Detecting Emerging Wearout Faults

Jared C. Smolens, Brian T. Gold
James C. Hoe, Babak Falsafi, Ken Mai

TRUSS
Computer Architecture Lab
Carnegie Mellon
http://www.ece.cmu.edu/~truss

© 2007 Jared Smolens

Motivation: wearout faults

- Device and wire wearout accelerates with scaling
  - E.g., gate oxide breakdown, negative-bias temperature instability, hot carrier injection, electromigration
- Devices slow gradually before failure
  - Soft breakdown—logic functions unaffected
  - Initially affects **small, unpredictable** subset of devices

Opportunity for in-field early wearout fault detection

© 2007 Jared Smolens

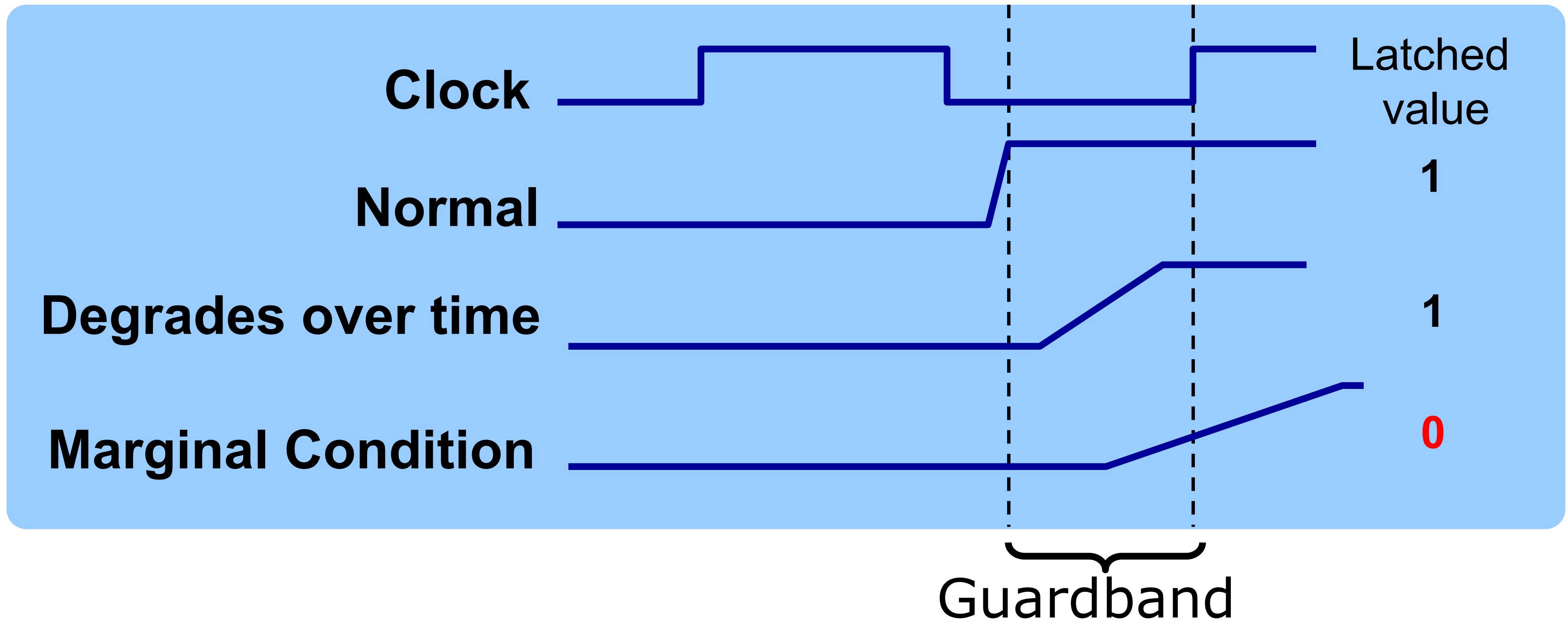
Problem: detecting faults early

- Faults initially hidden by guardband
  - Conservative design absorbs slowdown from soft breakdown
  - ➔ How to expose faults before they affect normal operation?
- Faults may be architecturally masked
  - At onset, few faults will be visible architecturally
  - Sources include programmatic and logical masking
  - ➔ How to detect faults when execution is still correct?

