Evaluating DSP Processor Performance

Berkeley Design Technology, Inc.

Introduction

Digital signal processing (DSP) is the application of mathematical operations to digitally represented signals. Because digital signals can be processed with cost-effective digital integrated circuits, DSP systems can economically accomplish complex tasks, such as speech synthesis and recognition, that would be difficult or impossible to accomplish using conventional analog techniques.

The market for products using DSP technology, such as wireless communication devices and digital audio appliances, is growing rapidly. Semiconductor manufacturers have responded to this demand by producing an expanding array of DSP processors—microprocessors designed specifically for digital signal processing. Selecting the right DSP processor for an application is a difficult and time-consuming task for DSP system designers. This paper presents a methodology for evaluating DSP processor performance, one of the key considerations in choosing a processor.

DSP Processors

Strictly speaking, the term "DSP processor" applies to any microprocessor that operates on digitally represented signals. In practice, however, the term refers to microprocessors specifically designed to perform digital signal processing. Because most signal processing systems perform complicated mathematical operations on real-time signals, DSP processors use special architectures to accelerate repetitive, numerically intensive calculations. For example, DSP architectures commonly include circuitry to rapidly perform multiply-accumulate operations, which are useful in many signal processing algorithms such as filtering. Also, DSP processors often contain multiple-access memory architectures that allow the processor to simultaneously load multiple operands, such as a data sample and a filter coefficient, in parallel with loading an instruction. In addition, DSP processors often include a variety of special memory addressing modes and program-flow control features designed to accelerate the execution of repetitive operations. Lastly, most DSP processors include specialized on-chip peripherals or I/O interfaces that allow the processor to efficiently interface with other system components, such as analog-to-digital converters and host processors.

This paper focuses on the performance of programmable DSP processors. The performance evaluation methodology we describe can also be applied to measure the DSP performance of general-purpose processors. In particular, engineers may be interested in evaluating the DSP performance of general-purpose processors that have been designed to provide some support for DSP. Many manufacturers of desktop general-purpose processors have enhanced the signal processing capabilities of their processors by adding instructions and hardware accelerators. For example, Intel has developed the MMX and SSE extensions for the Pentium processor line. And in the embedded systems arena, many microcontroller and embedded processor manufacturers have also added DSP functionality to their processors. An example is Hitachi's SH-DSP, which combines microcontroller and DSP features.

What is DSP Processor Performance?

DSP processor performance can be measured in many ways. The most common metric is the time required for a processor to accomplish a defined task. On the other hand, memory usage and energy consumption may be equally—or even more—important in some applications. This paper will examine all three of these metrics, with a focus on execution time.

Measuring DSP processor performance in a way that allows fair comparisons between processor families is difficult. Furthermore, performance measurements are only useful to the typical engineer if the measurements can be related to the requirements of particular applications. To address these challenges, Berkeley Design Technology, Inc. (BDTI) uses a two-fold methodology of algorithm kernel benchmarking and application profiling.

Traditional Approaches to Performance Measurement

MIPS, MOPS, and MACS

Traditional approaches to performance measurement often use very simple metrics to describe processor performance. The most common performance unit, MIPS (millions of instructions per second), is misleading because of the varying amounts of work performed by instructions—a typical instruction on one processor may accomplish far more work than a typical instruction on another processor. This is especially true on DSP processors, which often have highly specialized instruction sets. Without some gauge of the relative efficiency of different instruction sets, MIPS figures are only useful within the context of a single known processor architec-

ture. Similarly, MOPS (millions of operations per second) suffers from a related problem—what counts as an operation and the number of operations needed to accomplish useful work vary greatly from processor to processor.

Other commonly quoted performance measurement units can also be misleading. Because multiply-accumulate operations are central to many DSP algorithms, such as FIR filtering, correlation, and dot-products, some processor manufacturers quote performance in MACS (multiply-accumulates per second). However, DSP applications involve many operations other than multiply-accumulates, so MACS alone are not a reliable predictor of performance. Furthermore, most DSP processors have the ability to perform other operations in parallel with MAC operations. A processor's ability to perform such parallel operations can have large impact on inner-loop performance, but this is disregarded by MACS measurements.

