Laboratoire
Informatique
Robotique
Microélectronique
Montpellier

Hardware Independent Evaluation of Computer based System Reliability

Maha Kooli

Giorgio Di Natale

Alberto Bosio

Pascal Benoit

Lionel Torres

SETS'15

19-03-15

Outline

- Motivation and Objectives
- State of the art
- Proposed Approach
- Experiments and Results
- Conclusion and Perspectives

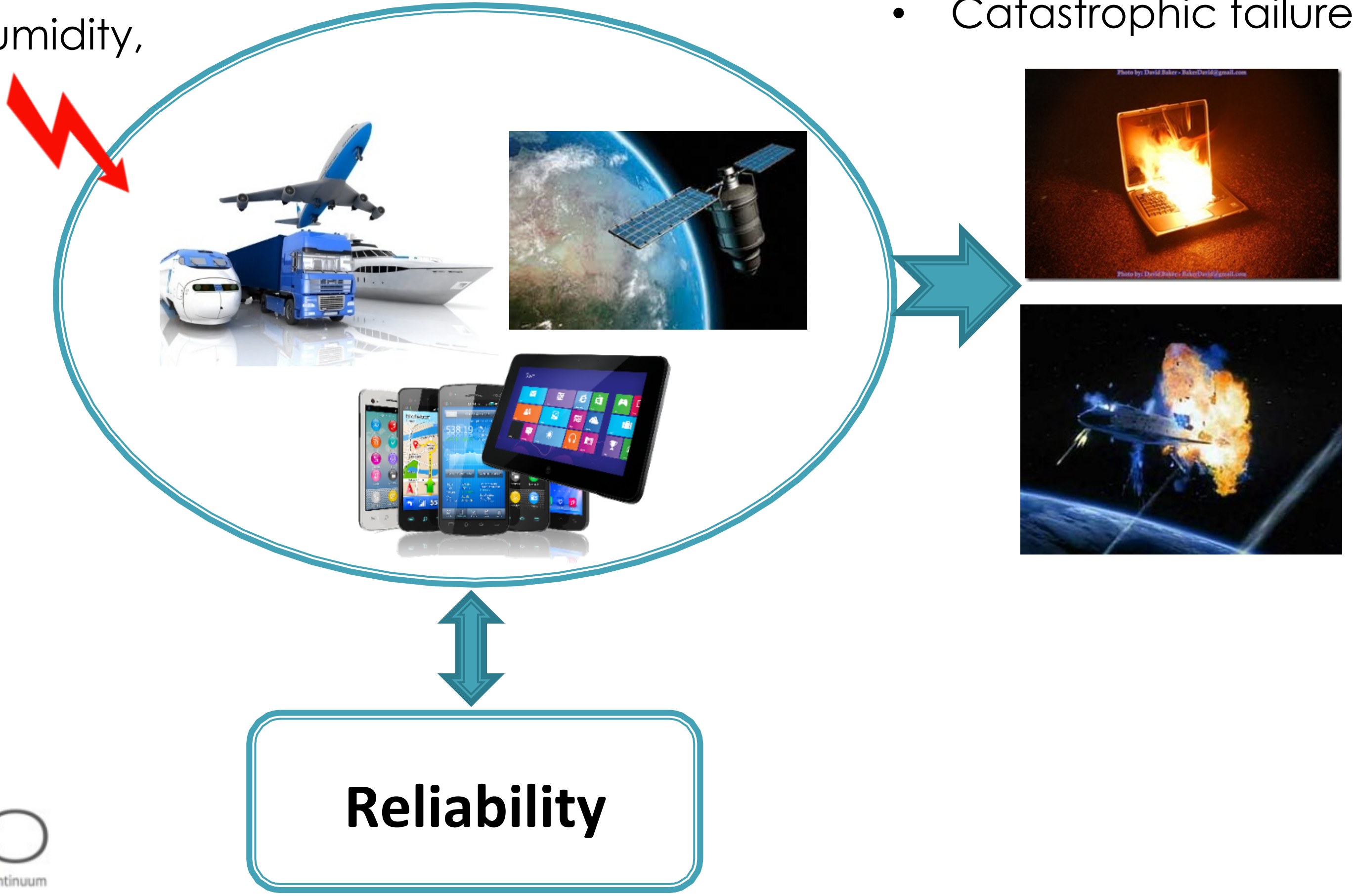
OUTLINE

- Motivation and Objectives
- State of the art
- Proposed Approach
- Experiments and Results
- Conclusion and Perspectives

Motivation

- Physical manufacturing defects
- Environmental perturbations (Temperature, Humidity, Radiations, …)

