SELECTING A HIGH-PERFORMANCE EMBEDDED PROCESSOR

Michael Slater
Microprocessor Report
Sebastopol, CA

Michael Slater is editor and publisher of Microprocessor Report, a biweekly newsletter on microprocessor technology. He also organizes the annual Microprocessor Forum conference, presents seminars on microprocessor technology, and consults on strategic microprocessor-related issues. He is the author of Microprocessor-Based Design (Prentice-Hall, 1989) and a regular contributor to EE Times. He has also been an R&D engineer at Hewlett-Packard and an independent engineering consultant.

Designers of high-performance embedded control systems are faced with a plethora of choices when selecting a microprocessor. In this paper, we discuss the key issues to consider when selecting a processor and review the major families of high-end microprocessors that are targeted at embedded control applications.

Selection Considerations

Selecting the best microprocessor for an application is a demanding task. First, you should decide what the most important criteria are for your application. If you’re selecting a processor for a single product and aren’t concerned with future generations, your task is simplified because you don’t need to evaluate the future potential of each microprocessor family. Generally, however, products are designed with at least the hope that there will be future versions, and you should consider how much performance “headroom” the application is likely to need as well as the chip vendor’s future product plans. Another factor to consider is the range of products that is planned. For example, a company designing a raster image processor for a phototypesetter needn’t be too concerned with the existence of low-cost members of the selected processor family, while a laser printer maker that wants to build low-end, inexpensive printers as well as high-end, high-speed printers needs a processor family with a broad price/performance range.

The first factor to consider is the maximum price that can be tolerated by the application. There’s no point in evaluating $100 processors if the total electronics budget is $50, and this is the quickest way to limit the range of options. At under $20, choices are quite limited for 32-bit processors; at $100 there are many more options. What really matters, of course, is total system cost, not processor chip cost, and this makes accurate pricing evaluations very difficult. For a full evaluation, you need to sketch out a complete system design using each processor, determine the memory speed required, and add up the cost of the processor and the support and memory chips.

The total system cost can vary considerably depending on the on-chip peripherals and
SELECTING A HIGH-PERFORMANCE EMBEDDED PROCESSOR

Michael Slater
Microprocessor Report
Sebastopol, CA

Designers of high-performance embedded control systems are faced with a plethora of choices when selecting a microprocessor. In this paper, we discuss the key issues to consider when selecting a processor and review the major families of high-end microprocessors that are targeted at embedded control applications.

Selection Considerations

Selecting the best microprocessor for an application is a demanding task. First, you should decide what the most important criteria are for your application. If you’re selecting a processor for a single product and aren’t concerned with future generations, your task is simplified because you don’t need to evaluate the future potential of each microprocessor family. Generally, however, products are designed with at least the hope that there will be future versions, and you should consider how much performance “headroom” the application is likely to need as well as the chip vendor’s future product plans. Another factor to consider is the range of products that is planned. For example, a company designing a raster image processor for a phototypesetter needn’t be too concerned with the existence of low-cost members of the selected processor family, while a laser printer maker that wants to build low-end, inexpensive printers as well as high-end, high-speed printers needs a processor family with a broad price/performance range.

The first factor to consider is the maximum price that can be tolerated by the application. There’s no point in evaluating $100 processors if the total electronics budget is $50, and this is the quickest way to limit the range of options. At under $20, choices are quite limited for 32-bit processors; at $100 there are many more options. What really matters, of course, is total system cost, not processor chip cost, and this makes accurate pricing evaluations very difficult. For a full evaluation, you need to sketch out a complete system design using each processor, determine the memory speed required, and add up the cost of the processor and the support and memory chips.

The total system cost can vary considerably depending on the on-chip peripherals and...
external memory requirements. Some high-performance microprocessors are nothing more than a basic processor core, while others include programmable chip select logic, programmable bus timing, DMA and interrupt controllers, and more. These features may or may not be valuable for your application.

