manuscript No.
(will be inserted by the editor)

Lili Tan

The Worst Case Execution Time Tool Challenge 2006: Technical Report for the External Test

Abstract The first Worst Case Execution Time WCET Tool Challenge, performed in 2006, attempted to evaluate the state-of-the-art in timing analysis for real-time systems and to encourage research and activities in the WCET community. The challenge assesses academic and commercial WCET tools. It comprises two approaches: self-evaluation by the respective tool developers and external evaluation by an independent and neutral test person. During the evaluation period, the test person visited the tool developers to test the tools and to perform measurements on-site. This paper describes the procedure and the results of the external evaluation.

Keywords Timing Analysis · Worst Case Execution Time · WCET · Hard Real Time

1 Introduction

The Worst-Case Execution Time (WCET) refers to the maximum execution time of certain program executables on certain target processors [1]. WCETs are often used for schedulability analysis and timing assurance in real-time and embedded systems, especially in safety-critical hard real-time systems, like avionic flight control or automatic auto brake control systems. Generally, WCETs are derived from the prediction of the upper bounds of execution times of program tasks. Safe and precise predictions of these upper bounds are the tasks of WCET tools. Reductions in overestimation improve the precisions of prediction and indicate the quality of a WCET tool.

The first Worst Case Execution Time Tool Challenge invited all tool providers to submit their tools to evaluate the state-of-the-art in timing analysis for real-time systems to encourage WCET research and activities in the WCET community [2]. The working group selected a test framework and a set of benchmark programs. A neutral person was chosen to conduct an unbiased evaluation. This is necessary for the evaluation of WCET tools since an automatic evaluation is not fully realized, and human efforts are still required [1]. In particular, the evaluation of tool-usability requires an externally conducted test.

The author, Lili Tan from the University of Duisburg-Essen has been assigned by the WCET Tool Challenge Working Group (WWG) to do the external test. Previous to this challenge, she had gained experience with the evaluation of industry-strength WCET tools for avionic applications in a high-lift flap control system. In such a system, domain standards, together with model-based software development environments using Matlab/Simulink and SCADE for automatic code generation, need to be taken into consideration.

The external test was performed in cooperation with the WWG. It started on September 29th and ended on November 13th 2006. During this time period, the author tested all of the tools and visited all tool providers except one who registered late. The test and its results were presented on November 17th at the ISoLA 2006 conference in Cyprus [3]. The results of the self-evaluation part of the WCET Tool Challenge are presented in [4].

This paper describes the external tests performed in the challenge and the test results obtained. The remainder of the paper is organized as follows. The test framework is introduced in Section 2. The implementation of the external test work is described in Section 3. The test results are presented and the usability of tools is assessed in Section 4. Section 5 concludes.

This work was supported by the ARTIST2 European Network of Excellence.

L. Tan
ICB/Computer Science, University of Duisburg-Essen
Tel.: +49 -201-183 2340
Fax: +49 -201-183 2149
E-mail: lili.tan@icb.uni-due.de

2 Test Framework

This section gives an overview of the general framework under which the submitted tools were evaluated.

2.1 WCET Tools under Test, Variants, and Updates

All of the WCET tools that entered the challenge have been registered for the external test. We briefly describe these tools in the sequence of registration. aiT WCET Analyzer (aiT) is a software product of the company AbsInt Angewandte Informatik GmbH [5]. The technology used in aiT was initially developed at Saarland University. Since 1998, AbsInt and Saarland University have been developing aiT jointly. The following variants participated: aiT for the Infineon C16X processor and with the Tasking compiler, aiT for ARM7 (TMS470) and the TI v2.17 compiler, and aiT for PowerPC MPC565 and the Diab v5.3 compiler.

Bound-T is a software product of the company Tidorum Ltd, which provided three test variants: Bound-T for Renesas H8/300 and the GCC compiler as well as the IAR compiler, Bound-T for SPARC V7/V8 (ERC32) and the BCC compiler [6].

The Swedish Worst Case Execution Time Tool (abbreviation: SWEET), is a research prototype provided by Mälardalen University [7]. SWEET supports the target processor ARM9 and the CtoNIC compiler.

MTime is a research prototype from the Real-Time Systems Group at Vienna University of Technology (TU-Vienna). MTime supports processor Motorola HCS12 with the COSMIC compiler [8] .

Chronos is a research prototype and GNU open source software developed by the National University of Singapore. Chronos is designed for SimpleScalar and uses the GCC compiler. It allows three types of configurations in the processor, i.e. Simple in-order, Complex in-order, and Complex out-of-order [9].

Tool developers were allowed to develop their tools and to submit their new improved software updates during the challenge. All updates and test variants are listed in Table 1.

2.2 Programs under Test

The WWG selected a set of 15 programs from the Mälardalen WCET benchmark and two PapaBench benchmarks as the basis for the evaluation [2], [7], and [10]. The programs are available in C source code. The C programs were compiled by the participants with the respective compilers.

The selected benchmarks consist of programs with the following features: loops, nested loops, and recursion. Details about the complexity of the benchmark test programs is described in [4] and [7].

2.3 Test Scope

The WWG decided that the tools should be tested with respect to three aspects, namely flow analysis, required user interaction, and performance [2]. The test was also expected to run with a three-round procedure, i.e. without manual annotation, minimal set of annotations, and finally an optimal set of annotations to improve the WCET quality to the best level [2].

- By means of the flow analysis, the ability of the tools in analyzing timing features of software programs are discussed in [4].
- In the first-round run without manual annotation, the percentage of input programs that a tool can analyze and deliver test results are counted and the automation rates are derived.
- In the final-round optimal run, as the WCET quality improves to the best level, the estimated WCETs are noted down and the tightness in reduction of overestimation is derived, by reference to the measured execution times. Furthermore, the percentage of input programs that a tool can handle are counted.
- Between the first-round and the final-round run, the test aspects concerning required user interaction and performance are evaluated and the usability is assessed based on the ISO 9421 standard.
- Prediction of the timing behavior of realistic processors requires that tools can handle performance-enhancing features such as caches, pipelines, and branch prediction. Hence, the complexity of supported target processors is taken into account in the evaluation.

2.4 Test Computer Specification

The computer used for running the external test is a laptop with an AMD Mobile Sempron TM Processor 2800+, 1.60 GHz, and 448 MB RAM. Three other PCs (AMD Athlon TM 2500+, 1.83 GHz, 992 MB of RAM), two with Windows XP and one with Linux platform at the University of Duisburg-Essen, have also been selected for executing Bound-T and for accessing Chronos by remote server.

3 Test Implementation

Based on the test framework, a schedule has been made to visit the tool developers and to test their tools. The tests were performed starting with aiT and continuing with Bound-T, MTime, SWEET, and Chronos. Roughly one week was spent on each tool (including traveling). Table 2 shows the final test schedule.

3.1 aiT

Test Initiation. The visit at AbsInt in Saarbrücken and Saarland University from October 4th to 6th. The visit included a training course, testing of the tools, feedback, discussions, and measurements on evaluation boards.

The training course covered many technical issues about aiT in detail. During the test, aiT for C16X, aiT for ARM7, and aiT for MPC565 were installed. All benchmark binary files were received and tested. Various additional topics were discussed with several specialists from AbsInt and the University of Saarland., e.g., annotations of floating point's implementations on integer machines, and annotation for the C runtime library.

