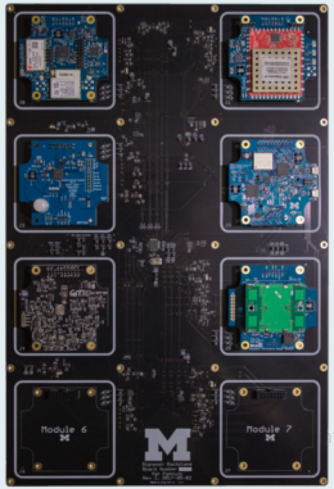
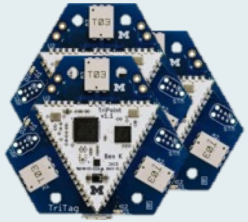
CSE 141L: Introduction to Computer Architecture Lab

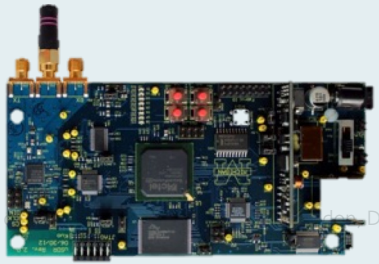
We will start ~12:05 today, but promptly at 12:00 in the future

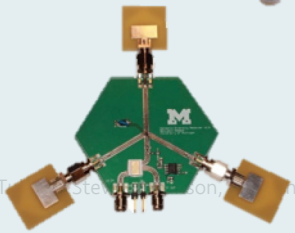
Pat Pannuto, UC San Diego

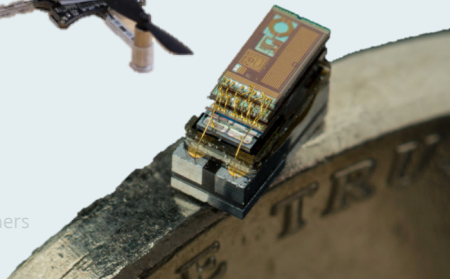
ppannuto@ucsd.edu











What is Computer Architecture and where does it fit in Computer (Science) Engineering?

- One view: what is an Architect and how do they fit in the creation of buildings?

Zaha Hadid

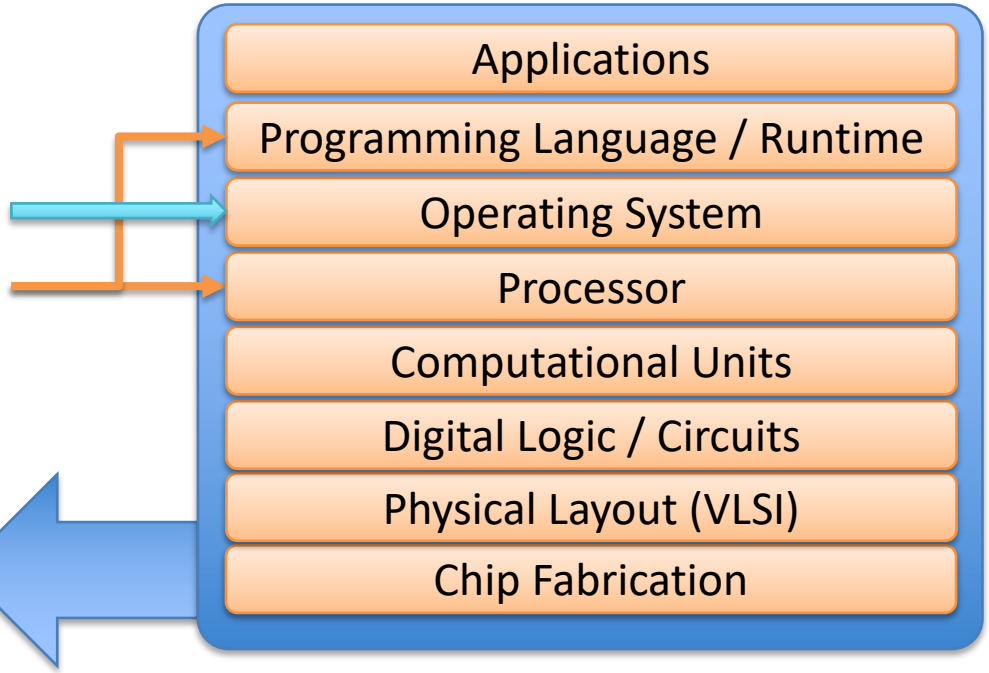
Port Authority Building in Antwerp, designed by Zaha

Graphics courtesy Wikipedia

Computer science is all about abstractions

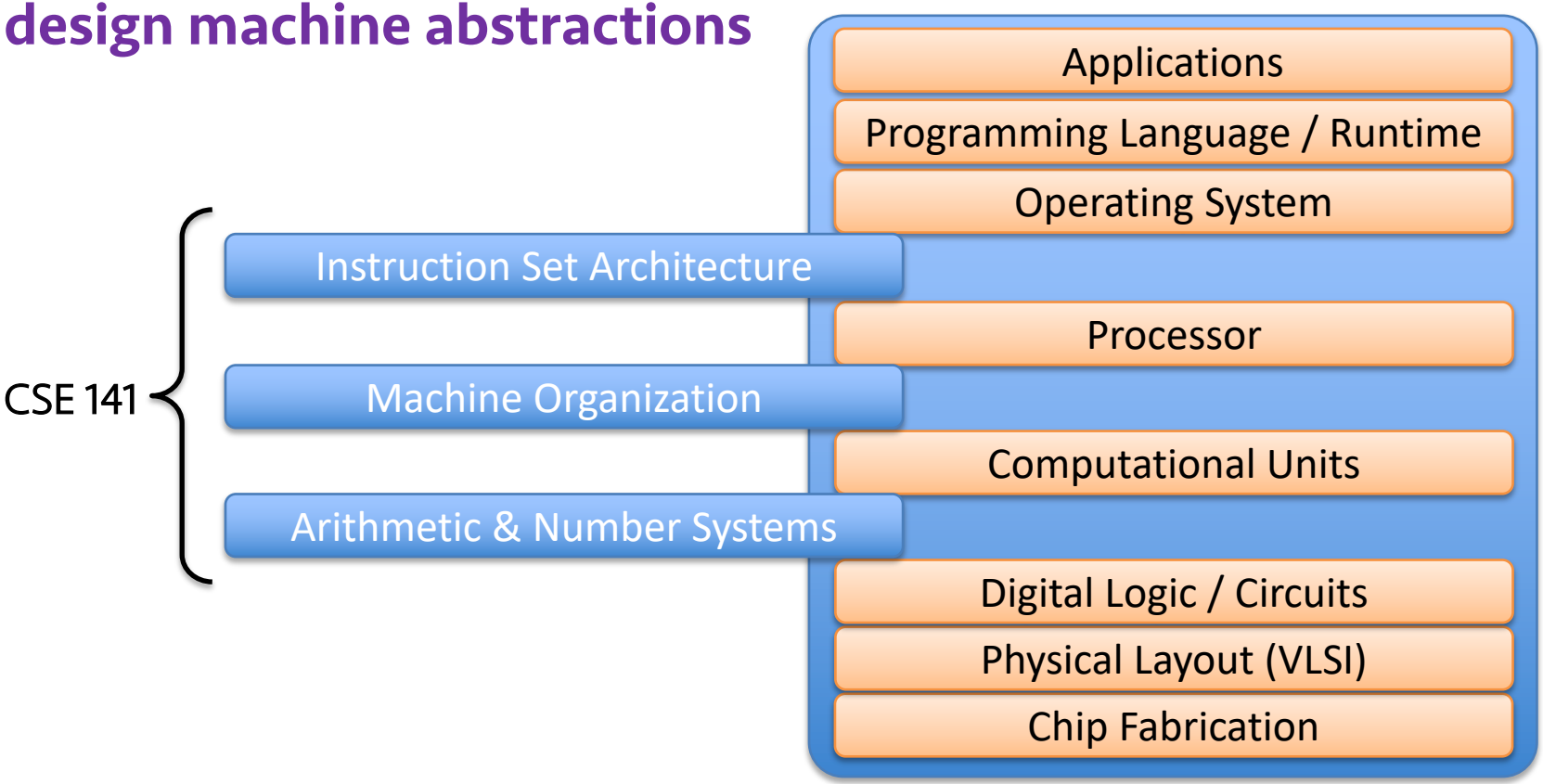
The person who designs this...

...only needs to worry about these



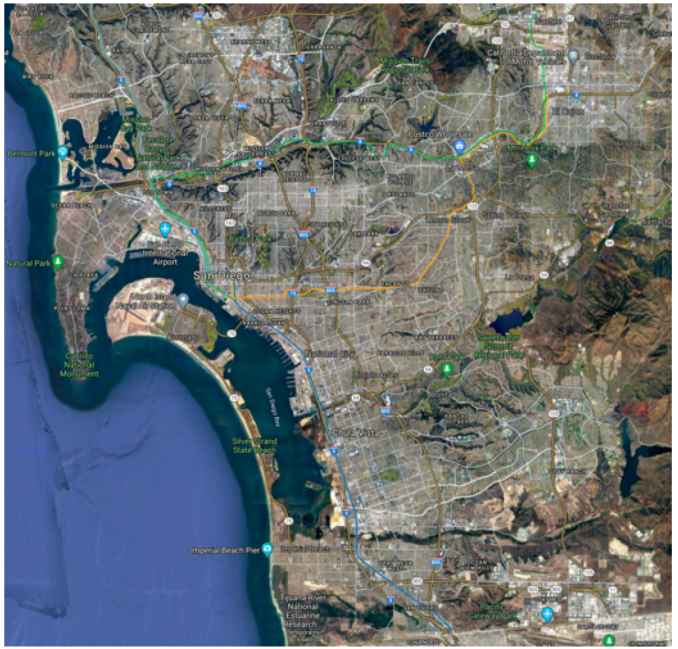
But who designs the big picture?

