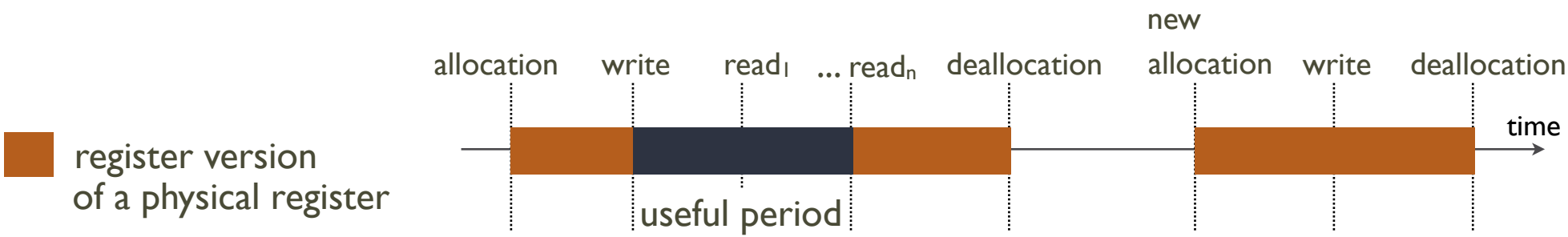
Using Register Lifetime Predictions to Protect Register Files Against Soft Errors

Pablo Montesinos, Wei Liu* and Josep Torrellas

University of Illinois at Urbana-Champaign
*Intel

i-acoma group

Motivation - 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 fault on it translates into an error
  - Assume 1-bit fault model
  - Assume no software masking

Motivation - 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 fault on it translates into an error
  - Assume 1-bit fault model
  - Assume no software masking

Motivation - Vulnerability analysis of registers

Only the useful periods need 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 some register versions more "useful" than others?
  - Yes, some versions are shorter than others

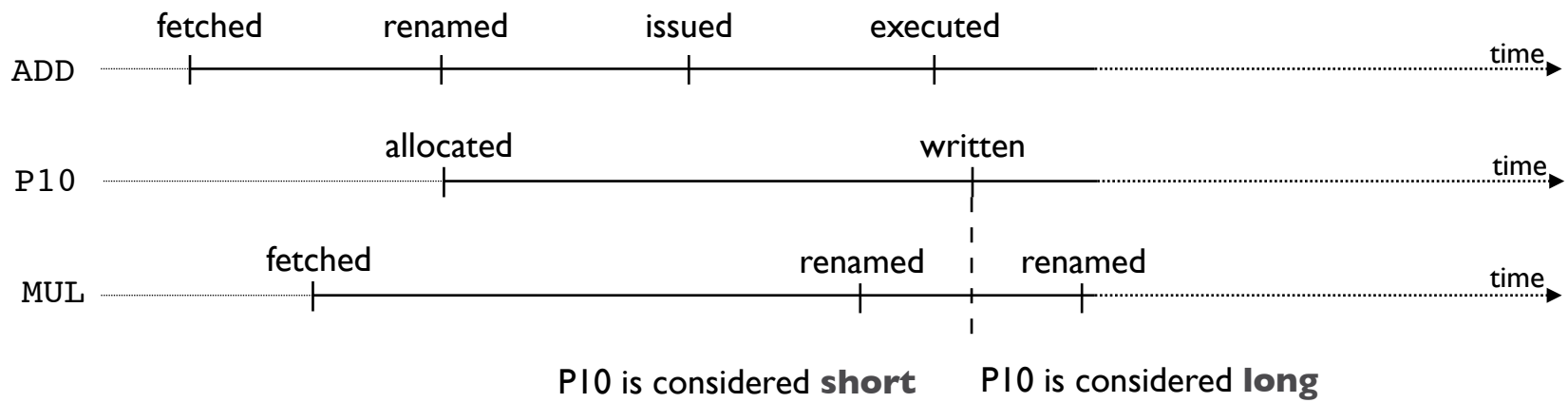
**(128 physical registers)

