The TMS320 Family of Digital Signal Processors

APPLICATION REPORT: SPRA396

Kun-Shan Lin
Gene A. Frantz
Ray Simar, Jr.
Digital Signal Processor Product
Semiconductor Group
Texas Instruments

Digital Signal Processing Solutions

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain application using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

TRADEMARKS

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

CONTACT INFORMATION

US TMS320 HOTLINE	(281) 274-2320
US TMS320 FAX	(281) 274-2324
US TMS320 BBS	(281) 274-2323
US TMS320 email	dsph@ti.com

The TMS320 Family of Digital Signal Processors

Abstract

The introduction of the TMS320C30 Digital Signal Processor, a floating-point 33-MFLOP device, allows users to represent multi-length floating-point math in terms of single length floating-point math. This allows applications which need this type of power (digital filtering, image processing, FFTs, etc.) to be much more accurate. This chapter explains how to extend the available precision of floating-point arithmetic on the TMS320C30. It is organized as follows:

- A description of the TMS320C30 DSP floating-point number representation
- A discussion of doublelength arithmetic and some basic definitions
- A discussion of the algorithms used along with the TMS320C30 implementation
- Error analysis
- Information about generating C-callable functions from assembly language routines

Accompanying graphics illustrate

- Single precision floating point format of the TMS320C30
- Singlelength and doublelength addition
- Singlelength and doublelength procude
- Doublelength quotient and square root

The chapter closes with a summary, a list of references, and three appendices which provide source listings for the extended-precision arithmetic.

Product Support

World Wide Web

Our World Wide Web site at www.ti.com contains the most up to date product information, revisions, and additions. Users registering with TI&ME can build custom information pages and receive new product updates automatically via email.

Email

For technical issues or clarification on switching products, please send a detailed email to (dsph@ti.com). Questions receive prompt attention and are usually answered within one business day.

The TMS320 Family of Digital Signal Processors

KUN-SHAN LIN, MEMBER, IEEE, GENE A. FRANTZ, SENIOR MEMBER, IEEE, AND RAY SIMAR, JR.

This paper begins with a discussion of the characteristics of digital signal processing, which are the driving force behind the design of digital signal processors. The remainder of the paper describes the three generations of the TMS320 family of digital signal processors available from Texas Instruments. The evolution in architectural design of these processors and key features of each generation of processors are discussed. More detailed information is provided for the TMS320C25 and TMS320C30, the newest members in the family. The benefits and cost-performance tradeoffs of these processors become obvious when applied to digital signal processing applications, such as telecommunications, data communications, graphics/image processing, etc.

Digital Signal Processing Characteristics

Digital signal processing (DSP) encompasses a broad spectrum of applications. Some application examples include digital filtering, speech vocoding, image processing, fast Fourier transforms, and digital audio [1]–[10]. These applications and those considered digital signal processing have several characteristics in common:

- mathematically intensive algorithms,
- real-time operation,
- sampled data implementation,
- system flexibility.

To illustrate these characteristics in this section, we will use the digital filter as an example. Specifically, we will use the Finite Impulse Response (FIR) filter which in the time domain takes the general form of

$$y(n) = \sum_{i=1}^{N} a(i) * x(n - i) \qquad (1)$$

where $y(n)$ is the output sample at time n, $a(i)$ is the ith coefficient or weighting factor, and $x(n - i)$ is the $(n - i)$th input sample.

With this example in mind, we can discuss the various characteristics of digital signal processing: mathematically intensive algorithms, real-time processing, sampled data implementation, and system flexibility. First, let us look at the concept of mathematically intensive algorithms.

Manuscript received October 6, 1986; revised March 27, 1987.

The authors are with the Semiconductor Group, Texas Instruments Inc., Houston, TX 77521-1445, USA.

IEEE Log Number 8716214.

Mathematically Intensive Algorithms

From (1), we can see that to generate every $y(n)$, we have to compute N multiplications and additions or sums of products. This computation makes it mathematically intensive, especially when N is large.

At this point it is worthwhile to give the FIR filter some physical significance. An FIR filter is a common technique used to eliminate the erratic nature of stock market prices. When the day-to-day closing prices are plotted, it is sometimes difficult to obtain the desired information, such as the trend of the stock, because of the large variations. A simple way of smoothing the data is to calculate the average closing values of the previous five days. For the new average value each day, the oldest value is dropped and the newest value added. Each daily average value (average (n)) would be the sum of the weighted value of the latest five days, where the weighting factors ($a(i)$'s) are 1/5. In equation form, the average is determined by

$$\text{average } (n) = \frac{1}{5} * d(n - 1) + \frac{1}{5} * d(n - 2) + \frac{1}{5} * d(n - 3) + \frac{1}{5} * d(n - 4) + \frac{1}{5} * d(n - 5) \qquad (2)$$

where $d(n - i)$ is the daily stock closing price for the $(n - i)$th day. Equation (2) assumes the same form as (1). This is also the general form of the convolution of two sequences of numbers, $a(i)$ and $x(i)$ [5], [6]. Both FIR filtering and convolution are fundamental to digital signal processing.

Real-Time Processing

In addition to being mathematically intensive, DSP algorithms must be performed in real time. Real time can be defined as a process that is accomplished by the DSP without creating a delay noticeable to the user. In the stock market example, as long as the new average value can be computed prior to the next day when it is needed, it is considered to be completed in real time. In digital signal processing applications, processes happen faster than on a daily basis. In the FIR filter example in (1), the sum of products must

be computed usually within hundreds of microseconds before the next sample comes into the system. A second example is in a speech recognition system where a noticeable delay between a word being spoken and being recognized would be unacceptable and not considered real-time. Another example is in image processing, where it is considered real-time if the processor finishes the processing within the frame update period. If the pixel information cannot be updated within the frame update period, problems such as flicker, smearing, or missing information will occur.

Sampled Data Implementation

The application must be capable of being handled as a sampled data system in order to be processed by digital processors, such as digital signal processors. The stock market is an example of a sampled data system. That is, a specific value (closing value) is assigned to each sample period or day. Other periods may be chosen such as hourly prices or weekly prices. In an FIR filter as shown in (1), the output $y(n)$ is calculated to be the weighted sum of the previous N inputs. In other words, the input signal is sampled at periodic intervals (1 over the sample rate), multiplied by weighting factor $a(i)$, and then added together to give the output result of $y(n)$. Examples of sample rates for some typical sampled data applications [2], [4] are shown in Table 1.

Table 1 Sample Rates versus Applications

Application	Nominal Sample Rate
Control	1 kHz
Telecommunications	8 kHz
Speech processing	8–10 kHz
Audio processing	40–48 kHz
Video frame rate	30 Hz
Video pixel rate	14 MHz

In a typical DSP application, the processor must be able to effectively handle sampled data in large quantity and also perform arithmetic computations in real time.

System Flexibility

The design of the digital signal processing system must be flexible enough to allow improvements in the state of the art. We may find out after several weeks of using the average stock price as a means of measuring a particular stock's value that a different method of obtaining the daily information is more suited to our needs, e.g., using different daily weightings, a different number of periods over which to average, or a different procedure for calculating the result. Enough flexibility in the system must be available to allow for these variations. In many of the DSP applications, techniques are still in the developmental phase, and therefore the algorithms tend to change over time. As an example, speech recognition is presently an inexact technique requiring continual algorithmic modification. From this example we can see the need for system flexibility so that the DSP algorithm can be updated. A programmable DSP system can provide this flexibility to the user.

Historical DSP Solutions

Over the past several decades, digital signal processing machines have taken on several evolutions in order to incorporate these characteristics. Large mainframe computers were initially used to process signals in the digital domain. Typically, because of state-of-the-art limitations, this was done in nonreal time. As the state of the art advanced, array processors were added to the processing task. Because of their flexibility and speed, array processors have become the accepted solution for the research laboratory, and have been extended to end-applications in many instances. However, integrated circuit technology has matured, thus allowing for the design of faster microprocessors and microcomputers. As a result, many digital signal processing applications have migrated from the array processor to microprocessor subsystems (i.e., bit-slice machines) to single-chip integrated circuit solutions. This migration has brought the cost of the DSP solution down to a point that allows pervasive use of the technology. The increased performance of these highly integrated circuits has also expanded DSP applications from traditional telecommunications to graphics/image processing, then to consumer audio processing.

