

# Build and Optimize a MicroBlaze Soft-Processor System Your Way

The Xilinx MicroBlaze solution makes custom processing on FPGAs easy.

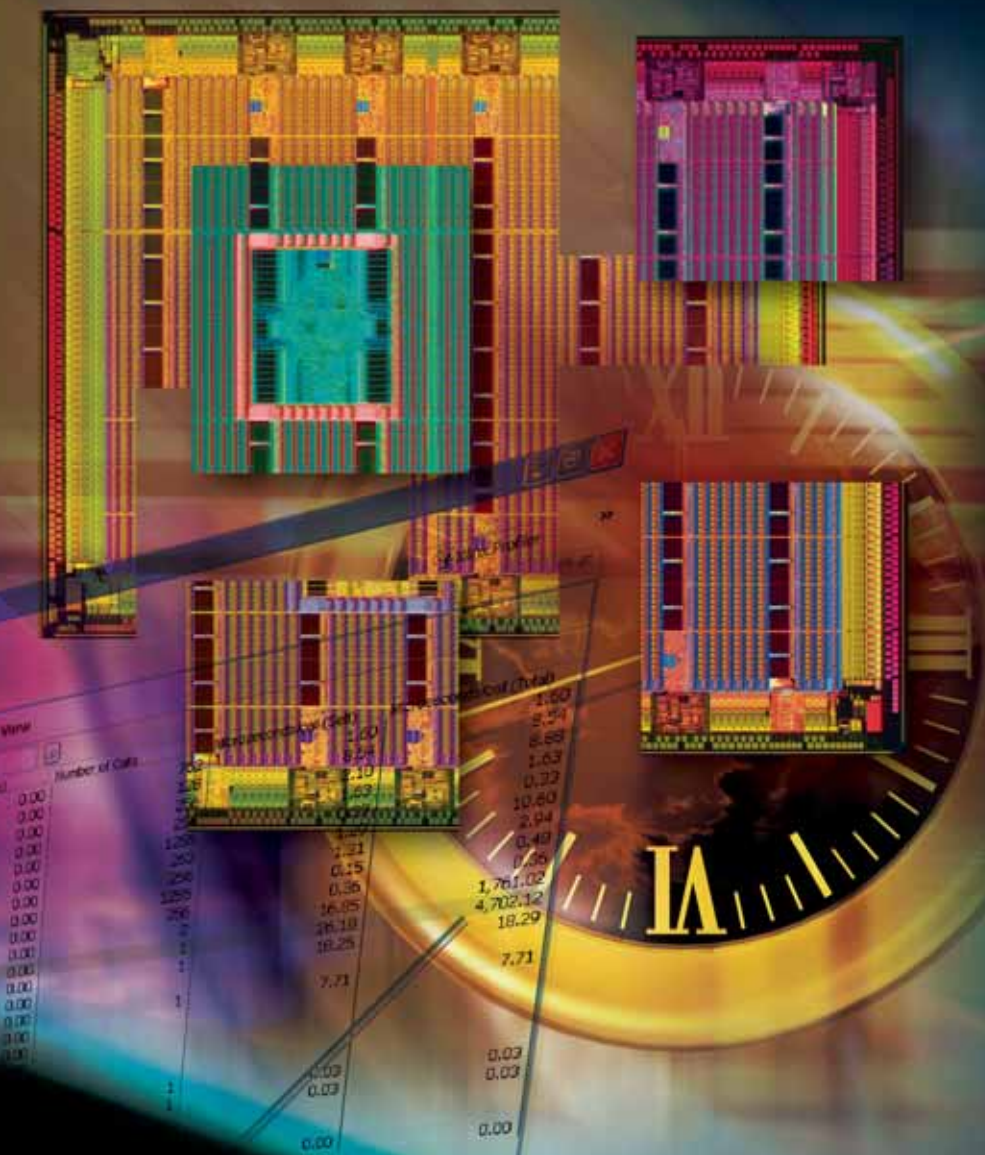
by Helen Yu  
Product Marketing Manager,  
Xilinx Embedded Processing Solutions  
Xilinx, Inc.  
[helen.yu@xilinx.com](mailto:helen.yu@xilinx.com)

Finding a processor to meet performance, feature, and cost targets is very challenging. The Xilinx® MicroBlaze™ soft-processor provides a scalable solution that is fully customizable, area-efficient, and can be optimized for your most cost-sensitive designs.

