Reed Electronics Group

MICROPROCESSOR REPORT

www.MPRonline.com

THE INSIDER'S GUIDE TO MICROPROCESSOR HARDWARE

SILICON HIVE BREAKS OUT

Philips Startup Unveils Configurable Parallel-Processing Architecture

By Tom R. Halfhill {12/1/03-02}

Parallel lines never meet, but great minds think alike. Maybe that explains the convergence of parallel processors at this year's Microprocessor Forum and Embedded Processor Forum. All over the world, maverick CPU architects are trying to exploit their soaring transistor budgets for something besides humongous on-chip caches.

The latest example—announced during the Extreme Processors session at **Microprocessor Forum 2003**—is a configurable parallel-processing architecture from Silicon Hive, a Netherlands-based startup funded by Philips Electronics. Silicon Hive has created what it calls an ultralong instruction-word (ULIW) architecture—an apt description. As with a very long instruction-word (VLIW) architecture, a special compiler bundles multiple operations together for simultaneous execution on a target processor. But with instruction words that stretch up to 768 bits long, each containing scores of operations, Silicon Hive's ULIW architecture surpasses every known VLIW machine. It's also highly configurable, allowing designers to alter almost every feature for optimum performance in vertical applications.

To support this unusual architecture, Silicon Hive has created a whole ecosystem: a tool chain for rapidly designing custom ULIW cores, a library of function units for designers to choose from, and adaptive software-development tools. Putting theory into practice, the company has developed two ULIW-based cores for licensing as synthesizable intellectual property (IP). Known as the Avispa ("wasp" in Spanish) and Avispa+, the preconfigured cores are designed for signal processing in orthogonal frequency-division multiplexing (OFDM) radio applications.

As a measure of what's possible with this architecture, Avispa+ can execute nine billion operations per second (GOPS) at a clock frequency of only 150MHz. When Avispa+ is fabricated in a 0.13-micron bulk CMOS process, core voltage is 1.0–1.05V and power consumption is only about 150mW under worst-case military conditions (125°C). The processor core will occupy a mere 4mm^2 of silicon. Although 150MHz is on the lower end of the clock-speed spectrum, the other statistics are impressive for a fully synthesized, standard-cell design that can issue 60 operations per clock cycle.

What's more, Silicon Hive says its proprietary tool chain can generate similar processor cores to customer specifications in a matter of days. The company's first customer (not yet publicly announced) expects to tape out an SoC with a customized ULIW core this quarter. That chip will be a media processor that replaces several ASICs in a low-end MPEG4 application. It may also be used for image stabilization in a mobile-phone video camera.

Essentially, Silicon Hive blends the design-time configurability offered by companies like ARC International, MIPS Technologies, and Tensilica with a parallel-processing architecture for compute- and data-intensive tasks. Silicon Hive's goal is to replace some of the general-purpose processors, DSPs, and ASICs in high-performance embedded systems with a customized, programmable, ULIW-based SoC coprocessor. The system may still require a general-purpose processor to run the operating system and handle control tasks, but it can offload the heavy number crunching or signal processing to the ULIW core.

One tradeoff of using a ULIW processor is that the compiler is responsible for scheduling instructions, synchronizing the pipeline, and allocating the CPU's resources, so programs must be recompiled for different implementations of the architecture. And as is always the case with extreme processors, Silicon Hive's greatest challenge will be explaining its unorthodox architecture to customers and justifying its benefits.

Taking a Guillotine to Overhead

Silicon Hive's cofounder and chief processor architect, Jeroen Leijten, argues that conventional microprocessor architectures are burdened with too much overhead to be the best choice for intensive processing. Conventional processors share function units, register files, buses, caches, and other resources throughout the chip, creating conflicts and interdependencies that have nothing to do with true data dependencies. As the chips grow more complex, they require increasing amounts of control logic to manage their resources and schedule instructions through their pipelines. Their centralized resources make it difficult to scale new designs in step with rising transistor budgets. And their instruction sets are rigid, conforming to a general-purpose model, with perhaps a few specialized extensions.

The solution, says Leijten, is to grant the computational logic more independence; replicate the function units on a larger scale; move as much control overhead as possible out of the processor and into the compiler; enable the processor to issue dozens of operations per cycle; and make it easier to customize the instruction set for specific applications.

