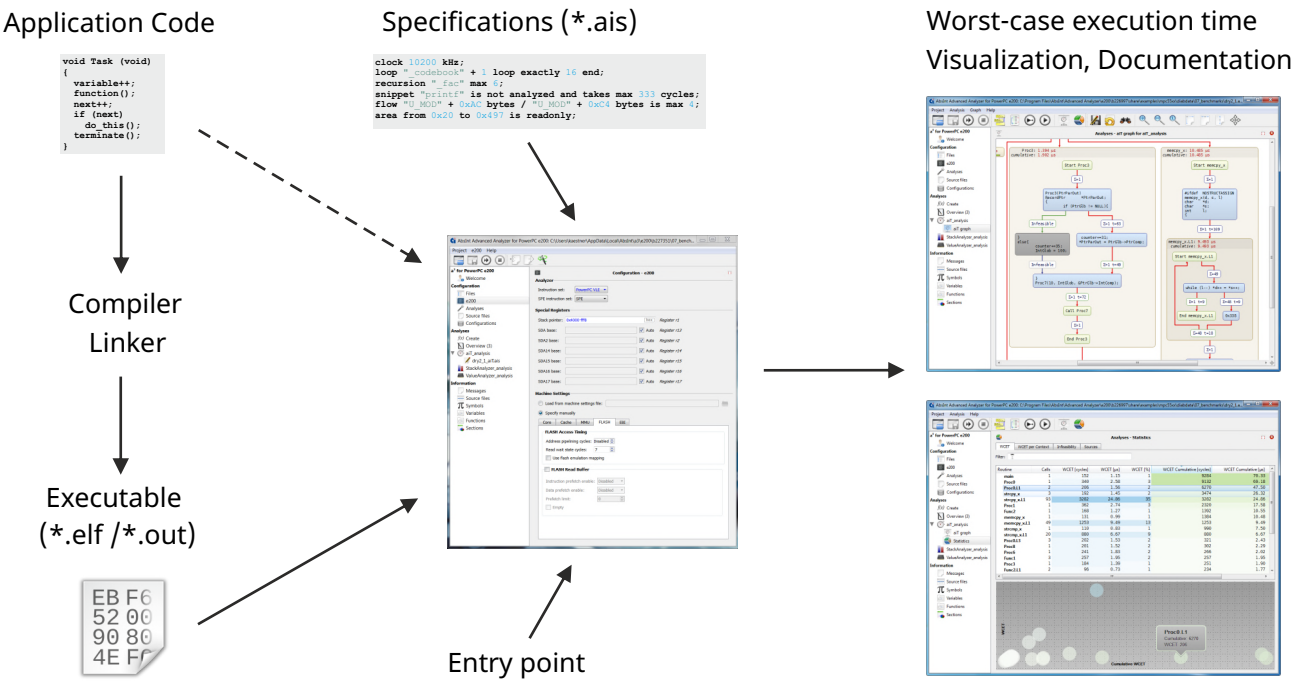
aiT Worst-Case Execution Time Analyzer

Timing Guarantees for Real-Time Systems

aiT WCET Analyzer computes **tight bounds** for the worst-case execution time of tasks in safety-critical systems. These bounds are **safe**, i.e. they are valid for any input scenario and each task execution.

aiT is based on statically analyzing a task's intrinsic **cache and pipeline behavior**, thus enabling the development of complex hard real-time systems on state-of-the-art hardware.



The Challenge:

- **Measuring** the execution time of a task is typically **not safe**. It is often impossible to prove that all the conditions determining maximum execution time are taken into account. Code instrumentation and debug information change the timing behavior.
- Hardware speculation by caches, pipelines, etc. complicates the task of determining the WCET, since the execution time of a single instruction may depend on the **execution history**.
- Analysis methods that do not consider **cache and pipeline behavior** typically seriously overestimate the WCET, leading to a substantial waste of hardware resources.

