International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 4, No.1 (Jan-2015)

http://dx.doi.org/10.12785/ijcds/040105

Performance Evaluation of Real-Time Systems

Mridula Sharma[1], Haytham Elmiligi[2], Fayez Gebali[1]

[1] ECE Department, University of Victoria, Victoria, BC, Canada
[2]Computing Science Department, Thompson Rivers University, Kamloops, BC, Canada

Received 15 Apr. 2014, Revised 17 Jul. 2014, Accepted 16 Aug. 2014, Published 1 Jan. 2015

Abstract: Performance evaluation of the Computer Systems has been a challenging task. Measurement of Worst Case Execution Time (WCET) is an important metrics for evaluating the performance of the Real Time Systems (RTSs). In this paper, we explore different methods used for WCET estimation. We mainly focus on three different techniques that are used for WCET estimation i.e. Static, Measurement-based, and Hybrid. The objective here is to provide a comparative analysis of various popularly used WCET measurement tools to guide researchers and engineers to be able use the most suitable techniques for their target applications. We also explore the probabilistic approach through model checking in our work.

Keywords: WCET, Performance Evaluation, Real-Time, Static Tools, Measurement based tools, Hybrid Tools, Simulation, Probabilistic Model

1. INTRODUCTION

Real-Time System (RTS) design has developed rapidly over the last decade. With the introduction of new emerging technologies and the development of consumer electronics, RTSs have become a very popular solution for researchers and engineers [1]. The need for RTSs in different industries is not rationalized only because they guarantee satisfying timing constraints while executing complex tasks, but also because they offer a reliable solution for portable electronic devices. RTSs have been successfully implemented on mobile devices to run many complex applications [2].

Designing a RTS is always a challenging task. Although various Real-Time Operating Systems (RTOSs) are currently available for developers [1], it is very difficult to compare the performance of these different RTOSs for the target application. Some research work has been done to address this problem for small microcontrollers [3]. However, there is no one answer for the performance evaluation question that works for all platforms. The problem becomes even more complex with RTS because not all RTS use RTOS so designers now have to spend a lot of time to study different implementations before choosing the right design for the target application.

In this paper, we tackle the performance evaluation problem and explore different methods to evaluate the performance of RTSs. We are extending the work done in [4] and present a comparative analysis of three different techniques that evaluate the performance of RTSs based on the estimation of Worst Case Execution Time (WCET). WCET is the measurement of the maximum possible time required to process a task on a given processor [5]. The system is thus defined to meet the deadline of the task in order to guarantee its functionality in all scenarios [6]. In **Figure 1**, the relativity of WCET with other real-time system parameters is depicted.

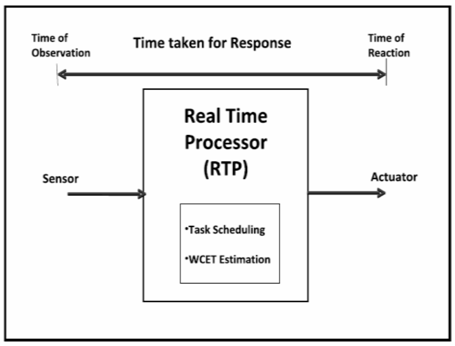

Figure 1: WCET for Real Time Systems

E-mail: naina@uvic.ca, haytham@ieee.org, fayez@uvic.ca

Although a large number of metrics are currently being used to measure the performance of embedded systems, such as response time, throughput, or utilization factor of the resources etc., we choose the WCET because it is the most significant factor for the assurance and verification of the reliability of RTSs [7].

The terms used for WCET estimates are:

- Safe bound - Estimate must be always greater than the actual execution time
- Tight bound - Estimate must be as close as possible to actual WCET
- Upper bound - The maximum possible time of the execution

The safe bound and the tight bound are useful in calculating the maximum time required for the overall process. For RTSs, the success depends on the guaranteed possibility of execution of all the tasks within their available possible time i.e. deadline. Schedulability is the process of defining the sequence of operation of different tasks in the system. Both schedulability and WCET analysis are critically important for the performance evaluation of RTSs as:

- WCET analysis defines the execution times of different tasks
- Schedulability analysis makes a decision about the task to be used for execution further [8].