Computer architects look at the system as a whole and design machine abstractions



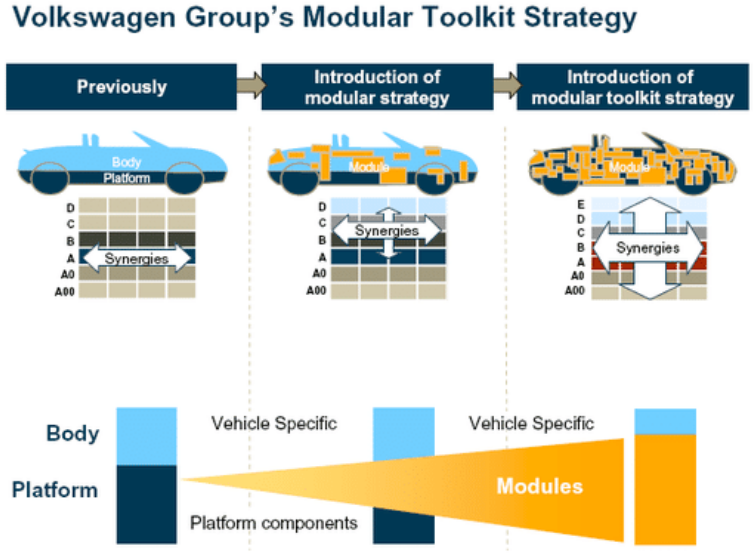
Good abstractions make it easier to focus on reasoning about one part of a large, complex system

- Which of these maps is easier to use to plan a trolley trip?

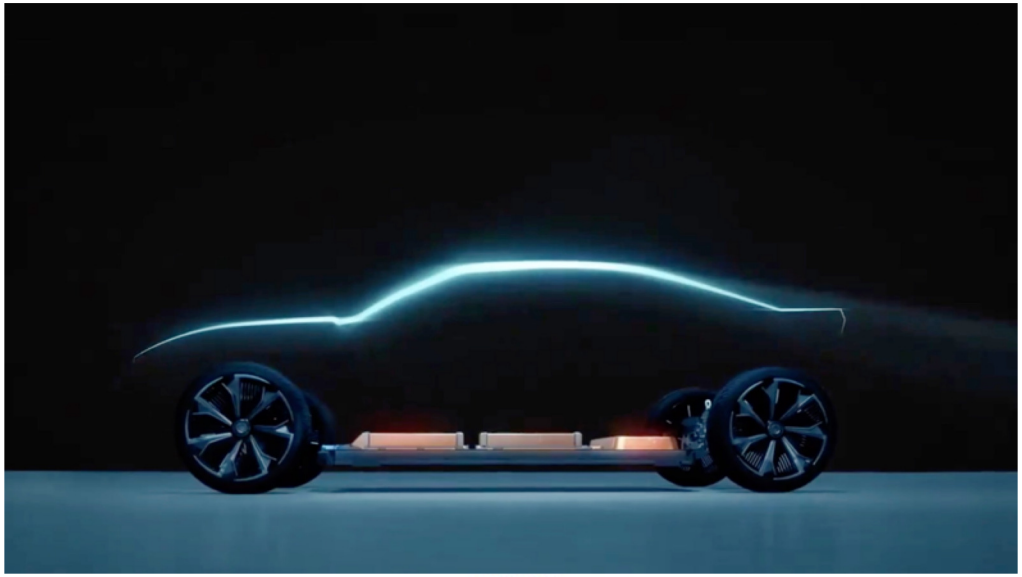


Good abstractions make it easier to focus on reasoning about one part of a large, complex system

- Modularization is fundamental to design in many domains



Modular Car Body Design and Optimization by an Implicit Parameterization Technique via SFE CONCEPT
Fabien Duddeck, Hans Zimmer

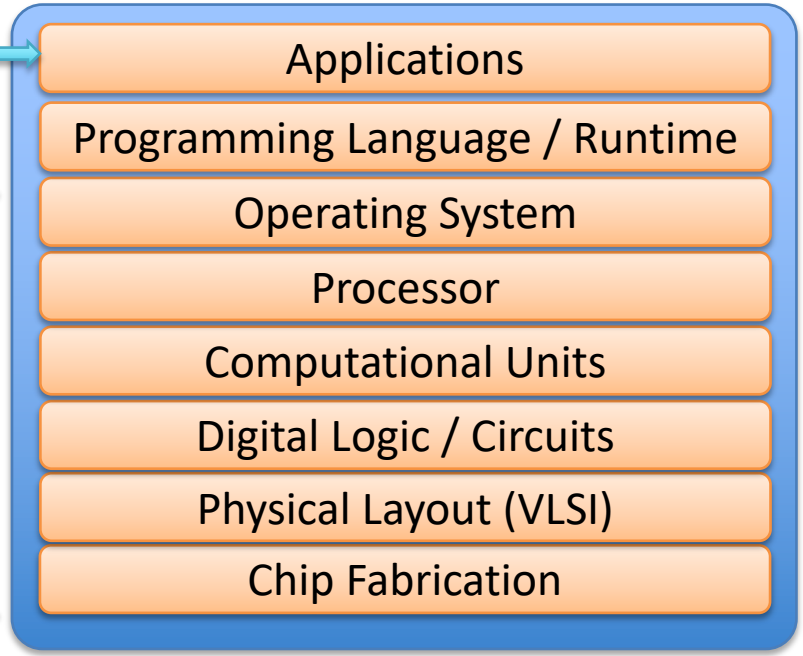


https://www.reddit.com/r/dataisbeautiful/comments/8m15g9/automobile_platform_sharing_work_in_progress/

But what if I'm not going to become a computer architect?

If I only want to build these…

…why do I need to know about any of this?



The real world is full of leaky abstractions

- **Goal:** Sum up all the entries of a two dimensional array
- Which of these implementations is faster?

```
int twoDarray[256][256];
int sum = 0;

for (int i=0; i<256; i++) {
  for (int j=0; j<256; j++) {
    sum += twoDarray[i][j];
  }
}
```

```
int twoDarray[256][256];
int sum = 0;

for (int i=0; i<256; i++) {
  for (int j=0; j<256; j++) {
    sum += twoDarray[j][i];
  }
}
```

Answer: "It depends"

This class is an opportunity to practice architecture

- You will implement a boutique processor
  - *Better* than general purpose for specific tasks
  - *Worse* (or incapable) at things it was not designed for
  - Can you think of some real-world examples of these?
- This will require end-to-end thinking
  - Lab 1: Design ISA
  - Lab 2: Basic Operation
  - Lab 3: Assembler
  - Lab 4: Whole Thing!

What you will be able to do after this class.

- Write top-notch SystemVerilog
- Employ top-notch HW Design Practices
- Design your own processor
- Design pipelined hardware
- Design HW for Altera
- Write serious amounts of code
- Work in a team
- Debug complex designs
- Think like a HW designer

Is this a good time to take 141L?

- Think carefully about the timing of this class.
- It's one of the most intense in CSE.
  - But also you also learn a lot.
  - If you get a solid "A" in this class, I have no trouble recommending you to a potential employer.
- Contra-indications:
  - Other major project classes
  - High course load
  - Large outside commitments
  - Haven't passed prereqs: CSE 110, 140/140L etc.
  - Haven't taken 141, or aren't enrolled.

Course Administrivia

- Instructor
  - Pat Pannuto
- TAs
  - Adithya Anand
  - Link Lin
  - Chavisa Thamjarat
  - Kanlin Wang

Question Triage: Who to ask what.

