

Advanced Architectures & Execution Models:

How New Architectures May Help Give
Silicon Some Temporary New Life
And Pave the Way for New Technologies

Peter M. Kogge

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

Assoc. Dean for Research, University of Notre Dame

IBM Fellow (ret)





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

ILP: Getting tougher & tougher to increase

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

Concurrency = Throughput

Latency

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

*Much less than peak
and degrading rapidly*



Why Is Zettaflops Even Harder?

- **Silicon density:** Sheer space taken up implies large distances & loooooong latencies
- **Silicon mindset:**
 - Processing logic “over here”
 - Memory “over there”
 - And we add acres of high heat producing stuff to bridge the gap
- **Thesis:** how far can we go with a *mindset* change



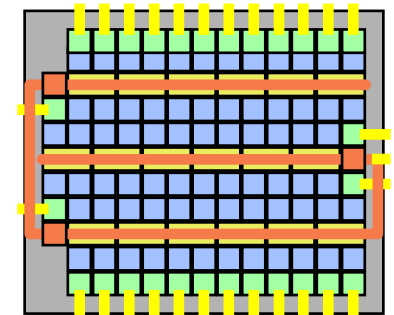
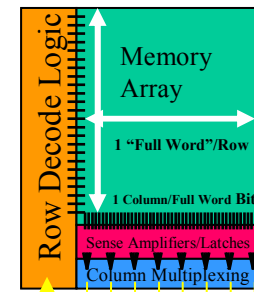
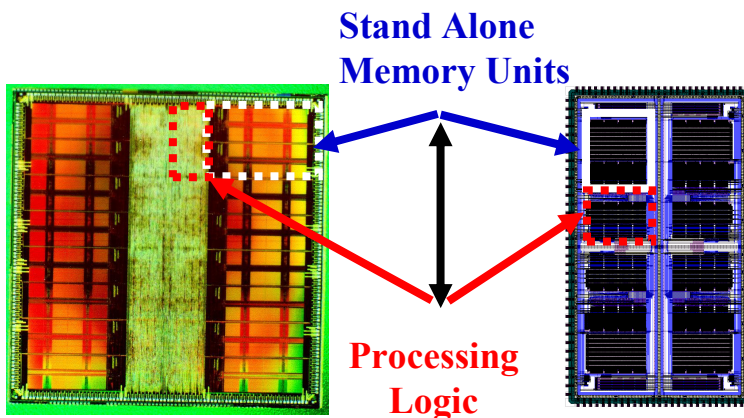
This Talk: Climbing the Wall a “Different Way”

- **Enabling concepts implementable in silicon:**
 - **Processing In Memory**
 - Lowering the wall, both bandwidth & latency
 - **Relentless Multi-threading with Light Weight Threads**
 - to *change* number of times we must climb it
 - to *reduce* the state we need to keep behind
- **Finding architectures & execution models that support both**
- **With emphasis on “Highly Scalable” Systems**

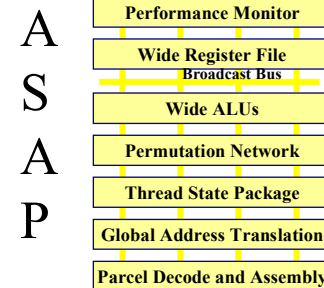


“Processing-In-Memory”

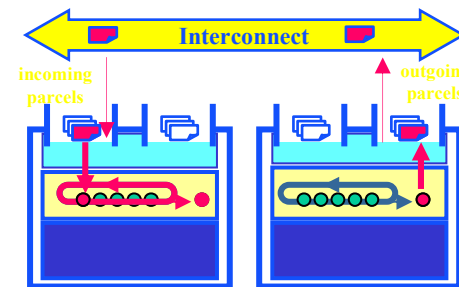
- High density memory on same chip with high speed logic
- Very fast access from logic to memory
- Very high bandwidth
- ISA/microarchitecture designed to utilize high bandwidth
- Tile with “memory+logic” nodes



Tiling a Chip



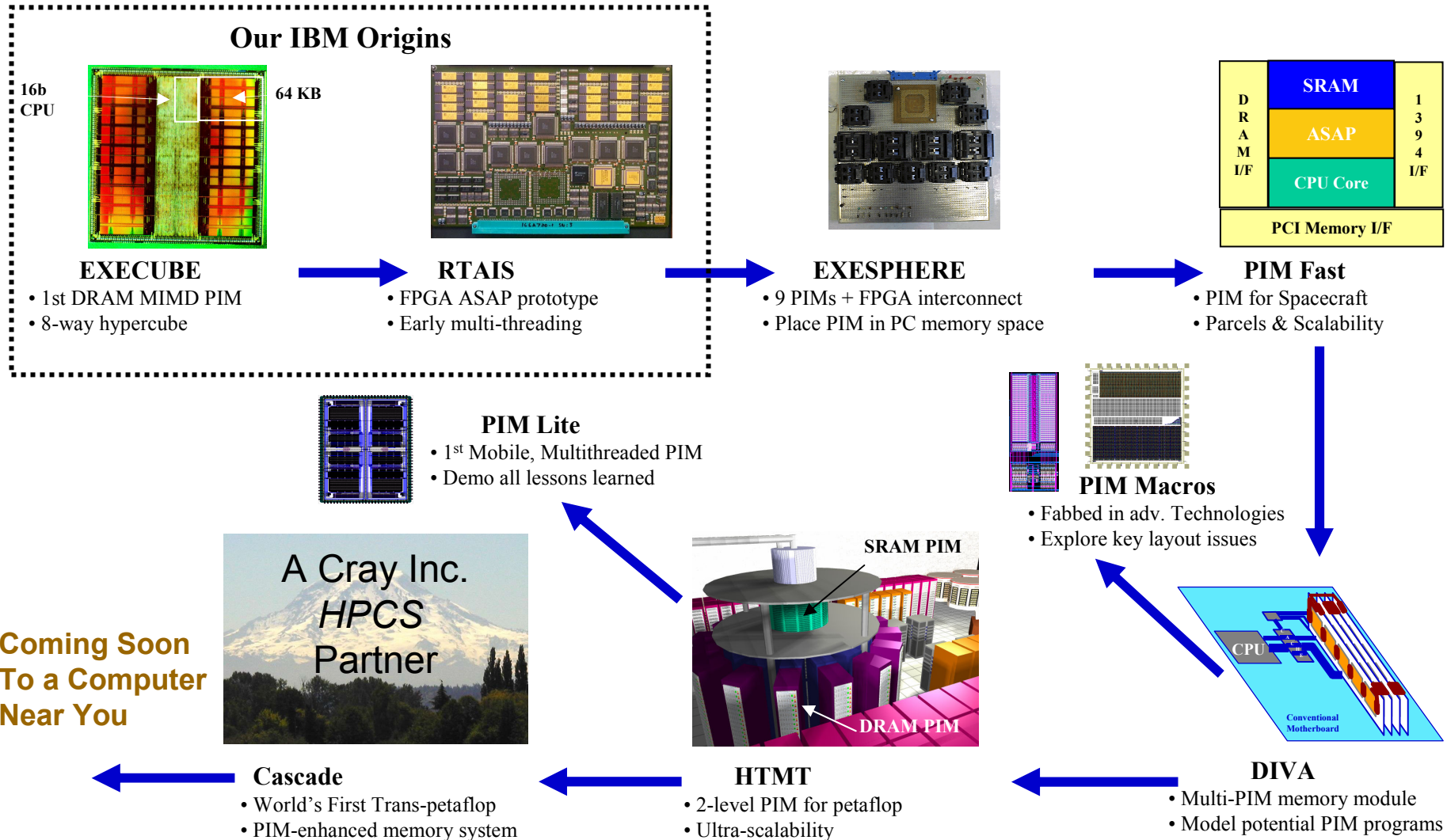
A Memory/Logic Node



Parcel = Object Address + Method_name + Parameters



A Short History of PIM @ ND



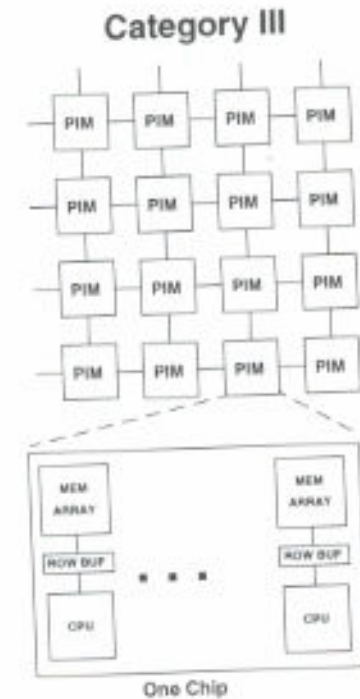
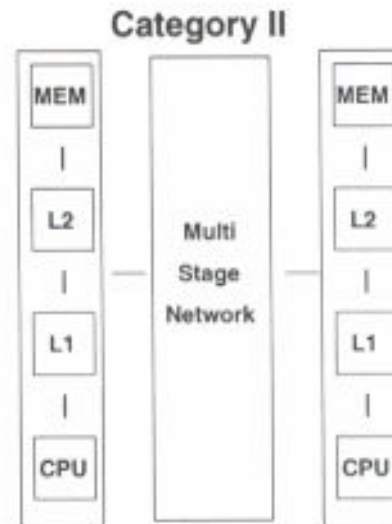
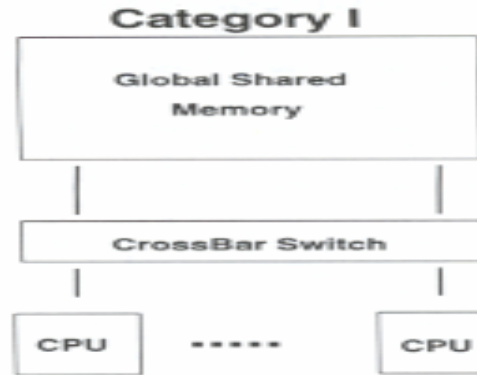


Acknowledgements

- My personal work with PIM dates to late 1980s
- But also! ND/CalTech/Cray collaboration now a decade old!



Architecture Working Group
1st Workshop on Petaflops Computing
Pasadena, Ca
Feb. 22-24, 1994





Topics

- **How We Spend Today's Silicon**
- **The Silicon Roadmap – A Different Way**
- **PIM as an Alternate Technology**
- **PIM-Enabled Architectures**
- **Matching ISA & Execution Models**
- **Some Examples**

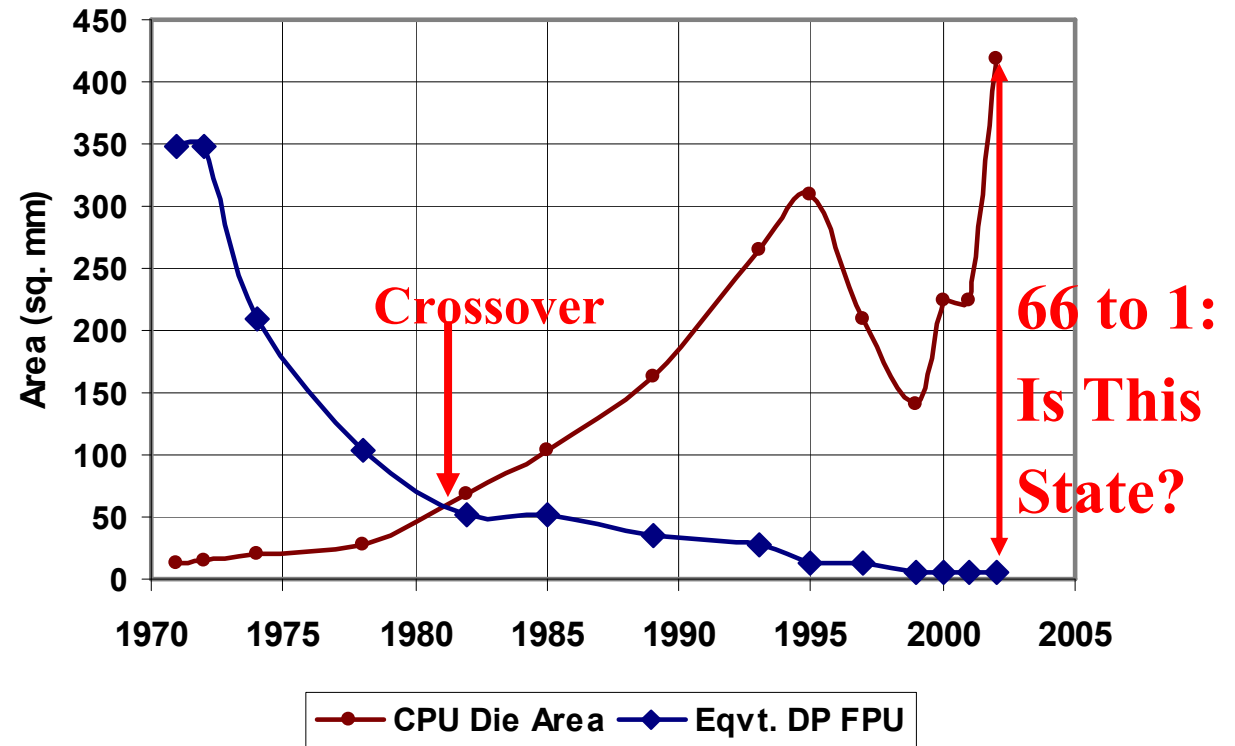
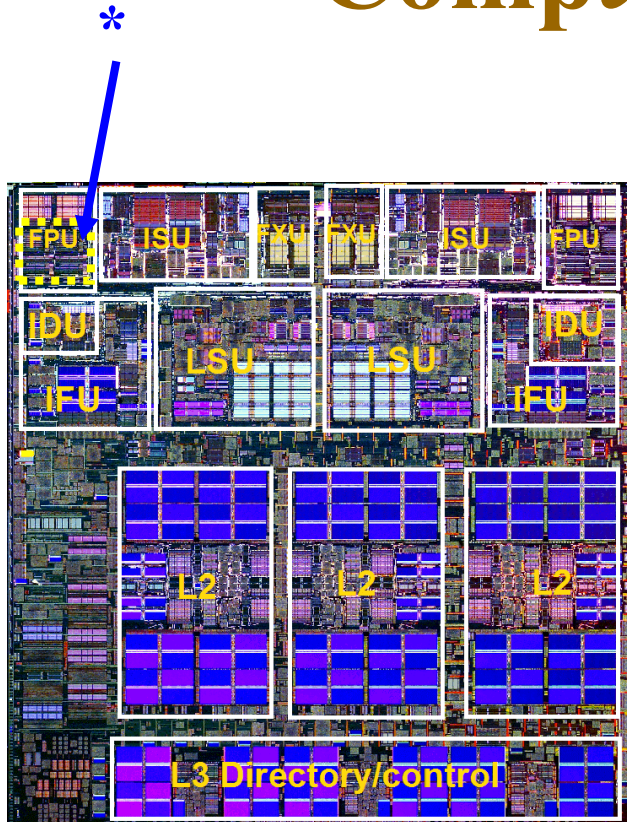


How We Spend Our Silicon Today: Or “The State of State”



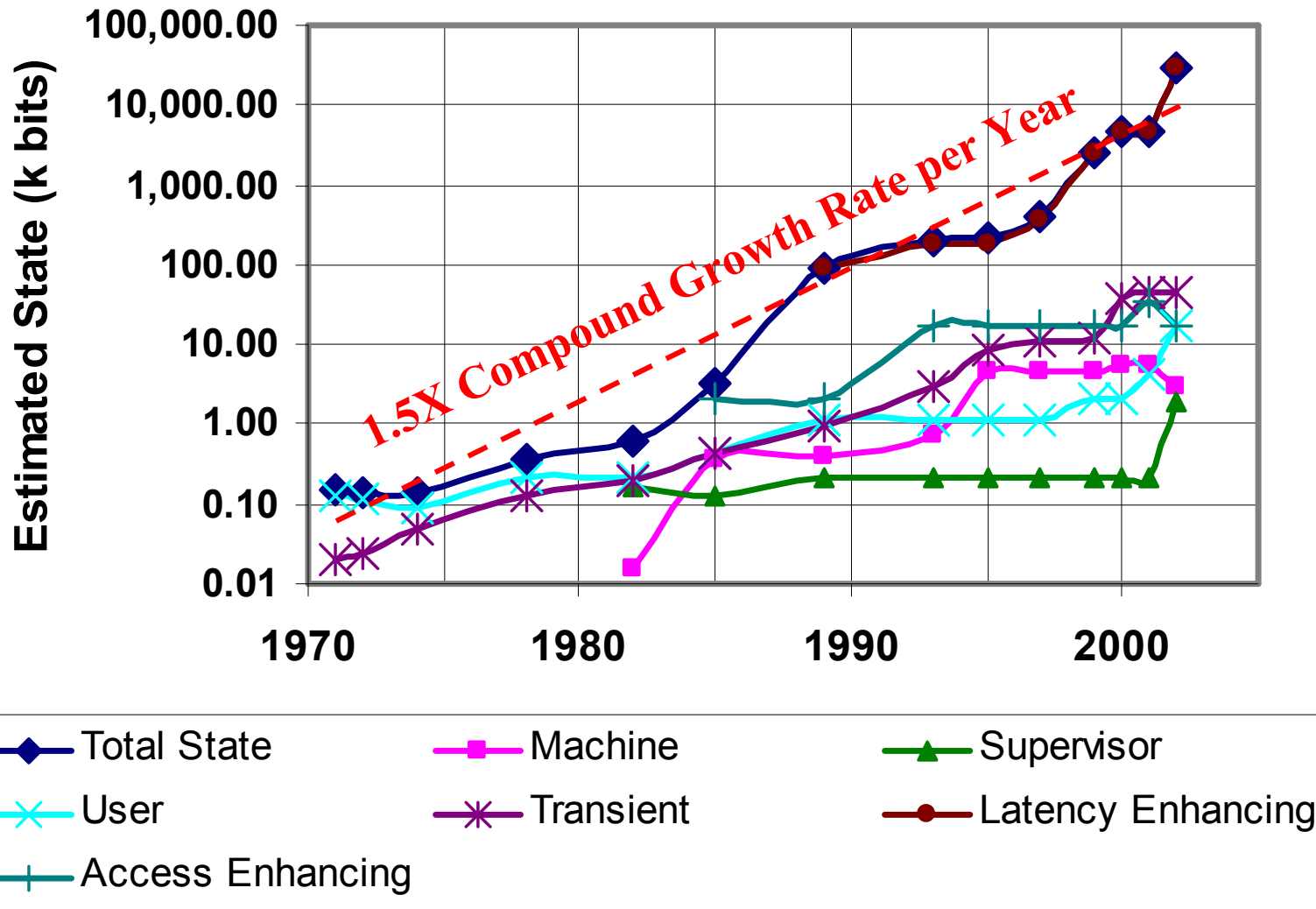
How Are We Using Our Silicon?

