



emmtrix
Technologies



Automation Tools for Better Code

Content

Performance Estimator

Page **6**

- Early in workflow
- Master your timing budgets
- Continuous integration support
- Function developer sees impact of implementation
- Intuitive visualization

TargetLink Support

Page **7**

Use Case ePE

Page **8**

Dependency Analyzer

Page **9**

- Identify data dependencies
- Verify your specification
- Document for (re-)certification
- Analyze event chains
- Data flow analysis

Use Case eDA

Page **10**

Performance Estimator Dependency Analyzer

AUTOSAR Support

Page **11**

CI/CD Workflow Support

Page **12**

Parallel Studio

Page **13**

- Parallel C code for Multi-/Manycore CPUs
- Optimize runtimes
- Interactive workflow
- Functional safety according to standards like ISO 26262, DO-178C and others

Code Vectorizer

Page **14**

- Easy exploitation of parallel vector hardware
- Correct-by-construction code generation
- Speedup > 10x

C++ to C Compiler

Page **15**

- Conversions:
 - Simulink to MATLAB®
 - MATLAB® / Octave / Scilab to C
 - C++ to C
- User-controlled optimizations
- Aimed at embedded systems and automatic analysis

Our Services

Page **15**

- Technical Consulting
- Integration & Tool Customization
- Trainings & Support



Welcome to emmtrix Technologies

We are an innovative company in the field of software development for embedded systems from Karlsruhe, Germany, founded in 2016.

As experts in compiler technology and static source code analysis, we help our customers analyze and optimize their code by offering software-tools and services.

Our aim is to provide comprehensive insight into the runtime performance of an application ([emmtrix Performance Estimator](#)) as well as the complexity/interdependency of the code ([emmtrix Dependency Analyzer](#)) early on in the development process. Armed with this information, engineers can make informed decisions right from the start of the project. Our analysis solutions can be integrated into existing workflows and run in continuous integration set-ups on a defined schedule complete with versioning and changelogs.

Going beyond analysis we also help with automated software parallelization ([emmtrix Parallel Studio](#) and [emmtrix Code Vectorizer](#)) to optimize performance on multicore and manycore processors, as well as accelerators such as vector processors, but also GPUs and DSPs. This parallelization can be qualified according to ISO 26262 / DO-178C upon request.

Your emmtrix team

What Our Clients Say

“ We have been working with emmtrix for a couple of years now and we find their technology of great interest. Their expertise in the field of source-to-source compiler technology and their tool suite **emmtrix Parallel Studio** help us in developing and improving our high-performance hardware solution. Sadahiro Kimura, Manager of Advanced Technology, NSITEXE

“ ePS shows where and how performance can be increased. As a „side-effect“ of using ePS the developer quickly learns to design his application to be suitable for multicore HW.

Arndt-Michael Meyer,
Solution & Partner Manager, ETAS GmbH

emmtrix Tool Overview



Some Supported Platforms

| | | | |
|----------------------|------------|-----------|-----------|
| AURIX™ TC2x-TC4x | | STM32 | iMX.8 |
| DSP | Jetson | | |



