DATE 13
Design, Automation & Test in Europe
18-22 March, 2013 - Grenoble, France
The European Event for Electronic System Design & Test

SlackProbe: A Low Overhead In Situ On-line Timing Slack Monitoring Methodology

Liangzhen Lai and **Puneet Gupta** – UCLA, US
Vikas Chandra and Rob Aitken – ARM, US

NanoCAD Lab

NSF

ARM®

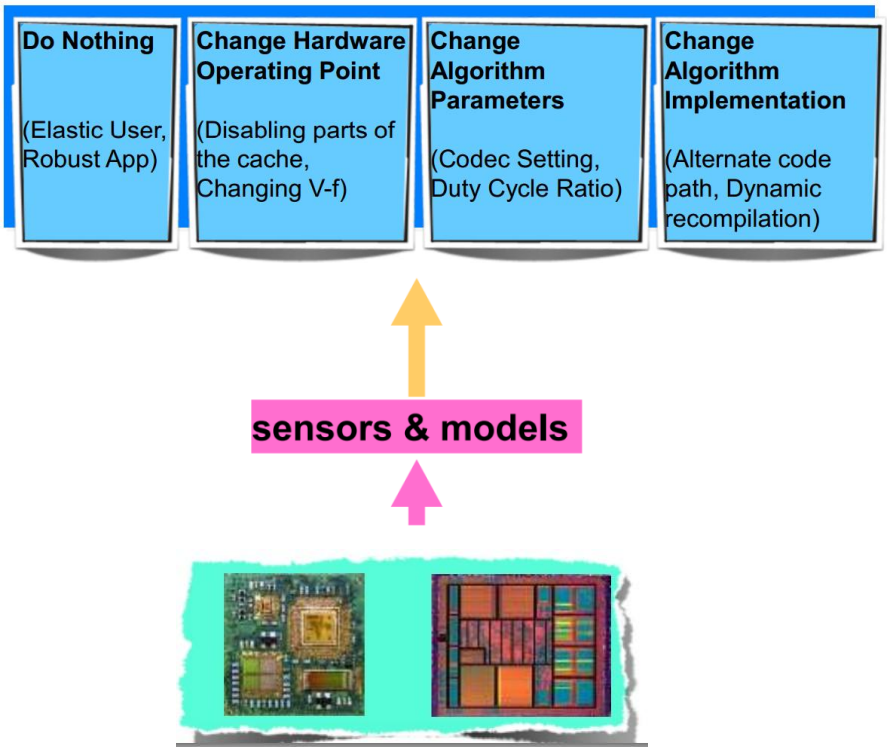
http://www.variability.org

Outline

- Motivation and Overview
- Monitoring Costs and Constraints
- Path Selection and Monitor Location Selection
- Experimental Results
- Conclusion

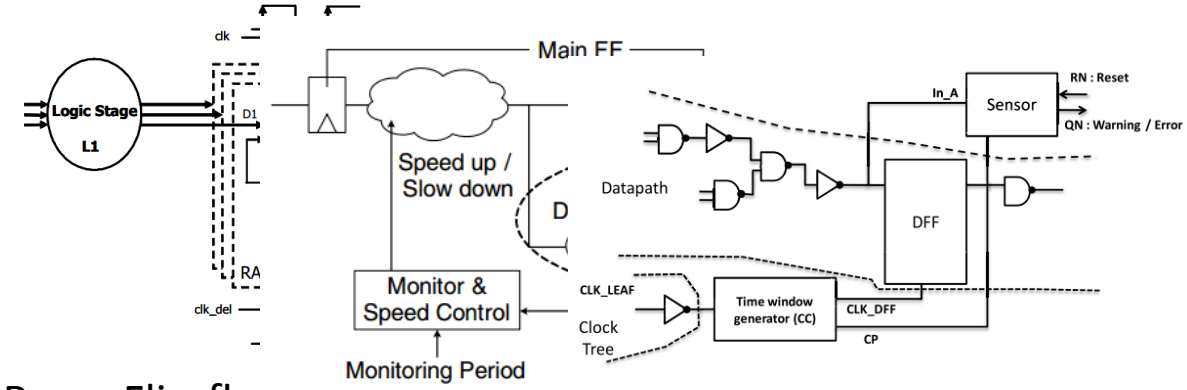
Motivation

- **Underdesigned and Opportunistic Computing**
  - Hardware monitors can help exploit the variability in conjunction of both *hardware* and ***software*** adaptation
- **Two class of delay monitors**
  - Replica monitor:
    - Cheap
    - Limited accuracy
  - In situ monitor
    - Accurate
    - Intrusive
    - High overhead