Compare CPU to a DP FPU



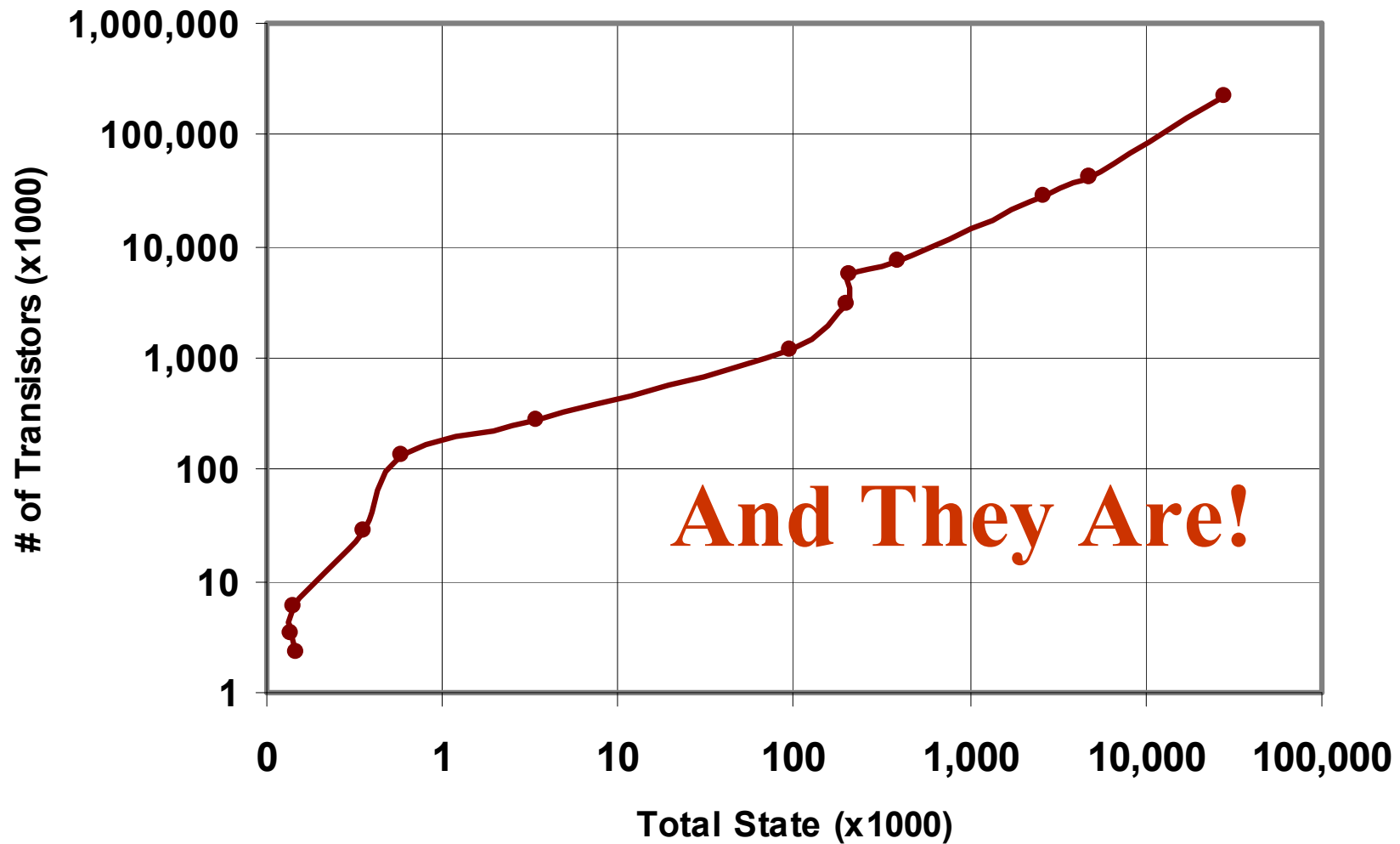


CPU State vs Time





So We Expect State & Transistor Count to be Related



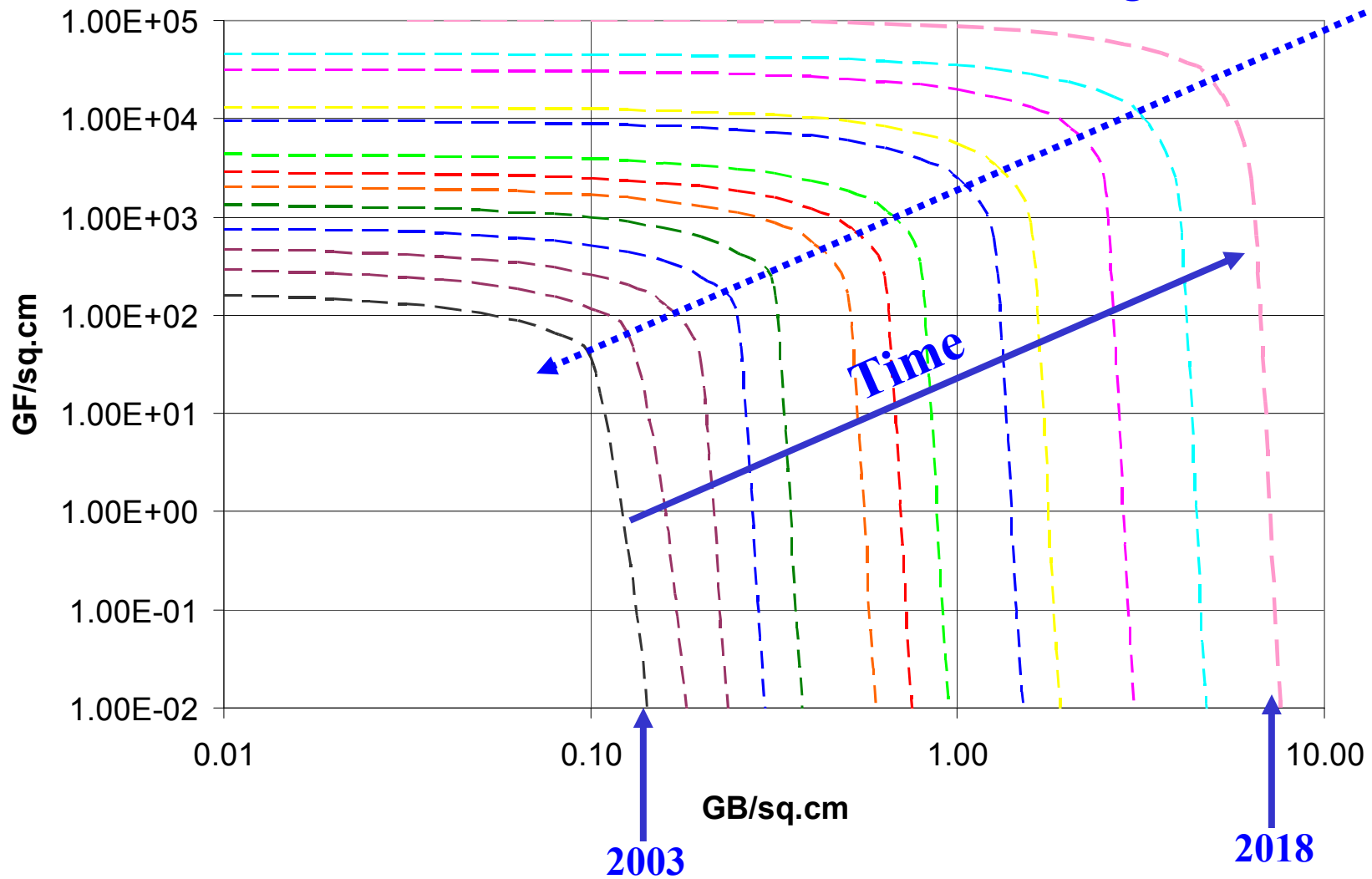


**The Silicon Roadmap:
Or
“How are we Using an Average
Square of Silicon?”**



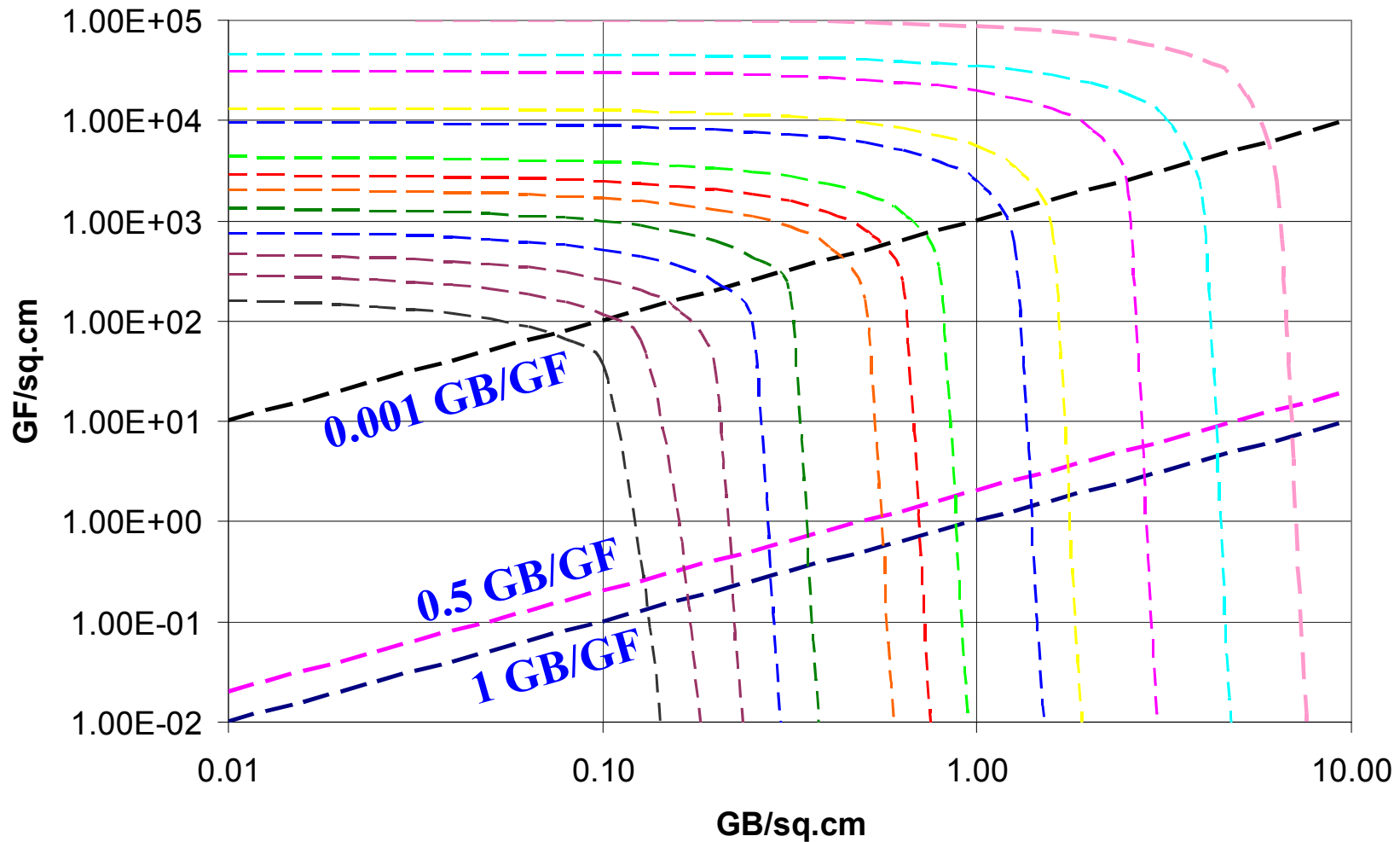
The Perfect “Knee Curves” No “Overhead” of *Any* Kind

“Knee”: 50% Logic & 50% Memory



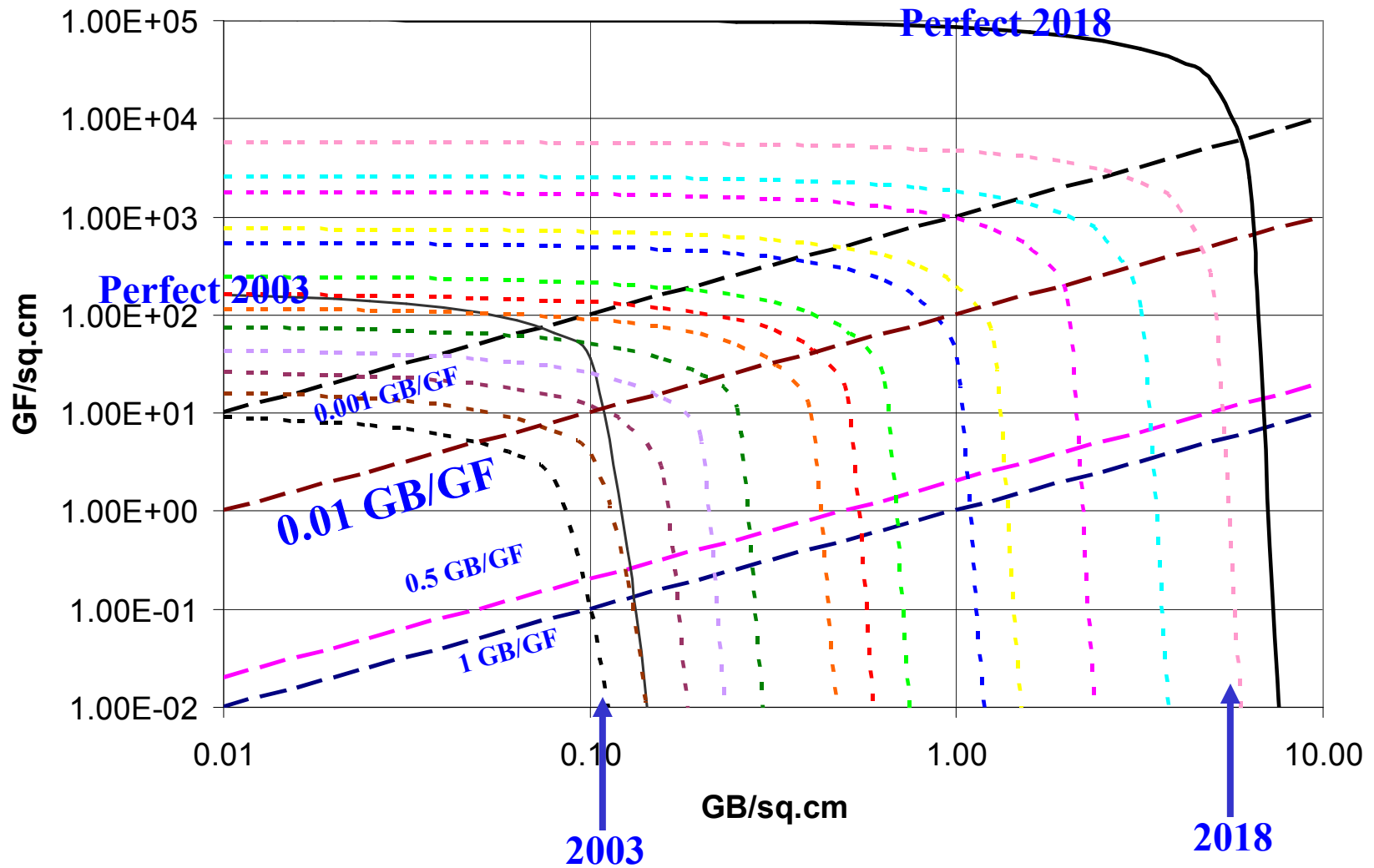


Adding In “Lines of Constant Performance”