- Catastrophic failure

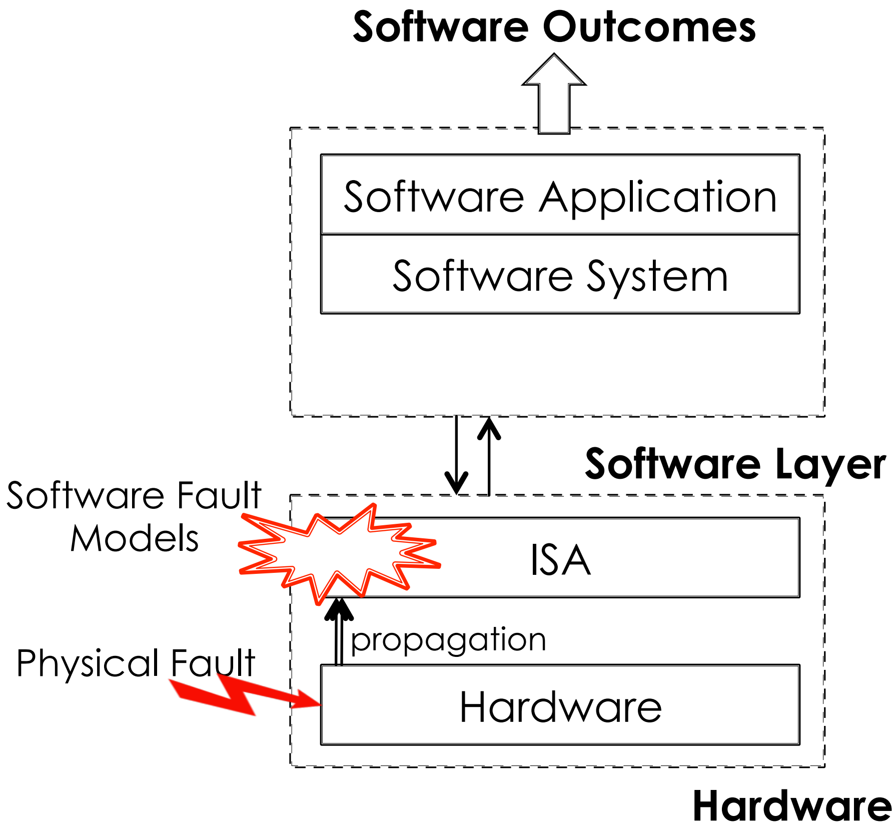


Reliability

Clereco

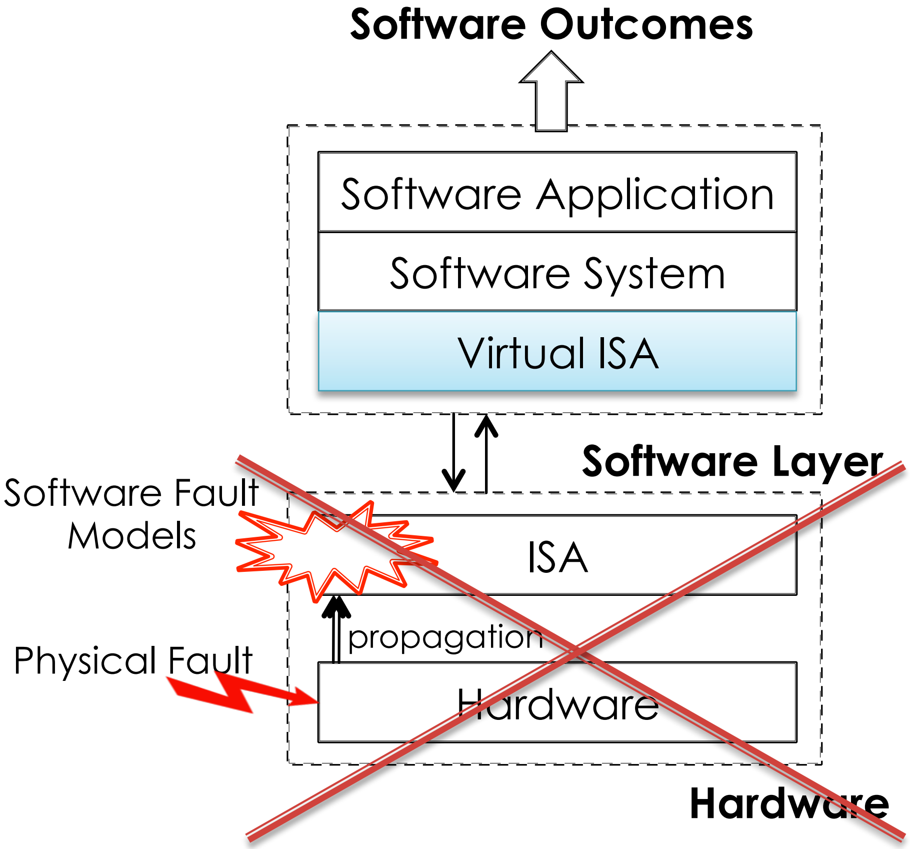
Cross-Layer Early Reliability Evaluation for the Computing cOntinuum

Objectives

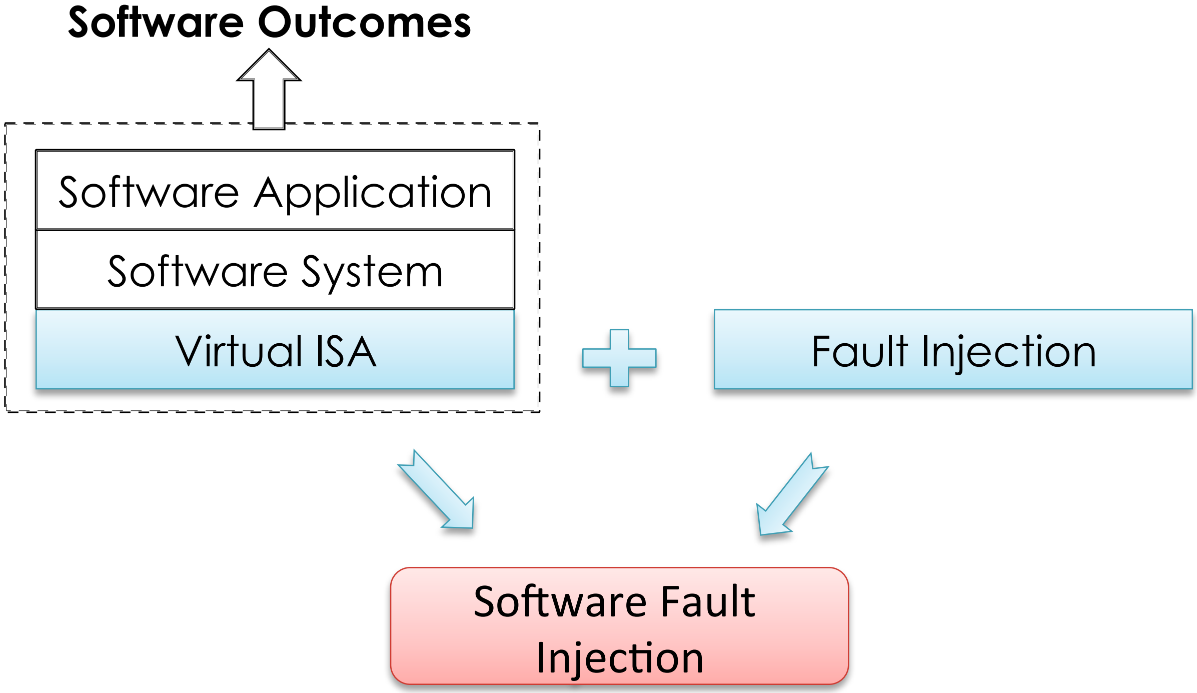


- **Objective**: Study the role of the software stack to evaluate the system reliability in an early design stage.

IDEA


Software Outcomes
Software Application
Software System
Virtual ISA
Software Layer
Software Fault Models
ISA
propagation
Physical Fault
Hardware
Hardware

IDEA


Software Outcomes
Software Application
Software System
Virtual ISA
Fault Injection
Software Fault Injection

Outline

- Motivation and Objectives
- State of the art
- Proposed Approach
- Experiments and Results
- Conclusion and Perspectives

STATE OF THE ART

System Reliability Evaluation

	Hardware Faults (Physical Fault: SEU, …)	**Software Faults** (Bugs, SW design faults, …)
Software	- Simulation-based Fault Injection: *Xception, Ferrari, …*	- Mutation Testing - Data Flow Graph - Control Flow Graph
Virtual Machine	- QEMU based Fault Injection - LLVM based Fault Injection: *LLFI, KULFI*	
Hardware	- Hardware Fault Injection: *Messaline, Mars, …* - Mutation Testing in the RTL level	