Prior Work

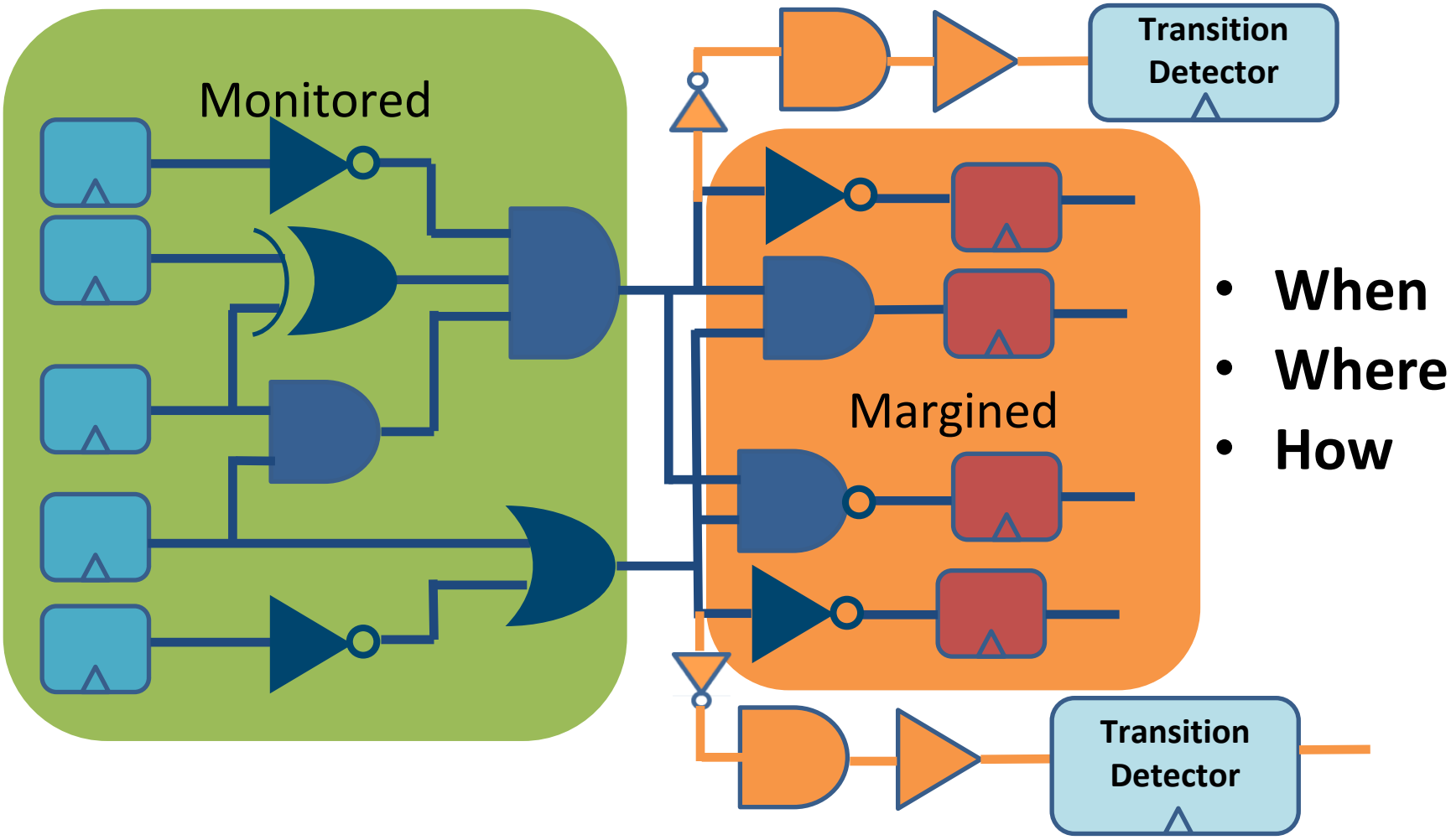
- **Rich literature on in situ timing slack monitoring**
  - (e.g. [MICRO03], [JSSC07], [TODAES07], [ISQED07], [JSSC09], [CICC09], [DFT10], [PATMOS09] etc.)
- **Mostly focus on monitor design and different ways of detecting errors**



Razor Flip-flo [CICC09]: dela [PATMOS09]: advancing clock

- **Considers inserting monitors *only* at destination registers → require large number of monitors**

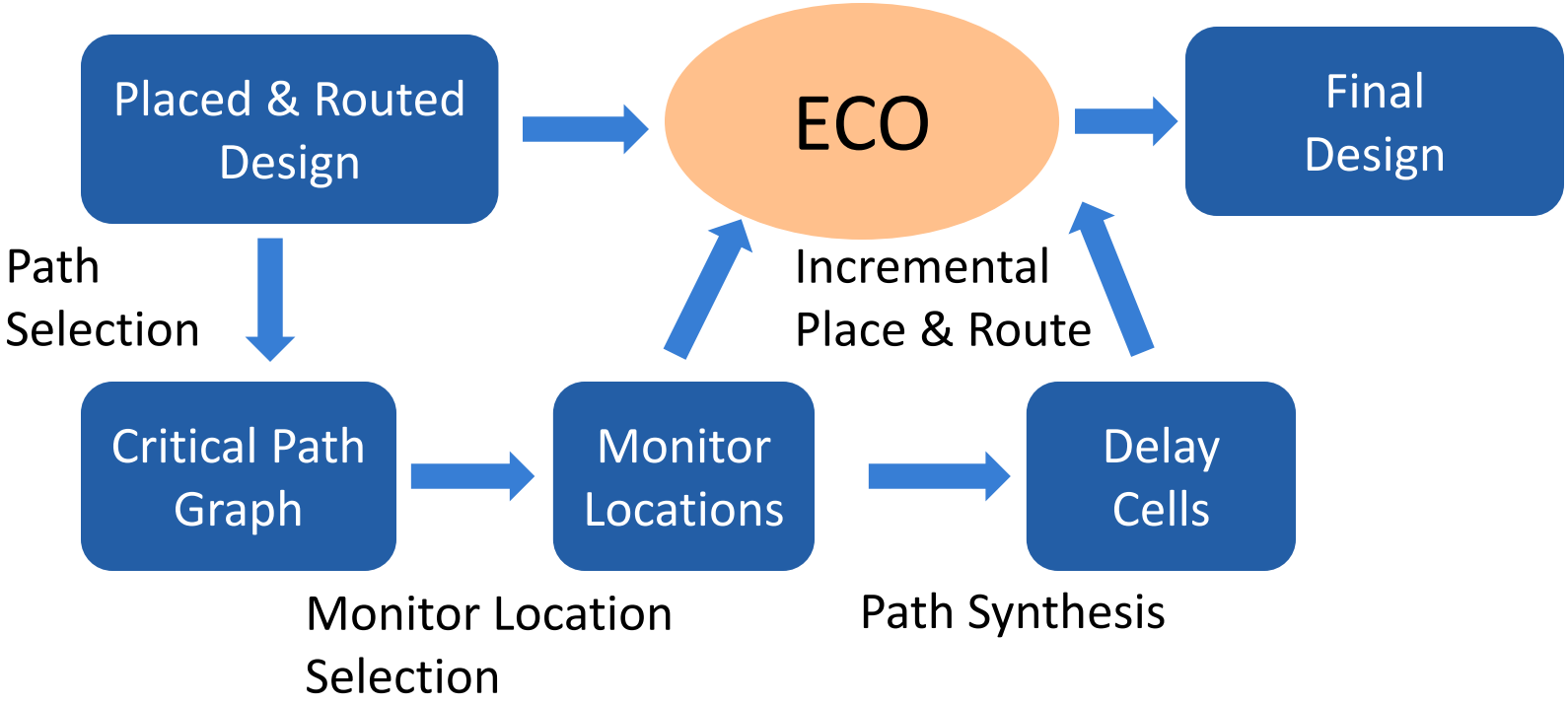
Slack *Probes*: Main Idea



- When
- Where
- How

Proposed Insertion Flow

- **Start with placed and routed design**
- **Incrementally place and route the monitors as Engineering Change Order (ECO)**

