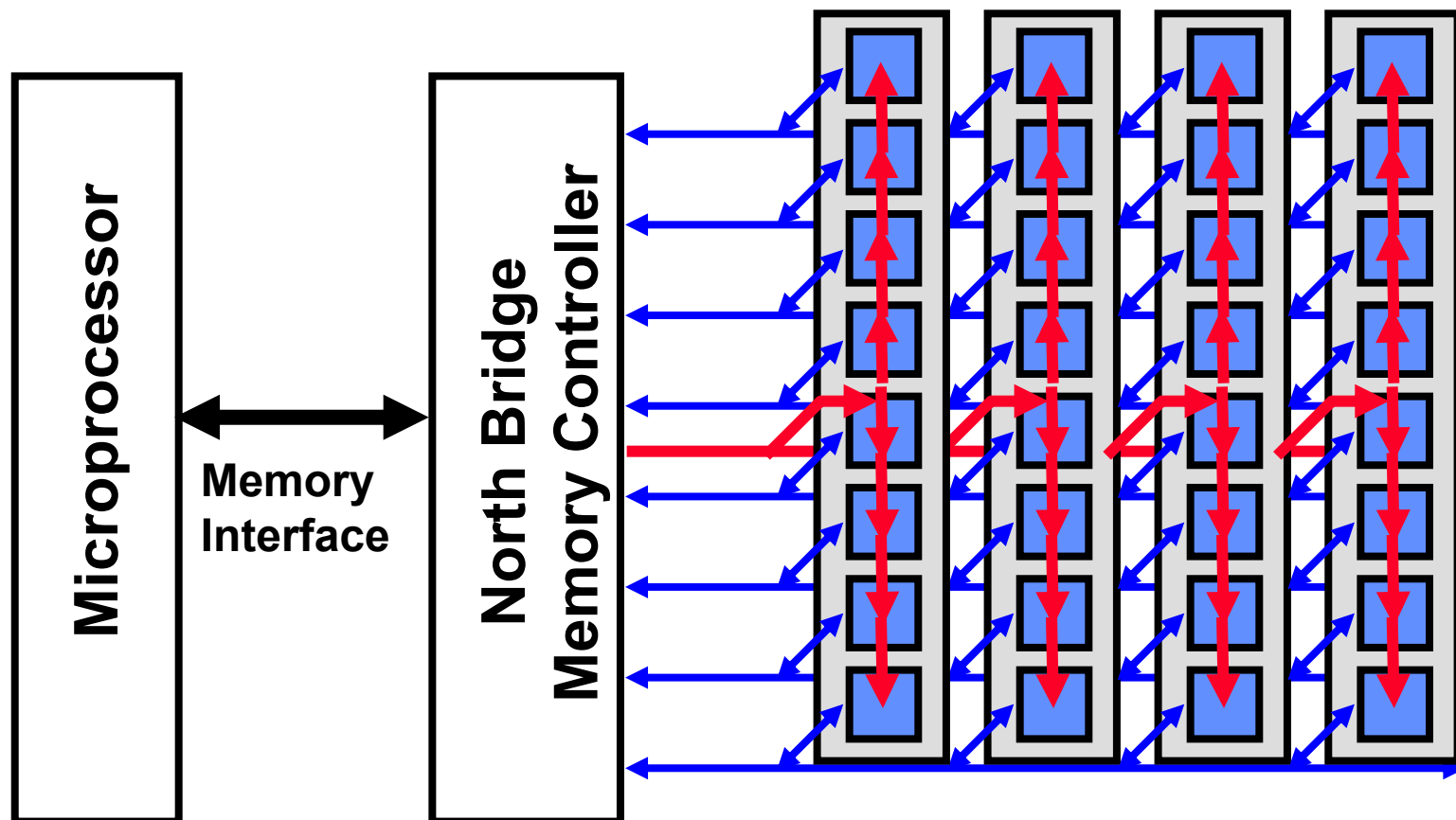


[http://upload.wikimedia.org/wikipedia/commons/d/d3/RAM\\_n.jpg](http://upload.wikimedia.org/wikipedia/commons/d/d3/RAM_n.jpg)





# Typical Electrical Configuration



State of the Art Peak Aggregate Bandwidth: ~ 6.4 GB/s





# Controller

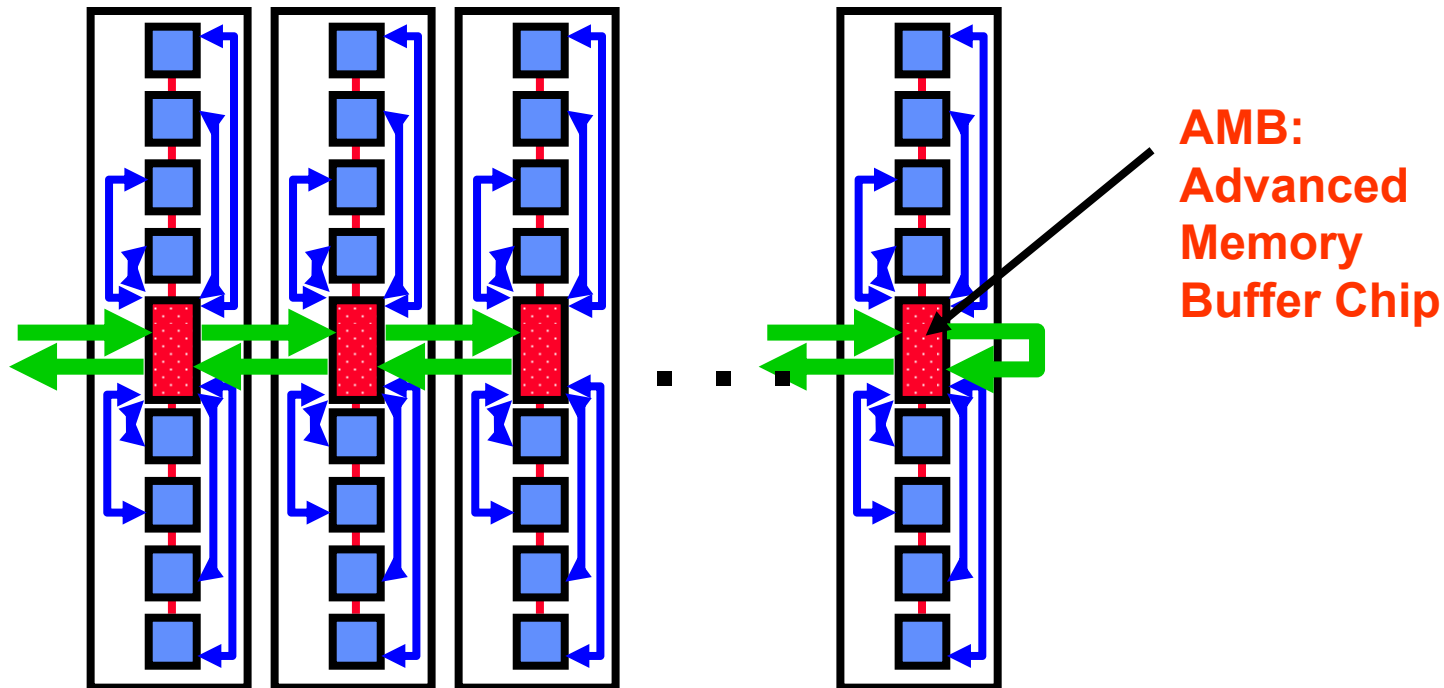
---

- **Functions:**
  - Aggregate bandwidth from many separate chips
  - Convert parallel memory ops from CPU to specific memory chip timing
  - And reassembling data from memory for relay to CPU
  - Handle refresh cycling of memory
  - Parity generation and/or checking
  - Southbridge connection to I/O interfaces
- **Until recently, a separate chip:**
  - Adds latency
  - But allows same MPU to use different memory types
- **Additional functionality:**
  - Interleave different requests to better utilize memory
  - More pipelining to increase pipelining
  - Multiple memory interfaces for concurrent memory bands





# A New Alternative: RL DRAM (Reduced Latency DRAM)



- Enter requests 1/cycle
- Request traverse to correct card
- Correct card starts operation, and forwards nop
- At end of line, “empty slot” returned other way
- When empty slot reaches card, replaced by data





---

## **Our Brave New World: Adding More Threads to a Single Die**

- **Multi-Threading**
- **Multi-Core**





# Technology Trends Forcing Parallelism

---

- ITRS predictions
  - Growing chip density
  - Power becoming paramount
  - Single core complexity becoming overwhelming
- Result: Classical Single thread performance flattening
- Answer: Relentless *Parallelism*:
  - Break program into independent *threads*
  - *Chip-level Multi-processing (CMP)*: multiple cores on same die
  - *Multi-thread parallelism*: executing multiple threads on same core (“virtual multi-core”)
- Both are possible – on same die





# Performance Gains from Explicit Parallelism

---

- ***Application speedup***: run all threads for one application execution at same time
  - Ideal speedup from  $N$  concurrent threads =  $N$
  - Limited by Amdahl's Law
- ***Throughput increase***: pipeline execution of different data sets through  $N$  steps/cores
  - Ideal throughput increase =  $N$
  - Limited by pipelining effects
- Different multi-core architectures emphasize different performance metrics





# Multi-Threading

---

- **Thread**: execution of a series of inter-dependent instructions in support of a single program
- Today's single threaded CPUs
  - Dependencies reduce ability to keep function units busy
  - Limited support for memory operations “in flight”
- **Multi-threading**: allowing multiple threads to take turns using same CPU logic
  - Typical requirement: multiple register sets
- Variations:
  - **Coarse-grained MT**: Change thread only at some major event
  - **Fine grained MT**: Change thread every few instructions
  - **Simultaneous MT**: interleave instructions from multiple threads





# MT Advantages

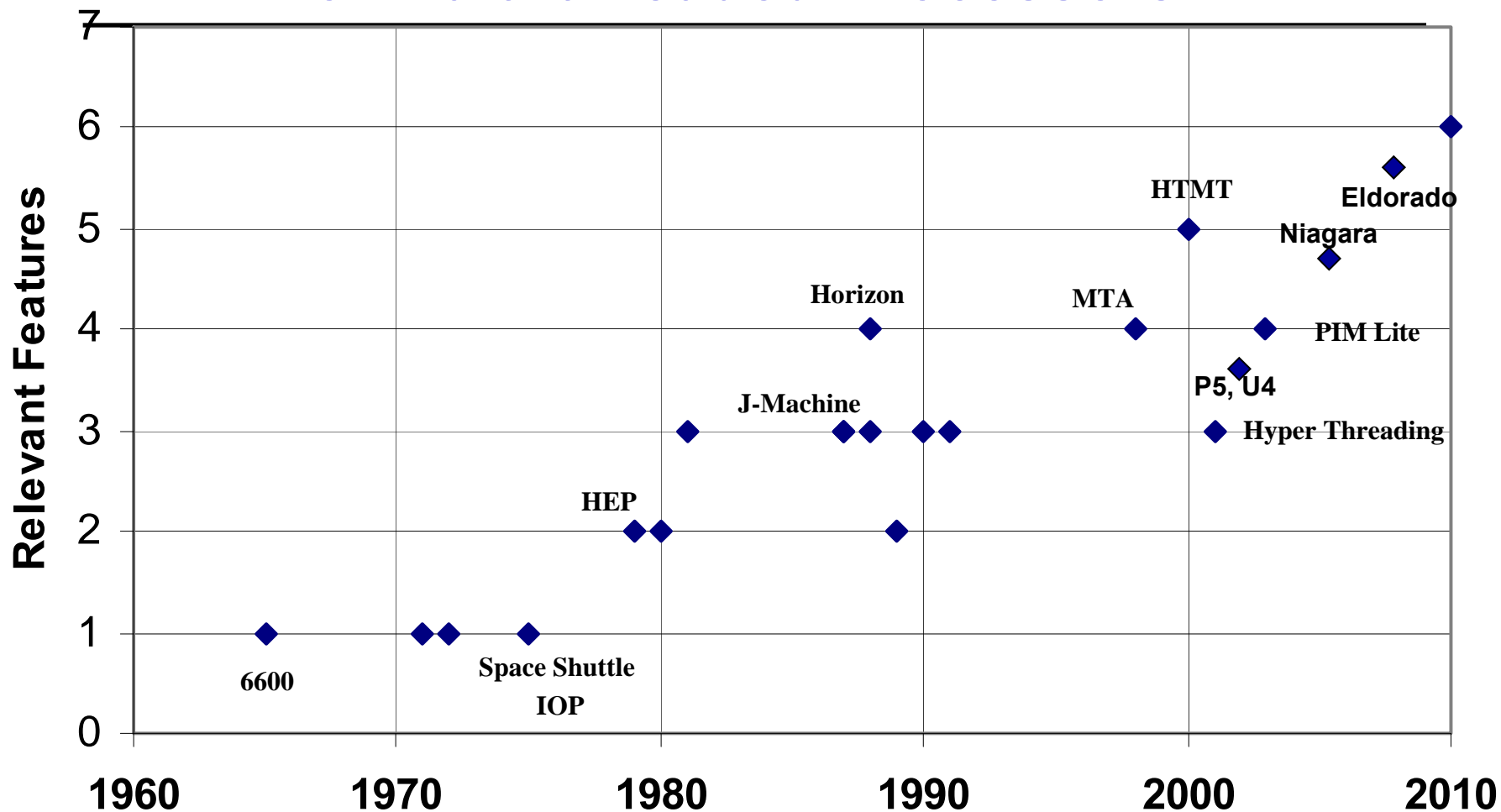
---

- **Hide long-latency memory operations**
- **Larger pool of unrelated instructions to feed function units**
- **Simplify scheduling of multiple activities**
- **In SMT designs: guaranteed independent instructions in pipelines eliminates need for expensive forwarding and reordering**