A recent development in DSP technology is the single-chip digital signal processor, such as the TMS320 family of processors. These processors give the designer a DSP solution with its performance attainable only by the array processors a few years ago. Fig. 1 shows the TMS320 family in graphical form with the y-axis indicating the hypothetical performance and the x-axis being the evolution of the semiconductor processing technology. The first member of the family, the TMS32010, was disclosed to the market in 1982 [11], [12]. It gave the system designer the first microcomputer capable of performing five million DSP operations per second (5 MIPS), including the add and multiply functions [13] required in (1). Today there are a dozen spinoffs from the TMS32010 in the first generation of the TMS320 family. Some of these devices are the TMS320C10, TMS320C15, and TMS320C17 [14]. The second generation of devices include the TMS32020 [15] and TMS320C25 [16]. The TMS320C25 can perform 10 MIPS [16]. In addition, expanded memory space, combined single-cycle multiply/accumulate operation, multiprocessing capabilities, and expanded I/O functions have given the TMS320C25 a 2 to 4 times performance improvement over its predecessors. The third generation of the TMS320 family of processors, the TMS320C30 [26], [27], has a computational rate of 33 million DSP floating-point operations per second (33 MFLOPS). Its performance (speed, throughput, and precision) has far exceeded the digital signal processors available today and has reached the level of a supercomputer.

It we look closely at the TMS320 family as shown in Fig. 1, we can see that devices in the same generation, such as the TMS320C10, TMS320C15, and TMS320C17, are assembly object-code compatible. Devices across generations, such as the TMS320C10 and TMS320C25, are assembly source-code compatible. Software investment on DSP algorithms therefore can be maintained during the system upgrade. Another point is that since the introduction of the TMS32010, semiconductor processing technology has emerged from 3-μm NMOS to 2-μm CMOS to 1-μm CMOS.

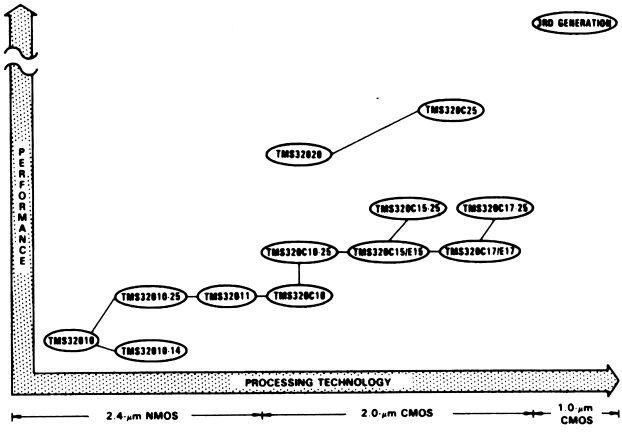


Fig. 1. The TMS320 family of digital signal processors.

The TMS320 generations of processors have also taken the same evolution in processing technology. Low power consumption, high performance, and high-density circuit integration are some of the direct benefits of this semiconductor processing evolution.

From Fig. 1, it can be observed that various DSP building blocks, such as the CPU, RAM, ROM, I/O configurations, and processor speeds, have been designed as individual modules and can be rearranged or combined with other standard cells to meet the needs of specific applications. Each of the three generations (and future generations) will evolve in the same manner. As applications become more sophisticated, semicustom solutions based on the core CPU will become the solution of choice. An example of this approach is the TMS320C17/E17, which consists of the TMS320C10 core CPU, expanded 4K-word program ROM (TMS320C17) or EPROM (TMS320E17), enlarged data RAM of 256 words, dual serial ports, companding hardware, and a coprocessor interface. Furthermore, as integrated circuit layout rules move into smaller geometry (now at 2 μm, rapidly going to 1 μm), not only will the TMS320 devices become smaller in size, but also multiple CPUs will be incorporated on the same device along with application-specific I/O to achieve low-cost integrated system solutions.

Basic TMS320 Architecture

As noted previously, the underlying assumption regarding a digital signal processor is fast arithmetic operations and high throughput to handle mathematically intensive algorithms in real time. In the TMS320 family [11]–[17], [26], [27], this is accomplished by using the following basic concepts:

- Harvard architecture,
- extensive pipelining,
- dedicated hardware multiplier,
- special DSP instructions,
- fast instruction cycle.

These concepts were designed into the TMS320 digital signal processors to handle the vast amount of data characteristic of DSP operations, and to allow most DSP operations to be executed in a single-cycle instruction. Furthermore, the TMS320 processors are programmable devices, providing the flexibility and ease of use of general-purpose microprocessors. The following paragraphs discuss how each of the above concepts is used in the TMS320 family of devices to make them useful in digital signal processing applications.

Harvard Architecture

The TMS320 utilizes a modified Harvard architecture for speed and flexibility. In a strict Harvard architecture [18], [19], the program and data memories lie in two separate spaces, permitting a full overlap of instruction fetch and execution. The TMS320 family's modification of the Harvard architecture further allows transfer between program and data spaces, thereby increasing the flexibility of the device. This architectural modification eliminates the need for a separate coefficient ROM and also maximizes the processing power by maintaining two separate bus structures (program and data) for full-speed execution.

Extensive Pipelining

In conjunction with the Harvard architecture, pipelining is used extensively to reduce the instruction cycle time to its absolute minimum, and to increase the throughput of the processor. The pipeline can be anywhere from two to four levels deep, depending on which processor in the family is used. The TMS320 family architecture uses a two-level pipeline for its first generation, a three-level pipeline for its second generation, and a four-level pipeline for its third generation of processors. This means that the device is processing from two to four instructions in parallel, and each instruction is at a different stage in its execution. Fig. 2 shows an example of a three-level pipeline operation.

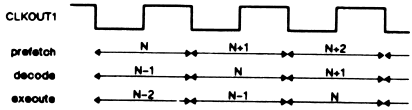


Fig. 2. Three-level pipeline operation.

In pipeline operation, the prefetch, decode, and execute operations can be handled independently, thus allowing the execution of instructions to overlap. During any instruction cycle, three different instructions are active, each at a different stage of completion. For example, as the *N*th instruction is being prefetched, the previous $(N - 1)$th instruction is being decoded, and the previous $(N - 2)$th instruction is being executed. In general, the pipeline is transparent to the user.

Dedicated Hardware Multiplier

As we saw in the general form of an FIR filter, multiplication is an important part of digital signal processing. For each filter tap (denoted by *i*), a multiplication and an addition must take place. The faster a multiplication can be performed, the higher the performance of the digital signal processor. In general-purpose microprocessors, the multiplication instruction is constructed by a series of additions, therefore taking many instruction cycles. In comparison, the characteristic of every DSP device is a dedicated multiplier. In the TMS320 family, multiplication is a single-cycle instruction as a result of the dedicated hardware multiplier. If we look at the arithmetic for each tap of the FIR filter to be performed by the TMS32010, we see that each tap of the filter requires a multiplication (MPY) instruction.

```
LT     ;LOAD MULTIPLICAND INTO T REGISTER
DMOV   ;MOVE DATA IN MEMORY TO DO DELAY
MPY    ;MULTIPLY
APAC   ;ADD MULTIPLICATION RESULT TO ACC
```

The other three instructions are used to load the multiplier circuit with the multiplicand (LT), move the data through the filter tap (DMOV), and add the result of the multiplication (stored in the product register) to the accumulator (APAC). Specifically, the multiply instruction (MPY) loads the multiplier into the dedicated multiplier and performs the multiplication, placing the result in a product register. Therefore, if a 256-tap FIR filter is used, these four instructions are repeated 256 times. At each sample period, 256 multiplications must be performed. In a typical general-purpose microprocessor, this requires each tap to be 30 to 40 instruction cycles long, whereas in the TMS320C10, it is only four instruction cycles. We will see in the next section how special DSP instructions reduce the time required for each FIR tap even further.

Special DSP Instructions

Another characteristic of DSP devices is the use of special instructions. We were introduced to one of them in the previous example, the DMOV (data move) instruction. In digital signal processing, the delay operator (z^{-1}) is very important. Recalling the stock market example, during each new sample period (i.e., each new day), the oldest piece of data (the closing price five days ago) was dropped and a new one (today's closing price) was added. Or, each piece of the old data is delayed or moved one sample period to make room for the incoming most current sample. This delay is the function of the DMOV instruction. Another special instruction in the TMS32010 is the LTD instruction. It executes the LT, DMOV, and APAC instructions in a single cycle. The LTD and MPY instruction then reduce the number of instruction cycles per FIR filter tap from four to two. In the second-generation TMS320, such as the TMS320C25, two more special instructions have been included (the RPT and MACD instructions) to reduce the number of cycles per tap to one, as shown in the following:

```
RPTK  255  ;REPEAT THE NEXT INSTRUCTION 256 TIMES
           (N + 1)
MACD       ;LT, DMOV, MPY, AND APAC
```

Fast Instruction Cycle

The real-time processing capability is further enhanced by the raw speed of the processor in executing instructions. The characteristics which we have discussed, combined with optimization of the integrated circuit design for speed, give the DSP devices instruction cycle times less than 200 ns. The specific instruction cycle times for the TMS320 family are given in Table 2. These fast cycle times have made the TMS320 family of processors highly suited for many real-time DSP applications. Table 1 showed the sample rates for some typical DSP applications. This table can be combined with the cycle times indicated in Table 2 to show how many instruction cycles per sample can be achieved by the various generations of the TMS320 for real-time applications (see Fig. 3).

