Formal Fault Tolerance Analysis of Algorithms for Redundant Systems in Early Design Stages

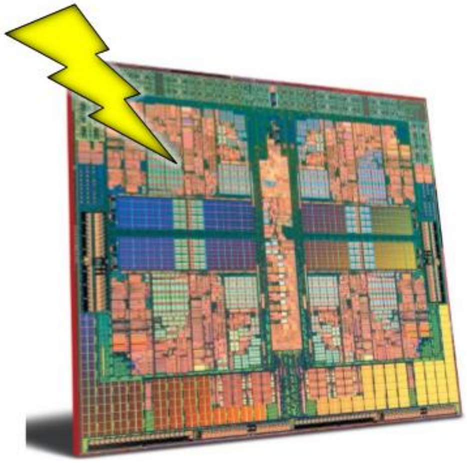
Andrea Höller, Nermin Kajtazovic,
Christopher Preschern and Christian Kreiner

Institute of Technical Informatics, Graz University of Technology

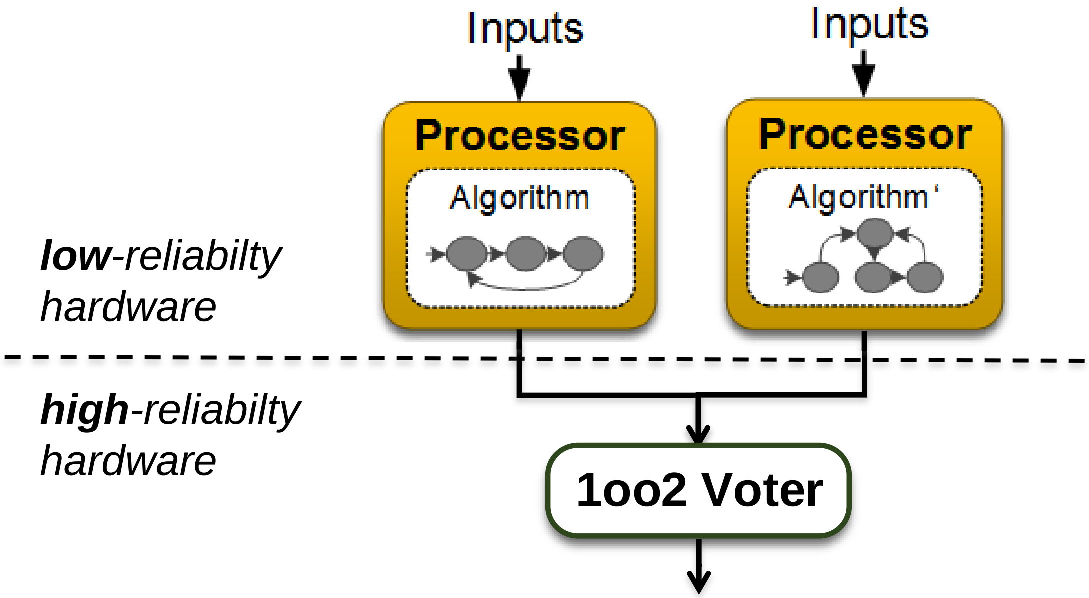
6th International Workshop on Software Engineering for Resilient Systems

Hardware Fault Tolerance is a Challenge

- Trend to use COTS-hardware components for safety-critical systems
- Increasing number of soft erros
- High level of fault tolerance required
- Hardware redundancy



