DSP Architectures: Past, Present and Future*

Edwin J. Tan, Wendi B. Heinzelman
Department of Electrical and Computer Engineering
University of Rochester
Rochester, NY 14627
{etan | wheinzel}@ece.rochester.edu

Abstract - As far as the future of communications is concerned, we have seen that there is great demand for audio and video data to complement text. Digital signal processing (DSP) is the science that enables traditionally analog audio and video signals to be processed digitally for transmission, storage, reproduction and manipulation. In this paper, we will explain the various DSP architectures and its silicon implementation. We will also discuss the state-of-the art and examine the issues pertaining to performance.

1 Introduction

In the last few years, the future of communications has been largely influenced by the rapid growth of wireless telephony, the Internet and mobile computing. The traditional purposes of signal processing such as modems, music synthesis and noise cancellation, while important, have been overtaken by the new-found Web based applications. These emerging technologies, especially in the area of wireless communications and Internet audio/video, have led to a 50% increase in DSP processor shipments in 1999 [1].

As a result of this rapidly expanding market, DSP vendors are vying for an ever larger slice of the pie. To entice end product manufacturers to adopt their chips as well as to meet the needs of the emerging technologies, new innovations in DSP capabilities are required. We will look at the traditional DSP as well as the current features and the historical concepts behind the DSP architecture.

Like its microprocessor counterpart, performance is of great interest. Benchmarking provides a common means for DSP users to evaluate and compare DSP chips in the market. These results show that DSP processors are also bounded by tradeoffs in terms of speed, power and computational tasks.

*This work was supported in part by the University of Rochester SEAS Graduate Student Fellowship No. 2-11144-1641.

2 DSP Processor Fundamentals

In the literature, the definition of a digital signal processor takes many forms. In a strict sense, a DSP is any microprocessor that processes digitally represented signals [2]. A DSP filter for example, takes one or more discrete inputs, $x_i[n]$, and produces one corresponding output, $y[n]$ for $n = \ldots, -1, 0, 1, 2, \ldots$, and $i = 1, \ldots, N$ [3], where n is the nth input or output at time n, i is the ith coefficient and N is the length of the filter. In effect, the DSP implements the discrete-time system. As its name implies, it is assumed that there must be some form of preprocessing if the signals are in the continuous time domain, and this is easily accomplished by an analog to digital converter (ADC).

In general, DSP functions are mathematical operations on real-time signals and are repetitive and numerically intensive. Samples from real-time signals can number in the millions and hence a large memory bandwidth is needed. It is because of this very nature that DSP processors are created with an architecture unlike those of conventional microprocessors. Most DSP algorithms are not complicated and only require multiply and accumulate calculations [4]. Most, if not all, DSP processors have circuitry built and hard wired to execute these calculations as fast as possible.

2.1 Processor Architectures

The signal processing algorithms and functions define a suitable architecture for implementation. We use a simple example of an FIR filter as a basis for the building blocks of the DSP architecture.

One algorithm used to create an FIR filter uses a direct form or tapped delay line structure with $M+1$ taps. The $M+1$ most recent input samples are saved as "filter states". According to Equation (1),

$$y(n) = \sum_{i=0}^{M} c_i x(n-i) \quad (1)$$

the products of each filter state $x(n-i)$ and its corresponding coefficient c_i are accumulated or added to produce the current output sample $y(n)$. We can also use the signal flow graph as shown in Figure 1 to represent this algorithm. However it is not clear as to the sequence of the computations since it looks like all the operations can be carried out at the same time.

This cannot be the case as operations have to follow a sequence for proper algorithm functionality. It is also not stated as to where the locations of the data operands and coefficients are before they are used in the computations. Thus, a more accurate picture has to be formed by using *micro-operations* at the register transfer level (RTL), sequenced temporarily from left to right as seen in Figure 2 [3].

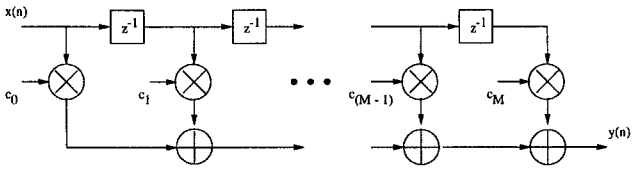

Figure 1: Tapped delay line structure of a FIR filter.

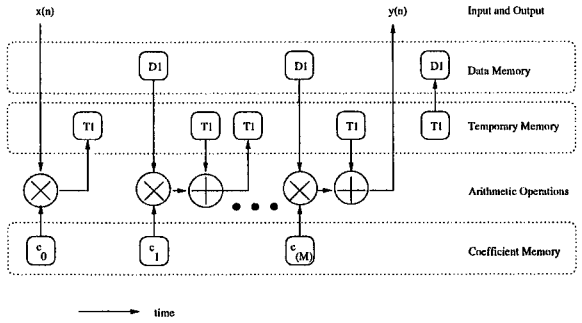

Figure 2: Register transfer level representation of a FIR filter.

The delayed inputs are stored in the data memory, *D1* and the coefficients, $c_0, c_1, \ldots c_{(M)}$ are located in the coefficient memory. The contents of both memories are fetched and multiplied together. The result is then added to the temporary memory, *T1*. *T1* is where the results of the previous taps are stored. This cycle is repeated with a different coefficient until completion, producing the final result as $y(n)$.

We can make certain assumptions for a fundemental general purpose DSP architecture. From our understanding of DSP algorithms, we see that most computations are multiply and add operations. Looking at the example from the previous section, we will require multiple memory units for storage of different data as well as memory for the arithmetic operation sequences. Registers can serve as temporary storage locations and buses will be needed to connect these units together.

At this point, the reader may be tempted to ask how this design is different from a general purpose microprocessor (GPP). If we recap the issues central to a DSP function, most DSP calculations are repetitive, require a large memory bandwidth and numeric precision, all executed in real time [5]. One might also argue that modern GPPs have clock speeds and cycles per intsruction (CPI) that outperform DSP processors but GPPs have operations and program flexibility that are unecessary for DSP [4]. DSPs must execute their tasks efficiently while keeping cost, power consumption, memory usage and development time low [5], especially in the age of mobile computing.

Since many signal processing applications process millions of samples of data for every second of operation, the minimum *sample period* is usually more important than the *computational latency* of the processor [3]. We define the sample period as the time between each sequential sample of the input data. The time difference between the input data and the result of its computation is known as the computational latency. Once the initial sample is calculated with a certain latency, the subsequent results will however, be produced at the sample period rate. As the number of calculations increases, the relatively larger latency of the processor will be negligible compared to the sample rate.

