Thesis Defense
Comprehensive Resiliency Evaluation for Dependable Embedded Systems

Yohan Ko

Dependable Computing Lab.
Dept. of Computer Science
Yonsei University

Committee

Prof. Kyoungwoo Lee
Prof. Bernd Burgstaller
Prof. Yo-Sub Han
Prof. Hyunok Oh

Outline

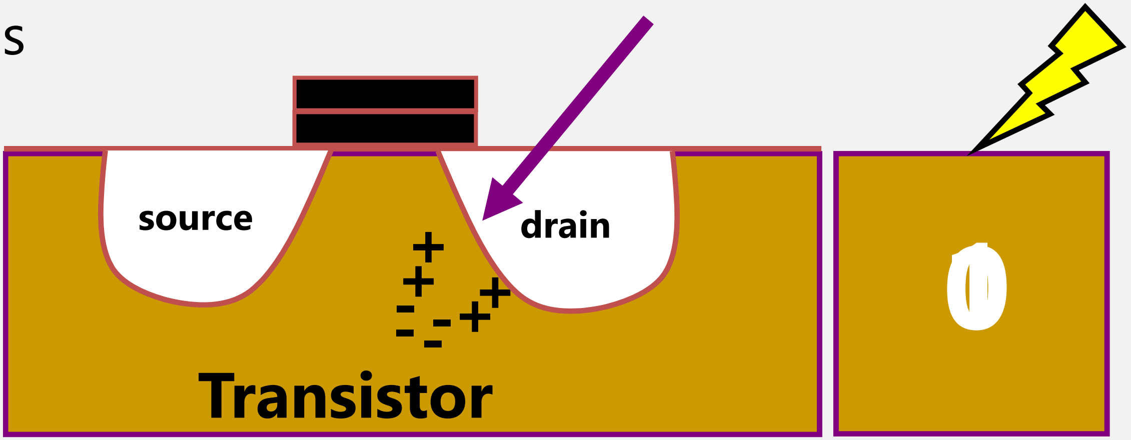
- Thesis reminder
- Comments from the previous presentation
- Response to comments
- Conclusion

Outline

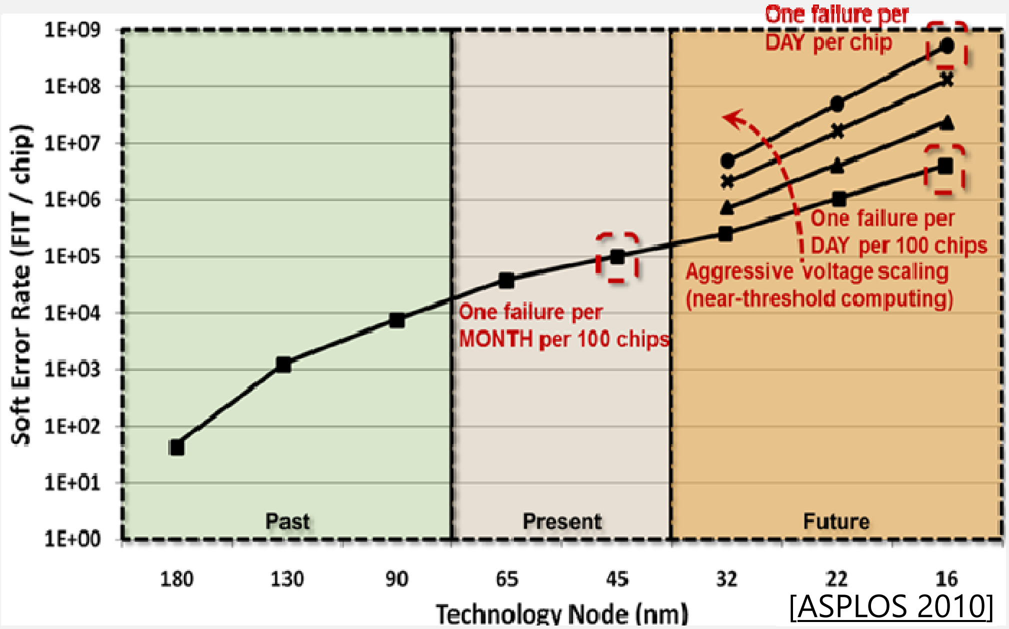
- Thesis reminder
  - How to quantify the resiliency of a processor
  - How to quantify the effectiveness of protection techniques
- Comments from the previous presentation
- Response to comments
- Conclusion

Soft errors?

- Charge carrying particles induce soft errors
  - Alpha particles
  - Neutrons
  - Cosmic ray



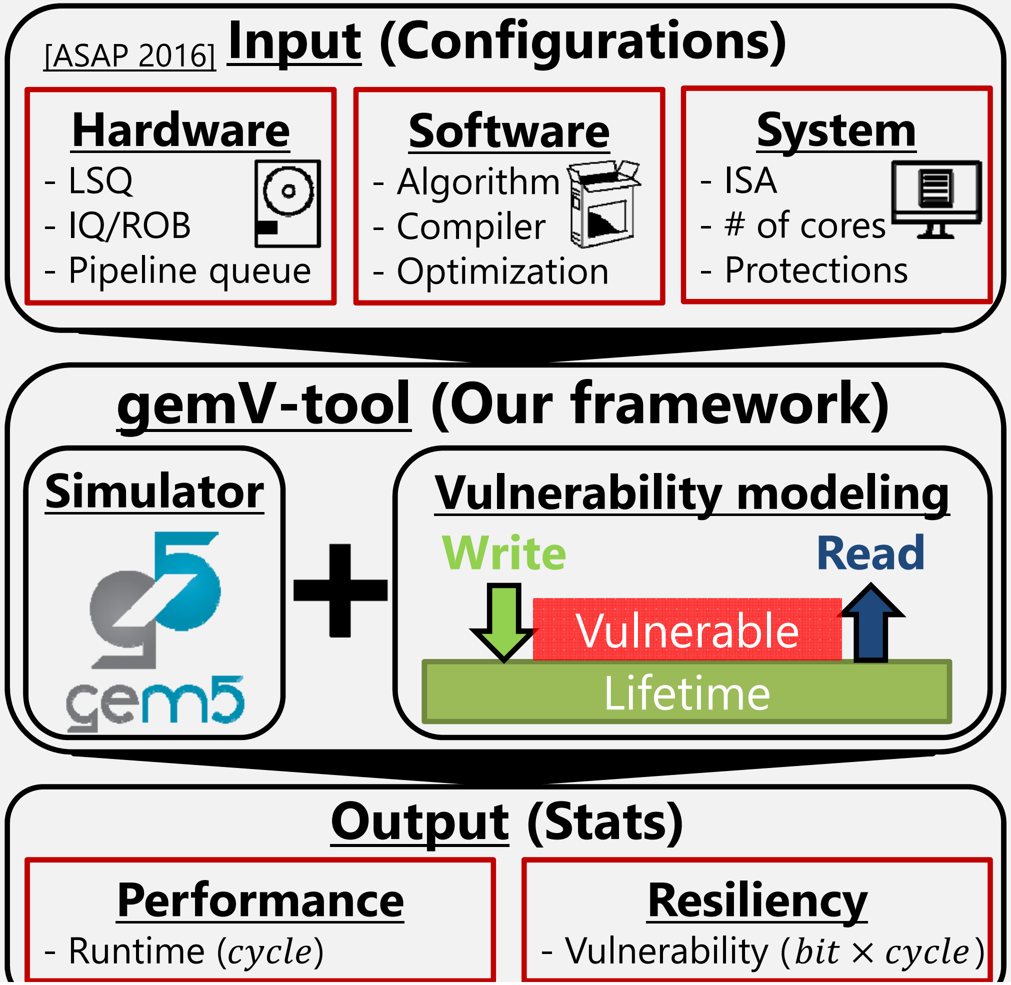
- Soft error rate
  - More than 1 bits in a chip
  - Exponentially increases with technology scaling and near-threshold computing



