Cahners
MICROPROCESSOR REPORT
VOLUME 13, NUMBER 12
SEPTEMBER 13, 1999
THE INSIDERS' GUIDE TO MICROPROCESSOR HARDWARE

# Intel Network Processor Targets Routers

## *IXP1200 Integrates Seven Cores for Multithreaded Packet Routing*

*by Tom R. Halfhill*

Only Intel could have this kind of luck: it gets sued by Digital Semiconductor for patent infringement, ends up acquiring its foe after an out-of-court settlement, gains a billion-dollar fab and a StrongArm license in the deal, and then discovers that it has also inherited a groundbreaking network processor that was secretly under development. Perhaps Intel should encourage competitors to file lawsuits more often.

The network processor is the IXP1200, unveiled last week at the Intel Developer Forum. It's an extremely dense device that integrates seven microprocessor cores on a single die for chip multiprocessing (CMP), as Figure 1 shows. Designed primarily as a substitute for the discrete RISC processors and ASICs in network routers, the IXP1200 can manage 24 independent threads of execution for the highly parallel, data-intensive task of steering packets toward their destinations on networks. Intel says a conservatively clocked (166-MHz) IXP1200 can perform layer-3 routing for 2.5 million 64-byte packets per second.

To meet the future demands of multigigabit and terabit routers, system designers can link dozens or even hundreds of IXP1200 chips together in a complex switching fabric. One vendor is already working on a 180-chip chassis. Priced to move at $200, the IXP1200 will send shock waves through an industry that's used to paying more for proprietary ASICs that can take years to develop.

More important, the IXP1200 is just the opening shot in a broad Intel assault on the communications market. It's the first product in what Intel calls the Internet Exchange Architecture (IXA), which will include companion chips, development tools, and future network processors with even more integration. Intel says it can scale the architecture to dozens of cores and peripherals on a die.

To promote IXA, Intel is starting a quarterly IXA Developer Forum, setting up a $200 million development fund and refocusing part of its $3.5 billion investment portfolio on communications. Although Intel won't comment on specific plans, more acquisitions are a sure bet. In the past three years, Intel has been snapping up networking and communications companies, including Case Technology, Dayna, Shiva, XLNT Networks, Dialogic, Softcom, and Level One. NetBoost is the latest target. It's apparent that Intel views communications as a strategic growth market, not just as a sideline to its desktop/server CPU business (see MPR 8/23/99, p. 24).

Oddly, Intel will market the IXP1200 and related chips under Level One's brand name, whether they are developed by Intel, Level One, or Softcom. An IXP1200-based router will have "Level One Inside," not "Intel Inside"—a strange outcome for a product that didn't originate at either company.

### Grab-Bag Surprise

Digital Semiconductor started working on the IXP1200 in late 1996 and was already deep into development when Intel acquired the company in 1998. After discovering the project,

