Review Approval

- 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 : 5225758 SAND Number : 2005-6499 P

Review Type : Electronic Status : Approved

Sandia Contact : DEBENEDICTIS,ERIK P. Submittal Type : Viewgraph/Presentation

Requestor : DEBENEDICTIS,ERIK P. Submit Date : 10/12/2005

Comments : This is a release for a Tutorial. The tutorial will comprise viewgraph presentations and other material.

Peer Reviewed? : N

Author(s)

David Keyes DEBENEDICTIS,ERIK P. DEBENEDICTIS,ERIK P.

Peter M Kogge

Event (Conference/Journal/Book) Info

Name : Supercomputing 2005 -- Tutorial Session

City : Seattle State : WA Country : USA

Start Date : 11/14/2005 End Date : 11/14/2005

Partnership Info

Partnership Involved : Yes

Partner Approval : Yes 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	PUNDIT,NEIL D.	PUNDIT,NEIL D.	10/13/2005
Conditions:			
Administrator Approver	LUCERO,ARLENE M.	KRAMER,SAMUEL	09/05/2007
Please add the funding statement: Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.			

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**

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

Extended Outline
SC|05 Tutorial M09

Erik DeBenedictis

David Keyes

Peter Kogge

Tutorial Goals

- Explore key issues in future of supercomputing
  - Algorithms, technology, architecture
- Motivate changes based on problem space
- Drive discussion based on "Moore's Law"
- Explore meaning of silicon's endpoints
- Discuss potential alternatives
- Use concept of *scaling* throughout
- Combine with "hands-on" participant-based projections
- Provide an overview of successor technologies

Definitions of Scaling

- A dry thin flake of epidermis shed from the skin
- To remove in layers or scales
- (Australian): To ride … without paying the fare.
- A progressive classification, as of size, amount, importance, or rank
- To alter according to a standard or by degrees; adjust in calculated amounts
- the act of arranging in a graduated series

From dictionary.com

Schedule

- 8:30 Introductory Comments
- 8:45 Algorithm Scalability
  - Review from Scales
  - Mesh Example
- 10:00 Break
- 10:30 Silicon Scaling
  - ITRS Roadmap
  - Microprocessors and Alternative Architectures
- Noon Lunch
- 1:30 System Scaling
  - End of the Roadmap
  - Projecting Applications Performance on Future Supercomputers
- 3:00 Break
- 3:30 Hands-On Exercises
- 4:30 Beyond Transistors
- 5:00 Conclusion

Review of Applications from SCaLeS

- Large-scale simulation going through phase change
- Complementary roles of algorithmic and architectural advances
- Lessons from recent Gordon Bell prizes
- Some simulation priorities and opportunities at and beyond the terascale
  - Magnetic fusion energy
  - Combustion
  - Climate
  - Astrophysics
  - Accelerator design
  - Lattice QCD

Algorithm Scalability – Mesh-based Example

- Application Models
  - Mesh-based algorithms
    - Discretizations
    - Solvers
    - Software
  - Resource scaling for mesh-based applications
  - Mesh-based kernels and architectural stress points
- Architectural Models
  - Key parameters
    - Processor
    - Memory system
    - Communication network
  - Estimating performance scalability
  - Opportunities for improving algorithm-architecture impedance match

ITRS Roadmap and Device Scaling

- MOSFET Geometry
  - Gates
  - Memory cells
- CMOS Scaling Laws (a la Mead and Conway)
- Scaling examples
  - μPs
  - Memories
  - Memory Bandwidth
  - Node-to-node communications rate

Scaling of μPs and Advanced Architectures

- Multi-Core Processors
  - Trading IPC for explicit parallelism
  - Core scaling
  - Bandwidth scaling
- Multi-threading Architectures
  - Latency hiding
  - Introducing locality-awareness & latency avoidance
- Processor in Memory Architectures
  - Latency and bandwidth scaling
  - PIM Bump and implications to inner loop memory requirements

End of the Roadmap

- ITRS: Uniform exponentials or something else?
  - SPEC processor numbers and implications
  - Total power off track
  - Some hint of clock rate problems
- Review of Burger and Keckler Study
  - Study of throughput under technology scaling
- Implications
  - Throughput scaling
  - Cache scaling
  - Bandwidth Scaling

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

Hands-On Exercises

- Organization
  - Leaders will go through a sample problem with the group first
  - Group divides into sections of 3-6 people each
  - Will hand out pertinent sections of ITRS and applications reference materials
  - Specific problems will be determined by the interests of the groups, with some sample problems given below:
- Problem #1: Project parameters of a $10M supercomputer in year 2016
- Problem #2: Project performance of supercomputer above on a legacy application
- Problem #3: Performance on mesh application
- Problem #4: Project parameters of a PIM architecture supercomputer

Beyond Transistors

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

Tutorial 123: Impact of Moore’s Law and Architecture on Application Performance, Session I: *Opportunities to Advance Science through Supercomputer Simulation*

David Keyes, Columbia University

Role of presentation

- Remind ourselves of some prime science and engineering customers
- Look anecdotally at a few demanding applications
  - SciDAC: climate, QCD, accelerator design, magnetic fusion energy, combustion, astrophysics
  - Bell: mechanics, seismology, aerodynamics
  - Race through the picture gallery – no time for the science, itself
- Look generically at PDE-based simulation and the basis of continued optimism for its growth – capability-wise
- Look at some specific hurdles posed by high-end architecture

Technical aspects of presentation

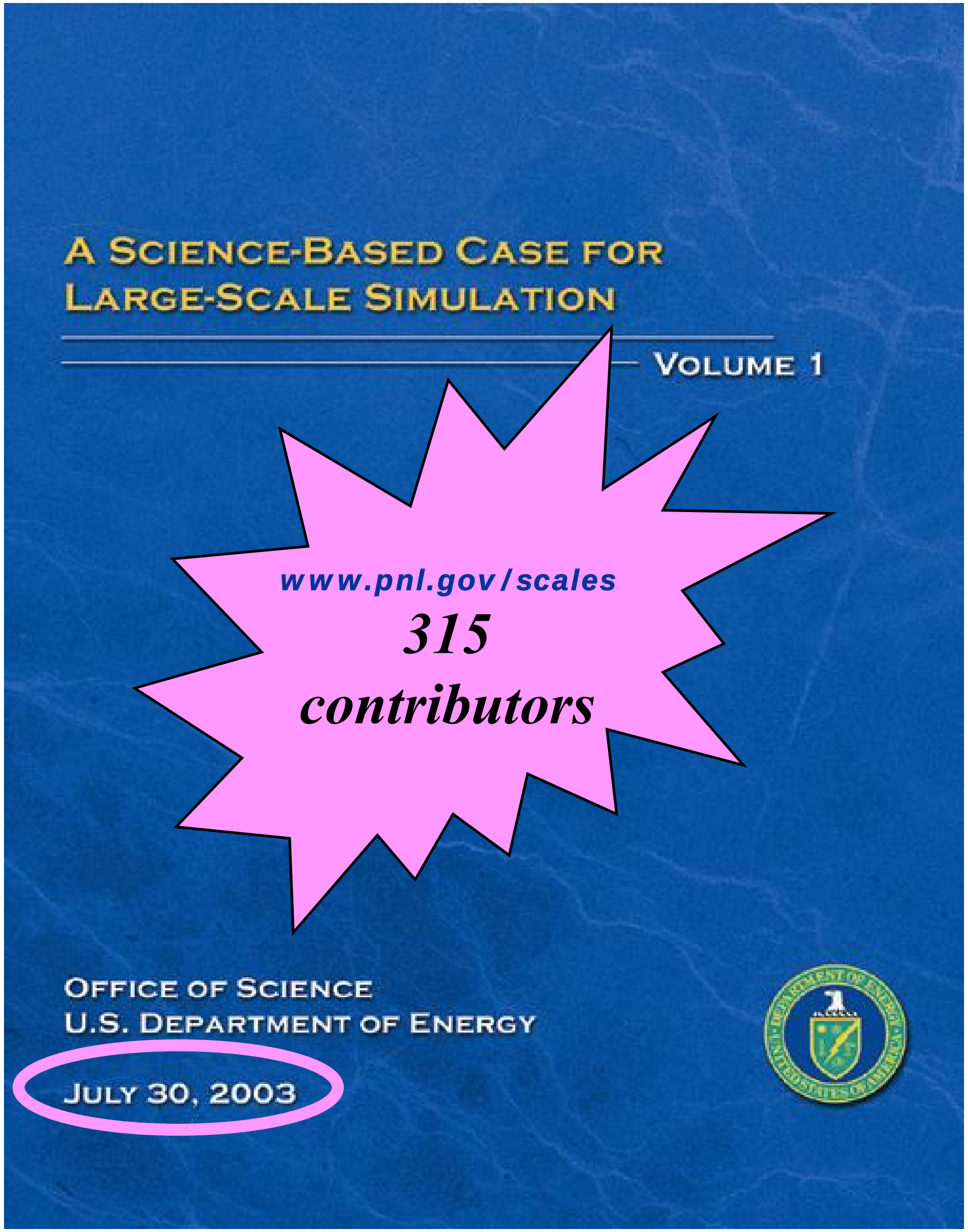
- Introduce a parameterized highly tunable class of algorithms for parallel implicit solution of PDEs
  - understand the source of its "weak scalability"
  - ignore other numerical analysis aspects, here
- Note some algorithmic "adaptations" to architectural stresses

Philosophy of presentation

- Applications are ***given***
- Architectures (hardware and software) are ***given***
- Algorithms ***must be created*** to bridge to hostile architectures for the sake of the applications
- Knowledge of algorithmic capabilities can usefully influence
  - the way applications are formulated
  - the way architectures are constructed

Context: recent reports promote simulation

- Cyberinfrastructure (NSF, 2003)
  - new research environments through cyberinfrastructure
- Facilities for the Future of Science (DOE, 2003)
  - “ultrascale simulation facility” ranked #2 in priority (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, 2005)
  - opportunities in dynamic, data-driven simulation and engineering design
- **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**



A SCIENCE-BASED CASE FOR
LARGE-SCALE SIMULATION

VOLUME 1

www.pnl.gov/scales

315 contributors

OFFICE OF SCIENCE
U.S. DEPARTMENT OF ENERGY

JULY 30, 2003

- 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

Gedanken experiment: How to use a jar of peanut butter as its price slides downward?

- In 2005, at $3.20: make sandwiches
- By 2008, at $0.80: make recipe substitutions for other oils
- By 2011, at $0.20: use as feedstock for biopolymers, plastics, etc.
- By 2014, at $0.05: heat homes
- By 2017, at $0.0125: pave roads ☺

NISTSRM 2387
1028
STANDARD REFERENCE MATERIAL®
2387
Peanut Butter
www.nist.gov/srm
NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

The cost of computing has been on a curve *much better than this* for two decades and promises to continue for at least one more. Like everyone else, scientists should plan increasing uses for it…

Gordon Bell Prize “price performance”

Year	*Application*	*System*	*$ per Mflops*
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 chromodyn	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

Price/performance has stagnated and no new such prize has been given since 2001.

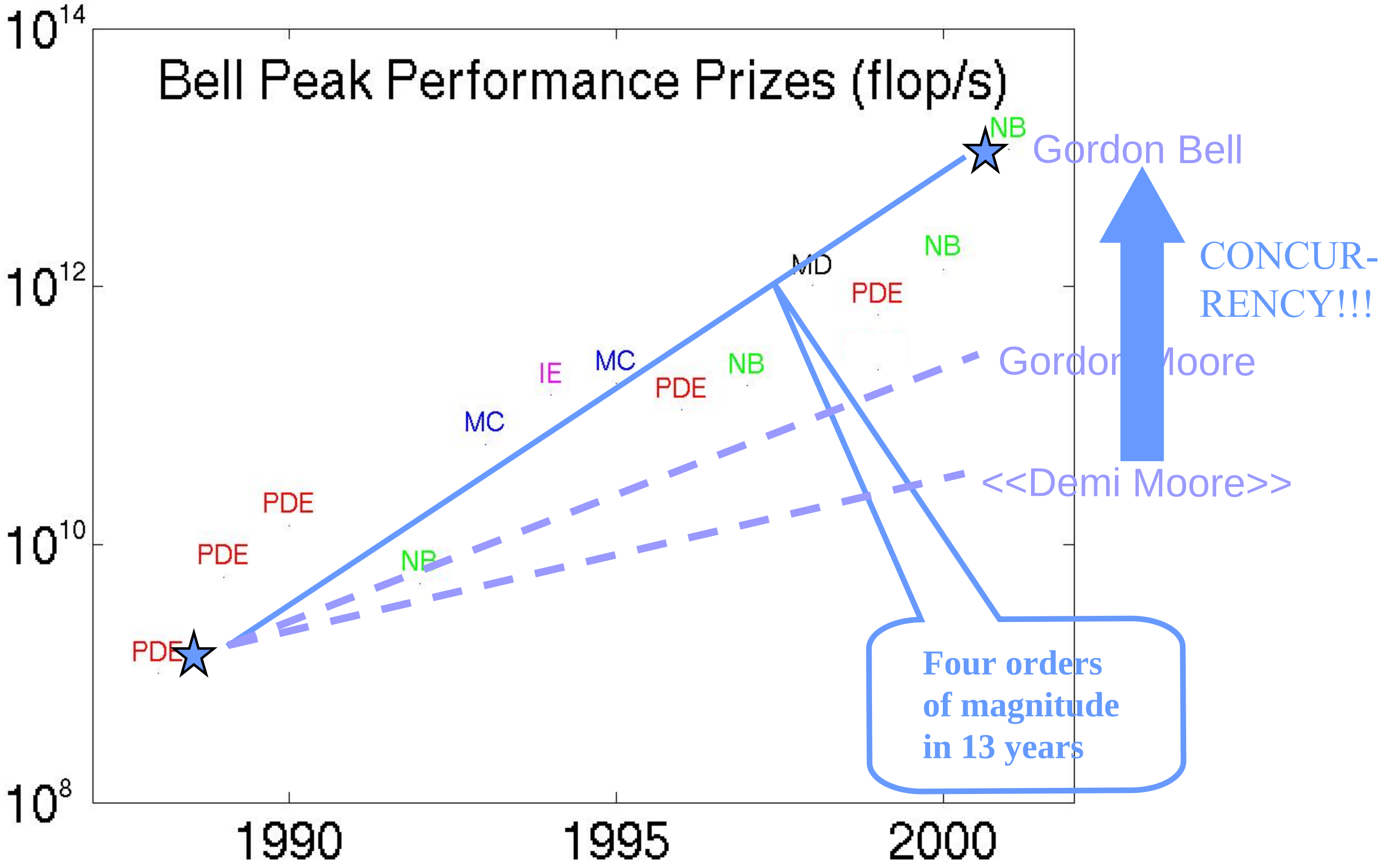
Gordon Bell Prize “peak performance”

Year	*Type*	*Application*	*No. Procs*	*System*	*Gflop/s*
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

Four orders of magnitude in 13 years

With 100 Tflop/s in 2005, peak performance on real applications continues on its trajectory!

Gordon Bell Prize outpaces Moore’s Law



Gordon Bell

CONCUR-
RENCY!!!

Gordon Moore

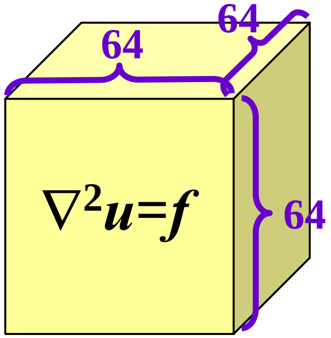
<<Demi Moore>>

Four orders of magnitude in 13 years

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

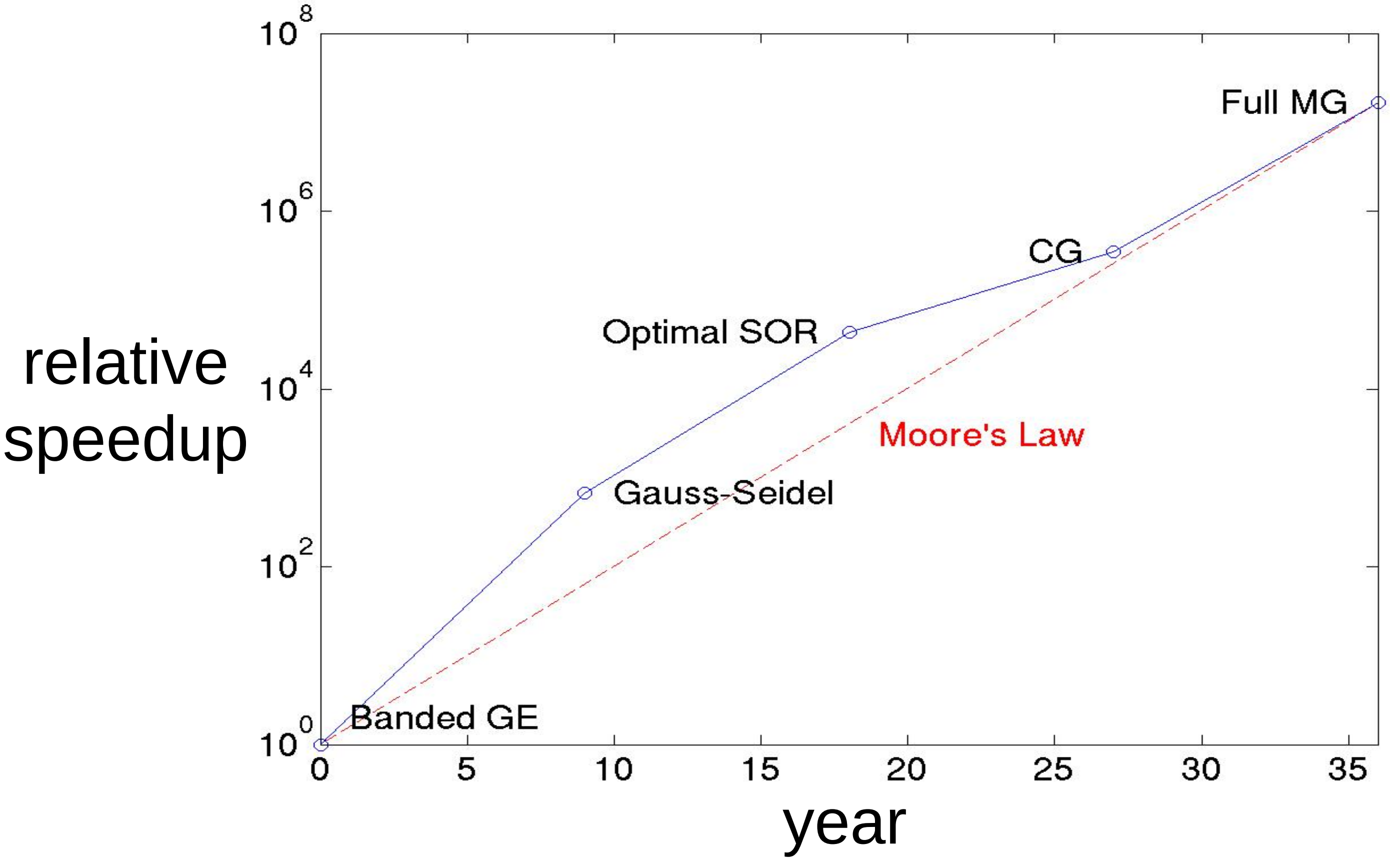


- If n=64, this implies an overall reduction in flops of ~16 million *

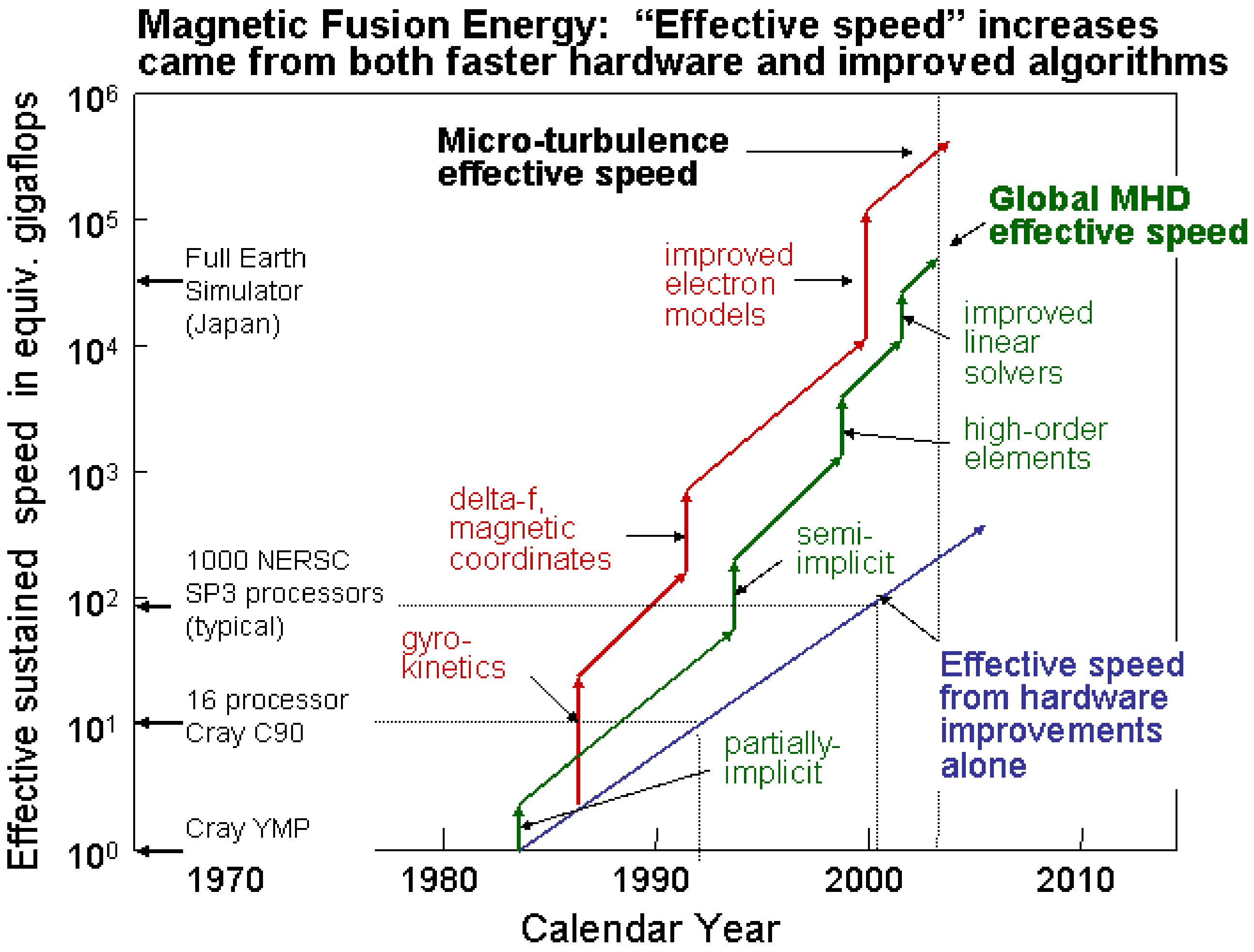
*Six-months is reduced to 1 s

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 ⇒ the same as the factor from algorithms alone!



“Moore’s Law” for MHD simulations



“Semi-implicit”:

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

“Partially implicit”:

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

Figure from SCaLeS report, Volume 2

“Moore’s Law” for combustion simulations

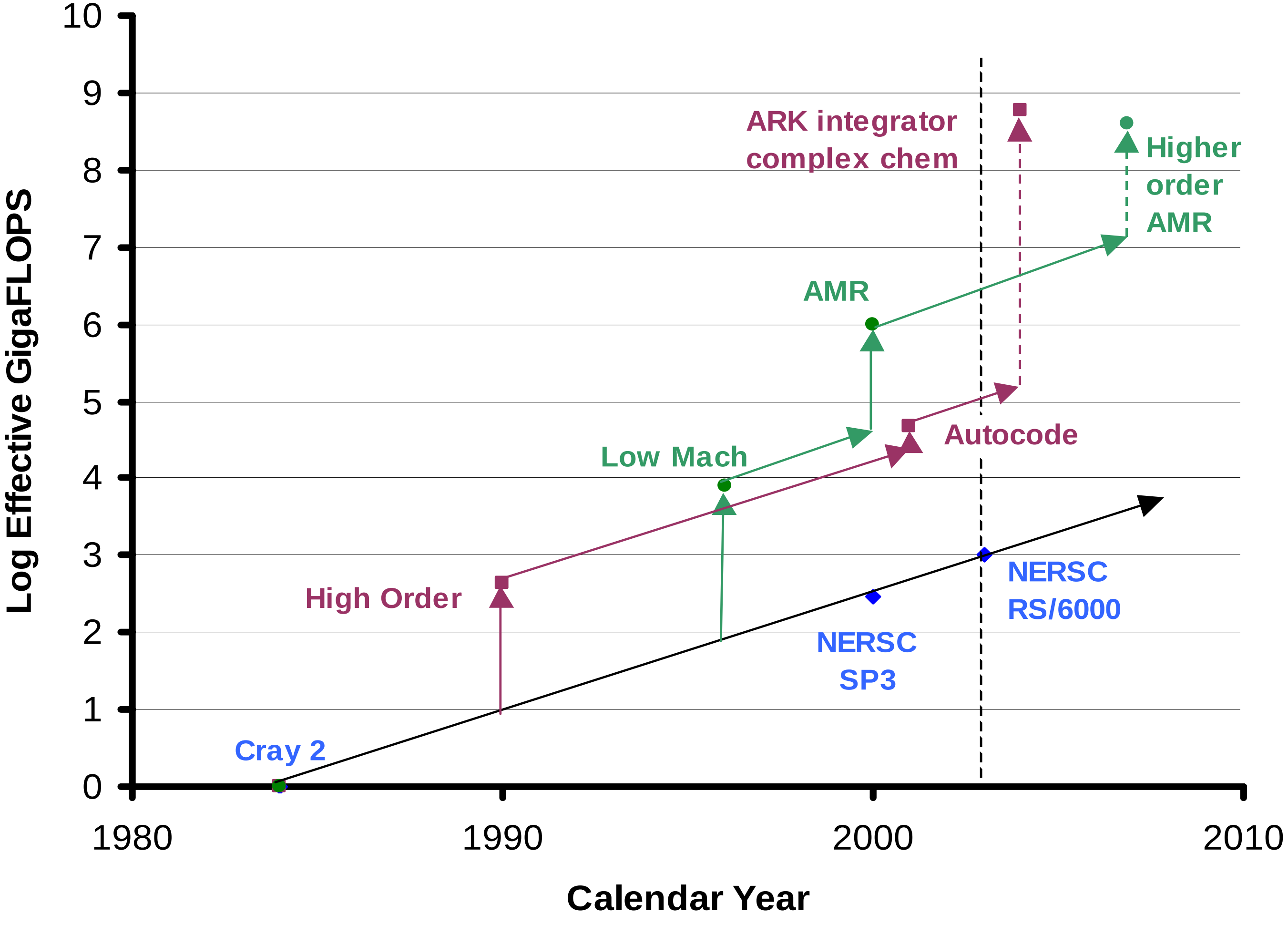
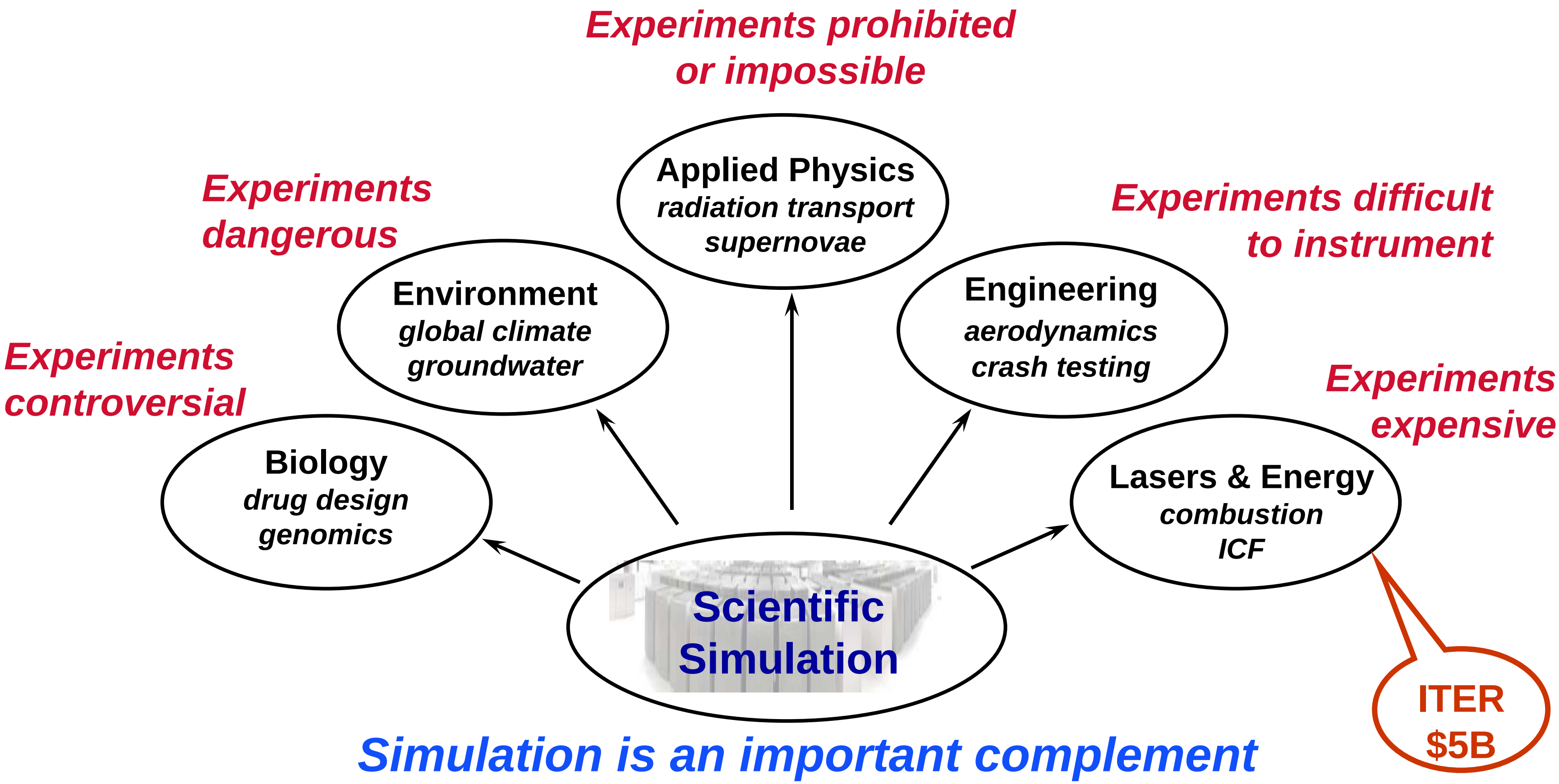


Figure from SCaLeS report, Volume 2

Terascale simulation can be pitched as an alternative to experimentation



Experiments prohibited or impossible

Applied Physics
radiation transport
supernovae

Experiments dangerous

Environment
global climate
groundwater

Experiments difficult to instrument

Engineering
aerodynamics
crash testing

Experiments controversial

Biology
drug design
genomics

Experiments expensive

Lasers & Energy
combustion
ICF

Scientific Simulation

ITER
$5B

Simulation is an important complement to experiment in many areas

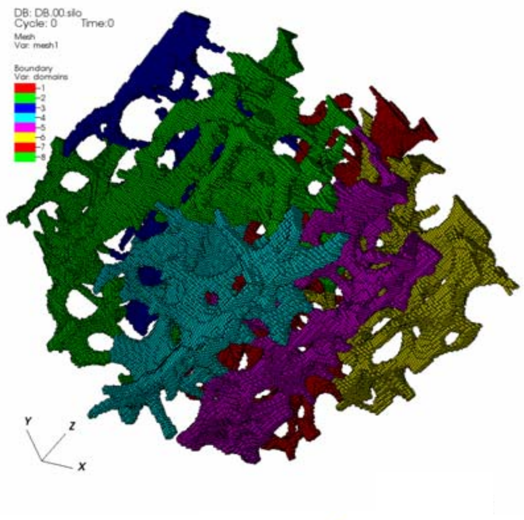
Heretofore difficult apps are now parallelized

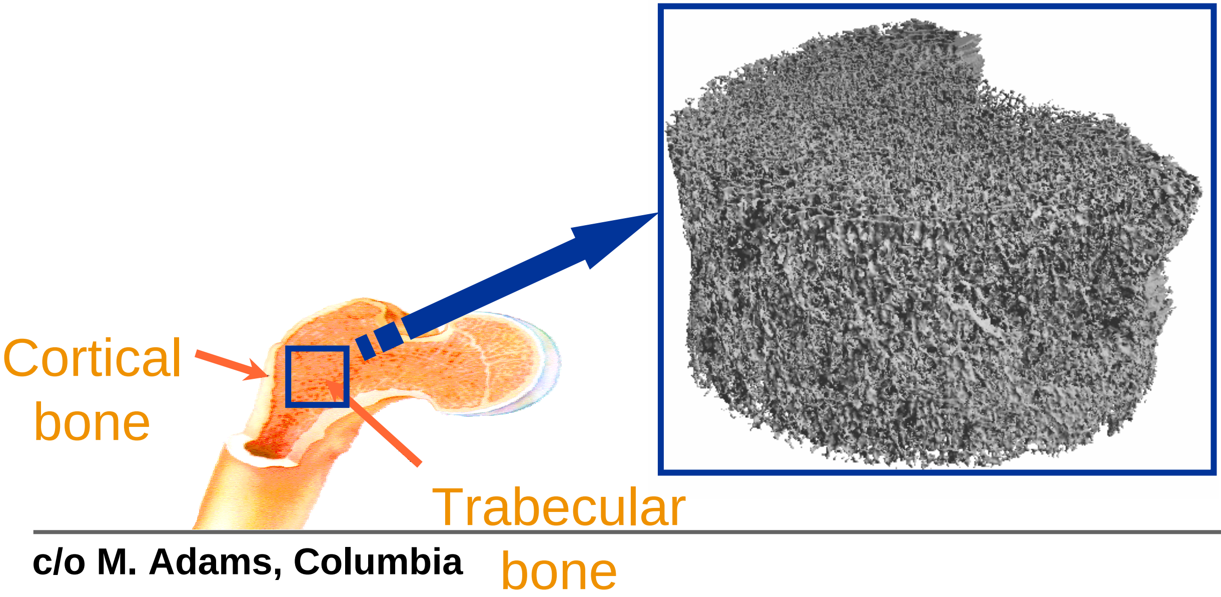
- Unstructured grids
- Implicit, as well as explicit, methods
- Massive spatial resolution
- Thousand-fold concurrency
- Strong scaling within modest ranges
- Weak scaling without obvious limits

See, e.g., Gordon Bell "special" prizes in recent years …

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



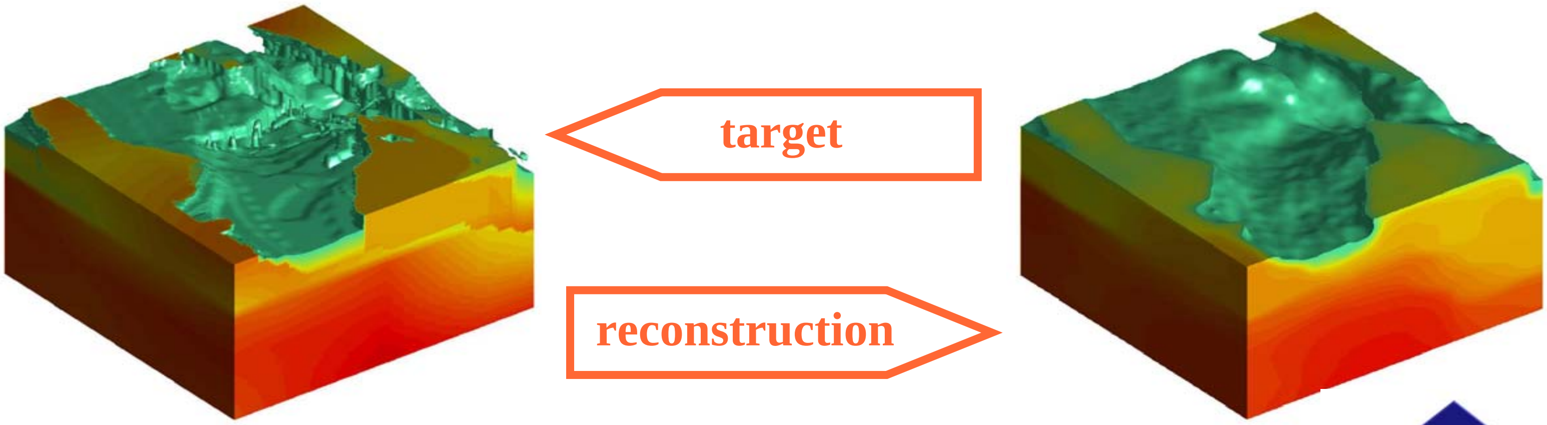


TOPS
Terascale Optimal PDE Simulations

c/o M. Adams, Columbia

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



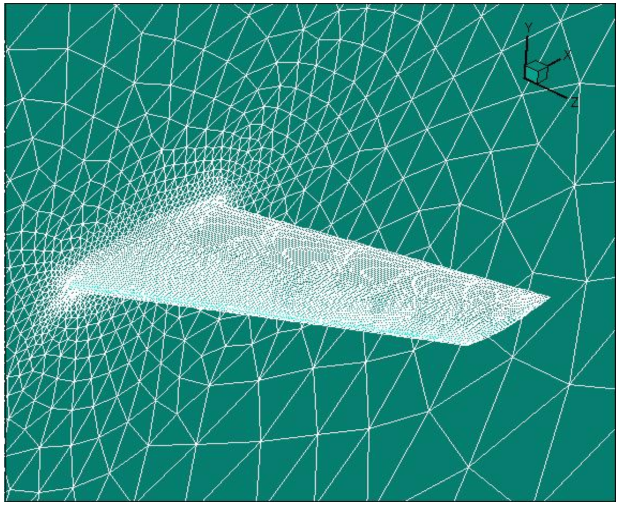
target

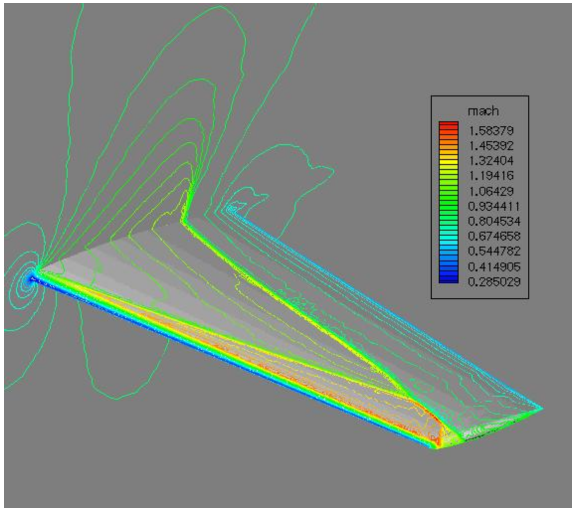
reconstruction

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



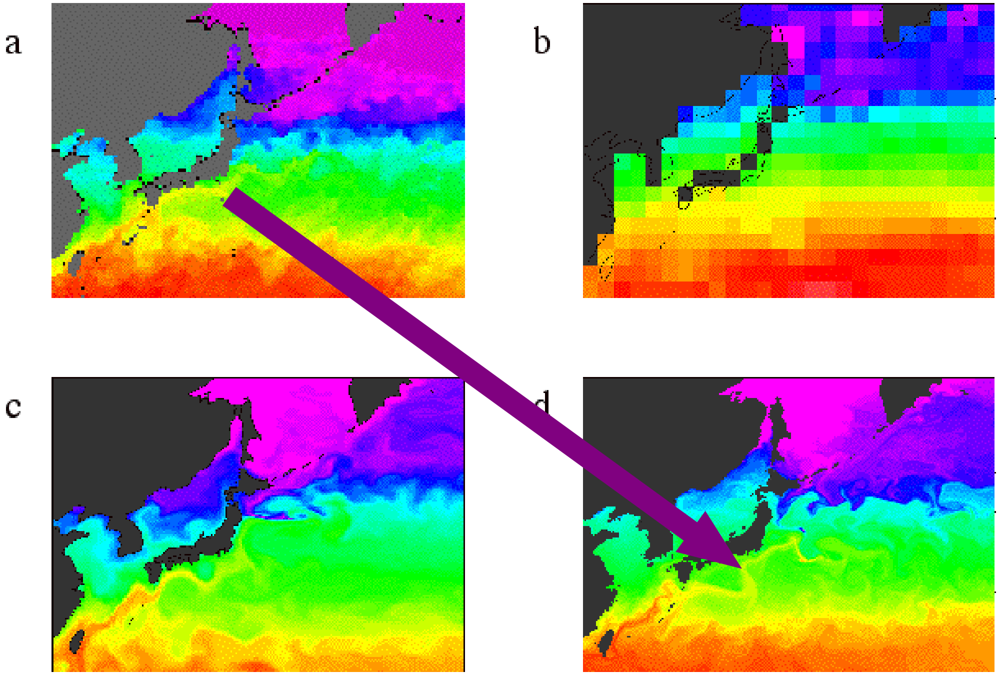


What would scientists do with 100-1000x? Example: predict future climates

- Resolution
  - refine atmospheric resolution from 160 to 40 km
  - refine oceanic resolution 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 ~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°, (c) 0.28°, and (d) 0.1°.

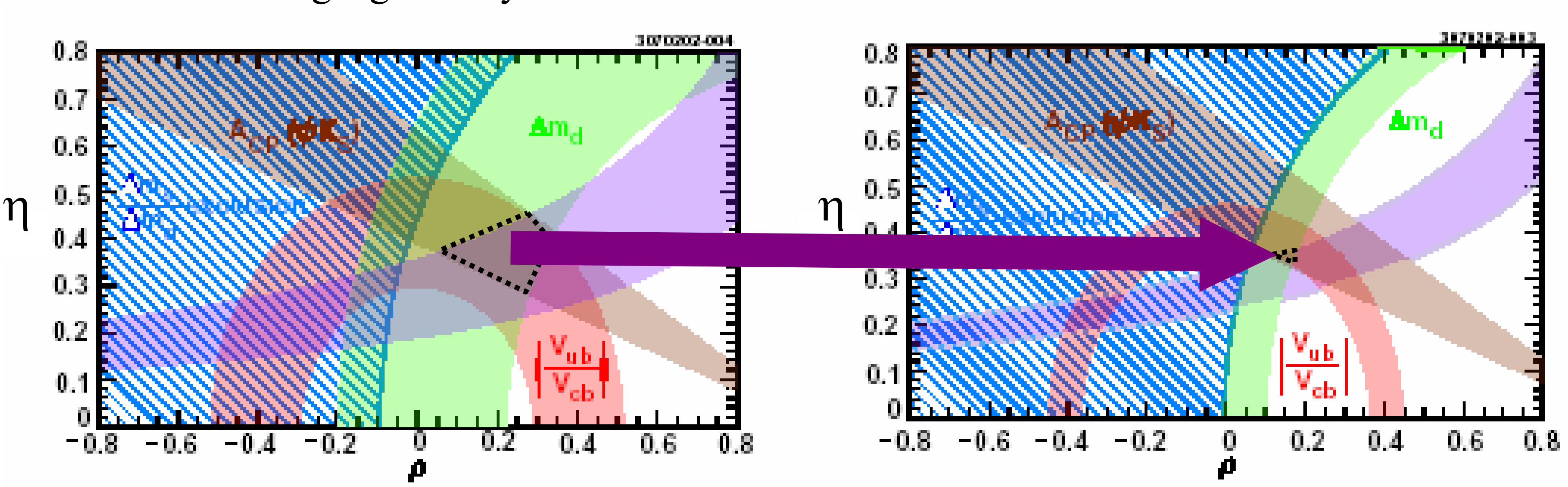


What would scientists do with 100-1000x? Example: probe structure of particles

- Resolution
  - take current 4D quantum chromodynamics models from 32×32×32×16 to 128×128×128×64
- 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

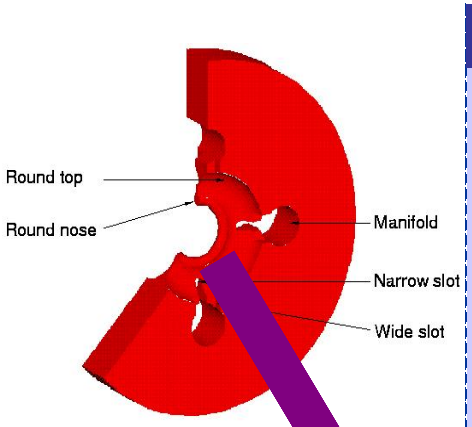
Constraints on the Standard Model parameters ρ and η. For the Standard Model to be correct, these parameters from the Cabibbo-Kobayashi-Maskawa (CKM) matrix must be restricted to the region of overlap of the solidly colored bands. The figure on the left shows the constraints as they exist today. The 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%.



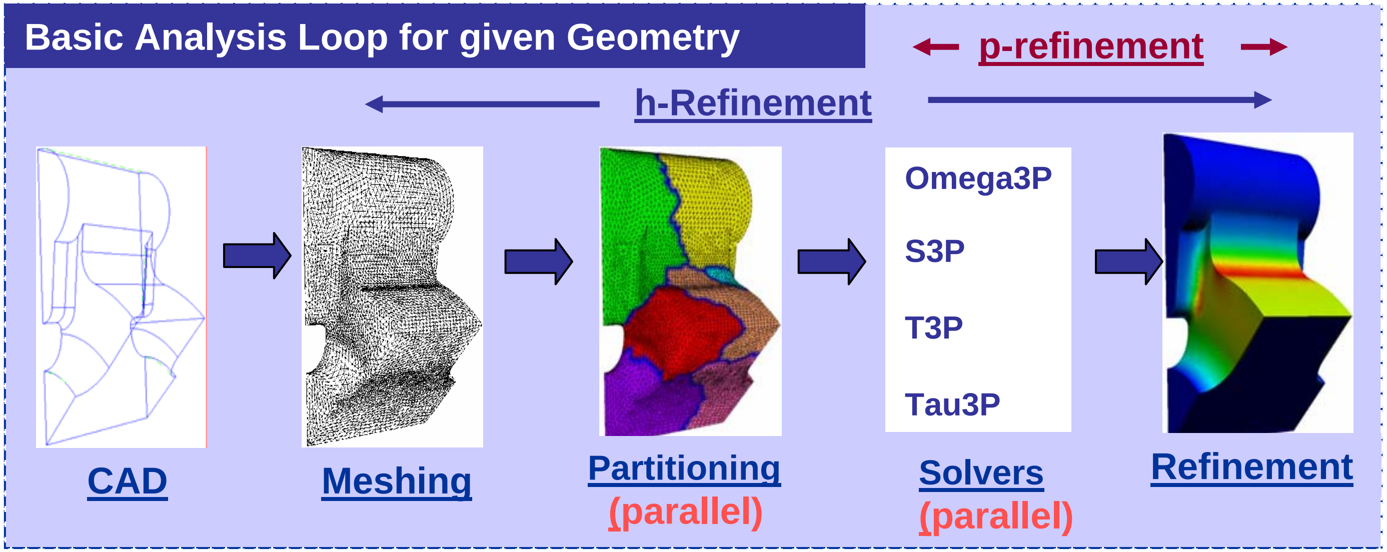
What would scientists do with 100-1000x? Example: design accelerators

- 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



DDS CELL



Basic Analysis Loop for given Geometry

← p-refinement →

← h-Refinement →

CAD → Meshing → Partitioning (parallel) → Solvers (parallel) → Refinement

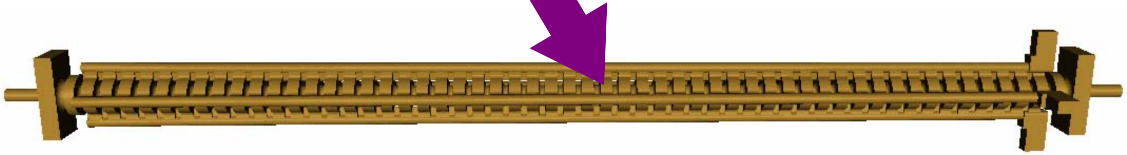
Omega3P

S3P

T3P

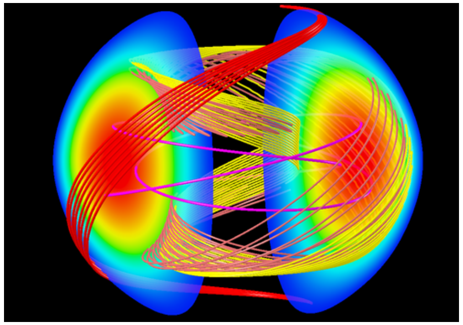
Tau3P

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

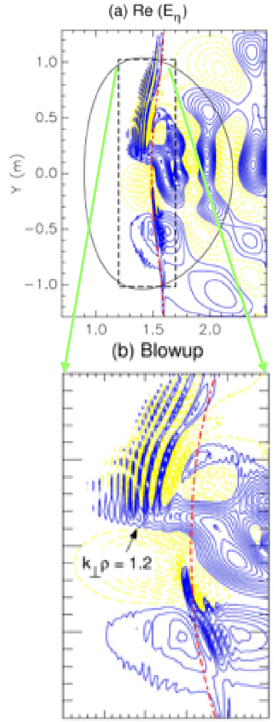


c/o K. Ko, SLAC

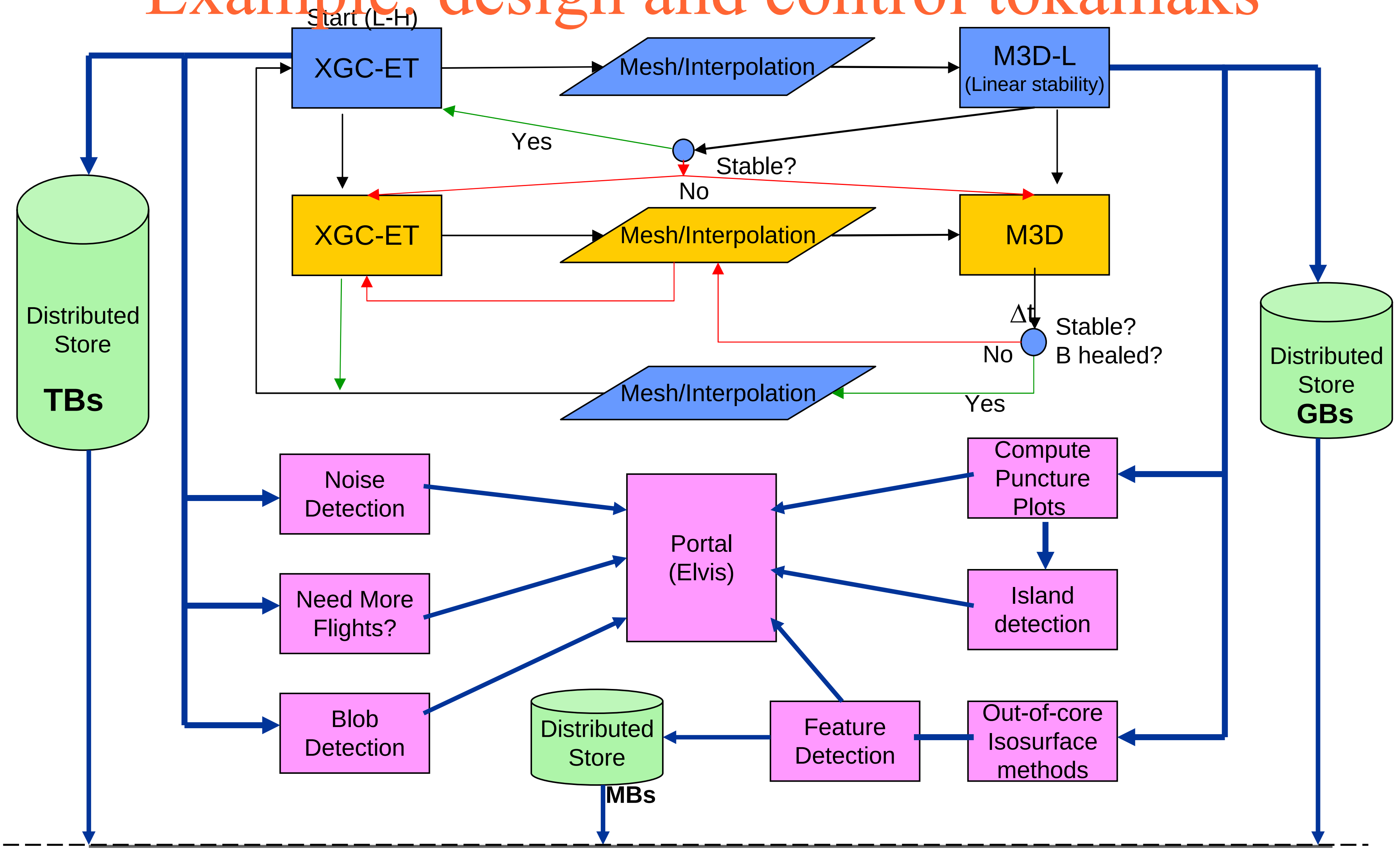
What would scientists do with 100-1000x? Example: design and control tokamaks



- 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



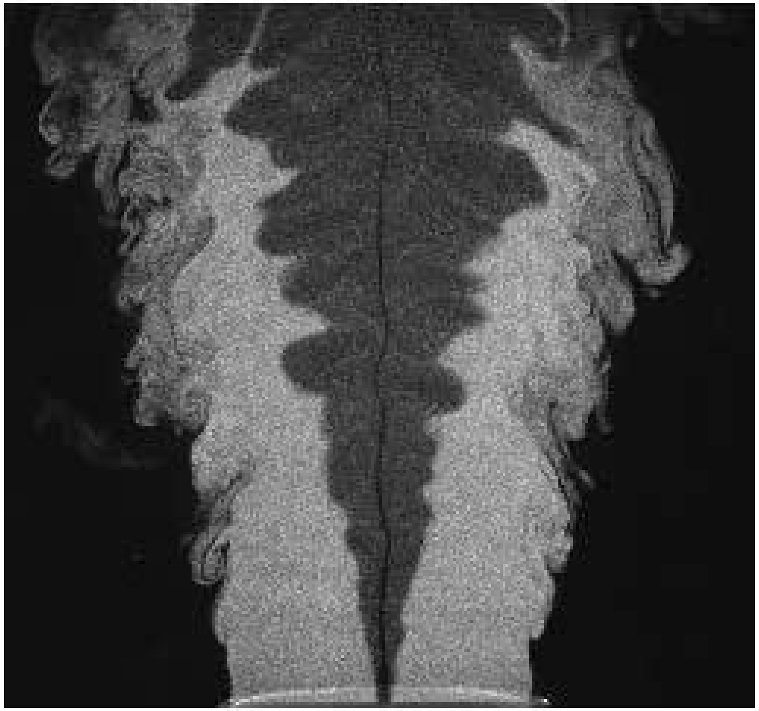
c/o S. Klasky, ORNL

IDAVE

What would scientists do with 100-1000x? Example: control combustion

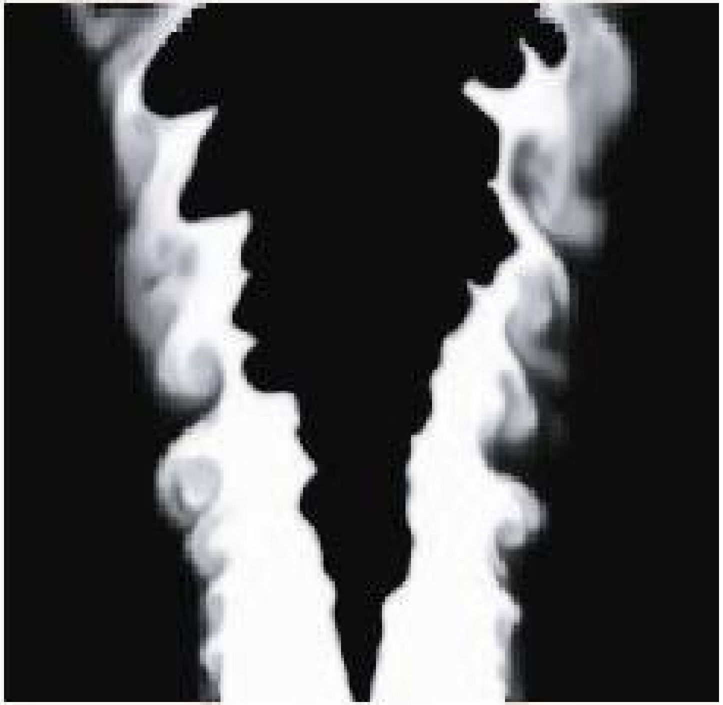
- Resolution
  - evolve 3D time-dependent large-eddy simulation (LES) codes to direct Navier-Stokes (DNS)
  - multi-billions of mesh zones required
- New "physics"
  - explore coupling between chemistry and acoustics (currently filtered out)
  - explore sooting mechanisms to capture radiation effects
  - 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

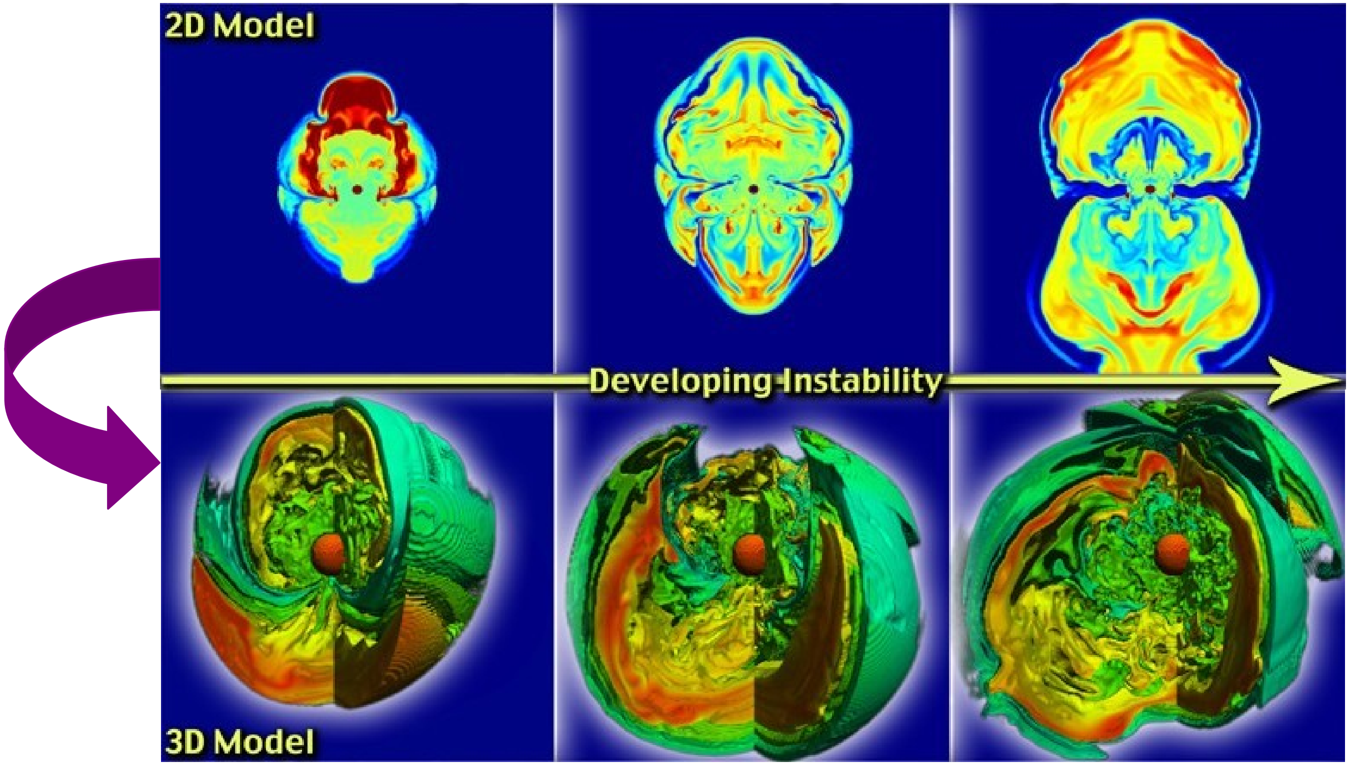
Images c/o R. Cheng (left), J. Bell (right), LBNL, and NERSC

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 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

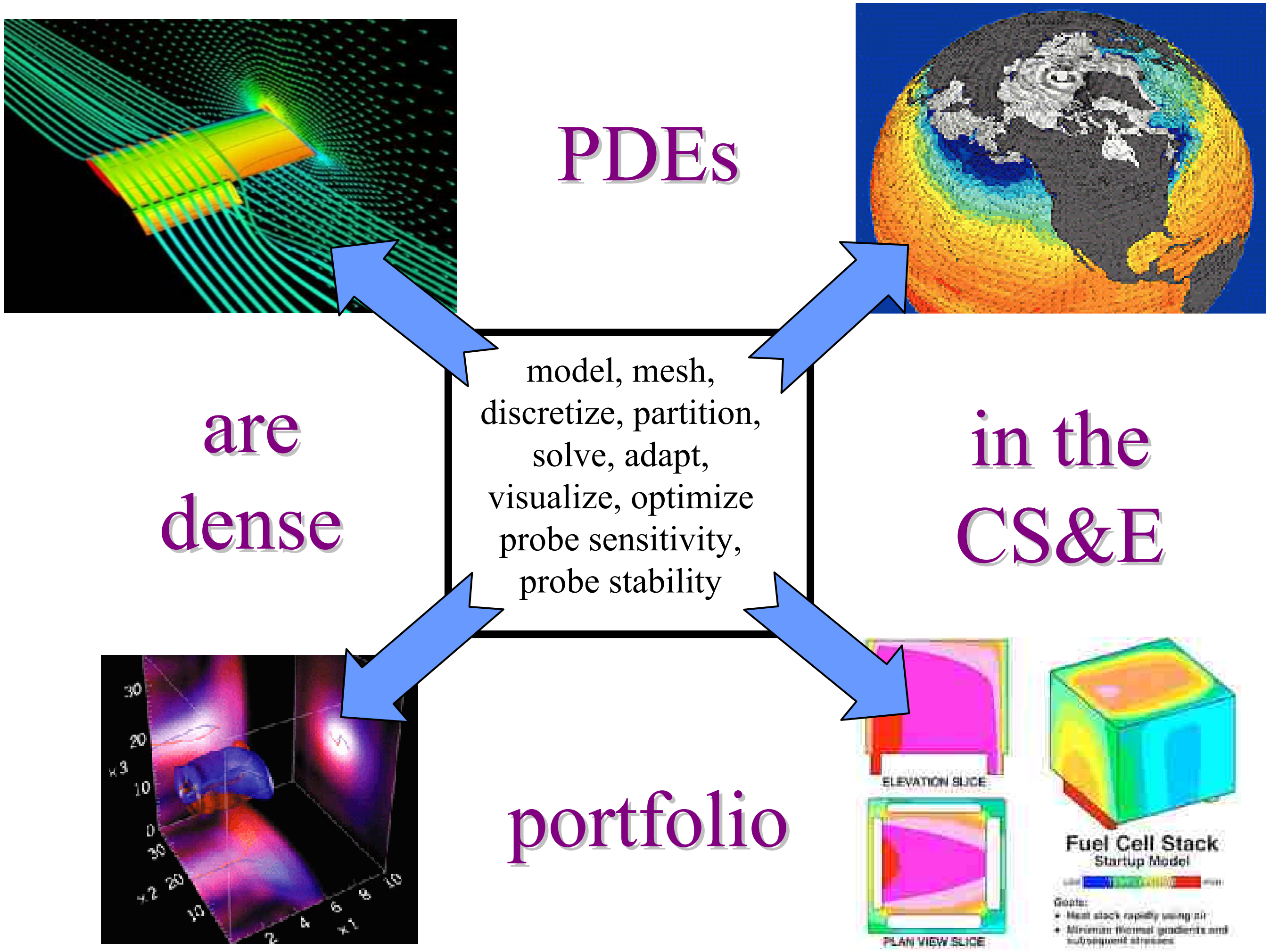
What would scientists do with 100-1000x? Example: probe supernovae



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.

c/o A. Mezzacappa, ORNL

"The partial differential equation entered theoretical physics as a handmaid, but has gradually become mistress." – **A. Einstein**



PDEs

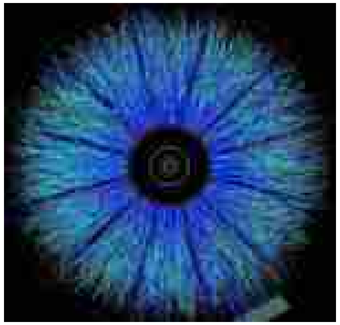
are dense

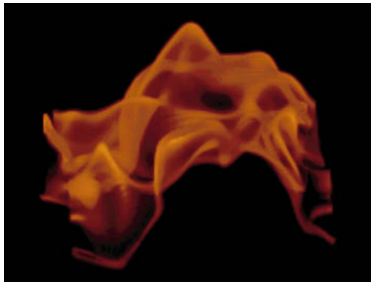
in the CS&E

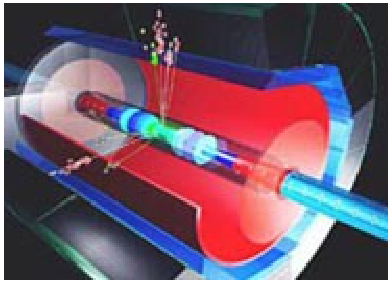
portfolio

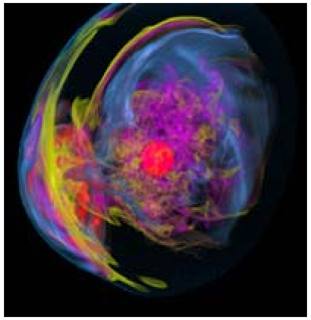
model, mesh,
discretize, partition,
solve, adapt,
visualize, optimize
probe sensitivity,
probe stability

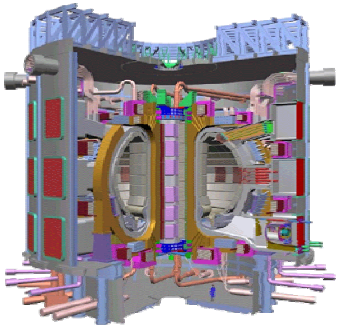
It’s *not* about the solver

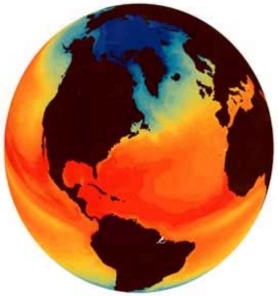




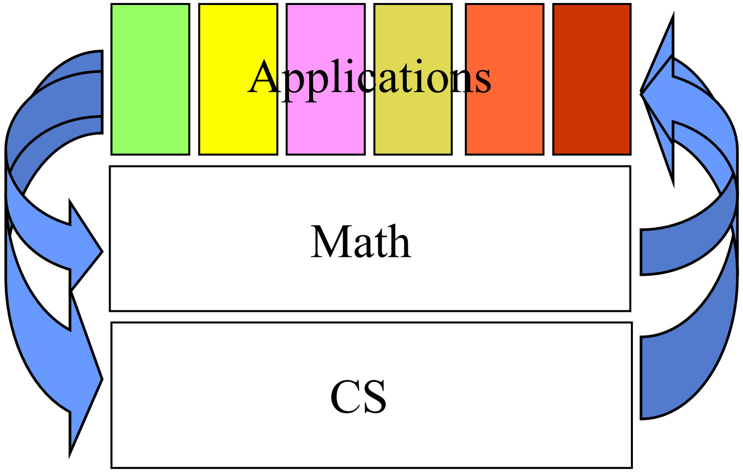








Applications drive



Applications

Math

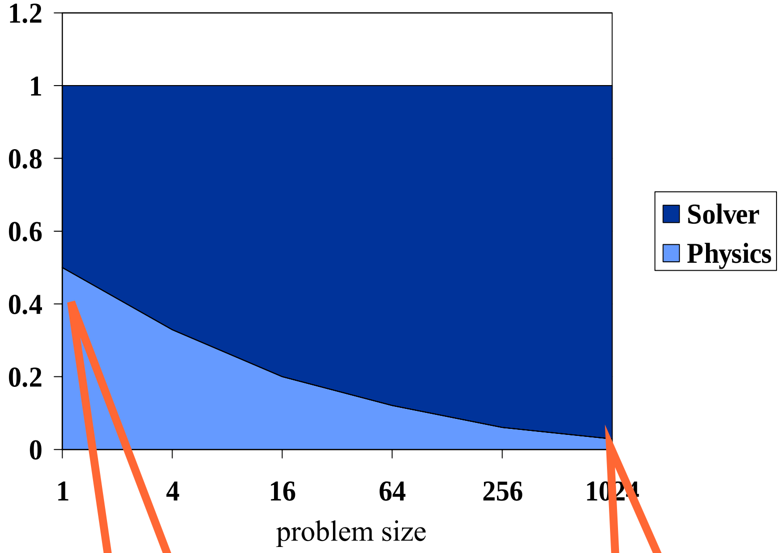
CS

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

Weak scaling limit, assuming efficiency of 100% in both physics and solver phases



Solver takes 50% time on 64 procs

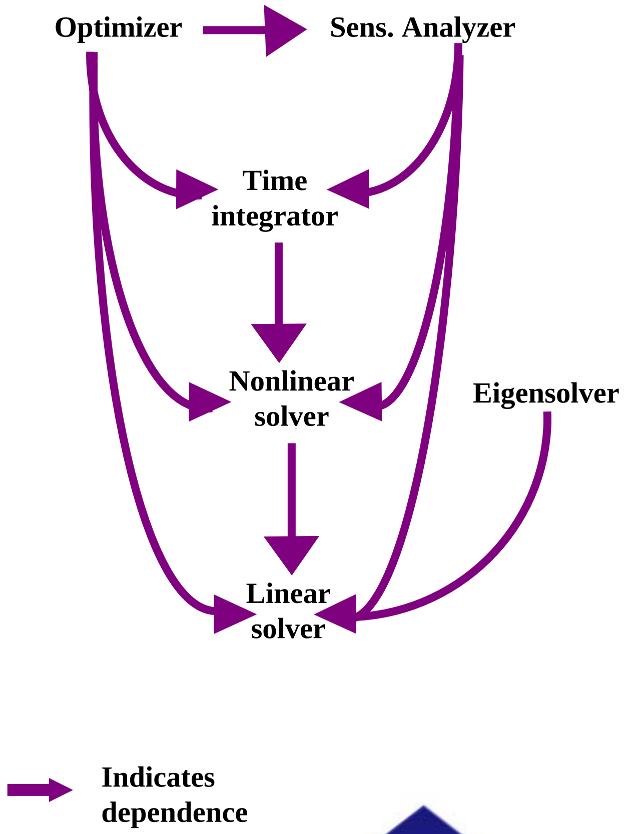
Solver takes 97% time on 64K procs

A central concept: solver toolchain

- From *solutions* to *sensitivity, stability, optimization*
- Nested modules
- Leveraged implementation of distributed data structures
- Hiding of communication and performance-oriented details so users deal with mathematical objects throughout

