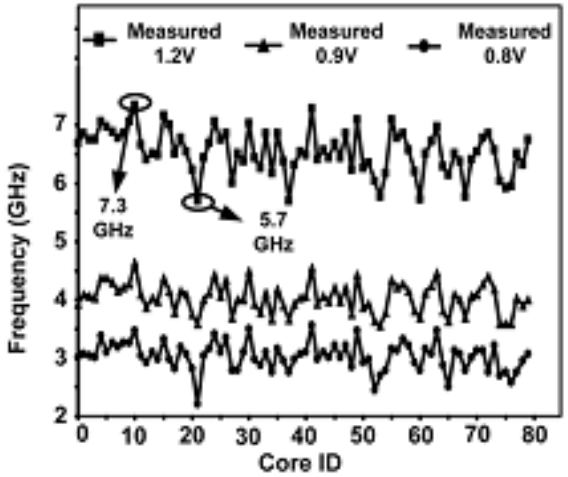
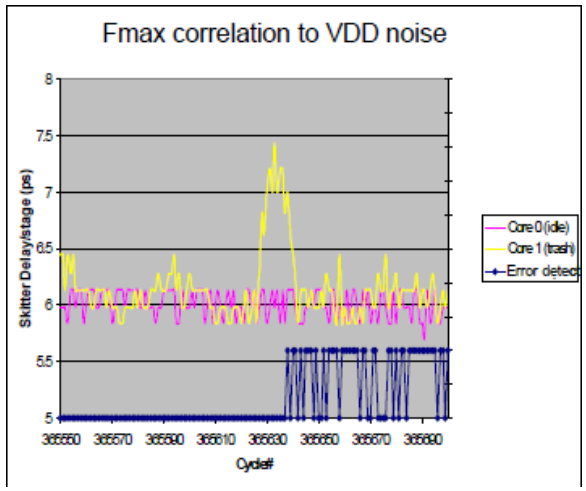
Stochastic Computing, Multi-modal Design, and Power-Balanced Processors: Throwing Kitchen Sink at the Power Problem

Asstnt Prof Rakesh Kumar

Department of Electrical and Computer Engineering

University of Illinois, Urbana-Champaign





L (nm)	250	180	130	90	65	45
Vt (mV)	450	400	330	300	280	200
σ-Vt (mV)	21	23	27	28	30	32
σ-Vt/Vt	4.7%	5.8%	8.2%	9.3%	10.7%	16%

Non-determinism ↑

Yield (for a given frequency/power target) ↓

Energy Cost (for a given yield) ↑

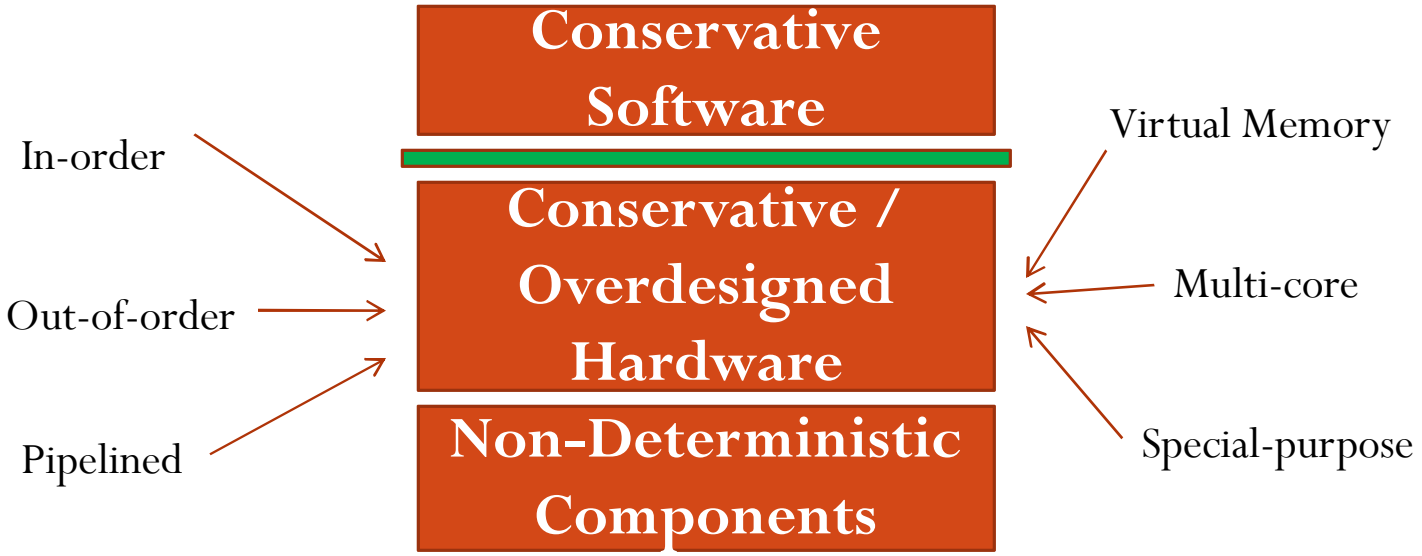
Status quo cannot continue: Need to find a solution to the non-determinism problem to limit power

A "buck-a-billion" in 2020; micro-watt computing devices; costs of non-determinism unacceptable for emerging applications

Non-determinism is NOT the Problem, It is How Computer System Designers Treat it

Software expects hardware to behave flawlessly in spite of non-determinism

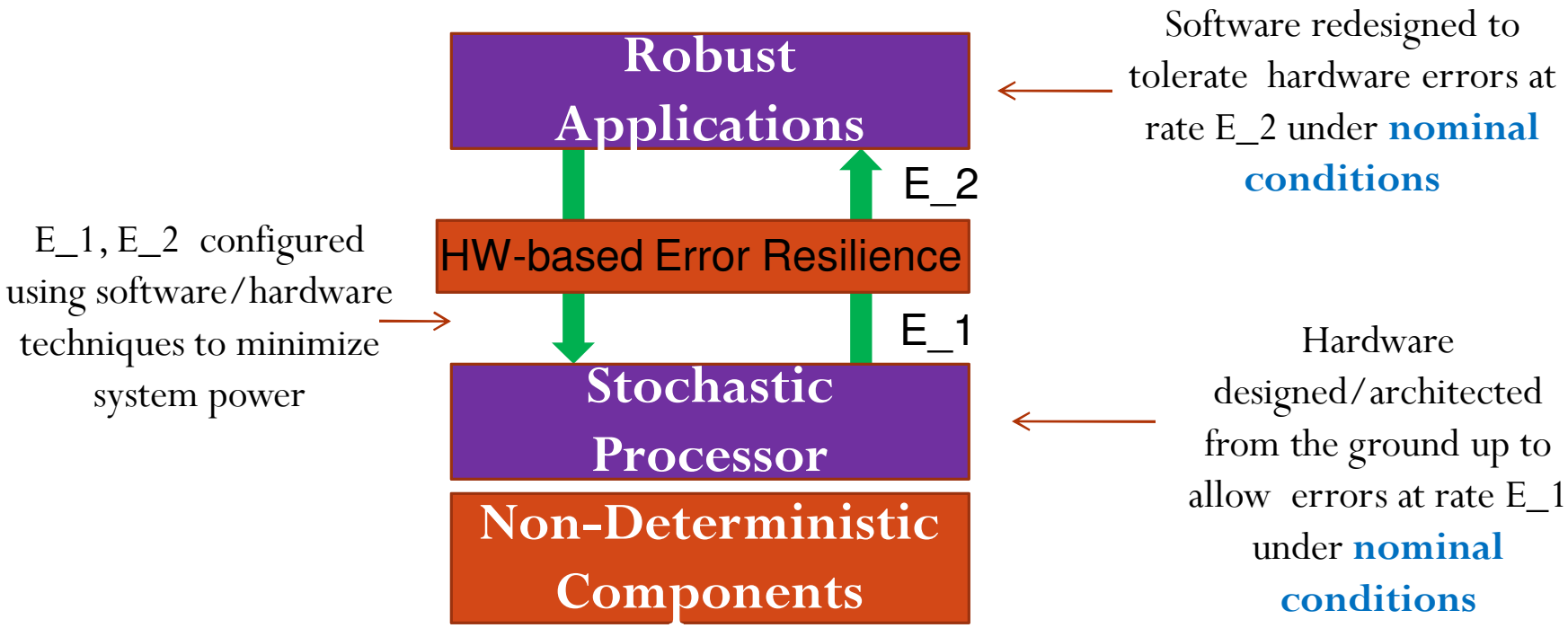
Hardware ALWAYS designed for correctness (under worst-case or nominal conditions); an attempt to meet the conventional software mindset at all cost



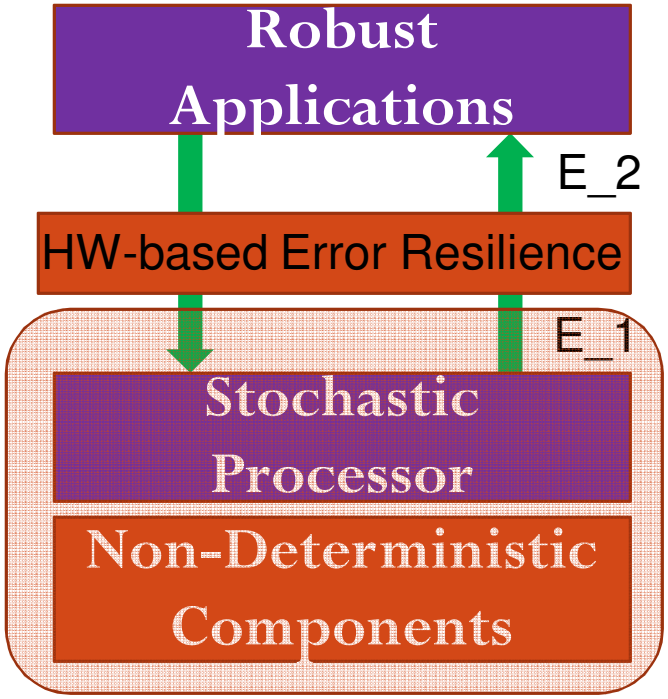
There would be no energy cost of non-determinism if non-determinism can be exposed directly to the software under nominal conditions as opposed to eliminating it or hiding it.

Research Vision: Computing with Stochastic Processors

- Rethink the Correctness Contract between Hardware and Software

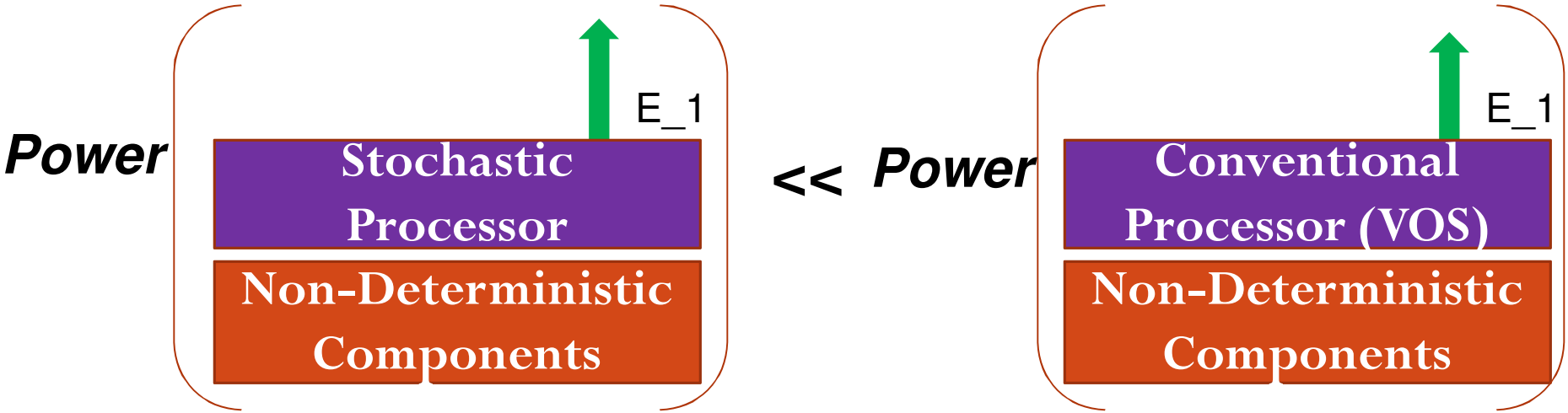


The goal of our research is to explore approaches to architect and design stochastic processors and robust applications.



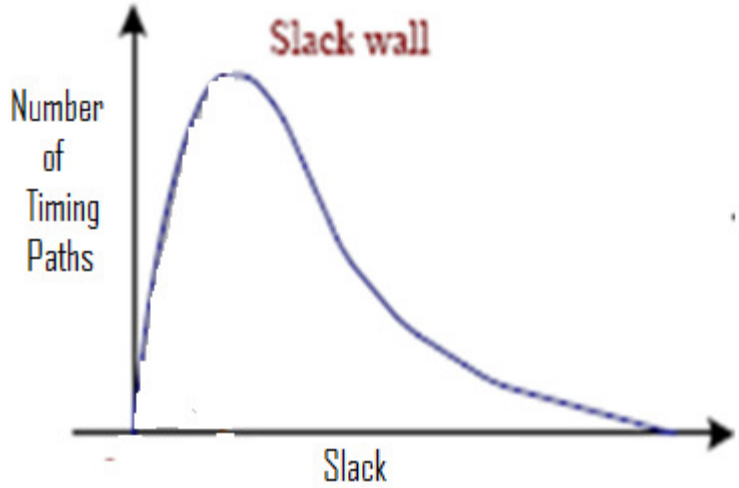
E_1 = Timing Error Rate
=Fraction of cycles during which at least
one latch/FF latches an incorrect value
=Fraction of cycles during which the output of at least one timing path
that is super-critical (i.e., has negative slack) toggles due to a change in the input