Table 2 TMS320 Cycle Times

Device	Cycle Time (ns)
TMS320C10*	160–200
TMS32020	160–200
TMS320C25	100–125
TMS320C30	60–75

*The same cycle time applies to all of the first-generation processors.

As we can see from Fig. 3, many instruction cycles are available to process the signal or to generate commands for real-time control applications. Therefore, for simple control applications, the general-purpose microprocessors or controllers would be adequate. However, for more mathematically intensive control applications, such as robotics and adaptive control, digital signal processors are much better suited [24]. The number of available instruction cycles is reduced as we increase the sample rate from 8 kHz for typical telecommunication applications to 40–48 kHz for audio processing. Since most of these real-time applications require only a few hundreds of instructions per sample (such as ADPCM [4], and echo cancelation [4]), this is within the reach of the TMS320. For higher sample rate applications, such as video/image processing, digital signal processors available today are not capable of handling the processing of the real-time video data. Therefore, for these

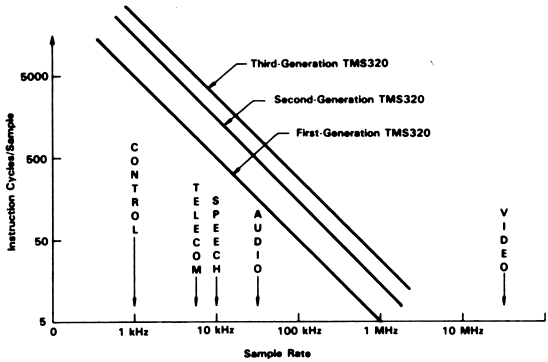


Fig. 3. Number of instruction cycles/sample versus sample rate for the TMS320 family.

types of applications, multiple digital signal processors and frame buffers are usually required. From Fig. 3, it can also be seen that for slower speed applications, such as control, the first-generation TMS320 provides better cost-performance tradeoffs than the other processors. For high sample rate applications, such as video/image processing, the second and third generations of the TMS320 with their multiprocessing capabilities and high throughput are better suited.

Now that we have discussed the basic characteristics of digital signal processors, we can concentrate on specific details of each of the three generations of the TMS320 family devices.

The First Generation of the TMS320 Family

The first generation of the TMS320 family includes the TMS32010 [13], and TMS32011 [17], which are processed in 2.4-μm NMOS technology, and the TMS320C10 [13], TMS320C15/E15 [14], and TMS320C17/E17 [14], processed in 1.8-μm CMOS technology. Some of the key features of these devices are [14] as follows:

- Instruction cycle timing:
  - -160 ns
  - -200 ns
  - -280 ns.
- On-chip data RAM:
  - -144 words
  - -256 words (TMS320C15/E15, TMS320C17/E17).
- On-chip program ROM:
  - -1.5K words
  - -4K words (TMS320C15, TMS320C17).
- 4K words of on-chip program EPROM (TMS320E15, TMS320E17).
- External memory expansion up to 4K words at full speed.
- 16 × 16-bit parallel multiplier with 32-bit result.
- Barrel shifter for shifting data memory words into the ALU.
- Parallel shifter.
- 4 × 12-bit stack that allows context switching.
- Two auxiliary registers for indirect addressing.
- Dual-channel serial port (TMS32011, TMS320C17, TMS320E17).
- On-chip companding hardware (TMS32011, TMS320C17, TMS320E17).
- Coprocessor interface (TMS320C17, TMS320E17).
- Device packaging
  - -40-pin DIP
  - -44-pin PLCC.

TMS320C10

The first generation of the TMS320 processors is based on the architecture of the TMS32010 and its CMOS replica, the TMS320C10. The TMS32010 was introduced in 1982 and was the first microcomputer capable of performing 5 MIPS. Since the TMS32010 has been covered extensively in the literature [4], [11]–[14], we will only provide a cursory review here. A functional block diagram of the TMS320C10 is shown in Fig. 4.

As shown in Fig. 4, the TMS320C10 utilizes the modified Harvard architecture in which program memory and data memory lie in two separate spaces. Program memory can reside both on-chip (1.5K words) or off-chip (4K words). Data memory is the 144 × 16-bit on-chip data RAM. There are four basic arithmetic elements: the ALU, the accumulator, the multiplier, and the shifters. All arithmetic operations are performed using two's-complement arithmetic.

ALU: The ALU is a general-purpose arithmetic logic unit that operates with a 32-bit data word. The unit can add, subtract, and perform logical operations.

Accumulator: The accumulator stores the output from the ALU and is also often an input to the ALU. It operates with a 32-bit word length. The accumulator is divided into a high-order word (bits 31 through 16) and a low-order word (bits 15 through 0). Instructions are provided for storing the high- and low-order accumulator words in data memory (SACH for store accumulator high and SACL for store accumulator low).

Multiplier: The 16 × 16-bit parallel multiplier consists of three units: the T register, the P register, and the multipler array. The T register is a 16-bit register that stores the multiplicand, while the P register is a 32-bit register that stores the product. In order to use the multiplier, the multiplicand

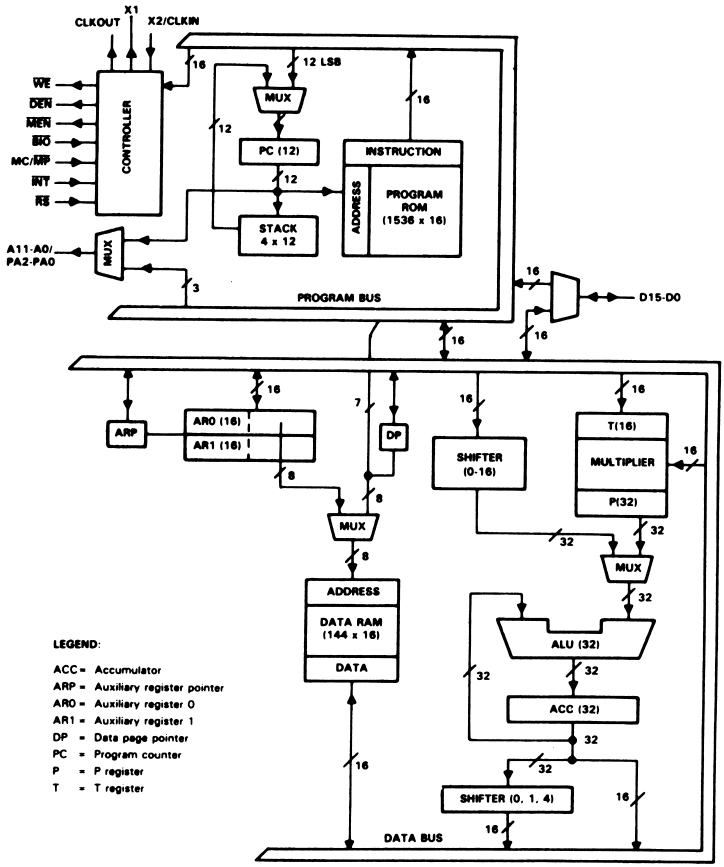


Fig. 4. TMS320C10 functional block diagram.

must first be loaded into the T register from the data RAM by using one of the following instructions: LT, LTA, or LTD. Then the MPY (multiply) or the MPYK (multiply immediate) instruction is executed. The multiply and accumulate operations can be accomplished in two instruction cycles with the LTA/LTD and MPY/MPYK instructions.

Shifters: Two shifters are available for manipulating data: a barrel shifter and a parallel shifter. The barrel shifter performs a left-shift of 0 to 16 bits on all data memory words that are to be loaded into, subtracted from, or added to the accumulator. The parallel shifter, activated by the SACH instruction, can execute a shift of 0, 1, or 4 bits to take care of the sign bits in two's-complement arithmetic calculations.

Based on the architecture of the TMS32010/C10, several spinoffs have been generated offering different processor speeds, expanded memory, and various I/O integration. Currently, the newest members in this generation are the TMS320C15/E15 and the TMS320C17/E17 [14].

TMS320C15/E15

The TMS320C15 and TMS320E15 are fully object-code and pin-for-pin compatible with the TMS32010 and offer expanded on-chip RAM of 256 words and on-chip program ROM (TMS320C15) or EPROM (TMS320E15) of 4K words. The TMS320C15 is available in either a 200-ns version or a 160-ns version (TMS320C15-25).

