Testability Driven Statistical Path Selection

Jaeyong Chung, Jinjun Xiong*,

Vladimir Zolotov*, and Jacob A. Abraham

Computer Engineering Research Center, University of Texas at Austin

**IBM T.J. Watson Research Center, Yorktown Heights*

Delay Test

	Silicon Debug	Manufacturing Test
Objective	Find speed-limiting Vectors	Check if DUT can run at a desired freq.
Where it is done	Microprocessor design	ASIC design/Micro. design
Need to hit Fmax	Necessary	Good correlation is ok
Effects of interest	Unmodelled effects, model/silicon mismatch (e.g., MIS, crosstalk, power droop)	Effects that differentiate each chips (e.g., process variation, delay defects)
Important step	Test vector generation	Path selection
Test type	Transition tests may work better than path tests	Path delay tests are appropriate

SSTA vs. STA in path selection

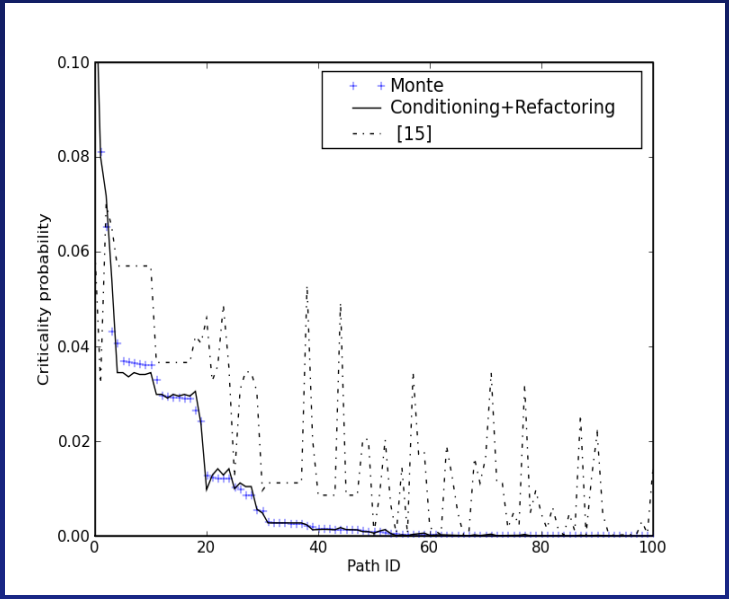
- SSTA can compute the fault probability for each path explicitly
  - Even If nominal delays are the same, the sensitivity to each source of variations can be different
- STA does not know correlations
  - may keep selecting paths in a chain of gates or in a small region
  - may select paths only sensitive to M1, etc

Path selection using SSTA

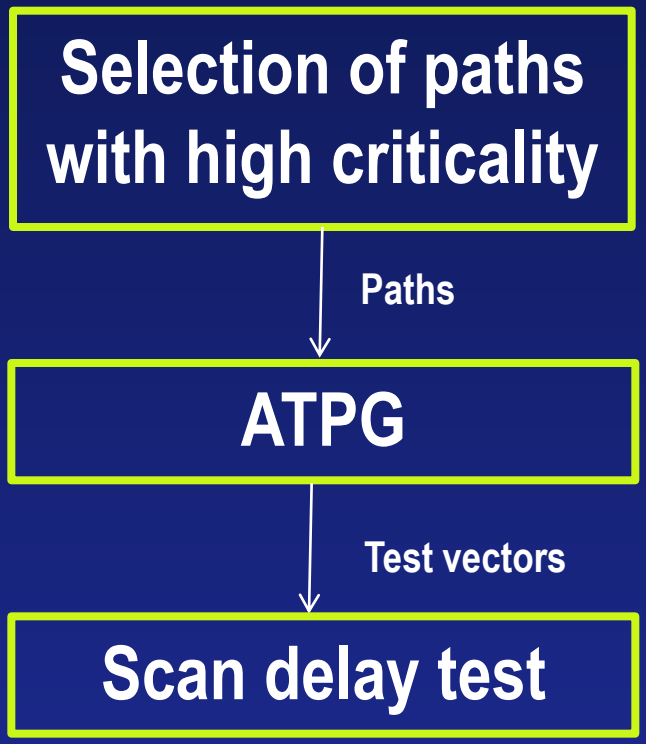
- Covering-based algorithms
  - The *process space coverage metric* (PCM) is used as the test quality metric
  - The objective is to maximize PCM when the number of paths to be selected is limited
  - The maximum coverage problem
  - Good paths for a given test clock period
- Criticality-based algorithms
  - Select paths with high criticality
  - Good paths irrespective of the test clock period

