ITEA Office
High Tech Campus 69 - 3
5656 AG Eindhoven
The Netherlands

T + 31 88 003 6136
E info@itea3.org
W www.itea3.org

ITEA 3 is a EUREKA strategic ICT cluster programme

Exploitable Results by Third Parties

ITEA3 16018 COMPACT

Project details

Project leader:	Wolfgang Ecker, Infineon Technologies AG, DE
Email:	Wolfgang.Ecker@infineon.com
Website:	https://www.edacentrum.de/compact/

Name: MOO Compiler		
Input(s)	Main feature(s)	Output(s)
▪ Target platform/processor characteristics. ▪ Program to be compiled. ▪ Program characteristics.	▪ A compiler tool with machine learning driven optimizations ▪ Optimizes for energy consumption, execution time and code size simultaneously ▪ Applies optimizations and optimization sequences depending on the characteristics of the input program.	▪ Optimized binary code for the target platform/processor.
Unique Selling Proposition(s):	▪ Multi-objective optimizations to better fit a programs binary code onto a target system with limited resources. ▪ Optimizes all objectives equitably. ▪ Includes optimization for energy consumption. ▪ Targets embedded and IoT platforms. ▪ Is easy to use by the end user despite of the included machine learning modules. ▪ Machine learning components are pre-trained (by the vendor). ▪ Is pre-trained per target platform. ▪ Requires only short training time and little training effort. ▪ No negative impact on compilation time. ▪ Is based on the popular LLVM open source compiler. (None of the above items is available in open source compilers.)	
Integration constraint(s):	Constraint(s): ▪ MOO (respectively the machine learning components) need to be retrained if code for a different target platform shall be generated. ▪ A sufficient amount of heterogeneous, real world training samples (i.e. programs) for a specific target platform is required to achieve good optimization quality and to increase the accuracy regarding the estimates and predictions of the machine learning components. No constraint(s): ▪ Is available for Windows and Linux host platforms. ▪ MOO can be used like the classic LLVM compiler it is based on.	

Name: MOO Compiler	
	▪ Does not need a change of the work flow for building programs, performing continuous integrations (CS) or integration into test cycles, etc.
Intended user(s):	Companies and engineers needing to create code for embedded or IoT targets with (very) limited resources regarding memory, processor performance, and energy supply.
Provider:	ABIX GmbH and research partners from the Vienna University of Technology: ▪ Institute of Computer Technology (ICT) ▪ Institute of Computer Engineering (ICE) - Embedded Computing Systems group (ECS)
Contact point:	Manfred Kreutzer – ABIX GmbH: mkreutzer@a-bix.com
Condition(s) for reuse:	▪ MOO Compiler (commercial product): Commercial license (details are to be determined) ▪ MLComp Compiler (research compiler): Research or open source license (details are to be determined)

Name: Tooling for Energy Optimization of Embedded Software		
Input(s):	Main feature(s)	Output(s):
▪ Software source code ▪ Platform constraints ▪ Application constraints	▪ Automated workflow of timing and power analysis ▪ Automated optimization workflow	▪ Analysis results ▪ Optimized source code

Unique Selling Proposition(s):	▪ Designed to be embedded in model-based firmware development with source code generation ▪ Can be used standalone or embedded in a workflow with IoT-PML and Enterprise Architect ▪ Automation of analysis and optimization tasks in one library ▪ Future versions will include automated optimization decisions
Integration constraint(s):	▪ Python >= 3.8 ▪ Python library: pydantic 1.7.3 ▪ Clang/LLVM 11.0 ▪ CMake >= 3.12 ▪ External analysis tools. Integrated support for Timing-Annotation (EKUT source level framework), Timing-Annotation+ETISS, External HW-Measurements with RedPitaya Board
Intended user(s):	▪ Embedded SW developers ▪ Researchers
Provider:	▪ Eberhard Karls Universität Tübingen (EKUT)
Contact point:	▪ Oliver Bringmann – oliver.bringmann@uni-tuebingen.de ▪ Michael Kuhn – michael.kuhn@uni-tuebingen.de
Condition(s) for reuse:	▪ Case-by-case decision
	Latest update: 2020-12-09

Name: MODELTime		
Input(s):	Main feature(s)	Output(s):
▪ SW source code with build environment or SW binary code ▪ HW platform(s) for benchmarking	▪ Fast and accurate timing estimations for the execution time of the input SW program considering its execution on the given HW platforms ▪ Integration in model-based development flow	▪ SW execution time prediction ▪ Visualization of timing properties directly in neoICME
Unique Selling Proposition(s):	▪ Fast and accurate timing estimations that are essential in developing an embedded system (MPSoC support and visualization extension). ▪ Measurement-based technique that implicitly models the different hardware resources included in HW processors.	
Integration constraint(s):	▪ LLVM Compiler Infrastructure 5.0 (or newer) ▪ Lauterbach TRACE32 tracer ▪ libboost ▪ Radare2	
Intended user(s):	▪ Software developers ▪ Hardware developers ▪ Reseach	
Provider:	▪ FZI Forschungszentrum Informatik	
Contact point:	▪ Alessandro Cornaglia – cornaglia@fzi.de ▪ Sebastian Reiter – sreiter@fzi.de	
Condition(s) for reuse:	▪ Trade secret	
	Latest update: 2020-12-07	

