IEEE/IFIP DSN 2020 Conference – June 29, 2020

Tutorial #1

Cross-Layer Soft-Error Resilience Analysis of Computing Systems

Alberto Bosio
École Centrale de Lyon, France

Stefano Di Carlo and **Alessandro Savino**
Politecnico di Torino, Italy

Dimitris Gizopoulos
University of Athens, Greece

Ramon Canal
Universitat Politècnica de Catalunya
Barcelona Supercomputing Center, Spain

IEEE/IFIP DSN 2020 Conference – June 29, 2020

Part #3

Resilience Assessment at the Microarchitecture Level

[Throughput and Accuracy]

Dimitris Gizopoulos – University of Athens, Greece

Computer Architecture Lab

http://cal.di.uoa.gr - http//www.di.uoa.gr/~dgizop

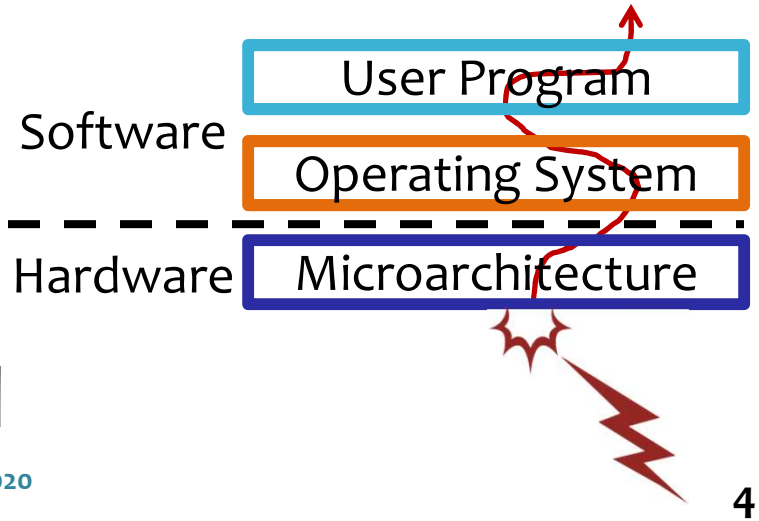
Material based on papers and tutorials from: DATE 2020, DSN 2019/2017, ISPASS 2019/2017/2016, IISWC 2019/2015, ICCD 2018, VTS 2018, ISCA 2017, MICRO 2017, VTS 2017/2016, ITC 2016, DFTS 2015, IOLTS 2017/2016/2014

Introduction to Microarchitecture Level Resilience/Reliability Assessment

Basic Concept – 1,2,3

- **Microarchitecture level reliability assessment**

1. **Employ early full system simulation model**
   - Cycle-accurate microarchitecture CPU model (*performance simulator*) and a full system stack (HW + SW + OS)
     - gem5, Marssx86, PTLsim, Flexus, GEMS
2. **Model faults** at the microarchitecture (hardware)
3. **Quantify and analyze the effects of faults** at the entire system stack: hardware + application + OS

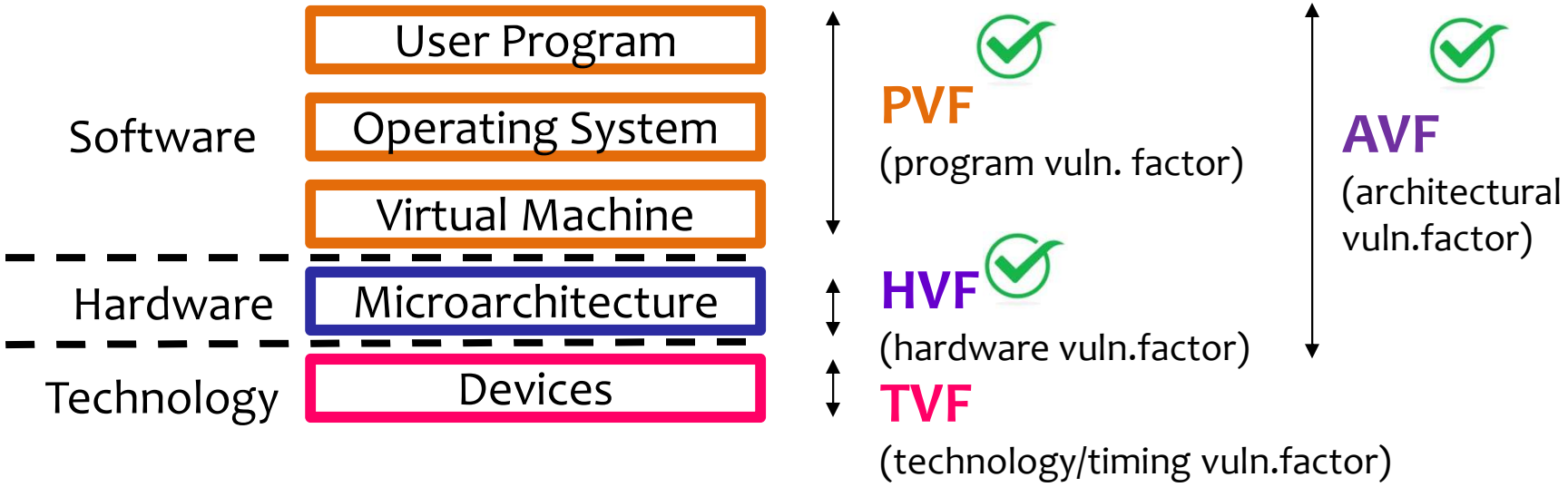


Early + Fast + Accurate

System Vulnerability Stack

(Sridharan and Kaeli, ISCA 2010)

- **Faults can be masked at different layers**
  - Faults starting point: devices/technology
  - Hardware masking (HVF)
  - Software masking (PFV)



Terminology

- **FIT – Failures-in-Time**
  - number of failures per 1 billion (10^9) hours of operation
- **MTTF – Mean Time To Failure**
  - 1/FIT
- **AVF – Architectural Vulnerability Factor [0…1]**
  - of a microprocessor structure – probability a soft error in structure's bit affects program execution – depends on microarchitecture & program
- **Silent Data Corruption (SDC)**
  - Error is not detected, program output is corrupted
- **Detected Unrecoverable Error (DUE)**
  - Error is detected, can't be recovered
- **Masked/Benign Error**
  - Program execution is not affected in any way

CAL@DI

Full CPU FIT_{SER} Calculation

- **For each hardware structure and program**
  - **Measure $AVF_{STRUCTURE}$** (injection or analytical)
- **Translate AVF to FIT**

$$FIT_{STRUCTURE} = FIT_{BIT} \times AVF_{STRUCTURE} \times \#Bits_{STRUCTURE}$$

technology | microarchitecture + software | size

- **For the entire CPU** (additive)

$$FIT_{CPU} = FIT_{S1} + FIT_{S2} + \ldots + FIT_{Sn}$$

Reliability Assessment – Options

- Level
  - Architecture/ISA – **hardware details are missing**
  - Microarchitecture – **early available hw model**
  - RTL – **late hw model**
- Simulation speed
  - Microarchitecture: **100x-1000x faster** than RTL*
- Method
  - Massive injections – **accurate but slow**
  - One-time analytical – **fast but pessimistic**

Abstraction Layer	Performance (cycles/sec) **
Software	3 x 10^9
Architecture	6 x 10^7
Microarchitecture	3 x 10^6 (simple CPU) 2 x 10^5 (detailed CPU)
Flip-flop	6 x 10^2

100X – 10,000X

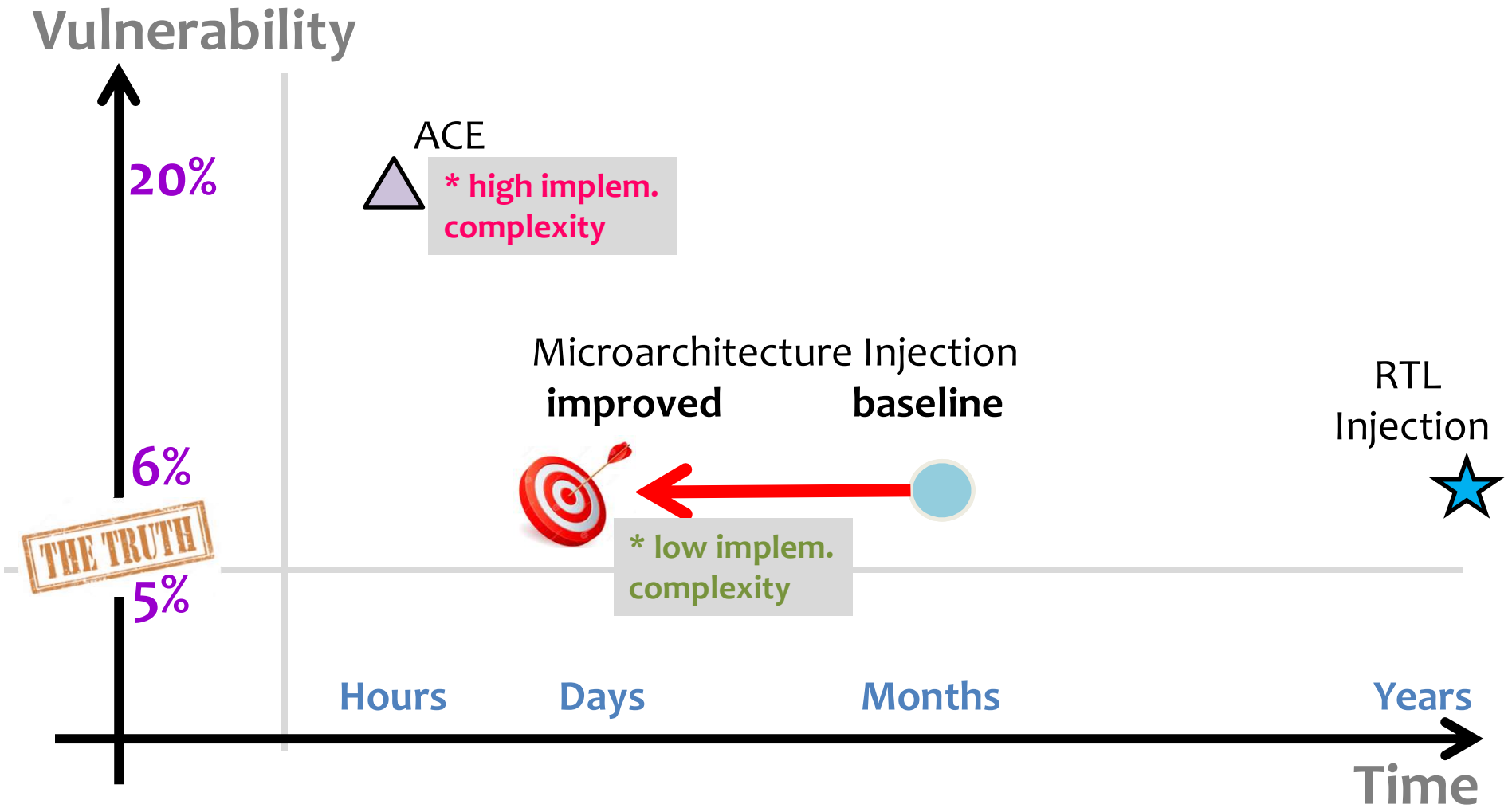
* S.Raach, A.Biswas, J.Stephan, P.Racunas, J.Emer, "A Fast and Accurate Analytical Technique to Computer the AVF of Sequential Bits in a Processor", ACM/IEEE MICRO 2015.

