Reed Electronics Group

MICROPROCESSOR REPORT

www.MPRonline.com

THE INSIDER'S GUIDE TO MICROPROCESSOR HARDWARE

UBICOM'S NEW NPU STAYS SMALL

IP3023 Packet Processor Has Efficient I/O Architecture

By Tom R. Halfhill {4/21/03-01}

Small is good if you're a jockey, a designer dog, or a microprocessor chip. Ubicom (meaning "ubiquitous communications") is a company that definitely thinks small when it designs packet processors for wired and wireless systems. Its latest NPU is the IP3023, which requires only about 50% as much silicon and 10% as much memory as some competing chips.

Ubicom designed the IP3023 for wireless access points, wireless LAN (WLAN) bridges, broadband modems, home routers, and other consumer or enterprise products that operate near the edge of a network. The company's goal is to slash the bill of materials (BOM) for those systems by offering an efficient packet processor that reduces or eliminates the need for off-chip memory and protocol-specific I/O chips.

The 32-bit IP3023 is the first implementation of Ubicom's next-generation processor architecture. Priced at only $12, it follows Ubicom's even more economical 8/16-bit IP2022 and IP2012 packet processors. (See *MPR 5/28/02-03*, "Ubicom Breaks New NPU Ground.") For $1 less than the price of the IP2022 at introduction two years ago, the IP3023 delivers many more features and 10 times more performance, as Table 1 shows.

All Ubicom NPUs reflect the company's philosophy that conventional microprocessor architectures are not optimal for embedded communications. Microprocessor architects conceived CISC, RISC, and VLIW decades ago for completely different systems and tasks: personal computers, workstations, servers, scientific computing, productivity software, and so forth. Although general-purpose architectures can, by definition, do virtually anything—often with the help of extensions—they were not designed for embedded communications. Furthermore, they often require additional chips to support common I/O and communications protocols such as USB, Ethernet, PCMCIA, and 802.11. Consequently, systems based on general-purpose processors tend to be more costly, occupy more board space, and consume more power than systems using application-specific chips.

ASICs and SoCs can be more efficient than general-purpose processors, but spinning a custom chip entails significant development risk, and it's usually built around microprocessor cores having the same CISC or RISC architectures as general-purpose processors. Moreover, ASICs and SoCs with application-specific logic are less flexible, even if they are programmable. As communication protocols like 802.11 keep evolving, systems that rely on custom parts with dedicated peripheral logic will become obsolete.

Ubicom's solution was to create both a new microprocessor architecture specifically for embedded communications and a new architectural genre akin to CISC or RISC. Ubicom refers to the new genre as MASI (multithreaded architecture for software I/O). The key elements of MASI are direct-memory instructions that avoid manipulating packet data in on-chip registers and buffers; deterministic hardware multithreading, which mixes instructions from multiple threads together in a single pipeline for rapid context switching; and the ability to implement popular I/O protocols in software instead of hardware.

So far, Ubicom has created two proprietary microprocessor architectures based on the MASI concept. The older IP2000 architecture is the foundation of the 8-bit IP2022 and IP2012 chips; the new (as-yet-unnamed) architecture underpins the 32-bit IP3023. Ubicom says other companies will eventually adopt some form of MASI, just as they have broadly adopted CISC, RISC, and VLIW for other purposes. Even if that prediction turns out to be a hollow marketing pitch, MASI seems to be working for Ubicom: the company has 23 publicly announced customers in the market for network devices and WLANs, including Askey, D-Link, InFocus, Linksys, NEC, and Siemens.

Minimizing Off-Chip Memory

A generation of engineers raised on RISC may find Ubicom's new architecture a little odd, but not illogical. Indeed, in most ways, it resembles RISC: it has fixed-length 32-bit instructions, single-cycle pipelined execution, and a simple instruction set (only 39 unique mnemonics, with variations for different operand types).

The IP3023 architecture departs from RISC in the canonical aspect of memory access. RISC architectures use load/store instructions to move operands in and out of registers from memory, and they use separate instructions to manipulate the operands in the registers without touching memory. In contrast, most instructions in Ubicom's architecture manipulate operands directly in memory, without ever moving the operands into registers.