Solver software toolchain

- Design and implementation of "solvers"
  - Linear solvers $$Ax = b$$
  - Eigensolvers $$Ax = \lambda Bx$$
  - Nonlinear solvers **(w/ sens. anal.)** $$F(x, p) = 0$$
  - Time integrators **(w/ sens. anal.)** $$f(\dot{x}, x, t, p) = 0$$
  - Optimizers
$$\min_{u} \phi(x,u) \; s.t. \; F(x,u) = 0, u \geq 0$$
- Software integration
- Performance optimization



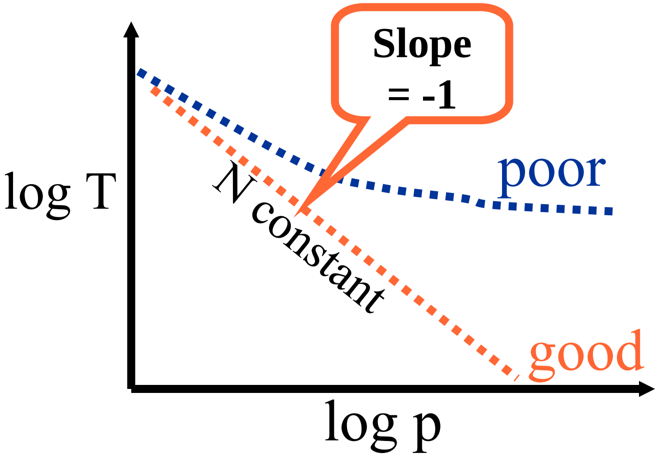
TOPS

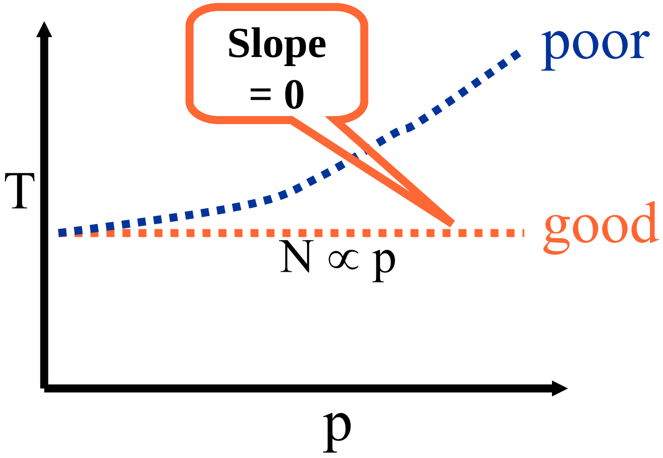
Terascale Optimal PDE Simulations

Two definitions of scalability

- "Strong scaling"
  - execution time decreases in inverse proportion to the number of processors
  - *fixed size problem overall*

- "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"





Preview: 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

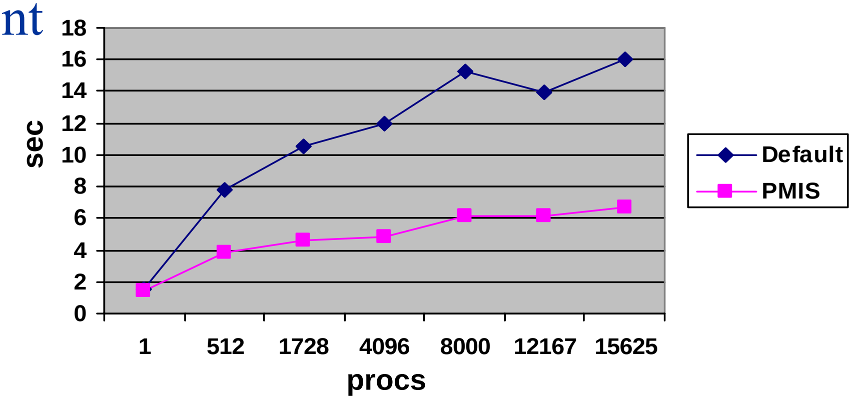


Figure shows weak scaling result for AMG out to 16K processors, with one 30× 30×30 block per processor (from 27K dofs up to 422M dofs)

c/o U. M. Yang, LLNL

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
    - 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

- 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 $\propto f_1 + (1 - f_1)/p$
- Speedup $= 1/[f_1 + (1 - f_1)/p]$

  [Table shows example for f_1 of 1%]

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

- Applies to any performance enhancement, not just parallelism

Resolution-limited progress

- Illustrate for CFL-limited time stepping
- Parallel wall clock time

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

- Example: explicit wave problem in 3D (α*=1, d=3*)

Domain	$10^3\times 10^3\times 10^3$	$10^4\times 10^4\times 10^4$	$10^5\times 10^5\times 10^5$
Time	1 day	10 days	3 months

- Example: explicit diffusion problem in 2D (α*=2, d=2*)

Domain	$10^3\times 10^3$	$10^4\times 10^4$	$10^5\times 10^5$
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

τ 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

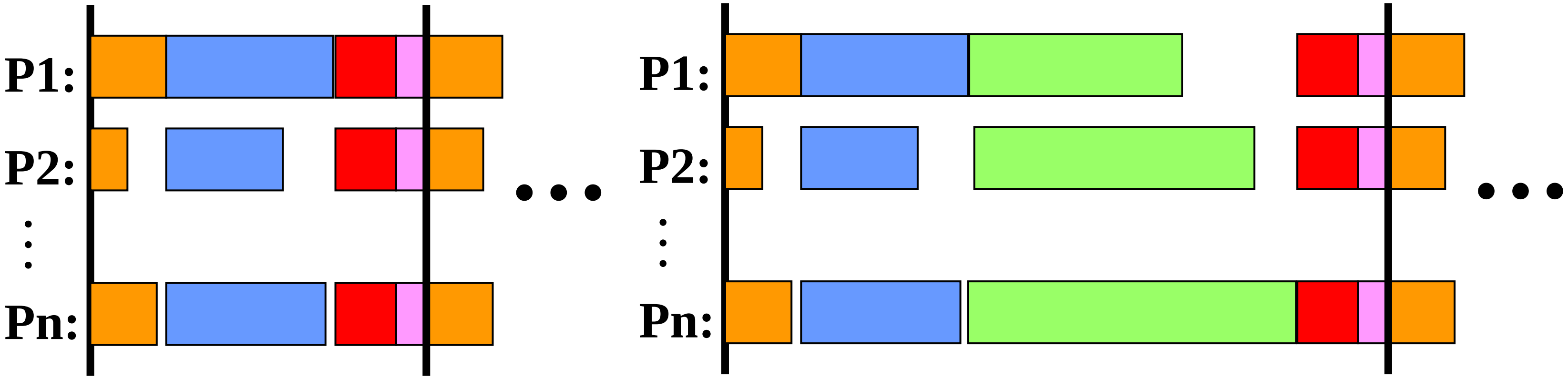
$O(MN/P)$ parallel wall clock time

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

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

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



Often neglected possibilities for scalability

- Parallelization in the time (or generally causal) dimension, particularly in nonlinear problems after spatial concurrency is exhausted
- Creating independent ensembles for asynchronous evaluation (parameter exploration or stochastic model) after space-time concurrency is exhausted on the direct problem
- Trading finely resolved discretizations (very sparse) for higher-order discretizations (block dense), or other algorithmic innovations that alter the granularity of bulk synchronous work between data movements

From generalities to a case study

- In the balance of this session, we focus in detail on the limits to performance of a prototypical unstructured mesh-based implicit computation
- With no dependence on numerical analysis other than to inform us about the essential kernels, we study the balance of computation and data motion (within a processor's own memory system and between the memory systems of different processors)
- We find that different kernels lead to different stresspoints among the architectural parameters of a hierarchical distributed memory machine
- Our study motivates the attention to architecture and the importance of extrapolating architectural parameters in the other sections of the tutorial

Case Study Model and Experiments on High-end Platforms:

Achieving High Sustained Performance in an Unstructured Mesh CFD Application

David Keyes, Columbia University

Acknowledgments for this section:

Kyle Anderson, NASA Langley Research Center & UT
William Gropp, Argonne National Laboratory
Dinesh Kaushik, Argonne National Laboratory
Barry Smith, Argonne National Laboratory

Motivation

- 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”

Plan for balance of Session I

- Background of 1999 Bell Prize winner in “Special” category
  - application
  - algorithm
- General characterization of PDE requirements
  - identification of common algorithmic building blocks
  - simple complexity analyses (computation, communication, inter-processor motion)
- Identification and illustration of bottlenecks on some of yesterday's important platforms
  - ASCI Red (Intel Pentium), ASCI Blue Mountain (SGI MIPS), ASCI Blue Pacific (IBM Power), Cray T3E (DEC Alpha)
- … and some of today’s
  - IBM BlueGene/L, NSF Teragrid, VaTech System X
- Speculation on useful algorithmic research directions

Euler simulation

- 3D transonic flow over ONERA M6 wing, at 3.06º angle of attack (exhibits λ-shock at M = 0.839)

- Solve

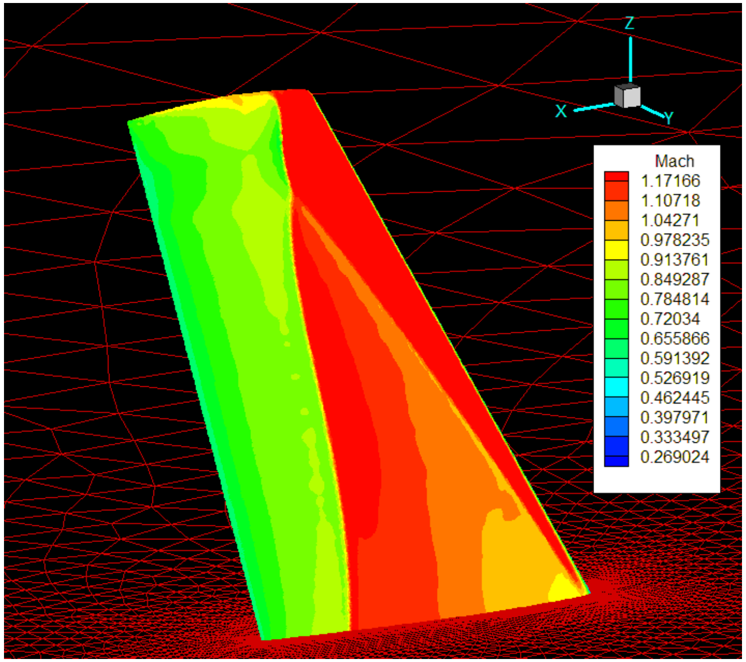
$$\frac{\partial Q}{\partial t} + \frac{1}{V}\oint_{\Omega}(\vec{F}\cdot\hat{n})d\Omega = 0$$

where

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

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

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



ρ = density
$\boldsymbol{U}$ = velocity
p = pressure
E = energy density

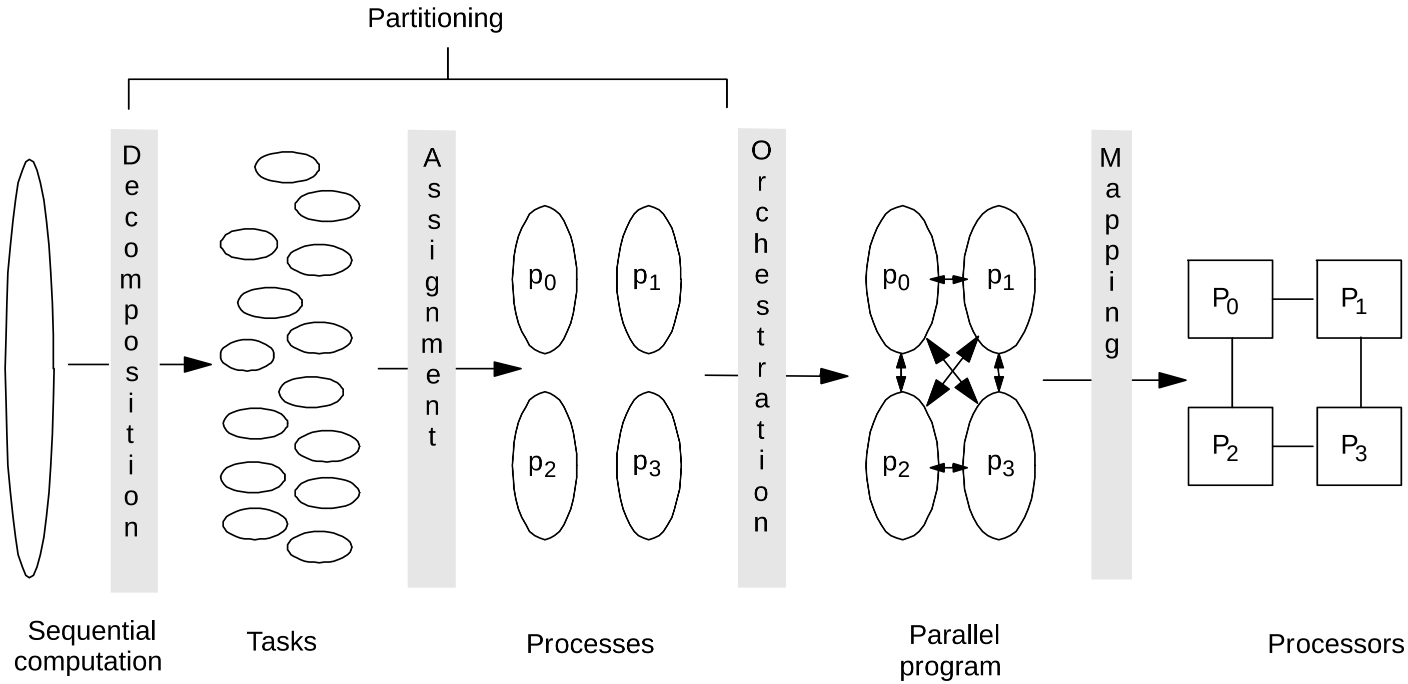
Background of FUN3D application

- 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 ASCI 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

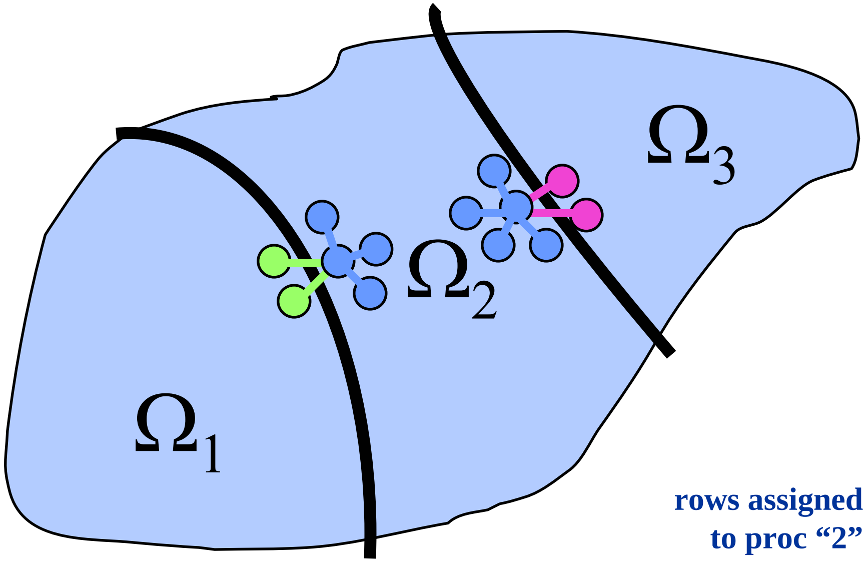
Four steps in creating a parallel program



- Decomposition of computation in tasks
- Assignment of tasks to processes
- Orchestration of data access, communication, synchronization
- Mapping processes to processors

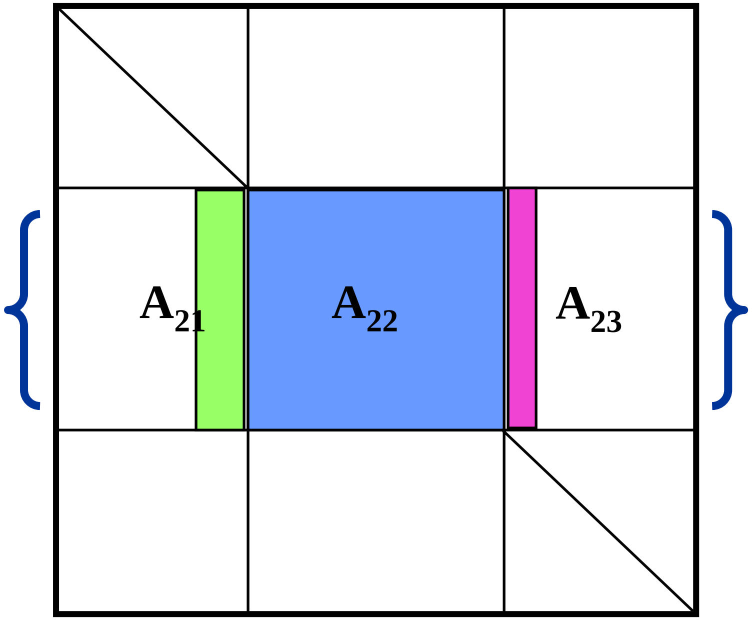
c/o Culler et al, UC Berkeley

SPMD parallelism w/domain decomposition



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

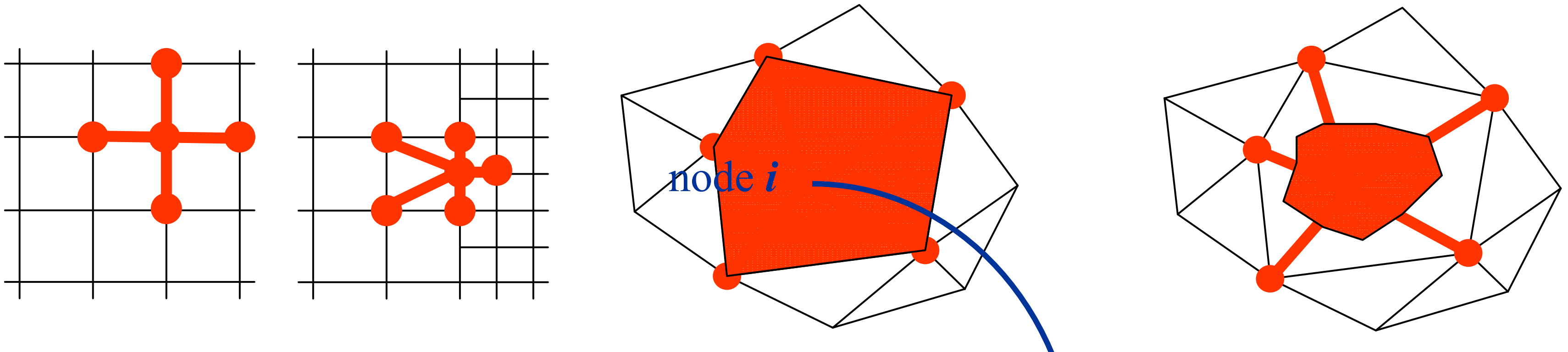
rows assigned to proc “2”

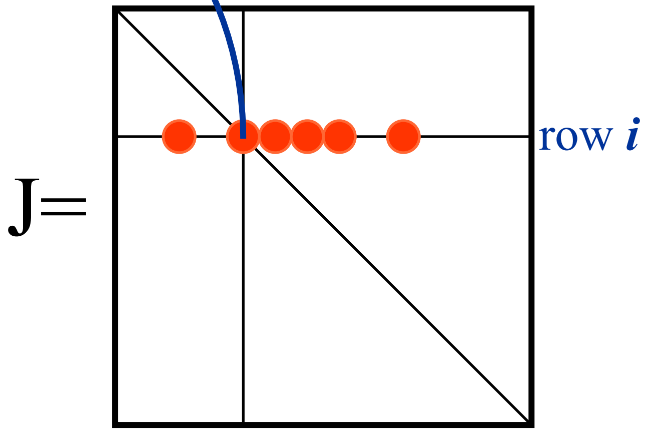


Partitioning of the grid induces block structure on the system matrix (Jacobian)

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

Algorithm: Newton-Krylov-Schwarz

Newton

nonlinear solver

asymptotically quadratic

Krylov

accelerator

spectrally adaptive

Schwarz

preconditioner

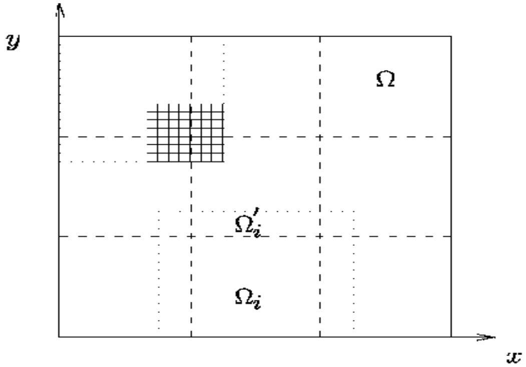
parallelizable

Merits of NKS algorithm/implementation

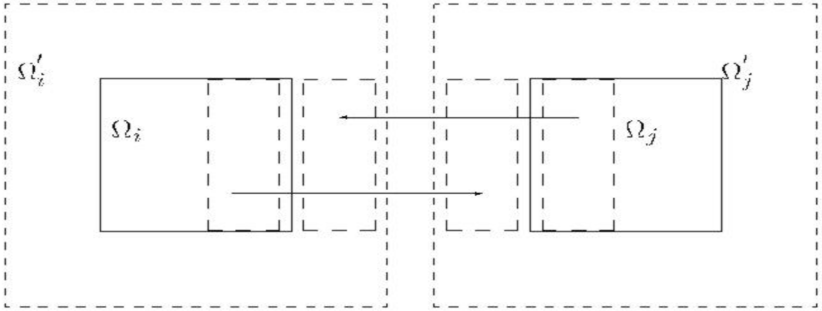
- Relative characteristics: the scaling “exponents” are *naturally* good
  - Convergence scalability
    - weak (or no) degradation in problem size and parallel granularity (with use of small global problems in Schwarz preconditioner)
  - Implementation scalability
    - no degradation in ratio of surface 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 Ω

- 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



$$A_i = R_i A R_i^T$$

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

$$B = \sum_i B_i$$

Iteration count estimates from 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

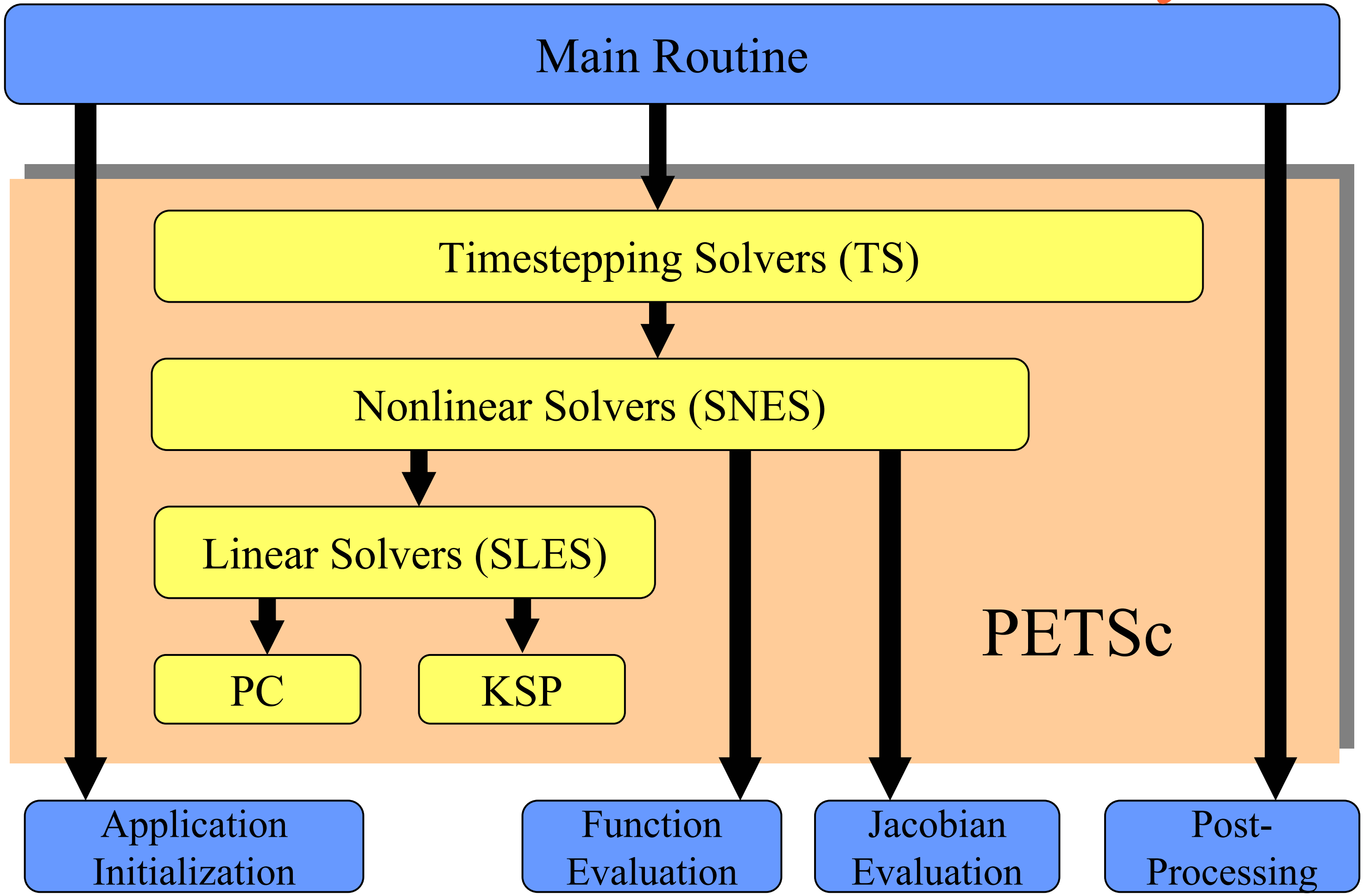
- Subdomain partitioning by one of the MeTiS graph algorithms
- SPMD "owner computes" PETSc implementation under the dual objectives of minimizing the number of messages and overlapping communication with computation
- Each processor "ghosts" its stencil dependences in its neighbors
- Ghost nodes ordered 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, http://www.tops-scidac.org

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, Nx
  - temporal grid points, Nt
  - components per point, Nc
  - auxiliary storage per point, Na
  - grid points in stencil, Ns
- Memory: $M \approx Nx \bullet (Nc + Na + Nc \bullet Nc \bullet Ns)$
- Work: $W \approx Nx \bullet Nt \bullet (Nc + Na + Nc \bullet Nc \bullet Ns)$

Explicit solvers

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

- 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

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

- 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.

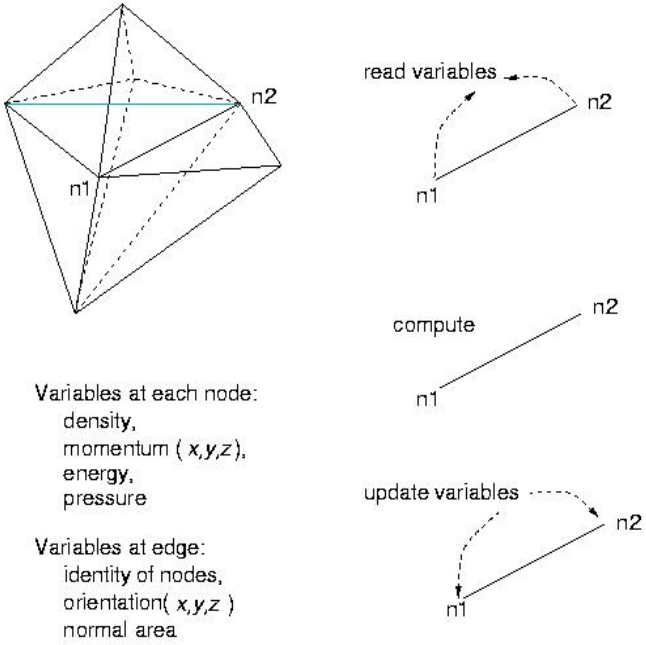
Primary PDE solution kernels

(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

Previews of observations for PDE 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
- Coarse grid in preconditioner can be 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

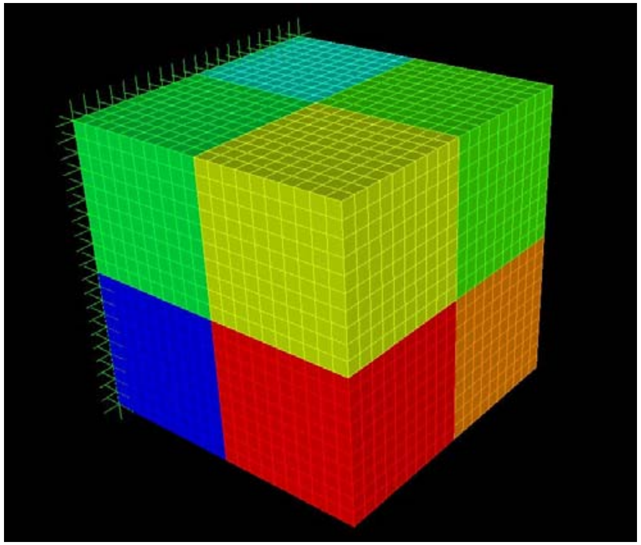
Observation #1:
Processor scalability no problem, in principle

- 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

Estimating scalability for bulk-synchronized PDE stencil computations

- Given complexity estimates of the leading terms of:
  - the concurrent computation (per iteration phase)
  - the concurrent communication
  - the synchronization frequency
- And a 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 $\theta \equiv \frac{32\,B^3}{243\,A^2 C}$),

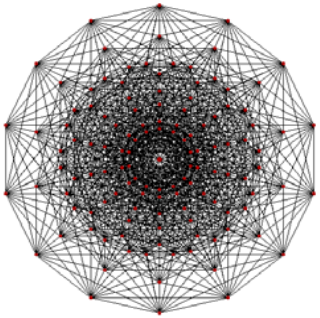
$$p_{opt} = \left(\frac{3A}{2C}\right)^{1/3}\left(\left[1+(1-\sqrt{\theta})\right]^{1/3} + \left[1-(1-\sqrt{\theta})\right]^{1/3}\right)\cdot n$$

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

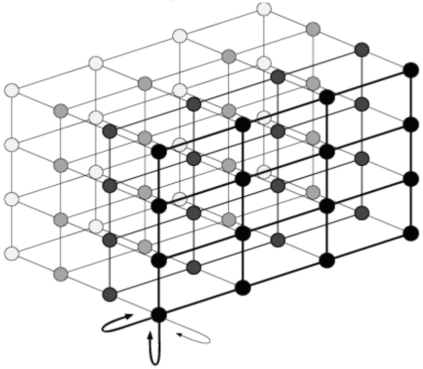
$$p_{opt} = \left(\frac{3A}{C}\right)^{1/3}\cdot n$$

Scalability results for domain-decomposed bulk-synchronized PDE stencil computations

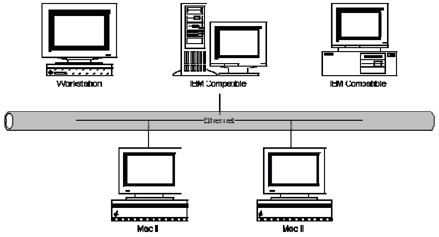
- With tree-based (logarithmic) global reductions and scalable nearest neighbor hardware:
  - optimal number of processors scales *linearly* with problem size



- 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")

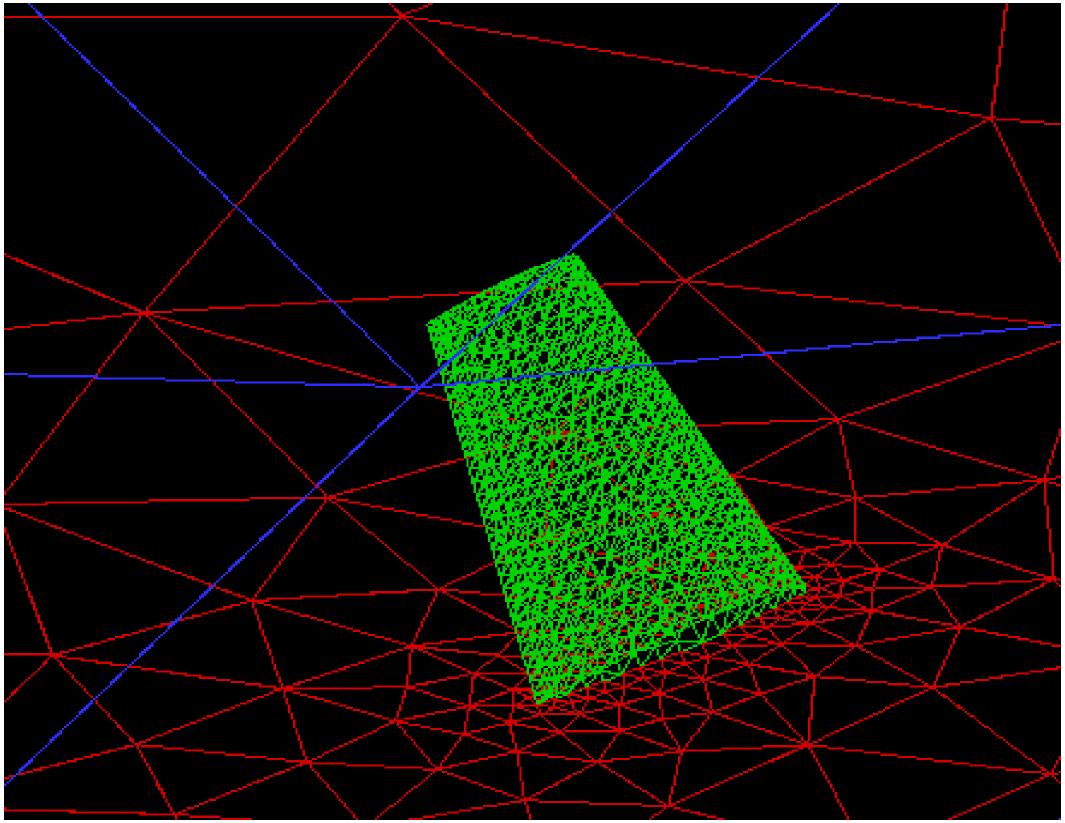


- With common network bus (heavy contention):
  - optimal number of processors scales as *one-fourth* power of problem size (not "scalable")

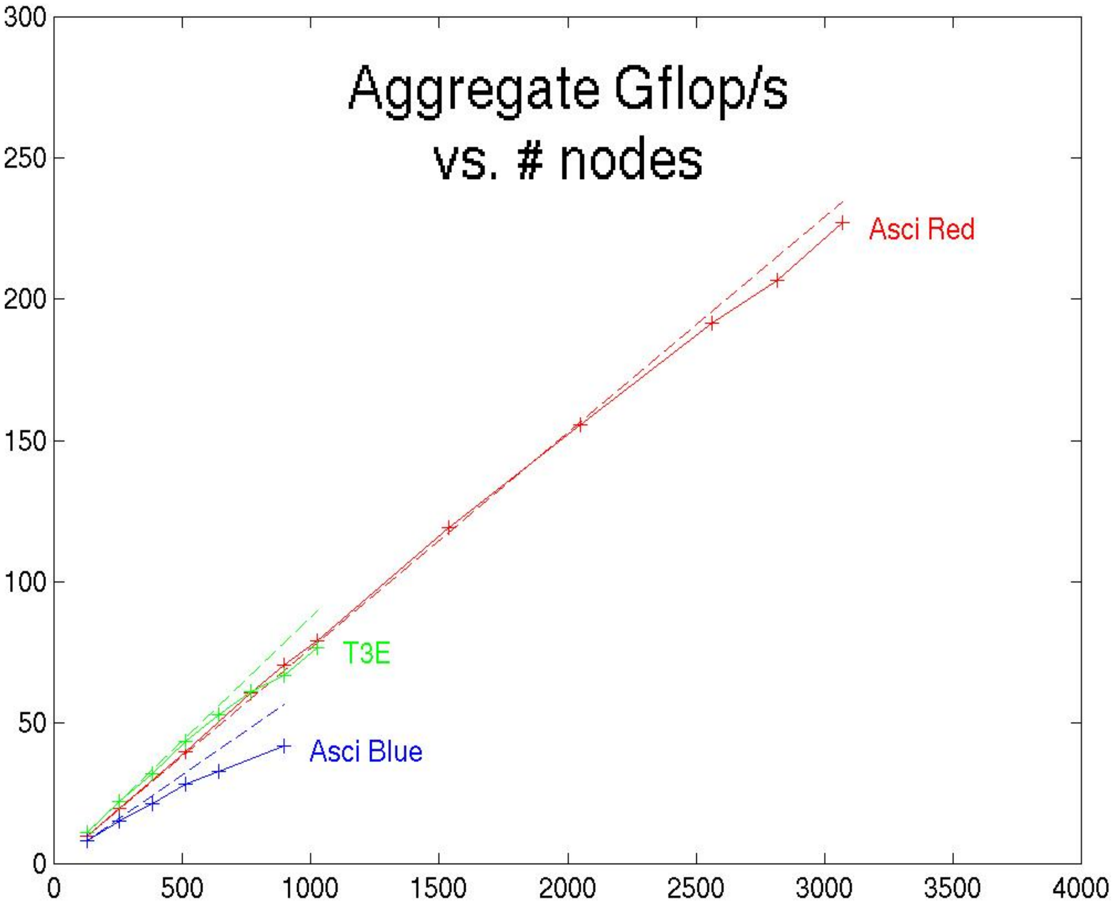


Surface visualization of test domain for Euler flow over an ONERA M6 wing

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



Fixed-size parallel scaling results (Flop/s)



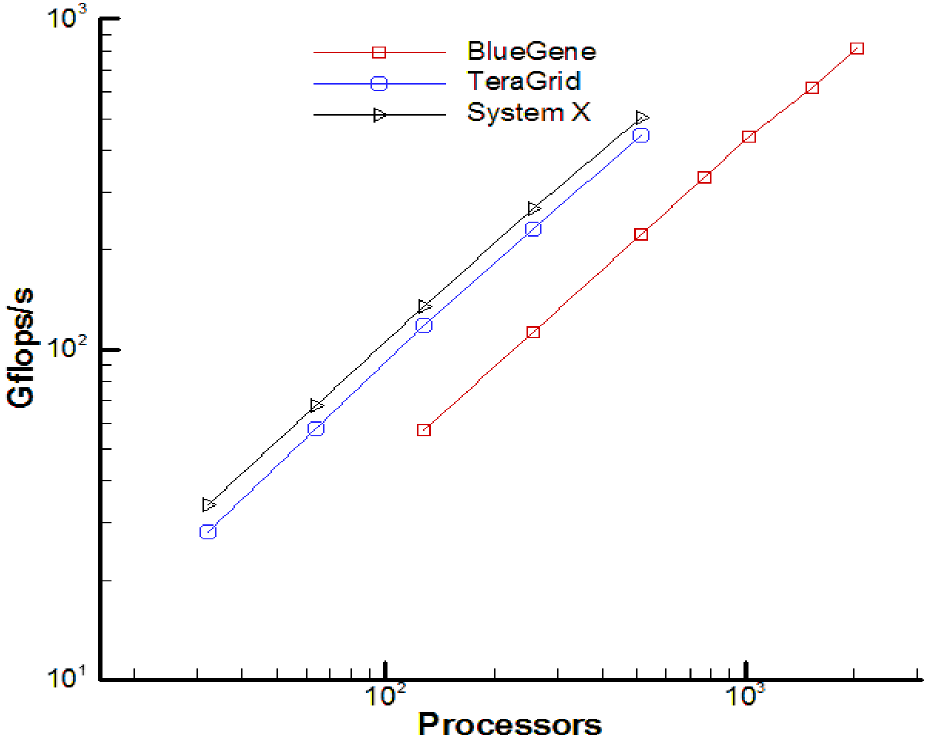
Parallel performance of PETSc-FUN3D

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

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

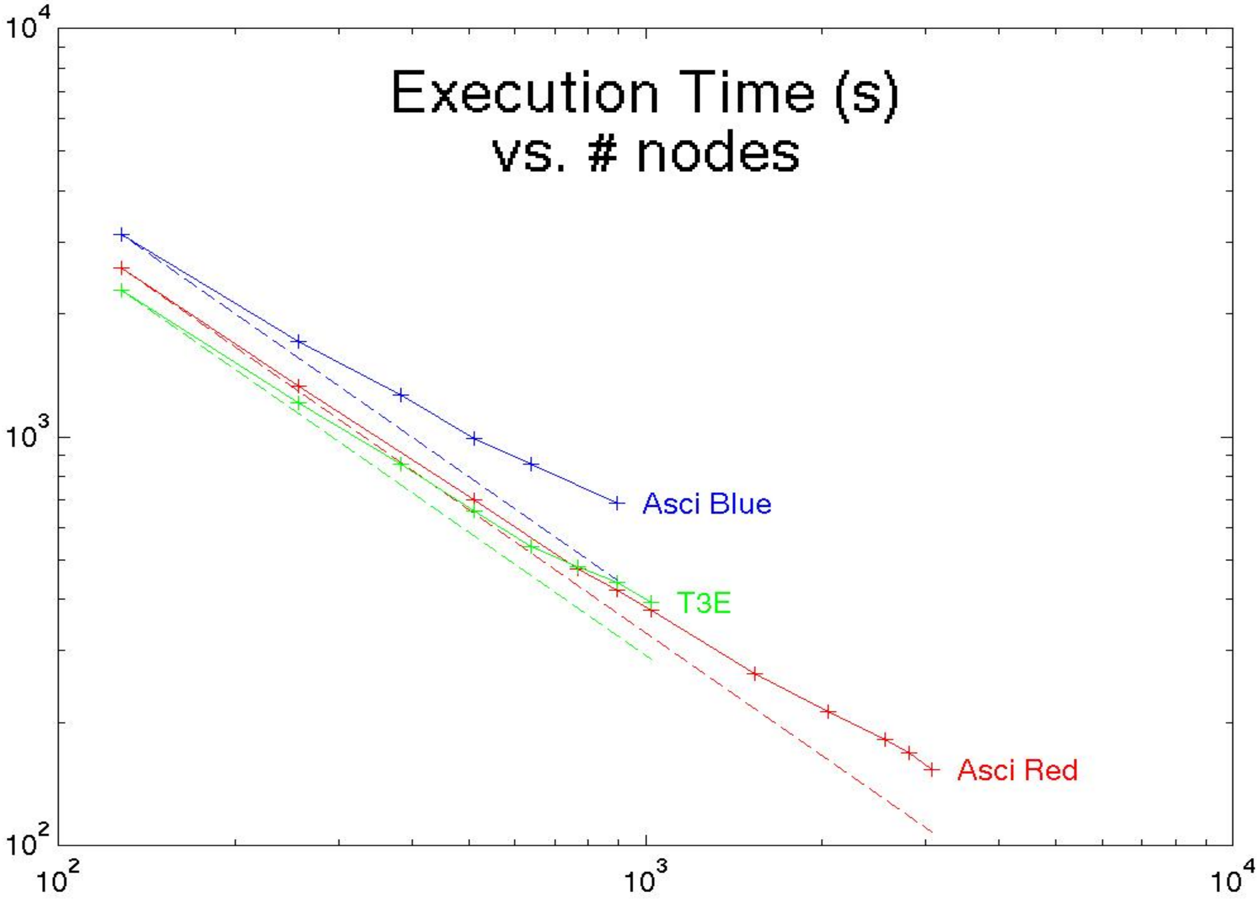
BlueGene: Dual 700 MHz IBM Processors with 4 MB L3 Cache

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



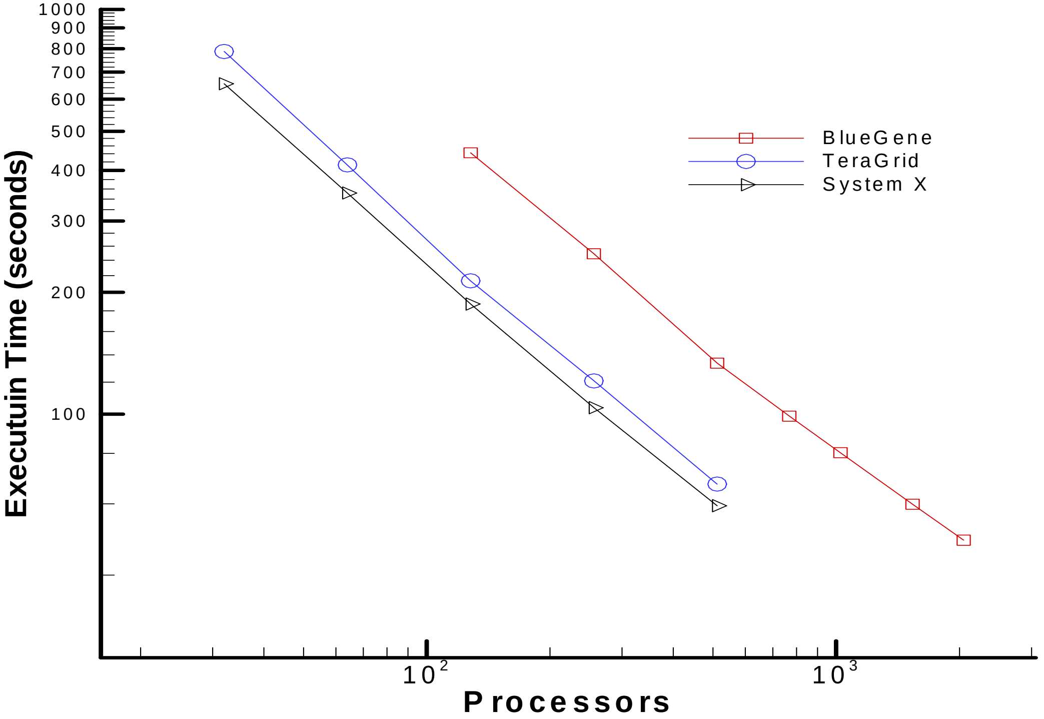
Fixed-size parallel scaling results

(time in seconds)



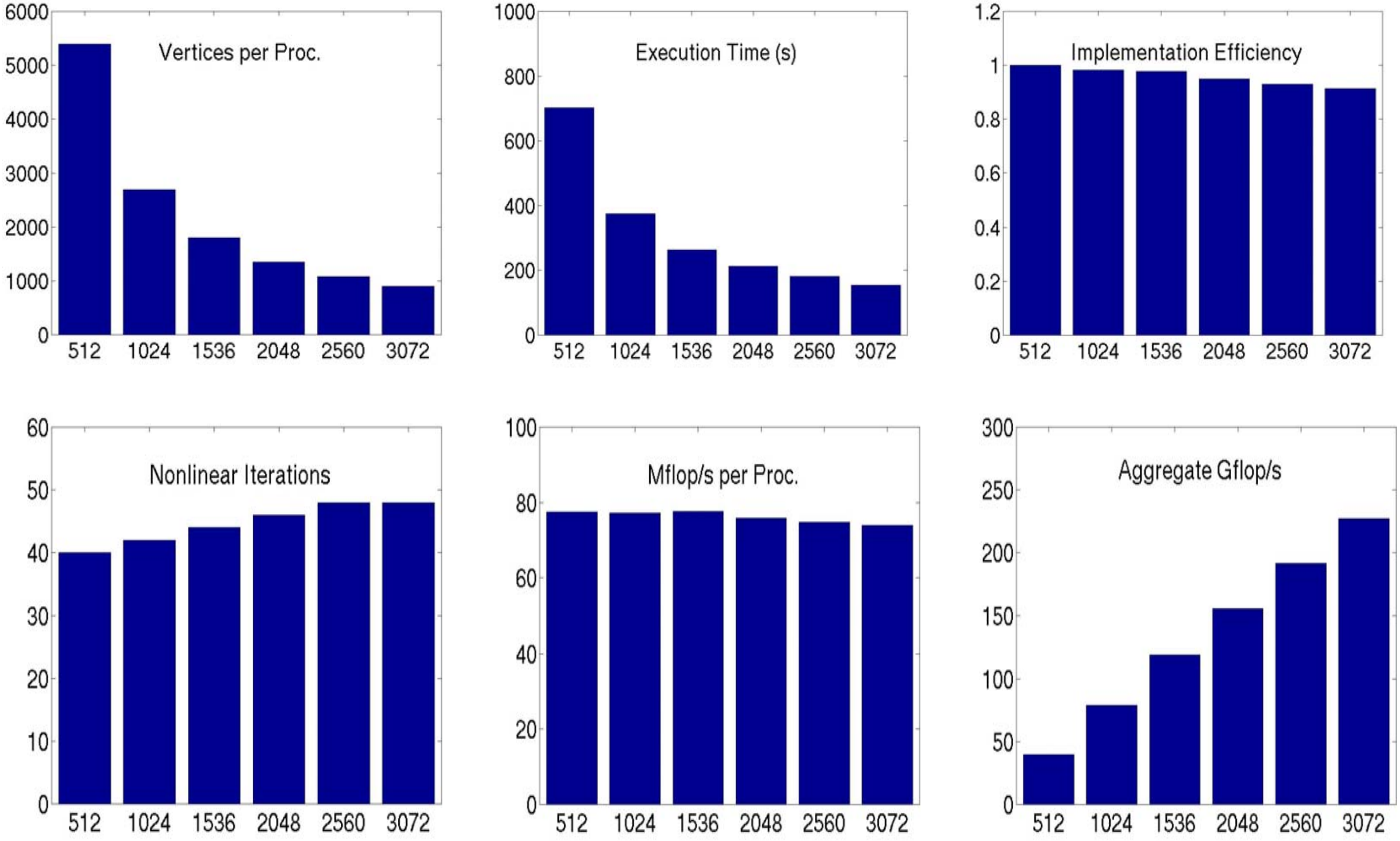
Parallel performance of PETSc-FUN3D

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



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 #2a:

Coarse grid can be a bottleneck

- Execution times for scaled 3D elliptic problem with various coarse grid components of preconditioner, over 64-fold range of size and processor number (NK= Newton-Krylov; NR = Newton-Richardson)
- Largest case has 2 million unknowns and 8x8x8 replicated coarse grid

Number of Processors	***1/h***	***1/H***	**2-level NK**	**V-cycle NK**	**F-cycle NK**	**F-cycle NR**
1	**32**	**2**	**21.5s**	**19.6s**	**19.6s**	**21.1s**
8	**64**	**4**	**26.0s**	**23.3s**	**24.3s**	**26.1s**
64	**128**	**8**	**36.5s**	**31.2s**	**30.8s**	**34.4s**
Scaled Efficiency			***0.59***	***0.63***	***0.64***	***0.61***

Observation #2a, continued:

Coarse grid can be a bottleneck

- Algorithmic scalability (linear iteration count per Newton step) for scaled 3D elliptic problem with various coarse grid components of preconditioner, over 64-fold range of size and processor number
- Largest case has 2 million unknowns and 8x8x8 replicated coarse grid

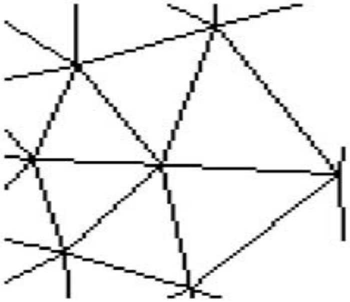
Number of Processors	***1/h***	***1/H***	**2-level NK**	**V-cycle NK**	**F-cycle NK**	**F-cycle NR**
1	**32**	**2**	**4.3**	**3.0**	**2.3**	**3.8**
8	**64**	**4**	**4.7**	**3.0**	**2.5**	**4.0**
64	**128**	**8**	**5.6**	**3.3**	**2.3**	**4.0**
Scaled Efficiency			***0.77***	***0.91***	***1.00***	***0.95***

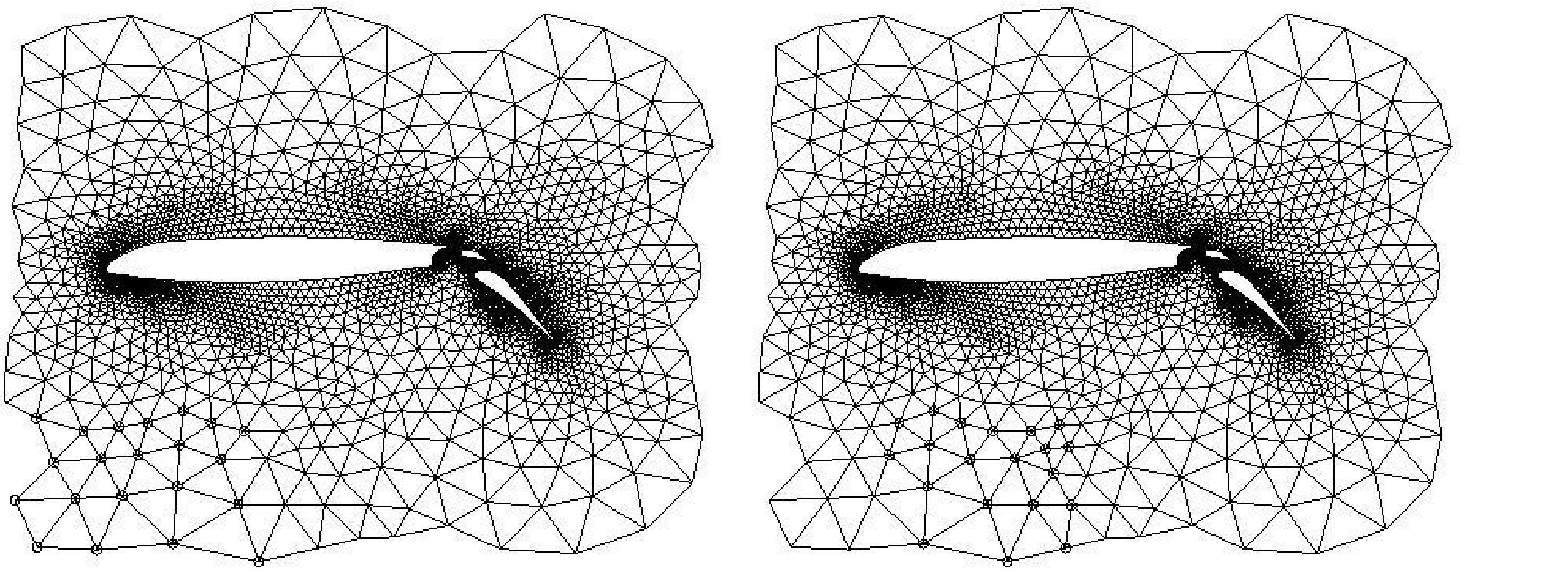
Observation #3:
Memory latency no problem, in principle

- 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

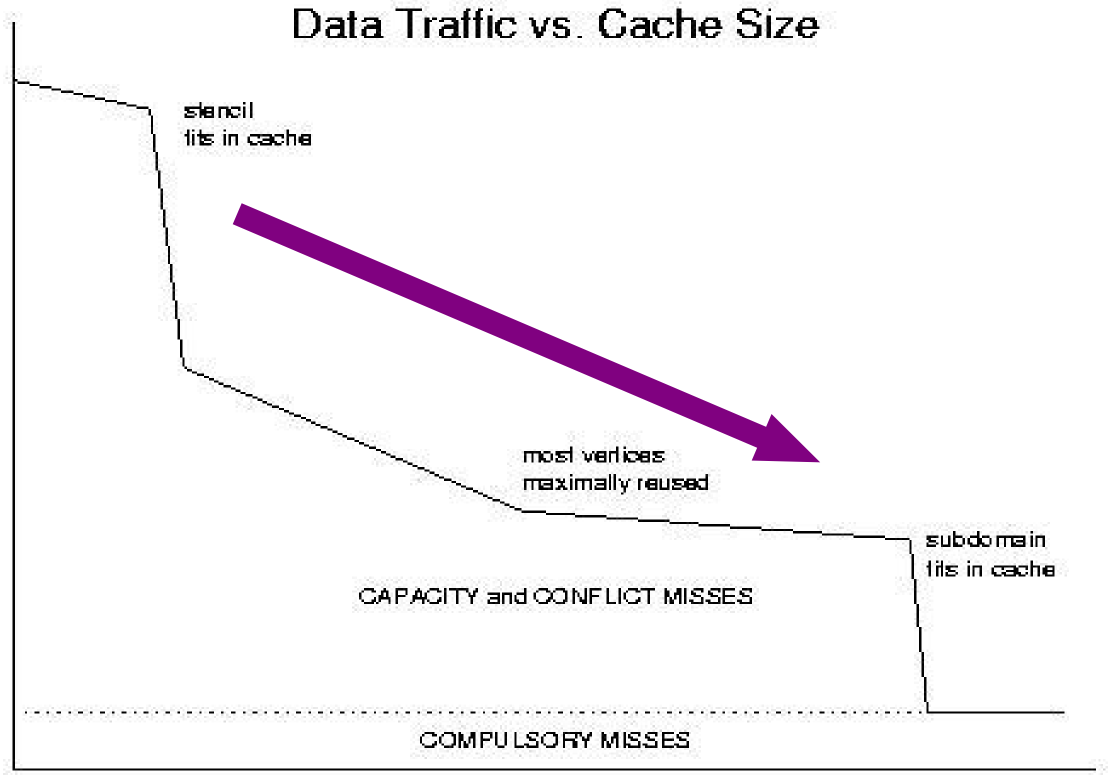
- Smallest: data for single stencil
- Largest: data for entire subdomain
- Intermediate: data for a neighborhood collection of stencils, reused as many times as possible





Gedanken 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, …, Un, Vn, Wn*
  - rather than *U1, U2, …, Un, V1, V2, …, Vn, W1, W2, …, Wn*
- *Sparse Jacobian blocking* for minimal integer metadata in manipulating a given amount of floating point physical data

Improvements 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

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

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
```

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), $x_1$(ja(j)), .....$x_N$(ja(j))
        do N fmadd (floating multiply add)
   enddo
    Store $y_1$(i) .....$y_N$(i)
 enddo
```

This version performs A • $\{x_1, \ldots, x_N\}$

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, and 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

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

Format	Number of Vectors	Bytes / fmadd	Bandwidth		MFlops	
			Required	Measured	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

- On 2.4 GHz P4 Xeon
- Stream performance 1973 MB/sec (triad vector operation) http://www.cs.virginia.edu/stream

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, $sum_1$ .....$sum_N$                          //  N Ld
        do j=1,ncol                                               // 1 Ld
          fetch ja(jrow), a(jrow), $x_1$(ja(jrow)), .....$x_N$(ja(jrow))
                                                                 // 1 Of, N+2 AT N+2 Ld
          do N fmadd (floating multiply add)                     // 2N Flop
        enddo                                                     // 1 Iop, 1 Br
        Store $sum_1$.....$sum_N$ in $y_1$(i) .....$y_N$(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" performance

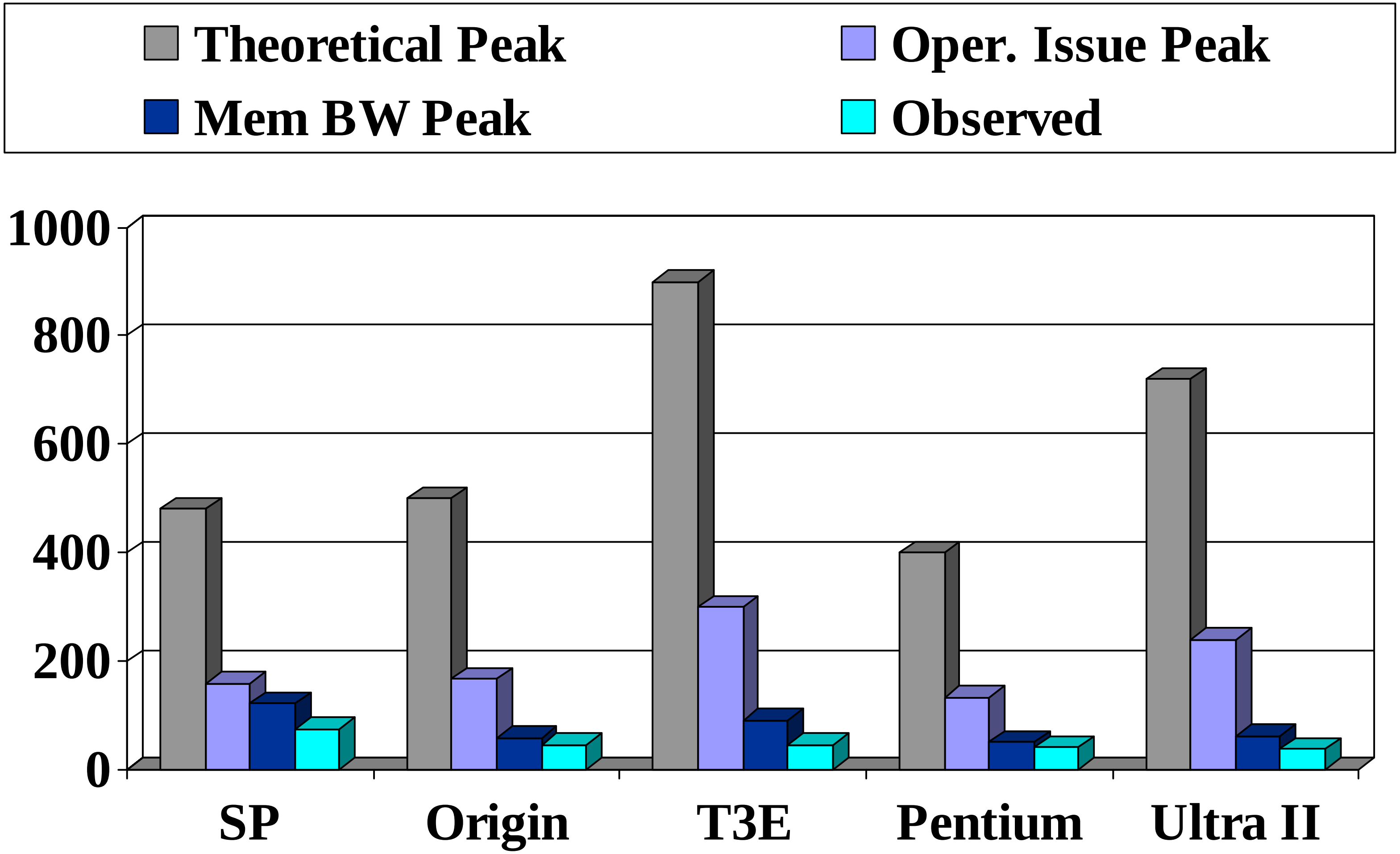
Significance of multivectors

- 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

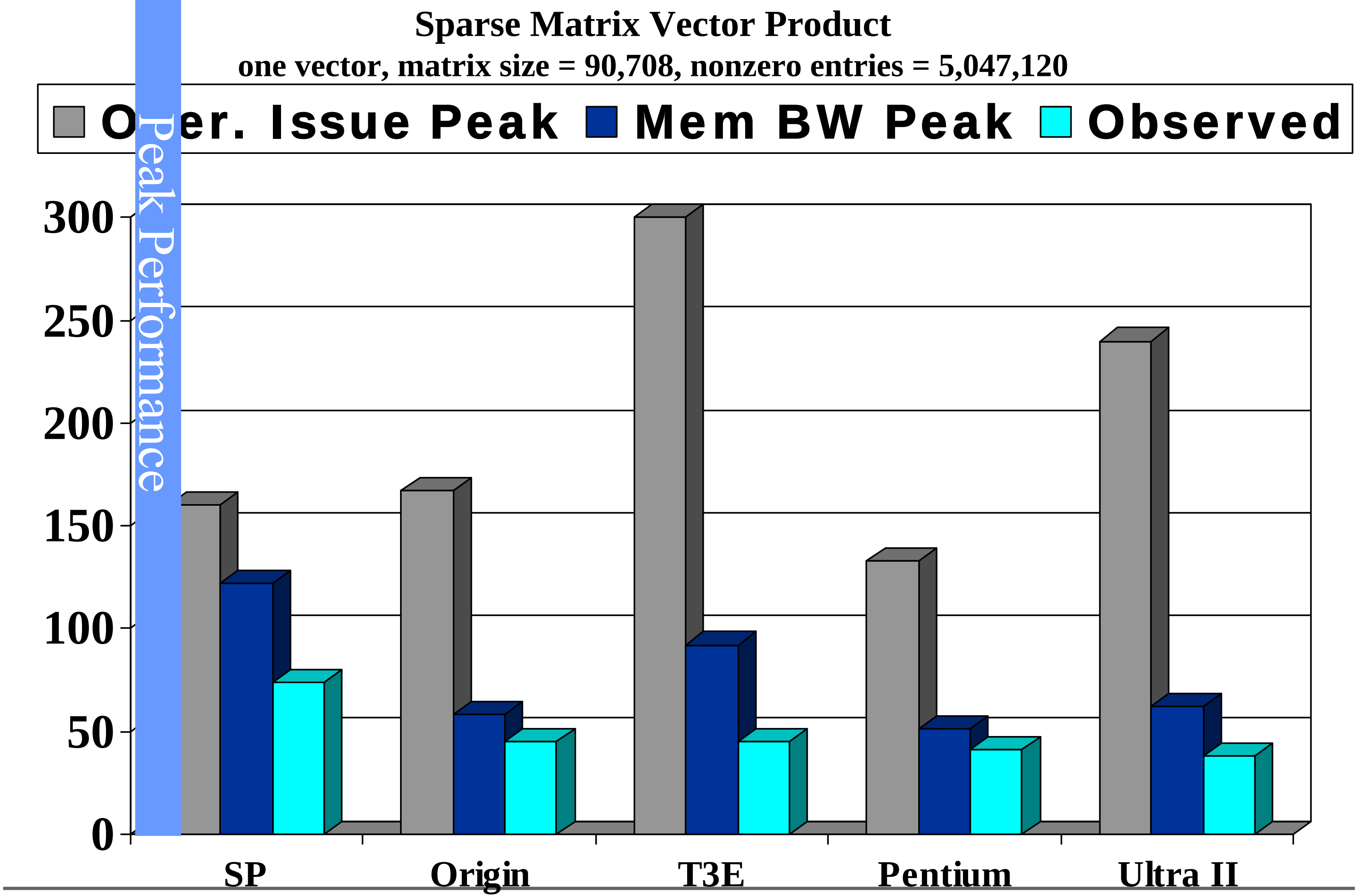
Sparse Matrix Vector Product

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



Realistic measures of performance

Peak Performance



Summary of observations for PDE 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
- Coarse grid in preconditioner can be 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

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

Performance improvement from algorithms (1)

- 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)

Performance improvement from algorithms (2)

- 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

Examples 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

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

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

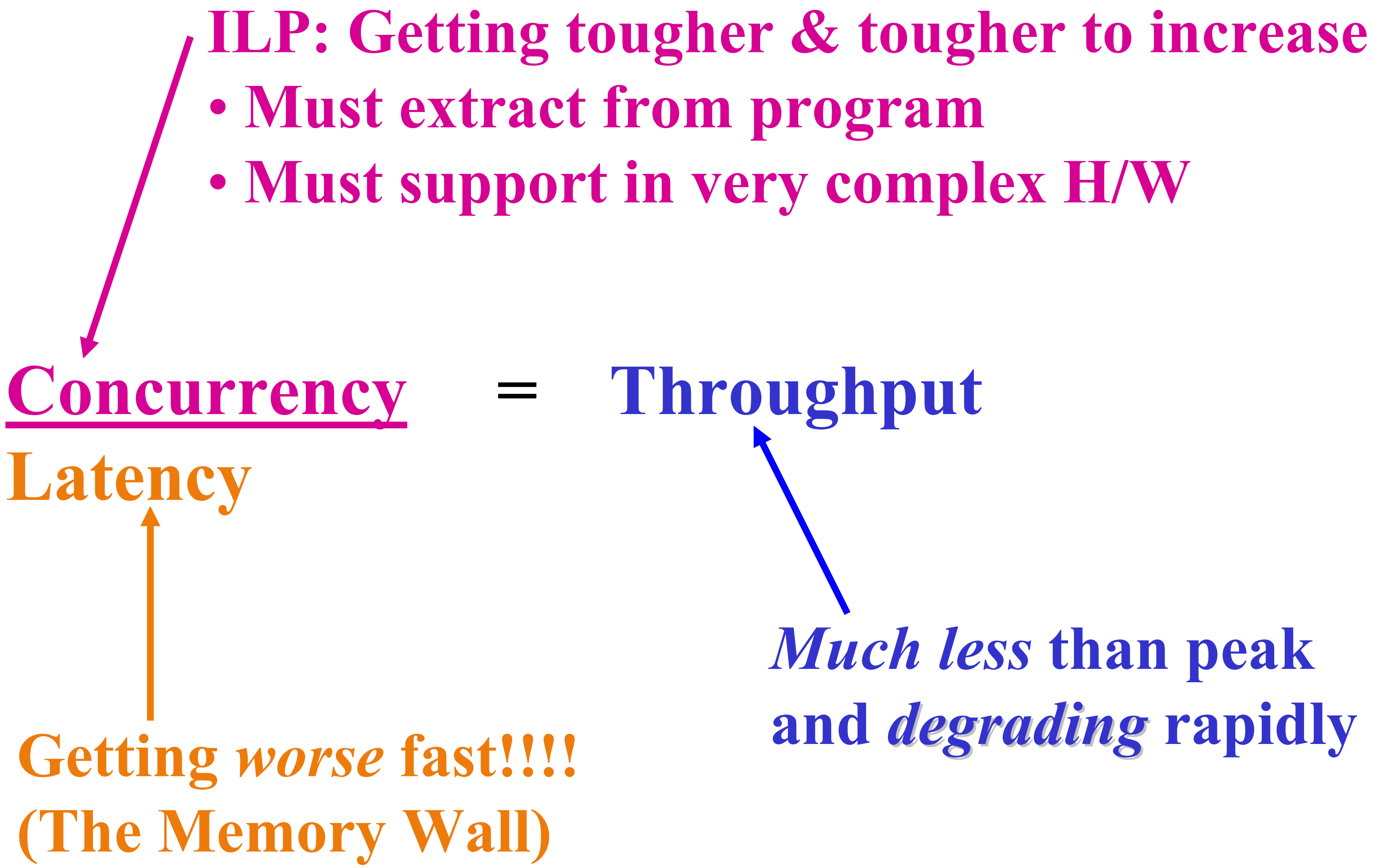
IBM Fellow (ret)

UNIVERSITY OF NOTRE DAME

Why Is Supercomputing Really Hard?