# A Brief History of Multi-threaded Processors







# Multi-Core

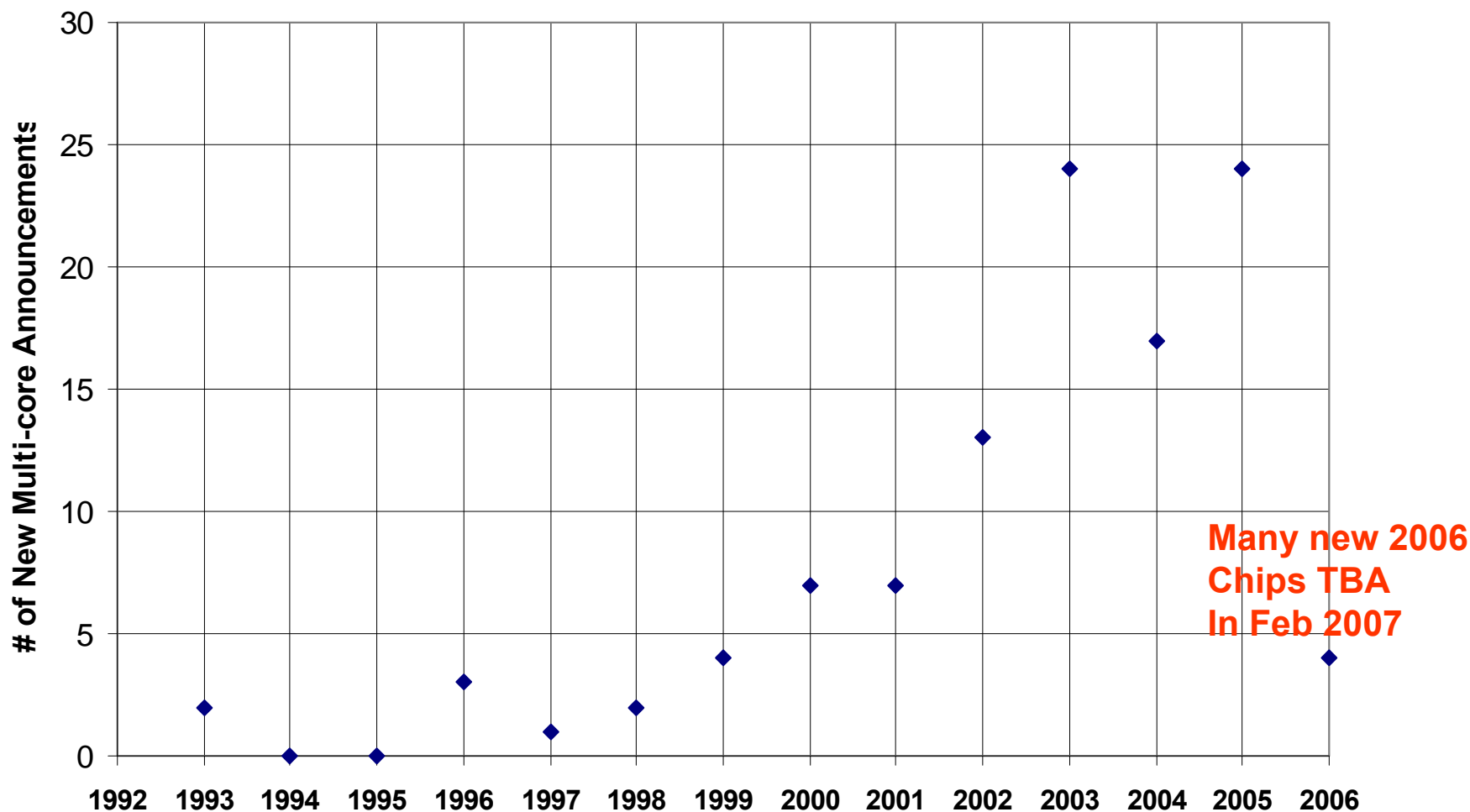
---

- More complex CPU cores no longer cost effective
  - High complexity & design costs
  - “Slow wires” make high clocks tough
  - Decreasing efficiency due to relatively slower memory
  - Need bigger caches for latency
  - *Power, Power, Power, ...*
- Solution: “reuse” simpler design in better technology & place *multiple cores* on same die
  - Combine with shared memory hierarchy





# The Tide of Announcements

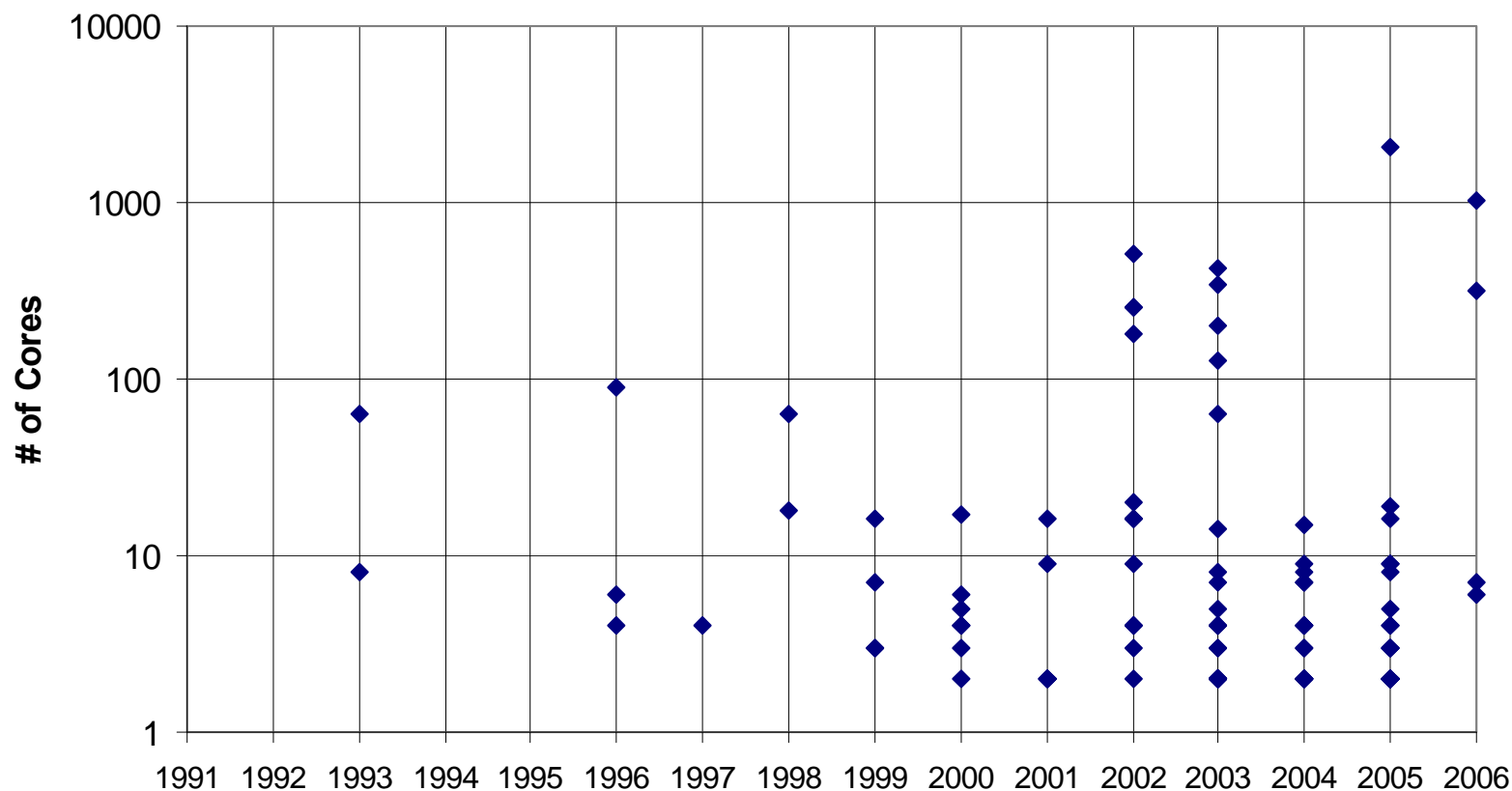






# The Number of Cores per Announcement

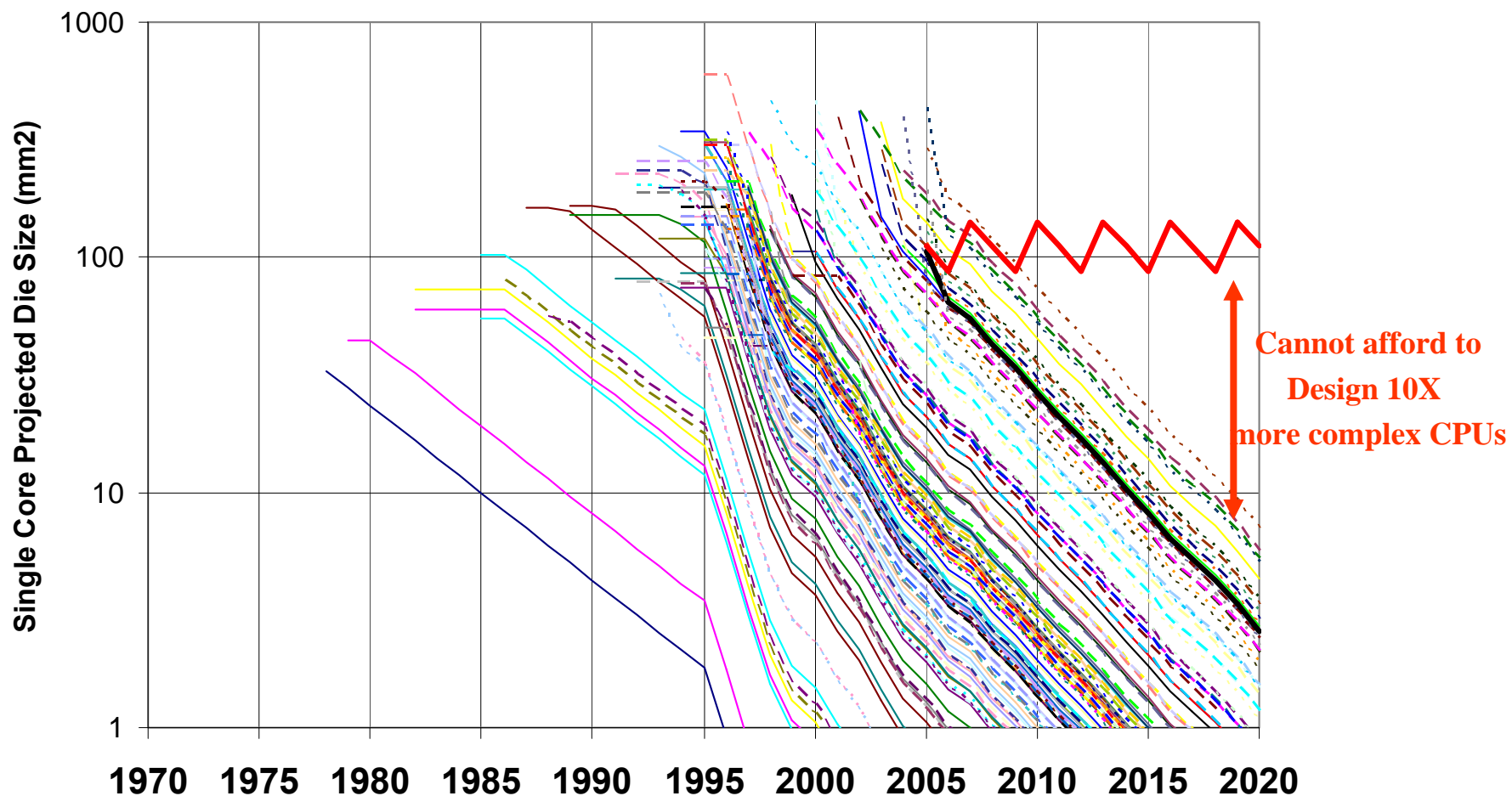
## Multi-core Announcements







# Scaling Today's Single Core uP Chips

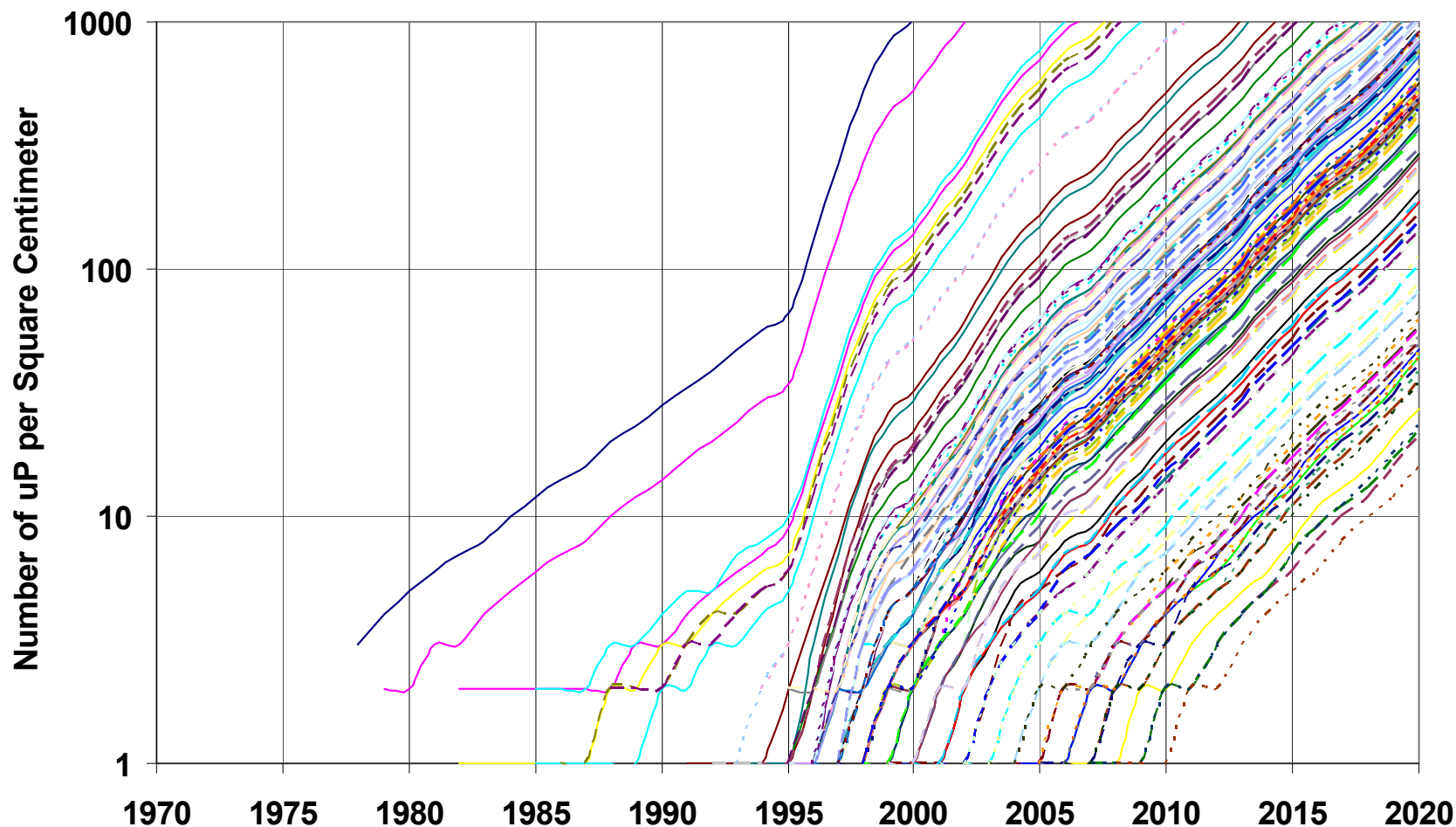


Each line represents the scaling of a unique real microprocessor chip from its inception





