Why Design Must Change: Rethinking Digital Design

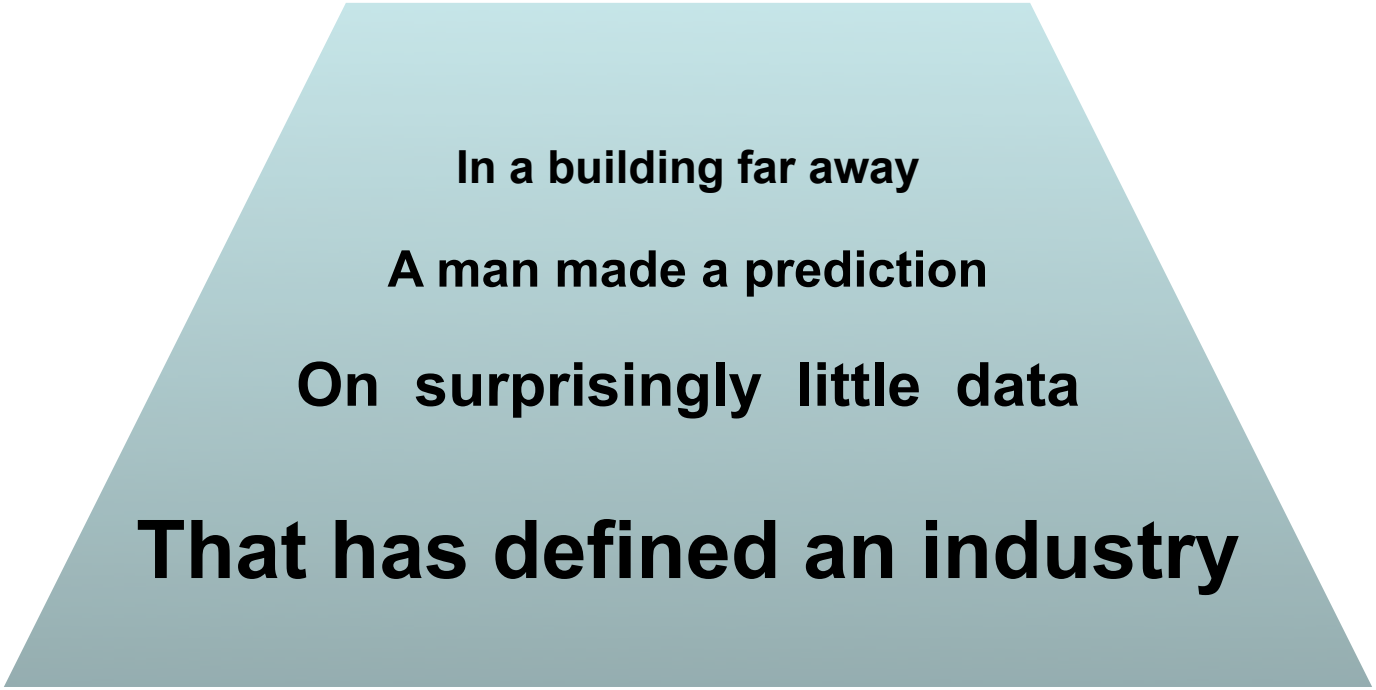
Mark Horowitz

Rambus, Inc.

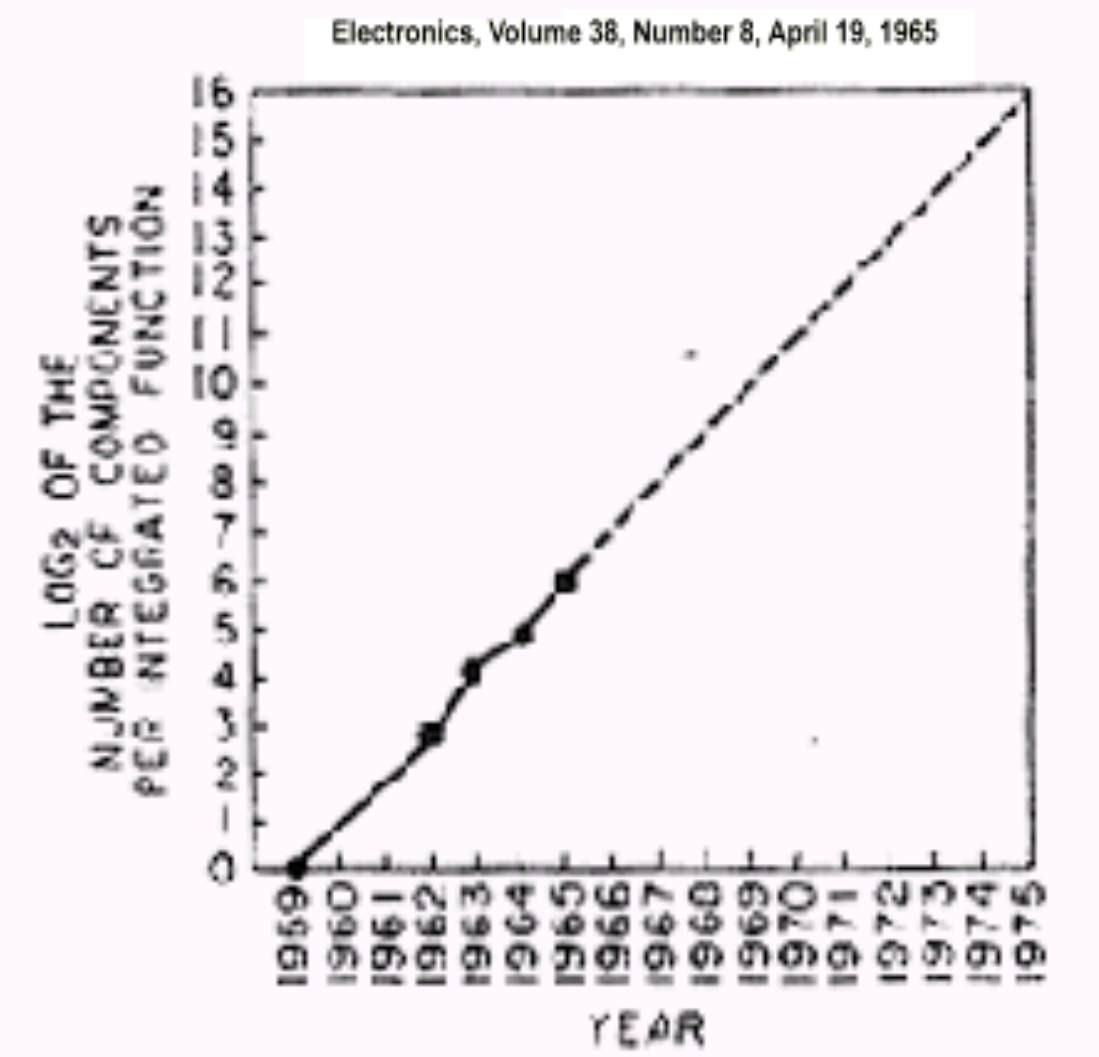
Stanford University

horowitz@stanford.edu

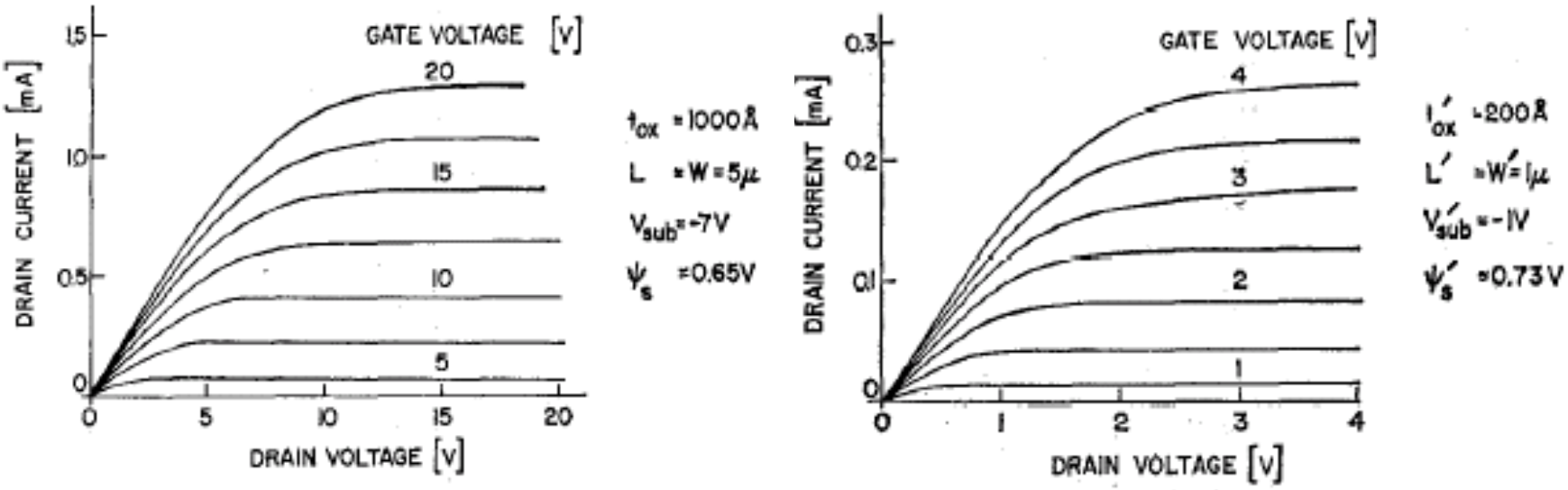
A Long Time Ago



Moore's Law


Electronics, Volume 38, Number 8, April 19, 1965
LOG2 OF THE
NUMBER OF COMPONENTS
PER INTEGRATED FUNCTION
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
YEAR

Dennard's MOS Scaling (1974)



JSSC Oct 74, pg 256

In this ideal scaling

- V scales to αV, L scales to αL
- So C scales to αC,
- E = V/L is constant, so i scales to αi (i/μ is stable)

The Triple Play

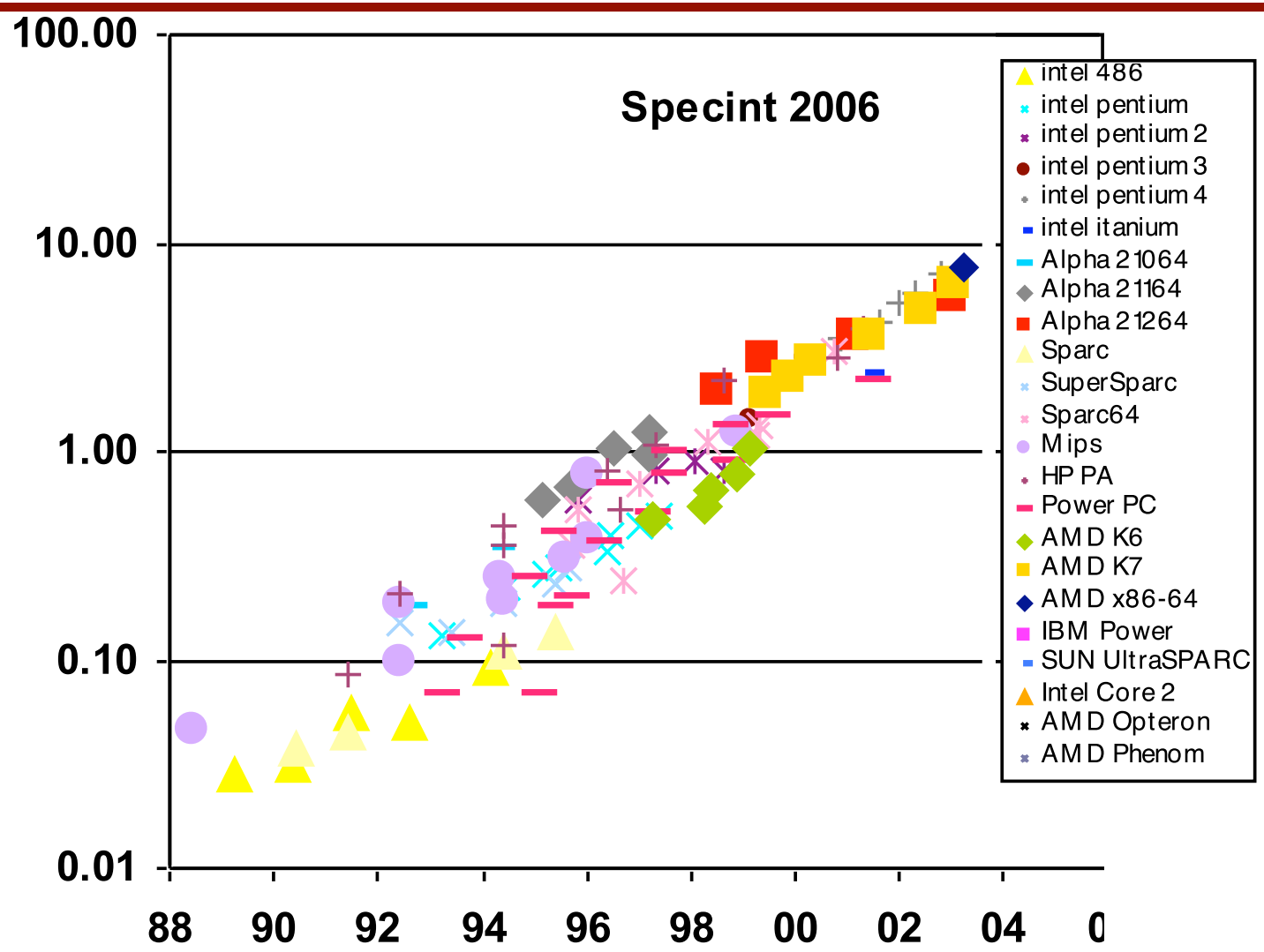
Using Dennard scaling rules

- Get more transistors, gates, $1/\alpha^2$
- Gates get faster, delay scales as α
- Energy per switch is reduced α^3

