

Accelerating PowerPC Software Applications

Using custom APU peripherals, C-to-hardware tools enable fast creation of Virtex-4 hardware accelerators.

David Pellerin
Chief Technology Officer
Impulse Accelerated Technologies, Inc.
david.pellerin@impulsec.com

Greg Edvenson
Senior Software Engineer
Pico Computing, Inc.
greg@picocomputing.com

Kunal Shenoy
Design Engineer
Xilinx, Inc.
kunal.shenoy@xilinx.com

Dan Isaacs
Director, Embedded PowerPC Marketing, APD
Xilinx, Inc.
dan.isaacs@xilinx.com

The Xilinx® Virtex™-4 FX family of FPGA devices provides embedded systems developers with new alternatives for creating high-performance, hardware-accelerated applications. With an integrated industry-standard PowerPC™ processor and innovative Auxiliary Processor Unit (APU) interface, the Virtex-4 FX device allows system designers to efficiently connect custom hardware accelerators to the integrated processor, yielding unprecedented performance.

In the past, software programmers who wanted to take advantage of FPGAs for algorithm acceleration have experienced significant technical barriers because of the complexity of writing low-level hardware descriptions to represent higher level software functions.

In this article, we'll show how the power of the Virtex-4 FX FPGA can be made readily available to embedded systems designers and software programmers through the use of software-to-hardware tools. The emergence of such tools bring the performance benefits of FPGAs to anyone who can program in C. Accelerated FPGA-based designs are now easier and more practical for a wide range of application domains, including image processing, DSP, and data encryption.

From Software to FPGA Hardware

By virtue of their massively parallel structures, FPGAs have the potential to dramatically accelerate embedded software applications. But because these devices require different (hardware-oriented) skills than traditional processors, the creation and programming of a system based on an FPGA has remained challenging for all but the most hardware-savvy software programmers. This is changing, however. With the introduction of simplified FPGA-based computing platforms and streamlined tools for platform building, most of the barriers to FPGA adoption have been removed. In addition, the introduction of software-to-hardware tools for FPGAs has dramatically improved the practicality of these devices as software-programmable computing platforms.

The tools that make this shift possible – enabling FPGA-based platforms to be considered viable alternatives to traditional processors for embedded systems – serve two basic needs. At the front end, software-to-hardware compiler tools accept high-level descriptions written in a language familiar to embedded software programmers. The de facto standard for embedded systems design is standard C, with C++ and Java beginning to make inroads as well.

At the back end, existing synthesis and place and route technologies are combined with system-level platform building tools, allowing designers to develop and target complete systems on programmable logic to specific development boards. Both of these needs are being met today, by tools currently available.

In the area of software-to-hardware compilation, compiler tools such as Impulse

CoDeveloper (Figure 1) can simplify the generation of FPGA hardware from higher level C-language descriptions of software algorithms. These tools provide the necessary bridge between the domains of software programming and lower level hardware design.

Serving the need for platform building tools is Xilinx Platform Studio, which supports a wide variety of Xilinx FPGA-based boards and systems. Platform Studio makes it practical for an embedded systems designer – who may have little or no expe-

rience with low-level FPGA design – to assemble a complete system within a single FPGA, including one or more processors and related peripherals. When these tools are combined with a platform-aware software-to-hardware compiler, the complete system can include custom accelerators originally written in C.

Accelerating Embedded Applications

The Virtex-4 FX family of devices provides an ideal platform for hardware acceleration of embedded applications. The Virtex-4 FX12 FPGA, for example, includes more than 12,000 logic cells; an integrated PowerPC 405 core, which can operate at speeds as fast as 450 MHz; and dual 10/100/1000 Ethernet MACs, configured by the processor through the device control register (DCR) interface or through the FPGA fabric.