Feature	Silicon Hive Avispa	Silicon Hive Avispa+
Architecture	ULIW	ULIW
Application Domain	ODFM radio	ODFM radio
Instruction-Word Width	486 bits	768 bits
Issue Slots Per Word	41 operations	60 operations
Instruction Memory	32K	48K
Arithmetic PSEs	4	4
Dual-Port Minicaches	—	4
Local Data Memory	4x4K dual port	4x4K single port
Viterbi Registers	—	4x48 bits
Control & I/O PSE	1	1
Function Units (Total)	75	103
Register Files (Total)	95	130
Clock Freq (0.13µ)	150MHz	150MHz
Core Area (0.13µ)	6.5mm^2	4mm^2
Power (150MHz)	~127.5mW	~150mW
Peak Performance	6.15 GOPS	9 GOPS
Availability	Now	Now

Table 1. Despite its significantly greater resources, Avispa+ is smaller than the initial Avispa core and consumes less power while delivering higher performance. Silicon Hive says it achieved this by making the local data memories in the PSEs single ported instead of dual ported, which simplified the routing and halved the size of the memories in the synthesized layout. To compensate for the single porting, Avispa+ adds small caches to the data memories.

None of these ideas is new, of course, although combining all of them in one architecture is unusual. Turning function units into miniprocessors, with their own local resources, is common in other parallel architectures recently described in *MPR*, such as those from ClearSpeed, Elixent, and PicoChip. (See *MPR 11/17/03-01*, "Floating-Point Buoys ClearSpeed," *MPR 7/21/03-01*, "Elixent Expands SoCs," and *MPR 10/14/03-03*, "PicoChip Makes a Big MAC.") Shifting control overhead and resource allocation into the compiler is the underlying philosophy of all VLIW architectures and Intel's EPIC VLIW variation. (See *MPR 10/6/99-01*, "Merced Shows Innovative Design.") Configurable instruction sets are the salient feature of the ARC, Tensilica, and most recent MIPS microprocessor cores. (See *MPR 9/22/03-01*, "ARC Accelerates Cryptography," *MPR 6/23/03-01*, "Tensilica's Software Makes Hardware," and *MPR 3/3/03-01*, "MIPS Embraces Configurable Technology.")

Silicon Hive's innovation lies in uniting extremely wide instruction-level parallelism with design-time configurability. The company has developed a sophisticated hardware/software design system that rapidly generates an optimized processor core, an instruction-set architecture (ISA), and a matching tool chain to customer specifications. This system generates the processor core from a flexible architectural template that can vary the number of processing units, function units, register files, interconnects, and local memories. To optimize the ISA, designers can add new instructions, function units, and registers, and they also determine the number of issue slots (operation subfields) within an instruction word.

Even the lengths of operations within the instruction words are configurable. Individual operations can vary in size, unlike the fixed subfields in most VLIW architectures. For instance, the Avispa+ signal-processing core has 768-bit instruction words and operations that vary in length from 1 bit to 30 bits. A single Avispa+ instruction word may contain up to 60 issue slots, so the average length of an operation is only 12.8 bits—more compact than a typical CISC instruction.

Like a VLIW compiler, however, the ULIW compiler must use inactive NOPs to pad any issue slots it can't fill with useful operations. Therefore, the average length of an active operation will be longer than 12.8 bits, depending on the nature of the program code.

Distributed Resources Avoid Conflicts

The foundation of Silicon Hive's ULIW architecture is a logic block called a processing and storage element (PSE). It's much more than a function unit and a little less than a self-contained processor. PSEs are configurable during the design phase of a ULIW core. Each PSE has its own function units, register files, issue slots, memory I/O, local interconnects, and interfaces with neighboring PSEs. ULIW processors can have many PSEs, which work together to form a common datapath.

For example, Avispa+ has four identical PSEs for arithmetic operations. Each PSE provides 12 instruction-issue slots, representing one or more function units per slot. Those function units include a 16-bit ALU; a 16 × 16-bit multiplier; a 40-bit adder/accumulator that, together with the multiplier, makes a 16 × 16-bit multiply-accumulate (MAC) unit with 40-bit accumulation; a 40-bit scaler for the accumulator; a 16-bit barrel shifter; two 16-bit address-generation units (AGU) with modulo addressing; a four-way SIMD add-compare-select unit that operates on 48-bit packed data for Viterbi acceleration; and two 16-bit load/store units connected to dual-ported minicaches and 4KB of single-ported local data memory.

