Formal Diagnosis of Hardware Transient Errors in Programs

Layali Rashid, Karthik Pattabiraman and
Sathish Gopalakrishnan

The Electrical and Computer Engineering Department
The University of British Columbia

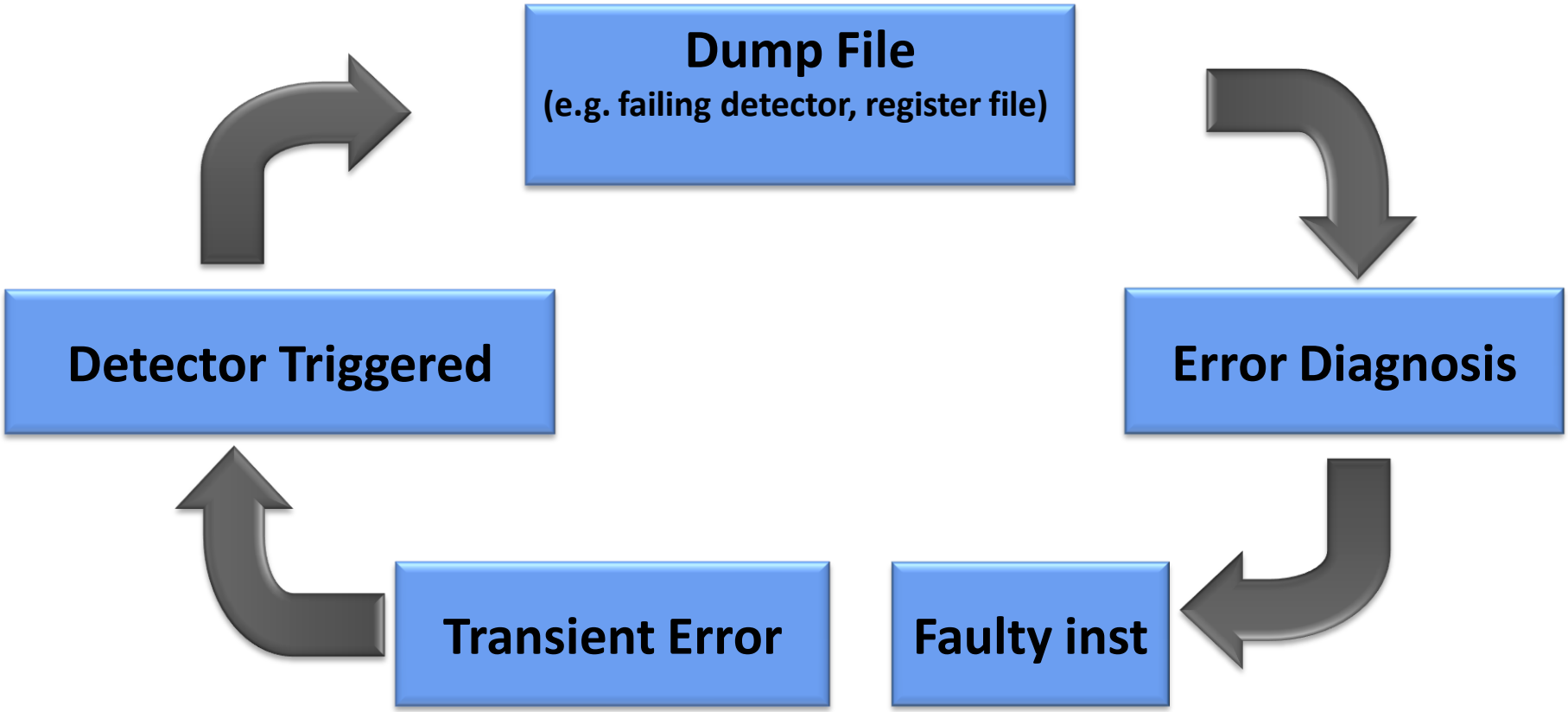
Contributions

- **Software-driven diagnosis of hardware transient errors**
  - Diagnosis: “isolate the first affected instruction”
    - Program-level analysis
  - Guarantees on the diagnosis
    - Completeness
    - Accuracy

