Characterization of Native Signal Processing Extensions

Jason Law
March 25, 2002

Abstract

Soon if not already, multimedia applications such as those that deal with the creation, processing, and communication of digital images, digital audio, and digital video are expected to become one of the dominant computing workloads [6]. Most modern general-purpose processors employ a set of instruction extensions to their architecture to enhance the performance of digital signal processing (DSP) and multimedia applications. It is now possible to perform many tasks, which have in the past required the use of a DSP, using only a general-purpose processor. This is known as Native Signal Processing (NSP). Traditionally, creating code that takes full advantage of NSP extensions is a difficult task that involves writing hand-tuned assembly code [1]. Evaluating the effectiveness of NSP extensions is therefore complicated since it is not easy to use popular benchmarks such as MediaBench, MiBench, or the test suites developed by BDTI and EEMBC. My objective in this survey is to summarize the NSP extensions to the x86 architecture, discuss the feasibility of using existing embedded benchmarks to characterize NSP extensions, discuss previous work in this area, and discuss possible methods to characterize the latest NSP extension to the x86 architecture, Intel's Streaming SIMD Extensions 2 (SSE2).

Native Signal Processing Background:

Most modern general-purpose processor architectures now include some sort of NSP extensions. MAX-1 (Multimedia Acceleration extensions) for HP's PA-RISC instruction set architecture (ISA) was the first NSP extension introduced in 1994. Sun then added to its workstations the Visual Instruction Set (VIS). HP also released MAX-2, an NSP extension to its PA-RISC 2.0 processor architecture [7]. During 1996 three NSP

extensions were released: Silicon Graphics released MDMX for its MIPS ISA, Digital released MVI for its Alpha ISA, and Intel released MMX (MultiMedia eXtensions) for its x86 ISA [4]. The MMX extensions on the Pentium and Pentium II included only integer SIMD instructions so Intel quickly released another set of extensions called SSE (Streaming SIMD Extensions) for the Pentium III that included many floating-point instructions [5]. In 1998, AMD introduced 3DNow!, a set of extensions to the x86 ISA that included floating-point instructions and was intended to complement MMX. Also in 1998, Motorola introduced the Altivec extensions to their PowerPC architecture. In November of 2000, Intel released the latest extension to its x86 ISA, Streaming SIMD Extensions 2 (SSE2).

Introduced with the Pentium 4, SSE2 represents a complete overhaul of all previous NSP extensions to the x86 architecture as well as the addition of several new instructions. SSE2 is a substantial evolution of SSE and MMX [5]. SSE2 enables the execution of two double-precision 64-bit floating-point operations per clock; while the original SSE only allowed four 32-bit single-precision operations. Engineering and scientific applications should benefit from this extra precision. SSE2 also extends the original MMX integer instructions from 64 bits to 128 bits. This allows two 64-bit integer operations to be performed at once. Also included in SSE2 are new cacheablity control instructions that allow extensive control of the data cache to minimize cache pollution and new data prefetch instructions to exploit concurrency between the memory and execution pipelines and to hide memory latency. Along with SSE2, Intel introduced several tools for the creation of programs that successfully utilize SSE2's enhancements.

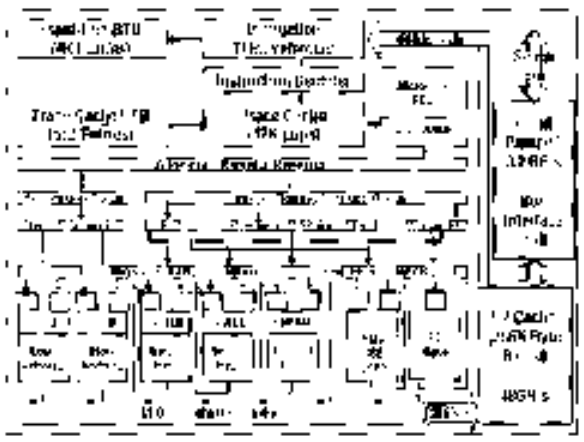

Figure 1: Pentium 4 Microarchitecture

These include compiler intrinsics (macro like functions) for the Intel C/C++ compiler, a vector class library in C++, and extended compiler optimizations for the Intel C/C++ compiler. The intention behind the development of these tools is to remove the necessity of writing applications or kernels that effectively use the NSP extensions in hand coded assembly. Using these tools it should be easier to characterize SSE2 than it was to characterize MMX and SSE.

