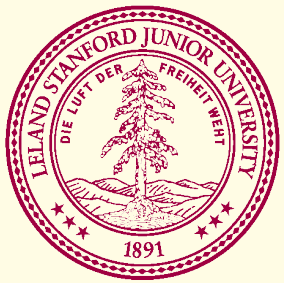
Rethinking Error Injection for Effective Resilience

Shahrzad Mirkhani[1] Hyungmin Cho[2] Subhasish Mitra[2] Jacob Abraham[1]

[1]University of Texas at Austin

[2]Stanford University

ASP-DAC, 2014



Motivation

Motivation

- Soft error analysis is an involved process
- Single bit-flip on flip-flops are accurate but slow
- High-level models are fast but inaccurate
- Error propagation analysis for a massive error injection is done
- Need to rethink conventional approaches to error injection

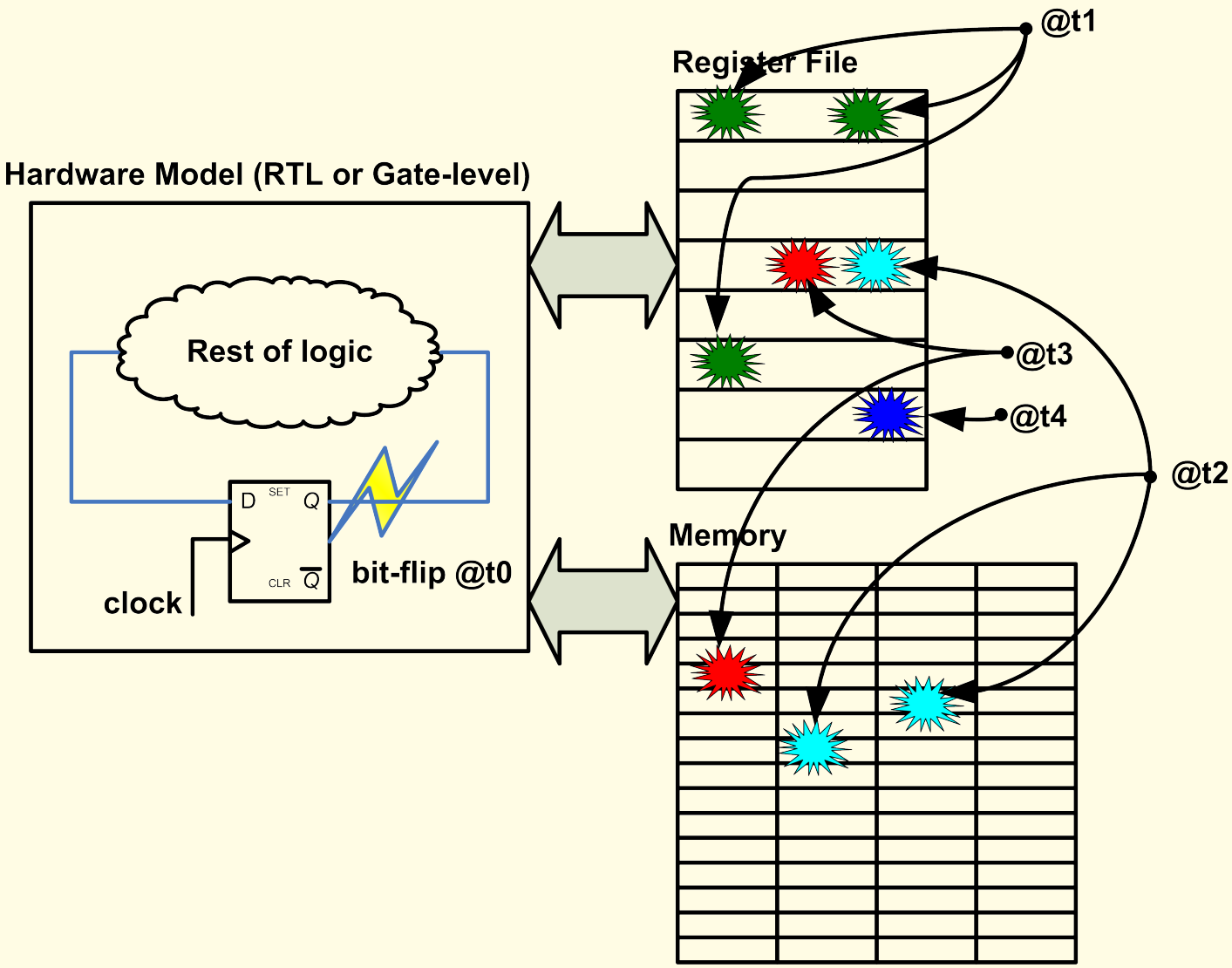
Hardware vs. Software-based Error Model

- Hardware-based (HW-based) error model
  - Single bit-flip on flip-flop
  - A close model to radiation-based error
- Software-based (SW-based) error model
  - Injection on memory bits/register bits
  - Single bit-flips
  - Usually follow uniform distributions
  - Some models inject error on used memory elements/registers

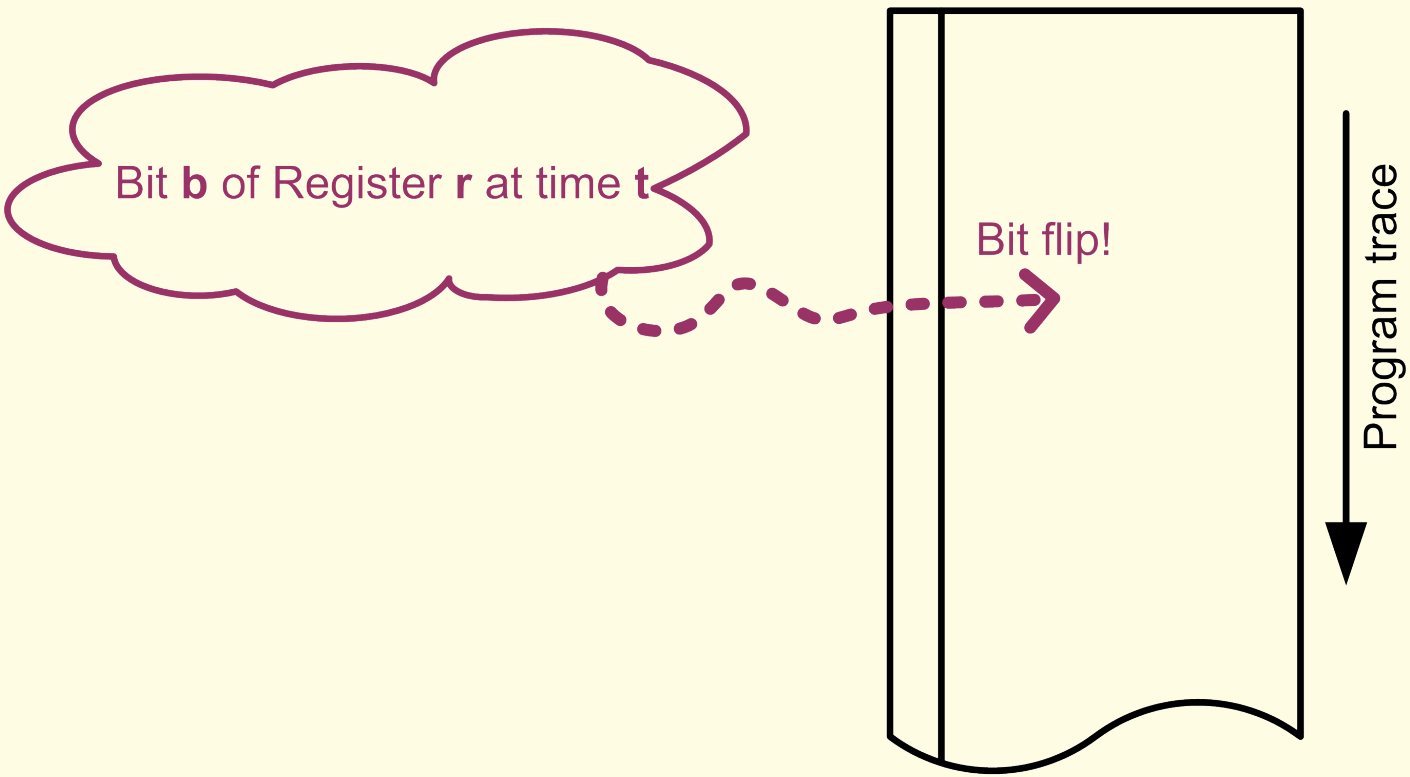
Error Propagation to SW Visible Components