Problem Formulation

- Path criticality (e.g., the probability that the path becomes critical) takes the fault probability and the correlations into account

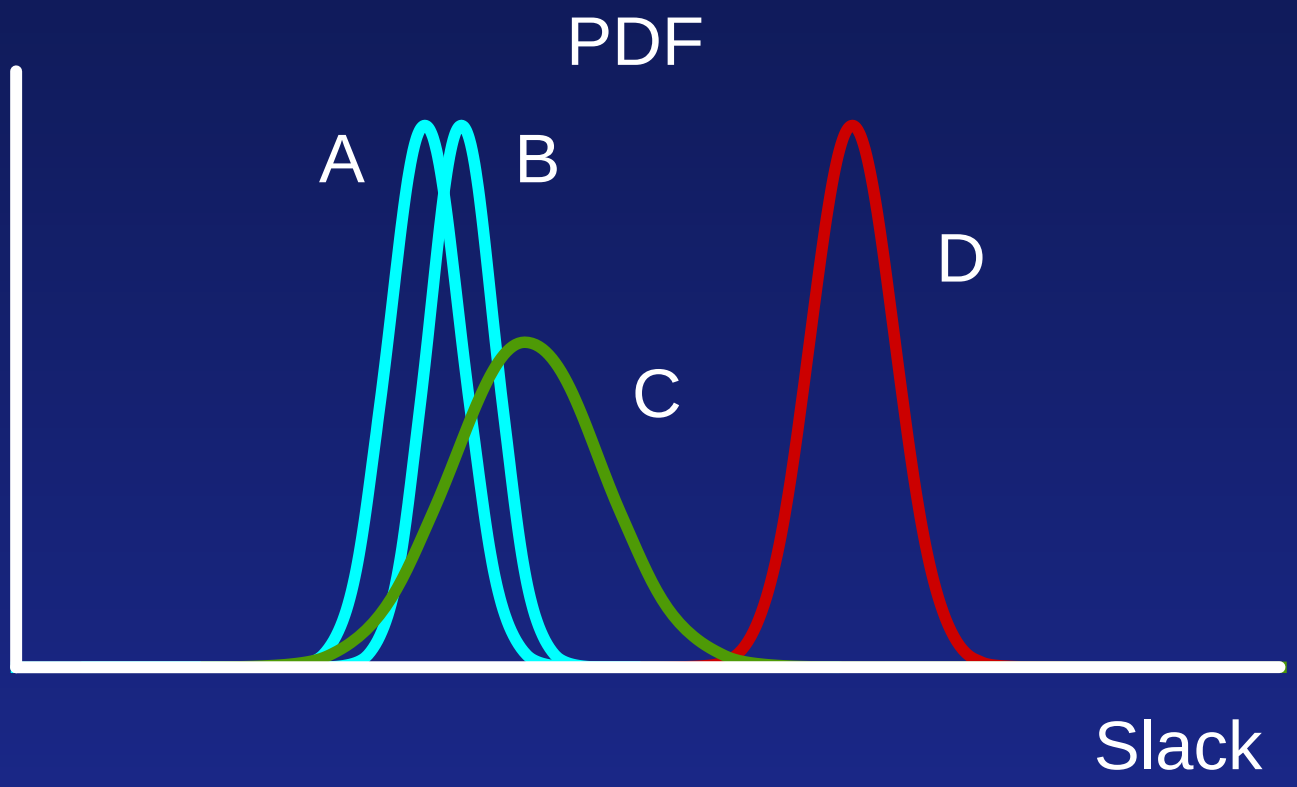


[J. Chung et al ASPDAC 2011]