[ASPLOS 2010]

[ASPLOS 2010] Shuguang Feng, Shantanu Gupta, Amin Ansari, and Scott Mahlke. Shoestring: probabilistic soft error reliability on the cheap. International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS). 2010.

How to quantify the resiliency of a processor



[ASAP 2016] **Input (Configurations)**

Hardware
- LSQ
- IQ/ROB
- Pipeline queue

Software
- Algorithm
- Compiler
- Optimization

System
- ISA
- # of cores
- Protections

gemV-tool (Our framework)

Simulator + **Vulnerability modeling**

Output (Stats)

Performance
- Runtime ($cycle$)

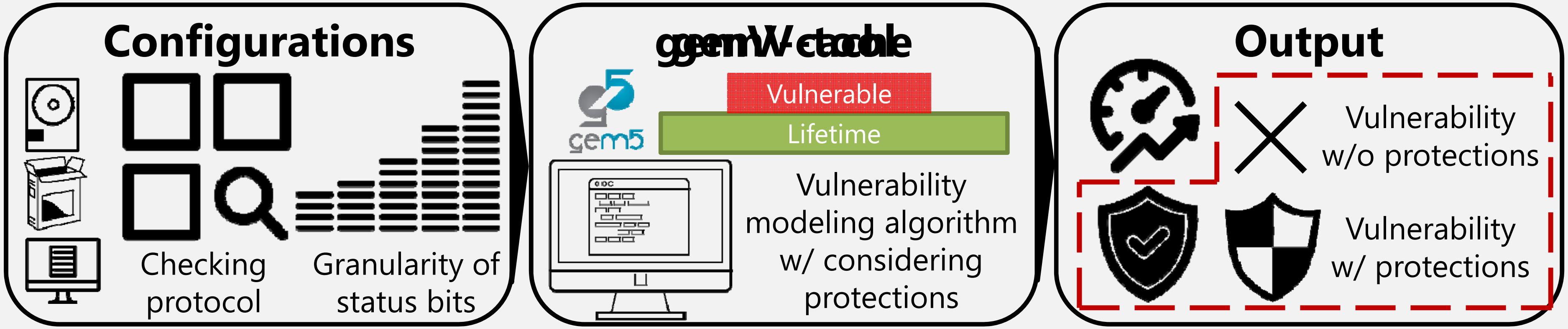
Resiliency
- Vulnerability ($bit \times cycle$)

- Hardware configuration
  - Issue width, ROB size, IQ size, LSQ size
- Software configuration
  - Compiler (gcc, LLVM)
  - Optimization options
  - Algorithm
- System configuration
  - ISAs (ARM, X86, POWER, SPARC)
  - Number of cores

Good for design space exploration in terms of performance and resiliency at the early design phase

How to quantify the effectiveness of protection techniques

- Design guidelines for resilient and efficient parity protected write-back L1 caches
- To do this, we have extended gemV-tool to gemV-cache



Configurations

Checking protocol

Granularity of status bits

gemV-tool

gemV-cache

Vulnerable

Lifetime

Vulnerability modeling algorithm w/ considering protections

Output

Vulnerability w/o protections

Vulnerability w/ protections

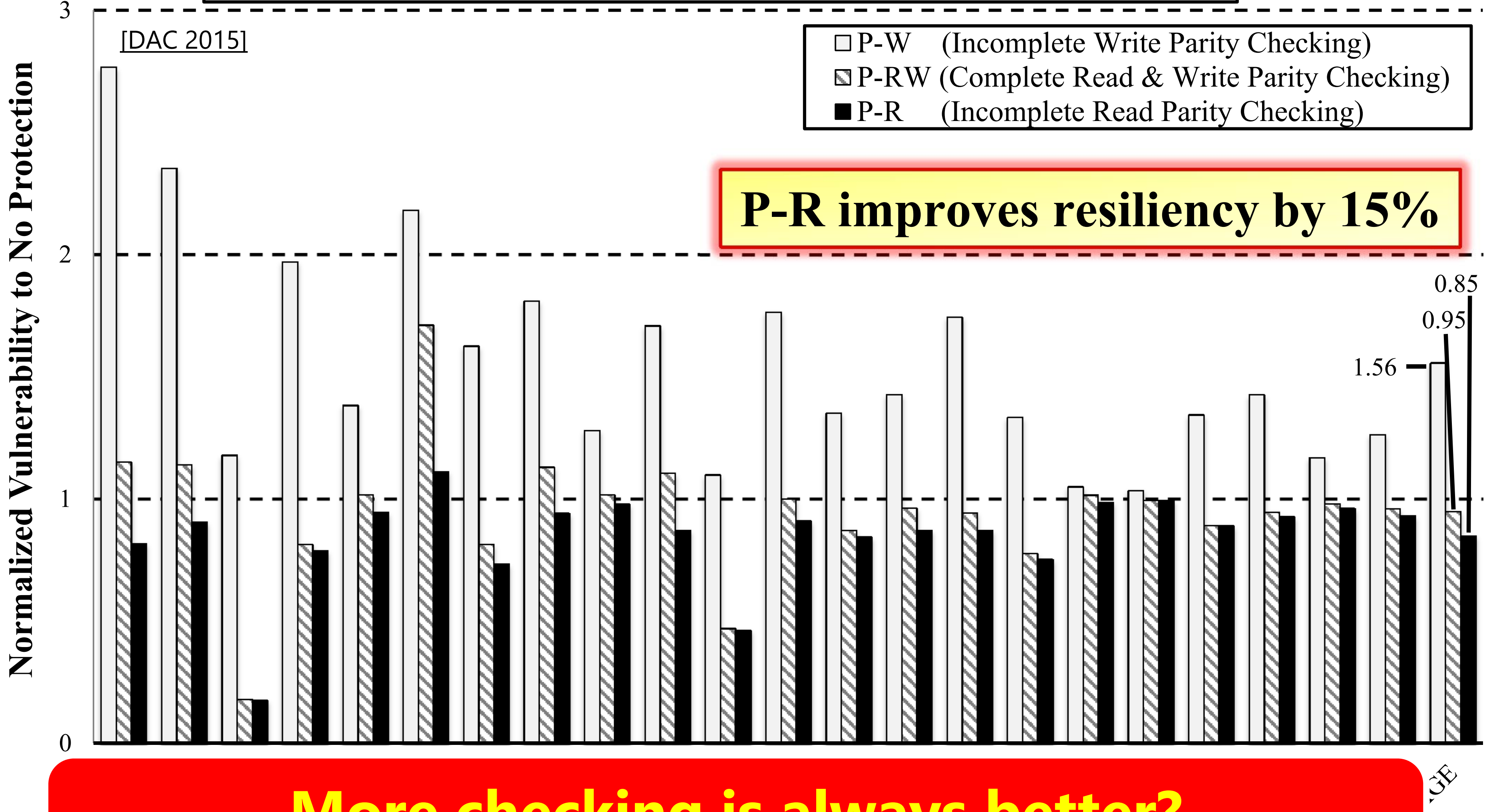
[DAC 2015]