MicroBlaze v4.00, the newest version of the 32-bit soft-processor core from Xilinx, raises the bar for flexibility and ease-of-use with new user-configurable hardware options, improved debug capabilities, and complete backward compatibility with earlier releases. In this article, you will learn how to increase the performance of a software-based filter design using the MicroBlaze processor and the award-winning Xilinx Platform Studio embedded tool suite.

## Build What You Need

The Xilinx MicroBlaze processor is a RISC-based, 32-bit reconfigurable soft-processor that can be customized with different peripherals and memory configurations. Unlike a hard processor, which is implemented in the FPGA at the transistor level, a soft core is an IP block. The MicroBlaze soft processor core is optimized for our FPGA architecture. One of the many advantages of soft-core designs is that they are very configurable. In other words, you do not have to incorporate every available feature. Instead, you can disable any feature that you are not using to save valuable logic resources on your device. As a result, the MicroBlaze solution allows you to tune the processor system architecture to your application.



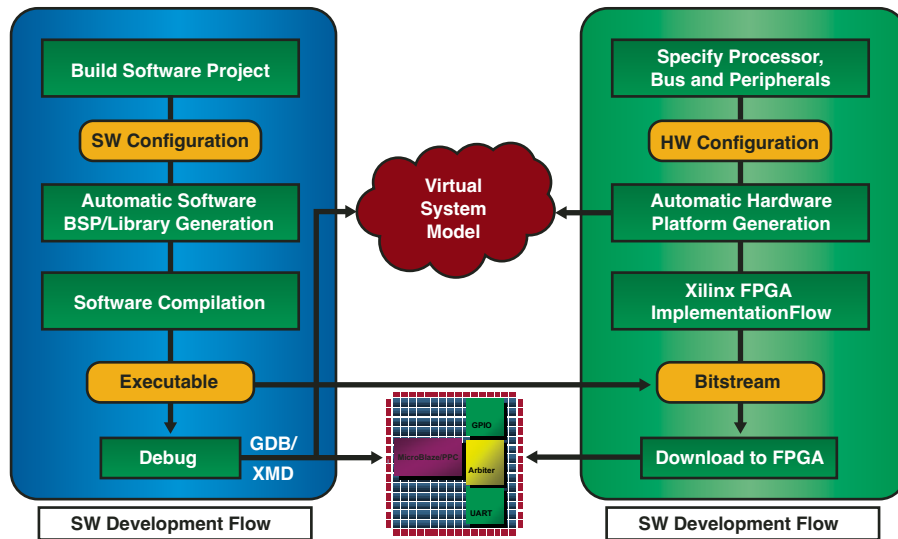


Figure 1 – The Xilinx Platform Studio (XPS) tool flow

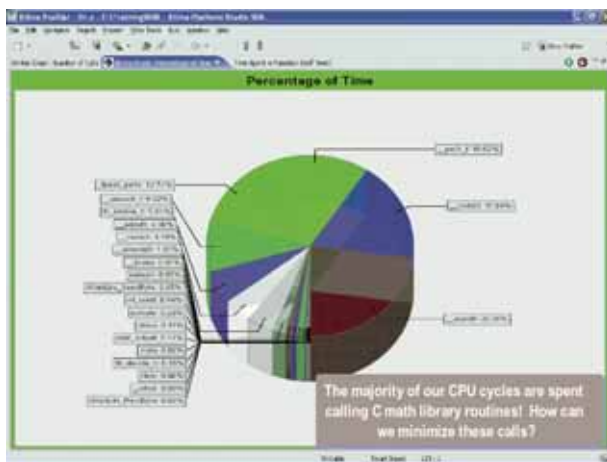


Figure 2 – Floating-point library calls occupy CPU time

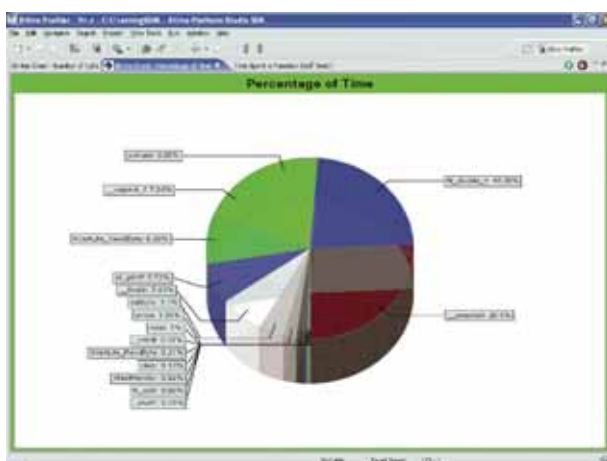


Figure 3 – Adding FPU improves performance

form software model and libraries for that system. A new feature in the XPS called the Virtual Platform Generator provides the capability to generate a virtual MicroBlaze platform model. Now you no longer have to prototype on a hardware development board at your desk or even go through the flow of developing that hardware. With Virtual Platform Generator, you can generate the software model, drivers, and libraries of your system and then compile, profile, and execute software on that model.