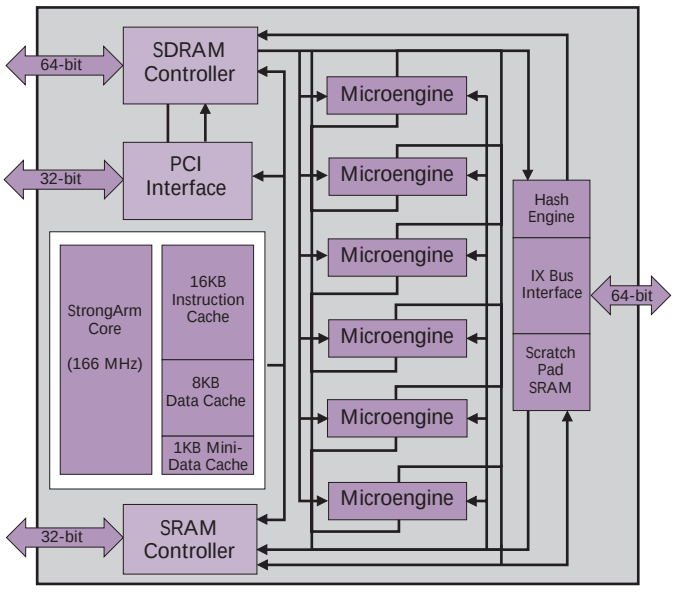


**Figure 1.** The IXP1200 packs seven processor cores on one die.

Intel didn't alter the architecture, but it did instruct the team to "finish faster," says architect Matthew Adiletta. As a result, early samples are in the hands of key customers today; general sampling will begin in 4Q99, with production scheduled for 1Q00.

That's fast work for a new microprocessor architecture this complex, especially with the disruption of a corporate acquisition. But from the beginning, says Adiletta, the design team focused on creating a streamlined architecture for packet processing and high-volume data throughput.

The IXP1200 starts with an SA-1100 StrongArm core and six new RISC cores with an entirely new instruction set. It adds a vast register file, replicated across all the RISC cores; shared memory resources in the form of on-chip scratchpad RAM and I/O buffers; and four integrated bus controllers. The chip connects to PCI, SRAM, SDRAM, and a new interface called the IX bus (for packet I/O and connections to companion chips).

Additional on-chip logic manages independent program threads with relatively little explicit coding by the programmer. There are also some on-chip queues for optimizing external memory accesses. All together, these features yield a device that's ideally suited for network routers (see sidebar).

Despite this level of integration, the IXP1200 is relatively compact and very manufacturable. The die size is only 126 $mm^2$, with 6.5 million transistors, packaged with 432 pins on an enhanced BGA package. Intel will manufacture the chips in the fab it acquired from Digital—the same plant in Hudson, Mass., where StrongArm chips are made.

But while Intel continues to produce the StrongArm in a 0.35-micron IC process, the IXP1200 will be manufactured at 0.28 micron with three metal layers—a little more advanced but hardly cutting edge. The core frequency is only 166 MHz. Obviously, this architecture has plenty of room to grow as it migrates to smaller geometries and higher clock frequencies. Even at 0.28 micron with a 2-V core and 3.3-V I/O, the chip consumes less than 5 W.

## Starting With a Blank Slate

Today's fastest routers combine a general-purpose RISC processor with ASICs that implement proprietary algorithms for packet processing, route-table management, and bus I/O. The ASICs are typically large (200–400 pins) and expensive ($200–$400), and take at least 18 months to develop. A line card for a high-end router might have a dozen ASICs and could retail for more than $50,000.

The IXP1200 tries to usurp both the general-purpose CPU and most of the ASICs. The chip's StrongArm core replaces the CPU. It runs the supervisory software that performs less-critical tasks, such as managing the router's lookup tables. The original designers at Digital chose a StrongArm core for this purpose because it's fast, consumes very little power, occupies a small amount of silicon, and is well supported by development tools. (It also happened to be available for free.) Digital's StrongArm expertise proved invaluable because there wasn't a reusable StrongArm core—

## An Architecture for Networking

Intel's IXA architecture departs so radically from that of general-purpose CPUs because the design team had very different goals. Its most obvious goal was to exploit the inherent parallelism of packet processing. A less obvious objective was to allow network operators to provide new levels of service.

Routing is an inherently parallel task because each packet that traverses a network is a self-contained package with its own destination header and data payload. A small router on a local-area network handles a relatively small number of destinations, such as the desktop PCs and network printers within the department of a company. But a large router on a wide-area network or an Internet backbone is the traffic cop for millions of unrelated packets on their way to thousands of destinations (such as Web servers and mail servers).

Packet switching allows routers to process each packet independently and out of order, in parallel with other packets. The receiving device at the destination is responsible for reassembling the packets in their original order.

