ANALOG DEVICES

AN-403
APPLICATION NOTE

ONE TECHNOLOGY WAY • P.O. BOX 9106 • NORWOOD, MASSACHUSETTS 02062-9106 • 617/329-4700

Considerations for Selecting a DSP Processor—Why Buy the ADSP-21060? (The Analog Devices ADSP-21060 SHARC vs. Texas Instruments TMS320C40)

INTRODUCTION

Rich, powerful instruction sets, floating-point precision, and high speed execution make floating-point Digital Signal Processors (DSPs) a popular choice for designers of computational systems. Designers of systems ranging from medical imaging to graphical-transform-engines (arcade games) choose among floating-point DSPs using criteria such as feature integration, computational features, and I/O capabilities. This application note discusses the features of two popular floating-point DSP families, the ADSP-2106x and TMS320C4x. Table I shows a comparison for two of the DSPs. The ADSP-21060 SHARC DSP beats the TMS320C40 with the following features:

- *Computational Core*—Many more arithmetic functions available as instructions, more flexible multifunction instructions, more memory addressing options, and more program sequencing control options in the ADSP-21060's computational core provide many advantages over the TMS320C40.
- *I/O Capabilities*—Much higher data throughput, more flexible DMA support, and a wider variety of I/O ports provide the ADSP-21060 with I/O connectivity unmatched by the TMS320C40.
- *Memory*—Enormous on-chip RAM–the ADSP-21060 has 128K × 32 bit words of on-chip RAM (by comparison, the TMS320C40 has 2K × 32 bit words of on-chip RAM).

Table I. Comparison of Floating-Point DSP Features

DSP Processor	ADSP-21060-160	TMS320C40-80
Computational Core Features	25 ns Instruction Execution Time 120 Peak MFLOPS 32 Data Registers Support for 32 Circular Buffers	25 ns Instruction Execution Time 80 Peak MFLOPS 12 Data Registers Support for 1 (Length) Circular Buffer
I/O Capabilities	10 DMA Channels 6 Link Ports 2 Serial Ports 240 Mbytes/sec Maximum Throughput Supports "Glueless" Shared Memory Multiprocessing	6 DMA Channels 6 COMM Ports No Serial Ports 80 Mbytes/sec Maximum Throughput Requires "Glue" Logic to Share Memory with Other Processors
Memory	128K × 32 Bit Words of On-Chip RAM	2K × 32 Bit Words of On-Chip RAM
Benchmark	.46 ms for 1024-Point Complex FFT	.97 ms for 1024-Point Complex FFT

Note that all ADSP-21060 performance numbers and features listed in this note ALSO APPLY to the half memory (ADSP-21062, 64K × 32 bit) and 3.3 V low power (ADSP-21060L & ADSP-21062L) members of this family of DSPs.

COMPARING THE DSP PROCESSORS—ADSP-21060 AND TMS320C40

The reason for comparing DSP processors is to determine which provides the features you need to develop your DSP system and run your DSP applications. This section first discusses the value of different types of comparison topics, then provides detailed technical information on each processor, and finally summarizes the comparison data on the processors.

Digital signal processors are microprocessors (or microcomputers) optimized to perform numeric operations. The DSP microcomputers (microprocessor plus integrated memory and peripherals) compared in this note support sustained arithmetic operations with efficient access to a dual data memory space. Because retrieving multiple operands per instruction from memory and performing math operations with them is crucial for DSPs, this note compares the following *computational core* and *memory addressing* features:

- *General Math Features*
  Fixed/floating-point data format
  Result rounding
  Arithmetic result related interrupt
  Bit-wise operation
  Single instruction arithmetic operation
  Parallel operation (complete on a single instruction cycle)
  Context switching (background registers)
- *Direct & Indirect Memory Addressing*
  Address range
  Addressing methods
- *Program Sequencing*
  Program branching
  Subroutine calls
  Interrupts
  Pipeline delays (and their effects–delayed branching & interrupt latency)

DSP systems consist of one or more DSPs connected to peripherals, external memory (if needed), and (often) some type of host processor. To keep DSP computational units operating efficiently, a DSP's I/O circuitry should maintain a steady, high speed flow of data without hampering the DSP with overhead (instruction cycles "wasted" on non-numeric operations). In this note, the following *DSP I/O capabilities* are compared:

- *DMA Support*
  Number and types of channels supported and data throughput
- *Communications Ports*
  Number and types of channels supported and data throughput
- *Serial I/O Support*
  Number and types of channels supported and data throughput
- *Multiprocessor Support*
  Types of host and interprocessor communications supported and data throughput

The I/O most often performed in any DSP is between the computational core and memory. To complete this DSP comparison, the following *DSP memory* features are analyzed:

- *On-Chip Memory*
  Internal memory size, memory access speed, and data throughput
- *Instruction Cache*
  Type, size, and advantages/disadvantages
- *Off-Chip Memory Support*
  Address range and approximate access speeds supported
- *Shared Memory (Multiprocessor) Support*
  Memory sharing techniques supported and data throughput

In the Comparison Summary section, tables show a side-by-side comparison of each DSP feature discussed in the individual processor sections.

THE TEXAS INSTRUMENTS TMS320C40

The TMS320C4x DSPs are general purpose 32 bit, floating-point DSPs. Depending on the device version, TMS320C4x parts include 2K $\times$ 32 bit words of on-chip program/data RAM, 4K $\times$ 32 bit words of ROM, a 128 word Least-Recently-Used instruction cache, a 6 or 12 channel DMA coprocessor, up to six COMM (byte-wide) ports, two timers, and multiprocessor support. The two variations of the TMS320C4x are the TMS320C40 (40 MIPS, six COMM ports, 32 address lines) and TMS320C44 (25 MIPS, 4 COMM ports, 24 address lines). TMS320C4x DSPs use a double instruction rate clock input (i.e., 80 MHz for the TMS320C40 and 50 MHz for the TMS320C44). Figure 1 shows a block diagram of the TMS320C40 architecture.

TMS320C40, Computational Core

In Figure 1, the TMS320C40's computational core is represented by the multiplier, barrel shifter/ALU, extended precision registers, and other registers. This part of the DSP's architecture performs general math, memory addressing, and program sequencing.

Data Formats

The TMS320C40 supports fixed- and floating-point data formats. The DSP's computational core (multiplier & ALU) operate on 32-bit fixed-point and 40-bit floating-point inputs. The 40-bit TMS320C40 floating-point format does not comply with the IEEE 754/854 standard, but a conversion instruction (one cycle of overhead) is available.