Ubicom refers to its approach as a memory-to-memory architecture. At first glance it seems like a yellowed page ripped from the databook of an old CISC processor. Classic CISC architectures born in the 1970s—such as the Intel x86 and Motorola 68K—also have instructions that perform arithmetic and logical operations directly on operands in memory. (Modern compilers, however, avoid using those instructions.) Direct-memory instructions made sense in the old days, when RAM was as fast as CPUs were, because there was no penalty for referencing off-chip memory. In the 1980s, RISC repudiated direct-memory instructions and introduced instruction sets that segregate load/store instructions from other types of operations. That made sense, too, as CPU clock frequencies outpaced RAM latencies, and superscalar pipelines imposed new complexities on instruction scheduling.

But just as women's skirts grow longer or shorter with the changing times, unfashionable ideas about microprocessor architectures sometimes become fashionable again. Ubicom argues that direct-memory instructions make sense for a packet processor. Network packets can be hundreds of bytes long, and they continuously stream through the processor as it checks for transmission errors and performs other relatively simple tasks. Usually, a packet processor—especially one in a network-edge device instead of in a core router—need not perform multiple operations on a single chunk of packet data. Therefore, it's unnecessary to hold a copy of the data in registers. RISC architectures designed for general-purpose computing are more likely to manipulate the same data with multiple instructions.

A typical example of an IP3023 memory-to-memory instruction is `add.4`, which adds two 32-bit values and stores the result. The format of this instruction—`add.4 d,s1,s2`—looks like a typical RISC format, with specifiers for two source registers (the operands to be summed) and a destination register (to store the result without overwriting the operands). In this case, however, the specifiers refer to real memory addresses, not to registers. The IP3023 has multiple addressing modes that can use immediate values or the contents of registers as offsets, with optional increments. Table 2 lists the complete IP3023 instruction set.

When the IP3023 does need to perform multiple operations on data, it can use registers as any RISC processor does, thanks to variants of the basic instructions. The IP3023 has 16 general-purpose registers, 32 bits wide, plus a special "source-3" register for instructions that need to manipulate a third 32-bit operand. There are also eight 32-bit memory-address registers and a 48-bit accumulator for the multiply-accumulate (MAC) instruction. As we'll explain later, the IP3023 has eight duplicate copies of this register file.

	Ubicom IP3023	Ubicom IP2022	Ubicom IP2012
Architecture Family	IP3000	IP2000	IP2000
Datapaths	32-bit	8-bit	8-bit
Memory Addressing	32-bit	16-bit	16-bit
Pipeline Depth	10 stages	4 stages	4 stages
Hardware Threads	8	2	2
Context Switch	0 clocks	3 clocks	3 clocks
Interrupt Latency	1 clock	3 clocks	3 clocks
Memory Architecture	Harvard	Harvard	Harvard
On-Chip Memory:			
Flash ROM	—	64K	64K
Program SRAM	256K	16K	16K
Data SRAM	64K	4K	4K
Ext Memory Ctrl	Flash/SDRAM	SRAM	—
Off-Chip Memory:			
Program (max)	4MB Flash	64K	—
Data (max)	8MB SDRAM	128K	—
MII Ports	4	0	0
Serdes Units	2	2	1
I/O Ports	5–7	2–3	2–3
GPIO (max)	106	52	48
Timers	9+1 watchdog	4+1 watchdog	4+1 watchdog
HW Rnd Generator	Yes	No	No
IC Process	0.13µm	0.25µm	0.25µm
Package	208-pin PQFP	80-pin PQFP	80-pin PQFP
Clock Speed	250MHz	120MHz/ 160MHz	120MHz
Relative Performance*	10x	1x	1x
Power (typ)	<1W @ 1.2V	300–900mW @ 2.7V	300–900mW @ 2.7V
Power (sleep)	<1mW	<1mW	<1mW
Availability	2H03	Now	Now
Price (100K units/mo)	$12	$8/$9	$7

Table 1. Ubicom's new IP3023 is a major upgrade from the existing IP2022 and IP2012 in almost every category. *Ubicom bases its performance estimate on the greater architectural efficiency of the 32-bit IP3023, not just on the 56% higher clock frequency.

