

Chapter 1

Introduction

The Motorola DSP56300 family of digital signal processors uses a programmable, 24-bit, fixed-point core. This core is a high-performance, single-clock-cycle-per-instruction engine that provides almost twice the performance of Motorola's popular DSP56000 family core, while retaining code compatibility. A variety of standard peripherals can be added around the DSP56300 family core (see **Figure 1-1**), such as serial ports, parallel ports, timers, different memory configurations (RAM and/or ROM), special-purpose coprocessors, and General Purpose Input/Output (GPIO) ports. Each peripheral interfaces to the DSP56300 core through a standard peripheral bus, allowing easy connection to standard or custom peripherals.

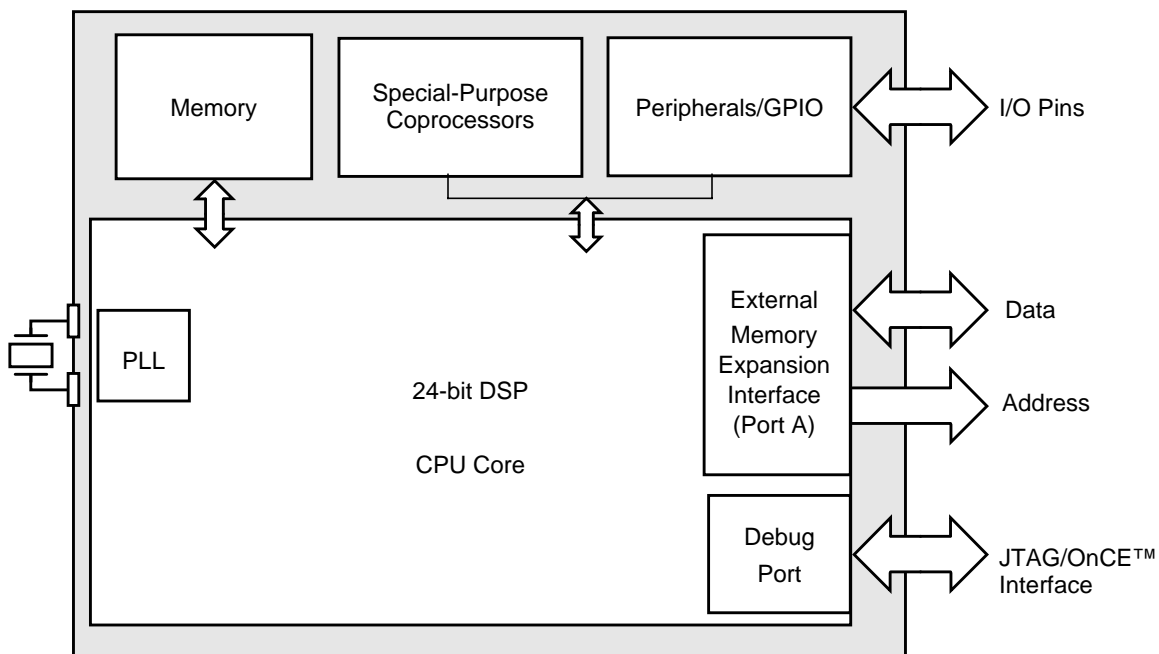


Figure 1-1. DSP56300 Family-Based DSP Chip

The combination of powerful instruction set, multiple internal buses, DMA channels, on-chip program and data memories, external buses, standard peripherals, and power management of the DSP56300 family make it an excellent solution for wireless or

wireline DSP applications from individual subscriber to infrastructure, as well as multimedia and high-end audio applications, including videoconferencing.

1.1 Core Overview

- One Million Instructions Per Second (MIPS) per MHz of operating speed
- Object code compatible with the DSP56000 core
- Highly parallel instruction set
- Data Arithmetic Logic Unit (Data ALU)
- Address Generation Unit (AGU)
- Program Control Unit (PCU)
- On-chip Instruction Cache Controller
- External Memory Interface (Port A)
- Phase Lock Loop (PLL)
- Hardware debugging support (JTAG TAP, OnCE™ module, and Address Trace Mode)
- Six-Channel Direct Memory Access (DMA) Controller
- Reduced power dissipation
 - Very low power CMOS design
 - Wait and Stop low-power standby modes
 - Fully-static logic