Name: neoICME		
Input(s):	Main feature(s)	Output(s):
▪ Optional: IP-XACT, Flattened Device Tree, C source code	▪ Modelling environment for IoT device software ▪ Utilization of graph database ▪ Support for bottom-up and top-down design flow	▪ Neo4j graph database ▪ Structural C source code for IoT software implementation

Unique Selling Proposition(s):	▪ Single source model for IoT software modelling ▪ Tool support for the IoT-PML-based modelling approach
Integration constraint(s):	▪ Neo4j Community Edition (> 3.2.14) ▪ Supported OS: Linux ▪ srcML (srcml.org) dependency for C/C++ source code analysis
Intended user(s):	▪ Software developers ▪ Researchers
Provider:	▪ FZI Forschungszentrum Informatik
Contact point:	▪ Sebastian Reiter – sebastian.reiter@fzi.de
Condition(s) for reuse:	▪ Trade secret
	Latest update: 2020-12-07

Name: Infineon Technologies		
Input(s):	Main feature(s)	Output(s):
Model of the driver and related hardware	Firmware code generation under consideration of the HW/SW interface	Optimized firmware code of the driver and HAL

Unique Selling Proposition(s):	▪ Automatic driver generation to reduce firmware development effort ▪ Driver optimization towards memory consumption and performance via AI guided generation of driver variants
Integration constraint(s):	▪ Python 3.x ▪ Python libraries for XML handling ▪ Mako template Engine ▪ Infineon proprietary code generation framework Metagen with DSL generation enhancement ▪ MetaFirm and associated MetaModels ▪ Enterprise Architect, SparX Systems ▪ Kaktus, Tampere University
Intended user(s):	▪ Software developers to automate the driver design and implementation ▪ Architects as contribution to a rapid starting point for system analysis ▪ Verification engineers as contribution to their testbenches
Provider:	▪ Infineon Technologies, Corporate Design Enabling and Services
Contact point:	▪ Infineon Technologies, wolfgang.ecker@infineon.com
Condition(s) for reuse:	▪ Infineon proprietary

Latest update: 2020-12-09

Name: COMPACT-specific adaption layer for crypto lib		
Input(s):	Main feature(s)	Output(s):
▪ Plaintext ▪ Ciphertext	▪ Key agreement ▪ Authenticated message encryption	▪ Ciphertext ▪ Plaintext

Unique Selling Proposition(s):	▪ Standard algorithms. ▪ High-speed implementation with platform-specific optimizations. ▪ Side-channel protection in theory and practice.
Integration constraint(s):	▪ Needs measurement campaign on every target platform.
Intended user(s):	▪ Industry customers with expert level knowledge.
Provider:	▪ Kasper-Oswald GmbH
Contact point:	▪ info@kasper-oswald.de
Condition(s) for reuse:	▪ Commercial, based on individual plan

Latest update: 2020-12-09

Name: COMPACT-specific adaption layer for crypto lib		
Input(s):	Main feature(s)	Output(s):
▪ Plaintext ▪ Ciphertext	▪ Wraps the implementation of cryptographic primitives into an easy to integrate library. ▪ Supports a very common use case: secure message exchange between two parties. ▪ Abstraction of HW-dependent features such as write/read to/from persistence storage (e.g., EEPROM).	▪ Ciphertext ▪ Plaintext

Unique Selling Proposition(s):	▪ Simple integration, lowers the possibility of error by non-expert integrator
Integration constraint(s):	▪ Requires measured (“certified”) crypto library (see above)
Intended user(s):	▪ Industry customers in general
Provider:	▪ Kasper-Oswald GmbH
Contact point:	▪ info@kasper-oswald.de
Condition(s) for reuse:	▪ Commercial, based on individual plan
	Latest update: 2020-12-09

Name: COMPACT-specific adaption layer for crypto lib		
Input(s):	Main feature(s)	Output(s):
▪ Pre-compiled Crypto library ▪ Hardware platform	▪ Executes measurement campaign to assert side-channel related properties of crypto library on actual hardware	▪ Statistical data
Unique Selling Proposition(s):	▪ Semi-automated framework	
Integration constraint(s):	▪ Needs adaptation to different platforms ▪ Requires good understanding of underlying run-time libraries and possible “quirks” affecting the measurement quality	
Intended user(s):	▪ INTERNAL	
Provider:	▪ Kasper-Oswald GmbH	
Contact point:	▪ info@kasper-oswald.de	
Condition(s) for reuse:	▪ Commercial, based on individual plan	
	Latest update: 2020-12-09	