3 Processor Evolution

Even though DSP processors have seen dramatic changes through the past couple decades, there are certain features central to most DSP processors in the market today. We already know that these processors need multiple memory banks with independent buses, but in addition, specialized instruction sets, addressing modes, control and peripherals are also required.

It is widely known in the industry that the general DSP architectures can be divided into three or four categories or generations [5] [6] and we will look at each of them in turn. We will not address custom DSP architectures for specific DSP algorithms in this paper.

3.1 Early Single Chip DSP Processors

The first single chip processors [7] were the foundation on which modern DSP processors were built. Although most of them were not commercially successful, manufacturers were quick to learn the pitfalls surrounding each of them. It is also interesting to note that among these early chip vendors, only one has maintained a DSP product line to this day.

In 1978, AMI released a "Signal Processing Peripheral" known as the S2811 which was designed to operate along with a GPP such as the 6800 from Motorola. The S2811's main function was to relieve the burden of performing math intensive subroutines from the main processor. In short, it behaved as a math coprocessor and was never used in large quantities in any end product.

A year later, Intel announced an "Analog Signal Processor" which had an analog to digital converter (ADC) and digital to analog converter (DAC) residing on the die. The disadvantage of this processor, 2920, was that it did not have a true multiplier. Multiplication was accomplished by bit shifting and adding partial products; thus the performance of the 2920 was only silghtly better than a GPP. Commercially, the chip was only used in modems.

AT&T's Bell Laboratories introduced the DSP1 in February of 1980. The DSP1 had most of the functional units seen in current DSP processors such as a multiply and accumulator (MAC), parallel addressing unit, control and data memories. Its success was also due to the fact that development tools were available for rapid prototyping. The DSP1 architectural heritage has survived until today, evolving into the DSP1600 processor family from Lucent Technologies.

3.2 First Generation Conventional

This class of architecture represented the first widely accepted DSP processors in the market, appearing in the early 1980's. There were a few key manufacturers that offered processors that share many similar traits. The chips were designed around a Harvard architecture with separate data and program buses for the individual data and program memories respectively. The key functional blocks were the multiply, add and accumulator units, but these processors could only perform fixed-point computations. The software that accompanied the chips had specialized instruction sets and addressing modes for DSP with hardware support for software looping.

These processors were the TMS320C10 from Texas Instruments and the ADSP-2101 from Analog Devices. A graphical representation of the general architecture is depicted in Figure 3.

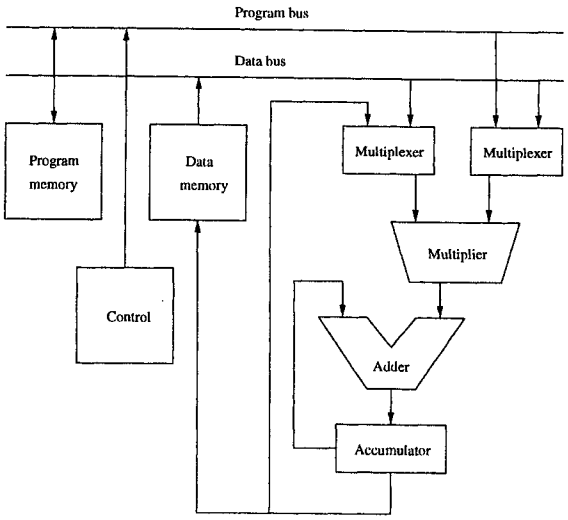

Figure 3: First generation conventional DSP architecture.

3.3 Second Generation Enhanced Conventional

The next stage of development started in the late 1980's/early 1990's, and variants of this architecture have lasted until today. These processors retain much of the design of the first generation but with added features such as pipelining, multiple arithmetic logic units (ALU) and accumulators to enhance performance. The advantage in this is that most processors are code compatible with their predecessors while providing speedup in operations.

Shrinkage of feature sizes also allowed more functional units to be included on the chip. Peripheral device interfaces, counters and timer circuitry, important to data acquisition, are now incorporated in the same die as the DSP. In addition, parallelism could be attained by duplicating key functional units.

The TMS320C20 from Texas Instruments combines both a pipelined architecture and an Auxillary Register Arithmetic Unit (ARAU). In addition, the on-chip data RAM can be configured either as data or program memory. The ARAU can provide address manipulation as well as compute 16-bit unsigned arithmetic, offloading some of the processing from the Central Arithmetic Logic Unit (CALU) [8].

Another example from this era is the Motorola DSP56002. It has a three-stage pipeline and peripherals such as a Serial Communications Interface (SCI), Synchronous Serial Interface (SSI), Parallel Host Interface, Timer/Event Counter and Phase Lock Loop (PLL). There are three RAMs, one for program and storage and two for data. A 24 by 24-bit multiplier is accompanied by two 56-bit accumu-

lators [9].

The latest Lucent Technologies' fixed point DSP16xxx family of processors looks very much like its predecessor introduced in 1990, the DSP1600. Its enhancement is in its data path, which includes two 16 by 16-bit multipliers, an additional 3-input adder and eight 40-bit accumulators [10]. The simplified data path is shown in Figure 4 for comparison with a traditional DSP architecture [10] [11]. Ignoring the shifters and swap/split multiplexers, the basic data flow of the multiplier, ALU and accumulator is conventional.

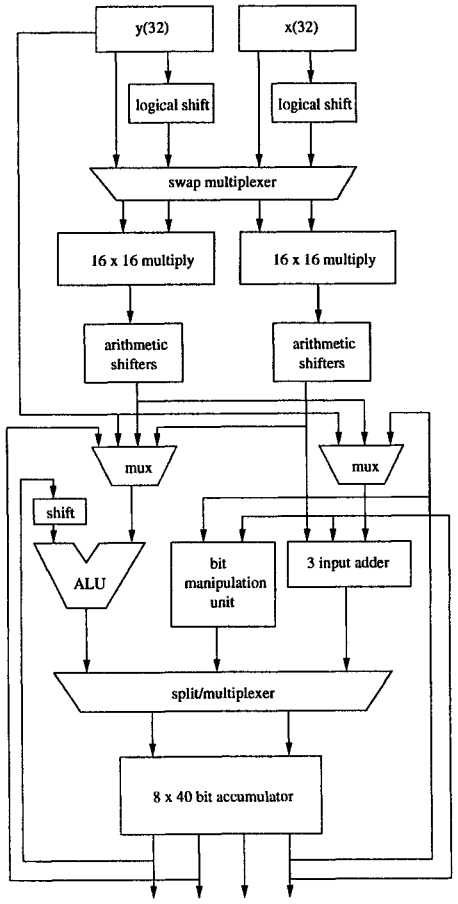

Figure 4: Simplified Lucent Technologies DSP16xxx data path.

