Speculative Early Register Release

J. Alastruey[+], T. Monreal[+], V. Viñals[+], M. Valero[*]

[+]gaZ - Universidad de Zaragoza

[*] Universidad Politécnica de Cataluña

HiPEAC - High-Performance Embedded Architecture and Compilation

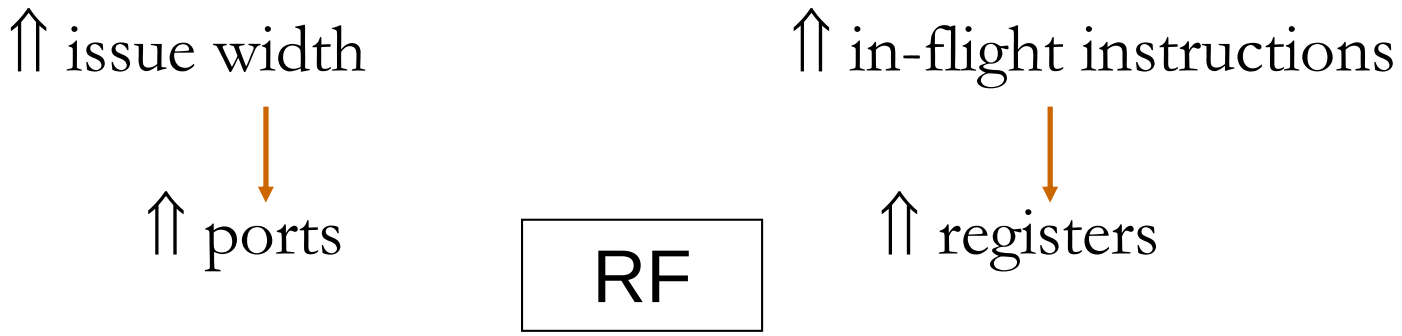
ACM International Conference on

Computing Frontiers, Ischia 2006

gaZ

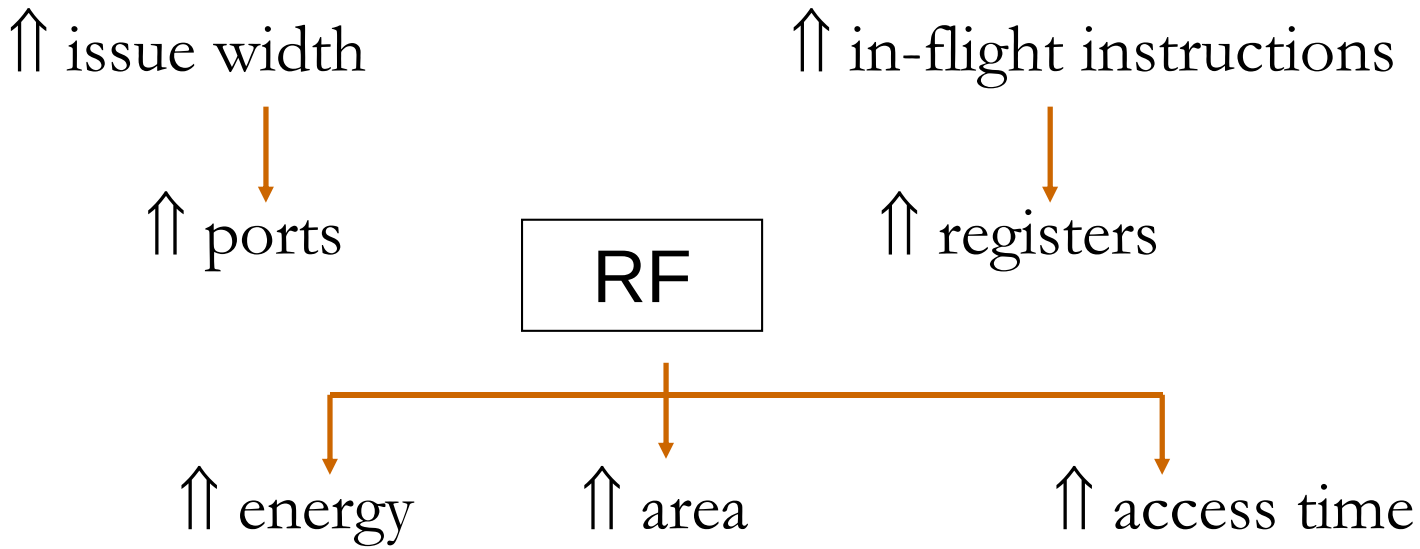
Overview

- Context
  - Register file of out-of-order superscalar processors
- Trends



Overview

- Context
  - Register file of out-of-order superscalar processors
- Trends



Register File is becoming more critical

Related work

- Register file organization
  - Two-level RF[1], distributed RF[2], RF banked[3], RF cache[4]
- Register allocation and release
  - Late allocation[5]
  - Early release[6,7,8]

gaZ

[1]R. Balasubramonian et al, “Reducing the Complexity of the Register File ...”, MICRO 01.
[2]E. Borch et al., “Loose Loops Sink Chips”, HPCA 02.
[3]J.L. Cruz et al., “Multiple-Banked Register File Architectures”, ISCA 00.
[4]J.A. Butts et al., “Use-Based Register Caching with Decoupled Indexing”, ISCA 04.
[5]T. Monreal et al., “Delaying Physical Register allocation …”, MICRO 99.
[6]M. Moudgill et al., “Register Renaming and Dynamic Speculation: an Alternative Approach”, MICRO 93.
[7]T. Monreal et al., “Hardware Schemes for Early Register Release”, ICPP 02.
[8]O. Ergin et al., “Increasing Processor Performance Through Early Release”, ICCD 04.

Motivation

- Conventional release is inefficient
  - Many physical registers will NOT be read in the future
    - $\Uparrow$ register file pressure $\Rightarrow$ $\Uparrow$ rename stalls

- Early release of dead physical registers
  - Register reuse $\Rightarrow$ $\Uparrow$ **IPC**
  - Reduce RF size
    - $\Downarrow$ energy (linear)
    - $\Downarrow$ area (linear)
    - $\Downarrow$ Taccess

      if RF in critical path, $\Downarrow$ Tcycle $\Rightarrow$ $\Uparrow$ **IPS**

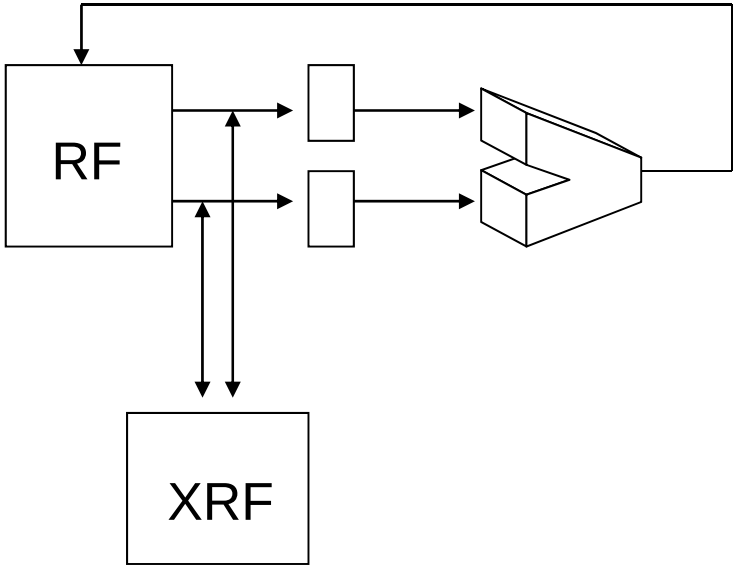
gaZ

Contributions

- Modify conventional release policy
  - Shift release from NV to LU
  - LU identification: Last-Use Predictor
  - Early released values sent to auXiliary Register File

```
V:    r1= ..
      ...
LU: .. = r1
      ...
NV: r1 = ..
```

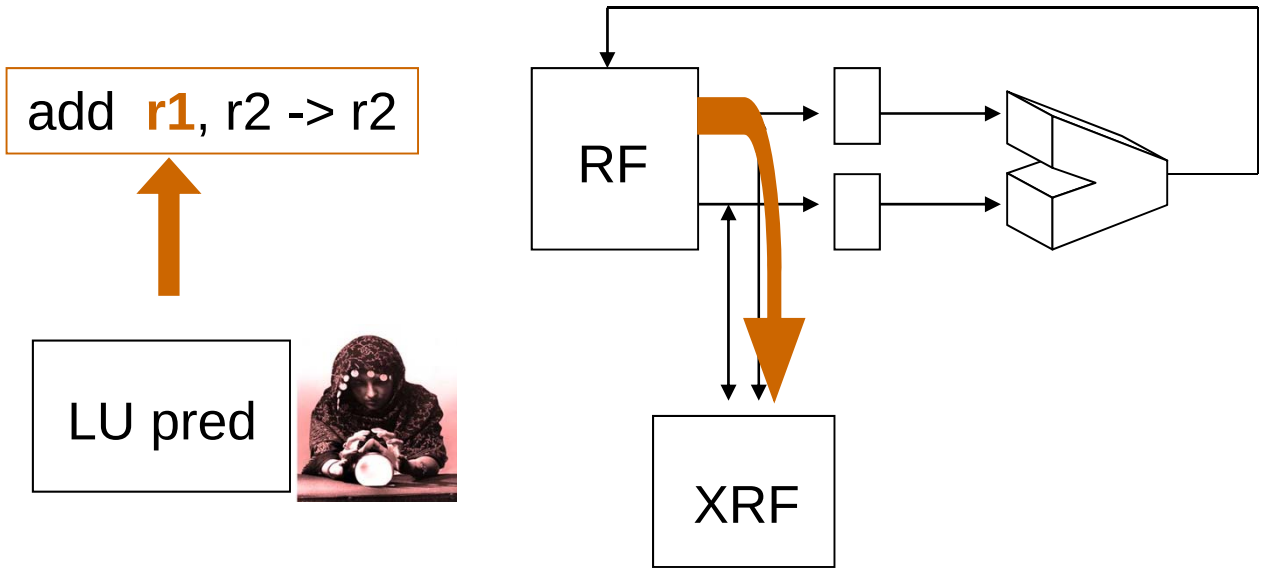
add r1, r2 -> r2



Contributions

- Modify conventional release policy
  - Shift release from NV to LU
  - LU identification: Last-Use Predictor
  - Early released values sent to auXiliary Register File

```
V:    r1= ..
      ...
LU: .. = r1
      ...
NV: r1 = ..
```

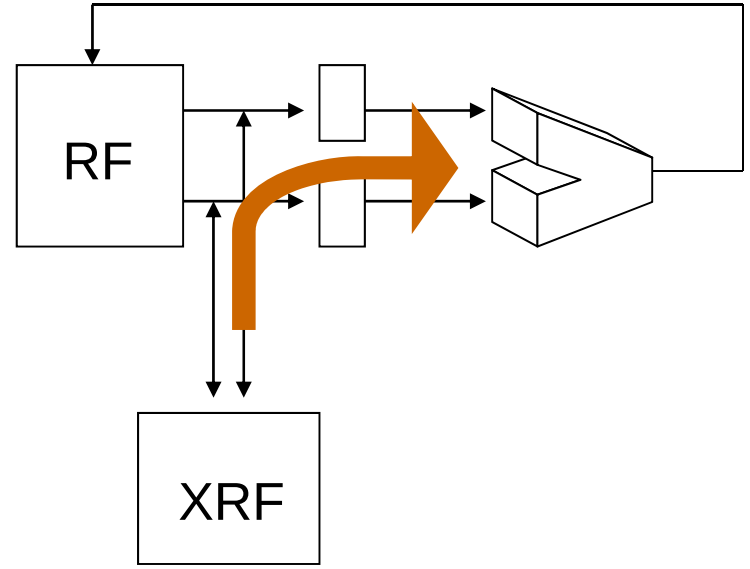


Contributions

- Modify conventional release policy
  - Shift release from NV to LU
  - LU identification: Last-Use Predictor
  - Early released values sent to auXiliary Register File

