

**Implementation of a 3-D Sonar Beamformer Using the
Computational Process Network Model on
a Synergy Quad PowerPC G4 with AltiVec Board**

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor: _____

**Implementation of a 3-D Sonar Beamformer Using the
Computational Process Network Model on
a Synergy Quad PowerPC G4 with Altivec Board**

by

Young Hyun Cho, B.A.

REPORT

Presented to the Faculty of the Graduate School of
The University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN ENGINEERING

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2001

**Implementation of a 3-D Sonar Beamformer Using the
Computational Process Network Model on
a Synergy Quad PowerPC G4 with AltiVec Board**

Young Hyun Cho, M.S.E.

The University of Texas at Austin, 2001

Supervisor: Brian L. Evans

Three-dimensional real-time digital sonar beamforming requires 4 to 12 GFLOPS, 1 to 2 GB of memory, and 100-200 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 16 336-MHz processors by utilizing the Visual Instruction Set (VIS) single-instruction multiple-data (SIMD) extensions and the Computational Process Network (CPN) dataflow model. In the report, I rewrite the horizontal and vertical beamforming kernels to use the Motorola AltiVec SIMD extension for the PowerPC. Then I develop a scalable beamforming software system using the CPN on a Synergy Quad 333-MHz PowerPC G4 symmetric multiprocessing (SMP) board.

While the SPARC VIS offers performance increases for signal processing kernels, AltiVec offers better performance due to its wider SIMD register size. In addition to SIMD integer operations, AltiVec can execute up to four 32-bit floating-point multiply and accumulate (MAC) operations per instruction. For the 128-bit SIMD AltiVec register operations, using data prefetching and permutation instructions are necessary to utilize the full capability of AltiVec.

For example, transposing matrices in the 3-D sonar beamformer is handled without computational overhead using permutation instructions. I evaluate the performance of vertical and horizontal beamforming kernels on the PowerPC and the UltraSPARC-II to compare the impact of the compiler, SIMD word alignment, and cache block alignment on performance.

For computationally intensive applications such as the 3-D sonar beamformer, scalability is a key aspect of the system. Thus, the Computational Process Network model is the design framework of the beamforming system. This programming model decouples the computation processes (nodes) from the communication processes (queues). In a 3-D beamforming system, the nodes consist of the sonar sensors, the vertical beamforming kernels and the horizontal beamforming kernels. These nodes communicate through the preallocated memory which work as FIFO queues. On an UltraSPARC-II multiprocessor system, the CPN 3-D sonar beamformer shows near-linear speedup up to 16 processors.

I port the CPN 3-D sonar beamformer from the Sun to the Quad PowerPC G4 SMP board using the new beamforming kernels and transposed queues. On the PowerPC board, I discover performance limitations due to the cache hierarchy. I evaluate the importance of interconnection in determining scalable performance with the high-memory bandwidth application which require relatively high memory bandwidth.

Table of Contents

Abstract	iv
Table of Contents	vi
List of Tables	ix
List of Figures	x
1. Introduction	1
1.1 Motivation	1
1.2 Report Summary	3
2. Background	5
2.1 Native Signal Processing Extensions	5
2.1.1 UltraSPARC Visual Instruction Set	6
2.1.2 PowerPC AltiVec	6
2.2 Beamformer	7
2.3 Computational Process Network Model	9
2.4 Conclusion	11
3. Beamformers using AltiVec	13
3.1 Beamforming Kernels	13
3.2 Implementation using AltiVec	15
3.2.1 Vertical Beamforming Kernel	16

3.2.2	Horizontal Beamforming Kernel	17
3.3	Kernel Benchmark	18
3.3.1	Performance Analysis	19
3.3.2	Cache Performance	20
3.4	Conclusion	21
4.	Scaling in Symmetric Multi-Processing Platforms	23
4.1	Multiprocessor	23
4.1.1	System Design	23
4.1.2	Scalability	24
4.1.3	Implementation Platform	26
4.2	Symmetric Multiprocessor Implementation	27
4.2.1	Ultra SPARC II Server	27
4.2.2	Synergy Quad PowerPC G4	29
4.3	Multi-threaded Beamforming System	31
4.3.1	Previous Implementation	32
4.3.2	Implementation using AltiVec	34
4.4	Performance and Memory	36
4.4.1	Memory Configuration	37
4.4.2	Performance Bottleneck	38
4.5	Conclusion	38
5.	Conclusion	40
5.1	Beamforming Kernel	40
5.2	Scalable System	42
5.3	Possible Future Implementation	44

BIBLIOGRAPHY	45
Vita	49

List of Tables

1.1	Three generations of low-volume 4-GFLOP sonar beamformer. COTS refers to commercial-of-the-shelf technology [2].	2
4.1	Horizontal Beamforming Benchmark using 336-MHz UltraSPARC- II Enterprise 40001 [2]	32
4.2	Vertical Beamforming Benchmark using 336-MHz UltraSPARC- II Enterprise 4000 [2]	32
4.3	Horizontal Beamforming Benchmark using 333-MHz Synergy Quad G4 VSS4 board	34
4.4	Vertical Beamforming Benchmark using 333-MHz Synergy Quad G4 VSS4 board	34
4.5	Test benchmark for the vertical kernel on the PowerPC board when reusing the cached data	36

List of Figures

2.1	PowerPC G4 AltiVec vector Unit	6
2.2	Sonar Beamformer	7
2.3	Ten vertical samples form a stave. An array of 800 sensors consists of 80 staves.	8
2.4	Block diagram of 3-D Sonar Beamformer. The input data rate is 160 MB/s and the output data rate is 72 MB/s.	8
2.5	Beamformer diagram in nodes and directed edge	10
3.1	Vertical kernel algorithm using AltiVec	16
3.2	Horizontal Kernel algorithm using AltiVec [12]	17
3.3	Vertical and Horizontal Kernel Benchmark [12]	19
4.1	Three different multiprocessor models: (a) is a special case of (b).	25
4.2	Sun Microsystems UltraSPARC II Enterprise 4000 server block diagram [16]	28
4.3	Synergy Microsystems VSS4 - Quad PowerPC G4 board block diagram	30

1. Introduction

The digital signal processor (DSP) market has been growing at an approximate rate of 40% per year since 1992 [1]. These predictions are based on the economic figures of ever increasing integration of the signal processors in many different consumer products such as cellular telephones, voiceband modems, ADSL modems, and audio CD players, as well as industrial applications that are computationally intensive.

As fast signal processing becomes essential part of the growing number of products and applications, flexible and inexpensive development platforms will become very important. To incorporate fast signal processing functions on desktop computers and servers, popular general-purpose processors such as Intel Pentium III, AMD K-7, UltraSPARC II, and PowerPC G4 include signal processing acceleration in the instruction set architecture. Since these are extensions to the core processor instruction set for the special purpose of signal processing, they are referred to as native signal processing (NSP) instructions. The general-purpose processors with NSP extensions are emerging system solutions for multimedia and signal processing applications, especially for desktop and server applications requiring high performance.

1.1 Motivation

NSP extensions to general-purpose processors allow programmers to implement signal processing algorithms efficiently in a convenient development

	Custom Hardware	Embedded COTS	Workstation
Development cost	\$2,000 K	\$500 K	\$100 K
Development time	24 months	12 months	6 months
Physical size (m^3)	0.067	0.067	0.089
Reconfigurability	low	medium	high
Software portability	low	medium	high
Hardware upgradability	low	medium	high

Table 1.1: Three generations of low-volume 4-GFLOP sonar beamformer. COTS refers to commercial-of-the-shelf technology [2].

environment that is the same as the target environment. In recent times, NSP extensions have featured fast and powerful functionality that is comparable to specialized digital signal processors. Digital signal processors have a throughput of one or two multiply-accumulate operations per cycle. Multiply-accumulate operations are fundamental in vector dot products, matrix multiplication, filters, and fast Fourier transforms. NSP instructions have a throughput of four or eight multiply-accumulate operations per clock cycle. Clock rates are currently three times higher for general-purpose processors. This motivated Allen and Evans [2] to explore the use of servers and desktop computers as target platforms for high-performance signal processing applications.

One high-performance signal processing application is real-time 3-D digital beamforming. Real-time 3-D digital sonar beamforming can require several billions of floating-point operations per second (GFLOPS) of computation and 100-200 MB/s of I/O. Before the 1990s, these beamformers were implemented in custom hardware because they required higher accuracy and memory bandwidth than commodity hardware systems could offer. However, with the rapid growth in demand and technological advances in programmable

digital signal processors, beamformers were built using commercial off-the-shelf (COTS) components by the mid 90s. The addition of native signal processing extensions to the Sun SPARC in 1995 and support by Unix operating systems for symmetric multiprocessing (SMP) made it possible to implement high-performance signal processing application on commodity multiprocessor servers. Allen and Evans use a commercial general-purpose 16-processor SMP server from Sun Microsystems to realize a 3-D sonar beamformer in real-time [2]. A comparison of these three generations of beamformer implementations is shown in Table 1.1.

