Register Packing

Exploiting Narrow-Width Operands for Reducing Register File Pressure

Oguz Ergin*, Deniz Balkan, Kanad Ghose, Dmitry Ponomarev

Department of Computer Science

State University of New York - Binghamton

*currently with Intel Barcelona Research Center

Outline

- Introduction and motivations
- Register Packing:
  - Conservative Packing
  - Speculative Packing
- Results and discussions
- Conclusion

Introduction

- Implications of larger instruction windows
  - Increases register pressure
  - Generally dealt with by using large register files
- Large register files have:
  - Higher access time or require multi-cycle access
  - Higher energy dissipation
- Need to decrease the register file pressure

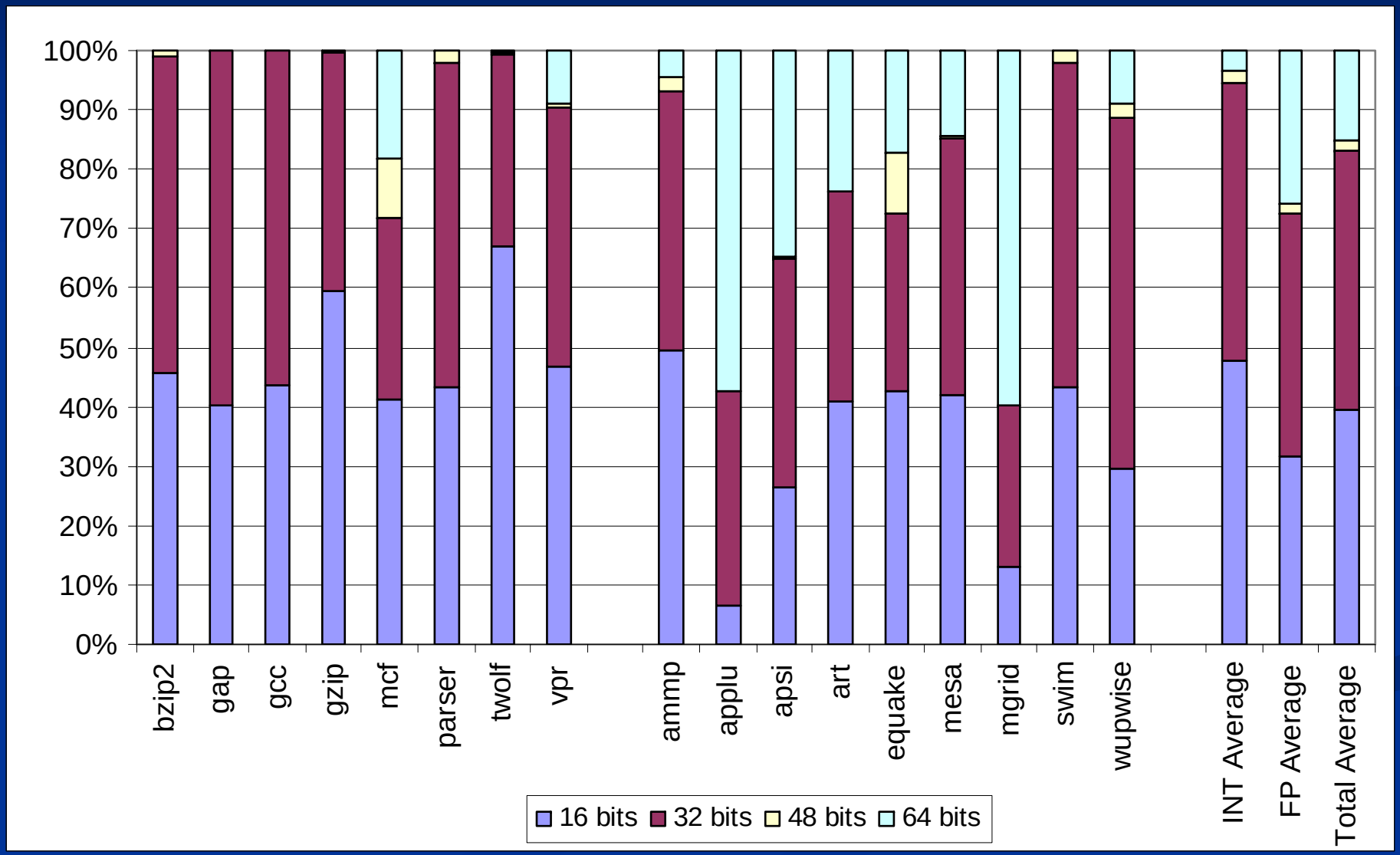
Motivations

- Many generated results have a lot of leading zeros or ones
  - Fewer bits are needed to represent the value
- Register files are thus not used efficiently

"Narrow" Values

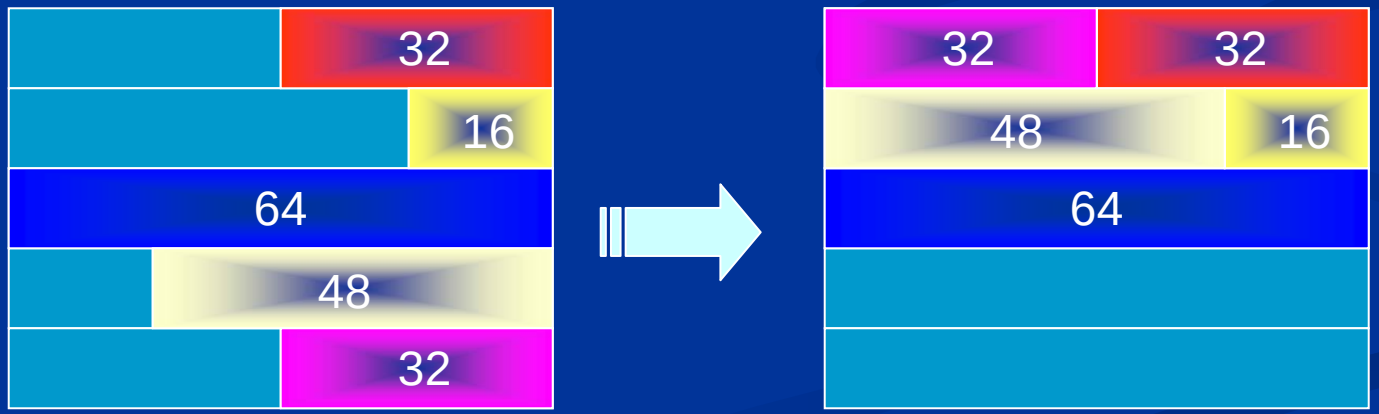
- Prefixes of all 1s can be replaced with a single 1 and the prefixes of all 0s can be replaced with a single 0.
  - 11111111 → 1 (width = 1)
  - 00000000 → 0 (width = 1)
  - 00000001 → 01 (width = 2)
  - 11111101 → 101 (width = 3)
  - 10101001 → 10101001 (width = 8)
- Narrow width operands do not use the full width of a register