TMS320C17/E17

The TMS320C17/E17 is a dedicated microcomputer with 4K words of on-chip program ROM (TMS320C17) or EPROM (TMS320E17), a dual-channel serial port for full-duplex serial communication, on-chip companding hardware (u-law/A-law), a serial port timer for stand-alone serial communication, and a coprocessor interface for zero glue interface between the processor and any 4/8/16-bit microprocessor. The TMS320C17/E17 is also object-code compatible with the TMS32010 and can use the same development tools. The

Table 3 TMS320 First-Generation Processors

TMS320 Devices	Instruction Cycle Time (ns)	Process	On-Chip Prog ROM (words)	On-Chip Prog EPROM (words)	On-Chip Data RAM (words)	Off-Chip Prog (words)	Ref
TMS32010	200	NMOS	1.5K		144	4K	[13]
TMS32010-25	160	NMOS	1.5K		144	4K	[13]
TMS32010-14	280	NMOS	1.5K		144	4K	[13]
TMS32011	200	NMOS	1.5K		144		[17]
TMS320C10	200	CMOS	1.5K		144	4K	[13]
TMS320C10-25	160	CMOS	1.5K		144	4K	[13]
TMS320C15	200	CMOS	4.0K		256	4K	[13]
TMS320C15-25	160	CMOS	4.0K		256	4K	[14]
TMS320E15	200	CMOS		4.0K	256	4K	[14]
TMS320C17	200	CMOS	4.0K		256		[14]
TMS320C17-25	160	CMOS	4.0K		256		[14]
TMS320E17	200	CMOS		4.0K	256		[14]

device is based on the TMS320C10 core CPU with added peripheral memory and I/O modules added on-chip. The TMS320C17/E17 can be regarded as a semicustom DSP solution suited for high-volume telecommunication and consumer applications.

Table 3 provides a feature comparison of all members of the first-generation TMS320 processors. References to more detailed information on these processors are also provided.

The Second Generation of the TMS320 Family

The second-generation TMS320 digital signal processors includes two members, the TMS32020 [15] and the TMS320C25 [16]. The architecture of these devices has been evolved from the TMS32010, the first member of the TMS320 family. Key features of the second-generation TMS320 are as follows:

- Instruction cycle timing:
  - -100 ns (TMS320C25)
  - -200 ns (TMS32020).
- 4K words of on-chip masked ROM (TMS320C25).
- 544 words of on-chip data RAM.
- 128K words of total program data memory space.
- Eight auxiliary registers with a dedicated arithmetic unit.
- Eight-level hardware stack.
- Fully static double-buffered serial port.
- Wait states for communication to slower off-chip memories.
- Serial port for multiprocessing or interfacing to codecs.
- Concurrent DMA using an extended hold operation (TMS320C25).
- Bit-reversed addressing modes for fast Fourier transforms (TMS320C25).
- Extended-precision arithmetic and adaptive filtering support (TMS320C25).
- Full-speed operation of MAC/MACD instructions from external memory (TMS320C25).
- Accumulator carry bit and related instructions (TMS320C25).
- 1.8-μm CMOS technology (TMS320C25):
  - -68-pin grid array (PGA) package.
  - -68-pin lead chip carrier (PLCC) package.
- 2.4-μm NMOS technology (TMS32020):
  - -68-pin PGA package.

TMS320C25 Architecture

The TMS320C25 is the latest member in the second generation of TMS320 digital signal processors. It is a pin-compatible CMOS version of the TMS32020 microprocessor, but with an instruction cycle time twice as fast and the inclusion of additional hardware and software features. The instruction set is a superset of both the TMS32010 and TMS32020, maintaining source-code compatibility. In addition, it is completely object-code compatible with the TMS32020 so that TMS32020 programs run unmodified on the TMS320C25.

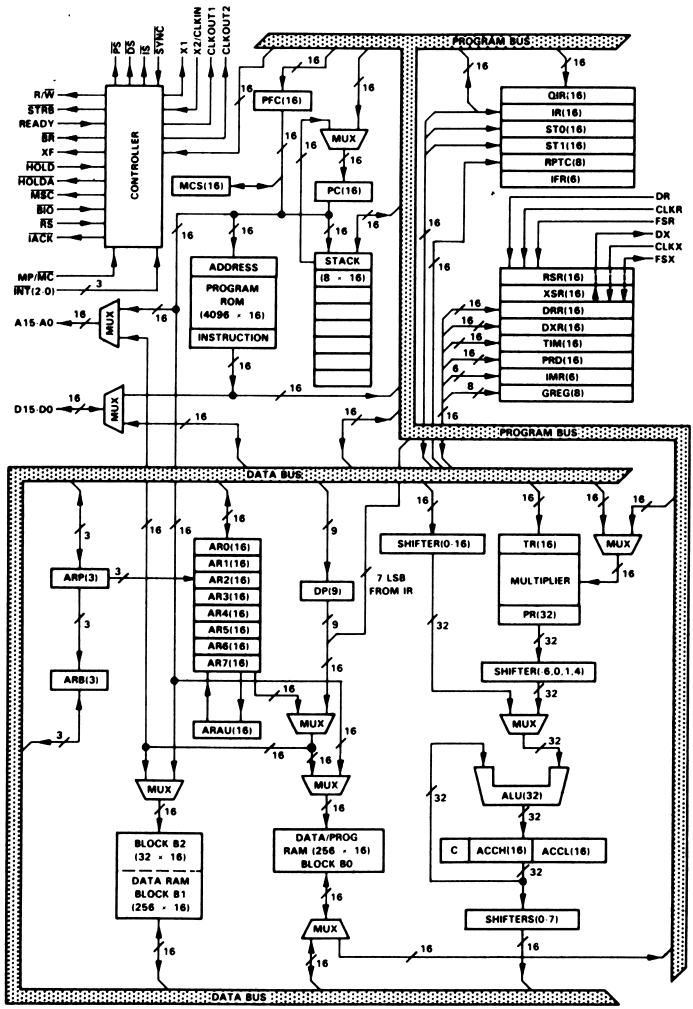
The 100-ns instruction cycle time provides a significant throughput advantage for many existing applications. Since most instructions are capable of executing in a single cycle, the processor is capable of executing ten million instructions per second (10 MIPS). Increased throughput on the TMS320C25 for many DSP applications is attained by means of single-cycle multiply/accumulate instructions with a data move option (MAC/MACD), eight auxiliary registers with a dedicated arithmetic unit, instruction set support for adaptive filtering and extended-precision arithmetic, bit-reversal addressing, and faster I/O necessary for data-intensive signal processing.

Instructions are included to provide data transfers between the two memory spaces. Externally, the program and data memory spaces are multiplexed over the same bus so as to maximize the address range for both spaces while minimizing the pin count of the device. Internally, the TMS320C25 architecture maximizes processing power by maintaining two separate bus structures, program and data, for full-speed execution.

Program execution in the device takes the form of a three-level instruction fetch-decode-execute pipeline (see Fig. 2). The pipeline is essentially invisible to the user, except in some cases where it must be broken (such as for branch instructions). In this case, the instruction timing takes into account the fact that the pipeline must be emptied and refilled. Two large on-chip data RAM blocks (a total of 544 words), one of which is configurable either as program or data memory, provide increased flexibility in system design. An off-chip 64K-word directly addressable data memory address space is included to facilitate implementations of DSP algorithms. The large on-chip 4K-word masked ROM can be used for cost-reduced systems, thus providing for a true single-chip DSP solution. The remainder of the 64K-word program memory space is located externally. Large

programs can execute at full speed from this memory space. Programs may also be downloaded from slow external memory to on-chip RAM for full-speed operation. The VLSI implementation of the TMS320C25 incorporates all of these features as well as many others such as a hardware timer, serial port, and block data transfer capabilities.

A functional block diagram of the TMS320C25, shown in Fig. 5, outlines the principal blocks and data paths within



LEGEND:

ACCH	Accumulator high	IFR	Interrupt flag register	PC	Program counter
ACCL	Accumulator low	IMR	Interrupt mask register	PFC	Prefetch counter
ALU	Arithmetic logic unit	IR	Instruction register	RPTC	Repeat instruction counter
ARAU	Auxiliary register arithmetic unit	MCS	Microcall stack	GREG	Global memory allocation register
ARB	Auxiliary register pointer buffer	OIR	Queue instruction register	RSR	Serial port receive shift register
ARP	Auxiliary register pointer	PR	Product register	XSR	Serial port transmit shift register
DP	Data memory page pointer	PRD	Period register for timer	AR0-AR7	Auxiliary registers
DRR	Serial port data receive register	TIM	Timer	ST0,ST1	Status registers
DXR	Serial port data transmit register	TR	Temporary register		

Fig. 5. TMS320C25 functional block diagram.

the processor. The diagram also shows all of the TMS320C25 interface pins.

In the following architectural discussions on the memory, central arithmetic logic unit, hardware multiplier, control operations, serial port, and I/O interface, please refer to the block diagram shown in Fig. 5.

Memory Allocation: The TMS320C25 provides a total of 4K 16-bit words of on-chip program ROM and 544 16-bit words of on-chip data RAM. The RAM is divided into three separate Blocks (B0, B1, and B2). Of the 544 words, 256 words (block B0) are configurable as either data or program memory by CNFD (configure data memory) or CNFP (configure program memory) instructions provided for that purpose; 288 words (blocks B1 and B2) are always data memory. A data memory size of 544 words allows the TMS320C25 to handle a data array of 512 words while still leaving 32 locations for intermediate storage. The TMS320C25 provides 64K words of off-chip directly addressable data memory space as well as a 64K-word off-chip program memory space.