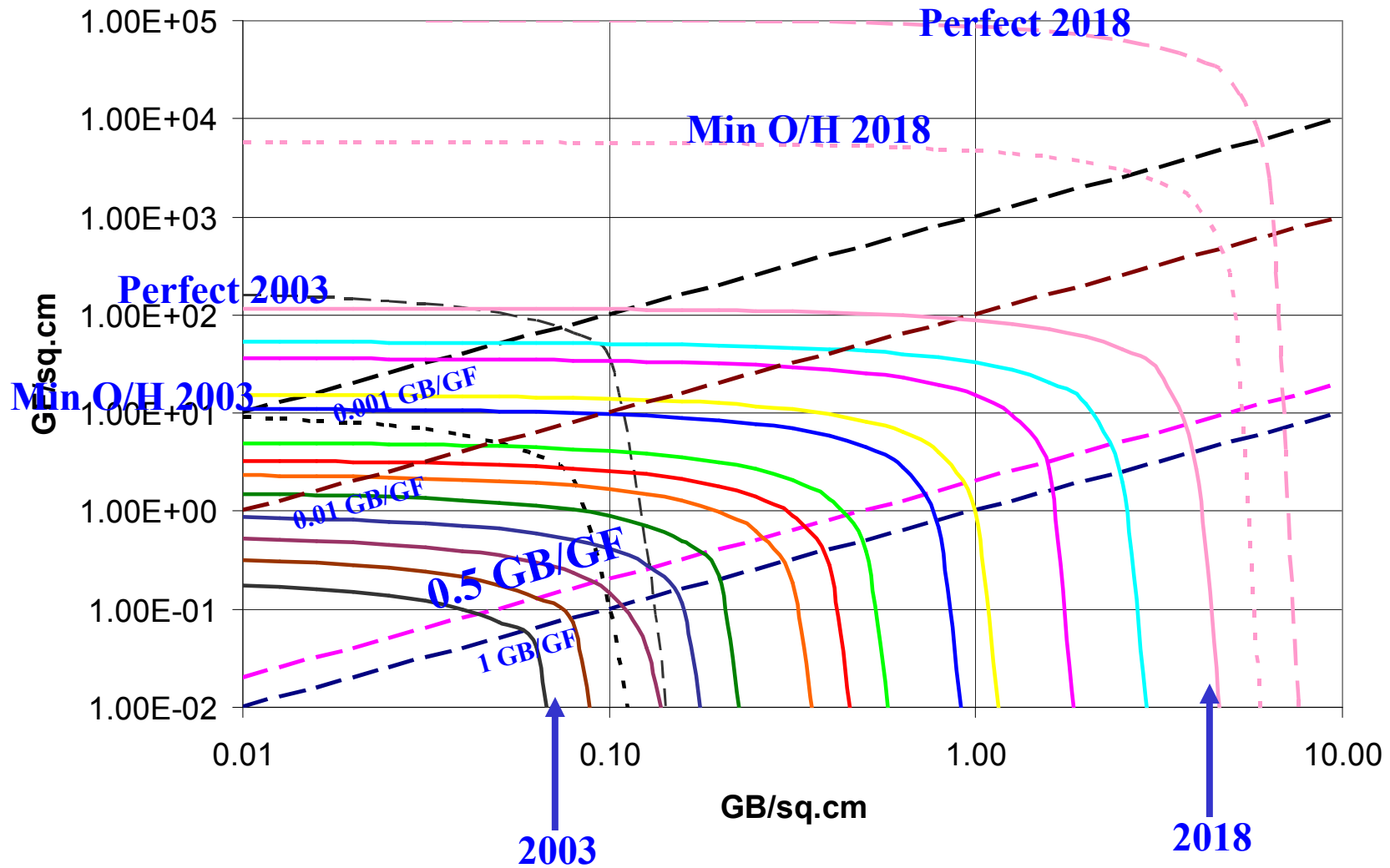


What if We Include Basic Overhead?



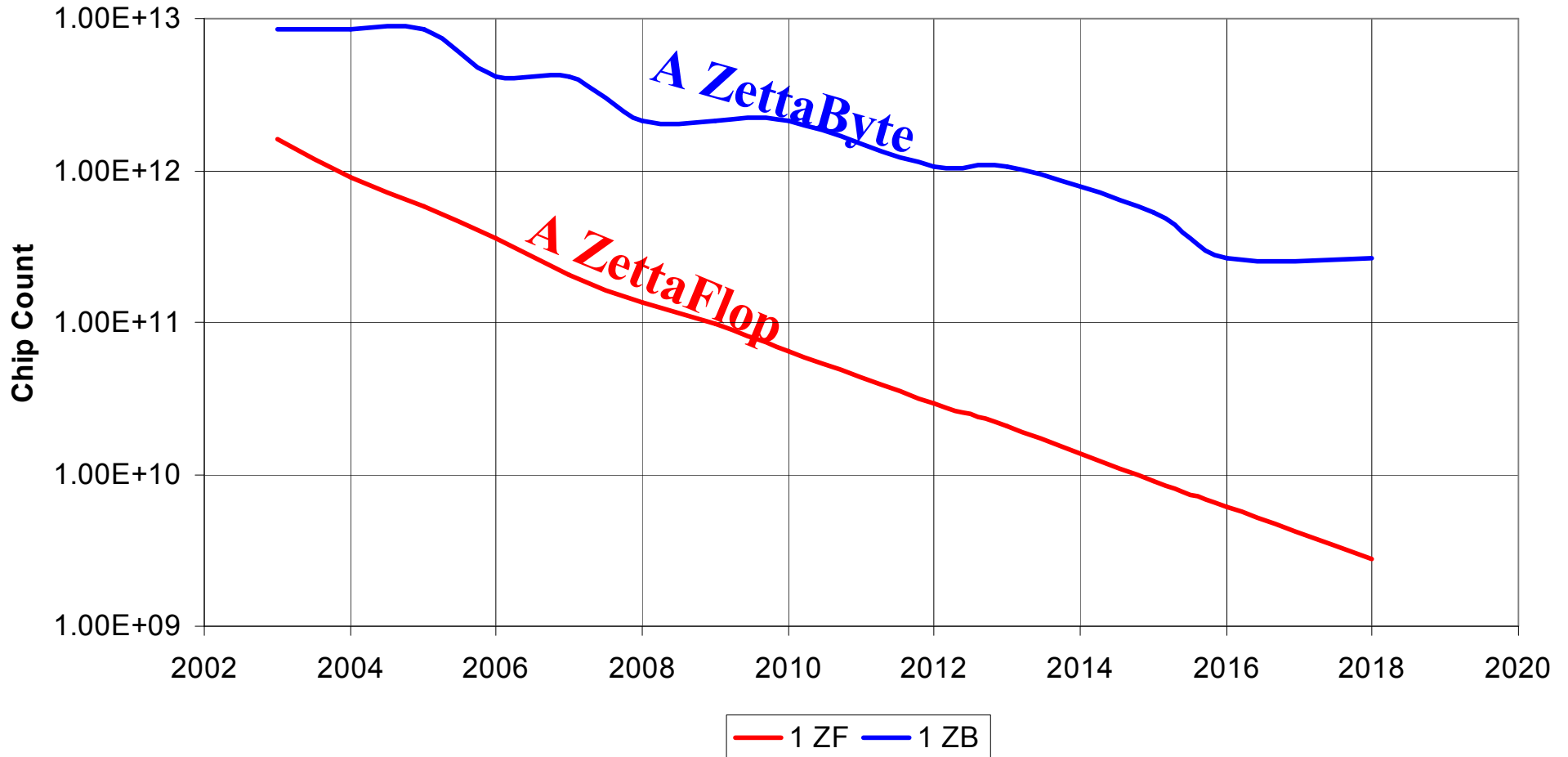


What If We Look At Today's Separate Chip Systems?





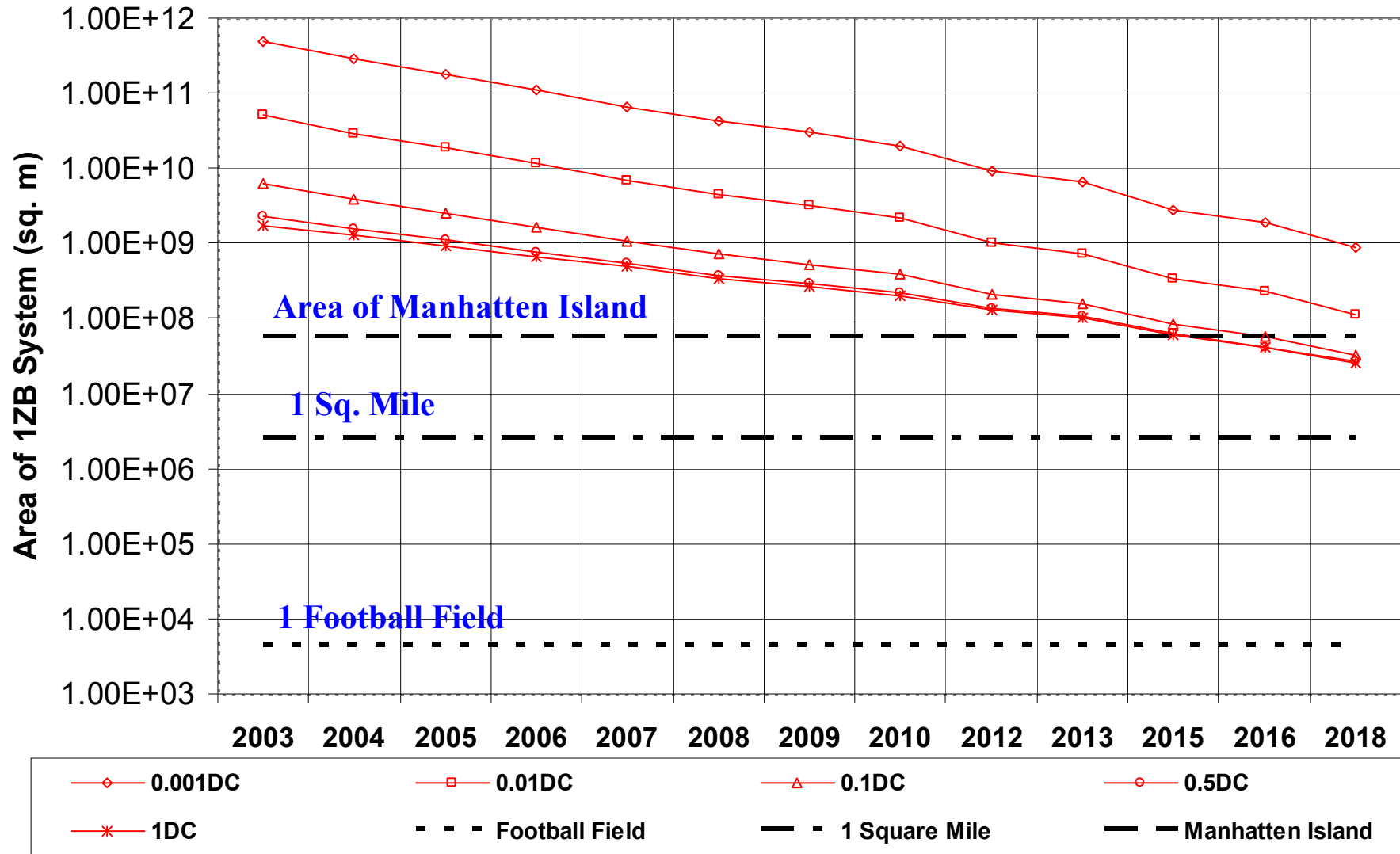
How Many of Today's Chips Make Up a "Zetta"?



And This Does Not Include "Routing"

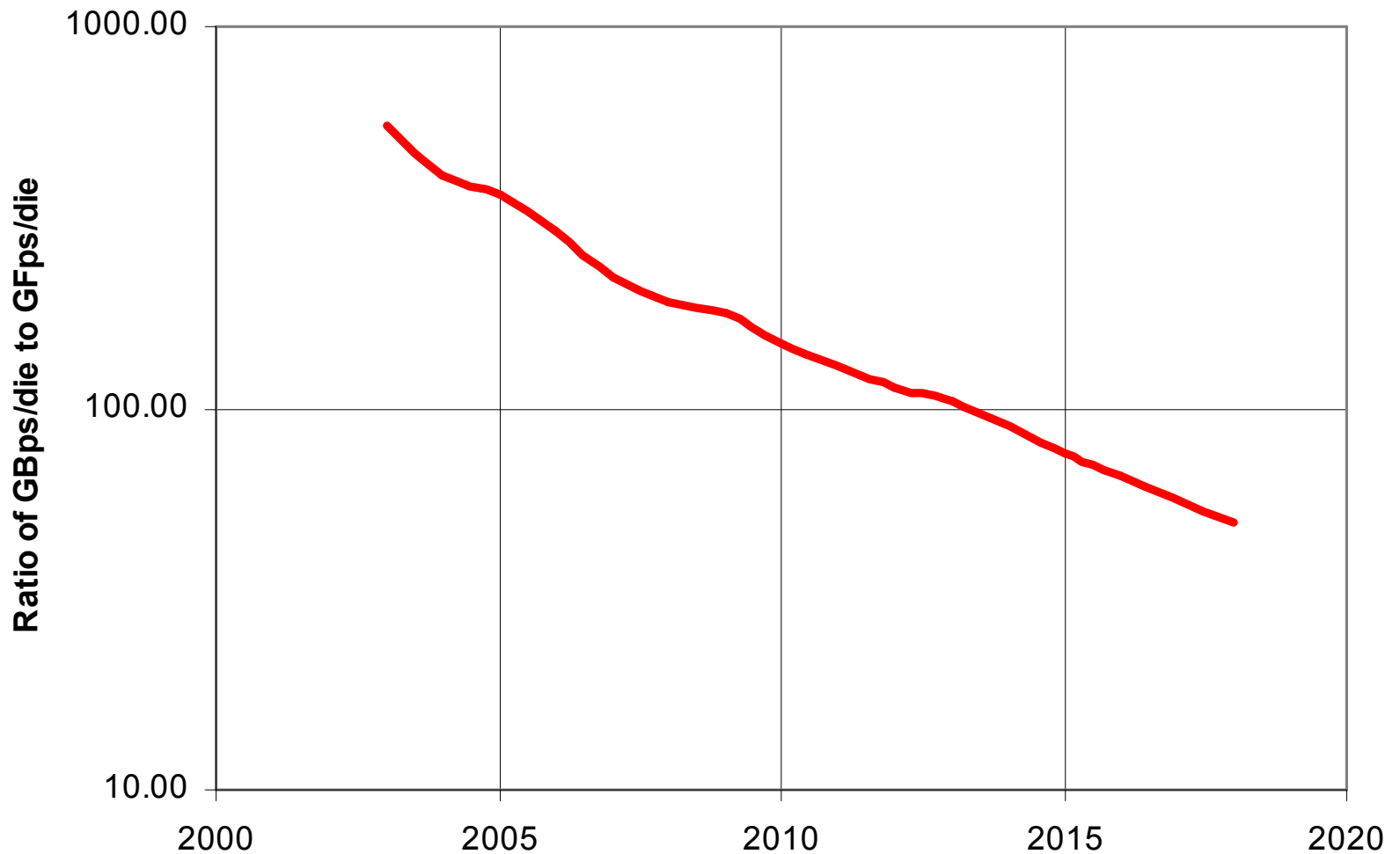


How Big is a 1 ZB System?





How Does Chip I/O Bandwidth Relate to Performance?





