# Computing's Energy Problem:

## (and what we can do about it)

**Mark Horowitz**

**Stanford University**

**horowitz@ee.stanford.edu**

# Everything Has A Computer Inside




# The Reason is Simple: Moore's Law Made Gates Cheap

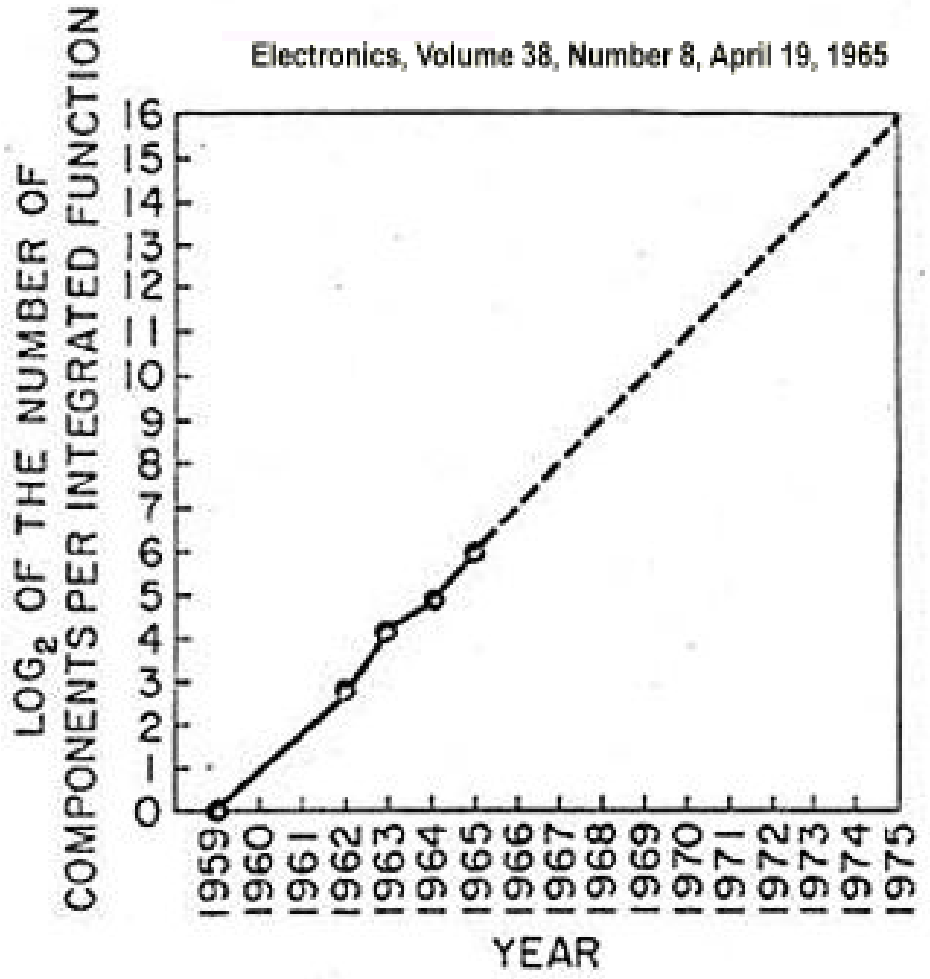
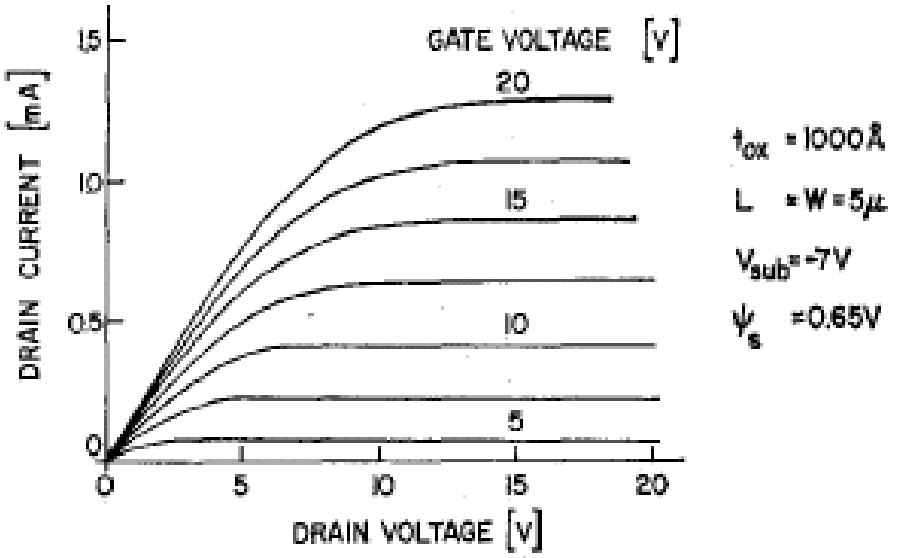
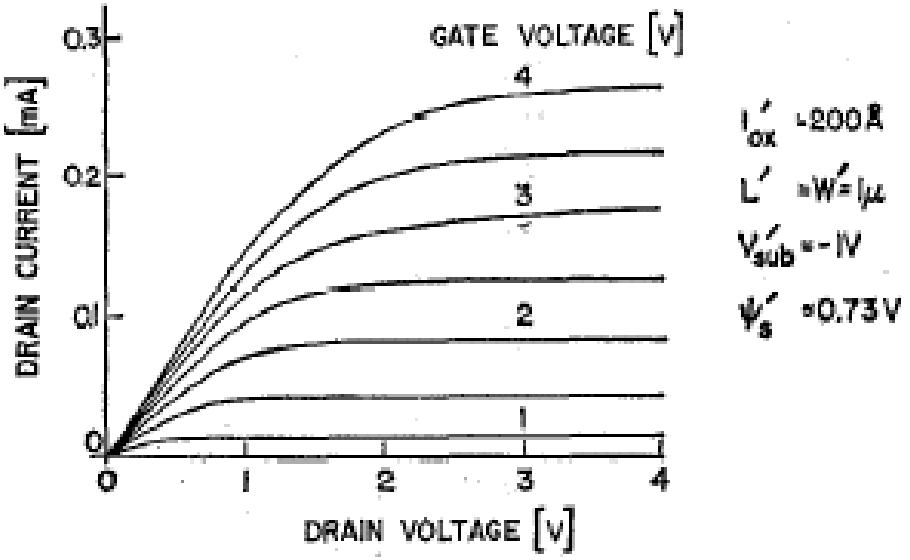



Fig. 2 Number of components per integrated function for minimum cost per component extrapolated vs time.

# Dennard's Scaling Made Them Fast & Low Energy




## The triple play:

| | | |
|---|---|---|
| • Get more gates, | $1/L^2$ | $1/\alpha^2$ |
| • Gates get faster, | $CV/i$ | $\alpha$ |
| • Energy per switch | $CV^2$ | $\alpha^3$ |

Dennard, JSSC, pp. 256-268, Oct. 1974

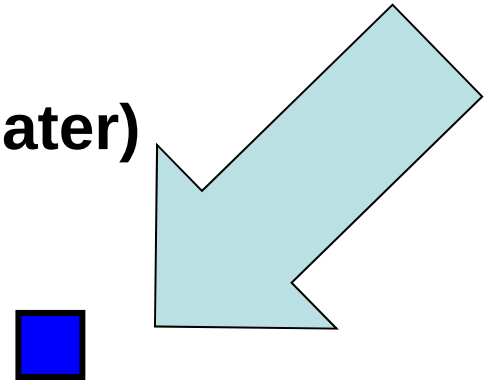
# Our Expectation

## Cray-1: world's fastest computer 1976-1982

- 64Mb memory (50ns cycle time)
- 40Kb register (6ns cycle time)
- ~1 million gates (4/5 input NAND)
- 80MHz clock
- 115kW

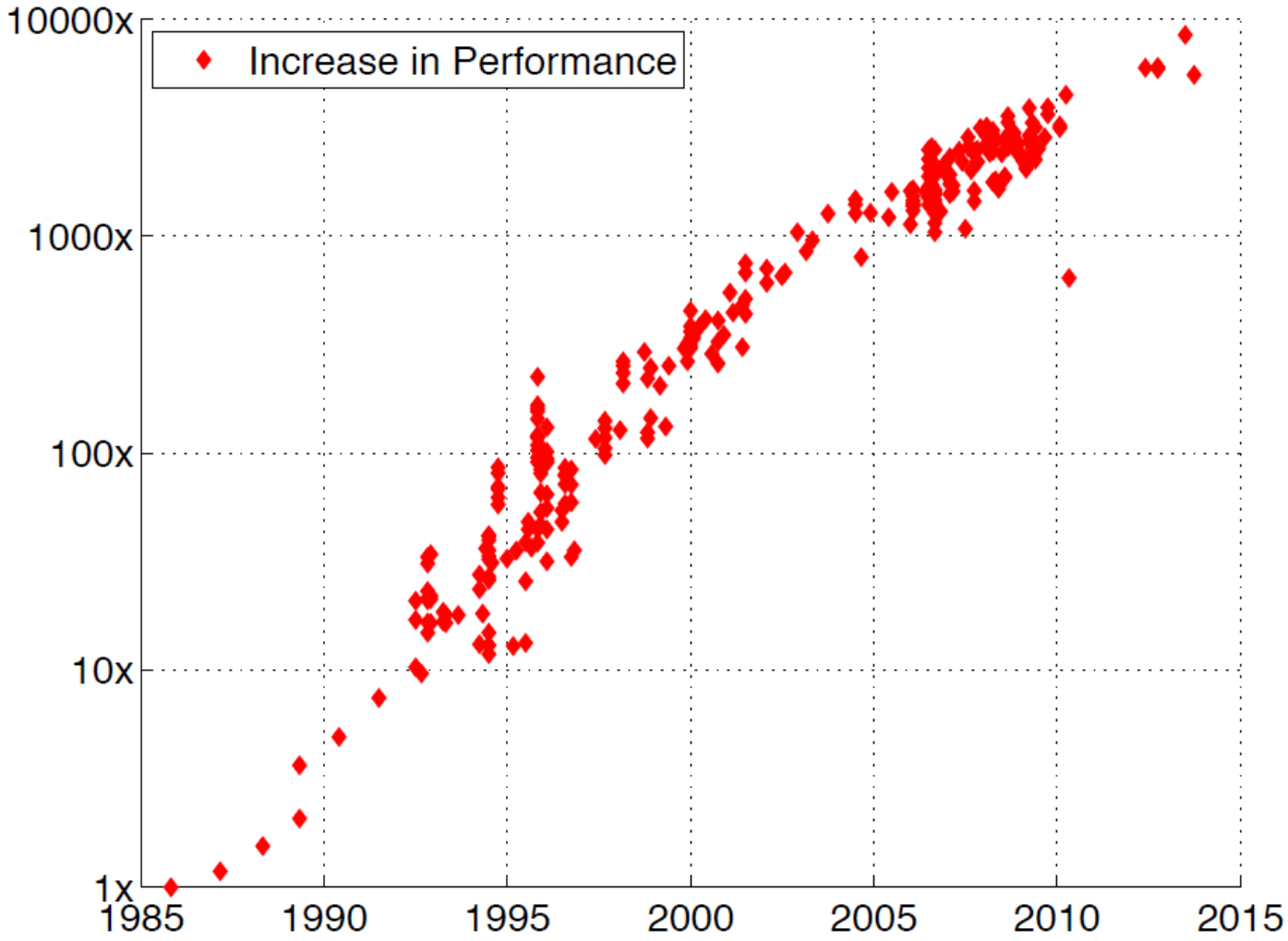
## In 45nm (30 years later)

