

# An Investigation of the Fault Sensitivity of Four Benchmark Workloads

SOBRES 2012, September 19th

Behrooz Sangchoolie, Fatemeh Ayatollahi,  
Johan Karlsson

Chalmers University of Technology



# Outline

- Motivations
  - Transistor reliability trends
- *Objective and Contributions*
- Experimental environment
- *Fault model and Target applications*
- Summary of the main results
  - Failure mode distribution
  - Software-Implemented Hardware Fault Tolerance (SIHFT)
  - *Value failure distribution vs. executed instructions*
  - Value failure distribution over registers (voter vs. core function)
- Conclusion
- Future work

# Transistor Reliability Trend

Shekhar Borkar, Intel Corp:

*“As technology scales, variability in transistor performance will continue to increase, making transistors less and less reliable. ....*

*...Finding solutions to these challenges will require a concerted effort on the part of all the players in a system design.”*

## DESIGNING RELIABLE SYSTEMS FROM UNRELIABLE COMPONENTS: THE CHALLENGES OF TRANSISTOR VARIABILITY AND DEGRADATION

AS TECHNOLOGY SCALES, VARIABILITY IN TRANSISTOR PERFORMANCE WILL CONTINUE TO INCREASE, MAKING TRANSISTORS LESS AND LESS RELIABLE. THIS CREATES SEVERAL CHALLENGES IN BUILDING RELIABLE SYSTEMS, FROM THE UNPREDICTIBILITY OF DELAY TO INCREASING LEAKAGE CURRENT. FINDING SOLUTIONS TO THESE CHALLENGES WILL REQUIRE A CONCERTED EFFORT ON THE PART OF ALL THE PLAYERS IN A SYSTEM DESIGN.

..... VLSI performance has increased by five orders of magnitude in the last three decades, made possible by continued technology scaling. This trend will continue, providing an integration capacity of billions of transistors; however, power, energy, variability, and reliability are barriers to future scaling.

Die size, chip yields, and design productivity have so far limited transistor integration in a VLSI design. Now the focus has shifted to energy consumption, power dissipation, and power delivery.<sup>1</sup> Transistor subthreshold leakage continues to increase, and those of us in this industry have devised leakage avoidance, tolerance, and control techniques for circuits.<sup>2</sup> As technology scales further we will face new challenges, such as variability,<sup>3</sup> single-event upsets (soft errors), and device (transistor performance) degradation—these effects manifesting as inherent unreliability of the

components, posing design and test challenges. This article discusses these effects and proposes microarchitecture, circuit, and testing research that focuses on designing with many unreliable components (transistors) to yield reliable system designs.

This problem is not new; we design systems to account for reliability issues. For example, error-correcting codes are commonly used in memories to detect and correct soft errors. Careful designing and testing for frequency binning copes with variability in transistor performance. What is new is that as technology scaling continues, the impact of these issues increases, and we need to devise techniques to effectively deal with them.

### Sources of variations

There are three major sources that cause variations in transistor behavior. The first source is