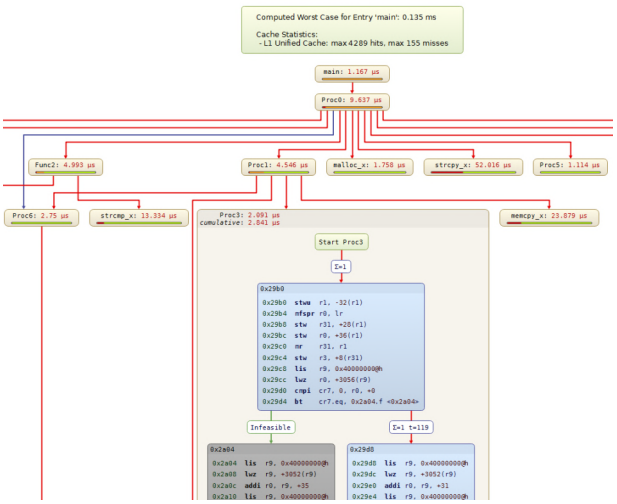
This is where aiT steps in:

- aiT-computed bounds are **valid for all inputs** and each execution of a task. Extensive timing testing is now a thing of the past.
- aiT directly analyzes binary executables. This means that **no modification of your tool chain** or the program's operational behavior and performance is required.
- aiT-computed bounds are tight and reflect the **real performance** of your system. Cache and pipeline effects are fully taken into account. Ensuring deadline adherence is no longer done at the expense of hardware resources.

AbsInt
AbsInt Angewandte Informatik GmbH Science Park 1 D-66123 Saarbrücken Germany
phone: +49 681 383 60 0 fax: +49 681 383 60 20 info@absint.com www.absint.com

aiT Features:

- **Visualization** of the analysis results providing detailed information about key timing aspects, e.g. the **worst-case path** or the **machine state** at any given program point.
- Various **statistics**, interactive **tables**, **graphs** and **charts** that let you quickly **identify bottlenecks** and other areas of interest.
- Analysis **report files** for **documentation** and **certification** purposes, as well as for **integration** with numerous software development tools.
- Graphical **comparison of different analysis runs**. Developers can quickly understand the effect of program modifications on worst-case timing.
- Qualification Support Kits are available providing support for automatic **tool qualification** up to the highest criticality levels (DO-178B, DO-178C, ISO26262, IEC 61508, EN 50128).



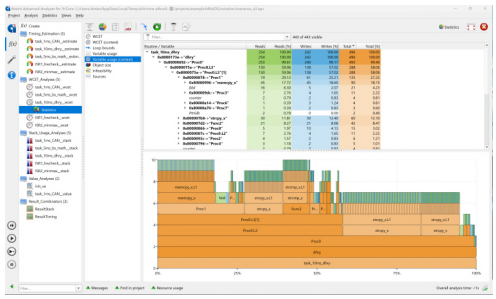
Why do you need aiT?

The worst-case execution time (WCET) of each task in a real-time system has to be known prior to its execution. In event-triggered or periodic systems (e.g. RMA), the WCET is required for schedulability analysis; in time-triggered systems (e.g. TTA, FlexRay, ...), it is required for determining a static schedule.

The increasing performance of microcontrollers enables more and more functionality to be implemented by a single embedded control unit. The software is complex and the timing behavior of the interacting software components rarely known. Typically it is not practical – or even possible – to test the system with all potential inputs.

aiT computes safe upper bounds on the WCET of each task, providing full data and control coverage, enabling timing safety.

- **Interprocedural** analysis enables cache and pipeline behavior to be precisely predicted.
- The analyzer can be run in **batch mode**, enabling **seamless continuous verification**.
- **Flexible annotation mechanism**. Developers can provide programmer-specific knowledge to aiT to further improve the analysis precision.
- aiT can be coupled with **model-based code generators** and **system-level scheduling tools** via an open XML-based interface to provide timing information in the development phase.
- **Variable usage per context** on the worst-case path is displayed as flame graph, helping to optimize the **allocation of objects to memory regions**.



Supported processors: PowerPC 5xx / e200 (55xx, 56xx) / 5777M / **5777C** / e300 (603e, 82xx, 83xx) / 750/ 755 / 7448 / 7447A, i386DX, AM486, Motorola 68020, ARM Cortex M0 / Cortex-M1 / Cortex-M3 / Cortex-R4F / Cortex-R5F, Infineon XMC4500 (ARM Cortex-M4), TI TMS320C3x, TMS320F28, C16x/ST10, XC2365A-104F80L, HC11, Star12/ HCS12/ HCS12X, TriCore 1197 / 1767 / 1782 / 1784 / 1796 / 1797, AURIX TC 2xx, AURIX TC3xx, NEC/Renesas V850, LEON2, LEON3, ERC32.

If your processor is not listed above, please contact us.