In addition, Avispa+ has a fifth PSE for control and I/O. Its function units include a 16-bit ALU; a branch unit; two 16-bit AGUs; two 16-bit load/store units connected to dual-ported minicaches and 32KB of single-ported local data memory; and a 32-bit load/store unit connected to a bus-master interface. Counting the four arithmetic PSEs and the control-I/O PSE, Avispa+ has 60 available issue slots in a 768-bit instruction word. Under ideal conditions, the core can actually issue 60 instructions per clock cycle, which adds up to 9 GOPS at 150MHz. Table 1 compares the vital features of the Avispa and Avispa+ cores.

Distributing resources across the processor is fundamental to the ULIW architecture. Function units have their own dedicated registers instead of sharing a centralized register file with other function units, although they can sometimes save results in the registers of other function units. Typically, a dedicated register file has no more than four registers, usually 16 bits wide. Yet register starvation shouldn't be a problem with this processor, and the core is smaller than it would be with a centralized register file large enough to serve its needs. Instructions are smaller, too, because they need fewer register-address bits. In fact, some instructions have no register addressing at all, because they use only one register. The compiler handles all register assignments, so the processor requires no control logic for avoiding register conflicts.

Figure 1 is a block diagram of the basic ULIW architecture. Because the architecture is configurable, this diagram doesn't represent any particular core implementation. The "network" bubbles represent the local interconnects between a PSE's register files and function units.

What keeps the PSEs from qualifying as self-contained processors is their lack of local instruction memory and autonomy. Although they have local data memories, they fetch instructions from a common on-chip instruction memory, whose size is configurable. The instruction memory's width is equal to the ULIW width, and the designer determines the depth. For instance, Avispa+ has 768-bit-wide instructions and enough on-chip memory for 512 instructions, or 48KB total. The earlier Avispa core also has enough memory for 512 instructions, but the instructions are "only" 486 bits wide, so instruction memory totals 32KB.

Unlike the array processors in PicoChip's PC101 and PC102 chips, Silicon Hive's PSEs aren't designed to work independently. No matter how many PSEs are present, they all work in parallel, and the software-development tools view them as a single datapath.

Two Modes of Execution

The ULIW architecture has two processing modes, which are really different methods of execution instead of distinct hardware modes with unique states. In what Silicon Hive calls *standard DSP mode*, the processor fetches a new instruction every clock cycle and is much like a conventional VLIW processor—the ULIW instructions may contain multiple operations that execute in parallel. Generally, this mode handles control code, the pre- and post-ambles of loops, conditional branches, and irregular code that has less parallelism than the innermost loops of algorithmic subroutines.

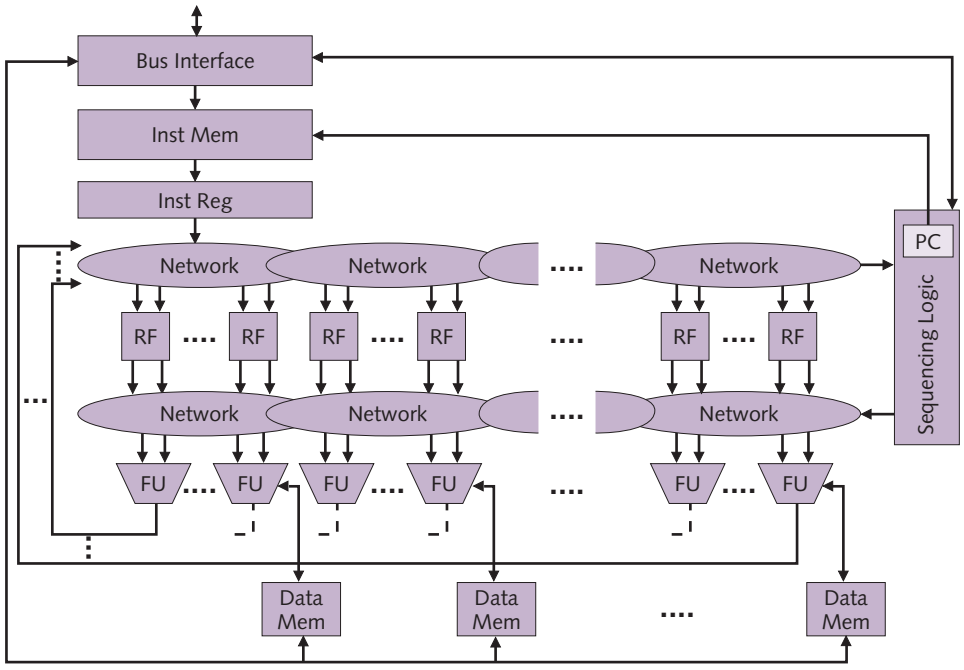

Figure 1. Block diagram of Silicon Hive's ULIW architecture. The number of processing and storage elements (PSE), register files, function units, and data memories is flexible, depending on the core configuration. The master/slave I/O bus is configurable; the Avispa and Avispa+ cores have a 32-bit I/O bus.

