Advanced Processing Techniques Using the Intrinsity™ FastMATH™ Processor

Originally Presented to the Embedded Processor Forum; May 1, 2002

Tim Olson, Senior Design Architect

Note: All information contained in this document that relates to the FastMATH processor also applies to the FastMATH-LP™ processor.

1. Abstract

The Intrinsity™ FastMATH™ processor combines a MIPS32™ core, a matrix/vector engine, and dual RapidIO™ interfaces for a 2 GHz adaptive signal processor™ product capable of up to 64 giga operations per second (GOPS) (peak) computing performance. (See Section 8 on page 7 for more information on performance.) It is an embedded microprocessor designed for areas such as adaptive signal processing, image analysis, and other fields that benefit from flexibility, speed, and an architecture optimized for mathematical computation. With its 4 × 4 matrix of processing elements, it can compete favorably with most custom hardware designs, but uses a standard software programming model. This enables field enhancements without having to change hardware. This document describes the architecture, what it can do, and techniques for using it effectively. Samples are available now.

2. Introduction

The FastMATH processor is a fixed-point embedded processor that delivers application-specific integratec circuit (ASIC) level performance for real-time adaptive signal processing and other vector and matrix math intensive problems. It combines multi-gigahertz clock speeds with native matrix data operations and high-speed I/O for efficient partitioning of complex problems across multiple units. With its clock speed, plus its strong architectural advantages, it delivers ASIC-level performance with the flexibility of a software design.

The FastMATH processor is a product of Intrinsity, Inc., a fabless semiconductor company located in Austin, Texas. Intrinsity's patented $Fast_{14}$™ Technology (14 is the atomic number of silicon) is a design methodology that delivers three to five times the speed of standard design approaches in any given process.

The design goals of the FastMATH processor were to architect a processor optimized for matrix and parallel vector algorithms, such as those encountered in real-time adaptive signal processing. Typically, the inputs consist of arrays of data under conditions that may be rapidly changing. In today's and tomorrow's systems, this means increased use of adaptive algorithms, such as least mean squares (LMS), that have increased computation and bandwidth requirements because they require the calculation of new coefficient values from previous data. However, it also means being able to change to new, more efficient, or better algorithms as they are discovered, and to change to accommodate new standards as they are developed. Intrinsity extends the usual single instruction, multiple data (SIMD) vector architecture to the next dimension: SIMD full matrix operations. The high processor speed is balanced by fast I/O so that it can accept high data rates.

Industry needs argue for a standard programming model that allows functionality to change without modifying the hardware. This is addressed by basing the FastMATH processor around the industry-standard MIPS32 instruction set architecture (ISA), augmented with new matrix and vector instruction support through the MIPS® application-specific coprocessor 2 extensions. The design features both native matrix and parallel vector operations, an unusually

large 1-Mbyte L2 cache with a one-cycle full matrix load (usable on the same cycle), and 4 Gbyte/s I/O throughput (with the industry-leading RapidIO standard), thus ensuring scalability to multiple processors.

The success of this design can be measured by the FastMATH performance on core industry tasks such as fast Fourier transforms (FFTs). The 2-GHz clock speed and architecture advantages combine for a net greater than **5x** performance advantage on the industry-standard Telemark (telecommunications) benchmark suite from the EDN Embedded Microprocessor Consortium (EEMBC®), compared with published results of the best performing digital signal processors (DSPs) and embedded microprocessors. This performance has been certified by an independent testing laboratory and the results are publicly available on the EEMBC Web site (www.eembc.org).

3. Overview of the Adaptive Signal Processor Architecture

The architecture is diagrammed in Figure 1. The scalar core is the world's fastest MIPS processor, running at 2 GHz in a 0.13-micron process. It features a single-cycle ALU with a 12-stage pipeline (with an early-completion capability) that performs dual dispatch of scalar and matrix instructions in a single instruction stream from a 16-Kbyte instruction cache. Matrix and scalar instructions are executed in parallel. The processor core executes the scalar and matrix memory load/store instructions and performs overall control flow. This core is also a feature of Intrinsity's FastMIPS processor.

