Optimization of Vertical and Horizontal Beamforming Kernels on the PowerPC G4 Processor with AltiVec Technology

Young H. Cho, David Brunke, and Greg E. Allen
Applied Research Laboratories:
The University of Texas at Austin
Austin, TX 78758 U.S.A.
{young, dbrunke, gallen}@arlut.utexas.edu

Brian L. Evans
Embedded Signal Processing Laboratory
The University of Texas at Austin
Austin, TX 78712-1084 U.S.A.
bevans@ece.utexas.edu

Abstract

Three-dimensional real-time digital sonar beamforming requires 4 to 12 GFLOPS, 1 to 2 GB of memory, and about 100 MB/s of I/O bandwidth. Allen and Evans have implemented a 4-GFLOP sonar beamformer in real-time on a Sun UltraSPARC II server with 16 333-MHz processors by utilizing the Visual Instruction Set (VIS) single-instruction multiple-data (SIMD) extensions. In this paper, we rewrite the horizontal and vertical beamforming kernels to use AltiVec SIMD extension for the PowerPC. AltiVec can execute up to four 32-bit floating-point multiply and accumulate (MAC) operations per instruction. In the PowerPC implementation, we prefetch and realign data for the 128-bit SIMD registers of AltiVec. We evaluate the performance of these beamforming kernels on the PowerPC and the UltraSPARC-II to evaluate the impact of the compiler, SIMD word alignment, and cache block alignment on performance.

1.0 Introduction

Since 1992, the digital signal processor (DSP) market has grown at a rate of 40% per year. In response, designers of general-purpose processors began to embed signal processing architectures into their processing cores. These native signal processing (NSP) extensions enable integrated processing system solutions for multimedia and signal processing applications [1].

Real-time digital sonar beamforming can require several GFLOPS of computation and 50-200 MB/s of I/O. Beamformers were initially implemented in custom hardware (before 1990), then in commercial off-the-shelf components (1990s). Work is underway to implement these systems in commodity multiprocessor servers. One recent implementation by Allen and Evans [2] uses a commercial general-purpose 12 processor symmetric multiprocessor (SMP) workstation from Sun Microsystems.

Signal processing kernels can exploit data parallelism by using Single Instruction Multiple Data (SIMD) arithmetic operations, which are available in native signal processing extensions [2,6], such as the Visual Instruction Set (VIS) on the Sun UltraSPARC-II processor [4]. By using a Sun Ultra Enterprise server with 12 333-MHz UltraSPARC-II CPUs, a real-time beamformer delivering 4 GFLOPS on 160 MB/s of streaming data was realized using 1.2 GB of memory [2,6].

The goal of our research is to further explore the effectiveness of NSP extensions by optimizing and assessing the performance of beamforming kernels using AltiVec on the PowerPC [5]. We evaluate the performance of these beamforming kernels on the PowerPC and the UltraSPARC-II to evaluate the impact of the compiler, SIMD word alignment, and cache block alignment on performance.

2.0 Native Signal Processing Extensions

Many high performance embedded applications are programmed on systems with a few general-purpose processors as system controllers with a larger number (possibly hundreds) of specialized DSPs to perform scientific calculations. However, this type of system has many disadvantages such as different programming platforms and unequal performance advances in processor technologies. Since 1995, many manufacturers of high performance general-purpose processors have integrated native signal processing instructions onto their processor cores to offer integrated solutions that combine the functions of both DSP and general-purpose processors. Such processors usually have a common programming and executing environment allowing easier and faster software development.

2.1 UltraSPARC Visual Instruction Set

A SIMD architecture allows an instruction to be performed on multiple data, thereby potentially increasing the computation performance proportional to the size of the SIMD registers. However, most SIMD architectures require that the input data is aligned in the memory according to the size of the register. Therefore, if the data is not aligned in the memory, additional instructions may be needed for the data permutation to align the data.

The Visual Instruction Set (VIS) is a set of signal processing instructions based on a SIMD architecture. The floating-point data of the UltraSPARC processor core is enhanced with graphics integer units to support VIS. With 50 new CPU instructions, VIS can perform integer

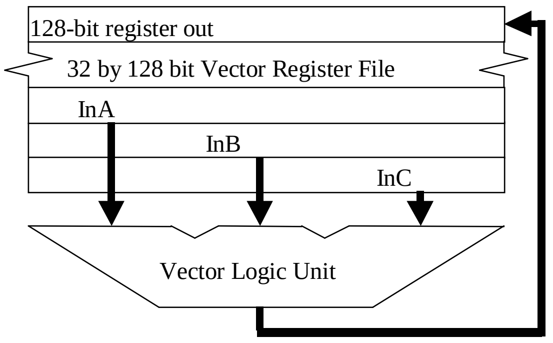


Fig. 1: Block Diagram of PowerPC AltiVec Unit

operations on multiple words with a single instruction. Thus, VIS can achieve up to four times speedup with 8-bit by 16-bit fixed-point multiplication using the SIMD arithmetic logic [4].

