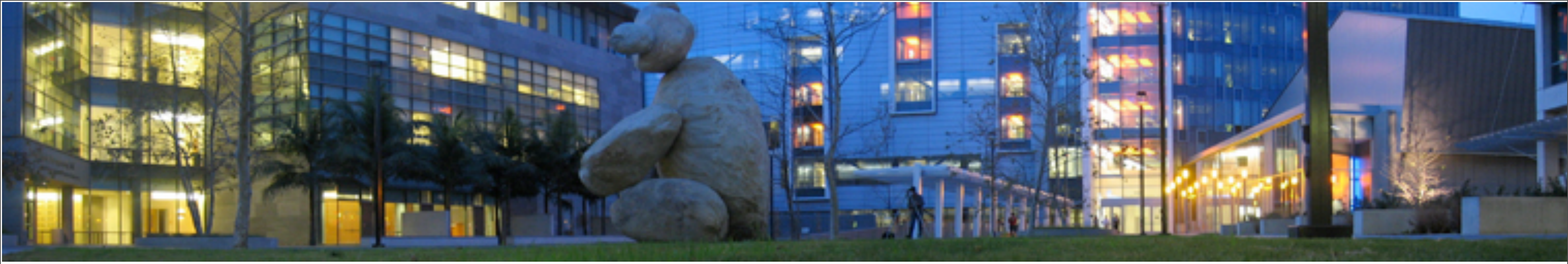


Exploring the Software Stack for Underdesigned Computing Machines

Rajesh Gupta
UC San Diego.

synergy
systems, networking and energy efficiency

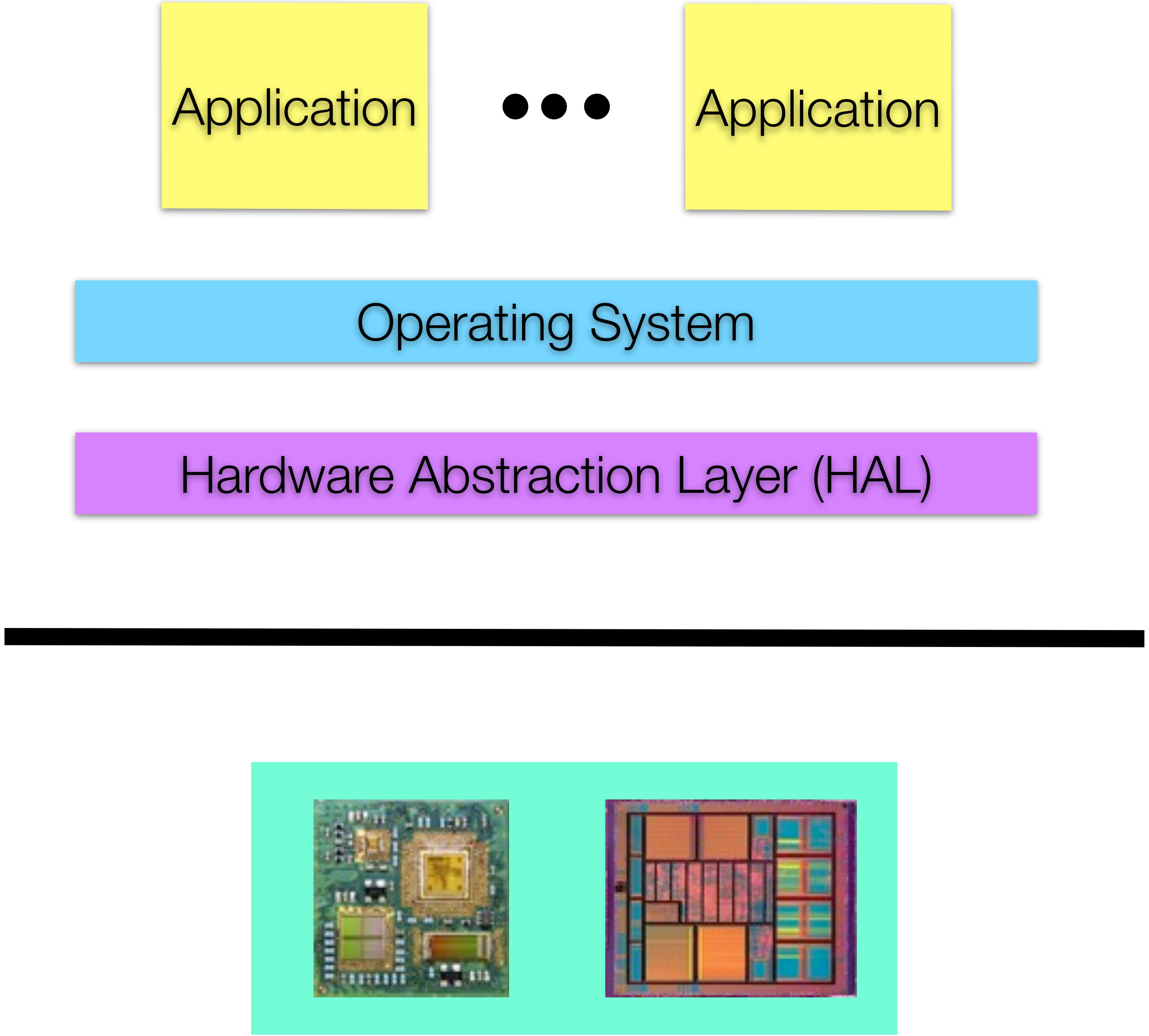
Exploring the Software Stack for Underdesigned Computing Machines

UCLA UCSD

MICHIGAN

The Hardware-Software Boundary

Idealization: hardware has rigid specifications



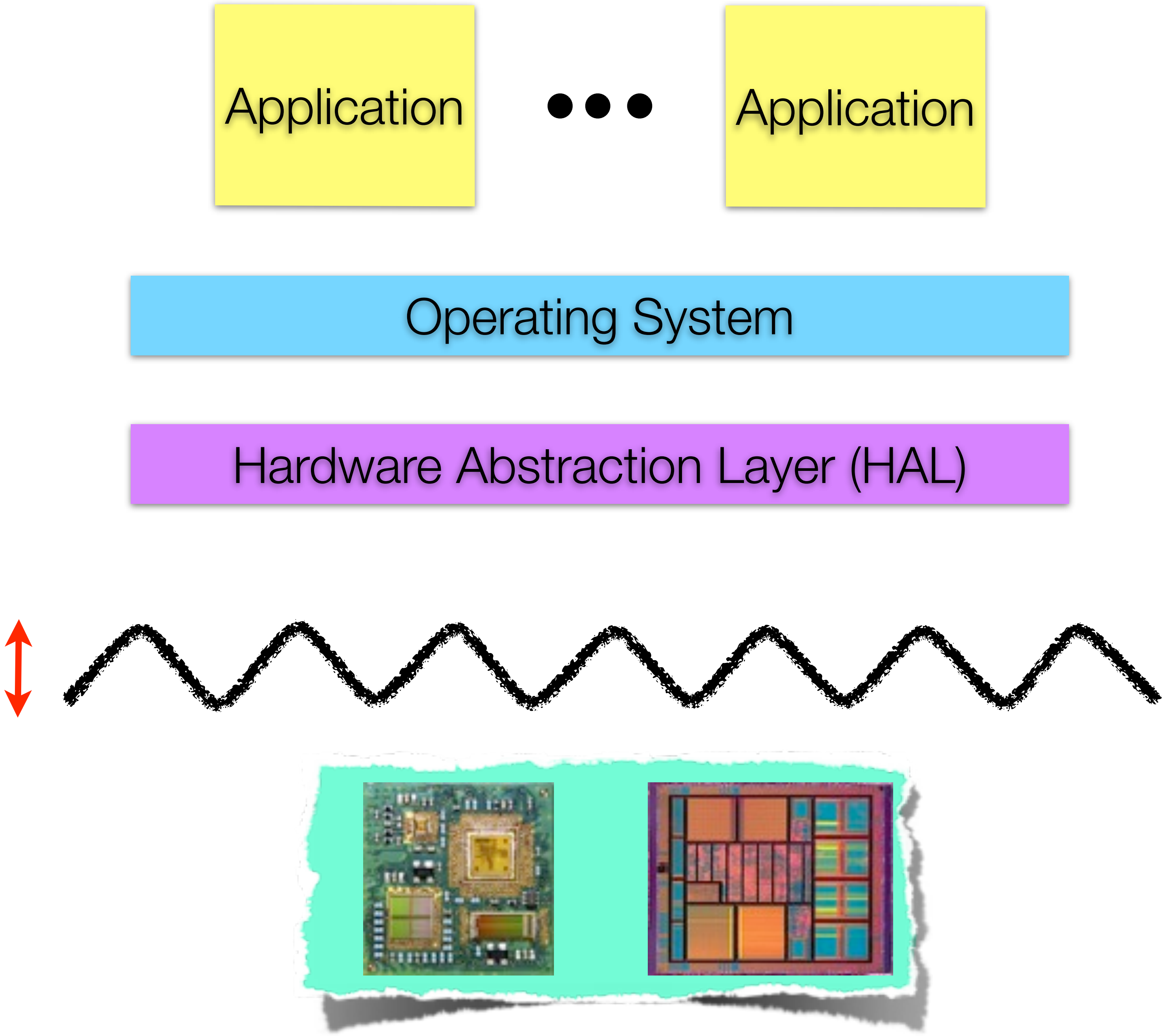
Application ••• Application

Operating System

Hardware Abstraction Layer (HAL)

The Hardware-Software Boundary

Reality: hardware characteristics are highly variable



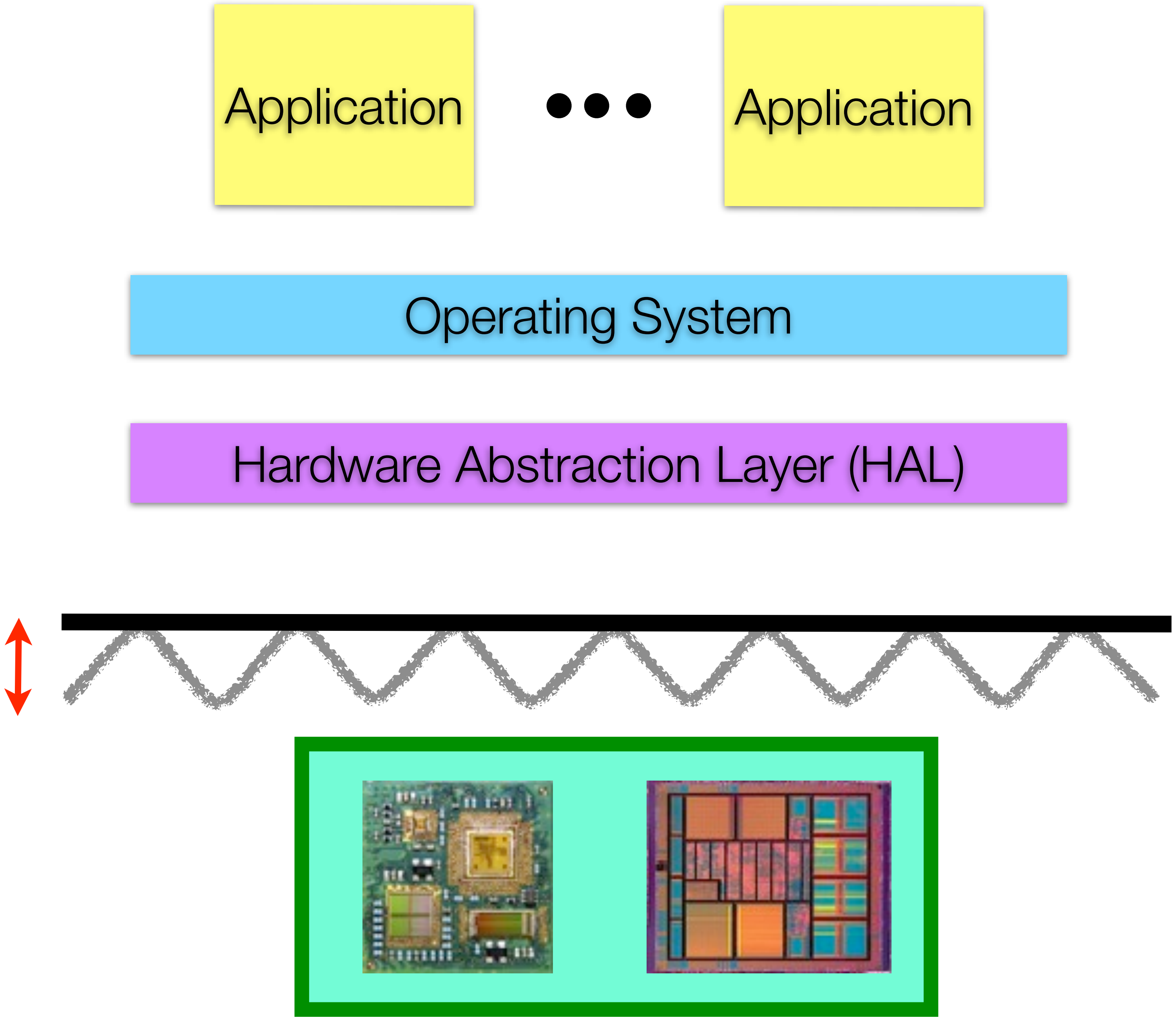
Application ••• Application

Operating System

Hardware Abstraction Layer (HAL)

The Hardware-Software Boundary

*Practice: **over-design** & **guard-banding** for illusion of rigidity*



Application

•••

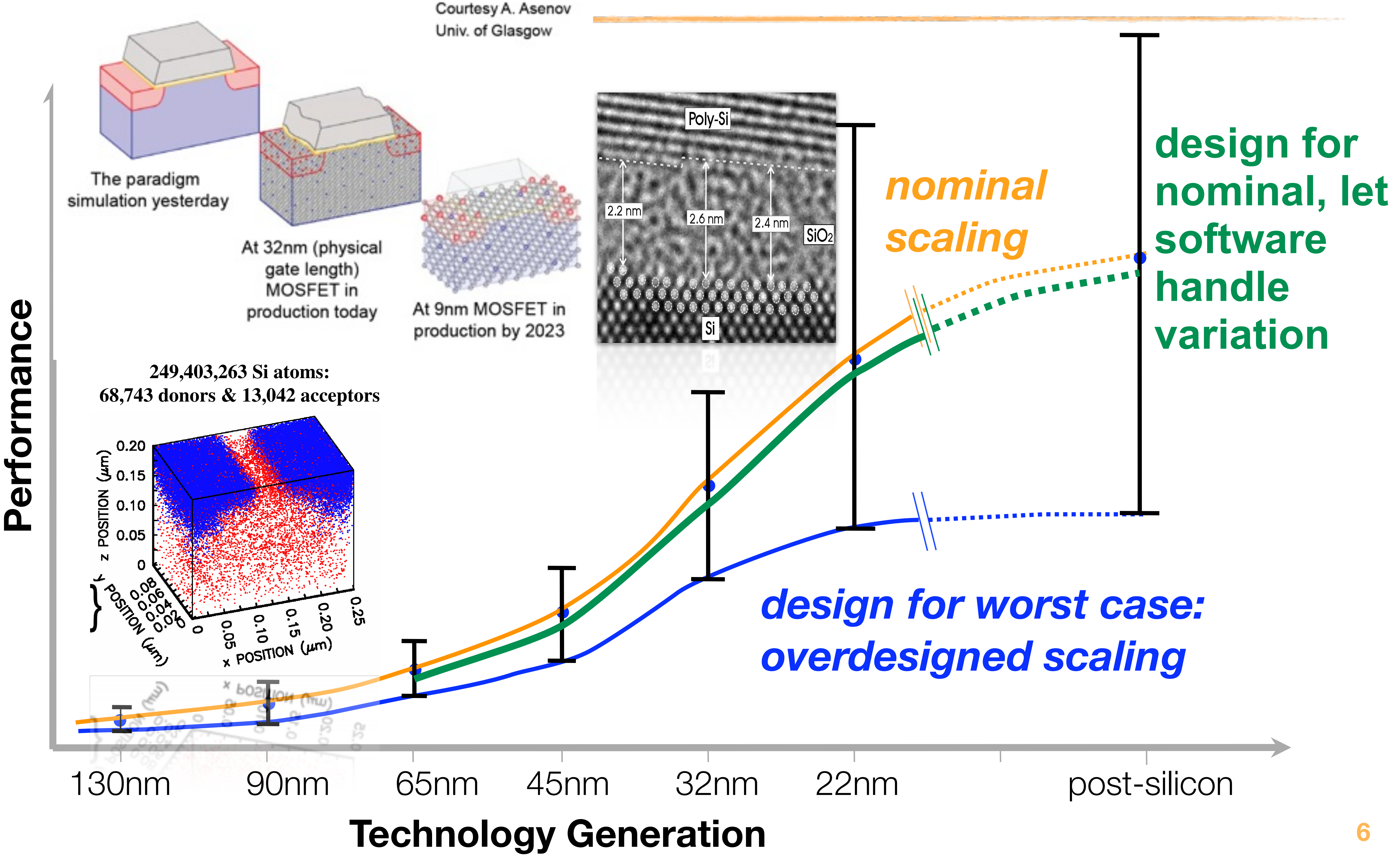
Application

Operating System

Hardware Abstraction Layer (HAL)

Manufacturing Variability Meets Moore’s Law:

From Chiseled Transistors to Molecular Assemblies



Courtesy A. Asenov
Univ. of Glasgow

The paradigm simulation yesterday

At 32nm (physical gate length) MOSFET in production today

At 9nm MOSFET in production by 2023

249,403,263 Si atoms:
68,743 donors & 13,042 acceptors

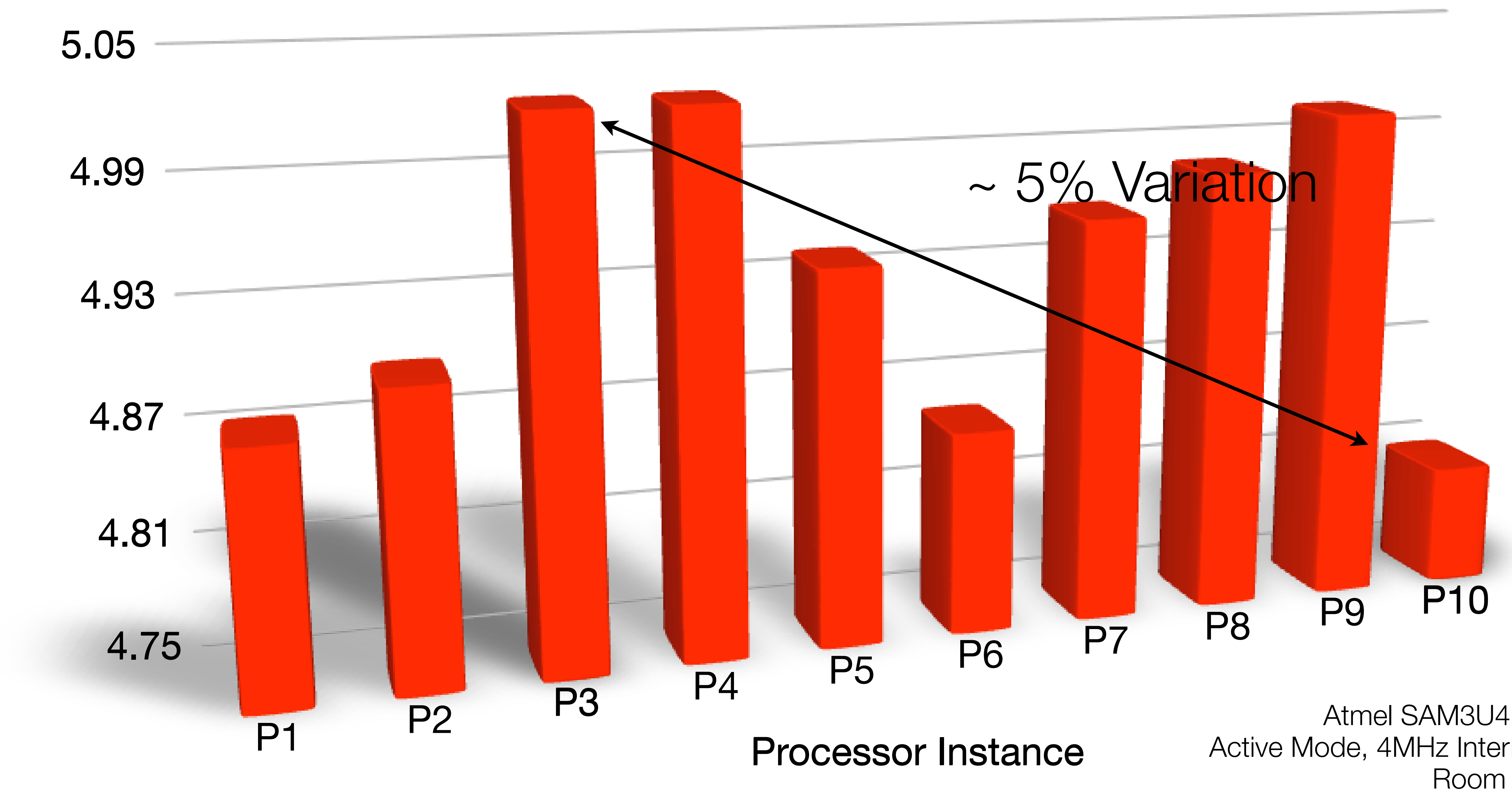
nominal scaling

design for nominal, let software handle variation

design for worst case: overdesigned scaling

Active Power Variability Across Instances

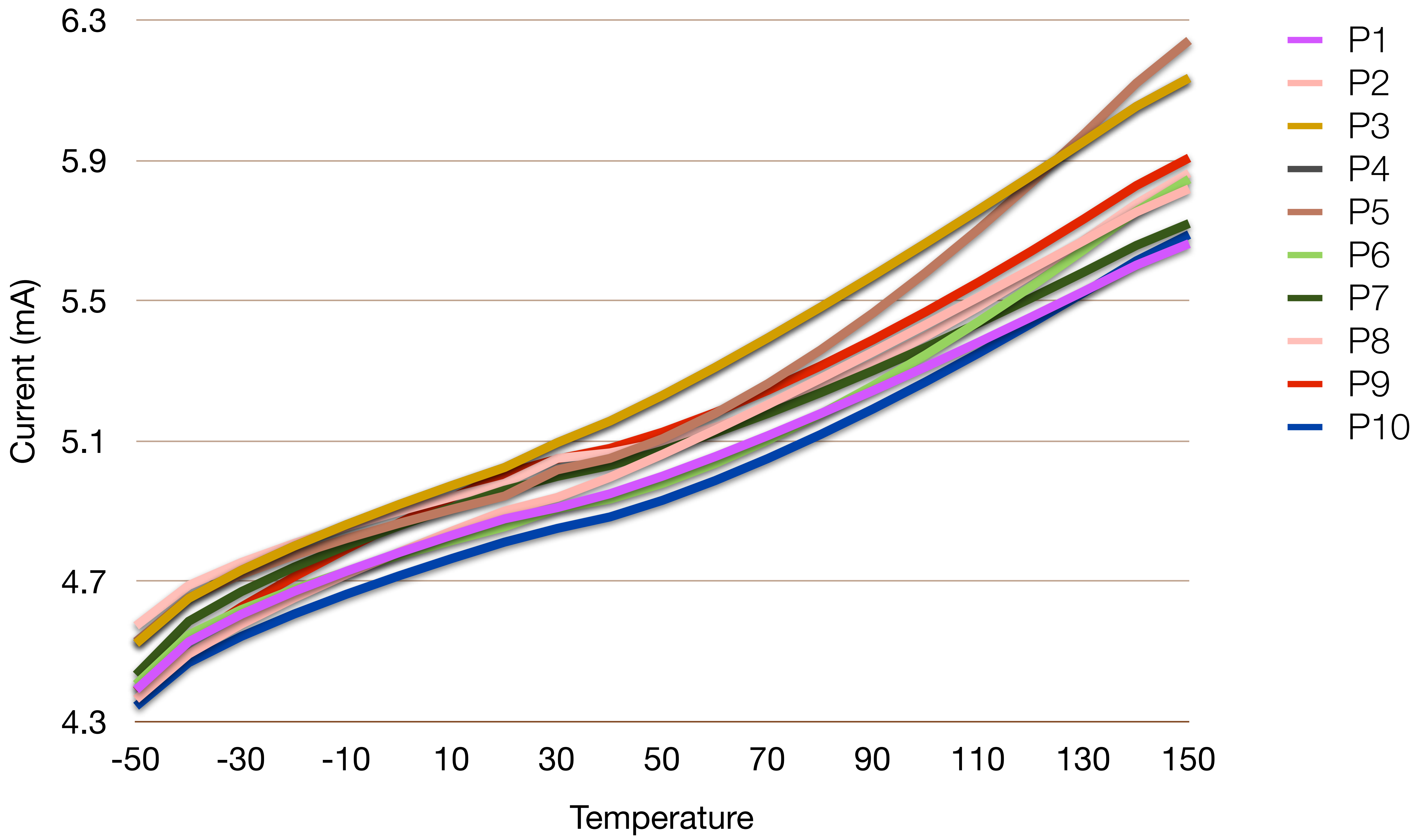
Cortex M3 Active Current @ Room Temperature