- Silicon density: Sheer space taken up implies large distances & ***looooooong*** latencies
- Silicon mindset:
  - Processing logic "over here"
  - Memory "over there"
  - And we add acres of high heat producing stuff to bridge the gap
- Questions:
  - Where are we going with "business as usual"
  - How far can we scale with a *mindset* (but not technology) change?
- And is it enough? (to be answered later)

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**

$$\frac{\text{Concurrency}}{\text{Latency}} = \text{Throughput}$$

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

***Much less* than peak**
and *degrading* rapidly

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

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 by other than clock rate
  - Line between "Logic" and "Memory" chips blurs
  - We will increase "*threads per die*" <u>not</u> IPC/core

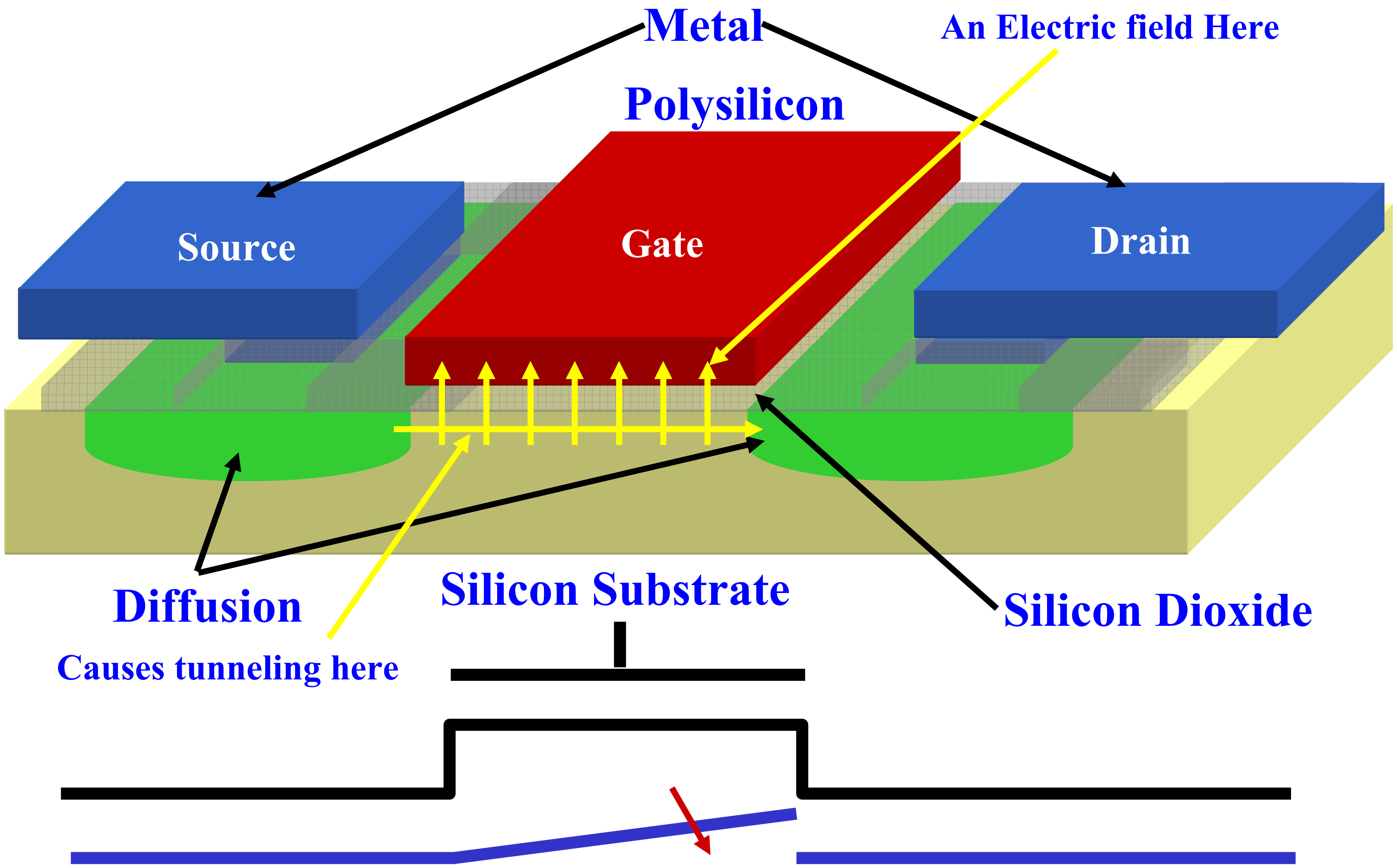
Outline

- 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



Metal

An Electric field Here

Polysilicon

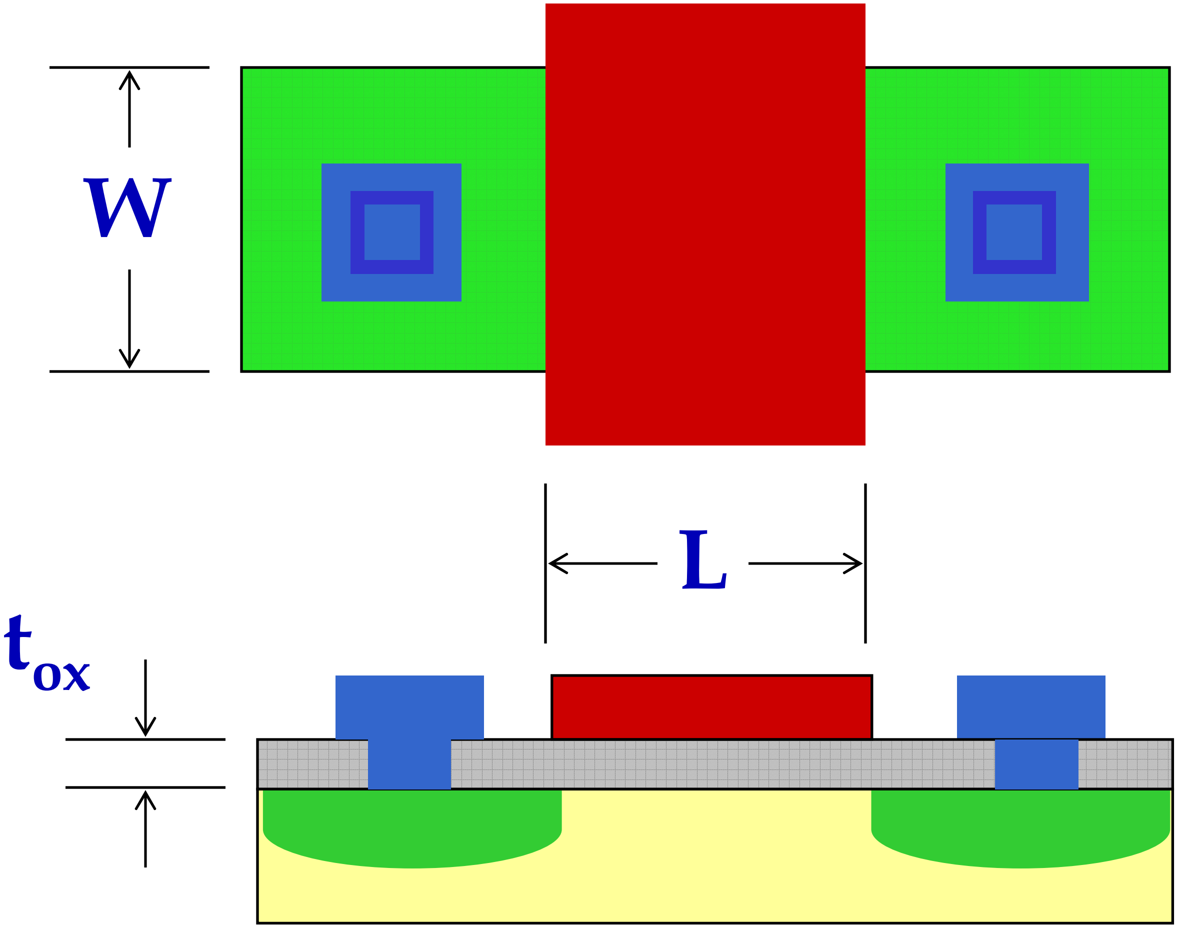
Diffusion

Causes tunneling here

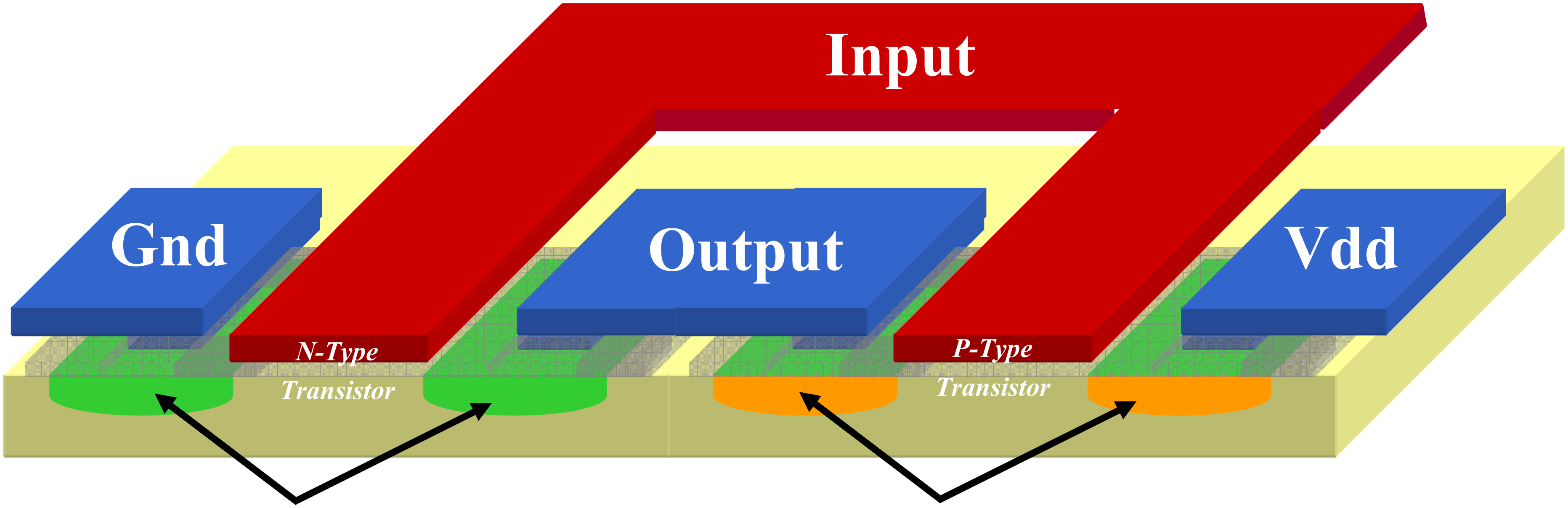
Silicon Substrate

Silicon Dioxide

Key Device Parameters



A Logic Inverter

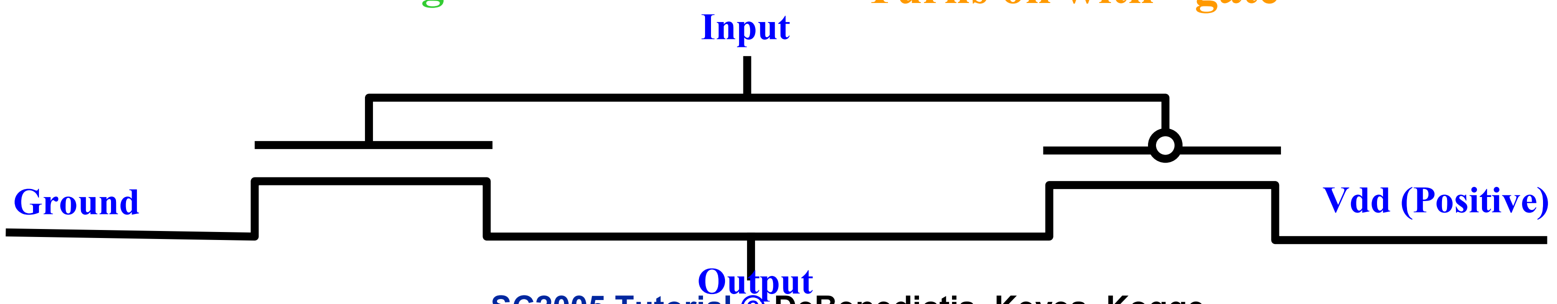


N-Type Diffusion/Transistor

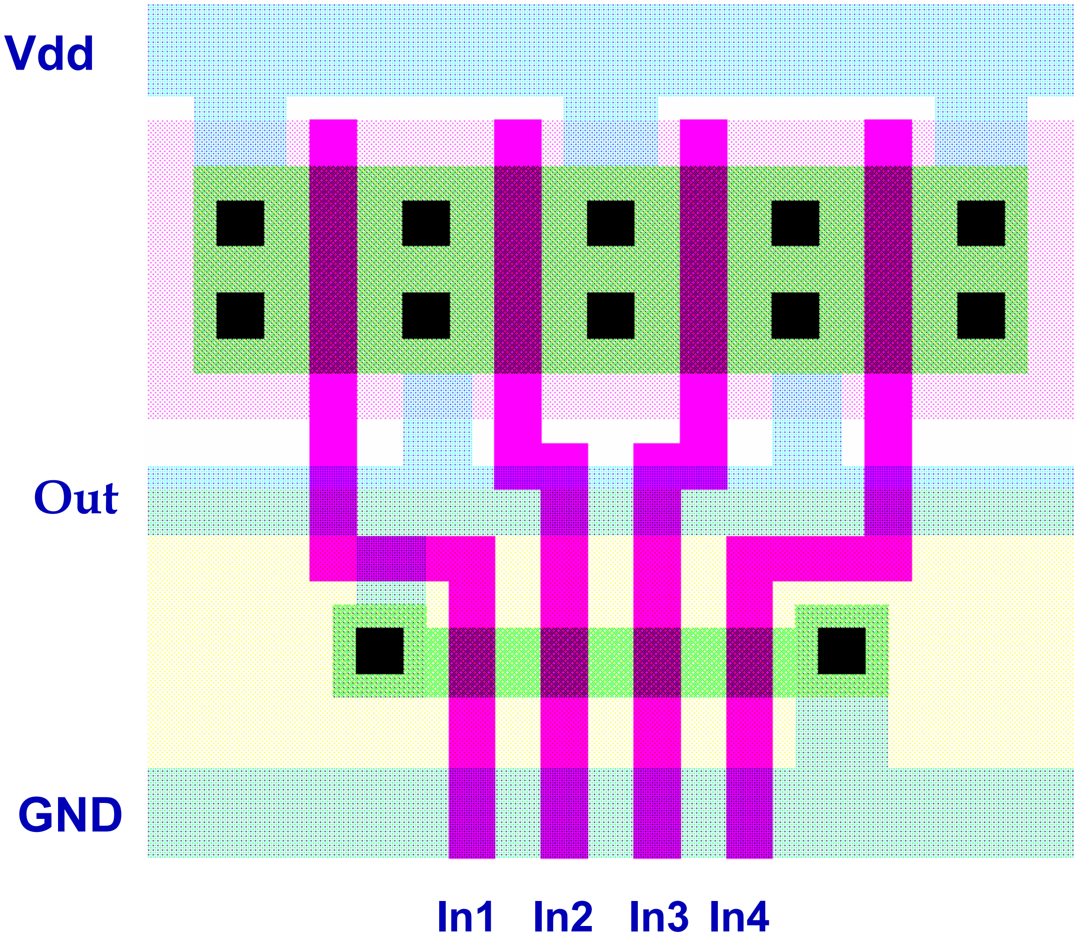
- electron rich
- Turns on with + gate

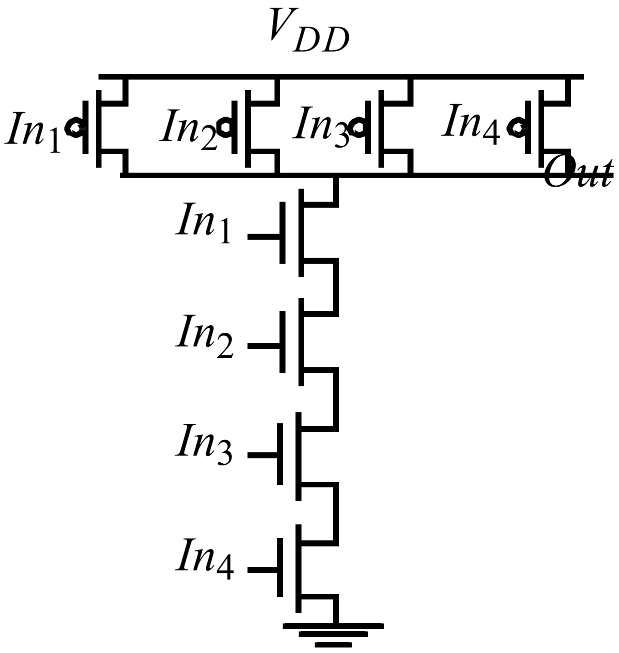
P-Type Diffusion/Transistor

- electron poor
- Turns on with - gate

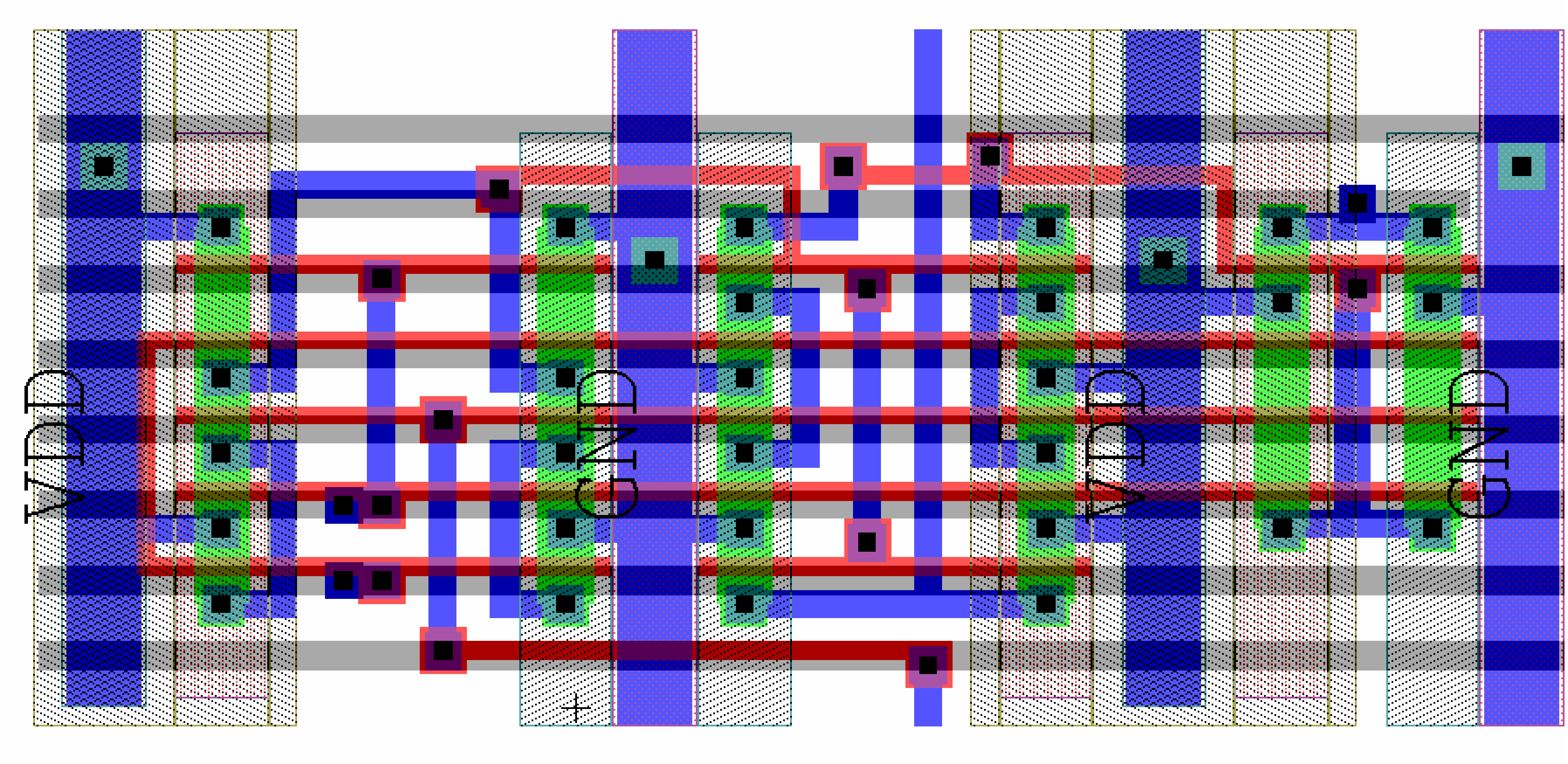


4-input NAND Gate



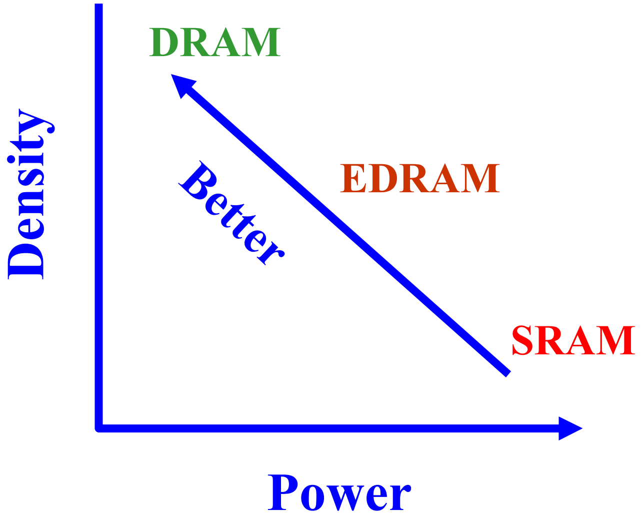


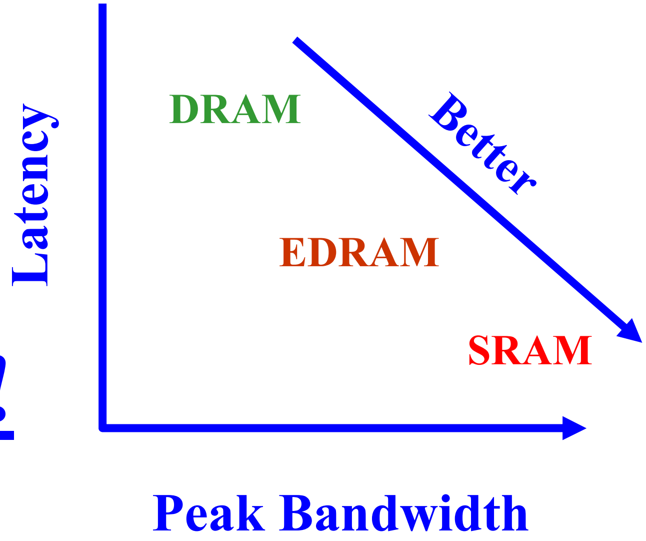
Full Adder



Key Types of Memory Cells

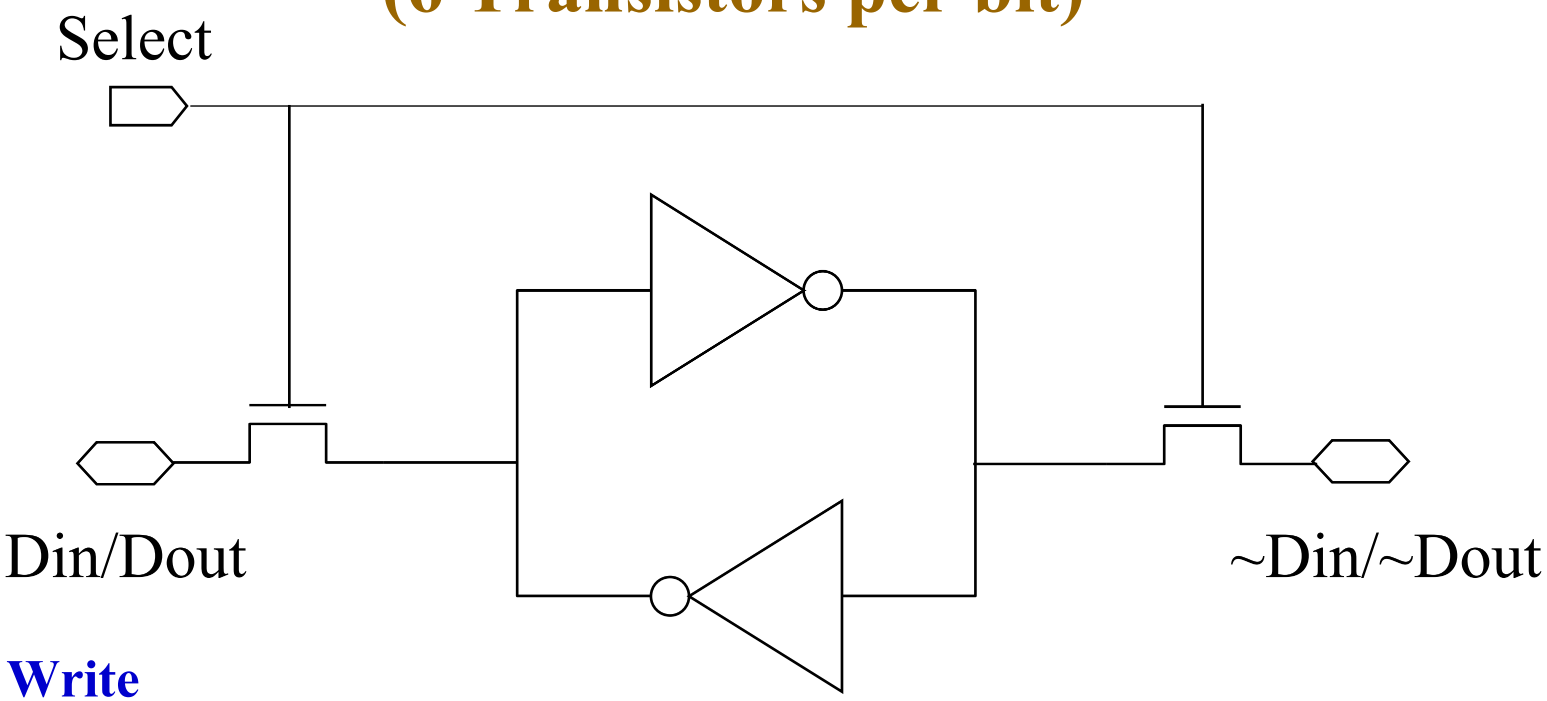
- Commodity DRAM
- Embedded DRAM
- SRAM
- Non-Volatile RAM
  - – NAND Type
  - – NOR Type





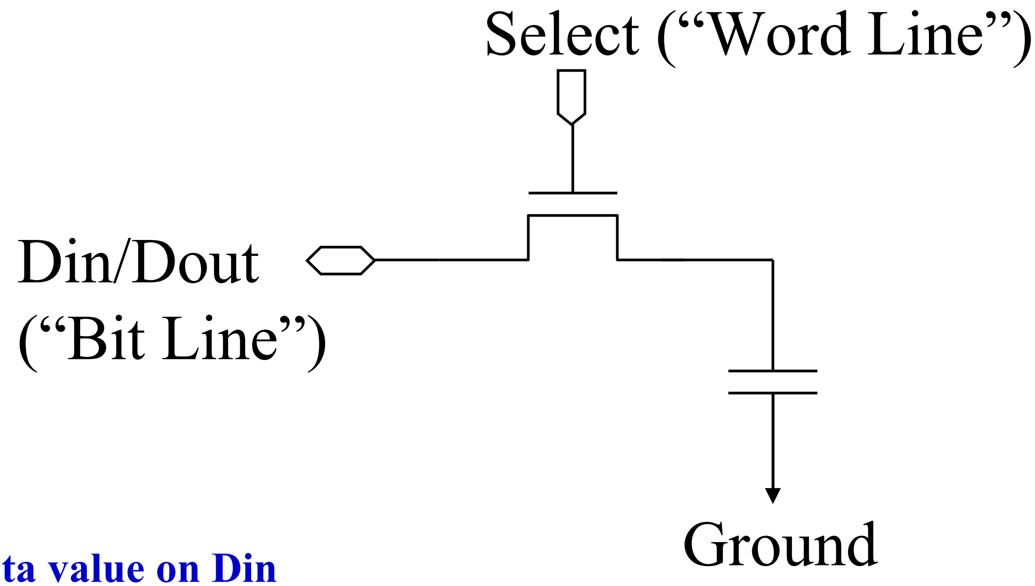
No single optimal choice!

Static RAM bit
(6 Transistors per bit)



- To Write
  - Place data and ~data on Din & ~Din
  - Raise Select
- To Read
  - Raise Select to couple latch to outputs
  - Sense output lines Dout & ~Dout
- In between, data stays latched in inverters

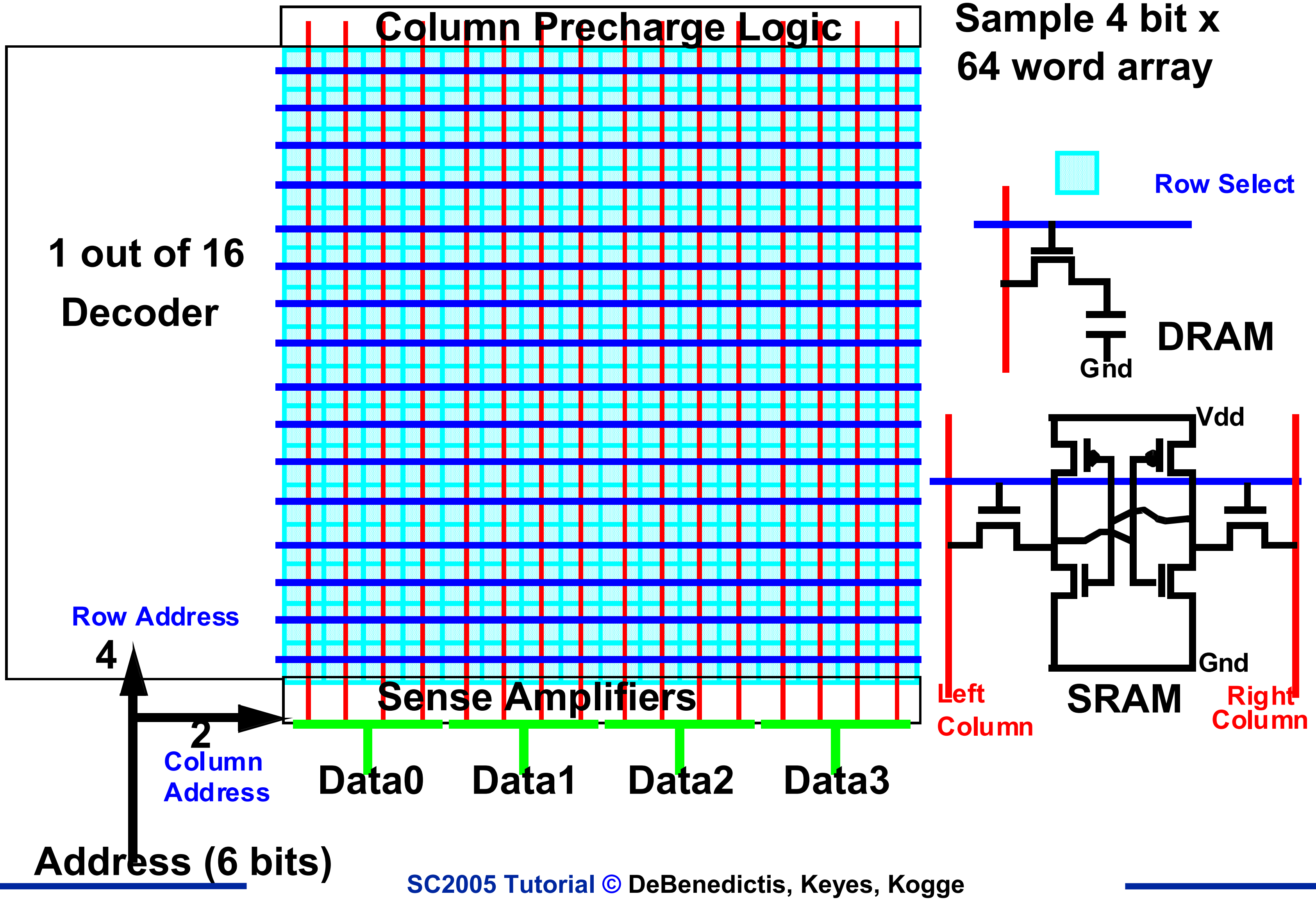
Charge-Based DRAM Bit
(1 Transistor)



- To Write
  - Place data value on Din
  - Activiate Select
  - Capacitor is charged/discharged
- To Read
  - Activate Select
  - Read value on capacitor from Dout
- But charge "leaks" away over time

Memory Arrays

Sample 4 bit x 64 word array



Column Precharge Logic

1 out of 16

Decoder

Row Address

4

2

Column

Address

Sense Amplifiers

Data0

Data1

Data2

Data3

Address (6 bits)

Row Select

DRAM

Gnd

Vdd

Gnd

Left

Column

SRAM

Right

Column

Compact DRAM Cells for Memory Arrays

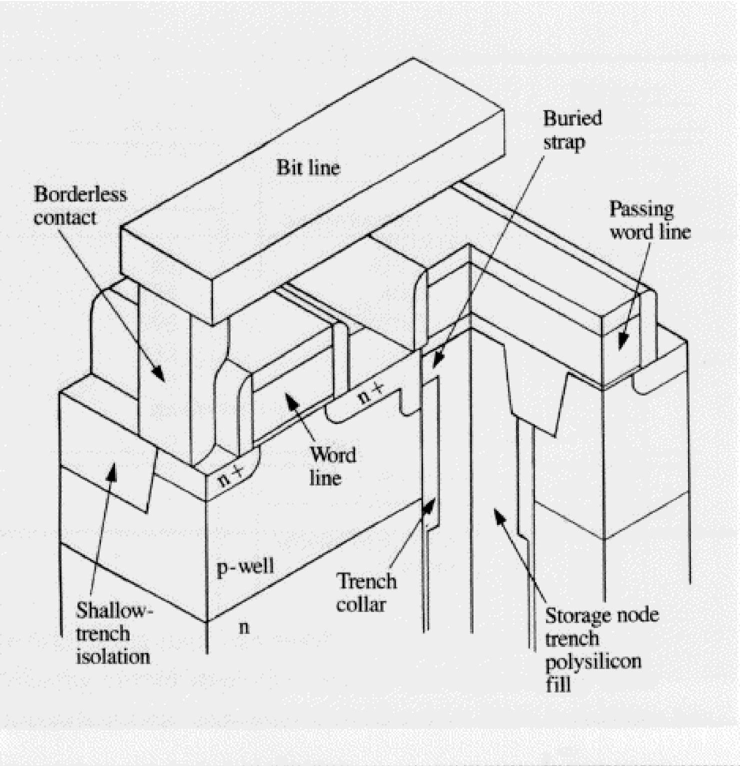
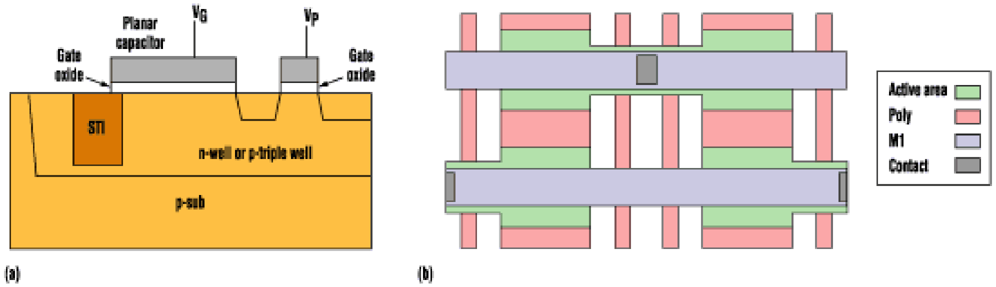
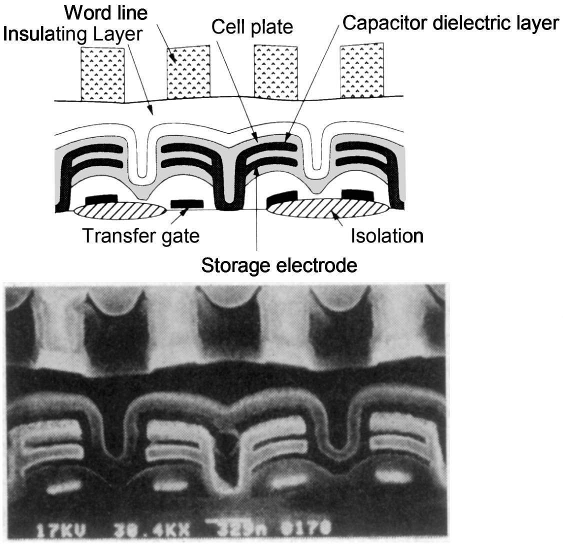


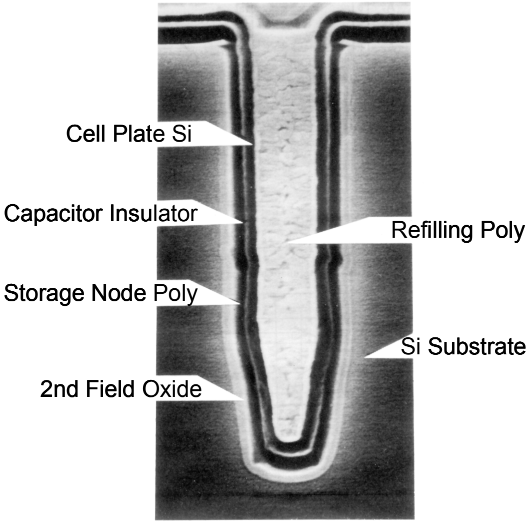
Figure 8

Cross section of 256Mb buried strap trench (BEST) DRAM cell.



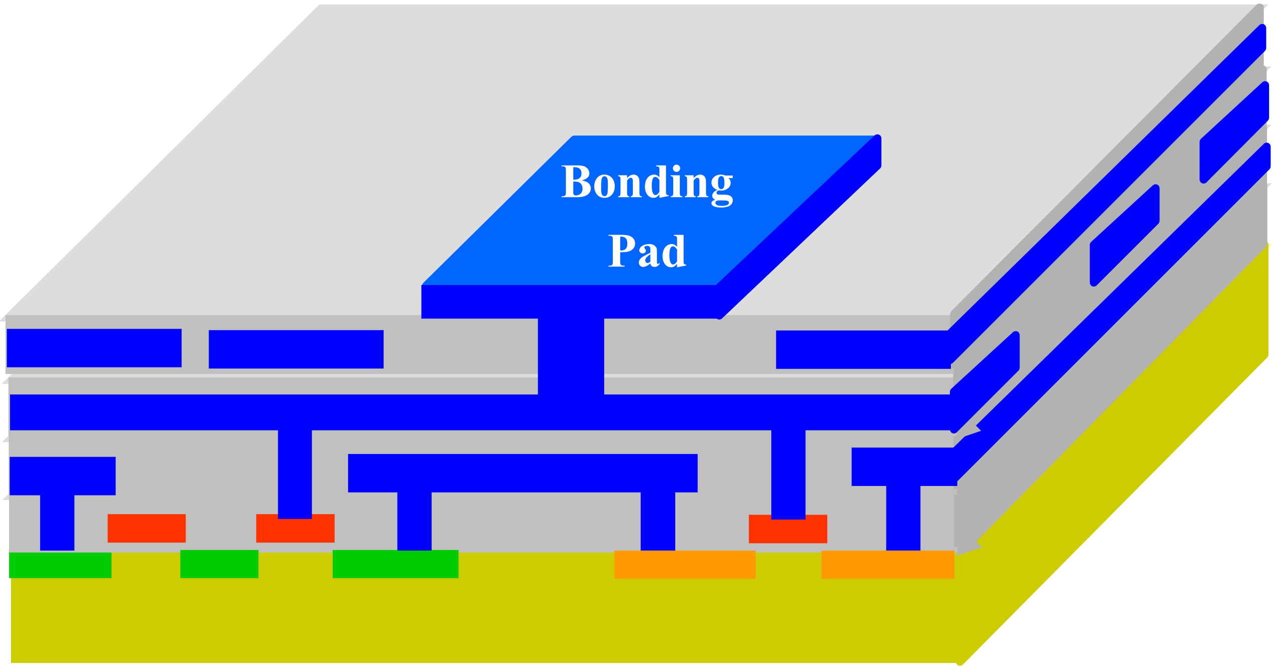


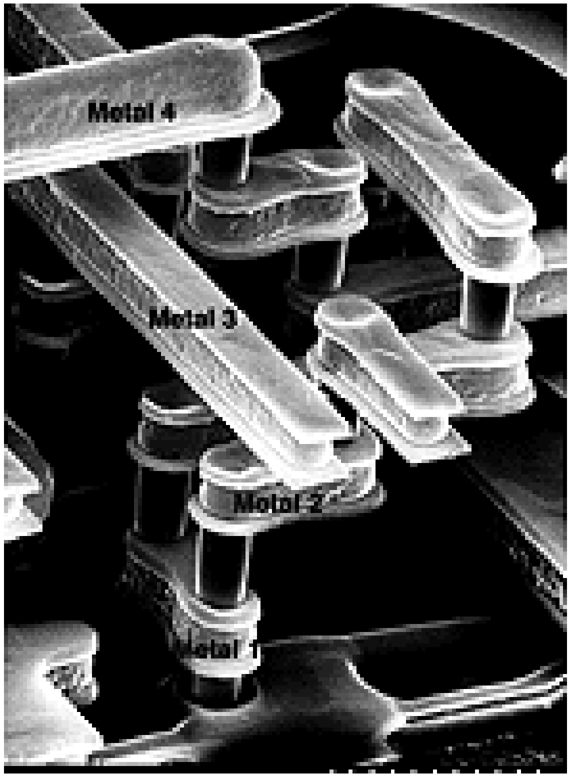
Stacked-capacitor Cell



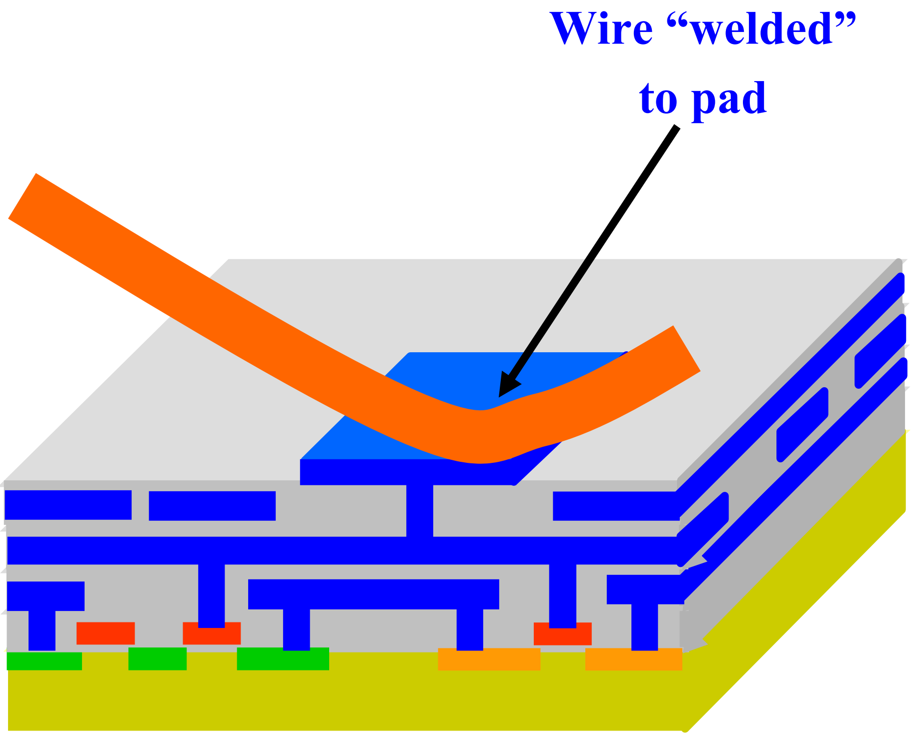
Trench Cell

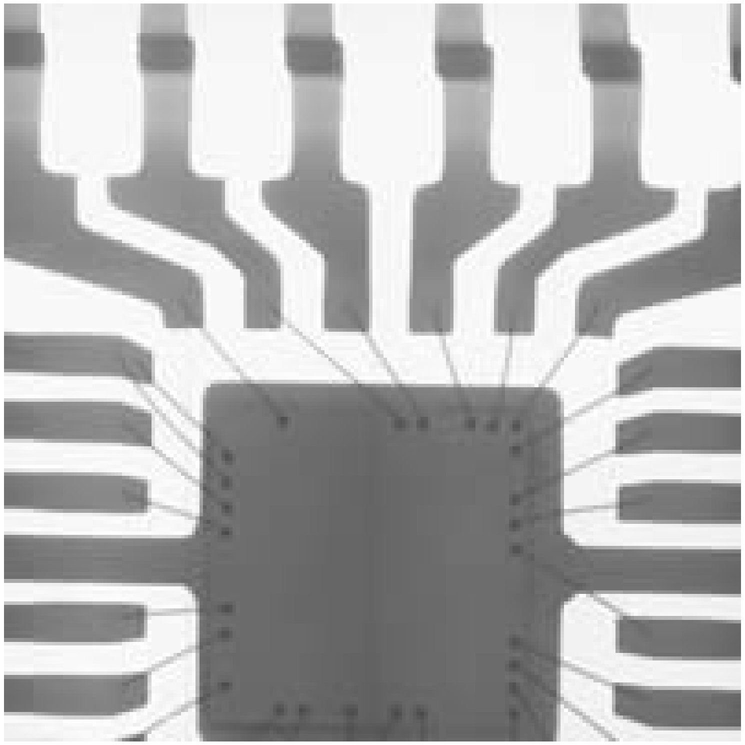
Multiple Levels of Metal





Off-Chip Interconnect: Wire Bond

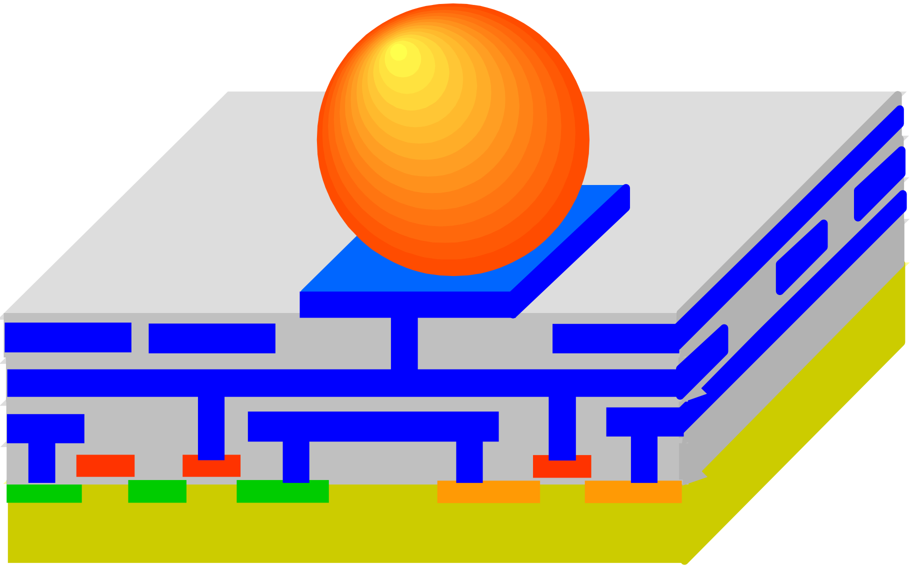




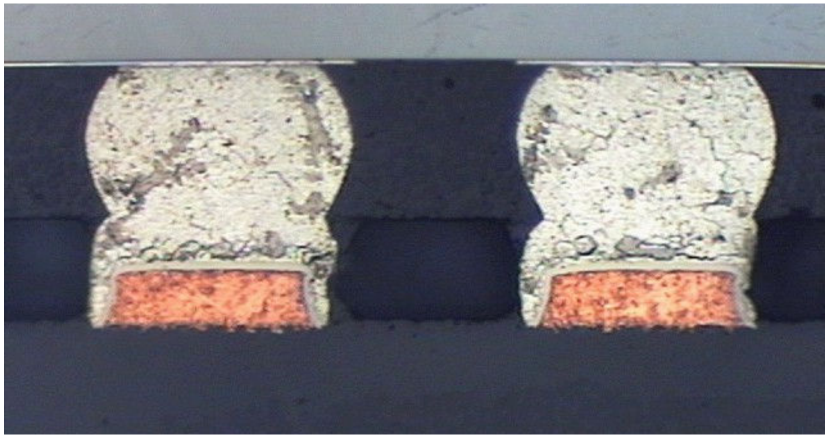
Wire Bond

Contacts available only from periphery of chip

Off-Chip Interconnect: Solder Ball



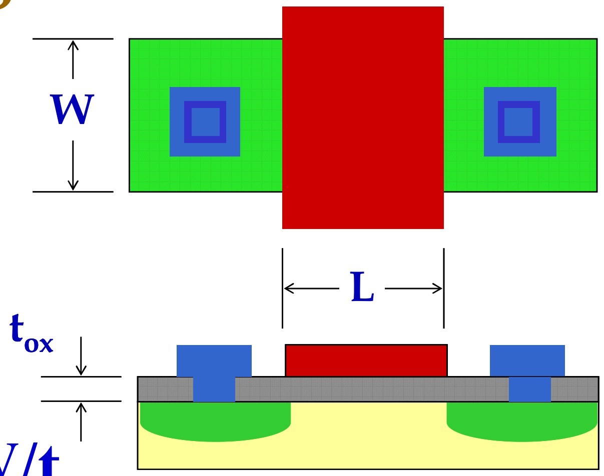
C4 Solder Ball



Allows an array of contacts over entire chip surface

Scaling

Device Scaling



- Key parameters: Gate length L, width W
- "On" resistance prop. to L/W
- "Delay" in turning transistor on
  - Function of capacitance of gate
  - In turn proportional to area/t_{ox} = LW/t_{ox}
- Decreasing L thus a "good thing"
- But desirable to keep minimum devices with "square" gates …. want to shrink W also
- Other "shrinkable" dimensions: t_{ox}, metal width, spacing between wires, …

"Scaling:" shrink a dimension by factor S

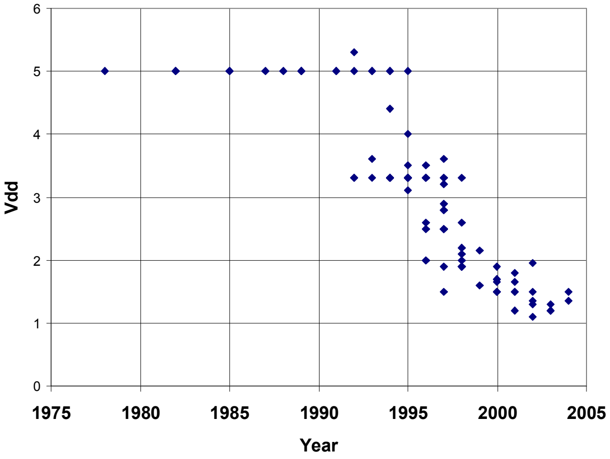
What Can Scaling Affect?

- Chip area to perform some function
  - If device & wire dimensions change by S
  - Then area changes by S^2
- Frequency of operation
  - Decreasing gate area decreases capacitance
  - Decreasing distance decreases R
    - But decreasing wire cross-section increases R
- Power to perform some function ~ C x F x V_{dd}^2
  - Decreasing gate area decreases aggregate capacitance C
  - Decreasing L decreases threshold voltage, which decreases needed V_{dd}
- Power density: power per unit area
  - Limiting factor for cooling considerations

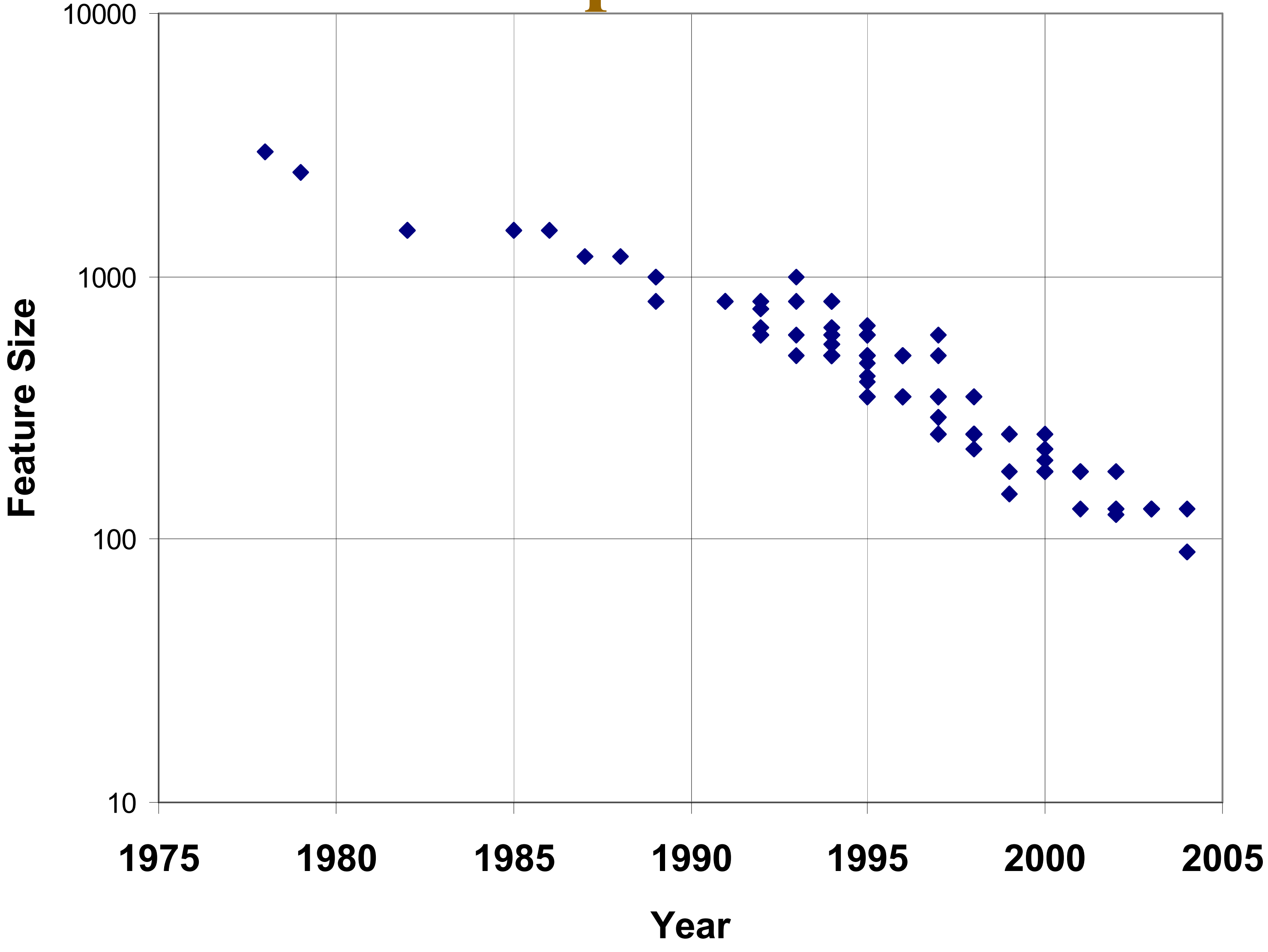
Bigger S factors are better

Approaches to Technology Scaling

- *Full scaling*: Ideal if possible
  - Keep E-field within gate capacitor constant
  - Requires scaling L, W, t_{ox}
  - Also scales voltage
  - Area shrinks, power drops, higher frequency
- *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
  - Lower peak clock, but better power & power density



Feature Size of Past Microprocessors



Approximate Scaling Relationships

	"Long Channel" Devices			"Short Channel" Devices		
Parameter	**Full**	**Fixed V**	**General**	**Full**	**Fixed V**	**General**
W, L	1/S	1/S	1/S	1/S	1/S	1/S
tox	1/S	1/S	1/S	1/S	1/S	1/S
Vdd	1/S	1	1/U	1/S	1	1/U
Circuit Area	**1/S^2**	**1/S^2**	**1/S^2**	**1/S^2**	**1/S^2**	**1/S^2**
Clock	S	S^2	S^2/U	S	S	S
Circuit Power	1/S^2	S	S/U^3	1/S^2	1	1/U^2
Power Density	1	**S^3**	S^3/U^3	1	**S^2**	S^2/U^2

Moore's Law:

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

ITRS

- The Process
- A Technology Node
- Key Technology Projections

International Technology Roadmap for Semiconductors

- Goal: predict semiconductor 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”

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)
- *DRAM*: high threshold transistors, few metal, cheap fab processes
  - – High Volume Commodity Dense memory part
- *Embedded DRAM*: DRAM circuits made on logic process (faster, but less dense)

Trends Driven by Scaling

- *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 Addressed

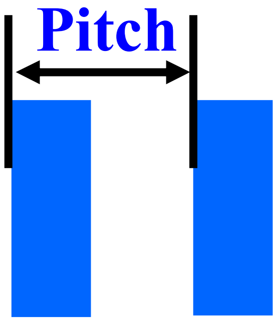
- System Drivers & Design
- Test & Test Equipment
- Process Integration, Devices, & Structures
  - Including RF, mixed signal, emerging
- Front End Processes
- Lithography
- Interconnect
- Factory Integration
- Assembly & Packaging
- Environmental Safety & Health
- Yield Enhancement
- Metrology
- Modeling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

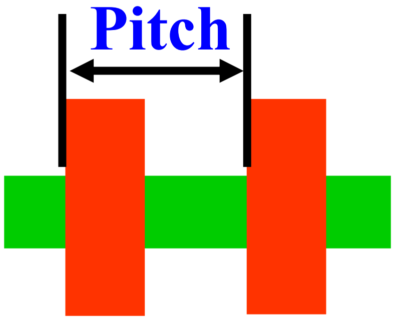
Technology Node

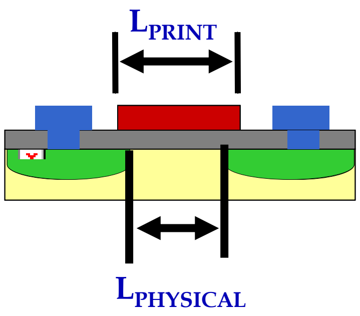
- Goal: "Label" state of the art to allow quick correlation to Moore's Law scaling
- *Technology Generation* for Year X:
  - Minimum feature size in any product in that year
- *Technology Node*:
  - A year in which technology generation provides ~4X functionality growth over prior Technology Node
  - Typically tied to DRAM, as that is usually smallest
  - Based on *Year of Production*

Interesting Feature Sizes

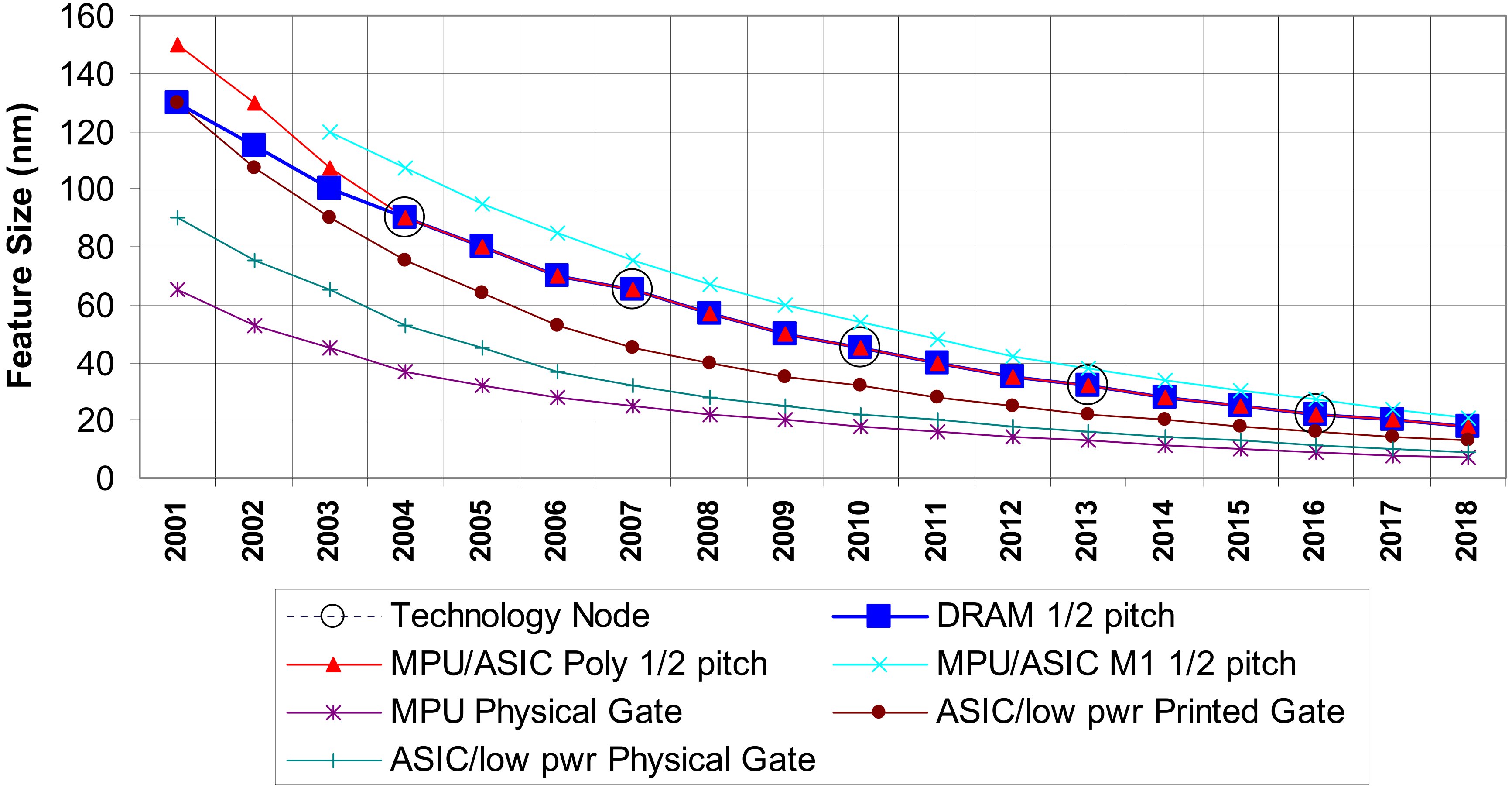
- ½ of minimum pitch between two DRAM metal lines
- ½ of minimum pitch between two microprocessor metal lines
- ½ of minimum pitch between two microprocessor poly lines
- Gate length of a microprocessor transistor gate “as printed”
- Gate length of a microprocessor transistor gate “as physically fabricated”



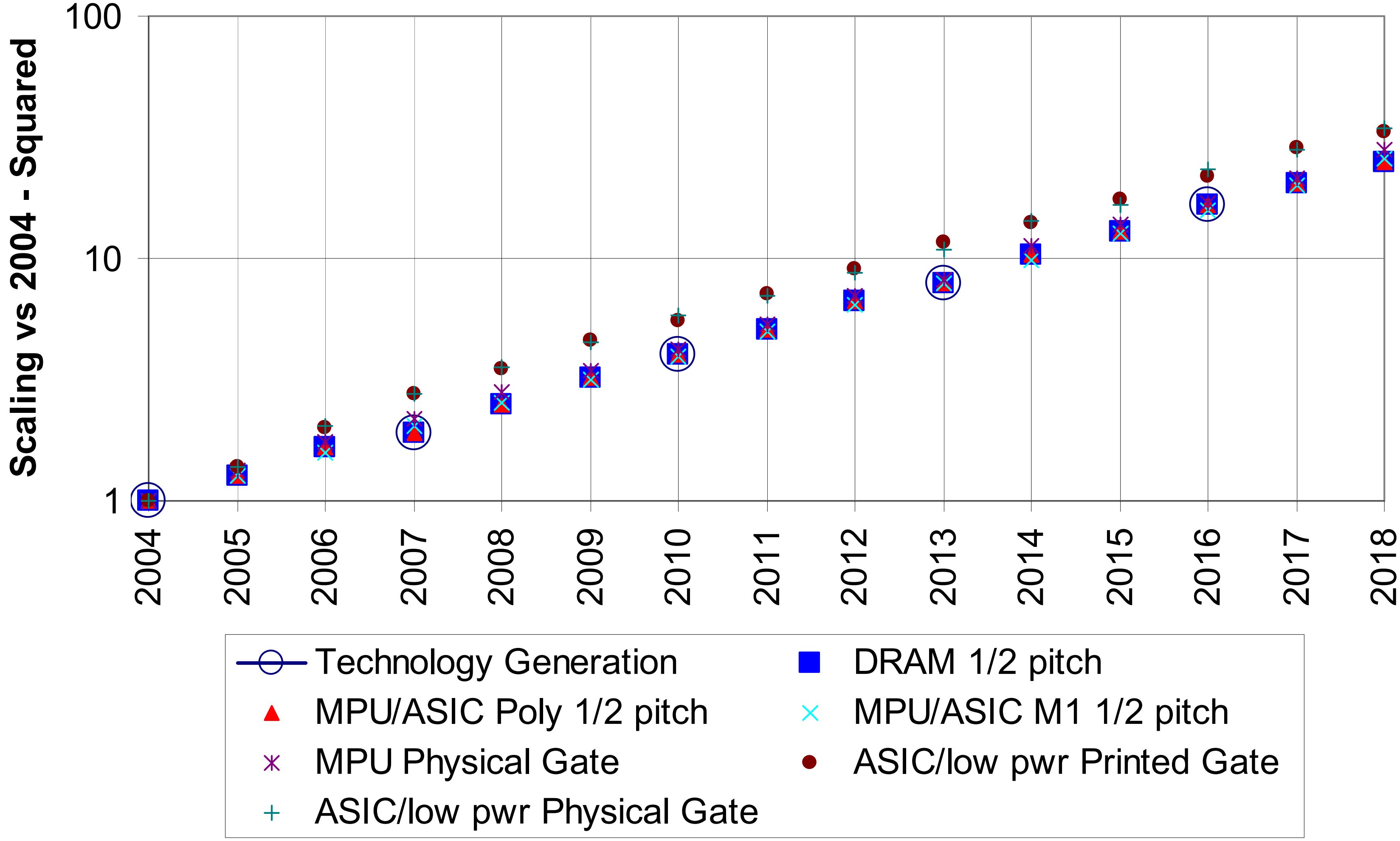




Feature Size Projections



Projections as Scale Factors



Basic area scaling doubles every 3 years

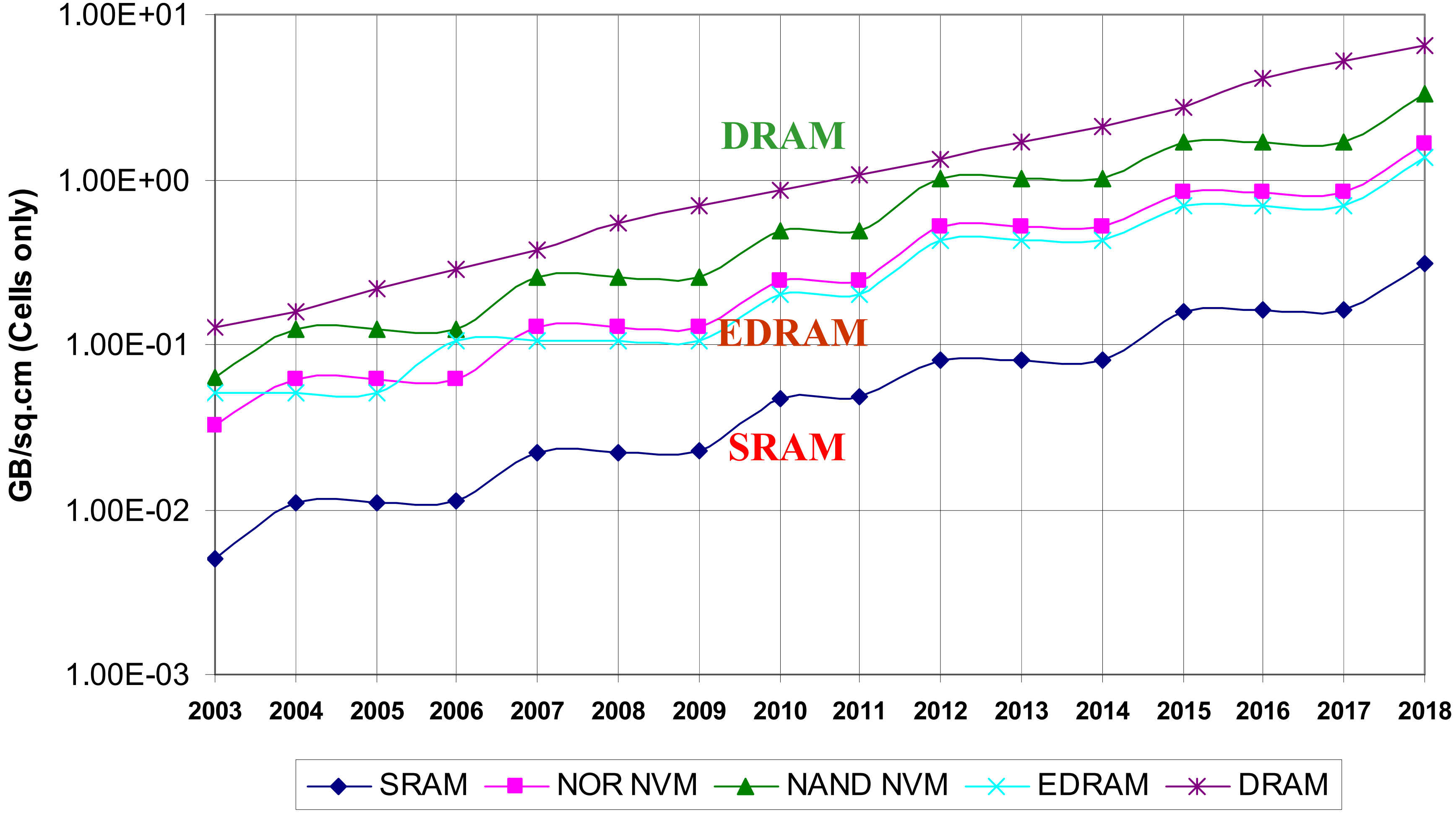
Comparison to Moore's Law

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

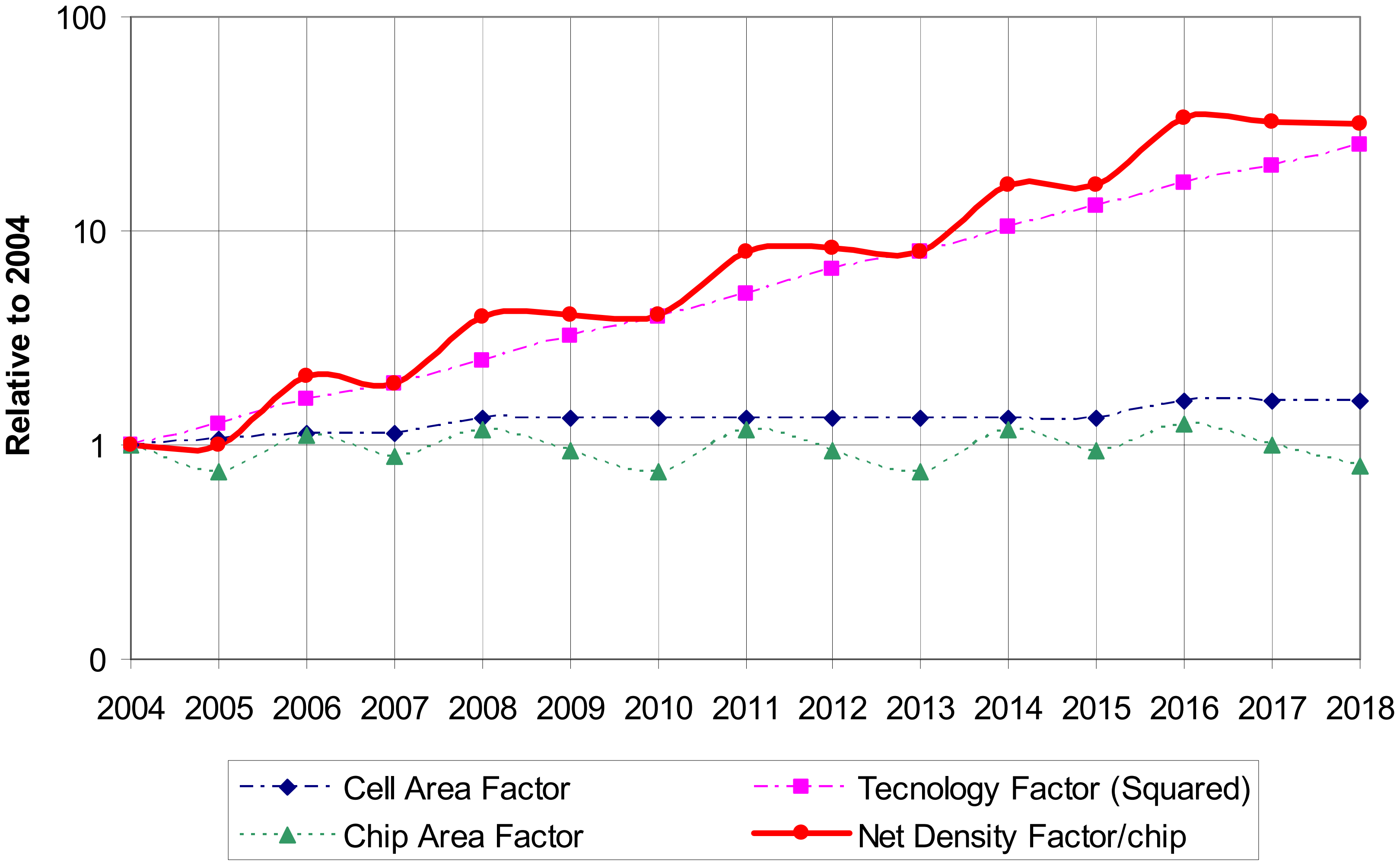
Commodity DRAM Capacity

- *Chip Capacity:* Product of
  - Cell area
  - Chip area
  - % of chip that is cell array
- *Cell area factor*:
  - technology-independent area of one bit
  - Decreasing slowly over time
- *Cell Area*: product of factor & feature size^2
- *Chip Area:* now chosen to maximize yield
- *Cell Array area*: % of chip that is cell
  - Constant at 63% in production