Ubicom says the greatest benefit of memory-to-memory operations lies in eliminating pairs of load/store instructions and their associated increment instructions when moving packet data through the processor. The IP3023 can replace three or four RISC instructions with a single auto-incrementing `move` instruction. Optimizing RISC compilers can eliminate some stores by reusing data already in registers, but Ubicom's compiler can make similar optimizations by using the register variants of instructions when the processor needs to perform multiple operations on data.

A big reason that RISC abandoned direct-memory instructions is the memory latency problem: today's processors are much faster than DRAM chips. That's a small issue with the IP3023, however, which is designed to store all system and applications software in on-chip memory whenever possible. Using a RISC-like Harvard memory architecture, the IP3023 has 256K of SRAM for program instructions and 64K of SRAM for data—all accessible in two clock cycles. Ubicom provides a very small real-time operating system (RTOS) for the IP3023, so the on-chip memory is sufficient for most of the processor's target applications. For larger applications, the IP3023 has an 8-bit interface for up to 4MB of external flash memory (for storing program code) and a 16-bit interface for up to 8MB of external SDRAM (for storing data). The processor can execute code directly from flash.

With generous amounts of SRAM on chip, the IP3023 doesn't need instruction and data caches or a memory-management unit (MMU). Omitting those features conserves silicon while eliminating sources of nondeterministic behavior that impair real-time response. Figure 1 shows a block diagram of the IP3023.

Omitting the caches also eliminates another place where, in other processors, multiple copies of packet data accumulate. Ubicom notes that a typical packet processor often maintains four copies of the data it is processing: incoming data in off-chip memory; another copy in the media-access controller's first-in, first-out (FIFO) input buffer; a third copy in the processor's data cache; and a fourth in the media-access controller's FIFO output buffer.

In contrast, the IP3023 maintains only one copy of packet data: the original data that flows from the network into the chip's internal memory, where instructions operate directly on the data. Hence, there are no memory contentions that could interfere with real-time response. The IP3023's two serializer/deserializer (serdes) units have FIFO buffers that hold only four bytes each, compared with the 3K FIFO buffers (1.5K input and output) typically found in other packet processors.

Zero-Cycle Context Switching

Although Intel trumpets the Pentium 4's Hyper-Threading technology, which can mix instructions from two different threads of execution in the same pipeline, Ubicom's IP3023 supports a similar form of hardware multithreading that can mix instructions from eight threads in a pipeline. This allows the IP3023 to switch contexts on any instruction with a zero-cycle penalty. (In this terminology, a "thread" is any process context, such as a coarse-grained program or a fine-grained thread within a program. Software need not be explicitly multithreaded to take advantage of hardware multithreading.)

As with Intel's Hyper-Threading technology, Ubicom's hardware multithreading relies on a duplicate register file and program counter for each context, so the IP3023 has eight register files and program counters. (Ubicom's architecture supports up to 32 simultaneous threads, but the IP3023 implements only eight.) Duplicate register files make it unnecessary to dump and restore the registers for each context switch. The processor merely changes a pointer to reference the register file and program counter for the active thread.