In WCET estimation, measurement of the upper bound may solve the issue of failure of the real-time systems. For the overall processing, multiple tasks needs to be processed in the defined sequence in a given defined time in RTSs. In RTSs, the tasks are constrained by deadlines. WCET of each task helps in ensuring that all the tasks are executed to make the system work correctly. Overall WCET can be calculated as the sum of the WCET estimations of different tasks required to be complete through different paths for the overall completion of the process [9]. It is also possible to take decision about the schedulability of the different sub-tasks only when their upper bounds are known. The WCET over-estimation of the sub-tasks will lead to a wrong decision about declaring a task non-schedulable and hence making invalid decisions. This may even lead to invalid or inaccurate selection of the required hardware for the final implementation [10]. WCET is realistic and relevant for the RTSs and is used extensively for designing and verifying the safety critical RTSs like vehicles or power plants etc. WCET is useful for both soft as well as hard RTS as:

- For the Soft real time systems WCET is required for system understanding
- For hard real time systems, WCET are useful for their guaranteed behavior [11]

Challenges for real-time systems

Traditionally, WCET analysis presumes non-interruptible program execution, but real-time systems work on the principles of task pre-emption. Including this, there are several issues which needs to be addressed while performing timing analysis of the real-time systems [8].

The common ones are:

- Control Flow: There may be multiple loops and through go-to statements, control may flow at different places, which creates problem for the static analyzer to analyze the loop bounds.
- Dynamic function calls: The dynamic calls may only be measured at run-time, so for static analyzers, it is an issue.
- Dynamic loop bounds: Same as previous, the loop bounds are not known until the program runs and in real-time systems, bound are not known till it runs in the actual environment. The tasks scheduling in real-time systems is based in the priority of the tasks to be processed.
- Interrupts call: As compared to traditional systems, real-time systems has to manage interrupts, so WCET analysis needs to have detailed interrupt information about the system.
- Shared Cache: In real-time systems, cache state changes because of the priority and task pre-emption, which leads to an overestimation of the WCET.

In spite of all these challenges, WCET analysis has been the most common and proven choice as it gives best time estimation for different tasks in execution [12] [8]. WCET analyzers calculates un-interrupted execution time of a program by presuming that the task alone has all the resources it need. The real-time systems has many other consideration like Cache and pipelining etc, and the tasks are scheduled keeping these factors in mind. WCET estimation helps in making the decisions about the scheduling of new tasks to ensure that all the tasks are processed in the given deadline. WCET estimations are not only required to calculate the execution time, but also helps in estimating arrival time, relative deadline or absolute deadlines of different tasks. WCET measure is equally important for all categories of [13].

There are number of tools used by the researchers or used and in the industrial setting for estimating WCET of the systems. In this paper, we are listing and comparing the commonly used WCET tools i.e. Chronos, UPPAAL, OTAWA, aiT, Bound-T, SWEET, Heptane, Volta, VisualSim, TimeWeaver and rapiTime. The rest of the paper is organized as follows. Section 2 talks about WCET tools, Section 3 provides their brief descriptions, section 4 has a comparative analysis of the tools and the Section 5 contains discussions. Finally we conclude the paper in Section 6.

2. CATEGORIES OF WCET TOOLS

The WCET tools are divided into three categories i.e. Static tools, Measurement based tools and Hybrid tools [9].

A. Static Tools

Static is the term used for mathematical modelling of the system, without running on a specified processor, rather by analyzing the sets of possible control flow paths through the program. These tools are based largely on high level abstraction of the systems and make lots of assumptions and finally provide asymptotic results [14]. Since static analysis can theoretically consider all possible execution of the program, it always provide safe estimates.

The static analysis is done in three phases:

- Phase I : Flow Analysis, where information about the possible program execution paths is derived. The flow analysis works on the software and is developed by analyzing the source or object code of a program to find the constraints on the program flow. It is used to find the bounds on the execution counts of different sub-tasks. In this phase the program is sliced into basic blocks which are sequences of instructions with no jumps within the block. These are then organized into a Control Flow Graph (CFG) that maintains the control flow of the program [15].

- Phase II : Low-level Analysis, where execution time of the sub-tasks are calculated based on the effect of the target architecture on execution time. The Low-level analysis is primarily for analyzing the underlining hardware based on its performance model.

- Phase III : Calculation Phase, where derived flow and timing information are combined for the overall WCET estimate [6]. The final estimation of the WCET is done using the methods of Integer Linear Programming (ILP) or Implicit-Path Enumeration Techniques (IPET).

Traditional WCET analysis methods involved program path analysis for determining the infeasible paths in the program's control flow graph presuming all the resources are available whenever needed. For complex real-time systems, timing effects of micro-architectural features of the system, such as pipelines, caches and branch prediction are also needed to be considered. The final WCET calculation is done via Integer Linear Programming, where feasible path is defined using the subtasks [16].

