# StarCore Reveals Its First DSP

## *Six-Issue VLIW Core Can Execute 1.2 Billion MACs/s, 3,000 MIPS*

*by Tom R. Halfhill*

Like Texas Instruments and Analog Devices (ADI), the Motorola-Lucent alliance known as StarCore is betting on the Great Wide Hope: VLIW. StarCore's new SC140 is the third recent DSP architecture to apply long instruction words and a wide-issue core to the challenge of delivering more instruction-level parallelism.

In the latest high-end DSPs, function units and MAC pipelines are sprouting faster than new metropolitan area codes. The SC140 has 16 function units, including four MAC units that can execute 1.2 billion multiply-accumulate instructions per second at the core's target frequency of 300 MHz. In all, the fixed-point SC140 can execute six instructions per cycle—equivalent to 10 basic operations—for a blazing raw score of 3,000 MIPS.

StarCore also boasts that the SC140 will have denser code, a "compiler-friendly" programming model, and much less power consumption than comparable DSPs. If StarCore delivers on all those promises, the SC140 will significantly alter the competitive landscape in the high-end DSP market.

But that's what it will take to compete with TI's popular 'C6x family (see MPR 2/17/97, p. 14) and ADI's future TigerSharc line (see MPR 12/7/98, p. 12). Both DSP architectures use wide-issue cores and VLIW to raise parallelism to new heights—er, widths. All three DSP architectures also bear some resemblance to the VLIW media processors from Philips and Equator, as well as to Intel's IA-64. VLIW pioneers Multiflow and Cydrome (see MPR 2/14/94, p. 18) might say "I told you so"—if they were still in business.

The SC140 is the first member of StarCore's SC100 DSP family (formerly known as the SC400; see MPR 10/26/98, p. 22). It's a core, not a chip; Lucent and Motorola will each design its own DSPs by adding on-chip memory, peripherals, and external buses. This dual-sourcing arrangement is another of StarCore's selling points, because virtually all DSP architectures tie customers to a single supplier. It doesn't necessarily mean that both partners will offer the same pin-compatible parts, however. Only the core architecture will be the same. Indeed, it's likely they won't compete against each other directly.

### Optimized for Communications

Not surprisingly, considering its Lucent-Motorola parentage, the SC140 is highly optimized for communications. It has lavish resources for bit-twiddling operations and even some instructions for Viterbi algorithms. As Figure 1 shows, there are four MAC units, four ALUs, four BFUs (bit-field units), two AAUs (address arithmetic units), one BMU (bit-manipulation unit), and a BRU (branch unit). There are also 51 program-visible registers: 27 address registers (32 bits wide), 16 general-purpose data registers (40 bits wide), and 8 loop registers (32 bits wide).

There are two 32-bit address buses that access a single 4 GB address space for instructions and data. Despite its unified address space, the SC140 still maintains separate program and data buses for high bandwidth. The program bus is 128 bits wide, which allows the SC140 to fetch six instructions per clock cycle. There are two 64-bit data buses, providing 4.8 GBytes/s of peak bandwidth at the target frequency of 300 MHz.

Each of the SC140's four MAC units can perform a 16 × 16 → 40-bit multiply-accumulate operation in a single cycle, or 1,200 million MACs (MMACs) at 300 MHz. All of the MAC units are identical, so they can handle any combination of four MAC instructions simultaneously. The ALUs and BFUs are equally orthogonal.

To keep those resources productively occupied, the SC140 can issue up to six instructions plus a branch on each clock. That can include four MACs and two MOVEs. By counting a MAC as two operations (multiply and add), the SC140 can execute the equivalent of 10 instructions per cycle, which yields the sky-high 3,000-MIPS figure.