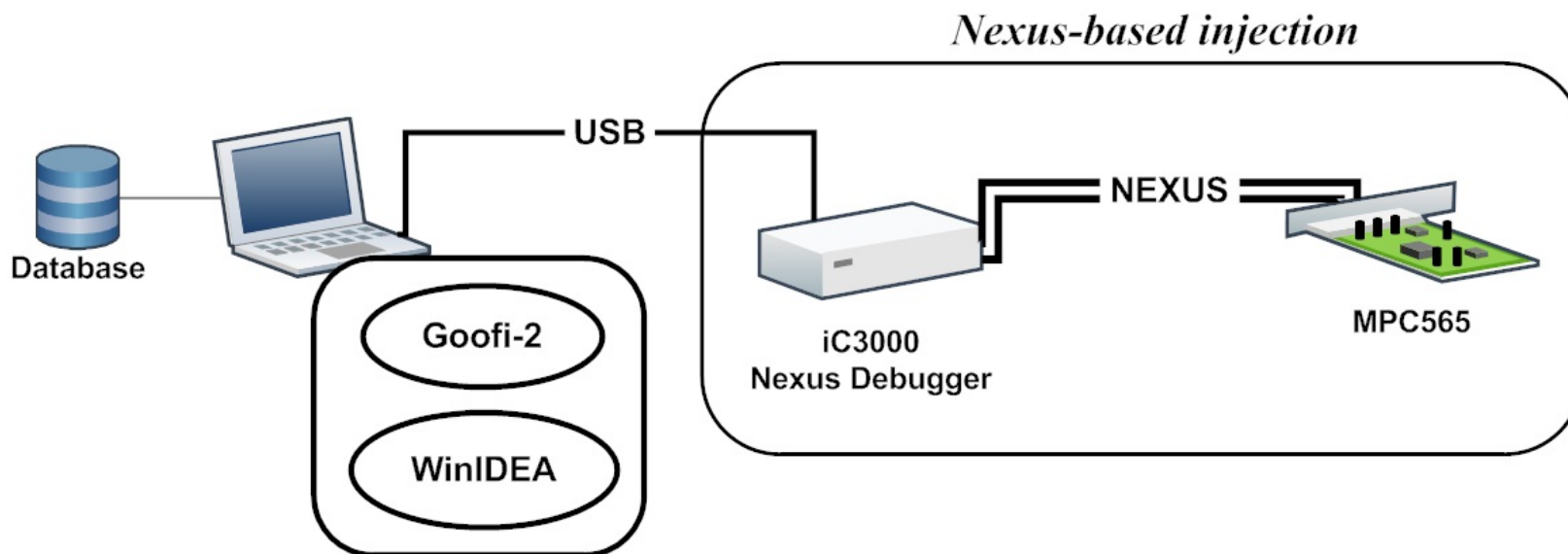
Shekhar Borkar  
Intel Corp.

Borkar, S.; "Designing reliable systems from unreliable components: the challenges of transistor variability and degradation," IEEE Micro, December 2005.

# Objective and Contributions

- Objective:
  - Conducting extensive fault injection experiments with and without software implemented hardware fault tolerance (SIHFT).
- Contributions:
  - An investigation of the failure mode distributions of four benchmark workloads.
  - Showing the effectiveness of the TTR-FR (triple-time Redundant execution with forward recovery) technique.
  - An analysis of the origin of the non-covered faults.

# Experimental Environmet



- The experiments are conducted automatically with the Generic Object-Oriented Fault Injection tool (Goofi-2)
- Nexus-based fault injection
  - Golden run
  - Fault injection experiments
  - SQL Database

# Target Applications

- Four workloads from the Mibench suite:
  - CRC-32 (32-bit Cyclic redundancy check)
  - SHA (Secure hash algorithm)
  - BinInt (Binary to integer converter)
  - Quicksort (Recursive sorting algorithm)
- Each workload is stimulated with nine different inputs. The combination of an input and a workload is called an *execution flow*.

Workload	Input variation
CRC-32 and SHA	Input length varies from 0 to 99 characters
BinInt	Input length varies from 0 to 31 characters (0s & 1s)
Quicksort	9 different permutation of an array of 6 integer elements

# Fault Model

- Single bit flips in ISA (Instruction Set Architecture) registers and main memory.
  - Bit flipping is used to emulate the effect of single event upset (SEU) errors.
  - A fault injection *experiment* consists of injecting one fault and observing its impact on a workload. For each execution flow **25000** experiments are conducted.

Target memory sections
Stack
Data
Sdata
Bss
Sbss

Target registers
General Purpose Registers(GPR)
Program Counter Register (PCR)
Link Register (LR)
Condition Register (CR)
Integer Exception Register (XER)

# Hardware Fault Sensitivity Measures for Software Components

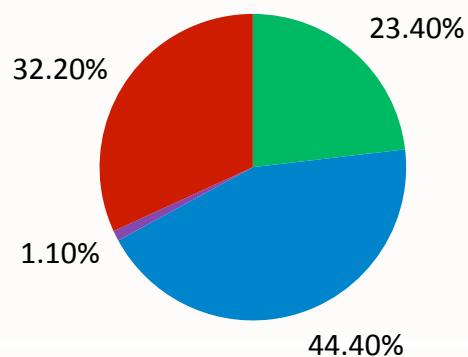
Measure	Failure mode	Definition
Failure Mode Distribution	No Impact (NI)	Errors that do not affect the output of the execution flow
	Time Out (TO)	Errors that cause violation of the timeout
	Detected by Hardware (DHW)	Errors that are detected by the hardware exceptions
	Detected by Software (DSW)	Errors that are detected by the software detection mechanisms
	Corrected by Software (CSW)	Errors that are corrected by the software correction mechanisms
	Value Failure (VF)	Erroneous output with no indication of failure (silent data corruption)
Error Coverage (COV)	$1 - VF$	The probability that a fault does not cause value failures



