1 INTRODUCTION

Overview

This manual, often referred to as a users programmer's guide, contains an instruction set description required for the design and programming of TigerSHARC®-based systems. In addition to this manual, hardware designers should refer to the *TigerSHARC® Data Sheet* for timing, electrical, and package specifications.

The TigerSHARC® 128-bit digital signal processor is a high performance next generation version of the ADSP-2106x SHARC. The TigerSHARC® sets a new standard of performance for digital signal processors, combining multiple computation units for floating-point and fixed-point processing as well as very wide word widths. The TigerSHARC® maintains a 'system-on-a-chip' scalable computing design philosophy, including a 6-Mbit, on-chip SRAM, integrated IO peripherals, a host processor interface, DMA controllers, link ports and shared bus connectivity for glueless MDSP (Multi Digital Signal Processing).

In addition to providing unprecedented performance in DSP applications in raw MFLOPS and MIPS, the TigerSHARC® boosts performance measures such as MFLOPS/Watt and MFLOPS/square inch in multiprocessing applications.

The processor operates with a two cycle arithmetic pipeline. The branch pipeline is two to six cycles, and because of this deep branch, a branch target buffer (BTB) is implemented to reduce branch delay. The two identical computation units support floating-point as well as fixed-point arithmetic.

High performance is facilitated by the ability to execute up to four 32-bit wide instructions per cycle. The TigerSHARC® uses a variation of a *static superscalar* architecture to allow the programmer to specify which instructions are executed in parallel in each cycle. The instructions do not have to be aligned in memory so that program memory is not wasted.

The 6-Mbit internal memory is divided into three 128-bit wide memory blocks. Each of the three internal address/data bus pairs connect to one of the three memory blocks. The three memory blocks can be used for triple accesses every cycle where each memory block can access up to four, 32-bit words in a cycle.

The external port cluster bus is 64 bits wide. The high IO bandwidth complements the high processing speeds of the core. To facilitate the high clock rate, the TigerSHARC® uses a pipe-lined external bus with programmable pipeline depth for inter-processor communications and for Synchronous SRAM and DRAM (SSRAM and SDRAM).

The four link ports support point to point high bandwith data transfer. Communication ports have hardware supported two-way communication.

Figure 1-1 on page 1-4 illustrates the micro architecture of the TigerSHARC®, showing a detailed block diagram of the processor and presenting the following architectural features:

- Dual computation blocks—X and Y—each consisting of a multiplier, ALU, shifter and a 32-word register file.
- Dual integer ALUs—J and K—each containing a 32-bit ALU and 32-word register file.
- Program sequencer—Controls the program flow. It contains an instruction alignment buffer (IAB) and a branch target buffer (BTB).
- Three 128-bit buses that provide high bandwidth connectivity between all blocks of 9.6 Gbytes/sec.

- External port Interface that includes the Host Interface, SDRAM controller, static pipelined interface, four DMA channels, four link ports—each with two DMA channels— and multiprocessing support.
- Three internal memory blocks—M0, M1 and M2—16K rows each and 128-bits wide (a total of 2Mbit).
- Debug features
- JTAG Test Access Port