Short and long register versions

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

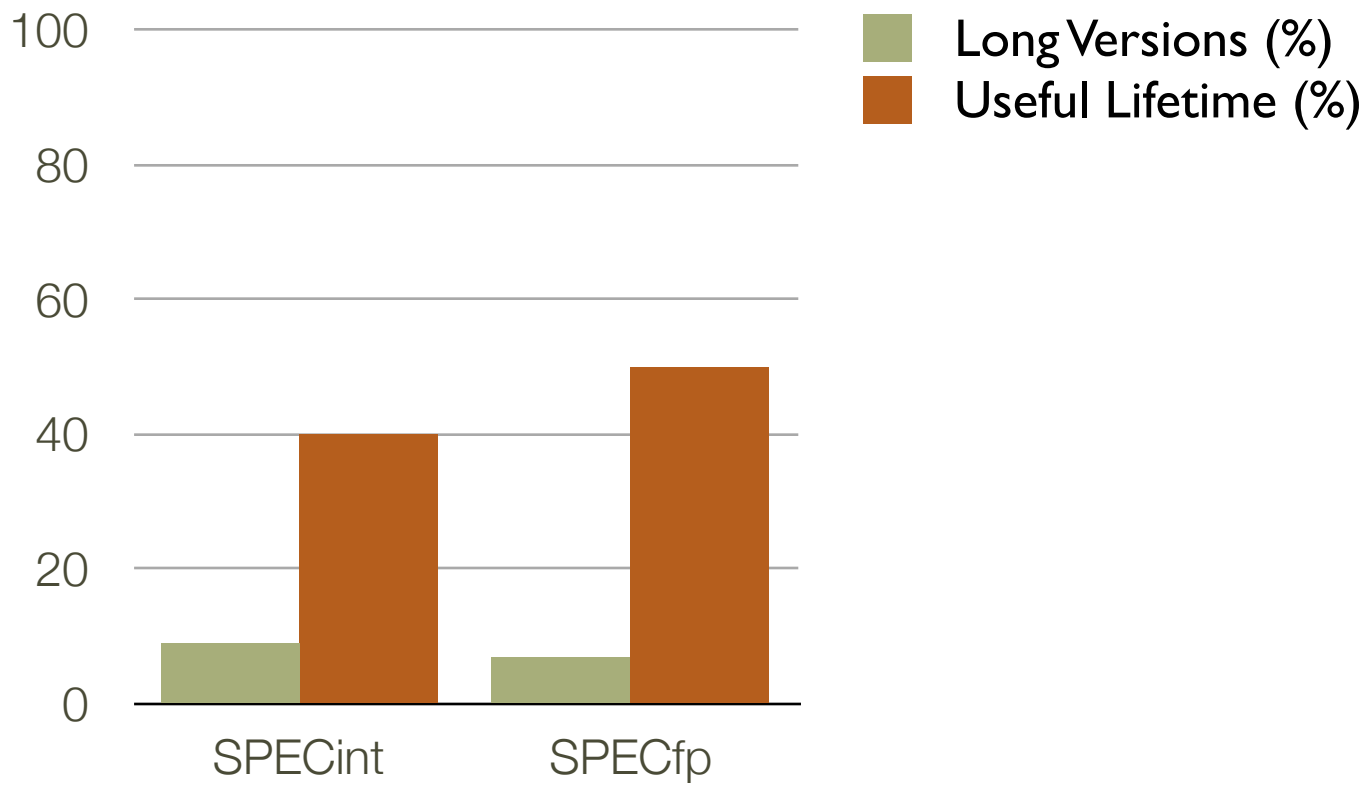
Is P10's version short or long?

Ponomarev *et al*, 2004



- Intuition: P10 knows that nobody will read from it after the MUL inst

Characterizing long and short versions



- A few register versions account for most of the useful lifetime

In summary...

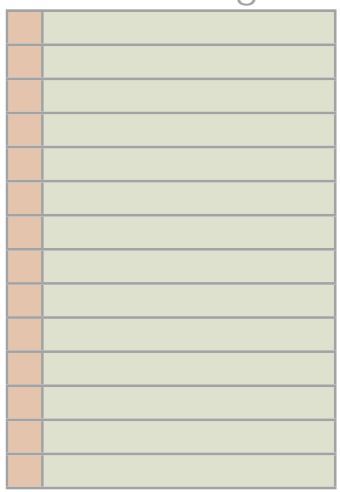
- Only a small fraction of a register's lifetime is useful
  - No need to protect it during its entire lifetime
- Only a few registers are in useful state at a given time
  - No need to protect them all
- A few register versions account for most of the useful lifetime of the register file
  - In case of doubt, protect those that are more "useful"

Our architecture

Shield protects register files against soft errors by only protecting registers during their useful lifetime