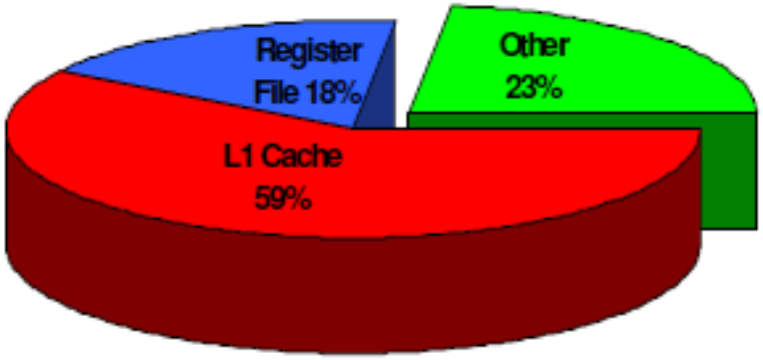
Hardware Design Goal

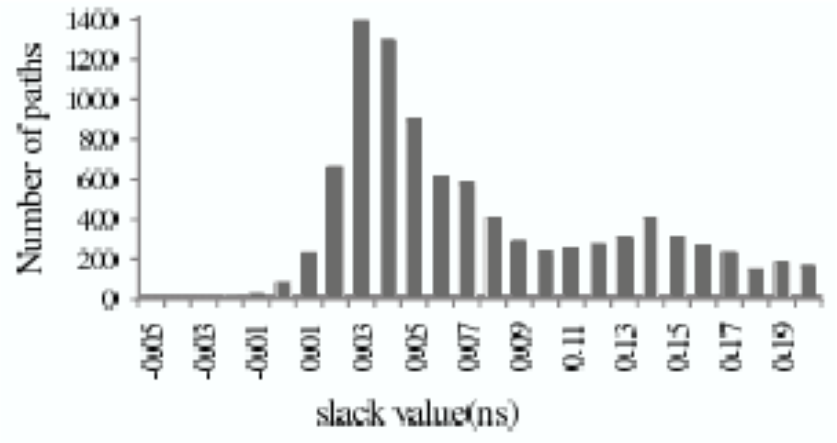


Using Conventional Processors for Non-zero Error Rate Operation

- Conventional processors have a timing slack wall





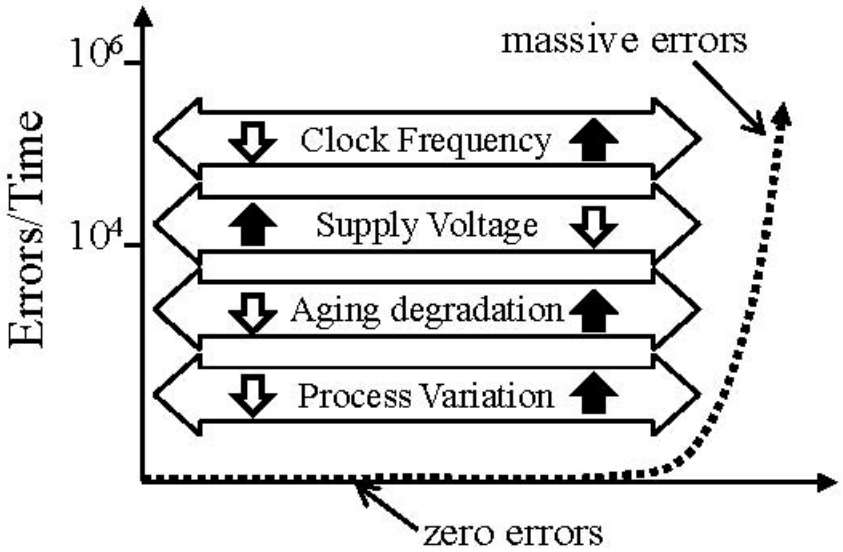


Courtesy: Northwestern

Using Conventional Processors for Non-zero Error Rate Operation

- Conventional processors have slack wall

	Error Rate (%)					
Module	1.0V	0.9V	0.8V	0.7V	0.6V	0.5V
lsu_dctl	0.00	0.23	8.60	29.46	45.13	54.90
lsu_qctl1	0.00	5.94	10.85	16.99	16.56	37.53
lsu_stb_ctl	0.00	0.08	0.65	5.19	11.79	22.38
sparc_exu_div	0.00	0.15	0.23	0.35	0.49	1.10
sparc_exu_ecl	0.00	3.31	10.97	87.08	88.93	73.03
sparc_ifu_dec	0.00	0.08	0.87	7.09	15.22	20.48
sparc_ifu_errdp	0.00	0.00	0.00	0.00	0.00	9.21
sparc_ifu_fcl	0.00	10.56	22.25	50.04	55.06	56.95
spu_ctl	0.00	0.00	0.00	1.30	2.96	35.53
tlu_mmu_ctl	0.00	0.01	0.02	0.06	0.14	0.19

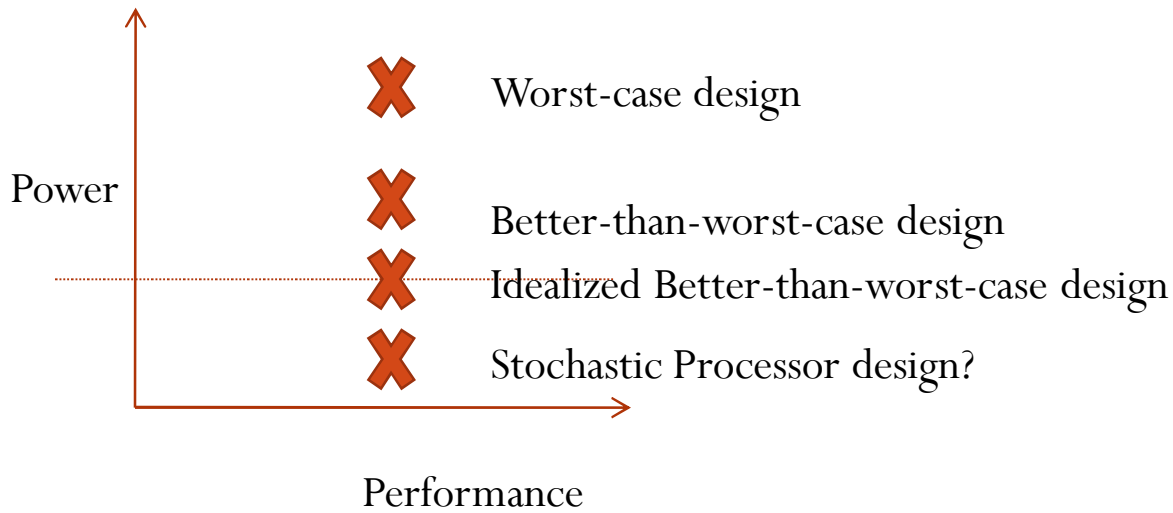


(HPCA2010, ASPDAC2010)

Too many errors produced if non-determinism is exposed (using VOS, for example), not much scaling possible before E_1 reached

Using Conventional Processors for Non-zero Error Rate Operation

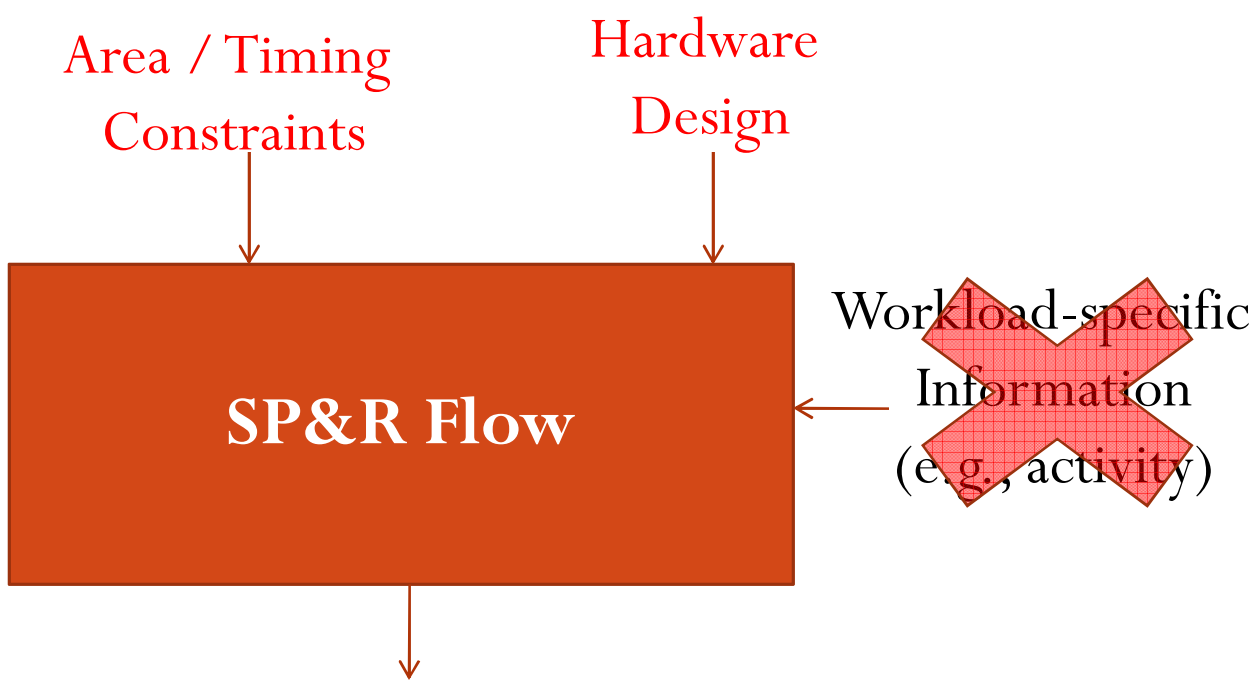
- Conventional processor have slack wall
- Conventional processors have inherently higher power/area as they are optimized for correct operation



Power difference between a conventional processor and a stochastic processor will become more pronounced as leakage increases

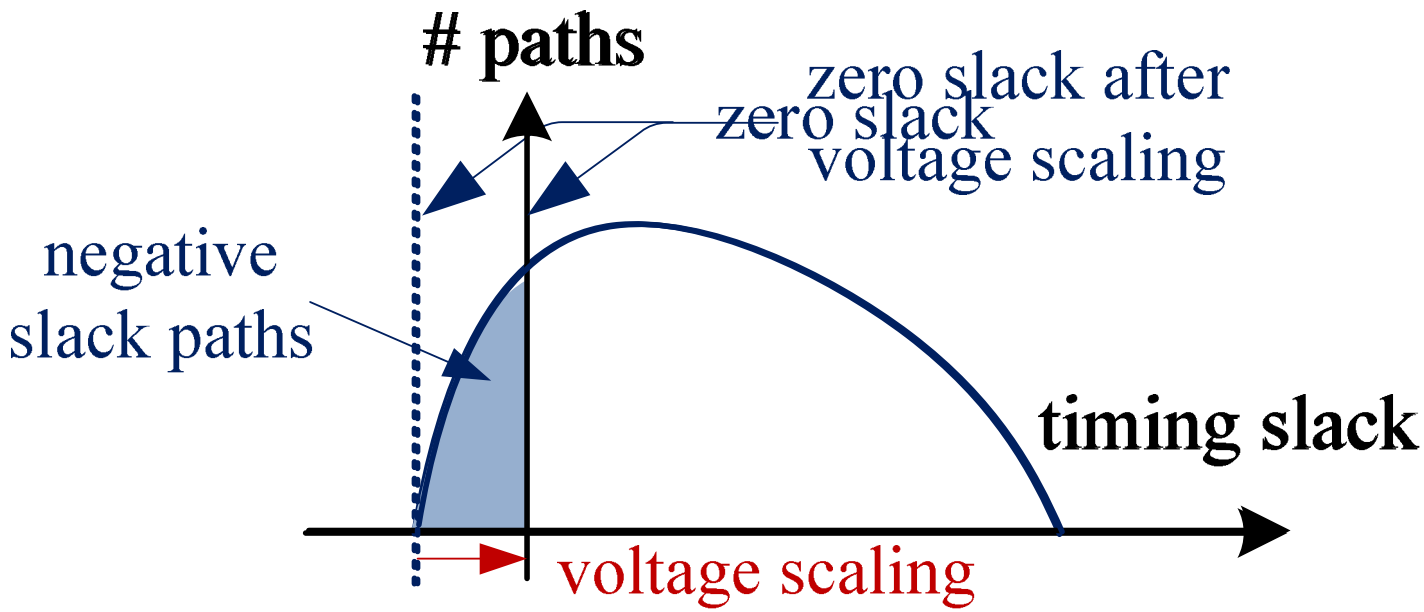
Using Conventional Processors for Non-zero Error Rate Operation

- Conventional processor have slack wall
- Conventional processors have inherently higher power/area as they are optimized for correct operation
- Conventional processors are workload-agnostic (STA, SSTA); therefore, heavily overdesigned for most workloads (false paths, etc)



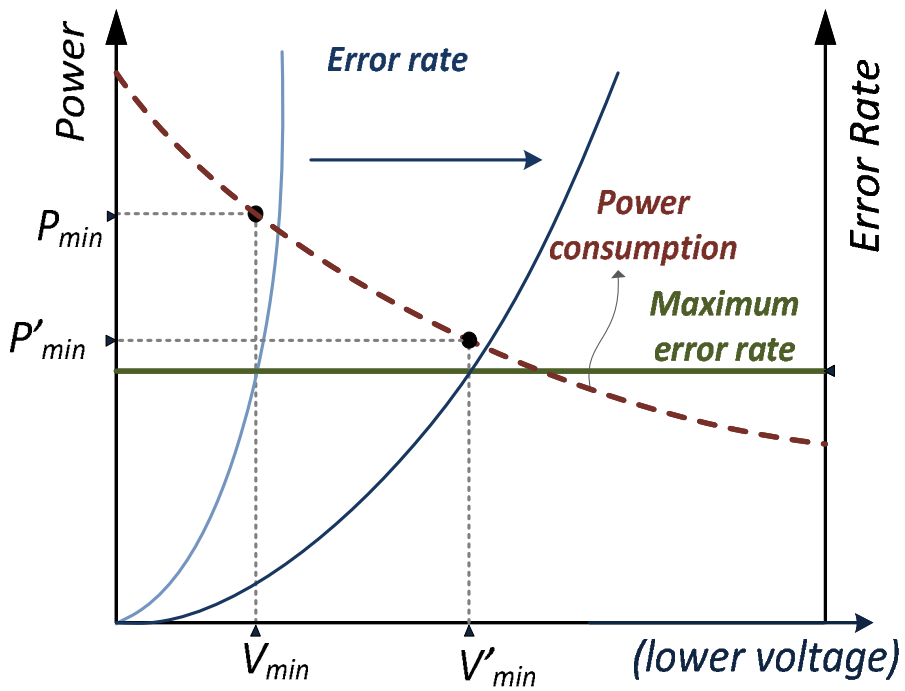
Design and Architecture of Stochastic Processors: Key Observation

- Error rate depends on path **slack** and **activity**
- Slack distribution determines which paths cause errors; activity determines the error rate contribution of the paths