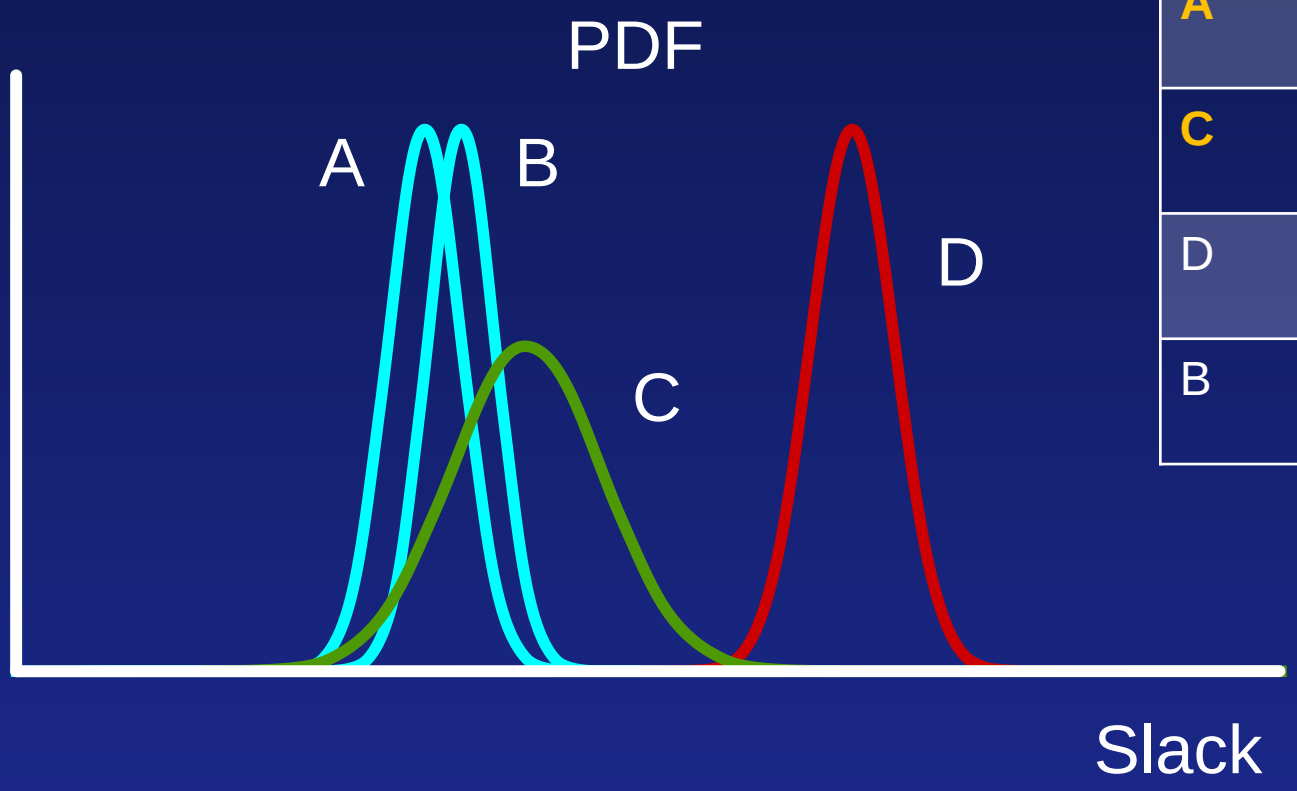


☹ Coverage loss due to untestable paths

An example



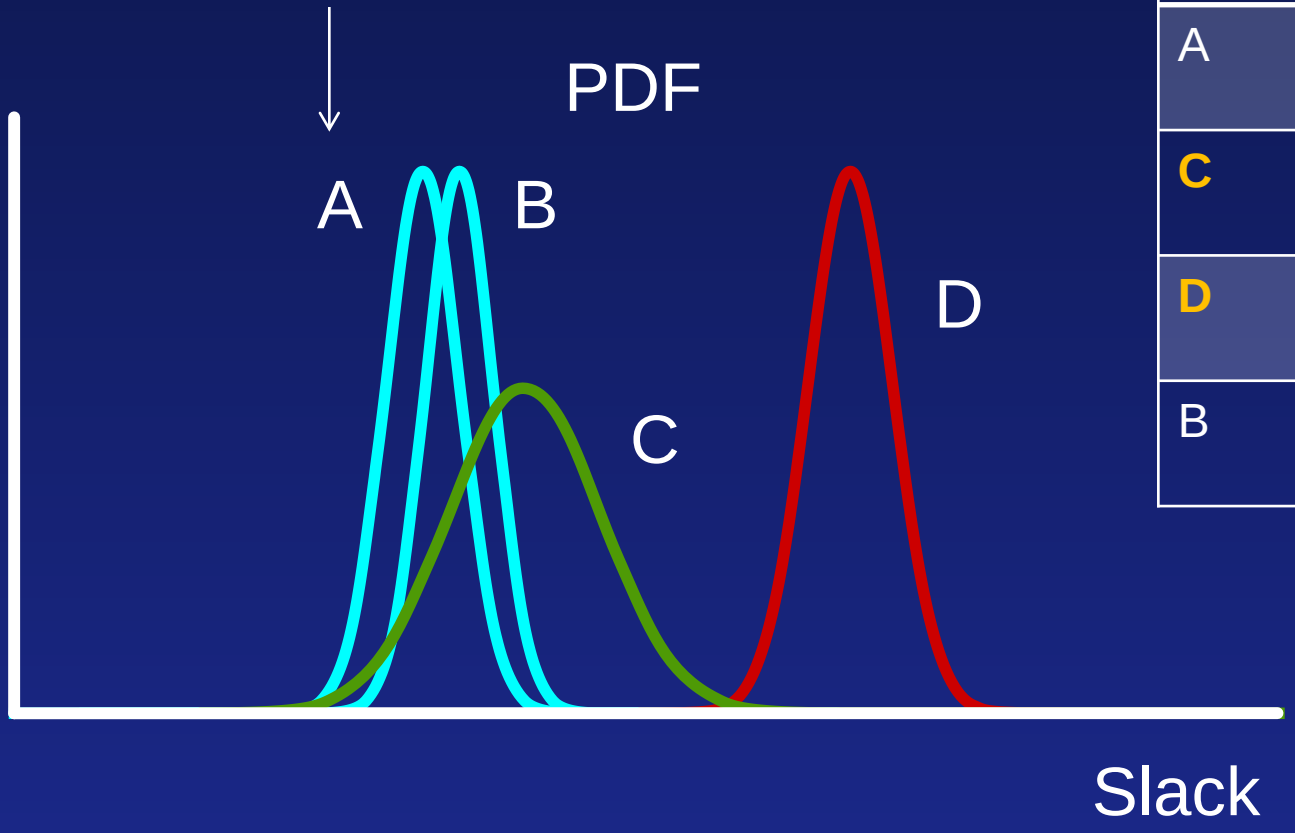
An example



SSTA (criticality)	STA
A	A
C	B
D	C
B	D

An example

Untestable path

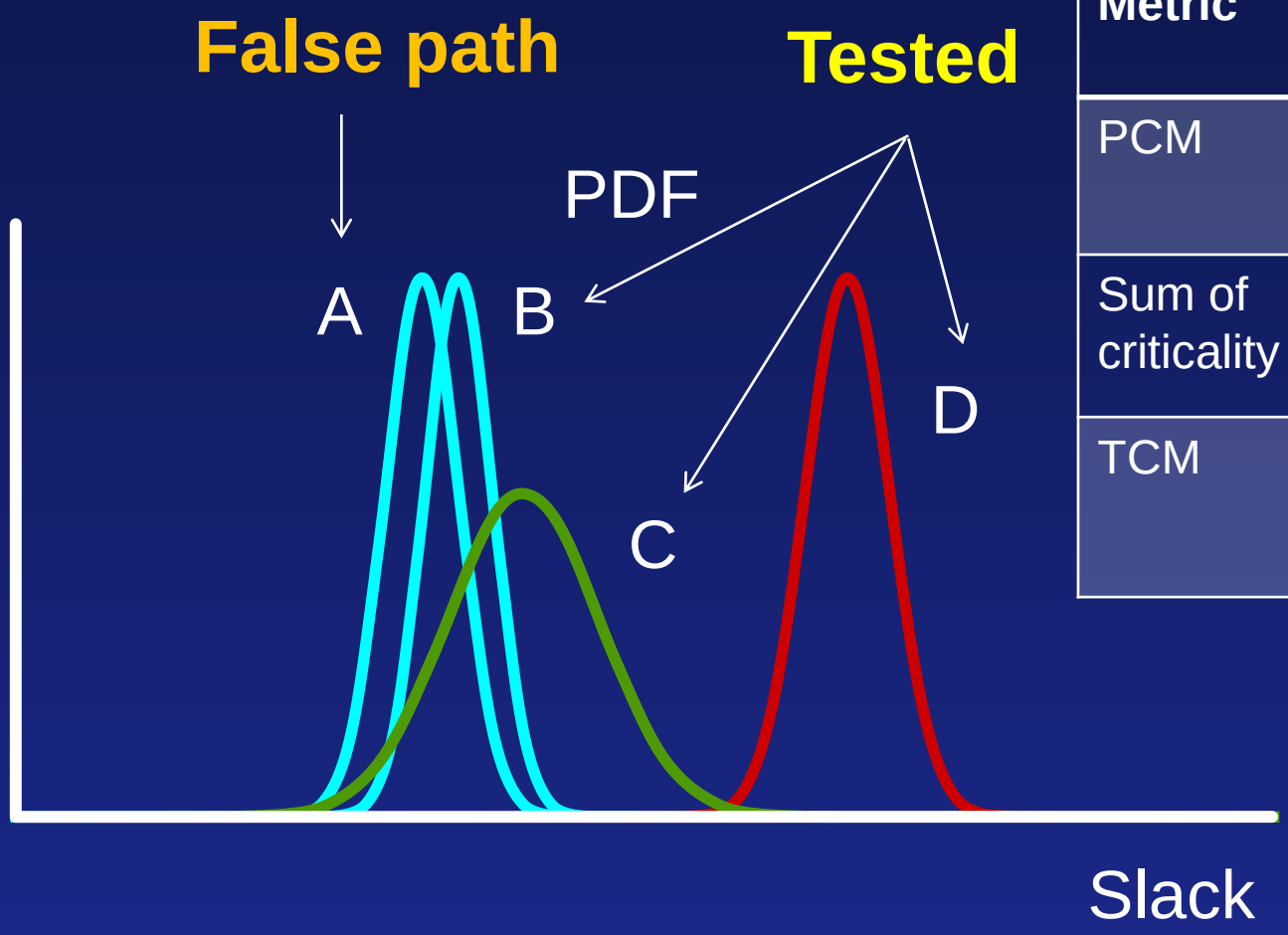


SSTA (criticality)	STA
A	A
C	**B**
D	**C**
B	D

Proposed Approach

- Consider the testability in the first place
  - Inspect only testable paths by integrating a SAT solver
- Define the *testable path coverage metric* (TCM)
  - The probability that a given path set contains the least slack testable path
