DSPS ADAPT TO NEW CHALLENGES

A WHITE PAPER

BY

BERKELEY DESIGN TECHNOLOGY, INC.

WWW.BDTI.COM

FEBRUARY 2003

Mainstream digital signal processor (DSP) architectures have undergone radical changes over the last five years. Where once all DSP architectures were quite similar, now there are a wide range of architectural approaches: VLIW, SIMD, and DSP-enhanced general-purpose processors, to name but a few. This white paper explores some of the key developments that have shaped current DSP architectures and looks at the some of the creative new approaches processor architects are taking to create faster, more economical, more energy-efficient DSPs.

Introduction

Mainstream digital signal processor (DSP) architectures have undergone radical changes over the last five years. This article explores some of the key developments that have shaped current DSP architectures and looks at some of the creative new approaches processor architects are taking to create faster, more economical, more energy-efficient DSPs.

DSPs as Filtering Machines

Up until the mid 1990's, most mainstream commercial DSPs looked pretty much alike. They were designed primarily with the needs of digital filtering (FIR and IIR) and similar algorithms in mind, with a few extra features to boost performance on FFTs. Processors typically included one MAC (multiply-accumulate) unit, one ALU (arithmetic-logic unit), and possibly a shifter, along with other hardware features that were tightly coupled to the requirements of these target algorithms.

Instruction sets were comprised of compound, complex instructions that crammed multiple operations into a single, fairly narrow (e.g., 16-bit) instruction word. For example, most DSPs provided an instruction that specified a multiply-accumulate, pointer updates, and two data moves from memory (which is what it takes to compute one tap of an FIR filter). Because of the narrow instruction width, it was necessary to limit the combinations of operations supported and also to limit instruction options—such as which registers could be used as sources or destinations. This approach yielded a high degree of efficiency on typical DSP tasks (in terms of speed, memory usage, energy consumption, and chip cost) but at the price of inflexible and awkward instruction sets. Developing an efficient compiler for these architectures was practically impossible. Thus, nearly all programming for them was done using assembly language and, because of the convoluted instruction sets, this was a painful and time-consuming process.

Nearly all programming for conventional DSP processors was done using assembly language, and because of the convoluted instruction sets this was a painful and time-consuming process.

DSP Apps Get Bigger, Hungrier

In the mid 1990s, several developments began to push DSP processor architectures in new directions. Perhaps most important among these trends, more and more applications began to incorporate DSP functionality and the demand for processors with DSP capabilities skyrocketed. Some of these new applications hungered for processing speeds beyond that available from traditional DSPs. Cell phones and telecom infrastructure emerged as "killer apps" for DSPs, and their signal processing workloads included some algorithms that bore very little resemblance to filters or FFTs, such as Viterbi decoding and unpacking bit streams. As DSP applications began to get bigger and more complicated, programming in assembly language became more than just annoying—it was starting to become a crippling limitation of the technology.

The Debut of VLIW for DSP

The changing requirements of DSP applications led, in 1996, to a pivotal new architecture from Texas Instruments (TI), the TMS320C62xx. With the 'C62xx, TI set its sights on producing the world's fastest and most compiler-friendly mainstream DSP processor. To meet these goals, TI scrapped nearly all of the attributes of conventional DSP processors. Where conventional DSPs executed one instruction per cycle, the 'C62xx was a "VLIW" (very long instruction word) architecture. This meant that it executed many (up to eight, in this case) instructions in parallel as part of a "long instruction word" rather than executing a single instruction at a time. Where conventional processors used complex, compound instructions, the 'C62xx used very simple, compiler-friendly, RISC-like instructions and relied on executing many of them per cycle to achieve its performance. Figure 1 shows a typical 'C62xx very long instruction word alongside a typical conventional DSP instruction sequence.

Unlike the highly tailored architectures of conventional DSPs (which were a nightmare for compiler developers) the 'C62xx's microarchitecture and instruction set was much more general-purpose. With its two multipliers and four ALUs, however, it was clearly intended to offer ferocious performance on DSP algorithms. (In fact, at the time, the 'C62xx was one of only a few processors that could execute two multiplies at a time; nowadays two is the minimum offered by new DSP architectures and four or even eight are becoming more common.)