1.1.1 Data Arithmetic Logic Unit (Data ALU)

The Data Arithmetic Logic Unit (Data ALU) performs all the arithmetic and logical operations on data operands in the DSP56300 core. The components of the Data ALU are as follows:

- Fully pipelined 24×24 -bit parallel Multiplier-Accumulator (MAC) unit
- Bit Field Unit, comprising a 56-bit parallel barrel shifter (fast shift and normalization; bit stream generation and parsing)
- Conditional ALU instructions
- 24-bit or 16-bit arithmetic support under software control
- Four 24-bit input general purpose registers: X1, X0, Y1, and Y0
- Six Data ALU registers (A2, A1, A0, B2, B1, and B0) that are concatenated into two general purpose 56-bit accumulators and accumulator shifters (A and B)

- Two data bus shifter/limiter circuits

The Data ALU registers can be read or written over the X Data Bus (XDB) and the Y Data Bus (YDB) as 24- or 48-bit operands. The source operands for the Data ALU, which can be 24, 48, or 56 bits, always originate from the Data ALU registers. The results of all Data ALU operations are stored in an accumulator. All Data ALU operations are performed in two clock cycles in pipeline fashion so that a new instruction can be initiated in every clock, yielding an effective execution rate of one instruction per clock cycle.

The Multiplier-Accumulator (MAC) unit comprises the main arithmetic processing unit of the DSP56300 core and performs all of the calculations on data operands. For arithmetic instructions, the unit accepts as many as three input operands and outputs one 56-bit result of the following form:

```
Extension:Most Significant Product:Least
Significant Product (EXT:MSP:LSP)
```

The multiplier executes 24-bit \times 24-bit, parallel fractional multiplies between two's-complement signed, unsigned, or mixed operands. The 48-bit product is right-justified and added to the 56-bit contents of either the A or B accumulator. A 56-bit result can be stored as a 24-bit operand by truncating or rounding the LSP. The LSP can be either truncated or rounded into the MSP.

1.1.2 Address Generation Unit (AGU)

The Address Generation Unit (AGU) performs the effective address calculations for addressing data operands in memory and contains the integer arithmetic and registers used to generate the addresses. The AGU operates in parallel with the other core resource, and so minimizes address-generation overhead of instruction sequences. It implements four types of address arithmetic:

- Linear
- Modulo
- Multiple wrap-around modulo
- Reverse-carry

These arithmetic types easily allow creation of data structures in memory for FIFOs (queues), delay lines, circular buffers, stacks, and bit-reversed FFT buffers. Data is manipulated by updating address registers (pointers) rather than moving large blocks of data. The contents of the address modifier register, Mn, define the type of arithmetic to be performed for addressing mode calculations. For modulo arithmetic, the contents of Mn also specify the modulus. All address register indirect modes can be used with any address

modifier. Each address register, R_n , has an associated modifier register, M_n . The following address modifier types are available:

- Linear addressing—Useful for general-purpose addressing
- Modulo addressing—Useful for creating circular buffers for FIFOs
- Multiple wrap-around modulo addressing—Useful for decimation, interpolation and waveform generation since the multiple wrap-around capability can be used for argument reduction
- Reverse-carry (bit-reverse) addressing—Useful for 2^k -point FFT addressing

The AGU is divided into halves, each with its own Address Arithmetic Logic Unit (Address ALU), one to generate 24-bit addresses every cycle for the X space and one for the Y space. Each Address ALU can update one address register from its respective address register file during one instruction cycle. Each Address ALU has four sets of register triplets; each triplet is composed of an address register, an offset register, and a modifier register. The contents of the associated modifier register specify the type of arithmetic to use in the address register update calculation. The modifier value is decoded in the Address ALU.

Each Address ALU contains a 24-bit full adder, which is an offset adder. A second full adder—which is a modulo adder—adds the summed result of the first full adder to a modulo value that is stored in its respective modifier register. A third full adder, which is a reverse-carry adder, is also provided. The offset adder and the reverse-carry adder operate in parallel and share common inputs. The only difference between them is that the carry propagates in opposite directions. The modifier value determines which of the three summed results of the full adders is output. For details on the AGU, see **Chapter 4, Address Generation Unit**.