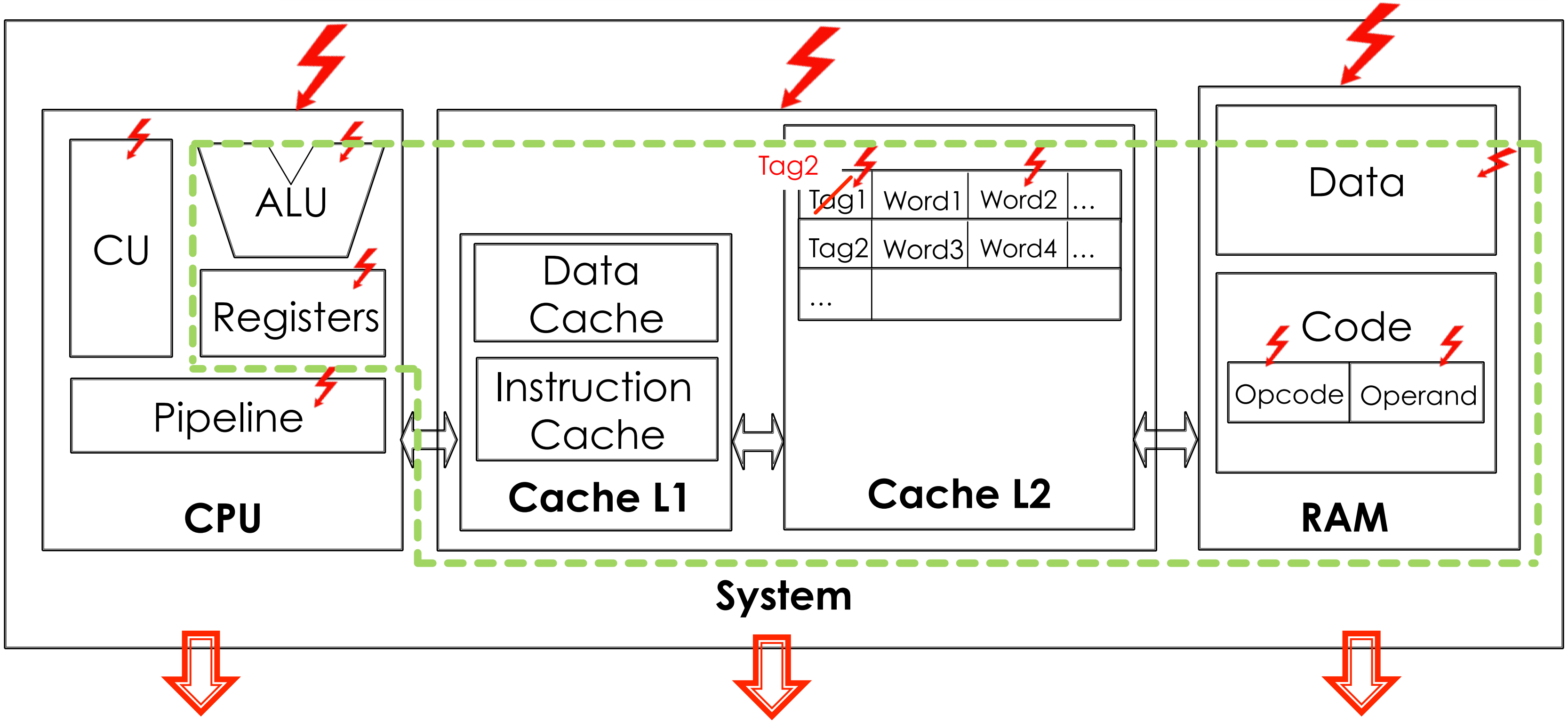
OUTLINE

- Motivation and Objectives
- State of the art
- Proposed Approach
- Experiments and Results
- Conclusion and Perspectives

Fault Models: Mutants

Fault Model	Description	Example
Wrong Data in an Operand	An operand of the VISA instruction changes its value	$A = B$ ↓ $A = B \oplus$ **Mask**
Instruction Replacement	An opcode in the VISA instruction is used in place of another	%A = add %B, %C ↓ %A = **sub** %B, %C

VALIDITY OF THE APPROACH

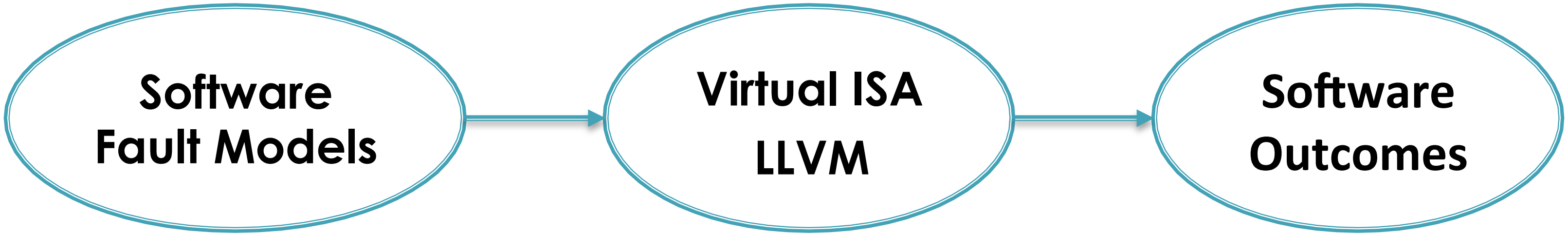


- Single Instruction Replacement
- Single Wrong Data in Operand

- Single Wrong Data in Operand
- Single Instruction Replacement
- Masked
- Multiple Wrong Data in Operand and/or Instruction Replacement

