The Design of a Register Renaming Unit

Benjamin Bishop, Thomas P. Kelliher[1], Mary Jane Irwin
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802
bishop@cse.psu.edu

Abstract

Register renaming is often used to improve performance in many high-ILP processors. However, there is a lack of publications regarding register renaming hardware design. This paper presents a detailed look at one possible implementation of a register renaming unit, as well as some possible optimizations.

1. Introduction

Register renaming is an important technique that is used to increase performance in many high-ILP processors. In fact, Johnson [2] states that register renaming alone increased the performance of a specific processor by 36%. Techniques for efficient register renaming are not currently well documented in the literature, however. In this paper, we attempt to fill this gap by presenting an HSPICE-verified design of a register renaming unit. In addition, we discuss factors in register renaming performance and examine some possible optimizations.

2. Design of a register renaming unit

The Instruction Set Architecture (ISA) provides a certain number of registers to the programmer (ISA registers). In order to remove false data dependencies, these registers can be transparently mapped onto a larger set of registers, which are hidden from the programmer (physical registers). These data dependencies are typically introduced due to the lack of ISA registers and the tendency of compilers to conserve registers [1].

The register renaming process can be described as follows: A group of instructions enter the register renaming unit. In parallel, data dependencies between these instructions are determined and possible source operand physical registers are determined by way of a Register Alias Table (RAT) and free list lookup. The assignment of source operands to physical registers depends on the outcome of the dependency check operation. When both the dependency check and register lookup operations are completed, new physical destination registers are chosen from the free list and are written into the RAT. Eventually, when a particular ISA to physical register binding has been replaced and all of the instructions using that physical register have completed or have been discarded, that physical register can again be added to the free list.

We have generated layout for the timing critical functions of the register renaming unit. This layout[2] was created by hand in Magic [6] and verified in HSPICE [4]. Issue widths of 1, 2, 4, and 8 have been considered. The following subsections present details about our implementation.

2.1. Register alias table

The RAT hardware was implemented as a specialized SRAM memory. It consists of sense amp and decode hardware along with an array of RAT cells connected by bitlines and wordlines. The floorplan is very similar to a typical SRAM memory, with a large array of cells with a small amount of additional decode and sense amp logic. For the decode logic, the address line drivers were assumed to be external. The RAT has 32 8-bit entries, appropriate for a processor with 32 logical and 256 physical registers. The RAT is also multiported, with the number of ports depending on the issue width of the processor. One important feature of the RAT is that, in order to support speculative execution, each RAT cell must contain a shift register for saving and restoring the state of the RAT. These shift registers form a "shadow table" that saves the RAT state when a branch is encountered and restores it when a branch misprediction is discovered. See Figure 1 for a detailed schematic of an example RAT cell.

[1]Goucher College, Baltimore, MD

[2]Layouts using MOSIS scalable design rules are available at http://www.cse.psu.edu/~mdl/glvlsi.tar.

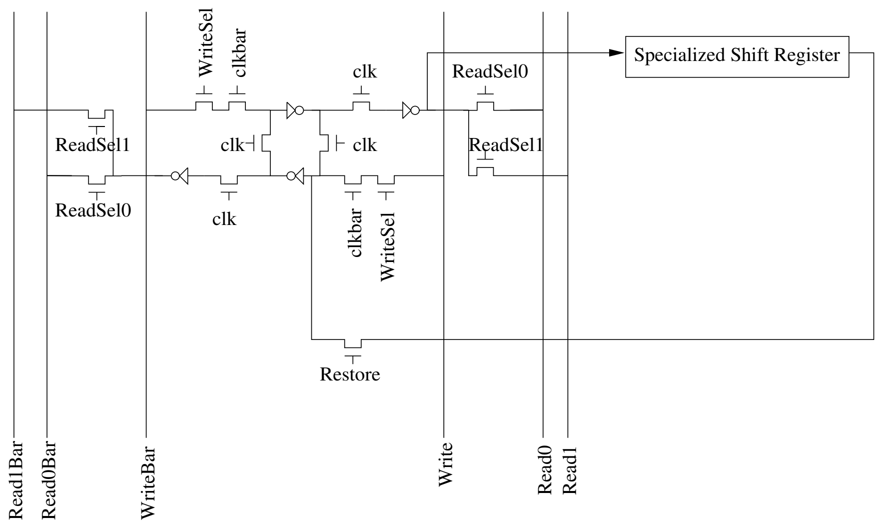

Figure 1. The schematic of an example register alias table cell.

