Slack Redistribution for Graceful Degradation Under Voltage Overscaling

Andrew B. Kahng[†], **Seokhyeong Kang**[†],
Rakesh Kumar[‡] and John Sartori[‡]

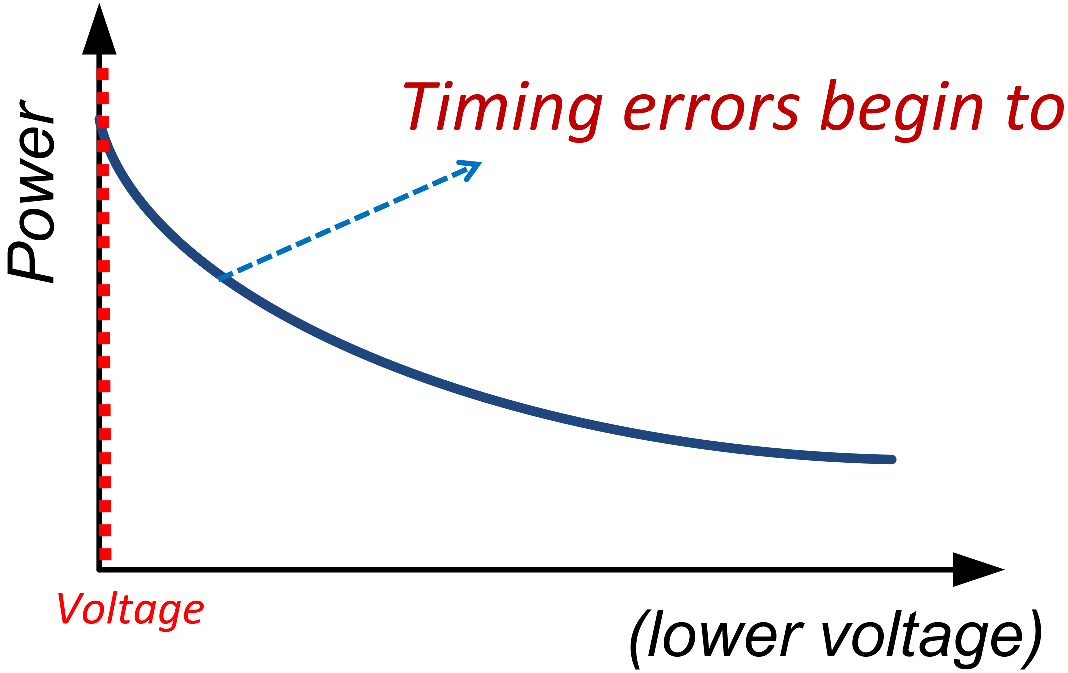
[†]VLSI CAD LABORATORY, UCSD
[‡]PASSAT GROUP, UIUC

Outline

- Background and motivation
  - Voltage scaling and BTWC designs
  - Limitation of Traditional CAD Flow
- Power-Aware Slack Redistribution
  - Our design optimization goal
  - Related work: *BlueShift*
  - Our Heuristic
- Experimental Framework and Results
  - Design methodology
  - Testbed
  - Results and analysis
- Conclusions and Ongoing Work

Reducing Power with Voltage Scaling

- Power is a first-order design constraint
- Voltage scaling can significantly reduce power
- Voltage scaling may result in timing violations



- *Voltage scaling is limited because of timing errors*

Better-Than-Worst-Case Design

- Better-Than-worst-case (BTWC) design approach
  - Optimize for normal operating conditions
  - Trade off reliability and power/performance
  - Have error detection/correction mechanism (e.g., *Razor**)

Traditional IC design

- Does not allow timing errors in STA
- Fixed target frequency and operating voltage

BTWC design

- Error correction architecture allows timing errors
- Overclocking or voltage overscaling

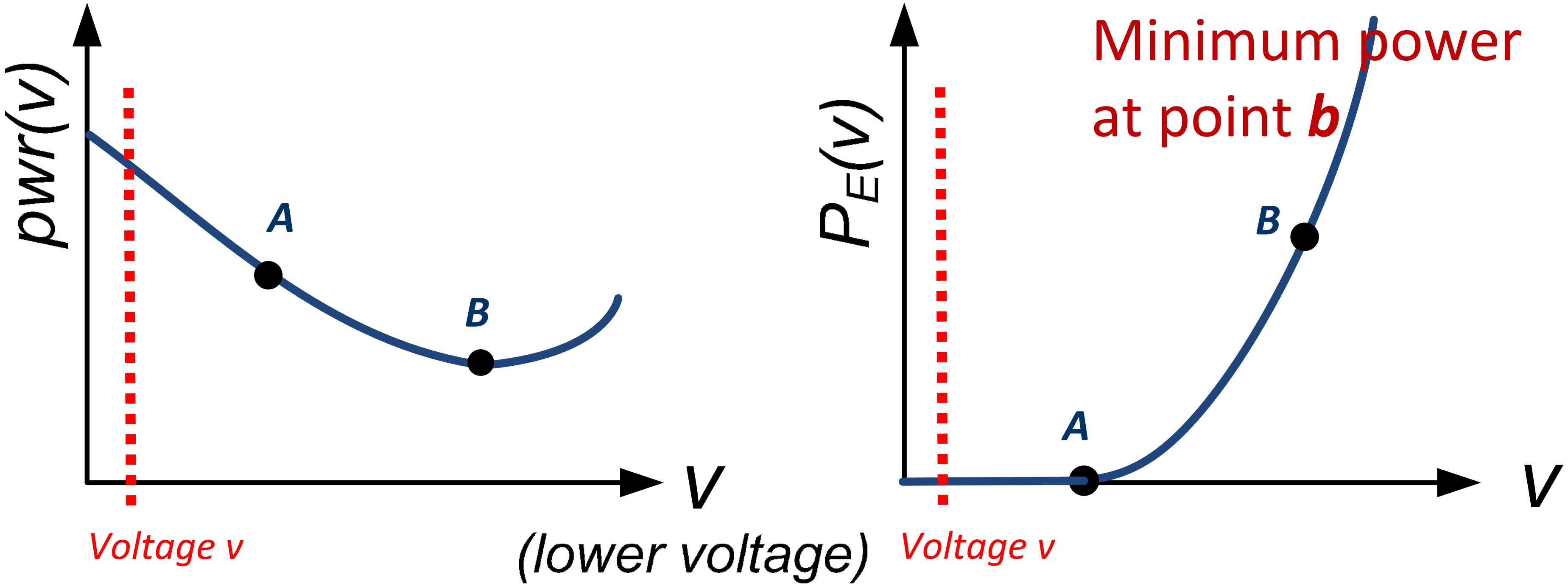
CPU, Heal Thyself...

* Ernst et al. "Razor: A low power pipeline based on circuit-level timing speculation", *Proc. MICRO* 2003.

- *BTWC design allows tradeoffs between reliability and power*

Voltage Scaling with Error Correction

- Error correction incurs power overhead



Minimum power at point ***b***

pwr(v) : Power consumption at ***v***

P_E(v) : Error rate at voltage ***v***

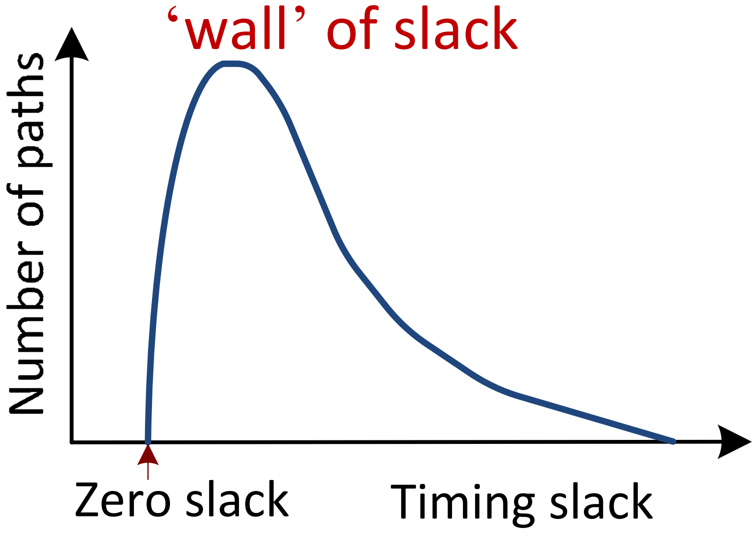
- *Overscaling is possible for Better-Than-Worst-Case designs*

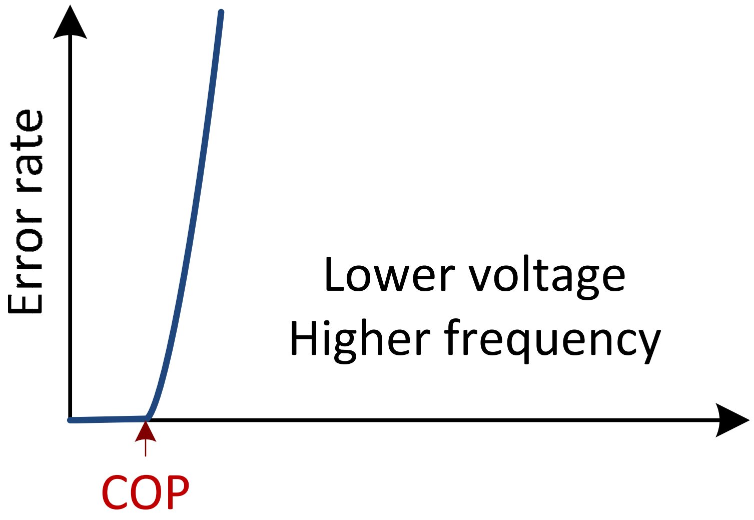
Limitations of Traditional CAD Flow

- Conventional designs exhibit critical operating points
  - Many paths have near-critical slack → *wall of (critical) slack*
  - Scaling beyond *COP* causes massive errors that cannot be corrected
  - Conventional designs fail critically when voltage is scaled down

- Error rate should be increased gracefully :

  ***gradual slope* slack**



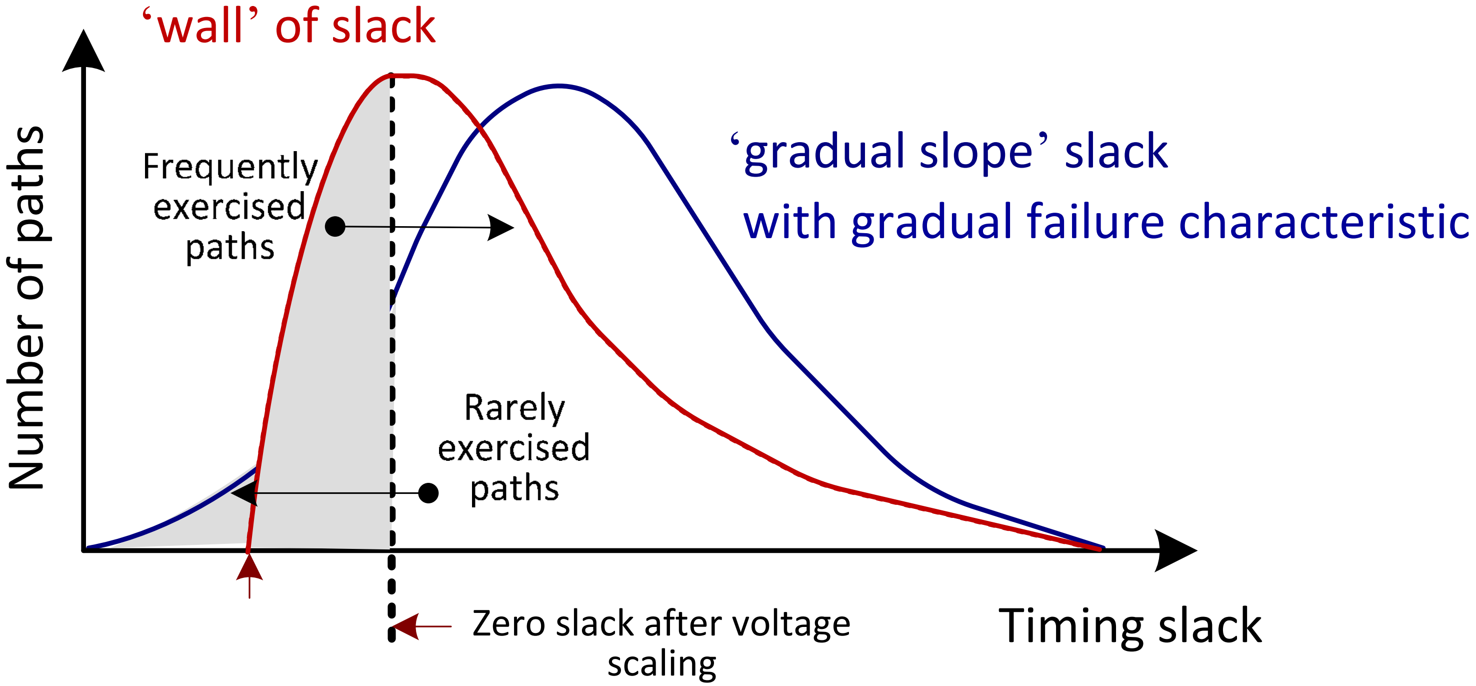


Outline

- Background and motivation
  - Voltage scaling and BTWC designs
  - Limitation of Traditional CAD Flow
- **Power-Aware Slack Redistribution**
  - **Our design optimization goal**
  - **Related work: *BlueShift***
  - **Our Heuristic**
- Experimental Framework and Results
  - Design methodology
  - Testbed
  - Results and analysis
- Conclusions and Ongoing Work