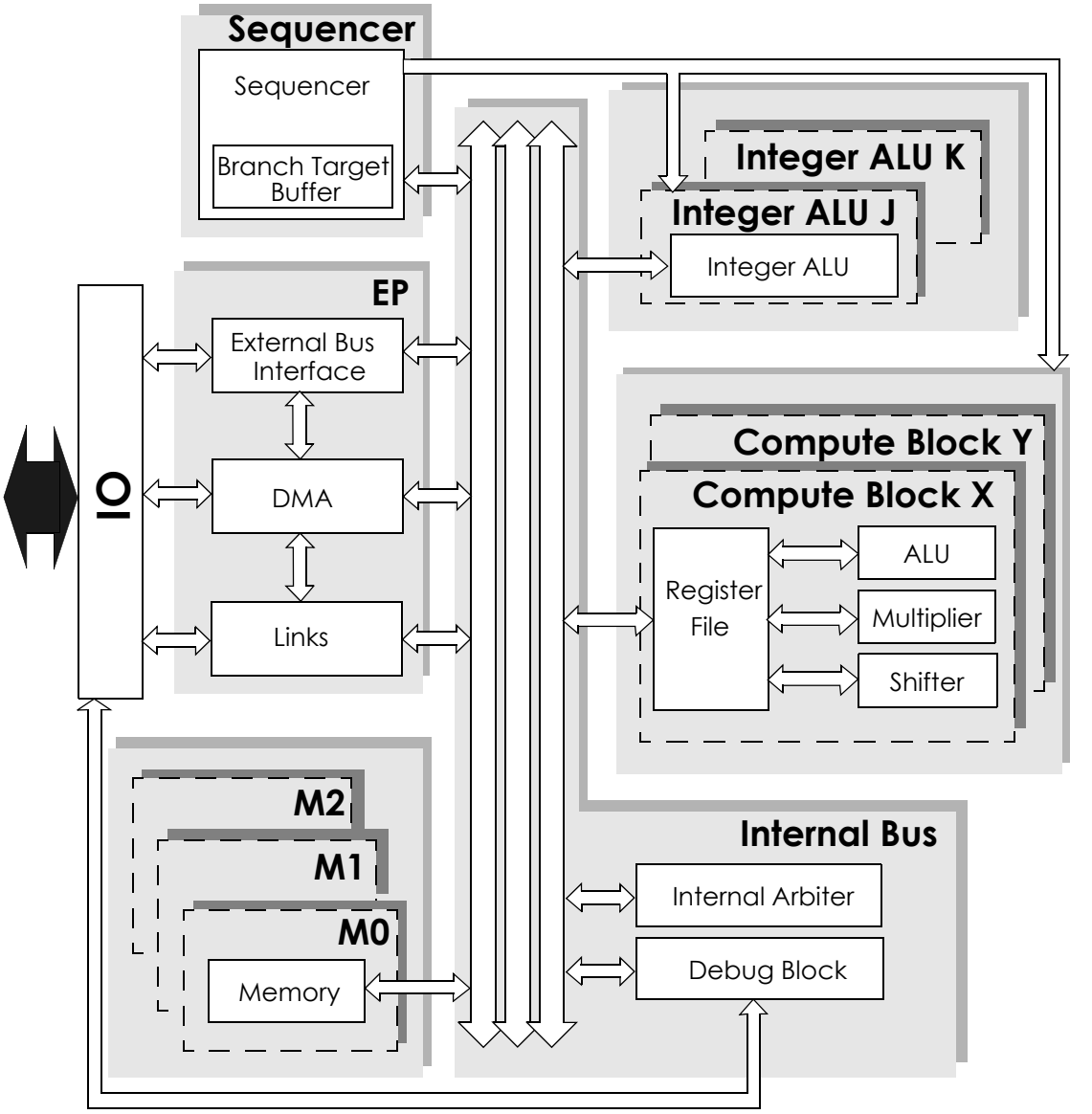

Figure 1-1. Chip Level Block Diagram

The TigerSHARC® external port provides interface to external memory, memory-mapped IO, host processor, and to additional TigerSHARC®s. The external port performs external bus arbitration as well as supplying control signals to shared, global memory and IO devices.

Figure 1- illustrates a typical single-processor system. Multiprocessor systems are illustrated in Figure 1-3 on page 1-6, and discussed later in "Scalability and Multiprocessing" on page 1-9.

