The *Shape* of Things to Come: Future Trends in HPC Architectures

Peter M. Kogge

Associate Dean for Research

McCourtney Prof. of CS & Engr

University of Notre Dame

IBM Fellow (retired)

My View: Future HPC Evolutionary Paths Are Multiplying

- Today: “Killer Micros” becoming “physics-limited” very hungry multi-core monsters

- Maturing Multi-threading & Tiling providing more nimble systems

- Is there an alternative evolutionary path we’ve ignored?

My Concern: We're Focused on the Wrong Aspect of the Wall

What about bandwidth?

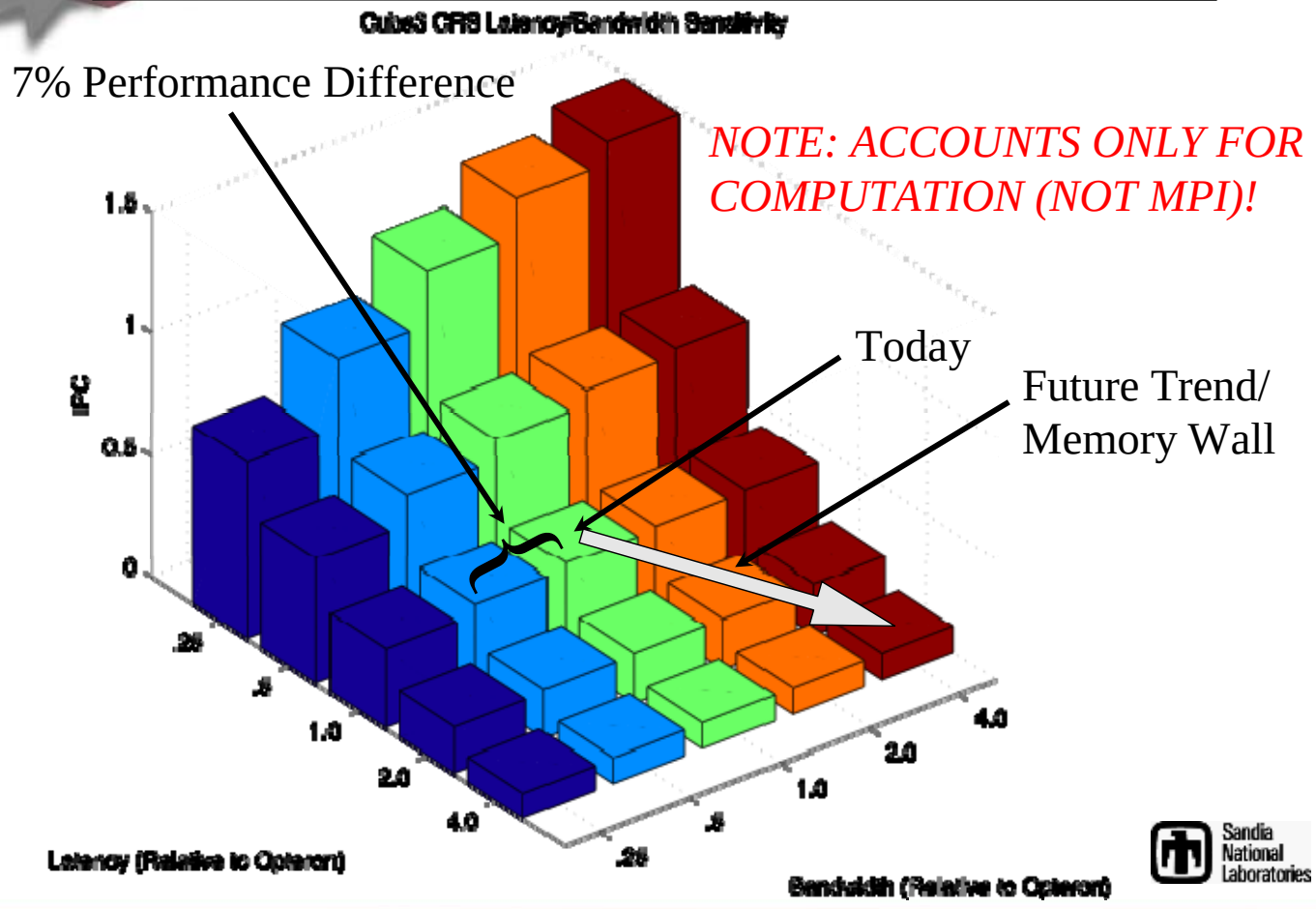
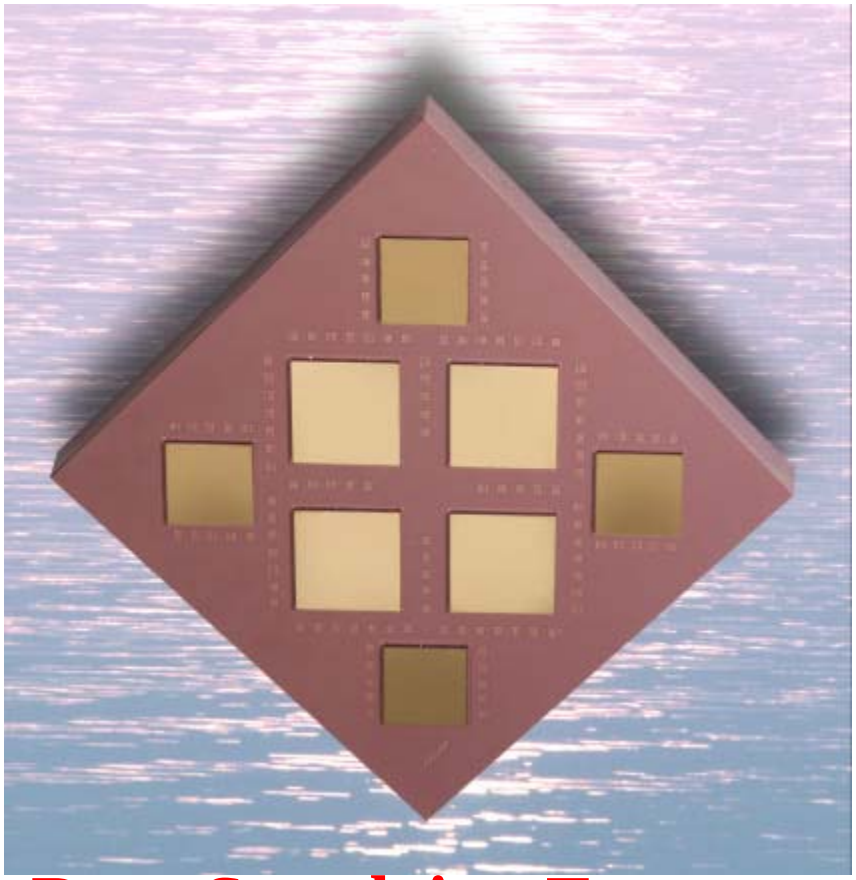


Chart courtesy Richard Murphy, SNL

Application: Trilinos

And Perhaps Missing Another Wall



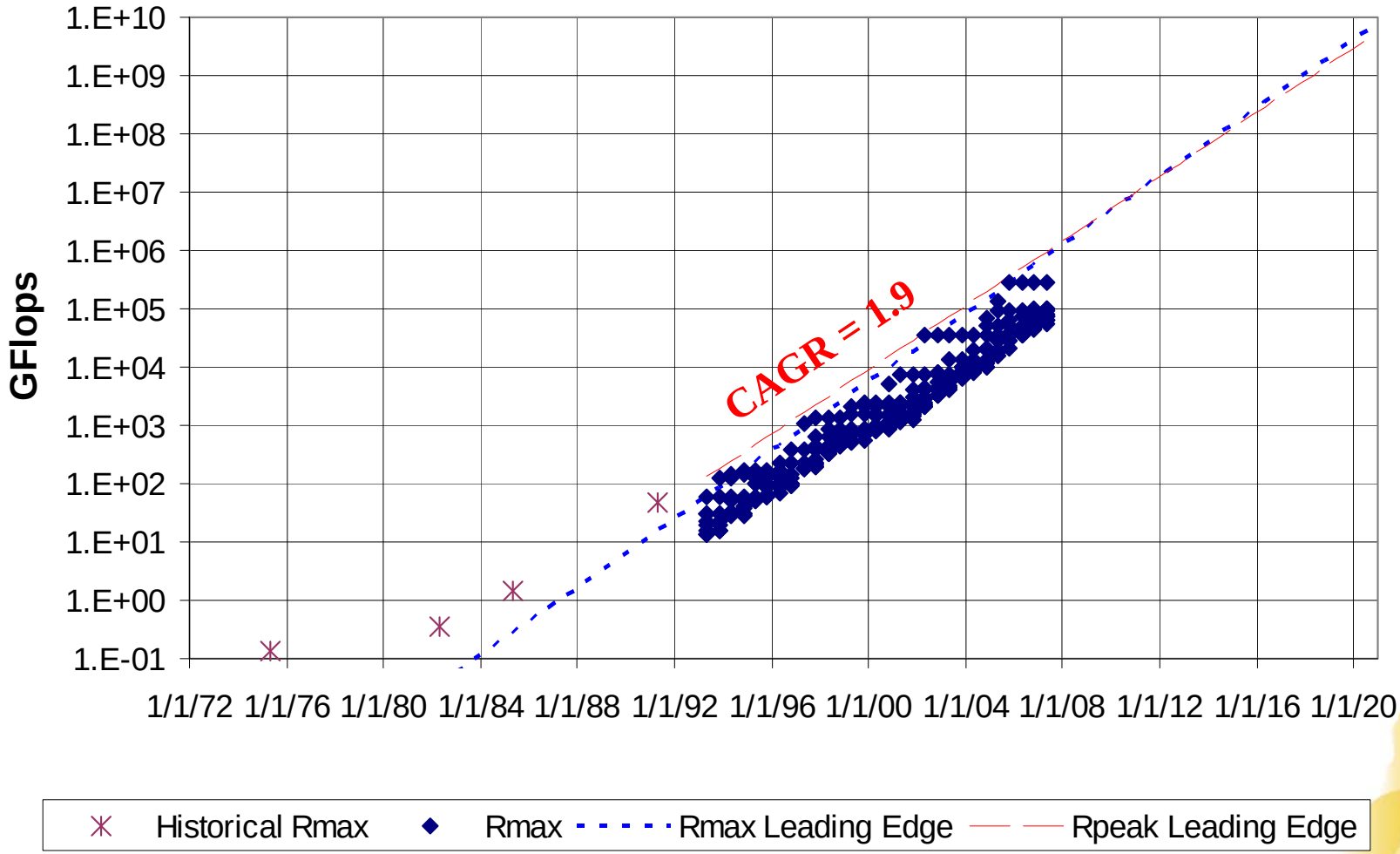
Does Supplying Energy and Getting Rid of Heat Dominate Area?

It Also Bothers Me That:

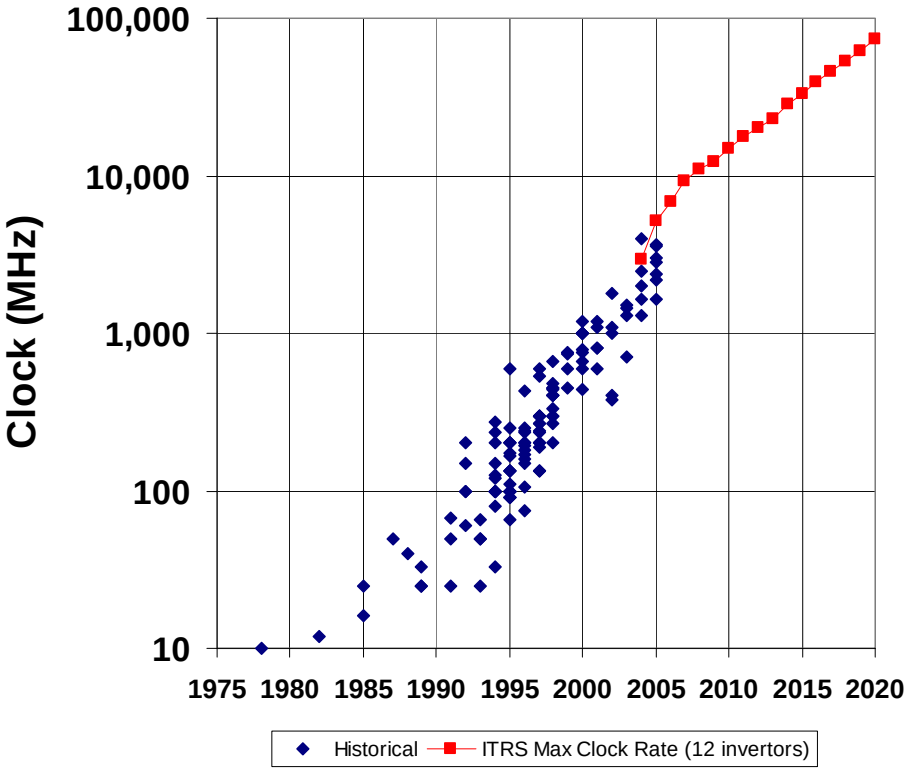
- Modern microprocessor state growing as Moore's Law
  - Regardless of the number of computational units
- Memory is as dumb as it was 50 years ago
- We insist on giving *persistent* names to the *tarballs* representing the physical cores
- And go to *great extremes* to separate the persistent names of memory from its location
- Newer classes of apps "visit" data irregularly
  - Where "caching" copies is wasted energy