The feedback on the usage of aiT given by the author was discussed. Additionally, we discussed the automatic analysis of code coming from model-based software development environments, such as SCADE or Matlab/ Simulink.

With support by AbsInt, the execution time measurements were performed on October 5th. All of the three target processors were measured with an ISYSTEMS ILA 128 logic analyzer. The measured values (see Table 6) and the estimated WCET were compared. Some of the measurements have been done several times. After the visit, several test results were further elaborated upon.

Working with aiT. The typical working pattern with aiT includes: invoke aiT either with the aiT icon or through command line, follow the hints and links of the GUI, provide annotations, take a look at aiSee, and execute the WCET calculation by mouse-click in the aiT GUI.

After specifying the CPU clock value for each target processor, aiT delivered WCET outputs for 45 test programs automatically without annotation (see Table 3. 54% of all 84 programs).

For the reduction of overestimation of the WCET results, human efforts are required. aiT assists the development of annotations in the following ways:

1. The aiT GUI gives hints for annotations guiding users throughout the test. Hints come along with links into the code (binary and source). This allows users to concentrate on WCET and avoids tedious actions, like invoking additional programs and changing interfaces.

2. aiT's visualization helps users to understand the timing behavior of programs. For the visualization, annotated call- and control-flow graphs can be dynamically explored with the help of the aiSee graph browser. Figure 1 illustrates a control-flow graph from the *edn* benchmark program by aiT for MPC565. Additional information like loop bounds can be shown on demand.

3. aiT detects loop bounds, counterchecks annotations with analysis results, and warns users if invalid annotations are found. Cases about warning users of their invalid annotations are found. To demonstrate this feature, we select the same example for the test program *edn* as mentioned above in Figure 1.

aiT for MPC565 detected for the *edn* benchmark that the loop shown in Figure 1 iterates 23 times (#23). If users give a wrong annotation like:

"loop "main" + 1 loop max 20"

in the annotation file *edn.ais*, then aiT for MPC565 delivers the warning message:

"powerdaan: Warning: In *edn.c*, line 244:
At address 0x17dc in routine 'main.L1':
loop seems to have more iterations than specified:
at least 22 instead of 21."

Figure 2 shows this message in red color in the aiT GUI Messages area. The annotation, "loop "main" + 1 loop max 20" was wrongly given in the annotation file. The link showing the address 0x17dc in the message window leads users directly to the position in the binary code, which is shown in the left side of the screenshot. It also directs to line 244 of the respective C source code file shown on the right side of the figure. These features help to avoid wrong annotations made by users and limit the chance to produce errors in WCET analysis.

4. aiT collects all settings of annotations, predicted results, and WCET details automatically into one report file. It is not necessary to open a text editor to copy, paste, and save the run messages.

5. aiT allows users to specify paths for executable files, setting files, annotation files, predicted results, and WCET graphs in GUI and in aiT project files. To re-run the programs, users do not need to keep the command line in mind, do not need to look up manuals, and do not need to specify the paths. aiT project works by mouse-click.

6. The analysis time of aiT for benchmark test programs is generally in seconds. The only exception encountered was by analyzing the test program *matmult* with optimal annotation for aiT for ARM7. It was not possible to analyze the program with these annotations on the test computer due to limited memory capacity. No such problem has been encountered on a larger computer reported by [4].

3.2 Bound-T

Test Initiation. The first installation of Bound-T and tests with example programs was finished and tests of benchmark test programs for H8/300 in *.coff format were performed before the test trip.

On October 26th, a Bound-T meeting was held in Västerås. The Bound-T developer introduced important technical issues about Bound-T. Hints concerning for assertions were explained in detail with an example. Further tests with assertions were continued latter on.

Working with Bound-T. The general working pattern with Bound-T includes: invoke Bound-T through the command line, follow the hints in the run messages, inspect source code programs and executable programs for loop bounds if necessary, look up hints in the manuals for developing annotations, execute the WCET calculation through the command line, and save the WCET output value for later use.

After invoking the Bound-T command line, Bound-T analyzed 13 test programs and delivered their WCET results without manual annotation (26% of 51 programs, see Table 3).

Bound-T displays run messages on the screen and an output graph. Both of the run messages on the screen and the output graph assist users for developing annotations. Figure 3 is an example of a screenshot of the Bound-T interface. In the figure, the analysis results for the benchmark program *cnt* in *.coff format are presented for the target processor H8/300. The loop bounds detected by Bound-T are listed in the run messages together with the function names, the C source file, and the line numbers. The overall WCET value for the main function of the program *cnt*, accompanied by WCET values of the functions called, is also listed in the run messages. Figure 4 shows the graphical output for the *cnt* program. The overall WCET value and the detailed calculation are illustrated graphically corresponding to the run message in the screenshot.

According to the run messages and output graph, users can check the C program file or the executable code if possible, for developing assertions.

On the test computer, we encountered some problems with the Windows version of Bound-T. No such problem was reported by users of the Linux version. Bound-T could handle 13 out of 17 test programs (76.5%). Bound-T failed on 4 programs. It was not able to handle recursion in the test programs *recursion* and nested loops in the *janne_complex* and *statemate*. It returned an error message about "irreducible flow-graph" while analyzing the $duff$ test program, which contained multiple entry points in a loop.

3.3 MTime

The visit to the Real-Time Systems Group at the TU-Vienna was from October 11th to 12th. MTime developers introduced the background, the architecture, and applications of their tool.

MTime is designed for WCET-oriented automatic test data generation with the functionality of model checking. It uses so-called *automatic parametrical CFG partition* to reduce complexity and applies segmentation to minimize the overall number of tests. Instead of modeling processors, it performs measurements of test data generated. The results of measurements are then used to calculate the final WCET.

MTime developers provided us with a PC and example test programs that they develop. The usability for the examples is similar to the other WCET research prototypes in the challenge. In addition, we installed and tested MTime on the test laptop. The installation procedure of MTime was rather straightforward as with other WCET tools tested in the challenge.

Due to the fact that MTime did not support function calls at that time, it was not possible to produce any result for the WCET Challenge benchmark programs.

3.4 SWEET

Test Initiation. A SWEET meeting was held at Mälardalen University on October 25th.

The SWEET developers introduced the architecture, the flow analysis, and the use of SWEET. SWEET was installed on the test computer, and some tests were performed during the visit. The programs under test included the 15 Mälardalen WCET benchmarks in intermediate NIC code format, which compiled by SWEET developers with a research compiler. The two PapaBench benchmark test programs were not included in the deliverable list and therefore no test was performed for them, because SWEET was not able to analyze the PapaBench programs.

After the visit, the 15 test programs, the four modes for each test program in flow analysis, every update of SWEET during the challenge were tested and the test results were also updated accordingly.

Working with SWEET. The typical working pattern with SWEET is: start SWEET in the Cygwin environment, specify a mode in the command line, run a test program, and find the WCET result in the run message on the monitor screen. By the use of multi-path mode with SWEET, users have to examine the suitable application's requirements.

Without manual annotation, SWEET analyze automatically all 15 Mälardalen benchmark test programs, using the simple path basic mode and simple path advanced mode defined by SWEET. The WCET analysis results delivered by SWEET include run messages in monitor and DOT graph files. Figure 5 is the graphical analysis result for the test program *janne_complex*. It illustrates the hierarchical structure of loops and the nested relation between inter and outer loops as detected by SWEET.