Key Benefits:

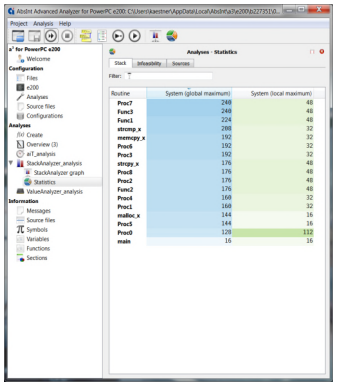
- aiT provides **WCET guarantees** and can replace error-prone methods based on testing and measuring,
⇨**enhancing safety**.
- aiT has been qualified as a **verification tool** according to various safety norms, including DO-178B/C for Level A software,
⇨**enabling certification of safety-critical real-time software.**
- aiT provides automatic tool support for calculating the WCET of your applications,
⇨**saving development time**.
- aiT safely determines the timing behavior of interacting software components,
⇨**enabling software integration**.

AbsInt Angewandte Informatik GmbH Science Park 1 D-66123 Saarbrücken Germany
phone: +49 681 383 60 0 fax: +49 681 383 60 20 info@absint.com www.absint.com

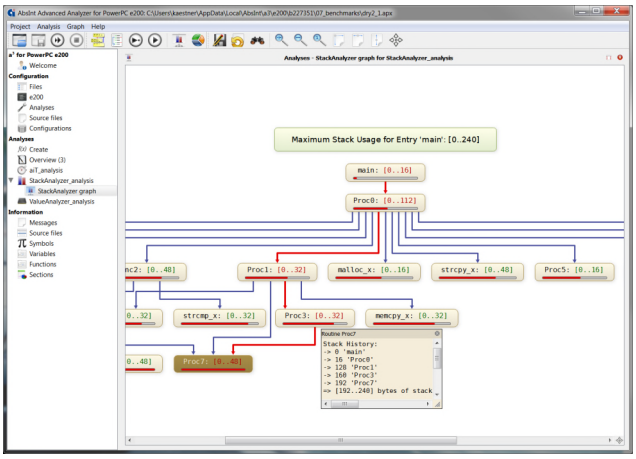
StackAnalyzer – Stack Usage Analysis

Stack overflow is now a thing of the past

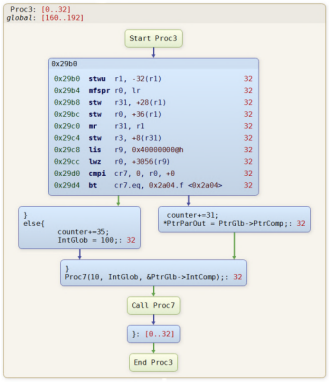
StackAnalyzer automatically determines the worst-case stack usage of the tasks in your application.



Stack usage contributions per function



Call graph with stack usage annotations



Control flow graph with stack usage annotations

Why do you need StackAnalyzer?

Stack memory has to be allocated statically by the programmer. Underestimating stack usage can lead to serious errors due to **stack overflows**. Overestimating stack usage means a waste of memory resources.

- **StackAnalyzer** provides **automatic** tool support to calculate the stack usage of your application. The analysis results are valid for **all inputs** and each task execution.
- **StackAnalyzer** analyzes the **binary executable** and does not rely on debug information, nor on instrumentation.
- **Inline assembly code** and **library function** calls are taken into account.
- **Recursions** and **function pointers** are taken into account.
- Automatic **visualization** of call/control flow graphs with stack usage.
- Current safety standards (DO-178B/C, ISO 26262, IEC 61508, EN 50128, etc.) require to ensure that no stack overflows can occur. With **StackAnalyzer**, you can **prove the absence of stack overflows**. AbsInt's Qualification Support Kits enable a **tool qualification** up to the highest criticality levels.

Supported processors and compilers

- C16x/XC16x/ST10 (Tasking/Keil)
- TriCore, incl. AURIX (Tasking/gcc/Diab)
- PowerPC 32-bit / 64-bit (**CompCert**/Diab/gcc/GHS/GHS Ada/GNAT/CodeWarrior/DDC-1 Score)
- ARM (**CompCert**/TI/ARM/gcc/GHS/IAR/Tasking/clang/HighTec/Diab/Keil MDK-ARM/GHS Ada)
- NEC/Renesas V850/RH850 (GHS/Diab/Renesas CS+)
- Renesas RX (IAR)
- Renesas SuperH (Renesas)
- TI C3x (TI)
- TI C28x (TI)
- TI MSP430(X) (IAR)
- x86 (**CompCert**/gcc/ICC/cygnus/clang)
- M68K (HP/EDS/gcc/Diab)
- FR81S (Fujitsu)
- MCS51 (TI CC254x) (IAR)
- MIPS32 (gcc)
- S12Z (CodeWarrior)
- HCS12(X/XE) (Hiware/Cosmic/IAR)
- LEON2/LEON3/LEON4 (gcc/GNAT/clang)
- ERC32 (gcc/GNAT/clang)
- Freescale ColdFire (HP/EDS/gcc)
- dsPIC (Microchip)
- MCS251 (Keil)
- RISC-V (**CompCert**/gcc)
- Nios II (gcc)
- RL78 (IAR) New