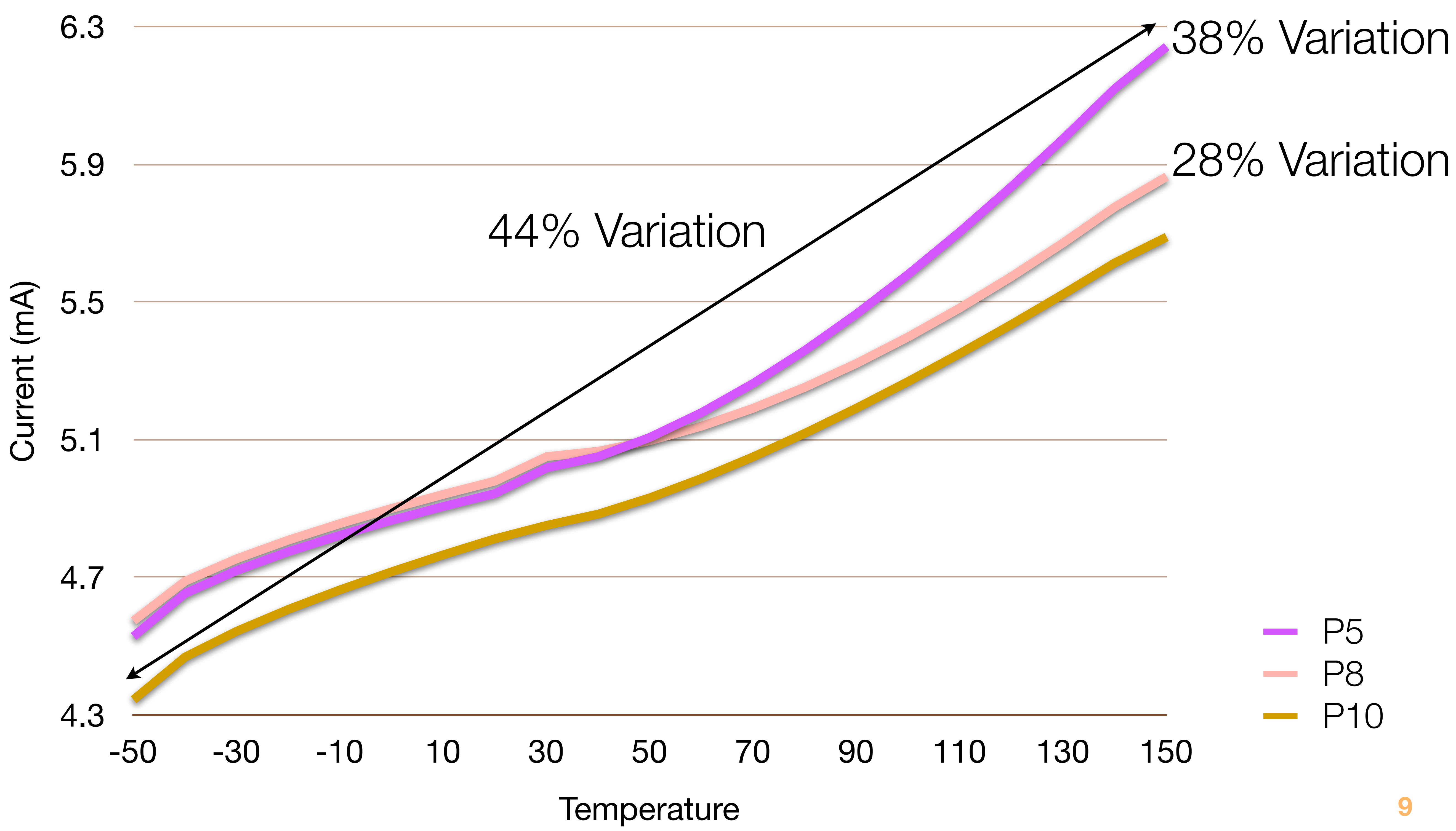
Atmel SAM3U4E Cortex M3
Active Mode, 4MHz Internal Oscillator
Room Temperature

UCLA

Active Power Variability Across Temperature

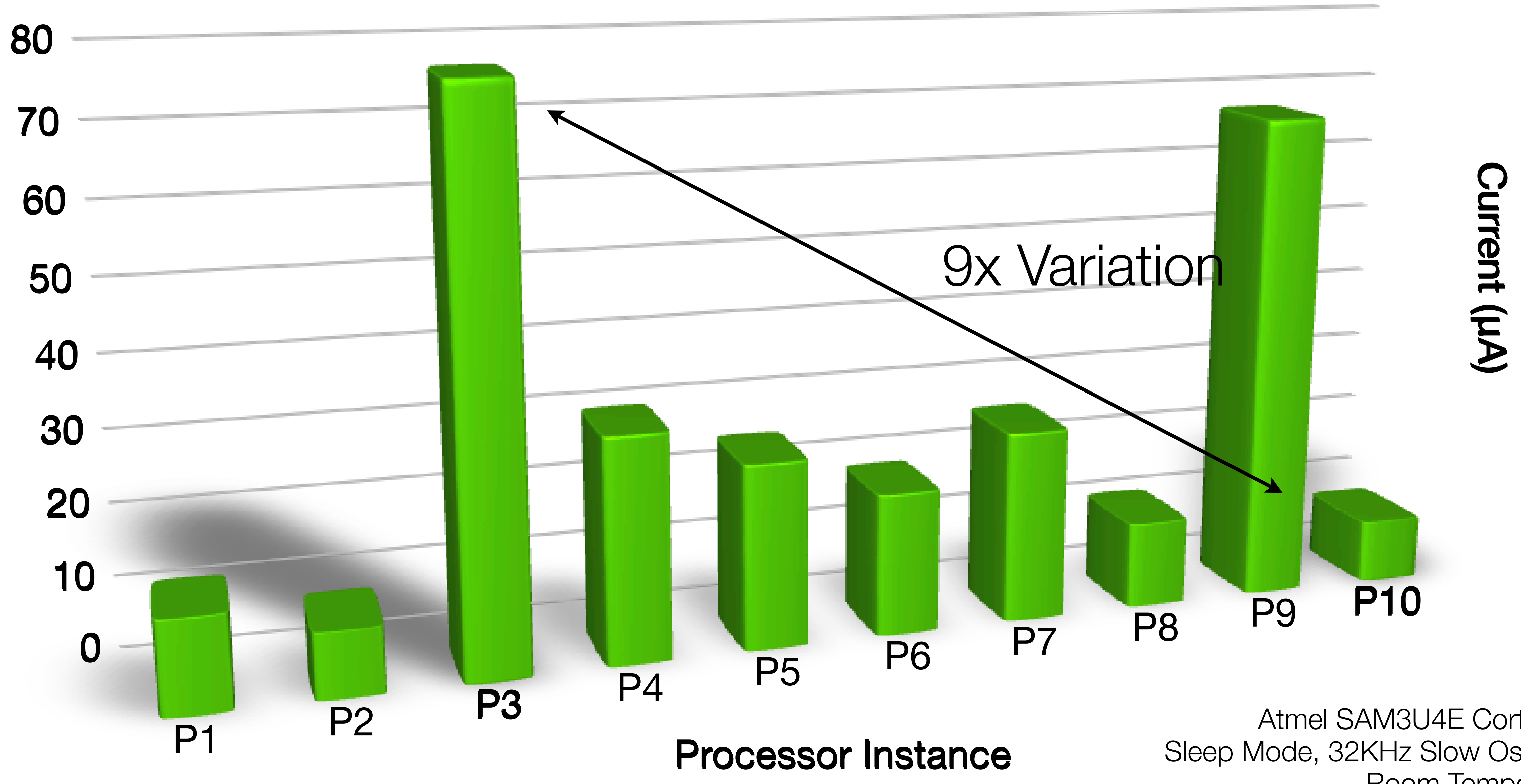


Active Power Variability Across Temperature



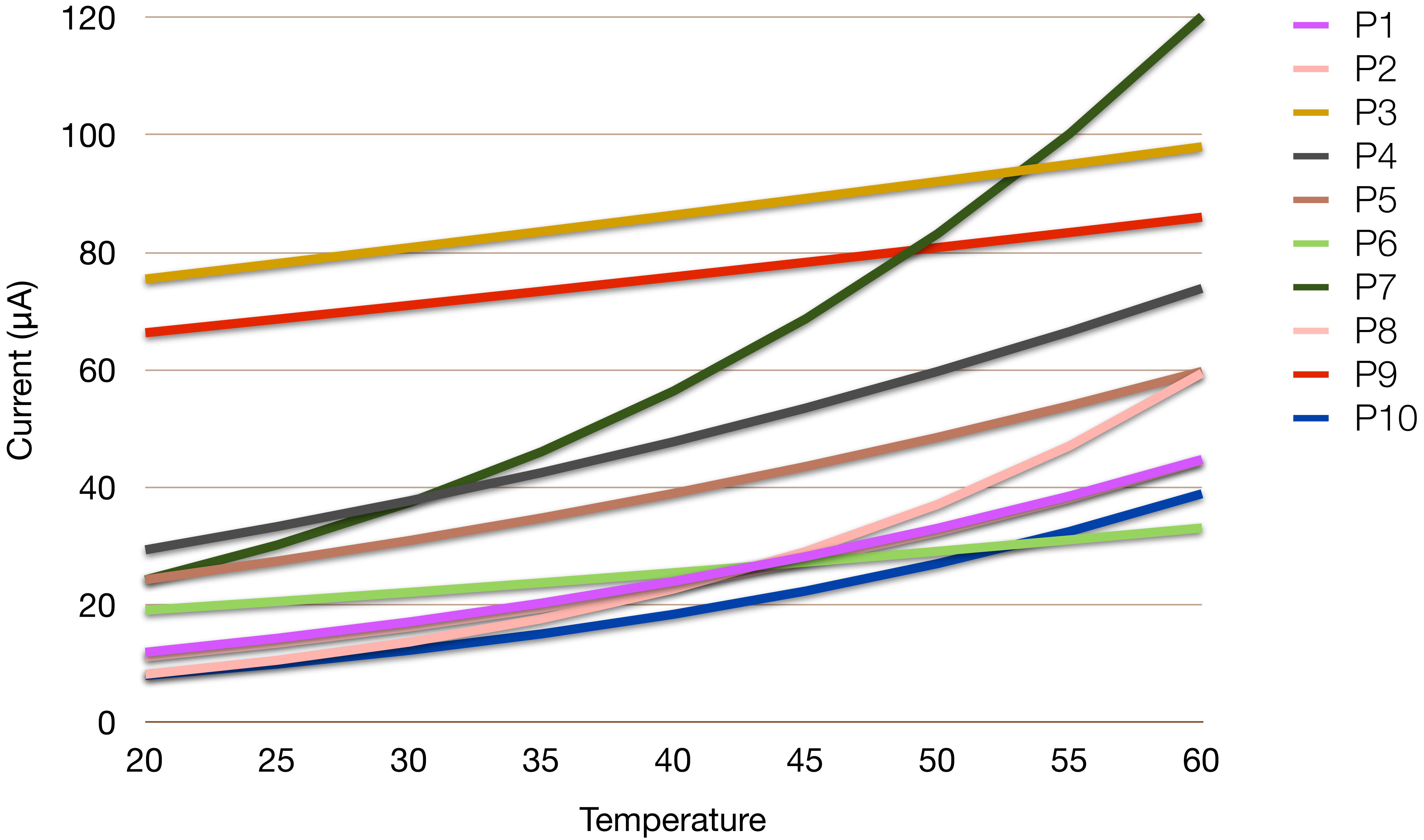
Sleep Power Variability Across Instances

Cortex M3 Sleep Current (Room Temperature)

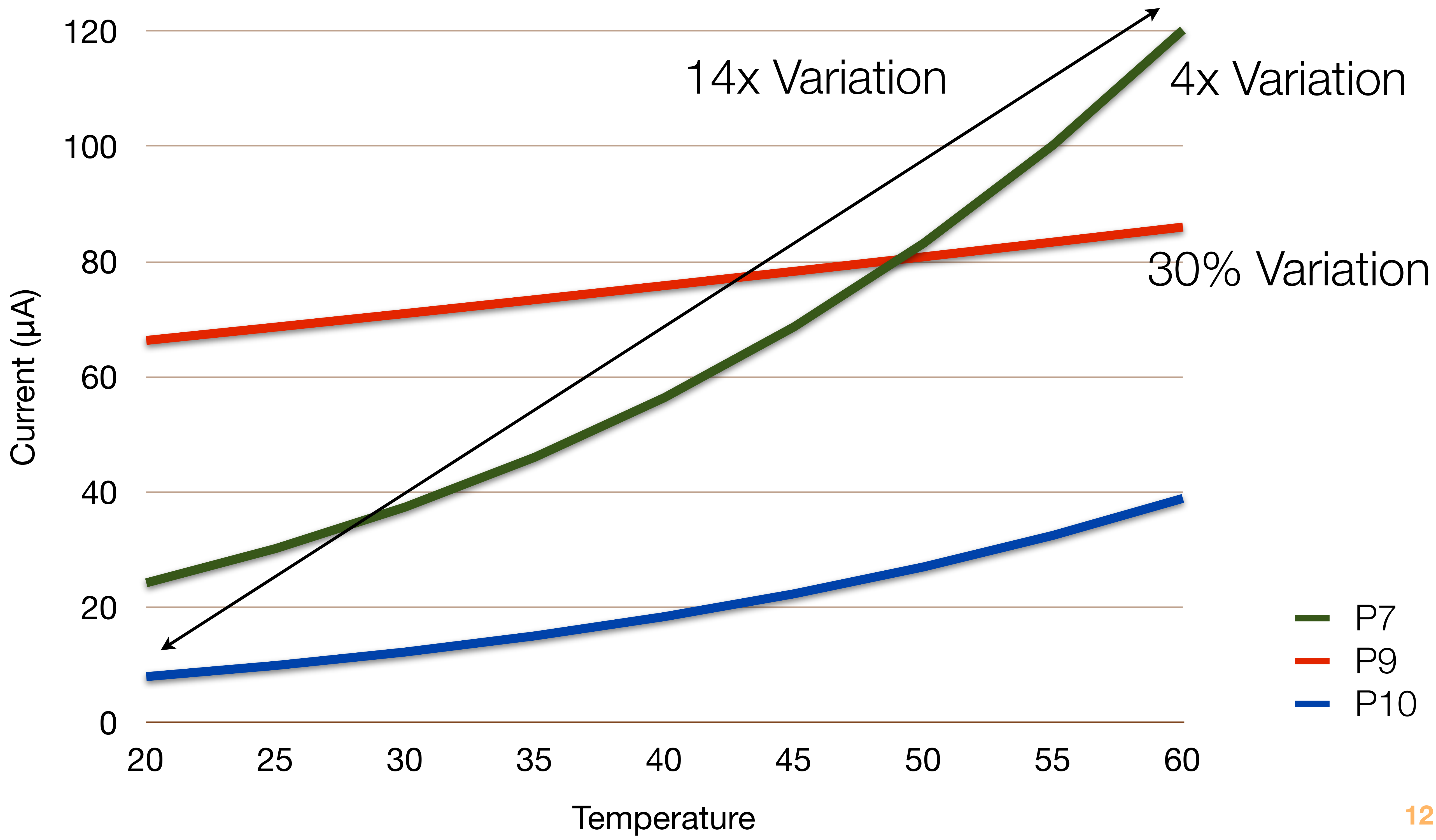


Atmel SAM3U4E Cortex M3
Sleep Mode, 32KHz Slow Oscillator
Room Temperature

Sleep Power Variability Across Temperature

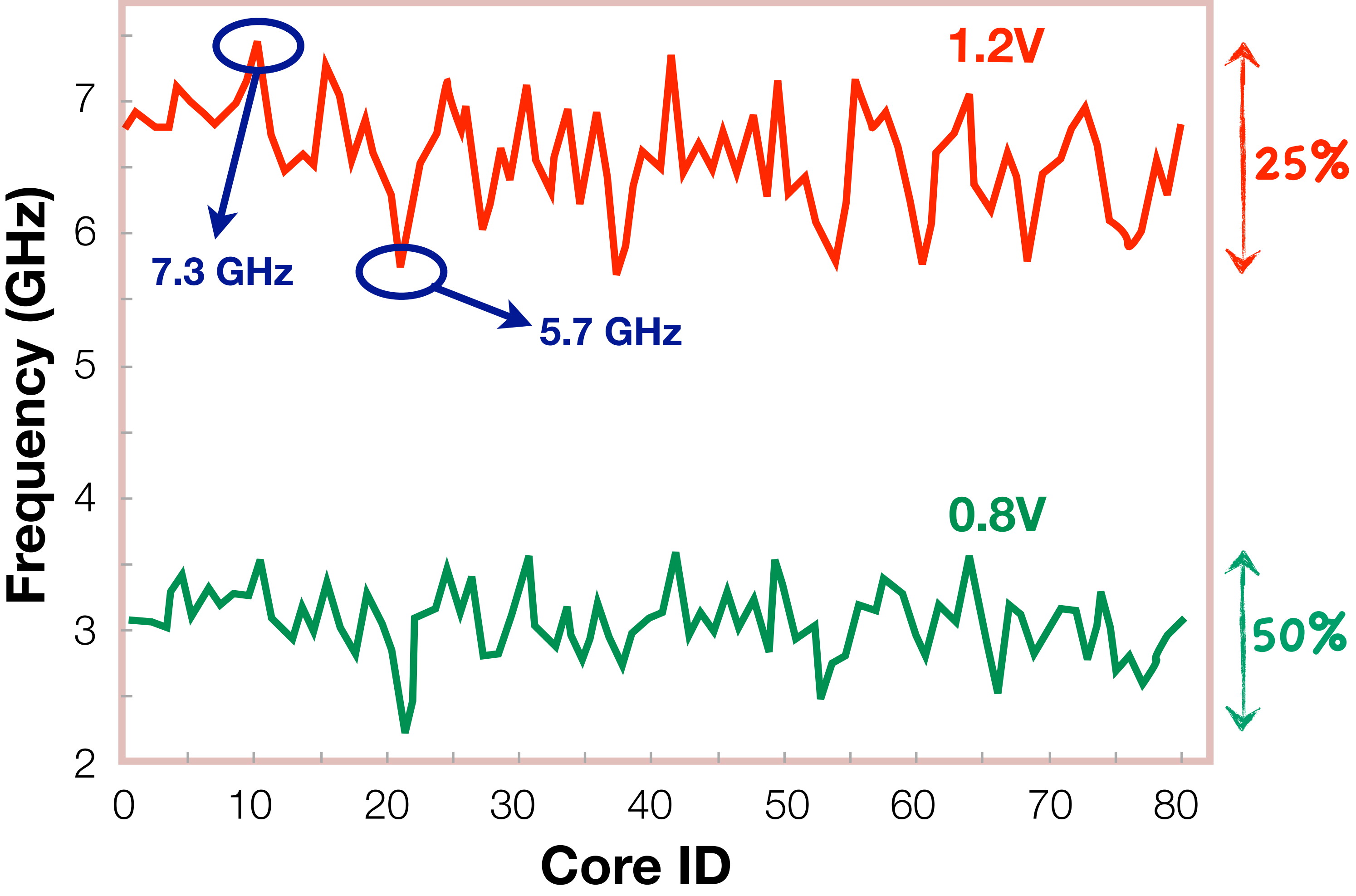


Sleep Power Variability Across Temperature



Source #1: Manufacturing Variability Example

Frequency variation in an 80-core processor within a single die in Intel's 65nm technology [Dighe10]