In basic mode, only simple loop bounds are calculated by the flow analysis, while in advanced mode all types of loop bounds as well as infeasible paths are calculated. The test results for the single path basic mode and single path advanced mode are presented in Table 3. By using the advanced mode, reduction in estimated WCETs can be observed in comparison with the test results (see Table 3 and Figure 6). This is a result of advanced loop bound calculation and infeasible path detection to reduce the estimated WCETs and the overestimation.

Some programs can be assigned with multi-path mode and run with SWEET. The suitable programs for the multi-path mode are *crc*, *edn*, *insertsort*, *janne_complex*, *ns*, and *nsichneu*. The test results are illustrated in Figure 6 and Table 4.

SWEET delivered 15 WCET test results for all of the 15 Mälardalen WCET benchmarks without annotation (see Table 3). In total, 88.2% of the test programs were analyzed (15 of 17). The test for program *nsichneu* in advanced mode could not deliver a result due to memory limitations of the test laptop. No such problem has encountered by other computer [4].

Advanced application with SWEET might benefit from a more detailed description of the syntax and semantics of annotation files, as well as automatical selection of appropriate modes where they are suitable.

3.5 Chronos

Test Initiation. The test of Chronos was performed in the last week of October and at the beginning of November 2006. Basically, Chronos can be used in two possible manners: command line and GUI. Chronos can also work with two different integer linear programming solvers. One is an open source program, "lp_solve" [11], which is also used by all other WCET tools. The other one is the commercial product from ILOG, the CPLEX LP solver [12]. Recommended by the Chronos developers, we used the CPLEX and tested Chronos on a remote server because of the licensing issues.

Working with Chronos. Working remotely through the internet and testing in different operating system platforms and with different runtimes, several non-WCET related implementation problems occurred and were solved within the test duration. In the remaining part of this section, we present the test procedure under which we achieved the test results.

The software X-Win32 and StarNetSSH was used to access the remote server in Singapore. Because of the extremely slow response resulting from the Java runtime in remote, Chronos GUI could not be used. We used command line interface instead. The benchmark programs were compiled through remote access by using GCC 2.2.2.3 and analyzed by the Chronos WCET analysis kernel. The estimated WCET values (see Table 3 and 5) were calculated by CPLEX. The simulated WCET values were obtained by using command line for simulation, a utility provided by the SimpleScalar simulator (see Table 6). The three processor configurations for both the estimated WCETs and simulated execution times require the corresponding configuration setting files, i.e., the Simple in-order, Complex in-order, and Complex out-of-order.

Chronos could handle 13 out of 17 benchmark test programs (76.5%) for all of the three processor configurations. Reasons why the test failed with 4 programs were: Chronos was not able to handle *recursion*; Chronos did not cope with non-deterministic control transfer in *cover* and could not handle *duff*, which contained multiple entry points in a loop. The *autopilot*, one of the PapaBench test programs, could not be analyzed by Chronos.

Although the external test for Chronos was remote, the tests for each benchmark executed averagely quickly using CPLEX.

4 Test Results

We present test results and interpret them in this section.

Generally, the test results of the different WCET tools are not directly comparable, as the tools are designed for different target processors. By analyzing different data, we gained a picture of each WCET tool and their main concepts. Therefore, we present those test results that best describe the characteristics of the various WCET tools, followed by a summary of the mutual features, from the aspect of functional and service quality.

The results presented consist of four parts corresponding to the three-round test procedure and a summary of general features of each tool:

- Test results without annotations
- Test results with annotation and the precision
- Usability assessment during developing annotations in the test procedure
- Summary of functional and service quality

4.1 Test Results without Annotations

The WCET values that were estimated by each tool in the first test round using no annotations are summarized in Table 3. They are the WCET prediction of different benchmark test programs on the supported target processors. The concrete and absolute WCET values in the table are not comparable as they are targeted for different processors. The numbers of benchmark test tasks that each WCET tool processed among the given benchmark test programs indicate the degree of automation of a tool.

4.2 Test Results with Annotations

After providing human interaction and annotations, the test results are improved from two aspects. From the aspect of quantity, the number of analyzable test programs increase; from the aspect of quality, the precision increases and the overestimation is reduced.

Servability of WCET Tools. The number of different programs that a tool can handle indicates the ability of a tool in analyzing input programs. In an effort to make the incomparable test results comparable, we decided to avoid comparing the concrete output values of each tool, but to treat each output abstractly as a success of the tool and use "1" as an indicator to represent that a result is produced. Table 4 presents these abstract service results. In addition, errors encountered - test computer related or not - are also presented in the table.

Estimated WCET with Annotations. Table 5 gives the WCET values calculated by aiT and Chronos with annotations. These WCET values will be used for the calculation of the precision of tools. Besides aiT and Chronos, no measurement value could be obtained from other tools in this challenge. Therefore, the WCET values calculated and their precision could only be evaluated for aiT and Chronos.

Measured and Simulated WCET. Table 6 gives the execution time of WCET measurements for the programs considered by aiT and Chronos respectively.

For aiT, the execution times were measured on real hardware using a logic analyzer. For Chronos, the SimpleScalar simulator has been used. No measurement of execution time has been provided by the other tools.

WCET Tightness. The WCET tightness in Table 7 is calculated from the the values of Table 5 and 6 for aiT and Chronos. The spectrum of tightness for both tools is also shown in Figure 7 and 8.

4.3 Usability Assessment

The ease to use a WCET tool contributes to the successful results of the WCET analysis. Defined by ISO 9241, usability refers to the extent to which a product, e.g., a software tool, can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use. The usability assessment for WCET tools in the challenge is based on the successes of WCET results achieved in analyzing the benchmark test programs, because the time and efforts spent for each tool are generally the same.

Our considerations for usability include: How many hints given by a tool are helpful in conducting WCET analysis for the selected benchmark test programs, rather than just reviewing users' manual. Tool developers should not expect users to read their document from the first to the last page. Users generally expect appropriate tips and hints appearing automatically at the appropriate moment when they use the tools.

The main problem detracting from the usability of a tool is that it lacks users-centric usability design by some tools in the up-to-know design status, e.g., frequency of switching software interfaces to look up manuals and to open different programs in order to develop assertions or to create graphs.

Some other factors, which come from algorithm and functionality design rather than usability design of a WCET tool, do really affect the assessment of usability. They detract the effectiveness, efficiency, and satisfaction from achieving the goal to successful analysis of the benchmark programs within the test duration. These factors include:

- Frequency of errors encountered
- Differences in derivation of analysis time for different test programs in a single tool
- Long analysis time combined with any one of the above factors.

The usability experienced by the author during the three-round test is summarized in Table 8. From the author's experience, aiT provides the best usability.

4.4 Summary of Functional and Service Quality

Table 9 gives an overview of the important indicators from test results, namely tightness, service rate, automation rate, and types of processors supported by the tools.

Tightness is a black box indicator for the functional quality of a WCET tool in that people are not necessary to indulge in analyzing the individual correctness of algorithms adopted by each tool. The best are given by aiT. aiT delivers the tightest WCET. The overestimation is less than 8%.

The service rate of a tool indicates the ratio of the number of input benchmarks that could be handled and the total number of the inputs. Failures encountered that solely result from the limits of the test computer are excluded from the calculation of service numbers and service rate. The best results come from aiT; it gets 100%.

