A Superpipeline Approach to the MIPS Architecture

Asghar Bashteen, Ivy Lui, Jill Mullan

MIPS Computer Systems
950 DeGuigne Drive
Sunnyvale, CA 94086

ABSTRACT

The next generation MIPS CMOS microprocessor, the R4000, uses a technique called superpipelining to achieve a high level of performance. This paper will discuss the evolution of the R4000 pipeline from the R3000 pipeline and the reasons why a superpipelined micro-architecture is chosen.

Keywords: superpipeline, superscalar, VLIW, issue restriction

1. Introduction

When calculating the performance of a computer system, the following equation describes the amount of time it takes to execute a given task:

$$T = i * cpi * t$$

where i = # of instructions in the task
cpi = the number of clock cycles it takes to execute an instruction
t = time per clock cycle

In order to reduce the time it takes to execute a given program, any or all of the three factors, i, cpi, or t, need to be reduced.

As long as the pipeline is full and there are no stalls, one instruction will execute in each cycle in a single instruction-issue machine.

Two of the factors which prevent a pipeline from being full are data dependencies for load instructions and the latencies for taken branches. The MIPS compiler can schedule useful instructions in the delay slots, thus keeping the pipeline full by taking advantage of instruction scheduling to reduce "i".

Sections 2 and 3 describe how the R3000 pipeline evolves to take advantage of process technology to reduce "t". Section 4 describes three techniques to reduce the "cpi" by exploiting instruction level parallelism. These techniques are superscalar[1], superpipeline[1] and VLIW[1]. It also explains why the superpipeline approach is chosen for the R4000. Section 5 describes the R4000 pipeline in more detail. Finally, section 6 mentions several improvements possible for the future.

2. The R3000 Pipeline

The R3000, the second generation microprocessor in the MIPS R Series Architecture, uses pipelining to achieve an execution rate of nearly one instruction per cycle. (The R3000 has a native cpi of 1.36 across a series of UNIX benchmarks). The R3000 utilizes a 5-stage non-interlocking pipeline to achieve its execution rate.

The pipe stages are defined as:

1. IF--Instruction Fetch. The I Cache is accessed and the tag is checked.

2. RF--Register Read

3. ALU--Add, logical, shift computation

4. DC--Data Cache Access and tag check

5. WB--Register File write

Figure 1 shows a detailed description of the R3000 pipeline. All the stages, except the cache access stages, really require half a cycle.

During the first half of the IF stage, an instruction address is selected by the branching logic and pre-translated to a physical address (ITLB). In the second half of the IF stage and the first half of the RF stage, the instruction is retrieved from the I-cache (ICACHE). In parallel, the TLB performs a full translation of the instruction's virtual address to check the quick translation and to refill the

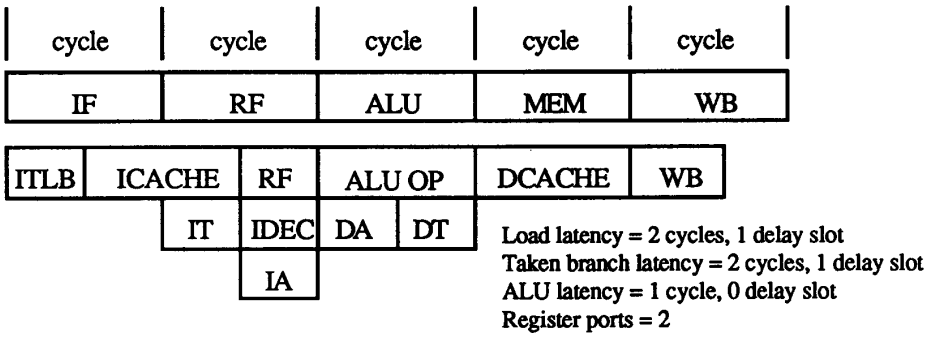

Figure 1: Detailed R3000 Pipeline

micro TLB (IT). During the second half of the RF stage, three blocks operate simultaneously to decode the instruction (IDEC), fetch operands from the register file (RF), and compute a new instruction address (IA). The simple and regular instruction formats of the MIPS R-Series Architecture allow exploitation of functional parallelism in the R3000.

