A Review of Performance Analysis (Benchmarking) Approaches for Embedded Microprocessors and Microcontrollers

by

Charles Robert Powers, B.S.

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 1998

A Review of Performance Analysis (Benchmarking) Approaches for Embedded Microprocessors and Microcontrollers

Approved by
Supervising Committee:

Brian L. Evans

Lizy K. John

Dedication

I dedicate this report to my wife Christine, whose patience and support were what enabled me to complete it; and to my daughter Andrea, who has only known a dad who is a student, and to whom I owe a lot of time, which I can now begin to make up to her.

Acknowledgements

I would like to express my gratitude to my management at Motorola, who have given me the support and encouragement I have needed to complete this effort. In particular, I would like to express my thanks to Nigel Allison, without whose help and guidance over the years I would never have been able to rise to the challenge that this program has offered.

I would also like to acknowledge the support of American Airlines, in whose airplanes and Admiral's Clubs much of this report was written.

April 14th, 1998

Abstract

A Review of Performance Analysis (Benchmarking) Approaches for Embedded Microprocessors and Microcontrollers

by

Charles Robert Powers, M.S.E.

The University of Texas at Austin, 1998

Supervisor: Brian L. Evans

This report outlines some of the approaches taken to analyze the performance and capabilities, or benchmark, embedded microprocessors and microcontrollers. Beginning with the advent of desktop computer benchmark suites, as both an engineering evaluation tool and a marketing tool, the use of benchmarks as a measuring stick for processor-based system performance has become pervasive. While many of the popular benchmarks used to measure the performance of desktop or laptop computers provide a reasonable system evaluation in that environment, the attempts to apply similar benchmark suites to embedded systems has so far met with little success. This is primarily due to the large control aspect of many embedded systems, in which hardware and software work together to produce the desired outcome. Because traditional computer benchmarks are based primarily on the speed at which data is processed, these benchmarks are rarely relevant to embedded applications. Some industry efforts are currently underway to develop benchmark suites which are more appropriate for measuring the performance of an embedded system, but the diverse and application-specific nature of most embedded systems make this a daunting task. This paper will review some of the previous and ongoing approaches used for benchmarking the performance of microprocessors and microcontrollers in embedded systems, and discuss why these have met with limited success. This paper proposes a comprehensive approach which can be taken to benchmark embedded microprocessors and microcontrollers, including two application examples to illustrate the difficulties which must be overcome when developing a valid benchmark suite for the embedded application space.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

The performance of computing systems such as mainframe computers, desktop computers or engineering workstations have been measured and analyzed since computing machines like ENIAC first appeared in 1946. However, with the advent of benchmark suites targeted specifically at desktop computers, as both engineering evaluation and marketing tools, the use of benchmarks as a popular measuring stick for microprocessor-based system performance has become pervasive. Performance measurements such as Dhrystone MIPs, SPECfp and SPECint are routinely mentioned when the capabilities of a microprocessor unit (MPU) are discussed, or when a microprocessor manufacturer introduces a new device or derivative.

While many of the popular benchmarks used to measure the performance of computers provide a reasonably comparative system evaluation in the desktop computer environment, most attempts to apply similar benchmark approaches to embedded systems have met with little success. Embedded systems, which make up the vast majority of today’s microprocessor-based components, are primarily designed around highly integrated microcontroller units (MCUs). These devices, which are based on a central processing unit (CPU) for processing instructions and data, may also contain a variety of communication and control oriented subsystems, such as timers, serial communication ports, analog-to-digital converters, and input/output signals for providing control capabilities.

While the number of MCU-based products far exceeds the number of desktop computers, these devices have been a much more difficult target for developing comprehensive and equitable benchmark suites. This is due, in large part, to the significant control aspect of many embedded systems, in which hardware and software work together to produce the desired outcome. In this type of environment, the speed with which raw data is processed by the CPU can be less important than how the input data is retrieved, and the how fast and accurate the resulting output operations are performed. The performance of these control functions by an MCU in any individual system will be based largely on the features and capabilities of that particular device, and any additional components in the system. The wide variety of available devices and feature sets makes it much more difficult to develop a single, comprehensive benchmark suite to measure embedded system performance.

This paper will review some of the approaches which have been taken toward benchmarking embedded systems, how successful they have been, and how they compare to the desktop benchmarking efforts. Chapter 2 is a discussion of benchmarks in general, including some of the categories of processor-based system benchmarks, and how they are used. Chapter 3 reviews some of the non-embedded system benchmarking efforts which have been made, while Chapter 4 reviews some of the current approaches to benchmark embedded systems, including a discussion of their limitations. Chapter 5 is a discussion of the concept of application-based performance analysis, which is one approach to developing a more realistic benchmark for an embedded system. Chapter 6 concludes with some comments on the future of benchmarking.

Chapter 2: What is Benchmarking

The analysis of the capabilities of microprocessor- or microcontroller-based systems has previously been defined at three levels: performance measurement, which is the empirical testing of the system hardware based on device specifications; performance evaluation, which is more of a conceptual evaluation of the system performance, perhaps based on a mathematical assessment of how the system might perform a certain task; and performance analysis, a combination of the previous two, which is intended to be a measurement of the performance of the overall system [1].

In less esoteric terms, the performance analysis of a processor-based system, whether a desktop computer powered by the latest version of the MPC750 PowerPC^{TM} microprocessor or a one-way, alpha-numeric pager utilizing an MC68HC11 microcontroller, should provide a measurement of the performance of a device as it operates in a particular application environment. This performance analysis might measure the operating speed of the MPU or MCU as it processes instructions, but optimally it should measure the overall system performance, including the functionality and capabilities of any additional system components, whether integrated or discrete, as well as overall cost, area and component count.

CATEGORIES OF PROCESSOR-BASED SYSTEM BENCHMARKS

The number and types of benchmarks which have been developed is as varied as the number of operating environments into which microprocessors have been placed. Several of the more general categories are described below.

Non-embedded microprocessor-based system benchmarks

This type of benchmark is actively being developed and refined today, and has generally been intended to provide an evaluator with an idea of the overall performance of a computer. It is based primarily on the performance of a system's microprocessor, and that microprocessor's interaction with the larger computer subsystems, such as mass storage or display devices. One example is the CINT95 benchmarking suite, a set of eight integer programs developed by the Standard Performance Evaluation Corporation (SPEC), to evaluate the integer-based processing performance of MPU-based computer systems [1].

This type of benchmark suite was originally designed to measure the data processing performance of an MPU as it interfaced with memory and other subsystems. Today, many computer benchmark suites are now being developed to provide specific performance analysis of subsystems, such as multimedia/graphics processing systems or audio/sound cards.

Embedded microprocessor-based system benchmarks

More recently, attempts have been made to develop performance benchmarks for systems which may have an MPU similar to those in desktop computers, but whose functions have a significant aspect of control. One example of this type of system might be a laser printer, which typically has a high performance MPU embedded in it, but which also has obvious control functions. In these systems today, the MPU is tasked with crunching raw data, with the peripheral control functions being performed by additional components entirely external to the MPU, or possibly through some limited component integration onto the MPU.

Benchmark suites for this type of embedded application often closely resemble desktop computer benchmarks in scope. These applications are typically well-defined, and quite often there is little difference in the implementation of the control functions from one system to another. This leaves the differentiation between systems to be made by the performance of the microprocessor.