The Way We Were

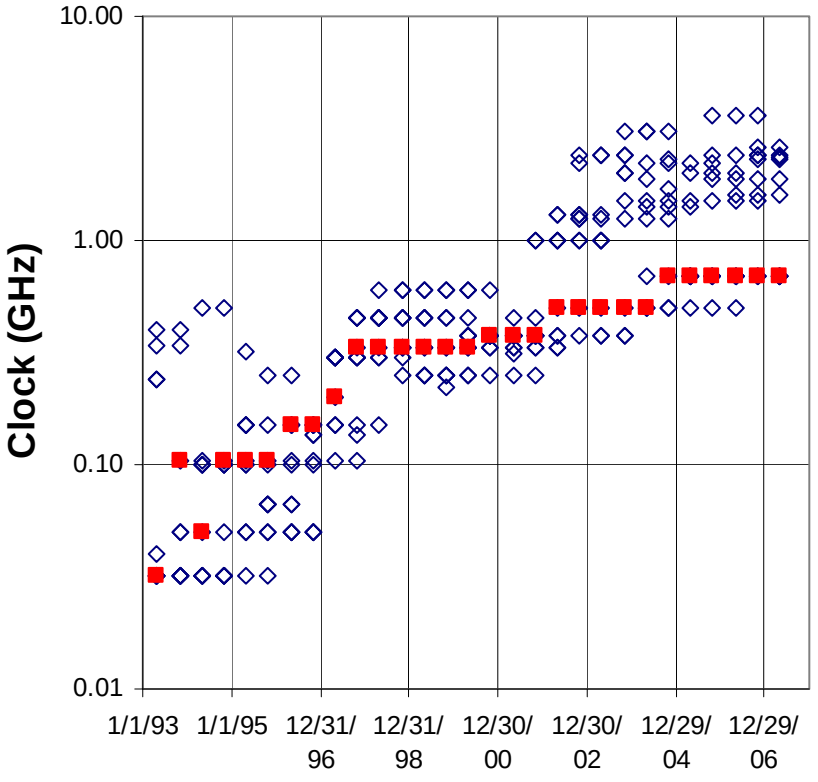
The Historical Top 10



Clock Rates

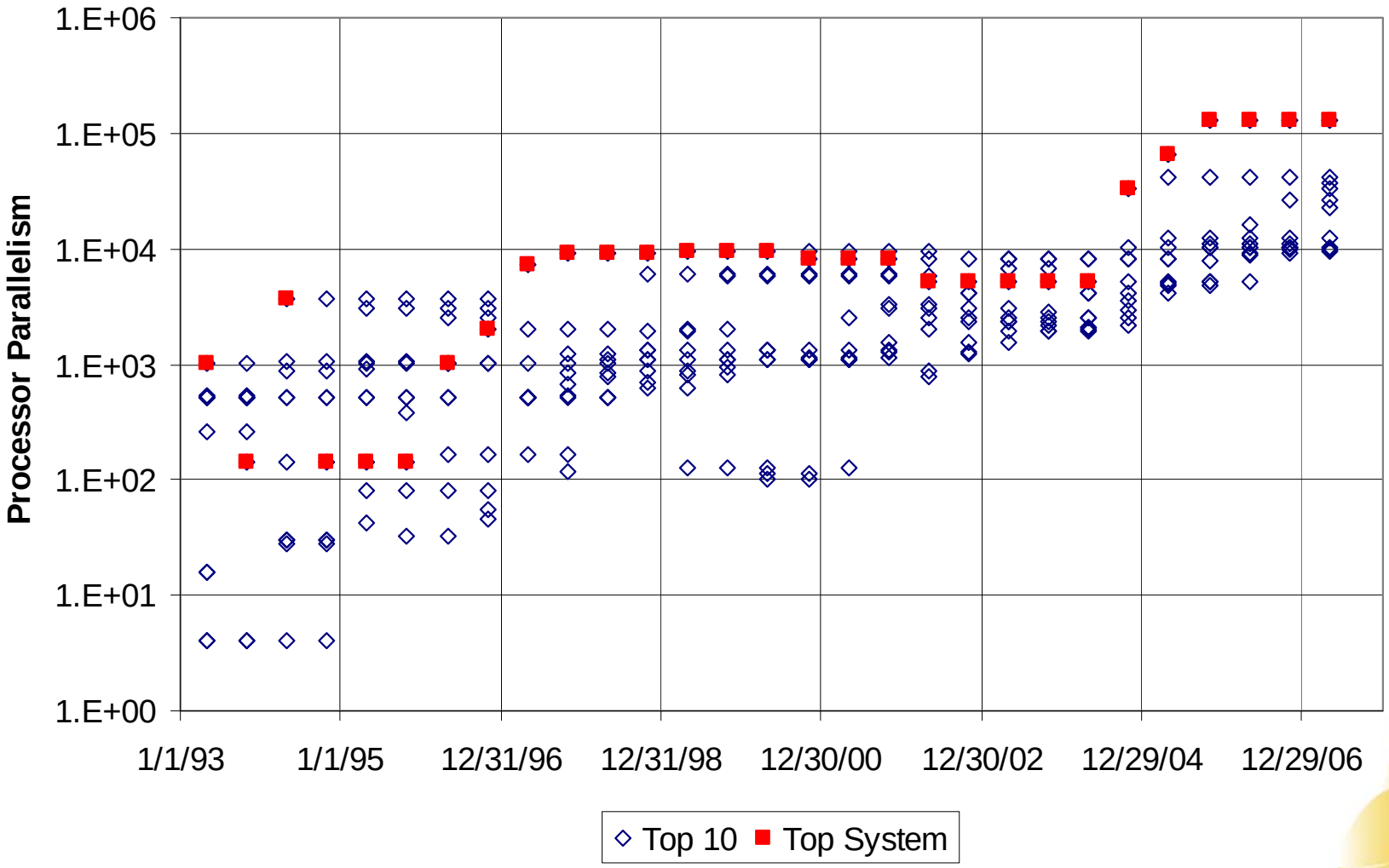


Microprocessors

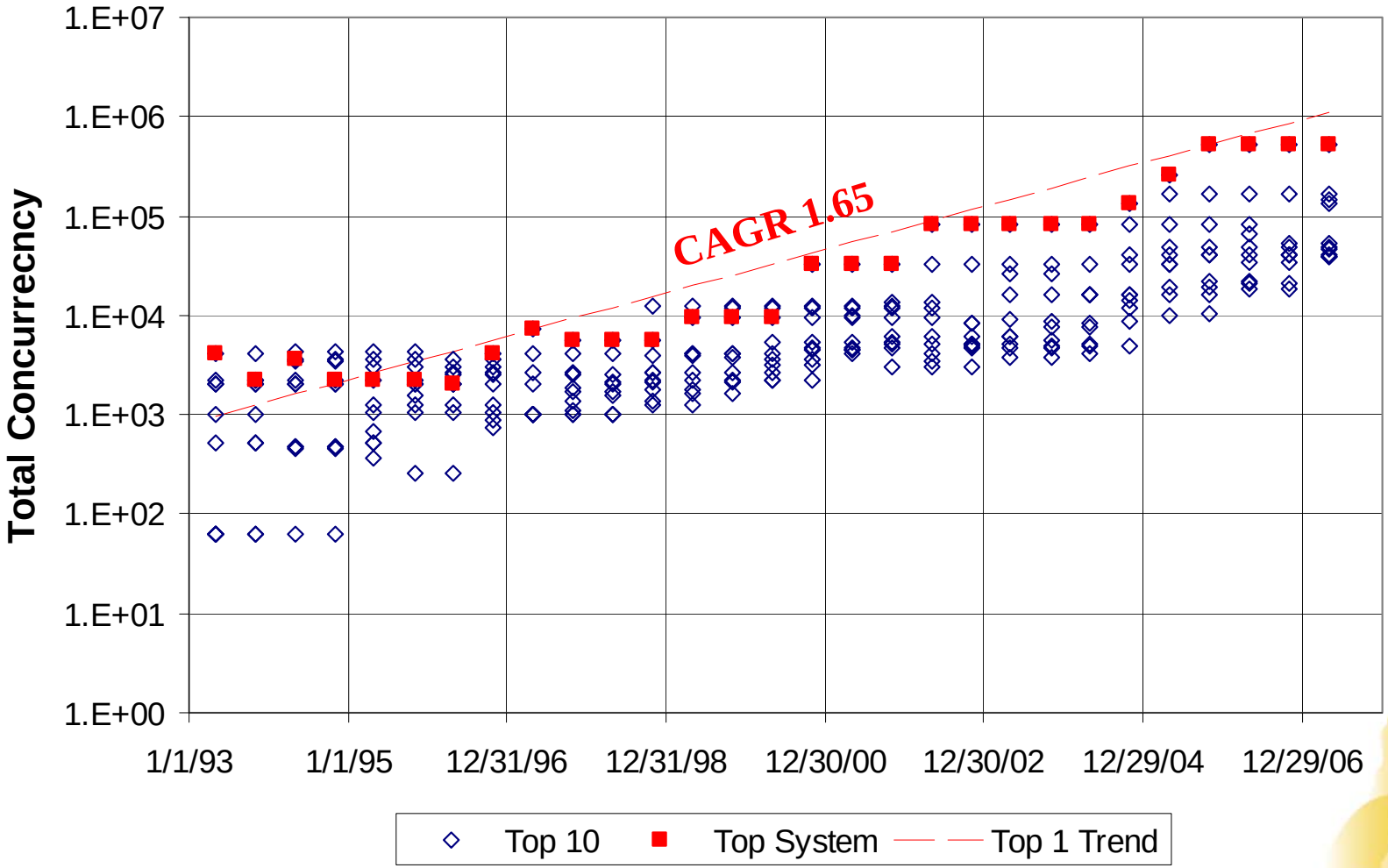


Top 10

Processor Parallelism



Concurrency: Flops per Cycle

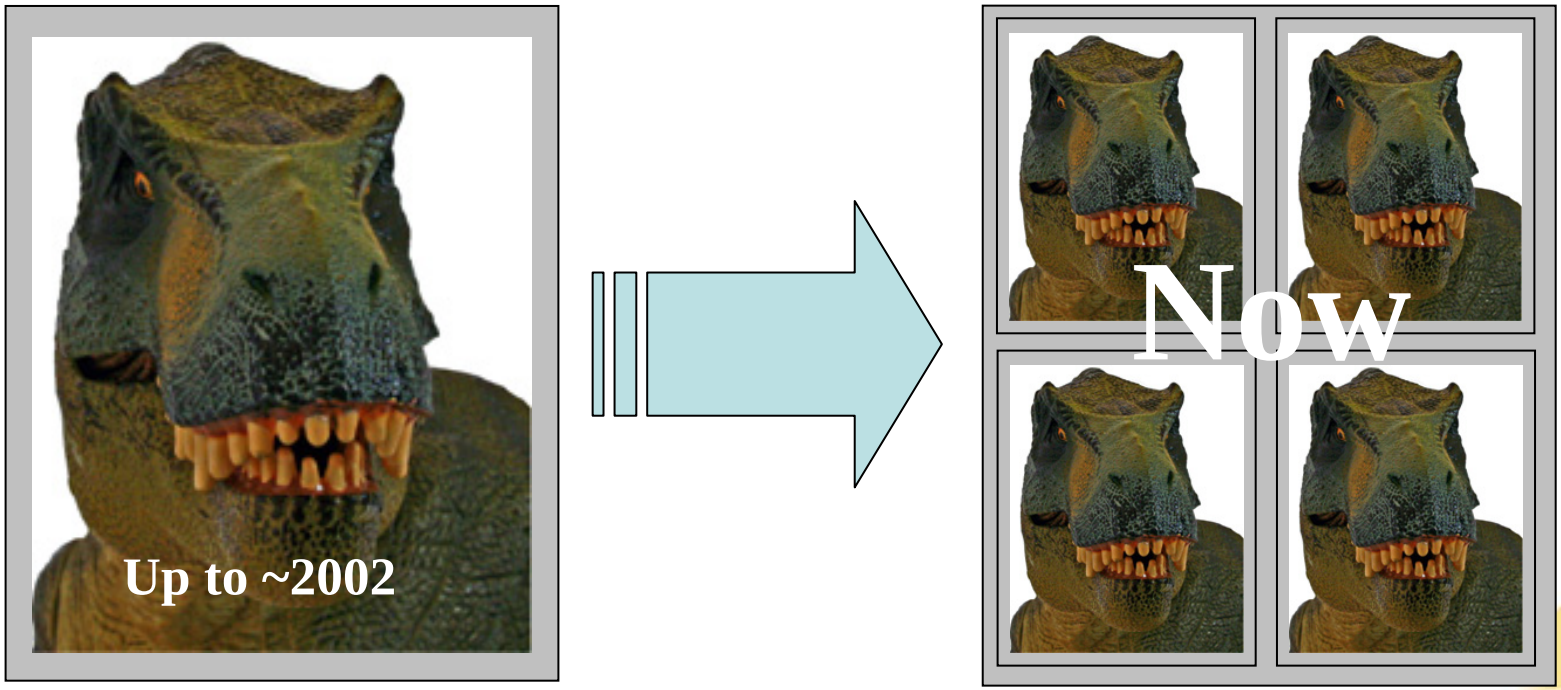


The Moore's Law We ~~Know~~ Knew & Love

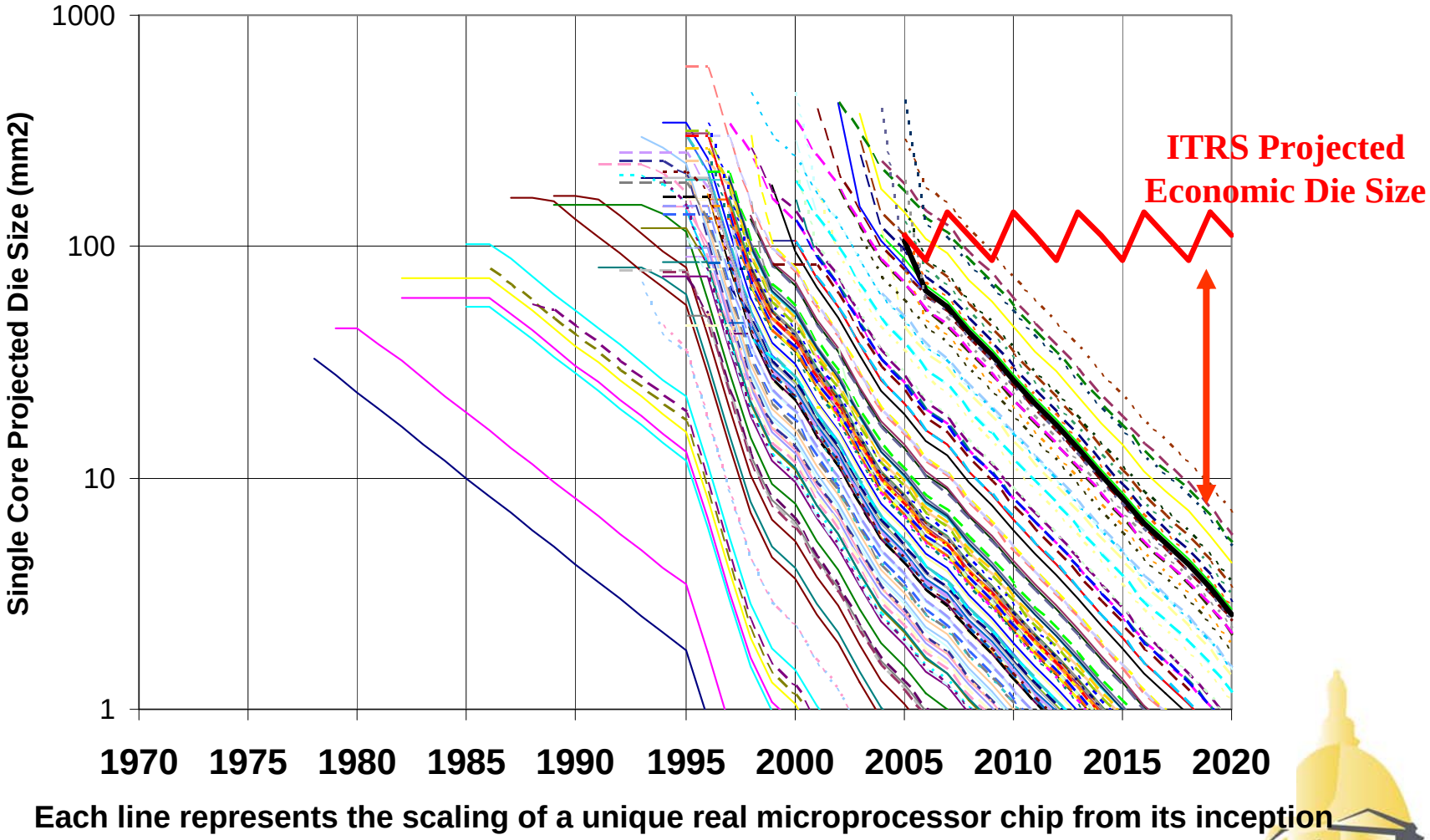
- Goal: 4X ***Functionality*** every 3 years
- Underlying technology improvement:
  - Growth in transistor density **Yes**
  - Growth in transistor switching speed **Yes**
  - Growth in size of producible die **Not in commercial volumes**
- Microprocessors: Functionality=IPS
  - ~1/2 from higher clock rate **No: heat**
  - ~1/2 from more complex microarchitectures **No: complexity**
- Memory: Functionality = Storage capacity
  - ~2X from smaller transistors **Yes**
  - Shrinkage in architecture of basic bit cell **Yes, but ..**
  - Increase in die size **Not at commercially viable prices**

And it is silent on inter-chip I/O

The Darwinian Multi-Core Evolution



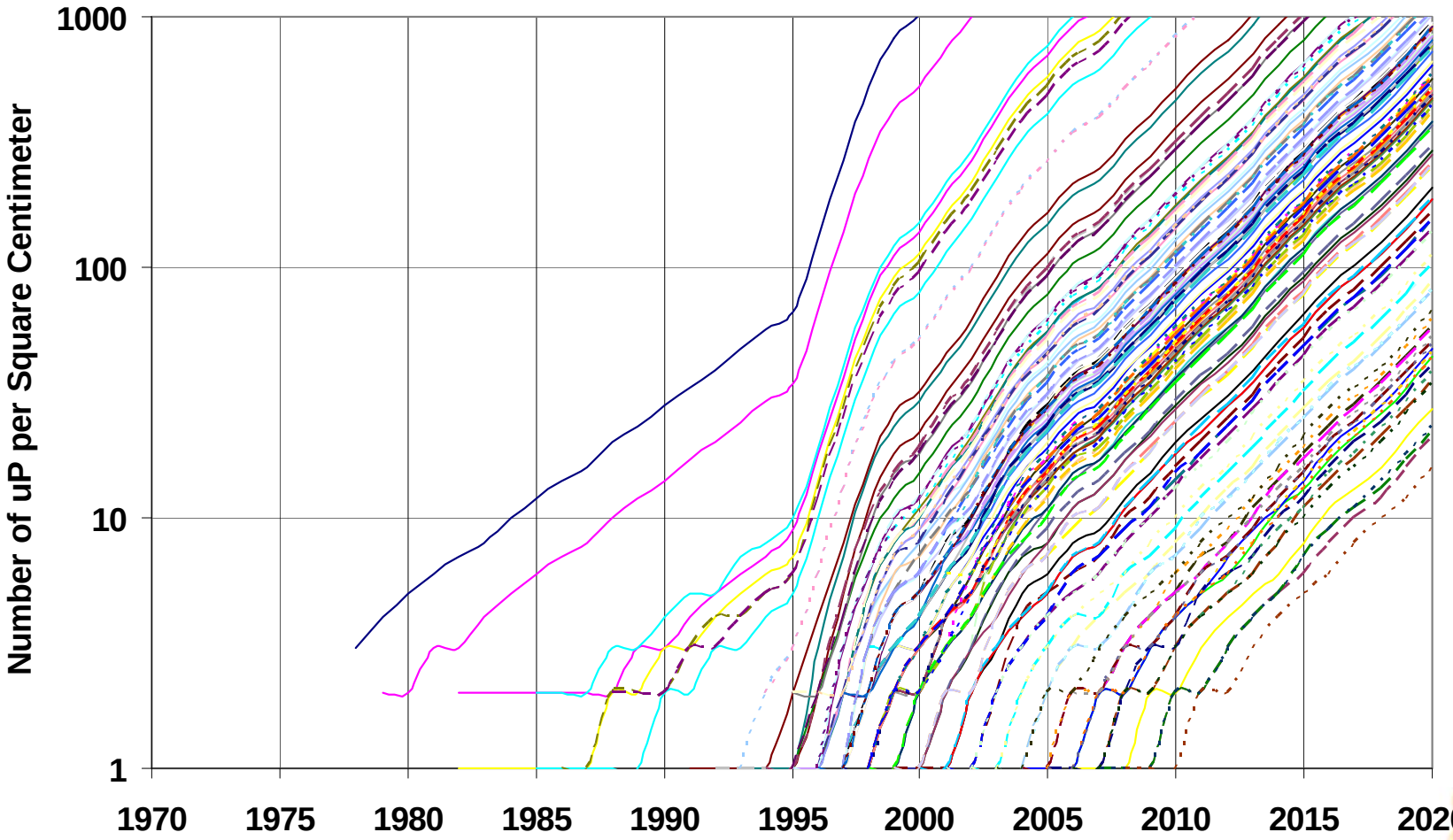
Area Scaling Alone Reveals the Rationale for Multi-Core



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

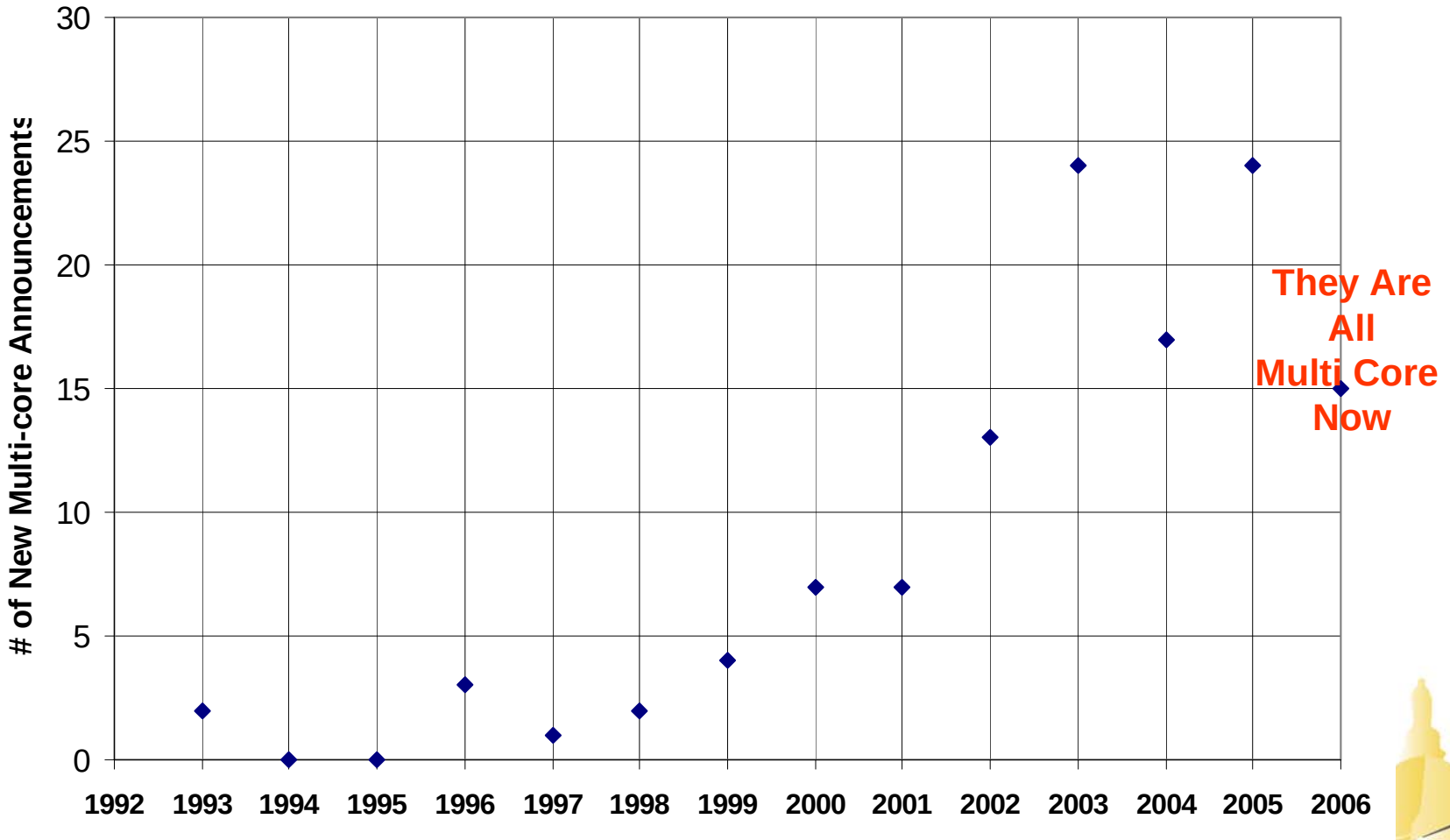
How Many Can We Fit on a cm²?

