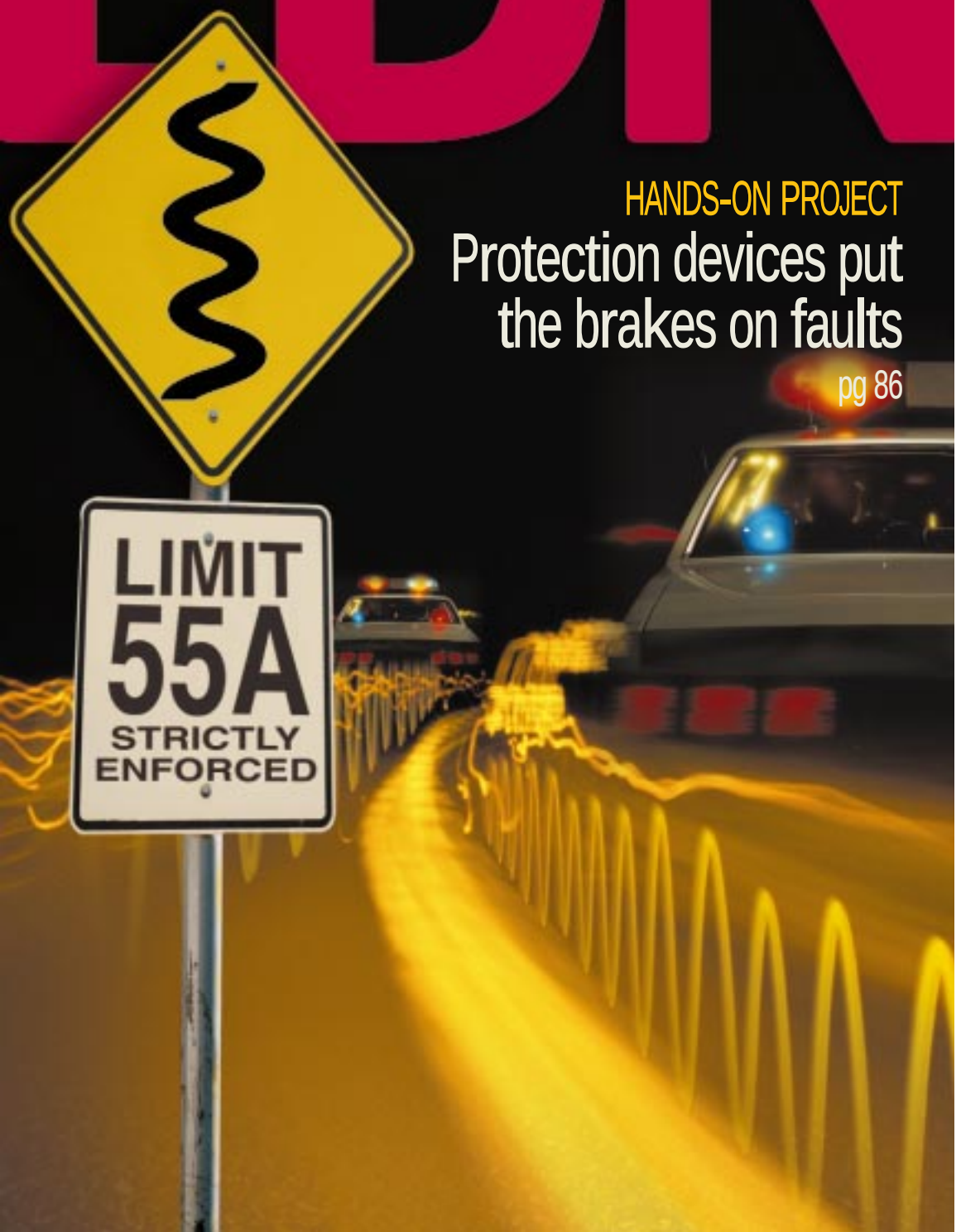
EDN
THE DESIGN MAGAZINE OF THE ELECTRONICS INDUSTRY
February 3, 2000
www.ednmag.com




HANDS-ON PROJECT

## Protection devices put the brakes on faults

pg 86

# Development tools unleash network processors' power

**A NEW GENERATION OF MULTI-PROCESSOR NETWORK PROCESSORS PROVIDES PERFORMANCE TO BUILD NETWORKING EQUIPMENT. BUT STATE-OF-THE-ART DEVELOPMENT TOOLS ALLOW YOU TO TAP INTO THAT PERFORMANCE. EXAMINE HOW THE VENDORS ALLOW YOU TO SIMULTANEOUSLY CONTROL AND MONITOR MULTIPLE PROCESSORS.**

People often want more than they have, and such is the case for data throughput on the Internet. This desire for more throughput ultimately translates into a need for more network processing power. To help solve this bandwidth problem, more than 20 companies are now players in the network-processor market, and many other companies are developing programmable technology that could play a major role in this market. But the ability to unleash the power of these processors depends heavily on the quality and availability of development tools. This situation is particularly true for more complex network processors from companies such as C-Port, IBM, Level One/Intel, Maker (now Conexant), and T.sqware.