1.2 Program Control Unit (PCU)

The Program Control Unit (PCU) performs instruction fetch, instruction decoding, hardware DO loop control, and exception processing. The PCU implements a seven-stage pipeline and controls the different processing states of the DSP56300 core. The PCU consists of three hardware blocks:

- *Program Decode Controller (PDC)*: Decodes the 24-bit instruction loaded into the instruction latch and generates all necessary pipeline control signals
- *Program Address Generator (PAG)*: Contains the hardware for program address generation, system stack, and loop control

- *Program Interrupt Controller (PIC)*: Arbitrates among all interrupt requests (internal interrupts and the five external requests \overline{IRQA} , \overline{IRQB} , \overline{IRQC} , \overline{IRQD} , and \overline{NMI}), and generates the appropriate interrupt vector address

PCU features include:

- Position independent code (PIC) support
- Addressing modes optimized for DSP applications (including immediate offsets)
- On-chip instruction cache controller
- On-chip memory-expandable hardware stack
- Nested hardware DO loops
- Fast auto-return interrupts
- Program Address Trace mode support

1.3 On-chip Instruction Cache Controller

The instruction cache functions as a buffer memory between external memory and the DSP core processor. When code executes, the code locations requested by the set of instructions are copied into the instruction cache for direct access by the core processor. If the same code is used frequently in a set of program instructions, storage of these instructions in the cache yields an increase in throughput, because the time required to access them through the external bus is eliminated. The DSP56300 instruction set provides specific cache instructions that permit you to lock sectors of the cache and to flush the cache contents under software control. The instruction cache can control either 1K of instruction cache memory, with the following features:

- Software controlled Cache Enable (CE) bit in the Extended Mode Register (EMR) in the Status Register (SR)
- Instruction Cache size of 1024 or 24-bit words
- 8-way, fully associative instruction cache with sectored placement policy
- 1- to 4-word transfer granularity
- Least recently used (LRU) sector replacement algorithm
- Transparent operation (i.e., no user management is required)
- Individual sector locking/unlocking
- Global cache flush controlled by software
- Cache controller status observable via the JTAG/OnCE port

1.4 Port A External Memory Interface

Port A is an external memory interface for memory expansion or memory-mapped I/O. Its programmable nature supports a low part-count connection to fast or slow SRAMs, DRAMs, I/O devices, and multiple bus master systems. The Port A data bus is 24 bits wide with a separate address bus that is 24 bits wide in some DSP56300 processors and less than 24 bits in others. External memory is divided into three possible $16\text{ M} \times 24\text{-bit}$ spaces: X data, Y data, and program memory. Each or all spaces can be accessed to a given external memory under software control. See the memory map in **Chapter 11, *Operating Modes and Memory Spaces*** for memory space that is not accessible over Port A. An internal wait state generator can be programmed to statically insert up to 31 wait states for access to slower memory or I/O devices. A Transfer Acknowledge ($\overline{\text{TA}}$) signal allows an external device to dynamically control the number of wait states inserted in a bus access operation. Bus arbitration signals allow an external device to use the bus while internal operations continue using internal memory. See the memory map in the device-specific user's manual for memory space that is not accessible.

The Address Attribute (AA) lines operate as memory-mapped chip selects or as address lines to external devices, depending upon the mode selected. Some DSP56300 chips have eighteen address lines. For these DSPs, if all four AA lines are used as address lines, the total addressable external memory per space (X data, Y data, and program) is $4\text{ M} \times 24\text{-bit}$. If all four AA lines are used, the memory must always be selected because no AA lines are available for chip select. As a result, an external read or write outside the 4M range could still go to the external memory (depending on the settings of the AA registers).

1.5 Phase Lock Loop (PLL) and Clock Generator

The clock generator in the DSP56300 core is composed of two main blocks:

- *Phase Lock Loop (PLL)*: Clock-input division, frequency multiplication, and skew elimination
- *Clock Generator (CLKGEN)*: Low-power division and clock pulse generation and change of low-power Divide Factor (DF) without loss of lock

The PLL allows the processor to operate at a high internal clock frequency using a low frequency clock input, a feature that offers two immediate benefits:

- A lower frequency clock input reduces the overall electromagnetic interference generated by a system.
- The ability to oscillate at different frequencies reduces costs by eliminating the need to add additional oscillators to a system.