Name: UML2 API		
Input(s):	Main feature(s)	Output(s):
▪ UML2 based models	▪ API to access UML2 based models (UML, SysML, BPMN, …)	▪ Access to that models using the API

Unique Selling Proposition(s):	▪ API provides the possibility to access model information - modeling-tool- and repository/dbms-neutral. The implementation of the concrete tool and repository has to be done e.g. for Matlab Stateflow, Enterprise Architect, Cameo Systems Modeller, …
Integration constraint(s):	▪ Just the API
Intended user(s):	▪ Tool Vendors
Provider:	▪ SSCE
Contact point:	▪ SSCE
Condition(s) for reuse:	▪ MIT license

Latest update: 2020-12-09

Name: IoT-PML MDG Technologie for Enterprise Architect		
Input(s):	Main feature(s)	Output(s):
▪	▪ Provides Toolboxes for IoT-PML for EA	▪ Wellformed IoT-PML

Unique Selling Proposition(s):	▪ This MDG Technology makes it easy to use IoT-PML
Integration constraint(s):	▪ Based on Sparx Systems Enterprise Architect
Intended user(s):	▪ Current target group: approx. 100.000 Software, Systems Modeller using EA already and beyond
Provider:	▪ SSCE
Contact point:	▪ SSCE (www.sparxsystems.eu/iot)
Condition(s) for reuse:	▪ Open Source using MIT license
	Latest update: 2020-12-09

Name: COMPACT Addin for Enterprise Architect		
Input(s):	Main feature(s)	Output(s):
	▪ Provides Methodoly Support for IoT-PML and Tool Integration	▪ Wellformed IoT-PML
Unique Selling Proposition(s):	▪ This Addin supports usage of IoT-PML and provides capability to integrate analyzer results, code generators and more	
Integration constraint(s):	▪ Based on Sparx Systems Enterprise Architect	
Intended user(s):	▪ Current target group: approx. 100.000 Software, Systems Modeller using EA already and beyond	
Provider:	▪ SSCE	
Contact point:	▪ SSCE (www.sparxsystems.eu/iot)	
Condition(s) for reuse:	▪ Closed Source, but FOC	
	Latest update: 2020-12-09	

Name: Kamel		
Input(s):	Main feature(s)	Output(s):
▪ IP-XACT IEEE-1685 models ▪ Kamel python models ▪ Python Mako templates	▪ Provides modeling template for the user in form of Kamel Python classes with methods (Kamel meta-model) ▪ Model generators (transformations) to target views ▪ Kamel API for Intercoupling of above model inputs and tools (like IP-XACT) and underlying template-based code generators ▪ Kactus2 API for open source IP-XACT tool interoperability	▪ Tailorable with Mako templates. Suitable target views are for example: ▪ Verilog, VHDL, SystemVerilog, SystemC ▪ SW API for HW ▪ Documentation of HW and low-level SW ▪ HW development tool scripts
Unique Selling Proposition(s):	▪ Provides means to model and automate majority of the RTL IP development tasks with light modeling overhead ▪ Not tied to used modeling platform/language or used target application programming language.	
Integration constraint(s):	▪ Python3 ▪ Mako python library (pip install Mako)	
Intended user(s):	▪ HW Architects, HW developers, Firmware SW developers	
Provider:	▪ Tampere university (TAU)	
Contact point:	▪ antti.rautakoura@tuni.fi, esko.pekkarinen@tuni.fi, timo.hamalainen@tuni.fi	
Condition(s) for reuse:	▪ Will be published as open source code library	
	Latest update: 2020-12-01	

Name: Methodology for Distributed CNN Inference on IoT Edge Devices		
Input(s):	Main feature(s)	Output(s):
▪ IoT Nodes ▪ CNN	▪ Code Generation of Distributed CNN Inference Software ▪ Optimization	▪ CNN Inference Software

Unique Selling Proposition(s):	▪ First approach for full distributed inference on IoT Edge nodes ▪ Pools memory resources of all devices in network
Integration constraint(s):	▪ Based on larger library, only larger Edge Devices supported ▪ Experimental, industrial adaption required
Intended user(s):	▪ Providers of IoT Sensor Device Networks that want to integrate AI
Provider:	▪ Technical University of Munich
Contact point:	▪ Daniel Mueller-Gritschneder - daniel.mueller@tum.de
Condition(s) for reuse:	▪ Methodology published ▪ Code not open source