These network processors, which "represent the Holy Grail—the panacea of network bottlenecks at the 'edge'" are especially complex because they each contain multiple processing engines, as well as a range of "system-level" resources. (**Reference 1**). Without high-

quality development tools, developing systems with these processors would be a nightmare. But what constitutes a high-quality development tool? And do any of these companies have the perfect solution?

### THREADS SEW UP EFFICIENT EXECUTION

One company in the network-processor market, Level One, designed its IXP1200 processor to serve as a data controller for applications that require access to fast memory subsystems, as a fast interface to I/O devices such as network media-access-controller (MAC) devices, and with enough processing power to efficiently manipulate data. No matter what type of data-manipulation application you build, the IXP1200's tool set provides the capability and flexibility to allow you to optimize performance. Regardless of whether you plan to design with this processor, you will find its development-tool technology interesting. The tool represents state-of-the-art technology because of its ability to simultaneously keep track of all of the chip's resources. However, because of the complexity of the chip; a unique instruction set lacking C support; and the lack of high-level modules, which, however, Level One hopes to provide soon, don't expect to program your application in an afternoon.

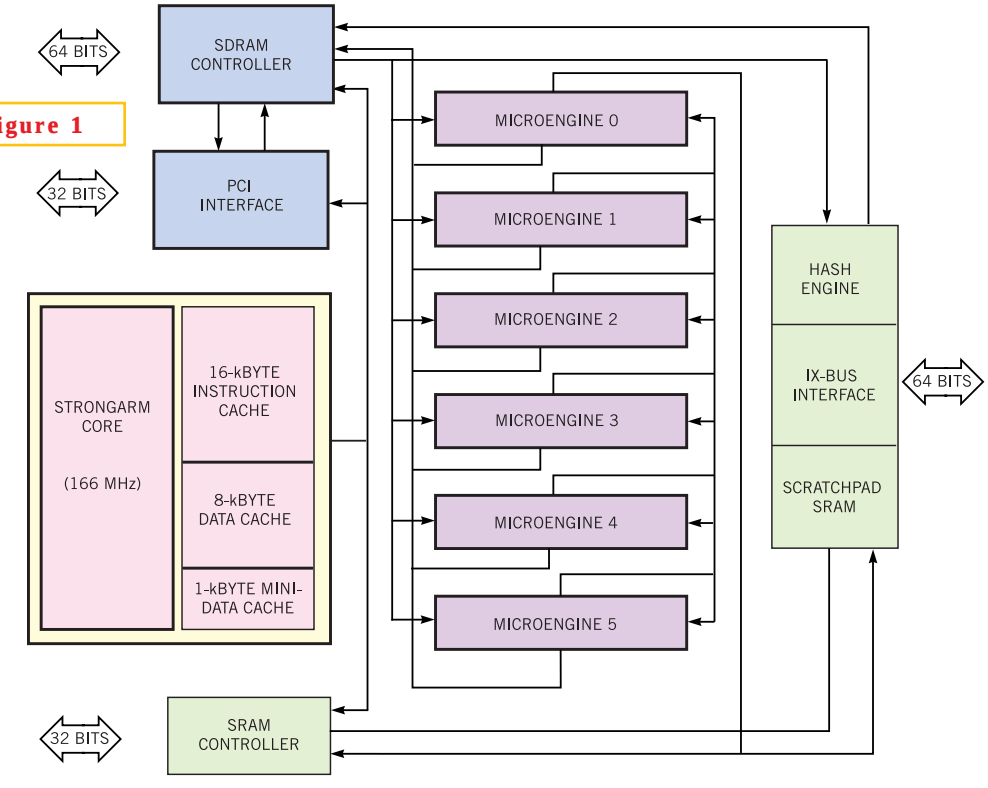
The most important feature of the IXP1200 is a StrongARM core, which Level One combined with six independent, programmable microengines (**Figure 1**). The company linked all the IXP1200's functional units using the chip's IX bus interface that provides a 4.2-Gbps interface to I/O devices. You can configure the IX bus as a single 64-bit, bidirectional bus or as two 32-bit, unidirectional buses operating at 66 MHz. The interface also contains a Hash unit, transmitting and receiving FIFO buffers, system-level control-status registers, and 1k word of scratchpad memory. Whereas the StrongARM core is responsible for the system's mundane management tasks, the microengines, 32-bit RISC processors, are the real workhorses. Each microengine has hardware support for four execution threads. This support comprises independent program counters and 32 data-transfer registers for each thread. Each microengine also contains 1024-word instruction storage and 128 general-purpose registers. The hardware-thread support yields single-cycle context switching between threads; however, you can use a deferred token within an instruction to obtain zero-overhead switching. The deferred token is one of three mechanisms that the IXP1200 supports to reduce or eliminate aborted cycles introduced as a result of a branch decision. A deferred token allows the microengine to execute the instruction that follows a branch decision before the branch takes effect. The IXP1200 assembler supports an optimization option that automatically performs deferred-branch optimization by recognizing when instructions can be deferred and rearranging the instruction accordingly. (The second mechanism relies on condition codes from a previous instruction. The third mechanism, guess branch, allows the microengine to prefetch instructions from the "branch-taken" path before the mechanism makes the branch decision.)

