Towards Early and Accurate Reliability Evaluation

Dimitris Gizopoulos
University of Athens
Computer Architecture Lab

Nikos Foutris
Manolis Kaliorakis
Sotiris Tselonis
George Papadimitriou

Cross-Layer Early Reliability Evaluation for the Computing Continuum

Clereco
Cross-Layer Early Reliability Evaluation for the Computing cOntinuum

cal@di

SEVENTH FRAMEWORK PROGRAMME

http://www.clereco.eu

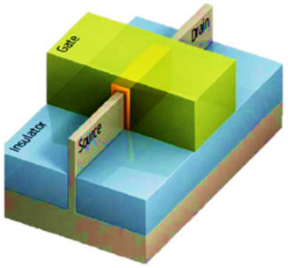
Reliability – Dependability

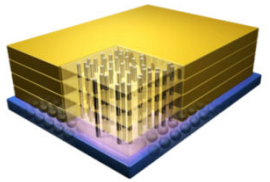
- **Importance** across the **Computing Continuum**

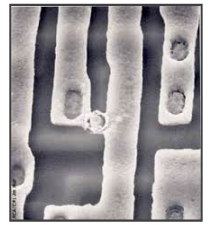
- Failure rates (MTBF):
  - Today: **days/weeks**
  - 2018: **mins/secs**?

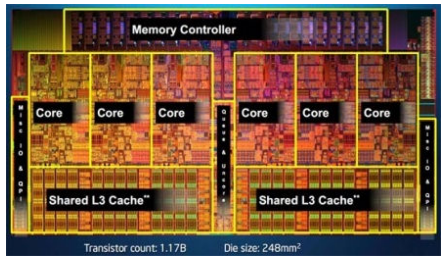
Drivers of **Un**-Reliability

- Devices shrinking (10nm and smaller)
- Processes (FinFET, scaled bulk, 3D stacks, spin logic, …)
- Variability
- Aging, wear-out
- Environment
  - radiation, temperature, humidity
- Many cores, many memories
- Heterogeneity









Clereco

cal@di

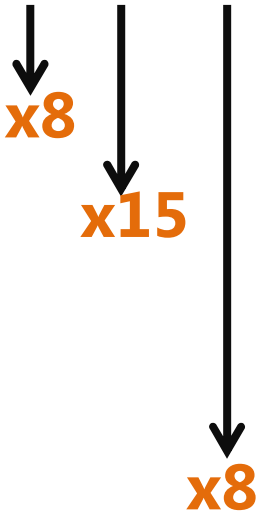
Protection

- Protection against (transient, permanent, intermittent) hardware faults **costs**
  - area, performance, power/energy, design time, …
- Protection: detection, diagnosis, recovery, repair

Protection: where and how much?

- Some components are more vulnerable
  - memory: DRAM, SRAM, registers
- But protection technique costs vary:

Protection Technique	Extra Storage
Parity	1.56%
SEC-DED	12.50%
DEC-TED	23.40%
Chipkill (IBM) DDDC (Intel)	12.50%
RAIM (zEnterprise)	40.60%
Mirroring (POWER7 RAS)	125.00%