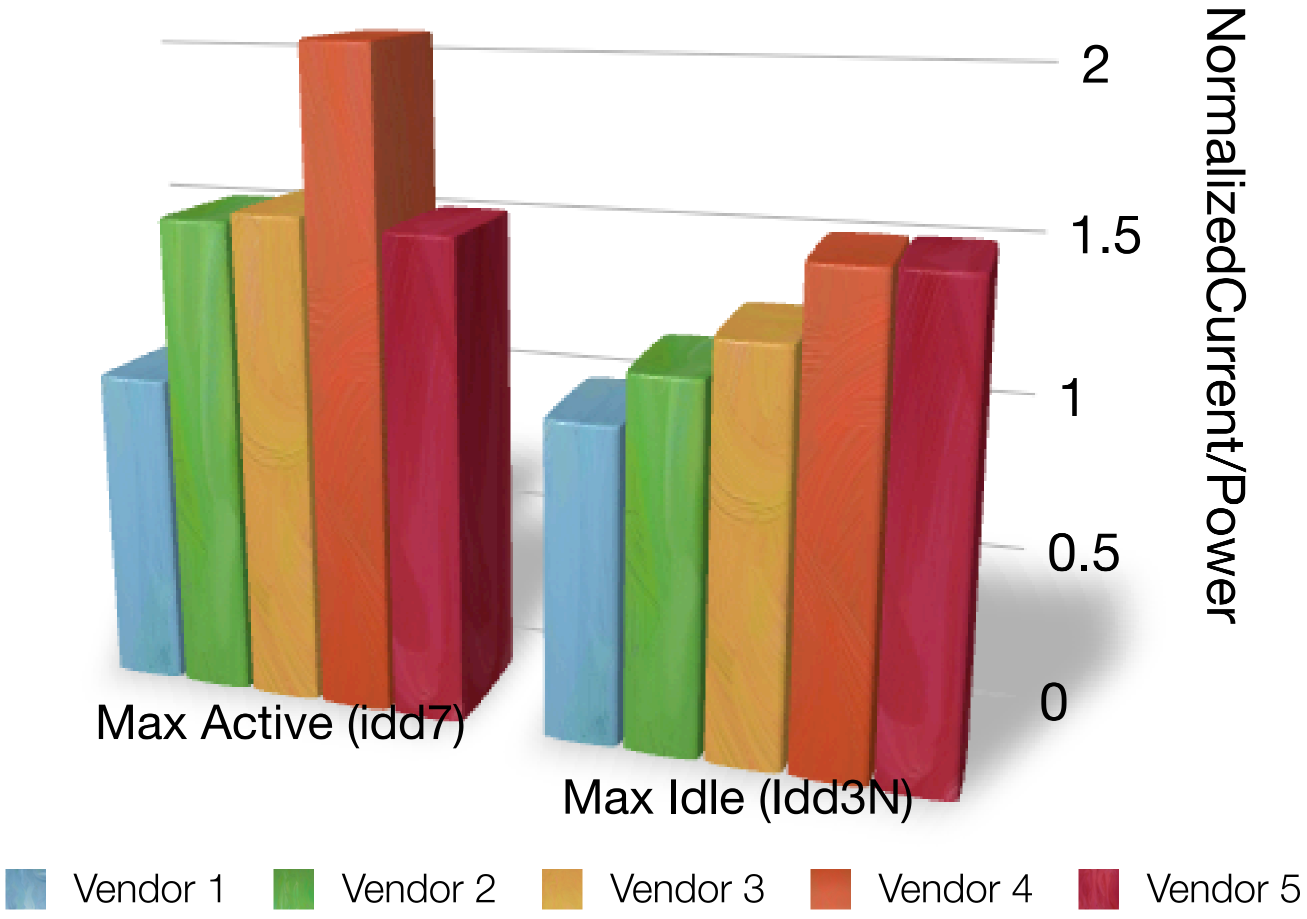


- Observables
  - Maximum speed, energy efficiency
- Mitigation mechanism
  - Computation fidelity

Permanence	Spatial Granularity	Temporal Rapidity	Magnitude
Permanent	Within & across part	Fixed	Large

Source #2: Vendor Variability Example

Power variation across five 512 MB DDR2-533 DRAM parts [Hanson07]

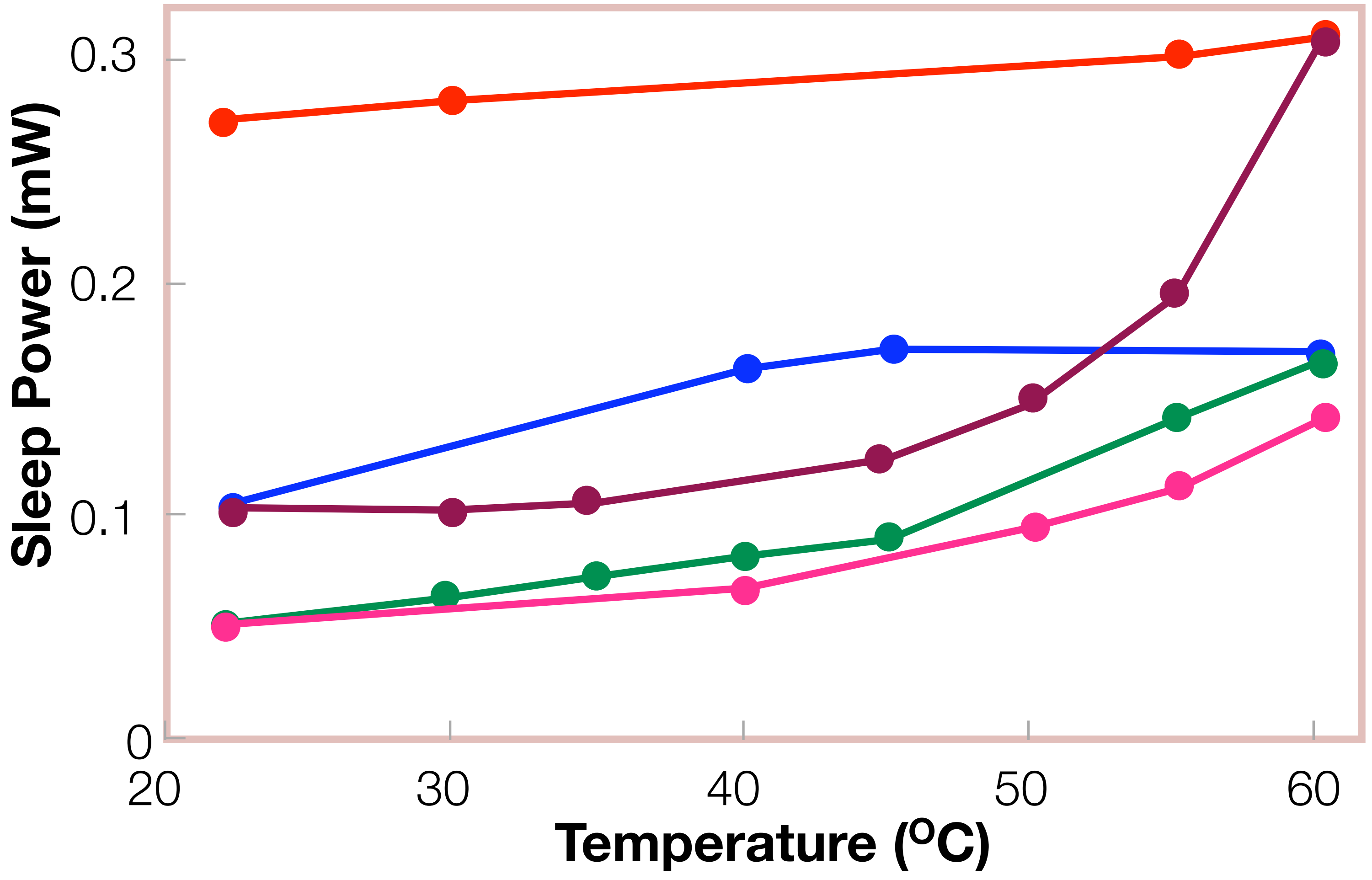


- Observables
  - Relative cost of memory and compute operations
- Mitigation mechanism
  - Algorithm selection

Permanence	Spatial Granularity	Temporal Rapidity	Magnitude
Permanent	Part-to-part	Fixed	Large

Source #3: Ambient Variability Example

Variation in P_{sleep} with temperature across five instances of an ARM Cortex M3 processor

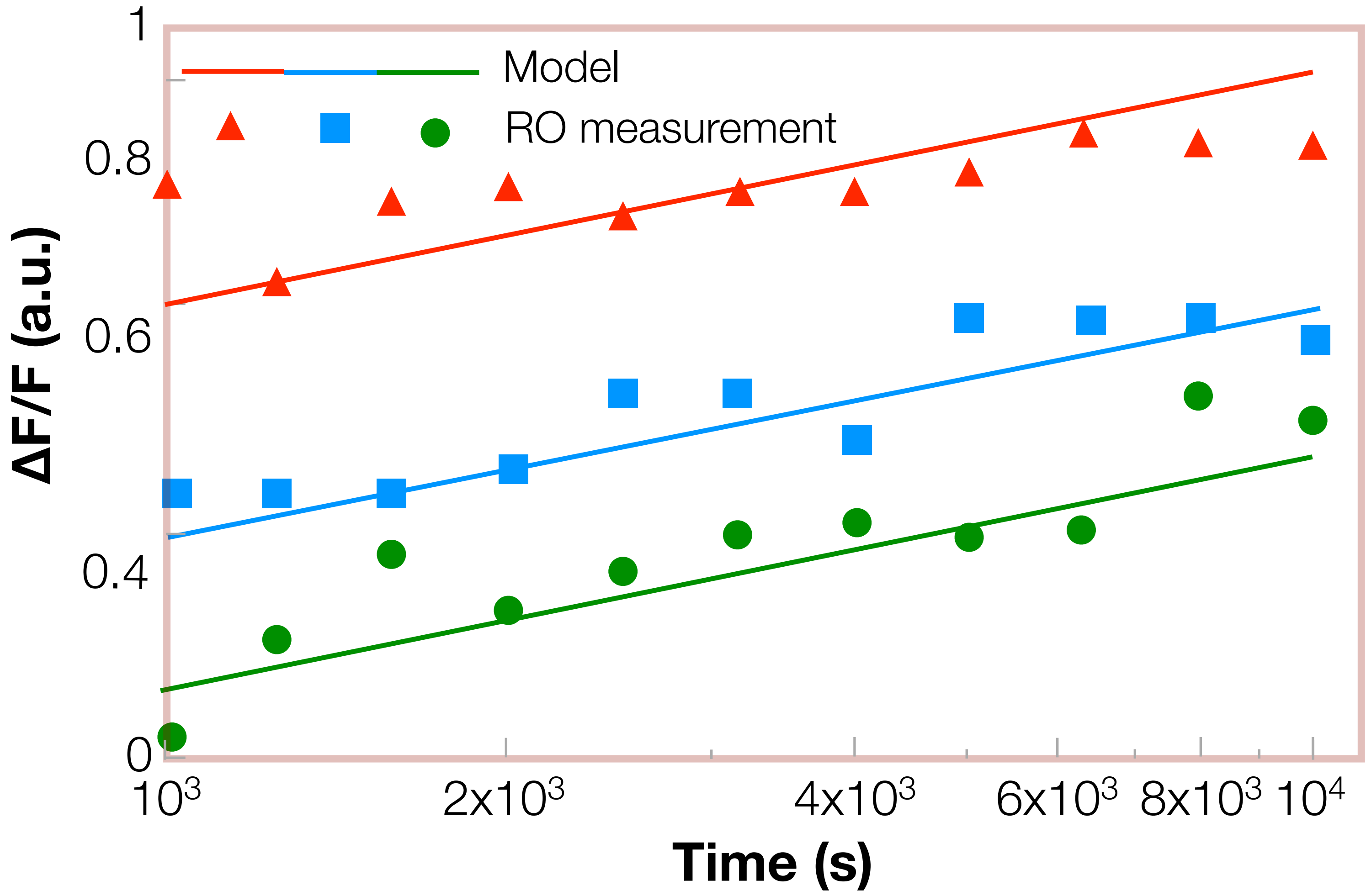


- Observables
  - Sleep mode power
- Mitigation mechanism
  - Adapt duty cycle ratio

Permanence	Spatial Granularity	Temporal Rapidity	Magnitude
Transient	Part-to-part	Medium	Large

Source #4: Aging Example

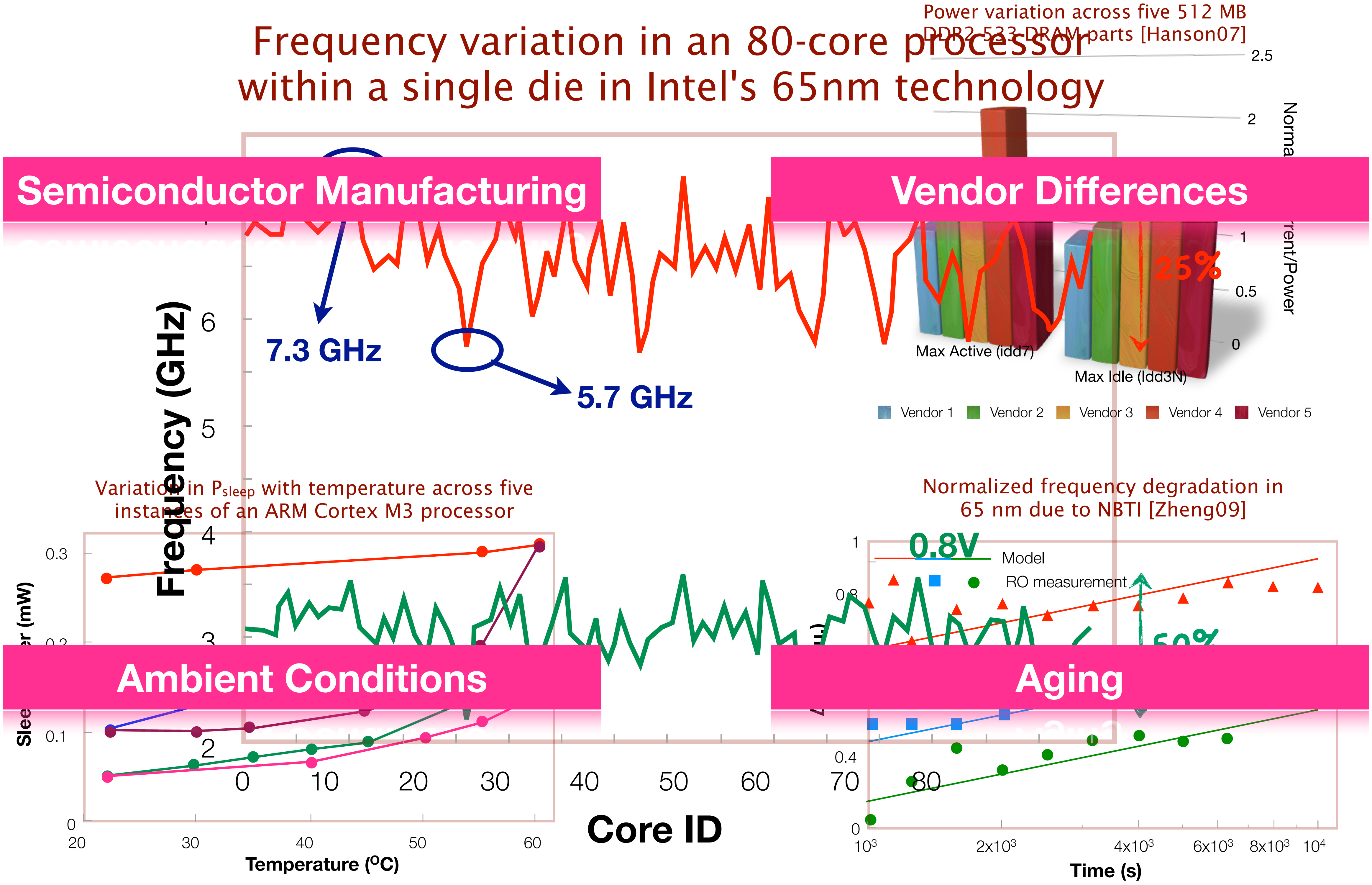
Normalized frequency degradation in 65 nm due to NBTI [Zheng09]



- Observables
  - Speed degradation, increased error
- Mitigation mechanism
  - Computation elasticity

Permanence	Spatial Granularity	Temporal Rapidity	Magnitude
Permanent	Within & across part	Slow	Medium

Sources of Variability



Frequency variation in an 80-core processor within a single die in Intel's 65nm technology

Power variation across five 512 MB DDR2-533 DRAM parts [Hanson07]

Variation in P_{sleep} with temperature across five instances of an ARM Cortex M3 processor

Normalized frequency degradation in 65 nm due to NBTI [Zheng09]

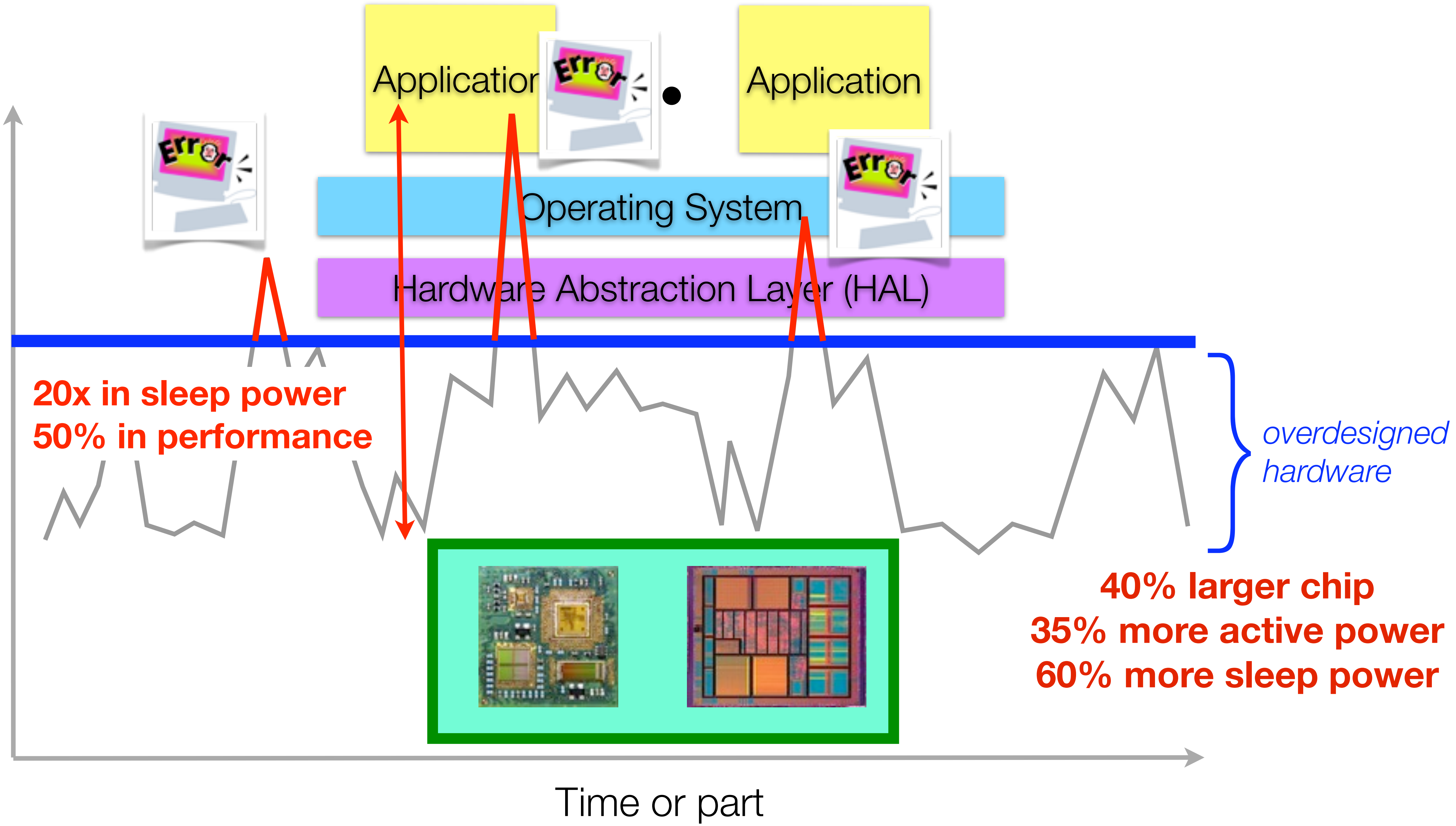
Semiconductor Manufacturing

Vendor Differences

Ambient Conditions

Aging

Let us take another look at the HW/SW stack



Application

Application

Operating System

Hardware Abstraction Layer (HAL)