1.6 Hardware Debugging Support

The DSP56300 core provides a dedicated user-accessible Test Access Port (TAP) based on the *IEEE 1149.1 Standard Test Access Port and Boundary Scan Architecture*. Problems associated with testing high-density circuit boards have led to development of this standard under the sponsorship of the Test Technology Committee of IEEE and the Joint Test Action Group (JTAG). The DSP56300 core implementation supports circuit-board test strategies based on this standard. The test logic includes a TAP consisting of four dedicated signal pins, a 16-state controller, and three test data registers. A Boundary Scan Register (BSR) links all device signal pins into a single shift register. The test logic is implemented utilizing static logic design and is completely independent of the device system logic.

An On-chip Emulation (OnCE) port supports hardware and software development on the DSP56300 core processor. It allows nonintrusive interaction with the core and its peripherals so that developers can examine registers, memory, or on-chip peripherals. This facilitates hardware and software development on the DSP56300 core processor. OnCE module functions are provided through the JTAG TAP pins. More information on the JTAG/OnCE port is provided in **Chapter 7, *Debugging Support***.

A third debugging feature is the Address Trace mode, which reflects internal Program RAM accesses at the external port. This mode is invoked by setting the Address Tracing Enable (ATE), which is bit 15 in the Operating Mode Register (OMR)¹. Once active, both internal and external program memory accesses are valid at the rising edge of CLKOUT. The \overline{BR} signal distinguishes internal from external accesses.

1.7 Direct Memory Access (DMA)

The Direct Memory Access (DMA) block permits data transfers without the interaction of the core program. It supports any combination of internal memory, internal peripheral I/O and external memory as source and destination during accesses. The DMA block has the following features:

- Six DMA channels supporting internal and external accesses
- One-, two-, and three-dimensional transfers (including circular buffering)
- End-of-block-transfer interrupts
- Triggering from interrupt lines and all peripherals

1. For details on the Operating Mode Register (OMR), see **Section 5.4.1.1, "Operating Mode Register."**

1.8 Introduction to Digital Signal Processing

Digital signal processing is the arithmetic processing of real-time signals that are sampled at regular intervals and digitized. Examples of digital signal processing include the following:

- Filtering
- Convolution (mixing two signals)
- Correlation (comparing two signals)
- Rectification, amplification, and/or transformation

Historically, all of these functions require analog circuits. Only recently has semiconductor technology provided the processing power necessary to perform these and other functions digitally using Digital Signal Processors (DSPs). **Figure 1-2** shows an example of analog signal processing. The circuit in the illustration filters a signal from a sensor using an operational amplifier and controls an actuator with the result. Since the ideal filter is impossible to design, the engineer must design the filter for acceptable response considering variations in temperature, component aging, power supply variation, and component accuracy. The resulting circuit typically has low noise immunity, requires adjustments, and is difficult to modify.