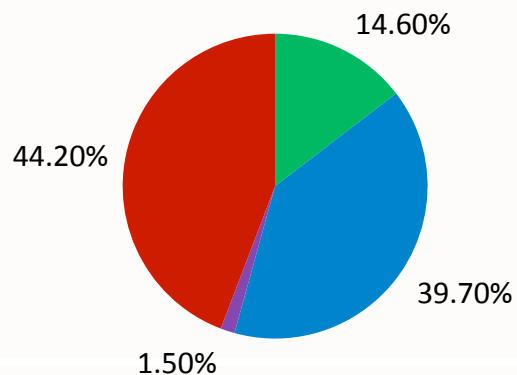
# Failure Mode Distributions

(with no Software Implemented Hardware Fault Tolerance)

## CRC-32




## SHA



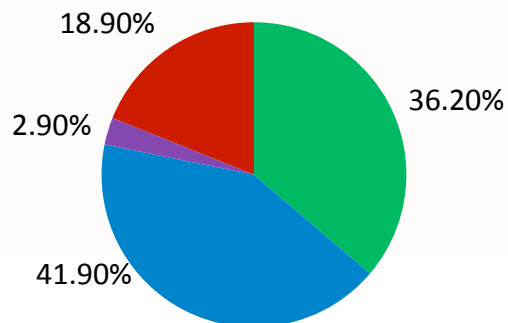
 Value failure

 No effect

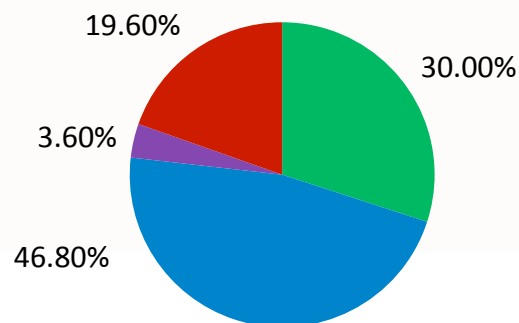
 Detected by hardware exception

 Time out

## BinInt



## Quicksort



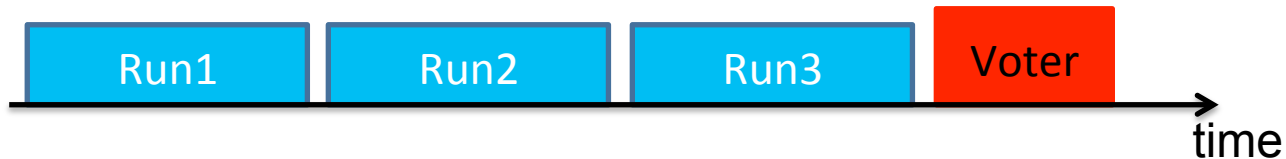
# Hardening Software Components Against Hardware Errors

## Goal

- To minimize the risk that hardware errors generate *undetected value failures*, a.k.a. *silent data corruptions*.

## How?

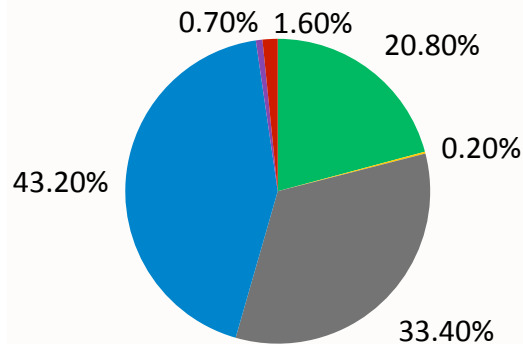
- A possible technique is the Triple-Time Redundant Execution with Forward Recovery.