Latest update: 2020-12-08

Name: Extendible Translating Instruction Set Simulator (ETISS)		
Input(s):	Main feature(s)	Output(s):
▪ Embedded SW ▪ Pipeline Description	▪ Simulation environment for Embedded SW (Instruction Set Simulator) ▪ Focus: RISC-V processors	▪ SW profiling information ▪

Unique Selling Proposition(s):	▪ Instruction Set Simulator extendible by timing models ▪ Advanced SW performance profiling
Integration constraint(s):	▪ Out-of-the-box Support for RISC-V ▪ Other processor ISAs need additional modeling effort
Intended user(s):	▪ SoC architects, Embedded SW developers
Provider:	▪ Technical University of Munich
Contact point:	▪ Daniel Mueller-Gritschneder - daniel.mueller@tum.de
Condition(s) for reuse:	▪ ETISS open source available – BSD license ▪ Github link: https://github.com/VP-Vibes

Latest update: 2020-12-08

Name: QEMU Memory Tracer (QMT)		
Input(s):	Main feature(s)	Output(s):
Compiled RISC-V SW Binary, Tracer Configuration	Traces and logs memory accesses (address regions are supplied with tracer configuration). Analyzes Stack and Heap Size of RV32	▪ Log file with memory access trace for RV32

Unique Selling Proposition(s):	▪ QEMU extension for memory analysis for RV32 processors
Integration constraint(s):	▪ Requires QEMU V2.12 (other versions are not tested yet)
Intended user(s):	▪ SW Engineers / RV32 SW Developers
Provider:	▪ Paderborn University / Heinz Nixdorf Institut
Contact point:	▪ Wolfgang Mueller / Circuit and System Design / Heinz Nixdorf Institute
Condition(s) for reuse:	▪ To be negotiated
	Latest update: 2020-12-09

Name: QEMU Timing Analyzer (QTA)		
Input(s):	Main feature(s)	Output(s):
▪ WCET annotations of Basic Blocks in aiT/Absint XML export format ▪ Compiled SW Binary compliant to the aiT ISA analysis	▪ Cycle accurate dynamic timing analysis of the execution of the SW binary	▪ Cycle accurate timing of the execution of the SW binary

Unique Selling Proposition(s):	▪ Dynamic cycle accurate worst-case timing analysis of individual SW binaries ▪ Preprocessor to import aiT/Absint timing analysis reports
Integration constraint(s):	▪ Can be integrated with QEMU V4.2 or higher
Intended user(s):	▪ SW Engineers / Embedded SW Developers
Provider:	▪ Paderborn University / Heinz Nixdorf Institut
Contact point:	▪ Wolfgang Mueller / Circuit and System Design / Heinz Nixdorf Institute
Condition(s) for reuse:	▪ BSD License Agreement

Latest update: 2020-12-09

Name: CAR DETECTOR		
Input(s):	Main feature(s)	Output(s):
▪ Video stream, still images	▪ System for detecting cars when they appear in video stream	▪ XML or other message with car position

Unique Selling Proposition(s):	▪ State-of-the-art detection accuracy
Integration constraint(s):	▪ Runs on a platform that supports Intel OpenVINO library
Intended user(s):	▪ Customs, border control
Provider:	▪ VISY Oy, Tampere, Finland
Contact point:	▪ Jyrki.selinummi@visy.fi
Condition(s) for reuse:	▪ Commercial license

Latest update: 2020-12-04

Name: Model Aware Debugger for Simulink and Stateflow (MADSL & MADSF)		
Input(s):	Main feature(s)	Output(s):
▪ Simulink and/or Stateflow model ▪ Linux based target platform	▪ Cross-level debugging between the model and the generated code running on the target processor ▪ Visualization of the model and corresponding code in a single GUI	▪ Interactive debugging session

Unique Selling Proposition(s):	▪ Debugging of Simulink models and mixed Simulink and Stateflow models at assembler/code level running on arbitrary target processors with Linux support. ▪ Allows to reconstruct model information from the generated code, to step simultaneously through the model and the generated code running on the embedded target device: - set breakpoints on hierarchical block ports (on activation of the block) - set breakpoints on transitions and/or states (incl. entry and exit actions). Designed to be embedded in model-based firmware development with source code generation ▪ Provides a GUI to visualize and to interact between the Simulink/Stateflow model, the generated source code and the assembler code ▪ Future versions will support bare metal system debugging for ARM and RISC-V base platforms
Integration constraint(s):	▪ MATLAB & MATLAB Coder ▪ Simulink & Simulink Coder ▪ Stateflow (optional) ▪ Clang/LLVM ▪ gdbgui ▪ External Linux based development board with GDB/LLDB support
Intended user(s):	▪ Embedded SW developers ▪ Researchers
Provider:	▪ OFFIS e.V. (OFFIS)
Contact point:	▪ Kim Grüttner – kim.gruettner@offis.de
Condition(s) for reuse:	▪ Case-by-case decision

Latest update: 2021-01-13