- How to compare SW- and HW-based errors?
- A SW-based error can be equal to a HW-based error
  - Models **ALL** propagations of a HW-based error to SW visible components
  - SW visible components consist of registers and memory elements
- A HW-based error might propagate to more than one SW-visible component
- Most SW-based error models inject **single** bit-flip on a SW visible component

Error Propagation to Software Visible Components - cont'd

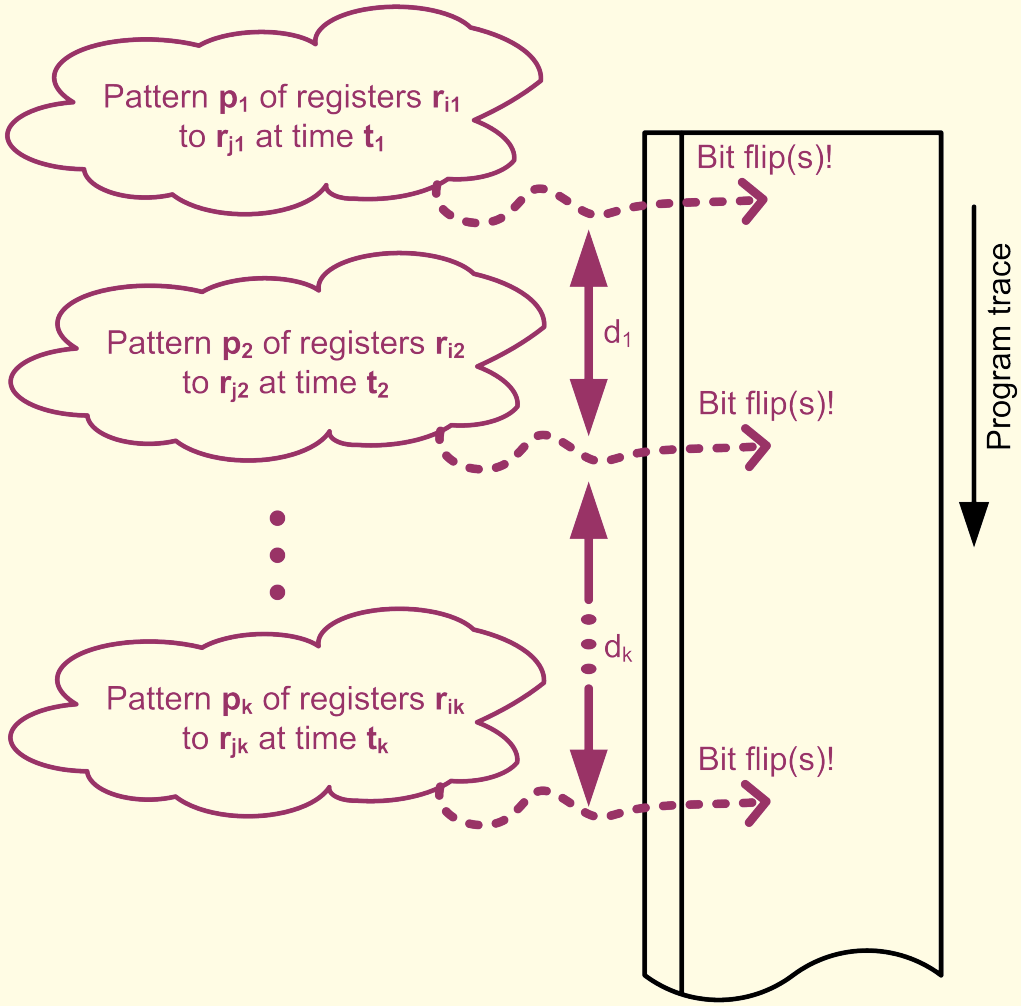


A Typical SW-based Error Model



b, r, and t are **random numbers**

A More Realistic SW-based Error Model



Patterns, register numbers, time intervals are **weighted random numbers**

Frequently Asked Questions

What is the big fuss about soft errors? Can't we just re-execute the application to deal with them?

- Re-run not possible in real-time systems
  - Voyager 2 data loss for weeks

[Photo from Wikipedia]

- Silent Data Corruption (SDC)
  - Important in life-critical systems

What are viable ways of evaluating the resilience of a design to soft errors?

