Military electronic equipment manufacturers at one time were categorized into two fairly distinct design types: hardware based and software based. Each offered significant advantages and disadvantages in speed and flexibility. Never, more flexible war-fighter support systems today take advantage of both hardware and software using hardware acceleration with FPGAs.

Hardware acceleration with FPGAs is important in military sensor applications for three reasons. First is to reduce system latency so that defensive systems can react faster to enemy threats, such as jamming or blinding. Second, as military threats have become more elusive and blend in with urban settings, systems are constantly under pressure to increase sensor resolution. And lastly, as military threats become more tactical, the Size, Weight, and Power (SWaP) of a system must be reduced.

Given these technology drivers, the military market for high-performance processing is expanding rapidly, though there are currently few application-specific (tailored application) coprocessing FPGA solutions available.

With the proliferation of distributed and multicore processing computers, complex signal processing applications can be accelerated with distributed functions. The processing cores, however, are still limited by cache size and coherency, memory bandwidth, and in some cases, power and cooling.

One approach to improving processing power and efficiency in single-core and multicore processors is to use FPGAs as application-specific coprocessors. These devices can be designed to fit into Intel Xeon and Advanced Micro Devices (AMD) Opteron sockets to replace one of the processors with pipelined FPGA logic. Alternately, FPGA “drop-in” modules can be added to processor memory subsystems to offload specific hardware pipeline functions. The advantages of these approaches have been explored by Altera Corporation and applied in some cases by FPGA users. Many significant advantages to sensor signal processing problems can be examined with three sample layouts: two example architecture sets for multicore processing and one architecture for single-core processing.

### Potential algorithmic returns

Hardware acceleration using an FPGA as a coprocessor is intended to offer 10x to 100x speed improvement for tailored algorithms, and anywhere from 3x to 50x speed improvement for the user application. These target numbers are based on commercial applications such as financial analysis, data warehousing, and biosciences.

<table>
<thead>
<tr>
<th>Application</th>
<th>Processor Only</th>
<th>FPGA Coprocessing</th>
<th>Speed Up</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Hough and inverse Hough processing</td>
<td>12 minutes processing time Pentium 4-3 GHz</td>
<td>2 seconds of processing time @ 20 MHz</td>
<td>370x faster</td>
</tr>
<tr>
<td>(2) Spatial Statistics (Two Point Angular Correlation Cosmology)</td>
<td>3,397 CPU Hours with 2.8 GHz Pentium (Approximate Solution)</td>
<td>36 hours (exact solution)</td>
<td>96x faster</td>
</tr>
<tr>
<td>(2) Black-Scholes (Financial Application (single precision floating point 2M points))</td>
<td>2.3M experiments/sec with a 2.8 GHz processor</td>
<td>299M experiments/sec</td>
<td>130x faster</td>
</tr>
<tr>
<td>(1) Smith Waterman ssearch34 from FASTA</td>
<td>6,461 sec processing time (Opteron)</td>
<td>100 sec FPGA processing</td>
<td>64x faster</td>
</tr>
<tr>
<td>(3) Prewitt Edge Detection (compute-intensive video and image processing)</td>
<td>327M Ciks (1 GHz processing power)</td>
<td>131K Ciks @ .33 MHz</td>
<td>83x faster</td>
</tr>
<tr>
<td>(1) Monte Carlo Radiative Heat Transfer</td>
<td>60 ns processing time (3 GHz processor)</td>
<td>6.12 ns of processing time</td>
<td>10x faster</td>
</tr>
<tr>
<td>(1) BJM Financial Analysis (5M paths)</td>
<td>6,300 sec processing time (Pentium 4-1.5 GHz)</td>
<td>242 sec of processing @ 61MHz FPGA</td>
<td>26x faster</td>
</tr>
</tbody>
</table>
Several selected algorithms taken from these commercial applications have been implemented and tested on FPGA coprocessors for evaluation. These are shown in Table 1.

In addition to the performance advantages of hardware acceleration, replacement of either an Intel or AMD processor significantly reduces power consumption and heating in a system. This is a result of the streamlining of software and hardware functions where they are best suited. Initial benchmarks on the expected power savings are algorithm dependent.

**Single-core coprocessing architecture**

The majority of military systems utilize single-core CPU architectures. Migrating to higher-performance processing systems, while still meeting government software code reuse objectives, is a problem that can be solved by offloading software functions into FPGA hardware.

Some signal processing problems are fairly simple, involving a single sensor with a single data processor. In order to optimize the data flow and computational bandwidth in this system, the coprocessing problem is very straightforward. Large functions that can be mathematically isolated and pipelined are performed outside a processor using a systems analysis process.

One such process utilizes a hardware and software architecture called IMPLICIT+EXPLICIT. It can be used with single or multiple microprocessor boards. This architecture combines an FPGA-based reconfigurable processor, MAP, in a peer relationship with a microprocessor. This peer relationship is achieved by connecting programmable logic to the microprocessor using the microprocessor’s memory DIMM slots. FPGAs are used to implement pipelined direct execution logic to perform functions such as DSP logic. This can accelerate sensor processing, allowing for less power consumption and greater sensor resolution.

Another processing challenge in sensor systems is moderating the information between sensor and microprocessor with minimal latency. This is performed with user logic chips diagrammed (Figure 1) in generic blocks. The controller is an FPGA that performs several functions including virtual to physical address conversion and DMA packet generation. The SDRAMs shown are part of a 64-bit global shared address space that includes the microprocessor memory. The direct execution logic that implements the user’s program resides in the two user logic FPGAs. These can perform up to sixteen 64-bit references per clock cycle to the 8 SRAM banks. In addition to the connection to the DIMM bus, data may also be received or sent using the GPIO port. This port is very useful for delivering sensor data directly to the user logic chips.