Problems

- **Complexity & Area Infeasible**
- **Flop numbers assume perfect utilization**
- **But latencies are huge**
 - **Diameter of Manhattan = 28 microseconds**
- **And efficiencies will plummet**
 - **At 0.1% efficiency we need area of Rhode Island for microprocessor chips**
 - Whose diameter is 240 microseconds
- **And we don't have enough pins!**



PIM as an Alternative Technology

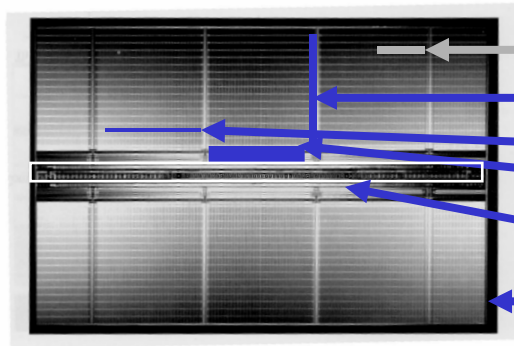


PIM Objective

- **Move processing logic onto dense DRAM die**
- **Obtain decreased latency & increased latency for local reference**
 - **Without needing pins**
- **AND simplify logic down to a simple core**
- **Thus allowing many more “processors”**
- **And off chip pins used for true remote reference**

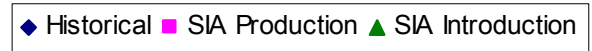
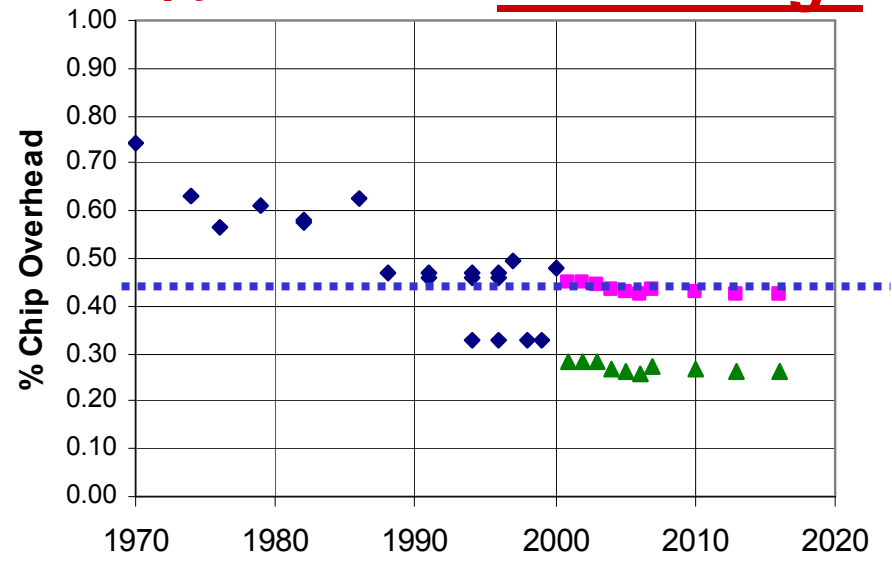
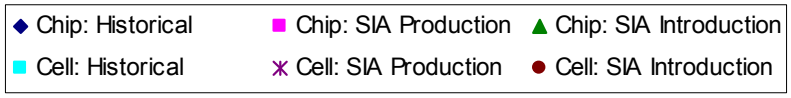
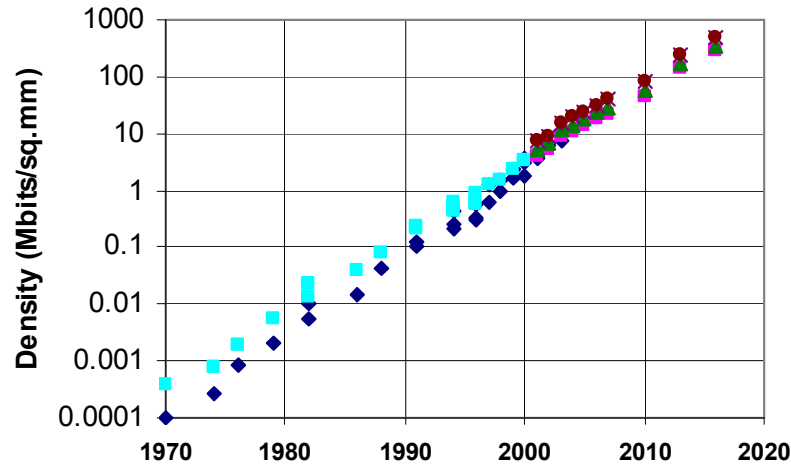


Classical DRAM



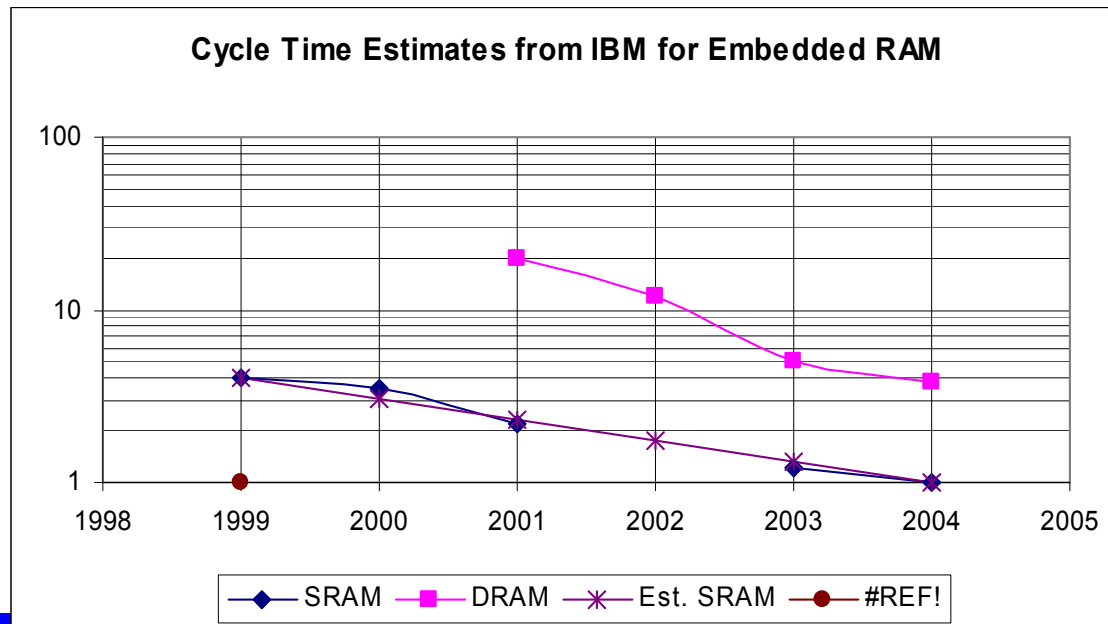
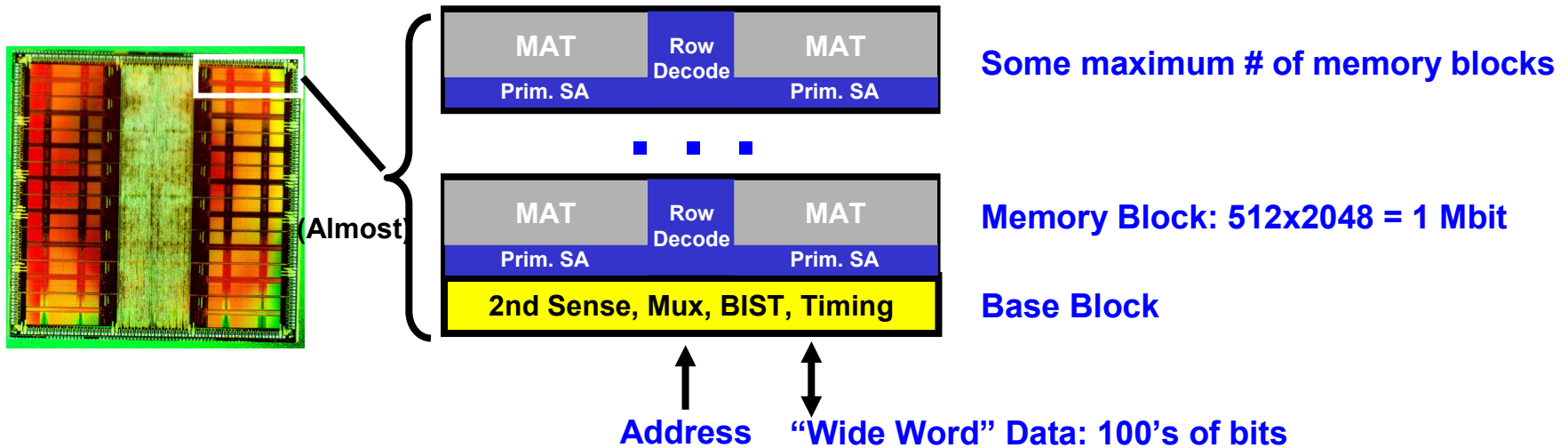
- Memory mats: ~ 1 Mbit each
- Row Decoders
- Primary Sense Amps
- Secondary sense amps & “page” multiplexing
- Timing, BIST, Interface
- Kerf

45% of Die is non storage





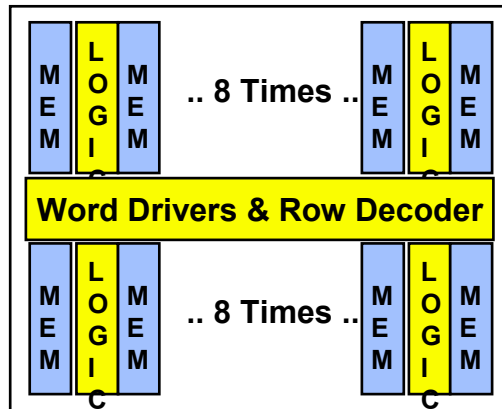
Embedded DRAM Macros Today



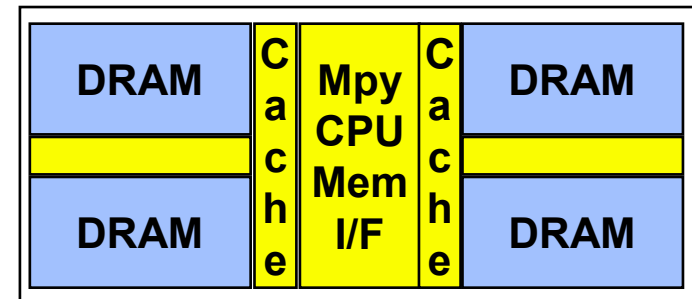


PIM Chip

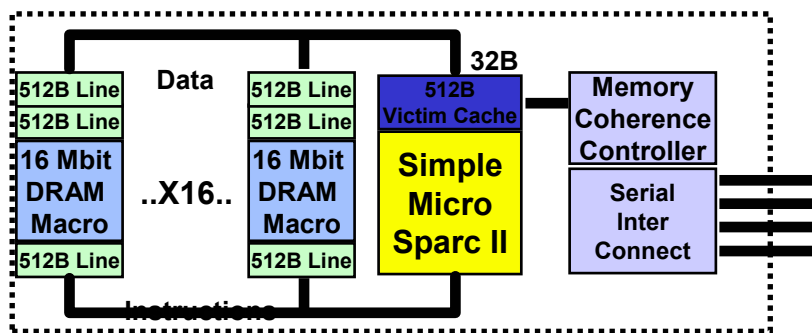
MicroArchitectural Spectrum



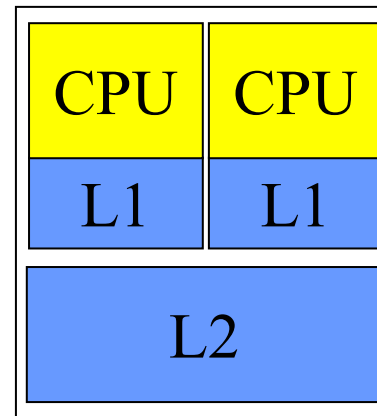
SIMD: Linden DAAM



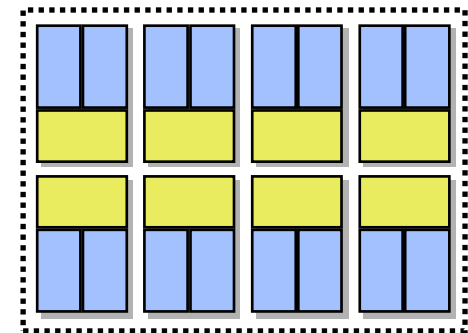
**Single Chip Computer:
Mitsubishi M32R/D**



**Complete SMP Node:
Proposed SUN part**



**Chip Level SMP:
POWER4**

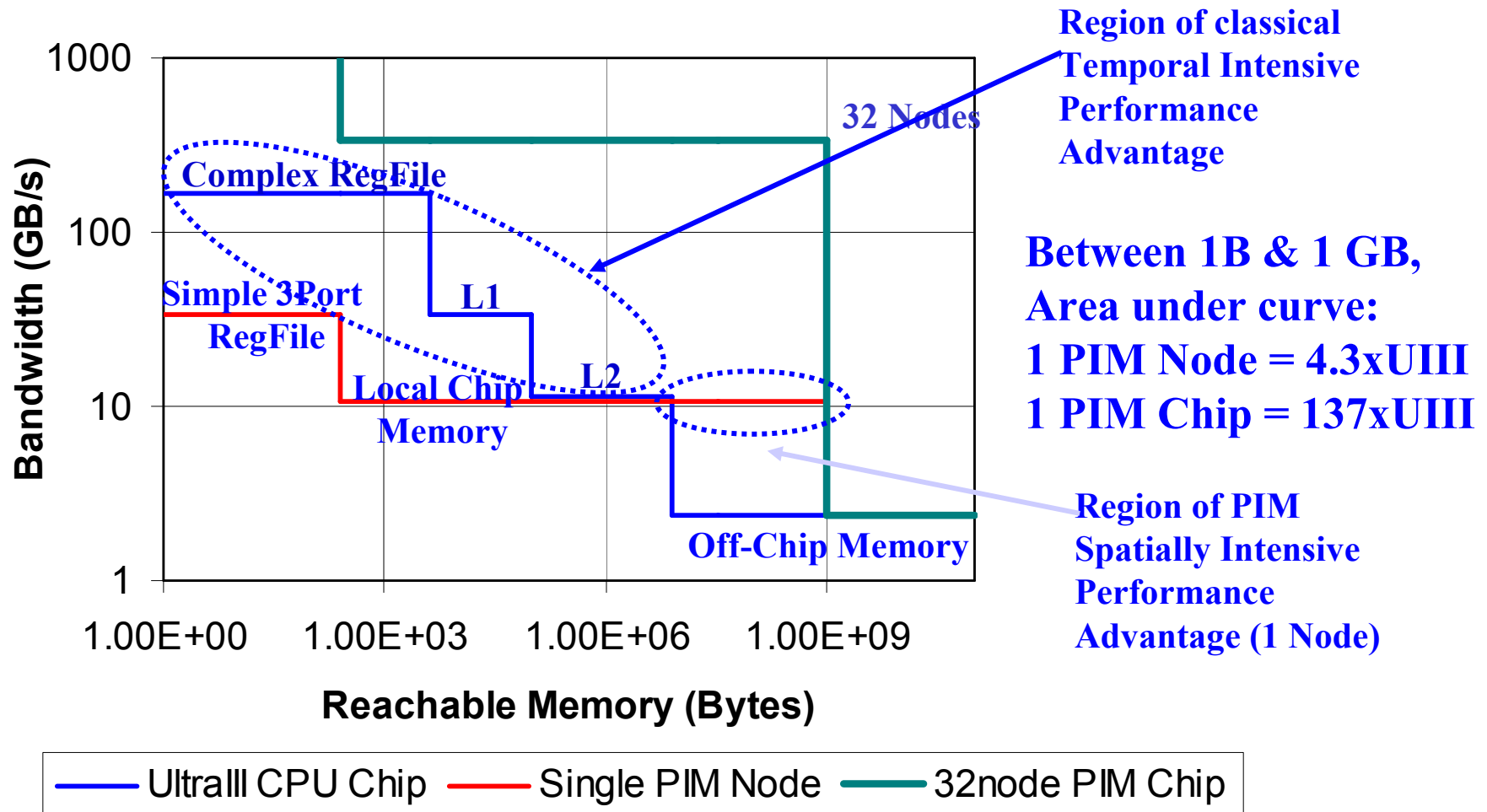


**Tiled & Scalable:
BLUE GENE,
EXECUBE**



The PIM

“Bandwidth Bump”





PIM-Based Architectures: System & Chip Level

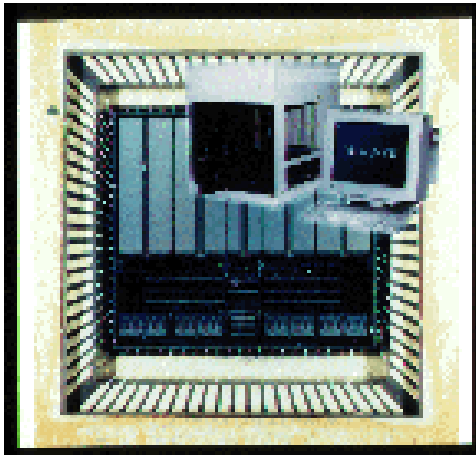
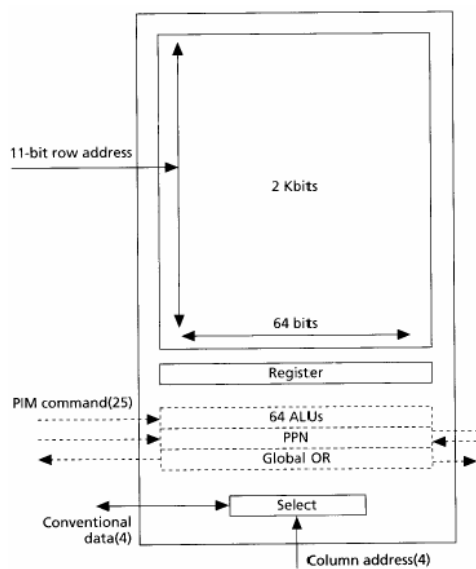


PIM System Design Space: Historical Evolution

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



TERASYS SIMD PIM (circa 1993)