- Architectural analysis
  - Like Architectural Vulnerability Factor (AVF) measurement
  - Lack of generality
  - Can be involved for some structures
- Error injection
  - Radiation to a chip
  - Modeling error and the design on FPGA
  - Modeling error in a simulation environment
- Place of injection
  - On any wire/component including combinational logic
  - On any memory component (flip-flops as well as system memory)
  - On SW-visible components only

What would be a good error model, and why? Is there any problem with this error model?

- Making logic (vs. memory arrays) error tolerant is expensive
- Important to be able to analyze error resilience of flip-flops
- Analyzing logic gate output errors not necessary [Seifert et al., 2012]
- Single bit-flip on a flip-flop is a golden error model
- Simulation speed too low for applications with reasonable size

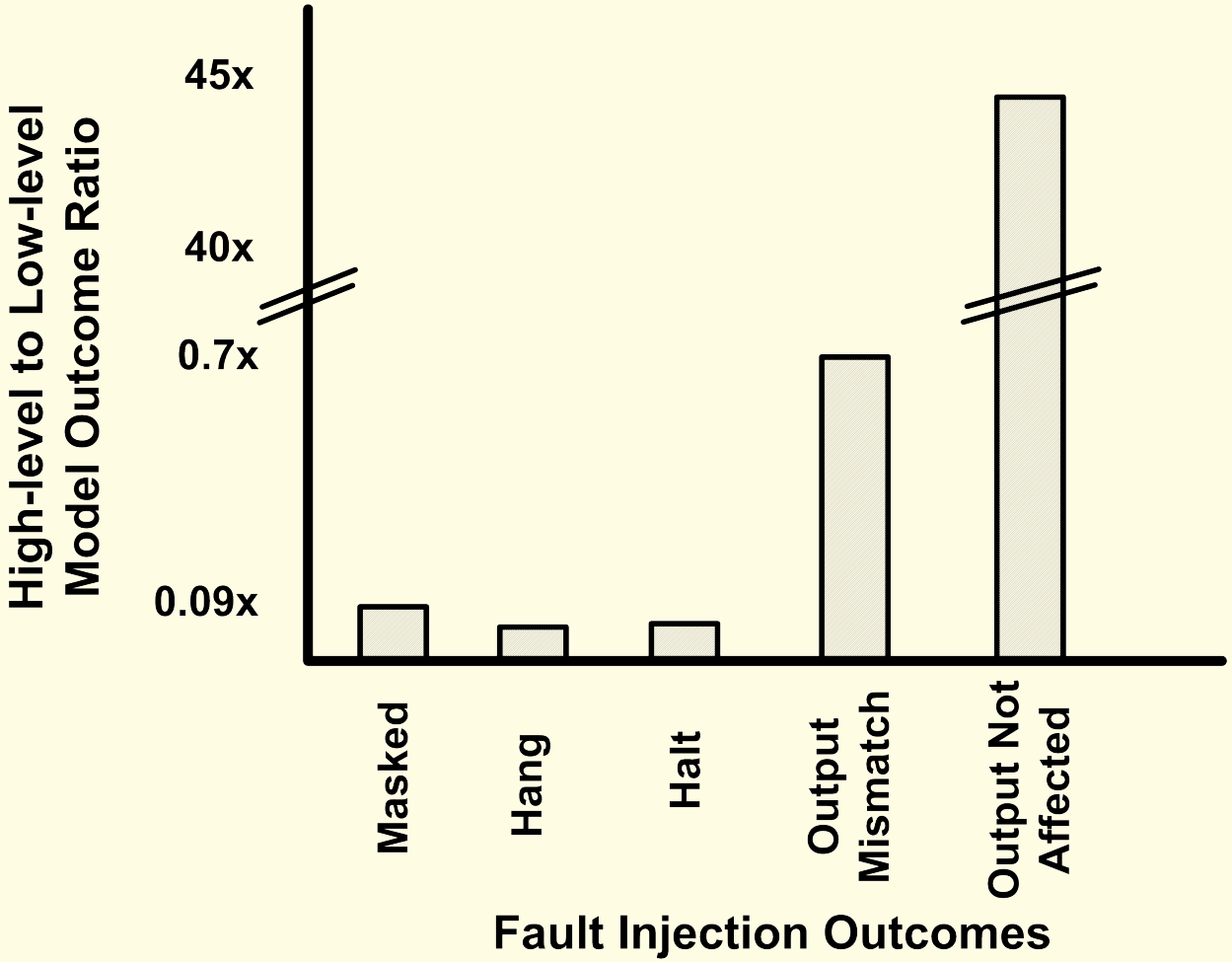
Why would we consider the single bit-flip on flip-flops the "gold standard"?