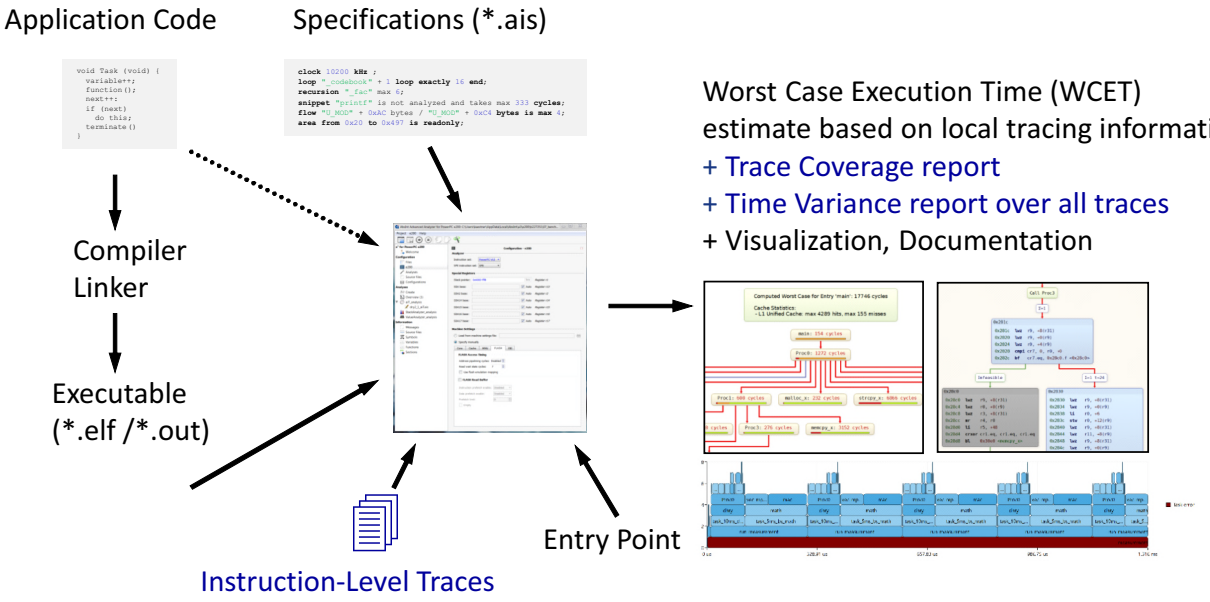
For further targets, please contact us.

AbsInt Angewandte Informatik GmbH Science Park 1 D-66123 Saarbrücken Germany
phone: +49 681 383 60 0 fax: +49 681 383 60 20 info@absint.com www.absint.com

TimeWeaver

Hybrid Worst-Case Execution Time Analysis

TimeWeaver combines static path analysis with real-time instruction-level tracing to provide worst-case execution time estimates. The computed time bounds provide valuable feedback for assessing system safety and for optimizing worst-case performance.



Why do you need TimeWeaver?

- **TimeWeaver** analyses all potential execution paths and computes the **longest path** based on the execution times of trace segments observed in real-time traces.
- **TimeWeaver** supports **non-intrusive tracing**, e.g. Nexus branch history target messages. The computed time bounds are compliant to requirements of safety standards like DO-178B, DO-178C, ISO 26262, etc.
- **TimeWeaver** reports **test coverage** information at the instruction level with respect to all possible execution paths for all considered trace segments. This gives valuable feedback for improving the test coverage of the system.
- **TimeWeaver** generates **customizable reports** and **visualizations** for **documentation** and **certification** purposes, e.g.:
  - global end-to-end time, based on the maximum observed trace segment times combined to an overall bound
  - end-to-end time bounds for specific functions, depending on trace points
  - GANTT chart of task execution times extracted from trace data
  - total interrupt blocking time per trace segment New
  - time variance of each trace segment New
  - trace coverage
  - maximum possible (based on static program analysis) and maximum observed iteration counts for loops
- **TimeWeaver** supports **batch mode** execution and integration in **continuous integration** frameworks.
- On Tricore AURIX devices, **TimeWeaver** supports highly efficient **interactive MCDS tracing** via Infineon DAS.