1.2 Report Summary

Signal processing kernels of the beamformer exploit data parallelism by using Single Instruction Multiple Data (SIMD) arithmetic operations, which are available in native signal processing extensions, such as the Visual Instruction Set (VIS) on the Sun UltraSPARC-II processor. By using a Sun Ultra Enterprise server with at least 10 450-MHz UltraSPARC-II processors and 1.2 GB of memory, a real-time beamformer delivering 4 GFLOPS on 160 MB/s of streaming input data can be realized.

In Chapter 2, I first define two native signal processing extensions, Sun's VIS and Motorola's AltiVec, which are used to implement the beamforming system. Then I summarize the components of the beamforming algorithm in [3]. I also briefly present the practical design decisions made for the implementation. In my multiprocessor implementation, I use the Computational Process Network model [2]. The Computational Process Network model de-

couples computation from communication and maps well to scalable software.

Chapter 3 describes the Allen and Evans implementation [2] of the beamformer and its real-time performance on an UltraSPARC II server. Then I further explore the effectiveness of NSP extensions by optimizing and assessing the performance of beamforming kernels using the Motorola AltiVec NSP extension on the PowerPC G4. I 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. Also in Chapter 3, I expand the implementation as a scalable software system framed under the Computational Process Network model. The scalability of the previous beamformer was tested on a 16-processor UltraSPARC II server, whereas the proposed beamformer was tested on a four-processor PowerPC G4 board. Although they are architecturally different targets, they are both symmetric multiprocessor (SMP) systems.

Chapter 4 begins with a brief summary of multiprocessor systems. When comparing the performance scalability of the system in Synergy Quad PowerPC G4 with UltraSPARC II SMP system, I address the impact of different software design techniques for the given hardware architectures.

In the concluding chapter, I summarize the performance comparison results of the beamforming kernel in UltraSPARC II and PowerPC G4 processors. I end the report with general suggestions for the software design techniques of high-performance applications to approach optimal performance with different multiple SIMD processor hardware architecture.

2. Background

2.1 Native Signal Processing Extensions

Many high-performance embedded applications are programmed on systems with a few general-purpose processors as system controllers and a larger number (possibly hundreds) of specialized digital signal processors (DSPs) to perform scientific calculations. However, this type of system has many disadvantages. First, the development environment (a workstation) is completely different from the target environment. Second, DSPs show 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 execution environment, which allows easier and faster software development. This allows the development and target environments to be the same. It allows leverage of the commodity desktop computers for system development and commodity servers for system deployment [4].

A SIMD architecture allows one instruction to be performed on multiple pieces of data, thereby potentially increasing the computation performance proportional to the size of the SIMD registers. However, most SIMD architectures require that the input data be aligned in memory according to the size of the SIMD register. Therefore, if the data were not aligned in memory, then

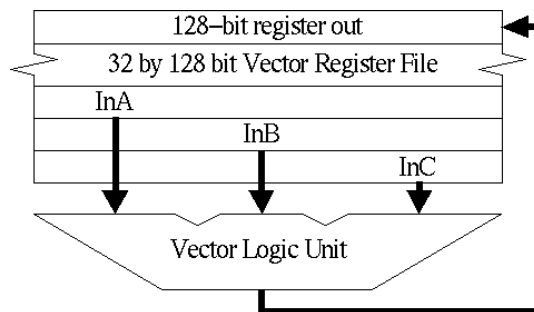


Figure 2.1: PowerPC G4 AltiVec vector Unit

additional instructions would be needed for the data permutation to align the data.

2.1.1 UltraSPARC Visual Instruction Set

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 a graphics integer unit to support VIS. With 50 new CPU instructions, VIS can perform an integer operation on multiple words with a single instruction. VIS registers are 64 bits long. Thus, VIS can achieve up to a four times speedup with an 8-bit by 16-bit fixed-point multiplication using the SIMD arithmetic logic [5].

2.1.2 PowerPC AltiVec

The vector unit for the AltiVec NSP extension is partitioned into a separate sub-unit of the processor as are with the floating-point and integer units. As shown in Figure 2.1, 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

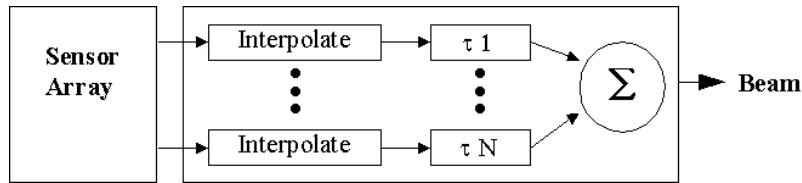


Figure 2.2: Sonar Beamformer

instruction [6].

In addition to SIMD arithmetic, the unit is capable of performing fast permutation on 128-bit wide data. The hardware permutation unit allows efficient implementation of signal processing algorithms such as matrix multiplication and fast Fourier transform. With such a powerful vector unit, AltiVec offers greater computing resources than VIS.

2.2 Beamformer

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

The time delay resolution required is typically several times the Nyquist rate (the sampling rate must be higher than the Nyquist rate to preserve the frequency content of the signals). Instead of sampling at a higher rate, digital interpolation with Finite Impulse Response (FIR) filters to gives a satisfactory time delay resolution. This is called digital interpolation beamforming, and is shown in Figure 2.2. Analog data is sampled at just above the Nyquist rate

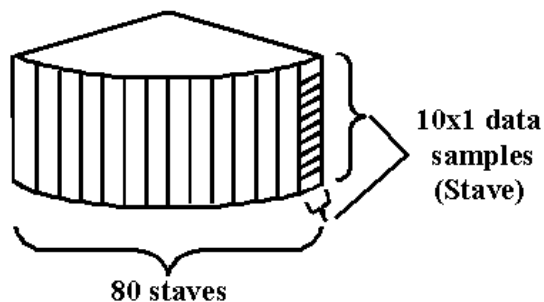


Figure 2.3: Ten vertical samples form a stave. An array of 800 sensors consists of 80 staves.

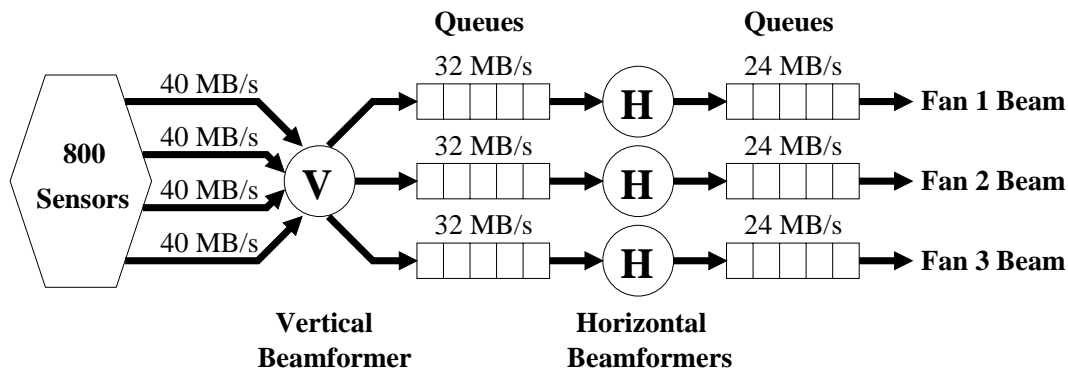


Figure 2.4: Block diagram of 3-D Sonar Beamformer. The input data rate is 160 MB/s and the output data rate is 72 MB/s.

and then interpolated, delayed, and summed [3].

For implementation, a 3-D beamformer can be decomposed into horizontal and vertical beamforming subsystems. This method of beamforming enables 3-D mapping of underwater surfaces. 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 Figures 2.3 and 2.4. For improved performance of the vertical beamformer, I use SIMD integer computation. The vertical beamformer also

converts the data to floating-point format for the following horizontal stages [8]. With the data input rate of 160 MB/s, the vertical subsystem needs to perform at 500 MFLOPS to run in real-time.

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 [1]. Each horizontal beamformer needs to process at the rate of 1200 MFLOPS to run in real-time. The overall system description is shown in Figure 2.4.

The system can also leverage the Computational Process Network model which is described in the next section [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 avoid artificial deadlock from writing to a full queue [9].