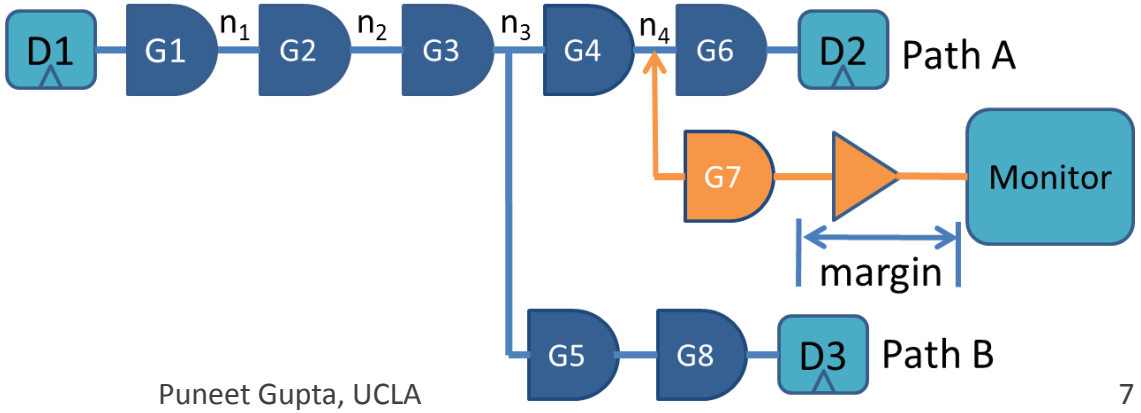


Outline

- Motivation and Overview
- Monitoring Costs and Constraints
- Path Selection and Monitor Location Selection
- Experimental Results
- Conclusion

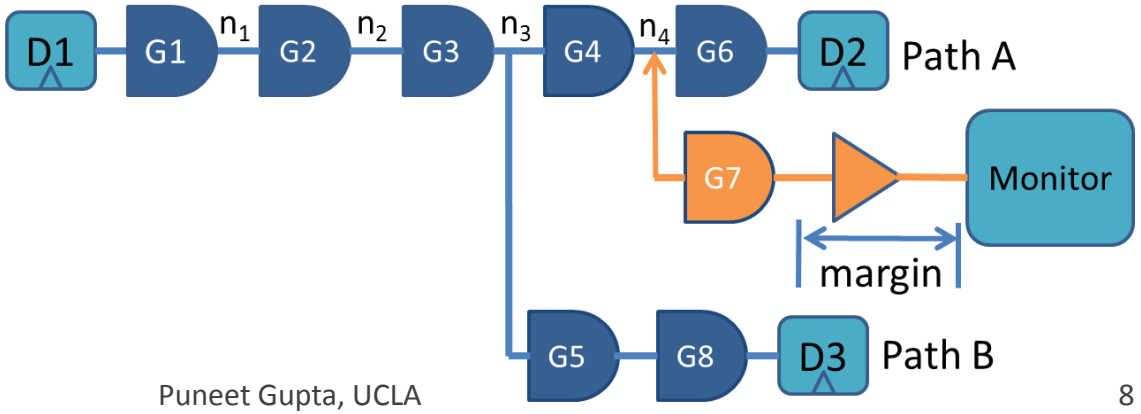
Monitoring a Path

- **Path A is monitored**
  - Need margin for unmonitored delay (i.e. G6)
- **Path B is *partially* monitored**
  - If intended application is transition detection
    - Need to detect every signal transition
    - Path B is monitored only when G4 is inverter/buffer
  - If intended application is speed sensing
    - Only to detect slow delay changes (e.g. process variation, aging etc.)
    - Paths passing through branching out net n_3 (i.e. Path B) is monitored
    - Need margin for delay of G4, G5 and G8



Delay Margin as Constraint

- **Tradeoff: monitoring coverage vs. margin required**
  - Monitor Path A only -> margin for G6
  - Monitor both Path A and B -> margin for G4, G5 and G8
- **Overall delay margin is dominated by largest margined monitor**
- **We define the margin cost as a maximum delay margin ε constraint on each monitor**
  - with a given margin ε, monitored branching out nets can be calculated



Monitor Cost Metric

- **Power Cost** $p_o + (\lambda_i p + l) d_i$
  - λ_i is the switching probability of net n_i, d_i is the delay of the path that is synthesized for monitor at n_i, *p and l* are the estimated dynamic and leakage power cost per unit delay, p_o is the static power overhead
- **ECO Cost** $a_t \cdot exp(-1 \times s_{n_i}) + a_r \cdot r_{n_i}$
  - s_{n_i} is the estimated timing slack after inserting the first minimum size inverter, r_{n_i} is the layout congestion, a_t and a_r are weighting parameters
- **Total Monitor Cost**

$$c_i = p_o + (\lambda_i p + l) d_i + a_t \cdot exp(-1 \times s_i) + a_r \cdot r_{n_i}$$