A register file containing eight Auxiliary Registers (AR0–AR7), which are used for indirect addressing of data memory and for temporary storage, increase the flexibility and efficiency of the device. These registers may be either directly addressed by an instruction or indirectly addressed by a 3-bit Auxiliary Register Pointer (ARP). The auxiliary registers and the ARP may be loaded from either data memory or by an immediate operand defined in the instruction. The contents of these registers may also be stored into data memory. The auxiliary register file is connected to the Auxiliary Register Arithmetic Unit (ARAU). Using the ARAU accessing tables of information does not require the CALU for address manipulation, thus freeing it for other operations.

Central Arithmetic Logic Unit (CALU): The CALU contains a 16-bit scaling shifter, a 16 × 16-bit parallel multiplier, a 32-bit Arithmetic Logic Unit (ALU), and a 32-bit accumulator. The scaling shifter has a 16-bit input connected to the data bus and a 32-bit output connected to the ALU. This shifter produces a left-shift of 0 to 16 bits on the input data, as programmed in the instruction. Additional shifters at the outputs of both the accumulator and the multiplier are suitable for numerical scaling, bit extraction, extended-precision arithmetic, and overflow prevention.

The following steps occur in the implementation of a typical ALU instruction:

1) Data are fetched from the RAM on the data bus.
2) Data are passed through the scaling shifter and the ALU where the arithmetic is performed.
3) The result is moved into the accumulator.

The 32-bit accumulator is split into two 16-bit segments for storage in data memory: ACCH (accumulator high) and ACCL (accumulator low). The accumulator has a carry bit to facilitate multiple-precision arithmetic for both addition and subtract instructions.

Hardware Multiplier: The TMS320C25 utilizes a 16 × 16-bit hardware multiplier, which is capable of computing a 32-bit product during every machine cycle. Two registers are associated with the multiplier:

- a 16-bit Temporary Register (TR) that holds one of the operands for the multiplier, and
- a 32-bit Product Register (PR) that holds the product.

The output of the product register can be left-shifted 1 or 4 bits. This is useful for implementing fractional arithmetic or justifying fractional products. The output of the PR can also be right-shifted 6 bits to enable the execution of up to 128 consecutive multiple/accumulates without overflow. An unsigned multiply (MPYU) instruction facilitates extended-precision multiplication.

I/O Interface: The TMS320C25 I/O space consists of 16 input and 16 output ports. These ports provide the full 16-bit parallel I/O interface via the data bus on the device. A single input (IN) or output (OUT) operation typically takes two cycles; however, when used with the repeat counter, the operation becomes single-cycle. I/O devices are mapped into the I/O address space using the processor's external address and data buses in the same manner as memory-mapped devices. Interfacing to memory and I/O devices of varying speeds is accomplished by using the READY line.

A Direct Memory Access (DMA) to external program/data memory is also supported. Another processor can take complete control of the TMS320C25's external memory by asserting HOLD low, causing the TMS320C25 to place its address, data, and control lines in the high-impedance state. Signaling between the external processor and the TMS320C25 can be performed using interrupts. Two modes of DMA are available on the device. In the first, execution is suspended during assertion of HOLD. In the second "concurrent DMA" mode, the TMS320C25 continues to execute its program while operating from internal RAM or ROM, thus greatly increasing throughput in data-intensive applications.

TMS320C25 Software

The majority of the TMS320C25 instructions (97 out of 133) are executed in a single instruction cycle. Of the 36 instructions that require additional cycles of execution, 21 involve branches, calls, and returns that result in a reload of the program counter and a break in the execution pipeline. Another seven of the instructions are two-word, long-immediate instructions. The remaining eight instructions support I/O, transfers of data between memory spaces, or provide for additional parallel operation in the processor. Furthermore, these eight instructions (IN, OUT, BLKD, BLKP, TBLR, TBLW, MAC, and MACD) become single-cycle when used in conjunction with the repeat counter. The functional performance of the instructions exploits the parallelism of the processor, allowing complex and/or numerically intensive computations to be implemented in relatively few instructions.

Addressing Modes: Since most of the instructions are coded in a single 16-bit word, most instructions can be executed in a single cycle. Three memory addressing modes are available with the instruction set: direct, indirect, and immediate addressing. Both direct and indirect addressing are used to access data memory. Immediate addressing uses the contents of the memory addressed by the program counter.

When using direct addressing, 7 bits of the instruction word are concatenated with the 9 bits of the data memory page pointer (DP) to form the 16-bit data memory address. With a 128-word page length, the DP register points to one of 512 possible data memory pages to obtain a 64K total data memory space. Indirect addressing is provided by the aux-

iliary registers (AR0–AR7). The seven types of indirect addressing are shown in Table 4. Bit-reversed indexed addressing modes allow efficient I/O to be performed for the resequencing of data points in a radix-2 FFT program.

Table 4 Addressing Modes of the TMS320C25

Addressing Mode	Operation
OP A	direct addressing
OP * (,NARP)	indirect; no change to AR.
OP *+(,NARP)	indirect; current AR is incremented.
OP *−(,NARP)	indirect; current AR is decremented.
OP *0+(,NARP)	indirect; AR0 is added to current AR.
OP *0−(,NARP)	indirect; AR0 is subtracted from current AR.
OP *BR0+(,NARP)	indirect; AR0 is added to current AR (with reverse carry propagation).
OP *BR0−(,NARP)	indirect; AR0 is subtracted from current AR (with reverse carry propagation).

Note: The optional NARP field specifies a new value of the ARP.

TMS320C25 System Configurations

The flexibility of the TMS320C25 allows systems configurations to satisfy a wide range of application requirements [16]. The TMS320C25 can be used in the following configurations:

- a stand-alone system (a single processor using 4K words of on-chip ROM and 544 words of on-chip RAM),
- parallel multiprocessing systems with shared global data memory, or
- host/peripheral coprocessing using interface control signals.

A minimal processing system is shown in Fig. 6 using external data RAM and PROM/EPROM. Parallel multiprocessing and host/peripheral coprocessing systems can be designed by taking advantage of the TMS320C25's direct memory access and global memory configuration capabilities.

In some digital processing tasks, the algorithm being implemented can be divided into sections with a distinct processor dedicated to each section. In this case, the first and second processors may share global data memory, as well as the second and third, the third and fourth, etc. Arbitration logic may be required to determine which section of the algorithm is executing and which processor has access to the global memory. With multiple processors dedicated to distinct sections of the algorithm, throughput can be increased via pipelined execution. The TMS320C25 is capable of allocating up to 32K words of data memory as global memory for multiprocessing applications.

The Third Generation of the TMS320 Family

The TMS320C30 [26]–[27] is Texas Instruments third-generation member of the TMS320 family of compatible digital signal processors. With a computational rate of 33 MFLOPS (million floating-point operations per second), the TMS320C30 far exceeds the performance of any programmable DSP available today. Total system performance has been maximized through internal parallelism, more than twenty-four thousand bytes of on-chip memory, single-cycle floating-point operations, and concurrent I/O. The total system cost is minimized with on-chip memory and on-chip peripherals such as timers and serial ports. Finally, the user's system design time is dramatically reduced with the availability of the floating-point operations, general-purpose instructions and features, and quality development tools.

The TMS320C30 provides the user with a level of performance that, at one time, was the exclusive domain of supercomputers. The strong architectural emphasis of providing a low-cost system solution to demanding arithmetic algorithms has resulted in the architecture shown in Fig. 7.

The key features of the TMS320C30 [26], [27] are as follows:

- 60-ns single-cycle execution time, 1-μm CMOS.
- Two 1K × 32-bit single-cycle dual-access RAM blocks.
- One 4K × 32-bit single-cycle dual-access ROM block.
- 64 × 32-bit instruction cache.
- 32-bit instruction and data words, 24-bit addresses.
- 32/40-bit floating-point and integer multiplier.
- 32/40-bit floating-point, integer, and logical ALU.
- 32-bit barrel shifter.
- Eight extended-precision registers.
- Two address-generators with eight auxiliary registers.
- On-chip Direct Memory Access (DMA) controller for concurrent I/O and CPU operation.
- Peripheral bus and modules for easy customization.
- High-level language support.
- Interlocked instructions for multiprocessing support.
- Zero overhead loops and single-cycle branches.

The architecture of the TMS320C30 is targeted at 60-ns and faster cycle times. To achieve such high-performance

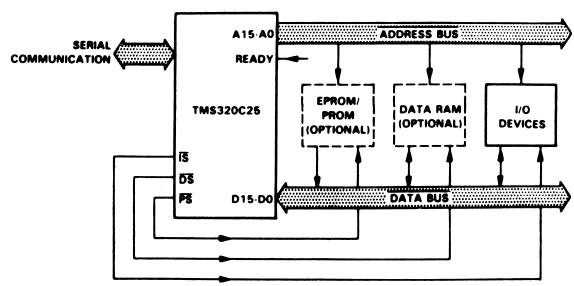


Fig. 6. Minimal processing system with external data RAM and PROM/EPROM.