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

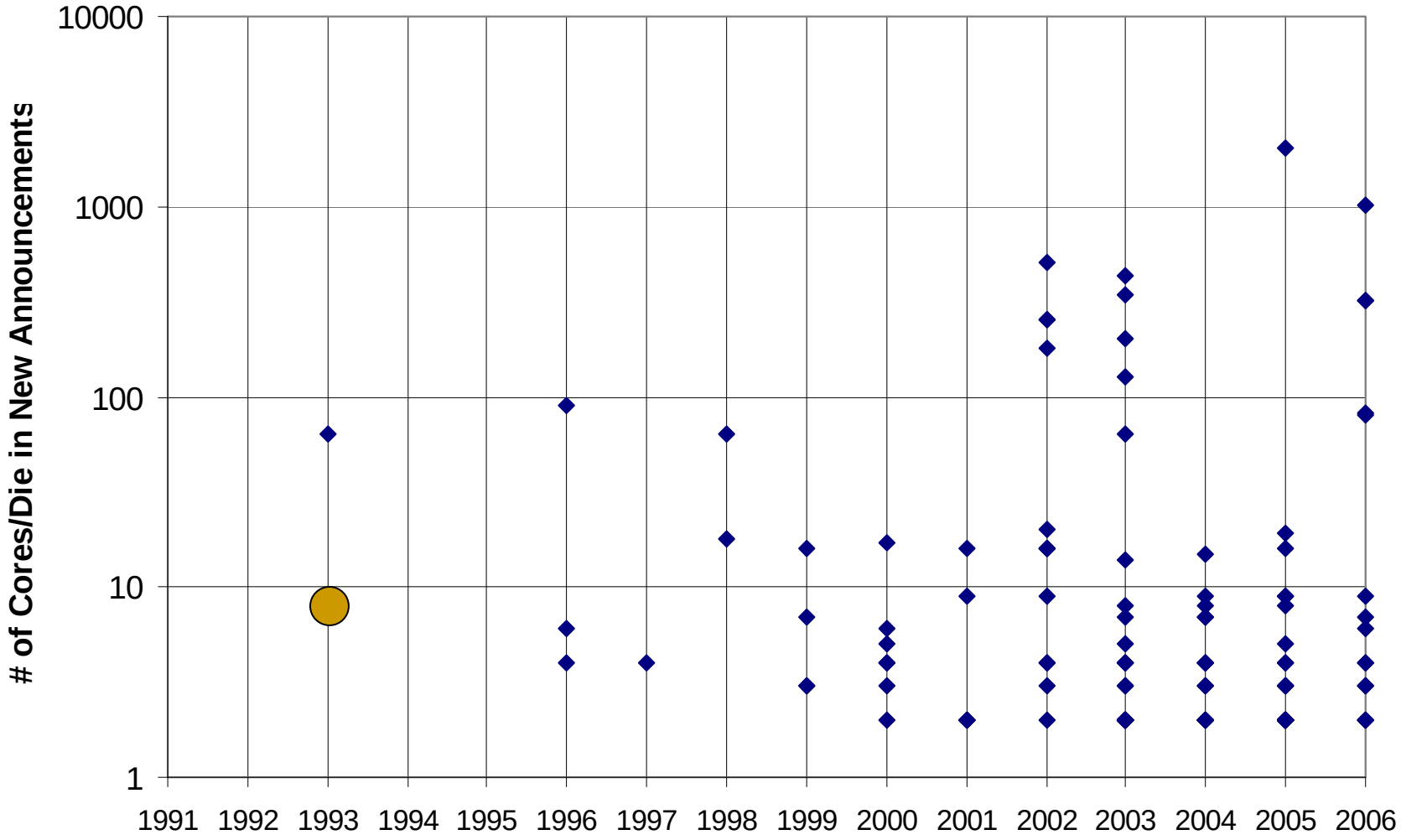


Answer Potentially 1000's!!!!

And a *Flood Tide* of Recent Announcements

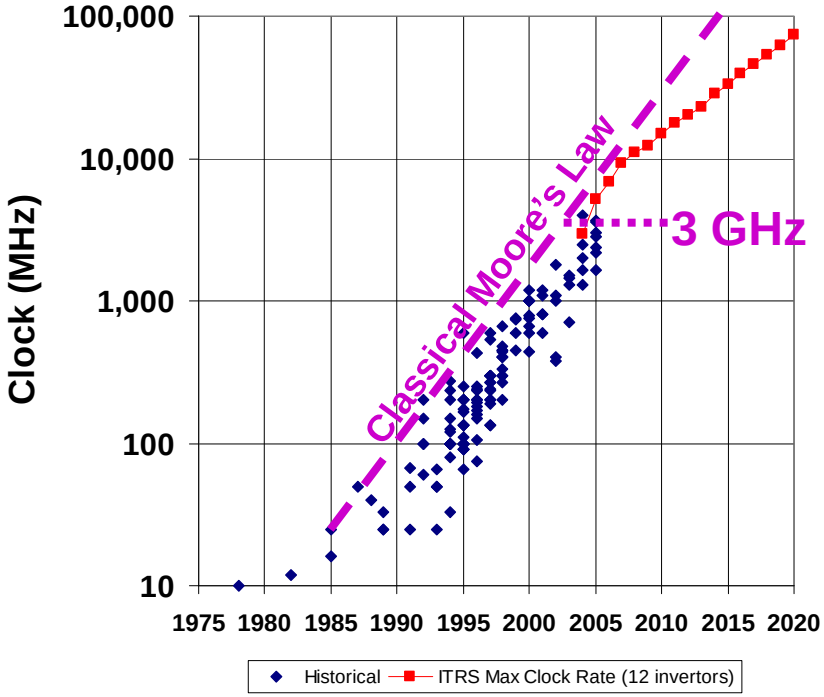


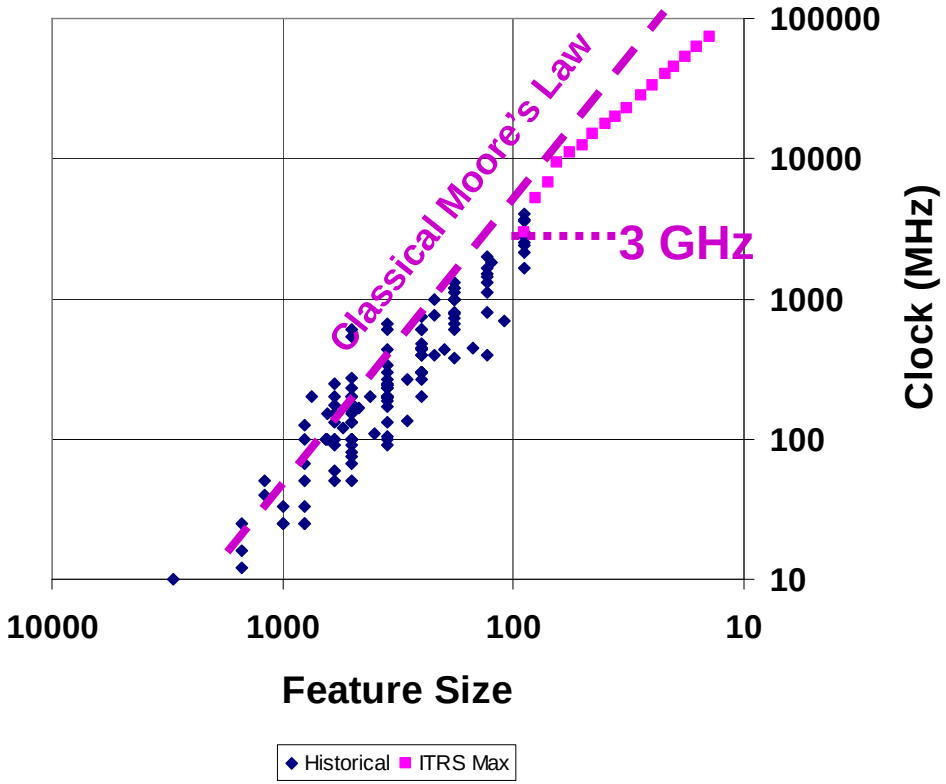
And Not Just “Twosies”



The Classical Limiting Factors for Microprocessor Chips: Power & Contacts

Peak Logic Clock Rates

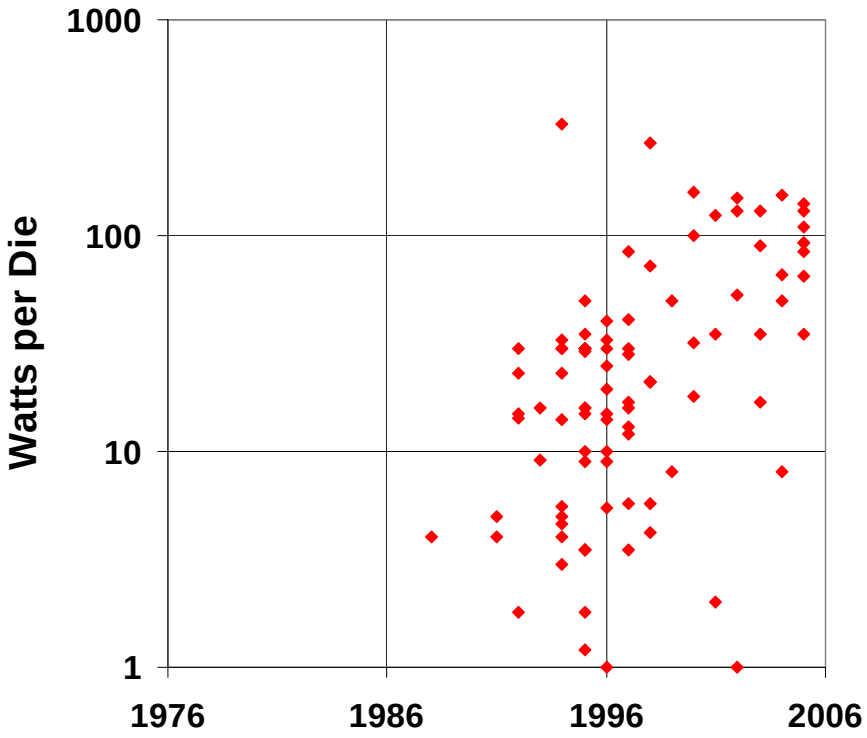


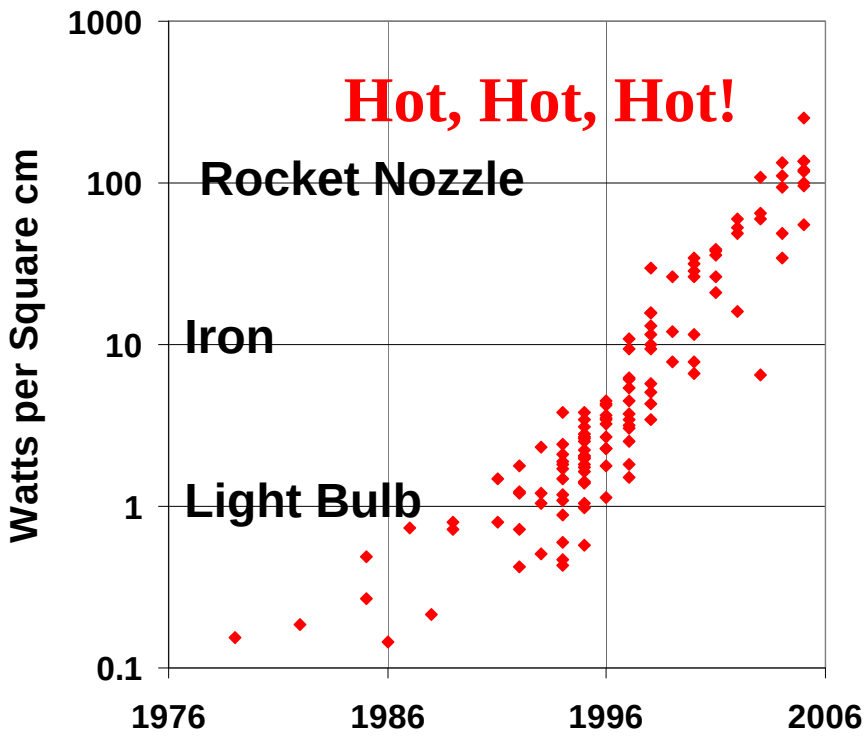


2005 projection was for 5.2 GHz – and we didn't make it in production. Further, we're still stuck at 3+GHz in production.

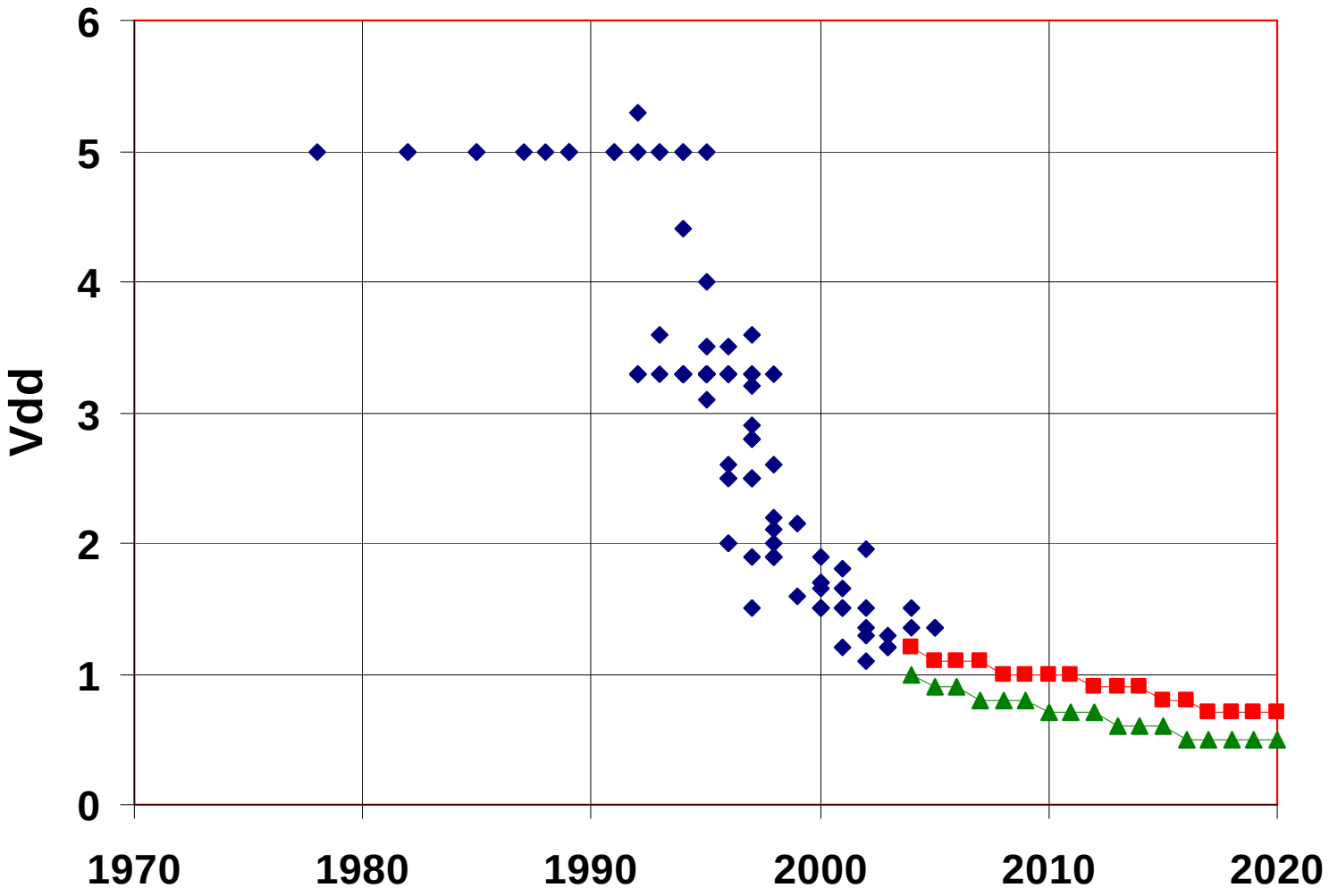
Why the Clock Flattening?