After ruling out processors that are too expensive, the minimum acceptable performance is the next factor to consider. For many applications, there is a critical performance level that must be achieved. An engine controller, for example, must be able to keep up with the pace of input signals at the highest engine speed and execute control algorithms quickly enough to fire the spark plugs at the required rate. Applications that involve signal processing, such as a fax modem, need to be able to implement the required algorithms quickly enough to handle the input sample rate. In some applications, such as a laser printer controller, the speed required is simply a matter of providing competitive performance. In some cases, the processor speed may be determined by what it takes to keep the user from getting frustrated. For the most demanding applications, all processors are slower than desired, and the selection process is simply a matter of finding the absolute fastest processor for the application.

Microprocessor performance evaluation is an enormous and complex topic, and covering the various benchmarks is beyond the scope of this paper. It is worth emphasizing, however, the key maxim of benchmarking: base your evaluation on software that is similar to your application. If your application requires floating-point calculations, it will, of course, perform much better on a processor with an on-chip or external floating-point unit. If it performs a lot of multiplications, it will do better on a chip with a fast multiply instruction than on one with only multiply-step. You can mislead yourself greatly if you assume that standard benchmarks are a good indicator of performance on your application.

Once you have characterized your application so you know what the performance bottlenecks are, the best way to compare the performance of various processors is to write a sample program that is representative of your application and measure its performance on a demonstration system. If you have a program in a high-level language and are a large prospective customer, most processor vendors will compile your program, run it on their hardware, and give you performance results. One factor to be aware of, however, is that the hardware used to test your software might not be typical of what you would build; for example, it may use a faster, more expensive memory system. If you need to squeeze every bit of available performance and plan to write critical routines in assembly language, comparing processors is more difficult. Hand coding your algorithm for a variety of processors is far more work that compiling the same C program for each processor, especially if the program is large or complex.

A host of support issues must also be considered. The quality of the high-level language compilers may vary considerably, and this is a difficult factor to evaluate. It is reassuring, at least, if compilers are available from several sources. The availability of in-circuit emulators, and the price and capability of the emulators, varies dramatically from one processor to another. Depending on your debugging needs, having an emulator might be important, or you might be able to use a software debugger, possibly coupled with a logic analyzer for real-time
tracing. Most processors are supported with disassemblers by at least one logic analyzer maker, but if you already have an investment in one type of logic analyzer, you may want to verify that your vendor will support the chosen processor.

Microprocessors with on-chip caches reduce the need for fast external memory, but they present a number of other problems. If your application has a critical timing loop, you must be able to ensure that the loop will be stored in the cache when it is needed. The cache locking facilities vary considerably from one processor to another. Some only allow the entire cache to be locked or not; others allow half the cache to be locked; and the most flexible, such as Fujitsu’s SPARClite, allow each cache line to be individually locked.

Caches also present debugging problems. If you simply hook a logic analyzer to the bus of a processor with on-chip cache, what you will trace is cache refill activity and memory write cycles—not instruction execution activity. Most microprocessors designed for embedded applications provide some mechanism for tracing instruction execution. In the worst case, the on-chip cache must be disabled, which makes it impossible to perform a true real-time trace. Many processors provide some mechanism for passing execution activity to a logic analyzer or emulator trace buffer. These mechanisms are generally proprietary and are supported only by emulators or logic analyzers that are developed in partnership with the microprocessor maker.

**Embedded Processor Types**

Nearly every microprocessor architecture now has some implementations that are promoted for embedded control. In almost every instance, the microprocessor vendor initially planned to sell its chips for general-purpose computer systems, but facing an uphill battle and limited opportunities in this area, refocused on embedded control. Microprocessor selection for general-purpose computers is driven by the availability of a large body of third-party software, and this makes it difficult for new architectures to gain a foothold. In embedded applications, however, a compiler, an assembler, a debugger, and perhaps a real-time operating system are generally all that is needed, and the system designer provides all the application software.

In a half-hearted attempt to penetrate the embedded market, some vendors of workstation microprocessors have simply dropped a few features from their workstation products and called them embedded processors. Examples of this are the embedded SPARC processors from LSI Logic and Cypress. These chips lack the on-board caches, peripherals, and other features required to be effective in embedded applications, and rarely have any in-circuit emulator support. Another example is Intel’s 376, a version of the 386SX in which real mode has been disabled (preventing it from running DOS) and paging has been disabled (preventing it from running Unix). This makes it possible for Intel to price the chip more aggressively for the embedded market without affecting its margins for the 386SX in general-purpose systems.

