

Slipstream Vs. DIVA

Version 0.1
December 12, 2002
By Aaron Kobayashi

1. Introduction

The microprocessor business is a very competitive one requiring constant innovation and self-reflection. In order for our company to be successful, we need to stand on the shoulders of those who have come before us. To do this, we need to be up to date with what research is currently being done in the area of microprocessors. In this paper we will be examining two recently discussed architectures—Diva and Slipstream. The goal of this paper is to compare and contrast these architectures from both a performance and practical standpoint while simultaneously gaining a rudimentary knowledge of the mechanisms that make each processor possible.

2. Design Goals and Motivations

Upon examination of these architectures, it was apparent that the design goals were different in many ways. The primary design goal of the Slipstream architecture is to increase performance of a processor by utilizing Chip Multiprocessor (CMP) or simultaneous multithreading (SMT). The design is fundamentally based on the assumption that programs contain non-essential code that can be safely (and speculatively) removed at runtime. A second processor then verifies the correctness of the first processor's assumption.

The DIVA architecture was designed to maximize reliability and minimize cost. The basic idea here is that full verification of a processor is both difficult and expensive. The cost of shipping a faulty design is even more expensive as demonstrated by Intel's FDIV bug. To combat this, the DIVA architecture implements a simple processor to check the correctness of the first processor. This reduces the verification time, as you do not need to fully verify the main processor and reduces the time to market while not significantly effecting performance.

3. Summary of the Slipstream architecture

The Slipstream architecture was designed around the idea that two processors working on the same instruction set in parallel can achieve greater performance than a single one can. The name is derived from the car-racing move of the same name where two cars follow each other very closely to enable both to go faster.

3.1 Conceptual Overview

The Slipstream architecture uses two execution units (either CMP or SMT) to achieve a higher IPC than a single execution unit of the same rating. The first unit dynamically skips computation of code it speculates to be non-essential to program flow to increase the overall IPC. This speculative execution stream is known as the advanced stream (A-stream). While the A-stream can execute code faster than a single equivalent processor, it is still speculative. The second execution unit known as the redundant stream (R-stream) runs the same program slightly behind the A-stream, but non-speculatively. It is the R-Stream's job to accept all control and data flow outcomes of the A-stream and check them against its own non-speculative version. If the two states do not agree, it is the R-stream's job to selectively repair the A-stream's corrupted architectural state.

3.2 The A-Stream and Instruction Removal

The A-Stream is able to speculatively remove instructions from the program flow by employing an instruction-removal predictor (IR-predictor). The IR-predictor is a modified branch predictor that generates the PC of the next block of instructions to be fetched by the A-stream. The predictor obviously cannot randomly skip instructions and still be efficient. Instead it accepts data from an instruction removal detector (IR-detector). This detector monitors the R-stream constantly for instructions that could have been removed from the program. Whenever it detects that such a case happened, it informs the IR-Predictor who modifies its tables to reflect the newfound data. Research has shown that just because a sequence of instructions happen once, does not mean it will happen exactly the same again. However research also indicates that if a segment is repeated, it is likely that it will keep repeating in the same fashion. These are the reasons that the IR-predictor employs a confidence factor when deciding if it is “safe” to skip instructions. When this confidence factor reaches a certain level, the predictor will skip execution of those blocks until the two execution units get out of sync.

In the unlikely event that the A-stream takes an incorrect direction, it has to be able to restore its architectural state to the correct path. It is the job of the recovery controller to maintain the addresses of memory locations that are potentially corrupted by the A-stream. Upon misdirection, the state of memory can be restored and all registers can be copied from the R-stream register set. This enables the A-stream to synch back up in the event of misdirection.

3.3 R-stream considerations

The R-stream is not nearly as complicated as the A-stream. You can think of the R-stream as a fast parallel checker for the A-stream. Probably the most common question that occurs pertains to the R-stream. Obviously the overall speed of the processor is limited to the speed of the R-stream. If the R-stream is not speculative, how is slipstreaming any faster than a single R-stream like processor? The answer is that because the A-stream is executing ahead and reporting results to the R-stream, it is able to fetch and execute more efficiently due to near perfect control and target predictions. As an added benefit, reliability against transient faults is increased due to the inherent redundancy of the slipstream architecture.

4. Summary of the DIVA architecture

The DIVA architecture is another design that attempts to utilize CMP or SMT technologies to create a “better” processor. Reliable operation is arguably the single most important attribute of any computer system and consequently this processor chooses to focus on reliability and verification problems rather than focus on the overall performance of the processor.