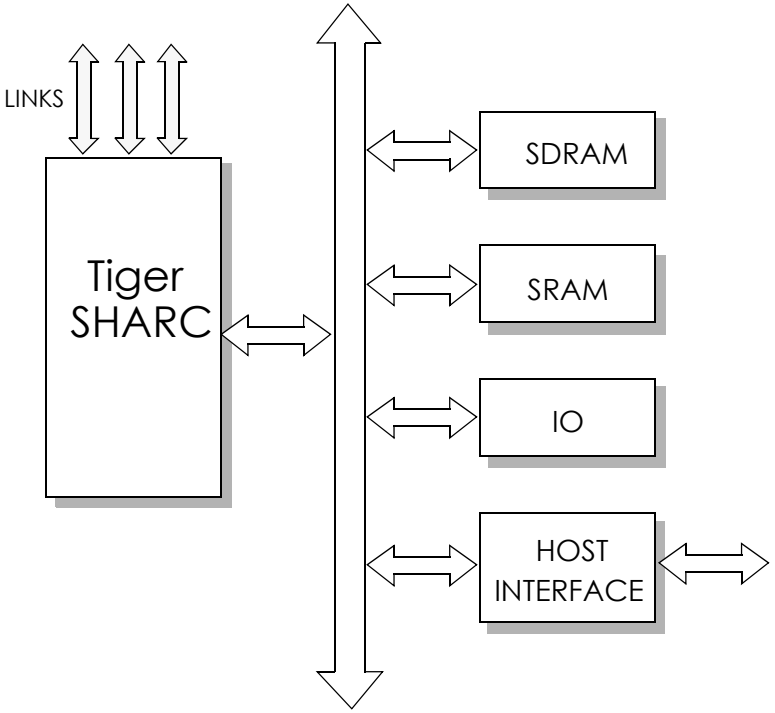

Figure 1-2. Single Processor Configuration

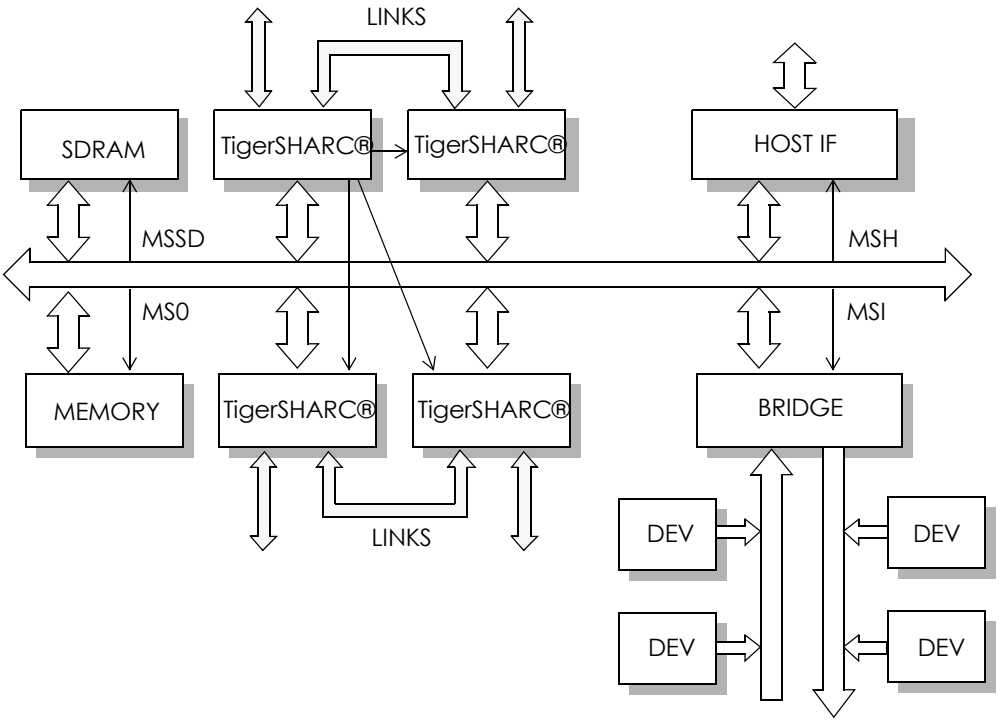

Figure 1-3. Multiprocessing Cluster Configuration

Key Architectural Features

Key architectural features are outlined in the following sub-sections.

Parallel Operations

The following section describes the TigerSHARC® parallel operations capability.

During compute intensive operations, one or both integer ALUs compute or generate addresses for fetching up to two quad operands from two memory blocks, while the program sequencer simultaneously fetches the next quad instruction from the third memory block. In parallel, the computation units can operate on previously fetched operands while the sequencer prepares for a branch.

While the core processor is doing the above, the DMA channels can be replenishing the internal memories in the background with quad data from either the external port or the communication ports.

Core Processor

The processing core of the TigerSHARC® reaches exceptionally high performance due to the following features:

- Computation pipeline
- Dual computation units
- Execution of up to four instructions per cycle
- Access of up to eight words per cycle from memory

The two computation units perform up to six floating-point or 24 fixed-point operations per cycle.