Memory Storage Density: Cells Only

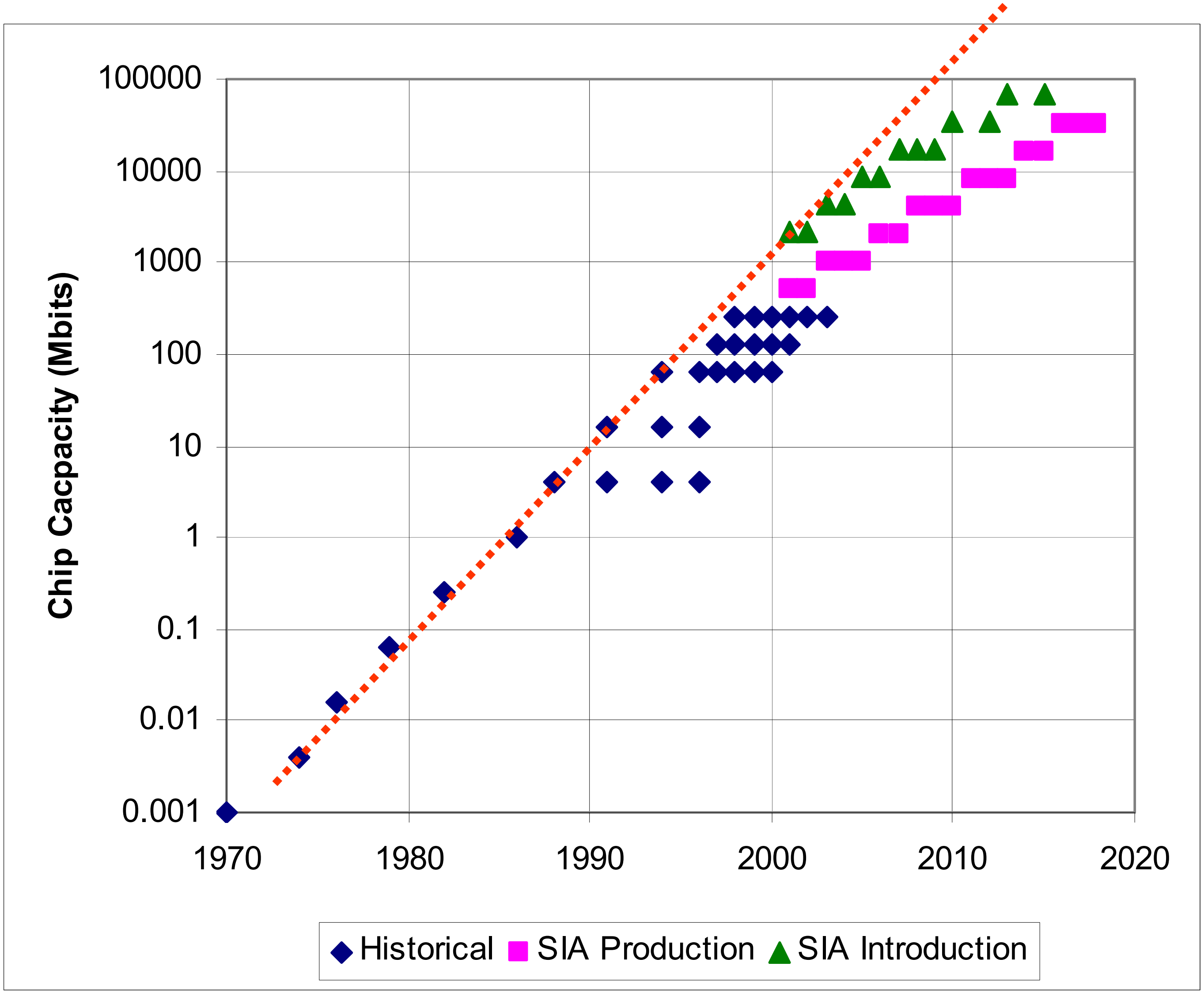


Change in DRAM Density Factors



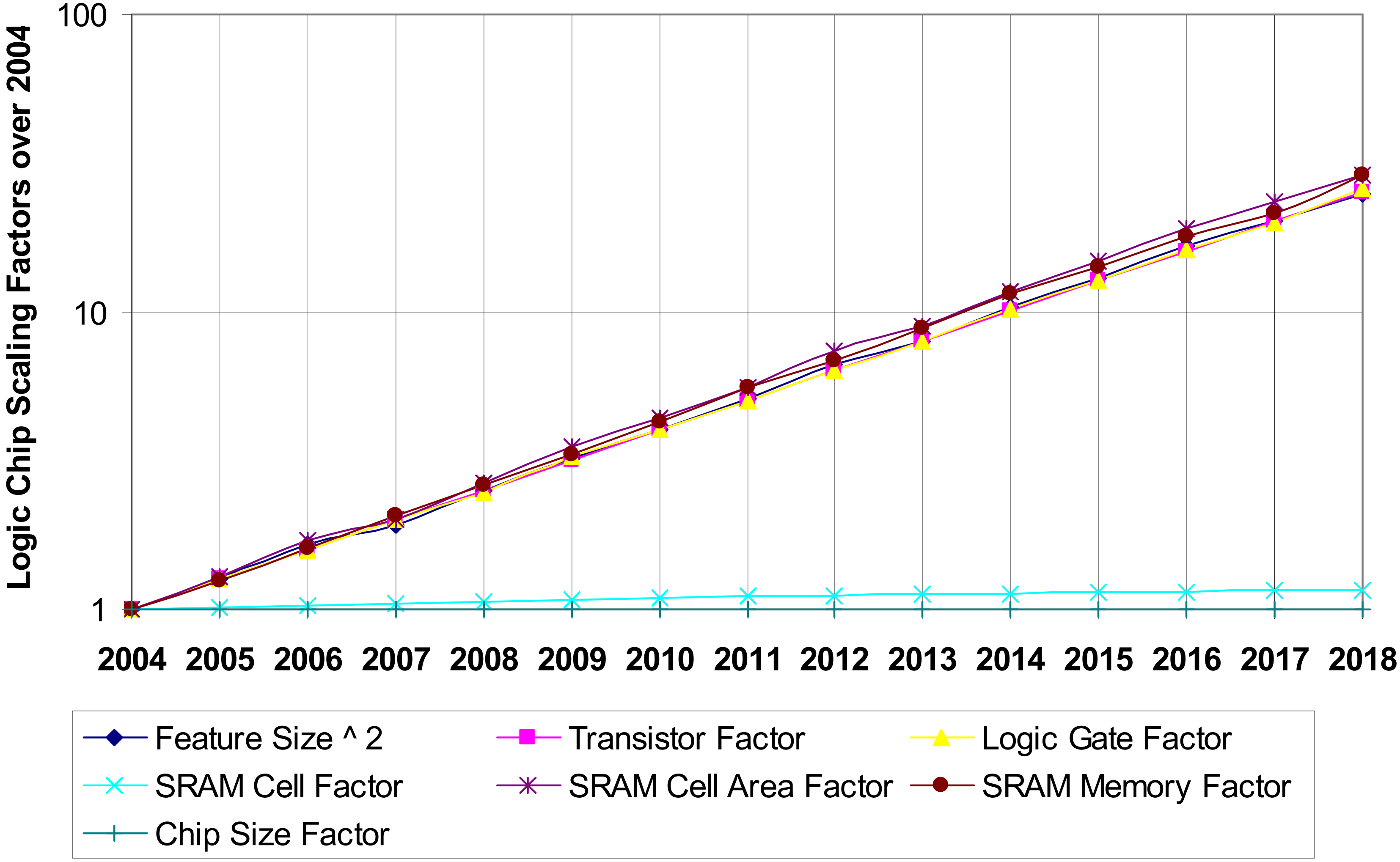
DRAM is now $-Driven – not Density-Driven

Chip Capacity



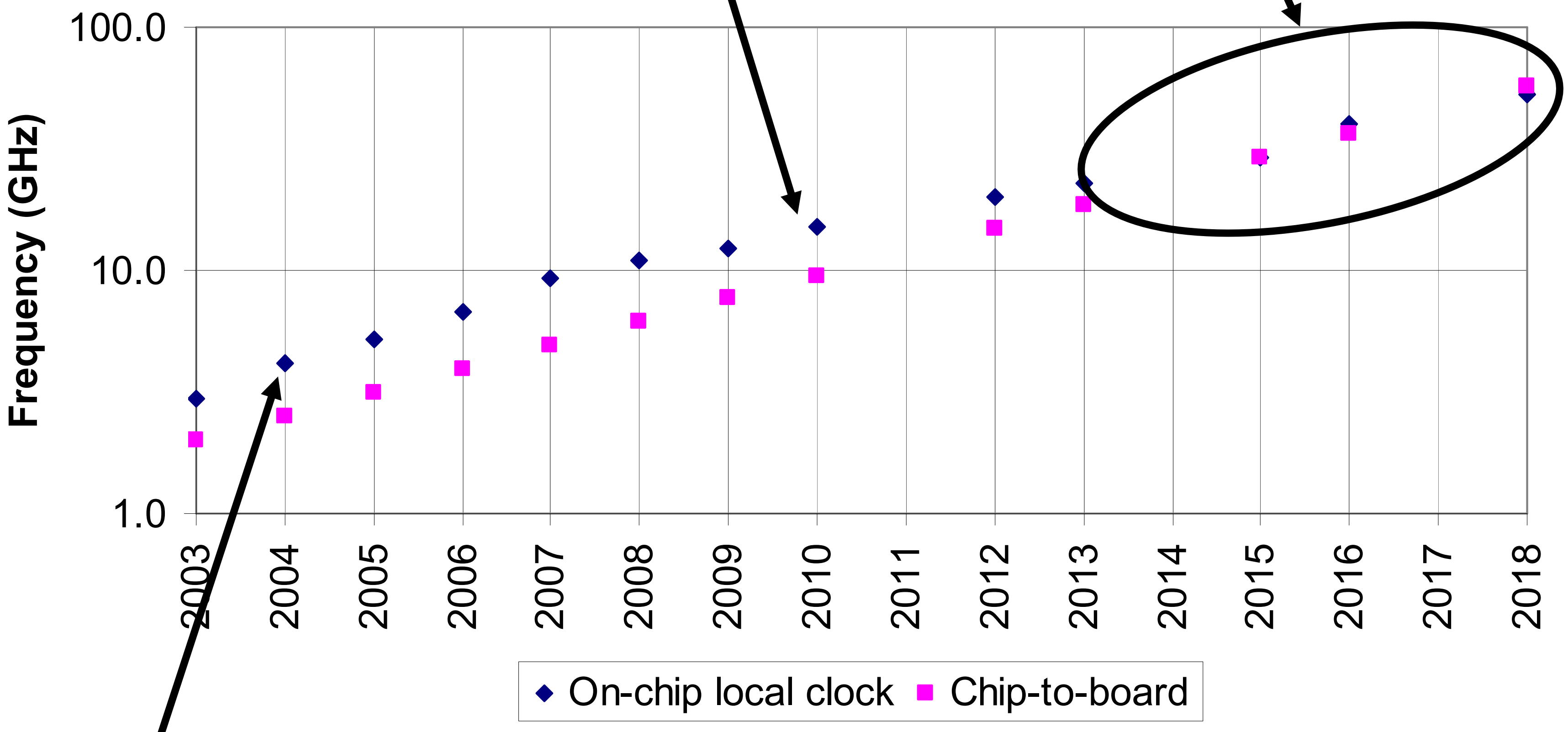
Chip Capacity is No Longer Following Original Moore's Law

Logic Chip Density Scaling



Logic functions per chip: ~2X every 3 years

Logic Clock Rates



On-chip clock rates are flattening

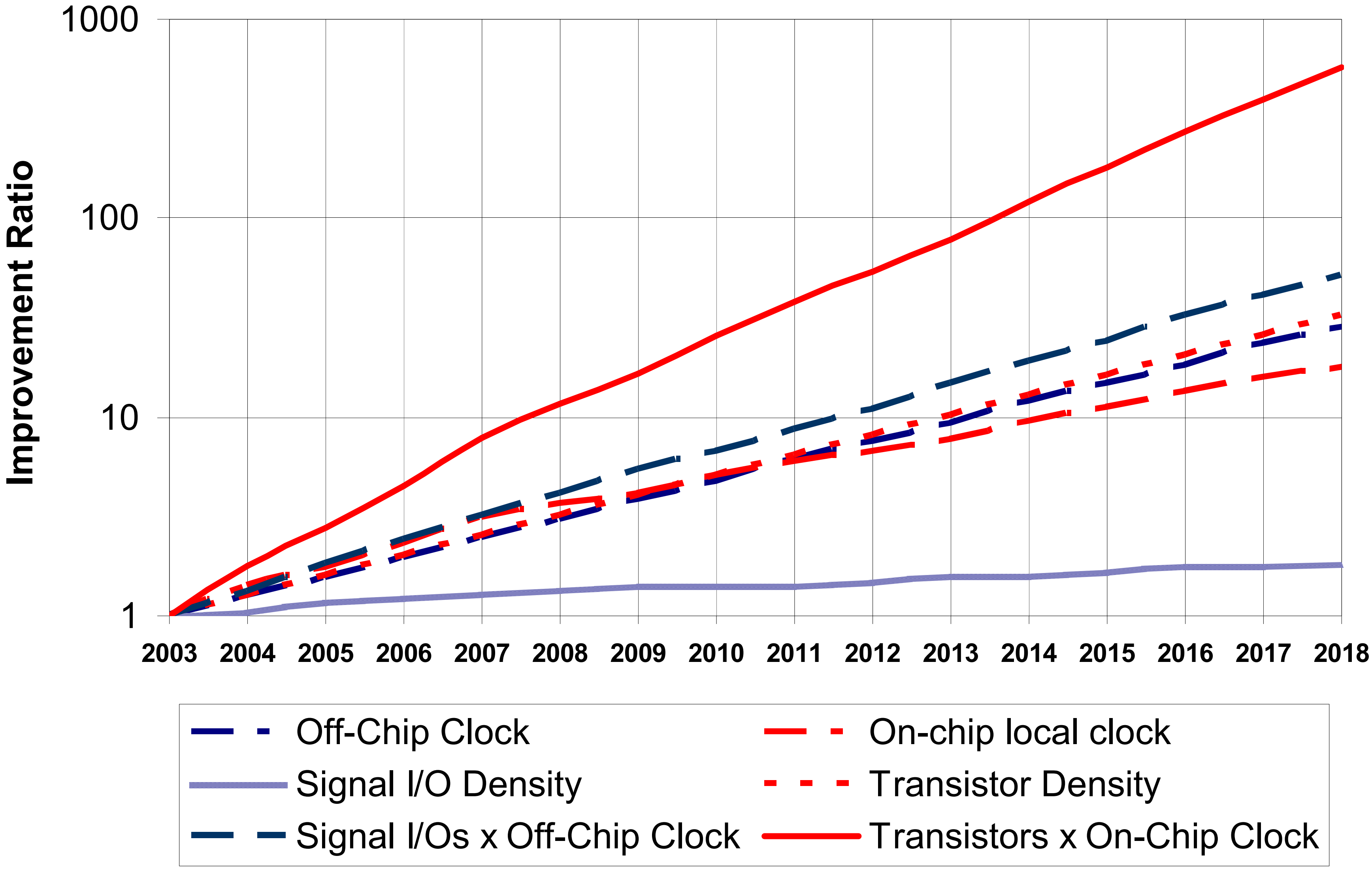
And then magic happens

2004 Projection was 5.2 GHz - and we didn’t make it!!!

Off Chip Bandwidth

- 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

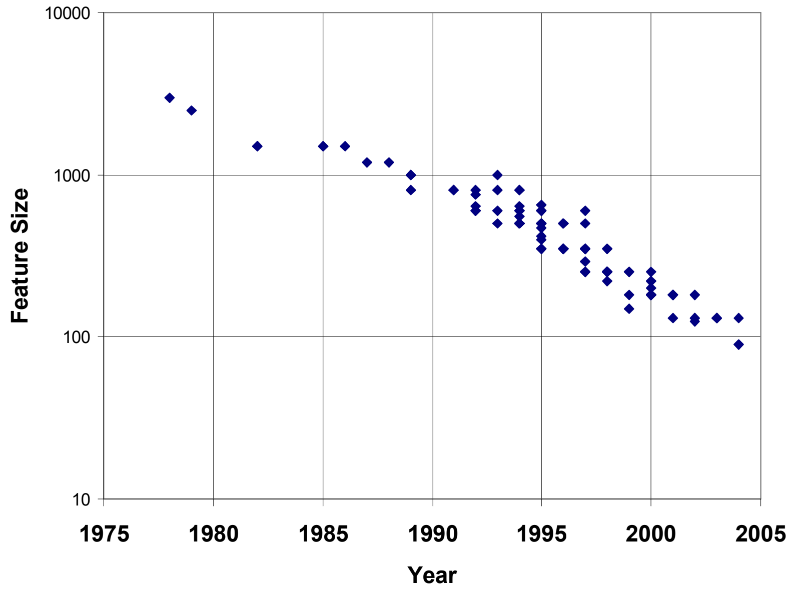
Relative Off-Chip Scale Factors

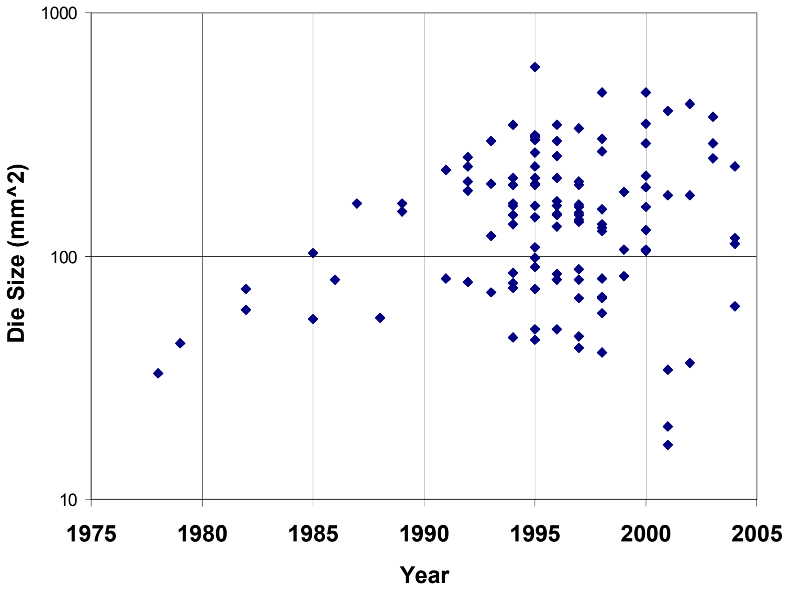


The Way We Were: A Brief Romp Thru Single Chip Microprocessor Land

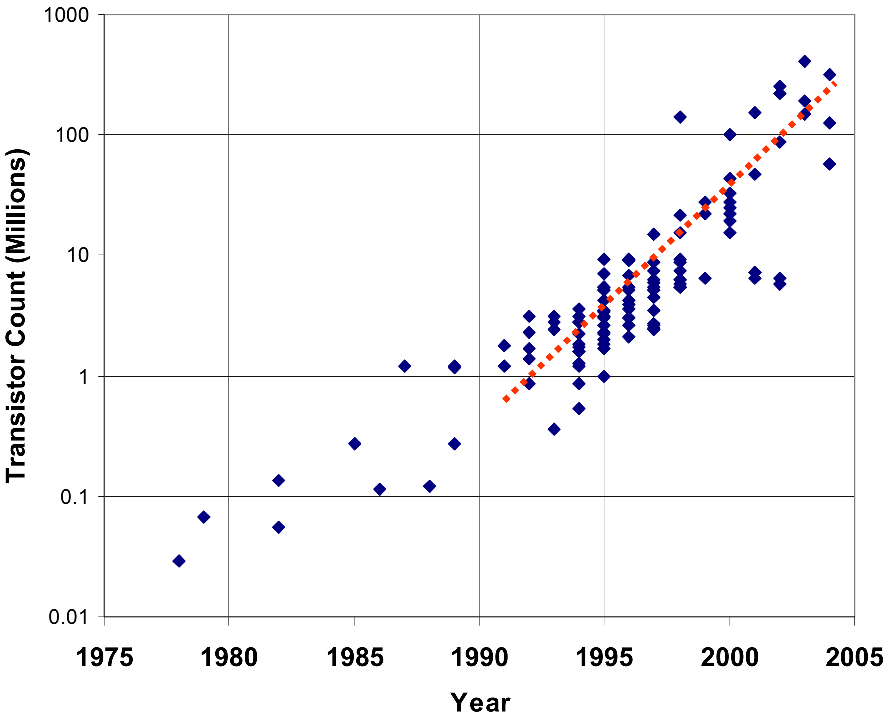
- Data from last 30 years of real chips

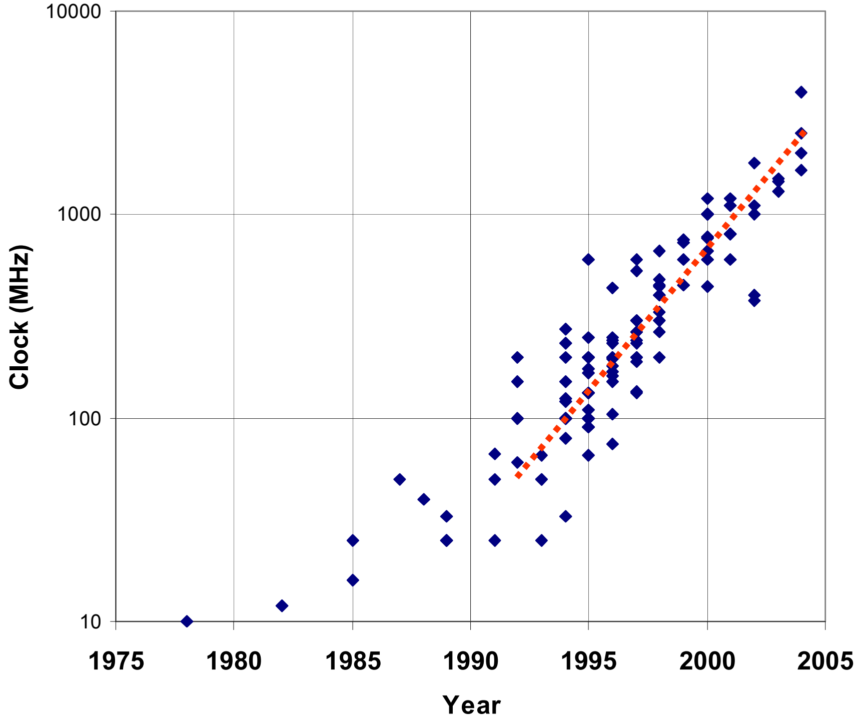
Historical Changes in Chip Parameters





Functionality = IPC x Clock





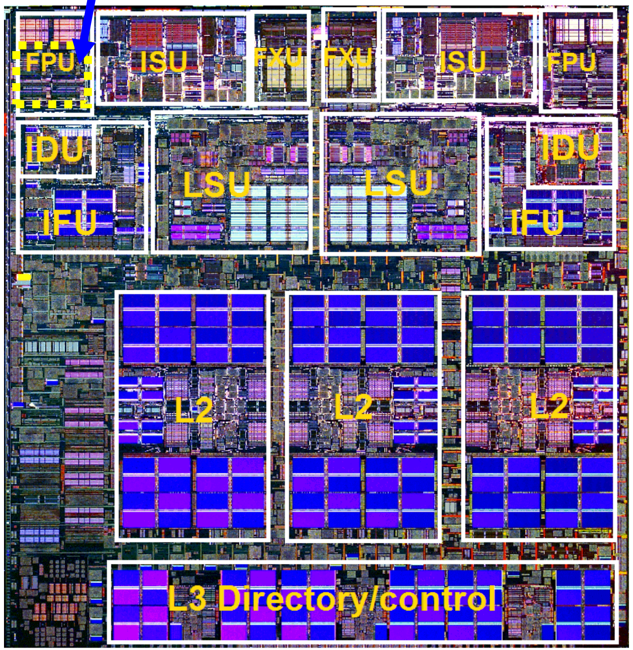
- ~ 4X per die every 3 years
- But: Most in cache
- And partially due to larger die
- And off-chip clock rates lagging

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

How Are We Using These Transistors

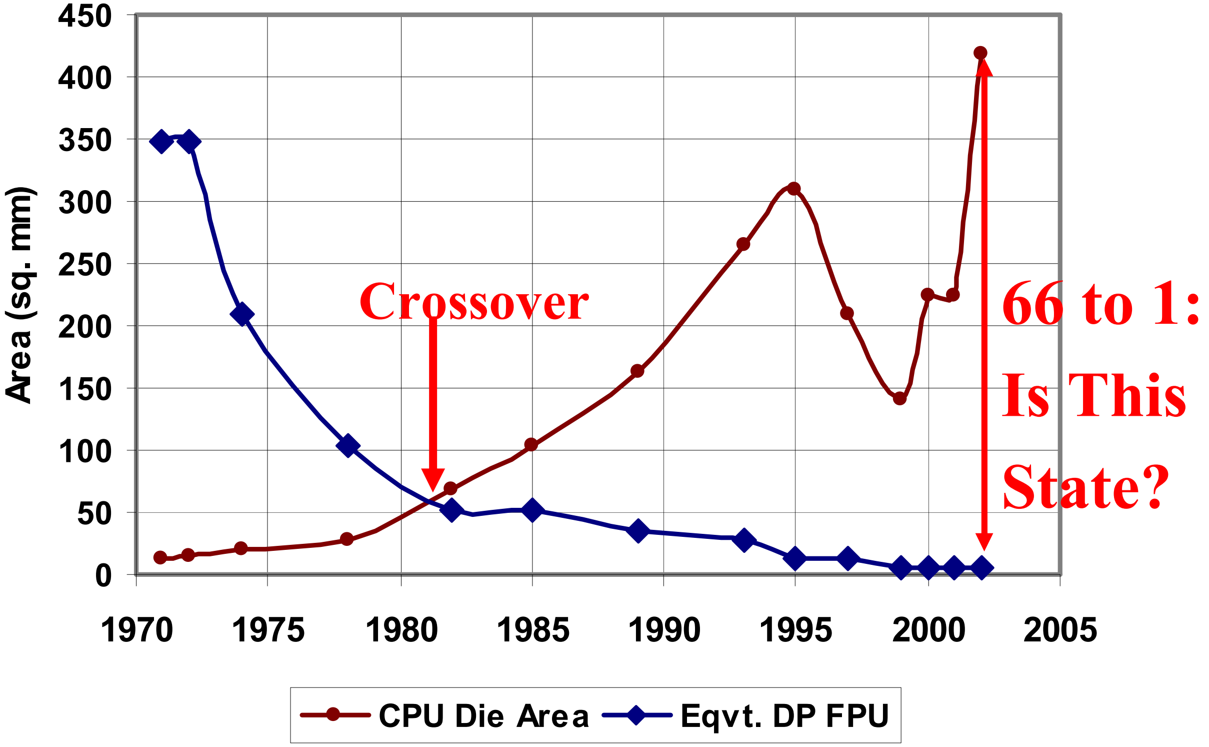
*

IBM P5 Dual Core



36MB SRAM L3 chip

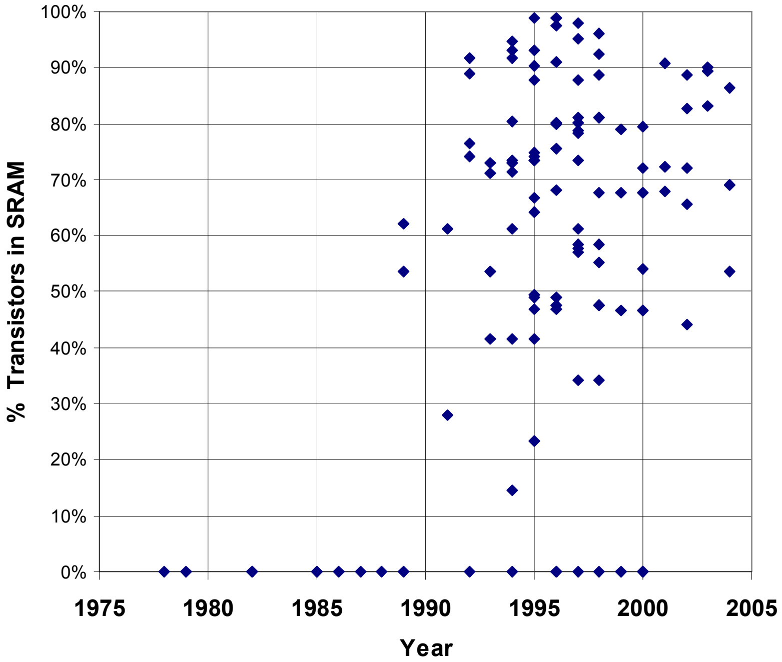
Intel Single Core Family



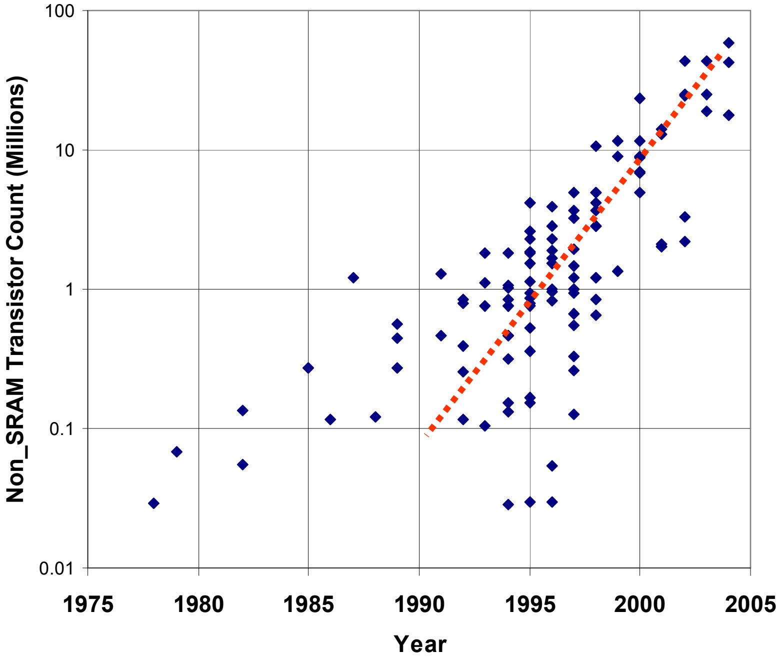
Crossover

66 to 1: Is This State?

Let’s Look at Transistor Count

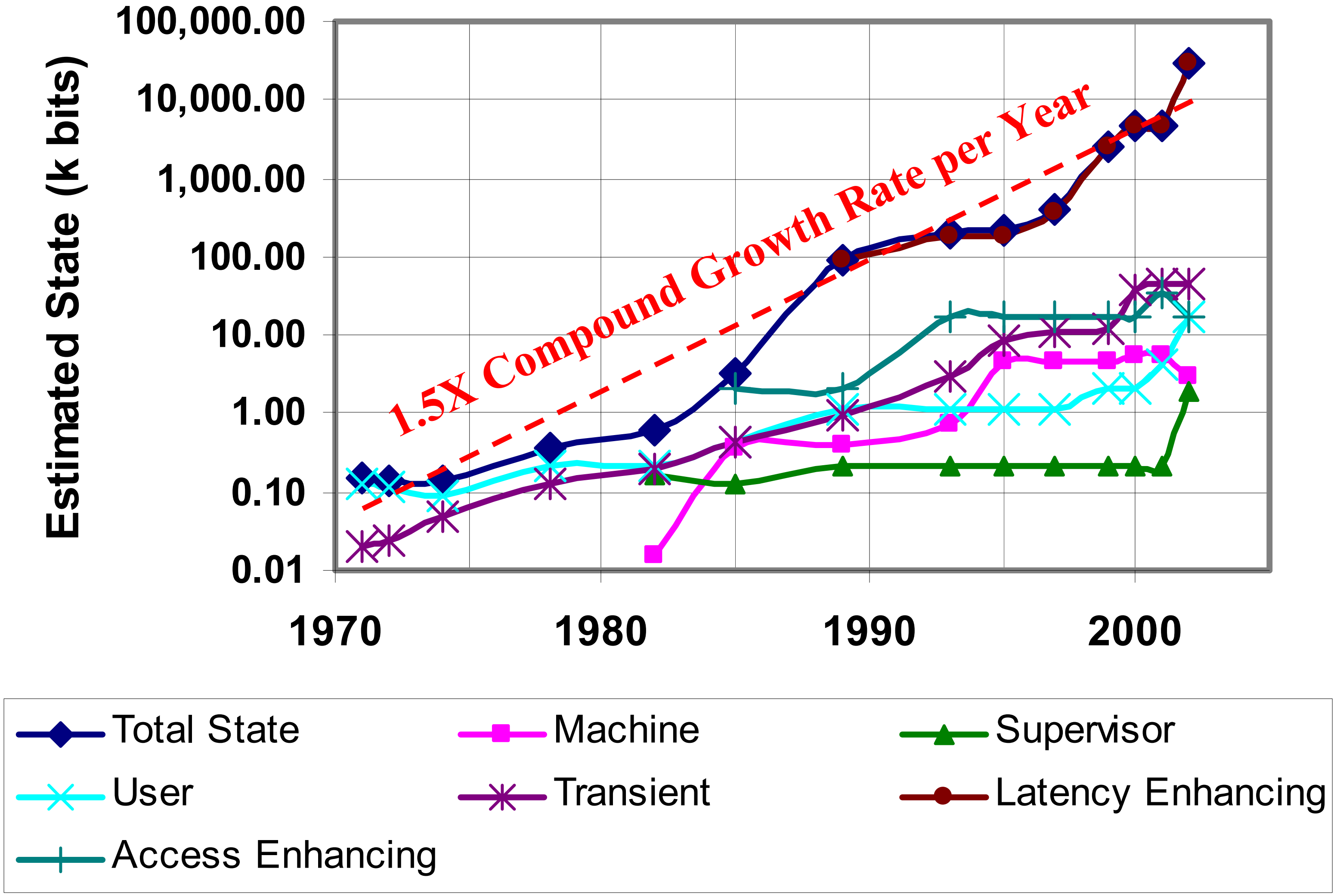


- Most of uP die = SRAM

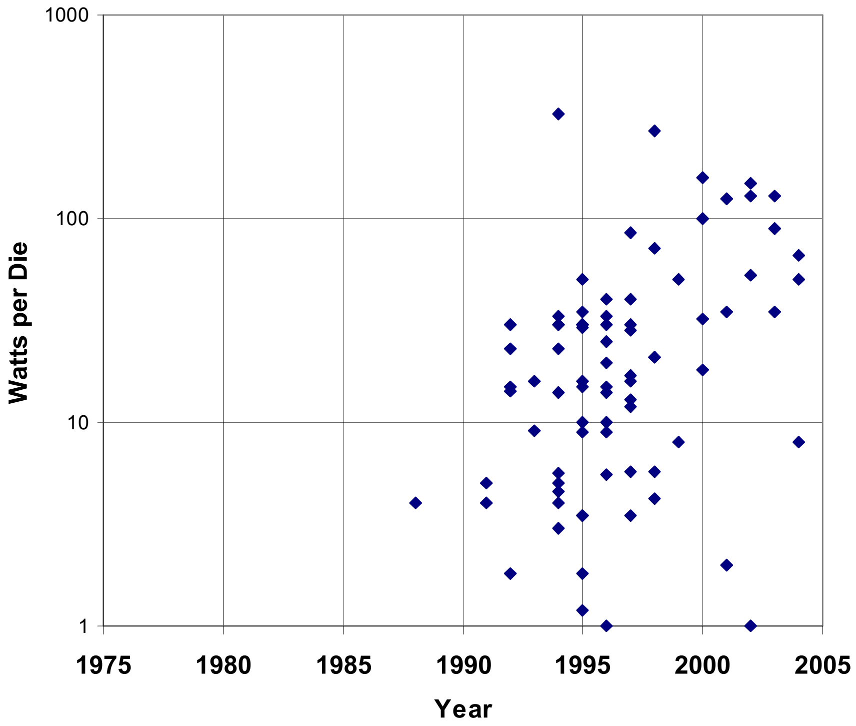


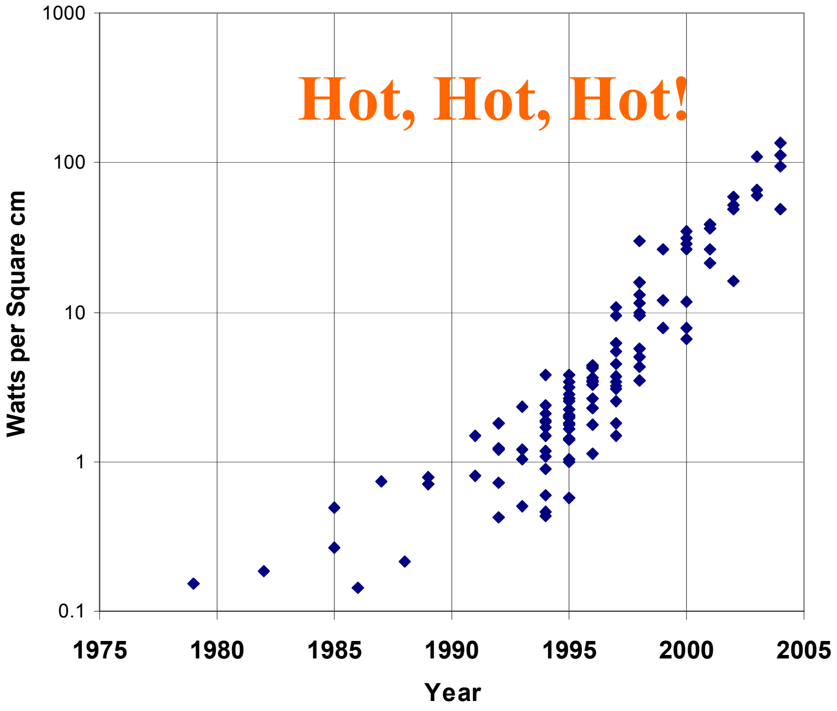
- Still ~4X every 3 years
- But N-way superscalar at best perhaps sqrt(N) IPC
- Again highly latency driven
- & hideously expensive to design

Core CPU State vs Time



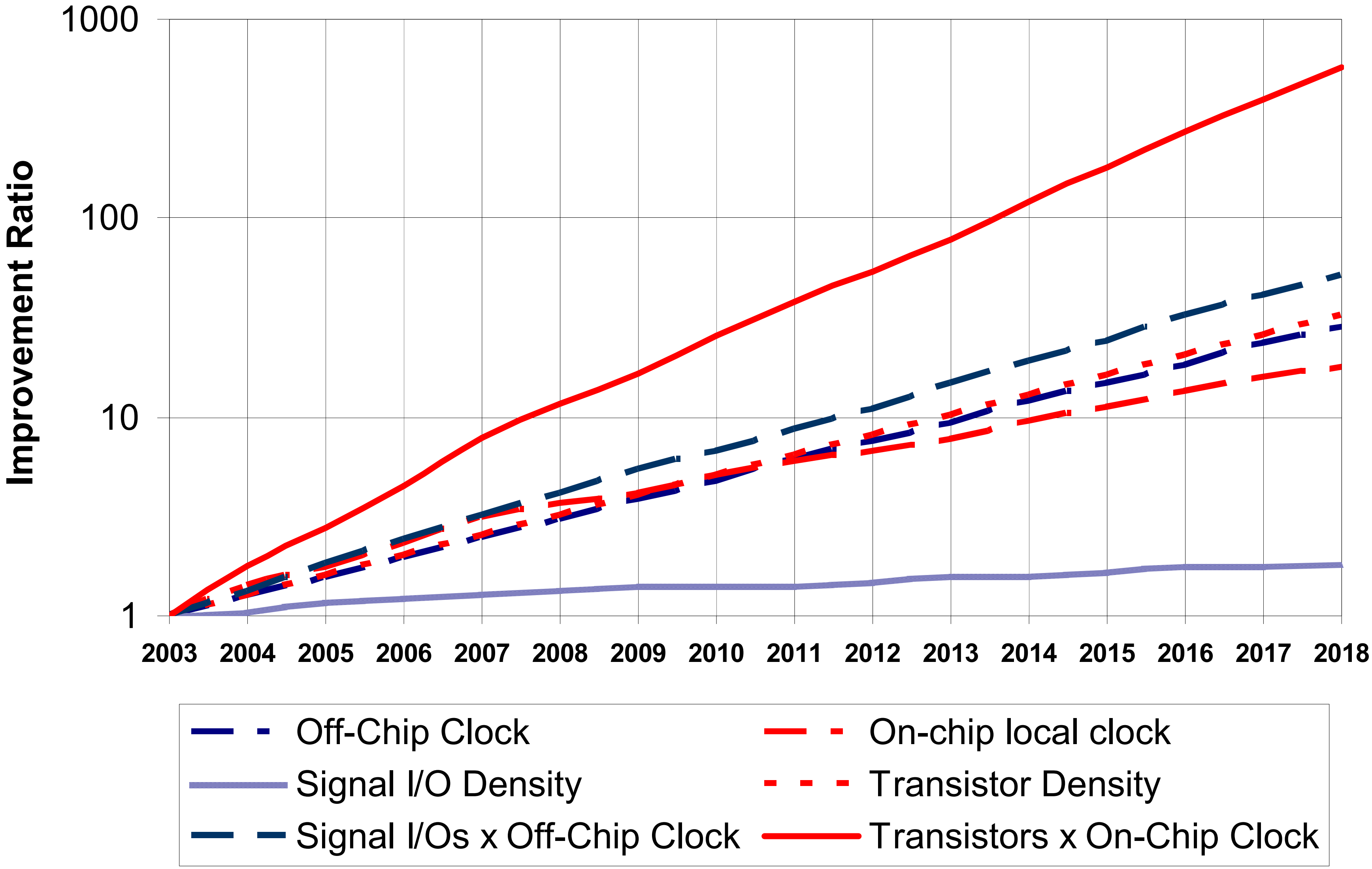
Power


1000
100
10
1
Watts per Die
1975
1980
1985
1990
1995
2000
2005
Year

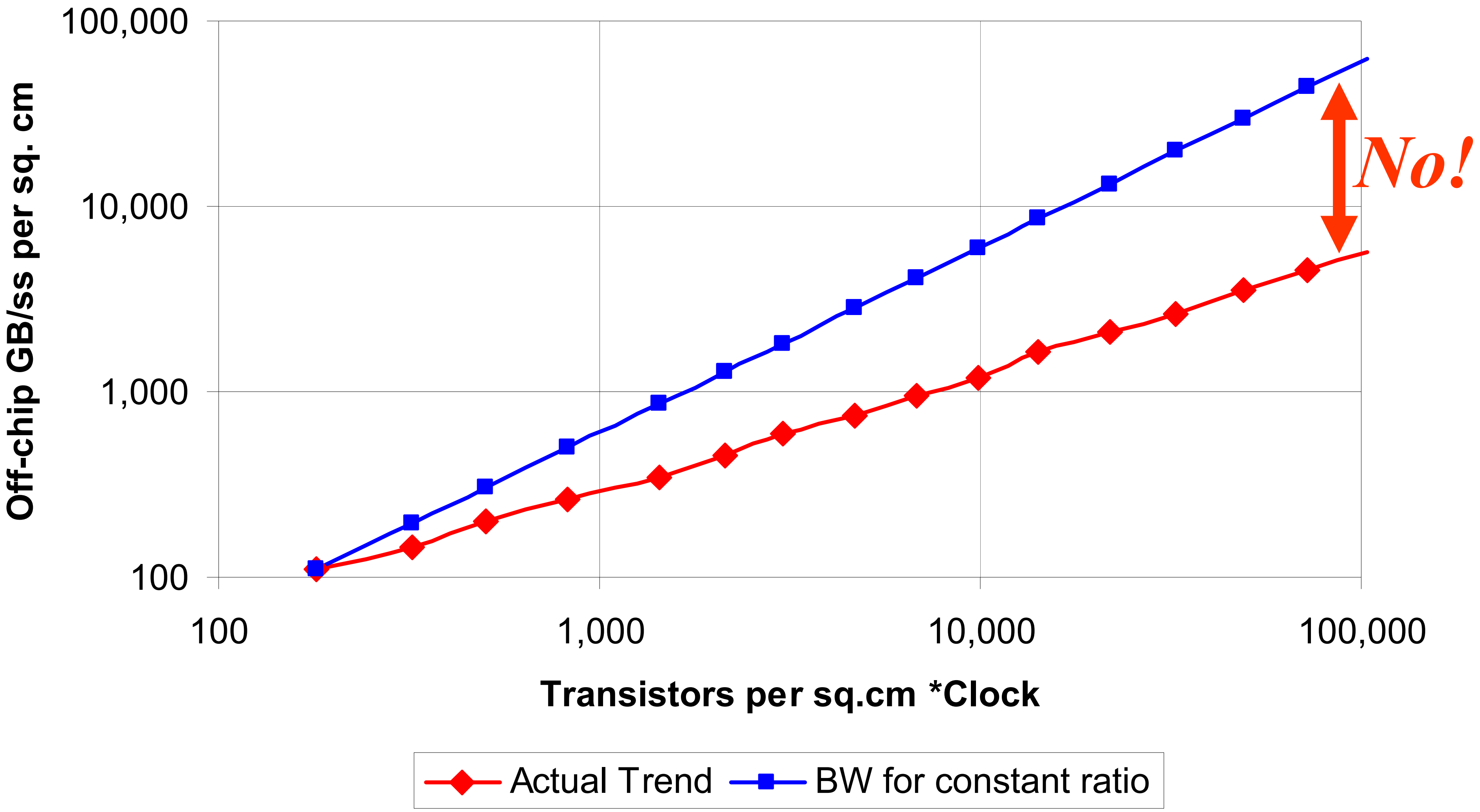

1000
100
10
1
0.1
Watts per Square cm
Hot, Hot, Hot!
1975
1980
1985
1990
1995
2000
2005
Year

Relative Off-Chip Scale Factors

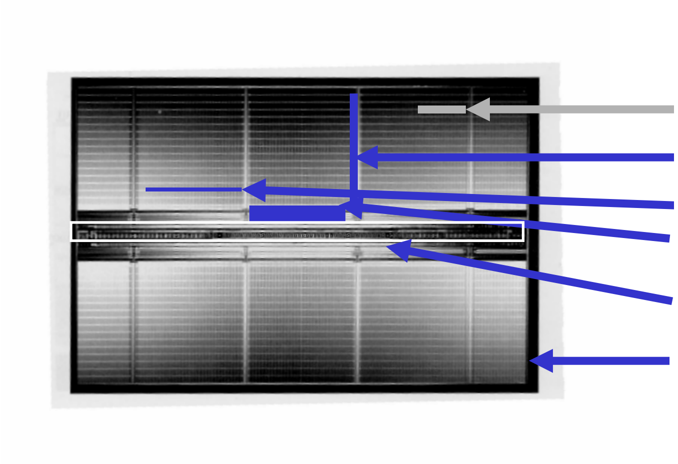
(Repeat of Earlier Chart)



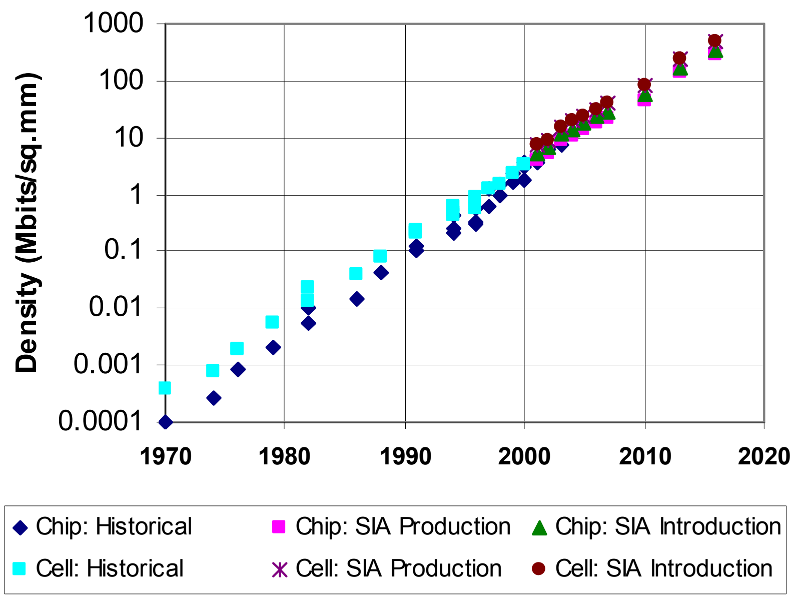
Does Logic Performance Match Off-chip Bandwidth Potential?



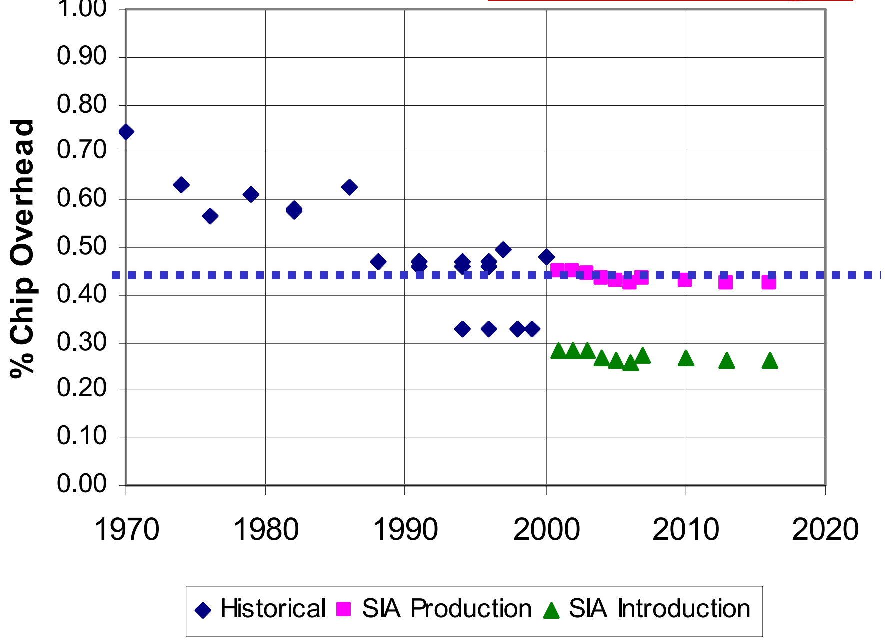
Classical DRAM



- Memory mats: ~ 1 Mbit each
- Row Decoders
- Primary Sense Amps
- Secondary sense amps & “page” multiplexing
- Timing, BIST, Interface
- Kerf



45% of Die is <u>non storage</u>



Memory Interfaces

- Today: DRAM chips separated from uP
- Latency: sum of
  - Time to get address from uP to DRAM
  - Time to access internal DRAM arrays
  - Time to pick out particular nibble
  - Time to send back to CPU
- Bandwidth: Function of
  - Number of pins off of uP die
  - Max signaling rate to DRAM
  - Ability of DRAM to overlap multiple operations

Improving only 7%/yr

Remember: DRAM uses slower transistors

Which leaves less space for memory

The Brave New World: Adding More Threads to a Single Die

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

Multi-Threading

- Thread: execution of a series of inter-dependent instructions in support of a single program
- Today's single threaded CPUs
  - Dependencies in program code reduce ability to keep function units busy
  - Limited in support for memory operations "in flight"
- Multi-threading: allowing multiple threads to take turns using same CPU logic
  - Typical requirement: multiple register sets
- Variations in terms of when/how instructions from all active threads are issued
  - Coarse-grained MT: Issue from one thread & change only at some major event
  - Fine grained MT: Change every few instructions
  - Simultaneous Multi-threading (SMT): actually interleave instructions from multiple threads

Advantages

- Hide long-latency memory operations by switching to other threads
- Have larger pool of unrelated instructions to use to feed function units
- Simplify scheduling of multiple activities and still guarantee forward progress for each
- In SMT designs: guaranteed independent instructions in pipelines eliminates need for expensive forwarding and reordering

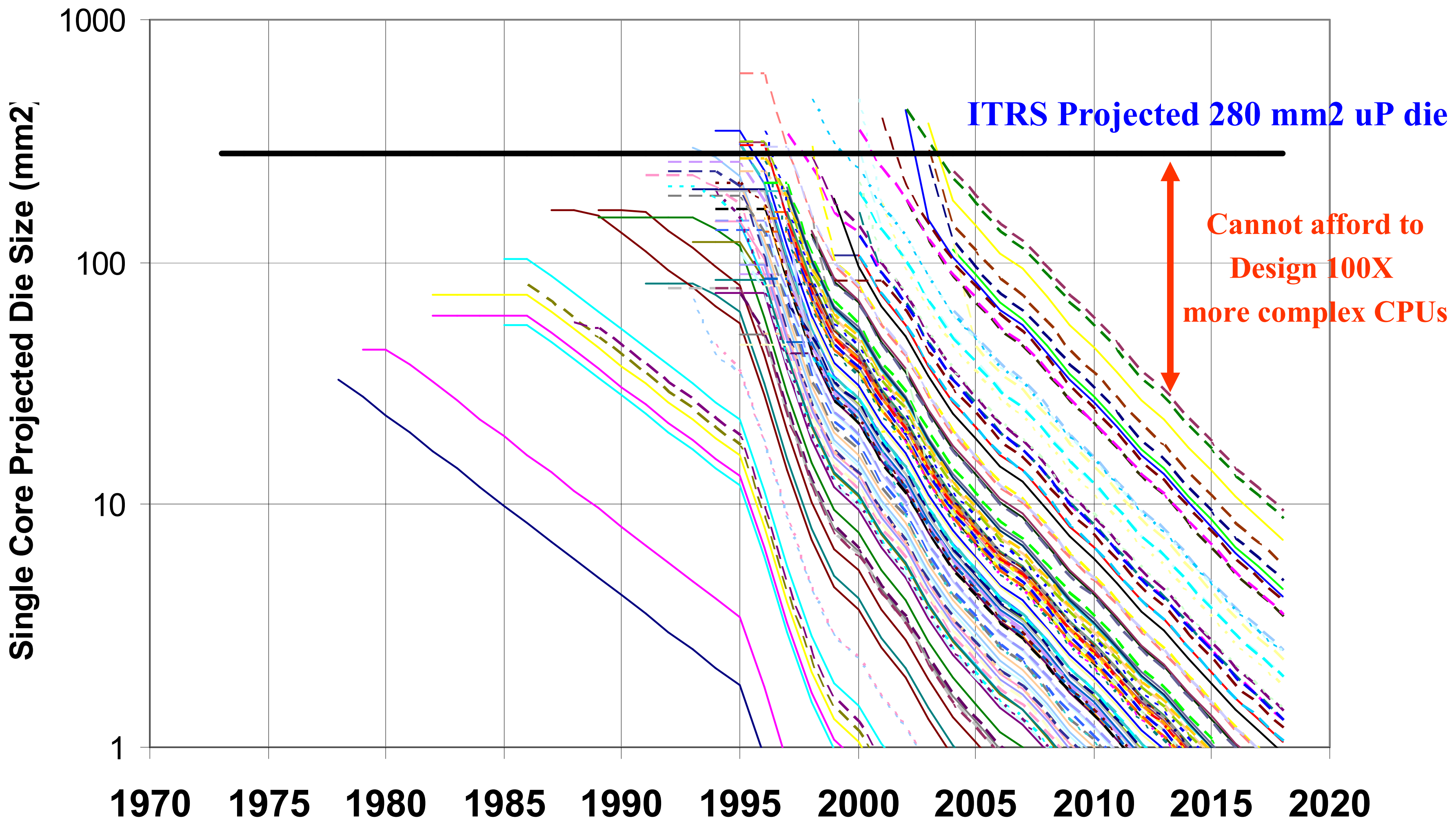
Examples of Multi-Threaded Designs

- 1960s: CDC 6600 I/O Processor
- 1970s: Space Shuttle I/O Processor
- 1980s: Denelcor HEP
- 1990s: Cray MTA
- Recent machines
  - Intel Hyperthreading: 2 threads/core
  - SUN MAJC chip
  - POWER5 dual thread dual core
  - PIM Lite: Multi-threading “at the memory”
  - Sun Niagara 8 core 4-way multi-threaded chip
  - Cray Coronado chip 32-way threading

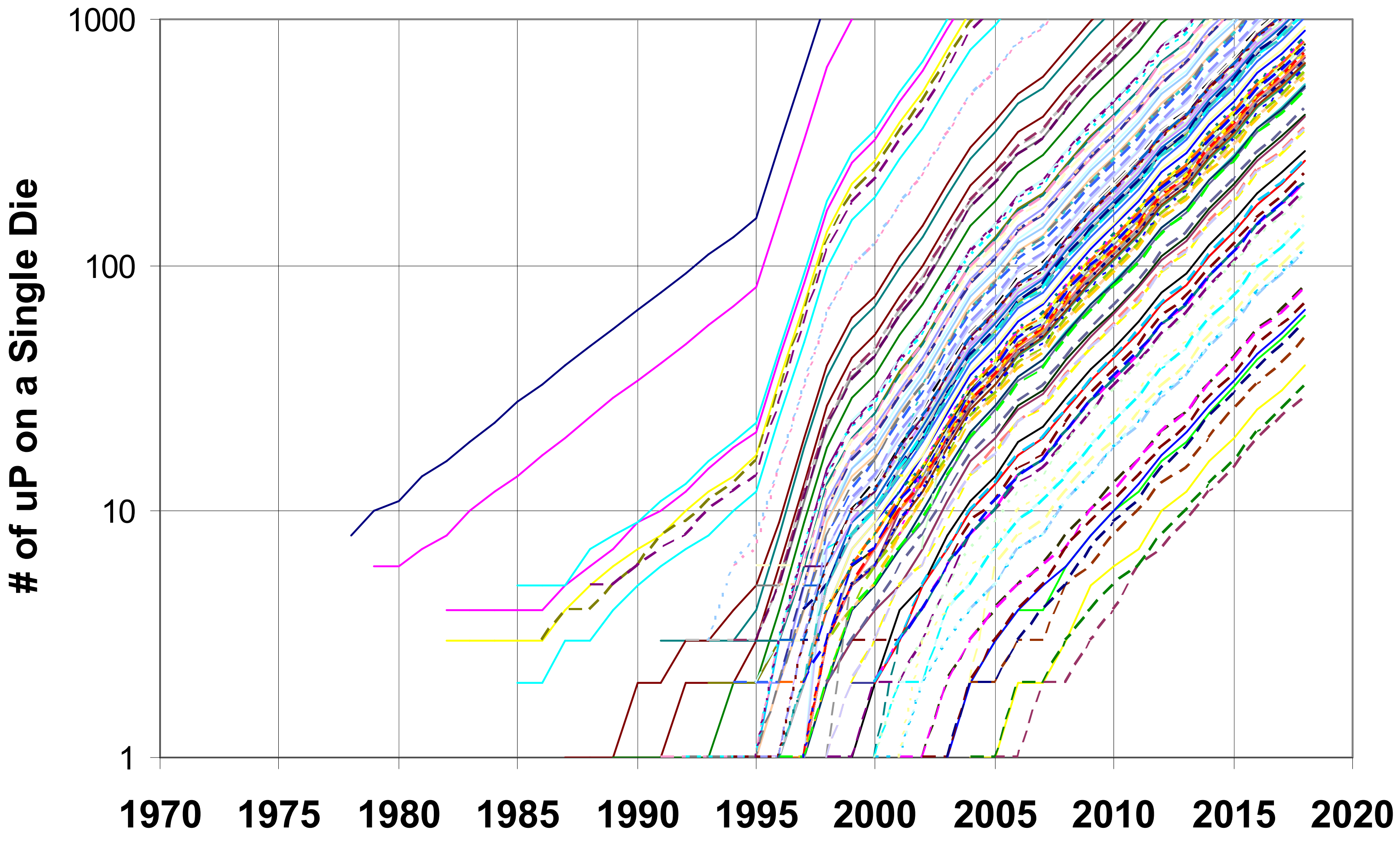
Multi-Core

- 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 but don't use inherent bandwidth
- Solution: "reuse" existing design in better technology & place *multiple cores* on same die
  - Combine with shared memory hierarchy

Scaling Today's uP Chips



Potential Multi-core Dies

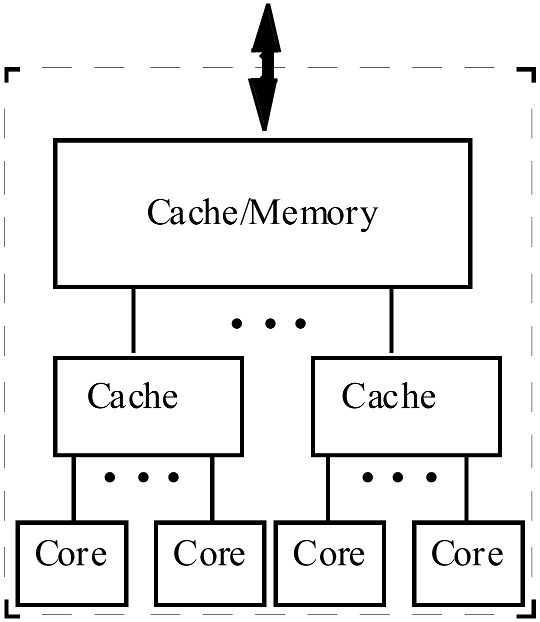


Assume we scale entire current single core chip & replicate to fill 280 sq mm die

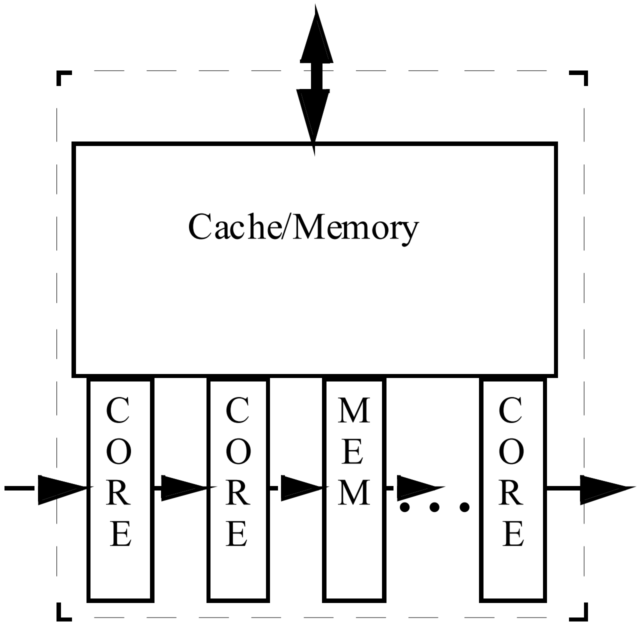
Examples of Multi-Core Designs

- 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
- Specialized chips
  - Network processors (up to 100s of cores)
  - Graphics & game processors
- Many multi-core designs also using multi-threaded cores

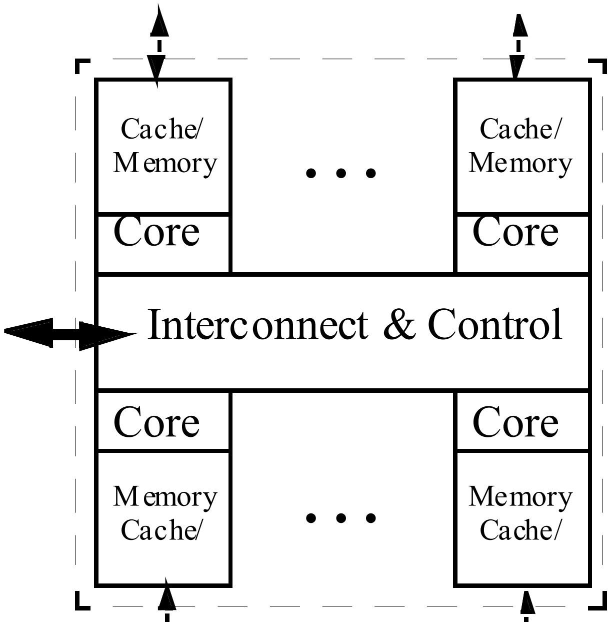
What is Today's Multi-Core Design Space



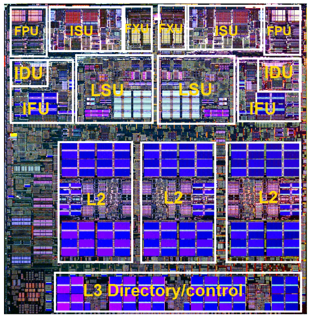
(a) Hierarchical Designs

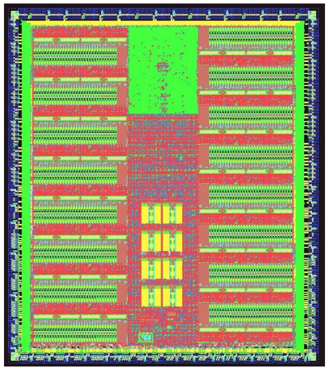


(b) Pipelined Designs



(c) Array Designs





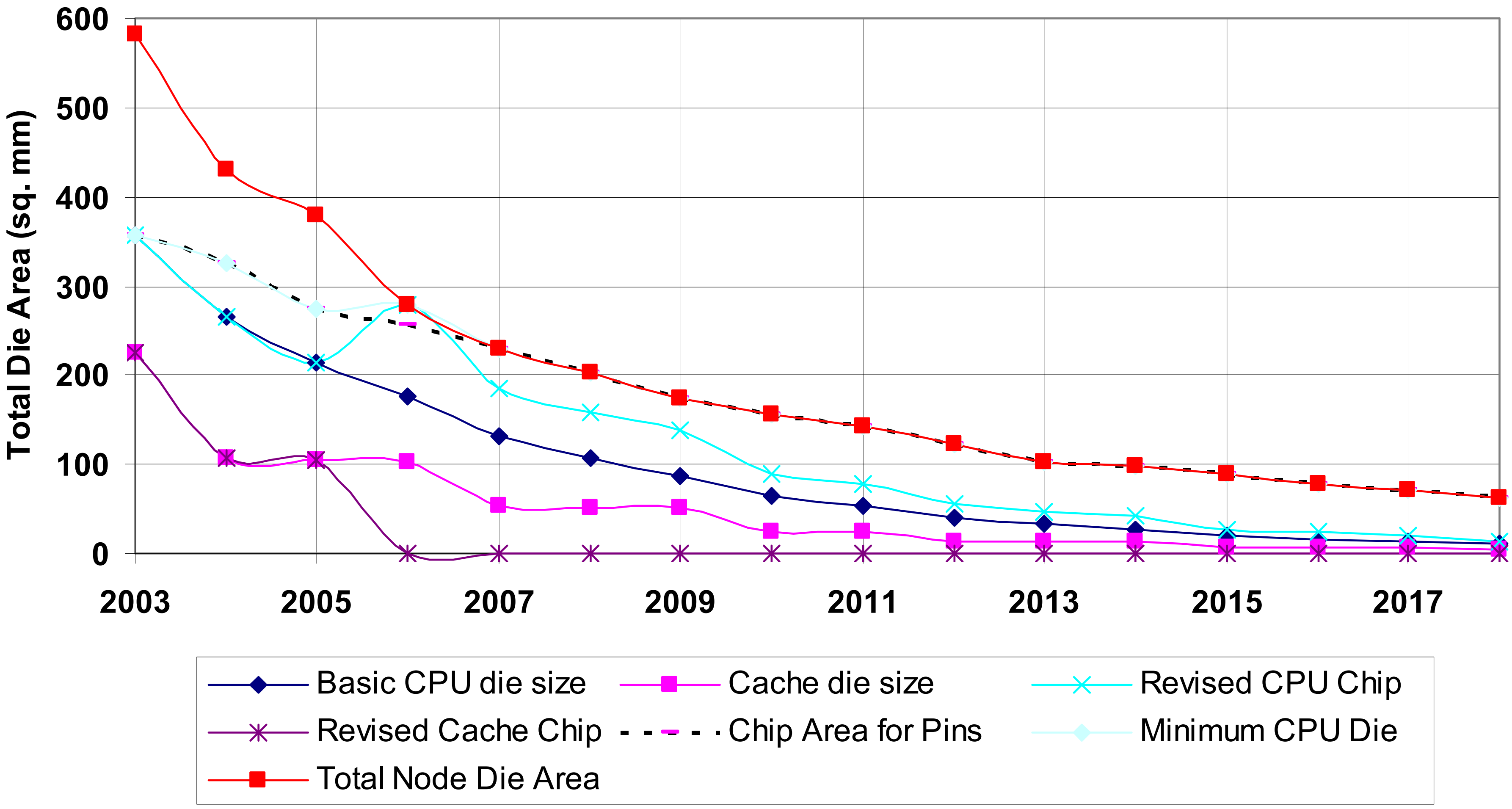
Sample Chips

	POWER5	X10q	Yukon
Year	2003	2003	2002
Technology	0.13 Logic	0.13 Logic	0.15 DRAM
0.18 Logic			
Area	389mm2	??	??
Type	Hierarchical	Pipelined	Array
Transistors	276M	114M/62L	??
Cores	2@19% each	200=68%	256=14%
Arch	MT-SMP	Systolic	2D SIMD
Core Clock	2GHz	200 MHz	200MHz
L2/Memory	1.9MB=27%	23%	16MB=41%
Contacts	5,400	1,280	
Memory	41%	23%	44%
Signal I/Os	2,313	845	
# Ports		10	
Data B/W	16GB/s	40Gbps	200MB/s
Internal BW			25.6GB

