

# Sutro Meeting #11

agenda

## Schedule

Energy-Efficient Learning

18:00, 11 May 2026, 380 Brannan

Agenda:

- introductions
- Yaroslav/Yad/Armin
- pizza

wifi: southparkcommons/whokilledkenny

## Motivation

Personal motivation: I think it would just be cool to rewrite all AI from scratch. I had a [20 year plan](#), but now that we have agents, I'm feeling more ambitious. And to do it in a maximally collaborative way, because it's more fun.

And to involve more people it needs to 1) have real world impact 2) have measureable sense of progress.

This is summarized in [Yaroslav's sutro planning sprint #1](#)

This is "week 2", by week 52, It would be nice to have something like nano GPT but designed from-scratch for existing hardware.

Why would this look different from existing approaches?

Existing methods were created at an era of different hardware

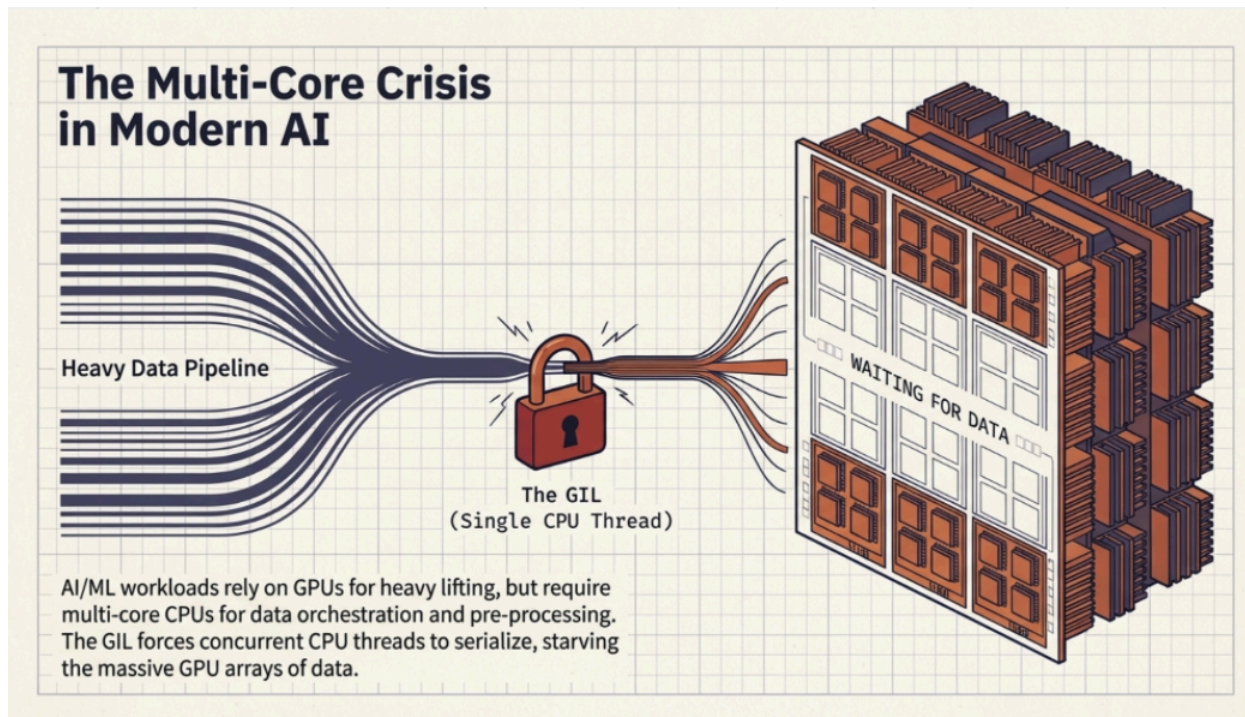
Gradient descent (1847), perceptrons (1958), backprop (1970/1986)

(chatgpt [timeline](#))

Software lags on hardware hardware change.