Want to observe faults before they affect general execution

© 2007 Jared Smolens

Solution for guardband



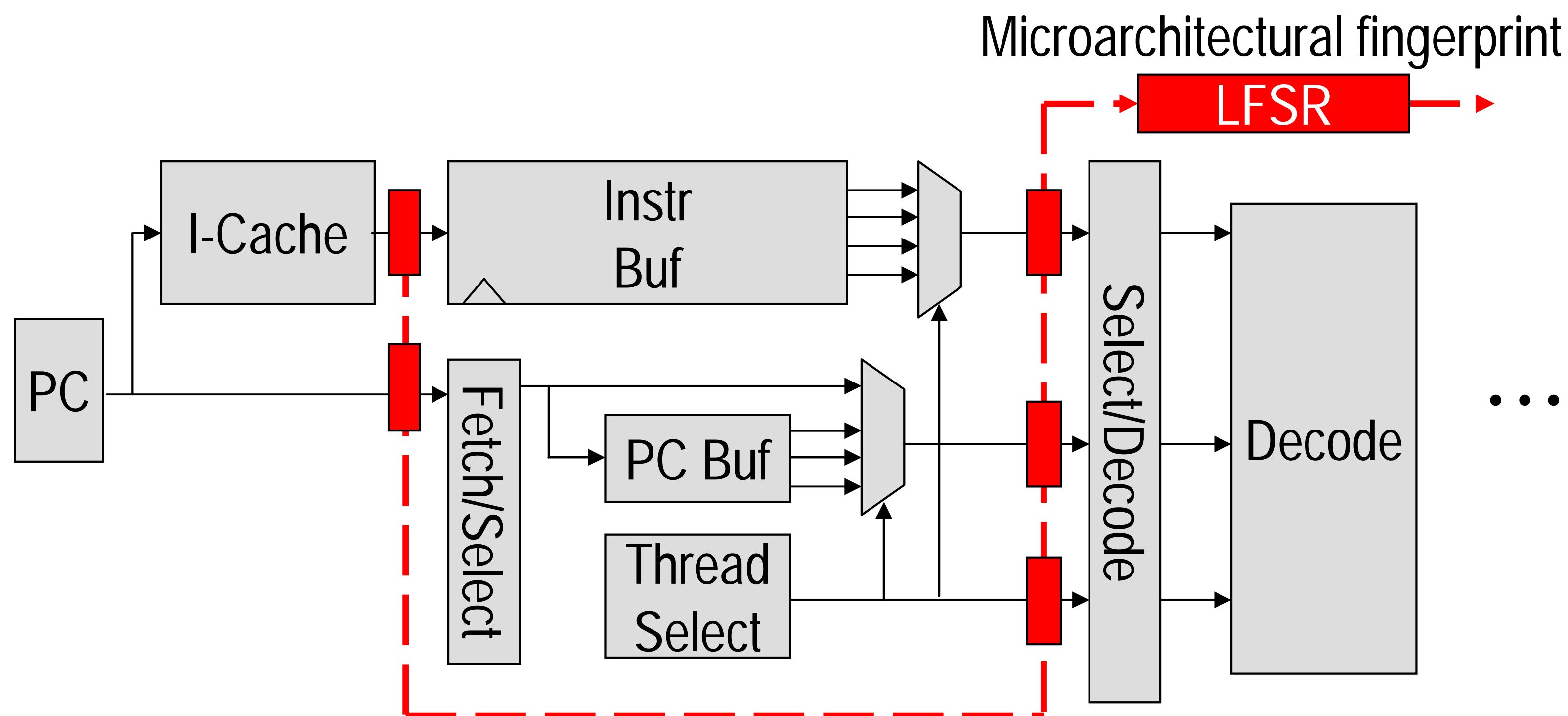
- Test at marginal conditions, gradually cut guardband
  - Combinational paths eventually miss timing

Wearout-induced slowdown causes missed timing under more conservative conditions

© 2007 Jared Smolens

Solution for masking

- Observe *internal* microarchitectural state updates
  - Leverage DFT hardware to capture internal nodes
  - Bring observation close to the actual faults
  - At-speed signature w/"microarchitectural fingerprints"



