Performance Optimization and Analysis of Blade Designs under Delay Variability

Dylan Hand[*], Hsin-Ho Huang[*], Benmao Chang[‡] ,
Yang Zhang[*], Matheus Trevisan Moreira[*†], Melvin Breuer[*],
Ney Laert Vilar Calazans[†], and Peter A. Beerel[*]

May 5th, 2015

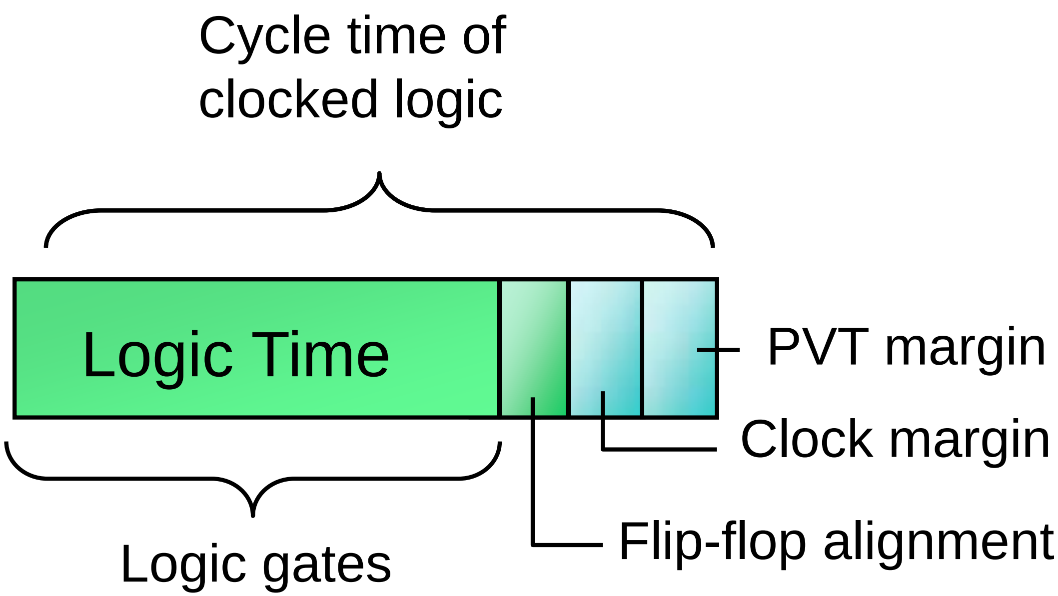
* University of Southern California, Los Angeles, CA
† Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, Brazil
‡ Dept. of Automation, Tsinghua University, Beijing, China

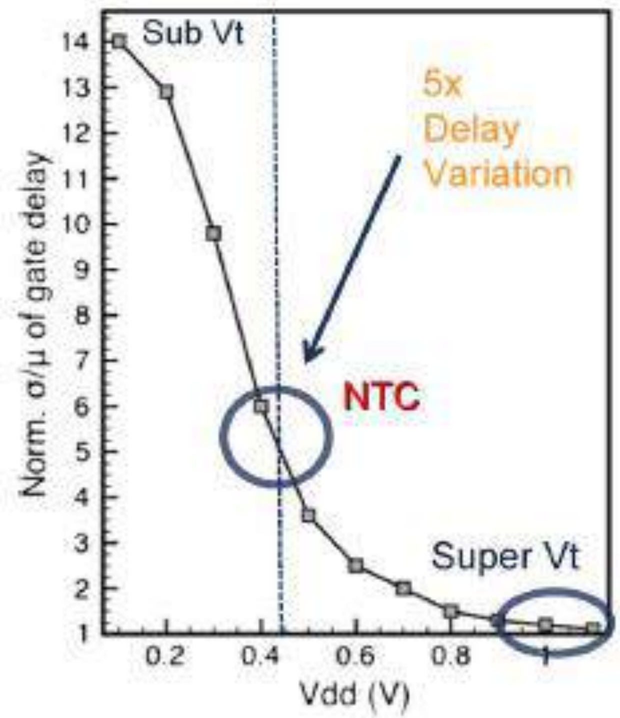
USC Viterbi School of Engineering

PUCRS

University of Southern California

The Context: Delay Overheads



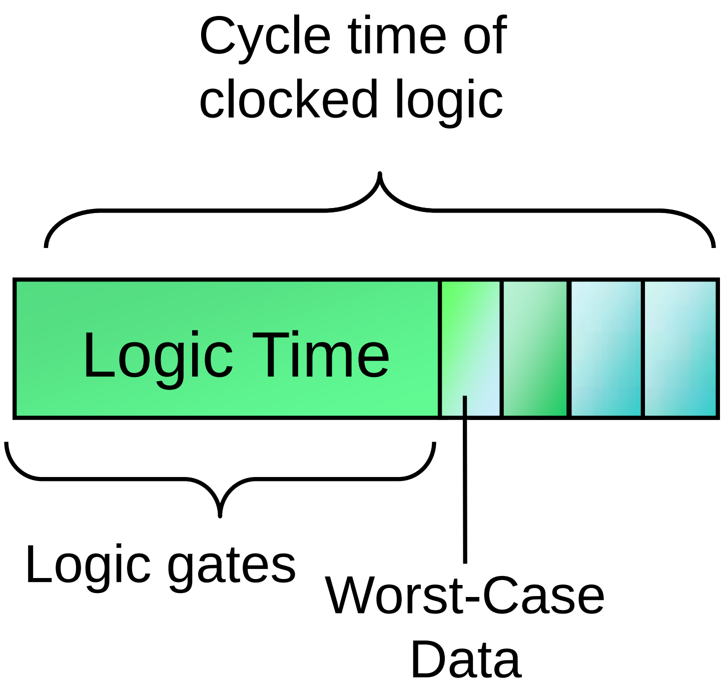


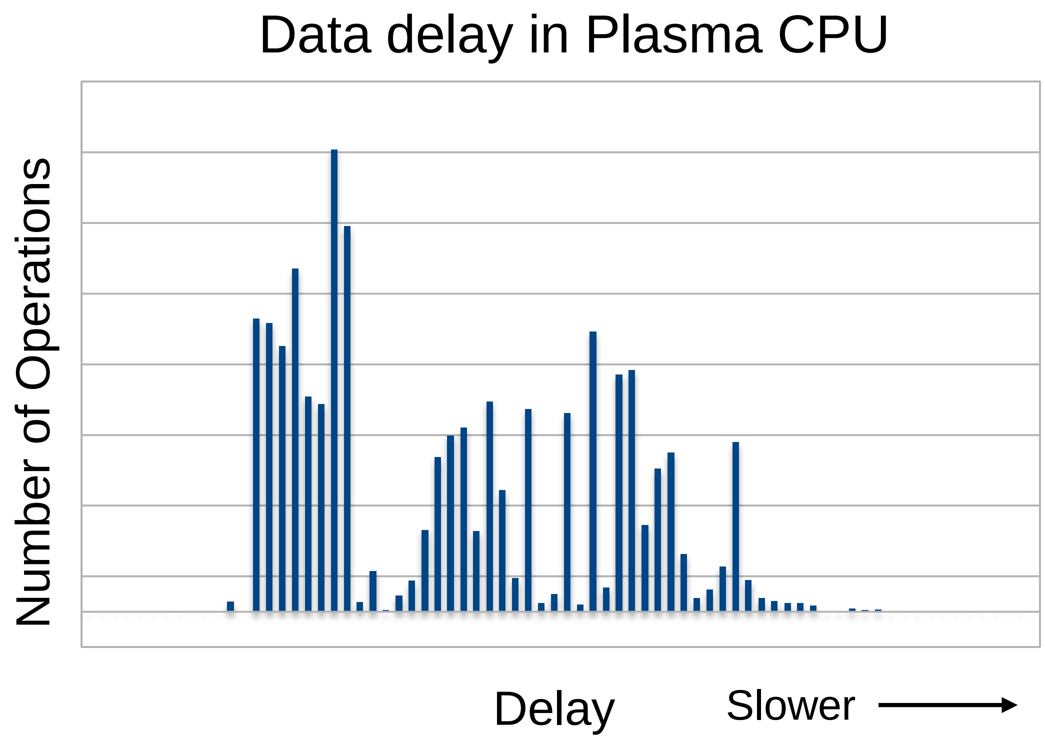
[Dreslinksi et al., IEEE Proc. 2010]

