Reed Electronics Group

MICROPROCESSOR REPORT

www.MPRonline.com

THE INSIDER'S GUIDE TO MICROPROCESSOR HARDWARE

PicoChip Preaches Parallelism

Massively Parallel PC101 Chip Has 430 16-Bit Processors

By Tom R. Halfhill {7/28/03-02}

Among the most unusual microprocessors unveiled at **Embedded Processor Forum 2003** was picoChip Design's new PC101, a massively parallel device that integrates 430 16-bit processors on a single die. Indeed, the PC101's resources are so abundant that, to some degree, they are expendable—the chip's internal bus fabric can bypass a few processors ruined by manufacturing defects.

Designed for cellular-telephony and wireless-network base stations, the PC101 is the first implementation of picoChip's picoArray architecture. It's based on a three-field long instruction word (LIW), but it has far greater execution resources than other LIW or VLIW processors. PicoChip believes that massive parallelism is the best approach for the compute-intensive tasks of wireless communications, because it can deliver high performance at low clock speeds, thereby saving power. In addition, dividing a complex application into many parallel tasks is well suited for large-team software development projects.

PicoChip, a fabless semiconductor company based in the U.K., will use TSMC to manufacture the PC101 in a 0.13-micron, eight-layer-metal, digital CMOS process. The chip is relatively large and packaged in a 528-pin BGA. Samples are available now, with volume production scheduled to begin in 1Q04.

Lots and Lots of Processors

PicoChip's PC101 is similar in concept to two other architectures presented at **Embedded Processor Forum 2003**: Elixent's D-Fabrix and Motorola's MRC6011. All three architectures have arrays of small processors or self-contained function units capable of executing tasks with a high degree of autonomy. An important goal of these designs is to avoid fetching instructions from off-chip or distant on-chip memory, which consumes I/O bandwidth better reserved for data. Their ability to execute tasks locally, autonomously, and in parallel makes them ideal for data-intensive applications, such as baseband processing in wireless networks. (See *MPR 7/21/03-01*, "Elixent Expands SoCs," and *MPR 7/14/03-01*, "Motorola Attacks ASICs.")

The arrays of local processors on these chips have their own instruction sets and tightly coupled resources, which may include register files, program memory, data memory, and instruction/data caches. Once programmed, the array processors can operate independently, coordinated by higher-level supervisory code that runs on a control-plane processor. A fabric of internal buses weaves the arrays together and allows the processors to share data.

In picoChip's case, at least, the processor-array approach to parallelism isn't a new concept. Some of the company's engineers are veterans of Inmos, which developed the parallel-processing Transputer in the 1980s. Inmos was absorbed by STMicroelectronics many years ago, but the Transputer lives on in legend and has inspired many offspring. Other key employees at picoChip acquired experience in microprocessor development and communications at companies like Conexant, Lucent, Marconi, Oak Technology, and Vodaphone. About half the engineering staff has systems-level experience from previous stints at Fujitsu, Lucent, Motorola, and other companies.

The basic building block of the PC101 is a 16-bit integer processor optimized for communications. Each one is

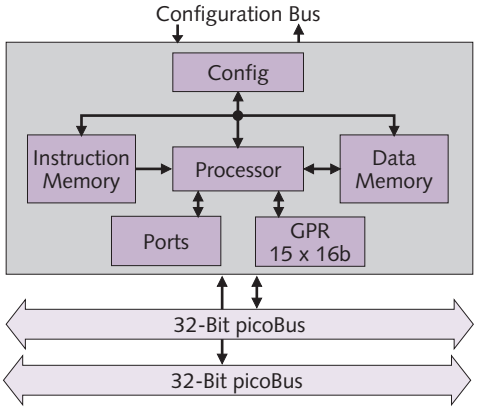


Figure 1. Each of the 430 processors in the PC101 has its own local memory, I/O interfaces to the data fabric, and connection to the global configuration bus. There are four different types of these processors in the PC101, but they share the same basic architecture and substantially the same instruction set. The amount of local memory varies: 768 bytes for each STAN or MAC processor; 8KB for each MEM processor; and 32KB for each CTRL processor.

roughly comparable to an ARM9 processor for control tasks or a Texas Instruments TMS320C55x DSP for signal-processing tasks. There are 430 of these processors in the PC101, but not all of them are the same.