- Our objective is to maximize TCM

TCM vs. other metrics



Metric	Dep. Clock	Coverage
PCM	Yes	Not 100%
Sum of criticality	No	Not 100%
TCM	No	100%

Maximizing TCM

- Let λ_{Ω} be the probability that a given path is critical among all paths in the circuit (the conventional path criticality)
- Let λ_{Ω_t} be the probability that a given path is critical among all testable paths in the circuit (criticality among testable paths)
- Due to the mutually exclusive nature of the critical events,
  - $TCM(\Pi) = \sum_{p \in \Pi} \lambda_{\Omega_t}(p)$

Maximizing TCM

- For $U \subset \Omega_t$, $\lambda_{\Omega_t}(p) \leq \lambda_U(p)$
- Our algorithm is based on the branch and bound framework proposed in [ICCAD 08, Vladimir et al]
  - Inspect paths in the depth-first manner
  - Maintain *k* best paths $\prod$
  - Whenever a new path is found,
    - Replace it with one path in $\prod$
    - Or just throw it away
  - Greedy Algorithm

An example

new path

	A	**B**	**C**	**D**
$\lambda_{\{A,B,C,D\}}$	**0.1**	**0.4**	**0.3**	**0.2**
λ_{Ω_t}	**0.08**	**0.3**	**0.1**	**0.1**