To steer a packet toward its target, a router must parse the header to find the destination address and other information, then shove the packet in the right direction as rapidly as possible—preferably at "wire speed," the maximum capacity of the network pipe. It's a data-intensive job, but processing power is important too, because each packet has multiple headers—one for each protocol that's wrapped around the payload. A typical packet will have a TCP header, an IP header, and an Ethernet header. Each header is several bytes long, and the packets can vary greatly in length, although 64-byte packets are the most common.

Network operators want routers that can look deeper into the headers to make more intelligent decisions about steering the packets. Faster routing is only one motivation. Commercial network operators also want to charge their users different rates for different levels of service, just as the postal service does.

Currently, all packets travel at "bulk rate," because routers can't tell them apart. Routers that can analyze headers more thoroughly without compromising wire speeds will allow network operators to sort packets into the digital equivalents of priority mail, first class, second class, bulk rate, and so on.

For example, in return for higher fees, the encrypted traffic on a corporation's virtual private network (VPN) could get higher priority on an Internet backbone than the idle gossip in an AOL chat room. But differentiating service without compromising wire speeds requires more processing power in the router—and hence, more powerful network processors.

the engineers had to extract one from the SA-1100 processor (see MPR 9/15/97, p. 1).

They also adapted a StrongArm PCI controller and designed a new SDRAM controller, integrating both peripherals onto the IXP1200. As the die photo in Figure 2 shows, those peripheral blocks and the StrongArm core occupy about half of the new chip.

The IXP1200 offloads the grunt work of packet processing onto the six RISC cores, which Intel calls microengines. All CPU architects dream of starting with a blank slate, free from the baggage of older architectures designed for different purposes. For this project, the architects decided that even the efficient StrongArm wasn't ideal for the task at hand, so they created an entirely new 32-bit architecture with a data-oriented instruction set.

Although they retained a few favorite StrongArm features—such as instructions that can perform an ALU operation and a shift in a single clock cycle—they discarded many other StrongArm characteristics, such as the fully conditional instruction set and the conventionally organized register file. As Table 1 shows, the designers added several new instructions for more efficient data throughput, byte manipulation, and pattern matching.

The microengines are truly independent cores. Each has its own physical register file, a 32-bit ALU with a basic five-stage pipeline, a single-cycle shifter, an instruction-control store (4K of SRAM), a microengine controller, and four program counters—one for each thread that a microengine can execute in parallel. The cores are functionally identical, so scaling to higher levels of CMP should be a step-and-repeat job. (Some of the physical layouts look different in the die photo because they're mirror images of the others.)

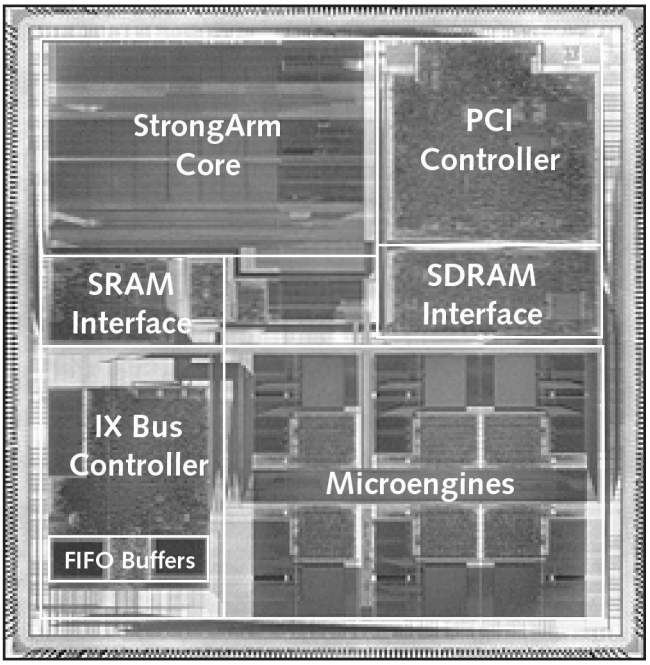


**Figure 2.** The IXP1200's die is 126 mm$^2$ in a 0.28-micron process.

Although the architecture defines a way for microengines on a chip to directly access each others' registers, Intel says this may not be operational in the initial version of the IXP1200. Instead, the microengines exchange data with each other through the shared scratchpad memory (4K of on-chip SRAM). Nonblocking internal buses allow the microengines to access the scratchpad without contention. Intel says this method requires less coherency logic than allowing the microengines to access each others' registers, so