ePE for Automotive Function Developers Using Simulink and TargetLink

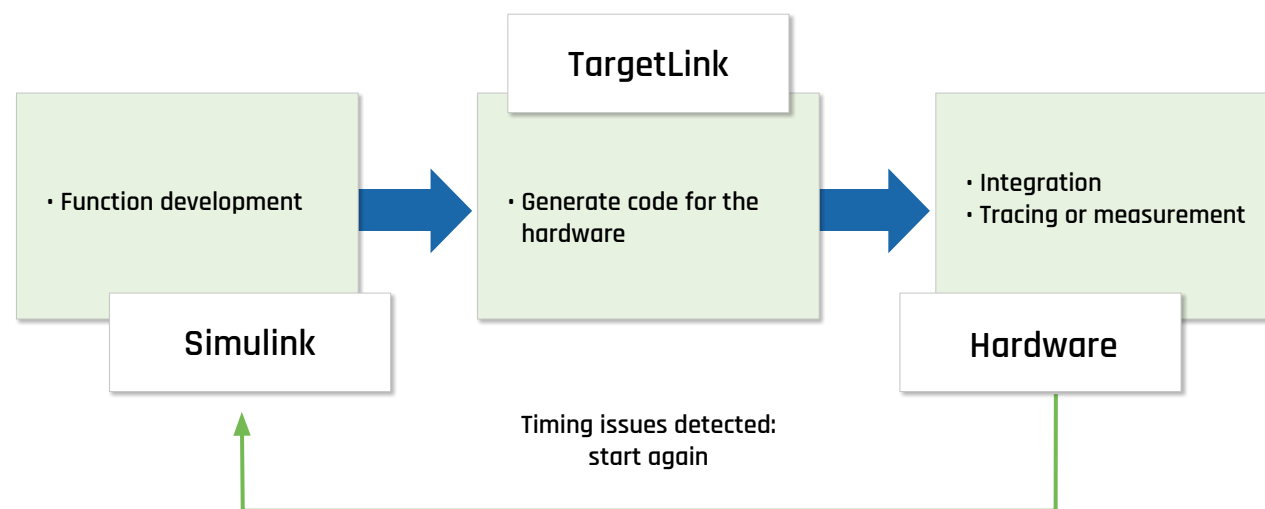
Use Case ePE

ePE calculates the runtime of Simulink blocks on a specified hardware platform based on static analysis of the source code generated from these blocks.

Here is a typical workflow in automotive:

- Functions are developed in Simulink
- TargetLink is used to generate C code for the target platform
- Timing is measured after integration of all functions on the hardware or a simulator
- In case of timing issues, the models have to be adapted and the steps are repeated

Today's Typical Model-Based Workflow in Automotive



Addressing timing issues in this way is expensive and time-consuming, as the issues are detected only after integration and measurements. Manually mapping these results back to Simulink is complicated as the granularity of the information is on the level of runnables or software components as specified in AUTOSAR and not on the actual model.

ePE can be integrated after code generation from the model. The user immediately receives feedback on the performance impact of implementation decisions without the need for integration or execution on hardware. Results of the automated analysis are presented in a hierarchical HTML report using the block names from the model. For all blocks, runtimes are provided for average, minimum, and maximum execution frequencies. This gives the function developer a much better idea of which parts of the application are taking the most time on the hardware. Optimizations are much easier and cheaper to implement at this stage than after integration on the hardware.

Dependency Analyzer

Identify the Event Chains of Your Applications

[emmtrix Dependency Analyzer \(eDA\)](#) assists you with the safety analysis of your applications: Results from a data dependency analysis of the source code can be used to:

- Verify freedom from interference
- Propagate different safety levels of variables
- Detect mixed-criticality dependencies (variables with different safety levels)
- Verify event chains between input and output signals of the system

A further use case is the optimization of your testing strategies by identifying which subsystems are affected by a change in the source code and, more importantly, which subsystems are not and therefore do not have to be tested again.

The tool can easily be integrated into CI flows offering automation, versioning and logging.

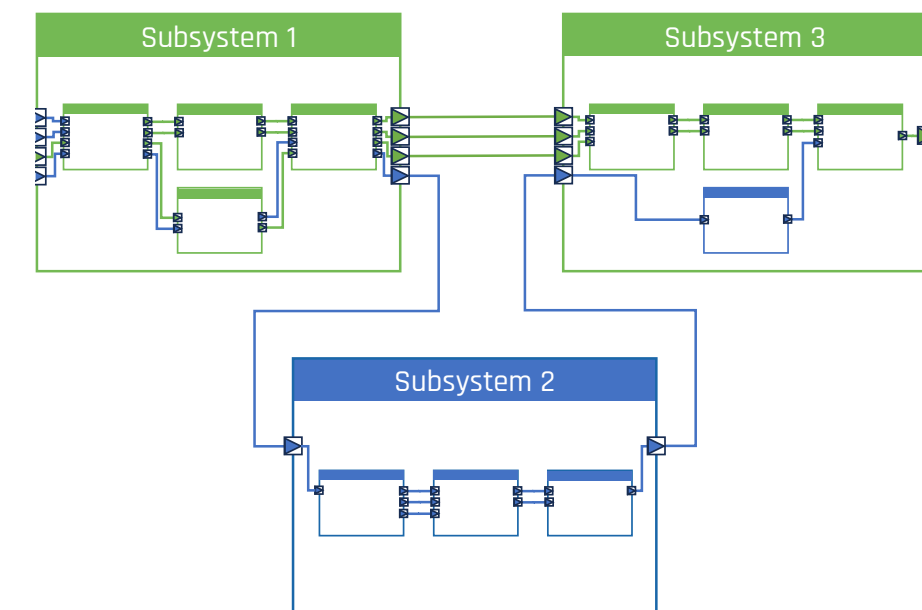
Features

- Analysis takes all possible paths of the control flow graph into consideration to ignore dependencies that can never occur
- Data and control dependencies between variables are calculated
- All calls to sub-functions are taken into account
- Supports analysis of programs consisting of multiple compilation units (source files)
- Supports analysis of delayed dependencies where values are stored in a variable and only fed to the output when the function is called again