- Results of single bit-flip on flip-flops very close to irradiation [Sanda et al., 2008]
- Can be implemented in simulation- or emulation-based environments
- Low speed of simulation-based analysis is mitigated by hierarchical methods
- Still too slow!

What are the benefits and problems with only considering single bit-flips in SW-visible components?

- Several models consider single bit-flip on SW-visible components
  - a random bit of a random register
  - a random bit of a random memory element
  - a random bit of a register to be written
  - a random bit of a memory element to be written
- Can be analyzed faster than the gold standard model
  - Can be executed by fast instruction set simulators
- Inaccurate! SDC and DUE rates different than gold standard results

Why do we call the results of SW-based error injection inaccurate?



[Cho et al., 2013]

Is comparing HW-based injection with SW-based bit-flips apples to apples?

- SW-based injections used for soft-error vulnerability analysis?
- A need for comparing with a golden model
- We need to compare SW-based bit-flips with HW-based injection

Wouldn't the performance of HW-based injection make this approach prohibitively expensive?

- Analysis needs register-transfer- or gate-level to some extent
- Low simulation speed compared to instruction set simulation
- Might be able to perform fast but more accurate SW-based injections
  - Abstract the results of a small sample of HW-injections
  - A guide for a better SW-based error model
- The answer depends on the risks to have unreliable systems

What would we do if we did not have access to the detailed hardware design, RTL or lower level descriptions?

- The only possibility is SW-based injection
- Still can utilize pattern of HW-based bit-flips of past experiences

What are some of the statistics of the errors seen at the SW-level due to a single bit-flip at the HW-level?

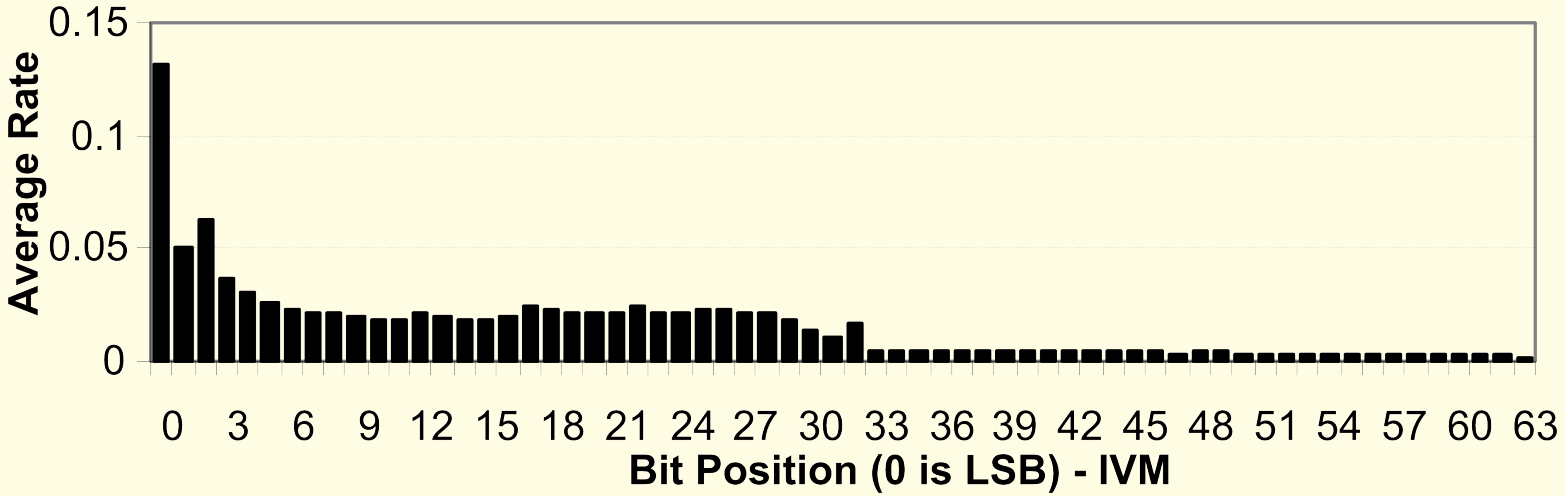
- Error propagation depends on
  - Circuit structure
  - Workload
  - Time of injection
- HW-based error injection experience done
  - On out-of-order IVM and in-order LEON3 processors
  - SPEC2000 benchmarks as workload
  - MinneSpec input sets as workload inputs
  - 160,000 injections for IVM (simulation)
  - 320,000 injections for LEON3 (FPGA)
- Error propagations to register file is studied