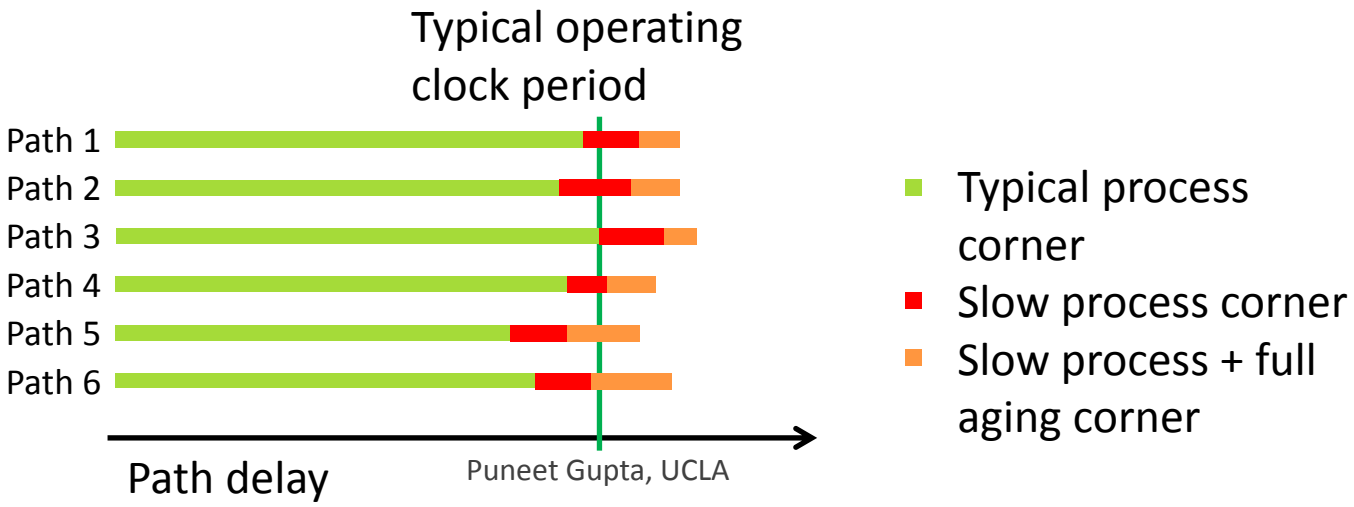
Outline

- Motivation and Overview
- Monitor Costs and Constraints
- Path Selection and Monitor Location Selection
- Experimental Results
- Conclusion

Corner-based Path Selection

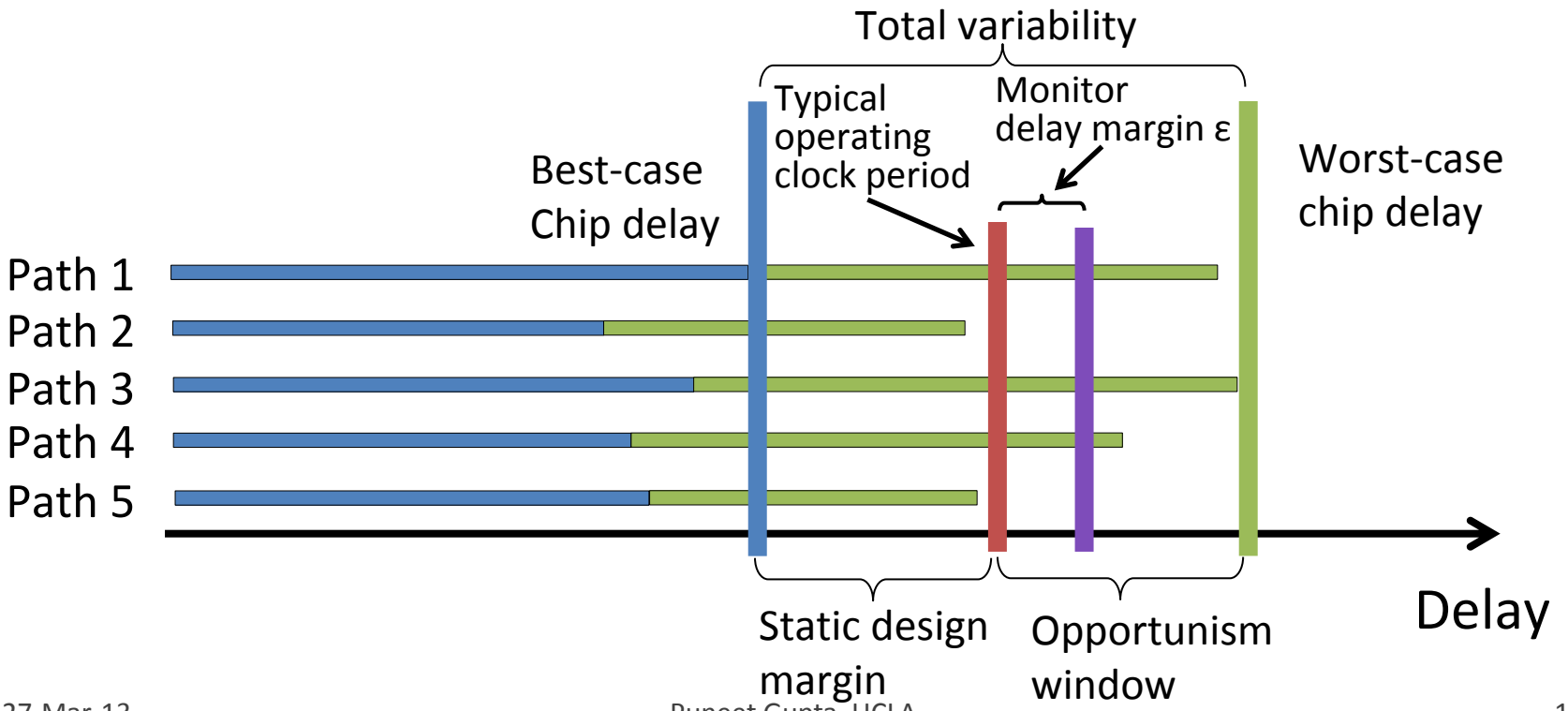
- **Define two corners:**
  - Typical operating corner
    - Targeting best-case (i.e. no flags reported by monitors) operating corner
  - Worst-case corner: margined worst case corner

- **Application Examples: Variation Aware Timing**
  - Typical operating corner = Typical process corner
  - Worst-case corner = Slow process + full aging corner