It is important to keep in mind the distinction between architecture and implementation. The fact that an architecture was originally designed for general-purpose computers does not
mean that it cannot be very effective in embedded applications, given a suitable implementa-
tion. For example, Fujitsu’s SPARC lite microprocessor implements the SPARC architecture
but includes many features designed especially for embedded control applications. Other
examples include the embedded MIPS processors from LSI Logic and IDT.

The RISC vs. CISC battle continues to rage in the embedded processor arena. There is no
doubt that RISC processors provide the highest performance for the broadest range of appli-
cations, but this does not mean that CISC processors are dead. Motorola and National have
each taken their CISC processor architectures and embellished them with auxiliary on-chip
processing units, such as the DSP support in several of National’s 32000-family chips and the
time processing unit in Motorola’s 68332. While these chips have a narrower range of appli-
cation that the RISC processors, for their targeted applications they provide higher perform-
ance at a lower price.

Motorola’s 68000 Family

Motorola’s 68000-family microprocessors were once the clear processors of choice for
embedded systems, and they still are the most successful in terms of production quantities.
They have been under persistent attack by a legion of RISC vendors, however, and Motorola
has been fighting back with lower-cost versions of its general-purpose processors and new
processors that offer more on-chip features.

The “EC” versions of Motorola’s standard 68000-family processors eliminate some fea-
tures and are less expensive that their standard counterparts, but don’t provide higher integra-
tion. The 68EC040 is a 68040 with the floating-point unit and memory-management units
removed. The 68EC030 is simply a 68030 with the memory-management unit disabled. It is
possible that Motorola will re-lay out the chip to eliminate the space used by the MMU, but for
now the EC030 is the same piece of silicon as the 68030—at less than half the price. Mo-
torola’s test costs are slightly reduced, but the need for lower prices to protect its embedded
market is the major reason for the price differential.

The 68EC020 and 68EC000, unlike the 68EC030 and 68EC040, are not pin-compatible
with their predecessors because they eliminate some signals and change some timing para-
meters. They are essentially updated versions of the old designs, taking into account how the
chips have been used and deleting signals that weren’t needed. Both the EC020 and the
EC000 replace the three-wire bus-arbitration handshake (bus request, bus grant, and bus
grant acknowledge) used by their predecessors with a simpler, more conventional re-
quest/grant pair. The 68EC020 reduces the address bus width from 32 to 24 bits, limiting the
address space to 16 Mbytes. The EC020 also drops four rarely-used control signals.

Motorola’s 68300 family is more interesting in that it provides a higher level of integra-
tion to reduce chip count for embedded applications. The chips in this family include a
programmable bus interface that eliminates external address decoders and other glue logic.
The on-chip memory and peripherals vary among members of the family. Table 1 summa-
rizes the key features of Motorola’s 68300 family devices. (Toshiba also makes two 68300
devices, the 68301 and 68303.)
Table 1. Motorola’s 68300-Family Microprocessors

<table>
<thead>
<tr>
<th>Part No</th>
<th>CPU Core</th>
<th>Clock Rate</th>
<th>Timers</th>
<th>Serial I/O</th>
<th>Parallel I/O</th>
<th>RAM</th>
<th>DMA</th>
<th>Chip Selects</th>
<th>Bus Interface</th>
<th>External Interrupts</th>
<th>Packages</th>
<th>Approx. Price (1000)</th>
</tr>
</thead>
<tbody>
<tr>
<td>H302</td>
<td>68HC000</td>
<td>16.67</td>
<td>(2) 16-bit</td>
<td>3 channels w/ comm processor</td>
<td>28</td>
<td>1152 dual-port</td>
<td>6 for on-chip ports</td>
<td>4</td>
<td>16 data 24 adrs</td>
<td>7</td>
<td>132 QGFP 132 PGA</td>
<td>$39.60</td>
</tr>
<tr>
<td>H310</td>
<td>CPU32</td>
<td>16.77</td>
<td>—</td>
<td>—</td>
<td>16</td>
<td>—</td>
<td>—</td>
<td>4</td>
<td>16 data 32 adrs</td>
<td>7</td>
<td>132 PQFP</td>
<td>$19.90</td>
</tr>
<tr>
<td>H311</td>
<td>CPU32</td>
<td>16.77</td>
<td>GPT</td>
<td>Queued serial module (GSM)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>11</td>
<td>16 data 24 adrs</td>
<td>7</td>
<td>132 PQFP</td>
<td>$27.50</td>
</tr>
<tr>
<td>H312</td>
<td>CPU32</td>
<td>16.77</td>
<td>TPU</td>
<td>QSM</td>
<td>—</td>
<td>2K</td>
<td>—</td>
<td>11</td>
<td>16 data 24 adrs</td>
<td>7</td>
<td>132 PQFP</td>
<td>$36.00</td>
</tr>
<tr>
<td>H3140</td>
<td>CPU32</td>
<td>16.77</td>
<td>(2) 16-bit</td>
<td>2 USARTs</td>
<td>16</td>
<td>—</td>
<td>2 channels</td>
<td>4</td>
<td>16 data 32 adrs</td>
<td>7</td>
<td>144 QGFP 145 PGA</td>
<td>$30.75</td>
</tr>
</tbody>
</table>