Using a MIPS core as the heart of the FastMATH processor gives it a standard programming model that is already supported by every major tools vendor. The MIPS core sends the matrix instructions in the instruction stream, along with scalar data from the MIPS core register file as needed, to the matrix engine. The instruction cache, a parallel 16-Kbyte L1 data cache for the MIPS core, and an integrated memory-management unit (with a fully-associative, 32-entry translation look-aside buffer (TLB)) all operate at 2 GHz.

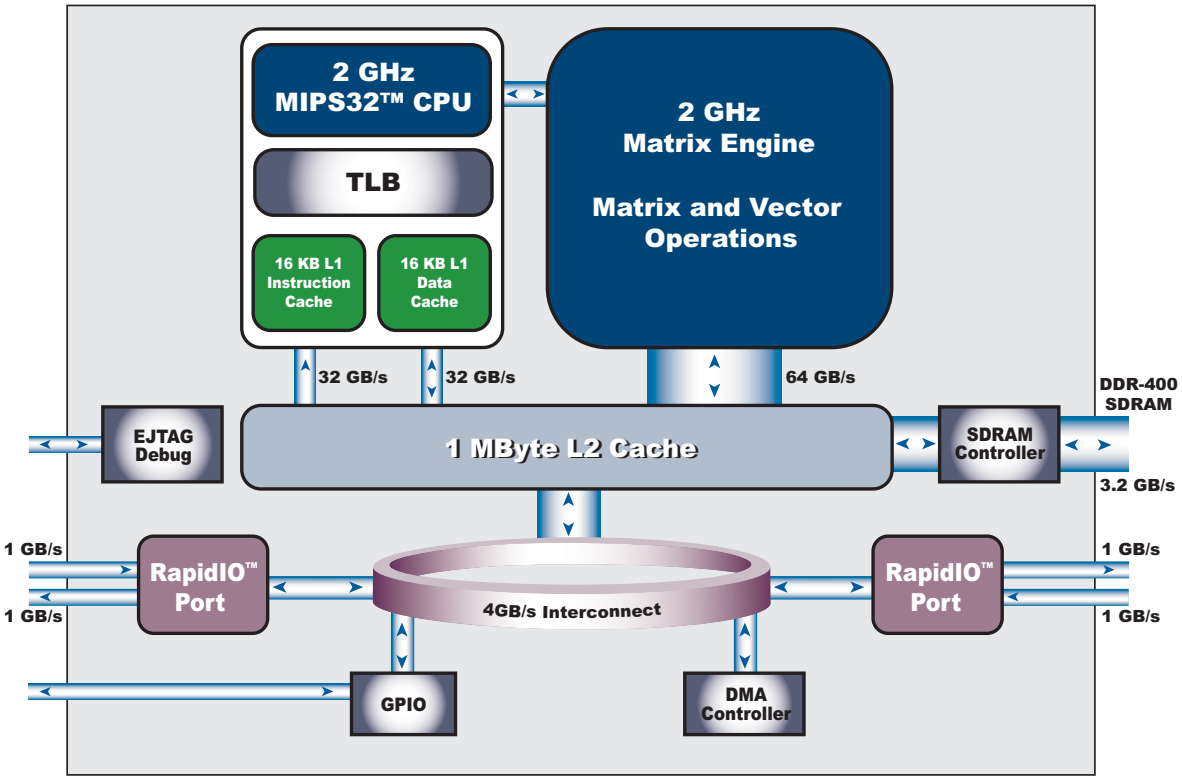

Figure 1: FastMATH processor block diagram

Like the scalar processor, the FastMATH parallel vector- and matrix-math engine also features 2 GHz clock speed. It consists of a 4 × 4 array of interconnected 32-bit ALUs, each with 16 32-bit registers and 2 independent 40-bit multi-

ply-accumulate registers (Figure 2). From a programmer's perspective, the individual register files form a set of 16 matrix registers, each holding 64 bytes of data, arranged as a 4 × 4 matrix of 32-bit elements. Each individual processing element in the matrix engine synchronously executes the same matrix instruction in parallel. Most matrix instructions execute pipelined in a single cycle. This allows single-cycle updates of matrix coefficients, either from calculations or from external I/O. Four condition codes are available to enable conditional processing, such as selection, on an element-by-element basis. Thus, branch instructions in the MIPS core are avoided.

