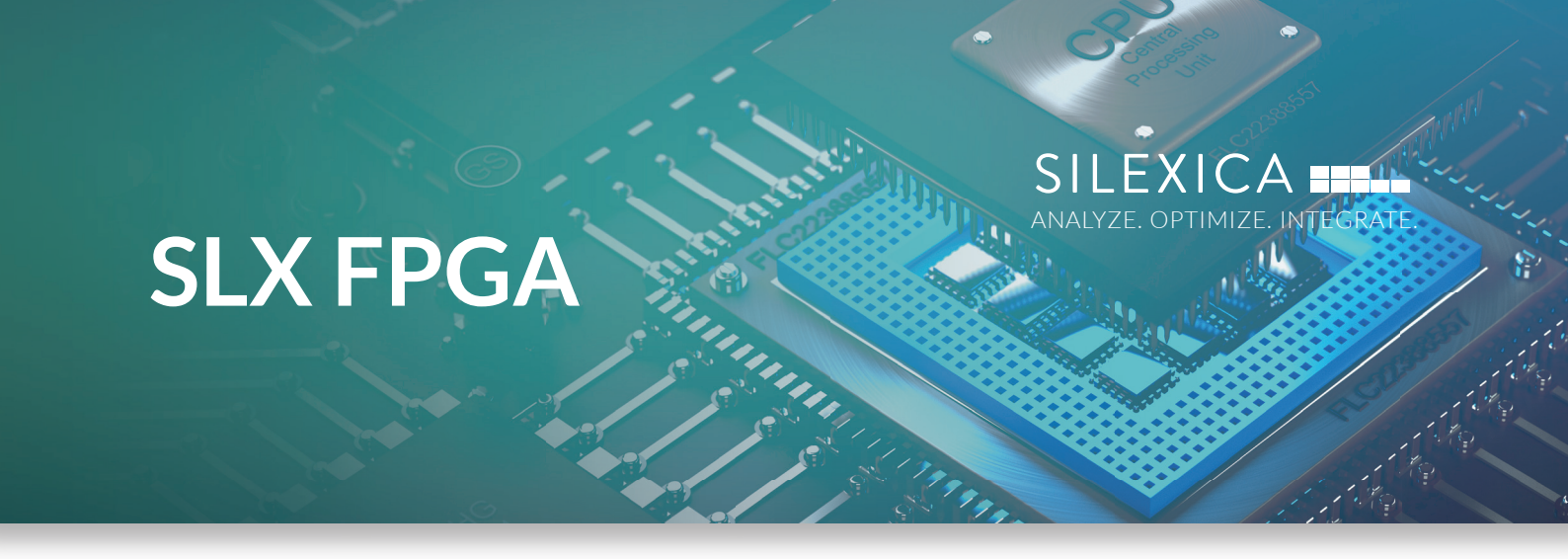


SLX FPGA

SILEXICA
ANALYZE. OPTIMIZE. INTEGRATE.

SLX FPGA - Accelerating the journey from C/C++ to Hardware

SLX FPGA helps to convert your C/C++ code into an FPGA bitstream easier, faster, and with higher performance. Leveraging standard HLS (High Level Synthesis) tools from FPGA vendors, SLX FPGA tackles the challenges associated with the HLS design flow including non-synthesizable C/C++ code, non-hardware aware C/C++ code, detecting application parallelism, where to insert pragmas, and how to determine optimal SW/HW partitioning. Using SLX FPGA enables you to get to market faster by leveraging the benefits of HLS for FPGA design entry. These benefits include improved productivity through designing at a higher level of abstraction, orders of magnitude faster simulation than traditional RTL simulation, and higher QoR through high-level optimizations and design space exploration.

SLX FPGA addresses the challenges of using a HLS design flow by performing static and dynamic code analysis and providing deep insights in to the user's C/C++ code. Through this code analysis, SLX FPGA identifies non-synthesizable C/C++ code, detects non-hardware aware data types, and pinpoints parallelism within the SW that can be implemented in HW for acceleration. SLX FPGA provides guided and automatic code refactoring for HLS synthesizability helping to solve the biggest barrier and most time-consuming aspects of using HLS design flows. SLX FPGA then uses the detected parallelism to automatically generate and insert HLS pragmas which optimize the design for performance and area utilization.