Looking inside the embedded processor block (see Figure 2), the PowerPC 405 CPU is directly coupled to the unique and innovative APU controller, which provides direct access to hardware accelerators implemented in the FPGA logic. The APU controller supports three classes of instructions: PowerPC floating-point instructions, APU load and store instructions, and user-defined instructions (UDI). UDIs are program-

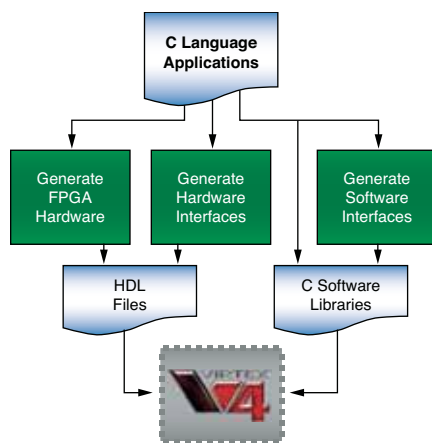


Figure 1 – Impulse CoDeveloper tools simplify the conversion of C subroutines to lower level FPGA logic and provide the necessary software-to-hardware communications on the Virtex-4 platform.

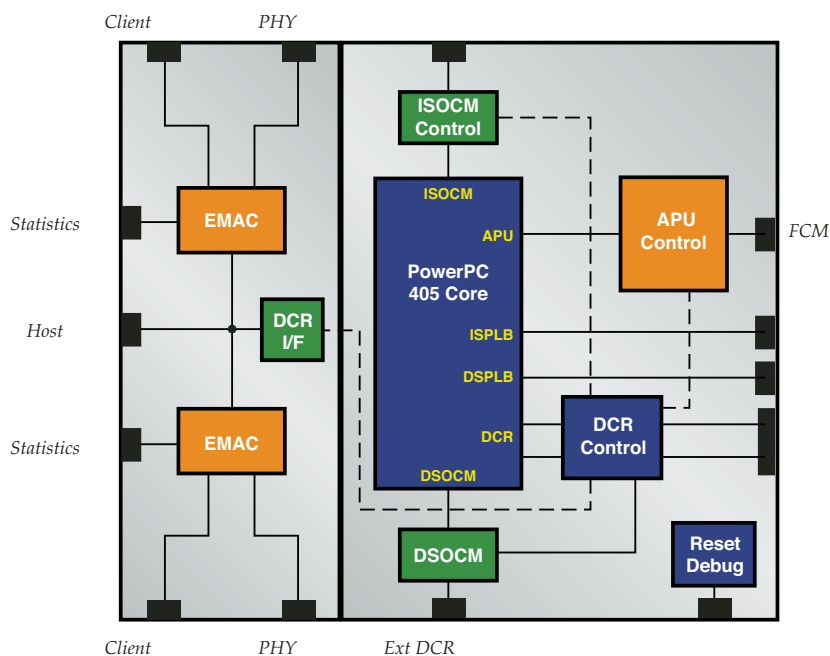


Figure 2 – The Virtex-4 FX12 embedded processing block includes the embedded PowerPC 405 processor and a high-performance APU interface, along with dual 10/100/1000 EMACs.

grammed into the APU controller either dynamically through the PowerPC 405 via the DCR bus or statically during FPGA configuration via the bitstream. The APU supported instructions are executed by hardware acceleration co-processing engines implemented in the FPGA logic.

When packaged in a highly integrated

compact device such as the Pico Computing E-12 card (see sidebar, “A Wide Range of Development Platforms”), the FX12 device becomes a complete embedded development platform that requires little or no hardware design expertise. For embedded application developers requiring a wider range of hardware

peripherals (such as direct access to video and audio signals), the Xilinx ML403 board, also based on the FX12 device, provides an excellent embedded systems development platform.

On-chip, the Virtex-4 FX APU controller provides a flexible high-bandwidth interface between the FPGA fabric and the

Taking Advantage of Parallelism in FPGAs

A key aspect of any software-to-hardware design flow is the use of parallelism to increase performance. When accelerating C applications using FPGAs, parallelism can be exploited at two distinct levels: at the application system level and at the level of statements (or blocks of statements) within a specific subroutine or loop.