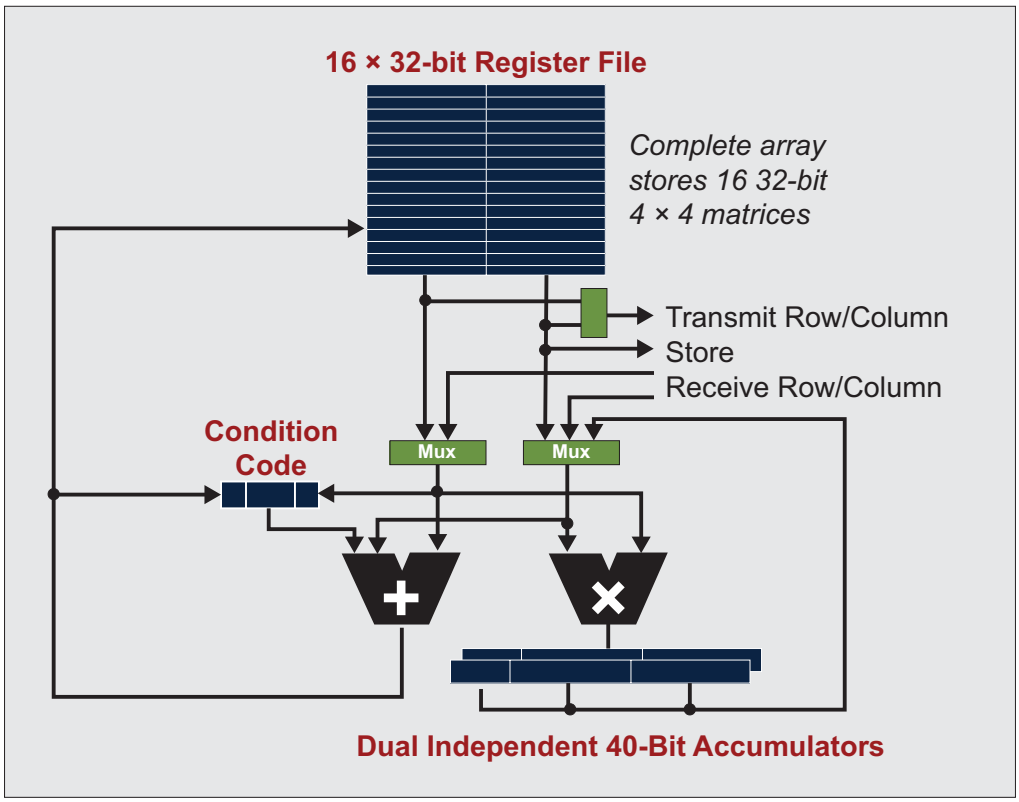

Figure 2: FastMATH processing element

In the matrix unit, the processing elements are connected by rows and by columns in a full mesh, unlike systolic arrays or other systems where the connections are to nearest neighbor only. This is a distinctive advantage of the FastMATH design. Each element can broadcast a value to all the other elements in its row or column and each element can use operands from local registers or from a broadcast during each operation. This arrangement optimizes matrix instruction execution.

The choice of a 4 × 4 arrangement of the processing units is a balance of the amount of parallelism needed in typical applications, the area required for computation versus L2 cache size, and the effect of the processing element interconnect on overall operation frequency. This arrangement gives the best of both worlds: a level of parallelism that fits target applications, combined with high operating frequency.

The 1-Mbyte L2 cache connects to both the FastMIPS and the FastMATH processors. It is a 16-bank, 4-way set-associative cache operating at 1 GHz, configurable as cache or static random-access memory (SRAM) in 256-Kbyte increments. It forms the coherency point for the system, maintaining data coherency across the MIPS core, the matrix engine, and the I/O subsystem. The link to the matrix engine is a 512-bit wide bus that can load or store an entire 64-byte matrix register in a single cycle. It supports up to 64 Gbyte/s in either direction (32 Gbyte/s sustained). The connection to the two MIPS unit L1 caches is also 512 bits wide. This unusually large L2 cache allows processing of large data sets directly in cache, without going to external memory.

