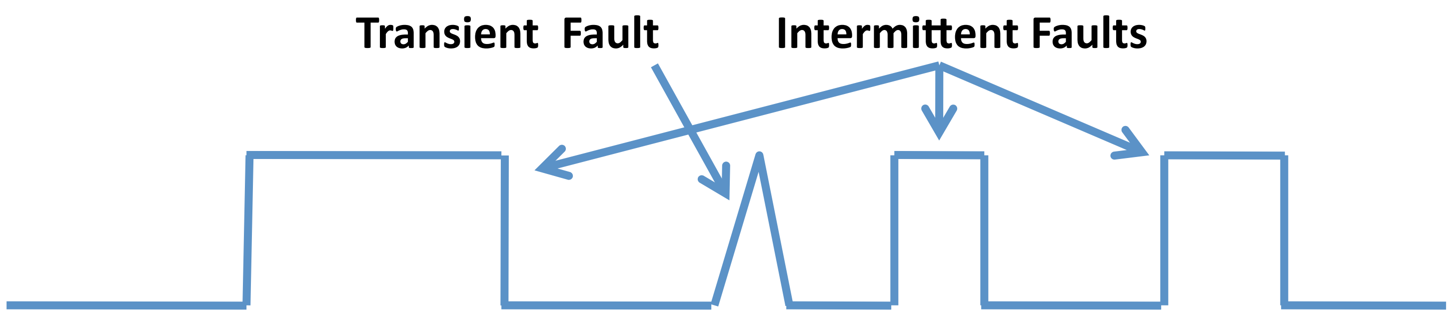
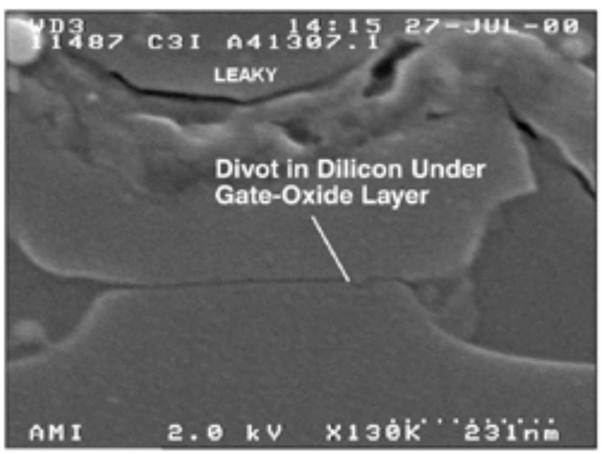
# BackTrack: Diagnosing Hardware Faults using Software Techniques

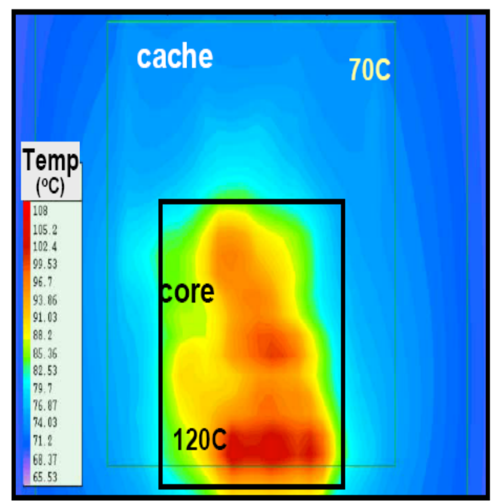
Layali Rashid
Jiesheng Wei
**Karthik Pattabiraman**
Sathish Gopalakrishnan

# Motivation: Intermittent Faults

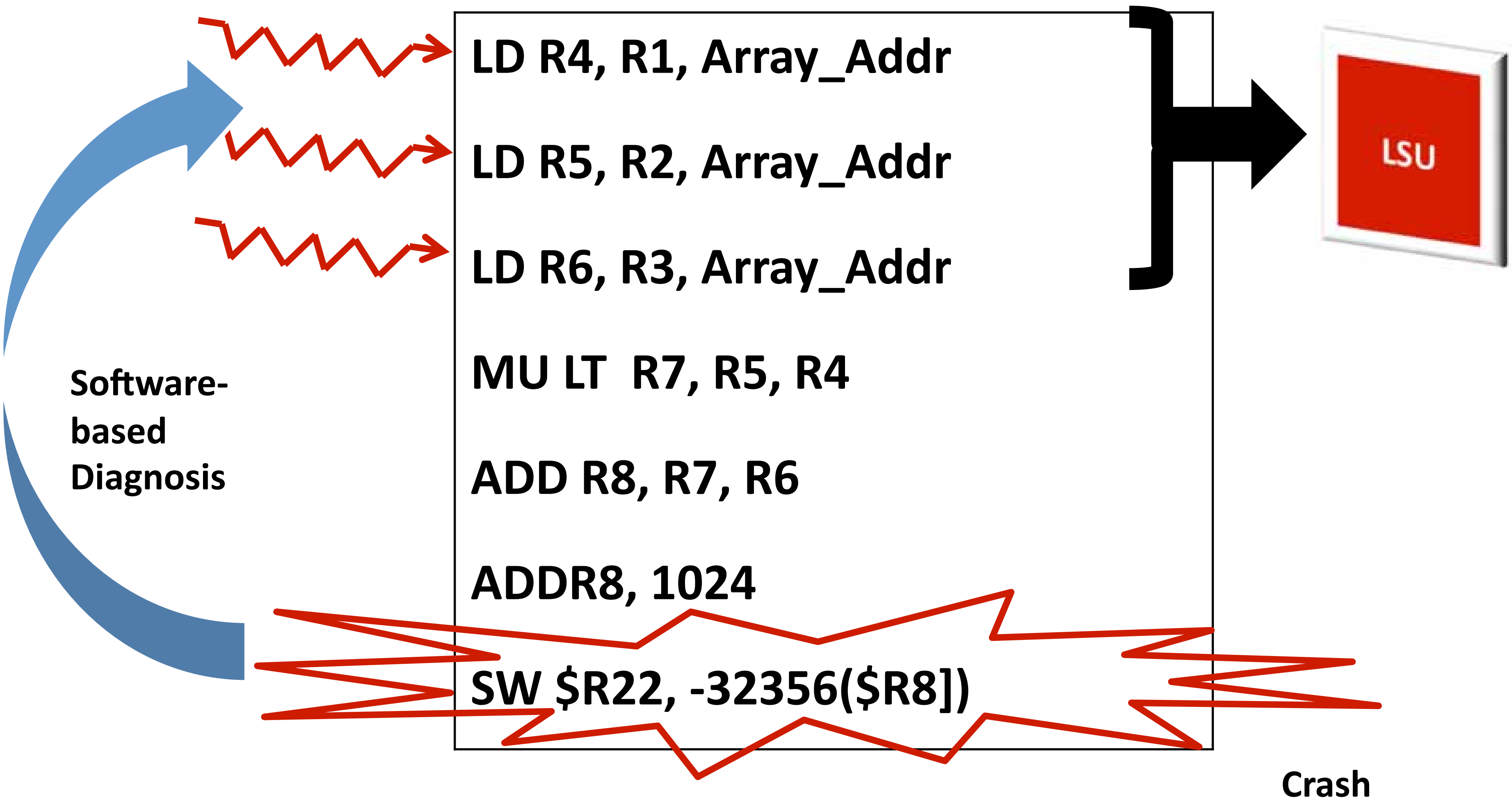
Intermittent faults are increasing in processors [Constantinescu'07][Sohi'08][Nightingale'11]