The relationship of these phases is shown in the **Figure 2**

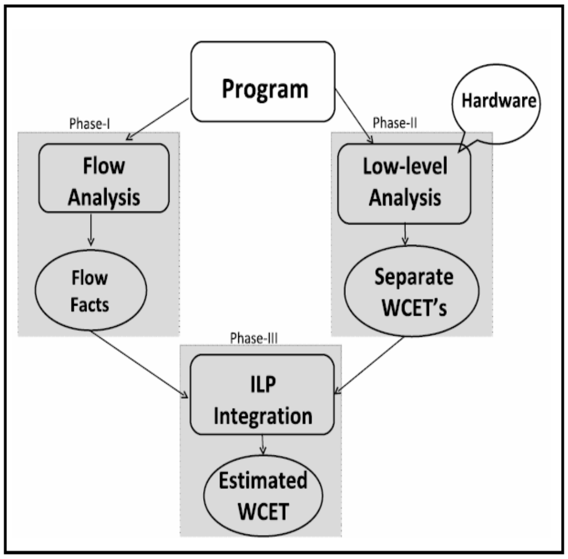

Figure 2 : Steps of Static WCET Analysis

Since static tools are able to provide safe upper bounds [8], and for the hard-real time systems, safety is critical, so, static methods is the better choice for analyzing real-time systems by the research groups [8].

B. Measurement based Tools

In the measurement based tools to calculate WCET, the calculations are done by actually running the code. Through these tools, measurements are done by running the program with proper input data on the host machine and measure the actual time taken to run the program using the measuring tools like oscilloscope, logic analyzers or in-circuit emulators. The host machine may be the real target system or some other simulated environment [17]. Once executed, the maximal and minimal values are calculated. These tools are primarily used for the industrial practices. Experimental results are found through the testbeds i.e. through the new build systems and then a safety margin is added to get the WCET. Each measurement runs through only through one path of the program, whereas there may be several possible execution path of a program. Even if exhaustic checking is done, it is not ensured that it may have run through the worst-case input, hence measured times are often the underestimation of WCET [14]. In the actual implementation, the WCET would never be reached i.e. the system will always be able to measure the time taken to process the given task. To get WCET, some safe margin is added to the calculated WCET, but it is not appropriately known about how much is the safe margin for the accurate WCET calculations. Because of this reason, the results are not very safe and tight [9] and hence these tools are only suitable for less time critical systems. For the more accurate results for RTSs, static methods are more used.

C. Hybrid Tools

Some specialized hybrid tools are also available, which combines both static and measurement based tools for the overall estimation. In most of the cases, measurements of sub-tasks are done using the measurement tools and then the static tool is used for calculating the estimated WCET. These tools identifies the single feasible path which is followed irrespective of the variation of the input and data to be processed. In hybrid analysis, measurements may replace or additionally used with the detailed low-level static WCET analysis. However, hybrid methods also, do not result in a safe WCET estimate, and hence be useful and appropriate for the soft real-time systems where such WCET estimates are not crucial [8].

D. Probabilistic Model

Although this is a comparatively new area or working, yet this is an important method as it may take into the consideration the features of the platform like cache, branch predictors, and dynamic out-of-order instruction scheduling. Since, The program is run on the actual machine in this model, it may be called as hybrid method. For the purpose of calculations, a program is run several times on the system, with numerous sets of random input data and the end-to-end execution time of the program runs are measured. Once the timing information is gathered with the input sets, extreme value statistics are deployed on the data. The main concern of this modelling is the right and/or left hand tail of a normal probability distribution, as opposed to the modelling of the average case with conventional statistics [18]. In this approach, the execution times of atomic units of execution, so-called basic blocks is measured, to obtain a probabilistic distribution. These basic blocks distributions are then combined by applying different rules for each control structure in the syntax tree in a bottom-up process. This results in a distribution for different execution times of an entire program [19]. Lv et al. [15] compared the performance of Model analysis and Static methods and concluded:

- Model checking only works well for simple programs,
- It is inclined to scalability problems when dealing with programs that have complex structures and large loop counts.

SPIN was used as the model checker in their work.

Another study by Huber and Schoeberl [20] is done on Java uni-processors and they concluded:

- Model checking is fast enough for local analysis and small applications.

They used UPPAAL as model checker and recommended that model checking is more important for code fragments and it may be combined with the IPET approach for attaining tight WCET bounds.

3. Description of the Tools

Numerous WCET tools are available free or commercially, belonging to the different categories. As listed in **Figure 3**, we discuss these tools in the subsequent paragraphs. The basic features of these tools are studied for the purpose of developing a comparative analysis in the next section.