### The Next Step – Profile Your Software Application

By now you have specified your hardware processor system and generated the virtual model and software libraries. The next step is to start a C application project using either XPS or the Eclipse-based XPS Software Development Kit (SDK) tools. Once the software project is built and compiled, you can run the executable and then step the application through the debugger tool.

Profiling in SDK is interactive and provides graphical profile views. In our MicroBlaze design example, the core arithmetic loop function implements a FIR filter function. This small loop of code makes calls to floating-point library functions, which consume a large part of the CPU time in the FIR function call (as shown in Figure 2). How can you avoid calling floating-point libraries? Well, you could convert to a fixed-point operation, but that would be quite time-consuming. Another option is to use a processor with an FPU. The FPU would be able to do these computations natively and thus avoid library calls (as shown in Figure 3).

Conveniently, the new MicroBlaze v4.00 includes an integrated FPU option (depicted in Figure 4). The FPU handles single-precision floating-point arithmetic and is compatible with IEEE-754. Its integration ensures high performance, low latency, and a compact design. Because the MicroBlaze FPU is yet another configurable feature of the MicroBlaze core, it takes no extra space in the FPGA if it is not needed.

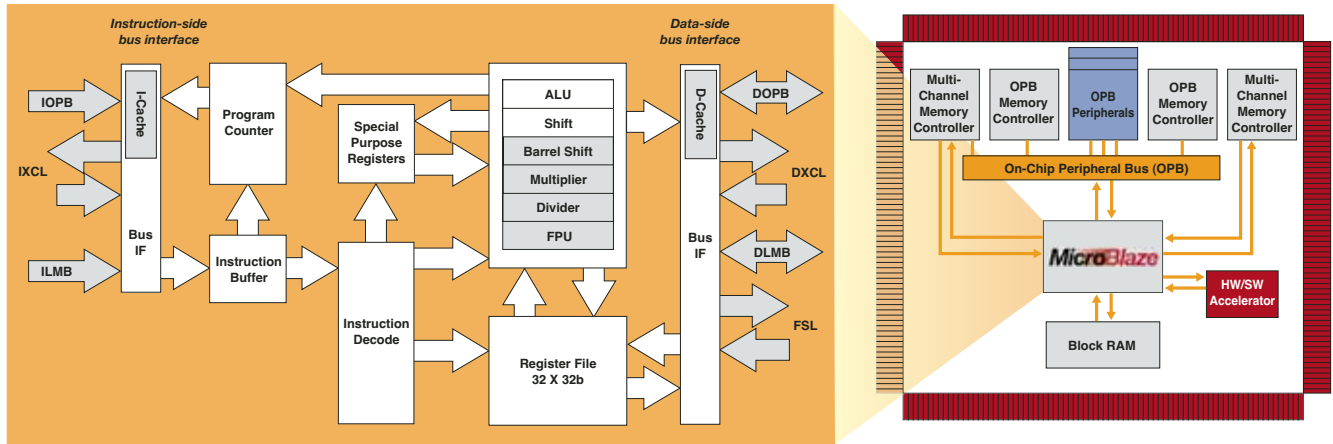
Let's look at how we might improve the performance of our FIR filter by enabling this MicroBlaze FPU.

### Getting Started with XPS

The Xilinx Platform Studio (XPS) embedded tool suite provides an integrated environment for creating the software and hardware specification flows for a MicroBlaze system, as shown in Figure 1. In this article, we'll discuss a MicroBlaze system that uses a floating-point finite impulse response (FIR) algorithm and its performance improvement after adding a floating-point unit (FPU) to the design.

The XPS base system builder (BSB) wizard helps you quickly build a working system targeted at an existing or custom development board. Based on your board selection, the BSB offers you a number of options for creating a basic system on that board, including processor type, debug interface, cache configuration, memory type and size, and peripheral selection. For each option, functional default values are pre-selected in the wizard.