2.2 Dependency Check Logic Design

The DCL was implemented with a group of comparators and priority encoders as shown in Figure 2. Figure 3 gives a schematic for the comparators used. Figure 4 shows one possible implementation of three input priority encoder from Figure 2. The I3 signal is given the highest priority and I1 the lowest. The select lines give the binary encoding of the highest priority set input signal. If no signals are set, the select lines are set to “00”. The random logic equations and the NAND/NOR mapping was accomplished through misII[3]. In order to limit fan-in, all logic was decomposed to 2-input gates.

3. Timing Information

Worst case circuit delay can vary dramatically with changes in technology. The circuits that we describe here (RAT/DCL) range in delay from 1.05ns to 2.13ns assuming a 3.3V supply voltage under MOSIS .35um simulation parameters. In figures 5 and 6 we show sample timing waveforms for the worst case delay for both the RAT and DCL assuming an issue width of 8.

4. Potential Optimizations

Assuming differential sense amps are being used, RAT bitline delay can be improved by using an asymmetric read port approach. In this scheme, the furthest readport for the cell value is made the nearest read port for the inverse of the cell value. This approach decreases the variation in read port delay, thus decreasing the maximum readport delay.

Again assuming differential sense amps, the maximum read delay can be reduced by equalizing each bitline and its complement before attempting a read operation. This approach decreases the variation in delay between bitlines and their complements, thus decreasing worst case bitline delay.

5. Conclusion

We have presented an HSPICE verified design for a register renaming unit. Additionally, optimizations have been suggested as possible routes to improve performance.

References

[1] A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers Principles, Techniques, and Tools*. Addison-Wesley, 1986.

[2] M. Johnson. *Superscalar Microprocessor Design*. Prentice Hall, 1991.

[3] R. Katz. *Contemporary Logic Design*. Benjamin Cummings/Addison Wesley, 1993.

[4] Meta Software. *HSPICE Users's Manual*. Meta Software, 1996.

[5] R. Nadkarni. Register renaming in dynamically scheduled processors. *M.S. Thesis, Electrical Engineering Department, Penn State University*, May 1998.

[6] J. K. Osterhout, G. T. Hamachi, R. N. Mayo, W. S. Scott, and G. S. Taylor. Magic: A VLSI layout system. In *Proc. 21st Design Automation Conf.*, pages 152–159. ACM/IEEE, 1984.

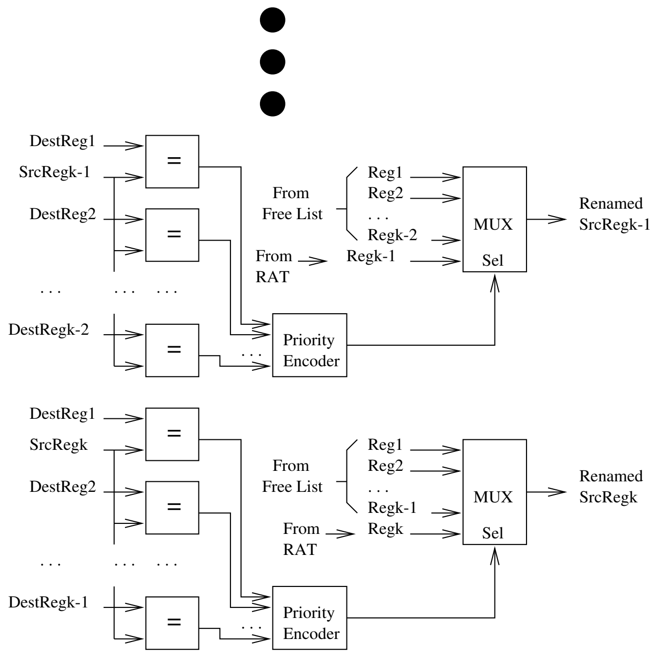

Figure 2. Block diagram of the dependency check logic.