- Design questions:
  - When to check for parity
    - At Reads
    - At Writes
    - At both Reads and Writes
  - Granularity of status bits
    - Block level
    - Word level

[DAC 2015] **Yohan Ko**, Reiley Jeyapaul, Youngbin Kim, Kyoungwoo Lee, and Aviral Shrivastava. Guidelines to design parity protected write-back L1 data cache. Design Automation Conference (DAC). 2015.

When should parity be checked?

Vulnerability of Parity-protected Cache: Checking Protocol



P-R improves resiliency by 15%

More checking is always better?

NO. P-R provides better resiliency than P-RW

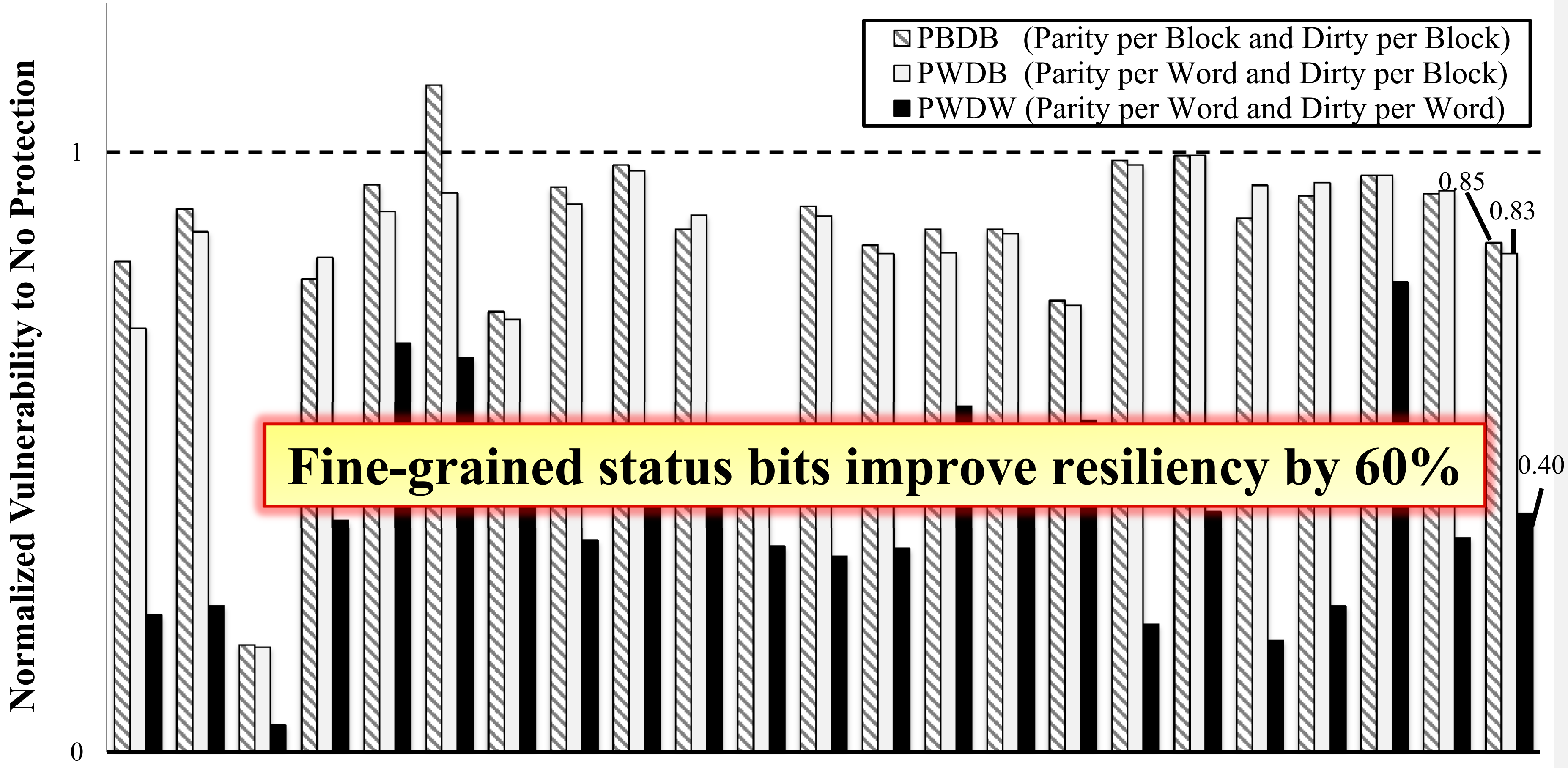
[DAC

write-back L1 data cache. Design Automation Conference (DAC). 2015.

At what granularity must we implement status bits?