Each multiplier and ALU unit can execute four 16-bit fixed- point operations per cycle (SIMD). This boosts performance of critical imaging and signal processing applications that use fixed-point data.

Internal Memories

The on-chip memory consists of three blocks of two Mbits each. Each block is 128 bits (four words) wide, thus providing high bandwidth sufficient to support both computation units, the instruction stream and external IO, even in very intensive operations. The TigerSHARC® provides access to program and two data operands without memory or bus constraints. The memory blocks can store instructions and data interchangeably.

Quad Instruction Execution

The TigerSHARC® can execute up to four instructions per cycle from a single memory block, due to the 128-bit wide access per cycle. The ability to execute several instructions in a single cycle derives from a *static superscalar*[*] architectural concept. This is not strictly a superscalar architecture because the instructions executed in each cycle are specified in the instruction by the programmer or by the compiler, and not by the chip hardware. There is also no instruction re-ordering. Register dependencies are, however, examined by the hardware and stalls are generated where appropriate. Code is fully compacted in memory and there are no alignment restrictions for instruction lines.

Quad Data Access

Instructions specify whether one, two or four words are to be loaded or stored. Quad words[†] can be aligned on a quad word boundary and long words aligned on a long word boundary. This, however, is not necessary when loading data to computation units because a data alignment buffer (DAB) automatically aligns quad words that are not aligned in memory.

---

[*] *static superscalar* is an ADI trademark

[†] A quad word is comprised of 128 bits of data.

Up to four data words from each memory block can be supplied to each computation unit, meaning that new data is not required on every cycle and thus leaving alternate cycles for IO to the memories. This is beneficial in applications with high IO requirements since it allows the IO to occur without degrading core processor performance.

Scalability and Multiprocessing

The TigerSHARC®, like its predecessor, the SHARC, is designed for multiprocessing applications. The primary multiprocessing architecture supported is a cluster of up to eight TigerSHARC®s that share a common bus, a global memory and an interface to either a host processor or to other clusters. This is detailed in "Scalability and Multiprocessing" on page 1-9. In large multiprocessing systems this cluster can be considered as an element and connected in configurations such as torroid, mesh, tree, crossbar or others. The user can provide a personal interconnect method or use the on-chip communication ports.

The TigerSHARC® improves on most of the multiprocessing capabilities of the SHARC and enhances the data transfer bandwidth by a factor of four. These capabilities include:

- On-chip bus arbitration for glueless multiprocessing
- Globally accessible internal memory and registers
- Semaphore support
- Powerful, in-circuit multiprocessing emulation

System Level Enhancements

The TigerSHARC® includes several enhancements that simplify system development. The enhancements lie in three key areas:

- Architectural features supporting high-level languages and operating systems
- IEEE 1149.1 JTAG serial scan path and on-chip emulation features
- Support of IEEE floating-point formats

High Level Languages

The TigerSHARC® architecture has several features that directly support high-level language compilers and operating systems:

- Simple, orthogonal instruction allowing the compiler to efficiently use the multi-instruction slots
- General purpose data and IALU register files
- 32- and 40-bit floating point, as well as 8-, 16-, 32- and 64-bit integer native data types
- 32- and 64-bit native data types
- Large address space
- Immediate address modify fields
- Easily supported re-locatable code and data
- Fast save and restore of processor registers onto internal memory stacks

Serial Scan and Emulation Features

The TigerSHARC® supports the IEEE standard P1149.1 Joint Test Action Group (JTAG) standard for system test. This standard defines a method for serially scanning the IO status of each component in a system. The JTAG serial port is also used by the TigerSHARC® EZ-ICE to gain access to the processor's on-chip emulation features.

IEEE Formats

The TigerSHARC® is compatible with the IEEE single-precision floating-point data format in all respects, except for the following:

- The TigerSHARC® does not provide inexact flags.
- NAN inputs generate an invalid exception and return a quiet NAN.
- Denormal operands are flushed to zero when input to a computation unit and do not generate an underflow exception. Any denormal or underflow result from an arithmetic operation is flushed to zero and an underflow exception is generated.
- Round-to-nearest and round-towards-zero are supported. Round to +- infinity are not supported.