Benefits

- Verify your expected dependencies
- Ensure that there are no unwanted connections between input and output signals
- Track down and document all modules affected by an input signal
- Identify which code will be affected by code changes to minimize re-testing effort
- Document all dependencies for any required certification process

Signal-Tracing across Subsystems



eDA for Dependency Verification of AUTOSAR Specifications

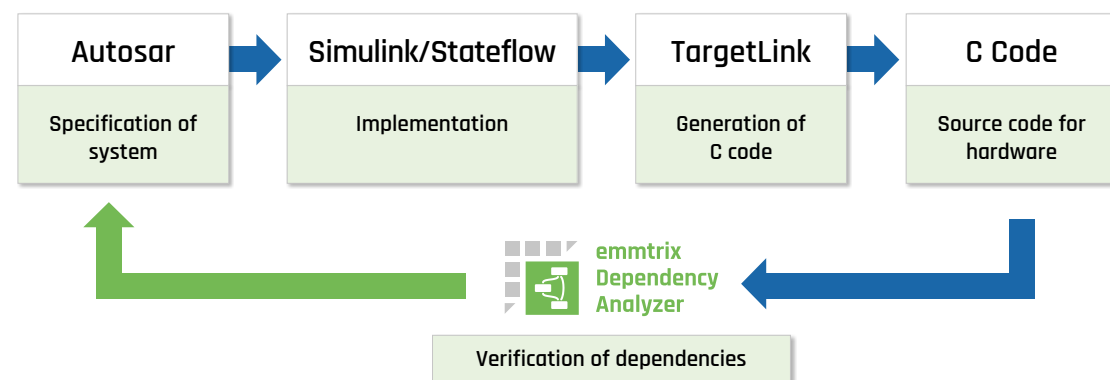
Use Case eDA

eDA extracts and visualizes the internal data dependencies between R-Ports and P-Ports (for more details see next page) of software components (SWCs) by analyzing their internal behavior. A typical development flow in the automotive domain includes the use of AUTOSAR to specify different software components and runnable entities to realize the desired functionality. Simulink and Stateflow are often used to implement the actual functionality. TargetLink is then used to generate the C code for the target platform. All data exchange in the application is typically modeled using AUTOSAR Runnables while the source code of the runtime environment is generated at the end when all components are integrated. eDA automatically analyzes the generated C code regarding its data dependencies. The AUTOSAR model is used as the basis to generate analyzable code and the results of the source code analysis are mapped back to the data prototypes, variables and SWC ports as defined in the AUTOSAR model. This approach allows to verify that, in the generated C code, all data dependencies are detected in line with the model and no undesired connections exist between input and output ports of the model.

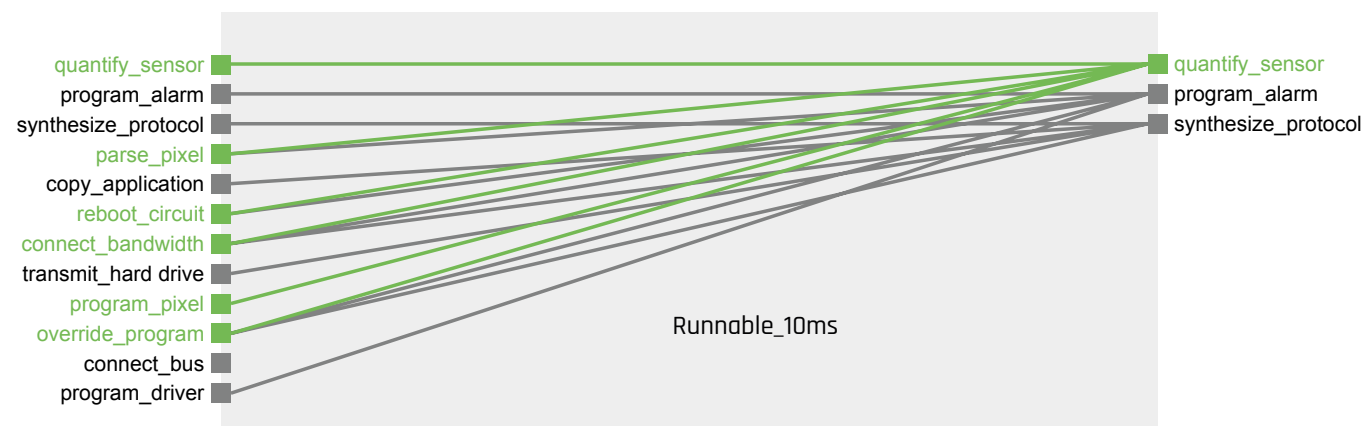
Results can be exported in exchange formats like XML or JSON or visualized like in the image/interactive graphical representation below.