Neither MIPS, MOPS, nor MACS address secondary performance issues like memory usage and power consumption. This is a severe limitation because execution time means little if application memory requirements exceed system constraints. Furthermore, if high memory consumption requires using slower external memory, then the processor's speed may be reduced. Likewise, in a portable application, a processor is unusable if its power consumption exceeds the available battery capacity. Many manufacturers quote a "typical" power consumption at a given clock rate. However, power consumption varies with different instructions and data values, so such specifications are suspect without details on the precise instructions and data used in the measurement. Furthermore, such measurements do not account for special power-saving modes available when a processor (or portions of it) is idle.

It should be noted that energy consumption, which determines battery life, is usually more important to designers than power consumption. A DSP processor that can execute its work quickly and then enter a power-saving mode may consume less energy in a particular application than another DSP processor with lower power consumption. BDTI's processor evaluations report energy consumption.

Application Benchmarking

A common approach used to benchmark computer systems is to use complete applications, or even suites of applications. Application benchmarking allows the memory usage and energy consumption performance of a processor to be measured. And it is more suited to comparisons between different processor families than MIPS, MOPS, or MACS.

This approach is used by the Standard Performance Evaluation Corporation in the popular SPEC benchmarks for general-purpose processors and systems. In DSP, examples of applications include speech coders (CELP, VSELP, GSM, etc.), modems (V.34, V.90, etc.), and disk drive servo control programs. This approach works best in cases where there is application software portability—i.e., when the application is coded in a high-level language like C. Benchmarking using applications written in a high-level language amounts to benchmarking the compiler as well as the processor. Unfortunately, because of the poor efficiency of compilers for the most cost-effective DSP processors and the demanding performance requirements of the applications, the performance-critical portions of DSP software are typically coded in assembly language. Thus a benchmarking methodology that measures the compiler and the processor together does not reflect the needs of typical DSP system designers.

Even if application benchmarks are coded in assembly language, one encounters four problems with application benchmarking for DSP. First, most DSP applications are not sufficiently well-defined to permit fair comparisons. For instance, two implementations of a standard modem may carry out arithmetic with different numerical precisions, depending on whether the objective is achieving the lowest possible error rate or minimizing demands on the processor. Second, with most complex applications, it's virtually impossible to ensure that software is optimal, or even near-optimal. Thus, application implementations may be benchmarking the programmer as much as the processor. Third, full application benchmarks tend to measure a system's performance, not just the processor's. Isolating the performance of a DSP processor from that of other system components like external memory and microcontroller coprocessors could be very difficult. Last, coding an entire application for multiple processors could take years of engineering time, making it an impractical approach for benchmarking.

Algorithm Kernel Benchmarking

Berkeley Design Technology's methodology of algorithm kernel benchmarking and application profiling is a practical compromise between oversimplified MIPS-type metrics and overly complicated application-based benchmarks. Algorithm kernels are the building blocks of most signal processing systems and include functions such as fast Fourier transforms, vector additions, filters,

etc. Algorithm kernels offer several compelling advantages as benchmarks:

- **Relevance**. Algorithm kernels can be selected by examining DSP applications and focusing on those portions of the applications that account for the largest share of the processing time. This guarantees their relevance.
- **Ease of specification**. By virtue of their modest size, algorithm kernels can be well-defined: a specification can state their input and output requirements, include test vectors to verify functional conformance, and indicate which algorithm variants and optimizations are allowable. For example, there are many techniques for implementing a FFT. Without specifying the exact type of FFT, one cannot fairly compare two processors' FFT execution times.
- **Optimization**. Because algorithm kernels are of a moderate size, a skilled programmer can write the code in assembly language and be fairly certain that his or her implementation is optimal, or very close to optimal, on a given processor.
- **Ease of implementation**. Due to their moderate size, algorithm kernels can be implemented in a reasonable amount of time, even with thorough optimization.

The BDTI Benchmarks™, the basic suite of algorithm kernels used in BDTI's DSP processor benchmarking, are shown in Table 1. BDTI calculates execution time, memory usage, and energy consumption for each benchmark. Most of the benchmarks involve transforming an input data set into an output data set. The exception is the Control benchmark, in which the processor must execute a contrived sequence of operations, such as conditional branching and subroutine calls, that are commonly needed in control code. As DSP applications become more complex and system designers try to achieve higher levels of system integration, DSP processors will increasingly be called upon to perform control functions.

