Reed Electronics Group

MICROPROCESSOR REPORT

www.MPRonline.com

THE INSIDER'S GUIDE TO MICROPROCESSOR HARDWARE

ITTY-BITTY 32-BITTERS

Tiny 32-Bit Processor Cores Race to Replace 8- and 16-Bit Chips

By Tom R. Halfhill {5/11/09-01}

Not everyone thinks Moore's law is a quota. Some CPU architects strive to design smaller and smaller microprocessor cores, bucking the trend toward larger processors. In the Lilliputian world of microcontrollers and deeply embedded systems, smaller is definitely better.

Microprocessor Report recently covered ARM's new Cortex-M0, the smallest implementation of the world's most popular 32-bit embedded-processor architecture. At a mere 12,000 gates, the Cortex-M0 is about one-third the size of ARM's all-time best seller, the 14-year-old ARM7TDMI. (See *MPR 3/2/09-01*, "ARM's Smallest Thumb.") However, our report overlooked two licensable processors that compete in the same flyweight class and have similar capabilities.

One challenger hails from ARM's home town of Cambridge, England. Cambridge Consultants offers a line of 16- and 32-bit processor cores in its XAP family, which now includes the hybrid 16/32-bit XAP5a. At a minimum implementation of 18,000 gates, it's 50% larger than the Cortex-M0 but promises frugality in other ways.

Across the Channel, a small French company named Cortus offers the 32-bit APS3 processor core. At 9,500 gates, it's 21% smaller than the Cortex-M0. While comparing the XAP5a and APS3 with ARM's processor, we will also take a closer look at Tensilica's Diamond Standard 106Micro. It's a 20,000-gate processor core aimed at the same market. Although these alternatives aren't as famous as ARM's products, they are worth considering when upgrading from 8- and 16-bit processors.

Of course, the main benefits of a 32-bit design are more processing power, larger memory addressing, and greater I/O bandwidth. All but one of the cores covered in this article have 32-bit flat memory addressing, so they can access up to 4GB of memory without bothersome tricks like segmenting or bank switching. And all but one have 32-bit-wide memory interfaces, sometimes with separate buses for instructions and data. The exception in both cases is the hybrid 16/32-bit XAP5a from Cambridge Consultants. It has 24-bit addressing, for a maximum of 16MB of memory, and a 16-bit memory interface.

The traditional drawbacks of 32-bit processors, compared with 8- and 16-bit cores, are their larger gate counts, higher power consumption, and inflated memory requirements for 32-bit code. However, the processors covered in this article are not traditional 32-bit processors. Some are no larger than an 8-bit 8051. Their economical gate counts reduce power consumption as well as silicon area. They conserve memory by incorporating 16-bit-long instructions in their instruction sets, sometimes approaching the code density of 16-bit processors.

All the processors covered here are licensable and synthesizable, so developers can easily integrate them with peripheral logic in programmable ASICs and SoCs. They are ideal for microcontrollers, deeply embedded systems, hard real-time applications, and low-power systems. The only smaller 32-bit programmable processors are roll-your-own custom designs. They tend to be application specific, riskier to implement, and less blessed with development tools and technical support.

Cortus APS3: King of Lilliput

Cortus is a privately owned company with eight engineers in Montpellier, France, and a sales/support office in Silicon

Valley. The founders and engineers have previous experience at Bosch, Infineon, Intel, and Synopsys. Cortus specializes in designing small, 32-bit processor cores to replace 8- and 16-bit cores in microcontrollers, which puts the company in direct competition with ARM's Cortex-M0. Cortus processors are currently used in applications as mundane as coffeemakers and as mission-critical as nuclear power plants. One customer (Certicom) just taped out a 45nm design, though most customers are using 0.18-micron processes.

Cortus created the APS architecture in 2004 and began shipping cores in 2005. The latest implementation is the 9,500-gate APS3, about 21% smaller than the base configuration of the Cortex-M0. An even smaller implementation of the APS3 is possible by omitting some optional features and synthesizing the core for minimum area. Cortus says that 9,500 gates represent a midpoint estimate, based on NAND2-equivalent gates after synthesis in Synopsys Design Compiler. (See the sidebar, "Gate Count? Depends Who's Counting.")

Fundamentally, the APS3 is a RISC architecture. Most instructions (including loads and stores) are simple enough to execute at a throughput rate of one instruction per clock cycle. However, like almost all 32-bit embedded processors, it improvises on textbook RISC principles in several ways.

Foremost is a mixed 16/32-bit instruction set. By shortening the simplest and most common instructions to 16 bits, the APS3 saves memory and partially overcomes the code-density advantage of 8- and 16-bit MCUs. Because Cortus designed the APS3 with a 16/32-bit instruction set from the start, it's a clean implementation, well suited for high-level compilers. No 16/32-bit mode switching is necessary. The Cortus GNU-based C/C++ compiler freely mixes 16- and 32-bit instructions together.

ARM Cortex processors can freely mix 16-bit and 32-bit instructions, but earlier ARM processors like the ARM7 and ARM9 families cannot. To execute 16-bit Thumb instructions, they require a mode switch. Likewise, ARM Cortex processors can handle interrupts with 16-bit instructions, but older ARM processors must revert to 32-bit mode. The Cortex-M0 instruction set consists almost entirely of 16-bit Thumb and Thumb-2 instructions, so mode switching isn't necessary, and interrupt handlers can use 16-bit code.

ARM's tradeoff for the Cortex-M0's compact instruction set is software compatibility. Although software written for the Cortex-M0 is upward compatible with other Cortex processors, existing 32-bit ARM software isn't downward compatible with the Cortex-M0. Of course, downward compatibility matters little to developers upgrading an 8- or 16-bit design.

The mixed 16/32-bit instruction set isn't the only reason the Cortus APS3 is about the same size as an 8-bit 8051. It also departs from the classic RISC model by having only 16 general-purpose registers (GPR), not the usual complement of 32 GPRs. Register R0 always equals zero, and R1 is reserved for the GNU stack pointer. That leaves 14 unallocated registers plus a dedicated return-address register. Small register files usually spell trouble for compilers, but Cortus claims the GNU C/C++ compiler is tuned so well for the APS3 that register congestion isn't an issue. In any case, the APS3 makes a common compromise in this respect. None of the other 32-bit processors covered in this article—including the Cortex-M0—has 32 GPRs.

Although the APS3's gate count may seem suspiciously small, Cortus didn't cut corners in other ways that stripped-down configurations of a processor core sometimes do. All GPRs have two read ports and one write port, so they can load two 32-bit input operands and save a 32-bit result. Also, the 9,500-gate configuration of the APS3 implements the registers in flip-flop logic, not in SRAM arrays. (All gate counts for the processors in this comparison assume that registers are implemented in flip-flops, not in SRAM.)

