National Enters Embedded RISC Market

Piranha Family Will Scale from Eight to Sixty-four Bits

by Brian Case

Just when it seemed safe to assume that the tide of new microprocessor architectures had ebbed, Gideon Intrater of National Semiconductor revealed another simple embedded architecture at last month's Microprocessor Forum. Dubbed Piranha, the architecture is intended to be used in application-specific embedded processors.

Now that National has discontinued its failed 32000 processor family, the company has essentially dropped out of the stand-alone embedded CPU market. National has no current plans to package a stand-alone Piranha CPU chip, but it will do so if sufficient customer demand materializes.

National is following a recent trend exemplified by Hitachi and NEC with their SH7000 *(see* ***080203.PDF****)* and V800 *(see* ***081303.PDF****)*: instead of using or licensing an existing architecture, these companies have defined their own captive architectures. The motivation is to minimize code size, die area, and power dissipation while maximizing performance. Other benefits of a proprietary architecture, such as saving a royalty fee on tens of millions of units, may also play a part.

Existing architectures are seen as either too power- and memory-hungry—often the case with RISCs—or lacking in performance—often the case with CISCs. National, like Hitachi and NEC, believes it has defined an architecture that optimizes all four of the metrics mentioned above. In contrast to other architectures, however, National has defined Piranha for implementations with data sizes from 8 to 64 bits. So far, the company has designed 16- and 32-bit Piranha cores and is already integrating them into custom chips.

Piranha Is Simple and Minimal

Based solely on a strict interpretation of the RISC tenets, the Piranha architecture is not completely a RISC. As shown in Figure 1, it has variable-length instructions and, with only two register fields, lacks three-address register-to-register operations.

Still, it seems fair to label Piranha RISC-like, especially in light of the instruction set shown in Table 1. Not only is the instruction set simple and modest in size, it also adheres to the RISC tenet of referencing memory only with load and store instructions. The first Piranha cores handle unaligned data accesses in hardware (instructions must be 16-bit aligned), but future cores can choose to trap on unaligned data accesses.

As in the SH7000 and V800 architectures, the basic Piranha instruction format is 16 bits long to minimize code size. The 16-bit format can encode either two register numbers or a register and a small four-bit constant.

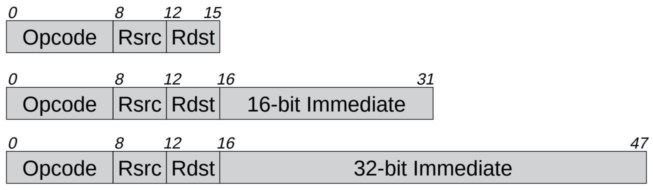

Figure 1. Piranha 32-bit instruction formats. The 16-bit Piranha uses only 2- and 4-byte formats

Mnemonic	Operands	Description
Moves		
MOVi	Rsrc/imm, Rdest	Move
MOVXi	Rsrc, Rdest	Move, sign extend
MOVZi	Rsrc, Rdest	Move, zero extend
Integer Arithmetic		
ADD[U]i	Rsrc/imm, Rdest	Add, signed or unsigned
ADDCi	Rsrc/imm, Rdest	Add, with carry
MULi	Rsrc/imm, Rdest	Multiply
SUBi	Rsrc/imm, Rdest	Subtract (dest – src)
SUBCi	Rsrc/imm, Rdest	Subtract, with carry
Integer Compare		
CMPi	Rsrc/imm, Rdest	Compare (dest – src)
Logical and Boolean		
ANDi	Rsrc/imm, Rdest	Logical AND
ORi	Rsrc/imm, Rdest	Logical OR
Scond	Rdest	Save condition as boolean
XORi	Rsrc,imm, Rdest	Logical Exclusive OR
Shifts		
ASHUi	Rsrc/imm, Rdest	Arithmetic left/right shift
LSHi	Rsrc/imm, Rdest	Logical left/right shift
Bit Test		
TBIT	Rsrc1/imm, Rsrc2	Test bit
Processor Register Manipulation		
LPR	Rsrc, Rproc	Load processor register
SPR	Rproc, Rdest	Store processor register
Branches		
Bcond	disp	Conditional branch
BAL	Rdest, disp	Branch and link
BR	disp	Branch
EXCP	vector	Vectored trap
Jcond	Rdest	Conditional jump
JAL	Rlink, Rdest	Jump and link
JUMP	Rdest	Jump
RETX		Return from exception
Load and Store		
LOADi	disp(Rbase), Rdest	Load, register relative
	disp(Rbase, Rbase+1), Rdest	Load, far relative
	disp, Rdest	Load, absolute
STORi	Rsrc, disp(Rbase)	Store, register relative
	Rsrc, disp(Rbase, Rbase+1)	Store, far relative
	Rsrc, disp	Store, absolute
Miscellaneous		
DI		Disable maskable interrupts
EI		Enable maskable interrupts
WAIT		Wait for interrupt