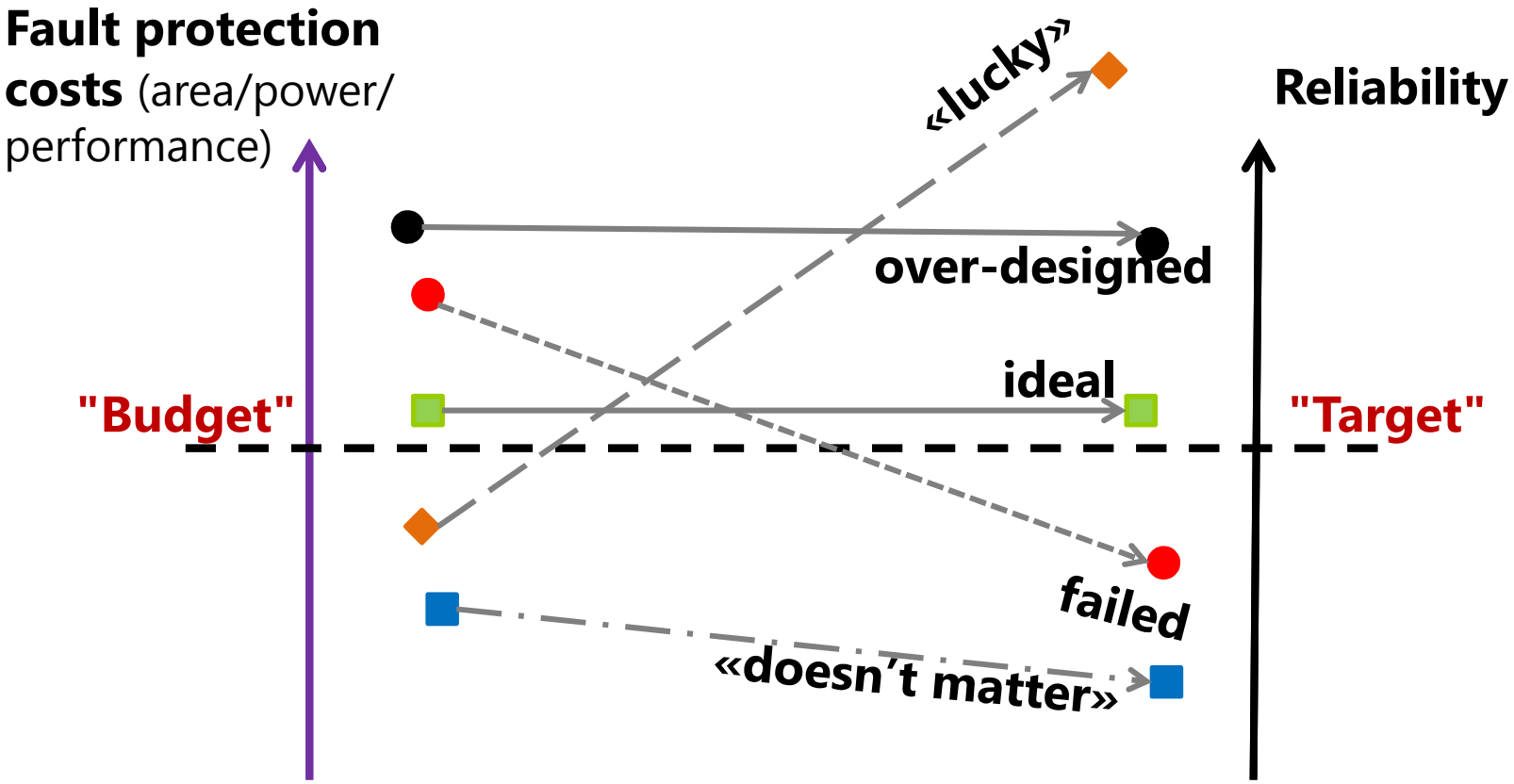
How much protection does a system need?

Y.Luo, *et. al.* "Characterizing Application Memory Error Vulnerability to Optimize Datacenter Cost via Heterogeneous-Reliability Memory", DSN, 2014.

How much protection: Decide Early

- **Early** protection decisions save costs **but should be accurate**
  - Ideally, no re-design cycles to enhance reliability
- But can "**early**" be **accurate** ?
  - System details missing
  - Unknown software
  - Unknown context
- **When can early be accurate ?**

Cost & Obtained Reliability



Micro-architectural Simulators

- **Reliability evaluation** using micro-architectural simulator models
  - This is **early**

  - Is it **fast enough**?

  - Is it **accurate enough**?

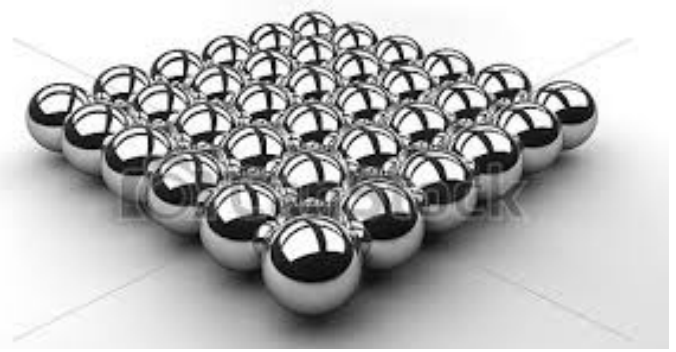
- For **which components** can this be good enough?

Components in uarch models

Component class	Realism of uarch models	Size	Vulnera-billity
Memory arrays (caches, regs, buffersm queues)	Yes	Large	High
Functional	No	Medium	Low
Control logic	No	Small/ Medium	Low

Micro-architecture Arrays

- **Storage arrays**
  - **registers**, **buffers**, **caches**, **memory**…
- Arrays/tables in the uarch models
- #Bits as in final hardware implementation
- Most **vulnerable**
  - DRAM, SRAM, flip-flops, …



Fault Injection or Analytical ?