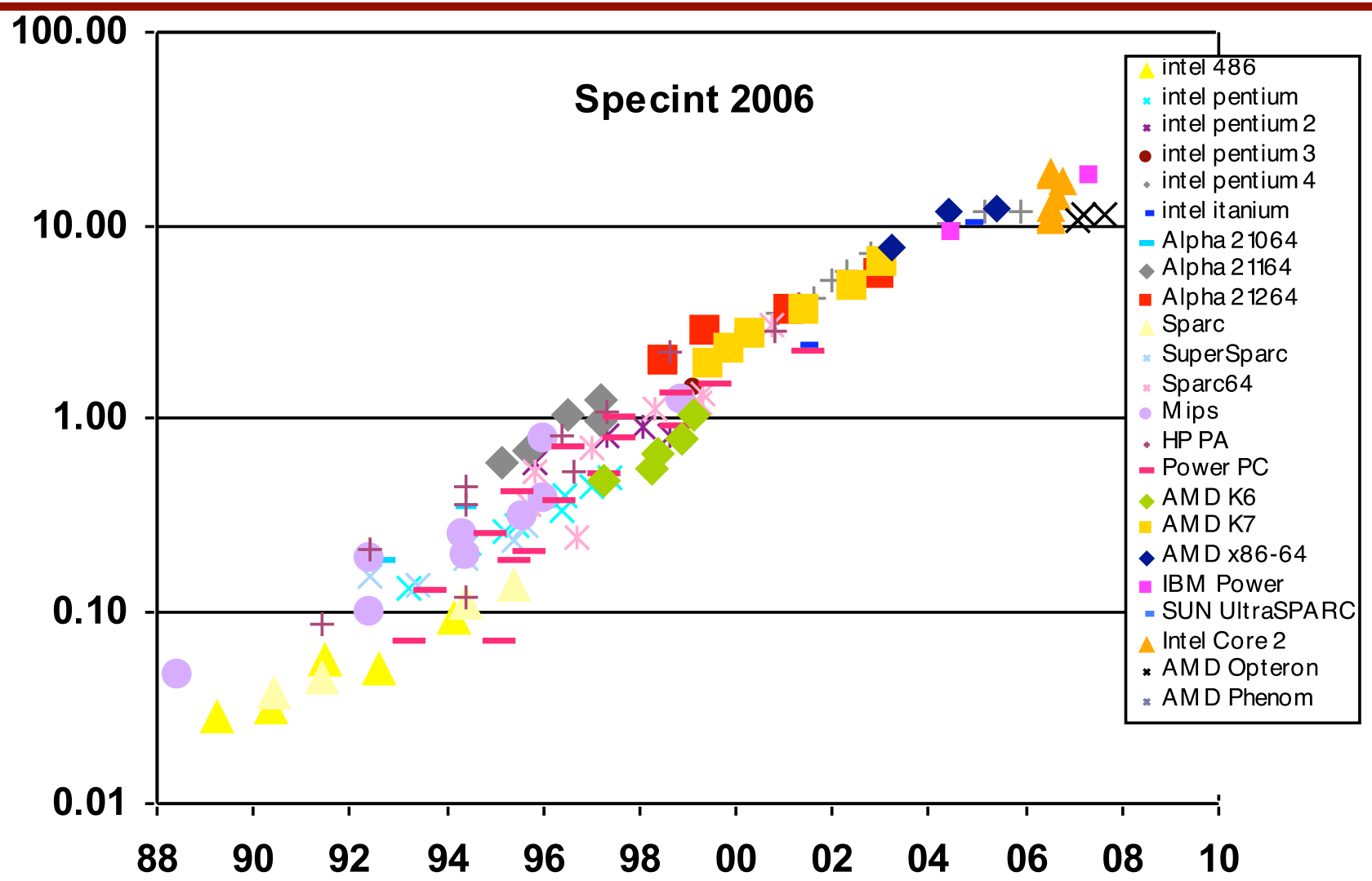
So we can compute $1/\alpha^3$ as many gate evals/sec

- At the same power and area as the previous design
- Architects take this to improve computer performance

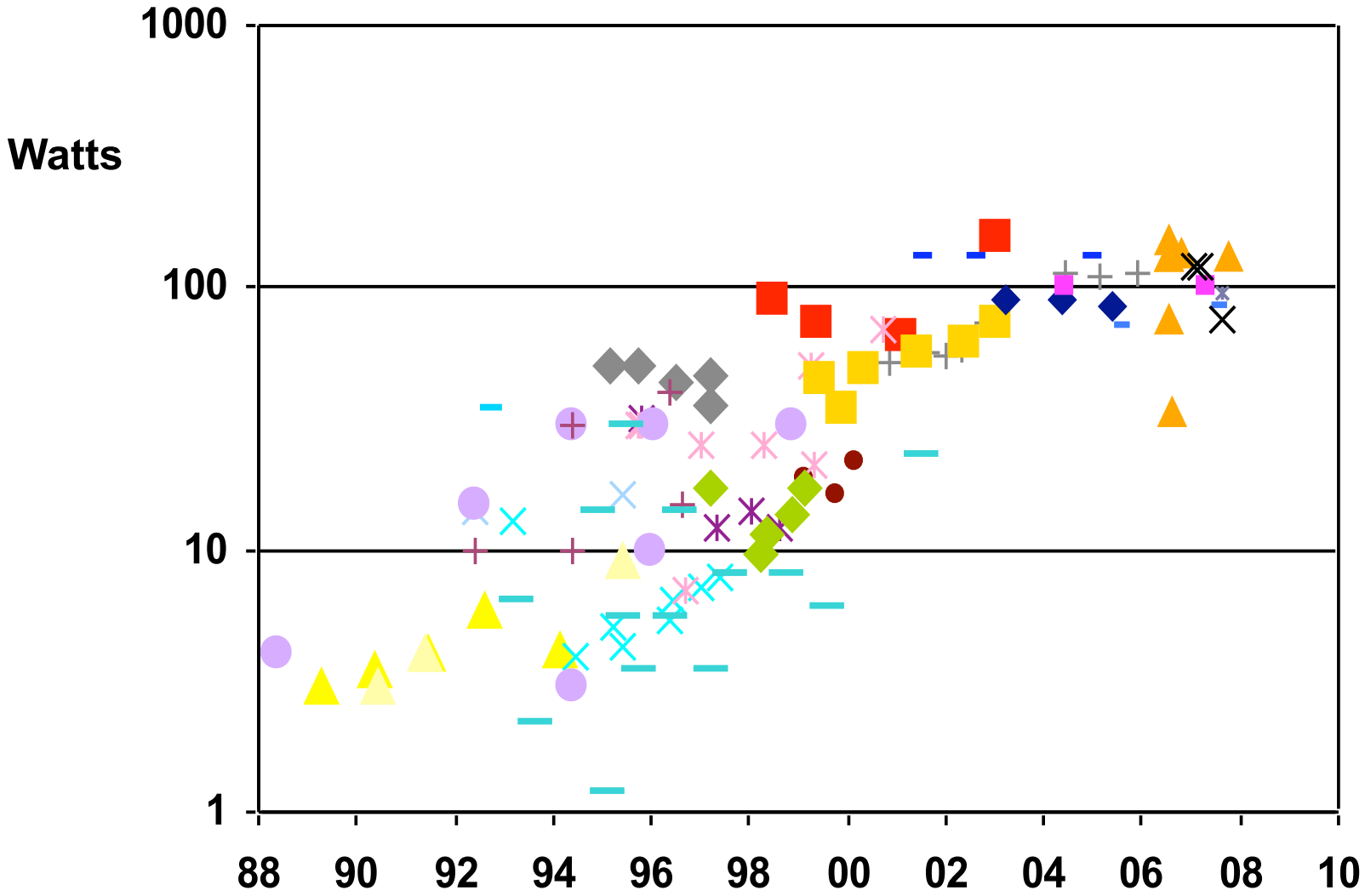
Computer Performance



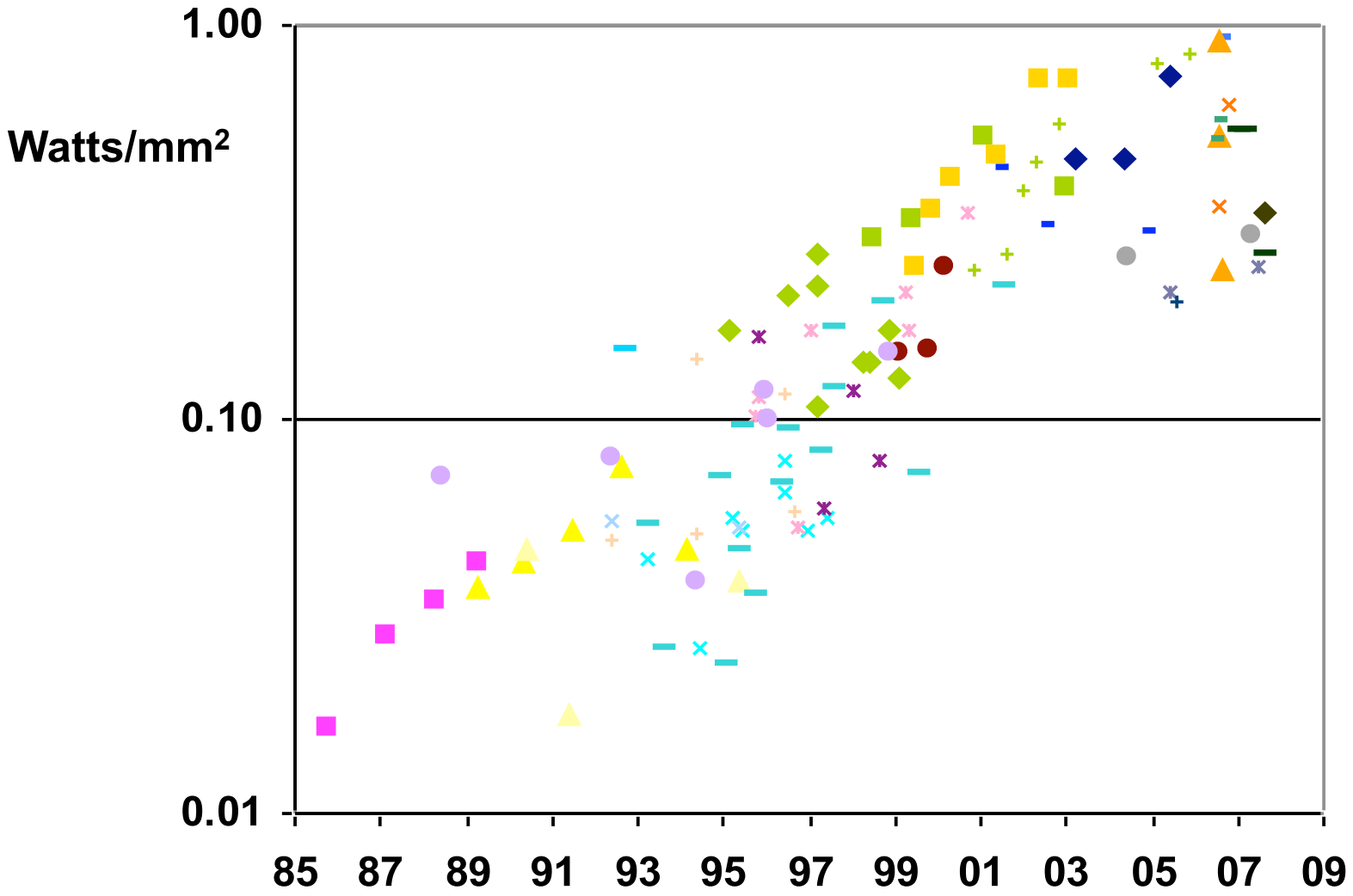
Or At Least We Used To



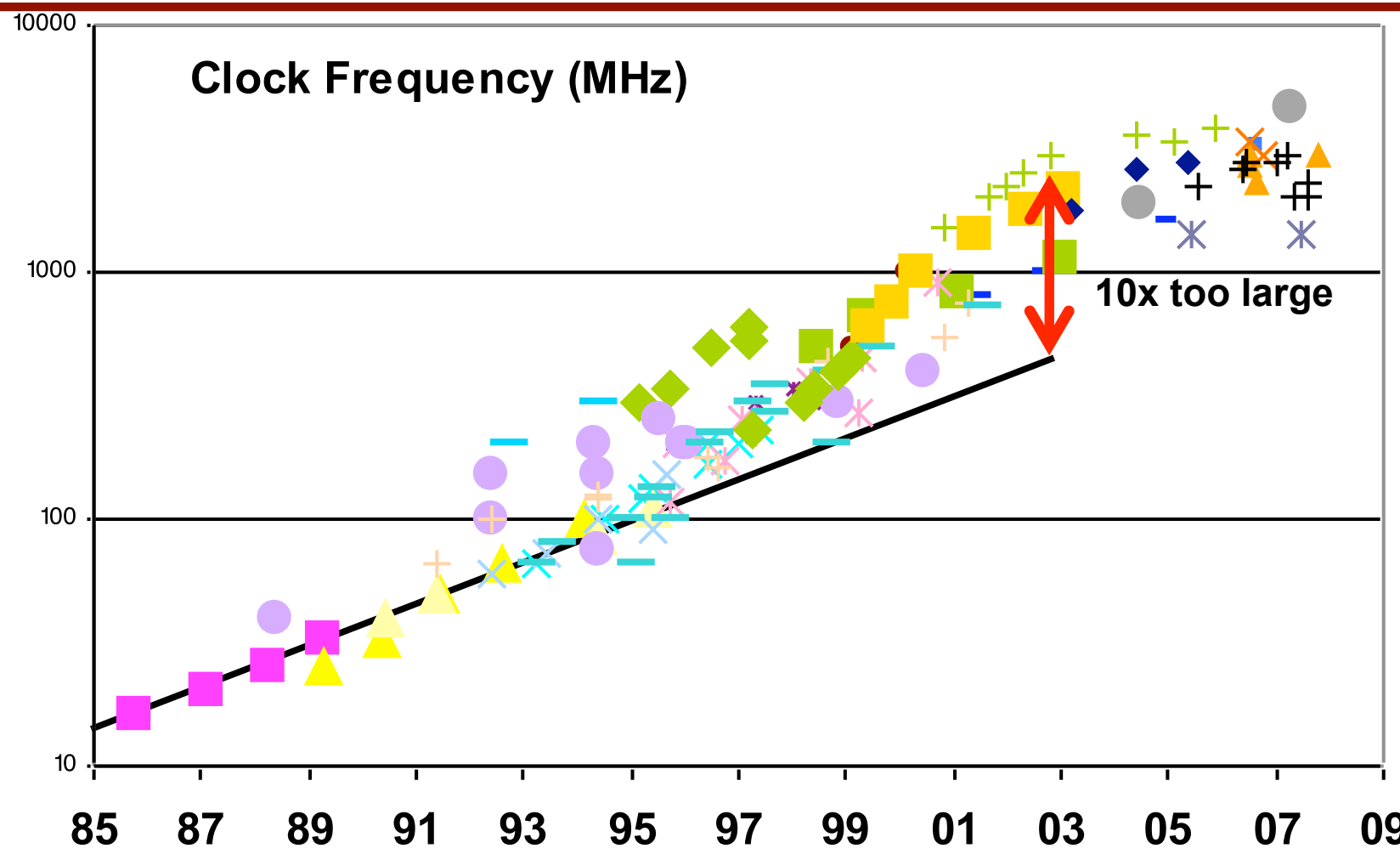
The Power Limit



Power Density Increased Too



Power Increased Because We Were ~~Greedy~~ Clever



So We Should Be Ok Now, Right?

