Shield:

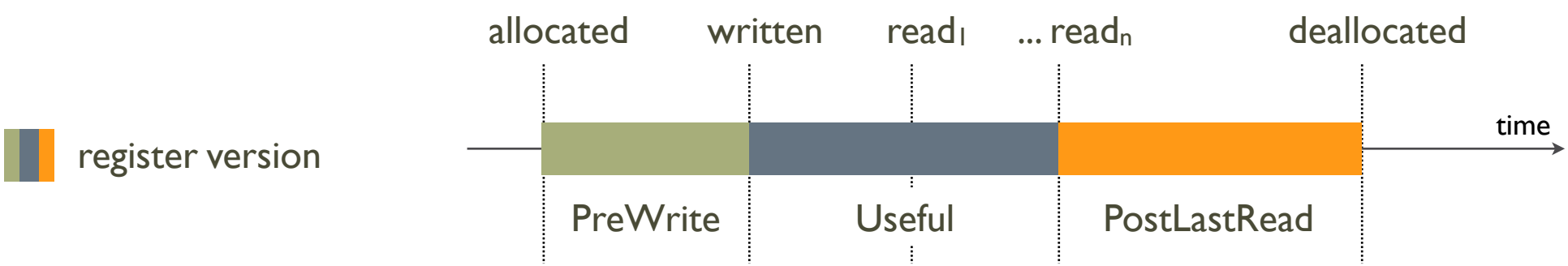
Cost-Effective Soft Error Protection For Register Files

Pablo Montesinos, Wei Liu* and Josep Torrellas

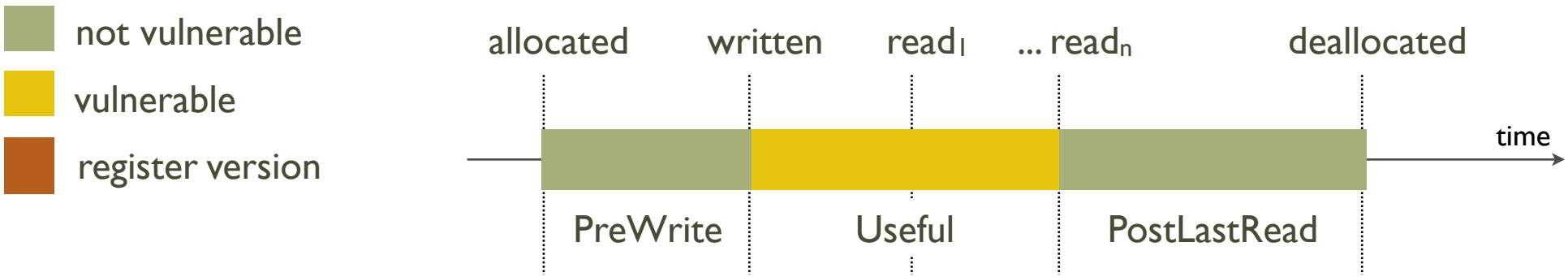
University of Illinois at Urbana-Champaign
*Intel

i-acoma group

Vulnerability analysis of registers



Vulnerability analysis of registers



- Vulnerability metric: AVF (Architectural Vulnerability Factor)
- Register File's AVF: fraction of time its registers are vulnerable
- A register is vulnerable if a change in its value will produce an error
- Only the Useful period needs to be protected

How vulnerable is the register file?