Digital Signal Processor-based system benchmarks

The significant increase in the number of embedded systems which contain a Digital Signal Processor (DSP) for performing real-time signal processing functions has naturally lead to a variety of efforts to develop benchmark suites to evaluate the performance of these processors, as well. Once used exclusively for processing audio and communications data, DSPs are now the fastest growing segment of the semiconductor market, with a 50% annual growth rate, and penetration into many different embedded application spaces. A DSP’s comparative capabilities for processing one or more incoming signals simultaneously is certainly of significant interest to designers of hi-fidelity audio or digital cellular telephone systems.

Historically, DSP benchmark suites have also been similar to desktop benchmarks, because the focus has been on the evaluation of a device’s ability to process and manipulate incoming data, and not on external control functions. Several standard algorithms have been repeatedly used for analyzing DSP performance, including the fast Fourier transform (FFT) and the finite impulse response (FIR) filter. This is changing, however, as more DSPs are becoming available with integrated control subsystems, and are being embedded into more control-type applications.

Highly embedded, microcontroller-based system benchmarks

While more than 5 billion consumer, automotive and communications devices have been produced with MCUs embedded in them, the performance of these devices has been the most difficult to analyze, because of their highly integrated nature, the wide variety of available devices, and the varied application spaces involved. Most of the published benchmarking data on MCUs is generally based on the speed of the CPU, with other CPU-related functions such as context-switching and interrupt servicing latency included. In order to benchmark the performance of a highly embedded processor in a realistic way, the application environment must also be included as a factor, as well as the control functions which contribute most to the system performance. This is the reason that most of today's benchmarks for MCUs do not provide a complete picture of a device's potential performance in an application.

VALUE OF BENCHMARKS

Benchmarking of processor-based systems provides two important sets of empirical data. The first is a set of metrics which indicate a particular device's or system's performance and capabilities for completing a specific set of functions. For example, developing and running a series of floating-point benchmark programs designed to emulate the operational requirements of a target application on a particular device can indicate to the system developer whether that device can meet the minimum requirements of the application. In this case, the result of the benchmark might be computational times for specific algorithms whose completion and efficiency are of critical importance to the application in question. This can certainly be of value when improvements in an existing design are being planned, and there is a need to understand if the current processor can meet the needs of the enhanced design.

The second type of data which might be obtained from benchmarking is a set of comparative metrics, which compare the performance of different devices operating under the same conditions and performing the same functions. When purchasing a new computer system, this type of data is intended to give the consumer some idea of how well the machines being considered compare. For designers of embedded systems, this information can be important during the system design phase, when the choice of processor for the system must be made.

Of course, benchmarking results are typically used for much more than tabulating empirical data. Today, with the large number of processors on the market from manufacturers such as Hitachi, Philips, Intel, Motorola and others, the intangible benefits derived from being able to advertise a processor's performance on a particular set of benchmarks is almost as important as the actual results. With some of today's advanced microprocessors, entire marketing campaigns have been designed around the benchmark suite performance of a device compared to those of its competitors. Of course, as with any marketing hype, this type of information must be considered in context, for sometimes the benchmarks used have been developed by the manufacturer specifically to highlight the capabilities of their own devices. The most glaring example of this is Intel Corp., which usually compares its new microprocessors only to other Intel microprocessors. But this type of information can provide some useful information when making a processor decision.

Chapter 3: Non-Embedded (Computer) Benchmarking Efforts

Non-embedded computer benchmarks are primarily targeted at desktop computer systems, such as personal computers or engineering workstations. Several of the most prevalent non-embedded benchmark suites are now so widely known outside of the technical world that many computer manufacturers typically advertise the performance of their new systems running one or another of these benchmark suites. What these benchmark suites are intended to measure, and who has developed them, is useful for understanding some of the current attempts being made to benchmark embedded systems.

THE GOALS OF NON-EMBEDDED PROCESSOR BENCHMARKING EFFORTS

The original intent of non-embedded microprocessor benchmarks was to provide an objective assessment of the performance of the microprocessor in a computer system. However, with the increase in the complexity of the microprocessors themselves, as well as the overall system architectures, the computer benchmark suites available today actually evaluate much more than just the processor. The benchmark suites available today typically provide a high-level assessment which includes the performance of the processor, the system memory, other some of the other major subsystems of the computer.

While today's complex computer systems allow only system-level performance assessment, this does not cause a serious problem in this application space, because the basics of every computer system are the same at the right level of abstraction. Each system contains one or more main processors, with varying amounts and types of memory (L1 cache, L2 cache and main memory), mass storage, a display subsystem, and logic devices to interface these systems together. In fact, for some computer evaluations, the only component being evaluated is the compiler used to compile the benchmark programs, since the remaining pieces of the system are virtually identical!

Because of the increasing system complexity, some of these benchmarking suites are now being expanded to specifically include the performance of some subsystems. For example, SPEC has created several working groups under the Graphics Performance Characterization Group to develop benchmarks for graphics subsystems. One example is the Multimedia Benchmark Committee (MBC), which was established by SPEC to develop a suite of benchmarks to evaluate the systems which deliver MPEG-based multimedia support.

Another intent of non-embedded benchmarks is to evaluate how well a processor or computer will run specific computer applications. As illustrated below, there are several benchmark suites available which utilize scripts to run popular software applications, recording the system performance results for each. This type of information might be particularly important to a business which is planning a computer purchase, and wants a system which will do the best job of running the applications they will use, for the best value.

SOME EXAMPLES OF NON-EMBEDDED BENCHMARK EFFORTS

To illustrate the efforts which have been made to develop benchmark suites for non-embedded systems, two organizations which currently provide benchmarking software are briefly described below. The first organization, the

Standard Performance Evaluation Corporation (SPEC), has created benchmarking suites by collecting and/or developing a set of application programs specifically designed to perform certain processing functions. These applications are then run on a system to evaluate its performance for that specific function. In contrast to SPEC, the Business Applications Performance Corporation (BAPCo) does not develop benchmarks consisting of custom software programs. Instead, they provide software script suites which exercise popular Microsoft Windows software applications, in order to evaluate the performance of desktop computer systems under conditions a user might encounter. More information on each of these organizations and their products is given below.

Standard Performance Evaluation Corporation

The Standard Performance Evaluation Corporation is a not-for-profit corporation set up to "establish, maintain and endorse a standardized set of relevant benchmarks that can be applied to the newest generation of high-performance computers" [1]. SPEC is divided into three groups: the Open System Group (OSG), established to develop benchmarks for computers operating in the UNIX / Windows NT environment, the High Performance Computing Group (HPCG), which targets platforms involved in high-performance numeric computing, and the Graphics Performance Characterization Group (GPC), which develops benchmarks for graphical subsystems. Current members of SPEC-OSG include Compaq Computer, IBM, Intel, Motorola and Unisys. HPCG currently counts Electronic Data Systems (EDS), Silicon Graphics and Sun Microsystems as members, and GPC members include Advanced Micro Devices, Apple Computer, Intel and Motorola.

SPEC’s primary role is to develop benchmark suites which are used to assess the comparative performance of the targeted computer systems. These benchmark suites, with supporting tool sets, are then licensed for a nominal fee. The members and licensees can then run these suites on their products and publicly report the results. SPEC also publishes these results, as well as other related news, in a quarterly newsletter.