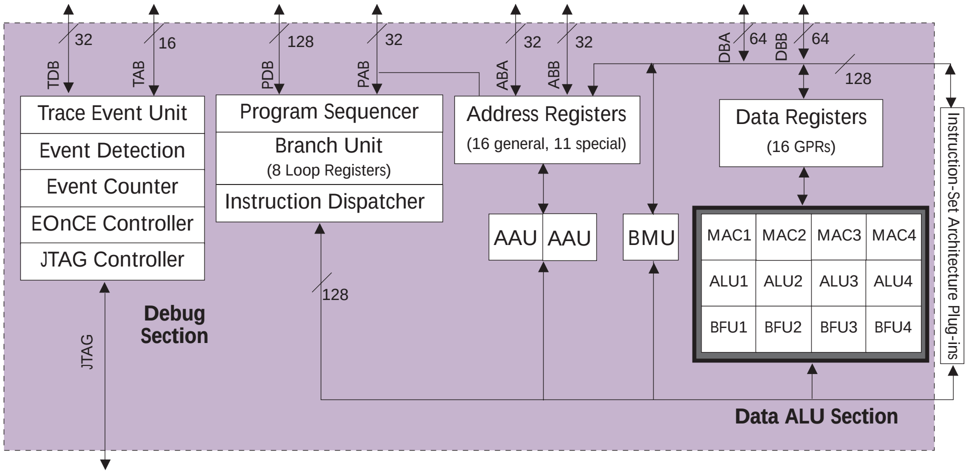

**Figure 1.** With 16 execution units—including four multiply-accumulate units, four bit-field units, and a single-bit manipulation unit—the SC140 DSP core is well suited to communications.

It's fair to compare that figure with TI's rating of 2,000 MIPS for the 250-MHz 'C6202 (the fastest 'C6x-series DSP), because TI is counting operations in the same way. In fact, the 'C6x DSPs perform a MAC by issuing separate multiply and add instructions to different function units. In this manner, a 'C6x can perform two 16 × 16 → 32-bit MACs per cycle, for a peak rate of 500 MMACs at 250 MHz.

ADI says a TigerSharc DSP will issue up to four instructions per cycle, including two 32 × 32 → 80-bit MACs or (by using SIMD operations) eight 16 × 16 → 40-bit MACs. At ADI's target frequency of 250 MHz, that would yield 4,000 MIPS or 2,000 16-bit MMACs. As Table 1 shows, that's twice as many MACs as the SC140 and four times as many as the 'C6x. Moreover, TigerSharc DSPs will perform both fixed-point and floating-point operations, while the SC140 is strictly a fixed-point device. In this category at least, TigerSharc appears to have a meaner bite.

## Clock Speeds Are Comparable

In the clock-speed race, the 300-MHz SC140 should be neck and neck with its competitors when it ships. TI is currently sampling the 250-MHz 'C6202 in a 0.18-micron five-layer-metal process; production is scheduled for 3Q99. ADI plans to sample the first 250-MHz TigerSharc in the middle of this year. That makes StarCore's target of 300 MHz look pretty good, but keep in mind that samples won't appear until 4Q99. By the time SC140 chips enter production in 2000, the competition plans to be faster too. TI claims it will deliver a 5,000-MIPS member of the 'C6x family in 2000.

StarCore's target frequency assumes production in Motorola's HIP-6 or Lucent's COM2 process. Both are 0.18-micron processes; HIP-6 has six layers of copper interconnects (see MPR 9/14/98, p. 1). To date, Motorola has produced only a few samples in this process (not SC140s). Based on those samples and models, StarCore predicts the SC140 core will consume about 198 mW at 1.5 V and 300 MHz, or 28 mW at 0.9 V and 120 MHz.

Those power-consumption figures look impressive, but they're only for the core—they don't include the power consumed by internal memory or on-chip peripherals, which are significant features of DSPs in this class. Until Lucent and Motorola announce SC140-based chips, it's not clear how much SRAM or how many peripheral functions the devices will integrate.

| | StarCore | TI | ADI |
|---|---|---|---|
| Feature | SC140 | 'C6202 | TigerSharc |
| Suppliers | Lucent, Moto | TI | ADI |
| Architecture | VLIW | VLIW | VLIW |
| Data Types | Fixed | Fixed* | Fixed & Float |
| Instr Length | 16–48 bits | 32 bits | 32 bits |
| VLIW Length | 96–128 bits | 32–256 bits | 32–128 bits |
| Execution Width | 6 instructions | 8 instructions | 4 instructions |
| Pipeline Depth | 5 stages | 11 stages | 8 stages |
| Clock Speed | 300 MHz | 250 MHz | 250 MHz |
| 16b MACs/s | 1,200 million | 500 million | 2,000 million |
| Raw MIPS | 3,000 | 2,000 | 4,000 |
| Power (Core) | 198 mW (est) | 390 mW (est) | Unknown |
| Production | 2000 | 4Q99 | 2H99 |

**Table 1.** All three of the VLIW-based DSP architectures are similar, but TI has been shipping for more than a year. *TI also offers the 'C67xx floating-point DSP. (Source: vendors)