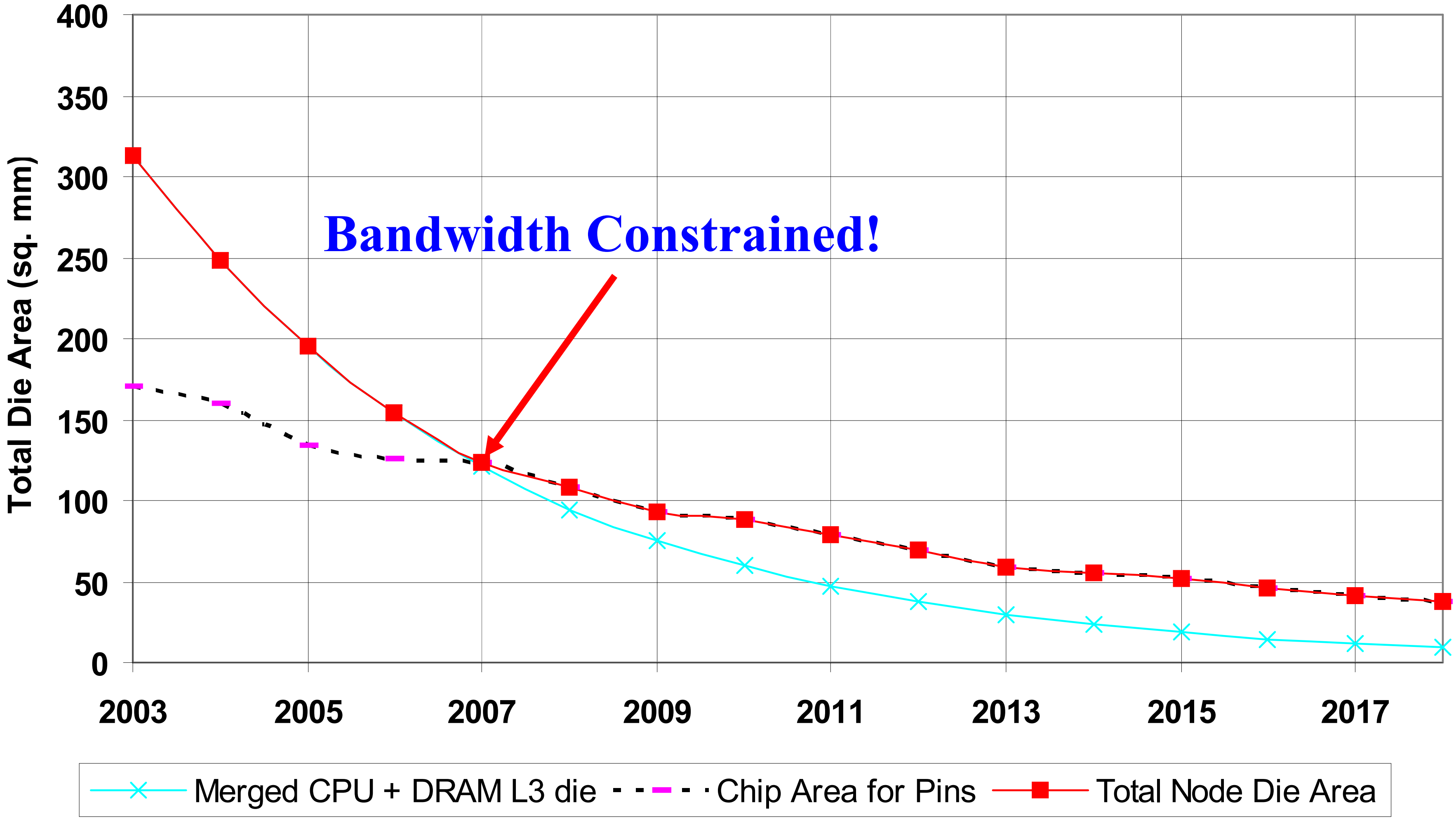
Multi-Core Projection Models

- Shrink: Take today & just shrink
- Shrink & Merge: replace L2/L3 SRAM with DRAM (& reduce clock)
- Constant die size: Add cores to fill die
- Single chip type: merge with memory
  - Ensure desired memory/performance ratio
- Consider for each model:
  - How many pins needed for constant bandwidth ratio

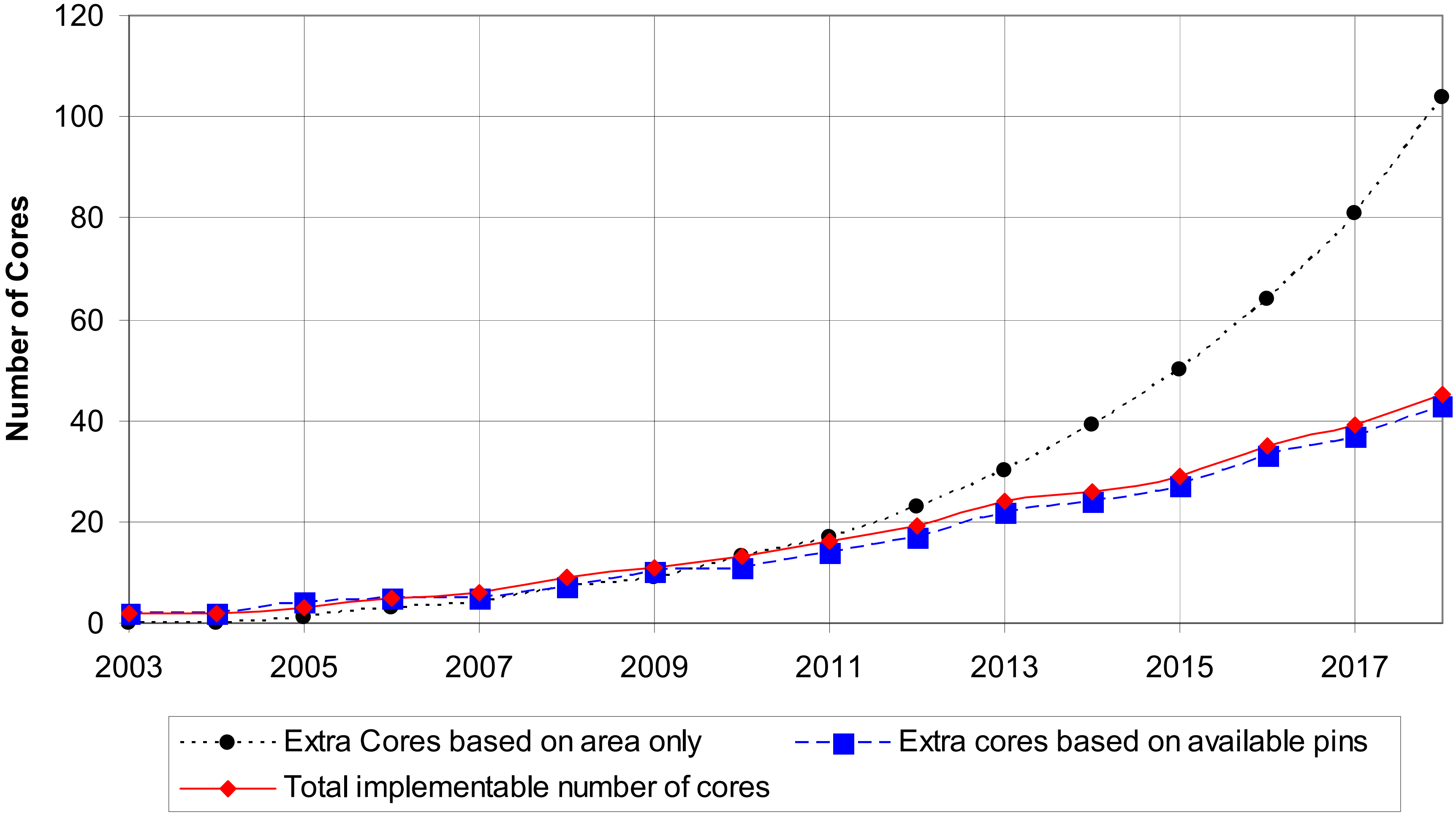
Shrink Model



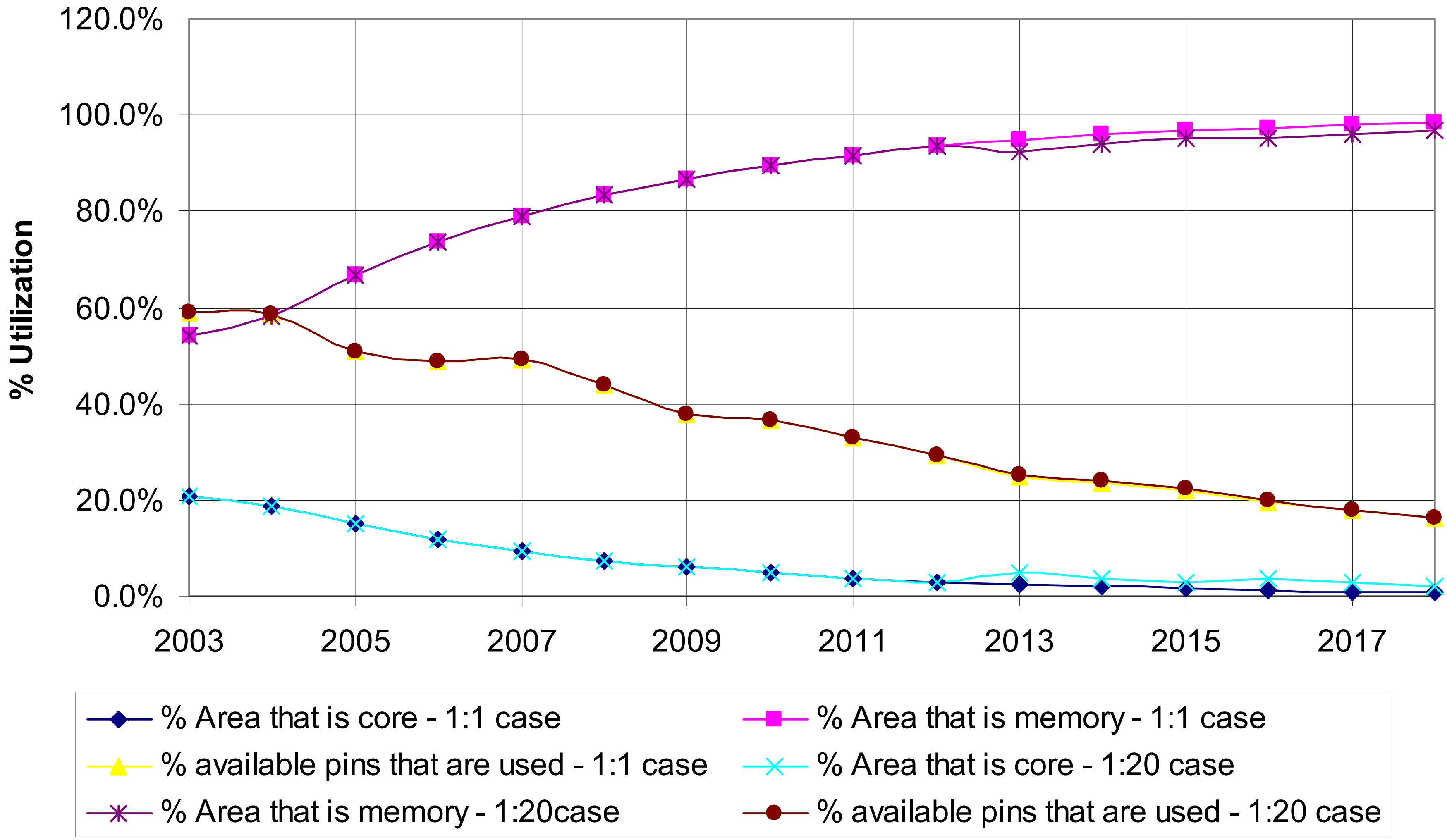
Shrink & Merge Model



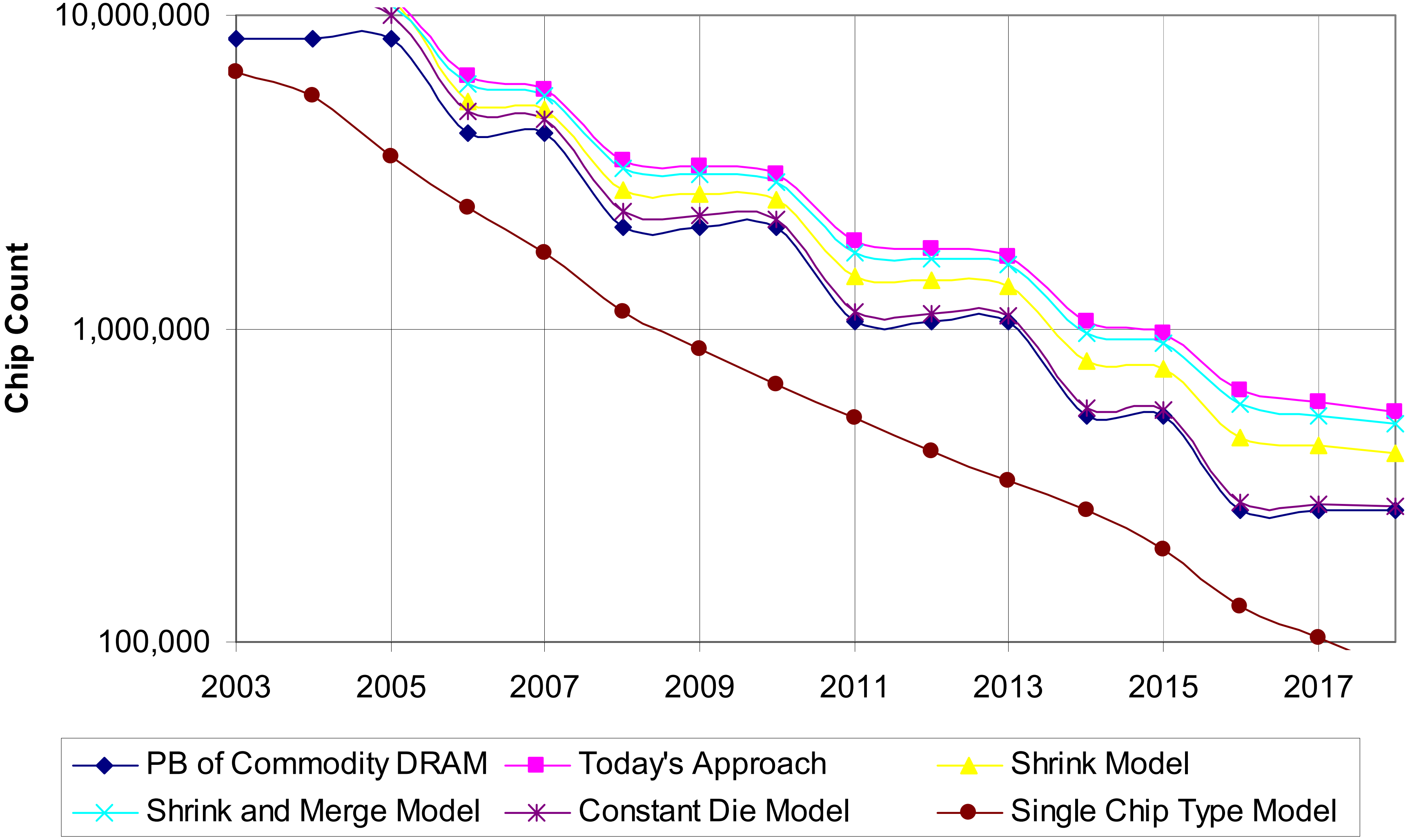
Constant Die Model



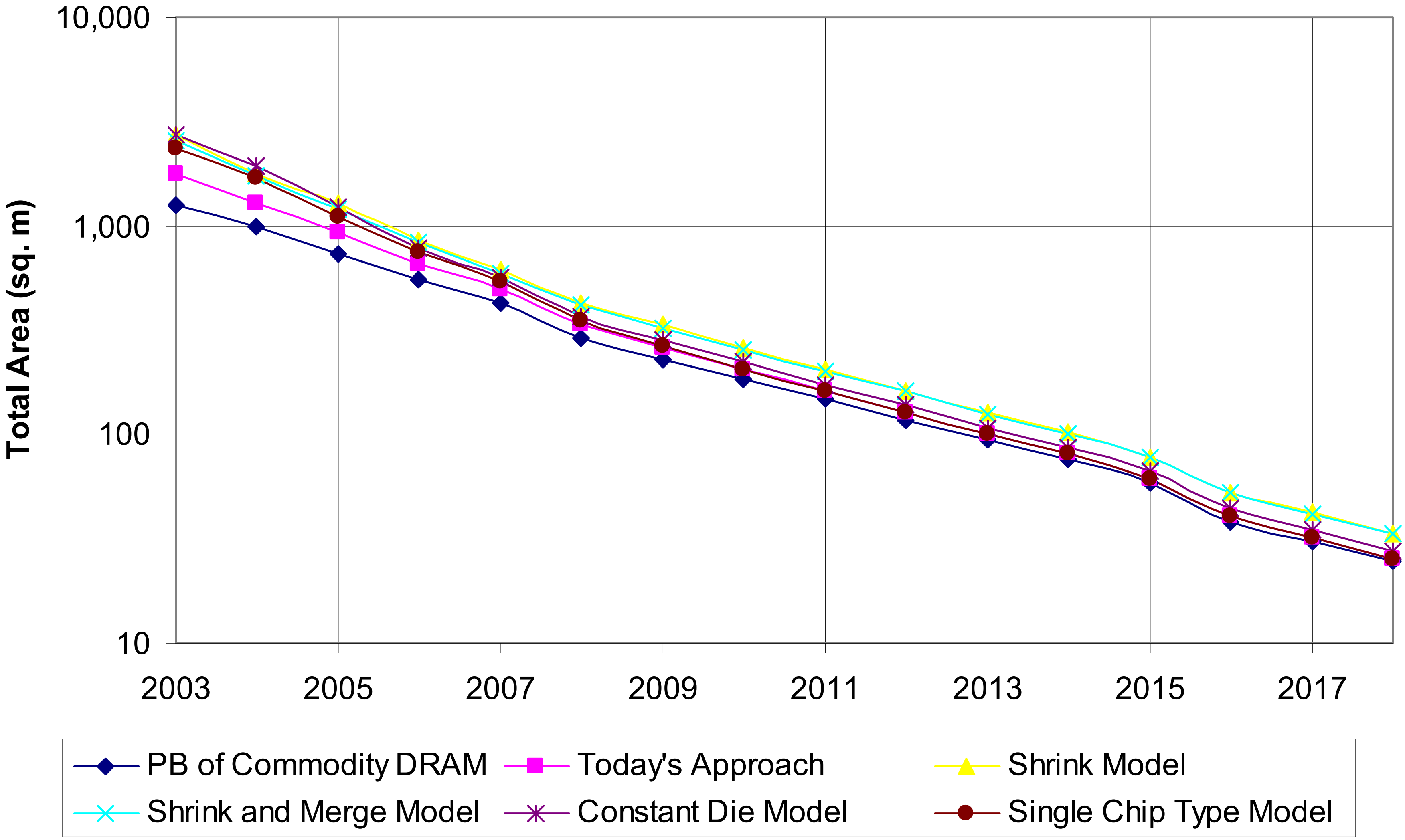
Single Chip Type Model (With Constant Die Size)



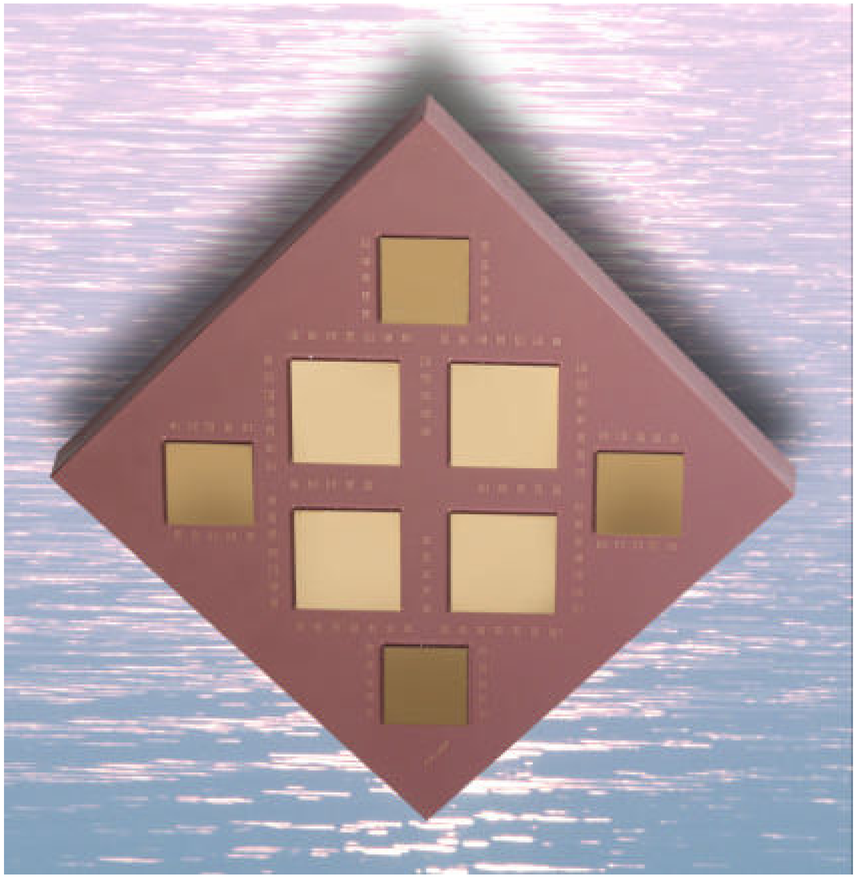
Chip Count for a Petabyte System



Silicon Area for a Petabyte System



Silicon Alone is not the Complete Story



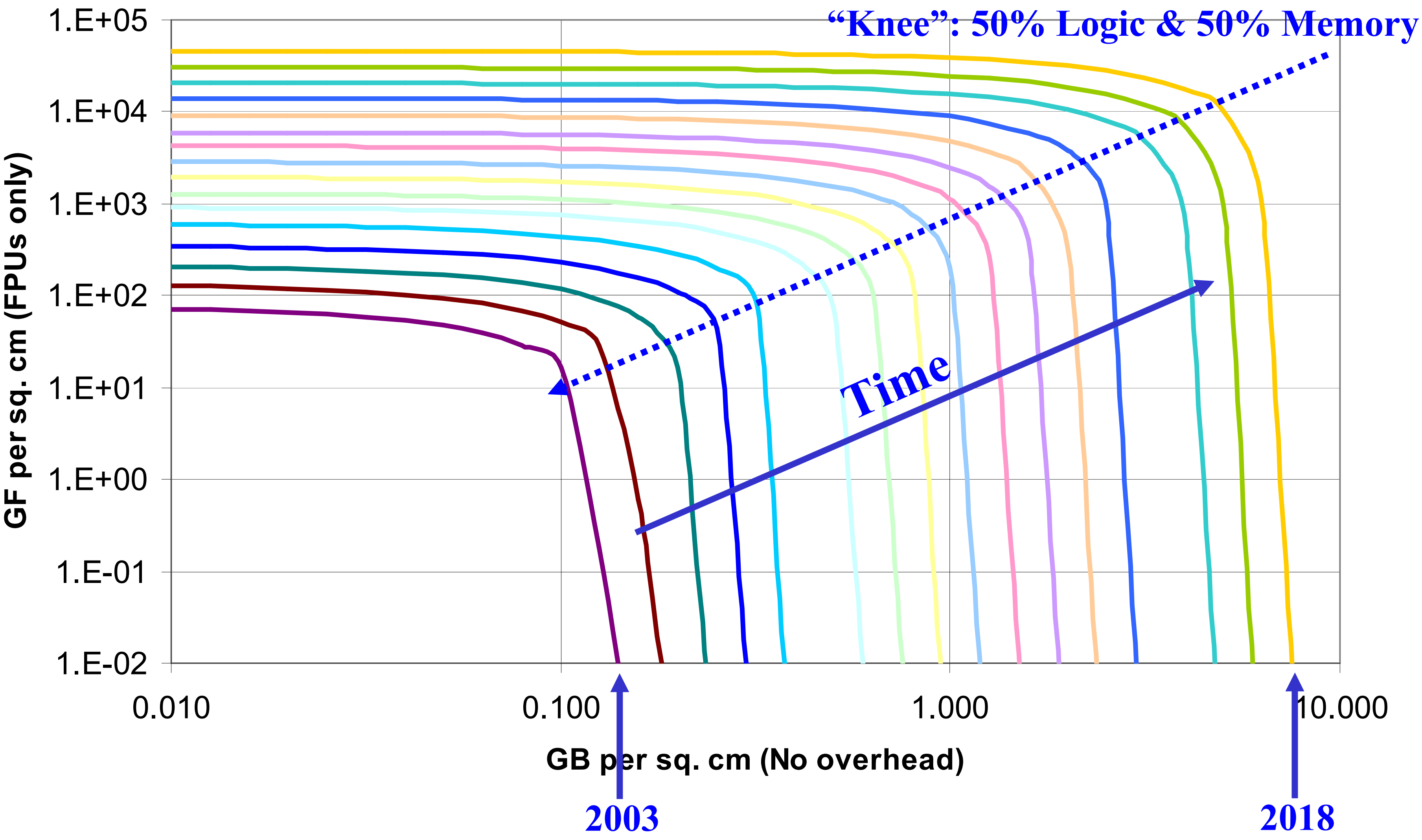
- Only 20% of MCM is silicon
- And we haven’t accounted for the heat sink!

Observations

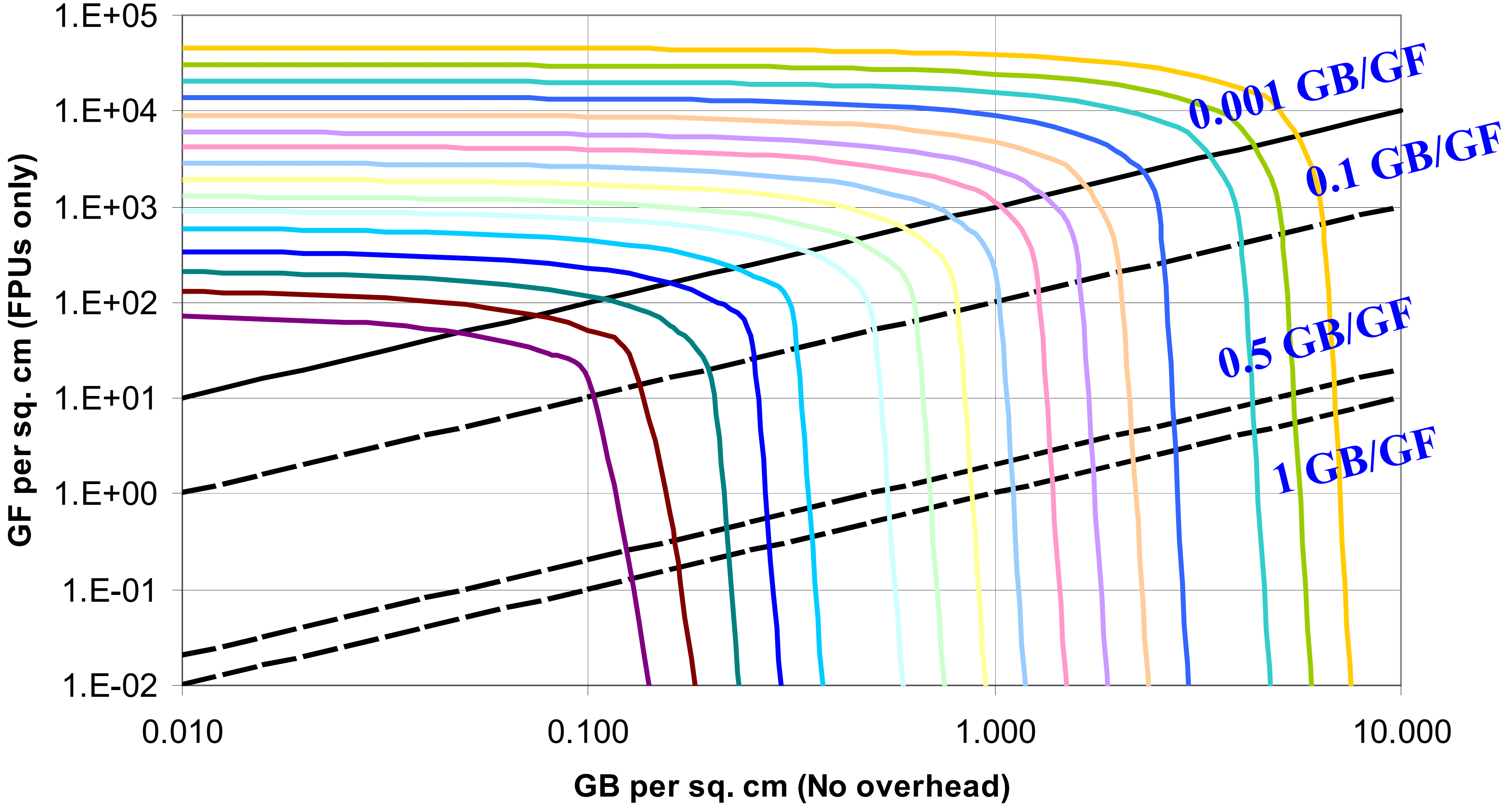
- Silicon growing irregularly in
  - Memory density per square cm
  - Performance possible per square cm
  - Off-chip I/O bandwidth per square cm
- 99% of today’s logic chips
  - Do no computation
  - And are mostly memory
- And we pay a ***huge*** overhead when
  - Densest memory technology not used
  - 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

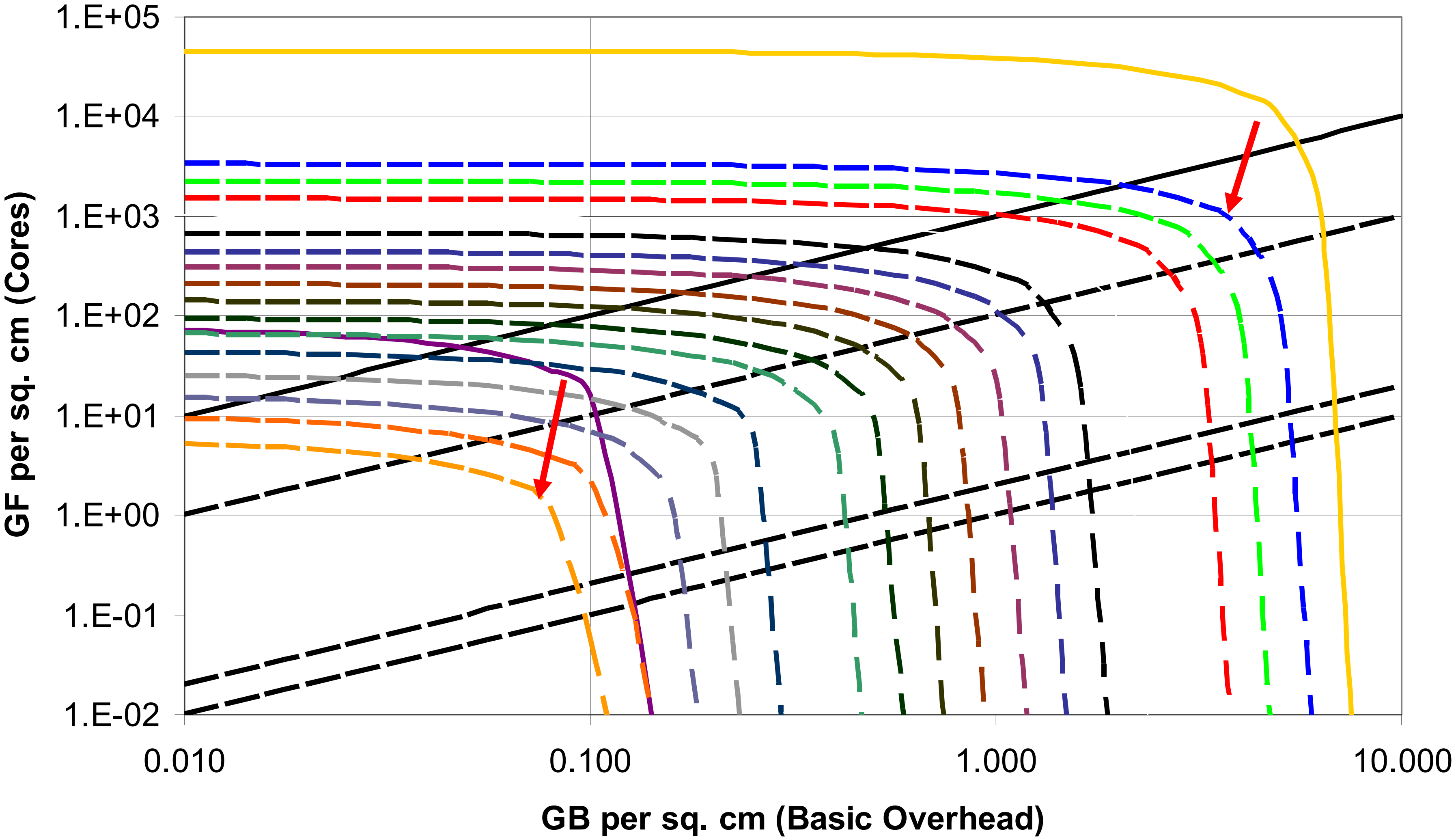
How can we use a sq. cm? (with no overhead)



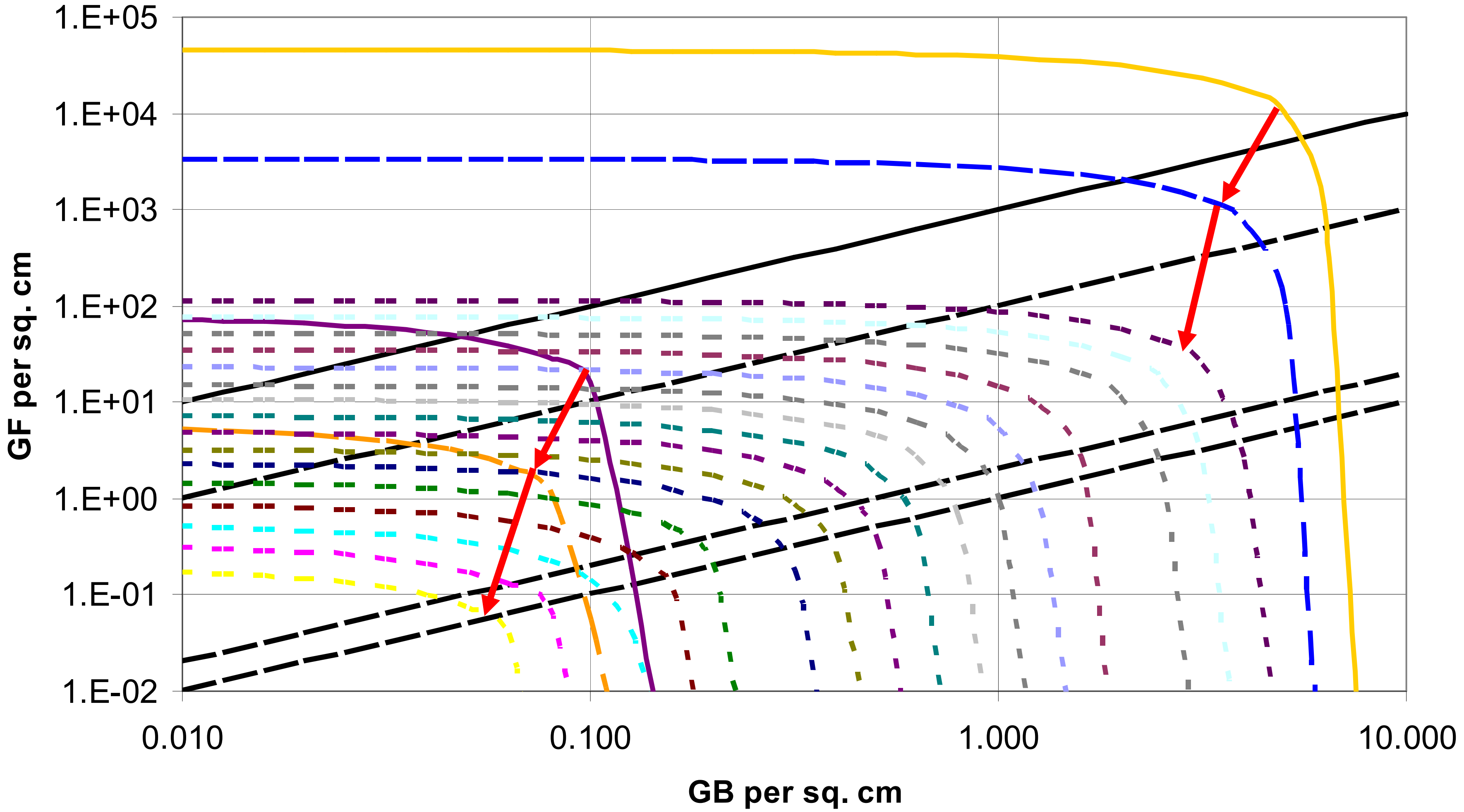
Adding In "Lines of Constant Performance"



Knee Curves with Basic Overheads

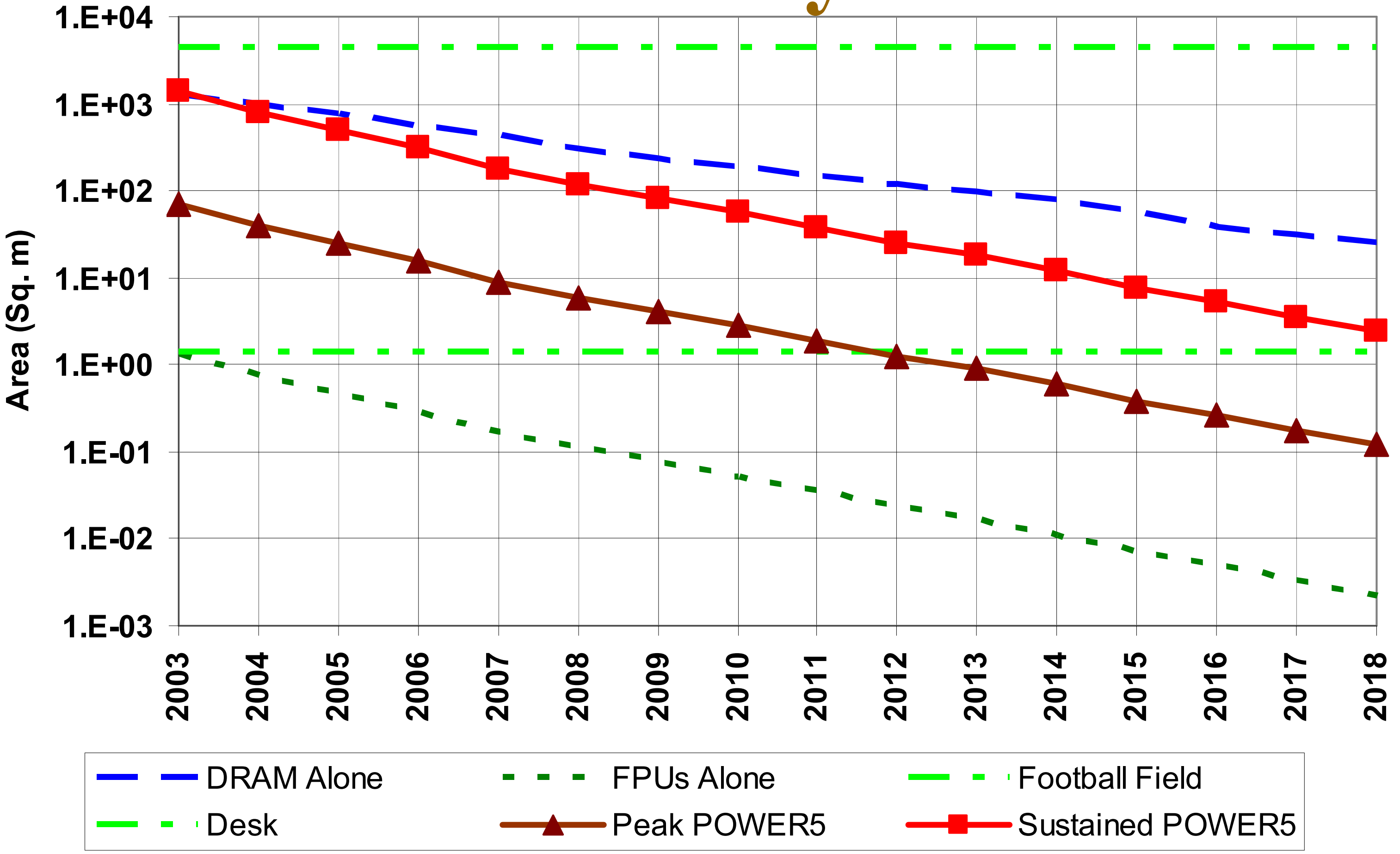


Knee Curves with Today's Overheads



Partitioning chips as we do today is hugely inefficient

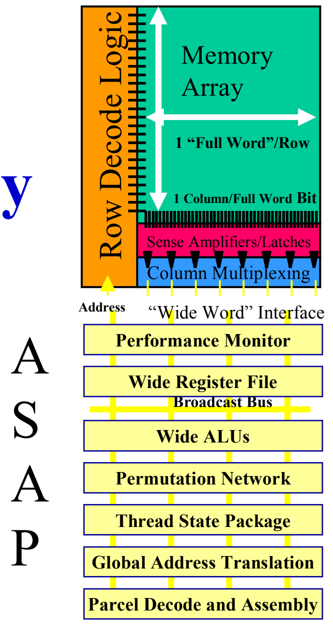
Minimal Size for a "Peta" System



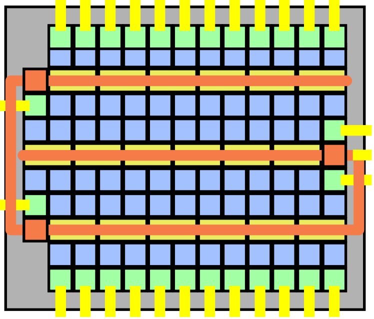
- **In terms of silicon area: "*It's the memory!*"**
- **We extract little benefit from most of our high cost logic**

“Processing-In-Memory”

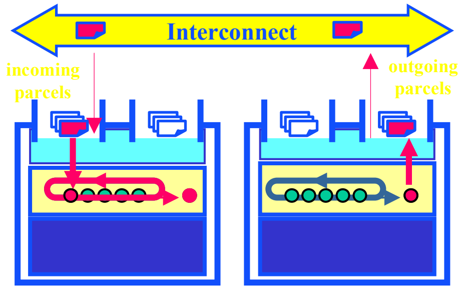
- High density memory on same chip with reasonable dense logic
- Very fast access from logic to memory
- Very high bandwidth
- ISA/microarchitecture designed to utilize high bandwidth
- Tile with “memory+logic” nodes



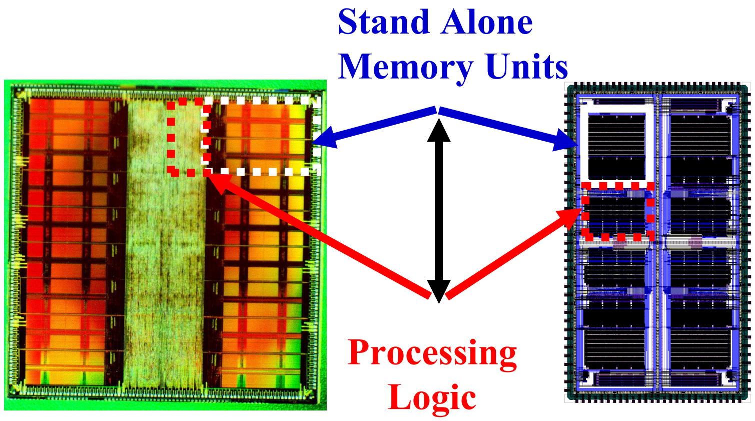
A Memory/Logic *Node*



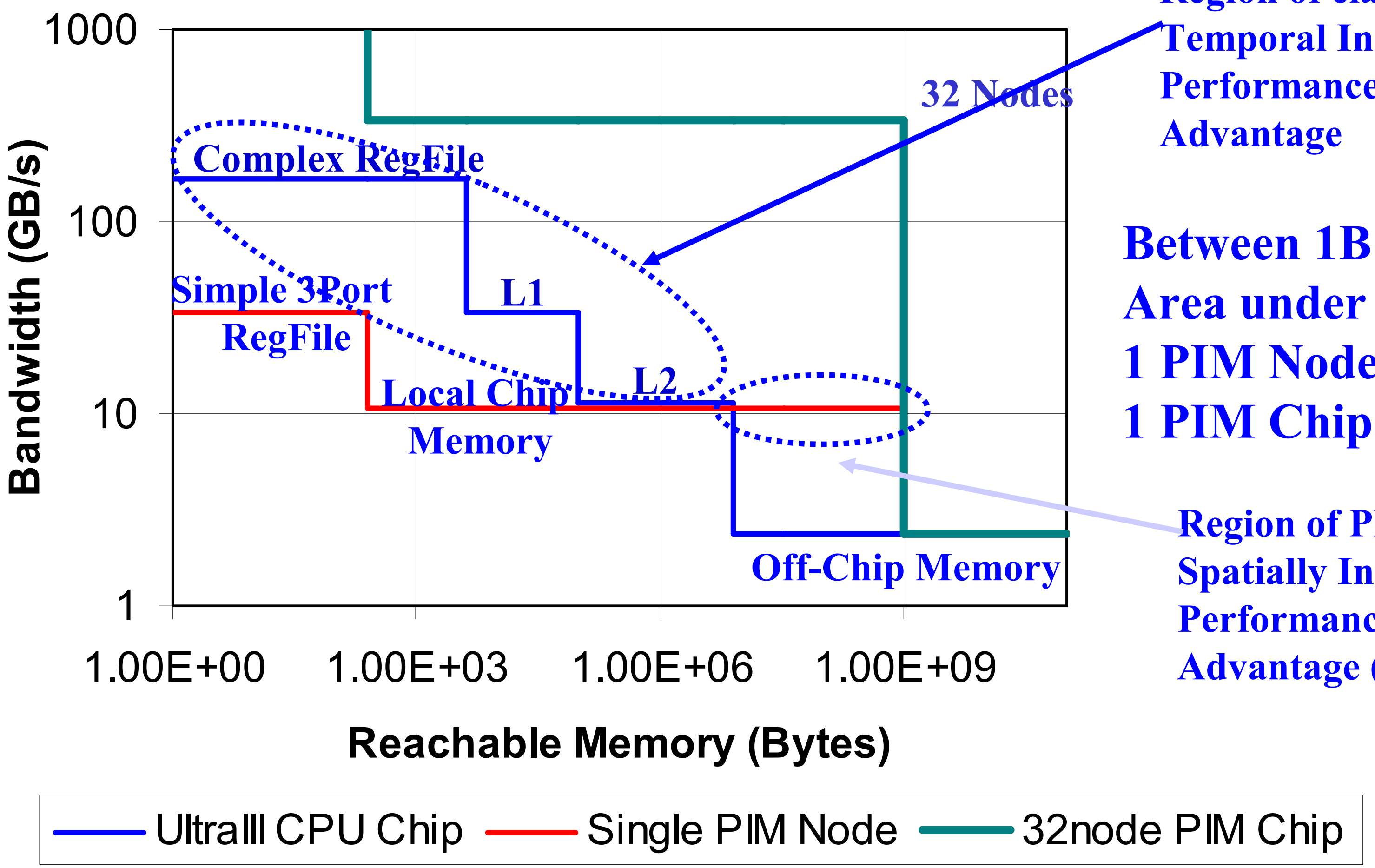
Tiling a Chip



Parcel = Object Address + Method_name + Parameters



The PIM “Bandwidth Bump”

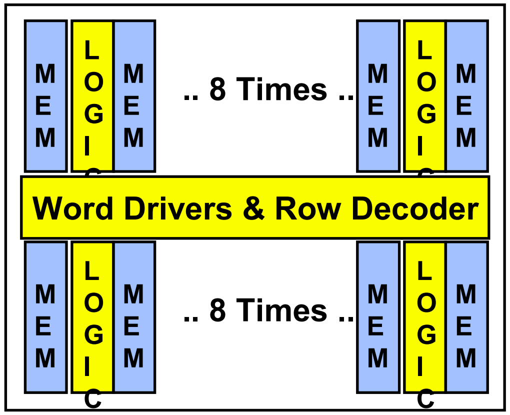


Region of classical
Temporal Intensive
Performance
Advantage

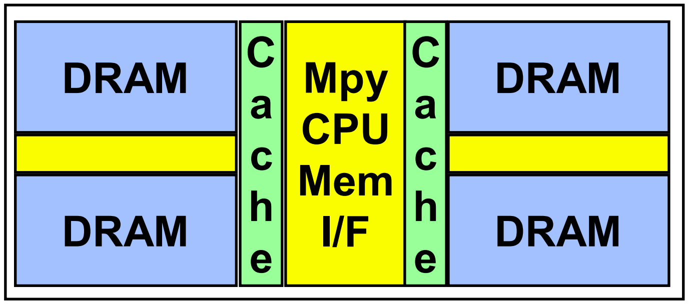
Between 1B & 1 GB,
Area under curve:
1 PIM Node = 4.3xUIII
1 PIM Chip = 137xUIII

Region of PIM
Spatially Intensive
Performance
Advantage (1 Node)

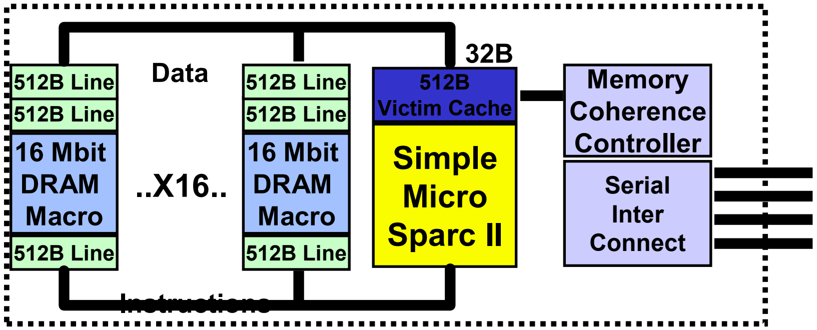
PIM Chip MicroArchitectural Spectrum



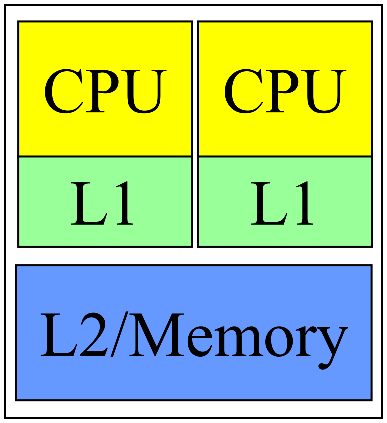
SIMD: Linden DAAM



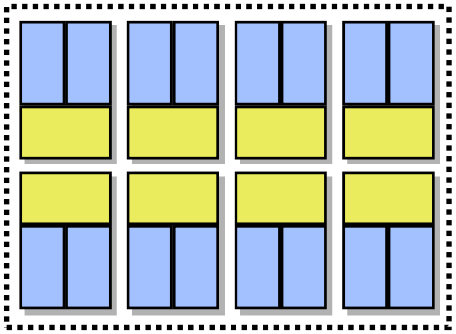
Single Chip Computer:
Mitsubishi M32R/D



Complete SMP Node:
Proposed SUN part



Chip Level SMP:
POWER4, BG/L

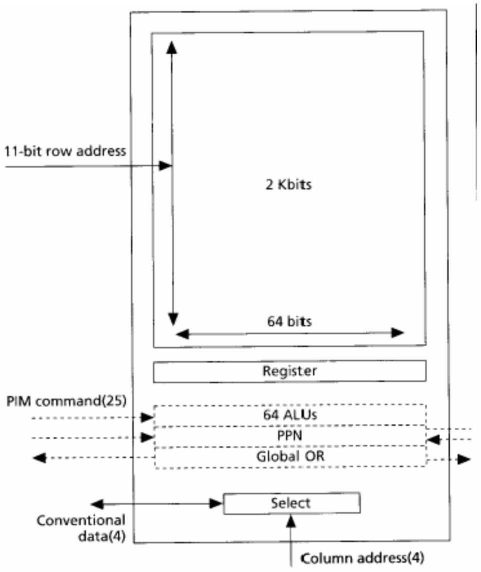


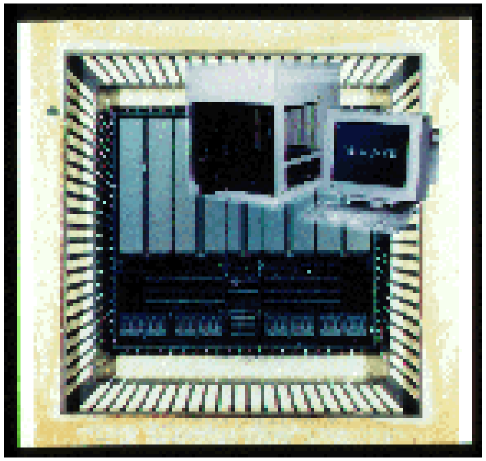
Tiled & Scalable:
BLUE GENE,
EXECUBE

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)

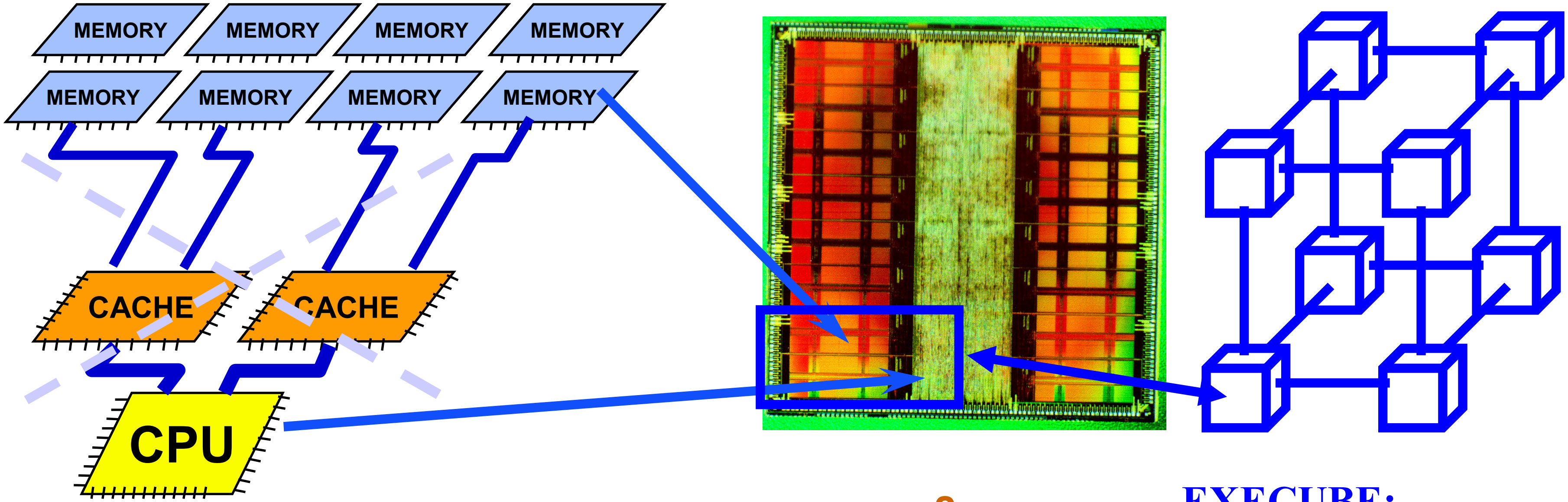




- Memory part for CRAY-3
- “Looked like” SRAM memory
  - With extra command port
- •128K SRAM bits (2k x 64)
- 64 1 bit ALUs
- SIMD ISA
- Fabbed by National
- Also built into workstation with 64K processors
  - 5-48X Y-MP on 9 NSA benchmarks

EXECUBE: An Early MIMD PIM (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**

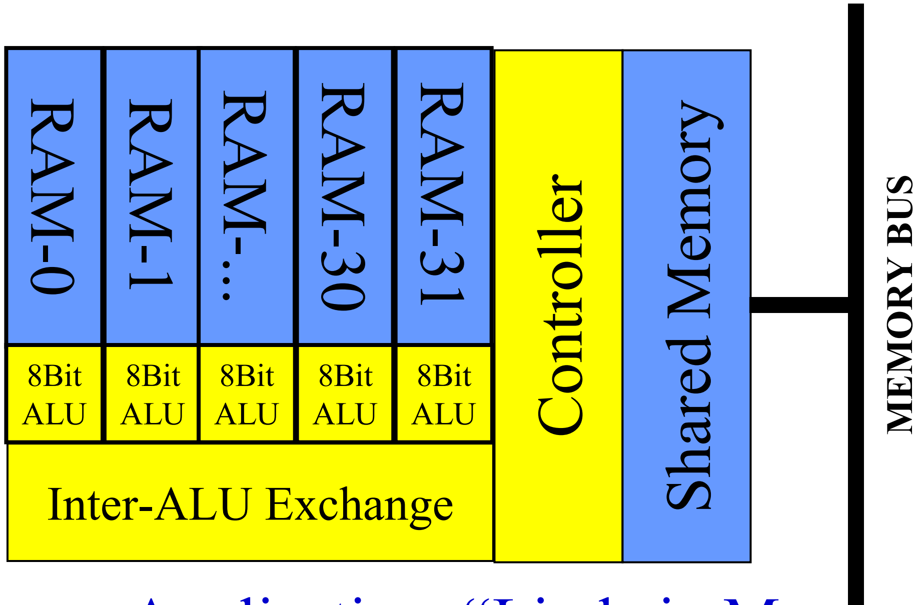


Include "High Bandwidth" Features in ISA

8 Compute Nodes on ONE Chip

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

RTAIS: The First ASAP (circa 1993)



RAM-0 | RAM-1 | RAM-... | RAM-30 | RAM-31

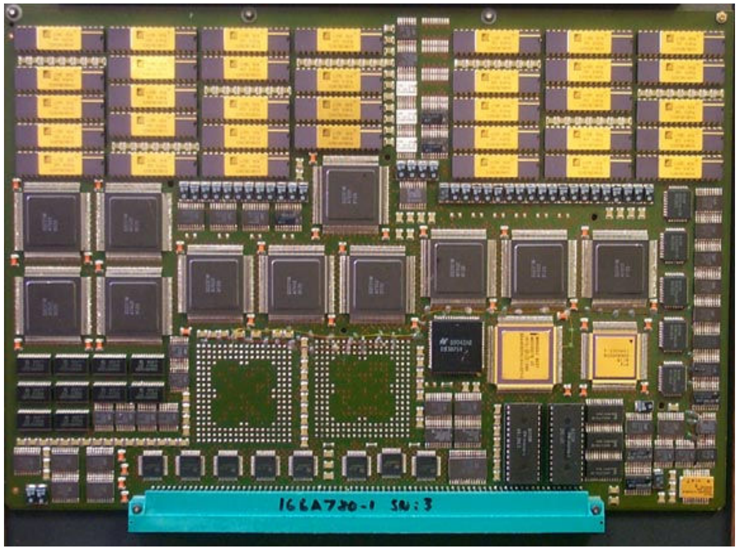
8Bit ALU | 8Bit ALU | 8Bit ALU | 8Bit ALU | 8Bit ALU

Inter-ALU Exchange

Controller

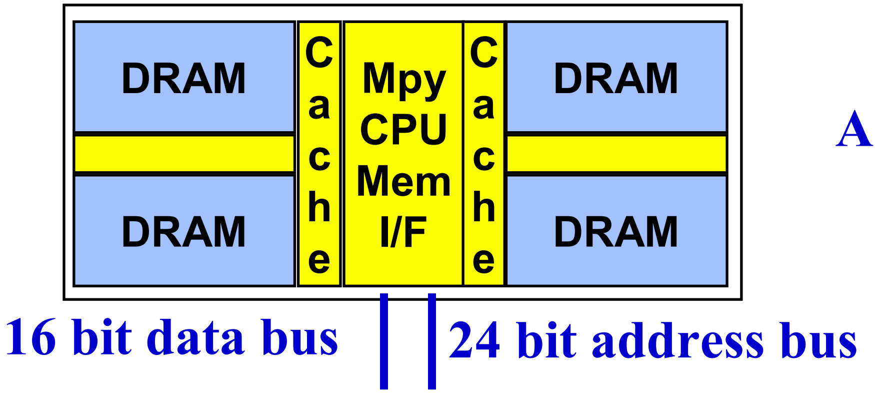
Shared Memory

MEMORY BUS

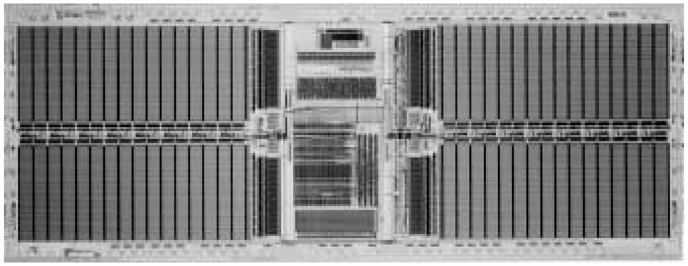


- Application: “Linda in Memory”
- Designed from onset to perform wide ops “at the sense amps”
- More than SIMD: flexible mix of VLIW
- “Object oriented” multi-threaded memory interface
- Result: 1 card 60X faster than state-of-art R3000 card

Mitsubishi M32R/D

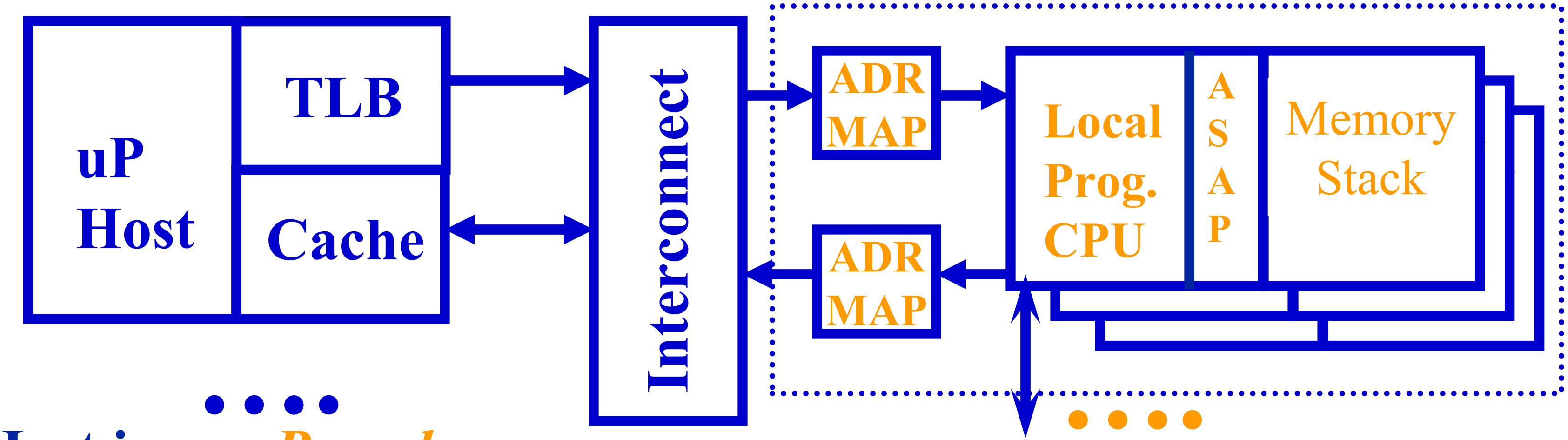


Also two 1-bit I/Os



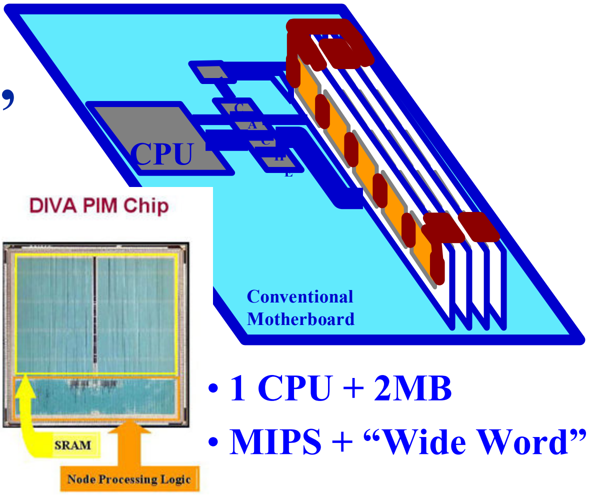
- 32-bit fixed point CPU + 2 MB DRAM
- “Memory-like” Interface
- Utilize wide word I/F from DRAM macro for cache line

DIVA: Smart DIMMs for Irregular Data Structures



Host issues *Parcels*

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



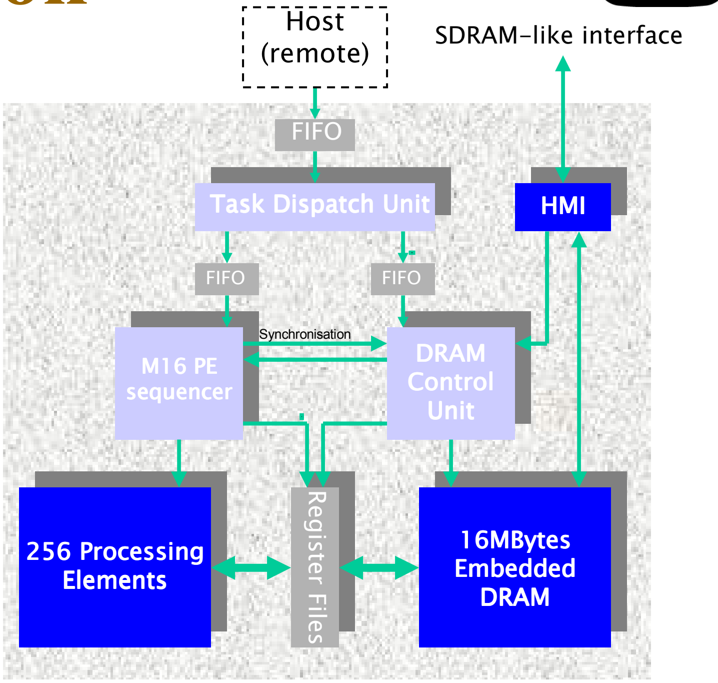
- 1 CPU + 2MB
- MIPS + "Wide Word"

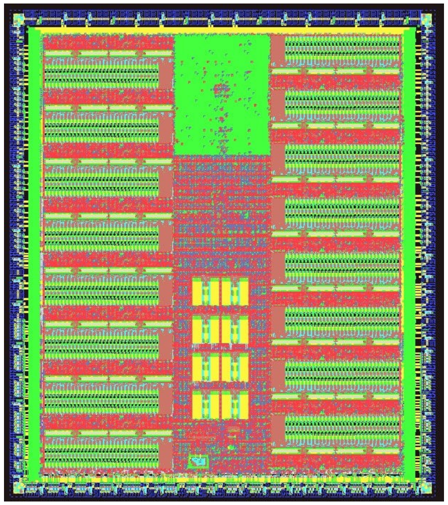
DIVA Functions:

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

Micron Yukon

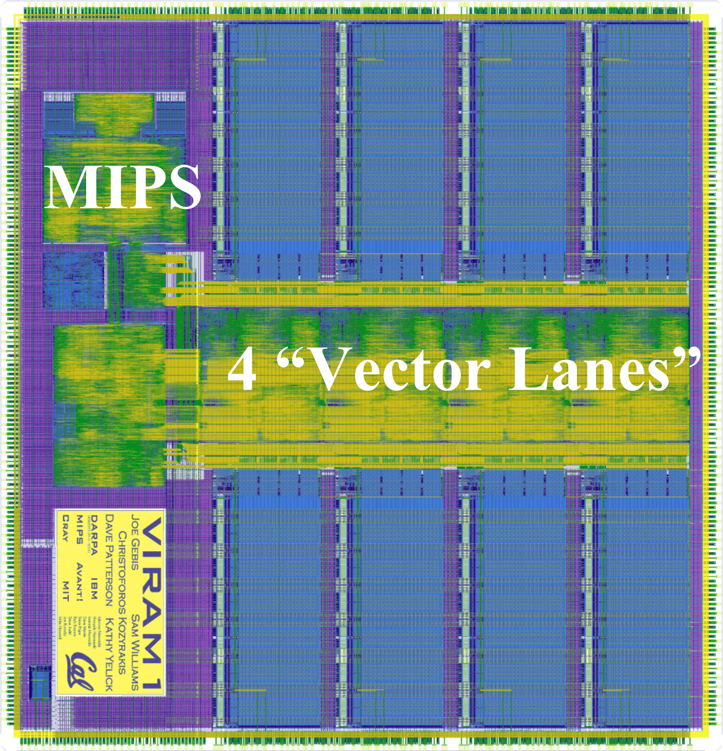
- 0.15µm eDRAM/ 0.18µm logic process
- 128Mbits DRAM
  - 2048 data bits per access
- 256 8-bit integer processors
  - Configurable in multiple topologies
- On-chip programmable controller
- Operates like an SDRAM