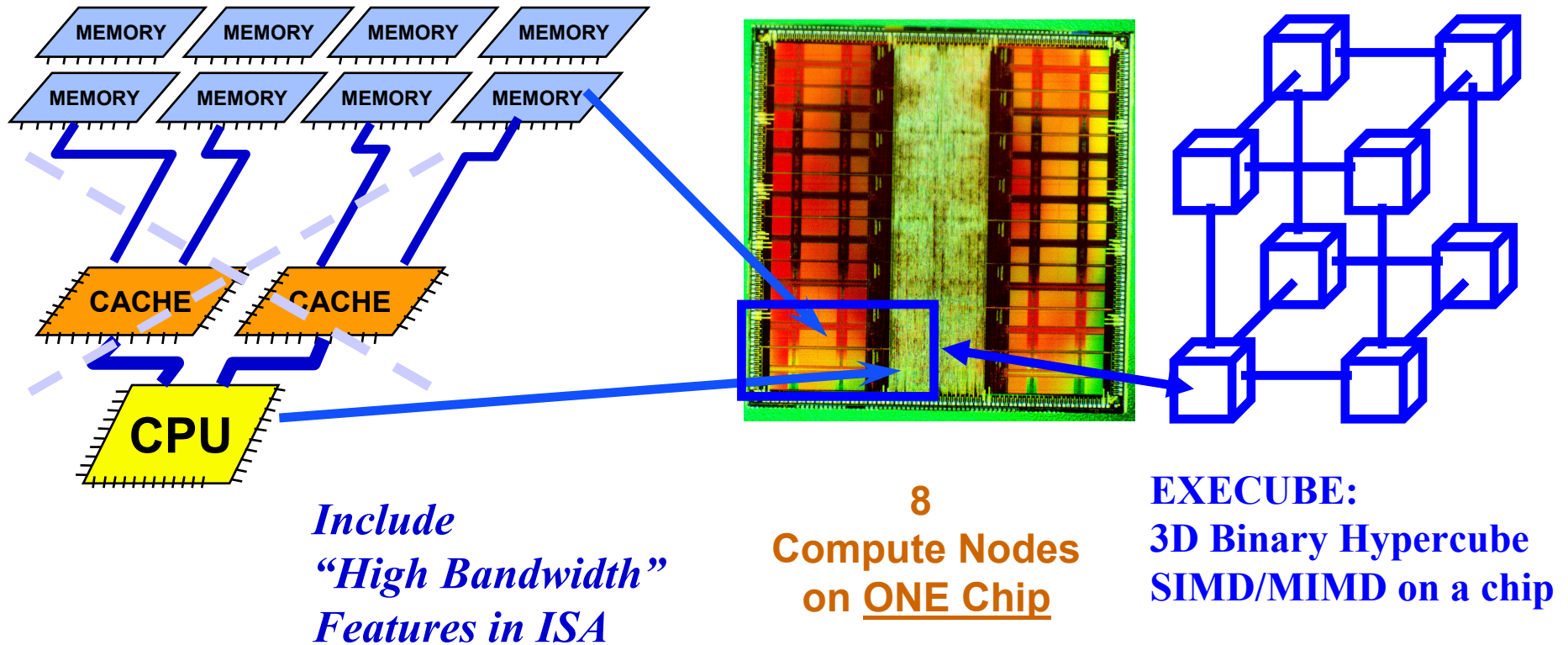


- Memory part for CRAY-3
- “Looked like” SRAM memory
 - With extra command port
- 128K SRAM bits (2k x 64)
- 64 1 bit ALUs
- SIMD ISA
- Fabled by National
- Also built into workstation with 64K processors
 - 5-48X Y-MP on 9 NSA benchmarks



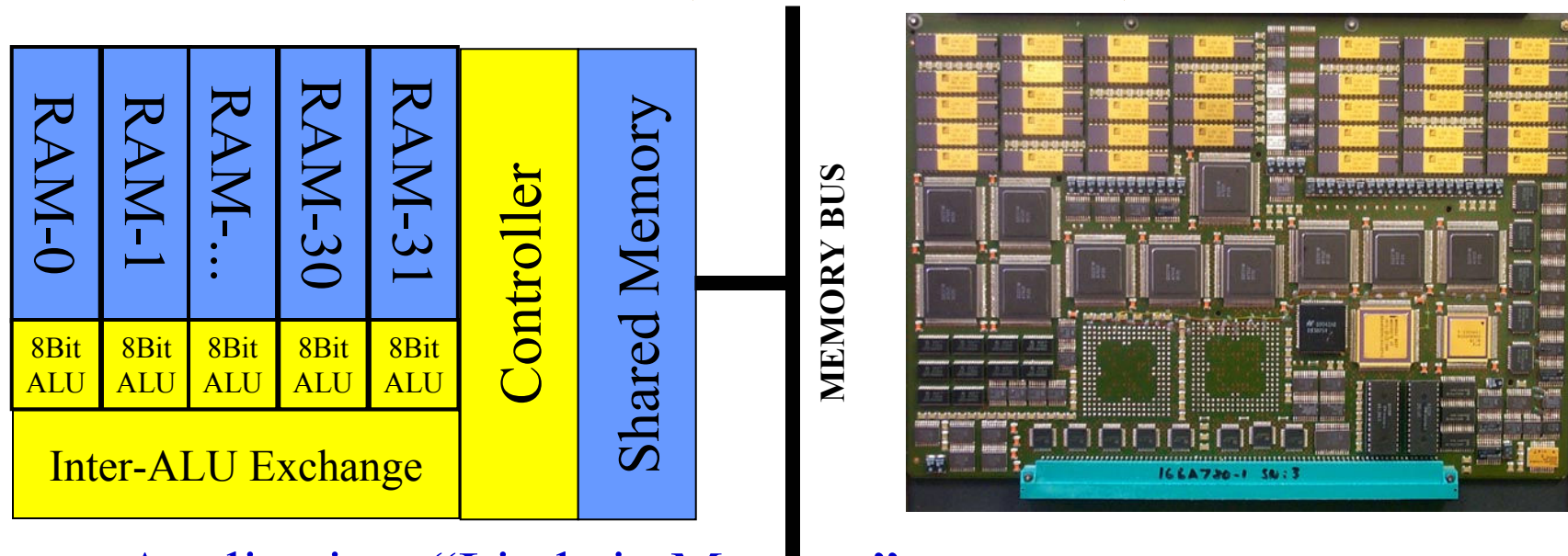
EXECUBE: An Early MIMD PIM (1st Silicon 1993)

- First DRAM-based Multiprocessor on a Chip
- *Designed from onset for “glueless” one-part-type scalability*
- On-chip bandwidth: 6.2 GB/s; Utilization modes > 4GB/s





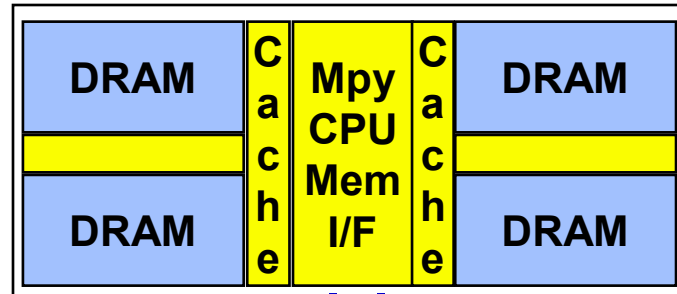
RTAIS: The First ASAP (circa 1993)



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

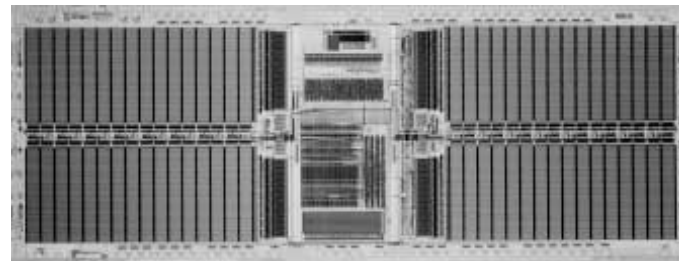


Mitsubishi M32R/D



Also two 1-bit I/Os

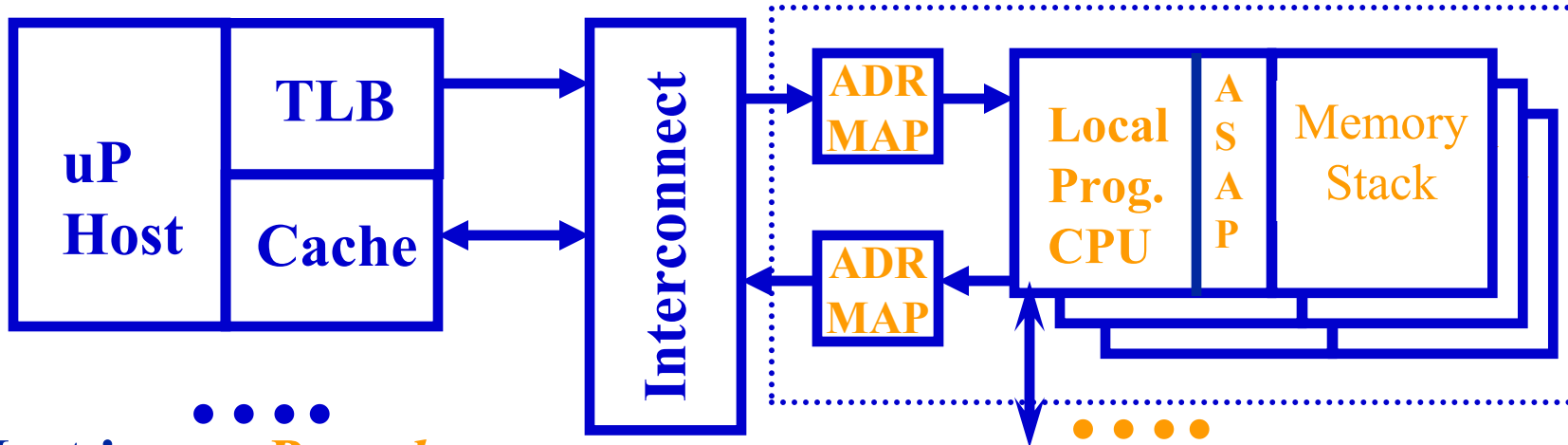
16 bit data bus | 24 bit address bus



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

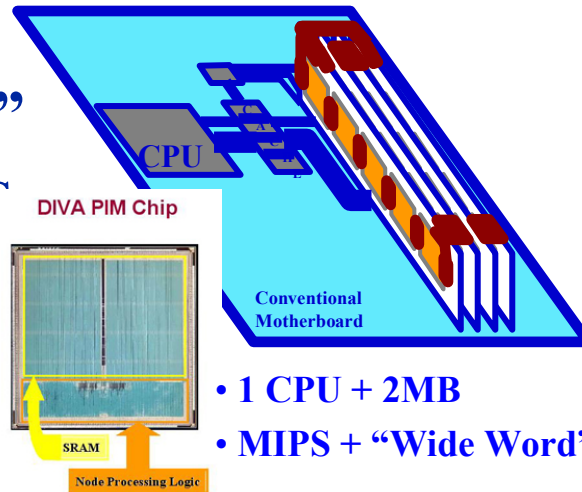


DIVA: Smart DIMMs for Irregular Data Structures



Host issues *Parcels*

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



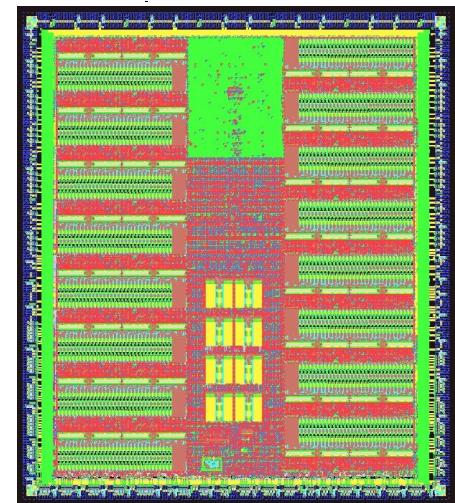
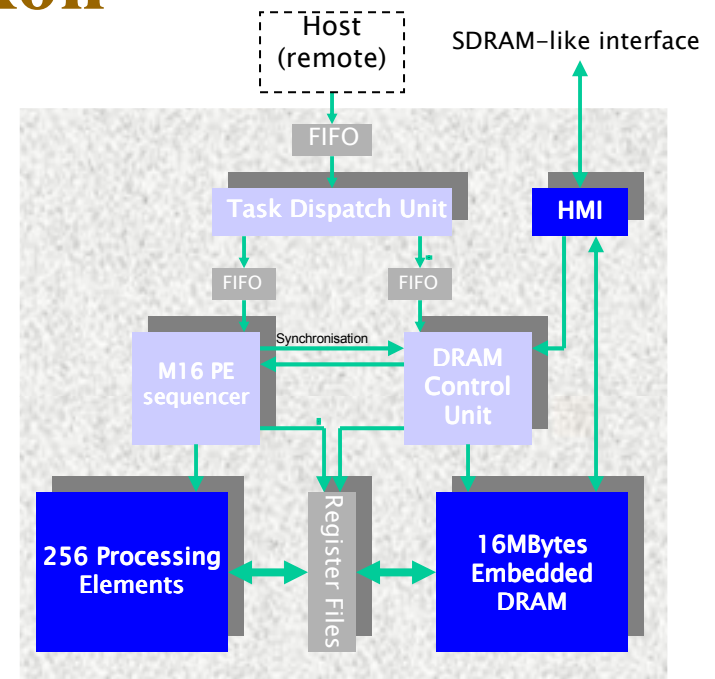
DIVA Functions:

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



Micron Yukon

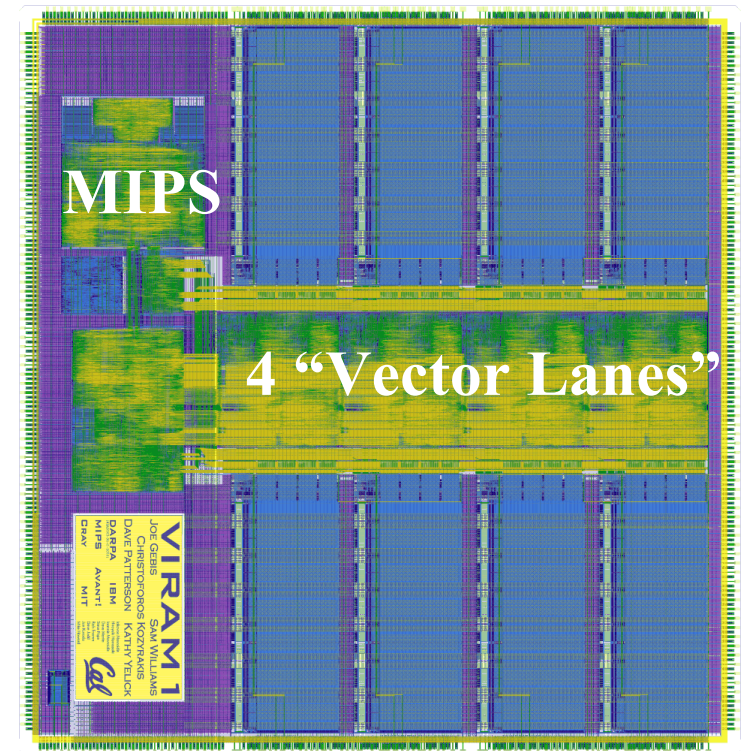
- **0.15 μ m eDRAM/ 0.18 μ m logic process**
- **128Mbits DRAM**
 - 2048 data bits per access
- **256 8-bit integer processors**
 - Configurable in multiple topologies
- **On-chip programmable controller**
- **Operates like an SDRAM**