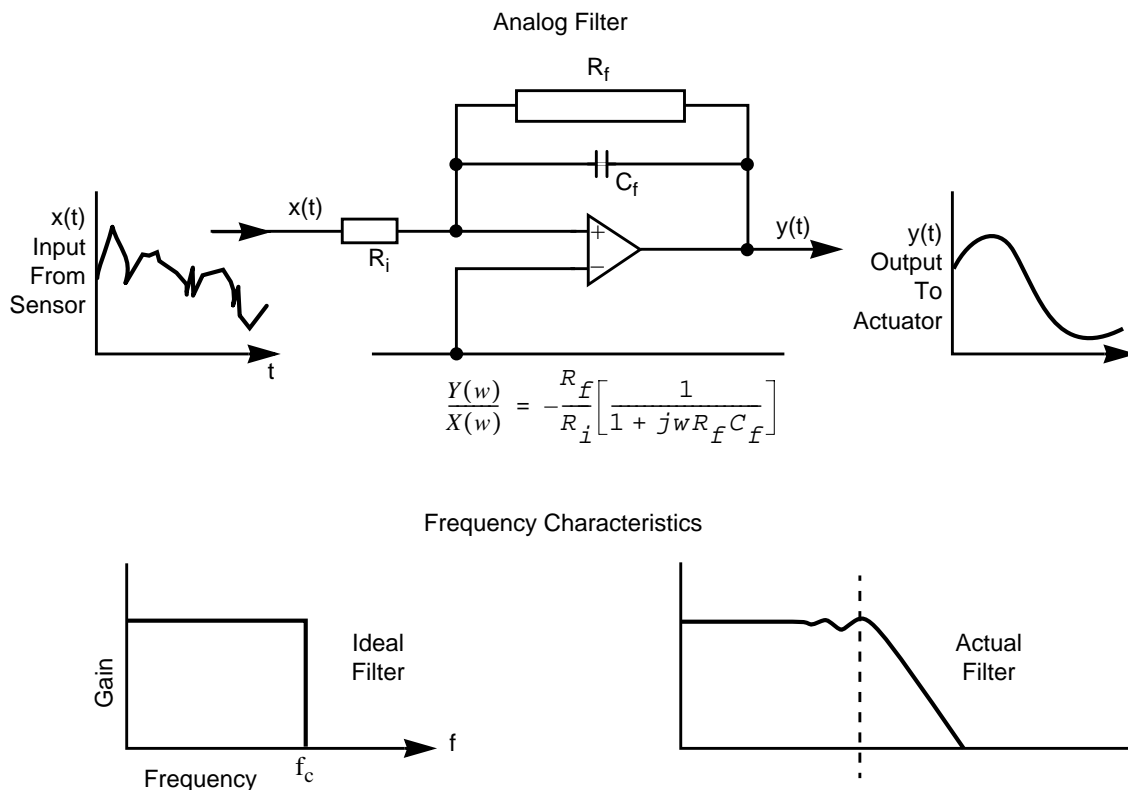


Figure 1-2. Analog Signal Processing

The equivalent circuit using a DSP is shown in **Figure 1-3**. This application requires an Analog-to-Digital (A/D) converter and Digital-to-Analog (D/A) converter in addition to the DSP. Even with these additional parts, the component count can be lower using a DSP due to the high integration available with current components. Processing in this circuit begins by band-limiting the input signal with an anti-alias filter, eliminating out-of-band signals that can be aliased back into the pass band due to the sampling process. The signal is then sampled, digitized with an A/D converter and sent to the DSP. The filter implemented by the DSP is strictly a matter of software. The DSP can directly employ any filter that can also be implemented using analog techniques. Also, adaptive filters are easy to implement using DSP but very difficult to implement using analog techniques.