Although there are ongoing attempts to create compiler technologies that can exploit both levels of parallelism with a high degree of automation, the best approach today is to focus automation efforts (represented by the software-to-hardware compiler) on the lower level aspects of the problem, while at the same time providing software programmers an appropriate and easy-to-use programming model that allows higher level, coarse-grained parallelism to be expressed. In this way programmers can make hardware/software partitioning decisions and experiment with alternative algorithmic approaches, leaving the task of low-level optimization to automated compiler tools. This approach is particularly useful for platforms such as the Virtex-4 device that include embedded processors.

A number of programming models can be applied to FPGA-based programmable platforms, but the most successful of these models share a common attribute: they support modularity and parallelism through a dataflow-like method of design partitioning and abstraction. Communicating sequential processes, or CSP, is one such programming model. CSP has proven to be highly effective in expressing application-level parallelism for FPGA targets. This programming model is directly supported in the Impulse C tools provided by Impulse Accelerated Technologies, Inc.

At the heart of the Impulse C programming model are processes and streams (Figure 4). Processes are independently synchronized, concurrently operating portions of an application that are written in a standard language (in this case C language). Processes perform the work of the application by accepting data, performing computations, and generating relevant outputs.

Unlike traditional C subroutines, processes are considered persistent; they are normally called once (whether in hardware or software) and continue as long as there is streaming data to be processed. The data processed by such an application flows from process to process by means of streams, or in some cases by means of messages or shared memories, which are also supported in the programming model. Streams represent one-way channels of communication between concurrent processes and are self-synchronizing with respect to the processes by virtue of buffering. The primary method of synchronization between processes is therefore the data being passed on the streams.

The key to allocating processing power within such a system – and using such a programming model – is to implement one or more processes in the FPGA to handle the heavy computation, and implement other processes on embedded or external microprocessors to handle file I/O, memory management, system setup, and other non-performance-critical tasks. Using tools such as those included with Impulse C, an application comprising multiple parallel C processes can be modeled entirely in software, verified using a standard desktop C debugging environment, and then, after the application is functionally complete, incrementally moved into the FPGA for further optimization and acceleration.

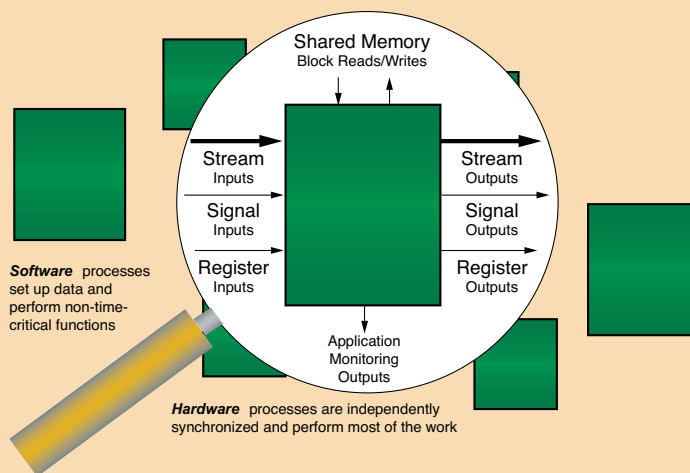


Figure 4 – The Impulse C programming model emphasizes the use of processes, streams, and shared memories for hardware/software partitioning.

pipeline of the on-chip PowerPC. Fabric co-processor modules (FCMs) implemented in the FPGA fabric are connected to the embedded PowerPC processor through the APU interface, allowing the creation of custom hardware accelerators. These hardware accelerators operate as extensions to the PowerPC, thereby offloading the CPU from demanding computational tasks.

Software engineers can access the FCM from within assembler or C code. Assembler mnemonics are available for user-defined instructions and pre-defined load/store instructions, enabling programmers to invoke hardware-accelerated functions into the regular program flow. Programmers can also define custom instructions designed specifically for the hardware functionality of the FCM. When combined with C-to-hardware compiler tools, the APU controller allows software programmers to create hardware-accelerated software applications with little or no FPGA design expertise.