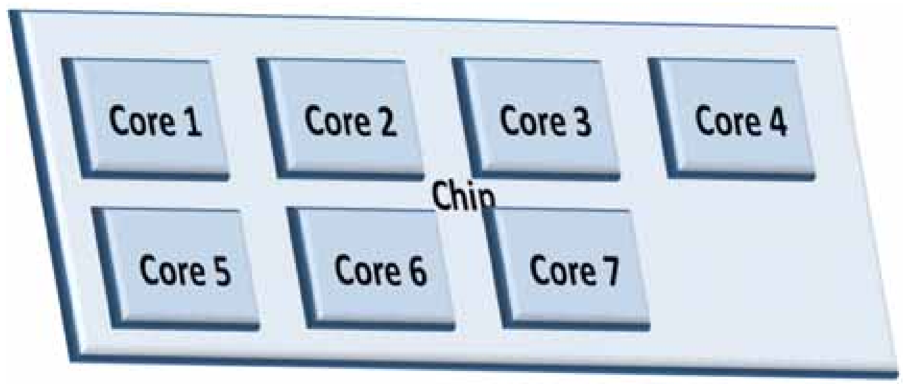
# Diagnosis: Overview



# Motivation: Why Diagnose ?

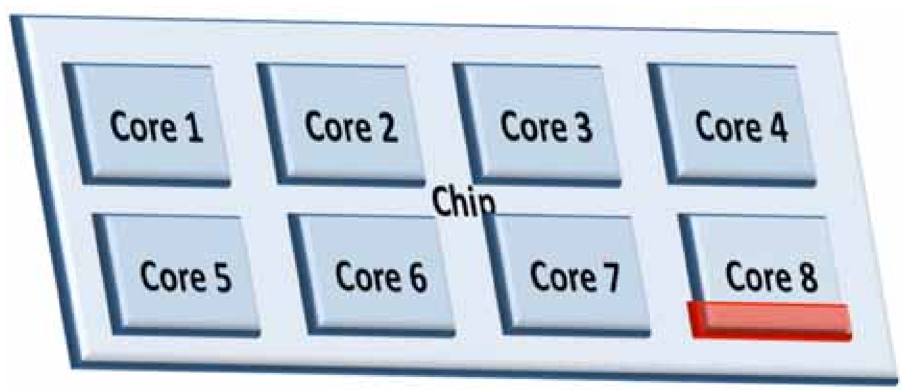
- ❑ Enable fine-grained recovery techniques
  - ❑ Increase the number of usable cores

without diagnosis




**Core 8 is disabled**

with diagnosis




**Only part of core 8 is disabled**

# Motivation: Why Software-based ?

- Low power and performance overheads during fault-free operation – **light-weight**

- No need for hardware changes - **compatible**

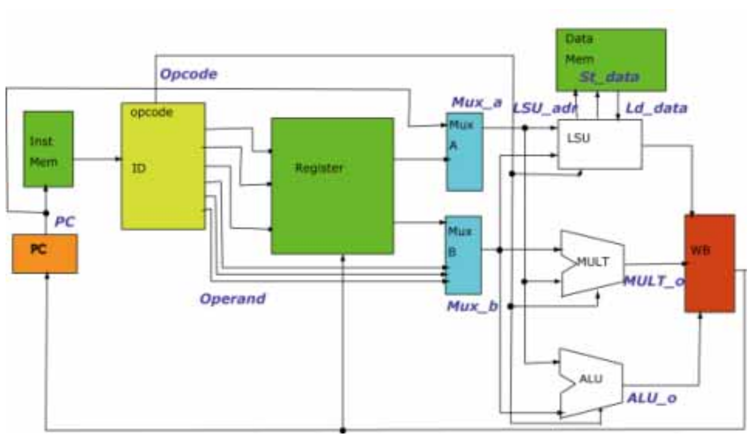
- No need to run tests or special diagnostics
  - **Only diagnose faults that cause appln. failures**

# Outline

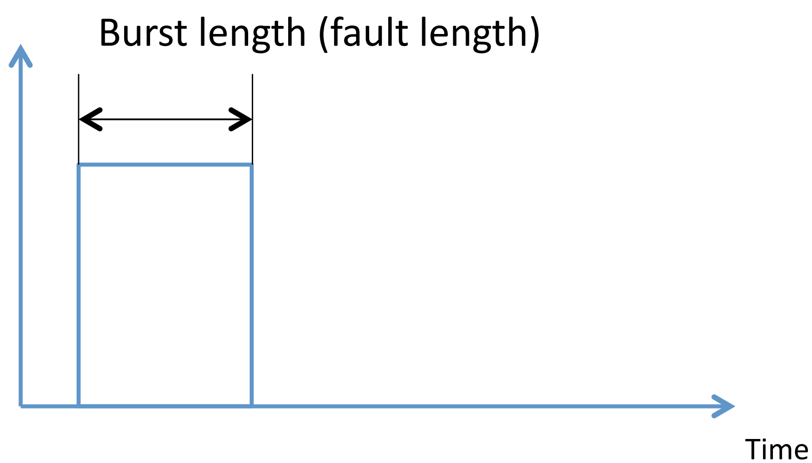
- Motivation
- Fault model
- Approach
- Results
- Conclusions

# Fault Model

- Single **signal** in processor experiences **stuck-at** zero/one fault for  a **specific time** duration

