06/18/03 05909 U.S. PTO

06-19-03

60479339 .061803 A/Prov

Attorney Docket No.: SRC028 PRO
Client/Matter No.: 80404.0033
Express Mail No.: EV335405698US

17612 U.S. PTO 60/479339 06/18/03

PROVISIONAL APPLICATION FOR PATENT COVER SHEET

This is a request for filing a PROVISIONAL APPLICATION FOR PATENT under 37 CFR 1.53(c)

PTO/SB/16 (10-01)

INVENTOR(S)

Given Name (first & middle, if any)	Family Name or Surname	Residence (City and State/Country)
Jon M.	Huppenthal	Colorado Springs, CO USA
Daniel	Poznanovic	Colorado Springs, CO USA
Jeffrey	Hammes	Colorado Springs, CO USA
Lisa	Krause	Minneapolis, MN USA
Jon	Steidel	Minneapolis, MN USA
David E.	Caliga	Colorado Springs, CO USA
Thomas R.	Seeman	Colorado Springs, CO USA
Lee A.	Burton	Divide, CO USA
Jeffrey Paul	Brooks	St. Louis. MN USA

☐ Additional inventors are named on the _ separately numbered sheets attached hereto

TITLE OF THE INVENTION (500 characters max)

BANDWIDTH EFFICIENCY AND UTILIZATION USING DIRECT EXECUTION LOGIC

CORRESPONDENCE ADDRESS

Direct all correspondence to:

☒ Customer Number **25235**

Type Customer Number here

Place Customer Number Bar Code Label Here

☐ Firm or Individual Name					
Address					
City		State		ZIP	
Country		Phone		Fax	

ENCLOSED APPLICATION PARTS (check all that apply)

☒ Specification *Number of Pages:* ***23***

☐ Drawing(s) *Number of Sheets:*

☐ Application Data Sheet. See 37 CFR 1.76

☐ CD(s), Number:

☐ Other (specify)

METHOD OF PAYMENT OF FILINGS FEES FOR THIS PROVISIONAL APPLICATION (check one)

☐ Applicant claims small entity status. See 37 CFR 1.27

☒ A check or money order is enclosed to cover the filing fees

☒ The Commissioner is hereby authorized to charge any additional filing fees or credit any overpayment to Deposit Account No. 50-1123

FILING FEE AMOUNT ($) **$160.00**

The invention was made by an agency of the U.S. Government or under a contract with an agency of the U.S. Government.

☒ No. ☐ Yes, the name of the agency and contract no. are

Respectfully submitted,

[illegible]

Name: William J. Kubida

Telephone: 719-448-5909

Date: 18 June 2003

Reg. No. 29,664

Docket No. SRC028 PRO

\\\CS - 80404/0001 - 61483 v1

60479339 .061803

Attorney Docket No. SRC028 PRO
Client/Matter No. 80404.0033
Express Mail Label No. EV335405698US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Provisional Application of:

Jon M. Huppenthal, Daniel Poznanovic, Jeffrey Hammes, Lisa Krause, Jon Steidel, David E. Caliga, Thomas R. Seeman, Lee A. Burton, and Jeffrey Paul Brooks

Serial No. -----------

Filed: Herewith

For: BANDWIDTH EFFICIENCY AND UTILIZATION USING DIRECT EXECUTION LOGIC

CERTIFICATE OF MAILING BY EXPRESS MAIL

MAIL STOP PROVISIONAL PATENT APPLICATION
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

The undersigned hereby certifies that the attached:

1. Provisional Application for Patent Cover Sheet;
2. Specification- 23 pages;
3. Check in the Amount of $160.00;
4. Certificate of Mailing by Express Mail; and
5. Return Card

relating to the above application, were deposited as "Express Mail," Mailing Label No.EV335405698US, with the United States Postal Service, addressed to MAIL STOP PROVISIONAL PATENT APPLICATION, Commissioner for Patents, P.O. Box 1450 Alexandria, VA 22313-1450, on 18 June, 2003.

18 June 2003
Date

[illegible]
Mailer

18 June 2003
Date

[illegible]
William J. Kubida, Reg. No. 29,664
HOGAN & HARTSON LLP
One Tabor Center
1200 17th Street, Suite 1500
Denver, Colorado 80202
(719) 448-5909 Tel
(303) 899-7333 Fax

\\\CS - 80404/0001 - 61483 v1

Bandwidth Efficiency and Utilization Using Direct Execution Logic

SRC Computers, Inc.

