Reed Electronics Group

MICROPROCESSOR REPORT

www.MPRonline.com

THE INSIDER'S GUIDE TO MICROPROCESSOR HARDWARE

MicroMIPS Crams Code

New Processor Cores Introduce Denser 16/32-Bit Instruction Set

By Tom R. Halfhill {11/16/09-01}

Smaller is usually better for embedded processors, so MIPS Technologies is slimming down its 1980s-vintage instruction-set architecture. A new set of 16- and 32-bit instructions—dubbed microMIPS—uses less memory than existing 32-bit MIPS instructions and the 16-bit extensions added in the 1990s.

MicroMIPS will debut early next year in two new embedded-processor cores, the MIPS32 M14K and MIPS32 M14Kc. The M14K is an improvement on the MIPS32 M4K processor, introduced in 2002. It's a relatively simple, cacheless core intended for 32-bit microcontrollers in automobiles, industrial machinery, consumer electronics, and office equipment.

Its bigger brother, the M14Kc, is an improvement on the MIPS32 4KEc processor, introduced in 2003. The M14Kc has an MMU with a translation lookaside buffer (TLB), making it suitable for sophisticated embedded operating systems that manage virtual memory. It's designed for advanced consumer electronics, including DTVs, DVD players, set-top boxes, home networking equipment, personal entertainment devices, and digital cameras. Figure 1 shows how the M14K and M14Kc fit into the MIPS product line.

Both new processors respond much faster to interrupts and have better debugging features than the MIPS cores they supersede. Both gain advantages in clock speed, power consumption, and core size when compared with ARM's Cortex-M3 and ARM926 processors, and they give ARM's new Cortex-A5 a run for the money.

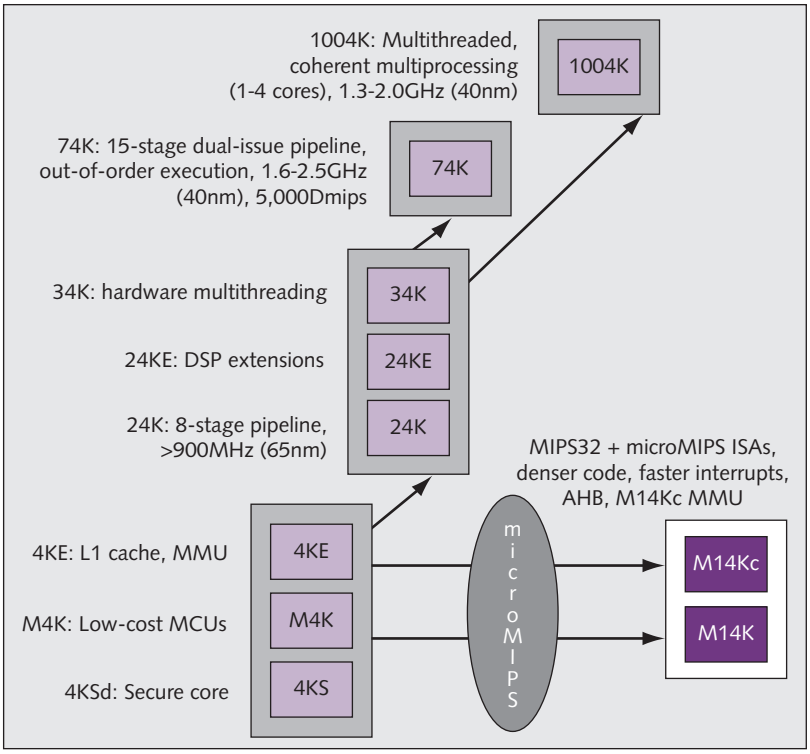

Figure 1. The new MIPS32 M14K and MIPS32 M14Kc processor cores introduce the microMIPS 16/32-bit instruction set and anchor the lower end of the MIPS product line. The M14K supersedes the M4K core, primarily for 32-bit microcontrollers. The M14Kc supersedes the 4KE core, offering an MMU for virtual-memory embedded operating systems.

(See *MPR 10/26/09-01*, "ARM's Midsize Multiprocessor.") Both have new atomic read-modify-write instructions. And the M14K has an optional accelerator that improves read/write performance with flash memory—especially useful for flash-based microcontrollers.

Overall, the new 16/32-bit microMIPS instruction-set architecture (ISA) is the most important feature of the M14K and M14Kc. According to MIPS, the memory footprint of executable code will shrink by about 35% while suffering a performance hit of only about 2%. Yet the new processors remain backward-compatible with the MIPS32 Release 2 ISA and existing MIPS32 software, because they include a MIPS32 instruction decoder.

MicroMIPS Outruns MIPS16e

One exception to backward compatibility is that the microMIPS ISA doesn't support the branch-likely instructions in MIPS32-R2. Those instructions, originally intended to tip off the processor that a branch is probable, are infrequently used and have been deprecated. The new microMIPS assembler automatically substitutes alternative instructions in their place, so the difference should be transparent to programmers.

Otherwise, existing MIPS32 software should run on the M14K and M14Kc without modification. Of course, developers must recompile to reap the advantages of the new microMIPS ISA. Recompiling embedded software is less earthshaking than recompiling PC software, so the requirement isn't onerous—and the gains are worth it.

Shrinking executable code by 35% while reducing performance by only 2% is impressive. At best, replacing every 32-bit instruction with a 16-bit equivalent would shrink the code by about 50%, so a 35% reduction for a real-world mix of 16- and 32-bit instructions is quite good. In comparison, the existing MIPS16e 16-bit extensions reduce code size by 25% to 30%. But the performance difference is striking. Whereas MIPS16e chokes throughput by about 30%, microMIPS exacts a mere 2% penalty. MicroMIPS code is smaller than MIPS16e code but plays bigger.