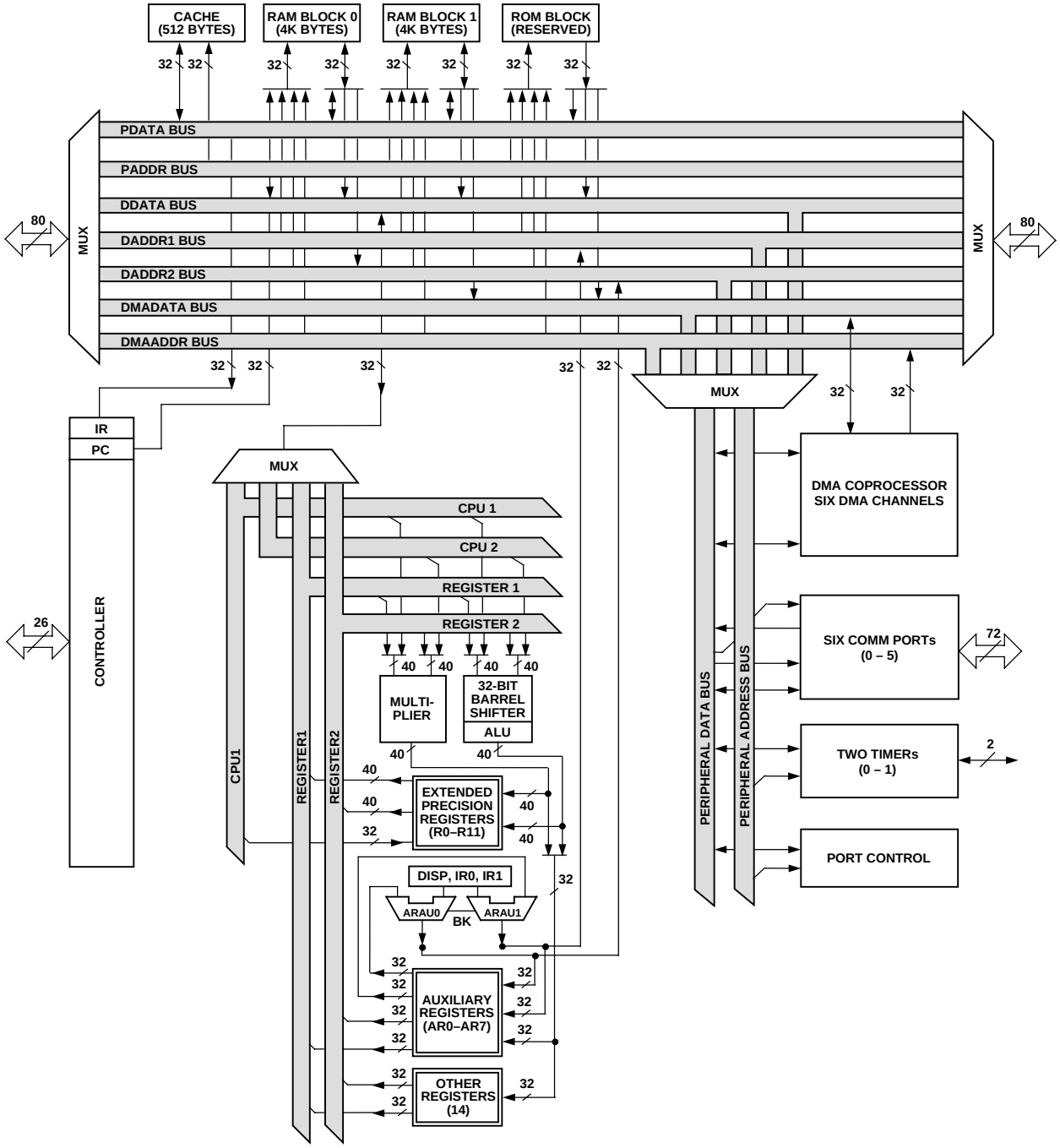

Figure 1. TMS320C40 Architecture Block Diagram

Result Rounding
A 32-bit fixed-point multiplication on the TMS320C40 yields a 32-bit output (from the 64-bit result). You can select either the upper or lower 32 bits to be written to the product's destination registers. The TMS320C40 supports result rounding with an instruction (RND) that takes an instruction cycle of overhead to complete.

Arithmetic Interrupts
To generate arithmetic result related interrupts (based on ALU/multiplier fixed- or floating-point overflow, floating-point underflow, or floating-point invalid operation), your TMS320C40 program must include branch instructions to trap these conditions.

Barrel Shifter Operation
The TMS320C40's barrel shifter provides support for left or right arithmetic, logical, and rotational shifts. With a bit mask stored in memory, the shifter can also perform bit test, set, toggle, and clear operations. Multiprecision shifts, however, are not supported.

Instruction Set
A DSP's architecture can reduce overhead through providing many single instructions for arithmetic operations. The TMS320C40's ALU instruction set satisfies basic requirements with support for addition, subtraction, absolute value, negate, and logic functions. The instruction set does not include some commonly required operations. The Processor Comparison Summary section discusses this issue in more detail, but Listing 1 provides one example of this lack of instructions.

Parallel Operation
As shown in Figure 1, the TMS320C40's computational core includes a register file (with twelve 32-bit or 40-bit registers) and computational units. This architecture supports parallel operations by letting the register file source two inputs to the multiplier while the two ALU inputs are fetched directly from memory or alternatively letting the inputs to the ALU be sourced from the register file while the multiplier inputs are fetched directly from memory. These multiple data paths to the ALU and multiplier make parallel operations (multiple operations that complete on a single instruction cycle) possible on the TMS320C40. The Processor Comparison Summary section compares the number and type of parallel operations supported by the TMS320C40 and ADSP-21060.

Memory Addressing
To perform multiplication/addition operations on each instruction cycle, a DSP's computational core needs new input data supplied at a corresponding rate. The TMS320C40 satisfies this requirement with two data accesses per instruction cycle using indirect memory addressing.

Direct addressing refers to memory accesses in which the data address is *directly* specified in the instruction. The TMS320C40's 32-bit instruction word lets the DSP directly address data only within 64K pages. The page base address is preloaded into the data page pointer (DP) register. When using direct addressing in your program, you must keep track of the data page pointer. This task adds another level of complexity to TMS320C40 software development. The TMS320C40 can perform either an ALU or multiply instruction (but not both) with a direct memory read within a 64K page unconditionally.

Indirect addressing refers to memory accesses in which the data address is *indirectly* specified in the instruction, an address generator specifies the address. The address generator uses index, modify, and length registers to generate an address based on the area of memory (index), pre- or post-modification of the location (to the next or previous location), and length of circular buffer (when needed).

The TMS320C40 supports indirect addressing with eight 32-bit, address registers (AR7-AR0). These eight registers can be used to address different areas of memory. Address registers can be pre- or post-modified by 0, 1, or by one of two preprogrammed modify registers (IR1, IR0). Also, address registers can be modified by an 8-bit offset.

Support for circular buffers is available on the TMS320C40, but there is only one length register (BK). If multiple circular buffers are used in one routine, they must all be the same length or the BK register must be modified for each circular data access (incurring lots of overhead). You must place circular buffers on the next power of two address boundary greater than the buffer length.

Program Sequencing
The last computational core feature in this description of the TMS320C40 is its support for program sequencing. A microprocessor's program sequencer is responsible for determining the flow of program execution. The program sequencing features valuable for a comparison of the TMS320C40 and ADSP-21060 are DO loops (repeated execution of a code block), branching (program execution jumps conditionally or unconditionally to a nonsequential address), interrupts, and pipeline delays.

```
LDF     R6,R7     ;     These three TMS320C40 instructions correspond
CMPF    R5,R7     ;     to following the ADSP-21060 maximum instruction—
LDFLT   R5,R7     ;     R7 = MAX(R5,R6)
```

Listing 1. TMS320C40 Instructions to Find the Maximum of Two Operands (vs. ADSP-21060).

You can implement DO loops in TMS320C40 programs using the block repeat and single repeat instructions. Two limitations on these structures are the lack of nesting and interrupt support. While you can nest DO loops on the TMS320C40, your program must save (and restore on return) the four registers controlling the repeat mode—incurring extra DSP overhead. The TMS320C40 single repeat instruction cannot be interrupted.

A TMS320C40 program can specify branch conditions (typically) based on the arithmetic results of a bit operation, a subtraction of two inputs, or an overflow. The TMS320C40 supports branching on floating-point overflows or underflows. Branch addresses can either be specified directly in the instruction word or can be indirectly specified through an index register. The TMS320C40 supports 24-bit addresses for unconditional branches, but is limited to a ±32K (16-bit) offset from the conditional branch address.