The automation rate is calculated as the ratio of the number of automatically analyzed programs and the number of all analyzable ones. Table 9 indicates that full automation has not been reached. The best results come from SWEET; it gets up to 88%.

The state-of-the-art WCET tools support different types of processors, which range from simple, through medium to very complex. aiT supports very complex processors like the MPC 565.

5 Conclusion

The external test for the WCET Tool challenge was accomplished.

All of the entered tools have presented their strengths of different aspects in analyzing WCET problems: aiT is able to handle every kind of benchmark and every test program that was tested in the Challenge. aiT is able to support WCET analysis even for complex processors. aiT and Chronos are able to demonstrate the precision of their WCET prediction. Both commercial tools aiT and Bound-T are able to handle the two fly-by-wire PapaBench test programs. SWEET is able to automatically analyze 88% benchmark test programs. The analysis time of Chronos was very short when using CPLEX. aiT demonstrates its leading position through all its features, which contribute to its position as an industry-strength tool satisfying the requirements from industry as posed by EADS Airbus and proven by the accomplishment in various projects.

The external test trips provided both the tools developers and the test person with the opportunity to discuss directly the feedback on using the tools. Feedback on the tools' usability and the possible software faults has been taken into account by the relative developers for the further development of their tools. All of the tool developers have provided their best support and cooperation for the external test.

This challenge has encouraged WCET research and activities in the WCET community. During the challenge, many developers were engaged in developing and improving their tools further. A total of 25 updated versions of the software were submitted and their latest releases have demonstrated the great improvements on the WCET results from several aspects.

At the ISoLA 2006 conference, the challenge has also caught the attention of more WCET developers. They showed their interest to participate in the next challenge. All of the tool developers want to enhance their tools for more intensive industrial applications.

This challenge reveals the success of the "Saarland Model" presented by Saarland University and AbsInt. The cooperation between scientific research and industry gives rise to a synergistic effect on all participants in the cycle of education, research, and industry in our society [13].

We anticipate positive effects of the challenge on the WCET community in research tool development, and industry applications.

Acknowledgements Thanks to the support by Klaus Echtle and Christina Braun at the University of Duisburg-Essen, ICB/Computer Science. Thanks to the support by Jan Gustafsson, Reinhard Wilhelm, and the WCET Tool Challenge Working Group. Thanks to the support by WCET tool developers during the WCET Tool Challenge 2006. They are Christian Ferdinand, Martin Sicks, Steffen Wiegratz, Florian Martin, Reinhold Heckmann, and Christoph Cullmann of AbsInt Angewandte Informatik GmbH, Stefan Thesing and Björn Wachter of Saarland University, Niklas Holsti of Tidorum Ltd., Björn Lisper, Jan Gustafsson, Stefan Byqde of Mälardalen University, Raimund Kirner, Ingmar Wenzel, and Berhard Rieder of TU-Vienna, Tulika Mitra, Abhik Roychoudhury, Vivy Suhendra, and Liangyun of National University of Singapore.

References

1. Wilhelm, R. et al.: The Worst Case Execution Time Problem -Overview of Methods and Survey of Tools. ACM Transaction on Programming Languages and Systems @20YY ACM 0164-0925/20YY/0500-00001, Pages 1-47.
2. WCET Tool Challenge 2006. http://www.idt.mdh.se/personal/jgn/challenge/, November 2006.
3. 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA20006), http://sttt.cs.uni-dortmund.de/isola2006/?id=program, 2006.
4. Gustafsson, J.: WCET Tool Challenge 2006. In Proceeding 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA2006), Cyprus, November 2006.
5. aiT WCET Analyzer (aiT), AbsInt. http://absint.com
6. Bound-T, Tidorum. http://www.tidorum.fi/bound-t.
7. MTime, Vienna real-time systems group. http://www.vmars.tuwientuwien.ac.at.
8. SWEET, Mälardalen University. http://www.mrtc.mdh.se/projects/wcet.
9. Chronos, National University of Singapore. http://www.comp. .nus.edu.sg/ rpembed/chronos
10. PapaBench, IRIT, http://www.irit.fr/recherches/ARCHI/MARCH/rubrique.php3?id_rubrique=97.
11. Berkelaar, M.: lp_solve, ftp://ftp.es.ele.tue.nl/pub/lp_solve.
12. CPLEX, ILOG, http://www.ilog.com/
13. Tan, L., Suhl L.: Scenario and Authoring Tool for Web-based Case Study Education in Operations Research/Management Science. In Proceeding World Association for Case Method Research and Application (WACRA'02), Mannheim, July 2002.