** J.Goodenough, R.Aitken, "Post-Silicon is Too Late – avoiding the $50 Million Paperweight Starts with Validated Designs, ACM/IEEE DAC 2010.

** H.Cho, S.Mirkhani, C.-Y.Cher, J.A.Abraham, S.Mitra, "Quantitative Evaluation of Soft Error Injection Techniques for Robust System Design", ACM/IEEE DAC 2013.

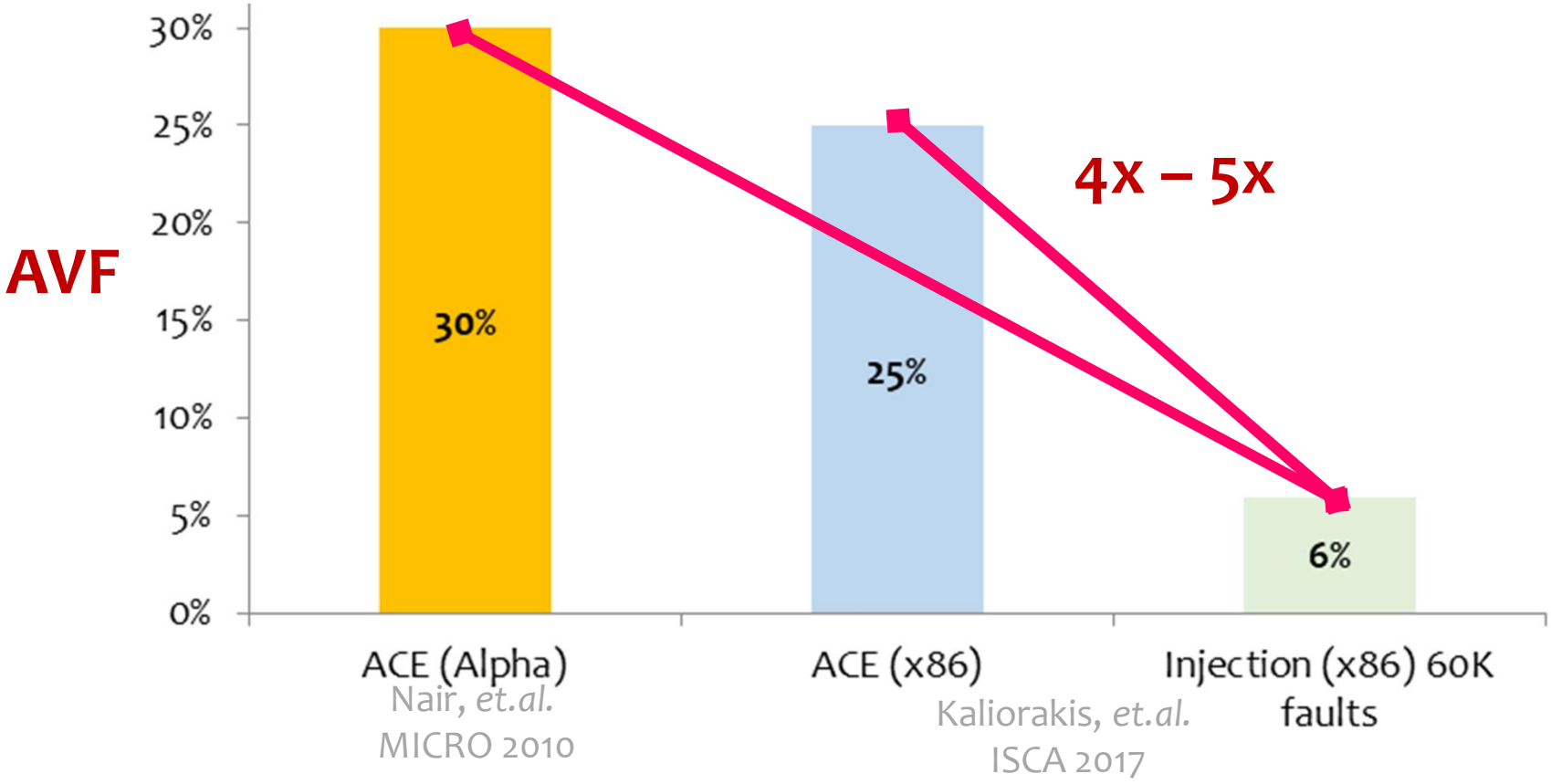
CAL@DI

Speed vs. Accuracy – example
(1 program + 1 hardware structure)



Injection vs. ACE

- Reg.file AVF (80 regs); same MiBench progs.



"MeRLiN: Exploiting Dynamic Instruction Behavior for Fast and Accurate Microarchitecture Level Reliability Assessment", M.Kaliorakis, D.Gizopoulos, R.Canal, A.Gonzalez, ACM/IEEE International Symposium on Computer Architecture (ISCA 2017), Toronto, Canada, June 2017.

Microarchitecture vs. RTL Speed

- **RTL** very accurate
- **But** very slow
  - Small **sim. window**
  - (100K-200K)
- Hardware **observation points** (CPU pinout)
  - CPU as a **hardware block**
  - System level studies impossible (incl. sw)

Benchmark	Speed (seconds/run)		
	RTL	GeFIN	Ratio
FFT	7001	39.1	179x
qsort	3157	23.9	132x
sha	3421	8.0	427x
s corners	1019	3.2	315x
s edges	874	3.6	242x
s smooth	893	3.7	241x
average			256x

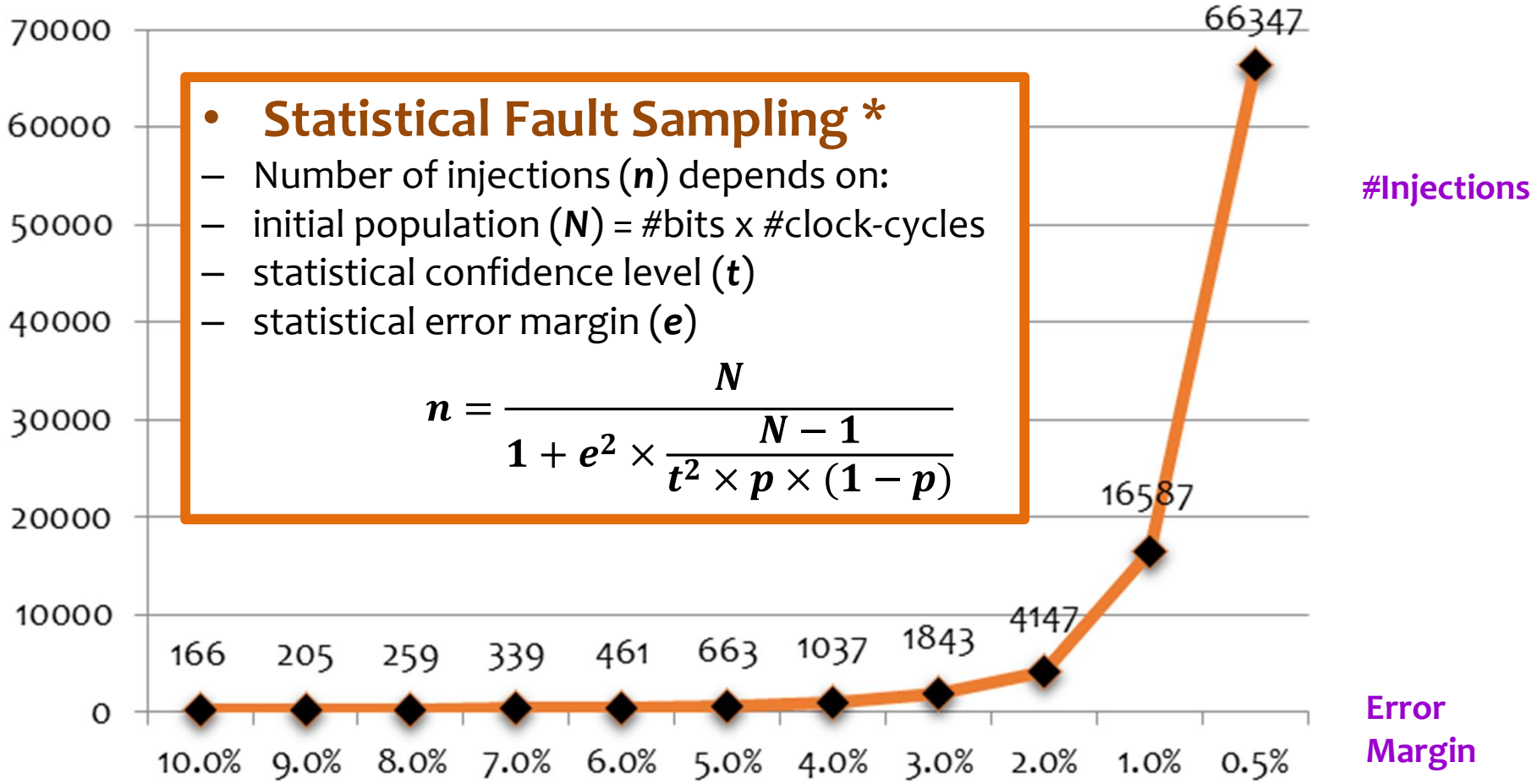
RTL model: ARM Cortex-A9
Microarchitecture model: Gem5 Cortex-A9
FFT execution = **~45 M cycles**
s smooth execution = **~3 M cycles**