Traditional synchronous design suffers from increased margins

- Worse at low and near-threshold regions

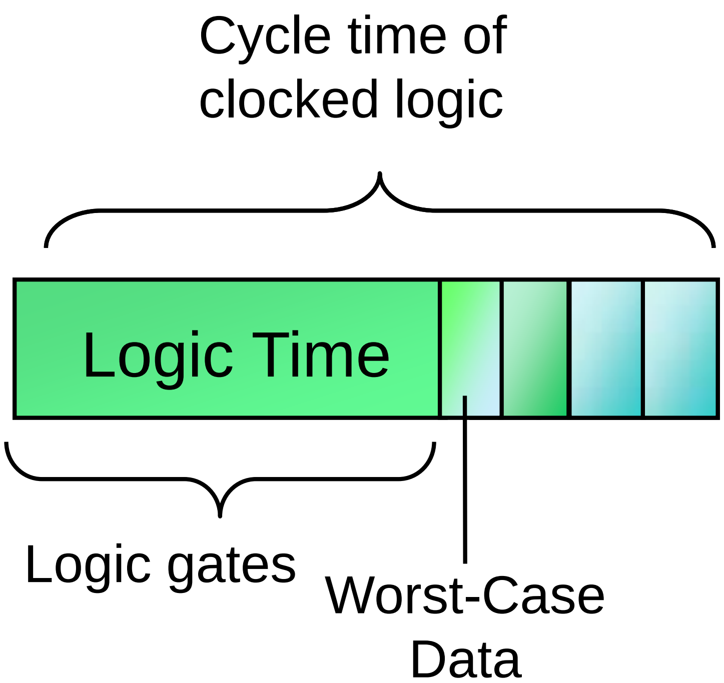
Potential of Average-Case Data

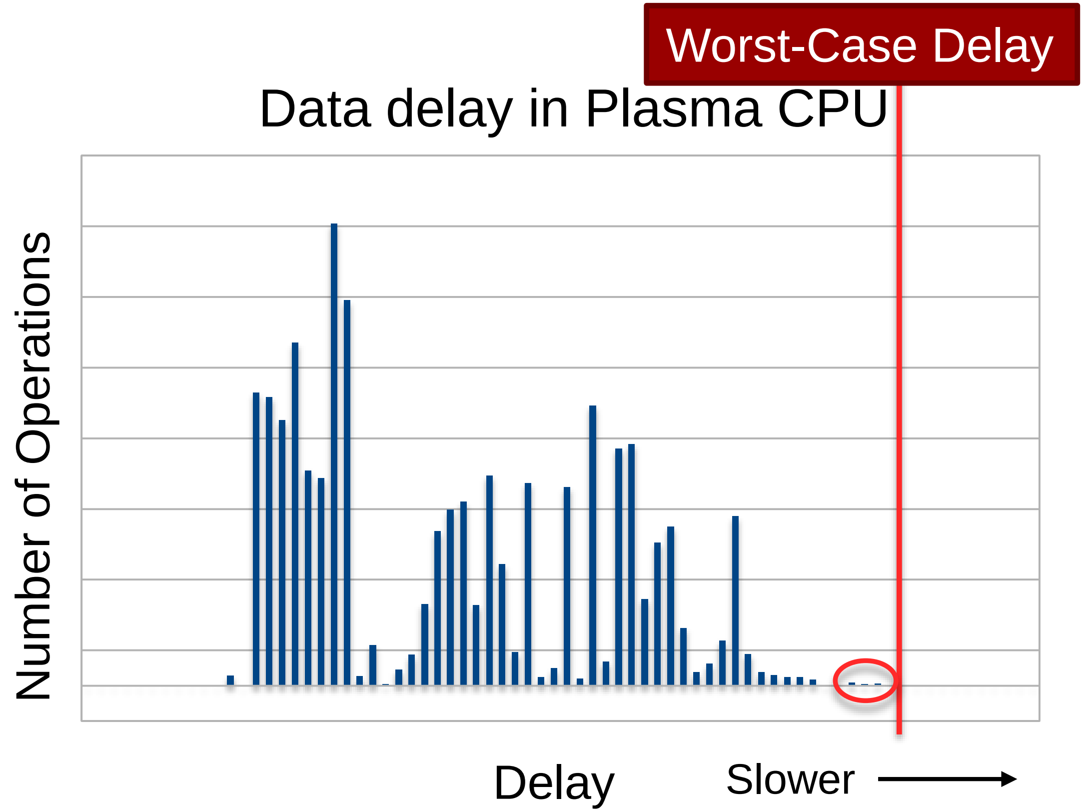




Delay variation due to data is rarely exploited in synchronous designs

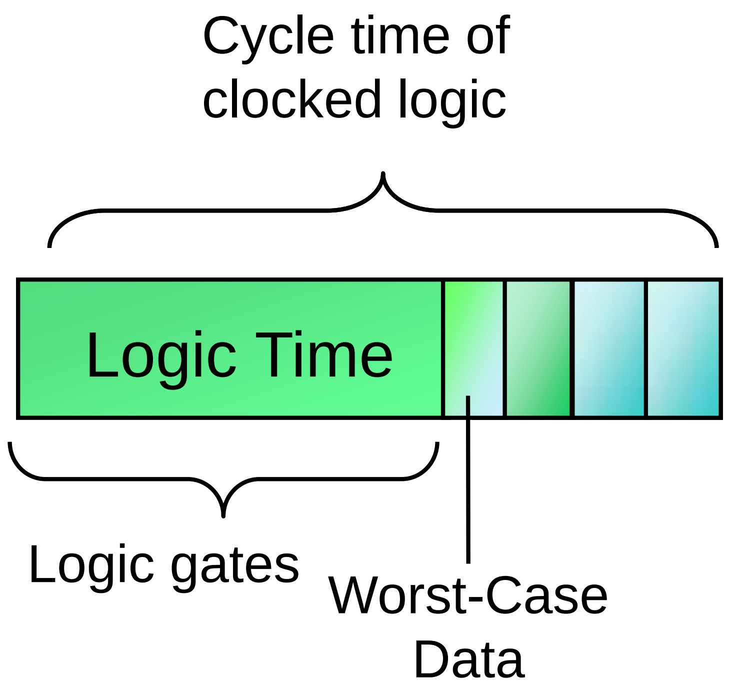
Potential of Average-Case Data

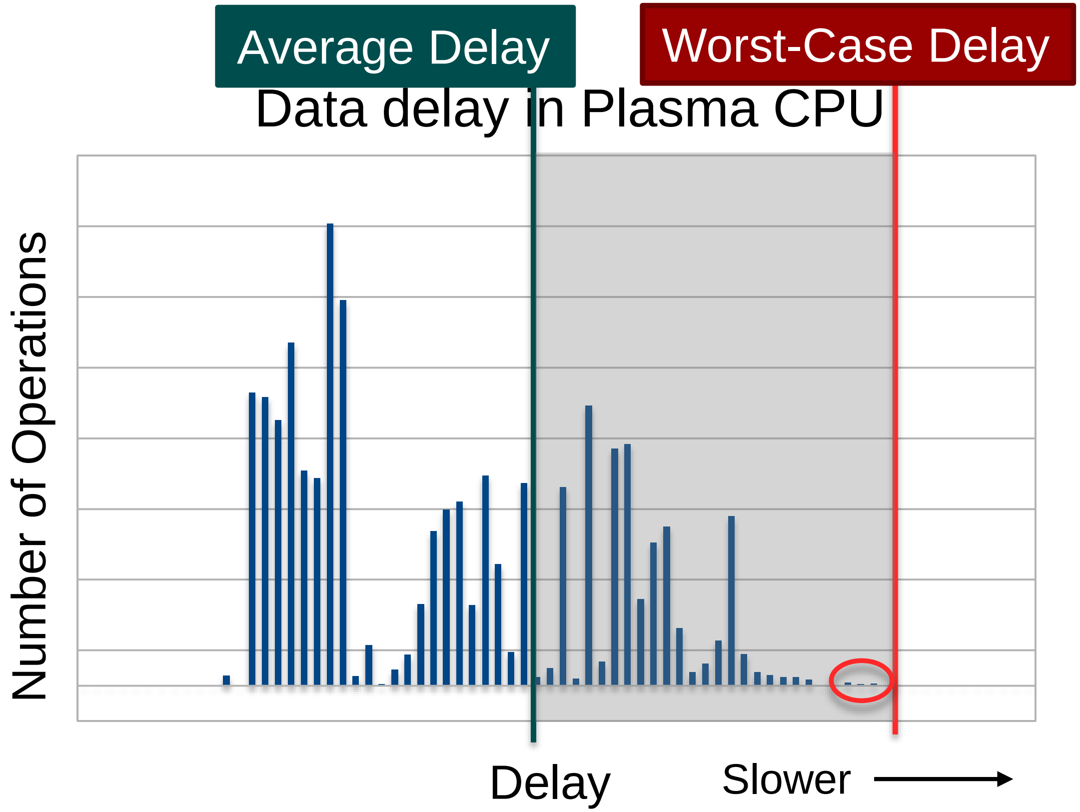




Delay variation due to data is rarely exploited in synchronous designs

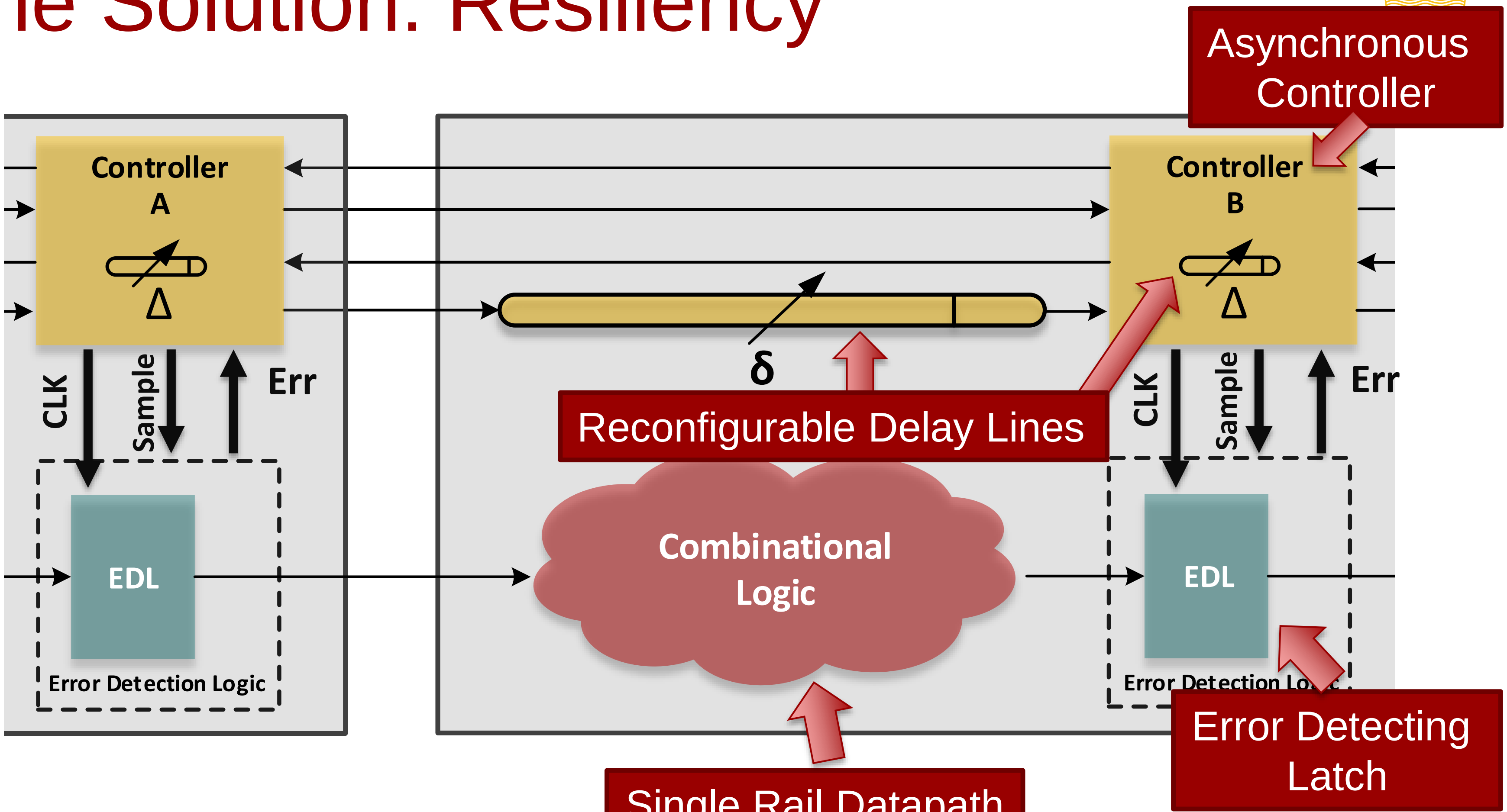
Potential of Average-Case Data





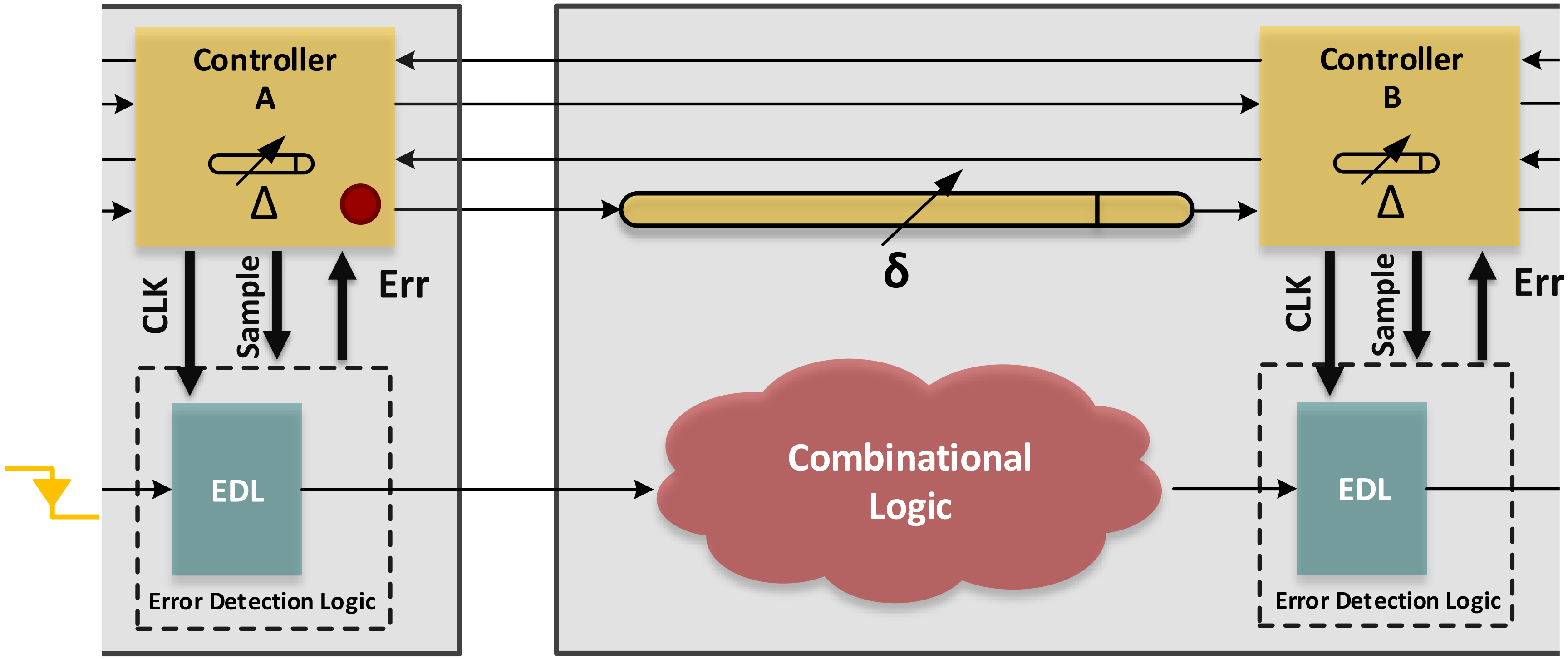
Delay variation due to data is rarely exploited in synchronous designs

One Solution: Resiliency



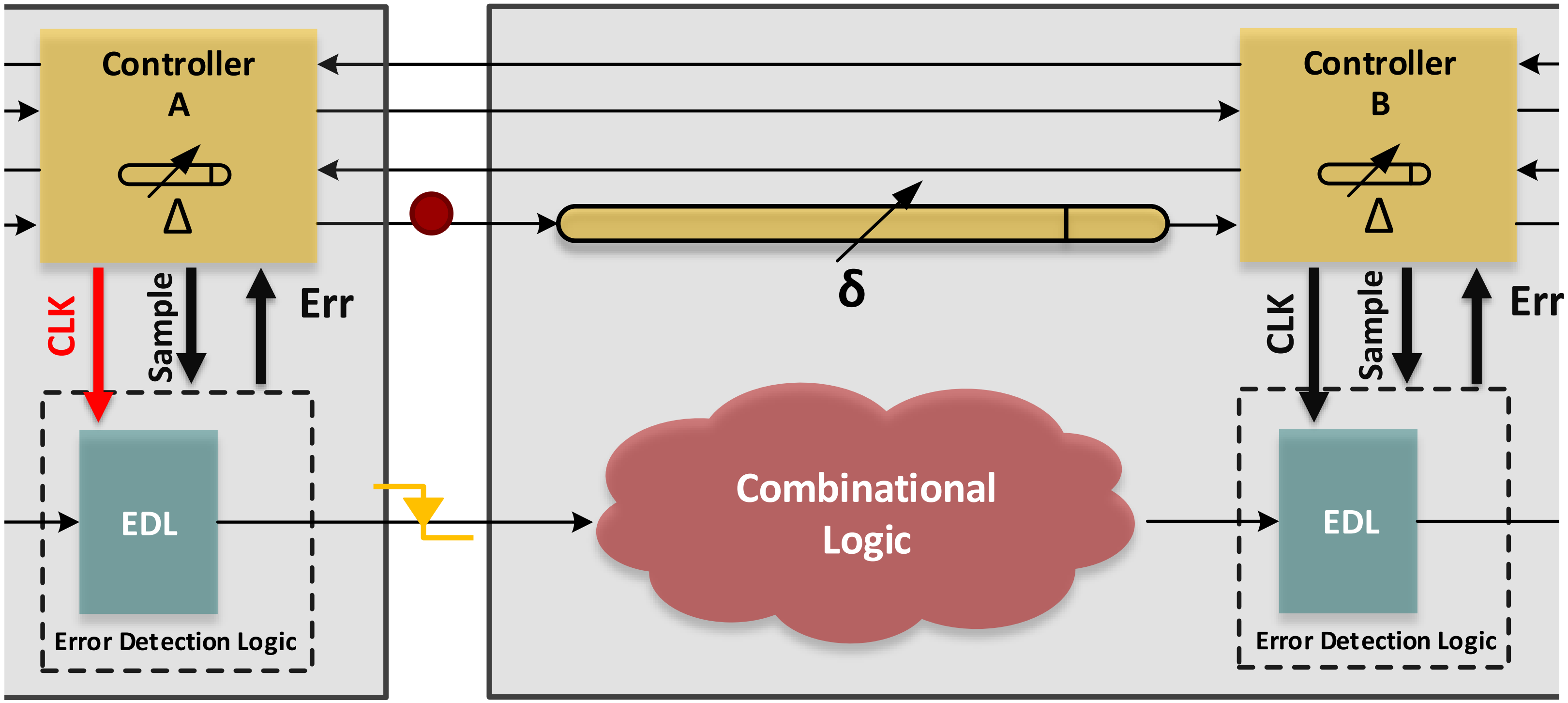
Timing errors delay handshaking by the resiliency window Δ