Table 1. The Piranha instruction set. Only the 16-bit machines implement the far-relative addressing mode. Byte, word, or long (B, W, D) can be substituted for the "i" placeholder in the mnemonics.

A 32-bit format is used to encode a 16-bit constant, and, as shown in Figure 1, the 32-bit machine uses a 48-bit format to directly encode a full 32-bit constant. Presumably, a 64-bit machine would have an 80-bit format to encode a 64-bit constant, but National says the prospect of a 64-bit machine is largely an insurance policy against possible future needs; there are no immediate plans for a 64-bit Piranha core. Eight-bit cores are more likely.

The precise details of the instruction formats and the instruction encodings highlight a unique aspect of the Piranha cores: the 16- and 32-bit versions are *not* binary compatible. Instead, compatibility is defined to be at the assembler mnemonic level. That is, the binary instruction encodings for the two existing cores are different, and National leaves open the possibility for two Piranha cores to have the same word length but different encodings. This would be useful if, say, some DSP instructions were added to form a new 32-bit core.

For each unique core, National will choose instruction encodings to minimize code size for the intended applications. Applications are compiled with different encoding schemes and code sizes are measured. Different encoding schemes are tried until minimal code size is obtained. National claims its engineers spent months choosing the instruction encodings for the first two cores. Both in-house and customer code were used to tune the encodings.

Figure 2 shows the simple Piranha register programming model. The four-bit register fields in the instructions permit 16 general-purpose registers. Although the subroutine-call instructions (branch-and-link and jump-and-link) can store their return address in any register, RA is used by the software calling convention. Register SP is fully general-purpose but is reserved by software as the program stack pointer. There is no "always-reads-as-zero" register as in most RISCs.

The PSR (processor status register) contains various status and control bits. Piranha uses condition codes, which are stored here. The interrupt-enable and instruction-tracing control bits are also in the PSR.

The ISP (interrupt stack pointer) register points to the stack used to save state when an interrupt or exception is taken. Only the PC and PSR are saved automatically. The INTBASE register defines the start of the 128-entry interrupt vector table.

The CFG (configuration) register serves no purpose in the first two cores and is currently undefined.

So far, only the 32-bit core implements the debug registers. The DCR (debug control register) has bits to control data- or instruction-address checking against the address in the CAR (compare address register). The DCR allows data-address checking to be selected for data reads, writes, or both. The DSR (debug status register) allows the debug exception handler to determine the kind of address match that occurred.

In the 32-bit Piranha core, all registers are 32 bits long. In the 16-bit core, all registers are 16 bits long except the PC, ISP, and INTBASE, which are 18 bits long.

The Piranha machines are little endian, and Figure 3 shows the memory maps for the first two cores. The entire 32-bit address space is available for any purpose except for the last 8M, which is reserved. The last 1K is reserved for interrupt control.