3.4 Third Generation Novel Designs

It is about this time that designers were looking at incorporating GPP architectures into DSPs. This speeds up computations while retaining the functions critical to DSP. Today's DSPs execute single instruction multiple data (SIMD), very long instruction word (VLIW) and superscalar operations. On the software side, more advanced debugging and application development tools have been created to complement multiple instruction hardware loops, modulo addressing and user defined instructions [5].

SIMD is characteristic of most high performance GPPs that are also capable of multimedia extensions (MMX) and AltiVec algorithms. SIMD allows one instruction to be executed on many independent groups of data. For SIMD to be effective, programs and data sets must be tailored for data parallel processing, and SIMD is most effective with large blocks of data [5]. In DSP, SIMD may require a large program memory for rearranging data, merging partial results and loop unrolling. The two common ways of implementing SIMD are to use split execution units and multiple execution units or data paths depicted in Figure 5 [5].

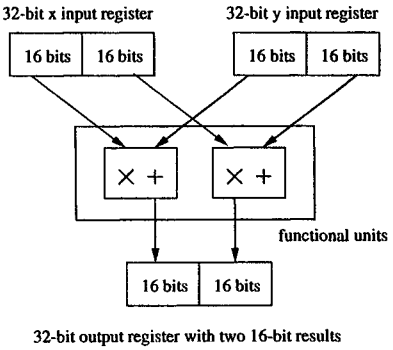

Figure 5: SIMD split execution unit data path.

VLIW processors issue a fixed number of instructions either as one large instruction or in a fixed instruction packet, and the scheduling of these instructions is performed by the compiler [12]. For VLIW to be effective, there must be sufficient parallelism in straight line code to occupy the operation slots. Parallelism can be improved by loop unrolling to remove branch instructions and to use global scheduling techniques, but then a disadvantage of VLIW is low code density if loops cannot be sufficiently unrolled.

Superscalar processors, on the other hand, can issue varying numbers of instructions per cycle and can be schedulled either statically by the compiler or dynamically by the processor itself [12]. As a result, superscalar designs may hold a couple of code density advantages over VLIW. This is because the processor can determine if the subsequent instructions in a program sequence can be issued during execution, in addition to running unschedulled programs.

Recently, we have seen that VLIW has been regaining popularity as a means to improve performance. The latest instruction set architecture co-developed by Intel and Hewlett-Packard, the IA-64, retains much of the VLIW flavor with nuances of CISC and

RISC [13]. This hybrid architecture is called explicitly parallel instruction computing (EPIC) and its main purpose is to permit the compiler to group instructions for parallel execution in a flexible fashion. This VLIW-like concept has also been ported to the DSP domain. The joint-venture between Motorola and Lucent Technologies, StarCore, has announced a new DSP architecture known as variable length execution sets (VLES). Like EPIC, VLES combines CISC, RISC and traditional VLIW into an architecture that tries to eliminate VLIW's two major disadvantages: code density and code scalability/compatibility [14].

The latest DSP chip family from Texas Instruments, TMS320C64x, combines both VLIW and SIMD into their architecture known as VelociTI.2. Instead of varying the length of instruction groups as in EPIC and VLES, this scheme improves the performance of VLIW by allowing execution packets (EP) to span across 256-bit fetch packet boundaries [15]. Each EP consists of a group of 32-bit instructions and the EP can vary in size as seen in Figure 6. By removing these boundaries, no operation (NOP) instructions, common in traditional VLIW, are eliminated and code size is minimized [16].

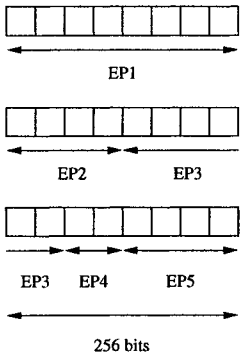

Figure 6: VelociTI.2 architecture allowing boundaryless EP.

The VelociTI.2 architecture implements the SIMD philosophy in providing two register file data cross paths [15]. Each data path includes a general purpose register file and can be used for either data, data address pointers or condition codes. The cross paths, 1X and 2X, allow functional units from one side to access a 32-bit operand from the register file of the other side. This access can be performed simultaneously while the other side's functional units are using operands from its register file. Figure 7 is a simplified diagram of the 'C64x data cross paths. The eight functional units are L1, L2, S1, S2, M1, M2, D1 and D2. DA1 and DA2 are the data address paths.

The debate between the merits of superscalar and VLIW is not only restricted to the GPP universe. LSI Logic Corp. prefers the superscalar approach and is currently the major manufacturer to champion this architecture. We know that a key difference between superscalar and VLIW lies in the hardware and software complexity respectively. The argument presented by LSI Logic is that the VLIW methodology requires a greater programming effort and is at the mercy of the compiler. A change in the processor hardware will also require a corresponding change in the compiler to preserve efficiency [17].

It is also common in DSP programming to hand code assembly code instructions for optimization, especially in loop operations. However, because of VLIW's bundling of instructions, it is difficult for programmers to track the multiple instructions for different functional units in a deeply pipelined structure. The ZSP architecture from LSI Logic, realized as the ZSP16400, intends to reduce programming and compiler complexity without loss of performance by implementing hardware techniques. In addition to adopting a five stage pipelined architecture, the superscalar ZSP16400 can issue up to four instructions per cycle to two MACs and ALUs. Many of the other features are almost identical to a GPP; a one cycle RISC instruction set, load/store architecture, fixed length instructions and hardware scheduling [18]. The ZSP processor architecture is shown in Figure 8 [18].

3.5 Power Considerations

Power has been a major concern lately with increased use of DSPs in mobile computing. Fortunately, the architectural concepts we discussed in this subsection help to play a part in reducing power consumption. Power dissipation is lowered as parallelism is increased with the use of multiple functional units and buses. As a result, power usage is reduced when memory accesses is minimized [19].