Hardware Redundancy

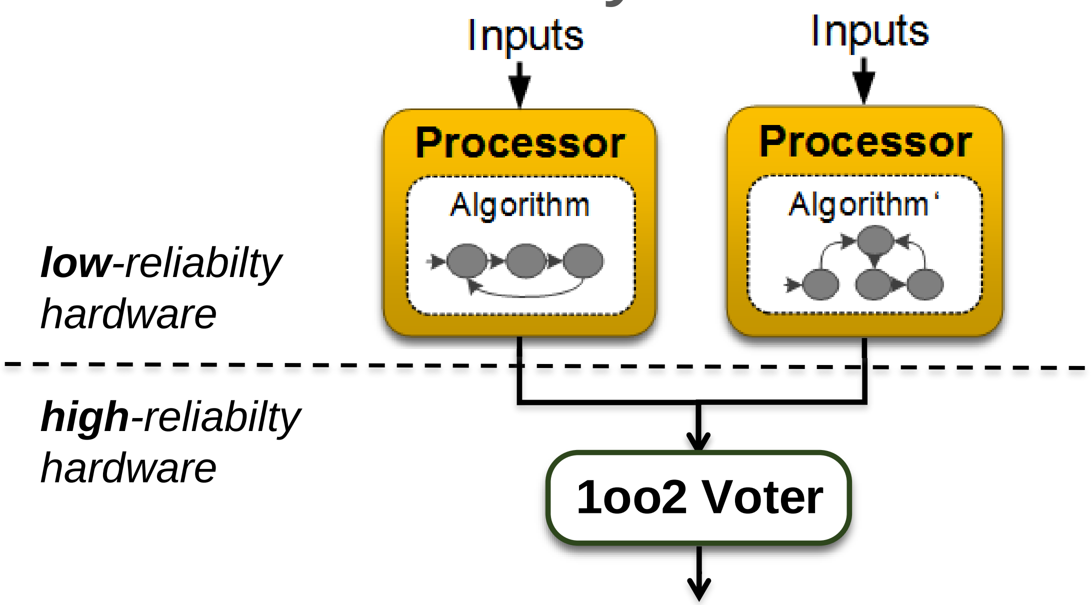


Which algorithms should be chosen to efficiently exploit the redundancy?

- Diversity to tolerate ...
  - systematic faults
  - random faults
- Inherent fault masking at algorithm-layer
  - Example: stuck-at-1 fault

1 OR 0 = 1 1 AND 0 = 0

Hardware Redundancy



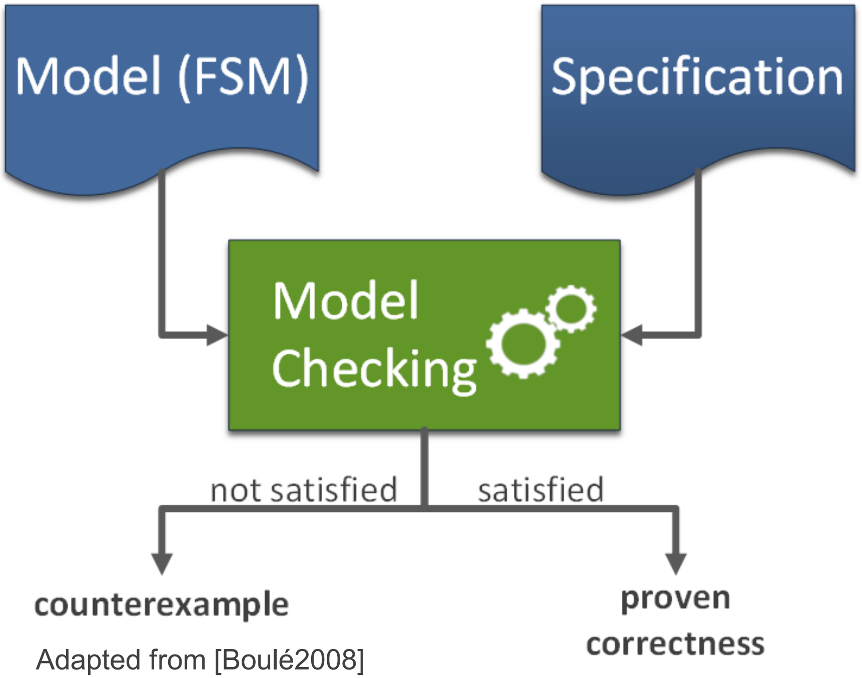
- Diversity to tolerate ...
  - systematic faults
  - random faults

Which algorithms should be chosen to efficiently exploit the redundancy?