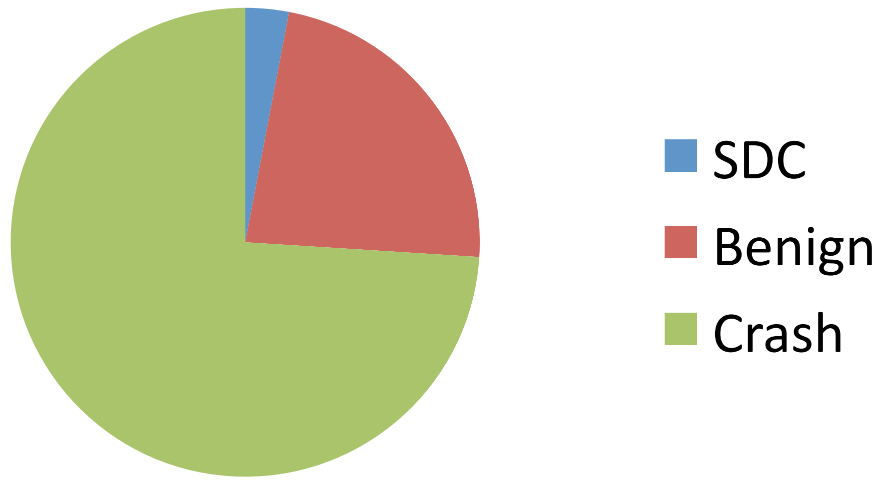



Spatial characterization




Temporal characterization

# Fault-Injection Study: Major Findings



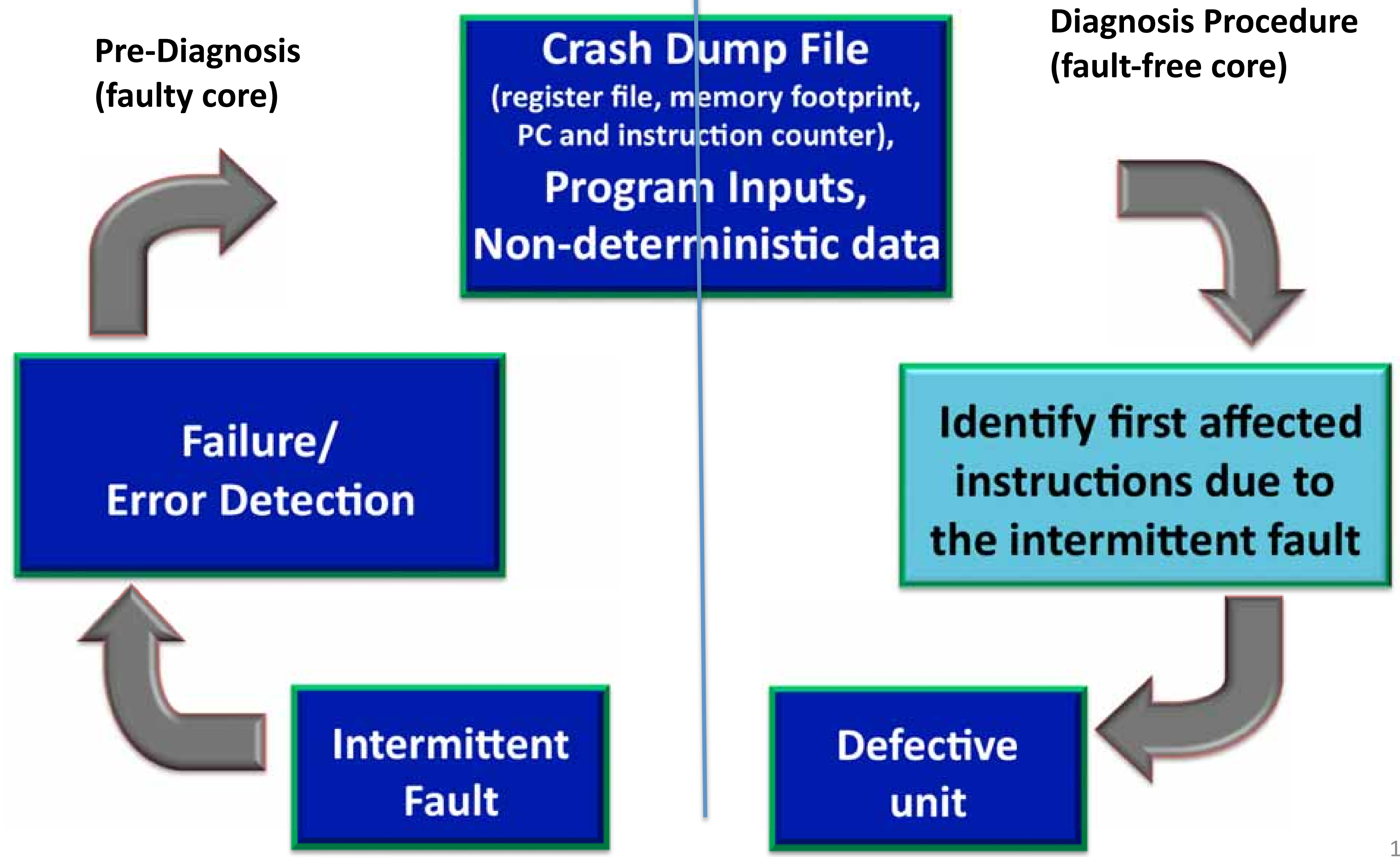

- Of the intermittent faults that are **non-benign**, **95%** result in a **program crash**

- **More than 90%** of the faults cause program to crash **within 500 instructions** from the fault

# Outline

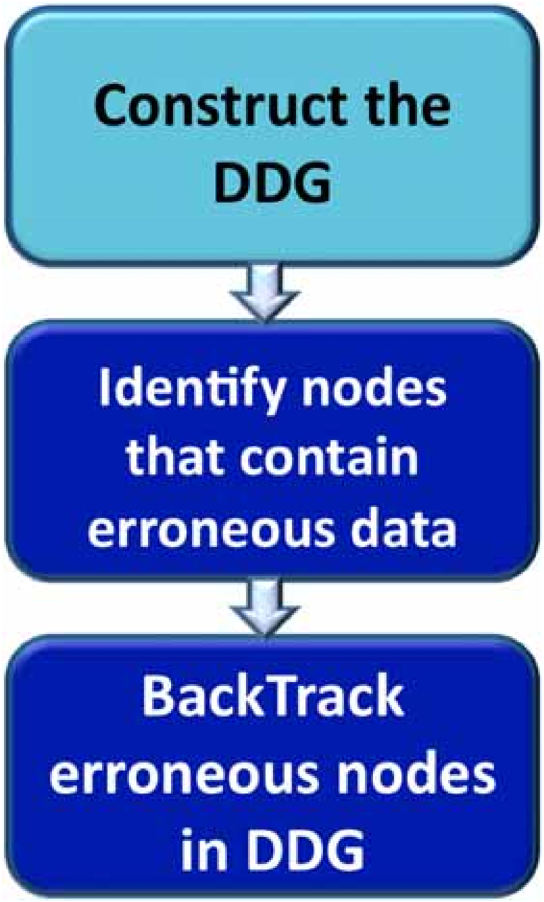
- Motivation
- Fault model
- Approach
- Results
- Conclusions

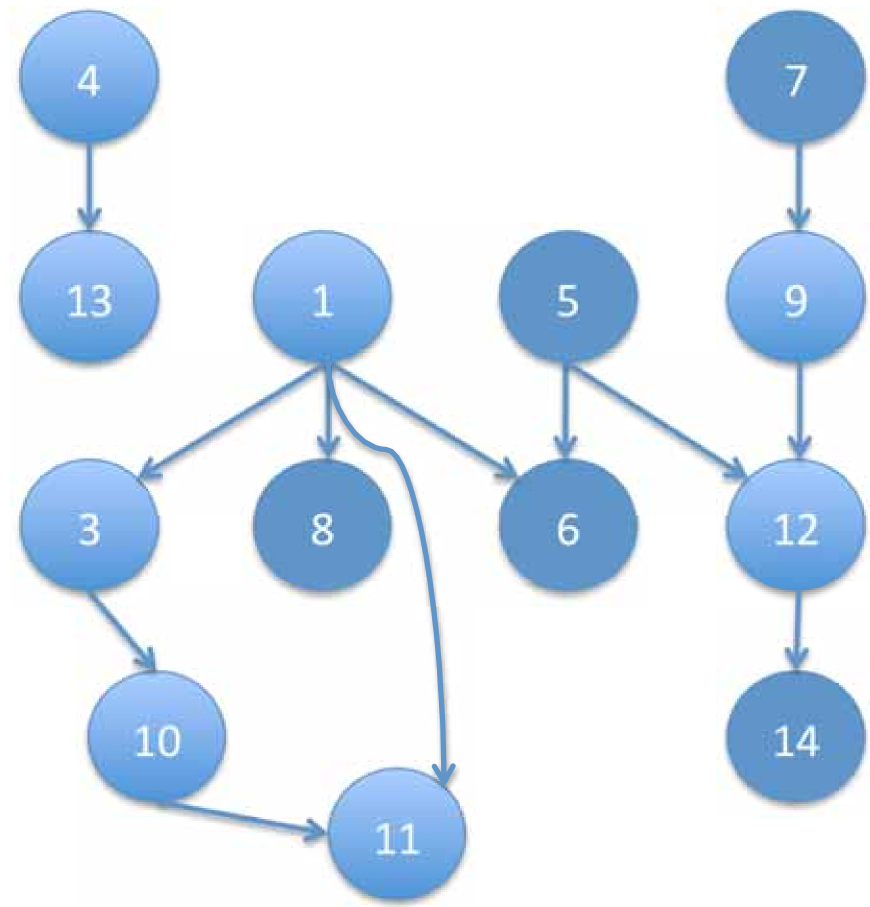
# Approach



Pre-Diagnosis
(faulty core)
Crash Dump File
(register file, memory footprint, PC and instruction counter),
Program Inputs,
Non-deterministic data
Diagnosis Procedure
(fault-free core)
Failure/
Error Detection
Identify first affected instructions due to the intermittent fault
Intermittent Fault
Defective unit