With the exception of the control benchmark, BDTI optimizes each benchmark for execution time. The Control benchmark is optimized for memory usage since memory usage is usually a greater concern than speed for control code.

Function	Description	Example Applications
Real Block FIR	Finite impulse response filter that operates on a block of real (not complex) data.	Speech processing (e.g., G.728 speech coding).
Complex Block FIR	FIR filter that operates on a block of complex data.	Modem channel equalization.
Real Single-Sample FIR	FIR filter that operates on a single sample of real data.	Speech processing, general filtering.
LMS Adaptive FIR	Least-mean-square adaptive filter; operates on a single sample of real data.	Channel equalization, servo control, linear predictive coding.
IIR	Infinite impulse response filter that operates on a single sample of real data.	Audio processing, general filtering.
Vector Dot Product	Sum of the pointwise multiplication of two vectors.	Convolution, correlation, matrix multiplication, multi-dimensional signal processing.
Vector Add	Pointwise addition of two vectors, producing a third vector.	Graphics, combining audio signals or images.
Vector Maximum	Find the value and location of the maximum value in a vector.	Error control coding, algorithms using block floating-point.
Viterbi Decoder	Decode a block of bits that has been convolutionally encoded.	Error control coding.
Control	A sequence of control operations (test, branch, push, pop, and bit manipulation).	Virtually all DSP applications include some control code.
256-Point In-Place FFT	Fast Fourier Transform converts a time-domain signal to the frequency domain.	Radar, sonar, MPEG audio compression, spectral analysis.
Bit Unpack	Unpacks variable-length data from a bit stream.	Audio decompression, protocol handling.

TABLE 1. BDTI Benchmarks.

Measuring Algorithm Kernel Execution

There are several ways to measure a processor's performance on an algorithm kernel benchmark. Cycle-accurate software simulators usually provide the most

convenient method for determining cycle counts. A cycle-accurate simulator models a processor's execution of instructions and keeps accurate cycle counts by making appropriate adjustments when factors such as pipeline interlocking or bus contention slow the operation of the processor. Software simulators offer a controlled, flexible, and interactive environment for testing and optimizing code. Some software simulators include support for macros or scripts that can automate performance measurement and functionality verification and allow engineers to quickly see how code changes affect a processor's performance.

Hardware-based application development tools can also be used to measure execution time and are needed to precisely gauge energy consumption. Hardware tools, such as emulators, allow the user to download code from a PC to the target processor. Using a debugger, most hardware emulators allow the processor to step through the code line by line, or to run the code until a breakpoint is reached.

Code can be run in continuous loops on development boards to measure energy consumption. Energy consumption is measured by isolating the power going to the DSP processor from the power going to other system components, running a benchmark in a repeating loop, and using a current probe to record the time-varying input current under carefully controlled conditions.

Such energy consumption measurements can be time-consuming and difficult. A less accurate but easier alternative is to obtain a credible estimate of typical power consumption and multiply it by the time taken to execute a benchmark. This is the approach taken by BDTI.

Determining benchmark performance for new processors without software or hardware development tools is a tedious and error-prone process. One must manually calculate the time required to execute each instruction in the benchmark and be careful to check that the benchmarks are functionally correct. Because pipeline interlocks or bus conflicts can slow execution time, the processor architecture must be thoroughly understood before instruction execution times are calculated.

Benchmark Results

Figure 1 shows the execution time results of several processors on the BDTI fast Fourier transform (FFT) benchmark. The FFT is a computationally efficient algorithm for computing the discrete Fourier transform, which converts time-domain signals into their frequency-domain representations. The results illustrate how architectural features affect a processor's performance, yielding benchmark results that are not what might be expected from a simple comparison of MIPS.