# What's The Multi-core Potential



Each line represents the scaling of a unique real microprocessor chip from its inception





# Examples of Multi-Core Designs

---

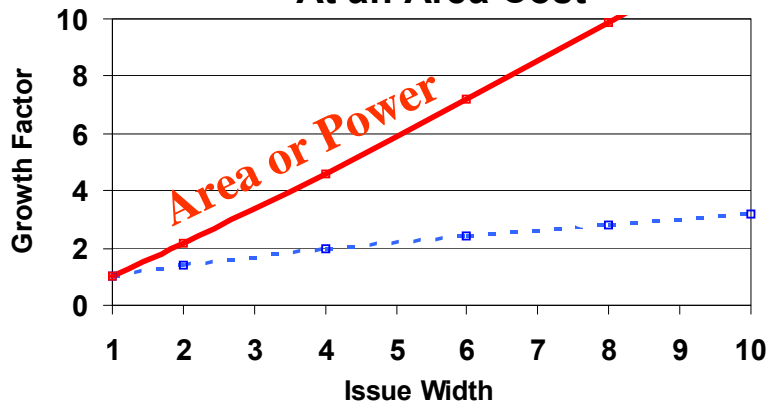
- **Microprocessors**
  - 1993: EXECUBE
  - IBM POWER4 dual-core
  - Intel XEON dual-core
  - Sun dual core UltraSPARC
  - IBM CELL 9 way
  - IBM Bluegene/L dual core with embedded DRAM
  - Sun Niagara 8 way core
  - Clearspeed Array Processors
- **Specialized chips**
  - Network processors (up to 100s of cores)
  - Graphics & game processors
- **Many multi-core designs also using multi-threaded cores**



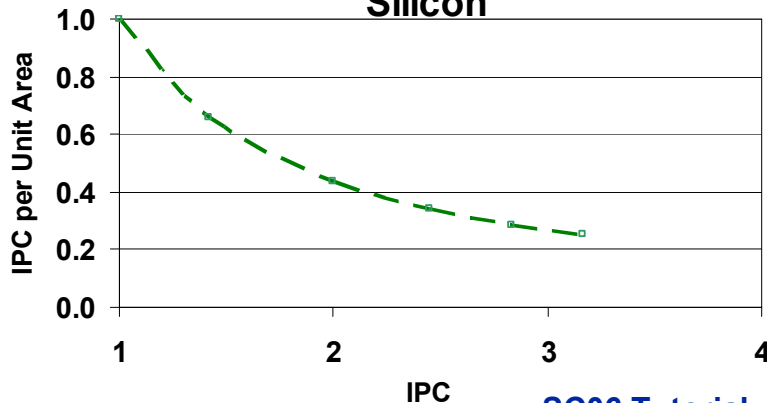


# Why are MC Cores Going Simple? Today's Single Threaded Core Performance = IPC x Clock

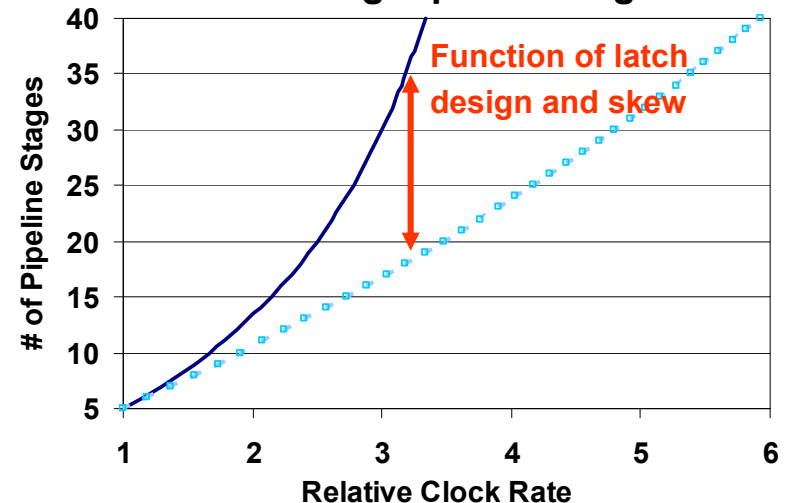
Issue Width Drives  
Microarchitectural Performance  
- At an Area Cost



Resulting Loss of Effective Use of  
Silicon



Increasing Clock Rate Requires  
Increasing Pipeline Stages

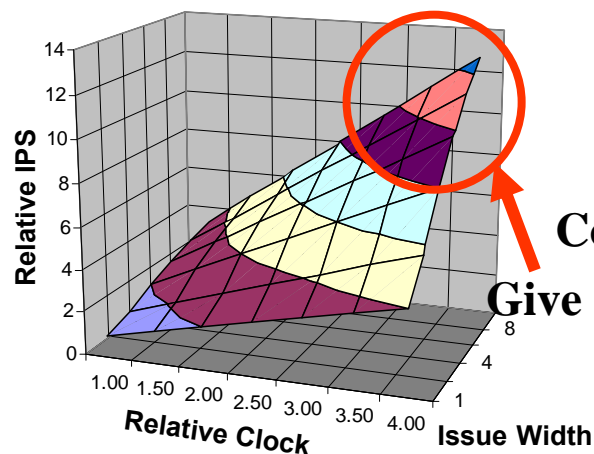


- More stages => higher branch & forwarding penalties
- Higher clock => larger relative memory latency
  - Requires bigger caches
- Result: performance now dominated by # of permitted outstanding loads



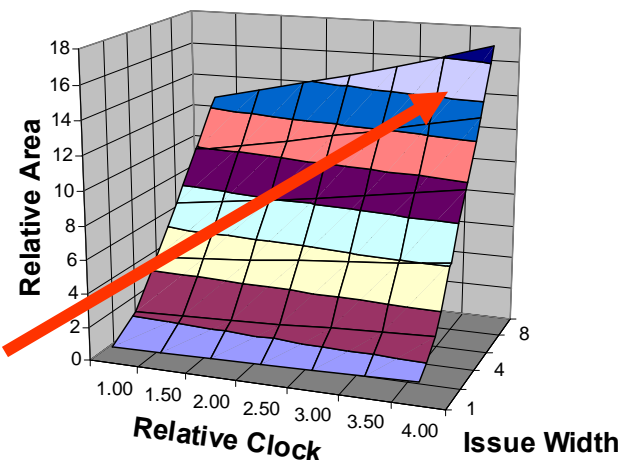


# Notional Core Design Space

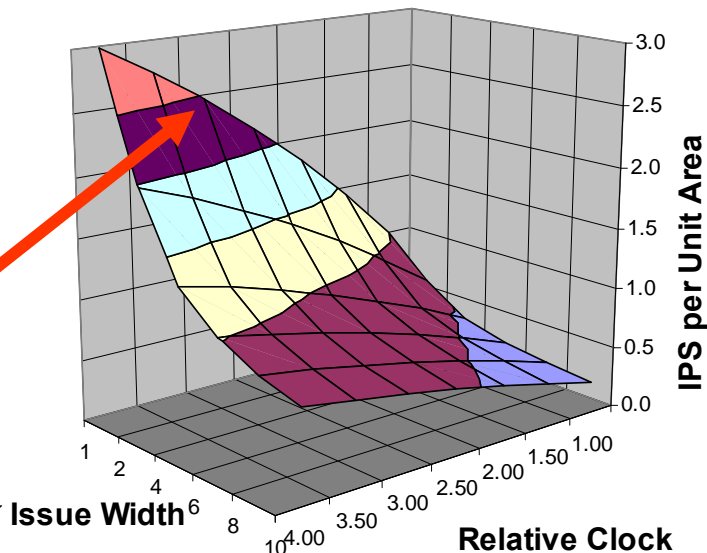


Complex designs  
Give most performance

But also largest  
area



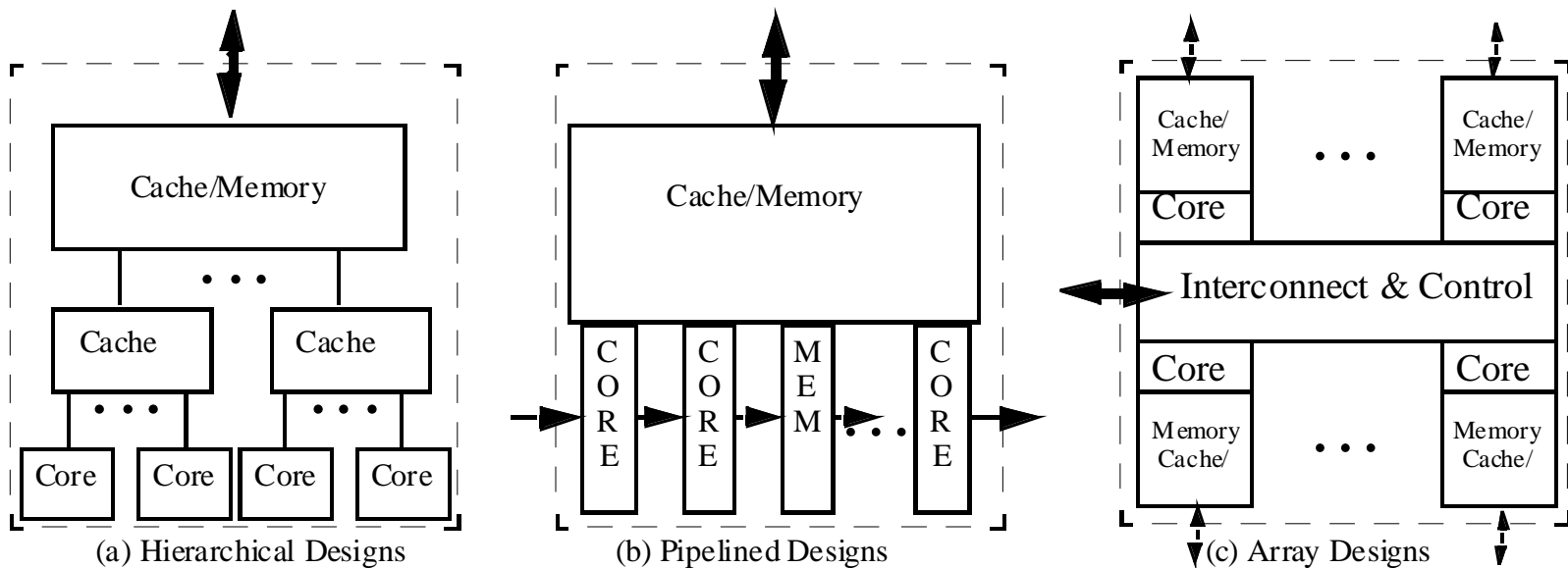
But simpler gives  
better performance/area







# What is Today's Multi-Core Design Space



- Intel Core Duo
- IBM Power5
- Sun Niagara
- ...

- IBM Cell
- Most Router chips
- Many Video chips

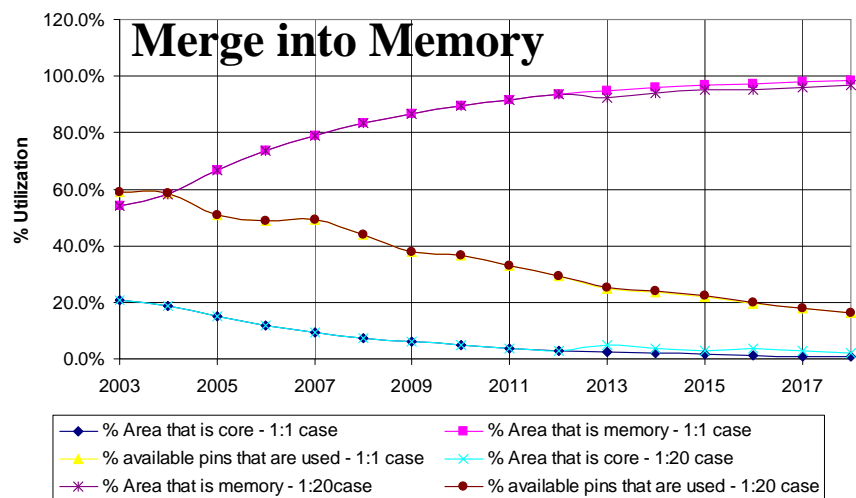
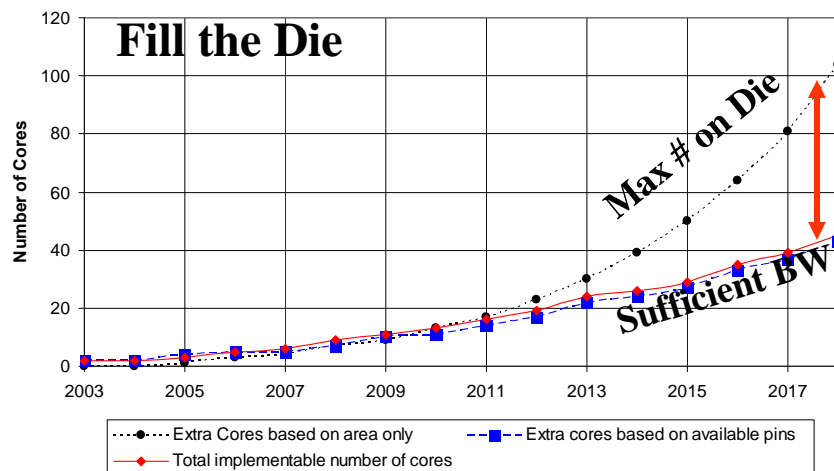
- Terasys
- Execube
- Yukon





# Multi-Core Projection Models

- **“Fill the die”**: Add cores to fill die
  - Contacts for external memory bandwidth will dominate die area
- **“Processing in Memory”**: merge with memory
  - Lots of local bandwidth, single part type design



See P. Kogge, “An Exploration of the Technology Space for Multi-Core Memory/Logic Chips for Highly Scalable Parallel Systems,” IEEE Int. Workshop on Innovative Architectures, Turtle Bay, Hawaii, Jan. 2005.