During the ALU stage, several different operations may occur, depending upon the type of instruction. The R3000 implements the MIPS R Series Architecture which is a load/store or register-based architecture in which memory accesses and data manipulations are performed in separate instructions. This allows data manipulation to be overlapped with memory accesses. If the instruction is a register-register instruction, the arithmetic or logical operation is performed (ALU OP). If the instruction is a memory reference (load or store) then the data virtual address is calculated (DA) and translated into a physical address (DT). The MEM stage is used to access the D-cache (DCACHE) if the instruction is a memory reference. In the final WB stage, the data is written into the register file during the first half of the cycle.

3. Optimizing the Pipeline for the R4000

The goal of the R4000 design is to improve performance over that of the R3000 while exploiting instruction level parallelism. One way in which this is done is through migration to a more advanced technology. This not only allows the cycle time to be cut in half, but it also allows instruction and data caches to be incorporated on chip and for the access time of the register file to be cut in half.

The R3000 already contains optimizations which can exploit the new process technology. Much of the hardware in the R3000 is used twice per cycle. The register file is used in one half of a cycle for reading, and in the other half for writing. While the R3000 ALU takes a full cycle, the 32-bit address adder takes only half a cycle, which implies that an ALU with similar latency is possible. The TLB and cache interface are used in one half of a cycle for instructions, the other for data. By incorporating on-chip instruction and data caches, separate instruction and data busses, and separate TLB's, the use of the TLB and caches is reduced to once per cycle which allows for further pipeline enhancements.

Figure 2 shows a modified R3000 pipeline with reduced latencies. Each of these cycles is half as long as the R3000 cycles because of the better process technology and improved circuit design of the R4000. The instruction cache and data cache stages take only half as long as the R3000 because the caches are on the same chip. Also, improved circuit design speeds up the register read and write by a factor of two. The tag check for the data cache is also moved to a separate cycle to allow more time for the data cache access and tag check. This can be done because of the direct-mapped data cache, allowing the data to be used by a dependent instruction prior to the completion of the tag check.

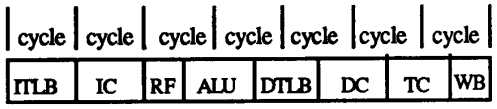

Figure 2: Modified R3000 Pipeline with reduced latencies

Because the R4000 caches are on-chip, the virtual-to-physical address translation can delay the cache access. This latency is improved by implementing virtually indexed caches and going to a parallel cache access and address translation. Figure 3 shows the optimized R3000 pipeline with parallel TLB and cache accesses.

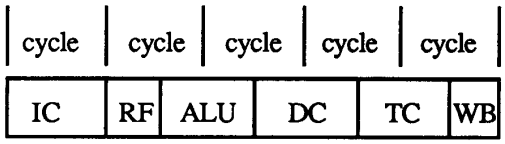

Figure 3: Optimized R3000 Pipeline with parallel TLB and cache accesses

4. Instruction Level Parallelism

Another way to reduce the CPI is to exploit instruction-level parallelism. Techniques to accomplish this include:

1. superscalars
2. superpipelines
3. VLIWs

A superscalar machine is one that can issue more than one independent instruction per clock cycle. Two or more instructions are fetched and decoded in parallel and are then executed, also in parallel.

To implement a superscalar machine, execution units and other processor internal resources must be replicated to avoid restrictions on the instruction issue. For example, to execute two load operations in the same cycle, the D-cache needs to be accessed twice in a given cycle. Similarly, to execute two ADD instructions in the same cycle requires two adders. If hardware functional units are not replicated, then certain combinations of instructions cannot be executed in parallel and hardware must exist to enforce the instruction issue restrictions. For example an implementation may allow the issue of one memory, one register, and one control instruction at a time, but not two of any identical type of instructions. Compilers must also be smart enough to schedule instructions efficiently and therefore existing binaries are not optimized to exploit the superscalar architecture.

Not only does much of the logic in a superscalar implementation need to be replicated, but complex control circuitry is needed to check for dependencies between multiply-issued instructions. This added complexity limits how much the cycle time can be reduced. Also, replicating functional units and incorporating complex control circuitry means that there is less area on the chip for other resources, such as larger on-chip caches or special function units.