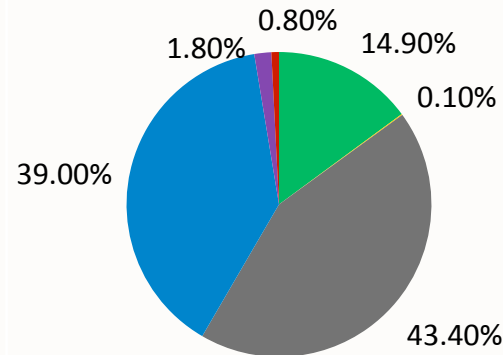
# Failure Mode Distributions

(with Triple-Time Redundant execution with Forward Recovery)

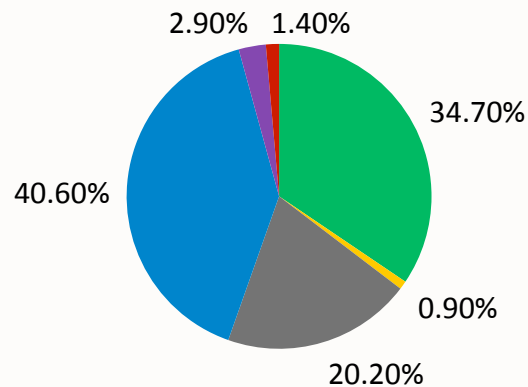
## CRC-32



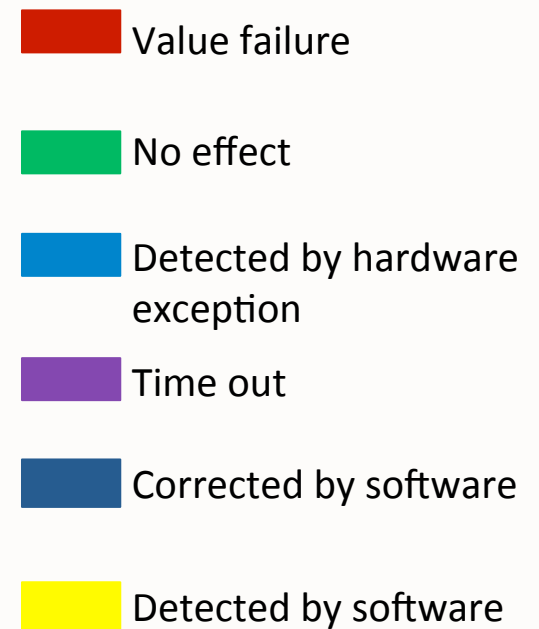
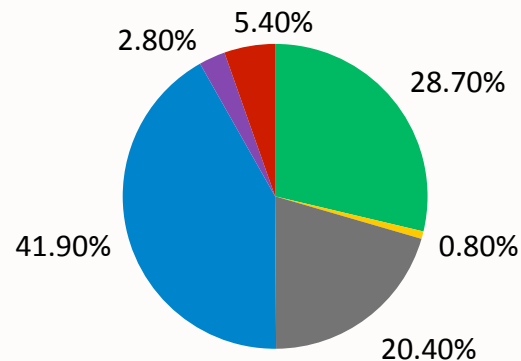
## SHA