MicroMIPS will help MIPS Technologies compete more strongly for austere embedded systems. In particular, MIPS hopes to win a larger share of the 32-bit microcontroller market, which is highly fragmented among multiple vendors. No single CPU architecture rules this market. Indeed, some microcontroller vendors hedge their bets by using different 32-bit architectures in different product lines. ARM has been making aggressive inroads with its Cortex-M3 processor and older ARM7- and ARM9-family cores. With the new M14K and M14Kc, MIPS has a chance to gain share before one architecture (probably ARM) dominates.

Shedding the TinyRisc Legacy

The MIPS architecture is overdue for this kind of overhaul. Keep in mind that this early RISC architecture was designed in the 1980s for high-performance workstations and servers, not for embedded systems. In 1996, MIPS licensee LSI Logic was among the first to recognize the potential of the architecture for small iron. To reduce the amount of memory required for 32-bit code, LSI and MIPS Technologies collaborated on a new subset of 16-bit instructions called MIPS-16 (or MIPS16). The new instructions debuted in LSI's MIPS-compatible TinyRisc processor core. (See *MPR 10/28/96-10*, "LSI's TinyRisc Core Shrinks Code Size.")

It wasn't a new idea. The year before, ARM had extended its 32-bit RISC instruction set with 16-bit Thumb instructions. (See *MPR 3/27/95-01*, "Thumb Squeezes ARM Code Size.") But for MIPS, then part of Silicon Graphics, MIPS16 foreshadowed a strategic shift toward the embedded market as Intel's x86 began squeezing the RISC architectures out of servers and workstations. In 1999, after spinning off from Silicon Graphics, MIPS Technologies introduced its first synthesizable embedded-processor cores. (See *MPR 5/31/99-04*, "Jade Enriches MIPS Embedded Family.")

Owing to legal entanglements with LSI, MIPS was unable to include MIPS16 in those first cores. After the problems were resolved, MIPS16 reappeared as MIPS16e. And that's where things stood until November 2, when MIPS announced microMIPS, a major overhaul of the 16/32-bit instruction set.

MicroMIPS isn't just an extension, as MIPS16 was. MicroMIPS redraws the opcode map—the normally sacrosanct definition of an ISA. In fact, existing MIPS32 code wouldn't run on the new M14K and M14Kc at all if each processor didn't have two instruction decoders. One decoder handles microMIPS code, and the other handles "legacy" MIPS32 code. MicroMIPS is almost a clean-slate rethinking of the MIPS architecture.

MicroMIPS adds 15 new 32-bit instructions and converts 39 existing 32-bit instructions to 16-bit format. MicroMIPS includes another 215 existing 32-bit instructions from the MIPS32 ISA but remaps their binary opcodes. MIPS says future MIPS64 processors could use microMIPS, too, so the ISA is a genuinely new direction for the company. Table 1 lists all the new instructions.

Although MIPS says the microMIPS ISA does "code compression," it's not really compression in the same sense as data compression. Instructions aren't compressed in memory and expanded after fetching. Instead, MIPS shortened the most commonly used 32-bit instructions to 16 bits, so the opcodes use only half as much memory. Register references and immediate values don't necessarily shrink along with the opcodes, so the ideal goal of 50% memory reduction is unattainable.

A Unified 16/32-Bit ISA

Unlike ARM's original Thumb, microMIPS is generally modeless. Programs can freely mix 16- and 32-bit instructions in a single stream, without switching modes. However, a mode switch is necessary if a program uses both

Instruction	Description	Instruction	Description
New 32-Bit MicroMIPS Instructions			
ADDIUPC	Add program counter with immediate	**JALX**	Jump and link, switch to microMIPS
BEQZC	Branch if = 0, no delay slot	**JALX32**	Jump and link, switch to MIPS32
BNEZC	Branch if <> 0, no delay slot	**LWM32**	Load multiple words
BGEZALS	Branch if >= 0, short delay slot	**LWP**	Load word pair
BLTZALS	Branch if <= 0, short delay slot	**LWXS**	Load word, scaled, indexed
JALRS	Jump and link, short delay slot	**SWM32**	Store multiple words
JALRS.HB	Jump and link with hazard barrier, short delay slot	**SWP**	Store pair of registers
JALS	Jump and link, short delay slot		
New 16-Bit MicroMIPS Instructions			
ADDIUR1SP	Add stack pointer with immediate, unsigned	**LWM16**	Load multiple words
ADDIUR2	Add register with encoded immediate	**LWGP**	Load word from GP
ADDIUSP	Increment stack pointer	**LWSP**	Load word from stack pointer
ADDIUS5	Add to any GPR	**MFHI16**	Move from high part of MDU output*
ADDIU16	Add, unsigned	**MFLO16**	Move from low part of MDU output*
ANDI16	Logical AND immediate	**MOVE16**	Move GPR to GPR
AND16	Logical AND	**MOVEP**	Move pair of registers
B16	Unconditional branch	**NOT16**	Invert value
BEQZ16	Branch if = 0	**OR16**	Logical OR
BNEZ16	Branch if <> 0	**SB16**	Store byte
BREAK16	Trigger breakpoint exception	**SDDBP16**	Trigger debug exception
JALR16	Jump and link	**SH16**	Store halfword
JALRS16	Jump and link, short delay slot	**SLL16**	Logical shift left
JR16	Jump register	**SRL16**	Logical shift right
JRADDIUSP	Jump register, increment stack pointer	**SUBU16**	Subtract, unsigned
JRC	Jump register, no delay slot	**SW16**	Store word
LBU16	Load byte, unsigned	**SWSP**	Store to stack pointer
LHU16	Load halfword, unsigned	**SWM16**	Store multiple words
LI16	Create immediate value	**XOR16**	Logical XOR
LW16	Load word		

Table 1. MicroMIPS instruction set. This table includes all the new or reformatted 16- and 32-bit instructions in the microMIPS ISA. The table omits the 215 microMIPS instructions whose binary opcodes differ from existing MIPS32 instructions but whose mnemonics and functions remain the same. (All those instructions are 32 bits long.) Most new instructions resemble existing instructions and will be familiar to MIPS assembly-language programmers. Note that some branch instructions lack the traditional MIPS delay slot—in effect, a null operation following a branch that gives the processor time to calculate the branch-target address. *These instructions refer to results from the 32- x 16-bit multiply-divide unit (MDU).

microMIPS and MIPS32 instructions, and special instructions are provided for this purpose.