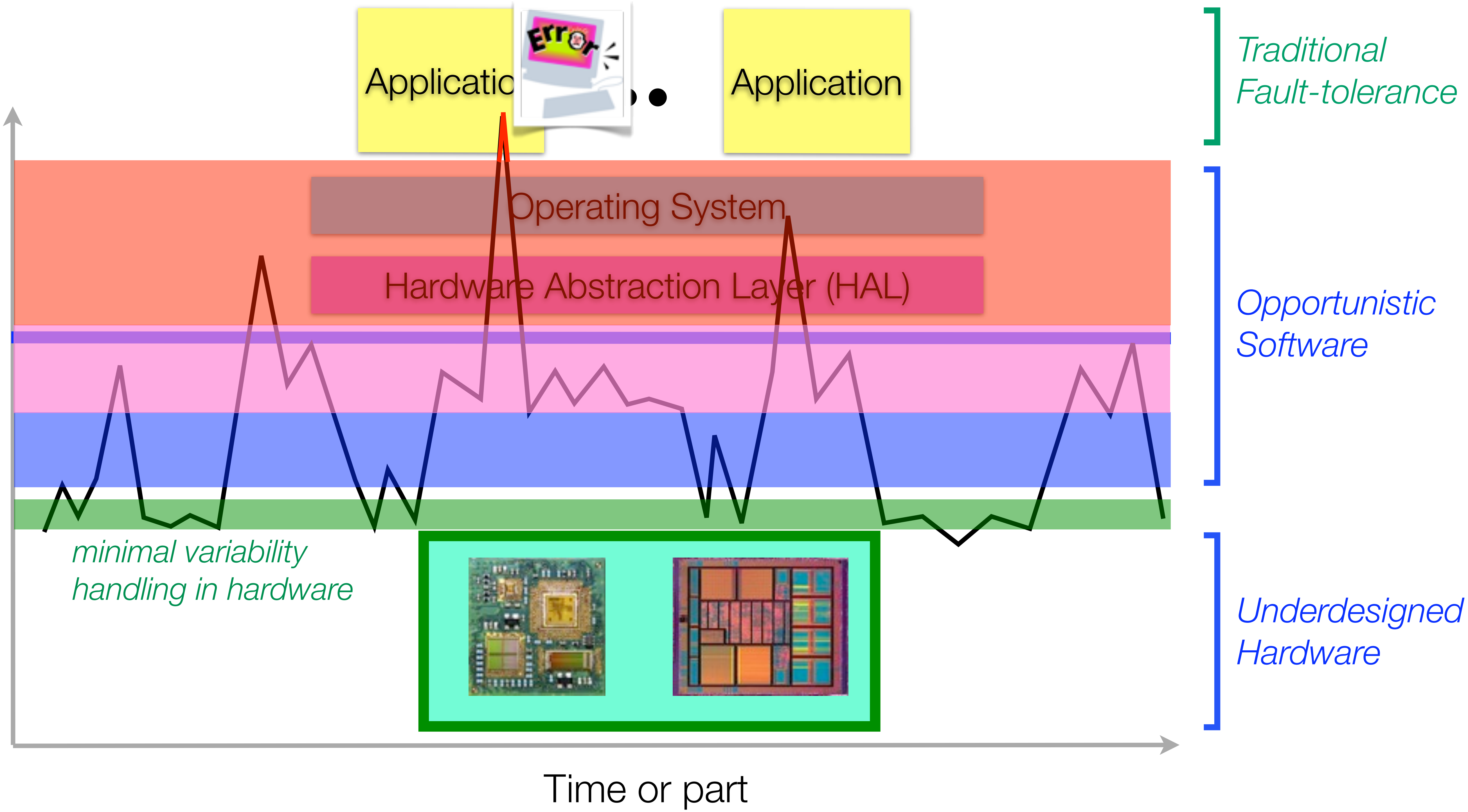
20x in sleep power
50% in performance

overdesigned hardware

40% larger chip
35% more active power
60% more sleep power

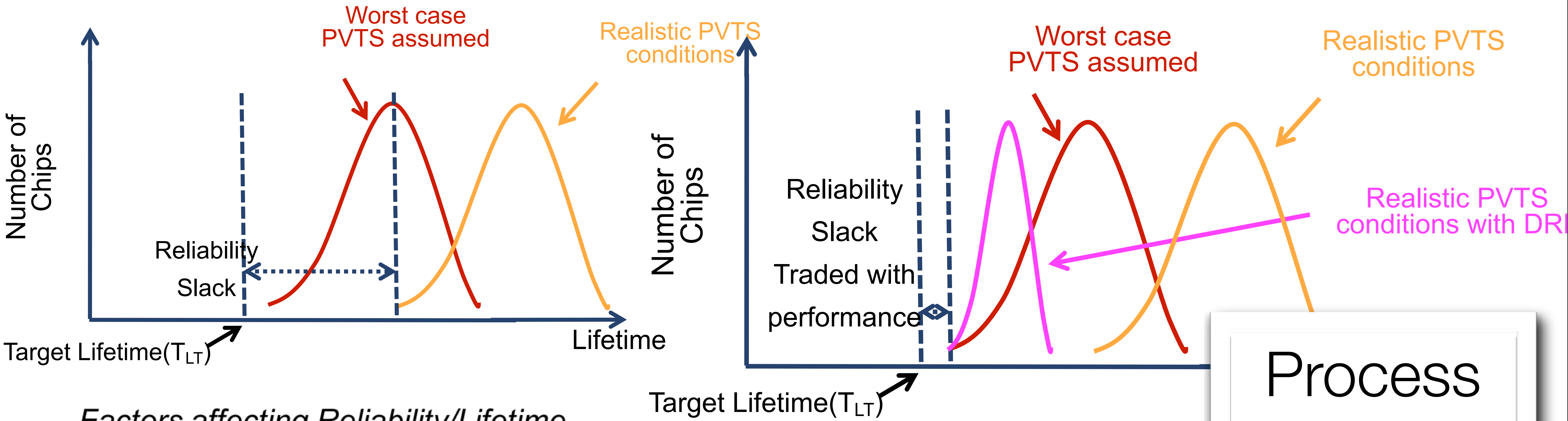
Time or part

Imagine a new hardware-software interface...



Hardware: Self-monitoring as opposed to self-healing

- Measure hardware signatures, use fluid constraints in HW design, error possibility in operation using simple device monitors
- Static and Dynamic Reliability Management



Factors affecting Reliability/Lifetime

- Inherent randomness
- Process (P)
- Voltage (V)
- Temperature (T)
- State (S)

Dynamic Reliability Management (DRI

Reliability Slack with Perforn

- Boost supply Voltage
- Allow higher temperature of operation
- No chip fails way after T_{LT}
- No chip fails before T_{LT}

Process
Circuit
Functional
System

Unified NBTI /Oxide degradation sensor

Process Node

45nm CMOS

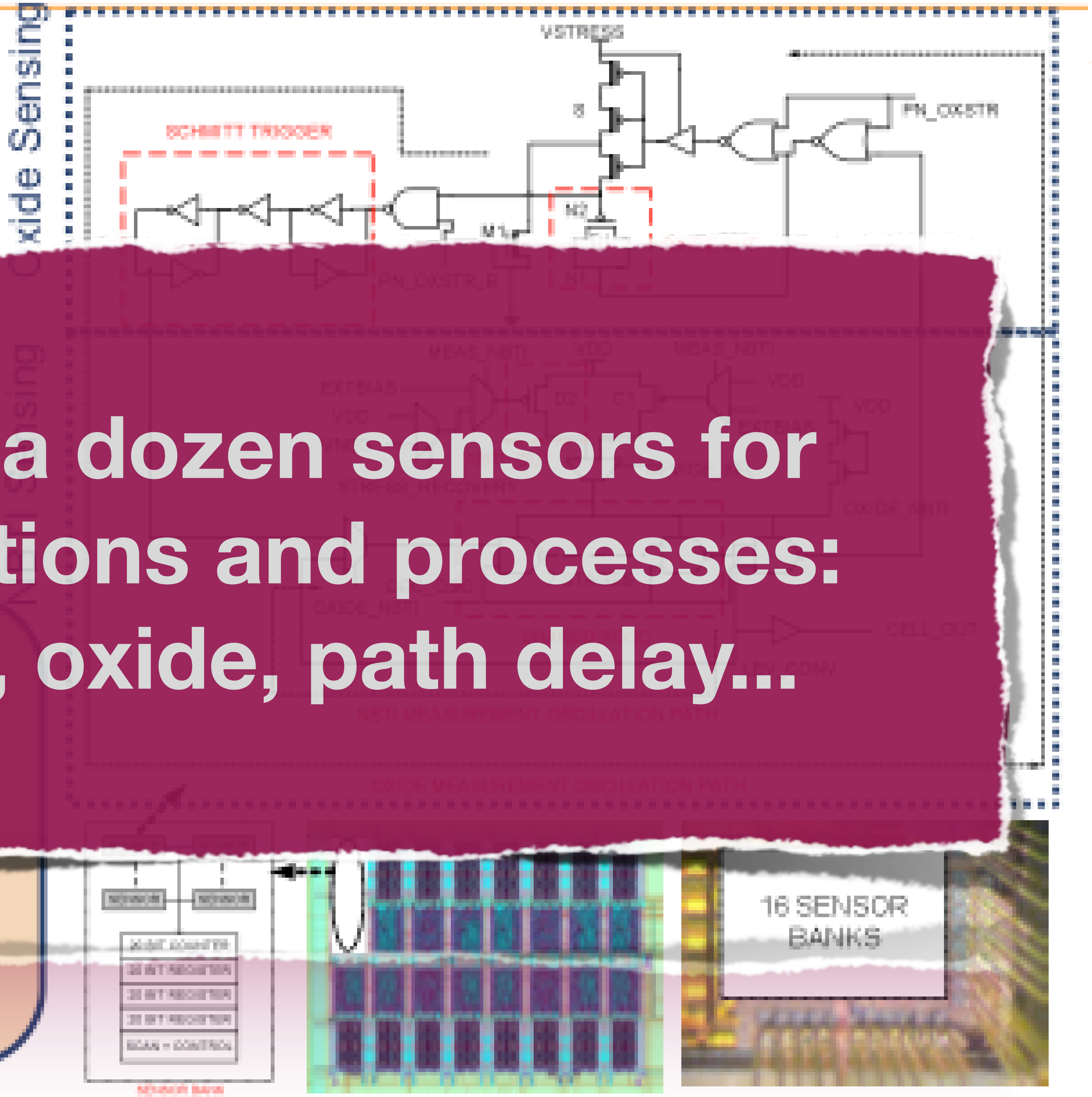
Power

10^5 times lower than prior work

Area

6 Flip-flops

- Ring oscillator-based
- Bias NBTI device in subthreshold to magnify ΔVth
- Gate-connected device used to monitor increase in oxide conductivity due to stress



Half a dozen sensors for variations and processes: nbti, oxide, path delay...

University of Michigan

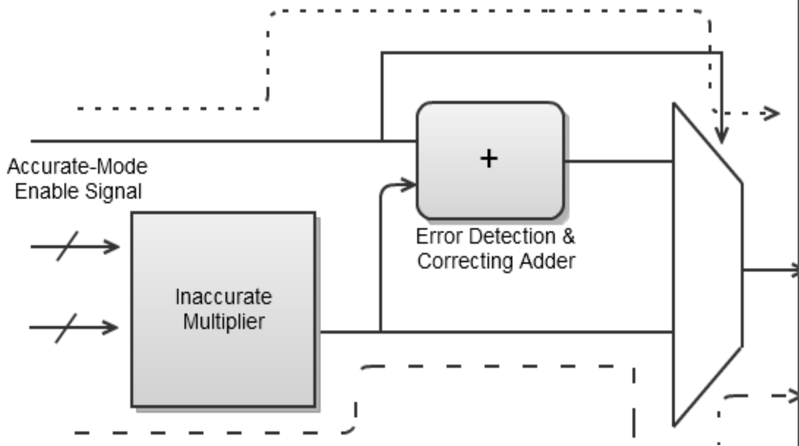
An Underdesigned Multiplier

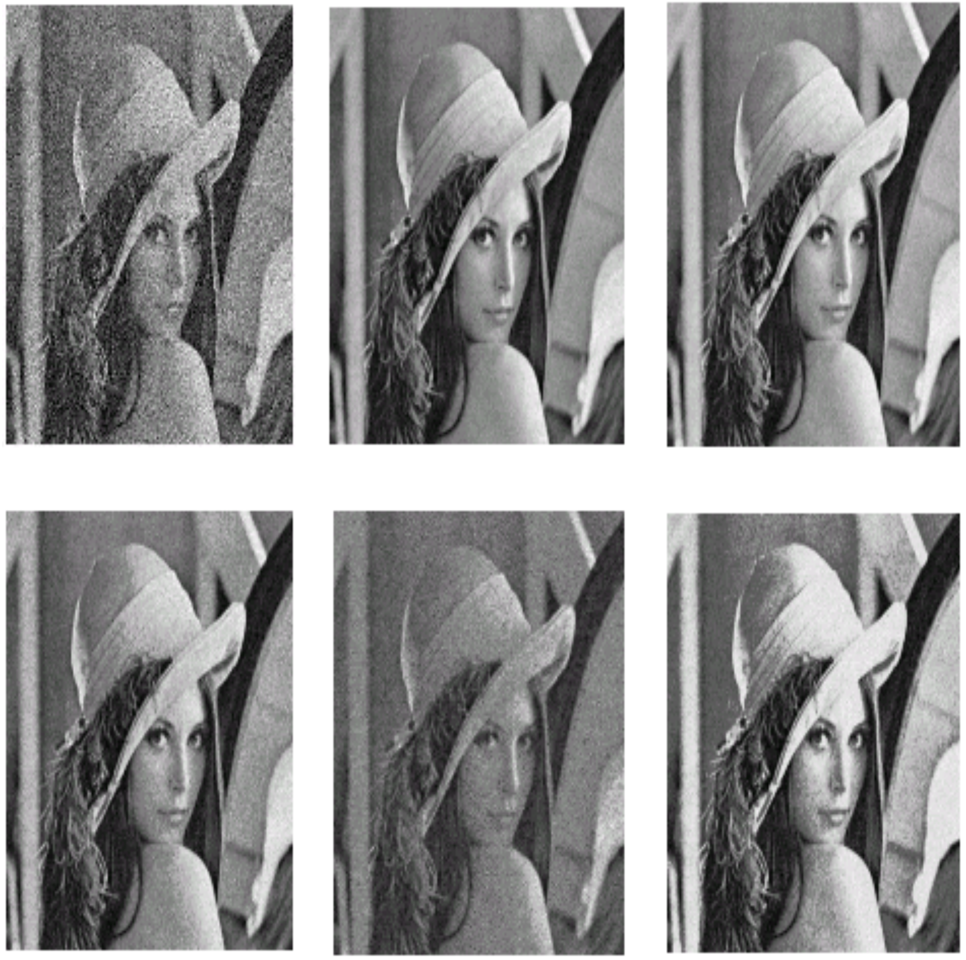
- Idea: change functional description of arithmetic units instead of voltage overscaling
- Basic building block: 2x2 multiplier
  - Computes 11 x 11 = 111 (not 1001)
  - Scalable to arbitrary bit widths by adding partial products
  - ~40% power reduction but ~8% power overhead in correct mode
  - Average error ~3.3%, max error ~22.2%
- Comparison with voltage overscaling (image filtering)

  a) **Inaccurate multiplier, 41.5% power reduction, SNR : 20.3dB**

  b) Voltage over-scaling, 30% power reduction, SNR : 9.16dB

  c) Voltage over-scaling 50% power reduction, SNR : 2.64dB


Accurate-Mode Enable Signal
+
Error Detection & Correcting Adder
Inaccurate Multiplier



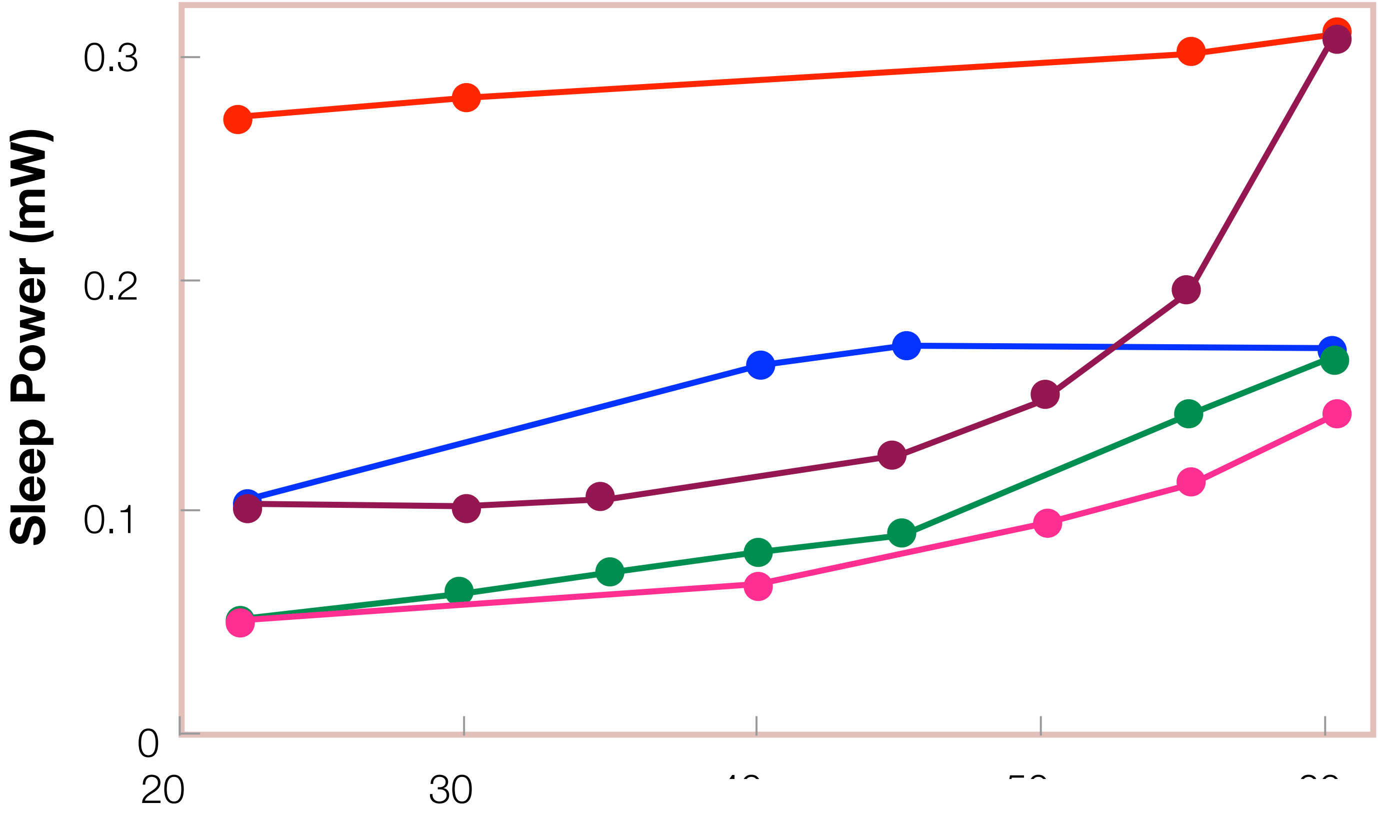
Provides ability to do designs with tunable error characteristics.

Puneet Gupta, UCLA

Variability-aware Duty-cycling

$$\text{Duty Cycle} = f(\mathbf{P}_{\mathbf{sleep}})$$

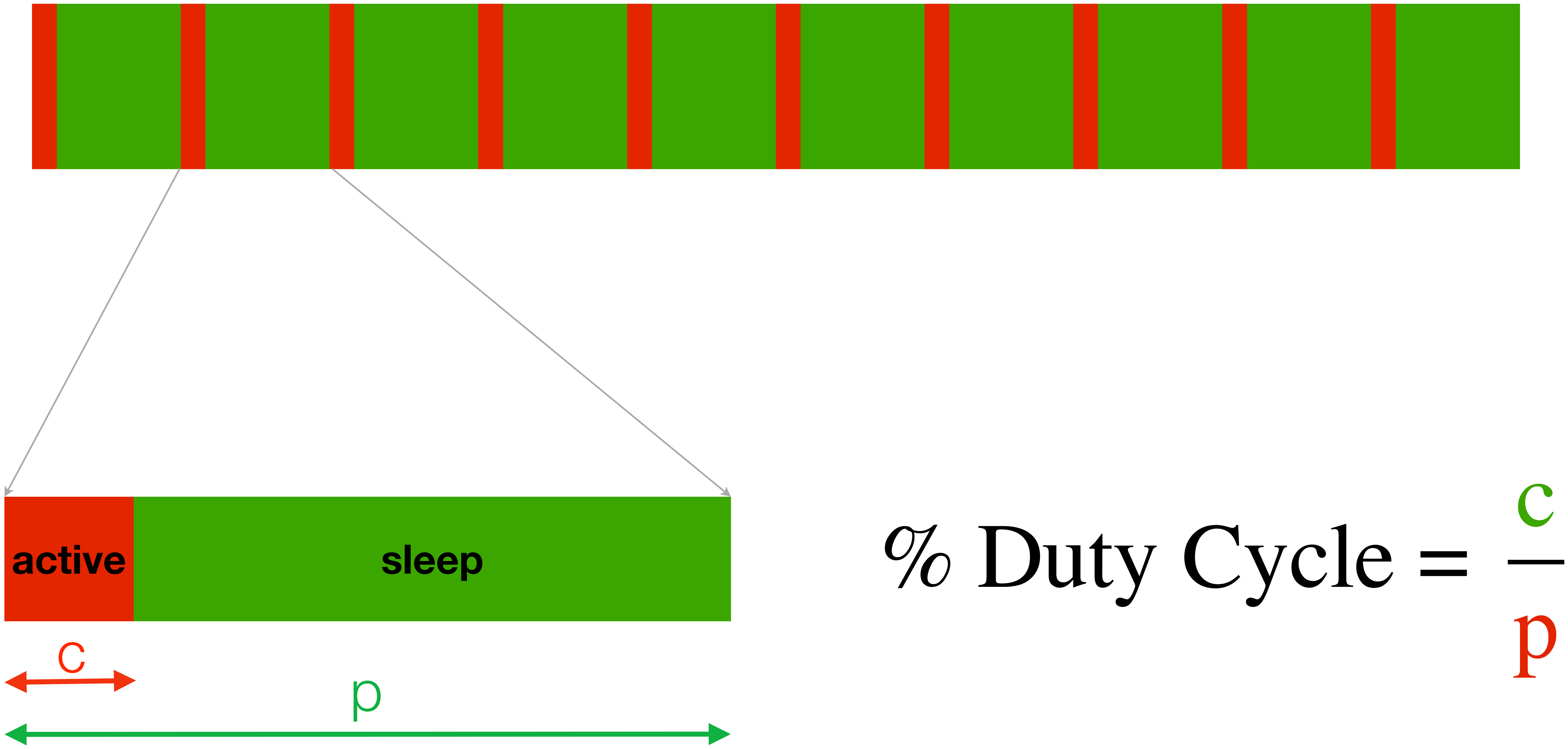
Atmel's ARM Cortex M3-based SAM3U Embedded Processor