Distribution of Widths



Exploiting Narrow Values

- Packing multiple results into a single physical register improves performance as the effective number of physical registers go up



Main Challenges

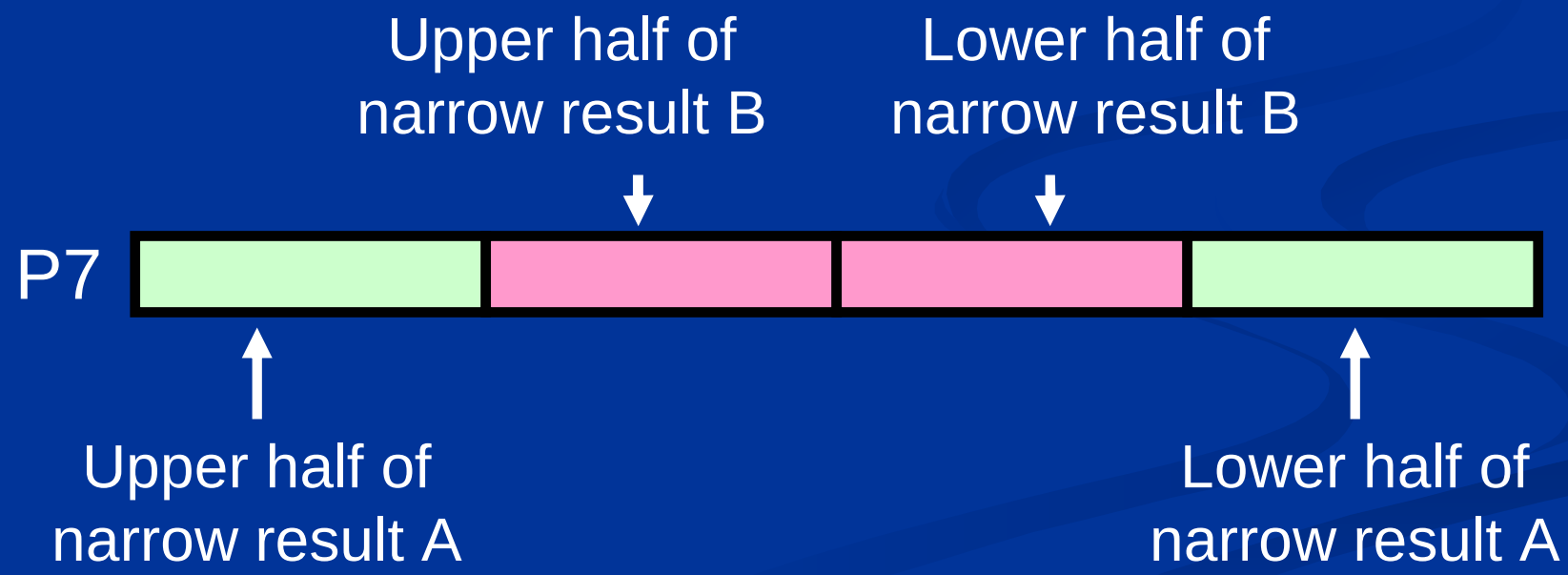
- Value widths are not known until the results are actually produced
  - Register allocation made to a result can change if the value turns out to be narrow
  - Consumers of the result have to be informed if it is reallocated to a different register based on its width
- If multiple results are packed into a common register some means must be provided to locate them unambiguously

Detecting Value Widths

- Have to quantize the widths to simplify implementation
  - Chunks of bytes or double bytes
- Width detection logic is embedded into the final stages of an execution unit
  - Techniques for detecting widths are well known – Leading Zero Detectors in floating point units

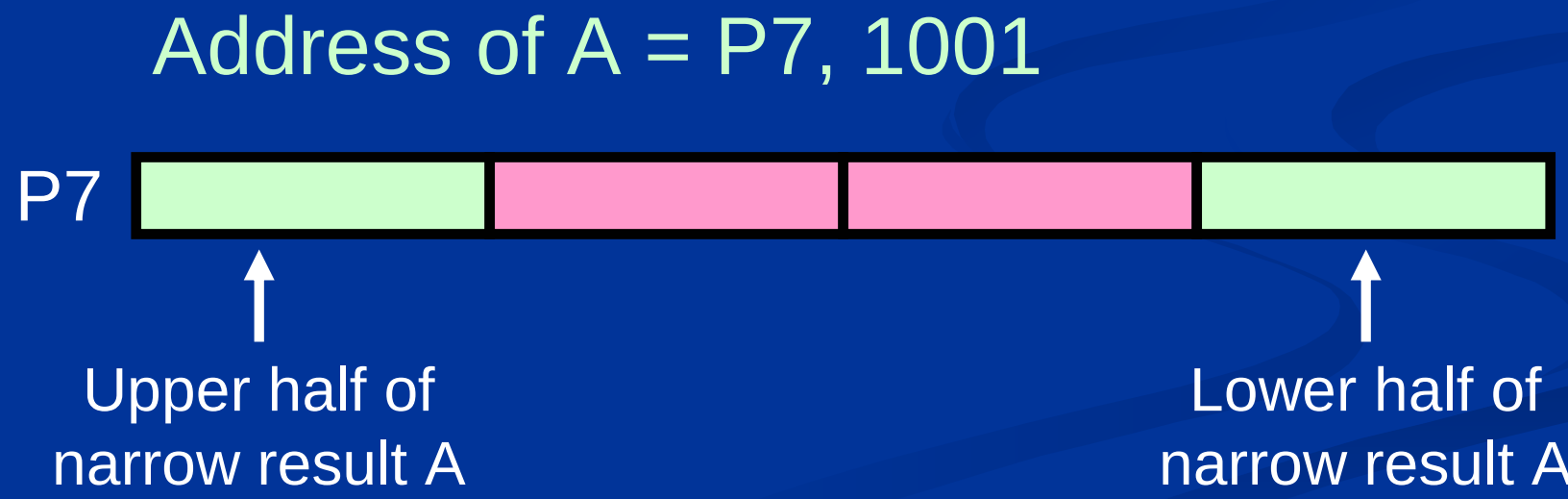
Storing Narrow Values in Registers

- Parts of a result do not need to be stored contiguously.