- < 3 $mm^2$
- > 1 GHz
- ~ 1 W



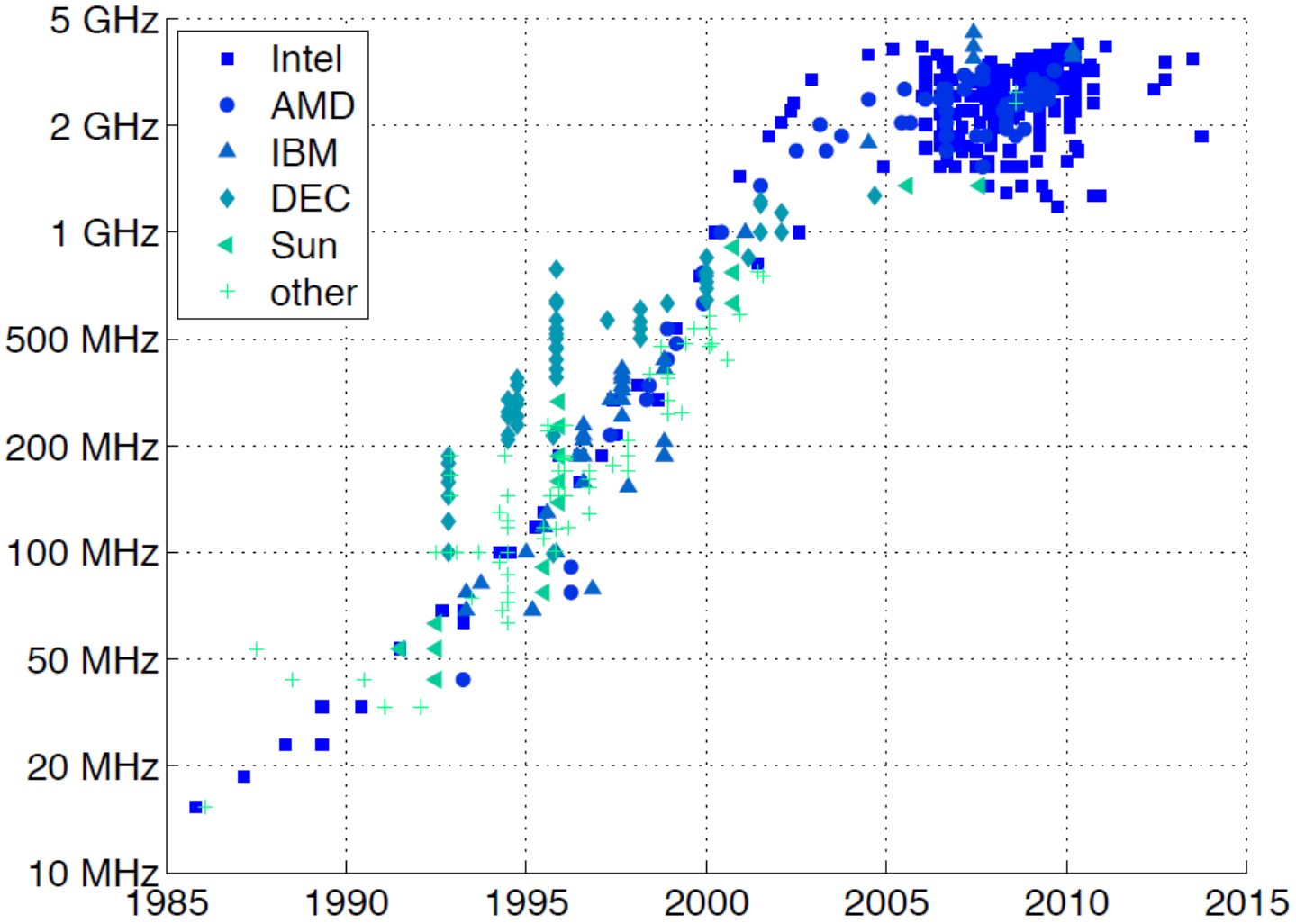
CRAY-1

# Supporting Evidence



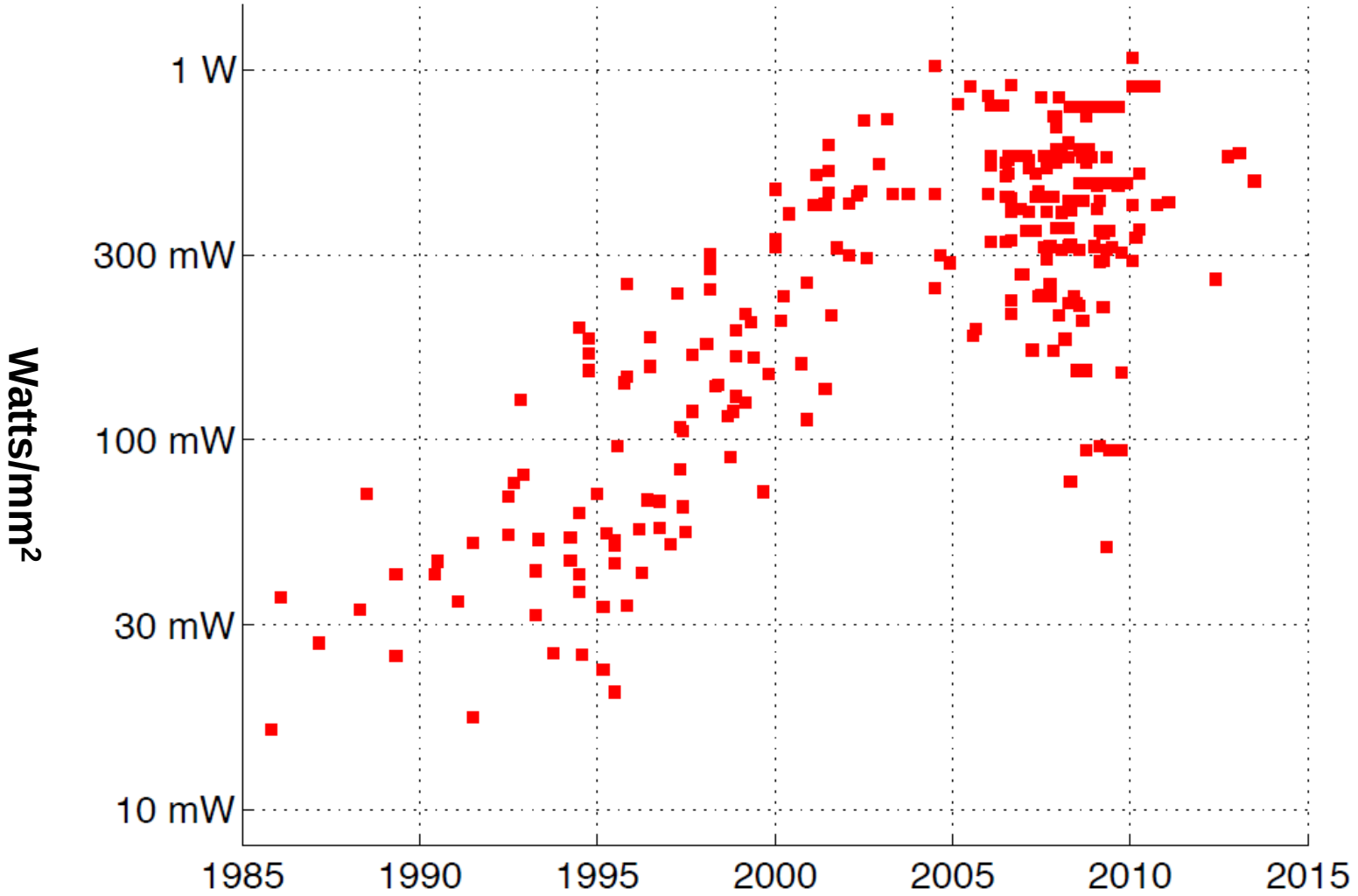

http://cpudb.stanford.edu/

# Houston, We Have A Problem



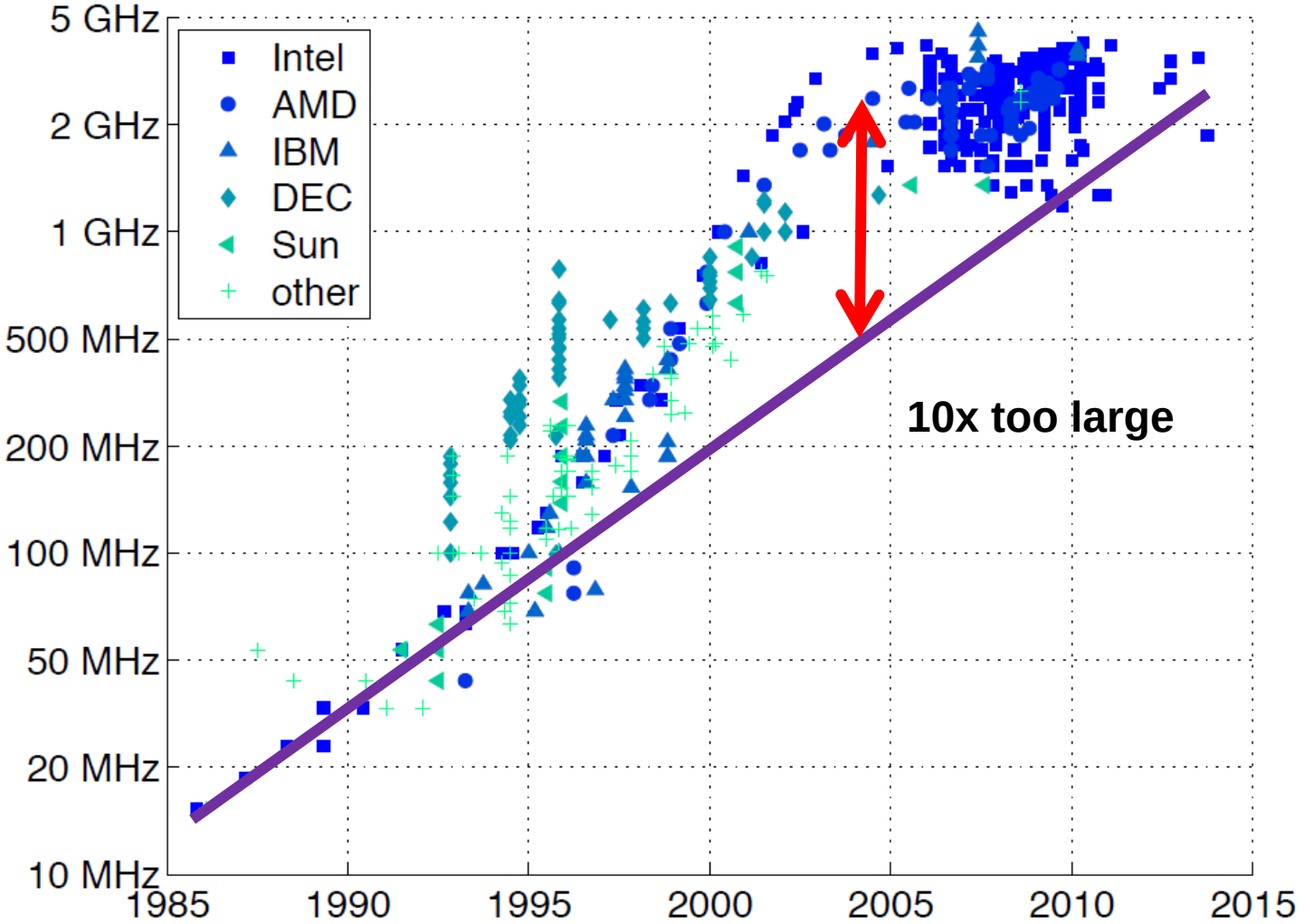

http://cpudb.stanford.edu/

# The Power Limit



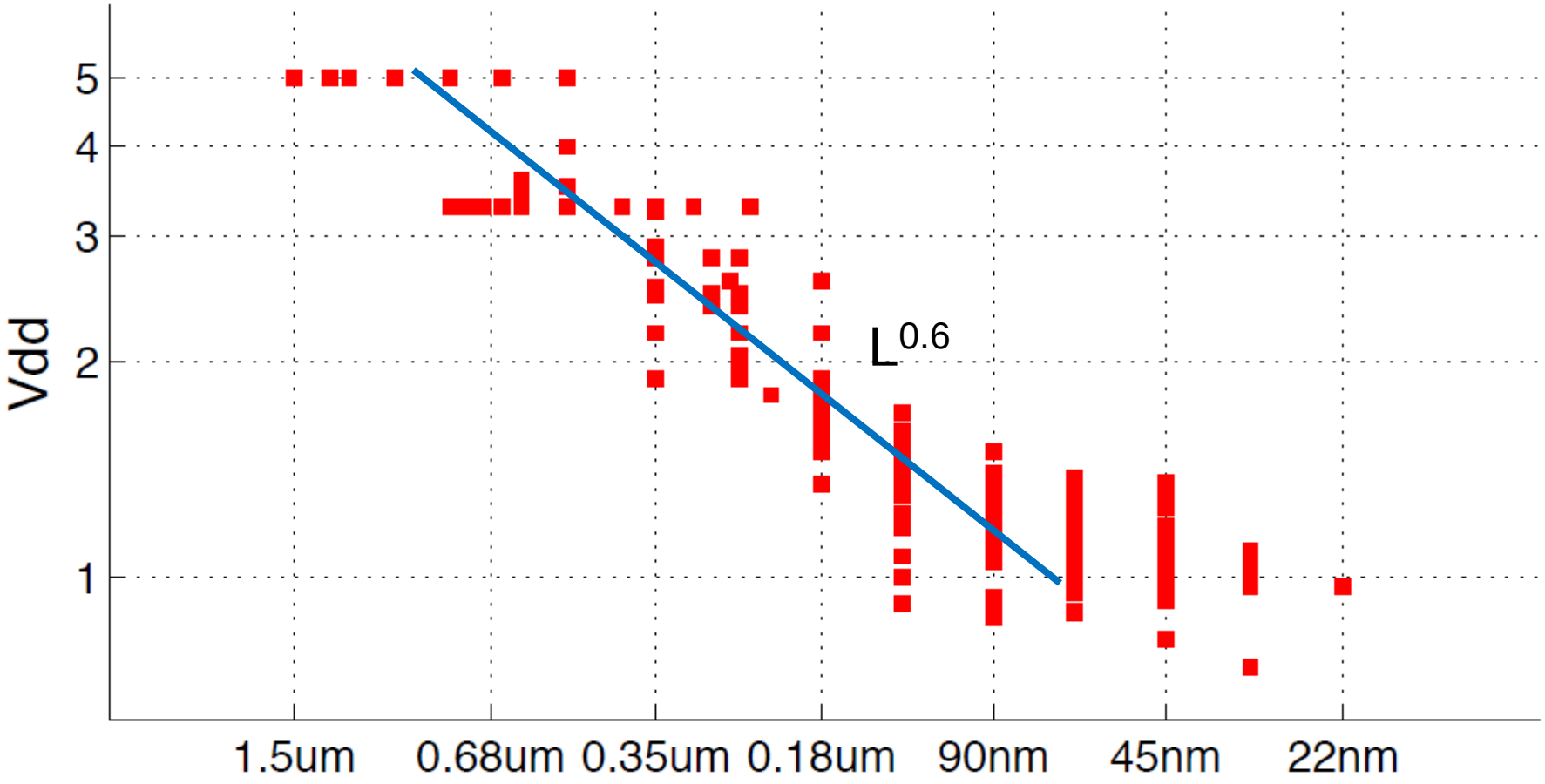

http://cpudb.stanford.edu/

# Power Increased Because We Were ~~Greedy~~ Clever




http://cpudb.stanford.edu/

# This Power Problem Is Not Going Away: $P = \alpha C * Vdd^2 * f$

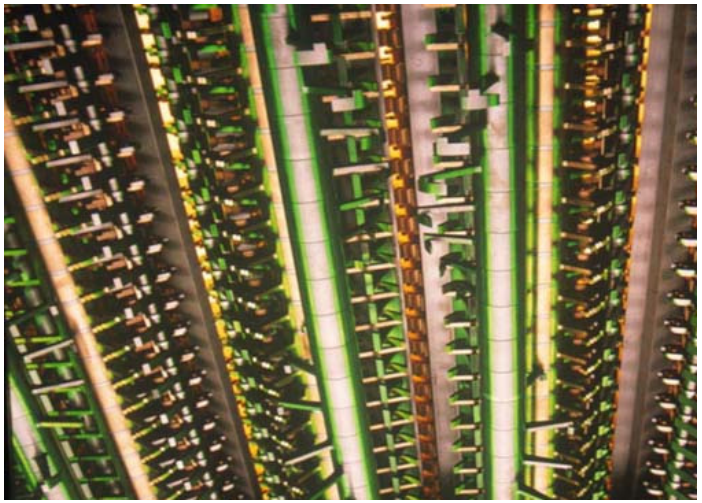


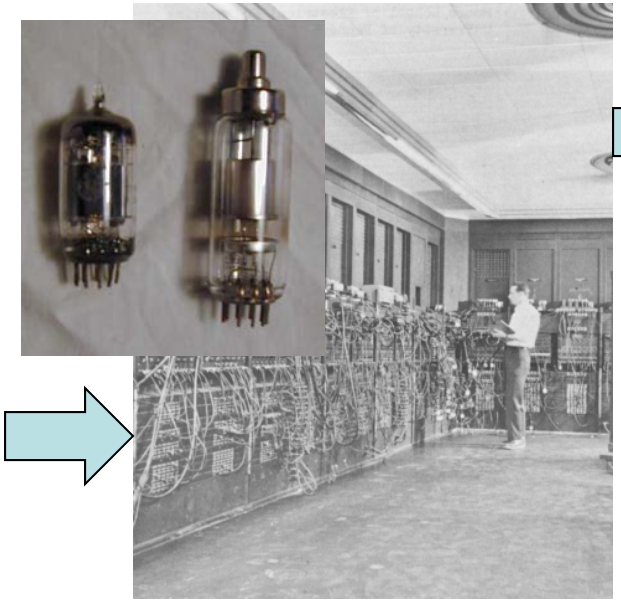

http://cpudb.stanford.edu/

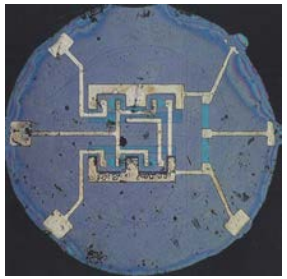
# Think About It

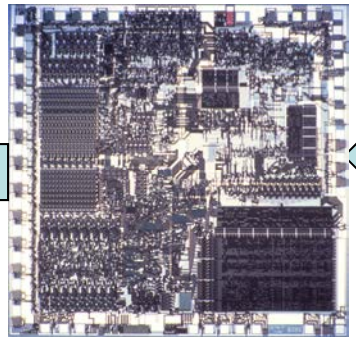