- Me:
  - "In lecture, …"
  - "I'm designing my own supercomputer, …"
- TAs:
  - "In the Altera Tools…"
- Me, TA
  - "In my 141L implementation…"
- TAs:
  - "In the 141 ISA project framework, …"
- Me, TA:
  - "In a 2-way set associative cache, …"
  - "In the book…"

Logistics

- Everything is on the course website
  - https://patpannuto.com/classes/2022/winter/cse141L/
  - ^This is also the homepage in Canvas
- We will use Piazza for Q&A
- We will use Gradescope to submit assignments

Class is not a competition

- My philosophy
  - I care whether you learn the material
  - The purpose of a grade is to assess how well you know the material in 141L
  - The purpose of a grade is not to "rank" students
  - I am most successful if everyone in class earns an A

Assessments & Workload

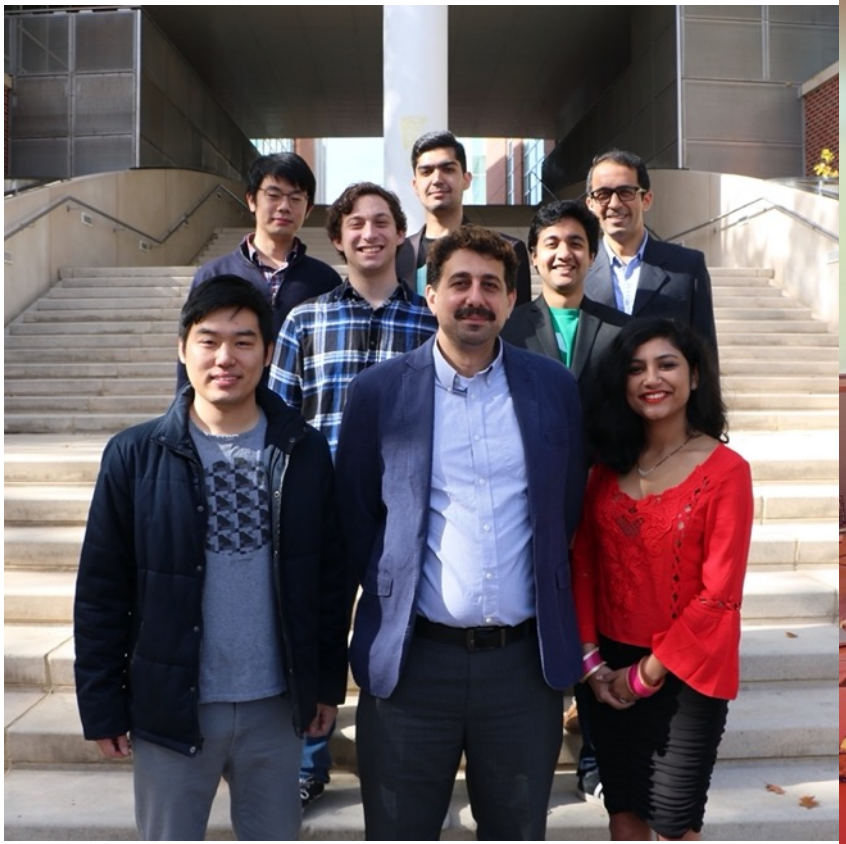
- Grading
  - Your grade is how well your final processor works
  - Milestones (Labs 1, 2, and 3) are to help you stay on track
- Building a custom processor is a lot of work!
  - And *design matters*
  - As you keep implementing, it gets harder and harder to change…

We'll take a short break here...

AND THEN SOME MODERN HIGHLIGHTS FROM HERE AT UCSD

But for the rest of today, I want to highlight the kinds of cool stuff that architects *do*

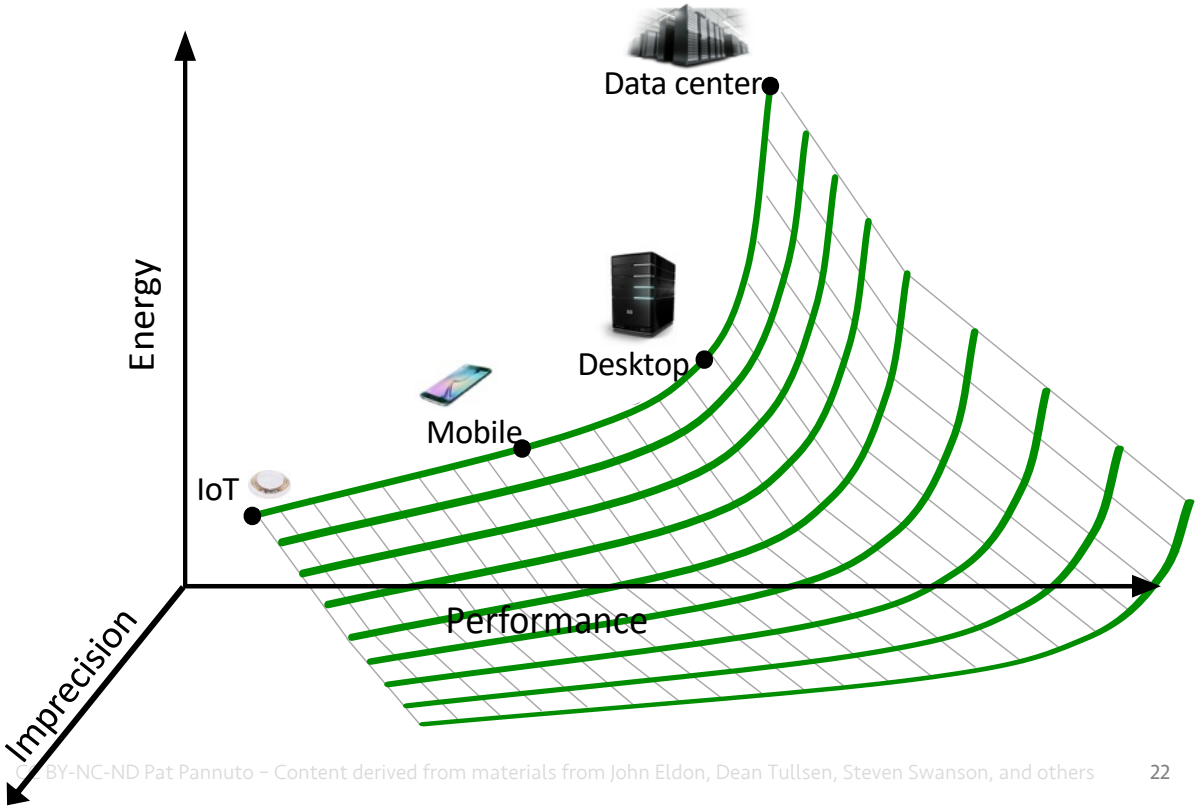
- UCSD has an amazing team of architecture faculty



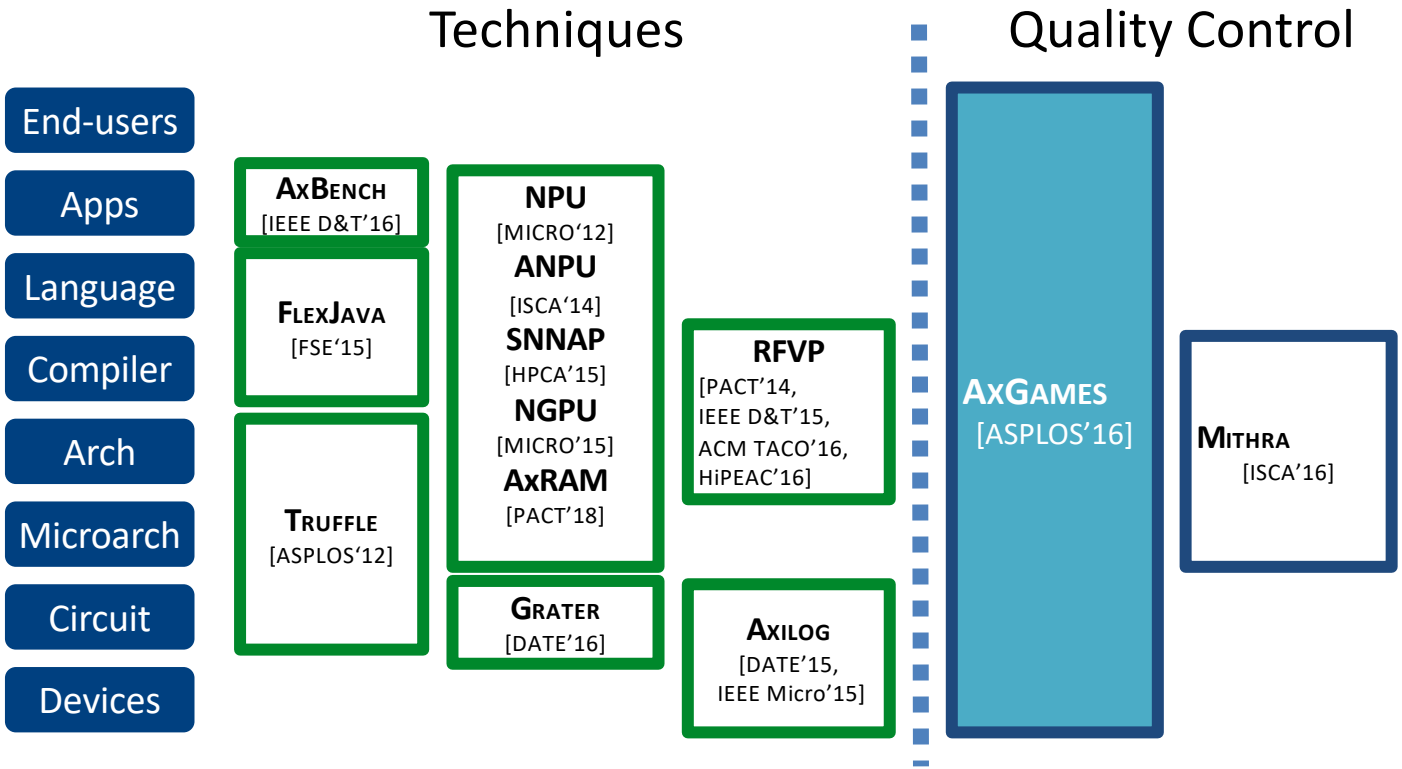
One wild idea: “Approximate Computing”

- Aka, what if 1 + 1 doesn’t *always* equal *exactly* 2?