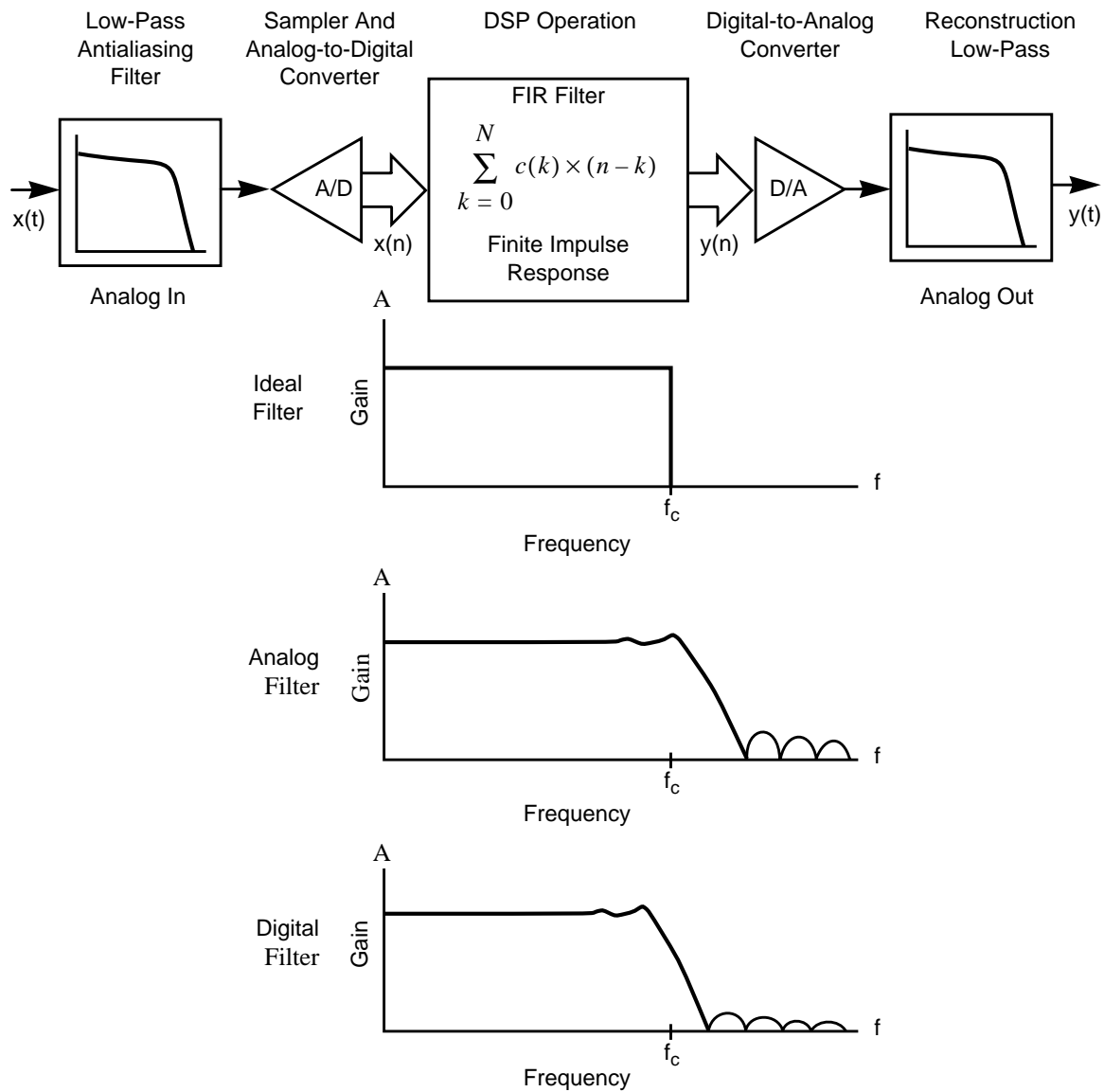


Figure 1-3. Digital Signal Processing

The DSP output is processed by a D/A converter and is low-pass filtered to remove the effects of digitizing. The advantages of using the DSP include:

- Fewer components
- Stable, deterministic performance
- No filter adjustments
- Wide range of applications
- Filters with much closer tolerances
- High noise immunity
- Easily implemented adaptive filters
- Built-in self-test capability
- Better power supply rejection

The DSP56300 family is not a custom IC designed for a particular application; it is designed as a general-purpose DSP architecture to efficiently execute commonly used DSP benchmarks and controller code in minimal time.

Figure 1-4 shows the following key attributes of a DSP:

- Multiply/Accumulate (MAC) operation
- Fetching up to two operands per instruction cycle for the MAC
- Program control to provide versatile operation
- Input/output to move data in and out of the DSP

The MAC operation is the fundamental operation used in DSP. The DSP56300 family of processors has a modified dual Harvard architecture optimized for MAC operations.

Figure 1-4 shows how the DSP56300 family architecture matches the shape of the MAC operation. The two operands, $C()$ and $X()$, are directed to a multiply operation, and the result is summed. This process is built into the chip using two separate memories (X and Y) to feed a single-cycle MAC unit. The entire process must occur under program control to direct the correct operands to the multiplier and save the accumulator as needed. Since the two memories and the MAC unit are independent, the DSP can perform two moves, a multiply and an accumulate, in a single operation. As a result, many DSP benchmarks execute very efficiently for a single-multiplier architecture.