* A.Chatzidimitriou, M.Kaliorakis, D.Gizopoulos, M.Iacaruso, M.Pipponzi, R.Mariani, S.Di Carlo, "RT Level vs. Microarchitecture Level Reliability Assessment: Case Study on ARM Cortex-A9 CPU", DSN-w, 2017

CAL@DI

#Fault Injections vs. Error Margin

- For 32K bits, 1B cycles, 99% confidence

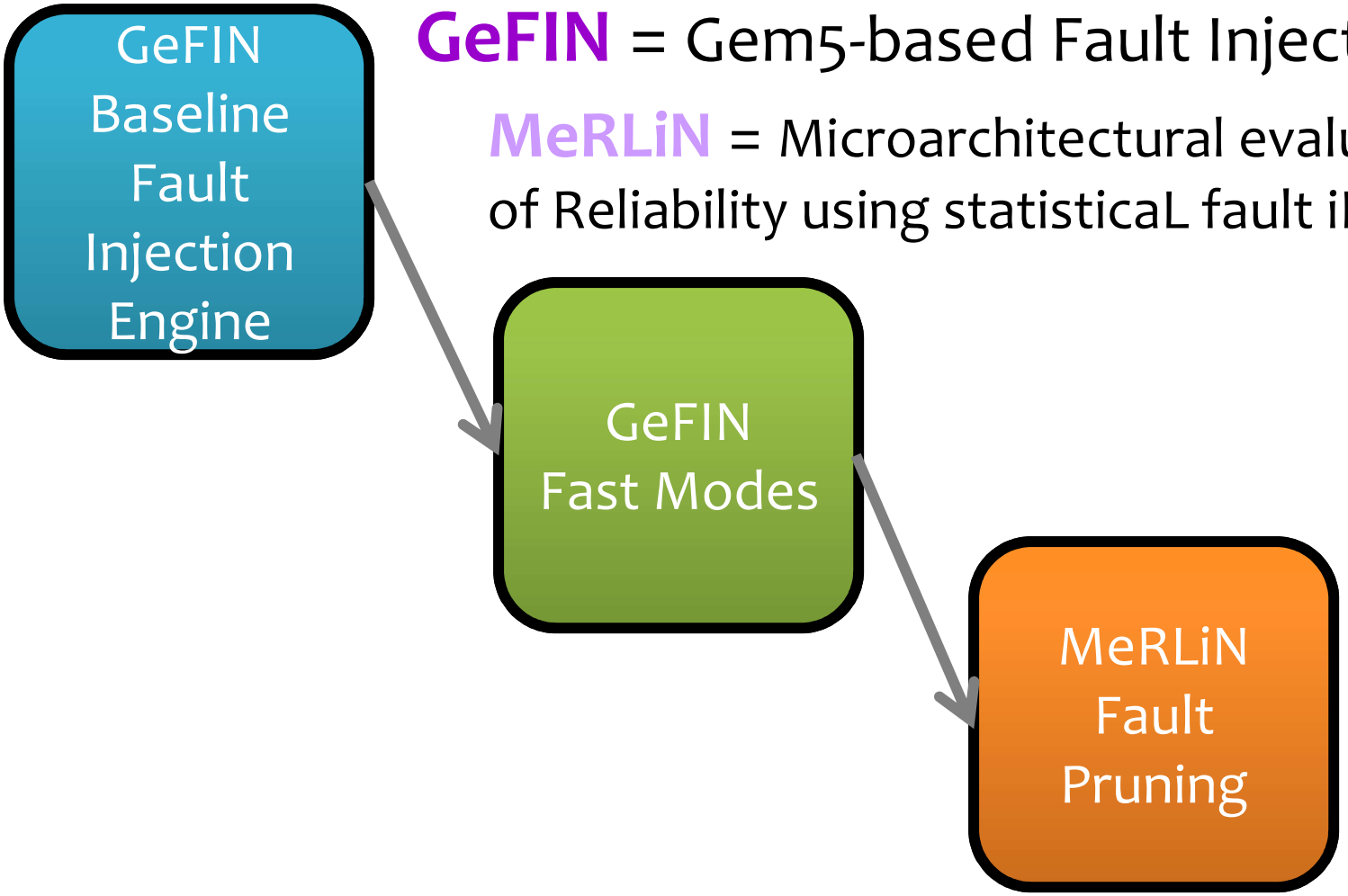


- **Statistical Fault Sampling ***
  - Number of injections (***n***) depends on:
  - initial population (***N***) = #bits x #clock-cycles
  - statistical confidence level (***t***)
  - statistical error margin (***e***)

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

* Leveugle, et. al., "Statistical Fault Injection: Quantified Error and Confidence", DATE, 2009

Tools to Speed Up Injections



GeFIN = Gem5-based Fault Injector

MeRLiN = Microarchitectural evaluation of Reliability using statisticaL fault iNjection

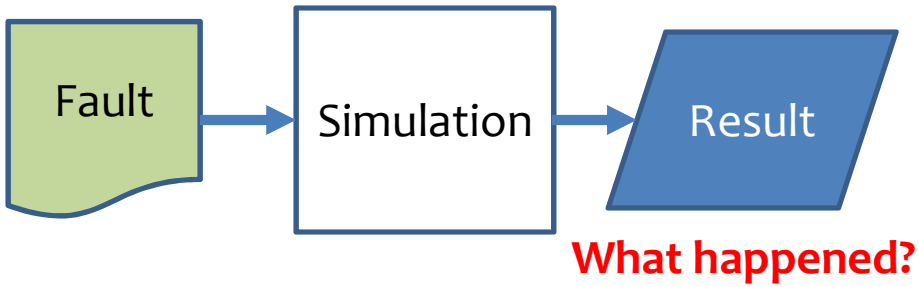
Microarchitecture-level Fault Injection

Baseline GeFIN Engine

Full system microarchitectural fault injection simulation (gem5 -> GeFIN)

A fault injection simulation consists of:

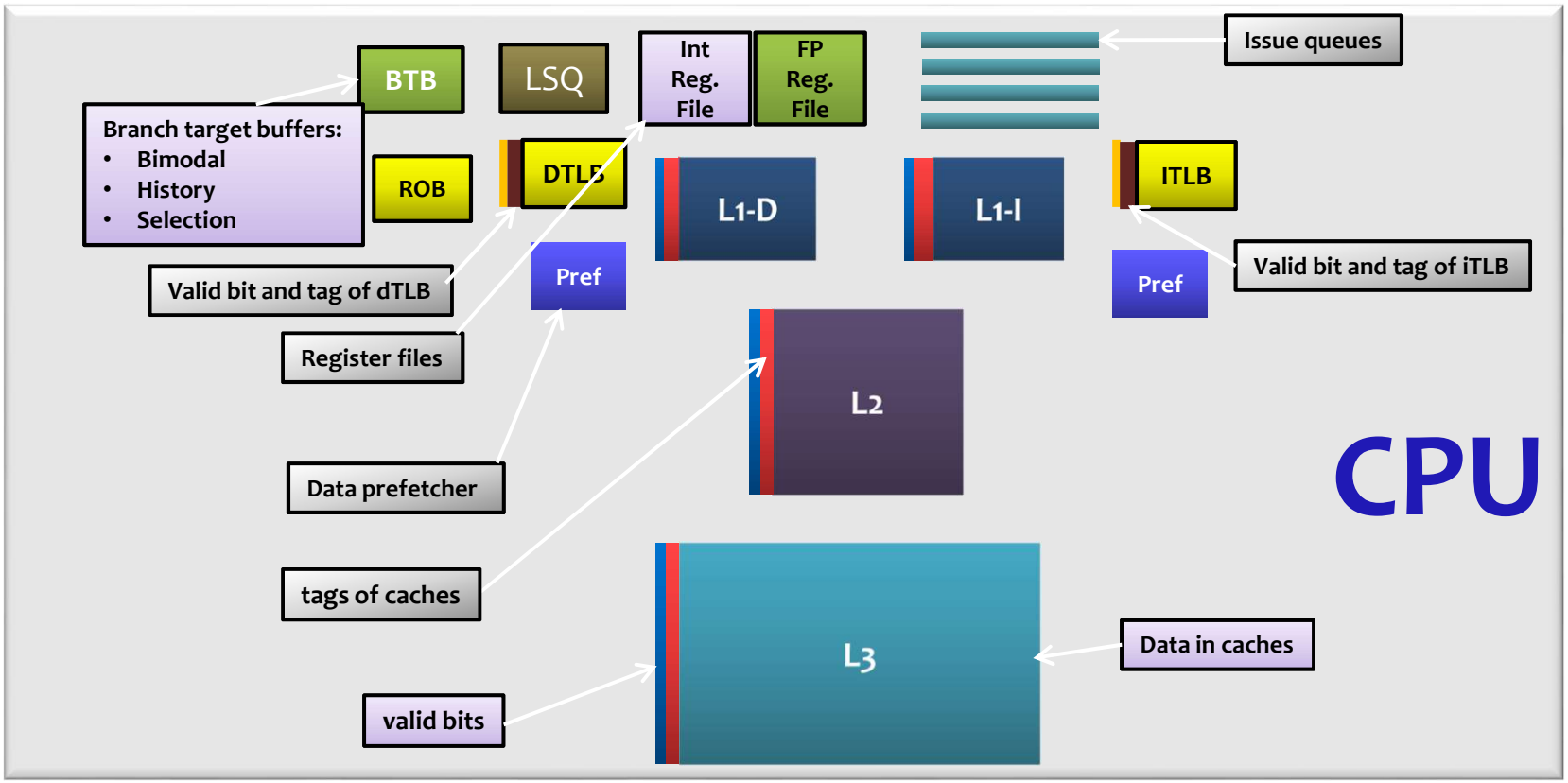
- A simulated full-system **hardware**
  - Detailed CPU microarchitecture
  - Peripheral devices
- An **operating system**
- An **application**
- A set of faults for **injection**