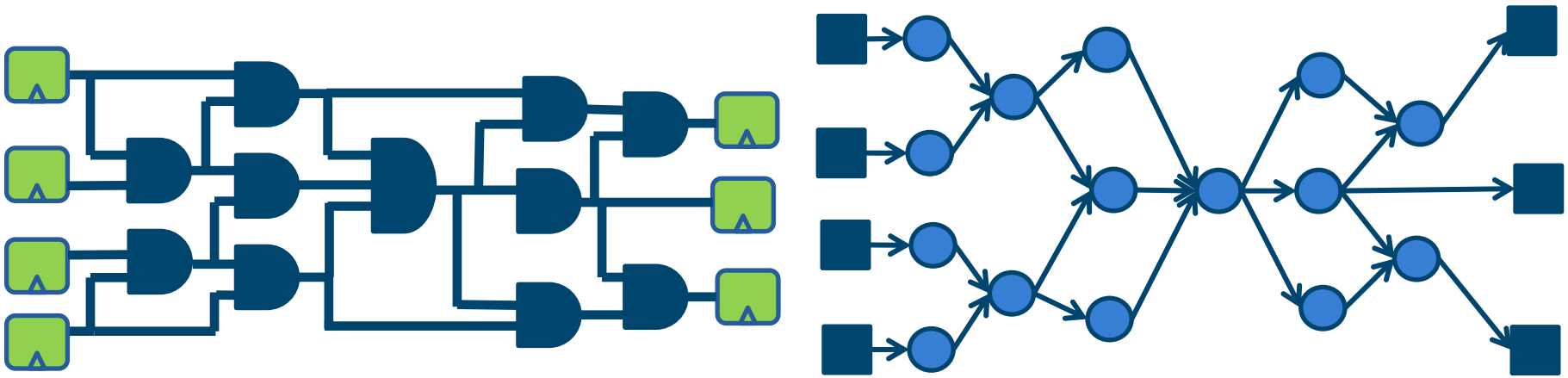
Opportunism Window

- **Opportunism window is defined the worst-case delay and typical operating clock period**
- **Metric to illustrate the benefit and cost of monitoring and monitor delay margin**



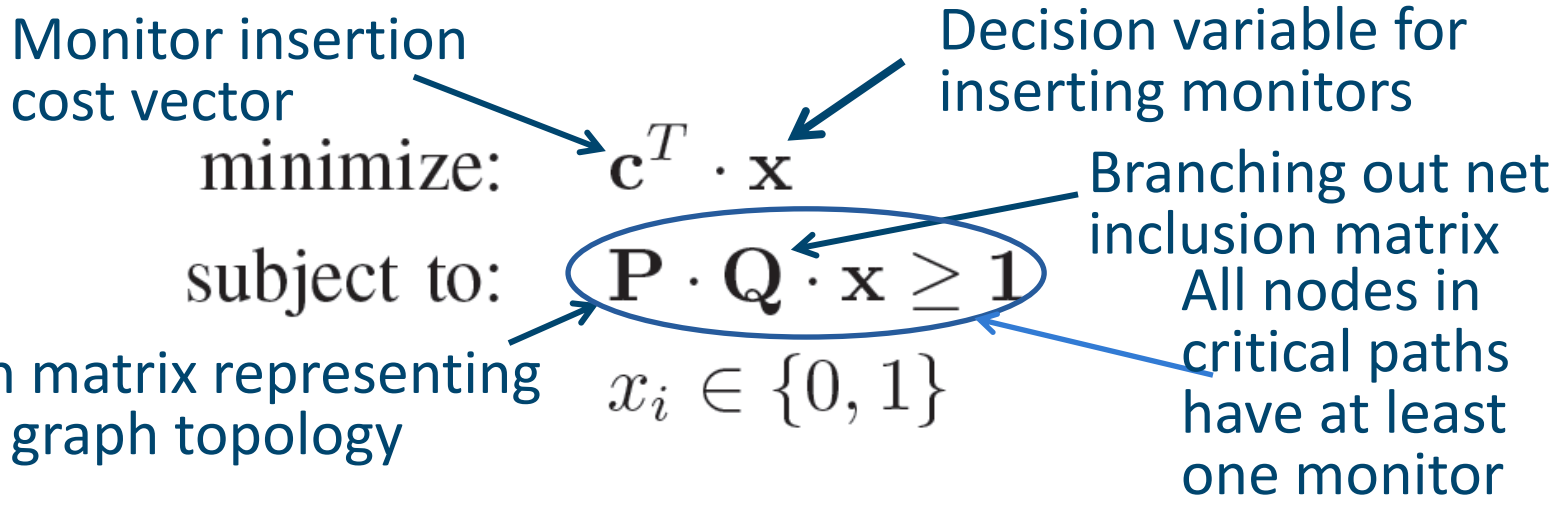
Critical Path Graph Construction

- Run STA with libraries of typical operating corner
- Set the typical operating clock period
- Run STA with libraries of worst-case operating corner
- Extract all cells/nets with negative slack
- Construct the critical path graph



Problem Formulation

- **Objective: minimize the monitor cost**
- **Constraint: All path are monitored**
- **The problem is first formulated as ILP as**



- **Entries in $\mathbf{P}$ and $\mathbf{Q}$ are binary numbers, given $x_i \in \{0, 1\}$, we can replace non-zero entry in $\mathbf{P} \cdot \mathbf{Q}$ with 1 and obtain $\mathbf{A}$**
- **The new constraint becomes $\mathbf{A} \cdot \mathbf{x} \geq 1$**

Problem Relaxation

- **Relax the constraint $x_i \in \{0, 1\}$ as $x_i \geq 0$**
- **The problem becomes an LP problem**

$$\begin{aligned} \text{minimize:} \quad & \mathbf{c}^T \cdot \mathbf{x} \\ \text{subject to:} \quad & \mathbf{A} \cdot \mathbf{x} \geq 1 \\ & x_i \geq 0 \end{aligned}$$

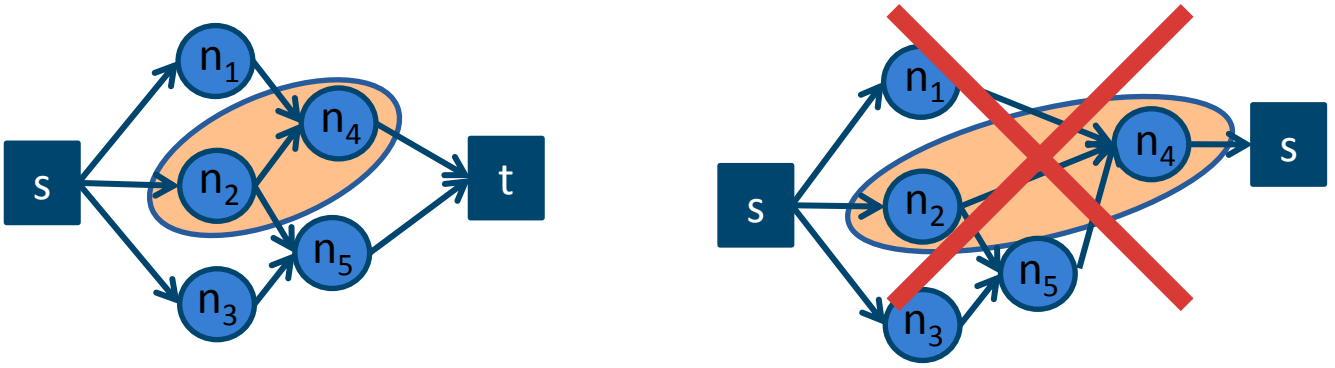
- **The dual of the LP problem is**

$$\begin{aligned} \text{maximize:} \quad & \mathbf{1}^T \cdot \mathbf{f} \\ \text{subject to:} \quad & \mathbf{A}^T \cdot \mathbf{f} \leq \mathbf{c} \\ & \mathbf{f} \geq 0 \end{aligned}$$