© 2007 Jared Smolens

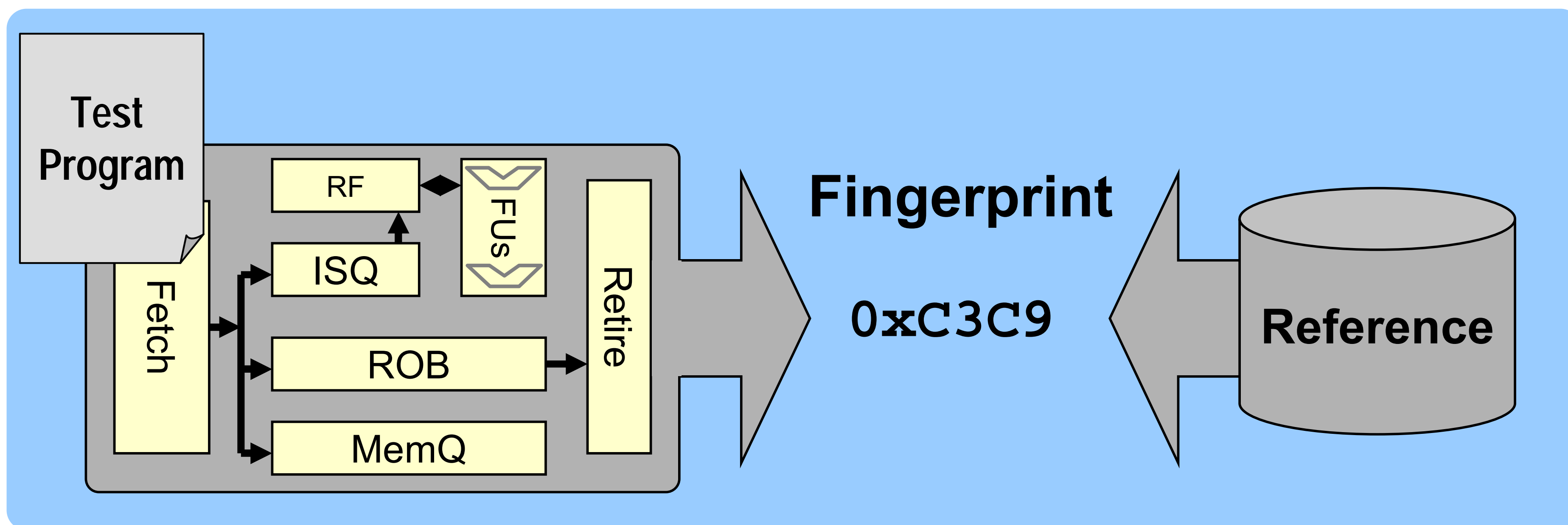
Outline

- Introduction
- **FIRST Methodology**
- Microarchitectural Fingerprints
- Wearout Fault Model
- Evaluation
- Conclusion

© 2007 Jared Smolens

Fingerprinting in Reliability & Self Test (1)

- Periodically enter FIRST test mode (e.g., once daily)
  - Initialize processor and load fault tests
  - Continuously monitor and summarize internal state
  - Compare w/reference (e.g., RTL or unstressed core)