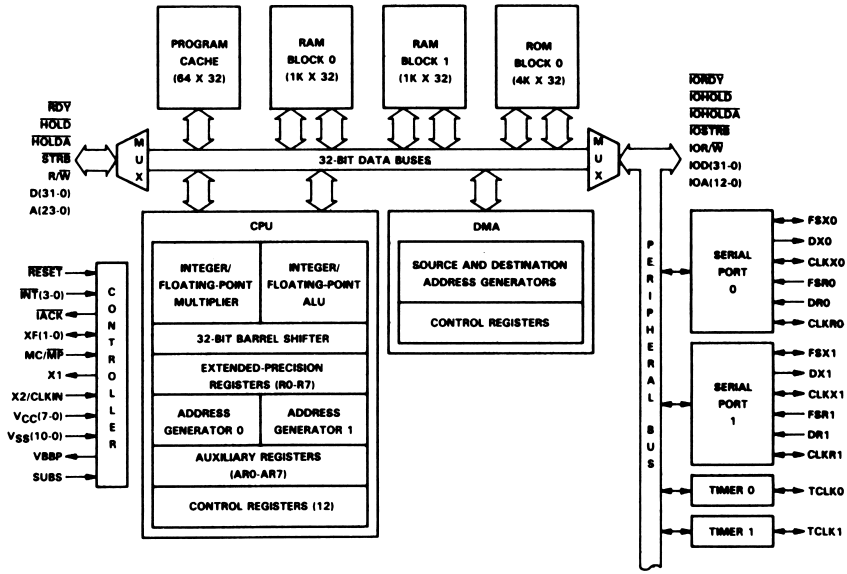


Fig. 7. TMS320C30 functional block diagram.

goals while still providing low-cost system solutions, the TMS320C30 is designed using Texas Instruments state-of-the-art 1-μm CMOS process. The TMS320C30's high system performance is achieved through a high degree of parallelism, the accuracy and precision of its floating-point units, its on-chip DMA controller that supports concurrent I/O, and its general-purpose features. At the heart of the architecture is the Central Processing Unit (CPU).

The CPU

The CPU consists of the following elements: floating-point/integer multiplier; ALU for performing floating-point, integer, and logical operations; auxiliary register arithmetic units; supporting register file, and associated buses. The multiplier of the CPU performs floating-point and integer multiplication. When performing floating-point multiplication, the inputs are 32-bit floating-point numbers, and the result is a 40-bit floating-point number. When performing integer multiplication, the input data is 24 bits and yields a 32-bit result. The ALU performs 32-bit integer, 32-bit logical, and 40-bit floating-point operations. Results of the multiplier and the ALU are always maintained in 32-bit integer or 40-bit floating-point formats. The TMS320C30 has the ability to perform, in a single cycle, parallel multiplies and adds (subtracts) on integer or floating-point data. It is this ability to perform floating-point multiplies and adds (subtracts) in a single cycle which give the TMS320C30 its peak computational rate of 33 MFLOPS.

Floating-point operations provide the user with a convenient and virtually trouble-free means of performing computations while maintaining accuracy and precision. The TMS320C30 implementation of floating-point arithmetic allows for floating-point operations at integer speeds. The floating-point capability allows the user to ignore, to a large extent, problems with overflow, operand alignment, and other burdensome tasks common to integer operations.

The register file contains 28 registers, which may be operated upon by the multiplier and ALU. The first eight of these registers (R0–R7) are the extended-precision registers, which support operations on 40-bit floating-point numbers and 32-bit integers.

The next eight registers (AR0–AR7) are the auxiliary registers, whose primary function is related to the generation of addresses. However, they also may be used as general-purpose 32-bit registers. Two auxiliary register arithmetic units (ARAU0 and ARAU1) can generate two addresses in a single cycle. The ARAUs operate in parallel with the multiplier and ALU. They support addressing with displacements, index registers (IR0 and IR1), and circular and bit-reversed addressing.

The remaining registers support a variety of system functions: addressing, stack management, processor status, block repeat, and interrupts.

Data Organization

Two integer formats are supported on the TMS320C30: a 16-bit format used for immediate integer operands and a 32-bit single-precision integer format.

Two unsigned-integer formats are available: a 16-bit format for immediate unsigned-integer operands and a 32-bit single-precision unsigned-integer format.

The three floating-point formats are assumed to be normalized, thus providing an extra bit of precision. The first

is a 16-bit short floating-point format for immediate floating-point operands, which consists of a 4-bit exponent, 1 sign bit, and an 11-bit fraction. The second is a single-precision format consisting of an 8-bit exponent, 1 sign bit, and a 23-bit fraction. The third is an extended-precision format consisting of an 8-bit exponent, 1 sign bit, and a 31-bit fraction.

The total memory space of the TMS320C30 is 16M (million) × 32 bits. A machine word is 32 bits, and all addressing is performed by word. Program, data, and I/O space are contained within the 16M-word address space.

RAM blocks 0 and 1 are each 1K × 32 bits. The ROM block is 4K × 32 bits. Each RAM block and ROM block is capable of supporting two data accesses in a single cycle. For example, the user may, in a single cycle, access a program word and a data word from the ROM block.

The separate program data, and DMA buses allow for parallel program fetches, data reads and writes, and DMA operations. Management of memory resources and busing is handled by the memory controller. For example, a typical mode of operation could involve a program fetch from the on-chip program cache, two data fetches from RAM block 0, and the DMA moving data from off-chip memory to RAM block 1. All of this can be done in parallel with no impact on the performance of the CPU.

A 64 × 32-bit instruction cache allows for maximum system performance with minimal system cost. The instruction cache stores often repeated sections of code. The code may then be fetched from the cache, thus greatly reducing the number of off-chip accesses necessary. This allows for code to be stored off-chip in slower, lower cost memories. Also, the external buses are freed, thus allowing for their use by the DMA or other devices in the system.

DMA

The TMS320C30 processes an on-chip Direct Memory Access (DMA) controller. The DMA controller is able to perform reads from and writes to any location in the memory map without interfering with the operation of the CPU. As a consequence, it is possible to interface the TMS320C30 to slow external memories and peripherals (A/Ds, serial ports, etc.) without affecting the computational throughput of the CPU. The result is improved system performance and decreased system cost.

The DMA controller contains its own address generators, source and destination registers, and transfer counter. Dedicated DMA address and data buses allow for operation with no conflicts between the CPU and DMA controller.

The DMA controller responds to interrupts in a similar way to the CPU. This ability allows the DMA to transfer data based upon the interrupts received. Thus I/O transfers that would normally be performed by the CPU may instead be performed by the DMA. Again, the CPU may continue processing data while the DMA receives or transmits data.

Peripherals

All peripheral modules are manipulated through memory-mapped registers located on a dedicated peripheral bus. This peripheral bus allows for the straightforward addition, removal, and creation of peripheral modules. The initial TMS320C30 peripheral library will include timers and serial ports. The peripheral library concept allows Texas Instruments to create new modules to serve a wide variety of applications. For example, the configuration of the TMS320C30 in Fig. 7 includes two timers and two serial ports.

Timers: The two timer modules are general-purpose timer/event counters, with two signaling modes and internal or external clocking.

Available to each timer is an I/O pin that can be used as an input clock to the timer or as an output signal driven by the timer. The pin may also be configured as a general-purpose I/O pin.

Serial Ports: The two serial ports are modular and totally independent. Each serial port can be configured to transfer 8, 16, 24, or 32 bits of data per frame. The clock for each serial port can originate either internally or externally. An internally generated divide-down clock is provided. The pins of the serial ports are configurable as general-purpose I/O pins. A special handshake mode allows TMS320C30s to communicate over their serial ports with guaranteed synchronization. The serial ports may also be configured to operate as timers.

External Interfaces

The TMS320C30 provides two external interfaces: the parallel interface and the I/O interface. The parallel interface consists of a 32-bit data bus, a 24-bit address bus, and a set of control signals. The I/O interface consists of a 32-bit data bus, a 13-bit address bus, and a set of control signals. Both ports support an external ready signal for wait-state generation and the use of software-controlled wait states.

The TMS320C30 supports four external interrupts, a number of internal interrupts, and a nonmaskable external reset signal. Two dedicated, general-purpose, external I/O flags, XF0 and XF1, may be configured as input or output pins under software control. These pins are also used by the interlocked instructions to support multiprocessor communication.

Pipelining In the TMS320C30

The operation of the TMS320C30 is controlled by five major functional units. The five major units and their function are as follows:

- *Fetch Unit* (F) which controls the program counter updates and fetches of the instruction words from memory.
- *Decode Unit* (D) which decodes the instruction word and controls address generation.
- *Read Unit* (R) which controls the operand reads from memory.
- *Execute Unit* (E) which reads operands from the register file, performs the necessary operation, and writes results back to the register file and memory.
- *DMA Channel* (DMA) which reads and writes memory concurrently with CPU operation.