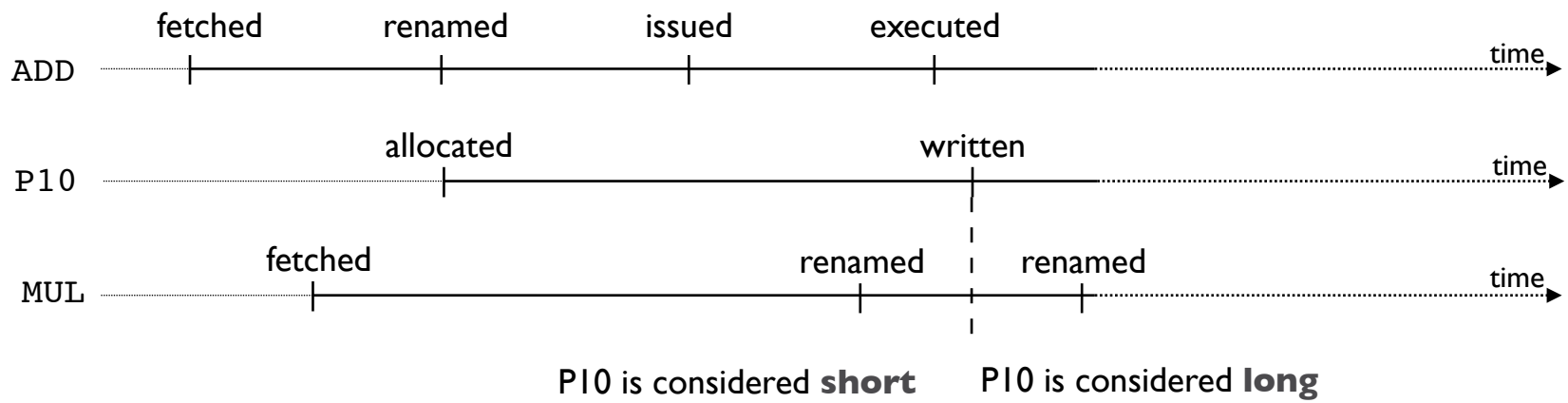
- What fraction of a register's lifetime is in useful state?
  - 22% for SPECint and 15% for SPECfp **
- How many registers are in useful state at a given time?
  - 20 for SPECint and 17 for SPECfp **
- Are all the register versions equally vulnerable?