Mani Srivastava,

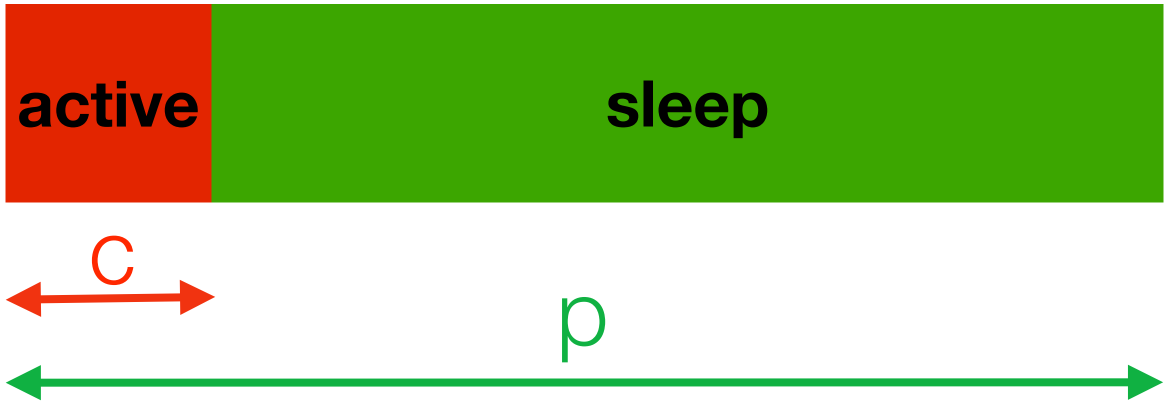
Puneet Gupta, UCLA

Duty-cycled Wireless Sensors



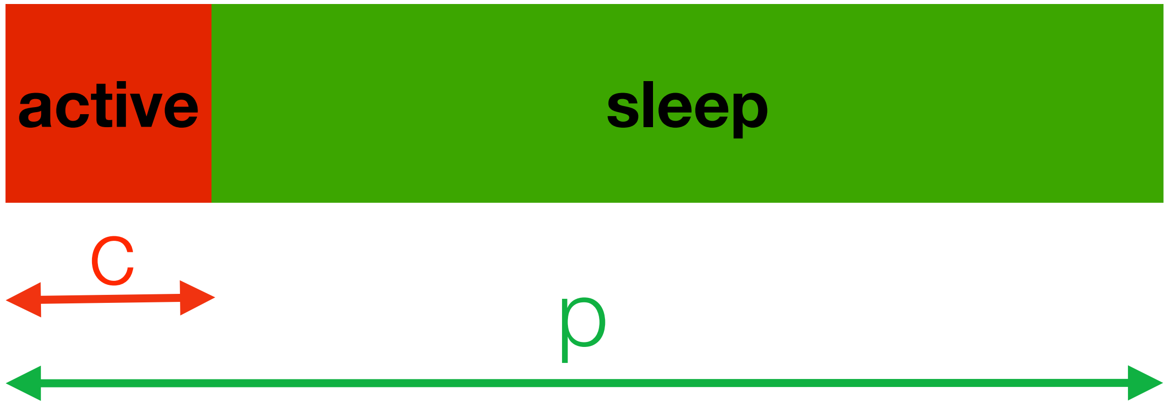
$$\% \text{ Duty Cycle} = \frac{c}{p}$$

Duty-cycled Wireless Sensors



$$\% \text{ Duty Cycle} = \frac{c}{p}$$

Duty-cycled Wireless Sensors



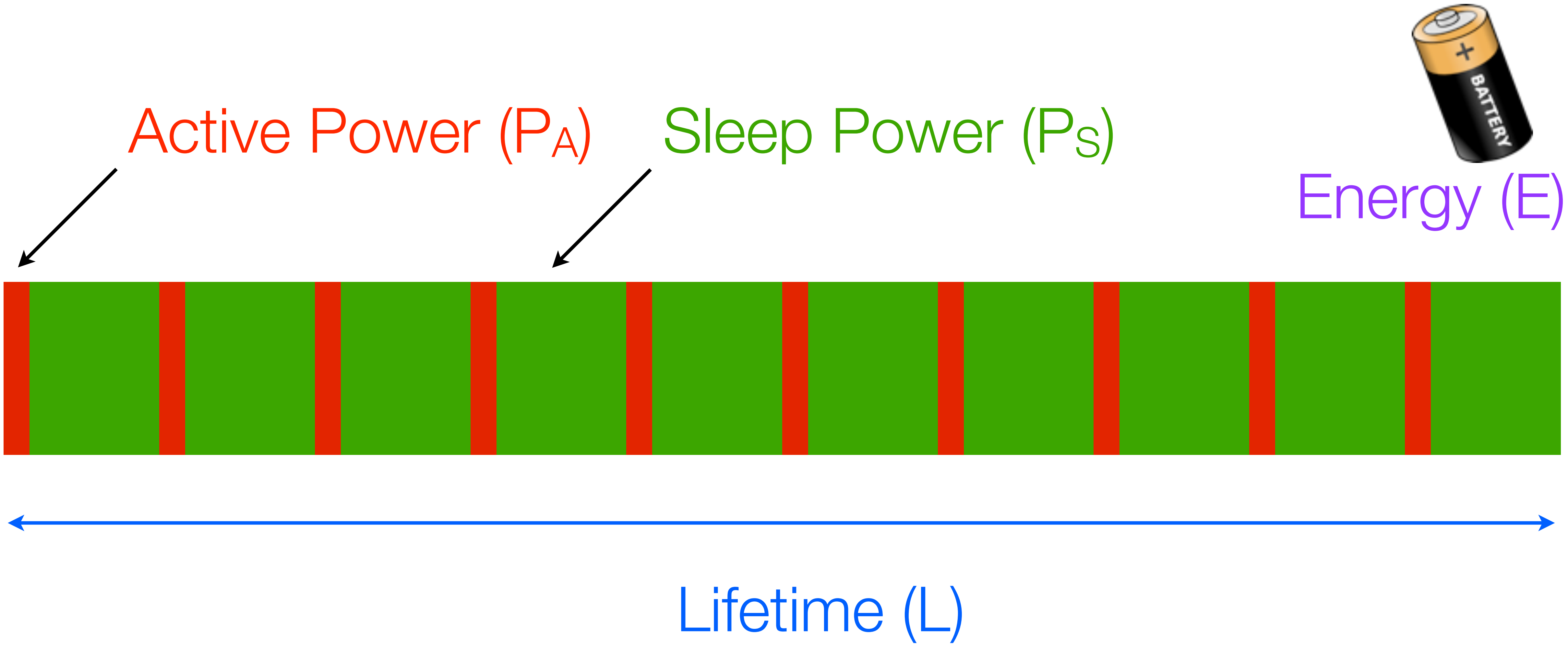
$$\% \text{ Duty Cycle} = \frac{c}{p}$$

↑% Duty Cycle ⇒ ↑Quality of Sensing

c↑, p↓

P(event detection)
# of data samples
Classification accuracy
...

Feasible Duty Cycle



Active Power (P_A)

Sleep Power (P_S)

Energy (E)

Lifetime (L)

$$\langle c,p \rangle = f\,(P_A, P_S, E, L, QoS)$$

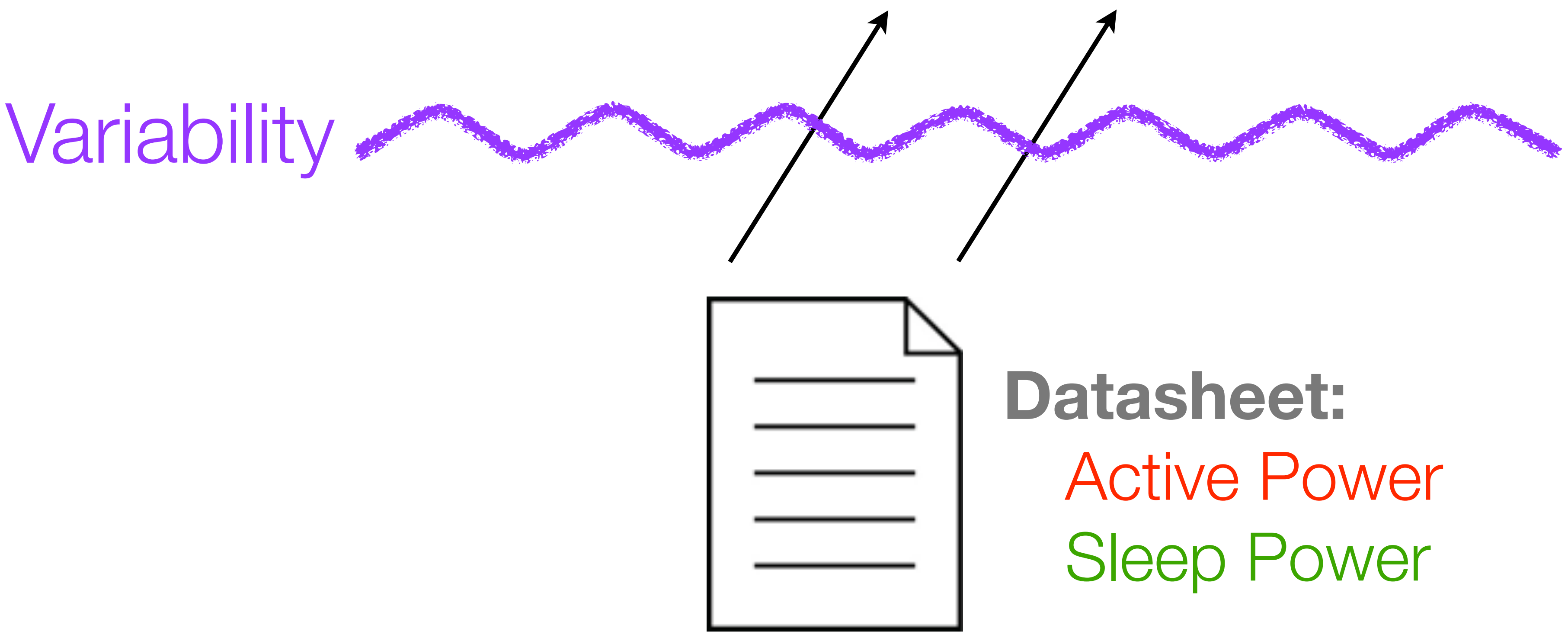
Note: transition time and power ignored here

Feasible Duty Cycle

$$<c,p> = f\ (P_A, P_S, E, L, QoS)$$

Feasible Duty Cycle

$$<c,p> = f\,(P_A,\ P_S,\ E,\ L,\ QoS)$$

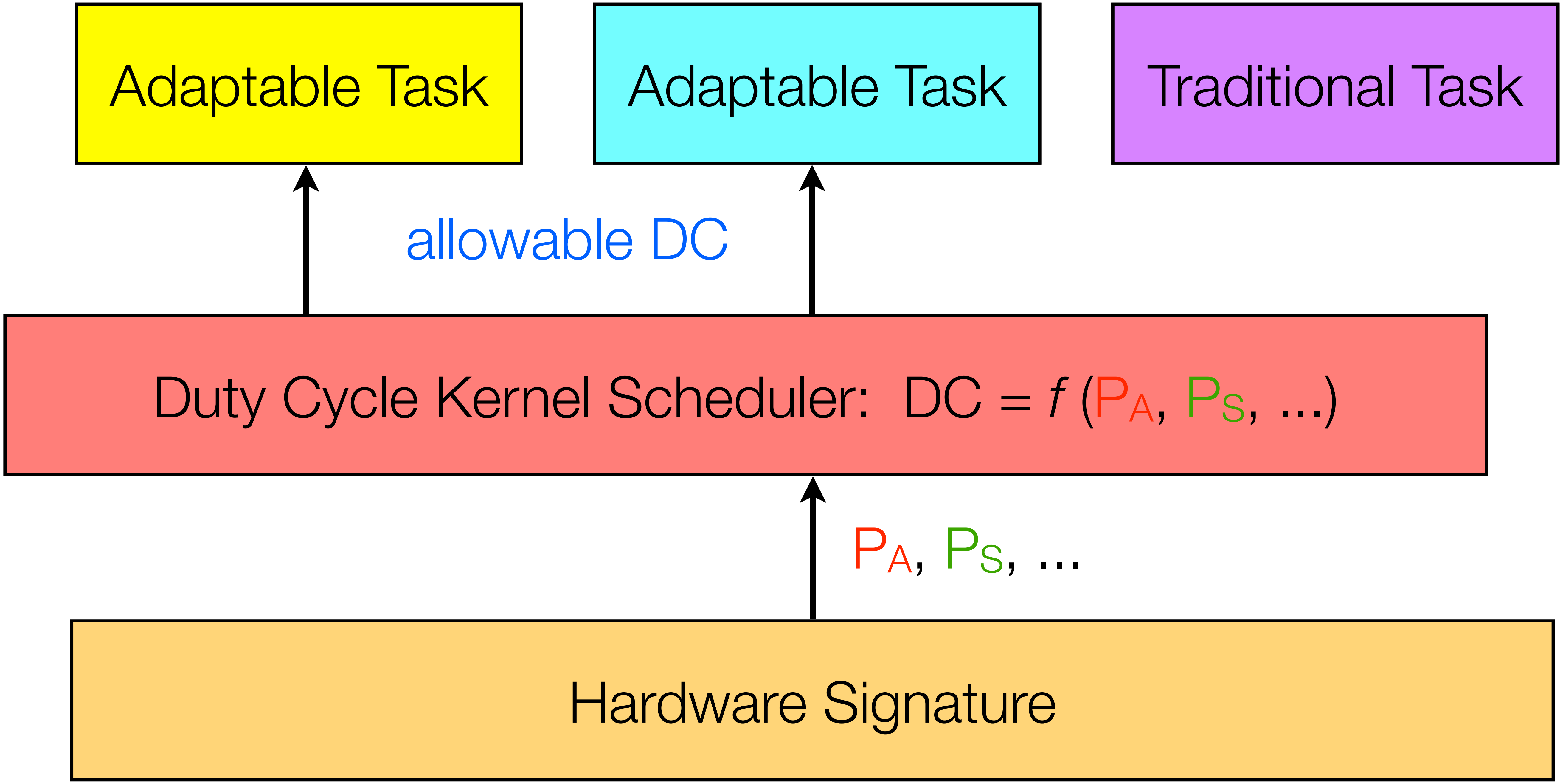


Variability

Datasheet:
Active Power
Sleep Power

Adapt duty cycle when P_A, P_S vary with instance and temperature.

Adaptable Duty Cycled Tasks in TinyOS



Adaptable Task

Adaptable Task

Traditional Task

allowable DC

Duty Cycle Kernel Scheduler: DC = f (P_A, P_S, ...)

P_A, P_S, ...

Hardware Signature

Hardware Variability Signatures

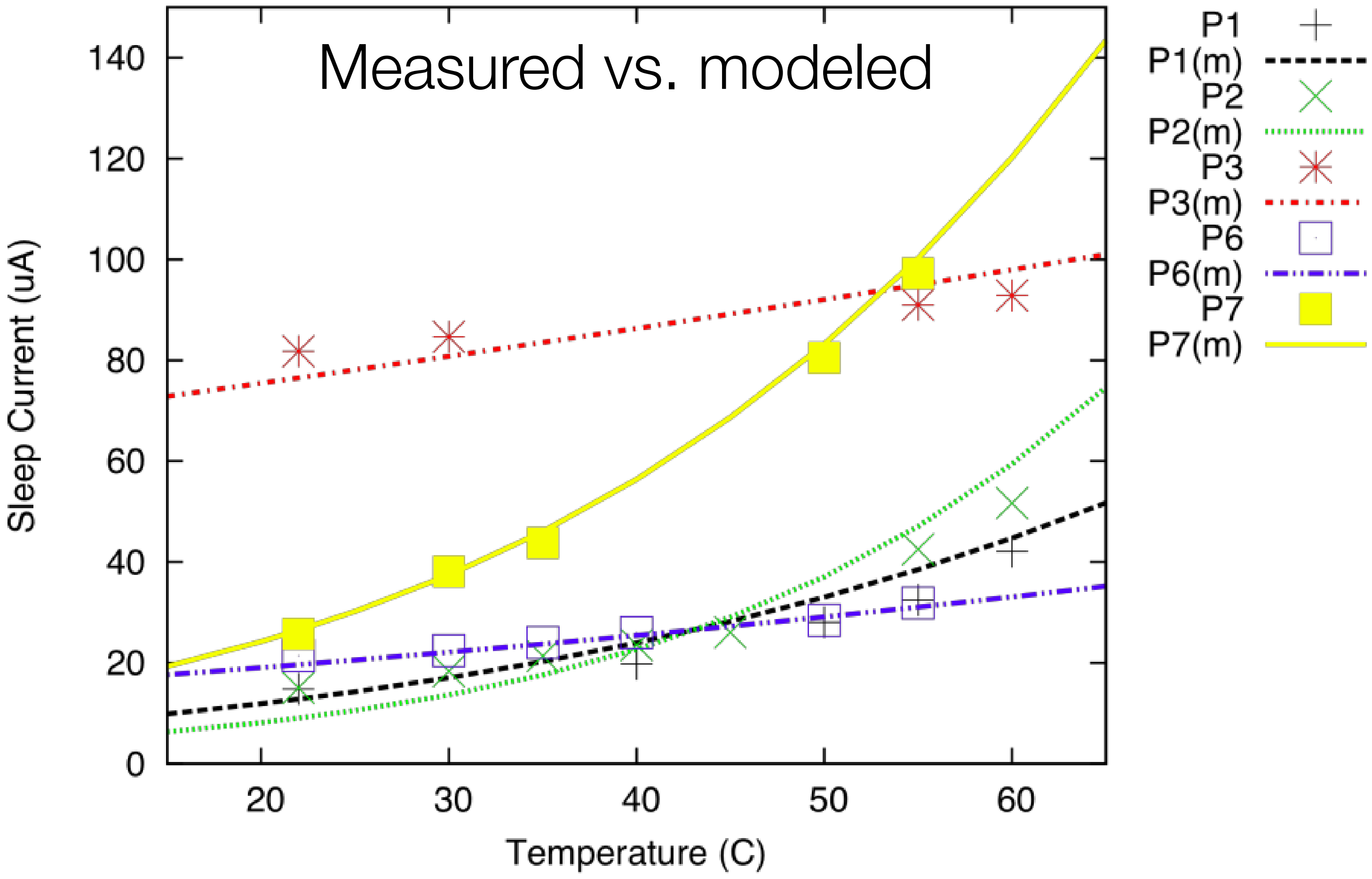
Analytic modeling of sleep power

$$P_{sleep} = V_{dd}(AT^2 e^{B/T} + I_{gl})$$

A and B are technology-dependent constants

I_{gl} is the temperature-independent gate leakage current

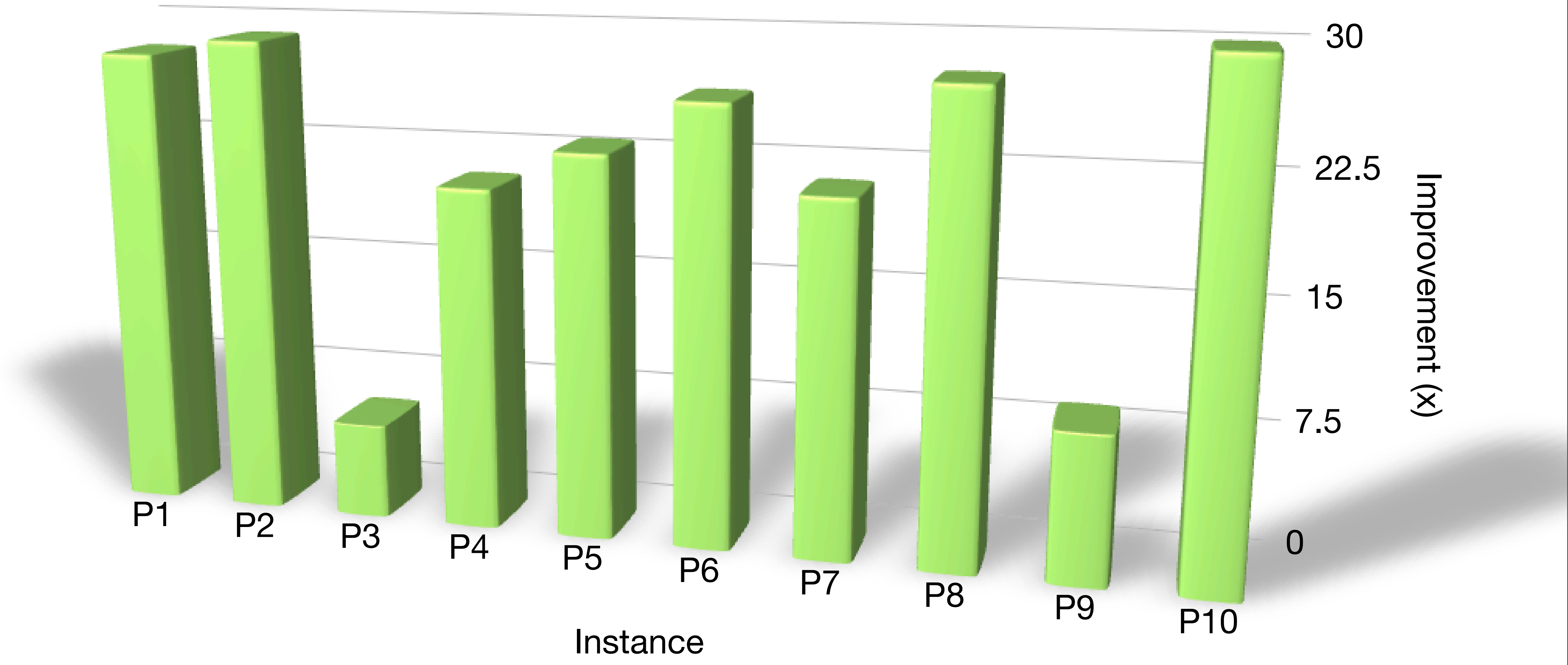
T is the core temperature.