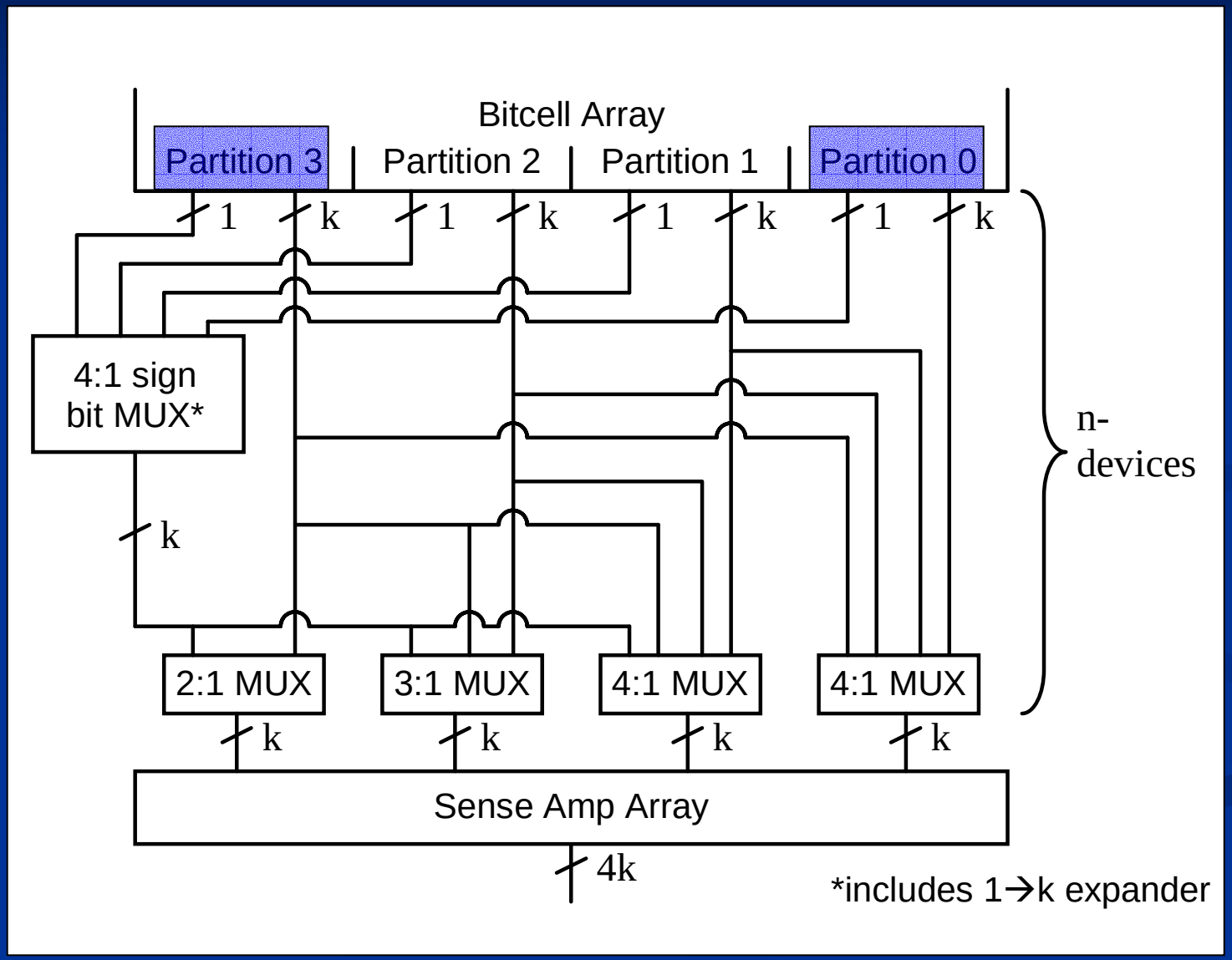


Addressing Narrow Values

- Use a bit mask to specify partitions holding components of the value along with the register address



Register Read Logic



Register Packing Alternatives

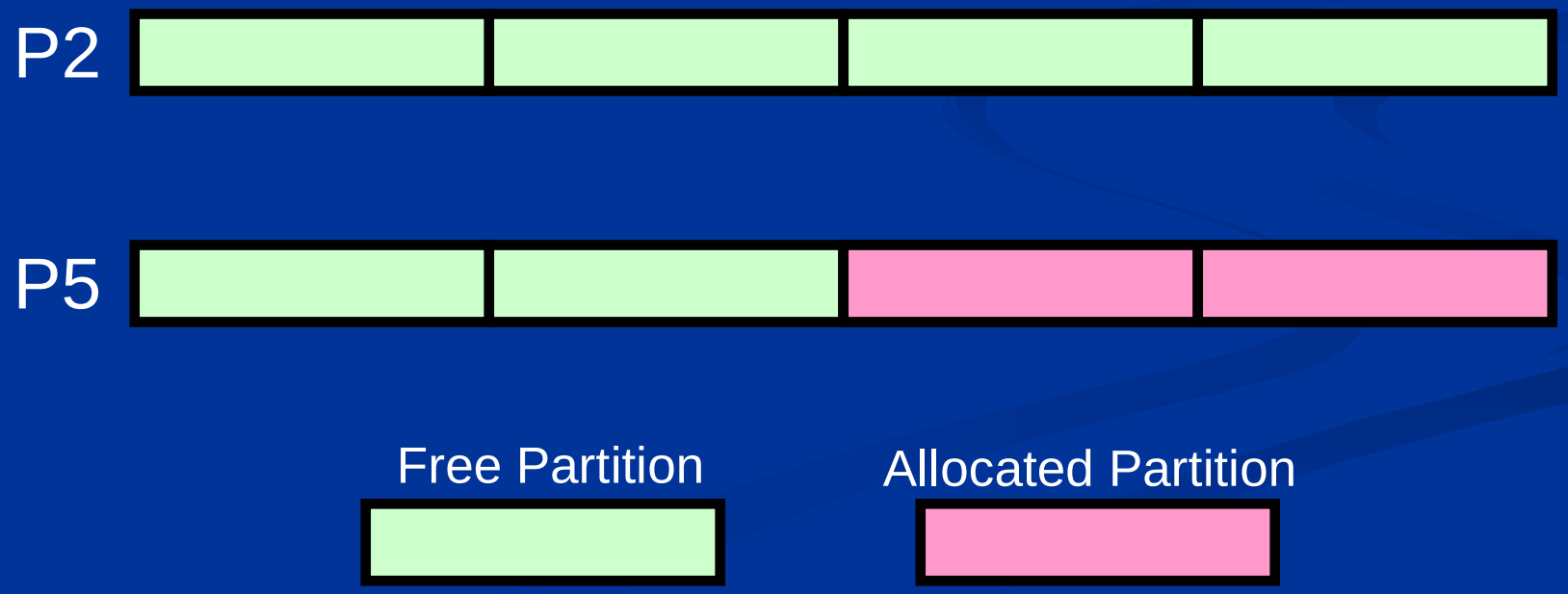
- Conservative Packing
  - Assume result to use the full width of a register at allocation time
- Speculative Packing
  - Predict the result width at allocation time and allocate accordingly

Conservative Packing

- Initially allocate a full-width register
- If the result turns out to be narrow:
  - Release the unneeded parts to the free pool
  - If there is a suitable partition: reallocate.

