# Performance Evaluation and Benchmarking of Native Signal Processing

Deependra Talla and Lizy K. John

Laboratory for Computer Architecture,
Department of Electrical and Computer Engineering,
The University of Texas at Austin, Austin, TX, USA 78712
deepu@ece.utexas.edu, ljohn@ece.utexas.edu

**Abstract**. DSP processor growth is phenomenal and continues to grow rapidly, but general-purpose microprocessors have entered the multimedia and signal processing oriented stream by adding DSP functionality to the instruction set and also providing optimized assembly libraries. In this paper, we compare the performance of a general-purpose processor (Pentium II with MMX) versus a DSP processor (TI's C62xx) by evaluating the effectiveness of VLIW style parallelism in the C62xx versus the SIMD parallelism in MMX on the Intel P6 microarchitecture. We also compare the execution speed of reliable, standard, and efficient C code with respect to the signal processing library (from Intel) by benchmarking a suite of DSP algorithms. We observed that the C62xx exhibited a speedup (ratio of execution clock cycles) ranging from 1.3 up to 4.0 over the Pentium II, and the NSP libraries had a speedup ranging from 0.8 to over 10 over the C code.

## 1 Introduction

Digital Signal Processing (DSP) and multimedia applications are ubiquitous and the demand for such capabilities on general-purpose processors is increasing. Multimedia and signal processing oriented workloads have been projected to be a dominant segment of future workloads [1]. Traditionally, DSP applications were being performed solely on a dedicated DSP processor that is specialized to do the intense number crunching tasks presented by various signal processing algorithms. However, there are several reasons as to why one might want to use a general-purpose processor instead of a DSP processor. The main driving force behind this initiative is the fact that instead of using a DSP processor for signal processing and a general-purpose processor for other tasks; space, cost and overall system complexity can be improved by moving the signal processing onto the general-purpose processor and eliminating the DSP processor. It was also found that in certain applications general-purpose processors outperform DSP processors [2].

General-purpose microprocessors have entered the multimedia and signal processing oriented stream by adding DSP functionality to the instruction set and also providing optimized assembly libraries. Perhaps, one of the earliest and most visible example of this approach to signal processing was Intel's Native Signal Processing

(NSP) initiative with the introduction of the MMX (commonly dubbed as MultiMedia eXtension) set to the traditional x86 architectures. In addition to MMX instructions, Intel also provides optimized C-callable assembly libraries for signal and image processing [3]. Such media processing capabilities are foreseen to make profound changes in the use of current microprocessors [4]. DSP and media processing applications have been known to be structured and regular. The approach adopted by NSP extensions such as MMX has been to exploit the available data parallelism using SIMD (single-instruction multiple-data) techniques. On the other hand, high performance DSP processors such as the Texas Instrument's TMS320C6201 utilize a VLIW (very long instruction word) architecture.

Previous efforts have analyzed the benefits of multimedia extensions on general-purpose processors [5][6][7]. Performance benefits of MMX instructions over non-MMX code for a Pentium processor (P5 microarchitecture, with MMX) were evaluated in [5]. Effect of Sun's VIS (visual instruction set) on a superscalar processor was measured in [6]. A number of DSP and general-purpose processors have been benchmarked by BDTI [8][9], however details on performance of individual benchmarks are not publicly available. The objectives of this paper are two-fold. We compare the performance of a general-purpose processor versus a DSP processor for several DSP kernels, followed by evaluating the effectiveness of Intel's NSP library by benchmarking a suite of DSP algorithms.

## 2 Performance Evaluation Methodology

To measure the effectiveness of performing signal processing on general-purpose processors, we chose the highest performance member of the Intel x86 architecture family – P6 microarchitecture (a Pentium II processor with MMX) [10]. It has a three-way superscalar architecture with dynamic execution. As a high performance DSP processor, Texas Instrument's TMS320C6201 – a 32-bit fixed-point, eight-way VLIW was evaluated [11].

DSP performance evaluation is still largely done by evaluation of kernels. We assembled a suite of benchmarks that reflect a majority of the DSP routines used in the real world. Our benchmark suite consists of the following DSP kernels:

- Dot product, autocorrelation, convolution, and vector-matrix operations
- Filtering (FIR – finite impulse response and IIR – infinite impulse response)
- Transforms (FFT – fast Fourier transform, DCT – discrete cosine transform, and DWT – discrete wavelet transform)
- Window functions (Hamming, Hanning, Kaiser, Bartlett, and Blackman)
- Signal generation functions (Gaussian, Triangle, and Tone)

Our performance evaluation utilized both C and assembly code versions of the benchmarks. Assembly code versions of the benchmarks were obtained from Intel and TI libraries, while the C versions were obtained from industry standard sources [12][13]. We used Microsoft Visual C++ version 5.0 and Intel C/C++ compiler for the Pentium II processor and the C62xx compiler for the C6201 processor. Maximum optimization was enabled during compilation in all cases for faster execution times.