[DAC 2015]

Vulnerability of Parity-protected Cache: Status Bits



Fine-grained status bits improve resiliency by 60%

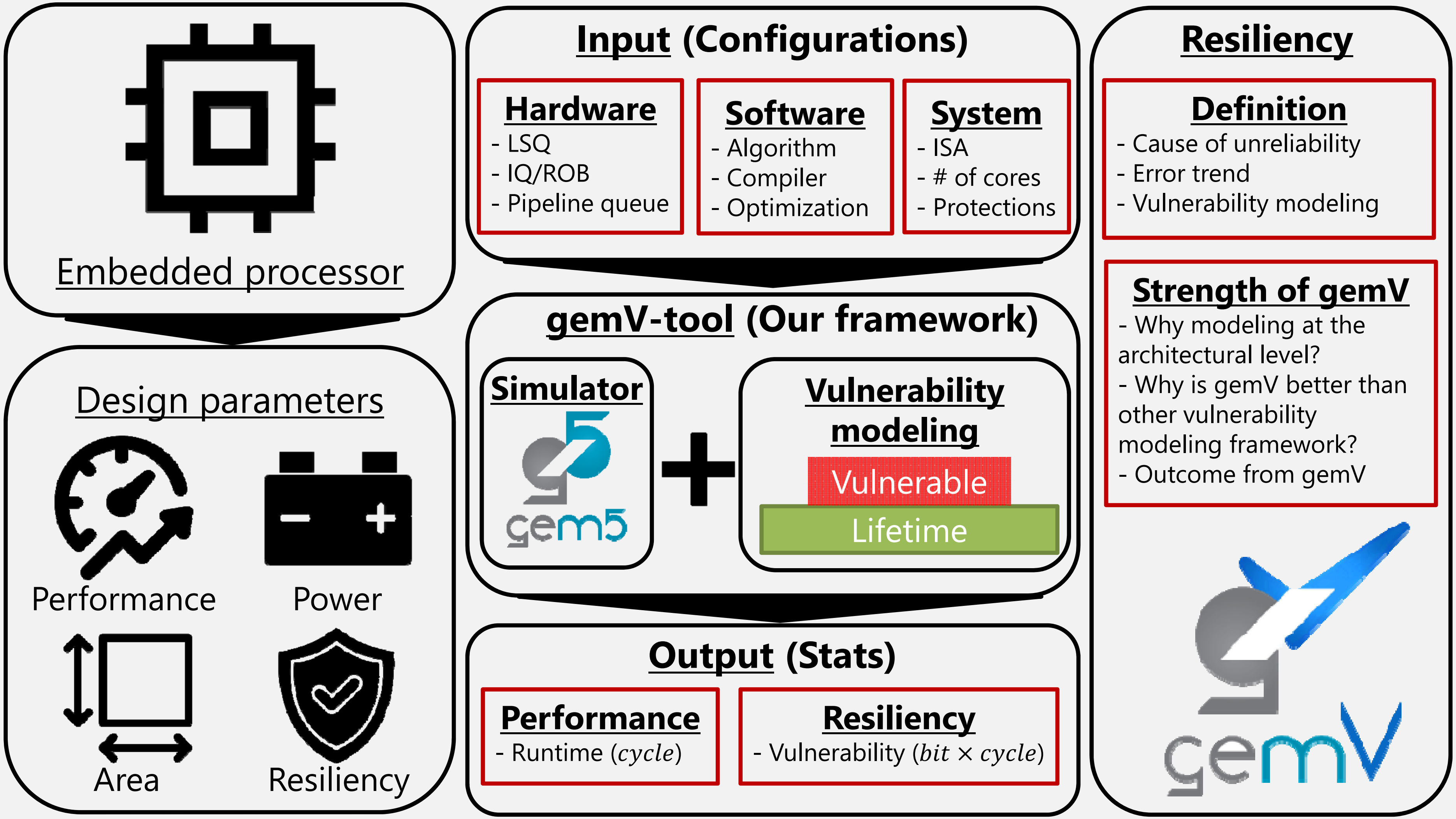
Finer granularity is always better?
YES. But, there is no medium granularity for protection

write-back L1 data cache. Design Automation Conference (DAC). 2015.

Outline

- Thesis reminder
- Comments from the previous presentation
  - Error modeling
  - Strength of gemV-tool
  - Use of gemV-tool
- Response to comments
- Conclusion

1st comment: Need of concrete error modeling



Embedded processor

Design parameters

Performance

Power

Area

Resiliency

Input (Configurations)

Hardware
- LSQ
- IQ/ROB
- Pipeline queue

Software
- Algorithm
- Compiler
- Optimization

System
- ISA
- # of cores
- Protections

gemV-tool (Our framework)

Simulator

Vulnerability modeling

Output (Stats)

Performance
- Runtime ($cycle$)

Resiliency
- Vulnerability ($bit \times cycle$)

Resiliency

Definition
- Cause of unreliability
- Error trend
- Vulnerability modeling

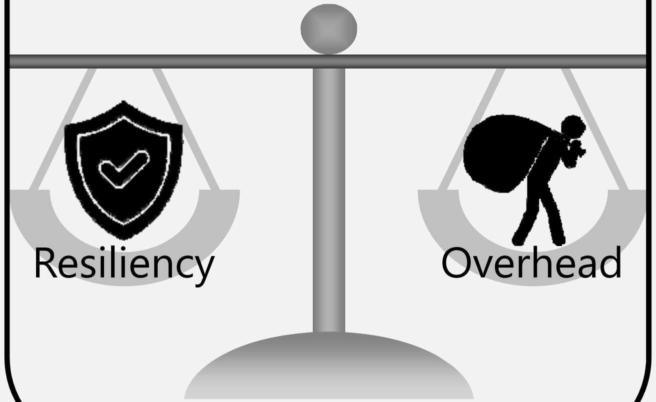
Strength of gemV
- Why modeling at the architectural level?
- Why is gemV better than other vulnerability modeling framework?
- Outcome from gemV

2nd comment: What can gemV-tool do?