- **Statistical Fault injection vs. Analytical Methods**
  - **SFI** is **accurate** but can be very **time consuming**
  - Analytical methods (such as **ACE-analysis for AVF estimations**) are **very fast** (single-pass) but can be very **pessimistic**
  - **Here we make the SFI choice to obtain accuracy and evaluate the speed of the measurement campaign**

Requirements for Accuracy

- A micro-architectural simulator can provide **accurate reliability evaluation through SFI** for **storage arrays** when:

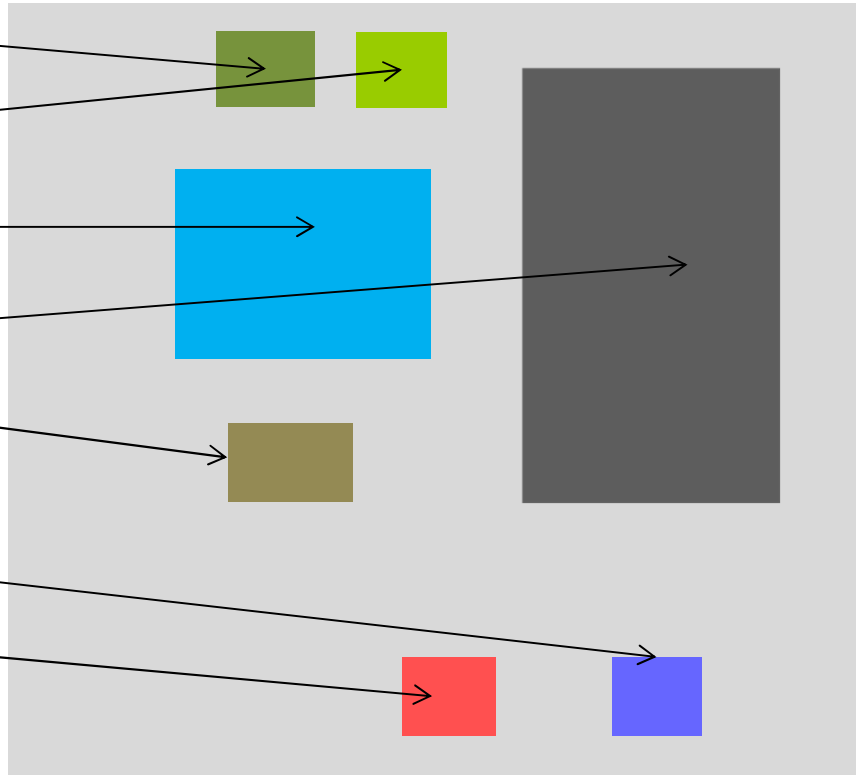
1. **All** important arrays are **modeled**
2. **#Injections is statistically** significant
3. **Injection throughput** is high

Marssx86-FI

- **Fault injector on Marssx86 uarch simulator**
  - **Transient/intermittent/permanent** faults
  - **Multiple** faults on single or **multiple components**
  - Full system simulation
- Fault behavior classes:
  - **SDC** (silent data corruptions)
  - **DUE** (detected unrecoverable errors)
  - **Masked** (benign)
  - **Deadlock**
  - **Livelock**

Marssx86 Enhancements

- Original Marssx86 model **enhancements**
  - **L1-I cache arrays**
  - **L1-D cache arrays**
  - **L2 cache arrays**
  - **L3 cache arrays**
  - **Branch target buffers**
  - **Data prefetcher**
  - **Instruction prefetcher**
  - …



Statistical Significance

- **Population** (N)
  - Bit positions (permanent faults)
  - Bit positions x Execution cycles (transient faults)
- **Confidence** (t)
- **Error margin** (e)
- Equally probable (sa0/sa1, 0-to-1/1-to-0 flips)
- **#Injections**
  - **$n = N / [1 + e^2 \times (N - 1)/(t^2 \times 0.25)]$**

R.Leveugle, *et. al.* "Statistical Fault Injection: quantified error and confidence", DATE, 2009.