In essence, these mode-switching instructions steer the subsequent instruction stream to the corresponding instruction decoder. As mentioned above, the M14K and M14Kc processors have separate decoders for each ISA. Although the dual decoders have some redundant logic, they don't lengthen the five-stage pipelines inherited from the ancestors of these processors. Normally, a program will use one ISA or the other, so mode switches should be rare.

Another difference between microMIPS and ARM's original Thumb is that interrupt handlers can mix 16- and 32-bit instructions. There's no need to switch modes for different-length instructions when entering an interrupt routine. In virtually every respect, microMIPS is a unified 16/32-bit ISA. (ARM's 16/32-bit Thumb-2, introduced in 2003, is similarly modeless and applicable to interrupt handlers; see *MPR 6/17/03-02*, "ARM Grows More Thumbs.")

Of course, chopping 32-bit instructions down to 16 bits entails some compromises. Most 16-bit instructions can access only eight of the 32 general-purpose registers (GPR) in the MIPS architecture, because the abbreviated instructions have only three register-address bits instead of five. This compromise saves six bits if an instruction accesses

three registers—two registers for input operands and one register to store the result. A few 16-bit instructions can access 16 of the 32 registers.

MIPS16e instructions can address only eight GPRs, so this compromise is nothing new. MicroMIPS does a little better than MIPS16e, because a few 16-bit instructions can read or write some operands in all 32 registers. The ARM architecture has only 16 GPRs to begin with, even for 32-bit instructions, so microMIPS doesn't suffer in comparison with Thumb-2.

Some 16-bit microMIPS instructions are limited to manipulating fewer operands or smaller immediate values—another common trade-off with shorter instructions. Indeed, in many respects, microMIPS resembles the unified 16/32-bit instruction sets of Thumb-2 and ARC International's ARCompact, which appeared in 2002. (See the sidebar, "ARCompact: An Elegant 16/32-Bit ISA," in *MPR 2/18/03-06*, "Soft Cores Gain Ground.")

New Atomic Instructions

The M14K and M14Kc processors have two additional new instructions that aren't part of the microMIPS ISA. Both are atomic memory operations. They are small but important additions to the MIPS architecture, which has been hampered by historical limitations in this regard. (In the 1980s, strict RISC liturgy frowned on CISC-like instructions that perform multiple operations directly on data in memory.) The new atomic instructions perform read-modify-write operations on memory and cannot be interrupted.

One new atomic instruction, ASET, sets an individual bit within a byte. The instruction format includes an offset from the memory address, so any bit within a 32-bit memory location can be flipped from 0 to 1. Until now, an equivalent operation required three instructions (load a byte, set the bit by applying an OR mask, save the byte), which were interruptible. The other new atomic instruction, ACLR, clears any bit within a byte. It, too, replaces three instructions (load a byte, apply an AND mask, save the byte) and is uninterruptible.

The ASET and ACLR instructions work by automatically disabling interrupts while they're busy. They may be unsuitable for some hard real-time code that absolutely can't wait for the instruction to finish an operation, particularly if the memory is slow. Generally, however, the atomic instructions will be useful when a program must modify data in memory without fear that an interrupt will override the operation and leave the data in an uncertain state.

New instruction sets require new software-development tools. CodeSourcery's SG++ GNU-based tools now support both MIPS32 and microMIPS. Programmers can use a traditional command-line user interface or an Eclipse integrated development environment (IDE). Also, MIPS has upgraded its Navigator Integrated Component Suite (ICS) to support microMIPS and MIPS32 in an Eclipse IDE. MIPS offers an instruction-set simulator (IASim), a cycle-accurate simulator (CASim), and an SoC development board with Xilinx FPGAs for the M14K and M14Kc processors.

In addition, MIPS has enhanced its System Navigator (a debug probe and profiler) to support both new processors, whose debug features are improved over existing MIPS processors. Among other things, the EJTAG debug channel works at speeds up to 50MHz, and new trace options help programmers find elusive bugs hidden in the mass of trace data.

M14K: Fast MCU Core

Figure 2 is a block diagram of the M14K. It's a small, cacheless core ideal for 32-bit microcontrollers, but it's not a stripped-down model, and some features are configurable.