- **The new LP problem resembles st-max-flow problem**
- **Further conversion into an edge-based formulation**

Key Observation of the Problem

- We denote I_{n_i} as the set of nets that are monitored by monitor inserted at i-th node (i.e. n_i)
- If we lumped all nets in I_{n_i} as a super node
- Since the nets in I_{n_i} are determined by the timing margin, which is monotonic in topological order
- Lemma 1: There is no cyclic flow around the super node



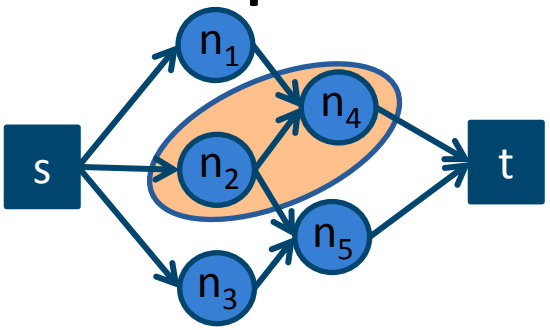
Path-based to Edge-based Conversion

- i-th row of $\mathbf{A}^T \cdot \mathbf{f}$ is the sum of all path flows that are monitored by the monitor inserted at the i-th node
- For example, the 4-th row is

$$f_{s->1->4->t}+f_{s->2->4->t}+f_{s->2->5->t}$$

$$\begin{aligned} \text{maximize:} \quad & 1^T \cdot \mathbf{f} \\ \text{subject to:} \quad & \mathbf{A}^T \cdot \mathbf{f} \leq \mathbf{c} \\ & \mathbf{f} \geq 0 \end{aligned}$$

- Since there is no cyclic flow around the super node
- Lemma 2: Paths enter the super node at most once
- Path flow sum equals the flow sum of the edges that goes into the super node

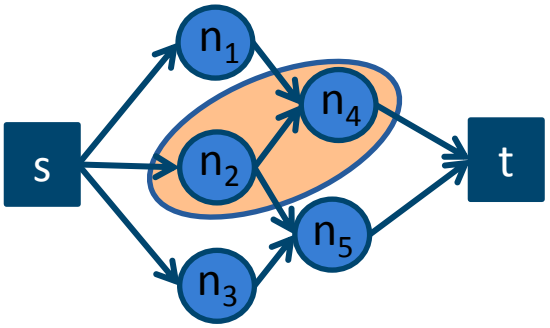


$$f_{s->1->4->t}+f_{s->2->4->t}+f_{s->2->5->t} = e_{1->4}+e_{s->2}$$

Path-based Edge-based

Extract Final Solution

- **Because of the relaxation on x_i, the result of the converted problem is just a lower bound of the formulated problem**
- **A valid solution can be obtained by extracting all nodes j that incoming edge flows into I_{n_j} equal the capacitance constraint**



n_4 is included in the solution if
$e_{1->4}+e_{s->2}= c_4$
May result in redundancy, e.g. both N_2 and N_4 can be included if
$f_{s->2->4->t}=1$, $C_2=1$, $C_4=1$

- **Some trimming is used to identify and delete the redundant nodes**
- **Experimental results show that in most of the cases, this method can achieve results that equal or very close to the lower bound values**

Outline

- Motivation and Overview
- Monitoring Costs and Constraints
- Path Selection and Monitor Location Selection
- Experimental Results
- Conclusion

Experiment Setup

- **Three commercial processor benchmarks**
- **Commercial sub-32nm process technology and libraries**
- **Path selection:**
  - Typical operating corner = TT library corner
  - Worst-case operating corner = SS library corner
- **Margin Calculation:**
  - Gate delay between TT and SS corner
- **Delay Margin: 5% of clock period at TT corner**

Processor	A	B	C
Gate count	10434	30296	58815
Register count	1191	3344	9516
Clock period at typical corner	1.01 ns	1.22 ns	1.43 ns
Clock period at slow corner	1.23 ns	1.48 ns	1.72 ns
Critical gate count	7246	21385	21630
Critical register count	852	2116	4195
Critical path ending point count [2]	1256	2637	4993