When the program sequencer branches execution to a subroutine (on a subroutine call) or interrupt, the return address is stored on the program counter (PC) stack. (Note that because the TMS320C40 does not support delayed subroutine calls/returns, all subroutines require eight cycles of overhead—four to call and four to return.)

The TMS320C40 responds to external interrupt inputs and has internal timers for generating periodic software interrupts. The TMS320C40 vectors to interrupts with no more than five cycle latency. Interrupts can be masked individually or can all be masked by one master bit (GIE). Because each interrupt only has one memory location, a four cycle, nondelayed branch to an interrupt subroutine must be placed at the interrupt vector address. Total interrupt latency is nine instruction cycles. Arithmetic and mode status must be manually saved when an interrupt occurs, incurring extra overhead cycles.

To optimize sequential program execution, the TMS320C40 uses a four level deep instruction pipeline. The pipeline influences execution branching by introducing a delay (three instruction cycles) as the pipe fills with instructions for the new branch. Your TMS320C40 program can use these cycles to execute instructions in the pipeline with the DSP's delayed branch feature. To take advantage of this feature, you must be able to place three instructions after the delayed branch to achieve efficient operation; this instruction placement is not always possible. Nondelayed branches require four cycles to execute on the TMS320C40.

Table II. TMS320C40, Computational Core Summary

Computational Core Feature		TMS320C40 Supports . . .
General Math Support		
Fixed/floating-point data format support	⇒	32-bit fixed-point, 40-bit (TI proprietary) floating-point; does not support IEEE 754/854 math
Result rounding support	⇒	Requires an extra instruction
Arithmetic result related interrupt support	⇒	Requires instructions trapping conditions
Barrel Shifter support	⇒	Left or right arithmetic, logical, and rotational shifts
Single instruction arithmetic operation support	⇒	Minimal (see Comparison Summary Section)
Parallel operation support	⇒	Minimal (see Comparison Summary Section)
Context switching (background registers) support	⇒	Not supported (no background registers)
Direct & Indirect Memory Addressing Support		
Address range support	⇒	Directly address data within 64K pages
Addressing methods supported	⇒	Direct and indirect addressing
Circular buffers supported	⇒	Does not support multiple length circular buffers without overhead
Program Sequencing		
DO loop support	⇒	No transparent nesting for DO loops
Program branching support	⇒	Branch to any 24-bit addresses on unconditional branches, but is limited to a ±32K (16-bit) offset from the conditional branch address
Subroutine calls	⇒	Nondelayed subroutine calls only (subroutine call & return have 8 cycles of overhead—minimum)
Interrupts	⇒	External input and internal timers, nine cycle latency
Pipeline delays	⇒	Four level pipeline, delayed branching supported

TMS320C40, I/O Capabilities

The DMA controller, COMM port, and external bus mux sections of the TMS320C40's architecture (Figure 1) provide the DSP with I/O access to external devices. This description of TMS320C40 I/O capabilities focuses on support for Direct Memory Accessing (DMA), communications I/O (external bus mux & COMM ports), and multiprocessor interfaces.

DMA lets the DSP (or external devices) access the DSP's memory and I/O ports without processor intervention. Because the DMA coprocessor and computational core cannot simultaneously access internal memory blocks, it is not always possible to achieve zero overhead DMA transfers with the TMS320C40. The TMS320C40's DMA controller has six (or twelve) DMA channels that can be configured to service either of the following configurations:

- Six 32-bit wide DMA channels to memory through the external bus mux (unified mode)
- Twelve 32-bit wide DMA channels to external processors or peripherals through COMM ports (split mode)

Communication Ports

The TMS320C40's external bus mux and six COMM ports provide interprocessor and peripheral communications. DMA channels through the external bus mux support data throughput of up to 80 Mbytes per second transferring data from external memory to the DSP and up to 53 Mbytes per second transferring data from the DSP to external memory. COMM ports each have an eight level deep by 32-bit wide FIFO buffer. The COMM ports support data throughput of up to 30 Mbytes per second.

Multiprocessor Support

The TMS320C40 provides multiprocessor system support with COMM port interprocessor communication (data flow multiprocessing), but requires external bus arbitration circuitry to support global memory sharing (cluster multiprocessing) using the external bus mux. Multiprocessing systems without shared memory can use the TMS320C40's COMM ports in split mode to DMA transfer up to 30 Mbytes per second for data flow multiprocessing. The TMS320C40 is not recommended for cluster multiprocessing applications because (even with external bus arbitration circuitry to support global memory) the DSP's architecture loads it with too much overhead for efficient global memory accesses in a cluster multiprocessing system.

Table III. TMS320C40, I/O Capabilities Summary

I/O Feature	TMS320C40 Supports . . .
Direct Memory Accessing (DMA)	
Number of DMA channels	⇒ 6 (unified mode) or 12 (split mode)
DMA channel configurations	⇒ Six channels through external bus mux or twelve channels through COMM ports
DMA data buffering	⇒ Eight level by 32-bit FIFO for COMM Port channels
Total DMA I/O throughput	⇒ 80 Mbytes per second from external to internal memory; 53 Mbytes per second from internal to external memory
Communication Ports	
Description of communication ports	⇒ Six 8-bit COMM ports
Total communications port throughput	⇒ 30 Mbytes per second
Serial Ports	
Description of serial ports	⇒ No serial ports
Multiprocessor interface	
Interprocessor communications support	⇒ Interprocessor communication through COMM ports
Shared global memory support	⇒ None—external bus arbitration circuitry required to support global memory through external bus mux

Table IV. TMS320C40, Memory Summary

Memory Feature	TMS320C40 Supports . . .
Internal memory size	⇒ 2K × 32 bits internal memory
Memory addressing	⇒ 32-bit address, 64K page size
Memory bus structure	⇒ Triple bussed between PDATA, DDATA, and DMADATA
Memory access (instruction cycles)	⇒ Internal memory accesses complete in one cycle for reads or writes ⇒ External memory reads complete in one cycle (unless followed by a write—then reads complete in two cycles) ⇒ External memory writes complete in two cycles
Instruction cache	⇒ Least-Recently-Used (LRU) instruction cache with entries for 128 instructions

TMS320C40, Memory

The RAM, ROM, and CACHE sections in the TMS320C40 architecture (Figure 1) represent the memory on the DSP. Three features of this internal memory make it useful to the floating-point DSP: size, access speed, and triple-bussing.

The internal memory on the TMS320C40 can hold only 2K × 32-bits of data or instructions. Using advanced programming techniques, the internal memory is large enough to hold only the most speed critical portions of your DSP algorithms (repeated looping sections and interrupt service routines). The time and effort spent fitting speed critical code into the small internal memory can add dramatically to the length and difficulty of software development projects. This internal memory is served by four address and three data busses (PDATA & PADDR, DDATA & DADDR1 & DADDR2, DMADATA & DMAADDR), allowing simultaneous access to instructions, data, and DMA data. This bus structure lets memory accesses to internal addresses by the TMS320C40 complete in one cycle for reads or writes. DSP external memory reads complete in one cycle (unless followed by a write, then they complete in two cycles) and external memory writes complete in two cycles.

To further reduce bus contention and streamline program execution, the TMS320C40 has a Least-Recently-Used (LRU) instruction cache with entries for 128 instructions (four blocks of 32 instructions). The DSP caches instructions to reduce the number of program and data memory accesses.