Native Signal Processing Pitfalls:

There are several drawbacks to using NSP extensions. NSP extensions often use SIMD semantics that require "packed" data types to expose data level parallelism. An overhead exists in packing and unpacking the data into these SIMD data types. Existing applications must be rewritten to explicitly take advantage of the NSP extensions. Further complicating this issue, NSP extensions vary widely across platforms, forcing the application developer to chose which platform to optimize for, or to spend a large amount of time optimizing for all platforms. [1, 8] Since the datatypes and operations introduced

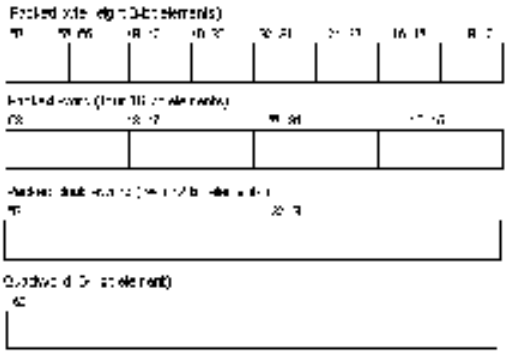

Figure 2: An Example of SIMD Packed Datatypes

by NSP extensions are not standardized across architectures, code generated for a specific extension is not portable. Currently, code that takes advantage of NSP extensions is either written by hand in assembly, as is the case for optimized kernels, or it is generated by macro like compiler intrinsics that is similar to function inlining [8].

Embedded Benchmarks:

Many benchmark suites for the evaluation of embedded and signal-processing systems exist. Most of these suites can be placed into two categories: those created by an independent company or a representative organization and developed to aid system developers in choosing a processor for their system, or those created by academia to fill a perceived void and to aid in research. The suites produced by EEMBC (EDN Embedded Microprocessor Benchmark Consortium) and by BDTI (Berkeley Design Technology, Inc.) are examples of the former. MediaBench and MiBench are examples of the latter. All of these benchmark suites attempt to simulate a real world workload that an embedded processor might be expected to handle. A significant question is how does a

typical embedded processor workload differ from a NSP workload on a general-purpose processor.

EEMBC (pronounced "embassy") was formed in 1997 to develop meaningful performance benchmarks for processors and compilers in embedded applications. The consortium consists of 45 semiconductor, intellectual property, compiler, and real-time operating systems (RTOS) companies [11]. The EEMBC suite consists of applications and kernels targeted at five major application areas: telecommunications, consumer, networking, office automation, and automotive/industrial. The suite is fairly simple to obtain and use; however, one must be a member of EEMBC first. A major drawback to using the EEMBC suite to evaluate SSE2 is that currently very few algorithms in the EEMBC suite include floating point calculations. This is because historically few embedded applications needed floating-point computations, and many of the processors EEMBC benchmarks have no floating-point capability.

BDTI is an independent company that analyzes digital signal processors. For a large fee they will allow you to see their results. BDTI has focused exclusively on DSPs since 1991. BDTI's benchmark suite consists of twelve kernels that execute commonly used basic blocks such as several forms of FIR's, an FFT, an IIR, a vector dot product, and a viterbi decoder. BDTI views applications to be overly complicated to use as benchmarks for DSPs [13]. This view limits the usefulness of BDTI's benchmark suite for evaluating NSP on general-purpose processors.

MediaBench is a benchmark suite developed at UCLA to fill the gap caused by a focus on instruction level parallelism between compilers and embedded application developers [2]. It consists of 19 publicly available applications coded in a high level

language such as C. Examples include: JPEG image compression and decompression, MPEG2 encoding and decoding, G.721 voice compression, PGP encryption, Ghostscript file I/O, and Mesa (a clone of the 3D graphics library OpenGL). These applications are similar to those of the EEMBC suite, but they are not grouped into categories, as are the applications in the EEMBC suite. [2] showed that the workload produced by SPECint differed significantly from that produced by MediaBench. Similarly to EEMBC's suite, MediaBench does not include floating-point calculations and this poses problems if MediaBench alone is used to evaluate SSE2.

MiBench is an embedded benchmark suite put together at the University of Michigan at Ann Arbor. It is modeled after EEMBC's benchmark suite, in that the applications are grouped into areas (MiBench has six areas unlike EEMBC's five): Automotive and Industrial Control, Consumer Devices, Office Automation, Network, Security, and Telecommunication [9]. Each area is composed of several applications that are representative of the workload an embedded microprocessor in such a system would encounter. The advantage of using MiBench over EEMBC's is that MiBench is freely available over the Internet. There are some issues with the way MiBench uses its applications; for the most part the applications are each run once and then terminate. This causes problems in that measurements are often hard to take for programs with short execution times. EEMBC's suite overcomes this by running its applications in a loop for a fixed amount of time. Also, MiBench has no test harness. Each application must be run manually. EEMBC's suite has a well-written and fully developed test harness. In this case you get what you pay for.

Model of Computation:

Many models of computation appear in native signal processing. Commonly kernels such as FIR filters and FFTs can be implemented as SDF or BDF systems. In fact, the nature of most signal processing algorithms allow the use such simple models. Streaming media systems for streaming video to clients or from servers readily fit this model. Other than the initialization stages, the data manipulation (coding, decoding, encryption, decryption, etc) is easy to implement as an SDF system.