| Name | Description |
|---|---|
| **Arithmetic, Rotate, and Shift Instructions** | |
| ALU | Perform ALU operation |
| ALU_SHF | Perform ALU and shift operation |
| DBL_SHIFT | Join 2 longwords, shift, save longword |
| **Branch and Jump Instructions** | |
| BR, BR=0, BR!=0 | Branch on condition code |
| BR>0, BR<0 | Branch on condition code |
| BR>=0, BR<=0 | Branch on condition code |
| BR=cout, BR!=cout | Branch on condition code |
| BR_BSET, BR_BCLR | Branch on bit set or bit clear |
| BR=BYTE, BR!=BYTE | Branch on byte equal |
| BR=CTX, BR!=CTX | Branch on current context |
| BR_INP_STATE | Branch on event state |
| BR_!SIGNAL | Branch if signal deasserted |
| JUMP | Jump to label |
| RTN | Return from branch or jump |
| **Read/Write Instructions** | |
| CSR | Reference a control/status register |
| FAST_WR | Write immediate data to CSRs |
| LOCAL_CSR_RD | Read CSR |
| LOCAL_CSR_WR | Write CSR |
| R_FIFO_RD | Read the receive FIFO buffer |
| T_FIFO_WR | Write to transmit FIFO buffer |
| PCI_DMA | Issue request to PCI unit |
| SCRATCH | Read/write scratchpad RAM |
| SDRAM | Read/write SDRAM |
| SRAM | Read/write SRAM |
| **Local Register Instructions** | |
| FIND_BSET | Find first bit set in any 16-bit reg field |
| FIND_BSET_WITH_MASK | Find first bit set with mask |
| IMMED | Load immediate word, sign-extend, shift |
| IMMED_B0, IMMED_B1 | Load immediate byte to a field |
| IMMED_B2, IMMED_B3 | Load immediate byte to a field |
| IMMED_W0, IMMED_W1 | Load immediate word to a field |
| LD_FIELD | Load byte(s) into specified field(s) |
| LD_FIELD_W_CLR | Load byte(s) into specified field(s) |
| LOAD_ADDR | Load instruction address |
| LOAD_BSET_RESULT1 | Load result of FIND_BSET instruction |
| LOAD_BSET_RESULT2 | Load result of FIND_BSET_MASK |
| **Miscellaneous Instructions** | |
| CTX_ARB | Swap contexts and wake on event |
| HASHx_48 | Perform 48-bit hash (x=1, 2, 3) |
| HASHx_64 | Perform 64-bit hash (x=1, 2, 3) |
| NOP | No operation |

**Table 1.** This is the complete IXP1200 microengine instruction set, although optional tokens permit many variations of these instructions. Notice the ALU-plus-shift instruction—inherited from StrongArm—and the numerous bit-manipulation and pattern-matching operations.

it should be easier to integrate larger numbers of microengines on future network processors.

## Event-Driven Programming

As might be expected for such a specialized architecture, the IXP1200 has some unusual characteristics. Of particular note are the provisions for multithreading and data throughput.

Intel refers to the IXP1200's style of executing multiple threads as "hardware multithreading," because special logic handles most of the busy work for programmers. In concept, it's remarkably similar to what Sun calls "vertical multithreading" in the new MAJC architecture (see MPR 8/23/99, p. 13). In both cases, the processor hides the long latencies of off-chip memory references by rapidly switching contexts with other threads.

