The Worst Case Execution Time Tool Challenge 2006: The External Test

Lili Tan, *University of Duisburg-Essen*

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 so to encourage research and activities in the WCET community. It applied two evaluation approaches to the tools submitted, self-evaluation and external evaluation. In order to balance users' effects and to achieve a comparable evaluation, an independent test person has been assigned by the WCET Challenge Working Group to perform the external tool evaluation.**

The test person visited the tool developers and evaluated the tools entered. This paper describes the testing procedures applied, the results obtained, and the evaluation made in the Challenge.

Index Terms— **Timing Analysis, Worst Case Execution Time (WCET), Hard Real Time**

I. INTRODUCTION

THE first Worst Case Execution Time (WCET) Tool Challenge invited all tool providers to submit their tools and to evaluate the state-of-the-art in timing analysis for real-time systems to encourage WCET research and activities in the WCET community [1]. 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 [2]. 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 evaluating 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 WCET Tool Challenge is also described in another ISoLA 2006 paper [4]. Details about the features of tools and test results are addressed in a technical report [6].

This paper is intended to describe the external tests performed in the Challenge and the test results obtained. The remainder of the paper is organized as follows. At first, the test framework is introduced in Section II. The implementation of the external test work is described in Section III. The test results are presented and the usability of tools is assessed in Section V. At the end, the author draws conclusions.

II. TEST FRAMEWORK

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

A. WCET Tools under Test, Variants, and Updates

All of the WCET tools that entered in the Challenge have been registered for the external test. Following is a short description about these tools in the sequence of registration.

aiT WCET Analyzer (aiT) is a software product of the company AbsInt Angewandte Informatik GmbH. 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 in the tests: 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. Bound-T 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.

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

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

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

Lili Tan is with the ICB/Computer Science, University of Duisburg-Essen, Schuetzenbahn 70, 45127 Essen, Germany, (e-mail: lili.tan@icb.uni-due.de).

Chronos is a research prototype developed by the National University of Singapore. Chronos uses SimpleScalar, the GCC compiler and allows two types of processor configuration, i.e., Simple in-order and Complex in-order.

All of the tool developers were allowed to develop their tools and to submit their new improved software updates during the Challenge. All updates for tools and test variants are listed in [6].

B. Programs under Test

The WWG selected a set of 15 programs out of the Mälardalen WCET benchmark and two PapaBench benchmark tasks as the basis for the evaluation. The programs are available in C source code [1]. The C-programs were compiled by the participants with the respective compilers.

C. Test Scopes

The WWG decided that tools should be tested for three aspects, namely flow analysis, required user interaction, and performance [1]. The test was designed to run with a three-round procedure, i.e. zero annotation, minimal set of annotations, and finally an optimal set of annotations to improve the WCET quality to the best level [1].

D. Test Computer Specification

The computer used for running the external test is an AcerSystem laptop with AMD Mobile Sempron ™ Processor 2800+, 1.60 GHZ, and 448 MB RAM.

Three other PCs (AMD Athlon ™ 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.

III. TEST IMPLEMENTATION

Based on the test framework, a schedule was made to visit the tool developers and 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 II in [6] shows the final test schedule.

A. aiT

1) Test Initiation

We visited 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. The tips and tricks for annotations are a great help for the improvement of WCET results. 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 VII) and the estimated WCET were compared. Some of the measurements have been done several times. After the visit, several test results were further elaborated via email and express replies were received.

2) Working with aiT

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

For the improvement 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 is very helpful and 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. A figure in [6] 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. To demonstrate these features, we present an example for the benchmark test program *edn*. aiT for MPC565 detected some iterations for *edn* automatically. A screenshot in [6] displays the automatically detected number of iterations - 23 times (#23) - in the main loop. 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 following warning message in aiT GUI [6]:

"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".

These features help to avoid wrong annotations made by users and limit the chance to produce errors.

4. aiT collects all settings 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 users' manuals, and do not need to specify the

paths. The handling of an aiT project works by simple mouse-clicks.

6. The analysis time of aiT for benchmark test programs is generally a number of 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.

The typical working pattern with aiT is: invoke aiT either with the aiT icon or through the 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.