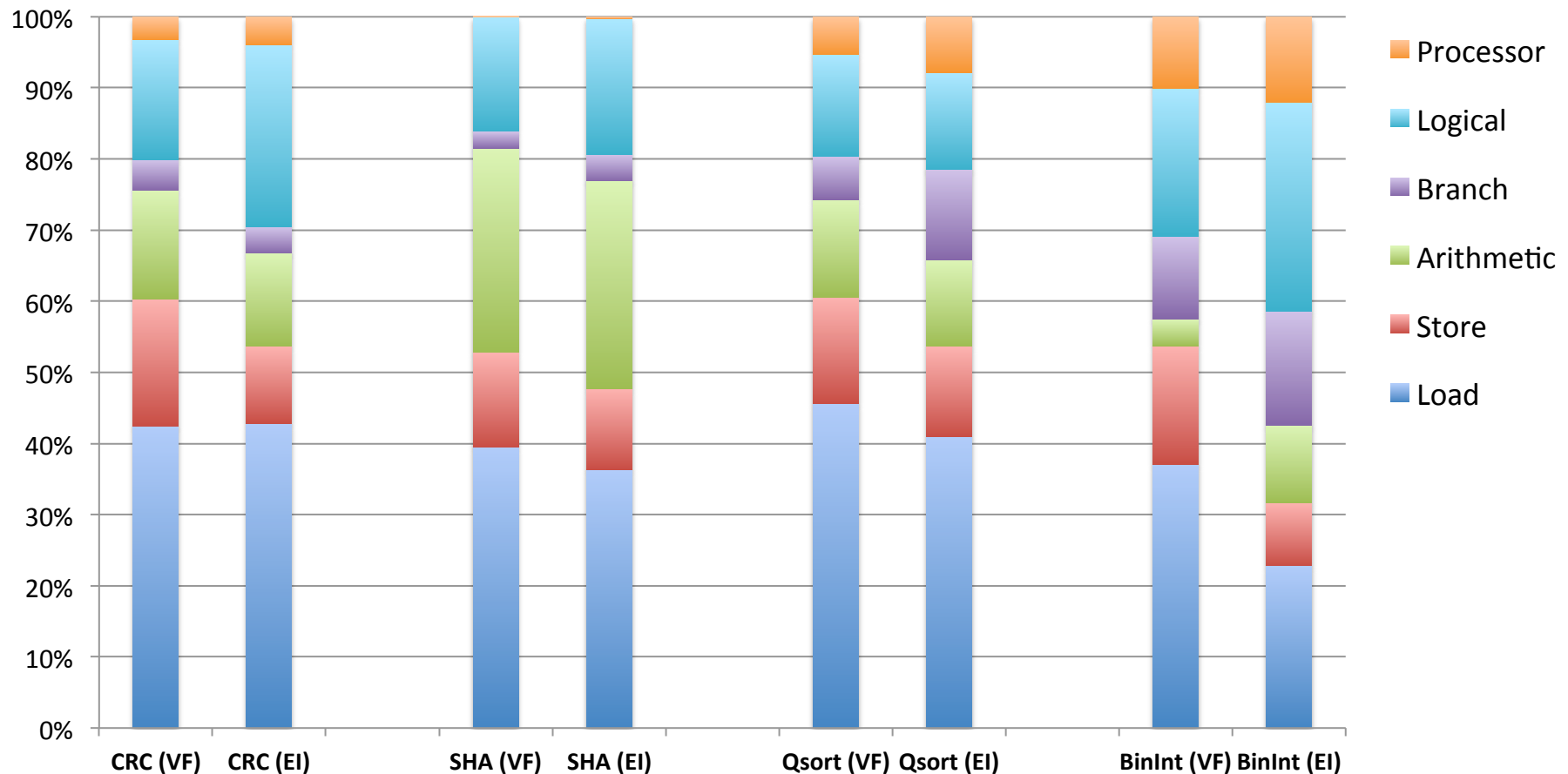
## BinInt



## Quicksort



# Value Failure Distribution vs. Executed Instructions

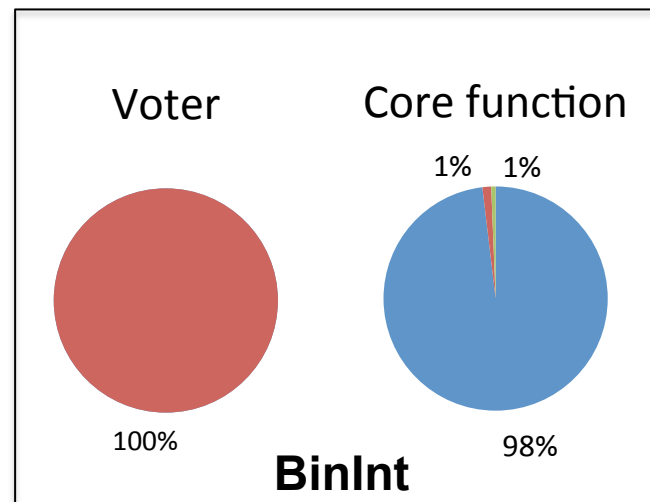
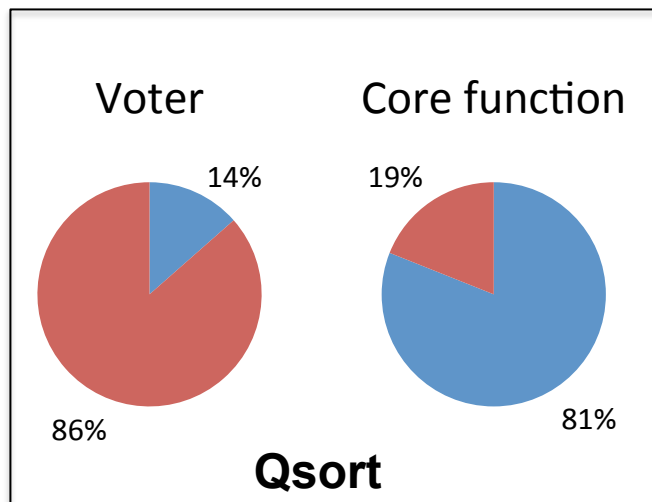
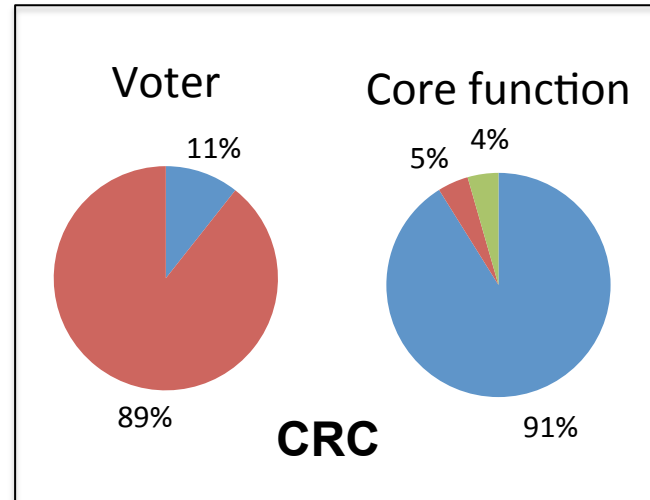
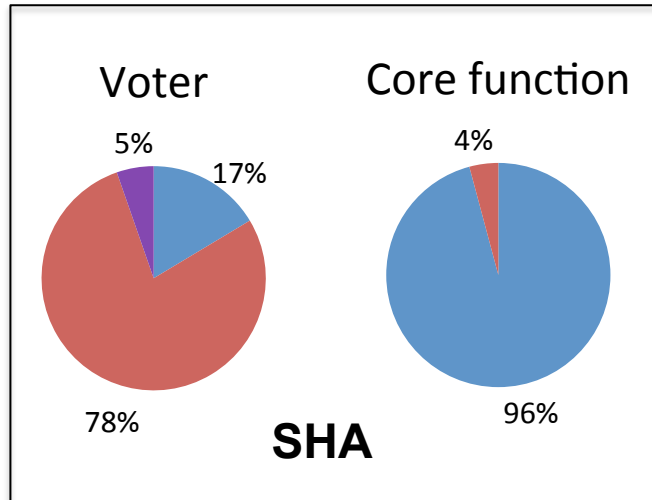


**Left bar (VF):** Value failure distribution over different instruction categories.

**Right bar (EI):** Percentage of the total number of executed instructions.

# Value Failure Distribution

(Voter vs. core function with respect to register faults)



# Conclusion

- Sensitivity to single bit-flip errors in the ISA registers and memory words varies for different programs.
- Software implemented time-redundant execution can to a great extent increase the error coverage.
- Software implemented time-redundant execution cannot alone achieve 100% error coverage. This technique needs to be complemented with various other mechanisms.
- The sensitivity of different registers varies in different parts of the source code.
- The number of executed instructions do not always have a correlation with the value failure distribution.

# Future Work

- Assessing the validity of the single-bit flip approximation.
- Extending the experiments with other target programs and different compiler optimizations.
