An Innovative High-Performance Architecture for Vector and Matrix Math Algorithms -

Presented at HPEC 2002; September 24, 2002; Poster A.2

Veeraragahavan Anantha, Ph.D.; Christophe Harlé, Ph.D.; Tim Olson; and George Yost, Ph.D.
Intrinsity, Inc.

1. Introduction

Intrinsity has designed an innovative embedded microprocessor for applications requiring very high performance, such as real-time signal processing tasks with large amounts of data. Traditionally, these applications have required hardware solutions such as ASICs, FPGAs, or DSPs. The Intrinsity FastMATH™ adaptive signal processor™ device, operates at 2 GHz clock speed and features an on-chip matrix co-processor for native matrix operations and single instruction, multiple data (SIMD) parallelization [1]. It is driven by a MIPS-based™ scalar engine that issues instructions to the matrix unit and executes MIPS32™ instructions in parallel. It is programmable in C or other languages. These processors feature fixed-point math. A one Mbyte data-coherent on-chip L2 cache and two independent bidirectional one gigabit per second RapidIO™ ports enable data transmission to keep pace with the processor speed.

We demonstrate the double advantages of the FastMATH architecture, plus its cycle speed, on several example algorithms. Poster session A.2 may be referred to for figures that illustrate the operation of the processor [2].

2. The FastMATH Processor

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. This implies use of adaptive algorithms that have increased computation and bandwidth requirements, compared with non-adaptive algorithms, because they require the calculation of new coefficient values from previous data. 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. We extend the usual SIMD vector architecture to the next logical dimension: SIMD full matrix operations. The high processor speed is balanced by fast I/O that can accept high data rates and support efficient multiprocessor configurations for scalability. I/O is facilitated by a descriptor-based DMA unit that can operate independently of the processors once the descriptors are loaded.

A MIPS® scalar core, which executes in parallel with the matrix unit, performs the scalar and matrix memory load-store instructions and performs overall control flow. It features a single-cycle ALU that performs dual dispatch of scalar and matrix instructions in a single instruction stream from a 16 Kbyte instruction cache. Because of the MIPS32 industry standard, the FastMATH processor may be programmed in the C language and industry-standard tool chains are applicable. For reference, a block diagram of the processor can be found on page 2 of the poster presentation [2].

The matrix unit consists of a 4×4 array of interconnected processing elements. Each element has two 40-bit multiply-accumulate registers (MACs). Each also has its own 16-entry, 32-bit register file. From the programmer's perspective, these elements act as a set of 16 independent matrix registers, each with one element attached to each processor. That is a total of 64 bytes of data arranged as a 4×4 matrix of 32-bit elements, which can be loaded by a single load instruction executed by the independent MIPS core. By the design of the pipeline, the load time can be completely hidden by matrix unit computation until the loaded values are required. The SIMD architecture can oper-

ate in parallel on all 16 words yielding, for example, a peak rate of 32 halfword additions per cycle, i.e., 64 giga operations per second (GOPS) at 2 GHz.

Numerous operations on the matrix registers at the single-bit, 8-bit, halfword, and word level are supported. Halfword summations and multiplications are convenient for 16-bit complex arithmetic: one can, for example, store the real parts in the upper halfwords and the imaginary parts in the lower halfwords. An advantage of the matrix architecture in comparison with vector architectures is that full single instruction, 4×4 matrix multiplications are supported by halfword (4 cycles execution time for 16 components in parallel). Any large matrix can be broken into 4×4 submatrices for multiplication by another matrix or a scalar with subsequent recombination.

The matrix unit supports a number of intrinsic instructions that operate on the matrix registers. Two entire matrix registers can be summed together at the packed 16-bit level, supporting 16-bit complex math with the real and imaginary parts stored by halfword or at the 32-bit level. Multiply-accumulate instructions operate at the halfword level, either element-by-element or, as stated above, as a matrix-matrix multiplication. Results accumulate in one of the MAC arrays. Block rearrangement of the matrix registers is also featured with a set of unique instructions that allow data interleaving, multiple data streams to be processed in parallel as 4×4 submatrices and other operations.

We demonstrate these features with a number of algorithms. All timing estimates are based on the Intrinsity cycle-accurate simulator and assume a cycle speed of 2 GHz.

3. Fast Fourier Transform

The Fast Fourier Transform (FFT) algorithm is well-known in the literature [3]. It is a key component of a large number of applications. For example, filtering tasks can often be done with two FFTs, an element-by-element multiplication of the resultant frequency coefficients, and an inverse FFT of the sequence of products. Many versions exist that are optimized for different situations. For an FFT with complex input data the FastMATH complex matrix and block rearrangement capabilities enable extremely efficient processing. These architectural advantages are approximately equal to the clock speed advantage for this algorithm, compared with competing solutions. For example, for 16-bit complex data a 1024-point radix-4 decimation-in-frequency FFT can be performed at the rate of 550,000 iterations per second. Thus, over 0.5×10^9 complex data points per second may be processed, a factor of five faster than the nearest programmable competitor [2]. An inverse FFT executes at the same speed.