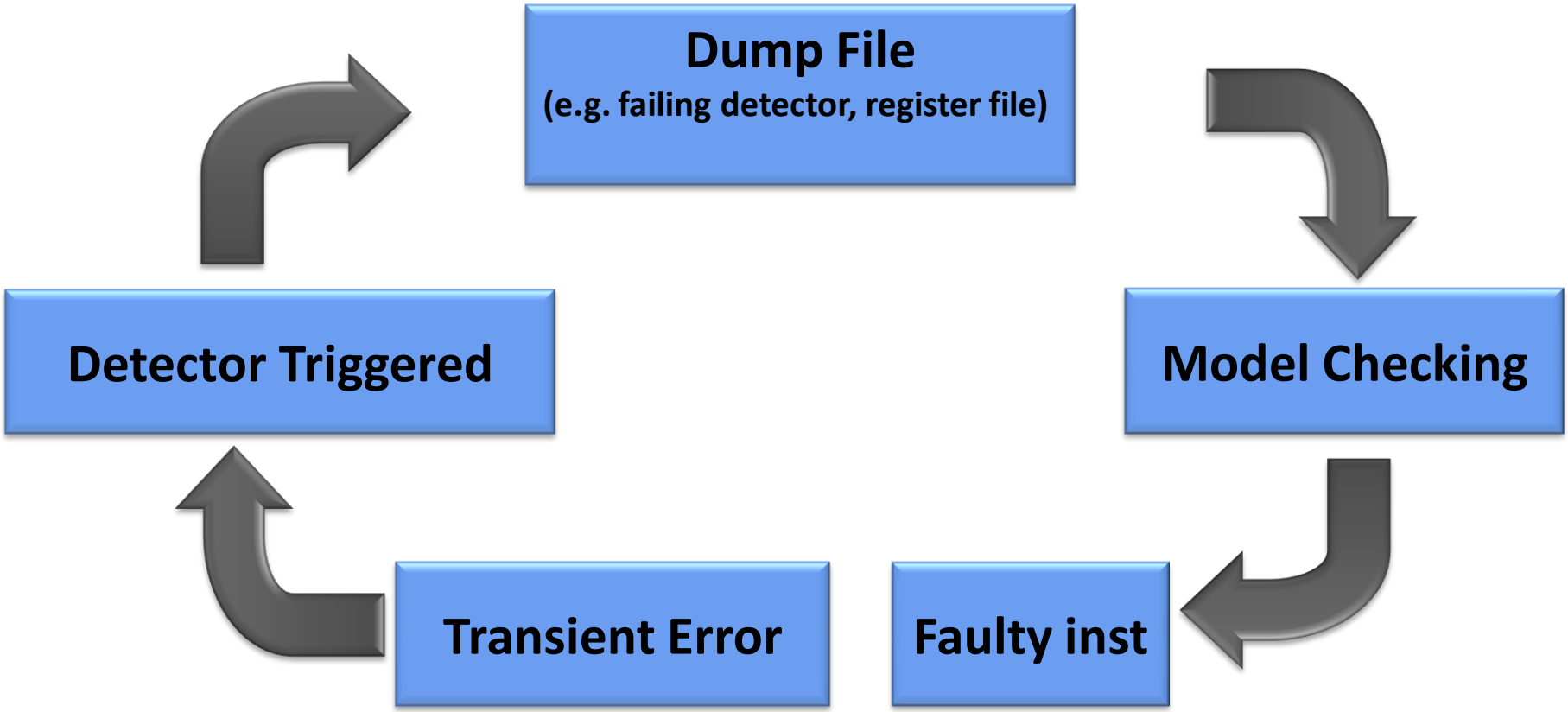
Why Software-Driven Diagnosis?

- No expensive hardware modifications.
- Minimal software instrumentation.
- Diagnose faults which manifest at the program-level only.
- Direct access to the affected device is not required.

Diagnosis Approach



Diagnosis Approach



Model Checking Using SymPLFIED

- **Formal model for analyzing programs[DSN'08]**
  - Evaluate the effect of transient hardware errors on programs.
- **Symbolic error propagation technique**
  - Represent errors using a single symbol (**err**) to avoid state space explosion.

Example: Factorial Program

```
1          movi $2, #1
2          read $1
3          mov $3, $1
4          movi $4, #1
5 loop:    setgt $5, $3, $4
6          beq  $5, #0, exit
7          mult $2, $2, $3
8          subi $3, $3, #1
9          assert($3 < $1 + 1)
10         beq  $0, #0, loop
11 exit:   prints "Factorial = "
12         print $2
```

Result variable

User input

Loops while $3 < $4

Error detector

Example: Error Propagation

```
1          movi $2, #1
2          read $1
3          mov $3, $1
4          movi $4, #1
5 loop:    setgt $5, $3, $4
6          beq  $5, #0, exit
7          mult $2, $2, $3
8          subi $3, $3, #1
9          assert($3 < $1 + 1)
10         beq  $0, #0, loop
11 exit:   prints "Factorial = "
12         print $2
```

$1 = 5

A transient fault, $3 = 13

Detector is triggered

Example: Error Propagation

```
1          movi $2, #1
2          read $1
3          mov $3, $1
4          movi $4, #1
5 loop:    setgt $5, $3, $4
6          beq  $5, #0, exit
7          mult $2, $2, $3
8          subi $3, $3, #1
9          assert($3 < $1 + 1)
10         beq  $0, #0, loop
11 exit:   prints "Factorial = "
12         print $2
```