The division of the user’s program between the microprocessor and programmable logic is accomplished using the Carte high-level language programming environment. This allows programmers to use ANSI C and FORTRAN to generate a single executable program that will control both FPGAs and the microprocessor. This allows the division of logic between hardware and software to be performed by a single architect.

Performance gains of this coprocessing approach have been demonstrated to several government customers, resulting in orders of magnitude improvement over a stand-alone 2.8 GHz Xeon processor. These applications include military imaging, radar signal processing, and image target recognition. Benchmark results for the complete application including data movement time are shown in Figure 2.

Some military applications may be more advanced, utilizing multiple sensors, sensors in multiple modes, or rapid system reconfiguration. For these systems where the user is willing to pursue a more difficult segmentation of logic in pursuit of higher performance, multicore processing is one of the latest trends in high-performance computing.
Multicore coprocessing architecture

Multicore processing is utilizing multiple CPUs to execute code in parallel on the same algorithm, but separate process threads. Multicore processing requirements are being significantly pushed by military applications in imaging and data processing where performance gains in single-core processors are becoming marginal. Providers of both Intel and AMD-based architectures are being asked for their solutions.

Accordingly, the goal of multicore coprocessing is the same as single core: to identify and isolate math functions that can be efficiently offloaded from software into hardware in order to accelerate sensor system performance. A set of notional coprocessing architectures for the Intel Xeon Quad Core (Figure 3) and AMD Opteron (Figure 4) layouts are shown. Figure 5 shows a representative system developed by XtremeData that utilizes an FPGA on an XDI module. In all three layouts, FPGAs are placed in CPU sockets and require application hardware to best utilize the FPGA logic.

In the Xeon architecture, a processor and a coprocessor are connected using the Intel Front Side Bus (FSB) architecture. A Northbridge (information available from Intel) is used to connect each FSB to one another. These are the normal pathways used by multicore CPU instructions. Access to front side bus interface standards is becoming more widely available to developers.

The AMD Opteron architecture uses direct HyperTransport interconnects between each processor/coprocessor socket. This 32-bit packetized data connection is similar to 2.5 V LVDS and can easily be integrated using available HyperTransport FPGA cores.

The Xtreme Data architecture is an implementation currently in use in the several nonmilitary markets. It utilizes a multiprocessor motherboard with an AMD and Intel socket-compatible module interface for the FPGA and other components. This multisocket architecture allows the application to take advantage of the processing power available in the FPGA. This coprocessor solution offloads high-resolution DSP algorithms in military imaging systems and performs Black-Scholes simulations for financial modeling.

The motherboard plugs into a standard commercial enterprise rack or blade server. It utilizes a low-latency, high-bandwidth HyperTransport interface to the other processor. The FPGA uses all resources intended for a CPU – power supply, heat sinks, HyperTransport links – and is programmed with on-chip memory controllers.

Implementation

In addition to the FPGA design and drop-in module, an application engineer has the task of modifying the application software to exercise the FPGA coprocessor for tailored hardware operations. There are several approaches to this process, which can be logically broken out into two steps.

First, a systems engineer needs to determine which functions in the system will be offloaded into the FPGA for accelerated processing. This can be accomplished by using traditional software profiling tools.
Second, the application engineer must identify the function calls and interfaces for coprocessing, as well as the task distribution among the remaining processors. In the case of the Carte Programming Environment, users have equated this effort as comparable to the effort required to convert a uniprocessor application into an MPI application.

As multicore processing becomes more prevalent, software tools will become available to help designers parallelize and compile their systems for N hardware nodes from other vendors. Function calls can be translated from C to efficient HDL using proprietary tools offered by FPGA providers. This application tailoring requires only basic FPGA knowledge, allowing the system designer to build algorithm architectures confidently without a full FPGA development staff.

Simplifying coprocessing
Using design software developed by FPGA providers and hardware partners, hardware acceleration is evolving from a specialized engineering effort into a powerful systems design tool. It is accessible to software designers today, and will only become more accessible in the future.

The next generation of embedded computation for military applications is likely to involve strong design partnerships between providers of software and pipelined hardware. More of these will be seen in the future as holistic development tools allowing simple partitioning of applications into the processor domain that better suits their execution.

As computing architectures become optimized for military applications, code portability and supportability may suffer. Engineers will then have a new challenge: balancing tailored performance versus code reuse.

J. Ryan Kenny is technical marketing manager in Altera’s military and aerospace business unit. He is responsible for creating FPGA-based technical solutions for the military data and signal processing market. He joined Altera in March 2007 and has more than 10 years of experience in space and defense electronics in the U.S. Air Force and at Lockheed Martin. He graduated from the U.S. Air Force Academy and completed an MSEE and MBA from California State University Northridge and Santa Clara University respectively. He can be contacted at rkenney@altera.com.

Bryce Mackin is the strategic marketing manager in Altera’s computer and storage business unit. His focus is on FPGA coprocessors for the high-performance computing market. Bryce has been in the computer and storage market for more than 10 years. Prior to joining Altera, he served as a strategic marketing manager for Xilinx, worked at Adaptec in several product marketing roles, and held the position of marketing chairman for the Storage Networking Industry Association (SNIA) IP Storage Forum. Bryce has a degree in Industrial Engineering from California State University in San Jose. He can be contacted at bmackin@altera.com.