V: r1= ..
...
LU: .. = r1
...
NV: r1 = ..

add r1, r2 -> r2



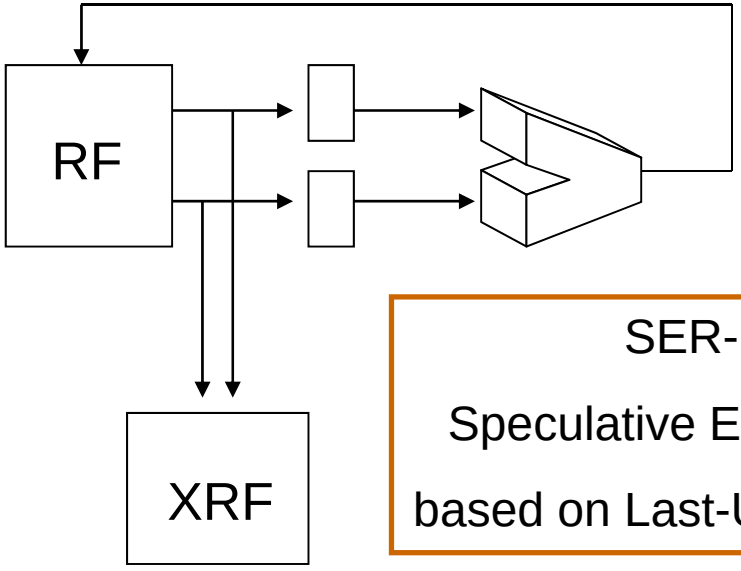
LU pred

Contributions

- Modify conventional release policy
  - Shift from NV to LU
  - LU identification: Last-Use Predictor
  - Early released values sent to auXiliary Register File

```
V:    r1= ..
      ...
LU: .. = r1
      ...
NV: r1 = ..
```

add **r1**, r2 -> r2



LU pred

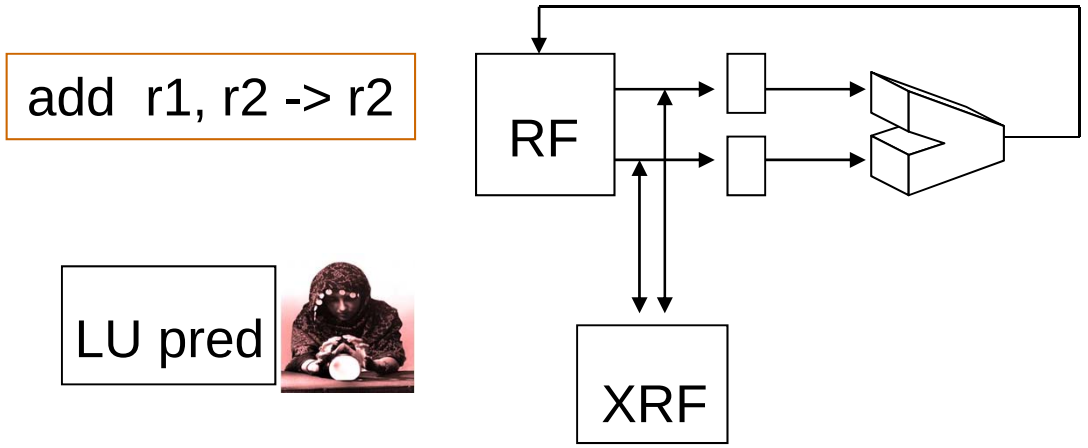
SER-LUP

Speculative Early Release

based on Last-Use Prediction

Contributions

- Analyze viability of mechanism
  - Last-Use Predictor
  - Auxiliary Register File (XRF)
- Analyze potential of SER-LUP policy
  - Abstract from concrete implementations
  - Oracle Last-Use predictor (SER-OB)



Outline

- **Conventional release policy vs SER-LUP**
- Basic structures: Last-Use predictor and XRF
- Results
- Related work
- Conclussions

gaZ

Conventional release policy

V: r2 = ..

...

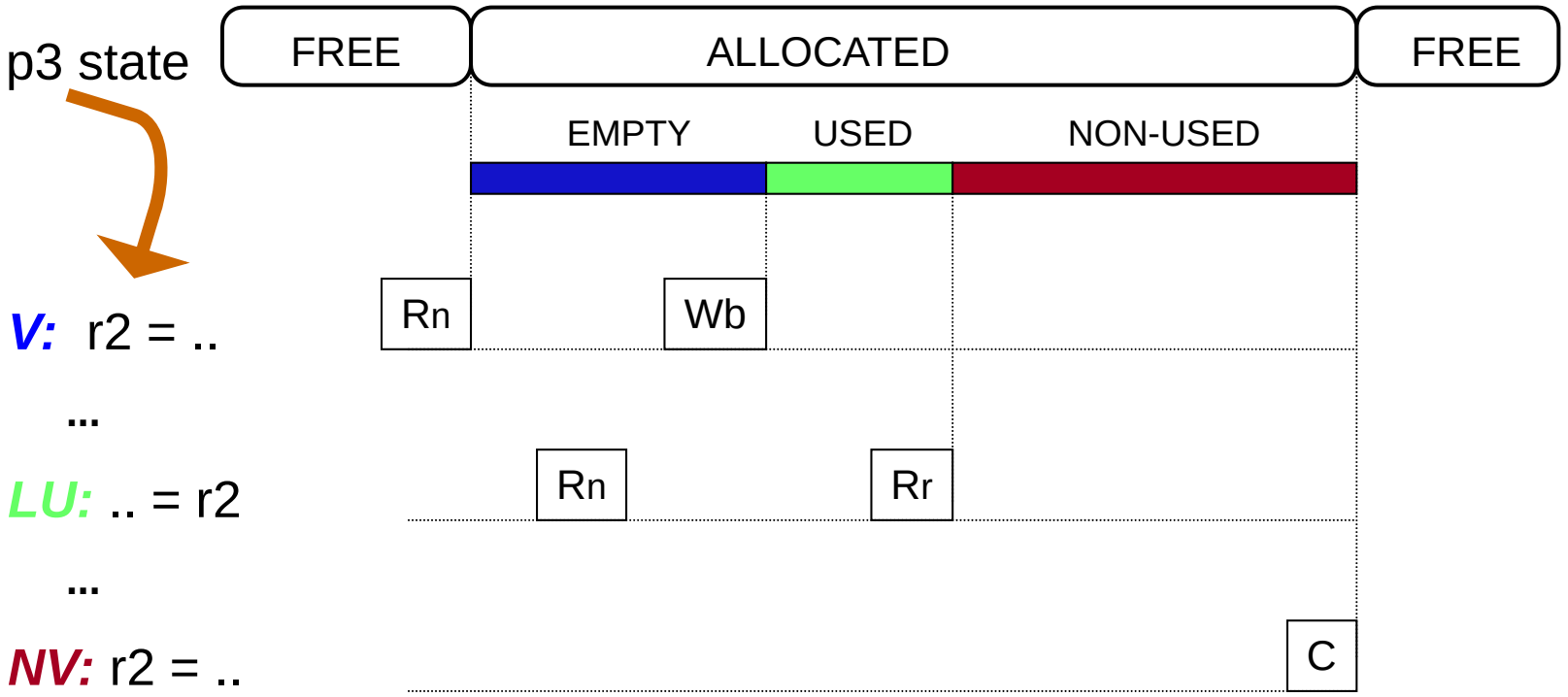
LU: .. = r2

...

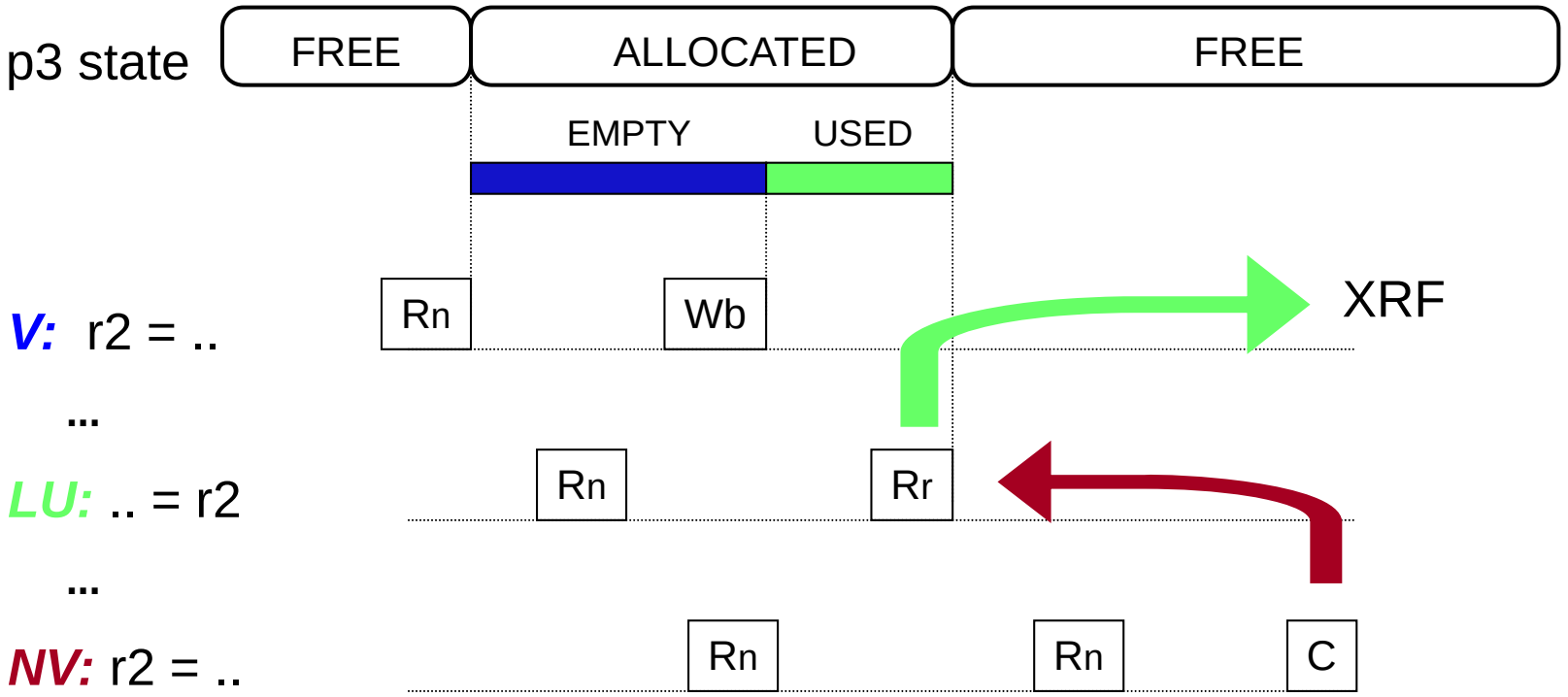
NV: r2 = ..