Examples:

## 1. Python GIL



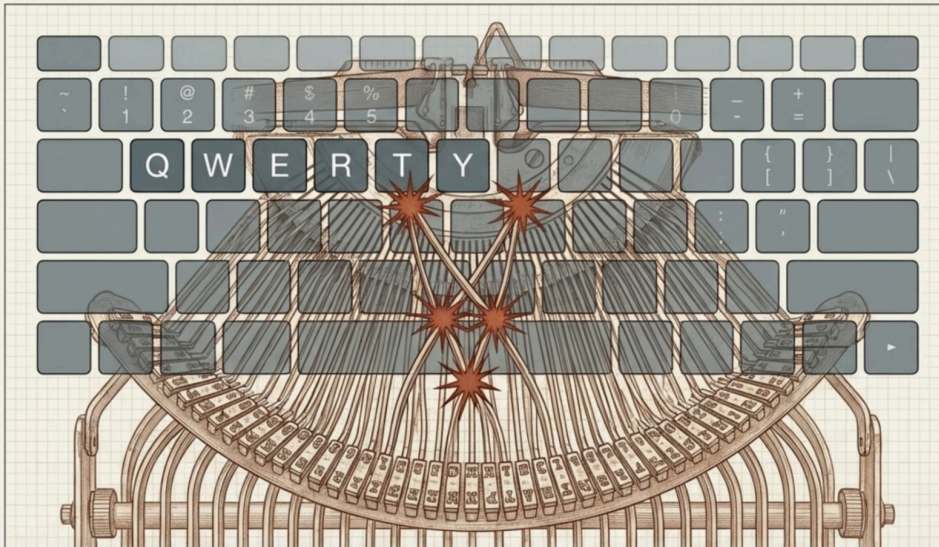
## 2. HDD hangover

B-Trees are a good feed to HDDs

After HDDs changed to SSD, spent 15 years tuning B-Trees until major rewrite (LST)

## 3. QWERTY

## Designing for Slower Speeds



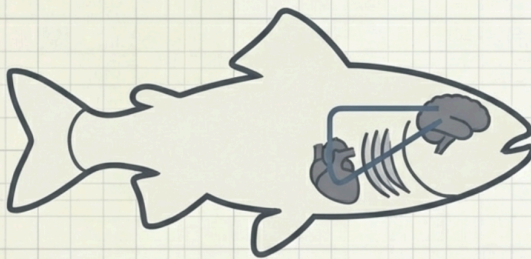
In the 1870s, typing speed was the enemy of the machine. Fast typists caused mechanical strike arms to collide and jam.

The QWERTY layout was engineered as an intentional physical bottleneck. It deliberately spread out common letter combinations to slow typists down and keep the machine chugging along.

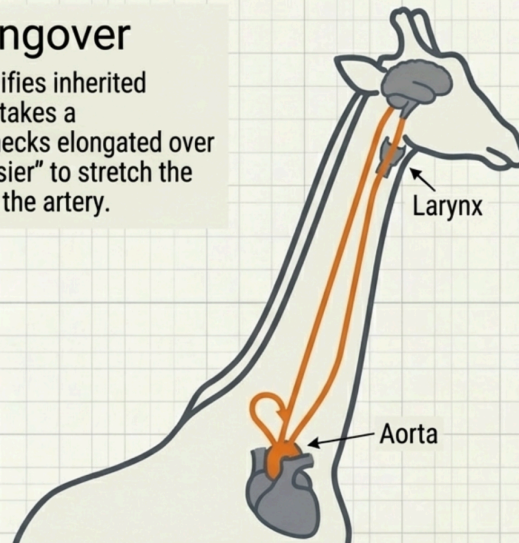
### 4. Laryngeal nerve

#### Biology: The Evolutionary Hangover

Natural selection cannot design from scratch; it modifies inherited structures. The recurrent laryngeal nerve in giraffes takes a preposterous 15-foot detour because, as vertebrate necks elongated over millions of years, incremental mutations made it “easier” to stretch the nerve than to structurally sever and reroute it across the artery.