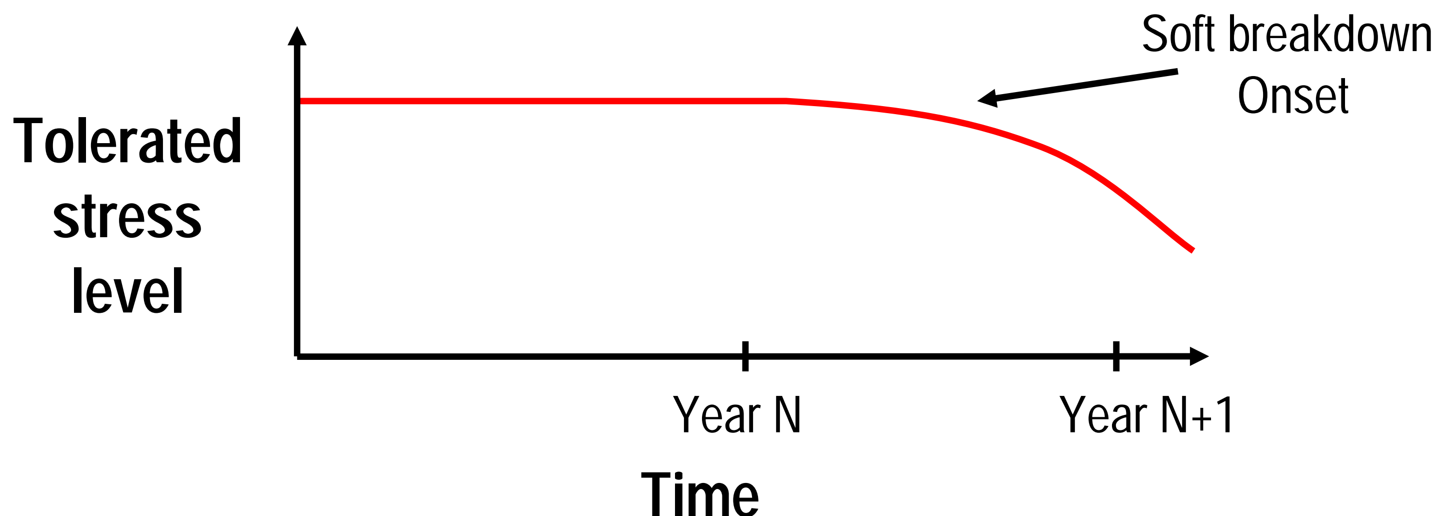
Signature comparison exposes faults

© 2007 Jared Smolens

Fingerprinting in Reliability & Self Test (2)

- Bring processor closer to marginal environmental conditions
  - E.g., V_{min}, F_{max} using existing knobs for power/thermal
  - Gradually decrease margin, repeat until signature mismatch

- Failures at more conservative conditions indicate wearout
  - Compare with failure results of prior tests
  - Long-term analysis may be done in software