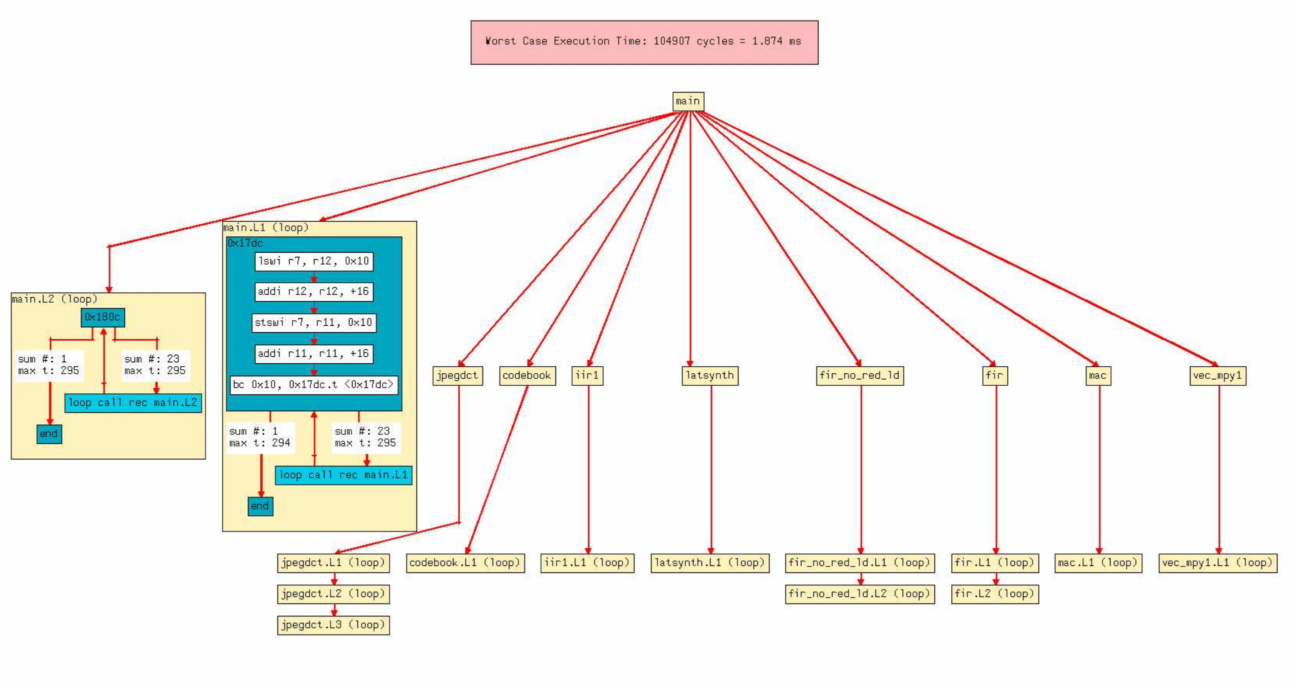
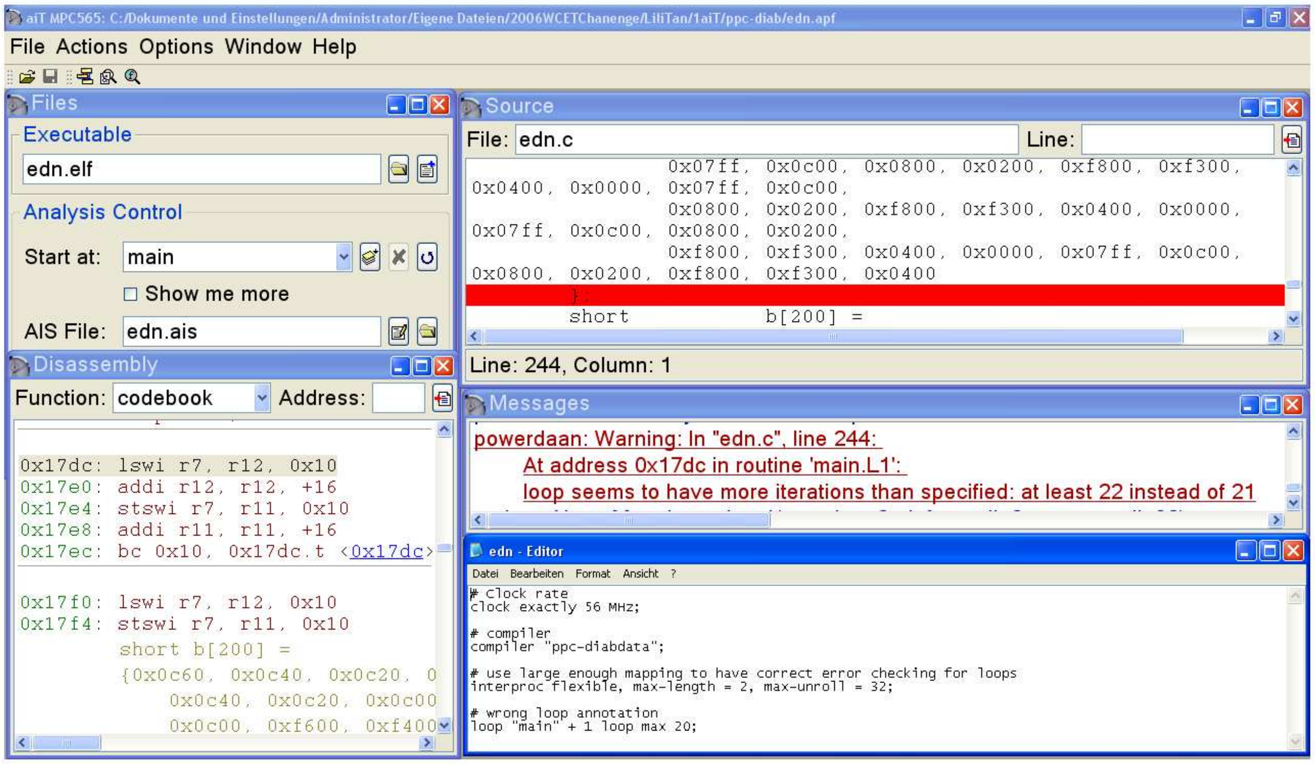


Fig. 1 Screenshot of aiT's visualization tool aiSee. It shows a combined function call and control flow graph. The graph contains timing details. The graph can interactively be explored within aiSee.


aiT MPC565: C:/Dokumente und Einstellungen/Administrator/Eigene Dateien/2006WCETChanenge/LiliTan/1aiT/ppc-diab/edn.apf
File Actions Options Window Help
Files
Executable
Analysis Control
Disassembly
Source
Messages
edn - Editor
Datei Bearbeiten Format Ansicht ?

Executable: edn.elf

Start at: main

☐ Show me more

AIS File: edn.ais

Function: codebook Address:

```
0x17dc: lswi r7, r12, 0x10
0x17e0: addi r12, r12, +16
0x17e4: stswi r7, r11, 0x10
0x17e8: addi r11, r11, +16
0x17ec: bc 0x10, 0x17dc.t <0x17dc>

0x17f0: lswi r7, r12, 0x10
0x17f4: stswi r7, r11, 0x10
        short b[200] =
        {0x0c60, 0x0c40, 0x0c20, 0
            0x0c40, 0x0c20, 0x0c00
            0x0c00, 0xf600, 0xf400
```

File: edn.c Line:

```
                0x07ff, 0x0c00, 0x0800, 0x0200, 0xf800, 0xf300,
0x0400, 0x0000, 0x07ff, 0x0c00,
                0x0800, 0x0200, 0xf800, 0xf300, 0x0400, 0x0000,
0x07ff, 0x0c00, 0x0800, 0x0200,
                0xf800, 0xf300, 0x0400, 0x0000, 0x07ff, 0x0c00,
0x0800, 0x0200, 0xf800, 0xf300, 0x0400
        };
        short           b[200] =
```

Line: 244, Column: 1

powerdaan: Warning: In "edn.c", line 244:
At address 0x17dc in routine 'main.L1':
loop seems to have more iterations than specified: at least 22 instead of 21

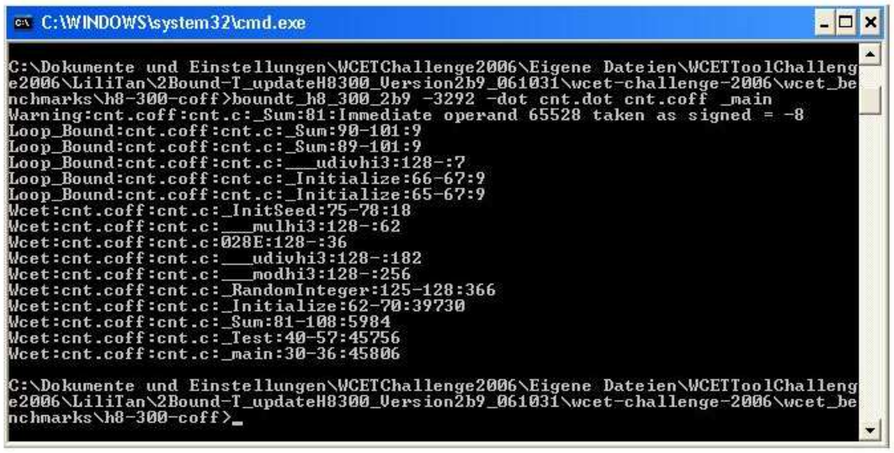
```
# Clock rate
clock exactly 56 MHz;

# compiler
compiler "ppc-diabdata";

# use large enough mapping to have correct error checking for loops
interproc flexible, max-length = 2, max-unroll = 32;

# wrong loop annotation
loop "main" + 1 loop max 20;
```

