TI's New 'C6x DSP Screams at 1,600 MIPS

Radical Design Offers 8-Way Superscalar Execution, 200-MHz Clock Speed

by Jim Turley and Harri Hakkarainen

In a leap to the front of the pack, Texas Instruments unleashed the most radical digital-signal processor to date, a VLIW design that runs at 200 MHz and executes up to eight instructions at once. Under ideal conditions, the high-end DSP runs at an astonishing 1,600 MIPS—5× to 10× the performance of today's leading DSP chips.

The new TMS320C6201 will be the first in a family of new DSPs from TI based on the 'C6x core. When the chip begins production in 3Q97, it will be the fastest—and nearly the most expensive—fixed-point DSP available. TI hopes to appeal to makers of telecommunications equipment, both wired and wireless.

The chip relies on very long instruction word (VLIW) techniques to achieve its impressive performance. VLIW has appeared recently in media processors from Chromatic and Philips, chips that are also DSPs in a sense. This once-esoteric approach is now becoming almost routine in highly parallelizable applications in media- and signal-processing.

Eight-Way VLIW, 200 MHz—Oh, My!

The 'C6x, which TI has been developing for three years, is incompatible with anything the company has done before. In many respects, the chip resembles a high-end RISC processor with an unusual instruction set more than a conventional fixed-point DSP.

The core is eight-way superscalar, has an 11-stage pipeline, and maintains two sets of four execution units, shown in Figure 1. The part is nominally divided in half, with 16 registers and four execution units on each side (A and B). Crossovers allow limited use of the A registers by B-side execution units, and vice versa. With all eight execution units running, the 'C6x can perform 1.6 billion operations per second at 200 MHz.

The four execution units on each side are a matched set. Each side contains a 40-bit integer ALU, a 40-bit shifter, a 16-bit multiplier, and a 32-bit adder that is also used for address generation. Each of the execution units has access to the same resources and, with only a few exceptions, completes its operation in a single clock cycle.

The two 40-bit ALUs, or L units, perform arithmetic and logical compares, normalization, and bit-count operations. All L-unit operations complete in one clock cycle.

Multiplication is handled by the M units, which can perform both signed and unsigned 16×16→32-bit multiplication. Latency is two cycles, with single-cycle throughput.

Both S units have a 32-bit ALU and a 40-bit shifter. The S units can perform some of the same 32-bit arithmetic operations as the L units, along with 32-bit and 40-bit shifts. The S units are also responsible for branching and branch-address generation. A 32-bit adder allows the D units to perform simple arithmetic operations, but their primary purpose is address generation.

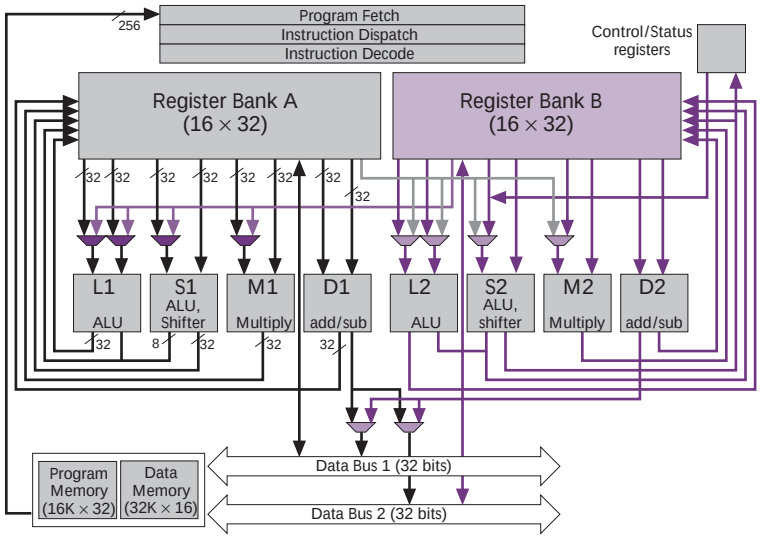

Figure 1. This block diagram of the 'C6x core illustrates the eight execution units, arranged in two sets of four. The 256-bit-wide instruction bus allows the chip to fetch eight 32-bit instructions per cycle.