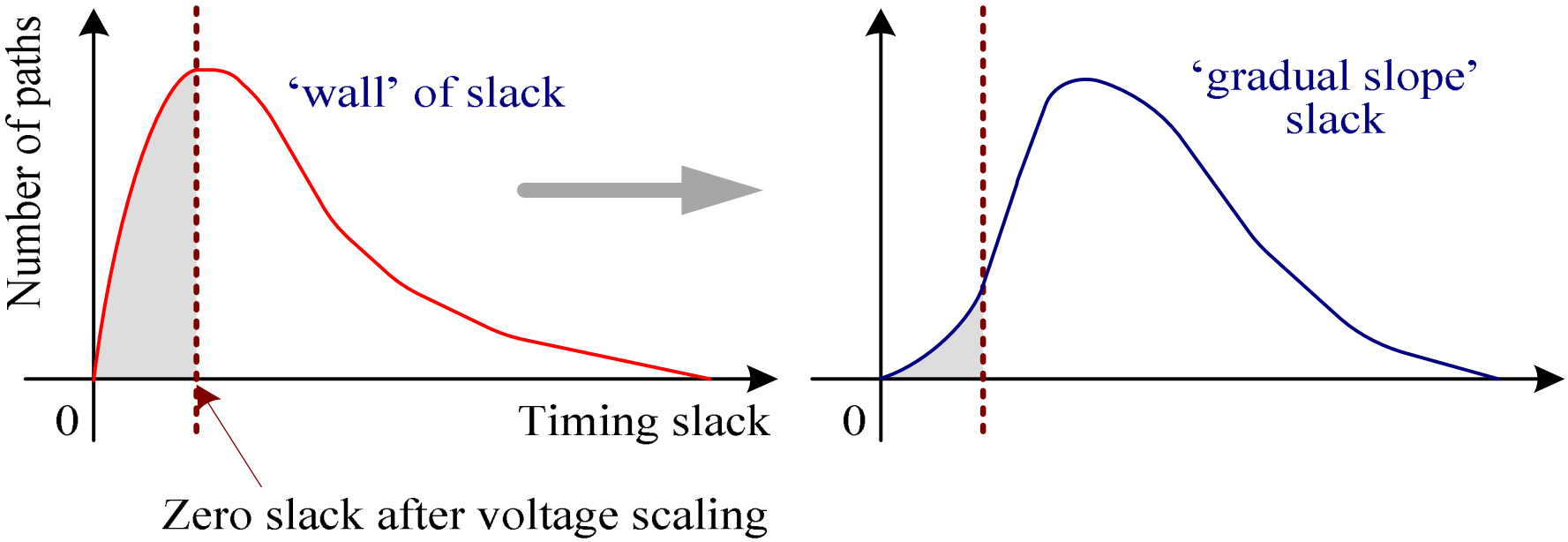
Design, Architecture, and Compilation Approaches to Manipulate Slack and Activity Distribution to Minimize system power when an Error Resilience Mechanism is Available

Soft Processor Design for Voltage Overscaling



(HPCA2010, ASPDAC2010)

Soft Processor Design: Power-aware Slack Redistribution for Stochastic Processors



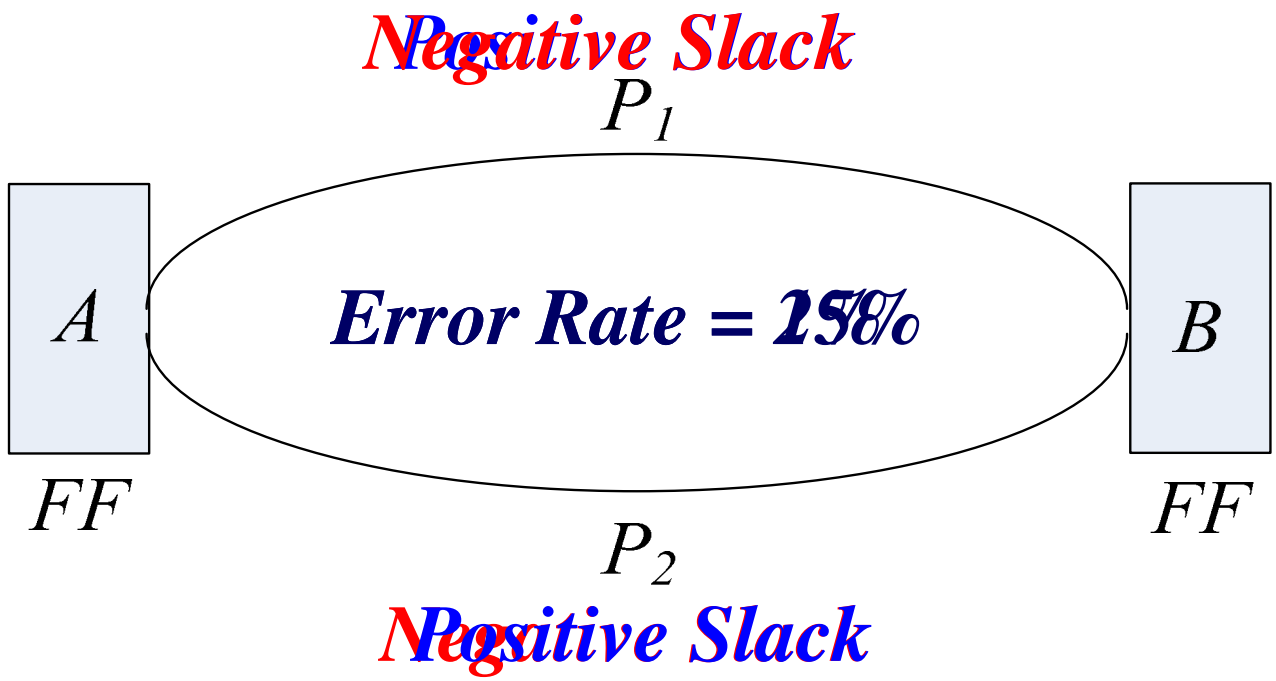
Insight: Frequently-exercised paths contribute most errors, but the number of such paths in a design is small

- **Optimize frequently exercised critical paths.**
- **De-optimize rarely exercised paths.**

Increase slack of (near) critical paths without consuming much extra power.

Both gradual failure and low power can be achieved.

Slack Re-distribution Example

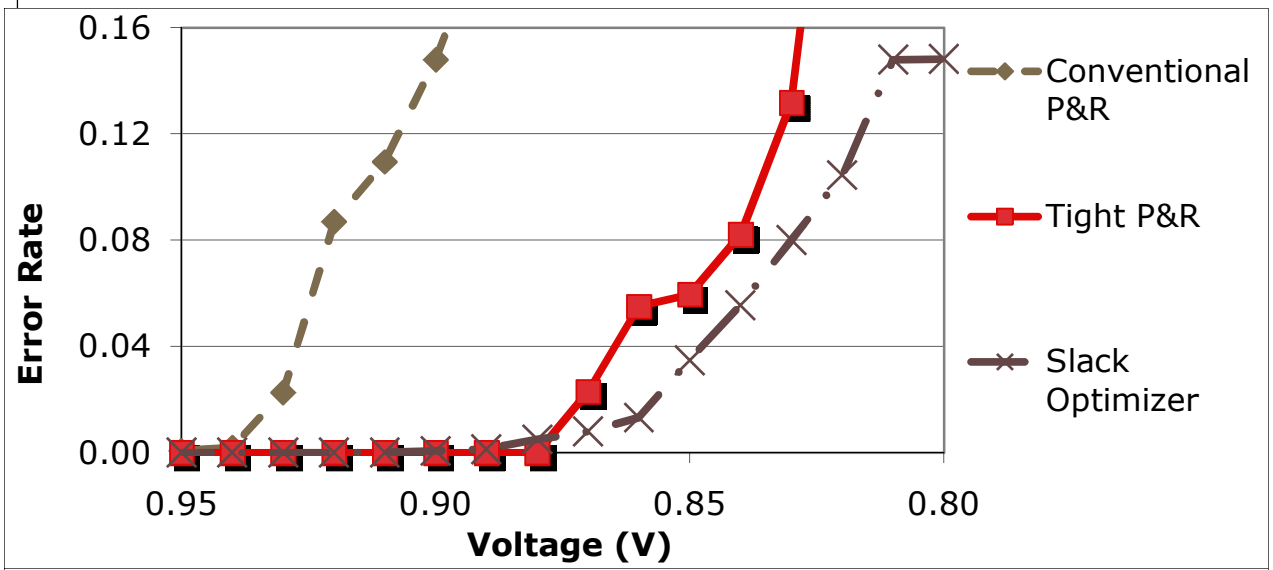


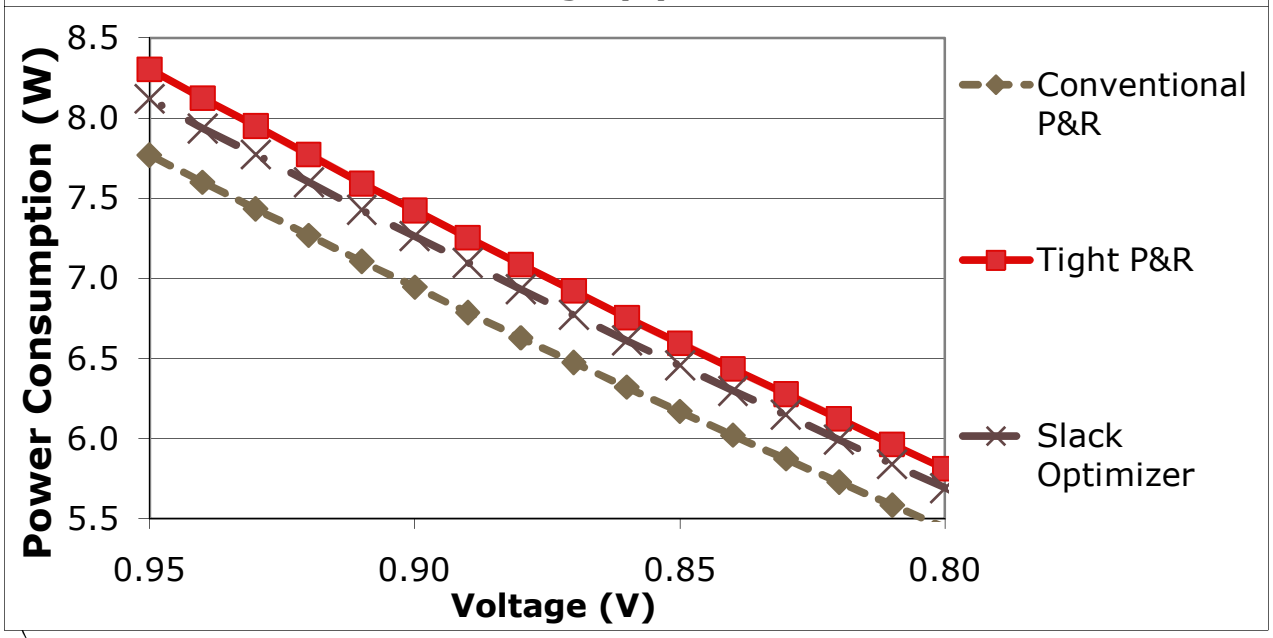
$TG(P_1) = 0.25$ $Slack(P_1) = -0.2$ **0.0**

$TG(P_2) = 0.01$ $Slack(P_2) = 0.1$ **-0.1**

Much smaller error rate at a given voltage => much smaller voltage for a given error rate

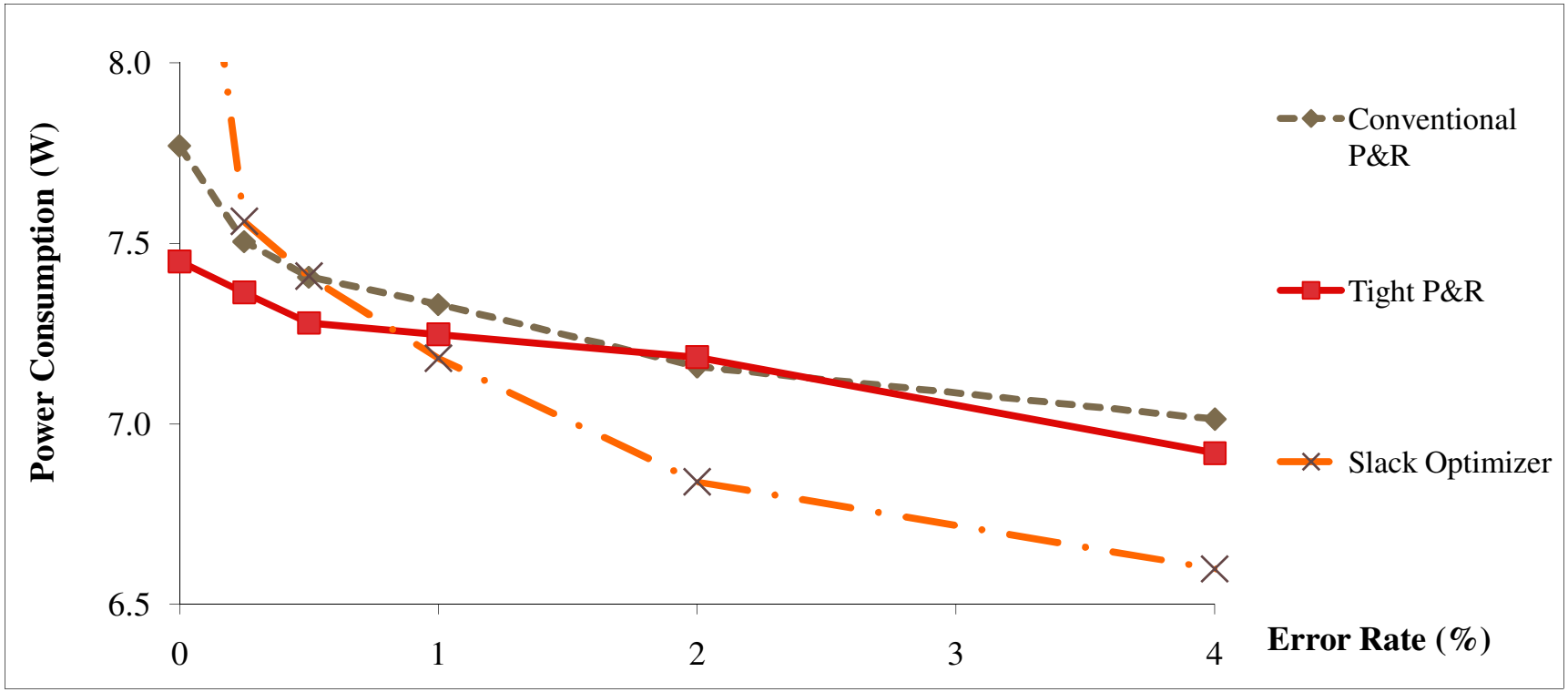
Processor Error Rate and Power





Slack Distributions indeed causes graceful degradation in reliability; Designs with comparable error rates have much higher power/area overheads.

