PRODUCT BRIEF

SILEXICA

SLX FOR C/C++

Optimizing C/C++ applications on complex multicore SoCs requires a complete understanding of the software and hardware. Developers are challenged with the task of optimizing sequential and parallel code for multicore SoCs that comprise a variety of compute engines. For the most efficient utilization of CPUs, DSPs and FPGAs, understanding the code structure and interdependencies between applications, tasks and variables is required for streamlined software development, refactoring and software design.

SLX enables the development and optimization of C/C++ applications for heterogeneous multicore platforms with unparalleled insights into the hardware and software interdependencies. This allows for code architecting to achieve the most optimal performance for a specific code on a particular multicore SoC. SLX blends hardware insights (including the target's processing micro-architecture, resource utilization, and memory and communication latencies) with software flow to give deep insights into execution.

ANALYZE

Analyze your software to fully understand your code and automatically identify further parallelization opportunities.

Absolute code understanding

OPTIMIZE

Optimize the distribution of your application for a multi-core system to achieve the most efficient utilization of driven by performance, power, and memory constraints.

Meet challenging requirements

INTEGRATE

Implement easy-to-use recipes and automatically generate code, instantly improving your software.

Faster time to market

FEATURES AND CAPABILITIES

ANALYZE

ANALYZE CODE

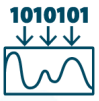
Dynamic code analysis is the foundation for a complete understanding of the software flow and interdependencies. Static analysis provides visibility into the code structure and static call trees; but alone fails to provide all the insights needed for code development on complex software with origins that include in-house, commercial, legacy, and open source. The problem is made more challenging for software developers as 3rd party modules are often delivered without source code to review. To meet the hurdles encountered of software development and optimization for multicore SoCs, SLX technology combines static analysis with dynamic analysis to provide the complete picture of software interdependencies.

SLX can be used early in the design cycle ahead of source code with software models executed on host-based SoC system models. For cross embedded software developers, SLX combines static analysis with execution traces captured from the SoC based target. Optimizing software on multicore systems based on multi-OS, multi-application and multi-tasking software modules requires in-depth insight to partition code for the optimal combination of SoC computing resources. The blend of static, dynamic and semantic code analysis allows SLX to identify modules suited for CPU or FPGA execution, acceleration engine off-load and exposing parallelization opportunities.

SLX provides application call graphs, memory and cache analysis, detailed variables and memory analysis dependencies, and communication and synchronization pattern analysis.

Static and Dynamic Source Code Analysis

Data and Control Dependencies



Cross-Target Performance Estimation

Call-graph and Variable Access

OPTIMIZE

FIND PARALLELISM

SLX understands the behaviour to give unprecedented insights into the execution behaviour. It weighs the costs and benefits of software optimization to further exploit the target's resources. SLX identifies:

System Stability Analysis

SLX gives insights into the stability with a powerful analysis framework, including randomization of scheduling to force different event chains, increasing load on communication channels or processors. Latency and throughput constraints can be verified automatically for a huge number of different scenarios.

Software Distribution for the System

Sophisticated optimization techniques perform "what-if" analysis to visualize and optimize the software for the hardware compute blocks on a multicore system. Optimizations are driven by performance, power and memory requirements for a specific code base given a combination of CPUs, DSPs, FPGAs, GPUs.

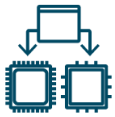
Parallelism Detection

A pattern-based framework identifies missed parallelism in the code that can be exploited to allow for code refactoring to partition modules for execution on a multicore system. Different levels of parallelism are supported including task, pipeline and data level parallelism.

Dlp, Tlp, Plp Extraction

Identification of Blocking Dependencies And Control

Automatic and User-Annotated Weighting

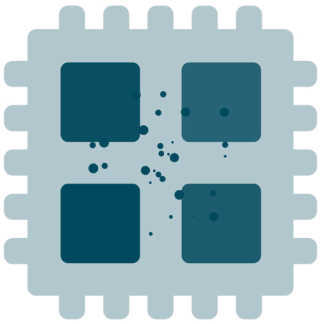


Target Specific

INTEGRATE

Guide and Rewrite

SLX helps users migrate their existing application either by offering powerful hints to help users rewrite code, or the source-to-source technology can rewrite code automatically by inserting pragmas for existing shared memory APIs, such as OpenMP 4.5, or customized internal workflows.



Silexica's unique source-to-source compiler technology allows a developer to significantly increase the turn-around time for software changes and his productivity level. Tying results to source code lines and variables during the Analyze and Optimize phases enables source-to-source automatic rewriting and ensures the feedback to users is accurate and comprehensive.

THE SILEXICA SOLUTION

SLX gives you absolute code understanding to meet the most challenging multicore system requirements.

Clear suggestions to spend less time during multicore migration

Mapping dependent code generation for multicores

Automatic insertion of pragmas (OpenMP, HLS, etc.)

AUTOSAR support