Previous Work:

An analysis of DSP and multimedia kernels and applications using both non-MMX and MMX enhanced code for the Pentium processor was conducted in [1]. Their work centered on the Pentium processor running several signal processing kernels and a few applications. The kernels tested were: fast fourier transform (FFT), finite impulse response filter (FIR), infinite impulse response filter (IIR), and matrix and vector arithmetic. The applications run were: JPEG image compression, an image manipulation program, G.722 speech encoding, and Doppler radar processing. The authors used programs they could find the source code for or could code themselves, so that optimization using MMX instructions could be accomplished. The authors were able to make several valuable conclusions. Among them were that using MMX instructions effectively to produce speedup is a complicated chore even using assembly libraries, especially in large and complicated C applications. They also found that using MMX instructions has the potential to reduce dynamic instructions and micro-operations, but the static code size of applications can be dramatically increased. Static code size is an important factor in designing embedded systems with limited memory. The authors also

experienced problems implementing the IIR filter. Because of a loss of precision in the filter coefficients when converted from floating-point to fixed-point representation, their filter became unstable. MMX did not include any floating-point operations. This is a factor that Intel addressed in SSE and SSE2.

An in-depth quantitative comparison of SIMD, VLIW and Superscalar architectures running signal processing and multimedia applications was conducted in [3]. A sample implementation of each architecture was chosen and several benchmarks were tested. A TI TMS320C62xx DSP, an Intel Pentium II with MMX, and simulations using Simplescalar were chosen to represent each architecture. The kernels tested included: dot product, autocorrelation, and an FIR. The applications tested were: audio effects, G.711 speech encoding, and ADPCM speech compression. The paper finds that for the kernels, both the SIMD and the VLIW architectures showed significant speedup over the baseline model. The VLIW architecture showed even better speedup than the SIMD architecture. The results for the applications were not as impressive. Neither architecture showed a speedup as large as they did for the kernels. The authors found that VLIW code is extremely vulnerable to naïve compilers. The VLIW architecture depends on compile-time scheduling of resources to achieve good performance. It was also found that although MMX enhanced code had a smaller execution time, the CPI of the MMX code was higher than that of the corresponding non-MMX code. But, significant speedup is achieved because each SIMD instruction in MMX does approximately four times as much work as does the non-SIMD instructions. Finally, out-of-order execution and branch prediction as found in modern Superscalar processors was observed to be important for media applications.

References

[1] R. Bhargava, L.K. John, B.L. Evans, R. Radhakrishnan, "Evaluating MMX Technology using DSP and Multimedia Applications," *Proc. 31st Annual ACM/IEEE International Symposium on Microarchitecture*, 1998, pp.37-46.

[2] C. Lee, M. Potkonjak, and W. Mangione-Smith, "MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems", *Proc. 30th Annual ACM/IEEE International Symposium on Microarchitecture*, 1997, pp 330-335.

[3] D. Talla, L. John, V. Lapinski, and B. Evans, "Evaluating Signal Processing and Multimedia Applications on SIMD, VLIW and Superscalar Architectures", *Proc. IEEE International Conference on Computer Design*, 2000.

[4] A. Peleg and U. Weiser, "The MMX Technology Extension to the Intel Architecture," *IEEE Micro*, vol.16, no. 4, pp 42-50, Aug. 1996.

[5] S. Raman, V. Pentkovski, and J. Keshava, "Implementing Streaming SIMD Extensions on the Pentium III Processor," *IEEE Micro*, vol. 33, no. 4, July 2000.

[6] P. Ranganathan, S. Adve, N.P. Jouppi, "Performance of Image and Video Processing with General-Purpose Processors and Media ISA Extensions," *Proc. of the 26th International Symposium on Computer Architecture*, 1999, pp.124-135

[7] R.B. Lee, "Multimedia Extensions for General-Purpose Processors," *IEEE Workshop on Signal Processing Systems*, 1997, pp. 9-23.

[8] T.M. Conte, P.K. Dubey, M.D. Jennings, R.B. Lee, A. Peleg, S. Rathnam, M. Schlansker, P. Song, A. Wolfe, "Challenges to Combining General-Purpose and Multimedia Processors," *IEEE Computer*, Dec. 1997, vol.30, no.12 p.33-7.

[9] M.R. Guthaus, J. Ringenberg, D. Ernst, T.M. Austin, T. Mudge, R.B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," *IEEE Workshop on Workload Characterization*. 2001, pp. 3-14.

[10] "Intel(R) Software College-Streaming SIMD Extensions 2 (SSE2)," *http://developer.intel.com/software/products/college/ia32/sse2*. March 2002.

[11] "EEMBC - Embedded Microprocessor Benchmarking Consortium," *http://www.eembc.org*. March 2002.

[12] "Berkeley Design Technology, Inc. -- Independent DSP Analysis - Optimized DSP Software," *http://www.bdti.com*. March 2002.

[13] BDTi, "Evaluating DSP Processor Performance," *http://www.bdti.com/articles/benchmark_2000.pdf*, March 2002.