2.3 Computational Process Network Model

The Process Network programming model can be used to construct parallel applications. In this model, each independent process communicate with other independent processes through unidirectional first-in first-out (FIFO) queues. There can be more than one input or output queue connected to each process. According to the model semantics, a process blocks (stops execution) when it attempts to read data from an empty queue. Otherwise, each process

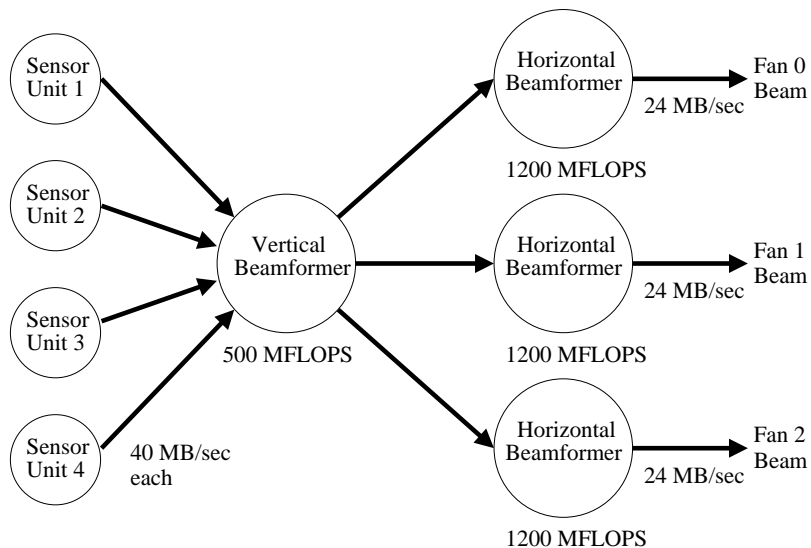


Figure 2.5: Beamformer diagram in nodes and directed edge

is allowed to produce output continually. Since the data producing process can potentially push more data to the queue than the consuming process, FIFO queues could grow infinitely large [10].

Since bounded memory is necessary in any practical system, infinite queues should not be allowed if at all possible. Thus, Parks [9] developed dynamic scheduling rules for a Process Network that always find a bounded memory schedule if one exists. The rules dictate that the process halts execution when reading from an empty queue or writing into a full queue. If the process is halted due to a full queue, i.e. artificial deadlock, then the length of the smallest full queue is increased until the corresponding process can write into the queue. When using a directed graph syntax for specifying a Process Network, each process is a node and each FIFO queues is a directed edge [2].

The Computational Process Network (CPN) model modifies the Pro-

cess Network model to allow all memory management operations to be performed by the unidirectional queues. CPN adds a firing threshold to each unidirectional queue. This enables the node to consume (dequeue) a smaller number of samples than the number of samples read. This better matches the processing of streaming data using overlapping blocks, e.g. filtering. If a bounded implementation were possible, then all of the streaming data could be managed by the bounded memory queue when the threshold is set with an experimentally determined value. The modified model allows each node to work on the memory itself directly, which makes data copying unnecessary.

The CPN programming model is suitable for constructing computationally intensive, streaming signal processing applications such as the real-time 3-D sonar beamformer in scalable software. Each of the sensor, vertical, and horizontal beamforming processes can be constructed into nodes and the pathways from the sensors to the vertical beamformer, from the vertical to horizontal beamformers, and to the resulting outputs can be represented with directed edges, as shown in Figure 2.5.

2.4 Conclusion

This chapter describes the background information necessary for constructing a scalable real-time 3-D sonar beamforming system on SMP workstations that have processors with native signal processing extensions.

First, NSP extensions on UltraSPARC-II and PowerPC G4 processors are examined. UltraSPARC-II's Visual Instruction Set (VIS) is specialized to process SIMD integer operations. AltiVec extension on PowerPC G4 has a

more powerful instruction set than VIS in that it uses registers twice as wide as VIS. In addition to integer operations, AltiVec also includes floating-point instructions that can execute up to four MAC per instruction cycle.

Second, the beamformer algorithm from [3] is described. The real-time 3-D sonar beamformer uses an 80 by 10 sensor array, one 500-MFLOP vertical beamformer, and three 1200-MFLOP horizontal beamformers.

The beamforming kernels can take advantage of NSP extensions of UltraSPARC-II and PowerPC G4 processors. However, the computational requirement exceeds the capability of a single processor. Thus, in order to achieve scalable performance on SMP machines, the kernels can be put into Computational Process Network model of the 3-D sonar beamformer.

The previous beamformer implementation by Allen and Evans [1] can meet the real-time constraints of the 3-D sonar beamformer on an UltraSPARC-II SMP server. The performance of new beamforming kernels using the AltiVec extension from Motorola is described in the next chapter for comparison with the previous UltraSPARC implementation.

3. Beamformers using AltiVec

Using Motorola PowerPC G4/7400 processors with AltiVec, I confirm the evaluations assessed with PowerPC AltiVec simulators, *Sim_G4* and *Apple AltiVec Emulator* [11]. I evaluate the results of the beamforming kernels programmed with AltiVec on the G4 processor in a real-time system context.

3.1 Beamforming Kernels

The beamforming algorithm described in Section 2.2 has two stages of computation. The first stage is the vertical beamformer, which computes three different vertical beams using the raw data collected by the sensors. Using the computation results from the vertical beamformer, the second stage contains three horizontal beamformers to compute the horizontal beams.

I implement vertical and horizontal beamformer using AltiVec extensions. I compiled and tested the kernels using the GCC compiler 2.95 with AltiVec enabling patches [7] in SMP Linux operating system version 2.2. An AltiVec enabling patch provided by Motorola is applied to the GNU C Compiler (GCC) to enable compilation of AltiVec instructions. 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 to the UltraSPARC processor, 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 [6, 7].

Previous implementation

The beamforming algorithm described in Chapter 2 was previously implemented on Sun UltraSPARC II servers. Sun UltraSPARC II processor's VIS signal processing extension is a set of SIMD integer instructions. The data structure of the input data for the vertical beamformer is in 16-bit integer format and the coefficient data is in floating-point data type. Thus, by scaling the coefficient data to be 16-bit integers, the vertical beamformer can use the integer native signal processing extensions. In the implementation, the vertical beamformer uses two 8-bit by 16-bit multiplications and two 32-bit additions for the theoretical peak performance of one 16-bit multiply and accumulate per two clock cycles (one instruction per cycle). VIS could not be used for the horizontal beamformer because higher precision is necessary for its interpolation step. The horizontal beamformer, therefore, uses the floating-point instructions of the UltraSPARC processor. Memory latency hiding techniques such as loop unrolling and software data prefetching are used to achieve near peak performance for the implementation [8].

By using architecture specific compiler optimization, partially hand-coded implementation, and data prefetching, the performance of the beamformer on UltraSPARC II processor gave a performance of 313.3 million integer operations per second (MIOPS) and 444.3 million floating-point operations per

second (MFLOPS) per processor [2].

3.2 Implementation using AltiVec

In the previous implementation, the input memory 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 at two consecutive times, the original queue structure would require the sample points to be referenced with two different addresses with an offset that is equal to the size of a stave. By transposing the queue, the samples for each stave become contiguous according to their sampling times. Such an arrangement would allow each vector load instruction to load up to four samples required for the calculation at once. I refer to the transposing of the queue as *cornerturning*.

Even though transposing queue data to implement *cornerturning* is efficiently handled by the vector load, it introduces a few problems. First, the index calculation becomes more complex because *cornerturning* requires the queue size to be added or subtracted when the kernel accesses 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 the input data structures. However, the second problem can be avoided by allocating the queues to be large enough to prevent artificial deadlock [12]. The queue sizes can be experimentally determined.

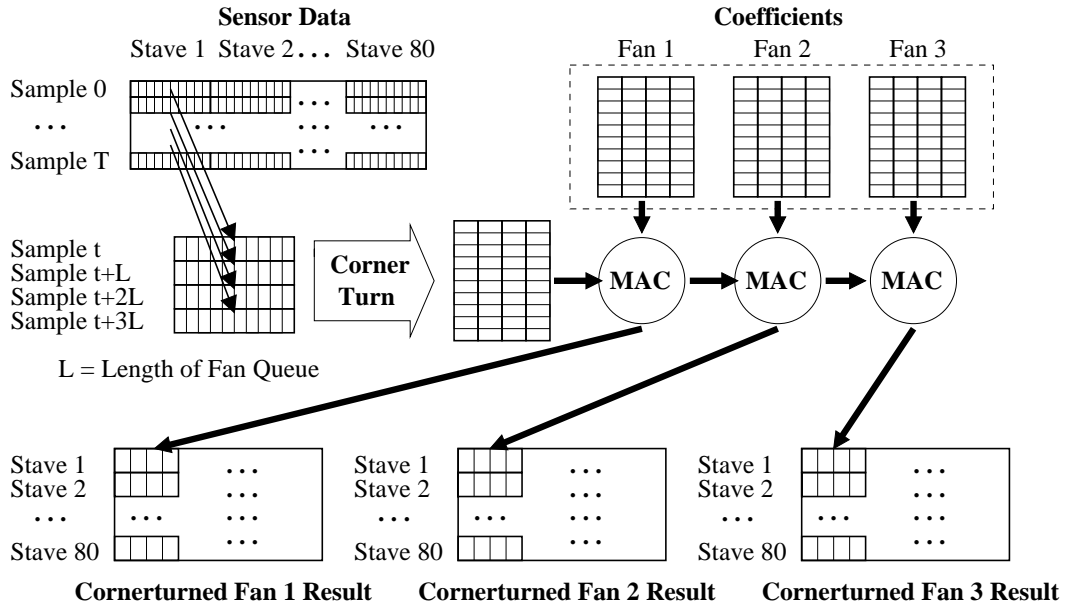


Figure 3.1: Vertical kernel algorithm using Altivec

3.2.1 Vertical Beamforming Kernel

As shown in Figure 3.1, the vertical beamformer needs 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 as 32-bit integers. To accommodate the horizontal beamforming kernel, the results are first converted to 32-bit floating-point numbers and then cornerturned 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.