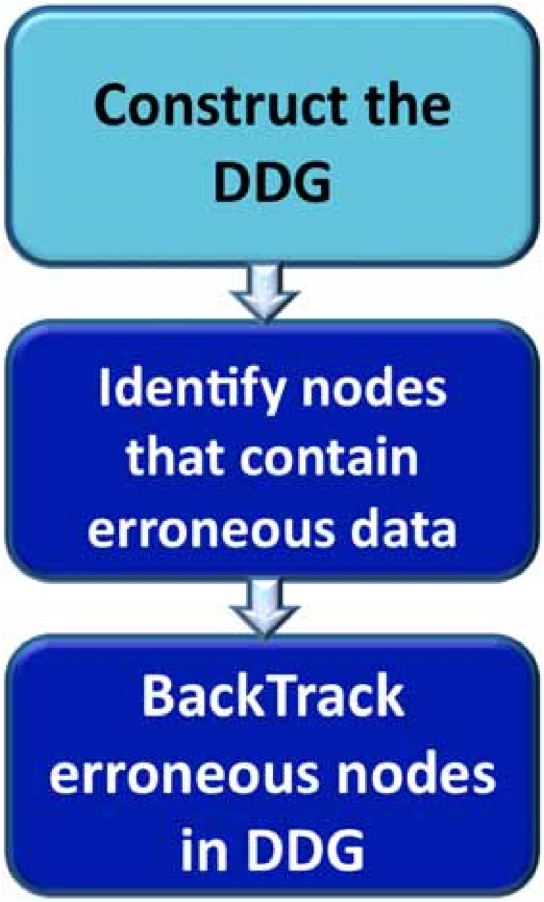
# Example

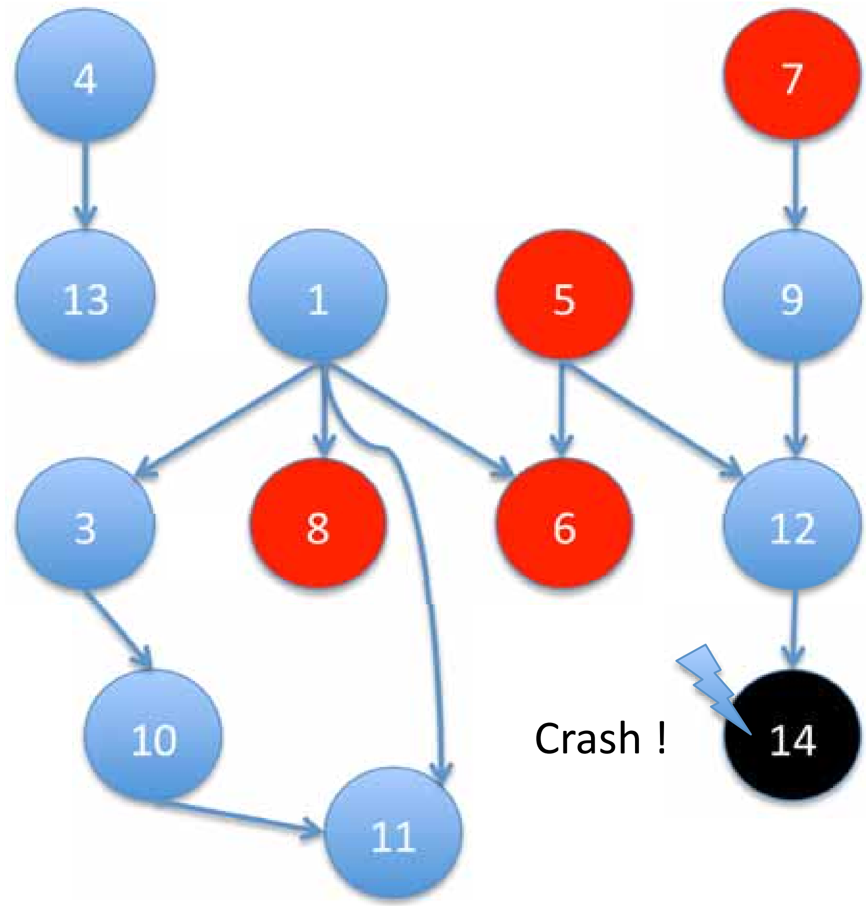



Nodes → Instructions. Edges → Dependencies

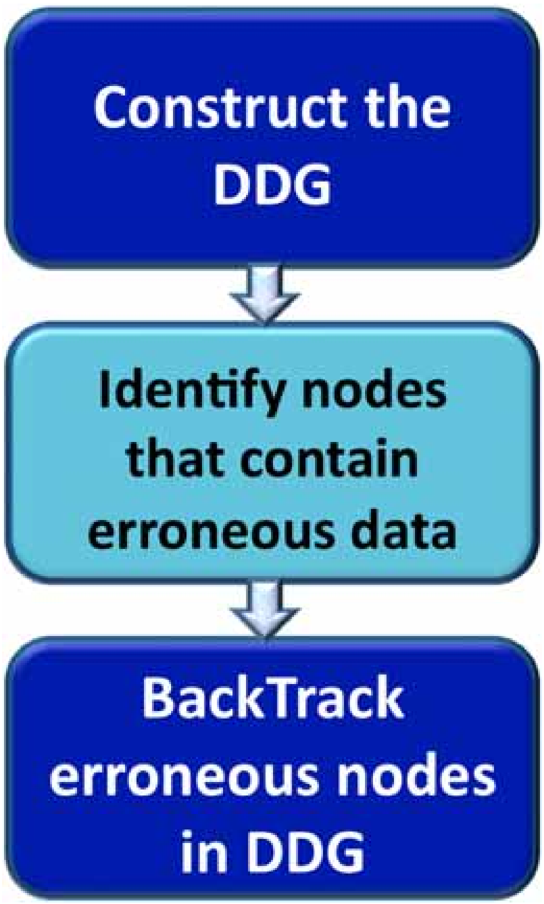
# Example

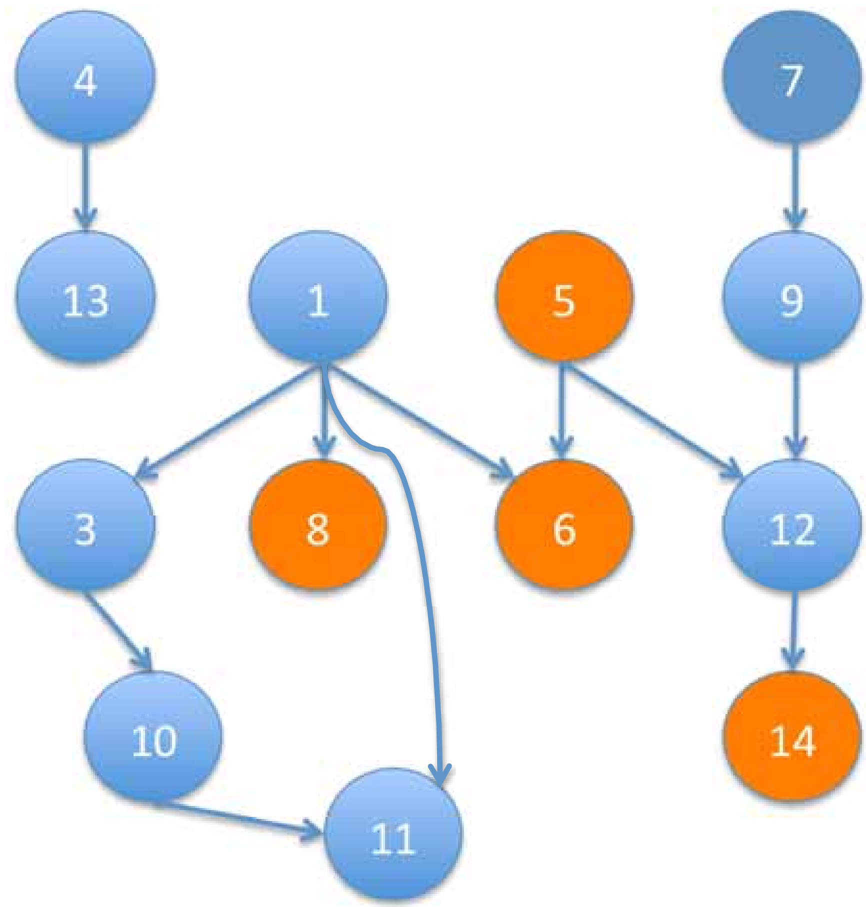



Assume intermittent fault affects nodes 5, 6, 7, 8.