- Inherent fault masking at algorithm-layer
  - Example: stuck-at-1 fault

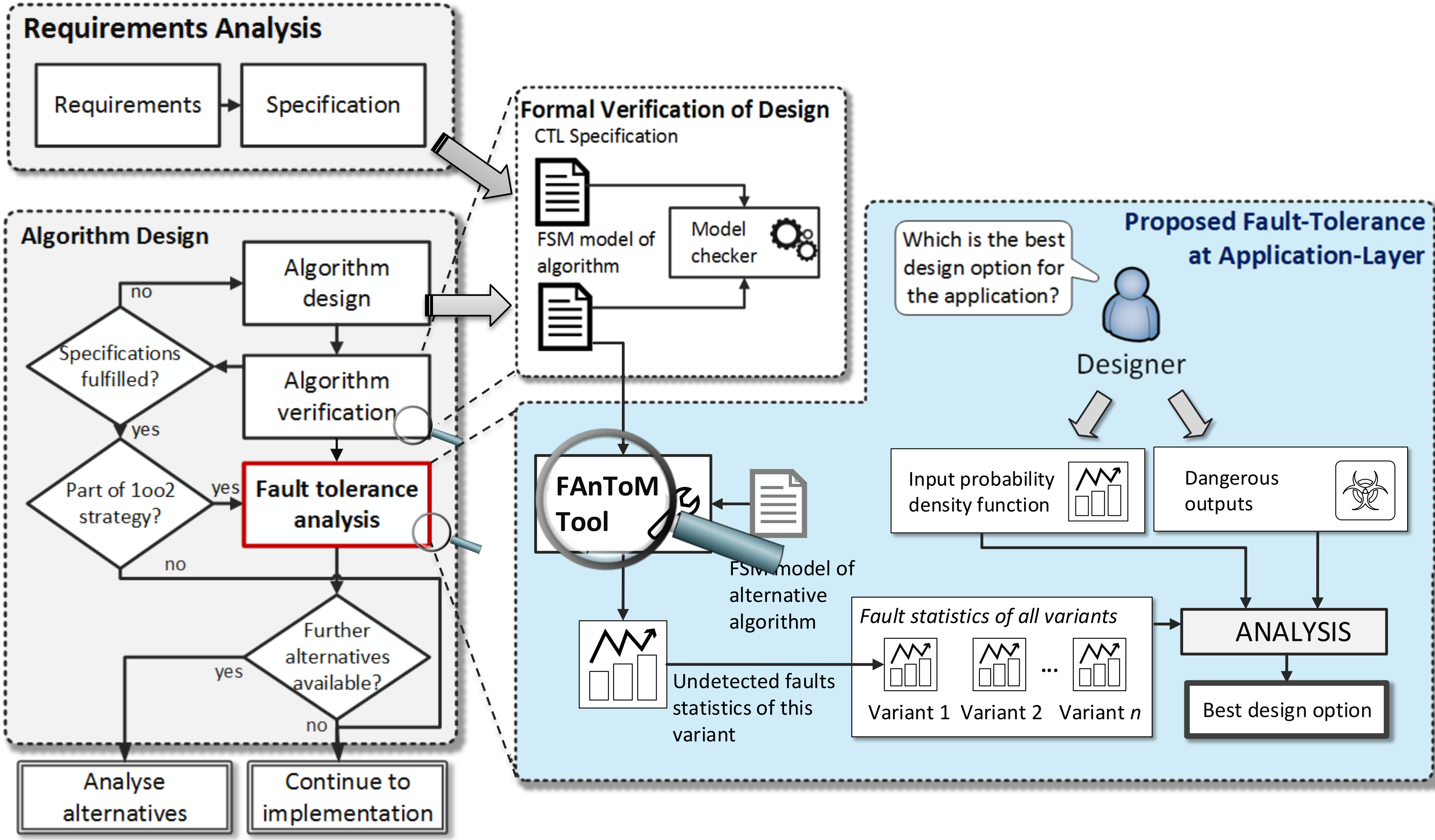
1 OR **1** = **1** 1 AND **1** = **1**