Fig. 2 Screenshot of the aiT GUI. It displays source code and binary code. It detects loop bounds and warns of possibly wrong annotations (see Message window). This helps aiT users to avoid errors.



```
C:\Dokumente und Einstellungen\WCETChallenge2006\Eigene Dateien\WCETToolChalleng
e2006\LiliTan\2Bound-T_updateH8300_Version2b9_061031\wcet-challenge-2006\wcet_be
nchmarks\h8-300-coff>boundt_h8_300_2b9 -3292 -dot cnt.dot cnt.coff _main
Warning:cnt.coff:cnt.c:_Sum:81:Immediate operand 65528 taken as signed = -8
Loop_Bound:cnt.coff:cnt.c:_Sum:90-101:9
Loop_Bound:cnt.coff:cnt.c:_Sum:89-101:9
Loop_Bound:cnt.coff:cnt.c:___udivhi3:128-:7
Loop_Bound:cnt.coff:cnt.c:_Initialize:66-67:9
Loop_Bound:cnt.coff:cnt.c:_Initialize:65-67:9
Wcet:cnt.coff:cnt.c:_InitSeed:75-78:18
Wcet:cnt.coff:cnt.c:___mulhi3:128-:62
Wcet:cnt.coff:cnt.c:028E:128-:36
Wcet:cnt.coff:cnt.c:___udivhi3:128-:182
Wcet:cnt.coff:cnt.c:___modhi3:128-:256
Wcet:cnt.coff:cnt.c:_RandomInteger:125-128:366
Wcet:cnt.coff:cnt.c:_Initialize:62-70:39730
Wcet:cnt.coff:cnt.c:_Sum:81-108:5984
Wcet:cnt.coff:cnt.c:_Test:40-57:45756
Wcet:cnt.coff:cnt.c:_main:30-36:45806

C:\Dokumente und Einstellungen\WCETChallenge2006\Eigene Dateien\WCETToolChalleng
e2006\LiliTan\2Bound-T_updateH8300_Version2b9_061031\wcet-challenge-2006\wcet_be
nchmarks\h8-300-coff>_
```

Fig. 3 Screenshot of the Bound-T user interface. It shows the command line used to invoke Bound-T, the detected loop bounds, and the WCET values for the functions of the analyzed program.

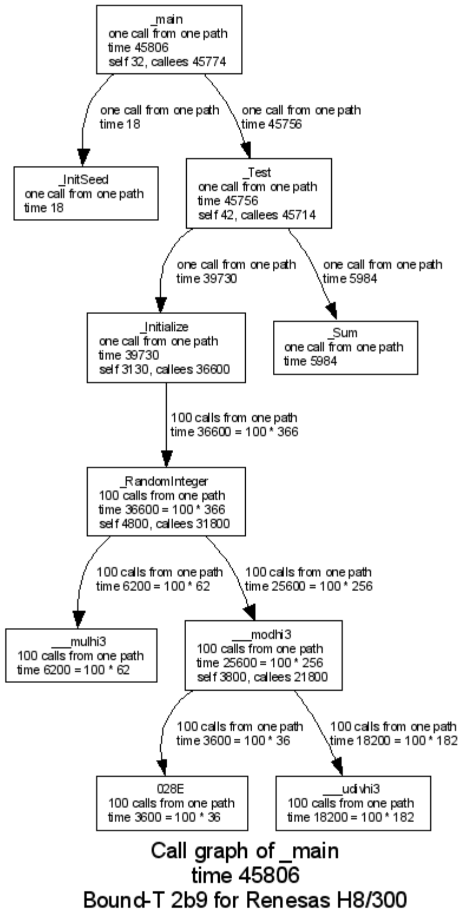


Fig. 4 Bound-T call graph. It displays the predicted WCET values for functions, the loop bounds, and the overall WCET.

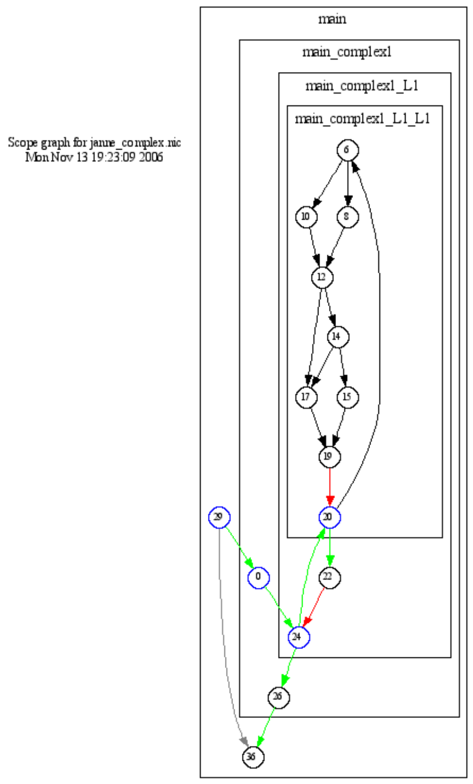


Fig. 5 SWEET scope graph. It displays the hierarchical structure and the nested relation between inter and outer loops that detected by SWEET.

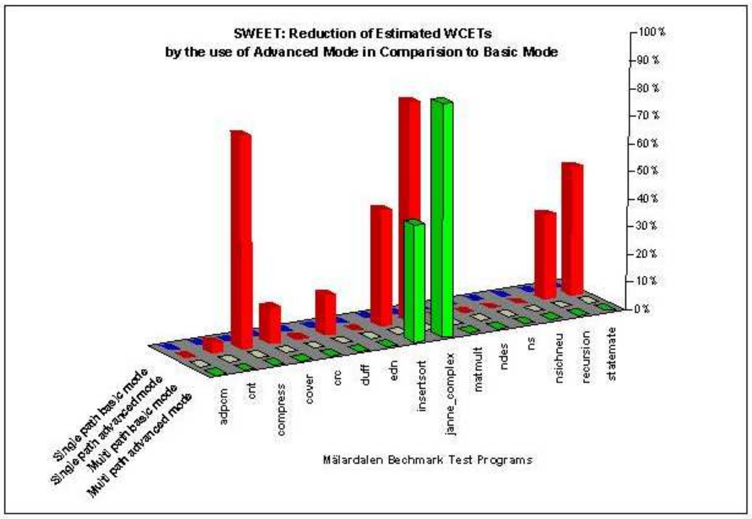


Fig. 6 SWEET: Reduction of Estimated WCETs by use of Advanced Mode in Comparison to Basic Mode. By use of advanced mode, advanced loop bounds, like triangle loops and infeasible paths are to be detected. As a result, the reduction of estimated WCETs for advanced mode is expected by comparison to basic mode.

Table 1 WCET Tools, Supported Target Processors and Compilers

Nr	Tool (Processor, Compiler)	Affiliation of Tool Developer
V1	aiT_C16X (Infineon C16x, Tasking v8.5)	AbsInt Angewandte Informatik GmbH, Germany
V2	aiT_ARM7(ARM7, TI v2.17)	AbsInt Angewandte Informatik GmbH, Germany
V3	aiT_MPC565 (PPC MPC565,Diab v5.3.1.0)	AbsInt Angewandte Informatik GmbH, Germany
V4	Bound-T H8/300_GCC (Renesas H8_300,GCC)	Tidorum Ltd., Finland
V5	Bound-T H8/300_IAR (Renesas H8_300,IAR)	Tidorum Ltd., Finland
V6	Bound-T SPARC (ERC32,GCC-based BCC)	Tidorum Ltd., Finland
V7	MTime (Motorola HCS12, COSMIC)	TU-Vienna, Vienna
V8	SWEET (ARM9, CtoNIC)	Mälardalen University, Västerås
V9	Chronos (SimpleScalar Simple in-order, GCC)	National University of Singapore, Singapore
V10	Chronos (SimpleScalar Complex in-order, GCC)	National University of Singapore, Singapore
V11	Chronos (SimpleScalar Complex out-of-order, GCC)	National University of Singapore, Singapore