The best known set of benchmark programs which SPEC has produced are known collectively as SPEC95, indicating the year that they were released. This benchmark suite, consisting of programs developed and/or collected by the OMG, has improved significantly since their original release in 1989. SPEC95 is actually two individual benchmark suites, CINT95 and CFP95. CINT95 is a set of programs which perform integer computations, and CFP95 is a set of programs which perform floating-point computations. These two benchmark suites are intended to evaluate the overall performance of a computer system, assessing either computer speed or throughput, depending upon how they are used. Table 1 lists the programs included in the CINT95 set, and Table 2 lists the CFP95 program set.

Table 1: CINT95 Benchmark Program Set

Program Name	Benchmark Application
099.go	A.I., plays the game of GO
124.m88ksim	Motorola 88k chip simulator
126.gcc	GNU C Compiler, builds code
129.compress	Compresses and decompresses file
130.li	LISP interpreter
132.ijpeg	JPEG image compression/decompression
134.perl	Manipulates strings in Perl
147.vortex	A database program

Table 2: CFP95 Benchmark Program Set

Program Name	Benchmark Application
101.tomcatv	Mesh generation program
102.swim	Shallow water mode, 1024 x 1024 grid
103.su2cor	Monte Carlo simulation
104.hydro2d	Hydrodynamic Navier Stokes equations
107.mgrid	Multi-grid solver in 3d potential field
110.applu	Partial differential equations
125.turb3d	Simulate isotropic turbulence in a cube
141.apsi	Solves weather / pollutants problems
145.fppp	Quantum chemistry
146.wave5	Electromagnetic particle simulation

In order to measure the computing speed of a system, each of the benchmark programs is run on that system, and the resulting “wall-clock” time to execute the program is compared to the time required to run that program on the base reference system, a Sun Microsystems SPARCStation 10/40. The result, the SPEC Ratio, is expressed as a ratio of the speed of the system being benchmarked to the speed of the reference system. For an overall *system speed* measurement, an aggregate result can be obtained by calculating the geometric mean of the individual ratios. When conservative compiler optimizations are used for the benchmark programs, these aggregate values are known as SPECint_base95 and SPECfp_base95. For aggressive compiler optimizations, these aggregate results are known as SPECint95 and SPECfp95.

In order to measure the computing *throughput* of a system, each of the benchmark programs is run on that machine for one week, which is considered the standard reference time. The resulting SPEC Rate for each program is then normalized with respect to the SPEC reference system. Similar to SPEC Ratio, an aggregate performance number is obtained by calculating the geometric mean of the individual SPEC Rates. As with the speed values, when conservative compiler optimizations are used the aggregate values are known as SPECint_rate_base95 and SPECfp_rate_base95, and with aggressive optimizations as SPECint_rate95 and SPECfp_rate95.

A sample of performance metrics collected and published which are based on the SPEC CINT95 and CFP95 benchmark program sets are listed in Table 3. The comparisons made, for several ~200 MHz systems, document the system tested, the CPU used, the clock speed of the system, the memory architecture (external memory such as main and L2 cache and instruction and data L1 cache), and the resulting SPECint95 and SPECfp95 speed performance means.

Table 3: CINT95 and CFP95 Performance Measurements

		Clock MHz	**Memory**	**SPECint**	**SPECfp**
System	**Processor**	**ext. / int.**	**ext + L1 I/D**	**base95**	**base95**
Sun SS10/40[1]	SuperSPARC	40	20/16	1.13	1.38
Dec 3000/700	Alpha 21064A	38/225	2M+16/16	3.66	5.71
Dell DimensionXPS	Pentium Pro	200	256+8/8	8.20	6.21
HP 9000/K[35]70	PA8200	200	2M/2M	14.60	23.00
IBM 43P/140	MPC604e	200	1M+32/32	7.22	6.91
Intel 82430VX	Pentium	66/200	512+8/8	5.10	4.18
Ross 200S-200/512	HyperSPARC	50/200	512+8/0	5.30	5.05
Sun Ultra2/1200	UltraSPARC	100/200	1M+16/16	7.67	11.10

Business Applications Performance Corporation

The Business Applications Performance Corporation (BAPCo) is also chartered to develop and distribute computer performance benchmarks. Like SPEC, BAPCo is a non-profit consortium of computer industry leaders who desire objective performance benchmarks for computer systems [2]. However, the approach to benchmarking the BAPCo members have taken differs significantly

[1] Reference machine for all SPEC benchmarks

from that taken by SPEC. BAPCo's members, which include Compaq Computers, IBM, Microsoft, Motorola and Texas Instruments, desired a set of computer benchmarks based on applications typically used in the business environment, rather than custom programs designed to evaluate certain computing functions.

The benchmark suites developed by BAPCo are intended to emulate a model of the workload of end users. These suites include extensive scripts and data sets that exercise some of the more popular Windows business applications to produce a performance measurement which closely corresponds to what a user would experience at the functional level when running those applications. These scripts and data sets are licensed to anyone who wishes to run the benchmarks; BAPCo does not benchmark systems themselves.

Table 4: SYSmark32/95 & NT Benchmark Applications

Application Category	**Windows Application Used**
Word Processing	Microsoft Word 7.0
	Lotus WordPro 96
Spreadsheet	Microsoft Excel 7.0
Database	Borland Paradox 7.0
Desktop Graphics	CorelDraw 6.0
Desktop Presentation	Lotus Freelance Graphics 96
	Microsoft Powerpoint 7.0
Desktop Publishing	Adobe Pagemaker 6.0

BAPCo currently has several benchmark suites available. One of the most popular BAPCo products is SYSmark32 for Windows 95 and Windows NT, which evaluates systems using 32-bit Windows applications running on Windows 95, Windows NT 3.51 or Windows NT 4.0. As with all of their benchmark products, SYSmark32/95 & NT uses several standard business applications to evaluate the performance of the system. Table 4 lists the application categories and Windows applications which are used in SYSmark32/95 & NT.

Table 5: SYSmark32/95 & NT Performance Measurements

		Clock	**Memory**	
System	**Processor**	**MHz**	**main / L2**	**SYSmark32**
Dell Dimension XPS	Pentium Pro	200	32M / 256	277
HP Vectra XU 6/200	Pentium Pro	200	32M / 512	274
IBM PC 365	Pentium Pro	200	32M / 256	259
Intel TX Lone Tree	Pentium/MMX	200	32M / 512	234
Compaq DeskPro 600	Pentium	200	64M / 256	183
NEC Powermate Enterprise.	Pentium/MMX	233	64M / 512	287
AMI Merlin DP	Pentium Pro	200	64M / 256	276
Intergraph TD-310	Pentium Pro	200	64M / 256	287

The BAPCo benchmark suites include the specified Windows applications, and the scripts and data sets necessary to exercise these applications. There are no optimizations or compiles for the user to be concerned with. The resulting performance numbers, based on a minimum of three performance runs, are given for individual application categories, and as an overall system performance number. Table 5 lists several systems evaluated using the SYSmark32/95 & NT benchmark suite, indicating the system tested, the processor, clock speed and memory configuration, and the overall SYSmark32 rating. All machines listed were running under the Windows NT 4.0 operating system.