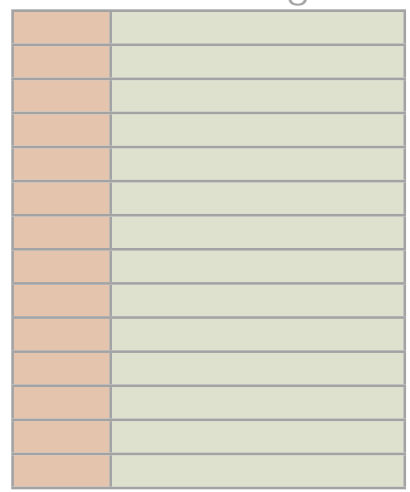
Protecting the register file

Parity Protected Register File



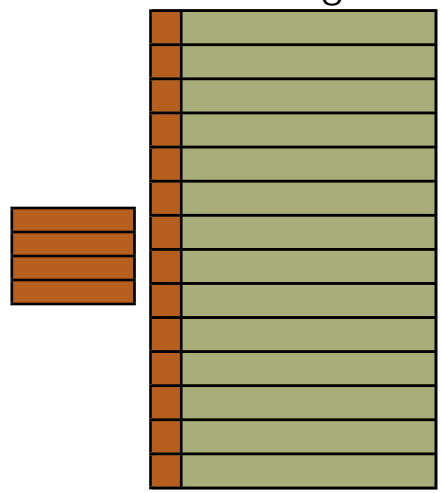
Detection

ECC-Protected Register File



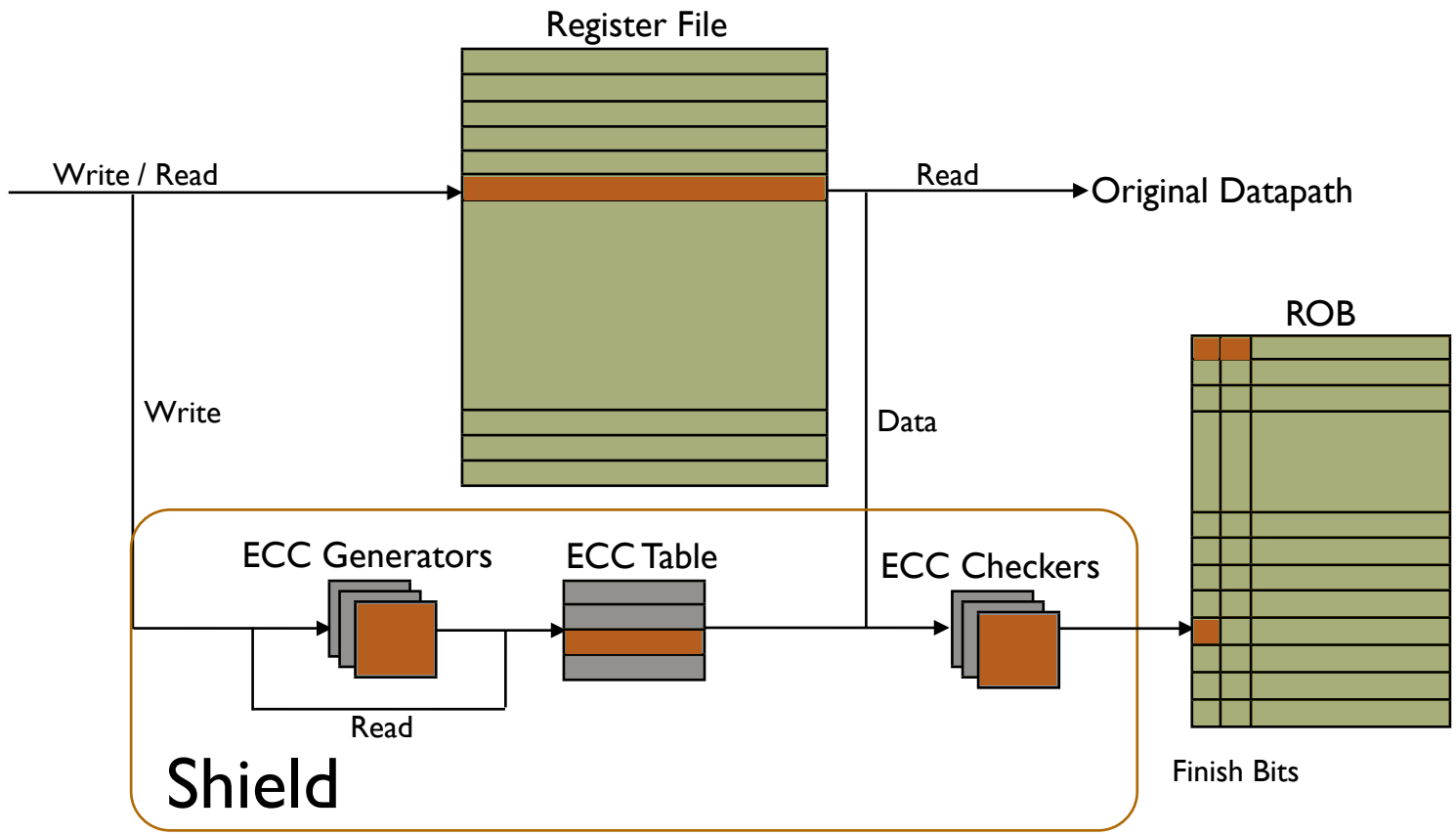
Detection
&
Recovery

Shielded Register File



Detection
&
Partial
Recovery

Shield Architecture



ECC Table entry allocation and deallocation

- Entries are allocated when registers are written
- Entries should be deallocated right after last read
  - Predicting last read for a version is complicated
  - If entry is deallocated before last read:
    - Register version not fully protected
  - If entry is deallocated after last read:
    - Entry becomes stale
    - Protecting stale entries hurts Shield's efficiency
    - Eviction signal sent to Shield when some registers are deallocated