Dual RapidIO ports provide high-bandwidth system I/O, ensuring scalability to multiprocessor operation. The RapidIO interface is a standard point-to-point fabric interface with broad industry support. With double-data-rate (DDR) clocking, it is capable of simultaneous 1 Gbyte/s input and output on each port – up to 4 Gbyte/s aggregate I/O. In addition to scalable performance, this provides glueless inter-processor connections with flexible system configurations.

An interconnect ring joins the RapidIO ports and a general purpose I/O (GPIO) port capable of 8- or 32-bit transfer to ROM, flash memory, SRAM, or external logic at bandwidths of over 200 Mbytes per second. A direct memory access (DMA) engine controls the I/O to reduce CPU overhead. The DMA is descriptor-based and features a two-channel RapidIO interface. In DMA operation, an on-chip memory controller can direct 64 bits of data on each clock edge in DDR-400 mode between the L2 cache and the synchronous dynamic random-access memory (SDRAM) for a sustained data rate of up to 3.2 Gbyte/s. It can address up to 1 Gbyte of memory and is capable of prefetching data from external SRAM memory directly into the L2 cache.

The roadmap to the future forecasts market-specific solutions and a next-generation 90 nanometer (nm) processor. The use of standard silicon technologies means that the Intrinsity dynamic logic advantage has a clear path to any direction the future provides.

4. Matrix Instructions

Matrix unit instructions fall into five broad classes. While every instruction can access the matrix register files, nearly every instruction can also access scalars from the MIPS unit or immediate constants, as needed.

The memory access class contains a load and a store instruction, each of which can move an entire 64-byte matrix (16 32-bit words) to or from one register of each processing element. This large data movement capability, coupled with parallel operations on all 16 words, make the FastMATH processor act effectively as a 512-bit machine.

The ALU/logical/comparison class can add or subtract an entire matrix, element by element, from one register to another. These instructions permit single-cycle operation at the 32-bit or packed 16-bit level. For complex arithmetic, this allows simultaneous operation on the real and imaginary parts. Comparison instructions can compare each element of a matrix in one register with a matrix in a different register or, of course, a scalar from the scalar unit or an immediate constant.

Multiply-accumulate instructions can perform element-by-element multiply and accumulate instructions on halfwords (16-bit values), operating on the upper or lower halfwords in any pairwise combination. Both signed and unsigned integer arithmetic is supported. The results are directed to one of the two accumulators associated with each processing element.

Interelement movement instructions use the mesh interconnections and can transpose a matrix stored in a register, perform shift operations, or block rearrangements of matrix elements across registers. Table lookups and row or column shifts are included in this class. Block rearrangement is a special feature of the FastMATH processor, unique among matrix engines, and will be described in more detail below. It can be used for matrix subblock creation and data interleaving, such as that encountered in communications applications.

Finally, an interelement computation instruction class can perform data movement followed by computation, pipeline fashion. It also includes matrix-matrix multiplications by halfwords and sums of rows or columns, as for dot products.

5. A Unique Feature of the FastMATH Instruction Set: Block Matrix Manipulation

Large matrix multiplication problems can be broken down into submatrix multiplications and the results combined when done. The problem is broken down into 4 × 4 submatrix problems where the data are stored and manipulated in the FastMATH element registers. The FastMATH processor provides an intrinsic 4 × 4 matrix-multiply instruction as a basic building block.

In order to prepare the data for submatrix multiplication it must be read into the unit from memory, where it is stored in normal form (Figure 3, top). The load instruction will read 16 consecutive aligned elements (i.e., a 1 × 16 row vector slice of the large matrix) into one 4 × 4 matrix register. Then the register will not contain a 4 × 4 submatrix as required by the multiplication. With the FastMATH architecture, the processor can execute 4 load instructions, each loading 16 similarly aligned elements from consecutive rows into consecutive matrix registers (Figure 3, middle, the matrix registers labeled m0 through m3). Each row is offset by the stride of the full matrix – the number of columns. Then, a single `block4` instruction will rearrange the elements of the four registers so that the first 16 elements from the first load (i.e., the first row of the large matrix) are now distributed into the first rows of the four registers, the second row likewise, and so on (Figure 3, bottom).