Most numerous are what picoChip calls the STAN (standard) processors, which carry out common integer tasks; there are 240 of them in the PC101. Next most numerous are the MAC (multiply-accumulate) processors; the PC101 has 120 of those. Each MAC processor can execute a 16- × 16-bit multiply-add operation in a single clock cycle, storing the results in a local 40-bit accumulator. Next are the MEM (memory) processors, 68 total, which have more local memory than the STAN and MAC processors. MEM processors typically carry out local control tasks or specialized communications tasks, such as fast Fourier transforms (FFT). Finally, there are two CTRL (control) processors, which supervise the PC101's other 428 processors. Figure 1 shows the basic architecture of a PC101 array processor.

Although labels such as "MAC processor" and "memory processor" make the array processors seem like function units, they are true processors, with their own instruction sets and register files. In a conventional microprocessor, function units almost always share a global register file with other units of the same type, and they rarely have their own instruction and data memories. The PC101's processors are more self-contained than function units, and they are also more versatile. All of them can execute basic integer instructions (two per clock cycle), load/store instructions, and branch instructions. Each processor has its own set of 15 general-purpose registers (16 bits wide), plus some special-purpose registers, such as the MAC accumulators.

The array processors execute instruction words that have one to three fields and can range up to 64 bits in length. Individual operations can vary in length from 8 bits to 64 (the only 8-bit operation is a NOP). Single-operation LIW instructions are typically 16 bits long. Most LIW instructions contain two or three operations and are 24–40 bits long.

As with other LIW/VLIW architectures, each instruction word is limited to scheduling certain types of operations in each field. For example, the first field in a PC101 instruction word must contain an ALU operation, and the second field must contain an ALU, load/store, or interprocessor PUT/GET operation. PUT/GET operations transfer data across the fabric to other processors.

What appears in the third field of an instruction word depends on the type of array processor that will execute the instruction. All processors in the PC101 can execute a branch instruction in the third field. The STAN processors can alternatively execute an application-specific instruction, such as a complex spread or de-spread operation for baseband processing, which can replace 50 conventional DSP operations with one single-cycle instruction. The MAC processors can execute a MAC instruction in the third field, and the MEM and CTRL processors can execute a multiply instruction. Table 1 illustrates the relationship between processors and instruction types in the PC101.

When a PC101-based system initializes, an external host processor or flash-memory controller loads the application code from off-chip memory. The PC101 distributes the code over the fabric's configuration buses to the instruction memories of the array processors. Then the array processors take over, executing their tasks by fetching the LIW instructions from their local instruction memories. Unless a processor must be reprogrammed at run time for a different phase of the program or to run a different program, it can continue to operate autonomously.

Processor Type	LIW Field 0 Operation Type	LIW Field 1 Operation Types		LIW Field 2 Operation Types	
STAN	ALU.0	PUT/GET	ALU.1 or Load/Store	Branch	App-Specific
MAC	ALU.0	PUT/GET	ALU.1 or Load/Store	Branch	MAC
MEM/ CTRL	ALU.0	PUT/GET	ALU.1 or Load/Store	Branch	Multiply

Table 1. Each field in the PC101's three-slot long instruction word (LIW) may contain a particular type of operation, depending on the type of processor (STAN, MAC, MEM, or CTRL). The first bits of each LIW indicate which fields will follow. The compiler omits unused fields, so there's no need for NOP placeholders.

PicoArrays Enable Parallel Processing

Stitching together hundreds of processors in a bus fabric creates a picoArray. The dense web of signal paths and switches allows any processor to share data with any number of other processors, and software developers can program the whole array as a coordinated group. Interprocessor communications are deterministic, because the software-development tools define

the program's signal paths at design time, somewhat as FPGA tools do.

Although the internal buses run at the same speed as the array processors (160MHz in the PC101), the maximum frequency of a signal path between any two processors is 80MHz, because two paths can share the same part of the bus fabric on alternate clock cycles. Likewise, external I/O is limited to 80MHz per port (on eight 16-bit ports). The PC101 relies on massive parallelism, not fleetness, to deliver high performance. Figure 2 shows a small section of the picoArray.

PicoChip says the physical limit on joining multiple picoArrays together is the propagation time for a global synchronization signal, which must reach all the arrays with no more than six nanoseconds of clock skew. To demonstrate the feasibility of multiple arrays, picoChip has designed a development board with four daughtercards, each mounting four PC101 chips. That's a total of 16 chips with 6,880 array processors in a space about 12 inches square, not counting some additional board space for a power supply, Ethernet ports, and a PowerPC host processor. Figure 3 shows how the IPI ports link multiple picoArray fabrics together.