Model Checking



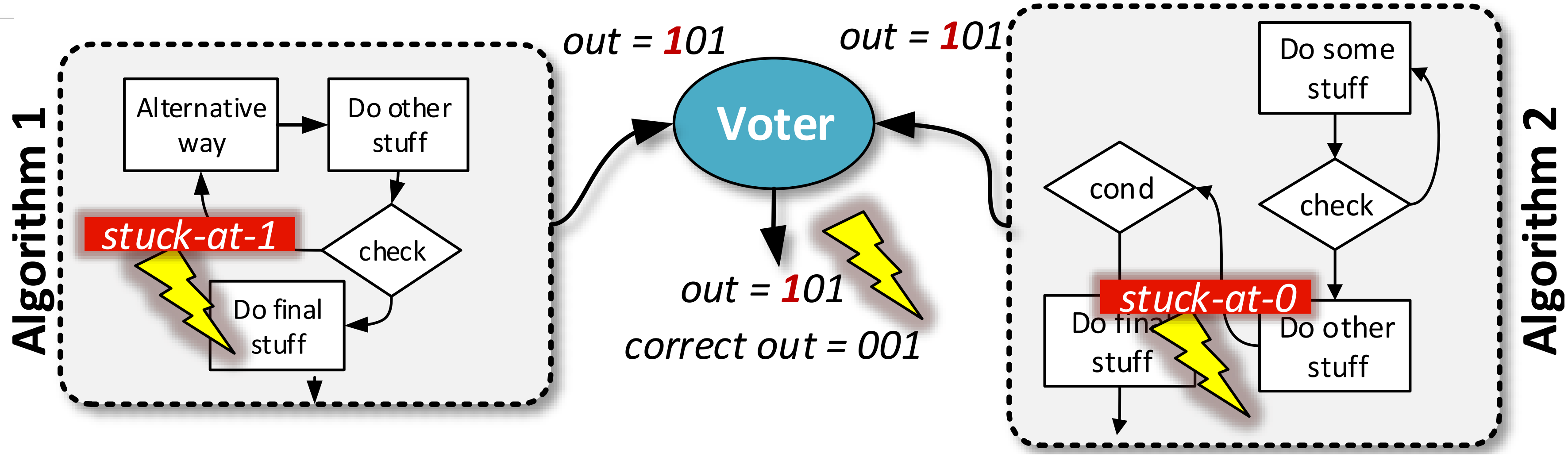
- NuSMV Tool
- Related work: Qualitative fault tolerance analysis [Huth2006, Krautz2006, Ezkiel2009]

