THE DSP32 DIGITAL SIGNAL PROCESSOR AND ITS APPLICATION DEVELOPMENT TOOLS

James R. Boddie, Renato N. Gadenz, Robert N. Kershaw, W. Patrick Hays, and James Tow

James R. Boddie *is head of the Signal Processing and Integrated Circuit Design Department at AT&T Bell Laboratories in Holmdel, New Jersey.* ***W. Patrick Hays*** *is a supervisor, and* ***Renato N. Gadenz*** *and* ***James Tow*** *are members of technical staff in that department.* ***Robert N. Kershaw*** *is a supervisor in the Digital IC Design Department at AT&T Bell Laboratories in Allentown, Pennsylvania. Mr. Boddie's department designs analog and digital signal processing integrated circuits and support systems. Mr. Boddie joined AT&T in 1977 and has a B.S.E.E. from Auburn University, an S.M. and E.E. from Massachusetts Institute of Technology (MIT), and a Ph.D. from Auburn University, all in electrical engineering. Mr. Gadenz, who designs and tests new VLSI circuits for digital signal processing, joined* (continued on page 104)

The WE® DSP32 digital signal processor is a high-speed, programmable, VLSI circuit with 32-bit floating-point arithmetic. The device can be used cost-effectively in a wide variety of complex digital signal processing applications, such as speech recognition, high-speed modems, low bit-rate voice coders, multichannel signaling systems, and signal processing workstations. In this paper, we review the architecture of the DSP32 and present its instruction set, including some examples. We also discuss the software and hardware support tools that are available for developing DSP32 applications.

An Expanding Family

The WE® DSP32 digital signal processor[1,2] is the latest member of a family of single chip, programmable digital signal processors developed at AT&T Bell Laboratories.

The first family member appeared in 1979 and was called simply DSP.[3] It was one of the first single-chip, programmable digital signal processors ever made.

As very-large-scale-integration (VLSI) technology advanced, the demand for the DSP justified its redesign. The new device—the WE DSP20—ran twice as fast, used less power, and cost less than the original DSP, yet was pin, architecture, and instruction set compatible.

DSP32 Features

Both the DSP and DSP20 are 20-bit, fixed-point processors and can execute instructions at the rate of 1.25 and 2.5 million instructions per second, respectively.

The DSP32, on the other hand, can execute 4 million instructions per second, operating on 32-bit, floating-point numbers. It also has a complete set of microprocessor instructions that operate on 16-bit, fixed-point numbers for ease of use in logic and control operations. In addition, the DSP32 offers both serial and parallel input/output (I/O) with direct-memory access (DMA) capability. (DMA allows data trans-

fers from an input buffer to memory or from memory to an output buffer, without program intervention.)

Like the previous DSPs, the DSP32 has an on-chip, random-access memory (RAM) for storing variable data, and an on-chip, mask-programmable read-only memory (ROM) for storing instructions and fixed data. Thus, the device can be customized for a wide variety of signal processing applications.

Although there are many different types of applications, most require real-time execution of repetitive multiplications and additions. Like its predecessors, the DSP32 is optimized for this type of computation. But it offers a new architecture that allows users to develop more complex applications requiring floating-point arithmetic.

The DSP32 can be cost-effective for a wide variety of digital signal processing applications, such as speech recognition, high-speed modems, low bit-rate voice coders, multichannel signaling systems, and signal processing workstations. Algorithms that are suited for floating point include matrix inversion; spectral analysis; high-quality, low bit-rate speech; graphics; and image processing.

- Our goal in designing the DSP32 has been to balance high performance with ease of use. By high performance, we mean fast execution of signal processing algorithms using a high-quality arithmetic. A major contributor to this goal is the use of a 32-bit, floating-point-data arithmetic unit.

An 8-bit exponent in the 32-bit, floating-point word yields a large dynamic range that makes overflow unlikely, while a 24-bit normalized mantissa gives high precision, independent of magnitude. Large dynamic range and high precision are often essential in advanced algorithms.

Floating point also contributes to ease of use, because it frees programmers from concern about intermediate scaling to avoid overflow or loss of precision. Further, algorithms that are initially developed on mainframe computers or array processors and use floating-point arithmetic can be easily adapted to run on the DSP32.

Ease of use applies not only to programming or interfacing to the DSP32, but to the support tools that are available for developing DSP32 applications. We will discuss the architecture, instruction set, and support tools later.