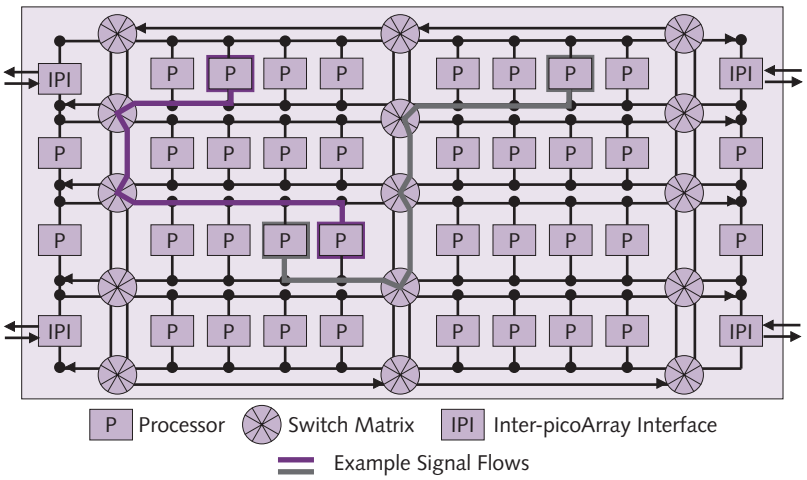


Figure 2. A picoArray organizes quartets of processors within a switched fabric, so any processor in the array can share data with any other processor. This is an abbreviated view; an actual picoArray is much larger. The inter-picoArray interfaces (IPI) provide links to other picoArrays on or off chip.

Parallel processing always seems to work better when sketched on a table napkin than when etched in silicon. The Achilles' heel is software development. One challenge is extracting enough parallelism from a program to keep the processors usefully busy. Another challenge is writing the program code—expressing the parallelism in a way that's explicit enough for the processors but not too confusing for the programmer. Some of picoChip's engineers have been working with parallel architectures since their Transputer days at Inmos, so they bring considerable experience to the problem.

Extracting parallelism is much easier for a communications processor than it is for a general-purpose processor running PC software. The key is to exploit data-level parallelism instead of instruction-level parallelism. With large amounts of data to crunch, a massively parallel processor like the PC101 can distribute the workload across dozens, or even hundreds, of array processors—without the complications of out-of-order execution, dynamic branch prediction, speculative execution, and the other exotic techniques required to wring even tiny amounts of instruction-level parallelism out of superscalar processors.

In addition to extracting data parallelism from communications algorithms, a processor like the PC101—or any processor designed for wireless infrastructures—can also mine a rich vein of parallelism from the datastreams of multiple users. A 64-channel wireless base station offers the opportunity of up to 64 different datastreams that have no mutual data dependencies.

PicoChip cites the example of a 1,024-point FFT, a typical algorithm in baseband processing. Distributing the workload across 14 array processors (3.25% of the PC101's resources) gets the job done in 51.2 microseconds at 160MHz; the rate of throughput is 19,500 data blocks per second. Doubling the number of array processors assigned to the task (to a total of 28) cuts the execution time in half (25.6 microseconds) and doubles the throughput (39,000 blocks per second)—a linear improvement in performance for a linear allocation of resources. Doubling the number of processors again (to a total of 56) will increase throughput to 78,000 blocks per second while allocating only 13% of the chip's processing resources.

One limit on this progression—besides the number of array processors—is I/O bandwidth. However, picoChip says that systems for which the PC101 is designed are more often