Integration with existing CI flows enables automatic monitoring of dependencies throughout the project lifecycle, so that changes are detected whenever the models or source code are modified.

Typical AUTOSAR Workflow with Integrated eDA



Internal Data Dependencies of a Runnable



Using AUTOSAR with emmtrix Tools

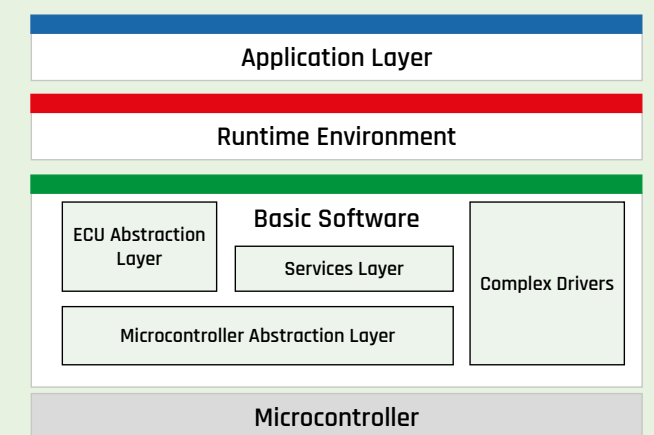
Enhanced Support for Automotive

The AUTOSAR classic platform is a standard specification that defines several abstraction layers from the microcontroller to the applications. The Basic Software (BSW) Layer consists of services such as operating system, communication, diagnostic, and hardware abstraction layers that provide standardized interfaces to the application layer. The Runtime Environment (RTE) acts as the middleware that enables communication between the application software and the basic software layer, abstracting the complexity of the hardware and the BSW modules. Finally, the Application Layer contains functional software components developed by OEMs or suppliers, which can be easily ported across different hardware platforms due to the abstraction provided by the RTE.

Features in an AUTOSAR Environment

- Automatic extraction of relevant runnables based on trigger events
- Performance estimation of individual runnables on different embedded architectures using, for example, Infineon AURIX, ARM Cortex or RISC-V processors
- Extraction of data dependencies with individual runnables to verify and obtain more detailed information on dependencies as specified in AUTOSAR
- Interactive dependency visualization per runnable
- Enables event chain analysis across multiple SWCs
- Tracking of signals and their safety levels throughout the system, e.g. to detect OBD-relevant parts

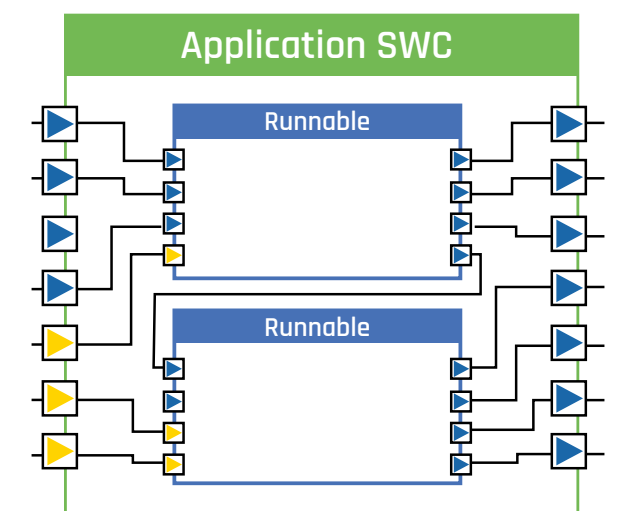
AUTOSAR Software Stack



The emmtrix analysis tools ePE and eDA are designed to analyze C code on the Application Layer in the form of runnables that are part of Application Software Components (SWCs) and make up the functional part of applications. They can be considered as running isolated from the rest of the system as all interaction with other runnables is handled through function calls to the RTE.

