ANALOG DEVICES

LOW-COST SIMD SHARC

Considerations For Selecting a DSP Processor – Why Buy The ADSP-21161?

The Analog Devices ADSP-21161 SIMD SHARC vs. Texas Instruments TMS320C6711 and TMS320C6712

Author : K. Srinivas

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 Signal processing 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 power and IO capabilities. This application note discusses the features of two popular floating point DSP processors – The ADSP 21161 compared with TMS320C6711 and TMS320C6712. Both the TMS320C6711 and TMS3206712 share the same TMS320C67xx floating point DSP core and a very similar set of onchip peripheral support. Hence both the processors are considered in this application note. Table 1 shows the comparison for two of the DSPs. The ADSP21161 offers the following features over the TMS320C6711/TMS32C6712 processors:

- Compute blocks – Two compute blocks PEx and PEy are available which work on different data values but both execute the same instruction thus improving on the code density. This is called SIMD mode of execution. The compute blocks consist of an adder, multiplier, barrel shifter and a dedicated register file to load and store the data values. Each compute block provides parallel (multi function) instructions. The combination of SIMD and multi function instructions generates as much as 600 Megaflops of peak compute power from ADSP21161.

- Data Addressing – ADSP21161 has two Data Address Generators (DAGs) to generate addresses for fetching data and/or programs on DM and PM buses. These DAGs execute instructions in parallel to the compute instructions. The DAGs support a variety of addressing modes and circular buffering.

- Memory – Enormous RAM is available on chip in ADSP21161 processor. There is 128 K bytes of RAM available in this processor. This is equal to 21.3K of ADSP-21161 instructions. The onchip memory is dual ported.

- I/O capabilities – Very flexible and on-intrusive DMA support that enables parallel transfers between any of 4 serial ports, 2 link ports and the external port. There are a total of 14 DMA channels are supported in ADSP21161. This allows for the connection of a number of peripherals to the DSP processor and ensures data availability.

Table I. Comparison of Floating point DSP features

DSP Processor -> Features	ADSP 21161	TMS320 C6712	TMS320 C6711
Computational core features	• 10 ns Instruction Execution Time (core Clock frequency of 100 MHz) • Peak MFLOPS = 600 • All instructions are single cycle execute. • Branch latency is 2 cycles (could be overcome with delayed branch) • 96 Universal registers (32, 40-bit data reg, 16 Index reg, 16 Modifier reg, 16 Length reg, 16 Base reg) • Support for 32 circular buffers. • Support for hardware loops. • Packed instruction execution from 8-, 16-, 32- and 48-bit wide memories	• 10 ns Instruction execution times (core clock executes at 100 MHz) • Peak MFLOPS = 600 MFLOPS • All floating-point instructions are multi-cycle execute. • Branch latency is 5 cycles (could over come with delayed branch) • 32 general purpose registers (includes data registers, address registers, all 32 bits) • Support for only 8 circular buffers. • No hardware loop support.	• 10 ns & 6.66 ns Instruction execution times (Runs at 100 MHz and 150 MHz) • Peak MFLOPS = 600 MFLOPS /900 MFLOPS • All floating-point instructions are multi-cycle execute. • Branch latency is 5 cycles (could over come with delayed branch) • 32 general purpose registers (includes data registers, address registers, all 32 bits) • Support for only 8 circular buffers. • No hardware loop support.
I/O Capabilities	• 4 Serial ports, 2 Link ports, SPI port, External port (32 bits or configurable to 48 bits if link ports are not used) can be used with 14 DMA channels. • All the DMAs are non-intrusive. Core can access internal memory in parallel to a DMA/IOP access. Always 1 64-bit word data can be written to memory at no extra cost of core cycles. • Full 48-bit-wide data bus when link ports are not used to enable 100 MHz, single cycle instruction execution from external SDRAM. • Dedicated SPI (Serial Peripheral Interface) port. • 400 Mbytes/sec maximum throughput through external memory interface.	• No Host port, 1 16-bit external memory interface, 2 Multichannel buffered serial ports can be used with 16 DMA channels. • All the DMA's are intrusive. They accomplish transfers through cycle stealing of core cycles. C6712 DMA can transfer data to L2 memory at 1 word per cycle only if there is no L1P miss and no L1D miss on the same data bank. If there is an L1P miss, the DMA stalls. • Support SPI interface • 200 Mbytes/sec maximum throughput through external memory interface.	• 1 Host port, 1 32-bit external memory interface, 2 Multichannel buffered serial ports can be used with 16 DMA channels. • All the DMA's are intrusive. They accomplish transfers through cycle stealing of core cycles. C6712 DMA can transfer data to L2 memory at 1 word per cycle only if there is no L1P miss and no L1D miss on the same data bank. If there is an L1P miss, the DMA stalls. • Support SPI interface • 400 Mbytes/sec maximum throughput through external memory interface.

	• Supports "Glueless" shared memory multiprocessing • When using 64 bit DMA packing modes, a maximum throughput of 800 Mbytes can be accomplished by the DMA.	• Requires "Gluelogic" for shared memory with other processors. • If the DMA can access L2 memory every alternate cycle on an average, without the core losing any cycles (may be ideal condition), the through put is still 400 Mbytes/sec.	• Requires "Gluelogic" for shared memory with other processors. • If the DMA can access L2 memory every alternate cycle on an average, without the core losing any cycles (may be ideal condition), the through put is still 400 Mbytes/sec.
Memory	• 1 Mega bit on chip RAM. ⇨ 128 K bytes on chip RAM • Dual ported on chip memory with unified address space. • Can store up to 21K 21161 instructions.	• ½ M bit on chip RAM. ⇨ 72K bytes of onchip memory (2 level memories increase access times) • Only 4K program memory and 4K data cache support parallel access (dual port). 64K of L2 RAM does not support dual port. • Can store up to 18K C6711 instructions	• ½ M bit on chip RAM. ⇨ 72K bytes of onchip memory (2 level memories increase access times) • Only 4K program memory and 4K data cache support parallel access (dual port). 64K of L2 RAM does not support dual port. • Can store upto 18K C6712 instructions.
Programmability	• Supports SIMD programming model that is suited for many types of applications. ⇨ Reduced code size. • Easy to write code in assembly as all instructions complete in 1 cycle.	• VLIW architecture requires instructions explicitly to program various compute units in the processor. This results in larger code sizes. • It is not easy to write code in assembly because instructions take multiple cycles to complete. Hence, user requires scheduling the instructions explicitly which is a difficult task.	• VLIW architecture requires instructions explicitly to program various compute units in the processor. This results in larger code sizes. • It is not easy to write code in assembly because instructions take multiple cycles to complete. Hence, user requires scheduling the instructions explicitly which is a difficult task.

Comparing the DSP Processors – ADSP 21161 & TMS320C6711/TMS320C6712

The reason for comparing the two DSP processors is to determine which DSP provides the features an application developer needs to develop a DSP system and run 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 access to 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 and Indirect Memory Addressing***
  Address range
  Addressing methods

- ***Program Sequencing***
  Program branching
  Subroutine calls
  Interrupts
  Pipeline delays (and their effects-delayed branching and 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 through put.

- ***Communications Ports***
  Type of external interfaces and channels to support communication with moderate speed peripherals.

- ***Serial I/O Support***
  Number and types of channels supported and data throughput.

- ***Multiprocessor Support***
  Types of host and interprocessor communication 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/dis-advantages

- ***Off-Chip Memory Support***
  Address range and approximate access speeds supported

- ***Shared Memory (Multiprocessor) Support***
  Memory sharing techniques supported and data through put.

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 TMS320C6711/TMS320C6712 DSP Processors

The TMS320C6711 and TMS320C6712 DSP processors from Texas Instruments belong to the same TMS320C6x floating point DSP family. They consist of the same DSP core (with varying clock frequencies), a very similar set of I/O interfaces (with very few differences). Hence the sections below will describe the architecture of both the devices. Where there are differences between both the devices, they are highlighted clearly.

