Tensilica CPU Bends to Designers' Will

Xtensa Processor Can Be Configured by Customers Before ASIC Synthesis

by Jim Turley

Proving there's no shortage of new ideas in processor design, Silicon Valley startup Tensilica has made the first public announcement of its new processor design. Tensilica's CPU core differs from most others in that it can be configured and customized by an individual ASIC developer. Tensilica bundles its basic CPU design with design tools that allow ASIC developers to create their own application-specific extensions.

Tensilica's RISC-inspired processor mixes 16-bit and 24-bit instructions, an orthogonal register file, and 78 immutable "base case" instructions. Customers are then free to add their own special-purpose instructions by defining them, using a subset of the Verilog hardware-description language. The resulting mix is then synthesized and fabricated as part of the larger ASIC design process.

Tensilica is poised to compete with better established 32-bit core vendors such as ARM and MIPS as well as fellow newcomers like Lexra and ARC Cores. Predictably, Tensilica claims its processor design is both more powerful than any of the existing processor-core alternatives and more flexible. Any performance claims are so far tough to verify, but in terms of flexibility, Tensilica may have a point.

Headed by All-Star Cast and Directors

Tensilica's org chart reads like a who's who of RISC and synthesis luminaries. The management and engineering teams include MIPS/SGI alumni Ashish Dixit, Earl Killian, Woody Lichtenstein, Dror Maydan, Chris Rowen, John Ruttenberg, and Keith van Sickle as well as Synopsys graduates Harvey Jones, Bernie Rosenthal, and Albert Wang. Beatrice Fu, late of Intel, serves as Vice President of Engineering. John Hennessey (Stanford), Kurt Kreutzer (UC Berkeley), and Monica Lam (Stanford) serve on Tensilica's technical advisory board.

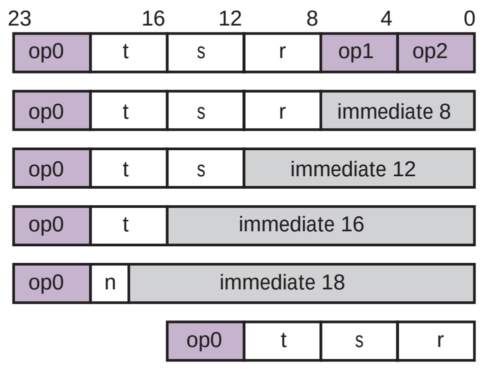

Figure 1. Xtensa instructions are encoded in either 16 or 24 bits, with the most significant bit determining word size.

The assembled combination of processor-design and synthesis-tool experience was a natural for Tensilica's goal: to push CPU design decisions down the microprocessor food chain to the ASIC designer. Rather than create yet another embedded microprocessor and offer it up for licensing, Tensilica has chosen to develop a set of tools that lets ASIC designers create their specialized microprocessor, merge it with their custom logic and memory, and synthesize the whole thing to create an ASIC.

To that end, Tensilica (the name is a play on tensile, as in flexible, silicon) created a CPU core descriptively named Xtensa. Tensilica president Rowen believes that "traditional" licensed CPU cores, such as MIPS, ARM, SPARC, and PowerPC, force the ASIC customer to "design around" a fixed processor that has been licensed to a limited number of semiconductor fabricators. The customer's algorithms and software must then be mapped onto the architecture and instruction set of that processor, a situation that has been true from the earliest days of digital computers.

A user-configurable microprocessor, the argument goes, can be customized by those who know the application's requirements and algorithms best: the customers. Rather than map the software onto a fixed instruction set, the instruction set can, to some extent, be mapped onto the algorithm.

All Processors Include 78 Basic Instructions

Looking at just the base-level Xtensa processor, one finds little to differentiate it from dozens of other RISC-inspired CPUs. One oddity is that Xtensa mixes 16- and 24-bit instruction words; there are no 32-bit instructions. The most-significant bit in each opcode identifies the instruction size, as Figure 1 shows. Most instructions specify three register operands, or two registers and an immediate (literal) value. Most operations are nondestructive, and Xtensa follows a load/store memory model. Tensilica believes Xtensa's reliance on short instruction words enhances the CPU's code density, negating the need for code compression.