Prices are for quantity 1000, in U.S., in least-expensive package, as of 4/91.

The 68300 family is really two families: the 6830x devices use a 68000 processor core, while the 6833x and 6834x devices use the “CPU32” processor core. The CPU32 core implements a subset of the 68020 instruction set, plus a few new instructions. The CPU32-based devices have a 32-bit processor core but a 16-bit data bus.

The 68340 is the most general-purpose, high-end device of the 68300 family, and includes an on-chip DMA controller. The 68330 is a stripped-down derivative of the 68340. It eliminates that chip’s DMA controller, serial interface, and timers, leaving only the CPU32 processor core and the system integration module (SIM). The 331 adds two serial channels and a general-purpose timer.

The 68332 was originally designed as an engine controller for General Motors, and it includes a complex “time processing unit.” This is a separate I/O processor, with fixed, on-chip microcode that performs functions such as stepper motor control, input timing, and pulse-width modulation. The 68332 provides high-performance timing capabilities without requiring a very fast processor core by off-loading the time critical functions to the dedicated time processing unit.

National’s 32000 Family

National Semiconductor’s 32000 family was originally developed for general-purpose computer applications, but National changed its plans in 1988 and has introduced a family of devices for embedded control applications, focusing on imaging applications such as laser printers and fax machines. The first two embedded devices were the 32CG16 and the 32GX32, which are simple derivatives of the earlier 32016 and 32532 general-purpose processors. A more interesting family of devices, summarized in Table 2, was introduced in 1990. These are the first general-purpose processors to include hardware support for digital signal processing functions.

The 32FX16 is based on the 32CG16 processor core, and it was designed especially to
support fax modem modulation and demodulation, as well as laser printer control. It provides a remarkably inexpensive solution for applications such as plain-paper fax machines, for which it can implement the fax modem as well as the laser printer control and raster image processor functions.

The 32CG160 adds on-chip peripherals and a fast multiplier to the 32CG16. It doesn’t have as much DSP capability as the 32FX16, but it has on-chip DMA and interrupt controllers plus faster BitBLT capability.

The 32GX320 is an enhanced version of the 32GX32 that adds a fast multiplier and special instructions for DSP support, plus the same set of on-chip peripherals as the 32CG160. This is the only device of the three to provide a 32-bit external data bus, and is based on the faster GX processor core with a 512-byte instruction cache and a 1K-byte data cache.

Most recently, National expanded its high-end offerings with the 32SF641 “Swordfish” microprocessor. This is a 32000-family device in name only; it implements an entirely new instruction set, following traditional RISC principles, and is not binary-compatible with the other members. It is a superscalar design, capable of executing two instructions per clock cycle, and it includes a very fast on-chip floating-point unit and a single-cycle integer multiplier. The efficient RISC design combined with the single-cycle multiplier and fast FPU make the Swordfish faster than traditional high-end DSPs on signal-processing benchmarks while providing much higher general-purpose computation performance.