POWER





Because Vdd No Longer Declining



Multi-core Power and Clock

Max Limit Will Grow only Slightly

Decreasing ~linearly with Technology

Assume Constant for Multicore

Increasing As Square with Technology

Chip Power = Cap/device*#_devices/core*cores/chip

*** Clock * Voltage^2**

Max Clock Rate Grows Rapidly with Technology

Reaching an Asymptotic Limit

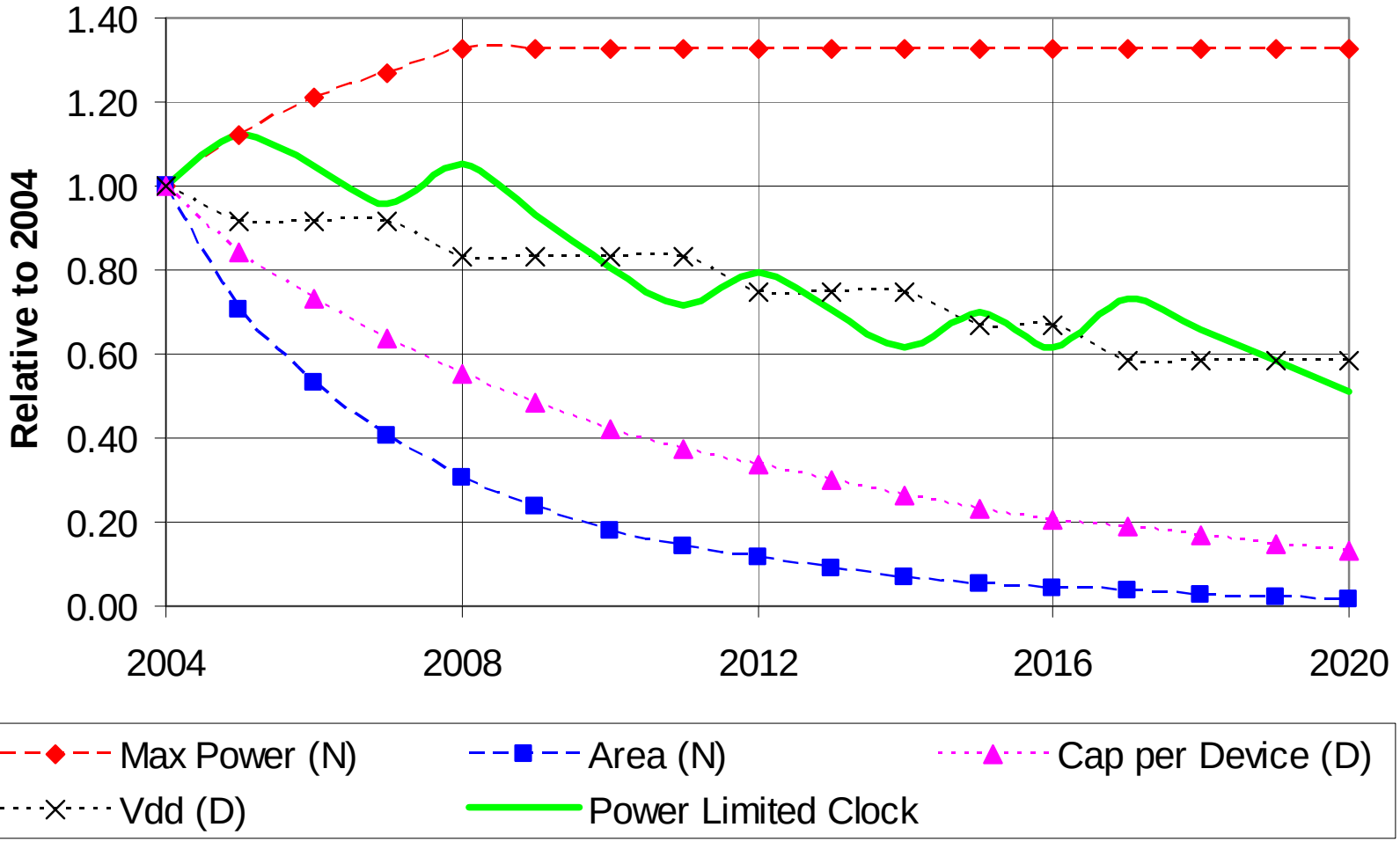
But ONLY KNOB to Balance Equation!!

Rewriting for Clock

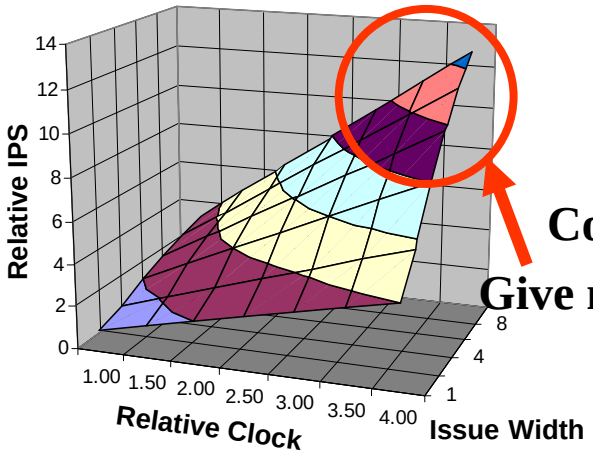
$$\text{Clock} = \frac{\text{Max\_chip\_power(T)} * \text{Reduction\_in\_core\_area}}{\text{Cap\_per\_device} * V2}$$

This now governs Core Frequency.
Not Faster Transistors!!!

Relative Change In Factors

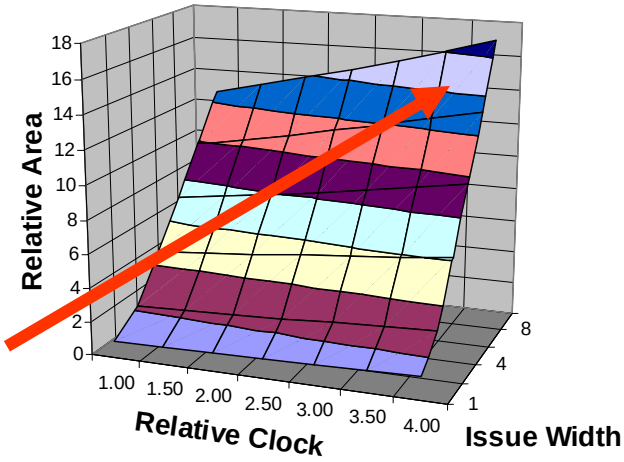


What Kind of Core Should We Replicate?



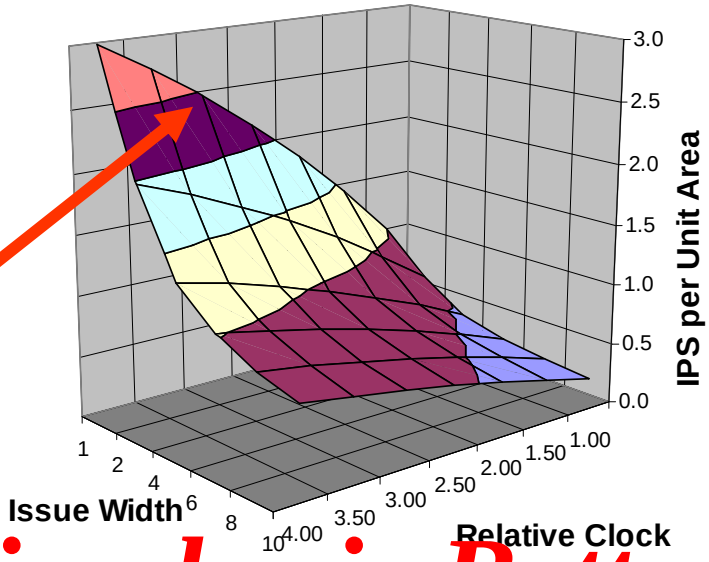
Complex designs

Give most performance



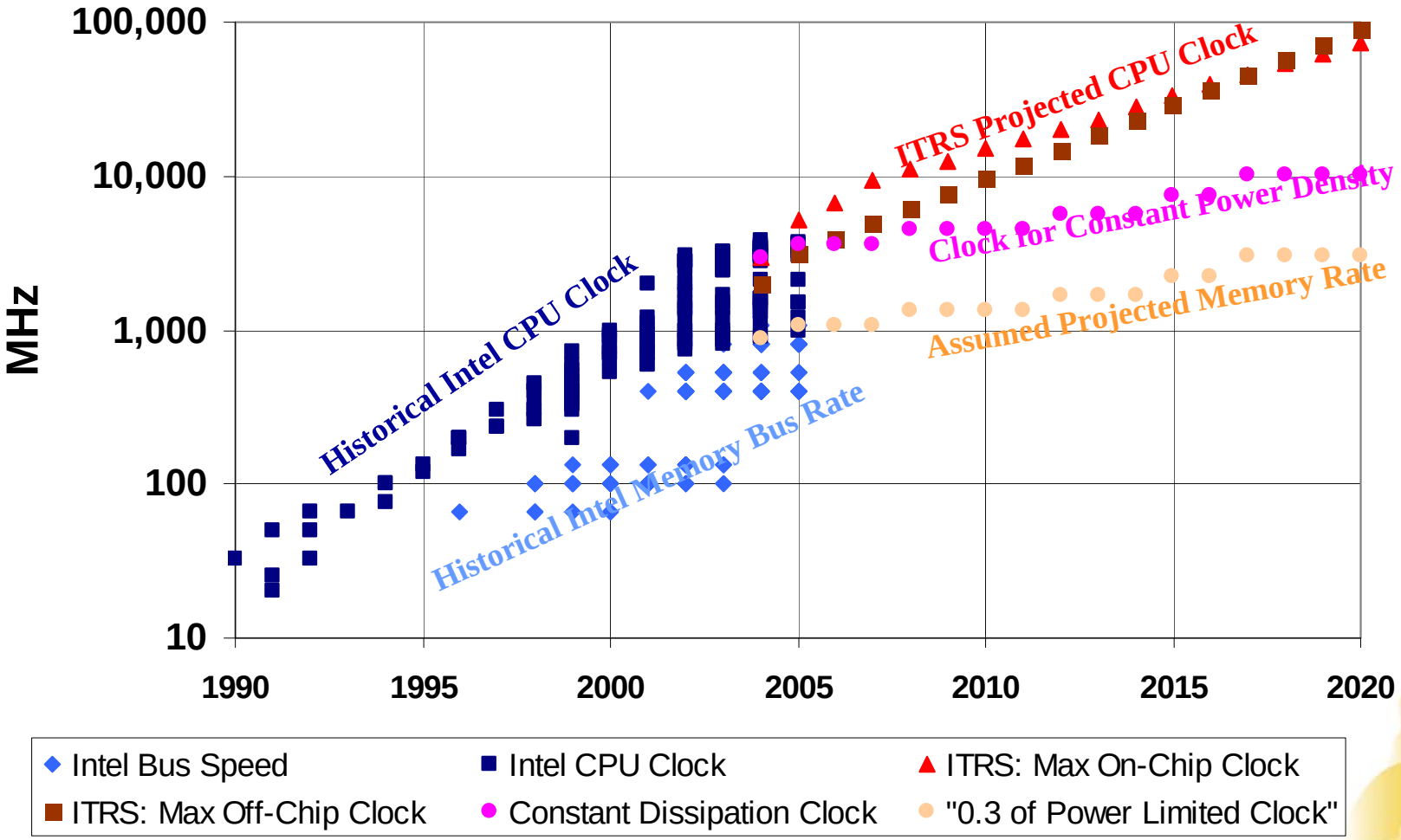
But also largest area

But simpler gives better performance/area

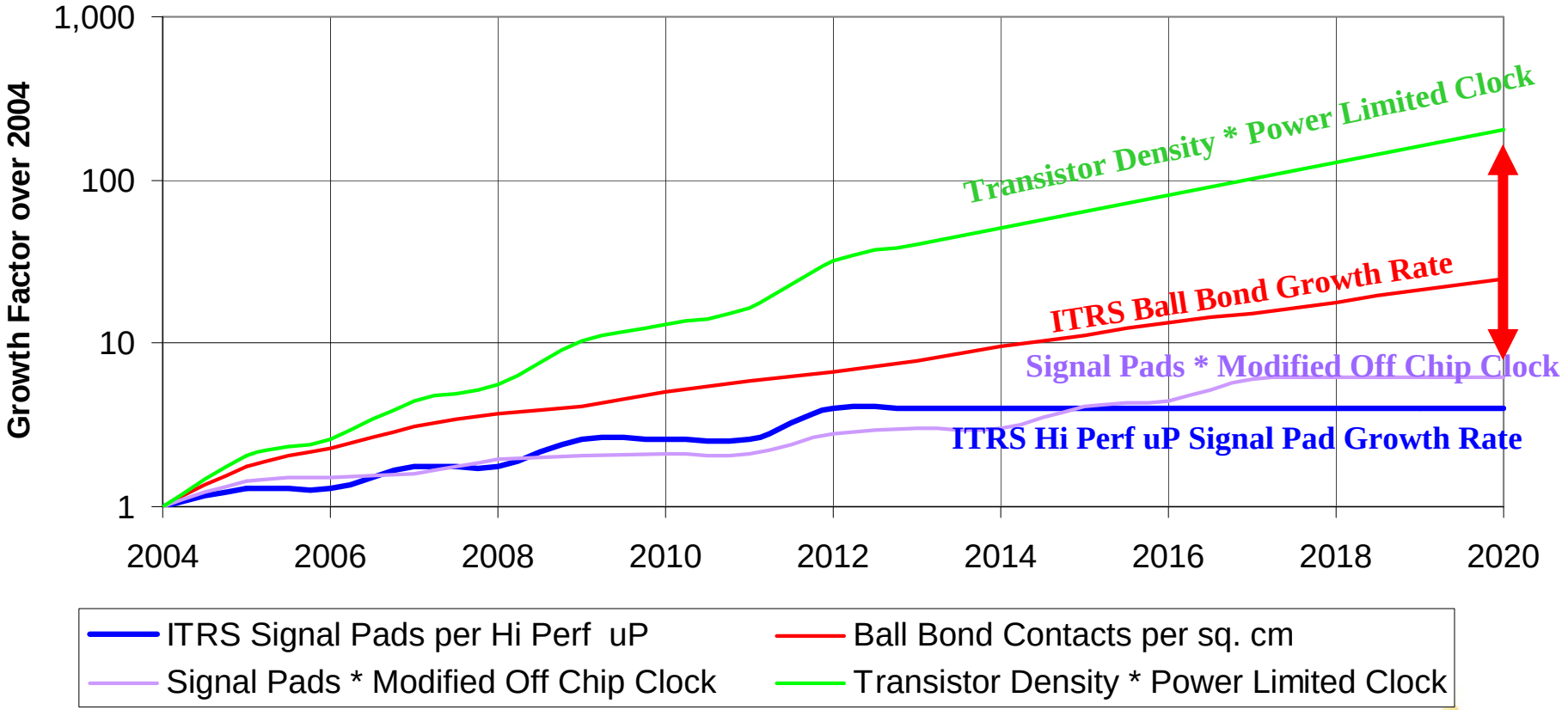


Simpler is Better

What About Memory Bus Clocks?



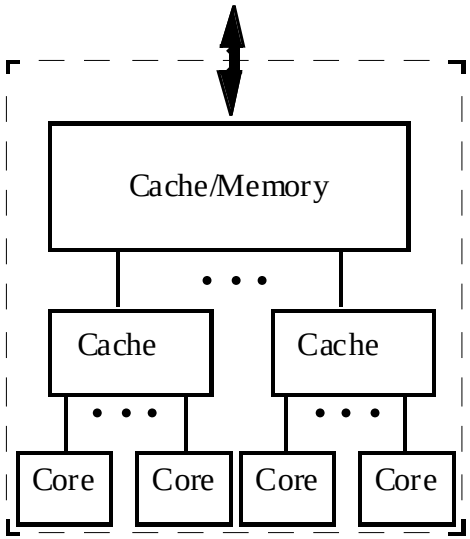
Does Logic Performance Match Off-chip Bandwidth Potential?



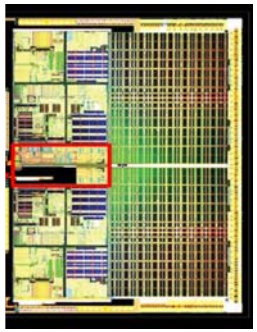
A Growing Mismatch!