RELEVANCE OF THESE EFFORTS TO EMBEDDED SYSTEMS EVALUATION

The examples of benchmarking suites for non-embedded computer systems illustrated above are only a small sample of those available. There are many other organizations, consortiums and private companies who publish benchmarking programs and test results comparing different computer systems and subsystems.

As mentioned earlier, the fact that these benchmarking efforts evaluate not only the processor, but also the major subsystems and, in some cases even the compiler used, is certainly acceptable for these systems, since they are essentially similar in functionality, with their primary tasks being the reading, processing, displaying and storing of data.

However, their relevance to the embedded application space is limited for precisely this reason. While the non-embedded systems' primary function is processing data, in the embedded space the control attributes of each application can be significant. If the only tasks of an embedded application which are evaluated are data processing, then the majority of the functionality would not be assessed, and any special attributes a device had for performing non-data processing functions would not be highlighted. In fact, many of the benchmarks published which compare different embedded processors fall short for this exact reason: they only evaluate a device’s ability to process data, and do not compare the capabilities of relevant control peripherals.

In the next chapter, several different examples of attempts to benchmark different types of embedded processors will be discussed, along with how successful they have been at assessing the complete system, rather than simply measuring a device’s capability for processing data.

Chapter 4: Embedded Benchmarking Approaches

In contrast to the computer system benchmarks described above, the development of equitable benchmarks for embedded systems, or even agreement on the best method of evaluation, is still elusive. On the scale from non-embedded systems to highly embedded systems, the further from a non-embedded system one gets, the less any valid performance information actually exists for the designer to rely on.

As previously mentioned, several benchmarking approaches, targeted at different types of devices or application spaces, are currently being used for embedded systems. These range from explicit processor-based performance analysis techniques, which are similar to those used for desktop computers, to attempts to define broad applications and then measure a device's ability to perform in that application space. The former method, which typically involves analyzing the capabilities of a device's instruction set for executing an algorithm or for processing certain types of data, is by far the most prevalent in the embedded world. This technique can certainly be relevant for some applications, particularly those involving real-time signal processing.

This technique is less attractive for analyzing highly embedded applications, where MCUs are the typical solution of choice. For analyzing the performance of these devices, this approach can provide an incomplete picture, at best, of a device's capabilities. In the embedded application space, a system-level approach to benchmarking is necessary to get a better picture of the capabilities of a particular device. Unfortunately, this approach has rarely been taken to its necessary limits in a meaningful way.

A large amount of benchmark information, from a variety semiconductor suppliers, software suppliers, system suppliers, universities and government organizations, has been published. Unfortunately, much of this data simply compares CPU core performance, making it less relevant for more highly embedded applications. What follows are a few examples intended to illustrate some of the different approaches which have been taken to obtain benchmark data, and why much of it is incomplete.

SOME EXAMPLES OF EMBEDDED BENCHMARK EFFORTS

To illustrate what efforts have been made to provide performance information for embedded solutions, a small cross-section of examples are illustrated below. One example illustrates the efforts being made to evaluate the performance of embedded digital signal processors using comparative algorithm execution performance. This is followed by two examples of the attempts, mostly by semiconductor vendors, to illustrate the performance advantages of their microcontrollers by comparing instruction set capability, The last example illustrates the most prevalent type of benchmark data available for embedded systems - the dreaded features matrix. This section will conclude with a brief discussion of a new effort to develop system-level benchmarks to try to achieve truly equitable analysis of highly-embedded MCUs in different application spaces.

Digital Signal Processor Performance Analysis

Today's signal processing systems share some aspects of both non-embedded and embedded systems. Because the focus of many of these systems is on real-time data processing, this lends itself to algorithm-based performance analysis of the DSP core, using approaches similar to those used for benchmarking non-embedded systems. However, the DSPs in these systems must also utilize a variety of internal and external peripherals which provide various control functions, so DSP benchmarks should also include control measurements such as interrupt service routine latency and peripheral interface capabilities. Some of the newer DSP families, from manufacturers such as Motorola and Texas Instruments, include a variety of these peripheral functions integrated directly onto the DSP, which may further complicate the development of DSP benchmarks in the future.

One source of extensive DSP benchmarking data is Berkeley Design Technology, Inc. (BDT). BDT has developed a wide array of algorithm-based benchmarking programs for DSPs, which they use internally to analyze and publish reports on many of the more popular DSPs. In addition, they also market performance analysis tools based upon their algorithms which are intended to allow customers to customize these benchmarks for their particular needs [3][4].

The BDT benchmarks are designed to evaluate the performance of the target device by implementing several different algorithm kernels which are considered typical for signal-processing applications. For each algorithm kernel, a set of specifications are developed to avoid ambiguity in implementation. This is intended to reduce, as much as possible, variations in results caused by overly simplified or overly complex interpretations of the algorithm description. Table 6 lists some of the algorithms which BDT includes in their benchmark suite. For each of these algorithms, the BDT benchmark suite provides an analysis of a device's performance based on parameters such as execution time, memory used, power consumption and the number of instruction cycle counts.

Table 6: BDT Benchmark Algorithm Functions

Algorithm Function	**Description**
Real Block FIR	FIR filter operating on a block of data
IIR	Cascade of biquad infinite impulse response filters
Vector Dot Product	Sum of pointwise multiplication of two vectors
Vector Max	Maximum value and location
Convolutional Encoder	IS-54 convolutional encoder
256-pt. FFT	256-point radix-2 in-place fast Fourier transform

BDT regularly publishes reports on their benchmark results. Figure 1 illustrates some of their more recent findings. This graph relates the comparative performance, in execution time, of several popular DSP devices performing a 256-point radix-2 in-place fast Fourier transform (FFT) algorithm.

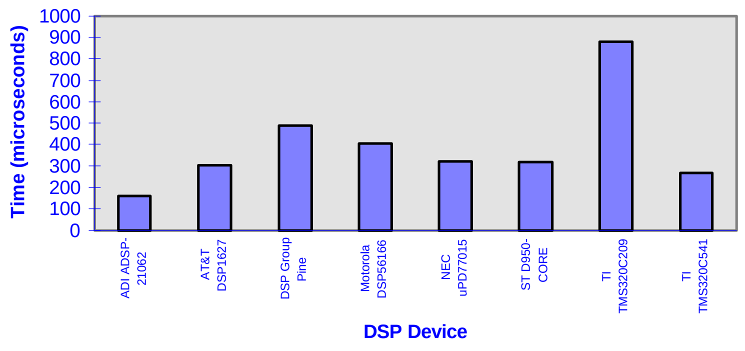

Figure 1: DSP Benchmark Results (source: Berkeley Design Technology, Inc.)

Microcontroller CPU Performance Analysis

The analysis of MCU performance is probably the most difficult, because of their unique nature, and the wide variety of devices available. Those organizations or individuals who actually have developed and run benchmark software for MCUs typically just focus on the performance of the CPU and it's instruction set, ignoring the contributions which a complex integrated peripheral set can make towards improving performance in a particular application. Of course, this approach is appreciated by device suppliers such as Hitachi and Intel, who have higher performing MCU cores and relatively small product portfolios. However, MCU manufacturers such as Motorola and Philips, who rely on high levels of integration and large product portfolios to differentiate themselves, are not as supportive of this CPU-centric approach to MCU benchmarking.