**AMD’s 29000 Family**

AMD’s 29000 was the first RISC microprocessor to be aggressively marketed for embedded control applications. The 29000 is a follow-on to AMD’s 2900-family bit-slice microprocessors in the sense that it addresses the same market, but there is no architectural similarity. The 29000 architecture is a classic RISC design, with one major exception: it has a large, 192-word register file. The register file can be used to implement a SPARC-like register windows scheme, or it can be used as a stack cache or simply as a large, flat register file.
### Table 3. AMD’s 29000-Family Microprocessors

<table>
<thead>
<tr>
<th>Device</th>
<th>FPU</th>
<th>MMU</th>
<th>Data Cache</th>
<th>Instr. Cache</th>
<th>Buses</th>
<th>Clock Speeds</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>29000</td>
<td>external coprocessor</td>
<td>yes</td>
<td>none</td>
<td>512-byte BTC</td>
<td>32 instr, 32 data, 32 adrs</td>
<td>16, 20, 25, 33 MHz</td>
<td>$66, $93, $121, $146</td>
</tr>
<tr>
<td>29005</td>
<td>external coprocessor</td>
<td>no</td>
<td>none</td>
<td>none</td>
<td>32 instr, 32 data, 32 adrs</td>
<td>16 MHz</td>
<td>$50</td>
</tr>
<tr>
<td>29030</td>
<td>external coprocessor</td>
<td>yes</td>
<td>none</td>
<td>8K</td>
<td>32 instr/data, 32 adrs</td>
<td>25, 33 MHz</td>
<td>$130, $177</td>
</tr>
<tr>
<td>29035</td>
<td>external coprocessor</td>
<td>yes</td>
<td>none</td>
<td>4K</td>
<td>32 instr/data, 32 adrs</td>
<td>16 MHz</td>
<td>$89</td>
</tr>
<tr>
<td>29050</td>
<td>on-chip</td>
<td>yes</td>
<td>none</td>
<td>1K BTC</td>
<td>32 instr, 32 data, 32 adrs</td>
<td>20, 25, 33, 40 MHz</td>
<td>$198, $225, $270, $325</td>
</tr>
</tbody>
</table>

Prices are for quantity 1000, in U.S., in least-expensive package, as of 7/91.

The 29000 family now has five members, as shown in Table 3. The original 29000 has no on-chip data cache and a special type of instruction cache, called a branch target cache (BTC). The BTC stores only the first four instructions after a change in program flow. The BTC is designed to provide the processor with instructions during the memory latency after a jump or call instruction is executed. The 29000 provides best performance with a high-bandwidth memory system that can provide the processor with instructions at a one-per-clock rate after the initial latency.

The 29000 uses a unique bus architecture. It has separate instruction and data buses but only a single address bus. The address bus normally provides data addresses, and it outputs an instruction address only when there is a change in program flow. Thus, the memory controller must have a counter to provide incrementing addresses during sequential program execution.

The 29005 is a cost-reduced version of the 29000 that is actually a partially functional version of the same die. The 29005 does not have the BTC or the MMU and is in a plastic package, but it is otherwise identical to the 29000. The 29050, the high-end member of the family, adds a fast on-chip floating-point unit to the 29000. It also includes a few performance enhancements, including a larger BTC.

The 29030 is the first of a new generation of 29000-family microprocessors. It replaces the BTC with an 8K-byte, conventional instruction cache, and replaces the separate instruction and data buses with a single, combined bus. The 29030 is designed to reduce system cost by eliminating the need for separate data and instruction memories. It also supports an optional half-speed bus clock, allowing a slower memory system to be easily accommodated.

The 29035 is a cost-reduced version with only half as much cache; as with the 29005, it gives AMD a way to sell partially functional chips and provides two price points from one chip design. This technique is also used by Intel’s 960-family processors and IDT’s embedded MIPS processors.