Matrix multiplication of properly blocked 4 × 4 submatrices is then accomplished by a single instruction. Instructions for multiplication of all pairwise combinations of halfwords are supported. The entire execution – 128 multiply-accumulate operations done in 4 cycles – is used to compute 16 resultant matrix elements. This takes a total of 2 nanoseconds (ns).

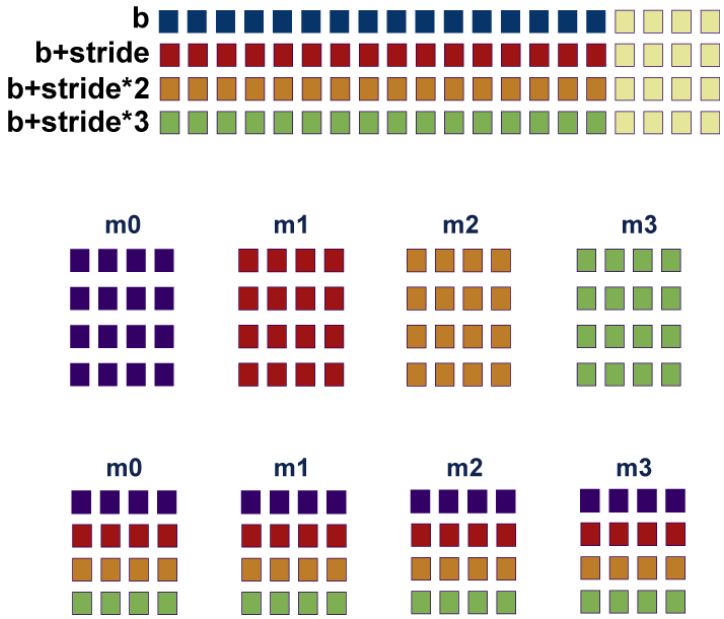

Figure 3: Loading and converting a large matrix into subblock format

6. An Example Application: Parallel User Interference Cancellation for a Wireless Base Station

In a code division multiple access (CDMA) wireless system, the users in a particular cell occupy the same frequency band and are differentiated by orthogonal error-correcting codes. Their signals may reach the base station via multiple paths with different delays and be subject to interference and other degradation. The principal source of interference is often the signals of the other users on the same frequencies whose codes may not completely cancel due to signal distortions and timing differences in the air link. Parallel user interference cancellation is a matrix-based, interference-reduction technique based on computing the cross-correlation between different users' codes with their different delays. This allows better subtraction of user co-interference. The heart of the procedure is a large matrix multiplication between a current decision matrix and the user cross-correlation matrix. The decision matrix is updated iteratively.

User interference reduction is extremely important in CDMA systems because it increases CDMA customer satisfaction and, at the same time, enables an operator to serve twice as many users with a single basestation. Then, operators do not need to install as many base stations, reducing their costs and their impact on the environment. This type of

very large, adaptive signal processing, linear algebra problem is well-suited to the FastMATH processor. Data from the antenna system enters via a RapidIO interface to the L2 cache. The DMA prefetch facility loads corresponding portions of the cross-correlation matrix, also into the L2 cache. The matrix unit performs the matrix multiplications via subblocks, as described above, and the results are stored and combined in the L2 cache. Without the high-speed I/O and the large L2 cache capable of concurrently holding the working set of data and results, these operations would be much more difficult.

A thorough investigation of parallel interference cancellation with the FastMATH processor has been done and the results are available in a white paper under nondisclosure agreement.