An example of this type of performance analysis data has been published by Philips Semiconductor, in order to demonstrate the performance advantage of their XA architecture over other MCU architectures [5]. For these benchmarks, the author has purposely focused on only the instruction set execution time and code density for a series of operations which were deemed typical for engine management applications. This results in a benchmark which, while ostensibly a system-level benchmark, is solely focused on CPU performance.

In order to define this benchmark, the author developed a list of operations believed to be important in a typical engine management application. Estimates were then made of the number of occurrences of each operation in a 2ms period, roughly equivalent to a single machine stroke. Following this requirements definition, assembly language code for each CPU being considered was written, and the execution time, based on a 16 MHz clock, and required code density was calculated.

The resulting data, for four different CPU cores, is listed here. Table 7 lists the functional descriptions of the operations used. The list in Table 8 contains comparative execution times for the XA architecture compared to the Motorola 68000 and Intel MCS-96 and MCS-51 architectures for several of the operations. It is interesting to note that the CPUs chosen by the author for comparison are all CPUs which are now rarely used for engine management applications, and in fact the 68000 CPU was never used in this application space.

Table 7: Philips Semiconductor Benchmark Operations

Operation	Occurrences
16 X 16 Multiply	12
Floating Point Divide (16:16)	4
Add / Subtract (24)	50
Compare (24)	13
CAN 2.0b Compare / Move (10 * 8)	80
Linear Interpolation (8 * 8)	20
Interrupts	10
Program Control Branches	500

Table 8: Philips Semiconductor Benchmark Results

CPU Core	16 X 16 Multiply	24 bit compare	Interrupt	Total Exec. Time
Motorola 68000	4.4us	3.2us	21.9us	1560us
Intel MCS-51	37.5us	9.98us	31.5us	5942us
Philips XA	.75us	1.06us	6.1us	402.6us
Intel MCS-96	1.75us	4.25us	12.8us	1089.24us

Another example of comparative performance analysis is a simple performance benchmark published by Avnet, Inc. [6], which compares the relative capabilities of the Intel MCS-96 and MCS-51 architectures when performing a relatively small subroutine. This subroutine includes multiply and divide operations, processing of overflow and underflow conditions, and indication of signed results.

While the actual details of the operations are not given, some of the code used is described, indicating that the expected operational improvements between the MCS-51 and MCS-96 MCU families are due primarily to the faster clock speed of the MCS-96 family, and the expansion of the instruction set, particularly the addition of the 32:16 divide instruction in the MCS-96 CPU. Table 9 lists the basic results, giving the overall speed of execution of the subroutine.

Table 9: Avnet Performance Benchmark Results

CPU @ Clock Speed	Execution Time
MCS-51 @ 12MHz	1642.0us
MCS-96 @ 12MHz	62.5us
MCS-96 @ 16MHz	47.1us
MCS-96 @ 20MHz	37.7us

It is interesting to note that this benchmark actually does take into account the integration of added functionality into the MCS-96 CPU architecture, particularly the 32:16 divide instruction. However, this is not really comparable to including the performance of peripheral functions, such as intelligent timers or I/O interfaces, in a performance benchmark. As an added instruction, it is relatively easy to include the divide instruction performance in a basic CPU benchmark.

MCU Features Comparison

Many attempts at benchmarking MCUs simply result in a chart listing different hardware features of each device, with no actual performance data supplied. This is the most common approach taken when comprehensive “Buyer’s Guides” or “MCU Comparisons” are published. Many electronics industry trade publications, such as Electronic Design News (EDN) or Embedded Systems Programming, publish this type of information on a regular basis. Typically, a matrix chart showing device features and basic capability is provided, along with a brief functional description of each device family. However, actual performance information is usually lacking, and the device descriptions are often provided by the manufacturers, ensuring that only positive attributes are brought out.

Table 10: EDN Microcontroller Directory Sample Device Information

Device	CPU Freq. (MHz)	Nonvolatile Memory (kbytes)	SRAM	Serial I/O	Other Features	Price (10000)
83C51FB	33	32 (ROM)	256	UART	Programmable counter array	$2.75
SAB C505C	20	0	512	USART	CAN 2.0b, watchdogs, fast POR	$4.77
8xC251Sx	16	16 (OTP)	1024	sync/ async	PWM, waveform capture	$15.00
H8/3337YFLH16	16	60 (FLASH)	2048	USARTs	Keyboard scan, I2C, PWMs, DACs	$13.00
PIC16C924	8	4 (EPROM)	176	I2C/SPI	25-ma source/sink I/O, PWM, LCD driver	$7.21
68HC05B32	4	32 (ROM)	528	USART	Charge Pump, A/D, DAC, PWM	$5.70
68HC11E9	3,2	12 (ROM)	512	SPI, SCI	EEPROM, Pulse width, RTI	$5.78
ST90135	25	64 (OTP)	12288	USART	DMA channel	$6.50
Z86C02	8	0.5 (EEPROM)	61	UARTs	Watchdog, comparators, stop-mode recovery	$0.80

An example of this type of information matrix is illustrated in Table 10 [7]. A matrix of information for several 8-bit microcontrollers is provided, including CPU characteristics, budgetary cost and additional features. In this type of list only sample members of a microcontroller family can be included, so the reader is given no real insight as to whether a device from a particular MCU family is appropriate for a particular application. The primary value in this type of comparison is simply to expose the reader to what manufacturer produces what MCU families.

LIMITATIONS OF THESE BENCHMARKING APPROACHES FOR EMBEDDED SYSTEMS

The examples described above are only a small cross-section of the available benchmark information on embedded products. However, the majority of this information has been collected using one of the approaches given. As already mentioned, some of the approaches described here may be appropriate for applications requiring real-time signal-processing, or some other data processing-intensive capability. For MCU-based embedded applications with a significant control component, only the application-level approach to performance analysis can really provide a broad understanding of the capabilities of a device in a specific application space. This still paints an incomplete picture, however, if the focus is entirely on the MCU controlling the application.

To measure the true performance and value of a particular device in an application, the scope of the benchmark must include an implementation of the entire system, or at least some significant subset. Only when all devices needed for fulfilling the performance requirements of the application are specified in the scope of the benchmark can a complete assessment of any particular MCU’s capabilities truly be evaluated. This type of true, system-level benchmarking has rarely been attempted, because of the complexity involved. However, one attempt is currently underway.

System-Level Performance Analysis

In an attempt to fill this need for meaningful, system-level benchmarks, a new industry consortium was formed in 1997 by EDN magazine. This organization, the EDN Embedded Microprocessor Benchmark Consortium (EEMBC) is made up of semiconductor design and manufacturing leaders, including Advanced Micro Devices, IBM, Motorola, NEC Electronics and Texas Instruments. The stated goal of EEMBC is to “collaboratively develop a suite of benchmarks that will help customers evaluate microprocessors and aid the industry in improving the performance and functionality of microprocessors used in embedded systems” [8]. While it is currently unclear what level of success EEMBC will achieve in developing full system-level benchmarks, they have targeted five market segments for defining applications and specific benchmarks: consumer, networking, telecommunications, office automation and automotive/industrial. For each of these target markets, a different EEMBC subcommittee has identified important system-level applications. For each of these applications, functional requirements are being defined, along with the metrics which must be reported.