Registers Are Massively Ported

From the programmer's perspective, the 'C6x has 32 general-purpose 32-bit registers: A0–A15 and B0–B15. Any register can hold any address or data value; 40-bit results are stored in adjacent registers. Five of the registers—A1, A2, and B0–B2—can be used for conditional tests (A0 is reserved for future expansion). Finally, four registers from each bank serve special duty as circular-addressing pointers.

Both register banks have nine 32-bit read ports and six 32-bit write ports, allowing all four execution units in a bank to access the same register simultaneously, if required. Two crossovers, one in each direction, provide limited access to the 16 registers from the opposite bank. Multiplier M1, for example, can take one operand from any B register. With only one pair of crossovers, data crossings are limited to one execution unit per bank per cycle. In the example, the L1, S1, and D1 units would all have to make do with data from the A registers. This restriction will probably be most apparent when programs try

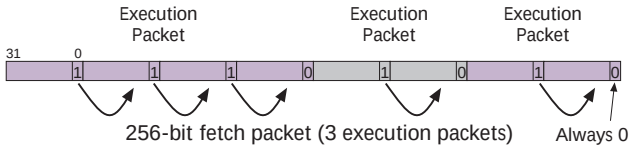

Figure 2. The least significant bit of each instruction in a fetch packet indicates whether the instruction can be executed in parallel with its successor. The instructions in this example will be executed in three separate packets of four, two, and two instructions.

to perform two simultaneous memory accesses with address pointers from the same register bank.

Although both the L and S units can perform 40-bit calculations, there is only one 40-bit write port per bank to the register file. It is the programmer's responsibility to avoid generating two 40-bit results in the same cycle.

Instruction Set Shows VLIW Techniques

The 'C6x is the first VLIW chip from TI and the first conventional DSP with VLIW instructions. The core devours eight 32-bit instructions at once from its on-chip 256-bit instruction bus. All eight instructions are sent to the eight execution units before the next 256-bit meta-instruction is fetched.

In TI's nomenclature, the eight-instruction group is known as a fetch packet. The 'C6x always fetches a complete fetch packet at once, and fetch packets must be 32-byte aligned. However, not all eight instructions in the fetch packet are necessarily executed simultaneously.

Although there are eight instructions in each fetch packet and eight execution units, each instruction does not necessarily correspond to one execution unit; the instructions are not position-dependent within the fetch packet, which is the traditional VLIW method. Instead, each instruction is encoded for a specific execution unit. An ADD instruction, for example, can be encoded for the L1, L2, S1, S2, D1, or D2 unit. Programmers can explicitly dictate the binding of instructions or leave it to the assembler or compiler.

Under ideal circumstances, all eight of the 'C6x's execution units can be kept busy on every cycle. In practice, data dependencies, resource conflicts, multicycle operations, and other realities of programming will force less-than-total utilization of the core's resources. Rather than waste space in the fetch packet by padding with NOPs, TI allows multiple groups of instructions in a single fetch packet.

Independent of the 256-bit fetch packet, the 'C6x defines an execute packet, which can be 1–8 instructions in the fetch packet. All instructions in an execute packet are dispatched together. It is the programmer's (or compiler's) responsibility to guarantee that all instructions in the execute packet can, indeed, be dispatched simultaneously—the 'C6x hardware does no dependency checking among instructions.

Execute packets are identified by the least significant bit in each instruction. If the bit is set, the instruction may be dispatched in parallel with the subsequent (next higher-addressed) instruction. If all eight instructions in the fetch packet have their LSB set, all eight will be dispatched simultaneously. If none have their LSB set, the eight instructions in the fetch packet will be executed serially. Figure 2 illustrates an example with a four-instruction execute packet followed by two two-instruction packets.