Surprisingly, none of the 29000-family chips have any on-chip peripherals, such as a DMA controller, interrupt controller, or timer, nor do they have any chip-select logic. AMD is likely to rectify this weakness in its next-generation devices.
Table 4. Intel’s 960-Family Microprocessors

<table>
<thead>
<tr>
<th>Device</th>
<th>FPU</th>
<th>MMU</th>
<th>Data Cache</th>
<th>Instr. Cache</th>
<th>Buses</th>
<th>DMA</th>
<th>Features</th>
<th>Clock (MHz)</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>i960KA</td>
<td>no</td>
<td>no</td>
<td>4 register sets</td>
<td>512 bytes</td>
<td>32 mux adrs/data</td>
<td>no</td>
<td></td>
<td>10</td>
<td>$27.10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16</td>
<td>$32.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20</td>
<td>$39.10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>25</td>
<td>$51.20</td>
</tr>
<tr>
<td>i960KB</td>
<td>on-chip</td>
<td>no</td>
<td>4 register sets</td>
<td>512 bytes</td>
<td>32 mux adrs/data</td>
<td>no</td>
<td></td>
<td>10</td>
<td>$35.20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16</td>
<td>$42.40</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20</td>
<td>$50.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>25</td>
<td>$66.60</td>
</tr>
<tr>
<td>i960SA</td>
<td>no</td>
<td>no</td>
<td>4 register sets</td>
<td>512 bytes</td>
<td>mux 16 data/32 adrs</td>
<td>no</td>
<td></td>
<td>10</td>
<td>$19.20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16</td>
<td>$23.10</td>
</tr>
<tr>
<td>i960SB</td>
<td>on-chip</td>
<td>no</td>
<td>4 register sets</td>
<td>512 bytes</td>
<td>mux 16 data/32 adrs</td>
<td>no</td>
<td></td>
<td>10</td>
<td>$25</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16</td>
<td>$30</td>
</tr>
<tr>
<td>i960MC</td>
<td>on-chip</td>
<td>yes</td>
<td>4 register sets</td>
<td>512 bytes</td>
<td>32 mux adrs/data</td>
<td>no</td>
<td>mil-spec</td>
<td>16</td>
<td>$513</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20</td>
<td>$590</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>25</td>
<td>$675</td>
</tr>
<tr>
<td>i960CA</td>
<td>no</td>
<td>no</td>
<td>1.5K RAM (not cache)</td>
<td>1K</td>
<td>32 adrs, 32 data</td>
<td>yes</td>
<td>Superscalar</td>
<td>16</td>
<td>$81.50</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>25</td>
<td>$97.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>33</td>
<td>$122.80</td>
</tr>
<tr>
<td>i960MM/MX</td>
<td>on-chip</td>
<td>yes</td>
<td>2K</td>
<td>2K</td>
<td>32 adrs, 32 data, 64 &quot;backside&quot;</td>
<td>no</td>
<td>mil-spec, extended architecture</td>
<td>25</td>
<td>$1582</td>
</tr>
</tbody>
</table>

Prices are for quantity 1000, in U.S., in least-expensive package, as of 7/91.

Intel’s 960 Family

Intel’s 960 originated in the now-defunct joint effort of Intel and Siemens to develop a fault-tolerant computer system, and it has its roots in the infamous iAPX432. Although it was originally intended as a general-purpose processor, Intel decided not to challenge the success of its 386 architecture by introducing a competing, general-purpose processor. Instead, the 960 has been marketed solely for embedded control. The initial versions, the 960KA and 960KB, are the chips that were developed for BiiN, but many of the more sophisticated capabilities (such as the MMU and object-oriented protection scheme) are not supported. Table 4 summarizes the current 960 family members.

While the 960 is commonly considered to be a RISC processor, it is not a pure RISC. It has complex addressing modes and a number of very complex, microcoded instructions. It does have the RISC attributes of a load/store instruction set, single-cycle execution of many instructions, pipelined implementation, and three-operand instructions.

The 960KA and KB do not have any on-chip peripherals or a programmable bus interface; this is a result of the fact that they were not originally designed for embedded applications. The only difference between the two is that the KB has an on-chip floating-point unit. Both versions have a 512-byte instruction cache. Instead of a data cache, the chips have four register banks, allowing rapid task switching—as long as you don’t run out of banks. The 960MC adds a memory management unit, but it is marketed only for military applications at a relatively high price. The KA, KB, and MC are actually all the same die, with different sets of functions enabled. The 960SA and SB are low-cost versions of the KA and KB that use a
Table 5. High-Integration Embedded SPARC and MIPS Microprocessors