$$P = \frac{\text{ENERGY}}{\text{OP}} \quad \frac{\text{OPS}}{\text{S}}$$

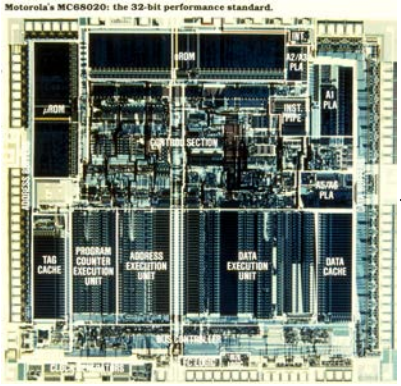
# Technology to the Rescue?







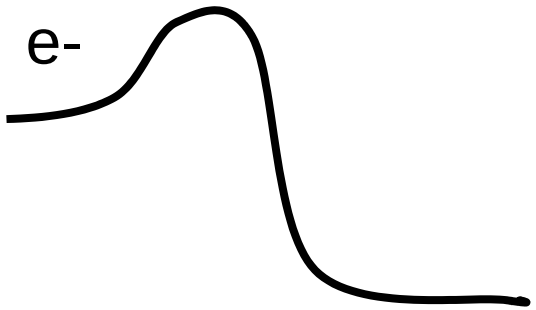





# Problems w/ Replacing CMOS

## Pretty fundamental physics

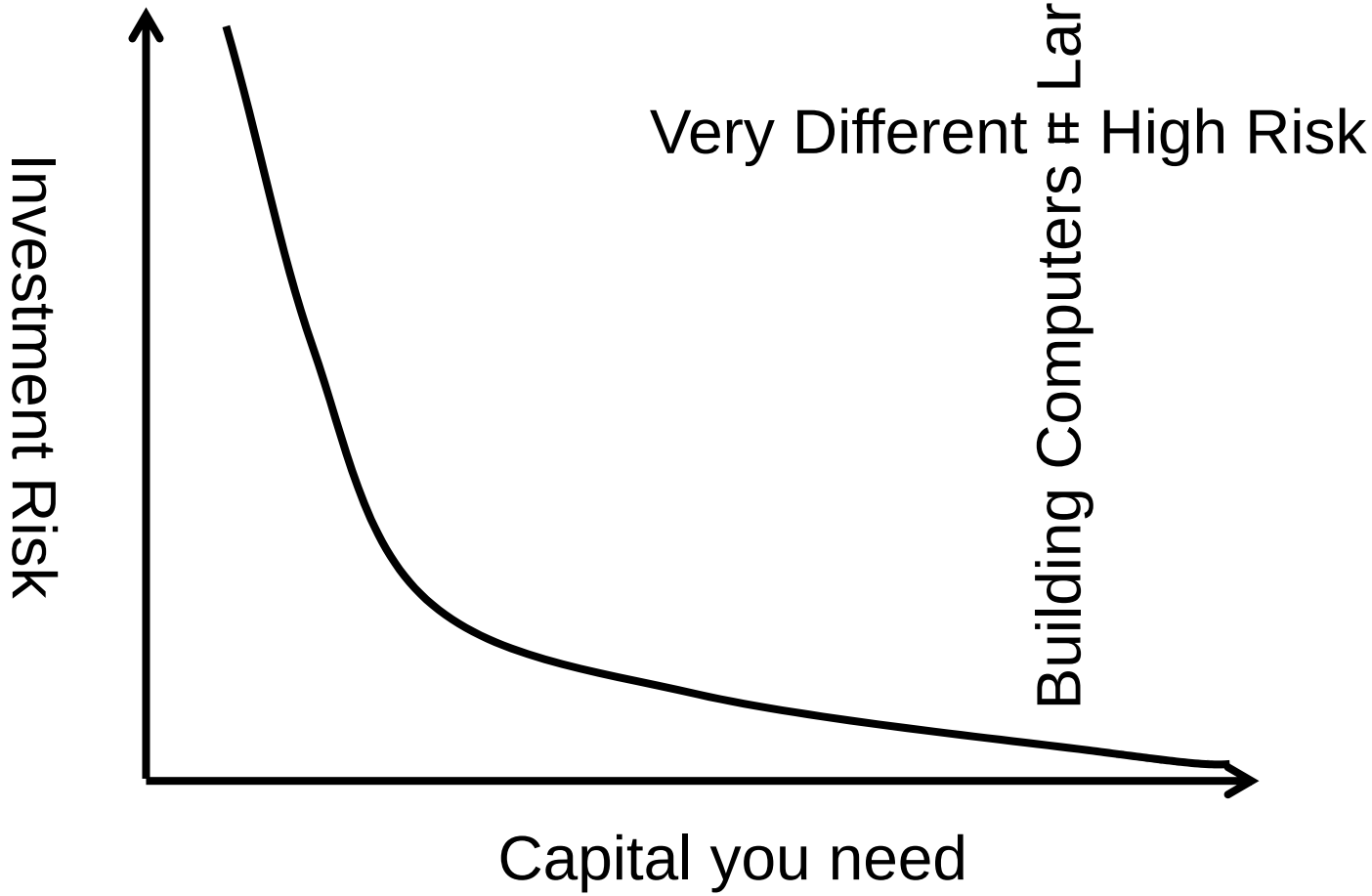
- Avoiding this problem will be hard




## Its capability is pretty amazing

- fJ/gate, 10ps delays, $10^9$ working devices

# Catch - 22



# The Truth About Innovation

# Start by creating new markets

# Our CMOS Future

## Will see tremendous innovative uses of computation

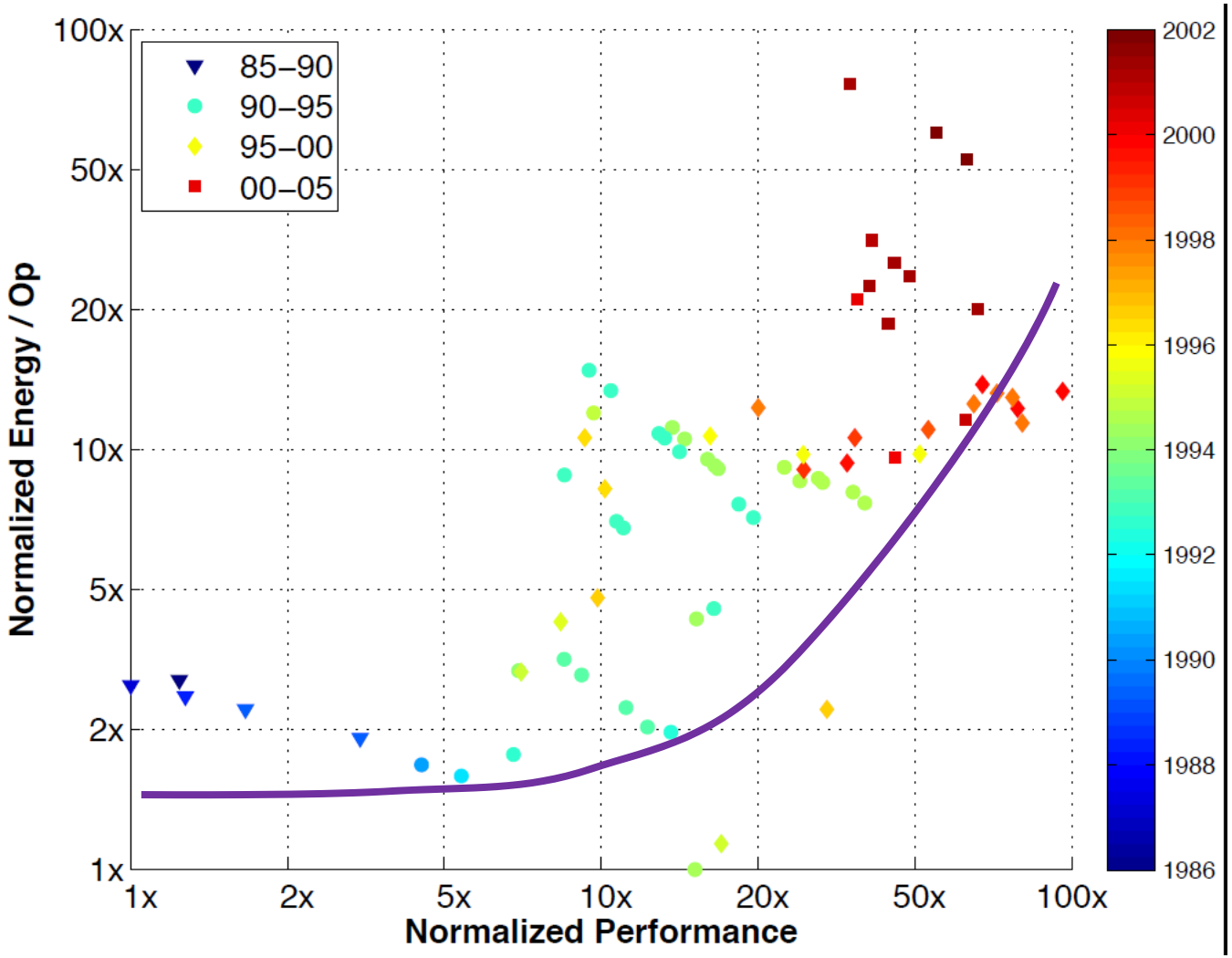
- Capability of today's technology is incredible
- Can add computing and communication for nearly $0
- Key questions are what problems need to be solved?