Our Design Optimization Goal

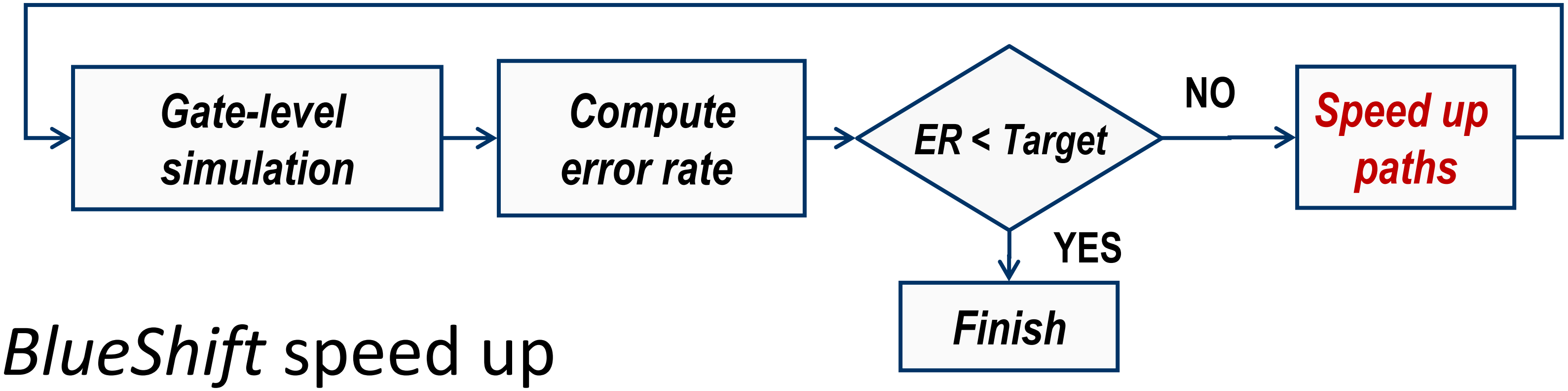
- **Problem**: Minimize power for a given error rate
- **Goal:** Achieve a ‘***gradual slope***’ **slack distribution**
- **Approach:** Frequently-exercised paths: upsize cells
  Rarely-exercised paths: downsize cells



- *We make a gradual slope slack distribution*

Related Work: *BlueShift*

- *BlueShift** : maximize frequency for a given error rate

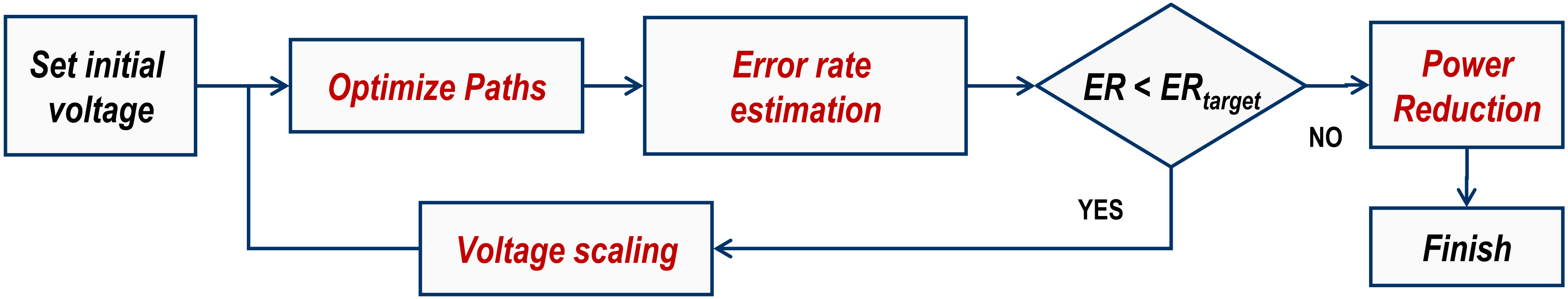


- *BlueShift* speed up
  - Paths with the highest frequency of timing errors
  - FBB (forward body-biasing) & Timing override
- Limitation
  - Repetitive gate level simulation – impractical
  - Design overhead of FBB
- *BlueShift is impractical with modern SOC designs*

* Greskamp et al. "Blueshift: Designing processors for timing speculation from the ground up", *HPCA* 2009

Our Heuristic

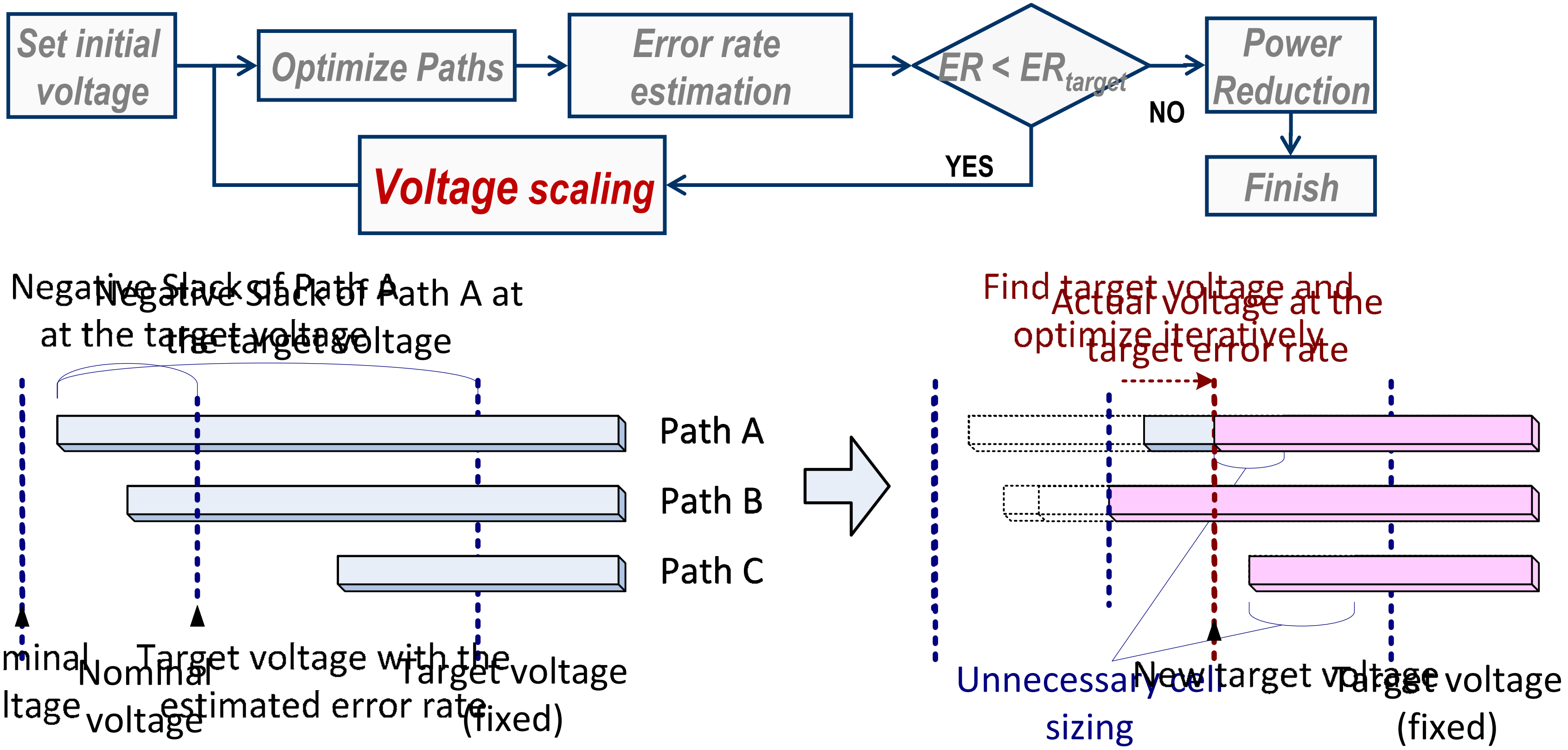
- Optimize slack distribution by cell swaps, exploiting switching activity information
- Iteratively scale target voltage the until error rate exceeds a target, and optimize negative slack paths



- *Our heuristic:*

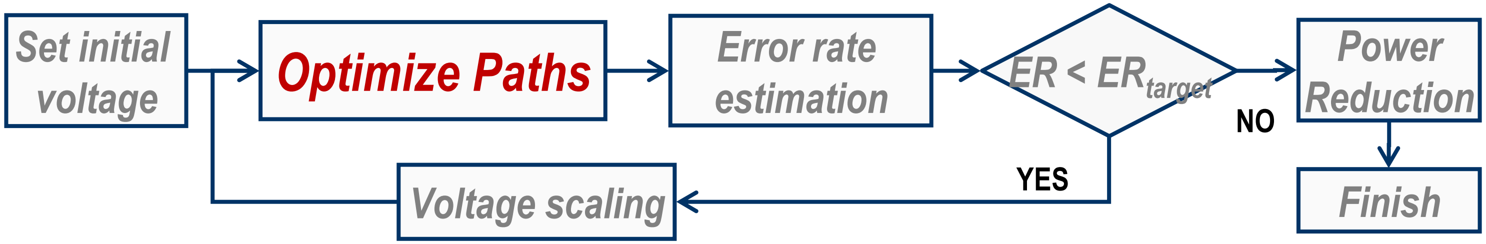
  Voltage scaling → Optimize paths → Power reduction

Heuristic Implementation – Voltage Scaling



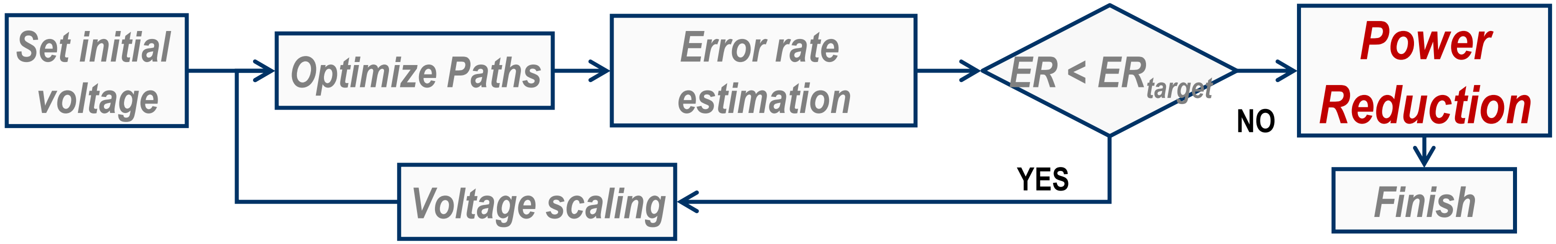
- Optimize with fixed target voltage
- Lower voltage incrementally
- Load a pre-characterized library at each voltage point
- *With iterative voltage scaling, we can find minimum operating voltage*