Reliability/Power Tradeoff

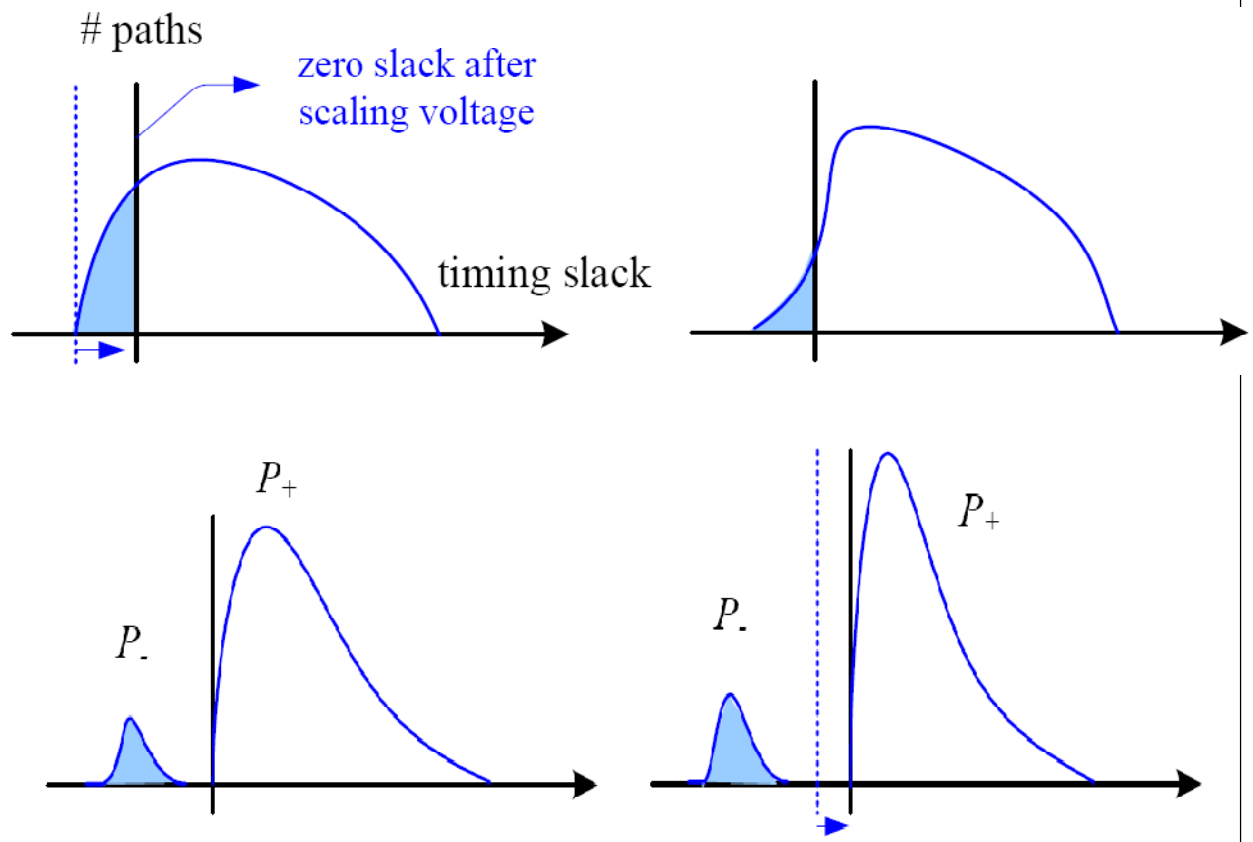


Slack-optimized design enjoys continued power reduction as error rate increases; first methodology that produces designs that allow voltage/reliability tradeoffs.

Recovery-driven Design: Targeting an Error Rate

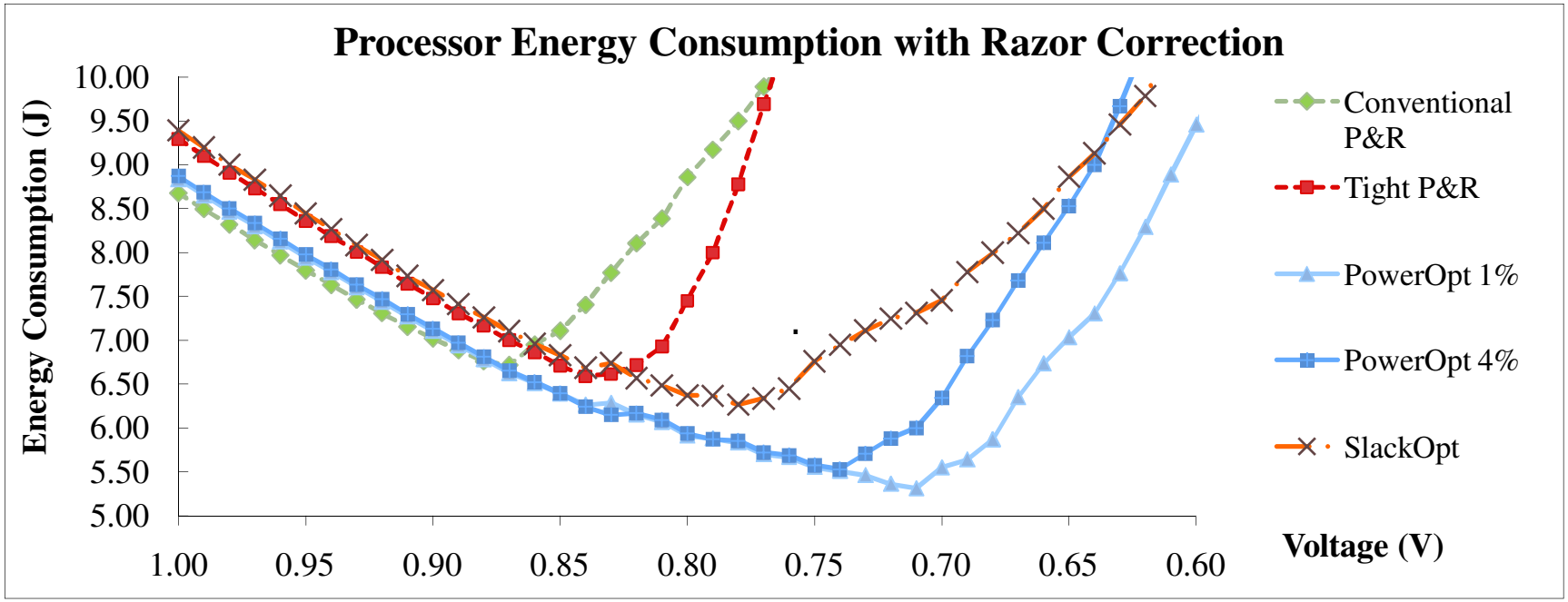
Insight: Save power by de-optimizing to the maximum extent possible the infrequently executed path that will make errors

- Scale down voltage
- Optimize negative slack paths to reduce error rate as much as possible
- Choose paths on which to allow errors until target error rate is reached
- Downsize all cells to maximum extent without increasing error rate



(DAC2010)

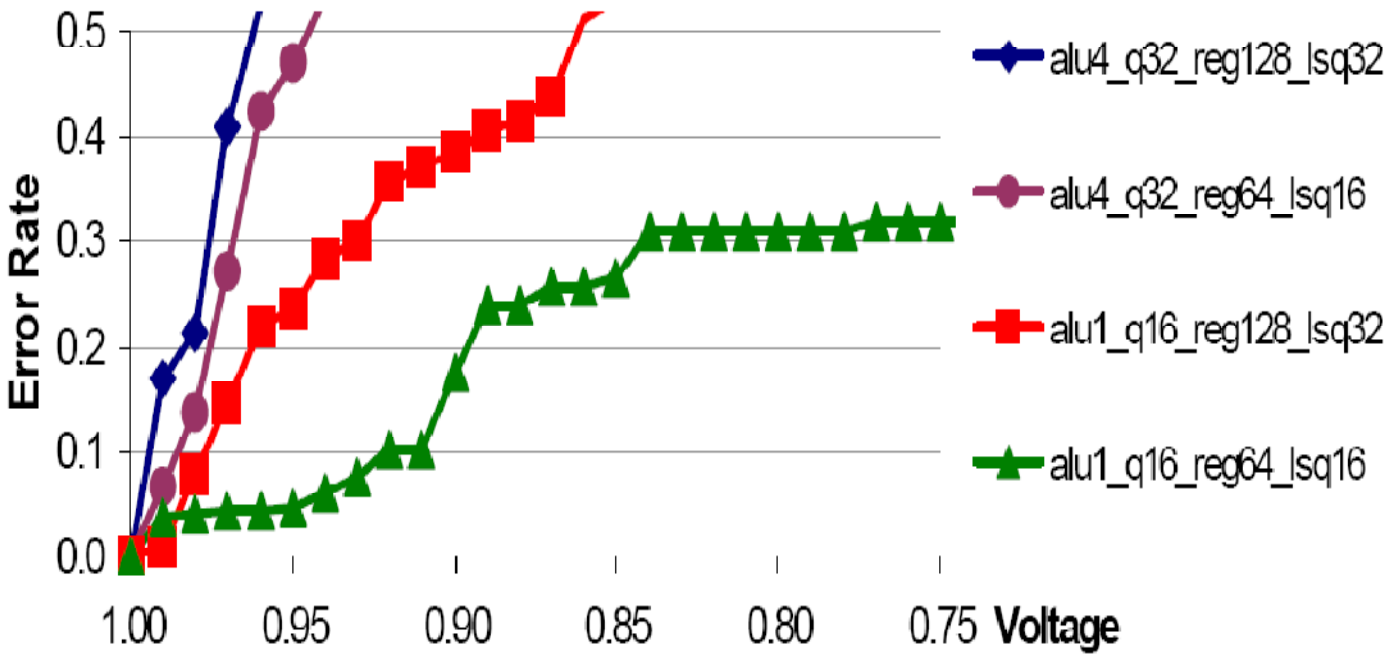
Optimizing for Razor



Minimum Power with Least Area Overhead: Reinforces the benefits of design methodologies to manage the number and nature of errors, even when no errors are exposed to software

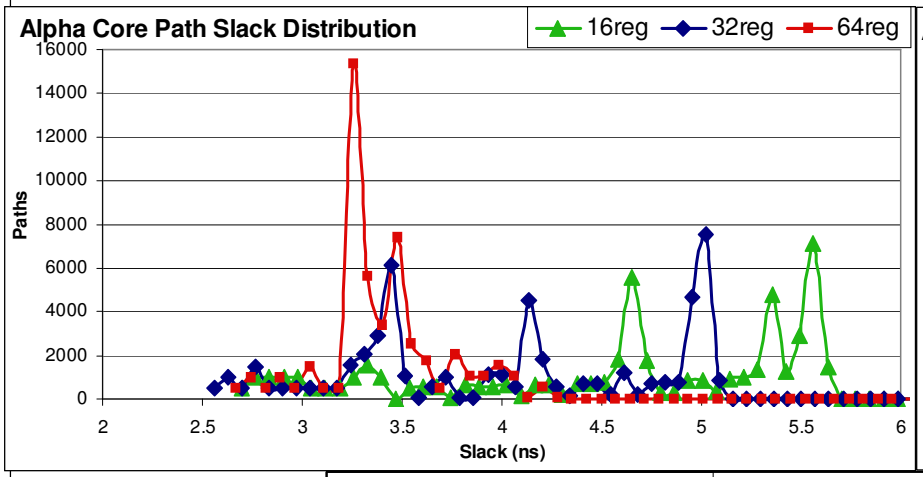
Error Rate Sensitivity to Architecture

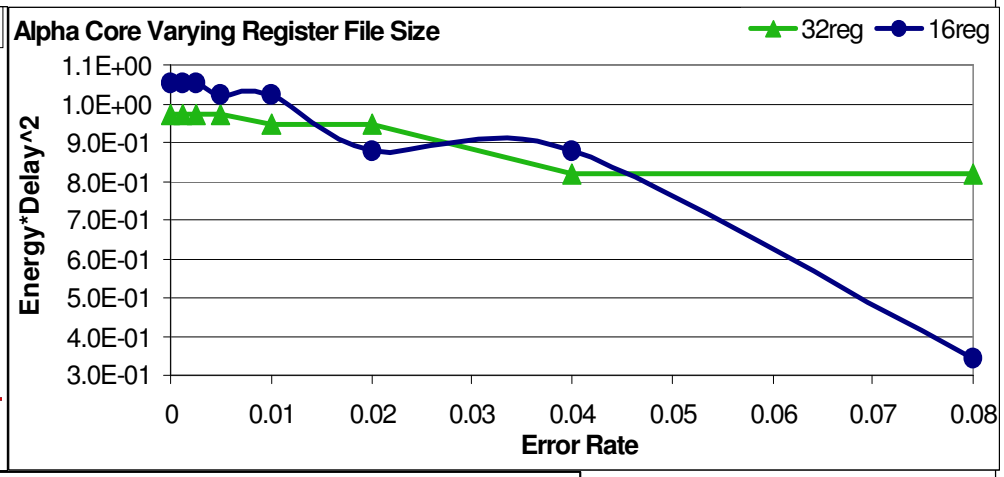
- Changes to microarchitecture affect slack and activity distribution of processor
- Error rate behavior can change significantly with change to architecture

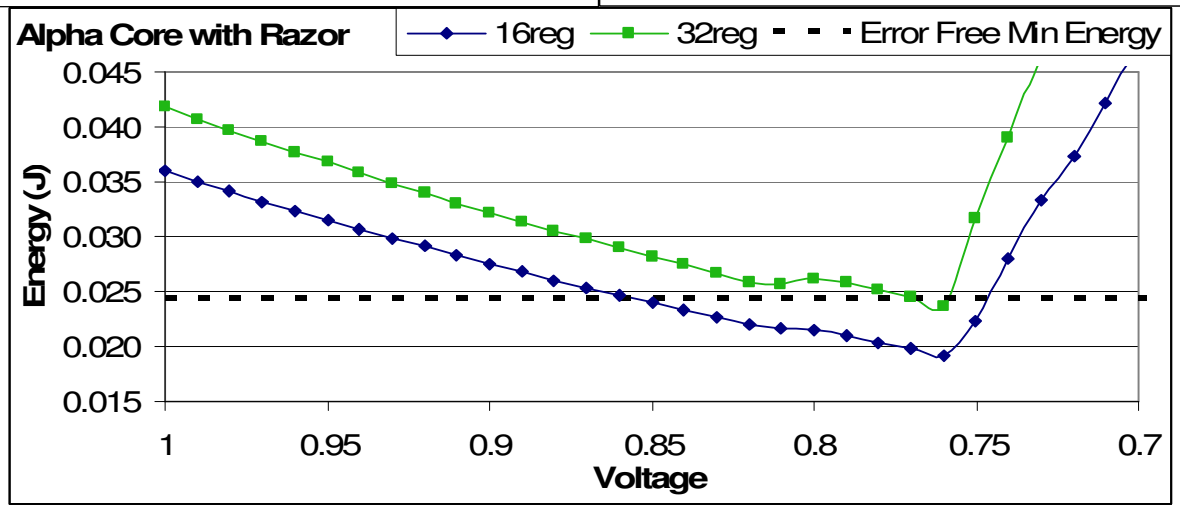


(ICCD 2010; DATE 2010; DAC 2010; CASES2011)