## Most performance system will be energy limited

- These systems will be optimized for energy efficiency

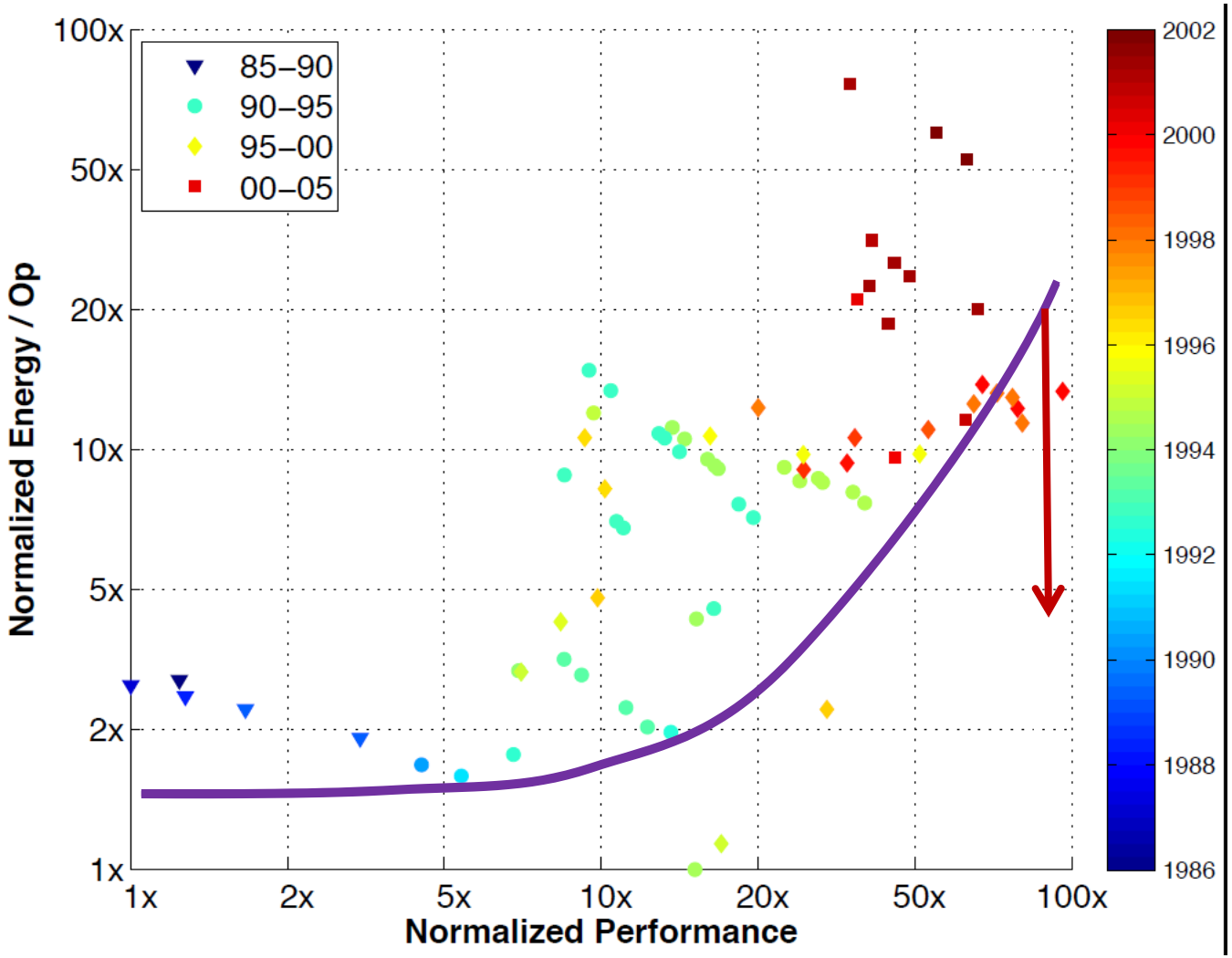
**Power = Energy/Op * Ops/sec**

# Processor Energy – Delay Trade-off



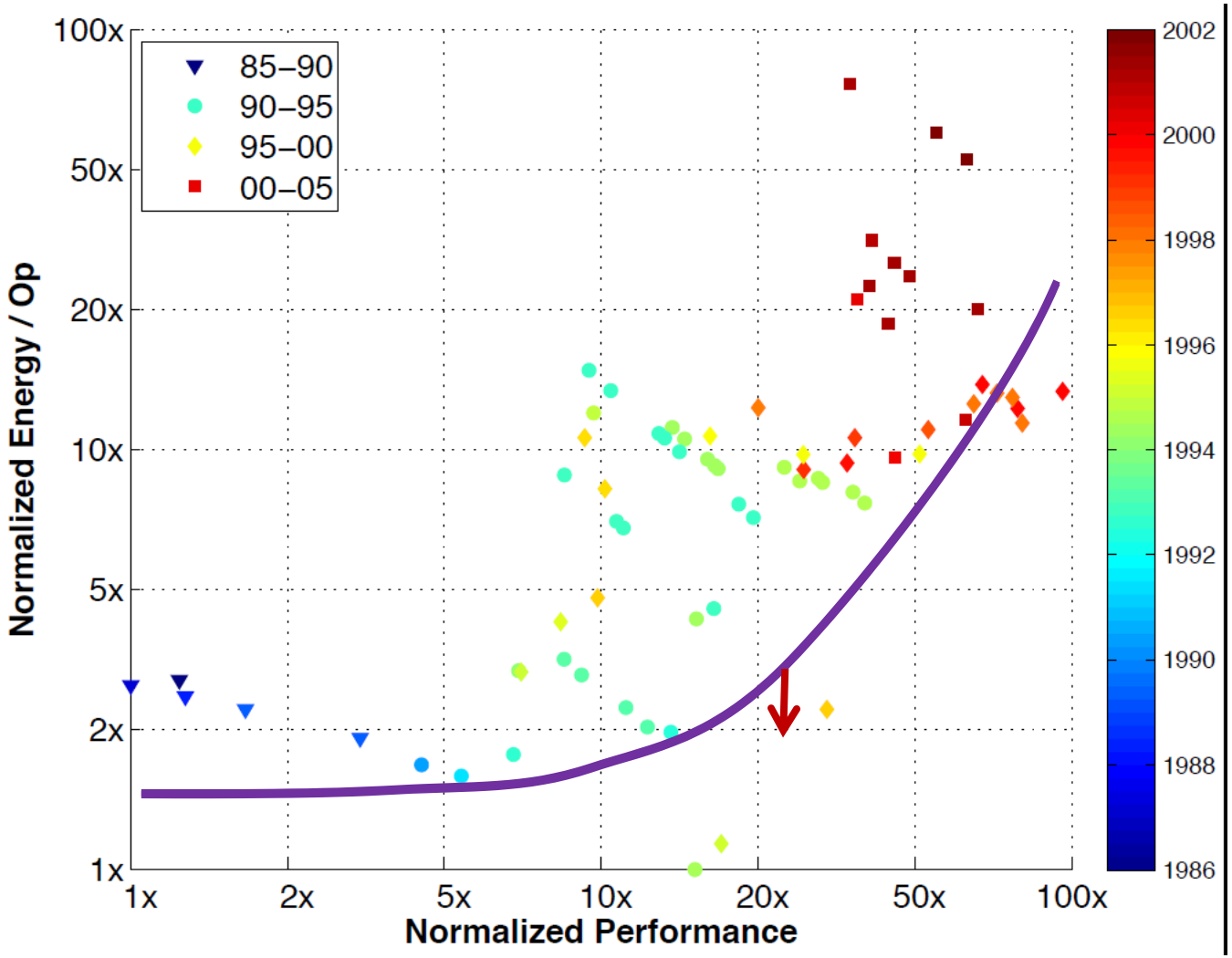

http://cpudb.stanford.edu/

# The Rise of Multi-Core Processors



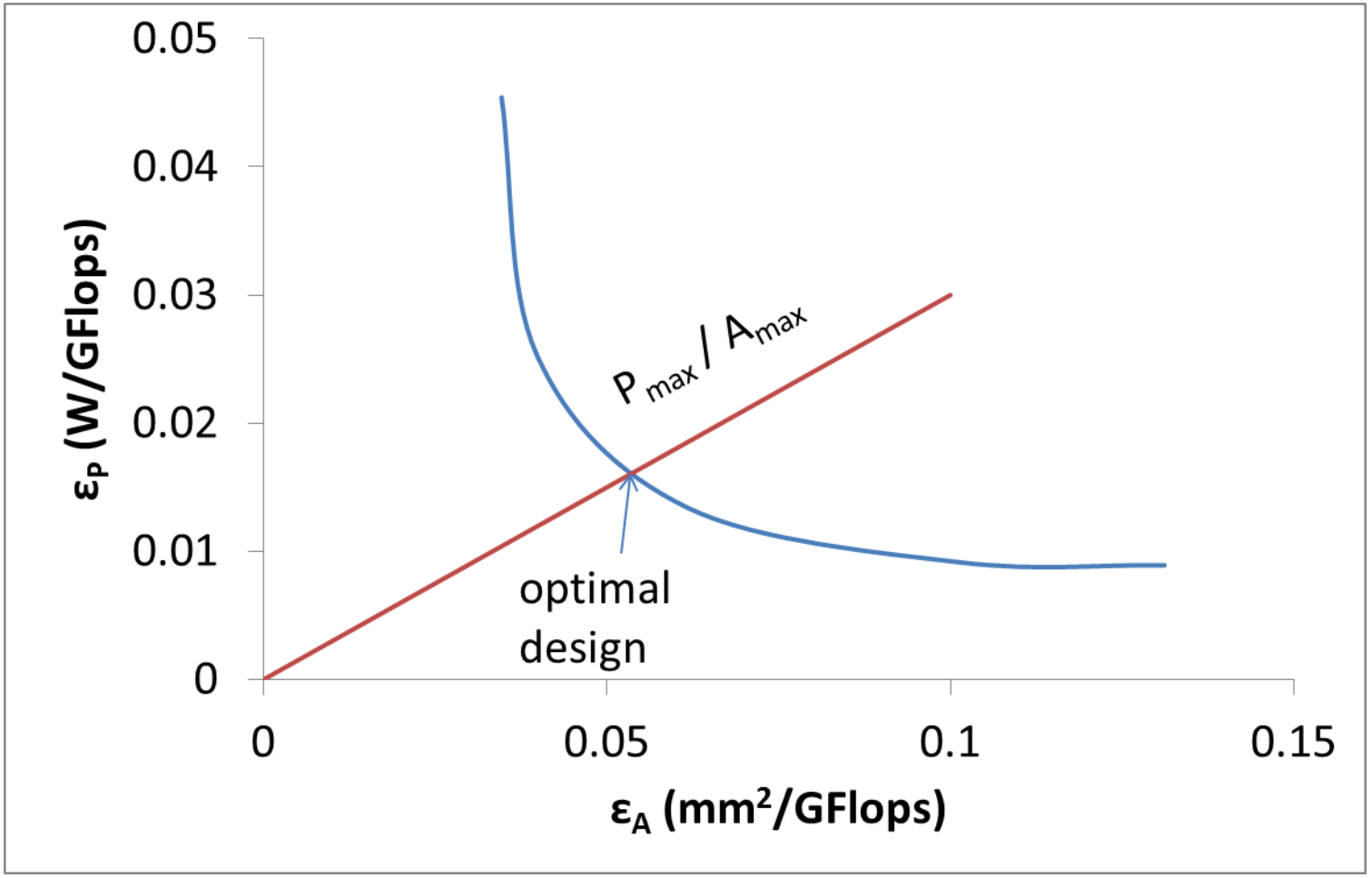

http://cpudb.stanford.edu/

# The Stagnation of Multi-Core Processors




http://cpudb.stanford.edu/

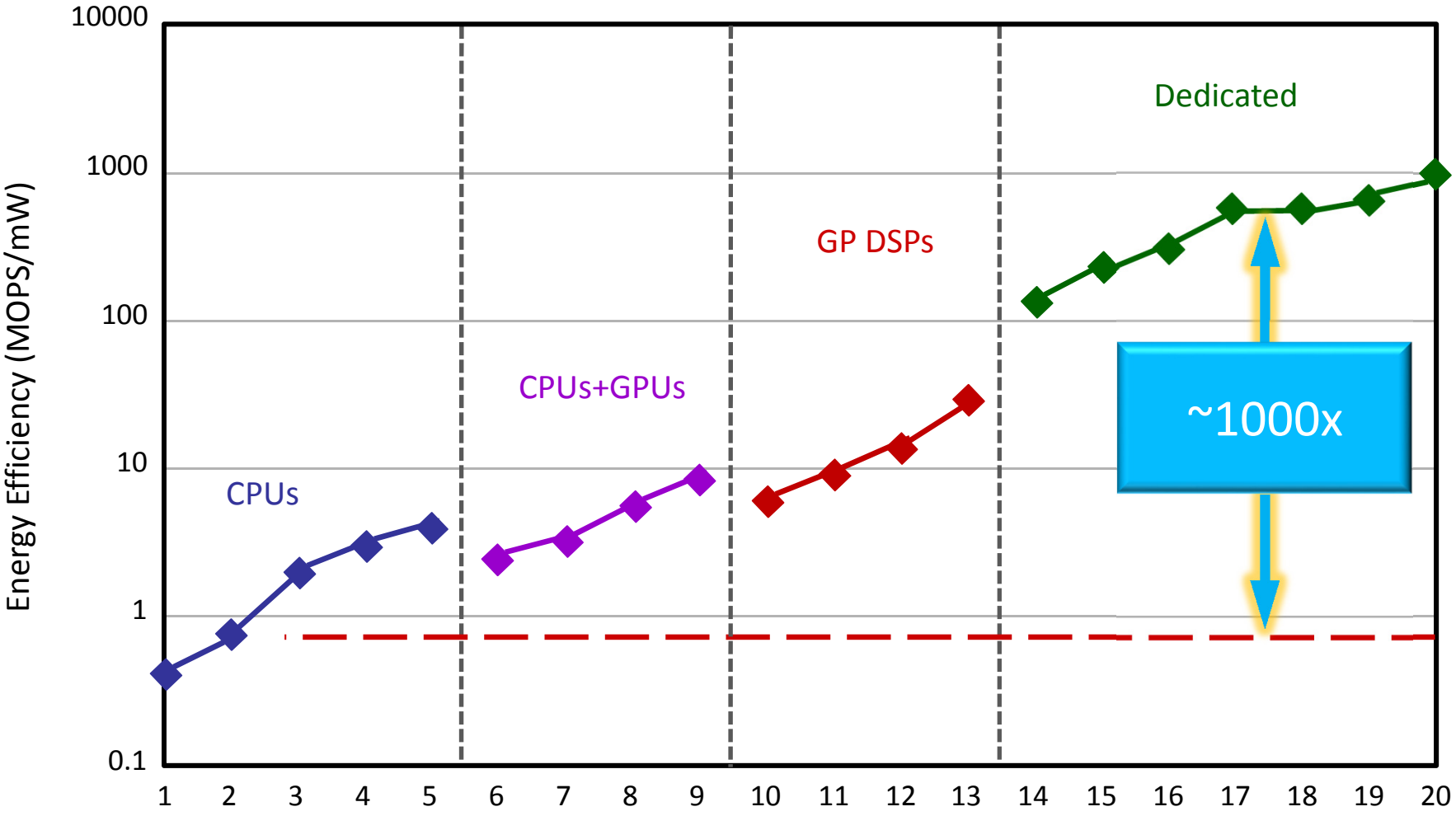
# Optimizing Parallel Machines (GPUs)




Galal et al, Trans on Computers, July 2011, pp 913,922

# Have A Shiny Ball, Now What?

# Signal Processing ASICs




Markovic, EE292 Class, Stanford, 2013

# The Push For Specialized Hardware

Dark Silicon and the End of Multi

Hadi Esmaeilzadeh† Emily Blem‡ Renée St. Amant§

†University of Washington ‡The University of Texas at

hadianeh@cs.washington.edu blem@cs.wisc.edu

ABSTRACT

Since 2005, processor designers have increase
ploit Moore's Law scaling, rather than focusing
formance. The failure of Dennard scaling, to which
ticore parts is partially a response, may soon limit mu
just as single-core scaling has been curtailed. This p
multicore scaling limits by combining device scaling,
scaling, and multicore scaling to measure the speedup pot
a set of parallel workloads for the next five technology gener
For device scaling, we use both the ITRS projections and
of more conservative device scaling parameters. To model sing
core scaling, we combine measurements from over 150 processor
to derive Pareto-optimal frontiers for area/performance and pow
er/performance. Finally, to model multicore scaling, we build a de
tailed performance model of upper-bound performance and lower-
bound core power. The multicore designs we study include single-
threaded CPU-like and massively threaded GPU-like multicore chip
organizations with symmetric, asymmetric, dynamic, and composed
topologies. The study shows that regardless of chip organization
and topology, multicore scaling is power limited to a degree not
widely appreciated by the computing community. Even at 22 nm
(just one year from now), 21% of a fixed-size chip must be powered
off, and at 8 nm, this number grows to more than 50%. Through
2024, only 7.9× average speedup is possible across commonly used
parallel workloads, leaving a nearly 24-fold gap from a target of
doubled performance per generation.