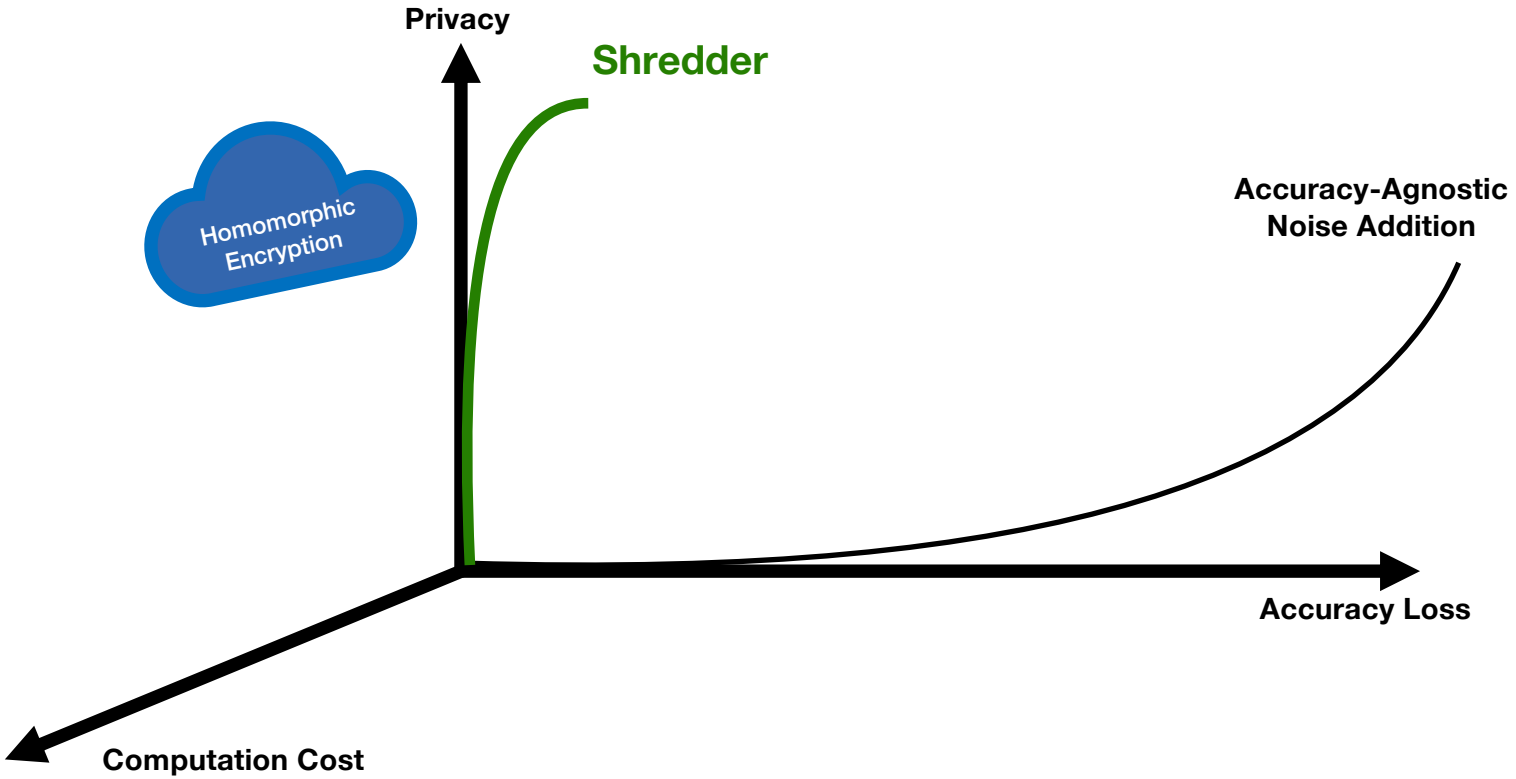
Embracing imprecision allows for major gains in performance and energy



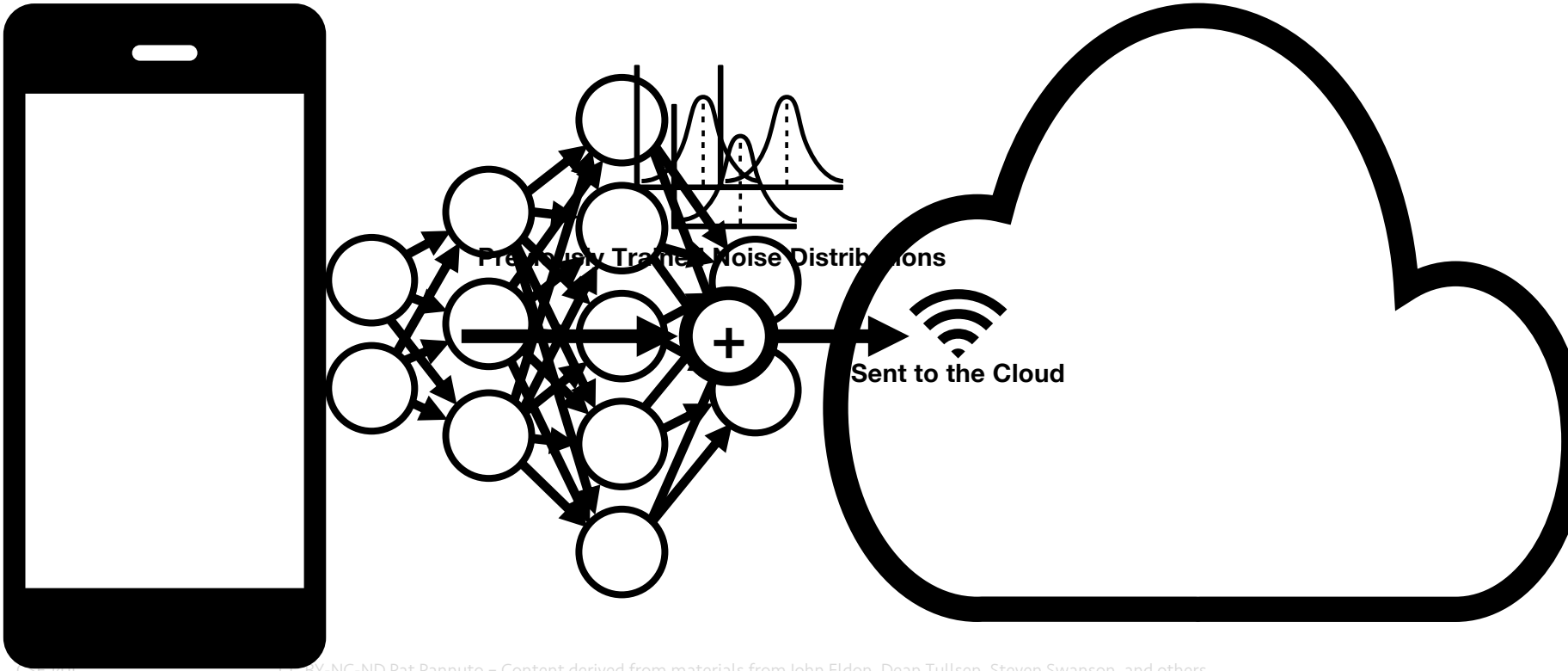
A cross-stack approach to enable approximation



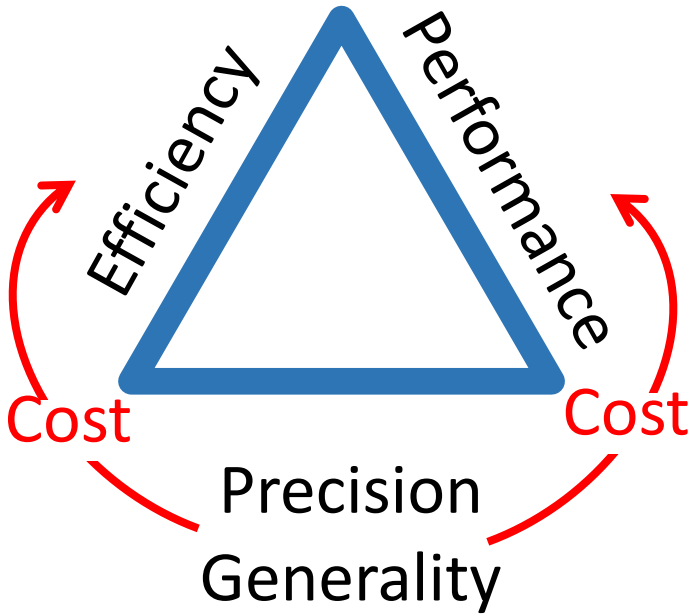
Privacy Preserving Techniques for Inference