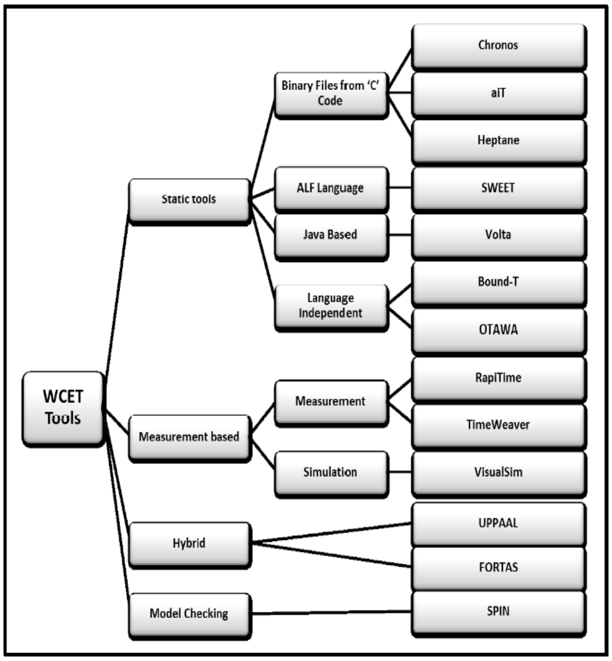

Figure 3: Categories of WCET Tools

A. Static Tools

1) Chronos :

It is an open source software developed at National University of Singapore (NUS) specifically for the academic research community. It generates WCET estimates by taking input in the form of C programs \cite{myref_4}. The binary version of the 'C' code is used to explore the internal features of the underlying processor. The different blocks of the code are analyzed by the Chronos Analyzer and then by using Integer Linear Programming (ILP) formulations, the WCET estimation is calculated. One of the characteristic feature of this tool is to be able to pass the parameters for defining different hardware features to make estimation simpler and more accurate. Chronos also provides support for some simulators to calculate WCET by running the code on the same processor configuration to compare the values of the estimated WCET and observed WCET. The results obtained through simulation are Observed WCET. These are less than actual WCET. The estimated WCET is always more than actual WCET [21].

Estimated WCET >=Actual WCET >= Observed WCET

Chronos is a popular tool as :

- It allows the definition of different hardware features for accurate estimations
- Provides support to some simulators to run the code and get actual WCET

2) aiT

It is from AbsInt Angewandte Infomatik, is another static tool, which works on abstract interpretations (so called aiT) [22]. It also works on the binary conversion of the code and reconstructs the CFG. It works on the optimized code but does not need any changes on the code for the implementation. The interpretations are followed by Value Analysis, Cache Analysis and Loop Bound Analysis for each sub-task. Using ILP again, the final WCET is estimated. This is not a open source tool, and supports many powerful simulator to provide a facility to calculate observed WCET for the purpose of final calculations. This tool is mainly used for the timing verification in the avionics, aeronautics and automotive industries. The results of the analysis can be visualized for the user through their own tool named aiSee. aiT tool can also take the hardware configuration as annotations e.g. memories, buses or register values and has support for large number of target hardwares [23].

Some features of aiT are:

- It is used for the timing verification in the avionics, aeronautics and automotive industries.
- Facility to pass the values for the hardware configuration as annotations.

3) Bound-T

It is developed by Tidorum Ltd. This is used to compute the upper bounds of WCET for real-time embedded software [24]. It is independent of the programming language, but takes input as binary instructions from the executable files. It can accept user's assertions on the program behavior through the interface and generates the output that may be used by many other tools as well. Bound-T can

- Run through many standard programming languages.
- be used for the small to medium sized embedded processors and supports pipelining feature
- Does not support cache or other micro-architectural features.

4) Heptane

Hades Embedded Processor Timing Analyzer is provided by Hades environment and also works on C code. Operating on two program representatives, namely program syntax tree and targeteable assembly code manipulation tool named Salto, this is a another processor independent tool and can be used over any underlined architecture and therefore used by researchers often. It is also able to consider cache, branching and pipe-lining concepts and hence is also suitable for the performance analysis of the multi-core systems [25].

- It is a promising tool for the performance evaluation of the multi-core systems.

5) OTAWA

A popular toolbox i.e. a combination of various tools, is developed by TRACES team at IRIT labs, University of Toulouse, France, and is used for Adaptive WCET Analysis. It works by creating an abstraction layer on the underlined hardware. It is an open source and because of the abstraction layer, it can run on any platform [6]. Toolbox contains various tools to facilitate the implementation of new analysis with limited effort by the programmers. This toolbox also provides the possibility of defining properties to annotate any kind of object defined in the library (e.g. an instruction) in a very convenient way. The toolbox has a many components like a language and a simulator to decode, disassemble and emulate the instructions. OTAWA successfully supports several architectures like PowerPC, ARM, Sparc or M68HCS. the most well known implementation project of OTAWA is in MERASA (Multi-core Execution of Hard Real-time Application Supporting Analysability) [26] architecture where it is the main static tool for their WCET analysis. For measurements, RapiTime is used in this project.

The exciting features of OTAWA are:

- Availability of the components like language and simulator to decode, disassemble and emulate the instructions.
- Possibility to annotate the objects stored in its library.