Fault Tolerance Analysis in Early Design Stages



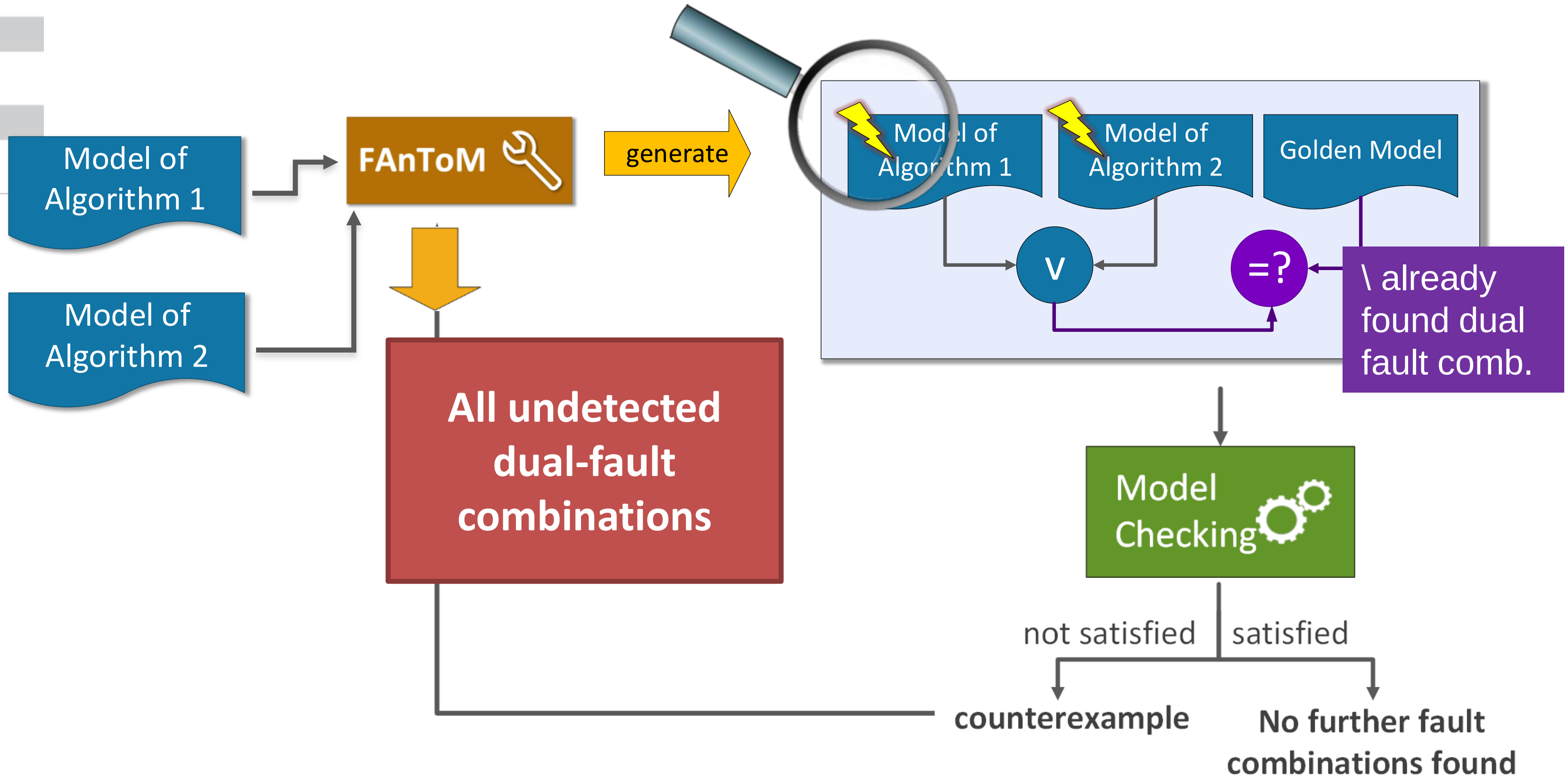
High-Level Evaluation of diverse Algorithms

→ Find all non-detected dual fault combinations



High-Level Evaluation of diverse Algorithms

→ Find all non-detected dual fault combinations



Fault Modeling

```
-- len…length of data variable, fi…should a fault be injected?
MODULE WORD(input, len, fi, my_var) --Data word with fault injection
 DEFINE     MAX_BIT_INJ := len - 1;
            fi_type     := my_var.fi_type; --fi-type of variant
 FROZENVAR inj_bit      := 0..MAX_BIT_INJ; --target bit of fi
 DEFINE out:= case
        fi & fi_type = stuck_at_0 : input & !(1 << inj_bit);
        fi & fi_type = stuck_at_1 : input |  (1 << inj_bit);
        fi & fi_type = bit_flip & my_var.fi_state = my_var.state :
                                                  input xor (1 << inj_bit);
         TRUE : input;    esac;
```