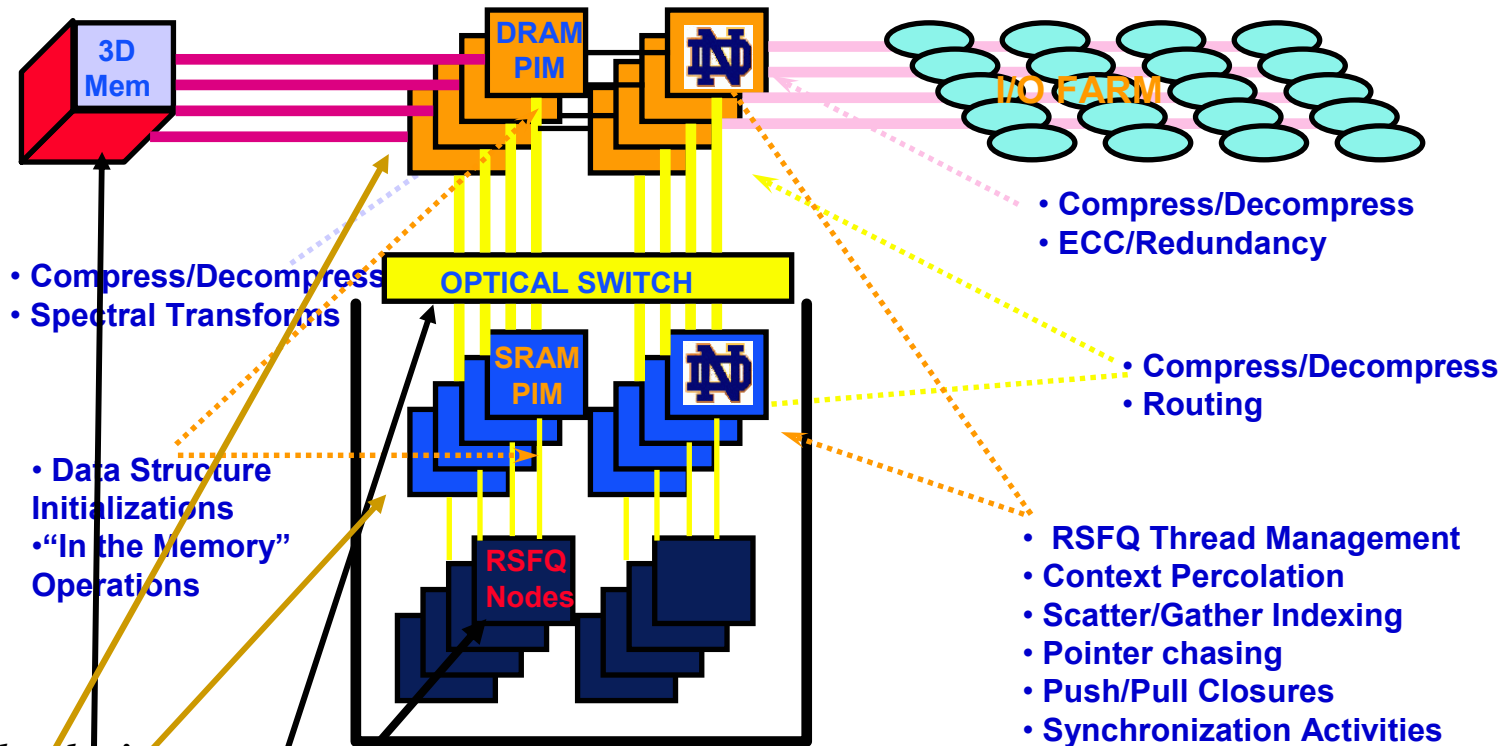
Berkeley VIRAM

- 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



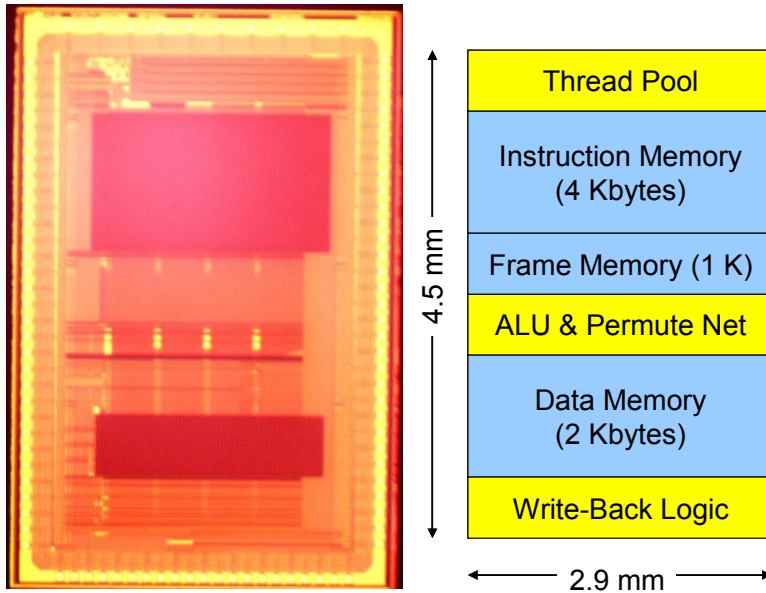
New Technologies:

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

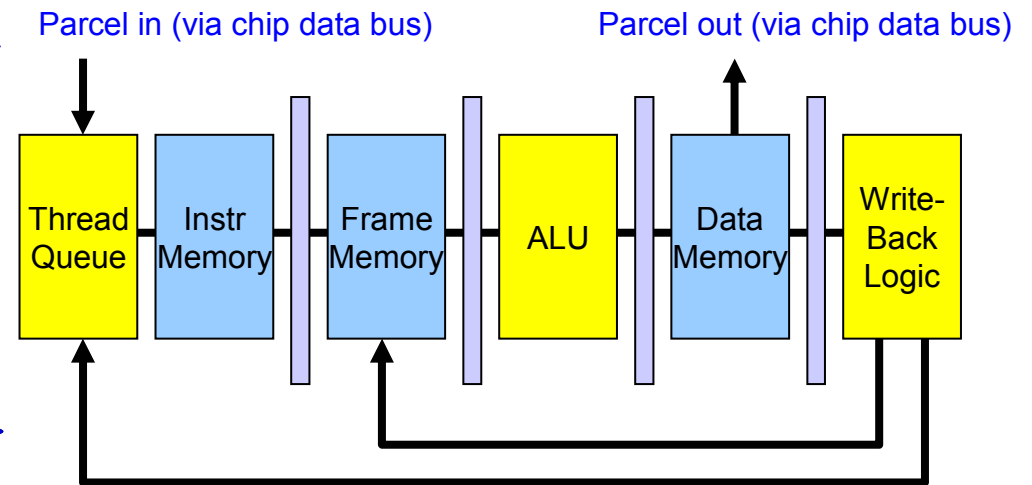
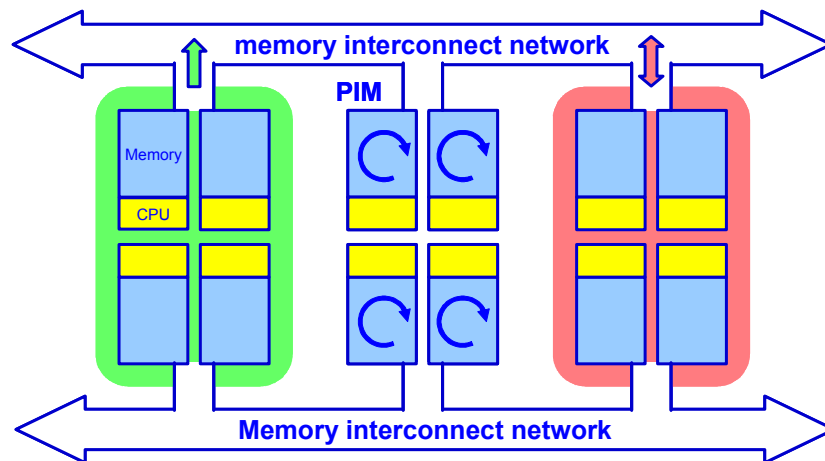
PIMs in Charge



PIM Lite

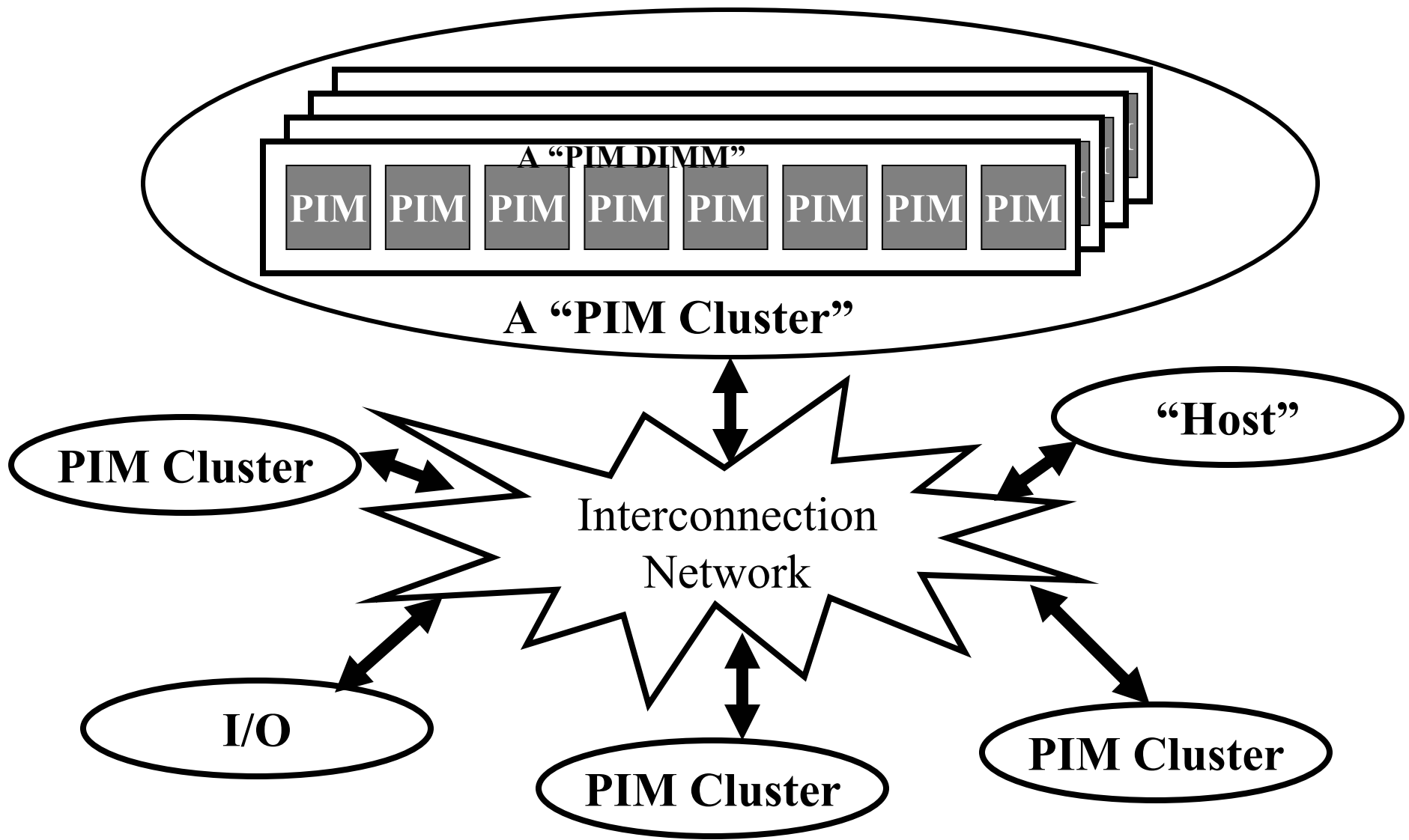


- “Looks like memory” at Interfaces
- ISA: 16-bit multithreaded/SIMD
 - “Thread” = IP/FP pair
 - “Registers” = wide words in frames
- Designed for multiple nodes per chip
- 1 node logic area ~ 10.3 KB SRAM (comparable to MIPS R3000)
- TSMC 0.18u 1-node 1st pass success
- 3.2 million transistors (4-node)



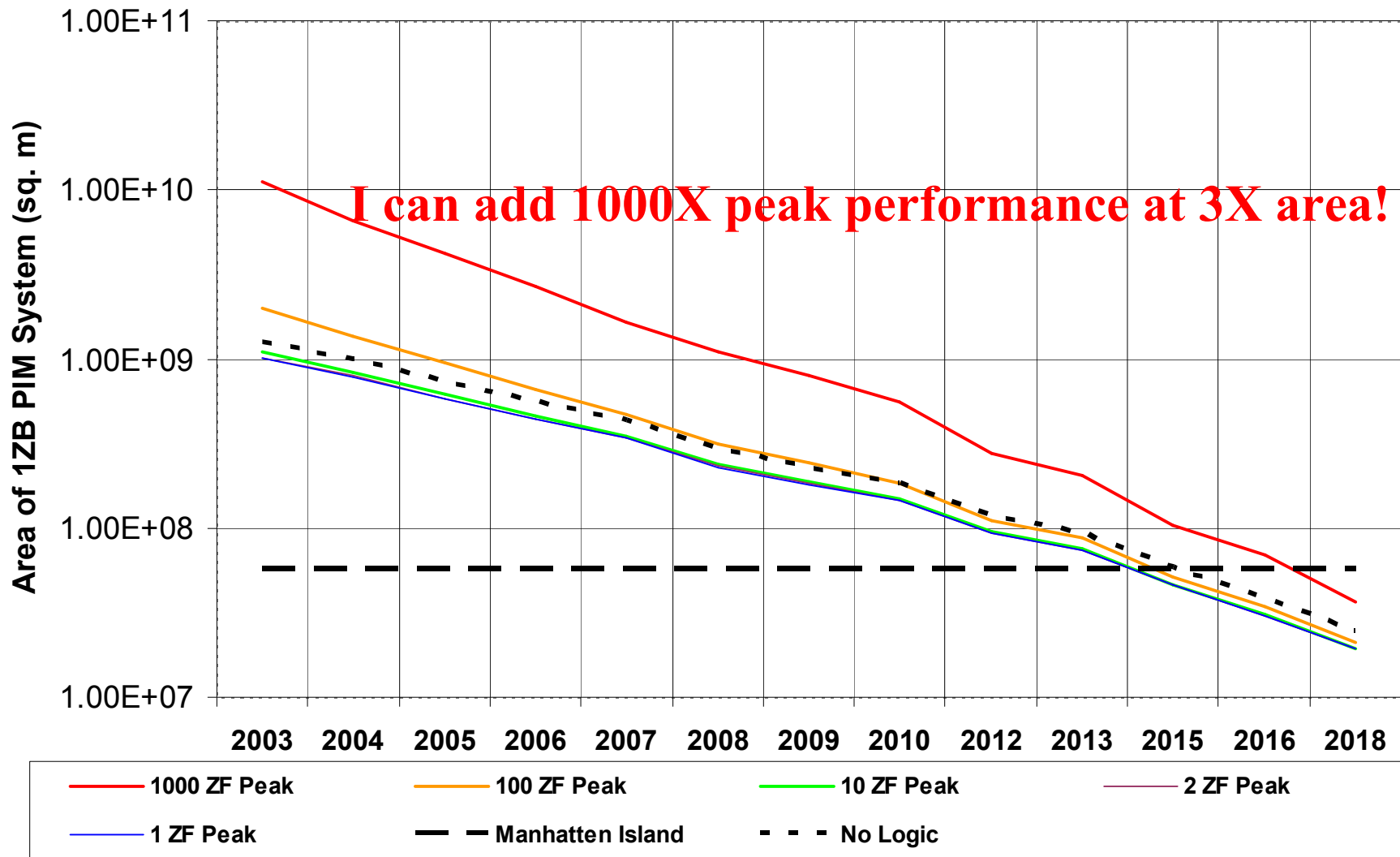


Next: An “All-PIM” Supercomputer





What Might an “All-PIM” Zetta Target Look Like?

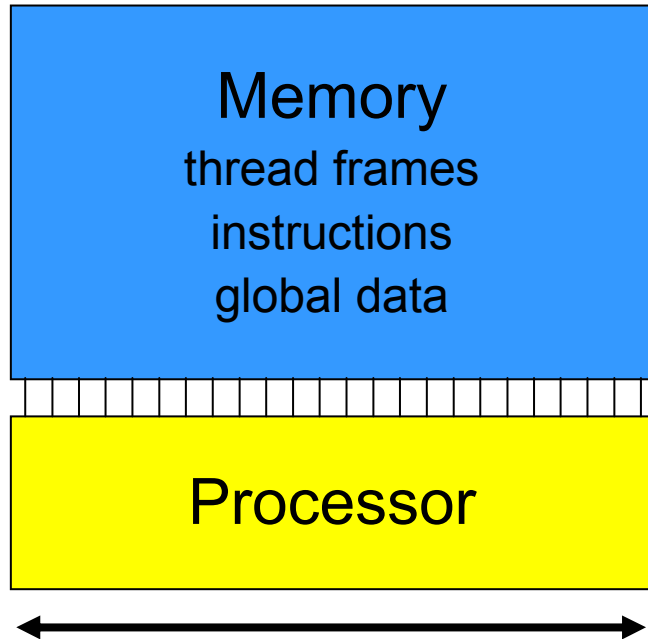




Matching ISA & Execution Models



Guiding Principles: Memory-Centric Processor Architecture



- ◆ Focus on memory, not logic
 - ◆ Make “CPUs” anonymous
- ◆ Maximize performance, keeping cost in check
 - ◆ make processor cost like a memory, not vice-versa!

- “How low can you go”
 - minimize storage for machine state
 - registers, buffers, etc.
 - don’t overprovision functional units

- ◆ “Wider is better”
 - swap thread state in single bus cycle
 - wide-word, SIMD data operations



Working Definitions

Light Weight State: Refers to computational thread

- Separated from rest of system state
- Reduced in size to ~ cache line

Serious Multi-threading

- Very cheap thread creation and state access
- Huge numbers of concurrent threads
- Support for cheap, low level synchronization
- Permit opportunities for significant latency hiding
- ISA level knowledge of “locality”



How to Use Light Weight Threads

- **Approaches to solving Little's Law problems**
 - Reduce # of latency-causing events
 - Reduce total number of bit/chip crossings per operation
 - Or, reduce effective latency
- **Solutions**
 - Replace 2 way latency by 1 way “command”
 - Let thread processing occur “at the memory”
 - Increase number of memory access points for more bandwidth
- **Light Weight Thread:** *minimal state needed to initiate limited processing in memory*



Parcels: The Architectural Glue

- **Parcel** = *Parallel Communication Element*
- Basic unit of communication between nodes
 - At same level as a “dumb memory reference”
- Contents: extension of “Active Message”
 - Destination: Object in application space
 - Method: function to perform on that object
 - Parameters: values to use in computation

Threads in Transit!!!