Once these functional definitions are completed, consultants will be used to develop reference software for the benchmark, which subscribers will be allowed to optimize as they see fit in order to evaluate the performance of their device(s) in the defined application. One or more certification agencies will also be identified, to review a subscribers optimized benchmark code in order to verify its similarity to the original reference. Once this is done, that subscriber will be able to publish the results as official EEMBC findings.

While the EEMBC organization seems to be making the most aggressive attempt to date at defining system-level benchmarks, it is still unclear how successful they will ultimately be. There are still many obstacles to be overcome for a group of this type of achieve consensus on this effort. These include technical problems such as reaching agreement on the scope and metrics of each system, and agreeing on methods of benchmark reference code development and compliance certification. Significant political problems must also be dealt with, such as ensuring that a group of semiconductor manufacturers can reach consensus on how to measure device performance without raising antitrust concerns in the industry.

While system-level benchmarks can be very difficult to define, if an application with clearly understood functionality is chosen, then it is certainly not impossible to define a fair way of measuring a particular solution for that application. The next chapter illustrates two example applications, with suggestions on how a system-level performance benchmark for each might be defined.

Chapter 5: Possible System-Level Application Benchmarks

The CPU-centric approach to performance benchmarks is really limited to basic data processing functions (such as multiply, divide, and compare) and device operations (such as interrupt servicing time, context switching time). The MCU-only, application-level approach to MCU performance analysis can be of greater value, since some of the added benefits of integrated peripherals can be documented. However, this can still paint an unfinished picture, if the focus is entirely on the MCU controlling the application.

SYSTEM-LEVEL BENCHMARKS: PROS AND CONS

To measure the true capabilities and performance of a particular MCU or DSP in an embedded application, measures such as part count, board space, interconnect complexity and relative component cost must be included in the analysis. Only in this way can all of the benefits derived from high levels of integration be highlighted by standardized performance analysis. Unfortunately, for this to be achieved the scope of the application must be such that an implementation of the application, or some significant subset, can be defined in the benchmark.

The benefits of this approach are many-fold. For MCU manufacturers, the system-level approach will not only measure the processor performance for their devices, but in applications where cost and space issues may outweigh raw CPU performance, advantages in these categories can also be highlighted. Benchmark implementers can make hardware/software partitioning decisions based on available device capability and performance. Also, the results allow the application developer to evaluate performance data based on all of the factors which are relevant to their implementation, rather than having to draw conclusions based on partial data. Overall, this approach provides more relevant results for the application developer, who needs to make an informed choice in which all factors have been accounted for.

Of course, this type of performance analysis is also the most difficult to define and to measure. Defining a benchmark based on a complete application can be very difficult, unless the application is self-contained, or can be subdivided into independent subparts. For example, an automotive remote keyless entry (RKE) application might be straightforward to benchmark, since the primary functionality remains the same from one system to the next. Conversely, an automotive general electronics module (GEM) would be much more difficult, because the desired functionality from one system to the next varies greatly.

Quite naturally, a benchmark which is difficult to define is also difficult to ensure adherence to the defined functionality. A system-level benchmark must result in quantifiable behavioral and implementation results which an application developer can use to assess performance. If the application in question does not have externally measurable functionality as well as data processing functionality, then it becomes very difficult to ensure that the spirit of the benchmark is being met; i.e., the implementation does not cheat in order to appear more suited to the application. Even when the benchmark has been implemented in accordance with the defined scope, with the amount of leeway which must be allowed for implementation, the resulting performance numbers can still vary greatly due to subtle functionality differences. Therefore, anyone wishing to define this type of benchmark must try to develop metrics which will as accurately as possible reflect the performance of the basic functionality, while allowing the flexibility required to differentiate between different solutions.

EXAMPLE APPLICATIONS FOR SYSTEM-LEVEL BENCHMARKS

Below are two example applications which might be targets for this type of performance analysis. These two applications, an automotive instrumentation system and a global positioning systems receiver, are both currently being considered by EEMBC for their benchmarking efforts. Each will be described briefly, along with some possible measures which may be used to analyze performance in a system-level benchmark.

Automotive Driver Instrumentation System

One of the most fundamental applications in today's automobile is the driver instrumentation system, known as the instrument cluster. This system is tasked with a variety of input, processing and output functions which are necessary to keep the vehicle operator informed of the state of the vehicle, and each of its critical systems. Because the basic functionality of every instrument cluster, including the sources of input data and the methods of displaying output data, is very similar, this application is ideal for developing a system-level benchmark. This type of benchmark would be very timely, too, as many automotive manufacturers are moving towards a higher level of electronics in their instrument clusters. Having a standard way of measuring suppliers' instrument cluster solutions would provide a valuable piece of information to an automotive manufacturer.

Instrument Cluster System Overview

The basic functionality of an instrument cluster includes collecting inputs from remote sensing units, processing this information as needed, and presenting this data to the driver. In more recent instrument clusters, this basic functionality has been enhanced by the addition of serial communication networks as a source of the data inputs, and character-based electronic displays (LED/LCD) for indicating gear shift position or odometer readings.

An instrument cluster system collects inputs from a variety of sources. In more basic systems, these inputs are collected directly from remote sensors. Examples include the fuel tank level sensor, the engine RPM sensor, the vehicle speed sensor, and the engine coolant temperature sensor. In newer systems, this input data is being supplied to the instrument cluster by a connection to a serial network, which connects the vehicle’s primary body electronics and powertrain systems to a single communication system. The addition of this communication port on new instrument cluster systems has helped reduce wiring harness complexity by providing most or all of the necessary vehicle data through a single communication port.

Regardless of the source of the inputs, all of the data received by the instrument cluster must be converted from raw form into information that can be displayed to the driver. For direct connections to vehicle sensors, this includes converting the data to a digital format, performing data algorithm processing where appropriate, and then converting the resulting data into a format which is appropriate for the display type. For data received from a communication network, format conversions are typically required. In addition, some additional processing may also be necessary in order to make display decisions. For example, the motion of the fuel in the tank must be taken into account when a low fuel level indication is being made to the driver.

The output of data to the driver can also take many forms. The most familiar forms include air-coil gauges, which are typically used to display data such as vehicle speed, engine revolutions and fuel level. Many indicator lamps are also used, to indicate turns, vehicle system faults, and high beam headlight status. Other types of information outputs are also common, but less recognized as instrument cluster functions. These can include warning chimes, instrument cluster backlighting, and odometer, trip meter and vehicle transmission (PRNDL) status. While these indications have historically been made mechanically, today's instrumentation systems are capable of electronic control of these outputs. For an example of the functions and subsystems in a typical instrument cluster display, refer to Figure 2.

Possible Instrument Cluster Benchmark Measurements

With this basic functionality common to all automotive instrumentation systems, it is rather straightforward to suggest some functional and physical implementation characteristics which might be used for evaluating the performance of this type of system. This system definition should include type and periodicity of inputs, required data processing and types and display of outputs. Because many of the outputs are discrete (lamps, chimes, etc), other issues such as cost and component count will also be important factors in the results.