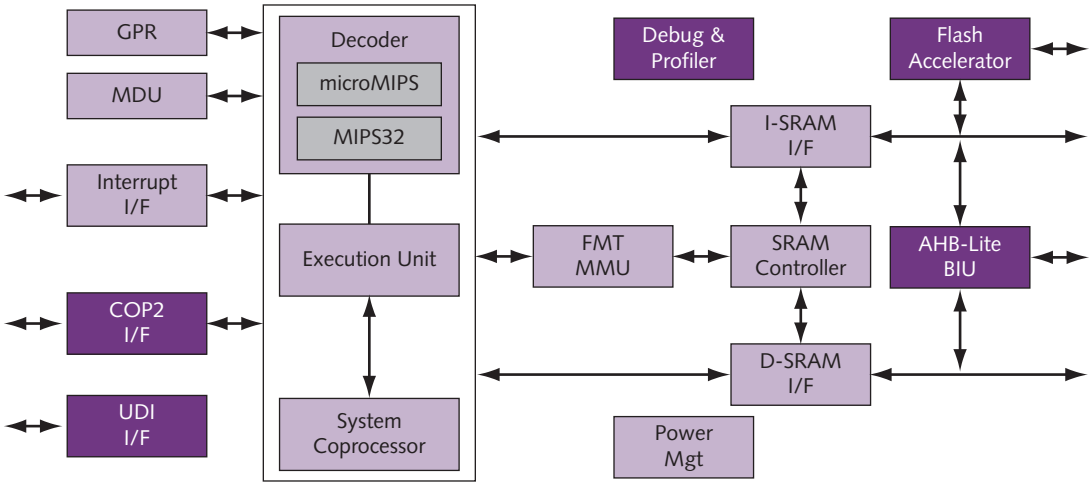

Figure 2. MIPS32 M14K processor block diagram. Parity-protected local memories substitute for L1 caches, so real-time applications can rely on deterministic behavior. Separate instruction decoders handle existing MIPS32 instructions and new microMIPS instructions. Chip designers can choose between a fast or small 32- x 16-bit multiply-divide unit (MDU). Optional features, shown in dark purple, include a debug unit, flash-memory accelerator, AHB-Lite bus controller, coprocessor interface (COP2), and user-defined instruction (UDI) interface.

Although it lacks the full-fledged MMU found in the M14Kc, it does have a simple memory manager with fixed mapping instead of a TLB. Chip designers can choose from two hardware multiply-divide units: "small" or "fast." The fast one multiplies 32- x 16-bit operands in one clock cycle and 32- x 32-bit operands in two cycles. Division has a latency of 12 to 33 cycles.

Another configurable feature is the register set. The MIPS32 standard is 32 registers, each 32 bits wide. Chip designers can implement additional sets of shadow registers for faster context switching. Instead of dumping registers to memory and restoring them after each switch, the processor can instantly change a pointer to a shadow set that holds the register values of the other context. Both the M14K and M14Kc processors allow developers to implement 1, 2, 4, 8, or 16 sets of 32 registers.

Optional features of the M14K processor include debug logic, an AMBA AHB-Lite bus controller, a coprocessor interface, a user-defined instruction (UDI) interface, and a flash-memory accelerator. The AHB-Lite controller has a unified memory interface for both instructions and data, instead of the separate instruction and data interfaces found on some other MIPS processors. Two unidirectional 32-bit channels handle loads and stores. The M14K has a "modified Harvard" memory architecture, because the unified bus splits into separate internal datapaths feeding two local memories for instructions and data. Each internal SRAM has 32-bit addressing, so (in theory) each memory can be as large as 4GB.

The optional coprocessor interface is compatible with the MIPS-standard COP2 interface found on other MIPS processors. It's 32 bits wide, and it allows chip developers to attach additional processor cores or application-specific logic. The M14K implementation isn't backward compatible with COP1, which was designed for a floating-point math coprocessor.

The optional UDI interface supports the MIPS CorExtend technology, which allows developers to add application-specific instructions. Although CorExtend doesn't make the M14K core as configurable as the processors from ARC and Tensilica, it's quite powerful, and it's an important feature missing from ARM processors. (See *MPR 3/3/03-01*, "MIPS Embraces Configurable Technology.")

Figure 3 shows another valuable option for the M14K—a flash-memory accelerator. This block will be especially appreciated in microcontrollers that store application code in slow, nonvolatile flash memory. Essentially, the accelerator is a configurable prefetch buffer in SRAM. Chip designers can implement two cache lines. Each line can be 32, 64, or 128 bits deep. Base addresses are programmable, so each line can point to any region of flash memory.

The prefetch buffer can load instructions faster than the processor can drain it. For example, assume that a 100MHz M14K processor is fetching a stream of eight instructions (including two loads or stores) from 50ns (20MHz) flash memory. Without prefetch, the processor averages 4.2 clock cycles per instruction fetch. With prefetch, and assuming a 100% hit rate in the buffer, the processor fetches one instruction every cycle. Assuming a 75% hit rate, the processor averages 1.8 cycles per instruction. Assuming a 50% hit rate, the average is 2.6 cycles per instruction. The flash accelerator is a relatively small feature, but it can improve a microcontroller's responsiveness.

Interrupting the ARM Race

In another bid to make its processors snappier, MIPS has significantly improved the response times for interrupts in both the M14K and M14Kc cores. MIPS processors aren't particularly sluggish in this regard, but ARM has been making similar improvements to Cortex-family cores intended for microcontrollers, so MIPS has to stay in the arms race. Faster interrupts will make the M14K and M14Kc more suitable for real-time systems.

Three enhancements make the difference. First, the M14K and M14Kc can prefetch the target address of an interrupt handler, saving precious time when the interrupt triggers. Second, the new processors adjust their stack pointers and perform other related chores in hardware instead of in software. Third, a new instruction (IRET) is a special return-from-interrupt that's better for this purpose than the usual ERET instruction. The IRET instruction explicitly supports the chaining of multiple interrupts. Figure 4 illustrates this process.

In the older M4K and 4KEc processors, the latency for the interrupt prologue and chaining is 34 clock cycles, and the epilogue requires another 32 cycles. That's a total of 66 cycles for two chained interrupts. In the new M14K and M14Kc processors, the latency for the prologue and chaining is 17 cycles, and the epilogue takes four cycles. That's a total of 21 cycles—more than three times faster.