Type of Parcels

- *Memory Access: “Dumb”* Read/write to memory
- *AMO*: Simple_prefix_op to a memory location
- *Remote Thread Invocation*: Start new thread at node holding designated address
 - Simple case: booting the original node run-time
 - More interesting: “slices” of program code
- *Remote Object Method Invocation*: Invoke method against an object at object’s home
- *Traveling Thread = Continuation*: Move *entire* execution state to next datum.



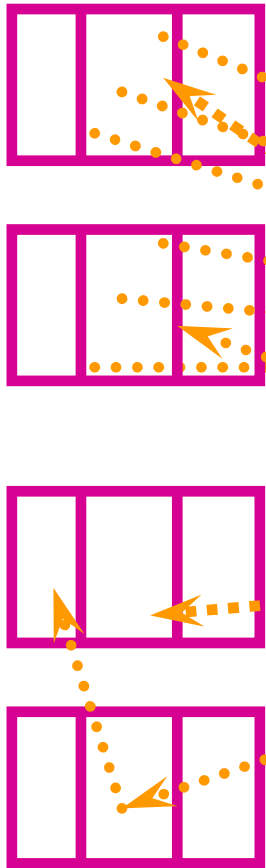
Software Design Space Evolution

- **Hardware “fetch_and_op”**
- **Generic libraries only**
- **App.-specific subprogram compilation**
 - **Explicit method invocation**
 - **Expanded storage management**
- **Explicit support for classical multi-threading**
- **Inter-thread communication**
 - **Message passing**
 - **Shared memory synchronization**
 - **Atomic multi-word transactions**
- **Pro-active data migration/percolation**
- **Expanded multi-threading**
 - **Extraction of very short threads, new AMOs**
 - **Thread migration**
- **OS, run-time, I/O management “in the memory”**

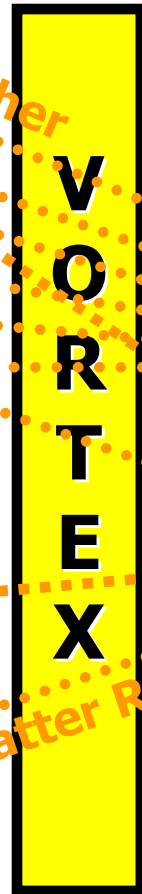


The HTMT Percolation Execution Model

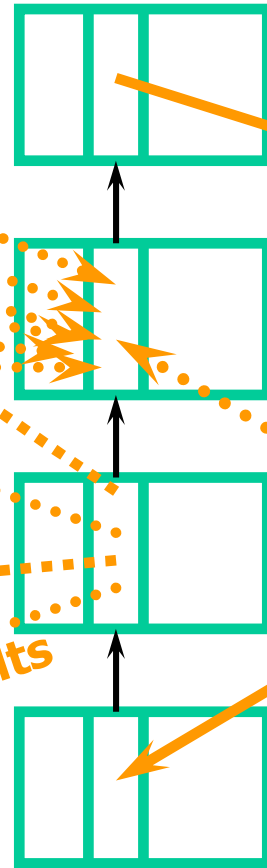
Data Structures
In DRAM



Gather
Data

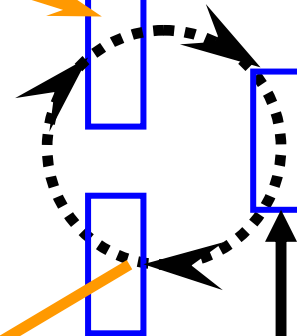


"Contexts" in SRAM



Working Data

"Contexts" in
CRAM



Scatter Results

Results



All Orange Arrows are parcels



More Exotic Parcel Functions

The “Background” Ones

- **Garbage Collection:** reclaim memory
- **Load Balancing:** check for over/under load & suggest migration of activities
- **Introspection:** look for livelock, deadlock, faults or pending failures
- **Key attributes of all of these**
 - Independent of application programs
 - Inherent memory orientation
 - Significant possibility of mobility



Examples:

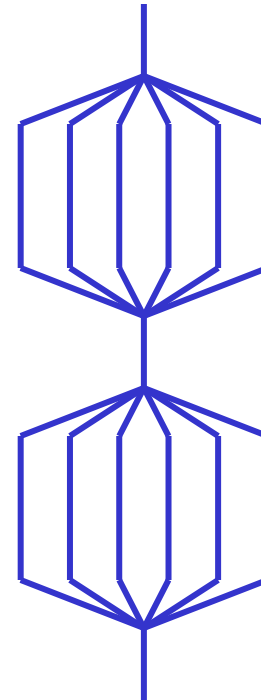
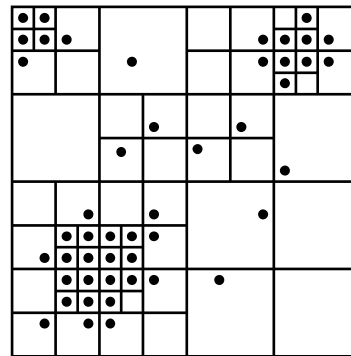
- **In-Memory Multi-Threading**
- **Traveling Threads**
- **Very Light Weight Thread Extraction**



N-Body Simulation

- Simulate motion of N bodies under mutual attractive/repulsive force. $O(N^2)$
- Barnes-Hut method
 - clusters of bodies approximated by single body when dense and far away
 - subdivide region into cells, represent using quad/octree
- Highly parallel
 - 90+ percent of workload can be parallelized

$$F_i = \sum_j \frac{Gm_i m_j}{\|\mathbf{x}_i - \mathbf{x}_j\|^2}$$

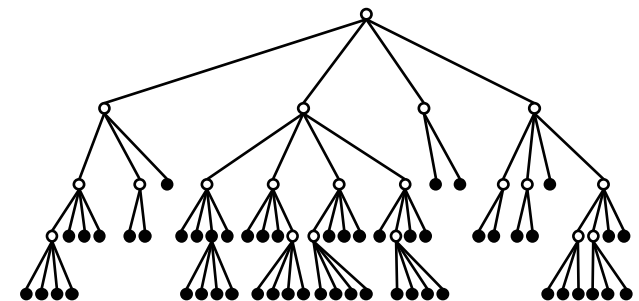


initialize

for each body
calculate net force

synchronize
update positions

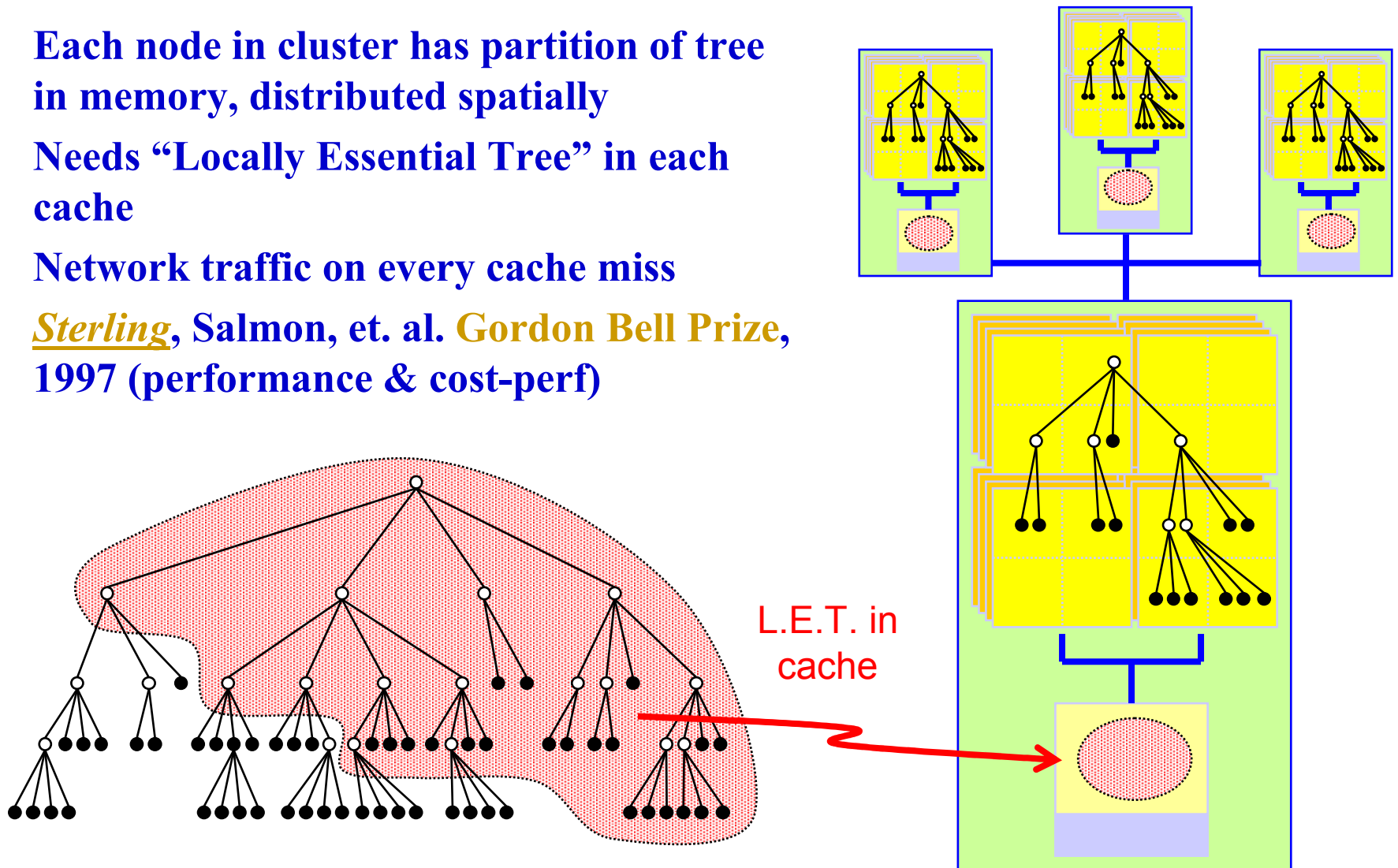
for each body
calculate net force





Heavyweight N-Body

- Each node in cluster has partition of tree in memory, distributed spatially
- Needs “Locally Essential Tree” in each cache
- Network traffic on every cache miss
- *Sterling*, Salmon, et. al. **Gordon Bell Prize, 1997** (performance & cost-perf)

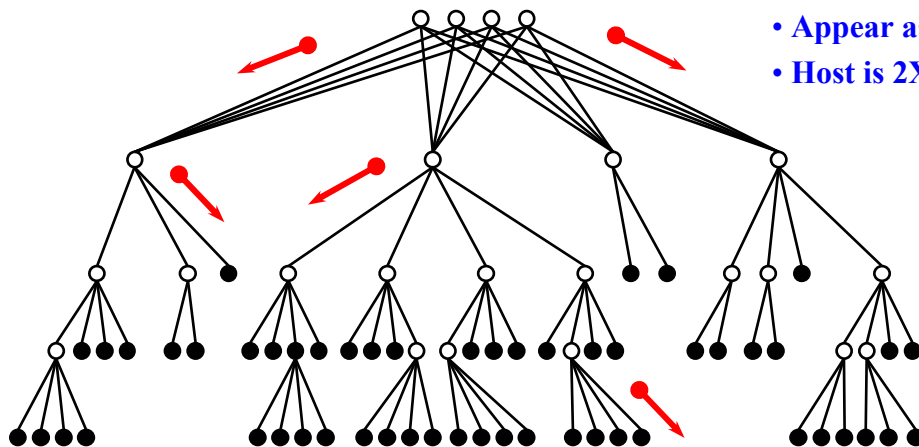




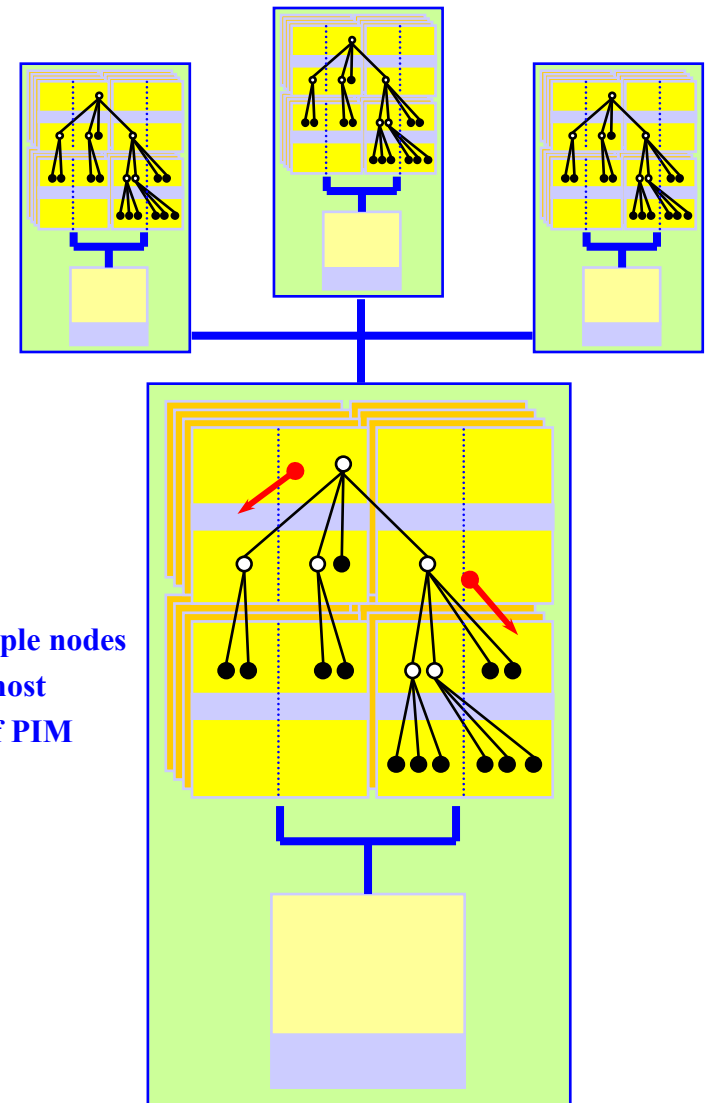
Multithreaded In-PIM N-Body

- Thread per body, accumulating net force, traversing tree in parallel, in-memory processing
- Very low individual thread state (net force, next pointer, body coordinates, body mass)
- Network traffic only when thread leaves partition—lower traffic overall

replicate top of tree to reduce bottleneck

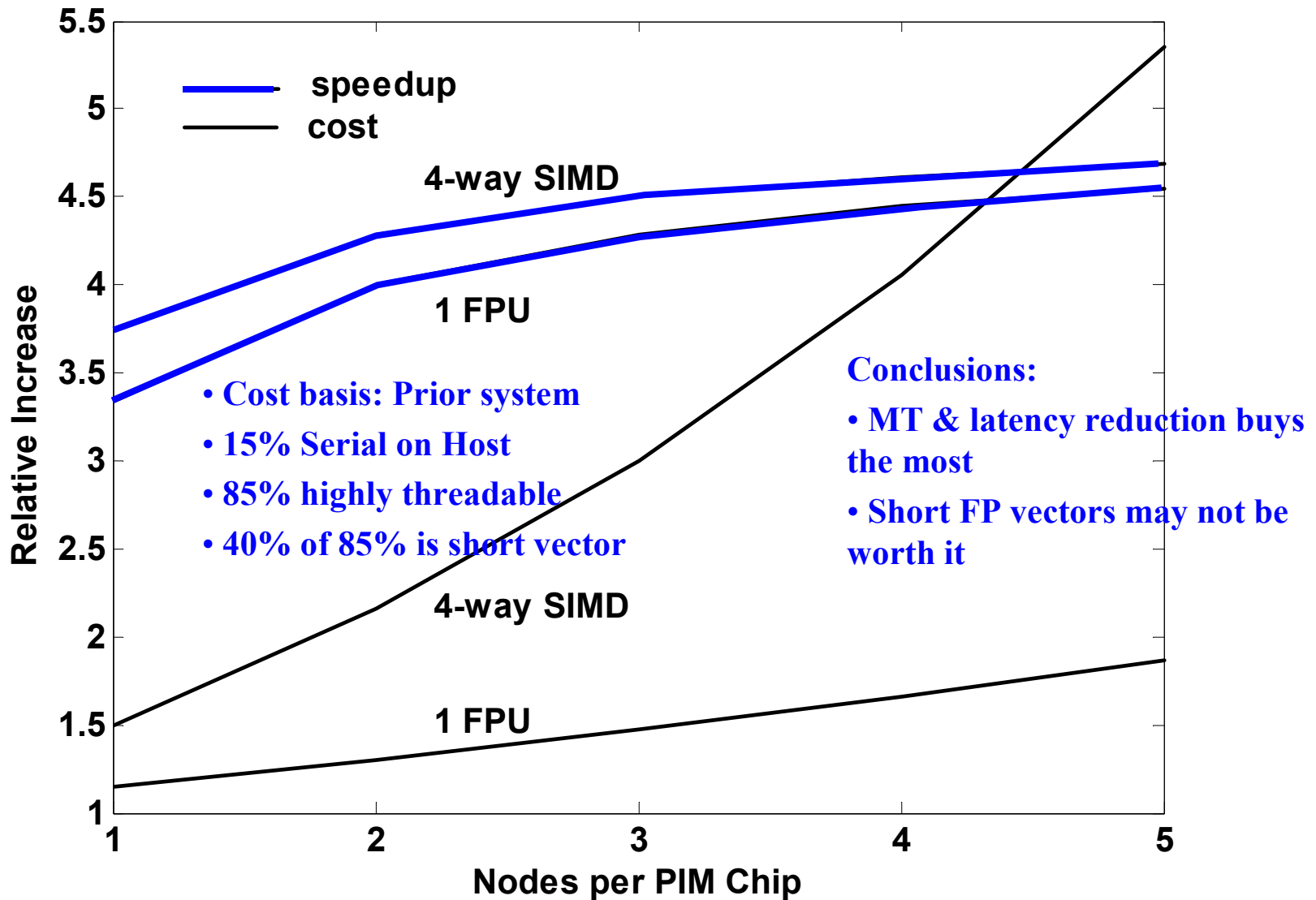


- 16 64 MB PIM Chips
 - Each with multiple nodes
- Appear as memory to host
- Host is 2X clock rate of PIM