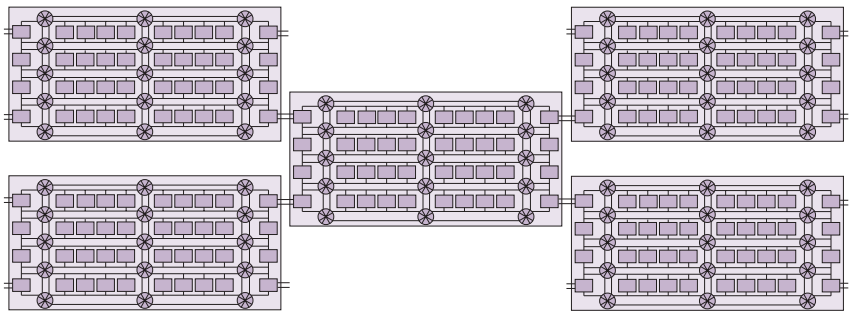


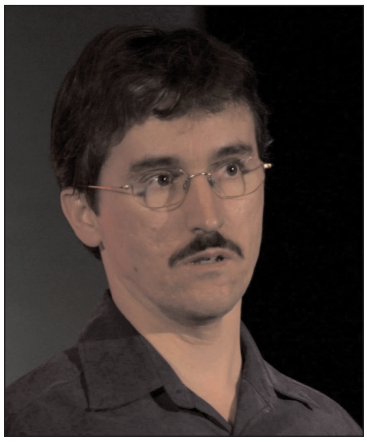
Figure 3. Multiple picoArrays can be chained together on a chip or across multiple chips. They function as one logical array, so programmers can configure them to carry out numerous tasks in parallel. PicoChip has designed a 16-chip development board with 6,880 array processors that can execute 4.4 trillion instructions per second or 384 giga-MACs per second (GMACS).

compute-bound than I/O-bound. A wireless base station might need to sample each datastream 31 million times per second (including oversampling), and the PC101 supports 80 million samples per second per I/O port. To provide more I/O bandwidth, the PC101 can use multiple ports in parallel.

Therefore, extending the previous example, it's possible to once again double the number of array processors assigned to an FFT (to a total of 112) and split them between two I/O ports, which would boost throughput to 156,000 blocks per second. With 224 processors assigned to four I/O ports, the PC101 could process 312,000 blocks per second. Such linearity is rare in parallel processing.

Special Tools and Some VHDL Required

Software development on the PC101 isn't radically different than it is with a general-purpose microprocessor or DSP, but there are some important differences. Programmers can write much of their application code with picoChip's ANSI C compiler and use picoChip's assembler for time-critical signal-processing code. Although the PC101's array processors have proprietary instruction sets, they are comparable to conventional DSP instruction sets, and the four different types of array processors share about 90% of the same instructions. PicoChip also provides a source-level debugger and a bit- and cycle-accurate simulator.



ROSS MEHAN

Peter Claydon, picoChip's chief operating officer and chief architect of the picoArray architecture, describes the PC101 at EPF2003.

The main difference in programming the PC101 is that software developers must use structural VHDL to specify the interprocessor signal flows. It's less complex than writing behavioral VHDL because there's no need for logic synthesis. (The PC101 is an off-the-shelf chip whose logic is fixed at manufacture.) Engineers who might otherwise be programming an FPGA or developing an ASIC will be comfortable with this phase of development, but it will be new to programmers with a pure software background, and it's unnecessary with general-purpose processors and DSPs. Later this year, picoChip plans to support a schematic editor that can import files from tools like Mathworks' *Simulink*.

After merging their program code with the VHDL structural description, developers use an automated tool developed by picoChip to map the code onto the picoArray. PicoChip refers to this step as "place and switch." It's not the same as the place-and-route phase of circuit layout, of course, because the PC101 has a fixed physical layout. Instead, it refers to the design-time step of mapping the application onto the picoArray for maximum run-time parallelism. The automated tool allocates the chip's signal paths and other resources to guarantee deterministic performance.

To ease software development, picoChip offers prewritten libraries for WCDMA (wideband code-division multiple access) cellular telephony in FDD (frequency-division duplexing) mode. These libraries comply with the 3GPP (3rd Generation Partnership Project) 4.2.0 specification. The company is currently developing additional libraries for WCDMA-FDD Release 5, WCDMA-TDD (time-division duplexing), CDMA2000, TD-SCDMA (time-division synchronous code-division multiple access), and 802.20.

Scrambling For Sockets

Today's base stations for cellular and wireless-data networks rely heavily on general-purpose embedded processors, DSPs, and ASICs. The dominant incumbent is Texas Instruments—by some estimates, TI chips occupy 80% of the available sockets. Everyone realizes that the transitions to 2.5G, 3G, 802.11g, and other new standards represent a golden opportunity to shuffle the deck. That's undoubtedly why we're seeing a proliferation of new DSPs, SoCs, ASSPs, and innovative architectures like the picoChip PC101, Elixent D-Fabrix, and Motorola MRC6011. Vendors are scrambling to differentiate themselves and to offer solutions that promise higher performance while driving down system costs.