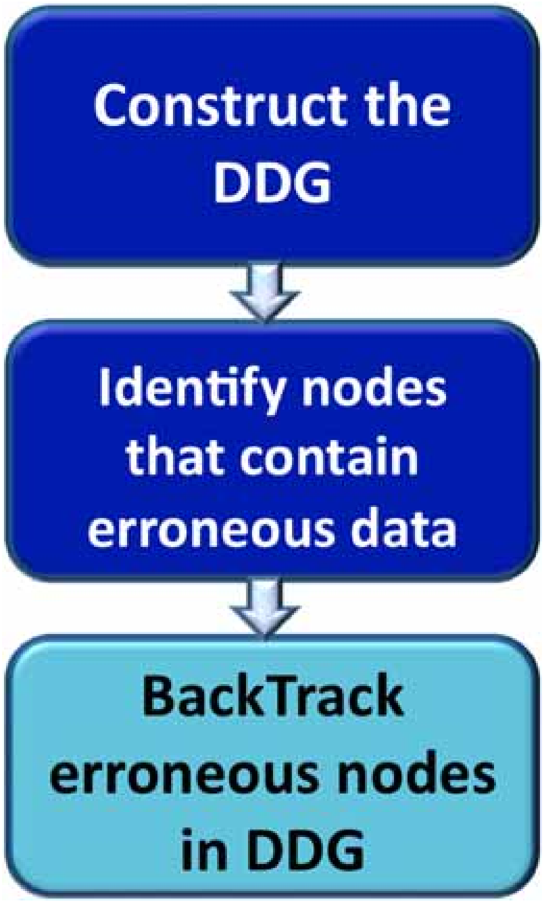
# Example

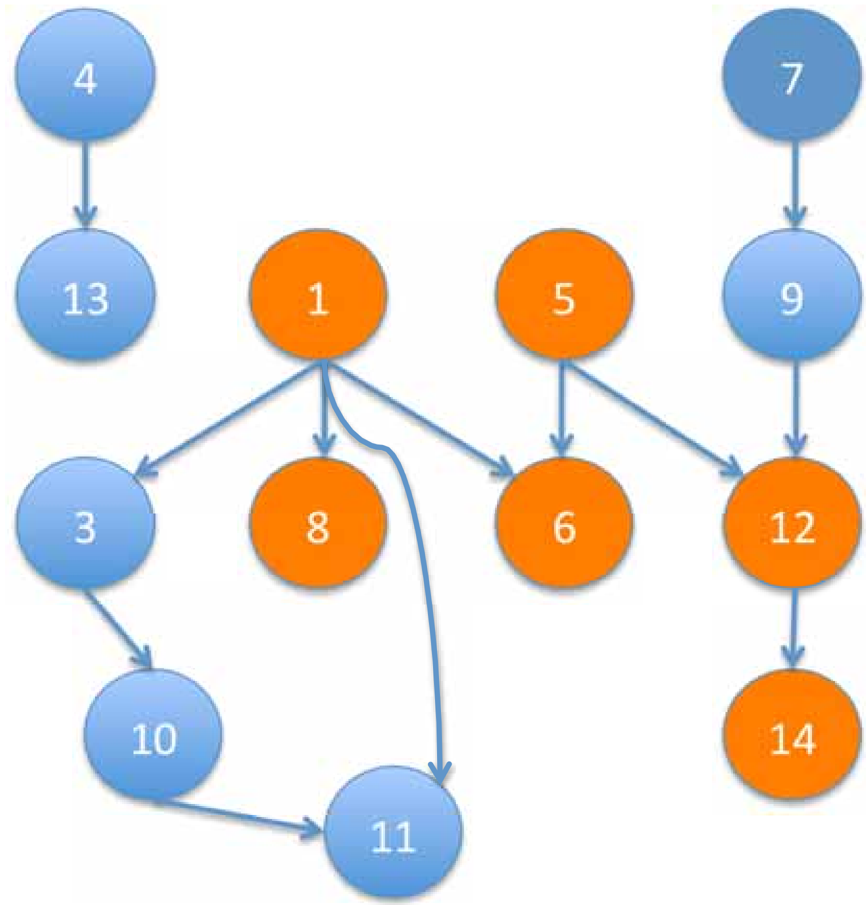



**Nodes 5, 6, 8, 14 → Strong clues**

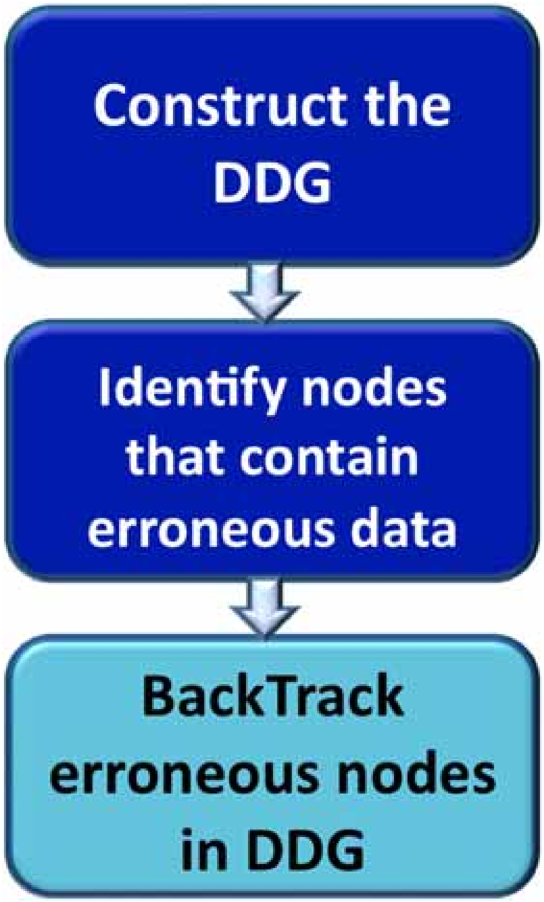
# Example

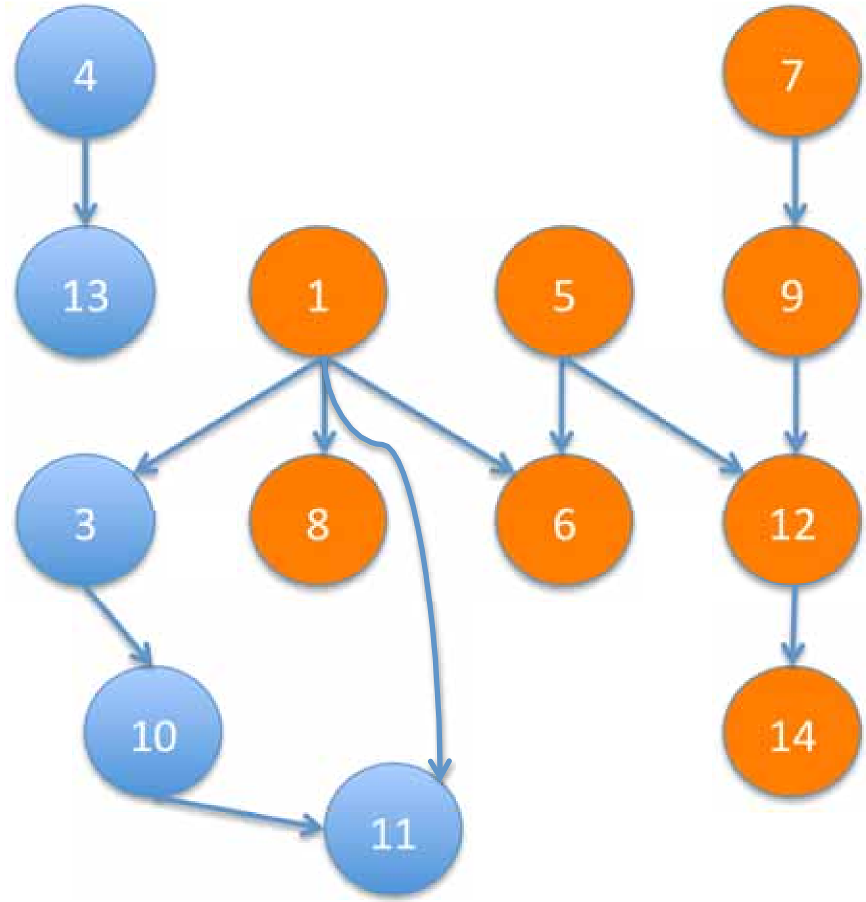



**Nodes 1, 12 → Weak clues**

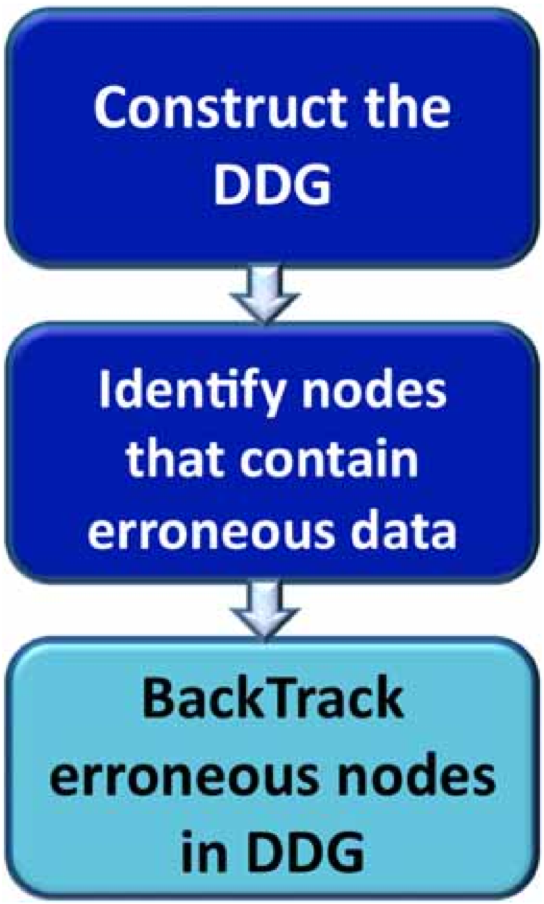
# Example

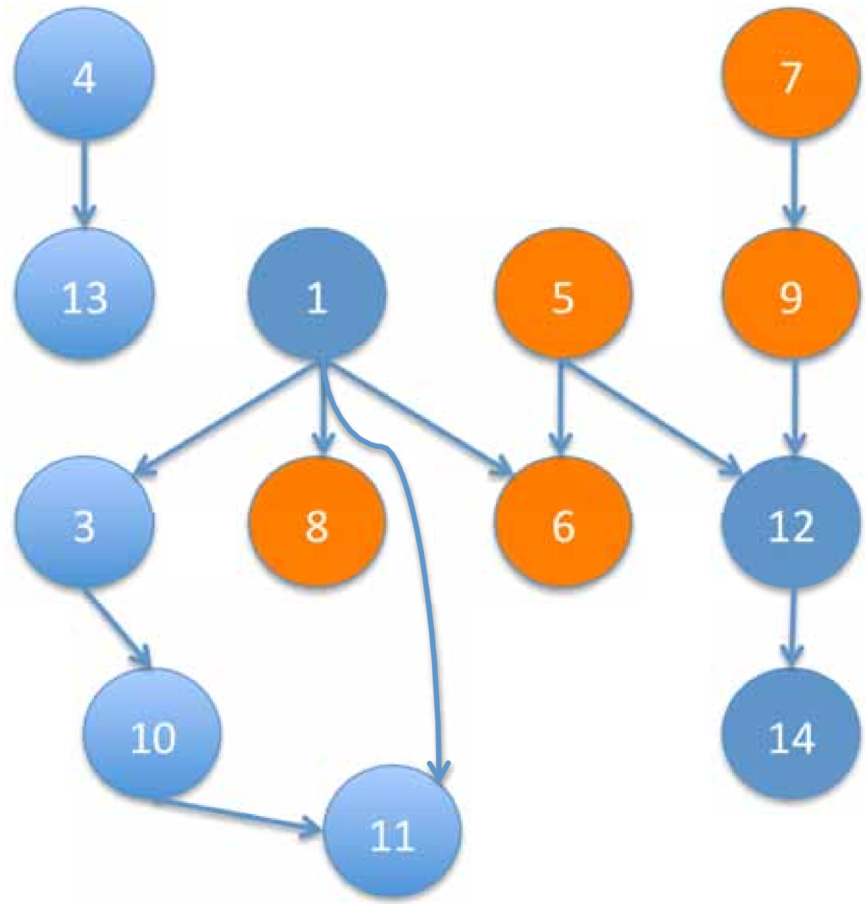