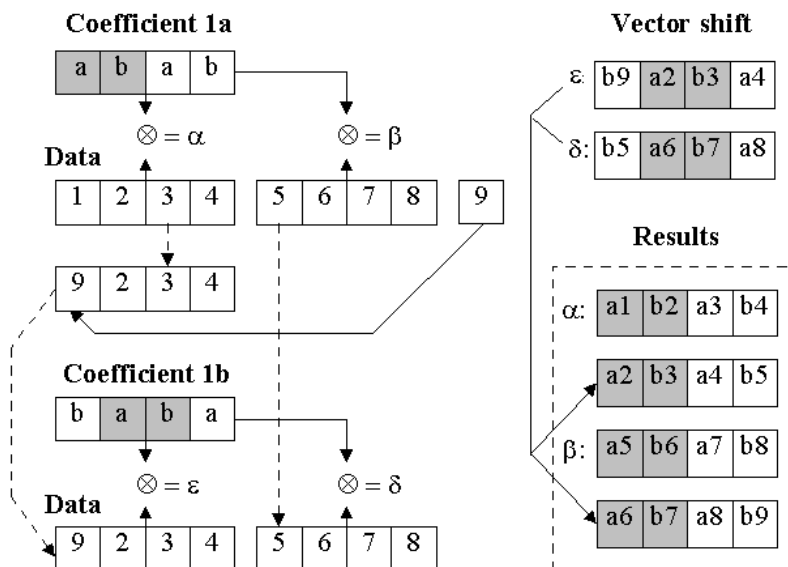


Figure 3.2: Horizontal Kernel algorithm using AltiVec [12]

The resulting vectors are stored in strides to accomplish cornerturning of the entire contents of the queue. The cornerturned data in the queue is fed into the horizontal kernels to reduce data permutations required for the input vectors.

The vertical beamforming algorithm performs dot products that require large amounts of sequential data in a short time. Therefore, every time that 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. I use the AltiVec data prefetching instructions to reduce data cache misses [12].

3.2.2 Horizontal Beamforming Kernel

The horizontal beamformer loads the cornerturned samples using the vector load operation. The number of load instructions is reduced due to the AltiVec vector operations. Then the data is permuted to meet the alignment

constraints for the instructions before the computation.

As described in Section 2.2, the horizontal beamforming algorithm performs two-point interpolation with consecutive samples from each stave. Because the input to the horizontal kernel has been cornerturned, consecutive samples in memory are consecutive samples of the same stave in time. As shown in Figure 3.2, data blocks labeled 1 through 9 represent sequential samples collected from a given stave.

To reduce the number of load and permute instructions, two sets of the same coefficients are stored, each of which is aligned differently by one floating-point word. The vector multiply-and-accumulate (MAC) instruction computes the result using two pairs of coefficients, which yields the two-point multiplication. Up to four floating-point MACs 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 cornerturned output queue of the horizontal beamformer [12].

3.3 Kernel Benchmark

Figure 3.3 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

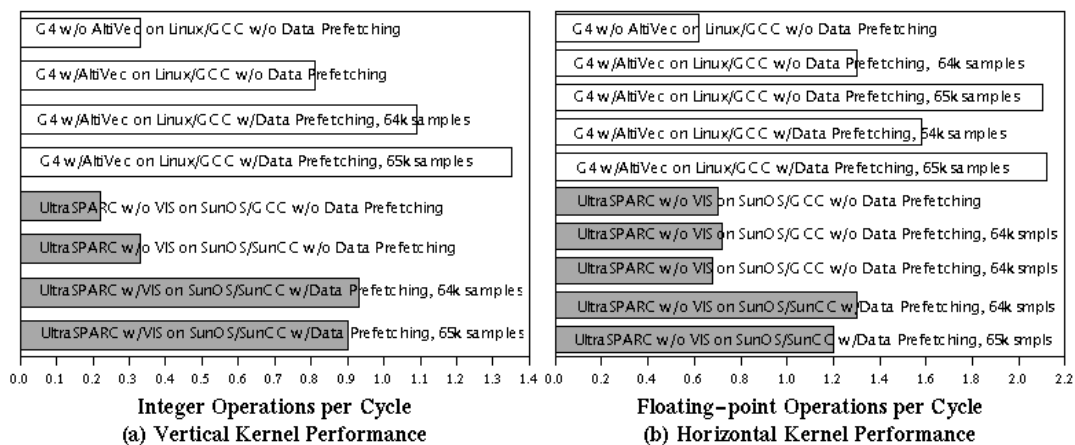


Figure 3.3: Vertical and Horizontal Kernel Benchmark [12]

the development platform such as the operating system and compiler, as well as the usage of the native signal processing extensions, prefetch instructions, and input data size.

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.

3.3.1 Performance Analysis

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 numbers of loop unrolling iterations for the generated code. Figure 3.3(b) shows the

performance of the best versions. which is approximately 1.2 to 1.4 times the performance of the Sun implementation.

In Sun implementation, use of VIS in the vertical kernel increased its performance by a factor of two. Since AltiVec uses registers that are twice as wide as VIS, use of AltiVec should increase the vertical kernel performance by two. However, using the AltiVec extensions in the code only doubled the performance of the kernel. 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. This indicated that the bottleneck in the vertical beamforming implementation was not in the instruction count. Thus, I turned my attention to cache performance of the machine.

3.3.2 Cache Performance

Level-1 (L1) cache in PowerPC G4 consists of 32 KB instruction and 32 KB data cache. Each cache is eight-way set associative and each block contains 32 bytes. Since there are eight blocks in each set, 32 KB of cache is divided into 128 sets [19].

In vertical beamformer, all input data is loaded and used only once. Therefore, the performance would not be effected even when the cache line of previously loaded data is replaced with the newly requested data. However, when the vertical beamformer invokes store, the alignment of the result queue can cause the cache line to be written back to the main memory before it can be replaced. Since L1 cache has 128 sets with 8 32-byte blocks, cache write-

back can occur with every vector store when the offset of the subsequent vector stores are 8192 bytes and its multiples.

The size of the result queue for the vertical beamformer is determined by the number of samples. Due to cornturning of the result queues, sequence of stores for the stave samples are spaced by the length of the samples. Therefore, the vertical kernel producing 65536 samples invokes most amount of cache write-back because the offset for each store is 262144 bytes which is a multiple of 8192 bytes. Figure 3.3 shows that allocating the queue lengths to be a 66560 samples instead of 65536 samples made the vertical beamforming kernel to perform about 20% better. As expected, a similar performance increase occurs when the input data queues sizes were not a multiple of 8192 bytes in horizontal kernel.

Since vertical beamformer requires constant stream of new input data, the performance is also largely determined by whether the input data is prefetched in to the cache from the memory. Through several iterations of performance tests, I found that an efficient use of the prefetch instruction in the vertical kernel increased its performance by 20%. However, the data prefetching did not affect the performance significantly in the horizontal beamformer. This is likely due to the kernel implementation that emphasizes coefficient and sample data reuse, which reduces the total number of cache misses [12].

3.4 Conclusion

This chapter affirms the importance of front and back-end compiler optimization, cache alignment and utilization, and NSP extensions of two dif-

ferent processors. Although PowerPC G4 technology is relative immature, the AltiVec implementation performed 1.56 to 1.83 times faster than Sun's VIS for the same clock speed and cache size.