Foutris, et al., "Versatile architecture-level fault injection framework for reliability evaluation: A first report", IOLTS,2014
Kaliorakis, et. al., "Differential Fault Injection on Microarchitectural Simulators", IISWC, 2015

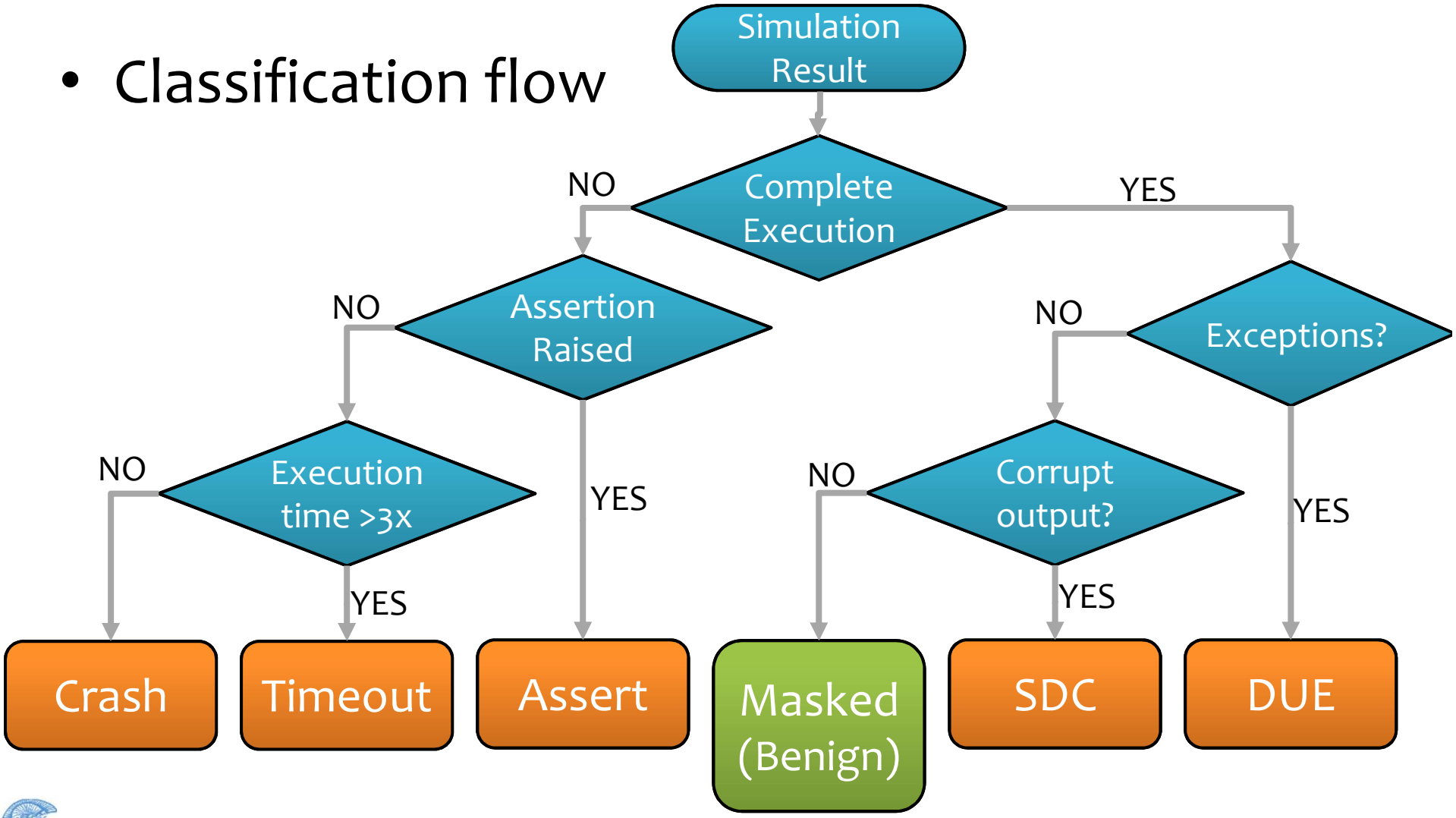
Target components to inject – Gem5

More than 50 hardware structures for injection: Caches, Registers, Register Files, Buffers, Queues, BPUs, BTBs, TLBs, Translation caches, etc.



Fault effects – Fine Granularity

- Classification flow



Vulnerability estimation

- **Statistical** sampling (by experiment)
  - Multiple **fault injection simulations**
  - **Classification** of each simulation, based on outcome
  - **Vulnerability** by measuring non-masked cases

$$AVF = \frac{\#vulnerable}{\#injections} = 1 - \frac{\#masked}{\#injections}$$

Case Study: Full CPU FIT Measurement

Comparing different CPUs

Configuration preset	Vendor
Arm Cortex A9 – like	 
Arm Cortex A15 – like	
Intel Skylake – like	

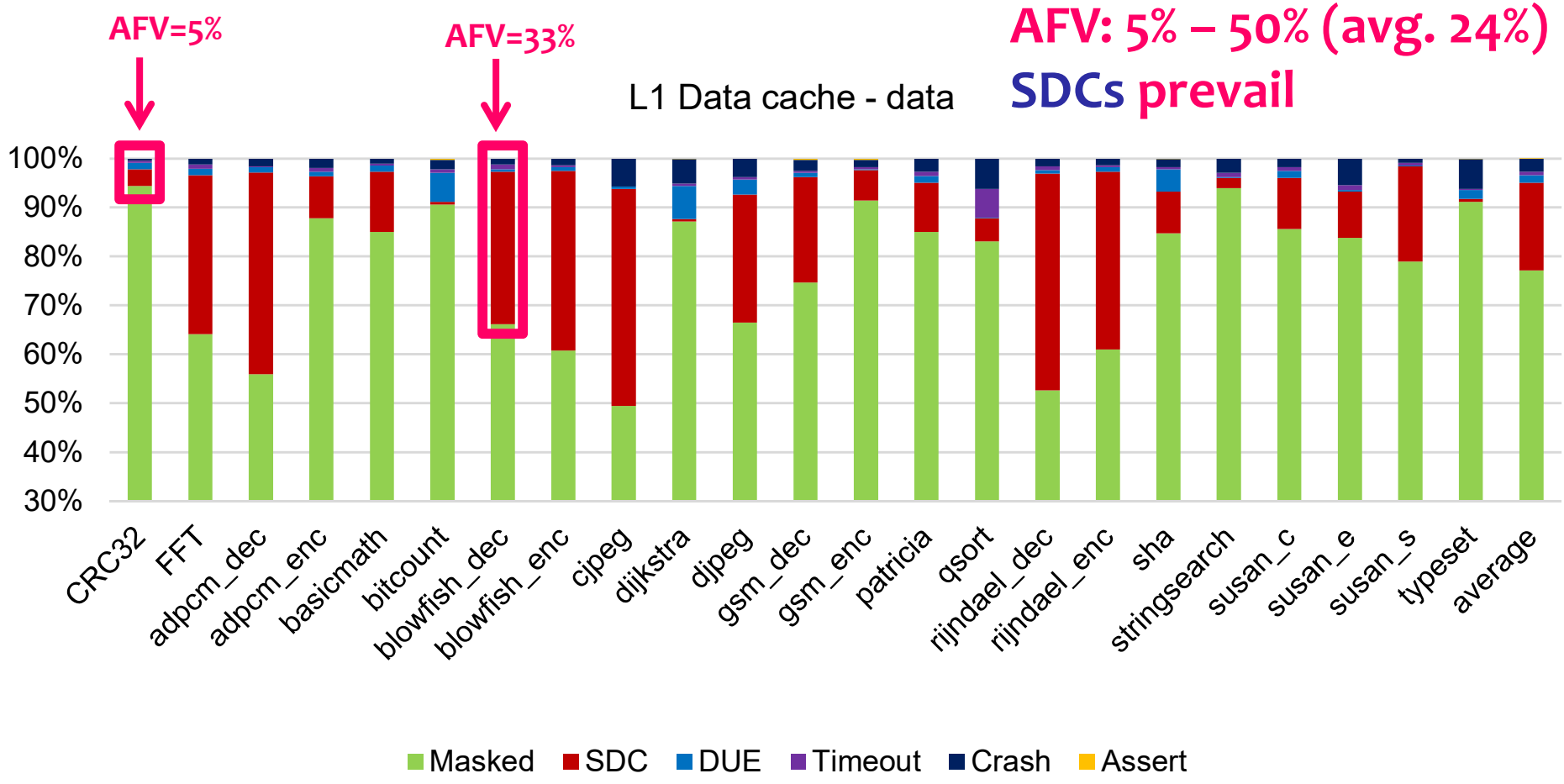
Chip failure rates (FIT) derived from:

26 hardware structures

24 workloads (mibench suite; 1M – 500M cycles)

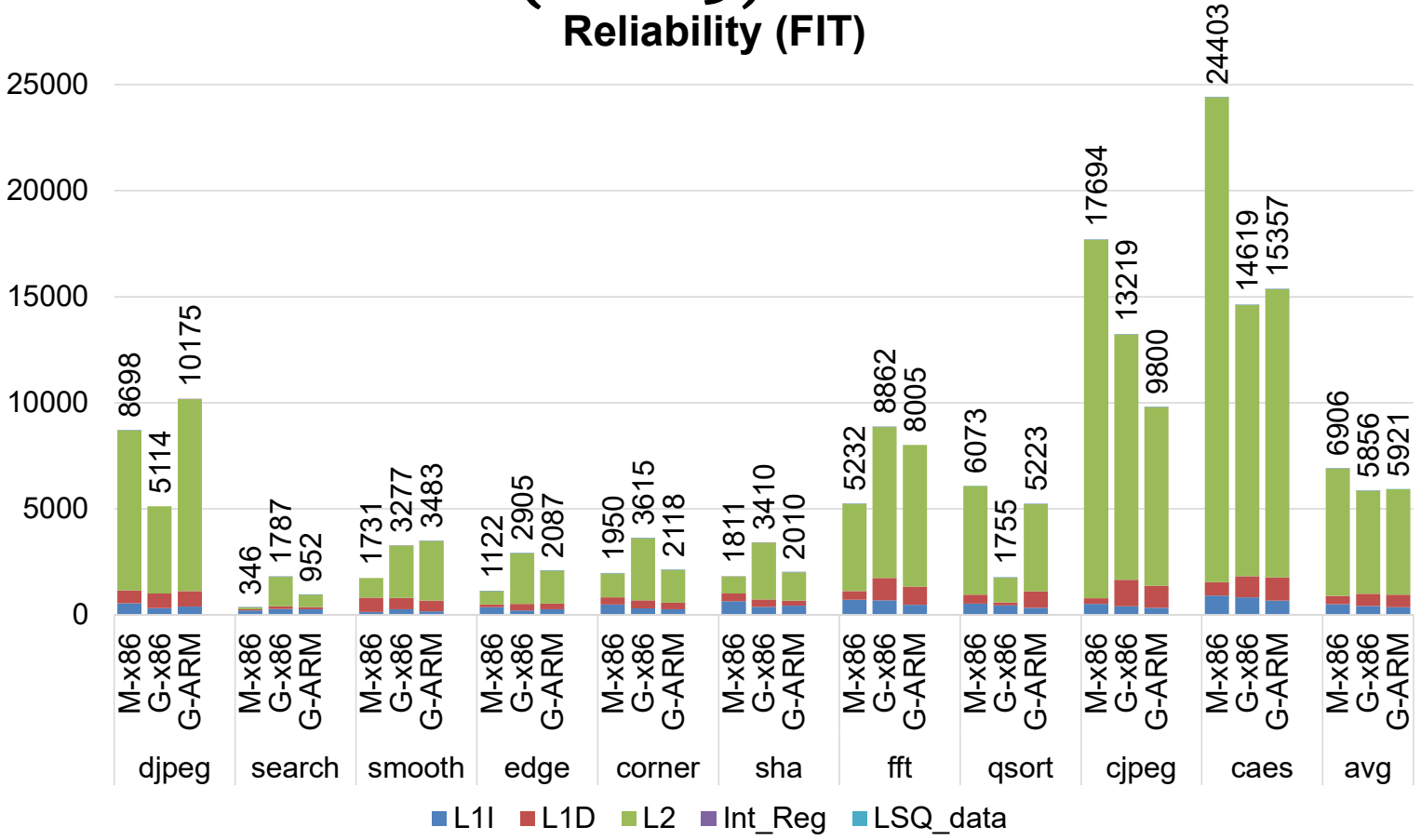
M.Kaliorakis,A.Chatzidimitriou, S.Tselons, D.Gizopoulos, "Differential Fault Injection on Microarchitectural Simulators", IISWC, 2015

A9 – AVF/Fault Effects – L1D cache



Comparison – CPUs and Injectors

- Two x86 models (Gem5 and Marssx86) and one ARM model (Gem5)



Uses of Fault Injection – Case Studies

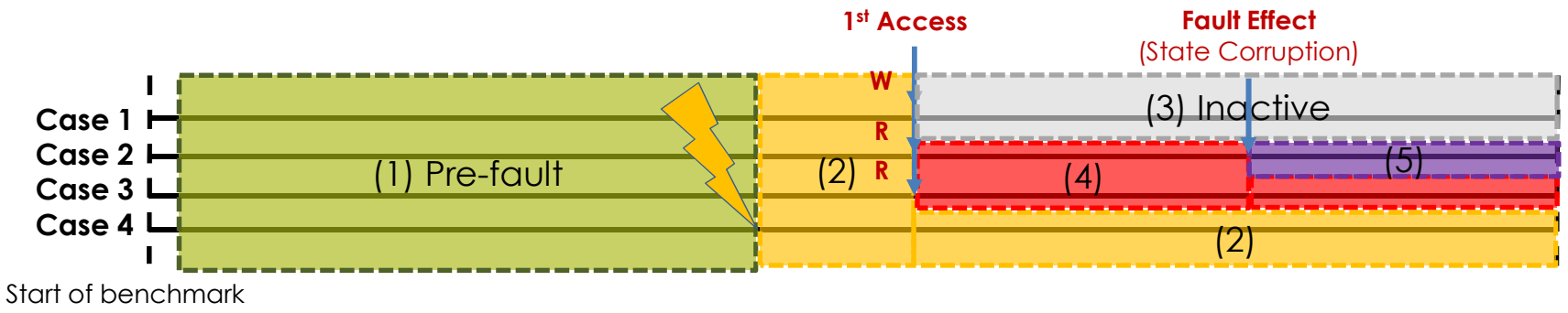
- **Check out the list of publications**
  - **Design exploration**
    - Microarchitectural attributes vs. vulnerability
  - **Performance** studies
    - Speculative components tolerance
  - **ISAs** comparisons
    - Arm vs. x86
  - **Compiler options** evaluation for vulnerability
  - **Error protection schemes** evaluation
  - **+Many more case studies**

CAL@DI

GeFIN fast modes
Accelerating a SFI campaign

"Anatomy of Microarchitecture-Level Reliability Assessment: Throughput and Accuracy", A.Chatzidimitriou, D.Gizopoulos, IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS 2016), Uppsala, Sweden, April, 2016.

Anatomy of a Fault injection simulation



- Fault **injection**
- First **access**
- First **visible effect**

- 1 **Pre-fault** epoch
- 2 **Idle** epoch
- 3 **Inactive** epoch
- 4 **Manifestation** epoch
- 5 **Corruption** epoch

- **Idle + Manifestation = ~13% of time**

“Squeeze” Single Injection Run Time

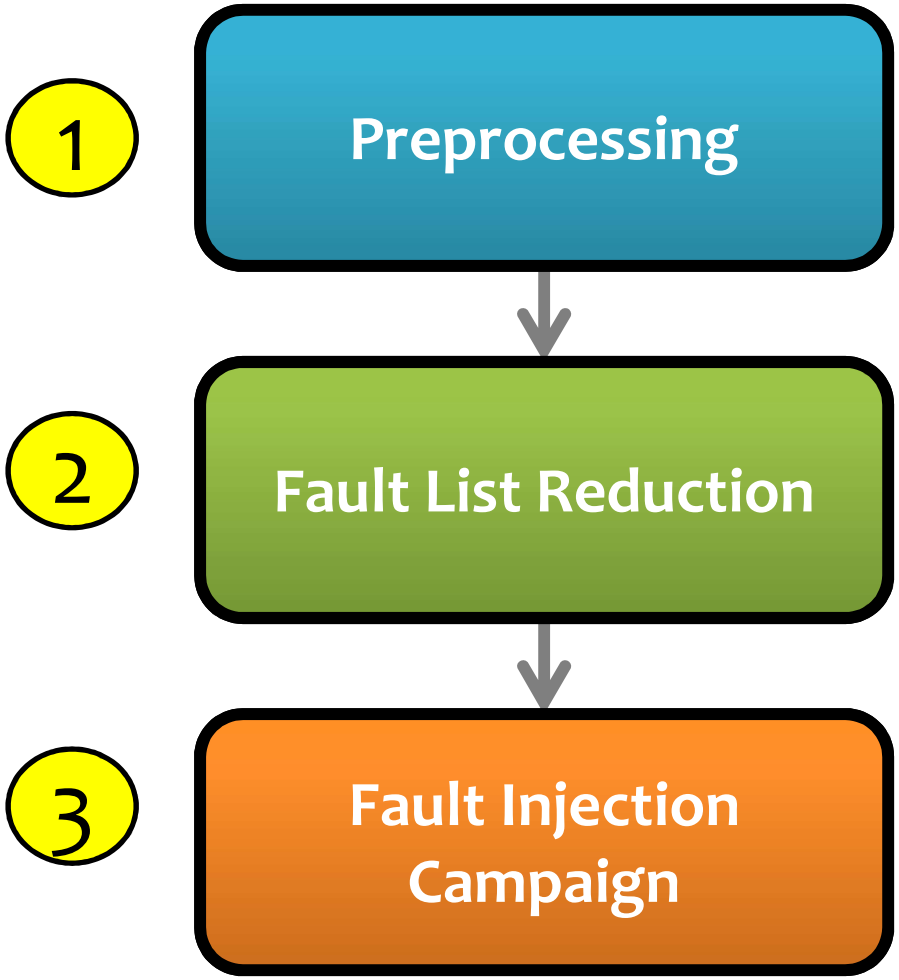
- Employ **everything** **imaginable** to accelerate single runs
  - [**Parallel** runs]
  - Fast **forwarding** (employ checkpoints)
  - Early **Stops** (overwritten, invalid entries)
  - Stop on Reaching **HVF** Boundary
  - Early **Limit**
  - Early **Switch** to Emulation
  - (*we keep going…*)

CAL@DI

MeRLiN Fault Classification & Pruning

"MeRLiN: Exploiting Dynamic Instruction Behavior for Fast and Accurate Microarchitecture Level Reliability Assessment", M.Kaliorakis, D.Gizopoulos, R.Canal, A.Gonzalez, ACM/IEEE International Symposium on Computer Architecture (ISCA 2017), Toronto, Canada, June 2017.