Using this feature on the IXP1200 requires minimal coding. For example, the microengine instruction set has an instruction for reading and writing to SRAM (where routers often store their IP-address lookup tables). An optional token appended to the instruction invokes multithreading :

```
SRAM [read, xfer2, addr_sram, 4],ctx_swap
```

This tells one of the IXP1200's microengines to read four 32-bit words from external SRAM (starting at symbolic address addr_sram) and store the values in four internal registers (starting at register xfer2). The final token, ctx_swap, gives the processor permission to swap out this thread in favor of another thread running on the same microengine until the memory operation is complete. The next line of code in the first thread won't execute until the processor signals the microengine that all four words of data have arrived in the destination registers.

In effect, it's event-driven assembly-language programming, with the events managed automatically by the processor. The IXP1200 can suspend execution of any thread until a specified event (in this case, reading four words of data from external SRAM) allows execution to resume. This allows the processor to hide memory latencies beneath the useful work performed by other threads.

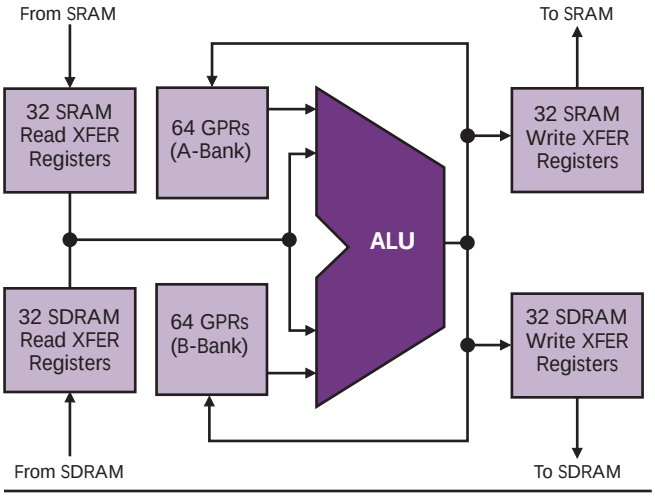


**Figure 3.** Each microengine has 256 registers in separate files and banks. This simplified figure omits the ALU shifter.

The IXP1200 has similar instructions for accessing external SDRAM, the internal scratchpad memory, and a pair of on-chip FIFO buffers for packet I/O. (Each FIFO buffer can hold 16 elements of 64 bytes, the most common size of Internet packets.) Context-swapping is particularly valuable for packet processing, because routers often hold packets in off-chip SDRAM while analyzing the headers.

A common task is to read several bytes of data in sequence—for instance, when a router prepares to examine the nested headers of a packet. A programmer can stack several read (or write) operations like the one shown above in a row, then append the ctx_swap token to only the last instruction in the series. The IXP1200 will enter all of the operations in a queue, swap contexts among other threads while accessing memory, and resume execution at the next line of code only after the last memory transfer is complete.

Other tokens allow additional efficiencies. For instance, the SDRAM instruction has an optimize_mem token that tells the IXP1200 to hold pending memory operations in a pair of special queues for the odd and even banks of SDRAM. While the SDRAM is performing row-and-column operations on one bank, it precharges the opposite bank, saving a few clock cycles as it switches back and forth. An arbiter in the IXP1200's integrated SDRAM controller manages this process. Other than appending the optimize_mem token to the SDRAM instruction, the process is transparent to the programmer.

## More Than a Thousand Registers

To support this data-intensive processing, the IXP1200's six microengines have a vast number of registers: 1,536 in all, not counting special control registers or StrongArm registers.

Each microengine core has 128 general-purpose registers (GPRs) and 128 transfer registers. All are 32 bits wide and are single-ported to reduce circuit complexity. Normally, a single-ported register file would impair performance, because some instructions would require multiple cycles to access all the registers they need. But, as Figure 3 shows, the IXP1200 divides the GPRs into two banks, so ALU instructions that need two source operands can read a value from each bank simultaneously.