7. An Example Application: Decimation in Frequency FFT

Fast Fourier transforms (FFTs) are the basis of many signal-processing algorithms. The heart of the algorithm is the so-called “butterfly” operation that performs a complex addition, subtraction, and multiplication. Each FastMATH matrix register can hold 16 complex values with 16-bit real values in the upper halfwords of the matrix elements and the imaginary values in the lower halfwords. With the unique FastMATH matrix instruction set, all real and imaginary parts can be added or subtracted in matrix registers with one single-cycle instruction, a total of 32 halfword arithmetic operations per cycle, demonstrating its vector-math capability.

High and low halfword multiplications are used to form intermediate products and the processing element accumulators will accumulate the required sums and differences for the complex-valued multiplications. For example, the real and imaginary parts of the calculations are given by Eq.(1).

$$\begin{aligned} Re &= re1 \times re2 - im1 \times im2 \\ Im &= re1 \times im2 + re2 \times im1 \end{aligned} \qquad \textit{Eq.(1)}$$

FastMATH packing instructions will pack the resultant real and imaginary parts back into the halfword packed matrix format with appropriate scaling.

The final stages of an FFT require multiplications, additions, and subtractions among elements of the same 4×4 submatrix. This is done using row and column selection operations that create copies of a matrix register with swapped elements, followed by the multiplications and accumulations as described above.

The key to efficient implementation of an FFT is the parallel dispatch of scalar and matrix unit instructions. Due to the pipeline design of the matrix engine, software pipelining and scheduling are not necessarily required. For example, the result of a load instruction is available for use in the same cycle by a subsequent matrix instruction. However, even better operation can sometimes be achieved by careful attention to the detailed ordering of operations. Ideally, code is written such that instructions can be dispatched in pairs, one to the scalar unit and one to the matrix unit. Table 1 illustrates how the FastMATH processor requires only 13 cycles for implementation of the inner loop of an FFT.

Table 1: Example of FFT inner-loop code

Instruction		Cycle
load.m	m2, 0(r3)	0
srah.m	m2, m2	0
load.m	m3, 0(r4)	2
srah.m	m3, m3	2
fftloop:		
addh.m	m2, m2, m3	0
store.m	m2, 64(r3)	1
subfh.m.m	m2, m1, m0	2
addu	r3, r3, 64	2
load.m	m1, 0(r5)	3
mulhh.m	a0, m2, m1	3
msubll.m	a0, m2, m1	4
mulhhl.m	a1, m2, m1	5
maddhlh.m	a1, m2, m1	6
addu	r4, r4, 64	6
load.m	m2, 0(r3)	7
srah.m	m2, m2	7
mflo.m	m0, a0	8
addu	r5, r5, 64	8
load.m	m3, 0(r4)	9
srah.m	m3, m3	9
mflo.m	m1, a1	10
packhh.m	m0, m0, m1	11
blt	r3, r6, fftloop	11
store.m	m0, -6(r4)	12

8. ECL-Certified Performance Evaluations on the EEMBC Telemark Benchmark Suite

The performance of the FastMATH processor, as officially certified by the Embedded Microprocessor Benchmark Consortium (EEMBC) Certification Laboratories (ECL) on the EEMBC Telemark benchmark suite, was released at the 2002 Embedded Processor Forum in San Jose, California. ECL is an independent third-party laboratory formed to provide "...fair, impartial, unbiased benchmark score certification for EEMBC..." [1]

The EEMBC benchmark suites are designed to test performance on different workloads and capabilities in real-world problems for different market areas.

The Telemark suite is designed to test the type of high data rate math-intensive applications encountered in the communications industry. The overall benchmark result is a scaled geometric mean of the number of iterations per unit

time of test cases run on five algorithm classes: three 256-point fast Fourier transforms (FFTs); four Viterbi rate 1/2, constraint-length six, decoders; three autocorrelations of different lengths; three fixed-point bit allocations; and three convolutional encoders.

ECL provides two certifications. Most relevant for many customers is for code optimized to demonstrate the best possible performance of the processors under evaluation (the "full fury" code). The results are checked by ECL as part of the certification process.

The ECL-certified results, mesaured on silicon, are shown in Figure 4. The Intrinsity FastMATH architecture offers performance greater than **5x** better than the best of the embedded microprocessors and DSPs.