Heuristic Implementation – Optimize Paths



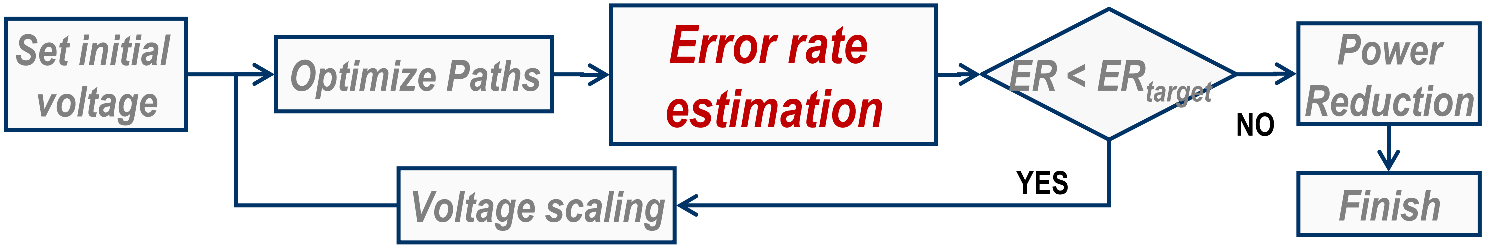
- **Main idea:** increase slack of frequently-exercised paths in order of increasing switching activity
- **Procedure**
  1. Pick a critical path ***p*** with maximum switching activity
  2. Resize cell instance ***c_i*** in ***p***
  3. If slack of path ***p*** is not improved, cell change is restored
  4. Repeat 2. ~ 3. for all cell instances in path ***p***
  5. Repeat 2.~ 4. for all critical paths
- *OptimizePaths procedure reduces error rates and enables further voltage scaling*

Heuristic Implementation – Power Reduction



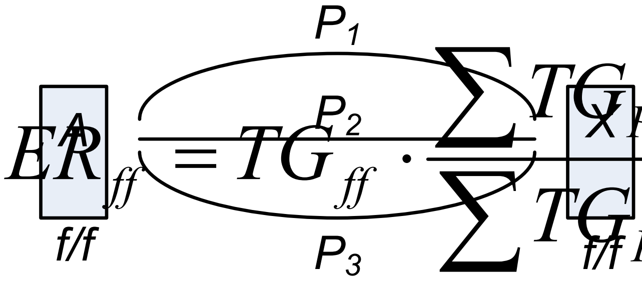
- **Main idea:** Downsize cells on rarely-exercised paths in order of decreasing toggle rate
- **Procedure**
  1. Pick a cell ***c*** with minimum toggle rate
  2. Downsize cell ***c*** with logically equivalent cell
  3. Incremental timing analysis and check error rate
  4. If error rate is increased, cell change is restored
  5. Repeat 1. ~ 4.
- *PowerReduction procedure reduces power without affecting error rate*

Heuristic Implementation – Error Rate Estimation



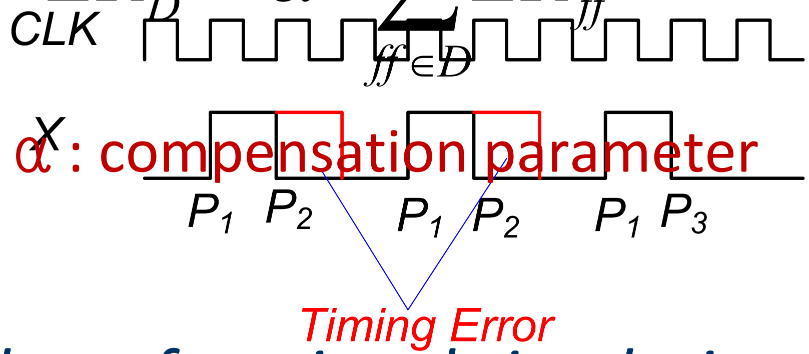
- Error rate estimation: use toggle rate from SAIF(Switching Activity Interchange Format)
- Error rate contribution of one flip-flop
- Error rate of an entire design

$$ER_{ff} = TG_{ff} \cdot \frac{\sum TG_{P_{NEG}}}{\sum TG_{P_{ALL}}}$$



$$ER_D = \alpha \cdot \sum_{ff \in D} ER_{ff}$$

α : compensation parameter



- We estimate error rates without functional simulation

$TG(P_1) = 0.3$	$Slack(P_1)$ = postive
$TG(P_2) = 0.2$	$Slack(P_2)$ = negative
$TG(P_3) = 0.1$	$Slack(P_3)$ = positive
$TG(X) = 0.6$	

$$ER(X) = TG(X) \cdot \frac{TG(P_2)}{TG(P_1) + TG(P_2) + TG(P_3)} = 0.2$$

Power Reduction Through Slack Redistribution

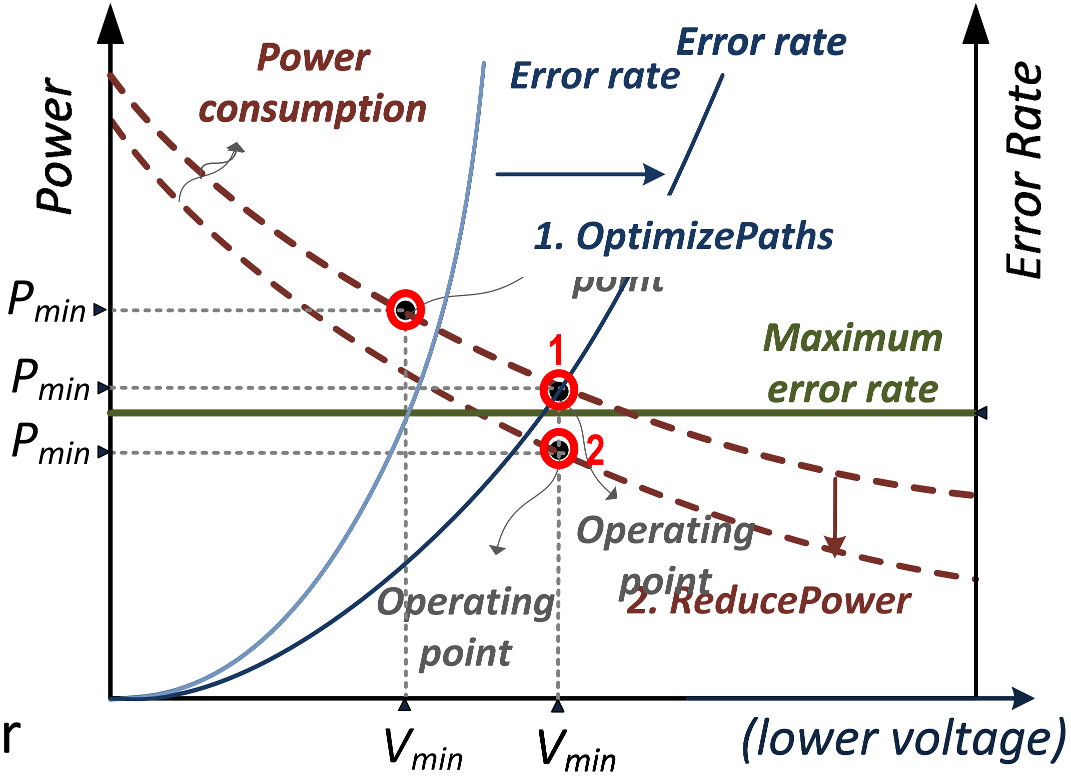
- Power consumption @BTWC
  - Minimum power P_{min} is obtained at minimum operating voltage V_{min}

1. OptimizePaths

- Minimize error rate
- Enable to scale voltage further

2. ReducePower

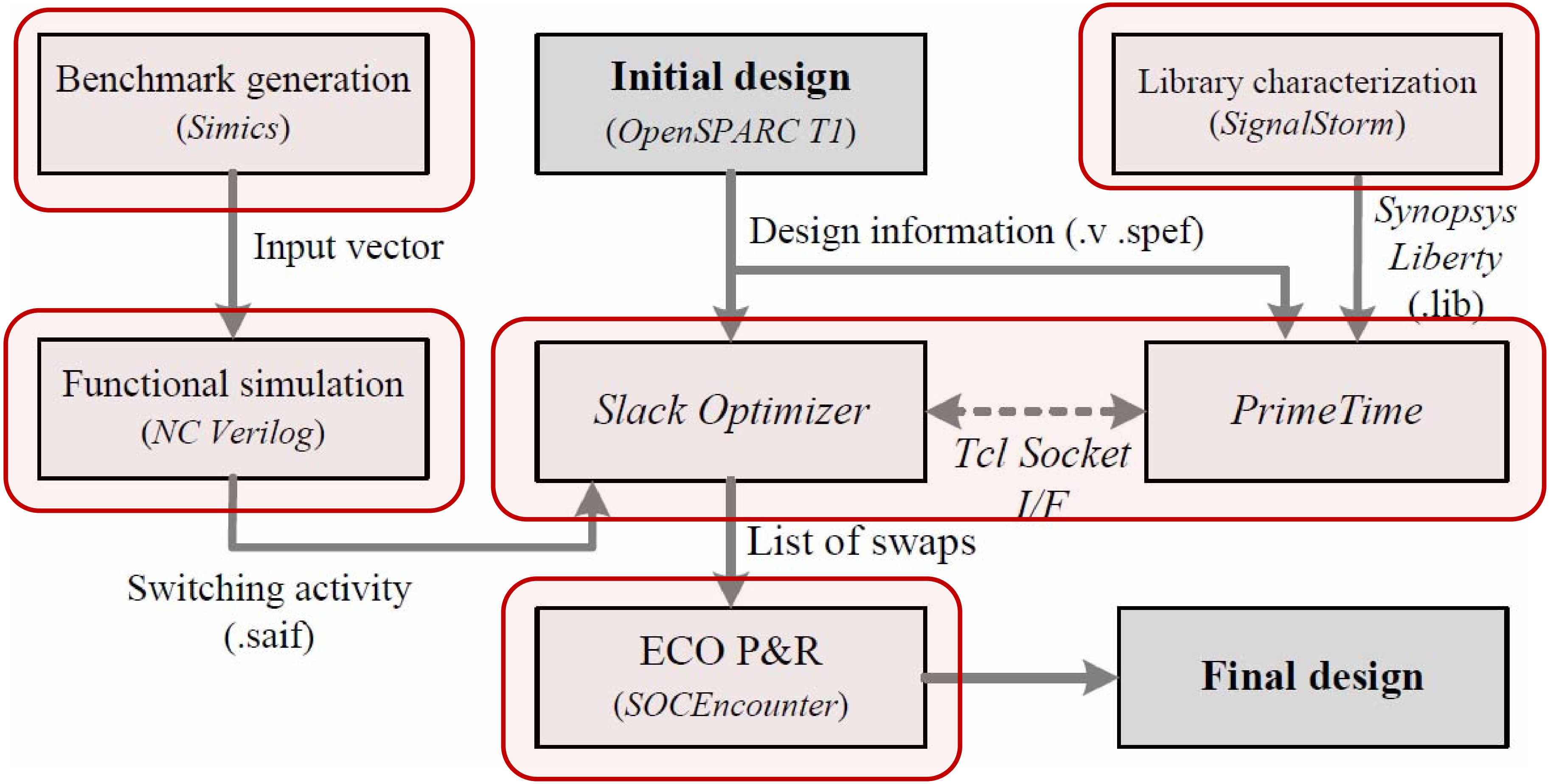
- Downsize cells
- Obtain additional power reduction



Outline

- Background and motivation
  - Voltage scaling and BTWC designs
  - Limitation of Traditional CAD Flow
- Power-Aware Slack Redistribution
  - Our design optimization goal
  - Related work: *BlueShift*
  - Our Heuristic
- **Experimental Framework and Results**
  - **Design methodology**
  - **Testbed**
  - **Results and analysis**
- Conclusions and Ongoing Work