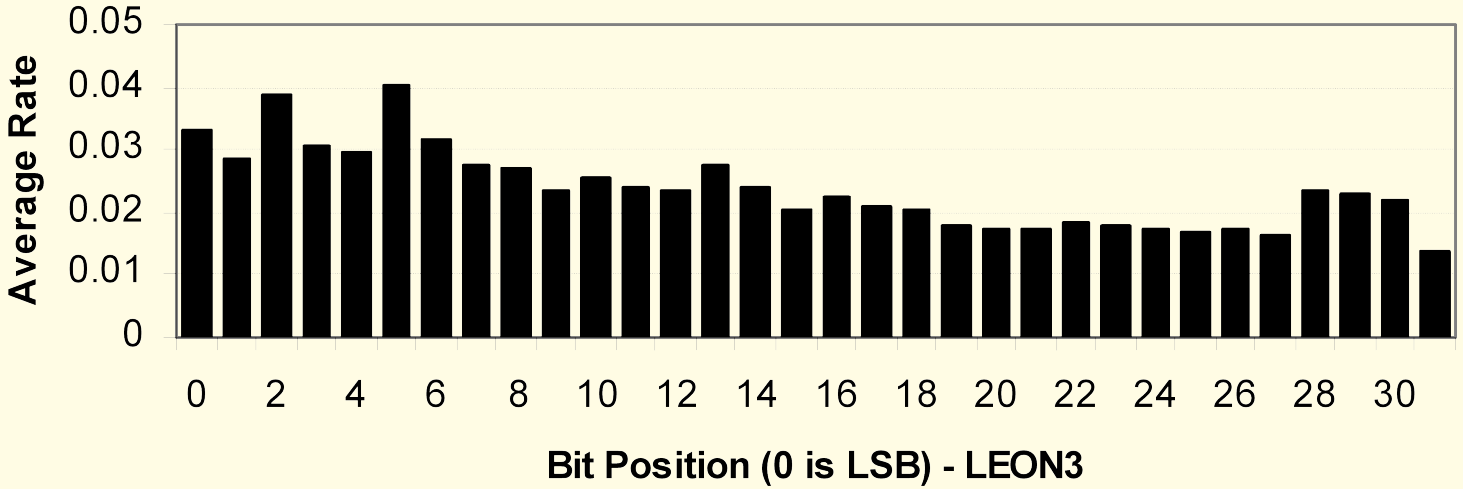
Experimental Results

Propagation Types

- Spatial
  - Register bit position(s) in each propagation
  - The number of affected bits in each propagation
- Temporal
  - The number of times error propagated to a register
  - The average time interval between each propagation