Instruction	Description	Instruction	Description
Arithmetic and Logical Instructions		**bset**	Bit set
add.[2/4]	16/32-bit add	**btst**	Bit test
addc	32-bit add with carry	**lsl.[2/4]**	16/32-bit logical shift left
and.[2/4]	Logical AND	**lsr.[2/4]**	16/32-bit logical shift right
cmpi	16-bit compare immediate	**merge**	32-bit bit-wise merge
crcgen	32-bit incremental CRC generation	**shftd**	64-bit funnel shift
lea.[1/2/4]	Load effective address	**shmrg.[1/2]**	Shift and merge 1 or 2 bytes
pdec	Pointer decrement	**Data Movement and Sign-Extension Instructions**	
mac	16x16-bit multiply-accumulate	**movei**	Move 16-bit immediate
mulf	16x16-bit signed fractional multiply	**moveai**	Move 24-bit immed into bits 30:7 of register
muls	16x16-bit signed integer multiply	**ext.[1/2]**	Sign-extend byte or 16 bits to 32 bits
mulu	16x16-bit unsigned integer multiply	**setcsr**	Set context-switch register
not.[2/4]	Logical NOT	**move.[1/2/4]**	8/16/32-bit move
or.[2/4]	Logical OR	**Control-Flow Instructions**	
sub.[2/4]	16/32-bit subtract	**jmp<cc>.c.t/f**	PC-relative cond jump with branch prediction
subc	32-bit subtract with carry	**call**	PC-relative call to subroutine
xor.[2/4]	Logical exclusive-or	**calli**	Address-indirect call to subroutine
Shift and Bit-Field Instructions		**ret**	Return from subroutine
asr.[2/4]	16/32-bit arithmetic shift right	**suspend**	Suspend current thread until interrupt
bclr	Bit clear	**bkpt**	Breakpoint
bfextu	32-bit bit-field extract	**Program Memory Access Instructions**	
bfrvrs	32-bit bit-field reverse, logical shift right	**iwrite**	32-bit write to program effective address
		iread	32-bit read from program effective address

Table 2. With only 39 unique mnemonics, the IP3023 instruction set is exceptionally small and tightly focused on packet processing. Note the unusual crcgen instruction, which performs a cyclic redundancy check on each byte of packet data—an operation that would normally require four to eight instructions and a 256-byte lookup table. (In this instruction-set table, brackets following mnemonics indicate variations of an instruction. Example: add.2 is a 2-byte add, and add.4 is a 4-byte add.)

Because multiple threads can share the pipeline, a context switch doesn't force the IP3023 to flush instructions from a previous thread out of the pipeline and restart it with instructions from the next thread, as other processors must do. Therefore, no time is wasted repriming the pump. Because the IP3023 lacks an instruction cache, it doesn't have to flush that, either. And because the IP3023 lacks a data cache, there are no issues with "dirty" cache data if one thread tries to manipulate a memory location that hasn't been updated with the latest cached results from another thread.

Thread conflicts are uncommon with the IP3023, because the direct-memory instructions don't hold results in registers and are atomic operations. Those characteristics should greatly reduce the chances of a pipeline-stalling data dependency. Although the latency of the on-chip memory is two cycles, the IP3023 avoids even that small penalty by interleaving instructions from different threads in the pipeline and by passing results to subsequent instructions before storing the results in memory. The pipeline bypassing means an instruction need not wait for a previous instruction to finish its atomic write operation before using the result.

When programmers must explicitly protect a memory location against interference from a different thread, they can use the IP3023's bit-set and bit-clear instructions as spin-lock operations. Those instructions set a condition code that other threads can check to determine if the value in a memory location has been altered.

Threads have two global priority levels (distinct from the multiple priority levels of interrupts). Hard real-time threads handle events that absolutely can't wait, such as the arrival of packet data over the network. Those events require the fastest possible context switching, because, as mentioned above, the IP3023 omits the large FIFO buffers found in other packet processors. For less-critical tasks—such as the operating-system and protocol-stack software—the IP3023 can use lower-priority non-real-time threads that need not be scheduled as often. Those threads can execute when a hard real-time thread is idle, so the processor doesn't waste any cycles.

Many other packet processors use multiple processor cores instead of hardware multithreading for division of labor. Multiprocessing may be a necessity for core routers that demand the highest possible performance. For less-demanding applications nearer the edge of a network, hardware multithreading offers the software illusion of multiprocessing without the additional silicon of multiple processor cores. Furthermore, the ability to interleave instructions from different threads in the same pipeline and bypass the results means the IP3023, unlike a multiprocessor chip, doesn't have to wait while moving data from one processor to another.