**(128 physical registers)

Short and long register versions

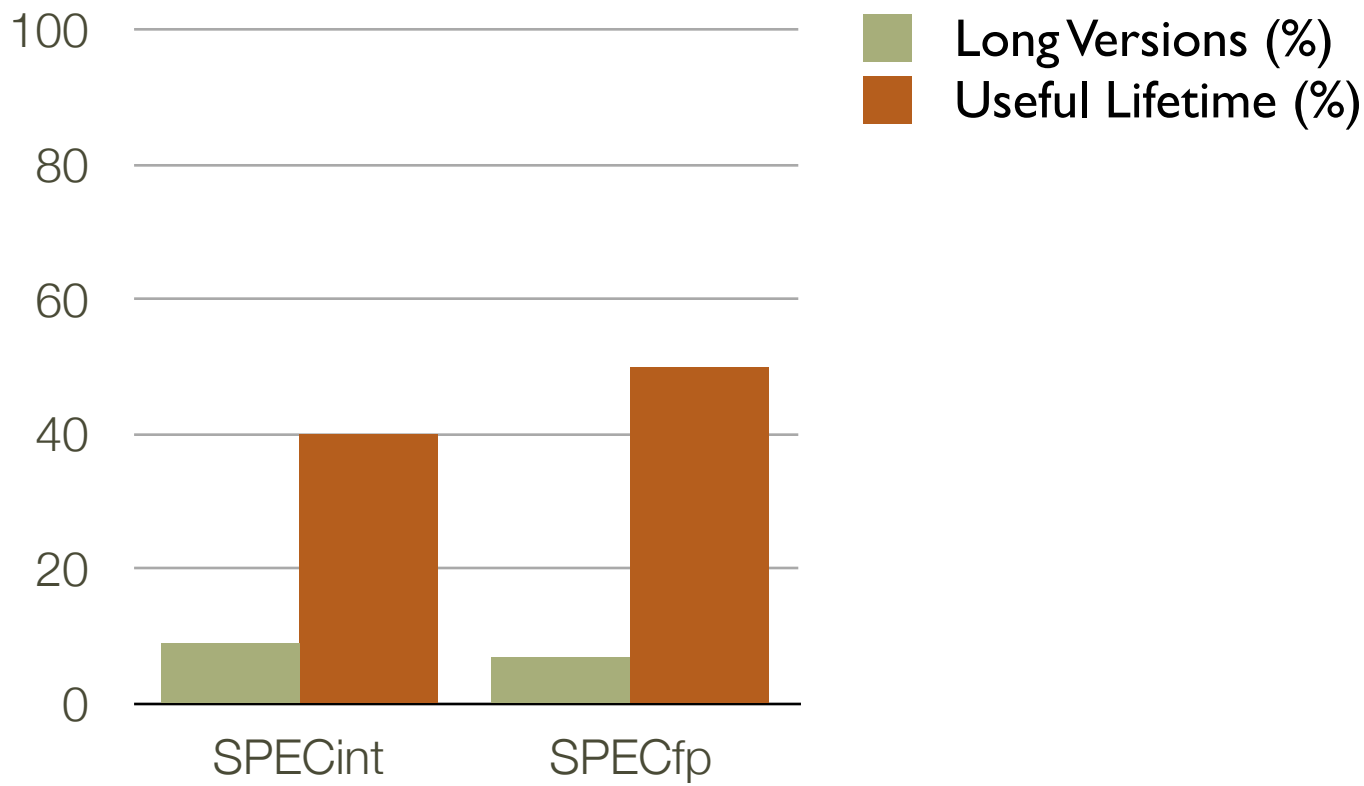
Ponomarev *et al*, 2004

Original Code	Renamed Code
`ADD R1, ,`	`ADD P10, ,`
`...`	`...`
`MUL R1, ,`	`MUL P20, ,`



- Short version: a new instruction renames the register before it is written

Characterizing long and short versions



- A few registers account for most of the useful lifetime of the register file

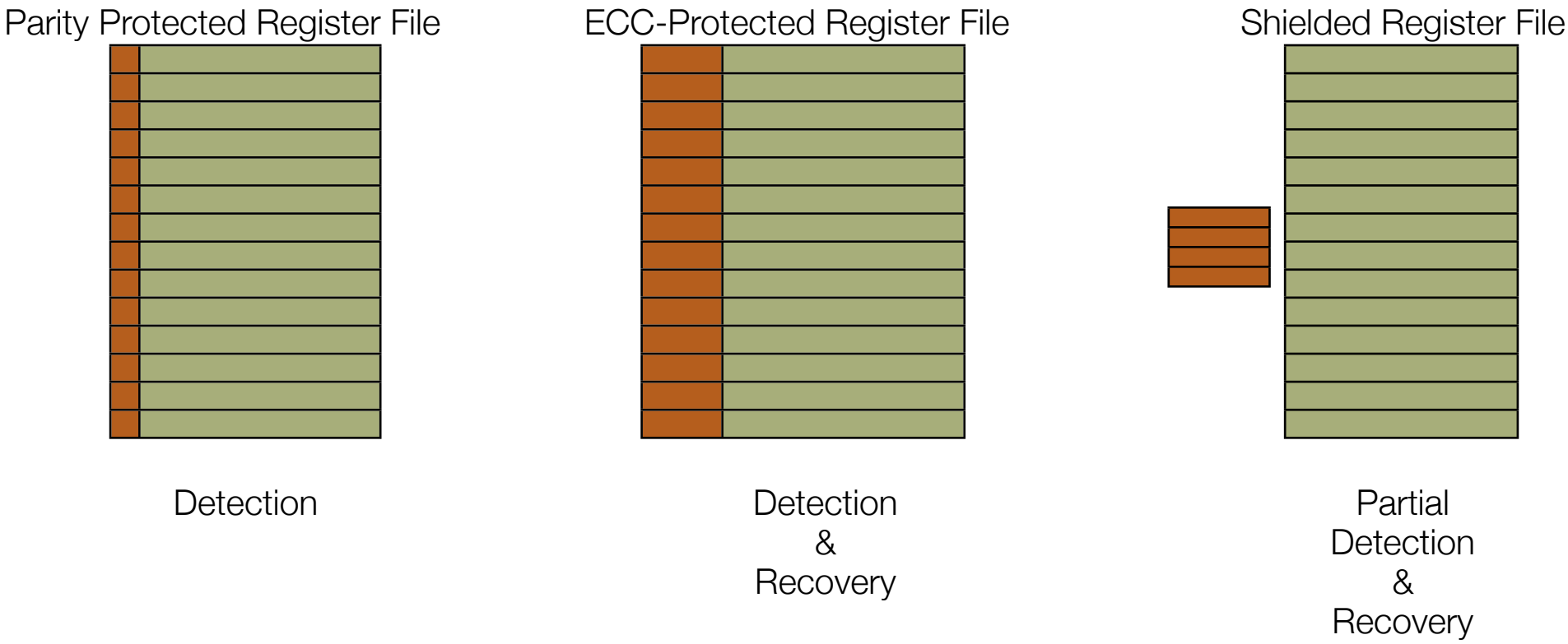
How vulnerable is the register file?