Reshaping the Slack Distribution: Register File Resizing

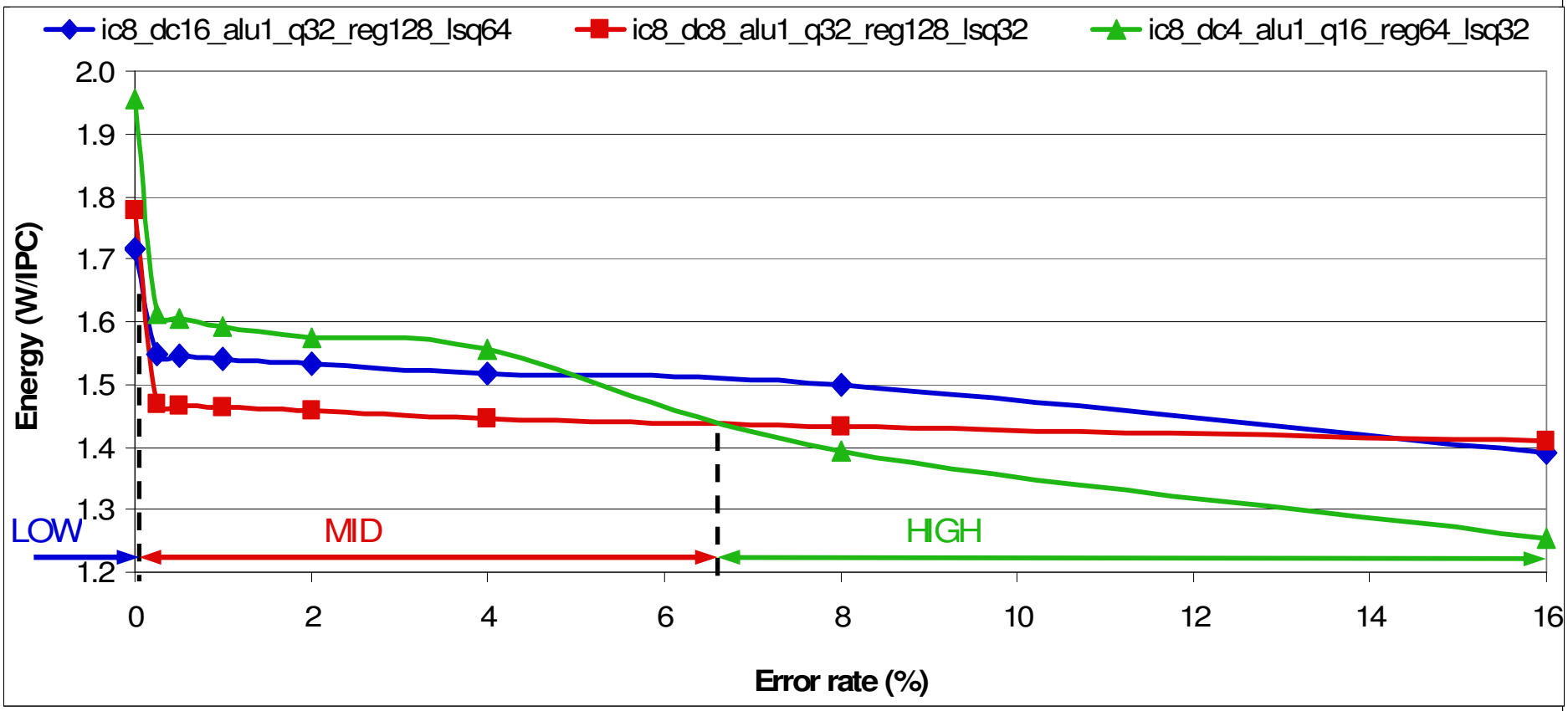






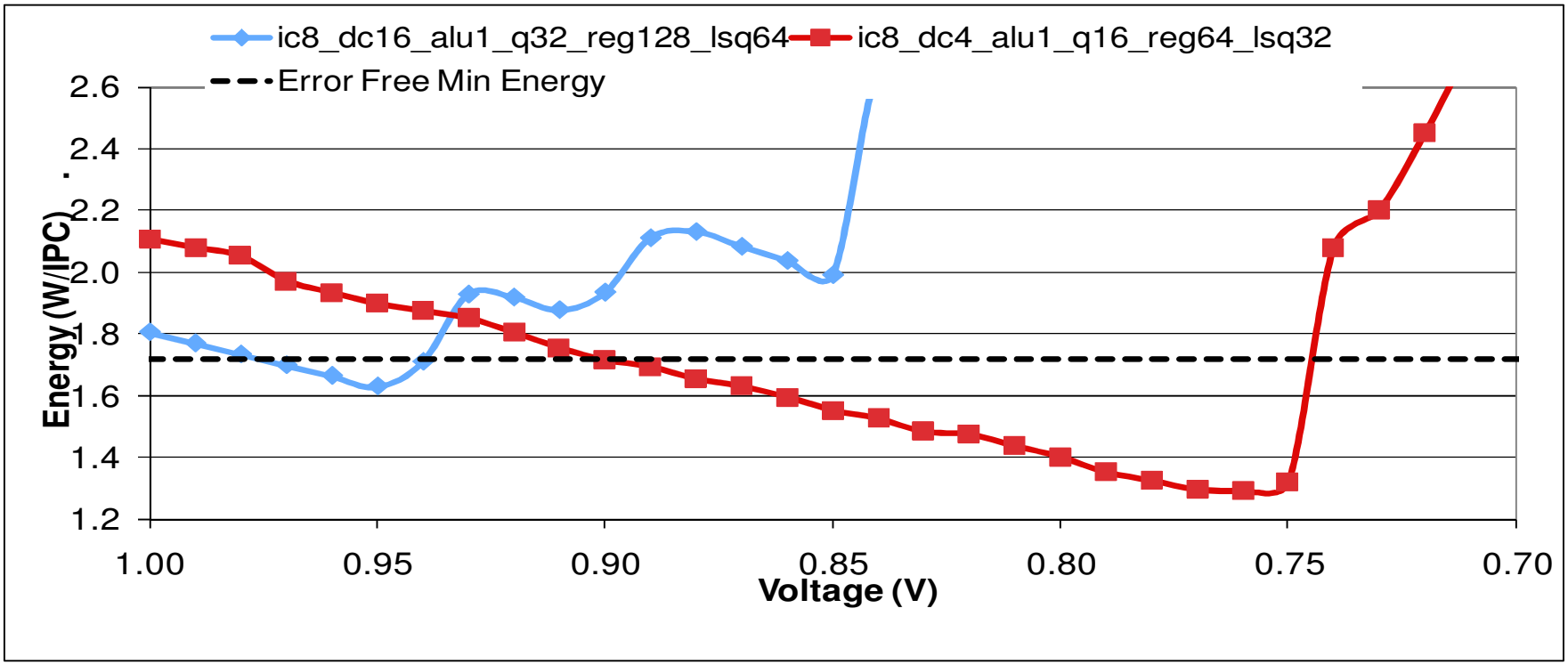
A smaller register file reduces regularity; increases efficiency (>21%) at non-zero error rates; With a large register file energy saving <2%

Design Space Exploration



As error rate increases, smaller regular structures and less complex logic become more efficient

Design Space Exploration



A resilience-optimized architecture achieves significantly higher (>25%) efficiency than correctness-optimized architecture

Summary so far

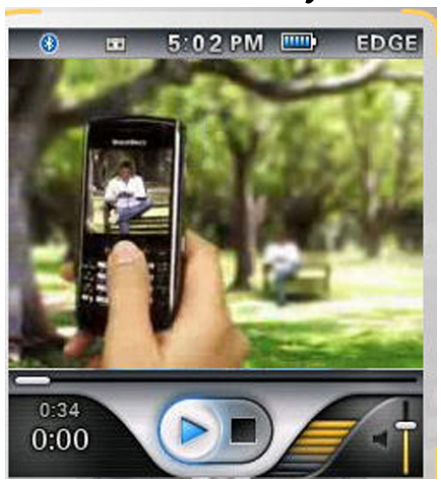
- Too much cost for computing with guarantees
- Processors need to be designed and architected from the ground up to manage the number and nature of errors (stochastic processors) to deal with the non-determinism problem for late-CMOS / post-CMOS technologies
- Conventional design and architecture approaches optimize for correct operation; inadequate when errors are allowed
  - Proposed Stochasticity-aware Architecture / Design Methodologies present significant power savings
- Current Work
  - Application Robustification
  - Approximate Synthesis
  - Dynamic Error management (Towards Best-case design)

Applying Current Hardware for Future Computing Systems

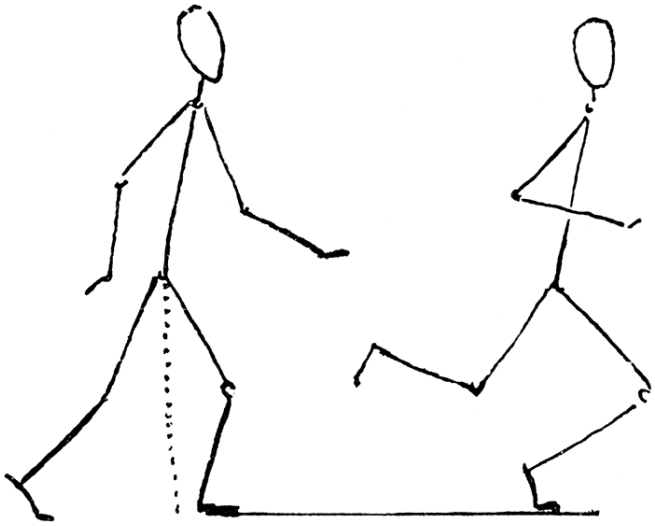
Is there a disconnect between future computing systems and trends and current practices in design and architecture?

Multi-modal Processing

- Increasingly, systems need to support multiple applications with drastically different performance requirements



Local Video Rendering: 20 GOPs/s
Email Browsing: < 1KOPs/s



Sprinting (Usain Bolt): 37Km/hr
Walking: 5Km/hr

- Common case may often be the low performance mode

Optimize for the common case (low energy mode), support capability to deliver high performance when needed

Multi-modal Processing

Aerobic Respiration
(Sitting, Walking, Jogging)

Anerobic Respiration
(Sprinting)

Optimize lifetime energy / average power based on R (duty cycle) and X (scaling factor)

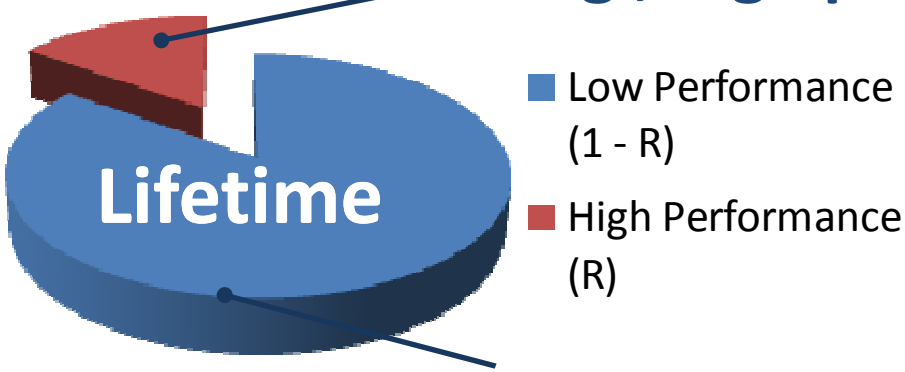
DVFS: Background and Motivation

- DVFS allows adaptation to:
  - Workloads and operating conditions
- DVFS processor operates at multiple power/performance points with different lifetimes
- Lifetime energy can be different in each scenario (R * X)

Different duty cycle (R)

Different frequency scaling (X)

e.g., high perf mode

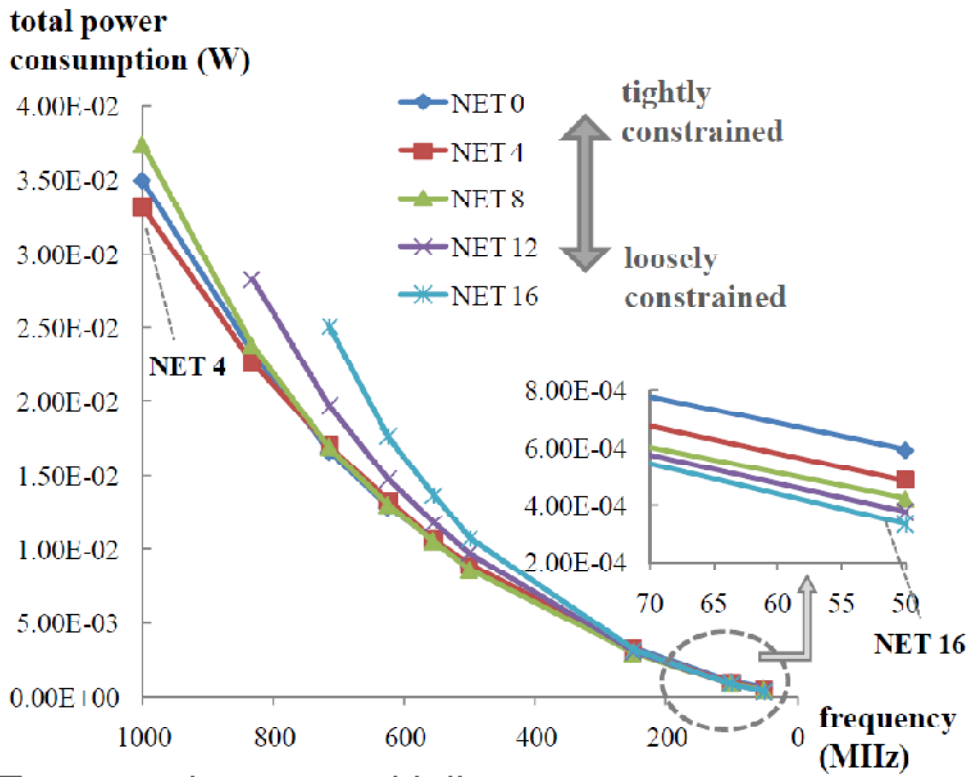


e.g., low perf mode

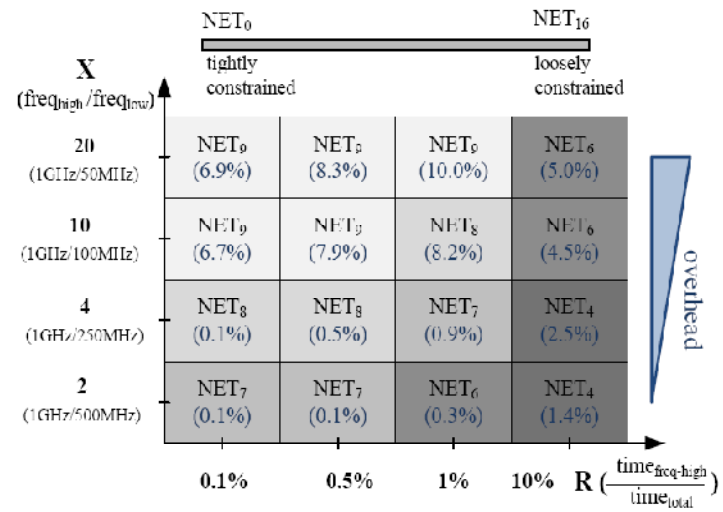
$$X = \frac{\text{clock frequency of high perf. mode}}{\text{clock frequency of low perf. mode}}$$