Another interesting consequence of hardware multithreading is that the IP3023 rarely needs to stall the pipeline after a branch or waste a NOP in a branch-delay slot. (Indeed, the IP3023 instruction set doesn't even have a NOP.) Most other processors cannot execute the instruction immediately following a branch until the outcome of the branch is known and the processor calculates the branch target address. With the IP3023, an instruction from a different thread can usefully occupy the branch-delay slot, effectively hiding the branch penalty.

If the branch-delay slot contains a consecutive instruction from the *same* thread, however, then a taken branch imposes a penalty. The processor must stop executing the delay-slot instruction and any subsequent instructions from the active thread while loading new instructions from the branch target address. Usually, the IP3023's thread scheduler can avoid this penalty by ensuring that a branch instruction isn't immediately followed by the subsequent instruction from the same thread. When this does happen, however, it is often because there is only one active thread. In that case, the hard real-time threads that handle I/O are idle, so the branch-delay penalty has little or no effect on packet processing. When demand picks up, the IP3023 starts executing multiple threads, so the thread scheduler has more flexibility to avoid branch-delay penalties.

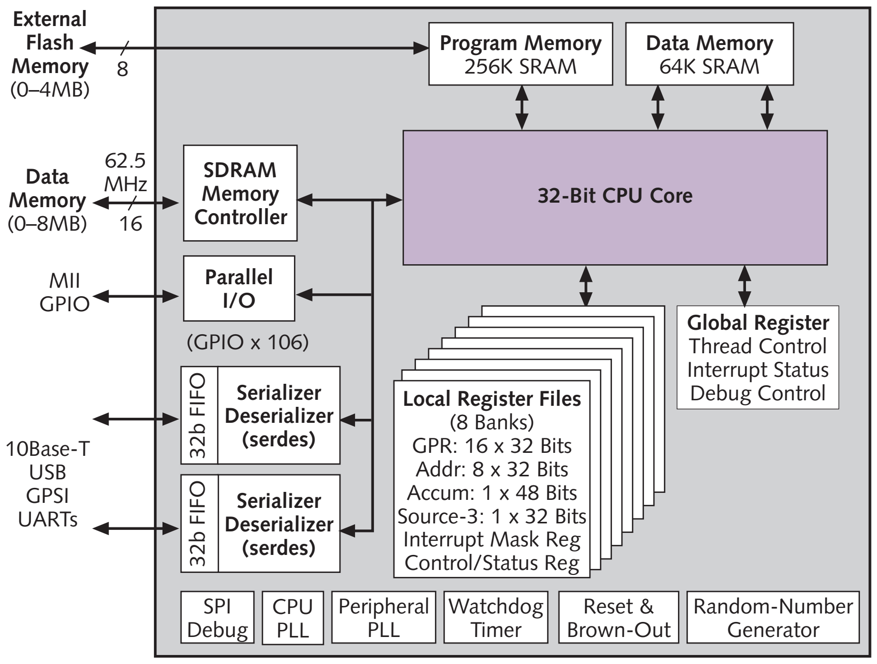

Figure 1. The IP3023 makes caches, MMUs, and DMA controllers redundant because it relies on fast internal memory, although external memory expansion is an option. Internal memory bandwidth is 1GB/s. Most of the 106 GPIO ports are used for software I/O in typical applications. Note the hardware random-number generator, which assists cryptography software.

In accordance with Ubicom's philosophy of keeping things simple and deterministic, the IP3023 doesn't try to reduce the branch-delay penalty with dynamic branch-prediction logic. Instead, the IP3023 uses static branch prediction: the compiler or assembler can set a special bit in the jump instruction to hint whether the branch is likely to be taken. The worst-case penalty for mispredicting a branch is seven cycles, and the average penalty is only two cycles—remarkable in a processor with a 10-stage pipeline and no branch-prediction hardware.