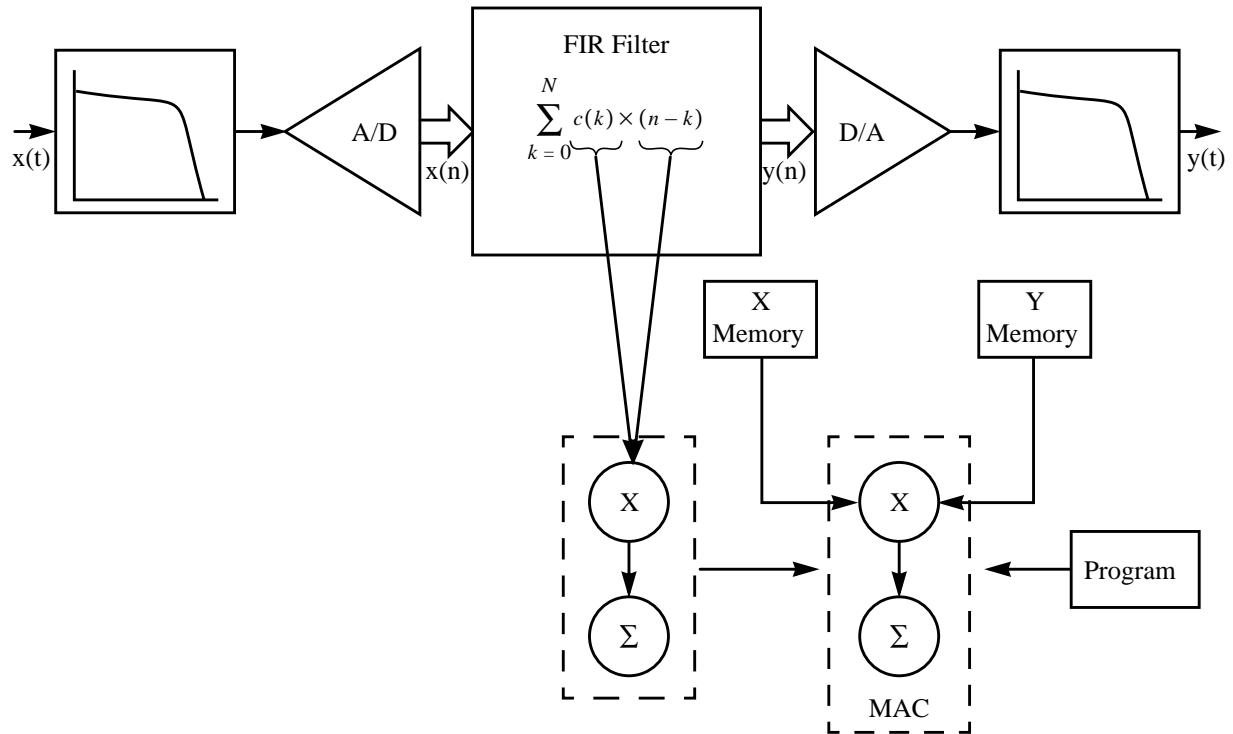


Figure 1-4. Mapping DSP Algorithms into Hardware

1.9 Summary of Features

The high throughput of the DSP56300 family of processors makes them well-suited for wireless and wireline communication, high-speed control, efficient signal processing, numeric processing, and computer and audio applications. The main features that contribute to this high throughput include the following:

- *Speed*: The DSP56300 family supports most high-performance DSP applications.
- *Precision*: The data paths are 24 bits wide, providing 144 dB of dynamic range; intermediate results held in the 56-bit accumulators can range over 336 dB.
- *Parallelism*: Each on-chip execution unit, memory, and peripheral operates independently and in parallel with the other units through a sophisticated bus system. The Data ALU, AGU, and program controller operate in parallel so that the following can execute in a single instruction:
 - An instruction pre-fetch
 - A 24-bit \times 24-bit multiplication
 - A 54-bit addition
 - Two data moves
 - Two address-pointer updates using either linear or modulo arithmetic

- *Flexibility*: While many other DSPs need external communications circuitry to interface with peripheral circuits (such as A/D converters, D/A converters, or host processors), the DSP56300 family provides on-chip serial and parallel interfaces that can support various configurations of memory and peripheral modules. The peripherals are interfaced to the DSP56300 family core through a peripheral interface bus that provides a common interface to many different peripherals.
- *Sophisticated Debugging*: Motorola's On-Chip Emulation (OnCE) technology allows simple, inexpensive, and speed independent access to the internal registers for debugging. With the OnCE module, you can determine easily the exact status of the registers and memory locations and what instructions were last executed.
- *Phase Locked Loop (PLL)-Based Clocking*: The PLL allows the chip to use almost any available external system clock for full-speed operation, while also supplying an output clock synchronized to a synthesized internal core clock. It improves the synchronous timing of the external memory port, eliminating the timing skew common on other processors.
- *Invisible Pipeline*: The seven-stage instruction pipeline is essentially invisible to the programmer, allowing straightforward program development in either assembly language or high-level languages such as C or C++.
- *Instruction Set*: The instruction mnemonics are similar to those used for microcontroller units, making the transition from programming microprocessors to programming the chip as easy as possible. New microcontroller instructions, addressing modes, and bit field instructions allow for significant decreases in program code size. The orthogonal syntax controls the parallel execution units. The hardware DO loop instruction and the repeat (REP) instruction make writing straight-line code obsolete.
- *Low Power*: Designed in CMOS, the DSP56300 family consumes very little power. Two additional low-power modes, Stop and Wait, further reduce power requirements. Wait is a low-power mode in which the DSP56300 family core is shut down, but the peripherals and interrupt controller continue to operate so that an interrupt can bring the chip out of Wait mode. In Stop mode, even more of the circuitry is shut down for the lowest power consumption. Several different ways exist to bring the chip out of Stop mode: hardware $\overline{\text{RESET}}$, $\overline{\text{IRQA}}$, and $\overline{\text{DE}}$.

