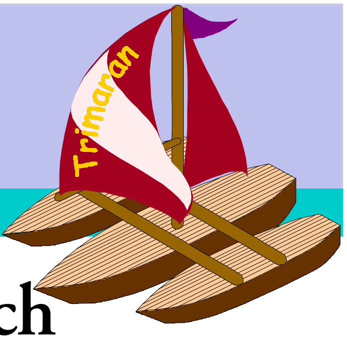
Trimaran Newsletter



Issue 1, Spring 2000

Related Links:

Trimaran Home

CAR at HP Labs

IMPACT at UIUC

ReaCT-ILP at NYU

Please send questions, comments and information for the newsletter to NewsEditor@Trimaran.org

The Parallelism Revealed

By Marc Saint, N.Y.U.

If electricity could travel at the speed of light, which it cannot quite do, then electricity would be able to travel a little under a foot in each clock cycle of a one gigahertz processor. In a complex processor, a lot has to happen in that clock cycle. Developing a processor that can actually perform complete operations in such a short time, or even on such a short path, is far beyond our current capabilities.

The situation is improved by pipelining within the processor, but once a processor has four or five pipeline stages, little is gained by further additions. On the other hand, we have greatly expanded chip capacity. The question then becomes: How can enhanced chip capacity be used to gain speed? The answer is that many operations can be performed at the same time by having a number of different pipelines assigned to different jobs (e.g.: branch units, arithmetic logic units and floating point units) and even multiple units of the same kind working simultaneously on different operations. Such capabilities have recently been incorporated into many processors in the most widely used processor families, such as Intel, MIPS and Motorola G3.

In the interest of backwards-compatibility, however, this parallel processing has been hidden from programs and compilers. The illusion of sequential processing is thereby maintained. Processor architecture in which operations are performed sequentially—even if this is illusory—are called Superscalar. Unfortunately, there is a substantial cost associated with the complex hardware needed to maintain the Superscalar illusion.

Increased cycle time is one of the costs of such complexity. Simplifying processor architecture and dropping the illusion of sequential processing, therefore, are ways of achieving speed gains, though there must be an attendant increase in the cleverness of the compiler. This model of the relationship between the compiler and its processor is called Explicitly Parallel Instruction Computing (EPIC). Trimaran is an infrastructure for supporting architecture and compiler research for a broad range of EPIC architectures.

Developing the sophisticated compilers required by

(continued on page 3, col. 1)

Research Around the World

By Rodric M. Rabbah, GA Tech.

Numerous statements have been made concerning the feasibility of programming multi-unit processors at the assembly language level, and many doubt that this is a reasonable undertaking. Consequently, this has become one of the most unexplored topics regarding Explicitly Parallel Instruction Computing (EPIC). These topics, however, form the foundation for the development of optimizing assemblers for EPIC processors at George Washington University. There, Trimaran has helped researchers explore methods to perform block formation, register allocation, instruction scheduling and code generation, using assembly language source code as input.

Trimaran has enabled such efforts worldwide, as numerous research groups have made significant usage of this compiler technology. Some have even contributed their internal Trimaran notes via the message forum, including a patch for compiling Trimaran on SOLARIS platforms. Such contributions are in keeping with the spirit of Trimaran, and we anticipate that they will continue.

For example, at the National University of Singapore, significant progress has been made towards a modified Trimaran infrastructure that will support an Itanium-like parametric processor configuration. This includes a reconfigurable cache simulator - developed in part at the ReaCT-ILP group at New York University - as well as IA-64 procedure call and register usage mechanisms.

The Predication Research Group, at UCSD, is working on extending the usefulness of predicated execution from both a compiler and architecture perspective. In an effort to enhance Trimaran's predicate-sensitive analysis tools, they implemented Predicated Static Single Assignment, which enables aggressive optimizations like predicate sensitive speculation and control height reduction.

At Rutgers, the PROLANGS group focuses on efficient, scalable analysis for C and Object Oriented languages (C++, Java), as well as optimizations of Java applications. There, Trimaran is used to investigate the degree to which Java's run-time exceptions hinder instruction scheduling. Using the back-end simulation and performance monitoring tools, new techniques may be found for allowing more efficient execution of Java programs

(continued on page 2, col. 1)

Related Links:

Trimaran Home

CAR at HP Labs

IMPACT at UIUC

ReaCT-ILP at NYU

Please send questions, comments and information for the newsletter to NewsEditor@Trimaran.org

Issue 1, Spring 2000

Research Around the World (cont. from front page)

containing run-time exceptions.

At Colorado State University, members of the Computer Science Department have been looking into ways of exploiting EPIC technology to benefit object-oriented programs. This research has primarily focused on virtual function calls, and how their costs can be reduced by use of EPIC features. A number of innovative ways have been proposed that transform the dynamic dispatch sequence of a virtual function call into a streamlined virtual function call, or into a statically-bound function call. These transformations rely on static analysis techniques such as type and constant propagation or class hierarchy analysis, along with front-end optimizations such as statically-binding virtual function calls that are monomorphic or with run-time class testing, and then using back-end optimizations that utilize control speculation, data speculation, predication and an exposed memory hierarchy. By use of the Trimaran research tool and analytical cost models, this research has successfully shown the quantitative benefits of applying EPIC technology towards object-oriented programs .