<table>
<thead>
<tr>
<th></th>
<th>IDT R3051/52</th>
<th>LSI LR33000</th>
<th>Fujitsu MB86930</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction Set</td>
<td>MIPS</td>
<td>MIPS</td>
<td>SPARC with</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>extensions</td>
</tr>
<tr>
<td>Instruction Cache</td>
<td>8K (3052)</td>
<td>8K</td>
<td>2K</td>
</tr>
<tr>
<td></td>
<td>4K (3051)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Cache</td>
<td>2K</td>
<td>1K</td>
<td>2K</td>
</tr>
<tr>
<td>Cache Organization</td>
<td>Direct-Mapped</td>
<td>Direct-Mapped</td>
<td>Two-Way</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Set-Associative</td>
</tr>
<tr>
<td>Snooping</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Cache Locking</td>
<td>No</td>
<td>Limited</td>
<td>Yes</td>
</tr>
<tr>
<td>MMU</td>
<td>Yes (E versions)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Static Design</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Write Buffer</td>
<td>4-Level</td>
<td>1-Level</td>
<td>1-Level</td>
</tr>
<tr>
<td>DRAM Control</td>
<td>No</td>
<td>Yes</td>
<td>Partial</td>
</tr>
<tr>
<td>Prog. Wait States</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Prog. Chip Selects</td>
<td>None</td>
<td>2 (fixed)</td>
<td>8 (programmable)</td>
</tr>
<tr>
<td>Breakpoint Registers</td>
<td>None</td>
<td>Address Only</td>
<td>Address and Data</td>
</tr>
<tr>
<td>Counter/Timers</td>
<td>None</td>
<td>2 General-Purpose</td>
<td>1 (Limited)</td>
</tr>
<tr>
<td>Address/Data Buses</td>
<td>Multiplexed</td>
<td>Non-Multiplexed</td>
<td>Non-Multiplexed</td>
</tr>
<tr>
<td>Price (25 MHz, 1000s, POFP)</td>
<td>$54 (R3051)</td>
<td>$99.95</td>
<td>&lt; $50 (20 MHz)</td>
</tr>
<tr>
<td></td>
<td>$81 (R3052)</td>
<td></td>
<td>&lt; $80 (30 MHz)</td>
</tr>
</tbody>
</table>

Prices are for quantity 1000, in U.S., in least-expensive package, as of 11/90.

16-bit external data bus and a simplified bus protocol. The 960SB is notable as the least expensive microprocessor with an on-chip floating-point unit.

The 960CA was the first superscalar microprocessor: it can decode and dispatch up to three instructions in a single clock cycle, although it can sustain two instructions per clock at best. In addition to being much faster than the KA/KB and SA/SB parts, it has a four-channel DMA controller, an interrupt controller, and a flexible bus interface controller. While the earlier parts were marketed for embedded control, the CA was actually designed for embedded control. It has a 1K-byte instruction cache, and instead of the multiple register banks of the earlier chips, it has a 1.5K-byte data RAM. This is not a cache, but a fixed RAM. It can, however, provide fast context switching by using it to store up to 15 alternate register sets which can be rapidly saved and restored.

The high-end 960-family devices are currently marketed for military applications only. The 960MM and 960MX add on-chip memory management and floating-point units to the CA, and also extend the CA's superscalar capabilities. The MX provides the full, object-oriented data security scheme (using a 33rd tag bit on all memory locations) that was originally developed by BiiN. So far, these are the only high-end 960 family members with on-chip floating-point.

Embedded SPARC and MIPS Processors

The most recent entrants to the embedded processor world are embedded versions of the SPARC and MIPS architectures from Fujitsu, IDT, and LSI Logic, as shown in Table 5. These
processors combine processor cores based on the well-known SPARC and MIPS architectures with peripherals, on-chip caches, and bus interfaces that make them well-suited for embedded applications. As the table illustrates, each vendor has provided a slightly different set of features, but the general concept behind all the devices is the same.

IDT’s R3051/3052 family is available with or without an MMU (an “E” suffix indicates a part with an MMU), while the others don’t have MMUs. IDT also offers two instruction cache sizes, 4K or 8K. The IDT chip lacks on-peripherals or a programmable bus interface, however.

Each company has a quite different view of how much cache is appropriate and how to split it between instructions and data; LSI offers an 8K instruction cache but only 1K of data cache, while Fujitsu provides 2K of each. LSI’s chip offers more DRAM control functions and more capable timers, but it is relatively expensive.