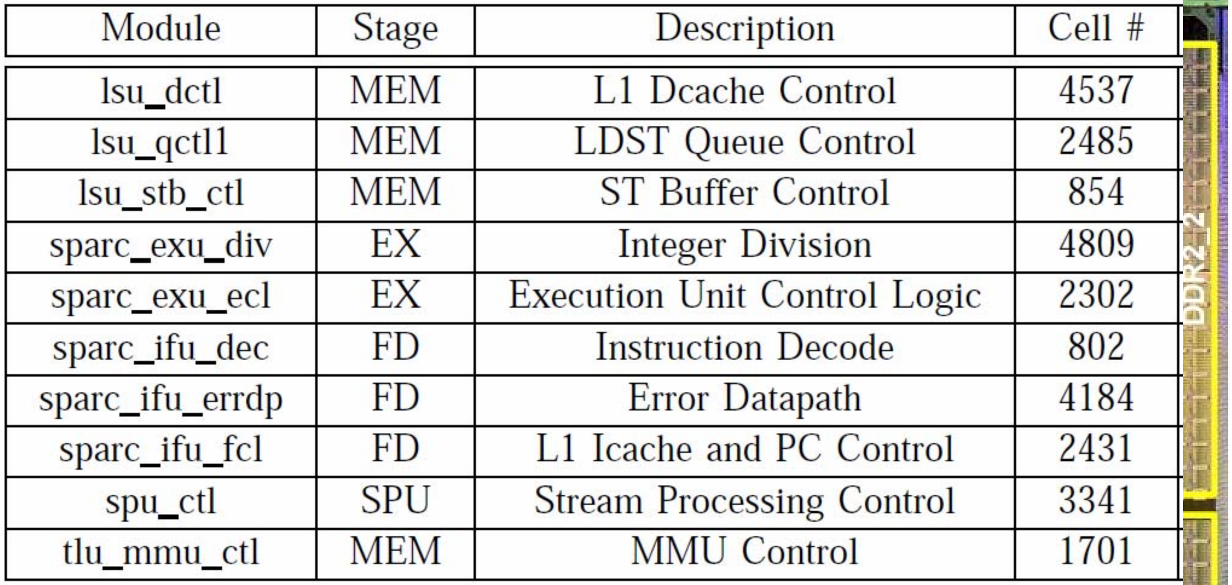
Design Methodology



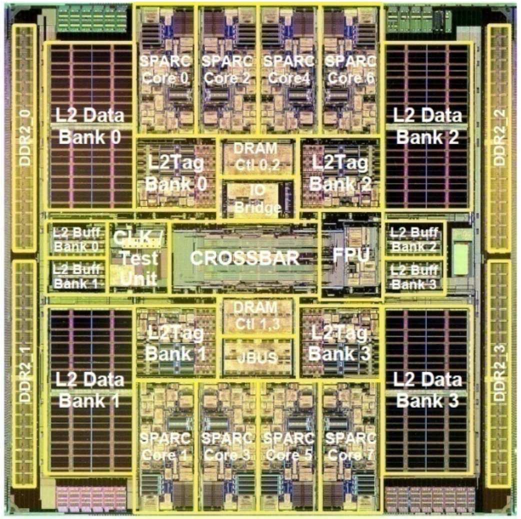
- [illegible]ation)
  - [illegible] *Synopsys PrimeTime*

Testbed

- Target design : sub-modules of *OpenSPARC T1*



Module	Stage	Description	Cell #
lsu_dctl	MEM	L1 Dcache Control	4537
lsu_qctl1	MEM	LDST Queue Control	2485
lsu_stb_ctl	MEM	ST Buffer Control	854
sparc_exu_div	EX	Integer Division	4809
sparc_exu_ecl	EX	Execution Unit Control Logic	2302
sparc_ifu_dec	FD	Instruction Decode	802
sparc_ifu_errdp	FD	Error Datapath	4184
sparc_ifu_fcl	FD	L1 Icache and PC Control	2431
spu_ctl	SPU	Stream Processing Control	3341
tlu_mmu_ctl	MEM	MMU Control	1701



- Benchmark
  - Ammp, bzip2, equake, sort and twolf
  - Make test vectors with 1 billion cycles for each sub-module
- Implementation
  - TSMC 65GP technology with standard SP&R flow

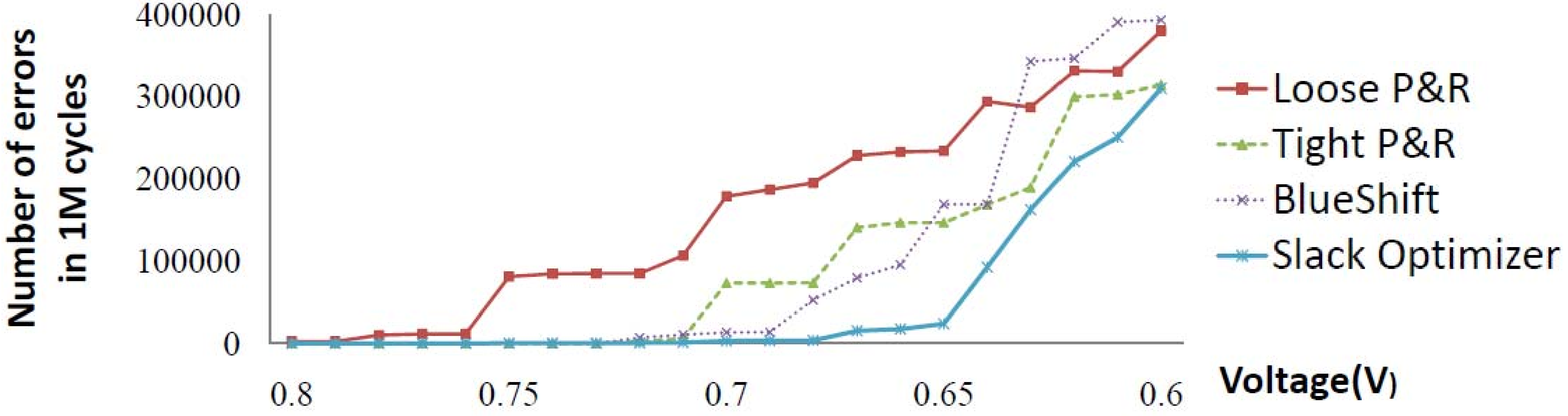
List of Experiments

- Design techniques
  1. SP&R with 0.8 GHz (loose constraints)
  2. SP&R with 1.2 GHz (tight constraints)
  3. *Blueshift*: timing override
  4. Slack Optimizer

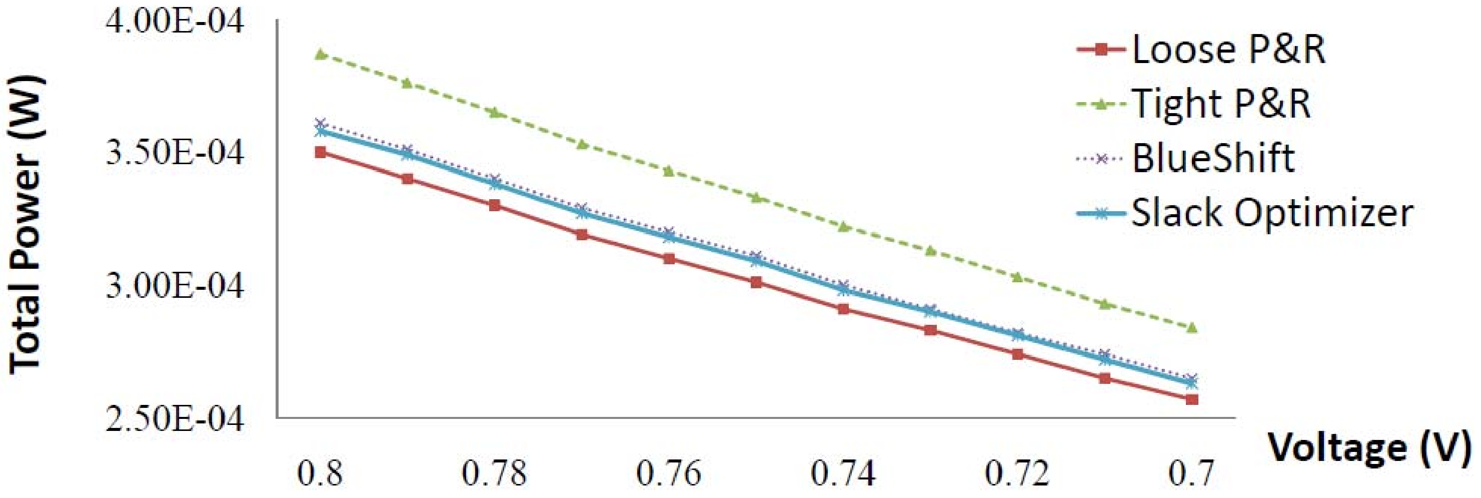
- Experiments compare all design techniques with respect to:
  1. Power consumption at each voltage point
  2. Actual error rates from gate level simulation
  3. Power consumption at each target error rate
  4. Estimated processor-wide power consumption

Error Rate and Power Results

- Error rate at each operating voltage (test case : *lsu_dctl*)