The 16-bit machine actually has an 18-bit linear address space. The first 128K—except for a hole reserved for interrupt control—can be used for any purpose. The

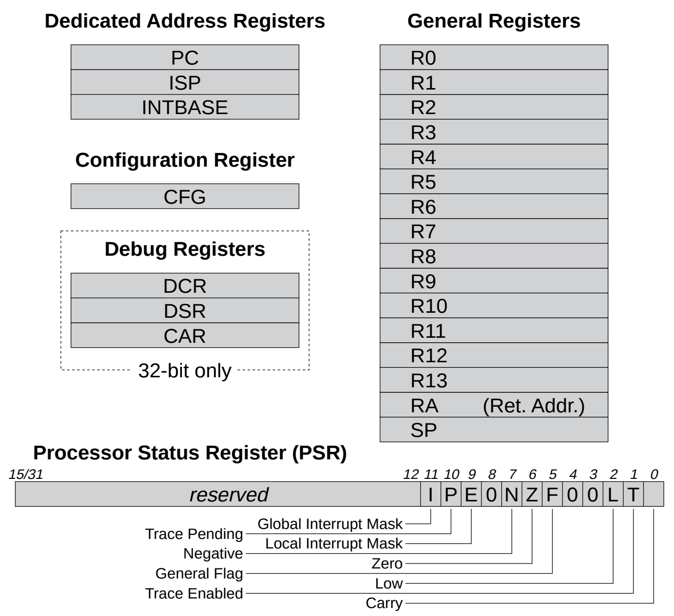

Figure 2. The Piranha programming model. In the 16-bit machine, the PC, ISP, and INTBASE are 18 bits long. In the 32-bit machine, all registers are 32 bits long.

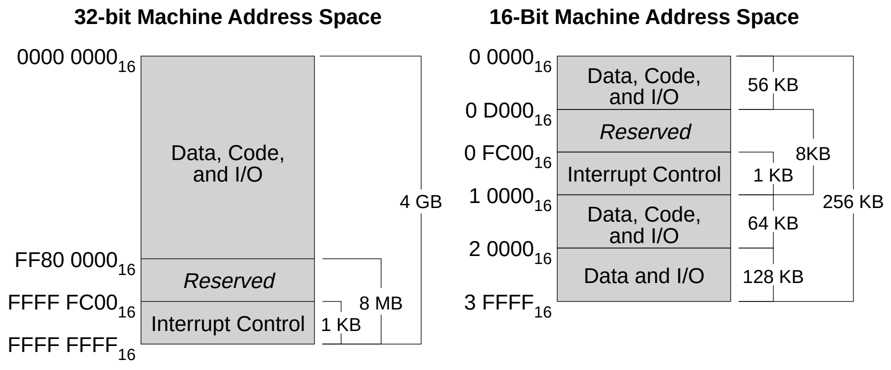

Figure 3. Address spaces for the 32-bit and 16-bit machines. The 32-bit machine has a fairly predictable address space, but the 16-bit machine has an address space that is larger than the natural word size.

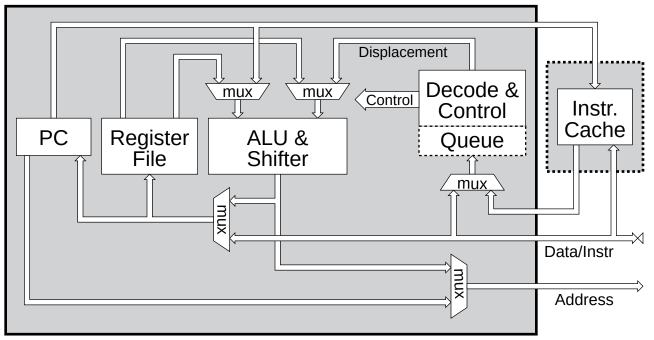

Figure 4. Block diagram of the simple hardware in the first Piranha implementations. Only the 32-bit implementation has the instruction queue and buses for the optional instruction cache.

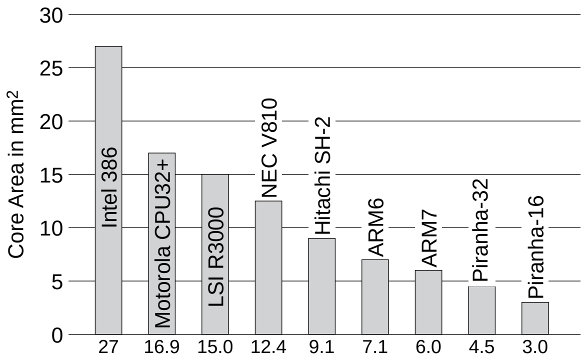

Figure 5. The simple Piranha cores achieve remarkably small implementations, beating even the tiny ARM cores. (Source: National)

last 128K can only be used for data and I/O. The 16-bit machine uses the far-relative addressing mode or a full 18-bit displacement to gain access to the entire 18-bit address space. The far-relative mode concatenates the values in a pair of adjacent registers to form a 32-bit address, of which only the low 18 bits are used to reference memory. Far-relative addressing is not used in the 32-bit machine.