Propagation Rate to Each Bit in Registers

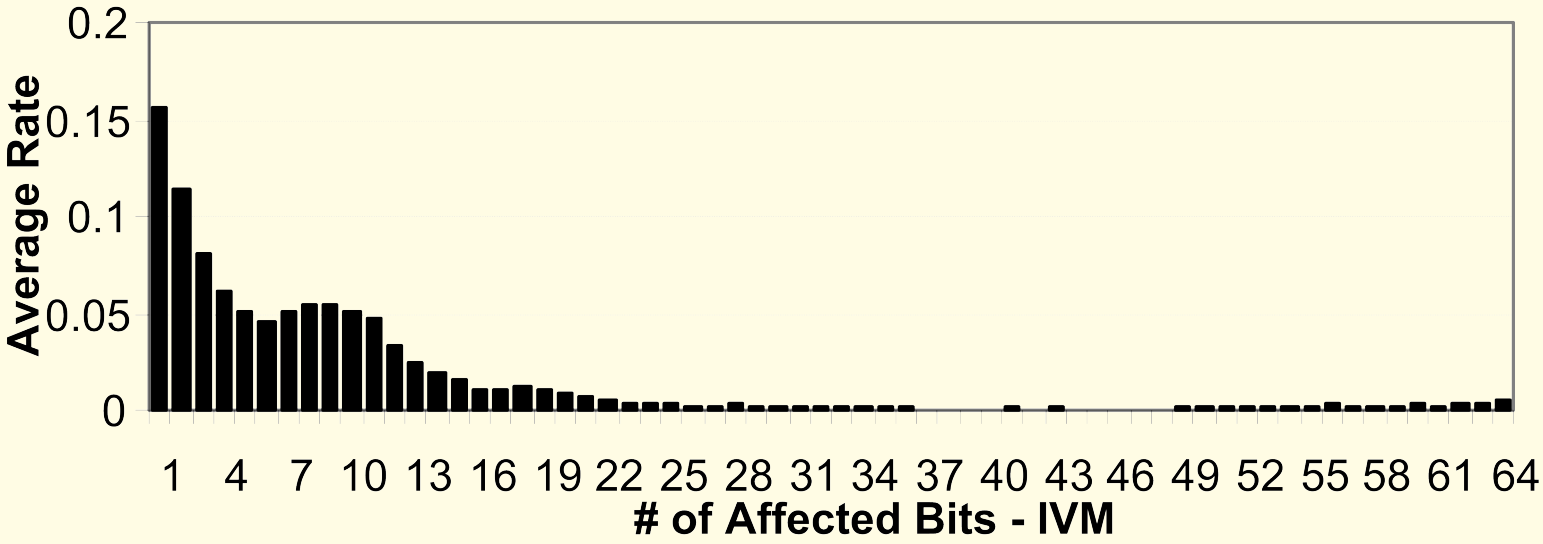


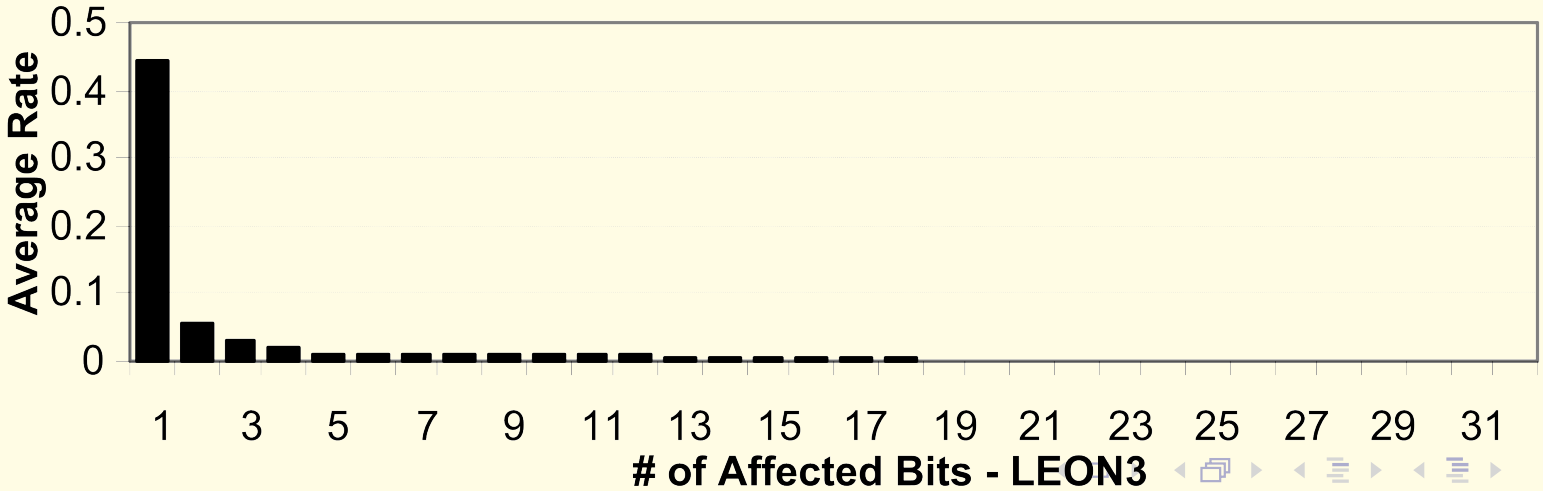


Propagation Rate to Each Bit in Registers-cont'd

- IVM
  - 14% of error propagations happen at bit 0
  - 88% of error propagations happen at bits 0-31
- LEON3
  - The distribution is close to uniform
  - Slightly more propagations happen at bits 0-15