MeRLiN's Flow



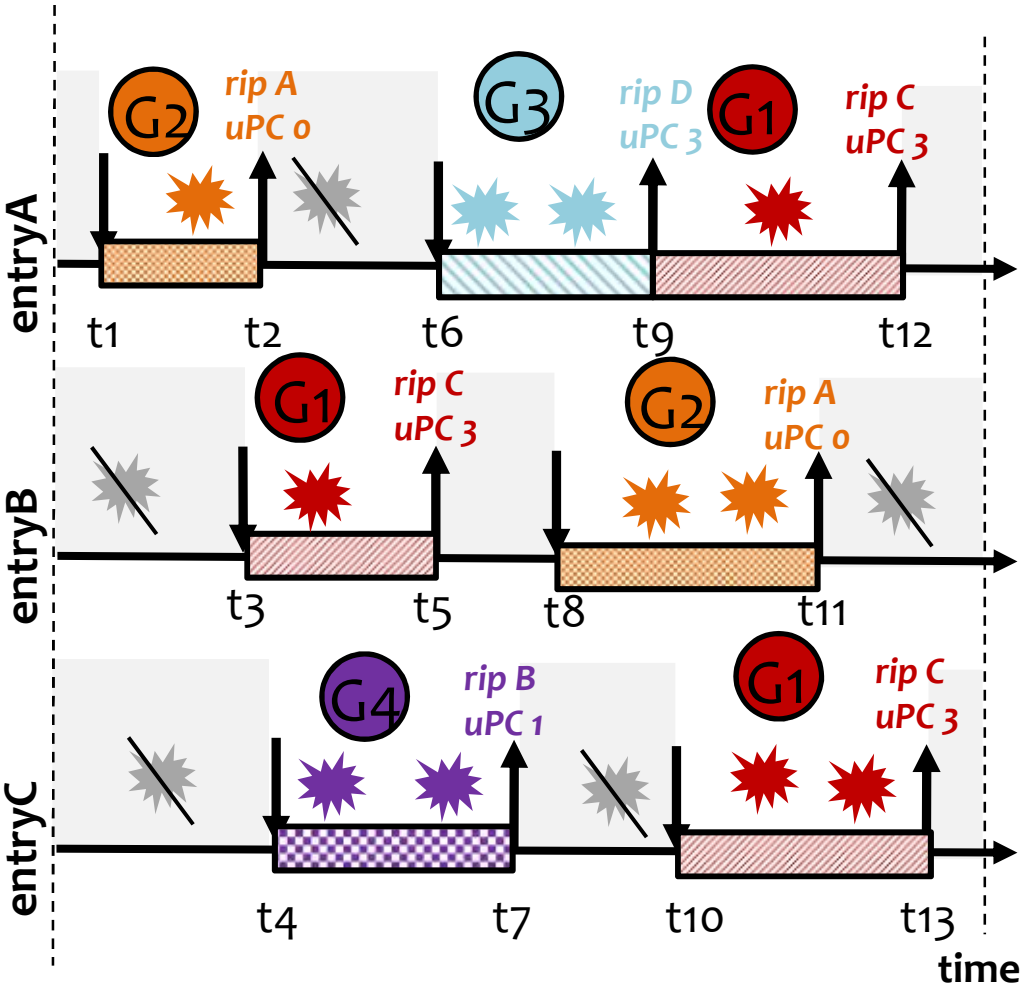
Two tasks:

1) Identification of faults that hit non-vulnerable intervals
2) Initial fault list generation

Implementation of a two-step group creation algorithm to reduce the remaining faults

Actual injections using the GeFIN microarchitectural simulator

MeRLiN – Fault List Reduction

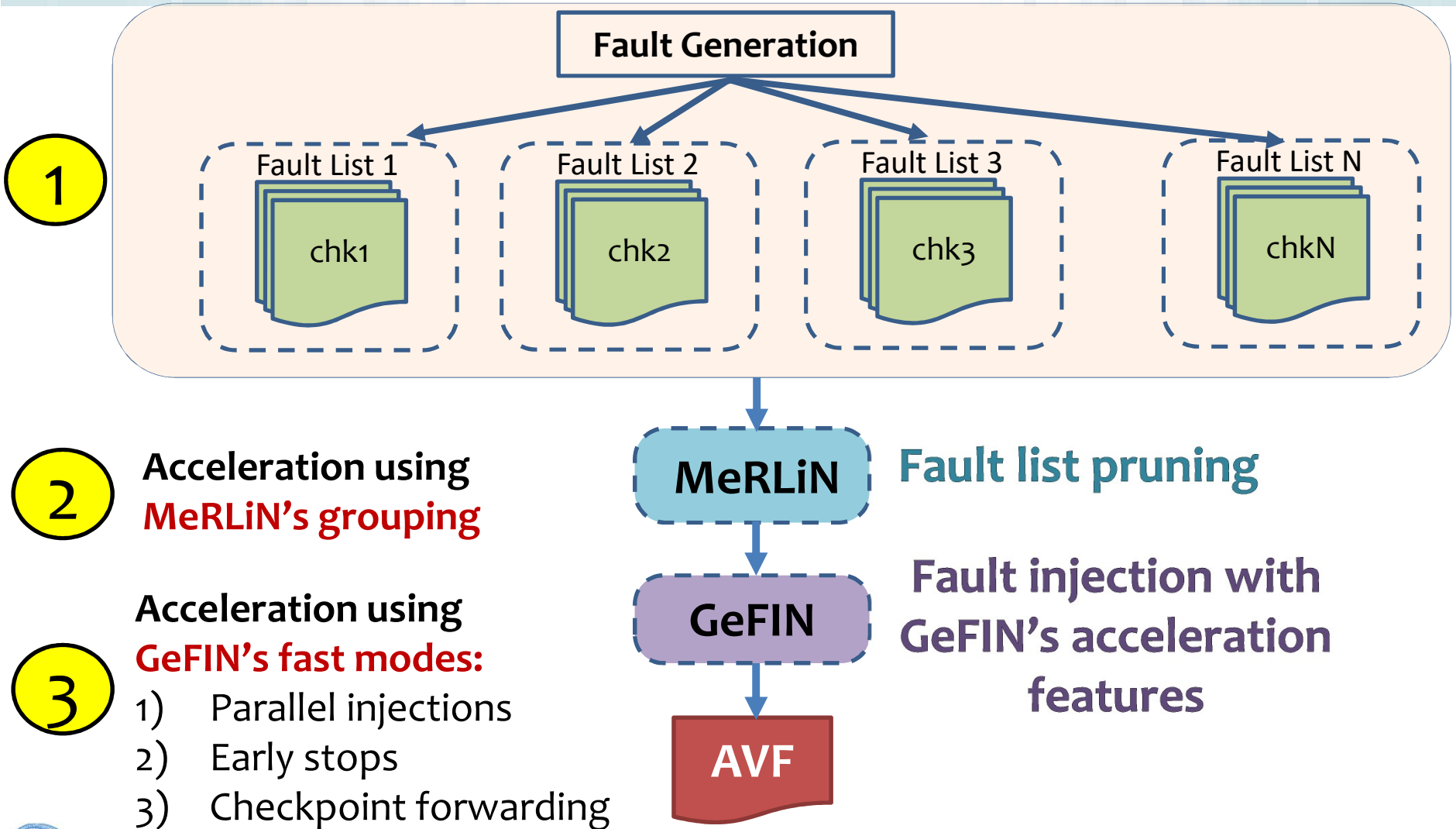


Group creation algorithm

- Faults of non-vulnerable intervals are considered as ***Masked***
- Group the rest faults according to the ***RIP*** and ***uPC***

GeFIN + MeRLiN combination
Complete example

GeFIN + MeRLiN Flow



Experimental Setup

- ✓ Simulate an **Intel x86 processor**
- ✓ Use of **3 MiBench benchmarks**
- ✓ 3 hardware structures (**RF** 168 regs., **SQ** 46 entries, 32KB **L1D**)
- ✓ Used **3 different machines** for our experiments:

23,873 faults per campaign:
error margin = 1%
confidence level = 99.8%

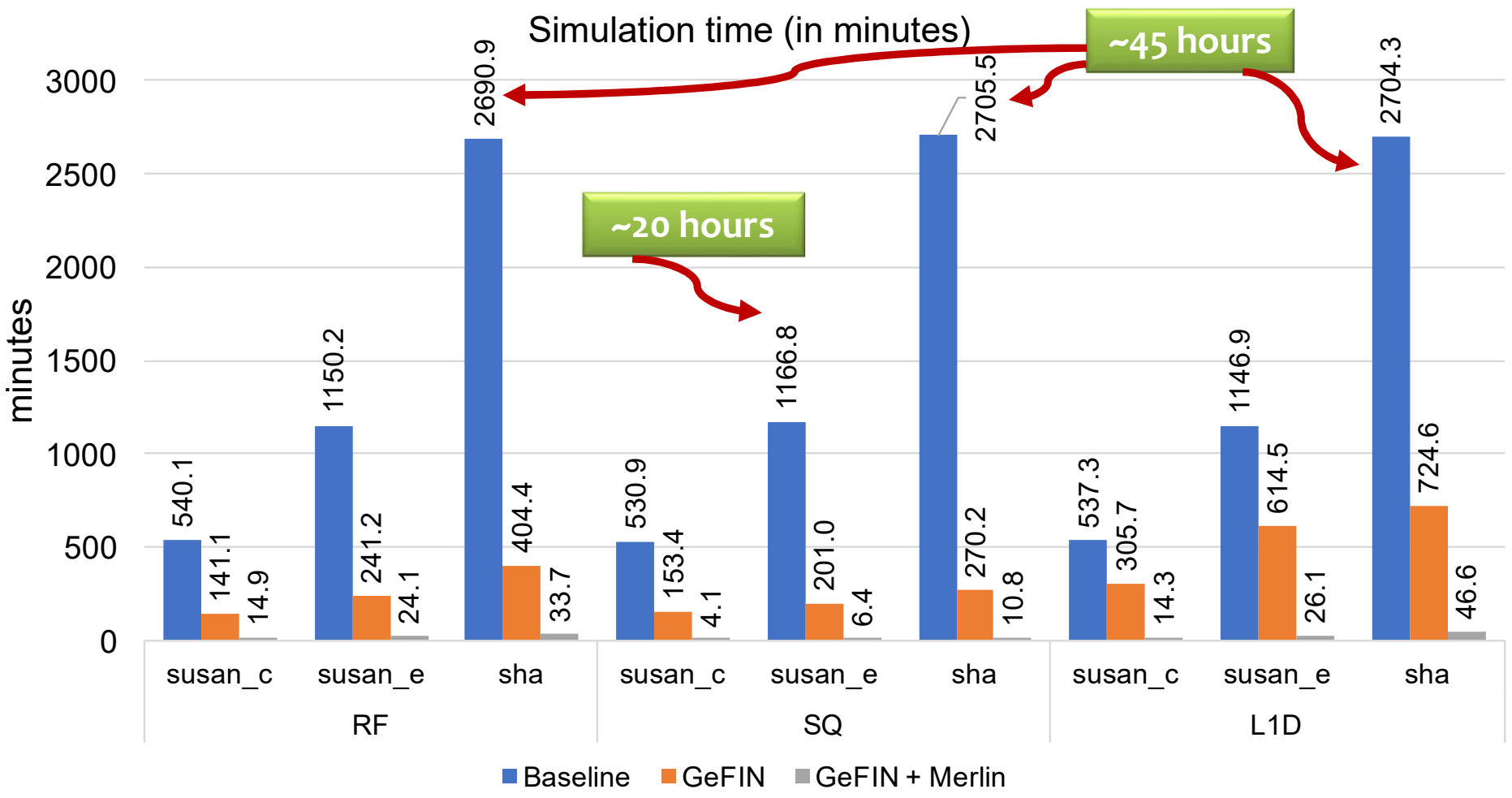
Machine	#1	#2	#3
benchmark	susan_c	susan_e	sha
simulation time of golden run	7.38 secs	16.23 secs	65.52 secs
# cores/threads	8/16	6/12	10/20
CPU type for simulations runs	Intel Xeon E5-2630 v3 @ 2.4GHz	Intel i7-4960X @ 3.6GHz	Intel Xeon E5-2690 v2 @ 3GHz
L3	20MB	15MB	25MB
RAM	64GB DDR4 @ 2133MHz	64GB DDR3 @ 1866MHz	96GB DDR3 @ 1866MHz
Kernel version	Linux kernel 3.13.0	Linux kernel 3.13.0	Linux kernel 3.13.0

SPEC

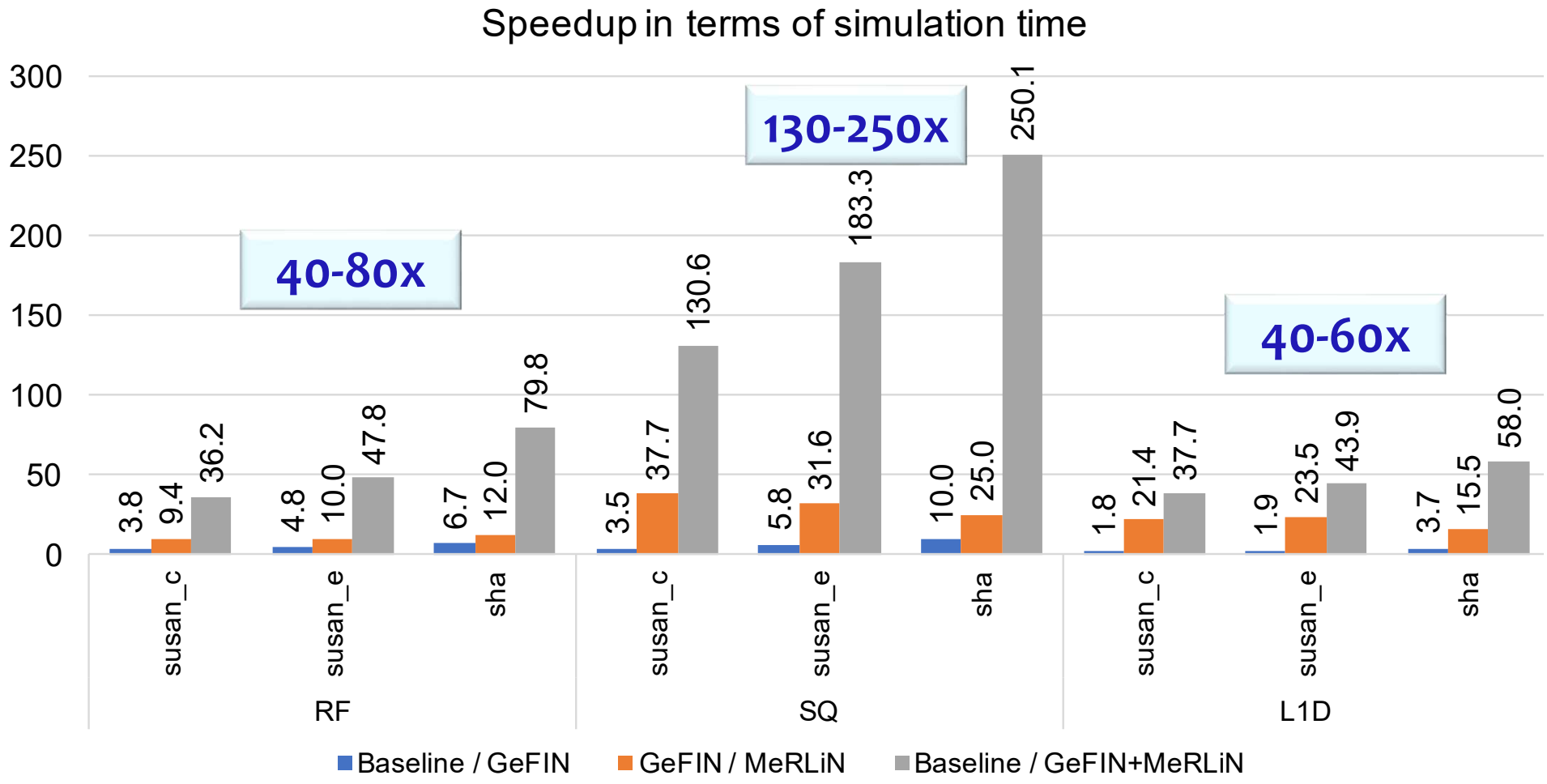
Linux

intel SKYLAKE

Actual Injection Time

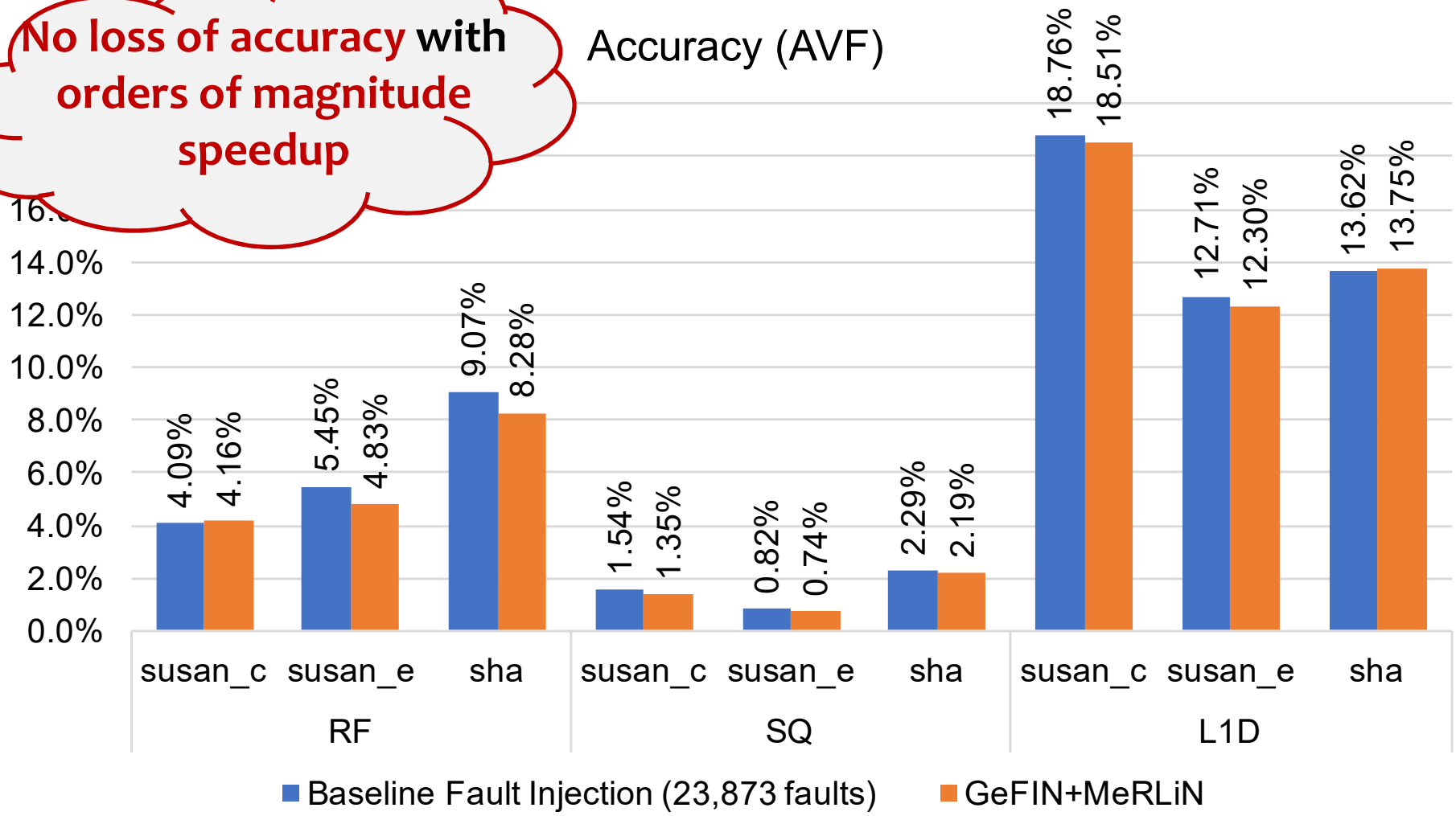


Speedups (Simulation Time)