Each instruction is operated upon by four of these stages; namely, fetch, decode, read, and execute. To provide for maximum processor throughput these units can perform in parallel with each unit operating on a different instruction. The overlapping of the fetch, decode, read, and execute operations of different instructions is called pipelining. The DMA controller runs concurrently with these units. The pipelining of these operations is key to the high per-

formance of the TMS320C30. The ability of the DMA to move data within the processor's memory space results in an even greater utilization of the CPU with fewer interruptions of the pipeline which inevitably yields greater performance.

The pipeline control of the TMS320C30 allows for extremely high-speed execution rate by allowing an effective rate of one execution per cycle. It also manages pipeline conflicts in a way that makes them transparent to the user.

While the pipelining of the different phases of an instruction is key to the performance of the TMS320C30, the designers felt it essential to avoid pipelining the operation of the multiplier or ALU. By ruling out this additional level of pipelining it was possible to greatly improve the processor's useability.

Instructions

The TMS320C30 instruction set is exceptionally well suited to digital signal processing and other numerically intensive applications. The TMS320C30 also possesses a full complement of general-purpose instructions. The instruction set is organized into the following groups:

- load and store instructions;
- two-operand arithmetic instructions;
- two-operand logical instructions;
- three-operand arithmetic instructions;
- three-operand logic instructions;
- parallel operation instructions;
- arithmetic/logical instruction with store instructions;
- program control instructions;
- interlocked operations instructions.

The load and store instructions perform the movement of a single word to and from the registers and memory. Included is the ability to load a register conditionally. This operation is particularly useful for locating the maximum and minimum of a set of data.

The two-operand arithmetic and logical instructions consist of a complete set of arithmetic instructions. They have two operands; src and dst for source and destination, respectively. The src operand may come from memory, a register, or be part of the instruction word. The dst operand is always a register. This portion of the instruction set includes floating-point integer and logical operations, support of multiprecision arithmetic, and 32-bit arithmetic and logical shifts.

The three-operand arithmetic and logical instructions are a subset of the two-operand arithmetic and logical instructions. They have three operands: two src operands and a dst operand. The src operands may come from memory or a register. The dst operand is always a register. These instructions allow for the reading of two operands from memory and/or the CPU register file in a single cycle.

The parallel operation instructions allow for a high degree of parallelism. They support very flexible, parallel floating-point and integer multiplies and adds. They also include the ability to load two registers in parallel.

The arithmetic/logical and store instructions support a high degree of parallelism, thus complementing the parallel operation instructions. They allow for the performance of an arithmetic or logical instruction between a register and an operand read from memory, in parallel with the storing of a register to memory. They also provide for extremely rapid operations on blocks of memory.

The program control instructions consist of all those operations that affect the program flow. This section of the instruction set includes a set of flexible and powerful constructs that allow for software control of the program flow. These fall into two main types: repeat modes and branching.

For many algorithms, there is an inner kernel of code where most of the execution time is spent. The repeat modes of the TMS320C30 allow for the implementation of zero overhead looping. Using the repeat modes allows these time-critical sections of code to be executed in the shortest possible time. The instructions supporting the repeat modes are RPTB (repeat a block of code) and RPTS (repeat a single instruction). Through the use of the dedicated stack-pointer, block repeats (RPTBs) may be nested.

The branching capabilities of the TMS320C30 include two main subsets: standard and delayed branches. Standard branches, as in any pipelined machine that comprehends them, empty the pipeline to guarantee correct management of the program counter. This results in a branch requiring, in the case of the TMS320C30, four cycles to execute. Included in this subset are calls and returns. A standard branch (BR) is illustrated below.

```
        BR      THREE   ; standard branch.
        MPYF            ; not executed.
        ADDF            ; not executed.
        SUBF            ; not executed.
        AND             ; not executed.
          .
          .
          .
THREE   MPYF            ; fetched 3 cycles after BR
                          is fetched.
          .
          .
          .
```

Delayed branches do not empty the pipe, but rather, guarantee that the next three instructions will be fetched before the program counter is modified by the branch. The result is a branch that only requires a single cycle. Every delayed branch has a standard branch counterpart. A delayed branch (BRD) is illustrated below.

```
        BRD     THREE   ; delayed branch.
        MPYF            ; executed.
        ADDF            ; executed.
        SUBF            ; executed.
        AND             ; not executed.
          .
          .
          .
THREE   MPYF            ; fetched after SUBF fetched.
          .
          .
          .
```

The combination of the repeat modes, standard branches, and delayed branches provides the user with a set of programming constructs which are well suited to a wide range of performance requirements.

The program control instructions also include conditional calls and returns. The decrement and branch conditionally instruction allows for efficient loop control by combining the comparison of a loop counter to zero with

the check of condition flags, i.e., floating-point overflow. The condition codes available include unsigned and signed comparisons, comparisons to zero, and comparisons based upon the status of individual condition flags. These conditions may be used with any of the conditional instructions.

The interlocked operations instructions support multiprocessor communication. Through the use of external signals, these instructions allow for powerful synchronization mechanisms, such as semaphores, to be implemented. The interlocked operations use the two external flag pins, XF0 and XF1. XF0 signals an interlocked-operation request and XF1 acts as an acknowledge signal for the requested interlocked operation. The interlocked operations include interlocked loads and stores. When an interlocked operation is performed the external request and acknowledge signals can be used to arbitrate between multiple processors sharing memory, semaphores, or counters.

Development and Support Tools

Digital signal processors are essentially application-specific microprocessors (or microcomputers). Like any other microprocessor, no matter how impressive the performance of the processor or the ease of interfacing, without good development tools and technical support, it is very difficult to design it into the system. In developing an application, problems are encountered and questions are asked. Oftentimes the tools and vendor support provided to the designer are the difference between the success and failure of the project.

The TMS320 family has a wide range of development tools available [25]. These tools range from very inexpensive evaluation modules for application evaluation and benchmarking purposes, assembler/linkers, and software simulators, to full-capability hardware emulators. A brief summary of these support tools is provided in the succeeding subsections.

Software Tools

Assembler/linkers and software simulators are available on PC and VAX for users to develop and debug TMS320 DSP algorithms. Their features are described as follows:

Assembler/Linker: The Macro Assembler translates assembly language source code into executable object code. The Linker permits a program to be designed and implemented in separate modules that will later be linked together to form the complete program.

Simulator: The Simulator simulates operations of the device in software to allow program verification and debug. The simulator uses the object code produced by the Macro Assembler/Linker.

C Complier: The C Compiler is a full implementation of the standard Kernighan and Ritchie C as defined in *The C Programming Language* [28]. The compiler supports the insertion of assembly language code into the C source code. The user may also write functions in assembly language, and then call these functions from the C source. Similarly, C functions may be called from assembly language. Variables defined in the C source may be accessed in assembly language modules and *vice versa*. The result is a complier that allows the user to tailor the amount of high-level programming versus the amount of assembly language according to his application. The C compiler is supported on the TMS320C25 and the TMS320C30.

Hardware Tools

Evaluation modules and emulation tools are available for in-circuit emulation and hardware program debugging for developing and testing DSP algorithms in a real product environment.

Evaluation Module (EVM): The EVM is a stand-alone single-board module that contains all of the tools necessary to evaluate the device as well as provide basic in-circuit emulation. The EVM contains a debug monitor, editor, assembler, reverse assembler, and software communications to a host computer or a line printer.

SoftWare Development System (SWDS): The SoftWare Development System is a PC plug-in card with similar functionality of the EVM.

Emulator (XDS): The eXtended Development System provides full-speed in-circuit emulation with real-time hardware breakpoint/trace and program execution capability from target memory. By setting breakpoints based on internal conditions or external events, execution of the program can be suspended and the XDS placed into the debug mode. In the debug mode, all registers and memory locations can be inspected and modified. Full-trace capabilities at full speed and a reverse assembler that translates machine code back into assembly instructions are included. The XDS system is designed to interface with either a terminal or a host computer. In addition to the above design tools, other development support is available [25]:

Applications

The TMS320 is designed for real-time DSP and other computation-intensive applications [4]. In these applications, the TMS320 provides an excellent means for executing signal processing algorithms such as fast Fourier transforms (FFTs), digital filters, frequency synthesis, correlation, and convolution. The TMS320 also provides for more general-purpose functions via bit-manipulation instructions, block data move capabilities, large program and data memory address spaces, and flexible memory mapping.

To introduce applications performed by the TMS320, digital filters will be used as examples. The remaining portion of this section will briefly cover applications, and conclude by showing some benchmarks.

Digital Filtering

As discussed several times in this paper, the FIR filter is simply the sum of products in a sampled data system. This was shown in (1). A simple implementation of the FIR filter uses the MACD instruction (multiply/accumulate and data move) for each filter tap, with the RPT/RPTK instruction repeating the MACD for each filter tap. As we saw earlier, a 256-tap FIR filter can be implemented by using the following two instructions:

```
RPTK   255
MACD   *-,COEFFP
```

In this example, the coefficients may be stored anywhere in program memory (reconfigurable on-chip RAM, on-chip ROM, or external memories). When the coefficients are

For this application, a large on-chip RAM of 544 words and on-chip ROM of 4K words on the TMS320C25 provides for a 256-tap adaptive filter (32-ms echo cancellation) to be executed in a single chip without external data or program memory.