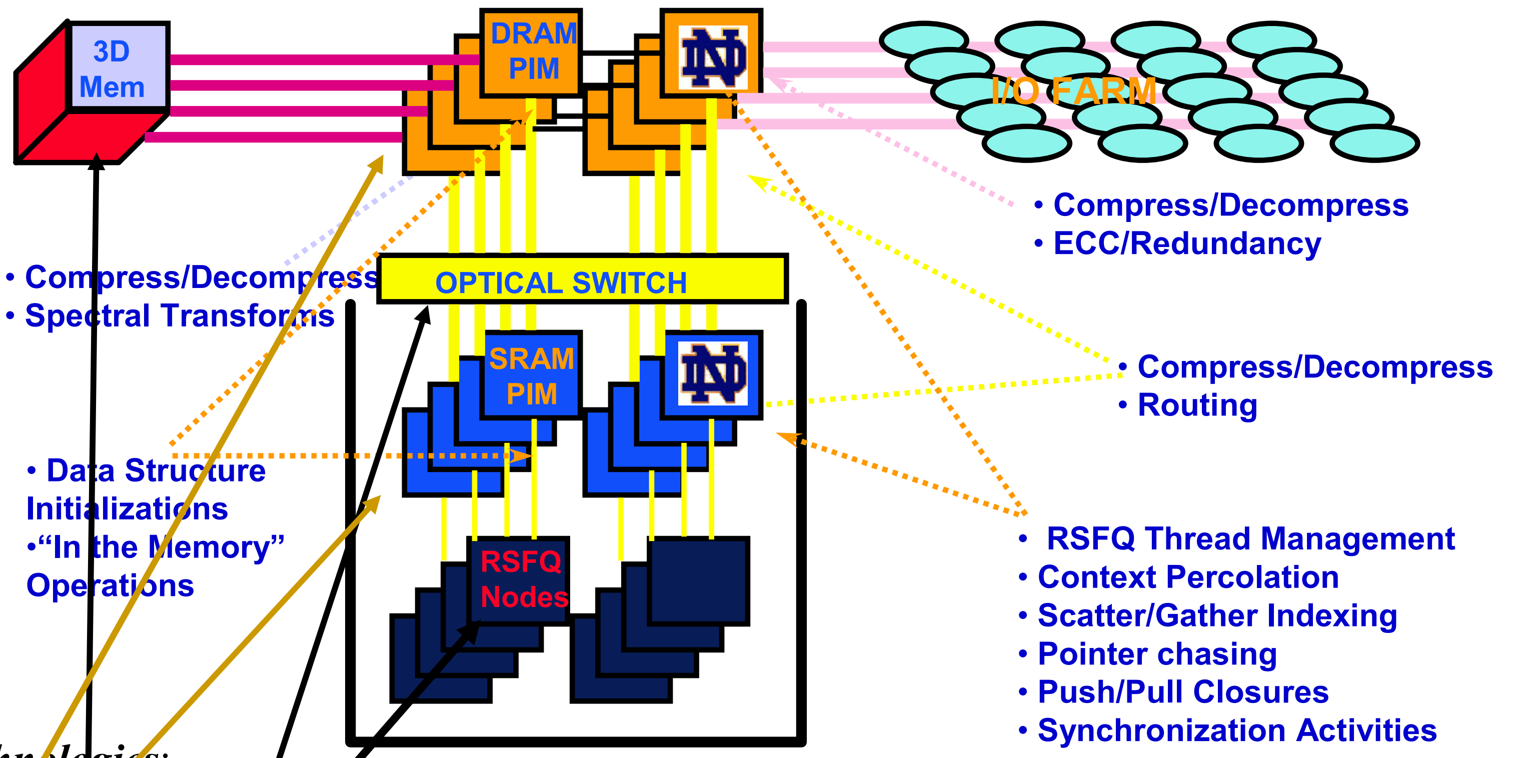


Berkeley VIRAM

- 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



The HTMT Architecture & PIM Functions



- Compress/Decompress
- ECC/Redundancy

- Compress/Decompress
- Spectral Transforms

- Compress/Decompress
- Routing

- Data Structure Initializations
- “In the Memory” Operations

- RSFQ Thread Management
- Context Percolation
- Scatter/Gather Indexing
- Pointer chasing
- Push/Pull Closures
- Synchronization Activities

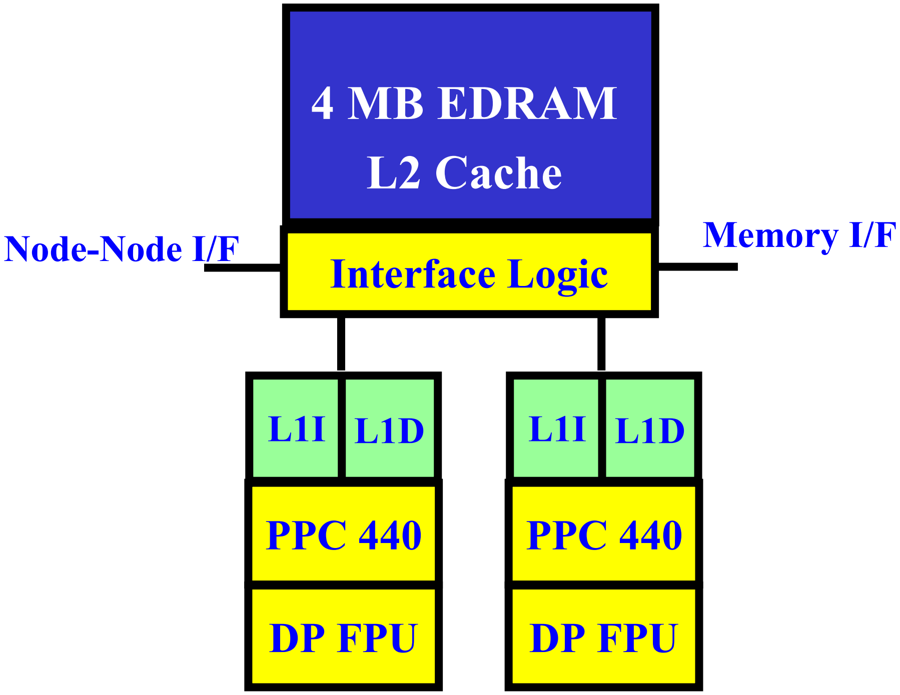
New Technologies:

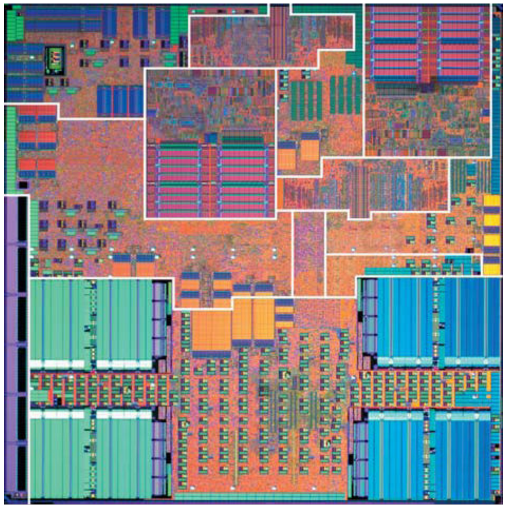
- Rapid Single Flux Quantum (RSFQ) devices for 100 GHz CPU nodes
- WDM all optical network for petabit/sec bi-section bandwidth
- Holographic 3D crystals for Petabytes of on-line RAM
- PIM ND for active memories to manage latency

PIMs in Charge

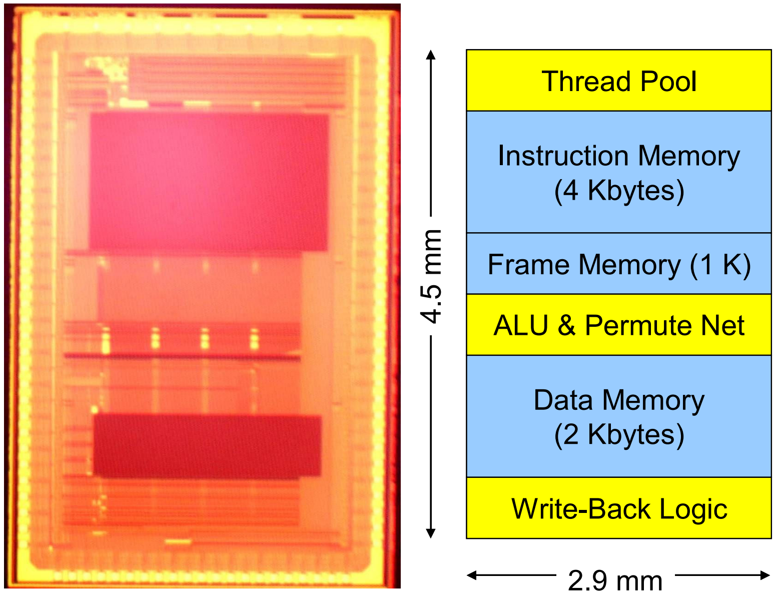
Bluegene/L

- Two relatively simple cores with dense embedded DRAM techology
- Designed to scale simply to bigger systems
- Basis for world’s fastest machine

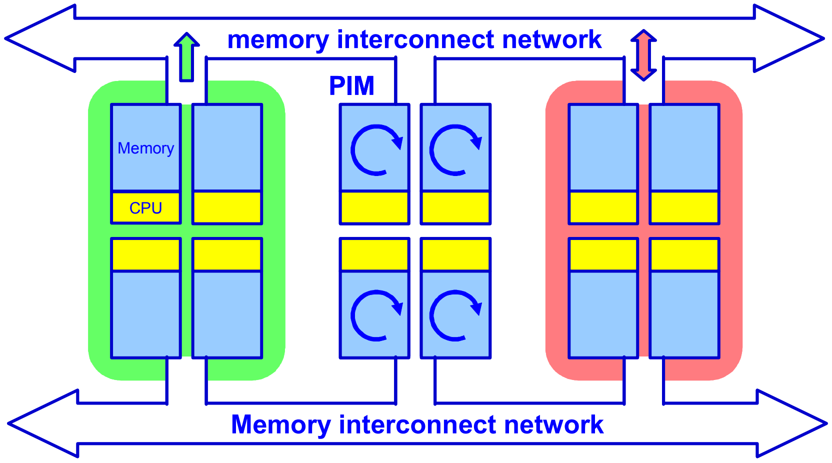


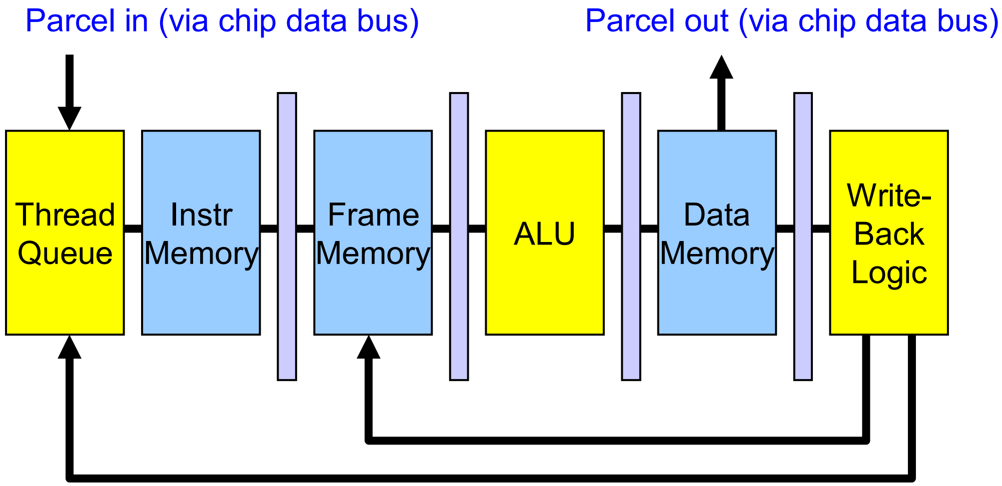


PIM Lite



- “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 1st pass success
- 3.2 million transistors (4-node)



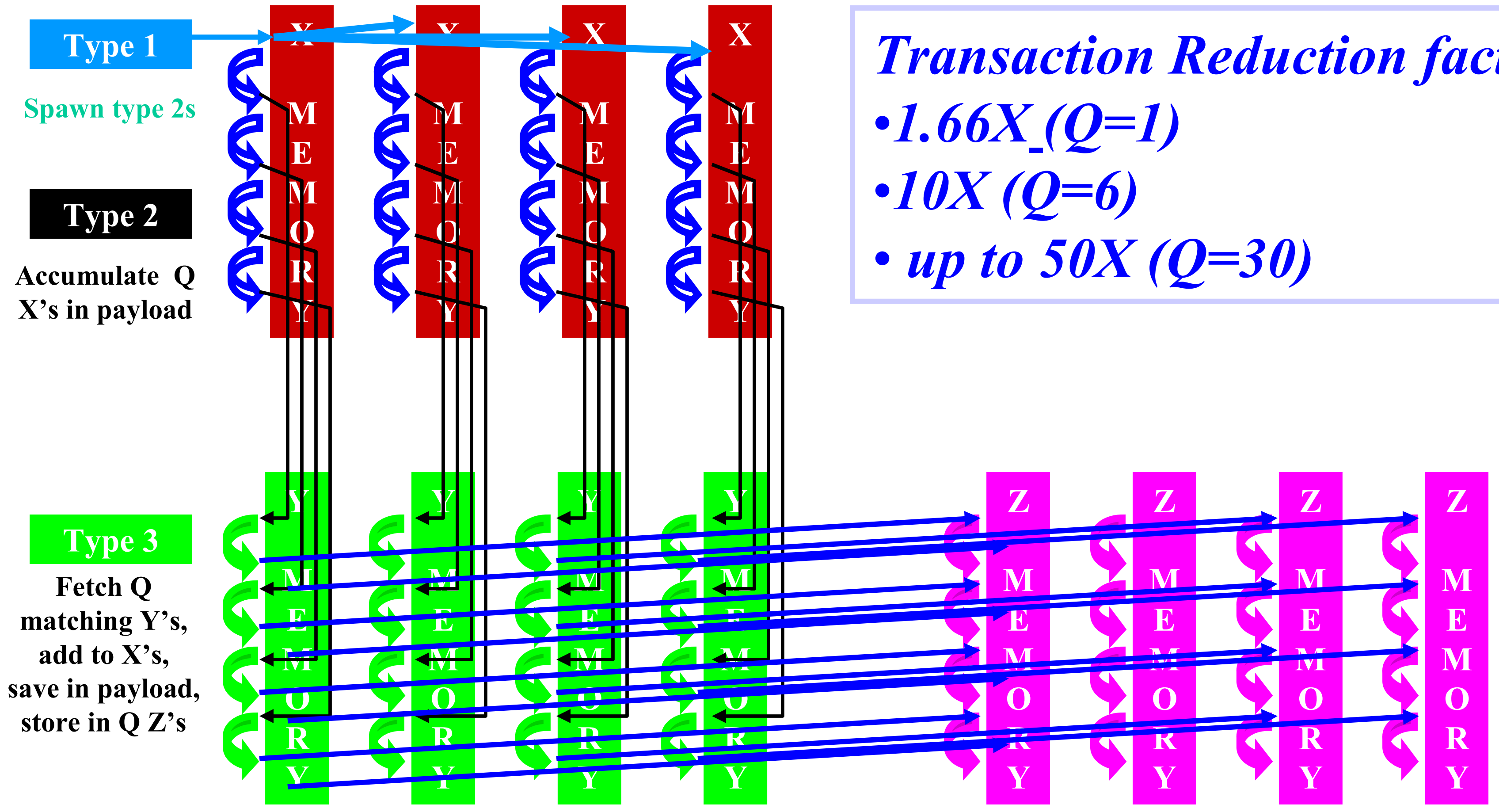


One Step Further: Allowing the Threads to Travel

- "Overprovision" memory with huge numbers of anonymous processors
  - Like PIM Lite, 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

Vector Add via Traveling Threadlets



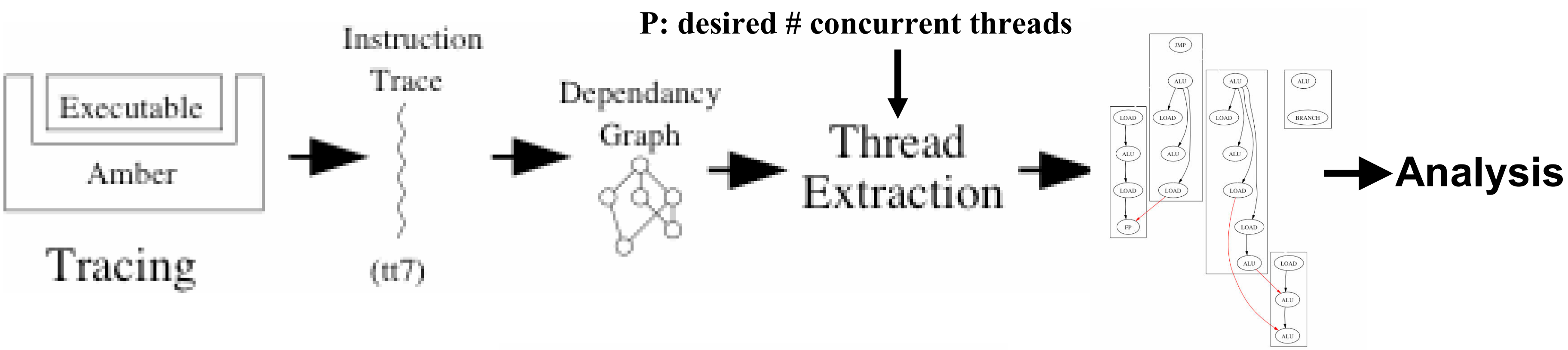
Transaction Reduction factor:

- *•1.66X (Q=1)*
- *•10X (Q=6)*
- *• up to 50X (Q=30)*

Stride thru Q elements

Trace-Based “Threadlet” Extraction & Simulation

- **Applied to large-scale Sandia applications over summer 2003**



P: desired # concurrent threads

Executable

Amber

Tracing

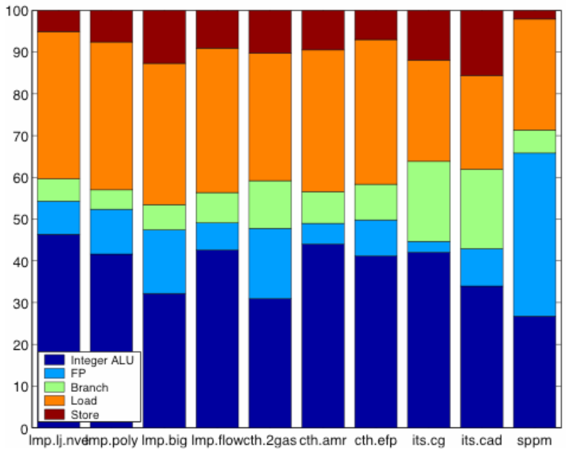
Instruction Trace

(tt7)

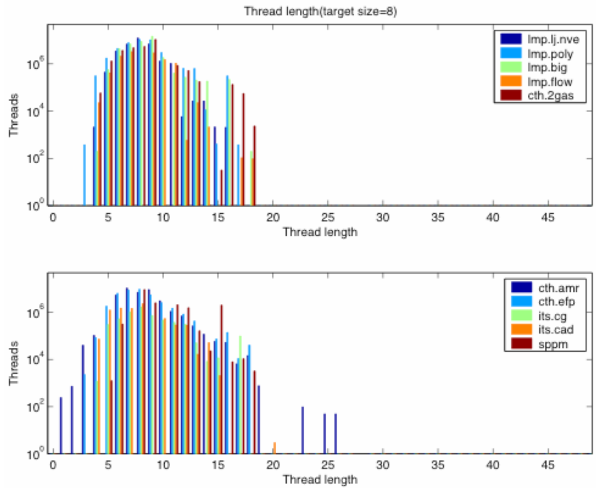
Dependancy Graph

Thread Extraction

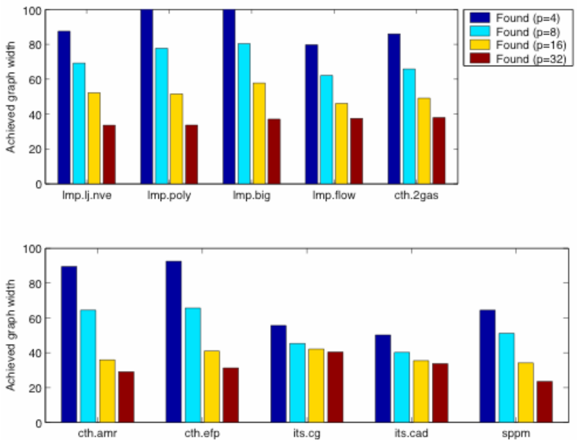
Analysis



From Basic Application Data

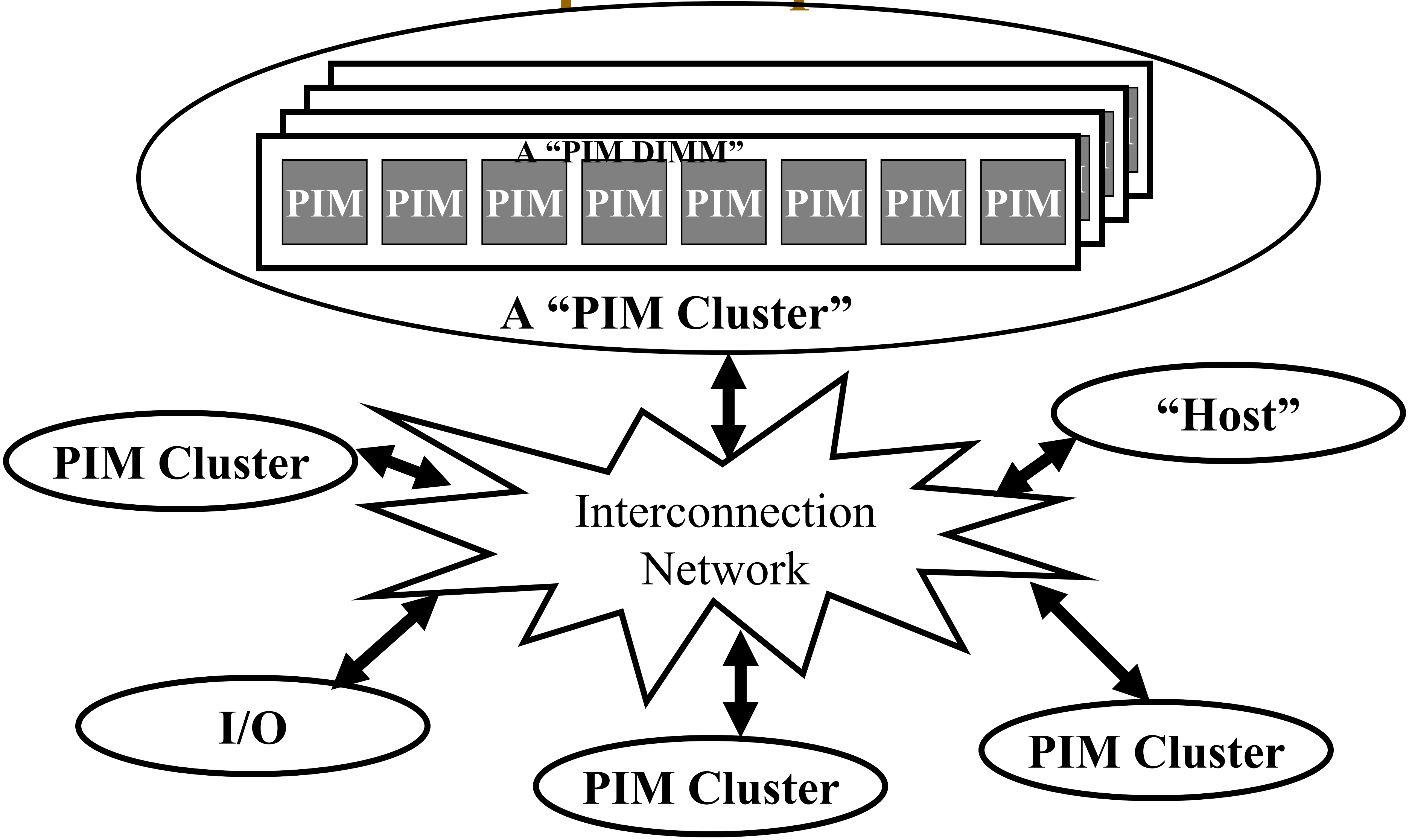


Through Detailed Thread Characteristics



To Overall Concurrency

Next: An "All-PIM" Supercomputer



Summary

Summary

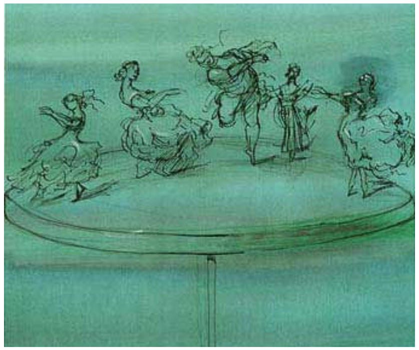
- When it comes to silicon: *It's the Memory, Stupid!*
- State bloat consumes huge amounts of silicon
  - That does no useful work!
  - And all due to focus on "named" processing logic
- Technology scaling progressing at uneven rates
  - Clocks slowing
  - Power limiting logic gate density
  - Off-chip 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

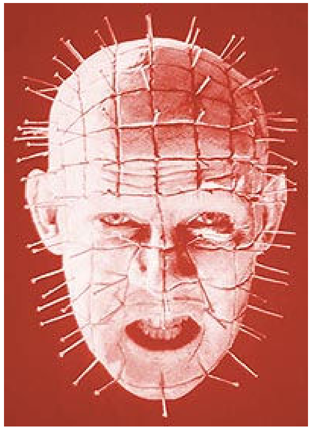
How Do We Make It Better?

- Focus on "cheap" logic in dense memory fab process
  - Don't fret the clock rate
- Reduce thread state
  - Cost of moving/copying state = line reference
- Simplify cores and "overprovision"
  - "Pitch-match" to memory macro
- Relentless multi-threading execution models
- 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

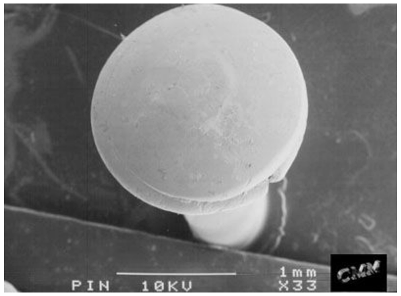
A Question from Salishan:

How many Cores can Fit on the Head of A Pin?

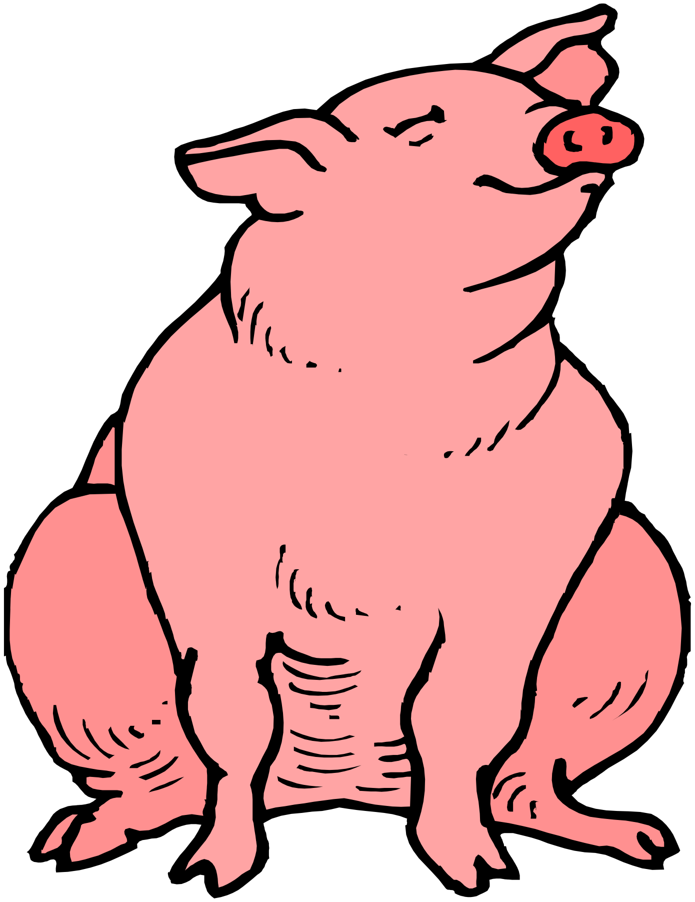




- Area of a pin = .015 sq. cm.
- Assume Darkhorse 8051 @ 7 KT
- 2018: 4200 cores, @ 53 GHz
  - = approx 20 TOPS
- But to make them dance we need memory
- At 50/50 Memory & Logic
  - 2100 Cores + 100MB
- New Term: ***PIMHEAD***



The Future



Will We Design Like This?

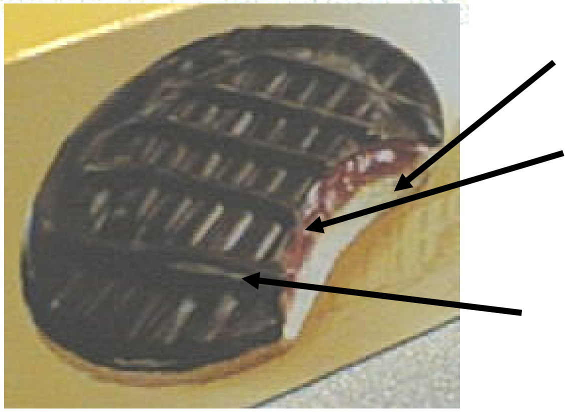
Or This?

Regardless of Technology!

PIMs Now In Mass Production

PIM'S
RASPBERRY
Classic European Biscuits
Le PIM'S
soft biscuits with raspberry filling
RASPBERRY
LU

- 3D Multi Chip Module
- Ultimate in Embedded Logic
- Off Shore Production
- Available in 2 device types



•Biscuit-Based Substrate

•Amorphous Doping for Single Flavor Device Type

•Single Layer Interconnect doubles as passivation

Tutorial 123

Erik P. DeBenedictis

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"
    - √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 √2 reductions in line width
  - Reducing line width by √2 doubles the number of devices
- However, ability to predict the future is imperfect →

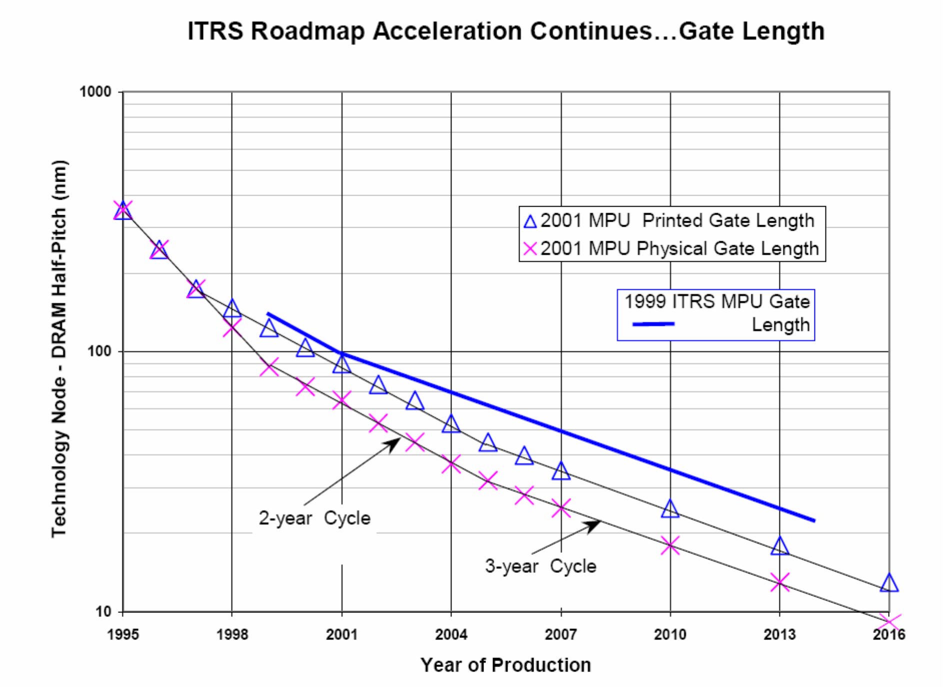


Figure 8 ITRS Roadmap Acceleration Continues—Gate Length Trends

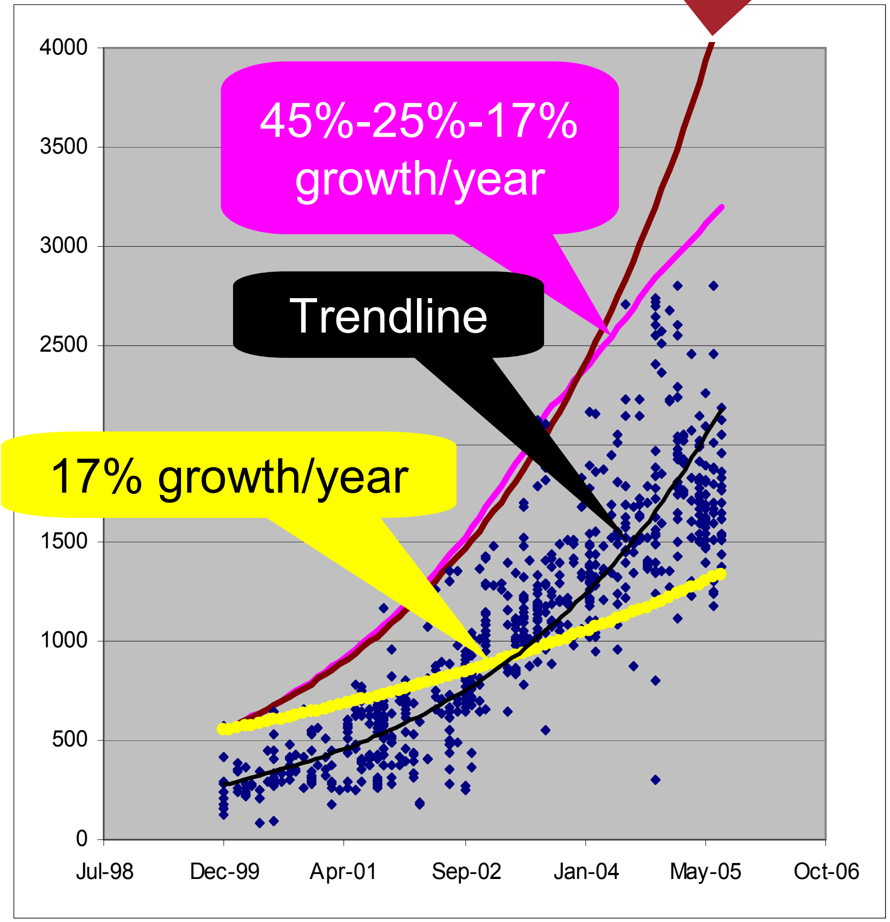
ITRS 2001 edition Executive Summary

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**



43% growth/year

45%-25%-17% growth/year

Trendline

17% growth/year

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

End of the Roadmap

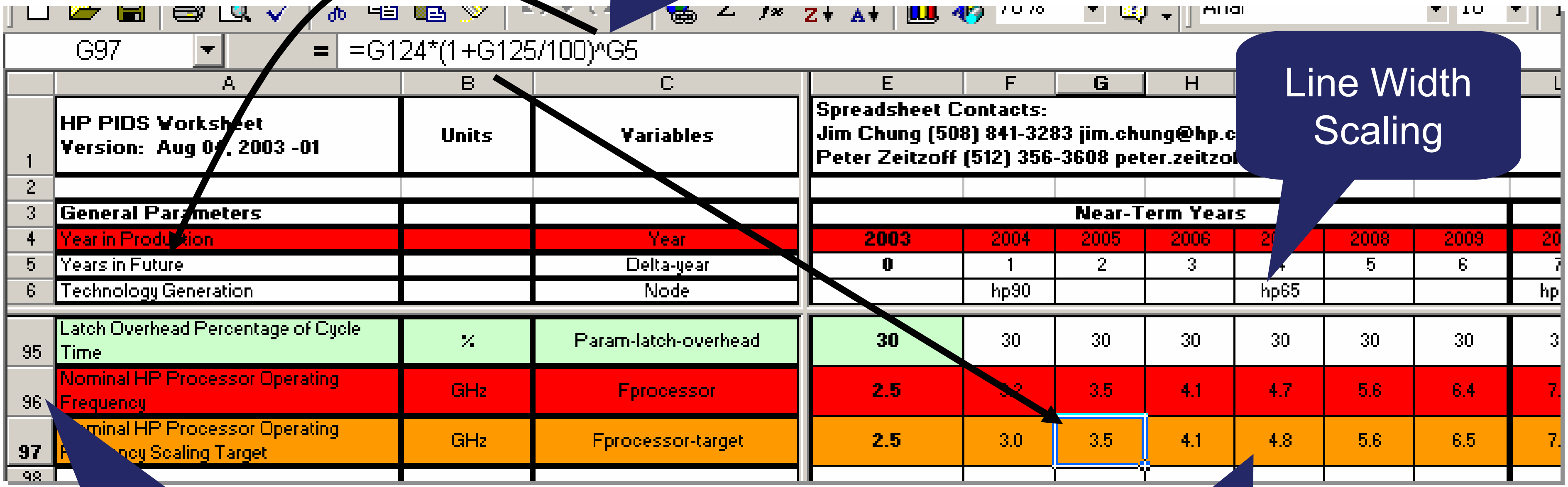
- ITRS: Exponentials, Innovations, and Equations
  - SPEC processor numbers and implications
  - The Big Spreadsheet
  - Total power and clock rate model
- Review of Burger and Keckler Study
  - Study of throughput under technology scaling
- Implications
  - Throughput scaling
  - Cache scaling
  - Bandwidth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



G97 = =G124*(1+G125/100)^G5

	A	B	C	E	F	G	H				L
1	HP PIDS Worksheet Version: Aug 0[illegible], 2003 -01	Units	Variables	Spreadsheet Contacts: Jim Chung (508) 841-3283 jim.chung@hp.c Peter Zeitzoff (512) 356-3608 peter.zeitzo							
2											
3	General Parameters			Near-Term Years							
4	Year in Production		Year	2003	2004	2005	2006	20[illegible]	2008	2009	20
5	Years in Future		Delta-year	0	1	2	3	4	5	6	7
6	Technology Generation		Node		hp90			hp65			hp
95	Latch Overhead Percentage of Cycle Time	%	Param-latch-overhead	30	30	30	30	30	30	30	3
96	Nominal HP Processor Operating Frequency	GHz	Fprocessor	2.5	3.2	3.5	4.1	4.7	5.6	6.4	7.
97	[illegible]minal HP Processor Operating F[illegible]ncy Scaling Target	GHz	Fprocessor-target	2.5	3.0	3.5	4.1	4.8	5.6	6.5	7.
98											

Line Width Scaling

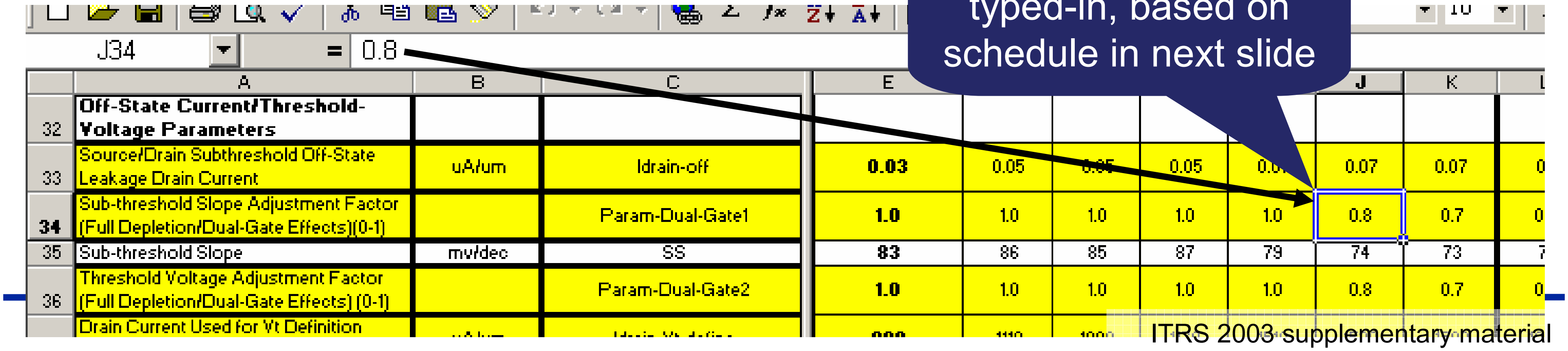
Fprocessor is result of 96 rows of targets, inputs, and iterative calculation

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, <u>leakage</u>, <u>geometry</u>, mobility, parasitic capacitance



J34 = 0.8

	A	B	C	E					J	K	L
32	**Off-State Current/Threshold-Voltage Parameters**										
33	Source/Drain Subthreshold Off-State Leakage Drain Current	uA/um	Idrain-off	**0.03**	0.05	0.05	0.05	0.0	0.07	0.07	0
34	Sub-threshold Slope Adjustment Factor (Full Depletion/Dual-Gate Effects)(0-1)		Param-Dual-Gate1	**1.0**	1.0	1.0	1.0	1.0	0.8	0.7	0
35	Sub-threshold Slope	mv/dec	SS	**83**	86	85	87	79	74	73	7
36	Threshold Voltage Adjustment Factor (Full Depletion/Dual-Gate Effects) (0-1)		Param-Dual-Gate2	**1.0**	1.0	1.0	1.0	1.0	0.8	0.7	0
	Drain Current Used for Vt Definition	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]					

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

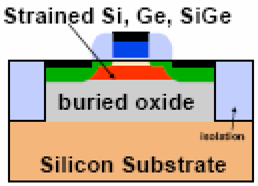
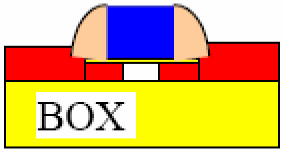
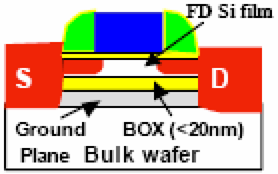
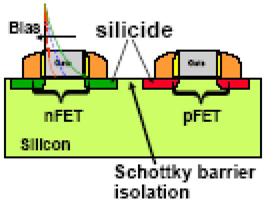
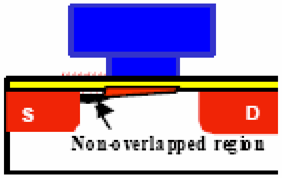
Schedule of Innovations

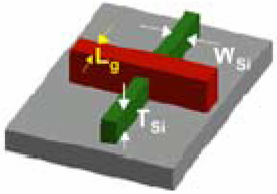
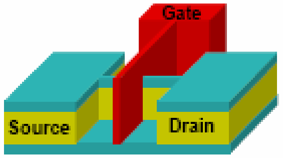
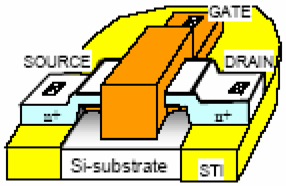
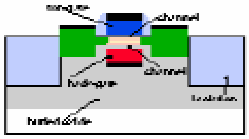
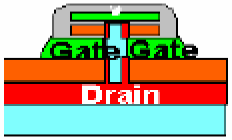
- To make the calculations fit the projection of a smooth "Moore's Law," certain variables must be adjustable
- The independent variables are a "schedule of innovations," or technology advances that must enter production on certain years

mid 2004	Strained Si
2008	Elevated S/D
mid 2007	High-k
mid 2007	Metal gate
mid 2008	Ultra-Thin Body (UTB) SOI, single gate
mid 2008	Metal Gate
mid 2010	Multiple Gate
mid 2013	Quasi-ballistic transport
	Etc.

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

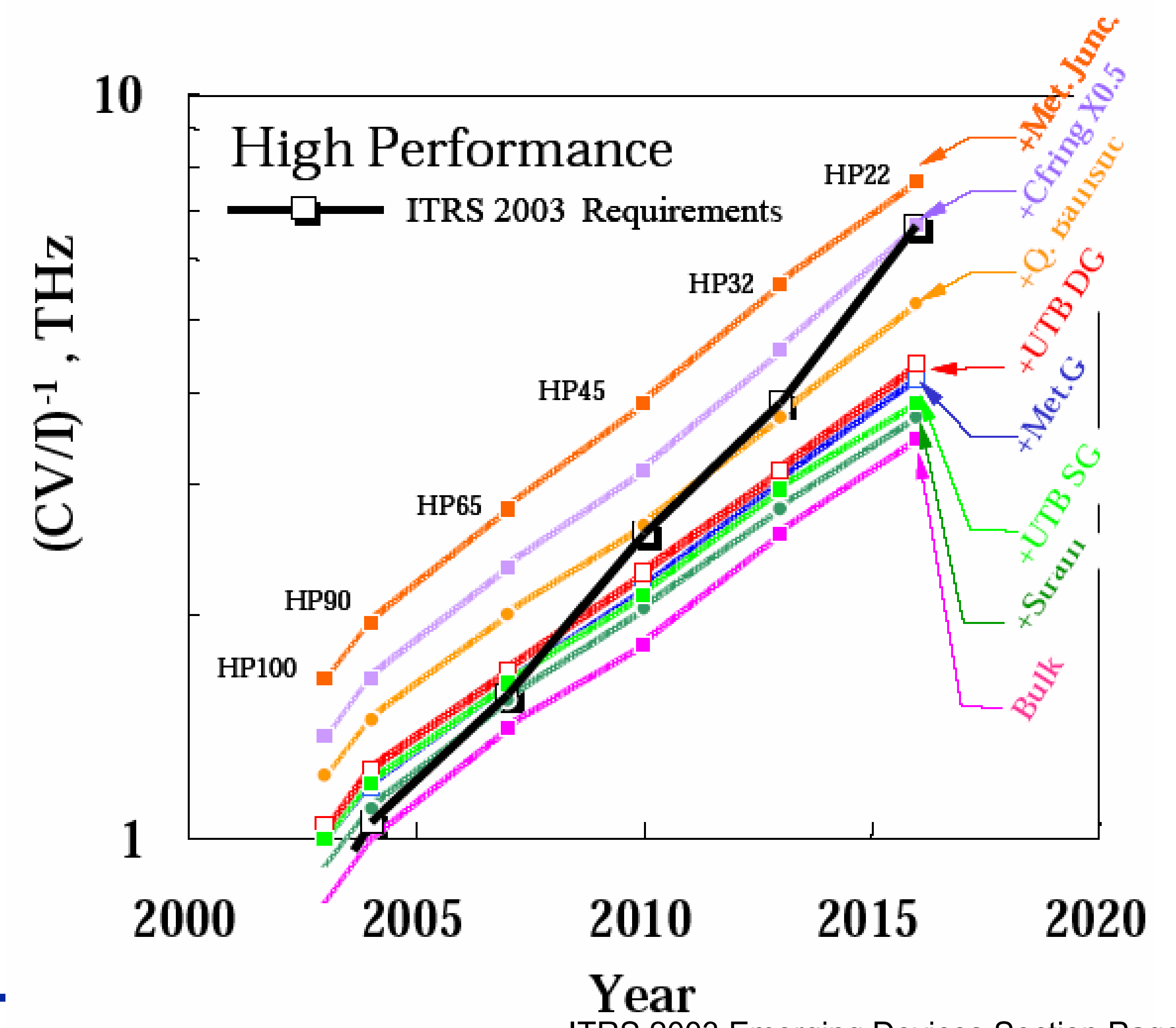
ITRS Transistor Geometries

Transport-enhanced FETs	*Ultra-thin Body SOI FETs*		*Source/Drain Engineered FETs*	
 Strained Si, Ge, SiGe buried oxide isolation Silicon Substrate		 FD Si film S D Ground Plane BOX (<20nm) Bulk wafer	 Bias silicide nFET pFET Silicon Schottky barrier isolation	
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

N-Gate (N>2) FETs	*Double-gate FETs*			
				
Tied gates (number of channels >2)	Tied gates, side-wall conduction	Tied gates planar conduction	Independently switched gates, planar conduction	Vertical conduction

ITRS 2003 Emerging Devices Section Pages 4 and 5

ITRS Technology Progression



ITRS 2003 Emerging Devices Section Page 12

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

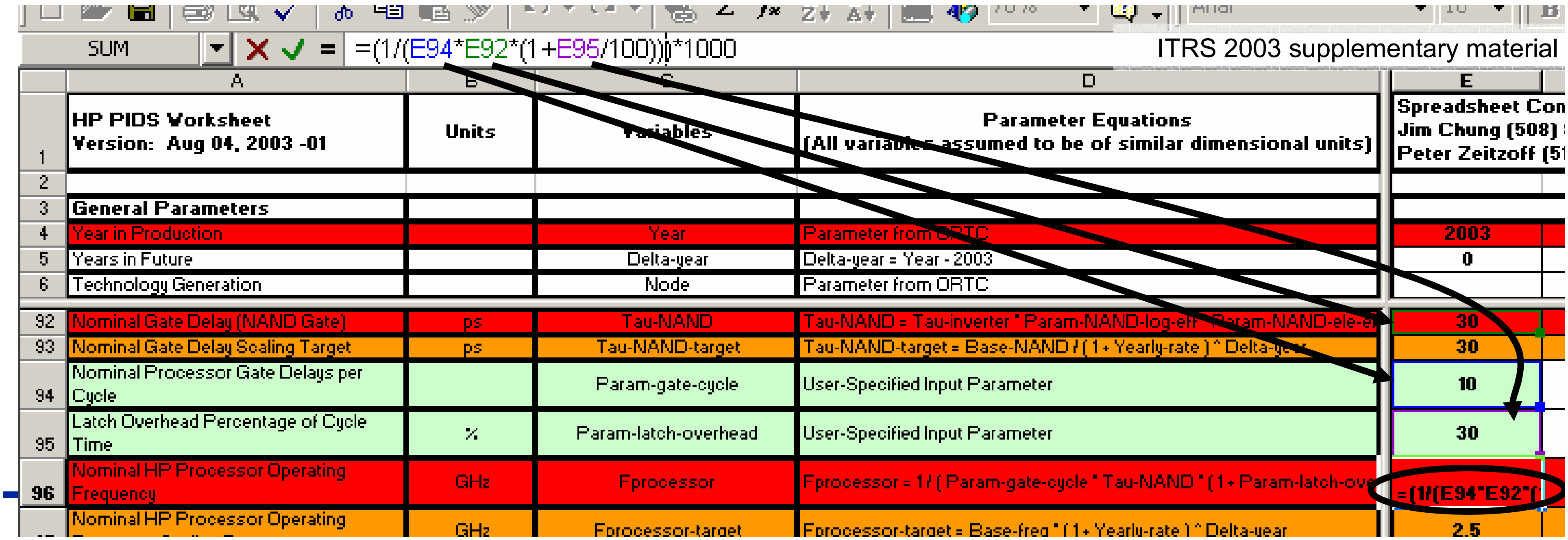
Power Dissipation

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

See “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^2 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** **Version: Aug 04, 2003 -01**	**Units**	**Variables**	**Parameter Equations** **(All variables assumed to be of similar dimensional units)**	**Spreadsheet Con** **Jim Chung (508)** **Peter Zeitzoff (5**
2					
3	**General Parameters**				
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-eff * Param-NAND-ele-e	**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-ove	**=(1/(E94*E92*(**
	Nominal HP Processor Operating	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

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(SpecFP)

engineering
← choices,
architecture

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

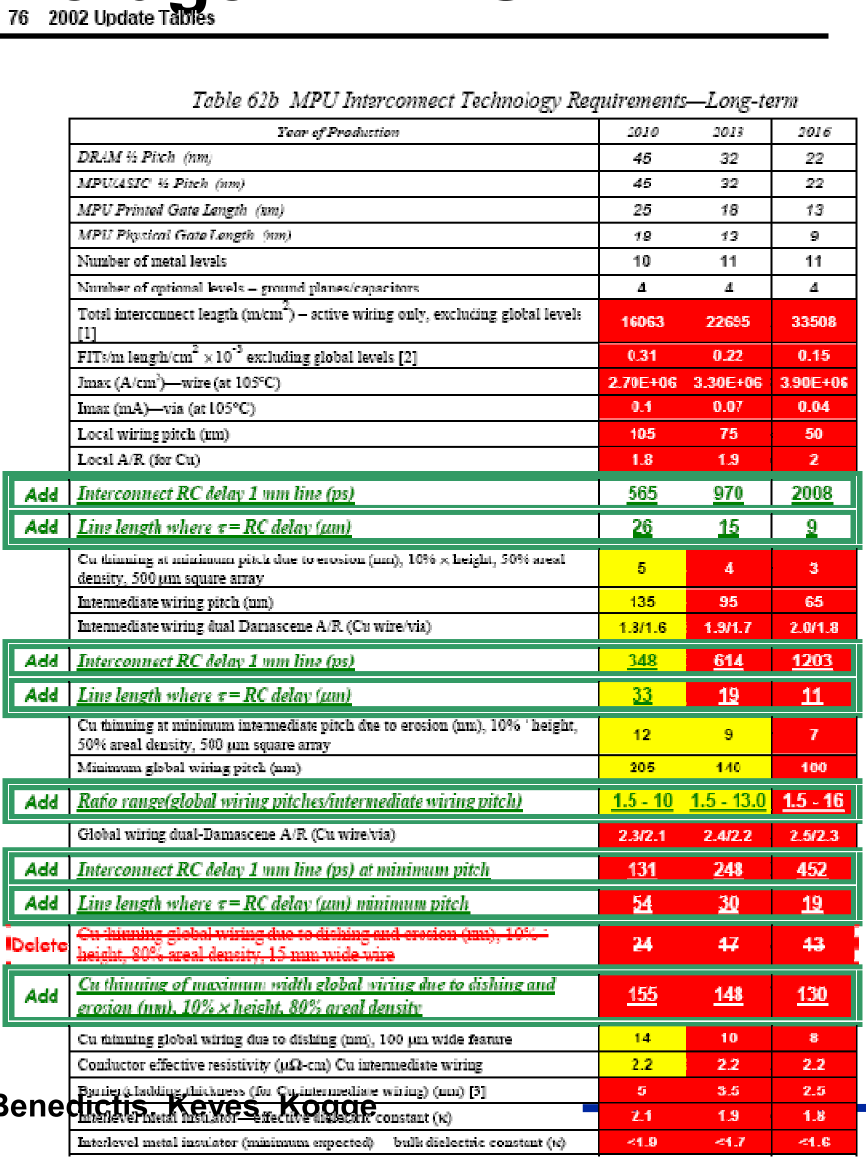
UT Austin Study (2000)

- The Study
  - Clock Rate versus IPC: The End of the Road for Conventional Microarchitectures, Vikas Agarwal, M.S. Hrishikesh, Stephen W. Keckler, Doug Burger. 27th 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



76 2002 Update Tables

Table 62b MPU Interconnect Technology Requirements—Long-term

	Year of Production	*2010*	*2013*	*2016*
	DRAM ½ Pitch (nm)	45	32	22
	MPU/ASIC ½ 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^2) – active wiring only, excluding global levels [1]	16063	22695	33508
	FITs/m length/cm^2 × 10^{-3} excluding global levels [2]	0.31	0.22	0.15
	Jmax (A/cm^2)—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 τ = RC delay (µm)*	26	15	9
	Cu thinning at minimum pitch due to erosion (nm), 10% × height, 50% areal density, 500 µm square array	5	4	3
	Intermediate wiring pitch (nm)	135	95	65
	Intermediate wiring dual Damascene A/R (Cu wire/via)	1.3/1.6	1.9/1.7	2.0/1.8
Add	*Interconnect RC delay 1 mm line (ps)*	348	614	1203
Add	*Line length where τ = RC delay (µm)*	33	19	11
	Cu thinning at minimum intermediate pitch due to erosion (nm), 10% ' height, 50% areal density, 500 µm square array	12	9	7
	Minimum global wiring pitch (nm)	205	140	100
Add	*Ratio range(global wiring pitches/intermediate wiring pitch)*	1.5 - 10	1.5 - 13.0	1.5 - 16
	Global wiring dual-Damascene A/R (Cu wire/via)	2.3/2.1	2.4/2.2	2.5/2.3
Add	*Interconnect RC delay 1 mm line (ps) at minimum pitch*	131	248	452
Add	*Line length where τ = RC delay (µm) minimum pitch*	54	30	19
Delete	~~Cu thinning global wiring due to dishing and erosion (nm), 10% × height, 80% areal density, 15 mm wide wire~~	24	47	43
Add	*Cu thinning of maximum width global wiring due to dishing and erosion (nm), 10% × height, 80% areal density*	155	148	130
	Cu thinning global wiring due to dishing (nm), 100 µm wide feature	14	10	8
	Conductor effective resistivity (µΩ-cm) Cu intermediate wiring	2.2	2.2	2.2
	Barrier/cladding thickness (for Cu intermediate wiring) (nm) [3]	5	3.5	2.5
	Interlevel metal insulator—effective dielectric constant (κ)	2.1	1.9	1.8
	Interlevel metal insulator (minimum expected) bulk dielectric constant (κ)	<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 →

See 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

See 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 μP

- Authors used SimpleScalar, cycle accurate simulator of a DEC Alpha 21264
- However, actually models hypothetical future μPs 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 μP core, and SPEC is for one core

See 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 μP 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)

See 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

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

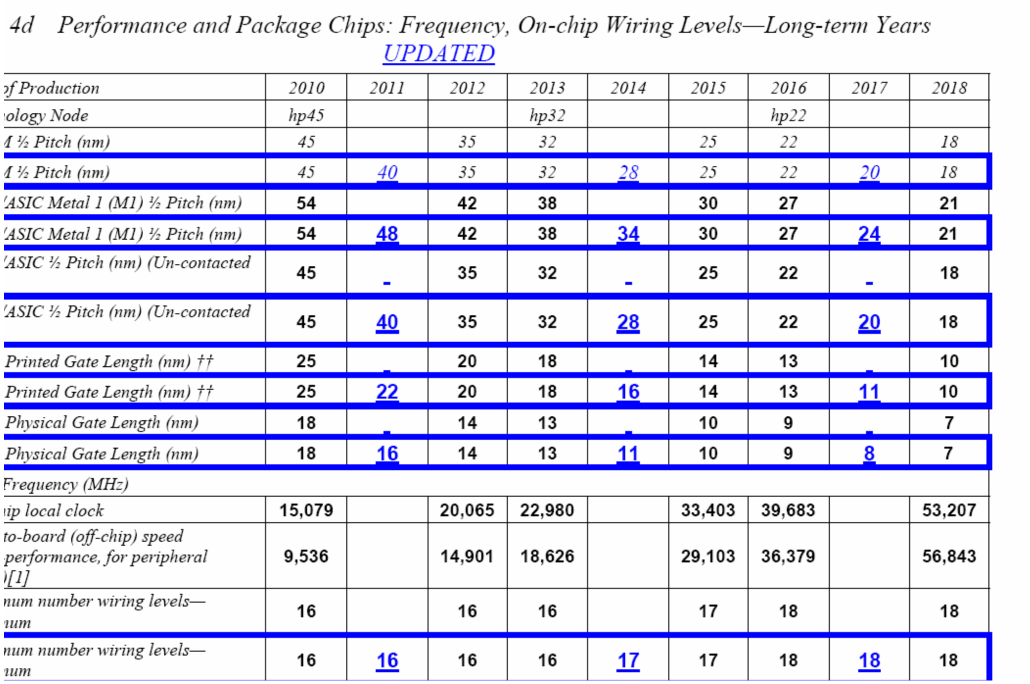
- Review of Issues
  - Thread speed & parallelism
  - Inner loop memory requirements
  - FLOPS/watt
  - Devices per chip (multi-core scaling)
  - Surface-to-area ratio
  - Load imbalance revealed by synchronization overhead
- Example
  - Instructor led example of projecting performance of a mesh algorithm

Thread Speed and Parallelism

- Runtime ≥ sequential ops÷thread speed
- Single thread FLOPS rate determined by
  - Gate speed
    - ITRS tell you this
  - Architecture
    - ~9 gate delays in a μP
    - 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



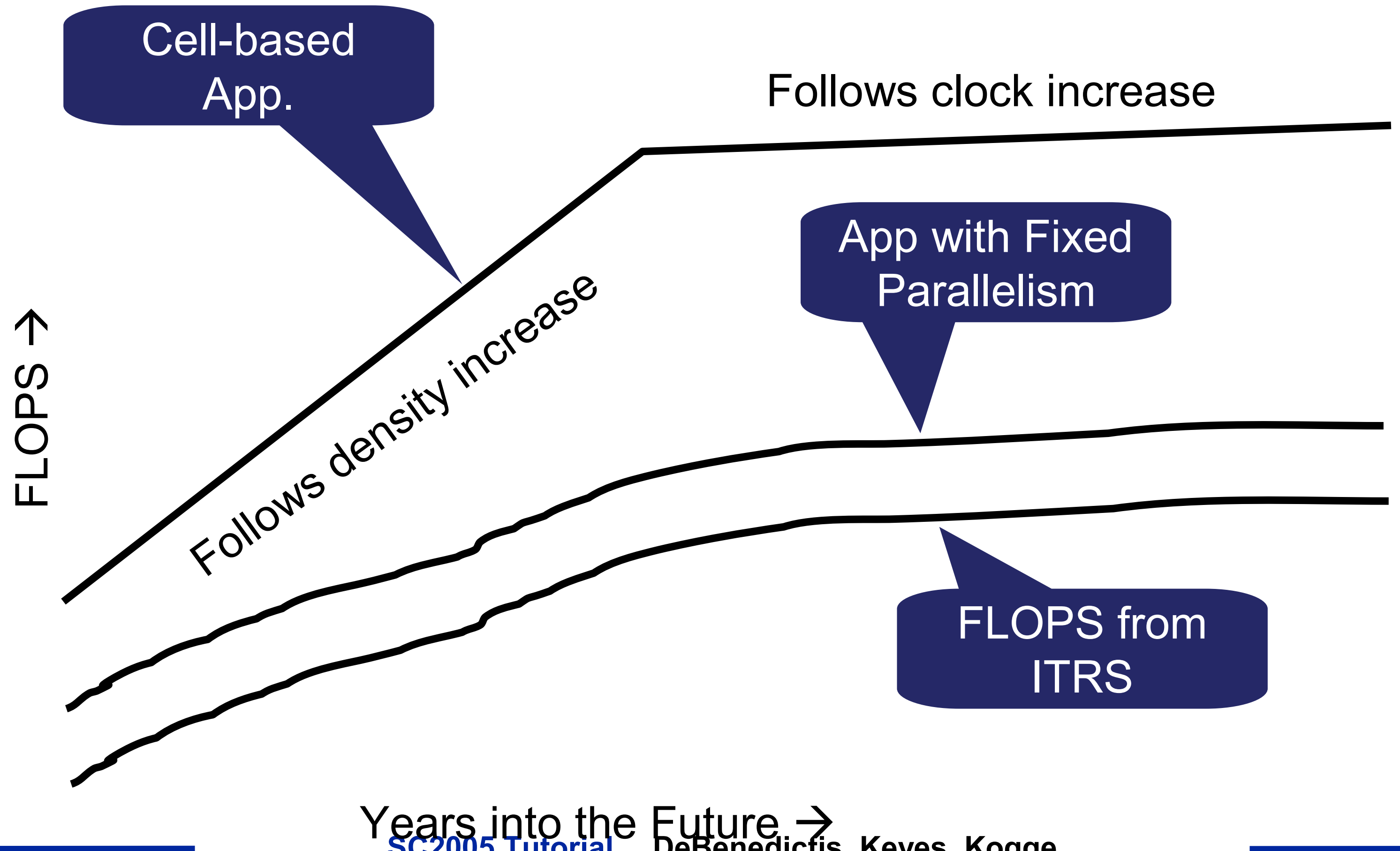
4d Performance and Package Chips: Frequency, On-chip Wiring Levels—Long-term Years

UPDATED

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

- 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

Implications of Thread Speed & Parallelism



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×d chip area
- Minimum access time will be 2d÷v
  - v is signal propagation velocity
  - modulo constants
- Is this some hypothetical architectural thing?
  - Not necessarily, applies to existing μPs 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

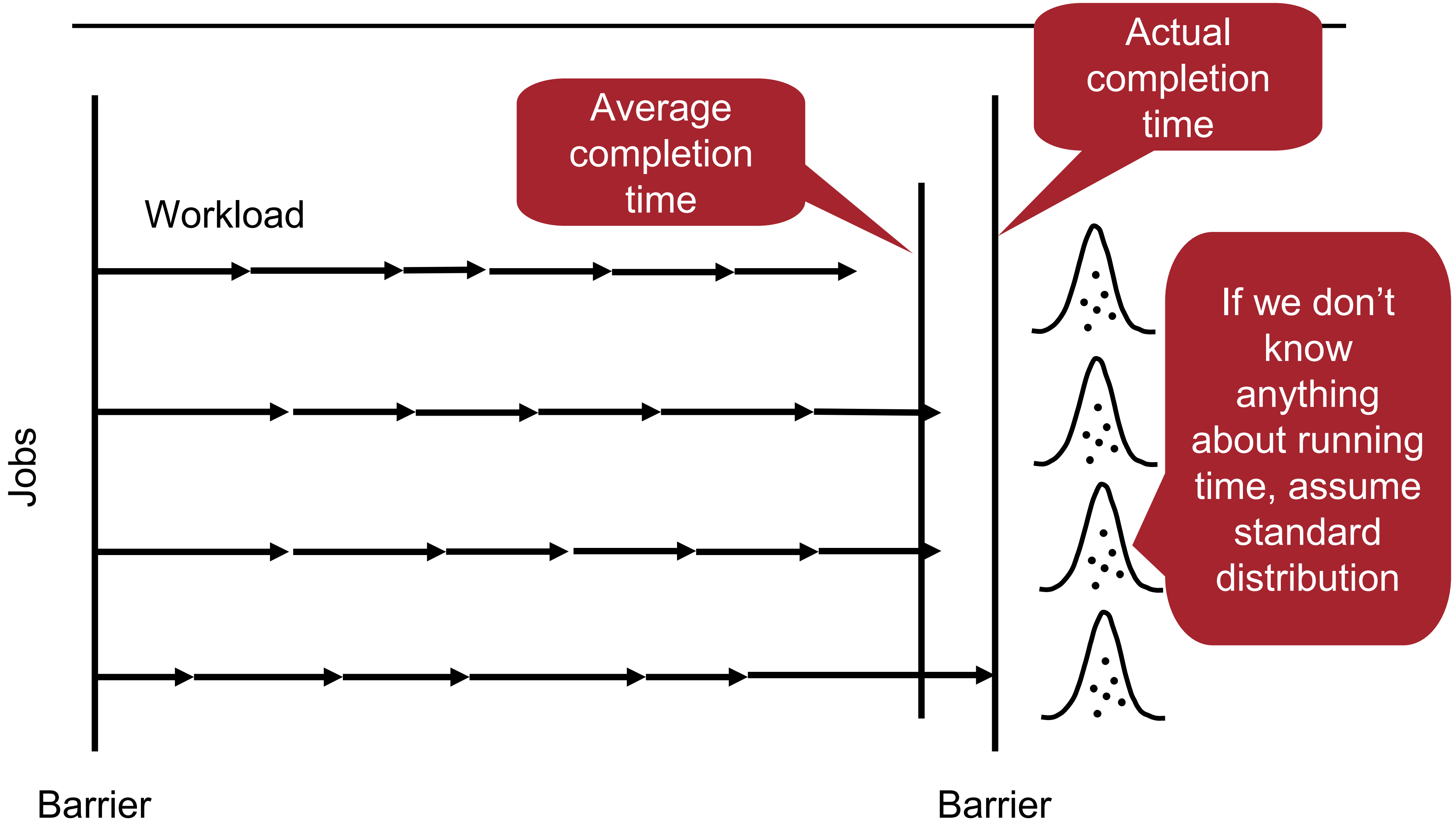
Bandwidth Scaling

- 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

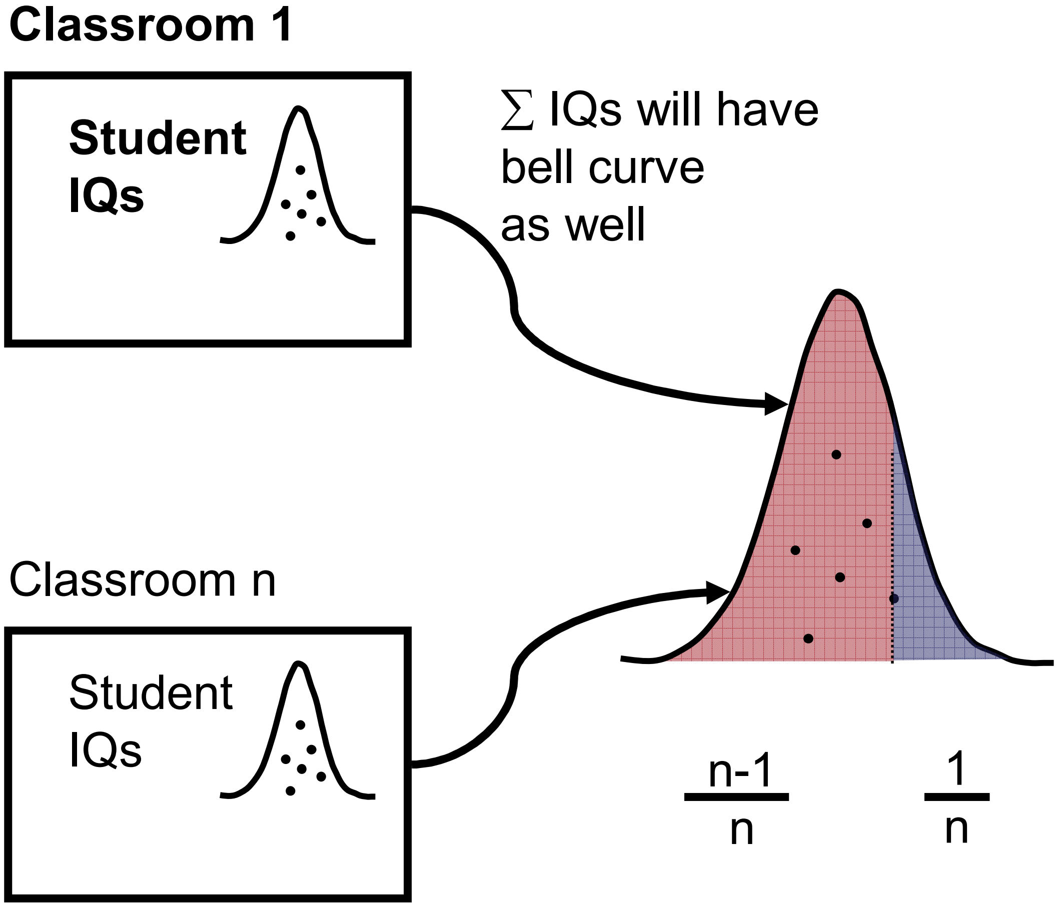
Projecting Applications Performance

- 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


Average completion time
Actual completion time
Workload
Jobs
If we don't know anything about running time, assume standard distribution
Barrier
Barrier

Maximum IQ of a Class in Your Kids School

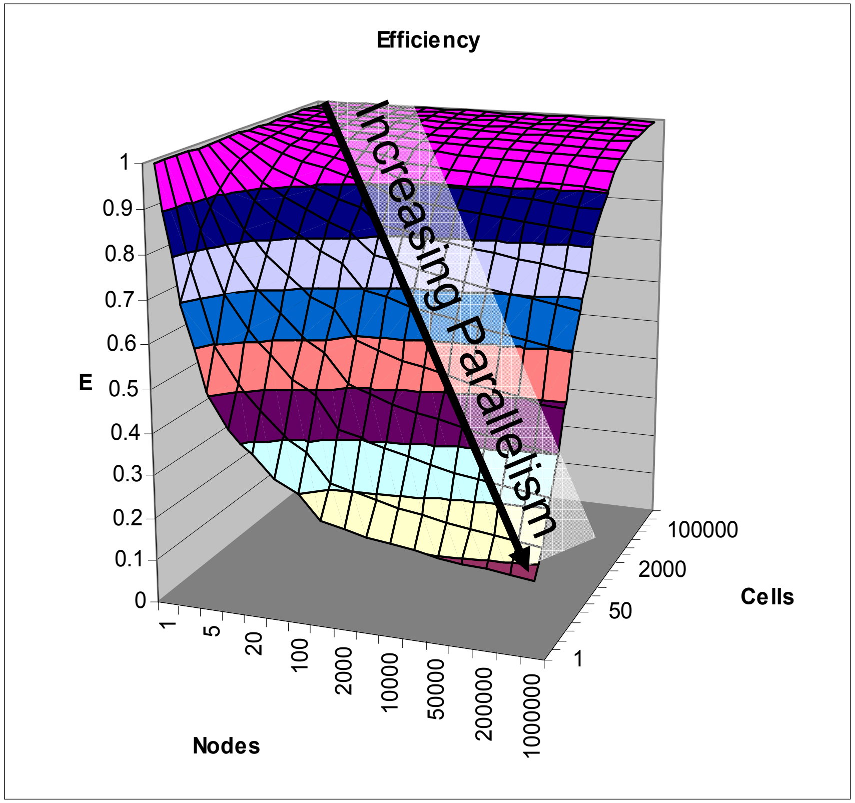


- Each child has average IQ 100 and std of 15
  - Mean and std of task runtime
- Each class has total IQ of n×100 and std of $n^{½}$×15
  - Statistics of per node time between barriers
- Max average is inverse of cumulative normal distribution evaluated at n

Efficiency Loss Due To Load Balance

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

- Plot Mean=Std



[Defining equations in PowerPoint notes]

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

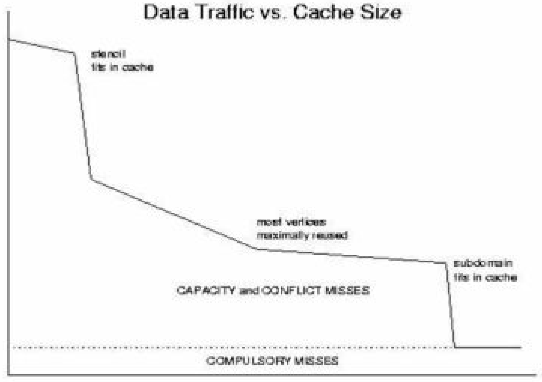
Example Problem: Future Mesh Problem

- 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 2½ cores, you’re probably out of luck.**

Hands-On Exercises

- Organization
  - Group divides into sections of 3-6 people each
  - Will hand out pertinent sections of ITRS and applications reference materials
- Problem #1: Project parameters of a $10M supercomputer in year 2016
- Problem #2: Performance on an application without source code available
- Problem #3: Performance on mesh application
- Problem #4: Performance on a PIM architecture supercomputer

Hands-On Exercises

- Organization
  - Group divides into sections of 3-6 people each
  - Will hand out pertinent sections of ITRS and applications reference materials
- Problem #1: Project parameters of a $10M supercomputer in year 2016
- Problem #2: Performance on an application without source code available
- Problem #3: Performance on mesh application
- Problem #4: Performance on a PIM architecture supercomputer

Problem 1: Hardware Projection

- Say you are in charge of buying a $10M supercomputer in the year 2016
- Project parameters for the supercomputer you'd like to buy, based on
  - Extrapolations from cost, performance, and configuration parameters of a recently constructed supercomputer of your choice
    - Instructors can provide information on Red Storm
  - Roadmap documents distributed in the session

Hands-On Exercises

- Organization
  - Group divides into sections of 3-6 people each
  - Will hand out pertinent sections of ITRS and applications reference materials
- Problem #1: Project parameters of a $10M supercomputer in year 2016
- Problem #2: Performance on an application without source code available
- Problem #3: Performance on mesh application
- Problem #4: Performance on a PIM architecture supercomputer

Problem 2: Black Box Software

- A chemist runs CHARMM on a 32 node cluster, 8 jobs at a time (4 node jobs)
- The user can’t get scaling beyond 4 nodes, and the user is a chemist uninterested in recoding
- Question: How much faster will each job run in 2016?
- Question: How many nodes will be required in 2016 to get 100× throughput increase?