Table 2 External Test Working Table

Nr.	Duration	Test Activities	Place
0	2006-09-29	Start right after the organizational decision made	Essen, Germany
1	2006-10-02 to 06	Test aiT, aiT trip (2006-10-03 to 06)	Saarbrücken, Germany
2	2006-10-09 to 13	MTime trip (2006 -10-10 to 13)	Vienna, Austria
3	2006-10-16 to 20	Test aiT, Test Bound-T	Essen, Germany
4	2006-10-23 to 27	Test Bound-T, Bound-T and SWEET trip (2006-10-24 to 28)	Västerås, Sweden
5	2006-10-30 to 11-03	Test Chronos, Test Bound-T, Test SWEET	Essen, Germany
6	2006-11-06 to 13	Report drafting, Test SWEET	Essen, Germany
7	2006-11-14 to 20	ISoLA conference trip, Test report on 2006-11-17	Paphos, Cyprus

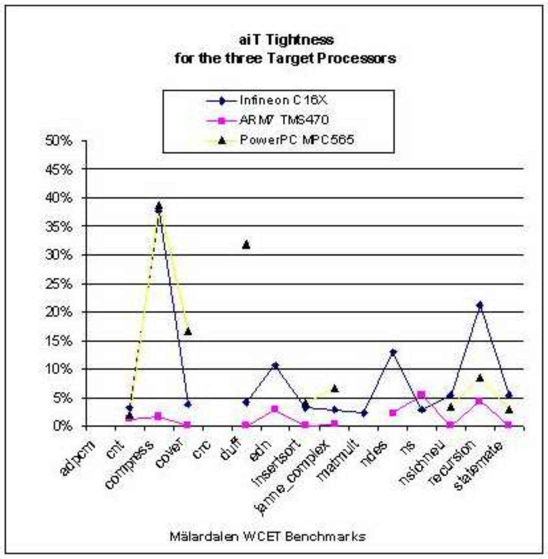


Fig. 7 aiT tightness for the three target processors Infineon C16X, ARM7 (TMS470), and Power PC MPC565. The average aiT's overestimation is less than 8 %.

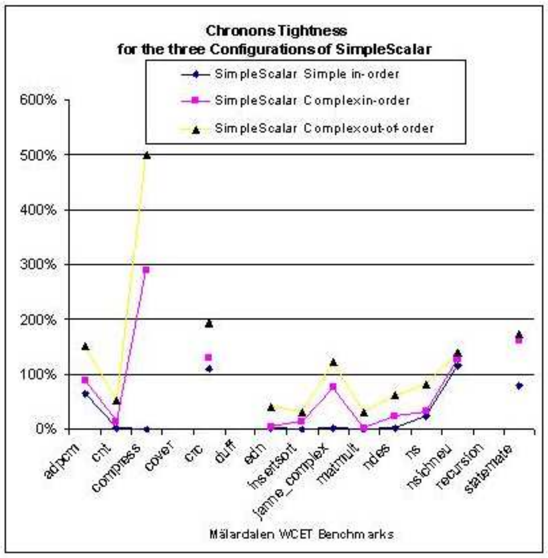


Fig. 8 Chronos tightness for the three SimpleScalar configurations: Simple in-order, Complex in-order, and Complex out-of-order. The average overestimation of Chronos is less than 90% for the three processor configurations.

Table 3 Estimated WCET Values* (in Cycles) without Annotations

Nr.	Benchmarks	aiT V1	aiT V2	aiT V3	Bound-T V4	Bound-T V5	Bound-T V6
1	adpcm						
2	cnt	32812	26572	7576	45806	78982	
3	compress						
4	cover	19459	6780	5451		10250	180
5	crc			107278	164118	268657	
6	duff						
7	edn		307889	104907			
8	insertsorts						
9	janne_complex						
10	matmult	1562815	523599	237736	1506520	3282132	
11	ndes	816337	194448	1944845		712454	4214
12	ns	238414	38043	34361	20976	256960	7097
13	nsichneu		41678	21362			
14	recursion						
15	statemate						
16	Fbw1: fly-by-wire test_ppm_task		9875	1242			
	Fbw2: fly-by-wire send_data_to_autopilot_task		3197	331			
	Fbw3: fly-by-wire check_mega128_values_task		4092	437			
	Fbw4: fly-by-wire servo_transmit	2390	1909	1249			
	Fbw5: fly-by-wire check_failsafe_task		4058	432			
17	Au1: autopilot radio_control_task		15972	2247			
	Au2: autopilot stabilisation_task		4239	340			
	Au3: autopilot link_fbw_send	170	144	81			
	Au4: autopilot receive_gps_data_tasks						
	Au5: autopilot navigation_task						
	Au6: autopilot altitude_control_task		915	95			
	Au7: autopilot climb_control_task		4129	247			
	Au8: autopilot reporting_task	8286	11172	4464			
18	Benchmarks analyzed	8	18	19	4	6	3
19	Total	28	28	28	17	17	17

Nr.	Benchmarks	SWEET V8[1]	SWEET V8[2]	Chronos V9	Chronos V10	Chronos V11
1	adpcm	2165650	2162122			
2	cnt	36719	35319	4896	6438	5401
3	compress	206480	49896			
4	cover	73128	63563			
5	crc	834159	830278			
6	duff	5525	4720			
7	edn	1425085	1425085	89401	113612	89030
8	insertsorts	31163	18167			
9	janne_complex	12039	2523	189	800	789
10	matmult	2532706	2532706	186903	191615	119526
11	ndes	795425	795425			
12	ns	130733	130631			
13	nsichneu	119707	N/A3			
14	recursion	29079	20033			
15	statemate	15964	8451			
16	Fbw1: fly-by-wire test_ppm_task					
	Fbw2: fly-by-wire send_data_to_autopilot_task					
	Fbw3: fly-by-wire check_mega128_values_task					
	Fbw4: fly-by-wire servo_transmit					
	Fbw5: fly-by-wire check_failsafe_task					
17	Au1: autopilot radio_control_task					
	Au2: autopilot stabilisation_task					
	Au3: autopilot link_fbw_send					
	Au4: autopilot receive_gps_data_task					
	Au5: autopilot navigation_task					
	Au6: autopilot altitude_control_task					
	Au7: autopilot climb_control_task					
	Au8: autopilot reporting_task					
18	Benchmarks analyzed	15	15	4	4	4
19	Total	17	17	17	17	17

*= The estimated WCETs here are not comparable since they assume different compilers and processors.
N/A = Memory exhausted in the test laptop.
V8[1] = Single path basic mode.
V8[2] = Single path advanced mode.