The DSP32 can do the following, all with 32-bit, floating-point precision and dynamic range:

- a 1024-point, complex, Fast Fourier Transform (FFT) in 19.2 ms (including bit reversal)
- a finite impulse response filter in 250 ns per tap
- a second-order section (four multiply) for a recursive, infinite-impulse-response (IIR) filter in 1 μs.

The device is available in two packages: a 40-pin dual in-line package, and a 100-pin pin array that can use external memory to expand the on-chip memory. The chip is manufactured in 1.5-μm effective channel length, n-type metal-oxide semiconductor (NMOS) technology, and has about 155,000 transistors in an 81-mm² area (12.70 mm by 6.35 mm).

The DSP32 operates with a 16.384-MHz clock and a single 5V power supply. For the 40-pin device, typical power dissipation is 1.8W, while worst-case power dissipation is 2.3W. For the 100-pin package, typical power dissipation is 2.0W and worst case is 2.6W. The device is now being manufactured in high volumes.

Recently, the DSP32 has been manufactured with a 10-percent photolithographic shrink. Samples will soon be available from two different processes; one leads to higher speed devices and the other to lower power devices.

Its Architecture

The DSP32's architecture consists of several specialized sections (Figure 1) that work in parallel to achieve a high throughput. A 32-bit bus interconnects these sections, and control is distributed among them.

The device has two execution units—the *data arithmetic unit* (DAU) and the *control arithmetic unit* (CAU)—and its on-chip memory includes 2048 bytes of ROM and 4096 bytes of RAM. The memory can be addressed as 8-, 16-, or 32-bit words and is organized to

access 32-bit data at the same speed as 8-bit data.

A flexible serial I/O (SIO) unit allows direct interface to a codec, a time-division multiplex line, or another DSP32, while an 8-bit parallel I/O (PIO) unit allows bidirectional communication with a microprocessor.

Data Arithmetic Unit. The DAU is the primary execution unit for signal processing algorithms. As Figure 1 shows, it contains a 32-bit, floating-point multiplier; a 40-bit, floating-point adder; and four 40-bit accumulator registers (a0 through a3) that provide temporary storage, thus reducing memory accesses.

The DAU has a multiply-add structure; that is, the multiplier output is directly connected to the adder input. It does 4 million instructions per second of the form, `A = B + C * D`. In this instruction, we have both a floating-point multiplication and a floating-point addition, which yields a throughput of 8 million floating-point operations per second.

The DAU multiplier inputs are 32-bit, floating-point numbers with a 24-bit mantissa and 8-bit exponent.

Inputs to the multiplier and adder can come from memory, I/O registers, or an accumulator (a0–a3). The adder inputs from memory or I/O are 8, 16, or 32 bits wide, while those from the multiplier or an accumulator are 40-bits wide (an 8-bit exponent and a 32-bit mantissa that includes eight guard bits).

These 40 bits of precision are maintained in the adder and the accumulators; the eight guard bits of the mantissa allow extra precision in intermediate accumulations. However, the 40-bit result in an accumulator is truncated to 32-bits when written to memory or I/O, or provided as an input to the multiplier.

The DAU converts floating-point data to and from 16-bit integer data. It also converts floating-point data to and from μ-law and A-law companded data formats.

Control Arithmetic Unit. The CAU—a 16-bit, fixed-point unit—has a dual function. It generates and post modifies addresses for accessing operands in memory, and it executes microprocessor instructions on 16-bit data for logic and control operations. (Post modify means increment the address *after* completing the operation.)

As Figure 1 shows, the CAU has 21 16-bit general purpose registers (r1 through r21); a 16-bit program counter (pc); and a full-function, arithmetic logic unit (ALU). All CAU registers are static and do not require refreshing.

For instructions that are primarily executed in the DAU, registers r1 through r14 are used as memory pointers and registers r15 though r19 hold address increments. A *pointer register* contains the address of an operand in memory that is to be either read or written, and an *increment register* contains a number that is added to the pointer to alter it. This addition is done in the ALU.

Register r20, also called PIN (pointer-in), is the pointer for serial DMA input, and register r21, also called POUT (pointer-out), is the pointer for serial DMA output. These registers can also be used as general-purpose registers, but their effect on DMA operations must be considered.