The TMS320C6711/TMS320C6712 are general purpose 32-bit, floating point DSPs. Each DSP processor has a total onchip memory of 72K bytes. The processors also have 16 channel DMA controller that support two multi-channel buffered serial ports, 16-bit host port interface and external memory interface.

The TMS320C6711 DSP processor executes with a clock frequency of either 100 MHz or 150 MHz.
The TMS320C6712 DSP processor executes with a clock frequency of 100MHz.

Figure 1 below shows a block diagram of the TMS320C6711 processor architecture.
Figure 2 below shows a block diagram of the TMS320C6712 processor architecture.

TMS320C6711/C6712 Computational Core
The TMS320C6711/TMS320C6712 DSP computational core consists of 2 sets of four execution units named L1, M1, S1, D1 (in the first set) and L2, M2, S2, D2 (in the second set). These units together accomplish data addressing/address generation, multiplications, arithmetic, logical and branch operations. Each unit is supported by 16 32-bit registers. These registers are common to all the units described above. Two of these registers can be combined as pair registers for performing extended precision arithmetic.

Data Formats
The TMS320C6711/TMS32C6712 supports operation on fixed and floating point data formats. Application developer needs to use the register pair to achieve extended precision (more than 32-bits) . IEEE floating point formats are supported.

Result Rounding
The TMS320C6711/TMS32C6712 processor supports rounding to nearest number, round towards 0, round up and round down modes. This needs to be configured in a configuration register. There is however no explicit *round* instruction if the user wishes to perform a round on a floating point operand.

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), the TMS320C6711/ TMS32C6712 must do a condition testing on the FADCR/FAUCR/FMCR registers. It may then execute a branch to the code segment that does error handling of those conditions.

Barrel Shifter Operation

The L, M and S units support left shift or right shift - arithmetic, logical and rotational shift operations. 32-bit and 40-bit shifts are supported. Bit extract, Bit Set and Bit Clear instructions are available bit field operations.

Figure : 1. TMS320C6711 Block Diagram

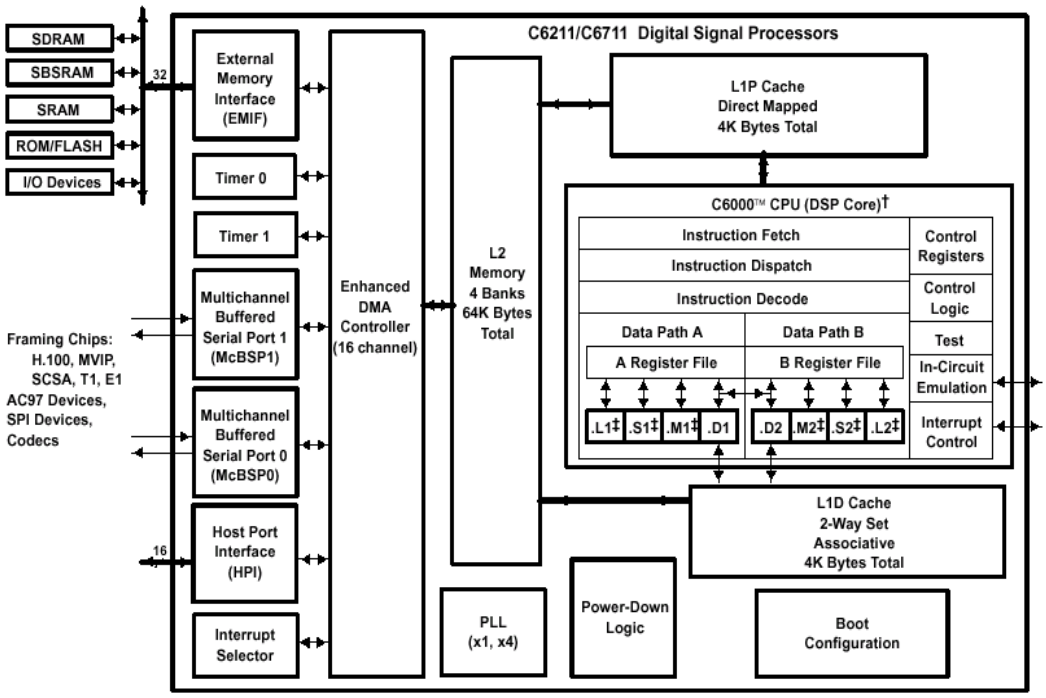
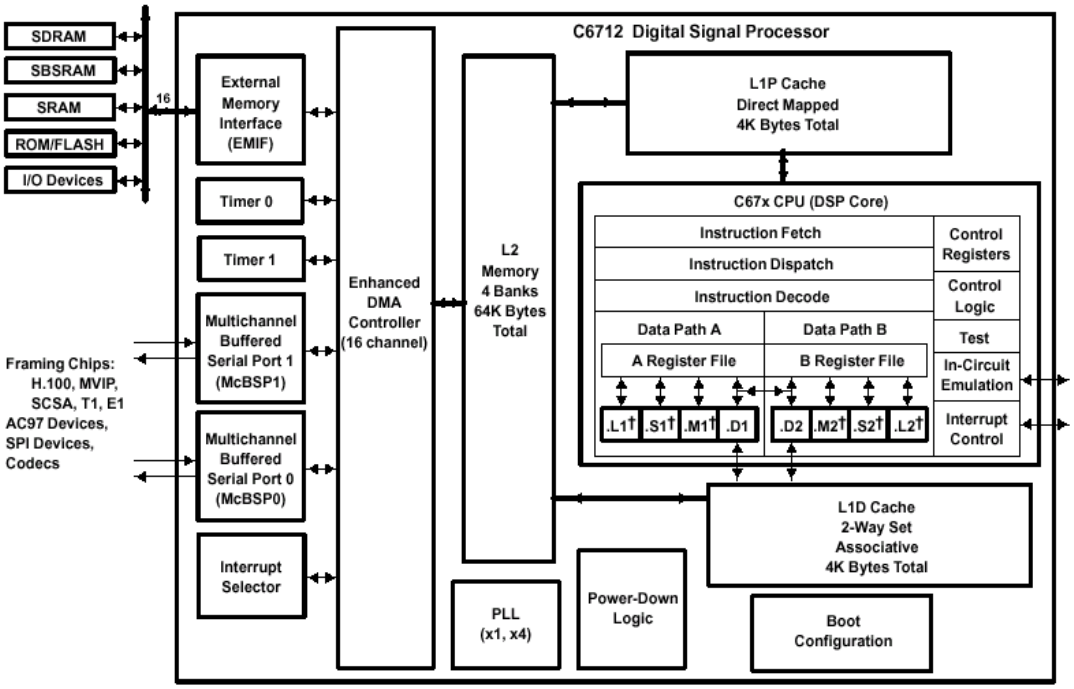


Figure : 2. TMS320C6712 Block Diagram



Note the differences between TMS320C6711 and TMS320C6712 in the host port interface and external memory interfaces. There is no host port interface in TMS320C6712 while the external memory interface in TMS320C6712 is 16 bits while it is 32 bits in TMS320C6711 processor.

Instruction Set

A DSP's architecture can reduce overhead providing many single instructions for arithmetic operations. The TMS320C6711/TMS32C6712 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. Consider the example given below :

Listing 1. Computing Average of two integers

TMS320C6711/ TMS32C6712	ADSP-21160 (SISD Mode)
LDW .D1 *A10, A1 (Load the value 2. Possible delays in fetching from memory !!!) || ADD .L1 A2, A3, A4 SHL .S1 A4, 1, A4	F2=(F0+F1)/2 – One single instruction.

The above listing illustrated a simple example to compute average of two numbers. It requires atleast 3 instructions in the case of TMS320C6711/12 processors while it takes one single instruction to accomplish the same in ADSP-21161 processor. Similar is the case with instructions like MAX, MIN etc. Cache misses will also add to the instruction execution times in the case of TMS320C6711/12 processors.