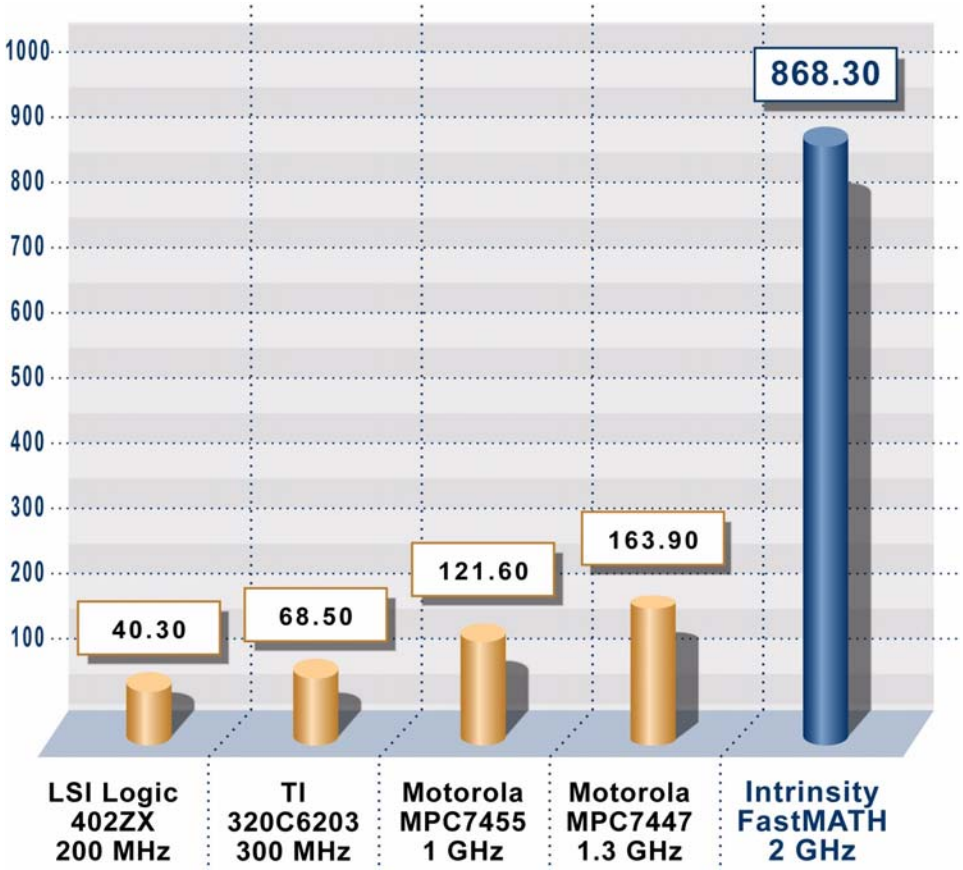

Figure 4: ECL-certified optimized Telemark benchmark suite performance

9. Summary

Intrinsity, Inc. has designed two 2 GHz processors for modern high-speed, high data rate applications. The FastMATH embedded processor has been optimized for parallel vector and matrix math, such as that demanded in real-time adaptive signal processing. The FastMIPS embedded scalar processor is also available as a stand-alone unit. The FastMIPS and FastMATH processors both provide MIPS32-compliant instructions. In the case of the FastMATH processor, the dual-issue instruction stream can process a scalar core instruction and a matrix instruction in the same cycle. The FastMATH adaptive signal processor product is capable of peak computation rates of 64 GOPS. In typical applications the average number of parallel operations per 1/2 ns cycle may be 30 or more. The MIPS ISA provides broad tool and system software support. Both processors feature high-performance I/O for scalability.

A software implementation of compute-intensive algorithms, especially those with high data rates, using the FastMATH or FastMIPS processors offers fast time-to-market, as well as the capability to field-change the algorithm

in response to changing standards and design improvements. As an industry-standard demonstration of the performance afforded by the high clock speed and the innovative architecture, the FastMATH processor delivers a more than **5x** speed improvement in the EEMBC Telemark benchmark suite, compared with the best published performance numbers for DSPs and embedded microprocessors. In addition, this speed advantage is offered in a standard MIPS programming model without the complexity of a custom processor requiring configuration.