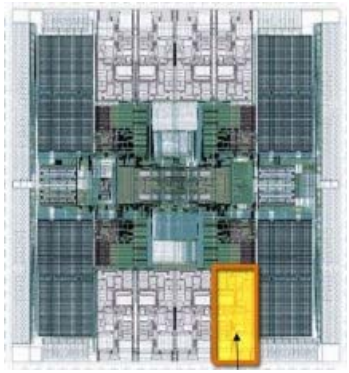
The Multi-Core Family Tree

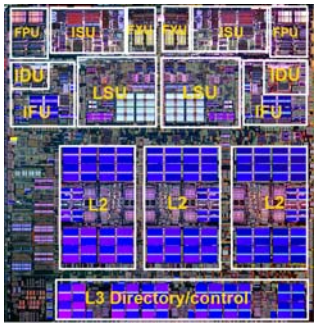
This may be the Architecture You Think of for Multi-Core



(a) Hierarchical Designs



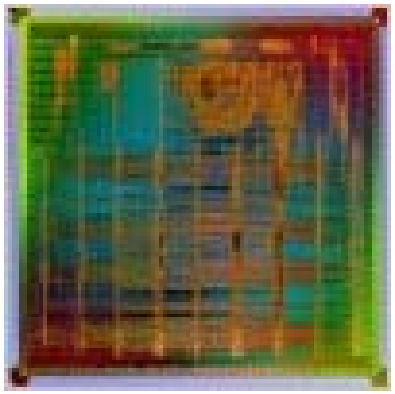


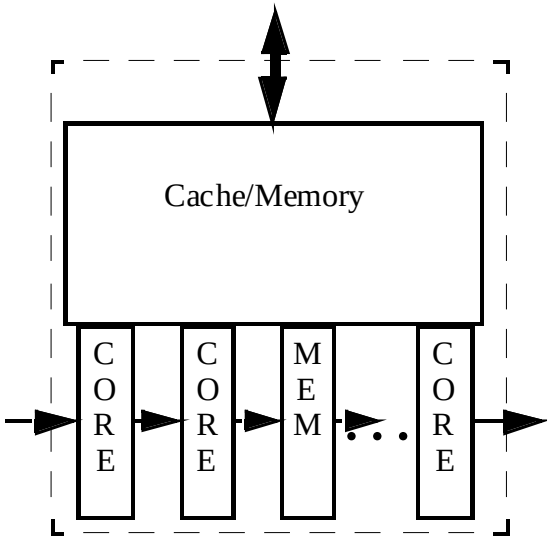


- **Intel Core Duo**
- **IBM Power5**
- **AMD Opteron**
- **SUN Niagara**
- **…**

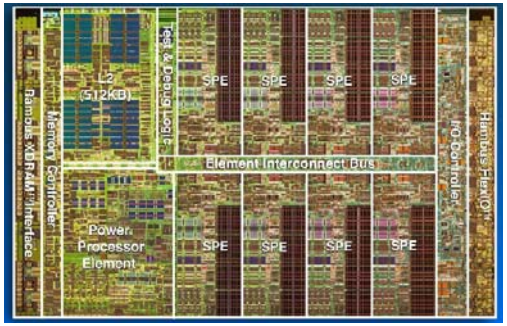
External Bandwidth = *sum* of escapes from cores

But There's at Least One Approach with Lower Bandwidth Needs





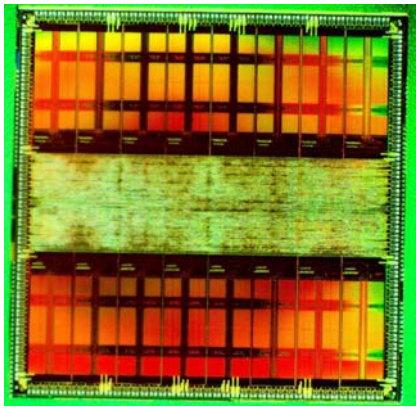
(b) Pipelined Designs

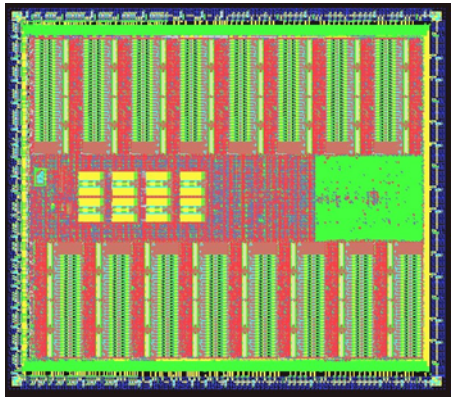


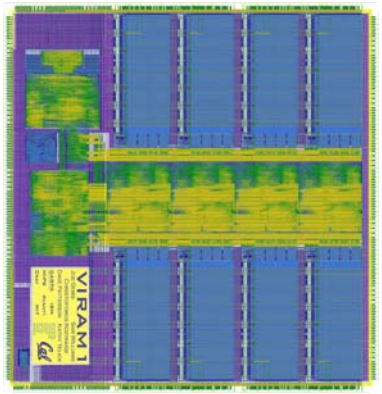
- ***Most Router chips***
- **Many Video chips**
- **Some aspects of IBM Cell**

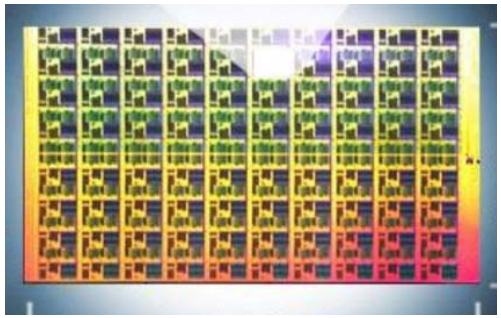
External bandwidth largely *independent* of # of cores

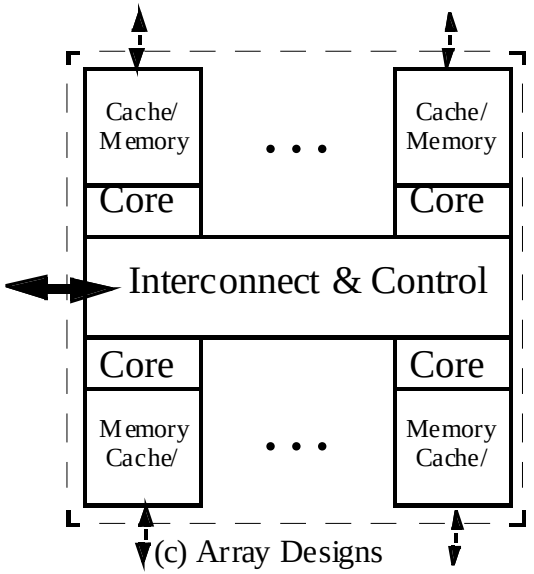
And then there's Array Approaches that Provide Significant Internal Memory







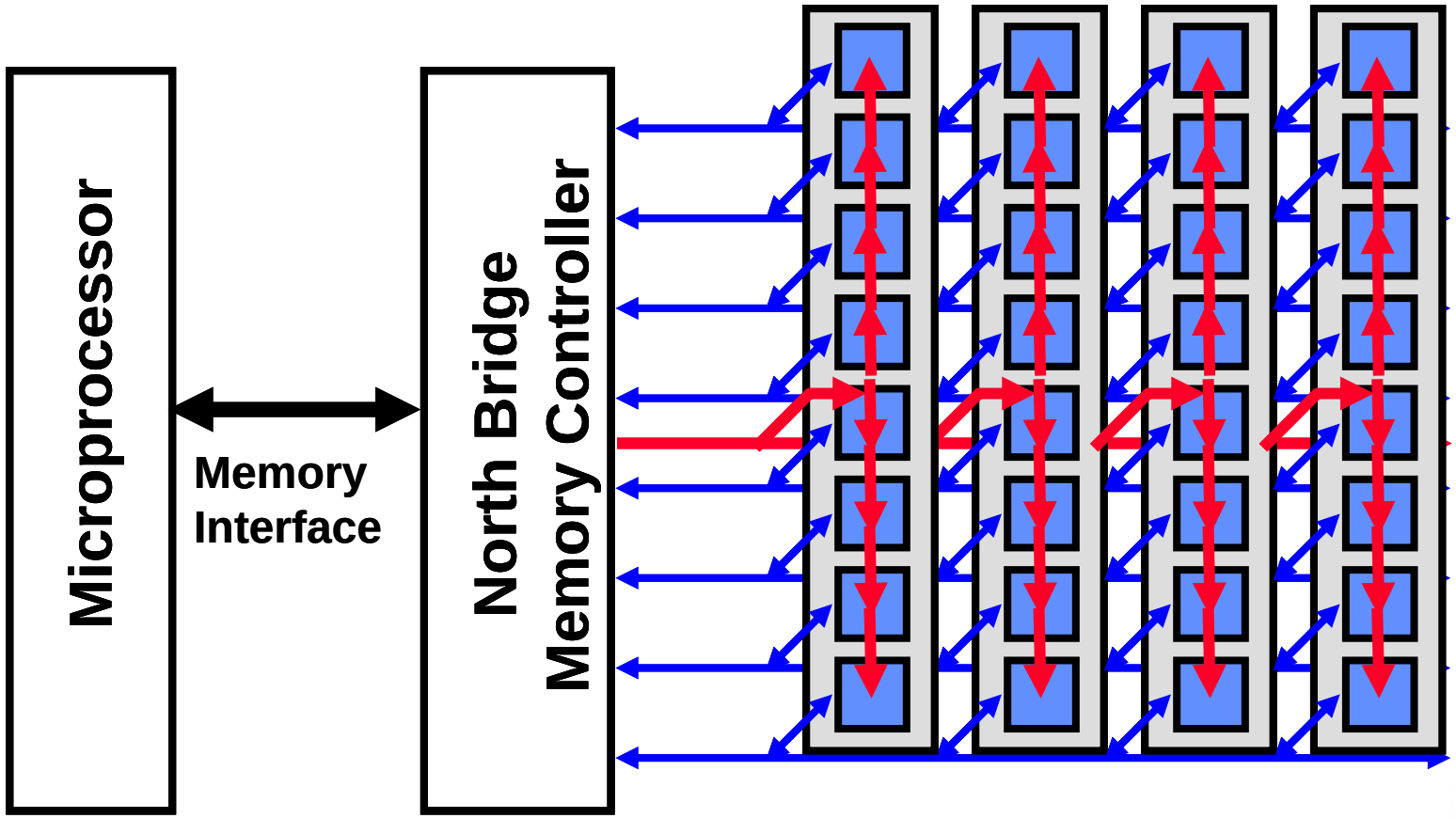




- Terasys
- *Execube*
- Yukon
- Intel Teraflop
- Some Aspects of Cell

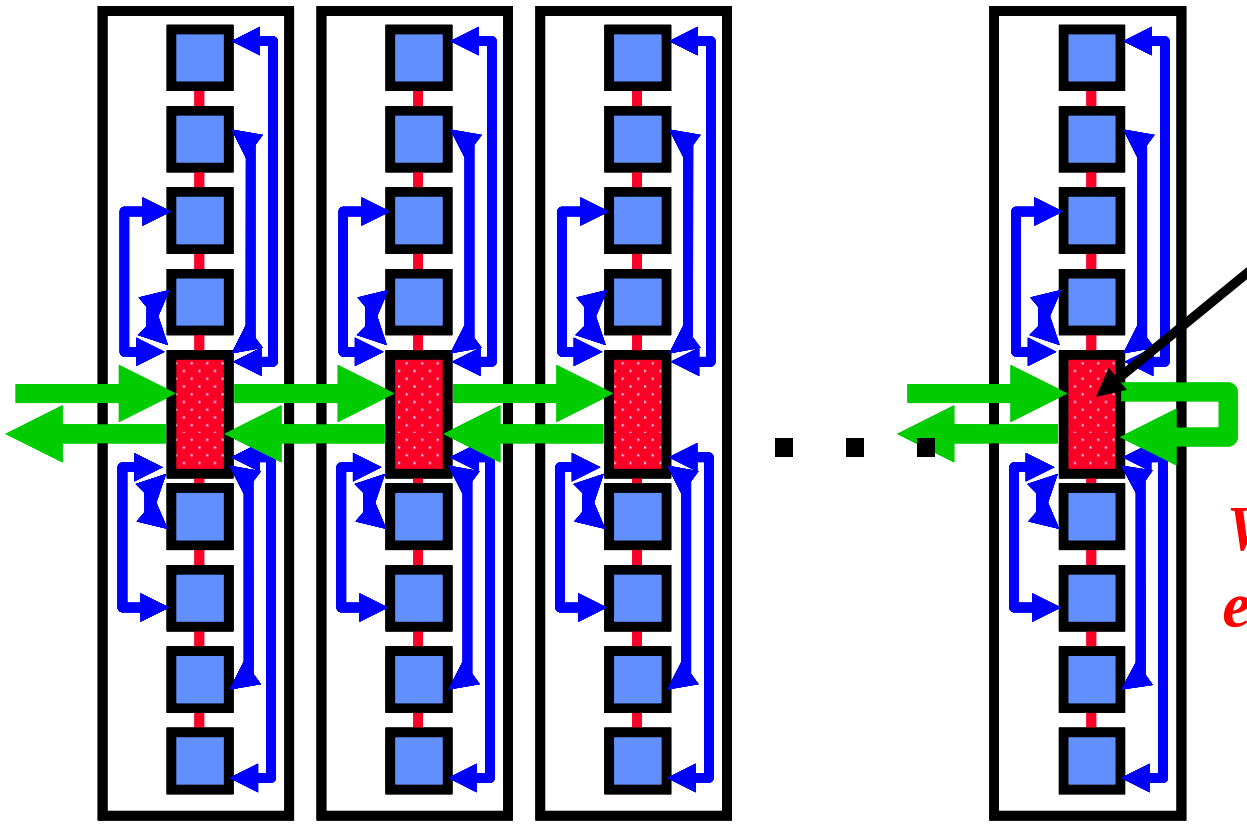
Particularly Effective for Weak Scaling Apps

And Today's Memory Architecture is Evolving to *Feed* the Beast



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

… But Not to Reduce Latency



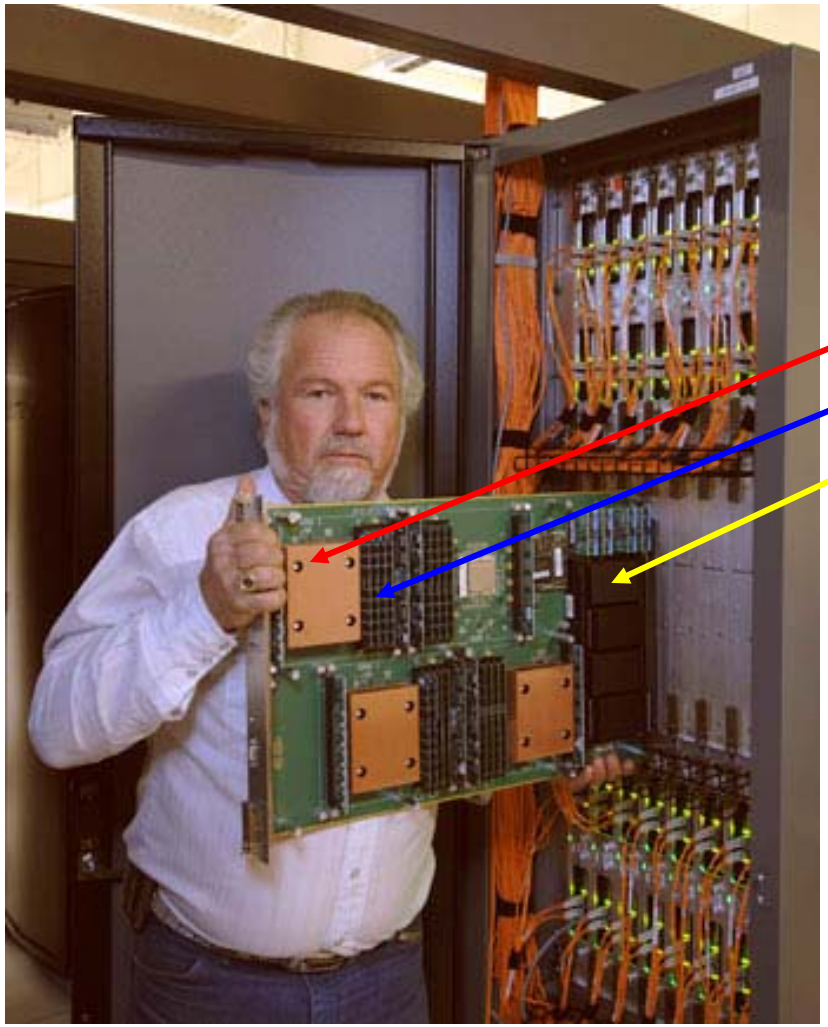
AMB:
Advanced
Memory
Buffer Chip

We've introduced 16 extra chip crossings!

… And at ~2X Power Increase

A Simple Case Study

A Modern HPC System



Computational Board

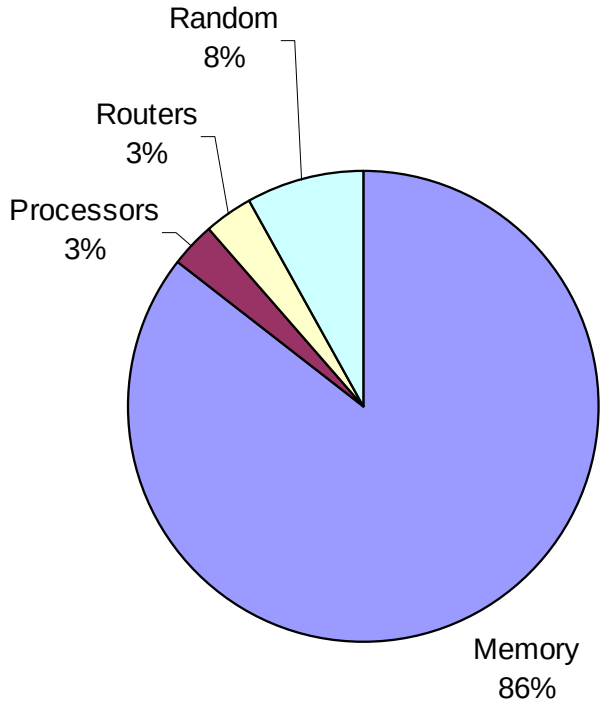
- 4 PE Nodes
- Each PE Node:
  - Dual core Opteron @ 2.6GHz
  - 4 DDR2 2GB DIMMs
- 4 Routers per Board

Key Ratios (all “Peak”)

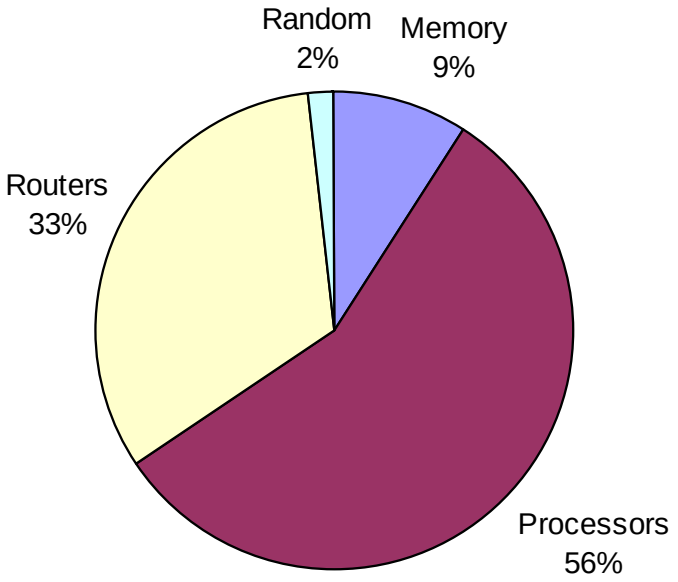
- 2 Flops per cycle per core
- 1.5B per Flop
- 1.25B/s of Memory BW per Flop per core
- 0.25B/s Link BW per flop per PE
- 0.06-0.25B/s of Bisection BW per Flop

What Are We Doing with the Total System Silicon?