Freq is scaling slowly

Die size is not growing

We have seen the light – parallelism

- No longer trying to make uniprocessors faster

Wrong Again – Dennard Scaling is Dead

Vth scaling has stopped

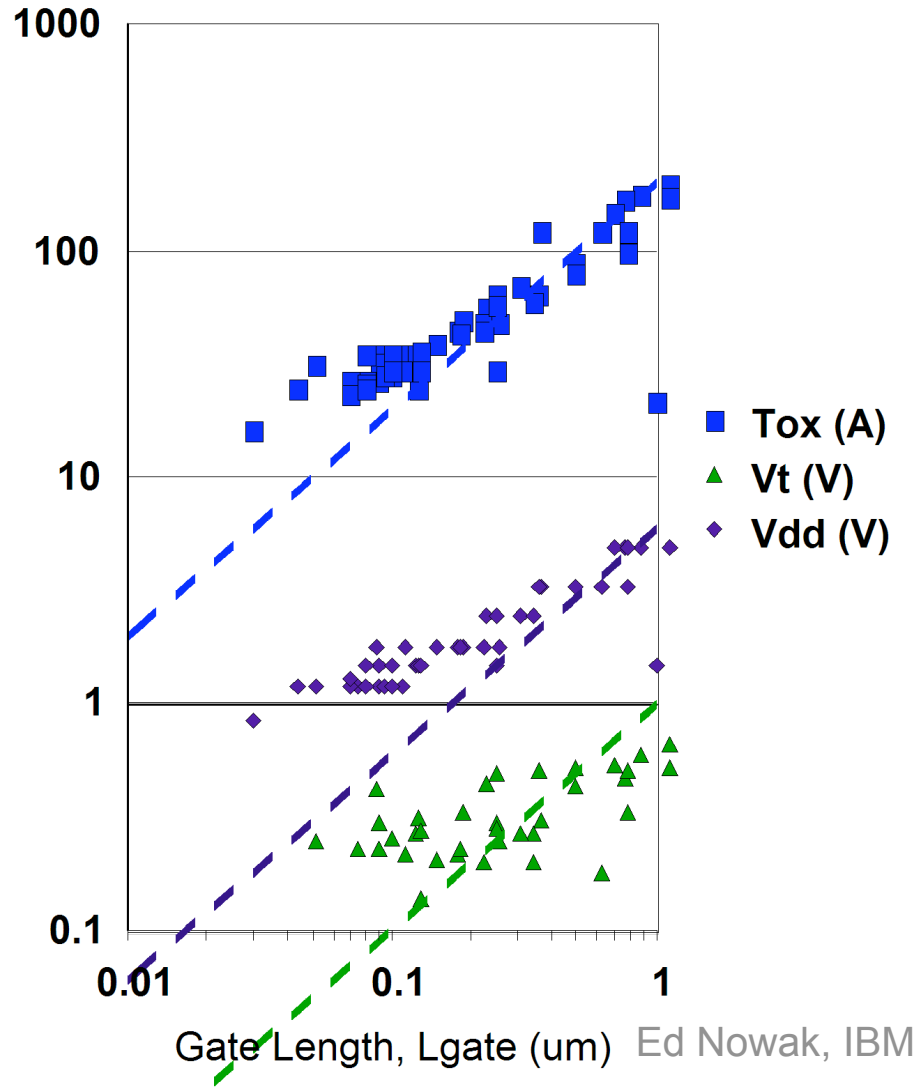
- kT/q does not scale
- Leakage grows if Vth scales

Vdd/Vth sets gate speed

- If Vth doesn't scale
- Vdd scaling will slow down

If Vdd does not scale

- Energy scales slowly



Ed Nowak, IBM

Technology Scaling Today

Device sizes are still scaling

- Cost/device is still scaling down
- This is what is driving scaling

Voltages are not scaling very fast

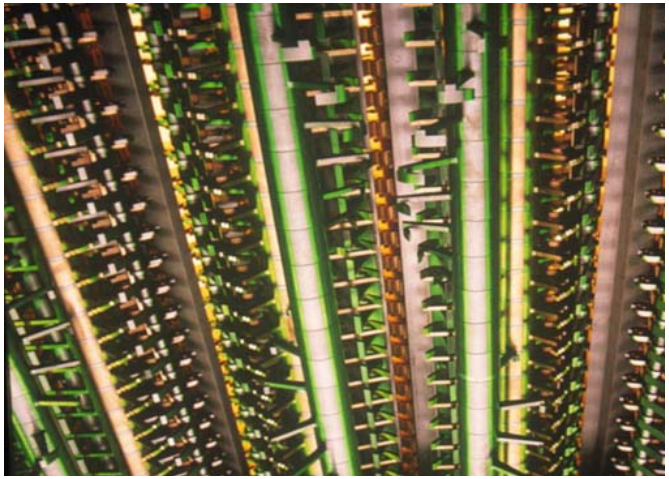
- Threshold voltages set by leakage
- Gate oxide thickness is set by leakage

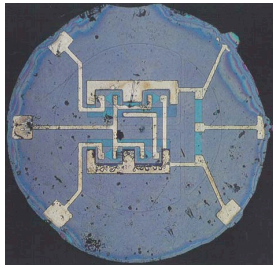
  This means that the channel lengths are not scaling

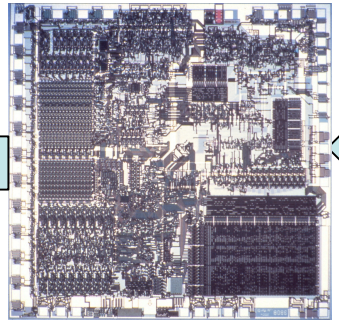
  Current is increasing by stressing silicon

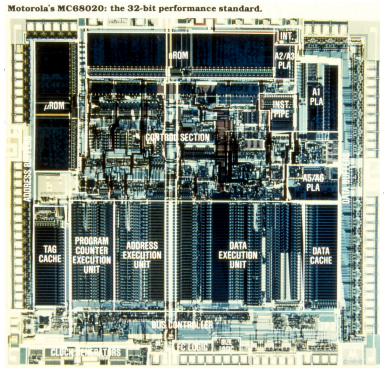
Now Vdd and Vth are set by optimization

What About Technology X?









X

Alternative Technologies

For computing, I am not optimistic

Current problems are set by Physics:

- Vdd set by kT/q

  Sets the on-off ratio
- Wire energy by $CVdd^2$

To get around these limitations

- Need to create something very different!

Problem with Different Technologies

If you build it, generally they don't come

- Unless they absolutely have to

Design processes have been optimized for silicon

- Working on making it better for over 30 years

Silicon has set:

- Notions of logic (binary signals), digital design styles
- Computing (distinct memory and logic)
- Relative size and speed of memory logic

No new technology will fit this mold well

- Changing the world is hard

Maturing of Silicon

Silicon will not disappear

- It will still be a huge business
  Growth rate is slower, Eventually very slow scaling

Silicon has become like concrete and steel

- Basis of a huge industry
- Critical to nearly everything
- But fairly stable and predictable

Will remain the dominate substrate for computing

- And performance be limited by power dissipation

This is an Exciting Future

Will see tremendous innovative uses of computation

- Capability of today's technology is incredible
- Can add computing and communication for nearly $0
- Key questions are what problems need to be solved?

Most performance system will be energy limited

- These systems will be optimized for energy efficiency
- At fixed power, more ops/sec requires lower energy/op

  Technology is no longer providing the needed reductions

Both will require us to rethink our approach to design!