**Nodes 1, 7, 9, 12 → Weak clues**

# Example



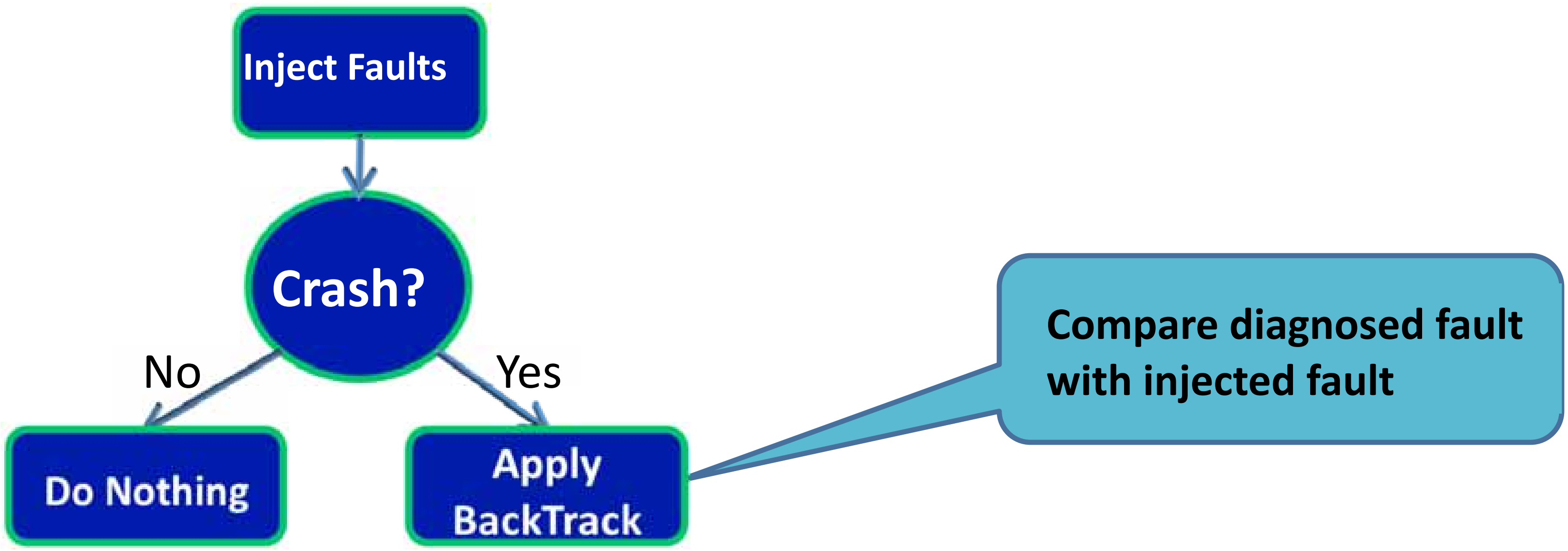

**Nodes 5, 6, 7, 8, 9 → Diagnosis solution**
**Nodes 5, 6, 7, 8 → Original fault**

# Outline

- Motivation
- Fault model
- Approach
- Results
- Conclusions

# Experimental Setup

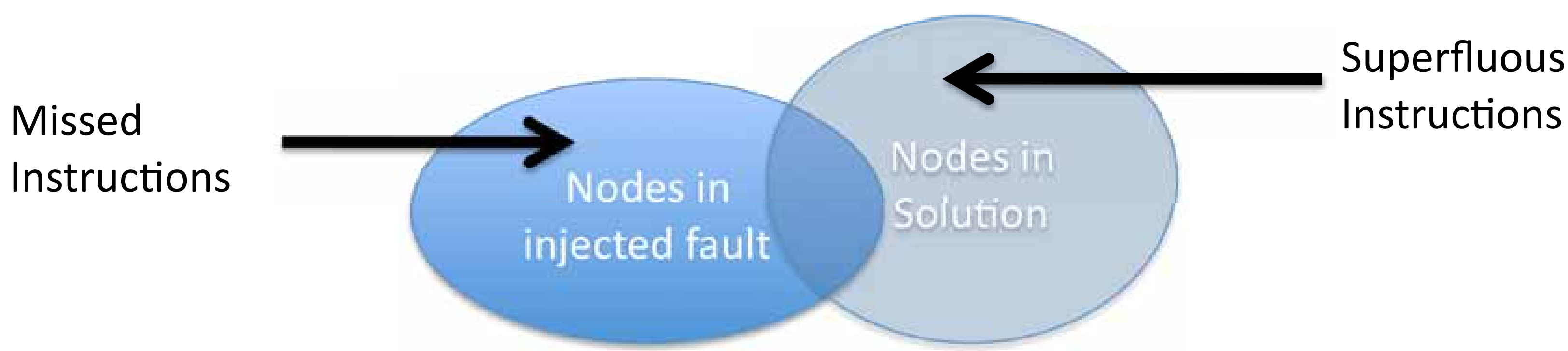
- **Siemens programs (100 – 1000 lines of code)**
- **Fault-injection in the SimpleScalar simulator**



