Banked Register Files for SMT Processors

Jessica H. Tseng and Krste Asanović
MIT Computer Science and Artificial Intelligence Laboratory
200 Technology Square, Cambridge, MA 02139
{jhtseng,krste}@csail.mit.edu

Abstract

Multiported register files are a critical component of high-performance superscalar microprocessors. Deeper pipeline speculation and higher instruction-level parallelism (ILP) of current processor designs push a growing requirement on both the number of ports and the number of registers. These increasing demands on register files cause the area of a conventional multiported regfile to grow more than quadratically with issue width [7]. The trend towards simultaneous multithreading (SMT) further increases register count as separate architectural registers are needed for each thread. For example, the proposed eight-issue Alpha 21464 design had a regfile that occupied over five times the area of the 64 KB primary data cache [3]. Hence, we examine the designs of banked multiported register files that consist of multiple interleaved banks of fewer ported register cells to reduce power, area, and access time. In this talk, we will present results that extend our previous work [4] on banked register files to SMT processors.

Banked register files designs have been shown to provide sufficient bandwidth for a superscalar machine, but previous proposed designs had complex control structures that would likely limit cycle time and add to design complexity [6, 1, 2]. We present a banked multiported regfile design together with a much simpler and faster control logic suitable for a deeply pipelined high-frequency superscalar processor [4]. Our control scheme does not place any register bank arbitration in the critical wakeup-select loop but instead speculatively issues potentially conflicting instructions. Bank conflicts occurs when too many instructions are trying to read or write the same bank at the same cycle. If any conflicts are found after issue, a pipelined recovery scheme quickly repairs the issue window and reissues conflicting instructions. In contrast to previous work [6, 1, 2], all conflicts are detected and resolved in one pipeline stage such that no write buffering or pipeline stalls are required.

The extra pipeline stage used for port arbitration and the possibility of bank conflicts in our design can impact processor performance. The additional pipeline stage causes an increase in branch misprediction latency while instances of a bank conflict add penalty cycles to repair the pipeline and delays the issuing of dependent instructions. Fortunately, the number of bank conflicts can be kept within a few percent of total instructions if we can remove the correlation between accesses to the same bank. It is observed that instructions which become ready in the same cycle tend to be issued together and some architectural registers are used more frequently than others. To avoid unnecessary read port contention and to remove the correlation between the same-cycle

read accesses, we implement two optimization techniques–*bypass-skip* and *read-sharing*. Bypass-skip avoids competing for register read ports for operands that will be sourced by the bypass network. Read-sharing fetches only once per register value from the regfile and shares it among the instructions that request the value.

To evaluate our work, we modified the SMTSIM [5] simulator to keep track of a unified physical register file organized into banks for both superscalar and SMT processor. One might expect that extending banked register file schemes to SMT processors would degrade performance more than in superscalar microprocessors because of SMT's higher IPC and register counts. Our initial data surprisingly reveals that the banked register files work better for the SMT processors than the single thread superscalar processors. This result is due to the SMT's ability to hide the branch misprediction penalty, as when one thread experiences a misprediction, other threads can continue to execute instructions. For an eight-issue SMT processor with 512 physical registers, by adopting a 16-banked register file design with four read ports and two write ports per bank, we can reduce regfile area by a factor of seven over a monolithic design while decreasing IPC by less than 2%. The ability to reduce area significantly with minimal performance degradation should make this approach attractive for chip multi-processors which are aiming to provide the highest possible thread throughput at low cost.

References

[1] R. Balasubramonian, S. Dwarkadas, and D.H. Albonesi. Reducing the complexity of the register file in dynamic superscalar processors. In *MICRO-34*, December 2001.

[2] I. Park, M. D. Powell, and T. N. Vijaykumar. Reducing register ports for higher speed and lower energy. In *MICRO-35*, Istanbul, Turkey, November 2002.

[3] R. P. Preston et al. Design of an 8-wide superscalar RISC microprocessor with simultaneous multithreading. In *ISSCC Digest and Visuals Supplement*, February 2002.

[4] J. H. Tseng and K. Asanović. Banked multiported register files for high-frequency superscalar microprocessors. In *ISCA-30*, June 2003.

[5] D. M. Tullsen, S. Eggers, and H. M. Levy. Simultaneous multithreading: Maximizing on-chip parallelism. In *ISCA-22*, 1995.

[6] S. Wallace and N. Bagherzadeh. A scalable register file architecture for dynamically scheduled processors. In *Proc. PACT*, October 1996.

[7] V. Zyuban and P. Kogge. The energy complexity of register files. In *Proceedings 1998 International Symposium on Low Power Electronics and Design*, pages 305–310, August 1998.