6) SWEET

SWEedish Execution Time Analysis Tool is a research prototype developed by a research team in Västerås, Sweden since 2001, and analysis the programs in Artist Flow Analysis Language(ALF), which has been developed for the flow analysis. ALF code can be generated from different sources, like C code and assembler code, and a number of translators are available for this conversion. The ALF tools like AlfBackend or Melmac are the examples. SWEET can handle complex 'C' features like pointers, unstructured code as well as recursion [27]. It works by collecting the information about loop bounds and infeasible paths to find the safe and tight WCET for the program in consideration. The main goal of SWEET's flow analysis is to automatically calculate flow information. It has flow analyzer, which takes input in ALF format and produce flow fact files which may be interpreted by aiT or Rapita or through the low sweet tool (developed by Uppsala University and Malardalen University), it may generate WCET/BCET analysis [22]. SWEET is well known for :

- It's ability to automatically calculate flow information.
- Ability to handle complex features like pointers, loops and recursion in a 'C' code

7) Volta

A special toolkit developed for the java language implementations, is developed at Vienna University of Technology, Austria. It is useful as java is an excellent choice for the real-time systems and in the past there are not tools for java based architectures \cite{ [7]}. To focus on Real-time Specification for Java (RTSJ), a WCET tool is required that could evaluate the performance of the Java-specific processors. Two implementations namely Javelin and WCA are out and unmaintained. Volta is a suite designed for analyzing real-time software on java processors. Having two tools namely Cascade – the control flow analyzer and Clepsydra – the static WCET analyzer, this suite is open source and can be used by the researchers of the field to build their own functionality on top. Cascade provides visualizations of control-flow graph that may even be used by other tools for the further processing and visualizations, whereas Clepsydra has a unique support for back annotation interactively through which this can be integrated into the real-time programmer's development environment. Volta has great usability ion future as:

- It works on RTSJ, which is promising technology, and
- Its two analyzers make it more adaptable and easy to use.

B. Measurement-based Tools(MBT)

1) RapiTime

An automated performance measurement timing analysis tool, is targeted at real-time, embedded applications. RapiTime collects execution traces to provide the execution time measurement statistics to the use to aid determination of worst-case execution time to guides for the optimization efforts [28]. This works on the source or the binary conversion of the source code. This tool can analyze complex embedded software comprising of more than 50,000 lines of 'C' or Ada code [29]. This tool produces easy to navigate WCET reports and graphs of execution of the worst case path and also any gaps in testing. Some of the reasons for the popularity of this tool are:

- It's ability to analyze complex embedded software
- It's ability to identify the worst case hotspots (which contribute heavily to WCET).
- Ability to produce easy to navigate WCET reports and graphs of execution

2) VisualSim

The modeling and simulation software, tha is used for system's engineering. This software has a graphical model editor, through which a real-time analysis modeling is generated. Based on the build-in libraries, which has pre-built parameterized building blocks, the three simulators can perform timed computations for measuring the timing, power and arbitration of the system. One of the simulators is used for the WCET measurements through the web browser interface [30]. It provides a graphical interface through which it is possible for the user to construct debug, simulate, analyze and share their specifications. It allows user to test the logic flow, operation correctness, debugging and system optimization to meet the requirements.

3) TimeWeaver

Another product of AbsInt's, and is usable for any processor. Designed with the main focus on simplicity for the measurements of the WCET in a fully automatic way, this time estimation tool takes input in the form of set of input traces and the analysis starting point. Using these traces, the ILP is constructed to represent the dynamic control-flow graphs based on the measurements. This is not a popular tool as the traces measurement is not fully achieved because of various limitations like smaller trace buffer etc. [31].

C. Hybrid Tools/Toolboxes

1) UPPAAL

It is a toolbox used for verification of real-time systems. This tool is jointly developed by Uppsala University and Aalborg University. It is targeted for the model-checking of the real-time systems, through the modeling language based on the timed-automata formalism [32]. The tool is a hybrid of an extended subset of C and TA. The most appropriate application of this tool is for the systems having a collection of non-deterministic processes with finite control structure and real-valued clocks and are communicating through channels and shared variables [10]. The main features of UPPAAL are:

- It is Well known for its efficiency and easy usability

- It can automatically generates the diagnostic trace which is used by many other simulators to generate graphical outcomes [33].

2) FORTAS

A FORmal timing Analysis Tool. Developed at Vienna University of Technology, this works specially for the estimation of the execution time for embedded real-time software [34]. The main focus here is also on the control software's written in C language. It uses abstract models of the software to derive test data automatically independently of the target hardware. This tool is based on the hybrid approach where analysis and decomposition of the code is done statically and execution time measurement is done on the target system to have the final WCET estimate. The main features of FORTAS are:

- It can provide refinement control to answer the specific need of the user.
- It is mainly used for the WCET estimation of embedded real-time systems.