Conservative Packing

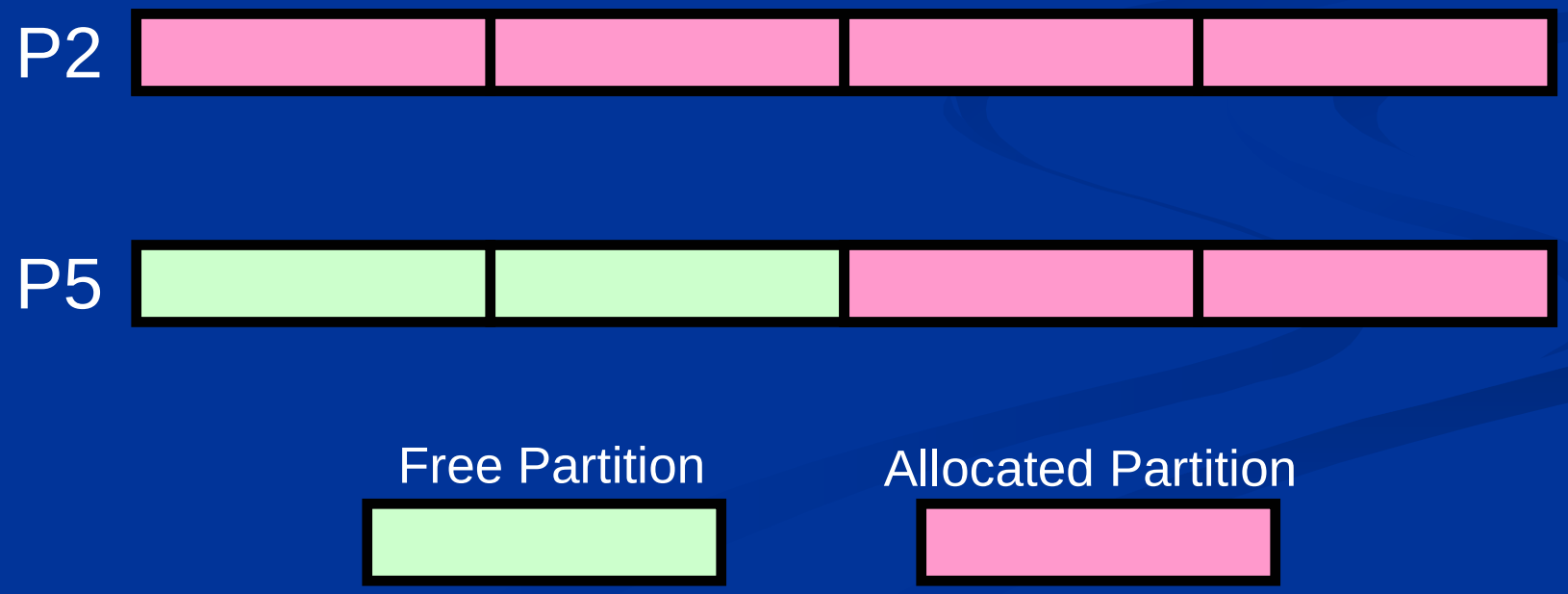
Instruction I is dispatched:



Conservative Packing

Instruction I is dispatched:

P2 is allocated

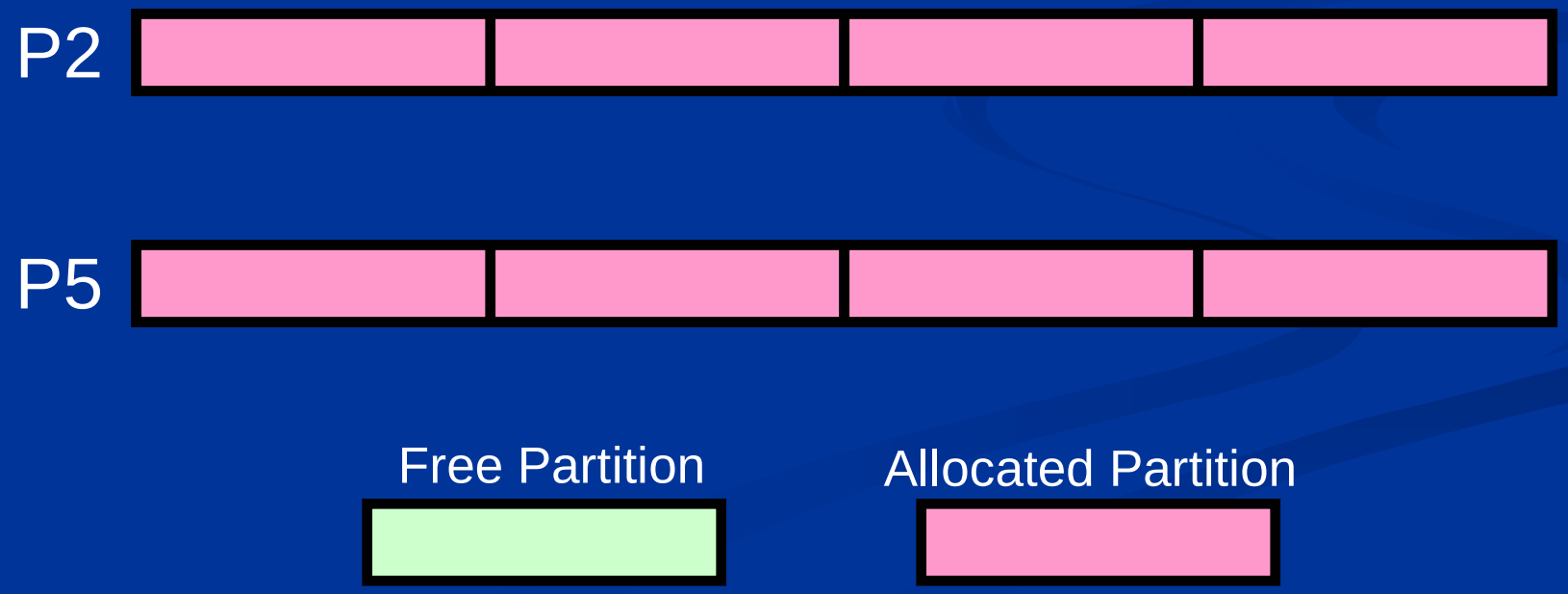


Conservative Packing

Instruction I is dispatched:

Width of result = 2 slots

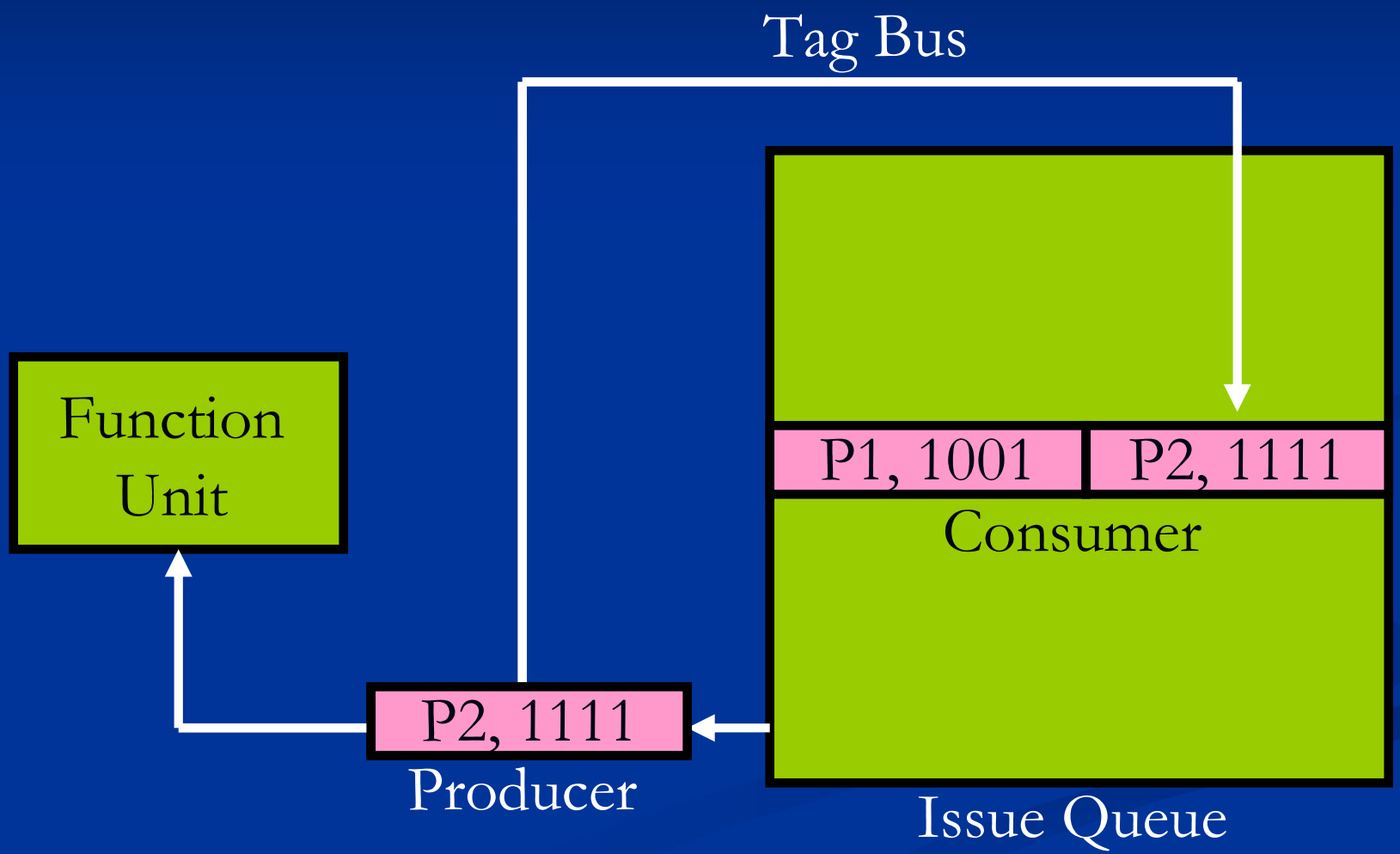
P5's upper half is allocated and P2 is released



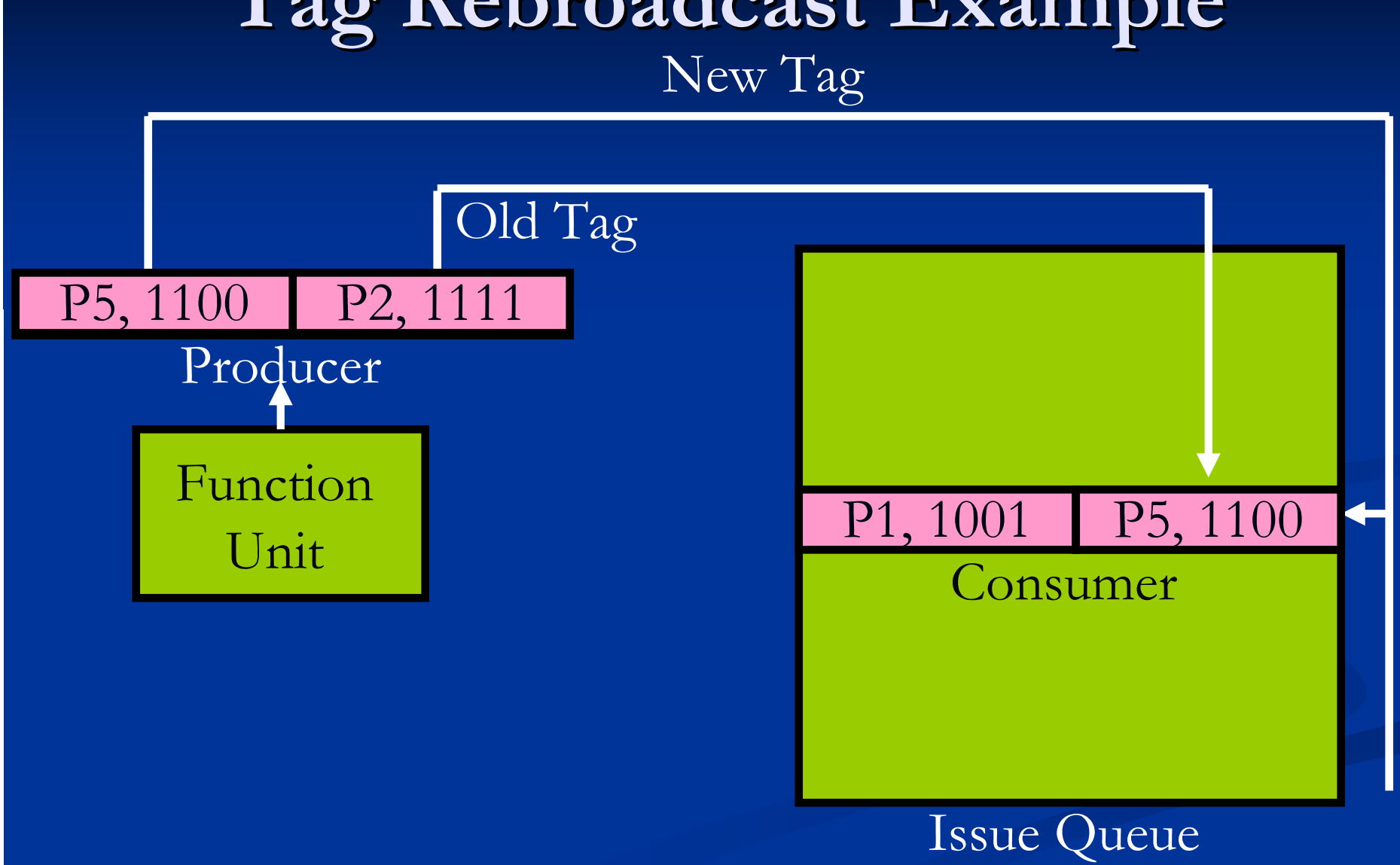
Taking Care of Reassignments

- Two broadcasts are needed
- First broadcast uses old tag (=originally assigned register id) to inform dependents that the result will be available shortly
- Second broadcast drives the old tag and the new tag (= newly-assigned register id + "parts" bits)
  - old tag is used to locate dependents
  - new tag picked up by matching entries and used later to read out source value from the register file

