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

The bar chart shows the distribution of widths for various applications, with bars color-coded by bit size (16 bits, 32 bits, 48 bits, 64 bits). The chart includes applications such as bzip2, gap, gcc, gzip, mcf, parser, twolf, vpr, ammp, applu, apsi, art, equake, mesa, mgrid, swim, wupwise, INT Average, FP Average, and Total Average.
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.

![Diagram showing the storage of narrow results A and B in register P7](image)

- Upper half of narrow result B
- Lower half of narrow result B
- Upper half of narrow result A
- Lower half of narrow result A
Addressing Narrow Values

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

Address of A = P7, 1001

P7

Upper half of narrow result A

Lower half of narrow result A
Register Read Logic

4:1 sign bit MUX*

2:1 MUX 3:1 MUX 4:1 MUX 4:1 MUX

Sense Amp Array

n-devices

*includes 1→k expander
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:

P2: Free Partition

P5: Allocated Partition
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

- Function Unit
- P2, 1111 (Producer)
- P1, 1001 and P2, 1111 (Consumer)
- Issue Queue
- Tag Bus
IPCs for Conservative Packing

![Graph showing IPCs for Conservative Packing](image-url)
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 not percolate down the memory hierarchy from L1
- Default prediction is full width
- Prediction bits are updated only on mispredictions
Width Prediction is Accurate!

![Bar chart showing prediction accuracy for various benchmarks]

- Prediction accuracy when counting the overpredictions as mispredictions
- Prediction accuracy when counting the overpredictions as correct predictions

The chart displays the prediction accuracy for various benchmarks, with bars indicating the percentage of accuracy for different categories.
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

flush all | steal from younger
Speedups of Speculative Packing

![Graph showing speedups of speculative packing for various benchmarks and architectural configurations.](image-url)
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
64-bit Apps vs. 32-bit Apps

- Can use fewer registers on 32-bit apps running on a 64-bit datapath: this may result in some energy savings
- See similar trends on data widths for 32-bit applications on 32 bit datapath
- Savings shown in running apps retargeted for 64 bits PISA ISA does have a fair number of 64 bit operands in FP benchmarks, integers holding addresses etc.
32-bit value widths