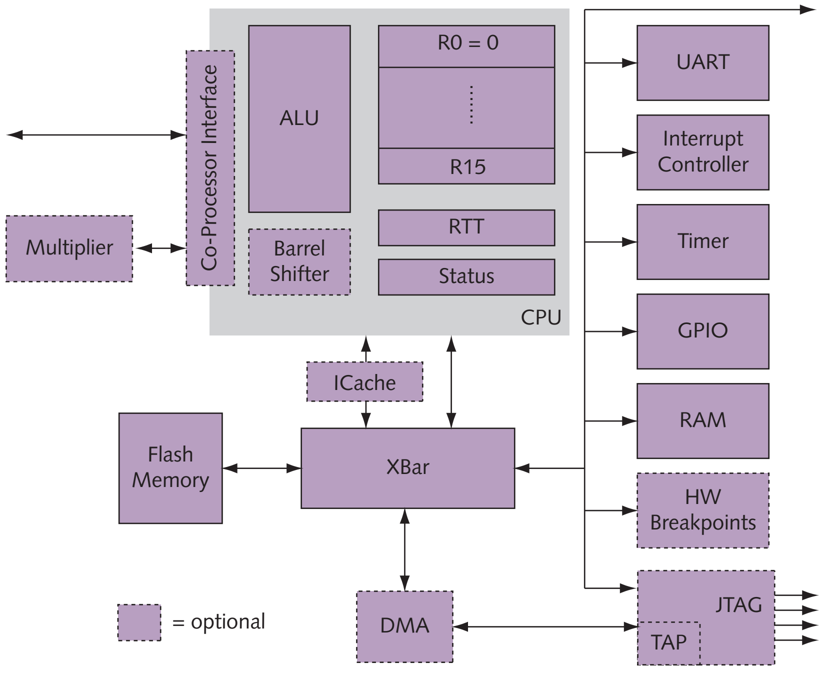

Figure 1. Cortus APS3 block diagram. During synthesis, developers can implement the memory interface as a split-bus Harvard or unified-bus von Neumann architecture and configure it for little- or big-endian memory addressing. DRAM, peripherals, and flash memory attach to the Cortus crossbar switch. Coprocessors and an instruction cache are optional. The 32-bit barrel shifter is optional but included in the 9,500-gate configuration.

Pipelined for Speed

Figure 1, a block diagram of the Cortus APS3, reveals a few more interesting features of this puny processor. By default, it has a Harvard memory architecture, with separate 32-bit I/O buses for instructions and data. Developers preferring a slimmer von Neumann memory architecture can unify the I/O buses with a crossbar switch.

The crossbar also serves as the interface to peripherals. It can host UARTs, an interrupt controller, timers, general-purpose I/O (GPIO) interfaces, and a JTAG interface—all of which Cortus offers at no extra cost. However, the crossbar interface is native to the APS3, not an industry-standard interface like ARM's AMBA High-Speed Bus (AHB or AHB-Lite) or AMBA Peripheral Bus (APB). If developers want to attach other peripherals, the crossbar does support an APB bridge.

Note that the APS3 interrupt controller is a peripheral, not an integral part of the processor core. It's not included in the 9,500-gate configuration of the APS3. Adding this controller—an indispensable feature—will nudge the APS3's gate count a little higher. Cortus says a controller supporting 10 or 20 interrupts will add only about 200 gates of logic. The controller is configurable and supports up to 256 interrupts, with 16 priorities. In practice, such a large number of interrupts is rarely needed, but they're available.

In comparison, the 12,000-gate base configuration of ARM's Cortex-M0 includes a nested vectored interrupt controller (NVIC) with one interrupt line. Developers can add as many as 32 lines, with four priorities. In addition, the Cortex-M0 base configuration includes a 32-bit AHB-Lite interface, an extra option for the APS3. On balance, then, the Cortex-M0 and APS3 are nearly the same size if similarly equipped. Cortus doesn't consider an AMBA bus to be an advantage in most small designs, because it introduces more I/O latency than the APS3's native I/O interface does.

After an interrupt, the APS3 can enter a service routine in only one or two clock cycles, except when a multicycle instruction is executing. Those (minority) instructions are interruptible but lengthen the interrupt latency to nine cycles, not counting delays caused by memory wait states. The APS3 also supports a nonmaskable critical-error interrupt.

One standout feature of the Cortus APS3 is an optional instruction cache. It's the only processor in this comparison with such an option. Usually, processor cores intended for microcontrollers shun caches, because their unpredictable behavior is intolerable in real-time systems. The APS3 allows developers to add an instruction cache as large as 2MB, with configurable cache lines (4 to 64K lines), line sizes (16 or 32 bytes), set associativity (one, two, or four ways), and burst modes. There's no option for a data cache.

Although Cortus says most APS3 customers don't implement an instruction cache, it's important for some applications. It allows programs to run directly from slow flash memory, eliminating the need for DRAM. It's particularly useful for very small, self-contained systems.

Another unusual feature of the APS3 is its instruction pipeline. For most operations, the pipeline is five stages long. Writeback requires seven stages. It's a relatively deep pipeline for such a small processor. Consequently, the APS3 can reach higher clock frequencies that most other processors in this class. In a now-pedestrian 0.13-micron process, the APS3's worst-case clock speed can exceed 300MHz. Developers have achieved 280MHz in an even older 0.18-micron process and an impressive 520MHz at 65nm. The only other processor in this group able to match those frequencies is Tensilica's Diamond Standard 106Micro, which has a similarly deep five-stage pipeline.

Customizing With Coprocessors

The Cortus APS3 has a versatile coprocessor interface that lets developers add application-specific instructions and other hardware. Cortus uses this interface to offer such tasty options as a 32-bit integer multiplier, a 16- × 16-bit multiply-accumulate (MAC) unit, a 32/64-bit FPU, and a 16-bit DSP.

Coprocessor instructions enjoy the same first-class status as native instructions. They have full access to the APS3's registers and are independently pipelined. In fact, a native instruction can bypass a coprocessor instruction and finish out of order if they have no mutual dependencies.

Cortus says the coprocessor interface is defined so abstractly that Verilog writers need no knowledge of the APS3's internal mechanisms, beyond the programmer-visible features that a software developer would know. To prove this point, Cortus says a doctoral student created a cryptographic engine for the APS3 as an educational project. Figure 2 is a high-level diagram of the APS3 coprocessor interface.

One optional coprocessor multiplies 32-bit integers in a single clock cycle. Without it, a slower math library handles those operations. Another optional coprocessor performs 32- and 64-bit IEEE floating-point arithmetic, which may be useful in some industrial-control applications. Both coprocessors are included with the APS3. Although the other processors in this comparison offer 32-bit integer multipliers as an optional or standard feature, only the Cortus APS3 includes an optional FPU.