Example Numbers of Injections

- Physical integer register file (IntRF)
  - 256 registers, 64-bit = **16,384 bits**
- or a **32KB** L1 D-Cache or L1 I-Cache

- Benchmark exec. time = **100M cycles**
- Error margin **1%**
- Confidence **99%**
- **16,587** injections [transient faults] (for each of the 3 cases) (8,243 injections for permanent faults)
  - **#injections saturates**

SFI Throughput

- Injection machine
  - Intel® Core™ i7-3970X @ 3.50 GHz (6 Cores, 12 Threads, 32 GB RAM, Ubuntu 14.04.1 LTS 3.13.0-36-generic x86_64

Time/ injection	#Injections/ component	#Components	#Benchmarks
~3 mins	**16,587**	3 (IntRF, L1D, L1I)	20

Total time (1 thread)	Total time (12 threads)	Total time (10 injection machines, 120 threads)
~2070 days	~ 175 days	**~ 18 days**

SFI Throughput (more accuracy)

- All same but 0.5% error margin (instead of 1%) and 99.8% confidence (instead of 99%)

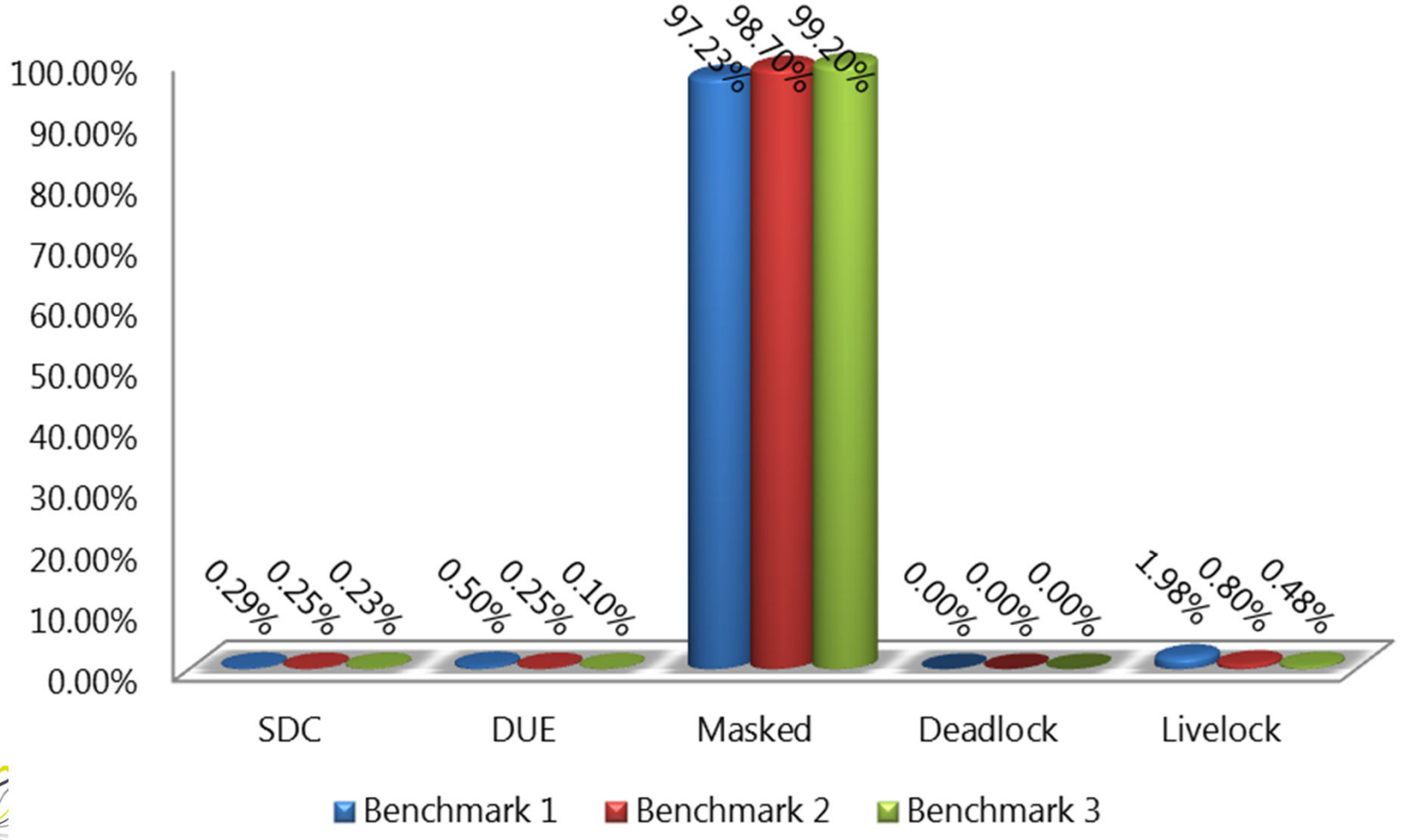
Time/ injection	#Injections/ component	#Components	#Benchmarks
~3 mins	**95,493**	3 (IntRF, L1D, L1I)	20

Total time (1 thread)	Total time (12 threads)	Total time (10 injection machines, 120 threads)
~12,000 days	~ 993 days	**~ 99 days**

Calculations refer to single transient faults. Larger numbers of fault injections are needed for multiple transient fault studies.