N-Body on PIM: Cost vs. Performance





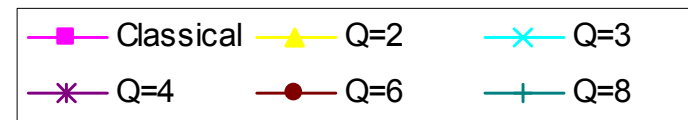
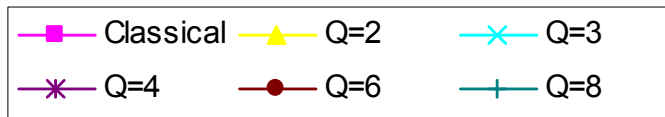
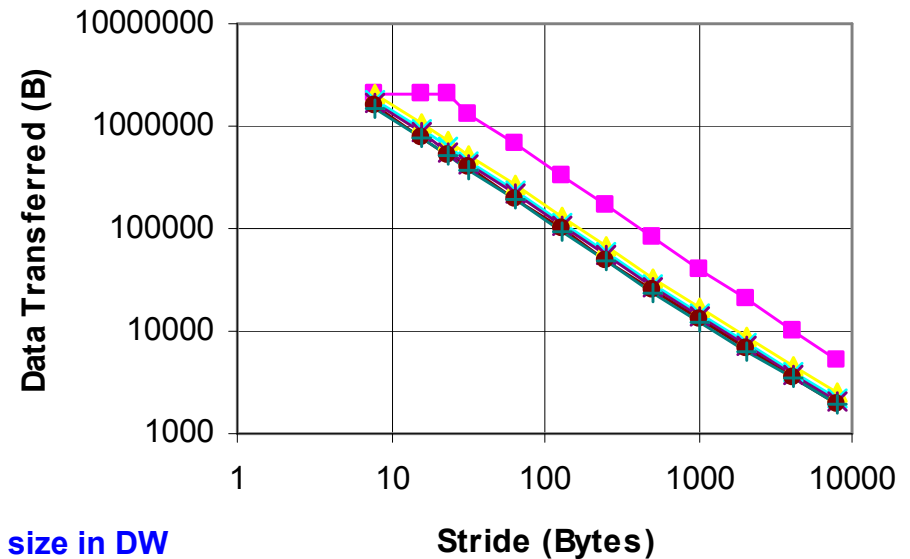
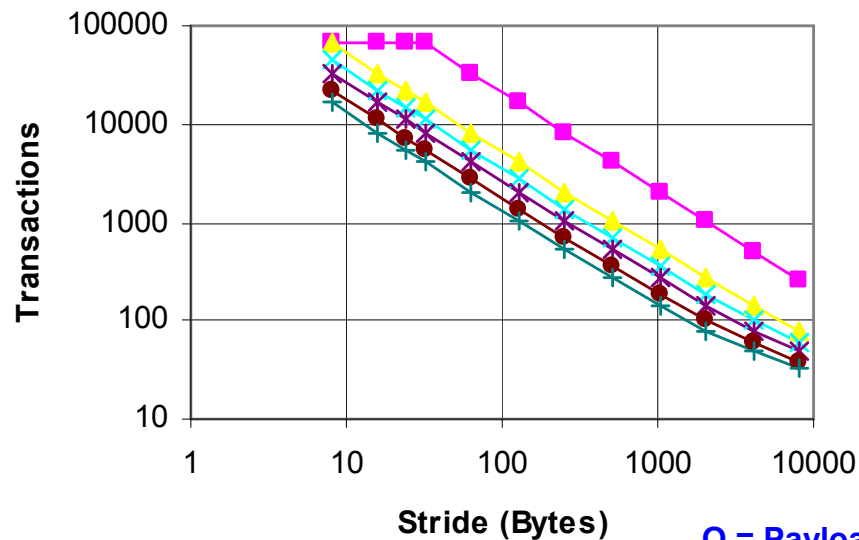
Example:

Traveling Thread Vector Gather

- **Given Base, Stride, Count, read strided vector to compact vector**
- **Classical CPU-centric approach:**
 - Issue “waves” of multiple, ideally K , loads
 - If stride $<$ block size, return cache line
 - Else return single double word
- **UWT thread-based**
 - Source issues K “gathering” threads, one to each PIM memory macro
 - Each thread reads local values into payload
 - Continuing dispatching payload when full



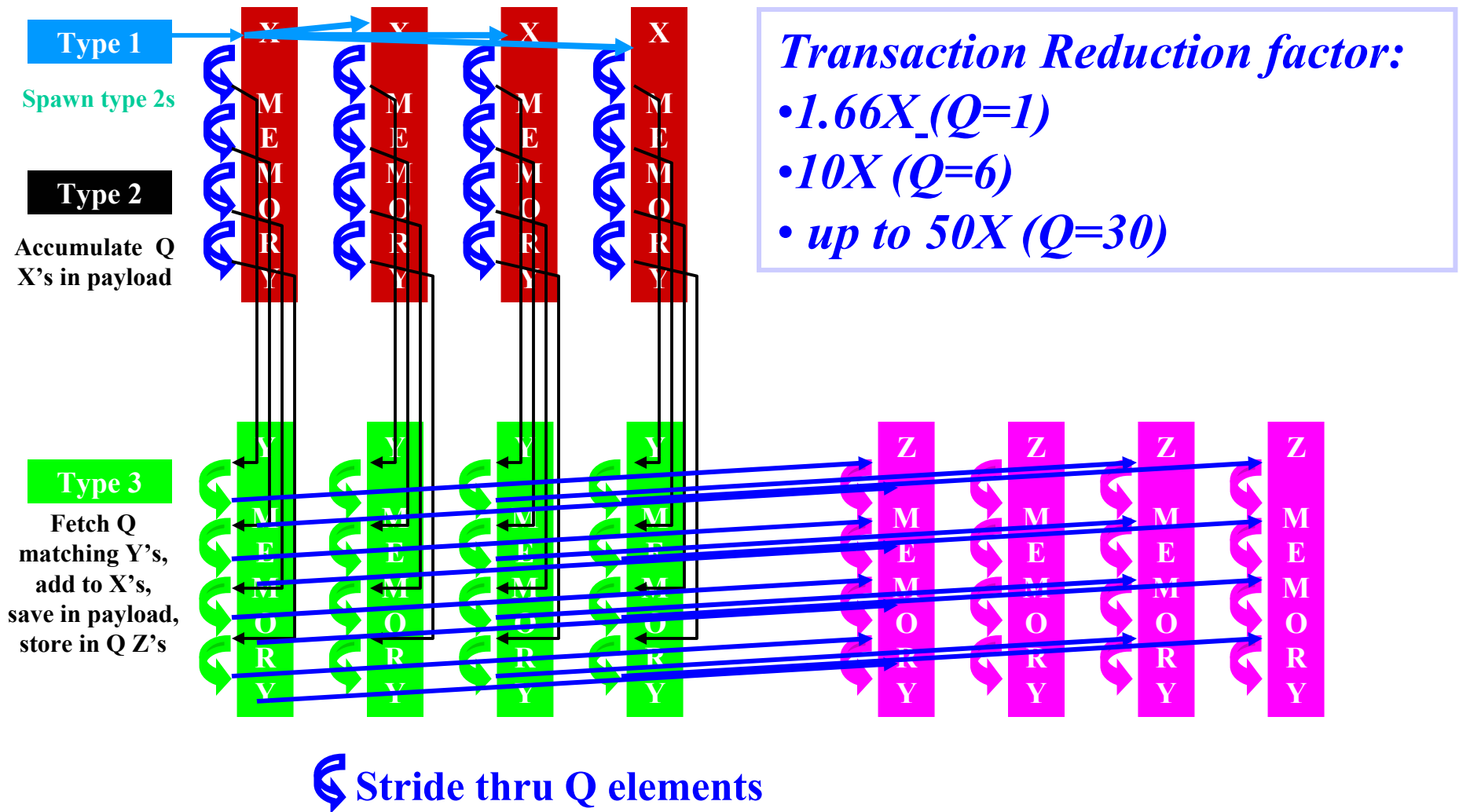
Vector Gather via LWT Traffic Estimates



- 4X+ reduction in Transactions
- 25%+ reduction in bytes transferred



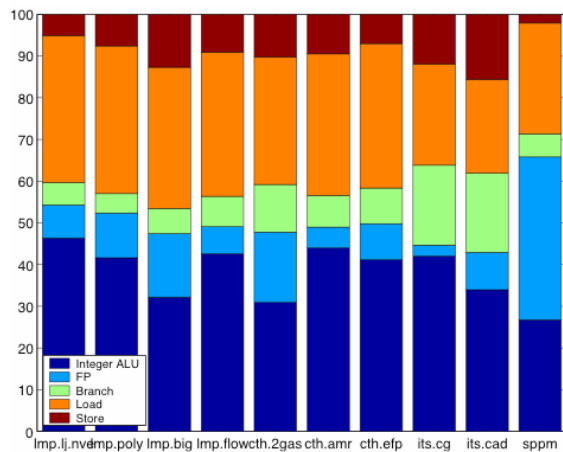
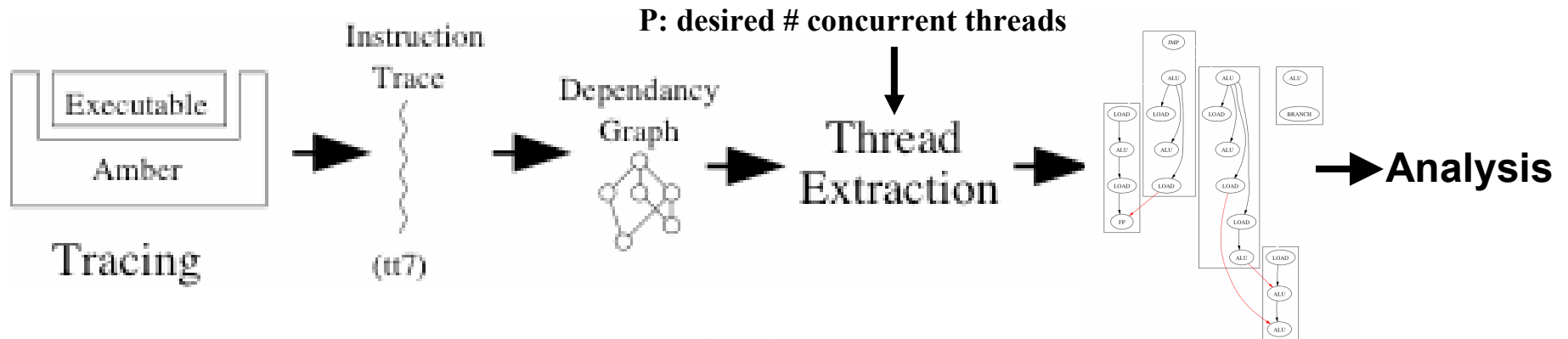
Vector Add via Traveling Threads



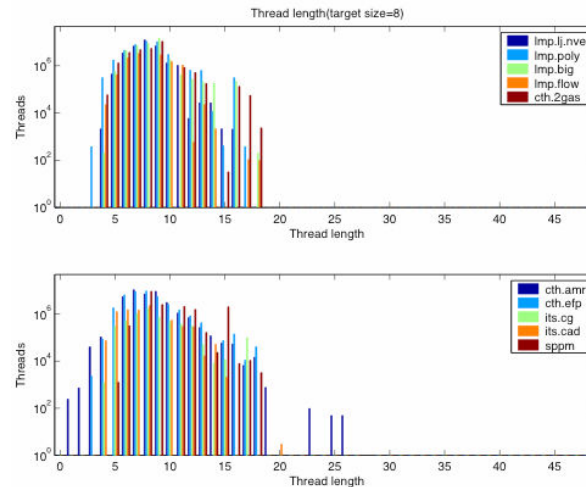


Trace-Based Thread Extraction & Simulation

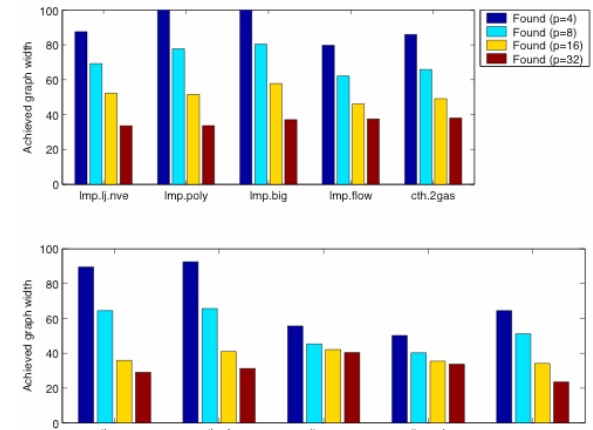
- Applied to large-scale Sandia applications over summer 2003



From Basic Application Data



Through Detailed Thread Characteristics



To Overall Concurrency



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
- **With today's architecture, we cannot support bandwidth between processors & memory**
- **PIM (Close logic & memory) attacks all these problems**
- **But it's still not enough for Zetta**
- **But the ideas may migrate!**



**A Plea to Architects
and Language/Compiler
Developers!**

Relentlessly Attack State Bloat

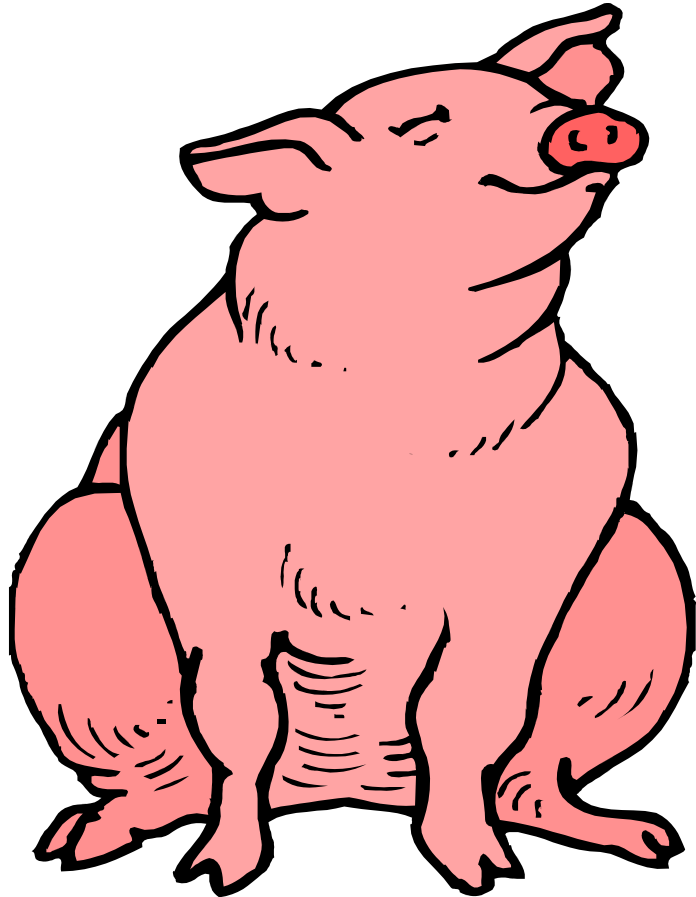
by

Reconsidering Underlying Execution Model

**Starting with Multi-threading
Of Mobile, Light Weight States
As Enabled by PIM technology**



The Future



Will We Design Like This?



Or This?

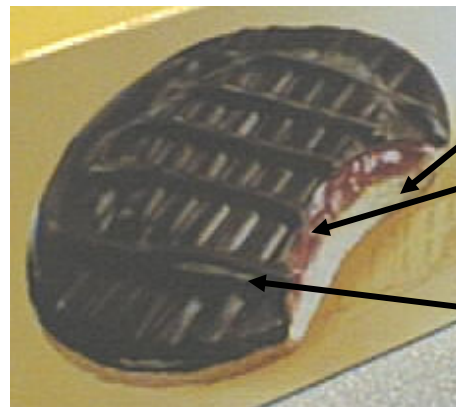
Regardless of Technology!



PIMs Now In Mass Production



- 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