Unlike the highly tailored architectures of conventional DSPs (which were a nightmare for compiler developers) the 'C62xx's microarchitecture and instruction set was much more general-purpose.

One other noticeable difference lurked within the 'C62xx's microarchitecture. Where most DSP processors used shallow pipelines (typically three to five stages), the 'C62xx employed an eyebrow-raising eleven stages. The processor's deep pipe and simplified instruction set allowed it to execute at then unheard-of clock speeds—upwards of 200 MHz. This clock speed, combined with the processor's high level of parallelism (which came from its ability to execute multiple instructions in parallel), gave it speeds on DSP

```
RPT #ntaps
        MAC *AR2+, *AR3+, A
```

Inner loop of an FIR filter on a conventional DSP (the Texas Instruments TMS320C54xx). The RPT instruction controls the loop, and the MAC instruction is executed once per cycle.

```
LOOP:
 ADD    .L1 A0,A3,A0
||ADD   .L2 B1,B7,B1
||MPYHL .M1X A2,B2,A3
||MPYLH .M2X A2,B2,B7
||LDW   .D2 *B4++,B2
||LDW   .D1 *A7--,A2
||[B0]   ADD .S2 -1,B0,B0
||[B0]   B .S1 LOOP
; LOOP ends here
```

Inner loop of an FIR filter on a VLIW DSP (the Texas Instruments TMS320C62xx). All eight instructions are executed in parallel in a single cycle. Together, these instructions compute two MACs.

Figure 1: These code snippets illustrate just how different the 'C62xx's instruction set is in comparison to that of a representative conventional DSP. Note that the 'C62xx computes two taps per cycle; the 'C54xx computes only one. The "||" notation indicates instructions that are to be executed in parallel as part of a very long instruction word.

algorithms that eclipsed those of conventional DSPs. Figure 2 illustrates the relative speeds of high-performance VLIW-based DSPs vs. conventional DSPs over the last two decades.

All Silver Linings Have Clouds

The 'C62xx was designed to be the fastest mainstream DSP around and it was quite successful in that respect. It suffered from some key flaws, however; probably the most important of which was that it had a hefty appetite for memory. Each of the 'C62xx's simple instructions consumed 32 bits and, because the instructions were simple, it took several of them to do the same work as was done by a single instruction on a conventional DSP. As a result, the 'C62xx's memory use was far higher than that of traditional DSPs. High memory usage translates into bigger die sizes, higher chip cost, and often higher energy consumption. Thus, although the 'C62xx offered some unique advantages and achieved a huge performance leap, it was not really suitable for cost- or power-sensitive applications. This limited its appeal since some of the most important markets for DSPs fall into those categories.

Because of its impressive speed and the promise of better compilability, the 'C62xx profoundly influenced the direction of subsequent DSP architectures. In fact, within a couple of years of the 'C62xx's introduction, all of the major DSP processor vendors were fielding VLIW architectures, each of which had its own approach to dealing with the challenges of reducing memory use and energy consumption. Some of the newest VLIW-

Conventional DSP architectures haven't been abandoned entirely-yet-but the trend is overwhelmingly in the VLIW direction.