Changes in signature failure conditions indicate wearout

© 2007 Jared Smolens

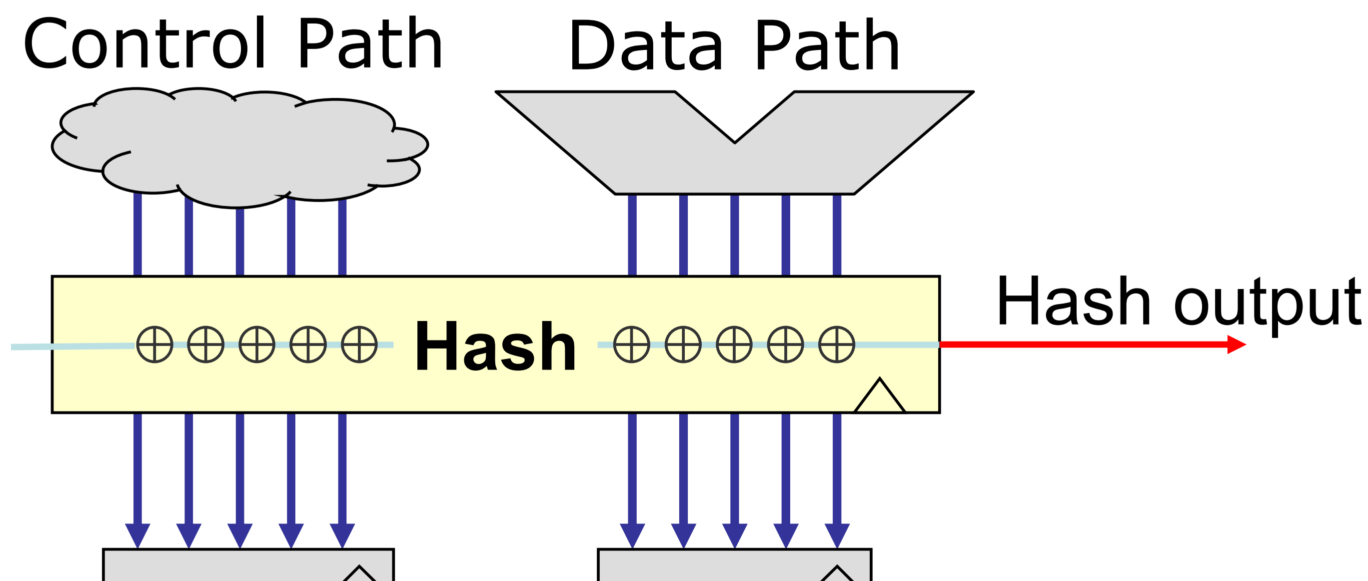
Outline

- Introduction
- FIRST Methodology
- **Microarchitectural Fingerprints**
- Wearout Fault Model
- Evaluation
- Conclusion

© 2007 Jared Smolens

Microarchitectural fingerprints

- Observe-only summary of internal processor state
  - Ex: writeback, store retirement, control logic
- Clock-for-clock correspondence w/reference

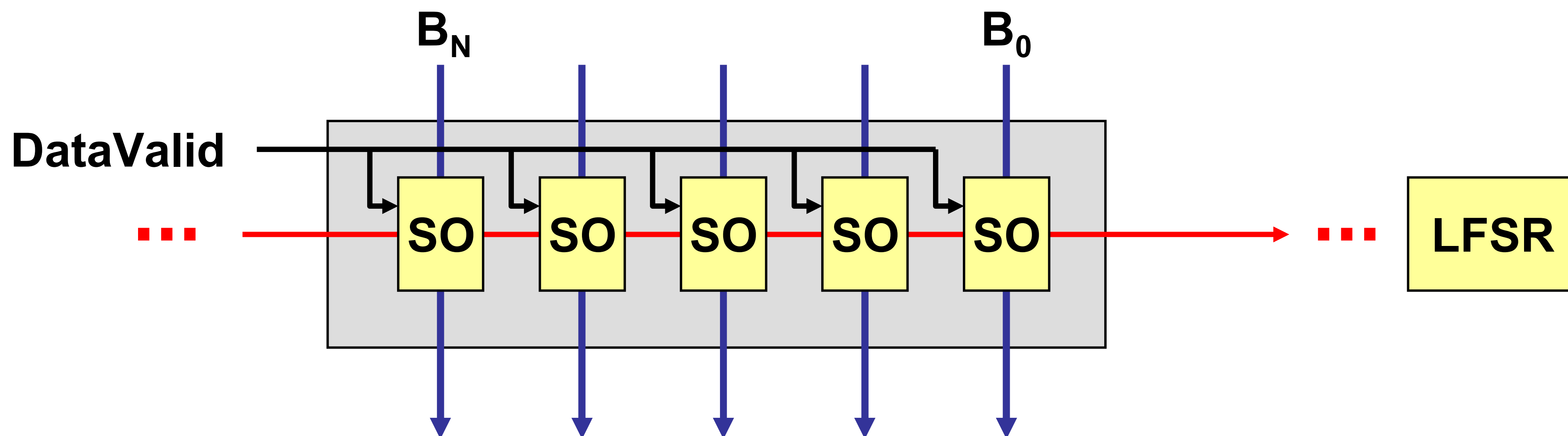


Necessary hardware <u>already available</u> in "scanout chains" for DFT support

© 2007 Jared Smolens

Collecting uArch fingerprints

- Selectively sample observe-only scanout nodes
  - Per-cell signature disable by using existing "Load" signal
- Compress within each module
  - Maintain separate module signatures
  - Compare periodically with reference



Use scanout as an active signature collector

© 2007 Jared Smolens

Outline

- Introduction
- FIRST Methodology
- Microarchitectural Fingerprints
- **Wearout Fault Model**
- Evaluation
- Conclusion