- Parameters of calibrated models are the hardware variability signatures passed to the software stack

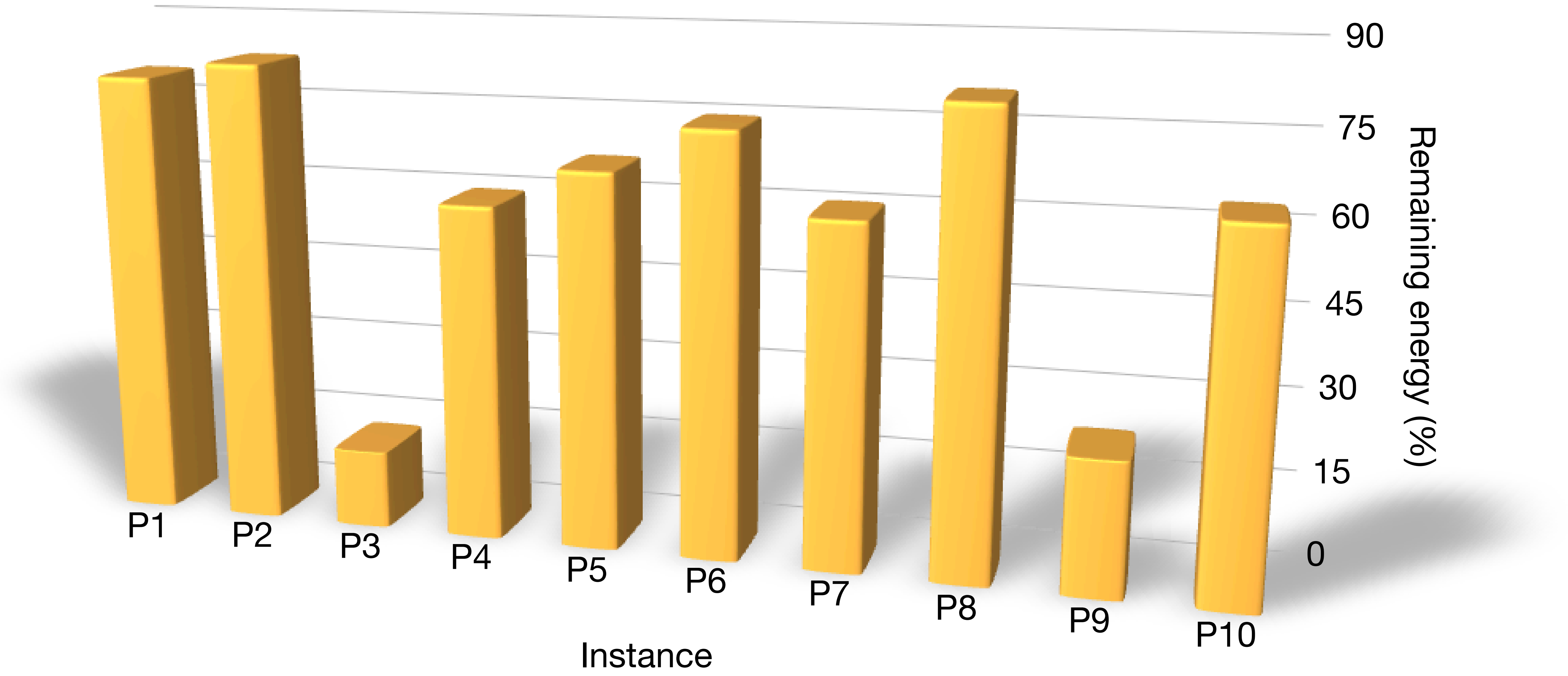
Improvement over Worst-Case Duty Cycle

Average: 22x improvement



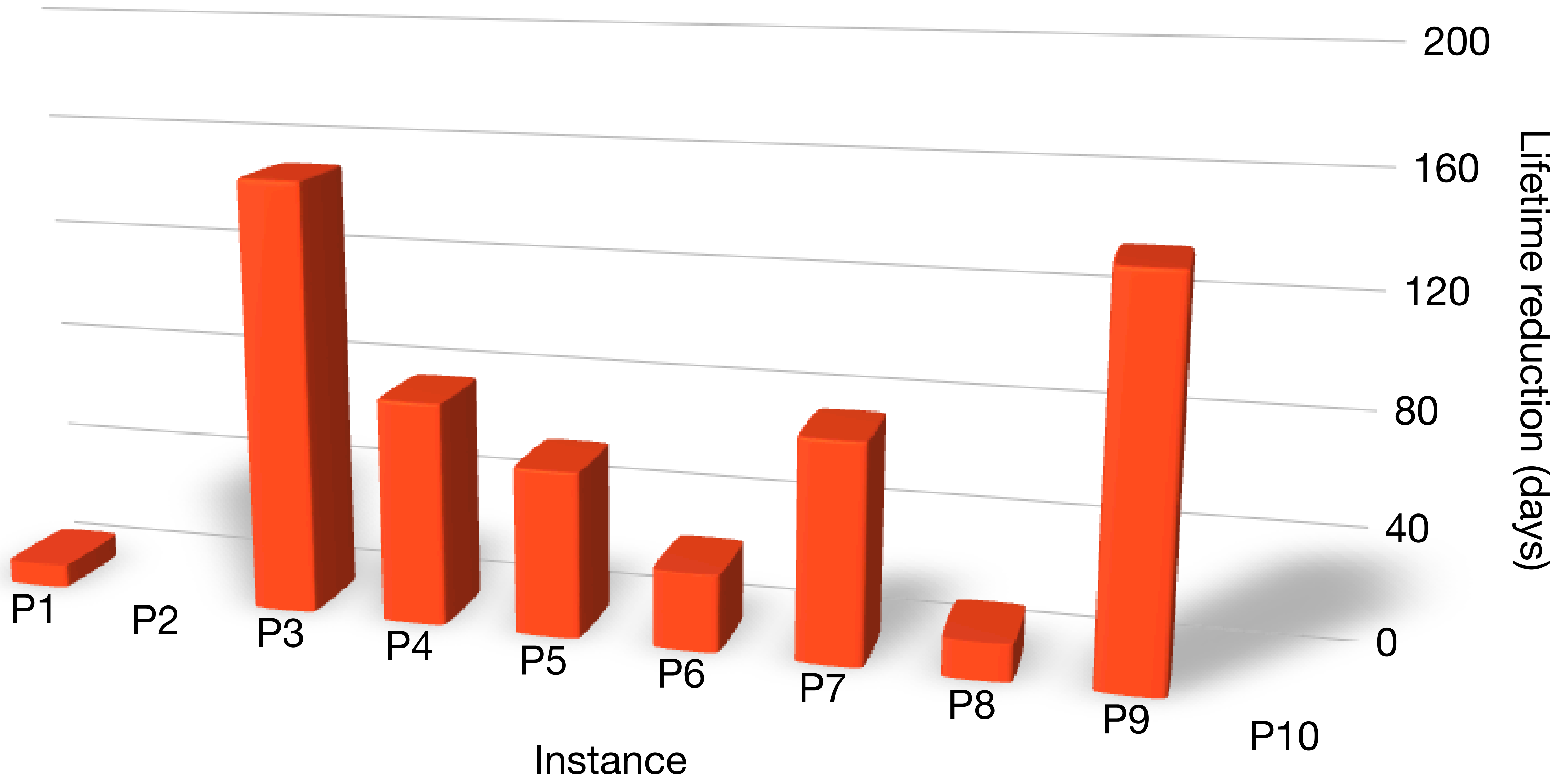
Energy Untapped by Worst-Case Duty Cycle

Average: 63% energy left untapped

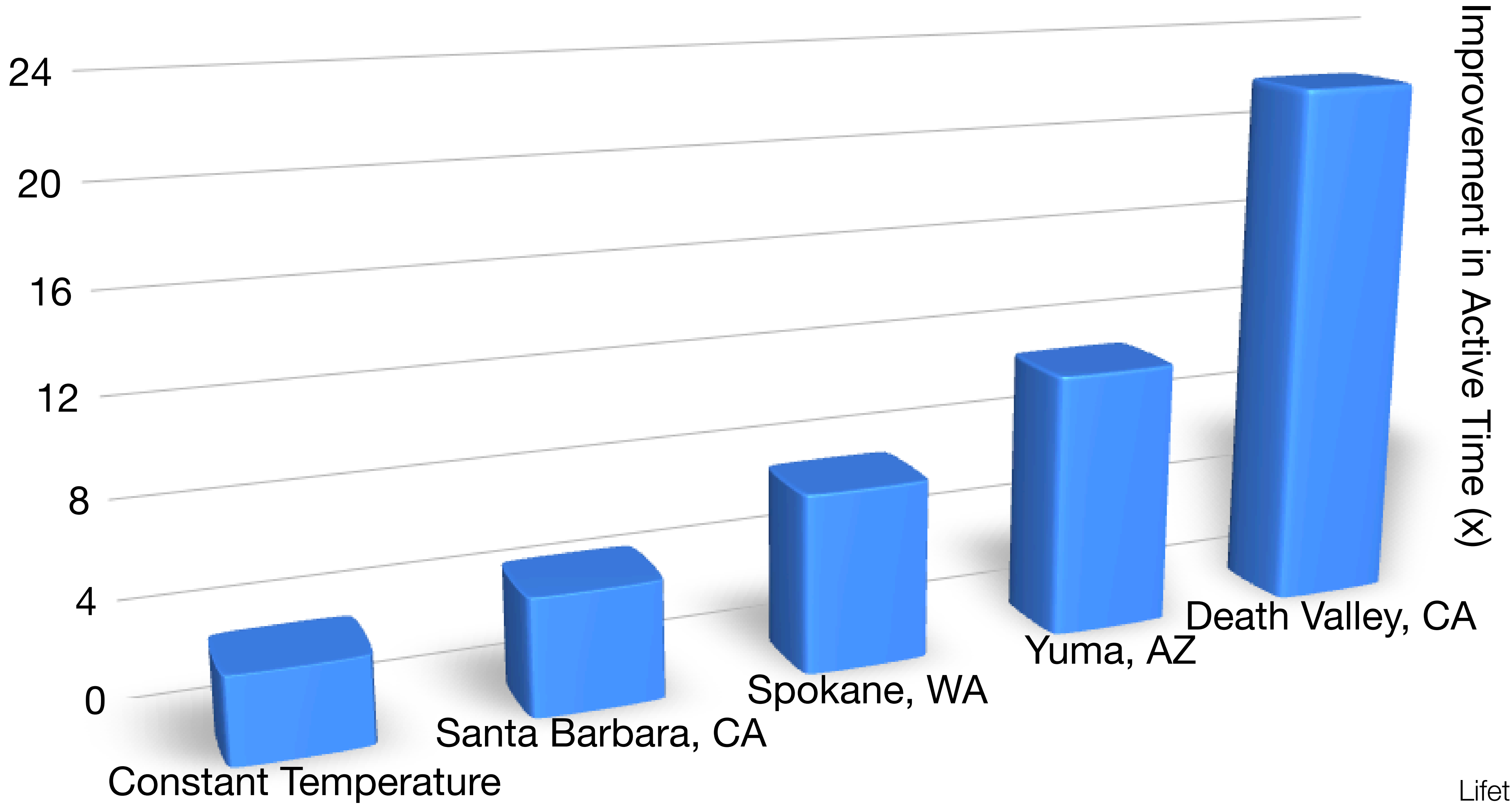


Lifetime reduction with Datasheet Spec DC

Average: 55 days short of one year's lifetime

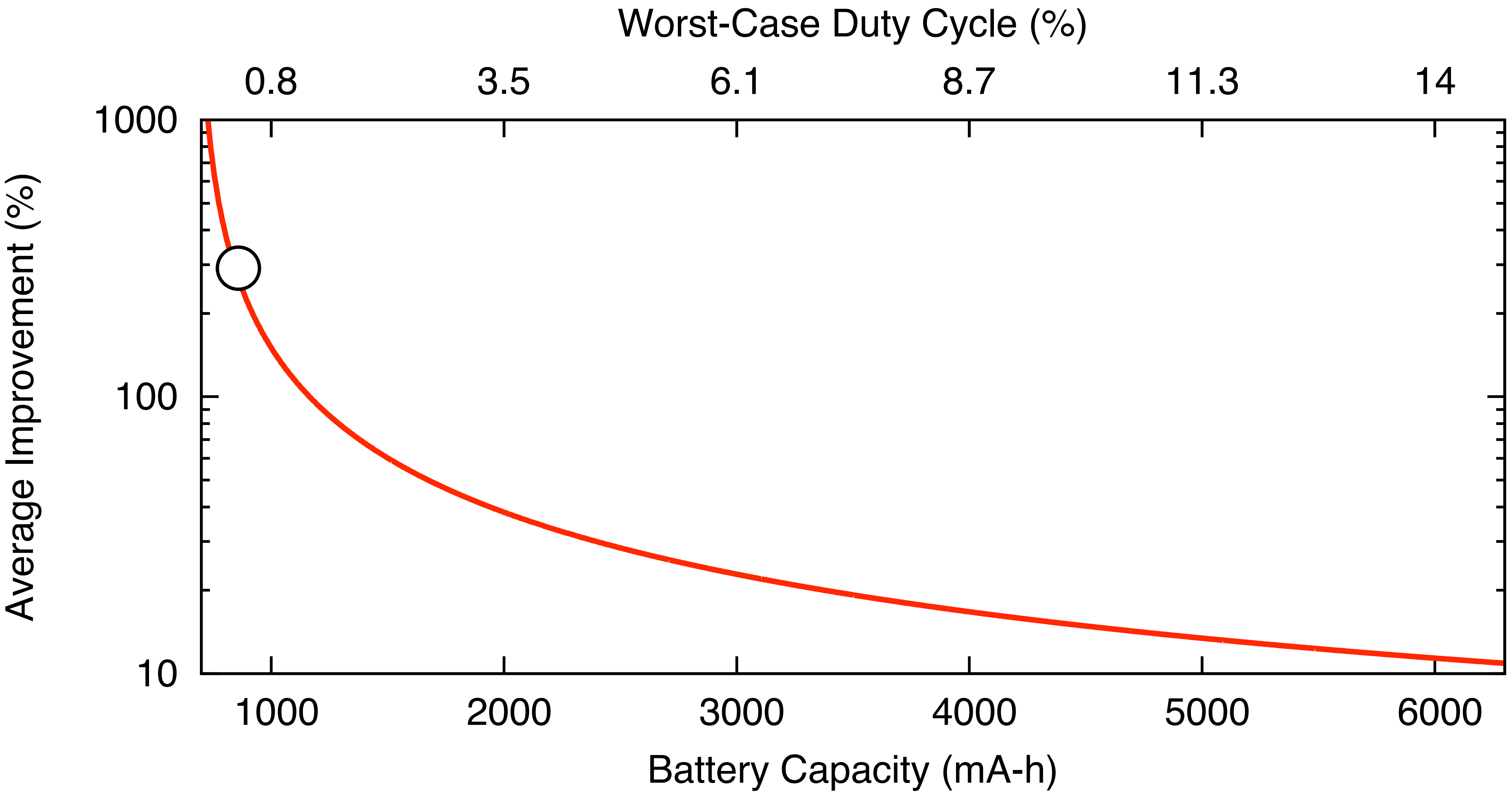


Average improvement by location

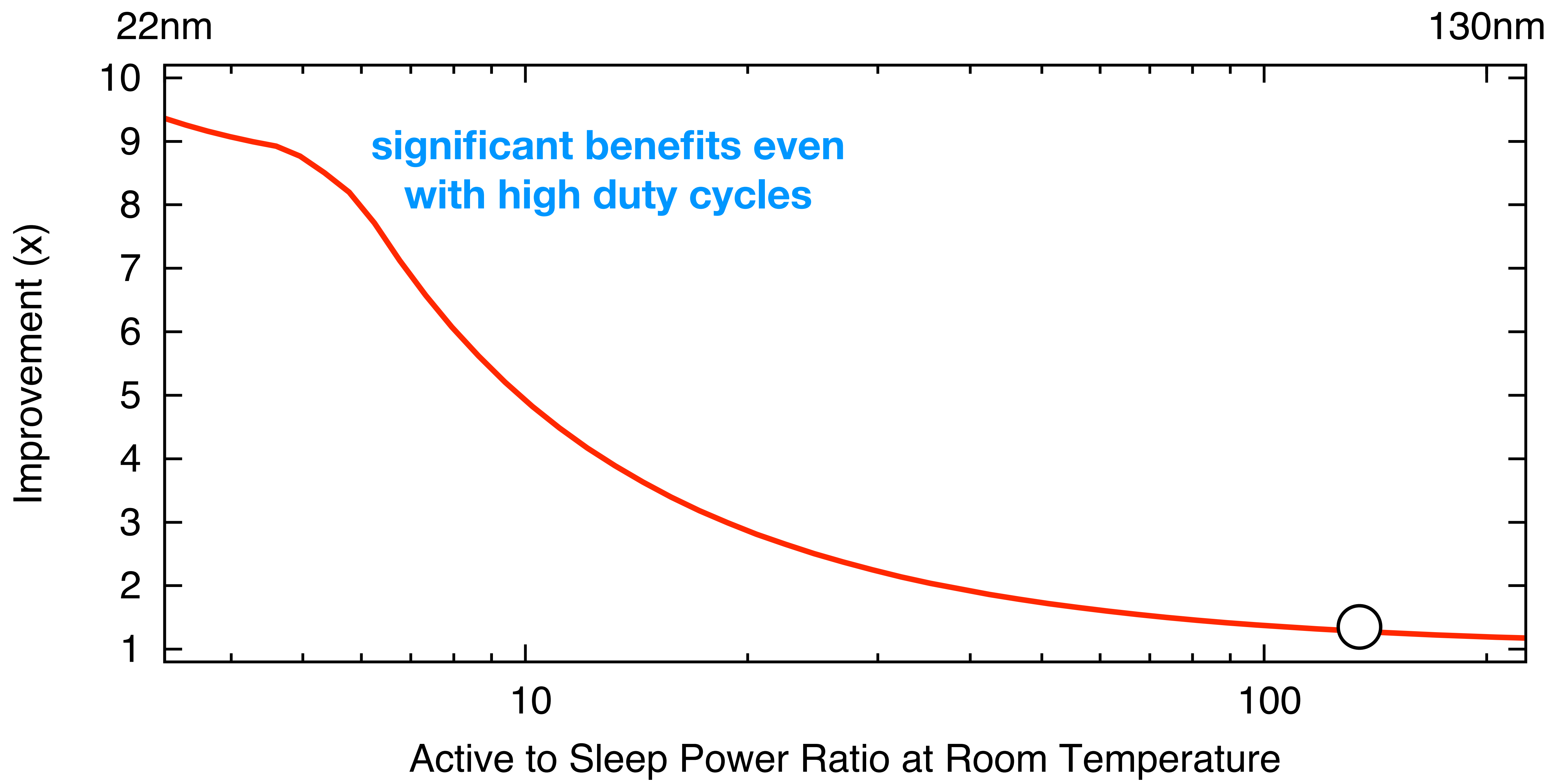


Lifetime: 1 year
Battery Capacity: 850 mAh
Temperature Profiles: NCDC hourly data, 2009

Benefits greater at smaller duty cycles

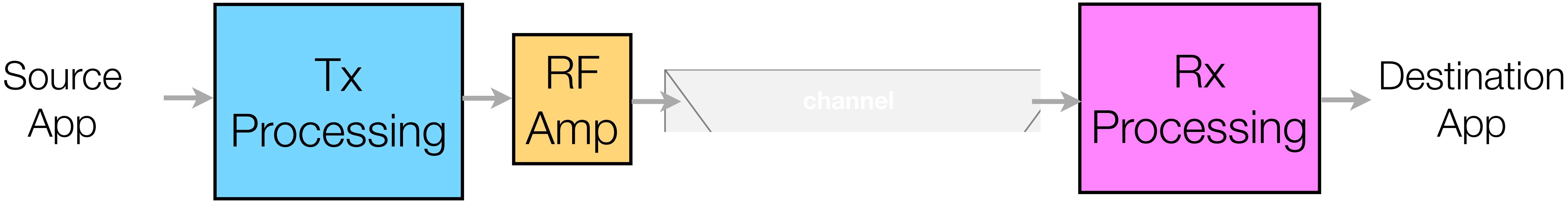


Benefits greater with newer technology



Worst-Case Duty Cycle: 10%
Temperature range: 60C

Another Example: Underdesigned Radios



Problem:	*error, deadline misses, & variability*	*error, loss & variability*	*error, deadline misses, & variability*
Current Practice:	*over-design for no error and minimum speed*	*tolerate via protocol and app level recovery*	*over-design for no error and minimum speed*
	↓		↓
	tolerate computational errors, deadline misses, and performance variation		*tolerate computational errors, deadline misses, and performance variation*

Underdesigned & Opportunistic Computing (UNO) Machines: From *Crash-and-Recover* to *Sense-and-Adapt*

Do Nothing

(Elastic User, Robust App)

Change Hardware Operating Point

(Disabling parts of the cache, Changing V-f)

Change Algorithm Parameters

(Codec Setting, Duty Cycle Ratio)

Change Algorithm Implementation

(Alternate code path, Dynamic recompilation)

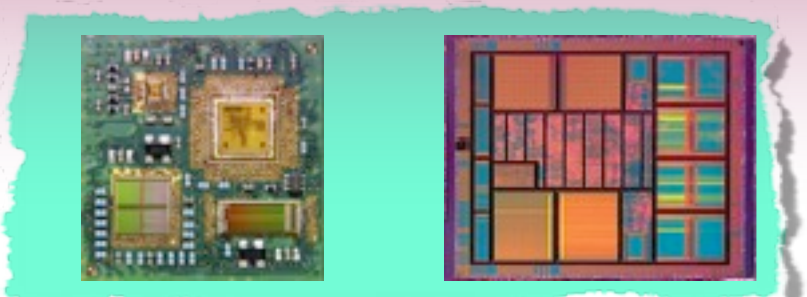
Change to Algorithm with Different Characteristics

(Dynamic linking to new library module)

How should hardware variability be exposed to the various software layers?