Motivation for Scenario-Aware Design

- Performance Vs. Power



Testcase: integer multiplier (MUL) of *OpenSPARC T1*



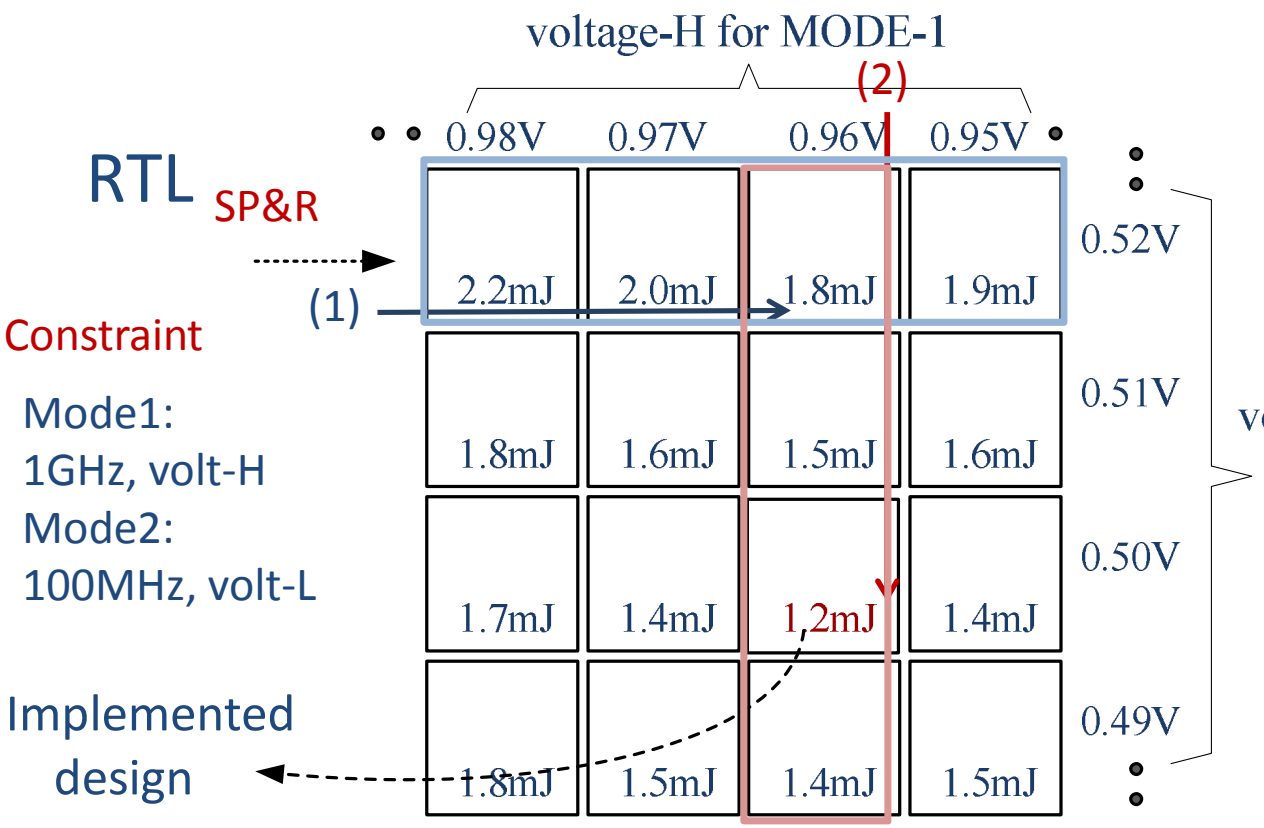
As voltage and freq. are scaled, minimum power netlist changes

Minimum energy netlist changes w.r.t the scenario (R and X)

- To minimize lifetime energy, design constraints should be chosen w.r.t the operating scenario

Context-Aware Multi-Mode Design

- **Conventional EDA tools**: require constraints (freq., voltage) before implementation (which constraints will provide minimum energy?)
- **Context-aware multi-mode design**: postpone constraint finalization and use information about operating scenario
- **Implementation**: find a minimum lifetime-energy design

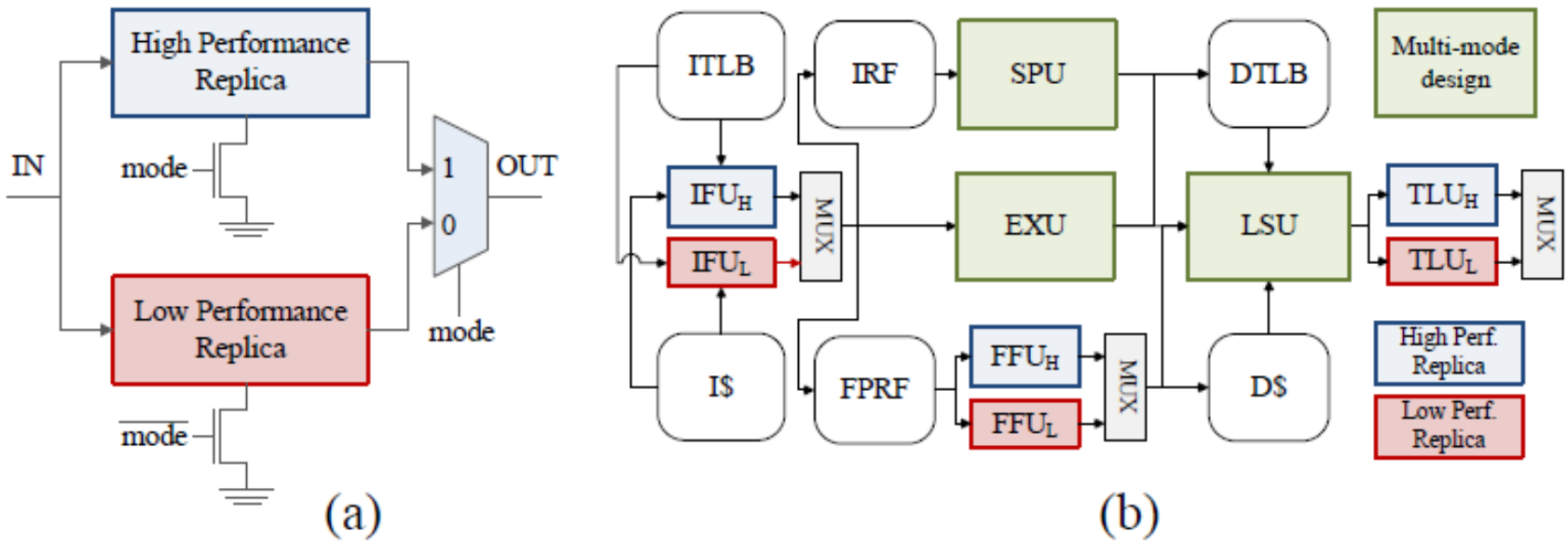


(1) implement w/ 1 GHz and X voltage, find the minimum energy design (w.r.t R, X)

(2) multi-mode optimization for mode 1 and 2, find the minimum energy design

Replication-Based DVFS Design

- Create replicas for each mode
- Replication incurs a large area overhead
- Selective replication
  - Replication benefits are different in each module
    → find optimal usage of replication (knapsack problem)



(a) Overview of replication (one is active while others are power gated)

(b) Example selective replication-based design

Implementation Results

- Lifetime energy comparison

		Context-aware multi-mode			Replication-based design		
(a) Energy reduction w.r.t. conventional MCMM (higher is better)							
MOD	X\R	0.5%	1%	5%	0.5%	1%	5%
EXU	5	1.5%	1.4%	2.3%	8.7%	8.3%	6.6%
	10	4.1%	4.9%	4.4%	9.6%	9.1%	7.4%
	20	13.0%	9.4%	2.6%	15.6%	12.1%	4.1%
FFU	5	3.8%	4.3%	6.9%	5.8%	6.2%	8.1%
	10	4.6%	4.9%	3.3%	6.1%	5.5%	3.4%
	20	4.0%	2.1%	1.8%	3.6%	2.1%	-1.7%
IFU	5	2.7%	2.7%	9.0%	0.4%	1.1%	4.1%
	10	2.7%	3.3%	7.1%	3.6%	3.5%	3.4%
	20	1.5%	2.4%	2.6%	0.3%	0.8%	1.8%
LSU	5	4.4%	5.4%	9.8%	6.2%	7.3%	12.1%
	10	1.6%	1.8%	9.7%	4.8%	6.2%	10.2%
	20	3.6%	2.3%	5.4%	5.5%	6.1%	7.2%
MUL	5	19.5%	12.4%	8.7%	25.4%	24.0%	16.6%
	10	4.0%	2.2%	5.4%	15.1%	14.2%	11.5%
	20	16.2%	6.5%	5.7%	23.1%	19.7%	11.5%
SPU	5	4.0%	3.9%	4.7%	1.4%	1.7%	3.0%
	10	2.6%	3.0%	4.4%	4.2%	3.9%	2.9%
	20	5.9%	4.6%	1.8%	8.0%	5.9%	1.0%
SUM	5	6.0%	5.1%	7.6%	8.2%	8.3%	9.1%
	10	2.0%	3.1%	6.8%	7.0%	7.1%	7.4%
	20	7.1%	4.5%	3.7%	9.1%	7.9%	5.1%

Testcase: OpenSPARC T1

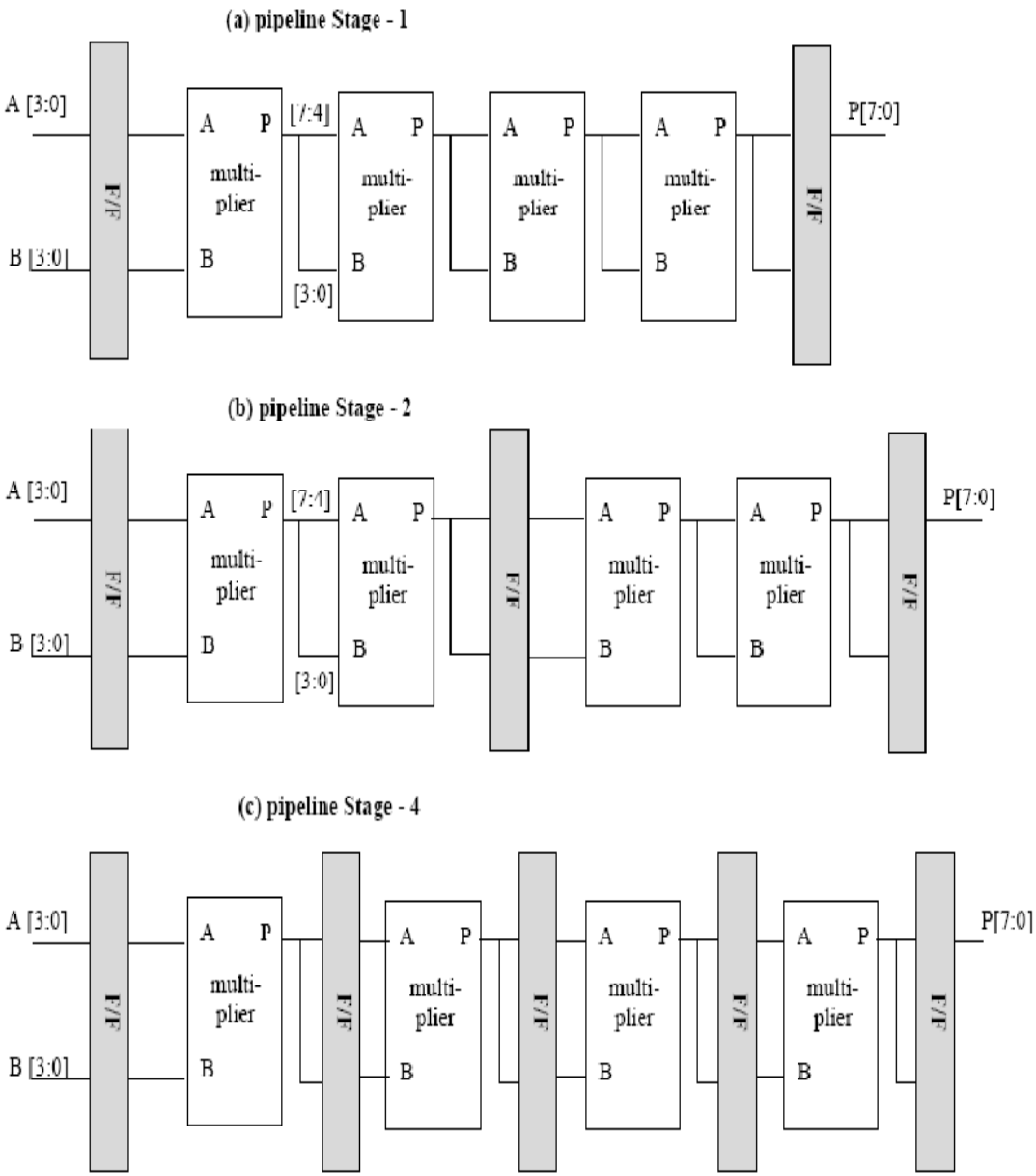
Context-aware multi-mode :
up to 19.5%, 7.6% (avg.) energy reduction

Replication-based design:
up to 25.4%, 9.1% (avg.) energy reduction

EXU, MUL have benefits on replication
IFU, SPU have benefits on multi-mode design