One reason the IP3023's pipeline is so deep is its memory-to-memory architecture, which requires two stages to read the operands from memory instead of from local registers. Hardware multithreading adds another stage at the beginning of the pipeline for scheduling threads. As Figure 2 shows, there are also two instruction-fetch stages and two writeback stages.

There is no load-use penalty with this pipeline, but there is a four-cycle penalty when an instruction needs a memory address that the previous instruction must calculate. Ubicom points out that this happens much less often than load-use penalties in other microprocessor architectures, because loading an address register is less common than loading data from memory. In any case, the `moveai`, `lea`, and `pdec` instructions avoid the address-calculation penalty altogether by executing three stages early in the address stage, not in the execute stage. Consequently, there is no penalty for adjusting the stack pointer.

Software I/O Saves Silicon

The third key characteristic of the IP3023's architecture is what Ubicom calls "software I/O": the ability to implement popular I/O and communications protocols in software instead of in hard-wired logic, whether that logic is on chip or off. This is another way the IP3023 saves silicon and reduces or eliminates the need for additional chips. Many SoCs devote significantly more silicon area to I/O logic than to their embedded-processor cores. In addition, software I/O makes the IP3023 more adaptable to evolving protocols, and it's possible to design systems that are remotely upgradable in the field.

Some industry-standard I/O protocols that the IP3023 can support in software are USB 1.1 (master or slave), 10/100 Ethernet, PCI, PCMCIA, Utopia, and Wi-Fi media-access controllers. It can also support garden-variety serial interfaces such as GPSI (general-purpose serial interface), SPI (serial peripheral interface), and UARTs, as well as almost any proprietary serial or parallel interfaces that developers want to implement. The chip has four media-independent interface (MII) ports for connections to external PHYs and up to 106 general-purpose I/O (GPIO) pins. Most GPIO pins serve dual duties; for instance, each MII port requires 16 GPIO pins. Only 32 pins are available for GPIO exclusively.

To make software I/O practical, of course, the processor must respond to real-time events quickly enough to avoid dropping even one bit of an incoming packet. That's what drove Ubicom to embrace hardware multithreading and eschew nondeterministic structures such as instruction and data caches. Other packet processors that have large FIFOs generate an interrupt after receiving a whole packet, but the IP3023 must trigger an interrupt every time the four-byte FIFO buffer in a serdes fills up. Because the data field of an Ethernet packet can be up to 1,500 bytes long, the IP3023 can generate 375 interrupts while processing a single packet. Clearly, this kind of interrupt-happy processing wouldn't be feasible without the low context-switching overhead of hardware multithreading.

It also explains why Ubicom provides its own proprietary control software for the IP3023. A conventional RTOS requires hundreds of kilobytes of memory and would find it difficult or impossible to keep up with the IP3023's unusually heavy demands for interrupt processing. Ubicom's proprietary ipOS requires only 50K to 100K of memory and two non-real-time threads for execution.

The IP3023 uses hard real-time threads to drive the software I/O pins. Developers can assign up to six of those threads to I/O. Interrupt-response times depend on how often a thread's instructions occur within the combined instruction stream of all threads. The IP3023 can schedule a thread's instructions to occur as frequently as 1 in 2 instructions or as infrequently as 1 in 64 instructions. The software I/O drivers written to date use 1-in-4 to 1-in-30 scheduling. The worst-case interrupt latency for a 1-in-4 thread is 16ns (4 × the 4ns cycle time at 250MHz), and the worst-case interrupt latency for a 1-in-30 thread is 120ns (30 × 4ns).

Each software I/O thread consumes less than 5% of the IP3023's available mips (250 mips at 250MHz). That makes software I/O virtually free, says Ubicom, because it requires no more mips than a general-purpose processor would waste on memory accesses and cache misses.