Hands-On Exercises

- Organization
  - Group divides into sections of 3-6 people each
  - Will hand out pertinent sections of ITRS and applications reference materials
- Problem #1: Project parameters of a $10M supercomputer in year 2016
- Problem #2: Performance on an application without source code available
- Problem #3: Performance on mesh application
- Problem #4: Performance on a PIM architecture supercomputer

Problem 3: Mesh Application

- Your task is to solve a mesh-based application on a billion point (1000^3) mesh
- Algorithm parameters
  - – 256 bytes data per mesh point
  - – 6 point stencil
  - – 5 global reductions per time step
- The year is 2014 and you have $20M to buy a machine
- How much wall clock time can you expect per time step?

Hands-On Exercises

- Organization
  - Group divides into sections of 3-6 people each
  - Will hand out pertinent sections of ITRS and applications reference materials
- Problem #1: Project parameters of a $10M supercomputer in year 2014
- Problem #2: Performance on an application without source code available
- Problem #3: Performance on mesh application
- Problem #4: Performance on a PIM architecture supercomputer

Problem 4: PIM Application

- You are to specify a supercomputer to solve a 100 million molecule DSMC problem in 2010
- Each DSMC molecule has float parameters x, y, z, vx, vy, vz, and may be one of 100 species
- Molecules spend about 3 time steps in a cell before moving to an adjacent cell
- Calculating the interactions and/or chemical reactions takes 5000 floating operations per molecule per timestep
- Assume the region is a regular cubic mesh
- How many cores and how much RAM per PIM chip would be required to solve the problem optimally

Beyond Transistors

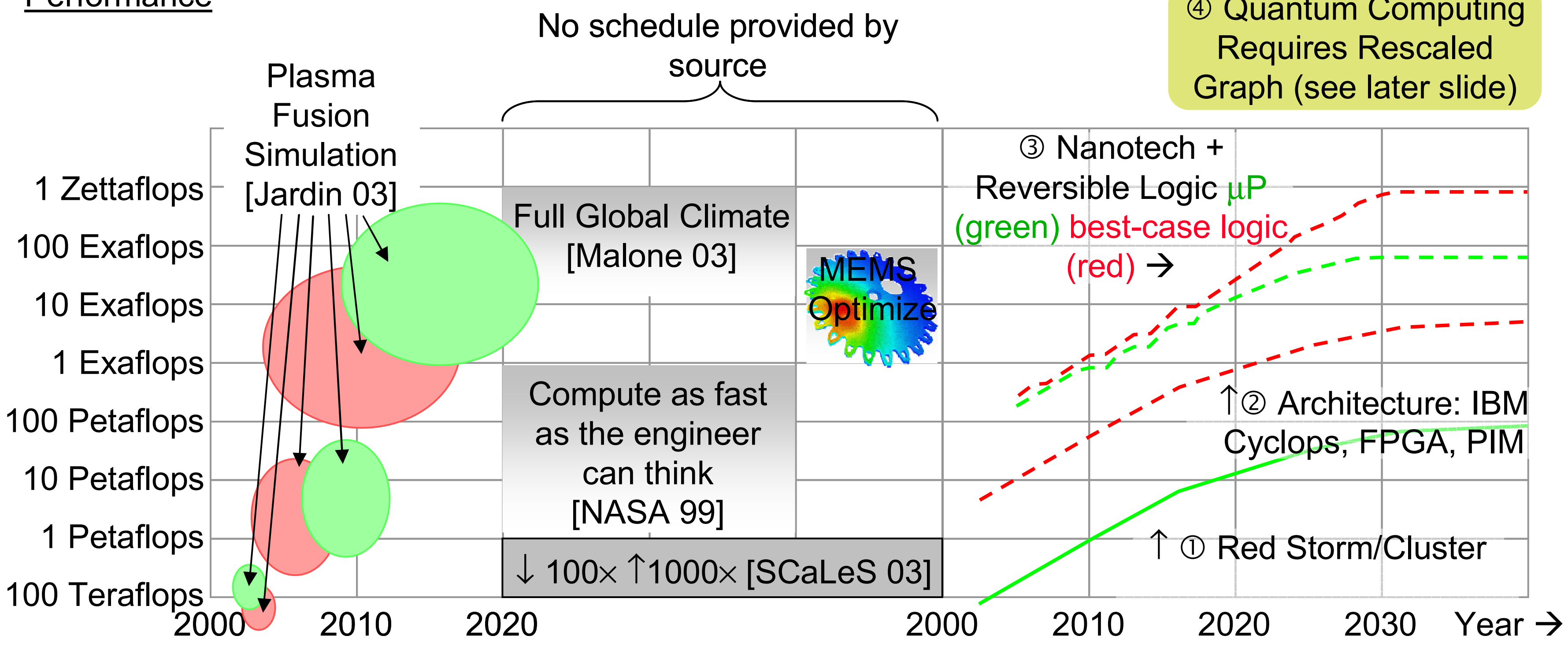
- Applications Requirements
- Thermodynamic limits to total power
  - Superconducting logic and Carnot cycle
- Upside potential of advanced architectures/PIM
- Some nanotech technologies on the horizon
- Reversible logic may defeat thermodynamic limitations
- Upside potential of quantum computing
  -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

Applications and $100M Supercomputers

System Performance

Applications

Technology



④ Quantum Computing Requires Rescaled Graph (see later slide)

No schedule provided by source

Plasma Fusion Simulation [Jardin 03]

Full Global Climate [Malone 03]

Compute as fast as the engineer can think [NASA 99]

↓ 100× ↑1000× [SCaLeS 03]

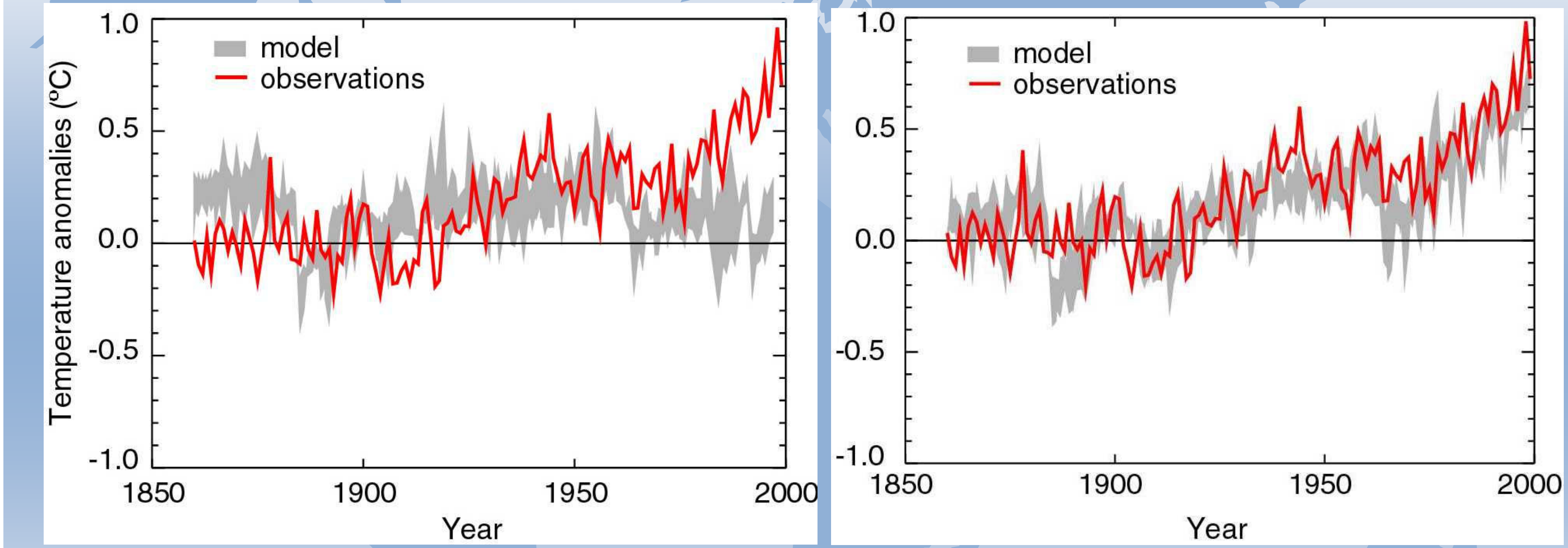
③ Nanotech + Reversible Logic µP (green) best-case logic (red) →

↑② Architecture: IBM Cyclops, FPGA, PIM

↑ ① Red Storm/Cluster

[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

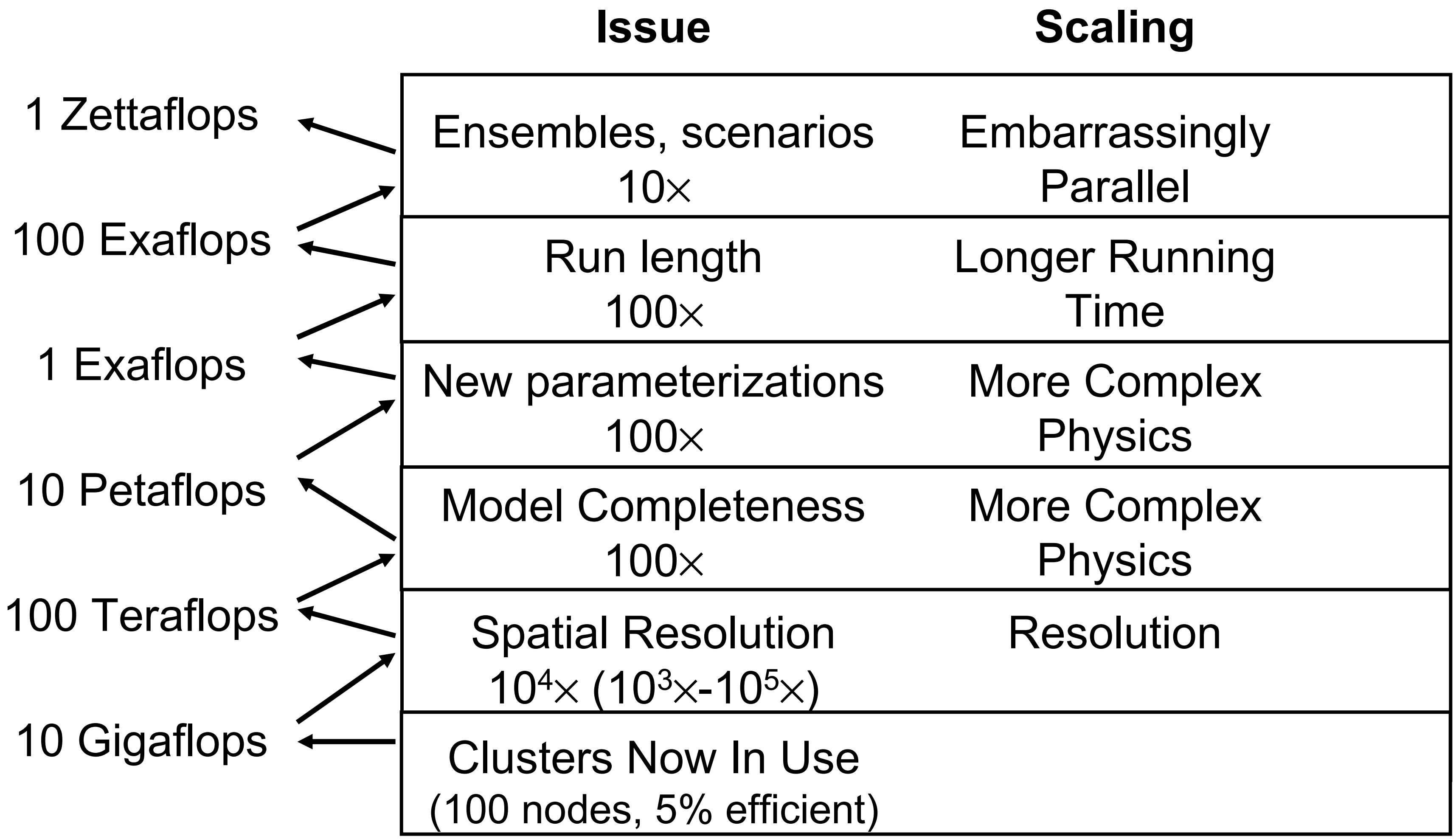


Stott et al, Science 2000

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

“..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



1 Zettaflops

100 Exaflops

1 Exaflops

10 Petaflops

100 Teraflops

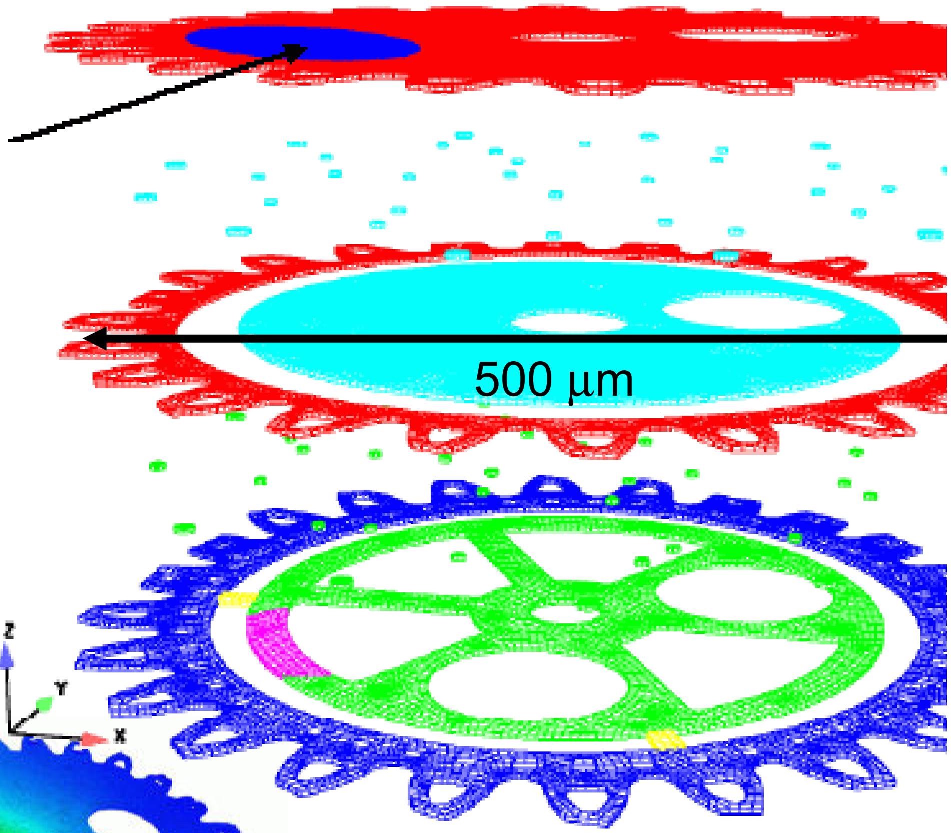
10 Gigaflops

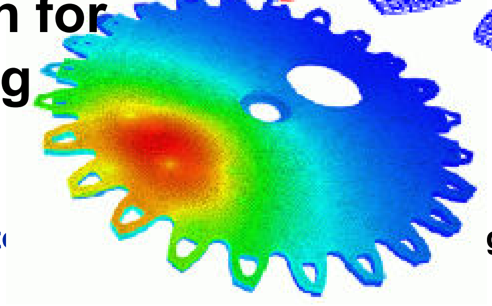
Issue	Scaling
Ensembles, scenarios 10×	Embarrassingly Parallel
Run length 100×	Longer Running Time
New parameterizations 100×	More Complex Physics
Model Completeness 100×	More Complex Physics
Spatial Resolution $10^4\times$ ($10^3\times$-$10^5\times$)	Resolution
Clusters Now In Use (100 nodes, 5% efficient)	

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)

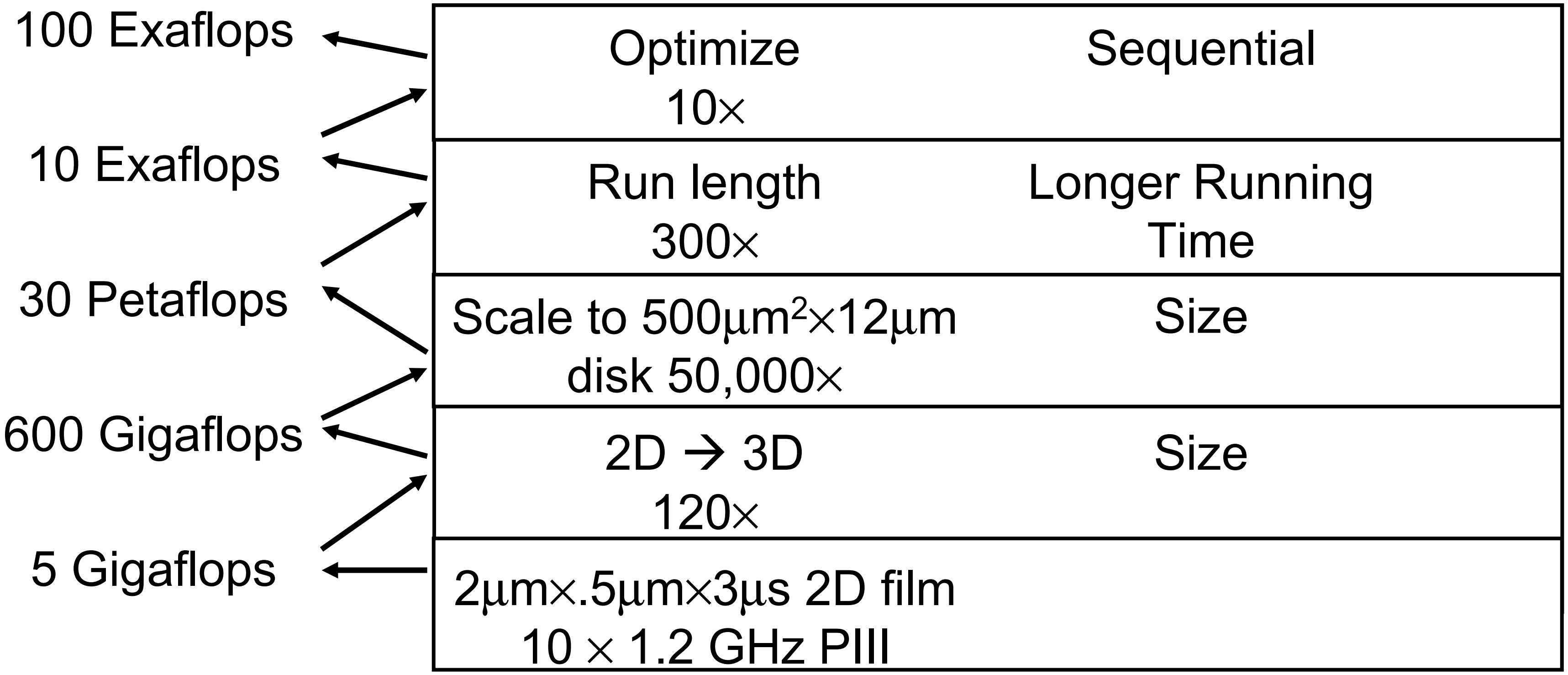
Exemplary Exa- and Zetta-Scale Simulations

- 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 500μm²×12μm disk 50,000×	Size
600 Gigaflops	2D → 3D 120×	Size
5 Gigaflops	2μm×.5μm×3μs 2D film 10 × 1.2 GHz PIII	

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

Landauer's Arguments

- Landauer makes three arguments in his 1961 paper
  - Kintetics 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:

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

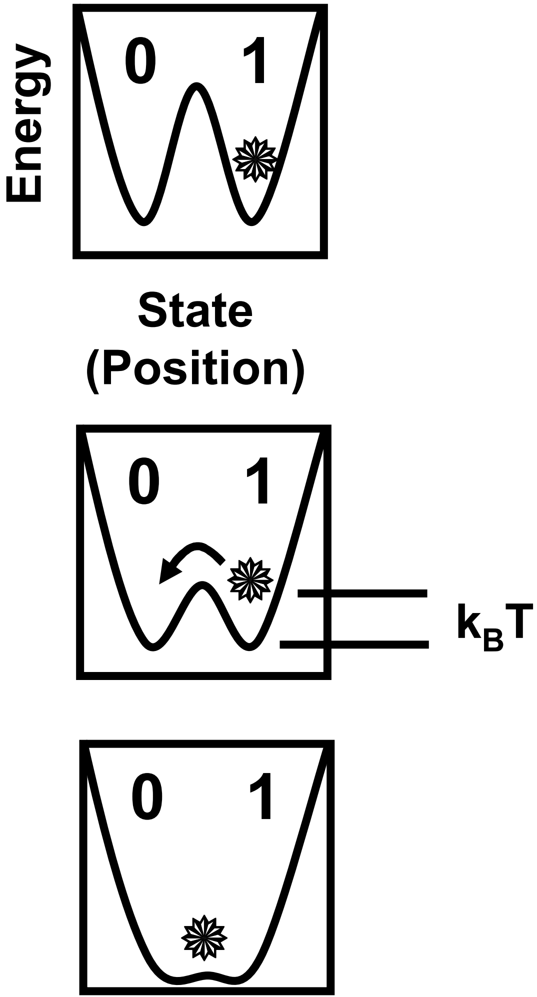
- After clearing the flip flop:

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

- Difference $k_B \log_e(2)$

Landauer’s Limit

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



Thermal Limit

- The probability of a "logic glitch" due to thermal noise is approximately e^{-N}, where $N=E_{sig}/k_BT$
- 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_BT energy per gate operation.
- ITRS predicts 30 k_BT. While Erik thinks such devices might be manufacturable, redundancy in logic design should outweigh benefit
  - Also, MPF observation about information representation

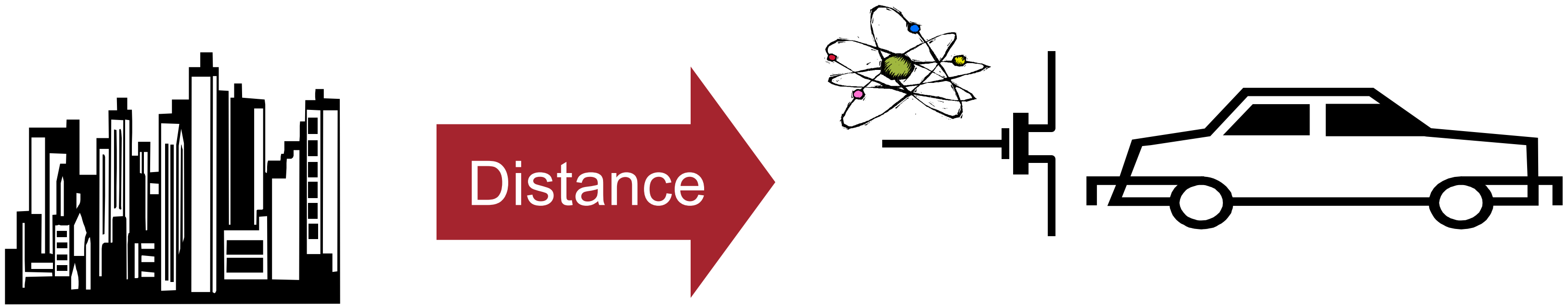
Metaphor: FM Radio on Trip to Portland

- You drive to Portland listening to FM radio
- Music clear for a while, but noise creeps in and then overtakes music

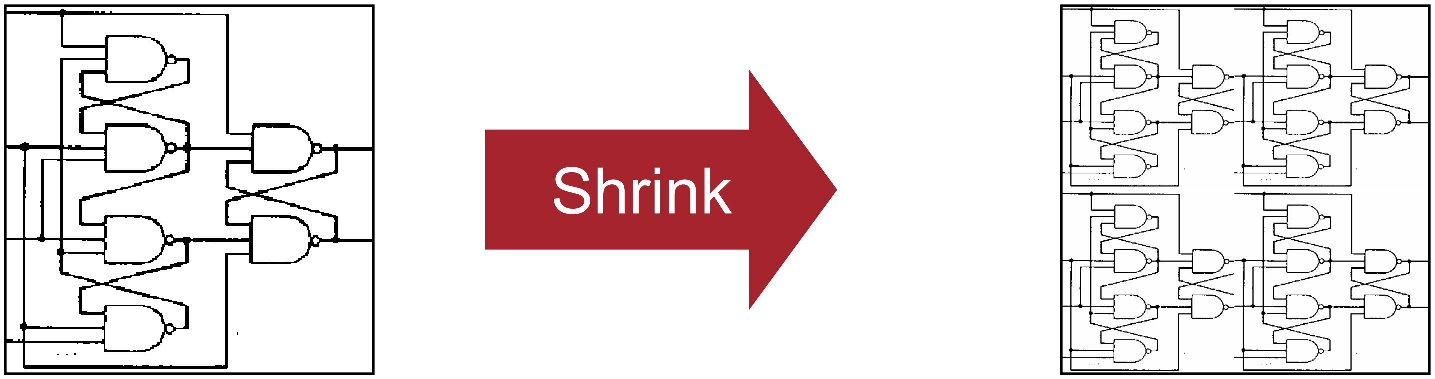
- Analogy: You live out the next dozen years buying PCs every couple years
- PCs keep getting faster
  - clock rate increases
  - fan gets bigger
  - won't go on forever
- Why…see next slide

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

FM Radio and End of Moore’s Law



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



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

Personal Observational Evidence

- Have radios become better able to receive distant stations over the last few decades with a rate of improvement similar to Moore’s Law?

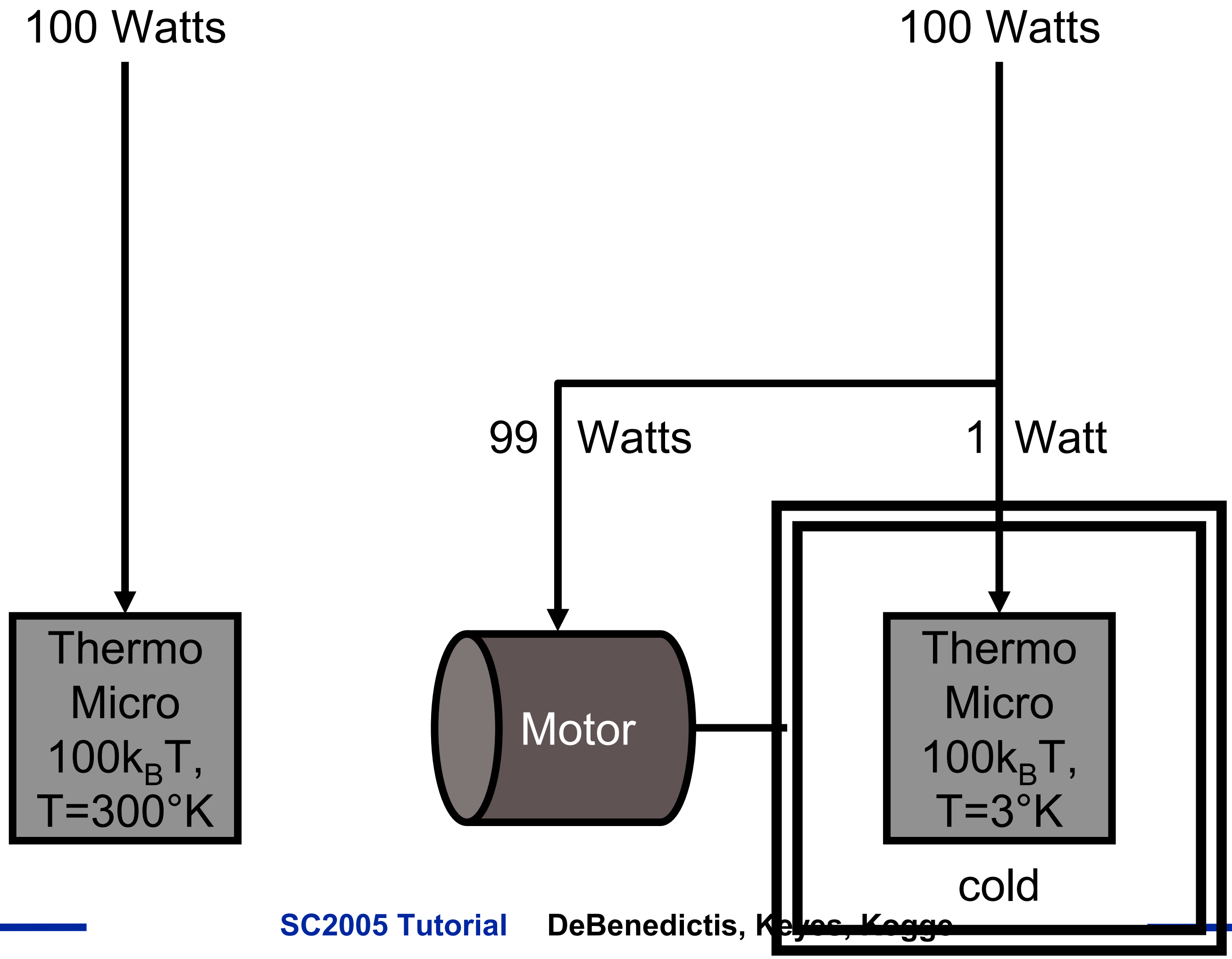
- You judge from your experience, but the answer should be that they have not.

- Therefore, electrical noise does not scale with Moore’s Law.

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


100 Watts
100 Watts
99 Watts
1 Watt
Thermo Micro $100k_BT$, T=300°K
Motor
Thermo Micro $100k_BT$, T=3°K
cold

Cutting Temperature

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

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!

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

$= \frac{300 - 3}{3}$

$= 99$

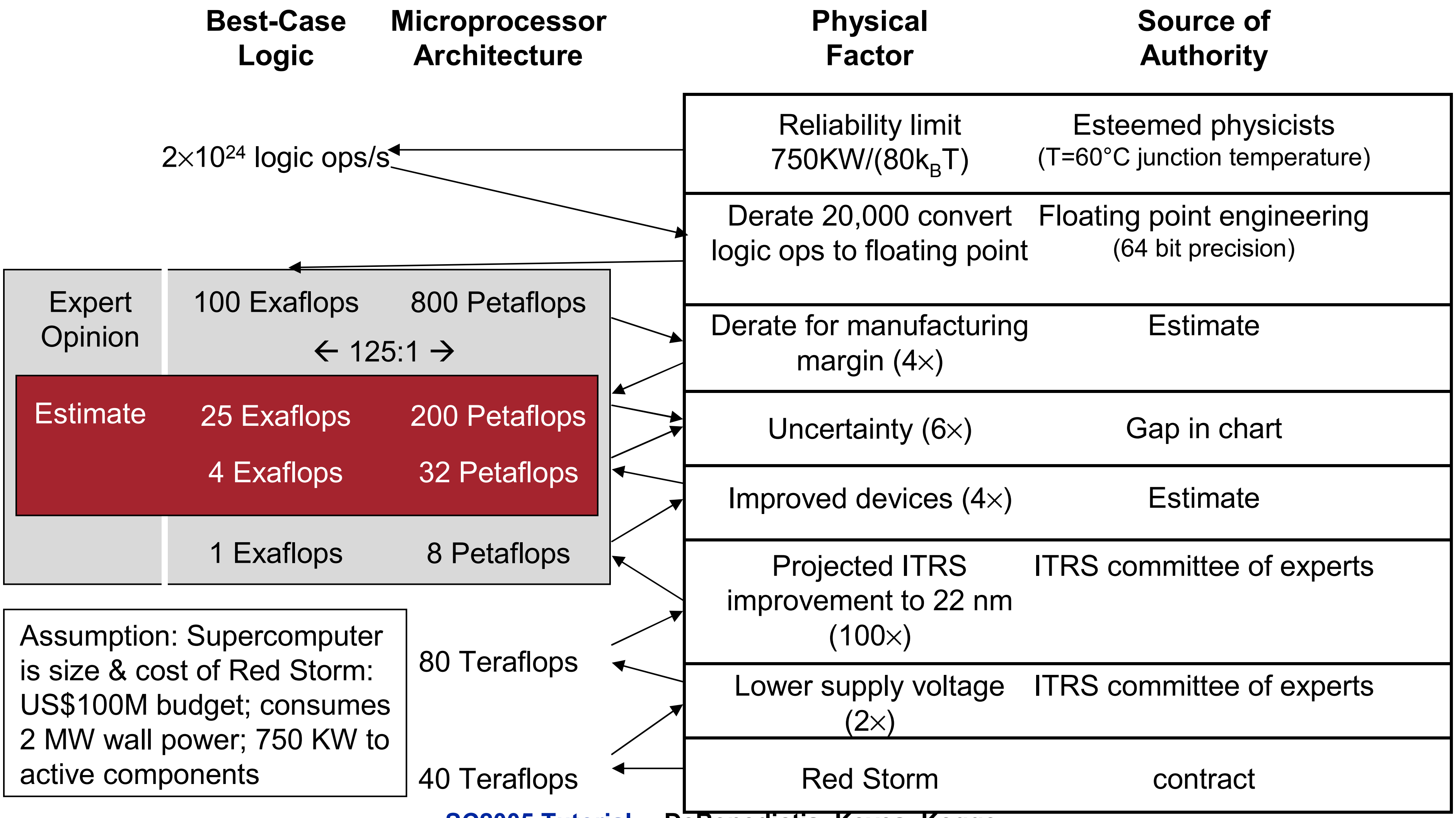
Thus, we cut device power to 1% of original power at the price of a refrigerator consuming 99% of the original power, for resulting total power consumption of 100% of original power.

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
2×10^{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
	80 Teraflops
	40 Teraflops

Physical Factor	Source of Authority
Reliability limit $750\text{KW}/(80k_BT)$	Esteemed physicists (T=60°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

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

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

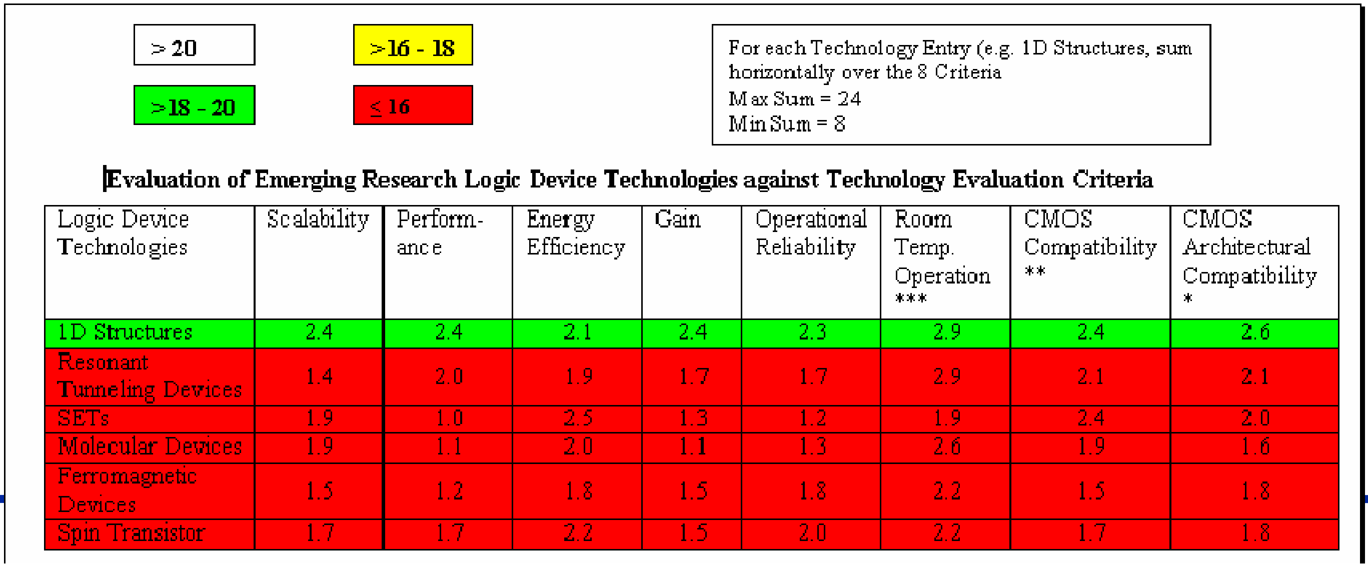
Transistors vs. Other Irreversible Devices

- Erik’s View
  - My contacts on the ITRS staff tell me they believe transistors will get to the ~30 k_BT level. If this is so, transistors will be difficult to beat in this domain.
  - At 30 k_BT, 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× 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



□ > 20

🟨 >16 - 18

🟩 >18 - 20

🟥 ≤ 16

For each Technology Entry (e.g. 1D Structures, sum horizontally over the 8 Criteria
Max Sum = 24
Min Sum = 8

Evaluation of Emerging Research Logic Device Technologies against Technology Evaluation Criteria

Logic Device Technologies	Scalability	Perform-ance	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 µs	8 nm-5 µm	4 aJ	
RSFQ	1 ps-50 ps	300 nm- 1µm	2 aJ	Larger
Molecular	10 ns-1 ms	1 nm- 5 nm	10 zJ	Slower
Plastic	100 µs-1 ms	100 µm-1 mm	4 aJ	Larger+Slower
Optical	100 as-1 ps	200 nm-2 µm	1 pJ	Larger+Hotter
NEMS	100 ns-1 ms	10-100 nm	1 zJ	Slower+Larger
Biological	100 fs-100 µs	6-50 µm	.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

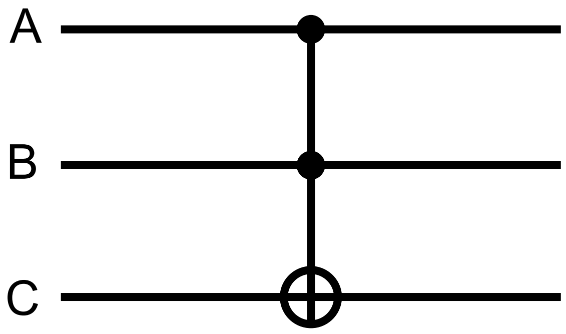
Data from ITRS ERD Section, data from Notre Dame

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

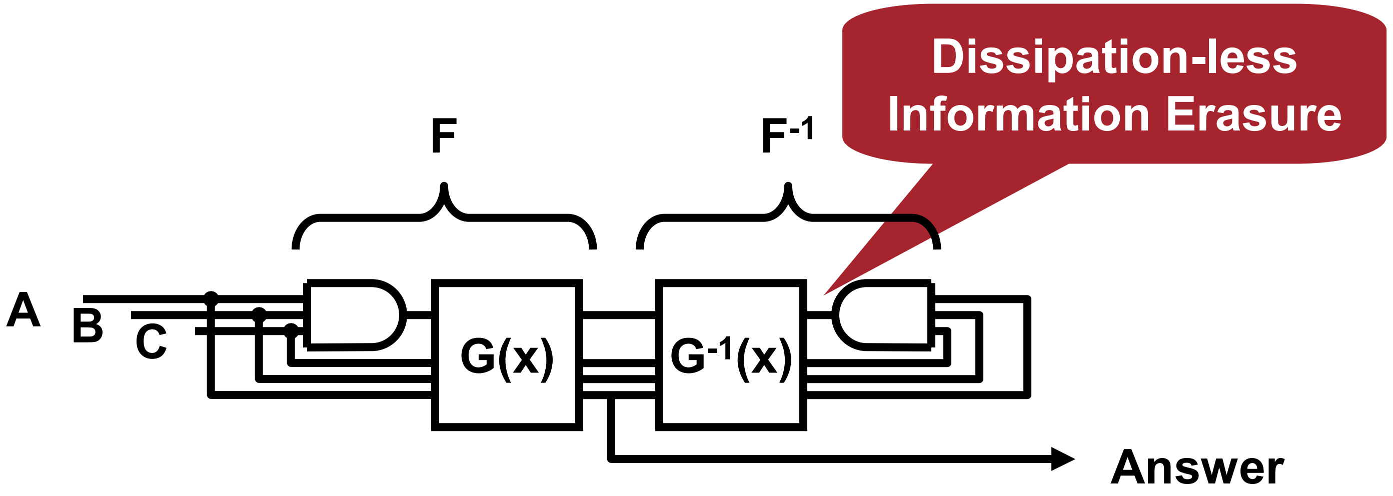
Reversible Logic – Toffoli Gate

- 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



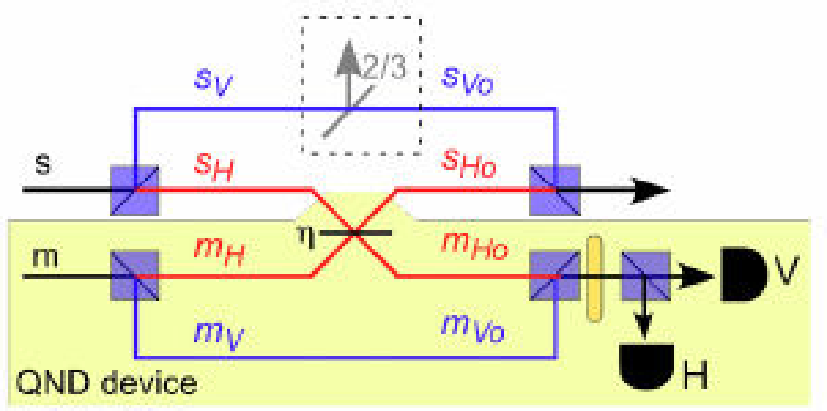
Reversible Logic Can Beat Landauer's Limit

- 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

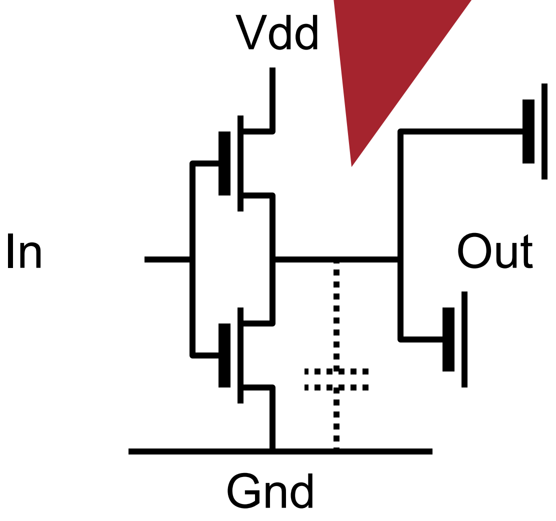


- Measuring a Photonic Qubit without Destroying It. GJ Pryde, JL O’Brien, AG White, SD Bartlett, and TC Ralph. Centre for Quantum Computer Technology, ...

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 $E_{sw} = ½ CV^2 >$ ~$100k_BT$ heat

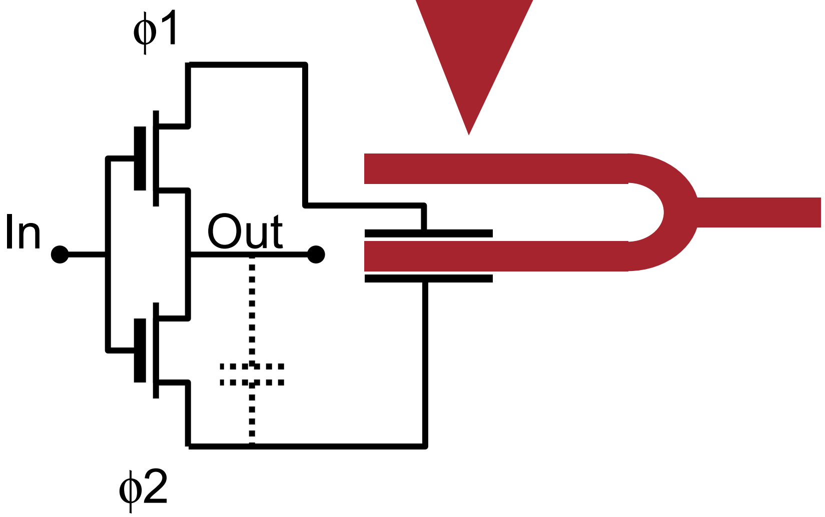
Signal energy must be greater than ~$100\ k_BT$ to avoid spontaneous glitches. To change a bit, convert energy to heat.



“Recycling” Power

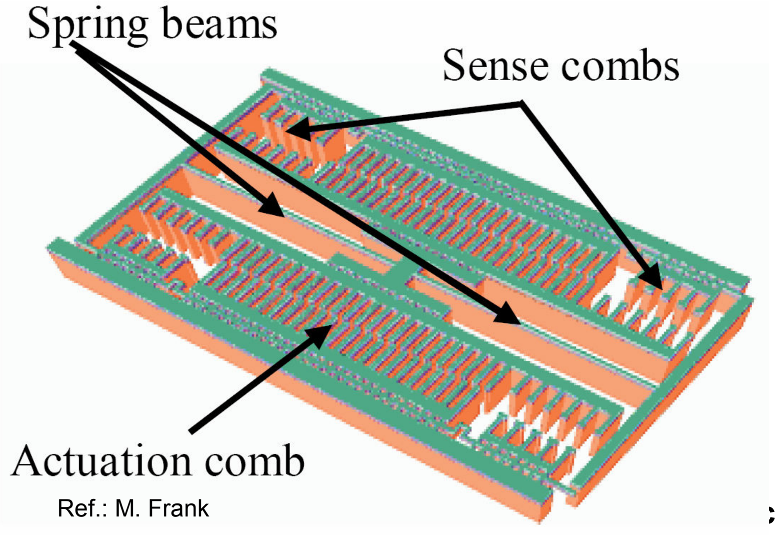
- The $100k_BT$ limit appears unbeatable, but the energy can be “recycled”
- Diagram shows a “SCRL” circuit with regular transistors
- Power comes through a largely loss less resonant device (tuning fork)
- No apology offered for the mechanical device; this is the price of progress

Signal energy must be greater than ~100 k_BT to avoid spontaneous glitches. However, signal energy is recycled by tuning fork



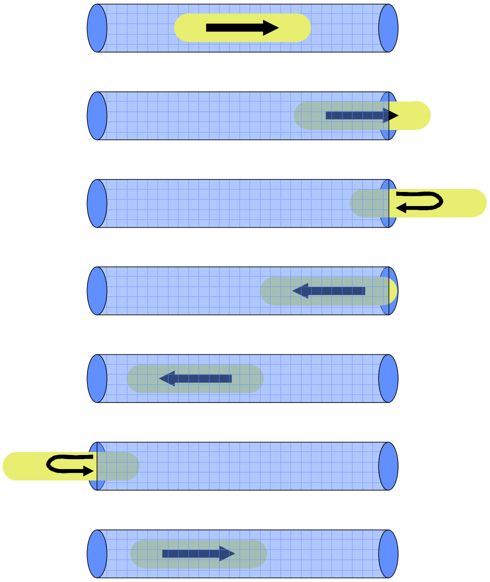
Resonant Clocks

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



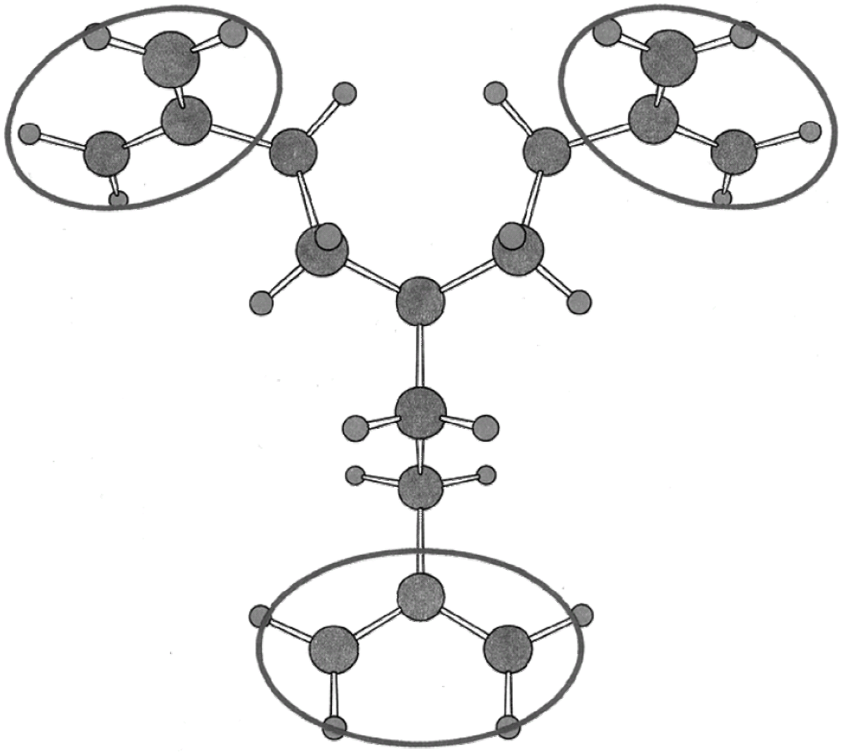
Ref.: M. Frank

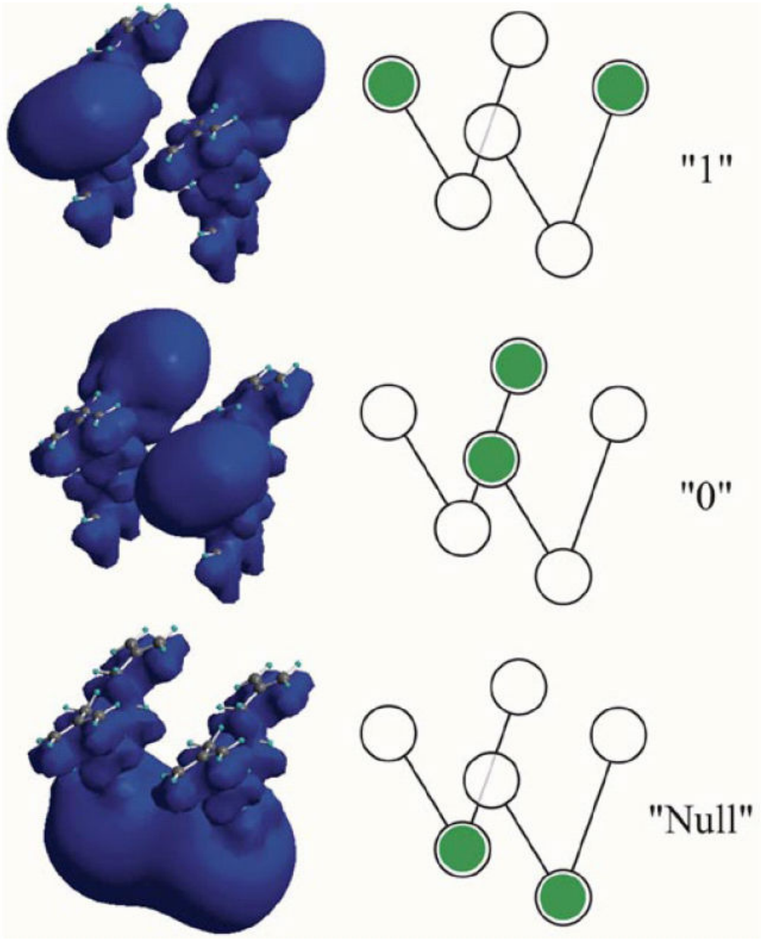
- Carbon Nanotube
  - Simulated to 50 GHz but not known how to fabricate at present



A New Computing Device: Quantum Dots

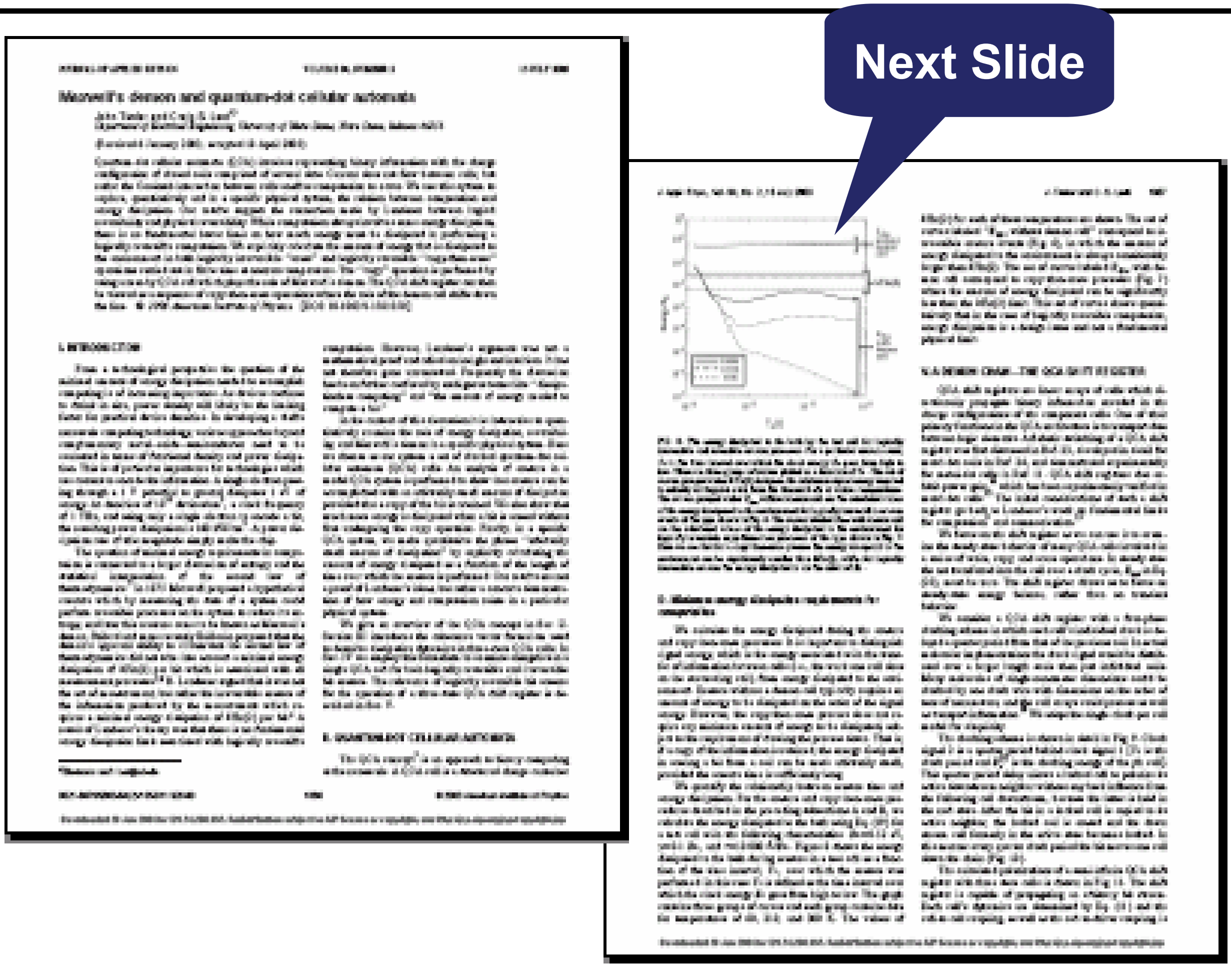
- Pairs of molecules create a memory cell or a logic gate





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

Upside Potential of Quantum Dots

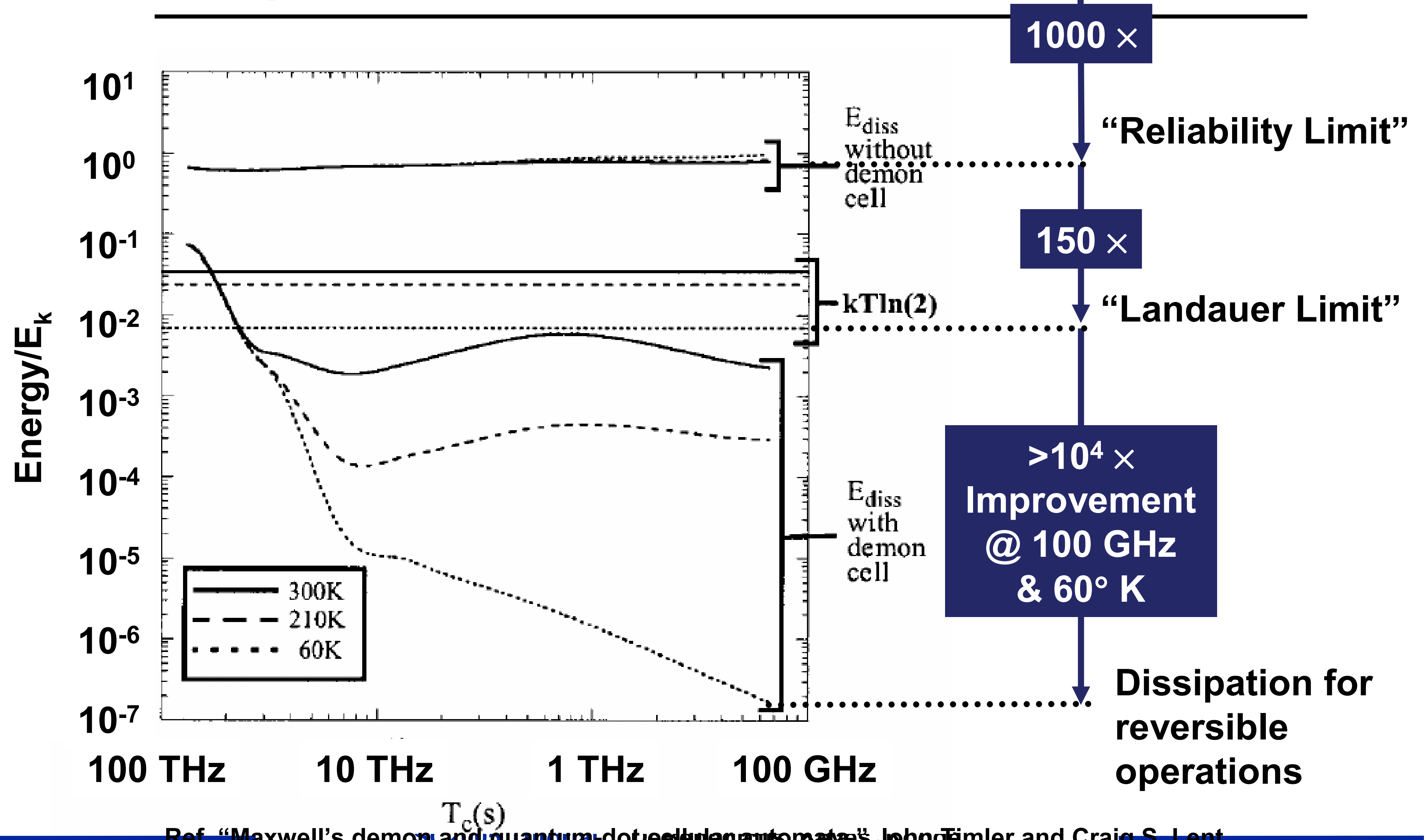


Next Slide

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

2004 Device Level

Upside Potential of Quantum Dots



1000 ×

“Reliability Limit”

150 ×

“Landauer Limit”

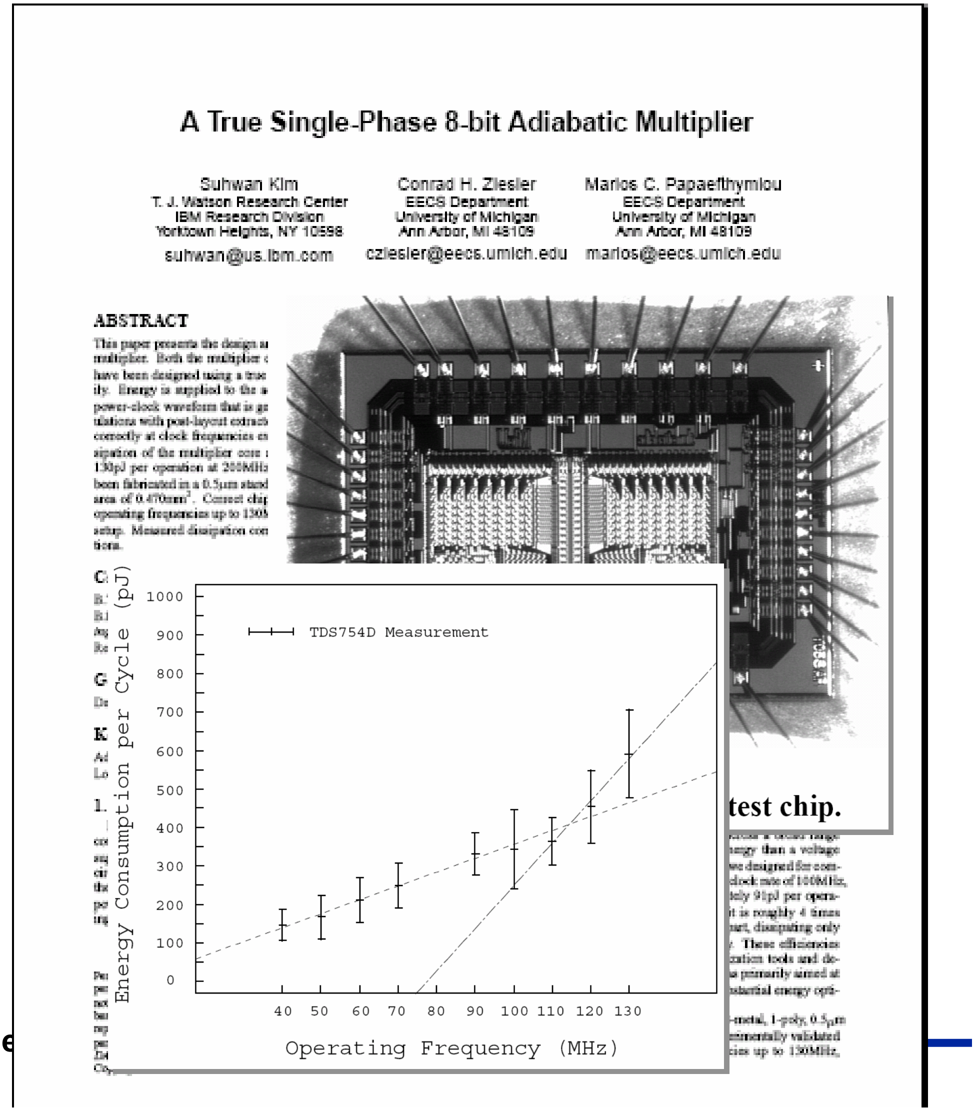
>10^4 × Improvement @ 100 GHz & 60° K

Dissipation for reversible operations

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 a
multiplier. Both the multiplier c
have been designed using a true
ily. Energy is supplied to the a
power-clock waveform that is ge
ulations with post-layout extract
correctly at clock frequencies ex
sipation of the multiplier core a
130pJ per operation at 200MHz
been fabricated in a 0.5μm stand
area of 0.470mm². Correct chip
operating frequencies up to 130M
setup. Measured dissipation con
tions.