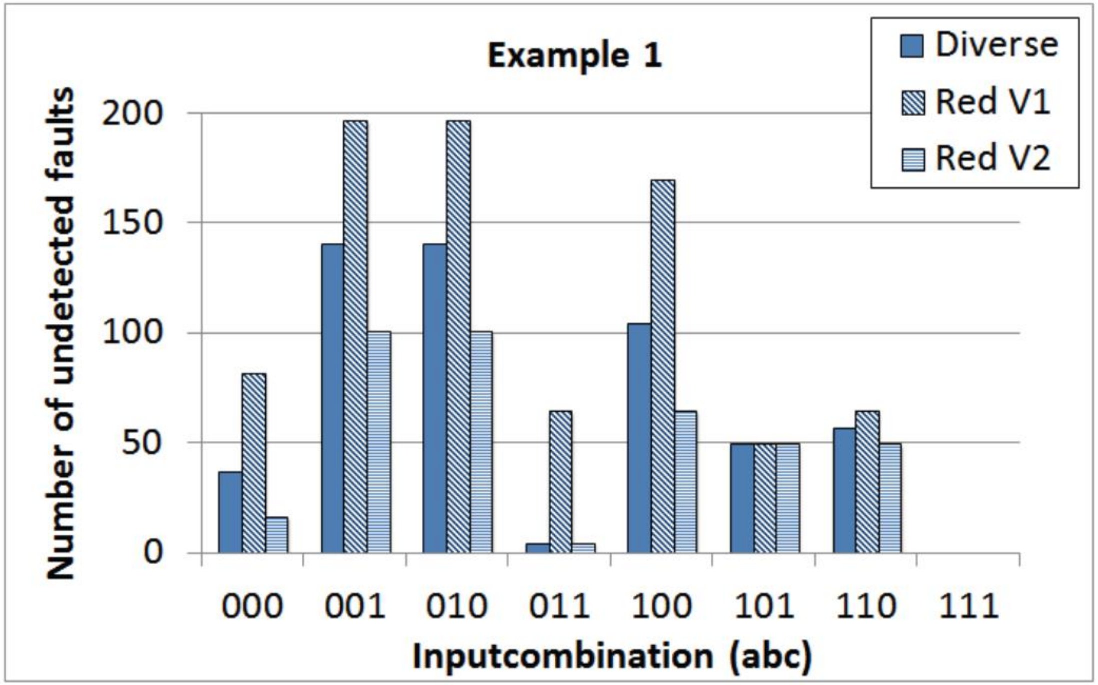
- Data faults
  - Stuck-at-x faults
  - Bit-flips
- Input variables to model random faults
  - Target signal
  - Fault type
  - Triggering time

Experimental Results

- Hydro-electrical power plant controller
- Simple boolean calculations
  - Example 1

$$\text{Variant 1}: (A \wedge B) \vee (C \wedge B) \vee (A \wedge C) \vee (B \wedge C)$$