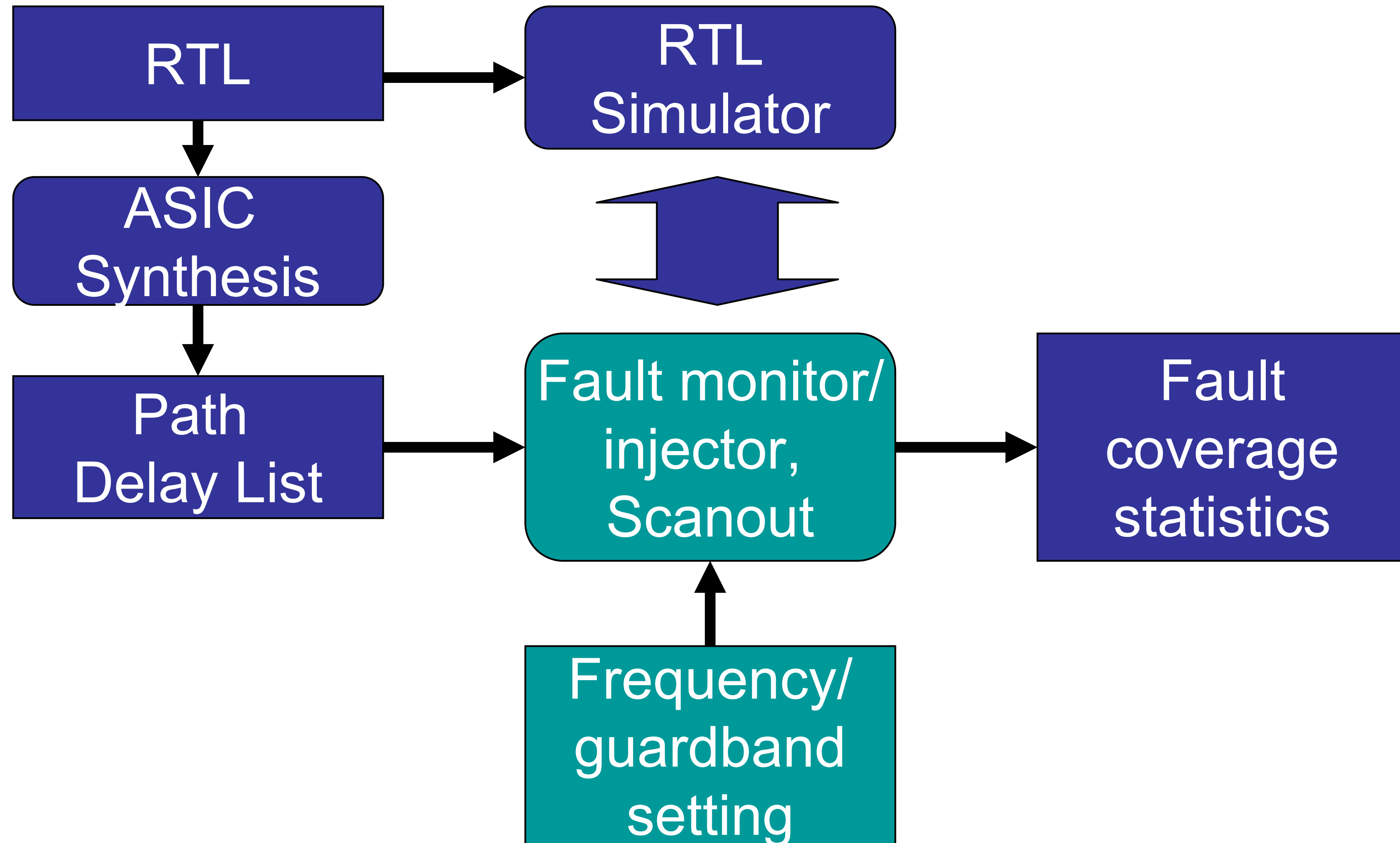
© 2007 Jared Smolens

Wearout Fault Modeling

- Motivation: Evaluate fault detection mechanisms
  - Detect when core **begins** to miss timing
- Problem: gate-level simulation slow
  - More detail than needed
- Solution: Simulate RTL with timing annotations
  - Balances speed and accuracy