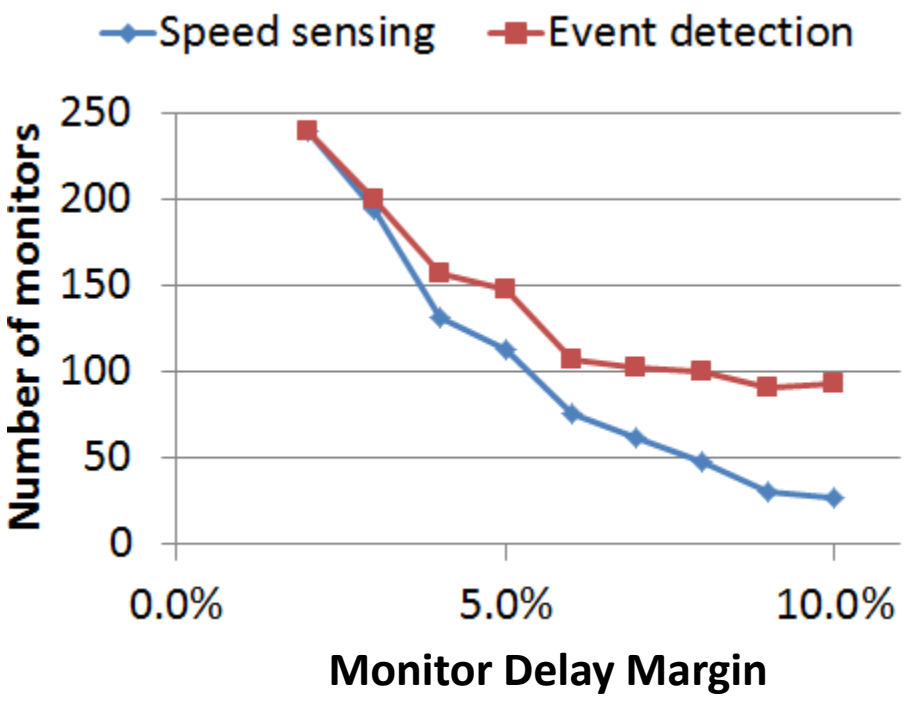
Experimental Results

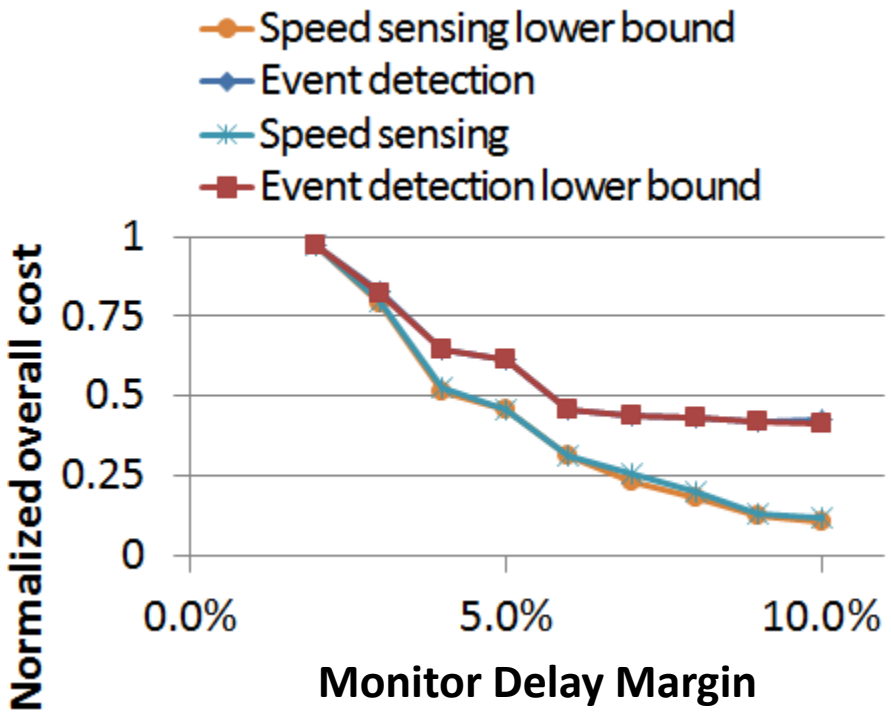
- **Baseline: Insert monitors at every path ending pins**
- **Event Detection:**
  - allow branching out nets within margin only at inverter/buffer
- **Speed Sensing:**
  - allow branching out nets within margin regardless of gate types

	Processor	A	B	C
Baseline	Monitor count	1256	2637	4993
	Normalized cost	12.16	8.95	14.05
SlackProbe Event Detection	Monitor count	148	480	510
	Normalized cost	1.34	1.69	1.57
	Normalized cost lower bound	1.34	1.59	1.53
SlackProbe Speed Sensing	Monitor count	113	311	374
	Normalized cost	1	1.07	1.08
	Normalized cost lower bound	1	1	1

Sweeping Delay Margin

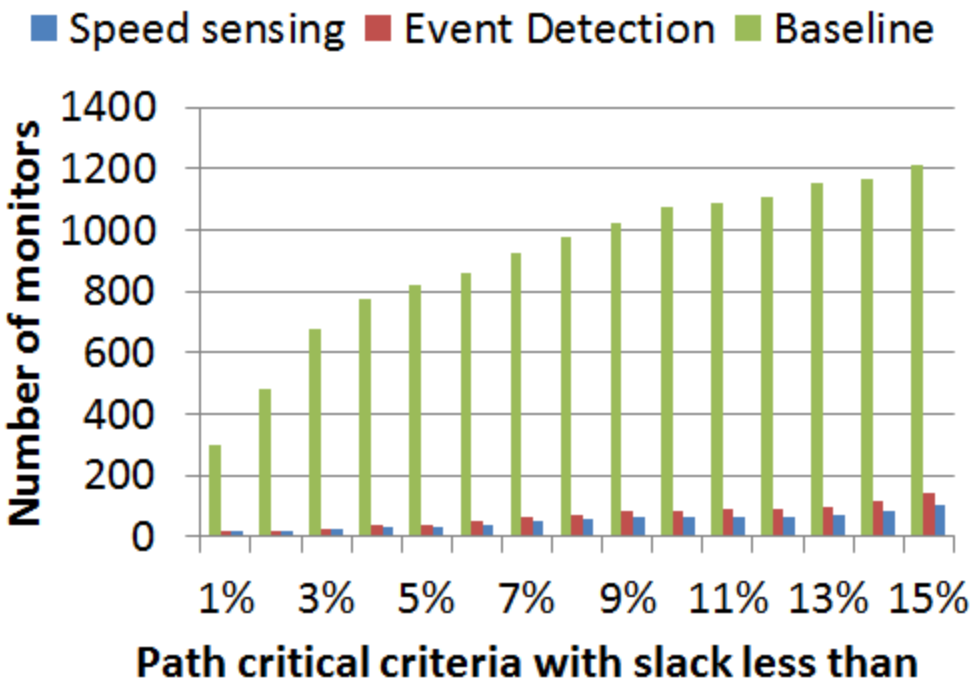
- **Sweeping the delay margin ε on processor A**
  - More benefit if we allow the monitors to be placed further away from destination registers
  - Results of our solution are very close to the lower bound values