Price & Availability

TI's TMS320C6201 is sampling now in a 352-contact BGA package. The 196-mm^2 part is fabricated in a 0.25-micron five-layer-metal process and runs from a 2.5-V supply. Pricing has been set at $135 in 1,000-unit quantities. Production is scheduled for 3Q97. For more information, contact TI (Denver) at 800.477.8924, extension 4500, or visit *www.ti.com/sc/C6x.*

VLIW Approach Breaks Up Basic Operations

Table 1 lists the entire 'C6x instruction set. Every 'C6x instruction (including the lone branch) can be made conditional, based on the zero/nonzero status of the five condition registers. Theoretically, all eight instructions in a packet could each be predicated on some different condition. This type of predicated execution is also used in the Philips TM-1, a VLIW media processor.

The 'C6x has none of the moderately complex instructions most DSP chips have. Multiply-accumulate, for example, is handled as a multiply followed by a separate add. Fetching a memory-resident coefficient would be handled as a third, independent, operation. Because it divides conventional DSP functions into separate instructions, the 'C6x needs a somewhat different definition of MIPS than most DSPs. A simple 16-bit MAC becomes three different operations on the 'C6x, making the metric of 1,600 MIPS somewhat misleading. Like early RISC chips, the 'C6x thrives on high clock rates and simple operations.

Indexed addressing and loops must also be explicitly coded in software. There is no intrinsic zero-overhead loop feature in the 'C6x. Loop counters must be decremented, and a conditional branch used to return to the top of the loop. Index registers do not automatically increment or decrement; their values must be modified explicitly.

Lots of Memory

TI's initial implementation of the 'C6x architecture is the 320C6201. The part has a whopping 128 Kbytes of on-chip memory, evenly divided between program and data space. The program memory has a 256-bit path into the 'C6x core, allowing it to transfer an entire eight-word fetch packet in one cycle. At the user's option, the program memory can also be configured as a 64K direct-mapped cache.

The '6201's data memory is divided into four 16K banks, each with a 16-bit bus to the execution units. All four

Mnemonic	Description	Ex Unit L	M	S	D
Arithmetic					
ABS	Absolute value	●			
ADD{U}	Add, signed, nonsaturating {unsigned}	●		●	●
ADDA{x}	Add {byte, half, word} nonsaturating				●
ADDK	Add with 16-bit constant			●	
ADD2	Add two 16-bit halves			●	
SADD	Add, saturating	●			
MPY{U/S}	Multiply lower halves {unsigned/signed}		●		
MPYH{U/S}	Multiply upper halves {unsigned/signed}		●		
MPYHL{U/S}	Multiply upper, lower halves {un/signed}		●		
MPYLH{U/S}	Multiply lower, upper halves {un/signed}		●		
SMPY	Multiply, shift left, and saturate	●		●	●
SUB{U}	Subtract, nonsaturating {unsigned}				●
SUBA{x}	Subtract address {byte, half, word}	●			
SSUB	Subtract, saturating	●			
SUBC	Conditional divide step			●	
SUB2	Subtract two 16-bit halves	●		●	●
ZERO	Clear destination			●	
CLR	Clear bit field			●	
SET	Set bit field			●	
EXT{U}	Extract {unsigned} bit field			●	
NEG	Negate	●		●	●
NORM	Normalize; find first nonredundant bit	●			
LMBD	Leftmost bit detection	●			
SAT	Saturate 40 bits to 32 bits	●			
Miscellaneous					
IDLE	Wait for interrupt				
NOP	No operation				

Mnemonic	Description	Ex Unit L	M	S	D
Logical					
AND	Logical AND	●		●	
NOT	Logical invert	●		●	
OR	Logical OR	●		●	
XOR	Logical XOR	●		●	
SHL	Shift left			●	
SHR{u}	Shift right {unsigned}			●	
SSHL	Shift left with saturation			●	
Jump/Branch					
B	Branch			●	
BIRP	Branch using interrupt return pointer			†	
BNRP	Branch using NMI return pointer			†	
Load/Store					
LD{x}	Load {byte, half, word, double}				●
LD	Load with 15-bit offset				*
ST{x}	Store {byte, half, word}				●
ST	Store with 15-bit offset				*
STP	Store to program space			†	
MV	Move register to register	●		●	●
MVC	Move control register			†	
MVK{L,H}	Move constant to {upper, lower} half			●	
Comparison					
CMPEQ	Compare for equality	●			
CMPGT	Compare for greater-than	●			
CMPGTU	Compare for greater-than, unsigned	●			
CMPLT	Compare for less-than	●			
CMPLTU	Compare for less-than, unsigned	●			