Outline for Rest of Talk

Explore energy-optimized designs

- Brief digression on DRAMs

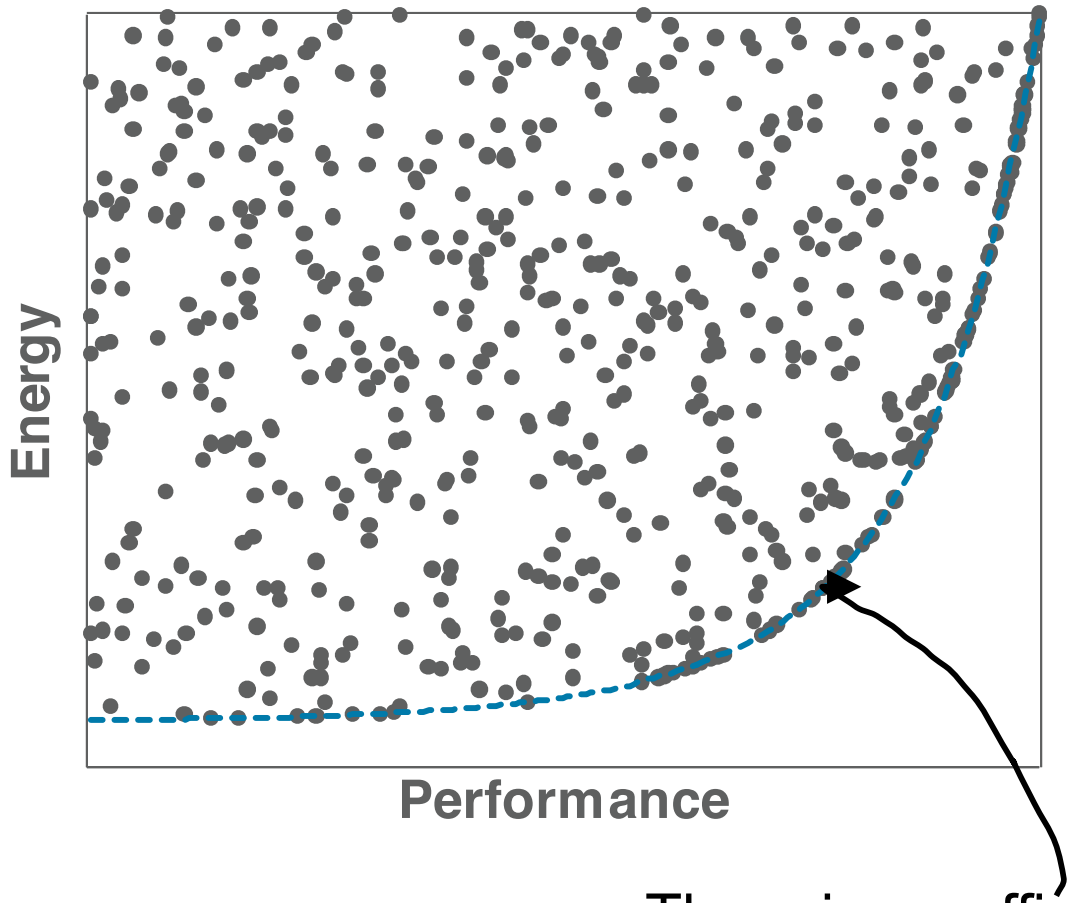
Need to create customized solutions

- Design NRE is the big problem

Turning the design process inside out

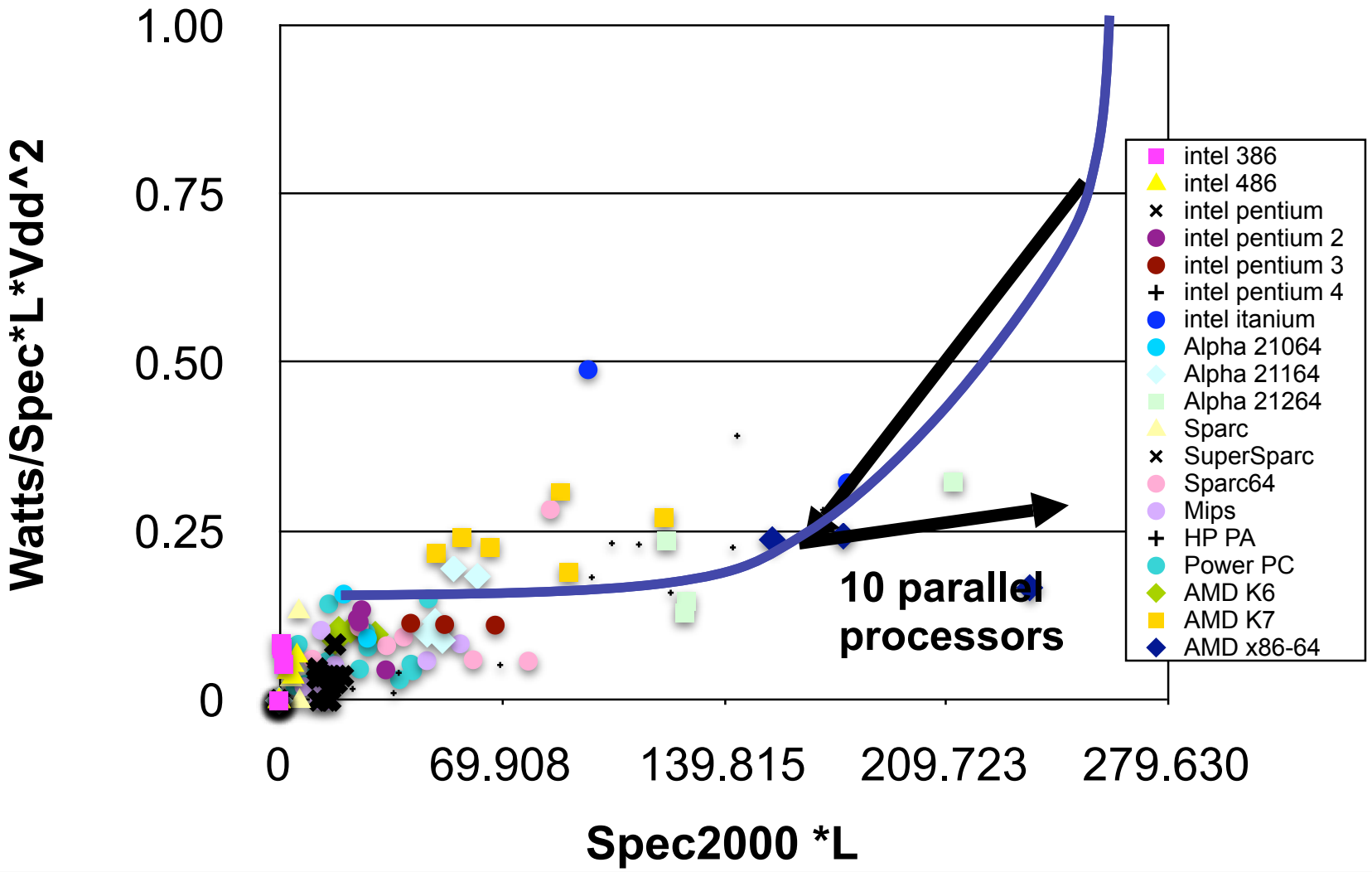
- Create chip generators not chips

To Optimize Performance/Energy: Sell High Energy / Buy Low Energy

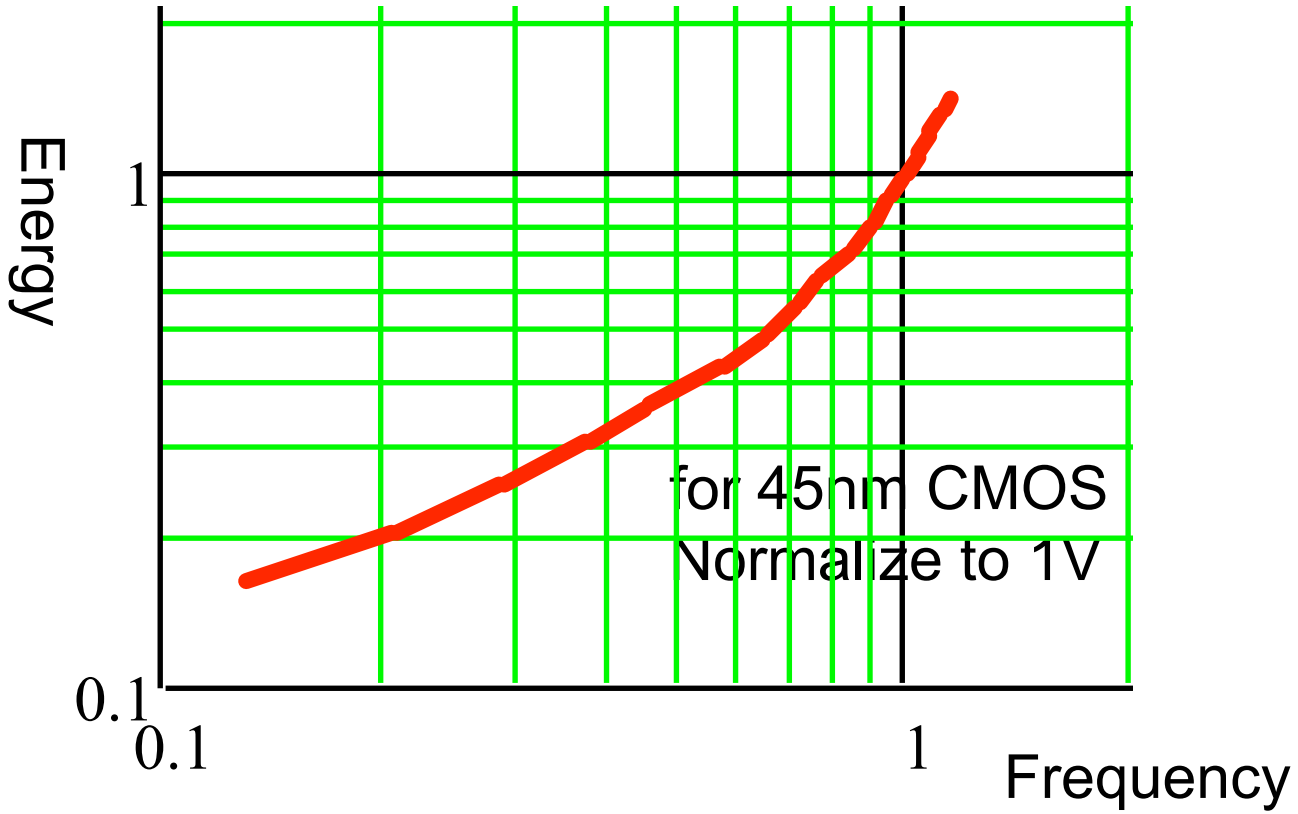


There is an efficient frontier

Exactly the Push for Parallelism

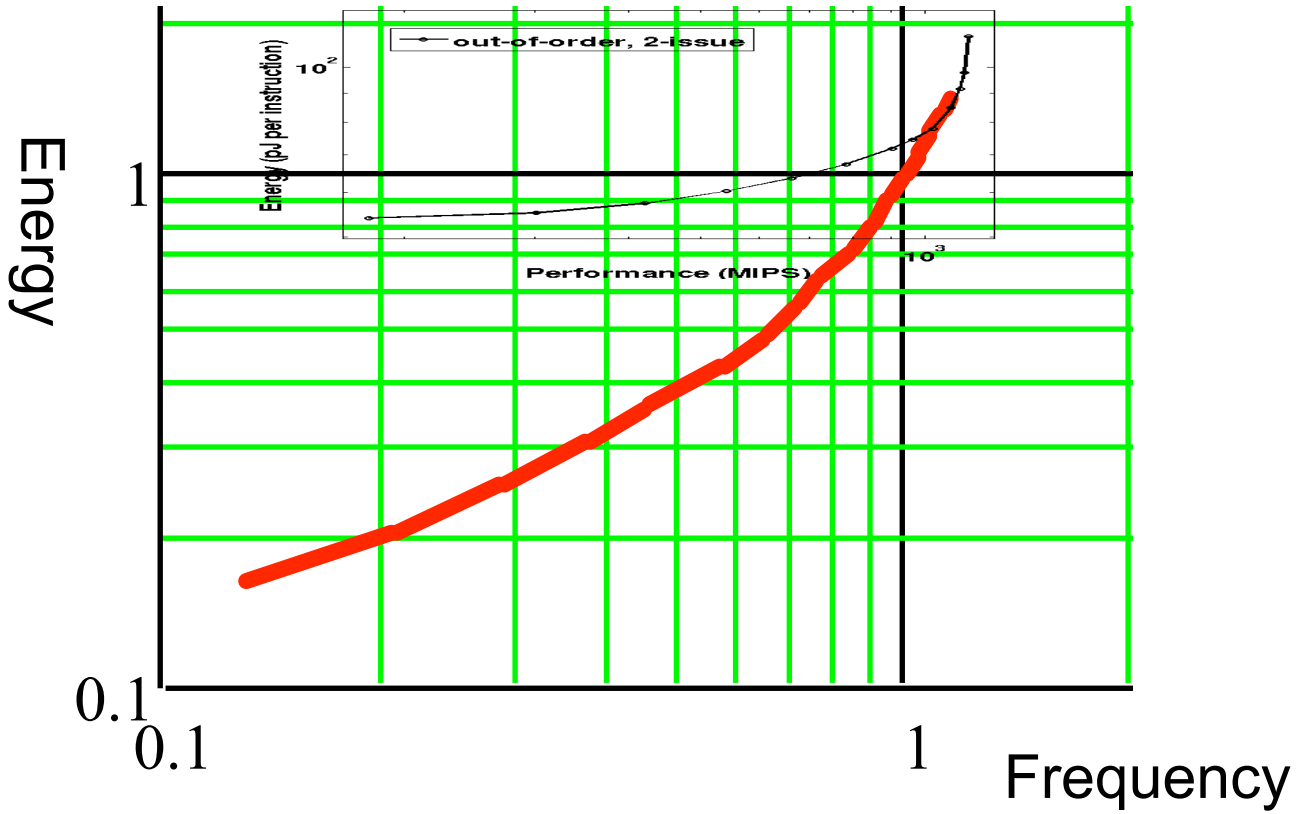


Sure We Can Scale Vdd



But we are just trading performance for energy

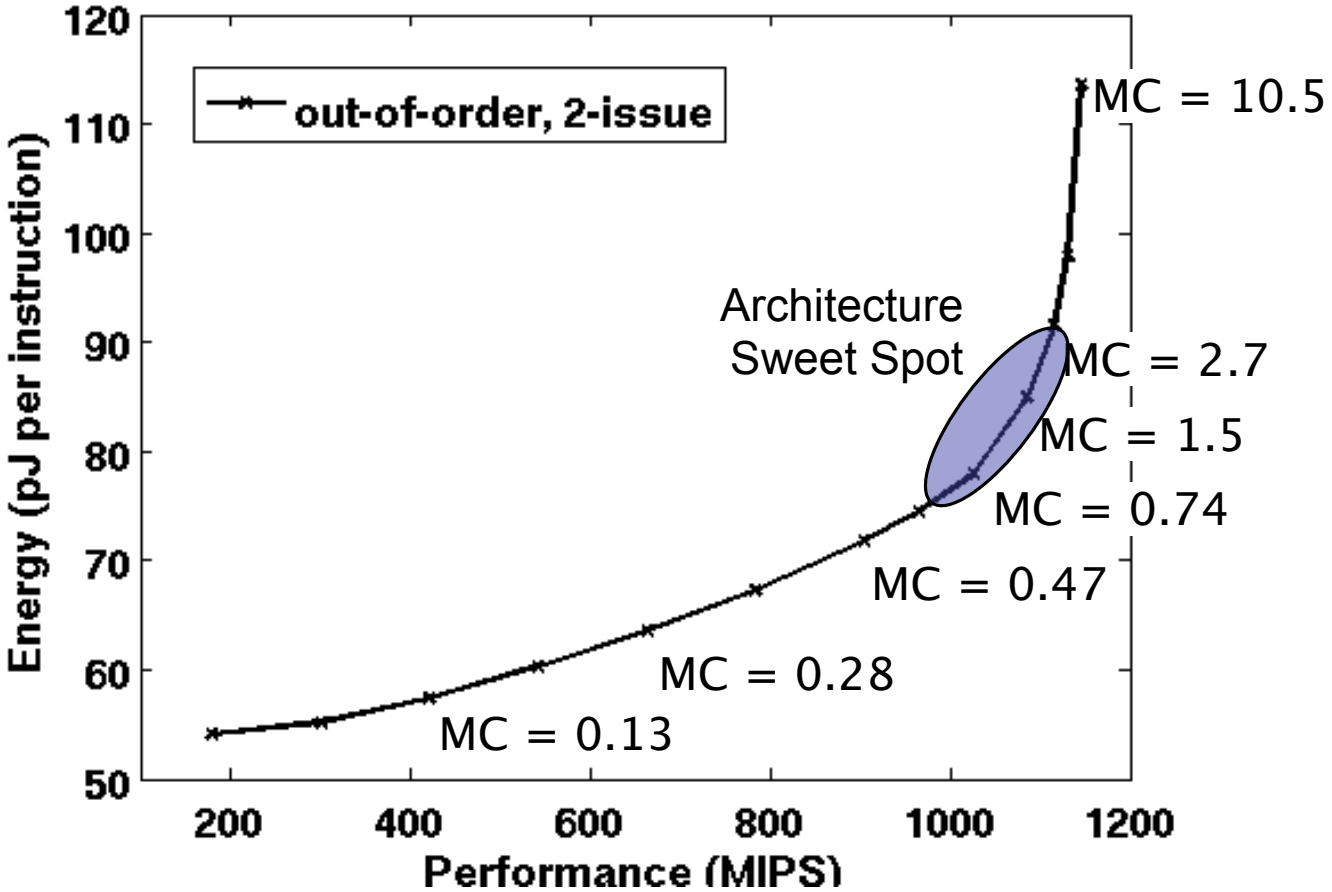
Vdd and Architecture Choices are Linked



Need to match marginal costs

- This is the slope on log/log plot

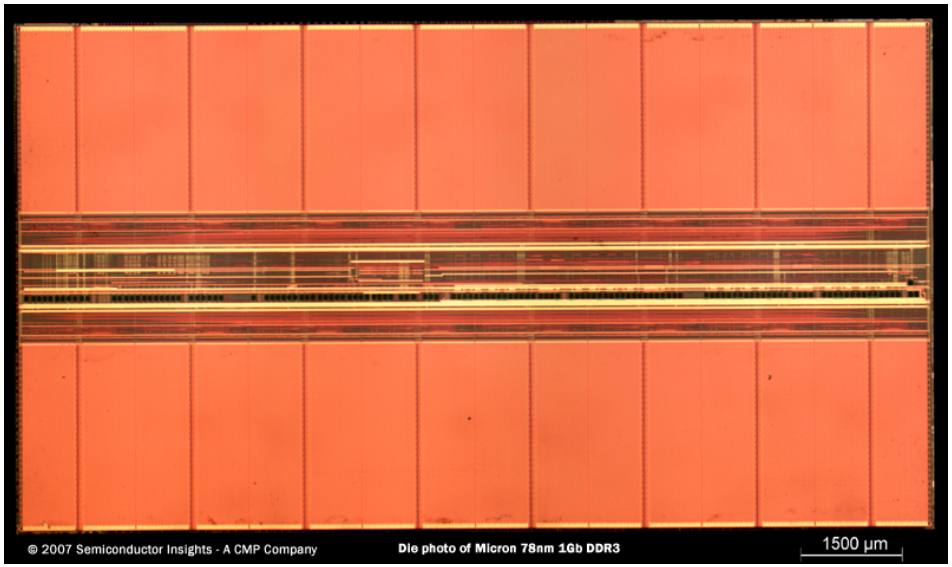
Useful Architectural Space is Small



Blame the DRAM

Current DRAMs are about 1nJ per 32bit word

- 30 pJ/bit
- Black box



This is high energy

- Limits energy saving

Critical issue in many high performance designs

TSV Does Not Help This Problem!