Increasing the word size as well as implementing a VLIW-like algorithm allows more data to be fetched per cycle and improves code density. An optimum code density saves power by scaling the instruction size to just the required amount. This is the basic idea of a variable length instruction set processor. The burst-fill instruction cache in the Texas Instruments' TMS320C55x family is flexible enough to be optimized based on the code type. This mechanism improves the cache hit ratio and reduces memory accesses. The core processor can also dynamically and independently control the power feeding the on-chip peripherals and memory arrays. If these arrays and peripherals are not used, the processor switches them to a low power mode and brings them back to

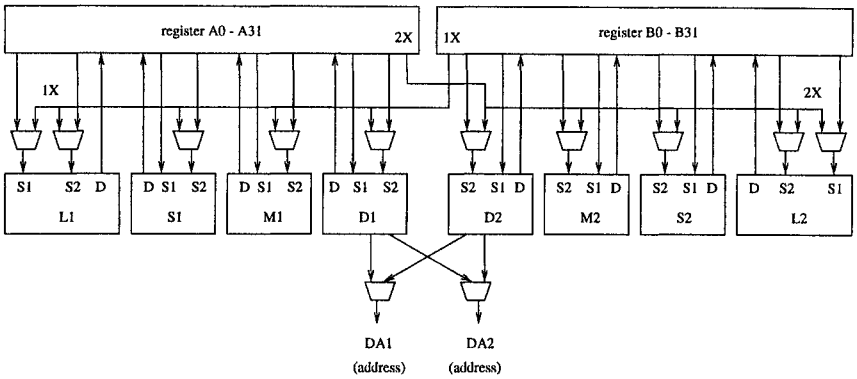

Figure 7: Simplified diagram of the 'C64x data cross paths.

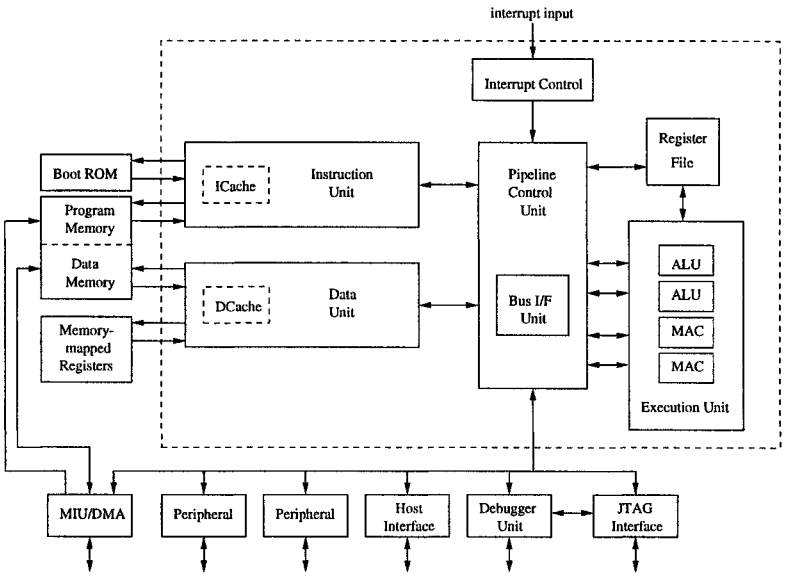

Figure 8: ZSP16400 processor architecture.

full power when an access is initiated without any latency [19]. Another feature unique to this DSP processor is that the CPU, DMA, peripherals, external memory interface (EMIF), instruction cache and the clock generator can be switched to a low activity *IDLE* mode using user controllable software.

We cannot neglect the effects of CMOS process technologies. Motorola's HIP-6 or Lucent's COM2 0.18 micron process will be used to fabricate StarCore's SC140. The six layer copper interconnect chip's core is estimated to draw 198 mW at 1.5 V and 300 MHz and 28 mW at 0.9V and 120 MHz with 3000 raw MIPS [14]. As a comparison, the 'C55x core consumes between 11.2 and 64 mW at 1.2/1.5 V and between 7 and 40 mW at 0.9 V. However, the 'C55x can only compute between 140 and 800 MIPS [20]. In summary, the low voltages are especially important as power consumption is defined by the product of load capacitance, voltage squared, transition frequency and clock frequency.

3.6 Fourth Generation Hybrids

The distinction between a true DSP product and a computer is getting fuzzier, and each day a larger percentage of internet traffic is composed of audio and video data. The problem lies in processing the audio and video at the users' end. Here lies the dilemma; the need for signal processing in a computer based environment, sometimes simultaneously with general purpose computing. Or in other cases, to be capable of processing some digital signals but not wanting to incur the extra cost of a dedicated DSP chip. Devices such as these process control signals as well as digital data.

Instead of a DSP core, the hybrids in the market now incorporate DSP circuitry with a CPU core. A positive side effect is that printed circuit board (PCB) real estate is saved, thereby reducing the size of the product, costs and more importantly the real savings in the power consumed. In the hybrid processor, the GPP instruction set is retained and the additional DSP instructions offload the processing from the GPP core.

The first commercially marketed hybrid, the Hitachi SH-DSP is geared towards cellular applications and control intensive DSP products such as disk drives and modems [21]. The SH-DSP is actually a combination of two cores, the SuperH RISC processor and the DSP core with on-chip peripherals such as timers, communications interfaces, I/O and memory controllers [22]. It is interesting to note that the RISC core implements the von Neumann architecture while the DSP is Harvard. Integer MAC operations are carried out by the RISC core while the DSP core processes the more complex DSP instructions. Both cores are fed from a load-store five-stage pipeline and both general purpose and DSP instructions can be in the same instruction stream.

The I, X, Y and Peripheral buses are the four internal buses with which the cores communicate with the memories and peripheral devices. The I bus is comprised of a 32-bit address and data bus known as the IAB and IDB respectively. This bus is used by both the RISC and DSP core to access any memory block, i.e., X, Y or external. The X and Y bus is only accessible to the DSP core for the on-chip X and Y memories, and each bus has a 15-bit address bus and 16-bit data bus. This address bus is actually padded with a zero in the LSB position since memory accesses are aligned on word lengths. Lastly, the Peripheral bus transmits bidirectional data to the I bus via the Bus State Controller (BSC) [22]. Figure 9 presents a simplified block diagram of the SH-DSP architecture. The DSP enhanced RISC core is represented by the Integer Unit.

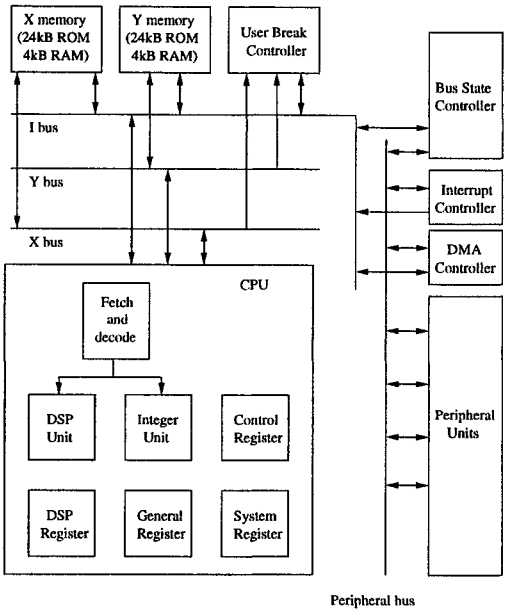

Figure 9: Simplified SH-DSP family processor architecture.