2.2 PowerPC AltiVec

The AltiVec vector unit is sectioned into a separate sub-unit of the processor as are with the floating-point and integer units. As shown in Fig. 1, the vector unit has its own 32 by 128-bit wide register file for use with 150 new floating point and integer SIMD instructions. It allows execution of up to four 32-bit floating point MAC operations per instruction [5]. AltiVec offers greater computing resources than VIS, and is potentially a much more powerful signal processing extension.

3.0 Beamforming Algorithm

Sonar beamformers use the output of an array of sensor elements to determine from which direction a sound is coming. We implement a time-domain beamforming algorithm which weights, appropriately delays and sums the outputs of the sensor array. The weighting of the sensor outputs helps to improve the spatial response [3].

The time delay resolution required is typically several times the Nyquist rate for preserving the signal frequency content. Instead of sampling at a higher rate, digital interpolation with Finite Impulse Response (FIR) filters to gives a satisfactory time delay resolution [3]. This is called digital interpolation beamforming, and is shown in Fig. 2. Analog data is sampled at just above the Nyquist rate and then interpolated, delayed, and summed.

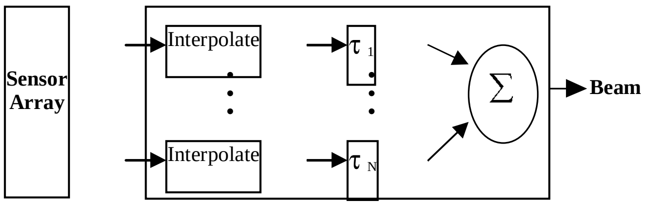


Fig. 2: Digital Interpolation Beamformer

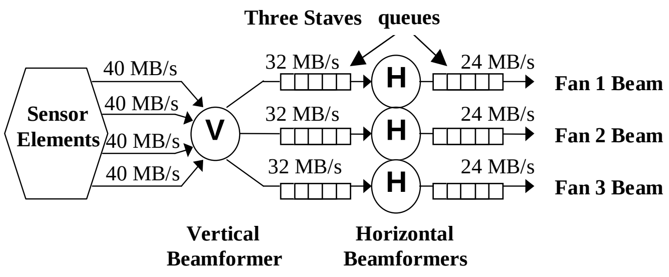


Figure 3: Block Diagram of the 3-D sonar beamformer

Our implementations decompose a 3-D beamformer into horizontal and vertical subsystems. This method of beamforming enables 3-D mapping of underwater surfaces [2]. The vertical beamformer computes three sets of vertical outputs called staves. Three dot products are computed with each column of 10 vertical transducers and three coefficient vectors as shown in Figs. 3 and 4. For improved performance, we use SIMD integer computation. The vertical beamformer also converts the data to floating-point format for the following horizontal stages.

Three identical horizontal beamformers process the three sets of stave results to calculate horizontal beams. The horizontal beamformers use digital interpolation beamforming. The interpolation in the horizontal beamforming kernel is simplified by using a two-point FIR filter, which gives enough accuracy for this system. The overall system description is shown in Fig. 3.

The system also leverages the Computational Process Network model [2]. The beamforming kernels are broken into many process nodes (threads) connected by queues. Since the application will run in real time, dynamically changing the queue sizes is not desirable. The queue sizes are set at initialization time to be large enough to never cause artificial deadlock from writing to a full queue.

4.0 AltiVec Implementation

Using recent PowerPC G4/7400 processors with AltiVec, we confirm the evaluations assessed with various PowerPC simulators [7]. We evaluate the results of the vertical and horizontal beamforming kernels programmed with AltiVec on the G4 processor in a real-time working context.

We implement the kernels in the Linux operating

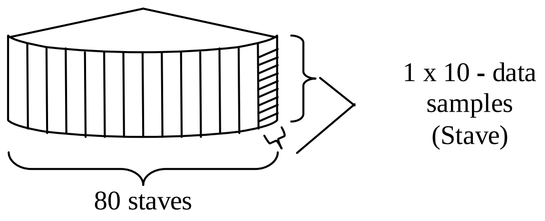


Figure 4: Sensor array model with 80 staves

system using the GCC compiler. An AltiVec enabling patch [8] provided by Motorola is applied to GNU C Compiler (GCC) to enable compilation of AltiVec instructions in C++. This simplifies development by allowing both the signal processing kernels and the supporting framework software to be written in the same language and environment.