$1 = 5

A transient fault, $3 = 13

Dump file:
Detector triggered
$1 = 5
$2 = 13
$3 = 12
$4 = 1
$5 = 1

Detector is triggered

Example: Error Diagnosis

```
1          movi $2, #1
2          read $1
3          mov $3, $1
4          movi $4, #1
5 loop:    setgt $5, $3, $4
6          beq  $5, #0, exit
7          mult $2, $2, $3
8          subi $3, $3, #1
9          assert($3 < $1 + 1)
10         beq  $0, #0, loop
11 exit:   prints "Factorial = "
12         print $2
```

A transient fault, $3 = err

True » Exit

False » Line 7

$2 = err

True » Line 10

False » Detector triggered

Example: Error Diagnosis

```
1        movi $2, #1
2        read $1
3        movi $3, $1
4        movi $4, #1
5 loop:  seq $5, $3, $1
6        beq $5, #0, exit
7        mul $2, $2, $3
8        sub $3, $3, #1
9        assert($3 < $1 + 1)
10       beq $0, #0, loop
11 exit: prints "Factorial = "
12       print $2
```

SymPLFIED's Solution
Instruction 3 Injected
Detector triggered
$1 = 5
$2 = err
$3 = err
$4 = 1
$5 = 1

Dump file:
Detector triggered
$1 = 5
$2 = 13
$3 = 12
$4 = 1
$5 = 1

Example: Error Diagnosis

```
1        movi $2, #1
2        read $1
3        movi $3, $1
4        movi $4, #1
5 loop:  mul $2, $2, $3
6        sub $1, $1, #1
7        movi $5, #1
8        sub $3, $3, #1
9        assert($3 < $1 + 1)
10       beq $0, #0, loop
11 exit: prints "Factorial = "
12       print $2
```

SymPLFIED's Solution

Instruction 3 Injected

$3 = err

$4 = 1

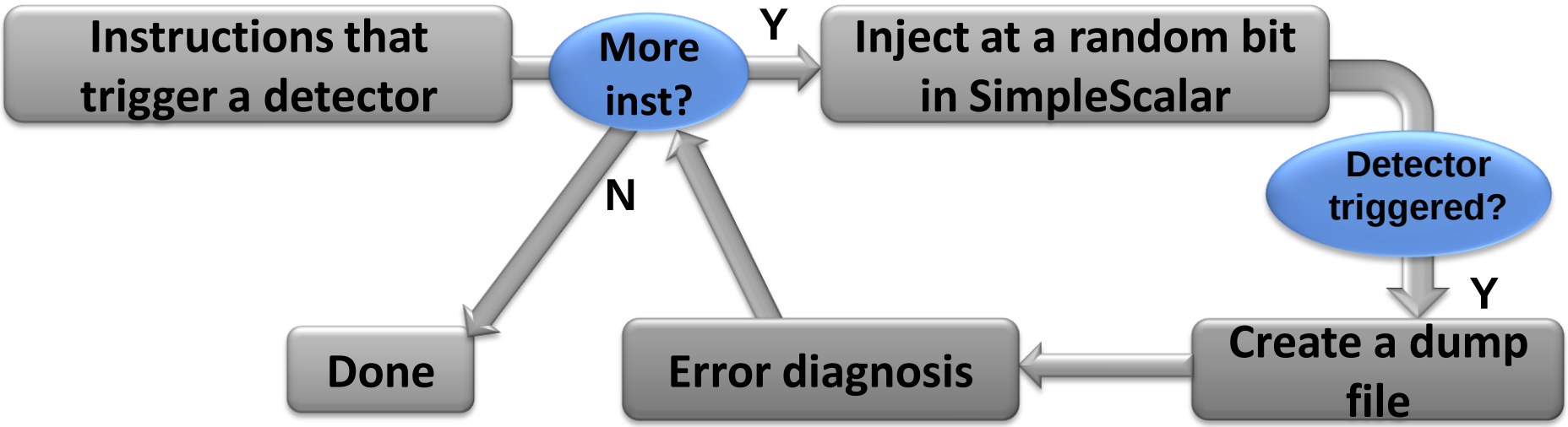
$5 = 1

Dump file:

The crash dump file can be used to identify the faulty instruction.

Experimental Methodology

- Enhance SymPLFIED to diagnose errors.
- Modify SimpleScalar simulator to inject faults.
- Evaluate for Matrix Multiply and Insertion Sort.



Results for Matrix Multiply

Number of detectors	1	4	6
Number of faults injected in SS	167	275	286
Number of faults detected in SS	74	135	150
Diagnosed faults (%)	100	77	80
Undiagnosed fault (%)	0	23	20

Results for Matrix Multiply (1)

Number of detectors	1	4	6
Number of faults injected in SS	167	275	286
Number of faults detected in SS	74	135	150
Diagnosed faults (%)	100	77	80
Undiagnosed fault (%)	0	23	20

- The proposed technique diagnoses 77%-100% of the detected errors for the matrix multiply program.
- The undiagnosed errors are implementation artifacts of the SymPLFIED tool.