gaZ

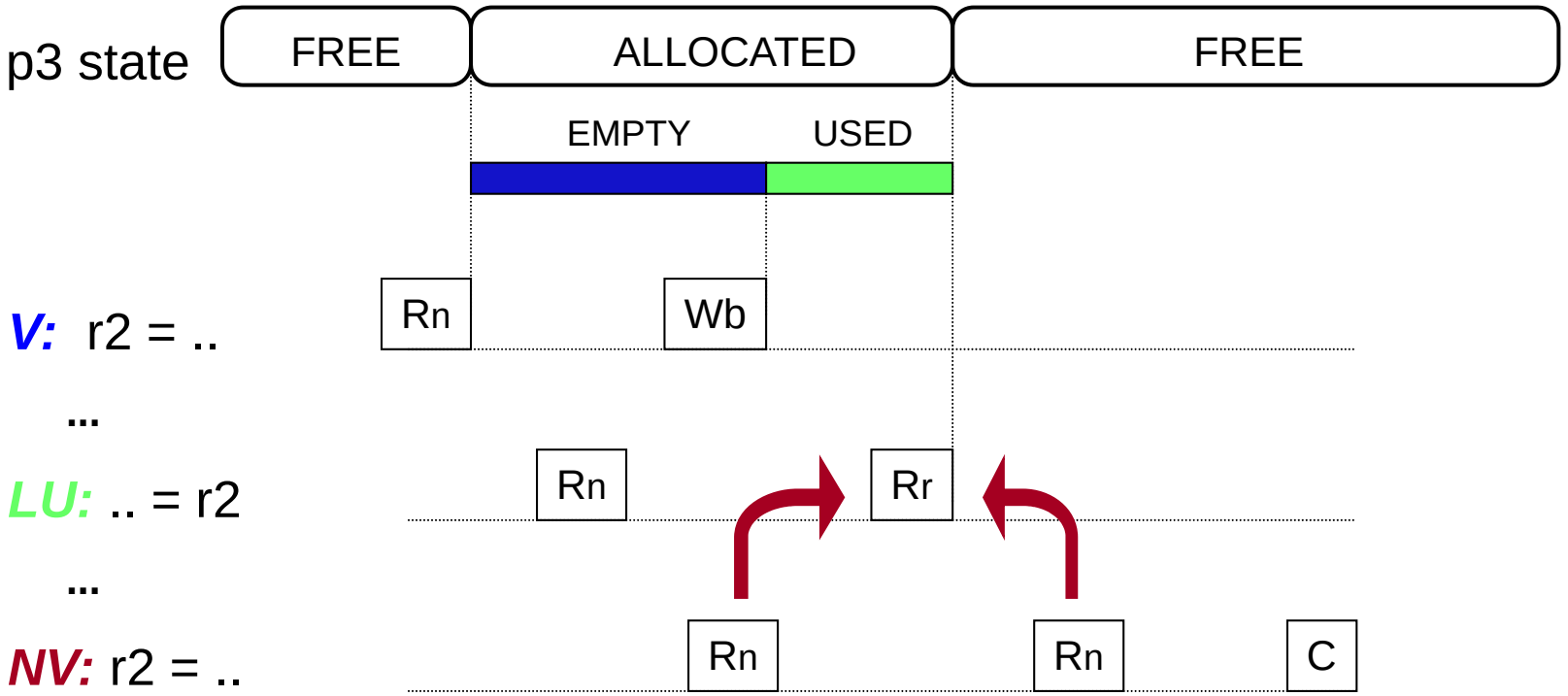
Conventional release policy



SER-LUP policy



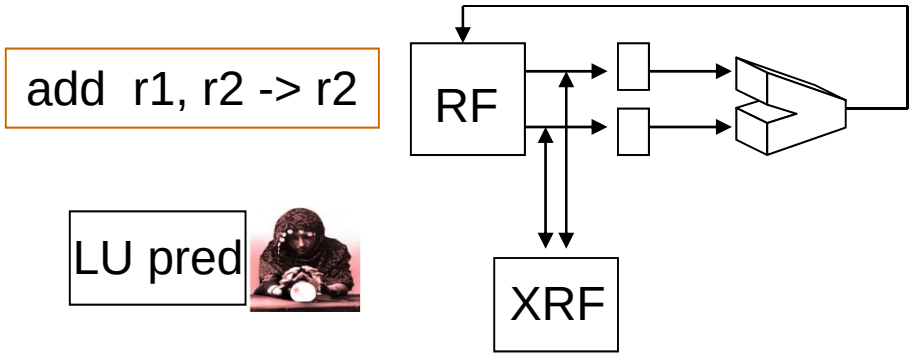
SER-LUP policy



No need to wait for NV

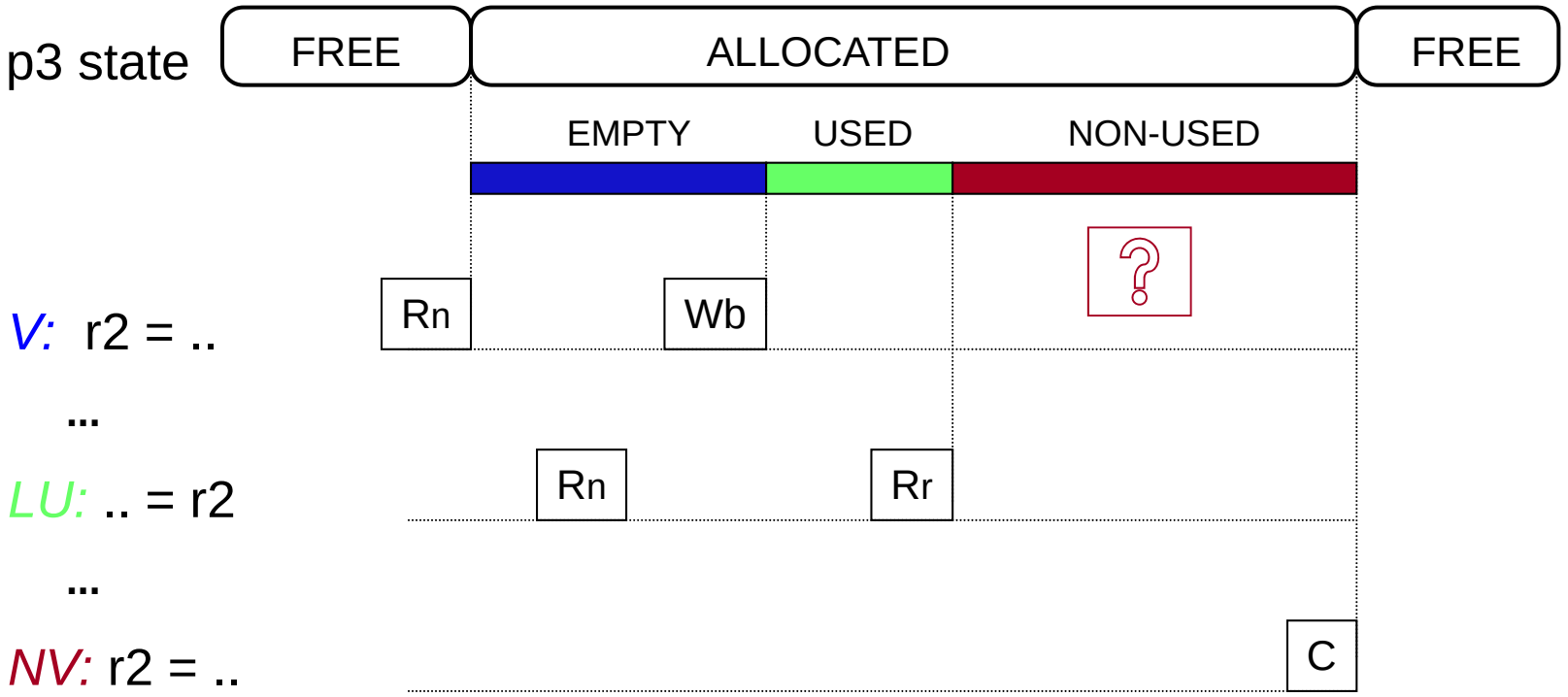
SER-LUP policy

- Replace conventional releasing policy
- Speculative Early Release



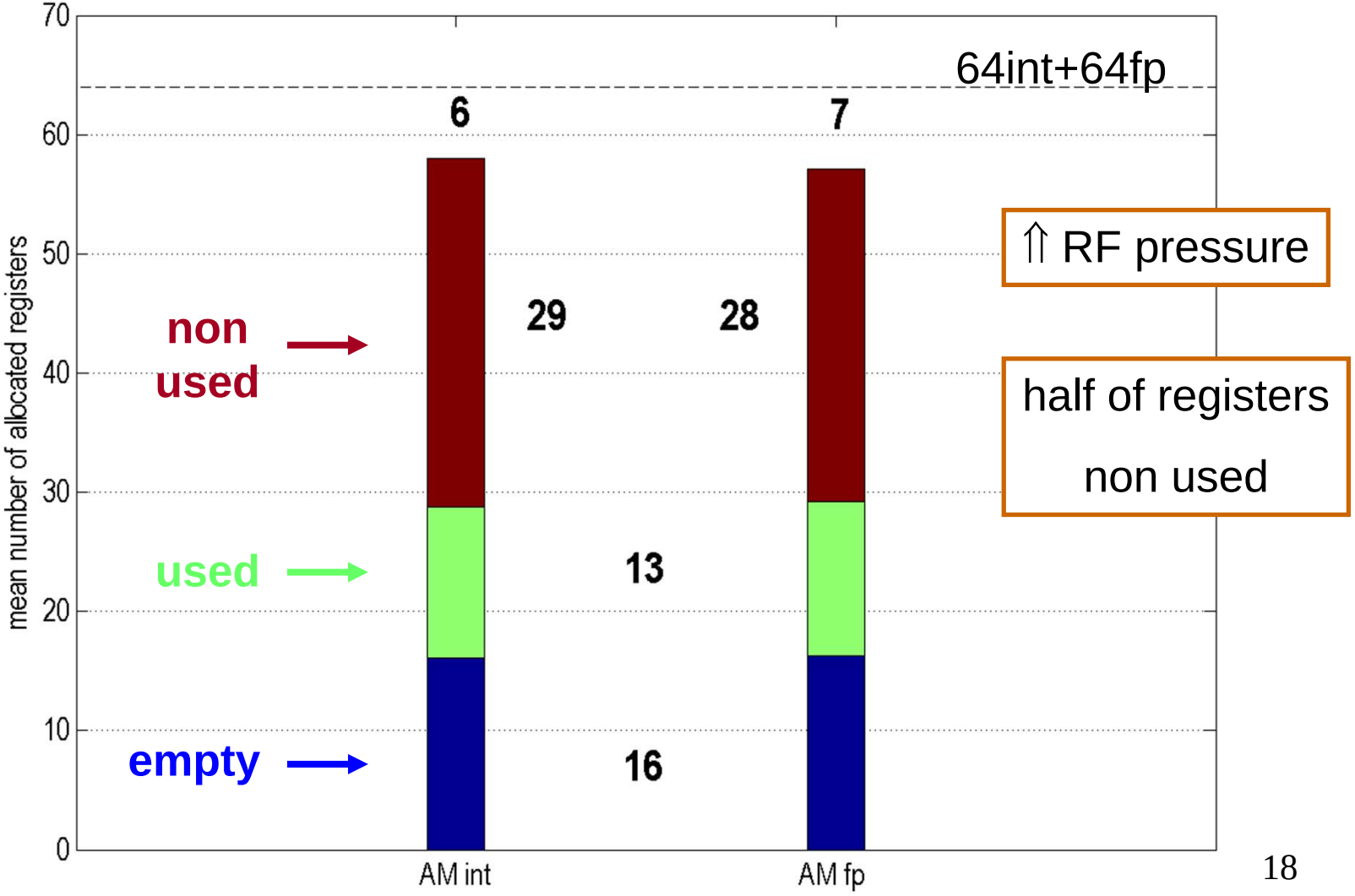
- Last-Use Predictor
  - Identify Last-Use instructions
- XRF
  - Stored early released registers
  - Supply prematurely released registers at a higher latency

