Siemens TriCore Revives CISC Techniques

New 32-Bit Design Emphasizes DSP Capability and Microcontroller Functions

by Jim Turley

Fashions run in cycles, and CPU designs abandoned in the '70s are again becoming chic. As evidence of this trend, Siemens rolled out TriCore, a retro-CISC architecture that combines microcontroller, DSP, and CPU features while simultaneously attempting to shrink code size and improve speed.

TriCore throws out nearly everything that RISC microprocessor design has taught in the past ten years. At last month's Microprocessor Forum, TriCore's chief architect, Rod Fleck, described a mixture of 16- and 32-bit instructions, separate address and data registers, multicycle instructions, multiple data types, complex interrupt handling, and a lopsided instruction set geared toward bit-twiddling.

Paradoxically, TriCore also includes some of today's newest thinking in embedded controllers. It handles digital signal processing (DSP), zero-overhead loops, and SIMD packed data types, and relies on a 128-bit path to local embedded DRAM. The first TriCore chips aren't expected to sample until 2Q98, but by the end of next year, Siemens hopes to make a dent in automotive and computer-peripheral markets with its unusual new family.

Register Set a Big Step Backward

TriCore includes 32 registers, plus a few control/status registers. Where TriCore differs from most recent microprocessors is its split between address and data registers. As Figure 1 shows, only 16 of the registers are used for handling data; the other half are address pointers.

TriCore's register set is further divided into quarters. Siemens segregates the registers into an "upper context" and a "lower context." When switching tasks, calling functions, or handling interrupts, only one context is saved or restored.

TriCore's lower context consists of eight data registers (D0–D7) but only six of the lower eight address registers (A2–A7). The other two address registers, A0 and A1, are global resources that are not saved or restored as part of a context switch. The global program counter, PC, is also considered part of the lower context. The upper context consists of the other eight data registers (D8–D15), six address registers (A10–A15), and PSW. Like A0 and A1, A8 and A9 are global address pointers.

TriCore's subdivided (and not very general) register set is a throwback to the CISC microcontrollers that Siemens knows so well. While the design might not be architecturally elegant, Fleck believes that it makes sense in the grubby world of real-life control code. The logical split allows Siemens to implement a physical split, placing the addresses and data registers in different areas of the chip, alleviating port congestion and simplifying routing.

Linked List Gives Fast Interrupt Response

Interrupts, traps, and function calls automatically save the upper context to on-chip memory, giving the trap handler, interrupt-service routine, or called function a clean set of registers with which to work. Software can save the lower context as well, if desired, with explicit instructions. TriCore automatically restores the upper context when resuming the interrupted task. Saving and restoring the upper or lower context takes just four clock cycles because of TriCore's wide 128-bit bus between the register file and dual 2K caches.

Saved contexts are stored in a linked list that TriCore chips maintain automatically. Each context store needs 64 bytes for the 15 registers plus a 32-bit link pointer to the next free context area. Internal head and tail pointers (in PCXI) allow TriCore to quickly locate the next (or previous) state.

As a safety net, TriCore also maintains a call-recursion counter for every task. The counter is incremented on every function call and restored to its previous state (rather than decremented) on every return. If the counter overflows, an exception occurs. Programmers can control the width of the counter (in bits), setting their tolerance factor for runaway recursion.

Instructions an Eclectic Mix

TriCore executes both 16- and 32-bit instructions, which may be freely intermixed. Each opcode includes a size bit, so TriCore's instruction decoder can identify long or short instructions immediately. The 16-bit instructions form a

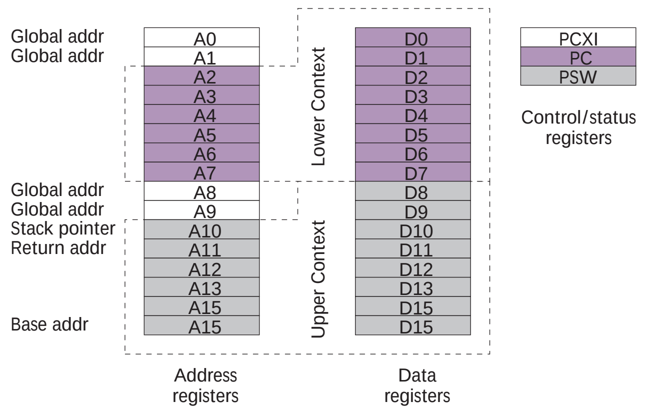


Figure 1. TriCore separates its address and data registers; upper and lower halves, or contexts, are saved and restored separately.