- What fraction of a register's lifetime is in useful state?
- How many registers are in useful state at a given time?
- Are all the register versions equally vulnerable?

How vulnerable is the register file?

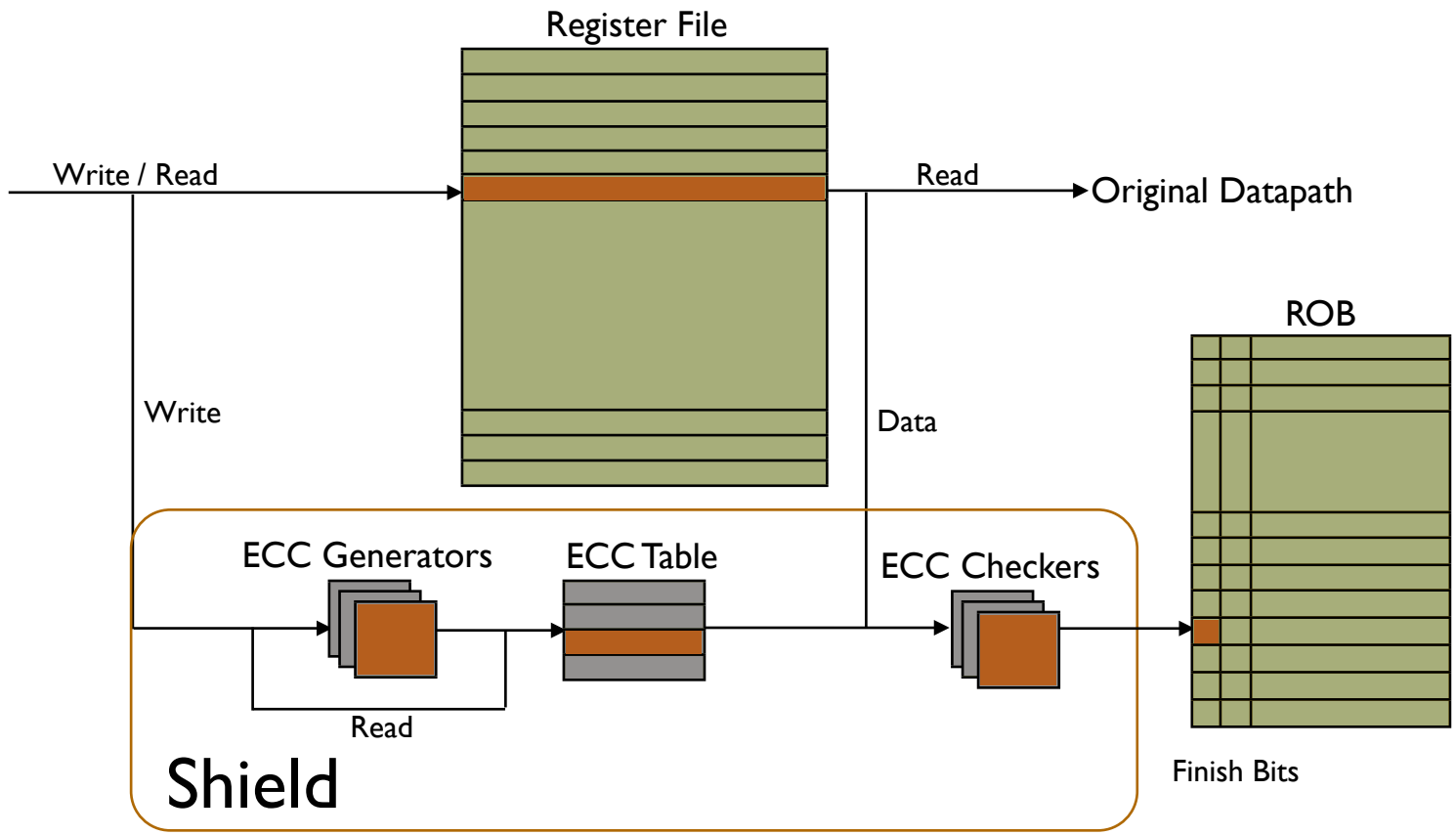
- What fraction of a register's lifetime is in useful state?
  - A small fraction
- How many registers are in useful state at a given time?
  - Just a few
- Are all the register versions equally vulnerable?
  - No, a few registers account for most of the useful lifetime