The beamforming algorithm requires 480 MFLOPS for the vertical beamformer and total of 3600 MFLOPS total for the three horizontal beamformers. Therefore, it is not possible to implement the entire system on single processor. The previous implementation by Allen and Evans [2] required approximately ten 450 MHz UltraSPARC II processors working in parallel in multiprocessing environment. In this chapter, I benchmark individual kernels written in AltiVec to be used in a multiple processor implementation. My goal is to implement a 4-GFLOP 3-D sonar beamformer in real-time using four 550-MHz PowerPC G4 processors.

4. Scaling in Symmetric Multi-Processing Platforms

Real-time 3-D sonar beamformer applications need to compute on the order of billions multiply-and-accumulate (MAC) operations per second. Scalability of the kernels is a key in achieving this performance on multiprocessor platforms available today. The beamforming algorithm implementation described in Chapter 2 is naturally matched to multiprocessor systems due to the data and functional parallelism present.

4.1 Multiprocessor

Many different types of multiprocessor environments are currently used and researched for different applications. These types of hardware drastically differ in performance and usage due to different sizes and configuration in processors, memory, and interprocessor communication bandwidth and latency.

4.1.1 System Design

With advances in the commodity workstation market, several types of cluster computers with high-speed interconnections have emerged from research and development. These multiprocessor workstations and servers are built with a relatively high number of processors ranging up to several thousands. Typically, each processor would have with its own memory, operating system, and interconnection device. Therefore, these systems usually occupy a large physical space. Examples of cluster computing include networks of workstations

and two-level multiprocessor of Myricom [13]. Memory coherency policy of such potentially large distributed systems may be maintained independently in the groups of processors via the Message Passing Interface (MPI) [16] or other generic types of communication protocols.

One of the most popular types of multiprocessor platforms is referred to as Symmetric Multi-Processing (SMP). This platform consists of a relatively low number of processors ranging from two to sixteen processors. The platform usually attempts to give scalable performance while hiding the effects of multiple processors by maintaining one large coherent memory. Since these multiprocessors reside on single board or a box, it takes relatively less physical space than a cluster of computers. SGI Power Challenge [14], UltraSPARC Enterprise 4000 with 16 UltraSPARC II processors, and Synergy SBC-VSS4 board with four PowerPC G4 processors [15] are all categorized as SMP systems.

There are systems that combine both types of platforms to produce more powerful multiprocessing solutions, e.g. clusters of SMP workstations. Such systems include CLUMPS of the University of California at Berkeley and the Sun Enterprise 10000. Convergence of different multiprocessing platform brings about generic standardization of the performance measurement. Measuring the performance trade-offs between clusters of computers and SMPs is rather a recent research topic which is becoming increasingly active [16].

4.1.2 Scalability

The primary problem to address in scalability is the interconnection. Memory bandwidth and the type of processor interconnection largely determine

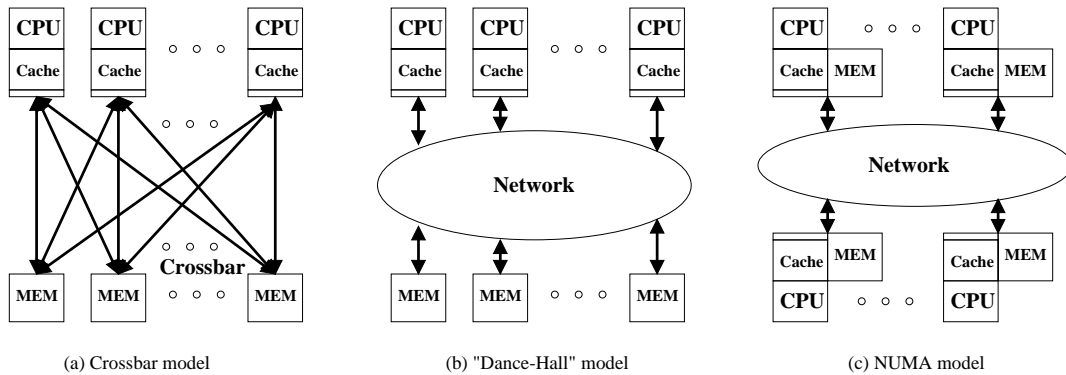


Figure 4.1: Three different multiprocessor models: (a) is a special case of (b).

the scalability of a system. The preferred multiprocessing environment from a performance point of view would be to have interconnections with the lowest memory latency while maintaining highest bandwidth between the processors and the memory. Other than for research purposes, such systems would be less feasible due to their high price. The key to producing the most practical high performance computer is in finding a good balance between cost and bandwidth to achieve the lowest cost to performance ratio.

As shown in Figure 4.1(a), system design model with crossbar processor to memory interconnection has a scalable memory with low latency; however, the implementation and maintenance costs are high because of uses of expensive technologies such as crossbar along with high-speed communication. The "Dance-hall" system design model attempts to achieve bandwidth that is still scalable, but the cost of interconnection technology is reduced by using a less expensive network configuration, such as the simple loop in Figure 4.1(b). Such modification, however, uniformly increases the memory latency, which impacts the overall performance of the scaled system. Another model

shown in Figure 4.1(c) uses distributed memory or non-uniform-memory-access (NUMA) configuration. NUMA constructs shared address space out of simple message transactions across a general purpose network, e.g. clusters of computers [15].

The above configurations and the combinations of the models make up the current systems with different trade-offs to facilitate balanced cost and performance. Therefore, the applications performance varies from one system to another based upon their resource requirements.

4.1.3 Implementation Platform

The sonar beamforming system is low-volume, computationally demanding application that must fit into a small physical space. In consideration of physical space and efficient implementation of the beamforming kernels using NSP extensions, I choose UltraSPARC Enterprise 4000 and Synergy Quad PowerPC G4 SMP systems, with each running a Unix operating system, for the target platforms.

The Enterprise 4000 server uses a combination of expensive crossbar technology and the “dance-hall” interconnection model. The Synergy server connects all four processors using the “dance-hall” model. As concluded in Chapter 3, kernels written in AltiVec extension of G4 outperform the UltraSPARC by factor of 1.5 to 1.8. However, in following sections, I observe a different behavior for the scaled version of the vertical beamformer on two different SMP systems. This illustrates the importance of interconnection in scalability. Thus, I address the impact of the different NSP utilization and

interconnection in performance of the entire system.

4.2 Symmetric Multiprocessor Implementation

4.2.1 Ultra SPARC II Server

Figure 4.2 shows the block diagram of Sun server I used for the beam-forming system. The system is composed of six dual 336-MHz UltraSPARC II processors CPU/Memory (CM) board with VIS. On each of the six CM boards, there are two CPUs with 4 MB of level-2 cache and 512 MB of memory. The size of the memory scales linearly as the number of CM boards increase [17].

Each CM board is connected by using the expensive crossbar interconnection model. As described in Section 4.1, this model is quite efficient due to fast interconnection to the local bank of memory. The processors, memory, and I/O interface are all connected through crossbar technology. The interconnection is controlled by a STP2202BGA dual system controller (DSC). The DSC supports a 288-bit wide memory bus running at 84 MHz [18]. The total memory bandwidth of 3.034 GB/s from each memory bank is connected to crossbar and shared by two UltraSPARC-II processors and the I/O interface. The connection between both processors and the crossbar is 128-bit wide running at 84 MHz. Therefore, theoretical maximum memory bandwidth of 1.344 GB/s is shared in the “dance-hall” model interconnection between each pair of processors.

The “dance-hall” interconnection model is used for the data accesses through the 2.83 GB/s Gigaplane bus for inter-board communication. Due to crossbar interconnection from memory to the I/O interface, memory accesses