subset of existing 32-bit instructions that sacrifices some flexibility, so in many cases a 32-bit instruction can be replaced with a shorter form to save code space. Unlike Thumb or MIPS-16, TriCore switches between 16- and 32-bit instructions without explicitly switching modes, saving a little in execution time and avoiding the need to segregate compressed and uncompressed code.

TriCore follows a basic load/store model, but the similarity to RISC chips ends there. The instruction set is quite rich, with most of the emphasis on numerical processing and integer DSP work. As Table 1 shows, TriCore has seven different forms of addition, for example, and a dozen different multiply-accumulate instructions. Far from being general purpose, TriCore was developed for motion control, signal processing, and compression/decompression work.

Most arithmetic operations can operate on bytes, halfwords (16-bit quantities in the Siemens argot), words, doublewords, and so-called Q-format numbers. Parallel (SIMD) byte and halfword operations are richly supported.

The Q15 and Q31 data types are simplified fractional representations, with one sign bit followed by 15 (or 31) bits of significance after the implied binary point. The advantage of Q-format over IEEE-754 floating point is that it is far easier to implement in silicon. Left-justified Q15 and Q31 numbers can be added by normal addition instructions without compromising precision. On the other hand, Q-format numbers do not have the dynamic range or flexible precision of true floating-point numbers. For many control systems, Q-format will do the trick with low hardware cost. For applications that require true floating-point math, however, Q-format won't cut it.

Strong Parallels and Bit Wise

In addition to the usual operations, TriCore can perform integer or Q-format arithmetic on packed data values. In a manner similar to MMX, two halfword values or four byte values can be added, subtracted, multiplied, multiply-added, or multiply-subtracted. The ADD.B instruction, for example,

Mnemonic	Description	Mnemonic	Description	Mnemonic	
Arithmetic		**Load/Store**		**Multiply/Divide**	
ADD(S)	Add (with saturation)	MOV	Copy register	MUL	Multiply 32 × 32 → 32
ADDC	Add with carry	LD	Load	MULS/R	… with saturation/rounding
ADDI	Add immediate value	ST	Store	MULM	Multiply 32 × 32 → 64
ADDIH	Add immediate to high half	LEA	Load effective address	MADD	Multiply-add 32 bits
ADDSC	Add scaled value	LDLCX	Load lower context	MADDS/R	… with saturation/rounding
ADDX	Add and generate carry	LDUCX	Load upper context	MADDRS	… with saturation and rounding
SUB(S)	Subtract (with saturation)	STLCX	Store lower context	MADDM(S)	Multiply-add 64 bits (with sat.)
SUBC	Subtract with borrow	STUCX	Store upper context	MSUB	Multiply-subtract 32 bits
SUBX	Subtract, generate borrow	SVLCX	Save lower context	MSUBS/R	… with saturation/rounding
RSUB(S)	Reverse subtract (with saturation)	RSLCX	Restore lower context	MSUBRS	… with saturation and rounding
ABS(S)	Absolute value (with saturation)	LDMDST	Load, modify, store	MSUBM	Multiply-subtract 64 bits
ABSDIF(S)	Absolute value difference (sat.)	MFCR	Move from special register	MSUBMS	… with saturation
DIFSC	Difference scaled addresses	MTCR	Move to special register	DVADJ	Adjust after division
CLO/CLZ	Count leading ones/zeros	SWAP.A	Swap with address register	DVINIT	Prepare for division
CLS	Count leading signs	**Comparison**		DVSTEP	Division step
MIN/MAX	Find minimum/maximum	EQ	Compare, equal	**Bit Manipulation**	
SAT	Saturate operand	NE	Compare, not equal	DEXTR	Extract from doubleword
Logical		GE	Compare, greater equal	IMASK	Create mask word
AND/OR	Logical AND/OR	LT	Compare, less than	INS	Insert single bit
NAND	Logical NAND	EQANY	Compare for equality	INSERT	Insert bit field
NOR	Logical NOR	EQZ	Compare address for zero	EXTR	Extract bit field
XOR	Logical exclusive-OR	**System**		**Flow Control**	
XNOR	Logical exclusive-NOR	ENABLE	Enable interrupts	J	Jump, unconditional
ANDN	Logical AND 1's comp.	DISABLE	Disable interrupts	JA/JI	Jump, absolute/indirect address
ORN	Logical OR 1's complement	DSYNC	Force pending data accesses	Jcc	Jump on condition cc
NOT	Logical invert	ISYNC	Flush execution pipeline	JL	Jump and link
SH/SHA	Logica/arithmetic shift	RSTV	Reset overflow flags	JLA/JLI	… absolute/indirect
SHAS	Arithmetic shift with saturation	SYSCALL	Force trap	LOOP	Initiate loop
Conditional		TRAPV/SV	Trap on overflow	BISR	Begin ISR
CADD(N)	Conditional add (1's comp.)	DEBUG	Enter debug mode	CALL	Call subroutine
CSUB(N)	Conditional subtract (1's comp.)	NOP	No operation	CALLA/I	Call absolute/indirect
CMOV(N)	Conditional move (1's comp.)			RET	Return from subroutine
SEL	Select operand	▨	16-bit instruction word	RFE	Return from exception