- Single Wrong Data in Operand
- Single Instruction Replacement

DESIGN AND IMPLEMENTATION

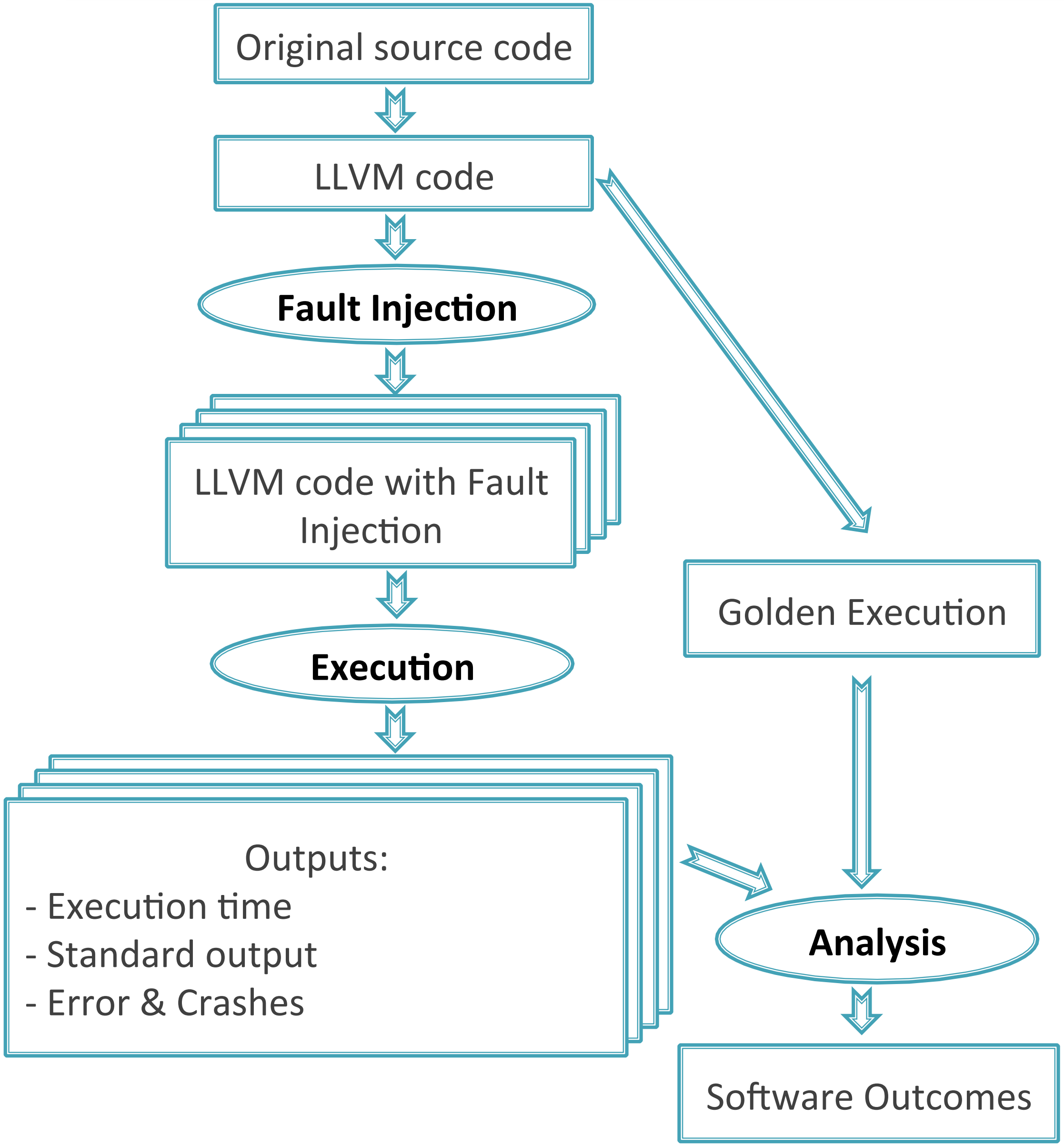


- A framework that performs complex analysis of software applications on different architectures.
- An assembler for a virtual hardware

Masked	The software produces correct results. All the faults are masked.
Silent Data Corruption (SDC)	The application outputs are different from the fault free outputs.
Detected	The fault has been detected by the application.
Crash / Unresponsive	The application stops working or it never stops.

```
; <label>:6
  %7 = load i32* %i
  %8 = load i32* %s
  %9 = add nsw i32 %8, %7
  store i32 %9, i32* %s
  br label %10
```

PROPOSED APPROACH



Original source code

LLVM code

Fault Injection

LLVM code with Fault Injection

Execution

Outputs:
- Execution time
- Standard output
- Error & Crashes

Golden Execution

Analysis

Software Outcomes

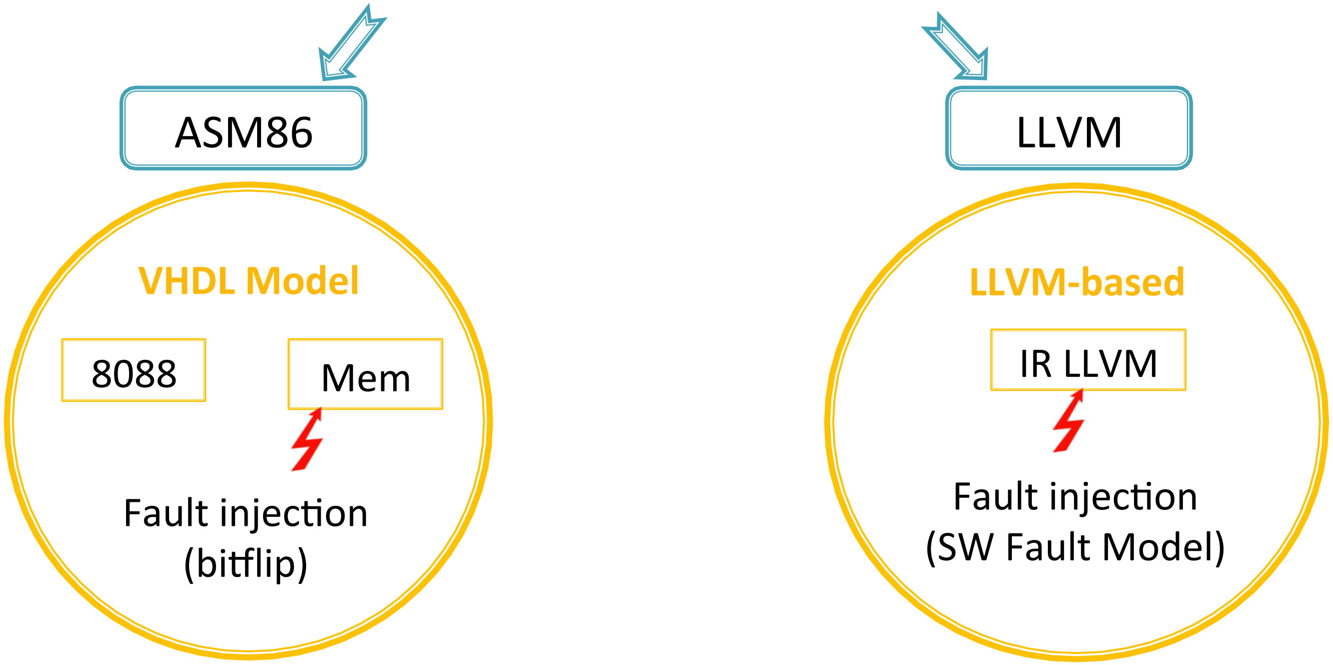
OUTLINE

- Motivation and Objectives
- State of the art
- Proposed Approach
- Experiments and Results
- Conclusion and Perspectives

EXPERIMENTS

C Source Code:

- Matrix multiplication (10x10 integer array)
- Matrix multiplication with duplicated variables
- Matrix multiplication with triplicated variables



ASM86

LLVM

VHDL Model

8088

Mem

Fault injection
(bitflip)

LLVM-based

IR LLVM

Fault injection
(SW Fault Model)

Sample Number of Fault Injection

$$n = \frac{N}{1 + e^2 \times \frac{N-1}{t^2 \times p \times (1-p)}}$$

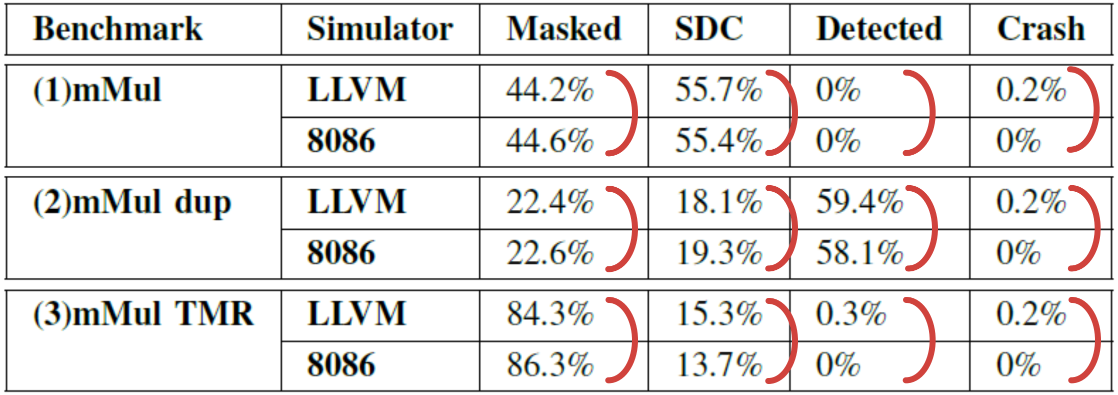
- **n**: number of faults to inject
- **N**: the number of all possible faults that can be injected
- **p**: an estimation of the value being searched
- **e**: margin of error
- **t**: expected confidence level

$$\lim_{N\to\infty}\left(\frac{N}{1 + e^2 \cdot \frac{N-1}{t^2 \cdot p \cdot (1-p)}}\right) \cong \frac{t^2}{4 \cdot e^2}$$