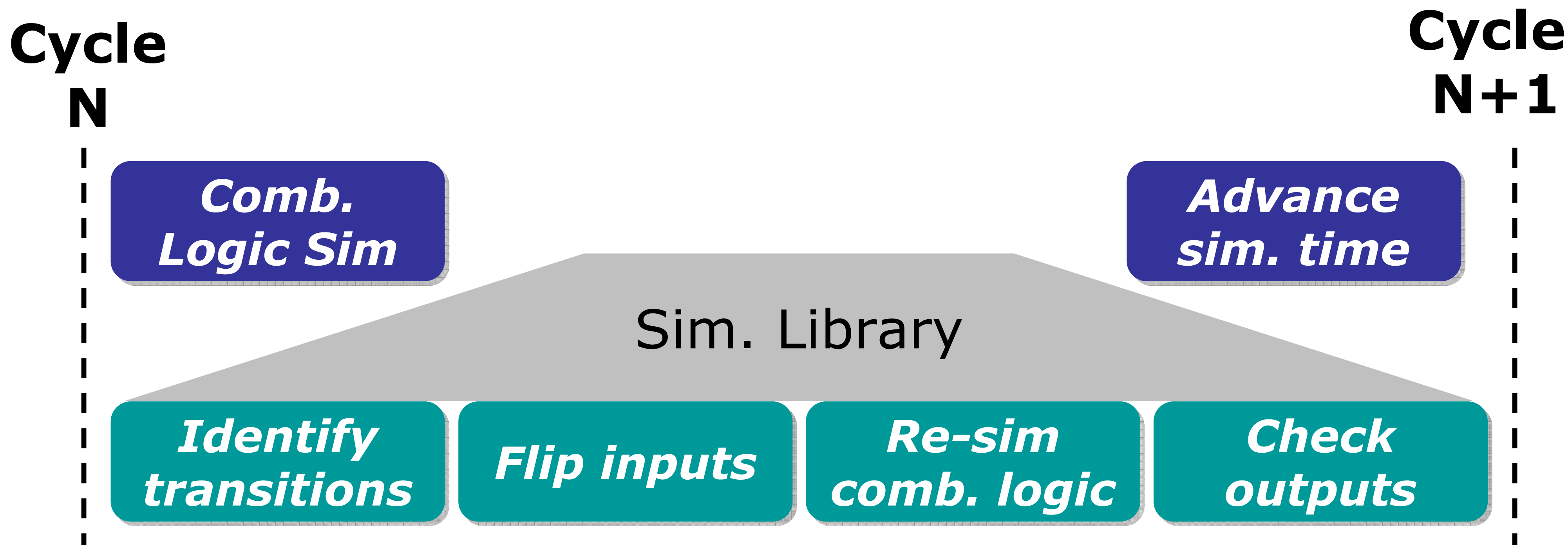
© 2007 Jared Smolens

Fault Model Tool Flow



© 2007 Jared Smolens

Fault Simulation Timeline



- On each cycle: look for missed timing on paths
  - Force previous cycle's input on missed paths
  - Check for changes at outputs (flip-flop inputs)

Evaluates and propagates non-masked timing faults

© 2007 Jared Smolens

Outline

- Introduction
- FIRST Methodology
- Microarchitectural Fingerprints
- Wearout Fault Model
- **Evaluation**
- Conclusion

© 2007 Jared Smolens

FIRST Injection Methodology

- Synthesized OpenSPARC T1 thread select unit
  - .18μm TSMC low-power process
  - Max true path 951ps
  - 186 flops, 6,292 paths
- Simulation parameters
  - Clock period varied between 900-955ps
  - 10K uniform random input vectors
  - 16-bit microarchitectural fingerprint LFSR

© 2007 Jared Smolens

FIRST Injection Results

Period (ps)	Possible Paths	Activated Paths	Propagated Paths
900	207	28	9
910	137	20	6
920	73	12	4
930	33	3	0
940	14	2	0
950	4	1	0
955	0	0	0

- “False” paths not propagated to outputs
- Microarchitectural fingerprints indicated signature mismatch on **any** propagated fault

© 2007 Jared Smolens

Conclusion

- Wearout faults shorten processor lifetimes
  - Precise failure times and locations not predictable
- Early detection opportunity
  - Want to detect wearout before affecting normal execution
  - Hampered by guardbands and fault masking
- FIRST methodology
  - Periodically inspect internal nodes for signs of wearout
  - Identify changing marginal conditions, expose faults hidden by guardbands

© 2007 Jared Smolens

For more information

Visit the TRUSS website:

http://www.ece.cmu.edu/~truss

© 2007 Jared Smolens