Q&A: The Evolution of FPGA Development Tools

National Instruments R&D Fellow Keith Odom discusses the increasing popularity of field-programmable gate arrays (FPGAs), the technology's evolution and how FPGA development tools need to evolve to meet the needs of developers and domain experts.

Q: How prevalent are FPGAs in engineering today?

FPGAs are becoming a part of most embedded designs. The reconfigurable nature of FPGAs is beneficial in addressing tight time-to-market constraints. There also is a trend of FPGAs replacing DSPs and ASICs for a variety of reasons. FPGAs are increasing the availability of built-in processors and dedicated DSP resources, thereby moving FPGAs away from primarily being used for glue logic and into the main logic component of an embedded system. Additionally, the increasing logic density has improved the logic per dollar cost, making it more economical to design with an FPGA rather than design an ASIC, except for very high-volume productions.

Q: How are FPGA development tools different from a few years ago?

As logic density increases and FPGAs are capable of solving more complex applications, designers look to development tools to aid them in adapting to new demands. We saw this when FPGA development tools changed from schematic capture tools to design languages such as VHDL and Verilog. Recently, we have seen greater adoption of SystemVerilog, C-based tools and graphical environments such as LabVIEW FPGA as alternatives to traditional register transfer level (RTL). There is a smaller, newer movement into system-level modeling tools and higher-level synthesis tools as well.

In addition, design cycle constraints have required developers to be more productive with their time. Many developers are integrating existing IP into new designs to help meet these design cycle constraints.

Q: What are the biggest challenges that developers face when working with FPGAs?

Some of the biggest design challenges developers face are shorter design cycles, increased logic density, strict power requirements, meeting timing closure and integration of IP. Furthermore, FPGAs are just one part of the embedded system. Hardware and software developers are collaborating on teams and struggle to partition their applications among the various components most effectively. The complexity of these systems is growing, and the tools need to be able to let developers work efficiently on their designs. Typically, as systems get more complex, the languages used to program the devices must move toward higher abstraction.

Q: How will development tools change to meet the needs of FPGA developers?

Tools are adapting to focus on improving the productivity of developers. This will initially be in the form of better IP integration within the tools and improved code sharing. Developers are likely to start using system-level modeling tools and higher level synthesis tools to abstract them from the details as system complexity expands. Additionally, these higher-level tools will allow domain experts to describe their algorithm sat a level they are most comfortable with, and the tools will generate efficient RTL code for them. This will open up FPGA programming to a new set of users who are not familiar with FPGAs.

LabVIEW FPGA is one example of a higher-level tool that has already allowed domain experts to program FPGAs. The number of software developers and domain experts is now greater than hardware engineers. Enabling this new large set of users to program FPGAs will create a much larger ecosystem, and we can expect to see some great innovations from this group.

NATIONAL INSTRUMENTS™

11500 N Mopac Expwy • Austin, TX 78759-3504 USA • Tel: (800) 433 3488 • Fax: (800) 683 9300 • info@ni.com • ni.com