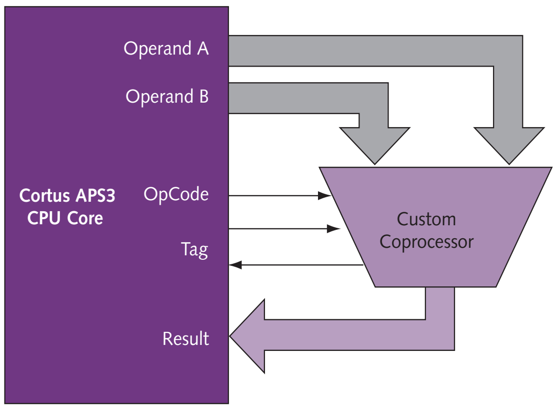

Figure 2. Cortus APS3 coprocessor interface. Important features are 32-bit datapaths (two ports for input operands and one for results) and a separate interface for fetching instruction opcodes. The APS3 supports as many as 16 coprocessors. Developers implement coprocessors in Verilog.

The most interesting coprocessor for the APS3 is a 16-bit fixed-point DSP that bears some resemblance to a Freescale Semiconductor 56000-family DSP. When fabricated at 0.18 micron, the Cortus DSP can reach 200MHz and perform all the same single-cycle operations as a 32MHz Freescale 56F8000 digital signal controller. (In other words, the Cortus DSP is six times faster.) This option differentiates the APS3 from other processor cores in this class.

The CPU/DSP tool chain is unified. Instead of using a special C compiler for the DSP, programmers use the CPU's compiler to call DSP library routines or to write in-line assembly language. (Cortus says a fast Fourier transform and other DSP algorithms can be written entirely in C.)

Further customization is possible by modifying the Verilog model of the APS3. Other CPU vendors strongly discourage, or outright forbid, their licensees from doing this. Cortus doesn't exactly encourage it, either, but it's an option. Cortus prefers to do this level of customization itself, offering design services at extra cost. On demand, Cortus stands ready to design custom instructions, coprocessors, and peripherals. As the smallest CPU vendor evaluated in this article, Cortus seems especially eager to bend over backward for customers.

Peripherals Included With CPU

Unlike most licensable-processor vendors, Cortus supplies peripheral intellectual property (IP) with the CPU core. Usually, peripherals are an additional expense. As mentioned above, the 32-bit integer multiplier and 32/64-bit FPU are included with the APS3.

Standard peripherals include a basic UART, a JTAG interface (IEEE 1149.1 standard), a breakpoint module (supporting three breakpoints), an instruction-cache controller, an interrupt controller, a GPIO controller (with up to 32 bidirectional I/O ports), a 16/32-bit timer, a master/slave Serial Peripheral Interface (SPI), an I^{2}C module, USB 1.1, USB 2.0, a Gigabit Ethernet controller, and generic memory controllers for SRAM, DRAM, and flash. All are configurable.

Together with the CPU, these peripherals are the makings of a respectable SoC, as Figure 3 shows. The only other vendor providing so much peripheral IP with a 32-bit processor core is Gaisler Research, whose SPARC-compatible LEON3 processor is too big to compete in this derby.

For software development, Cortus provides a GNU-based C/C++ compiler that runs on Linux or Windows and plugs into the Eclipse integrated development environment (IDE). As an alternative, developers can buy the Raisonance IDE

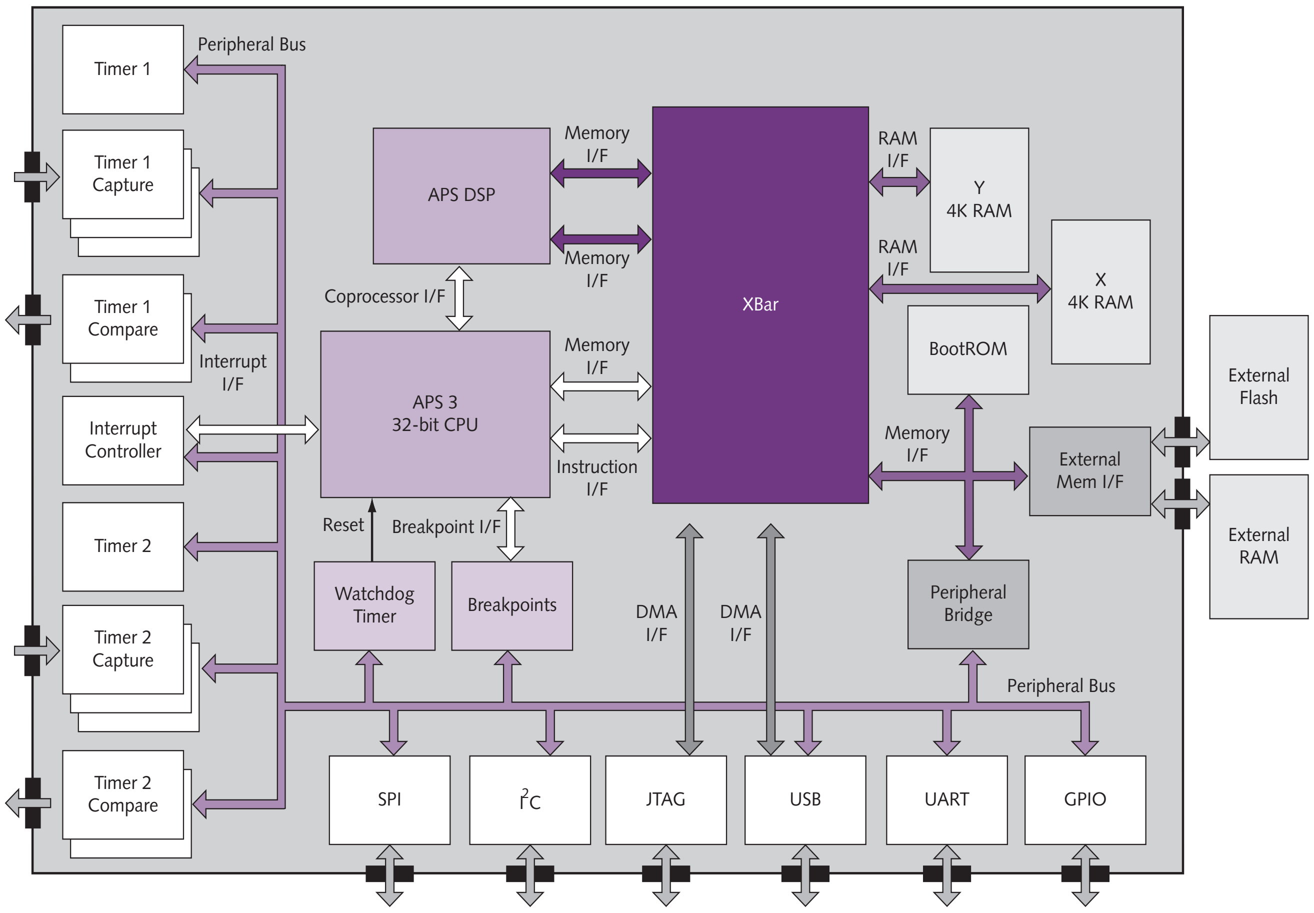

Figure 3. Cortus provides many commonly used peripheral-IP blocks with the APS3 processor core. They're a good start for designing a well-integrated SoC, and they save money. No other processor covered in this article comes with so much peripheral IP.