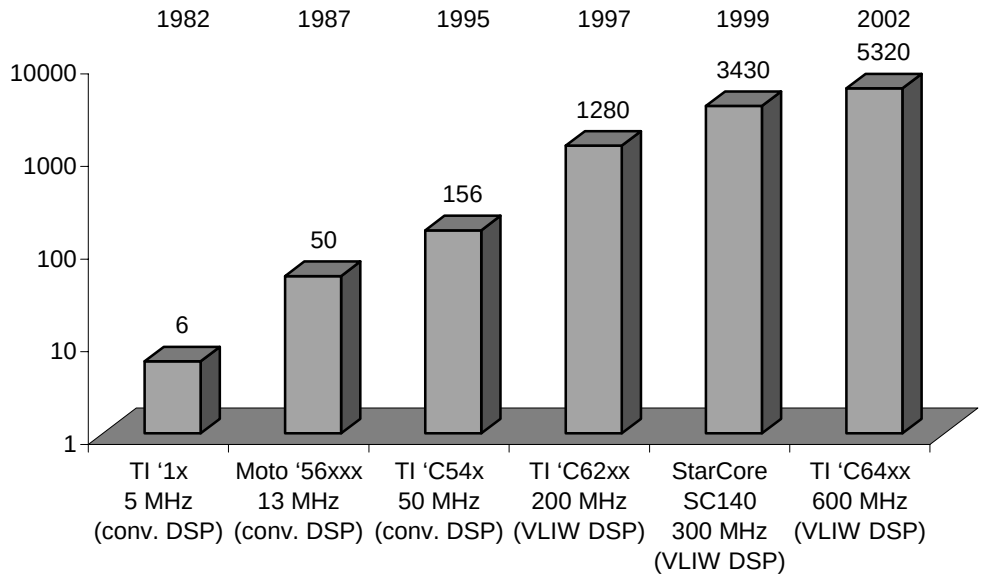


Figure 2: This chart of BDTImark2000™ scores illustrates the dramatic speed increases achieved by DSP processors over the last two decades through a combination of faster clock rates and more powerful architectures. (The BDTImark2000 is a DSP speed metric based on a processor's results on BDTI's suite of DSP benchmarks; higher is faster.)

based processors (for example, Analog Devices' Blackfin and TI's 'C55xx) have memory use and power consumption profiles that are even better than those of conventional DSPs. Conventional DSP architectures haven't been abandoned entirely—yet—but the trend is overwhelmingly in the VLIW direction.

Mixed-Width Instructions Find Favor

Where the 'C62xx used exclusively 32-bit instructions, today's DSPs (whether VLIW or not) often support multiple instruction widths to help keep memory usage low. For example, StarCore's VLIW-based SC140 uses 16-bit instructions with optional 16- or 32-bit prefixes to extend functionality where needed. The idea is that the processor can use short instructions that support fewer parallel operations and limited instruction options where possible (as in decision-making processing, often called "control code") and save the wider, more powerful instructions for DSP algorithm inner loops, which typically require greater parallelism and instruction flexibility. This approach tends to be very effective for trimming memory use without compromising the regularity (and hence, compilability) of the instruction set. Figure 3 shows the memory usage of various popular DSP processors ranging from conventional DSPs to VLIW-based DSPs with mixed-width instruction sets.

Another way to lower memory use in VLIW processors is to revert, to a small degree, to supporting multi-operation instructions—for example, to provide an instruction that performs a multiply-accumulate rather than requiring separate multiply and add instructions. The tradeoff here is compilability vs. memory usage and speed. Simpler instructions are more compiler-friendly, but you need more of them (and thus more memory) to perform a given task; more complex instructions get more work done per cycle for less memory—