THE ANALOG DEVICES ADSP-21060 SHARC

The ADSP-2106x SHARC DSPs are a family of general purpose 32-bit, floating-point DSPs. They are based on, and are code compatible with, the Analog Devices ADSP-21000 Family. Depending on the device version, the ADSP-2106x DSP includes up to 128K × 32 bits of on-chip memory, a 32-word bus conflict averting instruction cache, a DMA controller, six (4-bit wide, double clock speed) Link ports, two serial ports, and multiprocessor support. The four variations of the ADSP-2106x are ADSP-21060 (128K × 32 bits of on-chip memory), ADSP-21060L (+3.3 V version of ADSP-21060), ADSP-21062 (64K × 32 bits of on-chip memory), and ADSP-21062L (+3.3 V version of ADSP-21062). The +5 V and +3.3 V versions of the DSP run at the same instruction clock speed, 40 MHz. Figure 2 shows a block diagram of the ADSP-2106x architecture.

Table V.

ADSP-2106x Variations All Use 40 MHz Clock)	Variations Between Parts
ADSP-21060	128K × 32 Bits of On-Chip Memory
ADSP-21060L	3.3 V, Low Power Version of ADSP-21060
ADSP-21062	64K × 32 Bits of On-Chip Memory
ADSP-21062L	3.3 V, Low Power Version of ADSP-21062

ADSP-21060, Computational Core

In Figure 2, the ADSP-21060's computational core is represented by the data register file, barrel shifter, ALU, multiplier, and other blocks within the core processor section. This part of the DSP's architecture performs general math, memory addressing, and program sequencing.

Data Formats

The ADSP-21060 supports fixed- and floating-point data formats. The DSP supports two floating-point data formats, single-precision 32-bit (24-bit mantissa and 8-bit exponent–IEEE 754/854 compliant) format, and extended-precision 40-bit (32-bit mantissa and 8-bit exponent) format.

Multiplication on the ADSP-21060 operates on 32-bit fixed-point, 32-bit floating-point, or 40-bit floating-point inputs. For 32-bit fixed-point multiplies, you can send the upper or lower 32 bits of the 64-bit product to either one of 16 data registers or add the product to (or subtract it from) one of two 80-bit, fixed-point accumulators. Also, you can individually treat fixed-point inputs as signed/unsigned and fractional/integer. Products of 32-bit or 40-bit floating-point multiplies must be sent to one of the sixteen 40-bit data registers.

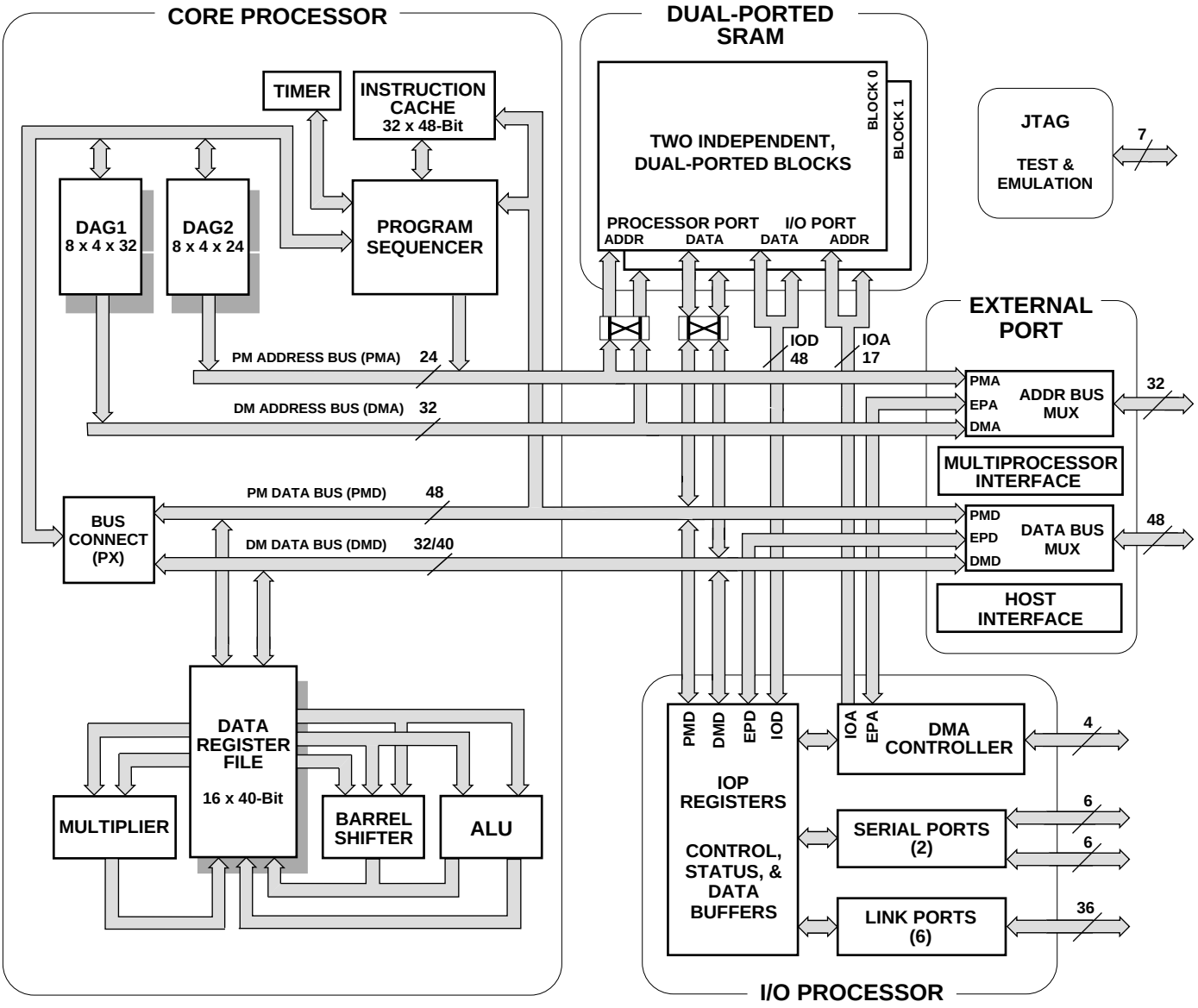

Figure 2. ADSP-21060 Architecture Block Diagram

Result Rounding
The ADSP-21060 supports result rounding with two IEEE rounding modes; round-to-nearest and round-toward-zero. Because one of the two modes is always enabled, ALU and multiplier results are automatically rounded to 32-bit or 40-bit floating-point numbers without additional overhead cycles.

Arithmetic Interrupts
The ADSP-21060's computational core can generate arithmetic result related interrupts based on ALU/multiplier fixed- or floating-point overflow, floating-point underflow, or floating-point invalid operation.

Barrel Shifter Operations
The barrel shifter in the ADSP-21060 provides extensive shifting support for left or right arithmetic, logical, and rotational shifts. The shift amount can originate from a register or can be specified in the instruction word. Shift results can be logically OR'd with other data registers. To support multiprecision shifting, the DSP supports bit-wise operations including bit test, set, toggle, and clear. Also, the shifter can extract arbitrary bit fields from one word then deposit the result anywhere inside another word, extract the exponent from a fixed-point input, and count leading ones or zeros in a fixed-point input.

Instruction Set
A DSP's architecture can reduce overhead by providing many single instructions for arithmetic operations. The ADSP-21060 instruction set includes many nontraditional processor functions in addition to those needed for the basic requirement of addition, subtraction, absolute value, negate, and logic functions. The Processor Comparison Summary section discusses this issue in more detail, but Listing 2 provides one example of the advantage these additional instructions provide.