The latest hybrid design comes from Motorola, using their M•Core RISC microcontroller unit (MCU) core in conjunction with a DSP56600 DSP core. The general idea is similar to the SH-DSP in which the hardware architecture is intended for cellular appliances. Both RISC cores are load-store, pipelined and require a single cycle to execute most instructions. The M•Core is a 32-bit processor but its instructions are 16-bit in length so as to reduce power consumption as well as cost [23]. The DSP core has only one

16 by 16-bit MAC but incorporates nested hardware for do loops [24]. In addition to MCU/DSP shared peripherals, the newest member of the DSP566xx family, the DSP56690, has many application specific DSP peripherals such as a baseband port interface, serial audio codec port, Viterbi accelerator and layer-1 encryption. These embedded features are designed to support the most common wireless protocols in use today; GSM, CDMA, TDMA, iDEN, GPRS [25]. To reduce die size and power consuption, the chips will eventually be manufactured in Motorola's HiPerMOS-6 (HIP-6) 0.18 micron technology.

3.7 Next Generations

Based on the current trends seen in DSP development, we can predict that the manufacturers will be following the path of GPP techniques. We have already seen that superscalar, VLIW and pipelining architectural methods, common in GPP, are in use in the latest DSP processors. Process technologies such as copper interconnect and submicron feature size will reduce chip area to enable smaller handheld devices. Since 1985, there has been an increase of almost 150% in DSP processor performance and this trend will certainly continue for the next few years [5].

This means that we can expect to see more on-chip peripherals and memory; the system-on-chip (SOC) may not be too far away. Clock speeds will increase to reduce MAC computation times, but supply voltage must also correspondingly drop to reduce power consumption. In Section 5 of this paper, we will look at a few potential techniques currently developed for GPP that we think may be adopted or modified for use in DSP.

3.8 Product Comparisons

In any study or evaluation, the inevitable question arises; which is the best? We will not attempt to answer the question but leave this as topic of discussion for the reader. Like in GPP studies, there are tradeoffs to be made and perhaps there is no clear choice. Perhaps another way of presenting this question is how to select the right DSP processor for each specific need, since one may be best suited for a particular application but be a poor choice for another purpose [26].

We first consider the arithmetic format since the main purpose of a DSP is to process numerical data. DSPs can be classified into one of two types; fixed or floating point. Most general DSPs are fixed point since the circuitry is easier to design and manufacture. Data widths of fixed point processors are also smaller than floating point, generally 16-bits as compared to 24-bits respectively. Thus fixed point DSPs are cheaper and consume less power than their floating point counterparts and are usually used in high volume, compact sized, low power embedded applications. However, floating point devices are easier to program since floating point arithmetic has more flexibility and a larger dynamic range. The dynamic range is the ratio between the smallest and largest numbers that can be represented. A fixed point program has to be carefully scaled by the user so as to preserve the numeric precision required within the tighter dynamic range.

Speed is a number that tends to get everyone's attention, and the most common number quoted for processors is the clock rate specified in mega/gigahertz. The clock cycle is then the inverse of clock rate. However, it is incorrect just to compare performance purely on this figure as the work done per clock cycle by each processor can vary, even though a significantly high clock rate can outperform an efficient lower clock rate processor. A related metric that is frequently quoted is MIPS or millions of instructions per second. The problem with MIPS is that it is dependent on the instruction set [12] [26]. The VLIW DSP processors issue and execute multiple simple instructions per cycle and these simple instructions typically perform fewer operations than conventional DSP instructions. Another common DSP function is multi-bit data shifting which some processors can implement with a single instruction while others may require multiple one-bit shift instructions. Speed and performance is a topic often debated and we will revisit this issue again in the benchmarking discussion.

We present a case study [27] to illustrate these tradeoffs. Texas Instruments has released their latest DSP processors at about the same time, but each from a different family line tailored to meet specific needs. The 'C6000 family is optimized for the highest performance in terms of speed and simplicity in programming with a high-level language. On the other hand, the 'C5000 family was designed to operate on very low voltages and power consumption. This is to target the consumer digital market and battery powered mobile products, whereas the 'C6000 is better suited for network communications that are powered by fixed line supplies. Some of these applications are modems, wireless base stations and machine vision systems. Architecturally, the 'C6000 incorporates many new techniques such as VLIW and special purpose instructions. This family is rated up to 8800 MIPS at 1.1 GHz. The 'C5000 is an enhanced conventional processor with an additional ALU and

MAC and will deliver between 140 and 800 MIPS while running on 0.9 V. This study shows that the power/performance tradeoff is still a real issue that has not been solved.

4 Benchmarking DSP

As in GPP, DSP users need a basis to compare performance. This being said, benchmarking is not an easy task as there are many factors to consider. Like microprocessors, DSP performance is not governed by just one factor but a series of specifications such as power, clock speed, word size, etc. Combinations of these factors may suit a particular application and not another [28].

Currently as well as in the past, the popular metric for quoting DSP performance is MIPS. However, MIPS is not clear as to what *any* processor is good at for the reasons we have already discussed. As a result, the GPP and computer community founded Standard Performance Evaluation Corporation (SPEC) in 1988 to develop a realistic set of standardized tests for rating computing systems [29]. SPEC is a suite of C language program fragments, applications and kernels that can be run on computer systems. The main advantage of suite testing is that the negative effect of one benchmark is offset by the rest of the programs in the suite [12]. Unfortunately, executing SPEC benchmarks on embedded and DSP processors would also be unrealistic as the SPEC suite of programs are general in nature and are not tailored for DSP. Also, it is common to manually code sections of DSP algorithms in assembly for optimum performance because C complilers are inefficient in translating C code to DSP assembly [2].

It is this reason that in 1994, an independent DSP technology analysis and software development firm, Berkeley Design Technology, Inc. (BDTi), developed a set of DSP algorithm kernel benchmarks [30]. These algorithm kernels are implemented in hand optimized assembly and so are not architecture dependent. It is even possible to run these benchmarks on GPPs for comparison with DSP processors. Unlike SPEC, BDTi implements and verifies these benchmarks but will publish these results on their web site without charge.

BDTi's concept is to strictly define these benchmarks and to allow only realistic optimizations. These benchmarks are described in Table 1 [2]. The algorithm optimizations are firstly for speed and then memory use except for the control benchmark in which memory is optimized first.