The embedded SPARC and MIPS processors represent a significant challenge to the dominance of AMD’s 29000 and Intel’s 960 families in high-end embedded control applications. These chips have only recently become available, so they are years behind Intel and AMD in collecting design wins. They include more on-chip peripherals and superior bus interfaces, however, and also benefit from their compatibility with leading workstations.

Conclusions

In the limited space available, we’ve quickly covered most of the major 32-bit embedded processors. There are several we skipped over, however, that deserve consideration.

Intel’s i860 (now available in two versions, the XP and the XR) doesn’t, for the most part, address the same markets as the processors described above. It is widely used in graphics accelerators, however, which are technically embedded applications. The key strengths of the i860 are its large on-chip caches (12K total in the XR and 32K total in the XP) and its very high floating-point performance. It also has special graphics instructions that are useful in some mid-range rendering applications.

VLSI Technology’s ARM (originally Acorn RISC Machine, now renamed Advanced RISC Machine) processors were among the earliest RISCs marketed for embedded applications, but VLSI lacked the marketing muscle to compete with Intel and AMD. The processor also lacks versions with a floating-point unit or an MMU. The development work for future versions has been taken over by a new company jointly funded by Acorn Computers, Apple Computer, and VLSI Technology, and a new version is expected to emerge from this effort this fall. ARM’s strength is in its low power consumption.

Inmos’ Transputer is the most unusual of the embedded processors. The key feature of the Transputer is its on-chip serial “links,” which are used primarily for connecting multiple transputers. The links are directly supported by the instruction set and enable fast, low-overhead communication between processors. Transputers are most attractive for applications that lend themselves to multiprocessor implementations. While often classified as a RISC
architecture, the Transputer has an unusual, stack-based instruction set architecture that is nothing like the more conventional RISCs. The latest Transputer, the T9000, dramatically raises the performance and introduces a new, faster link design.

Digital signal processors are another entire category of embedded processors that we have ignored in this paper. The classic features that distinguish a DSP from a general-purpose processor are a fast multiply-accumulate function, special addressing modes for handling circular buffers, and the ability to access multiple operands in a single cycle. They are also notoriously difficult to program. DSPs have traditionally been used in signal-processing applications, such as modems, but they are adding more general-purpose capabilities and are showing up in applications such as disk drive control. At the same time, some general-purpose processors are adding DSP capabilities, and the two classes of architectures may gradually become less distinct.

The variety of 32-bit embedded control processors is truly overwhelming, and making an optimal selection is a challenging task that is only getting more difficult. Perhaps you can take comfort in the fact that there are many good choices.

References

All of the microprocessors described above have been extensively covered in the *Microprocessor Report* newsletter. Single copies are available; call (707) 823-4004 for more information. The following list gives the issue date for selected articles covering the processors mentioned in this paper. A reprint book titled "Understanding RISC Microprocessors" is also available, and includes all the articles cited below and many more.

Motorola 68ECxx: “Motorola Streamlines 68000 Family,” 4/17/91
Motorola 68332: “Motorola Launches New Microcontroller Generation,” 5/89
Motorola 68340: “Motorola’s 68300 Family Gains New Member,” 9/19/90
AMD 29000: “AMD’s 29000 Exploits Large Register File,” 8/88
AMD 29050: “AMD Formally Unveils Long-Awaited 29050,” 10/3/90
AMD 29030/35: “AMD’s 29030 Lynx Reduces System Cost,” 5/15/91
Intel 960KA/KB: “Intel Unveils Radical New CPU Family,” 4/88
Intel 960CA: “Intel 80960CA Breaks the IPC Barrier,” 9/89
LSI Logic LR33000: “MIPS-Based Processor Provides High Integration,” 10/31/90
IDT R3051/52: “IDT Introduces Embedded MIPS Processors,” 10/3/90
Fujitsu SPARClite: “Fujitsu Introduces Embedded SPARC Processor,” 11/14/90
Inmos T4/T8 Transputers: “Transputer Provides MP Building Block,” 4/17/91
Inmos T9000 Transputer: “T9000 Boosts Transputer Performance,” 5/15/91
Intel i860XP: “Second-Generation i860 Premiers as i860XP,” 6/12/91