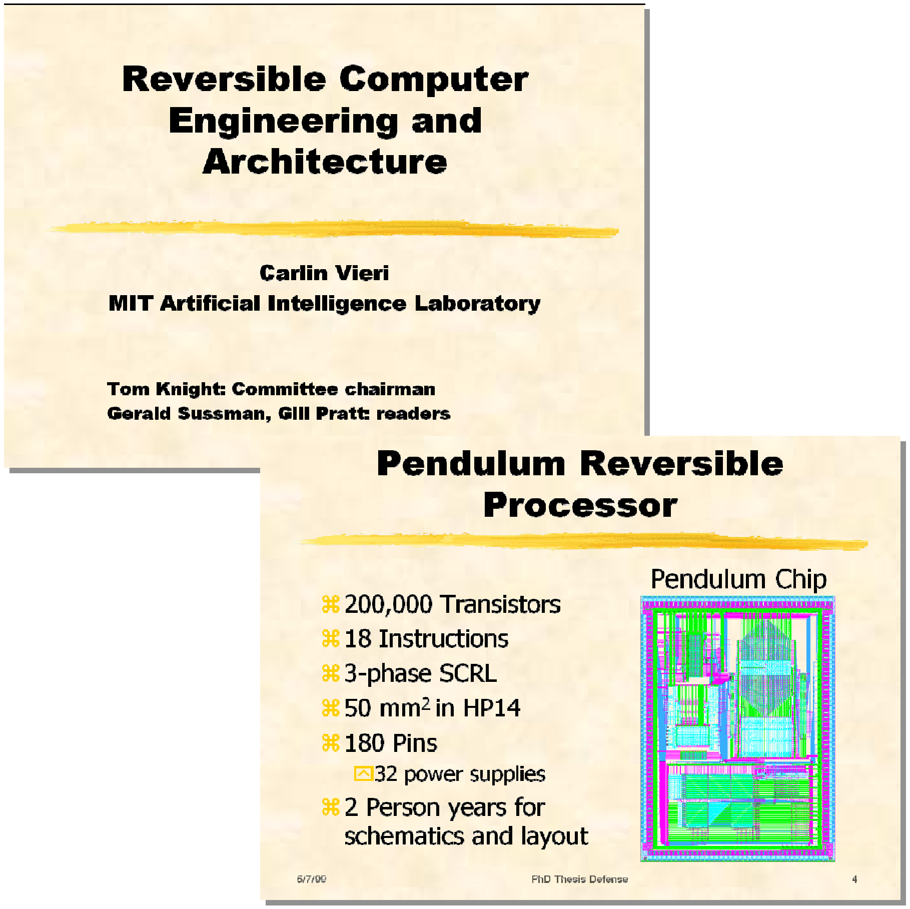
test chip.

ergy than a voltage
we designed for con-
clock rate of 100MHz,
tely 91pJ per opera-
it is roughly 4 times
part, dissipating only
. These efficiencies
cation tools and de-
s primarily aimed at
stantial energy opti-

-metal, 1-poly, 0.5μm
rimentally validated
cies 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 =



Reversible Computer Engineering and Architecture

Carlin Vieri
MIT Artificial Intelligence Laboratory

Tom Knight: Committee chairman
Gerald Sussman, Gill Pratt: readers

Pendulum Reversible Processor

- ⌘ 200,000 Transistors
- ⌘ 18 Instructions
- ⌘ 3-phase SCRL
- ⌘ 50 mm^2 in HP14
- ⌘ 180 Pins
  - 32 power supplies
- ⌘ 2 Person years for schematics and layout

Pendulum Chip

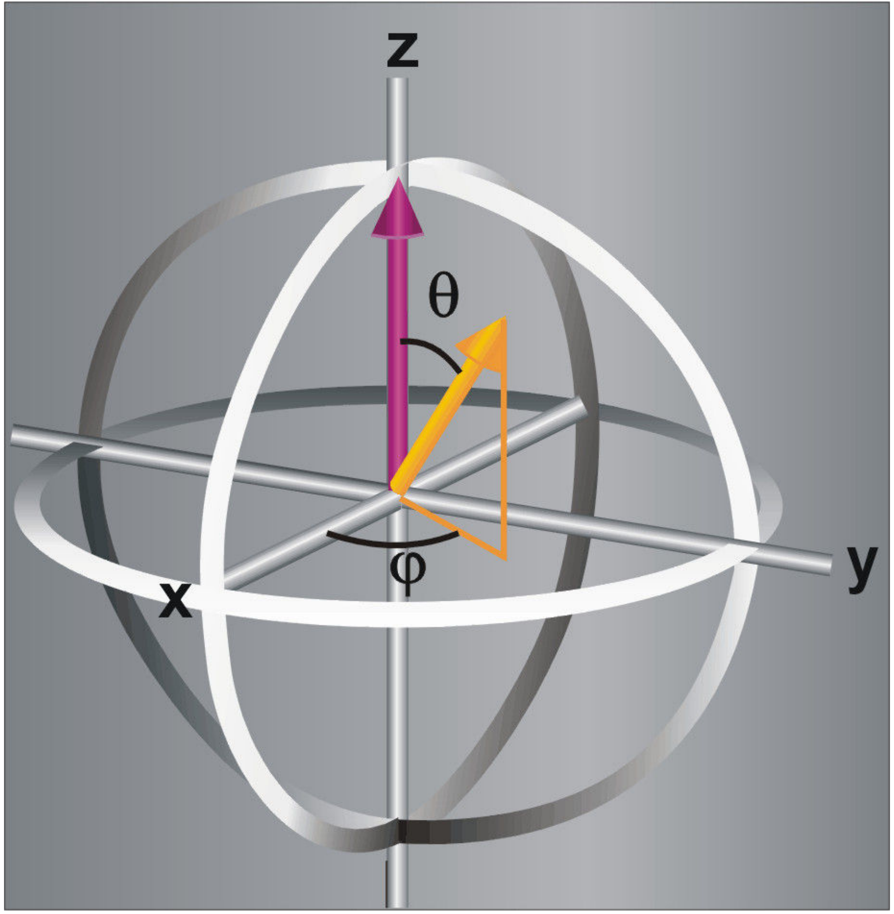
5/7/99 PhD Thesis Defense 4

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

Why Quantum Computing is Interesting

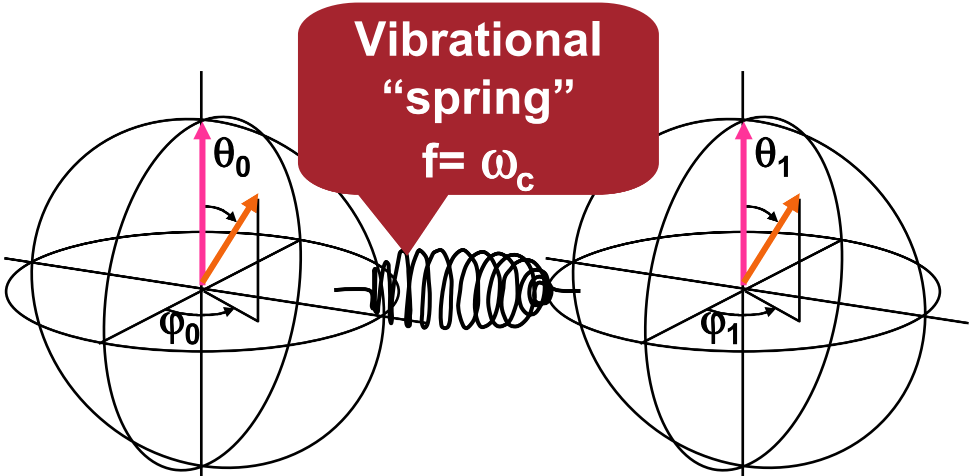
- A Superset of Digital
  - Spin “up” is a 1
  - Spin “down” is a 0
  - Other spins
    - Sidewise
    - Entangled
    - Phase
  - Like wildcards
    - 1011??????
    - Up to 2^N states → in “quantum parallel”



Ion Trap Quantum Gates

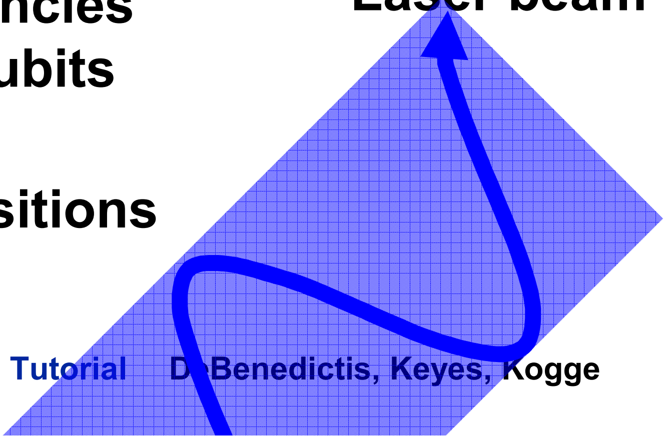
- Hyperfine (internal qubit) frequencies are ω_0 and ω_1
- Vibrational center of mass frequency is ω_c
- Laser at frequency $\omega_0 \pm \omega_c$ or $\omega_1 \pm \omega_c$ couples qubit from hyperfine state to vibrational state and back
- Appropriate frequencies selectively move qubits based on data
- Works on superpositions

- Two ions in an ion trap



Vibrational "spring" $f= \omega_c$

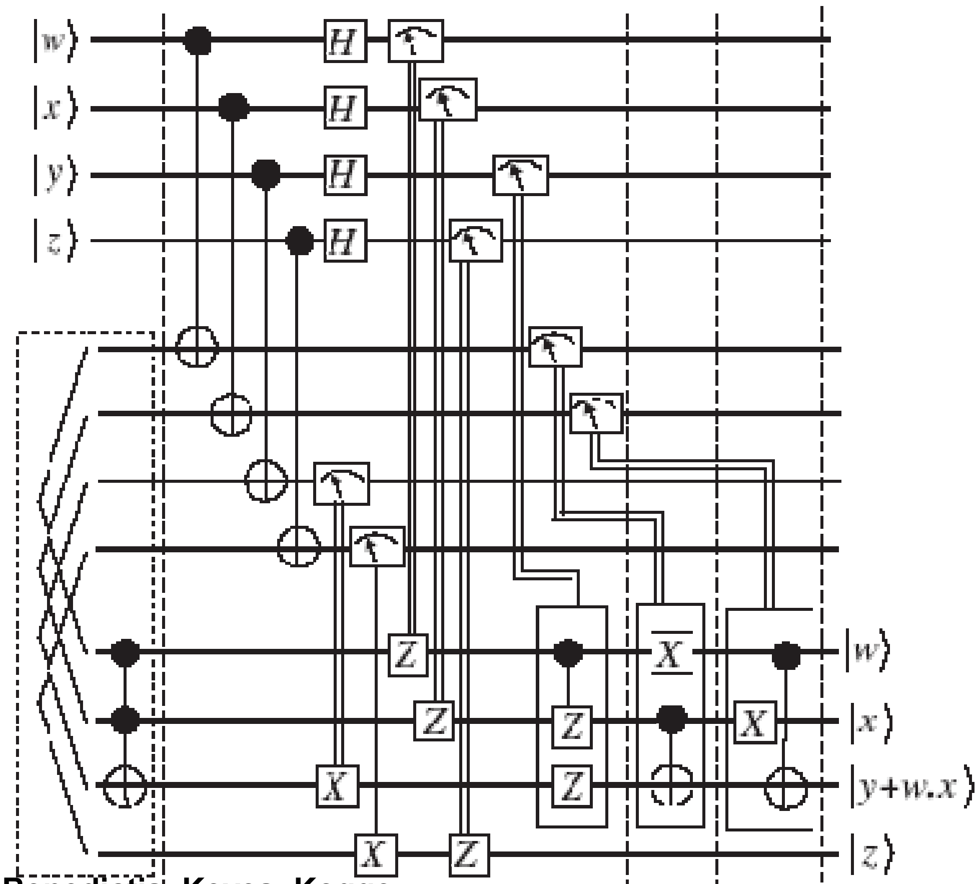
- Laser beam frequency ω



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, Ibinson, B, quant-ph/0311014

Beyond Transistors

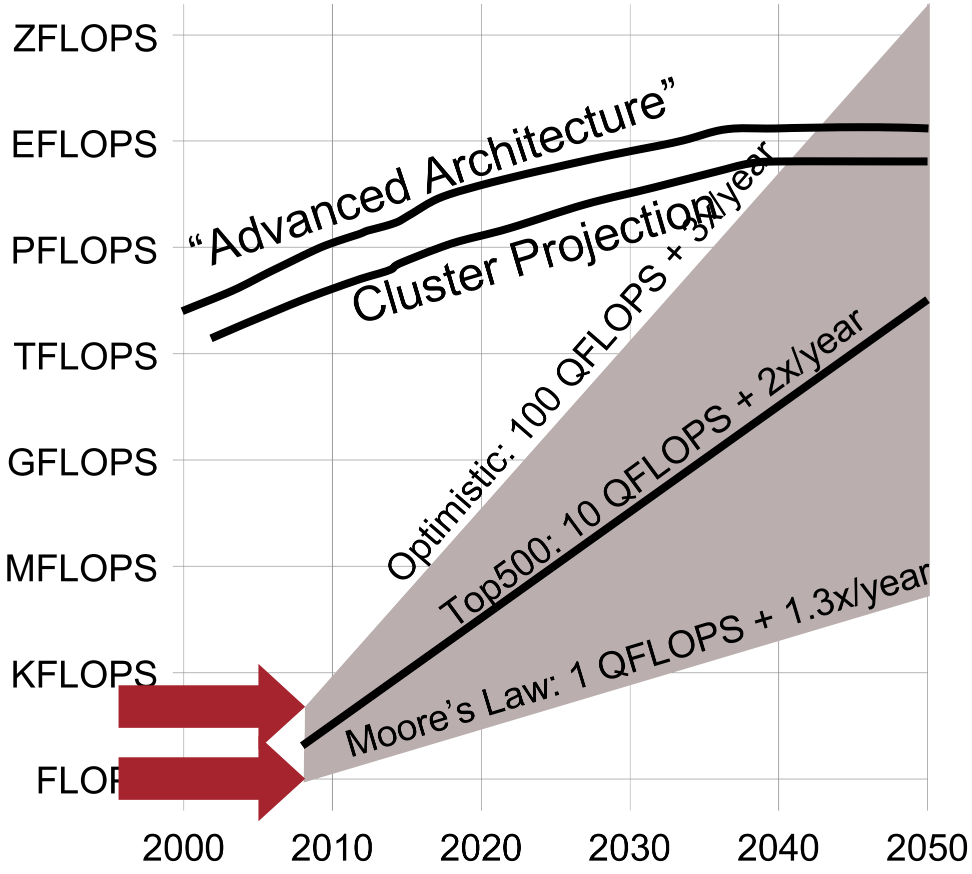
- Applications Requirements
- Thermodynamic limits to total power
  - Superconducting logic and Carnot cycle
- Upside potential of advanced architectures/PIM
- Some nanotech technologies on the horizon
- Reversible logic may defeat thermodynamic limitations
- Upside potential of quantum computing
  - 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

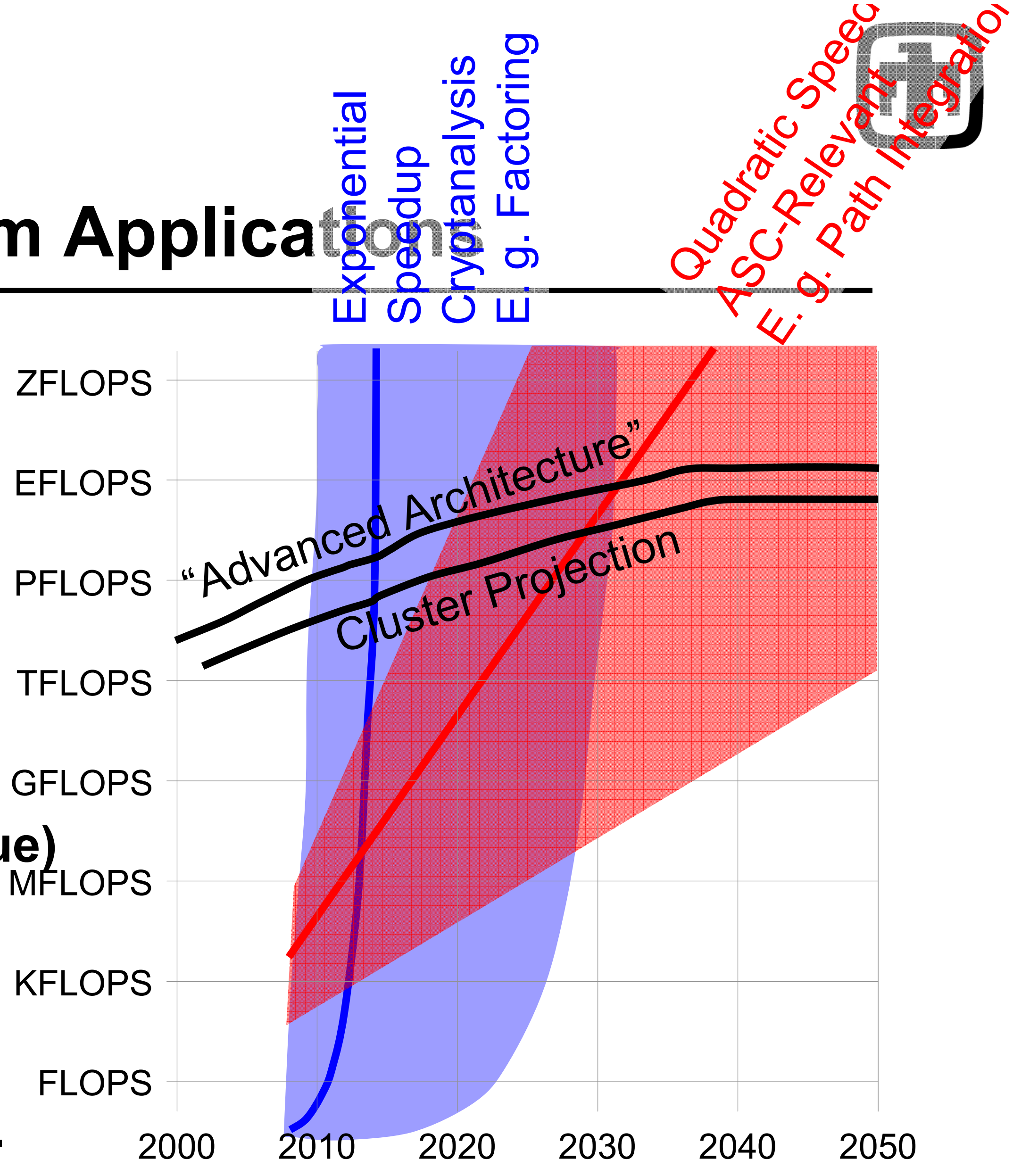
- There appears to be an engineering case for quantum computers of 1-100 Q-FLOPS
- One would expect an exponential growth rate for quantum computers similar to Moore's Law, but the rate constant is impossible to predict, so three possibilities have been graphed



Ref. "How to build a 300 bit, 1 Gop quantum computer," Andrew M. Steane, Clarendon Laboratory, UK, quant-ph/0412165

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. Actinide chemistry)**



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

Nanotech, Architecture, and Memory Wall

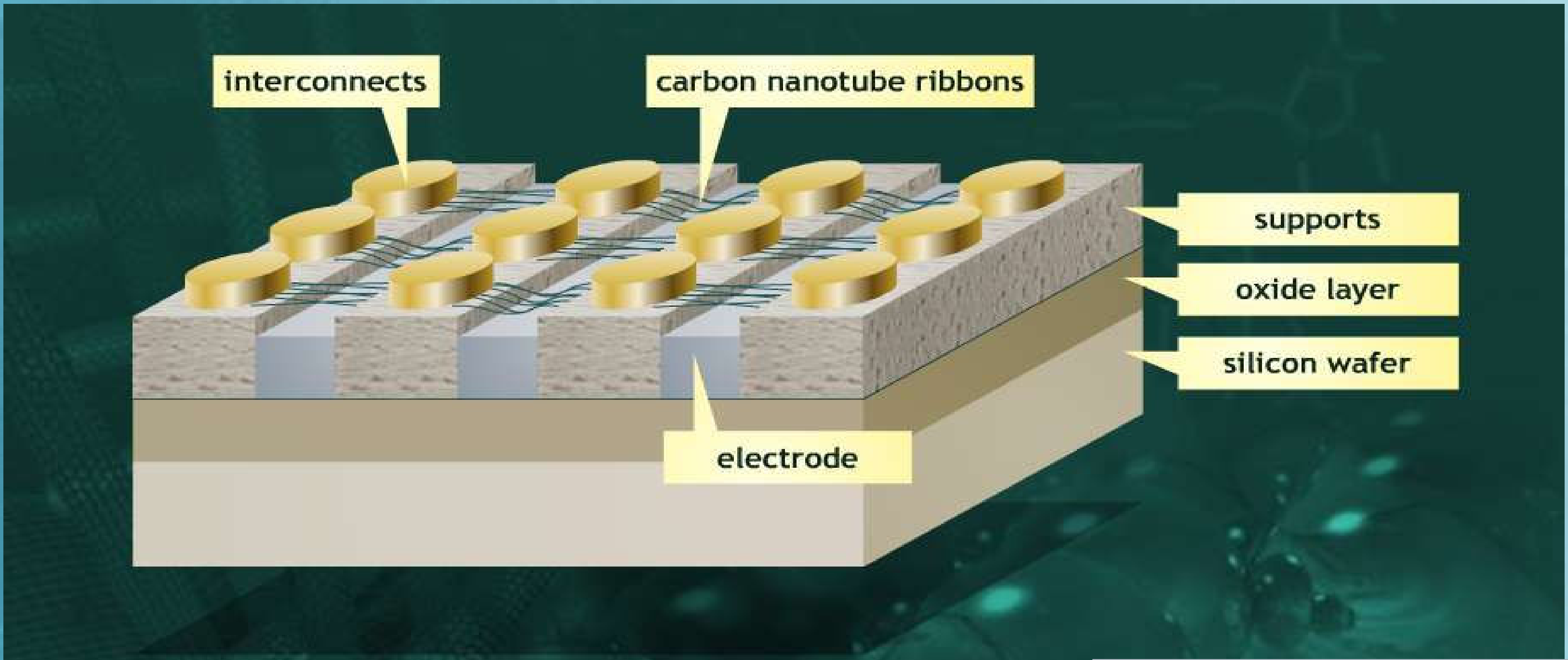
- There are many paths to future architectures, yet one looks especially likely to appear in a ~5 years
  - Logic per ITRS roadmap for transistors
  - Nanotech memory
    - Cleverly embedded
    - Multiple options
  - Architecture per continuation of “multi-core” trend
- Resulting computers would be of recognizable architecture, but more parallelism.
  - I believe the increase in parallelism will cause a crisis.

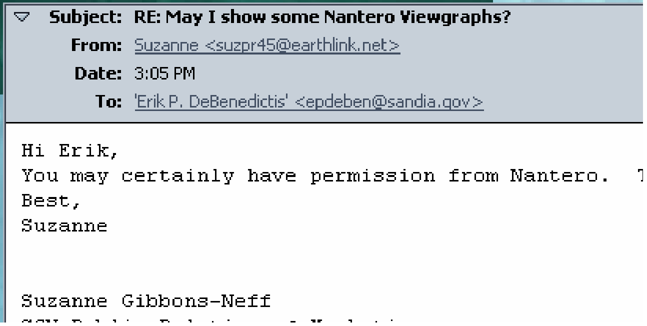
Nanotech Memory

- Common Feature
  - Some new device structure that holds information
  - CMOS process compatibility, typically through additional layers

- Many options
  - We'll review carbon nanotube arrays in the next few slides
  - We'll look at a table with other options

Nantero NRAM™ Device





Subject: RE: May I show some Nantero Viewgraphs?

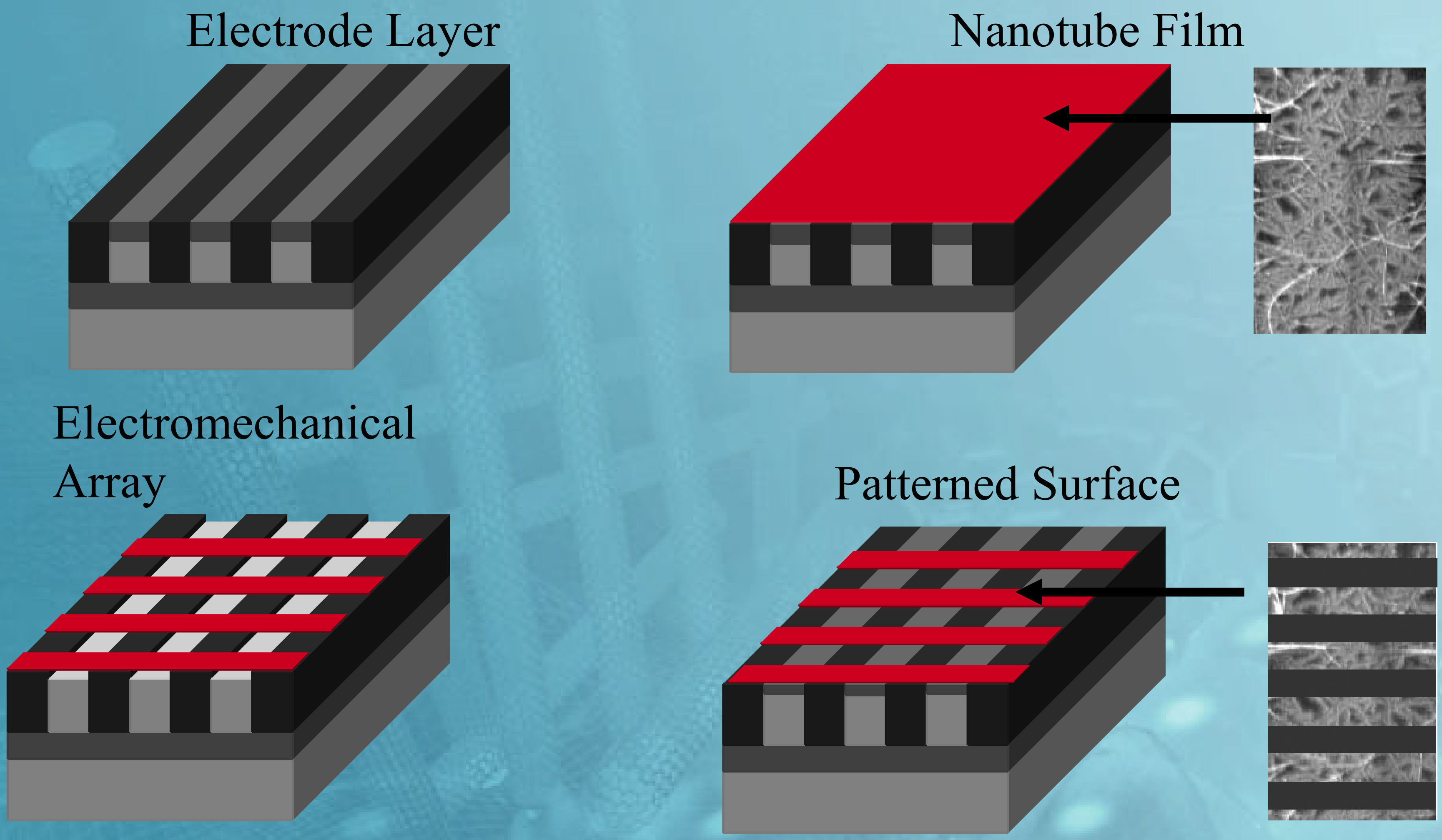
From: Suzanne <suzpr45@earthlink.net>

Date: 3:05 PM

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

```
Hi Erik,
You may certainly have permission from Nantero.
Best,
Suzanne


Suzanne Gibbons-Neff
```



Electrode Layer

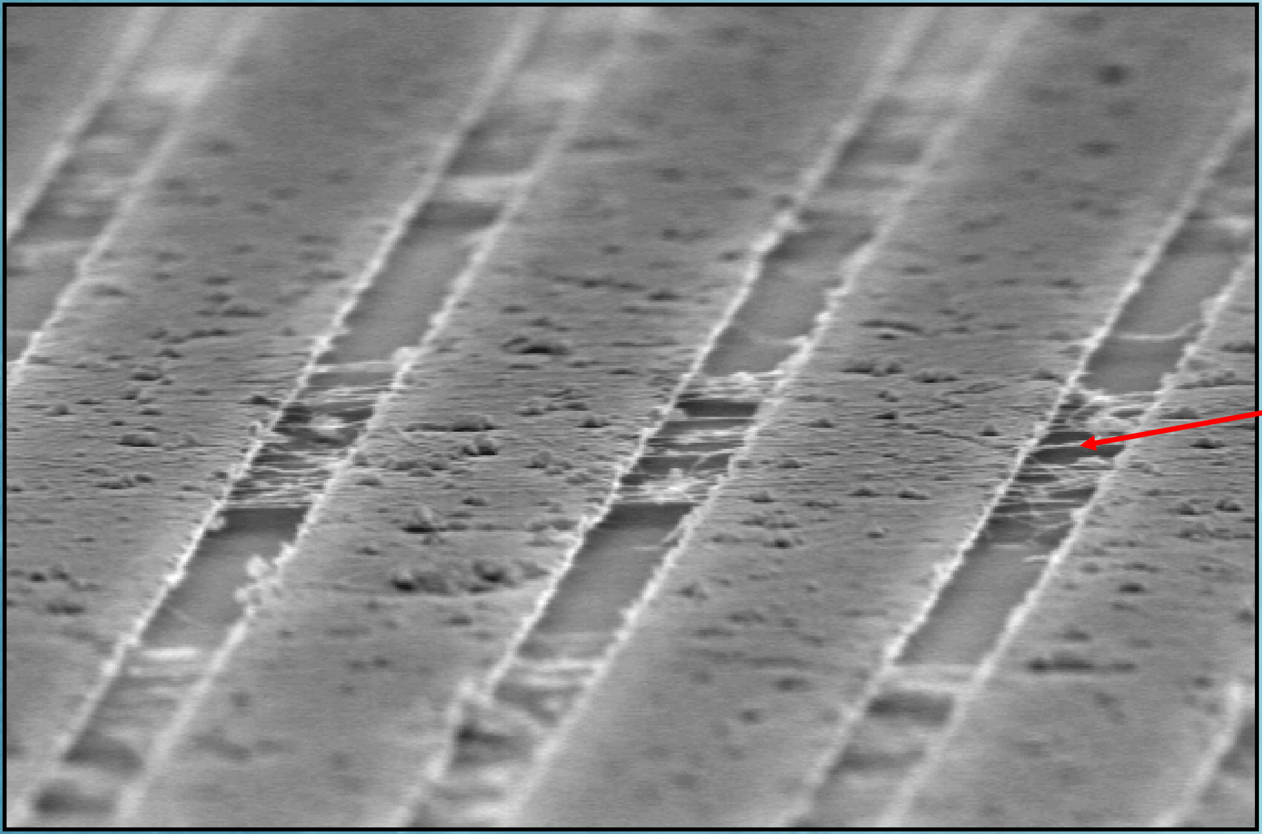
Nanotube Film

Electromechanical Array

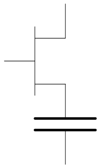
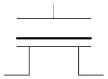
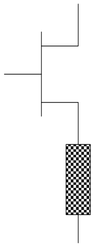
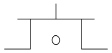
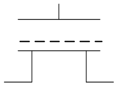
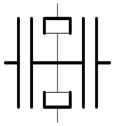
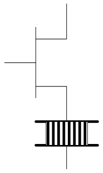
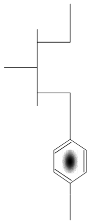
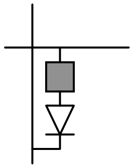
Patterned Surface

Nantero's Collaboration with ASML

- Proof of compatibility between equipment and nanotube process
- Creation of very-high-density bit arrays using 250nm stepper

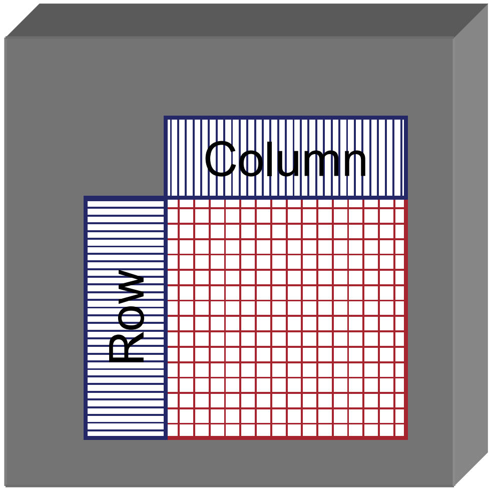


Nanotubes

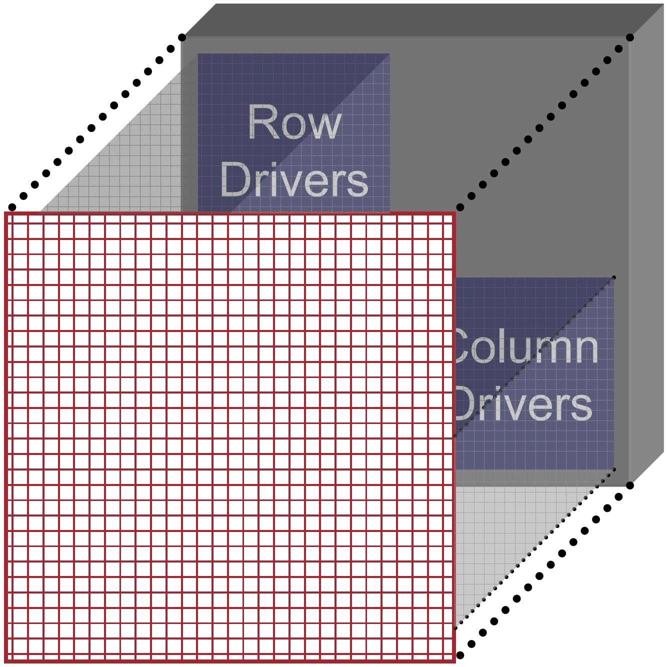
	Present Day Baseline Technologies		*Phase Change Memory**	*Floating Body DRAM*	*Nano-floating Gate Memory***	*Single/Few Electron Memories***	*Insulator Resistance Change Memory***	*Molecular Memories***	**Unipolar switching Memories**
Storage Mechanism									
Device Types	DRAM	NOR Flash	OUM	1TDRAM eDRAM	Engineered tunnel barrier or nanocrystal	SET	MIM oxides	Bi-stable switch Molecular NEMS	**Cross point**
Availability	2004	2004	~2006	~2006	~2006	>2007	~2010	>2010	**>2008**
Cell Elements	1T1C	1T	1T1R	1T	1T	1T	1T1R	1T1R	**1D-1R**

Nanoarray Architecture

- Low Road
  - Planar, conventional architecture

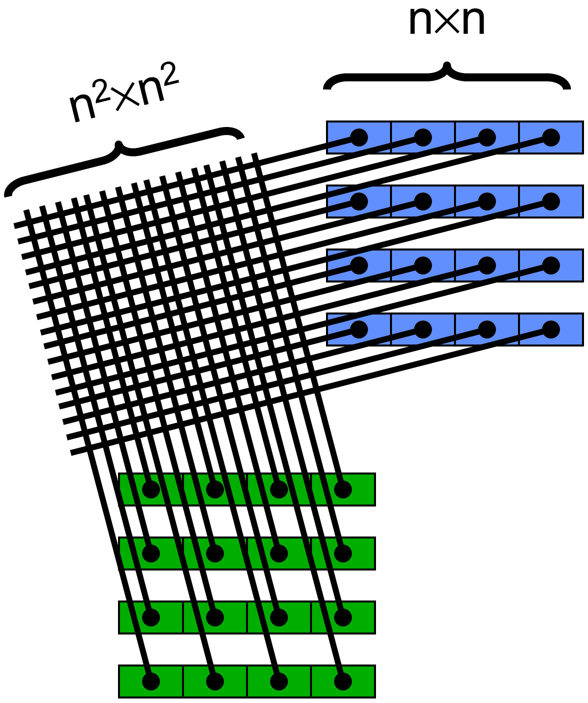


- High Road
  - Fabricate nanotech array on top of chip



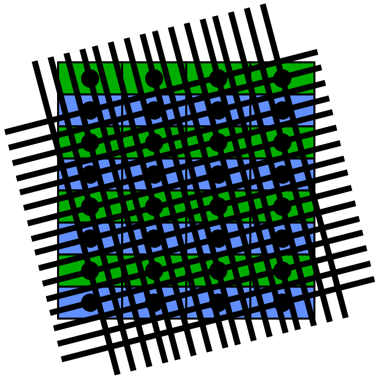
Thought Experiment – Skewed Nanoarray

- 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



Nano Memory Conclusions

- There seem to be a host of proposals for nano memory
- Some of these will appear in the next year
- The technologies tend to retain data with power off
- The technologies are pretty fast – DRAM speed or better

- Densities based on a cell with dimensions
  - Line-space × line-space
  - × sub lithographic linewidth
- 1 cm × 1 cm chip (@ $6F^2$)
  - 180 nm → 60 MBytes
  - 65 nm → 500 MBytes
  - 22 nm → 4 GBytes
  - 10 nm → 20 GBytes
- Multiple layers possible

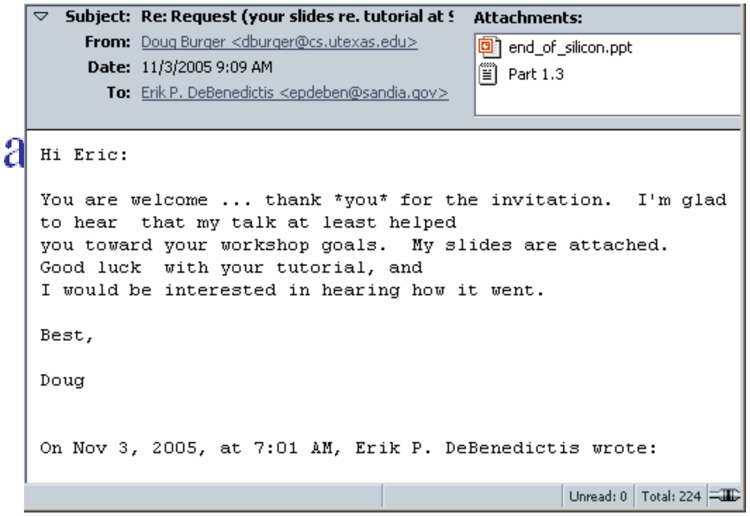
Architecture Trends

- Memory wall will disappear
  - If you can live with 360 MBytes-116 GBytes memory per chip (previous slide)
- Peak thread speed will grow more slowly than we like
- Power per gate-operation will level out (ugh) at thermodynamic limit
- Efficiency of architecture in converting power to FLOPS may be subject to improvement
- Chip-to-chip interconnect speeds difficult to predict at present

Architectures at the End of Silicon: Performance Projections and Promising Paths

Frontiers of Extreme Computing
October 24, 2005

Doug Burger
The University of Texas a



Subject: Re: Request (your slides re. tutorial at
From: Doug Burger <dburger@cs.utexas.edu>
Date: 11/3/2005 9:09 AM
To: Erik P. DeBenedictis <epdeben@sandia.gov>

Attachments:
end_of_silicon.ppt
Part 1.3

```
Hi Eric:

You are welcome ... thank *you* for the invitation.  I'm glad
to hear  that my talk at least helped
you toward your workshop goals.  My slides are attached.
Good luck  with your tutorial, and
I would be interested in hearing how it went.

Best,

Doug


On Nov 3, 2005, at 7:01 AM, Erik P. DeBenedictis wrote:
```

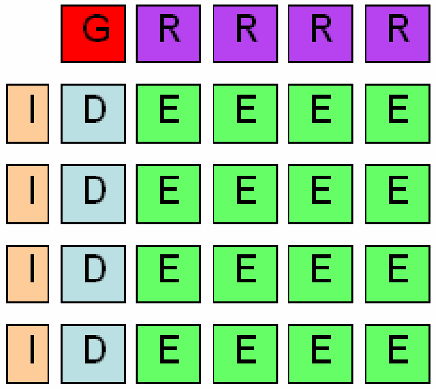
Burger’s Architecture, Erik’s Example

- Produce lots of puny cores that can be used individually or ganged together
- Roughly, n cores will have
  - n×power dissipation
  - n×memory
  - log n performance

This is no joke. Imposes huge win for parallelism in code

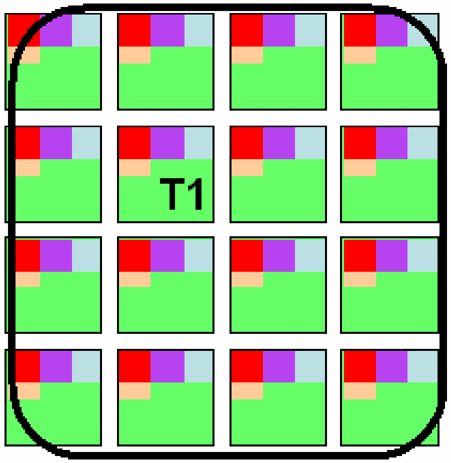
- Why does Erik show this as an example?
  - It seems to exemplify all the needed new features in proper balance
  - Practical systems may be a linear combination of Burger’s architecture and present-day ones
  - Also, future PCs may end up heterogeneous
    - Integrated graphics, …

Multigranular "Elastic" Threads



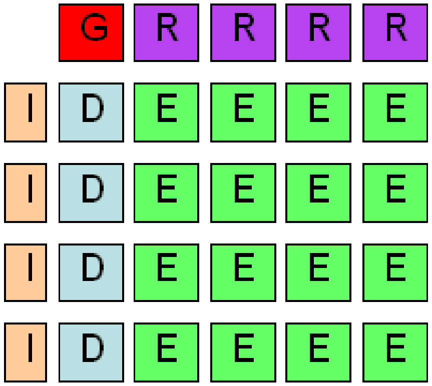
	G	R	R	R	R
I	D	E	E	E	E
I	D	E	E	E	E
I	D	E	E	E	E
I	D	E	E	E	E

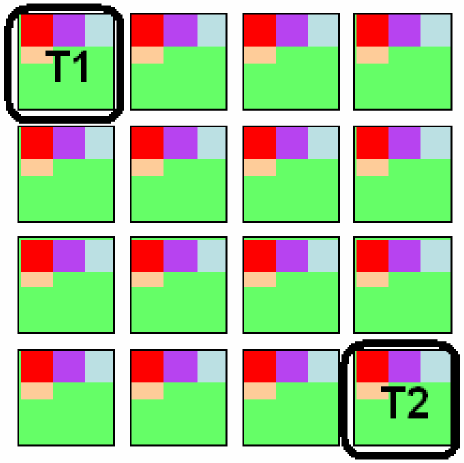
- Problems with TRIPS microarchitecture
  - Limited register/memory bandwidth
  - Number of tiles per core is fixed at design time
  - Multithreading is a hack to vary granularity
- Achievable by distributing all support tiles
  - Assume each tile can hold >= 1 block (128 insts.)
- Solutions being implemented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



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

Multigranular "Elastic" Threads

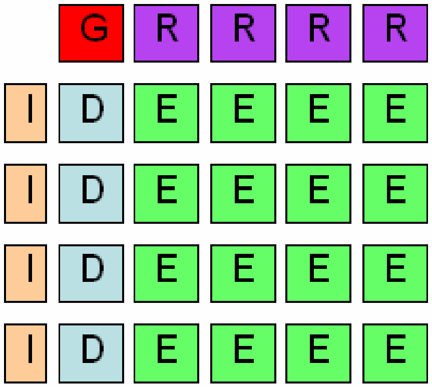




- Problems with TRIPS microarchitecture
  - Limited register/memory bandwidth
  - Number of tiles per core is fixed at design time
  - Multithreading is a hack to vary granularity
- Achievable by distributing all support tiles
  - Assume each tile can hold >= 1 block (128 insts.)
- Solutions being implemented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

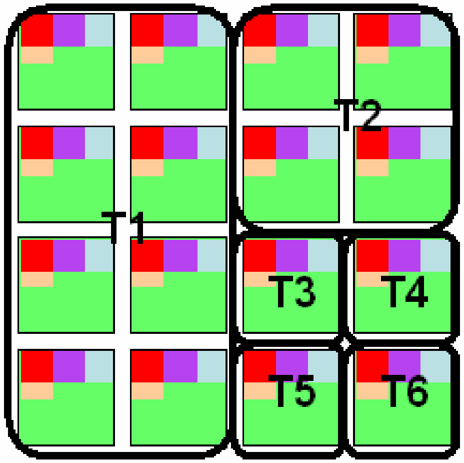
- **Config one: 1 thread, 16 blocks @ 8 insts/tile**
- **Config two: 2 threads, 1 block @ 128 insts/tile**
- **Config three: 6 threads, 1 thread on 8 tiles, 1 thread on 4 tiles, 4 threads on 1 tile each**

Multigranular "Elastic" Threads



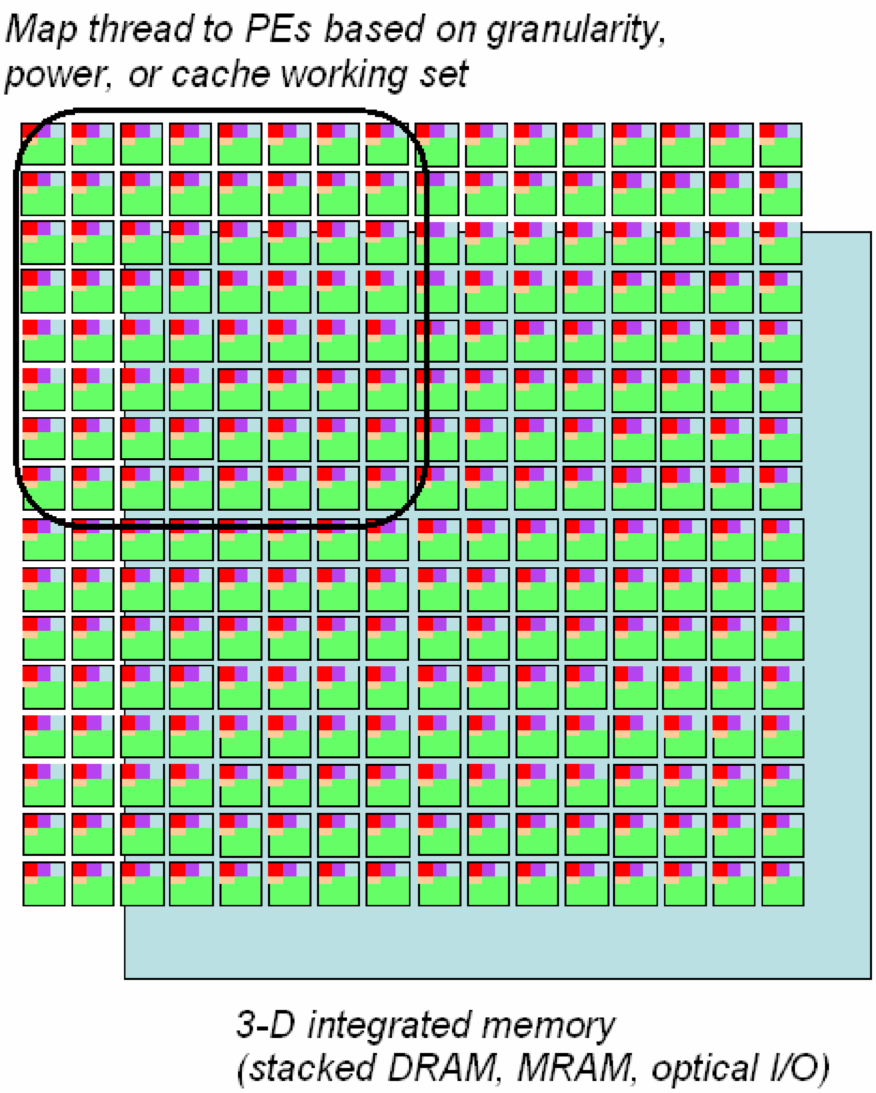
	G	R	R	R	R
I	D	E	E	E	E
I	D	E	E	E	E
I	D	E	E	E	E
I	D	E	E	E	E

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



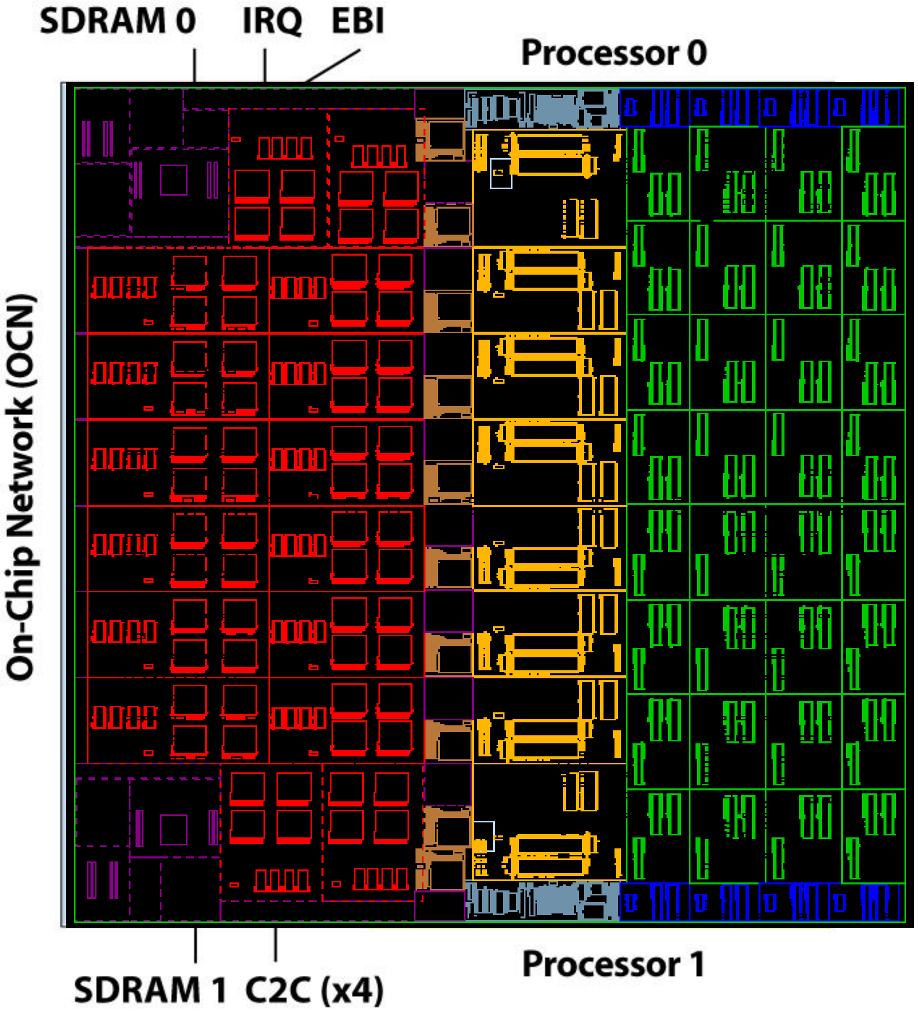
- **Config one: 1 thread, 16 blocks @ 8 insts/tile**
- **Config two: 2 threads, 1 block @ 128 insts/tile**
- **Config three: 6 threads, 1 thread on 8 tiles, 1 thread on 4 tiles, 4 threads on 1 tile each**

Looking forward


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?

Floorplan of First-cut Prototype



TRIPS Tiles and Interfaces

G: Processor control - dispatch, next block predictor, commit
R: Register file - 32 registers x 4 threads, register forwarding
I: Instruction cache - 16KB, 16-entry TLB variable-size pages
D: Data cache - 8KB, 64-entry load/store queue, 16-entry TLB
E: Execution unit - 128 reservation stations, integer/FP ALUs
M: Memory - 64KB, OCN router with 4 virtual channels
N: OCN network interface - OCN router, PA translation
DMA: Direct memory access controller
SDC: SDRAM controller
EBC: External bus controller (to PowerPC)
C2C: Chip-to-chip network links - to four neighbors

IRQ: Interrupt request - service request to PowerPC
EBI: External bus interfaces - command interface from PPC

Stacked Memory & Rack Parameters

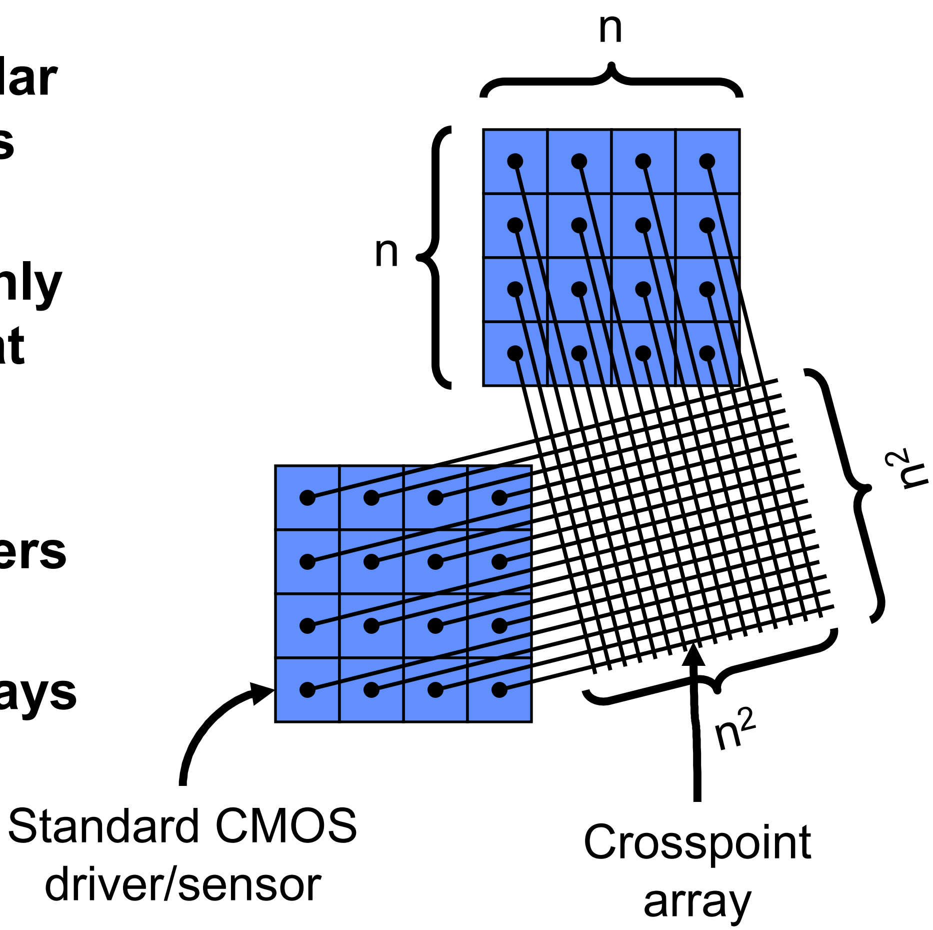
- Say we stack NRAM on top of a CPU chip like Burger proposes
- Arithmetic on amount of NRAM
  - 35 nm ½-pitch
  - chip is 1.5 cm x 1.5 cm
  - Bit cell is 2 x 2 linewidths or 4 x 4 ½-pitches
  - This would be (.015 m chip edge)^2/(35nm ½ pitch)^2/(16 sq ½-pitches/cell)/(8 bits/byte)/(10^9 bytes/GByte) = 1.5 GBytes

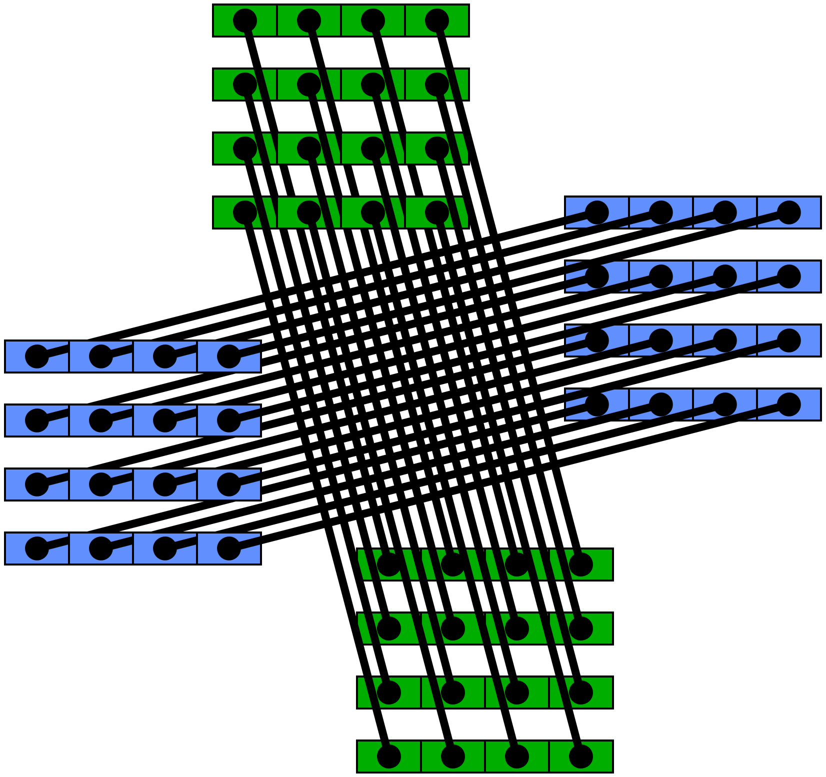
Stacked Memory & Rack Parameters

- Rack architecture (limited by 10 KW dissipation or 100 chips)
  - 150 GBytes "on chip" memory divided into 100 modules of 1.5 GBytes (how much external memory needed?)
  - 100 256-way SMPs – total 25,000 processors (but "flexible mapping" possible to give appearance of fewer processors with more memory each)
  - 200 Tflops peak/rack
  - Memory bandwidth: Not specifically limited due to PIM architecture

Thought Experiment – Skewed Nanoarray

- Problem is that molecular scale mask alignment is very hard
- Solution is to pattern only regular arrays of lines at the molecular scale →
- Diagram to right (from Likharev) uses $2n^2$ drivers to drive n^4 crosspoints
- Published design overlays everything





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

custom

ction”

table

Don’t exceed chip area:
140 mm^2 high volume
280 mm^2 maximum

atts air cooling or
vater cooling

an of CPU here

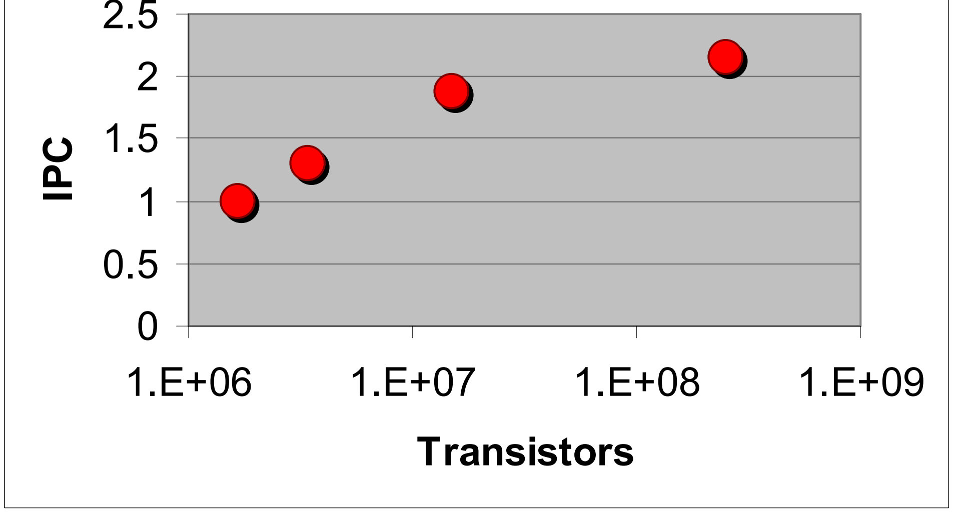
_____ DRAM bytes/chip

↑ Generation at “production” or “introduction” table 1c, 1d, 1e, 1f

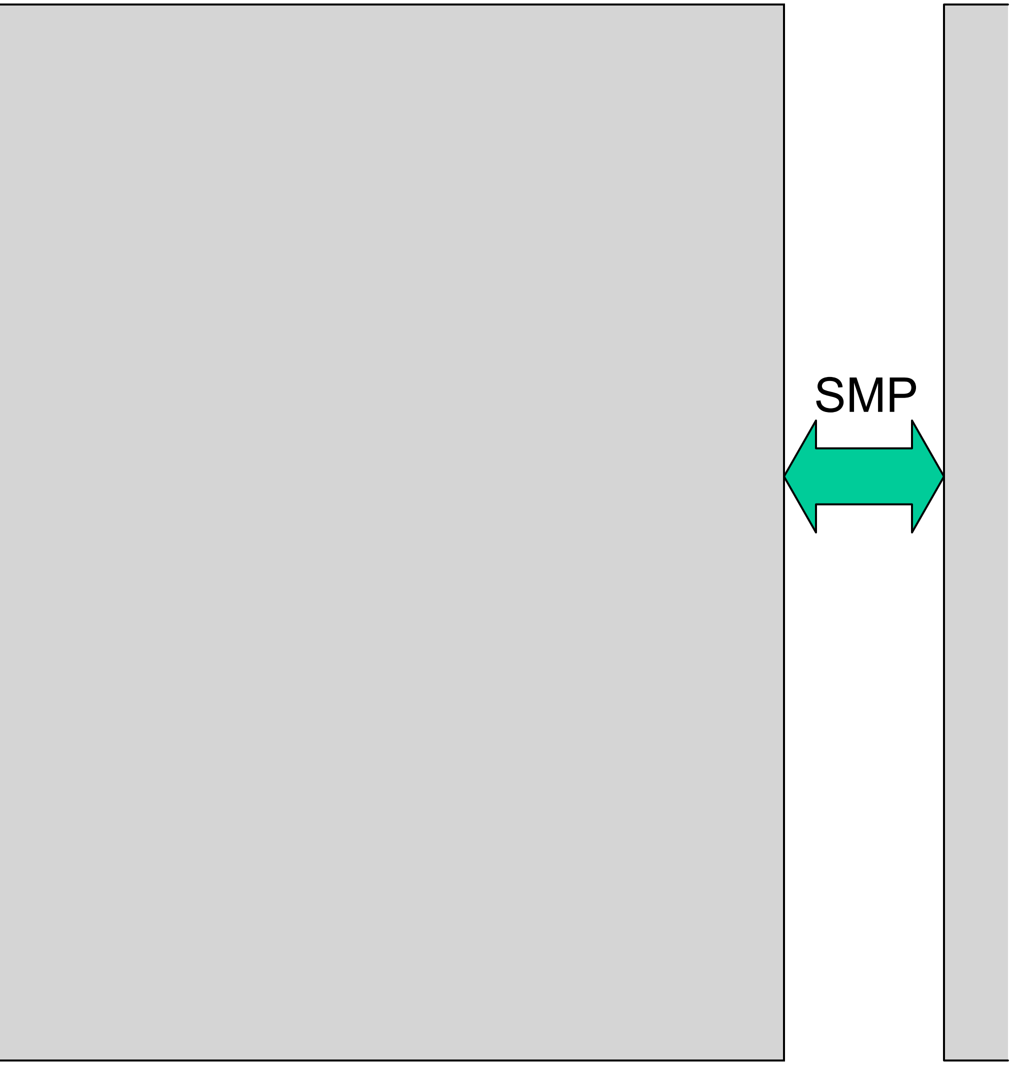
Bandwidth

↓ Table 3a, 3b; note that 1/2to 2/3 of pins are power

_____ # signal pins on chip
_____ internal clock rate
_____ total bandwidth =
_____ memory bandwidth +
_____ SMP bandwidth +
_____ network bandwidth



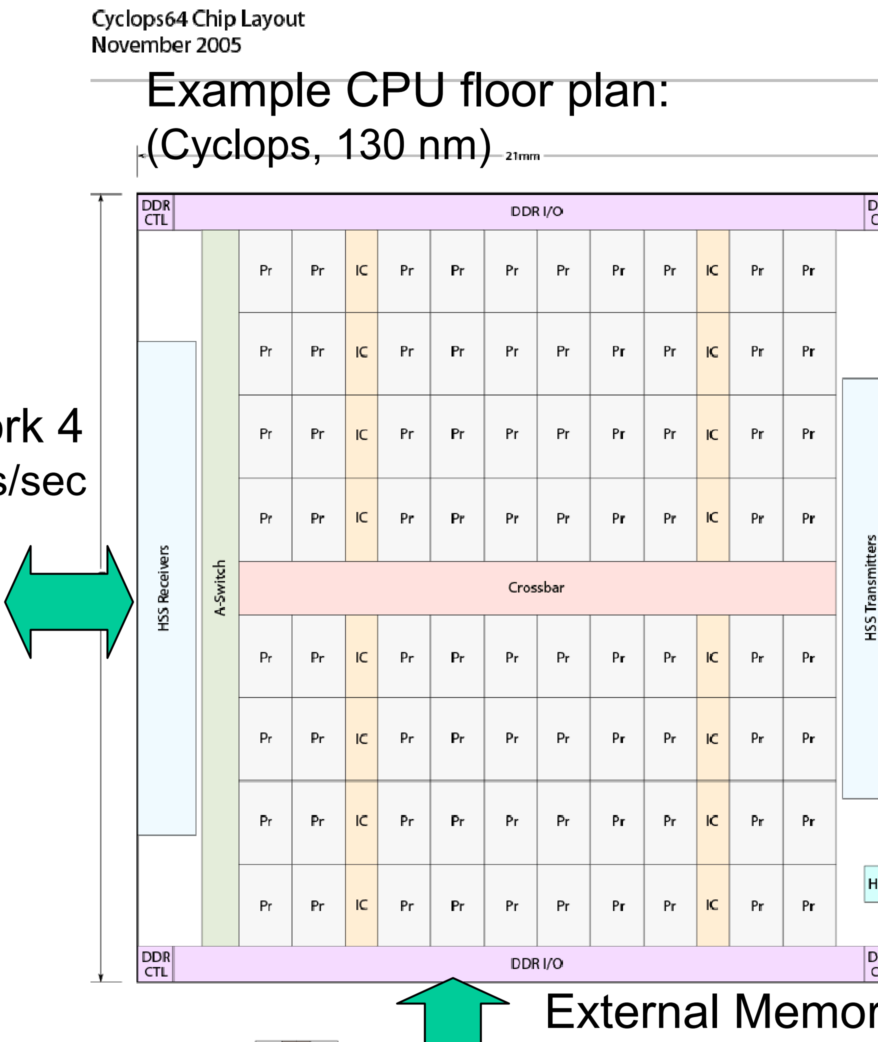
Note: These figures are from 200



↓

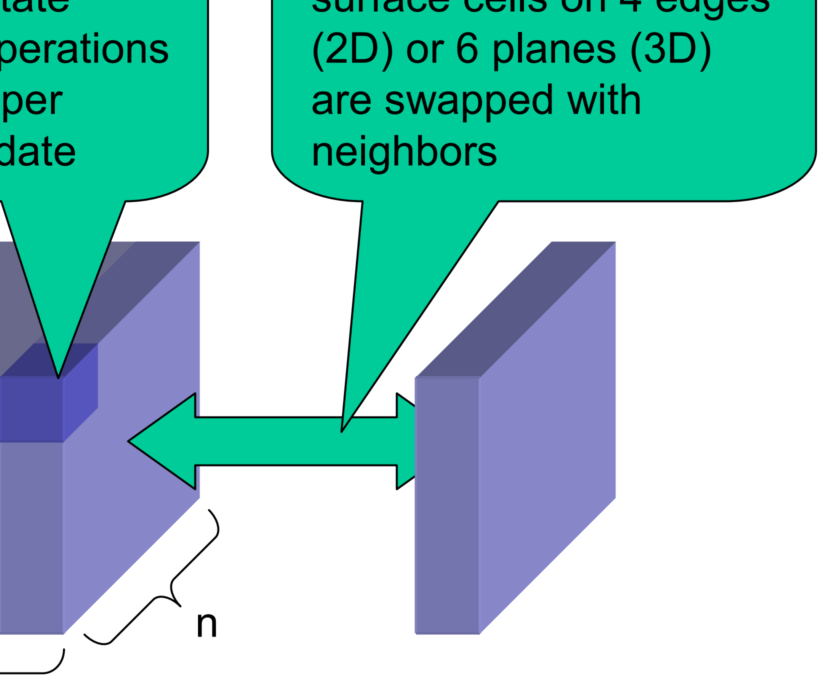
Network 4
GBytes/sec

Example CPU floor plan:
(Cyclops, 130 nm)



External Memor

xternal Memory



tate
perations
per
date

surface cells on 4 edges
(2D) or 6 planes (3D)
are swapped with
neighbors

mory
:
PS
k bandwidth

3 Dimensions
Need $n^3 \times$ K bytes memory
Each timestep max of:
$t_{comp} = n^3 \times G/FLOPS$
$t_{comm} = 6 \times n^2 \times K/\text{link bandwidth}$
$t_{sync} = *$

on:
mize data access bandwidth.
off chip, will it stream properly? The answer to
te time to be estimated
ations time based on data and communications

FLOPS or percentage of peak.

he relative standard deviation of the grind time is
of the concurrent phases becomes relatively
the problem size is small relative to the number

Architecture Plan
Workbook

SC 2005 Tutorial M0

Impact of Moore's Law and Arc
Application Performan

90 nm Linewidth

64 M

Custom Commodity or custom

↑ Choose “production” vs. “introduction” table

Bandwi

1216 #

533M in

20 GBytes/s to

16 GBytes/s m

0 S

4 GBytes/s N

CPU

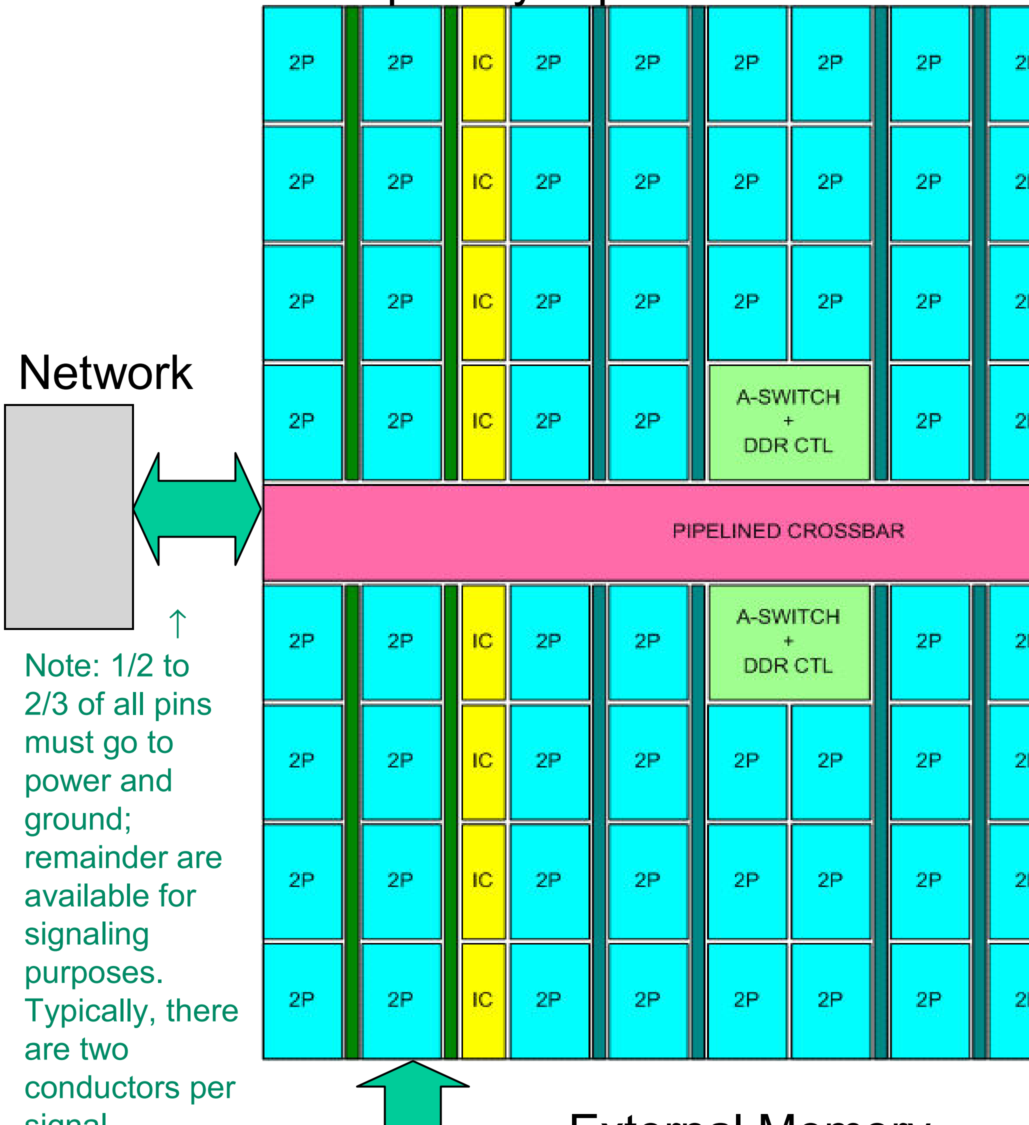
80 cores

64 KBytes cache/core

Don't exceed chip area

150 watts ← 100 Watts ±

Example: Cyclops



↑ Note: 1/2 to 2/3 of all pins must go to power and ground; remainder are available for signaling purposes. Typically, there are two conductors per signal