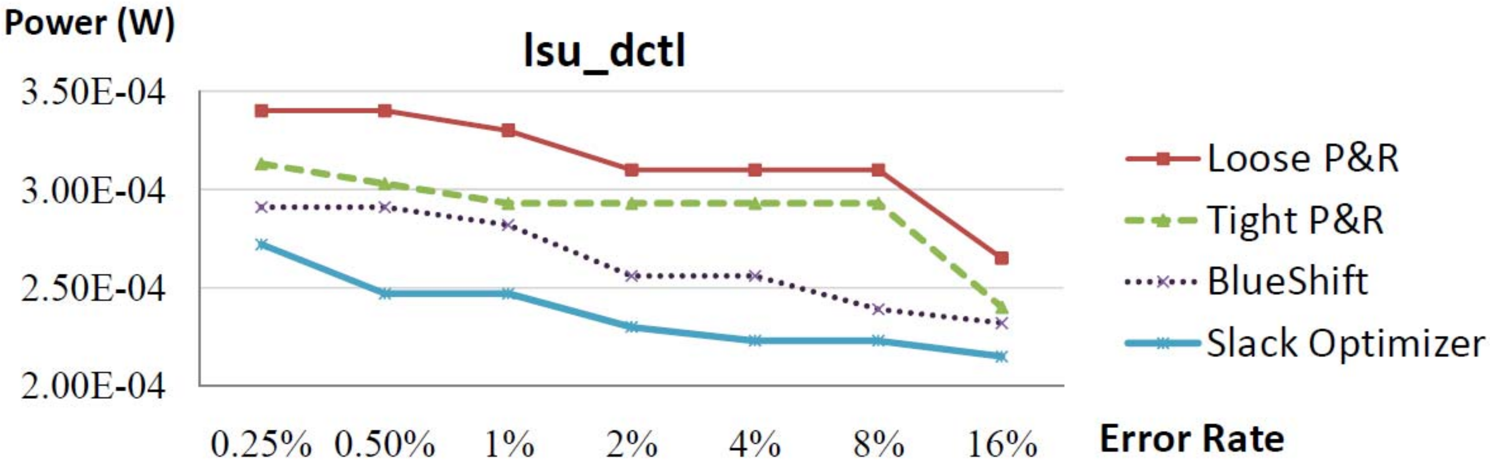


- Power consumption at each operating voltage

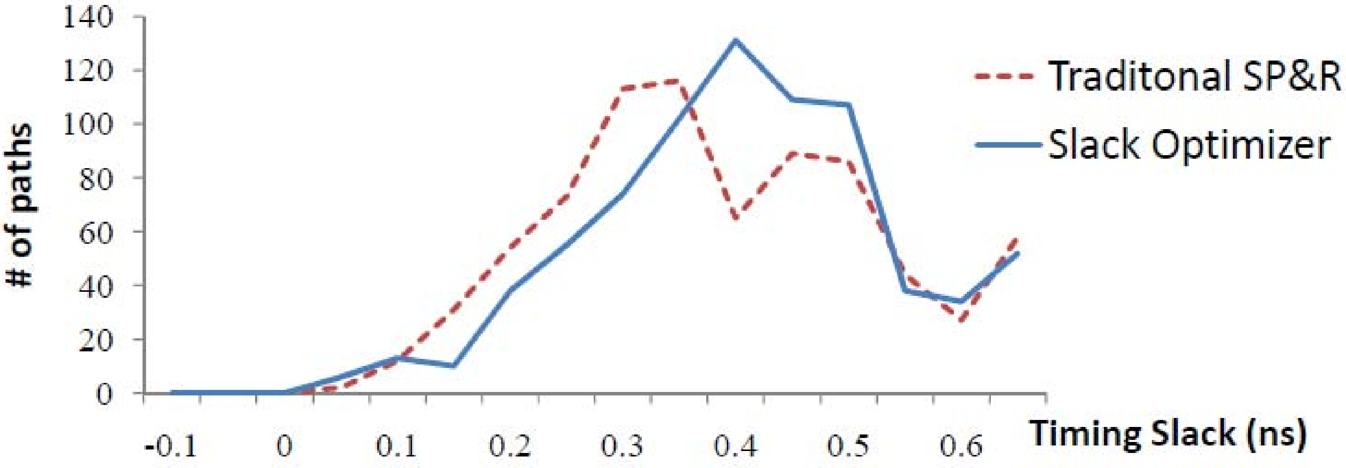


Comparison of Power and Slack Results

- Power consumption at each target error rate

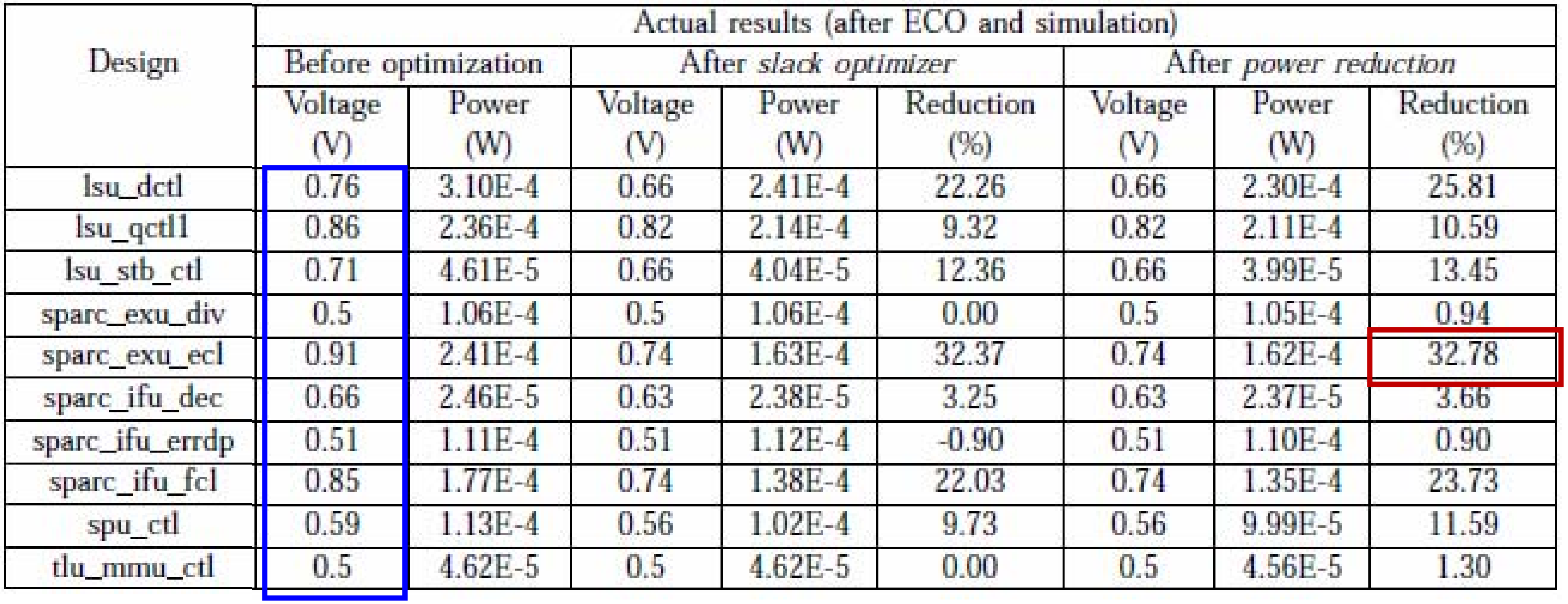


- Slack distribution



Power Reduction and Area Overhead

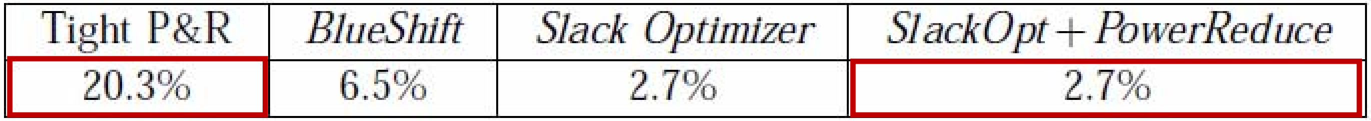
- Power reduction after optimization (@ 2% error rate)



Design	Actual results (after ECO and simulation)							
	Before optimization		After *slack optimizer*			After *power reduction*		
	Voltage (V)	Power (W)	Voltage (V)	Power (W)	Reduction (%)	Voltage (V)	Power (W)	Reduction (%)
lsu_dctl	0.76	3.10E-4	0.66	2.41E-4	22.26	0.66	2.30E-4	25.81
lsu_qctl1	0.86	2.36E-4	0.82	2.14E-4	9.32	0.82	2.11E-4	10.59
lsu_stb_ctl	0.71	4.61E-5	0.66	4.04E-5	12.36	0.66	3.99E-5	13.45
sparc_exu_div	0.5	1.06E-4	0.5	1.06E-4	0.00	0.5	1.05E-4	0.94
sparc_exu_ecl	0.91	2.41E-4	0.74	1.63E-4	32.37	0.74	1.62E-4	32.78
sparc_ifu_dec	0.66	2.46E-5	0.63	2.38E-5	3.25	0.63	2.37E-5	3.66
sparc_ifu_errdp	0.51	1.11E-4	0.51	1.12E-4	-0.90	0.51	1.10E-4	0.90
sparc_ifu_fcl	0.85	1.77E-4	0.74	1.38E-4	22.03	0.74	1.35E-4	23.73
spu_ctl	0.59	1.13E-4	0.56	1.02E-4	9.73	0.56	9.99E-5	11.59
tlu_mmu_ctl	0.5	4.62E-5	0.5	4.62E-5	0.00	0.5	4.56E-5	1.30

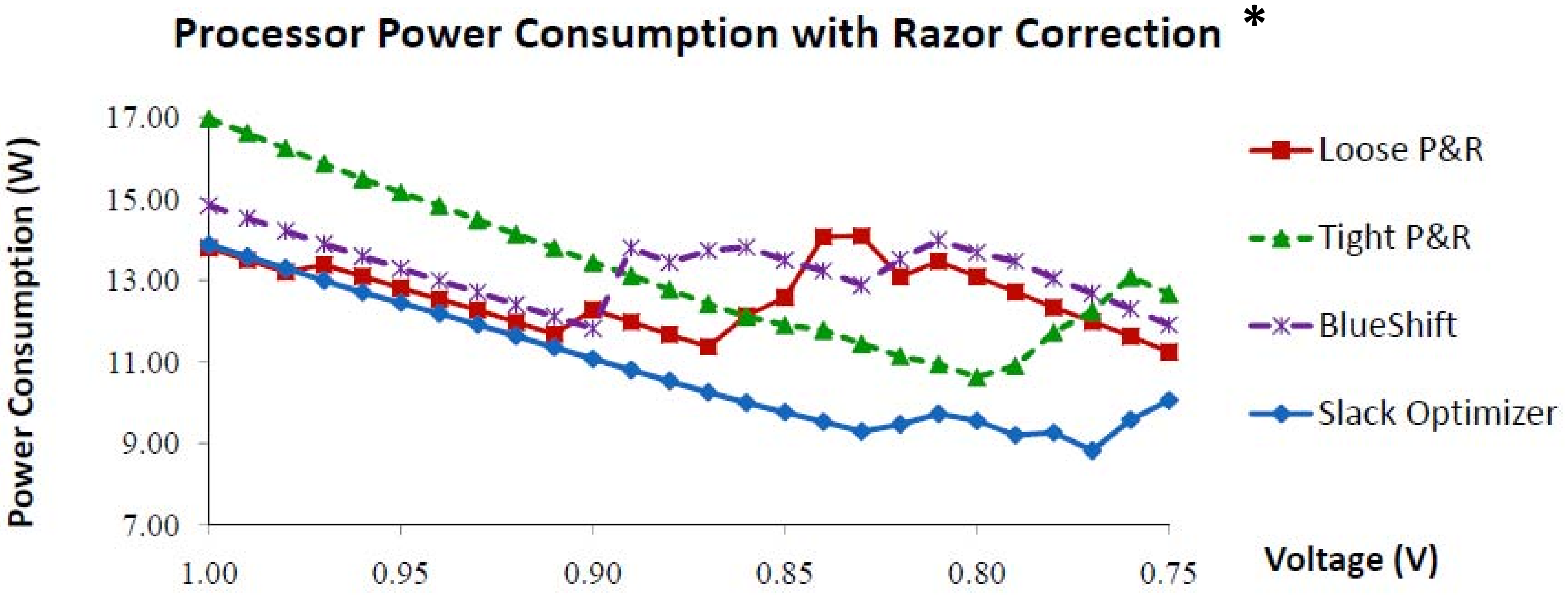
Max. 32.8 %, Avg. 12.5% power reduction

- Area overhead of design approaches



Tight P&R	*BlueShift*	*Slack Optimizer*	*SlackOpt + PowerReduce*
20.3%	6.5%	2.7%	2.7%

Processor-wide Results



*Kahng et al. "Designing a Processor From the Ground Up to Allow Voltage/Reliability Tradeoffs", *HPCA* 2010.

- *Slack optimization extends range of voltage scaling and reduces Razor recovery cost*

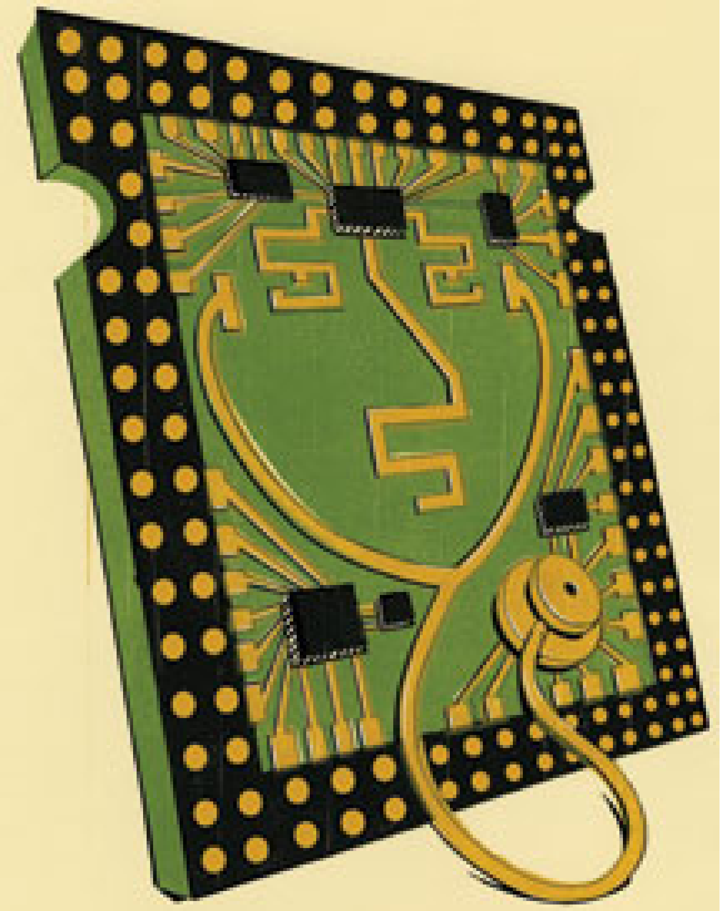
Conclusions and Ongoing Work

- Showed limitations of a BTWC design
- Presented design technique – slack redistribution
  - Optimize frequently exercised critical paths
  - De-optimize rarely-exercised paths
- Demonstrated significant power benefits of gradual slack design
  - Reduced power 33% on maximum , 12.5% on average
- Ongoing work
  - Reliability-power tradeoffs for embedded memory
  - Applying to heterogeneous multi-core architecture