Like the VIS extension in UltraSPARC, AltiVec is a SIMD extension in PowerPC. However, the programming approaches are very different due to different hardware design. One of the major differences is the width of the data that each processor uses in its extension instructions. For the most efficient usage, the input and output data used with AltiVec should be 128-bit aligned [5,8,9].

4.1 Process Network Programming Model

The inputs and the outputs of the beamforming kernels are queues. In the previous implementation, the queue was allocated contiguously with each set of stave samples addressed consecutively according to the time at which they were sampled. Since the two-point interpolation in the horizontal beamformer uses the samples collected from a stave two consecutive times, the original queue structure would require the sample points to be referenced with two different addresses distanced by the size of a stave. By transposing the queue, the samples for each stave become contiguous according to the sampling times. Such an arrangement would allow each vector load instruction to load up to four samples required for the calculation at once. We refer to the transposing of the queue as corner turning.

Even though the corner turning may benefit the vector load, it introduces a few problems. First, the index calculation becomes more complex, requiring the queue size to be added or subtracted to address the adjacent set of stave samples. Second, due to the necessary arrangement of the queue in memory, it is more difficult to implement dynamically growing queues. The effect of the first problem is inevitable without significant change in our data structure. However, the second problem disappears because we allocate the queue to be large enough to prevent artificial deadlock.

4.2 Vertical Beamforming Kernel

In the vertical beamformer, we need to compute three dot products on each set of ten 16-bit integer samples with three sets of coefficients. To avoid arithmetic overflow or underflow, the results are stored into 32-bit integers. To accommodate the horizontal beamforming kernel, the results are first converted to 32-bit floating-point numbers then corner-turned into the queue.

The input and output structure of the vertical beamforming kernel remains the same because of the way in which the hardware collects and stores the data into the queue. The vertical beamformer uses AltiVec instructions to load four 16-byte word aligned data from the queue at a time. The words are then transposed to perform dot products with three different sets of coefficients.

The resulting vectors are stored in strides to accomplish corner turning of the entire contents of the queue. The corner turned data in the queue is fed into the horizontal kernels to minimize data permutations required for the input vectors.

The vertical beamforming algorithm performs dot products requiring large amounts of sequential data in a short time. Therefore, every time the process attempts to access data that is not in the data cache, it is stalled until the block of the data is loaded into the cache. We use the AltiVec data pre-fetching instructions to reduce data cache misses.

4.3 Horizontal Beamforming Kernel

The horizontal beamformer loads the corner turned samples using the vector load operation. Although the number of load instructions is reduced due to the AltiVec vector operations, the data is permuted to meet the alignment constraints for the instructions before the computation.

As described earlier, the algorithm performs two-point interpolation with consecutive samples from each stave. Because the input to the horizontal kernel has been corner turned, consecutive samples in memory are consecutive samples of the same stave in time. As shown on Fig. 5, data blocks labeled 1 through 9 represent sequential samples collected from a given stave.

To reduce the number of load and the permute instructions, two sets of the same coefficients are stored, each of which are aligned differently by one. The vector multiply and accumulate instructions compute the result using two coefficient sets to yield the two-point