The entry point for analysis or optimization is typically a runnable, specifically the C function that implements the runnable's functionality. As the RTE can come from different vendors and not all developers in the supply chain have access to it, we generate our own custom RTE from the AUTOSAR specification to facilitate the analysis of the runnables. This covers all relevant topics such as data types, required functions and macro definitions to ensure correct analyzability of the source code.

Structure of a Software Component



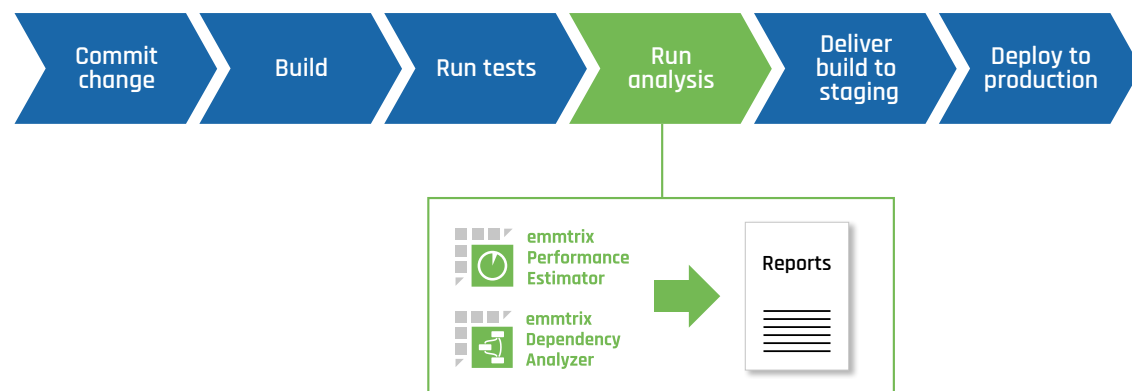
Automated Analysis in Your CI/CD Workflow

Enhanced Support for Automotive

Many modern software development methodologies rely on the use of CI/CD workflows (Continuous Integration/Continuous Delivery). The aim of such a workflow is to automate the process of building, testing and deploying software, so that changes to the source code in the version control system either fail quickly or are quickly pushed to test and staging environments for thorough testing.

The emmtrix tools **ePE** and **eDA** are designed to integrate directly into such flows to provide developers with additional information throughout the development process. A setup in a CI/CD pipeline based on tools like Jenkins might look like this:

Automated Analysis Integrated in CI/CD Workflow



An additional 'run analysis' pass can be added after the build step. Compilable source code is a prerequisite for analysis in both tools.

Available Reports

- Interactive HTML reports, which can be viewed directly as results of Jenkins in the browser
 - For ePE: searchable and sortable with options to hide or show items
 - For eDA: interactive visualisation that highlights relevant dependencies
- Results in XML and JSON formats for automated processing, such as ingestion into databases or use in other tools
- Detailed log files that record all analysis steps with warnings and other important information that can be used for further insight into the analysis

Some Benefits of Using emmtrix Tools in CI/CD Workflows

- Can be used early in development to track projects through their lifecycle and monitor how performance or dependencies change over time. This helps to pinpoint high-impact code changes and get a handle on application timing.
- (Function) developers can simply access the results in their browsers
- Automated monitoring of thresholds can automatically notify designers or architects when runtimes become too high
- Reporting can be focused on parts of the application to provide immediate feedback on the impact of code changes

Parallel Studio

The Tool for Parallel Programming

[emmtrix Parallel Studio \(ePS\)](#) helps you to optimize the performance of your embedded applications on multicore, manycore and DSP architectures as well as any combination of these processing units. Our tool automates and radically simplifies the parallelization process to the point where you simply need to take a few decisions to get good results. The patented graphical user interface (GUI), together with a number of reports, provides full transparency and leaves you in complete control at every step of the process.

Use your existing C code as starting point for the parallelization in **ePS**. Together with [ePS Qualification Kit](#) the parallelization can be performed for applications with functional safety requirements like ISO 26262 or DO-178C.