Introduction

In a previous paper, *Petaflop Computing Using Direct Execution Logic*, we showed that Teraflop-class processors will be available to direct execution logic (DEL) programmers in 2008. The challenge with this kind of computational capacity, of course, is how to keep it fed with operands. One key facet of this is the ability to exploit all available locality in a program or algorithm, or *bandwidth efficiency*. A second facet of this is the ability to run calculations at such a rate as to use all available memory bandwidth, or *bandwidth utilization.*

In this paper, we will show that the implicit features in modern microprocessor-based systems fail to achieve full bandwidth efficiency *or* utilization, even for fully optimized and tuned codes. By contrast, we will show that a programmer can use a high level language to directly manipulate the memory hierarchy and calculation pipeline via DEL achieving *both* maximum bandwidth efficiency *and* utilization across a wide variety of algorithms. The end result is a programmable, computational system that delivers the *maximum possible performance* for any given memory bus speed. This combination of efficiency and utilization yields an *orders of magnitude* performance benefit *even when utilizing an equivalent memory bus.* A diverse set of examples presented in this paper demonstrate between 6X and 216X performance gain through DEL.

The ability of DEL to use 100% of available bandwidth with near 100% efficiency is exciting today, but becomes even more exciting in the future as new interconnect technologies such as *stacked die* promise to substantially increase available memory bandwidth. We believe that performance via DEL will continue to scale perfectly into these bandwidth-rich environments.

Microprocessors: Reaching the Limits of Implicit Methods

Over the past 30 years, microprocessors have enjoyed annual performance improvements of about 50%. Designers have been taking advantage of higher frequencies, and have added a great deal of complexity primarily in two areas- memory hierarchies and searching for instruction level parallelism (ILP). For a short history of the microprocessor and these implicit design optimizations, see the Appendix.

Instruction Level Parallelism

Techniques, which automatically exploit ILP at execution time, have reached the point of diminishing returns. Of the 10 million transistors in an Intel Pentium III processor, for example, *more than half* are dedicated to instruction cache, branch prediction, out-of-order execution and superscalar logic. In addition, complex ILP techniques require chip-wide wires. In a few short years, it will take several clocks to travel across a microprocessor die so these complex techniques do not scale with integrated circuit technology. It appears that viable techniques to extract implicit parallelism from a serial instruction stream have simply run out of steam. According to Bill Dally, Pat Hanrahan and Ron Fedkiw at Stanford University:

The performance of the microprocessors ... is no longer scaling at the historic rate of 50% per year. Microprocessors have reached a point of diminishing returns in terms of gates per clock and clocks per instruction. As we enter an era of billion transistor chips, there is not enough explicit parallelism in conventional programs to efficiently use these resources...It is expected that without new innovations in parallel processor designs, microprocessor performance will only increase with the increase in gate speed, at a rate of about 20% per year. Such as change would have a major effect on the computer business, and the entire economy.[1]

Attaining high performance on microprocessors is beginning to require more explicit techniques from the compiler and programmer. For example, on the Intel Itanium processor, the compiler must carefully arrange instructions to attain 6-way instruction issue. Future microprocessor chips will have multiple processors on the same die and programmers and compilers will have to modify coding techniques to take advantage of parallelism at this level.

Attaining high performance using direct execution logic is much more explicit. A programmer and compiler explicitly identify parallelism and the compiler builds the necessary parallel logic pipelines. This technique has proven to be very effective. We have seen simple examples constructed in DEL where many *hundreds* of operations are executed in parallel each clock cycle. The available parallelism via DEL will increase linearly with integrated circuit density and so the technique will continue to scale.

DEL also allows the programmer to explicitly manipulate the *memory hierarchy*, avoiding the tradeoffs and complexities that are common in cache designs today. This allows the programmer to make the absolute most efficient use of on-chip storage and fully exploit available algorithm locality. In the next two sections, we will compare the implicit memory hierarchies found in microprocessors with the explicit memory hierarchy available to programmers through DEL.