In the other execution mode, *pure dataflow mode*, the processor doesn't fetch a new instruction every clock cycle. Instead, it executes a tight loop within a single instruction. This happens when the compiler can encapsulate the entire body of an inner loop in a single ultralong instruction word, typically when there is a great deal of exploitable parallelism.

In effect, dataflow mode is like repetitive execution within a programmable logic block of an FPGA: data flows through the block, the programmable logic operates on the data, and the processor doesn't have to keep fetching the same instructions over and over again.

Pure dataflow mode can handle sophisticated tasks. For instance, the Avispa and Avispa+ cores are designed to execute the butterfly routine of a 2,048-point complex fast Fourier transform (FFT) in dataflow mode with a single ULIW instruction. In fact, the processor spends 98% of its execution time for the FFT in a single-instruction, single-cycle loop.

Figure 2 illustrates the efficiency of pure dataflow mode. It's a screen shot of a Silicon Hive profiling tool that lets the CPU architect or application programmer visualize how efficiently a section of code is using the processor's resources. In this example, the tool is analyzing the execution of a 2,048-point FFT on an Avispa core that has four PSEs. The labels along the left represent the 41 issue slots for the function units of the PSEs. The numbers along the top represent sequential memory addresses for the routine. The columns indicate whether an issue slot is "active"—that is, occupied by a useful instruction. Dark squares in the columns indicate active slots; the gray squares indicate inactive slots (padded with NOPs). The numbers along the bottom indicate the percentage of active issue slots per instruction, obtained by dividing the number of active slots into the total number of slots available in this core. In other words, darker squares are better, and they tend to cluster around the innermost loops of the FFT, which the core executes in a single cycle.

An off-the-shelf compiler would be baffled by this extreme architecture, so Silicon Hive provides its own C compiler, HiveCC. It's a proprietary spatial compiler that uses an optimization technique known as constraint solving. HiveCC creates a mathematical constraint model of the

Figure 2. This screen shot of a Silicon Hive profiling tool shows an FFT executing on an Avispa ULIW core that has four PSEs and 41 instruction-issue slots. While executing the body of the innermost loop, the core keeps 31 out of 41 issue slots active (76%). Overall, the core is spending 98% of its execution time in the single-instruction, single-cycle loops of pure dataflow mode, as indicated by the darkest column of squares.

program's dataflow graph and matches it with a similar constraint model of the processor. Aggressive software pipelining and compiler-directed instruction scheduling and register allocation are key parts of this technique. Silicon Hive claims HiveCC can uncover much more instruction-level parallelism than compilers for other processors.

If that claim is true, HiveCC is a significant advance in applied high-performance computing. Past attempts at extreme instruction-level parallelism have fallen short—not for lack of issue width or processing resources, but because it's so difficult to find and exploit the inherent parallelism in real-world code. With instruction words that are hundreds of bits long containing dozens of issue slots, Silicon Hive's ULIW architecture is certainly wide enough. However, it seems that such wide instruction words would be overpopulated with NOPs much of the time. The FFT profile in Figure 2 is impressive, but is it representative?

To tame our skepticism, Silicon Hive offered the additional data shown in Table 2. It demonstrates how efficiently Avispa and Avispa+ can use their issue slots and execution resources when running a variety of signal-processing tasks. In pure dataflow mode, issue-slot utilization ranges from 36.6% to 82.9%, and execution utilization (the number of operations "in flight") ranges from 41.5% to 107.3%. (Execution utilization can exceed 100% because the function units are pipelined, so the number of operations in various stages of execution can exceed the number of issue slots.) In standard DSP execution mode (the second Viterbi example for Avispa+), both measures of utilization drop by almost half, but they are still respectable for a mode that's inherently less parallel.

Because the compiler, not the processor, is responsible for scheduling instructions and allocating the processor's resources, software must be recompiled for different ULIW implementations. For instance, without recompilation, a binary executable for Avispa won't run on Avispa+, or vice versa, even though they are similar cores.

Binary incompatibility among different implementations is a common limitation of VLIW-type architectures. It's what prompted Intel and Hewlett-Packard to devise EPIC—a clever variation of VLIW that allows different IA-64 processors to remain binary compatible. For a server-processor architecture, binary compatibility is vital, because users want to run their installed base of software without recompiling the code each time a new processor is introduced. Silicon Hive's ULIW architecture is intended for embedded processors, so compatibility with an installed base of software is somewhat less important.