Table 1. TI's eight-way 'C6x provides the usual complement of fixed-point DSP instructions, which it executes with a pair of four nearly identical execution units. About half of the instructions must be dispatched to a particular type of execution unit (L, M, S, or D), while some instructions, such as ADD and MV, can be handled by two or more different types of units, easing superscalar dispatch. *LD and ST with 15-bit offsets can execute only in the D2 unit. †MVC, BIRP, BNRP, and STP can execute only in the S2 unit.

banks can be accessed simultaneously, but two simultaneous accesses to the same bank are not allowed. An on-chip access rate of two words per cycle is comparable to that of most DSPs. So even though the 'C6x can execute far more instructions per cycle than other DSPs, it does not have commensurately higher data memory bandwidth.

Here, the 'C6x shows some RISC-like tendencies. The chip performs best with register-based operands, exacting a penalty for frequent memory accesses. TI expects the large, heavily ported register file will fill in for RAM in many cases.

Branches Introduce Huge Bubble

As in many high-end CPUs, the 'C6x's long pipeline is both its treasure and its burden. The chip could probably never reach 200 MHz or beyond without its 11-stage pipeline, but the long pipe also exacts a severe penalty for taken branches.

With no branch prediction, all taken branches introduce a five-cycle delay before the pipeline refills from the branch target, as Figure 3 illustrates. Unlike many CPUs, the part executes instructions in the branch delay slot, which in the 'C6x's case is a whopping 40 instructions (5 cycles × 8 instructions).

This huge delay slot leads to some unusual programming practices (as if 'C6x code weren't difficult enough to follow). One option is to make each instruction in the subsequent five fetch packets conditional, using the same condition as the original branch. Conversely, execution of these instructions could be predicated on the opposite condition, executing only if the branch is not taken.

Manual scheduling on the 'C6x exacts its own peculiar toll on programmers. Figure 4 shows the kernel of a two-tap FIR filter done in a single repeating execute packet. It performs two multiplies, two adds, and two loads while it decrements the loop counter and branches back to itself.

The effects of this packet are difficult to deduce from a cursory reading of the source code, complicated by the fact that adds, multiplies, loads, and branches all have different latencies (one, two, four, and five clocks, respectively).

On any given iteration of this loop, *n*, the 'C6x resolves the multiplies executed on iteration *n*-2, the additions from iteration *n*-1, the data loaded on iteration *n*-5, and the branch encountered on iteration *n*-6. Once under way, this loop executes two taps per cycle, which is better than most DSPs—and at 200 MHz, a lot faster.

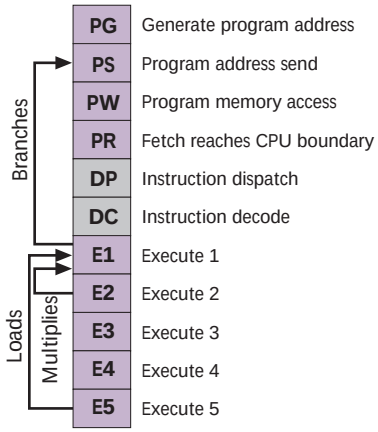

Figure 3. The 'C6x has an 11-stage pipeline, but most instructions complete after seven cycles. A 16×16 multiply operation requires one extra cycle; loads require four extra cycles to transfer data.

Dependencies Can Be Tricky to Identify

The architecture of the 'C6x is reminiscent of Intel's i860 in that it lays the pipeline open for the programmer to exploit. Crafting carefully arranged object code is crucial to extracting performance from the 'C6x. It will not be an easy task.