ANALYZE

Analyze your C/C++ code for HLS synthesizability, FPGA performance, hardware aware C/C++ code and parallelism

OPTIMIZE

Optimize your C/C++ code with guided and automatic code refactoring and pragma insertion

INTEGRATE

SLX FPGA integrates directly into standard HLS design flows

FEATURES AND CAPABILITIES

Analyze Code

SLX performs static and dynamic code analysis to provide a top to bottom view into the C/C++ source code. This system level view includes call graphs, function/task interdependencies, memory access patterns, array and data structure access analysis, shared variables, blocking states, and more.

SLX uses this analysis to provide actionable insights into:

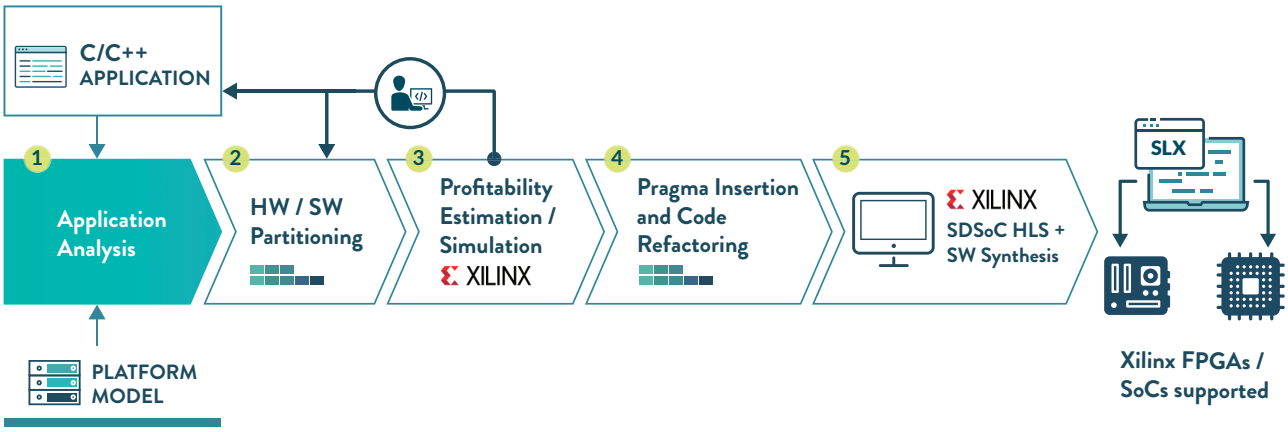
- **C/C++ code that is non-synthesizable by HLS compilers or non-hardware aware code that becomes slow or bloated in the FPGA**
- **Functions within the C/C++ code that can be implemented in parallel in the FPGA Logic for HW acceleration**
- **Hotspots in communication, computation, and memory access patterns**

Optimize Code

SLX FPGA utilizes the actionable insights provided by the top to bottom code analysis to significantly improve the HLS experience by eliminating many of the common challenges of using HLS to convert C/C++ to an optimized hardware/software design.

Optimization features include:

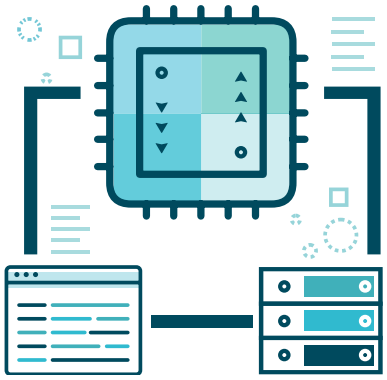
- **Delivery of guided and automated code refactoring to: 1. Convert code that is non-synthesizable into an FPGA and 2. Convert C/C++ code that is non-hardware aware.**
- **Implements parallel functions in FPGA logic by automatically inserting HLS pragmas into source code for an optimized hardware/software implementation. Optionally OpenMP annotations can be inserted to exploit embedded processors in the FPGA.**



Integrate

SLX FPGA integrates optimized C/C++ code and pragmas back into the original source code in preparation for HLS synthesis.

It is fully integrated with Xilinx Vivado HLS and the SDSoC Development Environment to create a complete path from C/C++ to FPGA synthesis. SLX FPGA can be used on the desktop from a powerful GUI, from command-line or integrated into your agile, continuous workflow.



SLX FPGA

The SLX FPGA tool provides an end-to-end flow for engineers working on mixed HW/SW designs. It can support projects in automotive, aerospace & defense, industrial, medical, video, 5G and more. You can find out more at our website here: www.silexica.com or arrange a meeting/demonstration with us at: meetus@silexica.com

Clear guidance into HLS Synthesizablity of C/C++ code

HW/SW Partitioning Exploration

Automatic insertion of pragmas (OpenMP and HLS)