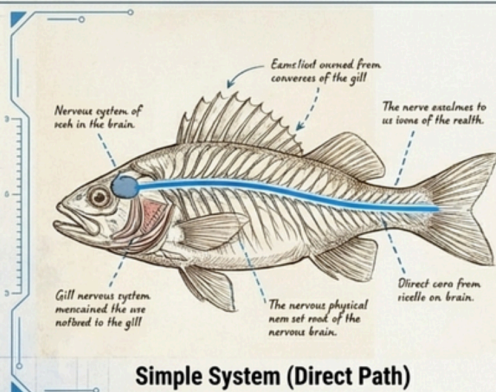
**Fish:** Direct neural path.



**Giraffe:** 15-foot path-dependent detour.

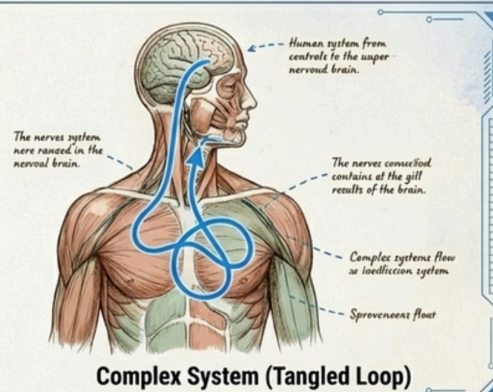


## Biological Tech Debt



### The Premise:

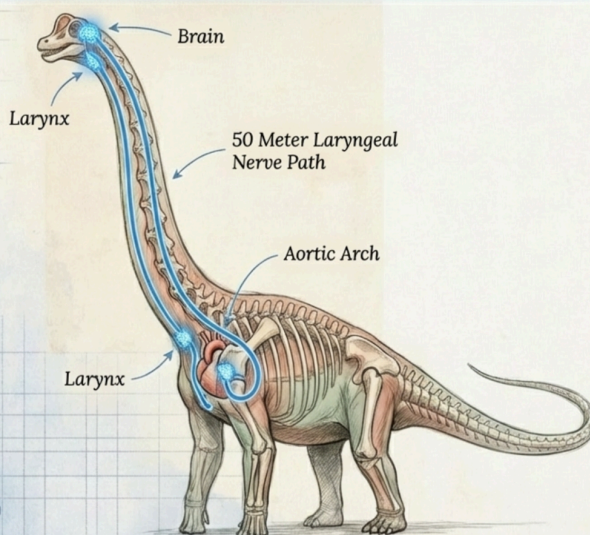
When complex systems evolve from simple ones without a top-down redesign, early bugs become permanent architectural constraints.



### The Context:

A biology lecture delivered to crypto-engineers to explain systemic bottlenecks and network inefficiencies.

## The Ultimate Bottleneck: The Sauropod Nerve



### The Flaw

The longest single cell in the history of life.

### The Origin

In early fish, this nerve ran a few inches straight from the brain to the gills. As necks lengthened over millions of years, the nerve was already trapped under the heart's aortic arch.

### The Result

50 meters of wasted distance to travel a few inches of actual space. Inefficiency by design.