With current DRAM interfaces

- Roughly ½ the power goes to I/O
- This uses 15-20 pJ/bit I/O

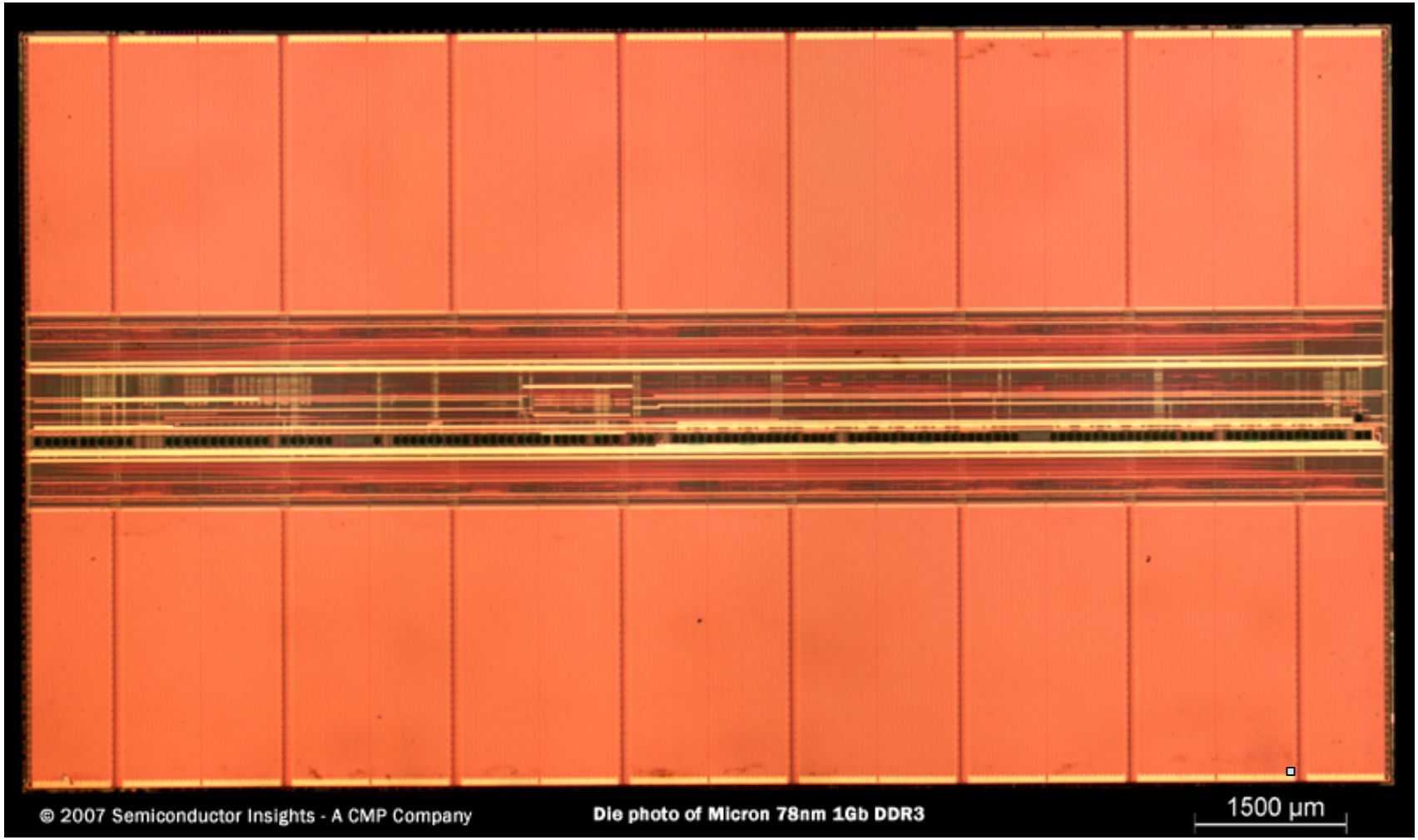
Intel / IBM / Rambus have all demo'd <3pJ/bit I/O

- Thus the interface power will be a small percent of the total

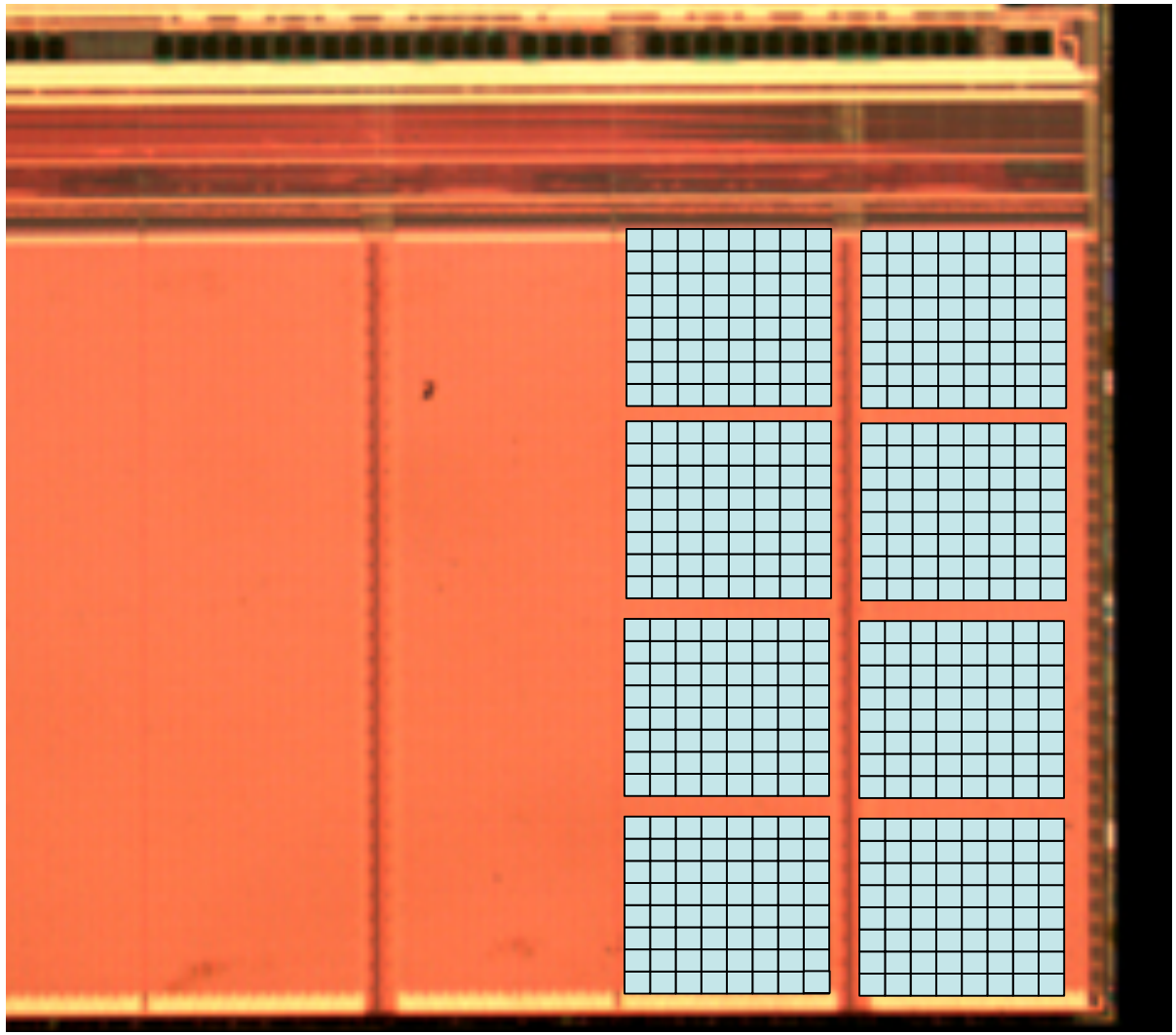
Rambus has described a fast wake up interface

- 20ns from power down (no clocks) to operation
- 3pJ/bit is the total DRAM and controller I/O energy

Energy Efficient DRAM



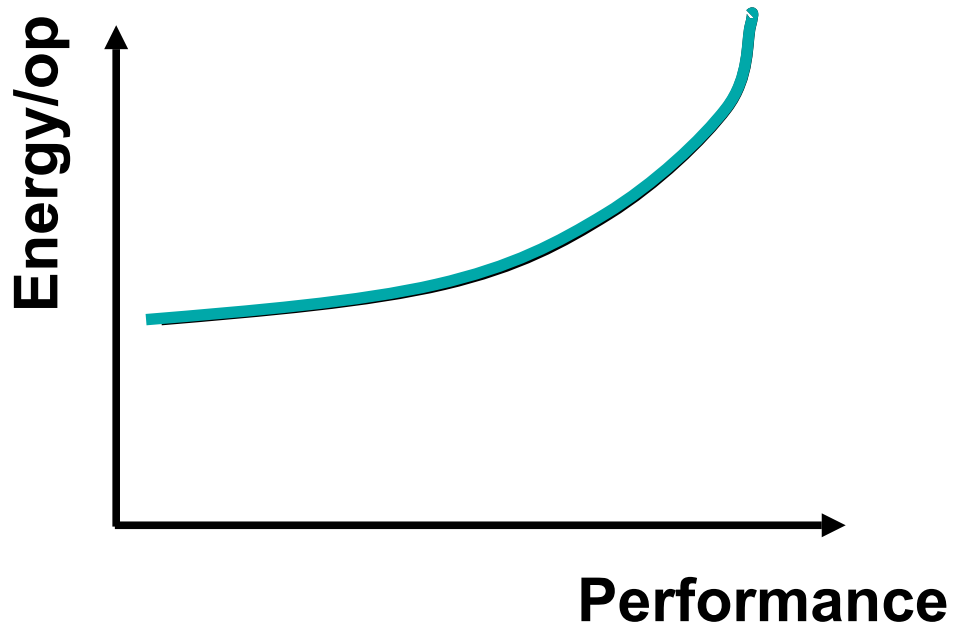
Energy Efficient DRAM



Key to High Performance/Low Energy: Problem Reformulation

Best way to save energy is to do less work

- Energy directly reduced by the reduction in work
- Reduces the time for the function as well



Exploit Specialization

Optimize execution units for specific applications

- Reformulate the hardware to reduce needed work
- Can improve energy efficiency for a class of applications

SIMD/vectors units can be more efficient than CPUs

- Exploit locality, reuse
- High compute density

ASICs are more efficient than DSP/Vector engines

- If we want efficiency, we need more application optimization

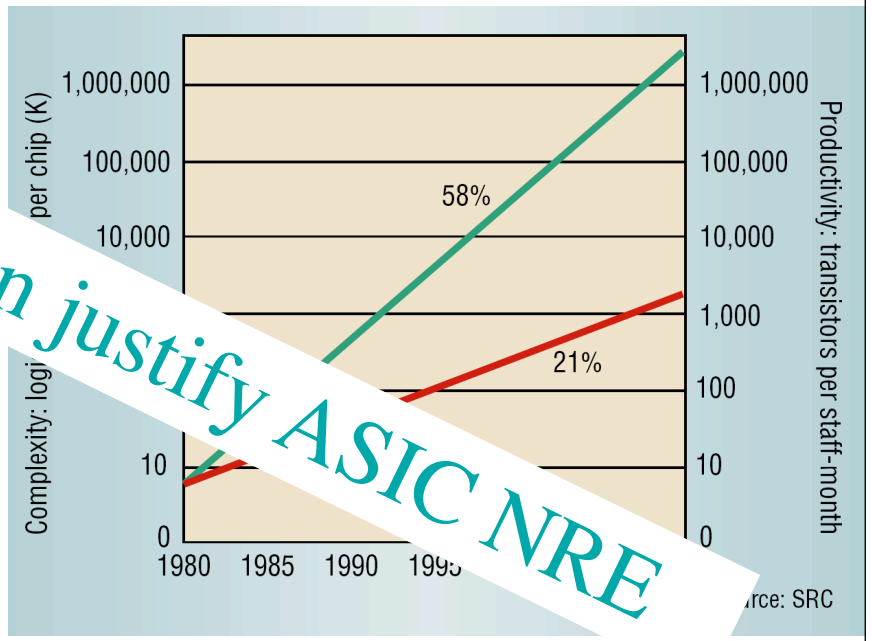
ASIC/SOC Design Trends

Rising non-recurring engineering costs

- Increasi[illegible]gn complexity
- Growing [illegible]omplexity
- Challenging phy[illegible]

- Mask and fabrication costs
  Less than 10% of total NRE

Few markets can justify ASIC NRE



So Close, and Yet So Far

Have a technology that is nearly infinitely capable

- But it costs too much to design ($20M)
- Unless you have a very large market to serve
- Fewer people are designing chips today

Reminds me of the mid 1980's

- Custom chips could do a lot, but were expensive to design
- Only got standard products
- But then synthesis and place and route were invented

  Created the ASIC business model

Need a new design innovation!

Gordon Moore Was (Is) a Smart Man

Design cost

Power dissipation

What to do with all the functionality possible

ftp://download.intel.com/research/silicon/moorespaper.pdf

Doing Better?

Chip design is expensive since chips are complex

But the building blocks are well known

- Many of the optimizations are well known too
- Designers often do many of the same steps
- Part of the reason for off-shoring

  Don’t need experience

Getting the system to work is hard

- There is a lot of turning the crank that is needed

Can we automate some of the crank turning?

What I Want

I want to work at the architectural level

- What this is depends on the system being built
- Highest level that controls the performance

  Hardware = the architecture/microarchitecture

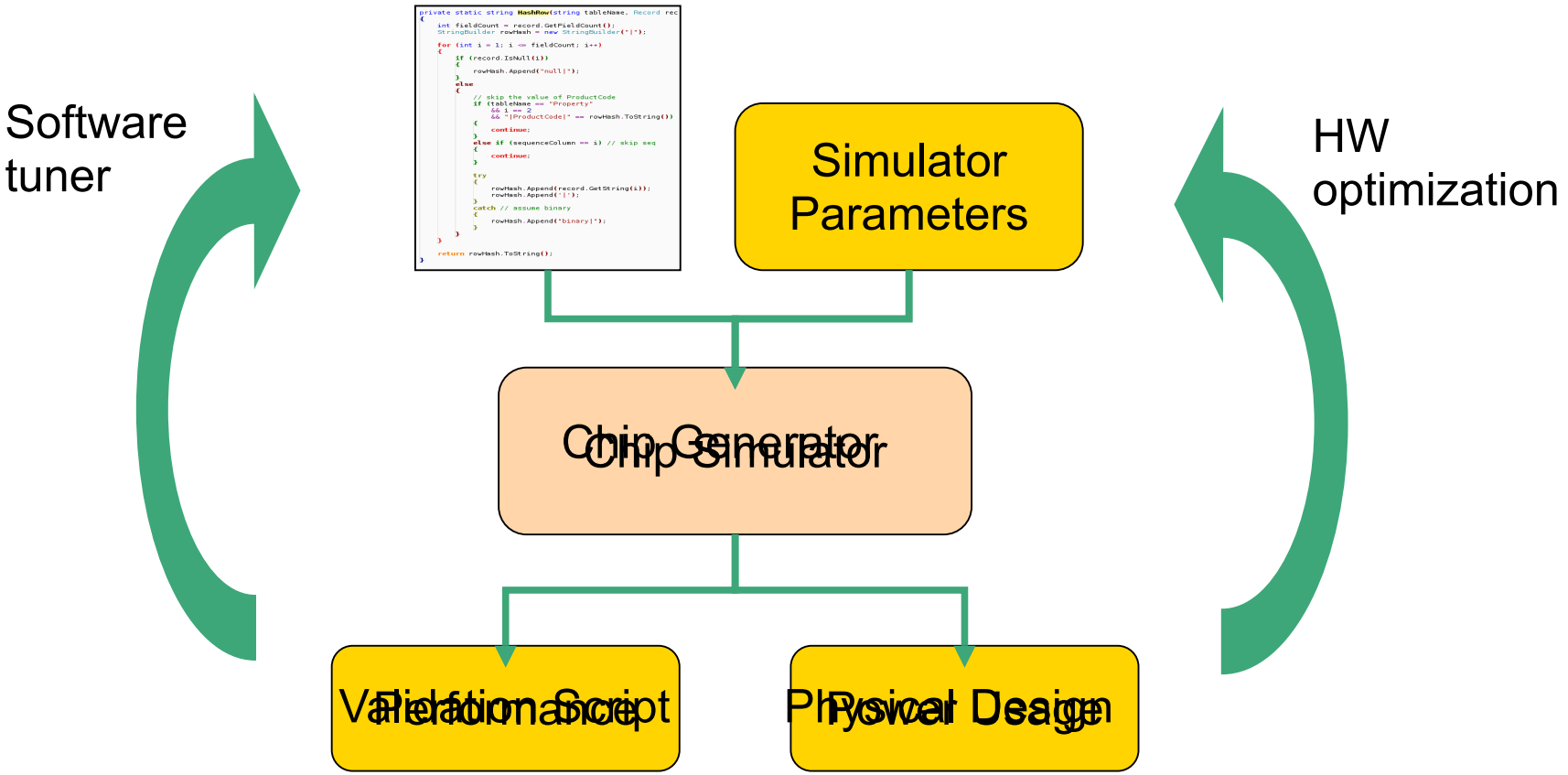
  Application = application architecture

Want to start with an architectural simulator

- With lots of parameters

  These exploit previous cleverness
- Which is extensible
- Has a software tool chain

I Want So Much More



I want a chip generator

Silicon Compilers Again?
Haven't We Been There, Done That

Well yes and no

- I do want the system to generate silicon (yes)
- I don't believe it will take application code and "compile" it (no)

Think of it as a way of encoding design knowledge

- We think of many alternatives in a design; then choose one
  For the next application perhaps another alternative is best
- We optimize designs at each level, but then freeze them
  What happens if we let the design remain flexible?