1.10 Manual Organization

This manual describes the DSP56300 family Central Processing Unit in detail. Use this manual in conjunction with the appropriate DSP56300 family member user's manual, which describes the memory, operating modes, and peripheral modules. The appropriate DSP56300 family technical data sheet describes timing, pinout, and packaging.

This manual presents practical information to help the user accomplish the following:

- Understand the operation and instruction set of the DSP56300 family
- Write code for DSP algorithms
- Write code for general control tasks
- Write code for communication routines
- Write code for data manipulation algorithms

Table 1-1 describes the contents of each chapter and each appendix.

Table 1-1 DSP Family Manual Chapters

Chapter/ Appendix	Title and Description
2	Core Architecture Overview —The DSP56300 family core architecture consists of an External Memory Interface (Port A), Data Arithmetic Logic Unit (Data ALU), Address Generation Unit (AGU), Program Control Unit (PCU), Direct Memory Access (DMA) controller, Phase Lock Loop (PLL) circuit, and a JTAG/On-Chip Emulation (OnCE) port. Chapter 2 describes each subsystem and the buses interconnecting the major components in the DSP56300 family central processing module. Chapter 2 also describes five of the six processing states (Normal, Exception, Reset, Wait, and Stop). The sixth processing state (Debug) is covered more completely in Chapter 7, Debugging Support .
3	Data Arithmetic Logic Unit —Data ALU architecture, its programming model, an introduction to fractional and integer arithmetic, and a discussion of other topics such as unsigned and multi-precision arithmetic on the DSP56300 family.
4	Address Generation Unit —AGU architecture, its programming model, addressing modes, and address modifiers.
5	Program Control Unit —Program controller architecture, its programming model, and hardware looping. Note, however, that the different processing states of the DSP56300 family core, including interrupt processing, are described in Chapter 2, Core Architecture Overview .
6	PLL and Clock Generator —Details the PLL, its programming model, and its general operation.
7	Debugging Support —Combined JTAG/OnCE port and its functions. These two are integrally related, sharing the same pins for I/O.
8	Instruction Cache —Operation of the Instruction Cache controller and memory space.
9	External Memory Interface (Port A) —The External Memory Interface, its programming model, and guidelines for interfacing SRAM and DRAM.
10	Direct Memory Access Controller —The six-channel DMA controller, its programming model, and interactions with the core and peripherals.
11	Operating Modes and Memory Spaces —Operating modes and memory spaces in the DSP56300 family.
A	Guide to the Instruction Set — The DSP56300 family instruction format as well as partial encodings for use in instruction encoding

Table 1-1 DSP Family Manual Chapters (Continued)

Chapter/ Appendix	Title and Description
B	Instruction Set — Each DSP56300 family instruction, its use, and its effect on the processor.
C	Instruction Timing — Various aspects of execution timing analysis for each instruction, sequences that may cause timing delays or stalls, and programming restrictions.
D	Benchmark Programs —DSP56300 family benchmark example programs and results.
E	From CDR Process to HiP Process — General differences between DSP56300 family derivatives that use Motorola's Communication Design Rules (CDR) process technology and derivatives that use Motorola's High-Performance (HiP) process technology; software and hardware design implications.

Note: The latest electronic version of this document as well as other DSP documentation (including user's manuals, product briefs, technical data sheets, and errata) can be found on the Motorola DSP World Wide Web site. See the back cover of this publication for the Motorola DSP World Wide Web site address.