While comparing the Pentium II and the C62xx, each of our benchmarks has four versions – Pentium II C code, Pentium II optimized assembly code with MMX, C62xx C code, and C62xx optimized assembly code. While evaluating the NSP libraries on the Pentium II processor, each benchmark has the following four versions – C code using 'float' data type, assembly code using 'float', assembly using 'double' and assembly using 'short' data types. It is in the 'short' data type that MMX instructions are used to take advantage of data parallelism. We quantified speedup as the ratio of the execution clock cycles with Pentium II C code as the baseline for the general-purpose versus DSP processors comparison and C code using 'float' data type as the baseline for measuring the effectiveness of NSP on the Pentium II processor.

We used VTune [10], performance counters on the Pentium II, and the C62xx stand-alone simulator [11] for measuring the execution times and profiling the instruction mixes of the algorithms on the Pentium II and C62xx. VTune is Intel's performance analysis tool that can be used to get complete instruction mixes of the code. The performance counters present in the Intel Pentium line of processors are used to gather the execution cycle count. Similarly, the stand-alone simulator of the C62xx is useful for quick simulation of small pieces of code - specifically to gather cycle count information.

## 3 Results

Pentium II processor (with MMX) and C62xx are compared utilizing C code and optimized assembly code for dot product, autocorrelation, FIR, IIR, and FFT. Not all benchmarks mentioned earlier were evaluated because optimized assembly code for C62xx was not readily available. Figure 1 below shows the results obtained for the comparison of the Pentium II and the C62xx processor.

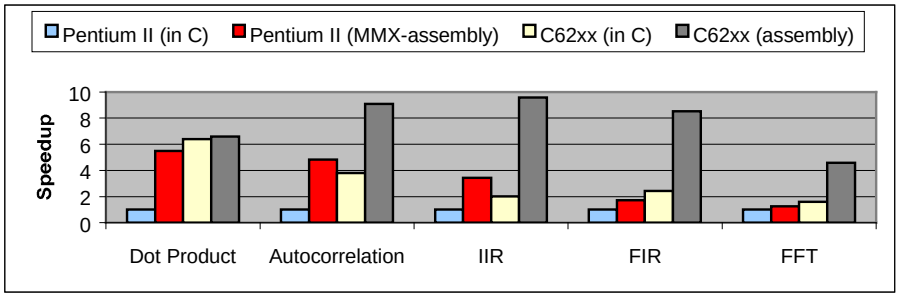


Figure. 1. Comparison of the DSP versus the general-purpose processor

Based on the execution cycle counts, the C62xx DSP is superior to the Pentium II for the DSP kernels considered. The C62xx has eight functional units (six ALU's and two load-store units) that allows it to simultaneously operate on multiple data elements. In almost all of the above benchmarks, the maximum instruction bandwidth of eight is being achieved in the case of the C62xx assembly. However, the compiler-generated code is not as efficient as the optimized assembly code. In the case of the Pentium II, the SIMD parallelism is 4 for 16-bit data. Speedup for Pentium II MMX

assembly is over 4 in the case of dot product and autocorrelation. This is because in the case of C code, the integer multiply takes 10 cycles on a Pentium II, while the MMX multiply and accumulate operation takes only 3 cycles. Both the dot product and autocorrelation involve a lot of multiply and accumulates. To make a fair comparison between the two processors, we pre-load the caches in the case of the Pentium II to eliminate cold starts.

An important aspect of the above comparison is that we are measuring the absolute number of execution clock cycles, and not the execution time. The fastest C62xx currently runs at 250 MHz, while the Pentium II processor can run at speeds up to 450 MHz. Other aspects worth mentioning are the price, power consumption, interrupt service latencies, and sizes of the two processors. The C62xx is approximately 1/15 in price, 1/3 in power consumption, 1/10 in interrupt service latency, and 1/100 in volume. Table 1 shows the effectiveness of Intel's NSP libraries. The C version uses a single-precision floating-point data type. The NSP 'float' is single-precision floating-point data, 'double' is double precision floating point, and 'short' is 16-bit fixed-point data using MMX.