Parallel Operation

As shown in Figure 1, the TMS320C6711/TMS32C6712's computational core includes two register files, one per set of execution units, each consisting of 16 32-bit registers. They again could be considered as 8 40/64-bit register pairs. The L, S, M and D units in each compute block can access the register files of the corresponding compute block. However, there are two cross data paths that allow a compute block to access the register file of the other compute block. The architecture supports parallel operations by letting register file source the required inputs to the L, S, M and D units simultaneously. These multiple data paths enable all the execution units to execute in parallel and complete the

instruction execution. However due to deeply pipelined architecture model of TMS320C6711/TMS32C6712 processor, many floating point instructions don't complete execution in a single cycle. Hence the user has to take care of the instruction execution delay and schedule subsequent instructions. The processor comparison summary section compares the number and type of parallel operations supported by the TMS320C6711/TMS32C6712 and ADSP-21161 processors.

Memory Addressing

To sustain a throughput of one multiplication and/or addition (key computational element in most DSP algorithms) on each instruction cycle, a DSP's computational core needs new input data supplied at corresponding rate. TMS320C6711/TMS32C6712 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 an instruction. The TMS320C6711/TMS32C6712 DSP processor does not support any direct addressing mode instruction. Hence the user always has to keep track of the register file and get a free register to accomplish a memory load through indirect addressing. In summary, the application developer needs to track the register resources and use a free register resource for memory loads and stores. In the absence of such a register, the user needs to save a register temporarily on stack and use that for memory access.

Indirect addressing refers to memory access in which the data address is indirectly specified in an instruction, an address generator specifies the address. The address generator supports two types of addressing modes – linear addressing mode and circular addressing mode.

In Linear Addressing Mode, the *Register Indirect* addressing mode is used to access memory using only the registers. In the case of *Register Relative* addressing mode, a 5 bit register offset field is specified. This is used in place of another register used for offset. The *Register Relative with 15 bit constant value* addressing mode is used to access 64K memory ranges. All of the above addressing modes can be used with any of *No register modifications / Post Modify / Pre Modify* operations. The TMS320C6711/12 assembly program requires the use of one of the data registers (described above) to specify the instruction address and modifier values. This means that users will have lesser number of registers for compute operations. This will increase the complexity of programs to perform intelligent register allocation in their programs. At times users may have to store certain registers in memory due to unavailability of registers. This increases the number of execution cycles of the program.
Support for circular buffering is available in TMS320C6711/TMS32C6712 processor, but there are only two registers to specify the block sizes of the circular buffer. These are specified in BK1 and BK2 fields in AMR register. Hence, the user cannot specify the different circular buffers of different sizes, but will have to use one of the sizes specified in either BK1 or BK2. Also, the user can set the circular buffers only at fixed step values (power of two). This requires the user needing to place the circular buffer on the next power of two address boundary greater than the buffer length. This turn results in wastage of data memory space.

Program Sequencing

The last computational core feature in this description of the TMS320C6711/TMS32C6712 is its support of program sequencing. A microprocessor's program sequencer is responsible for determining the flow of the program execution. The program sequencing functions used for a comparison of the TMS320C6711/TMS32C6712 and ADSP-21161 are

- Do loops (repeated execution of a code block)
- Branching (program execution jumps/calls conditionally or unconditionally to a non-sequential address)
- Interrupts
- Pipe line latencies

There are no instructions to implement DO loops in TMS320C6711/TMS32C6712 processors. Hence there is no hardware support to accomplish loops as in ADSP-21161 processor. Hence, if the user needs to implement the loops in TMS320C6711/TMS32C6712 processor, he needs to use a branch instruction to branch back to the beginning of the loop. Software must decrement the loop variable and perform condition checking for branching to the beginning of the loop. This in turn results in more requirement of the CPU MIPS. Since there is no hardware loop support in the TMS320C6711/TMS32C6712 processors, there is no hardware support for nested loops as well.

A TMS320C6711/TMS32C6712 program can specify branch conditions based on 5 condition registers. User software should be written to capture the arithmetic results of the bit operations, overflows, underflows, carry etc in the five condition registers (A1 A2, B0, B1, B2) and then use these condition registers for performing branches. Branch addresses can either be specified directly in the instruction word or can be indirectly specified through an index register. In the case where branch address is specified directly in the instruction itself, TMS320C6711/TMS32C6712 supports 23 bit addresses for unconditional and conditional branches. You can do branching on an index register by using the S2 execution unit of the TMS320C6711/TMS32C6712 processor.

There is no support to save the return address when a branch is executed (similar to a call instruction). User will have to save the return address in a software stack (defined by the user) and use that address (through indirect branching) to return back to the called address.

The TMS320C6711/TMS32C6712 responds to external interrupts. The processor responds to the interrupt and vectors to the interrupt service routine with an interrupt latency of atleast 15 cycles. User must write software to branch to a different location if the entire ISR cannot fit in the interrupt vector address.

The TMS320C6711/TMS32C6712 processor uses a 16 stage pipe line in an effort to increase the speed of different execution units within the core. This influence the branching by introducing a delay (5 cycles) as the pipeline fills with instructions for the new branch. However using the delayed branch feature, software could utilize the branch delay slots and schedule instructions. However it is not always possible to schedule the instructions in these delay slots. Also, considering the fact that many TMS320C6711/TMS32C6712 floating point instructions are not single cycle, user needs to identify when an instruction could be scheduled and when computed data values are available in the registers and then schedule the instructions. This complicates the programming model when doing assembly programming.

Table II. TMS320C6711/TMS32C6712, Computational Core Summary

Computational Core Feature	TMS320C6711/TMS32C6712 Supports
Direct & Indirect Memory Addressing Support	
Address range support	⇨ Addresses data using 32-bit addresses.
Addressing methods supported	⇨ Indirect addressing. (With register offset and constant offset)
Circular buffers supported	⇨ Supported, but with only 2 buffer sizes.
Context switching (background registers) support	⇨ Not supported. (No background registers)
Bit reversed addressing	⇨ Not supported implicitly within loads/stores. However an instruction is available to perform bit reversal. This would mean extra cycles in execution.
Program Sequencing	
Do loop support	⇨ No hardware loops are supported. User needs to use branches with delay slot scheduling.
Program branching support	⇨ Support for 23-bit direct branches or register based indirect branching.
Subroutine calls	⇨ Not available. Use branch instructions.
Interrupts	⇨ 16 cycle delay. External and internal interrupts available.
Pipeline delays	⇨ 16 deep pipeline. Requires scheduling instructions for good throughput.

TMS320C6711/TMS32C6712, I/O capabilities

The TMS320C6711 and TMS320C6712 have a similar set of I/O features. There are however two differences between the two processors.

- There is no host port interface for TMS320C6712 DSP processor

- The external memory interface for TMS320C6712 DSP processor is only 16-bits against 32 bits in TMS320C6711 processor.

Except for the above differences, all the I/O features described below are the same for both the DSP processors.

The enhanced DMA controller, external memory interface, two multichannel buffered serial ports, a 16-bit host port interface provide the DSP with I/O access to external devices. This description of TMS320C6711/TMS32C6712 capabilities focuses on support for Direct Memory Access (DMA), communications I/O (Serial ports, external memory interface, host port interface) and multiprocessor interfaces.

DMA lets the DSP (or external devices) access the DSP's memory and I/O ports without processor's intervention. Because the DMA co-processor and computational core cannot simultaneously access internal memory blocks in the TMS320C6711/TMS32C6712, it is not always possible to achieve zero overhead DMA transfers with theTMS320C6711/TMS32C6712. The TMS320C6711/TMS32C6712's EDMA controller has sixteen DMA channels that can be configured to serve either of the following configurations:

- External memory DMA interface
- Multichannel serial ports (upto 128 channels supported)
- Timer Interrupt based DMA channels

The EDMA controller in TMS320C6711/TMS32C6712 processor uses a RAM to store the DMA parameters of all the channels. It also supports two dimensional DMA transfers and DMA chaining. The EDMA has the ability to interrupt the core at the end of DMA transfer.