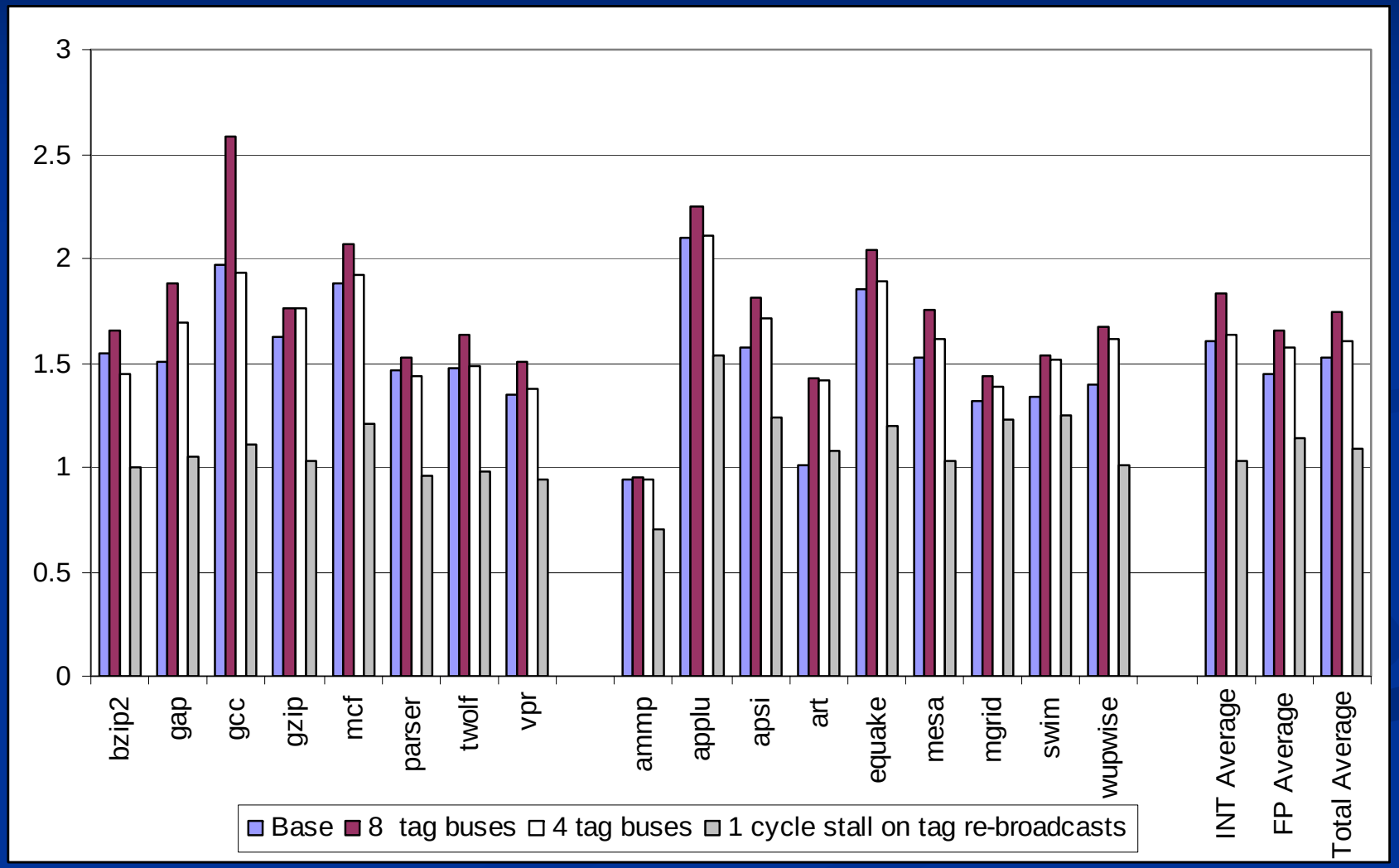
Tag Broadcast for Wakeup



Tag Rebroadcast Example



IPCs for Conservative Packing



Conservative Packing: Observations

- Extra broadcast is needed for all results that don't use all of the partitions within a register
- Performance is heavily constrained by the number of broadcast buses
  - 6% for 4 buses
  - 14% for 8 buses
  - -26% for 4 buses assuming an extra cycle delay for width estimation