- Applying additional software-implemented fault tolerant mechanisms to increase the error coverage.
- Investigating the impact of multiple-bit-flips on the failure mode distributions.
- Running fault-injection experiments on more complex applications like the Brake-by-wire system.
- Performing fault-injection experiments on AUTOSAR software components.
- Developing pre-injection analysis techniques for identifying transitively dead registers/memory words.
- And many more ...

# Further Reading

- Skarin, D.; Barbosa, R.; Karlsson, J.; “GOOFI-2: A Tool for Experimental Dependability Assessment”, in Proceedings of the 40th International Conference on Dependable Systems and Networks (DSN 2010), Chicago, Illinois, USA, June/July 2010
- Di Leo, D.; Ayatollahi, F.; Sangchoolie, B.; Karlsson, J.; Johansson, R.; “On the Impact of Hardware Faults - An Investigation of the Relationship between Workload Inputs and Failure Mode Distributions,” In The 31st International Conference on Computer Safety, Reliability and Security (SafeComp 2012), 2012.
- Ayatollahi, F.; Sangchoolie, B.; “On the Use of Assembly Code Metrics for Error Coverage Prediction,” Master of Science Thesis, Chalmers University of Technology, December 2012



# Acknowledgements

- Thank you all for attending this presentation.
- This work has partly been supported by VINNOVA-FFI BeSafe project.

# Questions?