C-to-Hardware Tools Increase Design Productivity

To make productive use of any computing platform, software programmers need appropriate compiler and debugging tools. Impulse C, from Impulse Accelerated Technologies, gives software programmers access to FPGAs by allowing hardware accelerators to be compiled directly from software descriptions.

These accelerators, which are typically represented by one or more software subroutines, are automatically compiled into efficient, high-performance hardware that can be mapped directly into FPGA gates. In the case of the Virtex-4 FPGA, Impulse C is also capable of automatically generating software/hardware interfaces using the APU. This is particularly useful for applications that combine both traditional and FPGA-based processing.

Because it is based on standard C, Impulse C allows FPGA algorithms to be developed and debugged using popular C and C++ development environments, including Microsoft Visual Studio and

GCC-based tools. The CoDeveloper software-to-hardware compiler translates specific C-language subroutines to low-level FPGA-hardware (see Figure 3) while optimizing the generated logic and identifying opportunities for parallelism. The compiler is also capable of unrolling loops and generating loop pipelines to exploit the extreme levels of parallelism possible in an FPGA. Instrumentation and monitoring functions generate debugging visualizations for highly parallel multi-process applications, helping system designers identify dataflow bottlenecks and other areas for acceleration.

For applications involving the embedded PowerPC and MicroBlaze™ processors, the Impulse C compiler automates the creation of hardware/software interfaces and generates outputs compatible with Xilinx Platform Studio. This makes it possible to create high-performance, mixed hardware/software applications for FPGA-based platforms without the need to write low-level VHDL or Verilog.

For large applications comprising multiple hardware and software elements, Impulse C includes interface libraries (see sidebar, “Taking Advantage of Parallelism in FPGAs”) and related compiler features, allowing parallelism to be expressed at the

design process is highly iterative, reflecting the fact that decisions made up-front (such as C coding styles and system-level partitioning decisions) may have a dramatic impact on the results obtained after C compilation, synthesis, place and route, and final bitmap generation. At each point in the process, the tools provide feedback, allowing you to evaluate and estimate performance before moving to subsequent (and perhaps more time-consuming) phases of the platform generation process.

Let’s summarize the steps required for a typical PowerPC-based application using Impulse and Xilinx tools:

1. The application is initially written in standard C, using common C development tools. These tools include readily available tools such as Visual Studio, Eclipse, or GCC and GDB, and may also involve more comprehensive cross-development tools. During this phase, a baseline for validation (a software test bench, also written in C) is established, allowing you to quickly test later design iterations.
2. A C profiler such as gprof may be invoked, or other, less sophisticated methods are used to identify computational hotspots. Often these hotspots can be isolated to a few C subroutines or inner code loops requiring acceleration. Application monitoring (made possible by instrumenting the C code during software testing) can help characterize these hotspots and analyze data movement.
3. Using software-to-hardware interface functions provided in the Impulse C library, data streams or shared memories create abstract connections between the main algorithm running on the PowerPC and hardware-accelerated subroutines running in the FPGA. The modified software algorithm, which now includes one or more independently synchronized processes, is simulated again in a standard C environment to ensure its correct behavior.

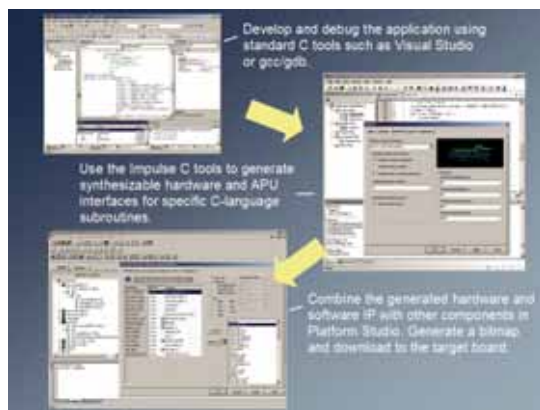


Figure 3 – The C-language-to-FPGA-accelerator design flow

level of multiple and independently synchronized processes. These processes can be mapped either to software running on an embedded PowerPC or MicroBlaze processor, or to FPGA hardware.