D. Model Checking Tool

1) SPIN

A generic verification system used for the design and verification of asynchronous process systems [35]. It has program like notations and is a widely used professional software tool. The Models are written in a simple language called Promela, and these models can be simulated randomly or interactively using SPIN. Spin can generate efficient verifiers that search for a counterexample to correctness specifications applied to a model [36].

4. COMPARATIVE ANALYSIS OF THE TOOLS

Error! Reference source not found. is summarizing our research:

TABLE 1:COMPARATIVE ANALYSIS OF THE TOOLS

Category	Tool Name	Supporting Language	Usability	Main Features	Hardware Support	References
Static Tools	Chronos	‘C’ Program	Open source, mainly used for research on single core as well as multi-core	Annotations for hardware features	Support large number of target hardware	[21]
	aiT	Binary conversion of 'C' code	Industrial tool used in Avionics, Aeronautics and Automotives	Annotations for hardware features	Support large number of target hardware	[22]
	Heptane	'C' code	Open Source	Supports cache architectures through several replacement policies	Processor independent	[25]
	SWEET	ALF Language	Research prototype used by Volvo CE	Supports pointers, recursion and unstructured code	ARM9 and NECV850E	[27]
	Volta	Java Language	Research project used for real time development environment	Provide Visual control Flow Graph	Java based Architecture	[37]
	Bound-T	Language independent	Used for real-time embedded systems	Accept user assertion through the interface	Supports large number of target hardware	[24]
	OTAWA	Language independent	Industrial tool	Few annotations for hardware featured through libraries	Platform independent	[6]
Measurement Based Tools	RapiTime	Binary conversion of 'C' and Ada	Commercial tool for real-time embedded systems	Identify WCET hotspots	Supports large number of target hardware	[28]
	TimeWeaver	Input in the form of input traces	Research tool for small embedded systems	Calculates Worst Case timing behavior on actual hardware	Processor Independent	[31]
	VisualSim	Not known	Industrial	Supports Web Browser interface	Supports large number of target hardware	[35]
Hybrid Tools	UPAAL	'C' and timed Automata (TA)	Model checking of the real-time systems	Efficiency and ease of use	Supports large number of target hardware	[10]

	FORTAS	'C'	Embedded real-time	May be refined for the specific need of the user	Processor independent	[34]
Model Checker	SPIN	Promela	specifying and verifying concurrent and distributed systems	Professional Tool	Supports large number of target hardware	[35]

5. Discussions

WCET estimation is an important metric for real-time systems as it provides a basis for timing analysis which is used with scheduling to get safe time estimations. The Static WCET estimation tools are well known for their ability to provide tight and safe upper bounds. Chronos WCET tool is used for the timing analysis of RTSs as it has the ability to get extended and modified to match the need and requirements of the user. This feature makes Chronos popular and relevant for the researchers in their further studies.

aiT is used in the industrial settings and provides the visual analysis of the produced results through the related component names aiSee. This is used commercially as this also provide support with many powerful simulators to calculate observed WCET for the purpose of final calculations and comparisons. Bound-T and Heptane are used for the performance evaluation of the embedded processors. These tools can also run on various platforms through many standard programming languages, and have support for cache, branching and pipe-lining etc., and hence is suitable for RTSs timing analysis. OTAWA, the toolbox, claims to provide the most accurate results within the competitive time frame [6]. OTAWA is already used in MERASA project (Multi-core Execution of Hard Real-time Application Supporting Analysability), where it is used for the static analysis for predicting timings for hard real-time tasks with RapiTime, which is its measurement based counterpart. SWEET analyzes program in ALF format and is quite automatic. Current real time systems are Java based, where Volta is used for WCET estimations. This tool has a definite future and usability for the real-time systems as more and more newer real-time systems are Java based.

Measurement based WCET tool, VisualSim is the set of three simulators used for measuring timing, power and

arbitration of the system and one of its tool works for WCET estimations. TimeWeaver is fully automatic, easy to use WCET tool that may run on any processor. The hybrid tool, UPAAL is the toolbox, having various tools and measures WCET using both the features of static and measurement based. it is known for its efficiency and easy usability and has the capability of automatically generating the diagnostic trace for many other simulators to be used.

Many groups use static WCET analysis tools for their research. Colin and Puaut used Heptane for their research on real-time systems [25]. SWEET and Bound-T are used by the researchers at Malardalen University for timing analysis of Real-time systems [11]. aiT is used as an integral part of the SCADE development environment (a DO-178B qualified environment) for Military and Aerospace Industries up to Level A [38].