The FFT is at the heart of orthogonal frequency-division multiplexing (OFDM). OFDM is a communications modulation technique in which blocks of symbols are divided among a number N of orthogonal carrier frequencies to be transmitted in parallel over a noisy channel. Multiple antennas may be used to achieve space diversity and for beamforming, as in the smart antenna application described below. After sampling and filtering, the data from each antenna are passed into an FFT block. The FFT block reconstitutes the symbol stream from each user at each antenna for input to the next stage, where the beamforming and/or symbol-rate processing is performed.

With this structure, we can estimate the fractional part of FastMATH computation necessary to keep pace with the input data rate in computing the FFT block. We assume that each user is able to utilize all N frequencies (in practice, this may or may not be achievable). If eight antennas are used, 10×10^6, 16-bit complex samples per second are collected from each, and a 1024-point radix-4 FFT is used, as described above, then only 14.4% of the processing capacity of a single FastMATH processor is required.

4. Smart Antenna Processing

A smart antenna array is a set of multiple antennas, with input processing such that desired signals are enhanced and interfering signals suppressed, for all signals of interest [4]. If the number of antennas is at least as great as the number of desired signals, then it is, in principle, possible to construct orthogonal beams for each signal. Such beams would have, for each signal, a maximum in the direction of the signal origination and nulls in every other direction. For communications and many other purposes this is rarely possible. Instead, one adopts one of a number of schemes that do the best possible job according to desired criteria.

One characteristic of smart antenna processing is that the signal originators may be mobile phones or other sources that move rapidly with time. Therefore, real-time adaptation is required. We have examined the capabilities of the FastMATH processor in one such scheme. After front-end processing that includes analog-digital (A/D) conversion, filtering, and possibly other processing, the stream of signals from each antenna is fed simultaneously to an adaptive weight-calculation block and a beamforming block. If blocks of M complex 16-bit samples $x(k)$, $k=0, \ldots, M-1$ from each antenna are taken, the weight-calculation step estimates an antenna-antenna covariance matrix R:

$$\hat{R} = \sum x(k)x^H(k) \quad . \qquad \textit{Eq.(1)}$$

Here, x^H is the Hermitian conjugate of the $N \times M$ matrix x. This matrix is inverted by Cholesky decomposition to estimate the weight-matrix

$$\hat{w} = \hat{R^{-1}} \sum d^*(k)x(k) \quad , \qquad \textit{Eq.(2)}$$

where d is a vector of reference signals appropriate for each input signal. For a CDMA communications application, for example, this might be the pilot signal. The beamformed output signal is then the matrix product of this estimated weight-matrix and the input data $x(k)$. The FastMATH solution can process 16 users in parallel, using the block-rearrangement instructions to put the data into matrix form so that the native matrix-matrix multiplication capabilities can be used.

For a WCDMA application example, we have estimated that 0.73 FastMATH processors can keep pace with the data from 64 voice users received on 16 antennas, with 4 rake fingers per user. The weights are updated every slot.

5. CDMA Multi-User Detection

This algorithm demonstrates the FastMATH processor's capability to distribute large computational and data-transfer workloads between multiple processors across the RapidIO interface. Multi-user detection is used in CDMA systems to mitigate interference [5]. Beginning with a stream of symbols from each user, an estimator $\hat{\hat{R}}$ for the user-user correlation matrix is constructed to express intersymbol interference. The input symbols actually observed are then a matrix product of $\hat{R}$ with the true symbols, for which we must then solve. This product is summed over a small number of symbol periods both before and following each desired symbol, because symbol periods for different users may overlap. In a WCDMA example we approach this problem by Jacobi iteration, which allows the solution to be broken down into partitions that may be distributed over multiple processors. In this way, the large computational workload is distributed efficiently. We calculate the correlation matrices $\hat{R}$ on-chip, taking advantage of the large coherent L2 cache in order to avoid all external memory references. The input data and partial calculation results are transferred between processors over the high-speed RapidIO interfaces. This data transfer takes place in parallel with the computation.

For the example of WCDMA a single processor can, in this manner, process up to 48 users, two processors up to 68 users, and four processors up to 134 users.

6. Conclusions

These examples demonstrate that high performance can be achieved on high data rate problems requiring real-time adaptation with a SIMD architecture that features intrinsic matrix math capabilities coupled with a high clock speed and high-speed I/O. This architecture enables efficient parallelization, as well as the matrix manipulation. For applications requiring multiple processors, RapidIO interconnection and a large L2 cache enable efficient operation without reference to external memory.

7. References

[1] Many additional details of the FastMATH processor may be found on the Intrinsity web site, www.intrinsity.com.
[2] "An Innovative High-Performance Architecture for Vector and Matrix Math Algorithms," Poster Session A.2, this conference.
[3] See, for example, H. K. Garg, *Digital Signal Processing Algorithms* (CRC Press, New York, 1998).
[4] See, for example, J. Litva and T. K-Y Lo, *Digital Beamforming in Wireless Communications* (Artech House, Boston, 1996).
[5] D. Koulakiotis, A.H. Aghvami, "Data Detection Techniques for DS/CDMA Mobile Systems: A Review", *IEEE Personal Communications*, June 2000.