ARM's Cortex-M0 and Cortex-M3 have similar interrupt latencies, but comparisons are tricky. As we noted in

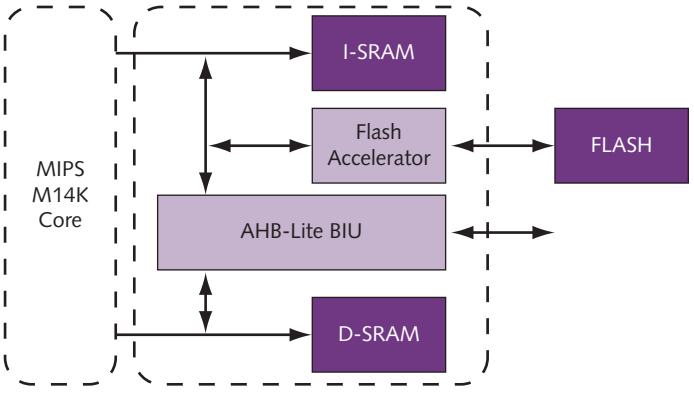

Figure 3. M14K processor flash-memory accelerator. This optional block adds an SRAM prefetch buffer to the M14K core. The two-line buffer is configurable to depths of 32, 64, or 128 bits. MIPS says the buffer can make flash memory seem four times faster. Performance scales even better at higher core frequencies.

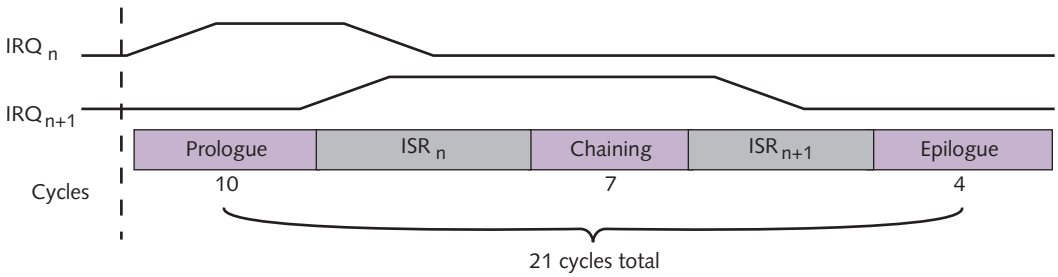

Figure 4. Interrupt chaining in the MIPS32 M14K and M14Kc processors. The first interrupt, IRQ_n, is quickly followed by a second interrupt, IRQ_{n+1}. The new IRET instruction jumps directly from IRQ_n to IRQ_{n+1} without returning to the main instruction stream. When the last interrupt in the chain terminates, IRET returns to the mainline code.

our March 2009 report on the Cortex-M0, the comparison depends on what's included in the calculation. By our count, the Cortex-M0 can handle a chained interrupt in 22 cycles, and the Cortex-M3 can handle a chained interrupt in 18 cycles. Our comparison (based on vendor-supplied data) may not be perfectly cycle-accurate, but it does appear that the latest MIPS and ARM cores for microcontrollers have similar interrupt latencies. (See *MPR 3/2/09-01*, "ARM's Smallest Thumb.")

Natively, the M14K and M14Kc cores support a vectored interrupt mode with only eight sources. To get more, chip designers must add an external interrupt controller, which can support as many as 256 sources. The M14K and M14Kc can also support up to 256 interrupt priority levels, versus 64 in the M4K and 4KEc processors. Any interrupt can switch to a shadow register set, which saves even more time, because the processor needn't flush and restore its registers before entering the interrupt handler.

MIPS has bundled all these improvements—shorter interrupt latencies, faster chaining, the IRET instruction, support for 256 priorities, and the new atomic instructions—into a new Microcontroller Application-Specific Extension (MCU-ASE) that may appear in other MIPS processors.

M14Kc Adds MMU and Caches

The M14Kc core is much like the M14K core, but it's designed for somewhat higher-end applications. It has a full-fledged MMU with a TLB, so it can manage embedded operating systems that address virtual memory. The TLB actually comprises three buffers: a unified TLB with 16 or 32 entries (configurable); a four-entry instruction TLB; and a four-entry data TLB. All are fully associative. The unified TLB can map up to 64 pages of virtual memory, and page sizes can range from 1KB to 256MB. Chip designers can replace the TLB with the same fixed-map MMU found in the M14K core, but that option would negate the primary advantage of the M14Kc.

Another difference between the M14Kc and its little brother is the first-level memory system. Instead of local memories, the M14Kc has configurable instruction and data caches, up to 64KB each. Caches can be one-, two-, three-, or four-way set-associative. Each way can be 1KB, 2KB, 4KB, 8KB, or 16KB. Parity bits help detect soft errors.

To supplement the caches, chip designers can add separate scratchpad RAMs for instructions and data. Each scratchpad has its own 32-bit I/O interface to local on-chip memory, bypassing the AHB-Lite bus. This arrangement provides fast, no-miss access and deterministic behavior—critical features in real-time applications. The scratchpads are configurable; each can be as large as 1MB. Optionally, a portion of scratchpad memory can replace one

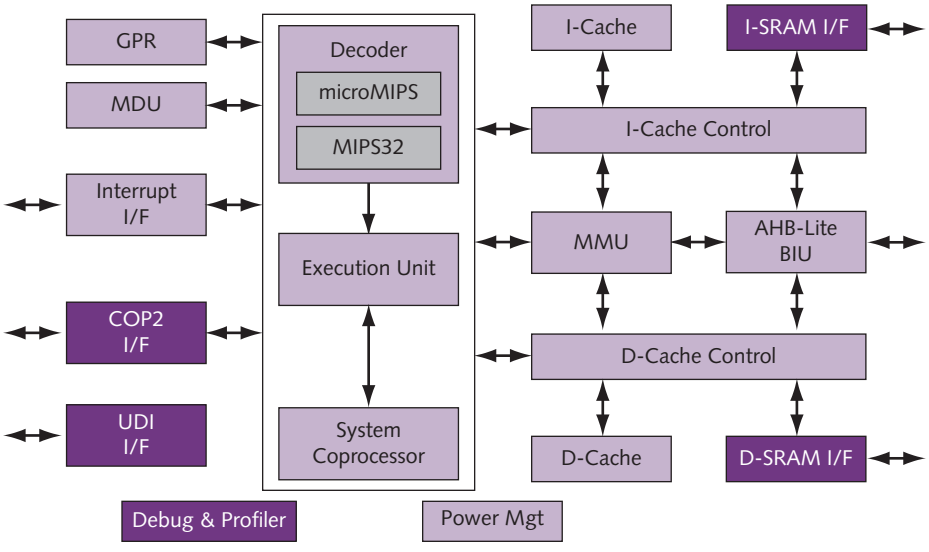

Figure 5. MIPS32 M14Kc processor block diagram. This core has much in common with the simpler M14K core. The main differences are parity-protected caches and a full-fledged MMU for running higher-end embedded operating systems. Optional features, shown in dark purple, include debug logic, dual scratchpad RAMs, the MIPS COP2 coprocessor interface, and CorExtend user-defined instructions (UDI).

way of a cache (up to 16KB). Figure 5 is a block diagram of the M14Kc core.