Simple Implementation Saves Die Area

Figure 4 shows a block diagram of the first two Piranha implementations. In keeping with the spirit of the intended applications, the implementations use a simple three-stage fetch-decode-execute pipeline. A more complex pipeline might have allowed a faster clock rate, but even this simple pipeline can run at 30 MHz in a 0.8-micron CMOS process. National says faster operation is not required for the intended applications and would only create EMI and other system-design problems.

The three-stage pipeline crams register-read, ALU/shift, and register-write operations into the single execute stage. This has the advantage of eliminating register forwarding and a few pipeline registers and saves the associated power. National says the idea of eliminating all transistors that were not absolutely necessary was a recurring theme of the Piranha project.

The resulting Piranha cores are two of the smallest high-performance cores available in the industry. Figure 5 compares core sizes for some embedded processor implementations. These cores are all in 0.8-micron CMOS, although specific process parameters vary. As shown, the two Piranha cores are the smallest, and the 16-bit core is about 50% of the size of the tiny ARM7. Other vendors have achieved smaller cores using 0.5-micron processes, including a 4-mm^2 ARM6 and Motorola's Coldfire, at 4.4 mm^2.

For practical purposes, the absolute area differences between the Piranha and ARM cores are not compelling in the context of a complete ASIC or custom IC that implements the CPU, RAM, ROM, peripherals, and system logic. The Piranha cores might save 3 mm^2, but the entire die would likely be 40 mm^2 to 80 mm^2. A savings of 5% is worthwhile but not compelling enough to switch architectures. Even 10% probably would not tilt the scale in Piranha's favor.

Stealing design wins, however, is not National's strategy. Piranha's target is applications that are changing from CISC to RISC—or are embedding a processor for the first time—and thus are open to all options.

To help attract these designs, one of Piranha's chief design goals is to save cost by reducing RAM and ROM requirements through high code density and hardware support for unaligned data, which facilitates the use of packed data structures. Figure 6 shows a code-size comparison between the Piranha-32 and some other architectures. Piranha achieves code density on par with the 68K and about 25% better than other RISCs. Though conveniently missing, the SH7000 and V800 should have code sizes close to Piranha's.

The core size advantage plus a 25% savings in code space could give Piranha a significant die area advantage over standard RISCs in some applications. In the

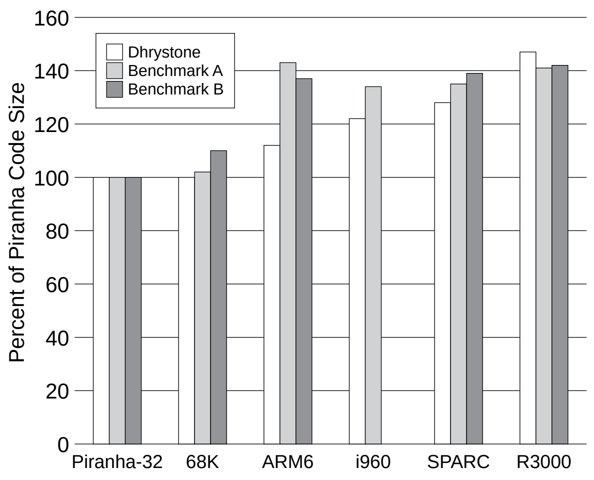

Figure 6. Relative code sizes. The Piranha-32 beats other RISCs on code density and can equal that of the 68000. (Source: National)

high-volume designs for which Piranha is intended, this difference could reduce die cost enough to be compelling. This explains why Hitachi and NEC designed their own architectures and why Motorola recently introduced a stripped-down 68K core *(see **081405.PDF**)*.