Pour $e = 1\%$, $t = 95\%$ ⟹ **10000 fault injections per program**

R. Leveugle, A. Calvez, P. Maistri, and P. Vanhauwaert, "Statistical fault injection: Quantified error and confidence," in Proceedings of the Conference on Design, Automation and Test in Europe, ser. DATE '09, 2009, pp. 502–506.

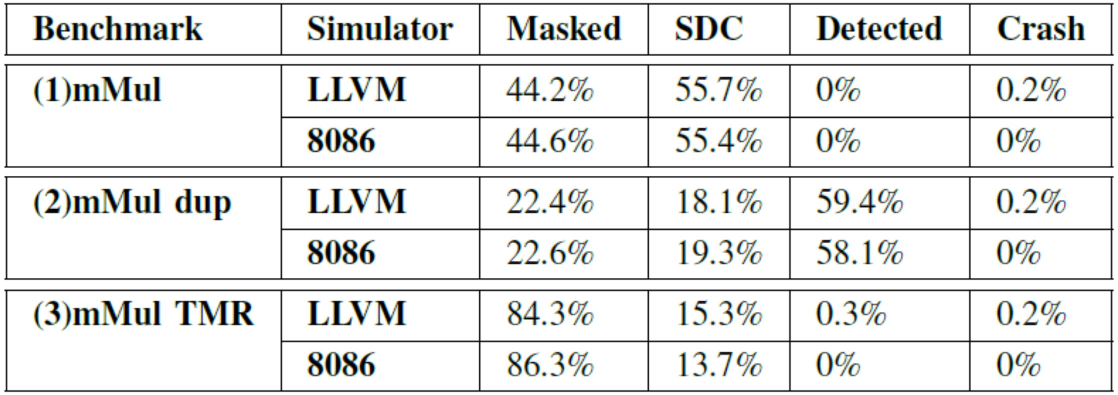
SIMULATION RESULTS



Benchmark	Simulator	Masked	SDC	Detected	Crash
(1)mMul	LLVM	44.2%	55.7%	0%	0.2%
	8086	44.6%	55.4%	0%	0%
(2)mMul dup	LLVM	22.4%	18.1%	59.4%	0.2%
	8086	22.6%	19.3%	58.1%	0%
(3)mMul TMR	LLVM	84.3%	15.3%	0.3%	0.2%
	8086	86.3%	13.7%	0%	0%

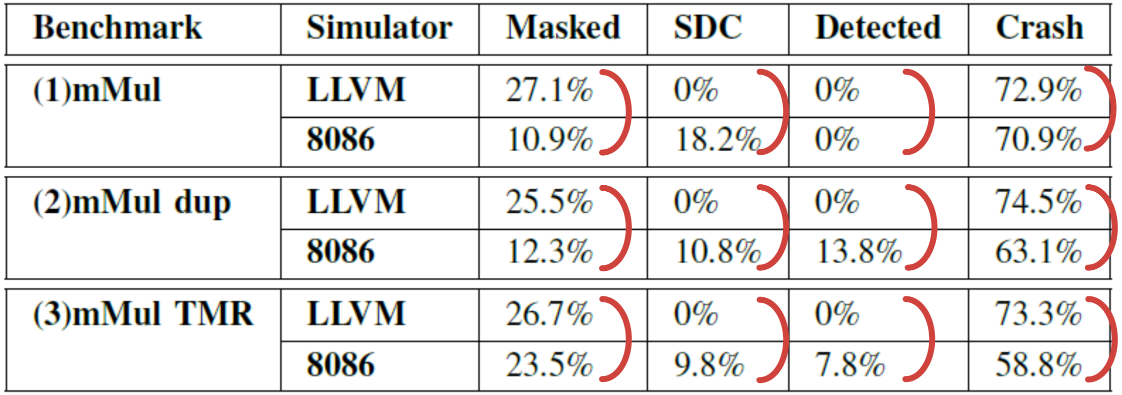
Results of simulations with a single transient fault injection in **data**

Simulation Results



Benchmark	Simulator	Masked	SDC	Detected	Crash
(1)mMul	LLVM	44.2%	55.7%	0%	0.2%
	8086	44.6%	55.4%	0%	0%
(2)mMul dup	LLVM	22.4%	18.1%	59.4%	0.2%
	8086	22.6%	19.3%	58.1%	0%
(3)mMul TMR	LLVM	84.3%	15.3%	0.3%	0.2%
	8086	86.3%	13.7%	0%	0%

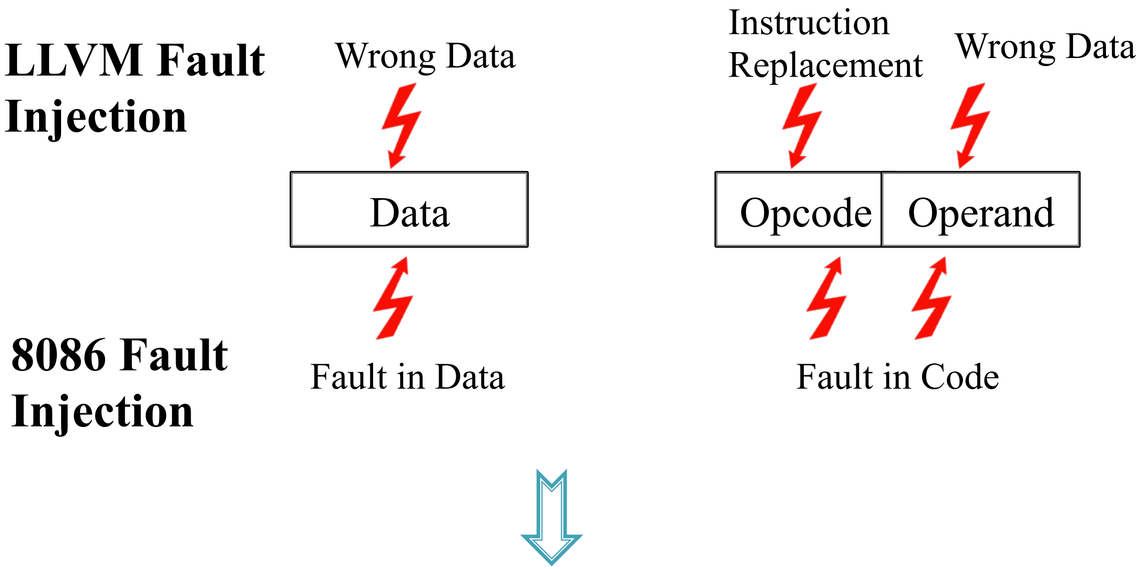
Results of simulations with a single transient fault injection in **data**