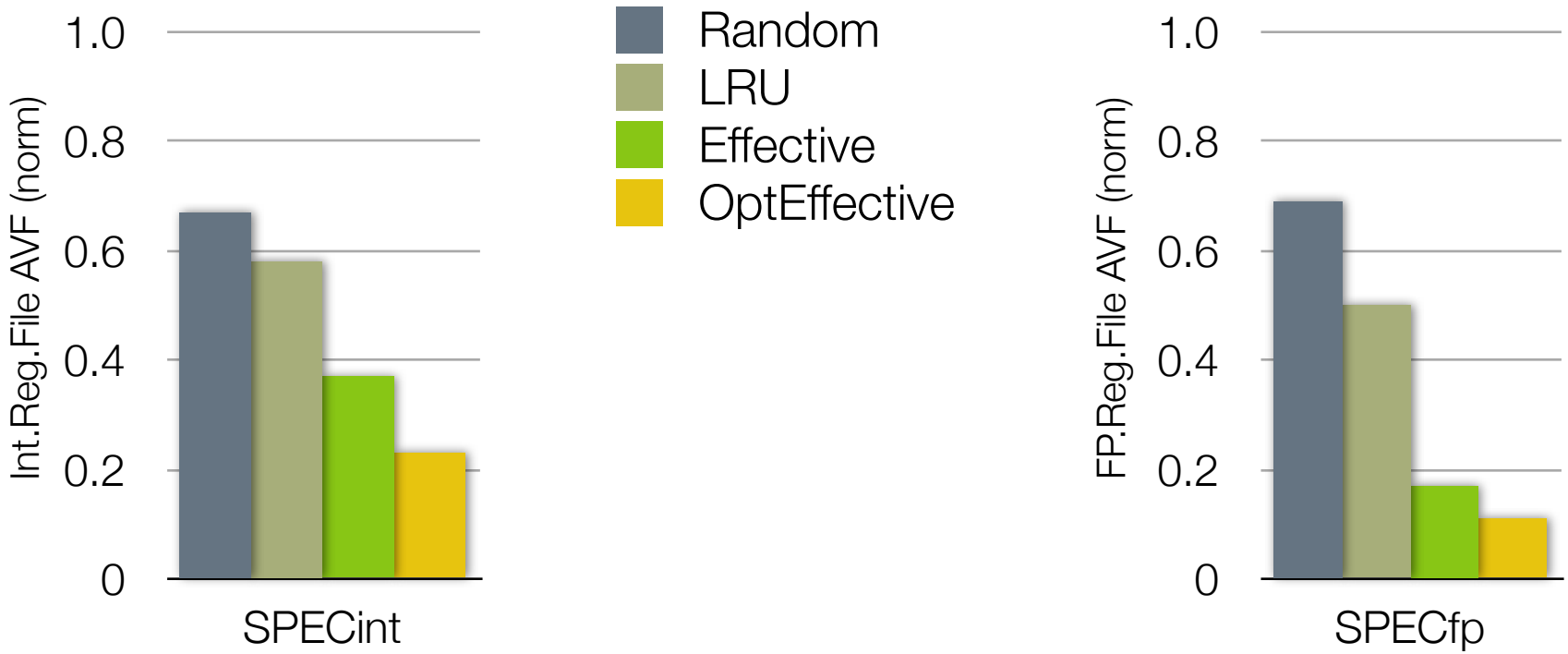
ECC Table entry replacement policies

- Shield uses an extended version of Ponomarev's predictor:
  - Predicts long-lived registers more accurately
  - Avoids protecting registers that are never read
- The *Effective* replacement policy*:*
  - Replaces versions with same or shorter lifespan
  - Example: short versions can only replace short or empty versions
- The *OptEffective* replacement policy:
  - *Effective* plus architectural information about the registers
  - Pins registers if known to be long lived (e.g: stack pointer)

Evaluation

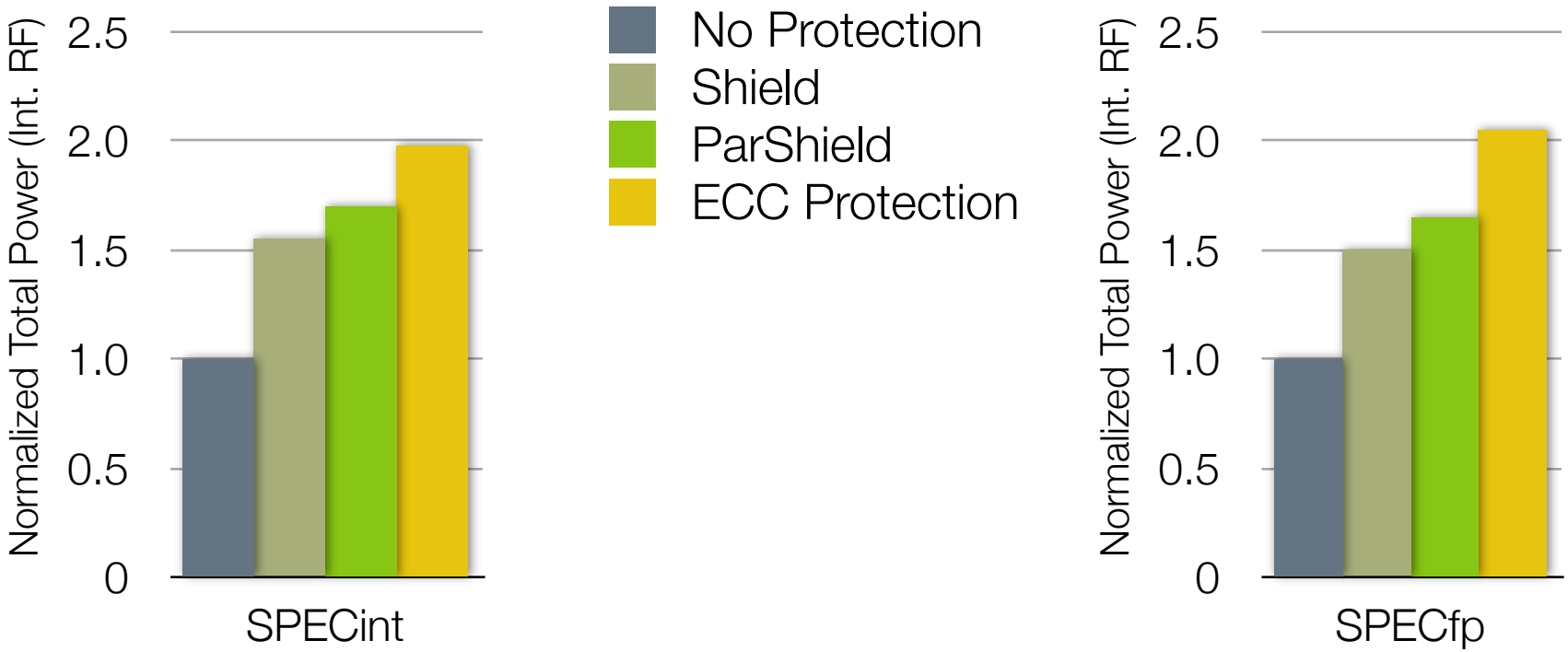
- Simulator: SESC, cycle-accurate execution-driven simulator
- # Registers:
  - Integer: 128
  - Floating point: 64
- # ECC Table entries:
  - Integer: 32
  - Floating point: 16
- Power modeled with a modified CACTI 4.0