Final Accuracy Estimation

No loss of accuracy with orders of magnitude speedup

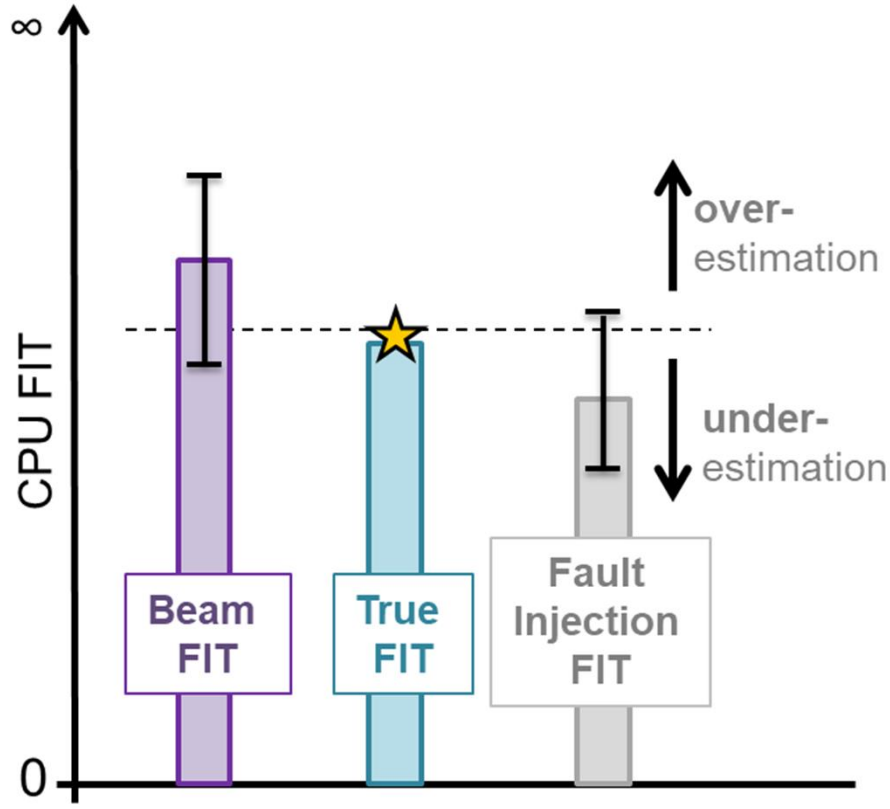


GeFIN vs. Neutron Beam

A.Chatzidimitriou, P.Bodmann, G.Papadimitriou, D.Gizopoulos, P.Rech, "Demystifying Soft Error Assessment Strategies on ARM CPUs: Microarchitectural Fault Injection vs. Neutron Beam Experiments", IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2019), Portland, OR, June 2019.

GeFIN vs. Neutron Beam Testing

- **Beam** vs. **Microarchitectural Fault Injection**
  - vs. the True System Reliability (FIT rate)

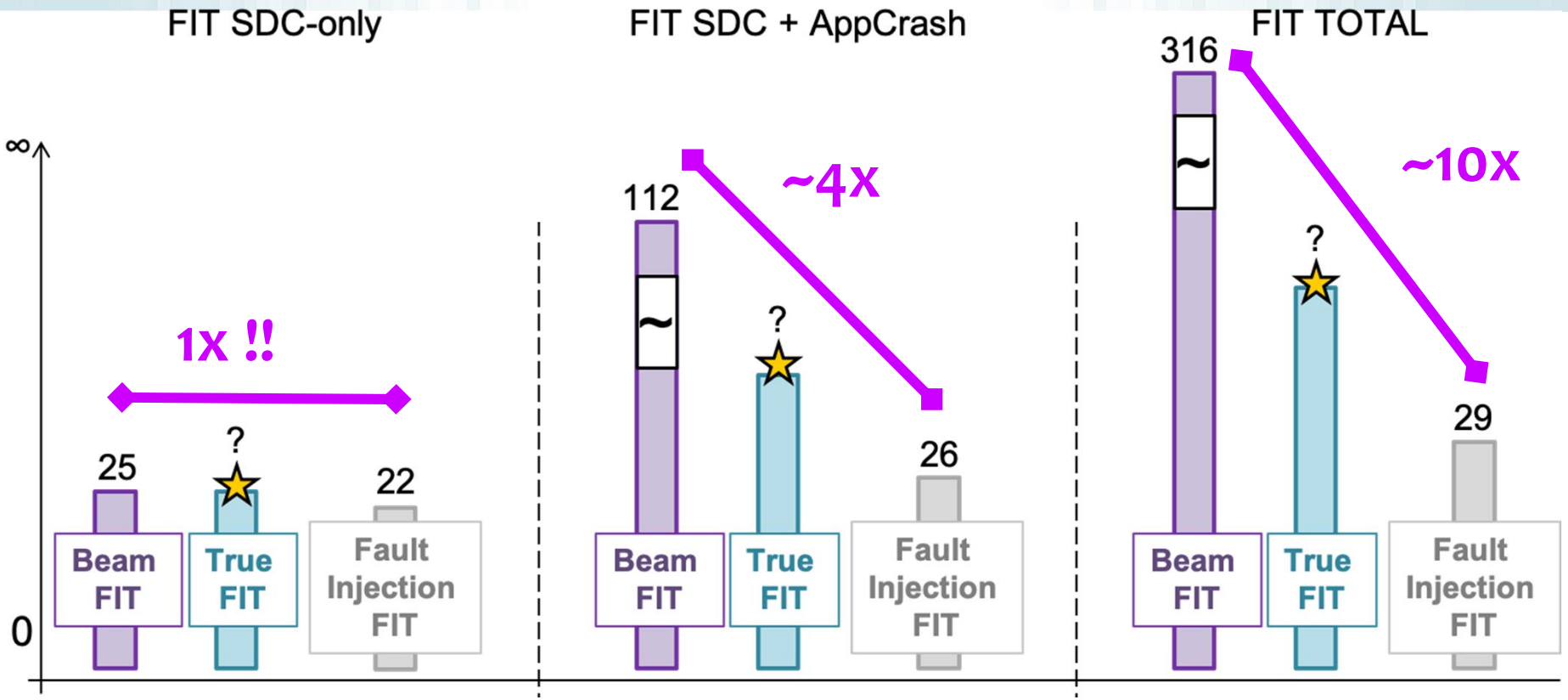


Assessing the ARM Cortex-A9

- Target platform: **ARM Cortex-A9**
  - ARM ISA and microarchitecture
- Any ISA or platform can be employed
  - Need simulation model and actual chip
- No RTL needed

Aspect	Neuron Beam	**GeFIN injection**
Microarchitecture	Cortex-A9	Cortex-A9*
Platform	Zynq 7000	VExpress
CPU cores	1*	1
L1 Cache	32 KB 4-way	32 KB 4-way
L2 Cache	512 KB 8-way	512 KB 8-way
Linux Kernel	3.14	3.13

Microarchitecture vs. Beam



- **SDCs**
  - CPU core
- **+AppCrashes**
  - CPU core
  - Memory hierarchy
- **+SysCrashes**
  - CPU cores, Peripherals, Interconnects, etc

Conclusions

- **Microarchitecture Level Fault Injection:**
  - Early – Fast – Accurate (GeFIN + MeRLiN)
- **Full System Level Effects of Faults**
  - Only level this is possible with max observability/flexibility
- **Design Exploration**
  - protection scheme
  - microarchitecture
  - reliability target (domain)
  - software version

Publications (*conferences*)

full list here: http://cal.di.uoa.gr

- **DATE 2020**
  - rACE – reverse ACE analysis
- **DSN 2019, 2017**
  - Microarchitecture fault injection vs. beam and vs. RTL
- **IISWC 2019, 2015**
  - spatial MultiBitUpsets in GeFIN, and baseline GeFIN
- **ISPASS 2019, 2016**
  - Hard errors in predictors, GeFIN fast modes
- **ISCA 2017**
  - MeRLiN fault pruning
- **ICCD 2018**
  - Hard faults in predictors
- **ITC 2016**
  - Cross layer analysis
- **VTS 2018, 2017, 2016**
  - Case studies using microarchitecture injection
- **DFTS 2015**
  - Design space exploration using microarchitecture injection
- **IOLTS 2017, 2016, 2014**
  - Case studies

Thank you.

Dimitris Gizopoulos – University of Athens, Greece

Computer Architecture Lab

http://cal.di.uoa.gr http//www.di.uoa.gr/~dgizop