In a superpipelined system, existing hardware is used several times per cycle by inserting pipeline registers to split up each pipe stage. Essentially each superpipeline stage operates at a multiple of the base clock frequency, the multiple depending upon the degree of superpipelining. Figure 4 illustrates the optimized R3000 pipeline using a superpipeline of degree two. This pipeline dispatches two instructions per cycle.

The key to superpipelining is to divide up the pipe stages so that each stage takes approximately the same amount of time. This requires careful partitioning of structures like caches which are usually singular structures. Pipeline frequencies must also increase as the depth of the pipeline increases, utilizing process technology and circuit design techniques at their best.

In a system incorporating VLIW, or very long instruction word, the instructions are typically hundreds of bits long. Each instruction can specify several different operations issued to the various functional units on the processor. The selection of which operations to be executed in parallel is performed at compile time. The instruction decode and scheduling logic circuitries are simpler than a superscalar machine, but the code can become much larger because the fixed format of the instruction includes bits for unused operations. Also, the degree of parallelism of a VLIW machine cannot be increased without requiring a new instruction format, thus machines do not maintain binary compatibility as the degree of parallelism increases. For these reasons, a VLIW approach is not considered as a viable alternative for the R4000.

A superpipelined R4000 implementation is chosen because it has several advantages over a superscalar implementation. First of all, less logic is needed because there is no need to replicate the internal functional units. Any available chip space can be used for other performance enhancement features or for the reduction of overall chip cost. Also, control logic is simpler because there is no need to issue instructions to several functional units in parallel, nor is there a need to keep track of issue restrictions. These advantages result in a faster cycle time, shorter design and test times, and a balanced improvement of integer and floating-point

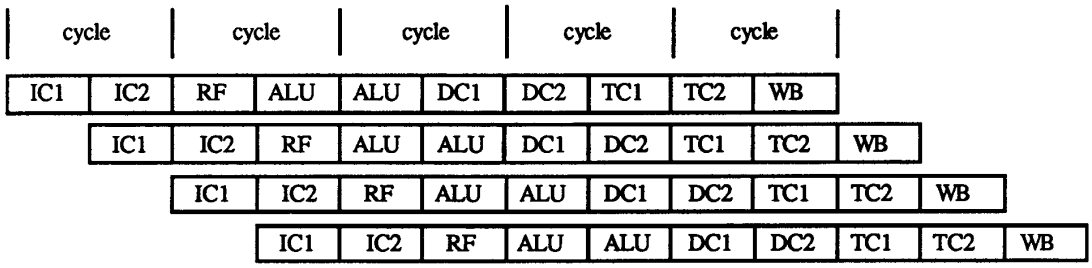

Figure 4: Superpipelined Implementation of the Optimized R3000 Pipeline

performance. Lastly, a superpipelined implementation gets more immediate performance boosts from existing binaries without recompilation. Further improvements can be obtained from compiler optimizations and advanced techniques such as software pipelining.

However, the superpipelined implementation shown in figure 4 needs further enhancements because the second of a "pair" of instructions in this implementation finishes a half cycle after a superscalar implementation at the same frequency and the same degree of parallelism.

5. The R4000 Superpipeline

Using the pipeline in figure 4 as a base, a high performance R4000 superpipeline is implemented. As was discussed earlier, compared to the original R3000 pipeline, the optimized R3000 pipeline has most of its stages executing in half a cycle. The IC and DC stages are sped up by incorporating on-chip caches, and the register read and write are sped up by improved circuit design. The only stage still utilizing a full cycle is the ALU stage. To speed this up, a much larger and specialized adder is designed. Figure 5 shows the R4000 pipeline.

With this pipeline, it is possible to execute dependent ALU operations without stalls and to execute loads and stores at twice the rate of the superscalar machine. Because the ALU stage is now only half a cycle, this pipeline is about 10% faster than a superscalar implementation of the optimized R3000 pipeline.

6. Possible Future Improvements

The largest source of lost performance in all of the multiple instruction issue pipelines is the cost of taken branches. In the R4000 pipeline, it can cost as much as 20-30%. See figure 6.