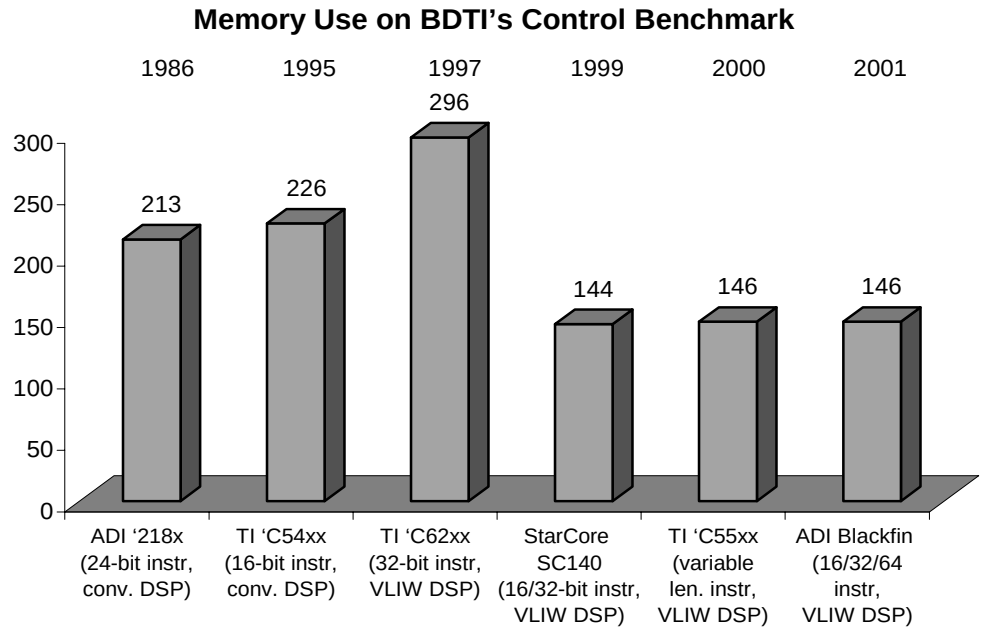


Figure 3: This chart of BDTImark2000™ scores illustrates the dramatic speed increases achieved by DSP processors over the last two decades through a combination of faster clock rates and more powerful architectures. (The BDTImark2000 is a DSP speed metric based on a processor's results on BDTI's suite of DSP benchmarks; higher is faster.)

but are harder for compilers to use efficiently. The newest VLIW architectures typically include some multi-operation instructions (often using SIMD techniques, described below) but these instructions are not nearly as complex as the instructions found on conventional DSPs.

Cloning Operations with SIMD

One of the reasons that the VLIW approach has become so popular is that it is able to yield strong performance across a range of algorithms. This is because VLIW processors have the flexibility to execute many different combinations of parallel operations—such as a multiply-accumulate, plus an add, plus a shift, plus a memory move or two. However, there are many algorithms that don't require this flexibility, whose performance can benefit from simultaneously executing multiple instances of the same operation—such as two or four multiply-accumulates at a time. This is the case in many filtering algorithms, for example. One way of boosting a processor's performance on these algorithms is to support "SIMD" (single instruction, multiple data) instructions.

In a SIMD instruction, the specified operation is executed on two (or more) operand sets to produce multiple outputs. This is often accomplished by having the processor treat registers as containing multiple data words; for example, a 32-bit register can be treated as containing two 16-bit data words, as illustrated in Figure 4. The benefit of SIMD is increased parallelism, and thus increased performance. Most DSPs include some support for SIMD, and many combine SIMD and VLIW. TI's 'C64xx, Analog Devices' TigerSHARC and Blackfin families, and StarCore's SC140 are all examples of this trend.

