

ESL Tools for FPGAs

Empowering software developers to design with programmable hardware.

by Milan Saini
Technical Marketing Manager
Xilinx, Inc.
milan.saini@xilinx.com

A fundamental change is taking place in the world of logic design. A new generation of design tools is empowering software developers to take their algorithmic expressions straight into hardware without having to learn traditional hardware design techniques.

These tools and associated design methodologies are classified collectively as electronic system level (ESL) design, broadly referring to system design and verification methodologies that begin at a higher level of abstraction than the current mainstream register transfer level (RTL). ESL design languages are closer in syntax and semantics to the popular ANSI C than to hardware languages like Verilog and VHDL.

How is ESL Relevant to FPGAs?

ESL tools have been around for a while, and many perceive that these tools are predominantly focused on ASIC design flows. The reality, however, is that an increasing number of ESL tool providers are focusing on programmable logic; currently, several tools in the market support a system design flow specifically optimized for Xilinx® FPGAs. ESL flows are a natural evolution for FPGA design tools, allowing the flexibility of programmable hardware to be more easily accessed by a wider and more software-centric user base.

Consider a couple of scenarios in which ESL and FPGAs make a great combination:

1. Together, ESL tools and programmable hardware enable a desktop-based hardware development environment that fits into a software developer's workflow model. Tools can provide optimized support for specific FPGA-based reference boards, which software developers can use to start a project evaluation or a prototype. The availability of these boards and the corresponding reference applications written in higher level languages makes creating customized, hardware-accelerated systems much faster and easier. In fact, software programmers are now able to use FPGA-based reference boards and tools in much the same way as microprocessor reference boards and tools.
2. With high-performance embedded processors now very common in FPGAs, software and hardware design components can fit into a single device. Starting from a software description of a system, you can implement individual design blocks in hardware or software depending on the applications' performance requirements. ESL tools add value by enabling intelligent partitioning and automated export of software functions into equivalent hardware functions.

ESL promotes the concept of "exploratory design and optimization." Using ESL methodologies in combination with programmable hardware, it becomes possible to try a much larger number of possible application implementations, as well as rapidly experiment with dramatically different software/hardware partitioning strategies. This ability to experiment – to try new approaches and quickly analyze performance and size trade-offs – makes it possible for ESL/FPGA users to achieve higher overall performance in less time than it would take using traditional RTL methods.

Additionally, by working at a more abstract level, you can express your intent using fewer keystrokes and writing fewer lines of code. This typically means a much faster time to design completion, and less chance of making errors that require tedious, low-level debugging.

ESL's Target Audience

The main benefits of ESL flows for prospective FPGA users are their productivity and ease-of-use. By abstracting the implementation details involved in generating a hardware circuit, the tools are marketing their appeal to a software-centric user base (Figure 1). Working at a higher level of abstraction allows designers with skills in traditional software programming languages like C to more quickly explore their ideas in hardware. In most instances, you can implement an entire design in hardware without the assistance of an

experienced hardware designer. Software-centric application and algorithm developers who have successfully applied the benefits of this methodology to FPGAs include systems engineers, scientists, mathematicians, and embedded and firmware developers.

The profile of applications suitable for ESL methodologies includes computationally intensive algorithms with extensive inner-loop constructs. These applications can realize tremendous acceleration through the concurrent parallel execution possible in hardware. ESL tools have helped with successful project deployments in application domains such as audio/video/image processing, encryption, signal and packet processing, gene sequencing, bioinformatics, geophysics, and astrophysics.

ESL Design Flows

ESL tools that are relevant to FPGAs cover two main design flows:

1. High-level language (HLL) synthesis. HLL synthesis covers algorithmic or behavioral synthesis, which can produce hardware circuits from C or C-like software languages. Various partner solutions take different paths to converting a high-level design description into an FPGA implementation. How this is done goes to the root of the differences between the various ESL offerings.