Embedded processor

Protection technique

Trade-off relationship

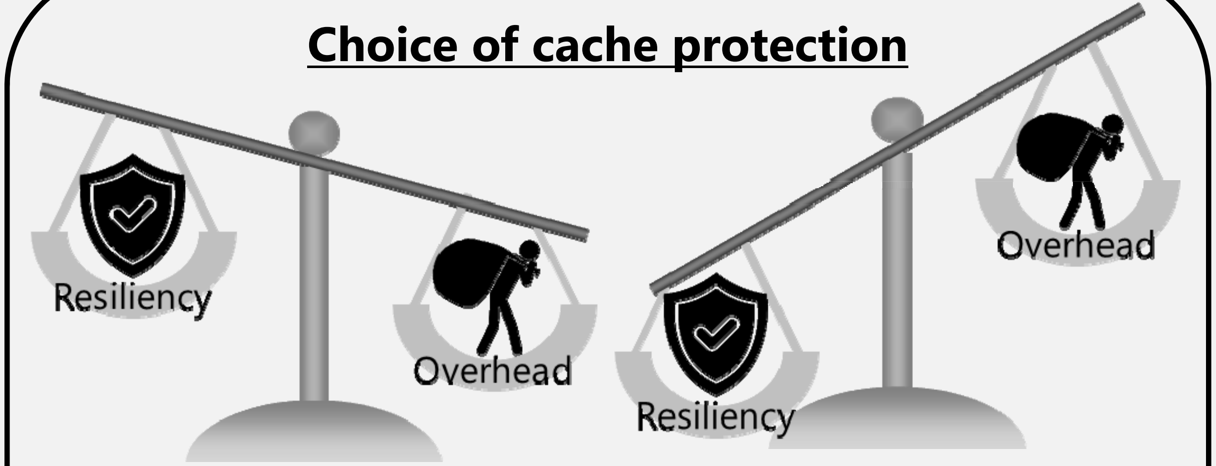


Parity protection guideline for L1 data cache

Checking protocol:
Checking at Read > Checking at Read & Write

Granularity of status bits (dirty and parity):
Coarse ≈ Medium < Fine

Choice of cache protection



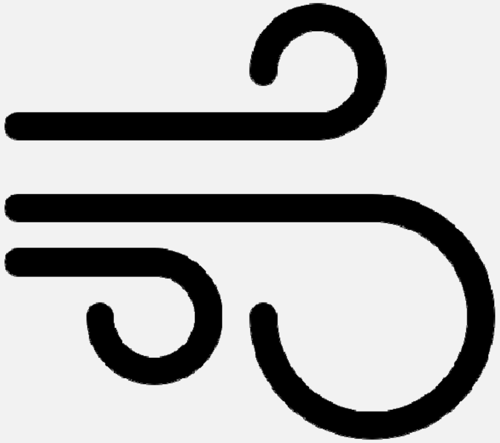
Parity protection:
Comparable overhead,
but not perfect reliability

ECC protection:
Huge overhead,
but perfect reliability

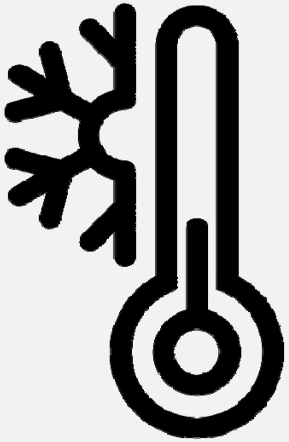
Outline

- Thesis reminder
- Comments from the previous presentation
- Response to comments
  - – Our soft error model
  - – Strength of our gemV-tool
  - – Use of gemV-tool to choose protection techniques
- Conclusion

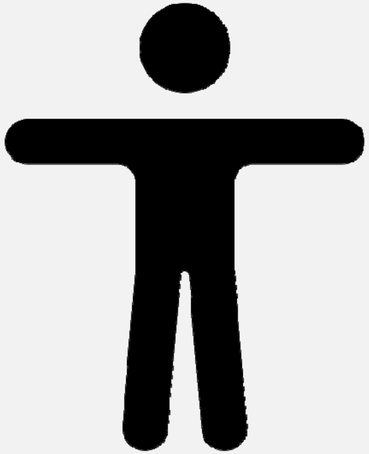
Let’s think about it



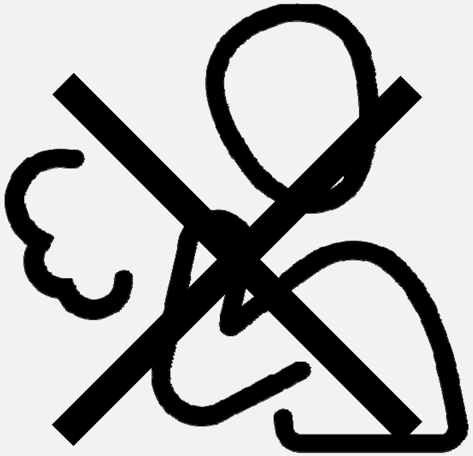
Wind



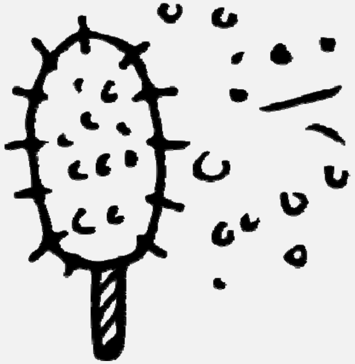
Cold temperature



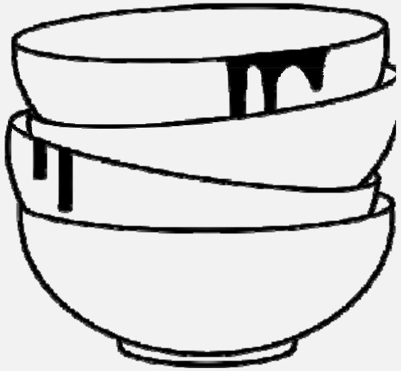
Human body



Catching a cold

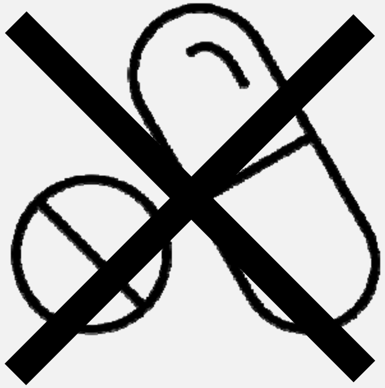


Microdust



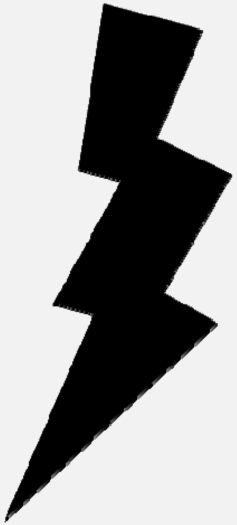
Dirtiness

Nutritional imbalance



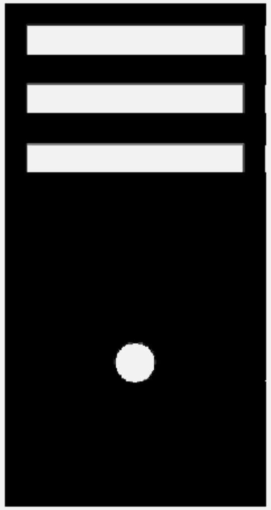
Cure-all medicine

How about computers?