High-Speed Modems: The TMS320 can perform numerous functions such a modulation/demodulation, adaptive equalization, and echo cancellation [21], [22]. For lower speed modems, such as Bell 212A and V.22 bis modems, the TMS320C17 provides the most cost-effective single-chip solution to these applications. For higher speed modems, such as the V.32, requiring more processing power and multiprocessing capabilities, the TMS320C25 and TMS320C30 are the designer's choice.

Voice Coding: Voice-coding techniques [3], [4], such as full-duplex 32-kbit/s ADPCM (CCITT G.721), CVSD, 16-kbit/s subband coders, and LPC, are frequently used in voice transmission and storage. Arithmetic speed, normalization, and the bit-manipulation capability of the TMS320 provide for implementation of these functions, usually in a single chip. For example, the TMS320C17 can be used as a single-chip ADPCM [4], subband [4], or LPC [4] coder. An application of voice coding is an ADPCM transcoder implemented in half-duplex on a single TMS320C17 or full-duplex on a TMS320C25 for telecommunication multiplexing applications. Another example is a secure-voice communication system, requiring voice coding, as well as data encryption and transmission over a public-switched network via a modem; the TMS320C25 offers an ideal solution.

Graphics/Image Processing Applications

In graphics and image processing applications [4], the ability to interface with a host processor is important. Both the TMS320C30 and the TMS320C25 multiprocessor interface enable them to be used in a variety of host/coprocessor configurations [4]. Graphics and image processing applications can use the large directly addressable external data space and global memory capability to allow graphical images in memory to be shared with a host processor, thus minimizing unnecessary data transfers. The indexed indirect addressing modes allow matrices to be processed row-by-row when performing matrix multiplication for three-dimensional image rotations, translations, and scaling.

The TMS320C30 has a number of features that support graphics and image processing extremely well. The floating-point capabilities allow for extremely precise computation of perspective transformations. They also support more sophisticated algorithms such as shading and hidden line removal, operations which are computationally intensive.

The large address space allows for straightforward addressing of large images or displays. The flexible addressing registers, coupled with the integer multiply, support powerful addressing of multiple-dimensional arrays. Vector-oriented instructions allow the user to efficiently manipulate large blocks of memory. Finally, the on-chip DMA controller allows the user to easily overlap the processing of data with its I/O.

High-Speed Control

High-speed control applications [4], [24] use the TMS320C17 and TMS320C25 general-purpose features for bit-test and logical operations, timing synchronization, and high data-transfer rate (ten million 16-bit words per second). Both devices can be used in closed-loop systems for control signal conditioning, filtering, high-speed computing, and multichannel multiplexing capabilities. The following demonstrates two typical control applications:

Disk Control: Digital filtering in a closed-loop actuation mechanism positions the read/write heads over the disk surface. Supplemented with many general-purpose features, the TMS320 can replace costly bit-slice/custom/analog solutions to perform such tasks as compensation, filtering, fine/coarse tuning, and other signal conditioning algorithms.

Robotics: Digital signal processing and bit-manipulation power, coupled with host interface, allow the TMS320C25 to be useful in robotics control [24]. The TMS320C25 can replace both the digital controllers and analog signal processing hardware for communication to a central host processor and for the performance of numerically intensive control functions.

Instrumentation

Instrumentation, such as spectrum analyzers and various high-speed/high-precision instruments, often requires a large data memory space and the high performance of a digital signal processor. The TMS320C25 and TMS320C30 are capable of performing very long-length FFTs and generating precision functions with minimal external hardware.

Numeric Processing

Numeric and array processing applications benefit from TMS320 performance. High throughput resulting from features, such as a fast cycle time and an on-chip hardware multiplier, combined with multiprocessing capabilities and data memory expansion, provide for a low-cost, easy-to-use replacement for a typical bit-slice solution. The TMS320C30's floating-point precision, high throughput, and interface flexibility are excellent for this application.

TMS320 Benchmarks

To complete the discussion on the applications that the TMS320 can perform, we will provide some benchmarks. The TMS320 has demonstrated impressive benchmarks in performing some of the common DSP routines and system applications. Table 5 shows typical TMS320 benchmarks [4].

Table 5 TMS320 Family Benchmarks

DSP Routines/Applications	First Generation	Second Generation	Third Generation
FIR filter tap	400 ns	100 ns	60 ns
256-tap FIR sample rate	9.25 kHz	37 kHz	>60 kHz
LMS adaptive FIR filter tap	700 ns	400 ns	180 ns
256-tap adaptive FIR filter sample rate	5.4 kHz	9.5 kHz	>20 kHz
Bi-quad filter element (five multiplies)	2 μs	1 μs	360 ns
Echo canceler (single chip)	8 ms	32 ms	>64 ms

Summary

This paper has discussed characteristics of digital signal processing and how these characteristics have influenced the architectural design of the Texas Instruments TMS320 family of digital signal processors. Three generations of the

TMS320 family were covered, and their support tools necessary to develop end-applications were briefly reviewed. The paper concluded with an overview of digital signal processing applications using these devices.

References

[1] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1975.

[2] A. V. Oppenheim, Ed., *Applications of Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1978.

[3] L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*. Englewood Cliffs, NJ: Prentice-Hall, 1978.

[4] K. Lin, Ed., *Digital Signal Processing Applications with the TMS320 Family*. Englewood Cliffs, NJ: Prentice-Hall, 1987

[5] A. V. Oppenhiem and R. W. Schafer, *Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1975.

[6] C. Burrus and T. Parks, *DFT/FFT and Convolution Algorithms*. New York, NY: Wiley, 1985.

[7] T. Parks and C. Burrus, *Digital Filter Design*. New York, NY: Wiley, 1987.

[8] J. Treichler, C. Johnson, and M. Larimore, *A Practical Guide to Adaptive Filter Design*. New York, NY: Wiley, 1987.

[9] P. Papamichalis, *Practical Approaches to Speech Coding*. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[10] R. Morris, *Digital Signal Processing Software*. Ottawa, Ont., Canada: DSPS Inc., 1983.

[11] K. McDonough, E. Caudel, S. Magar, and A. Leigh, "Microcomputer with 32-bit arithmetic does high-precision number crunching," *Electronics*, pp. 105–110, Feb. 24, 1982.

[12] S. Magar, E. Caudel, and A. Leigh, "A Microcomputer with digital signal processing capabflity," in *1982 Int. Solid State Conf. Dig. Tech. Pap.*, pp. 32–33, 284, 285.

[13] *First Generation TMS320 User's Guide*. Houston, TX: Texas Instruments Inc., 1987.

[14] *TMS320 First-Generation Digital Signal Processors Data Sheet*. Houston, TX: Texas Instruments Inc., 1987.

[15] *TMS32020 User's Guide. Houston, TX: Texas Instruments Inc., 1985.*

[16] *TMS320C25 User's Guide*. Houston, TX: Texas Instruments Inc., 1986.

[17] *TMS32011 User's Guide*. Houston, TX: Texas Instruments Inc., 1985.

[18] H. Cragon, "The elements of single-chip microcomputer architecture," *Comput. Mag.*, vol. 13, no. 10, pp. 27–41, Oct. 1980.

[19] S. Rosen, "Electronic computers: A historical survey," *Comput. Surv.*, vol. 1, no. 1, Mar. 1969.

[20] M. Honig and D. Messerschmitt, *Adaptive Filters*. Dordrecht, The Netherlands: Kluwer, 1984.

[21] R. Lucky *et al.*, *Principles of Data Communication*. New York, NY: McGraw-Hill, 1965.

[22] P. Van Gerwen *et al.*, "Microprocessor implementation of high speed data modems," *IEEE Trans. Commun.*, vol. COM-25, pp. 238–249, 1977.

[23] M. Bellanger, "New applications of digital signal processing in communications," *IEEE ASSP Mag.*, pp. 6–11, July 1986.

[24] Y. Wang, M. Andrews, S. Butner, and G. Beni, "Robot-controller system," in *Proc. Symp. on Incremental Motion Control Systems and Devices*, pp. 17–26, June 1986.

[25] *TMS320 Family Development Support Reference Guide*. Houston, TX: Texas Instruments Inc., 1986.

[26] R. Simar, T. Leigh, P. Koeppen, J. Leach, J. Potts, and D. Blalock, "A 40 MFLOPS digital signal processor: The first supercomputer on a chip," in *Proc. IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing*, Apr. 1987.

[27] *TMS320C30 User's Guide*. Houston, TX: Texas Instruments Inc., 1987.

[28] B. Kernighan and D. Ritchie, *The C Programming Language*. Englewood Cliffs, NJ: Prentice-Hall, 1978.