Table 1. TriCore has an unusually rich instruction set, including several 16-bit instructions (shaded). TriCore's designers paid particular attention to multiplication, accumulation, and reverse addition and subtraction—all functions used in digital-signal processing.

adds the four bytes from two registers and deposits four results in another register. As with most arithmetic operations, saturating, nonsaturating, signed, and unsigned variations are all available.

Parallel logical operations and comparisons are also supported. The EQANY.B instruction, for example, compares each byte from two registers and sets the LSB in the destination register if any of the four comparisons is equal. An alternate form compares all four bytes against a constant.

Moving beyond simple arithmetic, TriCore implements a bewildering combination of multiply, multiply-add, and multiply-subtract instructions with options for saturation, rounding, sign extension, packed data, integer or Q-format data types, alignment, and addressing modes. These, plus the zero-overhead LOOP, form the heart of TriCore's DSP capability. Combined with bit-reverse and circular memory addressing, TriCore will become a creditable digital-signal processor, Siemens believes. Simulations show the chip can sustain an FIR filter at two taps per cycle, twice the throughput of either Motorola's 56300 or TI's popular 'C54x parts.

Well beyond the ken of most microprocessors are TriCore's bit-manipulation instructions, such as INSERT, INS, and EXTR. The first two insert a single bit, or a set of contiguous bits, into any location in another register; the last instruction extracts any number of bits from any location in any register and copies them into another register, with either sign- or zero-extension to 32 bits. For I/O control and network addressing, among other applications, these operations are valuable.

The CLO, CLZ, and CLS instructions count leading ones, zeros, or sign bits, respectively. The variants CLx.H and CLx.B are interesting because they count leading bits in packed bytes or halfwords, returning two or four totals. All of these operations are useful for normalization, prioritization, encryption, error correction, and for managing graphics primitives, and they are time-consuming to perform any other way.

Conditional operations abound, even though TriCore has no condition codes. Conditional adds, subtracts, moves, and branches all depend on comparing a register (always D15 in 16-bit instructions) with zero. Conditional moves CMOV and CMOVN, for example, copy the contents of a register only if D15 is zero or nonzero, respectively. SEL is like CMOV, but it copies one of two values, depending on D15. Other conditional instructions can test any data register.

Integer division comes in the form of DVINIT, DVSTEP, and DVADJ, three instructions that perform step-wise division under software control. TriCore's support for division is better than in many older CPU families that have no divide instruction at all, but for ease of use it falls short of the single-instruction divide operations in the 68K and x86. At one bit per cycle, TriCore is the same speed as SuperH, which also requires explicit divide-step instructions. TriCore's DVSTEP always works in eight-cycle chunks, though.