Performance Is Competitive

Figure 7 shows the performance of Piranha and other cores on the Dhrystone benchmark. The Piranha cores achieve middle-of-the-road performance, but they do so without caches. National expects future Piranha cores to incorporate at least an instruction cache to deliver higher performance for certain applications.

Figure 8 shows yet another metric where Piranha comes out on top. Because of their small silicon area, the Piranha cores have the best MIPS per mm^2 rating of the cores shown. Until now, the ARM7 had an impressive lead in this category. Piranha's improvement is an indication of what can be achieved with a careful and harmonious design of both architecture and implementation.

Gideon Intrater describing National's new embedded-control architecture.

Huge Volume Required

Given the glut of embedded architectures, it seems strange that National would go to the considerable trouble of defining and supporting a new one. National claims, though, that in the volumes envisioned for Piranha, it makes good sense. The work expended will be amortized over tens of millions of units per year when major designs—which National says it has already identified—reach maturity. In such high volumes, a possible 20% savings in die area from core and instruction ROM reductions, combined with royalty savings, provides enough return on the investment. National says that if projected volumes were only on the order of those shipped by the current high-end RISCs, such as the 960 and 29K (currently shipping a few million per year), it would not have made sense to design Piranha, and National would simply have licensed an existing RISC.

National is targeting the data-moving and data-shaping applications that will be important in coming high-volume products. These applications are characterized by mobility, communications orientation, data compression, and mixed-signal (analog and digital) environments. National already has some experience with these applications through its processors, ASICs, and software for devices that combine printer, fax, modem, and answering-machine capabilities (such as HP's OfficeJet) and the iPOWER personal encryption application. National says many applications that do not already use a CPU are fertile ground for medium-performance, low-power embedded processor designs.

Compared with some other embedded processor vendors, National seems to view the high-performance embedded market in a different light. In addition to 32-bit applications, National wants to be able to target applications for 8- and 16-bit processors. With the easy scalability of Piranha and captive designs and expertise, National seems well positioned to do so. Piranha will not likely steal existing design wins from the 960, 29K, or any other camps, so National must prove that the design wins it has identified will justify its extensive investment in Piranha. ♦

For More Information

For more information on Piranha, contact Mark Throndson at 408.721.4957, 408.736.6017 (fax), or *tmetsc@esd.nsc.com* (e-mail).

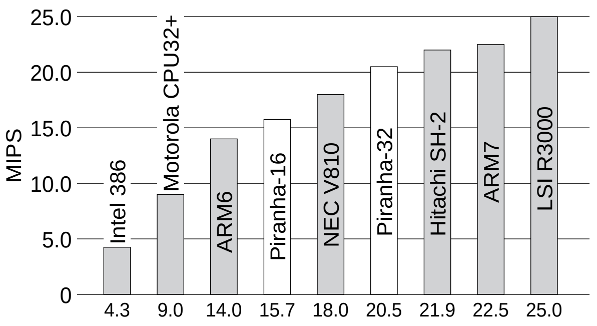

Figure 7. Performance at 25 MHz. The Piranha cores achieve middle-of-the-road—but competitive—performance. (Source: National)

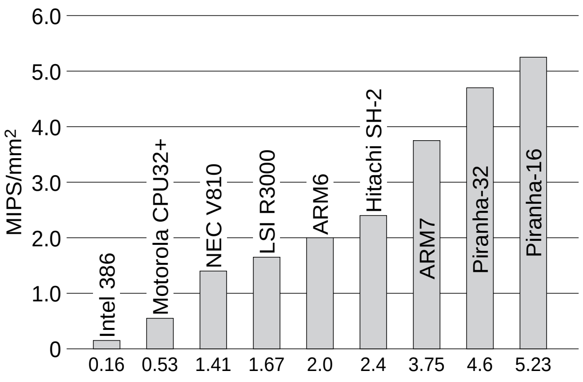

Figure 8. Because of their small implementations, the Piranha cores beat other processors in performance/mm^2 at 25 MHz.