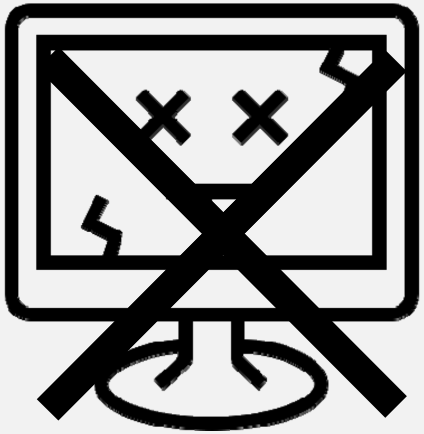


External charges

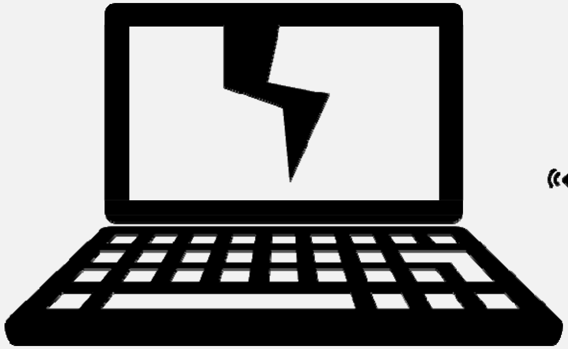
Soft error



Computer system



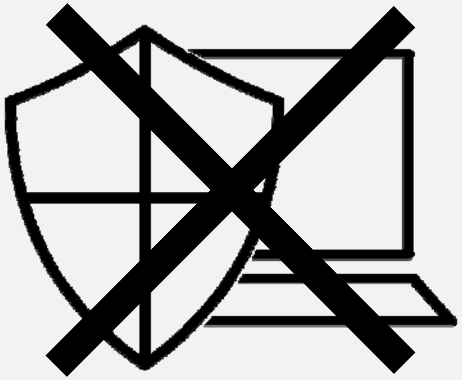
System failure



Hard error

Hacking

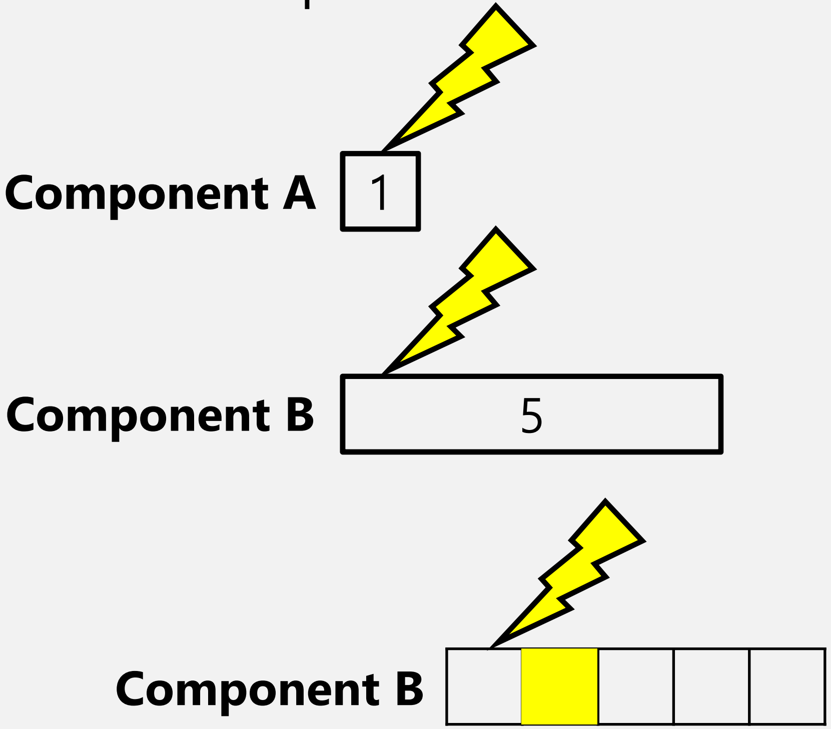
Software bug



Cure-all protection

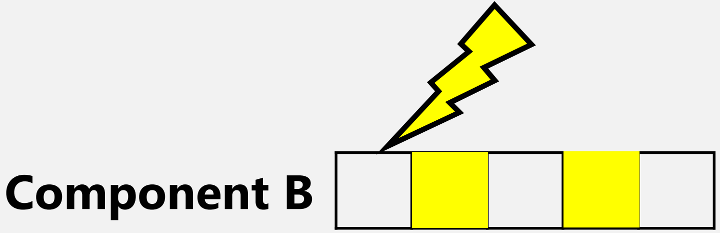
Our soft error model

1. Occurrence of soft errors are proportional to chip size of microarchitectural components [TACO 2013]
2. External charge usually induces single-bit soft errors, not multiple-bit soft errors [TECS 2016]



Number of soft errors
= Soft error rate × area × execution time
$= \alpha \times 1 \times \beta$

Number of soft errors
= Soft error rate × area × execution time
$= \alpha \times 5 \times \beta$