(RIDE7), a third-party commercial product for Windows. The Cortus APS3 instruction-set simulator is nearly cycle accurate and can model cache performance and memory wait states. Developers can also run a 100% cycle-accurate hardware/software cosimulation using System C, which is much faster than a Verilog simulation.

Better yet, the APS3 is so small that it runs handily in an affordable FPGA. For $99, Digilent sells an FPGA development board with a Xilinx Spartan3 XC-200. For larger system designs, a $150 Spartan3 XC-1000 is roomier. Cortus also has a small number of loaner 100MHz test chips fabricated at 0.35 micron.

Operating-system support—the Achilles heel of small CPU vendors—is skimpy. For now, Micrium's µC/OS-II is the only commercial option for the Cortus APS3. Depending on which modules a developer includes, this RTOS can squeeze into a few kilobytes of memory. One APS3 customer is using the open-source FreeRTOS kernel. Many of the deeply embedded systems for which the APS3 is intended don't bother with an RTOS at all, because developers prefer to run their software on bare metal.

XAP5a: Hybrid 16/32-Bit CPU

Cambridge Consultants is a British engineering firm founded in 1960, at the dawn of the IC era. After decades of offering engineering services, the company began licensing its first embedded-processor core in 1995 and its first 32-bit core in 2005. More than a billion of these cores are in silicon. Although processor-IP licensing isn't the company's main business, it recaptures some revenue lost to ARM and other vendors when providing design services to customers.

The Cambridge Consultants XAP ("zap") architecture is the foundation for a product line of 16- and 32-bit processor cores. For this article, we'll focus on the XAP5a. Frankly, we hesitated to include the XAP5a, because it's not a fully 32-bit processor. It's a 16/32-bit hybrid that takes even more liberties with classic 32-bit RISC principles than the other processors in this comparison. Despite those liberties, the XAP5a isn't the smallest core of the group. Nevertheless, we are including the XAP5a because it may represent the next evolutionary step for small 32-bit processor cores.

Some explanation is in order. The first popular RISC processors (such as MIPS and SPARC) appeared in the 1980s while the industry was advancing to 32-bit architectures. Because RISC was the new wave in microprocessor design, early RISC processors embraced 32 bits. These processors had 32-bit-wide datapaths, 32- × 32-bit GPRs, 32-bit-long instructions, 32-bit-wide I/O buses, and provisions for 32-bit memory addressing. They were high-performance microprocessors for workstations and servers.

Later, as RISC architectures lost ground to the x86 on desktops and in server closets, RISC sought refuge in the embedded market. Developers welcomed these clean, modern architectures, but the high-performance features became handicaps. In response, RISC architects began economizing to save gates in the processor core, reduce pins on the chip, and conserve memory in the system. It became common to see 32-bit RISC processors with 16-bit I/O buses, reduced memory addressing, smaller register files, and subsets of 16-bit instructions.

ARM's new Cortex-M0 is a good example of this evolution. It retains 32-bit datapaths, 32-bit GPRs, 32-bit memory addressing, and a 32-bit I/O bus. However, like all ARM processors, it has only 16 programmer-visible registers, and three of those are reserved for the program counter, stack pointer, and link register. Almost all Cortex-M0 instructions are 16 bits long—only six are 32 bits. Indeed, the Cortex-M0 instruction set is so condensed that it has only one-third as many instructions as the Cortex-M3.

Cambridge Consultants says the XAP5a carries this evolution (or *devolution*, depending on your point of view) to the next logical step. The XAP5a has 32-bit internal datapaths between its registers and function units. Otherwise, it looks more like a 16-bit processor than even the ARM Cortex-M0 does. The XAP5a has a 16-bit I/O bus, 16-bit memory addressing, a 16-bit-wide stack, and only eight 16-bit GPRs. (Unlike ARM processors, however, the XAP5a doesn't reserve GPRs for the program counter, stack pointer, and link register; it has separate registers for those purposes.) Most internal datapaths in the XAP5a are 16 or 24 bits wide. Arguably, the XAP5a is a 16-bit processor. However, it can natively perform 32-bit operations—even some wider operations—with complete transparency to high-level languages.

The XAP line does have some processors that are fully 32 bits. At Spring Processor Forum 2005, Cambridge Consultants introduced the 32-bit XAP3a, which we covered in detail shortly afterward. (See *MPR 6/13/05-01*, "XAP Takes the Stage.") The latest word from the company is that the XAP3 family will be dropped. Although another fully 32-bit XAP design is contemplated, Cambridge Consultants says most customers prefer a smaller processor core that can natively perform 32-bit operations without the overhead of a fully 32-bit microarchitecture. The 16/32-bit XAP5a is designed to meet those needs.

32-Bit Sleight-of-Hand

With a base configuration of 18,000 gates, the XAP5a is larger than the Cortex-M0 and Cortus APS3. It's slightly smaller than Tensilica's 106Micro. However, these differences seem great only in this context. Keep in mind that all these processors are less than half the size of an ARM-7TDMI, long the staple of small 32-bit microcontrollers and deeply embedded systems.

The XAP5a can natively perform 32-bit integer operations because it has instructions, registers, datapaths, and function units designed for that purpose. Instructions that manipulate 16-bit operands have 32-bit counterparts. In assembly language, programmers simply use the 32-bit version. For high-level programming, Cambridge Consultants has modified the GNU C/C++ compiler to do this automatically. When an operation manipulates a 32-bit integer variable or constant, the compiler emits the 32-bit version of the required instruction.

Register operations use similar sleight-of-hand. The XAP5a has eight 16-bit-wide GPRs. When an instruction has a 32-bit operand, it stores the value in two 16-bit registers. The pairing is largely transparent to programmers. In assembly language, the 32-bit version of an instruction merely references the source and destination registers as any 32-bit processor does. For example, the mov.r instruction copies a 16-bit value from one 16-bit register to another. Its 32-bit counterpart is mov.32.r, which copies a 32-bit value from one pair of 16-bit registers to another pair. The GNU C/C++ compiler automatically uses mov.32.r when necessary.

With only eight 16-bit GPRs to start with—a small register file—the XAP5a can allocate only four register pairs for 32-bit integers. That's very small. Other processors in this category have 16 GPRs for 32-bit values. A mitigating factor is that the XAP5a has dedicated 24-bit registers for its program counter, stack pointers, vector pointer, and breakpoints, whereas competing processors conscript GPRs for some of those purposes.