TigerSHARC® Core Architecture Blocks

The following sections summarize the features of the TigerSHARC® architecture. These features are described in greater detail in the following chapters.

Compute Blocks

The TigerSHARC® core processor contains two computation units known as compute blocks. Each compute block contains a register file and three independent computation units: an ALU, a multiplier, and a shifter. For meeting a wide variety of processing needs, the computation units process data in several fixed- and floating-point formats:

- *Fixed-point format*
  64 bits (long), 32 bits (word), 16 bits (short) and 8 bits (byte). For short fixed-point arithmetic, quad parallel operations on quad aligned data allow fast processing of array data. Byte operations are also supported for octal-aligned data. Refer to Figure 1-4 on page 1-13.

- *Floating-point format*
  Single floating-point and 40-bit floating-point operations are single or extended precision. The single floating-point format is the standard IEEE format, whereas the 40-bit extended-precision format occupies a double word (64 bits) and has eight additional LSBs of mantissa for greater accuracy.

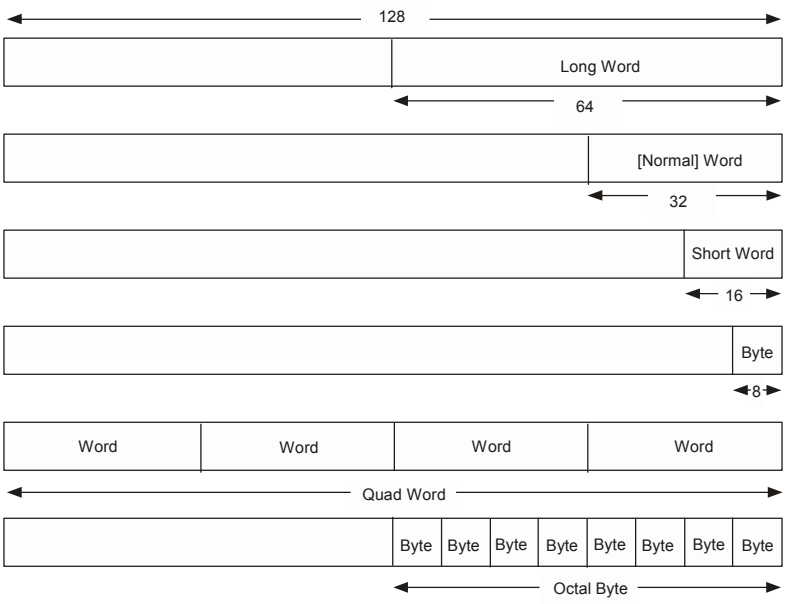

Figure 1-4. Word Format Definitions

ALU

The ALU performs a standard set of arithmetic and logic operations in both fixed-point and floating-point formats.

Multiplier

The multiplier performs floating-point and fixed-point multiplication as well as fixed-point multiply and accumulate.

Shifter

The shifter performs logical and arithmetic shifts, bit manipulation, field deposit and extraction.

Register File

A general-purpose, multi-port, 32-word data register file in each compute block is used for transferring data between the computation units and the data buses, and for storing intermediate results. All of these registers can be accessed as single, dual or quad aligned registers.

Execution Flow

The computation units perform single-cycle operations with a two cycle computation pipeline, meaning that results are available for use two cycles after the operation is begun (*see* Instruction Flow chapter of "TigerSHARC DSP Instruction Set Specification"). Hardware causes a stall if a result is not available in a given cycle (register dependency check).
Up to two computation instructions per compute block can be issued in each cycle, instructing the ALU, multiplier or shifter to perform independent, simultaneous operations.

IALUs

The IALUs provide memory addresses when data is transferred between memory and registers. Dual IALUs enable simultaneous addresses for multiple operand reads or writes. The IALUs allow computational operations to execute with maximum efficiency since the computation units can be devoted exclusively to processing data.

Each IALU has a multi-port, 32-word register file. Operations in the IALU are not pipelined. The IALUs support pre-modify with no update and post-modify with update address generation. Circular data buffers are implemented in hardware.