A block diagram of the ADSP-21060 register file and computational units is shown in the lower left quarter of Figure 2. The register file provides local storage for arithmetic input data and results.

The file consists of sixteen 40-bit registers. Within one instruction, all registers can be swapped with a set of secondary registers (background registers). The register file's secondary registers are typically used during interrupts or subroutine calls to eliminate time consuming memory transfers of register file data. Use of the 16 secondary registers provides a total of 32 register file registers. The ADSP-21060's large data register file lets you efficiently implement register intensive algorithms, such as a Radix-4 FFT.

Parallel Operations
In one instruction cycle, the DSP can perform two memory transfers, source two inputs to the ALU, source two inputs to the multiplier, and store ALU and multiplier results. The two memory transfers can be reads, writes, or a read and a write. These multiple data paths to the ALU and multiplier make parallel operations (multiple operations that complete on a single instruction cycle) possible on the ADSP-21060. The Processor Comparison Summary section compares the number and type of parallel operations supported by the ADSP-21060 and TMS320C40.

Memory Addressing
To perform multiplication/addition operations on each instruction cycle, a DSP's computational core needs new input data supplied at a corresponding rate. The ADSP-21060 satisfies this requirement with two data accesses per instruction cycle using indirect memory addressing.

Direct addressing refers to memory accesses in which the data address is *directly* specified in the instruction. The ADSP-21060's 48-bit instruction word lets the DSP directly address the full 24-bit program memory or 32-bit data memory in one instruction cycle with a single instruction. Any of the DSP's general purpose or data registers can be loaded from memory or stored into memory. The ADSP-21060 can also perform a single cycle ALU/multiplier operation and direct memory read or write within a 64 location offset conditionally. Any of the 16 index registers can be used as base addresses for this offset.

Indirect addressing refers to memory accesses in which the data address is *indirectly* specified in the instruction, an address generator specifies the address. The address generator uses index, modify, and length registers to generate an address based on the area of memory (index), pre- or post-modification of the location (to the next or previous location), and length of circular buffer (when needed).

The ADSP-2106x has two Data Address Generators (DAGs). One DAG is used to access data from data memory, the other lets you access data in program memory. Each DAG contains eight index registers and eight modify registers. Each index register can be used to address a different area in memory. Using indirect addressing, you can pre- or post-update the designated index register with any of the eight modify registers in the same DAG. The index register can also be modified by a 6-bit immediate offset.

```
R7 = MAX(R5,R6)    ;          This ADSP-21060 instruction for maximum corresponds to
                   ;          the following three TMS320C40 instructions—
                   ;          LDF    R6,R7  ;
                   ;          CMPF   R5,R7  ;
                   ;          LDFLT  R5,R7  ;
```

Listing 2. ADSP-21060 Instruction to Find the Maximum of Two Operands (vs. TMS320C40)

Each of the DSP's index registers has a corresponding base address and length register. The base address and length registers let you confine the index register value within a range of data addresses. Each time your program modifies the index register, the DAG transparently tests the resultant address. If the address is out of range, the DAG corrects the index register with the modulo address. This feature is referred to as modulo or circular data addressing. Because the DAGs contain 16 sets of index, base, and length registers, you can place up to 16 arbitrary length circular buffers anywhere in memory. And, with modulo addressing, maintaining an index within each buffer is automatic.

Both DAG's eight index, base, length, and modify registers have associated secondary registers (background registers) configured in two groups of four. During an interrupt or subroutine call, one or both of the DAG's register groups can be swapped in a single cycle with secondary registers saving the overhead of many individual register to memory transfers for a context save. The secondary DAG registers make available an additional 16 circular buffers for a total of 32 circular buffers.

Program Sequencing

The last computational core feature in this description of the ADSP-21060 is its support for program sequencing. A microprocessor's program sequencer is responsible for determining the flow of program execution. The program sequencing features valuable for a comparison of the ADSP-21060 and TMS320C40 are DO loops (repeated execution of a code block), branching (program execution jumps conditionally or unconditionally to a nonsequential address), interrupts, and pipeline delays.

You can use the ADSP-21060 DO UNTIL instruction for easy-to-code DO loop programming. This instruction supports efficient software loops without the overhead of additional instructions to branch, test a condition, or decrement a counter. DO UNTIL loops also provide zero overhead, six level deep loop nesting and respond to interrupts from any loop nesting level.

An ADSP-21060 program can specify branch conditions (typically) based on the arithmetic results of a bit operation, a comparison of two inputs, or an overflow. Additionally, this DSP can branch program execution based on any of the following conditions:

- Downcounter status
- Logic input status on any of four programmable input pins
- ALU status
- Multiplier status
- Shifter status
- System bit test status

The ADSP-21060 also has complex conditional instructions supporting execution of an arithmetic operation and a data move based on a single condition. The ADSP-21060 lets you specify branch addresses within the full 24-bit program address space in any branch instruction. When the program sequencer branches execution to a subroutine (on a subroutine call) or interrupt, the return address is stored on the program counter (PC) stack. An ADSP-21060 subroutine call is a single-cycle, single-operand instruction and uses a direct or indirect address.

To optimize sequential program execution, the ADSP-21060 uses a three level deep instruction pipeline. The pipeline influences execution branching by introducing a delay (two instruction cycles) as the pipe fills with instructions for the new branch. Using the DSP's delayed branch feature, you can hold off subroutine calls while the DSP executes the two subsequent instructions in the instruction pipeline. (You only need to place two instructions after a delayed branch to achieve efficient operation.) This delay eliminates the DSP overhead (two cycles while pipeline "empties") related to branching execution. Nondelayed branches require three cycles to execute on the ADSP-21060.

Note that on the ADSP-21060 all execution branches (JMP, CALL, RTS, or RTI) also can occur in parallel with a computation. This parallel execution along with the use of a delayed branch can completely eliminate all overhead related to branching. If properly coded, a program's use of a subroutine call and return, in many cases, incurs no branching overhead.

The ADSP-21060 responds to external interrupt inputs and has an internal timer for generating periodic software interrupts. The ADSP-21060 responds (vectors) to an interrupt with no more than four cycles of latency. An interrupt mask register allows interrupts to be individually enabled by setting bits. The DSP has a global interrupt mask bit, IRPTEN, that lets you mask out all interrupts. Because each interrupt has four reserved memory locations, you can code short interrupt service routines within the vector table—providing a fast response. You also can code a delayed branch within the vector table; an option that can provide no-overhead interrupts.

Interrupt support on the ADSP-21060 includes a five level deep status stack. When an interrupt occurs, the DSP automatically pushes the arithmetic, mode, and interrupt mask status on to the status stack. (Status can be manually pushed onto this stack for subroutines.) The ADSP-21060 also supports a mode that lets you nest interrupts. This mode is useful when higher priority interrupts must remain unmasked during execution of lower priority interrupt routines.