Even so, register congestion would seem to be a hazard for the XAP5a. In particular, compilers aren't as adept at managing registers as experienced assembly-language programmers are. But the XAP5a's designers insist that register congestion isn't a problem. They agree with Cortus that open-source developers have spent years tuning the GNU C/C++ compiler for efficient register management on a variety of CPU architectures.

Small Memory Trumps Small Logic

Figure 4 is a block diagram of the XAP5a. Notice the 32-bit datapaths connecting the registers and function units. Many XAP5a instructions use the common form of op r0,r1,r2, in which op is the instruction mnemonic, r0 is the destination register, and r1 and r2 are input operands. The 32-bit datapaths allow function units to shuttle 32-bit operands and results to and from the registers in one clock cycle. However, the 16-bit I/O bus requires two clock cycles to load or save a 32-bit value in memory.

In keeping with its frugal ways, the XAP5a can save 1-, 8-, 16-, or 32-bit values anywhere in its byte-addressable 16-bit memory. Therefore, 16- and 32-bit values can be unaligned on byte boundaries (albeit with a one-cycle access penalty). Software libraries support 64-bit long integers and floating-point math. Versatile block-move instructions can copy data from one byte address to another.

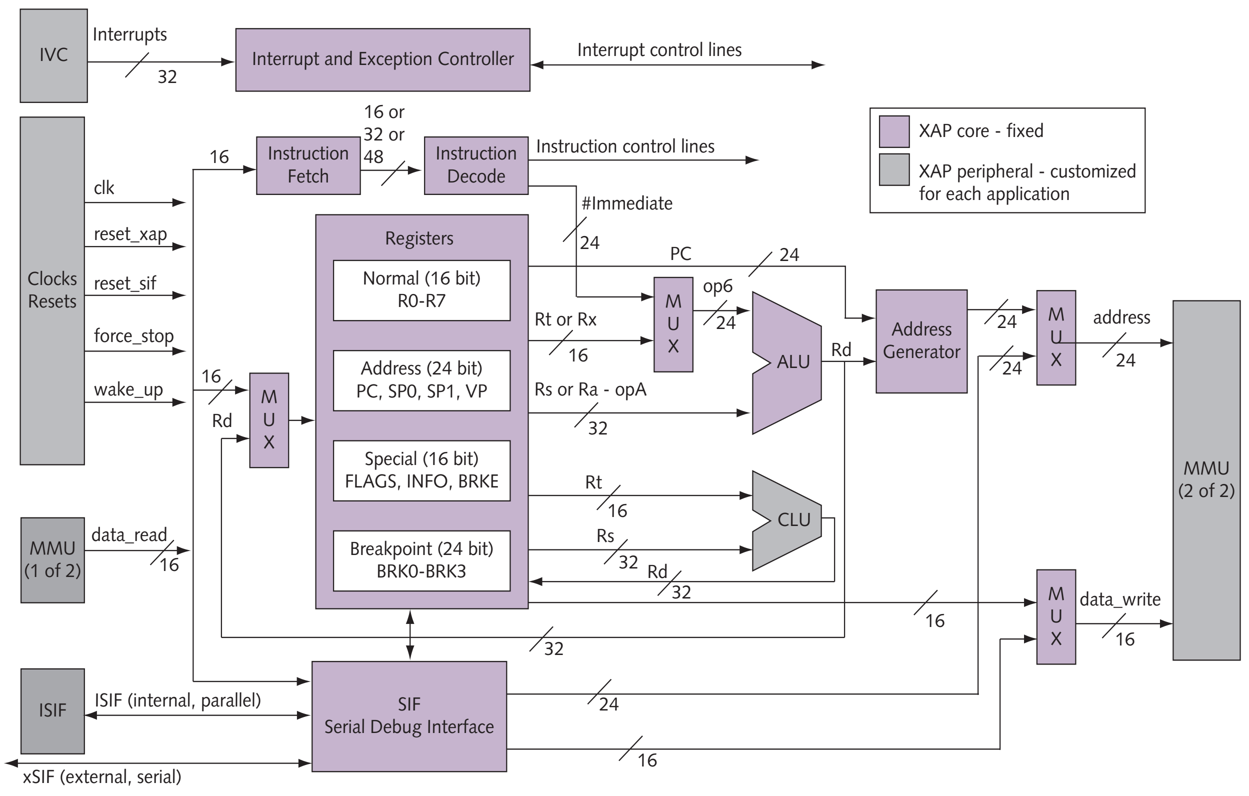

Figure 4. Cambridge Consultants XAP5a block diagram. Gray blocks are user-customizable components. The custom logic unit (CLU) and internal serial interface (iSIF) are optional. This hybrid 16/32-bit processor is basically a 16-bit architecture, but it has just enough 32-bit features to natively perform 32-bit integer operations with little or no bother for programmers. However, the 16-bit I/O bus limits throughput when compared with fully 32-bit processors. Although 24-bit pointers and address registers limit the XAP5a to 16MB of memory, it should be sufficient for deeply embedded applications.

Note that some “32-bit” operations actually are mixed 16/32-bit operations, producing a 16-bit result. For instance, the hardware divider applies a 16-bit divisor to a 32-bit dividend and yields a 16-bit quotient, possibly resulting in loss of precision when compared with a full 32-bit division. (It’s relatively fast—unsigned divides require 18 clock cycles, and signed divides require 20 cycles.)

When the XAP5a stores an address or pointer in memory or a GPR, the 24-bit value automatically sign-extends to 32 bits. Cambridge Consultants says it will be easy to design an enhanced version of the processor with full 32-bit memory addressing, should demand warrant it. Currently, most XAP5a customers use only 100KB to 2MB of memory.

Memory conservation is the prime philosophy of the XAP5a. Cambridge Consultants says that in a compact embedded system, small memory is more important than small logic. Developers should compare the memory requirements of competing processors instead of focusing on gate counts. Memory, not logic, tends to dominate the die of an ASIC or SoC. A processor that uses 16-bit memory and is designed around 16-bit data needs less memory than a processor oriented around 32-bit memory and 32-bit data. As an example, Cambridge Consultants cites a XAP5a-based medical chip designed for human implantation. RAM occupies 57.6% of the die, ROM occupies 35.6%, and the processor occupies only 6.8%.

In addition to conserving silicon, a small design can save energy if it runs slowly and executes programs directly from nonvolatile flash memory that’s powered down between events. Processors running at high clock speeds must copy the program and data from nonvolatile storage to RAM before responding. Cambridge Consultants asserts that fast 32-bit processors aren’t the best solution for small embedded systems, even if those processors are smaller than the XAP5a. Of course, this argument assumes the application doesn’t require a preponderance of 32-bit data processing with high throughput.

Customizable Instruction Set

A surprising feature of the XAP5a is that developers can create their own application-specific instructions. The XAP5a has a special function unit for this purpose: the custom logic unit (CLU). It’s visible as an optional component in the XAP5a block diagram, near the ALU (Figure 4).