The AMBA AHB-Lite bus controller is a standard feature of the M14Kc, not an option. Like the controller in the M14K, it has unidirectional 32-bit read and write channels that fetch instructions and data from unified memory. Once fetched, instructions and data follow different datapaths to the caches. By including AHB-Lite, MIPS is acknowledging the rising popularity of this bus interface, which originated at ARM. Fewer SoC designs seem to be using the Open Core Protocol (OCP) or IBM CoreConnect buses these days.

Comparing Power and Performance

Power-management features and configurable clock gating help reduce the power consumption of both new processors. These cores are small to begin with, occupying significantly less than one square millimeter of silicon even when optimized for speed and manufactured in an older 0.13-micron CMOS process. (It's still a common process for microcontrollers.)

Using TSMC's 130nm-G process, MIPS estimates that the M14K core will require only 0.35mm^2 of silicon when synthesized for minimum area and only 0.68mm^2 when synthesized for maximum speed. The worst-case maximum clock frequencies for those configurations will be 100MHz and 180MHz, respectively. At those frequencies, performance will range from 150 Dhrystone mips to 270Dmips. MIPS estimates that typical power consumption (as measured under simulation while running Dhrystone) will be 12mW for the area-optimized configuration and 39.6mW for the speed-optimized configuration.

In a more up-to-date (but hardly cutting-edge) TSMC 90nm-G process, the M14K core is even smaller. An area-optimized configuration will occupy only 0.21mm^2 of silicon, deliver 290Dmips at 193MHz, and consume only 11.6mW. A speed-optimized configuration will occupy 0.51mm^2, deliver 442Dmips at 295MHz, and consume only 35.4mW. Of all the aforementioned configurations, the area-optimized core in 90nm is the most energy efficient by far, delivering 25Dmips per milliwatt.

The M14Kc core is larger and more power hungry, but only in relative terms. In TSMC's 90nm-G process, an area-optimized configuration requires 0.37mm^2 of silicon, delivers 291Dmips at 194MHz, and consumes 15.5mW. A speed-optimized configuration requires 0.82mm^2, delivers 483Dmips at 322MHz, and consumes 48.3mW. The area-optimized core is the most energy efficient, delivering 18.75Dmips per milliwatt. Table 2 summarizes these estimates and notes the physical libraries used for synthesis.

Comparing MIPS and ARM

MIPS designed the M14K and M14Kc cores to challenge the ARM926EJ-S and Cortex-M3, two popular cores in 32-bit

Feature	MIPS M14K Speed Optimized	MIPS M14K Area Optimized	MIPS14Kc Speed Optimized	MIPS M14Kc Area Optimized
TSMC 90nm-G				
Transistors	Standard Vt	Standard Vt	Standard Vt	Standard Vt
Logic Library	Virage HS	TSMC HP	Virage HS	TSMC HP
Memory Library	Dolphin	Virage HD	Dolphin	Virage HD
Core Frequency	295MHz	193MHz	322MHz	194MHz
Core Area	0.51mm^2	0.21mm^2	0.82mm^2	0.37mm^2
Core Power (typical)	0.12mW / MHz	0.06mW / MHz	0.15mW / MHz	0.08mW / MHz
Dhrystone 2.1	442Dmips	290Dmips	483Dmips	291Dmips
Power Efficiency	12.5Dmips / mW	25Dmips / mW	10Dmips / mW	18.75Dmips / mW
TSMC 130nm-G				
Transistors	Standard Vt	Standard Vt	Standard Vt	Standard Vt
Logic Library	TSMC HP	Artisan	TSMC HP	Artisan Metro
Memory Library	Virage HS	Virage HD	Virage HS	Virage HD
Core Frequency	180MHz	100MHz	205MHz	100MHz
Core Area	0.68mm^2	0.35mm^2	1.29mm^2	0.61mm^2
Core Power (typical)	0.22mW / MHz	0.12mW / MHz	0.41mW / MHz	0.14mW / MHz
Dhrystone 2.1	270Dmips	150Dmips	307Dmips	150Dmips
Power Efficiency	6.8Dmips / mW	12.5Dmips / mW	3.6Dmips / mW	10.7Dmips / mW

Table 2. MIPS32 M14K and M14Kc processor power/performance comparison. All numbers are MIPS estimates based on simulations with preliminary RTL. Although this table specifies the physical libraries that MIPS used when compiling the SRAMs, the silicon-area measurements are for the CPU cores only, not including caches or local memories. Ditto for the power estimates, which represent "typical" consumption when running Dhrystone 2.1 with minimal I/O.