Table VI. ADSP-21060, Computational Core Summary

Computational Core Feature	ADSP-21060 Supports
General Math Support	
Fixed-/floating-point data format support	⇒ 32-bit fixed-point, 32-bit IEEE 754/854 standard floating-point, & 40-bit floating-point
Result rounding support	⇒ Automatic IEEE rounding modes; round-to-nearest and round-toward-zero
Arithmetic result related interrupt support	⇒ Interrupts on ALU and multiplier status available
Barrel shifter support	⇒ Extensive shifting support Left arithmetic shifts Right arithmetic shifts Logical shifts Rotational shifts
	⇒ Bit-wise operation support for multiprecision shifting Bit test, set, toggle, and clear Logically OR'ing shift result with other registers Shift amount from a register or instruction word Extract arbitrary bit fields and place result Extract exponent from a fixed-point input Count leading 1s or 0s in a fixed-point input
Single instruction arithmetic operation support	⇒ Exceptional (see Comparison Summary Section)
Parallel operation support	⇒ Exceptional (see Comparison Summary Section)
Context switching (background registers) support	⇒ Supported with register file and DAG secondary (background) registers
Direct & Indirect Memory Addressing Support	
Address range support	⇒ Directly address data throughout 32-bit address space
Addressing methods supported	⇒ Direct and indirect addressing
Circular buffers supported	⇒ Supports 32 arbitrary length circular buffers for zero overhead looping
Program Sequencing	
DO loop support	⇒ Transparent six level deep nesting of zero overhead DO loops
Program branching support	⇒ Branch to any 24-bit address on branches
Subroutine calls	⇒ Single-cycle, single-operand instruction using direct or indirect address for delayed or nondelayed subroutine calls
Interrupts	⇒ External input and internal timers, four cycle latency
Pipeline delays	⇒ Three level pipeline and delayed branch support make it easier to achieve zero overhead branching
	⇒ Compute or memory transfer allowed in parallel with branching operation

ADSP-21060, I/O Capabilities

The I/O processor and external port sections of the ADSP-21060's architecture (Figure 2) provide the DSP with I/O access to external devices. This description of ADSP-21060 I/O capabilities focuses on support for Direct Memory Accessing (DMA), communications I/O (external and Link ports), serial I/O (SPORTs), and multiprocessor interfaces.

DMA lets the DSP (or external devices) access the DSP's memory and I/O ports without processor intervention (overhead). The ADSP-21060's I/O processor has ten DMA channels that can be configured to service any combination of the following ports:

- Four 16-, 32-, or 48-bit wide DMA channels to memory through the external port
- Six 32- or 48-bit wide DMA channels to external processors or peripherals through Link ports
- Four serial DMA channels to external peripherals through SPORTs

Communication Ports

The ADSP-21060's external port and six Link ports provide interprocessor and peripheral communications. DMA channels through the external port each have a six level deep by 48-bit wide FIFO buffer. The external port supports data throughput of up to 240 Mbytes per second.

Link ports operate at double the DSP's internal clock speed and each have a two level deep by 48-bit wide FIFO buffer. These ports automatically pack (4-bit transfers into 32-bit or 48-bit words) and unpack (32-bit or 48-bit words into 4-bit transfers) data and instruction words. The Link ports support data throughput of up to 40 Mbytes per second each. Together, the six Link ports support total data throughput of 240 Mbytes per second.

Serial Ports

Serial I/O *(*through SPORTs on the ADSP-21060) provides interface support for a wide variety of peripheral devices including many industry standard data converters, codecs, and other processors. The SPORTs' configurable features include the following:

- Fully independent transmit and receive sections
- μ-Law and A-Law companding of data (channel selectable in TDM mode)
- Word lengths from 3 to 32 bits
- Big or little endian bit order format
- Right or left word justification and zero fill
- Internal or external clocking and synchronization
- Time Division Multiplexed (TDM) mode (supported in multichannel mode)

Multiprocessor Support

The ADSP-21060 provides multiprocessor system support with "glueless" external memory sharing (cluster multiprocessing) using the external port and Link port interprocessor communication (data flow multiprocessing). Built-in bus arbitration circuitry and bus master protocol support through the external port simplify design of multiprocessing systems that make use of shared memory (internal & external). Also, host interface support in the external port lets the DSP interface with any 16- or 32-bit host processor. Link ports, with their flexible configuration features and high throughput, are ideal for multiprocessing applications using multiple ADSP-21060s (or in combination with other processors) without shared memory.

Table VII. ADSP-21060, I/O Capabilities Summary

I/O Feature	ADSP-21060 Supports . . .
Direct Memory Accessing (DMA)	
Number of DMA channels	⇒ 10
DMA channel configurations	⇒ Any combination of six channels through external port, six Link ports, and two SPORTs (serial ports)
DMA data buffering	⇒ Corresponding combination of FIFOs and channels: Six level by 48-bit FIFO for external port channels Two level by 32-bit FIFO for Link port channels Two level by 32-bit FIFO for SPORT channels
Total DMA I/O throughput	⇒ 240 Mbytes per second
Communication Ports	
Description of communication ports	⇒ Six 4-bit Link ports operating at twice the DSP clock
Total communications port throughput	⇒ 40 Mbytes per second
Serial Ports	
Description of serial ports	⇒ Two serial ports (SPORTs) that provide built-in support for companding, configurable word size/ data format, and (in multichannel mode) Time Division Multiplexed (TDM) operation
Total serial port throughput	⇒ 20 Mbytes per second for four ports (2 RX & 2 TX)
Multiprocessor interface	
Shared global memory support	⇒ Unified multiprocessor memory map; up to six ADSP-21060s can directly access each others internal RAM and I/O registers at up to 240 Mbytes per second ⇒ "Glueless" shared memory through bus mastering and bus arbitration through external port
Interprocessor communications support	⇒ Interprocessor communication through Link ports

ADSP-21060, Memory

The dual-ported SRAM section in the ADSP-21060 architecture (Figure 2) represents the memory on the DSP. Three features of this internal memory make the ADSP-21060 unique among floating-point DSPs: size, dual-porting, and triple-bussing.

The enormous SRAM (4 Mbit on ADSP-21060) can hold 128K x 32 bits of DM data or 80K x 48 bits of PM instructions. This internal memory is dual-ported; simultaneously available to the DSP's computational core and I/O processor. This dual-ported architecture lets the I/O processor operate without incurring overhead on the computational core. Also, the memory is served by three address and data busses (PMA & PMD, DMA & DMD, IOA & IOD), allowing simultaneous access to instructions, data, and I/O data. This bus structure lets the ADSP-21060 complete memory accesses to internal or external addresses in only one cycle for reads or writes.

To further reduce bus contention and streamline program execution, the ADSP-21060 has a two way, set associative instruction cache with entries for 32 instructions. The DSP only caches instructions that conflict with program memory data accesses, making the cache much more efficient than a cache that has to store every instruction.

ADSP-21060 VS. TMS320C40—COMPARISON SUMMARY

This section provides a brief summary of the individual processor sections then provides a side-by-side comparison of ADSP-21060 and TMS320C40 benchmarks, instruction set, and hardware features not covered in the processor sections.

The previous two sections (one on each processor) describe the computational core, I/O capabilities, and memory features of the ADSP-21060 and TMS320C40. While these processors are both general purpose floating-point DSPs with DMA interfaces and internal memory, they differ radically in their support for floating-point operations, DMA throughput and configurations, and internal memory size and throughput. Table IX summarizes the most crucial DSP features that differentiate the ADSP-21060 and TMS320C40, but to get a complete comparison of the way these processors differ, you should refer to the subsection summary tables in the individual processor sections.