At the University of Delaware's Computer Architecture and Parallel System Laboratory (CAPSL) is looking to improve the performance of parallel and distributed systems and high performance compiler designs. There, Trimaran has been used to study the effects of the resource constraints on software pipelining and scheduling.

UCLA's Information and Computer Science Laboratory (ICSL) has used Trimaran's tool kit as part of a software-based solution to value-prediction. Value-prediction is a technique whereby the input to a control block is repeatedly monitored (along with the output) so that multiple visits to the block could be bypassed if incoming values reoccur regularly. In addition, Trimaran may play a role in their reconfigurable computing research, where a program is designed to tailor a specific machine's description according to the needs of a particular application's behavior.

The Reconfigurable Computer group at CMU has designed a pipelined, hardware-virtualizable, reconfigurable fabric called PipeRench. They are now exploring the design space for coupling such a fabric with a microprocessor. They believe that the machine description language, compiler back-end, and simulator tools in Trimaran should allow them to investigate a number of design points, and they hope to make Trimaran an integral part of the experimentation infrastructure.

The Architectures and Compilers for Embedded Systems (ACES) Laboratory is part of the Center for Embedded Computer Systems (CECS) at the University of California, Irvine. Research there is focused on developing state-of-the-art compilation and design space exploration (DSE) techniques for embedded systems with programmable components. Currently, this includes the development of an Architecture Description Language(ADL), called Expression, capable of specifying a wide variety of architectures (including RISC, DSP, VLIW, Superscalar) for the purpose of DSE and automatic software toolkit generation. They use Trimaran as a platform for DSE experiments on EPIC architectures, generating the MDES description from a high-level specification of the architecture in Expression. Trimaran also serves as an architecture evaluation tool. This approach has allowed for the evaluation of various techniques that enable rapid DSE of template-based processor systems.

If you would like to be included in our next issue of the Trimaran newsletter, please send a short paragraph describing your research areas and focus, including any publications and relevant web page information.

References and Abstracts (where available):

Arnold, M., M. Hsiao, U. Kremer, and B. Ryder. 1999. Instruction Scheduling in the Presence of Java's Runtime Exceptions. *Proceedings of the 12th International Workshop on Languages and Compilers for Parallel Computing (LCPC).*

Abstract: One of the challenges present to a Java compiler is Java's frequent use of runtime exceptions. These exceptions affect performance directly by requiring explicit checks, as well as indirectly by restricting code movement in order to satisfy Java's precise exception model. Instruction scheduling is one transformation, which is restricted by runtime exceptions since it relies heavily on reordering instructions to exploit maximum hardware performance. The goal of this study was to investigate the degree to which Java's runtime exceptions hinder instruction scheduling, and to find new techniques for allowing more efficient execution of Java programs containing runtime exceptions.

Carter, L., B. Simon, B. Calder, L. Carter, and J. Ferrante. 1999. Predicated Static Single Assignment. *International Conference on Parallel Architectures and Compilation Techniques (PACT).*

Abstract: Increases in instruction level parallelism are needed to exploit the potential parallelism available in future wide issue architectures. Predicated execution is an architectural mechanism that increases instruction level parallelism by removing branches and allowing simultaneous execution of multiple paths of control, only committing instructions from the correct path. In order for the compiler to expose predicated parallelism, traditional compiler data-flow analysis needs to be extended to predicated code. In this paper, we present Predicated Static Single Assignment (PSSA) to enable aggressive predicated optimization and instruction scheduling. PSSA provides efficient renaming and removes false dependencies in the presence of multiple control paths, which result from predication. We demonstrate the usefulness of PSSA for efficient Predicated Speculation and Control Height Reduction. These two predicated code optimizations are used during instruction scheduling to reduce the dependence length for the critical path through a predicated region.

Goldstein, S. C., H. Schmit, M. Moe, M. Budiu, S. Cadambi, R. R. Taylor, and R. Laufer. 1999. PipeRench: A Coprocessor for Streaming Multimedia Acceleration, *ISCA.*

Sadler, C., S. K. S. Gupta, and R. Bhatia. 1999. Applying Predication to Efficiently Handle Runtime Class Testing. *Submitted for publication.*

Abstract: Runtime class testing is a technique whereby virtual function calls are transformed into statically bound function calls through a series of conditional branches. Through this transformation, the overhead of virtual function calls can be significantly reduced. However, the drawback of these tests is that by relying on conditional branches, the amount of instruction-level parallelism (ILP) is limited and the mispredict penalties can be relatively high. We show that by using predication during class testing, these drawbacks can be eliminated, and the benefits of class testing can be improved upon. Predication is a supported feature in Explicitly Parallel Instruction Computing (EPIC) architecture, which converts control dependencies into data dependencies, and thus eliminates the mispredict penalties. With analytical cost models and experimental results, we show that predicated class testing can reduce the direct cost of virtual function calls by 43% on an average, which in contrast to other class testing techniques, is a performance improvement of up to 31% per function call. These results are based on architectural specifications that most modern architectures will exceed, and are considered conservative. The actual speedups resulting from predicated class testing are expected to be higher.