Benchmark	Simulator	Masked	SDC	Detected	Crash
(1)mMul	LLVM	27.1%	0%	0%	72.9%
	8086	10.9%	18.2%	0%	70.9%
(2)mMul dup	LLVM	25.5%	0%	0%	74.5%
	8086	12.3%	10.8%	13.8%	63.1%
(3)mMul TMR	LLVM	26.7%	0%	0%	73.3%
	8086	23.5%	9.8%	7.8%	58.8%

Results of simulations with a single transient fault injection in **opcode**

FINAL RELIABILITY EVALUATION



LLVM Fault Injection

Wrong Data

Data

Instruction Replacement

Wrong Data

Opcode

Operand

8086 Fault Injection

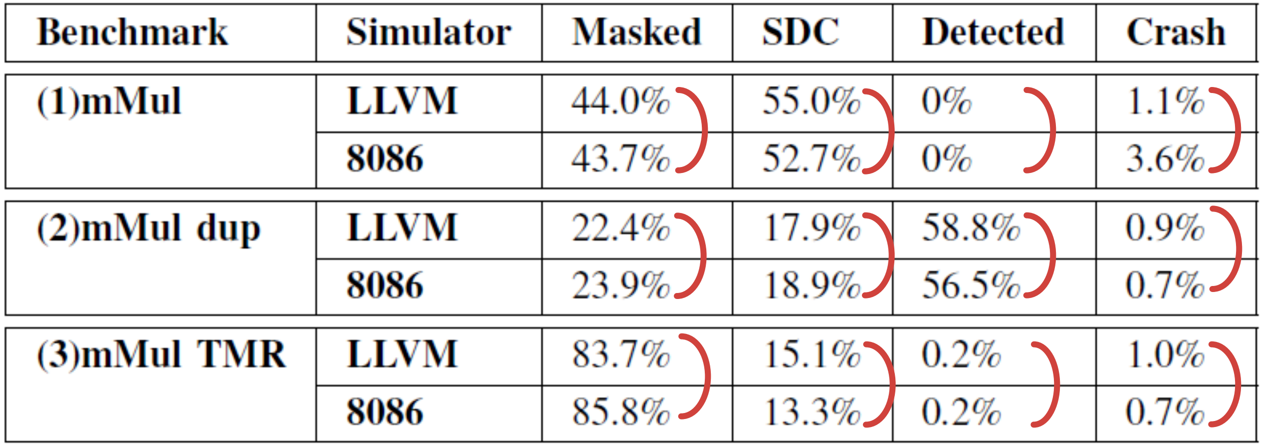
Fault in Data

Fault in Code

$$Outcome^{LLVM} = \left(\frac{Out^{WD} * (D+OP) + Out^{IR} * OC}{D+OC+OP} \right)$$

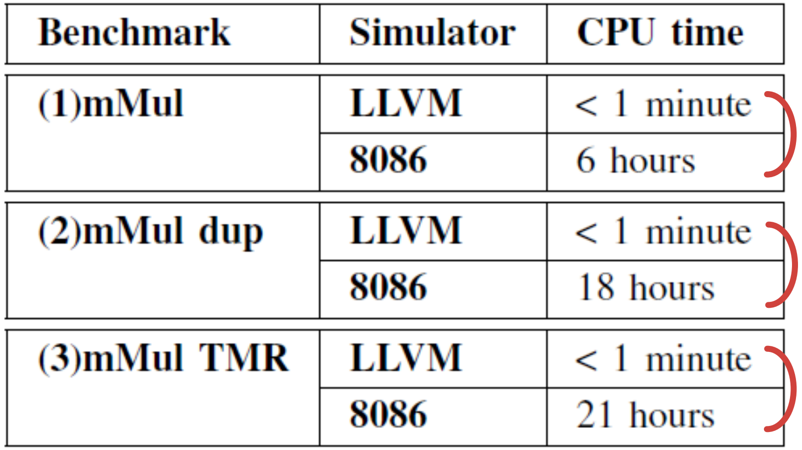
$$Outcome^{8086} = \left(\frac{Out^{Data} * D + Out^{Code} * (OC+OP)}{D+OC+OP} \right)$$

Final Reliability Evaluation



Benchmark	Simulator	Masked	SDC	Detected	Crash
(1)mMul	LLVM	44.0%	55.0%	0%	1.1%
	8086	43.7%	52.7%	0%	3.6%
(2)mMul dup	LLVM	22.4%	17.9%	58.8%	0.9%
	8086	23.9%	18.9%	56.5%	0.7%
(3)mMul TMR	LLVM	83.7%	15.1%	0.2%	1.0%
	8086	85.8%	13.3%	0.2%	0.7%

Evaluation of the overall system reliability



Benchmark	Simulator	CPU time
(1)mMul	LLVM	< 1 minute
	8086	6 hours
(2)mMul dup	LLVM	< 1 minute
	8086	18 hours
(3)mMul TMR	LLVM	< 1 minute
	8086	21 hours

Simulation Time

OUTLINE

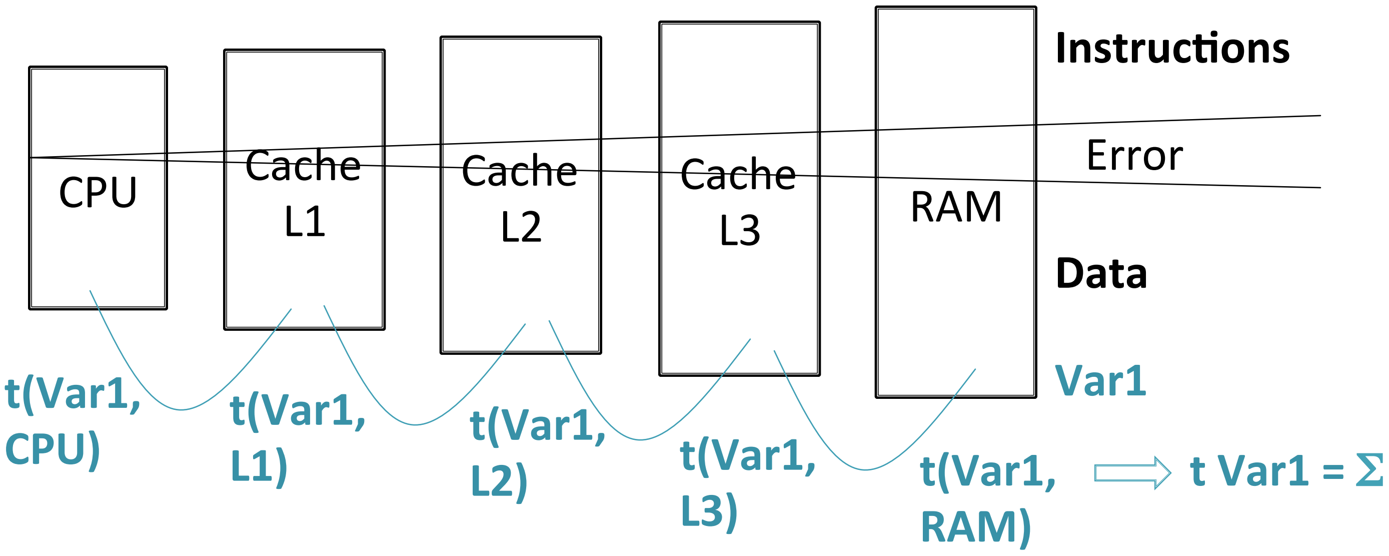
- Motivation and Objectives
- State of the art
- Proposed Approach
- Experiments and Results
- Conclusion and Perspectives