Variability signatures:
- *-cache bit map*
- *-cpu speed-power map*
- *-memory access time*
- *-ALU error rates*

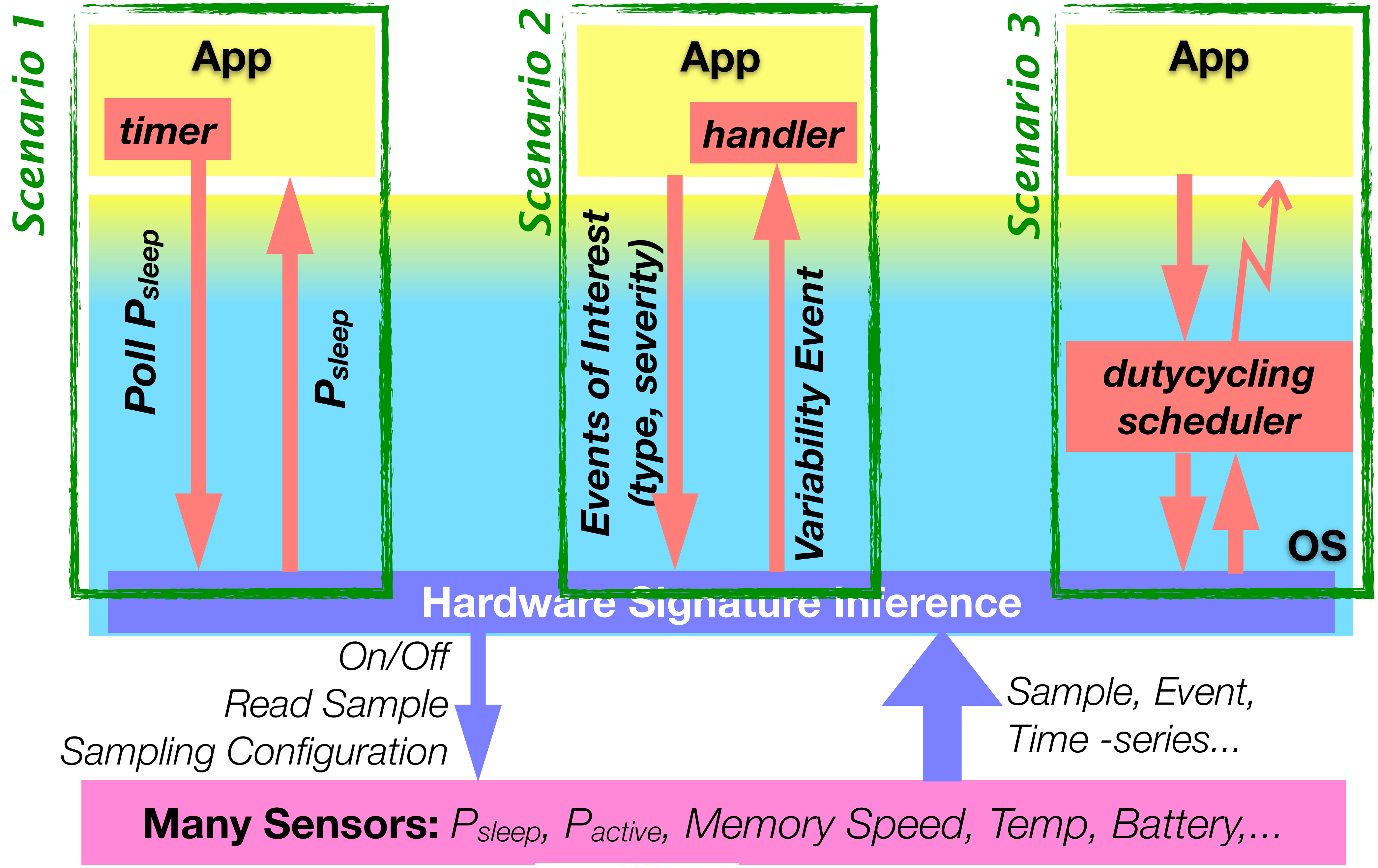
sensors & models



Underdesign Mechanisms
- *-stochastic processor*
- *-fluid hw constraints*
- *-application intent*

Variability manifestations
- *-faulty cache bits*
- *-delay variation*
- *-power variation*

Designing an UnO Stack for Variability-aware Duty-cycling



Scenario 1

App

timer

Poll P_{sleep}

P_{sleep}

Scenario 2

App

handler

Events of Interest (type, severity)

Variability Event

Scenario 3

App

dutycycling scheduler

OS

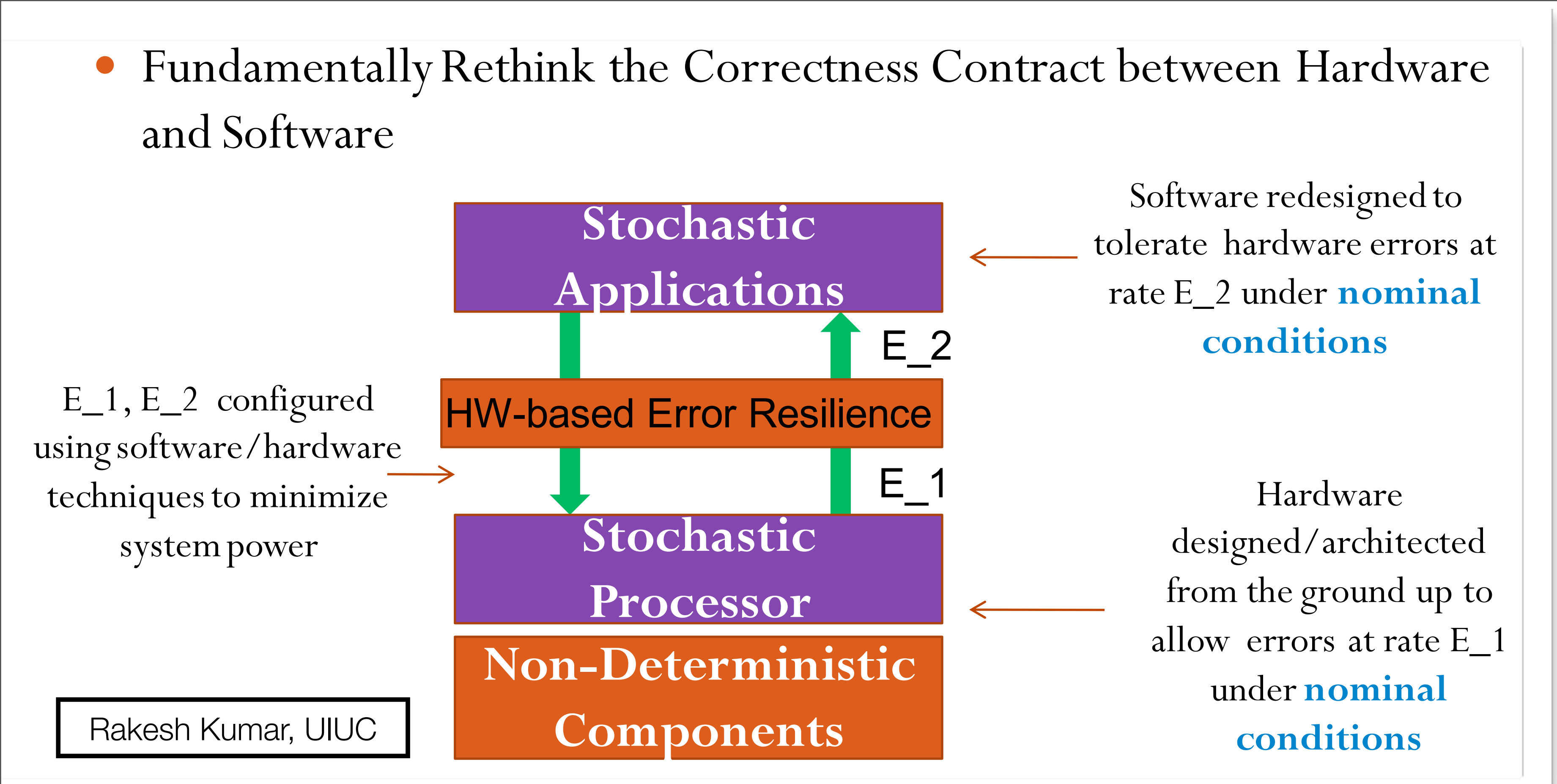
Hardware Signature Inference

On/Off
Read Sample
Sampling Configuration

Sample, Event, Time -series...

Many Sensors: *P_{sleep}, P_{active}, Memory Speed, Temp, Battery,...*

Duty Cycle = f($\mathbf{P_{sleep}}$, P_{active})



- Fundamentally Rethink the Correctness Contract between Hardware and Software

Stochastic Applications

E_2

HW-based Error Resilience

E_1

Stochastic Processor

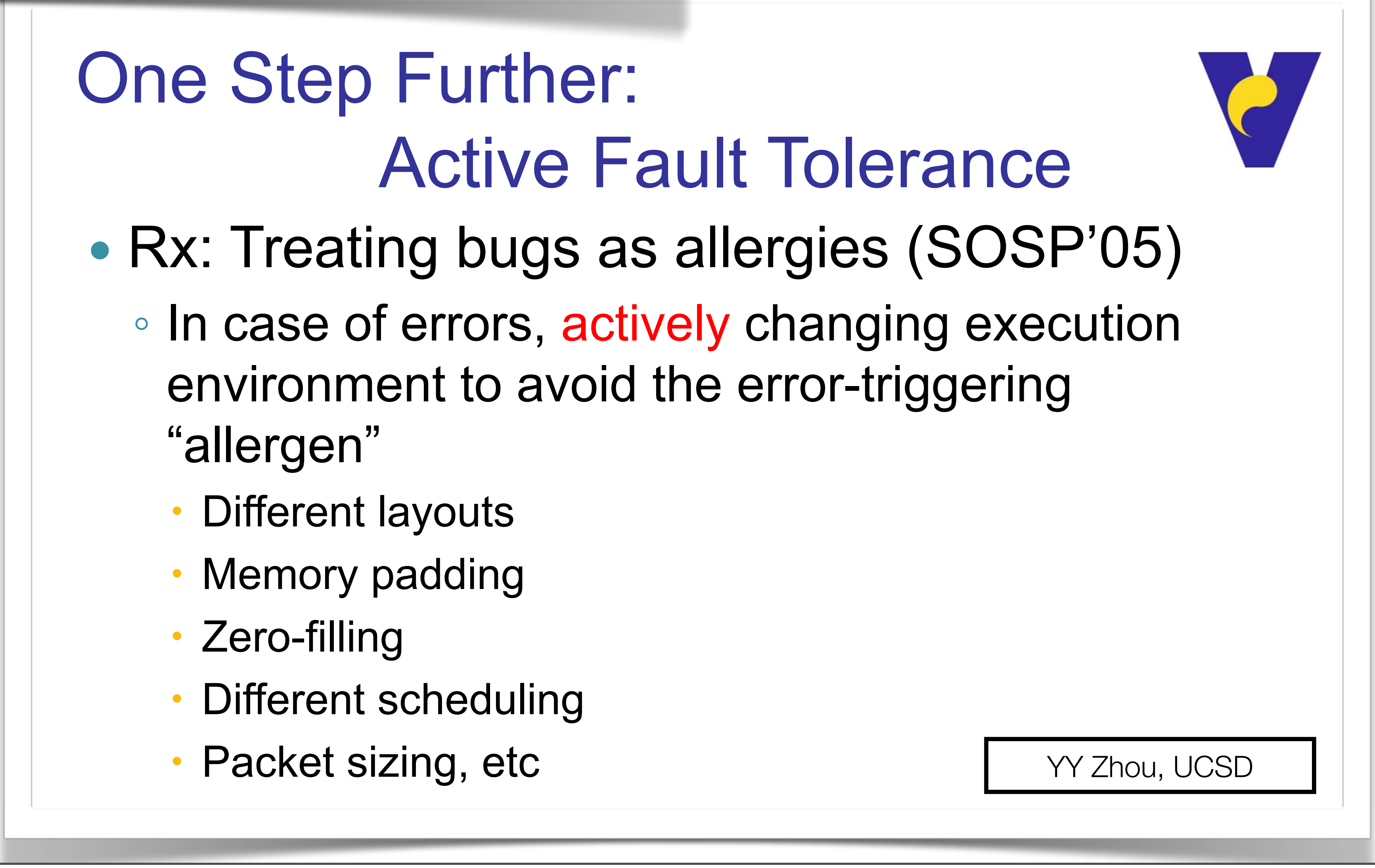
Non-Deterministic Components

Software redesigned to tolerate hardware errors at rate E_2 under **nominal conditions**

E_1, E_2 configured using software/hardware techniques to minimize system power

Hardware designed/architected from the ground up to allow errors at rate E_1 under **nominal conditions**

Rakesh Kumar, UIUC

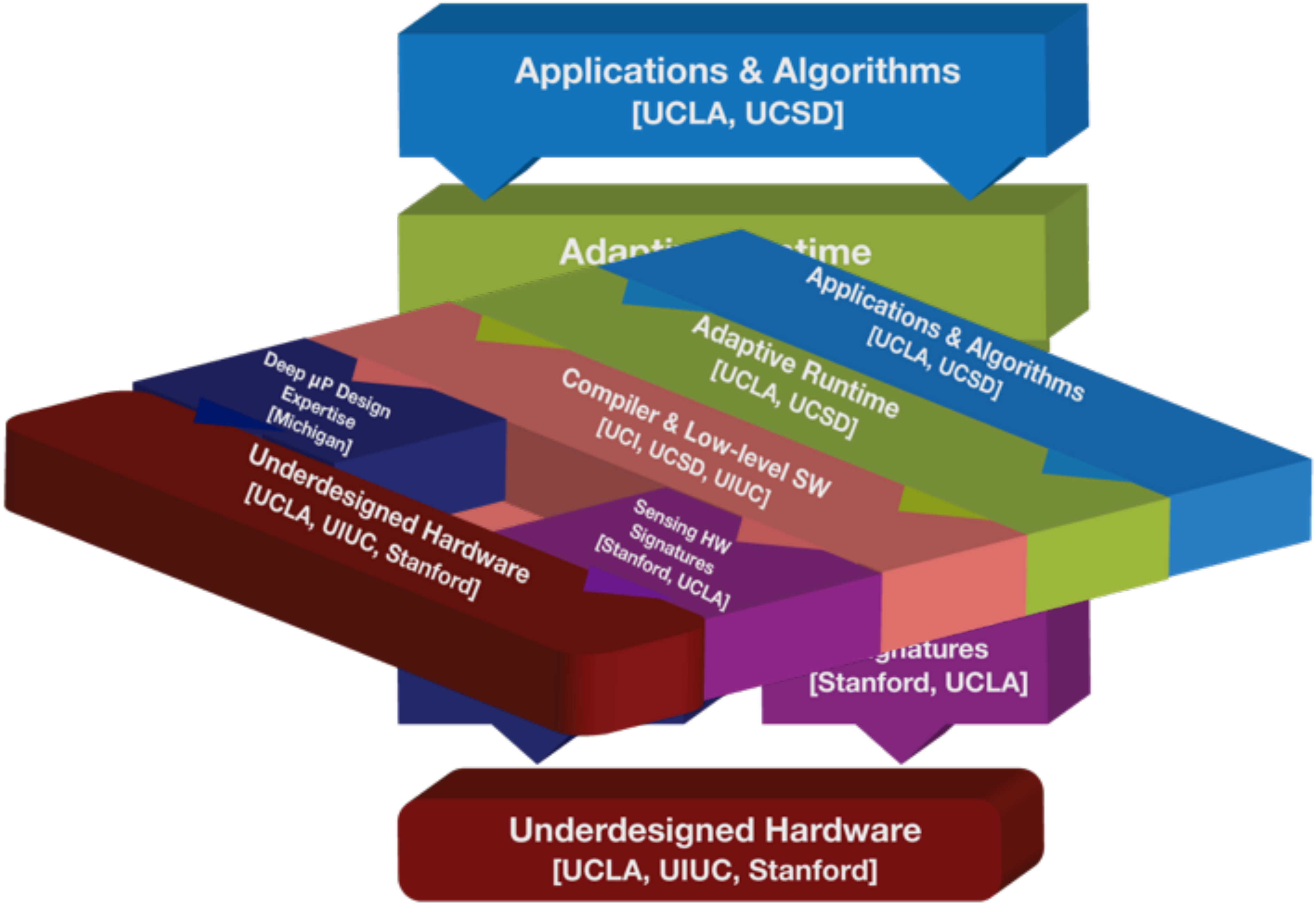


One Step Further: Active Fault Tolerance

- Rx: Treating bugs as allergies (SOSP'05)
  - In case of errors, actively changing execution environment to avoid the error-triggering "allergen"
    - Different layouts
    - Memory padding
    - Zero-filling
    - Different scheduling
    - Packet sizing, etc

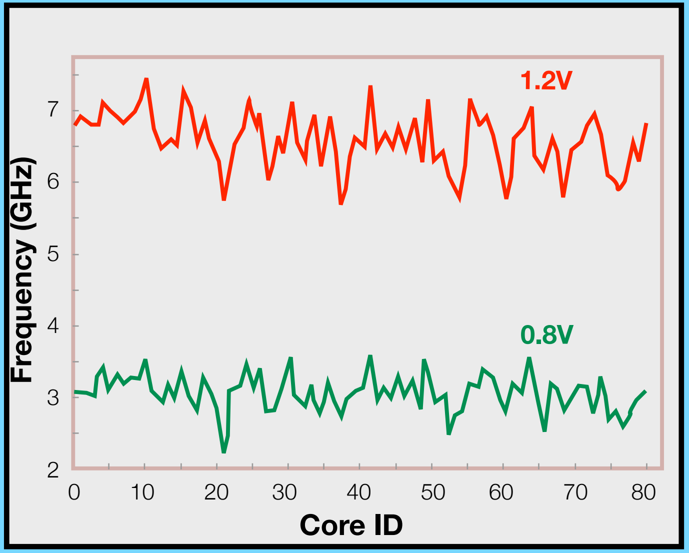
YY Zhou, UCSD

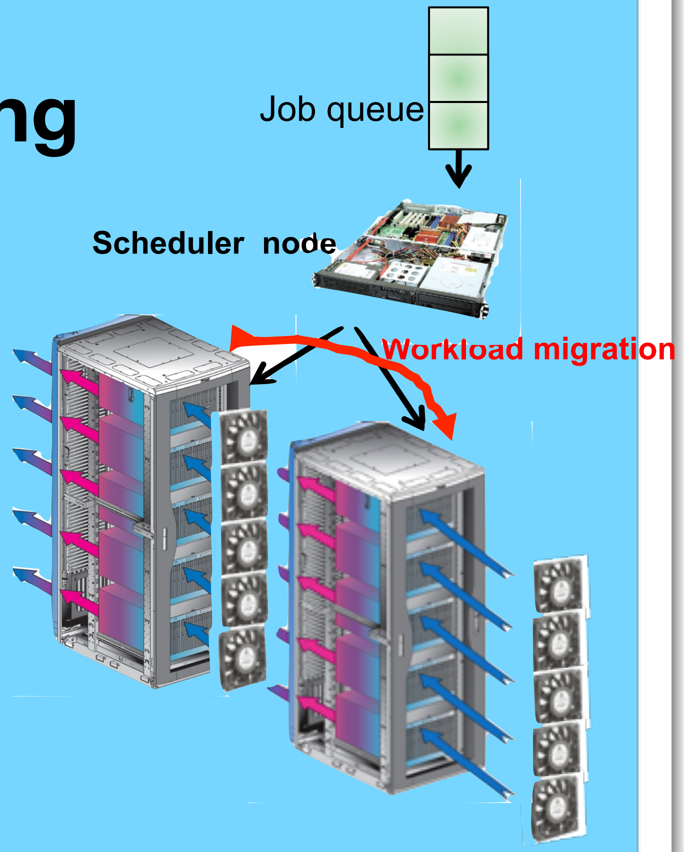
Realizing the Expeditions Project Vision


Applications & Algorithms [UCLA, UCSD]
Adapti time
Applications & Algorithms [UCLA, UCSD]
Adaptive Runtime [UCLA, UCSD]
Compiler & Low-level SW [UCI, UCSD, UIUC]
Deep μP Design Expertise [Michigan]
Underdesigned Hardware [UCLA, UIUC, Stanford]
Sensing HW Signatures [Stanford, UCLA]
gnatures [Stanford, UCLA]
Underdesigned Hardware [UCLA, UIUC, Stanford]

Realizing the Expeditions Project Vision

Testbed 1: General Purpose Computing



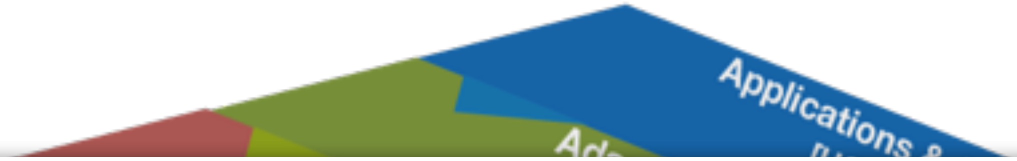


Instrumented Flash Servers in GreenLight Datacenter [UCSD]

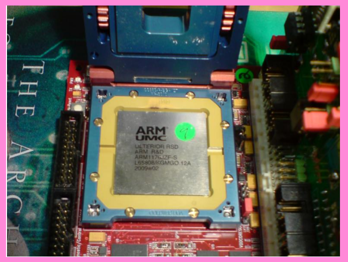
Off-line variability characterization and Run-time hardware signature sensing[UCSD, UCLA, UIUC]

Software Mechanisms for DB Querying and Map-Reduce Apps [UCSD, UCI]

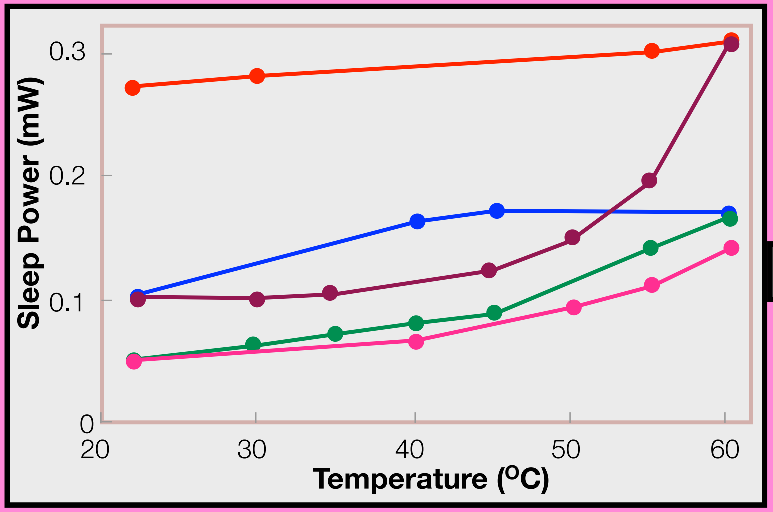
Realizing the Expeditions Project Vision



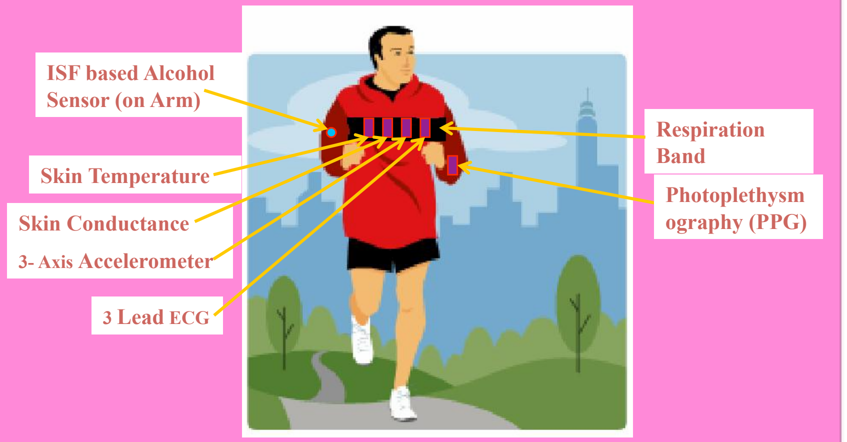
Testbed 2: Embedded Processing for Body Sensor Networks



Sensor Node with ARM Cortex M3 CPU with in situ Variability Sensor [UM, UCLA]



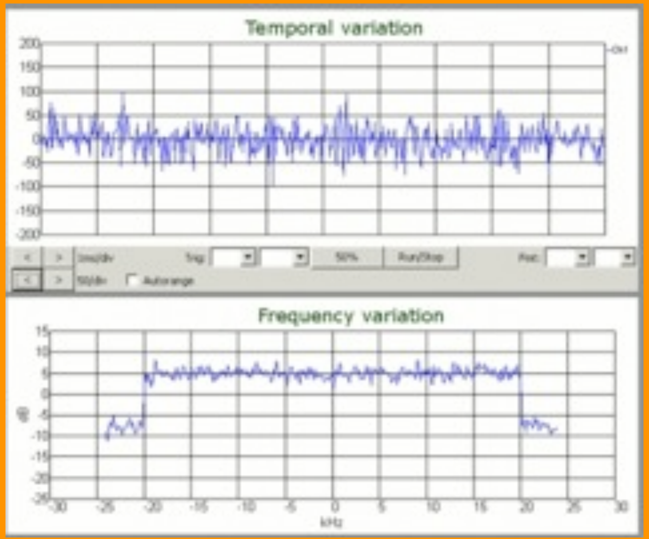
Off-line variability characterization and Run-time hardware signature sensing [Stanford, UCLA, UM]