The microengine threads determine when they should go to sleep and allow another thread to run, but the context-event arbiter determines which microengine thread wakes up. The context-event arbiter

**AT A GLANCE**

- State-of-the-art development tools are necessary to tap into the state-of-the-art power of network processors
- Level One's IXP12DE Workbench provides tremendous insight into a multiprocessor system
- Although many network processor vendors provide ready-made software components, it is still helpful if they also provide performance profiling tools.

Figure 1




**The IXP1200 packs six microengines that handle as many as 24 hardware-supported threads.**

makes this decision based on the status of the wake-up signal event that the thread enables. Example signal events are when a synchronous DRAM (SDRAM), SRAM, or PCI-bus request is ready. The IXP1200 has separate SRAM and SDRAM units that support multiple read and write queues. When a thread requests a memory resource, that request enters one of the associated queues for sequential access. An efficiently written thread would then switch context and allow another thread to execute while waiting for the signal event. This scheme not only improves the efficiency of off-chip memory accesses, but also simplifies the programming model because it allows you to think in discrete execution blocks. Furthermore, each thread of a microengine can be the exact same code as all the other threads, allowing you to easily expand the performance of your application.

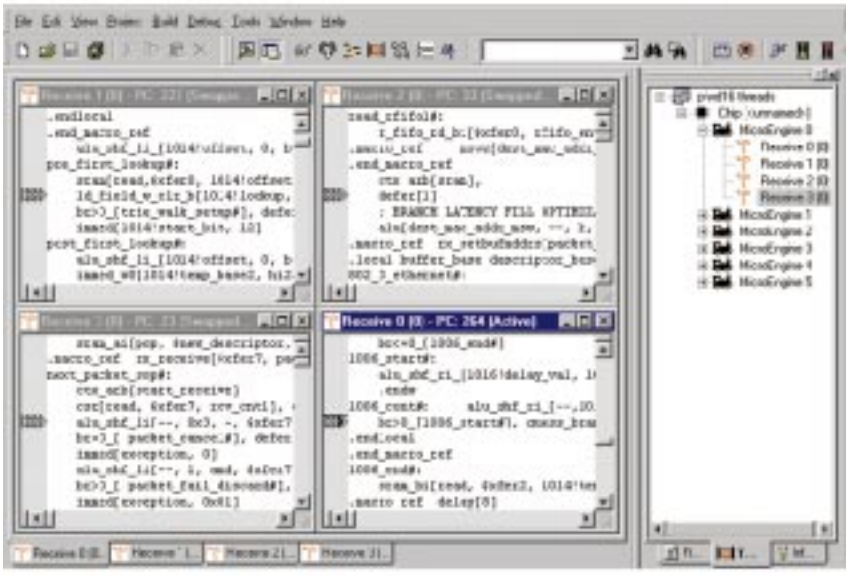


**Figure 2** The IXP12DE allows you to simultaneously view the execution state of the six microengines and their associated threads. The black arrow in the Receive 0 thread window indicates that this thread is currently executing.