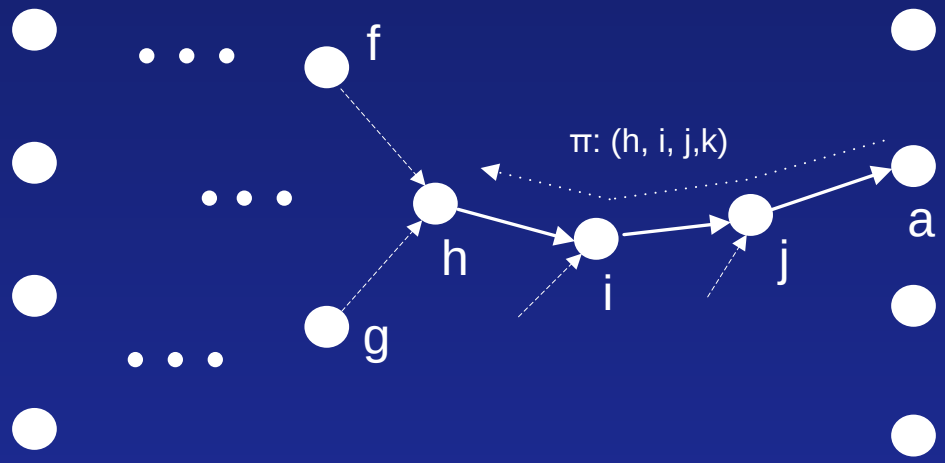
D	**B**	**C**

Maximizing TCM

- Criticality metric allows us the following modification
  - Whenever *m* new paths are accumulated,
    - Consider the replacement
    - Compute criticality among $\prod$ and the *m* new paths
    - Discard the worst *m* paths
- Lazy update so enhance the runtime
- Less greedy manner (i.e., more global view) so improve the QoR

Pruning

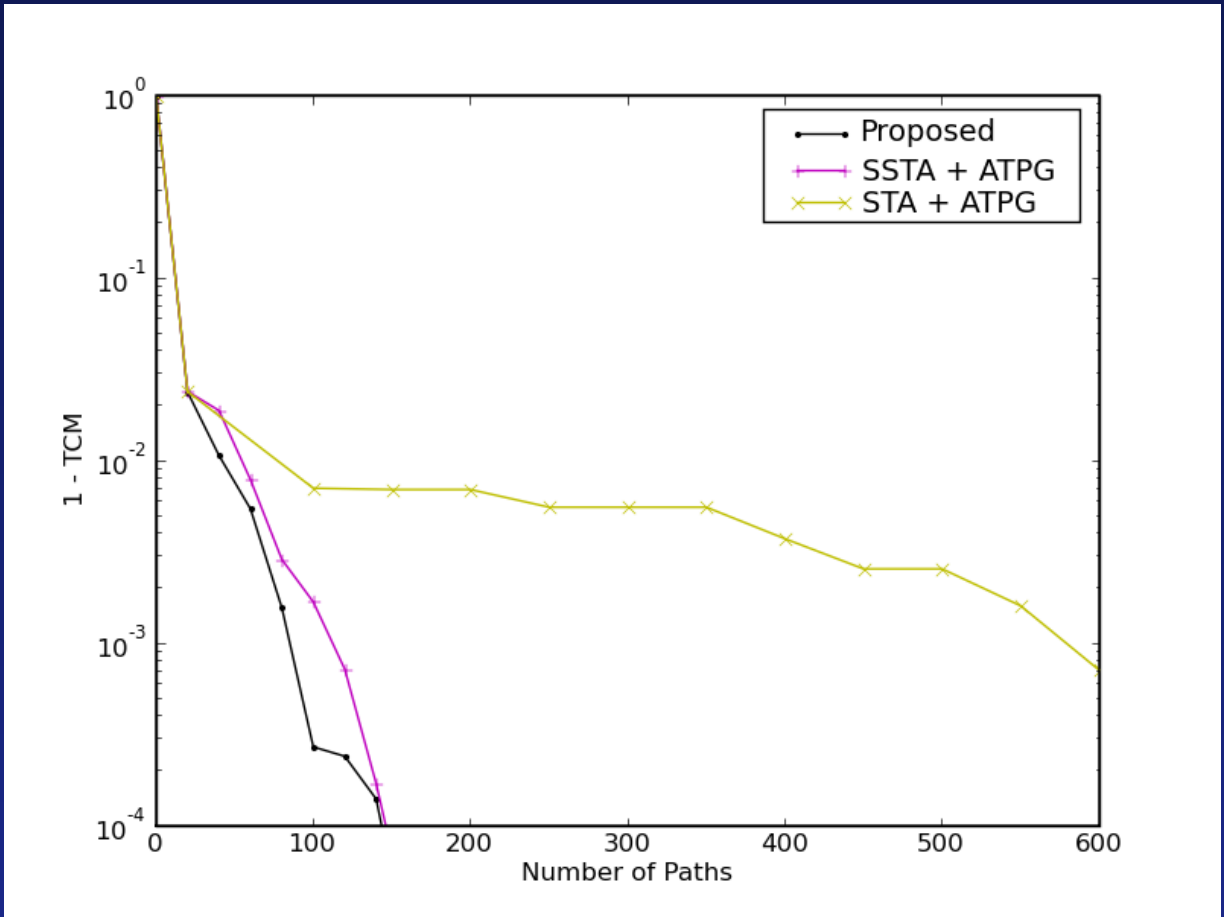
- To make the path selection using depth-first search practical
  - If a subpath is not testable, all paths going through the subpath are not testable
  - If all paths going through a branch are worse than the worst path in $\prod$, we can prune the branch