The transfer registers are divided into multiple banks as well. There are 64 SRAM transfer registers and 64 SDRAM transfer registers. These banks are further divided into groups of 32 read registers and 32 write registers. The banking scheme allows read/write instructions to retrieve and store results without the delays normally associated with single-ported register files, and without the complexity of multiple read and write ports. Furthermore, separating the register files into separate banks of GPRs and transfer registers allows the microengine instructions to encode 256 register

addresses in seven bits instead of eight, which helps keep the instruction length down to 32 bits.

All read and write operations to and from SRAM and SDRAM use the transfer registers. ALU operations use the GPRs, as in other CPU architectures, but in the IXP1200 they can also use transfer registers for source or destination operands. Thus, it's not always necessary to copy values from transfer registers to GPRs to perform common ALU operations. In many cases, however, programmers will copy transfer registers to GPRs to carry out a more complex series of operations or to make room for additional reads and writes.

The IXP1200 uses the SRAM instruction to access shared internal resources, such as the scratchpad memory, the two FIFO buffers for packet I/O, various control/status registers, and a polynomial hash engine. The hash engine implements some hashing logic in hardware so the processor can more quickly look up IP addresses when doing layer-2 and layer-3 routing. As Table 1 shows, there are special instructions for hashing one, two, or three values at once, and the values can be 48 or 64 bits long. Layer-2 routing requires three hashes per packet; layer-3 routing requires more, depending on the algorithm. The instruction set includes 64-bit hash instructions to accommodate future IPv6 addresses, which will be 64 bits long.

Nonblocking buses connect the hash engine, FIFO buffers, scratchpad memory, and other internal structures to the microengines and external memory. Intel says this "distributed data" architecture is scalable to many more than six microengines on a chip, because it eliminates the extra coherency logic that shared registers would require. Shared resources—such as the scratchpad RAM, SRAM control, and SDRAM control—all maintain their own memory pointers to avoid conflicts among the microengines. And the on-chip buses ensure that external memory, not internal bus bandwidth, is the ultimate bottleneck for data throughput.

For instance, the internal bus that feeds the 64-bit, 83-MHz SDRAM interface is 64 bits wide and runs at the IXP1200's core frequency of 166 MHz, providing twice as much bandwidth internally. The internal bus that feeds the 32-bit, 83-MHz SRAM interface is also 64 bits wide and runs at 166 MHz, providing four times as much internal bandwidth. To utilize that extra capacity, the IXP1200 uses the internal SRAM bus to service the external IX bus by alternating signals between odd and even clock cycles—essentially time-sharing both buses on the same internal pipe.

## Patterned After PCI

The 64-bit, 66-MHz IX bus is based on PCI, but it's faster. It has a simpler bus protocol than PCI and is demultiplexed, with sideband address control. Every little bit counts, because all packets arrive at the IXP1200 over this bus.

The IX bus is also the interface to companion chips or additional IXP1200 processors. Intel says one vendor is already working on a switching fabric that joins 180 IXP1200 chips together in one chassis.

### Price & Availability

For now, Intel is quoting only a quantity-one price for the IXP1200 processor: $200. Customers can get volume discounts, but Intel won't announce discount pricing until later. Some key customers are working with silicon samples today, and general sampling begins in 4Q99. Intel says the IXP1200 will enter volume production in 1Q00. For more information, go to *www.intel.com* and Level One's Web site at *www.level1.com*.

Numerous companion chips are under development at Intel subsidiaries Level One and Softcom, as well as at other companies. Intel's companion chips will be sold under the Level One brand. Some examples: a "scaling engine" that allows designers to link from 2 to 16 network processors together; the IXE2412, a 24-port Fast Ethernet switch with media-access control (MAC); the IXF440, an eight-port 10/100-Mb/s Ethernet MAC; the IXF1002, a two-port Gigabit Ethernet MAC; the IXF6400, an ATM segment-and-reassembly (SAR) formatting engine; and various transceivers.