B. Bound-T

1) Test Initiation

The first installation of Bound-T and tests with example programs were finished in Essen on October 15th. After an update of the Omega Calculator, all of the 15 Mälardalen WCET benchmarks for H8-300 in *.coff format were tested on October 22nd. A short feedback was emailed to the tool developer before the trip. On October 26th, a Bound-T meeting was held in Västerås, Sweden. The Bound-T developer introduced important technical issues about Bound-T. Tips and tricks for assertions were explained in detail with an example. Further tests with assertions were tried on this same day and on October 27th. Testing of all binary files for all target processors was finished after the visit. Tests with more assertions were done.

2) Working with Bound-T

After invoking Bound-T from the command line, Bound-T delivered 13 WCET results without annotation (see Table III, 26% of 51 programs).

Bound-T displays run messages on the screen and an output graph. Both of them are helpful for users while they develop annotations. A screenshot of the Bound-T interface in [6] presents the analysis results of benchmark test program *cnt* for the target processor H8-300. The loop bounds detected by Bound-T are listed in the run messages together with the function names of the C source file, and 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 [6]. A figure in [6] 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.

Human interaction with Bound-T includes: invoking Bound-T through the command line, following the hints in the run messages, looking up reasons and actions to be taken in the users' manuals, developing annotations, executing the WCET calculation through the command line, and saving the WCET output value for later use. 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%) Reasons why the test failed with 4 programs were: Bound-T was not able to handle recursion in the *recursion* and nested loops in the *janne_complex* and *statemate* test programs. It returned an error message about "irreducible flow-graph" while analyzing the *duff* test program, which contained multiple entry points in a loop.

C. MTime

The visit to the Real-Time Systems Group at the TU-Vienna was from October 11th to 12th.

MTime developers provided us with a PC and example test programs that they develop. The usability for the examples is similar to those from the other WCET tools in the Challenge. Due to the fact that MTime does not support function calls at that time, it was not possible to produce any result for the WCET Challenge benchmark programs.

D. SWEET

1) Test Initiation

A SWEET meeting was held at Mälardalen University, Västerås, Sweden 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 were the 15 Mälardalen WCET benchmarks in intermediate NIC code format. 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.

2) Working with SWEET

Without using any annotation, SWEET worked both for basic and advanced modes. 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 III. The run messages and DOT graphs output can optionally be saved for later use. A figure in [6] for the test program *janne_complex* illustrates the hierarchical structure of loops and the nested relation between inter and outer loops as detected by SWEET. The reduction in estimated WCETs by the use of the advanced mode can be observed by comparing the test results [6]. This is the effect from the advanced loop bounds calculation and infeasible path detection.

SWEET can also be assigned with annotations files and work in multi path mode. Using annotation files for some test programs, intervals can be assigned to variables. SWEET is then expected to calculate the WCETs concerning the effects resulting from multi-path execution instead of single path. The applicable circumstances must be investigated before using this mode. To construct the constraints for annotations, the intermediate NIC file must be consulted. Future users might profit from a formal and detailed description of the syntax and semantics of annotation files, as well as automatically processing of appropriate modes where they are suitable.

The working pattern with SWEET was: start SWEET in the Cygwin environment and find the WCET result in the run message on the monitor screen.

SWEET delivered 15 WCET test results for all of the 15 Mälardalen WCET benchmarks without annotations (see Table III). 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 the memory limitation in the test laptop.

E. Chronos

1) Test Initiation

The test of Chronos was performed in the last week of October and at the beginning of November 2006.

2) Working with Chronos

Working remotely through the Internet and testing in different operating system platforms and with different runtimes, several implementation problems occurred that did not relate to WCET research, but increased the test overhead during the test. However, these problems were solved at the end. 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. The Chronos GUI could not be used because of the extremely slow response resulting from the Java runtime in remote, as the Chronos GUI is written in Java. Instead of using the Chronos GUI, we switched to use the Chronos command line interface. The benchmark programs were compiled through remote access by using GCC2.2.2.3 and analysed by the Chronos WCET analysis kernel. The estimated WCET values were calculated by using CPLEX remotely [6]. The simulated WCET values were obtained by using the command line for simulation, a utility provided by the SimpleScalar Simulator [6]. The two processor configurations for both the estimated WCETs and simulated execution times require the corresponding configuration setting files, i.e.,