Price & Availability

Silicon Hive's Avispa and Avispa+ synthesizable processor cores are available for IP licensing now. In addition, the company will create custom implementations of the ULIW architecture to customer specifications. Licensing fees and terms have not been disclosed. For more information, visit *www.siliconhive.com*.

Silicon Hive Is the Queen Bee

Unlike ARC, MIPS, and Tensilica, Silicon Hive doesn't allow customers to operate the processor-configuration machinery themselves. There's no utility like ARChitect or the Xtensa Processor Generator that exposes the configurable features of the architecture directly to customers. Instead, Silicon Hive takes the customer's specifications, generates the core internally, and delivers the synthesizable model in VHDL or Verilog format. At that point, customers can use industry-standard EDA tools to complete their SoC design.

In practice, Silicon Hive expects to satisfy most customers by letting them choose from a collection of ULIW cores preconfigured for popular embedded applications. Only if a customer needs something special would Silicon Hive have to generate a wholly new implementation.

At the center of this system is a proprietary hardware-design language called The Incredible Machine (TIM). TIM allows Silicon Hive to configure a core by specifying high-level parameters: the number of function units, register files, and interconnects; the number of issue slots; the list of instructions each function unit can execute, and so forth. From this information, TIM deduces other parameters, such as the length of the instruction words.

TIM is a higher-level language than VHDL, Verilog, or Tensilica Instruction Extension (TIE) language. It can

Task	Processor Core	Issue Slots	Ops Issued Per Cycle	Issue-Slot Utilization	Op Execution Per Cycle	Execution Utilization
2K-Point FFT	Avispa	41	31	75.6%	41	100.0%
32x32-Bit DCT	Avispa	41	34	82.9%	44	107.3%
Baseband Decider	Avispa	41	15	36.6%	17	41.5%
Freq De-Interlacer	Avispa	41	15	36.6%	23	56.1%
Inter-Demodulator	Avispa	41	17	41.5%	23	56.1%
Intra-Demodulator	Avispa	41	17	41.5%	21	51.2%
Viterbi Decoder*	Avispa+	60	37	61.7%	59	98.3%
Viterbi Decoder**	Avispa+	60	22.8	38.0%	29.5	49.2%

Table 2. Silicon Hive collected this data by running some common signal-processing algorithms on simulations of the Avispa and Avispa+ cores. "Operations issued per cycle" excludes the NOPs required to pad inactive subfields in the instruction words. "Op execution per cycle" indicates how many non-NOP instructions were in various stages of execution. *A single-cycle loop executed in pure dataflow mode. **A six-cycle loop executed in standard DSP mode.

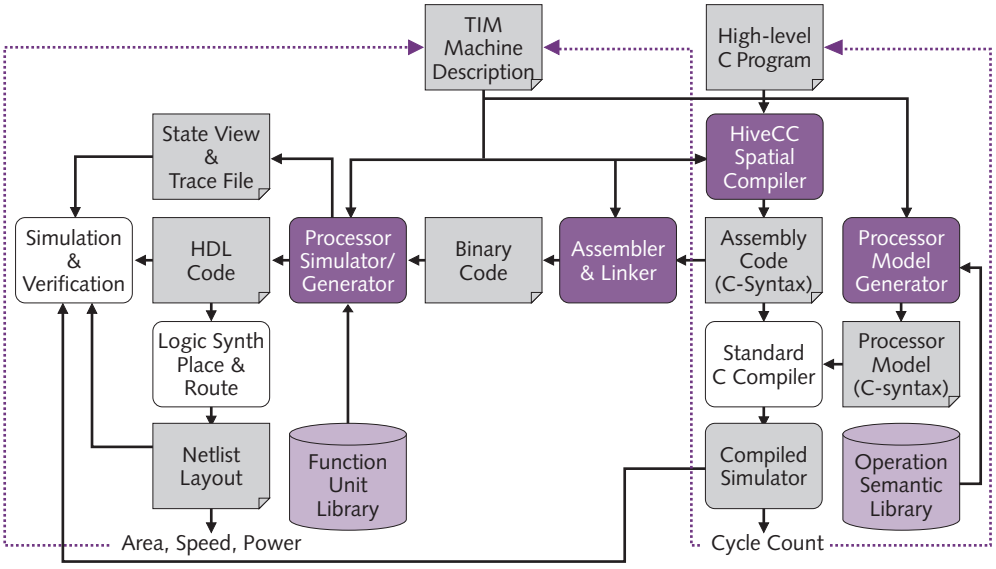

Figure 3. Silicon Hive's feedback-driven design system uses a proprietary configuration language called TIM to generate optimized implementations of the CPU architecture. Feedback from the cycle-accurate simulator helps the architect fine-tune the design by modifying the microarchitecture and the instruction set. Notice the function-unit library at the lower left—a collection of logic blocks precompiled in register-transfer-level VHDL.