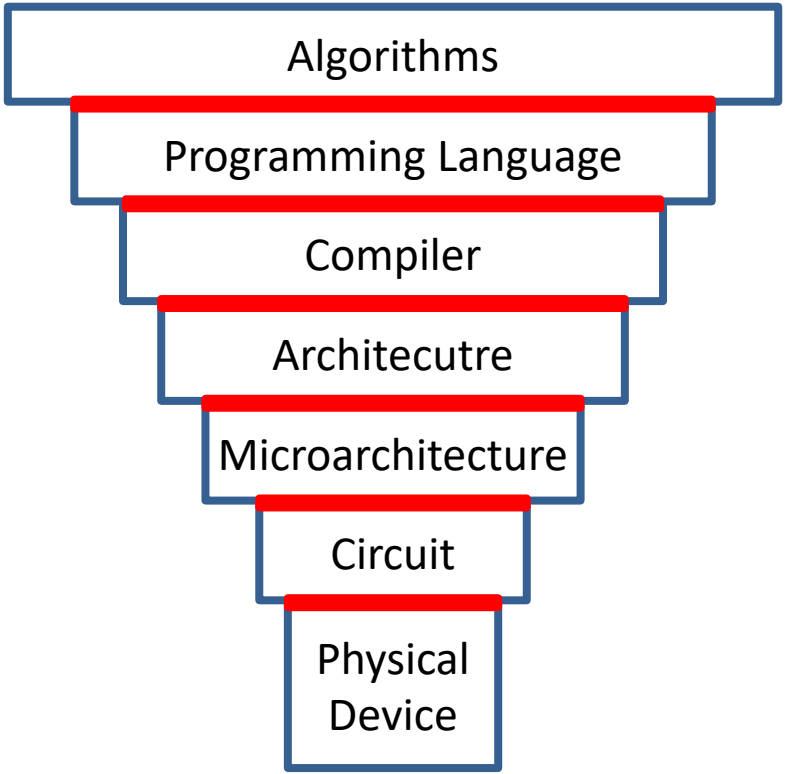


Execution Model



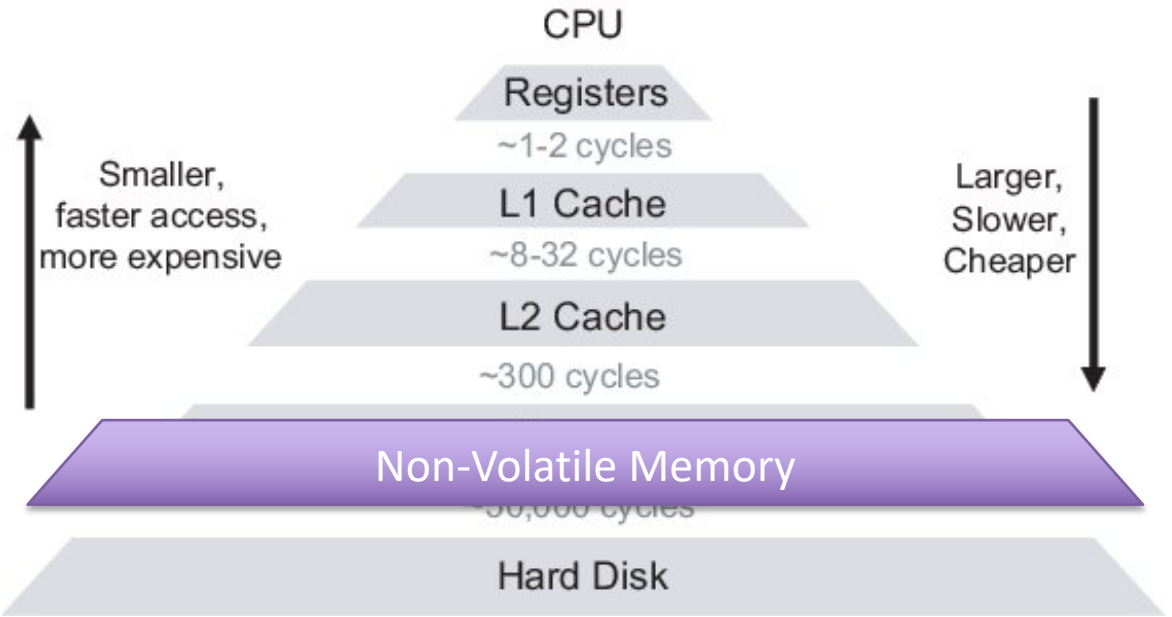
Rethinking the abstractions





Memory, Storage, Software, and Architecture in the NVSL

This is a slide you will encounter in many CE/CSE classes…



Chellappa, Srinivas & Franchetti, Franz & Püschel, Markus. (2007). How to Write Fast Numerical Code: A Small Introduction.

Applications

Tools

Libr es

Ope i Sy m

D ri te Sy ms

re

MARS

Willow

Moneta

QuickSAN

NOVA

Orion

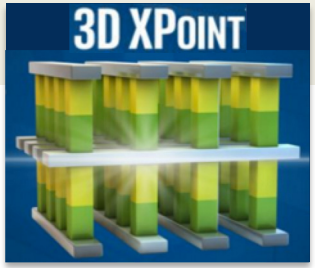
Ziggurat

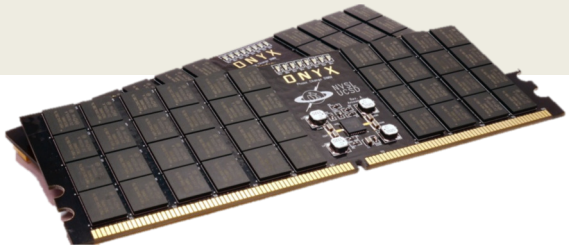
SubZero

NV-Heaps

Pangolin

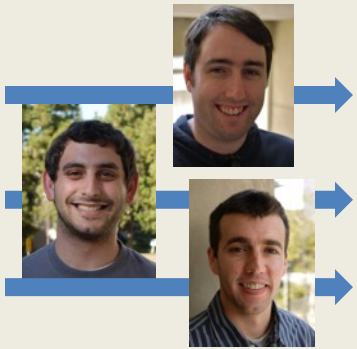
Pronto





NVSL Students Lead Industry

- We Built
  - Opt. SSD interface (2009)
  - Direct, remote SSD (2013)
  - First PCM SSD (2011)
  - PMEM prog. tools (2011)



- Industry Built
  - NVMe (2011)
  - NVMe over Fabrics (2016)
  - Optane (2016)
  - PMDK (~2014)

Mobilizing the Micro-Ops: Exploiting Context Sensitive Decoding for Security and Energy Efficiency

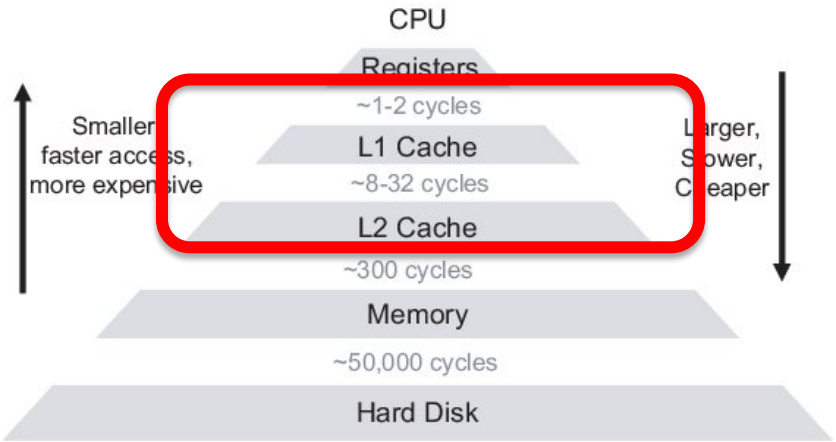
Leaky abstractions are not always just performance problems…

- This loop behaved differently because of how caches work

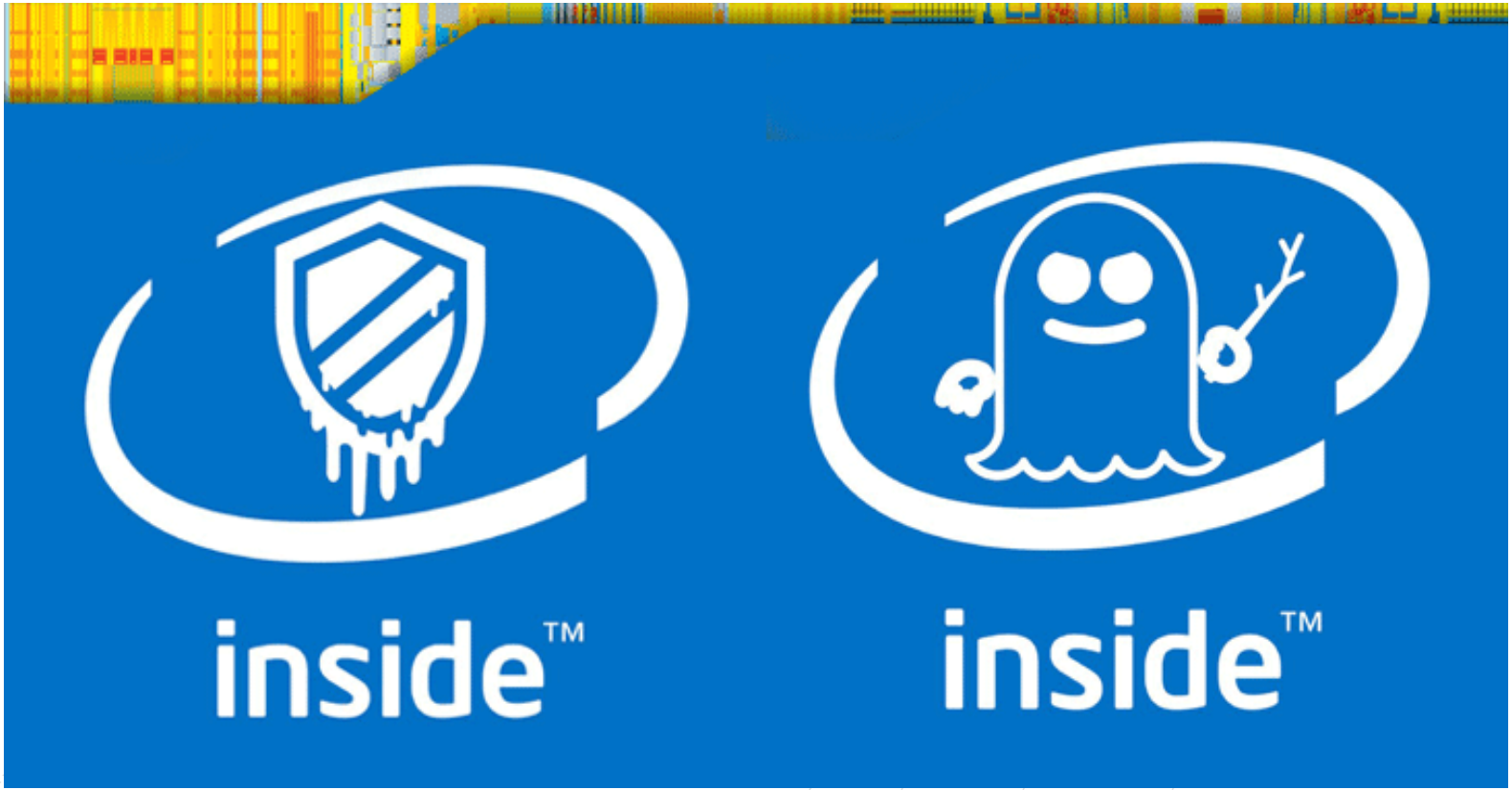
```
int twoDarray[256][256];
int sum = 0;

for (int i=0; i<256; i++) {
  for (int j=0; j<256; j++) {
    sum += twoDarray[i][j];
  }
}
```