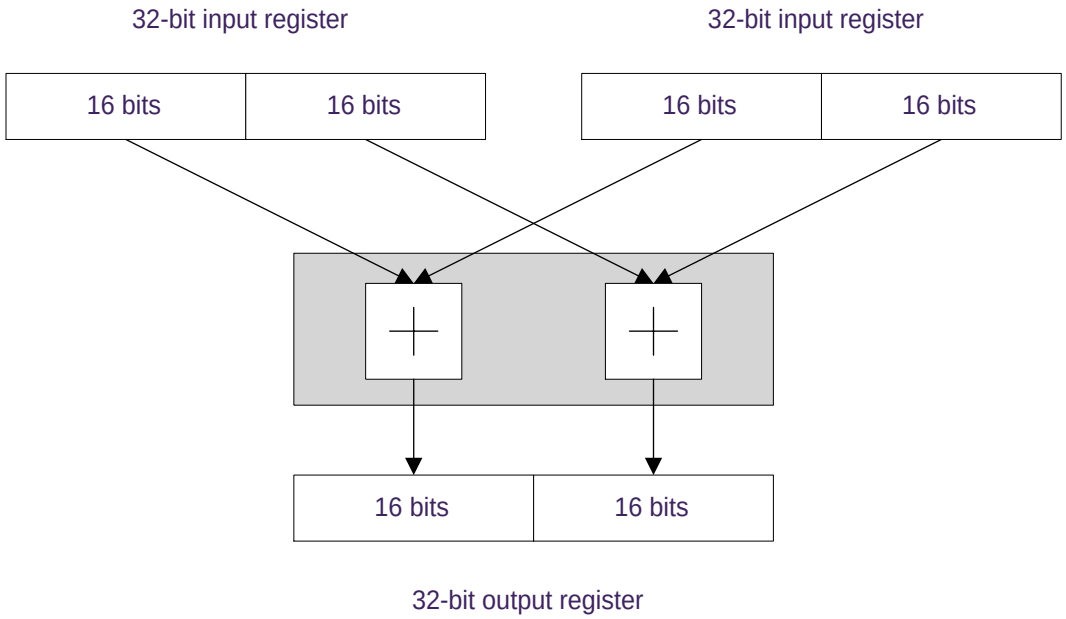


Figure 4: This diagram provides an example of the SIMD approach. The processor executes a single instruction specifying dual 16-bit adds by treating the two 32-bit input registers as each containing a pair of 16-bit input operands. The dual additions produce two 16-bit results packed into a single 32-bit register.

Flexible Data Widths

DSPs today are expected to handle not only a wider range of algorithms than in the past, but also—relatedly—a wider range of data types. Some image and video processing tasks are well served by 8-bit data; 16-bit data is common in telecommunications; and 24- or 32-bit data is common in high-fidelity audio. Whereas five years ago most high-volume DSPs supported only 16-bit fixed-point data, today's DSPs often support 8-bit, 16-bit, and sometimes 24- or 32-bit fixed-point data types. Processors that support multiple data widths provide the programmer with the flexibility to adjust data precision as needed at various points in an algorithm, thus reducing memory use.

Add-Ons and Customizations

One of the challenges facing today's DSP architects lies in trading off flexibility against application specificity. Architectures that are tuned for a specific application can be highly efficient in that application, but lose the flexibility to perform well in other applications. In addition, it's easier to design an efficient compiler for an architecture that's simple and general-purpose than for one with many application-specific architectural features.

One way to balance these tradeoffs is to use a DSP architecture that is fairly general and flexible and augment it with one or more highly specialized coprocessors. This is what TI has done with some members of the 'C64xx family which include on-chip Viterbi and turbo decoding coprocessors. This approach can deliver a noticeable performance boost with little effort on the part of the chip user—but only if the target application's requirements are a good fit for the capabilities of the coprocessor.

Another approach is to allow the chip user (the system designer) to customize the instruction set with application-specific instructions. This is the approach taken by Adelante Technologies, for example, with its Saturn DSP architecture. In addition to a typical 16/32-bit mixed-width DSP instruction set, Saturn supports user-defined "ASIs" (application-specific instructions). ASIs are 96 bits wide and enable the user to create instructions that use Saturn's execution units (four ALUs and two MAC units) in combinations not supported by the built-in instruction set. ASIs are stored in a special on-chip memory, which can be a ROM or a dynamically updatable RAM. Using customized instructions can increase the processor's performance in algorithm inner loops by allowing the user to create an instruction that performs exactly the operations needed by the algorithm. Of course, the range of operations that can be specified in a custom instruction is constrained by the computational resources available on the processor, and therefore there are inherent limitations on the performance gains that can be realized. This approach requires some effort to identify performance bottlenecks and design custom instructions, but does not require the user to modify the processor itself.

If coprocessors or custom instructions aren't enough to meet performance targets, a more aggressive approach is to license a customizable processor core and make modifications not only to the instruction set, but to the architecture itself. This can result in much higher performance gains, but the user must be willing to take on the high cost, lengthy development time, and risk associated with developing a custom chip. This approach is usually only appropriate for applications where the costs associated with modifying the architecture can be amortized over high product volumes. Tensilica is one example of a

licensable processor core vendor that supports modifications to its architecture and instruction set; the company offers a sophisticated design environment to aid in the customization process.

A DSP, or Not?