Texas Instruments' TMS320C6203 is a VLIW-based processor that can issue and execute up to eight instructions per instruction cycle. Hence, at the 300 MHz clock rate shown here, it has a MIPS rating of 2400 MIPS. However, its relative speed compared to another Texas Instruments DSP processor, the architecturally conventional TMS320C5416, is not nearly as high as the difference between the two processors' MIPS ratings suggests. Despite a MIPS ratio of 15:1, the TMS320C6203 executes the FFT benchmark only 7.8 times faster than the TMS320C5416. A major reason for this is that TMS320C6203 instructions are simpler than TMS320C5416 instructions, so the TMS320C6203 requires more instructions to accomplish the same task. In addition, the TMS320C6203 is not always able to make use of all of its available parallelism because of limitations such as data dependencies and pipeline effects.

This example illustrates why using MIPS ratings to compare the performance of different processors may be misleading, and why BDTI believes that algorithm kernel-based benchmarking provides much more meaningful results with which to compare processor performance.

Of course, one must be cautious when interpreting benchmark results. For example, a processor's data word width affects memory usage as well as numerical accuracy. The benchmark results for a finite impulse response filter implemented on a 24-bit processor might show 50% more data memory usage than the same filter

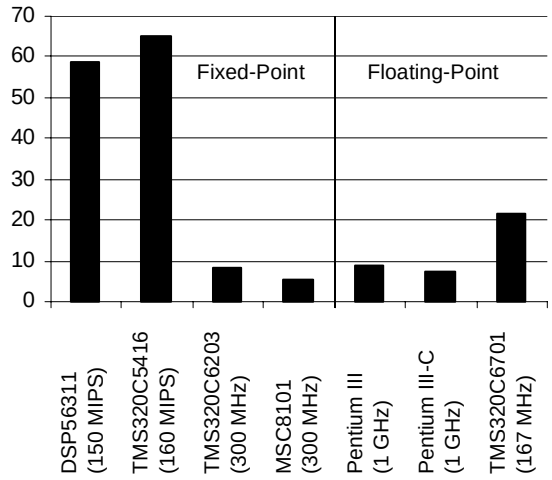

FIGURE 1. Execution times for a 256-point complex FFT, in microseconds (lower is better).

Note: Times are calculated for the fastest version of each processor projected to be available in June 2000. For the processors with on-chip cache, "-C" indicates performance with cache pre-loaded.

implemented on a 16-bit processor. This increased memory usage is a result of the extended precision of the 24-bit data. In fact, since the 24-bit processor is calculating the filter result to 50% greater precision, the 24-bit processor is in a sense performing more work—a fact not reflected in the benchmark results. If the application needs additional precision, the 24-bit processor may be an excellent choice. On the other hand, if 16-bit precision is sufficient, then the 24-bit processor may be a poor choice because it consumes more data memory.

Application Profiling

The results of algorithm kernel benchmarks are useful but incomplete without an understanding of how the kernels are used in actual applications. "Application profiling," which refers to a set of techniques used to measure or estimate the amount of time, memory or other resources that an application spends executing its various subsections, can be used to relate algorithm kernels to actual applications.

Application profiling at the algorithm kernel level looks at the number of times key algorithm kernels are executed when an application is run. This can be done in a number of ways. Code in high-level languages such as C, for example, is an excellent source of profiling information because most algorithm kernels can be identified as subroutines. If assembly code is available, profiling information may be extracted by running the code on an instruction set simulator equipped with profiling capabilities, or by setting break points in key sections of code to see how often they are executed. Profiling information can also be estimated by studying application specifications or block-level signal flow diagrams.

Application profiling allows developers to estimate the relative importance of each algorithm kernel benchmark in a particular application. Of course, it's not a perfect process. If the number of benchmarks is limited to a reasonable number, say ten or fifteen, then in many cases there won't be an exact match between every algorithm found in a complex application and a benchmark. Engineers will have to approximate some of the application's processing by using benchmarks that perform similar, but not identical, computations. It's also important to note that application profiling may not identify some of the optimizations that will be possible when assembly code is written. For example, a programmer may notice that a set of intermediate values used in one algorithm kernel is also used in a later algorithm kernel. By reusing the values, the programmer may be able to significantly reduce the amount of processing required in the second algorithm kernel.

A processor's performance on an application is estimated by combining the results of the benchmarks with the results of the application profiling. Multiplying the benchmark execution times by the number of occurrences of each benchmark (or a similar algorithm kernel) yields an estimate of the time required to execute the application. Comparing the application execution time estimates of different processors allows an engineer to gauge the relative suitability of each processor for the application.