To demonstrate the flexibility of software I/O, Ubicom says it wrote an 802.11g driver for a PCI interface on the

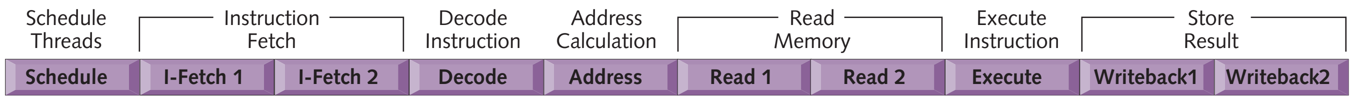

Figure 2. The IP3023's 10-stage pipeline separates the most vital operations into multiple stages. Note that the thread-scheduling stage precedes the instruction-fetch stages, so the processor always fetches instructions on the basis of which of the eight program counters (and therefore, which of the eight possible threads) currently has the highest priority.

Price & Availability

Ubicom plans to begin producing the IP3023 in volume in 2H03. It will be packaged in a 208-pin PQFP and priced at $12 each in quantities of 100,000 a month. For more information, visit *www.ubicom.com*.

older-generation IP2022 chip in a few weeks, even though the IP2022 has no specific hardware support for either 802.11g or PCI. Although the PCI interface runs at only 6MHz on the slower IP2022 chip, instead of the customary 33MHz, it is fast enough for 802.11g (54Mb/s, which requires less than 2MHz of 32-bit PCI bandwidth). More to the point, Ubicom's engineers didn't have to redesign the chip to adapt it for 802.11g—they merely had to write some software.

Some Design Tradeoffs With Software I/O

Comparing the IP3023 with a competing wireless NPU indicates that Ubicom's approach can indeed reduce the amount of silicon dedicated to peripheral I/O logic. As Figure 3 shows, Broadcom's BCM4710 is well-endowed with peripheral I/O and is 41% larger than the IP3023. Although lithography could account for all of that difference—the IP3023 is being fabricated in a 0.13-micron process, compared with 0.18 microns for the BCM4710—the Broadcom chip has several sizable I/O structures not found in the Ubicom chip. Conversely, the IP3023 devotes about two-thirds of its die to on-chip memory.

One difference between these chips is that the BCM4710's R3000 MIPS core has instruction/data caches and a translation-lookaside buffer, which the IP3023 doesn't need. Even so, the R3000 occupies less than 25% of the die. It's the I/O logic that makes a bigger difference. The BCM4710 dedicates large blocks of logic to a pair of Ethernet media-access controllers, a PCI interface, a PCMCIA interface, and a USB interface—I/O functions that the IP3023 can handle in software. Furthermore, for a typical wireless access point, the BCM4710 requires two SDRAM chips and a 4MB flash chip for the RTOS and application software, whereas the IP3023 gets by with no SDRAM and only 1MB of flash.

Of course, it's always possible to find advantages and disadvantages on each side when comparing chips as disparate as the IP3023 and the BCM4710. The Broadcom chip devotes its largest block of I/O logic—nearly one-third of the die—to a HomePNA baseband signal processor. The Ubicom chip doesn't have that logic and cannot implement a baseband processor in software, although it can use software I/O to interface with an external HomePNA PHY chip. In an application that requires HomePNA, the multichip Ubicom solution might be more expensive. (Unfortunately, Broadcom doesn't publicly disclose pricing, so we can't compare costs.)

For applications that don't need HomePNA, Broadcom makes the pin-compatible BCM4702, which is virtually identical to the BCM4710 except for the HomePNA logic. That would tip the scales in Broadcom's direction when comparing die sizes—and, presumably, the relative costs of the chips.

However, the Broadcom processors and other chips with lots of dedicated I/O logic would probably be at a cost disadvantage in an application for which the IP3023's software I/O is suitable. Any dedicated logic an application doesn't need is redundant. The corollary is that the BCM4710 and similar chips could be more suitable than the IP3023 for applications that can put all the dedicated I/O logic to good use.