Feature	MIPS M14K	MIPS M14Kc	MIPS M4K	MIPS 4KEc (Soft)	ARM Cortex-A5	ARM Cortex-M3
CPU ISA	MIPS32 R2	MIPS32 R2	MIPS32 R2	MIPS32 R2	ARMv7-A	ARMv7-M
Arch. Width	32 bits	32 bits	32 bits	32 bits	32 bits	32 bits
16-Bit Instr.	microMIPS	microMIPS	MIPS16e	MIPS16e	Thumb, Thumb-2	Thumb, Thumb-2
Related Core	MIPS32 M4K	MIPS32 4KEc	MIPS32 4K	MIPS32 4KE	Cortex-A8	—
GPUs	32 x 32 bits 1–16 sets	32 x 32 bits 1–16 sets	32 x 32 bits 1–16 sets	32 x 32 bits 1–16 sets	16 x 32 bits	16 x 32 bits
Pipeline Depth	5 stages	5 stages	5 stages	5 stages	8 stages	3 stages
Branch Pred.	—	—	—	—	Dynamic	Speculation
L1 Cache (I / D)	—	0–64K each	—	0–64K each	4K–64K each	—
L2 Cache	—	—	—	—	Optional 16K–8MB	—
Instr. RAM	0–4MB	0–1MB	0–4GB	0–1MB	—	0–1MB
Data RAM	0–4MB	0–1MB	0–4GB	0–1MB	—	0–1MB
Memory Management	MMU (Fixed map)	MMU (TLB)	MMU (Fixed map)	MMU (TLB)	MMU (TLB)	Optional MPU (8 regions)
Coherent MP	—	—	—	—	Optional 1–4 cores	—
Hardware Multiplier	Fast or small 32 x 16 bits	Fast or small 32 x 16 bits	Fast or small 32 x 16 bits	Fast 32 x 16 bits	Single-cycle 32 x 32 bits	Single-cycle 32 x 32 bits
External Interrupts	8 (vectored) 256 (external)*	8 (vectored) 256 (external)*	6 (vectored) 256 (external)*	6 (vectored) 256 (external)*	n/a	Configurable 1–240
Interrupt Latency	21 cycles (chained)	21 cycles (chained)	66 cycles (chained)	66 cycles (chained)	n/a	18 cycles (chained)
Interrupt Priorities	256	256	64	64	n/a	n/a
Privilege Modes	2	2	2	2	2 + TrustZone	2
Custom Extensions	Optional (CorExtend)	Optional (CorExtend)	Optional (CorExtend)	Optional (CorExtend)	—	—
System Interface	MIPS SRAM 1 or 2 x 32 bits or AHB-Lite 2 x 32 bits	AHB-Lite 2 x 32 bits	MIPS SRAM 1 or 2 x 32 bits	MIPS BIU 32 bits	AMBA-3 AXI 1 x 64 bits Optional 2 x 64 bits (multiprocessor)	AHB-Lite 2 x 32 bits
Coprocessor Interface	MIPS COP2 32 bits	MIPS COP2 32 bits	MIPS COP2 32 bits	MIPS COP2 32 bits	—	—
Core Frequency (Max)	295MHz (90nm-G)	322MHz (90nm-G)	200–414MHz (90nm-G)	250–420MHz (90nm)	480MHz–1.0GHz (40nm-LP, 40nm-G)	50–191MHz (90nm-G)
Core Area @ Max Freq	0.51mm^2 (90nm-G)	0.82mm^2 (90nm-G)	0.12–0.53mm^2 (90nm-G)	0.65–1.2mm^2 (90nm, 8K caches)	0.27mm^2 (40nm-LP)	0.25–0.37mm^2 (90nm-G)
Dhrystone 2.1	1.5Dmips / MHz	1.5Dmips / MHz	1.6Dmips / MHz	1.6Dmips / MHz	1.57Dmips / MHz	1.25Dmips / MHz
Power (typical)	35.4mW (90nm-G)	48.3mW (90nm-G)	8.0–62.1mW (90nm-G)	37.5–109.2mW (90nm)	57.6mW (40nm-LP)	2.0–13.3mW (90nm-G)
Power Efficiency Dmips / mW	12.5 (90nm-G)	10.0 (90nm-G)	10–40 (90nm-G)	6.1–10.6 (90nm)	13.0 (40nm-LP)	31.2–17.9 (90nm-G)
Introduction	Feb 2010	Feb 2010	2002	2003	2009	2004

Table 3. Feature comparison of the MIPS32 M14K, M14Kc, M4K, 4KEc, ARM Cortex-A5, and Cortex-M3 cores. The MIPS M14K will compete with the ARM Cortex-M3 for microcontroller designs. The MIPS M14Kc will compete with the ARM Cortex-A5 for SoC designs requiring a full-fledged MMU to run higher-end embedded operating systems. When comparing these performance specifications—all of them estimates provided by the vendors—mind the differences in fabrication processes. *MPR* was able to obtain estimates at 90nm for all cores except the new Cortex-A5, which is specified in a 40nm low-leakage process. *The MIPS processors require an external interrupt controller to support 256 interrupts. (n/a: data not available.)

microcontrollers and SoCs. Meanwhile, ARM was working on the Cortex-A5, which was announced less than two weeks before MIPS unveiled its new processors. The M14K and M14Kc look good against the existing ARM cores, generally beating them in clock-frequency headroom, size, and power. The Cortex-A5 is stiffer competition but not a clear winner.

Table 3 summarizes the features of the MIPS32 M14K and M14Kc cores and compares them with the new ARM Cortex-A5 and existing Cortex-M3. This table also includes the MIPS32 M4K and 4KEc processors that the new MIPS cores supersede. As usual, it's almost impossible to fairly compare these vendor-supplied power and performance estimates, because they assume different fabrication processes and other unknown variables (exact core configurations, logic-synthesis scripts, cell libraries, etc.). Nevertheless, a few things stand out.

The new microMIPS 16/32-bit instruction set should erase the code-density handicap that has dogged MIPS since ARM introduced the 16/32-bit Thumb-2 instruction set in 2003. Or, to put it another way, microMIPS should erase the performance handicap that dogged MIPS when programs used the MIPS16e extensions, which sap more throughput than Thumb-2 does.