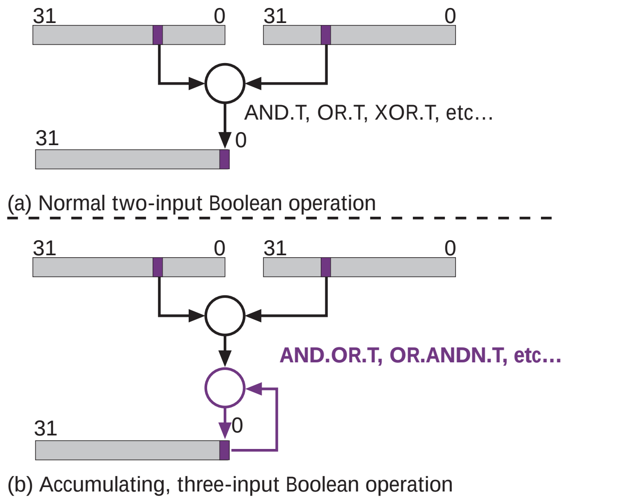

31
0
31
0
AND.T, OR.T, XOR.T, etc…
31
0
(a) Normal two-input Boolean operation
31
0
31
0
AND.OR.T, OR.ANDN.T, etc…
31
0
(b) Accumulating, three-input Boolean operation

Figure 2. TriCore's normal's two-input logical operations are complemented with a set of eight accumulating, three-input logicals.

Three-Way Logicals Tighten Code

One of the most interesting and unusual features of TriCore's instruction set is its ability to "accumulate" the results of logical (Boolean) operations. Using more than a dozen instructions for accumulating and three-way comparisons, programmers can implement complex multiway comparisons in a minimum amount of code.

As Figure 2a shows, the normal bit-wise comparison instructions, such as AND.T, OR.T, or XNOR.T, compare any two bits from any two registers and store the logical result in the least-significant bit of a third register (the rest of the destination register is cleared). All four basic Boolean operations (AND, OR, XOR, ANDN) and their opposites (NAND, NOR, XNOR, ORN) are available. Table 2 lists the accumulating logical operations.

TriCore goes a step further by allowing three-operand logicals. As Figure 2b shows, the result of a previous bit-wise logical operation can be included in a new logical operation, like a multiply-accumulate on a single bit. The logical opera-

Logical			
AND.AND.T AND.ANDN.T	OR.AND.T OR.ANDN.T	SH.AND.T SH.ANDN.T	SH.NAND.T SH.ORN.T
AND.OR.T AND.NOR.T	OR.OR.T OR.NOR.T	SH.OR.T SH.NOR.T	SH.XOR.T SH.XNOR.T
Comparison			
OR.EQ OR.NE	AND.EQ AND.NE	XOR.EQ XOR.NE	
OR.GE OR.GE.U	AND.GE AND.GE.U	XOR.GE XOR.GE.U	
OR.LT OR.LT.U	AND.LT AND.LT.U	XOR.LT XOR.LT.U	

Table 2. TriCore can perform two parallel logical operations or arithmetic comparisons simultaneously. The Boolean result can be "accumulated" with other results, allowing programmers to efficiently code sequences of comparisons and/or logical operations.

Price & Availability

The first Siemens TriCore chip (which has not been named) is expected to sample in 2Q98; production is set for 2H98. For more information, contact Siemens Microelectronics (Cupertino, Calif.) at 408.895.5004 or visit *www.sci.siemens.com/tricore.*

tions don't even have to be the same, so the first and second bits can be AND-ed while the third bit is OR-ed, for example. This construct maps well to common multipart comparisons in high-level languages and allows programmers (or compilers) to efficiently encode complex relational tests.

The comparisons need not be limited to single bits, either. Using instructions like OR.EQ, AND.LT, or XOR.GE, programs can cumulatively compare 32-bit numbers using signed or unsigned arithmetic comparisons. By accumulating a running Boolean result through multiple comparisons, TriCore programmers can avoid peppering their code with a lot of short conditional branches that bloat code size and lead to pipeline bubbles.