Serial Ports

There are two multi channel buffered serial ports. It provides full duplex communication with independent framing and clocking for receive and transmit. It has direct interface to industry standard devices (SPI devices, AC97 devices etc.). Each serial port can support upto 128 channels. They also support A-law and u-law companding capabilities. It also provides interface to I2S devices. Supports data sizes of 8, 12, 16, 20, 24 and 32 bits.

External Memory Interface

TMS320C6711 External Memory Interface

The TMS320C6711 32-bit external memory interface supports glueless interface to external devices like SB-SRAM, SDRAM, Asynchronous SRAM, ROM, an external shared memory device. It supports interface up to 4 banks of 64M bit SDRAM. The TMS320C6711 also supports 8-bit, 16-bit and 32-bit external interfaces in asynchronous mode. However appropriate packing has to be done by the software.

TMS320C6712 External Memory Interface

The TMS320C6712 16-bit external memory interface supports glueless interface to external devices like SB-SRAM, SDRAM, Asynchronous SRAM, ROM, an external shared memory device. It supports interface up to 4 banks of 64M bit SDRAM. Appropriate packing has to be done by the software to pack the 16-bit instruction/data fetched from external memory into 32-bits.

Impact of 16-bit external memory interface

Due to the 16-bit external memory interface, all fetches (both instruction and data) require double the number of clock cycles and hence the double the time compared to a 32-bit external interface. This is also compounded by the fact that TMS320C67xx processors have poor code densities since they are VLIW processors. This would require a lot of code and data to be placed in external memory. Hence the core will be starved of data periodically where there is external memory access.

Host Port Interface

TMS320C6711 Host Port Interface

The 16-bit host port interface of the TMS320C6711 processor allows host access of the internal memory of the device. The host interface provides direct access to memory mapped peripherals of the TMS320C6711 processor. One DMA channel is dedicated for host port interface.

TMS320C6712 Host Port Interface
There is no host port interface existing in TMS320C6712 processor. Hence a TMS320C6712 system need to be a stand alone system. If the host needs to be integrated with TMS320C6712 based system, one of the serial ports may have to be used thus reducing the number of serial devices that can be integrated with the system. In such cases, additional software and hardware need to be implemented. This also results in slow host interaction with the DSP and a lot of additional software overhead on the DSP to handle host communications.

Multiprocessor Support
TMS320C6711
The TMS320C6711 provides multiprocessor system support using the Host Port Interface, but requires external bus arbitration circuitry to support global memory sharing (cluster multiprocessing) using the external bus. Multiprocessing systems without shared memory can use the TMS320C6711's host port interface to access the internal memory of the device. The device is not recommended for cluster multiprocessing applications because (even with external bus arbitration circuitry to support global memory) the DSP's architecture loads it too much with overhead for efficient global memory accesses in cluster multiprocessing system. This interface will allow integration with host processor only. There is no seamless support for DSP to DSP integration for this device. Hence system designers require to design bus arbitration logic. There is also no support for shared global memory for this device.

TMS320C6712
Since there is no host port interface in the TMS320C6712 processor, it is not suitable for multiprocessing system.

Table III. TMS320C6711/TMS320C6712, I/O Capabilities Summary

I/O Feature	TMS320C6711/TMS320C6712 Supports …
Direct Memory Accessing (DMA)	
Number of DMA channels	⇨ 16 DMA channels (intrusive with core. Executes through cycle stealing)
DMA channel configurations	⇨ 4 external interface channels, 4 serial port channels, 1 host port channel, 2 timer based general DMA channels, 4 channels for chaining based on completion of certain DMA channels, 1 SDRAM based channel. (ADSP-21161 DMA is very simple to program)
Total DMA I/O throughput	⇨ 200M bytes per second in TMS320C6712 ⇨ 400M bytes per second in TMS320C6711
Communications Ports	
Description of communication ports	⇨ 2 muti-channel buffered serial ports supported.
Total communications port throughput	⇨ 25M bytes per second through serial ports.
Multiprocessor interface	
Interprocessor communications support.	⇨ Host processor communications through host port in TMS320C6711
No seamless support for DSP to DSP integration.	⇨ No Inter processor communications possible in TMS320C6712
Shared global memory support	⇨ None. External bus arbitration circuitry required to support global memory access through external bus mux.

TMS320C6711, Memory
The L1P Program Cache, L1D Data Cache, L2 Unified Memory RAM/Cache sections in the TMS320C6711/TMS320C6712 architecture represent the internal memory of the DSP. The L1P and L1D memories can be accessed simultaneously and independently for simultaneous progam and data access.

The internal memory on the TMS320C6711/TMS320C6712 can hold only 4K bytes of L1D cache, 4K bytes of L1P cache, 64K bytes of L2 cache. However the code densities that can be achieved for various applications on this processor will be less. This can be proved by considering a simple example of an instruction that we wish to perform a floating point mac, add, subtract, 2 loads with address pointer updates. It requires atleast six 32-bit instructions to accomplish the above tasks. Compiler inefficiencies also should be considered while computing the code densities. It is very difficult to write assembly code in this processor and one needs to rely a lot on the compiler.

The L2 memory can be configured as RAM or cache. L1D and L1P can operate only as cache. However, access to L2 memory from EDMA can happen only when L2 memory is configured as RAM. A cache miss from L1P takes 5 cycles while a cache miss from L1D takes 4 cycles to fetch from L2 cache.

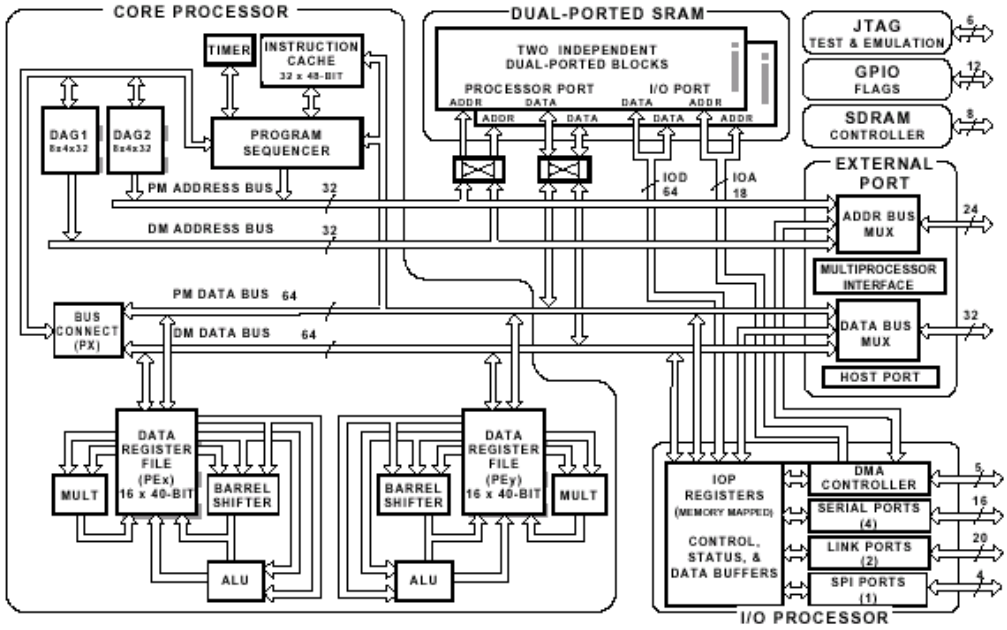
Table IV. TMS320C6711/TMS320C6712, Memory Summary

Memory Feature	TMS320C6711/TMS320C6712 Supports …
Internal Memory Size	⇨ 18K x 32 bits internal memory
Memory Addressing	⇨ 32 bit address
Memory bus architecture	⇨ Independent program and data buses to L1P and L1D memories. ⇨ Single 256 bit bus to L2 memory and this could result in core stalls upon conflict.
Memory Access (instruction cycles)	⇨ L1D Internal memory accesses complete in one cycle for reads and writes. ⇨ External memory access depends on the data availability in L2 memory.
Instruction Cache	⇨ L1P (4K) can be used as cache with LRU algorithm.