The base instruction set, which all Xtensa processors will share, consists of 78 instructions, listed in Table 1. The base ISA includes the usual logical and arithmetic operations, as well as some not-so-common conditional moves and zero-overhead loop instructions. By maintaining a core set of instructions for all Xtensa processors, Tensilica can guarantee at least some level of software compatibility among different customers' implementations. This allows a single operating-system port to run on all Xtensa-based CPUs.

Register Windows Make a Return

Tensilica has revived register windowing with Xtensa, though not the same way that SPARC processors implement the feature. The register file is logically maintained as a circular queue, as in SPARC processors, though only 16 registers are visible at any one time. On subroutine calls, programmers can choose to overlap the called procedure by zero, four, eight, or twelve registers. The overlapped registers pass parameters into and out of called procedures without pushing and popping values on a stack. The Xtensa C compiler automatically selects the maximum amount of overlap needed to preserve static data.

With no register overlap, Xtensa's 32-register file can maintain two concurrent processes without overflowing. (Developers can optionally double the number of registers, to 64, with a synthesis option.) Specifying a nonzero register overlap increases the potential number of concurrent processes. When the number of physical registers is exceeded, Xtensa spills 4, 8, or 12 registers to an external stack. The physical register file is refilled when code executes an RETW to a call frame that is not in the register file.

Register windows have never been a popular architectural feature because they are either huge, awkward to maintain, and take a long time to switch contexts (as in SPARC), or they are so small that they provide little benefit. Windowed registers are a bit more useful in embedded systems, which generally have few (or no) independent tasks or contexts. The shallower call/return stack of embedded systems makes small circular register files (as in PicoJava, TriCore, and the PSC1000) more fruitful.

Designer-Defined Instructions

The real charm of Xtensa is not its basic architecture but its extensibility. Tensilica allows ASIC designers to extend Xtensa's instruction set using a software tool called TIE (Tensilica instruction extension language), a subset of Verilog. Customers write TIE scripts that define the mnemonic, the binary opcode, and the input and output functions of the new instruction. Tensilica's synthesis tools then automatically create the decoding logic required to implement the new instruction.

Tensilica's flexibility is not total. There are significant limits to the changes a customer can make. All Xtensa processors start with the same hardware resources (ALU, multiplier, data paths, etc.) and the same basic set of 78 instructions. To this baseline architecture, customers can add an arbitrary number of new instructions (at least, until they deplete the unused opcodes).