You can use HLL synthesis for a variety of use cases, including:

- Module generation. In this mode of use, the HLL compiler can convert a functional block expressed in C (for example, as a C subroutine) into a corresponding hardware block. The generated hardware block is then assimilated in the overall hardware/software design. In this way, the HLL compiler generates a submodule of the overall design.

Module generation allows software engineers to participate in the overall system design by quickly generating, then integrating, algorithmic hardware components. Hardware engineers seek-

ing a fast way to prototype new, computation-oriented hardware blocks can also use module generation.

- Processor acceleration. In this mode of use, the HLL compiler allows time-critical or bottleneck functions running on a processor to be accelerated by enabling the creation of a custom accelerator block in the programmable fabric of the FPGA. In addition to creating the accelerator, the tools can also automatically infer memories and generate the required hardware-software interface circuitry, as well as the software device drivers that enable communication between the processor and the hardware accelerator block (Figure 2). When compared to code running on a CPU, FPGA-accelerated code can run orders of magnitude faster while consuming significantly less power.

2. System modeling. System simulations using traditional RTL models can be very slow for large designs, or when processors are part of the complete design. A popular emerging ESL approach uses high-speed transaction-level models, typically written in C++, to significantly speed up system simulations. ESL tools provide you with a virtual platform-based verification environment where you can analyze and tune the functional and performance attributes of your design. This means much earlier access to a virtual representation of the system, enabling greater design exploration and what-if analysis. You can evaluate and refine performance issues such as latency, throughput, and bandwidth, as well as alternative software/hardware partitioning strategies. Once the design meets its performance objectives, it can be committed to implementation in silicon.

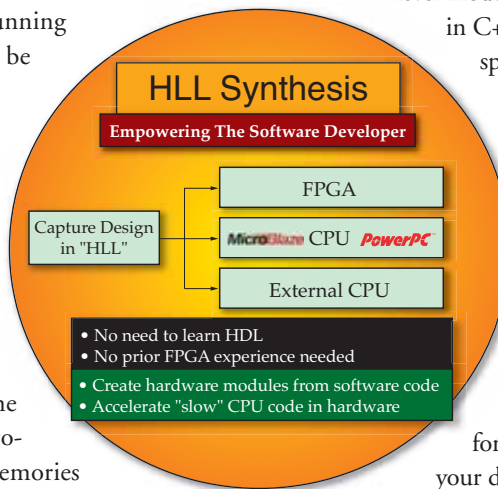


Figure 1 – Most of the ESL tools for FPGAs are targeted at a software-centric user base.

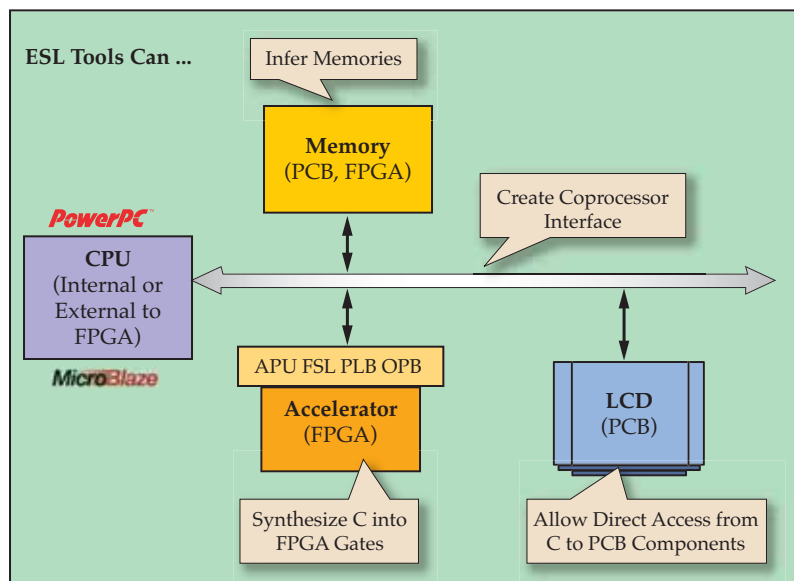


Figure 2 – ESL tools abstract the details associated with accelerating processor applications in the FPGA.