Conventional release policy



Conventional release policy

- 256-ROB, 8-way (spec2000)



Outline

- Conventional release policy vs SER-LUP
- **Basic structures**
  - **Last-Use Predictor**
  - **XRF**
- Results
- Related work
- Conclussions

gaZ

Patterns of use

- Last-use information of instruction registers

```
        addq  r1, r2 -> r3
        stl   r1-> 0(r3)
        ldl   0(r3) -> r3
        bne   r2, A
        ...
A:      ldl   8(r1) -> r3
        addq  r2, r2 -> r1
```

last-use		
s1	**s2**	**d**
n	n	n
n	n	-
y	-	y
n	-	-
y	-	n

gaZ

Patterns of use

- Last-use information of instruction registers

```
        addq  r1, r2 -> r3
        stl   r1-> 0(r3)
        ldl   0(r3) -> r3
        bne   r2, A
        ...
A:      ldl   8(r1) -> r3
        addq  r2, r2 -> r1
```

last-use		
s1	**s2**	**d**
n	n	n
n	n	-
y	-	y
n	-	-
y	-	n

pattern of use

0s+0d

Patterns of use

- Last-use information of instruction registers

```
        addq  r1, r2 -> r3
        stl   r1-> 0(r3)
        ldl   0(r3) -> r3
        bne   r2, A
        ...
A:      ldl   8(r1) -> r3
        addq  r2, r2 -> r1
```

last-use		
s1	**s2**	**d**
n	n	n
n	n	-
y	-	y
n	-	-
y	-	n

pattern of use → 1s+1d

Patterns of use

- Last-use information of instruction registers

```
        addq  r1, r2 -> r3
        stl   r1-> 0(r3)
        ldl   0(r3) -> r3
        bne   r2, A
        ...
A:      ldl   8(r1) -> r3
        addq  r2, r2 -> r1
```

last-use		
s1	**s2**	**d**
n	n	n
n	n	-
y	-	y
n	-	-
y	-	n

2s+1d, 2s+0d

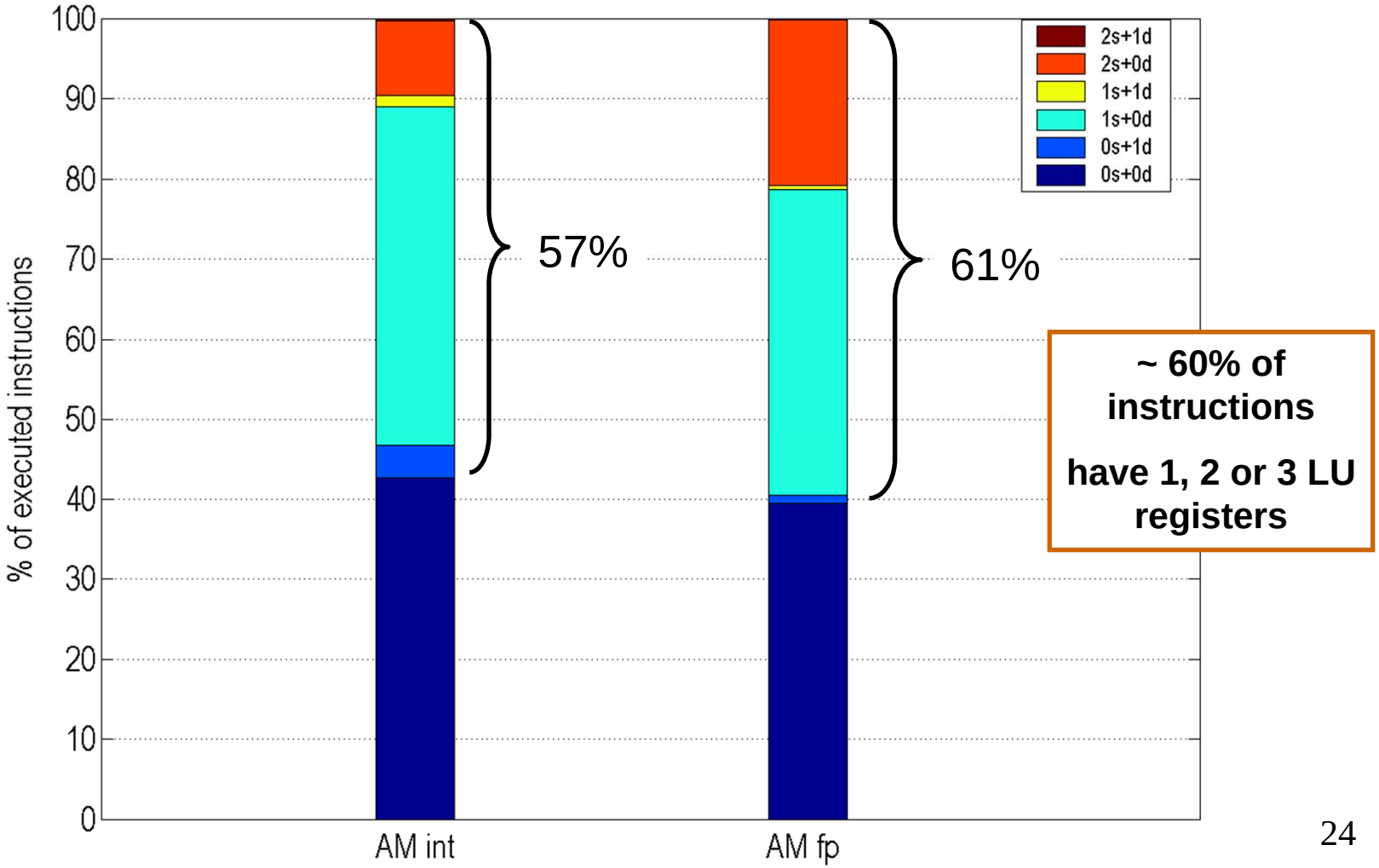
1s+1d, 1s+0d,

0s+1d, 0s+0d

gaZ

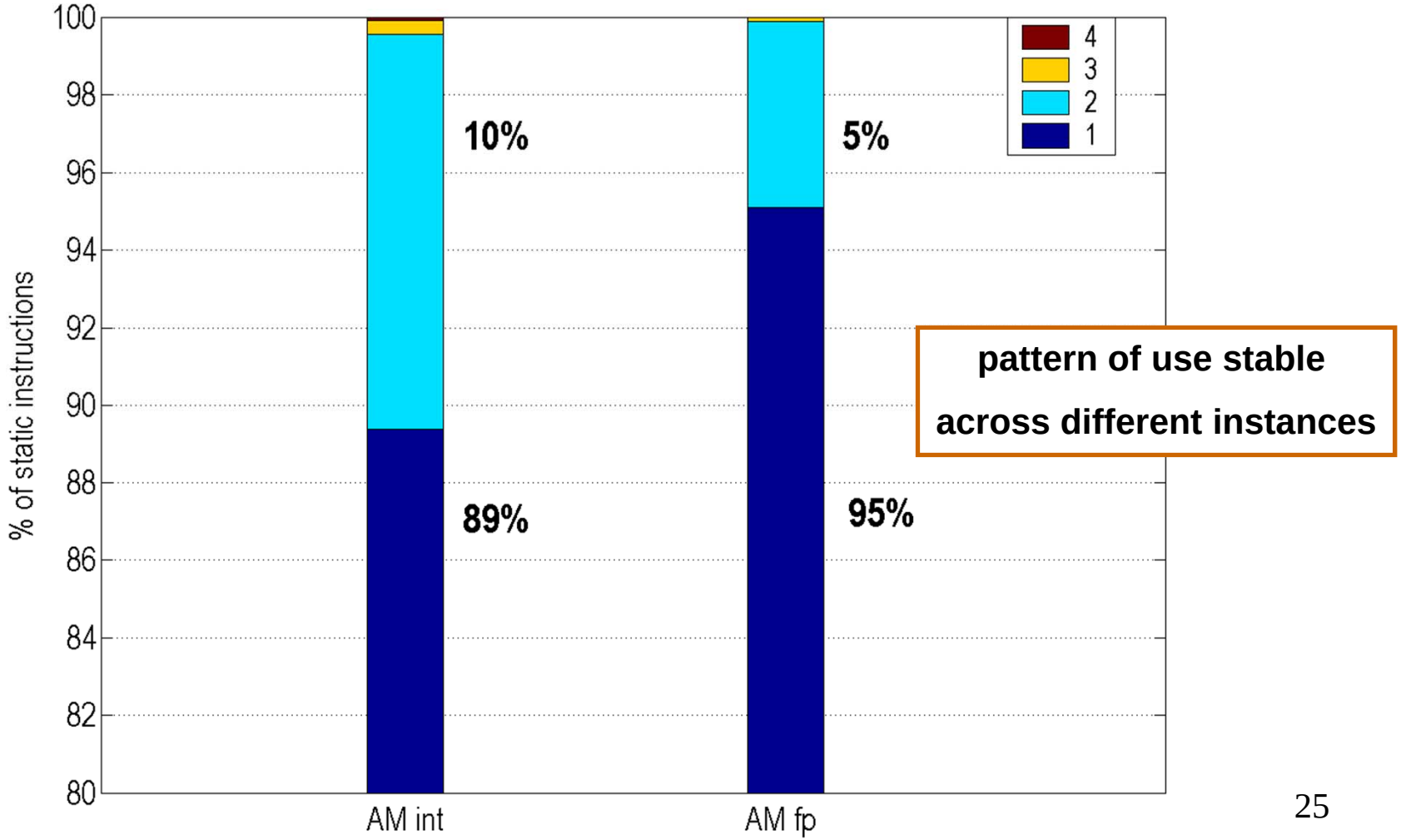
Characterization of patterns of use

- Dynamic frequency (spec2000)