1. simple in-order: in-order pipeline with perfect cache and perfect branch perdition ,
2. complex in-order: in-order pipeline with cache modelling and branch-perdition

Chronos could handle 15 out of 17 benchmark test programs (88.2%) for all of the two processor configurations. Reason why the test failed with 2 programs was: Chronos was not able to handle recursion. 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 quickly using CPLEX. In addition, Chronos is GNU open source software and the source codes of Chronos tool are accessible. The chance to do experiments with the source code and to investigate the internal WCET analysis processes is therefore given.

IV. Test Results

We present the test results and try to decipher their meanings 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
- Summary of functional and service quality

A. 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 III. 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.

B. 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; while from the aspect of quality, the precision increases and the overestimation is reduced. Detailed values are found in [6].

1) Service of WCET Tools

The number of different programs that a tool can handle indicates the ability of a tool in serving 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 IV presents the abstract service results. In addition, errors encountered - test computer related or not - are also presented in the table.

2) Estimated WCET with Annotations

The estimated WCET values in [6] are used for the calculation of the precision of tools.

3) Measured and Simulated WCET

The measured and simulated execution time is listed in [6].

4) WCET Tightness

The WCET tightness in Table V represents the ratio of measured execution time and predicted WCET (as found in [6]). No measurement value has been obtained from other tools. Therefore, the WCET values calculated and their precision could only be evaluated for aiT and Chronos.

C. 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 helpful tips and hints are given by a tool rather than just reading them in the users' manual. Tool developers should not expect users to read their document from first to last page. Users generally expect appropriate tips and hints appearing automatically at the appropriate moment when they use the tools. User manuals are expected to be complete but concise. A document is only necessary if the tool is not self-explanatory.

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 belong to algorithm and functionality design of a WCET tool rather than to usability's design themselves, do affect the assessment of usability. They detract the effectiveness, efficiency, and satisfaction from achieving the goal to successful analysis of the benchmark programs. They include the following factors:

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

The usability experienced by the author during the three-round test is summarized in Table I. From the author's experience, aiT is the tool that provided the best usability.

D. Summary of Functional and Service Quality

Table II 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 II 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.

V. Conclusion

Although the WCET Tool Challenge 2006 has been run within a very compressed time frame, we survived. All of the tools, aiT, Bound-T, MTime, SWEET, and Chronos have been tested. The entered WCET tools have demonstrated many positive test results from the aspects of functionality and service quality.

Many tools have proved their abilities in dealing with programs of complex structures, even with recursive functions and multiple loop headers, which do not often occur in real applications. Some tools have shown their strengths in various areas: aiT is able to handling 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 can also 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% of benchmark 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 to AbsInt and Saarland University in Saarbrücken, TU-Vienna in Vienna, and the University of Mälardalen in Västerås have proved to be an efficient way to give direct feedback to developers, especially those developers who had provided us with a trial version in advance. Feedback on the usability of tools and on the possible software faults has also been taken into account by 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 positive 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 showed great interest in analyzing industrial application code with their tools.

This challenge reveals the very successful "Saarland Model" demonstrated by Saarland University and AbsInt. The cooperation between scientific research and industry has an important synergistic effect on all participants in the cycle of education, research, and industry in our society [5].

In the future, more successful results in WCET scientific research, in tool development, and in industry application can be expected. We look forward to seeing the positive effects of such challenge on the WCET community.

ACKNOWLEDGMENT

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 from AbsInt Angewandte Informatik GmbH, Stefan Thesing from Saarland University, Niklas Holsti of Tidorum Ltd., Björn Lisper, Jan Gustafsson, Stefan Byqde of Mälardalen University, Raimund Kirner, Ingomar Wenzel, and Berhard Rieder of TU-Vienna, Tulika Mitra, Abhik Roychoudhury, Vivy Suhendra, and Liangyun of National University of Singapore.

REFERENCES

[1] (WCET Tool Challenge 2006) J. Gustafsson. (2006, November). Available: http://www.idt.mdh.se/personal/jgn/challenge/.