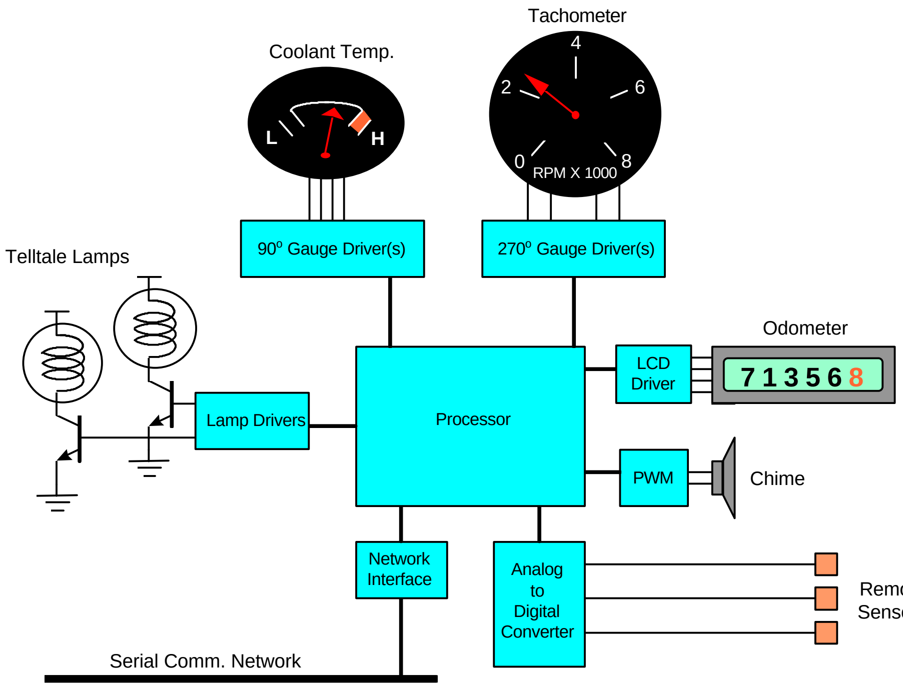

Figure 2: Basic Instrument Cluster System

The system definition should include as data sources direct sensor connections as well as a connection to a serial communication system. The sensor connections can typically be read directly at a rate which depends upon the function: fuel and temperature change relatively slowly, while engine revolutions can change more quickly. The data received from the serial network typically comes at a predefined rate, regardless of the rate of change of the source.

Data processing would include processing algorithms such as data averaging over several readings, processing and filtering messages received from the serial network, and calculating display criteria such as the delay before indicating a low fuel level, in order to filter out sloshing in the fuel tank.

Outputs should include all of the basic outputs: air-coil gauges, lamps, and chimes. Additional displays could be included such as an LCD or LED display for displaying odometer or trip meter readings, or for displaying ASCII text messages to the driver. Additional functionality might include a real-time clock.

Table 11 lists some of the application-specific behavioral and implementation requirements and functions necessary for the operation of an automotive instrumentation system, along with some of the possible measures which could be evaluated by an instrument cluster benchmark. This list does not focus on the data processing capabilities which might also be applicable, but on the non-traditional types of measures for a benchmark. While it is not a comprehensive list, it does illustrate the wide variety of information which might be used to judge the performance of an MCU and related components in this type of system-level benchmark.

Table 12 lists some generic application measures which might also be used for system-level benchmarks. This list contains some implementation measures which might be applied to any application space for which a system-level benchmark is being developed.

Table 11: Example Application-Specific Behavioral and Implementation Measures for an Instrument Cluster System-Level Benchmark

Function / Issue	**Possible Behavior/Implementation Measure**	**Relative Importance[2] (1 - 10)**
Sensor Connection	Analog to Digital (A/D) conversion resolution;	8
	A/D conversion speed	7
Network Connection	Link Layer message handling/filtering;	7
	Network physical layer capabilities	5
Gauge Drivers	Major gauge (270^o) drive capability;	10
	Minor gauge (90^o) drive capability;	10
	Gauge update speed;	7
	Automatic vs. Manual update	5
Lamp Drivers	Voltage capability;	8
	Low-side vs. High-side drive	6
Backlighting	Pulse Width Modulation frequency range	4
Chimes	Pulse Width Modulation frequency range	4
Alternative Displays	LCD / LED Display capabilities;	3
	Display update speed	3
Real Time Clock	Real time calculation	1
Integration Level	Integrated features (gauge drivers, lamp drivers, serial port, display drivers, A/D converters, etc.)	5

[2] For relative importance: 1 = lowest, 10 = highest

Table 12: Example Implementation Measures for All System-Level Benchmarks

Function / Issue	Possible Implementation Measure	Relative Importance[3] (1 - 10)
Component Count	Number of components required	7
Component Height	Board spacing;	5
	Minimum enclosure area	5
Interconnects required	Interface requirements for external components	4
Board Space	Relative cost of all devices;	10
	Area required	8
EMI / EMC	Electromagnetic emissions (high speed interfaces, multiple oscillators);	10
	Electromagnetic susceptibility	10

Global Positioning System Receiver

Another application which is seeing more use, and competition, today is a receiver for the Global Positioning System (GPS). GPS is a constellation of 24 satellites circling the earth which transmit a stream of data from which a receiver's position, elevation, velocity and direction can be derived. Like the instrument cluster example, the GPS system functionality is basically the same regardless of

[3] For relative importance: 1 = lowest, 10 = highest

the implementation, and therefore a possible candidate for a system-level benchmark.

GPS System Overview

A basic GPS receiver is designed to receive very weak signals broadcast from one or more of the satellites in the GPS constellation. The number of satellite signals received determines, up to a point, the accuracy of the position. A basic GPS system includes a radio frequency device (RFIC) for receiving the broadcast signals from the satellites. The intermediate frequency (IF) output of the RFIC is passed into an application-specific device referred to as a correlator. The correlator receives the digital data and, depending upon the number of channels it is designed for, differentiates between the satellite signals, and processes the received signal for each satellite.

The output of each channel of the correlator is then processed to calculate the latitude and longitude position of the receiver, the elevation, speed and direction the receiver is moving, and the local time at the receiver's position. To get a relatively accurate position reading, at least three satellites must be in sight of the GPS receiver. Some elements, such as the local time and the datum format used for calculating the latitude and longitude, are based on either accepted constants or on inputs from the user, such as the local time zone.

The outputs for the receiver can be displayed on a simple LED or LCD display, or can be fed into a more complex system, such as a navigation system which might use the GPS output to determine the user's current position. More complex GPS receiver systems might provide functions such as recording path segments or displaying which satellite's signals have been acquired. In addition, to boost the signal strength of the received broadcast, some GPS receivers also make use of a low-noise amplifier (LNA) to amplify the received signal before it is processed. For an illustration of the basic components which comprise a GPS receiver, refer to Figure 3.