AVF reduction



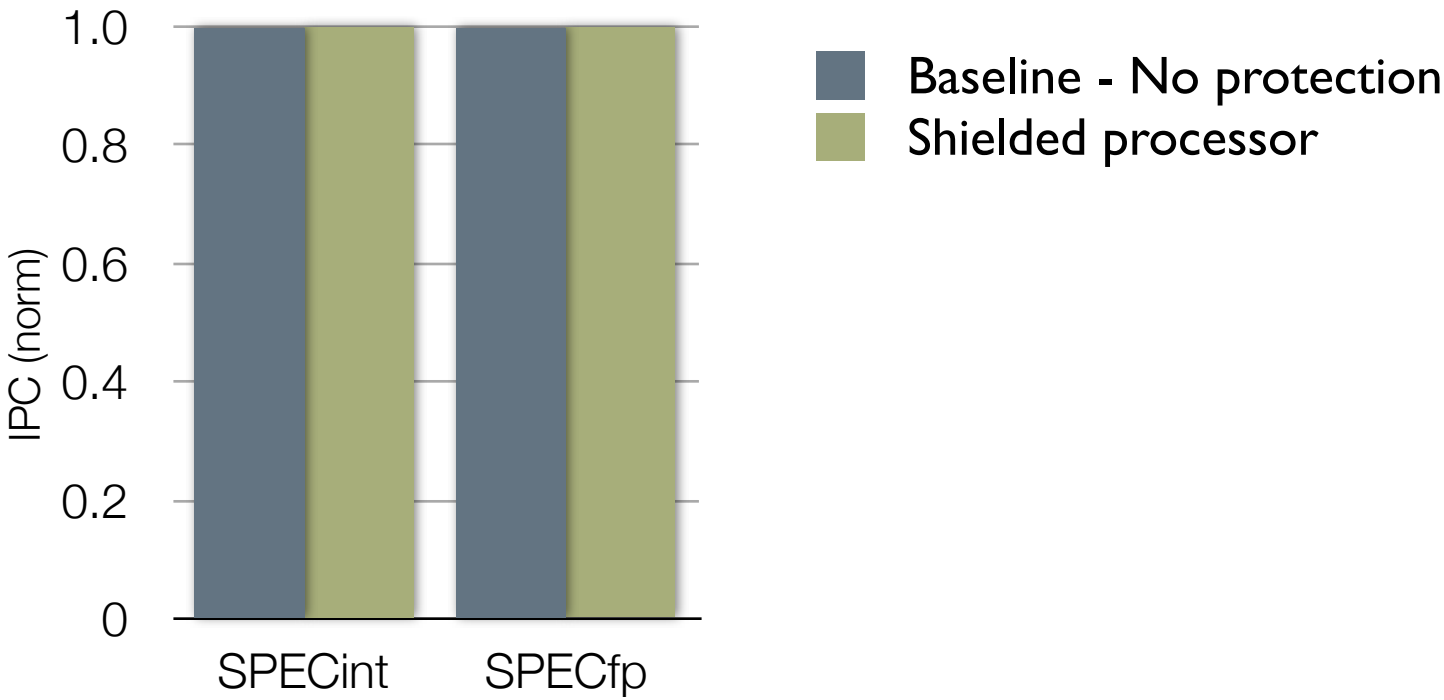
- Why we are not perfect:
  - Prediction accuracy
  - Certain program phases have many useful registers simultaneously

Power Usage



- Savings due to:
  - Fewer ECC generators and checkers
  - Fewer operations performed on them

Performance impact



- The ROB provides enough slack: no performance loss

Conclusions

- Shield increases the resistance of register files to soft errors
  - No need to protect the entire lifetime of the register versions
  - Performance is not affected
  - Modest power and area consumption
- Shield reduces the SDC AVF of the register file
  - Integer: Up to 84% (73%on average)
  - FP: Up to 100% (85% on average)

Using Register Lifetime Predictions to Protect Register Files Against Soft Errors

Pablo Montesinos, Wei Liu* and Josep Torrellas

University of Illinois at Urbana-Champaign
*Intel

i-acoma group