Shield's goal: protect the register file



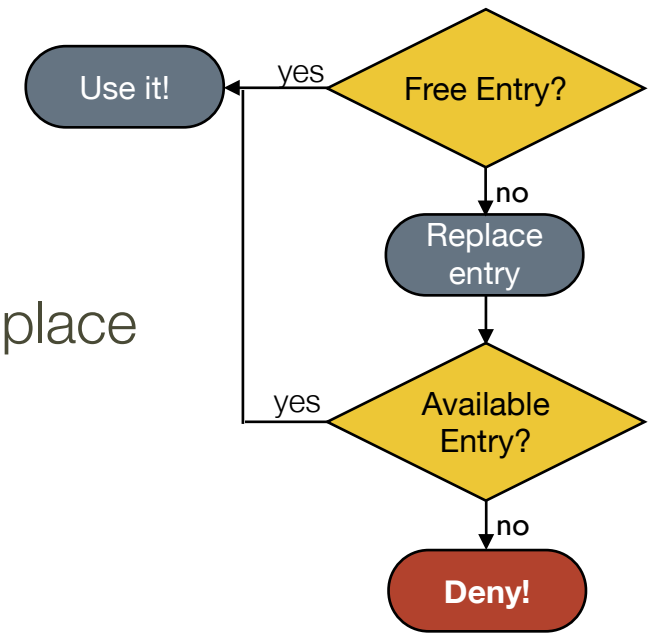
- Defend the register file by only protecting registers in useful state

Shield Architecture



ECC Table entry allocation

- Entries are allocated when registers are written
- Try to minimize the impact on the AVF
  - Use lifespan prediction to choose the entry to replace
  - Replacement policy might deny the allocation



ECC Table entry replacement policies

- Shield uses an extended version of Ponomarev's predictor:
  - Predicts long-lived registers more accurately
  - Avoids protecting dead versions
- The *Effective* replacement policy*:*
  - Replaces versions with same or shorter lifespan
  - Performs aging
- The *OptEffective* replacement policy:
  - *Effective* plus architectural information about the registers
  - Pins registers if known to be long lived (e.g: stack pointer)