| Benchmark | C version | NSP float | NSP double | NSP Short |
|---|---|---|---|---|
| *FFT* | 1 | 2.57 | 1.77 | 2.4 |
| *DCT* | 1 | 3.51 | 2.3 | 7.21 |
| *DWT* | 1 | 2.3 | 1.7 | 1.32 |
| *FIR* | 1 | 0.81 | 0.81 | 0.86 |
| *IIR* | 1 | 1.7 | 1.53 | 2.03 |
| *Hamming* | 1 | 2.44 | 1.66 | 3.84 |
| *Hanning* | 1 | 2.59 | 1.72 | 3.9 |
| *Kaiser* | 1 | 0.62 | 0.73 | 0.42 |
| *Bartlett* | 1 | 3.04 | 1.8 | 5.15 |
| *Blackman* | 1 | 1.87 | 1.51 | 3.38 |
| *Gaussian* | 1 | 0.99 | 0.94 | 0.72 |
| *Triangle* | 1 | 0.77 | 0.47 | 0.61 |
| *Tone* | 1 | 10.3 | 10.3 | 20.75 |
| *Vector/Matrix* | 1 | - | - | 2.47 |
| *Convolution* | 1 | 2.29 | 2.4 | - |

Table. 1. Speedup for NSP library code over C code

The 'short' data type version of the benchmarks has the potential to use MMX instructions to exploit data parallelism. It is interesting to note that the 'short' data versions for DWT, Gaussian, and Kaiser are slower than the 'float' versions. We believe this is true because that the library code for 'short' does an internal conversion to 'float' as was also observed in [5]. Several benchmarks used MMX instructions to achieve the speedup – vector/matrix (91% of dynamic instructions were MMX related instructions), FIR (10%), IIR (73%), FFT (3%), DCT (57%), Blackman (87%), Bartlett (80%), Hamming (86%), and Hanning (86%). In general as expected, the 'float' version of the benchmarks are faster than the 'double' versions, and the 'short' versions (that use MMX) are faster than all other versions.

## 4 Conclusions

In this paper, we evaluated the effectiveness of native signal processing on a Pentium II processor. First, we compared several DSP kernels on the Pentium II with respect to the C62xx DSP processor followed by a comparison of efficient C code with Intel's NSP libraries. Our major observations are:

- TI's optimized C62xx assembly code yielded a speedup of 1.2 in the case of the dot product to over 4.0 in the case of the FIR filter over optimized MMX assembly code on a Pentium II processor.
- Compiler generated code for the C62xx showed a speedup ranging from 1.6 to 6.0 over the Pentium II processor.
- NSP libraries are faster than using standard C code for many of the benchmarks. Speedup ranged from 0.8 to over 10.0. Some benchmarks like triangle wave generation and Kaiser windowing function were slower for the NSP libraries over the C code. Such anomalies make it a difficult task to predict if using NSP assembly code is faster than standard C code.

**Acknowledgements**

This work is supported in part by the State of Texas Advanced Technology Program grant #403, the National Science Foundation under Grants CCR-9796098 and EIA-9807112, and by Dell, Intel, Microsoft, and IBM.

**References**

1. C.E. Kozyrakis and D.A. Patterson, "A New Direction for Computer Architecture Research", IEEE Computer Magazine, pp. 24-32, Nov. 1998.
2. G. Blalock, "Microprocessors Outperform DSPs 2:1", MicroProcessor Report, vol. 10, no. 17, pp. 1-4, Dec. 1995.
3. Intel, "Performance Library Suite". http://developer.intel.com/vtune/perflibst/index.htm.
4. K. Diefendorff and P.K. Dubey, "How Multimedia Workloads Will Change Processor Design", IEEE Computer Magazine, pp. 43-45, Sep. 1997.
5. R. Bhargava, L. John, B. Evans, and R. Radhakrishnan, "Evaluating MMX Technology Using DSP and Multimedia Applications", IEEE Micro-31, pp. 37-46, Dec. 1998.
6. P. Ranganathan, *et. al*, "Performance of Image and Video Processing with General-purpose Processors and Media ISA extensions", ISCA-26, pp. 124-135, May 1999.
7. R.B. Lee, "Multimedia Extensions For General-Purpose Processors", Proc. IEEE Workshop on Signal Processing Systems, pp. 9-23, Nov. 1997.
8. G. Blalock, "The BDTIMark: A Measure of DSP Execution Speed", 1997. White Paper from Berkeley Design Technology, Inc. http://www.bdti.com/articles/wtpaper.htm.
9. J. Bier and J. Eyre, "Independent DSP Benchmarking: Methodologies and Latest Results", Proc. ICSPAT, Toronto, Sep. 1998.
10. Intel Literature, P6 architecture developer's manuals. http:/developer.intel.com.
11. TMS320C6x Optimizing C Compiler User's guide, Texas Instruments Inc. Lite. Num. SPRU187B.
12. Siglib version 2.4, Numerix Co Ltd. http://www.numerix.co.uk.
13. P.M. Embree, "C Algorithms for Real-Time DSP", NJ: Prentice Hall, 1995.
14. V. Zivojnovic, J. Martinez, C. Schläger and H. Meyr. "DSPstone: A DSP-Oriented Benchmarking Methodology". Proc. of ICSPAT'94 - Dallas, Oct. 1994.
15. EDN Embedded Microprocessor Benchmark Consortium, http://www.eembc.org.