Categories and Subject Descriptors: C.0 [Computer Systems Or-
ganization] General — Modeling of computer architecture; C.0
[Computer Systems Organization] General — System architectures

General Terms: Design, Measurement, Performance

Keywords: Dark Silicon, Modeling, Power, Technology Scaling,
Multicore

Conservation Cores:
Reducing the Energy of Mature Computations

Ganesh Venkatesh Jack Sampson Nathan Goulding Saturnino Garcia
Vladyslav Bryksin Jose Lugo-Martinez Steven Swanson Michael Bedford Taylor

Department of Computer Science & Engineering
University of California, San Diego

{gvenkatesh,jsampson,ngouldin,sat,vbryksin,jlugomar,swanson,mbtaylor}@cs.ucsd.edu

Abstract

Growing transistor counts, limited power budgets, and the breakdown of voltage scaling are currently conspiring to create a *utilization wall* that limits the fraction of a chip that can run at full speed at one time. In this regime, specialized, energy-efficient processors can increase parallelism by reducing the per-computation power requirements and allowing more computations to execute under the same power budget. To pursue this goal, this paper introduces *conservation cores*. Conservation cores, or *c-cores*, are specialized processors that focus on reducing energy and energy-delay instead of increasing performance. This focus on energy makes c-cores an excellent match for many applications that would be poor candidates for hardware acceleration (e.g., irregular integer codes). We present a toolchain for automatically synthesizing c-cores from application source code and demonstrate that they can significantly reduce energy and energy-delay for a wide range of applications. The c-cores support patching, a form of targeted reconfigurability, that allows a move to new versions of the software they target. Our results show that conservation cores can reduce energy consumption by 11× and by up to 2.1× for whole applications and the useful lifetime of individual processors.

power. Consequently, the rate at which we can switch transistors is far outpacing our ability to dissipate the heat created by those transistors.

The result is a technology-imposed *utilization wall* that limits the fraction of the chip we can use at full speed at one time. Our experiments with a 45 nm TSMC process show that we can switch less than 7% of a 300mm² die at full frequency within an 80W power budget. ITRS roadmap projections and CMOS scaling theory suggests that this percentage will decrease to less than 3.5% in 32 nm, and will continue to decrease by almost half with each process generation—and even further with 3-D integration.

The effects of the utilization wall are already indirectly apparent in modern processors. Intel's Nehalem provides a "turbo mode" that powers off some cores in order to run others at higher speeds. Another strong indication is that even though native transistor switching speeds have continued to double every two process generations, processor frequencies have not increased substantially over the last 5 years.

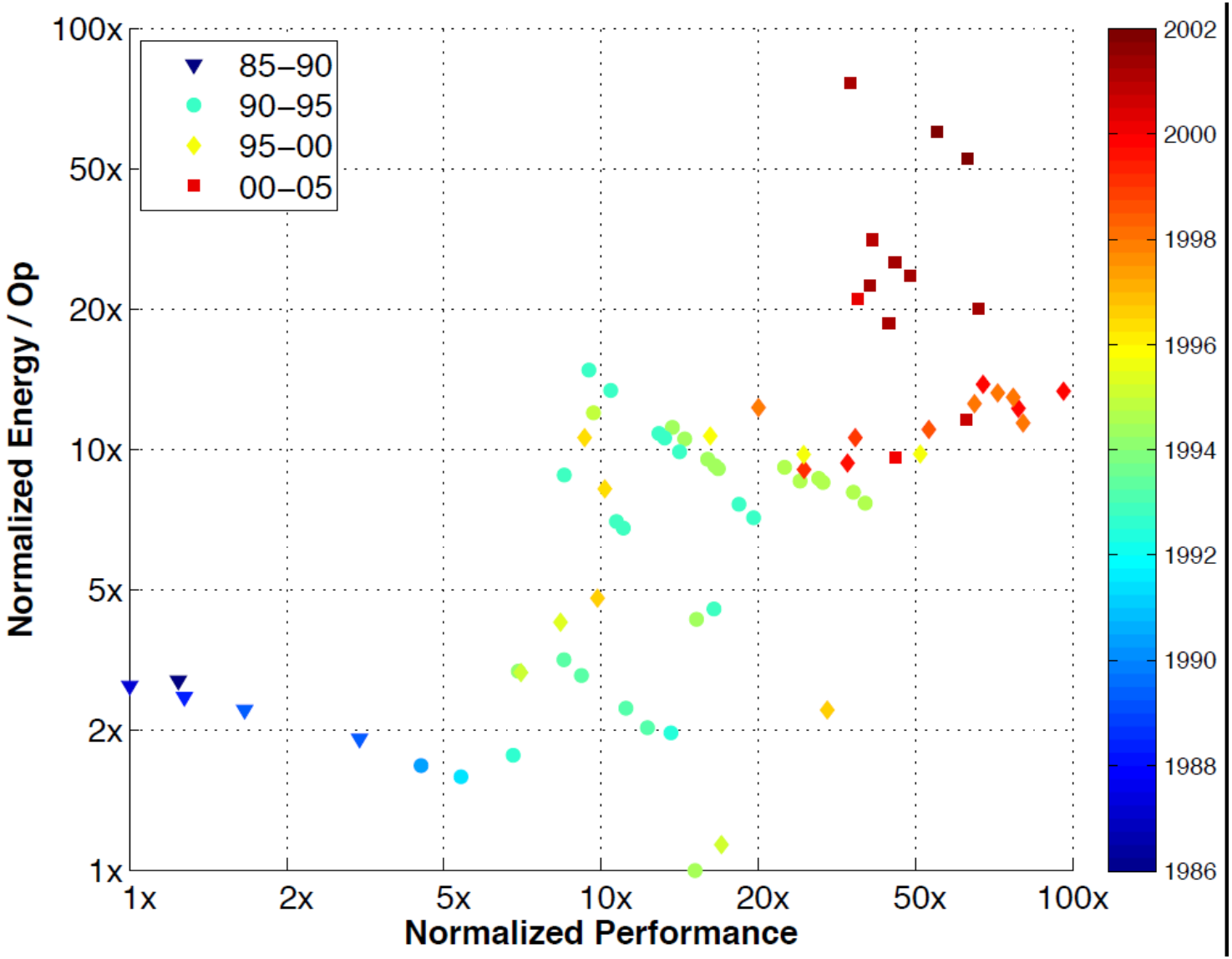
In this regime, reducing per-operation energy [19] translates directly into increased potential parallelism for the system: If given computation can be made to consume less power at the same level of performance, other computations can be run in parallel without violating the power budget.

This paper attacks the utilization wall with *conservation cores*. Conservation cores, or *c-cores*, are application-specific hardware circuits created for the purpose of reducing energy consumption in computationally-intensive applications. Since it is no longer possible to run the entire chip at full frequency at once, it makes sense to optimize the portions of the chip that are running so that it is possible for the application at hand. In effect, architects to trade area for energy in order to mitigate the utilization wall has made this trade-off possible. Continued increases in transistor