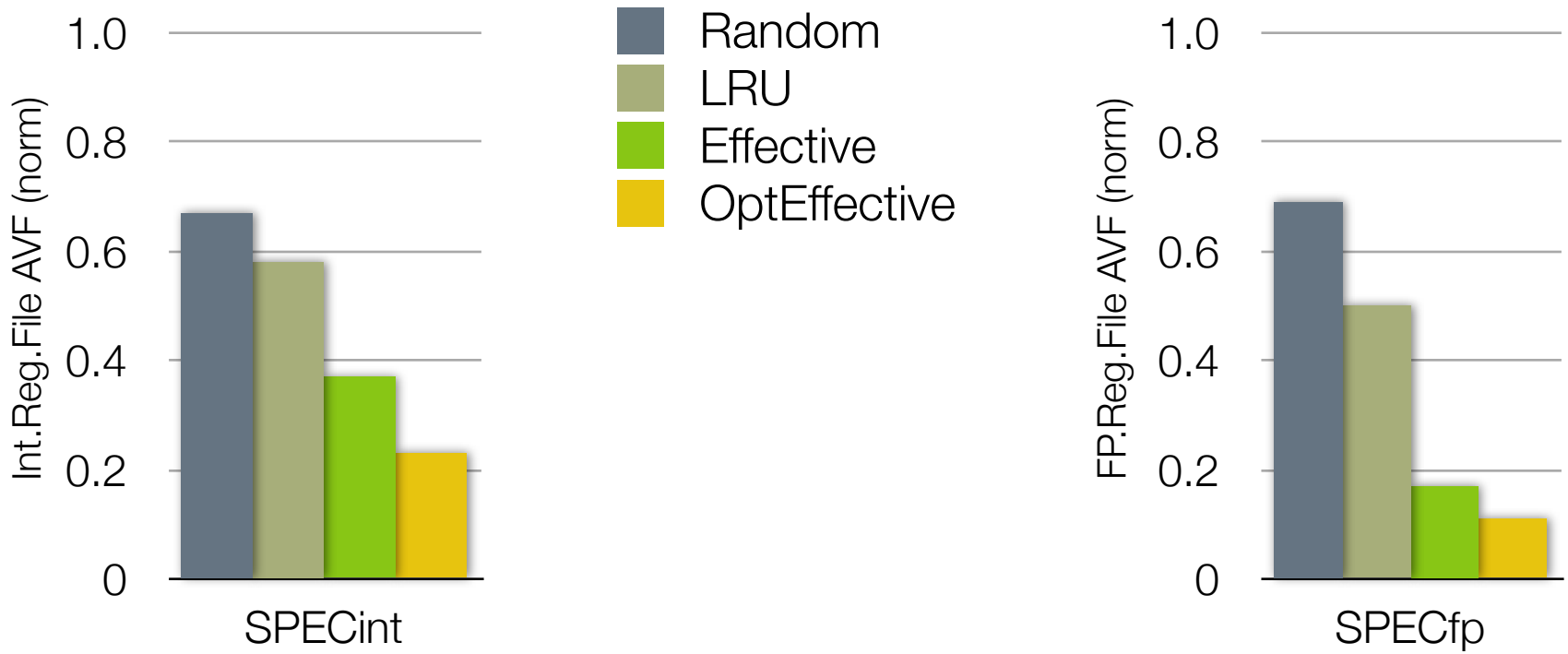
ECC Table entry deallocation

- Entries should be deallocated right after last read, but:
  - Replacement policy may deallocate it earlier
  - Predicting last read for a register impossible
- An entry is stale if it protects a register that won't be read again
  - Protecting stale entries hurts the efficiency of Shield
  - Send eviction signal to the ECC table when some registers are deallocated
- The accuracy of the deallocation will affect the AVF significantly