Architects added "hidden" caches: faster, intermediate memories

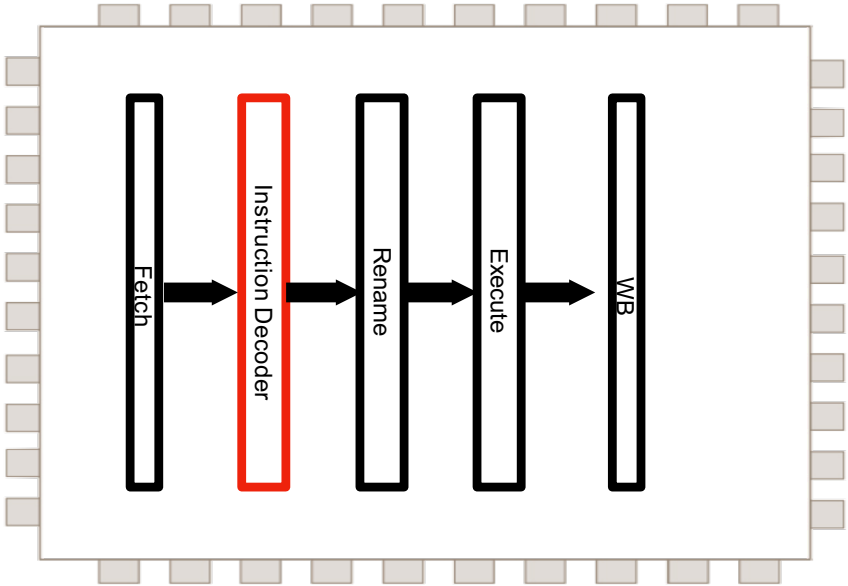


Leaky abstractions can be security threats!