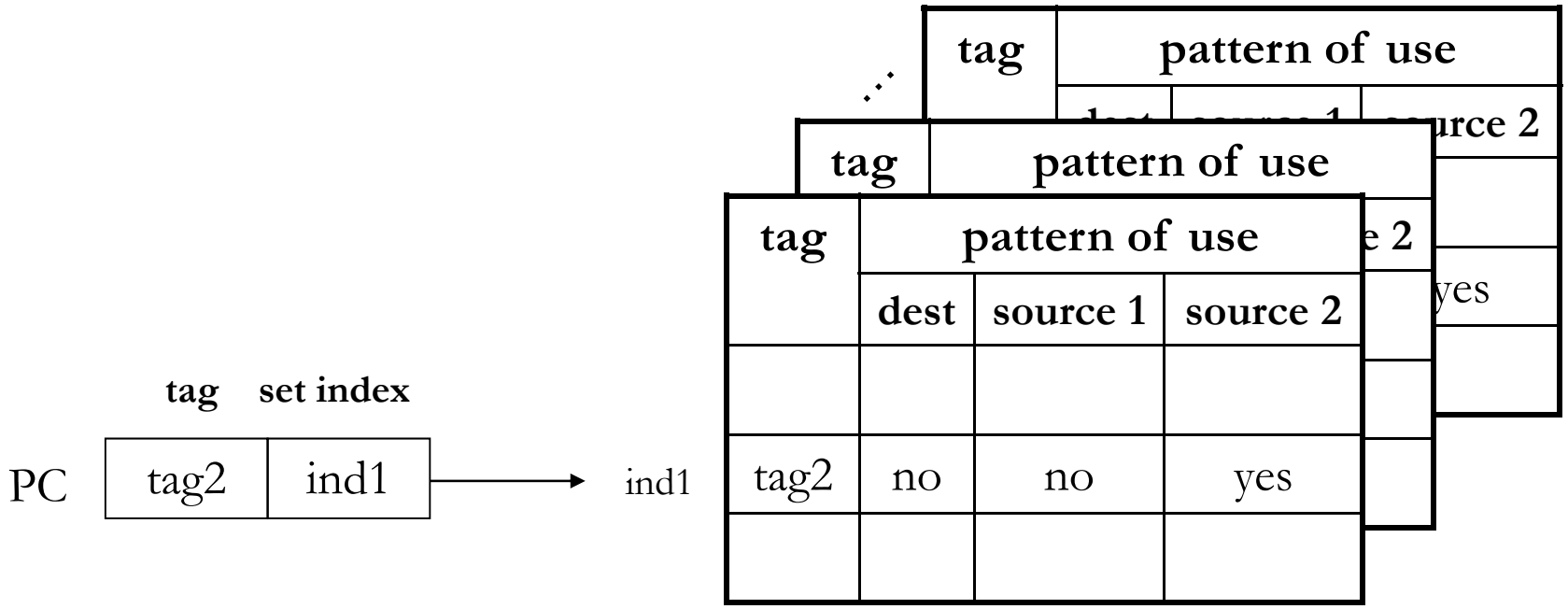
Characterization of patterns of use

- Stability (spec2000)



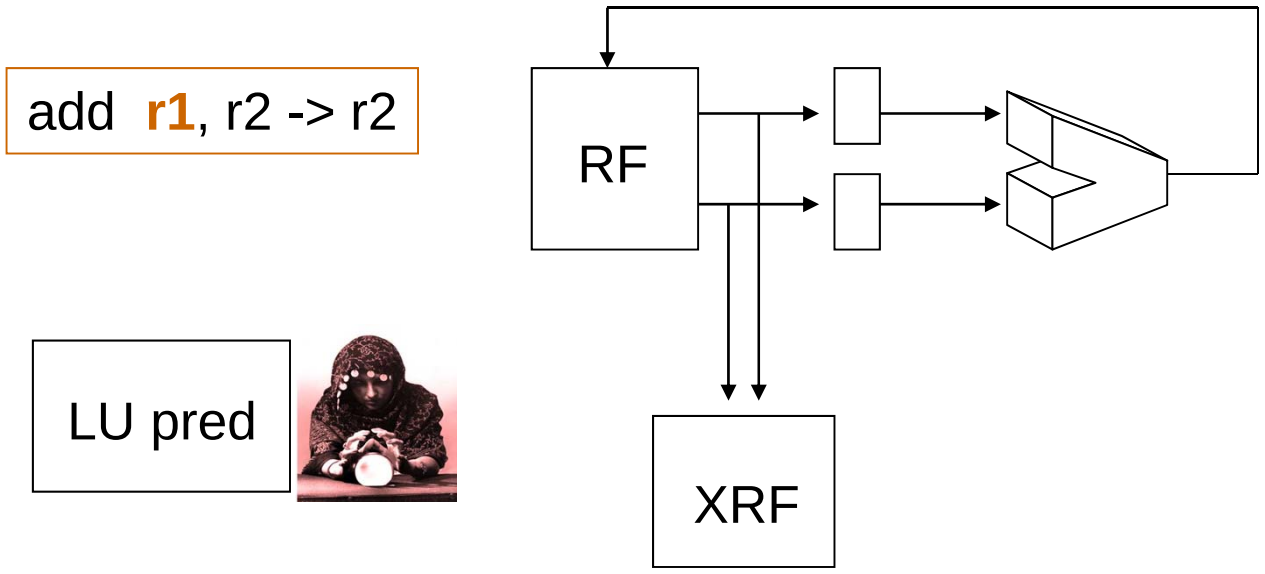
Last-Use Predictor

- Organization of a **simple** last-use predictor
  - **Sticky:** records "is a LU register" information of commited instructions



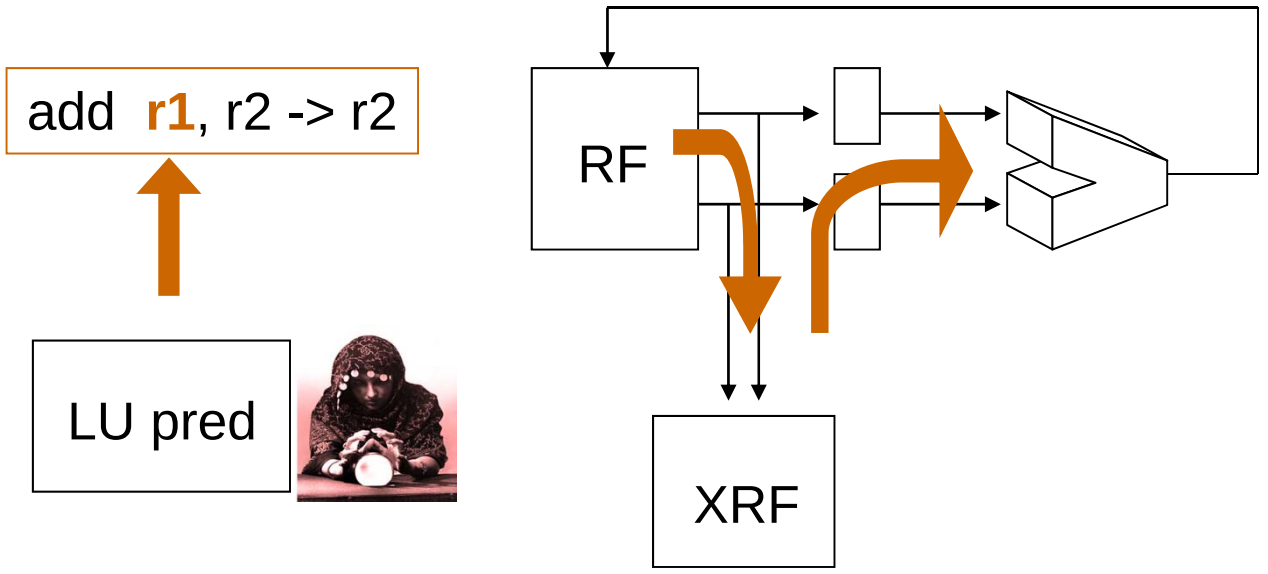
Last-Use Predictor

- **Last-use predictor** mispredictions
  - Wrong "LU register" predictions: premature release
  - Wrong "not LU register" predictions: lost opportunity



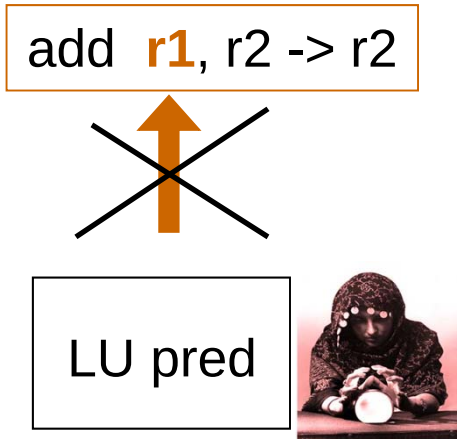
Last-Use Predictor

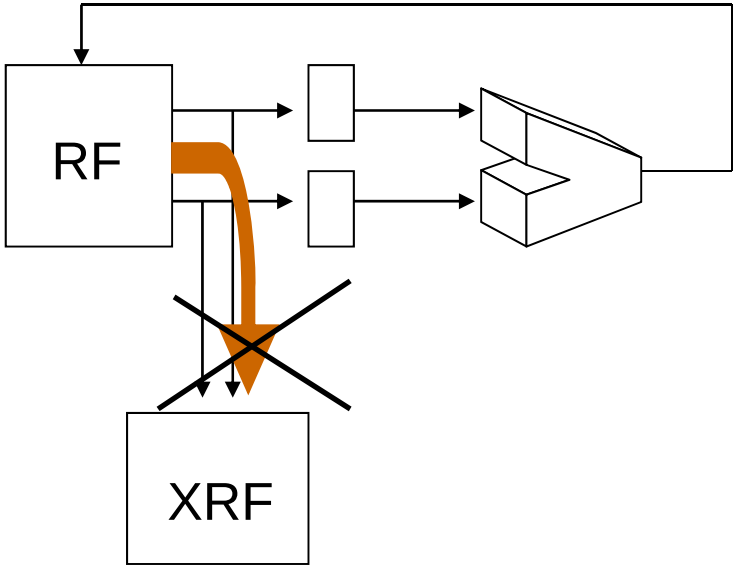
- **Last-use predictor** mispredictions
  - Wrong "LU register" predictions: premature release
  - Wrong "not LU register" predictions: lost opportunity