One Solution: Resiliency



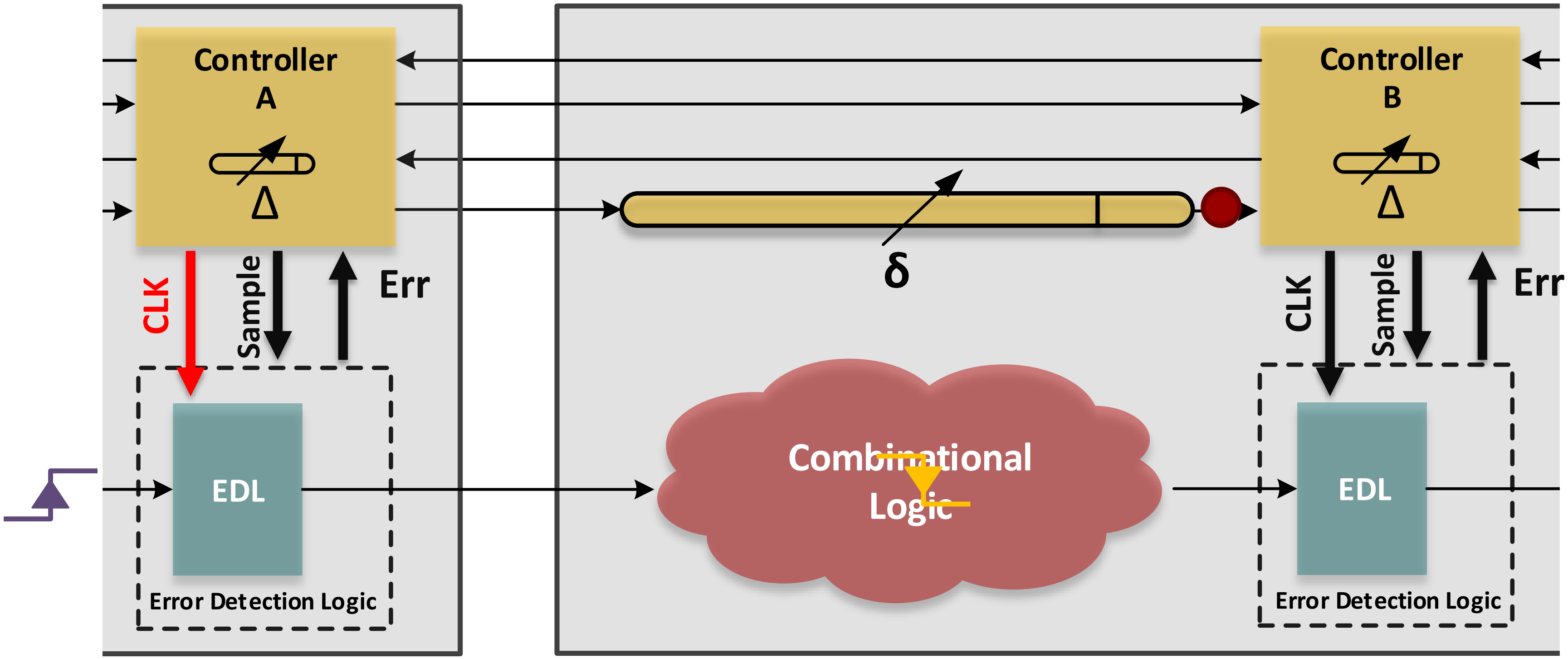
Timing errors delay handshaking by the resiliency window Δ

One Solution: Resiliency



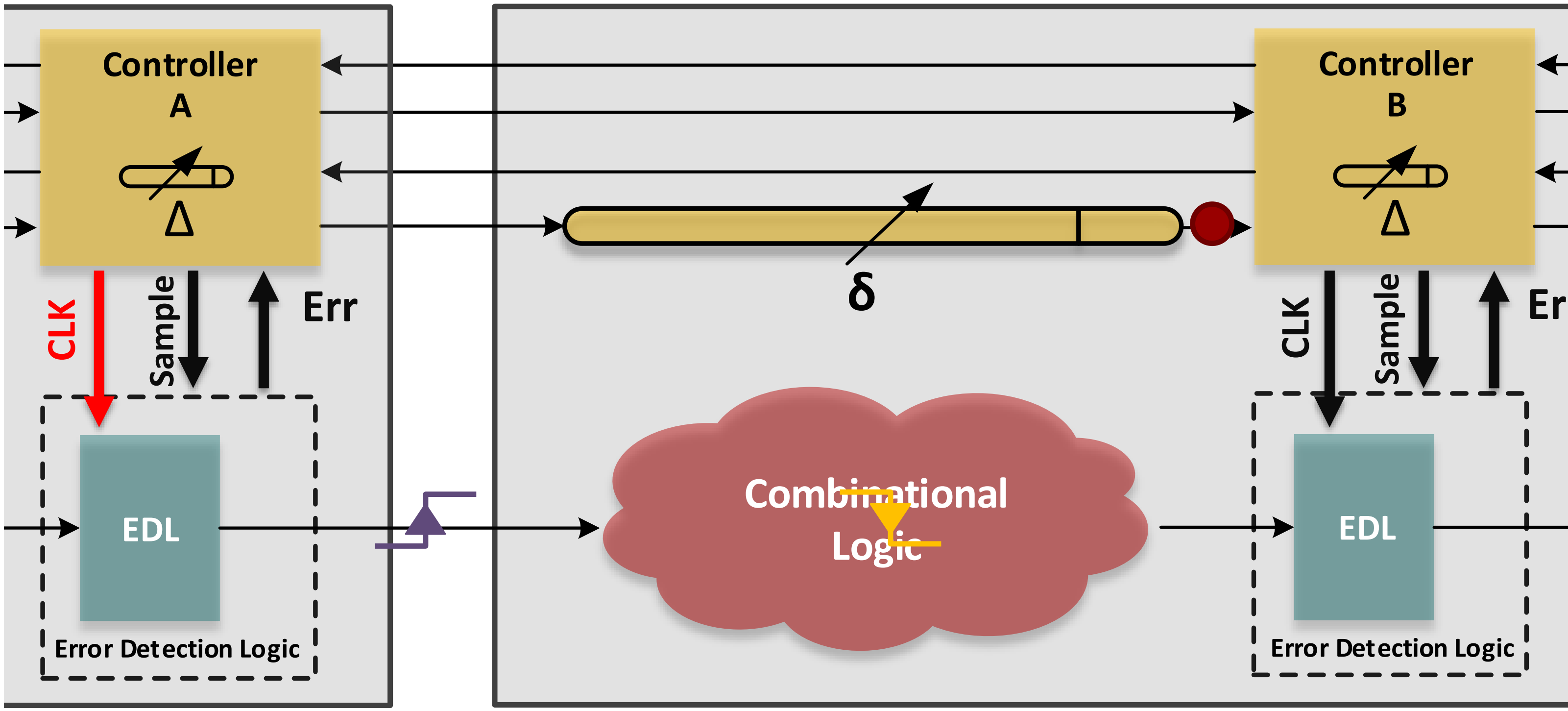
Timing errors delay handshaking by the resiliency window Δ

One Solution: Resiliency



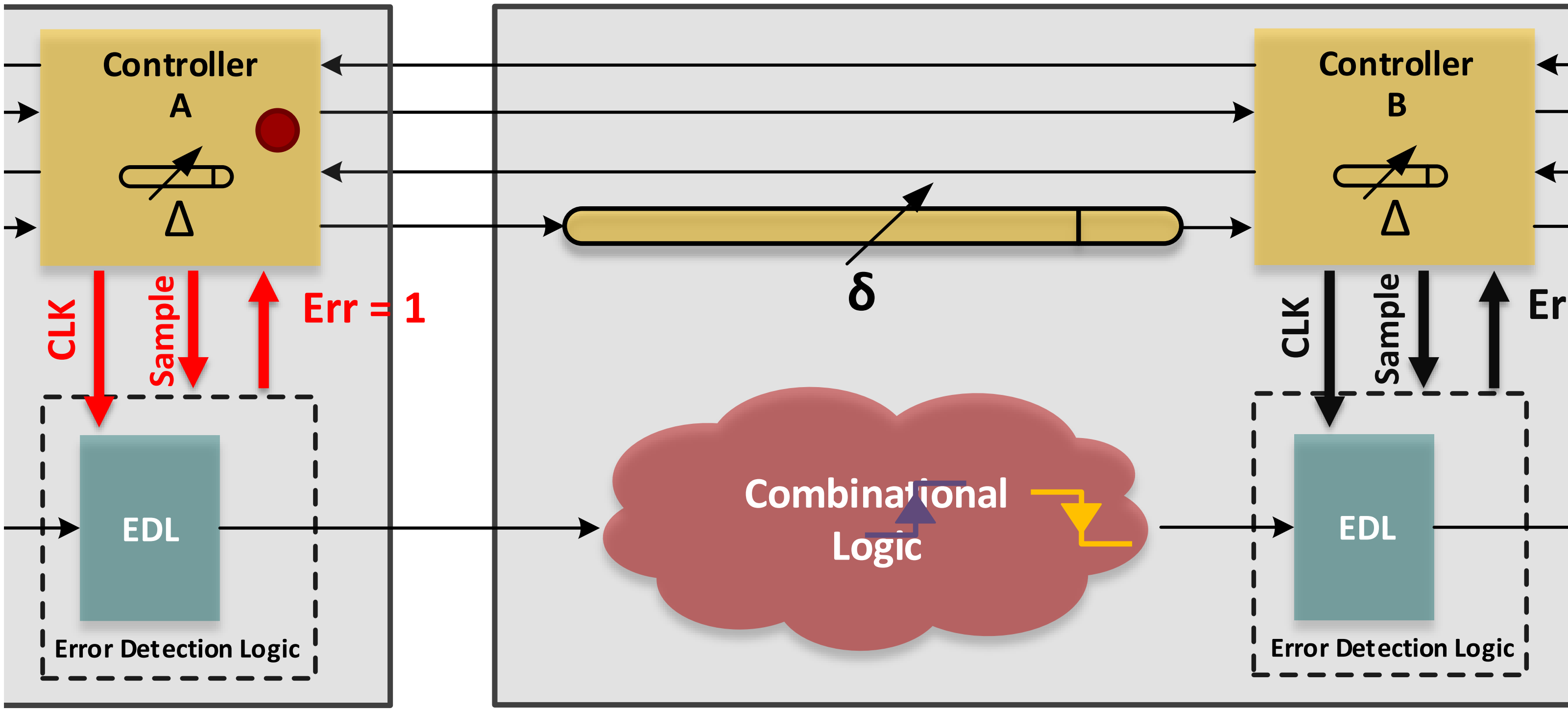
Timing errors delay handshaking by the resiliency window Δ