For indirect addressing, one of the registers in the register file can be modified by another register in the file or by an immediate 8- or 32-bit value, either before (pre-modify) or after (post-modify) the access. For circular buffer addressing, a length value can be associated with the first four registers to perform automatic modulo addressing for circular data buffers; the circular buffers can be located at arbitrary boundaries in memory. Circular buffers allow efficient implementation of delay lines and other data structures, commonly used in digital filters and Fourier transformations. The TigerSHARC® circular buffers automatically handle address pointer wraparounds, reducing overhead and simplifying implementation.

The IALUs can execute standard standalone ALU operations on IALU register files (without memory access).

Program Sequencer

The program sequencer supplies instruction addresses to memory, and together with the IALUs, allows computational operations to execute with maximum efficiency. It supports efficient branching using the branch target buffer (BTB), which reduces branch delays for conditional and unconditional instructions.

The TigerSHARC® has four general-purpose external interrupts, IRQ_{3-0}. The processor also has internally generated interrupts for the two timers, DMA channels, link ports, arithmetic exceptions, multiprocessor vector interrupts, and user-defined software interrupts. Interrupts can be nested through instruction commands, have a short latency and do not abort currently executing instructions. Interrupts vector directly to a user-supplied address in the interrupt table register file, removing the overhead of a second branch.

Internal Buses

The processor core has three buses, each one connected to one of the internal memories. These buses are 128 bits wide to allow up to four instructions, or four aligned data words, to be transferred in each cycle on each bus. On-chip system elements also use these buses to access memory. Only one access to each memory block is allowed in each cycle, so DMA or external port transfers must compete with core accesses on the same block. However, because of the large bandwidth available from each block, not all the memory bandwidth can be used by the core units, leaving some bandwidth available for use by DMA or bus interface other masters' transfers.

Quad Data Accesses

Each move instruction specifies whether a single*, dual† or quad word is accessed from each memory block. Two memory blocks can be accessed on each cycle because of the two IALUs. Dual word accesses can be used to supply two aligned words to one compute block or one aligned word to each compute block. Quad word accesses may be used to supply four aligned words to one compute block or two aligned words to each compute block—see "Long and Quad Data Access" on page 2-41. This is useful in applications that use real/imaginary data, or parallel data sets that can be aligned in memory—as are typically found in DSP applications. It is also used for fast save/restore of context during C calls or interrupts.

* A single word is defined as having 32 bits.

† A dual or double word is comprised of 64 bits.

Internal Memory

The TigerSHARC® contains three blocks of two Mbits each of on-chip, 128-bit wide SRAM.

Each memory block is organized as 64K words of 32 bits each. The accesses are pipelined to meet one clock cycle access time needed by the core, DMA or by the external bus. Each access can be of up to four words. Memories (and their associated buses) are a resource that must be shared between the compute blocks, the IALUs, the sequencer, the external port and the link ports. In general, if during a particular cycle more than one unit in the processor attempts to access the same memory, one of the competing units is granted access, while the other is held off for further arbitration until the following cycle—*see* "Bus Arbitration Protocol" in the *TigerSHARC® Hardware Specification*. This type of conflict only has a small impact on performance due to the very high bandwidth afforded by the internal buses.

An important benefit of large on-chip memory is that by managing the movement of data on- and off-chip with DMA, a system designer can realize high levels of determinism in execution time. Predictable and deterministic execution time is a central requirement in DSP and real time systems.

Programming Model

Instruction Set

The TigerSHARC® instruction set provides a wide variety of programming capabilities. The ability to execute up to four independent 32-bit instructions in each cycle greatly enhances performance.

The execution of up to four instructions in parallel enables the use of simultaneous computations with data transfers and branching or looping. These operations can be combined with few restrictions.

The IALU provides flexibility in moving data as single, dual, or quad words. Every instruction can execute with a throughput of one per cycle. IALU instructions execute with a single cycle of latency while computation units have two cycles of latency. Normally, there are no dependency delays between IALU instructions, but if there are, three or four cycles of latency can occur. The processor implements a static branch prediction mechanism: correctly predicted branches incur no overhead cycles if the branch target address is stored in the branch target buffer (BTB), otherwise the penalty is two cycles; incorrectly predicted branches incur a penalty of three to six cycles, depending on the type of conditional execution. See “Branch Unit Pipe” on page 3-16 for more details.