[2] R. Wilhelm et al. "The Worst Case Execution Time Problem –Overview of Methods and Survey of Tools" (TECS-2004-0004.R1, ACM Transaction on Programming Languages and Systems).

[3] ISoLA 2006, http://sttt.cs.uni-dortmund.de/isola2006/?id=program,

[4] J. Gustafsson, "WCET Tool Challenge 2006", In Proc. 2nd International Symposium on Leveraging Applications of Formal Methods (ISoLA'06), November 2006.

[5] L. Tan and L. Suhl, "Scenario and Authoring Tool for Web-based Case Study Education in Operations Research/Management Science (Proceeding of World Association for Case Method Research and Application)", 2002.

[6] L. Tan, "The Worst Case Execution Time Tool Challenge 2006: Technical Report for the External Test", ICB/Computer Science, University of Duisburg-Essen, 2007.

TABLE I

USABILITY ASSESSMENT: TAKING INTO ACCOUNT BOTH THE MÄLARDALEN AND THE PAPABENCH BENCHMARK PROGRAMS

Tool	Facility Available for Supporting Annotation	Annotation Error Warning	Accomplish the intended task within acceptable time	Analysis time Acceptability
aiT	aiT GUI, aiSee GDL, Hints, Links, User manual.	Yes, cases had been found.	Yes	Acceptable
Bound-T	Run messages, Graph, User manual.	No case founded during the challenge.	Generally yes	Generally acceptable
SWEET	Run messages, Graphs.	No case founded during the challenge.	Yes	Mostly acceptable
Chronos	GUI*, User manual.	No case founded during the challenge.	Generally yes	Good**

* = GUI could not be tested in the challenge. ** = Test with CPLEX integer linear programming solver, Acceptable = Average analysis time and their variations are within a few seconds or minutes.

TABLE II

OVERVIEW OF FUNCTIONAL AND SERVICE QUALITY OF WCET TOOLS

Tool	Average Tightness	Benchmarks failed with Fatal Errors	Service Number	Bench-marks under test	Average Service Rate	Programs Proceeded Automatically	Programs under Test	Average Auto-mation Rate	Complexity of Processors Supported*
aiT	7~8 %	0	17	17	100%	45	84	54%	Simple, Medium, Very complex
Bound-T	N/A	4	13	17	76.5%	13	51	26%	Simple, Medium
SWEET	N/A	2	15	17	88.2%	15	17	88%	Medium
Chronos	50%	2	15	17	88.2%	8	34	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.

TABLE III

ESTIMATED WCET VALUES * WITHOUT ANNOTATIONS

Nr	**Bench-mark Program**	**aiT C16X C16x (cy)**	**aiT ARM7 TMS470 (cy)**	**aiT PPC MPC 565 (cy)**	**Bound-T Renesas H8/300_ GCC (cy)**	**Bound-T Renesas H8/300_ iar (cy)**	**Bound-T SPARC V7/V8 (ERC32) (cy)**	**SWEET ARM9_C TONIC Single path basic mode (cy)**	**SWEET ARM9_C TONIC Single path advanced mode (cy)**	**Chronos Simple Scalar Simple in order (cy)**	**Chronos Simple Scalar Complex in order (cy)**
1	adpcm							2165650	2162122		
2	cnt	32812	26572	7576	45806	68982		36719	35319	4896	6438
3	compress							206480	49896		
4	cover	19459	6780	5451		10250	180	73128	63563		
5	crc			107278	164118	268657		834159	830278		
6	duff							5525	4720		
7	edn		307889	104907				1425085	1425085	89401	113612
8	insertsort							31163	18167		
9	janne_ complex							12039	2523	189	800
10	matmult	1562815	523599	237736	1506520	3282132		2532706	2532706	186903	191615
11	ndes	816337	194448	144845		712454	4214	795425	795425		
12	ns	238414	38043	34361	20976	256960	7097	130733	130671		
13	nsichneu		41687	21362				119707	N/A		
14	recursion							29079	20033		
15	statemate							15964	8451		
16	Fbw1		9875	1242							
	Fbw2		3197	331							
	Fbw3		4092	437							
	Fbw4	2390	1909	1249							
	Fbw5		4058	432							
17	Au1		15972	2247							
	Au2		4239	340							
	Au3	170	144	81							
	Au4										
	Au5										
	Au6		915	95							
	Au7		4129	247							
	Au8	8286	11172	4465							
18	Counted	8	18	19	4	6	3	15		4	4
19	Total	28	28	28	17	17	17	17		17	17

cy = cycle, * = The estimated WCETs here are not comparable since they assume different compilers and processors.