# Another Reason for Multi-Core: Yield Enhancement

---

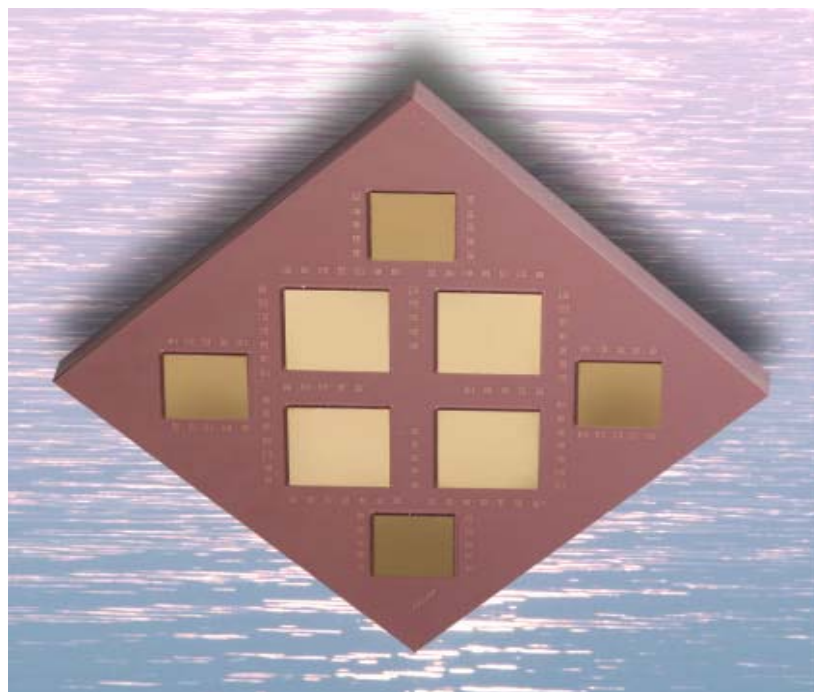
- Add extra cores for redundancy
  - Requires associated interconnect
- Sell die with less than full performance
- Recent case study (Kogge, IWIA, Jan. 2006)
  - Goal: “cheapest” chip with constant storage/MIPS/I/O
  - Core IPC assumed  $\sqrt{\text{area}}$
  - Parameters: ITRS roadmap, die size, core complexity
  - Approach: sweep parameter space for highest perf/wafer
- Key results
  - Yield considerations favor smaller die
  - Optimal core microarchitecture: simplest
  - Adding purely redundant cores of little value
  - Selling partially good die reasonable good idea





# Silicon Alone is not the Complete Story

---







# Observations

---

- Silicon growing irregularly in
  - Memory density per square cm
  - Performance possible per square cm
  - Off-chip I/O bandwidth per square cm
- 99% of today's logic chips
  - Do no computation
  - And are mostly memory
- And we pay a huge overhead when
  - Densest memory technology not used
  - Memory & logic on separate chips
- *It's the interconnect to memory, stupid!*





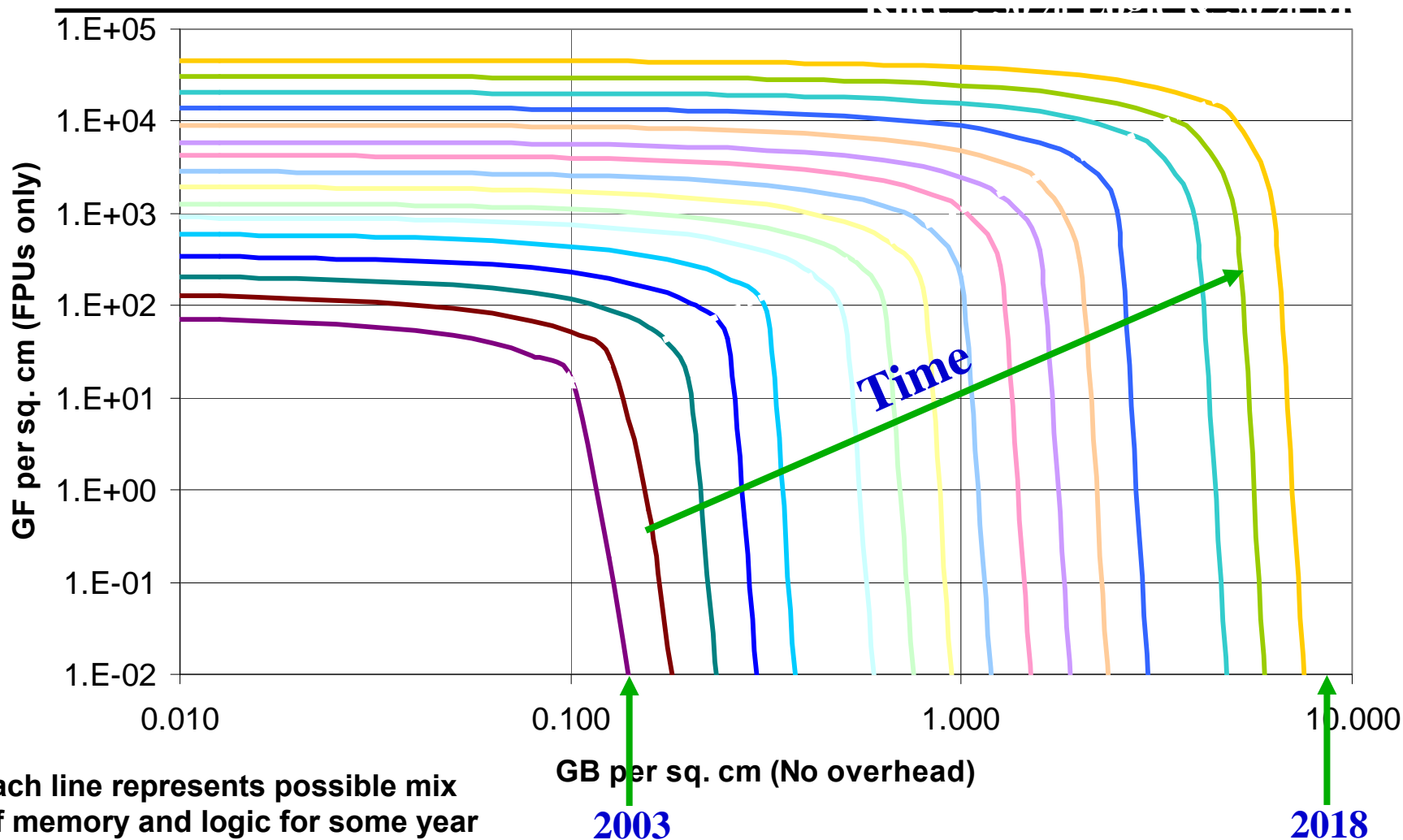
---

# **A Contrarian's View Processing in Memory: The Grand Synthesis of Logic and Memory**





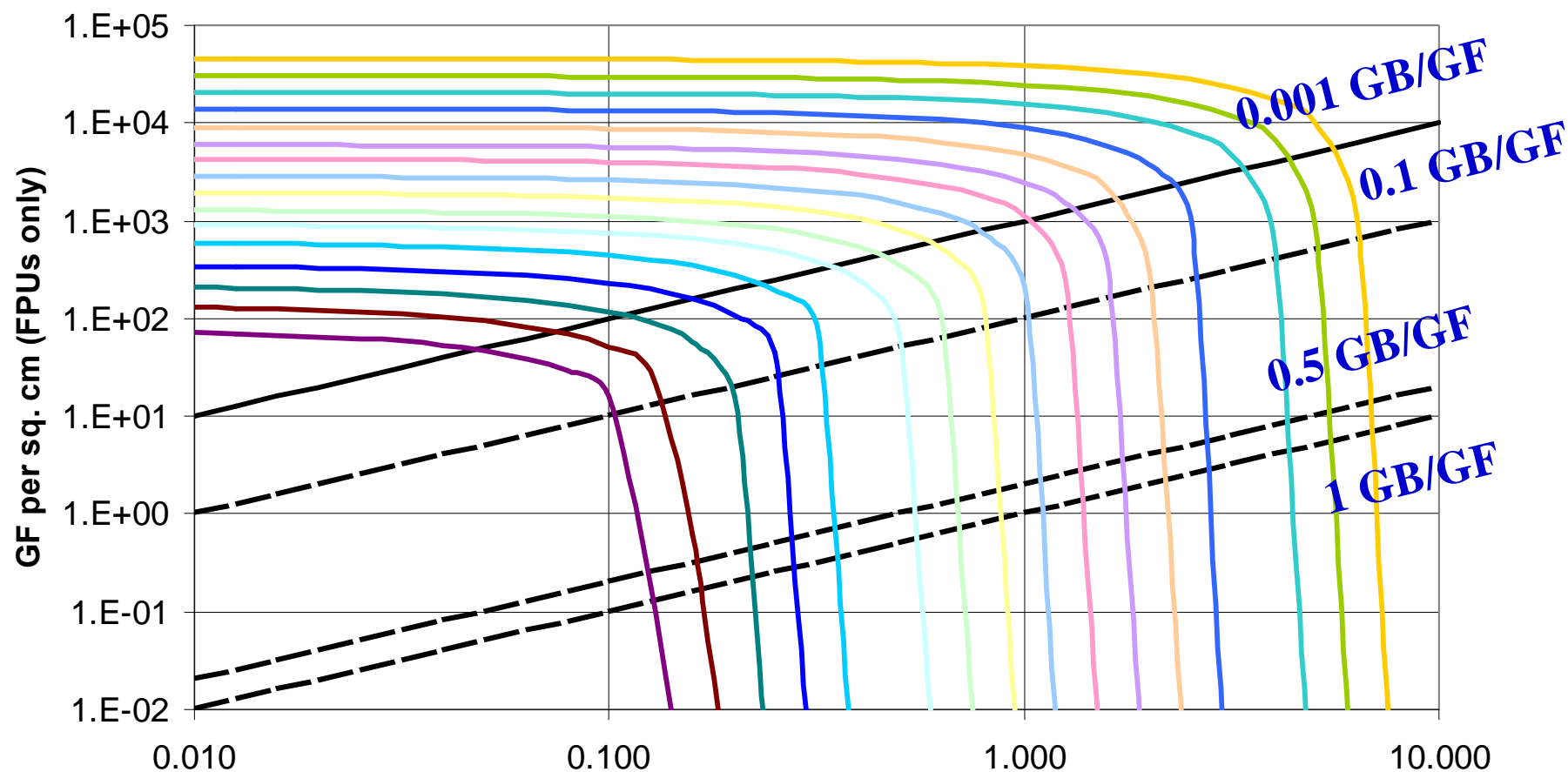
# How can we use a sq. cm? (with no overhead)







# Adding In “Lines of Constant Performance”



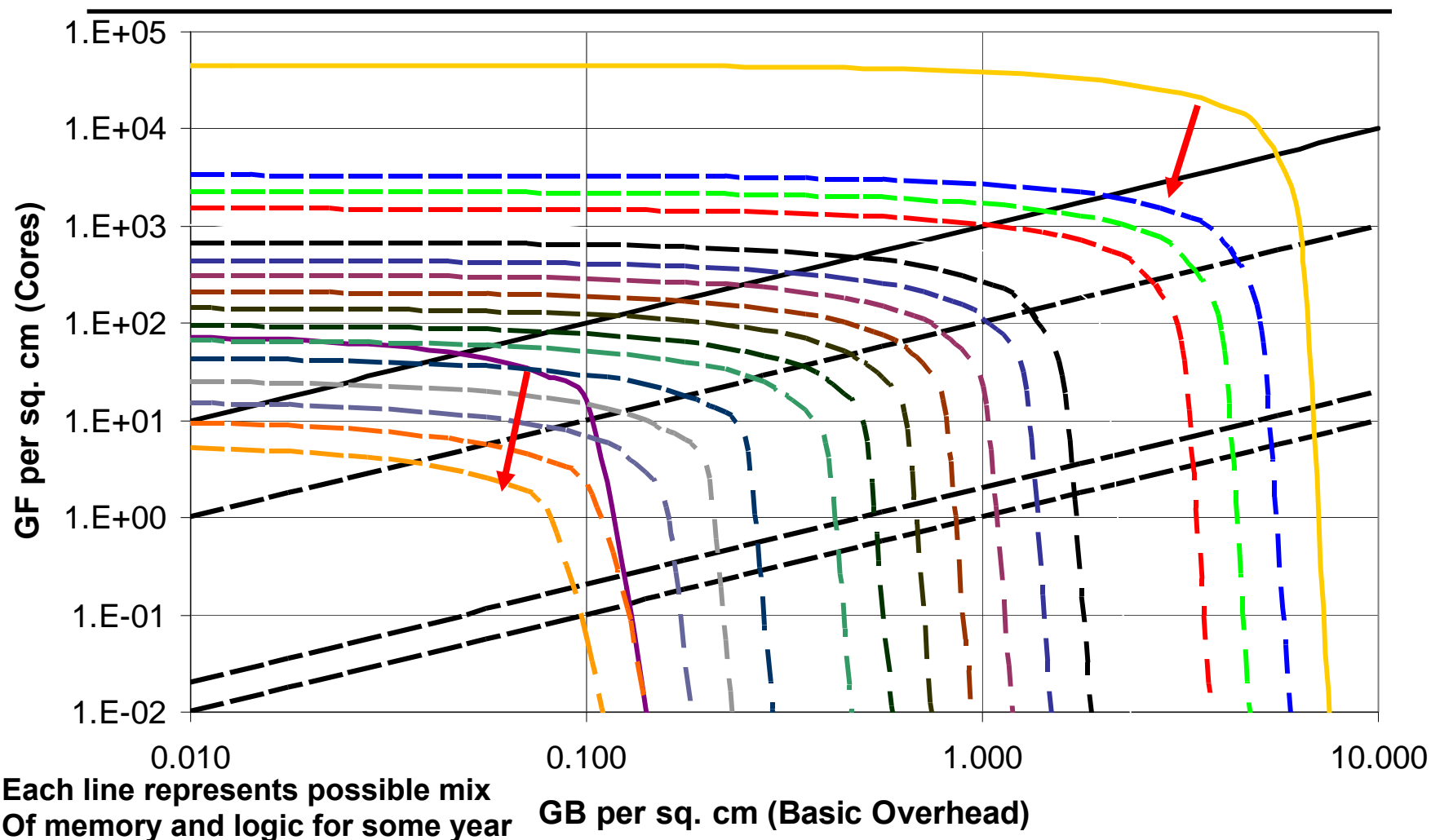
Each line represents possible mix  
Of memory and logic for some year

GB per sq. cm (No overhead)