THANK YOU

BACKUP

CPU, Heal Thyself

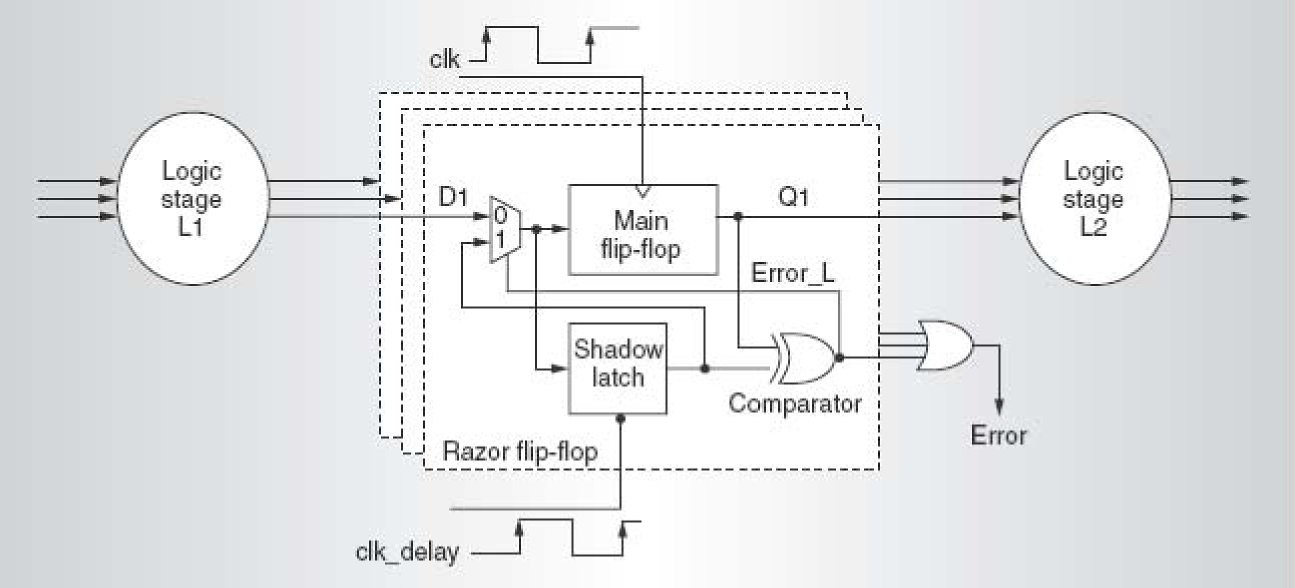


- *Razor** system
  - Timing errors can be corrected
  - Manage the trade-off between system voltage and error rate
- New design methodology is needed

* Razor: A low power pipeline based on circuit-level timing speculation. *In International Symposium on Micro architecture*, December 2003.

Razor – How it works

- Razor Implementation



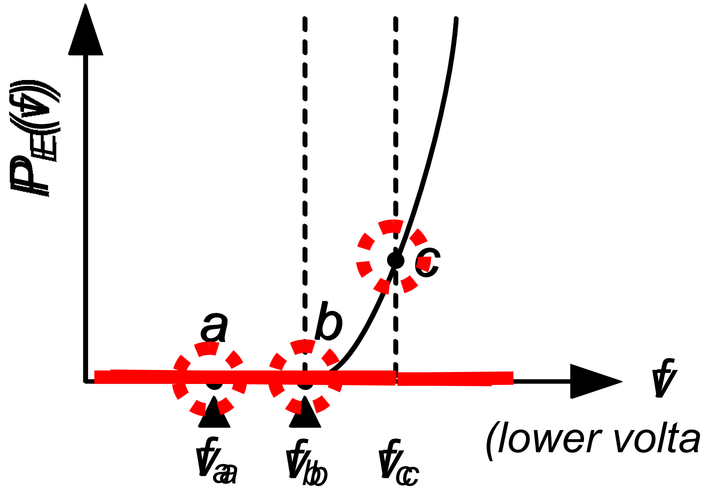
  - Razor: A low power pipeline based on circuit-level timing speculation. In International Symposium on Microarchitecture, December 2003.
  - Main flip-flop latches at T, but Shadow latch latches at T+skew
  - If a timing violation occurs, main flip-flop will latch incorrect value, but shadow latch should latch correct value
  - Comparator signals error and the late arriving value is fed back into the main flip-flop

BTWC: Voltage Scaling

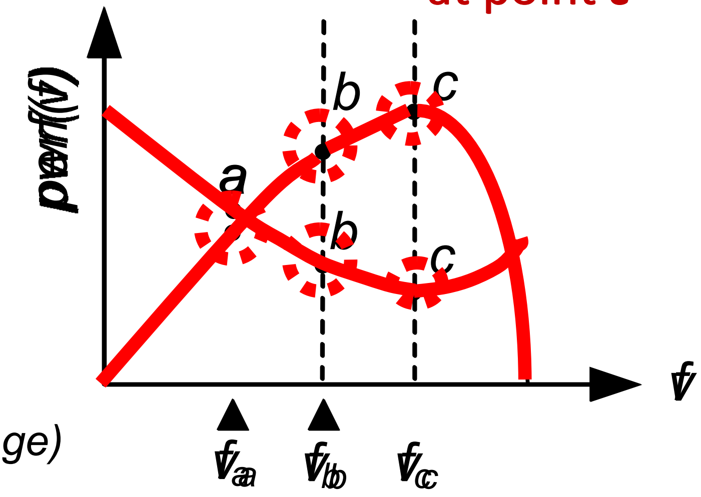
- Overclocking case
- Voltage scaling

Maximum performance at point *c*

Minimum power at point *c*



(lower voltage)