Last-Use Predictor

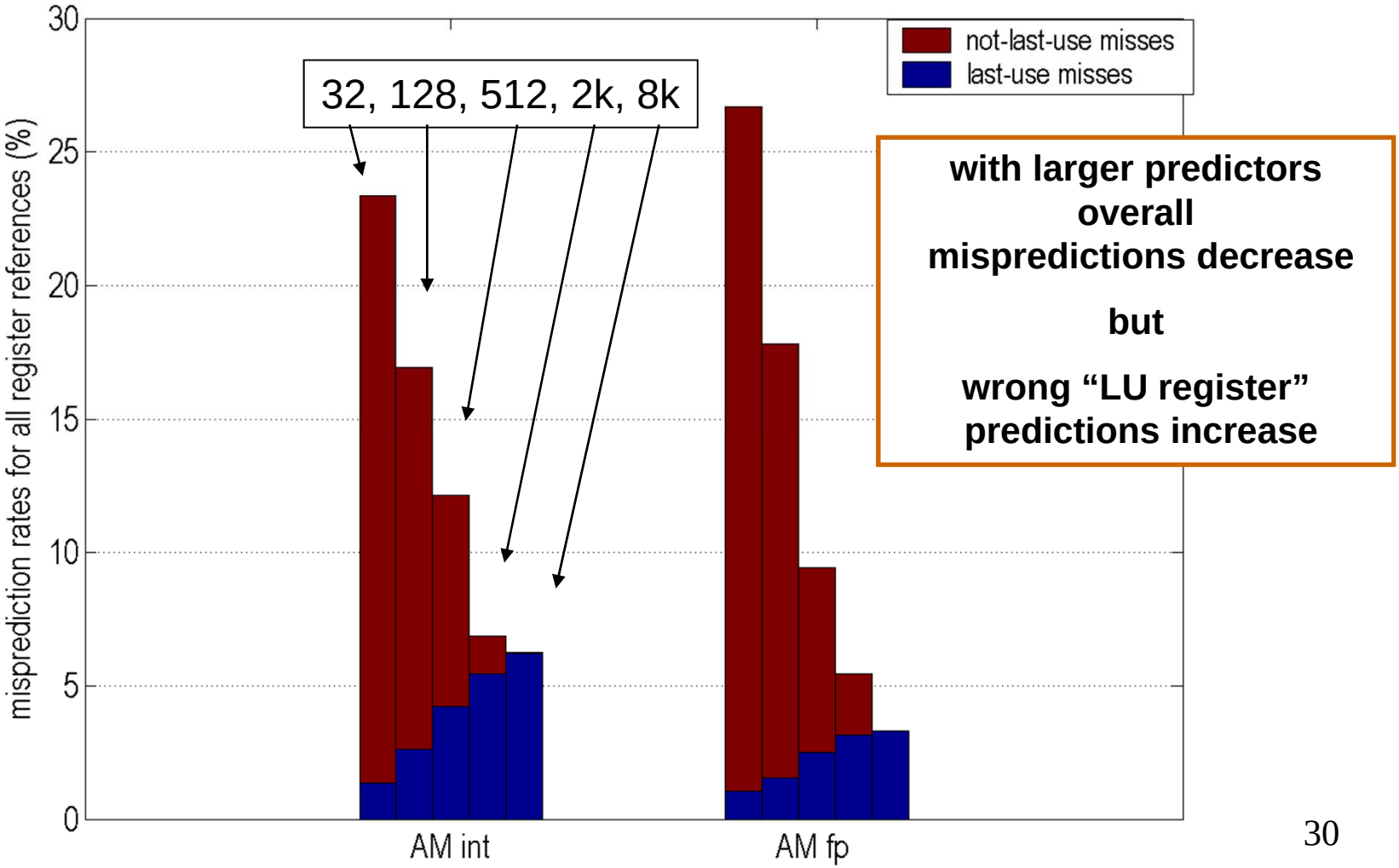
- **Last-use predictor** mispredictions
  - Wrong "LU register" predictions: premature release
  - Wrong "not LU register" predictions: lost opportunity





Last-Use Predictor

- Misprediction rate for register references (spec2000)



Outline

- Conventional release policy vs SER-LUP
- **Basic structures**
  - **Last-Use Predictor**
  - **XRF**
- Results
- Related work
- Conclussions

gaZ

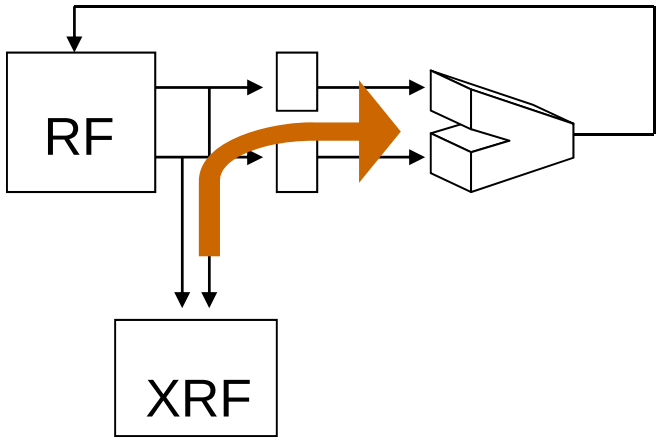
XRF

- Stores early released values
- Supplies unexpected uses (UUs)
  - wrong "last-use registers" predictions
  - out-of-order execution
  - mispredicted branches

```
V:          r1=
...
predLU:     = r1
...
UU:         = r1
```

add **r1**, r2 -> r2

LU pred



XRF

- Cases of unexpected uses (UU)
  - wrong "last-use registers" prediction
  - out-of-order execution UU (oooUU)
  - mispredicted path UU (mpUU)

oooUU

```
V:    r1=..
      ...
U:    .. = r1
      ...
LU:   .. = r1
```

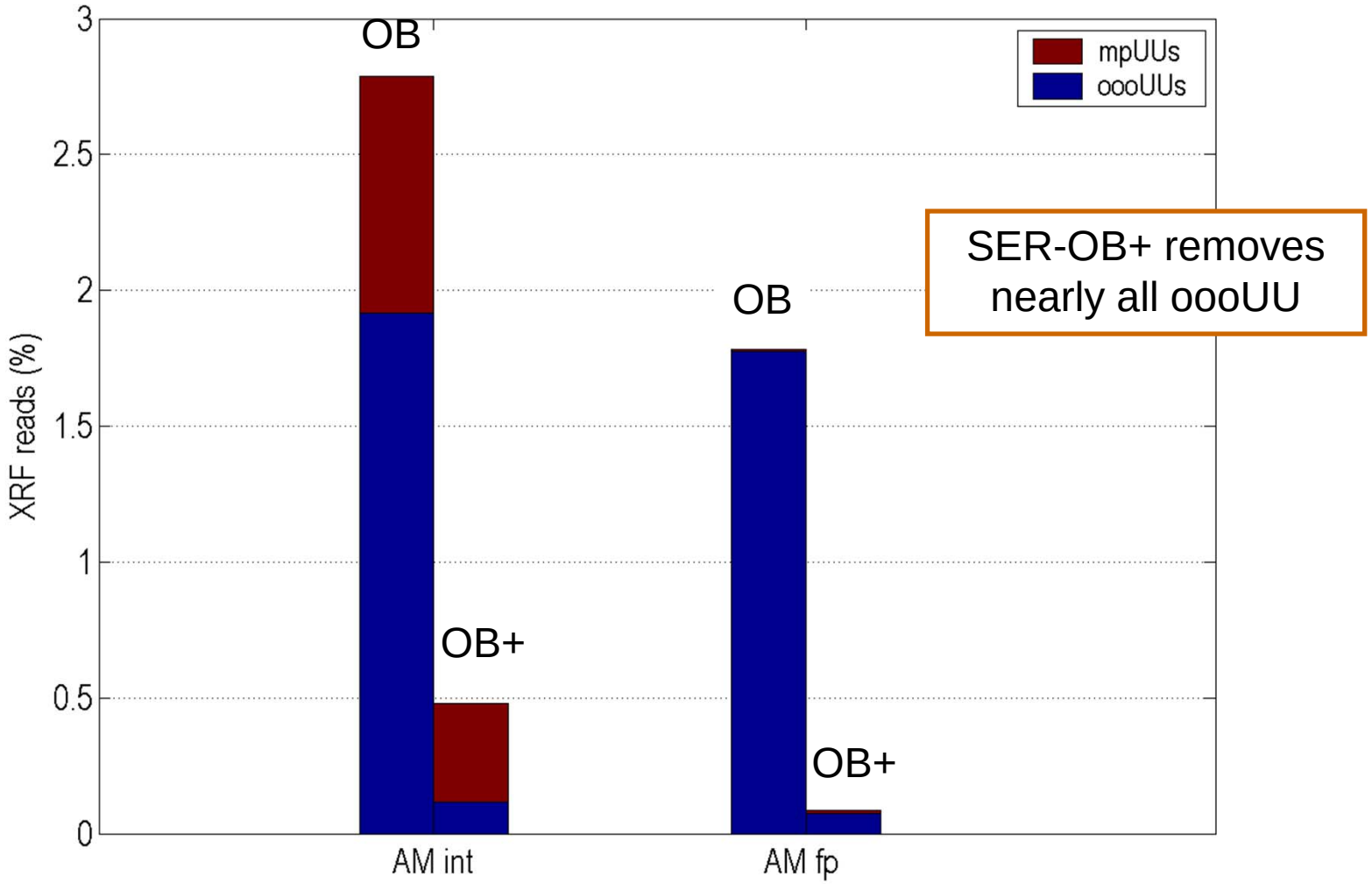
If LU executes before U

U becomes oooUU

gaZ

XRF reads

- SER-OB vs SER-OB+ (spec2000)



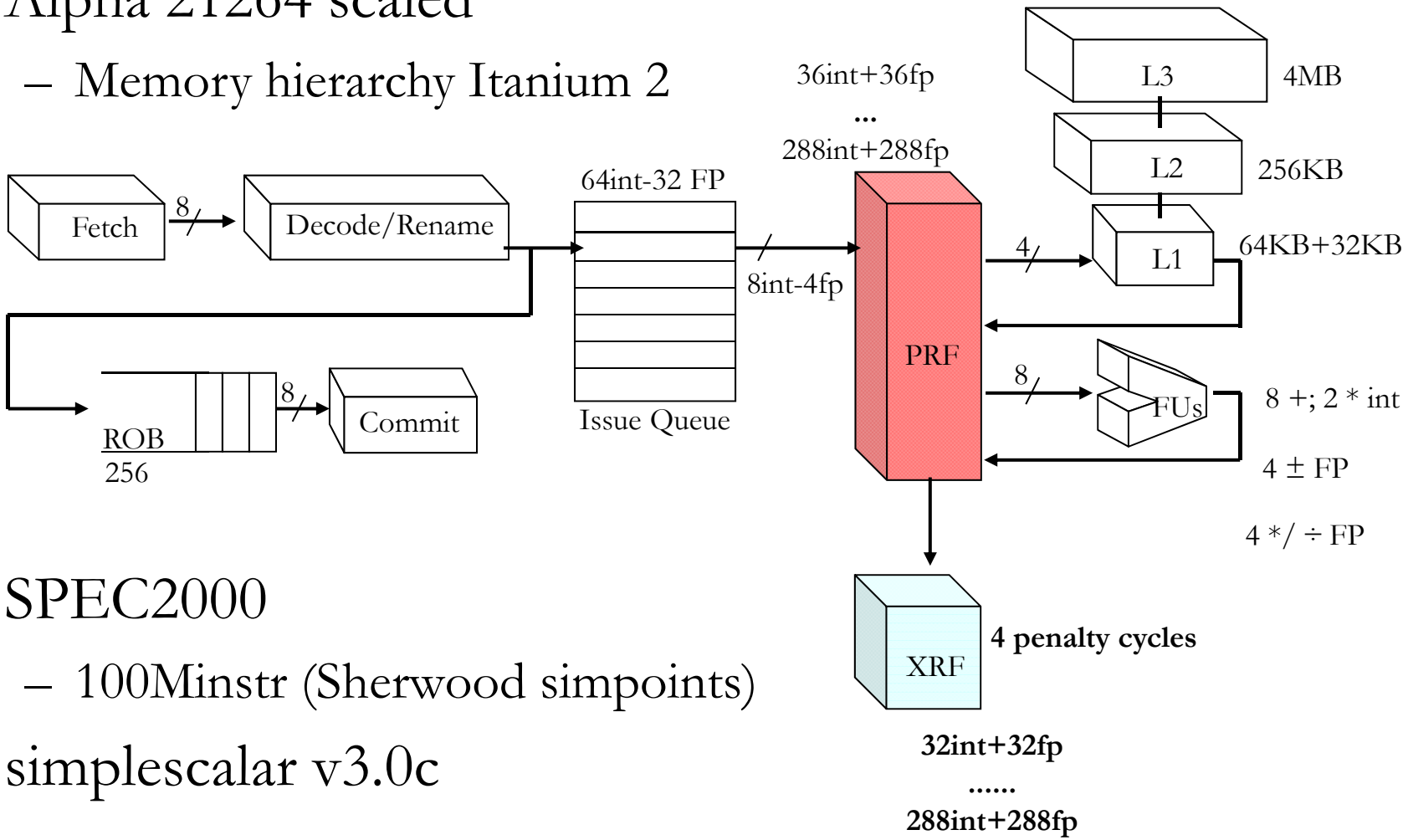
Ouline

- Conventional release policy vs SER-LUP
- Basic structures: Last-Use predictor and XRF
- **Results**
- Related work
- Conclussions