For comparison purposes, consider the 'C6202, which is richly endowed in this respect: it has two 128K blocks of program memory and two 64K blocks of data memory. (Programmers can reconfigure one of the 128K blocks as a direct-mapped cache.) TI says the 1.8-V core consumes about 560 mW at 250 MHz, and the whole chip consumes about 1.9 W. TI estimates that a 1.5-V version of the core, scheduled for production in 4Q99, will dissipate about 390 mW. In the unlikely event that TI and StarCore are measuring power consumption in exactly the same way, the SC140 core appears to consume about half as much wattage as a 'C6202 core, even at a 20% higher clock frequency. But comparisons are iffy, because they're based on simulations, models, incomplete chips, and immature fabrication processes.

Another unknown is whether the SC140's pipeline will affect its ability to scale to higher clock speeds. Whereas the 'C6x chips have an 11-stage pipeline and TigerSharc DSPs have eight stages, the SC140 pipeline has only five stages. In general, longer pipelines permit higher clock frequencies. If the SC140 can keep pace with the superpipelined 'C6x in comparable processes, it will be a credit to StarCore's circuit-design skills.

Of course, a short pipeline also means the SC140 should suffer from fewer pipeline bubbles after a taken branch than 'C6x and TigerSharc chips. But TigerSharc has branch prediction, while the SC140 does not. In this version of the architecture at least, StarCore is trading off the complexity of branch-prediction logic by expecting smart tools (or smart programmers) to prevent the worst attacks of pipeline indigestion.

One factor in the SC140's favor is the uniformity of its instruction set: most operations have a single-cycle latency. Specifically, all ALU instructions and most moves to internal memory take one cycle. (Some moves have more complex addressing modes that require two cycles.) All bit-mask instructions and read-modify-write instructions execute in two cycles on registers and on internal memory when using simple addressing modes. Branches, with a few exceptions, take four cycles. The 'C6x instruction set is less uniform: adds, multiplies, loads, and branches all have different latencies (one, two, four, and five cycles, respectively).

The SC140's bandwidth to external memory will depend on the buses that Lucent and Motorola wrap around the core. The 250-MHz 'C6202 has a 125-MHz 32-bit memory bus that transfers 500 MBytes/s. ADI says TigerSharc chips will do 800 MBytes/s.

## Don't Call Me VLIW

StarCore prefers to describe its flavor of VLIW as variable-length execution sets (VLES), just as Intel prefers explicitly parallel instruction computing (EPIC) for IA-64. Part of the reason, no doubt, is that StarCore doesn't want to be associated with Cydrome and Multiflow—the VLIW pioneers that caught too many arrows in their backs. But there is some justification for StarCore's name change, because VLES is an amalgam of CISC, RISC, and classic VLIW.

The basic length of an SC140 instruction is 16 bits. Some instructions, however, can have one or two 16-bit prefixes to make room for predication flags and additional operands. This is an attempt to achieve the code density of CISC without sacrificing the extended register addressing of longer RISC instructions or the conditional execution that seems to be a hallmark of new architectures. By grouping up to six variable-length instructions into a long instruction word, the SC140 mixes all of those philosophies with VLIW.

As a result, SC140 instructions can be 16, 32, or 48 bits long, and instruction bundles can vary from 96 to 128 bits. The limit is 128 bits, because a single bundle can have no more than two instruction prefixes. (These limits apply only to the SC140 core. Future members of the SC100 family may allow different numbers of instructions and prefixes per bundle.)

Because of its short instructions, SC140 binaries should be only a little larger than ARM binaries in Thumb mode and about half the size of 'C6x binaries, according to StarCore's estimates. The former comparison is more flattering than the latter. The 'C6x and TigerSharc DSPs are at a disadvantage, because their instructions are always 32 bits long.

StarCore's spin on VLIW is based on the assumption that control code accounts for about 80% of a typical DSP program yet consumes only 20% of the execution time. Conversely, the actual signal-processing code accounts for about 20% of the program but runs 80% of the time. SC140 programs can use simple 16-bit instructions for the mundane control code and reserve the more complex 32- and 48-bit instructions for the critical DSP code. About 82% of the basic 180 instructions have 16-bit forms; only about 15% can stretch to 48 bits.

By allowing both the instructions and their bundles to vary in length, the SC140 offers two advantages over the more rigid forms of VLIW: code compatibility and code density. In the classic version of a statically scheduled VLIW machine, the position of each instruction in a bundle maps directly to the arrangement of function units. If the compiler can't find an instruction that can execute in parallel with the others, it must fill that instruction slot with a NOP. Such tight coupling between the code and the core minimizes complexity, but any changes to the microarchitecture will break the software.