## FIRING UP THE ENGINES

Level One's development tool kit, the IXP12DE Workbench, allows you to tame all the capability of this network processor. Like most other tool kits, the Workbench comes with a microcode assembler/optimizer and a debugger that performs the standard breakpoints and memory and register viewing. But the most impressive portion of the Workbench is the Transactor, a cycle-accurate simulator.

To activate the Transactor, you click on the "Start Debugging" button. This step initializes the Transactor and loads the microcode and the script files. Script files provide batch-style control of the Transactor or hardware. The scripts perform functions such as initializing the I/O ports and generating the simulated packets.

Once the StrongARM core loads the microcode, the Workbench allows you to access the execution state of all threads in the project. **Figure 2** shows the four execution threads from Microengine 0. You should use a dual-monitor setup if you're serious about debugging on Level One's development platform because of the immense amount of information that it can display simultaneously. You can open more execution thread windows by expanding the other microengines and double-clicking on the desired thread. Close inspection of the thread-execution window reveals arrows that represent the current program counters for each thread. The arrows are grouped as sets of five subarrows; the subarrows indicate which stage of the pipeline the instruction is in. A darkened arrow indicates that an instruction is executing, and a clear arrow indicates that the thread is swapped out and pending execution while it waits for its signal event. (A yellow signal=aborted cycle, a red signal=stalled, and a blue signal=idle.)

The thread-history window uses the same color coding as the pipeline-stage arrows (**Figure 3**). You can use this window to analyze the execution status of each microengine thread on a cycle-by-cycle basis. The vertical dashed line indicates the current analysis point. So, for example, if you want to determine what instruction sequence caused the stall cycle for the Receive 0 thread, you right-click on the red portion of the history line and select "go to instruction." The green arrow in the code window shows the corresponding instruction. By clicking on the "Queues" box in the thread-history window, you can also view the status of all the memory queues and use this information to determine where the bottlenecks lie.

To fine-tune the IXP1200's operation, the Workbench provides a variety of statistical information, such as breakdown of each microengine and execution-thread usage (**Figure 4**). Level One provides a sample reference design that simulates a packet-forwarding design in a network application with 16 100-Mbit Fast Ethernet ports. The receiving scheduler detects incoming packets on the MAC ports and assigns them to receiving microthreads. Receiving threads parse the packet headers, perform look-ups, and store and queue the packet. The transmitting arbiter prioritizes the queues. The transmitting scheduler assigns packets to transmitting microthreads that send the

**TABLE 1—EFFECTS OF INCREASING WORKLOAD FOR IXP1200**

**Data throughput = 2,398,081 packets/sec**

| | % Executing | % Idle |
|---|---|---|
| Micro0 | 65.5 | 20.2 |
| Micro1 | 61 | 25.4 |
| Micro2 | 60.9 | 25.1 |
| Micro3 | 70.2 | 11 |
| Micro4 | 88.9 | 0.2 |
| Micro5 | 87.1 | 0.1 |

**Data throughput with sample Layer 4 code=2,150,537**

| | % Executing | % Idle |
|---|---|---|
| Micro0 | 78.9 | 4.2 |
| Micro1 | 75.9 | 7.4 |
| Micro2 | 75.7 | 7.1 |
| Micro3 | 76.9 | 5.4 |
| Micro4 | 87.8 | 1.2 |
| Micro5 | 86.7 | 0.5 |

packet on the forwarding port. This hands-on project reference design configures the 12 threads of microengines 0 to 2 as receiving threads, the four threads of Microengine 3 as transmitting threads, one thread of Microengine 4 as a receiving scheduler, and two threads of Microengine 5 as transmitting schedulers and transmitting arbiters. You can determine the data throughput and microengine usage for this "application," receiving and transmitting 1000 packets (**Table 1**). Although the data-throughput rate decreases if you include some sample Layer 4 code in the receiving threads, the usage of microengines 0 to 2 significantly increases. Layer 4, the transport layer of the Open Systems Interconnection (OSI) reference model, manages end-to-end communications and monitors and controls the connection between distant computers whose communications may have to travel across more than one network to make and maintain a connection.