Data and Instruction Cache

The design goal of any cache memory system is to minimize memory bus transactions by taking advantage of application locality. Microprocessors are fixed logic devices. As such, decisions have to be made at design time that optimize for the "average" case, if such a thing exists. Of course, the corollary of this is that the design will not be optimal for most applications.

Typical microprocessors exploit data locality implicitly via a rather complex data cache structure fraught with tradeoffs. Some of the design parameters include cache size (which has an inverse relationship to cache latency), the number of cache levels, cache-line width, the associativity and replacement policy, the write policy, the number of physical registers and the instruction cache design. A cache hierarchy that is optimized for the SPEC benchmark, for example, may not be a good choice for database applications or sparse matrix calculations. No cache design works well with irregular access patterns and strided memory references. For a full discussion of cache design tradeoffs, see Appendix B.

Because there are so many design choices when constructing caches, it is difficult to write portable, optimal code that will be optimized across cache designs. An example of an attempt in this regard is the Automatically Tuned Linear Algebra Software (ATLAS) project. This software configures its blocking strategy based on what it can learn about the underlying cache hierarchy of a processor.

Just as the limits of effective ILP are being reached, the limits of implicit data locality are also being reached. We are seeing more explicit data motion constructs creeping into microprocessor instruction sets in the form of pre-fetch instructions and cache "hints". In the next section, we will see that the tradeoffs made in today's implicit devices can become programming choices in a DEL environment, essentially creating optimized hardware tuned for individual programs.

[1] B. Dally, P. Hanrahan, R. Fedkiw, *A Streaming Supercomputer*, September 18, 2001.

Memory Hierarchies in a DEL Environment

In this section, we'll look at the explicit memory management constructs available in SRC's DEL environment. These constructs allow a user to essentially instantiate locality when and where it is needed within an algorithm. Unlike conventional data caches, these mechanisms are parallel and extensible in nature, with no fixed bandwidth limitation or sequential access restrictions.

The SRC-6E is a general purpose computing system with reconfigurable processing elements, called MAP processors. The SRC compilation system is designed to turn user codes and algorithms into direct execution logic within the MAP processor. The memory hierarchy on the SRC-6E processor is described below:

1. System Memory (DDR DRAM): The MAP processor can access memory that is shared with the microprocessor.

2. On-board memory (SRAM): The MAP processor is configured with 6 banks of SRAM memory, each capable of delivering one word per clock cycle to the MAP (4.8 GB/sec). Each of the 6 banks is 4 Mbytes in size for a total of 24 Mbytes of storage. This memory can be randomly addressed with gather and scatter type operations.

3. On-chip Block RAM (BRAM). The MAP processor has 288 sections of Block RAM memory. Each section is 36 bits wide and 512 elements deep and can deliver two words per cycle. When configured to deliver 64-bit operands, the 288 banks of Block RAM can deliver up to 288 64-bit operands per cycle (210.4 GB/sec). Note that this memory can be randomly addressed.

4. Register Storage: If the Configurable Logic Blocks (CLB) within the MAP are configured for storage, 1000s of registers can be configured. These can be used to pass operands from one computational unit to the next, or to construct other storage elements such as FIFO queues and look-up tables. Registers are created as needed on the SRC MAP processor and available bandwidth in this construct can reach into the 100s of GB/sec.

On the SRC MAP processor, memory is managed explicitly using constructs in either C or Fortran. The following table shows how this is expressed in these high level languages and how these constructs are built in hardware:

Table 1: Memory hierarchy programming constructs on MAP.

Storage Construct	*Source Code Construct*	*Built from*
System Memory	Pre-fetch subroutine or function call.	DDR DRAM chips.
On-Board Memory	COMMON blocks in FORTRAN Static Structures in C	On-Board SRAM chips.
Addressable on-chip Memory	Local Arrays within DEL routine	On-Chip BRAM
FIFO Queues. Etc.	Called as subroutine or function within DEL routine.	On-Chip CLBs or On-Chip BRAM
Registers	Scalar Variables and internally used by hardware macros.	On-Chip CLBs.
Wires	Scalar Variables and internally used by hardware macros.	On-Chip routing resources

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | sales@docketalarm.com | 1-866-77-FASTCASE

fastcase® Smarter legal research.