

Easy FPGA Development

If current trends continue, FPGAs may become easier to program than DSPs.



by Kenton Williston
Site Editor
DSP DesignLine
kentonwilliston@yahoo.com

The evolution of FPGAs has paralleled that of DSPs in many ways. In the

early days of DSP, programs were written entirely in assembly language. Today, DSP programmers typically start with a high-level language such as C and then optimize the “hot spots” using intrinsics or hand-coded assembly.

Similarly, FPGA designs were once crafted in Verilog or VHDL. Today, FPGA designers often start with a high-level description in an ESL tool, optimizing hot spots using IP blocks or hand-coded HDL.

This transition is a good thing. Five years ago, you couldn't build much of anything in an FPGA without hard-core FPGA skills. This kept FPGAs out of reach for most DSP programmers. Thanks to high-level tools, the average DSP programmer can now make the leap to FPGA design with a reasonable amount of effort – and produce reasonably optimal designs.

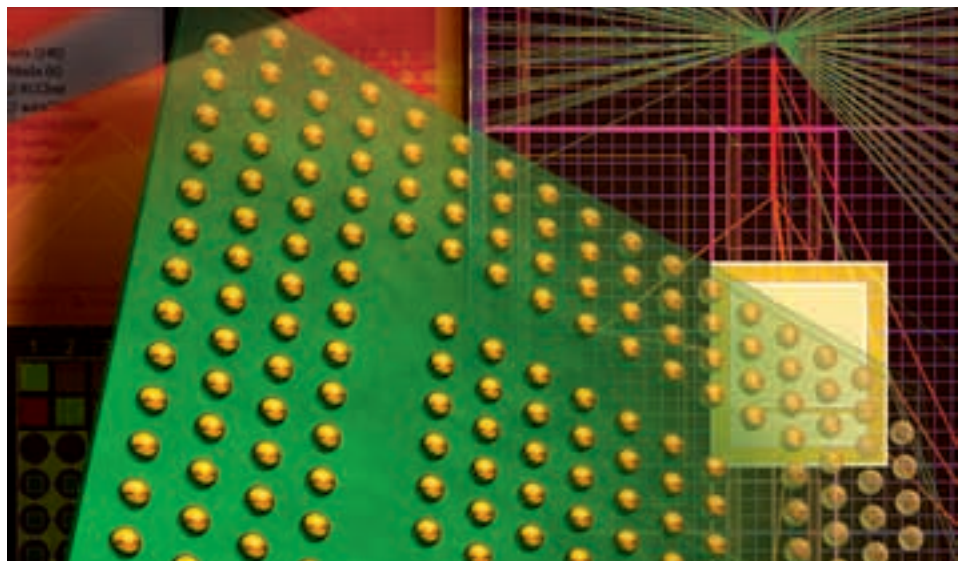
Good examples of this trend are the AccelDSP™ synthesis tool and System Generator for DSP tool, both from Xilinx, which integrate into the ubiquitous Simulink environment. These tools allow DSP developers to go straight from algorithm design to FPGA implementation in a familiar tool flow.

FPGAs are also following the path of DSPs in the area of intellectual property. Not long ago, DSP programmers had to write all of their own code. Today, DSP

developers can turn to outside sources for “commodity” software. FPGA design has followed a similar path – designers who once had to develop applications on their own now have access to IP blocks ranging from FFTs to video encoders. These IP blocks make development faster and easier, giving DSP designers a way to translate

The growing product lines also let designers migrate from one FPGA to another. For example, a design that starts out on a smaller FPGA can migrate to a larger FPGA if the design needs more functionality.

Although the evolution of FPGAs has mimicked that of DSP in many regards, there is one area in which FPGAs are making a dra-



their algorithm knowledge into FPGA designs. For example, in System Generator, designers can use the FFT IP block to try out different FFT variants and explore different trade-offs between performance and resource utilization.

Finally, FPGA families are growing – much in the same way as DSP processor families have grown – to include products that span a broad range of price and performance points. This makes FPGAs suitable for a wider range of applications. For example, Xilinx® products range from the Virtex™-5 FPGA, which is suitable for high-performance DSP applications, to the low-cost Spartan™-3A DSP device.

matic departure. As processors move to smaller process nodes, FPGA designers get more gates to play with. This makes it easier and easier to produce a design with the desired performance.

In contrast, process scaling has reached a point of diminishing returns for traditional single-core processor architectures, forcing vendors to turn to multi-core architectures. Although multi-core processors are very powerful, they are much more difficult to program than single-core architectures. Thus, DSPs are generally becoming harder to program while FPGAs are becoming easier to program. If this trend continues, we may see the day when FPGAs are easier to program than DSPs. ●●