Hardware has changed

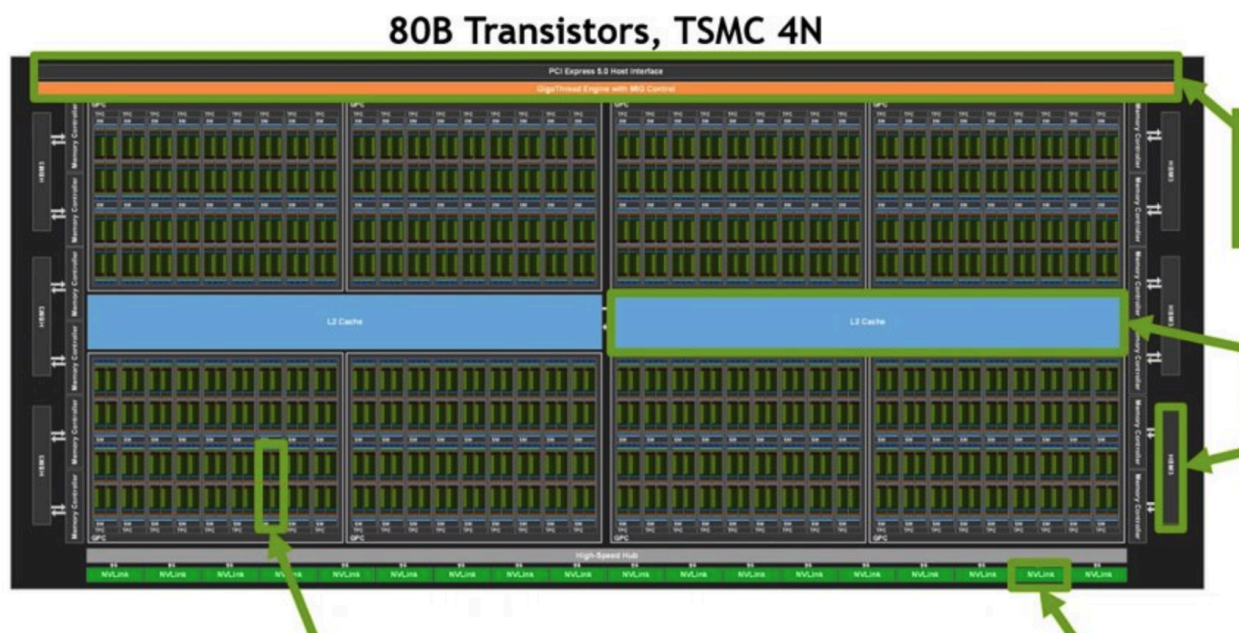


multicore

**The single-core lock-in:** Gradient descent (1847), perceptrons (1958), backprop (1970/1986), and LSTMs (1997) were all invented before IBM released the first true multicore processor in 2001, with consumer dual-core arriving in 2005. The forward→backward→update pipeline is a serial dependency chain that acts exactly like a GIL.

We ported our CPU algorithms by assuming GPU is a big CPU.  
16 bit→32 bit→512 bit etc

Modern Processors can execute different instructions.



Mega-kernels [chatgpt \(public\)](#), [claude \(public\)](#) [gemini \(public\)](#)

<https://hazyresearch.stanford.edu/blog/2025-05-27-no-bubbles>

memory cost

Memory used to be plentiful, now it is scarce

([claude](#))

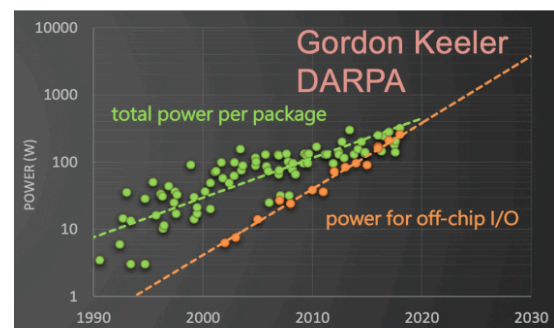
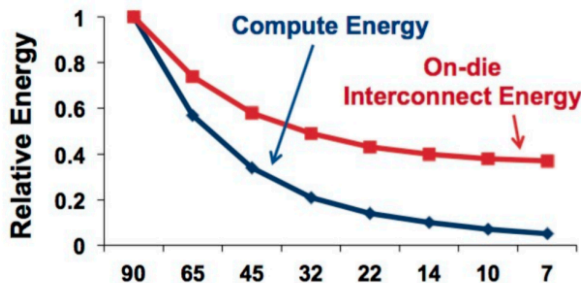
**The memory blindness:** The memory wall was identified in 1995 — CPU performance doubled every 18 months but DRAM bandwidth improved only 7–10% annually ([source](#)).

Chen Ding systems seminar <https://www.youtube.com/watch?v=0kxo9tJ2N8c>



John Shalf  
Department Head for Computer Science  
Lawrence Berkeley National Laboratory

Memsys 2025 Keynote: Codesign for Energy Efficient Computing  
Adaptable Memory Systems for the Future of AI and HPC



locality

Almost all the entire cost is the transportation cost.