6. Conclusions

WCET estimation is an important metrics for the performance evaluation of the RTSs. Numerous WCET estimation tools are available as research projects or commercial products. Some of these tools are used purely for research purposes, whereas few are used in the industry. The commonly used category is static analysis, where WCET estimation is done without running the code on the actual hardware. The commonly used static WCET measurement tool is Chronos, which is widely applied in many research and industrial settings. Other well adapted static tools are aiT, bound-T, OTAWA, FORTAS, Heptane and SWEET. These all are used for performing timing analysis in common language 'C'. Volta is a special WCET tool, specifically designed for Java based Real-time systems. Measurement based tool, RapiTime is used for performance evaluation of the MERASA project with OTAWA. UPAAL is the hybrid toolbox which can also be used as the Model checker. WCET tools are used for measuring the performance of the real-time embedded systems, but are only used for single-core machines. In this paper, we presented an overview of the tools for the real-time systems to helps researchers to understand the functions of currently available WCET tools. We would like to continue working in this field further to be able to get a tool specifically useful for the multi-core RTSs.

7. References

[1] P. Hambarde, R. Varma and S. Jha, "The Survey of Real Time Operating System: RTOS," in *2014 International Conference on Electronic Systems, Signal Processing and Computing Technologies (ICESC)*, 2014.

[2] Z. Liu and S. Song, "An embedded real-time finger-vein recognition system for mobile devices," *IEEE Transactions on Consumer Electronics*, pp. 522-527, 2012.

[3] S.-L. Tan and B. Tran Nguyen, "Survey and performance evaluation of real-time operating systems (RTOS) for small microcontrollers," *IEEE Micro*, 2009.

[4] M. Lv, N. Guan, Y. Zhang, Q. Deng, G. Yu and J. Zhang, "A Survey of WCET Analysis of Real-Time Operating Systems," in *International Conference on Embedded Software and Systems, 2009*, 2009.

[5] J. Gustafsson, "Usability Aspects of WCET Analysis," in *11th IEEE international symposium on Object Oriented Real-Time Distributed Computing (ISORC), 2008* , 2008.

[6] C. Ballabriga, H. Cassé, C. Rochange and P. Sainrt, "OTAWA: An Open Toolbox for Adaptive WCET Analysis.," *Springer*, vol. 6399, pp. 35-46, 2010.

[7] T. Harmon and R. Klefstad, "A Survey of Worst-Case Execution Time Analysis for Real-Time Java," in *IEEE international conference on parallel and distributed processing*, 2007.

[8] B. Lisper and M. Santos, "Model Identification for WCET Analysis," in *15th IEEE Real-Time and Embedded Technology and Applications symposium*, 2009.

[9] R. Wilhelm, J. Engblom, A. Ermedahl and N. Holsti, "The Worst-case Execution-time Problem:Overview of Methods," *ACM trans. Embed, Computer Systems*, vol. 7, no. 3, pp. 1-53, 2008.

[10] F. Cassez and J. L. Bechennec, "Timing Analysis of Binary Programs with UPPAAL," in *13th International Conference on Application of Concurrency to System Design (ACSD)*, 2013.

[11] *Worst Case Execution Time Analysis*, Malardalen Real-Time Research Center, 2013.

[12] S. M. Petters, "How much worst case is needed in WCET estimation," in *2nd International Workshop on Worst Case Execution Time Analysis*, Vienna, 2002.

[13] L. K. Chong, C. Ballabriga, V.-T. Pham, S. Chattopadhyay and A. Roychoudhury, "Integrated Timing Analysis of Application and Operating Systems code," in *34th IEEE Real-Time Systems Symposium (RTSS)*, 2013.

[14] L. Tan, "The worst-case execution time tool challenge 2006," *International Journal on Software Tools for Technology Transfer*, vol. 11, no. 2, pp. 133-152, 2009.

[15] M. Lv, Z. Gu, N. Guan, Q. Deng and G. Yu, "Performance Comparison of Techniques on Static Path Analysis of WCET," in *EEE/IFIP International Conference on Embedded and Ubiquitous Computing*, 2008.

[16] S. Chattopadhyay and A. Roychoudhury, "Unified Cache Modeling for WCET Analysis and Layout Optimizations," in *30th IEEE Real-Time Systems Symposium,*, 2009.

[17] Jong-In Lee et al., "A Hybrid Framework of Worst-Case Execution Time Analysis for Real-time Embedded Systems," in *Aerospace Conference*, 2005.

[18] J. Engblom, A. Ermedahl, M. Sjödin, J. Gustafsson and H. Hansson, "Worst-case execution-time analysis for embedded real-time systems," *International Journal on Software Tools for Technology Transfer*, vol. 4, no. 4, pp. 437-455, 2003.

[19] K. Höfig, "Failure-Dependent Timing Analysis - A New Methodology for Probabilistic Worst-Case Execution Time Analysis," *Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance*, pp. 61-75, 2012.

[20] B. H. Schoeberl and M. Schoeberl, "Comparison of Implicit Path Enumeration and Model checking based WCET Analysis," in *9th International Workshop on Worst-Case Execution Time Analysis*, 2009.