Essentially, the CLU is a tightly coupled coprocessor. Thanks to tight integration, custom instructions can access the XAP5a’s GPRs as readily as ALU instructions can. Usually, coprocessors have their own registers and exchange data with the CPU via load and store operations. (As described above, the Cortus APS3 is another exception to this rule—its optional coprocessors enjoy equal access to GPRs, too.)

Cambridge Consultants has predefined 15 template instructions that developers can customize. All are 32 bits long and have an undefined 13-bit immediate value, which developers can use for any purpose. All are recognized by the assembler and GNU C/C++ compiler. One of these template instructions operates on an immediate value alone. The others can access one, two, or three registers as well as an immediate value.

To define custom instructions, developers must get their hands dirty in Verilog. There’s no high-level design-automation tool like those available for the customizable processors from ARC International, MIPS Technologies, and Tensilica. ARC and Tensilica have the best tools in this field. However, Tensilica’s entry in this flyweight class of processors—the Diamond Standard 106Micro—is a preconfigured core that rules out additional instruction-set customization. The ARC 605 is configurable, but at 30,000 gates for the base configuration, it would be the heaviest core if included in this group.

The option to create custom instructions partly compensates for the XAP5a’s 16-bit deficiencies. Application-specific instructions can accelerate critical tasks much more efficiently than simply cranking up the processor’s clock speed. Of course, the additional logic also inflates the core. Overindulgence in this feature could push the XAP5a out of consideration, if core size matters.

XAP5a Has Privileged Execution

Two more important features are the XAP5a’s privileged execution modes and user-configurable memory-management unit (MMU). These features work together to give developers control over software execution in mission-critical and security-conscious applications. Privileged execution is built into the core and supports a supervisor mode, user mode, and trusted mode.

The MMU isn’t a full-fledged memory manager capable of addressing virtual memory. However, it can trigger exceptions if a program tries to access regions of memory that are off limits. The MMU can throw four different exceptions, including two that signal when a user program attempts to access instructions or data outside its restricted memory region. Exceptions trigger a context switch that calls a handler routine. This feature allows developers to supervise user-level processes and protect multiple user tasks from each other.

One competing processor, Tensilica’s 106Micro, has a similar memory-protection unit but no privileged-execution modes. The 106Micro can reserve multiple regions of memory for different programs. Unauthorized attempts to access a protected region will trigger an exception.

Revealing its 16-bit heritage again, the XAP5a has the shortest pipeline of all the processors in this group. With only two stages—barely a pipeline at all—the XAP5a strains to reach a maximum worst-case clock frequency of 175MHz when fabricated in a generic 90nm CMOS process. Even the three-stage ARM Cortex-M0 can hit 270MHz or so at that node. The Cortus APS3 and Tensilica 106Micro are the speed demons in this class. Thanks to their five- or seven-stage pipelines, they can hit 300–400MHz at 90nm.

The XAP5a's lower headroom for clock frequency is worse than it seems. With a 16-bit I/O bus—every other processor in this group has a 32-bit bus—the XAP5a requires two clock cycles to load or save a 32-bit value. In comparison with the other processors, the XAP5a's narrower bus halves the effective bus frequency when loading or saving 32-bit data. Also, many of the XAP5a's 206 instructions are 32 bits or 48 bits long. (The 32-bit instructions contain 16-bit immediate values, and the 48-bit instructions contain 24- or 32-bit immediates.) The bus must labor for two or three cycles just to load one of these instructions. In contrast, the fully 32-bit Cortex-M0 has a largely 16-bit instruction set—only six of its 58 instructions are 32 bits long.

No worries, says Cambridge Consultants. The XAP5a is designed for embedded applications that mainly use 16-bit data, plus a few 32-bit operations. As a controller, the XAP5a will stay busy crunching data from an analog-to-digital converter (ADC) whose resolution is typically 16 bits or less. Example applications include Bluetooth wireless radios, Zigbee wireless controllers, smart utility meters, touch screens, heart pacemakers, hearing aids, and similar embedded systems. An 8- or 16-bit processor may not have enough memory or throughput for these systems, especially if it must perform 32-bit operations often enough to tax a software library. Hence the 16/32-bit XAP5a.

It's a sound argument. Nevertheless, the 16-bit aspects of the XAP5a make it the slowest processor in this group. As measured by the dubious but ubiquitous Dhrystone 2.1 benchmark, the XAP5a executes 0.68Dmips per megahertz. The Cortus APS3 manages 0.88Dmips per megahertz and can reach much higher clock speeds in the same fabrication process. ARM rates the Cortex-M0 at 0.9Dmips per megahertz, and Tensilica pegs the 106Micro at 1.22Dmips per megahertz. Both processors have much greater clock-frequency headroom than the XAP5a.

In power efficiency, the XAP5a fares a little better. At 175MHz, it achieves 27.2Dmips per milliwatt. In comparison, the 106Micro delivers 23–28Dmips/mW and the APS3 delivers 36.6Dmips/mW. The Cortex-M0 handily beats them all with 60Dmips/mW. Overall, the most attractive features of the XAP5a are customizable instructions, privileged execution, and compact 16-bit memory.

Tensilica's Mighty Mouse

Although Tensilica is famous for its user-customizable Xtensa processor cores, its entry into this class is the preconfigured Diamond Standard 106Micro. It's a factory-configured derivative of the Xtensa 7 core, ready for deeply embedded applications out of the box. Tensilica recently added the 106Micro to the Diamond Standard series introduced three years ago. (See *MPR 3/29/06-01*, "Tensilica's Preconfigured Cores," and *MPR 12/4/06-02*, "Tensilica Upgrades Xtensa Cores.")

Several features set the 106Micro apart from the other processors considered here. First, it has an unusually memory-efficient instruction set, because the standard instruction length is 24 bits instead of 32 bits. It also has the usual subset of 16-bit instructions. Datapaths and registers are 32 bits wide, so the 106Micro is a true 32-bit processor, but its 24-bit instructions require 25% less memory than conventional RISC instructions. Even so, the ARM Cortex-M0 is hard to beat in this category, because its instruction set consists almost entirely of 16-bit Thumb and Thumb-2 instructions.

The second standout feature of the 106Micro is its memory-protection unit. It's not a full-fledged MMU, so it doesn't support virtual memory. Nor does it support privileged execution, as the XAP5a does. However, the 106Micro does allow developers to rope off regions of memory to provide some protection for vital tasks. In addition, the 106Micro has a Harvard memory interface, which segregates instructions and data.

Third, the 106Micro's 20,000-gate core includes features that are optional or external components for most other cores. It has a 32-bit multiplier, a 16-bit timer, an interrupt controller, and on-chip debug logic with JTAG and trace ports. Adding these features to some of the other cores would inflate their nominally lower gate counts.