Bill Dally AHA [retreat](#)

| Component             | Energy         | What it represents                                      |
|-----------------------|----------------|---------------------------------------------------------|
| On-chip communication | ~100 fJ/bit-mm | Moving one bit one millimeter across on-chip wires      |
| Small SRAM access     | ~50 fJ/bit     | Reading/writing one bit from a small (local) SRAM macro |
| Arithmetic (add)      | ~1 fJ/bit      | One bit of addition logic toggling                      |

Bill Dally's ACM "Computational Model" ([claude](#)), chatgpt [again](#)



**Updated numbers (2024-2025 era, ~3-5nm)**

There hasn't been a single authoritative "Horowitz 2.0" table for 3nm/5nm — the community still cites the 2014 numbers and scales them. But from recent published chips and papers, here are approximate updated figures:

| Operation               | ~45nm (Horowitz 2014) | ~7nm (Dally 2022) | ~3-5nm (2024-25 est.)          |
|-------------------------|-----------------------|-------------------|--------------------------------|
| 32-bit int ADD          | 0.1 pJ                | 0.02 pJ (20 fJ)   | ~10-15 fJ                      |
| 32-bit FP MUL           | 3.7 pJ                | ~0.5-0.8 pJ       | ~0.3-0.5 pJ                    |
| 8-bit int MAC           | ~0.2 pJ               | ~0.03 pJ          | ~0.02 pJ                       |
| On-chip wire (1mm, 64b) | ~5 pJ                 | ~1.9 pJ           | ~1.5-2 pJ (wires not scaling)  |
| SRAM read (8KB)         | ~5 pJ                 | ~1-2 pJ           | ~0.5-1 pJ                      |
| SRAM read (1MB)         | ~5 pJ                 | ~5 pJ             | ~3-5 pJ (dominated by wires)   |
| DRAM read (64b)         | ~640 pJ               | ~300-640 pJ       | ~200-500 pJ (HBM: ~3-7 pJ/bit) |
| Die-to-die (short)      | ~5 pJ/bit             | ~0.5 pJ/bit       | ~0.25-0.3 pJ/bit               |

## The synthesis: a corrected energy ladder for 7nm

| Operation (64 bits) | Dally CACM | Jouppi 7nm        | vs add   | vs D. ... |
|---------------------|------------|-------------------|----------|-----------|
| Int32 add           | 0.020 pJ   | <b>0.030 pJ</b>   | 1×       | 1/12,000  |
| FP32 add            | —          | <b>0.380 pJ</b>   | 13×      | 1/950     |
| FP32 multiply       | —          | <b>1.31 pJ</b>    | 44×      | 1/275     |
| 8KB SRAM read       | 0.64 pJ    | <b>7.5 pJ</b>     | 250×     | 1/48      |
| 32KB SRAM read      | —          | <b>8.5 pJ</b>     | 283×     | 1/42      |
| 1MB SRAM read       | —          | <b>14 pJ</b>      | 467×     | 1/26      |
| 1mm wire (64b)      | 1.9 pJ     | <b>~1.9 pJ</b>    | 63×      | 1/190     |
| 40mm wire (64b)     | 77 pJ      | <b>~77 pJ</b>     | 2,567×   | 1/4.7     |
| 256MB SRAM read     | 58 pJ      | <b>~65 pJ</b>     | 2,167×   | 1/5.5     |
| HBM2 read           | 320 pJ     | <b>250–450 pJ</b> | ~10,000× | ~1×       |
| DDR3/4 read         | —          | <b>1300 pJ</b>    | 43,300×  | 3.7×      |

Italicized "vs add" column uses Jouppi 7nm Int32 add (30 fJ) as the denominator. "vs DRAM" uses 350 pJ HBM2. Bold = directly measured by Jouppi 2021. The biggest gap between Dally's CACM number and Jouppi's measurement is the 8KB SRAM access (0.64 → 7.5 pJ, ~12×).