Another alternative is to tailor an ASIC or SoC for a specific application, thereby avoiding the redundant I/O logic and wasted silicon in standard parts. However, the high nonrecurring engineering (NRE) costs of spinning a custom chip discourage the design of larger numbers of small chips having just enough application-specific logic. Instead, it seems more economical to design a smaller number of larger chips

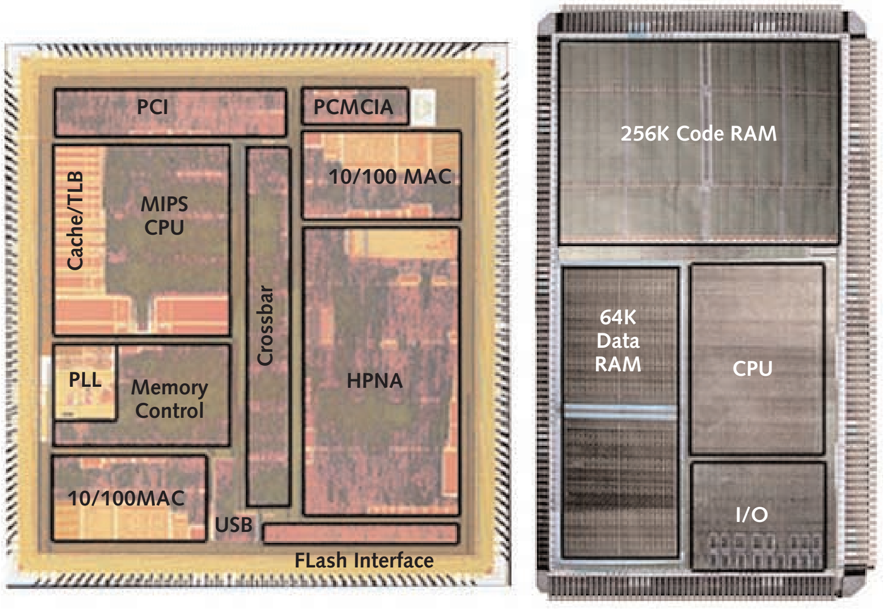

Figure 3. This die photo of the Broadcom BCM4710 (left) and die plot of the Ubicom IP3023 are reproduced at the same scale. (Ubicom is awaiting first silicon for the IP3023, so a die photo was not available.) Note that while Broadcom dedicates most of the BCM4710's logic to peripheral I/O, Ubicom devotes very little logic to I/O and quite a bit of silicon to SRAM. The on-chip memory inflates the IP3023's die size but reduces or eliminates the need for external memory.

that have enough dedicated logic to handle many similar applications. This approach inflates the manufacturing cost of each chip, however, and probably makes some of the I/O logic redundant in any single application. If Ubicom's standard-part IP3023 can handle a particular application with software I/O, it's certainly less risky than designing a custom chip. And the BOM cost—even in high volumes—should be comparable.

MASI Makes Sense

It seems bold for a small company like Ubicom to proffer a new architectural genre like MASI. Even mighty Intel is having trouble gaining traction for EPIC, its much grander architectural venture. But it's hard to argue with results. Ubicom's existing IP2022 validates the concepts of memory-to-memory instructions for packet processing, deterministic hardware multitasking, and software I/O. The IP3023 builds on that experience with a more-powerful 32-bit architecture. If MASI weren't an efficient way to design a low-cost packet processor, Ubicom couldn't sell the chip for only $12.

Other packet processors based on MIPS or ARM cores have the advantage of familiar architectures, even if, in Ubicom's eyes, they are "obsolete" RISC engines. However, the IP3023's architecture is so RISC-like and simple (only 39 instructions) that the learning curve for developers isn't too steep. It will be even smoother for existing users of Ubicom's IP2022. Although they will have to rewrite their software for the IP3023, the GnuPro C compiler has the same front end and uses the same Unity integrated development environment on both platforms. Ubicom says most IP2022 users could rewrite their assembly-language programs in C for the IP3023 and still get better performance. The company provides software I/O drivers to implement an Ethernet media-access controller, 802.11 media-access controller, USB, PCI/CardBus, PCMCIA, IDE, ISA, Utopia, GPSI, SPI, and UARTs on the IP3023. Developers can write additional drivers for similar protocols.

Core routers and gigabit switches aren't Ubicom's game. But for low-cost network-edge products, Ubicom's IP3023 drops the BOM. ◊