Conclusion

- LLVM based Fault Injection tool independent of the hardware architecture
- High abstraction of the fault models
- Efficient approach in term of reliability evaluation and simulation time

Perspectives

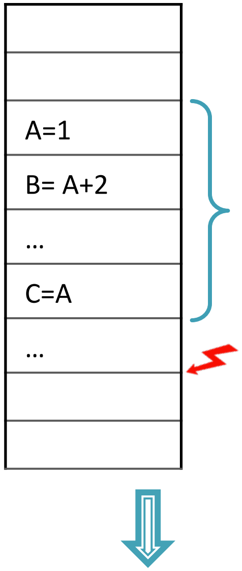
Residence of Variables



PERSPECTIVES

SIMULATION NUMBER REDUCTION

1. **Life Time of Variable**



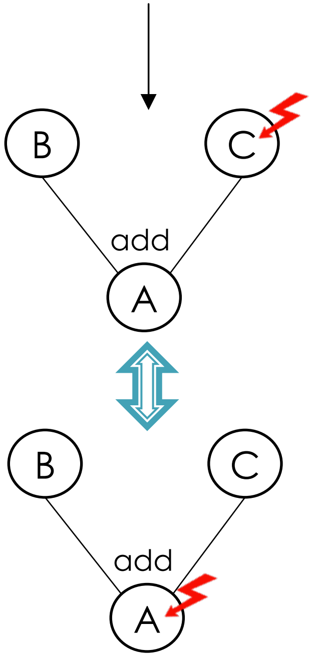
Life Time of A

Fault on A

Fault masked

2. **Fault Equivalence:** Data Dependency Graph

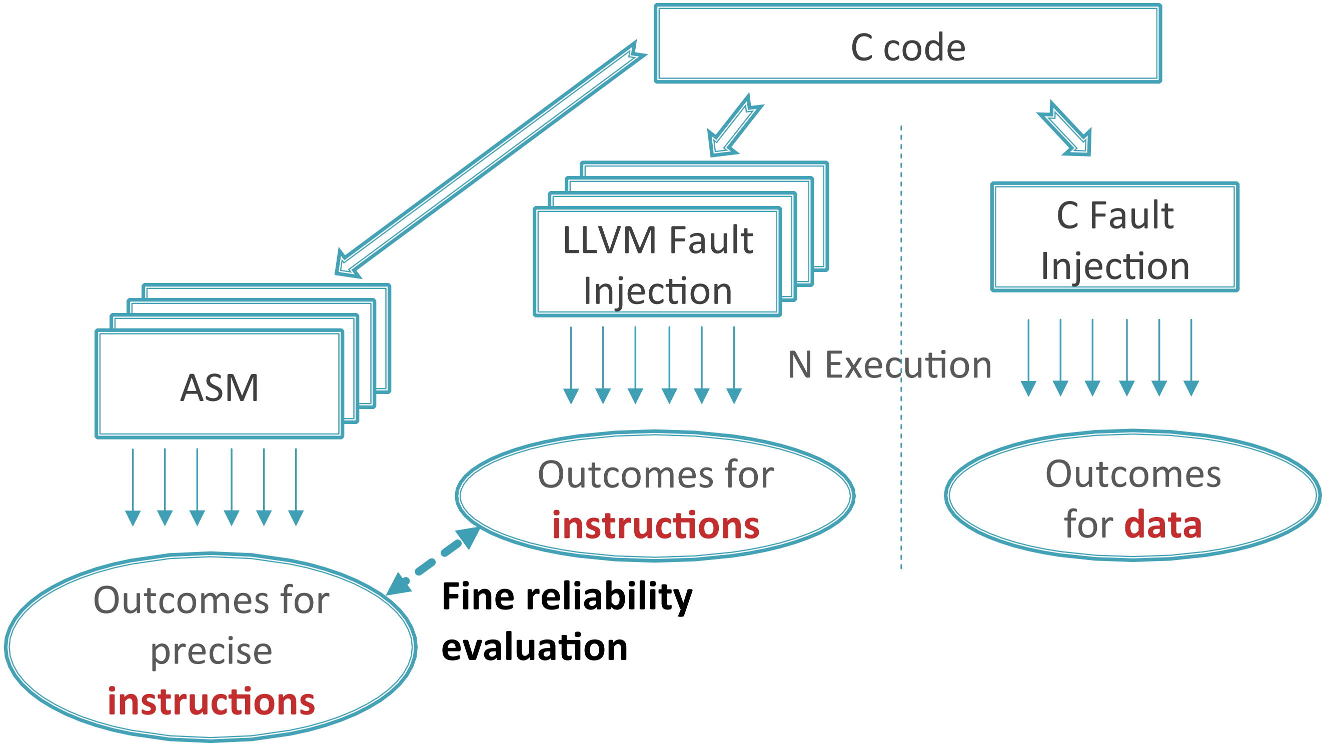
A = B + C



Thanks for your attention

Questions?

PERSPECTIVES



Fine reliability evaluation

PROPOSED APPROACH

```
; <label>:6
  %7 = load i32* %i
  %8 = load i32* %s
  %9 = add nsw i32 %8, %7
  store i32 %9, i32* %s
  br label %10
```

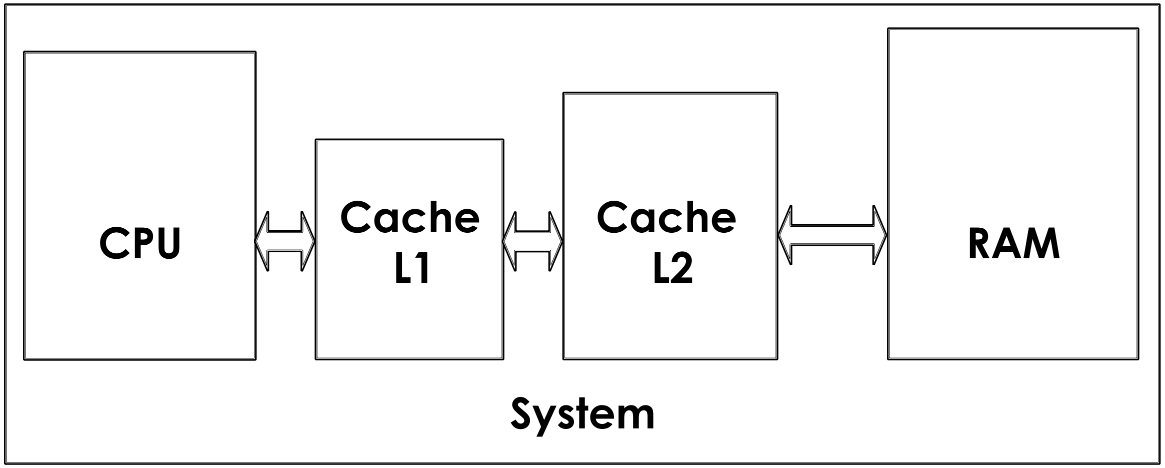
```
; <label>:6
  %7 = load i32* %i
  %7FI = xor i32 %7, 8
  %8 = load i32* %s
  %9 = add nsw i32 %8, %7FI
  store i32 %9, i32* %s
  br label %10
```