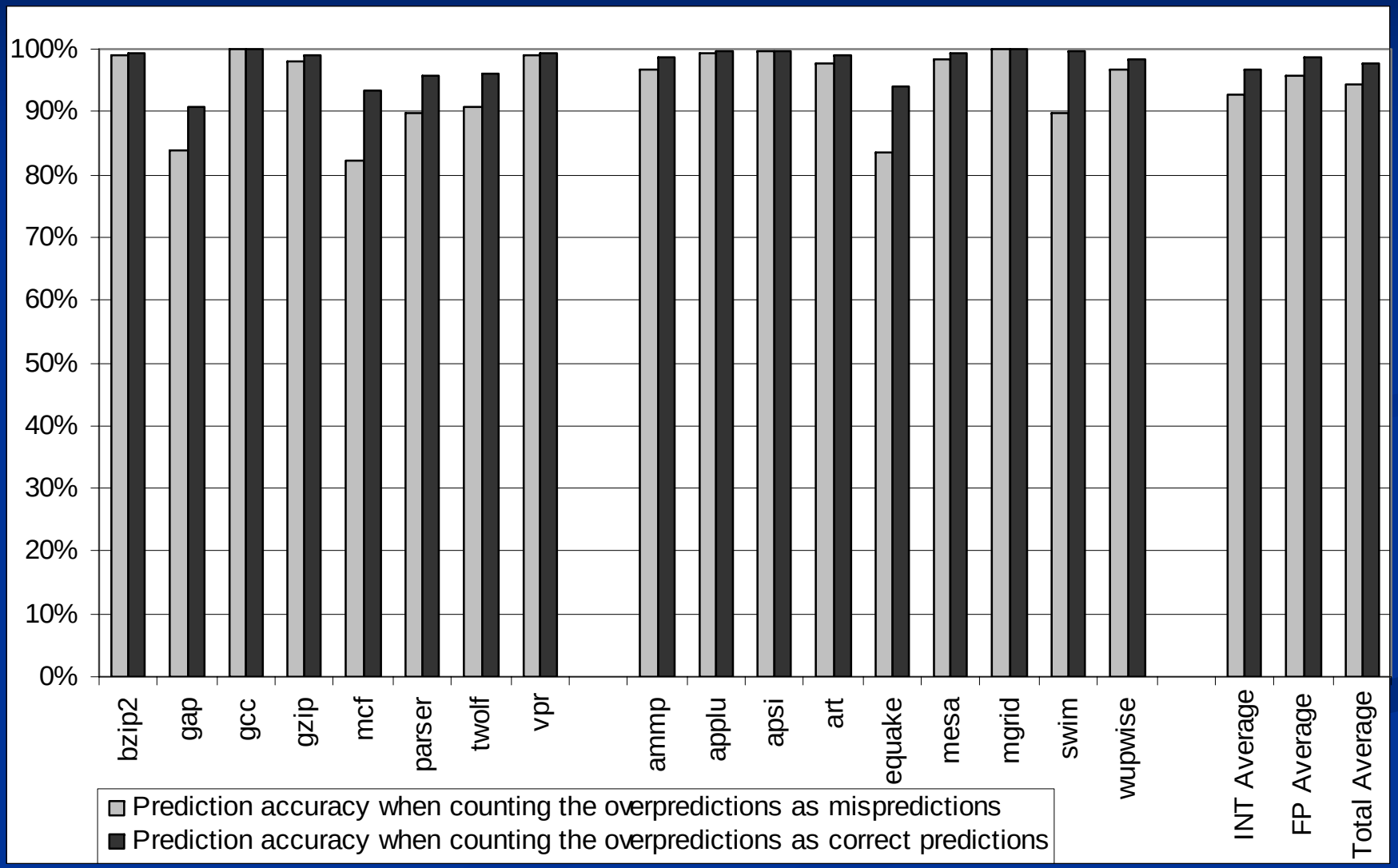
Speculative Packing

- Predict the width of the result and allocate accordingly
- Width overprediction: two choices here
  - Release unused parts of register – rebroadcast only the parts bits
  - Do not release unused parts – no rebroadcast is needed
- Width underprediction: requires reallocation and an update broadcast

Width Predictor

- Width prediction bits are maintained within the L1 I-Cache
- Prediction bits do no percolate down the memory hierarchy from L1
- Default prediction is full width
- Prediction bits are updated only on mispredictions

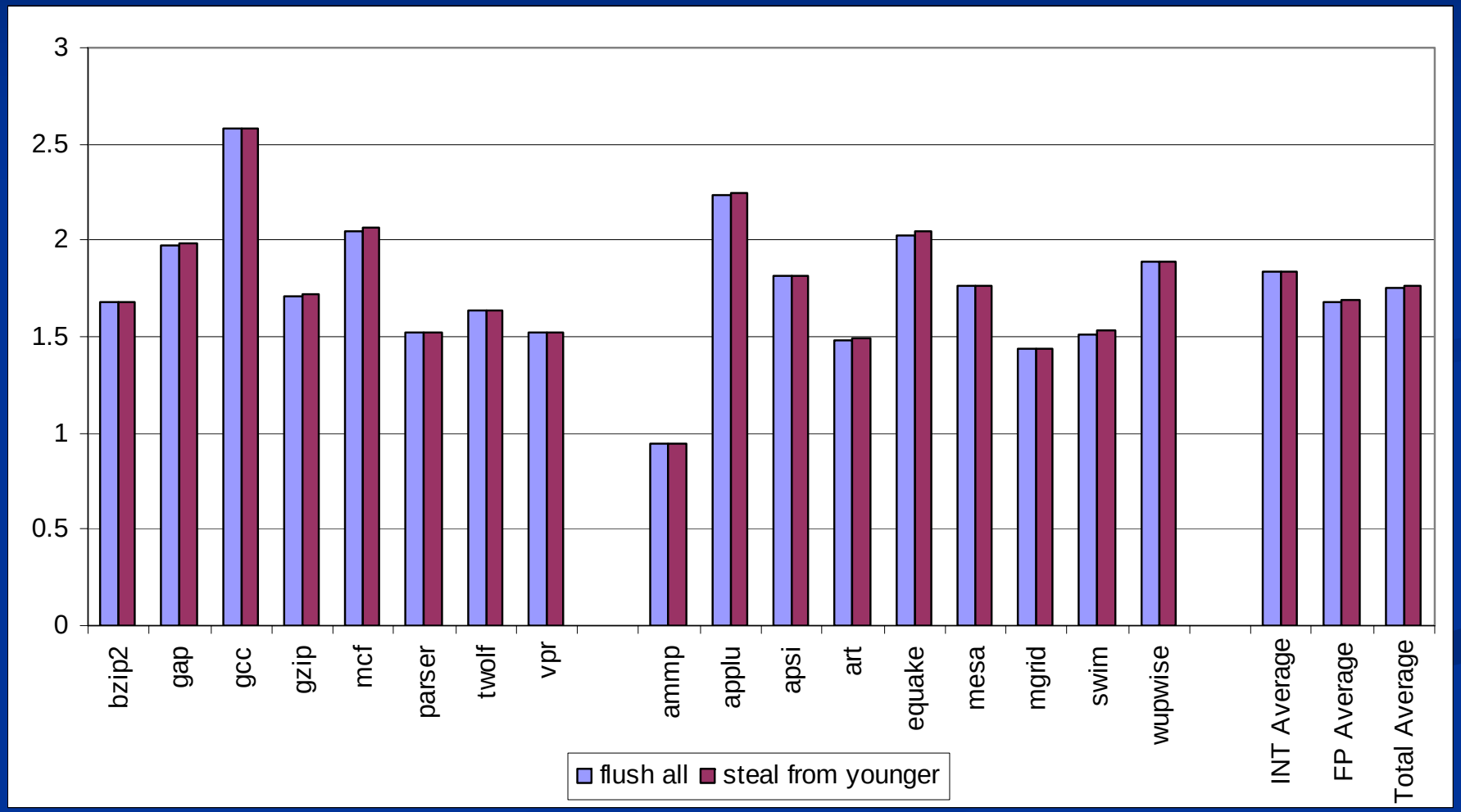
Width Prediction is Accurate !