Started thinking about this for VLSI design

- Now I think it is much more important for embedded systems

Rethinking Digital Design

Build chip generators, not chips

- No one builds a single chip, it is too expensive
  Everyone is planning a follow on chip
- Basically create the design so it can be leveraged, explicitly
  And I am not talking about reusing the parts, but the system
- Initially make it just flexible enough to be useful
  Don't try to create an impossible system

Chip generators solve many problems

- Raises the level of design
- Allows the design to be globally optimized
- Flexibility is done in a constrained environment
  Don't have arbitrary connection of complex blocks

Turns Today's Design Process Inside Out

Conventional System-on-Chip Design:

- Designer creates system from complex components
  - **IP components are designed in advance**
  - **End with a squashy design of fixed components**
- SoC designer has to connect multiple IP blocks:
  - **Interface adapters between different blocks**
  - **Complex verification**

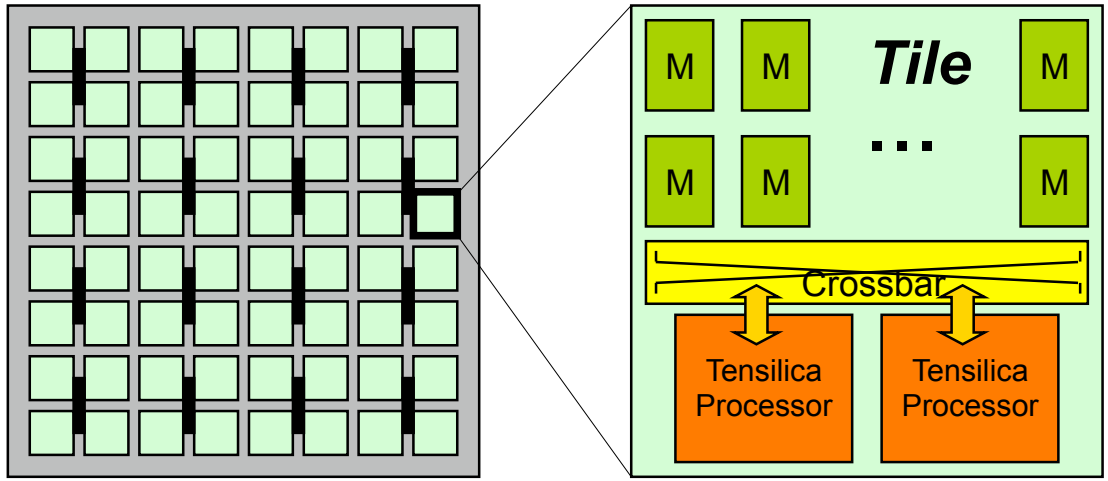
Generator:

- Designer tunes parts in a "fixed" system architecture
  - **Fixed design of squashy components**
- Functional interfaces remain constant
  - **Reusable validation**

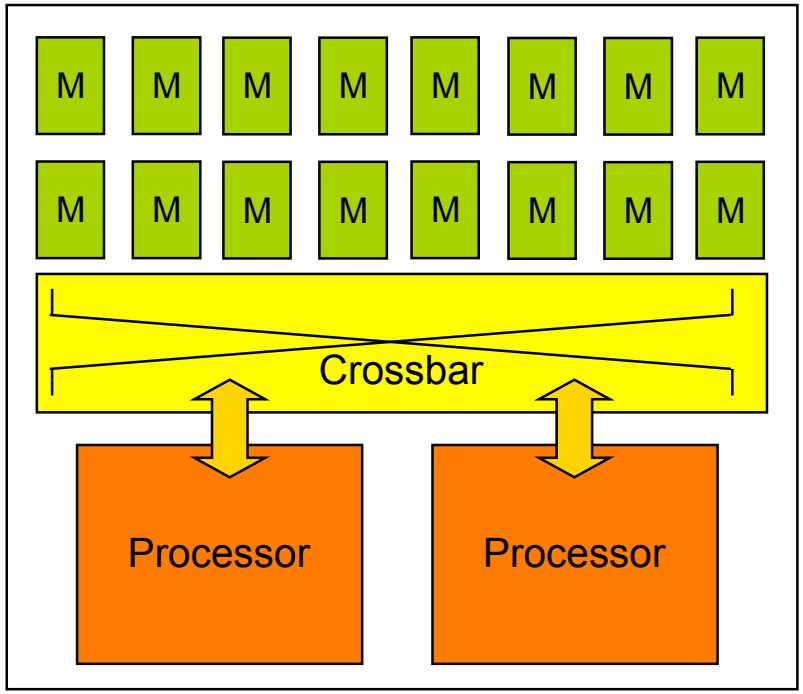
How to Test This Idea: Reconfigurable Chip Multi-Processor

Smart Memories - Scalable modular tiled architecture

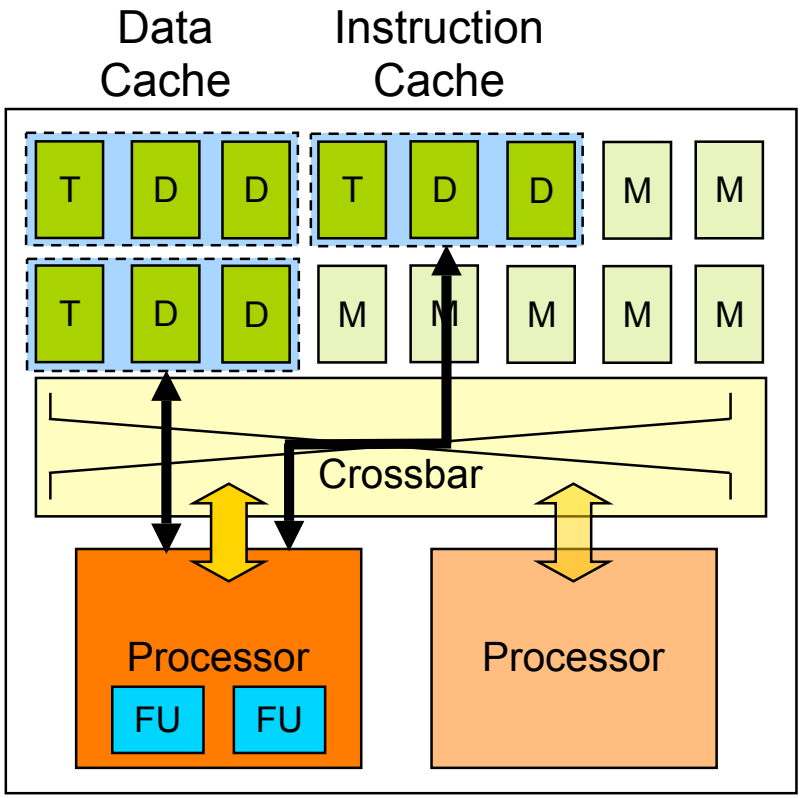
- Various memory structures supported:
  Caches, local memories/scratchpads, FIFOs
- Supports multiple programming models including:
  Cache coherence, streaming, transactional memory



Smart Memories - A “Pretend” Generator

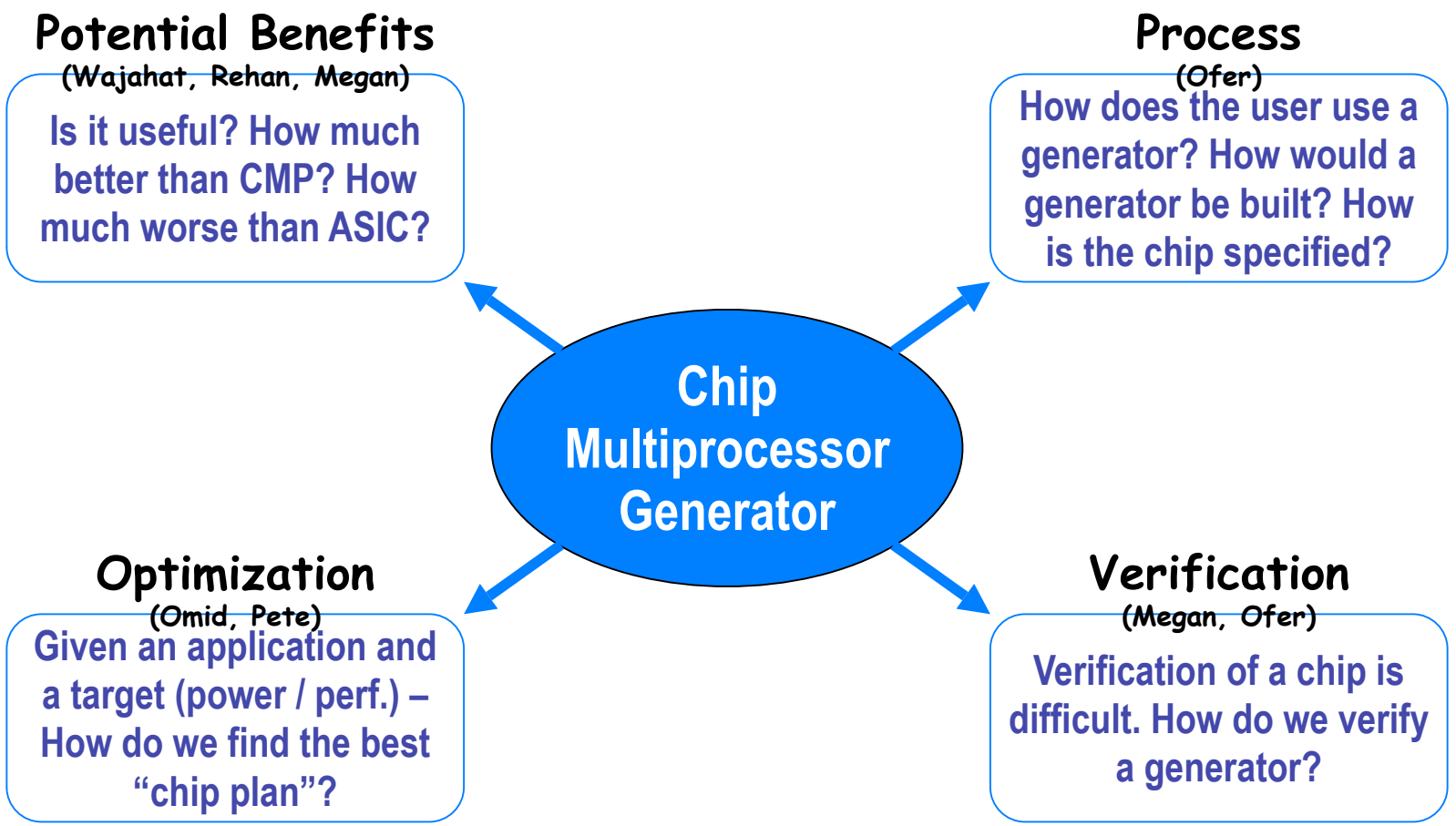


Smart Memories Architecture: Single Tile



Chip Generator Derivative

Use it to Explore Exciting New Challenges



Key Question:
What is the Benefit of a Chip Generator?

What is the potential gain in energy efficiency?

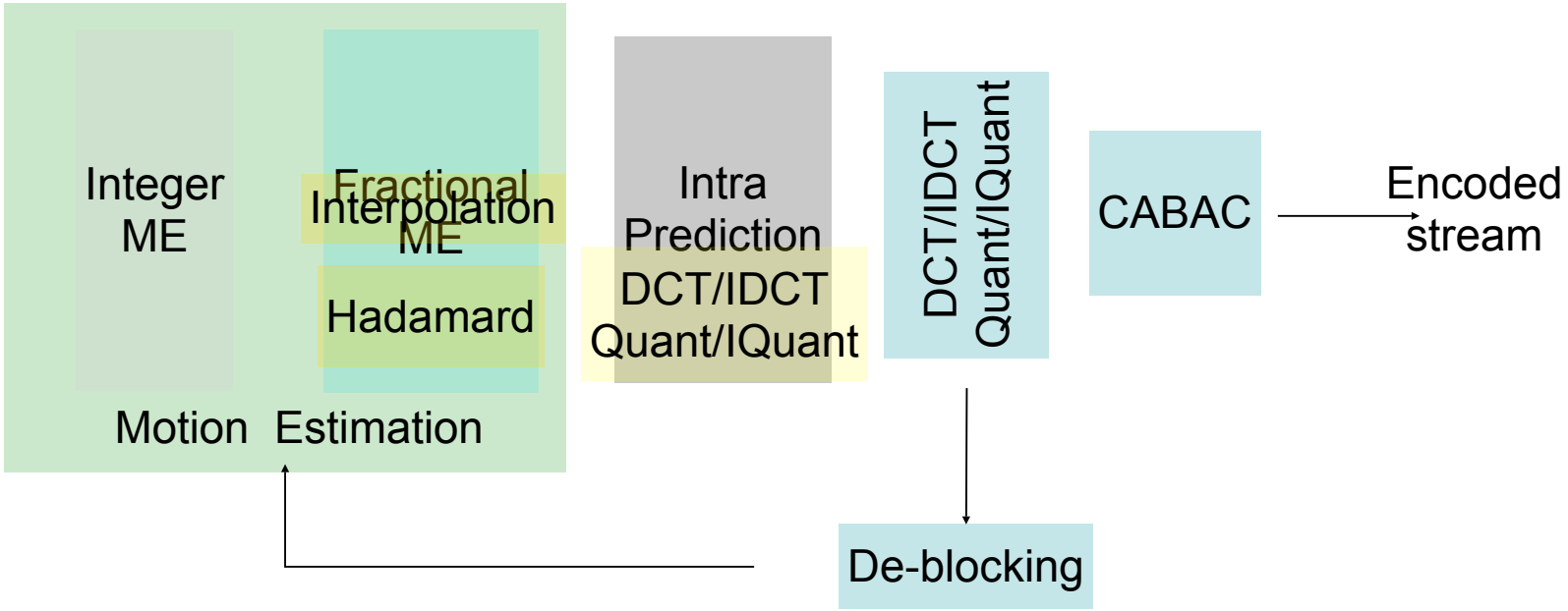
- Can we remove limitations of processing elements
  - **Overheads and limitations of parallel execution units**
  - **"Instruction" fetch overheads**
- Limitation on memory / communication elements

How application specific are these optimization?