Table VIII. ADSP-21060, Memory Summary

Memory Feature	ADSP-21060 Supports . . .
Internal memory size	⇒ 128K × 32 bits internal memory
Memory addressing	⇒ 32-bit address, unified multiprocessor memory space
Memory bus structure	⇒ Dual-ported between core and I/O, triple bussed between PM, DM, and IOM
Memory access (instruction cycles)	⇒ Internal or external memory accesses complete in one cycle for reads or writes
Instruction cache	⇒ Two-way, set-associative instruction cache with entries for 32 instructions
Memory Accessibility	⇒ Unified memory map lets up to six ADSP-21060 DSPs access each others internal memory as if it were their own; no external arbitration hardware or processor overhead required
External memory address decoding	⇒ Four internally decoded chip select lines let you use slower, lower cost external SRAMs with the ADSP-21060

Table IX. ADSP-21060 vs. TMS320C40, "Key" Features Comparison Summary

DSP "Key" Features	ADSP-21060-160	TMS320C40-80
Computational core performance	120 MFLOPS	80 MFLOPS
Internal RAM size	128K × 32 bits	2K × 32 bits
Internal RAM I/O overhead	No overhead, eliminated by dual-ported RAM	Overhead possible if memory bus contention occurs
Shared global memory support	⇒ Unified multiprocessor memory map; up to six ADSP-21060s can directly access each others internal RAM and I/O registers at up to 240 Mbytes per second ⇒ "Glueless" shared memory with bus mastering & bus arbitration through external port	⇒ None, external bus arbitration circuitry required to support global memory through external bus mux
DMA data throughput		
Off-chip to on-chip transfers	⇒ 240 Mbytes per second	⇒ 80 Mbytes per second
On-chip to off-chip transfers	⇒ 240 Mbytes per second	⇒ 53 Mbytes per second
Communications port transfers	⇒ 40 Mbytes per second (Link)	⇒ 30 Mbytes per second (COMM)
Serial port transfers	⇒ 20 Mbytes per second	⇒ No Serial Ports

Benchmarks drawn from commonly used algorithms can provide one good side-by-side comparison of DSPs. All digital signal processors have the ability to execute FIR filters at one instruction cycle per filter tap. DSP algorithms (such as filters) using general purpose features and instructions demonstrate the performance differences between DSPs on "real-world" operations.

One typical example benchmark algorithm for DSPs is the Fast Fourier Transform (FFT). Listings 3 and 4 show a comparison of ADSP-21060 and TMS320C40 assembly code for the core of their corresponding (recommended) Radix-2 FFT algorithms. (Note that the ADSP-21060's FFT core loop (FFT butterfly) executes in four instruction cycles and the TMS320C40's executes in eight.)

```
/* The following ADSP-21000 instructions correspond to the fft equations:                    */
/* (From ADSP-21000 Family Applications Handbook 1, Pages 208 & 223)                         */
/*    X0' = X0 + (CX1 + SY1)                                                                 */
/*    X1' = X0 – (CX1 + SY1)                                                                 */
/*    Y0' = Y0 + (CY1 – SX1)                                                                 */
/*    Y1' = Y0 – (CY1 – SX1)                                                                 */

R8=R1*R6,     R14=R11-R14,     DM(I2,M0)=R10,     R9=PM(I11,M8);
R11=R1*R7,    R3=R9+R14,       R9=R9-R14,         DM(I2,M0)=R13,      R7=PM(I8,M8);
R14=R0*R6,    R12=R8+R12,      R8=DM(I0,M0),      PM(I10,M10)=R9;
R12=R0*R7,    R13=R8+R12,      R10=R8-R12,        R6=DM(I0,M0),       PM(I10,M10)=R3;
```

Listing 3. ADSP-2106x Radix-2 Complex DIT FFT Butterfly (4 Instructions)

```
*   AR'=AR+(CBR+SBI)              The following TMS320C40 instructions make
*   AI'=AI-(SBR-CBI)              up the core butterfly of the complex,
*   BR'=AR-CBR+SBI)               radix-2 DIT FFT from TMS320C4x User's
*   BI'=AI+(SBR-CBI)              Guide, page 12-113
*
      mpyf   *+ar1,r6,r5          ;  r5 = BI * SIN , (AR' = r5)
||    stf    r5,*ar2++
      subf   r1,r0,r2             ;  (r2 = TI = r0 - r1)
      mpyf   *ar1,r7,r0           ;  r0 = BR * COS , (r3 = AI + TI)
||    addf   r2,*ar0,r3
      subf   r2,*ar0++,r4         ;  (r4 = AI - TI , BI' = r3)
||    stf    r3,*ar3++
      addf   r0,r5,r3             ;  r3 = TR = r0 + r5
      mpyf   *ar1++,r6,r0         ;  r0 = BR * SIN , r2 = AR - TR
||    subf   r3,*ar0,r2
      mpyf   *ar1++,r7,r1         ;  r1 = BI * COS , (AI' = r4)
||    stf    r4,*ar2++
      addf   *ar0++,r3,r5         ;  r5 = AR + TR, BR' = r2
||    stf    r2,*ar3++
```

Listing 4. TMS320C40 Radix-2 Complex DIT FFT Butterfly (8 Instructions)

The key to the ADSP-21060's assembly code efficiency (i.e., four instructions for the FFT instead of eight) lies in the DSP's ability to simultaneously generate the sum and difference of the same two inputs, perform a multiply, store one number in memory, and fetch another from memory, all in a single cycle. The TMS320C40 cannot store data to memory while performing a multiply and add. Also, the TMS320C40 cannot simultaneously add and subtract the same two inputs. Table X lists algorithm benchmarks comparing the ADSP-21060 and TMS320C40.

Table X. Common Benchmarks for DSP Performance: ADSP-21060 vs. TMS320C40

DSP Benchmarks	ADSP-21060-160 (at 25 ns instruction cycle)	TMS320C40-80 (at 25 ns instruction cycle)
1024-point complex, FFT	0.46 ms	0.97 ms
Divide, 32-bit floating-point	150 ns, 6 cycles	225 ns, 9 cycles
Whetstone/ms (compiled C code)	53,094	37,323
Dhrystone/s (compiled C code)	82,987	48,780

Tables XI, XII, XIII and XIV provide some additional side-by-side comparisons of the DSP's architecture and performance features. These tables use the following conventions:

A **check**	✓	indicates that the DSP has the feature or supports the function
A **zero**	Ø	indicates that the DSP does not provide the feature or function
A **number**	32-bits	indicates the type of support the DSP provides

Table XI. DSP Math Operations of the ADSP-21060-160 vs. TMS320C40-80

Digital Signal Processor Math Operations Supported . . .		ADSP-21060	TMS320C40
Multiplier includes:	32-bit fixed-point input	✓	✓
	Unsigned fixed-point input	✓	✓
	40-bit floating-point results	✓	✓
	40-bit floating-point inputs	✓	✓
	Fractional fixed-point input	✓	Ø
	32-bit floating-point inputs	✓	Ø
ALU supports:	40-bit floating-point	✓	✓
	32-bit fixed-point	✓	✓
	Multiprecision addition & subtraction	✓	✓
	Seed 1/x, seed $1/\sqrt{x}$	✓	✓
	Minimum, maximum, average, clip, & scale	✓	Ø
	Absolute value of (x+y) & (x-y)	✓	Ø
	Simultaneous (x+y) & (x-y)	✓	Ø
	8-bits accumulated compare status	✓	Ø
Barrel shifter supports:	Logical & arithmetic Shift	✓	✓
	Rotate	✓	✓
	Bit-wise test, clear, toggle, & set	✓	✓
	Shift & logical OR with register (extended precision shifts)	✓	Ø
	Field extract & deposit	✓	Ø
	Exponent extract	✓	Ø
	Count leading ones & zeros	✓	Ø
Multifunction (simultaneous) Operations:	Fixed- or floating-point multiply & add	✓	✓
	Fixed- or floating-point multiply & subtract	✓	✓
	Fixed- or floating-point multiply, add, & subtract	✓	Ø
	Floating-point multiply & convert (fixed-to-floating-point)	✓	Ø
	Floating-point multiply & convert (floating-to-fixed-point)	✓	Ø
	Floating-point multiply & find average	✓	Ø
	Floating-point multiply & find absolute value	✓	Ø
	Floating-point multiply & find maximum	✓	Ø
	Floating-point multiply & find minimum	✓	Ø

Legend: A check (✓) indicates the DSP has the feature, a zero (**Ø**) indicates the DSP does not have the feature, and a number quantifies the feature (if necessary).