The results of the benchmarking can be presented in terms of cycle counts or execution time and BDTi

Function	Description
Real Block FIR	Finite impulse filter operating on real data.
Complex Block FIR	FIR filter operating on complex data.
Real Single Sample FIR	FIR filter operating on single sample of real data.
LMS Adaptive Filter	Least mean square adaptive filter operating on single sample of real data.
Two-Biquad IIR	Infinite impulse response filter operating on single sample of real data.
Vector Dot Product	Sum of pointwise multiplication of two vectors.
Vector Add	Pointwise addition of two vectors resulting in third vector.
Vector Maximum	Locate the value of the maximum vector.
Viterbi Decoding	Decode a block of convolutionally encoded bits.
256-Point-In-Place FFT	Fast Fourier transform to convert time domain signal to frequency domain.
Bit Unpack	Unpack variable length data from a bit stream.
Control	Sequence of control operations e.g. test, pop, push, branch and bit manipulation.

Table 1: BDTi benchmark suite.

has chosen the latter in their reports. In some cases, a single numerical figure is required for quick comparison and BDTi has formulated a score known as the BDTImark2000, based on their application kernels suite. The BDTImark2000 represents DSP speed and so a higher figure denotes better performance. The limitation of this score is that it is only an estimate of the processor's execution speed and not of specific applications. Hence a processor's strength in a particular application may not be reflected in its BDTImarks score [31]. Energy consumption is obtained by multiplying the typical power usage of the processor by the execution time of a benchmark. Although it may be a less accurate method, it is less time consuming and easier to implement [2].

Another aspect of characterizing performance by BDTi is application profiling. The results give a better picture of how kernels are actually used in applications. Essentially, profiling calculates the execution frequency of kernels in applications and it presents developers the relative weightings of each algorithm kernel benchmark in a certain application. The comparisons between processors can be made by comparing the product of kernel execution times and its number of occurences in the application.

Recently a path taken by the industry was to follow in the footsteps of SPEC. In April of 1997, a consortium of 21 manufacturers led by a trade publication, Electronic Design News (EDN), started work on a new set of benchmarks targeted towards embedded processors. This group is formally known as the EDN Embedded Microprocessor Benchmark Consortium (EEMBC), pronounced as "embassy". Their aim was to develop a group of tests that can be easily ported to different processor architectures, hardware platforms and operating systems while measuring specific areas of the processor's performance independently [32]. The benchmark scores are reported individually for each test, unlike SPECint95 or SPECfp95 [33]. However, EEMBC also provides a single composite scores for faster comparisons between processors. This score is determined by assigning a weight to each of the test in the suite. The EEMBC composite scores are created for each of the five application test suites, for example, the composite score for telecommunications is called $Tele_{mark}$, and for networking, Net_{mark}.

The EEMBC suite of tests are written in ANSI C for portability and are similar to BDTi's idea of algorithm kernels. Since the embedded market is diverse, the benchmark algorithms are divided into five categories so as to allow testers to quantify their products with relevant tests, but there is nothing preventing testers to run their processors on other test categories. These five areas are automotive/industrial, consumer, networking, telecommunications and office automation. The six telecommunications tests would apply for purely DSP testing, and these algorithms are listed in Table 2 [34].

Algorithm	Description
Autocorrelation	A voice data array and number of lags (time delays) input is used to calculate an array of sum of products for each lag.
Bit Allocation	The input data bits are allocated equally over a series of buffers (frequency bins) using a water level algorithm.
Inverse Fast Fourier Transform (iFFT)	Converting frequency domain data into time domain data using iFFT.
Fast Fourier Transform (FFT)	Converting frequency domain data into time domain data using FFT.
Viterbi Decoder	To recover an output data packet from an encoded input data packet by decoding.
Convolutional Encoder	An output data stream is generated from an input data stream using a linear shift register and table lookup.

Table 2: EEMBC Version 1.0 telecommunications test suite.

The testing procedure is similar to SPEC in which DSP vendors will perform the benchmarking and report the results to EEMBC. It becomes official only after EEMBC verifies the results with the identical test parameters used by the manufacturers in EEMBC's testing lab, EEMBC Certification Laboratories (ECL). Both unoptimized and "tweaked" results can be reported to EEMBC [28]. However, EEMBC allows anyone to run these tests at any time and publish the results, but without EEMBC's official stamp of approval. Scores are currently reported in terms of iterations per second so higher figures represent better performance.

Prior to these proposed standards, DSP manufacturers often had their own set of benchmarks and methods of obtaining performance figures. Most of these benchmark tests are still based on DSP algorithms such as FIR filtering, MAC operations and Viterbi decoding, but usually optimized and coded in the chip's assembly language. More often than not, the specifications are still quoted in MIPS. It is clear from the discussions at the begining of this section that other factors besides MIPS or even speed may have to be used to characterize DSP processors. Lucent Technologies has proposed a new measure of DSP performance called the Applications Cube [35] seen in Figure 10. The three parameters of the cube, power, performance and cost, define the axes of the cube in three dimensions. Each DSP processor when quantified in these three terms for a particular application, results in the volume of the cube and the smaller the volume, the more suited the DSP is to the appplication or function. The power parameter is represented by milliamperes (mA) per function for a certain voltage. Performance is still measured in MIPS per function and the cost is calculated as code size per function.

To round off our discussion on benchmarking, we present the different benchmark scores [5] [6] [36] [37] [38] [39] [40] [41] [42] for selected fixed point DSP processors which we have discussed, for comparison in Table 3. The EEMBC scores are not listed because some DSP processors have not been formally tested with EEMBC as of yet.

Although MIPS and to a certain extent, BDTImarks are not truly accurate benchmarks, they corelate and it can be seen that in general, a high MIPS figure will also result in a high BDTImarks number.

5 Future Directions

The trend to port GPP architectural techniques will most likely continue. At the same time, new methods of implementing these techniques are also being carried out for GPP. Rather than lagging behind GPPs, we see this as an opportunity for these ideas to be applied to DSP concurrently.

An interesting idea that has been explored in digital circuits is the concept of asynchronous logic design. Although asynchronous circuits have been researched in the 1950's, the study is regaining popularity because of the positive benefits derived from the technique. Asynchronous circuits have the potential of providing high performance in terms of reducing data dependent delays, improving the elasticity of pipelines [43] and a higher operating speed [44]. More importantly for the area of mobile computing and Internet appliances is the benefit of low power dissipation and electromagnetic compatibility (EMC) [43] [44] [45].