Supported architectures and trace formats

- All PowerPC boards able to emit Nexus program trace messages (IEEE-ISTO 5001, class 2 or higher), e.g.: PowerPC QorIQ P204x/P30xx/P40xx/P50xx (e500mc core), PowerPC QorIQ T series (e5500/e6500 core), PowerPC Qorivva line MPC55xx/MPC56xx/MPC57xx (e200 core).
- ARM using cycle-accurate ETM traces, e.g.: Cortex-A53, Cortex-R5F.
- TriCore AUDO family (e.g. TC1796), TriCore AURIX (e.g. TC275), and TriCore AURIX 2nd Generation (e.g. Tc3xx).
- Lauterbach Trace32 BRANCHFLOW export trace New
- NEC/RENESAS V850 and RH850

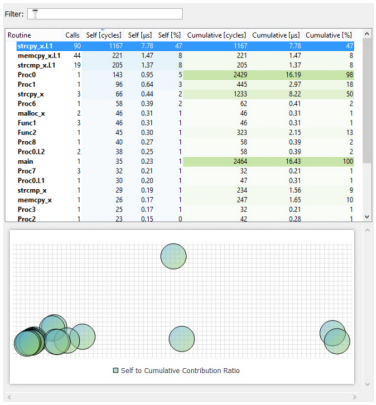
AbsInt

AbsInt Angewandte Informatik GmbH Science Park 1 D-66123 Saarbrücken Germany
phone: +49 681 383 60 0 fax: +49 681 383 60 20 info@absint.com www.absint.com

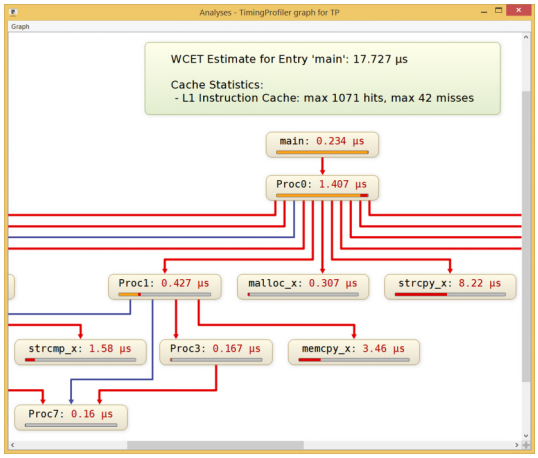
TimingProfiler

Monitoring Timing Behavior During Code Development

TimingProfiler helps developers identify application parts that are causing unsatisfactory execution times. It is ideally suited for constantly monitoring timing behavior during software development and in model-based development environments. **TimingProfiler** delivers results as soon as there is compiled code. It can be used early in the process when measurements on physical hardware are costly or not even possible.



Global contributions



Call graph with timing information

Usage of variables

Why do you need TimingProfiler?

- TimingProfiler helps address timing behavior continuously during development **from early stages on**.
- Developers can **immediately understand** the timing effects of their implementation decisions.
- TimingProfiler **visualizes** the call and control flow graph with timing information and displays relevant information about the executable.
- TimingProfiler computes worst-case execution time estimates for a slightly idealized model of the target processor. In contrast to aiT, it cannot derive guaranteed timing bounds, but efficiently computes **timing estimates**.
- **No access to physical hardware** and **no code instrumentation** are required.
- TimingProfiler automatically explores **all execution paths** in a program for all potential inputs.
- **No effort** is needed to set up and execute elaborate timing measurements.
- TimingProfiler can be **easily integrated** into the development process and used in **continuous test and integration** frameworks.
- Developers can **identify bottlenecks** early and **avoid late-stage integration problems**.

Supported processors and compilers

- TriCore/AURIX (Tasking/gcc/Diab)
- PowerPC (**CompCert**/Diab/gcc/GHS/CodeWarrior/DDC-1 Score)
- NEC/Renesas V850and RH850 (GHS/Diab/Renesas CS+)
- ARM (Cortex-M/Cortex-R) (**CompCert**/TI/ARM/gcc/GHS/ Tasking/Keil MDK-ARM/clang/Diab, GHS Ada)

For further targets, please contact us.

- dsPIC (Microchip)
- MCS251 (Keil)
- LEON2/LEON3/LEON4 (gcc/GNAT)
- RISC-V (**CompCert**/gcc)
- Nios II (gcc)
- MIPS32 (gcc)
- RL78 (IAR) New

AbsInt Angewandte Informatik GmbH Science Park 1 D-66123 Saarbrücken Germany
phone: +49 681 383 60 0 fax: +49 681 383 60 20 info@absint.com www.absint.com