Results for Matrix Multiply (2)

Number of detectors	1	4	6
Number of faults injected in SS	167	275	286
Number of faults detected in SS	74	135	150
Diagnosed faults (%)	100	77	80
Undiagnosed fault (%)	0	23	20

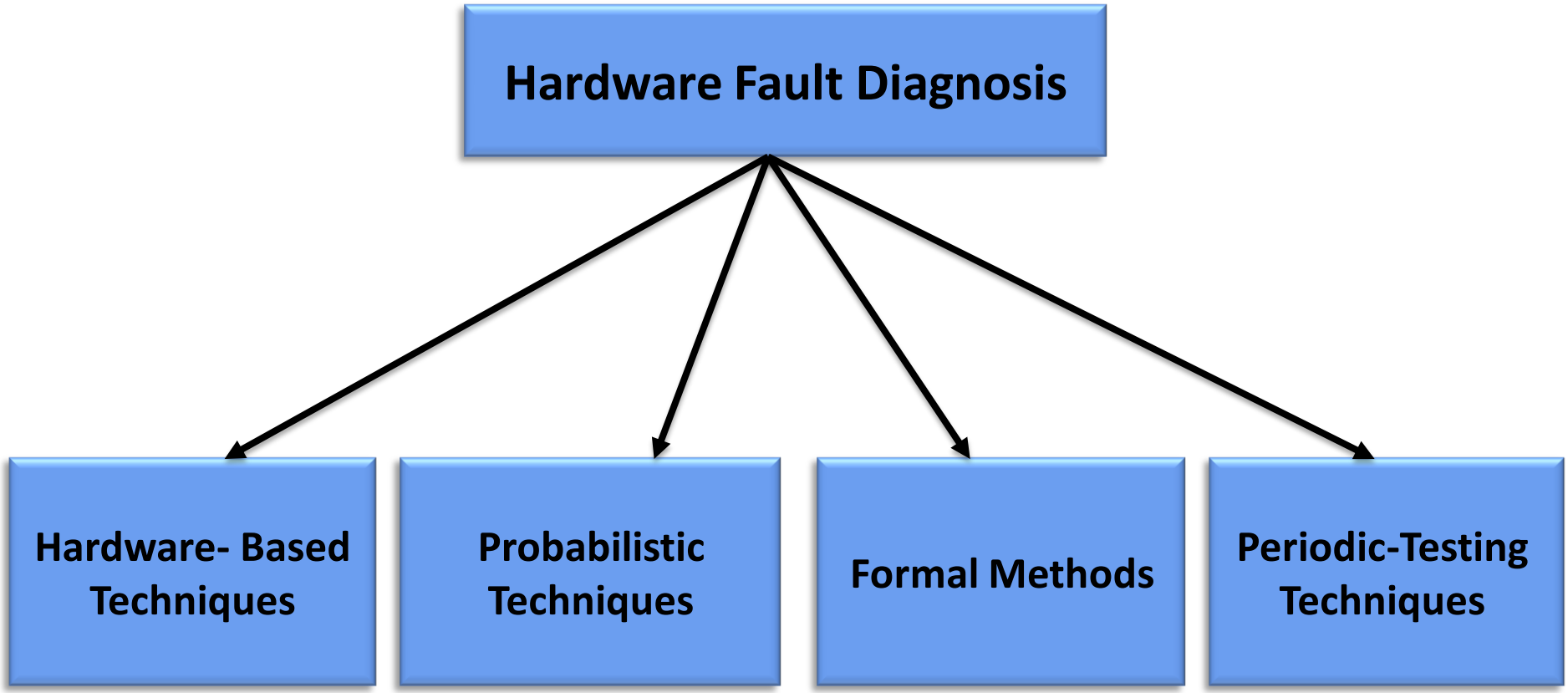
- The number of faults injected in SimpleScalar is proportional to the number of detectors.
- Adding more detectors increases the diagnosis accuracy.

Conclusions and Future Work

- **Software diagnosis of hardware faults is possible and can be automated using formal techniques**.
  - Our diagnosis method is able to diagnose significant number of errors using a few detectors.
- Future Work
  - Investigate improvements with limited hardware support.
  - Improve scalability using heuristics.
  - Extend to intermittent & permanent faults.

Backup Slides

Related Work



Results for Insertion Sort

Number of detectors	1	4	7
Number of faults injected in SS	11	165	198
Number of faults detected in SS	8	64	83
Diagnosed faults (%)	100	87	89
Undiagnosed fault (%)	0	13	11