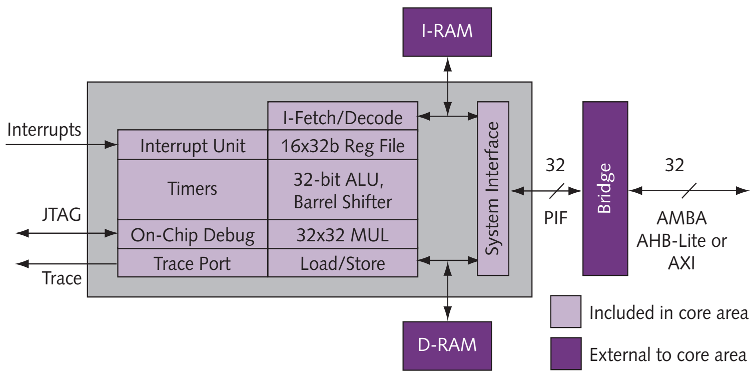

Figure 5. Tensilica Diamond Standard 106Micro block diagram. The 106Micro is a preconfigured version of the Xtensa 7 microarchitecture, introduced in 2006. Its 16-entry register file and 16/24-bit instruction set help keep this 32-bit core small. Yet it retains a 32- × 32-bit multiplier, integrated debug logic, and a Harvard memory interface. Tightly coupled instruction and data memories can be as large as 128KB each. An external bus bridge connects Tensilica's Processor Interface (PIF) to an industry-standard AMBA or AXI interface. Tensilica offers free IP for an AHB-Lite bus.

Fewer Interrupts, But Enough

Tensilica cut some corners to keep the 106Micro small. The 32-bit multiplier is relatively slow, and the interrupt controller isn't as good as those in competing processors. It supports only 15 interrupt lines and two priority levels—the other processors have at least twice as many. For most applications, that's not a handicap. Figure 5 is a block diagram of the 106Micro.

Thanks to its relatively deep five-stage pipeline, the 106Micro is probably the swiftest processor in this group. It can reach a maximum worst-case clock frequency of 400MHz in a speed-optimized 90nm-G process. The next-fastest processor is the Cortus APS3, which reaches 300MHz in a slower 130nm-G process. (Direct comparisons are hampered, because Cortus customers have taped out designs at 130nm, 65nm, and 45nm, but not at 90nm.)

The 106Micro delivers the best throughput, too, as measured by Dhrystone: 1.22Dmips per megahertz. Therefore, its power/performance efficiency is relatively high: 23–28Dmips/mW, depending on the synthesis optimization and target fabrication process.

A final consideration is that Tensilica provides a bus bridge that adapts its proprietary Xtensa Processor Interface (PIF) to the de facto standard AMBA AHB-Lite or AXI interfaces. It's easy to drop the 106Micro into an SoC design for integration with widely available peripheral cores. However, Tensilica's AHB-Lite bridge isn't counted in the 106Micro's 20,000 gates. ARM does count a 32-bit AHB-Lite interface in the 12,000-gate base configuration of the Cortex-M0.

Size Matters, But Not Much

Table 1 summarizes the world's smallest 32-bit licensable processor cores. Additional candidates might be the ARC 605 (30,000 gates) and MIPS M4K (37,500 gates). Their gate counts are somewhat higher while still undercutting the ARM7TDMI. However, ARC and MIPS aren't pursuing the 8/16-bit replacement market as aggressively as ARM, Cortus, Cambridge Consultants, and Tensilica. We included

Feature	ARM Cortex-M0	Cambridge XAP5a	Cortus APS3	Tensilica 106Micro
CPU Architecture	ARMv6-M	XAP5	Cortus APS	Xtensa 7
Architecture Width	32 bits	16 / 32 bits	32 bits	32 bits
Configurable ISA	—	Yes	Yes (via coprocessors)	Preconfigured
Instruction Lengths	16, 32 bits	16, 32, 48 bits	16, 32 bits	16, 24 bits
Instruction Pipeline	3 stages	2 stages	5–7 stages	5 stages
General-Purpose Registers	16 x 32 bits*	8 x 16 bits (4 x 32 bits)	16 x 32 bits	16 x 32 bits
L1 Cache (I / D)	—	—	I-cache 0KB–2MB	—
Instr. RAM (Local)	—	—	—	0–128K
Data RAM (Local)	—	—	—	0–128K
32-Bit Multiplier	2 options	Yes	Optional	Yes
32-Bit Divider	—	32 → 16 bits	—	—
Memory Space	4GB	16MB	4GB	4GB
Memory Architecture	von Neumann	von Neumann	Harvard or von Neumann	Harvard
Memory Management	—	Memory-protection unit	—	Memory-protection unit
System Interface	1 x AHB-Lite 32 bits	Native 1 x 16 bits	Native 1 x 32 or 2 x 32 bits† + 32-bit coprocessor I/F	Xtensa 32-bit PIF AHB-Lite / AXI bridge
External Interrupts	1–32 4 priorities	32 4 or 16 priorities	Up to 256 16 priorities	15 2 priorities
Nonmaskable Int.	Yes	Yes	Yes	Yes
Debug Module	Yes	Yes	Yes	Yes
Core Freq (Max) (IC Process)	~270MHz (90nm G)	175MHz (90nm G)	>300MHz (130nm G)	400MHz (90nm G)
Core Size (Base)	12k gates	18k gates	9.5k gates	20k gates
Dhrystone 2.1	0.9Dmips / MHz	0.68Dmips / MHz	0.88Dmips / MHz	1.22Dmips / MHz
Power (Typical) (IC Process)	0.015mW / MHz (90nm G)	0.025mW / MHz (90nm G)	0.024mW / MHz (130nm G)	0.044–0.054mW / MHz (90nm G)
Power Efficiency	60Dmips / mW	27.2Dmips / mW	36.6Dmips / mW	23–28Dmips / mW

Table 1. Feature comparison of the ARM Cortex-M0, Cambridge Consultants XAP5a, Cortus APS3, and Tensilica Diamond Standard 106Micro. As usual with comparison tables of this sort, be wary of the power-consumption and performance specifications. *MPR* obtained the data from vendors, whose engineers measured the performance of their products under varying conditions. Gate counts are also variable. Real-world results depend on such factors as the core configuration, synthesis optimization, cell library, layout, and fabrication process. *Three of ARM's GPRs are reserved for the program counter, stack pointer, and link register. †AHB-Lite and APB are optional for the Cortus APS3.

Gate Count? Depends Who's Counting

In the Middle Ages, philosophers reputedly debated the number of angels that could dance on the head of a pin. Today, engineers—and, more often, marketing managers—debate the number of logic gates in their processor cores. Both debates have no end, unless everyone agrees on the same terms.

In days of old, when technicians carved logic gates into sheets of rubylith with X-Acto knives, it was actually possible to count gates with some certainty. Today, the design process is much more abstract and automated. CPU architects create a model of the processor in a hardware-abstraction language, such as Verilog, then use a logic synthesizer such as Synopsys Design Compiler to generate a lower-level rendering. Later, highly automated place-and-route tools generate the layout. True gate-level design is reserved for a few critical paths or abandoned altogether.