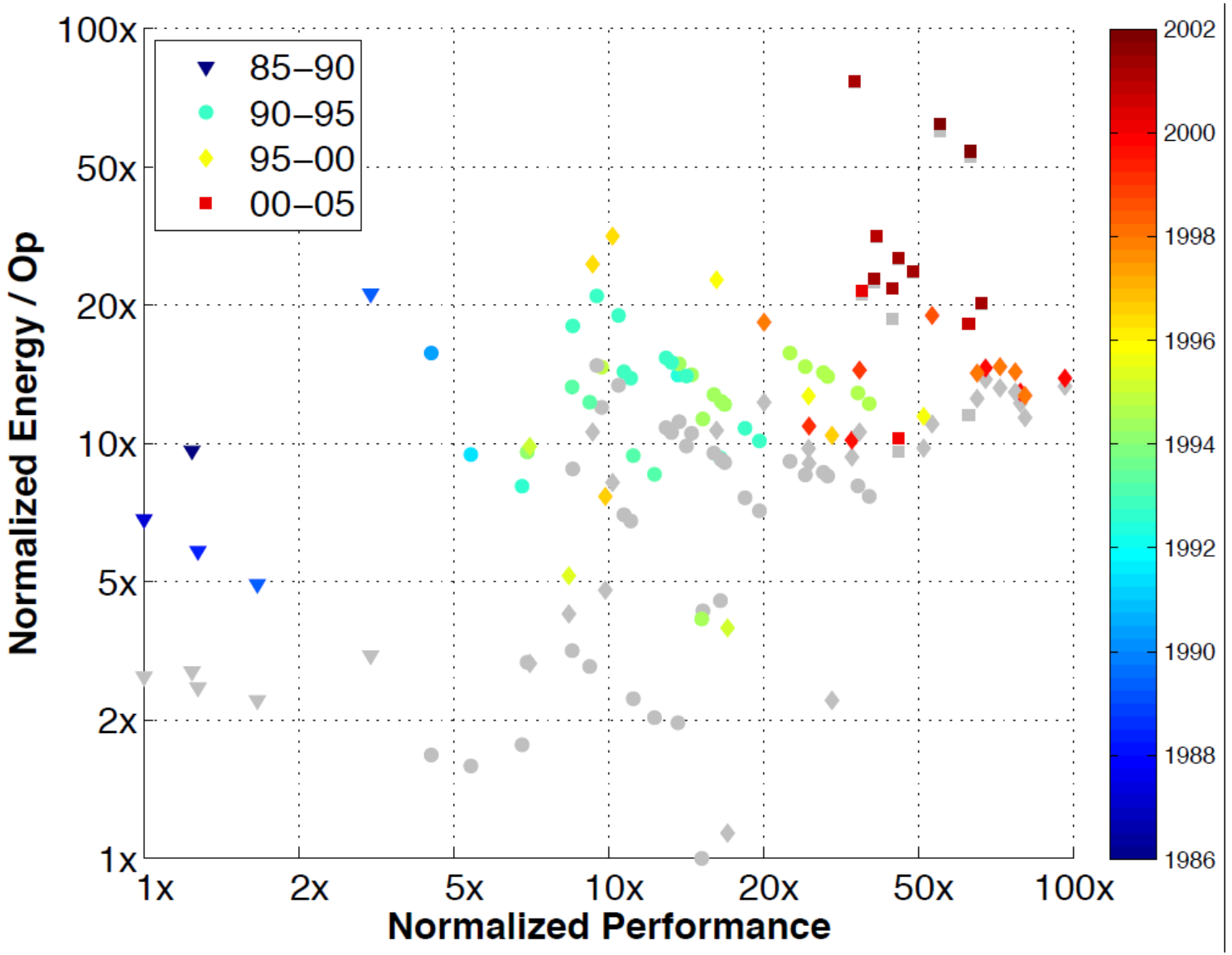
# Before Talking About Specialization

**WE SHOULD CHECK ONE MORE THING FIRST**

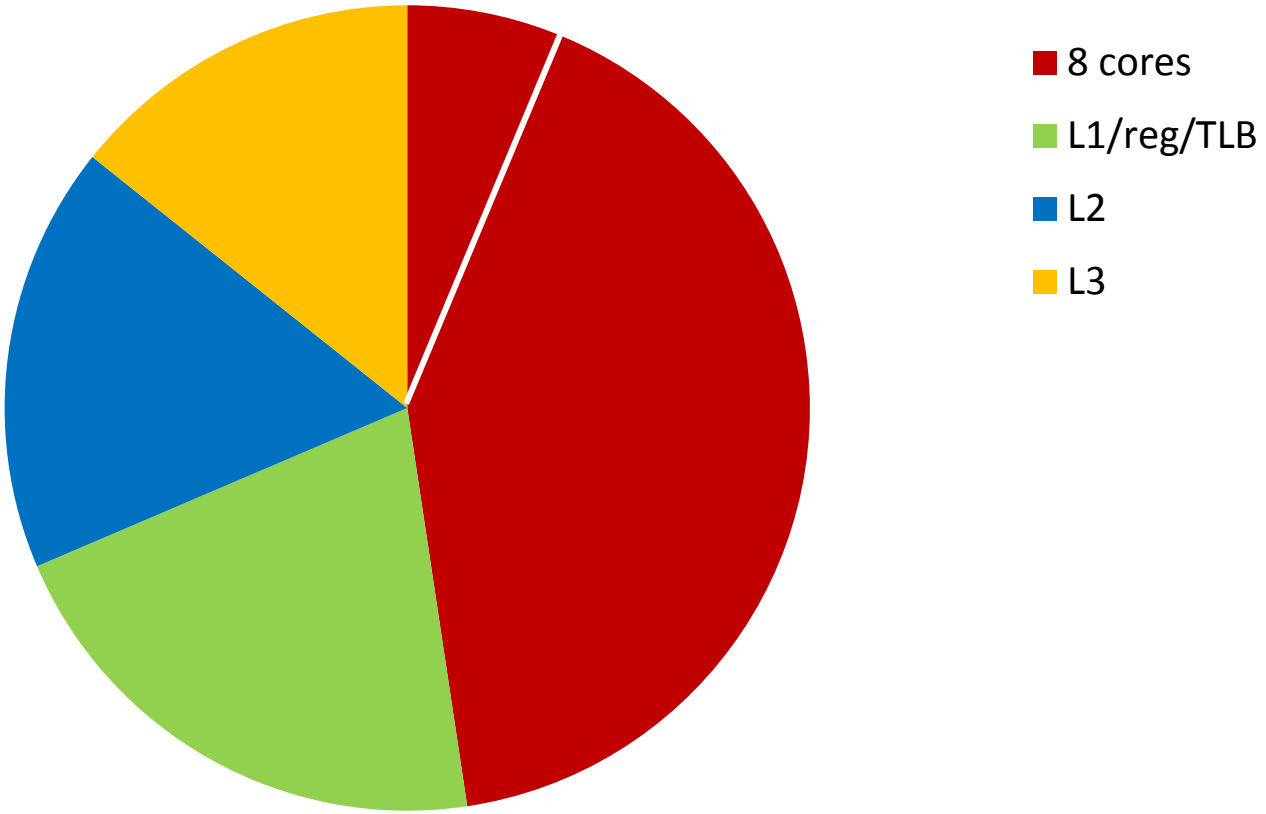
# Don’t Forget Memory System Energy



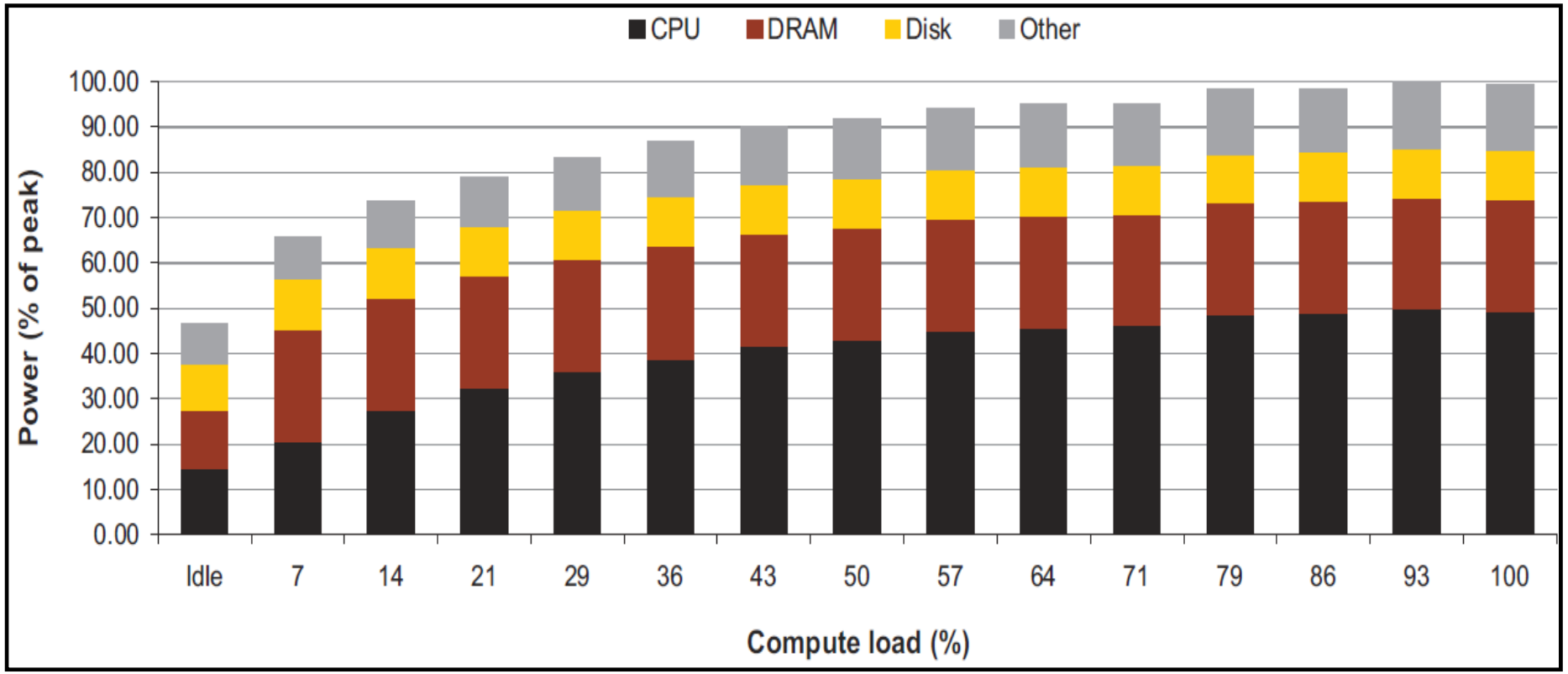
# Processor Energy w/ Corrected Cache Sizes



# Processor Energy Breakdown



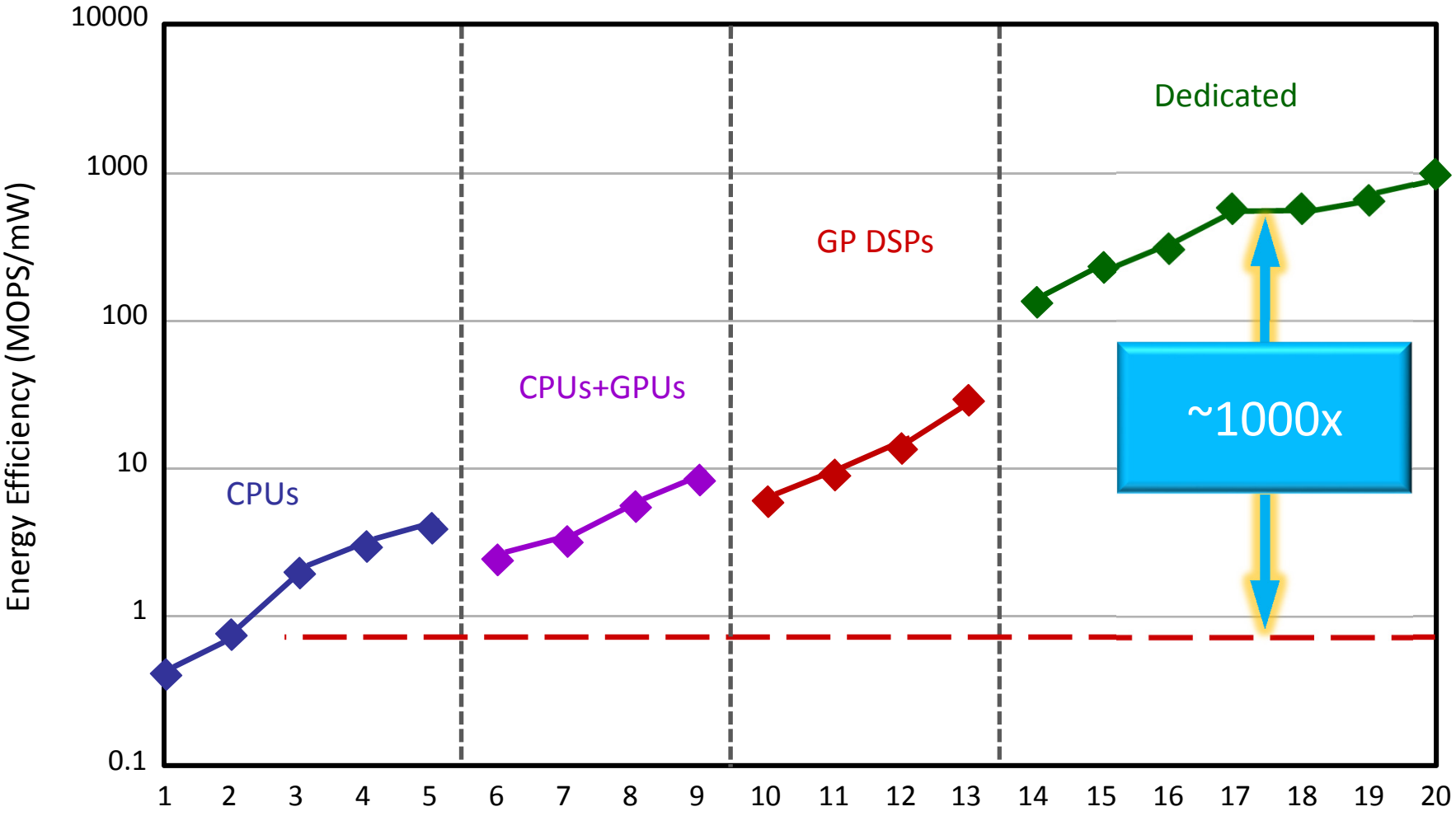
# Data Center Energy Specs




Malladi, ISCA, 2012

# SO HOW WILL ACCELERATORS HELP?

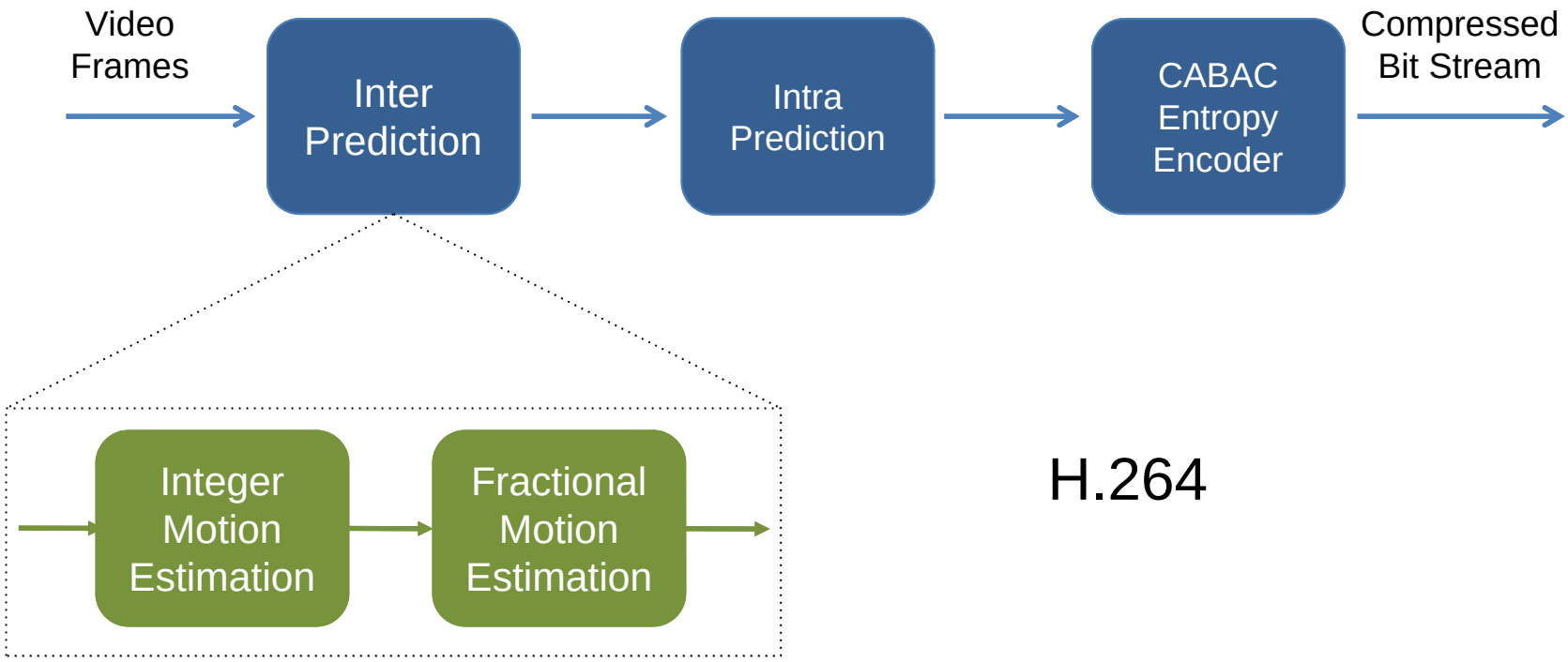
# What Is Going On Here?



# ASIC's Dirty Little Secret

## All the ASIC applications have absurd locality

- And work on short integer data




H.264

90% of Execution time is here

Hamid et al, ISCA, 2010

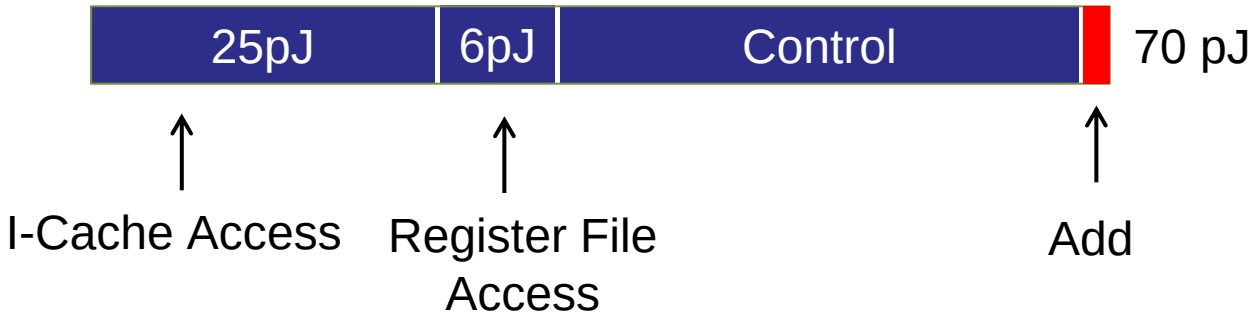
# Rough Energy Numbers (45nm)

| Integer | |
|---|---|
| Add | |
| 8 bit | 0.03pJ |
| 32 bit | 0.1pJ |
| Mult | |
| 8 bit | 0.2pJ |
| 32 bit | 3 pJ |