One Solution: Resiliency



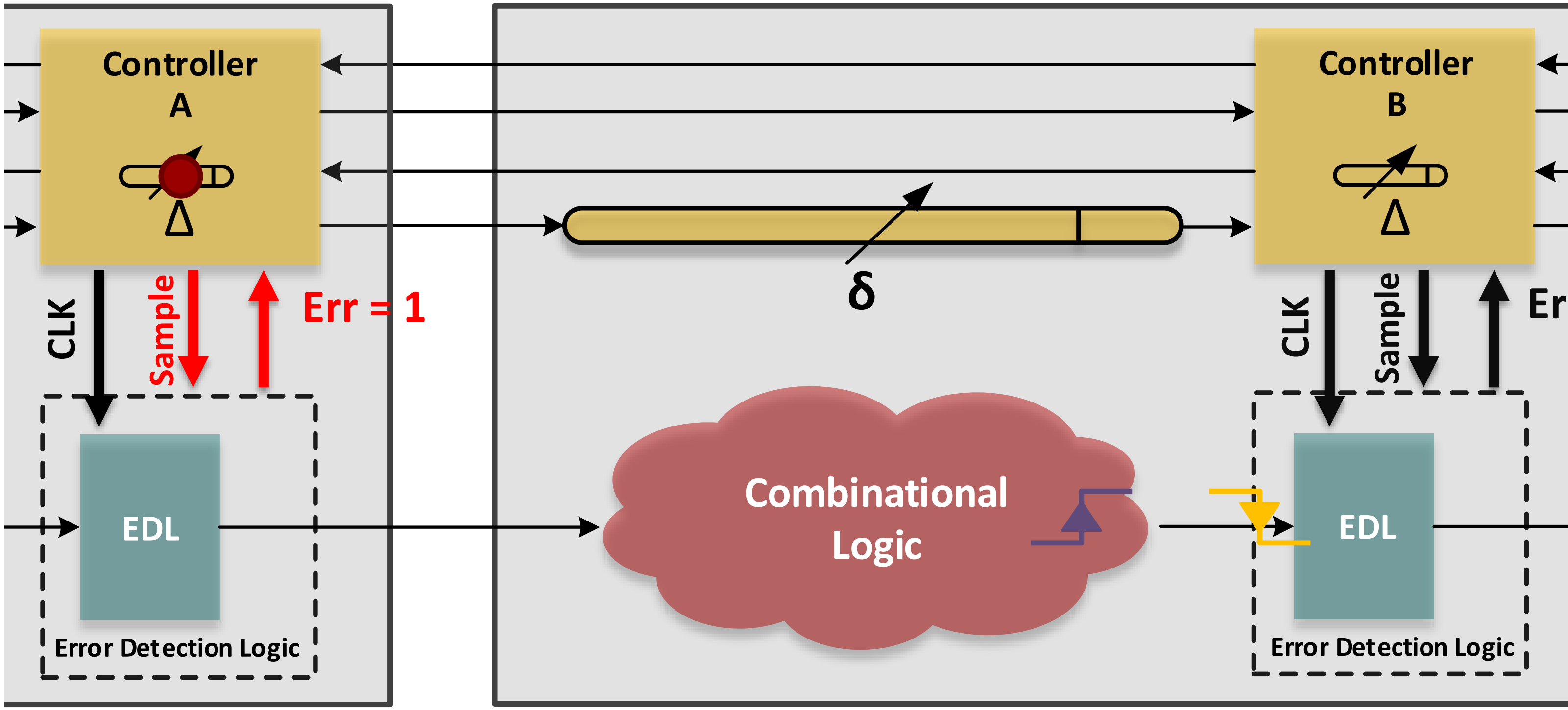
Timing errors delay handshaking by the resiliency window Δ

One Solution: Resiliency



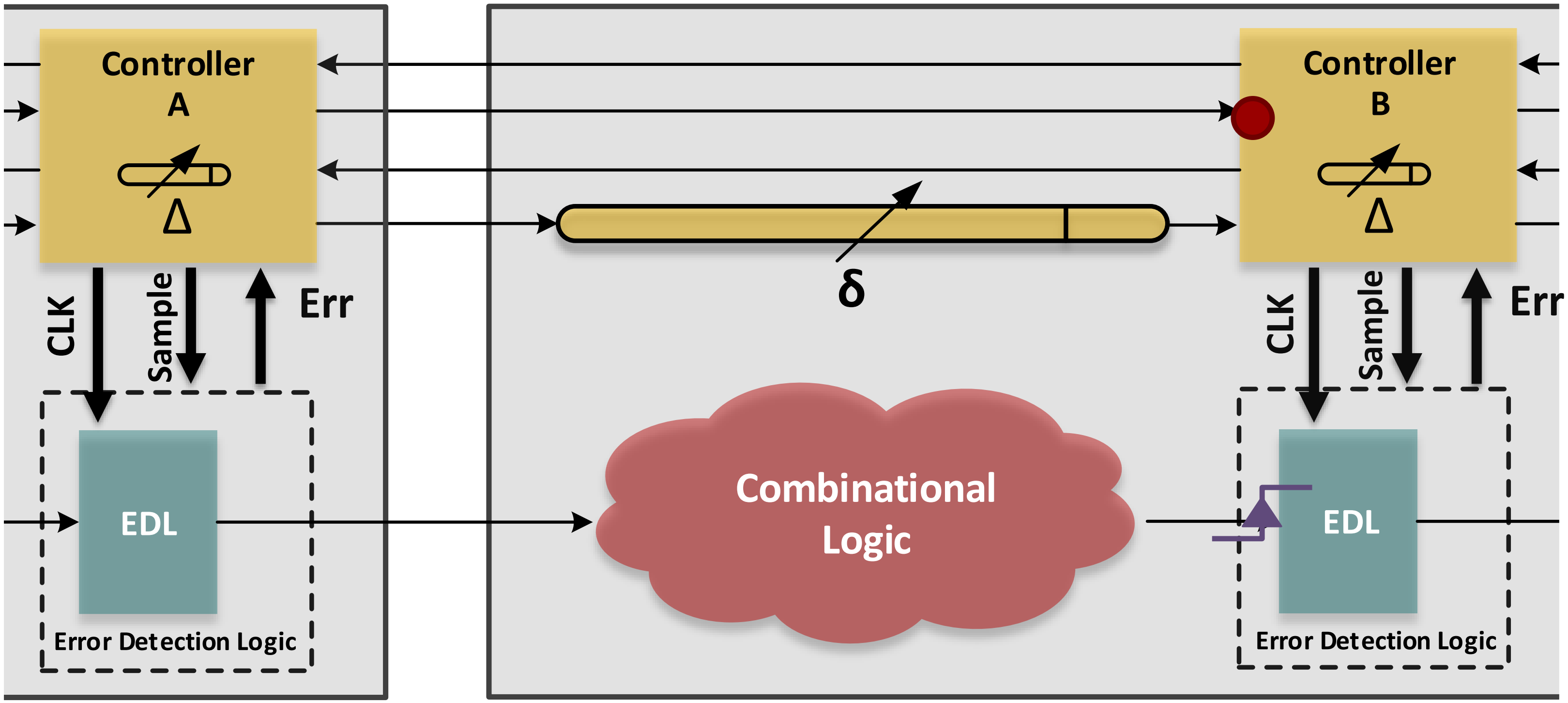
Timing errors delay handshaking by the resiliency window Δ

One Solution: Resiliency



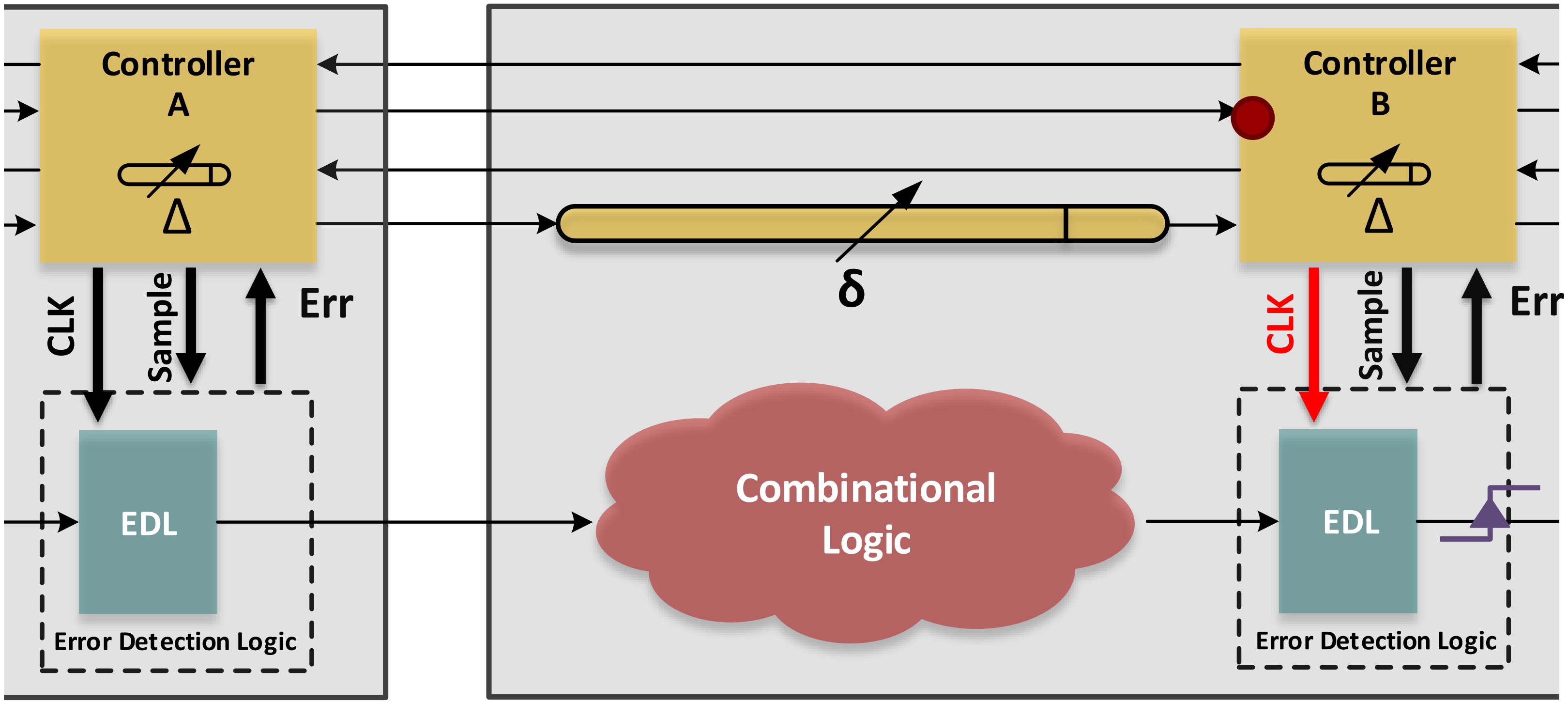
Timing errors delay handshaking by the resiliency window Δ

One Solution: Resiliency



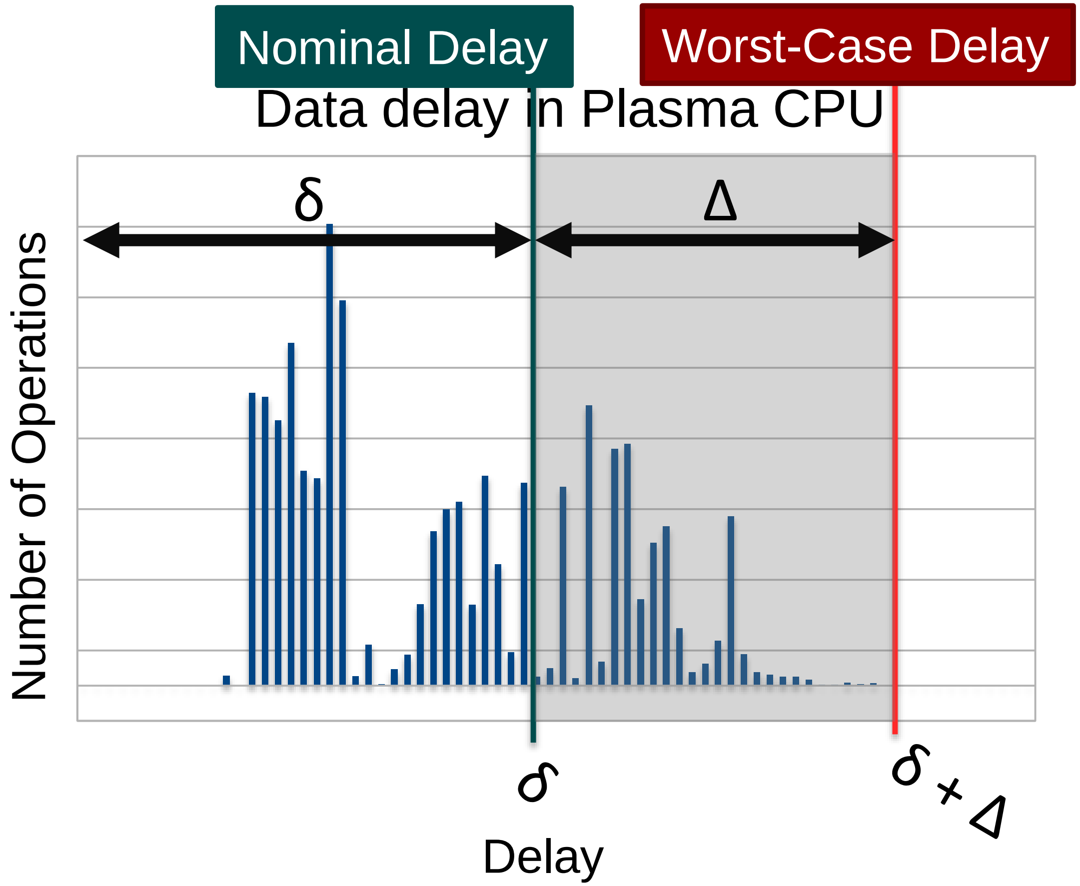
Timing errors delay handshaking by the resiliency window Δ