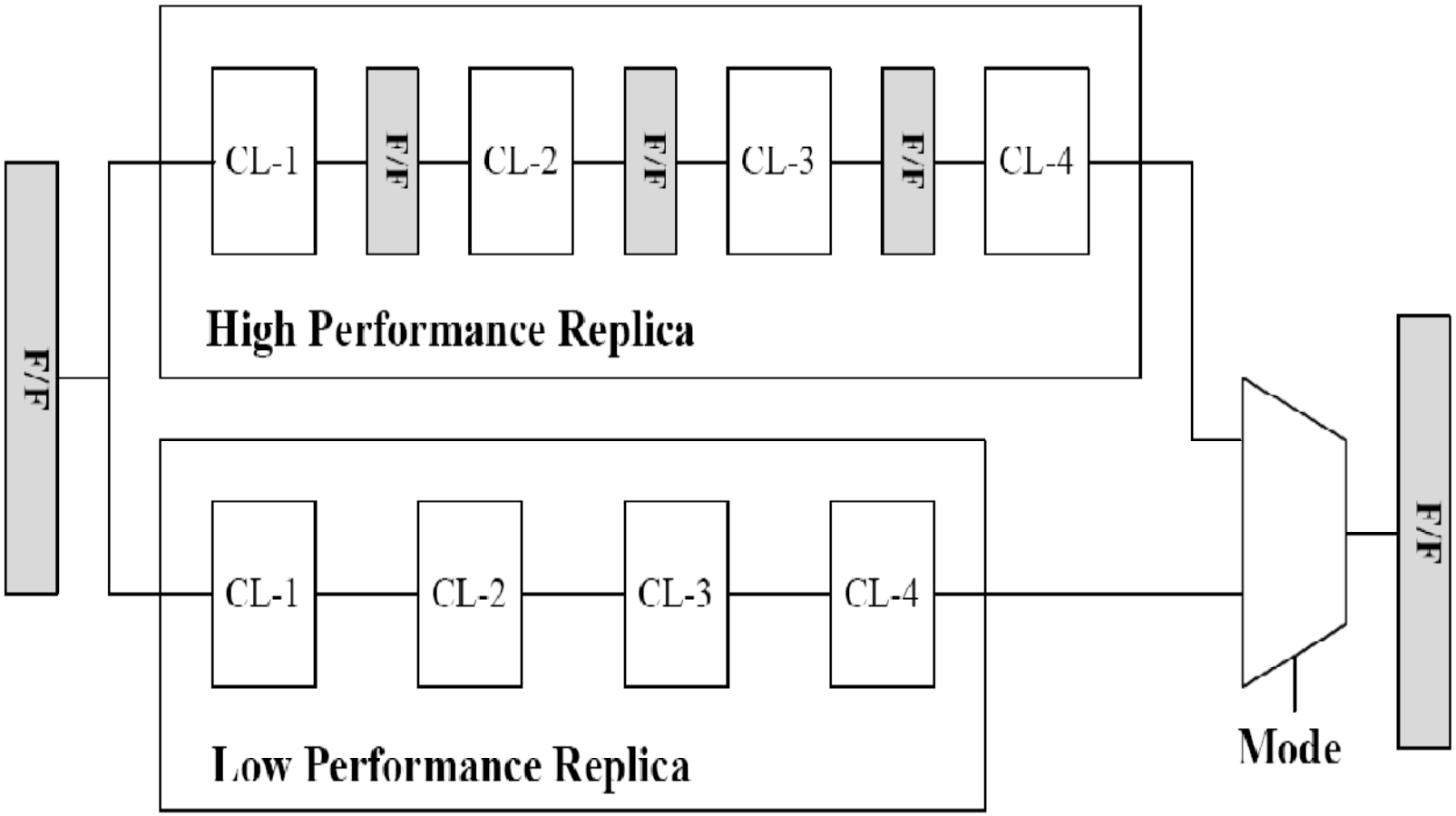
We can select more beneficial approach for each submodule

Optimizing Pipeline Depth



pipeline operating frequency (MHz)	stage-1 operating voltage (V)	stage-1 total power (W)	stage-1 leakage power (%)	stage-2 operating voltage (V)	stage-2 total power (W)	stage-2 leakage power (%)	stage-4 operating voltage (V)	stage-4 total power (W)	stage-4 leakage power (%)
1000	N/A	N/A	N/A	1.16	5.15E-04	3.4	0.84	3.52E-04	3.0
500	N/A	N/A	N/A	0.76	1.09E-04	4.7	0.65	1.07E-04	5.2
200	0.96	4.53E-05	4.9	0.58	2.40E-05	9.3	0.53	2.99E-05	11.3
100	0.74	1.37E-05	7.7	0.51	1.06E-05	15.3	0.47	1.29E-05	19.9
50	0.62	5.04E-06	13.5	0.45	5.46E-06	21.8	0.41	6.10E-06	31.0

Replication-Based Design + Pipeline Optimization

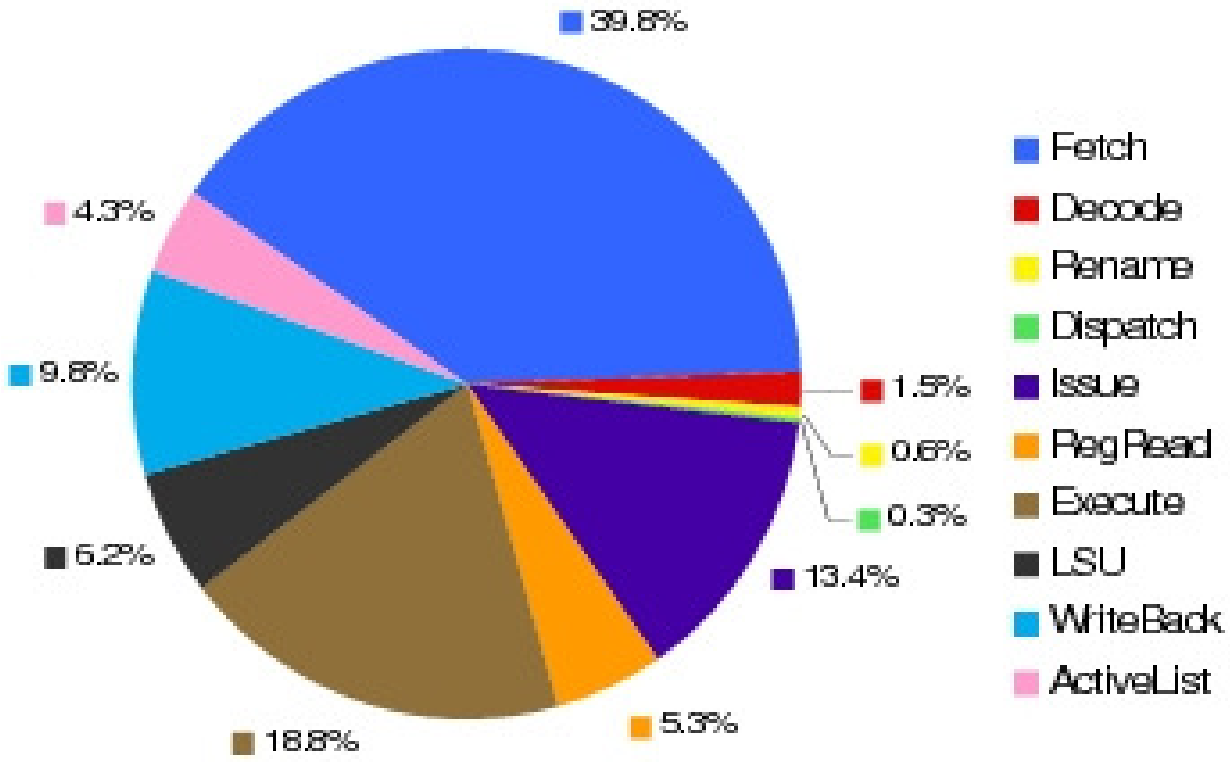


- Different replicas have different optimal pipeline depths. We observed many other microarchitectural tradeoffs as well.

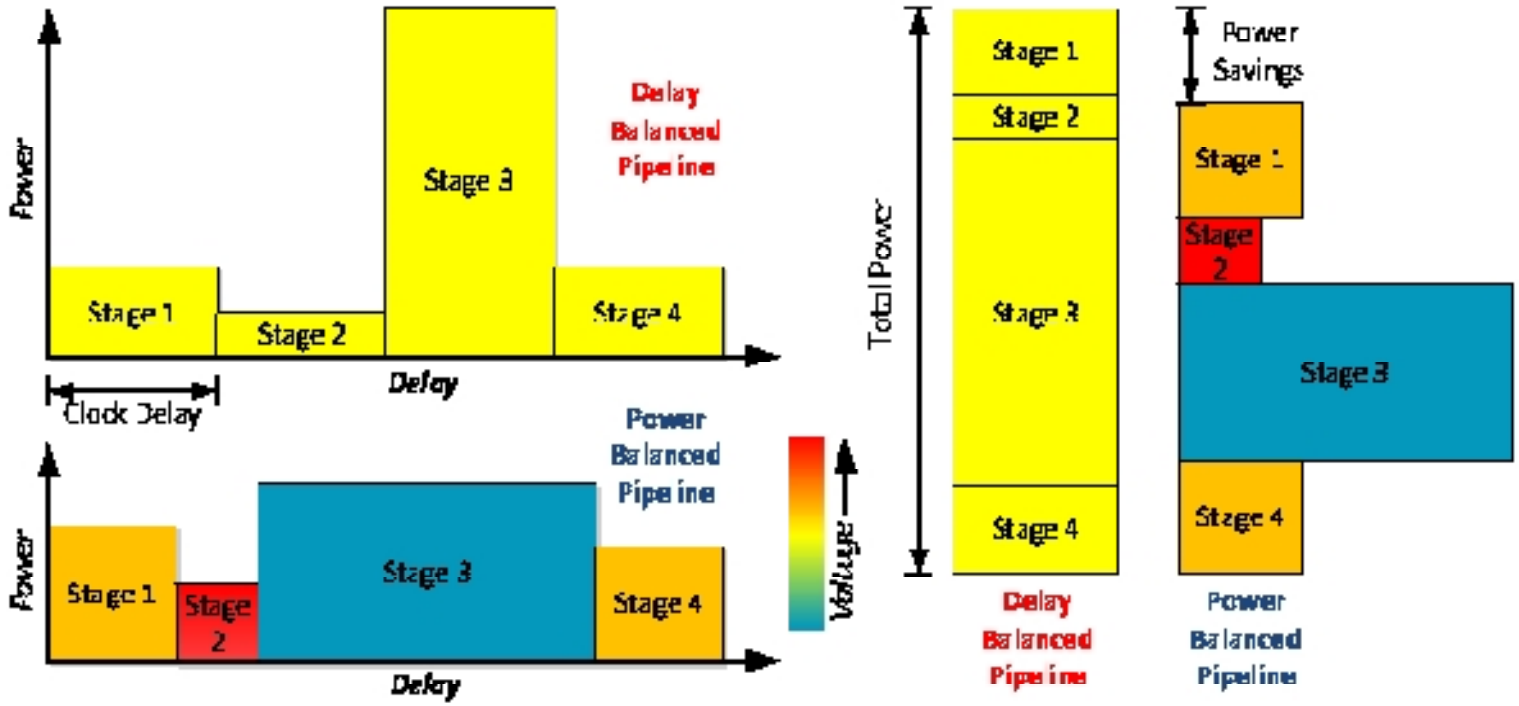
Current Work looks at different microarchitecture + design-level techniques for multimodal processing

The Power Imbalance

- Fetch power is > 100x that of the Dispatch stage in the FabScalar pipeline



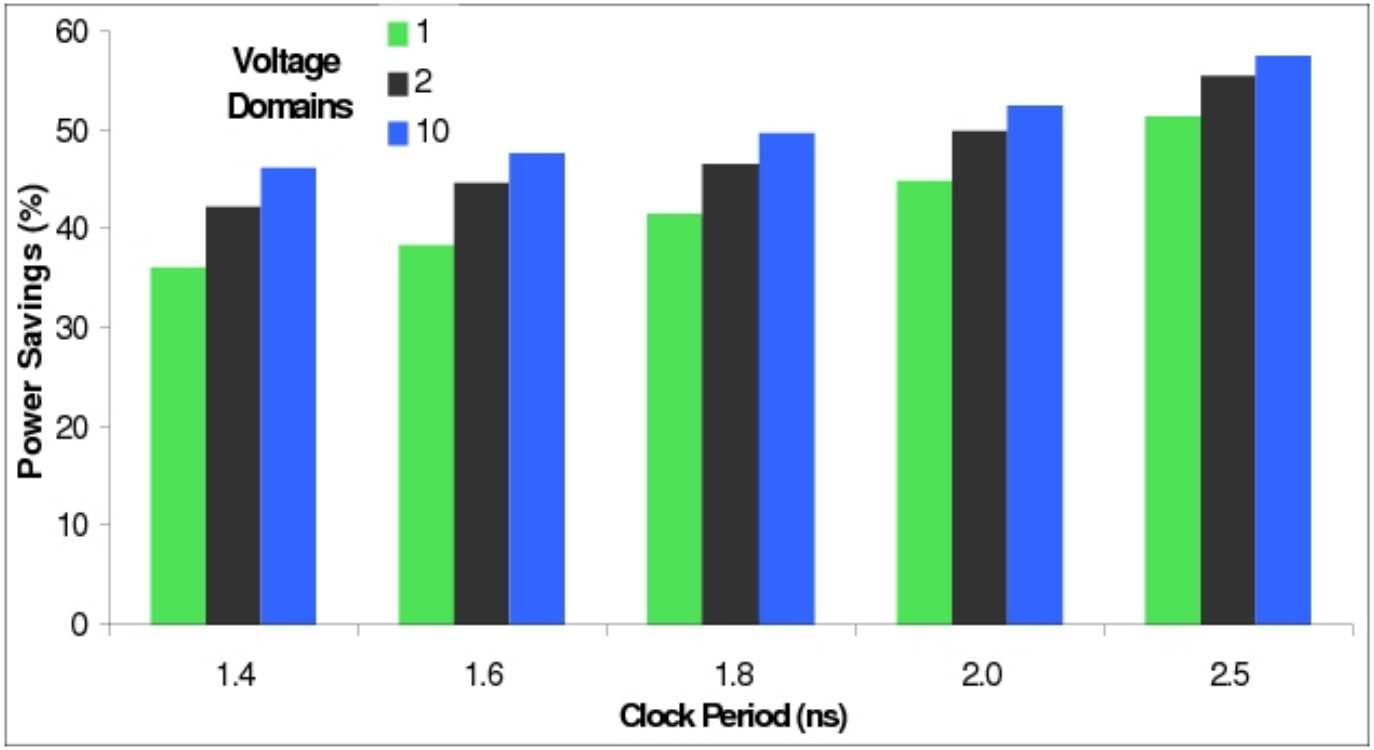
Power Balanced Pipelining



Cycle Stealing performed in conjunction with choosing different operating voltage for different stages or choosing different timing constraints for different stages