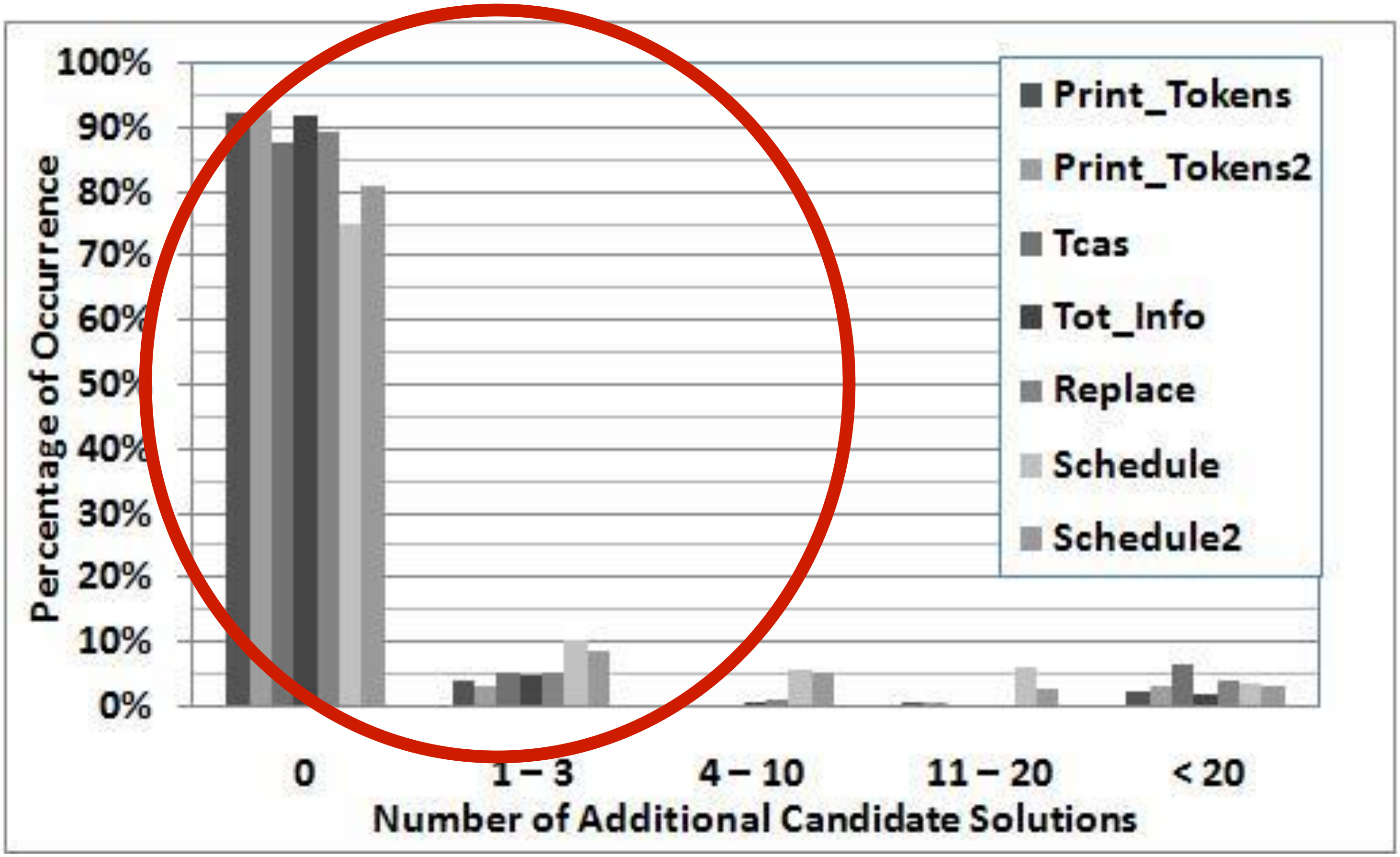
# BackTrack: Evaluation of Accuracy

- **Number of additional solutions found**
  - If multiple solutions, choose random one to compare with

- **Number of “missed” and “superfluous” instructions**
  - Missed instructions = Fault nodes – Solution nodes
  - Superfluous instructions = Solution nodes – Fault nodes