Stephen Jones ([video](#))

SPEED OF LIGHT = 300,000,000 M/S  
COMPUTER CLOCK = 3,000,000,000 Hz

So in 1 clock tick light travels 100mm (~4 inches)

SPEED OF LIGHT = 300,000,000 M/S  
COMPUTER CLOCK = 3,000,000,000 Hz  
SPEED OF ELECTRICITY = 60,000,000 M/S  
IN SILICON

So in 1 clock tick electricity travels 20mm (~0.8 inches)

Metric

Ding reuse distance

<https://arxiv.org/abs/2312.14441>

Motivated by Ding (data movement distance) and Bill Dally Manhattan Distance proposed in ACM "on the model of computation" [post](#)

Continuous LRU stack: this is a generalization of a cache hierarchy where instead of having traditional L1, L2, HBM hierarchy with costs like (todo: wesley smith), each byte occupies it's own level of hierarchy. Reading bytes from this incurs the cost equal to square root of distance from the top of the stack. Writing to the stack is free. So for instance, to compute  $a+b$ , we assume  $a, b$  has been loaded onto the stack, the cost would be.

## Bill Dally 2D model

<https://cacm.acm.org/opinion/on-the-model-of-computation-point/>

Bill Dally proposed computation model where memory access incurs cost proportional to Mahnhattan Distance. To make it compatible with existing algorithm which don't specify  $x, y$  locations when storing data, we augment this with continuous LRU stack

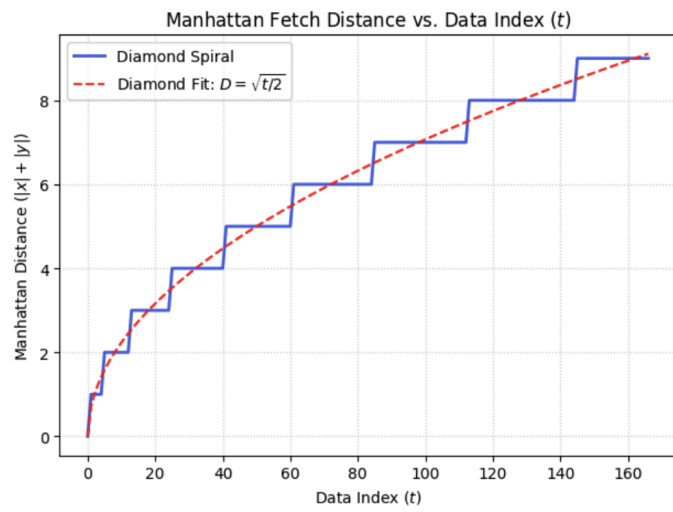
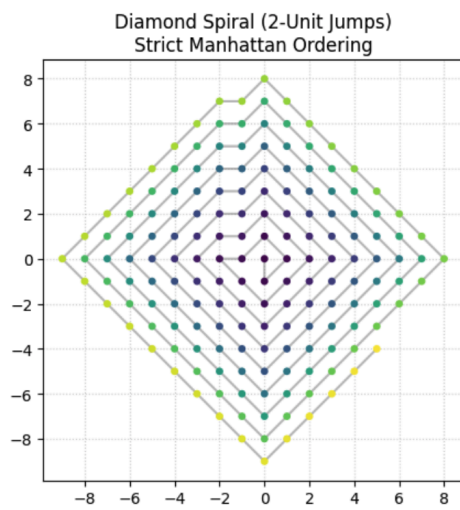
<https://github.com/cybertronai/simplified-dally-model>

far less expensive than a global memory access. On-chip SRAM memories, for example, are built from small 8KB (64Kb) sub-arrays. Accessing 64b from a sub-array costs 0.64pJ and takes 300ps. Accessing a 256MB memory (approximately  $100\text{mm}^2$ ) constructed from these sub-arrays costs 58pJ and 12.3ns—of which 57.4pJ and 12ns are due to communication—15mm each way, 30mm total. Because communication completely dominates the cost of computation, our model of computation must consider communication, and to do so, it must have a model of location.