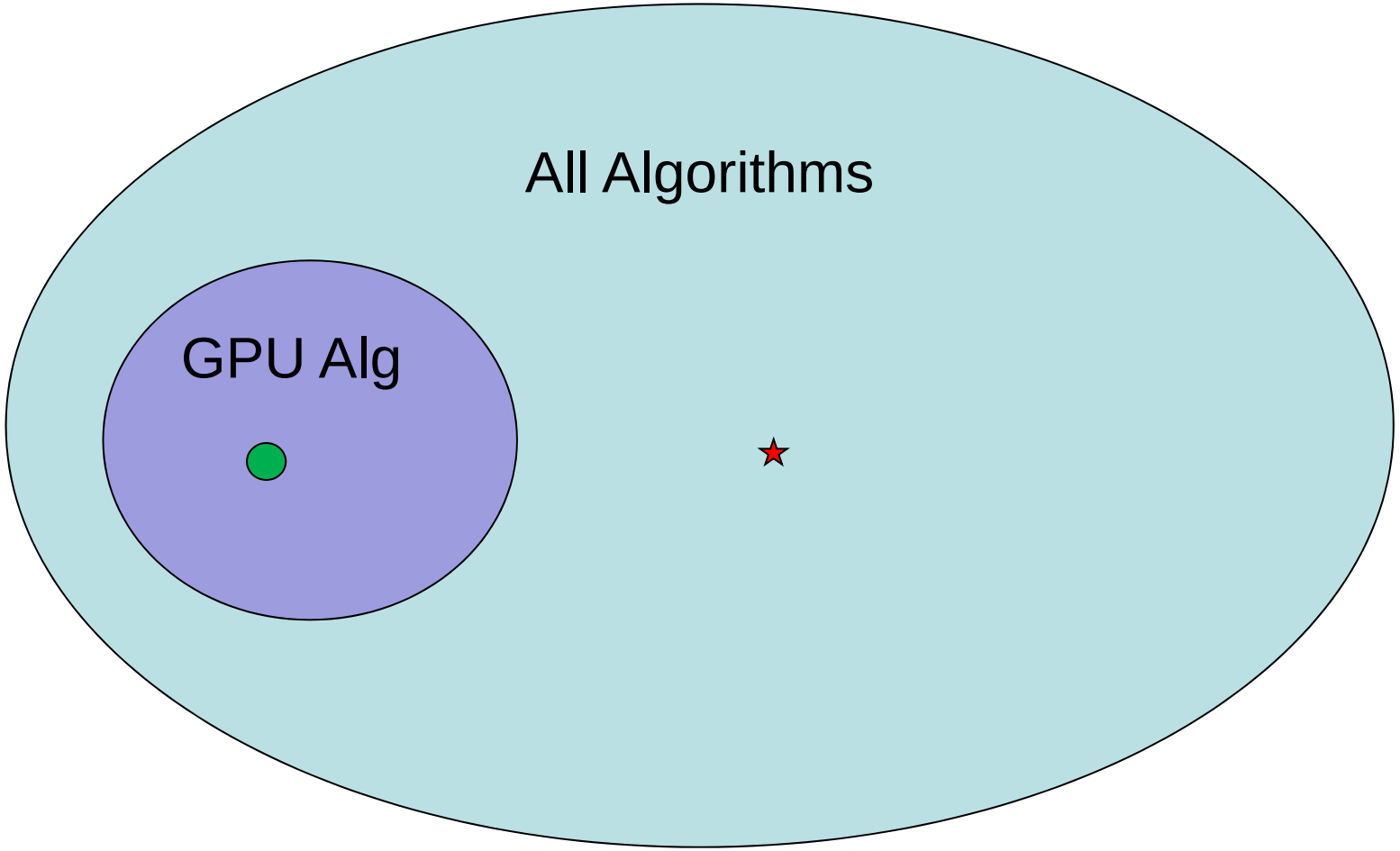
| FP | |
|---|---|
| FAdd | |
| 16 bit | 0.4pJ |
| 32 bit | 0.9pJ |
| FMult | |
| 16 bit | 1pJ |
| 32 bit | 4pJ |

| Memory | |
|---|---|
| Cache | (64bit) |
| 8KB | 10pJ |
| 32KB | 20pJ |
| 1MB | 100pJ |
| DRAM | 1.3-2.6nJ |

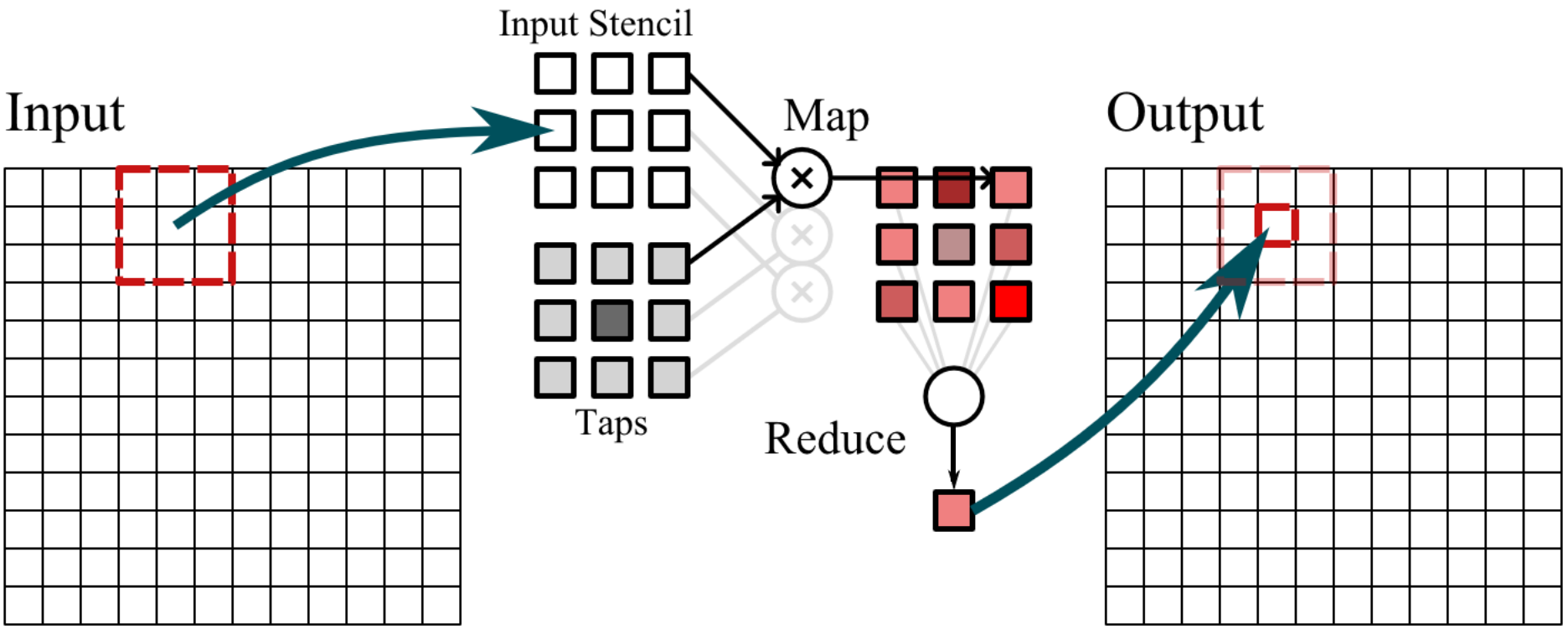
## Instruction Energy Breakdown



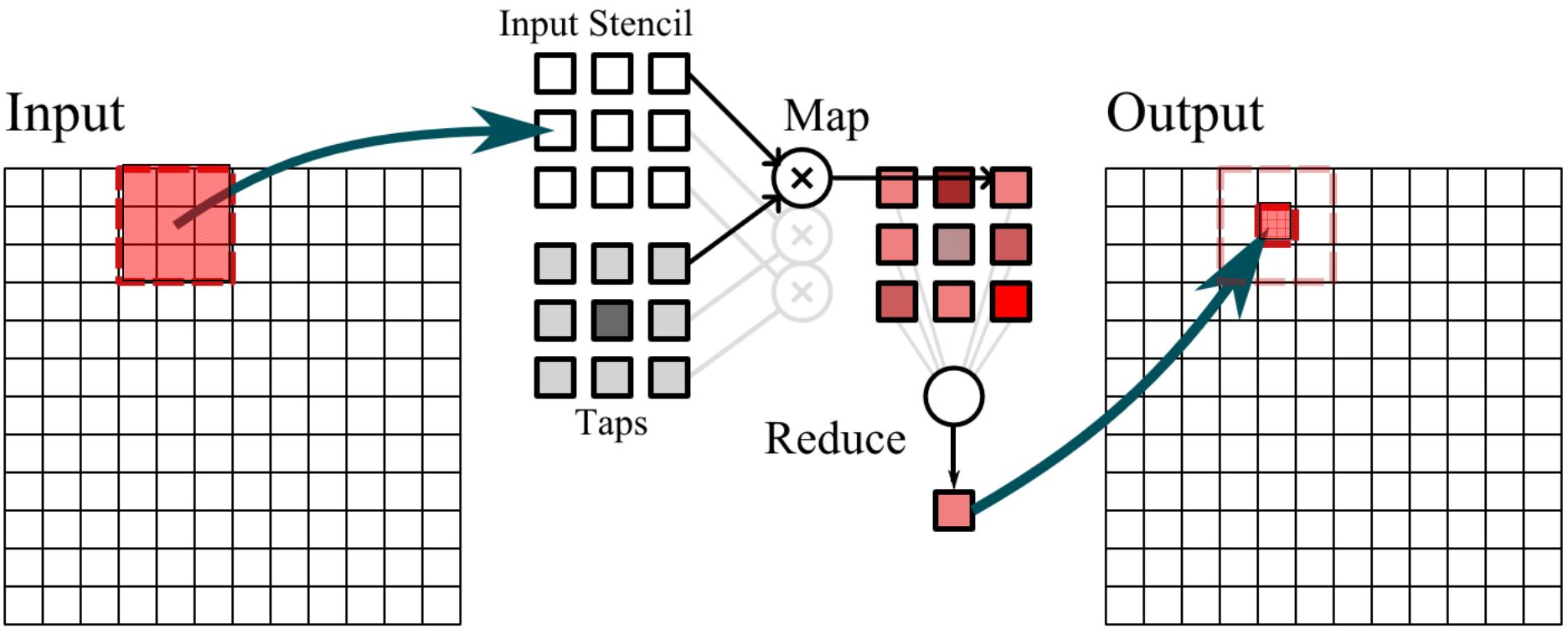
# The Truth:
## It's More About the Algorithm then the Hardware