Gate counts are a rough way of expressing the size of a microprocessor core independently of the fabrication process used to manufacture the chip. Silicon area, not gates, is what really matters. But the silicon area required for a processor core depends on numerous factors, not least the chip-fabrication process. To ease comparisons among different fabrication processes, CPU vendors often state the core's "gate count." But this gate count is a further abstraction that also depends on several factors.

Process Technology	NAND2 Gates	Silicon Area	Clock Frequency
AMS C35 0.35 Micron (Typical)	10,182	1.11mm^2	100MHz
TSMC 18G 0.18 Micron (Typical)	9,939	0.17mm^2	33MHz
TSMC 18G 0.18 Micron (Slow)	14,635	0.24mm^2	200MHz
TSMC 13G 0.13 Micron (Typical)	7,051	0.08mm^2	33MHz
TSMC 13G 0.13 Micron (Typical)	12,303	0.15mm^2	150MHz
TSMC 65GP-LV 65nm (Slow)	8,276	0.04mm^2	520MHz
Xilinx FPGA Virtex-5	1,048 slices	n/a	110MHz

Table 2. Estimated gate counts when implementing the Cortus APS3 processor core in different fabrication processes. These are "NAND2-equivalent" estimates and assume a midsize NAND2 gate, not the smallest possible NAND2 gate. Note that even when the fabrication process and gate type are constant—as in the TSMC 13G example—the estimated gate counts can vary widely, depending on whether synthesis was optimized for speed or area.

First: what kind of gate is the vendor counting? Synthesis compilers have lots of flexibility when implementing logic. They use all types of gates available in the cell library for the target fabrication process. Counting those gates would yield an accurate number, but only for processors synthesized under the same conditions.

To derive a process-independent gate count, CPU vendors typically divide the silicon area of the processor core by the area required for a single "NAND2-equivalent" gate. Theoretically, the compiler could synthesize any design using only NAND2 gates and inverters, though it would be very inefficient. The convention of counting NAND2-equivalent gates is purely an abstraction to make comparisons easier. But even this abstraction is subject to debate, because there are different subtypes of NAND2 gates that vary in size.

For example, Cortus says one configuration of its APS3 processor core requires about 9,500 gates in a 0.35-micron fabrication process. Another configuration of the APS3 in this process requires about 10,200 gates. For both estimates, Cortus assumes a *mid-size* NAND2 gate. In contrast, Cambridge Consultants says its estimate of 18,000 gates for the XAP5a processor assumes the *smallest* type of NAND2 gate (NAND2x1 in TSMC nomenclature), which yields the highest theoretical gate count. If Cortus assumed an equally small NAND2 gate for its estimates, the gate counts for the APS3 would be higher.

Even when assuming the same type of NAND2 gates, the count can vary widely in different fabrication processes and when the processor is synthesized for different performance targets. The accompanying table shows the variability of gate counts for the APS3. The same principles apply to all synthesizable processors.

In this article, *Microprocessor Report* quotes gate counts from the CPU vendors. Your mileage may vary. Vendors can fudge these numbers by tinkering with the configurations of their processor cores, synthesizing the logic for different performance targets, and basing the count on different-size NAND2 gates. However, overindulging in these tricks would risk spoiling a customer relationship forged on an unrealistic gate count. Our advice is to treat the vendor's gate count as a first approximation. For more data, consult the vendor. All the processors covered in this article are so small and so close in size that gate counts shouldn't be the deciding factor.

the ARC 605 and MIPS M4K in our recent coverage of the ARM Cortex-M0. (See *MPR 3/2/09-01*, "ARM's Smallest Thumb.")

Ironically, core size may be the *least* important factor when choosing among these processors. All are so tiny that their size differences tend to sift out in the mix. All are significantly smaller than an ARM7TDMI, heretofore the default choice when upgrading from 8 or 16 bits. When equipped with approximately equivalent features, the ARM Cortex-M0 and Cortus APS3 look like the smallest of the small, followed closely by Tensilica's 106Micro and Cambridge Consultants XAP5a. All are microscopic, so features other than gate counts loom larger.

If privileged execution or memory protection matters, the XAP5a and 106Micro emerge from the pack. For raw speed, the deeper-pipelined APS3 and 106Micro deliver the goods. For cost, we deem Cambridge Consultants, Cortus, and Tensilica to be more open to negotiation on licensing fees and royalties than industry leader ARM—though tough times soften the heart of many a sales rep. Cortus is notable for including valuable peripheral IP with its CPU license. ARM's Cortex-M0 base configuration includes an interrupt controller (an included but external component for the Cortus APS3) and an AHB-Lite bus (not included with the APS3). Tensilica offers a free AHB-Lite bus for the 106Micro.

For the best selection of development tools and operating systems, ARM is the obvious choice. But even though the Cortex-M0 is well supported, its mostly 16-bit instruction set isn't compatible with the larger catalog of tools and software offered for other 32-bit ARM processors. The 106Micro runs closely behind ARM in this category, because it's software-compatible with Tensilica's Xtensa architecture and existing tool chain. It will run the same operating systems as other Xtensa processors. Cambridge Consultants and Cortus have somewhat less support, especially from third parties, though Raisonance RIDE7 is an option for the APS3.

Power consumption, usually a vital factor in small embedded systems, probably won't drive the decision in this case. All these cores are tiny and approximately equal in size. All will probably lose more energy to static leakage than to dynamic switching. All have power specifications measured by their vendors under varying conditions and unverified by an independent party. As soft cores, all are heavily dependent on synthesis optimizations and fabrication processes. With one important exception, there's little to distinguish these cores with regard to power.

That exception is a feature of the Cortex-M0. If developers implement this core in an ultralow-leakage 0.18-micron process with an ARM power-management kit and cell library, the Cortex-M0 can enter a special deep-sleep mode. The processor can stop its clock and shut down the main power rail, drawing less than 50 nanoamps while asleep. Yet wakeup is almost instantaneous, because a separate low-power domain preserves the processor's registers. (We described this feature in detail in our report on the Cortex-M0.) Cortus says an APS3 customer has implemented a similar stop-clock and power-down sleep mode, but it's a custom design.

So those are the tradeoffs. When an 8- or 16-bit engine runs out of gas, any of these 32-bit subcompacts will carry a project across the finish line. ◇

Price & Availability

All the processor cores described in this article are available for licensing now. Vendors don't publicize their up-front licensing fees and royalties—which, in any case, are probably more negotiable now than ever. For more information, visit these vendor websites:
www.arm.com/products/CPUs/ARM-Cortex-M0.html
www.cambridgeconsultants.com/xap
www.cortus.com/index.php?page=products
www.tensilica.com/products/diamond/di_106micro.