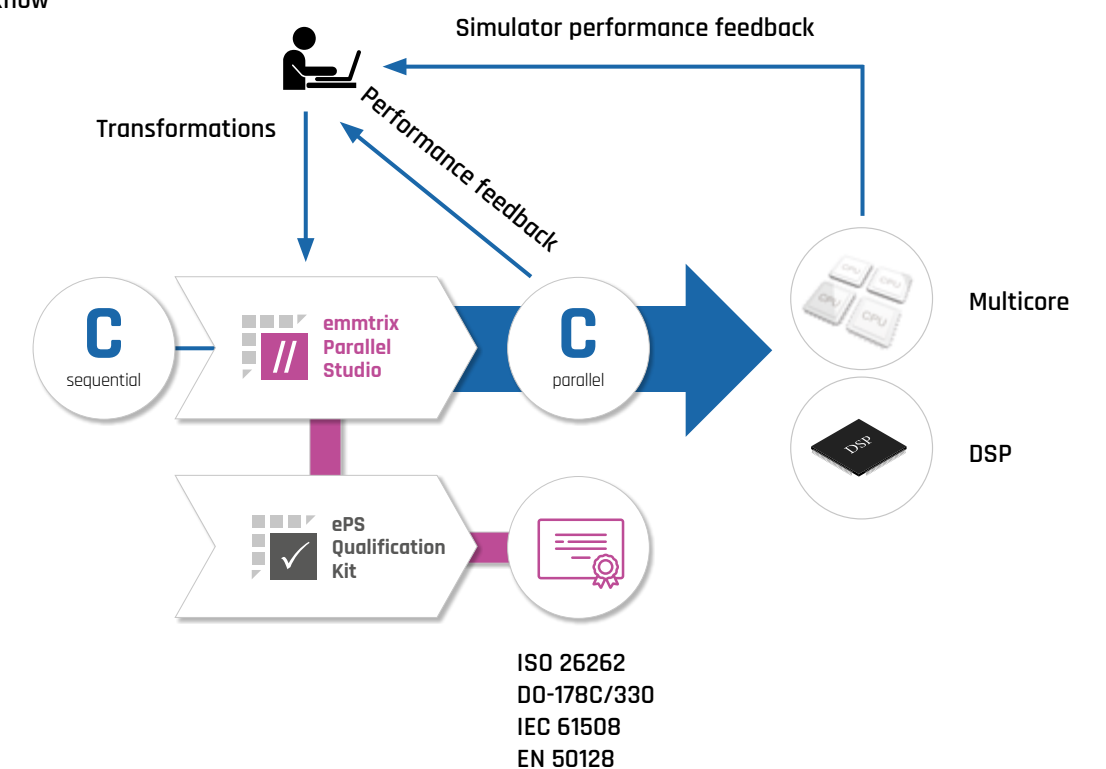
Features

- Automated generation of parallel code
- Interactive optimization with user-friendly Eclipse-based GUI
- Interactive code transformations to optimize parallel code
- Parallelization on runnable-level or function-level and sub-function-level
- Direct deployment of the parallelized program to evaluation boards

Benefits

- Improved application response time and processing throughput
- Correct-by-design approach
- Integrated functional tests for sequential and parallel code (ISO 26262 and DO-178C)
- Easy workflow integration

Parallelization Workflow



Code Vectorizer

The Tool to Vectorize Your Application

The vector units of high-performance microcontrollers promise to speed up the execution of data-parallel applications based on linear algebra by factors greater than 10. Programming such accelerators manually is challenging because it requires deep knowledge of their instruction set and microarchitecture. [emmtrix Code Vectorizer \(eCV\)](#) is your solution to simplify this task significantly.