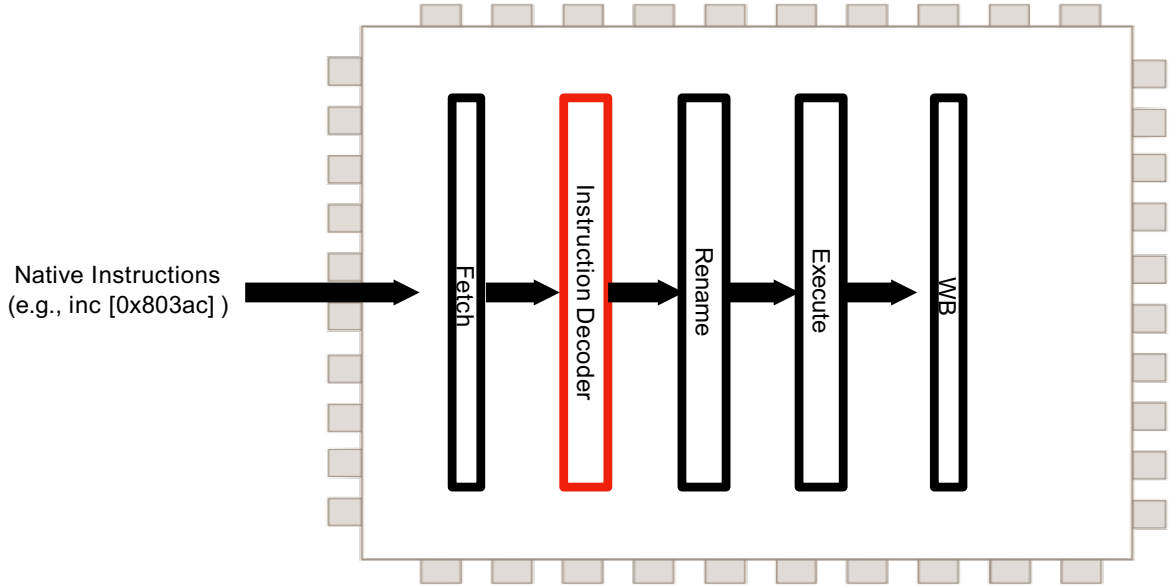
Mobilizing the Micro-Ops

Exploiting Translated ISAs



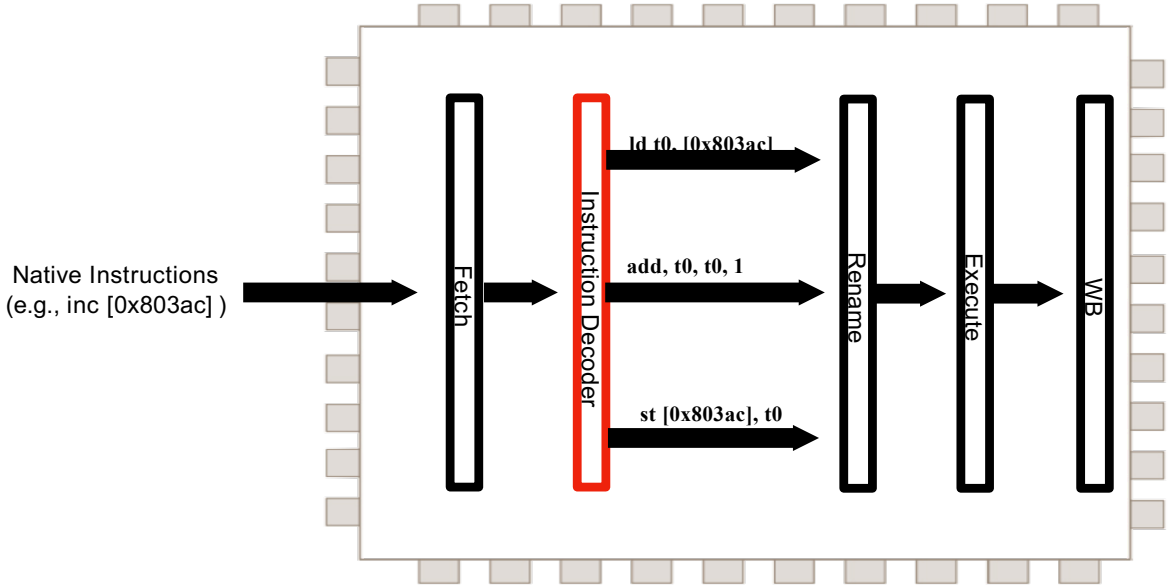
Mobilizing the Micro-Ops

Exploiting Translated ISAs



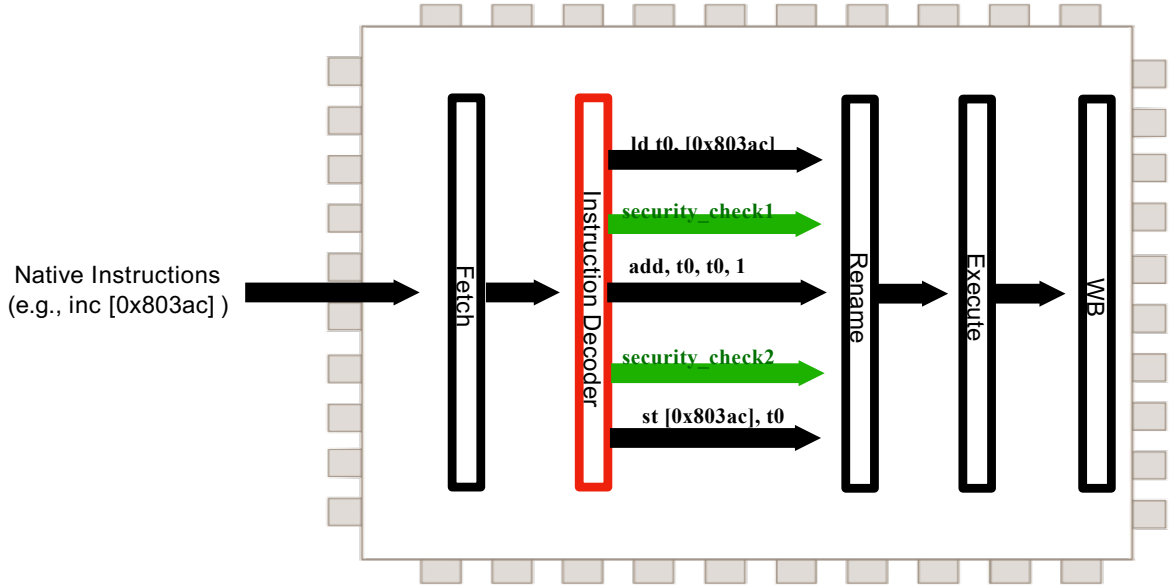
Mobilizing the Micro-Ops

Exploiting Translated ISAs



Mobilizing the Micro-Ops

Exploiting Translated ISAs



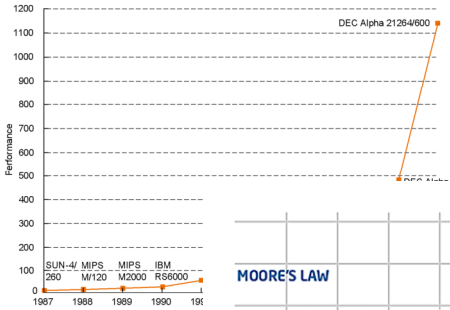
Context Sensitive Decoding fixes a leaky abstraction

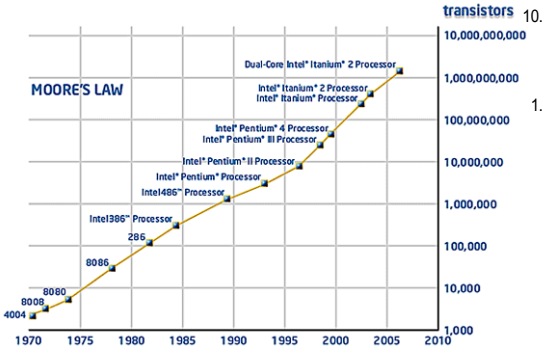
- Eliminating cache side channels via cache obfuscation
- Energy and Performance optimization via selective devectorization
  - ISCA 2018
  - IEEE Micro Top Picks in Computer Architecture

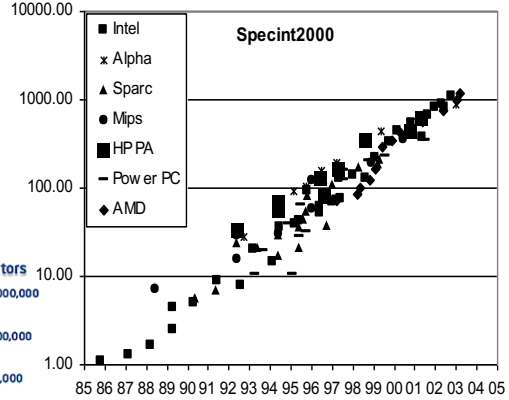
- Spectre mitigation via targeted insertion of fence micro-ops (Context Sensitive Fencing)
  - ASPLOS 2019

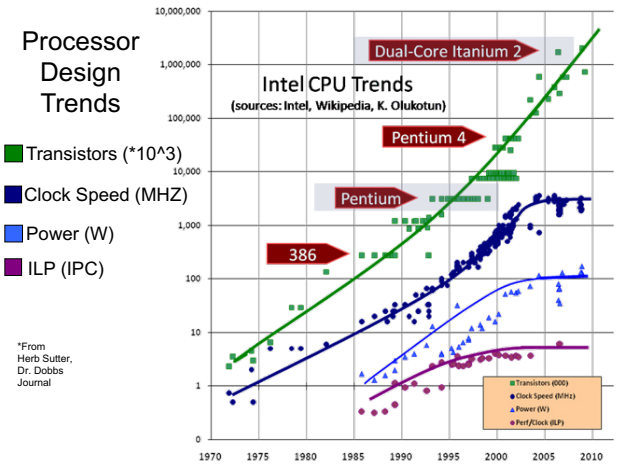
Performance was king, until we unplugged computers

- A lot of "classic" architecture research is makes sure graphs continue to go up and to the right