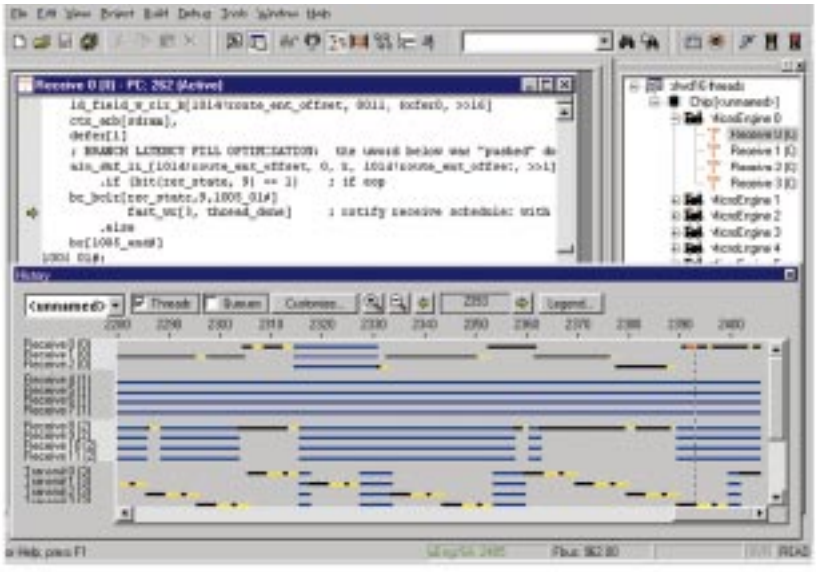


**Figure 3** **A history window in the IXP12DE graphical user interface allows you to view the execution history of all 24 threads, as well as the memory-queue status. You can use this information to determine which operations caused the thread to abort or stall and cause memory-queue overflows.**

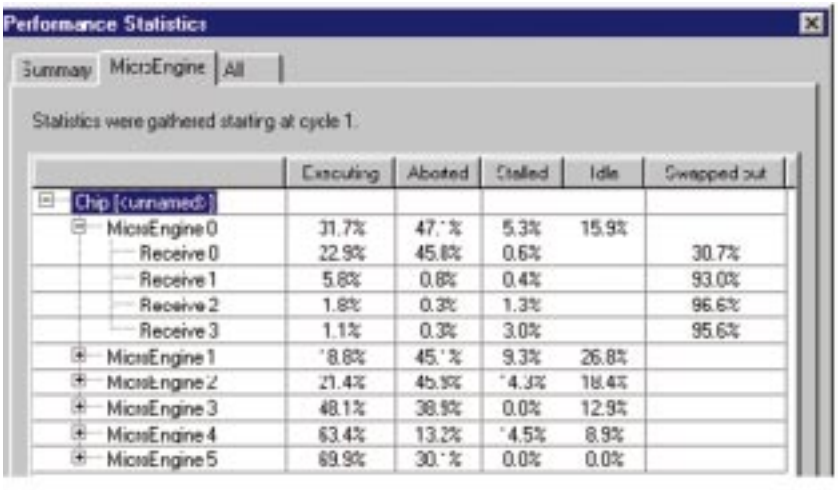



**Figure 4** **Performance statistics show how efficiently your application is using the microengines. This small sample shows the statistics that the IXP12DE can provide.**

## PLUG-IN DEVELOPMENT

Although Level One's Workbench provides many options to allow you to write code and tune your application, the company is working on ready-made code macros to facilitate higher level programmability. Another company, T.-sqware, encourages its customers not to program its TS704 network edge processor. Instead, T.sqware provides modules in binary format that developers can "glue together." Microcode modules may support different communication protocols, and you can use several of them simultaneously. The company also provides an edge-processor interface (EPI) that helps you interface host applications with the TS704. The goal is to abstract the TS704's four SPARClet CPU cores (each with its own HDLC engine) away from the programming details.

Although T.sqware provides high-level modules, the modules have a wide range of parameterizations. However, this high level of abstraction makes it difficult for you to fine-tune your application. Furthermore, T.sqware provides no performance profiler, although you can rig up a loop-back cable on the edge of the chip to evaluate performance. But this approach doesn't help to pinpoint bottlenecks.

The TS704 also includes a diagnostic system unit (DSU), an essential ingredient for debugging this multiprocessor chip. The DSU includes three 256-entry trace buffers for execution and data. The DSU begins a trace when it hits a breakpoint. You can set breakpoints on a per-processor basis as well as any memory reference.

