PRODUCT BRIEF

SILEXICA

SLX FOR SCHEDULING DESIGN

(non-Autosar projects)

SLX for Scheduling Design provides a number of solutions, predominantly for automotive projects that do not involve AUTOSAR Classic or Adaptive. By giving a full understanding of the dependencies between functions and tasks, the scheduling can be easily optimized.

By using the power of SLX's Function Level Parallelism and Task Level Parallelism capabilities, power consumption can be reduced while still meeting crucial system latency constraints. SLX for Scheduling Design operates in three phases, Analyze, Optimize and Integrate:

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 on your target platform, driven by performance, power and memory constraints.

✓ **Meet challenging requirements**

IMPLEMENT

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

✓ **Faster time to market**

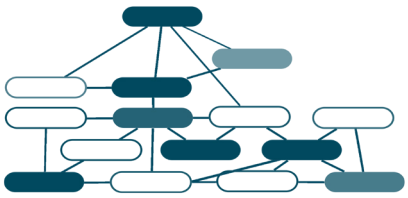
FEATURES AND CAPABILITIES

ANALYZE

Analyze code

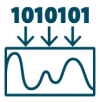
Optimal scheduling requires knowledge of the dependencies between functions and tasks and how often data is accessed. SLX puts analytical results in the human domain with deep analysis for:

- ***Task Data Dependencies***
  Find and display data dependencies across task borders
- ***Function Dependencies***
  Display data dependencies of runnables assinged to a single task
- ***Function Call Graph***
  Get deeper insights to the implementation of your runnables utilizing the function call graph
- ***Code Analysis Graph***
  Get a system wide overview of data elements and accessing functions
- ***Interactive Dependency Graph***
  Add or remove dependencies between functions graphically



Static and Dynamic Source Code Analysis

Cache-, Memory- and Communication Analysis



Cross-Target Performance Estimation

C/C++, data-flow, task-graphs

OPTIMIZE

Function Parallelization

The next step sees SLX optimize the execution of functions and tasks mapping. Function Level Parallelization applies the proven technology Silexica developed for sequential code, as they are also sequential program elements.

Function Parallelization

Function Level Parallelization speeds the execution of tasks without changing the properties of the original application. SLX implements a scheduling algorithm to distribute a task's functions among multiple processors, ensuring data-dependencies are satisfied. The results can be visualized as Gantt charts and generated schedules can be applied to a legacy system. SLX also supports Task Level Parallelism by decoupling producer-consumer dependencies, given that the most recent data is not critical for the application behavior. This enables the producer and consumer to run in parallel, for an additional efficiency.

Memory Mapping

Based on the analyzed variable accesses and the generated multicore schedule, SLX maps variables to the selected platform's memories efficiently. The mapping can be shown visually using mapping tables and charts to help investigate the memory consumption of your application. The variable mapping reduces the amount of costly cross core communication.

Power Optimized Parallelization

Combining Function Level Parallelism and Task Level Parallelism, SLX reduces the execution time allowing changes to system voltage and frequency. This reduces power consumption while still meeting system latency constraints. Alternatively, the increased capacity can be used to augment application features.

Power, performance, and memory-driven SW distribution

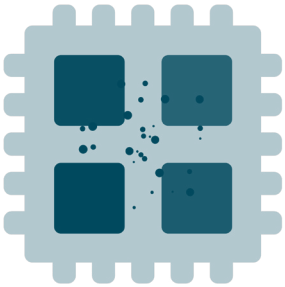
Optimized task schedules under tight timing constraints

Selection of DVFS power states to decrease peak and average power

IMPLEMENT

Faster Time to Market

SLX supports the export of AMALTHEA model to allow for seamless integration with 3rd party tools. AMALTHEA is a xml-based exchange format for embedded multi-core systems. The AMALTHEA format allows for storing both the software and hardware model of the system. The type of information stored in the model consists of Task, Runnable, Variable, Scheduler, OS, Memory and Microcontroller information.



THE SILEXICA SOLUTION

SLX improves your time-to-market, your feature set, and lowers costs and power.

Clear suggestions to spend less time during multicore migration

AMALTHEA support

Automatic insertion of multicore syncronisation primitives