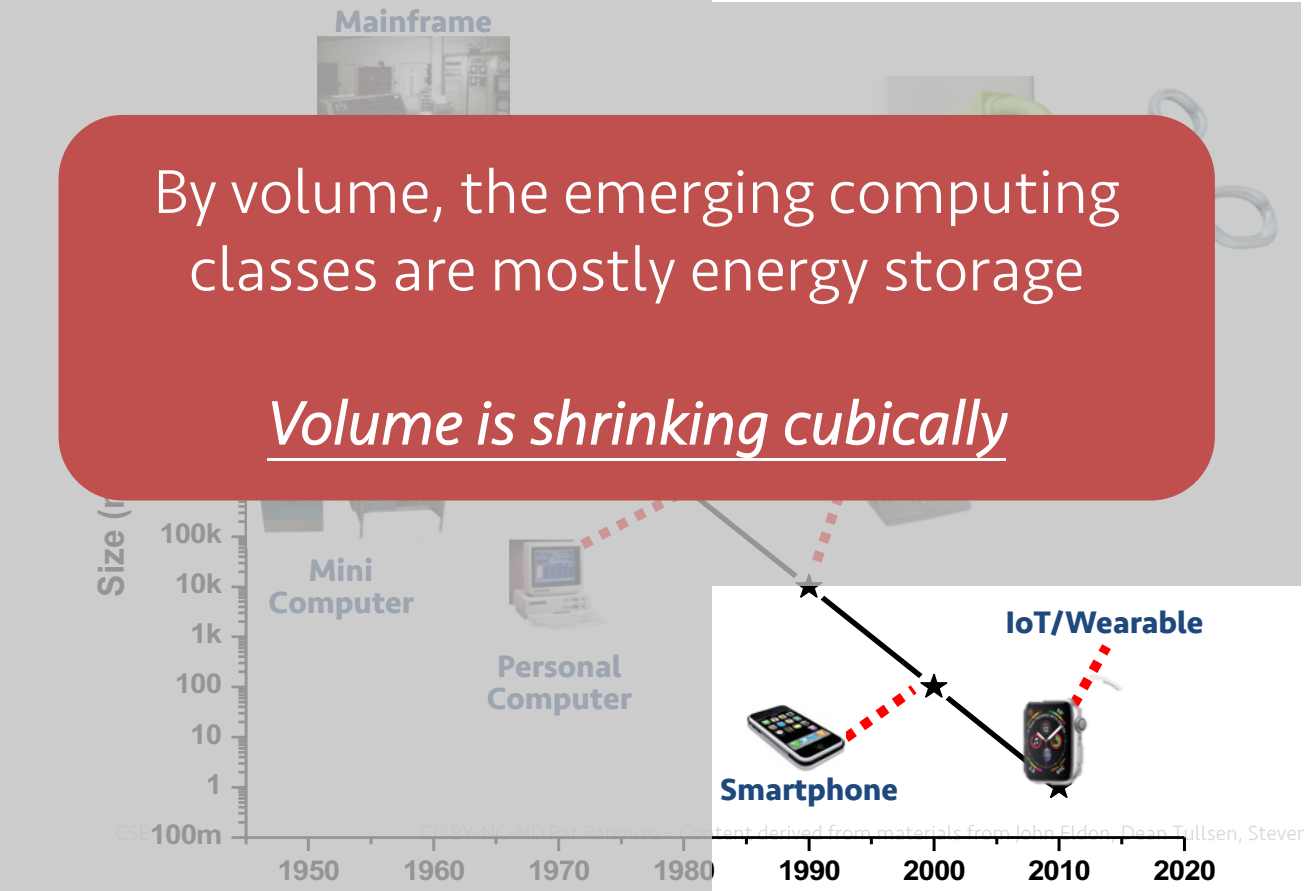


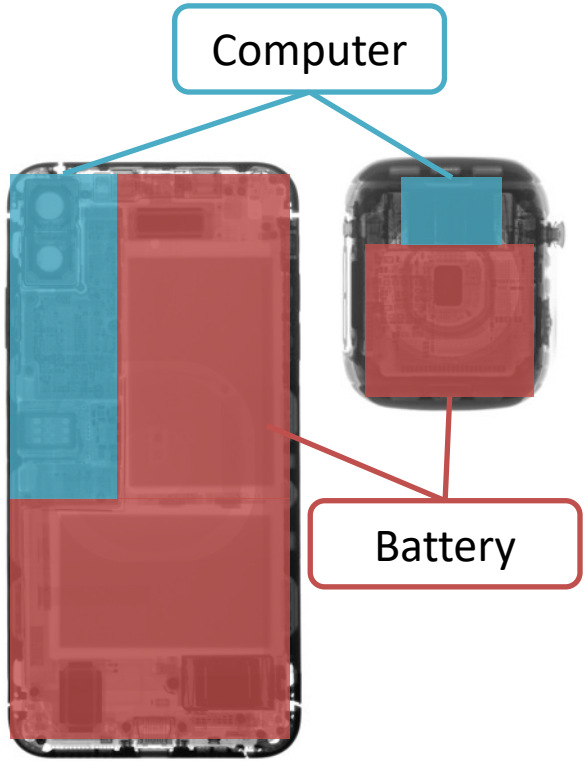


I spend my time on graphs that go down and to the right

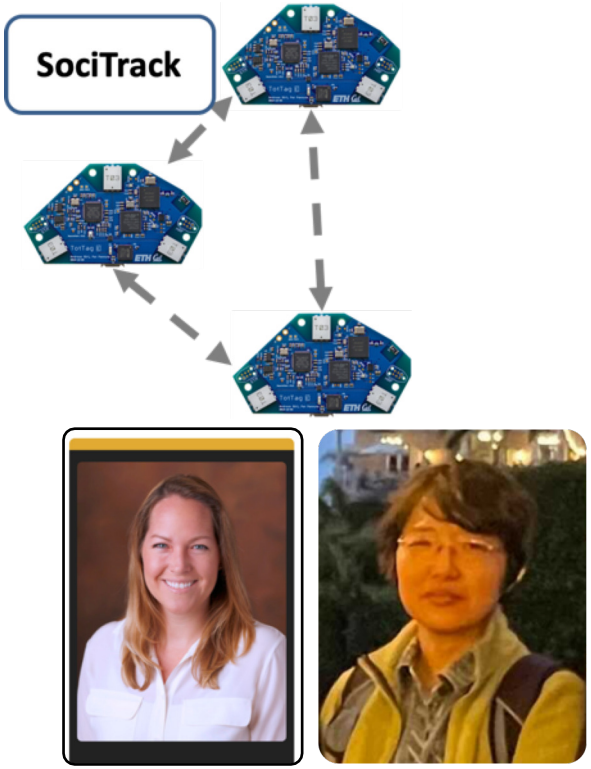
By volume, the emerging computing classes are mostly energy storage

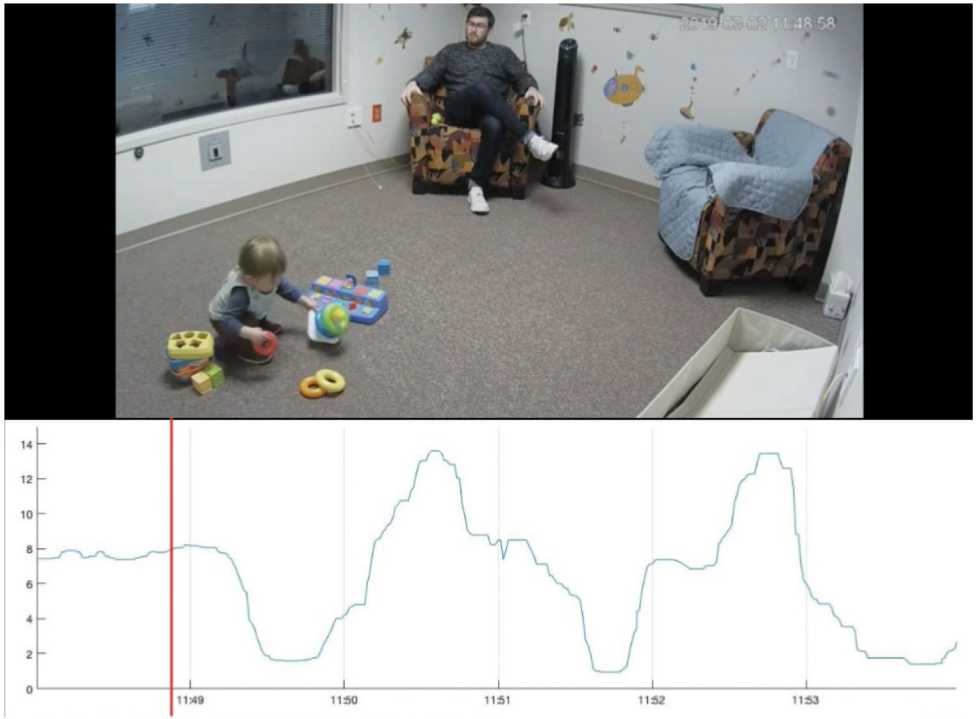
Volume is shrinking cubically





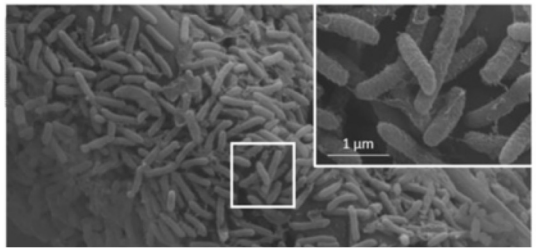
What can we do with resource-constrained computing?

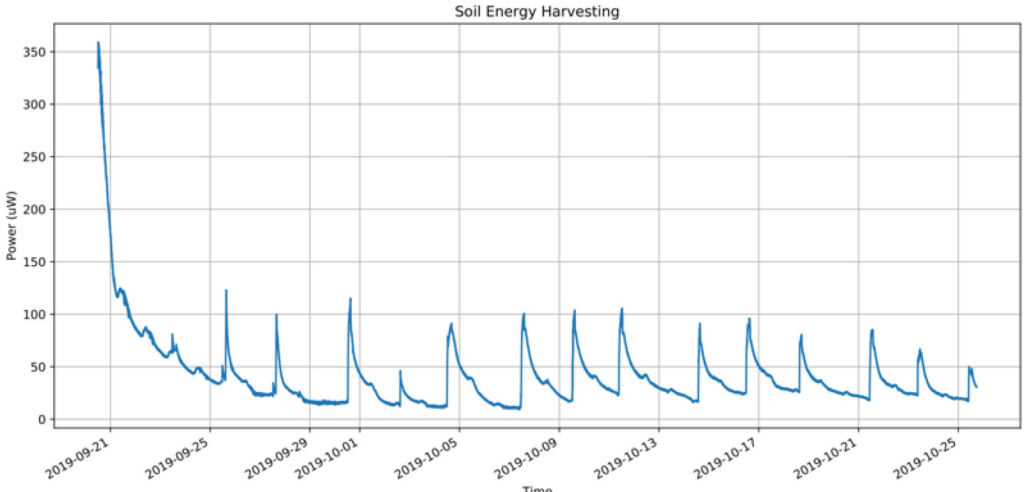




Where can we find resources for constrained computing?

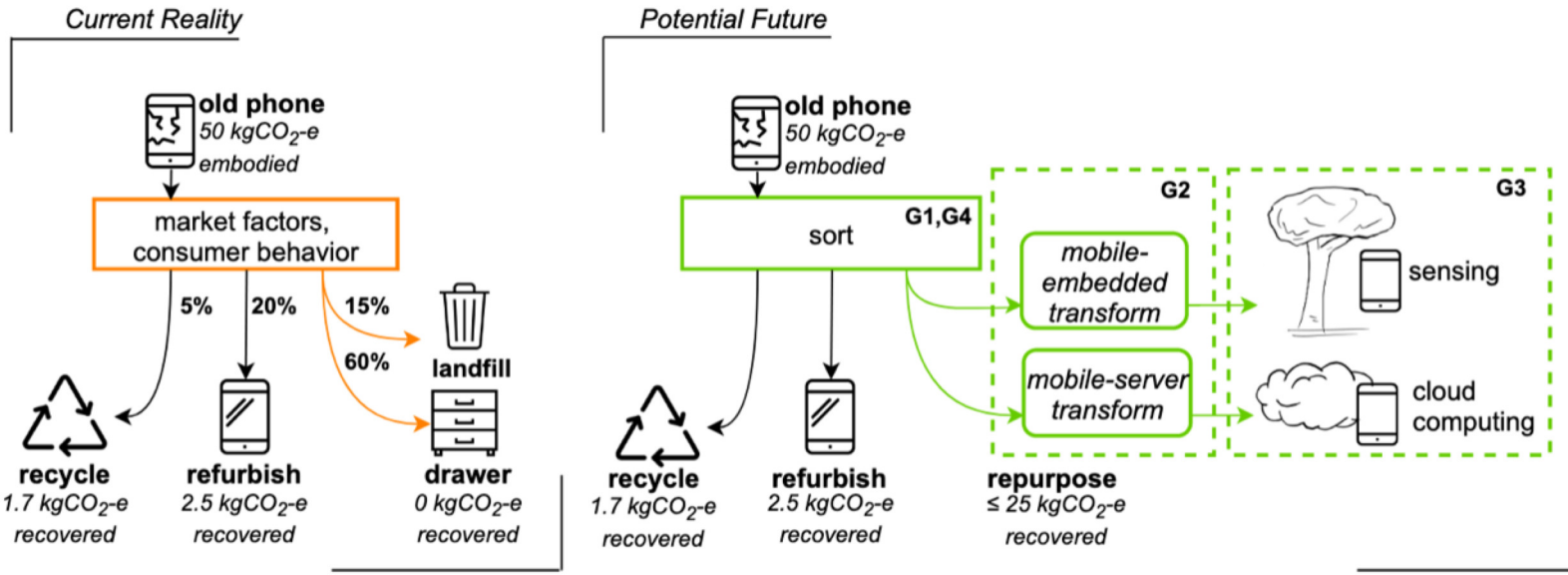
- Soil Microbial Fuel Cells to power computational systems?





What can we do with "old," under-resourced computers?

- How do we divert and re-purpose e-waste?



Wednesday: SystemVerilog

- For those looking to get a jump:
  - Install / Open the course tools
    - ModelSim
    - Quartus
  - [links on the course website]