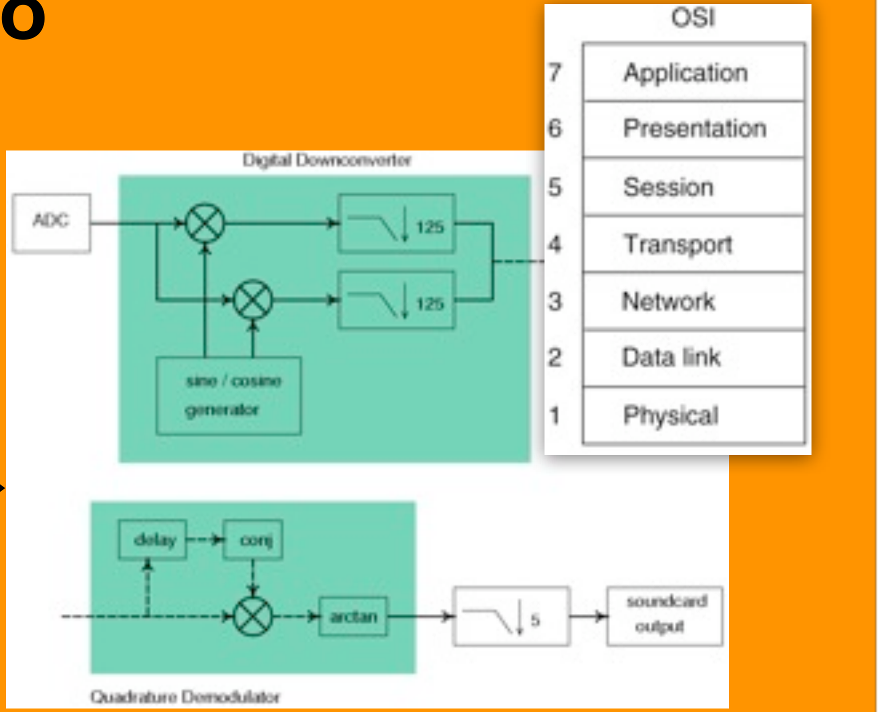


OS, PL, and App Mechanisms for Distributed Sensing [UCLA, UCI, UCSD]

Realizing the Expeditions Project Vision

Testbed 3: Software Radio





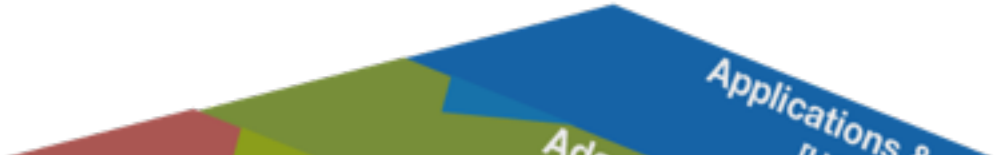
OSI	
7	Application
6	Presentation
5	Session
4	Transport
3	Network
2	Data link
1	Physical

ARM Cortex M3 CPU & Underdesigned DSP Accelerators [UM, UCLA]

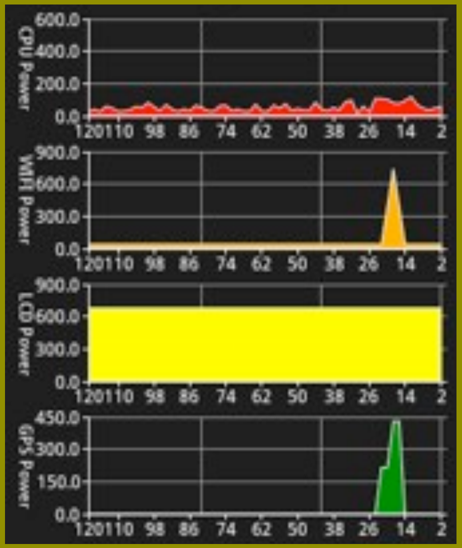
Off-line variability characterization and Run-time hardware signature sensing[UCLA, Stanford, UM]

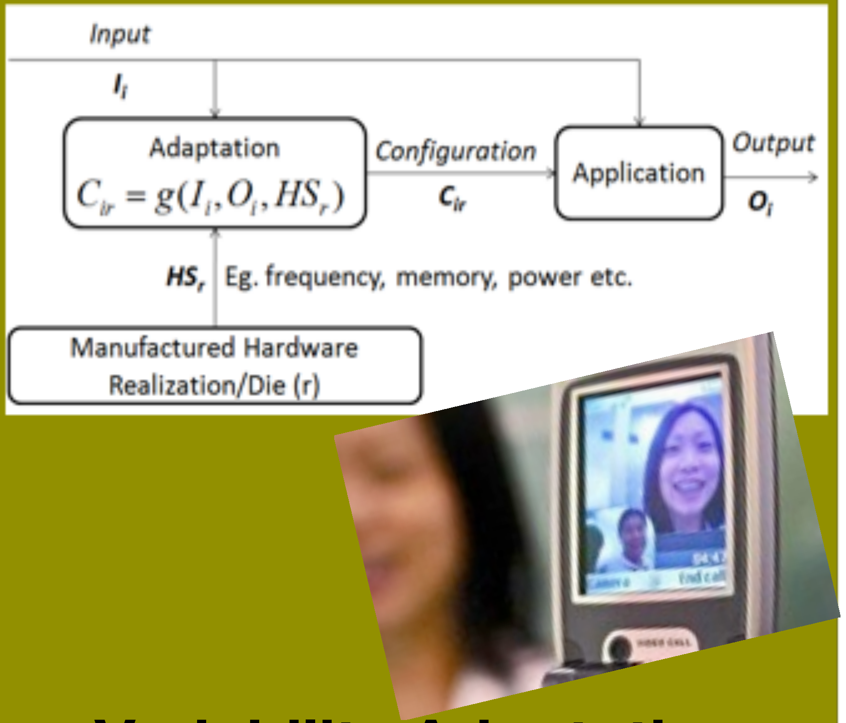
Variability-aware GNU Radio + N/W Protocol Stack under Linux [UCLA, UCSD]

Realizing the Expeditions Project Vision



Testbed 4: Mobile Computing for Multimedia



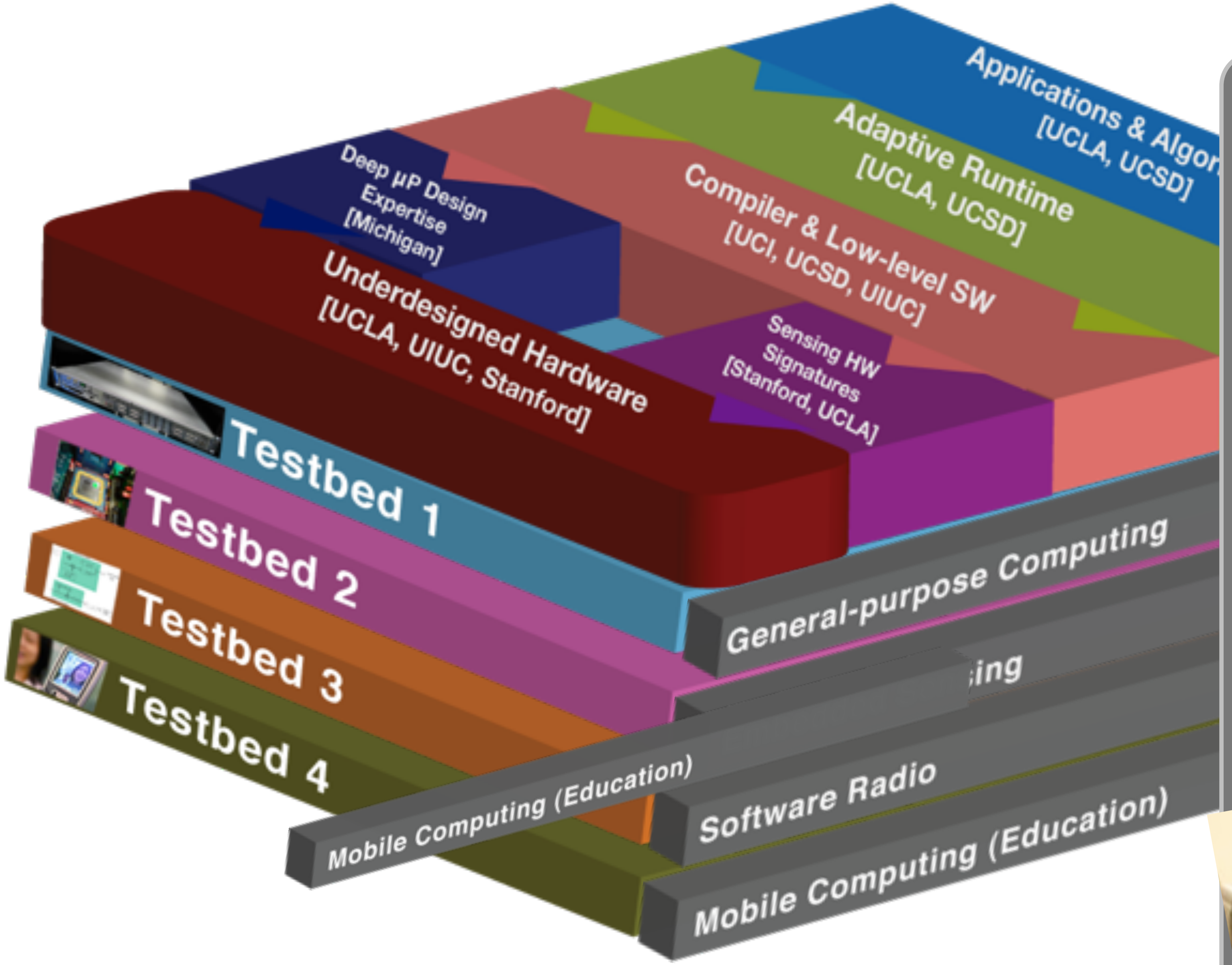

Input
I_i
Adaptation
$C_{ir} = g(I_i, O_i, HS_r)$
Configuration
C_{ir}
Application
Output
O_i
HS_r
Eg. frequency, memory, power etc.
Manufactured Hardware Realization/Die (r)

Instrumented Android Smartphone [UCLA]

Off-line & S/W-inference based run-time power & error variability characterization [UCLA]

Variability Adaptation Mechanisms for VP8 Codec [UCI, UCLA]

Realizing the Expeditions Project Vision



Outreach

Physically-minded Computing

COSMOS
LACC
…

Variability Expedition:
A Paradigm Shift to Fluid HW-SW Interfaces

Opportunistic SW

- **Radical departure from hard failures to soft variability**
  - Work through hardware variability
    - rather than over-designed hardware and fault-handling software
  - Software becomes a significant part of the solution to variability
- **Software adapts to part as manufactured rather than as designed**
  - opportunistically exploit application elasticity
  - adaptation <u>simplifies</u> the structure of software layers

C

/

Softv
Ec
Scaling

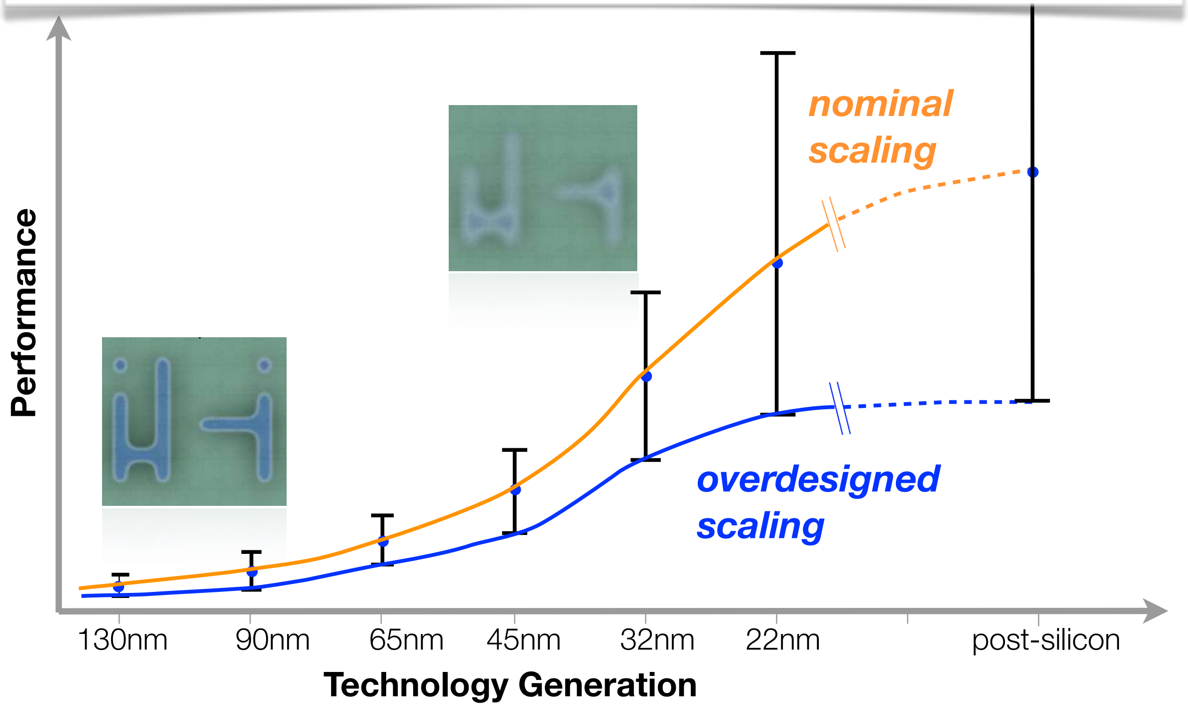
Problems

Wall

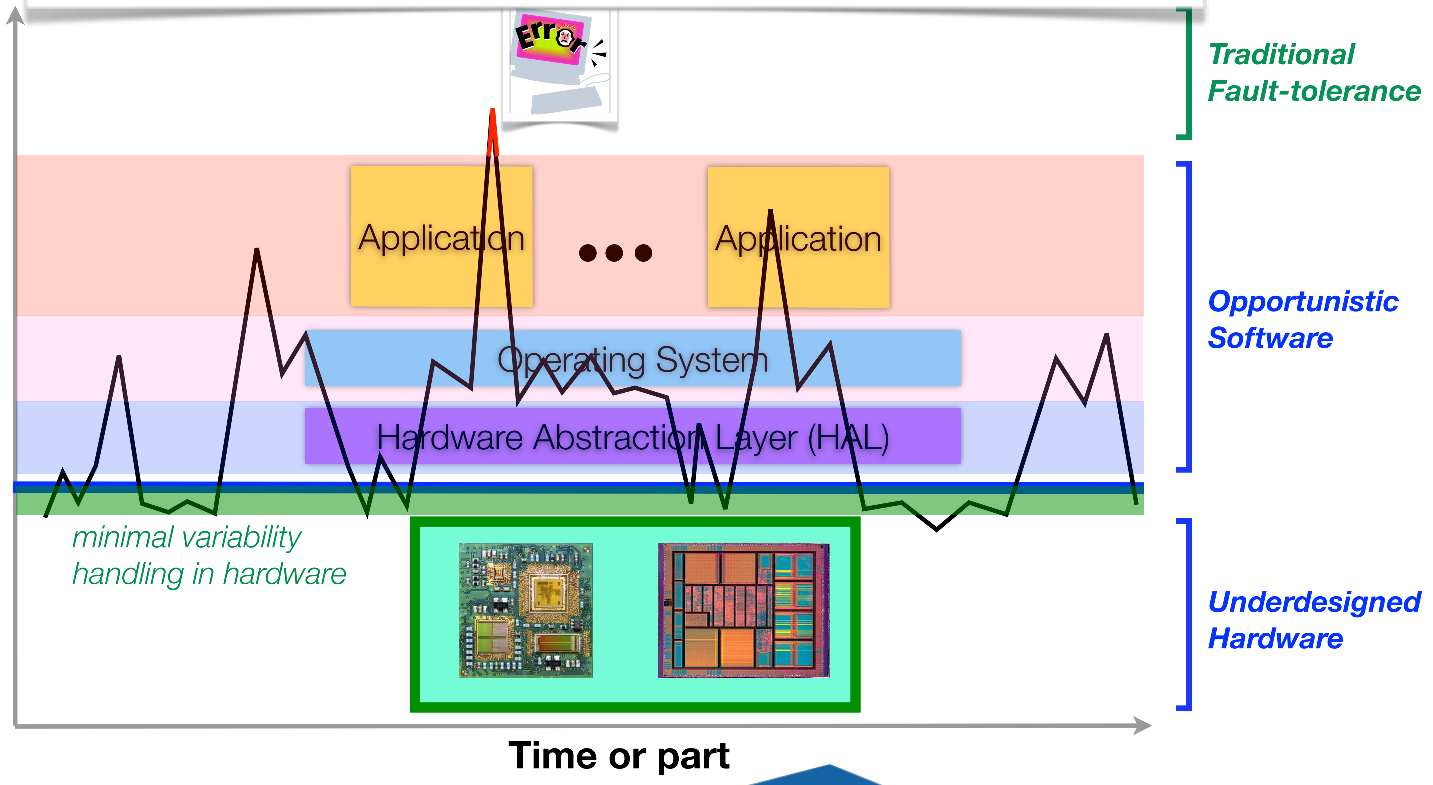
rds
ally-
ed
Computing

Variability-Aware Software for Efficient Computing with Nano-scale Devices

Problem: Increasing variability in nanoscale devices leading cause of overdesigned hardware.

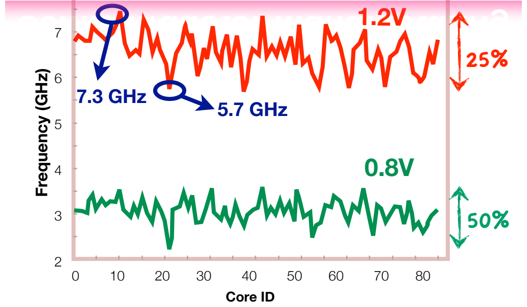


Goal: Re-architect the hardware-software stack



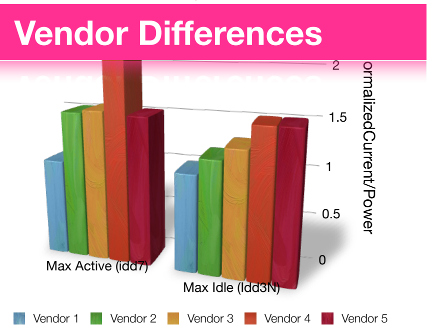
Frequency variation in an 80-core processor within a single die in Intel's 65nm technology

Semiconductor Manufacturing



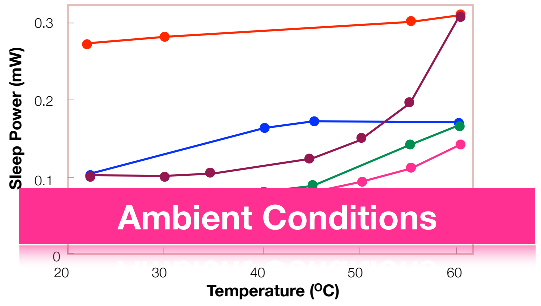
Power variation across five 512 MB DDR2-533 DRAM parts [Hanson07]

Vendor Differences



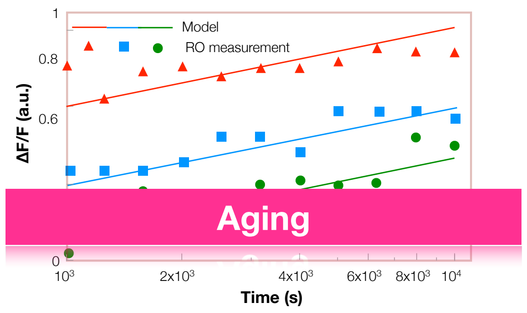
Variation in P_{sleep} with temperature across five instances of an ARM Cortex M3 processor

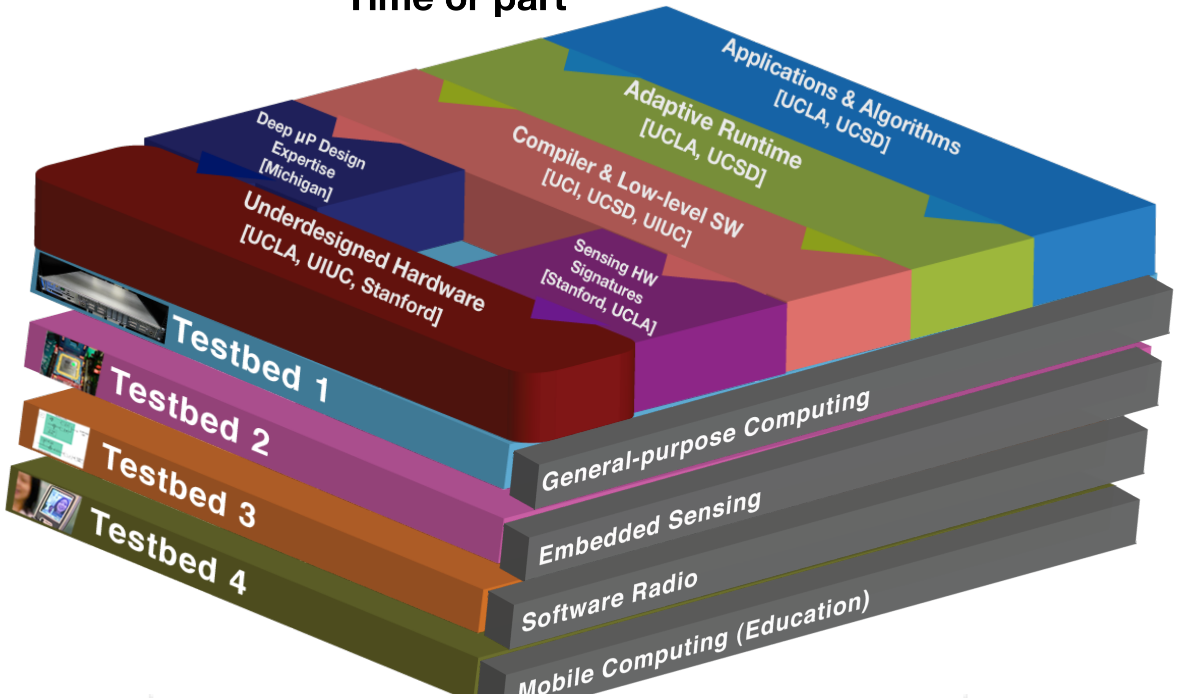
Ambient Conditions



Normalized frequency degradation in 65 nm due to NBTI [Zheng09]

Aging




Applications & Algorithms [UCLA, UCSD]
Adaptive Runtime [UCLA, UCSD]
Compiler & Low-level SW [UCI, UCSD, UIUC]
Deep µP Design Expertise [Michigan]
Underdesigned Hardware [UCLA, UIUC, Stanford]
Sensing HW Signatures [Stanford, UCLA]
Testbed 1
Testbed 2
Testbed 3
Testbed 4
General-purpose Computing
Embedded Sensing
Software Radio
Mobile Computing (Education)

http://www.variability.org

UCSanDiego UCLA MICHIGAN STANFORD UNIVERSITY UCIRVINE

NSF

Thursday, July 14, 2011