The new MIPS processors significantly improve their interrupt-handling performance and flexibility, achieving approximate parity with ARM's processors. Code density and interrupt handling are especially important for microcontrollers, a renewed target for MIPS. Even if MIPS doesn't clearly beat ARM in every category, reaching parity is a worthy accomplishment.

MIPS has two advantages that can boost performance without cranking up the clock speed, but they demand extra effort from developers. One advantage is register shadowing. By configuring an M14K or M14Kc core with additional sets of GPRs—up to 16 sets of 32 registers—developers can accelerate context switching and interrupt handling. ARM's single set of only 16 registers seems claustrophobic in comparison. Another MIPS advantage is CorExtend, which lets designers add application-specific instructions. CorExtend requires diligent performance profiling and RTL handiwork, but the payoffs can be enormous.

ARM has advantages, too. The Cortex-A5 leaves behind the aging AHB bus and adopts the latest AMBA-3 AXI standard. AXI is more efficient than AHB-Lite, which MIPS offers for the M14K and M14Kc. And, for higher-performance applications, the Cortex-A5 offers the option of symmetric multiprocessing with up to four cores and memory coherency.

Although the MIPS COP2 interface makes multicore designs possible, intercore communications and memory coherency are exercises for the designer, not off-the-shelf features. To match the multicore flexibility of the Cortex-A5, MIPS developers must step up to the MIPS32 1004K multiprocessor core. (See *MPR 4/28/08-01*, "Multicore Multithreading With MIPS.")

MIPS Fares Well Against ARM

Perhaps the most surprising result of comparing the new MIPS processors with ARM's best cores is that ARM no longer has a clear advantage in power consumption, core area, and performance. Usually, those are ARM's strengths.

For instance, using the data in Tables 2 and 3, we can compare the two microcontroller cores—the MIPS M14K and ARM Cortex-M3—in the same TSMC 90nm-G process. An area-optimized M14K will consume 11.6mW at 193MHz in 0.21mm^2 of silicon. A speed-optimized Cortex-M3 will consume 13.3mW at 191MHz in 0.37mm^2 of silicon. The M14K requires less power and silicon at virtually the same clock frequency. In power efficiency, the M14K wins, too: 25Dmips per milliwatt versus 17.9Dmips per milliwatt.

Note that we compared an area-optimized M14K with a speed-optimized Cortex-M3. That's because a speed-optimized M14K can reach a much higher clock frequency (295MHz). Assuming the two processors are clocked to deliver similar performance, the M14K will use less power and silicon. (The M14K has a throughput advantage of 1.5Dmips per megahertz versus 1.25Dmips per megahertz for the Cortex-M3.)

ARM says an area-optimized Cortex-M3 consumes a mere 2.0mW. But it would be a trifle larger than the M14K core (0.25mm^2 versus 0.21mm^2), and ARM specifies a clock frequency of only 50MHz. In contrast, MIPS specifies 193MHz for its area-optimized M14K core in the same process. Slowing the M14K core to 50MHz would drop power consumption to about 3.0mW, only a milliwatt more than the Cortex-M3. Realistically, a developer targeting a 90nm process probably needs more than 50MHz for the target application.

Comparing the M14Kc with the new Cortex-A5 is almost pointless, because we have 90nm data for the MIPS processor and 40nm data for the ARM processor. That's a difference of two process generations. Also, the MIPS32 24K or 1004K processors are probably better matches for the Cortex-A5, in terms of features.

Nevertheless, the M14Kc looks pretty good. Fabricated in 90nm-G, a speed-optimized M14Kc consumes 48.3mW at 322MHz in 0.82mm^2 of silicon. Fabricated in 40nm-LP, even an area-optimized Cortex-A5 consumes more power: 57.6mW versus 48.3mW. The Cortex-A5 might fare better when fabricated in 40nm-G, which is leakier than 40nm-LP but uses less dynamic power.

As we would expect from the huge difference in process technologies, the 40nm Cortex-A5 is much smaller (0.27mm^2 versus 0.82mm^2) and faster (480MHz versus 322MHz) than the 90nm M14Kc. Assuming two process shrinks to 40nm, the M14Kc should be very competitive with ARM's latest core.

Competition Heats Up

Historically, ARM tends to have the smallest, lowest-power processors, whereas MIPS tends to excel in high performance.

Price & Availability

The MIPS32 M14K and MIPS32 M14Kc processors are synthesizable cores delivered in Verilog format. MIPS expects to ship the final RTL in February 2010. MIPS doesn't publicly disclose licensing fees or royalties. For more information:
www.mips.com/products/processors/32-64-bit-cores/mips32-m14k/
www.mips.com/products/processors/32-64-bit-cores/mips32-m14kc/

To some extent, those tendencies reflect the history of each company. Almost from the start, ARM focused on embedded systems, whereas MIPS originally focused on workstations and servers.

Lately, ARM has been reaching toward higher performance, because that is where ARM's biggest market (mobile phones) is going. Meanwhile, MIPS is migrating toward lower power, because that's where its biggest market (consumer electronics) is going. At the same time, both companies fear encroachment by the x86, because mobility is where Intel's biggest market (personal computing) is going.

For the harried engineers at ARM, MIPS, and Intel, life is getting tougher. But it's good news for chip designers, who are getting more and better choices. Embedded-processor cores are growing more powerful and power efficient. The MIPS M14K and M14Kc are arriving hot on the heels of the ARM Cortex-A5 and Tensilica's new Xtensa LX3 and Xtensa 8 (which *MPR* will cover soon). Another competitor, ARC International, was recently acquired by Virage Logic, which can offer one-stop shopping for SoC designers. (See *MPR 9/14/09-01*, "Summer Shopping Spree.")

The M14K and M14Kc processors—and, especially, the microMIPS ISA—move MIPS Technologies in a direction the company needs to go. They are worthwhile upgrades from existing MIPS processors and the MIPS32 ISA. They will strengthen MIPS's competitive position against ARM, its strongest foe.