gaZ

Simulated processor

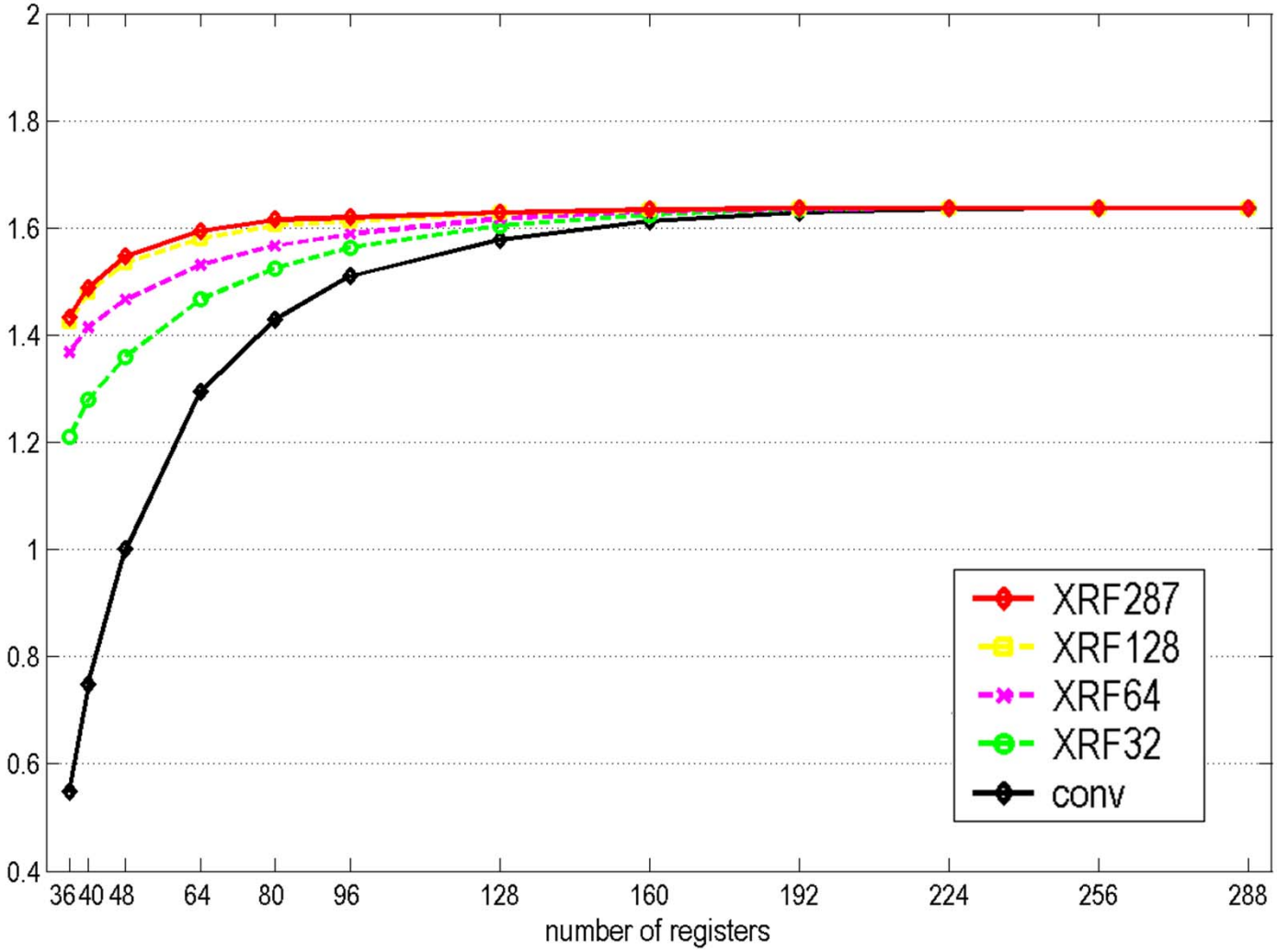
- Alpha 21264 scaled
  - Memory hierarchy Itanium 2



- SPEC2000
  - 100Minstr (Sherwood simpoints)
- simplescalar v3.0c

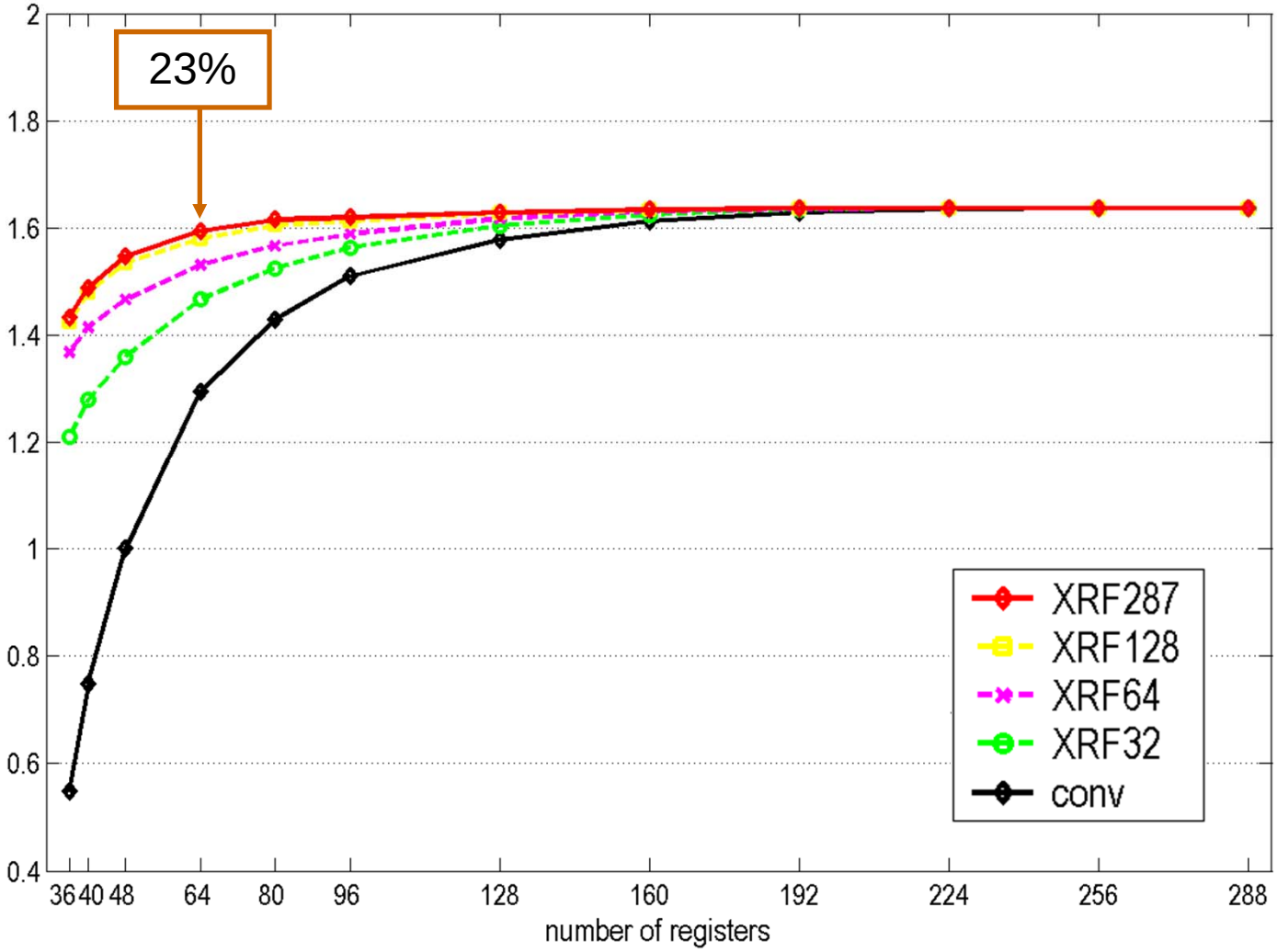
Results: SER-OB+

- IPC vs number of physical registers (specINT2000)



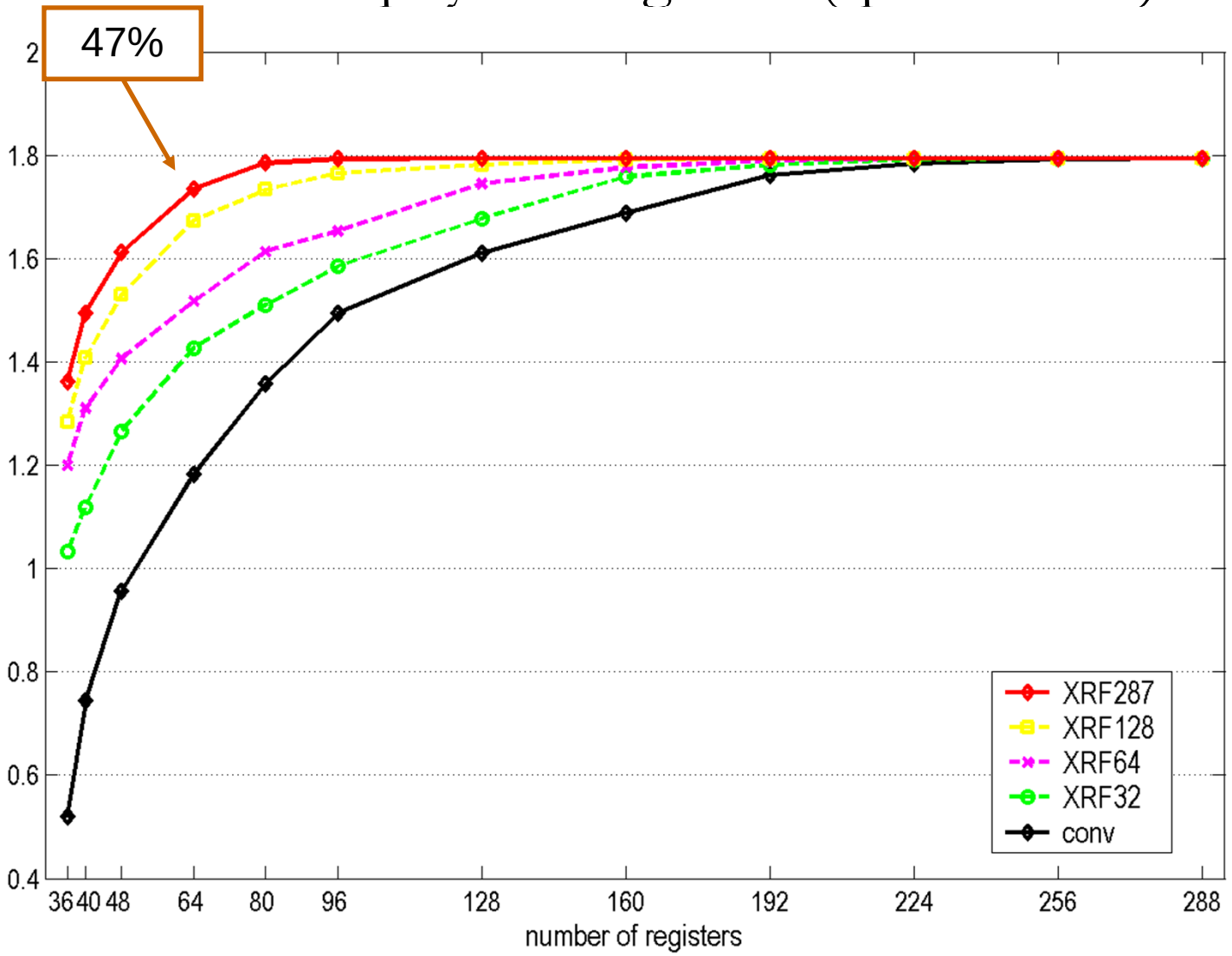
Results: SER-OB+

- IPC vs number of physical registers (specINT2000)