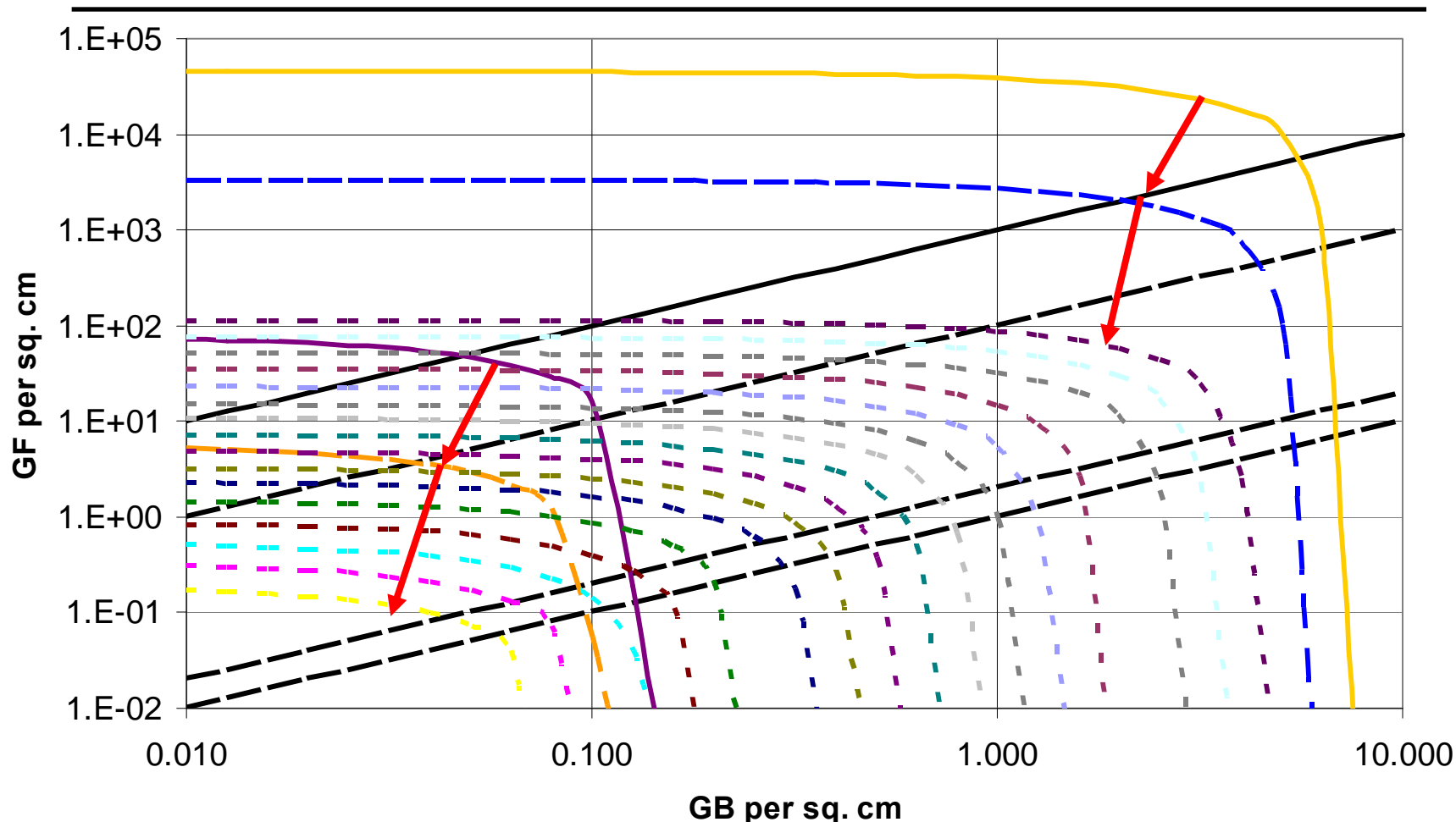
# Knee Curves with Basic Overheads







# Knee Curves with Today's Overheads

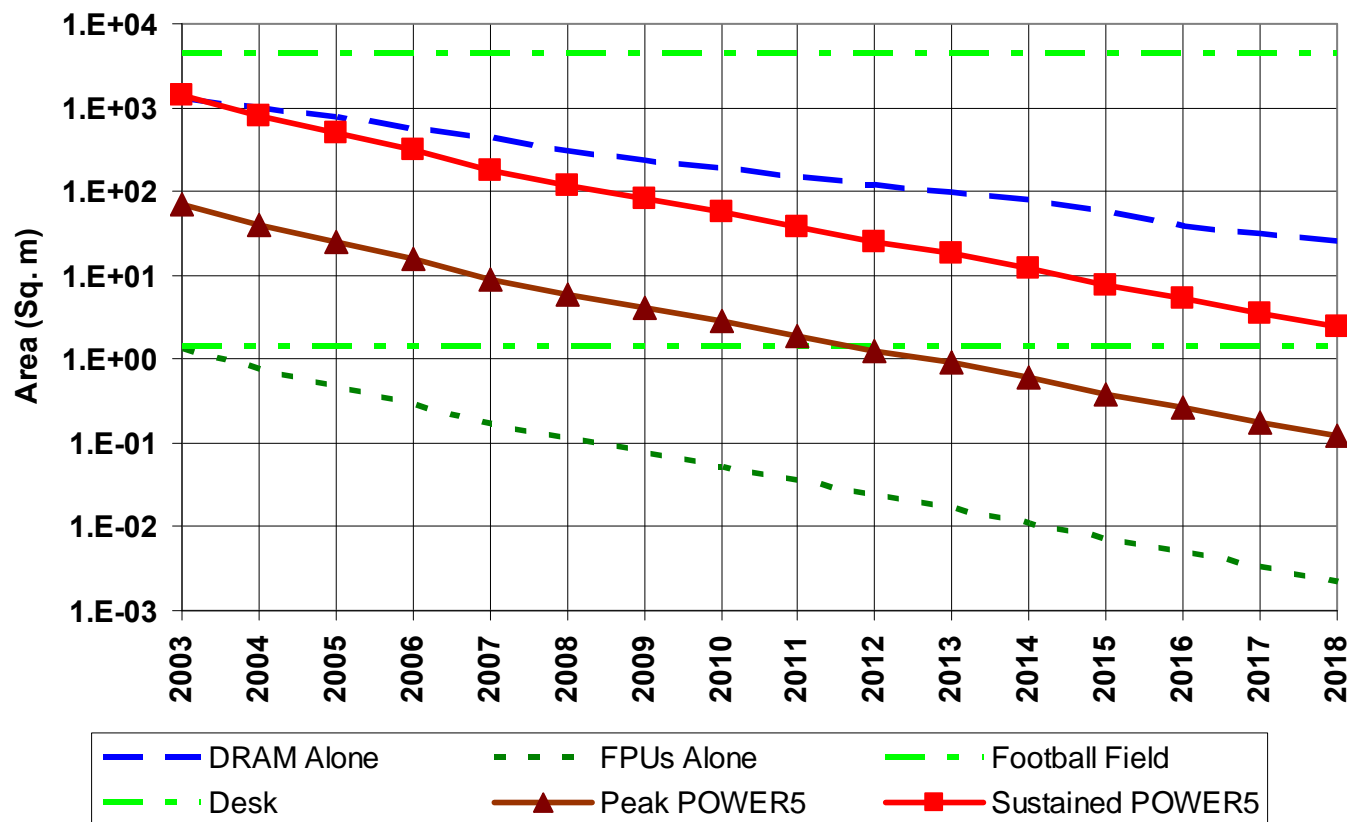


**Partitioning chips as we do today is hugely inefficient**





# Minimal Size for a “Peta” System



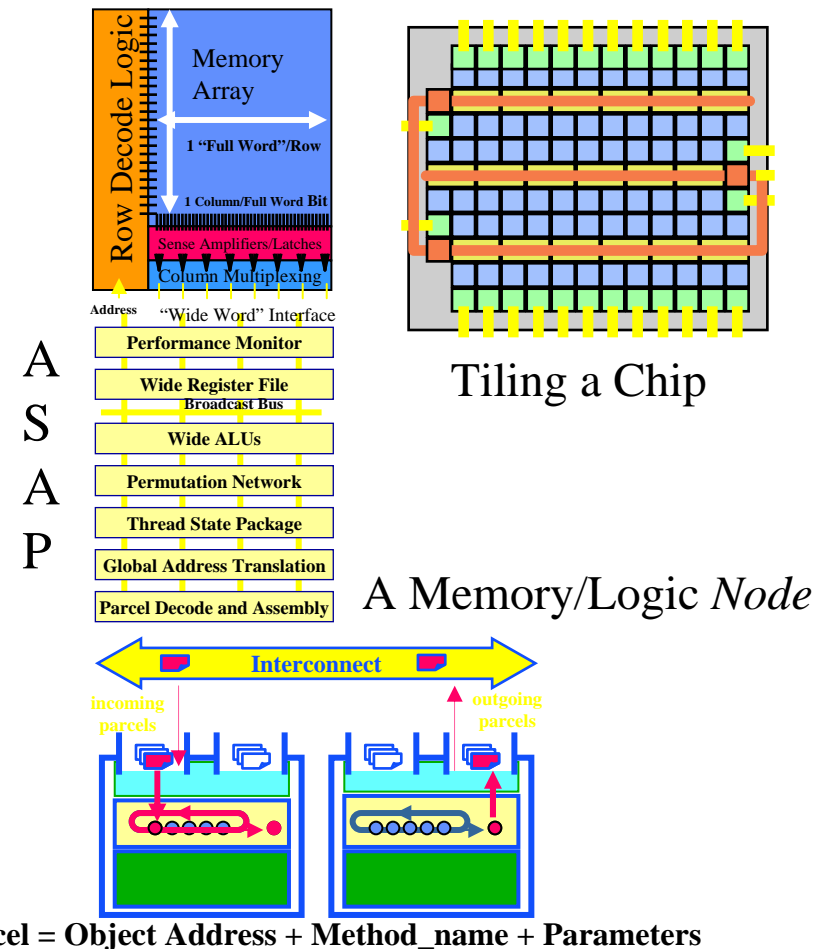
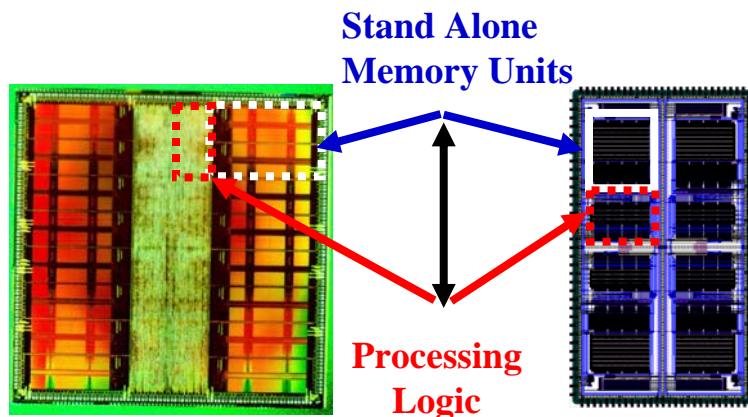
- In terms of silicon area: *“It’s the memory!”*
- We extract little benefit from most of our high cost logic





# “Processing-In-Memory”

- High density memory on same chip with reasonable dense logic
  - Not just caches
- Very fast access from logic to memory
- Very high bandwidth
- ISA/microarchitecture designed to utilize high bandwidth
- Tile with “memory+logic” nodes