High performance is a strong point of the PC101. Despite its rather sluggish clock frequency, it can execute 24,800 16-bit MMACS (million MACs per second). That's 10 times as many 16-bit MMACS as TI's fastest TMS320C64x DSPs, which run at the much higher clock rate of 720MHz. One PC101 could replace multiple DSPs in a base station and still have enough capacity to handle additional tasks.

Another popular measure of raw performance is integer operations per second. Each array processor in the PC101 can execute two 16-bit operations per clock cycle, so the grand total is 137.6 GOPS (2 operations × 430 processors × 160MHz = 137.6 giga-ops). That's without counting a MAC as two operations (multiply and add). Of course, numbers like this are an unrealistic expression of peak performance that would never be sustained in a real-world application, but it does provide another data point in the absence of formal benchmarks.

Although fixed-function ASICs also deliver high performance, they sacrifice flexibility after the finished silicon arrives from the foundry. A programmable processor like the PC101 adapts more easily to evolving industry standards and product requirements. With the PC101, it's possible to design a base station that's upgradable in the field. In addition, the PC101 will be available as a standard part, whereas a custom ASIC requires an expensive and risky development project.

Rematch: Brainiacs vs. Speed Demons

Parallel architectures like picoChip's PC101, Elixent's D-Fabrix, Motorola's MRC6011, PACT's XPU128, and the

CMU/STMicroelectronics PipeRench take a fundamentally different approach than do conventional architectures that rely more heavily on speed. The foremost example of the latter is Intrinsity's FastMath, winner of the *MPR* Analysts' Choice Award for Best Extreme Processor. (See ***MPR 2/18/03-05***, "Extremely High Performance.") Based on a MIPS32 processor core, the FastMath screams at 2.0GHz, even though it's fabricated in a TSMC 0.13-micron CMOS process almost identical to the one that gets the PC101 to only 160MHz.

To borrow a concept from the past days of *MPR*, communications processors are staging a rematch of the brainiacs versus the speed demons. The outcome of that contest among PC, workstation, and server processors has been mixed. Although Intel has refitted the x86 with a superpipeline to create a 3.2GHz Pentium 4 speed demon, the same company has sired a 410-million-transistor, 1.5GHz Itanium 2 brainiac. A similar dichotomy is now evident among communications processors.

Despite the impressive parallelism shown by picoChip, Elixent, and others, the speed demons make a strong case for a traditional execution model. Intrinsity says a single FastMath processor can support 128 to 200 user datastreams in a CDMA2000 or WCDMA cellular base station, with an external FPGA or ASIC to handle the rake receiver and multipath searcher functions. A future FastMath chip will integrate those functions to deliver single-chip baseband processing. Ultimately, Intrinsity's goal is to slash base-station costs to less than $1 per channel.

What a FastMath processor lacks in parallelism, it can make up for with brute force. In addition, software development is more straightforward and requires no special tools. It's the same lesson learned from the previous matchup of brainiacs and speed demons—there's more than one way to skin an algorithm.

Price & Availability

PicoChip has development kits with samples of the PC101 available now. Volume production is scheduled for 1Q04. PicoChip hasn't announced chip prices but says PC101 pricing will be comparable to that of a high-end DSP. For more information, see *www.picochip.com*.

Direct comparisons between picoChip's PC101 and competing chips will have to wait until picoChip announces prices, power-consumption specifications, and a firm delivery date for production quantities. A chip as complex as the PC101 will almost certainly be larger and more expensive than a conventional microprocessor or DSP, though perhaps more economical if it replaces multiple processors and ASICs. Power consumption will vary widely, depending on workload, but picoChip estimates 3–4W will be typical.

The PC101's "self-healing" ability to bypass some array processors ruined during fabrication should reduce manufacturing costs, because picoChip won't have to discard every die that has a few spot defects. However, the company may have to accept a somewhat lower price for chips that don't make the top bin-sort.

Perhaps the biggest challenge for a small startup company like picoChip lies in standing out from the crowd. The onrush of communications processors for cellular and wireless-data networks is reminiscent of the flood of network processors for switches and routers a few years ago. When dozens of vendors vie for the attention of relatively few customers, getting a foot in the door can be the biggest step of all. ◇