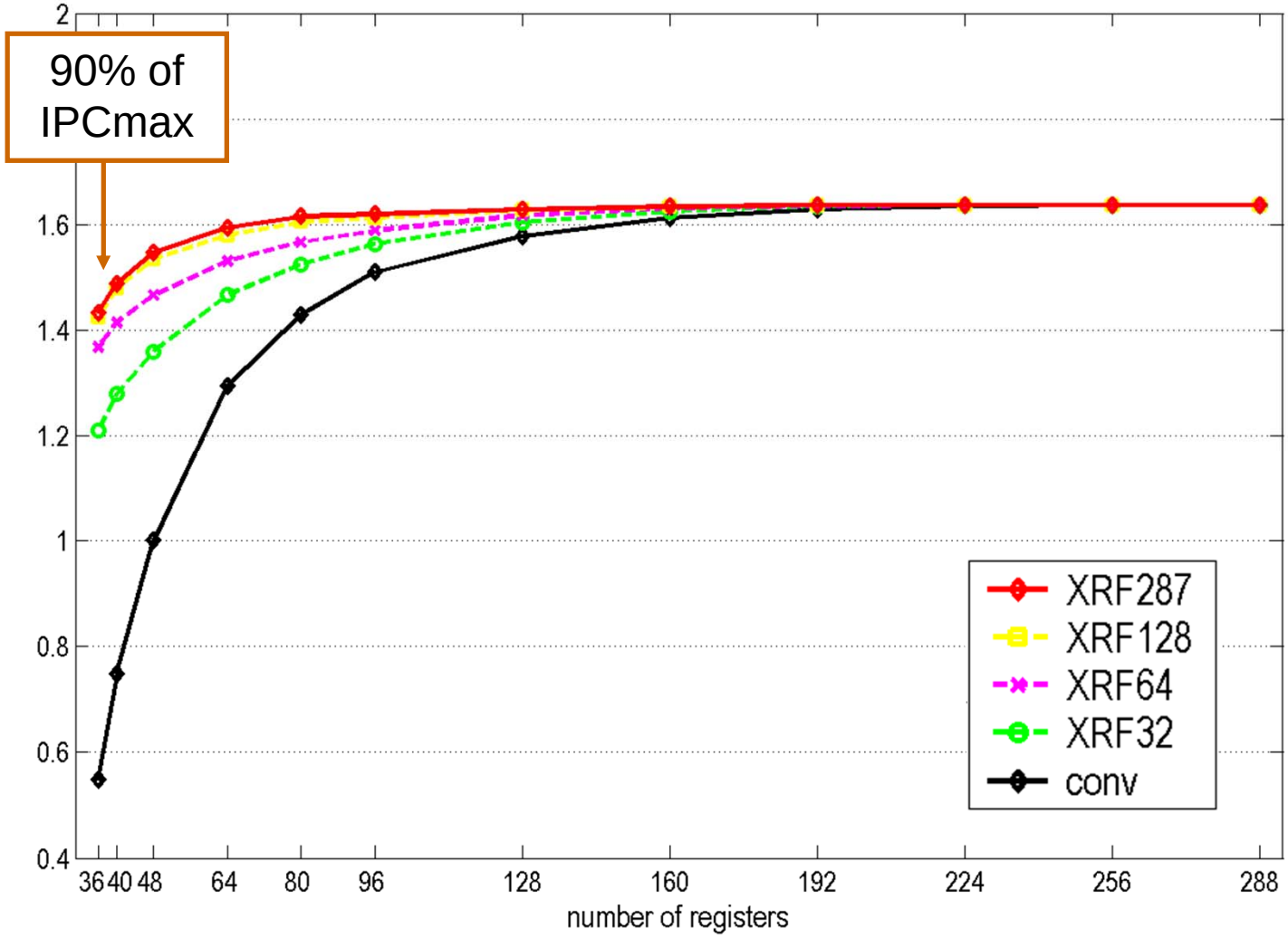
Results: SER-OB+

- IPC vs number of physical registers (specFP2000)



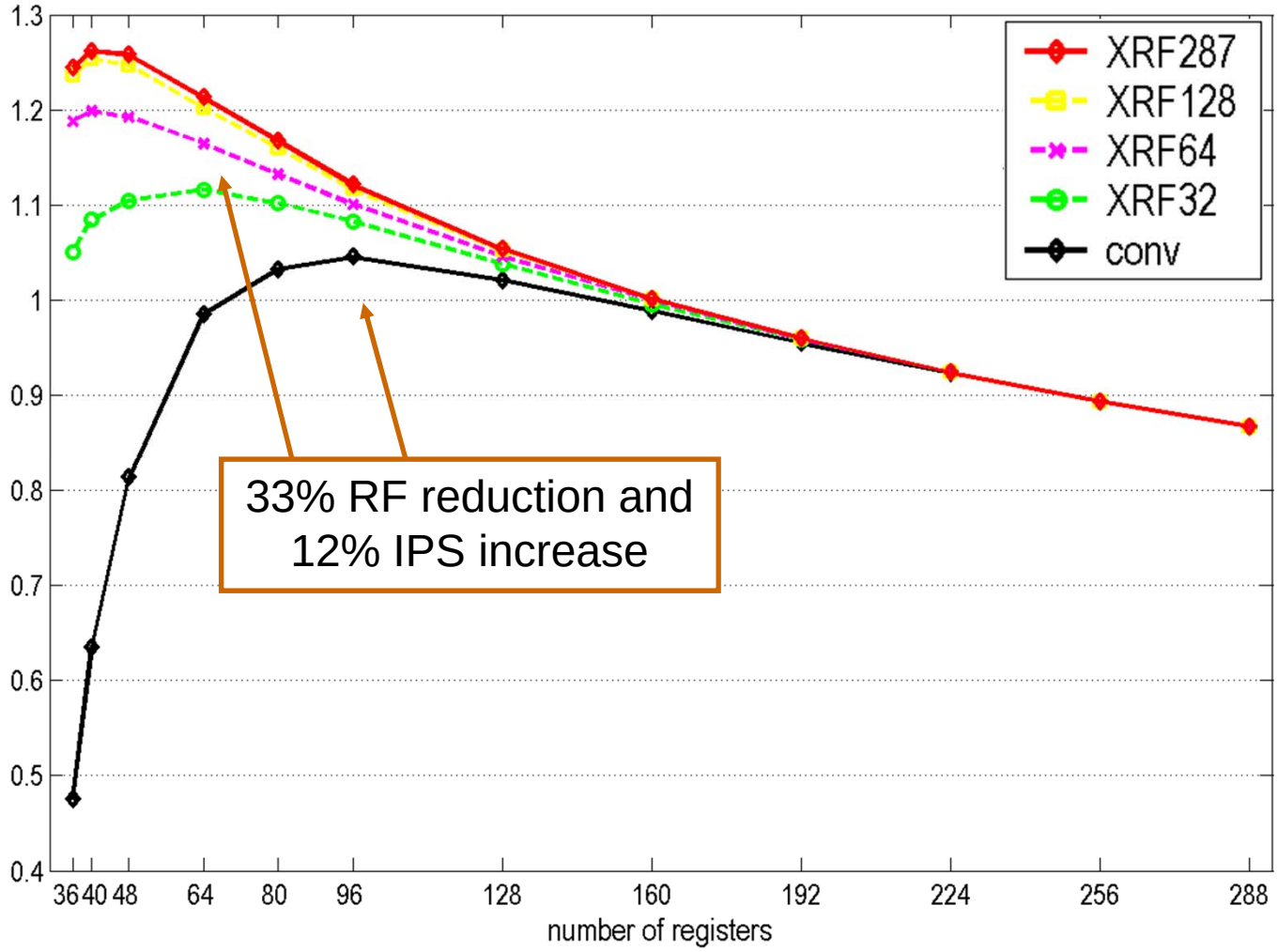
Results: SER-OB+

- IPC vs number of physical registers (specINT2000)



Results: SER-OB+

- BIPS conv. vs SER-OB+ (specINT2000, Rixner 0.18μ)



Outline

- Conventional release policy
- Idea: SER-LUP
- Basic structures: Last-Use predictor and XRF
- Results
- **Related work**
- Conclussions

gaZ

Related work

- Conservative early register release
  - Wait until NV is nonspeculative
  - Early released value will not be read anymore
    - Martínez et al. (MICRO-02), Monreal et al. (ICPP-02)

- Speculative early register release (SER)
  - Do NOT wait until NV is nonspeculative
  - Early released values may be read
    - Balasubramoniam et al. (MICRO-01), Ergin et al. (ICCD-04)

gaZ

LU identification does not need NV renaming

Supplies UU from XRF

Outline

- Conventional release policy
- Idea: SER-LUP
- Basic structures
- Results
- Related work
- **Conclussions**

gaZ

Conclussions

- Analysis potential SER-LUP policy
  - LU prediction + XRF
- Viability of a simple pattern-of-use predictor

gaZ

Conclussions

- Analysis potential SER-LUP policy
  - LU prediction + XRF
- Viability of a simple pattern-of-use predictor
- Great potential improvement (IPC)
  - int: 23%
  - fp: 47%

  64 register file
- Significant RF reduction
  - 36int reach 90% IPC of 224int
  - 33% RF reduction with 12% IPS increase

gaZ

Future work

- Real hardware mechanism
  - low-ported XRF
  - improve last-use predictor
- Goal
  - complex-effective mechanism
  - speedup near to upper bound

gaZ

Speculative Early Register Release

J. Alastruey[+], T. Monreal[+], V. Viñals[+], M. Valero[*]

[+]gaZ - Universidad de Zaragoza

[*] Universidad Politécnica de Cataluña

HiPEAC - High-Performance Embedded Architecture and Compilation

ACM International Conference on

Computing Frontiers, Ischia 2006

gaZ