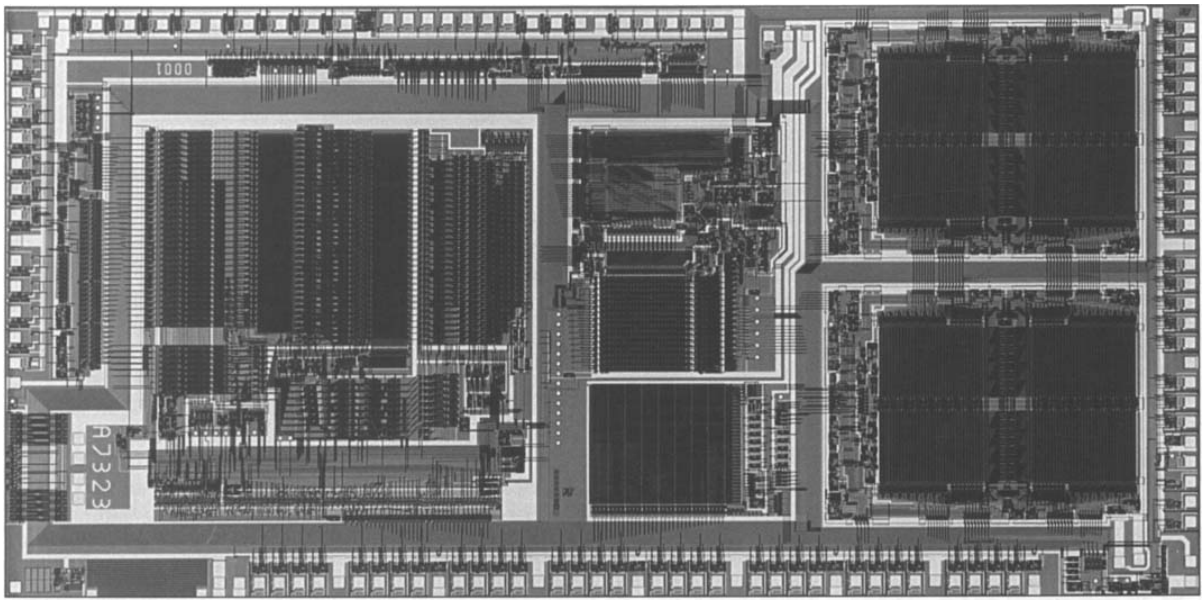
As stated above, the CAU also executes its own set of instructions. They implement two's complement, fixed-point arithmetic operations; logic functions; control operations like conditional branching; and data moves between memory, I/O, and the CAU registers. Executing logic and control operations in the CAU not only makes programming the device easier but also enhances the DSP32 performance.

Memory. The DSP32 provides on-chip memory (Figure 1) that includes a 512 by 32-bit ROM and a 1024 by 32-bit RAM that is divided into two equal 512 by 32-bit segments.

Data can be 8, 16, or 32 bits wide, and memory is uniformly byte addressable. This means that the four individual bytes and the two 16-bit words in each 32-bit word can be addressed independently.

Byte addressability is important because, besides using 32-bit instructions and floating-point data, the DSP32 uses 16-bit fixed-point integers and 8-bit companded (μ-law and A-law) data.

The RAM is dynamic and is refreshed either auto-



matically once every 32 instructions or under program control. The ROM is masked programmed.

For the 100-pin pin array package, an additional 56 kbytes of memory can be accessed externally. If standard byte-wide memory chips are used for expanding memory, no additional interfacing devices are needed. Also, if this external memory is fast enough (80-ns access time), there is no speed penalty when accessing it.

The DSP32 memory space (ROM, RAM, and external memory) is logically divided into two banks: *Lower bank 0*, which can be expanded with external memory, and *Upper bank 1*. Memory space can be configured in four different ways using two of the DSP32 pins.

To achieve maximum throughput, memory accesses must alternate between the two memory banks. As one memory bank is accessed, the other memory bank is being addressed. However, if the user chooses not to interleave, the DSP32 automatically inserts a wait state when two consecutive accesses occur to the same memory bank. This flexible memory accessing makes the DSP32 easier to program.

Instructions can be stored anywhere in the address space (ROM, RAM or external memory) and can be executed from any memory without a speed penalty, if interleaving between the two memory banks is used. Each instruction cycle consists of four machine states, and one read or write operation can occur during each state.

Serial I/O. The SIO is used for serial-to-parallel conversion of input data and parallel-to-serial conversion of output data. Serial input and output operations are independent and asynchronous with respect to each other and to program execution. The SIO control signals allow direct interface to a codec, a time-division multiplexed line, or another DSP32.

Input to the SIO (Figure 1) is loaded into the input shift register (`isr`), and then into the input buffer (`ibuf`). SIO outputs are loaded into the output buffer (`obuf`), and then put into the output shift register (`osr`). This double buffering permits a second serial transmission to begin before the first transmission has been processed.