The power savings in CMOS circuits is a result of minimizing signal transitions. Asynchronous circuits by nature do not use power in the idle areas of the circuit and can instantly activate circuits, modules and storage elements when required, even without software assistance [46]. In circuits where radio frequencies are highly sensitive such as pagers, cellphones and wireless devices, the frequency spectrum harmonics of the supply current can cause interference in the signal reception. The clock in a synchronous system can contribute significantly to this frequency spectrum whereas an asynchronous system would not [43]. We have seen examples of asynchronous logic successfully applied to application specific DSP such as pagers [45] and hearing aids [44]. We believe that future work can be done to incorporate this technique to general purpose DSP processors. The execution pipeline of the AMULET2e embedded controller [46] has a shift and multiply functional unit providing data to an ALU. This bears a close resemblance to the architecture of a conventional DSP processor and modifications may possibly convert the AMULET2e for DSP use.

While not related to asynchronism, perhaps a related circuit technique is wave-pipelining. Although its roots are in pipelining, the "reduced" use of registers, hence clocking elements, makes processing a series of inputs into a combinational circuit asynchronous. The pipelining effect is caused by the inherent delays (resistances and capacitances) of the circuit. The advantages of wave-pipelining are a simplified clock distribution scheme and a very high pipeline rate. Wave-pipelining has been utilized in specialized DSP circuits, for example multipliers [47] and adaptive filters [48], but detailed studies into its use in general purpose DSP processors could be carried out further.

Regarding software issues, we must not overlook the importance of compilers as well. The popularity and ease of use of commercial general purpose DSP processors is largely due to the availability of development tools [5]. However the compilers and tools are still inefficient in certain areas and hence require hand optimization in the resulting assembly code for better performance. More research can be done to improve software tools to reduce manual coding; perhaps the first step is to reevaluate the choice of programming languages.

6 Conclusion

We began by briefly reviewing a digital signal processing algorithm that influences the building blocks

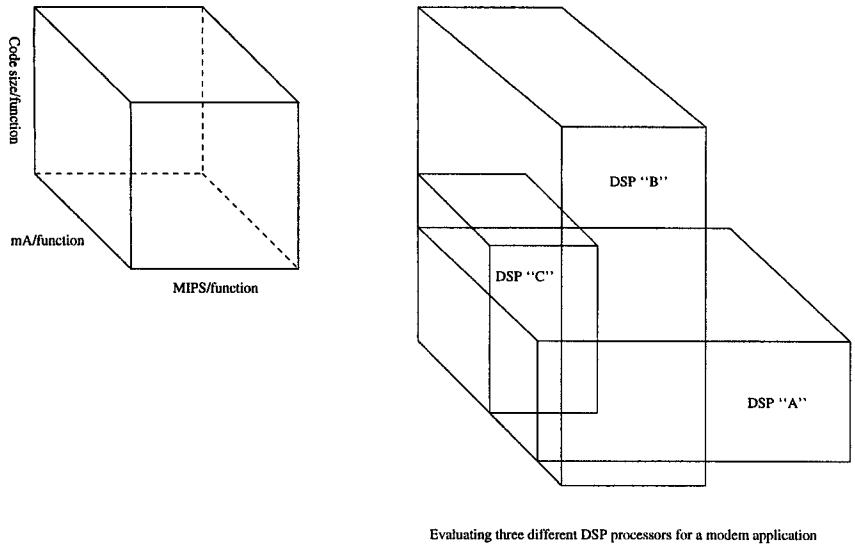

Figure 10: The Lucent Technologies Applications Cube.

Manufacturer	Processor Family	Speed (MHz)[1]	Architecture	MIPS[2]	BDTImark2000
Texas Instruments	TMS320C1x	20	conventional	8.77	n/a
Hitachi	SH2-DSP	100	hybrid	100[3]	280
Motorola	DSP563xx	150	hybrid	150	450
Analog Devices	ADSP-21xx	75[4]	conventional	75	230
Texas Instruments	TMS320C54xx	160	enh. conventional	160	500
Lucent Technologies	DSP16xxx	170	enh. conventional	170[5]	810
LSI Logic	ZSP164xx	200	superscalar	400	n/a
Texas Instruments	TMS320C62xx	300	VLIW	2400	1920

Table 3: Comparison of benchmark scores for fixed point commercial DSP processors.

in a DSP processor. Most DSP operations can be simplified into multiplications and additions, so the MAC formed the main functional unit in early DSP processor. Designers later incorporated techniques from GPPs to enhance the performance of DSPs. These architectures included pipelining, VLIW, superscalar, and although not discussed in this paper, branch prediction and speculation. These ideas will stay with DSP processors and we think that the differences between the DSP and GPP will become more blurred in the future.

There has been a drive to develop new benchmarking schemes to measure performance since MIPS is becoming less acccepted as a reliable metric. We see two different groups taking an identical approach by using application kernels and a commercial vendor that also measures performance in terms of power consumption, performance and cost. Power issues are gaining importance as DSP processors are incorporated into handheld, mobile and wireless devices. Finally, we discussed techniques such as asynchronous digital circuits and wave-pipelining that could be exploited in the future for performance as well as power awareness.

[1]Instruction clock speed for fastest device in the family
[2]MIPS for corresponding device speed
[3]Assuming one instruction per clock cycle
[4]Input clock is 37.5 MHz
[5]10 ns multiply and accumulate instruction cycle time

7 Acknowledgments

The authors would like to thank Dr. Stephen McAleavey for his time and effort in directing the authors to relevant sources of DSP material and for his insights into the subject.

References

[1] Forward Concepts, "Forward Concepts DSP Market Bulletin", World Wide Web, `http://www.forwardconcepts.com/press25.htm`, 7th Apr. 2000.

[2] Berkeley Design Technology, Inc., "Evaluating DSP Processor Performance", World Wide Web, `http://www.bdti.com/articles/benchmk_2000.pdf`, 2000.

[3] V. K. Madisetti, *VLSI Digital Signal Processors: An Introduction to Rapid Prototyping and Design Synthesis*, Boston, MA: Butterworth Heinemann, 1995.

[4] J. S. Thompson, S. K. Tewksbury, "LSI Signal Processor Architecture for Telecommunications Applications", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol ASSP-30, No. 4, pp. 613-631, Aug 1982.

[5] Berkeley Design Technology, Inc., "The Evolution of DSP Processors", World Wide Web, `http://www.bdti.com/articles/evolution.pdf`, 2000.

[6] L. Geppert, "High-Flying DSP Architectures", *IEEE Spectrum*, pp. 53-56, Nov 1998.

[7] J. R. Boddie, "The First Single-Chip DSP", World Wide Web, `http://www.lucent.com/micro/dsp/single.html`, 2000.

[8] Texas Instruments, Inc. *TMS320C2x User's Guide*, Austin, TX, Jan 1993.

[9] Motorola, Inc. *DSP56002 Technical Data*, Rev. 3, AZ, 1996.

[10] J. Bier, "DSP16xxx Targets Communications Apps", *Microprocessor Report*, Vol. 11, No. 12, pp. 11-15, Sep 1997.