One Solution: Resiliency



Timing errors delay handshaking by the resiliency window Δ

Resiliency Performance Benefit



Key Question: How do we set δ to optimize performance

Outline

Performance Optimization

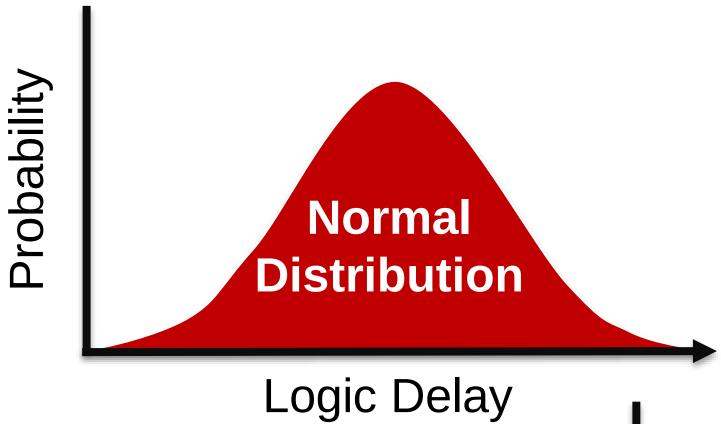
- Delay models
- Impact of delay line quantization
- Impact of metastability
- Comparison to Bubble Razor

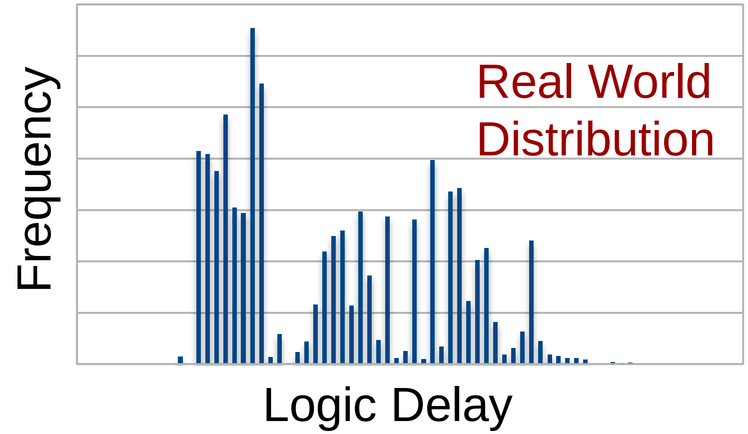
Case Study

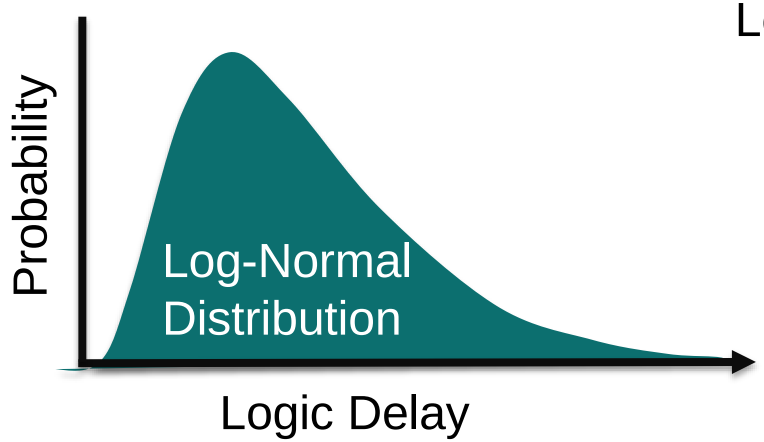
- Analyze and optimize a 3-stage Blade CPU

Conclusions and Future Work

Delay Models



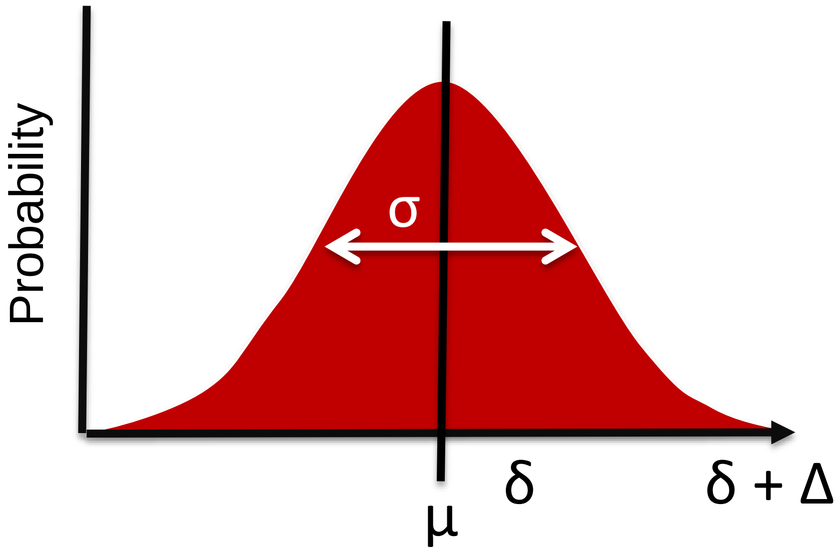




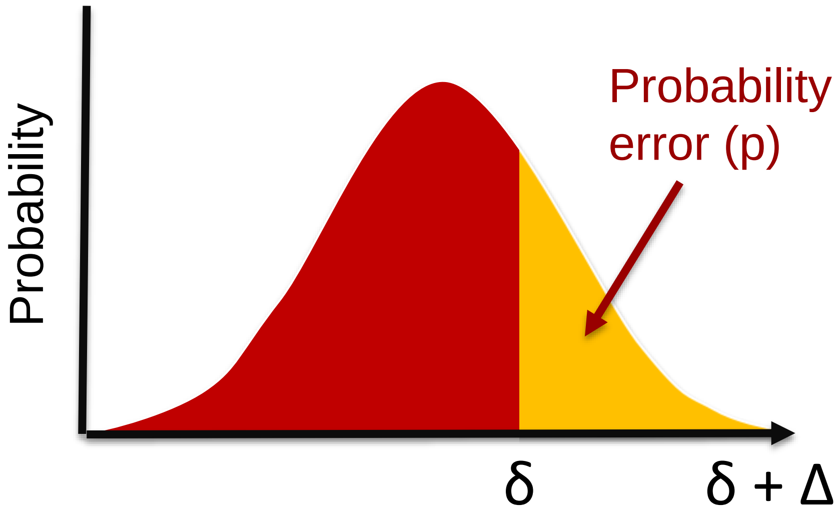
Our approach

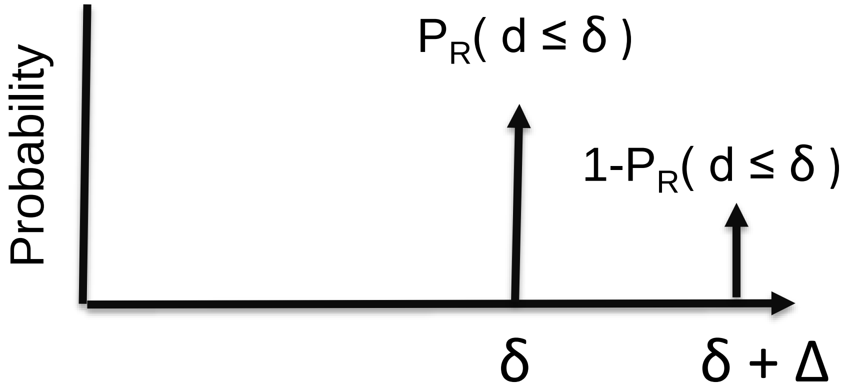
- Analyze the performance of Blade for a variety of delay models

Optimal Average-Case Performance



Optimal Average-Case Performance





: Average delay of Blade stage

Definitions

- C : Clock Period / Cycle Time
- EC : Effective Clock Period
- p : Probability of error
- d : Average delay of Blade stage

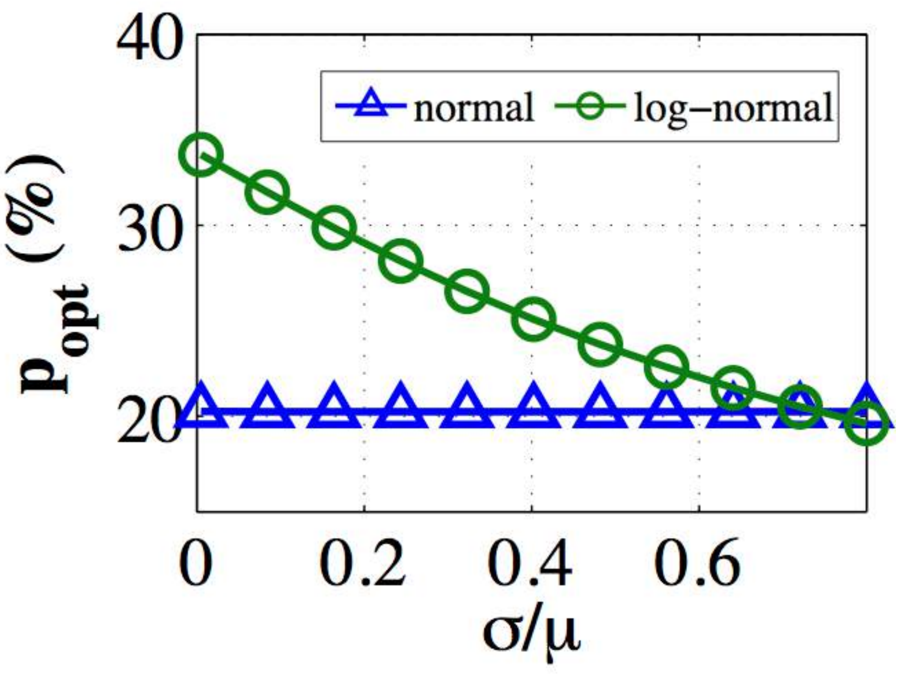
$$C = \delta + \Delta$$

$$\bar{d} = \delta + p * \Delta$$

Optimal performance achieved by minimizing d

Assumes backward latency is hidden via latch retiming

Optimal Probability of Error - p_{opt}



Higher Variance

p_{opt} observations

- Varies between 20% and 35% for log-normal distributions
- Significantly higher than in sync resiliency
- Constant for normal distributions!

Proof of constant p_{opt}

Assume worst case delay per stage is constant $K = \delta + \Delta$

Worst case delay is set by mean, variance, and SER $K = \mu + m * \sigma$

Systematic Error Rate (ξ) sets the worst-case delay per stage, K

$$\xi = 1 - [P_R\{d \leq C\}]^N$$

$$m = f(\xi)$$

Proof of constant p_{opt}

Assume worst case delay per stage is constant $K = \delta + \Delta$

Worst case delay is set by mean, variance, and SER $K = \mu + m * \sigma$

Recall: $\bar{d} = \delta + p * \Delta = (1 - p) * \delta + p * K$

For Normal distribution:

$$(1 - p) = \frac{1}{2}\left[1 + \text{erf}\left(\frac{\delta - \mu}{\sqrt{2}\sigma}\right)\right]$$

$$(1 - 2p) = \text{erf}\left(\frac{\delta - \mu}{\sqrt{2}\sigma}\right)$$

Taking inverse error function of both sides:

$$\text{erf}^{-1}(1 - 2p) = \frac{\delta - \mu}{\sqrt{2}\sigma}$$

$$\boxed{\delta = \sqrt{2}\sigma[\text{erf}^{-1}(1 - 2p)] + \mu}$$

Proof of constant p_{opt}

Assume worst case delay per stage is constant $K = \delta + \Delta$

Worst case delay is set by mean, variance, and SER $K = \mu + m * \sigma$

Recall: $\bar{d} = \delta + p * \Delta = (1 - p) * \delta + p * K$

Rewrite: $\bar{d} = (1 - p)[\sqrt{2}\sigma[\mathrm{erf}^{-1}(1 - 2p)] + \mu] + \mathrm{p} * \mathrm{K}$

Minimize $\bar{d}$ by taking derivative and setting it equal to zero :