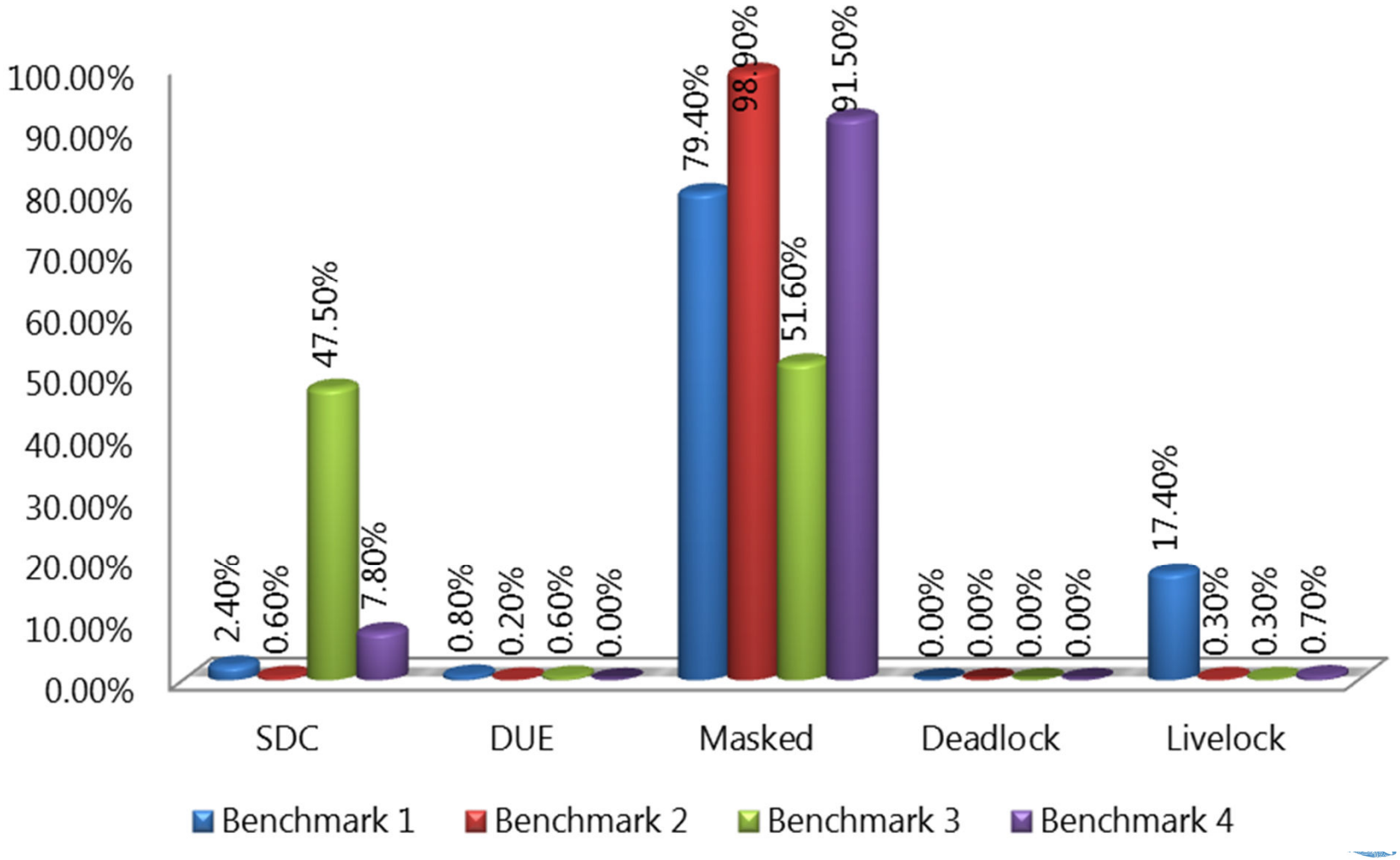
Results (1)

- Integer Register File



Results (2)

- L1 DCache



Conclusions

- **Micro-architectural simulators** for **SFI** and Reliability Evaluation of Storage Arrays
- **Early** evaluation
- Simulators need **enhancements**
- Fault injection **throughput** depends on:
  - Benchmarks number and execution times
  - Seeked accuracy (confidence, error margin)
  - Arrays sizes and numbers
  - Available servers



Thank you.

Dimitris Gizopoulos
University of Athens
Computer Architecture Lab



http://www.clereco.eu