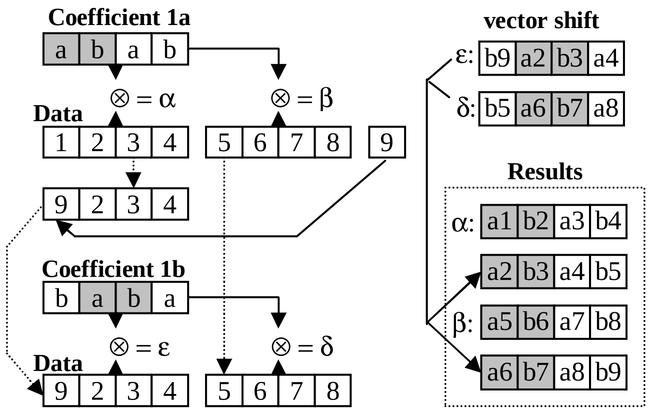
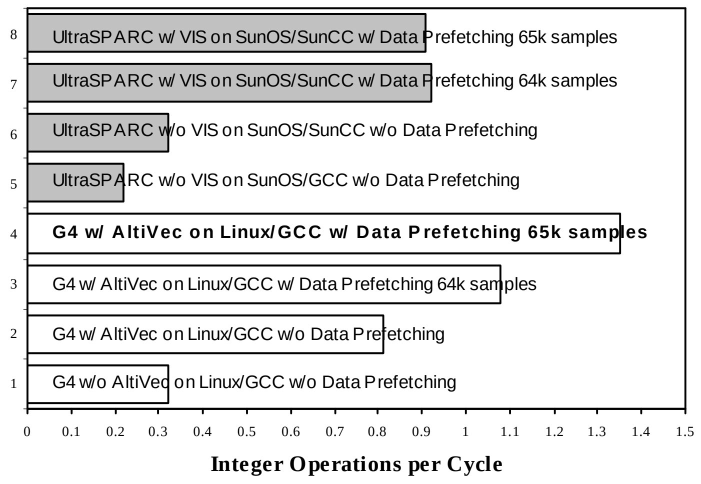
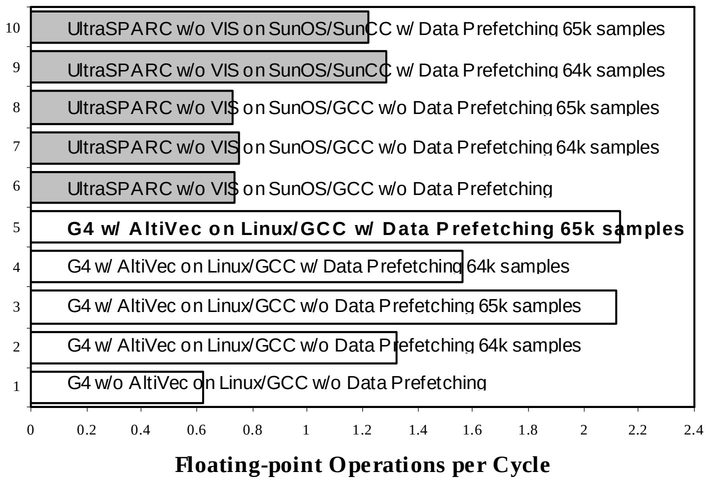


Figure 5: AltiVec Implementation of 2-point Interpolation



(a) Vertical Kernel Performance



(b) Horizontal Kernel Performance

Figure 6: Beamforming Kernel Performance Measurements

multiplication. Up to four floating-point multiply and accumulates are computed with each AltiVec instruction. These results are then summed and permuted to yield two-point interpolated beams.

By unrolling the instruction loops, more data independent instructions can be executed at the same time. Thus, the vector unit pipeline can be filled without having to wait for the expensive floating-point calculation to complete. The results are once again placed in a corner turned output queue of the horizontal beamformer.

5.0 Results and Analysis

Fig. 6 shows the performance measurements in operations per cycle for the new beamforming kernels written in AltiVec versus previous implementation in VIS for UltraSPARC. The results are organized according to the development platform such as the operating system and compiler, as well as the usage of the native signal processing extensions, pre-fetch instructions, and input data size.

For the horizontal kernel, performance is measured from several versions of loop-unrolled kernels. Due to different optimizations used in compilers and the processor architecture, the compiled kernels gave different optimal number of loop unrolling iterations for the generated code. Fig. 6b shows the performances of the versions with the best number of unrolled loops.

As performance results indicate, the compiler plays a large role in the performance. Unlike SunCC, GCC is a generic, non-commercial C compiler that does not consider many potential architectural advantages of each processor. Even with optimization and architectural tuning flags enabled, GCC compiled code performed as much as 50% slower than the SunCC compiled code on the same machine.

Adding VIS to the vertical kernel increased the performance by a factor of two to three, whereas adding AltiVec into the kernel increased the performance by a factor of four. Initially, adding the AltiVec extensions to the code doubled the performance of the kernel whereas the upper limit was four due to the SIMD architecture. This discouraging result led us to focus on the overhead from vector permutation. Altering the algorithm to reduce the vector instructions in the code did not significantly increase the performance, which indicated that the bottleneck of the vertical beamforming implementation was not the instruction count.

So, we turned our attention to the memory performance. Through several iterations of performance tests, we found that adding prefetch instructions increased the performance up to 1.4 times. We also unexpectedly discovered that by not allocating the queue lengths to be a multiple of 16 Kbytes, we increased the performance by 1.2 times. By examining the instructions generated from the code, we found that this alignment was most likely causing cache thrashing. With these adjustments, the best kernel with AltiVec executed over four times faster than the version without the AltiVec.