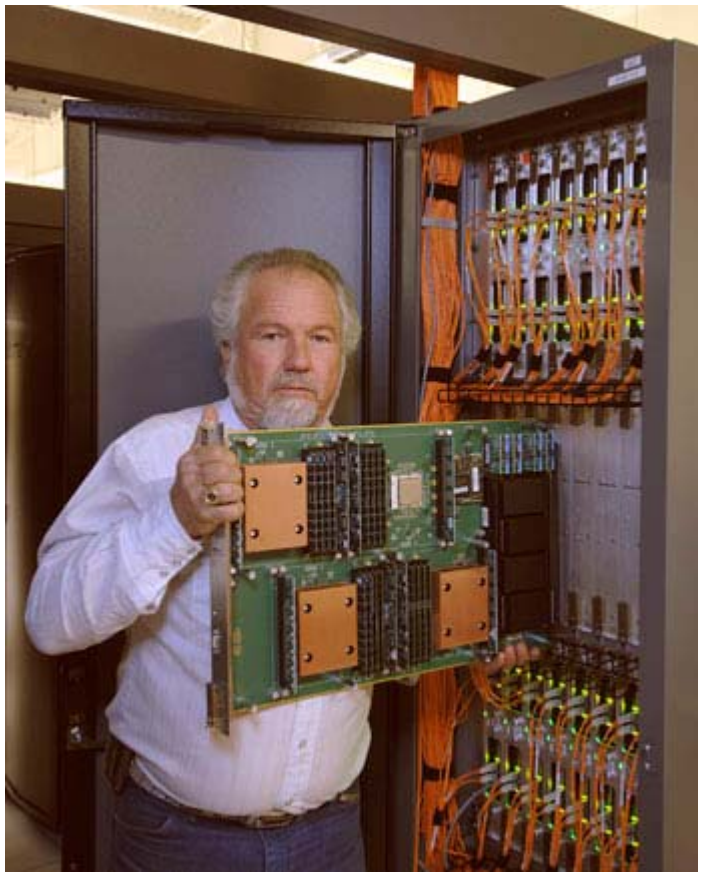
Silicon Area Distribution



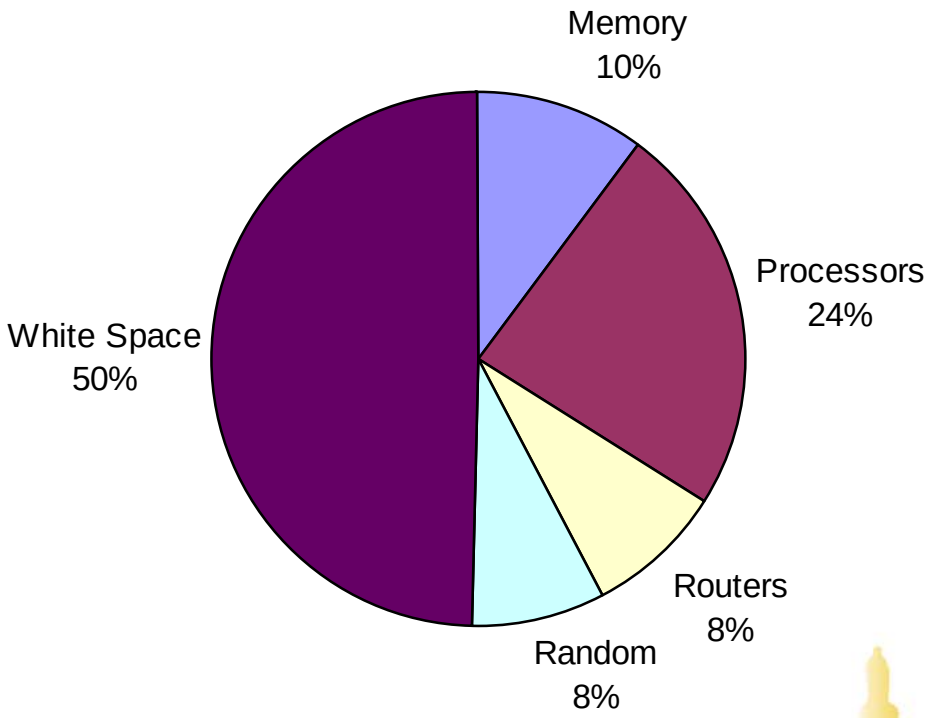
Power Distribution



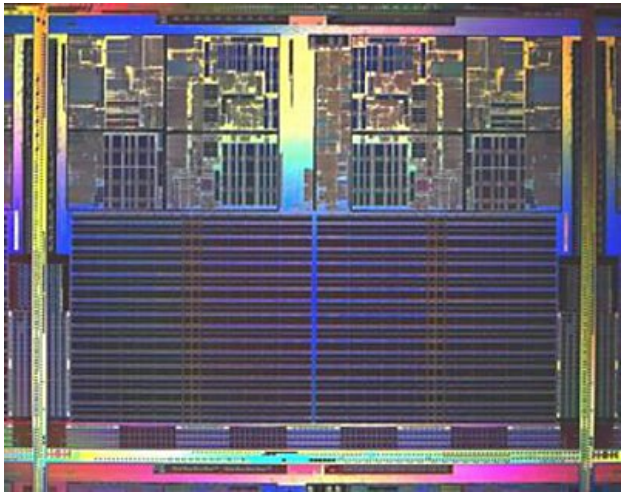
What Is the Board Space Utilization Like?



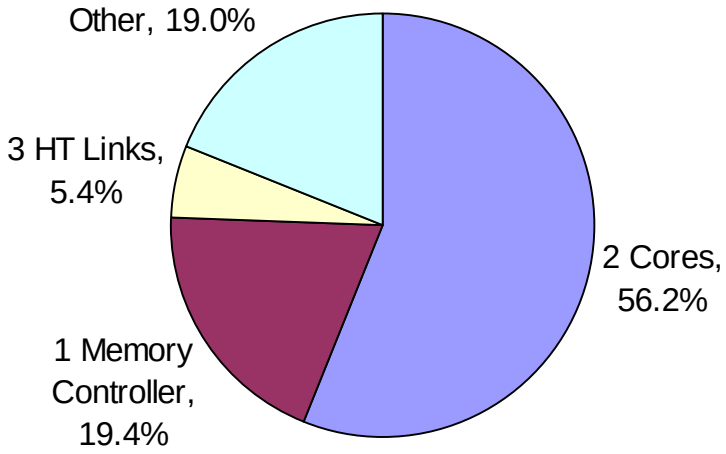
Board Space Distribution

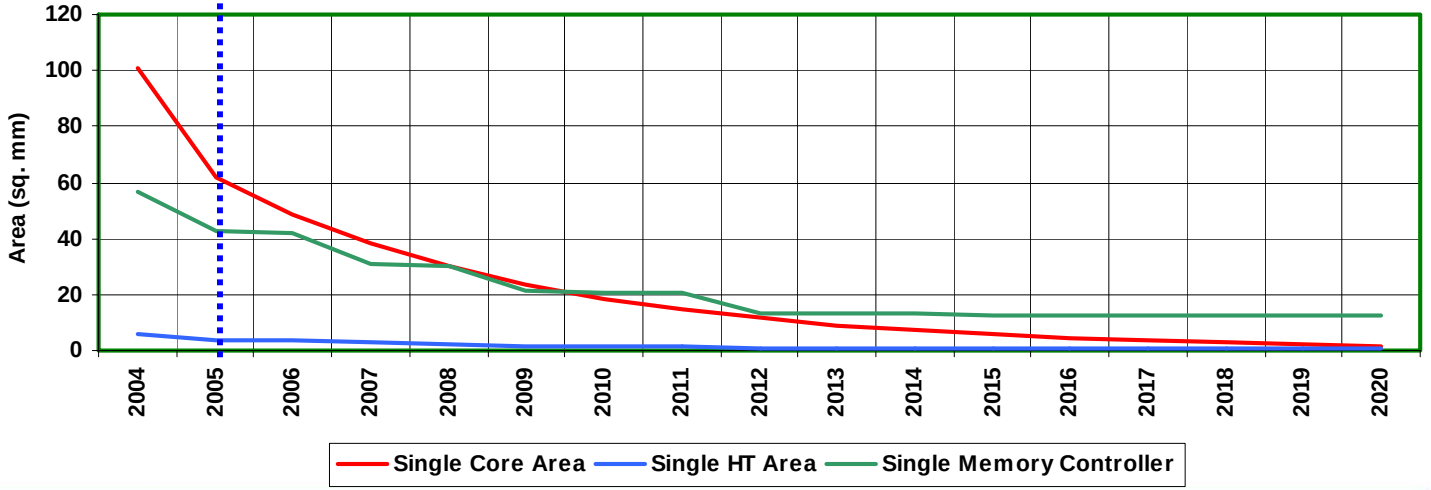


A Dual Core Processor Chip



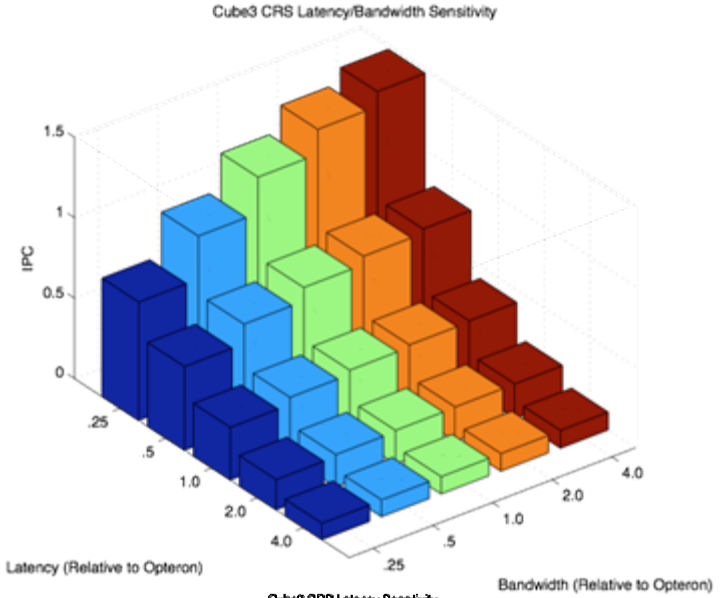
http://techreport.com/reviews/2005q2/opteron-x75/dualcore-chip.jpg

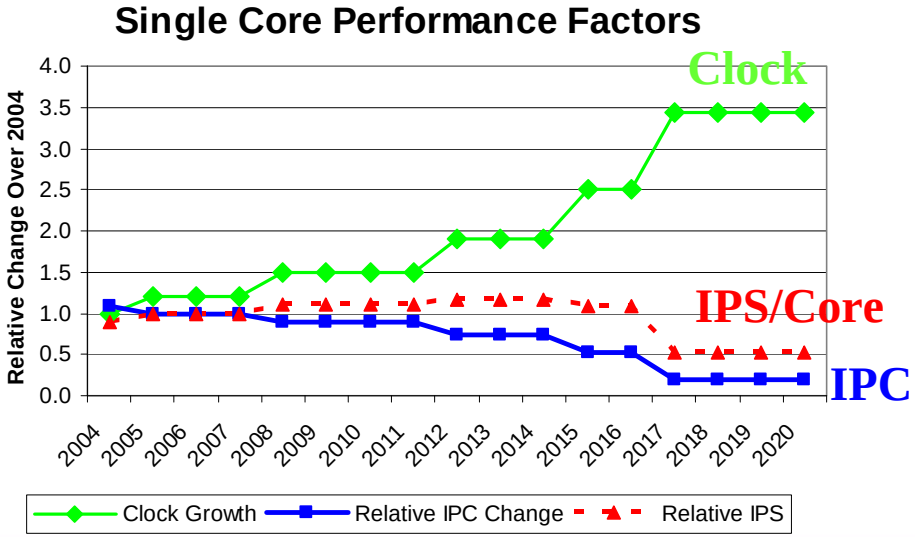


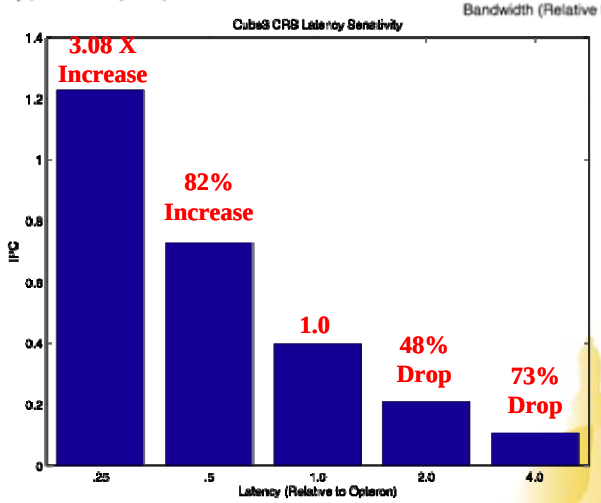


Some Projections

- Off chip memory controls performance
- IPC/core more sensitive to latency than bandwidth
- "Flat" off chip physical latency => relative latency grows with clock



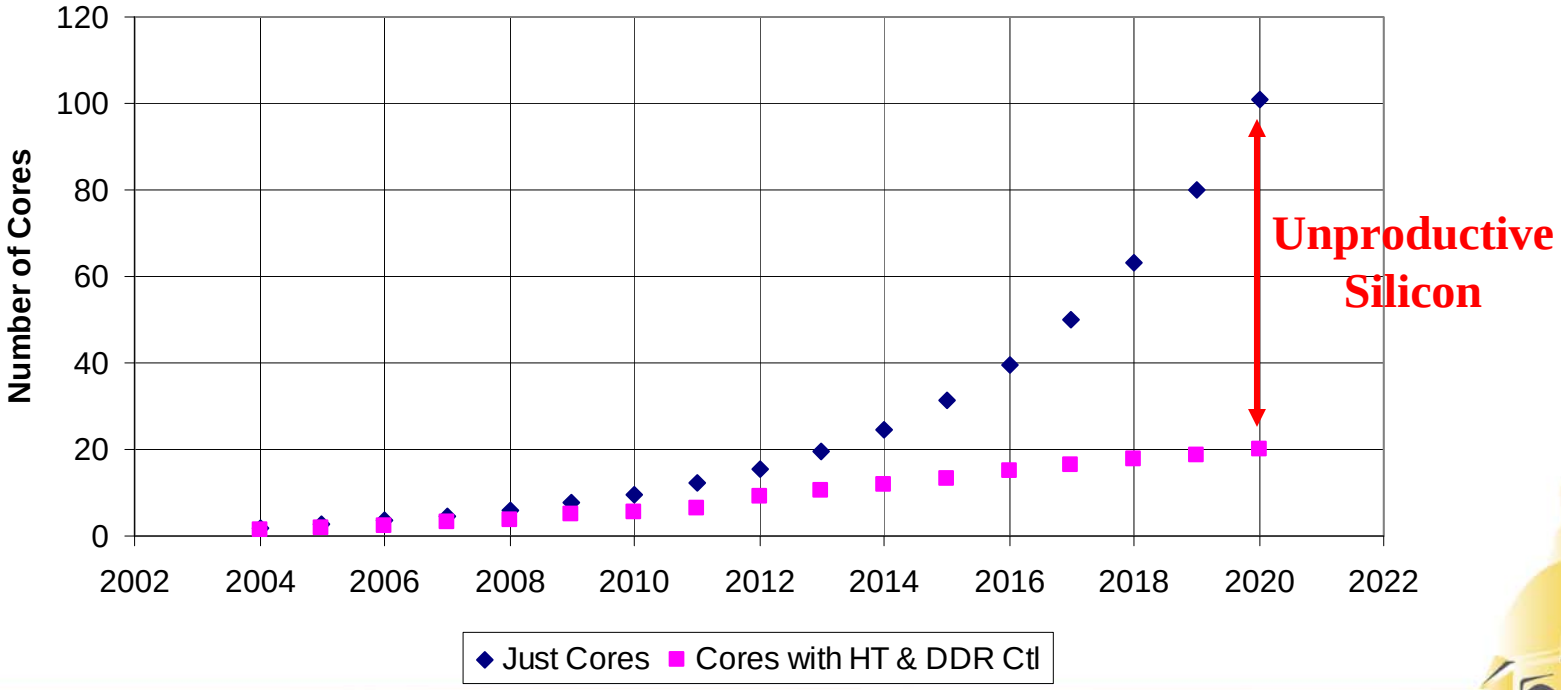




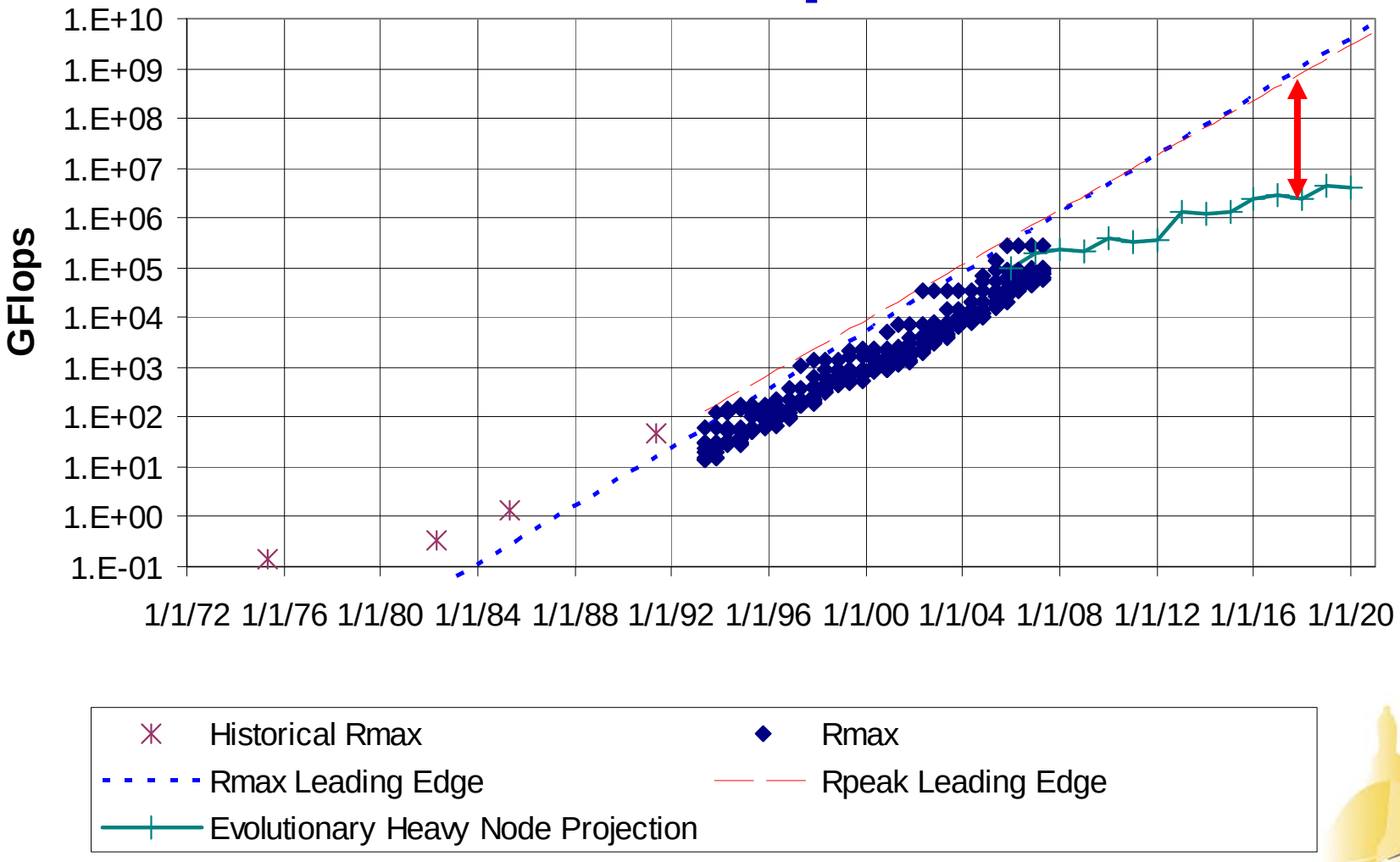
Ack. R. Murphy, SNL

Where Does This Lead Us?