Simultaneous Propagation to Certain Number of Bits in Registers

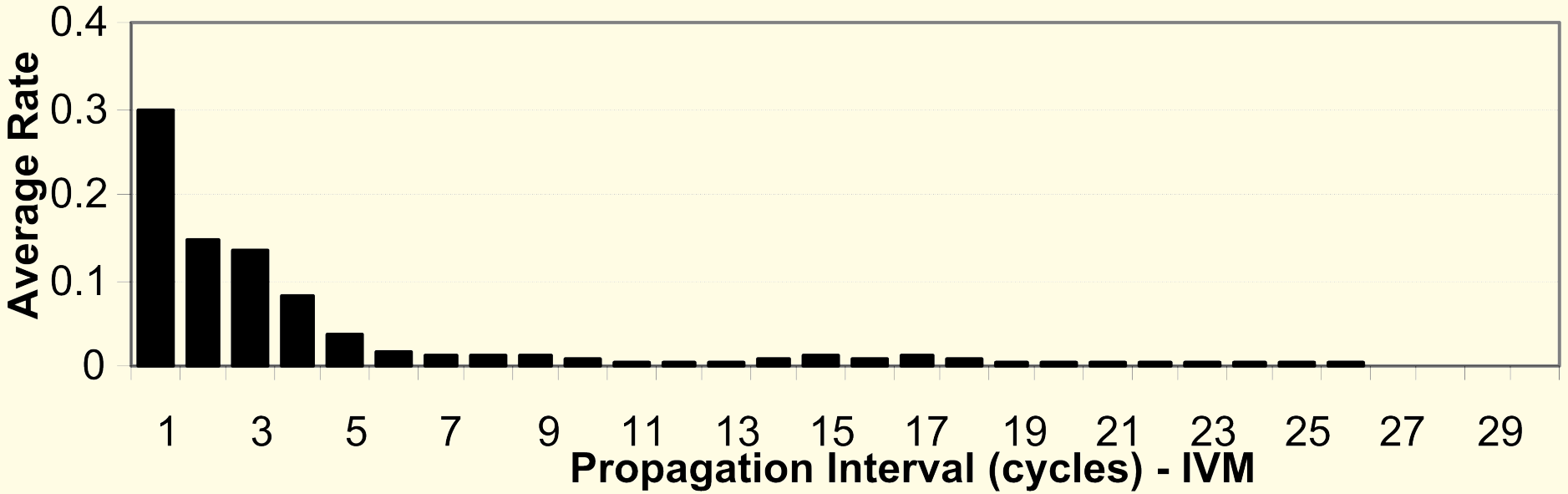




Simultaneous Propagation to Certain Number of Bits in Registers-cont'd

- IVM
  - Only 15% of propagations caused single bit flip in registers
  - 11% of propagations caused 2 simultaneous bit flips in registers
  - More than 60% of propagations cause 2-10 bit-flips
  - 16+ bit flips happen rarely compared to 15- bit flips
- LEON3
  - Single bit flips are very probable (around 45% of times)
  - 2 and 3 simultaneous bit flips happen more frequently than 4+ bit flips
  - Although 2+ bit flips is not very probable, it should be considered

Cycle Interval Between Each Propagation to a Register



Cycle Interval Between Each Propagation to a Register-cont'd

- For the cases of propagation in different cycles (multi-cycle propagation)
- 35% of the propagations are multi-cycle propagations
- How many cycles are propagations apart from each other?
- Done for IVM only
- 30% of multi-cycle propagations happen after 1 cycle
- Around 40% of multi-cycle propagations happen between 2 to 5 cycles
- Propagations happen between 6+ cycles are rare

Conclusions

- Single bit-flip SW-based injection means that HW-based error propagations:
  - are uniformly distributed for each bit
  - happen only in 1 bit (no simultaneous propagations)
  - happen only one time during an injection
- Statistics on error propagation depend on the design architecture
  - LEON3 statistics closer to the single bit-flip SW-based model
  - Less inaccuracy for SW-based injection in LEON3 vs. IVM expected
- The random error generator function should be changed to consider:
  - number of simultaneous bit flips
  - bit position of the bit flip(s)
  - multi-cycle injections
  - cycle interval between each bit flip (in multi-cycle injections)

Work in Progress

- Analyzing propagations to memory elements
- Analyzing the error patterns in memory elements and registers
  - Which bits are more probable to become erroneous at the same time?
- Changing the random error generator based on the information
- Injection based on the new random error generator and calculate the new inaccuracy level