The horizontal kernel also executed approximately four times faster with the AltiVec extensions. A similar performance increase occurred when the input data queues sizes were not a multiple of 16 Kbytes. However, the data pre-fetching did not affect the performance significantly in the optimal case. This is likely due to our kernel implementation that emphasizes coefficient and sample data reuse, thereby reducing the total number of cache misses.

6.0 Conclusion

In this paper, we program horizontal and vertical beamforming kernels for a 3-D sonar system on the PowerPC G4 processor using GNU C compiler. We develop a version of each beamformer with and without the PowerPC AltiVec extensions. The vertical beamforming kernel requires rearrangement and cornerturning of the data for efficient use of the 128-bit long word SIMD units. For the horizontal beamforming kernel, a new data structure has to be integrated to minimize the number of permutations needed to align the data for more efficient use of the SIMD instructions.

We evaluate the performance of our beamforming kernels on the PowerPC G4 and previously developed beamforming kernels on the UltraSPARC-II

- with and without handcoded AltiVec extensions,
- with and without data prefetching, and
- with and without alignment of data blocks on cache boundaries.

Compiler optimization and SIMD word alignment improve performance on both processors. Better backend compilation methods optimized for the processor can make the same code execute two to three times faster. The longer the SIMD word, the more difficult it is to align data with a SIMD register. Because a SIMD word on the PowerPC is twice that of the UltraSPARC's SIMD word, the AltiVec implementation of the horizontal beamformer requires use of permute instructions to align the data.

For cache usage, aligning blocks of data along cache boundaries improved performance on the UltraSPARC but degraded performance on the PowerPC. The architectural difference in the caches can cause the PowerPC G4 cache to be used inefficiently when the code for the UltraSPARC is ported directly onto the G4. This was first observed in the vertical kernel implementation under PowerPC, as there was a 15 to 20% performance loss whenever the sample size was a multiple of 16 kbytes.

PowerPC with AltiVec extensions outperformed UltraSPARC with VIS despite the fact that the compiler optimization used for the PowerPC is relatively immature. In the vertical beamformer, the overall performance was approximately 1.56 times that of UltraSPARC whereas the horizontal beamformer was about 1.83 times.

The benchmark measurements of the current implementation indicates the possibility of running the 4-GFLOP beamforming system on six 450-MHz PowerPC G4 processors replacing our current system which requires ten 450-MHz UltraSPARC II processors. However, our ultimate goal is to implement the beamforming algorithm on one quad-PowerPC G4 SMP system. By optimizing the code with precise examination of the low-level instruction trace, as it has been done for the UltraSPARC implementation, we predict a successful implementation in the near future.

7.0 References

[1] J. Bier, "DSP on General Purpose Processors," *MicroDesign Resources Dinner Meeting Slides, Berkeley Design Technology, Inc.*, Jan. 1997.

[2] G. E. Allen and B. L. Evans, "Real-Time Sonar Beamforming on Workstations Using Process Networks and POSIX Threads," *IEEE Trans. on Signal Processing*, vol. 48, no. 3, pp. 921-926, March 2000.

[3] R. G. Pridham and R. A. Mucci, "A Novel Approach to Digital Beamforming," *Journal of Acoustical Society of America*, vol. 63, no. 2, pp. 425-434, Feb. 1978.

[4] Sun Microsystems, "VIS Instruction Set User's Manual," *http://solutions.sun.com/embedded/databook/pdf/manuals/805-1394-01.pdf.*

[5] AltiVec Programming Environment/Interface Manual, *Motorola Inc.*, 1998.

[6] G. E. Allen, B. L. Evans, and L. K. John, "Real-Time High-Throughput Sonar Beamforming Kernels Using Native Signal Processing and Memory Latency Hiding Techniques," *Proc. IEEE Asilomar Conf. on Signals, Systems, and Computers*, Oct. 25-28, 1999, vol. I, pp. 137-141, Pacific Grove, CA.

[7] H. Nguyen and L. K. John, "Exploiting SIMD parallelism in DSP and multimedia algorithms using the AltiVec technology," *Proc. ACM Int. Conf. on Supercomputing*, June 20 - 25, 1999, pp. 11-20, Rhodes Greece.

[8] Motorola, "AltiVec Technology." *http://www.mot.com/AltiVec.*

[9] AltiVec Information Source. *http://www.altivec.org.*

[10] J. D. Allen and D. E. Schimmel, "Issues in the Design of High Performance SIMD Architectures," *IEEE Trans. on Parallel and Distributed Systems*, vol. 7, pp. 828-839, Aug. 1996.