For all such applications, integration of front-end compiler tools and back-end platform building tools is important. The

4. C-language subroutines (or processes, as they may now be referred to) representing hardware accelerators are analyzed and optimized by the Impulse C compiler, resulting in hardware description files compatible with FPGA synthesis tools. Optimization reports generated in this phase help you understand the impact of various coding styles, and make appropriate revisions in the original C code for improved performance. During this compilation process, additional compiler outputs are generated that represent hardware-to-software interfaces, including (in the case of the Virtex-4 FPGA) the necessary APU interface logic. Software run-time libraries are also generated at this point, corresponding to the abstract stream and shared memory interfaces specified on the processor side of the application.

5. The generated hardware and software files are exported from the Impulse tools (as a PCORE peripheral) and imported directly into the Xilinx Platform Studio environment.

The stream and shared memory interfaces defined in the C application are mapped to APU, PLB, or other interfaces where appropriate, along with other components (such as standard processor peripherals or non-standard IP blocks) to create the complete system. From within the Platform Studio interface, the entire application (both hardware and software) is built, resulting in a downloadable bitmap.

Evaluating FPGA Acceleration

Using the Pico E-12 card and the Xilinx ML403 development kit, in conjunction with Impulse C and Platform Studio, we set out to compare the relative performance of the embedded PowerPC 405 processor both with and without APU hardware acceleration – and using only C programming techniques. To investigate a range of potential application domains, we selected the following three representative algorithms:

1. An image filter. This algorithm allowed us to evaluate two pipelined hardware

routines for processing a stream of image data. The algorithm chosen for this experiment is a relatively simple 3 x 3 edge-detection function operating on a 512 x 512 image buffer. This algorithm allowed us to quickly evaluate the performance of data streaming through the Virtex-4 APU interface, as well as the potential speedups of using multiple, pipelined hardware processes.

2. A triple-DES encryption engine. This algorithm allowed us to evaluate the impact of various C-level optimization strategies, as well as the practicality of adapting and optimizing legacy C code for a streaming programming model. One million character blocks (of eight characters each) were processed to obtain performance numbers for this test.

3. A fractal image generator. This algorithm is computationally intensive and can be characterized in many ways to explore the size/performance space. For this experiment, we created a single hardware process in the FPGA as an APU peripheral. This hardware process communicates with a single controlling software process running on the embedded PowerPC. The design of this algorithm, which generates a 1024 x 768 pixel image with a selectable level of image accuracy, is scalable such that additional hardware accelerator processes can be easily added, up to the limit of the target FPGA.

For each of these algorithms, various combinations of compiler loop unrolling, pipelining, and maximum stage delays

were selected in the Impulse C compiler. In this way the applications could be optimized (in most cases without modifying the original C code) to obtain a desirable balance of size, cycle delays, and maximum clock speed in the generated hardware. In most cases, we determined that using a relatively low clock rate (50 MHz) in the FPGA fabric – in combination with increased cycle-by-cycle throughput (through the use of automated pipelining) – produced the best overall results given the nominal overhead of software-to-hardware data communication.

Using these algorithms as a baseline, numerous tests were performed in which the same C code was compiled both to the FPGA (as an APU accelerator) and to the embedded PowerPC processor as a software-only application. The results of these tests are summarized in Table 1.

As the chart shows, the hardware-accelerated algorithms show an impressive increase in performance, even at reduced FPGA clock rates, compared to the PowerPC software-only version.

Conclusion

In this article, we have demonstrated how it is possible, using an FPGA-based platform and C-to-hardware tools, to create highly accelerated systems without low-level hardware design skills. The Virtex-4 FX device, when implemented in a card such as the Pico E-12 or on a prototyping board such as the Xilinx ML403, promises to revolutionize the way that FPGA devices are applied for high-performance embedded computing.