$$\text{Variant 2}: B \wedge (A \vee C) \vee C \wedge (A \vee B)$$

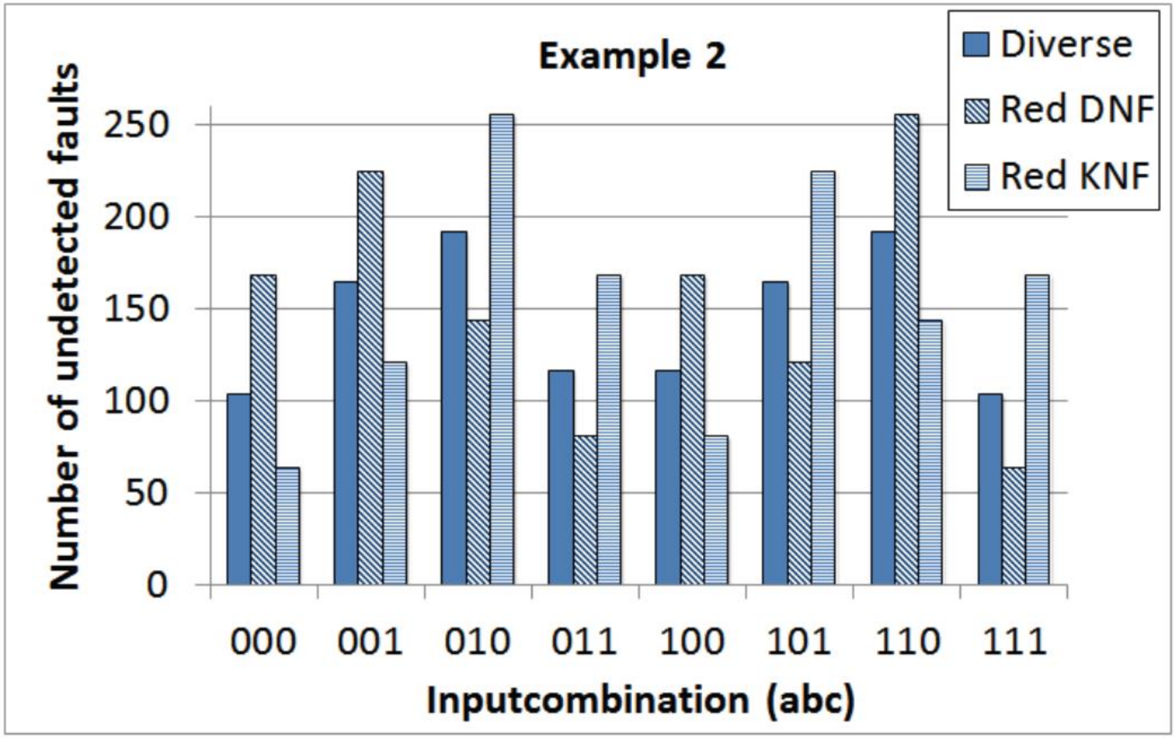


Experimental Results

- Simple boolean calculations
  - Example 2

$$\mathrm{DNF}: (\bar{A} \wedge B \wedge \bar{C}) \vee (\bar{A} \wedge B \wedge C) \vee (A \wedge \bar{B} \wedge C) \vee (A \wedge B \wedge C)$$

$$\mathrm{CNF}: (A \vee B \vee C) \wedge (A \vee B \vee \bar{C}) \wedge (\bar{A} \vee B \vee C) \wedge (\bar{A} \vee \bar{B} \vee C)$$



Conclusion

- Formal quantitative fault tolerance analysis
- Applicable in early design stages
- Understanding of algorithm-specific fault tolerance
  - Influence of input
  - Probability of certain errorneous output value
- Scalability issues
- Future work
  - Application for control-flow analysis and security
  - Handle more complex designs

Thank you very much for your attention!

Any questions?

References

[Boulé2008] M. Boulé and Z. Zilic. Generating hardware assertion checkers: for hardware verification, emulation, post-fabrication debugging and on-line monitoring. Springer Verlag, 2008.

[Ezekiel2009] J. Ezekiel and A. Lomuscio. Combining fault injection and model checking to verify fault tolerance in multi-agent systems. In Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1, 2009.

[Huth2006] Huth, M., Ryan, M. Logic in Computer Science: Modeling and reasoning about systems. Cambridge University Press, 2006

[Krautz2006] Krautz et al. Evaluating coverage of error detection logic for soft errors using formal methods. DATE 2006

[Ezkiel2009] Ezekiel, J. Lomuscio, A.: Combining fault injection and model checking to verify fault tolerance in multi-agent systems. AAMS, 2009