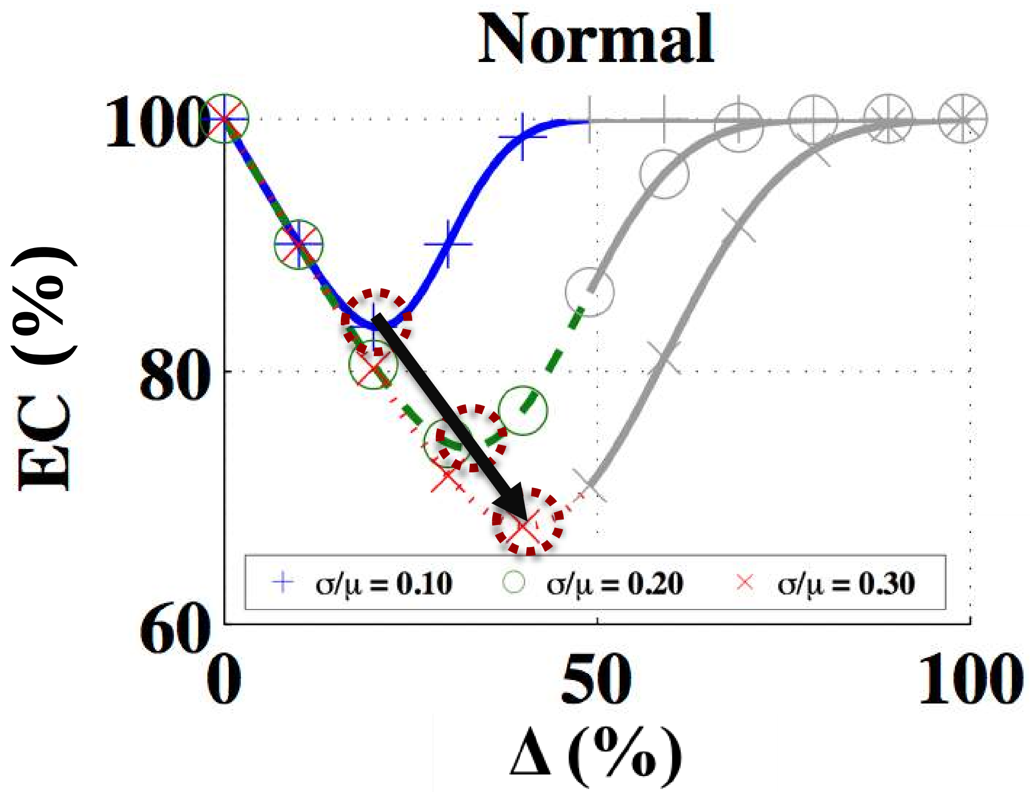
$$\frac{\partial \bar{d}}{\partial p} = (1 + y)\left[\sqrt{2}\frac{\partial \mathrm{erf}^{-1}(y)}{\partial y}\right] - \sqrt{2}\,\mathrm{erf}^{-1} y + m = 0$$

$$y = 1 - 2p$$

Note *y* and *p* are independent of σ and μ! *m* depends only on ξ

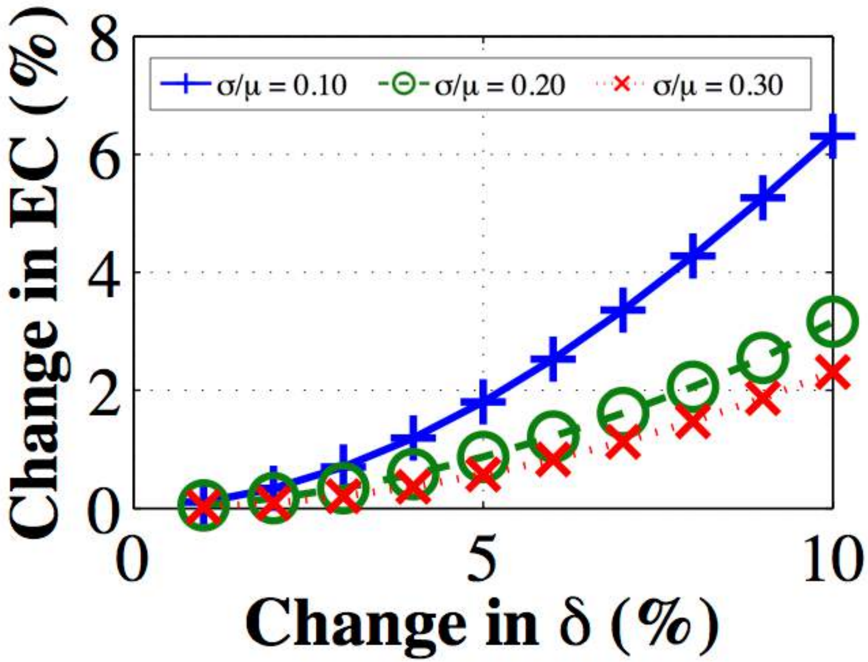
Implication: Tuning of delay line may target fixed probability!

Optimal Size of Resiliency Window - Δ

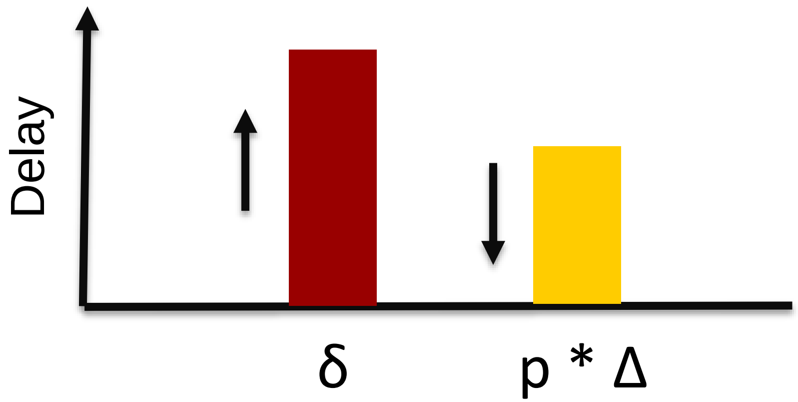


Blade supports maximum Δ of 50% of clock cycle
Optimal Δ is larger for designs with high-variance!

Delay Line Quantification Effects

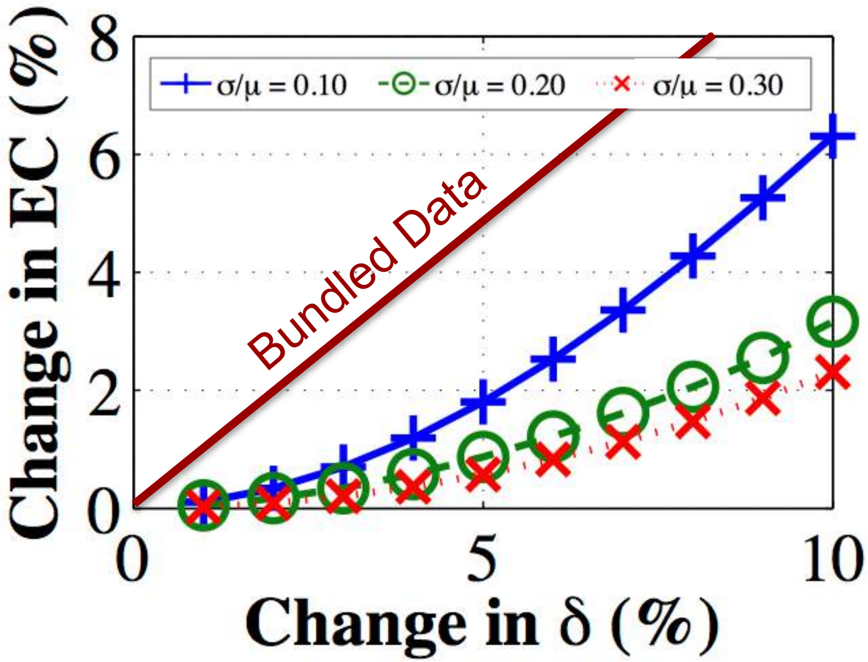


Recall: $\bar{d} = \delta + p * \Delta$

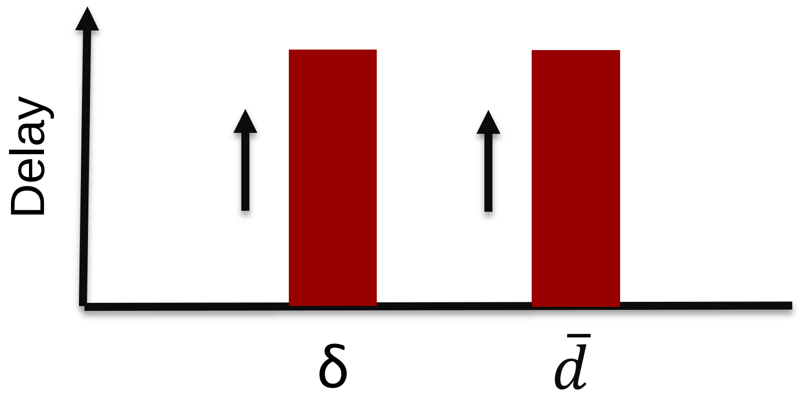


*Quantification effects reduced due to inherent tradeoff between nominal delay δ and error penalty p * Δ*

Delay Line Quantification in BD



Bundled Data: $\bar{d} = \delta$



Linear relationship between delay line quantization and average stage delay in Bundled Data