The Analog Devices ADSP-21161 SHARC DSP Processor

The ADSP-2116x SHARC DSPs are a second generation family of general purpose 32-bit, floating point DSPs. They are based on, and are code compatible with, the Analog Devices ADSP-2106x SHARC family. The ADSP-21161 device has an on-chip memory of 1 Megabits. This is equivalent to 128K bytes of memory. The device also consists of an instruction cache to accomplish a three bus performance by the core. The I/O processor consists of a 14 channel DMA controller, 2 link ports (each 8 bit wide operating at the clock speed of the core), SPI port (2 DMA channels are shared with link ports for transmit and receive SPI data), four serial ports (each providing 2 channels) and an external port (4 channels). The device also supports glueless multiprocessing. Figure 2 shows a block diagram of the ADSP-21161 processor architecture.

Figure 2. ADSP-21161 Architecture Block Diagram



ADSP-21161 Computational Core

In figure-2, the computational core consists of two processing elements PEx and PEy, program sequencer and two Data Address Generators (DAGs). Each processing element consists of a data register file, ALU, multiplier, barrel shifter. This part of the DSP's architecture performs general math. The program sequencer is responsible for instruction fetch, control flow instruction execution. It consists of interrupt logic, cache and stacks. The DAGs are responsible for data fetch on both the onchip buses from internal and external memories.

Another key feature of the ADSP-21161 processor is its SIMD architecture. SIMD stands for **S**ingle **I**nstruction **M**ultiple **D**ata. In the SIMD model of program execution, the same instruction is dispatched to both the processing elements, but the data on which both the processing elements operate upon will be different. Application program developers typically utilize the implicit parallelism in many DSP algorithms and utilize both the processing elements.

Data Formats

The ADSP-21161 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-21161 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 bit of the 64

bit product to either one of the 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.

Result Rounding
The ADSP-21161 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-21161'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 ADSP-21161 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 any where inside another word, extract the exponent from a fixed point input and count leading ones or zeros in a fixed point output.

Instruction Set
A DSP's architecture can reduce overhead by providing many single instructions for arithmetic operations. The ADSP-21161 instruction set includes many non-traditional 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 the issue in more detail, but listing 2 provides one example of the advantage these additional instructions provide. It should also be noted that the instruction set from the traditional ADSP-21060 (SHARC) processor has not been changed in the ADSP-21161 processor. The same instruction gets executed on both the processing elements but on their respective register sets (data).

Register File
The block diagram of the ADSP-21161 register file and computational units is shown in the lower quarter of figure 2. The register file provides local storage for arithmetic input data and results. Note that there is one each register file dedicated for both the processing elements.

Each register file in each of the processing elements consists of sixteen 40 bit registers. Hence in summary, there are a total of 32 registers in both the processing elements available at any point of time. Within one instruction, all registers can be swapped with a set of secondary registers (background registers). The secondary registers are typically used during interrupts or subroutine calls to eliminate time consuming memory transfers of register file data. Use of these 32 secondary registers provides a total of 64 register file registers. The ADSP21161's large data register file lets you efficiently implement register intensive algorithms , such as Radix-4 FFT.

Parallel Operations
In one instruction cycle, the DSP can perform four 32-bit word memory transfers, source four inputs to the two ALUs in PEx and PEy, source four inputs to the two multipliers in PEx and PEy, and store ALU and multiplier results. The four 32-bit word memory transfers can be either reads or writes of both. 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-21161. The processor comparison summary section compares the number and type of parallel operations supported by theADSP-21161 and TMS320C67112.

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-21161 satisfies this requirement with four 32bit -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-21161's 48-bit instruction word lets the DSP directly address the full 32-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. In SIMD mode, register reads and writes happen from the register pairs of both the processing elements. This no explicit loading of registers in PEy is not required in SIMD mode. The ADSP-21161 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-2116x has two Data Address Generators (DAG's). 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.

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 DAG's contain 16 sets of index, base and length registers, you can place up to 16 arbitrary length circular buffers anywhere in the 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 may 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-21161 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-21161 and TMS320C6711 are DO loops (repeated execution of the code block), branching (program execution jumps conditionally and unconditionally to a non sequential address), interrupts and pipe line delays.

Application developers can use the ADSP-21161 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 a zero overhead, six level deep loop nesting and respond to interrupts from any loop nesting level.

An ADSP-21161 program can specify branch conditions (typically) based on the arithmetic results of a bit operation, 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 of any of four programmable input pins
- ALU status
- Multiplier status
- Shifter status
- System bit test status

The ADSP-21161 also has a complex conditional instructions supporting execution of an arithmetic operation and a data move based on single condition. The ADSP-21161 lets you specify branch addresses within the full 32-bit program address space in any branch instruction. When the program sequencer branches execution to a subroutine (upon execution of a subroutine *call* instruction) or an interrupt, the return address is stored in the PC stack. An ADSP-21161 subroutine call is a single cycle, single operand instruction and uses a direct or indirect address.

To optimize sequential program execution, the ADSP-21161 uses a three level deep instruction to pipeline. The pipeline influences execution branching by introducing a delay (two instruction cycles) as the pipeline 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. (Users 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. Non delayed branches require three cycles to execute on ADSP-21161.

Note that on the ADSP-21161 all execution branches (JMP, CALL, RTS or RTI) also can occur in parallel with a computation. This parallel execution along with the use of delayed branch can completely eliminate all overhead related to branching. If properly coded, a program's use of subroutine call and return, in many cases incurs no branching overhead.

The ADSP-21161 responds to external interrupt inputs and has an internal timer for generating periodic software interrupts. The ADSP-21161 responds (vectors) to an interrupt with no more than four cycles 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 can also code delayed branch within the vector table; an option that can provide no-overhead interrupts.

Interrupt support on the ADSP-21161 includes a fifteen level deep status stack. When an interrupt occurs, the DSP automatically pushes the arithmetic, mode and interrupt mask status on the status stack. (Status can be manually pushed onto its stack for subroutines) The ADSP-21161 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 V. ADSP-21161, Computational Core Summary

Computational Core Feature	ADSP-21161 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	⇨ Specified in a control register.
Arithmetic result related interrupt support	⇨ Interrupts on ALU/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 fixed point input.
Single instruction arithmetic operation support	⇨ All arithmetic instructions (fixed or floating point) take single cycle.
Parallel operation support	⇨ Can execute upto 8 parallel instructions. Easy to schedule code.
Context switching (background registers) support	⇨ Supported.
Direct & Indirect Memory Addressing Support	
Address range support	⇨ Addresses entire 32-bit address space.
Addressing methods supported	⇨ Direct and Indirect addressing
Circular buffers supported	⇨ Supported with arbitrary lengths.
Context switching (background registers) support	⇨ Supported.
Bit reversed addressing	⇨ Supported implicitly and through instr.
Program Sequencing	
Do loop support	⇨ 6 level deep nesting with zero overhead.
Program branching support	⇨ Branch to any 24 bit addresses.
Subroutine calls	⇨ Supported.
Interrupts	⇨ External input and internal timers, 4 cycle latency.
Pipeline delays	⇨ Three level pipe line with delayed branch support. Compute or memory transfer allowed in parallel with branch operation

ADSP-21161, I/O Capabilities

The I/O processor and external port sections of the ADSP-21161 architecture (figure 2) provide the DSP with I/O access to external devices. This description of ADSP-21161 I/O capabilities focuses on support for Direct Memory Access (DMA), communications I/O (external and Link ports), serial I/O (SPORTS) and multiprocessor interfaces.

DMA lets DSP (or external devices) access the DSP's internal memory and I/O ports without processor intervention (overhead). The ADSP-21161's I/O processor has 14 DMA channels that can be configured to service any combination of the following ports :

- Four 8, 16, 32 bit wide DMA channels to memory through external port.
- Eight serial DMA channels to external peripherals through SPORTS.
- Two shared DMA channels to external processors or peripherals through link ports or SPI port.