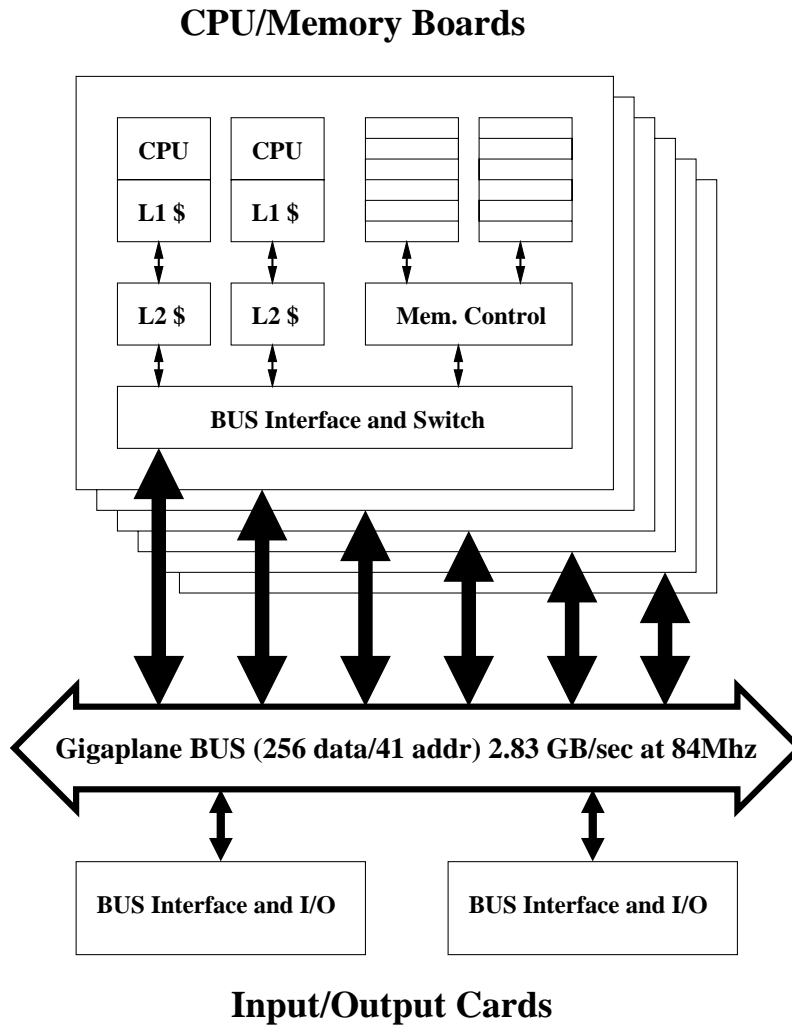


Figure 4.2: Sun Microsystems UltraSPARC II Enterprise 4000 server block diagram [16]

from the I/O do not use the bandwidth allocated for the processor. Also, since two CPUs share the on-board memory, if the data needed by either processor is located in the local memory, it allows the processors to load the data from the local memory directly without the use of Gigaplane. Hence, the Sun Enterprise 4000 SMP server integrates the combination of the crossbar and “dance-hall” interconnection models to maintain high memory bandwidth while keeping the latency and cost low.

The initial implementation of the real-time sonar beamforming system on a commodity server runs on the Sun SMP server. The system was developed under the Solaris 2.7 Unix operating system using the Sun C compiler customized for the UltraSPARC processor to use the architectural features efficiently. Allen and Evans [2] give detailed benchmark figures and analysis of the implementation which shows results indicating a possible real-time implementation of the system on 12 336-MHz UltraSPARC-II processors.

4.2.2 Synergy Quad PowerPC G4

The design of the Synergy Quad G4 board is much like the CM board of the Sun Enterprise server. Each of the four MPC7400 (PowerPC G4) processors has 2 MB of level-2 (L2) cache for faster load/store without having to access the slower on-board memory. 512 MB of on-board synchronous dynamic random-access-memory (SDRAM) is shared by the four processors via PowerPC 60X Bus controlled by a memory controller. Although the PowerPC G4 processor supports a newer MPX Bus, the Synergy board does not utilize it because Motorola does not yet have a memory controller chip which supports the MPX Bus with four processors. Therefore, the board is designed with an

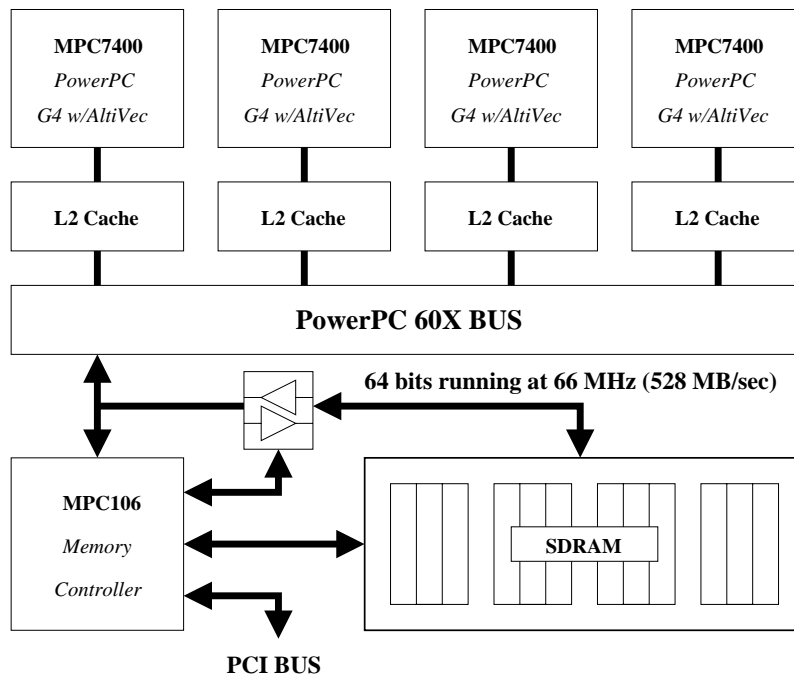


Figure 4.3: Synergy Microsystems VSS4 - Quad PowerPC G4 board block diagram

older memory controller (MPC106) which supports the 60X Bus and up to four PowerPC processors [19, 20, 21].

MPC106 on the Synergy board implements 64-bit wide bus running at 66 MHz. With this memory controller configuration, the theoretical maximum memory bandwidth is 528 MB/s. This memory bandwidth is about one-sixth of the bandwidth available on a single Sun CM board. In addition, the Synergy board shares this memory bandwidth among four processors, whereas each processor on Sun CM board has up to 672 MB/s of memory bandwidth. Assuming that each G4 processor is saturating the memory bandwidth, the theoretical maximum memory bandwidth for each processor is 132 MB/s, which is about five times lower than that of the Sun CM board [21].

For the beamforming application, a major advantage of the Synergy board over the Sun server is the PowerPC G4 (MPC7400) processor. As shown in the performance benchmarks in Chapter 3, the beamformer kernels using AltiVec outperforms the kernels written in VIS by a factor of 1.5 to 1.8 times. However, the “dance-hall” interconnection model with four processors and legacy shared-memory interconnection are much less efficient compared to the crossbar model on the Sun Enterprise CM board.

4.3 Multi-threaded Beamforming System

Many computationally intensive signal processing algorithms such as real-time sonar beamforming have high degrees of parallelism. Dividing the computation algorithm into multiple independent tasks is possible because the number of divisible tasks containing data-dependent samples are usually much

Implementation	Time (s)	MFLOPS	Speedup	Percent Utilization
kernel	7.252	443.6	(1.000)	(100.00%)
2 threads	3.691	871.6	1.965	98.24%
4 threads	1.895	1697.5	3.827	95.67%
6 threads	1.277	2518.6	5.678	94.63%
8 threads	0.973	3306.7	7.454	93.18%

Table 4.1: Horizontal Beamforming Benchmark using 336-MHz UltraSPARC-II Enterprise 40001 [2]

Implementation	Time (s)	MFLOPS	Speedup	Percent Utilization
kernel	4.037	311.7	(1.000)	(100.00%)
2 threads	2.082	604.3	1.939	96.94%
4 threads	1.128	1115.9	3.580	89.50%
6 threads	0.802	1543.9	4.953	82.55%
8 threads	0.661	1905.0	6.112	76.40%

Table 4.2: Vertical Beamforming Benchmark using 336-MHz UltraSPARC-II Enterprise 4000 [2]

larger than the number of processors. The beamforming system uses the optimized kernels described in Chapter 3 and divides the workload into several threads in order to run them in parallel on different processors. To facilitate parallel execution, lightweight POSIX threads are used for the vertical and horizontal kernels. The threads are encapsulated in the nodes, which are then interconnected via FIFO queues according to the scalable computational process network model described in Section 2.3.

4.3.1 Previous Implementation

A previous implementation of the 3-D sonar beamforming system currently runs on UltraSPARC-II Enterprise 4000 SMP server from Sun. The system is built on CPN model in which each node containing the beamform-

ing kernels optimized with the VIS extension and interconnected with FIFO queues as shown in Figure 2.5. Benchmark results are produced from multiple 336-MHz UltraSPARC-II processors running in the “real-time” class under the Solaris 2.7 operating system [2].

Performance analysis The measured performance in Tables 4.1 and 4.2 shows near linear scalability in the horizontal kernel while the vertical kernel executes less efficiently as the number of the threads increases. Although the three horizontal kernels combined require more than seven times the MFLOPS than the single vertical kernel, the vertical kernel requires larger memory bandwidth. This is because the algorithm for the vertical beamformer uses streams of new samples only once to calculate three sets of new data while horizontal beamformer reuses the same set of data a number of times equal to the number of beams (61 beams). Therefore, vertical beamformer has to continually access the main memory while horizontal beamformer can reuse the data in the L2 cache after the first access to main memory.

When examining the execution of the kernel in terms of system architecture, a single vertical beamforming thread can execute on single CM board without having to access the Gigaplane, but multiple threads running on more than two processors need to access the Gigaplane to communicate. Since inter-board memory access is slower due to the interconnection model, it is not surprising to see a more drastic slowdown with the vertical than the horizontal beamforming kernel due to memory latency.