Experimental Setup

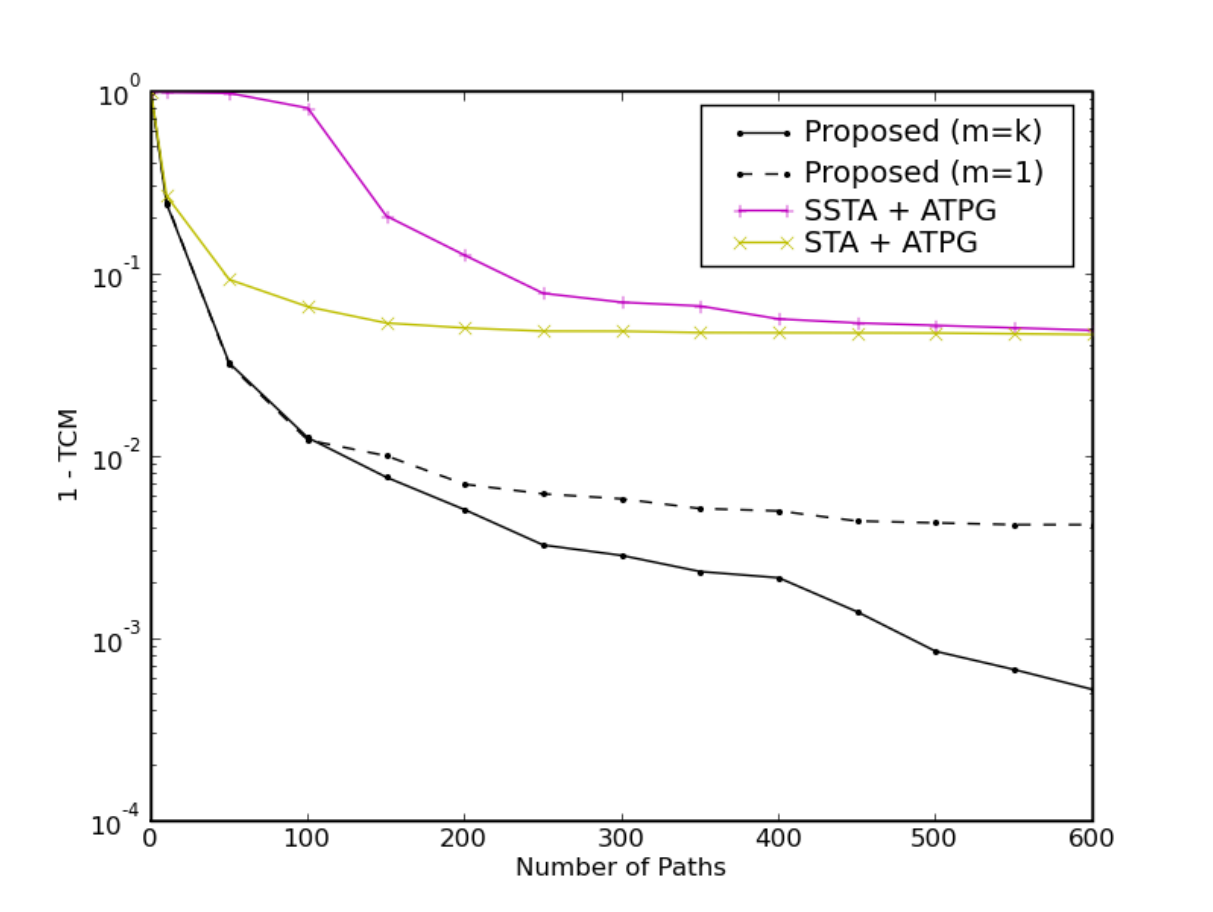
- SSTA algorithm used: [Visweswariah DAC 2004]
- Spatial correlation model: A quad tree with 3 levels
  - 4%, 5%, and 6% variation at 1st, 2nd, and 3rd level
  - 21 global sources of variation
- 5% random independent variation
- *Minisat* is integrated
- Proposed method is compared with:
  - STA+ATPG
  - SSTA+ATPG