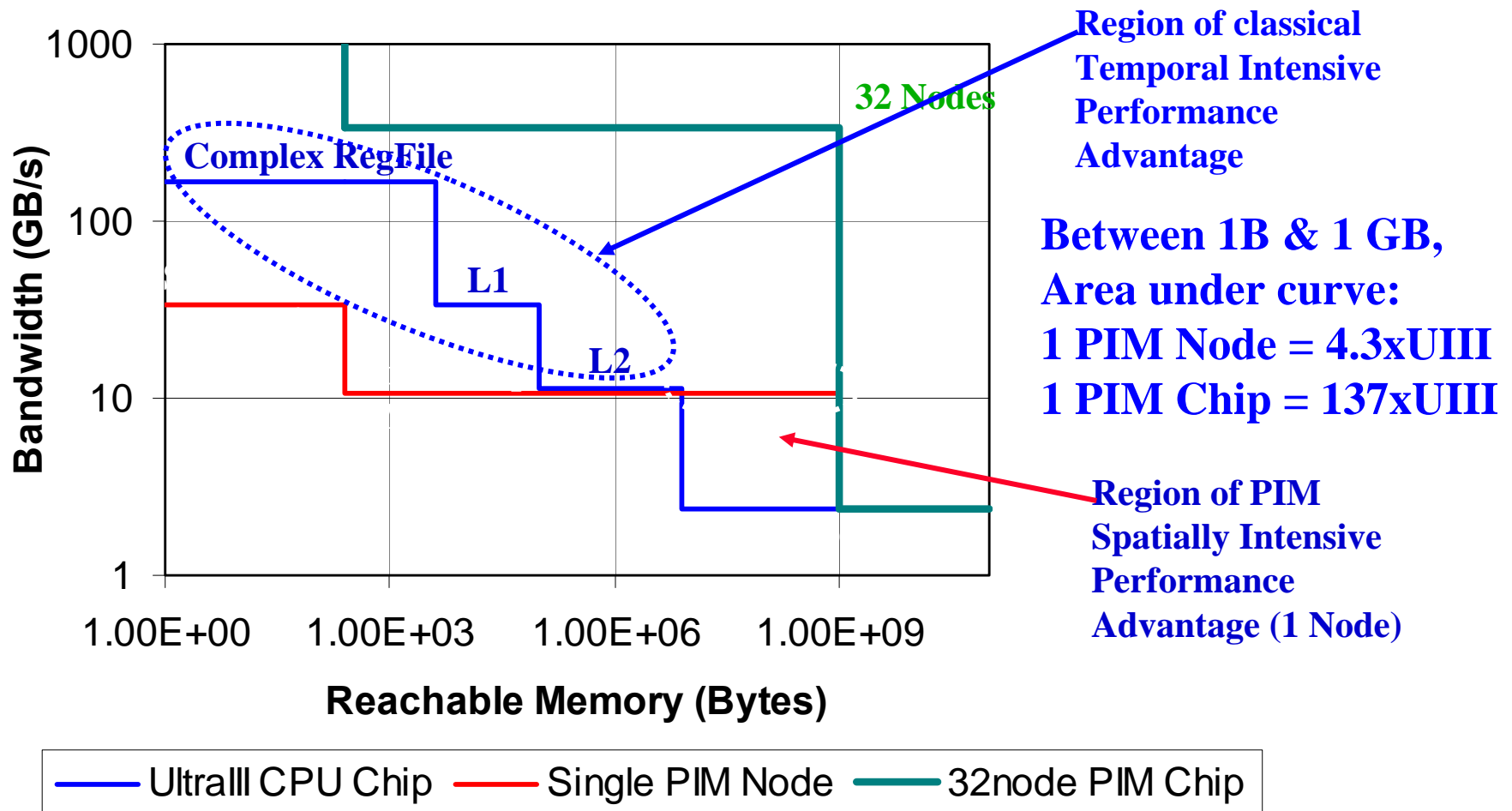






# The PIM

## “Bandwidth Bump”

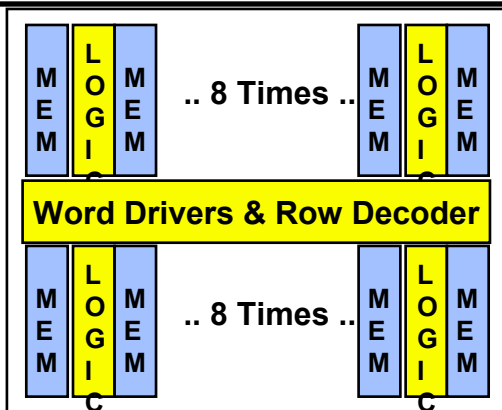




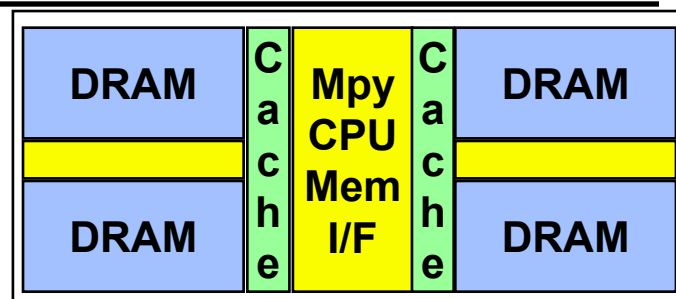


# PIM Chip

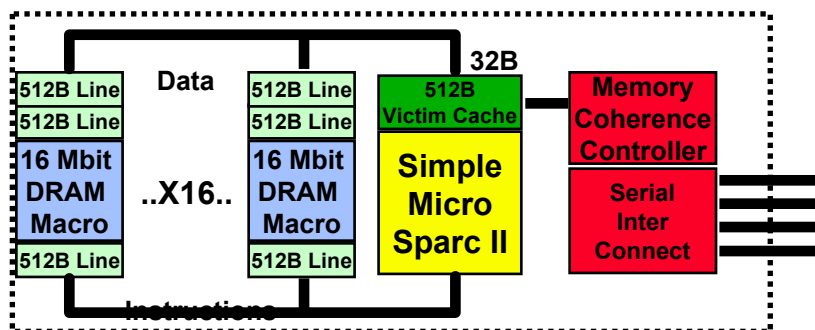
## MicroArchitectural Spectrum



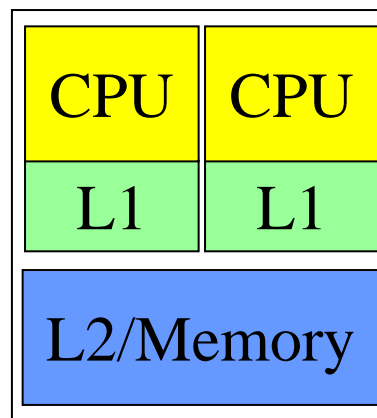
**SIMD: Linden DAAM**



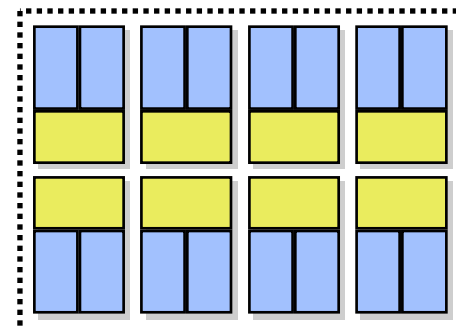
**Single Chip Computer:  
Mitsubishi M32R/D**



**Complete SMP Node:  
Proposed SUN part**



**Chip Level SMP:  
POWER4, BG/L**



**Tiled & Scalable:  
BLUE GENE,  
EXECUBE**





# PIM System Design Space: Historical Evolution

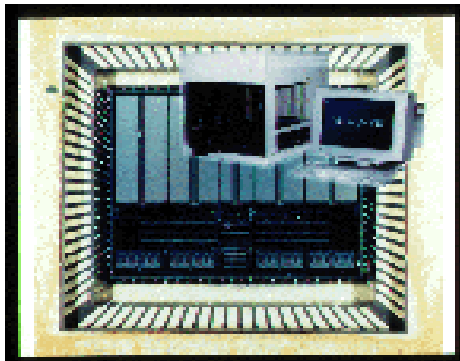
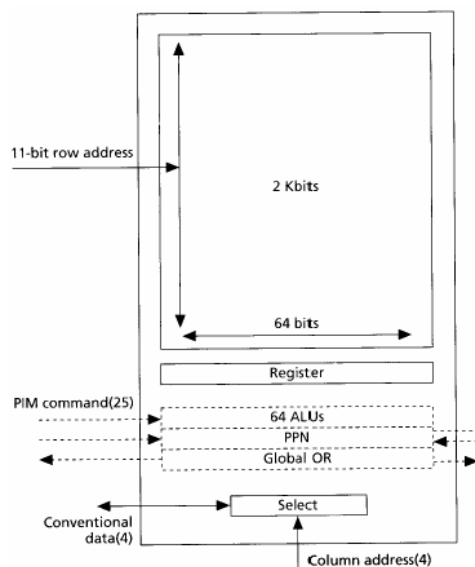
---

- Variant One: **Accelerator** (historical)
- Variant Two: **Smart Memory**
  - Attach to existing SMP (using an existing memory bus interface)
  - PIM-enhanced memories, accessible as *memory* if you wish
  - Value: Enhancing performance of *status quo*
- Variant Three: **Heterogeneous Collaborative**
  - PIMs become “independent,” & communicate as peers
  - Non PIM nodes “see” PIMs as equals
  - Value: Enhanced concurrency and generality over variant two
- Variant Four: **Uniform Fabric (“All PIM”)**
  - PIM “fabric” with fully distributed control and emergent behavior
  - Extra system I/O connectivity required
  - Value: Simplicity and economy over variant three
- Option for any of above: **Extended Storage**
  - Any of above where each PIM supports separate dumb memory chips





# TERASYS SIMD PIM (circa 1993)



- Memory part for CRAY-3
- “Looked like” SRAM memory
  - With extra command port
- 128K SRAM bits (2k x 64)
- 64 1 bit ALUs
- SIMD ISA
- Fabled by National
- Also built into workstation with 64K processors
  - 5-48X Y-MP on 9 NSA benchmarks

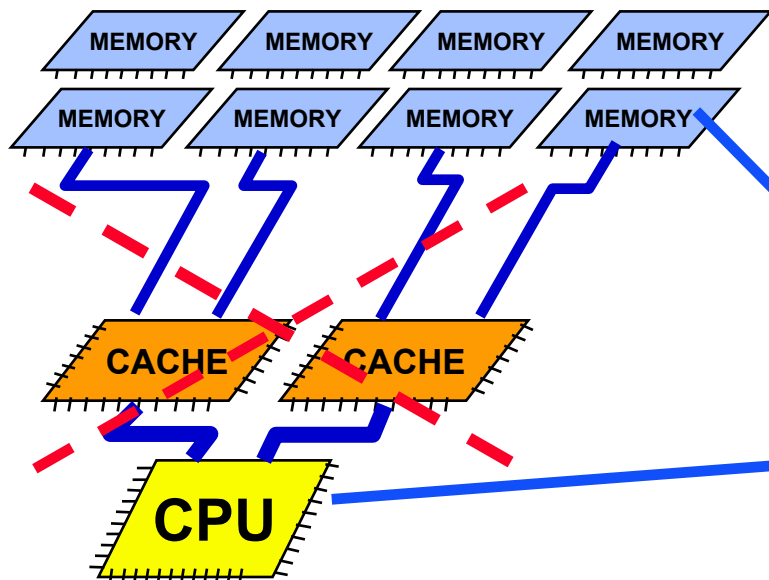
Gokhale, M., Holmes, B., Iobst, K.: Processing in Memory: the Terasys Massively Parallel PIM Array. Computer , 28(3):23--31, April 1995.



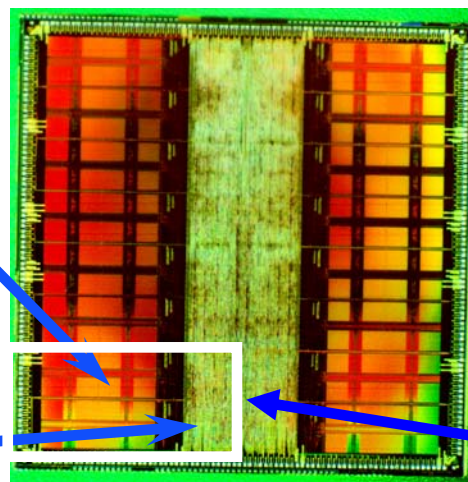


# EXECUBE: An Early MIMD PIM & 1<sup>st</sup> True MC (1st Silicon 1993)

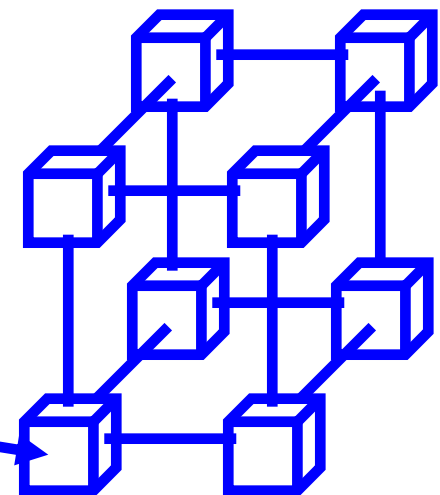
- First DRAM-based Multi-Core with Memory
- *Designed from onset for “glueless” one-part-type scalability*
- On-chip bandwidth: 6.2 GB/s; Utilization modes > 4GB/s



*Include  
“High Bandwidth”  
Features in ISA*



8  
Compute Nodes  
on ONE Chip



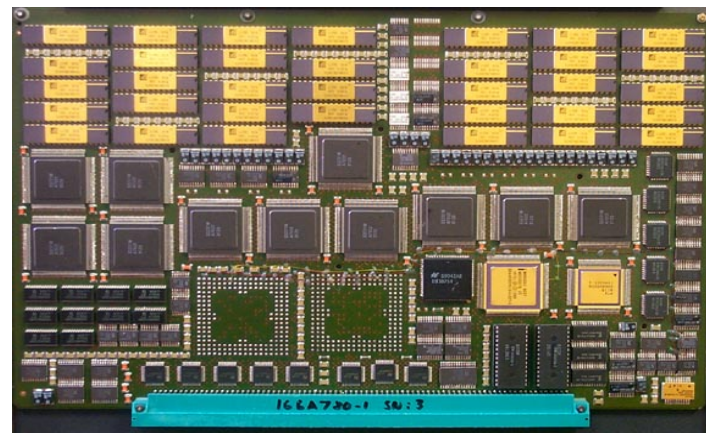
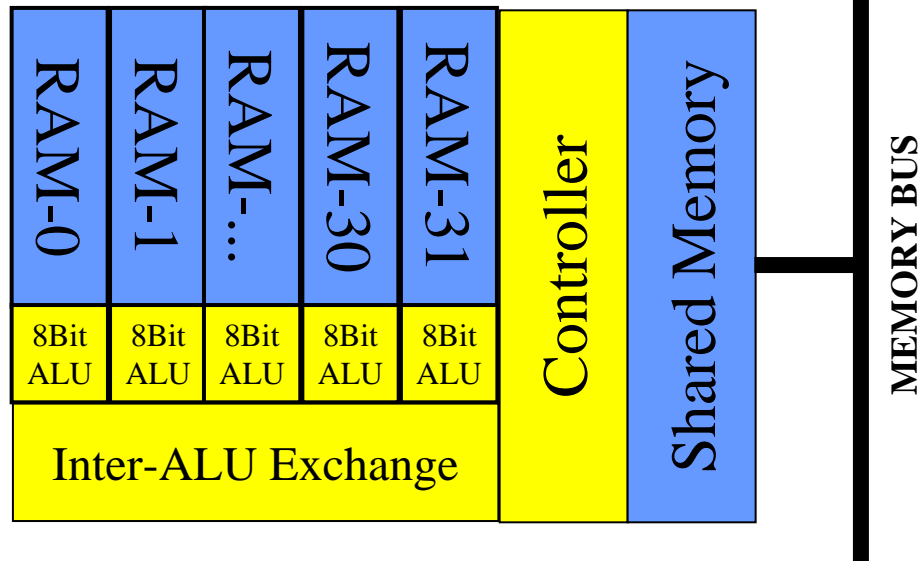
**EXECUBE:**  
3D Binary Hypercube  
SIMD/MIMD on a chip

Kogge, “EXECUBE,” ICPP, 1994.





# RTAIS: The First ASAP (circa 1993)

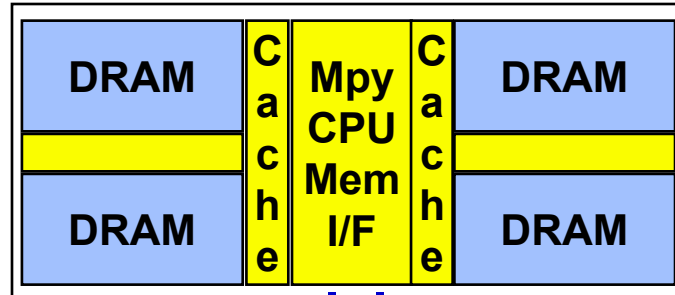


- Application: “Linda in Memory”
- Designed from onset to perform wide ops “at the sense amps”
- More than SIMD: flexible mix of VLIW
- “Object oriented” multi-threaded memory interface
- Result: 1 card 60X faster than state-of-art R3000 card