- If not too specific, we can build a better universal machine
  - **Or at least build machines for wide application classes**

H.264

Encoder:

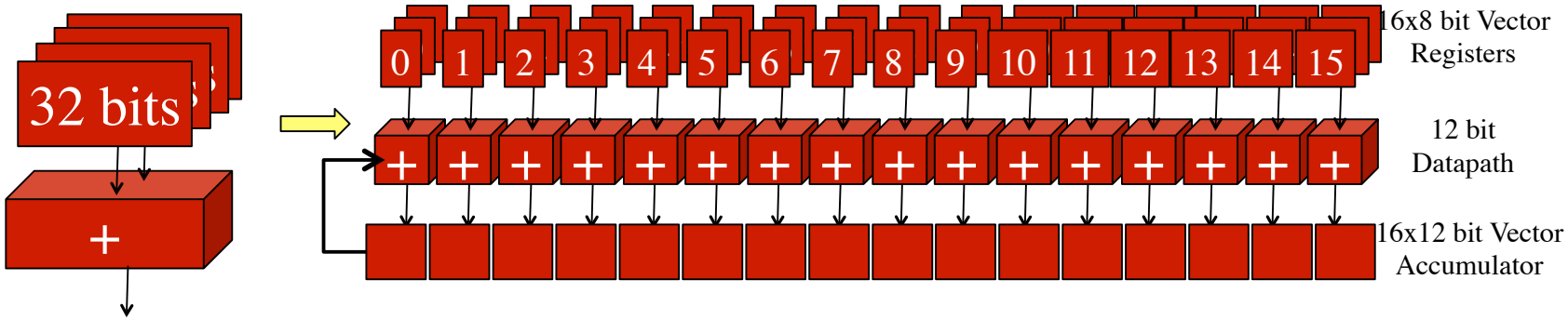


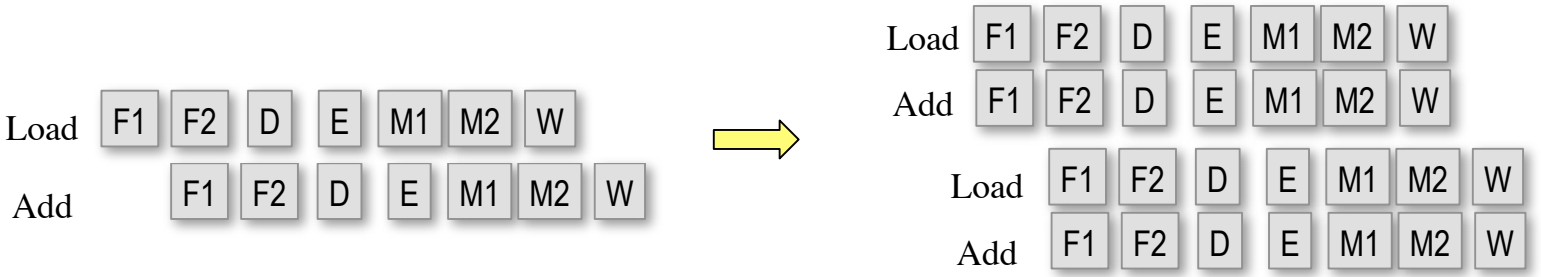
Efficiency ratio (mW/frame) is 500x worse in a processor than an ASIC

- We wanted to understand why

General Purpose Extensions: SIMD

16 Way SIMD, 2 Slot VLIW





110K gates, 20x performance improvement

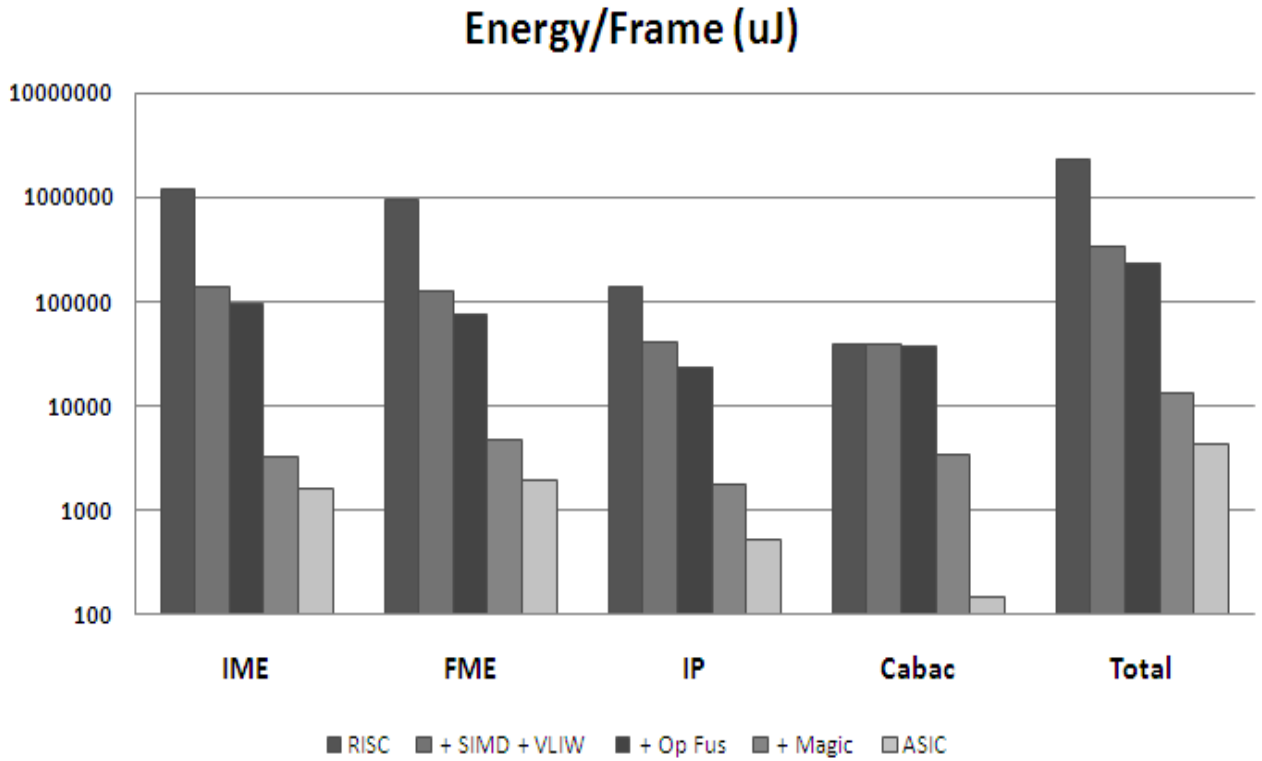
Final Result is Within 3x of ASIC

Generic optimization (data parallel / subword)

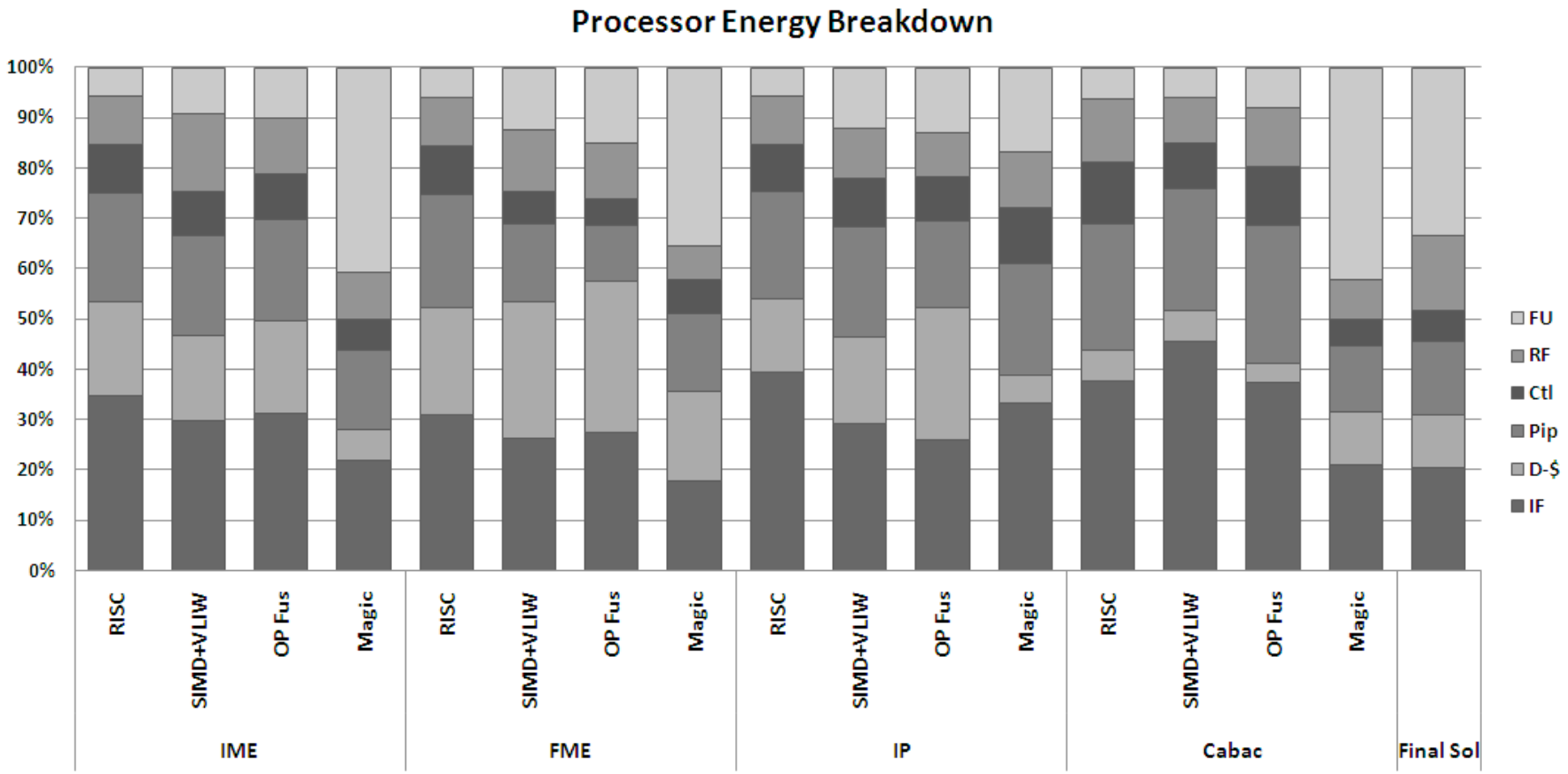
- Are good, but only give about 20x in performance 8x in energy

Final optimizations are very application specific

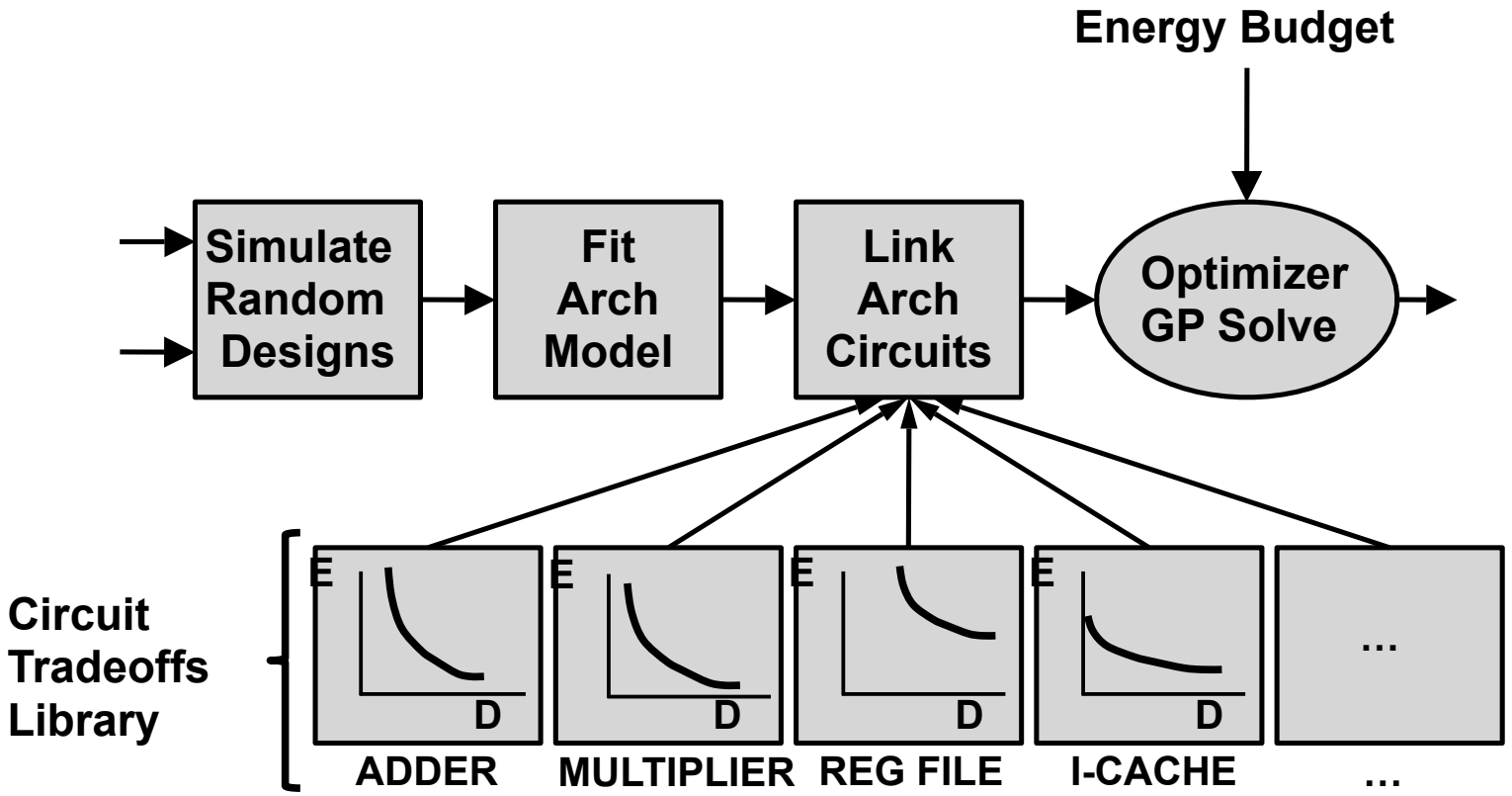
- But are worth 10-20x in energy!