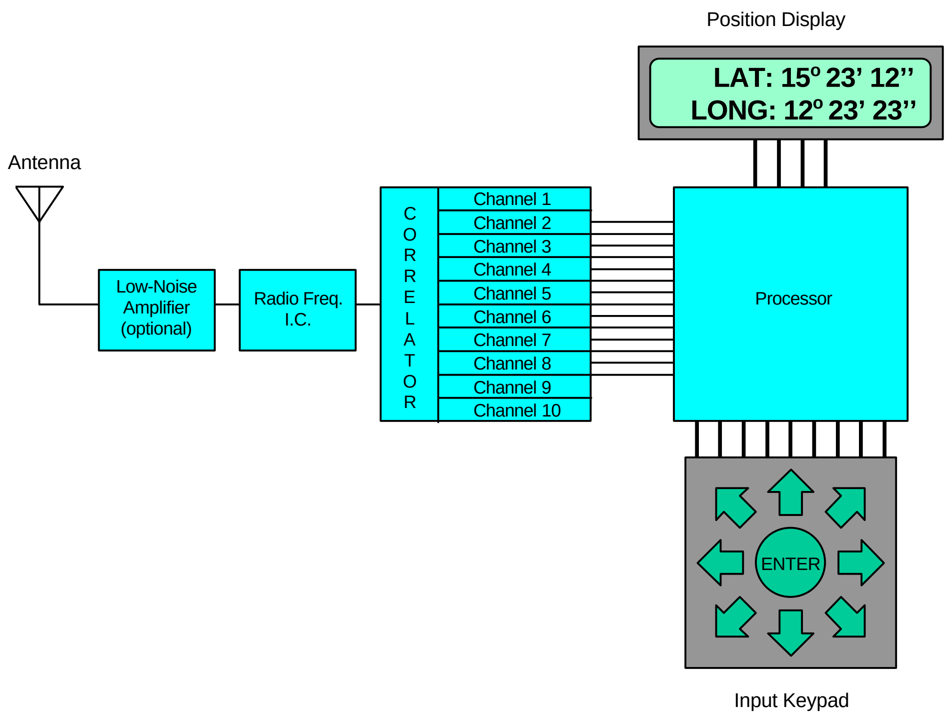

Figure 3: Basic Global Positioning Systems Receiver

Possible GPS Receiver Benchmark Measurements

While a considerable amount of CPU processing is required for converting the correlator outputs into a receiver's position, additional factors such as the number of channels the correlator can simultaneously process or the datum types which the GPS receiver supports are also important to the application designer. Cost and component count is also an important factor, since the goal of many of today's systems is to be as compact and cost-effective as possible.

A system benchmark definition for a GPS receiver would need to specify the expected signal strength of the incoming signal, the minimum number of satellite signals which would be expected to be received, and the type and format of the output display. Additional types of outputs might be specified, such as the ability to enter the local time zone, or to specify the datum format to be supported. Since the source of the GPS signals is fixed and understood, the performance of a GPS receiver would be measured primarily on the accuracy of the position output data, and the ability of the user to provide inputs to modify that output.

Table 13 lists some of the application-specific behavioral and implementation requirements and functions necessary for the operation of the GPS receiver, along with some of the possible measures which could be evaluated by a GPS receiver benchmark. Due to the nature of a GPS receiver application, the possible measures rely a bit more on CPU processing power than the instrument cluster benchmark example. As with the instrument cluster benchmark, Table 12 lists some of the generic implementation measures which might be used for the GPS system-level benchmark.

Table 13: Example Application-Specific Behavioral and Implementation Measures for a GPS System-Level Benchmark

Function / Issue	Possible Behavior/Implementation Measure	Relative Importance[4] (1 - 10)
Signal Strength	RFIC capabilities; Low noise amplifier requirements	10
Correlator capability	Number of channels supported;	8
	Time to first fix;	7
	Time to reacquire after loss of signal;	9
	Software vs. Hardware correlator implementation	4
GPS Datum Format	Fixed vs. standard / user defined formats	6
Data Available	Latitude, Longitude, Velocity, Elevation, Direction, Local time	8
Data Accuracy	Position error;	10
	Position error between signal locks	8
Integration level	Integration features (correlators, A/D converters, timers, serial ports)	5

[4] For relative importance: 1 = lowest, 10 = highest

These two application examples, while not comprehensive in their scope, help to illustrate what types of measures could be made in a system-level benchmark. They also give an indication of how difficult it is to define this type of benchmark, and what types of applications are suitable targets. In more complex application spaces, it may be necessary to define the scope of a system-level benchmark as a suitable subset of the entire application. However, even this approach may be too difficult for some applications.

Chapter 6: Conclusion

In summary, the vast majority of the published performance analysis results for embedded microprocessors and microcontrollers have, at best, provided only partial information. These approaches, which include the analysis of data processing capabilities, the comparison of instruction set and other CPU capabilities, and MCU features comparisons, while all providing relevant information, also each have their shortcomings in providing a complete assessment of a device's capabilities in an application.

In order to fully evaluate a device's capability in a particular application space, a system-level approach must be taken. This means defining the scope of the benchmark to measure not only the performance of the CPU, but also the capabilities of any integrated or discrete peripherals, how efficiently the processor interfaces to these peripherals, and the overall cost, component count and area required to assemble a complete solution. The resulting metrics must reflect the entire system performance, rather than just a single component. While this is certainly the most difficult approach, and in some application spaces is probably not possible, this is the only approach which will provide an evaluator with the most complete assessment of a device's ability to meet a system designer's needs.

References

[1] “SPEC Frequently Asked Questions (FAQ)”, Standard Performance Evaluation Corporation (SPEC), 1995, Manassas, VA
- http://www.specbench.org/spec/faq/

[2] “BAPCo Frequently Asked Questions (FAQ)”, Business Applications Performance Corporation, 1998, Santa Clara, CA
- http://www.bapco.com/info.htm

[3] Lapsley, P., Bier, J., “DSP Benchmarks: Methodology and Results”, Reprinted from: *Proceedings of the International Conference on Signal Processing Applications and Technology*, 1994
- http://www.bdti.com/articles/method.htm

[4] Lapsley, P., Blalock, G., “Evaluating DSP Processor Performance”, Berkeley Design Technology, Inc., 1996, Fremont, CA
- http://www.bdti.com/articles/wpeval.htm

[5] Roy, S., “AN703: XA benchmark versus the architectures 68000, 80C196, and 80C51”, Philips Semiconductors, 1996
- http://www-us2.semiconductors.philips.com/acrobat/8206.pdf

[6] Hartman, R., “MCS-96 vs. MCS-51: A Performance Benchmark”, *Hamilton-Hallmark Technology Review*, Avnet, Inc, 1997,
- http://www.hh.avnet.com

[7] Levy, Markus, “EDN’s Annual Microprocessor/Microcontroller Directory”, *EDN Magazine*, September 1997

[8] “New Consortium to Drive Benchmarking Standards for Embedded Microprocessors”, EDN Embedded Microprocessor Benchmarking Consortium, 1997,
- http://www.eembc.org

Vita

Charles (Chuck) Powers was born in Monroe, MI on January 24th, 1962, the son of Cornelius Powers and Doris Griggs Powers. After graduating from New Braunfels High School in New Braunfels, TX, he attended Texas Tech University, where he received a Bachelor of Science degree in Engineering Technology in August, 1986. After graduation, he spent 19 months working for Texas Instruments as a test engineer, before moving to his current employer, Motorola, in 1988. While with Motorola, Chuck has held positions of increasing responsibility in product and applications engineering, leading to his current position as manager of Systems Engineering for the Intelligent Transportation Systems operation of the Semiconductor Products Sector. He has authored or co-authored more than 20 papers, articles and application notes, and is also actively participating in leadership roles in several industry standards committees. He and his wife, Christine, have a daughter named Andrea, and a second child due in August, 1998.

Permanent address: 6408 Convict Hill Road
Austin, TX 78749

This report was typed by the author.