Features

- Functional testing of vector code independent of target platform
- Code transformations improving data level parallelism and optimizing code for vectorization
- Integration of target platform simulators for performance estimation
- Vectorization-aware code generation from Simulink® models
- Code Fusion: block-crossing vectorization of Simulink® models
- Generation of C code with vector extensions using generic libraries or target specific intrinsics

Benefits

- No need to write vectorized code manually
- Easy exploitation of parallel vector hardware
- Limited hardware knowledge required
- Reduced testing effort
- Functional testing without hardware
- Short development cycles

C++ to C Compiler

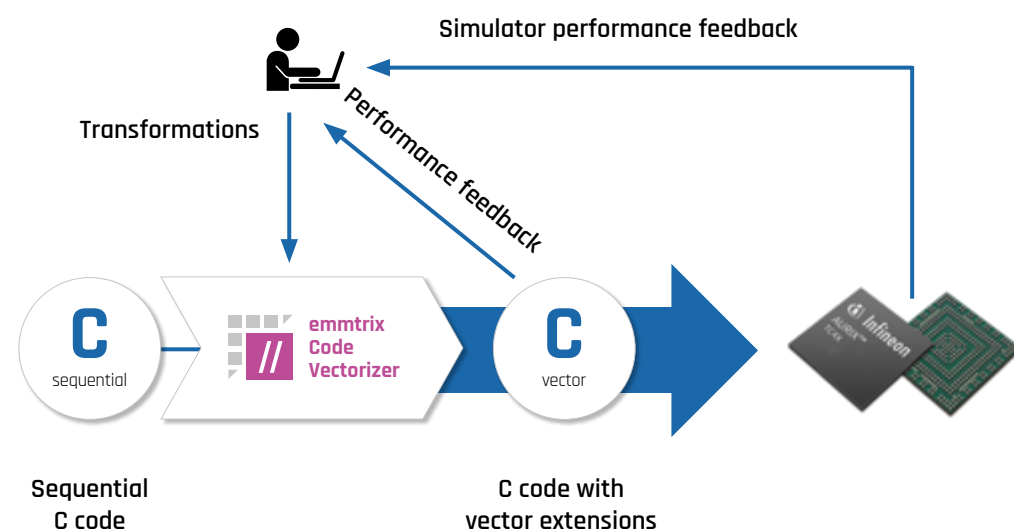
Support for C++ Code

[emmtrix C++ to C Compiler \(eCPP2C\)](#) automatically translates your C++ source code into analyzable C code. The design goal was to keep the binary compilation of the original C++ code and the binary compilation of the translated C code mostly identical. This guarantees the functional correctness of the generated C code. [eCPP2C](#) utilizes the LLVM/Clang compiler technology to enable support of the latest features of the fast evolving C++ standard. In combination with [emmtrix Parallel Studio](#), [eCPP2C](#) enables software parallelization of C++ applications.

Features

- Translation of C++ to C source code
- Utilizes latest LLVM/Clang compiler technology
- Guarantees functional correctness of generated C code by verification tool
- eCPP2C Qualification Kit (ISO 26262, DO-178C/330 or any comparable standard) can be provided on request
- Demystifies how your C++ code is compiled to assembler
- Can be used in combination with (certified) C compilers and C code analysis tools
- Is integrated into emmtrix Parallel Studio GUI to enable C++ code parallelization

Vectorization Workflow



Our Services

Technical Consulting

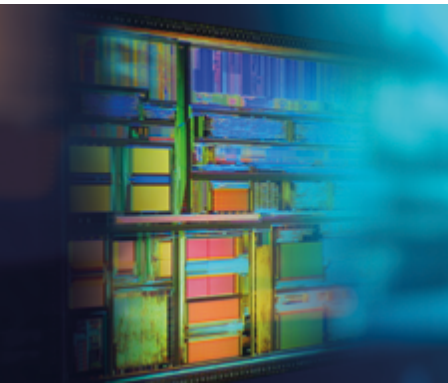
- Performance optimization for single-core architectures (e.g. cache optimization, floating-to-fixed-point conversion)
- Deployment of applications on multicore architectures, DSPs and GPUs (shared/ distributed memory, homogeneous/ heterogeneous)
- Evaluation and selection of appropriate single- and multicore architectures individually and with DSP and GPU accelerators if applicable

Integration & Tool Customization

- Customization of emmtrix tools for your target domain's requirements
- Individual interfaces for the seamless integration of emmtrix tools into your existing workflow
- New product features on demand
- Support for your target architecture of choice (i.e. multicore, DSP, GPU)

Trainings & Support

- User training for all our software packages
- Customisable installation and integration support
- Individual trainings upon request



emmtrix Technologies GmbH
Haid-und-Neu-Straße 7
76131 Karlsruhe / Germany

Phone: +49 721 1803-2880
E-Mail: contact@emmtrix.com

www.emmtrix.com

Automation Tools for Better Code