Sadler, C., S. K. S. Gupta, and R. Bhatia. 1999. Applying Predication to Reduce Cost of Virtual Function Calls. *ICCD Workshop on Hardware Support for Objects and Microarchitectures for Java.*

Abstract: The direct costs of virtual function calls in object-oriented programs is a runtime overhead incurred by the number of operations required to compute a target function address, and the time to perform these operations. We present a technique that uses an HPL-PD

(continued on page 3, col. 1)

Related Links:

Trimaran Home

CAR at HP Labs

IMPACT at UIUC

ReaCT-ILP at NYU

Please send questions, comments and information for the newsletter to NewsEditor@Trimaran.org

The Charted Course

By Marc Saint, N.Y.U.

HPL-PD, Intel IA-64 and other EPIC architectures envision exposing not only parallelism, but also the memory hierarchy as another important part of overcoming bottlenecks in processing speed. Accordingly, the next release of Trimaran will include a fully integrated programmable memory hierarchy simulator. The machine specification already in use by the compiler and simulator has been expanded to include a detailed specification of the memory hierarchy and its behavior. The memory simulator will produce detailed information on hits, misses and actual locations of requested data. It will also provide data on processor stalls from memory delays. This information and analyses can be used to investigate memory related compiler optimizations through profiling.

"This extension will make Trimaran an attractive tool for those researchers who want to prepare for IA-64"

On another development front, the ReaCT-ILP Group is now working closely with colleagues at the University of Singapore in order to produce a machine description and to extend Trimaran so as to permit accurate simulations of the Intel IA-64 processor and memory hierarchy architectures. This extension is expected to be available as a separate package close to the time of the next Trimaran release. This extension will make Trimaran an indispensable tool for those researchers who want to prepare for IA-64 development and assess future performance under IA-64.

Research References (cont. from page 2)

architectural feature known as predication to reduce the direct costs of virtual function calls. This technique is based on the possibility that the same virtual function table will be shared between virtual function calls, and whereby exploits this possibility by interleaving the function calls for objects whose type cannot be determined statically. With cost models we show that predication will eliminate redundant loads of the virtual function table from memory, and thereby reduce the impacts of memory latency on the overall runtime performance of virtual function calls.

Taylor, R. R. and S. C. Goldstein. 1999. A High-Performance Flexible Architecture for Cryptography. *CHES*.

Parallelism Revealed (cont. from front page)

EPIC architectures is a very difficult task. In EPIC architectures, the processor makes no scheduling decisions. It only follows the orders of the compiler. The compiler must therefore fully comprehend the data dependencies between operations in the program and be aware of the resources available in the explicitly parallel processor. It must negotiate these constraints and possibilities into a schedule of EPIC operations that is as efficient and parallel as possible. In other words, the compiler figures out ahead of time which operations can occur parallel to each other and instructs the processor accordingly.

In order for the compiler to manage such complex responsibilities, it must have complete information about the parallelism available in the processor. EPIC significantly differs from traditional Superscalar computing, therefore, in that instruction level parallelism (ILP) is completely exposed so as to enable the compiler to find the best ways to exploit the potential of the available functional units. The schedule worked out by the compiler can then be executed by the processor without concern for scheduling conflicts between units. The advantage of EPIC is that it allows for simpler, cleaner and thus faster hardware. It eliminates interlocking – where the hardware delays the execution of an operation until the inputs have been computed — and achieves greater parallelism than does on-the-fly scheduling within the processors of superscalar machines.

Currently, Trimaran includes an EPIC compiler which can compile ANSI C code for any processor that falls within the parameterized HPL-PD architecture class. Trimaran also includes a simulator and performance analysis tools. Groups are currently working on compiling for and simulating additional architectures, including Intel's IA-64 (see article *The Charted Course* on this page).

Trimaran was developed by a consortium of three groups: the compiler and architecture research (CAR) group at Hewlett Packard (HP) labs, the IMPACT Group at University of Illinois at Urbana-Champaign (UIUC) and the ReaCT-ILP Group at New York University.

The Crew

This newsletter was brought to you by the combined efforts of Marc Saint, Rodric M. Rabbah, and professor Benjamin Goldberg. We would also like to thank those who have contributed extensions to the Trimaran framework, as well as the many groups that have incorporated Trimaran into their research and contributed to the Trimaran user commnunity through comments, suggestions, and messages posted to the Trimaran forum.