Experimental Results



eth_45 (82.4%)

Experimental Results



tv80_45 (0.03%)

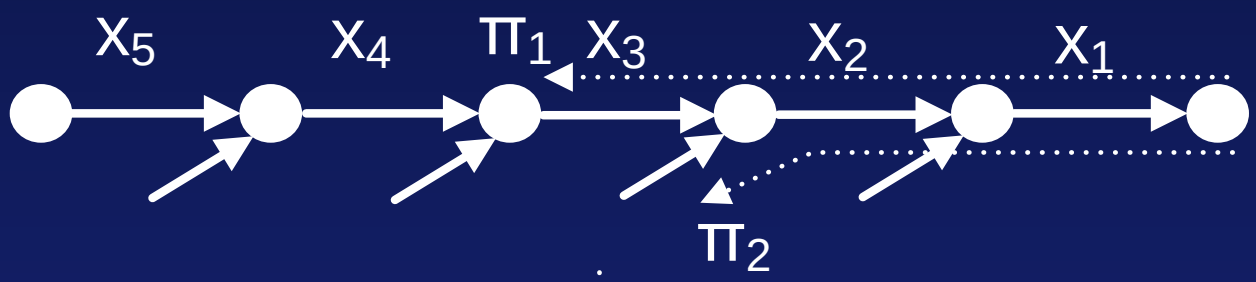
Experimental Results

Circuit	SSTA+ATPG		Proposed	
	TCM	CPU(s)	TCM	CPU(s)
C17	100%	0.000	100%	0.000
C432	37%	0.082	78%	0.019
C499	88%	0.030	90%	0.079
C880	91%	0.010	90%	0.005
C1355	86%	0.028	87%	0.058
C1908	0%	464.9	86%	0.238
C2670	87%	0.029	87%	0.030
C3540	0%	0.747	100%	0.134
C5315	100%	0.061	100%	0.056
C6288	0.6%	18369.2	99%	50.8
C7552	100%	0.507	100%	0.063
Avg.	62%	1712.3	92%	4.68

Conclusions

- Proposed method achieves 47% better QoR in TCM and up to 361X speedup compared to SSTA+ATPG
- Our method actually involves the test generation
- The remaining work is to load the test patterns on a tester
  - This allows us to demonstrate the benefits of statistical methodology on a silicon
    - Short turn-around time
    - Apples-to-apples comparison

Novel way of SAT Integration



1. Assuming x_1=1, check if satisfiable

2. Assuming x_1=1 and x_2=1, check if satisfiable

3. Assuming x_1=1, x_2=1 and x_3=1, check if satisfiable

→ the solver returns UNSAT

Conventional flow

1. Assuming x_1=1, x_2=1, x_3=1, x_4=1 and x_5=1, check if satisfiable

→ the solver returns UNSAT with conflict assumptions which can be x_1=1, x_3=1

Proposed flow