Because the R4000 instruction cache is 64-bits

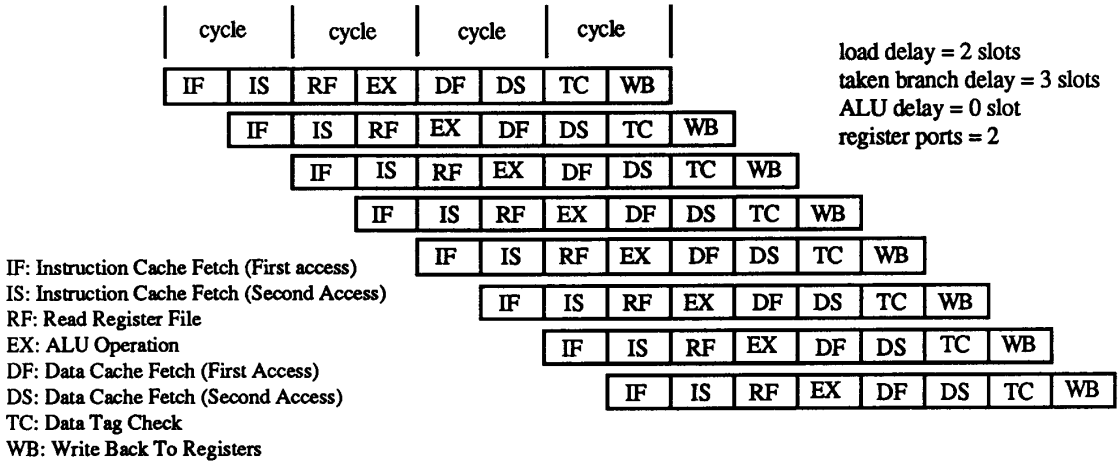

Figure 5: R4000 Pipeline

wide (to match the data cache) and it is pipelined, the instruction bandwidth is twice what is required. In the future, this bandwidth can be used to solve the taken branch penalty.

The extra instruction bandwidth can also be used to create a superscalar superpipelined processor. However, mainly vectorized floating-point code will be able to exploit this kind of instruction parallelism. Non-vector integer code will require other techniques for performance improvement.

7. Conclusion

The R4000 is a superpipelined architecture. A superpipeline design is chosen for several reasons. First, there are no instruction issue restrictions with the R4000 superpipeline, like there would have been in a superscalar implementation. Various combinations of independent instructions, including ALU/ALU and load/load can be executed without any pipeline stalls. This has not been demonstrated in any commercially available superscalar processor. The compiler is also simpler and instruction scheduling can be more efficient. Finally, there is no need to replicate functional units, as would be necessary in a superscalar implementation. This simpler logic allows for a quicker design and test cycle and allows space on the processor for future performance enhancements. Given the available process technology, a superpipelined implementation has the greatest benefits.

Acknowledgements

We thank Peter Davies, Dan Freitas, Ed Hudson, Mark Johnson, Earl Killian, John Kinsel, Paul Ries, Tom Riordan and Tom Vo for defining the R4000 pipeline. In addition, the efforts of the MIPS R4000 engineering staffs make the concept a reality. Also, thanks are given to all the reviewers in the company for their suggestions in improving this paper.

References:

[1] Jouppi, Norman P. and Wall, David W. "Available Instruction-Level Parallelism for Superscalar and Superpipelined Machines". ACM 1989.

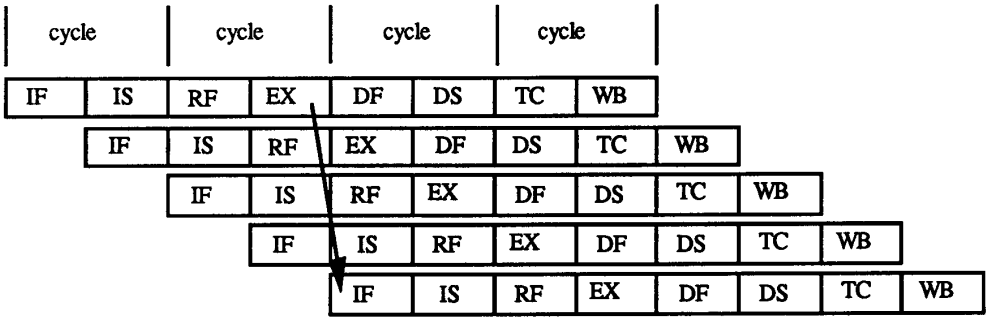

Figure 6: Branch Delay Slot of the R4000 Pipeline