...today's ESL technologies are ready to deliver substantial practical value to a potentially large target audience.

The Challenges Faced by ESL Tool Providers

In relative terms, ESL tools for FPGAs are new to the market; customer adoption remains a key challenge. One of the biggest challenges faced by ESL tool providers is overcoming a general lack of awareness as to what is possible with ESL and FPGAs, what solutions and capabilities already exist, and the practical uses and benefits of the technology. Other challenges include user apprehension and concerns over the quality of results and learning curve associated with ESL adoption.

Although paradigm shifts such as those introduced by ESL will take time to become fully accepted within the existing FPGA user community, there is a need to tackle some of the key issues that currently prohibit adoption. This is particularly important because today's ESL technologies are ready to deliver substantial practical value to a potentially large target audience.

Xilinx ESL Initiative

Xilinx believes that ESL tools have the promise and potential to radically change the way hardware and software designers create, optimize, and verify complex electronic systems. To bring the full range of benefits of this emerging technology to its customers and to establish a common platform for ESL technologies that target FPGAs in particular, Xilinx has proactively formed a collaborative joint ESL Initiative with its ecosystem partners (Table 1).

The overall theme of the initiative is to accelerate the pace of ESL innovation for FPGAs and to bring the technology closer to the needs of the software-centric user base. As part of the initiative, there are two main areas of emphasis:

1. Engineering collaboration. Xilinx will work closely with its partners to continue to further increase the value of ESL product offerings. This will

include working to improve the compiler quality of results and enhance tool interoperability and overall ease-of-use.

2. ESL awareness and evangelization.

Xilinx will evangelize the value and benefits of ESL flows for FPGAs to current and prospective new customers. The program will seek to inform and educate users on the types of ESL solutions that currently exist and how the various offerings can provide better approaches to solving existing problems. The aim is to empower users to make informed decisions on the suitability and fit of various partner ESL offerings to meet their specific application needs. Greater awareness will lead to increased customer adoption, which in turn will contribute to a sustainable partner ESL for FPGAs ecosystem.

Getting Started With ESL

As a first step to building greater awareness on the various ESL for FPGA efforts, Xilinx has put together a comprehensive ESL website. The content covers the specific and unique aspects of each of the currently

available partner ESL solutions and is designed to help you decide which, if any, of the available solutions are a good fit for your applications. To get started with your ESL orientation, visit www.xilinx.com/esl.

Additionally, Xilinx has also started a new ESL for FPGAs discussion forum at <http://toolbox.xilinx.com/cgi-bin/forum>. Here, you can participate in a variety of discussions on topics related to ESL design for FPGAs.

Conclusion

ESL tools for FPGAs give you the power to explore your ideas with programmable hardware without needing to learn low-level details associated with hardware design. Today, you have the opportunity to select from a wide spectrum of innovative and productivity-enhancing solutions that have been specifically optimized for Xilinx FPGAs. With the formal launching of the ESL Initiative, Xilinx is thoroughly committed to working with its third-party ecosystem in bringing the best-in-class ESL tools to its current and potential future customers. Stay tuned for continuing updates and new developments. 🌈

Partner	FPGA Synthesis	Xilinx CPU Support	FPGA Computing Solution
Celoxica	●	●	Handel-C, SystemC to gates
Impulse	●	●	Impulse C to gates
Poseidon	●	●	HW/SW partitioning, acceleration
Critical Blue	●		Co-processor synthesis
Teja		●	C to multi-core processing
Mittrion	●		Adaptable parallel processor in FPGA
System Crafter	●		SystemC to gates
Bluespec	●		SystemVerilog-based synthesis to RTL
Nallatech	●	●	High-performance computing

Table 1 – Xilinx ESL partners take different approaches from high-level languages to FPGA implementation.