Implementation	Time (s)	MFLOPS	Speedup	Percent Utilization
kernel	3.899	825.1	(1.000)	(100.00%)
2 threads	2.548	1262.4	1.530	76.50%
4 threads	1.303	2469.7	2.993	74.83%
6 threads	1.679	1915.5	2.322	58.04%
7 threads	1.596	2015.1	2.442	61.06%

Table 4.3: Horizontal Beamforming Benchmark using 333-MHz Synergy Quad G4 VSS4 board

Implementation	Time (s)	MFLOPS	Speedup	Percent Utilization
kernel	3.912	321.7	(1.000)	(100.00%)
2 threads	4.303	292.4	0.909	45.45%
4 threads	4.472	281.4	0.875	21.87%
6 threads	4.341	289.9	0.901	22.53%
8 threads	4.446	283.0	0.880	21.99%

Table 4.4: Vertical Beamforming Benchmark using 333-MHz Synergy Quad G4 VSS4 board

4.3.2 Implementation using AltiVec

My implementation using AltiVec extension runs on Synergy Quad PowerPC G4 board with four 333-MHz G4 processors running LinuxPPC operating system. As with the previously implemented system on the Sun, the software is built on CPN programming model with nodes and FIFO queues. Benchmark for the scalable beamforming kernels is compiled using GCC compiler with data prefetch and unaligned sample blocks for the best performance as reported from the single thread kernel in Chapter 3.

Performance analysis As shown on Table 4.3, the horizontal beamformer scales linearly at about 75% per processor from two to four threads. When number of threads exceeds the number of available processors, the performance

decreases to about 60% per processor. When there are 1-4 threads, each processor can dedicate all of its computation power to a single thread. When there are more threads than the number of processors, more than one thread has to run on one or more of the processors. Such distribution of tasks causes process context switches to occur in processors with more than one thread. A context switch between two threads can be very expensive, especially when the scheduler decides to move the thread from one processor to another. When the thread is moved from one processor to another, all of the data modified in the L2 cache of the previous processor has to be written back to the main memory and when the data is attempts to load the data, cache misses occur.

Table 4.5 shows unexpectedly disappointing benchmark numbers for the scalable vertical beamformer on the PowerPC. Figure 3.3 shows the performance of the vertical kernel running on LinuxPPC machine with single G4 processor. The peak benchmark number under single processor machine shows the kernel running at about 1.4 operations per second. Using this benchmark value, I estimate that the single threaded kernel on the Synergy board would run at about 450 MFLOPS per thread. However, the peak MFLOPS that the Synergy board sustains less than 320 MFLOPS total.

Moreover, the new benchmark indicates performance decrease of 10% when multiple threads are executed. This observation suggests that the performance bottleneck is related to the common resources of all four processors on the Synergy board. The obvious shared resources are the 60X Bus and the memory. Using a simple micro-benchmark of a sequential byte copy function reveals that the total memory bandwidth available for *all of the processors* on

Implementation	Time (s)	MFLOPS	Speedup	Percent Utilization
kernel	1.123	1120.4	(1.000)	(100.00%)
2 threads	0.941	1336.6	1.193	59.65%
4 threads	0.645	1949.5	1.740	43.50%
6 threads	0.643	1956.7	1.745	43.66%
17 threads	0.448	2807.1	2.505	62.64%

Table 4.5: Test benchmark for the vertical kernel on the PowerPC board when reusing the cached data

Synergy board is 110.345 MB/s for read and 156.098 MB/s for write, while Sun independently sustained 159.901 MB/s for read and 189.395 MB/s for write for *each processor*.

According to the implementation algorithm, the vertical beamformer requires 0.333 bytes of data reads and 0.200 bytes of data writes for each operation. For example, to produce a performance of 1000 MFLOPS, the kernel needs 333 MB/s memory read and 200 MB/s memory write bandwidth. Using the 110.345 MB/s memory read bandwidth, I calculate the possible performance of the vertical beamformer to be about 331 MFLOPS, which is about what Synergy board yielded. To verify the memory bandwidth bottleneck, a new test benchmark in Table 4.5 is generated from the modified kernel which strictly uses the level-2 cache instead of the main memory. Although performance of multiple threads does not scale linearly, the vertical kernel scales to factor of 2.5 over four processors as the number of threads is increased.

4.4 Performance and Memory

For many years, commodity processor speeds have been faster than commodity memory access rates. With recent introduction of processors with

memory bandwidth hungry SIMD vector instruction sets, the difference in speed has become larger than ever. In real-time data streaming applications, memory bandwidth usually becomes the bottleneck of the system performance.

4.4.1 Memory Configuration

In vector processing systems, memory bandwidth can considerably increase with the interleaving memory access [22]. The concurrent memory access from multiple processors can cause memory bank conflict disabling the memory to utilize the full bandwidth. In some applications, these conflicts make the memory bandwidth become the bottleneck of the software performance. To minimize the memory contention between processors, few design techniques are implemented in current SMP systems.

One method is to have large level 2 (L2) caches that are locally in connected to each processor. Thus, when the processor accesses the memory, the local copy of the data can be stored in the L2 cache. As long as the data is not accessed by the other processors, it can be used multiple times without having to access the main memory. Another solution to memory contention problem is to simply increase the clock speed or the number of memory access ports to service multiple processors. In many shared memory SMP systems, a combination of these design techniques are implemented [23].

With careful analysis of the memory access profile, one can also decrease the undesirable effects of memory contention by organizing each run-time process to use data that are spatially close and yet independent from the data used by other processes. Such methods may become necessary for processes

that require high memory bandwidth. However, the analysis is often tedious and differs from one architecture to another. For some kernels, analysis simply cannot yield improvements because of the predetermined computation algorithm or data structure. In such instances, the architecture of the hardware system solely determines the performance [24].

4.4.2 Performance Bottleneck

In vertical kernel benchmark on Sun UltraSPARC, I observe that the UltraSPARC-II chip performs at about 311.7 MFLOPS without saturating its memory bandwidth. Therefore, as the number of threads increases, the performance scales. From my benchmarking, I believe that the PowerPC G4 processor with AltiVec can potentially outperform UltraSPARC-II processor by a factor of four for the same clock rate. However, the interconnection between the processors and memory on the Synergy board does not provide enough memory bandwidth to keep up with the requests from the bandwidth hungry beamforming kernels. For such applications, the performance will naturally increase proportionally to the number of required memory requests from the processors.

4.5 Conclusion

In this chapter, I describe three different types of multiprocessing platforms — symmetric multiprocessor (SMP), clusters of computers, and combinations of both. For 2 to 16 processors, SMP is popular. Clusters of computers are usually implemented for larger systems. In these systems, performance is

determined by processor computation power and the interconnection between the processors and memory.

This chapter also examines the impact of processor and interconnection performance of two SMP servers running real-time sonar beamforming kernels. The interconnection architecture and the performance benchmarks are used to compare Sun UltraSPARC-II Enterprise server with Solaris 2.7 SMP operating system and Synergy Quad PowerPC G4 server with LinuxPPC 2.2 SMP operating system.

Analysis of the benchmarks and machine specifications shows that the PowerPC G4 processor is capable of outperforming Sun UltraSPARC-II processor by a factor of four in sonar beamforming kernels. However, as the number of threads grow, the performance of the Synergy board does not scale up due to insufficient memory bandwidth, whereas the Sun server implementation scales with little overhead. These results reaffirm the importance of interconnection technology in the design of multiprocessor systems.

5. Conclusion

5.1 Beamforming Kernel

I initially program horizontal and vertical beamforming kernels for a 3-D sonar beamforming system on the PowerPC G4 processor using version 2.95 of the GNU C compiler. Then I 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 reduce the number of permutations needed to align the data for more efficient use of the SIMD instructions.

I evaluate the performance of my beamforming kernels on the PowerPC G4 and previously developed beamforming kernels on the UltraSPARC-II:

- with and without hand-coded 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 back-end 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 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 an array size was a multiple of 8192 bytes.

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 speedup was approximately 1.56 times that of UltraSPARC whereas the horizontal beamformer was about 1.83 times.

The benchmark measurements of the kernel indicates the possibility of running the 4-GFLOP beamforming system on six 450-MHz PowerPC G4 processors replacing the current system which requires ten 450-MHz UltraSPARC II processors. However, my 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, I predict a successful implementation in the near future.