Wrong Data in an Operand

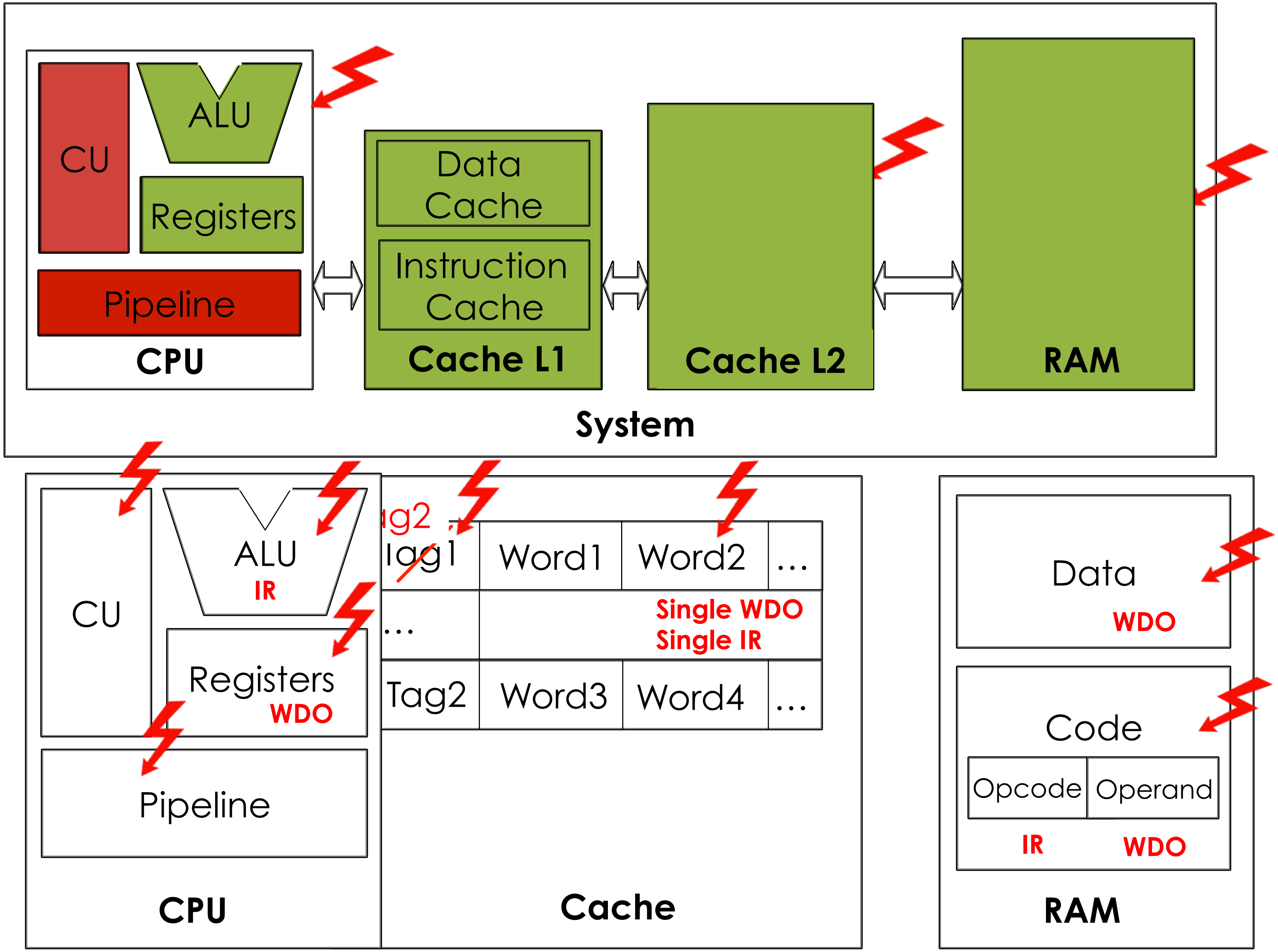
```
; <label>:6
  %7 = load i32* %i
  %8 = load i32* %s
  %9 = sub nsw i32 %8, %7
  store i32 %9, i32* %s
  br label %10
```

Instruction Replacement

Validity of the Approach



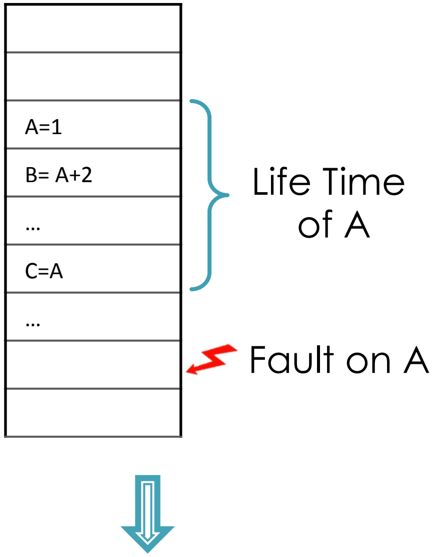
Validity of the Approach



Perspectives

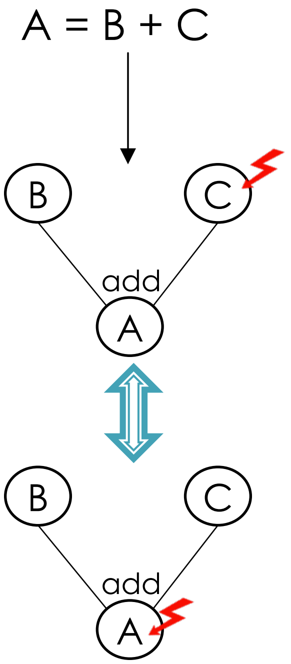
Simulation Number Reduction

1. Life Time of Variable



Fault masked

2.1. Fault Equivalence: Data Dependency Graph



2.2. Fault Equivalence: Instruction Analysis

Statistics (%) of Instruction Replacement

	add	mul	call	br	store	Invalid
add	25,0%	15,7%	1,7%	0,5%	7,3%	49,8%

Outcome of 1 simulation	Mask-ed	SDC	crash			
Number of required simulation	1	All	1	1	1	1