## MAKER MAKES IT, OR YOU MAKE YOUR OWN

Maker Communications also offers a multiprocessor network engine. The company provides a fairly complete set of tools, including a GNU-based C compiler, which should be available this quarter. However, the company lacks an assembly optimizer, so if you are writing in assembly, you must comprehend scheduling issues. One of the challenges of developing a C compiler is ensuring that the resultant binary is optimized enough to satisfy the performance-hungry network stack. Like the IPX1200, the most important performance features relate to scheduling that deals with memory la-

tencies, such as loads and stores, but other features are also important. These features allow an application to take advantage of the data-handling capabilities of Maker's processor. To allow you to profile your application, Maker provides a cycle-accurate Verilog simulator. Although the simulator is slow, you can use it to model the entire chip plus the Universal Test and Operations Physical Interface for ATM (UTOPIA) bus, PCI, and system memory.

Maker also supplies its Gesim postprocessing tool. This tool generates a log file from the simulator output and then provides a graphical view of the application's profile displaying a code-coverage profile, a stall profile, and an execution trace. You can also view buffer usage and all data going across the processor buses. Red bars in Gesim's profile window indicate the number of stalls on instructions within the corresponding subroutine, and the blue bars indicate how many instructions the CPU executes (**Figure 5**).

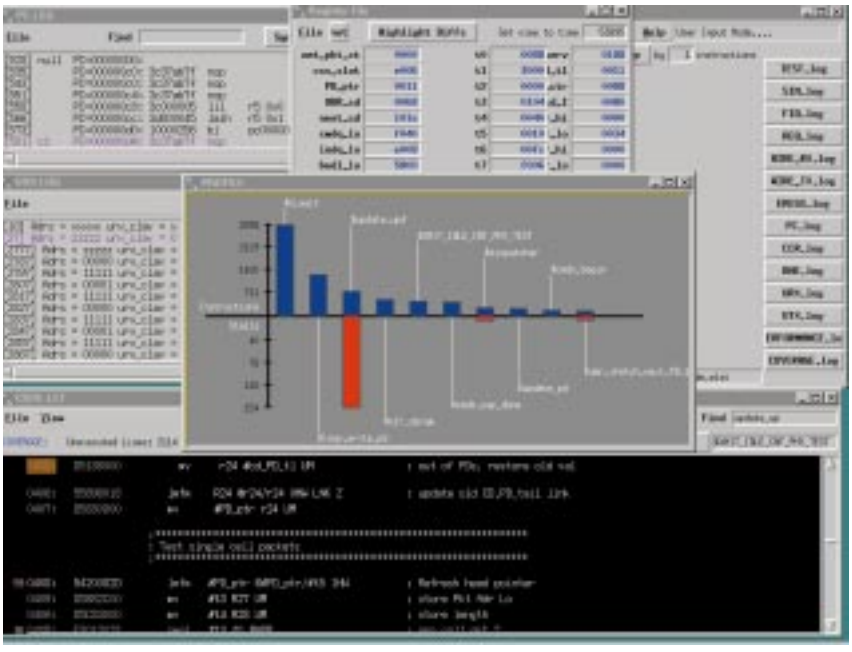


**Figure 5** **Maker Communications' profile window indicates the efficiency of its multiprocessor-network processor.**

Another interesting multiprocessing product comes from C-Port Corp. The C-Ware development platform comprises the C-5 processor, communication-programming interfaces (CPIs), a development environment, and a reference library of networking software. The processor's architecture, containing 16 channel processors and five coprocessors inherently handles cell and packet processing, table-look-up processing, and queue management. Typical CPIs include buffer, media, look-up, interprocessor-messaging, and diagnostic services—sufficient to get you on your way to design completion. The C-Ware tool set includes a compiler and a cycle-accurate simulator that works faster than Verilog. The C-Port tools include a traffic scriptor that allows you to generate simulated traffic for input to the simulator. The simulator supports most of the functions of the 16 channel processors and coprocessors. While the simulator is running, the C-Ware performance analyzer gathers detailed data about your application. You can use this data to develop a database query to display application events and usage queries. You can configure an application-event query to display data as the number of clock ticks between application events on channel processors. A usage query allows you to select a set of resource-usage data on one or more of the C-5's processing resources.

In addition to the huge growth in data traffic, today's networks are more complex and require more flexibility. These networks use multiple technologies, and they often grow to span hundreds or thousands of ports and therefore demand higher performance. Semiconductor vendors are enabling these capabilities by delivering high-performance multiprocessing platforms, but the quality of the tools lets you access that power.□

You can reach Technical Editor Markus Levy at 1-530-672-9113, fax 1-530-672-9103, markus@embedded-benchmarks.com.

Reference

1. Cravotta, Nicholas, "Network processors: The sky's the limit," *EDN*, Nov 24, 1999, pg 108.