- Use density increase to replicate cores
- Keep clock flat to minimize power
- Still need additional I/O for both bandwidth & latency management (reduce queuing delays by multiple banks)

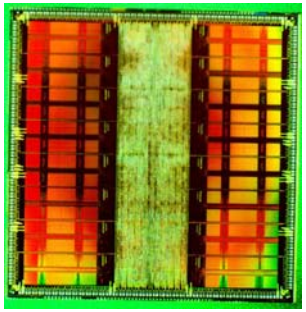


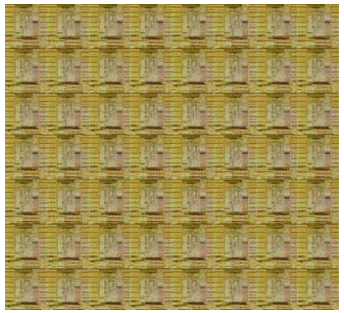
So What May This Mean to the Top 500?

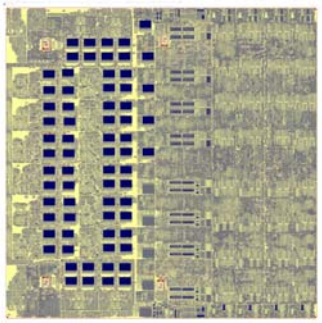


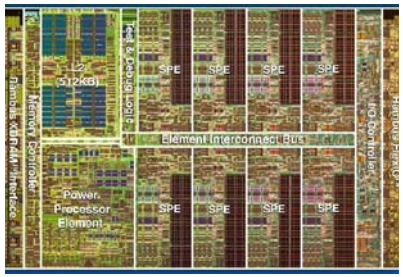
The Emergence of More Organized Architectures

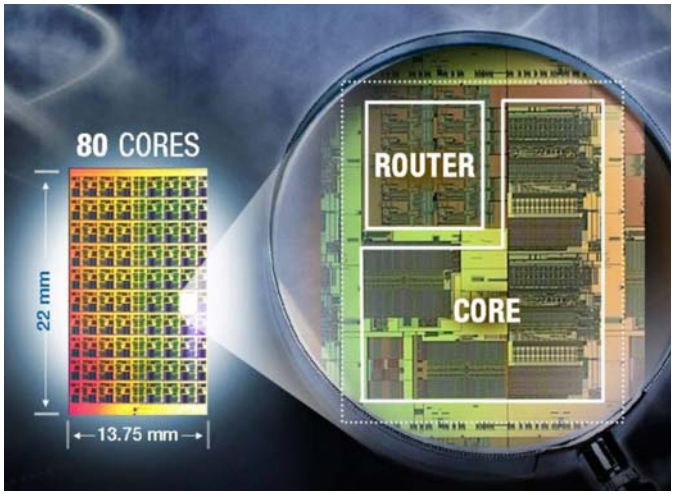
Tiling & Local Memory Regularizes Layout, Lowers Latency, Reduces Off-Chip Bandwidth Needs









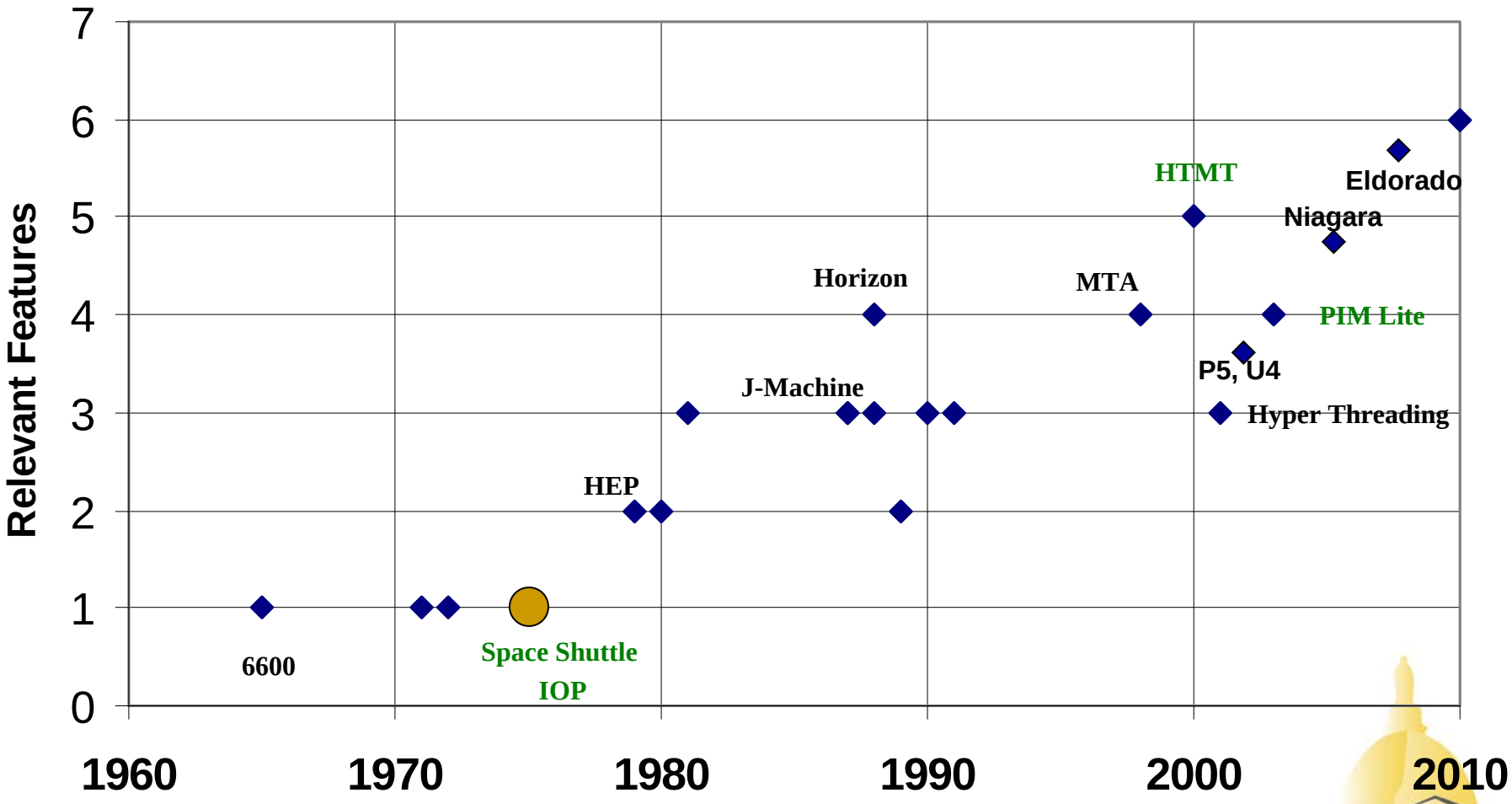


- Work well with partitionable algorithms
- Good fit for applications that support weak scaling
- Inter-core communication DOES NOT USE CONTACTS
- Compiling problem: placement of kernels AND data structures to minimize inter-core bandwidth
- Problems with global synchronization

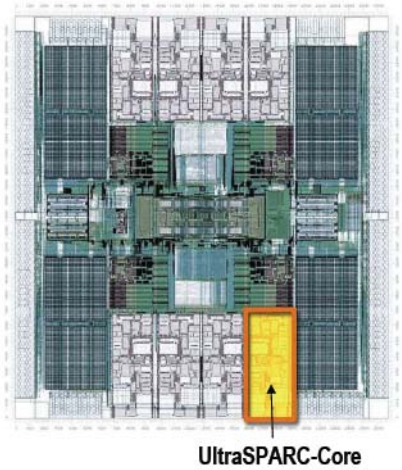
Multi-Threading

- Provide explicit latency hiding
- Permits simpler cores with more efficient use of data flow
- Increase potential for memory references “in flight”
- Shares path to memory
- But still doesn’t help “single thread” performance in terms of chained memory references
- Nor reduction of off-chip bandwidth (and contacts)

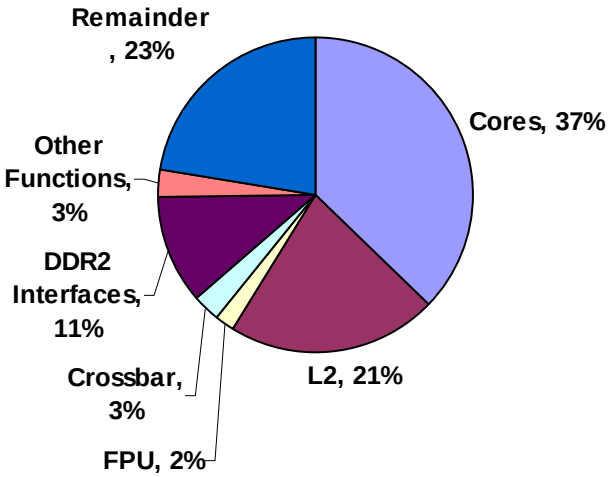
A Brief History of Multi-threaded Processors

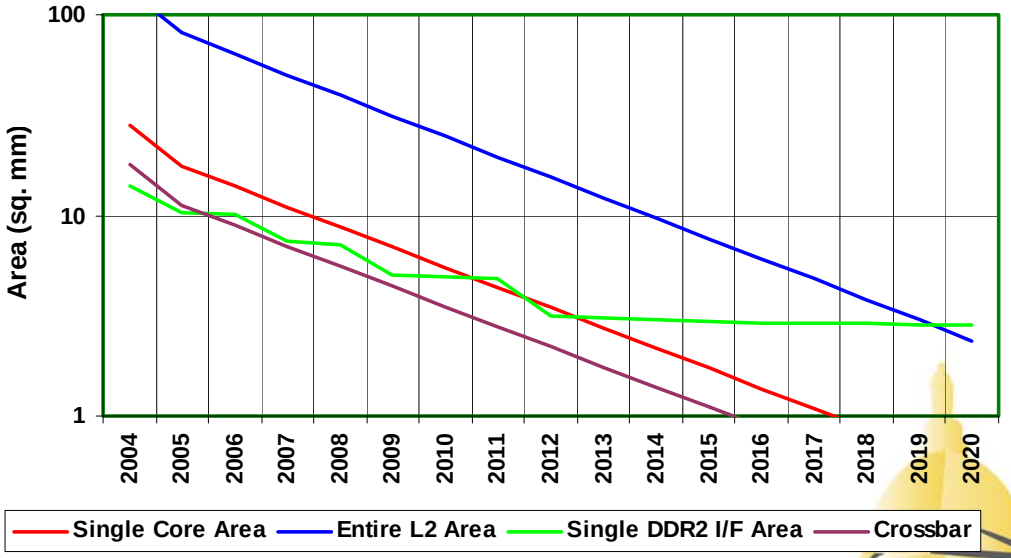


Sun's Niagara

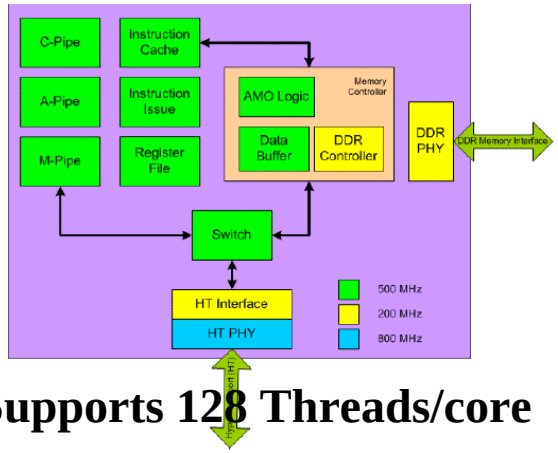


- 8 4-way multi-threaded single issue cores
- 3MB 12 bank shared L2
- 4 DDR2 Memory Interfaces
- Measured 5.76 IPC vs Peak of 8 on Java Business B/M
- 63W @90nm (2W cores)





Cray's XMT



Supports 128 Threads/core

Figure 2. MT processor block diagram

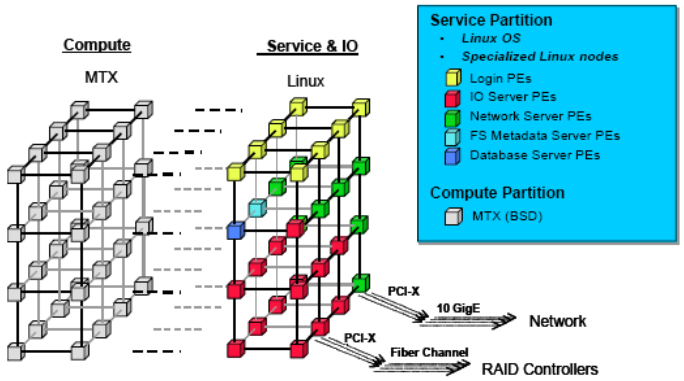


Figure 1. Eldorado system architecture

Table 3. Sparse matrix-vector multiply

System	T (sec.)
IBM Power4 1.7 GHz (1 P)	26.10
MTA-2 (1 P)	7.11
MTA-2 (2 P)	3.59
MTA-2 (4 P)	1.83
MTA-2 (8 P)	0.94
Eldorado (576 P) *(estimated)*	0.043
Eldorado (2112 P) *(estimated)*	0.016
Eldorado (8064 P) *(estimated)*	0.006

Table 5. RandomAccess

System	Giga updates per second
Cray X1 800 MHz (60 P)	0.0031
IBM Power4 1.7 GHz (256 P)	0.0055
MTA-2 (2 P)	0.041
MTA-2 (5 P)	0.204
MTA-2 (10 P)	0.405
Eldorado (576 P) *(estimated)*	17.32
Eldorado (2112 P) *(estimated)*	47.57
Eldorado (8064 P) *(estimated)*	121.0

Table 4. Linked-list search

System	Time (sec) N = 5000	Time (sec) N = 10,000
SunFire 880 MHz (1 P)	9.3	107.0
Intel Xeon 2.8GHz (1 P)	7.15	40.0
MTA-2 (1 P)	0.485	1.98
MTA-2 (2 P)	0.053	0.197
Eldorado (576P) *(estimiated)*	0.0014	0.0058
Eldorado (2112 P) *(estimiated)*	0.0005	0.0020
Eldorado (8064 P) *(estimiated)*	0.0002	0.0008

John Feo, David Harper, Simon Kahan, Petr Konecny, "Eldorado", Computing Frontiers, 2005

Some Interesting Comparisons

Core	L1	FPU	Area	pJ/~
Niagara-I	24	No	11.92	1719
Niagara-II	24	yes	23.85	2364
MIP64	64	yes	9.59	
MIPS64	40	No		436

So Multi-Threading is not Free

Problems Still Remain

- Programming models not changed
- States still very heavy
- Compiling to specific cores
- Data partitioning
- Problems with coherency
- Doesn't address barriers, sync points, …
- Doesn't help emerging low reuse apps
  - AMR
  - Data mining
  - Graph traversals
  - Non-numeric solvers such as SAT

Are We Ready for a Mutation?