Application profiling can be illustrated with the example of a hypothetical 10-band audio graphic equalizer. A stream of digital audio samples enters the graphic equalizer at a fixed sampling rate. The equalizer's output is produced by filtering the input samples with 10 separate cascaded biquad IIR filters. We will estimate the time required per sample as 10 times the time needed to perform BDTI's eight-biquad IIR filter benchmark. Figure 2 illustrates the estimated execution times of four 16-bit fixed-point processors from Analog Devices, Motorola, Lucent Technologies, and Texas Instruments. Since the maximum allowable execution time is the reciprocal of the system's sampling rate, the execution time estimate indicates whether or not a processor has enough performance to implement the equalizer. The results suggest that these processors could all easily handle stereo operation at sampling rates above 48 kHz. However, processing requirements beyond the filtering itself, such as control code, must also be considered. BDTI's

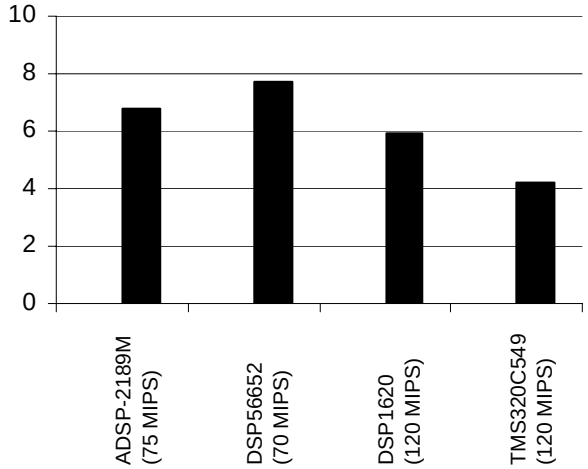

FIGURE 2. Execution times for graphic equalizer, in microseconds (lower is better)

Note: Execution times are calculated for the fastest version of each processor available in December 1999.

Control benchmark can be used to compare processor's relative efficiency at executing control code.

Other Considerations

Although performance is a leading consideration, many other factors affect the choice of a DSP processor. Application development tools, for instance, cannot be overlooked. Without effective application development tools, writing application software can be difficult no matter how strong the processor's performance. Likewise, chip vendor and third-party application engineering support can be invaluable when problems arise. Additionally, designers cannot overlook physical size considerations and must choose a processor that is available in an appropriate package.

Cost is another critical concern. There are two ways to view the ratio of cost to performance. In some instances, additional performance beyond the required minimum will remain unused. In this situation, designers typically seek the lowest-cost processor with adequate performance. At other times, the excess performance may allow additional features to be added to the product. Or, the designer may want a line of code-compatible DSP processors with performance levels appropriate for different members of an entire product line. In this situation, a cost-execution time product metric (the execution time of a processor multiplied by the unit cost) may be useful. Figure 3 shows the cost-execution time product of several processors on BDTI's FFT benchmark.

Designers must also remember that minimizing system cost may not always mean minimizing DSP processor cost. For example, one processor may use memory more efficiently than a slightly less expensive processor. If the lower memory usage can eliminate one memory chip from the system, the more expensive processor may minimize system cost. Designers must also weigh the cost of engineering time and carefully consider how the quality of application development tools will affect product development schedules.

Lessons Learned

There is no easy way to evaluate DSP processor performance meaningfully. Traditional performance units like MIPS and MOPS are misleading and do not reflect many relevant factors like application execution time, memory usage, and energy consumption. Application benchmarks, too, suffer from limitations that make fair comparisons difficult. Fortunately, a methodology of algorithm kernel benchmarking and application profiling provides good estimates of processor performance weighted to the target application.

We expect that DSP systems will continue to become more sophisticated and demand greater computational performance. At the same time, semiconductor vendors will continue to develop more powerful DSP processors and integrate these processors with other system components such as microcontrollers and peripherals. As systems become more complicated and processor choices grow, designers will need good estimates of a processor's DSP performance. The methodology outlined above will be an excellent starting place for calculating these estimates.