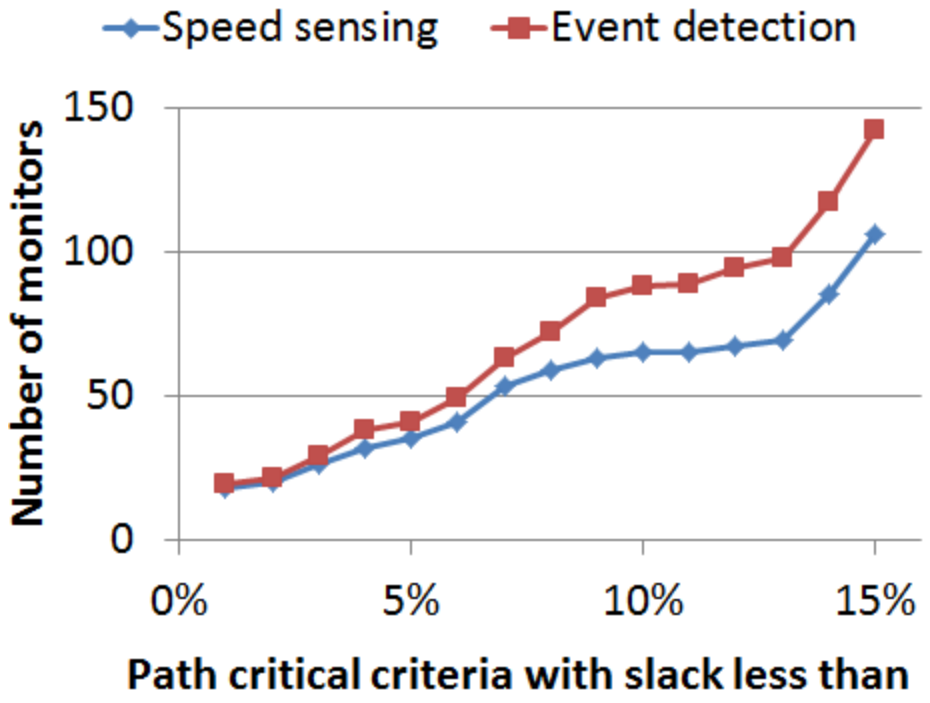




Sweeping Path Selection Criteria

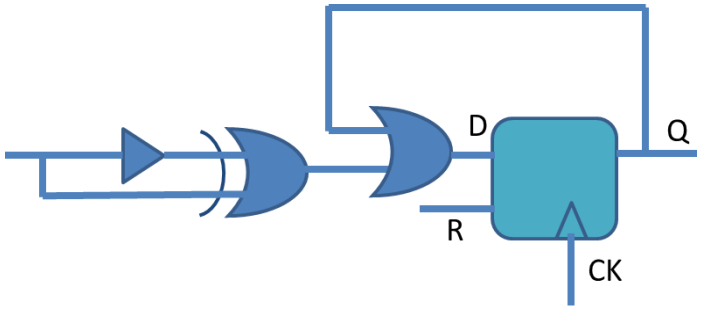
- **Path selection: Extract paths with slack less then specific amount at TT corner on processor A**
- **Average 15X reduction for event detection and 18X for speed sensing compared to baseline**

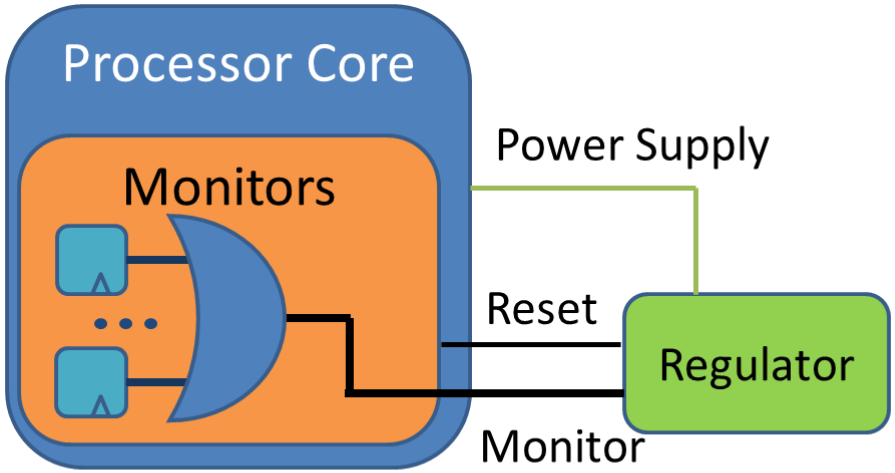


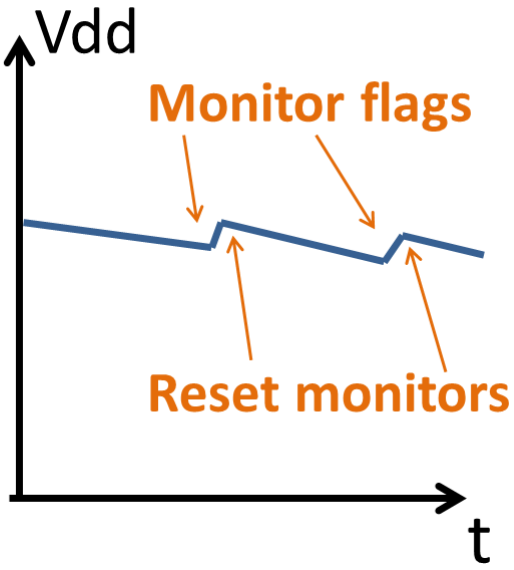


Adaptive Voltage Scaling Application

- **Monitor structure**
  - Use standard cells only(6 gates)
  - Sticky flag with external reset
- **Targeting application**
  - OR tree to connect monitor flags as one bit primary output
  - Global monitor reset as a primary input







Implementation Results

- **Full Implementation on processor A as ECO**
  - Including OR tree and global reset
- **Instances used for delay matching is small**
  - Minimum 6 gates are required for a monitor
- **Overhead is not tiny**
  - Better and more efficient monitor design can reduce it significantly

	Implementation I	Implementation II
Target delay margin	5%	8%
Number of monitors	113	48
Additional instances	711	327
Instances per monitor	6.3	6.8
Additional power overhead	13.65%	6.38%

Conclusion

- **We proposed SlackProbe, a low overhead in situ on-line timing slack monitoring methodology**
- **SlackProbe achieves an order of magnitude reduction in monitor number compared to the number of path ending pins**
- **Future work will incorporate the monitors in more applications and improve the monitor insertion to be less intrusive**

Q & A

- **Thanks!**