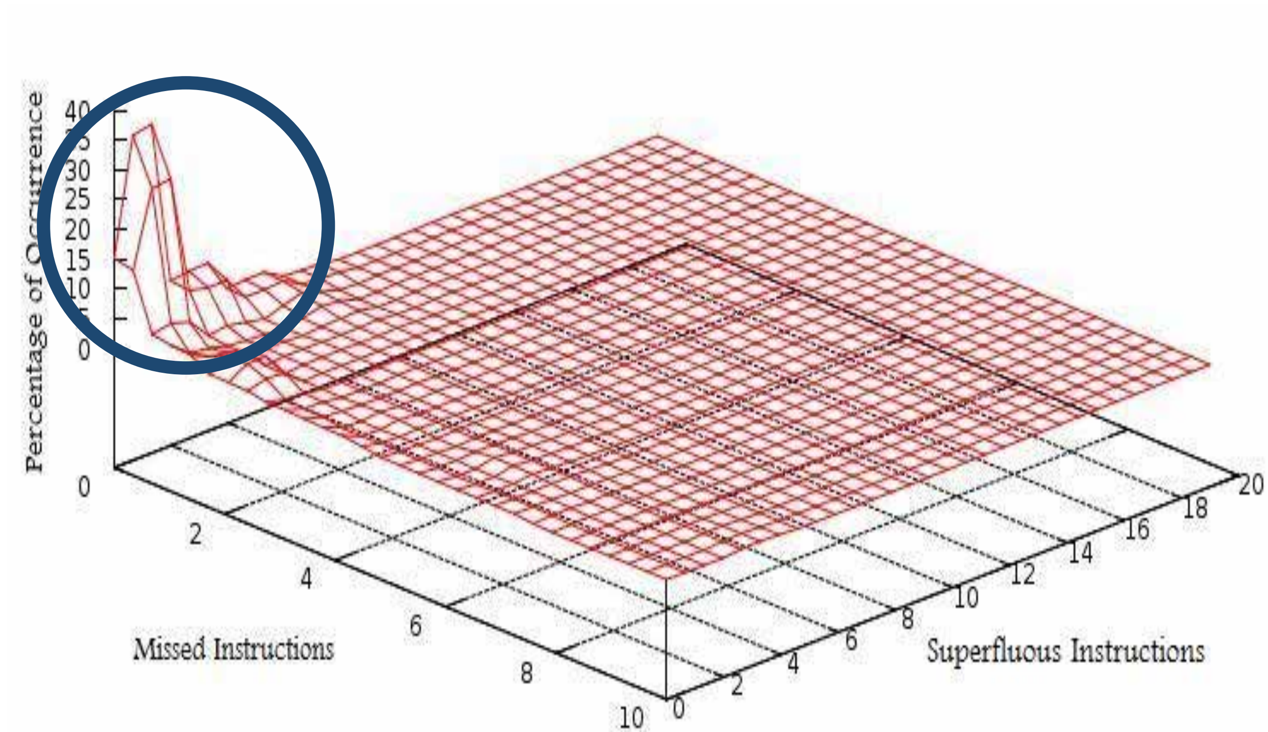


# BackTrack: Additional Solutions




- ❑ 87% of the diagnosed faults have NO additional solutions.
- ❑ 96% of the diagnosed faults have fewer than SEVEN solutions

# BackTrack : Superfluous & Missed Instructions




- ❑ **74% of the solutions have at most two missed or two superfluous instructions**

# Outline

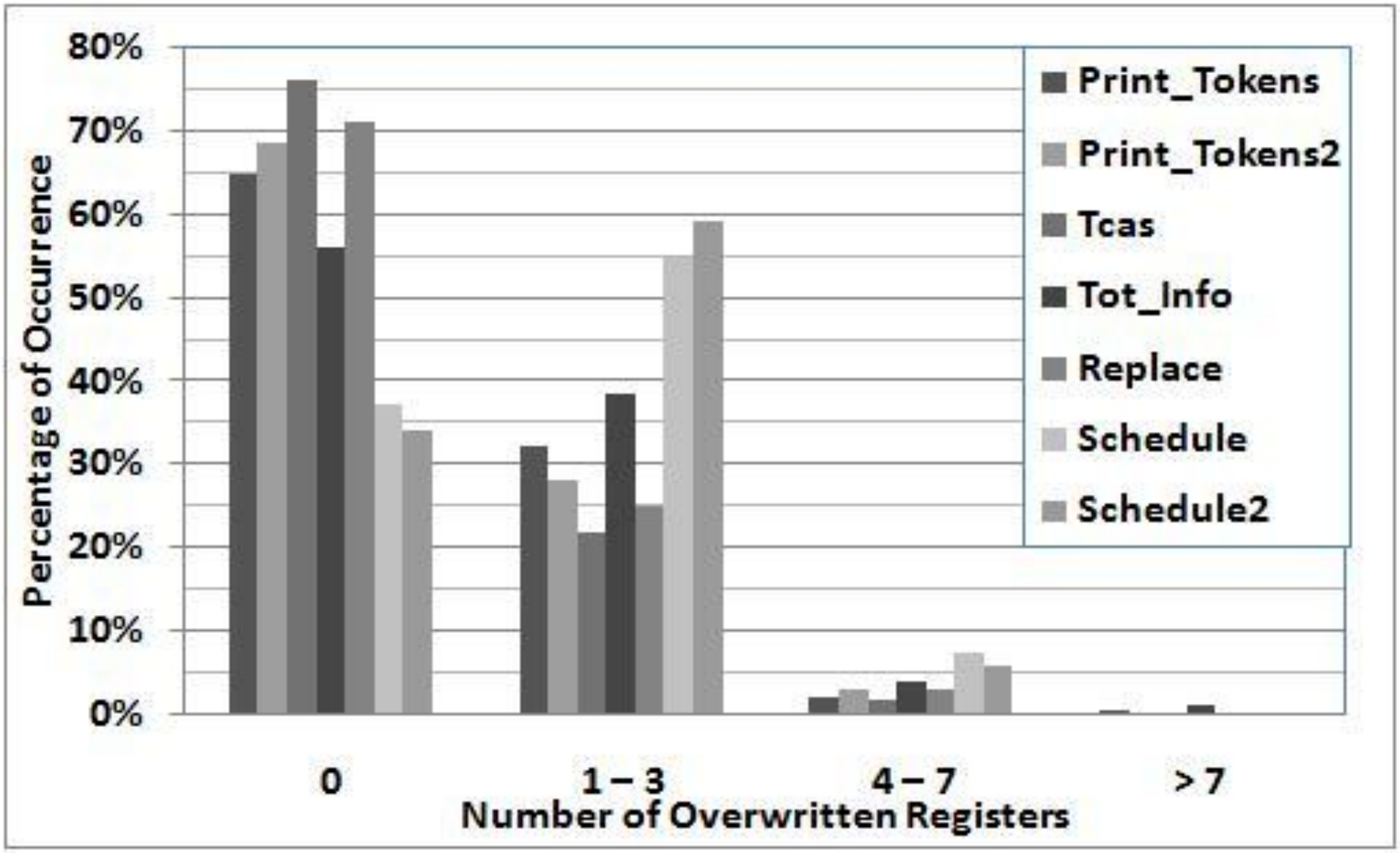
- Motivation
- Fault model
- Approach
- Results
- Conclusions

# BackTrack: Conclusions

- **Software-only techniques can be effective for diagnosing intermittent hardware faults**
- **Can diagnose most intermittent faults uniquely to within two instructions**
- **Future Work**
  - Isolate defective units based on statistical analysis
  - Improve the accuracy of the diagnosis procedure
  - Consider larger programs and other fault types

# BACKUP SLIDES

# Results: Limits of Software Diagnosis




- ❑ Few registers are overwritten before a failure → The data corrupted in a program by an intermittent fault is mostly intact

# Implications for Diagnosis

- Of the intermittent faults that are non-benign, **95%** result in a **program crash**
  - **Focus on crash-causing errors for diagnosis**

- **More than 90%** of the faults cause program to crash **within 500 instructions** of the fault
  - Fault propagation is limited in programs
  - Most of the fault's evidence is intact after crash