References

[1] *Buyer's Guide to DSP Processors*, Berkeley, California: Berkeley Design Technology, Inc., 1994, 1995, 1997, 1999, 2001. This 846-page technical report discusses DSP benchmarking methodology in detail and contains extensive benchmarking data for popular DSP processors. The report provides execution-time application profiling data from several common DSP applications. Excerpts from this report, as well as a pocket guide to DSP processors, are available at www.BDTI.com.

[2] *Inside the StarCore SC140*, Berkeley, California: Berkeley Design Technology, Inc., 2000. This report provides a comprehensive qualitative analysis of the StarCore SC140's architecture and features, along with a quantitative analysis based on results from BDTI's DSP benchmark suite. The SC140's performance is compared to that of key competitors, with benchmark results analyzed in terms of underlying

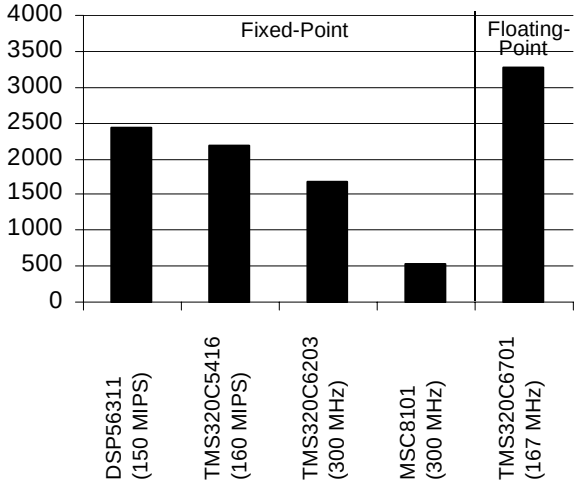

FIGURE 3. Cost-execution time product for a 256-point complex FFT, in microsecond-dollars (lower is better).

Note: Results are calculated for the fastest version of each processor projected to be available in June 2000. Costs are projected quantity 10,000 pricing for June 2000.

architectural strengths and weaknesses. The report includes coverage of Motorola's SC140-based MSC8101.

[3] Phil Lapsley, Jeff Bier, Amit Shoham, and Edward A. Lee, *DSP Processor Fundamentals: Architectures and Features*, Berkeley, California: Berkeley Design Technology, Inc., 1996. An introductory textbook on DSP processor architectures which discusses how processor architecture affects performance.

About Berkeley Design Technology

Berkeley Design Technology, Inc. (BDTI) is a software and technical services company focused on digital signal processing (DSP) technology. The company was founded by U.C. Berkeley faculty and researchers.

BDTI specializes in the analysis, benchmarking, evaluation, and development of technology used to implement DSP applications. Specifically, the company:

- Performs in-depth technical evaluations of microprocessors.
- Develops DSP application software and firmware.
- Publishes technical reports and books on DSP technology, including *Buyer's Guide to DSP Processors*, *Inside the StarCore SC140*, and *DSP Processor Fundamentals.*
- Analyzes DSP algorithms and applications.
- Evaluates design tools and advises on tool selection and design methodologies.
- Develops specifications and recommendations for new DSP processors, software, and tools.
- Provides DSP-related training classes.

BDTI customers include:

3Com	Mentor Graphics
ARM	Microsoft
AMD	MIPS
Analog Devices	Motorola
Cadence	National Semiconductor
Cisco	NEC
Compaq	Nokia
Conexant	Philips
CSF Thomson	Principal Financial
Dow Chemical	Raytheon
DSP Group	RealNetworks
E.M. Warburg Pincus	Replay Networks
Ericsson	STMicroelectronics
Fujitsu	Sony
Hewlett-Packard	StarCore
Hitachi	Sun Microsystems
IBM	Synopsys
Infineon Technologies	Texas Instruments
IDT	VLSI Technology
Intel	Wind River Systems
LSI Logic	Xilinx
Lucent Technologies	Zoran

BERKELEY DESIGN TECHNOLOGY, INC.
2107 Dwight Way, Second Floor
Berkeley, CA 94704 U.S.A.
+1 (510) 665-1600
Fax: +1 (510) 665-1680
Email: info@BDTI.com
web: www.BDTI.com

International Representatives

JAPAN
Shinichi Hosoya
Japan Kyastem Co.
Tokyo, Japan
+81 (425) 23 7176
Fax: +81 (425) 23 7178
bdt-info@kyastem.co.jp