Communication Ports
The ADSP-21161's external port and two link ports provide interprocessor and peripheral communications. DMA channels through the external port each have 8 deep 64-bit wide FIFO buffer. The external port supports data throughput of up to 400 Mbytes per second. The ADSP-21161 processor also has an on-chip SDRAM controller that enables glueless integration with an external SDRAM of up to 256 Mbit SDRAMs. The interface also enables connecting up to 4 external memory banks without any decode logic (The decode logic is built into the processor itself).
Link ports operate at the frequency of the DSP's internal clock speed and have a two level deep by 48-bit wide FIFO buffer. These ports automatically pack (8 bit transfers into 32-bit or 48-bit words) and unpack (32 bit or 48 bit words into 8 bit transfers) data and instruction words. The link ports support data throughput of up to 100 M bytes per second each. Together the two link ports support total data throughput of 200 M bytes per second.

Serial Ports
Serial I/O (through SPORTS on the ADSP-21161) provides interface support for a wide variety of peripheral devices including many industry standard data converters, codecs and other processors. The SPORT's configurable features include the following :

- ADSP-21161 has 4 serial ports (8 channels supported)
- Fully independent transmit and receive sections
- U-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
- Time Division Multiplexed (TDM) mode
- I2S support (A popular industry standard interface)

Serial Peripheral (Compatible) Interface
Serial Peripheral Interface (SPI) is an industry standard synchronous serial link, enabling the ADSP-21161 SPI-compatible port to communicate with other SPI-compatible devices. SPI is a 4-wire interface consisting of two data pins, one device select pin, and on clock pin. The ADSP-21161 SPI port supports:

- Full-duplex, supporting both master and slave modes
- Support for slave serial boot mode from 8-, 16-, or 32-bit host SPI devices
- Operates in a mult-master envoronment by interfaceing with up to 4 other SPI-compatible devices, either acting as a master or slave
- Programmable baud rate and clock phase/polarities
- Supports the use of open drain drivers to support multi-master environments to avoid data contention

Multiprocessor Support
The ADSP-21161 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 and external). Also, host interface support in the external port lets the DSP interface with any 8-bit or 16-bit or 32-bit host processor. Link ports, with their flexible configuration features and high throughput, are ideal for multiprocessing applications using multiple ADSP-21161s (or in combination with other processors) without shared memory.

Table V1. ADSP-21161, I/O Capabilities Summary

I/O Feature	ADSP-21161 Supports …
Direct Memory Accessing (DMA)	
Number of DMA Channels	⇨ 14
DMA Channel Configurations	⇨ 4 external port channels, 2 channels shared between link ports and SPI port, 8 serial port channels
DMA data buffering	⇨ 8 deep 64 bit external port FIFO, 2 deep 48 bit wide link port FIFO, 2 deep 32 bit wide serial

	port FIFO, 2 deep 32-bit SPI FIFO
Total DMA I/O throughput	⇨ 800 M bytes per second.
Communication Ports	
Description of communication ports	⇨ 2 8-bit link ports operating at core frequency.
Total communications port throughput	⇨ 200 M bytes per second.
Serial Ports	
Description of serial ports	⇨ 4 serial ports (each with 2 channels) that provide built in support for companding, configurable word size/data format and TDM operation in multi channel mode. ⇨ SPI compatible port (8-bit, 16-bit and 32-bit)
Multiprocessor interface	
Shared global memory support	⇨ Unified multiprocessor memory map; upto 6 ADSP-21161s can directly access each others internal RAM and I/O registers at upto 400M bytes per second. ⇨ Glueless shared memory through bus mastering and bus arbitration through external port.
Inter processor communications support	⇨ Inter processor communication possible through link ports as well.

ADSP-21161 Memory

The dual ported SRAM section in the ADSP-21161 architecture (figure 2) represents the memory of the DSP. Three features of this internal memory map the ADSP-21161 unique among the floating point DSPs : size, dual-porting and triple bussing.

The onchip SRAM (1 bit SRAM) can hold 32K x 32 bits of DM data or 21.3K 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 memory architecture lets the I/O processor operate without incurring overhead on the computational core. Also, the memory has three address and data buses (PMA & PMD, DMA & DMD, IOA & IOD), allowing simultaneous access to instructions, data and I/O data. This bus structure lets the ADSP-21161 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-21161 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.

Table VII. ADSP-21161, Memory Summary

Memory Feature	ADSP-21161 Supports
Internal Memory Size	⇨ 32K x 32 bits internal memory
Memory addressing	⇨ 32 bit address, unified Multi Processor 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 access complete in one cycle for reads or writes
Instruction Cache	⇨ Two way set associative 32 entry cache
Memory Accessibility	⇨ Unified memory map lets 6 ADSP-21161 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 lets user use slower, low cost external SRAMs.

When accessing external memory, the ADSP-21161's integrated SDRAM controller enables the ADSP-21161 to transfer data to and from synchronous SDRAM at the core clock frequency (100 MHz). The SDRAM interface provides a glueless interface with standard SDRAMs - 16 Mbit, 64M-bit, 128 Mbit, and 256 Mbit. The total addressable space when mapping SDRAM to all 4 banks is 254 M-words.

ADSP-21161 Vs. TMS320C6711/TMS320C6712 – Comparison Summary

This section provides a brief summary of the individual processor sections then provides a side by side comparison of ADSP-21161 and TMS320C6711/TMS320C6712 benchmarks, instruction set, and hardware features not compared 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-21161 and TMS320C6711/TMS320C6712. 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 1X summarizes the most crucial DSP features that differentiate the ADSP-21161 and TMS320C6711/TMS320C6712, 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-21160 Vs TMS320C6711 "Key" Features Comparison Summary

DSP "Key" Features	ADSP-21161	TMS320C6712	TMS320C6711
Computational core performance	600 MFLOPS	600 MFLOPS @ 100 MHz	600 MFLOPS @ 100 MHz 900 MFLOPS @ 150 MHz
Internal RAM size	32K x 32 bits	18K x 32 bits	18K x 32 bits
Internal RAM I/O overhead	No overhead. Eliminated by dual ported RAM	Overhead possible since I/O happens through cycle stealing.	Overhead possible since I/O happens through cycle stealing.
Shared global memory support	• Unified multiprocessor memory map; upto 6 ADSP-21161s can directly access each others internal IOP registers up to 400M bytes per second. • Glueless shared memory with bus mastering and bus arbitration through external port	• None. External bus arbitration circuitry required to support global memory through external bus mux.	• None. External bus arbitration circuitry required to support global memory through external bus mux.
IO data through put • Off chip to on-chip transfers • On chip to off-chip transfers • Communications port Xfers • Serial port transfers	• 400M bytes per second • 400M bytes per second • 200M bytes per second (link) • 50M bytes per second	**(IDEAL CASE)** • 200M bytes per second • 200M bytes per second • No link ports • 25M bytes per second (Note that we need to add cycles lost by EDMA due to L1P and L1D access to L2 simultaneously. Hence the through put figures of TMS320C6712 are much less than the figures given above. The figures vary from application to application and cannot be	**(IDEAL CASE)** • 400M bytes per second • 400M bytes per second • No link ports • 25M bytes per second (Note that we need to add cycles lost by EDMA due to L1P and L1D access to L2 simultaneously. Hence the through put figures of TMS320C6712 are much less than the figures given above. The figures vary from application to application and cannot be computed)

		computed)	

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 FIR filter. Listings 2 and 3 below show a comparison of ADSP-21161 and TMS320C6711/TMS320C6712 assembly code for the core of their corresponding (recommended) algorithms.