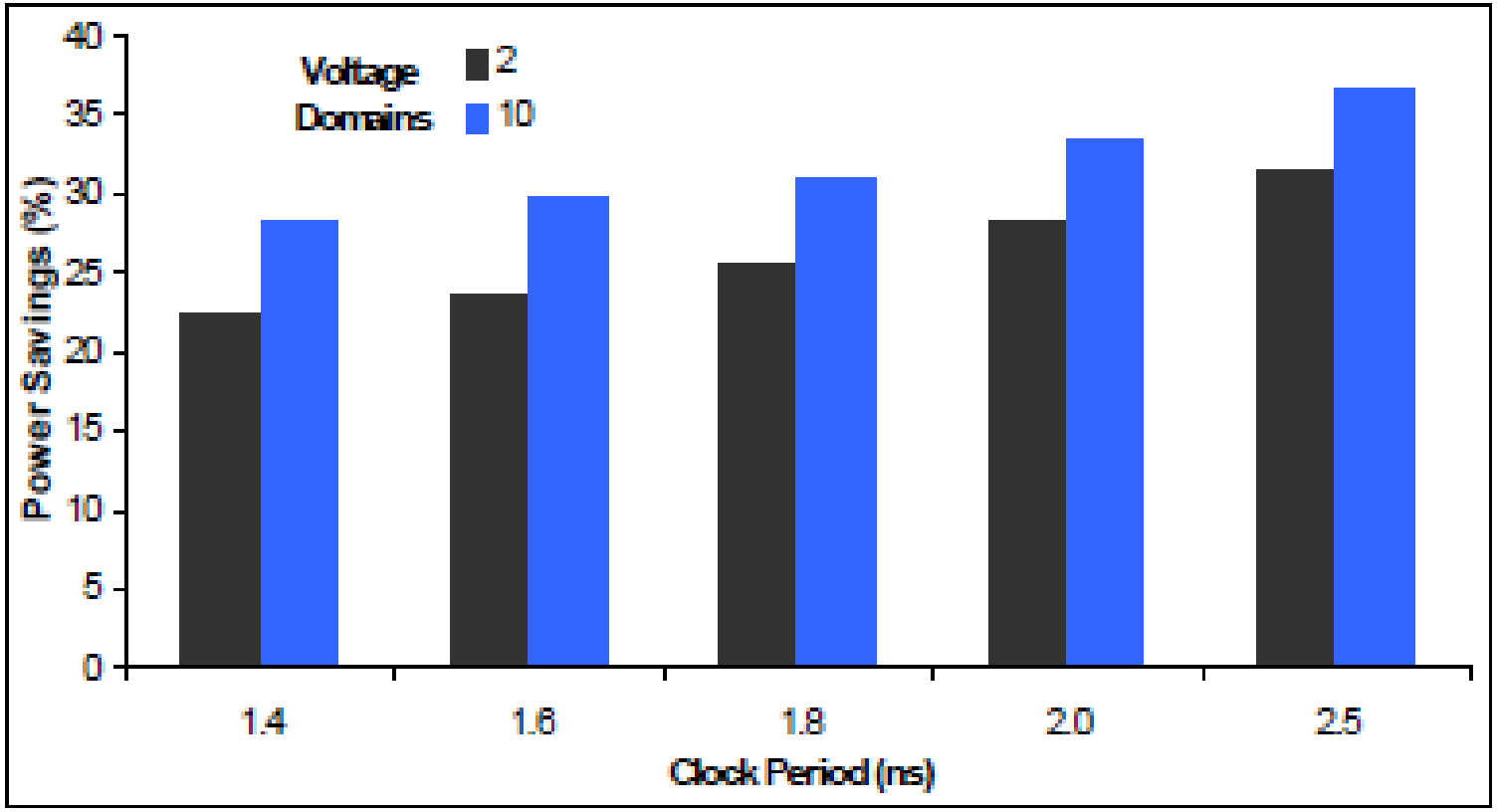
Design-level Power Balancing

- Resynthesize complex stages with a relaxed timing constraint rather than using a lower voltage; this can also be combined with a multiple voltage rail design



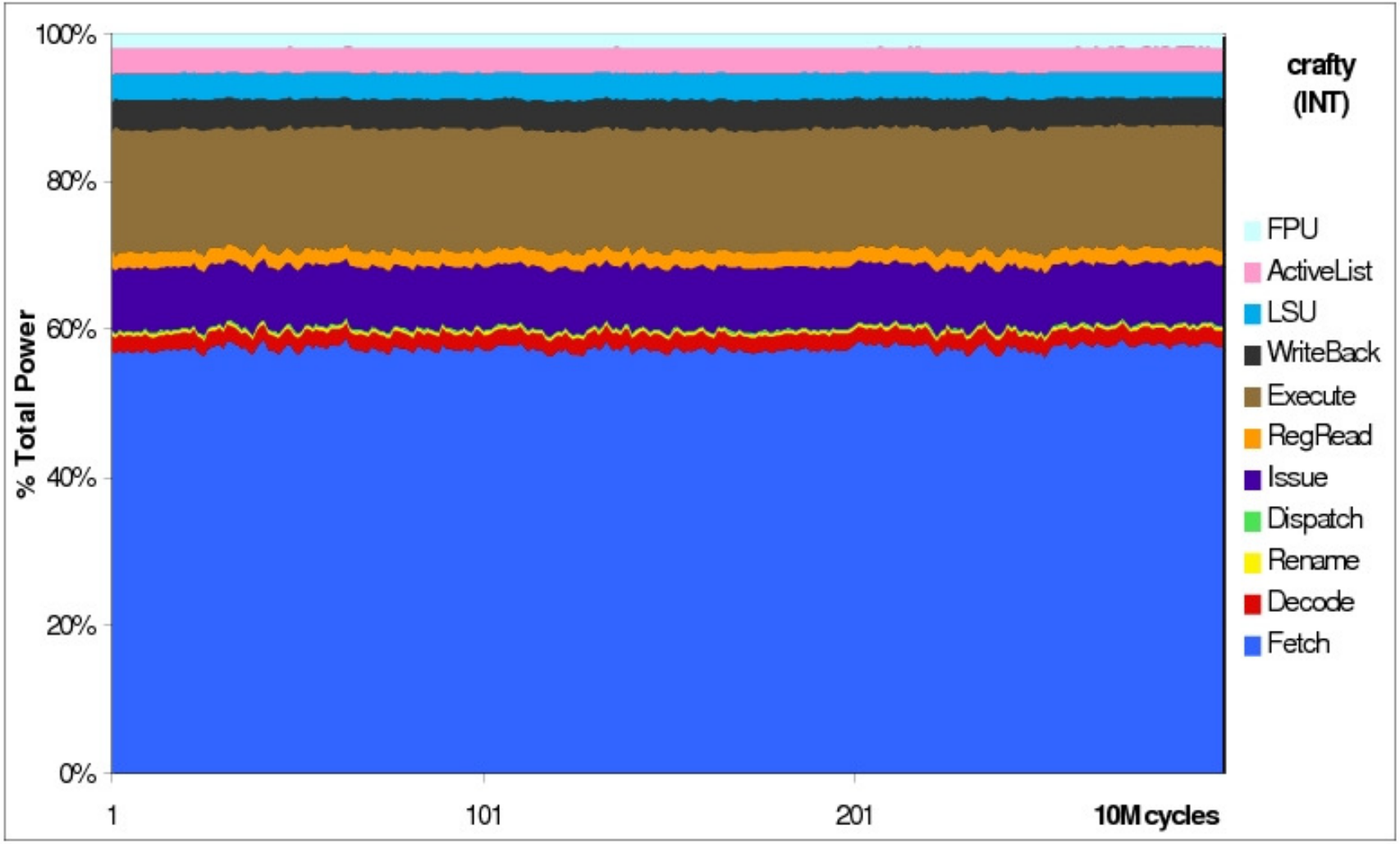
Cycle Stealing Implemented statically in the clock network during the Clock Network synthesis

Power Balancing through Post-silicon Tuning

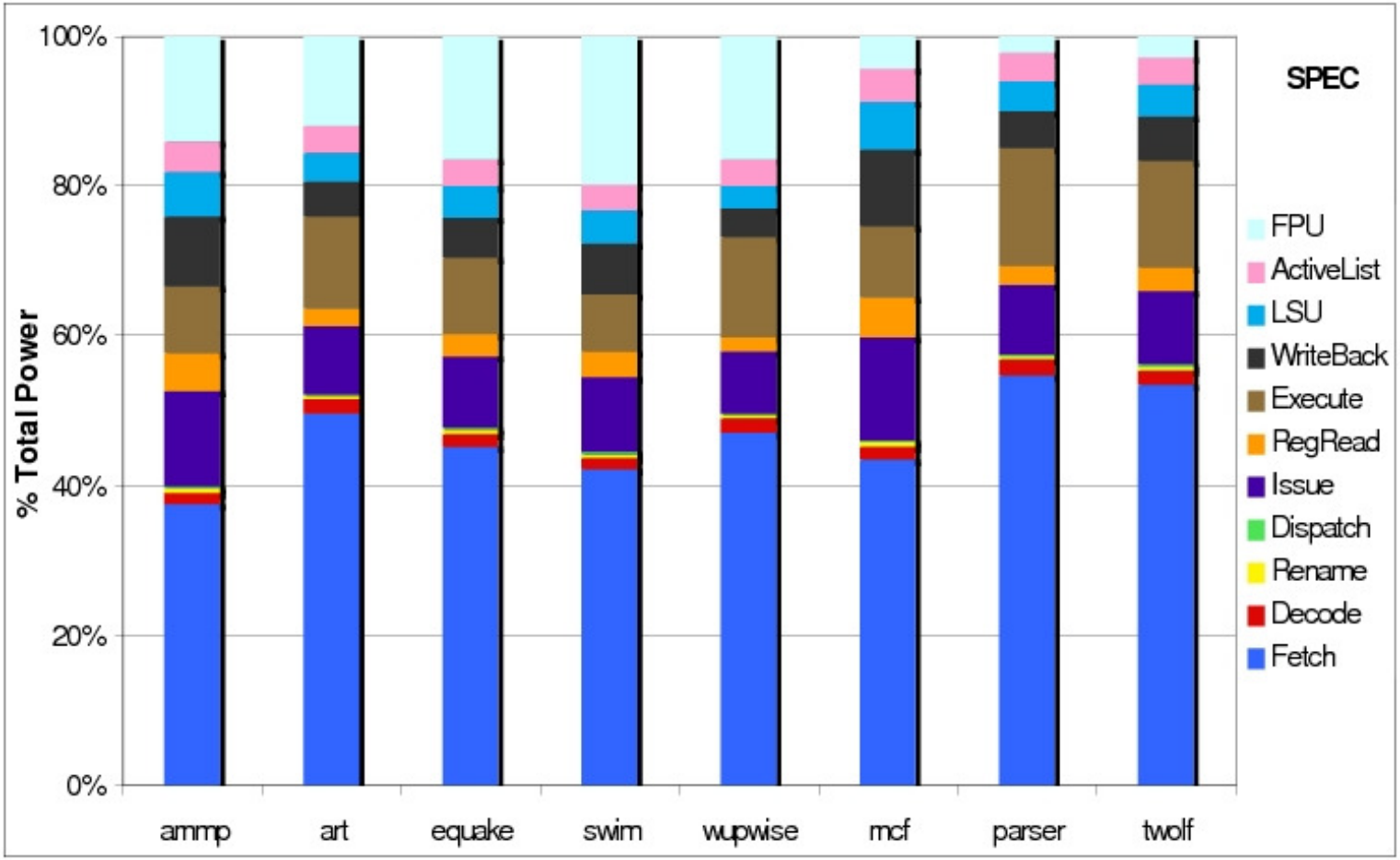


Cycle Stealing Implemented Using Tunable Delay Buffers

Relative power is basically constant for any given workload

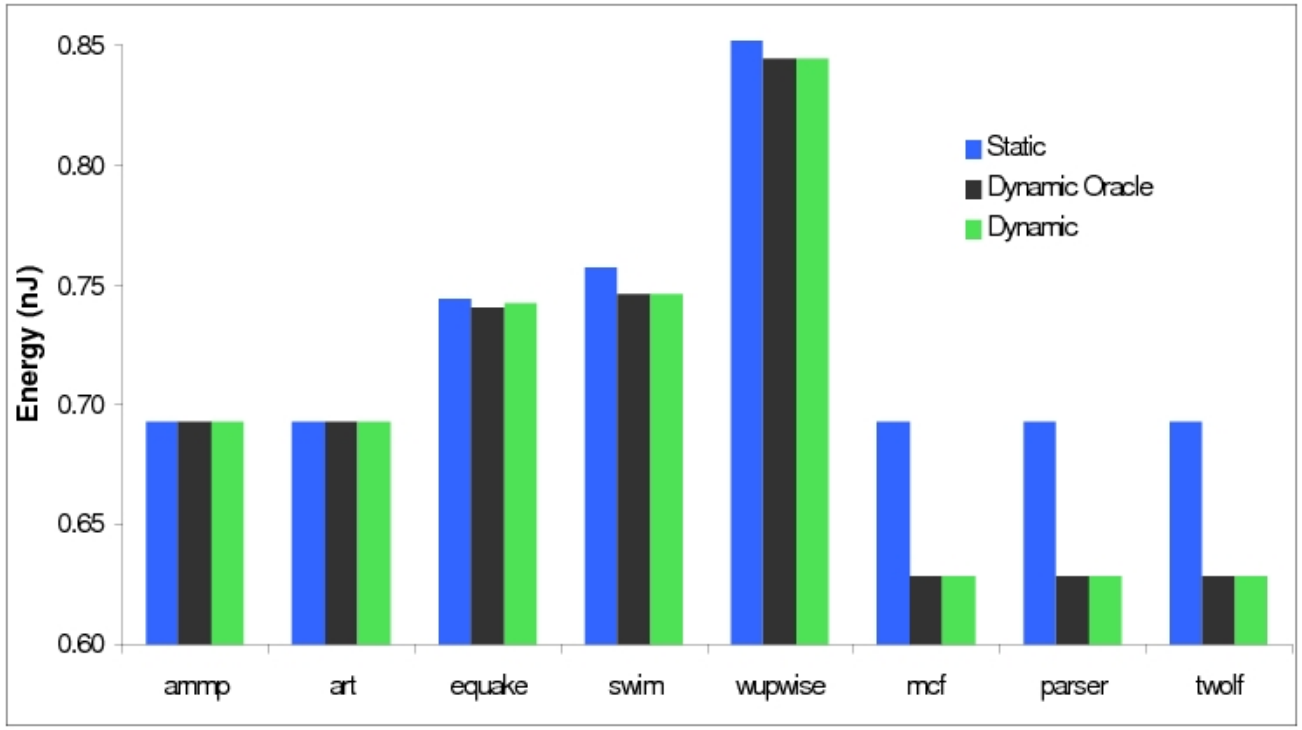


Dynamic optimization: Preserving savings when a stage's power footprint changes drastically



Static vs. Dynamic Power Balancing

- At most 10% difference in savings
- Dynamic adaptation only requires two operating modes



Summary

- Computing Trends and technology limits require very aggressive approaches to power reduction
- Stochastic Computing is a promising approach to lower power computing for future applications and technologies
- Several opportunities exist to align current practices in hardware design and architecture with emerging computing trends
  - Multimodal Processing
  - Power Balanced Pipelining