In recent years, it has become more common for system designers to opt not to use a DSP processor at all, instead choosing a general-purpose processor (GPP) that has been enhanced with DSP features to handle the DSP tasks in an application. Nearly all GPPs today include some support for signal processing, often in the form of SIMD instruction set extensions. DSP-enhanced general-purpose processors generally can't match the efficiency (in terms of size, power consumption, and price) of DSP processors on DSP tasks, but in applications that can tolerate lower efficiency in exchange for supporting a mainstream operating system, for example, this can be an attractive choice.

Yet another possibility is to choose a chip that includes both a GPP and a DSP. TI's OMAP1510 chip is a prominent example of this approach and includes an ARM9 microcontroller and a 'C55xx DSP. This solution enables the system designer to use a GPP for tasks like a GUI, network protocol stacks, and business applications, while using the DSP for the computationally demanding, signal-processing-intensive portions of the application. Of course, the dual-core approach introduces complications in programming, system design, and debugging.

DSP-enhanced general-purpose processors generally can't match the efficiency (in terms of size, power consumption, and price) of DSP processors on DSP tasks, but in applications that can tolerate lower efficiency in exchange for supporting a mainstream operating system, for example, this can be an attractive choice.

Choices, Choices

The last five years have seen enormous changes in DSP architectures and instruction sets. The range of solutions available to DSP system designers has become incredibly diverse, and identifying the best processor for a given application is becoming increasingly complicated. The good news is that, somewhere out there in the mind-boggling array of choices, there's probably a solution that will fit just right.

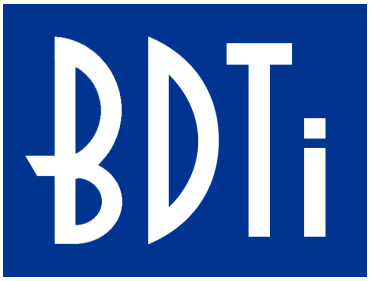
About Berkeley Design Technology, Inc.

Founded in 1991 and located in Berkeley, California, Berkeley Design Technology, Inc. (BDTI) helps companies develop, select, and use digital signal processing (DSP) technology to achieve key business objectives. Systems developers and processor designers look to BDTI for processor benchmarking, technology evaluation, and advice on DSP product development. IP providers and product manufacturers rely on BDTI for creative solutions to implementation challenges. BDTI provides:

- Software Development Services
  - Audio and video
  - Communications
  - General DSP component libraries
- Consulting and Analysis
  - Processor benchmarking.
  - In-depth analysis of microprocessors, tools, algorithms, and software.
  - Publication of technology analysis reports including *Buyer's Guide to DSP Processors*, *Inside* reports, and *Focus* reports.
  - Seminars on DSP applications, implementations, and technology.

BDTI Customers Include…

3Com
3DSP
Agere Systems
Altera
ARM
AMD
Analog Devices
Apple Computer
ARC
Bang & Olufsen
Cadence Design Systems
Canon
Cirrus Logic
Cisco
Conexant
Creative Technologies
CSF Thomson
E.M. Warburg, Pincus
Euphonix
Ericsson
Faraday
Fujitsu
General Dynamics
Glenayre
Hewlett-Packard
Hitachi Semiconductor
Hynix
IBM Microelectronics
IDT
Infineon Technologies
Intel
Intersil
Intrinsity
Lockheed Martin
Loral Rolm
LSI Logic
Mentor Graphics
Mercury Computer
Microchip
Microsoft
MIPS
Mitsubishi Semiconductors
Motorola
National Semiconductor
NEC Electronics
Nokia
Northern Telecom
Philips Semiconductors
RCA
RealNetworks
Samsung
Siemens
Sony
StarCore
STMicroelectronics
Sun Microsystems
SuperH
Synopsys
Tensilica
Texas Instruments
Thales Group
Thomson Consumer
Toshiba
Wind River Systems
Xilinx
Zoran



Berkeley Design Technology, Inc.
2107 Dwight Way, Second Floor
Berkeley, California 94704 USA
Phone: +1 (510) 665 1600
Fax: +1 (510) 665 1680
Email: info@BDTI.com
Web: www.BDTI.com