The roadmap to the future takes advantage of the use of standard silicon technologies to implement Intrinsity's $Fast_{14}$™ Technology in the coming 90 nm semiconductor manufacturing process, providing continuing industry-leading performance in a programmable solution.

Samples are available today, with full production slated for late 2003.

10. Acronyms

ASIC	application-specific integrated circuit
CDMA	code division multiple access
DDR	double-data-rate
DMA	direct memory access
DSP	digital signal processor
EEMBC	EDN Embedded Microprocessor Benchmark Consortium
ECL	EDN Embedded Microprocessor Benchmark Consoritium Certification Laboratories
FFT	fast Fourier transform
GOPS	giga operations per second
GPIO	general purpose input/output
ISA	instruction set architecture
LMS	least mean squares
NDA	nondisclosure agreement
nm	nanometer
ns	nanosecond
SDRAM	synchronous dynamic random-access memory
SIMD	single instruction, multiple data
SRAM	static random-access memory
TLB	translation look-aside buffer

11. References

[1] Alan R. Weiss and Markus Levy, "ECL First Impressions Testing™: Issues and Answers for the Embedded Industry," an ECL White Paper. <http://www.eembc.org/Pdf/ecl%20white%20paper.PDF> (accessed May 8, 2002)

12. Revision History

Revision	Date	Brief Description
1.0	5/1/02	Presentation
1.1	5/8/02	EEMBC results
1.2	5/13/02	Expanded EEMBC results
1.3	3/26/03	New format and logo with tag line
1.4	7/14/03	Content changes, based on results with real silicon

Copyright © 2002-2003 Intrinsity, Inc. All Rights Reserved. Intrinsity, the Intrinsity logo, “the Faster processor company”, Adaptive Signal Processor, Fast14, FastMATH, and FastMATH-LP are trademarks/registered trademarks of Intrinsity, Inc., in the United States and/or other countires. RapidIO is a trademark of the RapidIO Trade Association. MIPS, MIPS32, and FastMIPS are trademarks/registered trademarks of MIPS Technologies, Inc. EEMBC and First Impressions Testing are trademarks/registered trademark of EDN Embedded Microprocessor Benchmark Consortium. All other trademarks/registered trademarks are the property of their respective owners.

INTRINSITY, INC. MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, AS TO THE ACCURACY OF COMPLETENESS OF INFORMATION CONTAINED IN THIS DOCUMENT. INTRINSITY RESERVES THE RIGHT TO MODIFY THE INFORMATION PROVIDED HEREIN OR THE PRODUCT DESCRIBED HEREIN OR TO HALT DEVELOPMENT OF SUCH PRODUCT WITHOUT NOTICE. RECIPIENT IS ADVISED TO CONTACT INTRINSITY, INC. REGARDING THE FINAL SPECIFICATIONS FOR THE PRODUCT DESCRIBED HEREIN BEFORE MAKING ANY EXPENDITURE IN RELIANCE ON THE INFORMATION CONTAINED IN THIS DOCUMENT.

No express or implied licenses are granted hereunder for the design, manufacture or dissemination of any information or technology described herein or the use of any trademarks used herein.

Any and all information, including technical data, computer software, documentation or other commercial materials contained in or delivered in conjunction with this document (collectively, “Technical Data”) were developed exclusively at private expense, and such Technical Data is made up entirely of commercial items and/or commercial computer software. Any and all Technical Data that may be delivered to the United States Government or any governmental agency or political subdivision of the United States Government (the “Government”) are delivered with restricted rights in accordance with Subpart 12.2 of the Federal Acquisition Regulation and Parts 227 and 252 of the Defense Federal Acquisition Regulation Supplement. The use of Technical Data is restricted in accordance with the terms set forth herein and the terms of any license agreement(s) and/or contract terms and conditions covering information containing Technical Data received between Intrinsity, Inc. or any third party and the Government, and the Government is granted no rights in the Technical Data except as may be provided expressly in such documents.