Intel isn't ignoring the software side either. Although the StrongArm core on the IXP1200 already has plenty of development tools, Intel had to create entirely new tools for the microengine instruction set. Third-party companies will get into the game too, but for now Intel is offering the IXP1200 Developer Workbench, an integrated development environment that includes a symbolic assembler, a debugger, and a cycle-accurate simulator.

The debugger has some nifty features, such as a single-step mode that lets programmers view multiple threads on the screen simultaneously. A special cursor indicates which line of code is currently executing, and even which pipeline stage the instruction occupies. The simulator gathers statistics on microengine utilization and displays a graphical screen that allows programmers to view stalled threads, idle states, the latencies of memory operations, and other events.

Early adopters of the IXA architecture and third-party developers include Broadband Access Systems, Cabletron, CAG Technologies, Cisco, Harris & Jeffries, Hi/fn, InterNiche, Newbridge Networks, Omneon Video Networks, Trillium, T.Sqware, and Wind River.

## A New Class of Processors

Earlier this year, when Intel first disclosed it would introduce a network processor, reports that the chip would have a StrongArm core misled some observers to assume Intel was merely recycling an existing architecture to pick up a little extra business. Now it's plain that Intel has much bigger things in mind. PC processors will remain Intel's mainstay for years to come, and IA-64 provides an opportunity to gain

ground in workstations and servers, but IXA represents an aggressive push into a major growth market.

According to Cahners In-Stat, the worldwide market for routers generated $2 billion of revenue during 2Q99 alone—a 33% increase over 2Q98. Of that total, $833 million was spent on high-end routers, which are by far the most profitable. That was a 42% increase over 2Q98. And the high end is growing faster than the low end, with recent quarterly gains of 16–17% this year.

Of course, Intel isn't the only company to recognize a golden opportunity. In a preemptive strike, IBM Microelectronics and C-Port both announced new network processors at about the same time as Intel and also formed an alliance to develop common APIs for their chips. It's virtually impossible to verify competing performance claims without standard benchmarks and production silicon, but we will compare the Intel, IBM, and C-Port processors in more detail in future issues.

The networking/communications market is growing so hot that several other companies are also touting "network processors"—perhaps encouraged to stretch the definition by the newness of the category. An example is the 61-million-transistor Alliance IPRP-V4, which is 1% logic and 99% fast memory (see MPR 8/2/99, p. 14). Those kinds of chips certainly have a place, but they're not in the same category as true microprocessors like the IXP1200.

The strength of network processors is that they're programmable, off-the-shelf parts specifically created for networking applications. Over time, they will almost certainly displace the general-purpose CPUs and ASICs in today's fast routers. Network processors have a performance advantage over CPU architectures designed years ago for other tasks, and they have numerous advantages over ASICs. Custom ASICs are difficult, expensive, time-consuming, and risky to design, and they're not as adaptable to evolving standards. Network processors are more flexible and should allow router vendors to compress their time-to-market schedules. Even if other solutions cost less per chip—an assumption that Intel's aggressive pricing of the IXP1200 casts into doubt—the price delta probably won't be significant for fast routers, which can cost tens of thousands of dollars.

Network processors may also be suitable for other tasks that handle heavy streams of data. Some possibilities are RAIDs in large servers and the video-switching equipment at cable-TV head-end plants. Just as nobody envisioned all of the applications for PC processors when they first appeared in the 1970s, it's shortsighted to assume that all the applications for network processors are known in the 1990s.

Intel's bombshell validates network processors as a category distinct from other types of processors, sends an unmistakable message about future pricing strategy, and sets the bar for potential competitors. All that with a chip that Intel acquired by accident. It's like Bill Gates winning the lottery. M