Data widths can be 8, 16, or 32 bits. The I/O control register `ioc` in the SIO is used to select various I/O conditions, bit lengths, internal or external clocks, and internal or external synchronization signal, thus allowing flexibility when interfacing to external hardware.

For example, in the active mode, the DSP32 generates the I/O clocks and the synchronization signal for external hardware. In the passive mode, the DSP32 acts as a slave and the external hardware generates its clocks and synchronization signal.

SIO transfers could occur under program control or using direct memory access. To enable DMA, the user

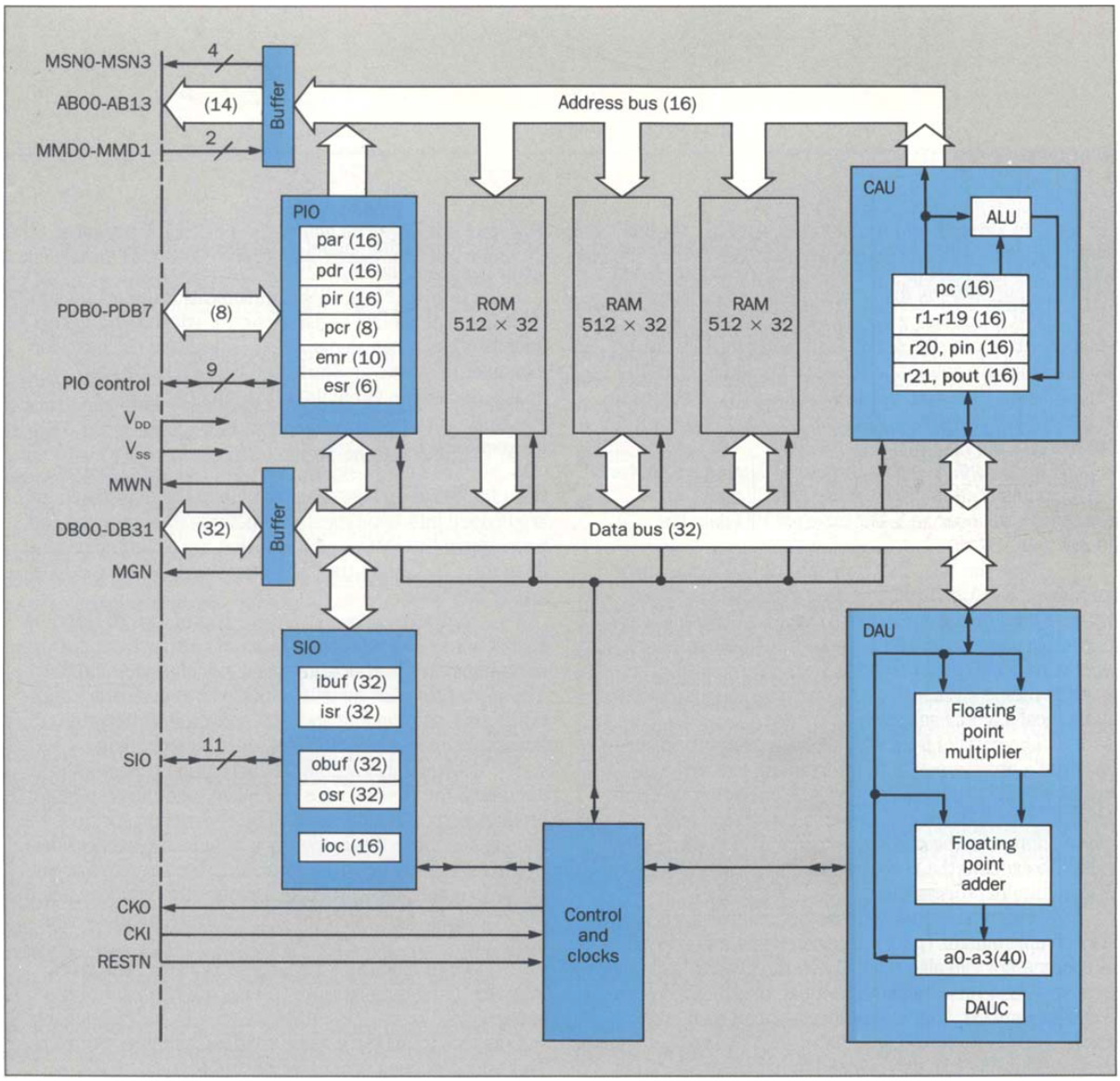


Figure 1. Block diagram of the DSP32 digital signal processor. The numbers in parentheses designate the size of a particular register or bus. The symbols at the left represent pins on the device package.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.