describe a register file by simply specifying the number of registers, the widths of the registers, and the number of read/write ports. From these descriptions, TIM invokes prewritten blocks of VHDL or Verilog. A mere 300 lines of TIM can result in 100,000 lines of Verilog or VHDL. TIM also drives the development-tool generator that creates a matching assembler, linker, C compiler, instruction-set simulator, and cycle-accurate simulator. Figure 3 is a flowchart of Silicon Hive's design system.

Silicon Hive says it can create an optimized implementation of its ULIW architecture in a matter of hours, depending on the amount of fine-tuning the customer requires. After Silicon Hive delivers the design in VHDL or Verilog format, final synthesis and layout might take a few hours to a few days, depending on the design's complexity and the amount of simulation and verification required. Turn-around time is comparable to what is possible with the customer-driven design tools that other configurable-processor vendors offer.

Even so, we think it would be advantageous for Silicon Hive to put customers in the driver's seat with a graphical user interface that exposes the ULIW architecture. It would push more design work outside the company and give customers a greater sense of control over their configurations. The disadvantage is that it would require more customer education and tech support, but that doesn't seem to hinder other configurable-processor vendors.

A Hybrid Strategy for Licensing IP

It's interesting to compare Silicon Hive's business model with those of rival configurable-processor vendors: ARC, MIPS, and Tensilica. Silicon Hive has more in common with MIPS than with ARC or Tensilica, but its strategy is unique.

Like MIPS, Silicon Hive is licensing preconfigured soft processor cores as well as configurable cores. Avispa and Avispa+ serve as proof-of-concept implementations of the ULIW architecture, and they are also ready-to-use licensable IP. To date, with a few minor exceptions, ARC and Tensilica haven't licensed preconfigured cores, preferring instead to let their customers do the tweaking.

Silicon Hive departs from all three companies' practice by keeping the configuration tools in-house and by offering an extreme parallel-processing architecture instead of a general-purpose RISC architecture. Despite the vast differences among these architectures, Silicon Hive may compete for some designs head-to-head against ARC, MIPS, and Tensilica, because DSP extensions and other customizations can make the RISC cores suitable for signal-processing applications.

Another departure for Silicon Hive is its willingness to develop application software for customers. When a customer licenses a ULIW processor core, Silicon Hive offers a seat license for the HiveCC tools or the option of a design-services license. In general, ARC, MIPS, and Tensilica don't offer to write application software for their customers, although they might make an exception if it helps to close an especially lucrative deal.

Offering software-design services could be a smart way for Silicon Hive to hedge its bets. So far, ARM is by far the most successful provider of microprocessor IP. Everyone else is struggling to keep their heads above water, and a few companies (Lexra, PicoTurbo, and Philips's own TriMedia spin-off) have either drowned or retreated from processor-IP licensing. The differentiation could be an advantage for Silicon Hive, especially if it helps to lure potential customers that are wary of writing software for an oddball architecture—even though Silicon Hive insists that the C programming model is no different from that of any other microprocessor.

The greatest challenge for any extreme-processor architecture is proving it deserves to exist. There are plenty of familiar microprocessor architectures for customers to choose from. For that matter, as past issues of *MPR* attest, there are also plenty of unfamiliar microprocessor architectures to choose from. To stand out from the crowd, it would

help if Silicon Hive had verifiable benchmark results comparing the performance of ULIW cores against competing processors across a wide range of software. The company needs to make a strong case that ULIW is a valid architectural exercise—not just VLIW on Viagra.

Even if a ULIW processor plays a secondary role as a specialized coprocessor to a host processor, it may not be a better option than a DSP or ASIC. An off-the-shelf DSP doesn't require such a great leap of faith, and an ASIC—though perhaps not programmable—can deliver the same or greater performance.

To win designs, Silicon Hive needs solid benchmark data, successful examples in silicon, and open-minded customers. With two ULIW cores on the shelf and one early customer near tapeout, the company already has a toehold in the market. ◇