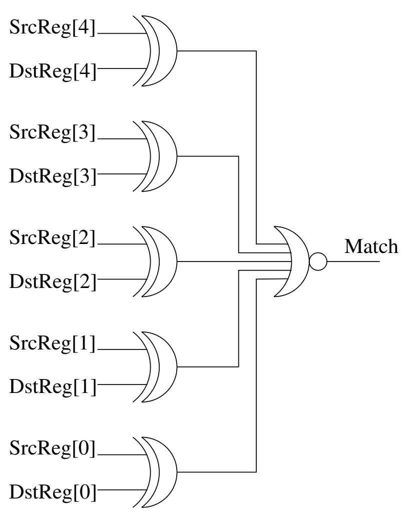

Figure 3. Schematic of the comparator hardware.

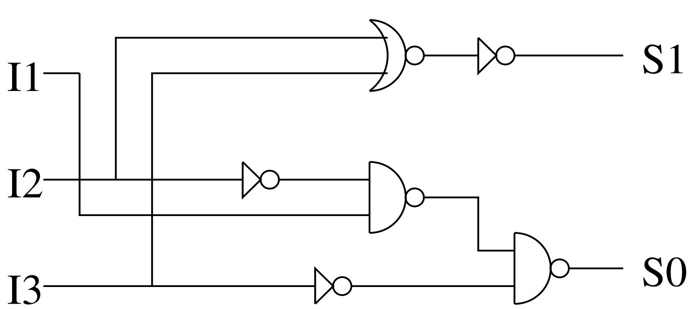

Figure 4. A gate-level example of a priority encoder.

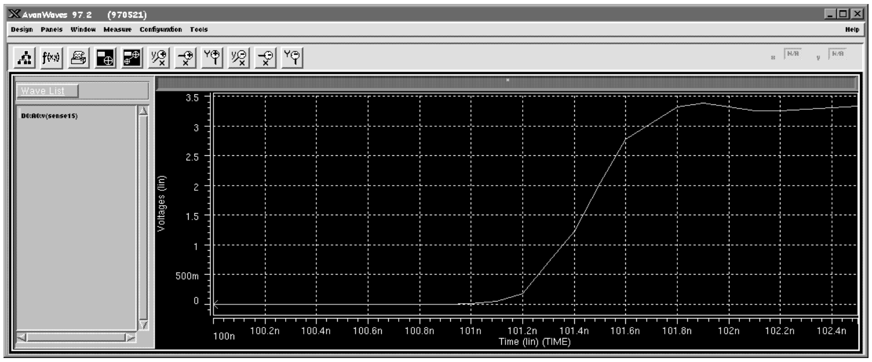

Figure 5. Critical path RAT delay waveform (for .35um/3.3v/iw8 delay=1.45ns)

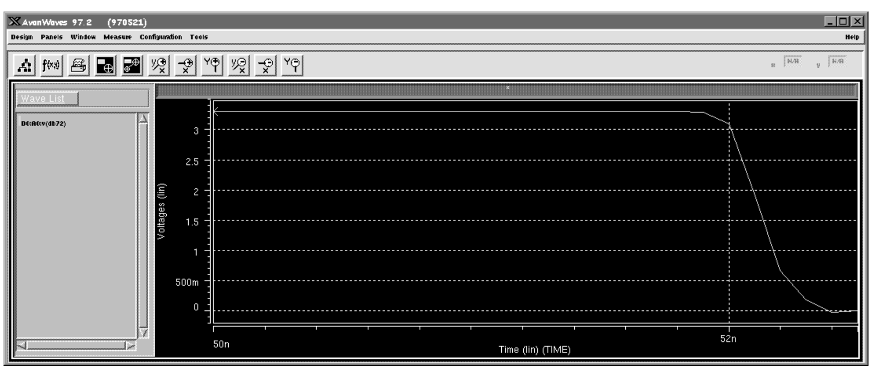

Figure 6. Critical path DCL delay waveform (for .35um/3.3v/iw8 delay=2.13ns)