Problem: Required Ops Are Low Energy



Key Piece of Domain Knowledge: Energy Performance Optimizer



Will be published at DATE 2010

Generating Circuit Tradeoffs

Synthesize each block for various delay targets

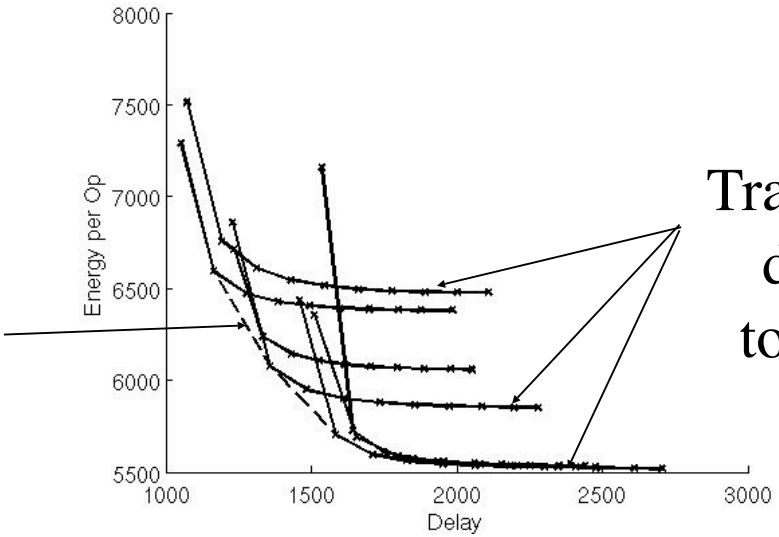
- To produce different circuit topologies

Optimize each topology for gate sizing

- Use LSGS to produce optimal tradeoff between energy and delay

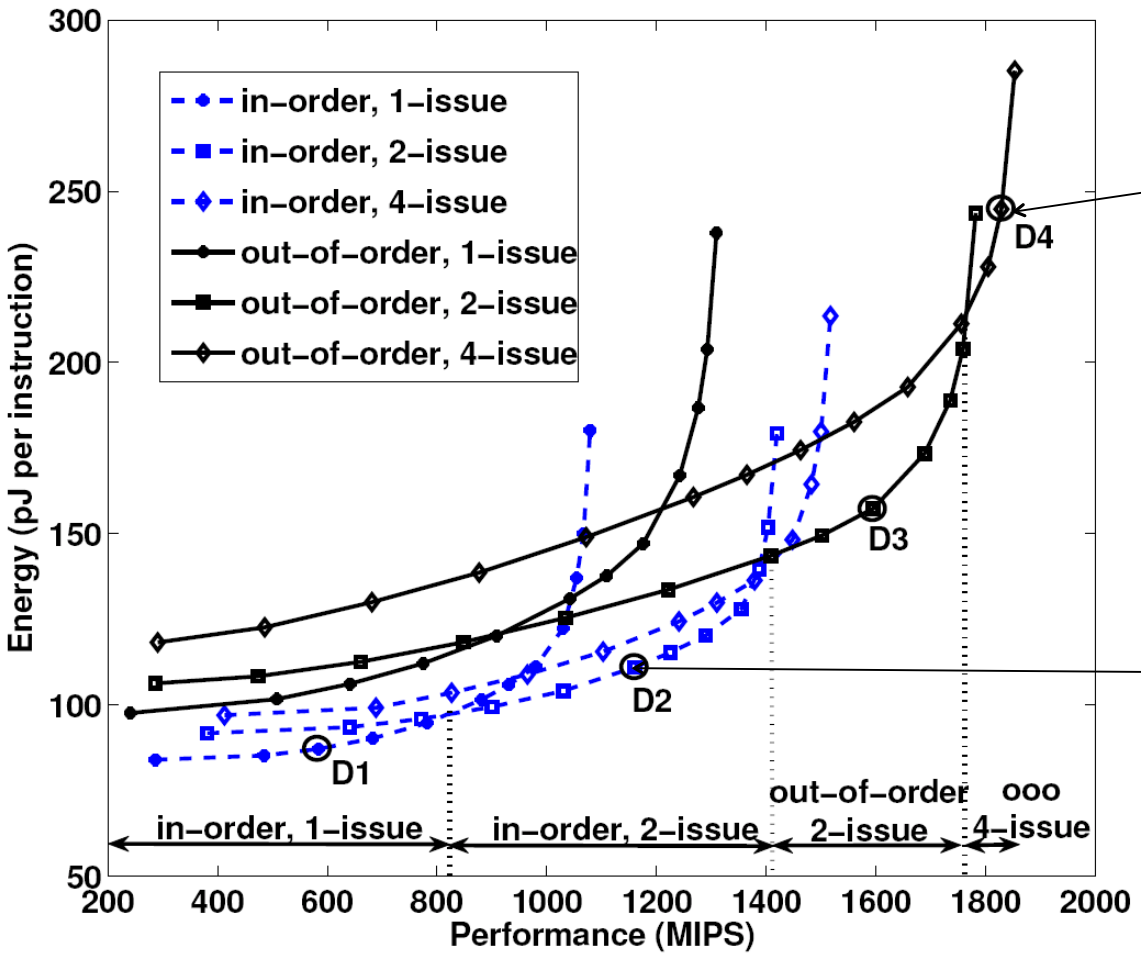
Merge circuit tradeoffs of each topology to results

Overall circuit tradeoff



Tradeoffs for different topologies

Preliminary Optimization Results



Clock Cycle: 16.3 FO4
Integer Unit: 1 cycle
I-cache size: 32Kb
D-cache size: 42Kb
Fetch lat: 2.1 cycles
D-cache lat: 1.1 cycle
IW size: 9 entries
ROB size: 32 entries
BTB size: 1024 entries
…

Clock Cycle: 16.9 FO4
Integer Unit: 1 cycle
I-cache size: 32Kb
D-cache size: 11Kb
Fetch lat: 1.6 cycles
D-cache lat: 1 cycle
IW size: NA
ROB size: NA
BTB size: 90 entries
…

Verification In The Chip-Gen Sense

Design verification is the biggest hurdle

- Said to account for 50%-70% of chip labor costs
- Expected to be main challenge of chip generator as well

We do not believe in “correct by construction”

- Everything can have a bug
- Need to generate verification collateral too

Creates opportunities for creating new tools

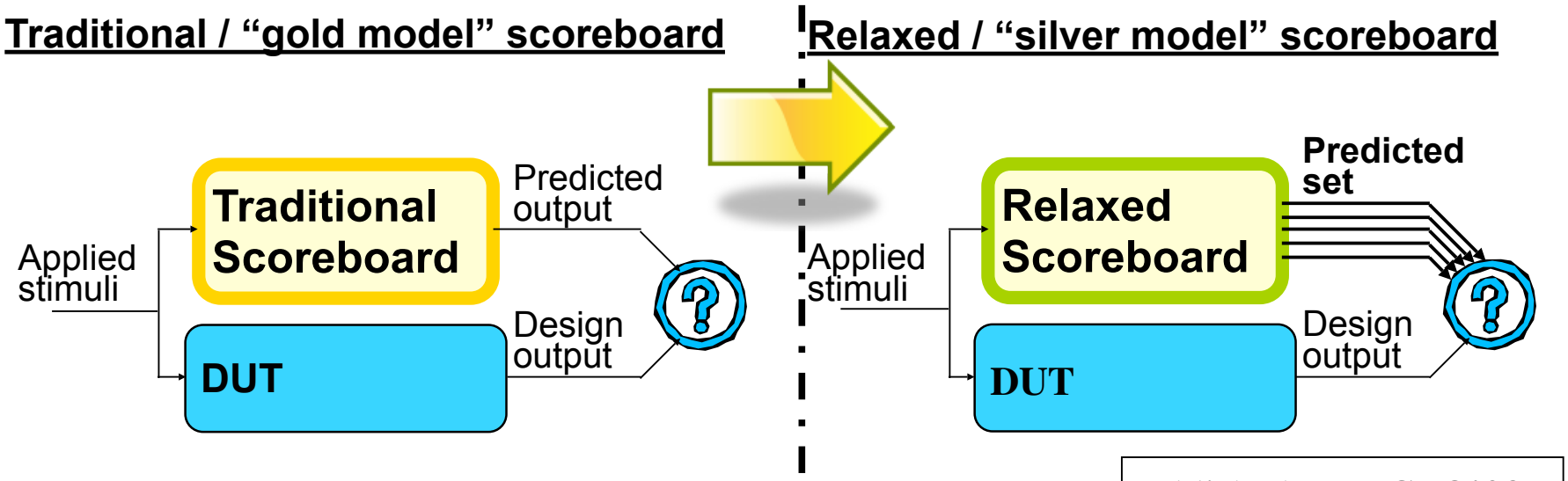
- Like the generic optimizer
- Surprisingly a generator can be easier to validate …

The Relaxed Scoreboard

Like other golden models,

- This is a global design checker but…
- It does not compare an observed value to a single value.

  Instead we keep a set of "possibly correct values"



Published at MICRO'08

Could Generators Help Validation?

Thought generators make validation harder

- Adds new state, new things to check
- Grows design options

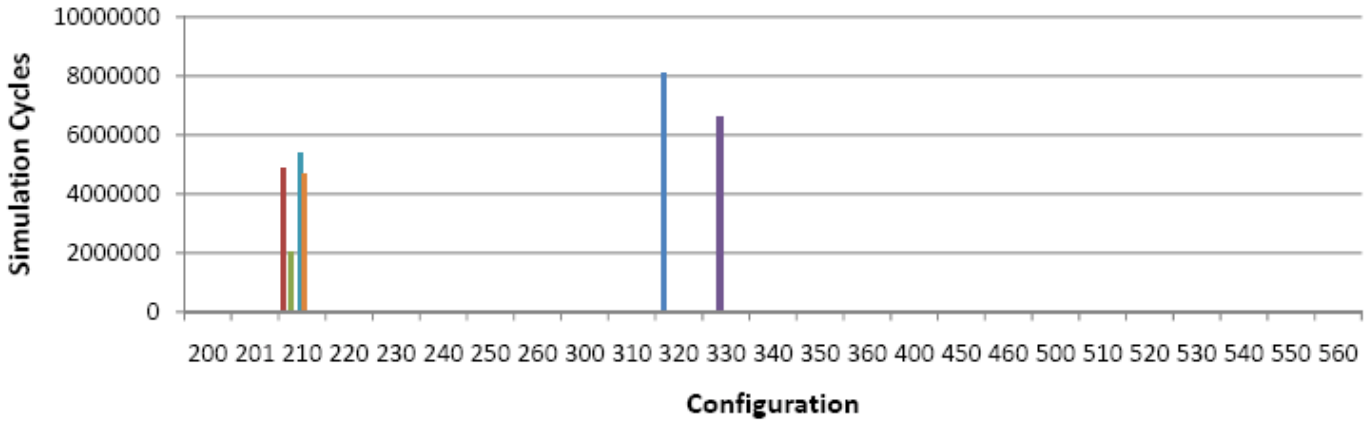
But we sometimes add flexibility to check designs

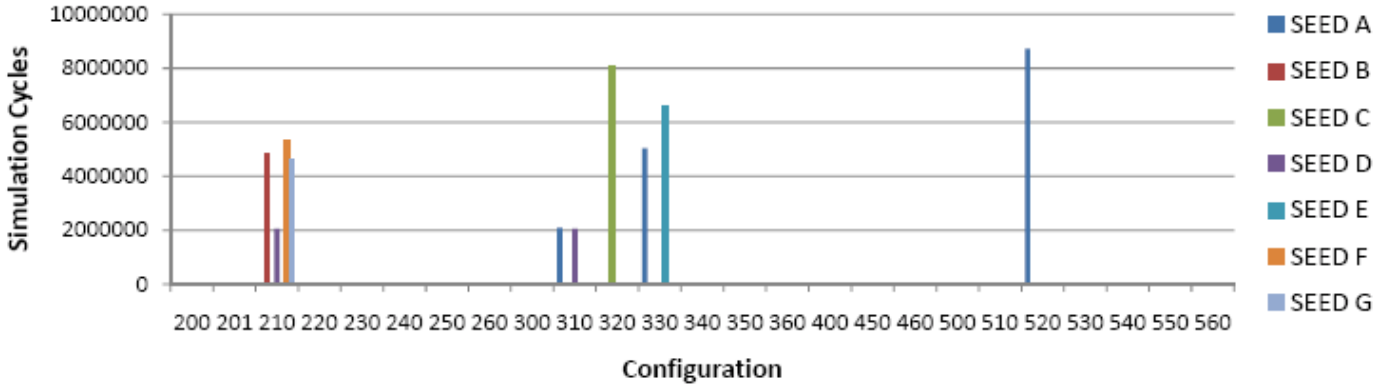
- Change queue sizes, ordering, timing
- Add randomness to stimulus

If the generator creates different variants

- Isn’t it just adding a different type of randomness?

Promising Data





Conclusions

The technology engine driving IT is slowing down

- Power efficiency is the real problem

Need to enable efficient application creation

- Both in the $ used to solution and Watts/Performance

Need to rethink design

- Turn conventional process inside out
- Codify tasks we know how to do for this application domain
- And validation might actually become easier too.