Table 4 Servability of WCET Tools with Annotations

Nr.	Benchmarks	aiT V1	aiT V2	aiT V3	Bound-T V4	Bound-T V5	Bound-T V6
1	adpcm	1	1	1	1	1	1
2	cnt	1	1	1	1	1	1
3	compress	1	1	1	1	1	1
4	cover	1	1	1	1	1	1
5	crc	1	1	1	1	1	1
6	duff	1	1	1	N/A1	N/A1	N/A1
7	edn	1	1	1	1	N/A2	N/A2
8	insertsorts	1	1	1	1	1	1
9	janne_complex	1	1	1	N/A1	N/A1	N/A1
10	matmult	1	N/A3	1	1	1	
11	ndes	1	1	1	1	1	1
12	ns	1	1	1	1	1	1
13	nsichneu	1	1	1	1	1	N/A2
14	recursion	1	1	1	N/A1	N/A1	N/A1
15	statemate	1	1	1	N/A1	N/A1	N/A1
16	Fbw	1	1	1	1	1	1
17	Au	1	1	1	1	1	1

Nr.	Benchmarks	SWEET V8[3]	SWEET V8[4]	Chronos V9	Chronos V10	Chronos V11
1	adpcm	1	1	1	1	1
2	cnt	1	1	1	1	1
3	compress	1	1	1	1	1
4	cover	1	1	N/A1	N/A1	N/A1
5	crc	1	1	1	1	1
6	duff	1	1	N/A1	N/A1	N/A1
7	edn	1	1	1	1	1
8	insertsorts	1	1	1	1	1
9	janne_complex	1	1	1	1	1
10	matmult	1	1	1	1	1
11	ndes	1	1	1	1	1
12	ns	1	1	1	1	1
13	nsichneu	1	N/A3	1	1	1
14	recursion	1	1	N/A1	N/A1	N/A1
15	statemate	1	1	1	1	1
16	Fbw	N/A1	N/A1	1	1	1
17	Au	N/A1	N/A1	N/A1	N/A1	N/A1

1 = Program analyzable.
Fbw = fly-by-wire.
Au = autopilot.
N/A1 = The tool does not handle this problems with the release at that time.
N/A2 = Microsoft Windows informed that the Omega Calculator had conflict with Microsoft Windows System.
N/A3 = Memory exhausted in the test computer.
V8[3] = Multi-path basic mode.
V8[4] = Multi-path advanced mode.

Table 5 Estimated WCET Values in Cycles with Annotations

Nr.	Benchmarks	aiT V1	aiT V2	aiT V3	Chronos V9	Chronos V10	Chronos V11
1	adpcm	558342	1375886	430274	265588	347742	317354
2	cnt	20250	17053	7376	4896	6438	5401
3	compress	37570	20280	9461	5873	29215	28487
4	cover	10452	6780	5006	N/A1	N/A1	N/A1
5	crc	275910	213337	98830	47786	61849	53275
6	duff	7196	4612	1355	N/A1	N/A1	N/A1
7	edn	927068	307889	88381	89401	113612	89030
8	insertsorts	4870	3992	1838	901	1549	1245
9	janne_complex	1330	829	383	189	800	789
10	matmult	956710	N/A3	237736	186903	191615	119526
11	ndes	453348	194448	130025	66655	107589	85918
12	ns	75712	38043	18215	8199	9991	8676
13	nsichneu	29840	18827	8327	13609	97908	97525
14	recursion	10076	7451	5527	N/A1	N/A1	N/A1
15	statemate	2620	3812	1294	2007	16185	16103

N/A1 = The tool does not handle this problems with the release at that time.
N/A3 = Memory exhausted in the test computer.

Table 6 Measured and Simulated Execution Times

Nr.	Benchmarks	aiT V1	aiT V2	aiT V3	Chronos V9	Chronos V10	Chronos V11
1	adpcm	buffer	buffer	buffer	160891	183526	126258
2	cnt	19622	16853	7235	4792	5586	3515
3	compress	27308	19970	6824	5859	7504	4744
4	cover	10080	6778	4299	N/A	N/A	N/A
5	crc	buffer	buffer	buffer	22688	26861	18098
6	duff	6919	4610	1028	N/A	N/A	N/A
7	edn	838686	299734	buffer	87444	108973	62995
8	insertsorts	4720	3990	1770	897	1364	949
9	janne_complex	1294	827	359	185	454	356
10	matmult	936602	438435	buffer	186899	185937	90834
11	ndes	401294	190530	buffer	65600	86639	53625
12	ns	73738	36097	buffer	6577	7568	4784
13	nsichneu	28328	18825	8052	6305	42966	40931
14	recursion	8318	7143	5096	N/A	N/A	N/A
15	statemate	2486	3810	1260	1120	6207	5898

N/A = not applicable.
buffer = Because of the buffer limitation, it is not possible to measure the WCETs.

Table 7 WCET Tightness

Nr.	Benchmarks	aiT V1	aiT V2	aiT V3	Chronos V9	Chronos V10	Chronos V11
1	adpcm	N/A	N/A	N/A	65.07%	89.48%	151.35%
2	cnt	3.2 %	1.19%	1.95%	2.17%	15.25%	53.66%
3	compress	37.58%	1.55%	38.64%	0.24%	289.33%	500.48%
4	cover	3.69%	0.03%	16.45%	N/A	N/A	N/A
5	crc	N/A	N/A	N/A	110.62%	130.26%	194.37%
6	duff	4.05%	0.04%	31.81%	N/A	N/A	N/A
7	edn	10.54%	2.72%	N/A	2.24%	4.26%	41.33%
8	insertsorts	3.18%	0.05%	3.84%	0.45%	13.56%	31.19%
9	janne_complex	2.7%	0.24%	6.69%	2.16%	76.21%	121.63%
10	matmult	2.15%	N/A	N/A	0.0%	3.05%	31.59%
11	ndes	12.97%	2.06%	N/A	1.61%	24.18%	60.22%
12	ns	2.68%	5.39%	N/A	24.66%	32.02%	81.35%
13	nsichneu	5.34%	0.01%	3.42%	115.84%	127.87%	138.27%
14	recursion	21.13%	4.31%	8.46%	N/A	N/A	N/A
15	statemate	5.39%	0.05%	2.7%	79.2%	160.75%	132.02%

N/A = not applicable.

Table 8 Usability Assessment: Taking into account both the Mälardalen and the PapaBench Benchmark Programs

Tool	Average Tightness for Annotation	Warning of Annotation Errors	Accomplishment of Intended Tasks in Time	Acceptability in Analysis Time
aiT	aiT GUI, aiSee GDL, Hints, Links, User manual.	Cases were found.	Yes	Acceptable
Bound-T	Run messages, Graph, User manual.	No case was founded.	Generally yes	Generally acceptable
SWEET	Run messages, Graphs.	No case was founded.	Yes	Mostly acceptable
Chronos	GUI^1, User manual.	No case was founded.	Generally yes	Good^2

GUI^1 = Chronos GUI could not be tested in the challenge.
Good^2 = Test withinteger linear programming solver CPLEX.
Acceptable = Average analysis time and their variations are within a few seconds or minutes.

Table 9 Overview of Functional and Service Quality of WCET Tools

Tool	Average Tightness	Benchmarks not Handled by the Tool	Benchmarks Analyzed	Benchmarks under Test	Average Service Rate
aiT	7-8%	0	17	17	100%
Bound-T	N/A	4	13	17	76.5%
SWEET	N/A	2	15	17	88.2%
Chronos	81-89%	4	13	17	76.5%

Tool	Programs Analyzed Without Annotation	Programs Tasks under Test	Average Automation Rate	Complexity of Processor Supported*
aiT	45	84	54%	Simple, Medium, Very Complex
Bound-T	13	51	26%	Simple, Medium
SWEET	15	17	88%	Medium
Chronos	12	51	24%	Configurable Simulated Processor

N/A = No measured WCET was available and no WCET tightness was available at this time.
* = The classification of the processors type is based on the challenge statement.