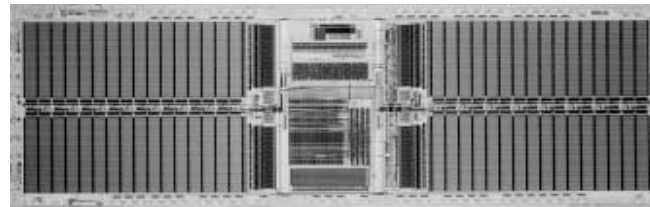
# Mitsubishi M32R/D



Also two 1-bit I/Os

16 bit data bus

24 bit address bus



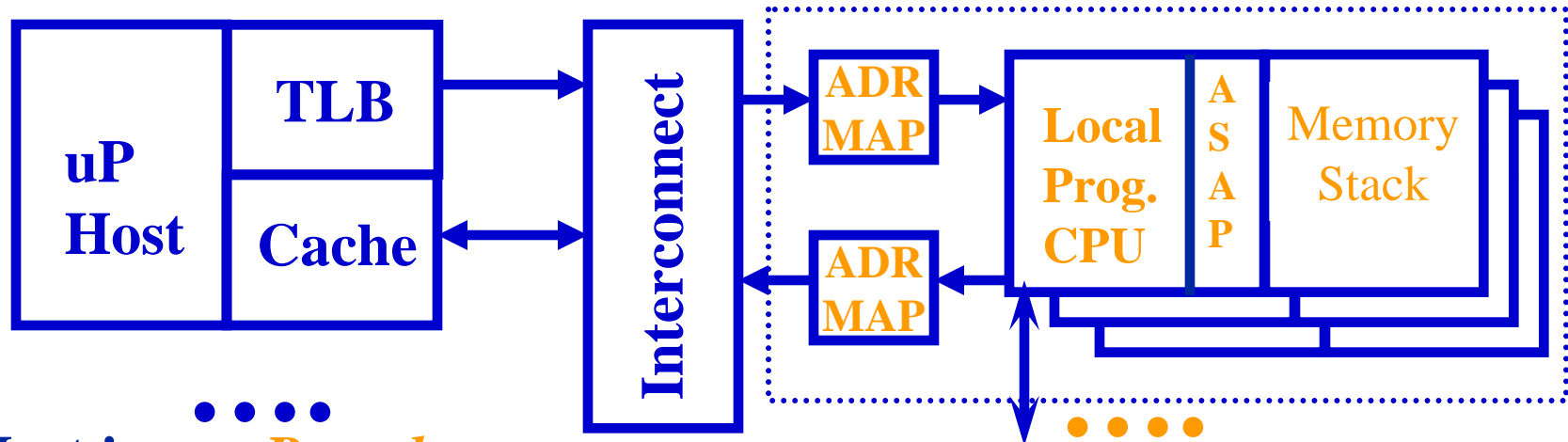
- 32-bit fixed point CPU + 2 MB DRAM
- “Memory-like” Interface
- Utilize wide word I/F from DRAM macro for cache line

Yasuhiro Nunomura et al, “M32R/D-Integrating DRAM and Microprocessor,” IEEE Micro, Nov/Dec 1997



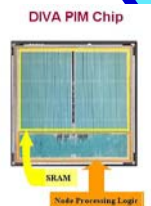
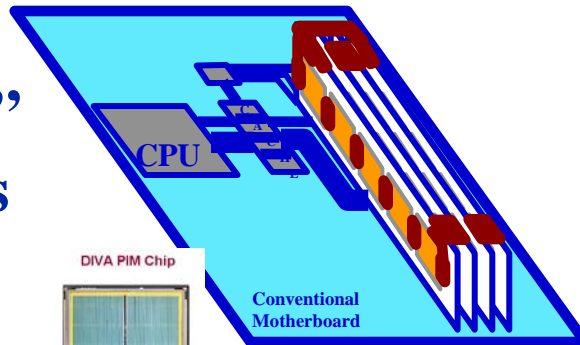


# DIVA: Smart DIMMs for Irregular Data Structures



Host issues *Parcels*

- Generalized “Loads & Stores”
- Treat memory as *Active Object-oriented store*



- 1 CPU + 2MB
- MIPS + “Wide Word”

DIVA Functions:

- Prefix operators
- Dereferencing & pointer chasing
- Compiled methods
- Multi-threaded
- May generate parcels

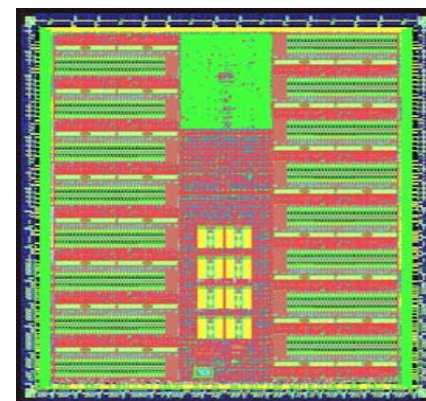
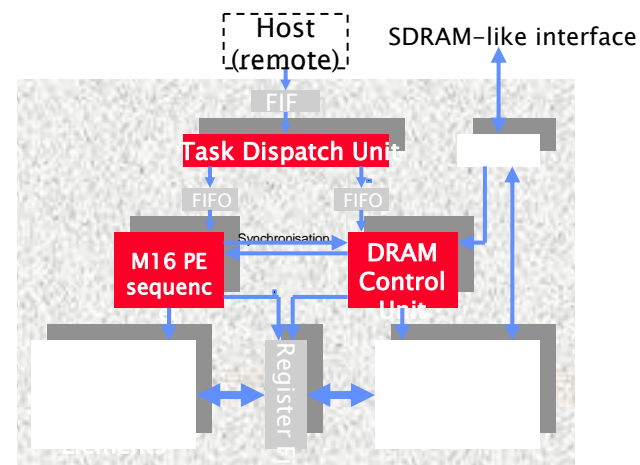
Draper, et al. The Architecture of the DIVA Processing-In-Memory Chip. ICS'05





# Micron Yukon

- 0.15 $\mu$ m eDRAM/ 0.18 $\mu$ m logic process
- 128Mbits DRAM
  - 2048 data bits per access
- 256 8-bit integer processors
  - Configurable in multiple topologies
- On-chip programmable controller
- Operates like an SDRAM



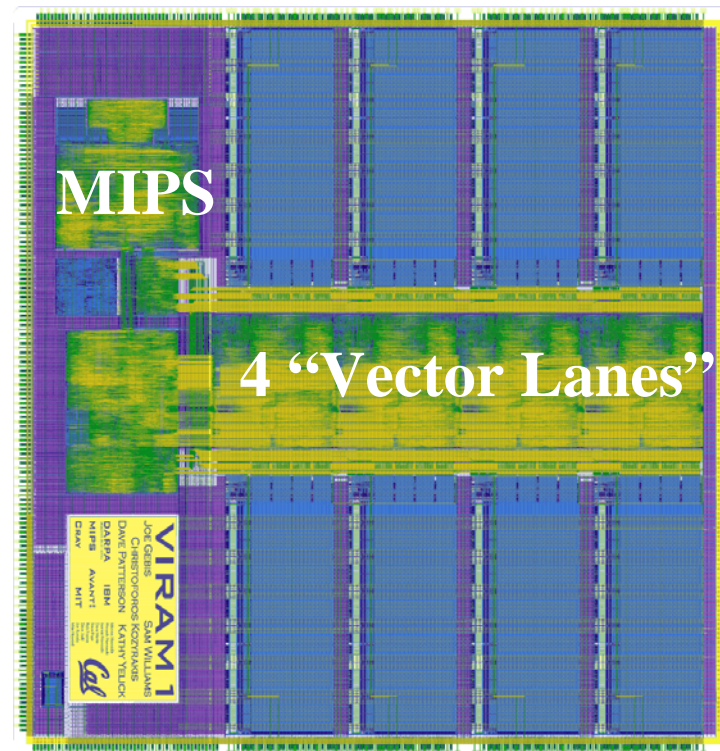
G. Kirsch, "Active Memory: Micron's Yukon," IPDPS 2003.





# Berkeley VIRAM

- System Architecture: single chip media processing
- ISA: MIPS Core + Vectors + DSP ops
- 13 MB DRAM in 8 banks
- Includes flt pt
- 2 Watts @ 200 MHz, 1.6GFlops

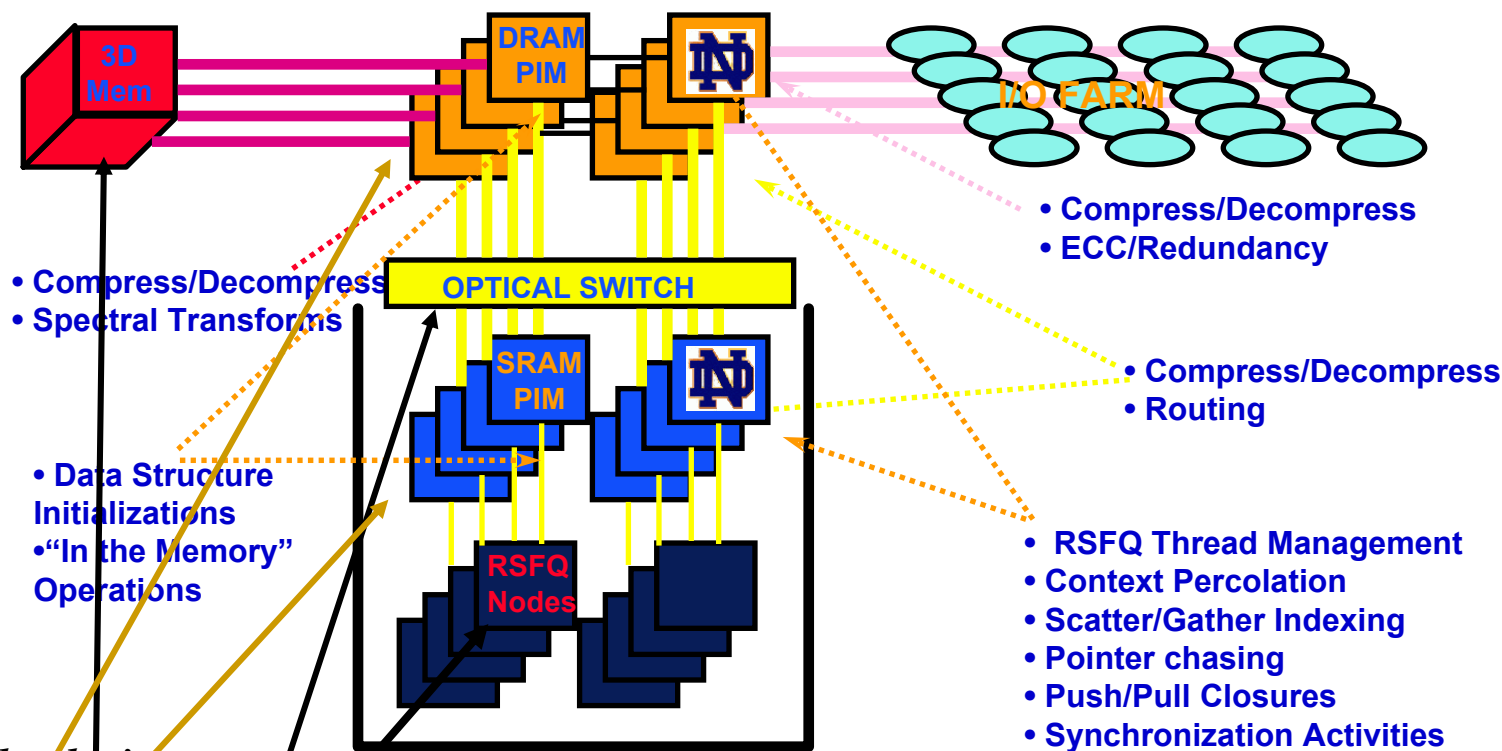


<http://iram.cs.berkeley.edu/papers/2000.HotChips.VIRAM.pdf#search=%22VIRAM%22>






# The HTMT Architecture & PIM Functions



## New Technologies:

- Rapid Single Flux Quantum (RSFQ) devices for 100 GHz CPU nodes
- WDM all optical network for petabit/sec bi-section bandwidth
- Holographic 3D crystals for Petabytes of on-line RAM
- PIM  for active memories to manage latency

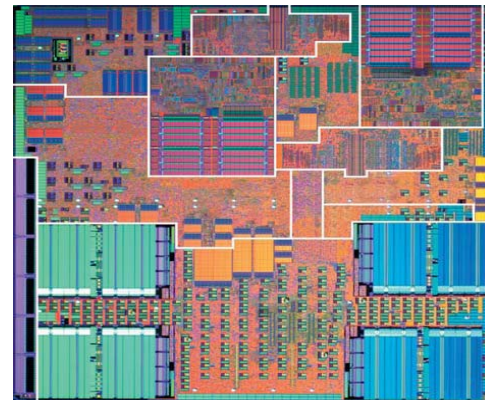
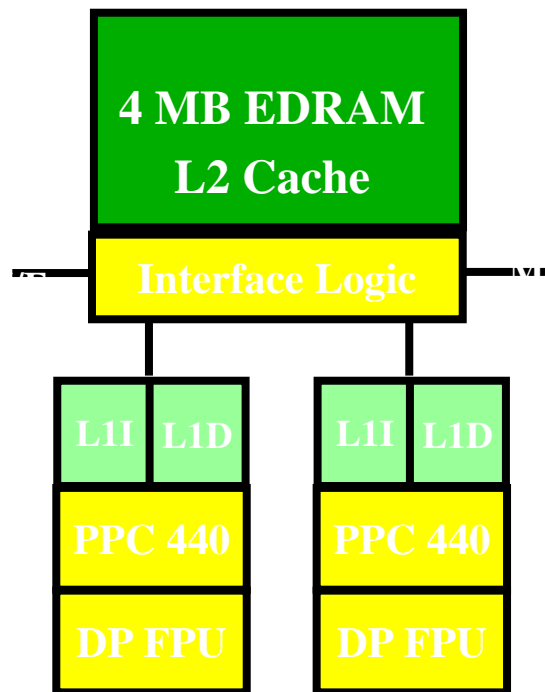
**PIMs in Charge**





# Bluegene/L

- Two simple cores with dense embedded DRAM technology
- Included 4MB of on-chip embedded DRAM
- Designed to scale simply to bigger systems
- Basis for several of world's TOP500 machines

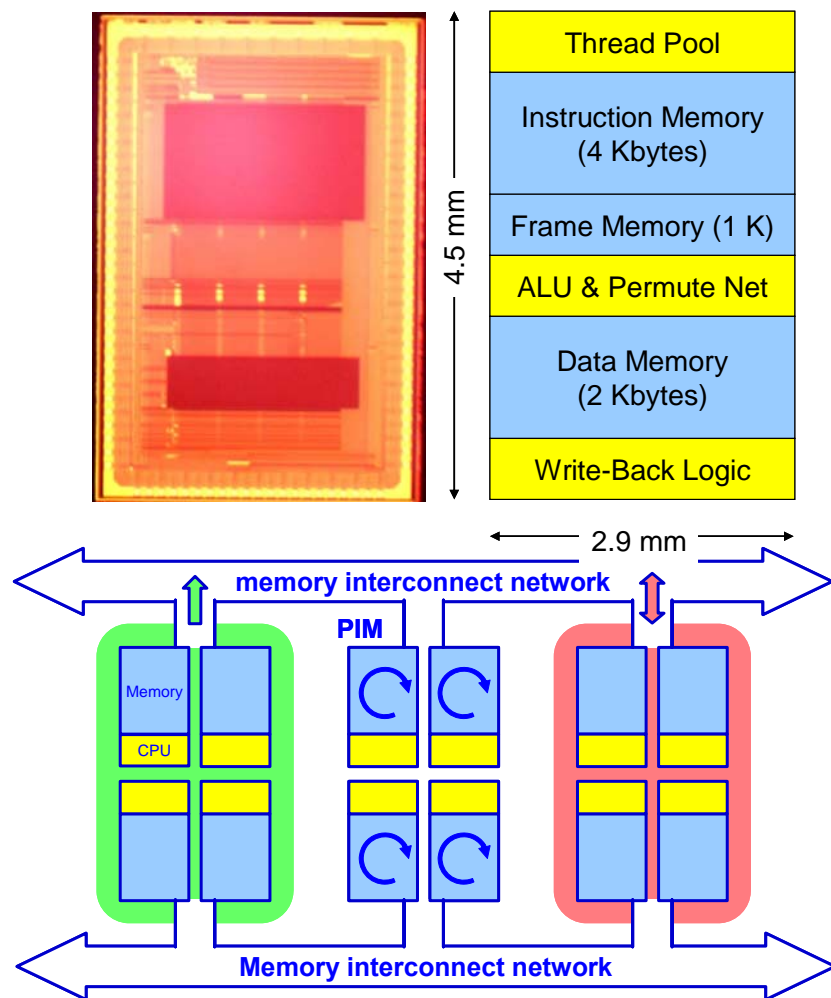


S. S. Iyer, et al, "Embedded DRAM: Technology platform for the Blue Gene/L chip," IBM J. R&D, Volume 49, Number 2/3, Page 333 (2005)