Once you have defined the system architecture, you can then generate the virtual plat-



□ Basic Processor Functions    □ Configurable Functions    ■ Designer Defined Blocks    ■ Peripherals – Xilinx or 3rd Party or Designer Defined

Figure 4 – MicroBlaze v4.00 architecture

**FPU to the Rescue**

To compare performance results, you will need to profile a floating-point FIR filter application on two different configurations of the MicroBlaze processor. The flat profile views in Figures 5 and 6 display the time spent on the main function call with and without an FPU, respectively. Adding the FPU to the design reduced the number of CPU cycles required by this application from 62 milliseconds to about 4.7 milliseconds – a 13x improvement. In addition, the FIR function call is now more than 50 times faster than in the previous run without an FPU. By simply enabling the MicroBlaze FPU option, you have substantially improved the performance of the MicroBlaze floating-point FIR design.

To further enhance system performance, consider implementing the entire FIR function in hardware as an assist engine connected to a fast simplex link (FSL) interface. The FSL is a simple yet powerful point-to-point interface that connects user-developed co-processors to the MicroBlaze processor pipeline. Like the FPU option,

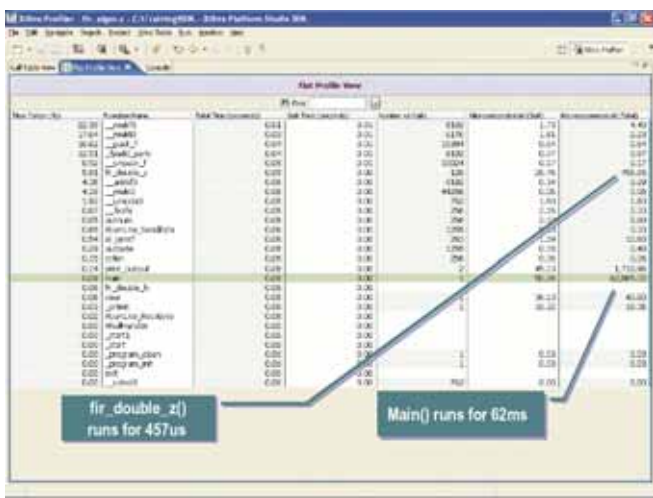


Figure 5 – Flat profile view for FIR filter design without FPU

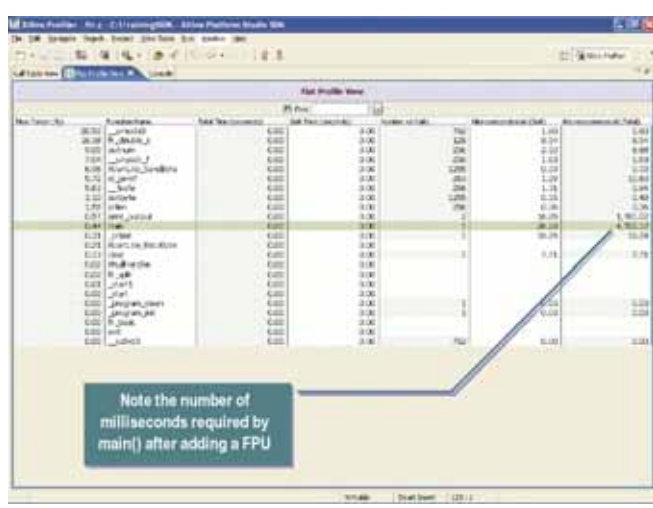


Figure 6 – Flat profile view for FIR filter design with FPU

the FSL interface is a configurable feature within the MicroBlaze processor. This high level of configurability allows you to tailor your processor system to the exact needs of the target embedded application, which provides great flexibility but does not add to the cost if these features are not used.

**Conclusion**

This sample MicroBlaze FIR filter application has demonstrated how an algorithm that is a candidate for hardware acceleration can – with minimal work – be implemented on a mixed hardware/software platform for the purpose of performance evaluation. Utilizing the Xilinx Platform Studio embedded tool suite, you can generate cycle-accurate software models and profile the performance of software running on a fully custom virtual system. You can quickly tune the performance of your processor architecture and system architecture to achieve an optimal balance of performance versus hardware resources.

For more information on the MicroBlaze soft-processor core, visit [www.xilinx.com/microblaze](http://www.xilinx.com/microblaze).