As mentioned previously, the chip does no dependency checking, and multiple writes to the same destination register give undefined results. Avoiding this condition can be harder than it sounds, because not all instructions have the same latency. Issuing an ADD one cycle after an MPY with the same destination will cause a failure because of their different latencies. In the MAC example, on the other hand, an ADD must be scheduled at least one cycle after the MPY for the result to propagate correctly.

Packing two mutually exclusive conditional instructions in the same execute packet is not a programming error and, in fact, can be a good idea. Programmers can create their own conditional moves, adds, or other functions simply by including in the same packet duplicate instructions that are based on opposite states of the same condition. Again, there is an opportunity for mischief here, as the 'C6x software tools cannot check for conflicting instructions that are made conditional on the contents of unrelated registers.

Manually scheduling eight execution units could be a Sisyphian task for any but the most gifted DSP codesmith. TI rightly recognizes that software-development tools are going to be key to the acceptance of the 'C6x family. The company created a specialized C compiler and an optimizing assembler. Both will be available when the first chip ships in 3Q97.

DSP programmers have historically shunned C compilers, preferring to write in assembly language. C, they say, was never designed for signal processing. Given the 'C6x's architecture, though, those times may be coming to an end. The chip is too complex for assembly-level coding. 'C6x programmers are about to go through the same convulsion as RISC programmers of a decade ago, finally moving from assembly to C.

```
loop:
        ADD.L1    A0,A3,A0  ;A0=A0+A3
||      ADD.L2    B1,B7,B1  ;B1=B1+B7
||      MPYHL.M1X A2,B2,A3  ;A3=A2(hi)×B2(lo)
||      MPYLH.M2X A2,B2,B7  ;B7=A2(lo)×B2(hi)
||      LDW.D2    *B4++,B2  ;load into B2
||      LDW.D1    *A7--,A2  ;load into A2
||      ADD.S2    -1,B0,B0  ;decrement counter
|| [B0] B.S1 loop           ;branch if B0 nonzer
```

Figure 4. In this example of an FIR filter, eight instructions fit in a single fetch packet, executing in parallel and calculating two taps per iteration.

Performance Estimates Look Impressive

The final question is whether the performance of the 'C6x is as daunting as its programming model. Although no concrete numbers are available, Berkeley Design Technology has run some benchmarks on a cycle-accurate simulator of the 'C6x.

The simplistic claim of 1,600 MIPS at 200 MHz would suggest the 'C6x is 15–40 times faster than currently available fixed-point DSPs. In reality, because the 'C6x treats every movement of data as a discrete operation, the net performance is not as great, but still impressive. A simulated 'C6x finished the BDT radix-2, 256-point FFT benchmark in 21 microseconds, which is about five times faster than Motorola's 563xx-100, eight times faster than Lucent's DSP16xx-120, and 12.5 times faster than TI's own 'C54x-50.

Code density, not surprisingly, was not even close. On the same FFT code, the 'C6x binary was about 5–6 times larger than that of either the 563xx or the 'C54x. A 256-bit instruction word definitely takes its toll in code density.

TI has priced the 'C6201 at $135 in 1,000-piece quantities. While that price certainly makes the part one of the most expensive fixed-point DSPs around, its impressive performance makes it a good overall value for those applications that can make use of its capabilities.

Finding those applications will be TI's challenge. Few kinds of systems are big enough to need a DSP of this magnitude. TI expects to replace collections of ganged DSPs, such as those used in simulators, central-office switches, modem banks, and cellular base stations. This last market may be TI's best bet. Digital PCS (personal communications service) is just taking off, and its dependence on smaller "microcells" should spur demand for compact but powerful DSPs. Replacing racks of discrete DSPs with a few 'C6x processors could make such equipment smaller, cooler, and more reliable.

Crucial to the acceptance of the 'C6x will be the software tools. TI needs to deliver excellent tools and tolerate some delays while DSP programmers overcome the emotional hurdle of writing high-performance DSP code in C. If the company can execute well here, the 'C6x should perform well in some new high-end applications. M

Harri Hakkarainen is an author of Berkeley Design Technology's Buyer's Guide to DSP Processors, *the 1997 edition of which will be available from MicroDesign Resources in April.*