$P_E(f)$: Error rate at frequency ***f***

$P_E(v)$: Error rate at voltage ***v***

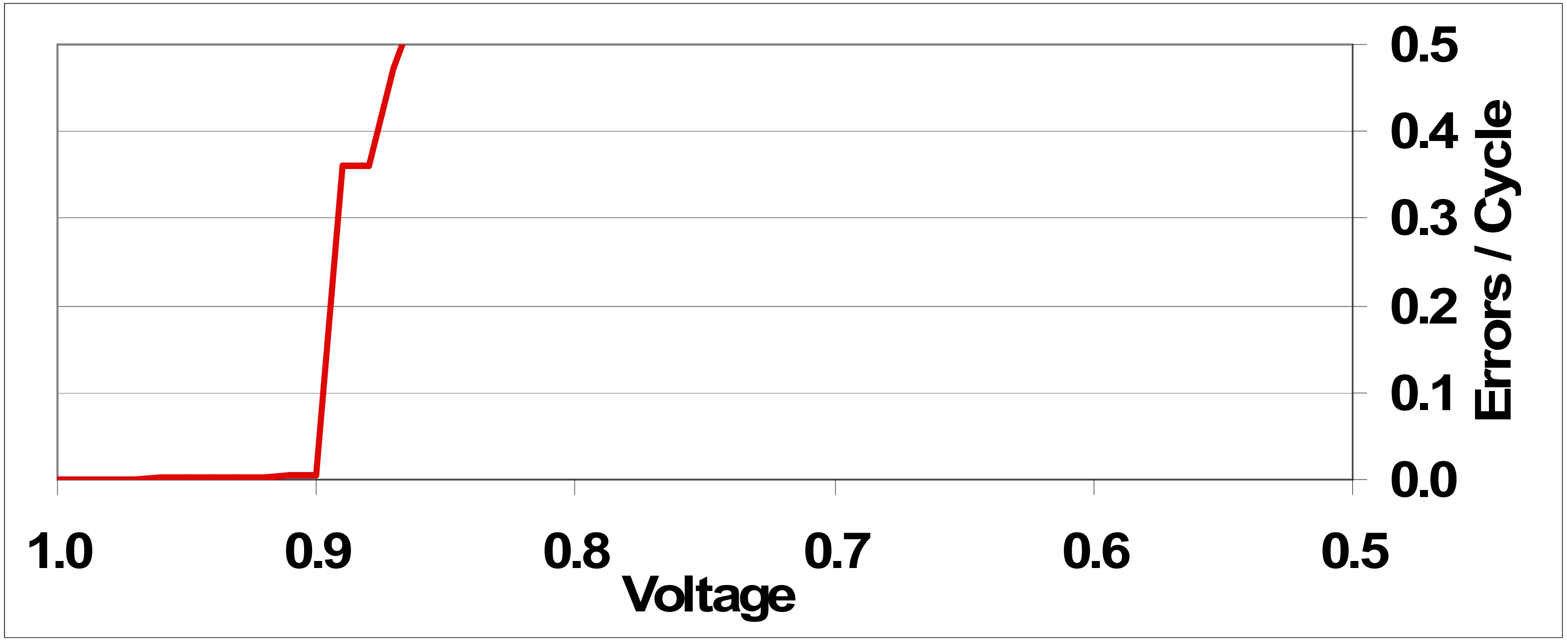
perf(f) **:** Performance at ***f***

pwr(v) **:** Power consumption at ***v***

- Error correction needs additional clock cycles and incurs power overhead

Limitation of Voltage Scaling

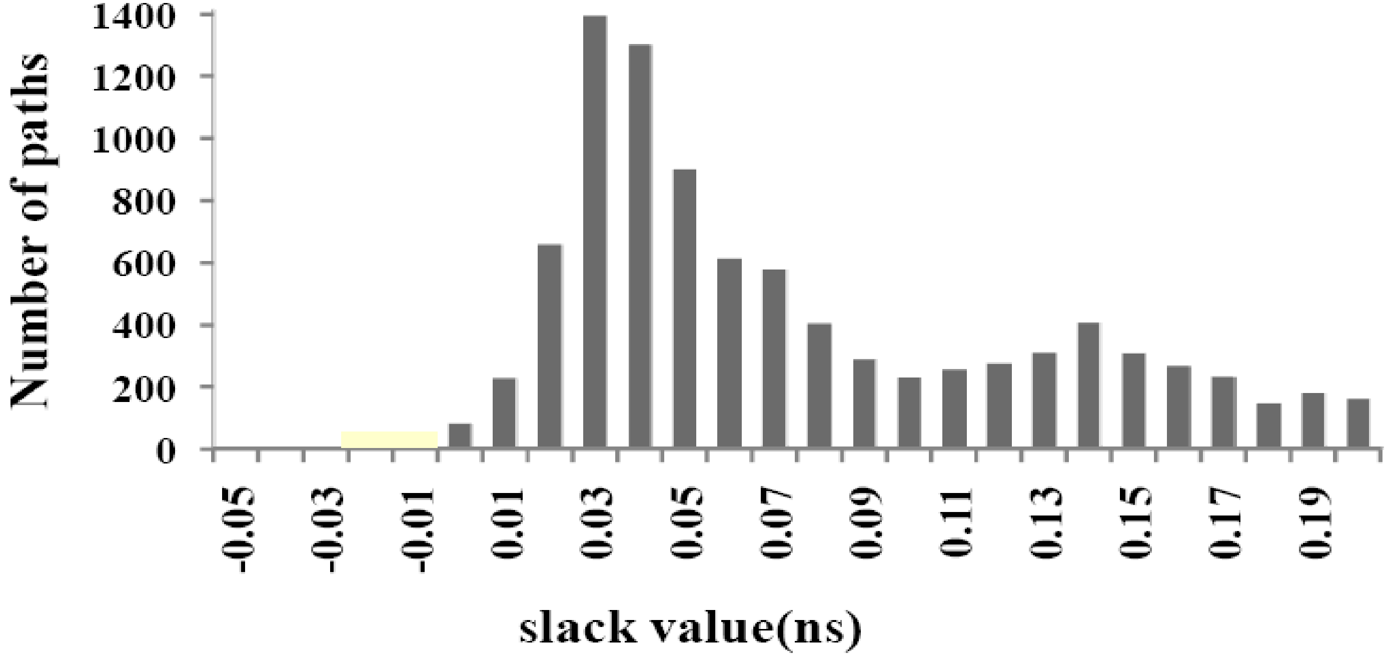
- At some voltage, circuit breaks down



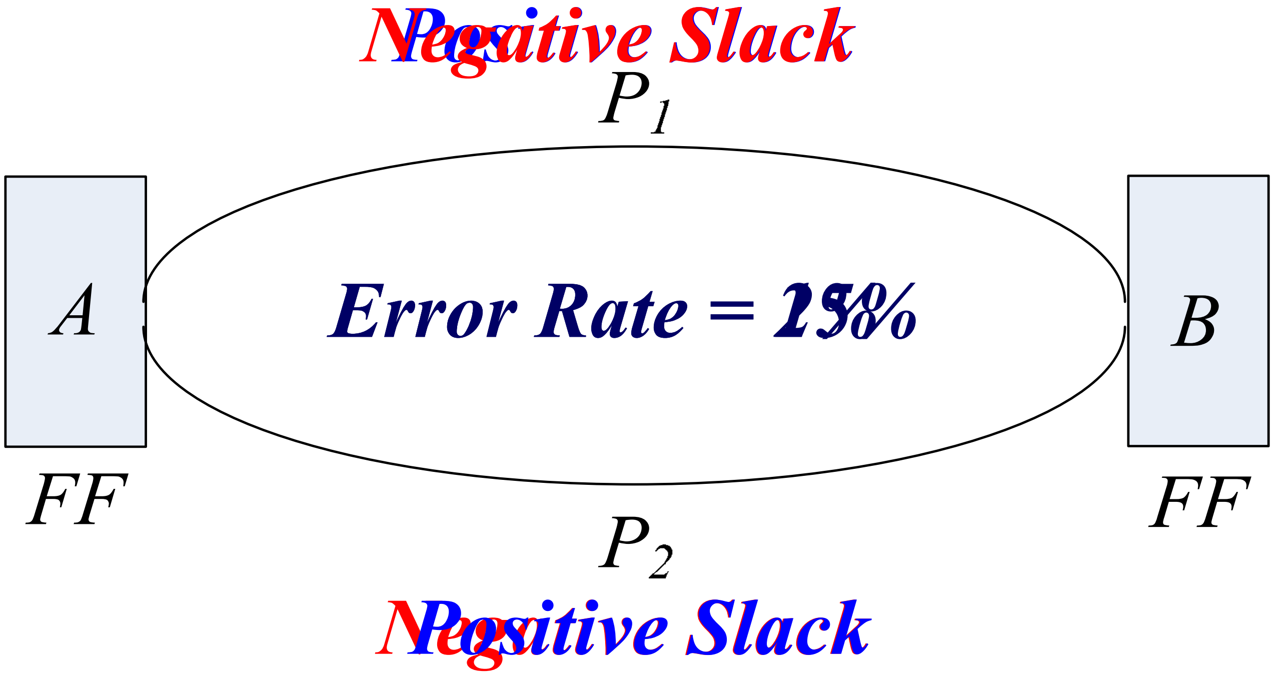
Voltage scaling must halt after only 10% scaling.

Reason for Steep Error Degradation

- Critical paths are bunched up in traditional designs.



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**

Heuristic Implementation – Error Rate Estimation

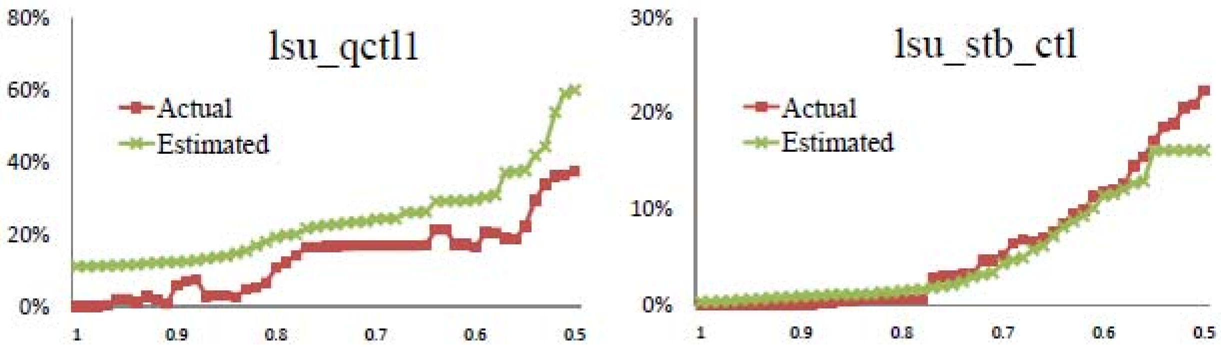
- Error rate contribution of one flip-flop

(1) $$ER_{ff} = TG_{ff} \cdot \frac{\sum TG_{P_{NEG}}}{\sum TG_{P_{ALL}}}$$

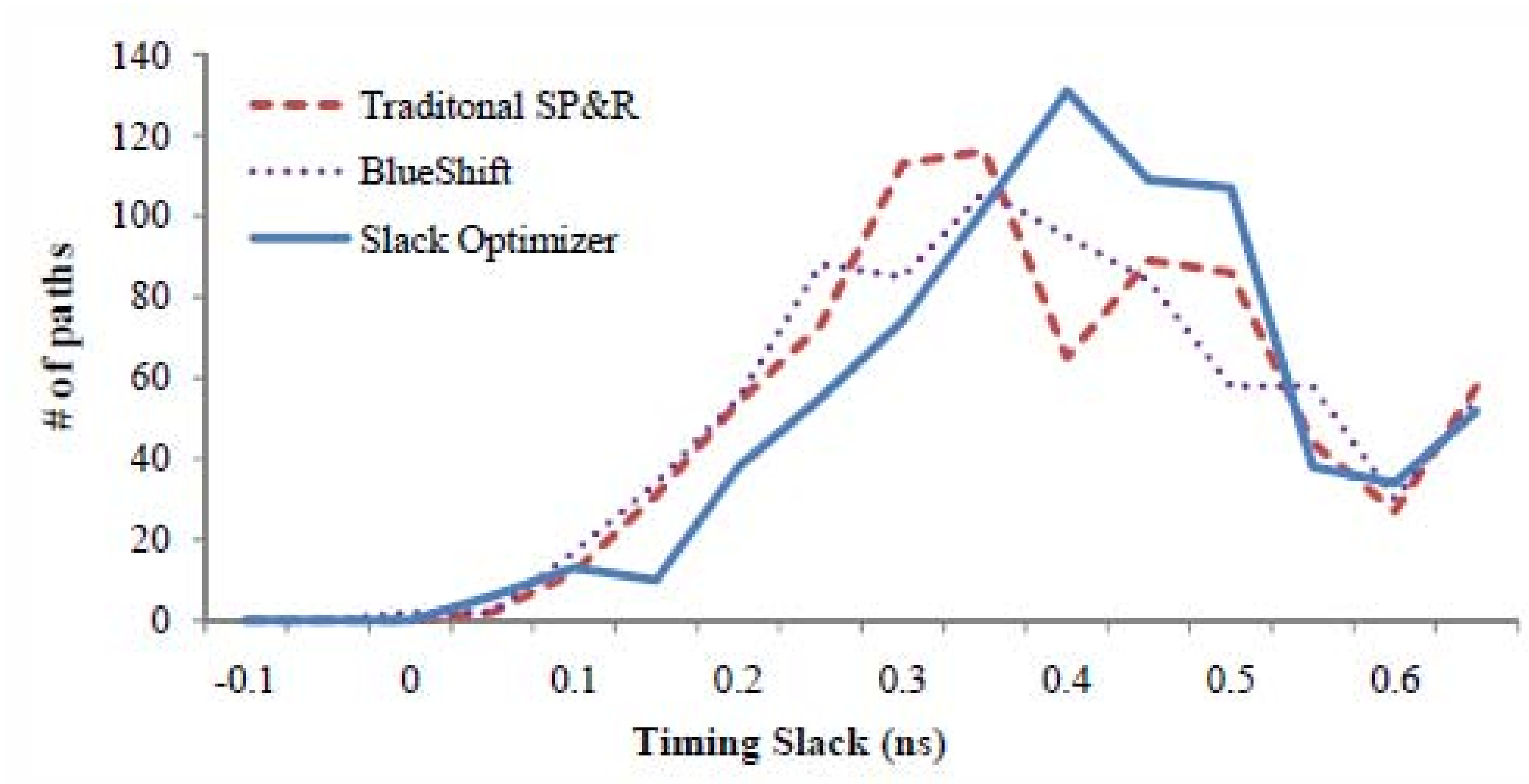
- Error rate of an entire design

(2) $$ER_D = \alpha \cdot \sum_{ff \in D} ER_{ff}$$ α : compensation parameter

- Actual vs. estimated error rates

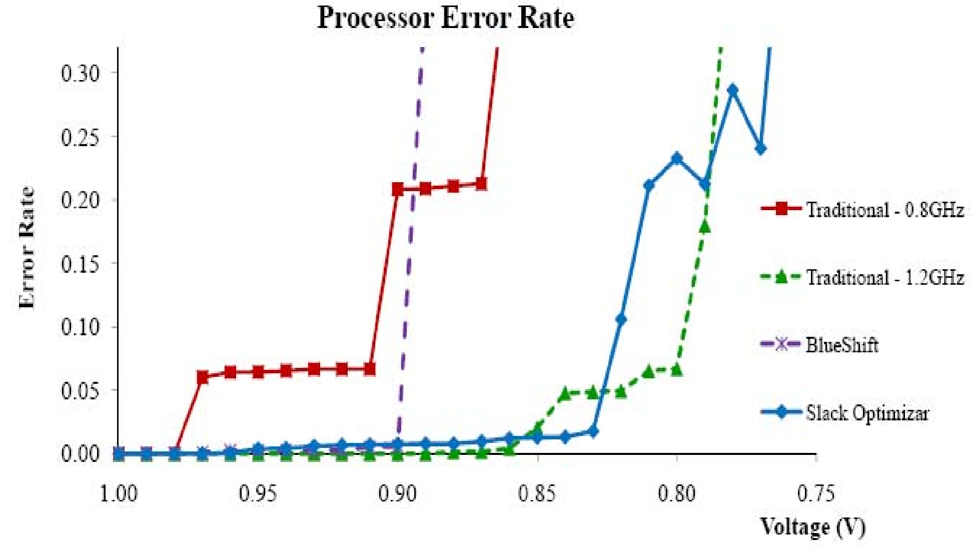


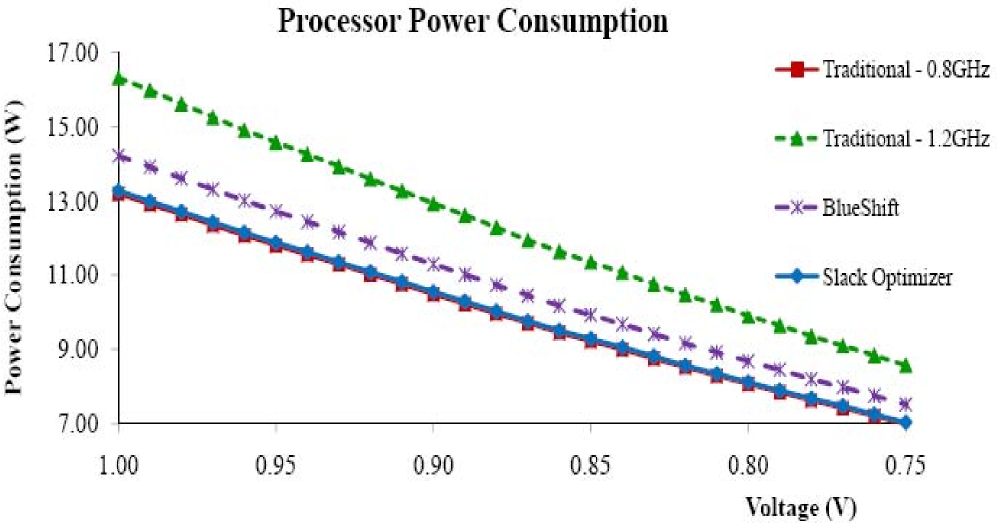
Gradual Slack Distribution



Slack optimization achieves gradual slack distribution.