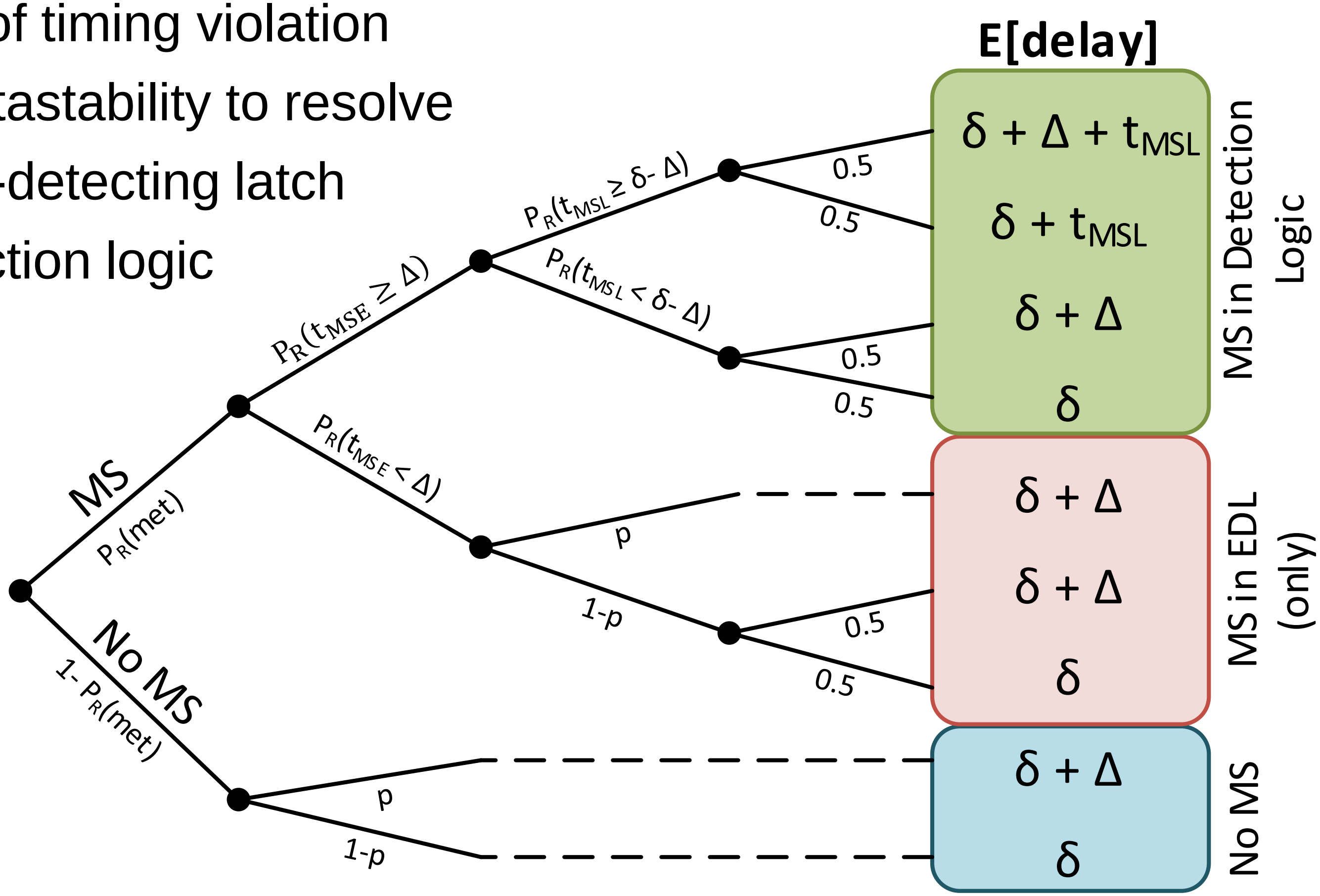
Metastability Effects

p : probability of timing violation

t_{MS} : time for metastability to resolve

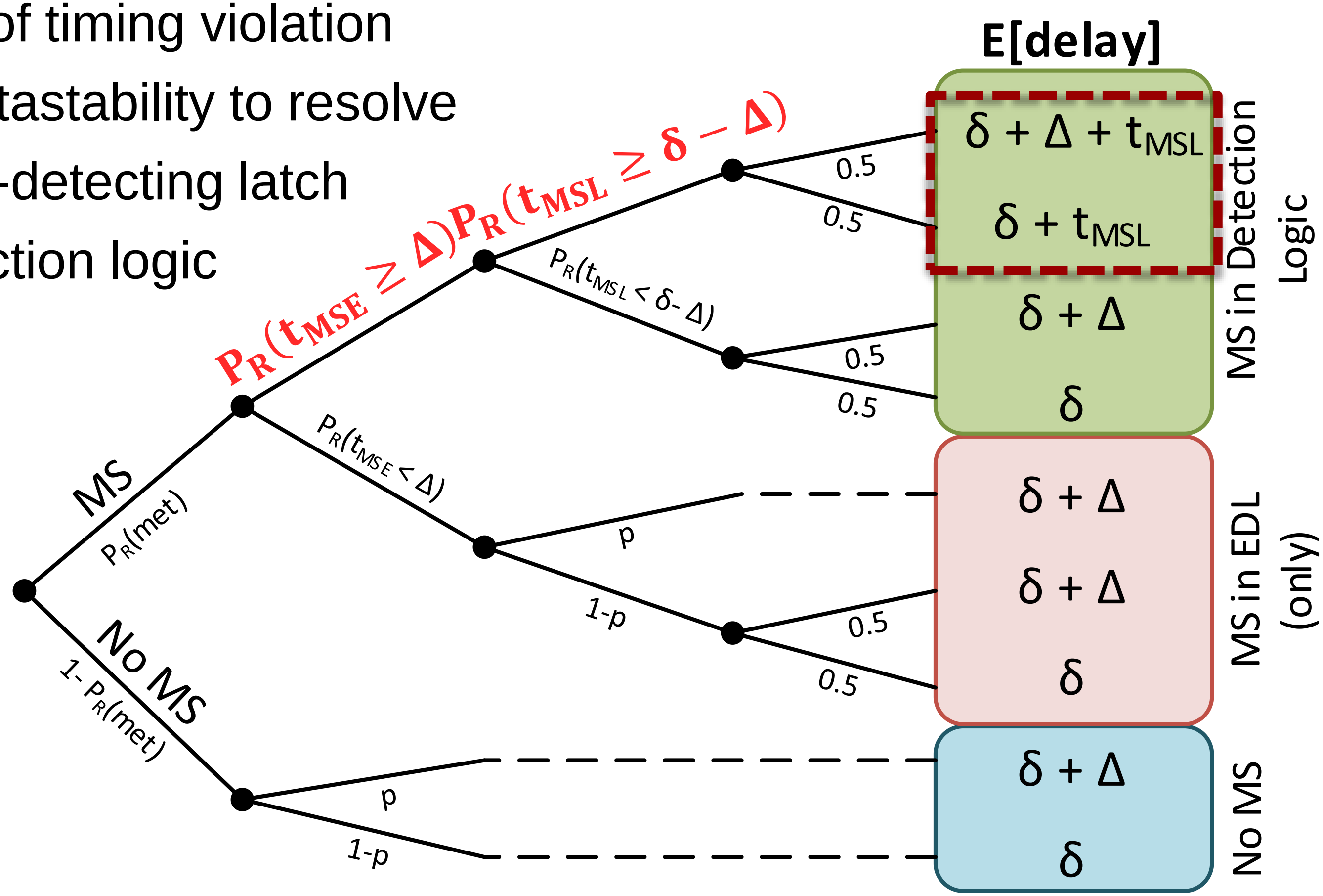
t_{MSE} : MS in error-detecting latch

t_{MSL} : MS in detection logic



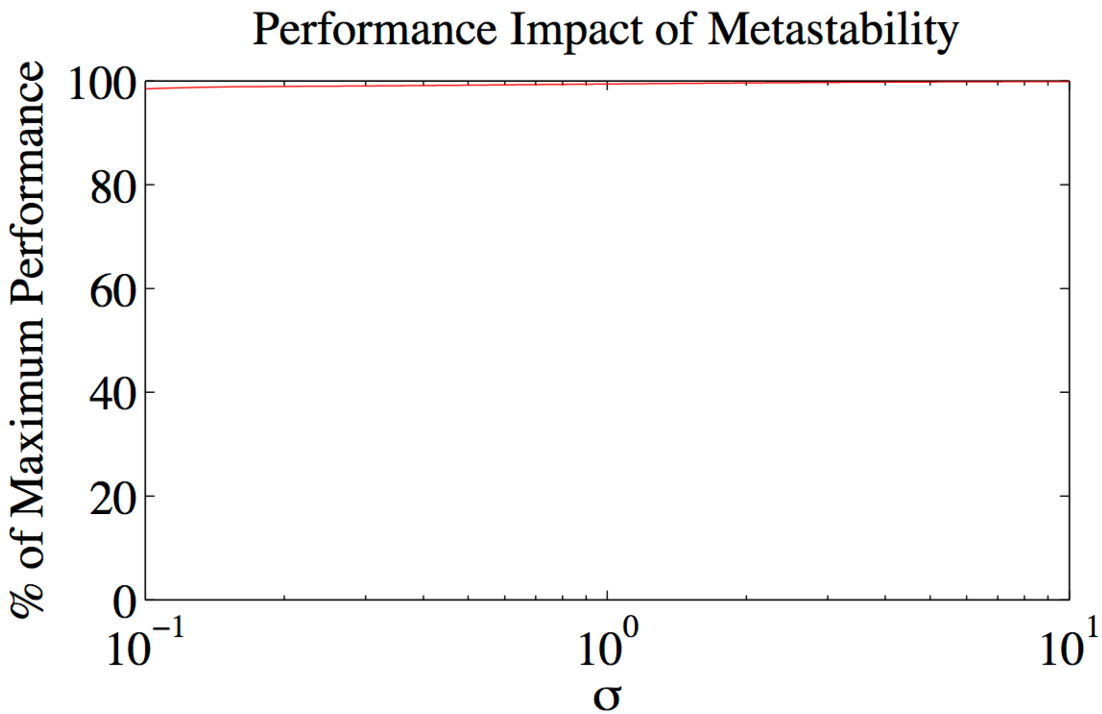
Metastability Effects

- p : probability of timing violation
- t_{MS} : time for metastability to resolve
- t_{MSE} : MS in error-detecting latch
- t_{MSL} : MS in detection logic



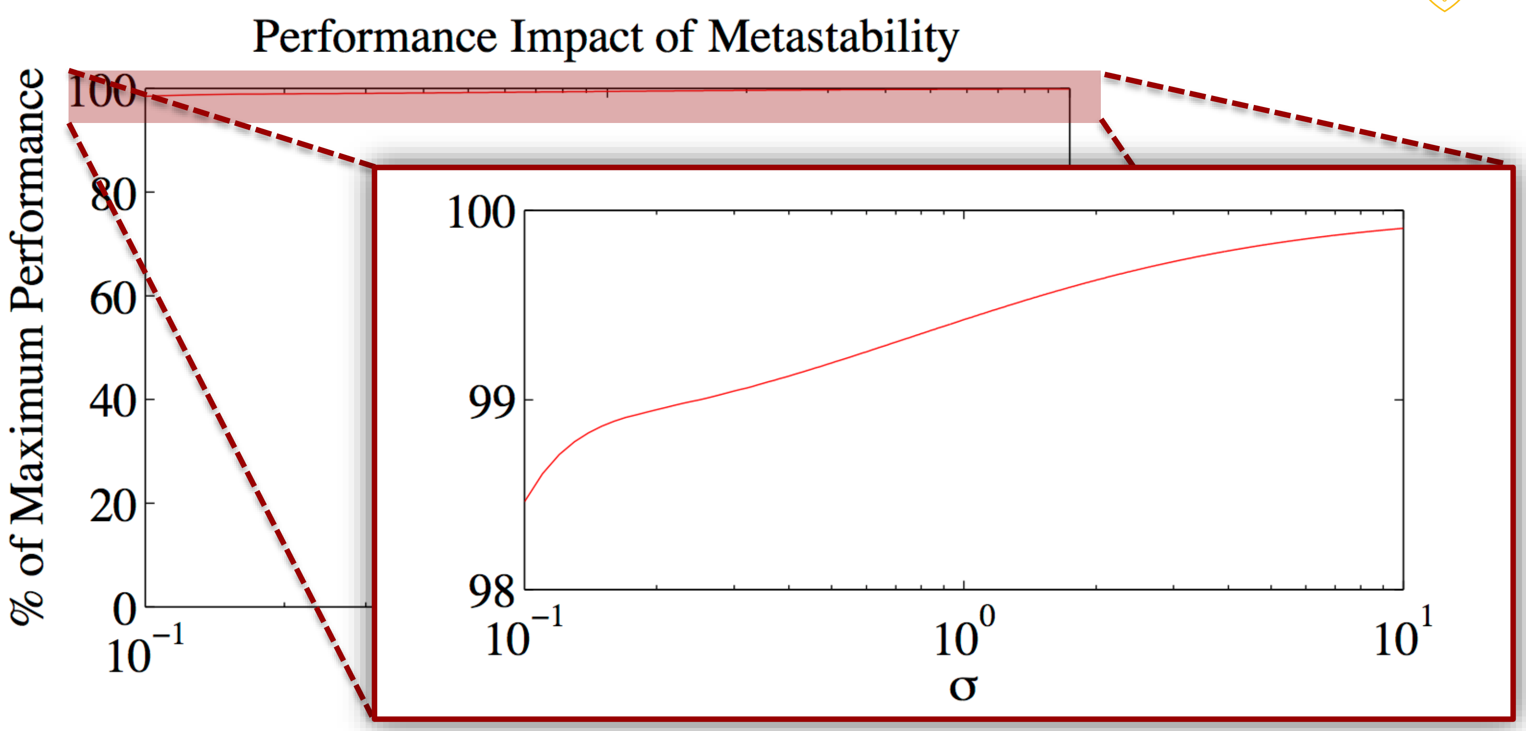
Metastability resolution times most often hidden!

Metastability Effects



$$P_R(met) = \int_{\delta - \frac{W_1}{2}}^{\delta + \frac{W_1}{2}} N(x, \mu, \sigma^2) dx \qquad P_R(t_{MST} \geq T | met) = e^{-\lambda_C T}$$

Metastability Effects



$$P_R(met) = \int_{\delta - \frac{W_1}{2}}^{\delta + \frac{W_1}{2}} N(x, \mu, \sigma^2)dx \qquad P_R(t_{MST} \geq T | met) = e^{-\lambda_C T}$$

Comparison to Sync Resiliency

Synchronous

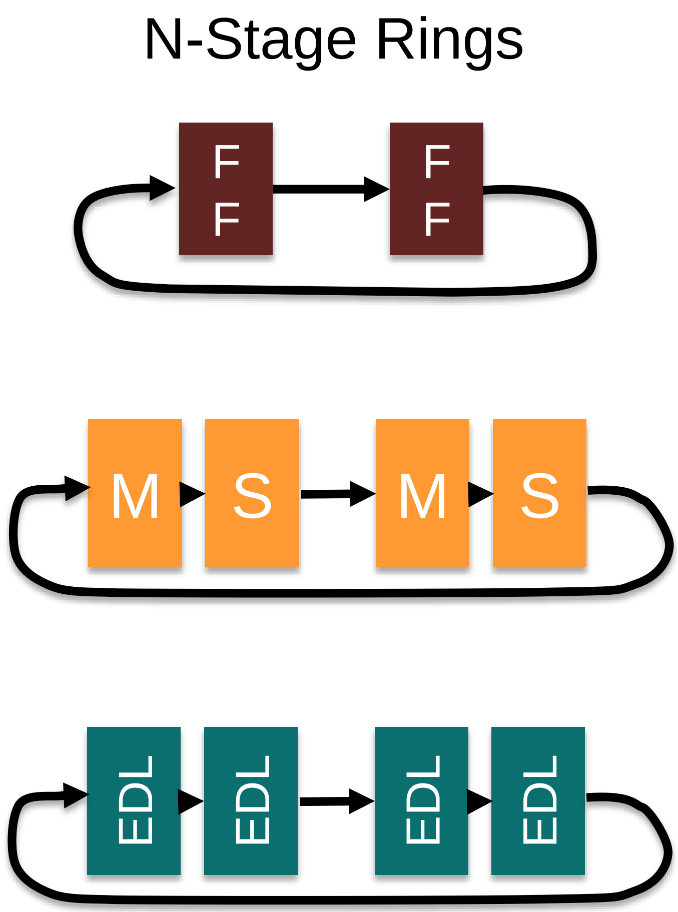
- EC set by systematic error rate

Bubble Razor [Zhang,2014]

- $EC = C[2 - (1 - p)^{2N}]$

Blade

- $EC = \delta + p_{opt} * \Delta$



Comparison to Sync Resiliency

Synchronous

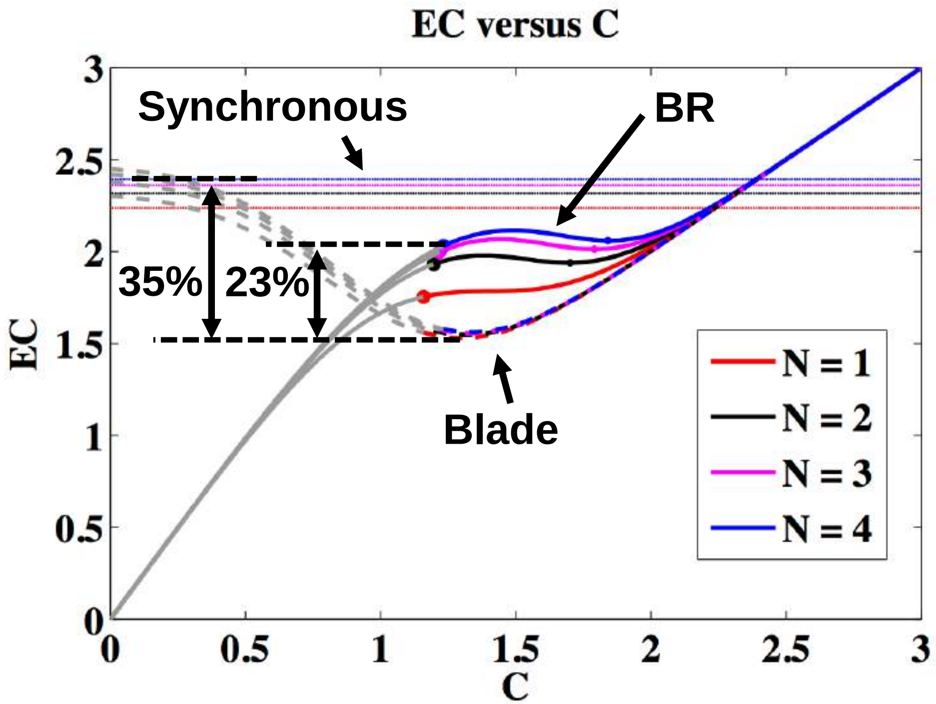
- EC set by systematic error rate

Bubble Razor [Zhang,2014]