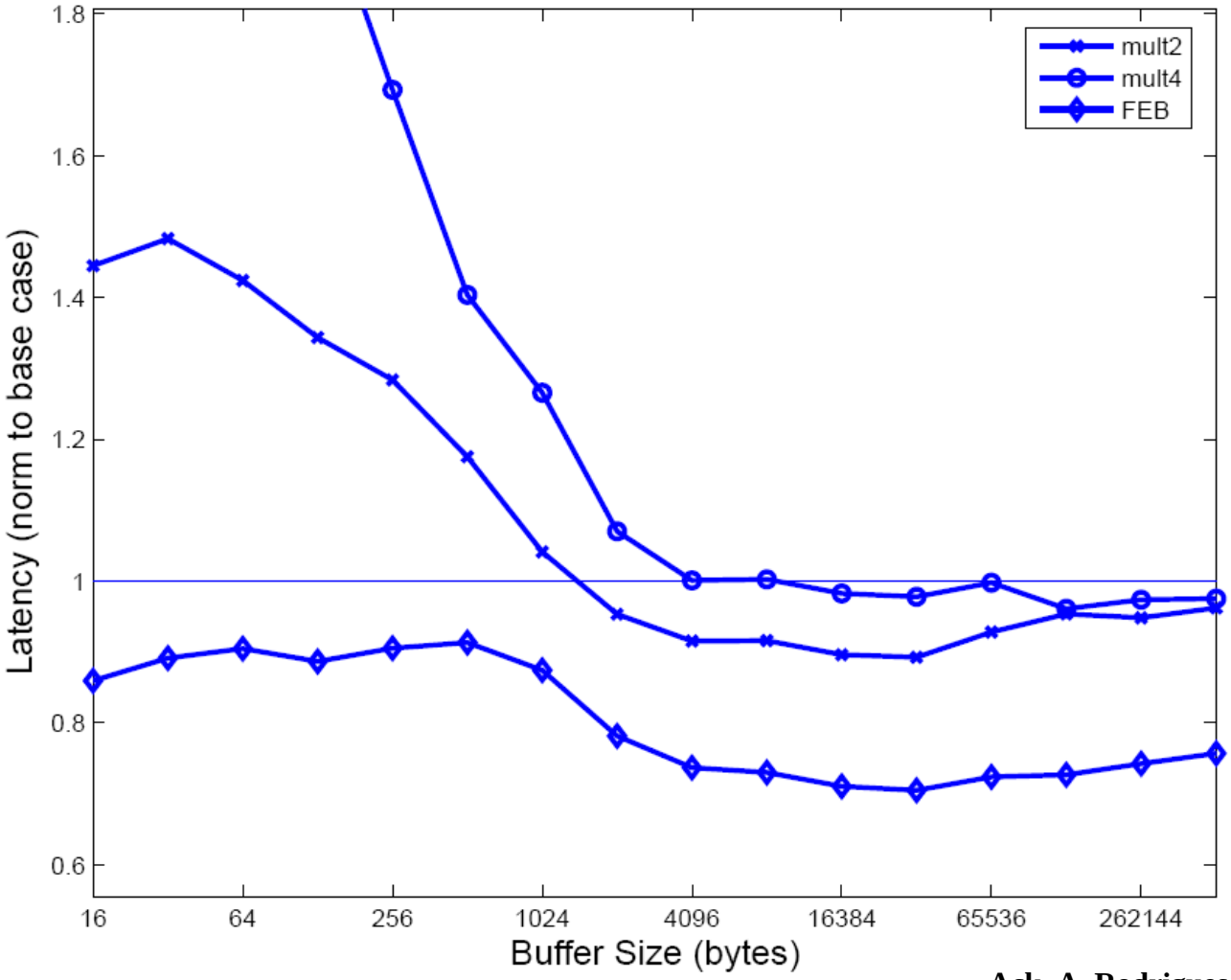
Ideas

- Ultra light weight “butterflies” take functions to the data flowers
  - Memory reference becomes “traveling threadlet”
- But, like flowers, data can respond to the touch of the butterfly.
  - Add small amount of metadata to each word
- Finally, it’s the “flowers” whose location is important

Adding Metadata to the Memory

- "Special Values"
  - Uninitialized, error code, null
- Full/Empty bits
  - And multiple flavors of "empty"
  - Esp. "empty pending outstanding value"
  - Greatly simplifies Producer/Consumer
- Forwarding
- Locked
- Traps
- Especially interesting when aliased to thread state registers

Full/Empty Bits & MPI



Ack. A. Rodrigues, SNL

One Step Further: Allowing the Threads to Travel

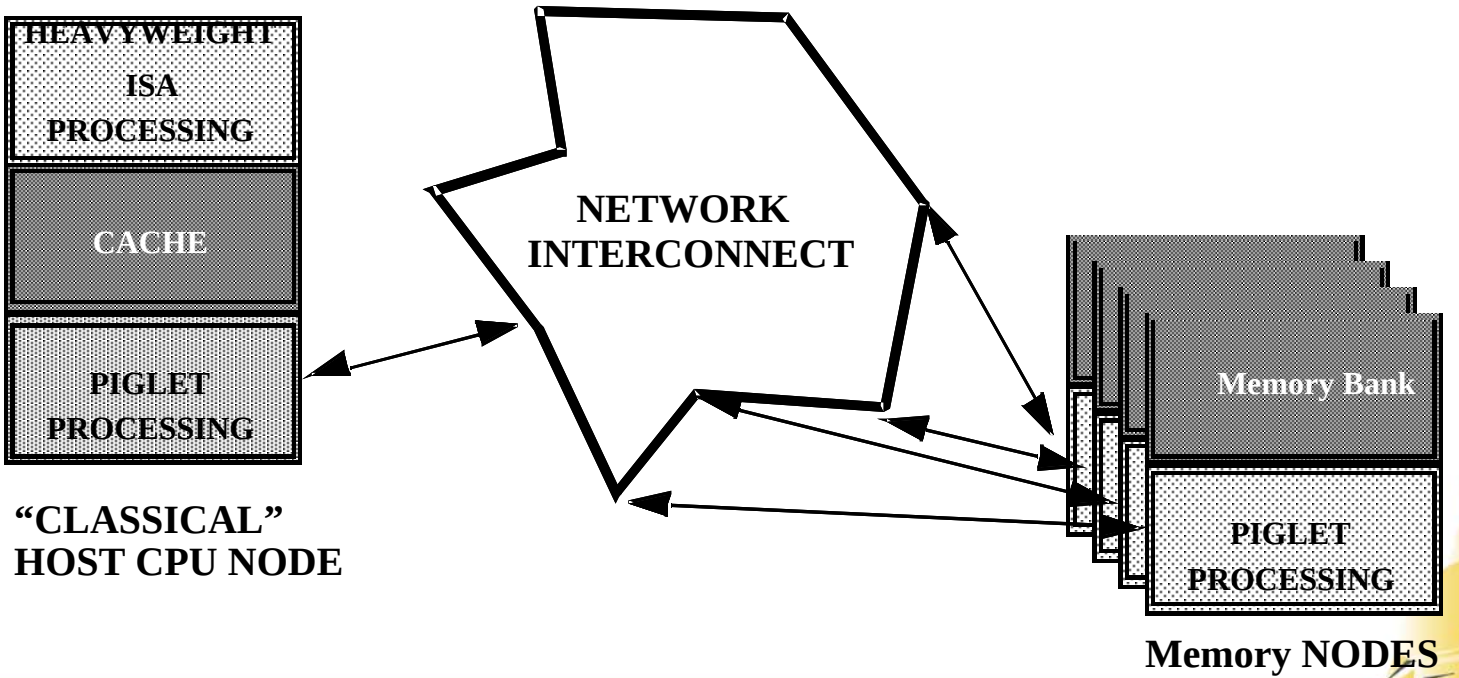
- "Overprovision" memory with huge numbers of anonymous execution sites
  - Place at bottom of, or near, memory
- Reduce state of a thread to a memory reference
- Make creating a new thread "near" some memory a cheap operation
- Allow thread to "move" to new site when locality demands
- Don't require target to maintain code

Latency reduced by *huge* factors

"Piglet" Processing At Base of Memory

Target Address	Operands & Working Registers	PC	Code
Additional Data Payload			

THREADLET FORMAT



Types of Piglet Programs

- Classical memory operations
- Atomic Memory Operations
- Short Vector to Memory
- “Object-oriented” method evaluation at the object
- Small slices of programs

Example: AMO

- AMO = Atomic Memory Operation
  - Update some memory location
  - With guaranteed no interference
  - And return result
- Parcel Registers: A=Address, D=Data, R=Return Address
- Sample Code:

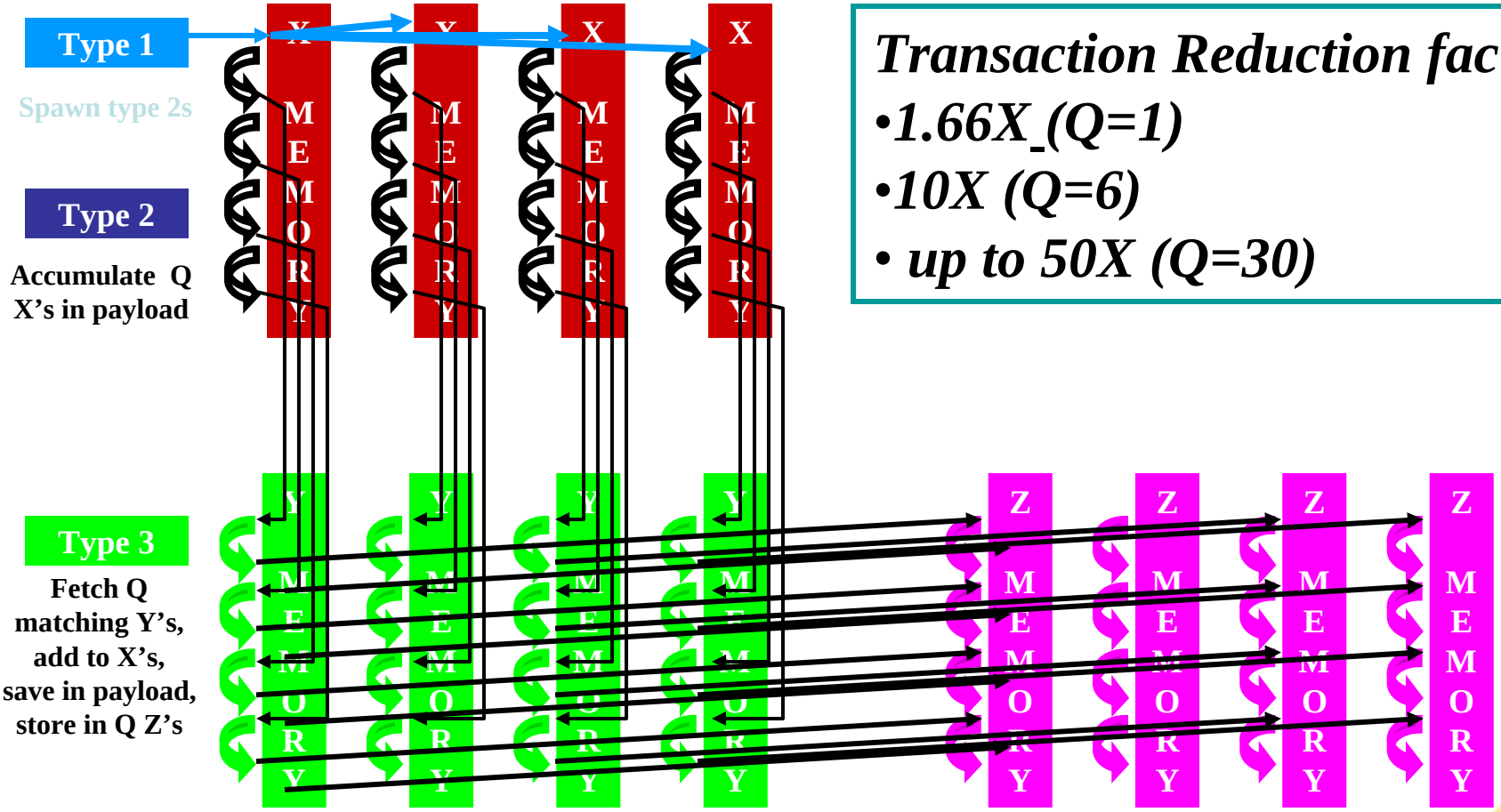
```
MOVE
L1: LOCK & LOAD
OP
STORE & RELEASE L1
SWAPRA
MOVE A
STORE
QUIT
```

Atomic Update "At the Memory"

Return Result

Bottom Line: 2 network transactions rather than up to 6!

Vector Add (Z[I]=X[I]+Y[I]) via Threadlets



Transaction Reduction factor:

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

Stride thru Q elements

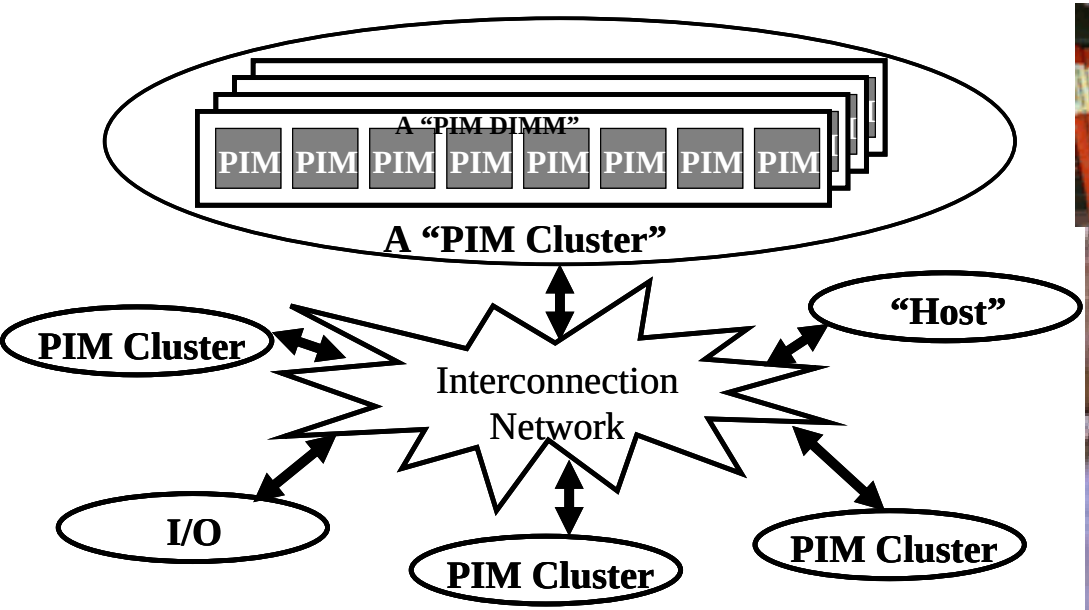
Conclusions

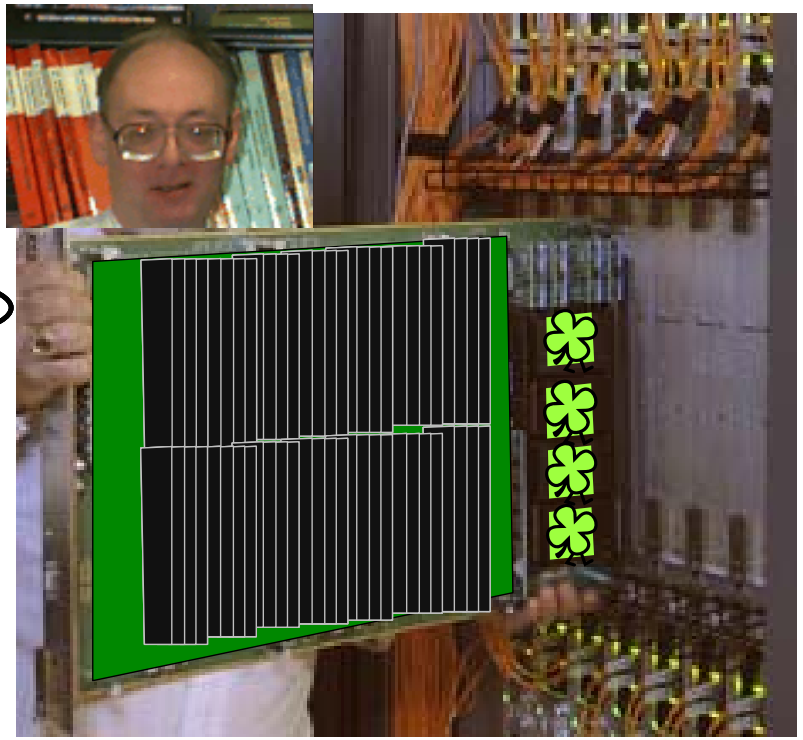
Conclusions

- (Hierarchical) Multi-core has taken over
  - But clock rate will be limited by power
  - And # of useable cores by contacts
- Simpler cores: more area/energy efficient
  - But we can't use all them in hierarchical architectures
- Latency will stifle single-thread performance
- Multi-threading provides better utilization
  - But at an energy cost
- Pipelined/Array chips reduce need for off-chip bandwidth
  - But then run into power-limiting clock problem
  - And require 2D data/code partitioning of code
- Are there alternatives that don't fix code to cores?

BEST HPC Architecture != Best commodity architecture

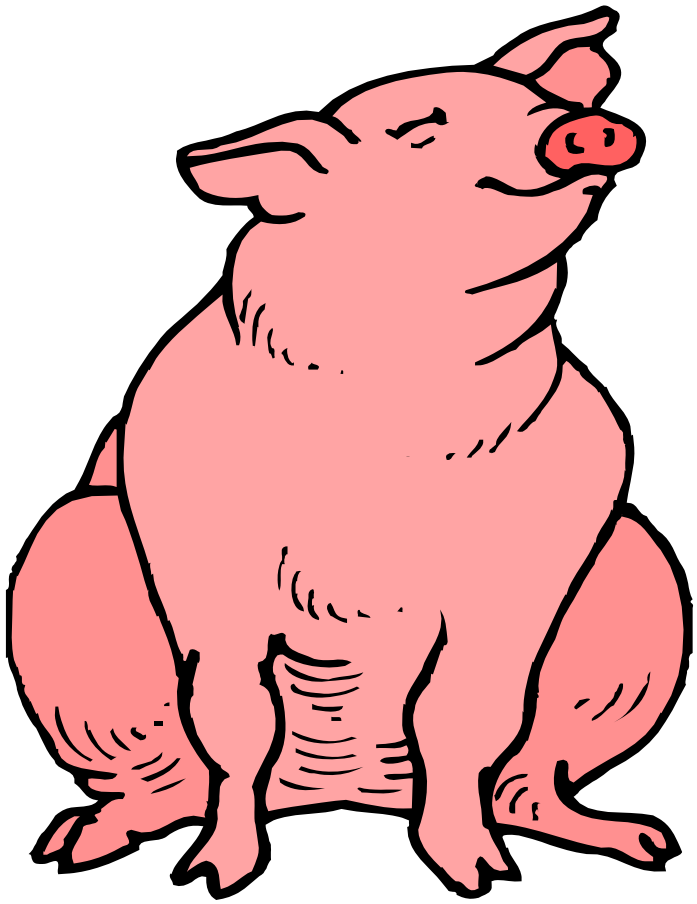
A Personal Goal





- Huge increase in silicon per board
- Level out power dissipation

The Future



Will We Design Like This?

Or This?