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.

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.

TABLE IV
SERVABILITY OF WCET TOOLS WITH ANNOTATIONS

Nr.	Benchmark Program	aiT C16X C16X (cy)	aiT ARM7 TMS470 (cy)	aiT PPC MPC565 (cy)	Bound-T Renesas H8/300_ GCC (cy)	Bound-T Renesas H8/300_ iar (cy)	Bound-T SPARC V7/V8 (ERC32) (cy)	SWEET ARM9_CTONIC Single path and multi-path, basic mode (cy)	SWEET ARM9_CTONIC Single path and multi-path, advanced mode (cy)	Chronos Simple Scalar Simple in-order (cy)	Chronos Simple Scalar Complex in-order (cy)
1	adpcm	1	1	1	1	1	1	1	1	1	1
2	cnt	1	1	1	1	1	1	1	1	1	1
3	compress	1	1	1	1	1	1	1	1	1	1
4	cover	1	1	1	1	1	1	1	1	1	1
5	crc	1	1	1	1	1	1	1	1	1	1
6	duff	1	1	1	N/A1	N/A1	N/A1	1	1	1	1
7	edn	1	1	1	1	N/A2	N/A2	1	1	1	1
8	insertsort	1	1	1	1	1	1	1	1	1	1
9	janne_complex	1	1	1	N/A1	N/A1	N/A1	1	1	1	1
10	matmult	1	N/A3	1	1	1	1	1	1	1	1
11	ndes	1	1	1	1	1	1	1	1	1	1
12	ns	1	1	1	1	1	1	1	1	1	1
13	nsichneu	1	1	1	1	1	N/A2	1	N/A3	1	1
14	recursion	1	1	1	N/A1	N/A1	N/A1	1	1	N/A1	N/A1
15	statemate	1	1	1	N/A1	N/A1	N/A1	1	1	1	1
16	papa_bench fly_by_wire	1	1	1	1	1	1	N/A1	N/A1	1	1
17	papa_bench autopilot	1	1	1	1	1	1	N/A1	N/A1	N/A1	N/A1

CY = cycle, 1 = No fatal errors, Input service indicator, N/A1 = the tool does not handle this program 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.

TABLE V
WCET TIGHNESS

Nr	Benchmark Program	aiT C16X C16X	aiT ARM7 TMS470	aiT PPC MPC565	Chronos SimpleScalar Simple in order	Chronos SimpleScalar Complex in order
1	adpcm	N/A	N/A	N/A	65,07%	89,48%
2	cnt	3,20%	1,19%	1,95%	2,17%	15,25%
3	compress	37,58%	1,55%	38,64%	0,24%	289,33%
4	cover	3,69%	0,03%	16,45%	0,18%	4,67%
5	crc	N/A	N/A	N/A	110,62%	130,26%
6	duff	4,05%	0,04%	31,81%	0,39%	12,37%
7	edn	10,54%	2,72%	N/A	2,24%	4,26%
8	insertsort	3,18%	0,05%	3,84%	0,45%	13,56%
9	janne_complex	2,78%	0,24%	6,69%	2,16%	76,21%
10	matmult	2,15%	N/A	N/A	0,00%	3,05%
11	ndes	12,97%	2,06%	N/A	1,61%	24,18%
12	ns	2,68%	5,39%	N/A	24,66%	32,02%
13	nsichneu	5,34%	0,01%	3,42%	115,84%	127,87%
14	recursion	21,13%	4,31%	8,46%	N/A	N/A
15	statemate	5,39%	0,05%	2,70%	79,20%	160,75%

N/A = not applicable. Because of the buffer limitation, it was not possible to measure the WCET values.