Listing 2. ADSP-21161 FIR filter (core loop has one instruction)

```
        bit set MODE1 CBUFEN;           /* Circular Buffer Enable, one cycle effect latency */
        nop;                            /* Circular Buffering not in effect until next cycle */

        s0 = dm(i0, m1);                /* move pointer to delay[1] */

        bit set MODE1 PEYEN;            /* SIMD Mode Enable, one cycle effect latency */
        s0 = dm(i0, m2);                /* load s0 with the value of delay[1] for SIMD store,
                                           move pointer to delay[0] */

        dm(i0,m3)=f0, f4 = pm(i8,m9);           /* transfer sample to delayline, done in SIMD to
                                                   load end of buffer + 1 */
                                                /* to compensate for circular buffer issue
                                                   described above, read 2 coeffs */

        f8=f0*f4, f0=dm(i0,m3), f4=pm(i8,m9);   /*samples*coeffs,read 2 samples, read 2 coeffs */
        f12=f0*f4, f0=dm(i0,m3), f4=pm(i8,m9);  /*samples*coeffs,read 2 samples, read 2 coeffs */
        lcntr=r3, do macs until lce;                    /* FIR loop */
macs:   f12=f0*f4, f8=f8+f12, f0=dm(i0,m3), f4=pm(i8,m9); /* samples * coeffs, accum, read 2
                                                            samples, read 2 coeffs */
        f12=f0*f4, f8=f8+f12, s0=dm(i0,m2);     /* samples * coeffs, accum, dummy read to move
                                                   pointer to oldest sample */
        f8=f8+f12;                              /* final SIMD accum */
        r12=s8;                                 /* move PEy total into PEx register file */

        rts (db);
        bit clr MODE1 CBUFEN | PEYEN;   /* Circular Buffer Disable, SIMD Mode Disable */
        f8=f8+f12;
```

Listing 3 : TMS320C6711/TMS320C6712 FIR Filter

```
_fir:
*** BEGIN Benchmark Timing ***
B_START:
* Prolog Begins ********************************************************************
LDDW .D1 *A4++[1],B1:B0          ; load x1:x0 from memory
|| MV .L1X B4,A8                    ; f ptr_h = h
|| SUB .S1 A8,4,A2                  ; f ocntr = numY - 4
|| SHL .S2 B6,1,B9                  ; f B9 = (numH) << 1

LDDW .D1 *A8++[1],A5:A4          ; load h1:h0 from memory
|| MV .S2X 4,B8                        ; f ptr_x = x
```

```
|| SUB .S1X B6,4,A0  ; f ireset = numH – 4

 LDDW .D2 *B8,B5:B4 ; load x3:x2 from memory
|| MV .L2X A0,B2  ; icntr = ireset
|| MV .L1 A0,A1  ; lcntr = ireset
|| SUB .S2 B9,8,B9  ; f xreset = B9 – 8

 LDW .D2 *+B8[2],A3 ; load x4 from memory

 LDDW .D2 *B8++[1],B1:B0 ; @ load x1:x0 from memory

 LDDW .D1 *A8++[1],A5:A4 ; @ load h1:h0 from memory

 LDDW .D2 *B8,B5:B4 ; @ load x3:x2 from memory
|| MPYSP .M1X B1,A5,A9 ; prod1 = x1 * h1
|| MPYSP .M2X B0,A4,B6 ; prod0 = x0 * h0

 LDW .D2 *+B8[2],A3 ; @ load x4 from memory
|| MPYSP .M1X B4,A5,A9 ; prod3 = x2 * h1
|| MPYSP .M2X B1,A4,B6 ; prod2 = x1 * h0

OLOOP:
   [A1] LDDW .D2 *B8++[1],B1:B0 ; @@ load x1:x0 from memory
|| MPYSP .M1X B5,A5,A9 ; prod5 = x3 * h1
|| MPYSP .M2X B4,A4,B6 ; prod4 = x2 * h0
|| B .S1 LOOP  ; if(icntr) branch to LOOP

   [A1] LDDW .D1 *A8++[1],A5:A4 ; @@ load h1:h0 from memory
|| MPYSP .M1 A3,A5,A9 ; prod7 = x4 * h1
|| MPYSP .M2X B5,A4,B6 ; prod6 = x3 * h0
|| ZERO .S1 A7  ; sum1 = 0
|| ZERO .S2 B7  ; sum0 = 0
**** Loop Begins ****************************************************************
LOOP:
   [A1] LDDW .D2 *B8,B5:B4 ; @@ load x3:x2 from memory
|| MPYSP .M1X B1,A5,A9 ; @ prod1 = x1 * h1
|| MPYSP .M2X B0,A4,B6 ; @ prod0 = x0 * h0
|| ADDSP .L1 A7,A9,A7 ; sum1 = prod1 + sum1
|| ADDSP .L2 B7,B6,B7 ; sum0 = prod0 + sum0

   [A1] LDW .D2 *+B8[2],A3 ; @@ load x4 from memory
|| MPYSP .M1X B4,A5,A9 ; @ prod3 = x2 * h1
|| MPYSP .M2X B1,A4,B6 ; @ prod2 = x1 * h0
|| ADDSP .L1 A7,A9,A7 ; sum3 = prod3 + sum3
|| ADDSP .L2 B7,B6,B7 ; sum2 = prod2 + sum2
|| [A1] SUB .S1 A1,2,A1  ; if(lcntr) lcntr -= 2

   [A1] LDDW .D2 *B8++[1],B1:B0 ; @@@ load x1:x0 from memory
|| MPYSP .M1X B5,A5,A9 ; @ prod5 = x3 * h1
|| MPYSP .M2X B4,A4,B6 ; @ prod4 = x2 * h0
|| ADDSP .L1 A7,A9,A7 ; sum5 = prod5 + sum5
|| ADDSP .L2 B7,B6,B7 ; sum4 = prod4 + sum4
|| [B2] B .S1 LOOP  ; if(icntr) branch to LOOP

   [A1] LDDW .D1 *A8++[1],A5:A4 ; @@@ load h1:h0 from memory
|| MPYSP .M1 A3,A5,A9 ; @ prod7 = x4 * h1
|| MPYSP .M2X B5,A4,B6 ; @ prod6 = x3 * h0
|| ADDSP .L1 A7,A9,A7 ; sum7 = prod7 + sum7
|| ADDSP .L2 B7,B6,B7 ; sum6 = prod6 + sum6
|| [B2] SUB .D2 B2,2,B2  ; if(icntr) icntr -= 2
```

```
||[!B2] SUB .S2 B8,B9,B8 ; o if(!icntr) ptr_x -= xreset
||[!B2] SUB .S1X A8,B9,A8 ; o if(!icntr) ptr_h -= xreset
**** Loop Ends ************************************************************
 ADDSP .L1X B7,A7,A7 ; o temp1 = sum0 + sum1
|| [A2] SUB .D1 A8,16,A8 ; o ptr_h -= 16
|| [A2] LDDW .D2 *B8++[1],B1:B0 ; p load x1:x0 from memory

 ADDSP .L2X B7,A7,B7 ; o temp2 = sum2 + sum3
|| [A2] LDDW .D1 *A8++[1],A5:A4 ; p load h1:h0 from memory

 ADDSP .L1X B7,A7,A7 ; o temp3 = sum4 + sum5
|| [A2] B .S1 OLOOP  ; o if(ocntr) branch to OLOOP
||[!A2] B .S2 B3  ; f if(!ocntr) return
|| [A2] LDDW .D2 *B8,B5:B4 ; p load x3:x2 from memory

 ADDSP .L2X B7,A7,B7 ; o temp4 = sum6 + sum7
|| [A2] LDW .D2 *+B8[2],A3 ; p load x4 from memory

 STW .D1 A7,*A6++[2] ; o store temp1
|| [A2] LDDW .D2 *B8++[1],B1:B0 ; p load x1:x0 from memory
|| MV .S2X A6,B6  ; o B6 = A6

 STW .D2 B7,*+B6[1] ; o store temp2
|| [A2] LDDW .D1 *A8++[1],A5:A4 ; p load h1:h0 from memory
|| [A2] MV .S1 A0,A1  ; p lcntr = ireset
|| [A2] MV .S2X A0,B2  ; p icntr = ireset

 STW .D1 A7,*A6++[1] ; o store temp3
|| [A2] LDDW .D2 *B8,B5:B4 ; p load x3:x2 from memory
|| [A2] MPYSP .M1X B1,A5,A9 ; p prod1 = x1 * h1
|| [A2] MPYSP .M2X B0,A4,B6 ; p prod0 = x0 * h0

 STW .D1 B7,*A6++[1] ; o store temp4
|| [A2] SUB .S1 A2,4,A2  ; o if(ocntr) ocntr -= 4
|| [A2] LDW .D2 *+B8[2],A3 ; p load x4 from memory
|| [A2] MPYSP .M1X B4,A5,A9 ; p prod3 = x2 * h1
|| [A2] MPYSP .M2X B1,A4,B6 ; p prod2 = x1 * h0
* Outer Loop Ends ************************************************************
B_END:
```

Note the difference in the code size for this simple FIR filter. One key reason for the increase in the code size is the lack of hardware loop support in TMS320C6711 processor. Hence the user needs to use a branch instruction to implement the loops. But in an effort to reduce the branch penalties, code has to be scheduled in the delay slots of the branch. This results in unrolling the loops and hence results in increase in code size. The code size increase also results in performance reduction in TMS320C6711/TMS320C6712 as it will result in cache miss penalties (L2 cache) and hence the reduction in performance.