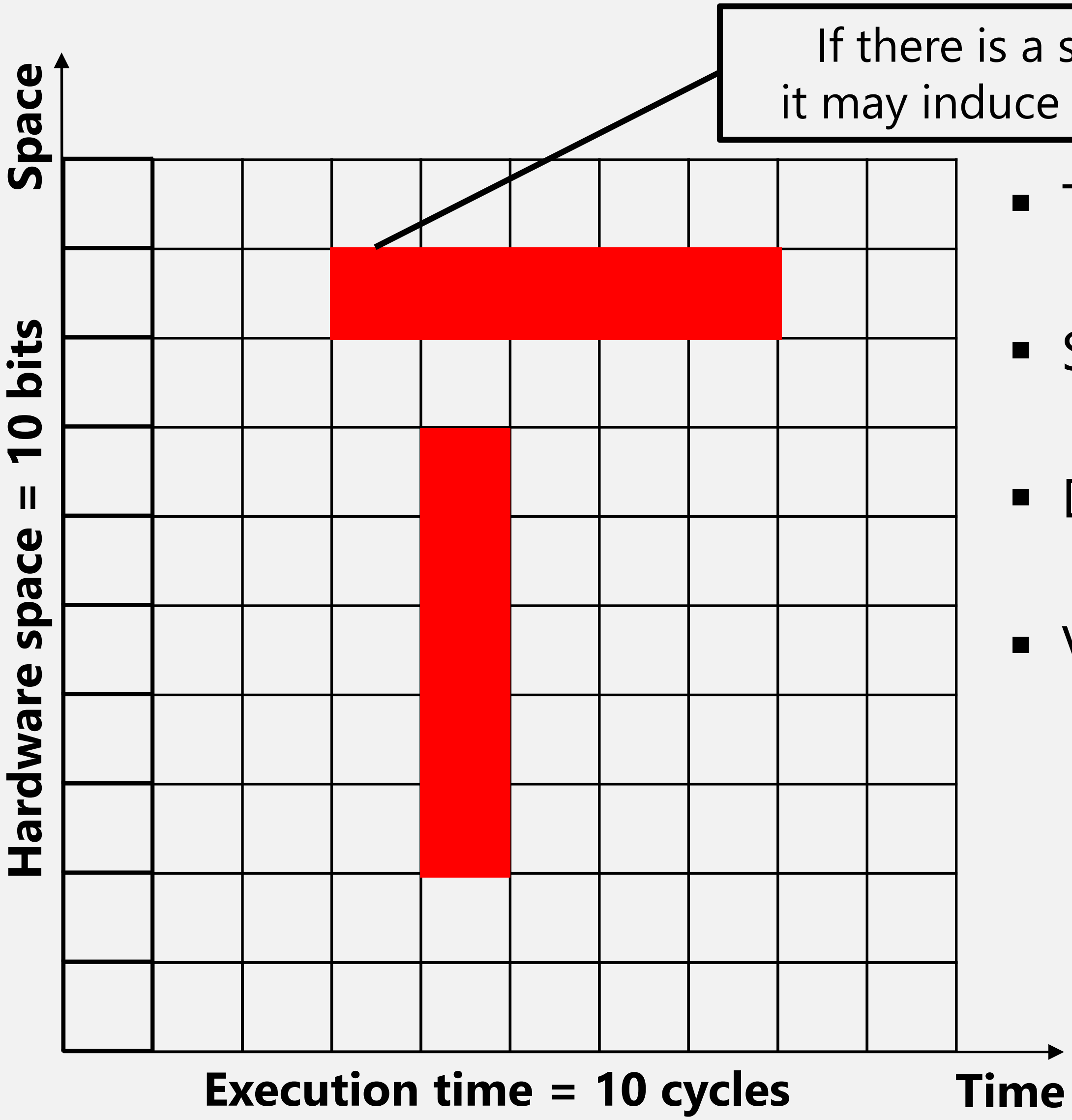
Soft error rate = α

Soft error rate = $^{1}/_{100} \times \alpha$

[TACO 2013] Jongwon Lee, **Yohan Ko**, Kyoungwoo Lee, Jonghee M. Youn, and Yunheung Paek. Dynamic code duplication with vulnerability awareness for soft error detection on VLIW architectures. ACM Transactions on Architecture and Code Optimization (TACO). 9, 4, Article 48. January 2013.

[TECS 2016] **Yohan Ko**, Jihoon Kang, Jongwon Lee, Yongjoo Kim, Joonhyun Kim, Hwisoo So, Kyoungwoo Lee, and Yunheung Paek. Software-Based selective validation techniques for robust CGRAs against soft errors. ACM Transactions on Embedded Computing Systems (TECS). 15, 1, Article 20. January 2016

What is vulnerability?


If there is a single-bit flip,
it may induce system failures
Space
Hardware space = 10 bits
Execution time = 10 cycles
Time

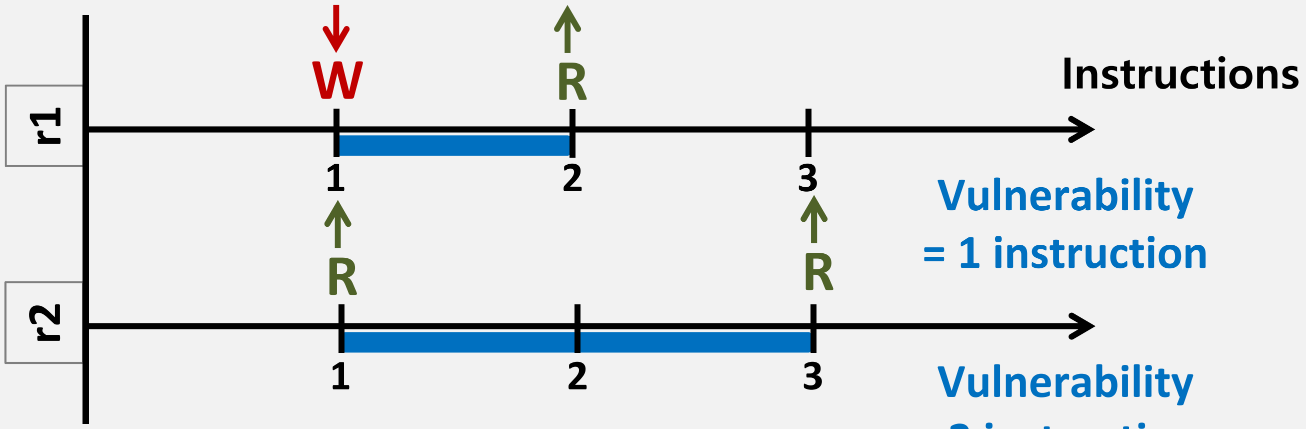
- Temporal domain
  - Execution time
- Spatial domain
  - Hardware bits
- Design space
  - 10 × 10 = 100 bit × cycles
- Vulnerability [MICRO 2003]
  - 5 + 5 = 10 bit × cycles

[MICRO 2003] Shubhendu S. Mukherjee, Christopher Weaver, Joel Emer, Steven K. Reinhardt, and Todd Austin. A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor. International Symposium on Microarchitecture (MICRO). 2003.

Vulnerability modeling at the architectural level

```
1:    ADD r1, r2, r3
2:    SUB r5, r1, r4
3:    STORE r2, r6
```

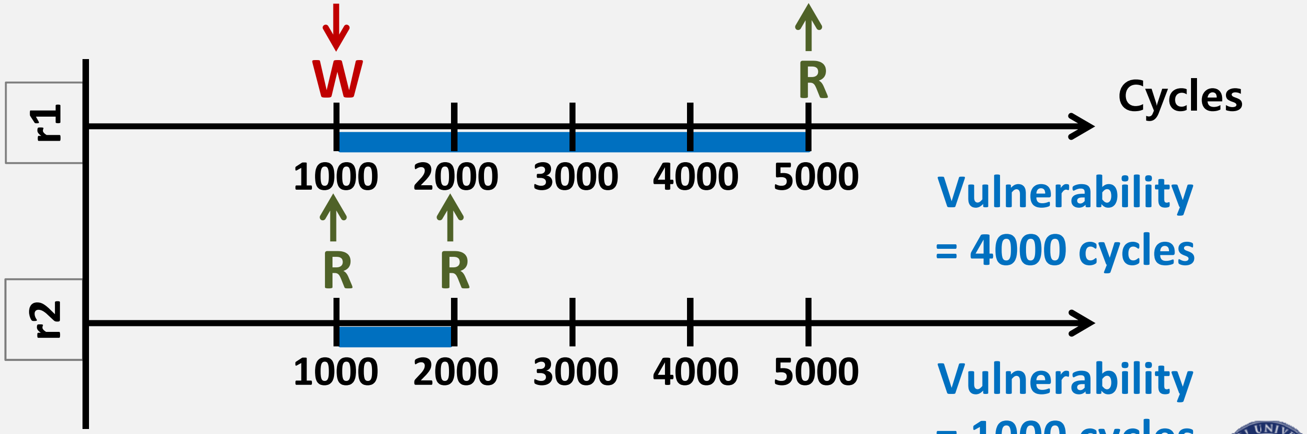
Software [WRFET 2008]



Vulnerability = 1 instruction

Vulnerability = 2 instructions

Architecture



Vulnerability = 4000 cycles

Vulnerability = 1000 cycles

[WRFET 2008] Vilas Sridharan and David R. Kaeli. Quantifying software vulnerability. Workshop on Radiation Effects and Fault Tolerance in Nanometer Technologies (WRFET). 2003.