All three of the VLIW DSP architectures—the SC140, the 'C6x, and TigerSharc—avoid this problem because their instruction bundles can vary in size and the instructions within the bundles are relatively position independent. If the compiler can't fill every slot in a bundle with a parallel instruction, it can truncate the bundle. In other cases, groups of parallel instructions can span multiple bundles. There's no absolute dependence between the configuration of the bundles and the microarchitecture. Therefore, binaries should be portable (though not necessarily optimized) across all members of the family.

This leads to the SC140's second advantage: code density. Variable-length bundles don't need to be fattened with the empty calories of NOPs. This is a big improvement over classic VLIW, and the memory saved is especially important for an embedded processor. What sets the SC140 apart in this comparison is that its instructions, not just the bundles, can vary in size. The basic instruction length of 16 bits should result in smaller programs than the 32-bit instructions of the 'C6x and TigerSharc DSPs.

### Price & Availability

StarCore partners Lucent and Motorola plan to sample their first DSPs based on the SC140 core in 4Q99. Production schedules and pricing will vary, depending on what each partner adds to the basic core. Lucent and Motorola can add different amounts of memory, various on-chip peripherals, and their own I/O buses.

For more information about StarCore, go to *www.lucent.com/micro/starcore/*.

## From C to Shining C

One downside of the StarCore architecture (and those of other VLIW DSPs) is the difficulty of programming it in assembly language. The SC140 is a statically scheduled processor with 16 function units, optional instruction prefixes, and variable-length instruction bundles. It has about 180 instructions, not counting the numerous addressing variations. (Table 2 shows a condensed version of this rich instruction set.) The assembly-language programmers who can quickly grok this architecture should be easy to find—they're the ones whiling away their lunch hours finding alternate solutions to Fermat's Last Theorem. Mere mortals will seek refuge in compilers.

StarCore predicts that SC140 programmers will write about 90% of their code in C or C++, reserving inline assembly for tight DSP loops. StarCore is showing off some loops generated by an early compiler and assembly-code optimizer, but it remains to be seen how well these tools work in practice. The combination of short pipelines and relatively uniform instruction latencies should make creating these tools easier—at least when compared with the much longer pipeline and more-variable latencies of the 'C6x.

Developers can choose from several tool chains. StarCore is providing baseline tools, including a C compiler, a linker, an assembler, an assembly-language optimizer, and a simulator. Lucent and Motorola will each offer its own higher-level tools, such as integrated development environments (IDEs) and symbolic debuggers.