The TigerSHARC® assembly language is based on an algebraic syntax for ease of coding and readability. A comprehensive set of development tools supports program development.

Relative Addresses For Relocation

Most instructions in the TigerSHARC® support PC relative branches to allow code to be relocated easily. Also, most data references are “register relative”, which means they allow programs to access data blocks relative to a base register.

Conditional Execution

All instructions can be executed conditionally (a mechanism also known as predicated execution). The condition field exists in one instruction in an instruction line, and all the remaining instructions in that line either execute or not, depending on the outcome of the condition.

Internal Transfer

Most registers of the TigerSHARC® are classified as universal registers (Uregs). Instructions are provided for transferring data between any two Uregs, between a Ureg and memory, or for the immediate load of a Ureg. This includes control registers and status registers, as well as the data registers in the register files. These transfers occur with the same timing as internal memory load/store.

Context Switching

The TigerSHARC® provides the ability to save and restore up to eight registers per cycle onto a stack in two internal memory blocks when using load/store instructions. This fast save/restore capability permits efficient interrupts and fast context switching. It also allows the TigerSHARC® to dispense with on-chip PC stack or alternate registers for register files or status registers.

Nested Call and Interrupt

Nested call and interrupt return addresses (along with other registers as needed) are saved by specific instructions onto the on-chip memory stack , allowing more generality when used with high level languages.
Non-nested calls and interrupts do not need to save the return address in internal memory, making these more efficient for short, non-nested routines.

Branch Target Buffer

The TigerSHARC® achieves its fast execution rate by means of an eight-cycle pipeline.

The branch penalty in a deeply pipelined processor such as the TigerSHARC® can be compensated for by the use of a branch target buffer (BTB) and branch prediction. The branch target address is stored in the BTB. When the address of a jump instruction, which is predicted by the user to be taken in most cases, is recognized (the tag address), the corresponding jump address is read from the BTB and is used as the jump address on the next cycle. Thus the latency of a jump is reduced from three to six wasted cycles to zero wasted cycles. If this address is not stored in the BTB, the instruction must be fetched from memory.

Incorrectly predicted branches are expensive in terms of wasted cycles and it is best to use conditional instructions instead of branches whenever possible. All TigerSHARC® instructions are conditional as described in "Branch Unit Pipe" on page 3-16.

Other instructions also use the BTB to speed up these types of branches. These are interrupt return, call return and computed jump instructions.

In this Manual

This manual provides detailed information about the TigerSHARC® software in the following chapters:

- Introduction (*this chapter*)

  This chapter provides a general description of the architectural features, core architectural blocks, a programming model, a chapter listing, and listing of related documentation.

- Instruction Set

  This chapter describes the ADSP-TS001 (TigerSHARC®) instruction set in detail, starting with an overview of the instruction line and instruction types.

- Instruction Flow

  The chapter describes how the TigerSHARC® reads instructions instructions from memory into the instruction alignment buffer in quad words.

- Programmer's Quick Reference Guide

  This chapter contains a concise description of the TigerSHARC® programming model and assembly language. It is intended to be used as an assembly programming reference.

- Appendix A: Registers

  This appendix lists and describes the DSP's control and status registers, providing the address, bit definitions, and initialization values for each.

- Appendix B: Instruction Decode

  This appendix identifies operation codes (opcodes) for instructions. Use this chapter to learn how to construct opcodes.

Booting

The internal memory of the TigerSHARC® can be loaded from an 8-bit EPROM using a boot mechanism at system powerup. It can also boot using another master or through one of the link ports. Selection of the boot source is controlled by external pins.

Additional Literature

The following publications can be ordered from any Analog Devices sales office:

- *TigerSHARC® Data Sheet*
- *TigerSHARC® Hardware Specification*
- *TigerSHARC® Family Hardware & Software Development Tools Data Sheet*
- *TigerSHARC® Family Assembler Tools & Simulator Manual*
- *TigerSHARC® Family C Tools Manual*
- *TigerSHARC® Family C Runtime Library Manual*