5.2 Scalable System

In implementing scalable beamformer on multiprocessing systems, I define different types of multiprocessing. I define basic three types of multiprocessing environments. The most popular platform is symmetric multiprocessing. An SMP platform usually has a large shared memory which serves relatively small number of processors. Another type of platform is clusters of computers. Each computer, consisting of one processor with its own independent memory and I/O, is connected with fast network interface with high-bandwidth and low-latency. The computers communicate necessary information using protocols for passing messages, e.g. Message Passing Interface (MPI). The third type of multiprocessing is all of the combinations of previous two groups of environments. Most SMP computers on the market have interfaces that easily allow constructions of clusters of SMPs.

Designing powerful multiprocessing platforms not only depends on high performance processors but also on the interconnection model(s). Important trade-off factors in selecting an interconnection are cost and bandwidth. One can construct fast interconnection between shared memory and the processor by using crossbar technology. However, as the number of processors increases, the cost of the technology can grow exponentially. With fast interconnection, scalability of the multiprocessor performance becomes efficient. Less expensive option of interconnection is based on the “dance-hall” model. Interconnection using this model scale the size of the memory proportional to the number of memory, but the interconnection latency uniformly increase. Therefore, the memory access from each processor becomes computationally

expensive. Most shared memory systems today use either model or a combination of both models in the interconnection. Then there is a multiprocessing environment based on distributed or NUMA memory model. Most clusters of computer systems are based on this model, where each independent computers pass messages through a high-speed network for communication.

I look at the system architecture of the two SMP servers on which the beamforming system is implemented. The first beamforming system by Allen and Evans [2] is on a Sun UltraSPARC-II Enterprise server with the Solaris operating system version 2.7. I modify the kernels to use Altivec to run on Synergy Quad PowerPC G4 server with LinuxPPC SMP operating system version 2.2. The Sun server implementation has 16 UltraSPARC-II processors with clock rate of 336 MHz arranged in combinations of the crossbar and the “dance-hall” based interconnection to give high bandwidth at a lower cost. The Synergy PowerPC server uses the simple “dance-hall” based interconnection. Given more powerful and bandwidth hungry processors, the slower and smaller interconnection is not well-suited for many signal processing applications with performance that needs to scale with the memory bandwidth.

Benchmark analysis predicts the linear scaling of the beamformer on Sun server, but it reveals the mismatch of the processor and the available interconnection bandwidth on Synergy server for the sonar beamforming application. The result from memory benchmarks shows that the memory bandwidth is the performance bottleneck on the Synergy server. Furthermore, modified beamforming kernel suggests possible performance of the Altivec extension to be four times as powerful as the VIS extension if each processors had sixteen

times the currently available memory bandwidth on the board.

On the Quad Synergy board, I discover that the performance of highly scalable applications like the sonar beamformer is not only dependent on the computation power of the processor but on the performance of the interconnection. I suggest couple of different ways to efficiently use the available bandwidth. In designing multiprocessor with bandwidth hungry SIMD processors, faster interconnection with high bandwidth would be beneficial for the successful implementation of various signal processing algorithms.

5.3 Possible Future Implementation

The original sonar beamforming algorithm is used for all the implementations referred in the report. The vertical beamformer requires 480 MFLOPS of computation while its results are passed through FIFOs to three horizontal beamformers, which require 1200 MFLOPS each. Using the benchmarks for the scalable kernels, I can extrapolate that the sonar beamforming system, given the enough memory bandwidth, can possibly be implemented on six G4 processors running at 450 MHz.

Motorola released a new PowerPC G4 processor on January 9, 2001, which can have clock rates as high as 733 MHz. If I can make an assumption that the new processor performance increase proportional to the G4 processors used in my implementations, then the entire beamforming algorithm maybe possible to implement using only three processors.

BIBLIOGRAPHY

- [1] B. L. Evans, “EE382C Embedded Software Systems Lecture 2: Digital Signal Processors,” *Lecture Notes*, University of Texas at Austin, Fall 2000, http://www.ece.utexas.edu/~bevans/courses/ee382c/lectures/02_signal_processing/
- [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, Mar. 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] J. Bier, “DSP on General Purpose Processors,” *MicroDesign Resources Dinner Meeting Slides*, Berkeley Design Technology, Inc., Jan. 1997.
- [5] Sun Microsystems, “VIS Instruction Set User’s Manual,” <http://solutions.sun.com/embedded/databook/pdf/manuals/805-1394-01.pdf>.
- [6] Motorola Inc., “AltiVec Programming Environment/Interface Manual,” 1998.
- [7] Motorola Inc., “AltiVec Technology.” <http://www.mot.com/AltiVec> and <http://www.altivec.org>.

- [8] 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.
- [9] T. M. Parks, "Bounded Scheduling of Process Networks." *Technical Report UCB/ERL-95-105*, Ph.D. Dissertation, EECS Dept., University of California Berkeley, Berkeley, CA 94720-1770, Dec. 1995.
- [10] G. Kahn, "The semantics of a simple language for parallel programming." *Info. Proc.*, pp. 471-475, Aug. 1974.
- [11] 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.
- [12] Y. H. Cho, D. Brunke, G. E. Allen, and B. L. Evans, "Optimization of Vertical and Horizontal Beamforming Kernels on the PowerPC G4 Processor with AltiVec Technology", *Proc. IEEE Asilomar Conf. on Signals, Systems, and Computers*, Oct. 29-Nov. 1, 2000, vol. 2, pp. 1670-1674, Pacific Grove, CA.
- [13] Y. H. Cho, "Optimized Automatic-Target-Recognition Algorithm on Scalable Myrinet/Field Programmable Array Nodes," *Proc. IEEE Asilomar Conf. on Signals, Systems, and Computers*, Pacific Grove, CA, Nov. 2000.
- [14] N. M. Amato, J. Perdue, A. Pietracaprina, G. Pucci, and M. Mathis, "Predicting Performance on SMPs. A Case Study: The SGI Power Challenge,"

Proc. of the 14th IEEE Int. Parallel and Distributed Processing Symp.,
May 2000.

- [15] Synergy Microsystems, "SBC-VSS4 VME Single Board,"
<http://www.synergymicro.com/VME/VSS4.html>.
- [16] D. E. Culler, "Convergence of Parallel Architecture," *Course Lecture Note: CS258*, University of California at Berkeley, CS Division, Berkeley, CA, Spring 1999.
- [17] Sun Microsystems, "Specification for UltraSPARC-II Enterprise 4500 server," <http://www.sun.com/servers/midrange/e4500/specs.html>
- [18] Sun Microsystems, "STP2202BGA DSC Datasheet," <http://www.sun.com/microelectronics/databook/datasheets.html>
- [19] Motorola Inc., "MPC7400 RISC Microprocessor User's Manual." <http://www.mot.com>.
- [20] Motorola Inc., "Designing G4 Systems," Application Notes, <http://www.mot.com>.
- [21] C. D. Bryant, M. J. Garcia, B. K. Reynolds, L. A. Weber, G. E. Wilson, "The Performance and PowerPC Platform Specification Implementation of The MPC106 Chipset," *Proc. of IEEE Compton*, pp. 132-139, Los Alamitos, CA, 1996.
- [22] 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.

- [23] G. E. Blelloch, P. B. Gibbons, Y. Matias, and M. Zargha, "Accounting for Memory Bank Contention and Delay in High-Bandwidth Multiprocessors," *IEEE Trans. on Parallel and Distributed Systems*, vol. 8, no. 9, pp. 943-951, Sept. 1997.
- [24] C. Fricker, "On Memory Contention Problems in Vector Multiprocessors," *IEEE Trans. on Computers*, vol. 44, no. 1, pp. 92-105, Jan. 1995.

VITA

Young Hyun Cho was born in Seoul, South Korea on February 25, 1974, the son of Sorae Cho and Kyung Cho. After completing his work at Chatsworth High School, Chatsworth, California in 1992, he entered University of California at Berkeley in Berkeley, California. While he was attending Berkeley, he worked as Research Assistant for the Networks of Workstations (NOW) group in the Computer Science department. He also taught Computer Architecture course as a Teachers Assistant for two semesters. After he graduated from Berkeley with his Computer Science degree in May of 1996, he began to work as a Computer Engineer at Myricom, Inc. His accomplishments in the company included research and development projects such as Automatic Target Recognition and Myricom Two-Level Computers, contracted under DARPA, as well as commercial products such as Myrinet to Optical fiber network interface and 32 port Myrinet Switch. On August of 1999, he took a leave of absence from Myricom to continue his education in the University of Texas at Austin.

Permanent address: 10849 Lurline Avenue
Chatsworth, CA 91311

This report was typeset¹ with L^AT_EX by the author.

¹L^AT_EX document preparation system was developed by Leslie Lamport as a special version of Donald Knuth's T_EX program for computer typesetting. T_EX is a trademark of the American Mathematical Society. The L^AT_EX macro package for The University of Texas at Austin report format was written by Khe-Sing The.