[11] Lucent Technologies, Inc., *DSP16000*, Allentown, PA, Sep 1997.

[12] J. L. Hennessy, D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 2nd ed., San Francisco, CA: Morgan Kauffman Publishers, Inc., 1996.

[13] L. Gwennap, "Intel, HP Make EPIC Disclosure", *Microprocessor Report*, Vol. 11, No. 14, pp. 1, 6-9, Oct 1997.

[14] T. R. Halfhill, "StarCore Reveals Its First DSP", *Microprocessor Report*, Vol. 13, No. 6, pp. 13-16, May 1999.

[15] Texas Instruments, Inc. *TMS320C64x Technical Overview*, Dallas, TX, Feb 2000.

[16] Texas Instruments, Inc. *TMS320C64x Digital Signal Processor Core*, Dallas, TX, 2000.

[17] LSI Logic Corp., "Compiler-friendly Architectures for DSP", World Wide Web, `http://www.zsp.com/compiler.html`, 2000.

[18] LSI Logic Corp., "An Overview of the ZSP Architecture: White Paper", World Wide Web, `http://www.zsp.com/pdf/zsp_whitepaper_v2.pdf`, 1999.

[19] Texas Instruments, Inc., *TMS320C55x Technical Overview*, Dallas, TX, Feb 2000.

[20] Texas Instruments, Inc., *TMS320C55x Digital Signal Processor Core*, Dallas, TX, 2000.

[21] Hitachi America, Ltd., "SuperH - Digital Signal Processor (SH-DSP)", World Wide Web, `http://www.hitachi.com/rd/Micropro.html`, 1999.

[22] Hitachi Micro Systems, Inc., *SH-DSP Microprocessor Overview*, Ver. 0.1, Nov 1996.

[23] J. Turley, "M•Core Shrinks Code, Power Budgets", *Microprocessor Report*, Vol. 11, No. 14, pp. 12-15, Oct 1997.

[24] Motorola, Inc., *DSP56651 16-bit Digital Signal Processor Family Manual*, Austin, TX, 1996.

[25] T. R. Halfhill, "Motorola Cellular DSP Does It All", *Microprocessor Report*, Vol. 13, No. 16, Dec 1999.

[26] Berkeley Design Technology, Inc., "Choosing a DSP Processor", World Wide Web, `http://www.bdti.com/articles/choose_2000.pdf`, 2000.

[27] Texas Instruments, Inc., *TMS320C64x DSP and TMS320C55x DSP Cores: Do Something Phenomenal*, Dallas, TX, 2000.

[28] J. Turley, "What's the Best Way to Benchmark?", *Microprocessor Report*, Vol. 13, No. 3, pp. 3, Mar 1999.

[29] The Standard Performance Evaluation Corporation, "SPEC's Background", World Wide Web, http://www.specbench.org/spec/, 1999.

[30] Berkeley Design Technology, Inc., "Independent DSP Benchmarks: Methodologies, Results, and Analysis", World Wide Web, http://www.bdti.com/articles/esc99/benchmar/index.htm, 1999.

[31] Berkeley Design Technology, Inc., "The BDTImark: A Measure of DSP Execution Speed", World Wide Web, http://www.bdti.com/articles/wtpaper_1999.pdf, 2000.

[32] J. Turley, "Embedded Benchmark Work Under Way", *Microprocessor Report*, Vol. 12, No. 5, pp. 13, Apr 1998.

[33] T. R. Halfhill, "Embedded Benchmarks Grow Up", *Microprocessor Report*, Vol. 13, No. 8, pp. 1, 6-9, Dec 1999.

[34] EDN Embedded Microprocessor Benchmark Consortium, "Telecommunications Benchmark Datasheets", World Wide Web, http://www.eembc.org/Benchmark/datasheet/Telecomm/default.asp, 2000.

[35] Lucent Technologies, Inc., *A New Measure of DSP*, Allentown, PA, Sep 1997.

[36] Texas Instruments, Inc., "SMJ320E14 DIGITAL SIGNAL PROCESSOR", World Wide Web, http://www.ti.com/sc/docs/products/dsp/smj320e14.html, 2000.

[37] Analog Devices, Inc., *DSP Microcomputer ADSP-2189M*, Norwood, MA, 2000.

[38] Hitachi America, Ltd., "SH-DSP SH7410 Programming Manual", World Wide Web, http://www.semiconductor.hitachi.com/products/product_abstract.cfm?p_id=214, 1997.

[39] Lucent Technologies, Inc., *DSP16210 Digital Signal Processor*, Allentown, PA, Sep 1997.

[40] Berkeley Design Technology, Inc., "Pocket Guide to DSP Processors and Cores", World Wide Web, http://www.bdti.com/pocket/pocket.htm, 2001.

[41] Texas Instruments, Inc., *STRATEGIC DSP PLATFORM PRODUCT GUIDE*, Dallas, TX, 2000.

[42] EDN Embedded Microprocessor Benchmark Consortium, "Telecommunications Benchmark Scores", World Wide Web, http://www.eembc.org/Benchmark/score/scorereport.asp, 2001.

[43] C. H. Van Berkel, M. B. Josephs, S. M. Nowick, "Scanning the Technology: Applications of Asynchronous Circuits", *Proceedings of the IEEE*, Vol. 87, No. 2, pp. 223-233, Feb 1999.

[44] L. S. Nielsen, J. Sparsø, "Designing Asynchronous Circuits for Low Power: An IFIR Filter Bank for a Digital Hearing Aid", *Proceedings of the IEEE*, Vol. 87, No. 2, pp. 268-281, Feb 1999.

[45] J. Kessels, P. Marston, "Designing Asynchronous Standby Circuits for a Low-Power Pager", *Proceedings of the IEEE*, Vol. 87, No. 2, pp. 257-267, Feb 1999.

[46] S. B. Furber, J. D. Garside, P. Riocreux, S. Temple, P. Day, J. Liu, N. C. Paver, "AMULET2e: An Asynchronous Embedded Controller", *Proceedings of the IEEE*, Vol. 87, No. 2, pp. 243-256, Feb 1999.

[47] D. Gosh, S. K. Nandy, "A 400 MHz Wave-Pipelined 8 x 8-bit Multiplier in CMOS Technology", *Proceedings of the IEEE International Conference on Computer Design*, 1993, pp. 198-201.

[48] W. P. Burleson, C. Y. Lee, E. J. Tan, "A 150 MHz Wave-Pipelined Adaptive Digital Filter in 2μm CMOS", *VLSI Signal Processing Workshop, VII*, 1994, pp. 296-305.