Table XII. ALU Instructions of the ADSP-21060-160 vs. TMS320C40-80

'060 Instruction	Description	ADSP-21060	TMS320C40
ABS X	Determines the absolute value of fixed- or floating-point operand.	✓	✓
-X	Negates the fixed- or floating-point operand by twos complement.	✓	✓
COMP(X,Y)	Compares the fixed-point fields or floating-point fields in X with Y.	✓	✓
RECIPS X	Creates a seed for 1/X, the reciprocal of floating-point operand X.	✓	✓
RSQRTS X	Creates a seed for $1\sqrt{X}$ the reciprocal square root of floating-point operand X.	✓	✓
FLOAT Y	Converts the fixed-point operand to a floating-point operand.	✓	✓
FLOAT X BY Y	Adds the fixed-point scaling factor Y to the fixed-point X operand and converts X to a floating-point result.	✓	Ø
FIX X	Converts the floating-point operand to a twos-complement, 32-bit fixed-point integer.	✓	✓
FIX X BY Y	Adds the fixed-point scaling factor Y to the floating-point X operand and converts X to a twos-complement, 32-bit fixed-point integer.	✓	Ø
(X+Y)/2	Adds the fixed- or floating-point operands and divides the result by two.	✓	Ø
MIN(X,Y)	Finds the smaller of the fixed- or floating-point operands in X and Y.	✓	Ø
MAX(X,Y)	Finds the larger of the fixed- or floating-point operands in X and Y.	✓	Ø
CLIP X BY Y	Returns the ABS of the fixed- or floating-point operand in X if ABS X is less than the ABS Y. Returns either ABS Y (if X is positive) or –ABS Y (if X is negative) if ABS X is greater than the ABS Y.	✓	Ø
X+Y,X-Y	Adds or subtracts the fixed- or floating-point X and Y operands.	✓	Ø
SCALE X BY Y	Returns the exponent of the floating-point X operand (scaled by adding the fixed-point Y operand) as a scaled floating-point value.	✓	Ø
MANT X	Extracts the mantissa from the floating-point X operand.	✓	Ø
LOGB X	Returns the exponent of the floating-point X operand as an unbiased, twos-complement, fixed-point integer.	✓	Ø
COPY X SIGN TO Y	Copies the sign of the floating-point X operand to floating-point operand Y.	✓	Ø
ABS (X+Y)	Adds the floating-point operands and returns the absolute value of the normalized result.	✓	Ø
ABS (X-Y)	Subtracts the floating-point operands and returns the absolute value of the normalized result.	✓	Ø

Legend: A check (✓) indicates the DSP has the feature, a zero (**Ø**) indicates the DSP does not have the feature, and a number quantifies the feature (if necessary).

Table XIII. Program Sequencing Features of the ADSP-21060-160 vs. TMS320C40-80

Digital Signal Processor Program Sequencing Supports...	ADSP-21060	TMS320C4*0*
Instruction clock	**40 MHz (1x)**	**80 MHz (2x)**
⇒ The 'C40 requires an 80 MHz (2x instruction rate) clock input. This signal generates much more system RF noise than the '060's 40 MHz (1x) clock input.		
Instruction pipeline levels	**3**	**4**
⇒ This is the number of overhead cycles for a nondelayed branch, and implies the number of instructions you must place in pipeline after a delayed branch for efficient operation; '060-two instructions & 'C40-three instructions.		
Memory locations at interrupt vector	**4**	**1**
⇒ This number indicates type of programming you can place in your interrupt vector table. The four instruction locations available at each '060 vector implies that many types of interrupt service routines can be coded right in the interrupt vector table for faster response and less execution branching. The single instruction location at each 'C40 vector implies that almost all interrupt service routines require branching.		
Zero overhead loop nesting levels	**6**	**1**
Instruction cache	✓	✓
Single-cycle/operand branching with 24-bit address	✓	✓
Delayed branch	✓	✓
Conditional branch with loop abort	✓	✓
Delayed branch returns	✓	**Ø**
Delayed branch subroutine calls	✓	**Ø**
Pre-modified indirect branching	✓	**Ø**
Compute in parallel with conditional branch (JUMP, CALL, RTS, or RTI)	✓	**Ø**
Data transfer in parallel with branch (JUMP)	✓	**Ø**
Interrupts on index/address register modulo overflow	✓	**Ø**
Interrupts on arithmetic status	✓	**Ø**
Status stack	✓	**Ø**

Legend*:* A check (✓) indicates the DSP has the feature, a zero (**Ø**) indicates the DSP does not have the feature, and a number quantifies the feature (if necessary).

Table IV. Data Addressing Features of the ADSP-21060-160 vs. TMS320C40-80

Digital Signal Processor Memory Addressing Supports . . .	ADSP-21060	TMS320C40
Number of index registers	**16**	**8**
Number of modify registers	**16**	**2**
Number of length registers	**16**	**1**
Number of base address registers	**16**	**Ø**
Secondary index, modify, length, base registers	**Full Set**	**Ø**
Direct addressing (read or write)	**32-bit**	**16-bit**
Direct read, one compute ('060 = conditional or unconditional execution) ('C40 = two operand compute only)	**6-bit**	**16-bit**
Compute, modify by immediate offset ('C40 = two operand compute only)	**6-bit**	**8-bit**
Direct read, two computes ('060 = conditional or unconditional execution)	**6-bit**	**Ø**
Direct write, one compute ('060 = conditional or unconditional execution)	**6-bit**	**Ø**
Direct write, two computes ('060 = conditional or unconditional execution)	**6-bit**	**Ø**
Bit-reversed addressing	✓	✓
Single cycle—two indirect writes, no compute ('C40 = internal RAM access only)	✓	✓
Single cycle—two indirect writes, one compute ('C40 = internal RAM access only)	✓	**Ø**
Single cycle—two indirect writes, two computes ('C40 = internal RAM access only)	✓	**Ø**
Single cycle—two indirect reads, two computes ('C40 = but not if followed by an external memory write)	✓	✓
Single cycle—indirect read & write, one compute ('C40 = internal RAM access only)	✓	✓
Single cycle—indirect read & write, two computes ('C40 = internal RAM access only)	✓	**Ø**

Legend: A check (✓) indicates the DSP has the feature, a zero (**Ø**) indicates the DSP does not have the feature, and a number quantifies the feature (if necessary).

CONCLUSION

For high performance computational systems, designers must look beyond instruction execution speed to compare DSPs. You must look at computational efficiency, I/O capabilities and data throughput, system cost, and level of integration. The Analog Devices' ADSP-21060 and ADSP-21062 SHARC processors offer uncompromising performance, I/O peripheral interface, and function integration.

REFERENCES

The following sources contributed information to this application note:

Buyer's Guide to DSP Processors, Berkeley Design Technology, Inc., February, 1994.

TMS320C4x User's Guide, Texas Instruments, (Revision C, 2564090-9761, 1993).

ADSP-2106x SHARC User's Manual, Analog Devices, Inc ., (1st Ed., 82-000795-02, 1995).

PRINTED IN U.S.A.

E2038–8–8/95