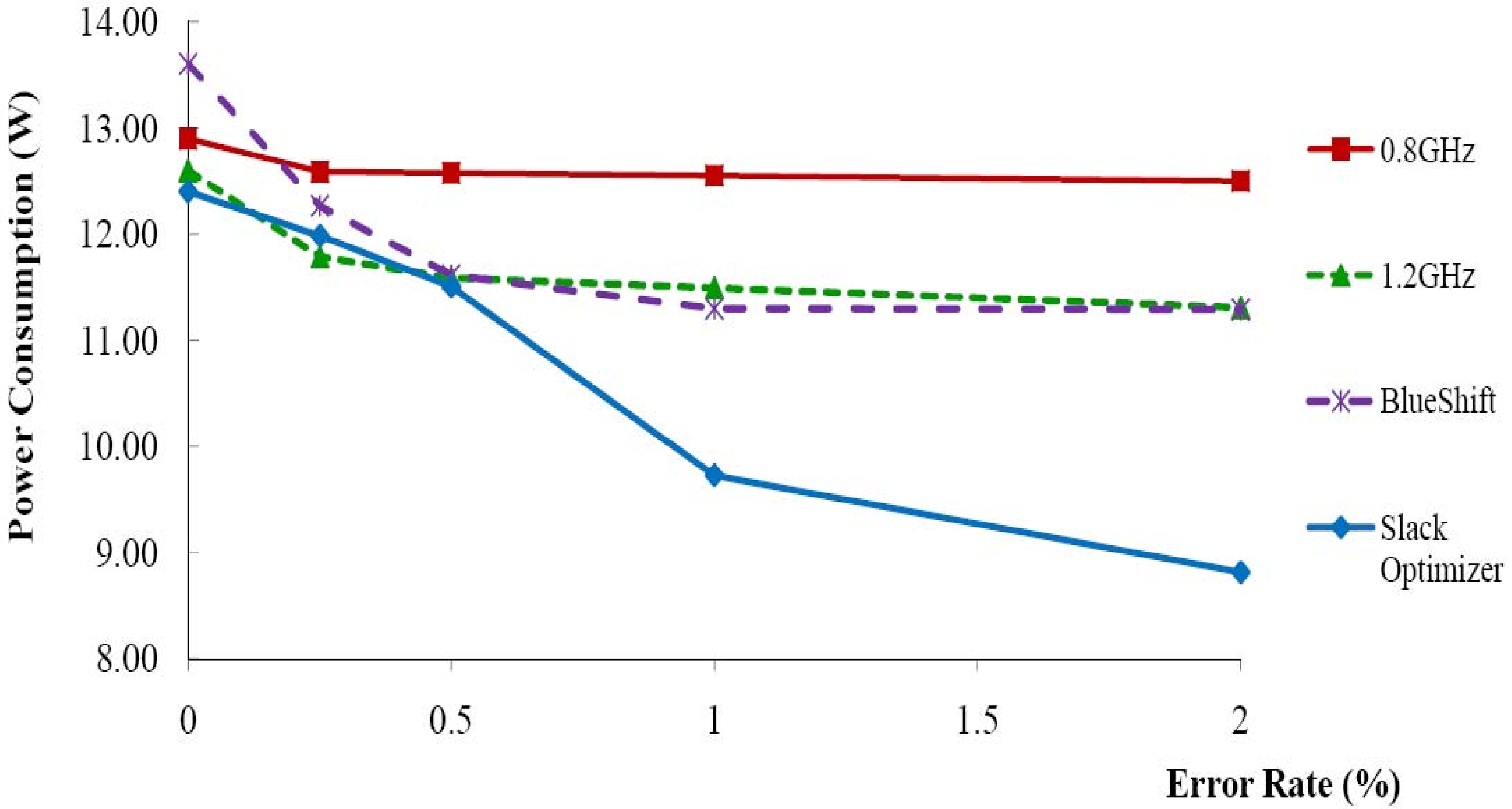
Processor Error Rate and Power





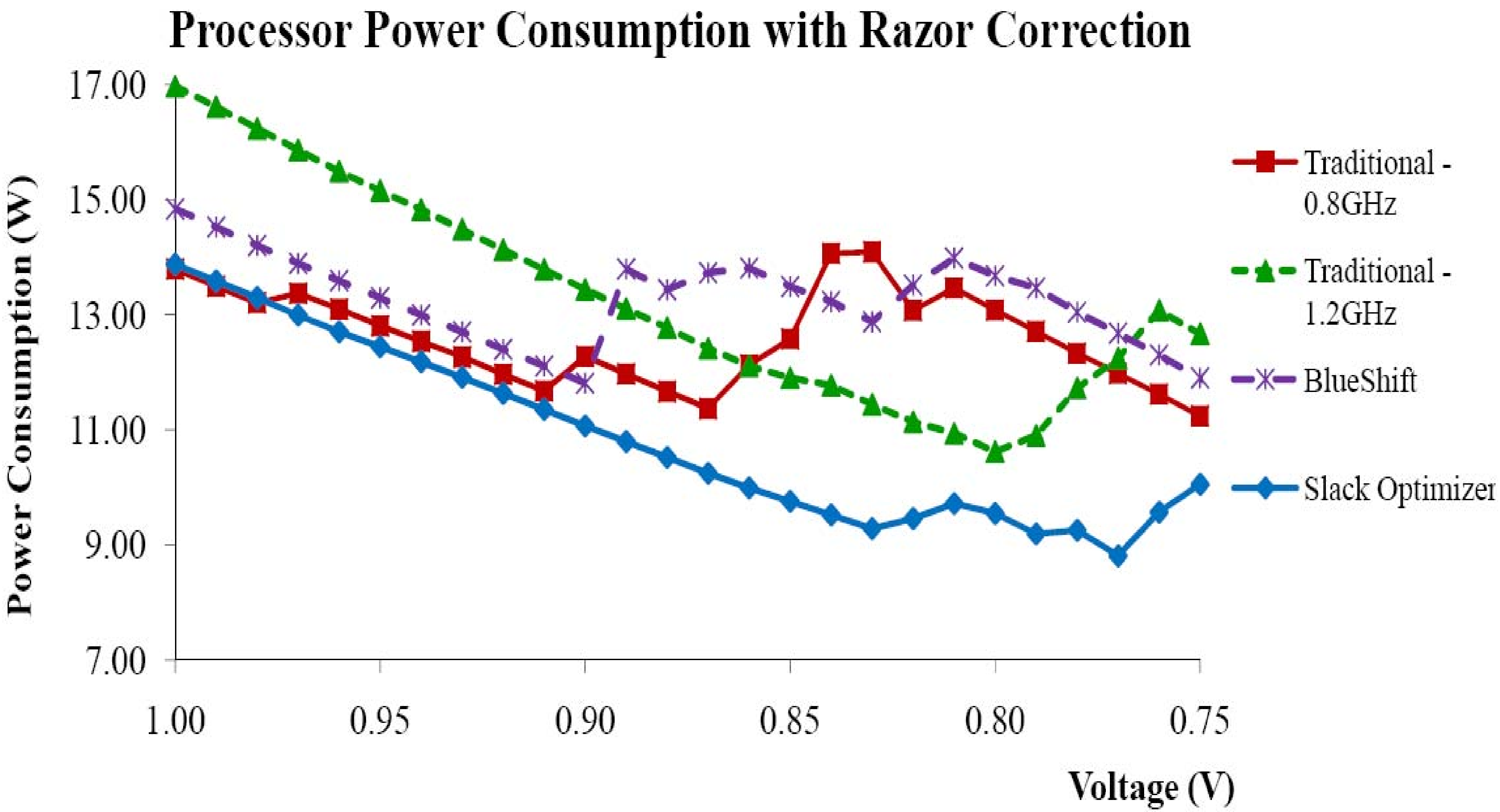
Designs with comparable error rates have much higher power/area overheads.

Reliability/Power Tradeoff



Slack-optimized design enjoys continued power reduction as error rate increases.

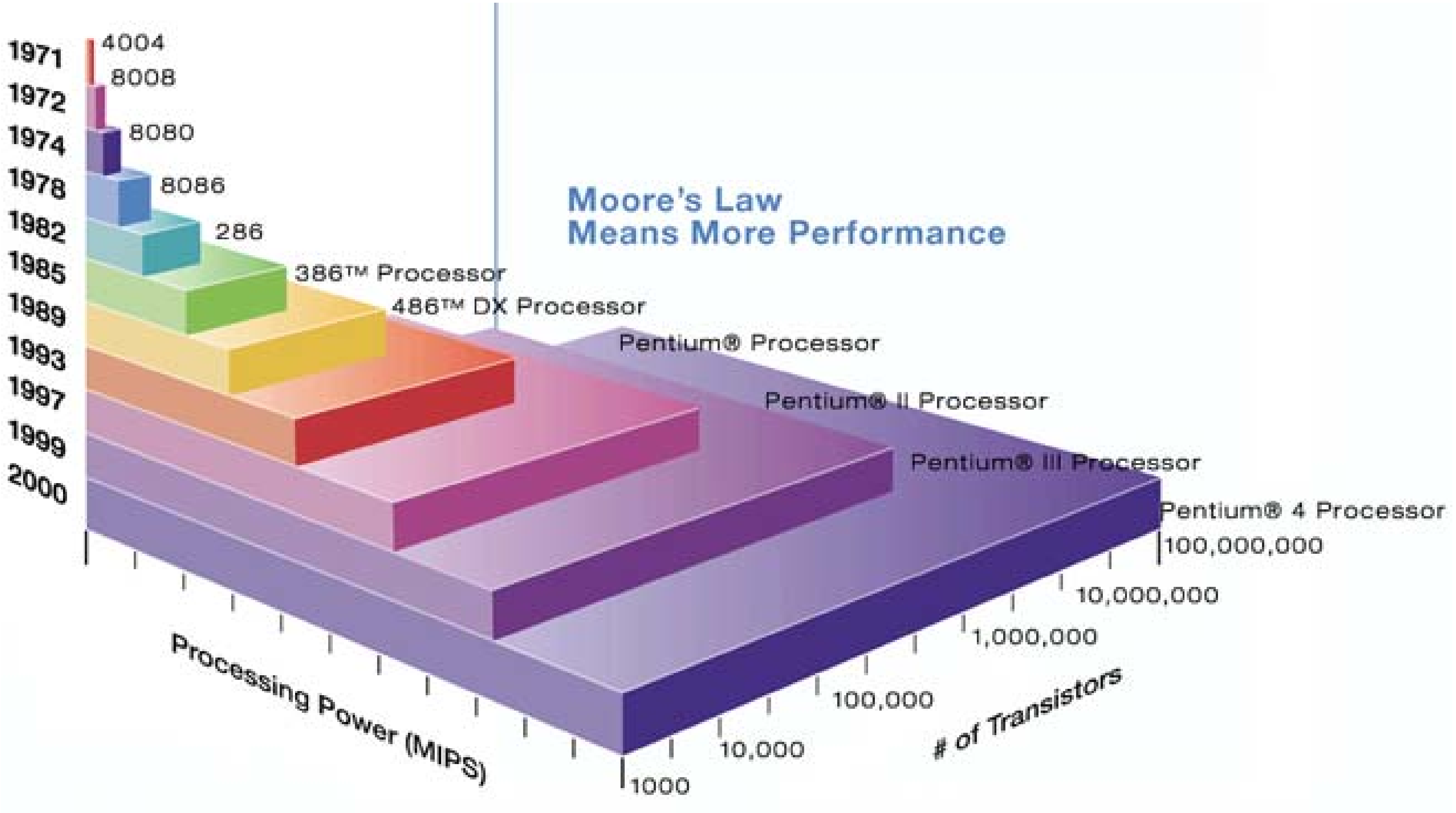
Enhancing Razor-based Design



Slack optimization extends range of voltage scaling and reduces Razor recovery cost.

Moore's Law

- Power consumption of processor node doubles every 18 months.



Power Scaling

- With current design techniques, processor power soon on par with nuclear power plant