Mnemonic	Description	Mnemonic	Description	Mnemonic	Description
Arithmetic		**Conditional Branches**		**Data Transfer**	
ADD	Add	BEQ	Branch if equal	L8UI	Load 8 bits, unsigned
ADDI	Add immediate	BEQI	Branch if equal, immediate	L16SI	Load 16 bits, signed
ADDX2	Add, shift by 1	BEQZ	Branch if equal to zero	L16UI	Load 16 bits, unsigned
ADDX4	Add, shift by 2	BNEZ	Branch if not equal to zero	L32I	Load 32 bits
ADDX8	Add, shift by 3	BGE	Branch if greater/equal	L32R	Load 32 bits, PC-relative
ADDMI	Add immediate, shift by 8	BGEI	Branch if greater/equal, immediate	S8I	Store 8 bits
SUB	Subtract	BGEU	Branch if greater/equal, unsigned	S16I	Store 16 bits
SUBX	Subtract, shift by 1	BGEUI	Branch if greater/equal, uns, imm	S32I	Store 32 bits
SUBX4	Subtract, shift by 2	BGEZ	Branch if greater/equal to zero	MOVEQZ	Move if equal to zero
SUBX8	Subtract, shift by 3	BLT	Branch if less than	MOVGEZ	Move if greater/equal to zero
Shift and Logical		BLTI	Branch if less than, immediate	MOVI	Move immediate
AND	Logical AND	BLTU	Branch if less than, unsigned	MOVLTZ	Move if less than zero
OR	Logical OR	BLTUI	Branch if less than, uns, immediate	MOVNEZ	Move if not equal to zero
XOR	Logical exclusive-OR	BLTZ	Branch if less than zero	**Flow Control**	
NEG	Negate	BNE	Branch if not equal	CALL0	Call, zero register overlap
SLL	Logical shift left	BNEI	Branch if not equal, immediate	CALLX0	Call via register
SLLI	Logical shift left, immediate	BALL	Branch if all bits set	RET	Return from CALL
SRA	Arithmetic shift right	BNALL	Branch if not all bits set	J	Jump, unconditional
SRAI	Arithmetic shift right, immediate	BANY	Branch if any bits set	JX	Jump via register
SRC	Shift right through carry	BNONE	Branch if no bit set	**Miscellaneous**	
SRL	Logical shift left	BBC	Branch if bit clear	RSR	Read special register
SRLI	Logical shift left, immediate	BBCI	Branch if bit clear, immediate	WSR	Write special register
SSA8B	Set shift amount, little-endian	BBS	Branch if bit set	DSYNC	Synchronize load/store
SSA8L	Set shift amount, big-endian	BBSI	Branch if bit set, immediate	ESYNC	Serialize execution
SSAI	Set shift amount, immediate	LOOP	Loop	ISYNC	Synchronize fetch
SSL	Set shift amount for left shift	LOOPGTZ	Loop if greater than zero	RSYNC	Register read, synchronized
SSR	Set shift amount for right shift	LOOPNEZ	Loop if not equal to zero	EXTUI	Extract unsigned immediate

Table 1. All Xtensa processors share a baseline instruction set of 78 instructions, which users can extend using a Verilog-like compiler. The base set includes normal arithmetic and logical operations but no multiply-accumulate, media, or SIMD operations.

Price & Availability

Tensilica's Xtensa microprocessor generator is available now. Licensing fees start at $250,000 for Web-based access to the tools and a manufacturing license. Royalties are negotiable.

For more information, contact Tensilica (Santa Clara, Calif.) at 408.986.8000 or visit *www.tensilica.com.*

There are only a few basic limitations on new instructions: they must be encoded in either 16 or 24 bits; they must execute in a single cycle; and they must access only register-based operands. All new instructions will, by definition, require additional hardware resources. If the designer can describe the necessary hardware function in TIE, the synthesis tools will generate it as necessary.

Within these boundaries, users are free to develop their own special-purpose instructions. Rowen cites common examples of arithmetic minimum, maximum, or sum-of-absolute-differences instructions. Users can also create their own simple SIMD instructions that, for example, sum two eight-bit operands in parallel. This would be implemented with a new adder that has a segmented carry chain, allowing two unrelated addition operations at once.

Predefined Options Provide a la Carte Selection

Tensilica has already defined about two dozen prepackaged options for the basic Xtensa core. Option packages consist of a few specialized instructions and the additional hardware resources needed to support them. Some examples add support for prioritized interrupts, timers, a 32-bit multiply/divide unit, a floating-point coprocessor, a larger register file, debug visibility, or increased code density.

Many of these options are roughly analogous to the T, D, M, and I options for the ARM7 and ARM9 designs. ARC Cores has a similar package of specialized ISA options, and MIPS licensees have MIPS-16 code compression. Motorola's ever-evolving ColdFire product line includes chips with MAC units and other options, although these modifications are controlled by Motorola, not (directly) by its customers.

Synthesis Tools Are Key

The entire Xtensa processor core is synthesized from Verilog code. Rowen points out that the processor can be synthesized using virtually any standard-cell library and memory generator. Tensilica neither recommends nor requires any special libraries; even the register file is implemented as gates, not as a specially hand-packed multiported macro cell.

Such a level of generic, lowest-common-denominator synthesis would normally exact a heavy toll on performance, a situation Tensilica claims to have overcome. The company suggests a practical target frequency of 150–175 MHz in a "generic" 0.25-micron CMOS process. Speeds of up to 250 MHz are possible under less pessimistic process, voltage, and temperature conditions, according to Tensilica.