Evaluation

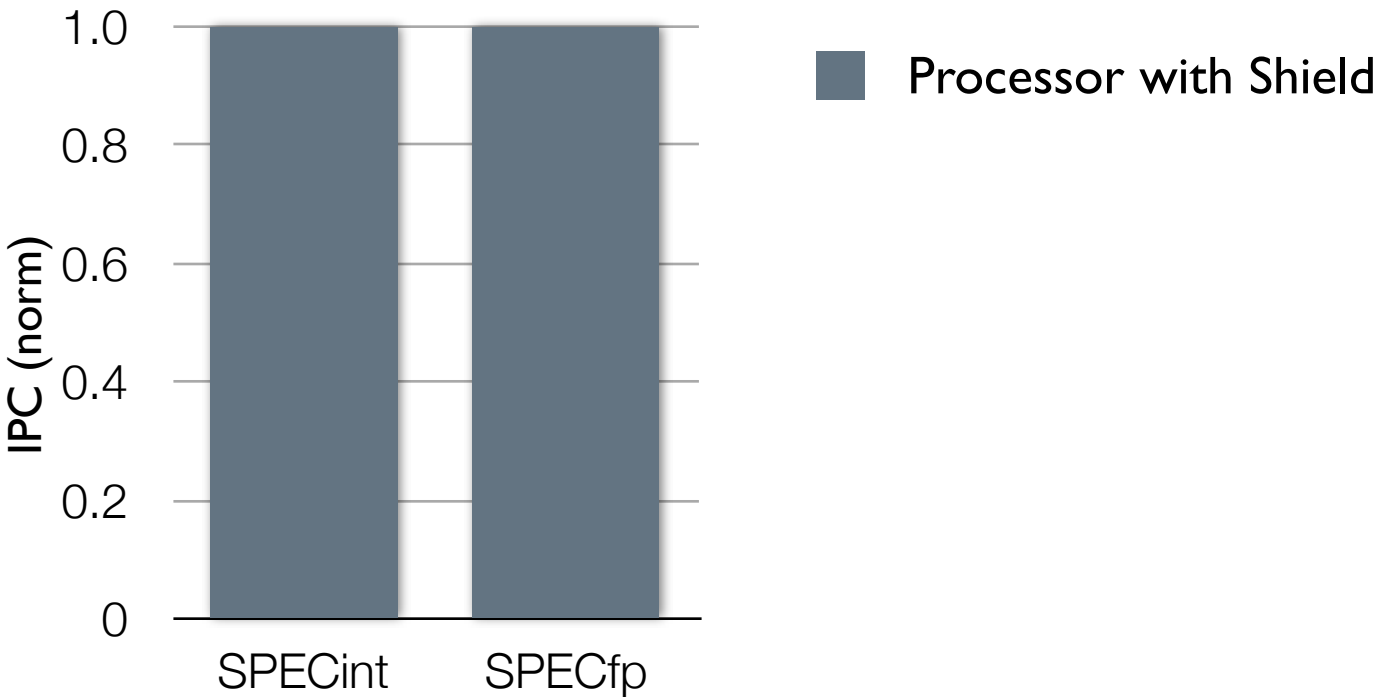
- Simulator: SESC, cycle-accurate execution-driven simulator
- # Registers:
  - Integer: 128
  - Floating point: 64
- # ECC Table entries:
  - Integer: 32
  - Floating point: 16

AVF reduction



- More effective in floating point applications:
  - Register versions` lifespan is more predictable
  - Fewer floating point registers in useful state

Performance impact



- The ROB provides enough slack: no performance loss

Power and Area impact

- Power modeled with a modified CACTI 3.2
- Shield introduces:
  - 45% power overhead over a plain register file
  - 10% area overhead over a plain register file
- Approx to a full parity protection for the register file (Montecito)
- Full parity protection can be added to Shield without much additional overhead

Conclusions and future work

- Shield increases the resistance of register files to soft errors
  - Performance is not affected
  - Modest power and area consumption
  - No need to protect the entire lifetime of the register versions
- Shield reduces the AVF of the register file
  - Integer: Up to 84% (73%on average)
  - FP: Up to 100% (85% on average)
- Our future work includes
  - Compiler support
  - Augment Shield with a parity protected register file

Questions

Shield: Cost-Effective Soft-Error Protection for Register Files

Pablo Montesinos, Wei Liu[+] and Josep Torrellas

University of Illinois at Urbana-Champaign

[+]Intel