## Square root energy dependence

[https://colab.research.google.com/drive/1Gt6DNTmnzQsqgLOtwFdmKaAKQfP-hP\\_G#scrollTo=VL9pVgm3orY3](https://colab.research.google.com/drive/1Gt6DNTmnzQsqgLOtwFdmKaAKQfP-hP_G#scrollTo=VL9pVgm3orY3)





|  | d  | Wire length |
|--|----|-------------|
|  | 1  | 1           |
|  | 2  | 2           |
|  | 3  | 2           |
|  | 4  | 2           |
|  | 5  | 3           |
|  | 6  | 3           |
|  | 7  | 3           |
|  | 8  | 3           |
|  | 9  | 3           |
|  | 10 | 4           |
|  | 11 | 4           |
|  | 12 | 4           |
|  | 13 | 4           |
|  | 14 | 4           |
|  | 15 | 4           |
|  | 16 | 4           |

# Cache Hierarchy Optimization

Leonid Yavits, Amir Morad, Ran Ginosar

Department of Electrical Engineering, Technion - Israel Institute of Technology, Haifa, Israel

**Abstract**— Power consumption, off-chip memory bandwidth, chip area and Network on Chip (NoC) capacity are among main chip resources limiting the scalability of Chip Multiprocessors (CMP). A closed form analytical solution for optimizing the CMP cache hierarchy and optimally allocating area among hierarchy levels under such constrained resources is developed. The optimization framework is extended by incorporating the impact of data sharing on cache miss rate. An analytical model for cache access time as a function of cache size is proposed and verified using CACTI simulation.

**Index Terms** — Chip Multiprocessor, Cache Hierarchy, Analytical Performance Models, Resource Allocation Optimization

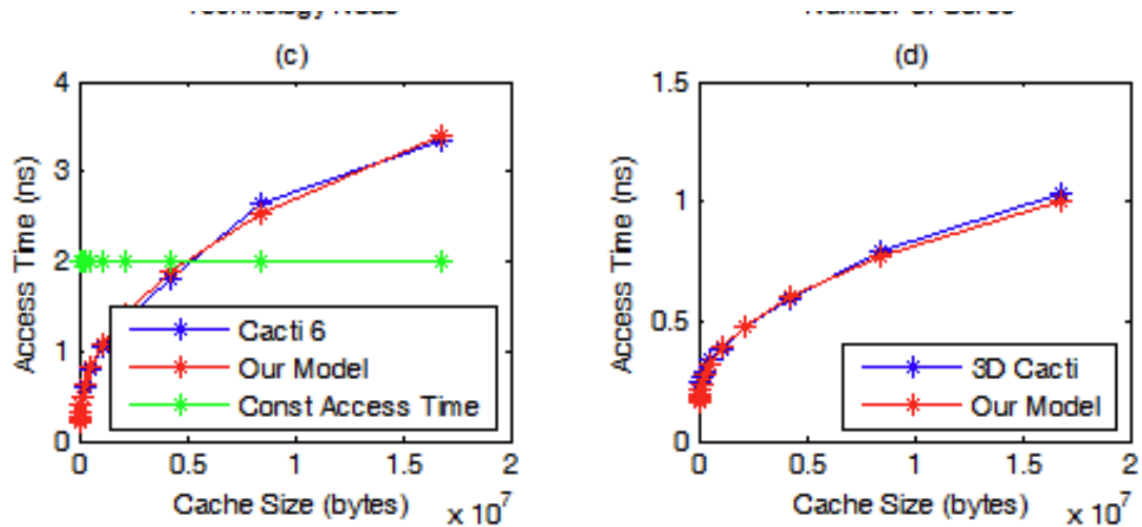


Fig. 1. (a) Power law exponent  $\beta$  vs. technology node; (b) Normalized Cache per Core Area vs. Number of Cores; (c) Cache access time vs. Cache size (CACTI 6); (d) Cache access time vs. Cache size (3D CACTI)

Submission