[21] Xianfeng Li et al., "Chronos: A timing analyzer for embedded software," *Science of Computer Programming*, vol. 69, no. 1-3, pp. 56-67, 2007.

[22] C. Ferdinand, "Worst case execution time prediction by static program analysis," in *18th International Parallel and Distributed Processing symposium,*, 2004.

[23] J. Gustafsson and A. Ermedahl, "Experiences from Applying WCET Analysis in Industrial Settings," in *10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing*, 2007.

[24] Bound-T, "Bound-T Manual," 2013. [Online]. Available: http://www.bound-T.com/user-guide.pdf.

[25] A. Colin and I. Puaut, "Worst-case execution time analysis of the RTEMS real-time operating Systems," in *13th Euromicro Conference on Real-Time Systems*, 2001.

[26] T. Ungerer, F. Cazorla, P. Sainrat, G. Bernat, Z. Petrov, C. Rochange, E. Quiñones, M. Gerdes, M. Paolieri, J. Wolf, H. Cassé, S. Uhrig, I. Guliashvili, M. Houston, F. Kluge, S. Metzlaff and J. Mische, "Merasa: Multicore Execution of Hard Real-Time Applications supporting Analyzability," in *IEEE conference on micro*, 2010.

[27] D. Barkah, A. Ermedahl, J. Gustafsson, B. Lisper and C. Sandberg, "Evaluation of Automatic Flow Analysis for WCET Calculation on Industrial Real-Time System Code," in *20th Euromicro Conference of Real-Time Systems, (ECRTS'08)*, 2008.

[28] "RapiTimeExplained.pdf," Rpita Systems, 2008. [Online]. Available: http://www.rapitasystems.com/system/files/RapiTimeExplained.pdf.

[29] "RapiTime Worst-Case execution time analysis," Rapita systems, 2011. [Online]. Available: http://www.artist-embedded.org/docs/ToolsPlatforms/CompilersTA/RapiTime/RptmeWrstCsetoolkit.p.

[30] "Visual Sim Introduction," Mirabilis Design, 2013.

[31] Reinhard von Hanxleden et al., "WCET Tool Challenge 2011: Report," in *Workshop on Worst-Case Execution Time Analysis*, 2011.

[32] C. Thrane and U. Sorensen, "Slicing for uppaal," in *Annual IEEE Conference Student Paper,*, 2008.

[33] K. G. Larsen, P. Pettersson and W. Yi, "Upaal in a nutshell," *International Journal on Software Tools for Technology Transfer*, vol. 1, pp. 134-152, 1997.

[34] S. Bunte and M. Tautschnig, "A Benchmarking Suite for Measurement-Based WCET Analysis Tools," *IEEE International Conference on Software Testing Verification and Validation Workshop*, no. April, pp. 353-356, 2008.

[35] G. Holzmann, "The model checker SPIN," *IEEE Transactions on Software Engineering*, pp. 279-295, May 1997.

[36] A. Metzner, "Why Model Checking Can Improve WCET Analysis," *Lecture Notes in Computer Science*, vol. 3114, pp. 334-347, 2004.

[37] T. Harmon, M. Schoeberl, R. Kirner and R. Klefstad, "A Modular Worst-case Execution Time Analysis Tool for Java Processors," in *Real-Time and Embedded Technology and Applications Symposium,*, 2008.

[38] *aiT reference Manual,* The AbsInt Group, 2013. [Online]. Available: www.absint.com/ait/

Mridula Sharma received her Master's degree in Computers Applications from India in 2006. She also did Masters of Science (Information Technology) in 2003. She is currently a Ph. D. candidate at the University of Victoria and she is also teaching in the Computing Science department at Thompson Rivers University, Kamloops, BC, Canada. Mridula's research work focuses on the design automation of real-time multi-core systems.

Haytham Elmiligi received his Ph.D. degree in Electrical Engineering from the University of Victoria, Victoria, BC, Canada, in 2011. He is currently affiliated with the IMS research group at the University of Victoria and the Computing Science department at TRU. Haytham is a senior member IEEE and a book series Co-editor with CRC press. Haytham's research work strongly relies on combining advanced analytical models and optimization techniques to improve the overall performance of multi-core systems.

Fayez Gebali received the BSc degree in electrical engineering (first class honors) from Cairo University, the BSc degree in mathematics (first class honors) from Ain Shams University, and the PhD degree in electrical engineering from the University of British Columbia where he was a holder of the NSERC postgraduate scholar- ship. Dr. Gebali is a professor of computer engineering and Chair of Electrical & Computer Engineering Department at University of Victoria. His research interests include parallel algorithms, 3D IC design, hardware verification and security, and wireless communications.