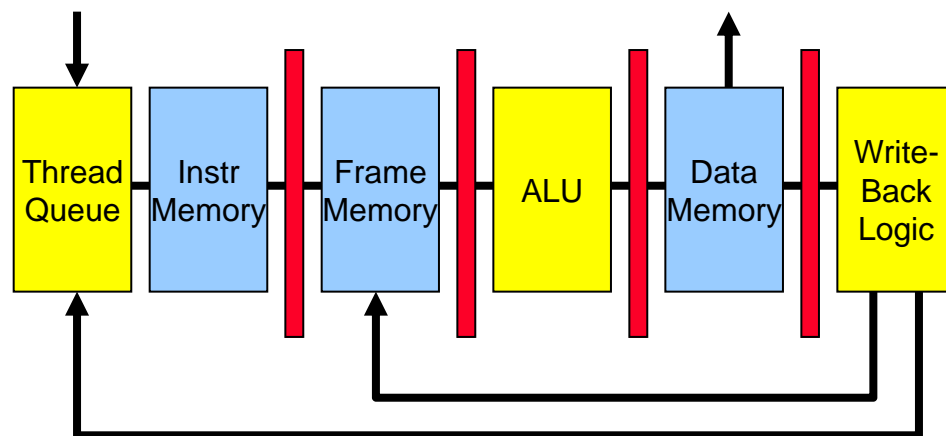




# PIM Lite



- “Looks like memory” at Interfaces
- ISA: 16-bit multithreaded/SIMD
  - “Thread” = IP/FP pair
  - “Registers” = wide words in frames
- Designed for multiple nodes per chip
- 1 node logic area ~ 10.3 KB SRAM (comparable to MIPS R3000)
- TSMC 0.18u 1-node 1<sup>st</sup> pass success
- 3.2 million transistors (4-node)

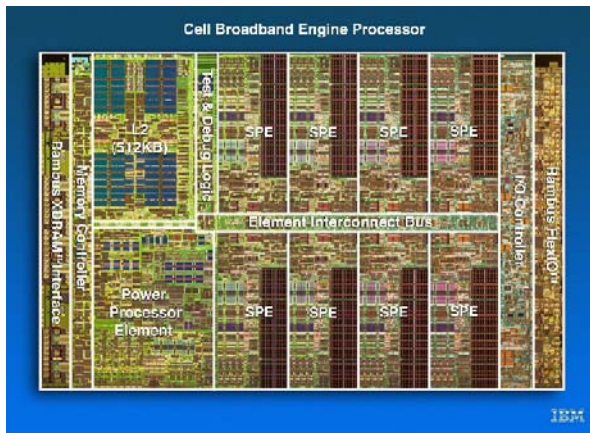
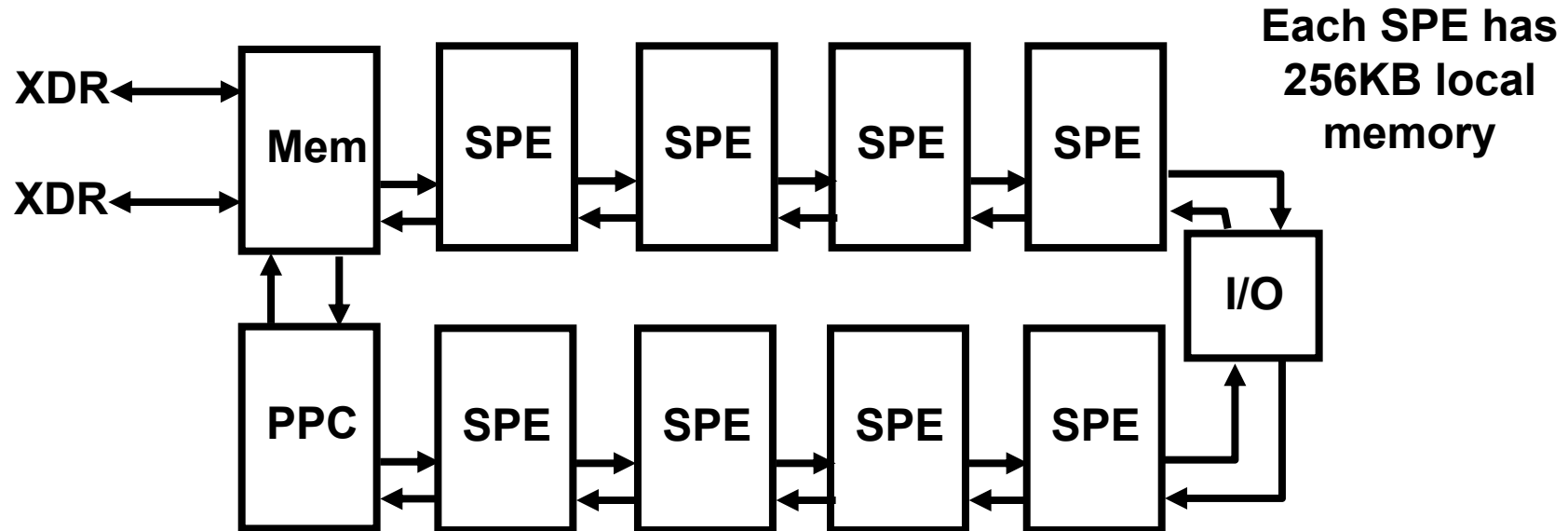






# CELL

## (A Pipelined, Array, Hierarchical MC Chip)



Roadrunner system: 16K MC Optrons  
+ 16K Cell chips

[http://www.research.ibm.com/cell/cell\\_chip.html](http://www.research.ibm.com/cell/cell_chip.html)



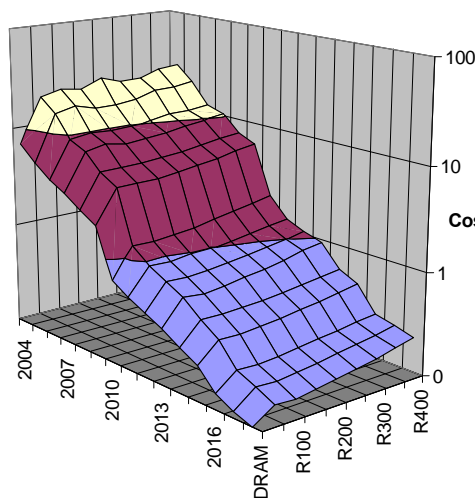
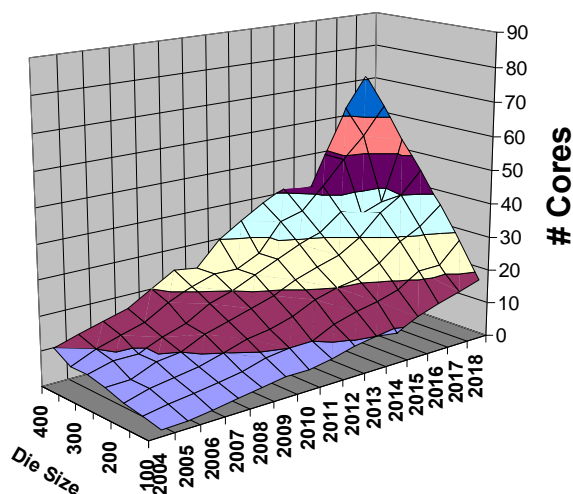


# Projecting Ahead: Optimizing the Multi-Core PIM Chip

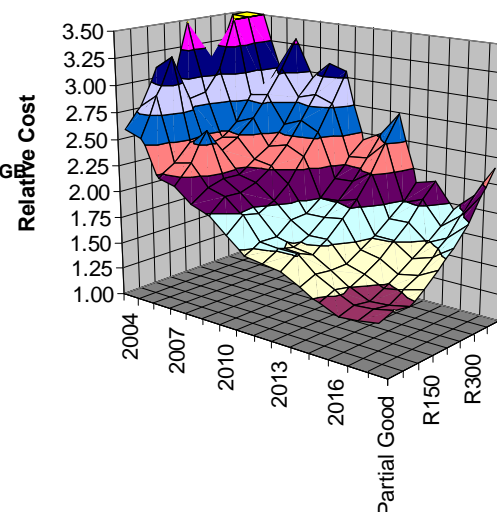
What is optimal # cores/die?

- Assuming fixed memory per unit of processing
- Considering yield effects (smaller die=>more good die)
- Assuming cross-bar inter-core interconnect
- Considering adding redundant cores

Core Count in an Packed Die with Crossbar and I/O



Cost of Configurations Relative to Commodity DRAM



See Kogge & Brockman, "Redundancy in Multi-core Memory-rich Application-Specific PIM Chips," IWIA 2006





## One Step Further: Allowing the Threads to Travel

---

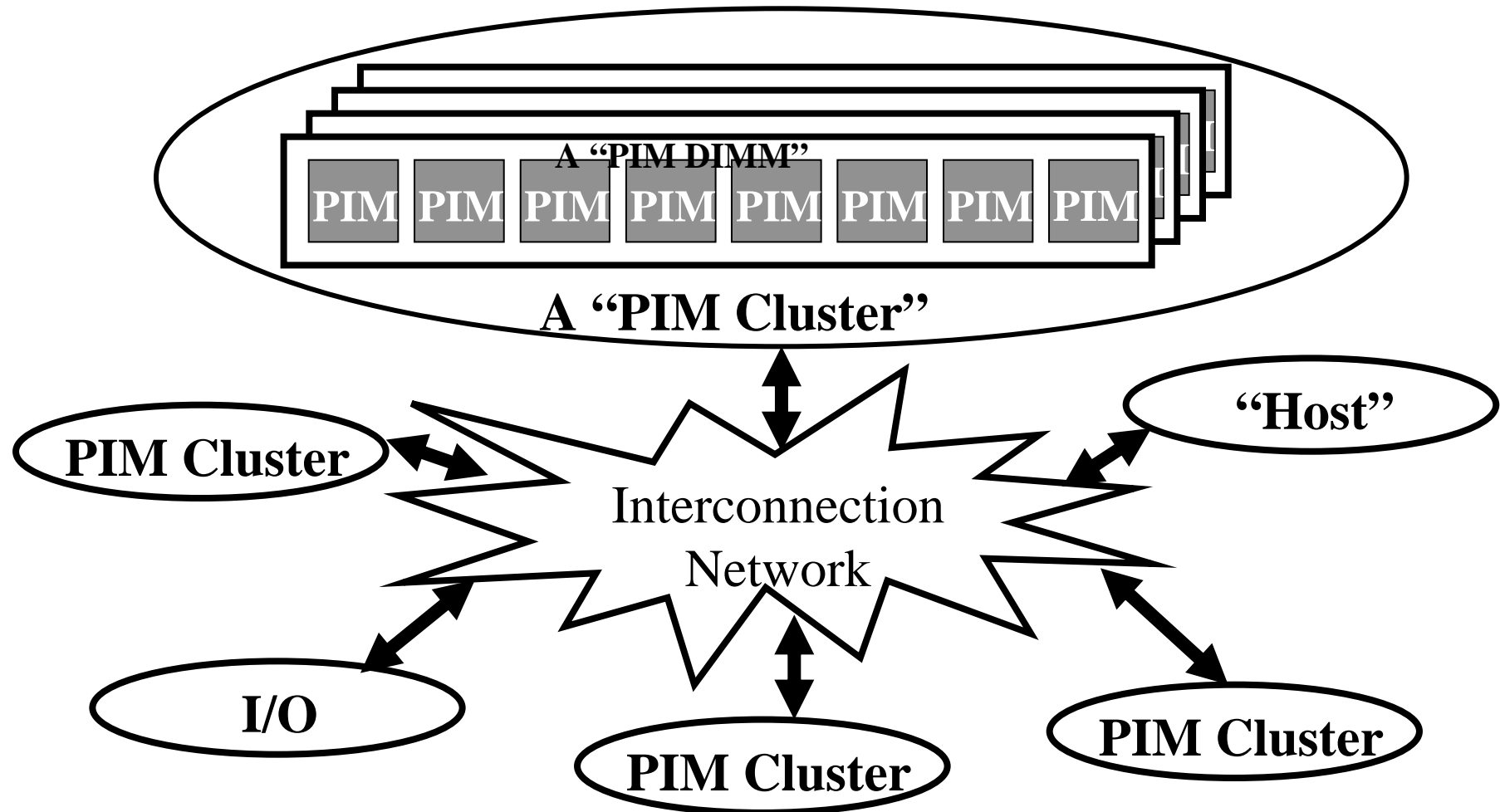
- “Overprovision” memory with huge numbers of anonymous processors
  - Each multi-threaded
- Reduce state of a thread to ~ a cache line
- Make creating a new thread “near” some memory a cheap operation
- Allow thread to “move” to new site when locality demands

Latency reduced by *huge* factors





## Next: An “All-PIM” Supercomputer







---

# Summary





# Summary

---

- When it comes to silicon: *It's the Memory, Stupid!*
- Technology scaling progressing at uneven rates
  - Transistor density continuing improvement
  - Power limiting clock rate growth
  - Voltage improvement slowing
  - Off-chip I/O becoming a killer
- Today's solution: multi-core, multi-threaded uP dies
  - Increases # of threads per core
  - But doesn't solve bandwidth to memory problem
- State bloat consumes huge amounts of silicon
  - That does no useful work!
  - And all due to focus on “named” processing logic





# How Might We Make It Better?

---

- **Reduce thread state**
  - Cost of moving/copying state => line reference
- **Relentless multi-threading execution models**
- **Simplify cores and “overprovision”**
  - “Pitch-match” to memory macro
- **Focus on “cheap” logic in dense memory fab process**
  - Don’t fret the clock rate
- **Change execution model from “named” core to anonymous core “nearest” memory object**
  - A “Traveling Thread” need never “wait” for processing resources
  - Convert two way latencies to one way