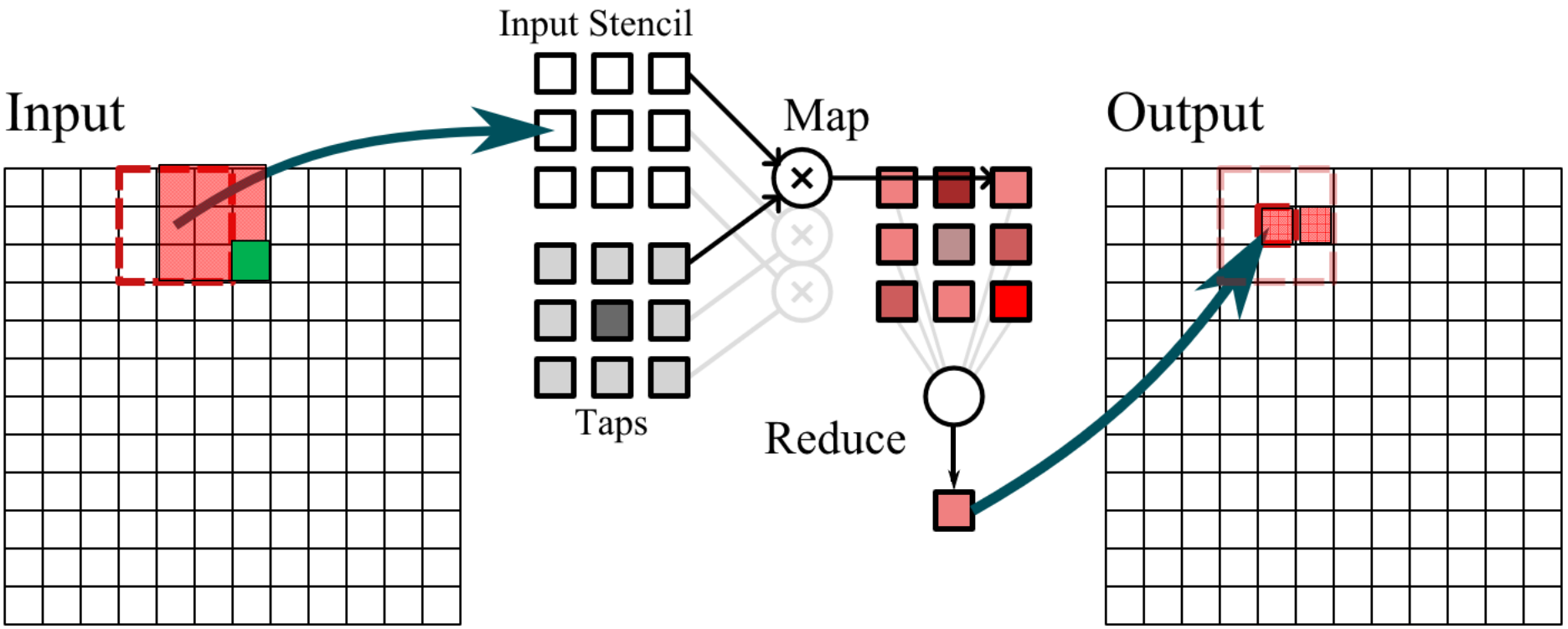
# Highly Local Computation Model



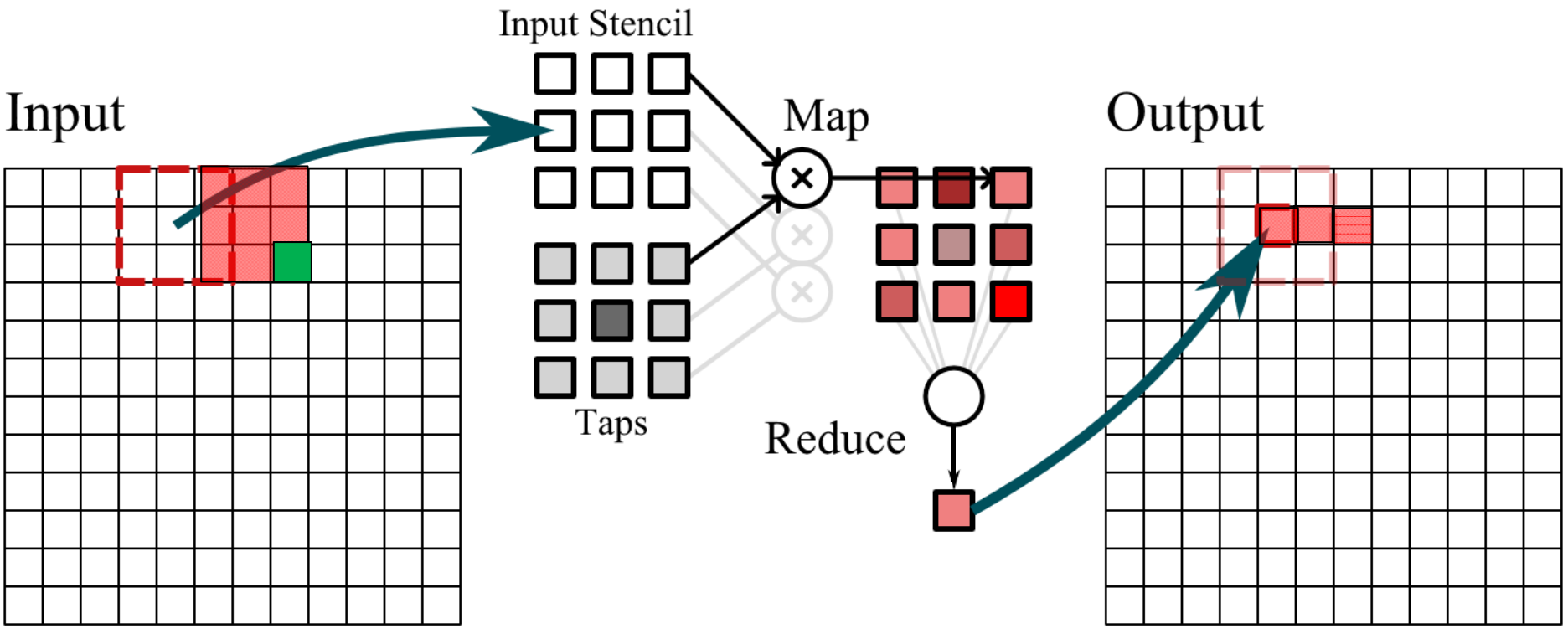
# Highly Local Computation Model



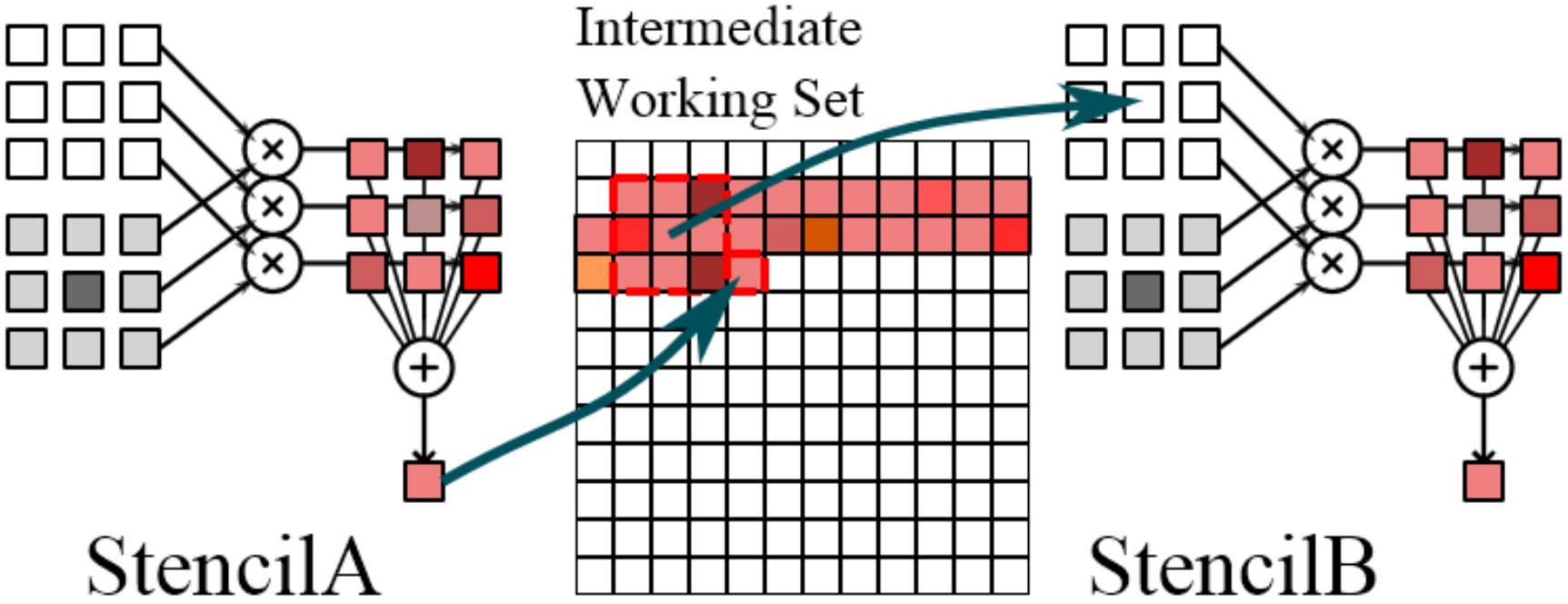
# Highly Local Computation Model



# Highly Local Computation Model



# Compose These Cores into a Pipeline




**Program in space, not time**

- Makes building programmable hardware more difficult

# Working on System to Explore This Space

## Takes high-level program

- Graph of stencil kernels

## Maps to hardware level assembly

- Compute graph of operations for each kernel

## Currently we map the result to:

- FPGA, custom ASIC

# Enabling Innovation

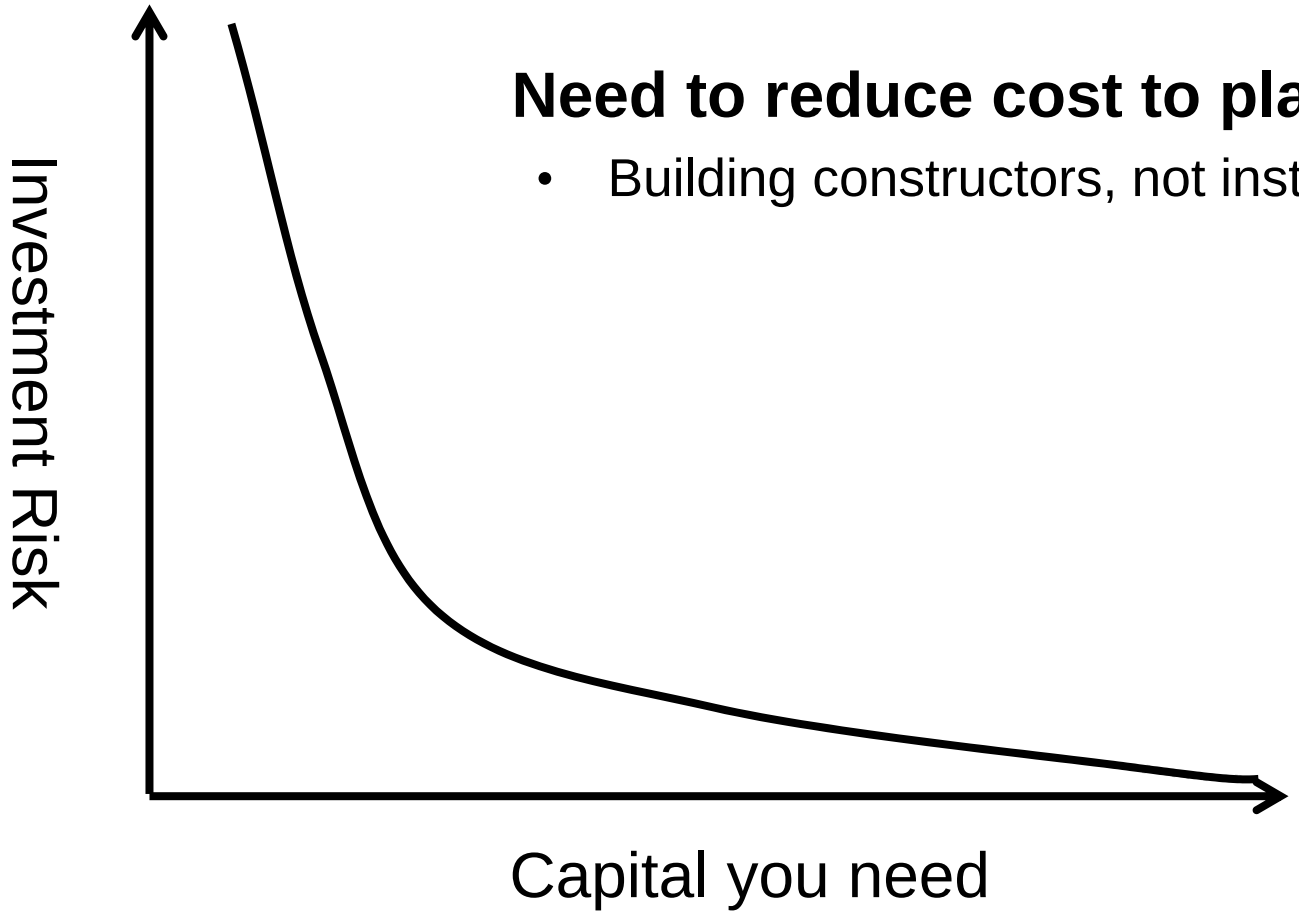
## You don’t just compile applications to efficiency

- Need to tweak the application to fit constraints

## Need to enable application experts to play

- They know how to “cheat” and still get good results

# Remember This Trade-off?



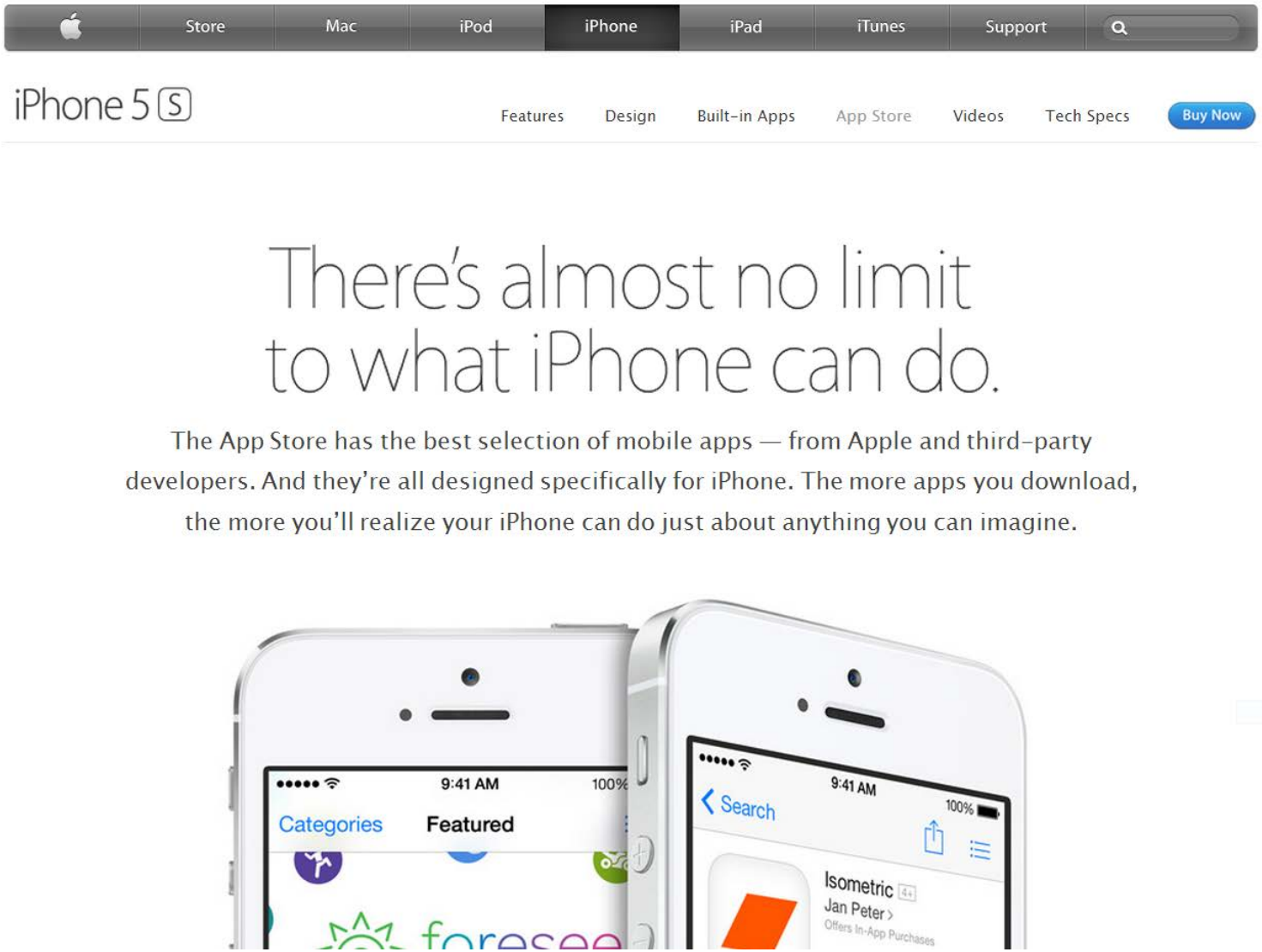

**Need to reduce cost to play**

- Building constructors, not instances

http://genesis2.stanford.edu/

# Not All Systems Are On The Bleeding Edge

# App Store For Hardware



# Challenge


## What Arduino can do

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment

## Community

The community of Arduino enthusiasts is vast, and includes region specific groups and special interest groups. The community is an excellent further source of assistance on all topics such as accessory selection, project assistance, and ideas of all sorts.

# A New Hope

**If technology is scaling more slowly**

- We can incorporate current design knowledge into tools
- To create extensible system constructors

**If killer products are going to be application driven**

- Application experts need to design them

**We can leverage the 1st bullet to enable the 2nd**

- To usher in a new wave of innovative computing products