4.1 Conceptual Overview

The DIVA architecture uses two execution units for the purpose of increasing reliability and reducing verification time. DIVA implements dynamic verification to permit detection and recovery of all functional and electrical faults in the main processor core, both permanent and transient. It does this by using a deeply speculative DIVA core and a functionally and electrically robust DIVA checker. When instructions complete, their inputs and outputs are forwarded in program order to the DIVA checker whose job it is to verify the correctness of all computation and permit correct results to commit changes to the architectural state.

Additionally if an incorrect result is generated, the simple core will replace the result with the correct result and send that on to commit.

While this approach seems strange, there are many advantages to dynamic verification. This approach reduces the verification time of the processor thereby reducing its time to market. It also allows transistors outside the checker unit to scale smaller without fear of radiation interference. Thirdly it reduces the need to produce electrically robust designs empowering designers with the ability to tighten voltage and timing margins to create faster and cooler processor implementations. Finally it is possible to simplify the processor by eliminating infrequently used functionality of the core processor.

4.2 The DIVA Checker Architecture

When the DIVA core finishes execution of a process, it passes the results to the DIVA checker through two different pipelines. The computation pipeline verifies the integrity of all core processor computation and control. The communication pipeline ensures that register and memory communication occurred without error. When both pipelines report valid transfers, the checker can make sure that execution succeeded. Correctness is verified by checking for:

- Correct Computation – All operations produce the correct results given their inputs
- Correct Communication – The last write to memory is visible by the next read of that address.
- Correct Control – Processor control changes as defined by the semantics of the branch.
- Forward progress – The processor implements a watchdog timer to detect if the processor is not making forward progress. This can happen on occasion when the processor deadlocks for one of numerous reasons. If the watchdog times out the processor is reset, the checker core executes that instruction, and the core is set to execute the instruction following the deadlocked instruction.

If all of these conditions are met, the checker can safely retire its result.

Exceptions are also handled by the DIVA checker architecture. In the event that a fault is detected, the checker will correct the errant value/instruction, flush the pipeline, and restart the DIVA checker and processor core.

5. A Brief Summary of Experimental Results

5.1 The Slipstream architecture

The Slipstream architecture was tested through simulation both CMP and SMT implementations of the architecture. It was shown that the CMP performs on average 12% better in the slipstream form than as a single processor, however the results were varied, as applications like jpeg were not reduced due to the fact that it is already highly parallel. It is also noted that as the window size and issue bandwidth of the core is increased, the benefits of slipstreaming diminish rapidly.

5.2 The DIVA architecture

The DIVA architecture was tested through simulation. The processor implemented was a large window, 4 instruction wide processor with a 256 entry reorder buffer and a 64 entry load/store buffer. Testing showed that unless an extra memory port was added, structural hazards on the data memory became frequent and adversely affected the processor. When these hazards were

removed, the average slowdown dropped to below 1%. Exceptions were also shown to have little affect on the core processor even at very high exception rates. Ultimately the DIVA architecture is thought to be able to reduce the size and power requirements of the overall processor without sacrificing performance whoever a quantitative assessment is not made in this paper.

6. Comparison of Technology

6.1 Common Features

- Utilization of two processors to achieve the work of a single processor
- Utilize one processor to do work and one to check for valid progress

6.2 Differences

- DIVA focuses on reliability and cost, Slipstream on speed
- Slipstream assumes that both execution units are verified and are equivalent
- DIVA assumes that only the checker core is validated and does not need to be as powerful
- DIVA can slow a processor down by up to 3%, Slipstream can increase speed by an average of 12 %
- DIVA can be brought to the market very quickly and even released as beta hardware. The slipstream architecture will need to be fully verified before release.

7. Summary

While the microprocessor industry is currently in a very advanced field, many innovations continue to take place. Processors are starting to become specialized for their environment and if our company wants to compete, we will need to embrace this trend. The two processors discussed above demonstrate this difference as they utilize similar ideas to fill the needs of different markets. One is introduced as a very fast architecture while the other is shown to be faster to develop and more robust.

References:

- Z. Purser, K. Sundaramoorthy, and E. Rotenberg, "A Study of Slipstream Processors," *33rd International Symposium on Microarchitecture*, December 2000.
- T.M. Austin, "DIVA: A Dynamic Approach to Microprocessor Verification," *Journal of Instruction-Level Parallelism*, Vol. 2 (2000).