Software-to-hardware tools such as Impulse C, when combined with the platform building capabilities of Platform Studio, make programming for such devices practical and efficient. 🌈

Application	PowerPC Only (300 MHz)	PowerPC/APU (300/50 MHz)	Acceleration
Image Filter (512 x 512 Image)	0.1414 sec	0.0124 sec	11.4 X
Encryption (8M Characters)	2.257 sec	0.0667 sec	33.84 X
Fractal Image (10K Max Iterations)	660 sec	31 sec	21.29 X

Table 1 – Virtex-4 APU acceleration results

A Wide Range of Development Platforms

Providing embedded application developers – software programmers – with an easy-to-use hardware platform is a critical first step in making FPGAs viable as embedded development platforms. A growing number of vendors are offering FPGA-based prototyping and high-performance computing platforms ranging from low-cost, single-FPGA systems to larger FPGA grids intended for hardware-accelerated computing.



Figure 5 – The Pico Computing EP-12 card packages the FX12 or LX-25 device with a CompactFlash interface in an extremely compact form-factor.

There are two versions of the Pico E-12. The Logic Optimized (LO) version is based on the Virtex-4 LX-25 device, while the Embedded Processor (EP) version is based on the Virtex-4 FX12 device, with its integrated PowerPC processor.

In either case, the FPGA on the E-12 card is configured from the 64 MB of on-board flash memory using an on-board loader. The unique design of this loader allows new FPGA images to be swapped into the FPGA on demand. A large number of FPGA images can be stored in flash memory, and any image can be loaded at any time through on-board software or external software communicating with the E-12 card through its external interfaces. The contents of the 128 MB of external RAM remain intact through the swapping sequence, allowing subsequent FPGA images to operate on existing RAM data.

The Xilinx ML403 (shown in Figure 6), the first of several Virtex-4 FX embedded processing development boards, combines the Virtex-4 FX12 with a wide variety of software-configurable interfaces, including network interfaces; serial, parallel, and USB ports; LVDS and D/A and A/D interfaces; and a VGA driver. As such, the ML403 is ideal for embedded systems designers requiring direct FPGA access to external hardware devices.

Figure 7 shows a comparison between the APU with Impulse-generated hardware accelerators and a processor/software-only implementation. Both systems are utilizing the ML403 in this example. You can see that the APU-accelerated version is significantly faster than the processor-only version.

The Pico E-12 card mentioned in the main article (which is available from Pico Computing, www.picocomputing.com) is a CompactFlash form-factor package that draws well under 2W. It features 10/100/1000 Ethernet, 64 MB of Flash, 128 MB of RAM, and a wide host of peripheral adapters such as A/D, D/A, asynchronous serial, synchronous serial, CAN bus, relay control, and JTAG. The card is supported by platform development and programming tools appropriate for software developers. The Pico E-12 platform advances desktop and portable computing by providing massively parallel hardware computing resources in a low-power, self-contained package (Figure 5).



Figure 6 – Xilinx ML403 development system

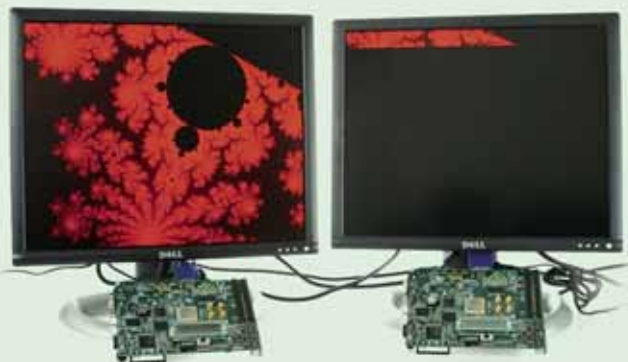


Figure 7 – Xilinx ML403 development system showing APU hardware accelerated implementation versus software only executing on the PowerPC