Instruction Set Hardly Orthogonal

TriCore is not without its peculiarities. Its split register set will generate split opinions; separate address and data registers are shunned these days. The argument follows that for split versus unified caches: some algorithms use more data, while others need more address pointers. Unifying the register file (or the cache) allows dynamic allocation of a scarce resource.

The split register file also means separate instructions for the address registers, crowding TriCore's opcode map. Instructions like ADD.A and SUB.A are needed to update address pointers; special MOV instructions move the contents of address registers around. A split in the data path also forces TriCore to keep its loop counter in an address register to relieve congestion of the data registers in filter operations.

Flow-control instructions with absolute addresses are limited to 24-bit pointers, at best. TriCore handles this by separating the four most significant bits and left-justifying them, then using the lower address bits unchanged. This technique sections TriCore's 4G address space into 16 equal-sized regions, with the lower 1M of each directly accessible to other code.

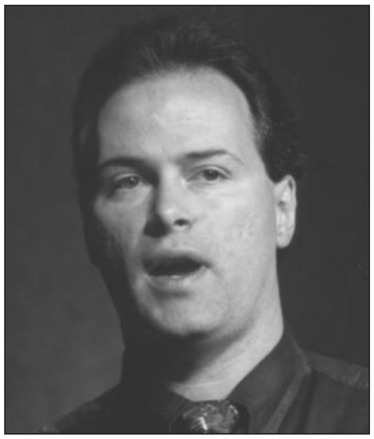
Ugly Has a Place, Too

With TriCore, Siemens has thrown out the book of accepted design principles and relied on its years of experience designing and selling microcontrollers to industrial and commercial OEMs. TriCore is different, unconventional, and sometimes awkward, but it seems to have the tools to get the job done quickly.

TriCore is only the second microprocessor architecture to be designed from the ground up around embedded DRAM (Mitsubishi's M32R/D is the other). TriCore's context-switch mechanism relies on a fast, wide connection to on-chip memory, and Siemens's Fleck says no TriCore chip will be made without it.

TriCore is also one of the few microprocessors to deal with control and signal processing in a single instruction set. ARM's Piccolo and Hitachi's SH-DSP were both grafted onto the original design and force tradeoffs in DSP-versus-controller performance. But those grafts can also be rejected if users are interested in just the core ARM or SuperH processor, a choice that Siemens doesn't offer.

Motorola's ColdFire and M•Core lines have both taken their first small steps toward signal processing, but no 32-bit processor out there can match the dizzying assortment of number- and bit-manipulation options TriCore will have. The important details, like bit-reverse and circular addressing, rounding, saturation, and sticky overflow bits, all give TriCore a much more credible claim to being a DSP than most microprocessors can make. On the downside, TriCore doesn't offer the X/Y data memories DSP programmers admire, but that don't map well to C code.


MICHAEL MUSTACCHI

Siemens CPU director Rod Fleck talks about the CPU/DSP capabilities of TriCore at the Forum.

The inevitable price for this complexity is circuit density. Siemens has not released any details of die size, transistor count, or clock speed. The first TriCore chip, with 128K of on-chip DRAM and a pair of 2K caches, has not taped out and samples aren't expected until 2Q98. That's a long time to wait before customers can make informed choices—time that Motorola, Hitachi, and untold hordes of ARM vendors can use to strengthen their leads.

There's a reason CPU designers have scrapped complex and irregular instruction sets and embraced—more or less—the principles of RISC design in recent years. Simpler architectures make simpler CPU cores, which run faster and are easier to scale to multiple execution units. During his presentation, Siemens's Fleck hinted at plans for TriCore chips with more execution units in the not-too-distant future, but with all of TriCore's complexity, that may be tough to do.

With production volumes probably coming in 1999, it's still too early to be certain about much to do with TriCore. Certainly the family will compete with ARM, ColdFire, M•Core, SuperH, and MIPS chips. But for those applications that emphasize signal processing over numerical processing, bit-twiddling over floating-point handling, and code density over software compatibility, TriCore appears well positioned to pick up a portion of new embedded designs. M