The complexity involved in hand coding the assembly programs in TMS320C6711/TMS320C6712 processor can be visualized from the sample code given above. Since there is a multi-cycle latency involved in the execution of some of the instructions, user needs to schedule the instructions for optimal usage of all the execution units. This is quite a complicated task and has to rely on tools like compilers for accomplishing the task. Since the compiler generated code would not be as efficient as hand written code, some more MIPS would be lost in the process of compilation.

Table IX. Core Benchmarks for DSP performance : ADSP-21160 Vs TMS320C6711/TMS320C6712

DSP Benchmarks	ADSP-21161 (at 100M Hz)	TMS320C6711/ TMS320C6712 (at 100M Hz)	TMS320C6711 (at 150M Hz)

1024-point complex Radix-4 FFT	92 µ seconds Memory : 1200	180µ seconds Memory : 1200	120µ seconds
Matrix multiply (10x1 * 1x10 matrix multiply)	0.9µ seconds Memory : 96	1.2µ seconds Memory : 320	0.8µ seconds
Complex FIR filter (100 coefficients, 100 output samples)	206µ seconds Memory : 102	215µ seconds Memory : 320	144µ seconds
Vector addition (100 vector)	0.108µ seconds Memory : 72	0.108µ seconds Memory : 120	72µ seconds
Vector dot product (100 vector)	0.056µ seconds Memory : 78	0.074µ seconds Memory : 188	0.050µ seconds

Note :

1. ***The bench marks given above for TMS320C6711/TMS320C6712 processor assume single cycle memory access. This is however unlikely over repeated execution of the application for different data frames due to the cache based memory architecture of the processor. Hence it will result in a higher MIPS count and hence more time to execute.***
2. ***Power consumption : At 150 MHz, the power consumption will be 1.5 times the power consumption at 100 MHz for the TMS320C6711/TMS320C6712 device.***
3. ***Memory is specified in bytes.***

The tables below 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**	Indicates that the DSP does not provide the features or function
32-bits	**A number**	Indicates the type of support the DSP provides

Table X. DSP Math operations of the ADSP-21161 Vs TMS320C6711/TMS320C6712

Digital Signal Processor Math Operations Supported …	ADSP-21161	TMS320C6711
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	✓	✓
ALU supports :		
40-bit floating point	✓	✓
32-bit fixed point	✓	✓
Multiprecision addition and subtraction	✓	✓
Seed 1/x, seed 1/√x	✓	✓
Minimum, maximum, average, clip & scale	✓	Ø
Absolute value of (x+y) & (x-y)	✓	Ø
Simultaneous (x+y) & (x-y)	✓	Ø
8-bit accumulated compare status	✓	Ø
Barrel shifter supports :		
Logical and arithmetic shift	✓	✓
Rotate	✓	✓
Bit-wise test, clear toggle and set	✓	✓
Shift & logical OR with register (extended precision shifts)	✓	Ø
Field extract and deposit	✓	✓
Count leading ones and zeros	✓	Ø
Multifunction (simultaneous operations)		
Fixed or floating point multiply and add	✓	✓
Fixed or floating point multiply and subtract	✓	✓
Fixed or floating point multiply, add and subtract	✓	Ø
Fixed point multiply and convert (fixed to floating point)	✓	✓

Fixed point multiply and convert (floating to fixed point)	✓	✓
Floating point multiply and find average	✓	Ø
Floating point multiply and find absolute value	✓	✓
Floating point multiply and find maximum	✓	Ø
Floating point multiply and find minimum	✓	Ø

Table XI. ALU instructions of the ADSP-21161 Vs TMS320C6711/TMS320C6712

ADSP-21161 Instruction	Description	ADSP-21161	TMS320 C6711
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/√X the reciprocal square root of floating point operand X	✓	✓
FLOAT Y	Converts the fixed point operand to 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 point or floating point operands and divides the result by 2	✓	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 us less than the ABS Y. Returns ABS Y (if X is positive) or –ABS Y if (X is negative) if ABS X is greater than 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

Table XII. Program Sequencing Features of the ADSP-21161 Vs TMS320C6711/TMS320C6712

Digital Signal Processor Program Sequencing Supports …	ADSP-21161	TMS320C6711
Instruction clock	100 MHz	100M/150 MHz
Instruction pipeline levels ⇨ 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; 'ADSP-21161-two instructions & 'C6711-five instructions (requires intelligent code scheduling)	3	16
Interrupts		
⇨ Latency	4	15 cycles
Zero overhead loop nesting levels	6	0
Instruction cache	✓	✓
Single-cycle/operand branching with 24-bit address	✓	✓ (23 bits)
Delayed branch	✓	✓
Conditional branch with loop abort	✓	Ø
Delayed branch returns	✓	Ø
Delayed branch subroutine calls	✓	Ø
Pre-modified indirect branching	✓	✓
Compute in parallel with conditional branch (JUMP, CALL, RTS, or RTI)	✓	✓
Data transfer in parallel with branch (JUMP)	✓	✓
Interrupts on index/address register modulo overflow	✓	Ø
Interrupts on arithmetic status	✓	Ø
Status stack	✓	Ø

Table XIII. Data Addressing features of the ADSP-21161 Vs TMS320C6711/TMS320C6712

Digital Signal Processor Memory Addressing Supports …	ADSP-21161	TMS320C6711
Number of index registers	16	No specific index registers.
Number of modify registers	16	No specific modify registers.
Number of length registers	16	No specific length registers.
Number of base address registers	16	No specific base registers.
Secondary index, modify, length, base registers	Full set	Not available
Direct addressing (read or write)	32-bit	Not available
Direct read, one compute	6-bit	Not available
Compute, modify by immediate offset	6-bit	Not available
Direct read, two computes	6-bit	Not available
Direct write, one compute	6-bit	Not available
Direct write, two computes	6-bit	Not available
Bit-reversed addressing	✓	Not available directly. Should use an extra instruction.
Single cycle—two indirect writes, no compute	✓	✓
Single cycle—two indirect writes, one compute	✓	✓
Single cycle—two indirect writes, two computes	✓	✓
Single cycle—two indirect reads, two computes	✓	✓
Single cycle—indirect read & write, one compute	✓	✓
Single cycle—indirect read & write, two computes	✓	✓

Conclusion

For high performance computational systems, designers must look beyond instruction execution speed to compare DSPs. One must look at computational efficiency, I/O capabilities and data throughput, system cost, level of integration , programmability, easy of use and efficiency of tools. The Analog Devices ADSP-21161 SIMD SHARC processor offer uncompromising performance, I/O peripheral interface and function integration.

References

The following sources contributed information to this application note:

1. *TMS320C6000 CPU and Instruction Set Reference Guide*, Texas Instruments
2. *TMS320C6000 Onchip Peripherals Reference Guide*
3. *ADSP-21161 SHARC User Manual,* Analog Devices, Inc.