- $EC = C[2 - (1 - p)^{2N}]$

Blade

- $EC = \delta + p_{opt} * \Delta$



Normal Distribution

Comparison to Sync Resiliency

Synchronous

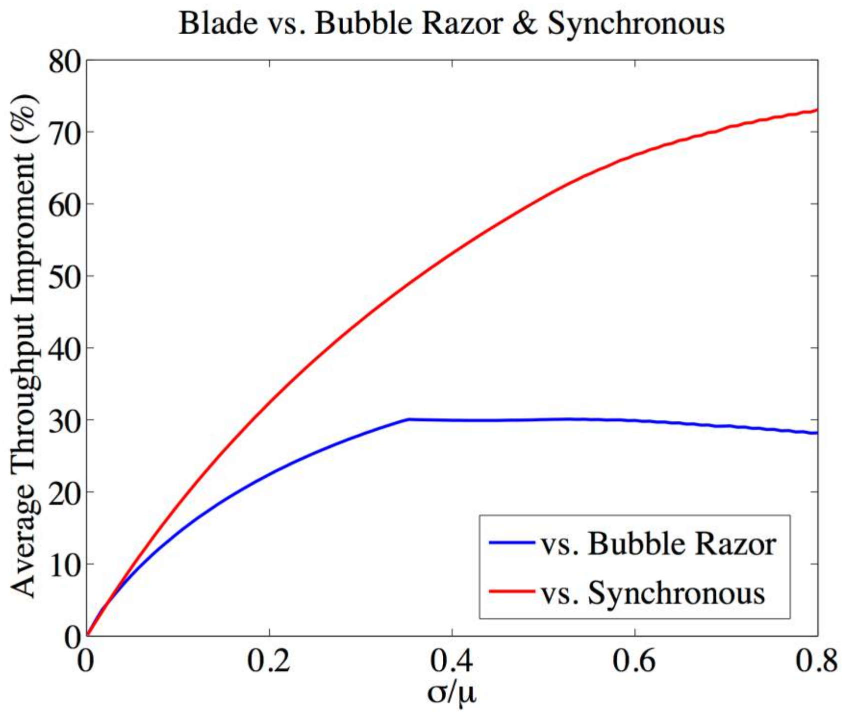
- EC set by systematic error rate

Bubble Razor [Zhang,2014]

- $EC = C[2 - (1 - p)^{2N}]$

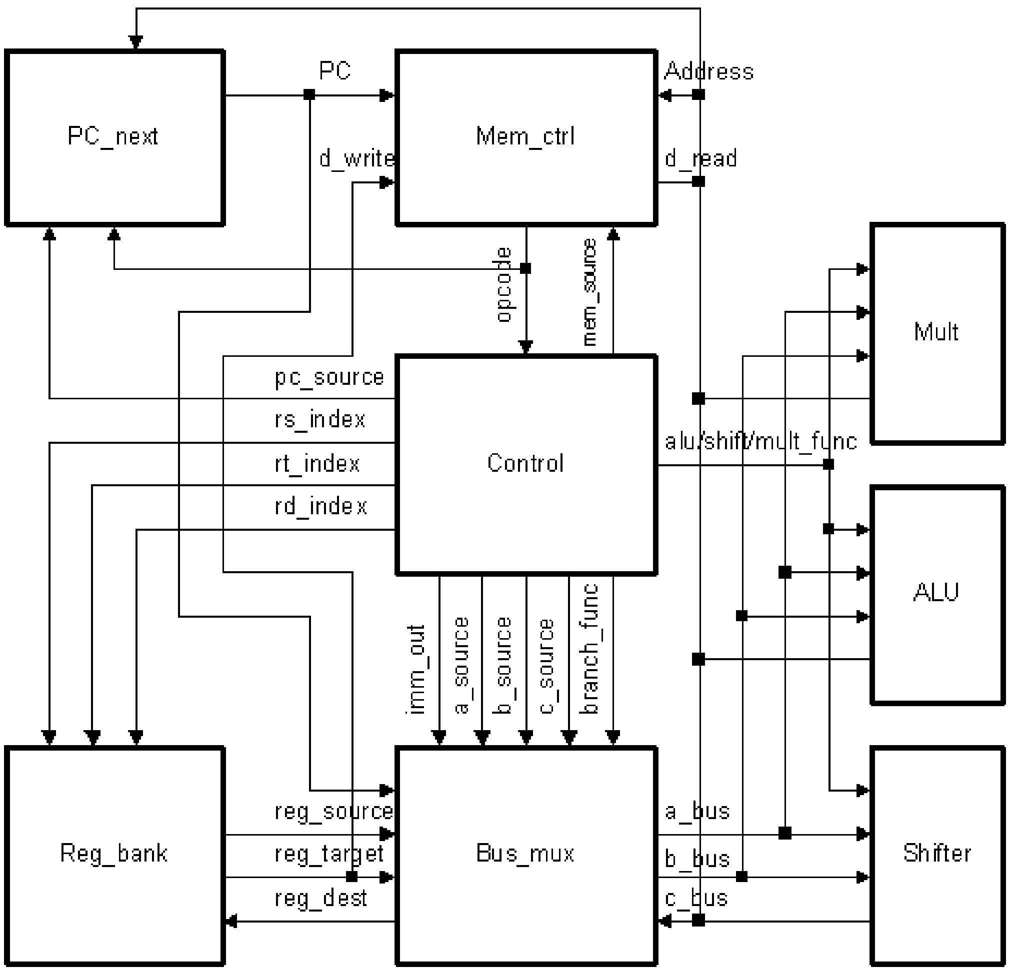
Blade

- $EC = \delta + p_{opt} * \Delta$



Normal Distribution

Application to 3-Stage CPU



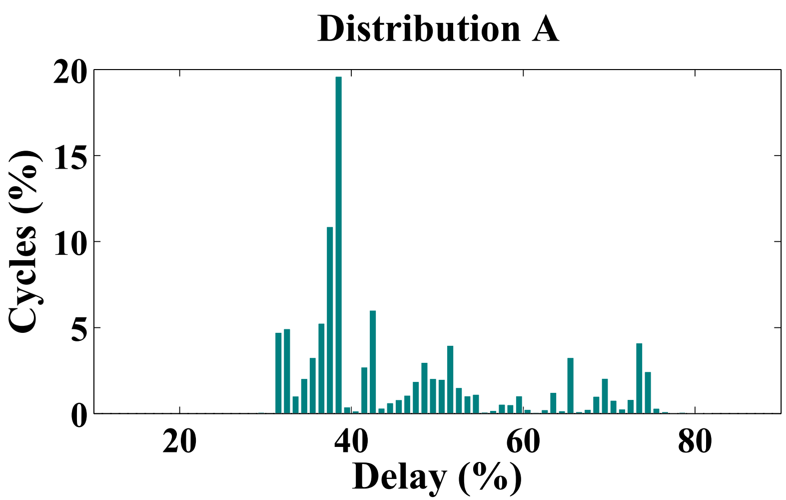
Plasma MIPS OpenCore

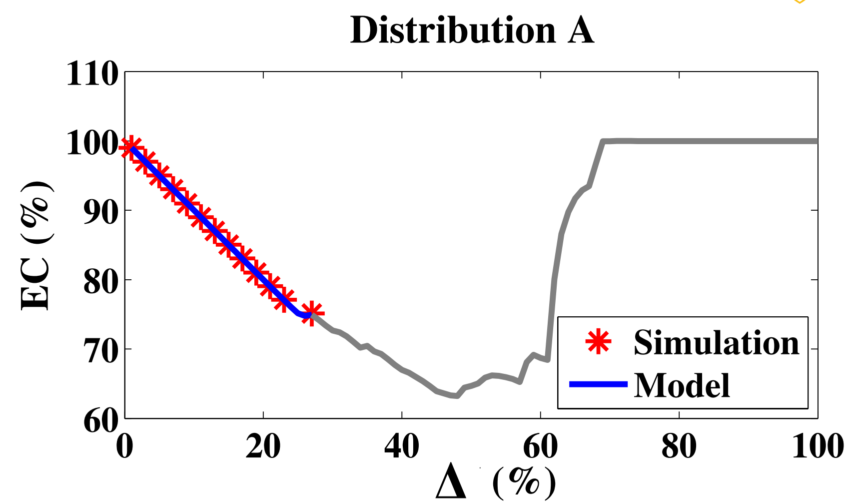
28nm FDSOI

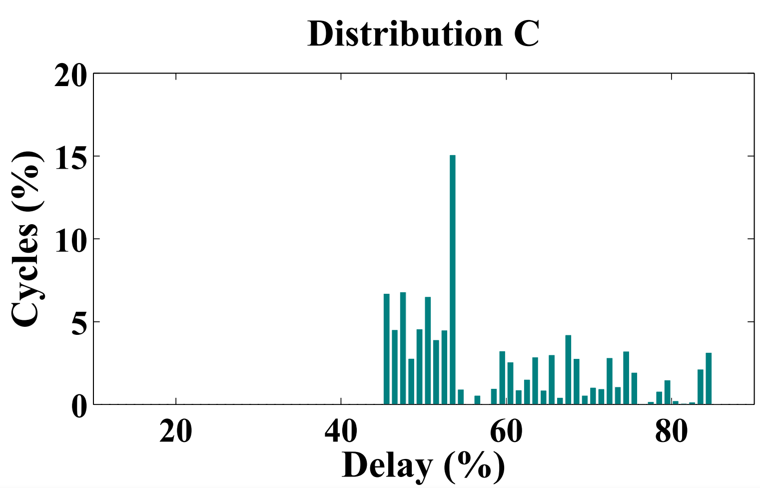
Compare mathematical model of optimal resiliency window (Δ) with simulation results

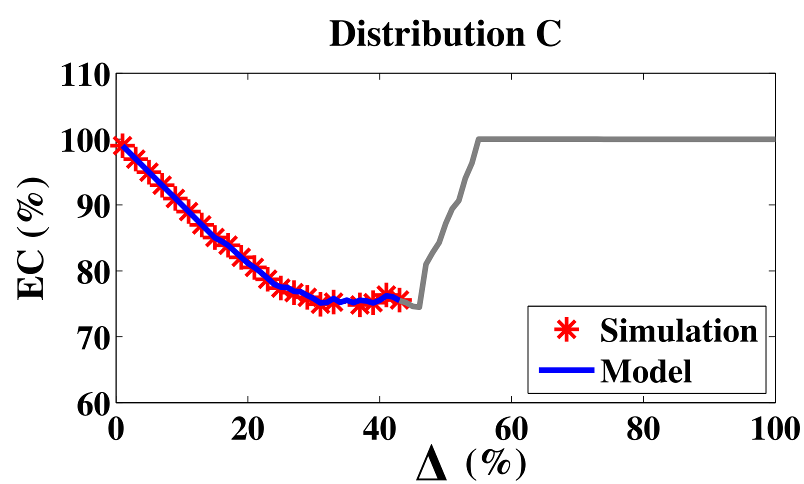
[1] http://opencores.org/project,plasma

Application to 3-Stage CPU









Application to 3-Stage CPU

	Distribution A		Distribution B		Distribution C	
	Model	Sim	Model	Sim	Model	Sim
Δ_{max}	27%		35%		43%	
Δ_{opt}	26%	27%	34%	35%	39%	37%
EC_{opt}	74.8%	75.1%	71.3%	71.9%	75.1%	74.8%

Model estimated optimal Δ within 5.4%

- Optimal EC within 99%

Application to 3-Stage CPU

	Distribution A		Distribution B		Distribution C	
	Model	Sim	Model	Sim	Model	Sim
Δ_{max}	27%		35%		43%	
Δ_{opt}	26%	27%	34%	35%	39%	37%
EC_{opt}	74.8%	75.1%	71.3%	71.9%	75.1%	74.8%
Ideal Δ_{opt}	*48%*		*53%*		*46%*	
Ideal EC_{opt}	*63.2%*		*67.8%*		*74.5%*	

Model estimated optimal Δ within 5.4%

- Optimal EC within 99%

Model allows estimation of optimal Δ w/o limitations of simulated design

Summary and Conclusions

Performance model

- Use either analytical and real world delay distributions
- Predicts performance within 99% accuracy

Comparison to sync N-stage rings

- 23% better than Bubble Razor
- 35% better than traditional designs

Several interesting conclusions

- Optimal error rate is relatively high and may be constant
- Programmable delay line need not be fine-grained
- Metastability impact is negligible
- Supporting larger resiliency windows may be useful

Questions?