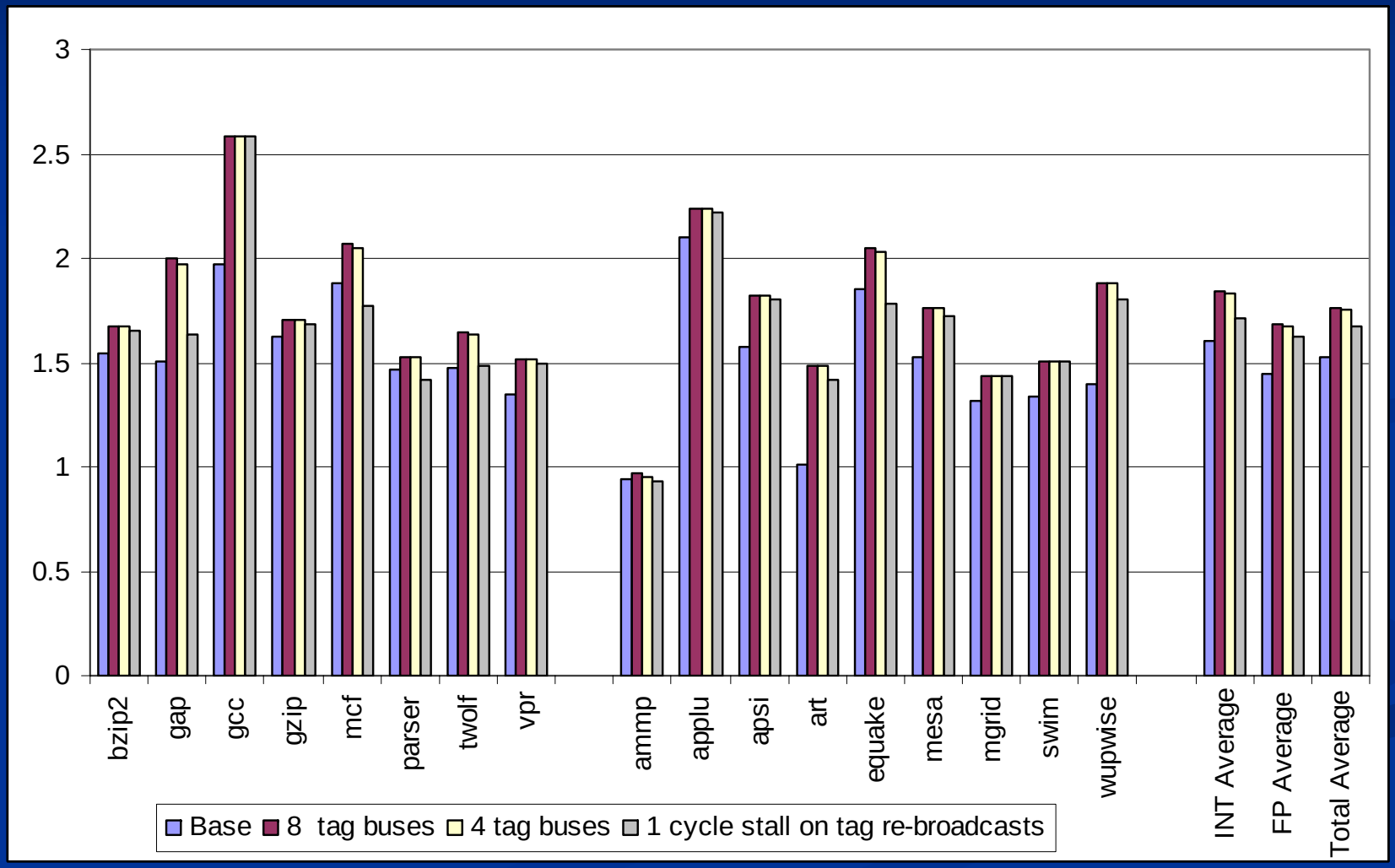
Deadlock Avoidance

- If there is a misprediction and there are no free register parts available:
  - Stall writeback and wait
    - This can still cause a deadlock if the instruction is the oldest in the pipeline
  - Create an exception and squash all instructions younger than the instruction (including itself)
  - Steal a register from a younger instruction and squash all instructions coming after the owner

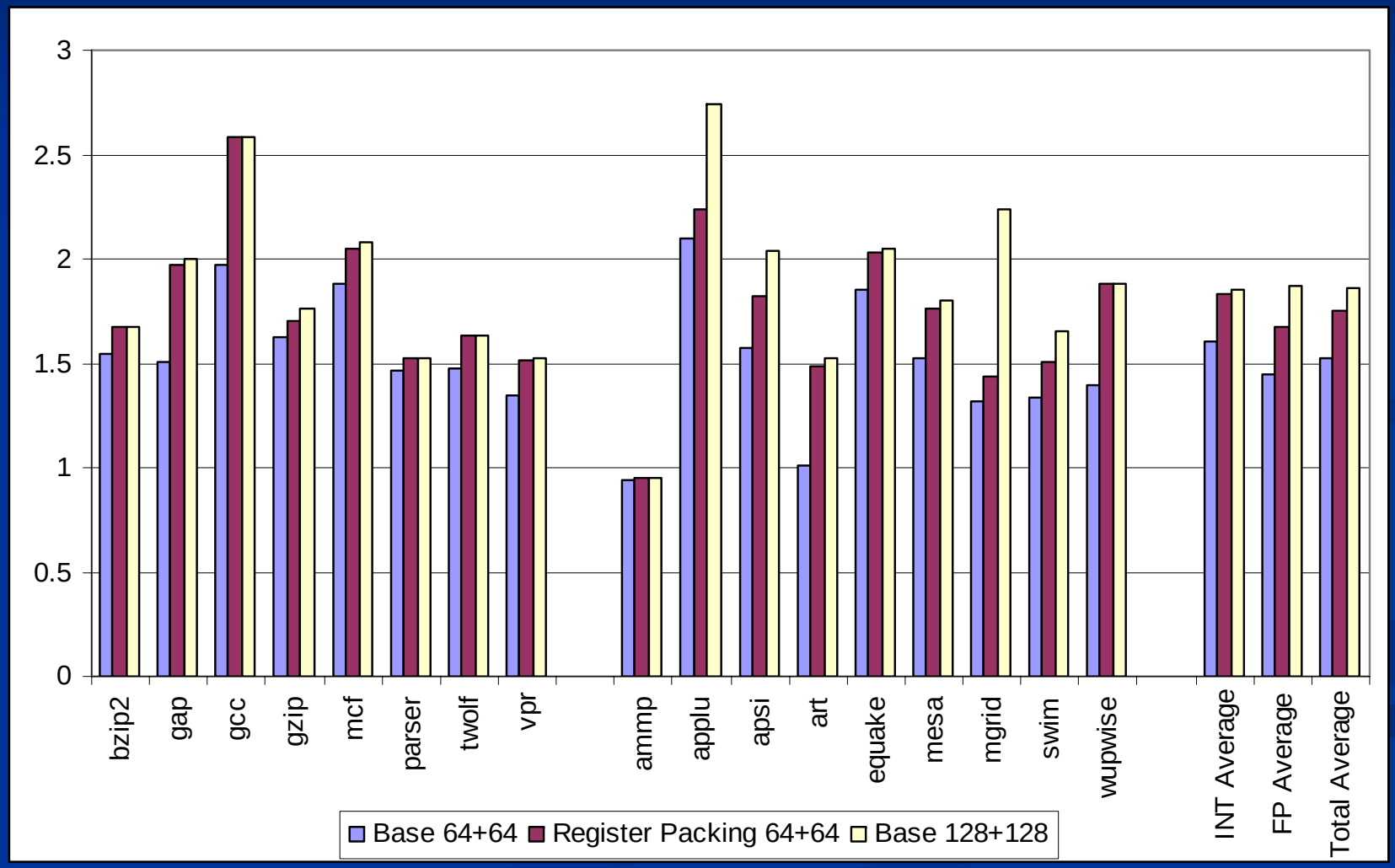
Comparison of Deadlock Avoidance Schemes



Speedups of Speculative Packing



Performance of Packing



Conclusions

- We proposed and evaluated two register packing schemes
- Because of the high number of tag broadcasts Conservative Packing suffers in performance
- Speculative Packing results in 15% IPC improvement on the average with 64 fp and 64 int registers (with tag bus sharing)

Thank You !

Oguz Ergin
Department of Computer Science
State University of New York - Binghamton
www.cs.binghamton.edu/~oguz

Intel Barcelona Research Center