Tensilica's clock-speed claims don't seem out of line with what other vendors are producing. Microprocessors with optimized 0.25-micron layouts run at about 300 MHz today. ARM claims that its synthesized ARM7TDMI-S core can reach 90% of the clock speed of a hand-optimized design (though at a 2× to 3× penalty in die size), so a 175-MHz Tensilica part doesn't seem unreasonable.

Regarding size, the base implementation should require fewer than 25,000 gates and need less than 1 mm^2 of silicon, according to Tensilica. Power consumption is likewise modest, estimated to be less than 0.5 mW per MHz. All these parameters put Xtensa at the low end of the power, size, and gate-count scales compared with other 32-bit processors. Realistically, at those scales, Xtensa's die size, transistor count, and power consumption are nearly irrelevant in the larger environment of a complete ASIC. Putting multiple processors on a single die becomes practical at these scales.

Software Tools As Important As the Hardware

Hardware engineers may applaud Xtensa's flexibility, but software developers may abandon their medication regimen at the prospect of creating and debugging code for a chip with no fixed instruction set. Tensilica claims this eventuality has been more than covered with a software tool chain that is self-configuring, keeping pace with the CPU changes.

As part of the hardware-synthesis process, Tensilica's tools also modify the supplied GNU compiler, assembler, linker, debugger, profiler, simulator, and libraries to support any new instructions. The C language definition does not change, of course, but the compiler is altered with intrinsics that access new functions. The assembler, debugger, and other tools also will understand user-defined instructions and correctly disassemble and profile them. Users can thus define, synthesize, profile, and tune their processor in a big hardware/software feedback loop, looking for the best combination that achieves the desired optimization.

Third-party operating systems and other software tools are ported to only the basic Xtensa instruction set, ensuring compatibility with any Xtensa implementation. Currently, Tensilica has struck deals with ISI and Wind River for the pSOS and VxWorks real-time operating systems.

User Configurability the Start of a Trend

Tensilica's first public licensee is Zilog, an often overlooked processor supplier. Zilog intends to use Tensilica's processor in its own line of communications-related processors, making tools available to its many far-flung design centers.

Tensilica's business is to license its CPU core and related development tools to semiconductor houses, EDA companies, and large ASIC developers. For the most part, this is not a significantly different model than that adopted by Rambus, MIPS, or other newly minted IP companies.

One difference between Tensilica and MIPS is that Tensilica must maintain software-development tools and keep them in sync with its processor products. Unlike a strict processor-IP company, Tensilica cannot simply update and license its hardware cores; it must also maintain a watch on its software tools or risk diluting the advantage of its extensible processor.

The nearest extant example of Tensilica's strategy is ARC Cores (see MPR 7/8/96, p. 8), which also licenses an extensible microprocessor. Like Tensilica, ARC's CPU is synthesized and user extensible. Also like Tensilica, ARC has developed a portfolio of tested instruction-set extensions, its software tools track changes to the hardware architecture, and users can download interim CPU designs to an FPGA. ARC has a few years' head start over Tensilica and the advantage of revenue from more than 30 existing licensees.

Making technical comparisons between configurable microprocessors is a bit like pushing a rope. Tensilica and the other synthesizable or configurable processors all promise roughly equal performance (in clock speed) in a given process. In each case, real performance differences will be due to customer-designed extensions. And assuming such extensions could be implemented on any underlying architecture, we're back where we started.

Decisions of this sort might be based on design tools, licensing costs, degree of configurability, or ease of use. Processor-IP companies are typically mum on detailing their licensing fees, though most follow the standard practice of charging an upfront fee followed by royalties.

Tensilica is among the first in a new wave. ASIC designers who are comfortable with synthesis and who can benefit from configurability will find Xtensa an interesting alternative to traditional fixed CPU cores. Tensilica's tool chain is complete and well thought out. Xtensa's underlying architecture is sound, but it's only the starting point for custom departures. It is the customers—not the vendor—that make configurable processors valuable. ASIC designers are simply choosing a palette on which to create their perfect chip. M