What makes our gemV-tool better?

Configurations

Embedded processor

Vulnerability

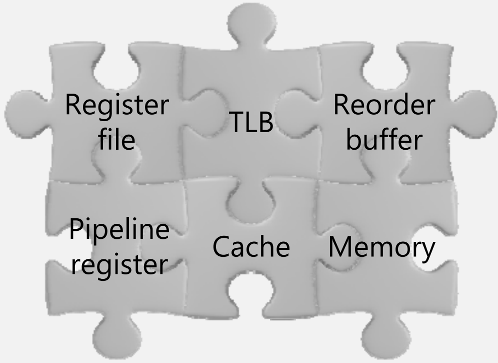
Versatile

Hardware

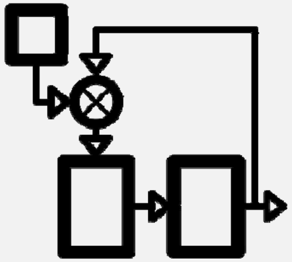
Software

System

Comprehensive



Accurate

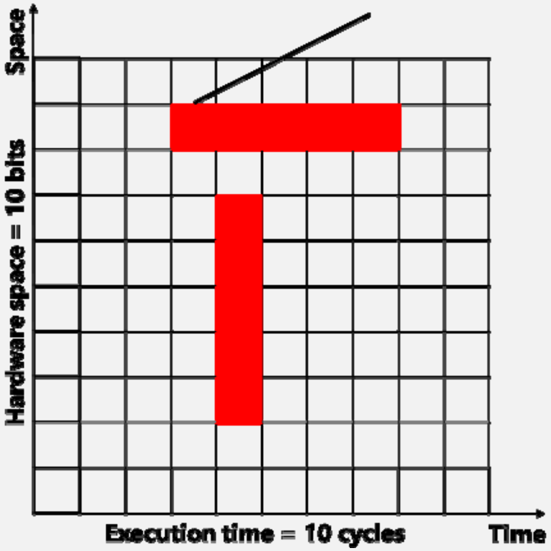


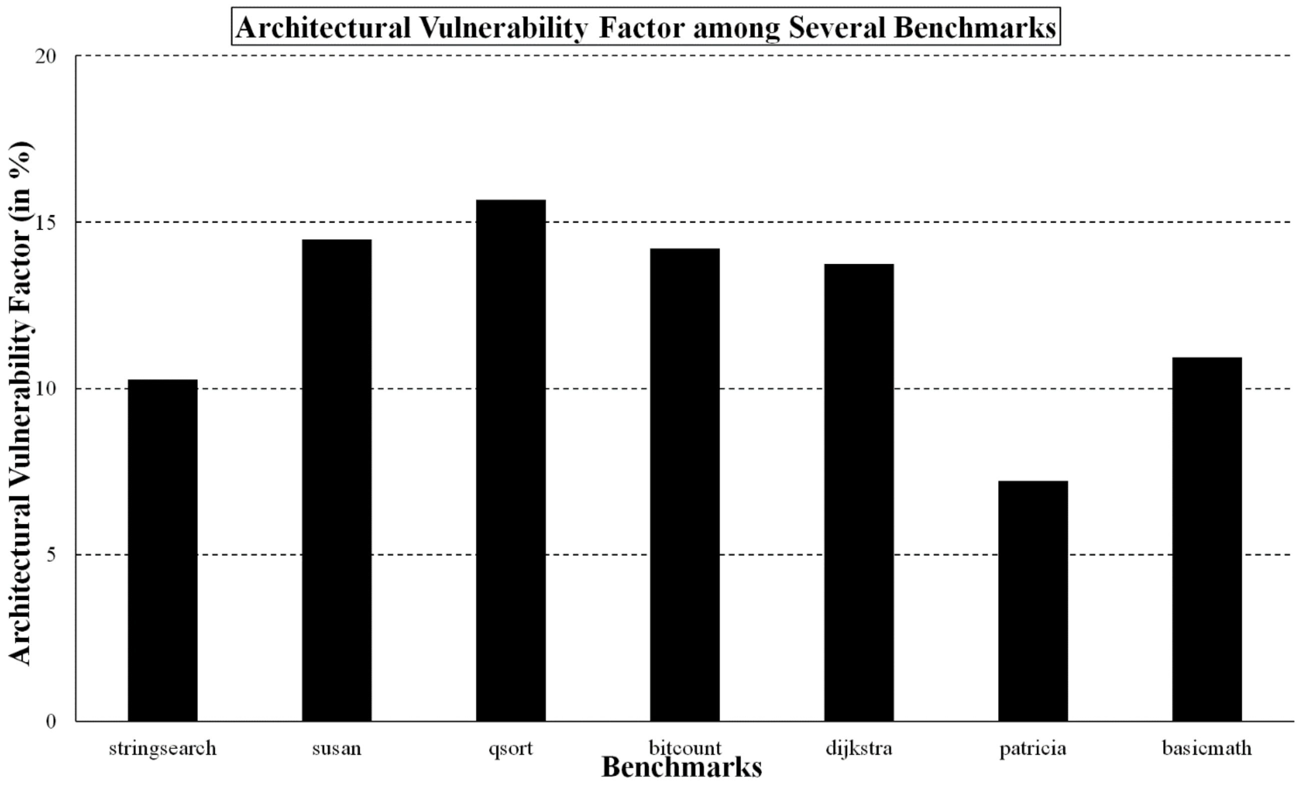
Vulnerability modeling

Fault injection

Outcome from gemV

- What is the probability that a single-bit soft error in a computer system results in system failure?
  - Architectural vulnerability factor
  - $AVF = \frac{\sum Vulnerability\ of\ all\ the\ system\ components}{System\ hardware\ bits \times Execution\ time} = \frac{5+5}{10 \times 10} = 10(\%)$


Space
Hardware space = 10 bits
Execution time = 10 cycles
Time


Architectural Vulnerability Factor among Several Benchmarks
Architectural Vulnerability Factor (in %)
stringsearch
susan
qsort
bitcount
dijkstra
patricia
basicmath
Benchmarks