| Mnemonic | Description | Size (bits) | Latency (cycles) | W | S/U | Mnemonic | Description | Size (bits) | Latency (cycles) | W | S/U |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDL1A | Add, shift left 1 | 16 | 1 | | | INSERT | Insert bit field, integer | 32 | 1 | | |
| ADDL2A | Add, shift left 2 | 16 | 1 | | | LSLL | Bidirectional shift left | 16 | 1 | | |
| ADR | Add & round | 16 | 1 | | | LSR | Logical shift right | 16 | 1 | • | |
| ASL | Shift left 1 bit | 16 | 1 | • | | LSRA | AGU shift right | 16 | 1 | | |
| ASL2A | Shift AGU left 2 bits | 16 | 1 | | | LSRR | Shift right (multiple) | 16 | 1 | | |
| ASLA | Shift AGU left 1 bit | 16 | 1 | | | MAC | Multiply-accumulate | 16–32 | 1 | | |
| ASLL | Shift left, multiple bits | 16 | 1 | | | MACR | MAC & round, fractional | 16 | 1 | | |
| ASR | Shift data right 1 bit | 16 | 1 | • | | MACSU | Fractional MAC | 16 | 1 | | • |
| ASRA | Shift ALU right 1 bit | 16 | 1 | | | MAX | Find larger value | 16 | 1 | | |
| ASRR | Shift data right, multiple | 16 | 1 | | | MAX2 | Find larger value, hi/lo | 16 | 1 | | |
| BMCHG | Bit-masked change | 32 | 2 | • | | MAX2VIT | Viterbi find larger value | 16 | 1 | | |
| BMCLR | Bit-masked clear | 32 | 2 | • | | MAXM | Find larger absolute | 16 | 1 | | |
| BMSET | Bit-masked set | 32 | 2 | • | | MIN | Find smaller value | 16 | 1 | | |
| BMTSTC | Bit-masked test | 32 | 2 | • | | MPY | Multiply fractional | 16 | 1 | | • |
| BMTSTS | Bit-masked test (16b) | 32 | 2 | • | | MPYR | Multiply & round | 16 | 1 | | |
| CLB | Count leading bits | 16 | 1 | | | RND | Round data reg | 16 | 1 | | |
| CMPEQ | Compare for equality | 16 | 1 | • | | ROL | Rotate left | 16 | 1 | | |
| CMPEQA | Compare AGU regs | 16 | 1 | | | ROR | Rotate right | 16 | 1 | | |
| CMPGT | Compare for > | 16 | 1 | • | | SAT.F | Saturate fraction | 16 | 1 | | |
| CMPGTA | Compare for > (AGU) | 16 | 1 | | | SAT.L | Saturate long word | 16 | 1 | | |
| CMPHI | Compare high bits | 16 | 1 | | | SBC | Saturate with carry | 16 | 1 | | |
| CMPHIA | Compare AGU high bits | 16 | 1 | | | SBR | Subtract & round | 16 | 1 | | |
| DMACSS | Double MAC, shift right | 16 | 1 | | • | SUBL | Shift left & subtract | 16 | 1 | | |
| EXTRACT | Extract bit field | 32 | 1 | | • | SXT.x | Sign extension | 16 | 1 | | |
| IFc | Conditionally execute set | 16–32 | 1 | | | SXTA.x | Sign extension, AGU | 16 | 1 | | |
| ILLEGAL | Execute illegal exception | 16 | 5 | | | VTL | Viterbi shift left | 16 | 1 | | |
| IMAC | Integer MAC | 16 | 1 | | • | ZXT.x | Zero extension | 16 | 1 | | |
| IMPY | Integer multiply | 16 | 1 | • | • | ZXTA.x | Zero extension (AGU) | 16 | 1 | | |

**Table 2.** This table shows about one-third of the SC140's instruction set. The large variety of bit-manipulation and compare instructions makes the SC140 especially suitable for communications—there are even two instructions for Viterbi algorithms. (A bullet in the W column means an instruction has variations that operate on word-sized quantities. A bullet in the S/U column means an instruction has variations that operate on signed and unsigned quantities.)

A third-party supplier, Green Hills Software, plans to ship an SC140 version of its well-known Multi IDE at the end of this year. This product will include a C/C++ compiler, DSP data-visualization utilities, a cycle-accurate SC140 simulator, and hooks for other analysis tools, such as MatLab.

High-level tools, not architectural differences, may determine which of the high-end DSPs delivers the best real-world performance. Architecturally, all of these DSPs are in the same class, and all pose similar difficulties for assembly-language programmers. Few desktop/server CPUs are as complex, yet the programmers who are writing code for mainstream CPUs have for years been using high-level languages almost exclusively. Furthermore, nearly all mainstream CPUs can compensate for a programmer's shortcomings to some degree by dynamically reordering the instructions at run time.

Not so for these statically scheduled VLIW DSPs. Poorly written or poorly compiled code effectively turns them into low-performance scalar processors. That's why high-level tools will make a significant competitive difference.

## Experience Matters

TI is the front runner in this contest, and not just because the company has the largest DSP market share. The 'C6x was the first high-end DSP architecture to use VLIW. It has been shipping for more than a year, and currently there are three fixed-point and two floating-point devices in the line. TI developed its tools in-house, with close collaboration between the compiler writers and DSP architects. It was the first DSP vendor to offer an assembly-language optimizer that's closely matched to the architecture. All this gives TI a significant lead that competitors are hard pressed to match.

To develop comparable tools, StarCore is drawing upon the resources of both alliance companies plus a third-party vendor. This approach should foster more competition and give customers more choices, although it remains to be seen whether better tools will result. Likewise, the chip-level competition between Lucent and Motorola should also give customers the luxury of multiple suppliers for the StarCore architecture, if not for individual chips.

The SC140's shorter instructions should give it an advantage in code density over the 'C6x and TigerSharc DSPs. If StarCore's power-consumption projections are in the ballpark, the SC140 will be more suitable for mobile applications as well.

All of those advantages will help offset TI's head start. If Lucent and Motorola can build on this solid foundation with competitive implementations, they will secure a strong position in the growing market for high-end DSPs. M