New Trends and Issues Proceedings on Advances in Pure and Applied Sciences

No 12 (2020) 36-43

www.propaas.eu

Optimisation of scheduled tasks by real-time measurement and correlation

Berkay Saydam*, Red Pine Software, Akdeniz, 1353. Sk. No:2, 35210 Konak/İzmir, Turkey https://orcid.org/0000-0002-1262-9207
Cem Orhan, Red Pine Software, Akdeniz, 1353. Sk. No:2, 35210 Konak/İzmir, Turkey https://orcid.org/0000-0001-7413-451X
Niyazi Toker, Red Pine Software, Akdeniz, 1353. Sk. No:2, 35210 Konak/İzmir, Turkey
Mansur Turasan, Red Pine Software, Akdeniz, 1353. Sk. No:2, 35210 Konak/İzmir, Turkey

Suggested Citation:
Saydam, B., Orhan, C., Toker, N. & Turasan, M. (2020). Optimisation of scheduled tasks by real time measurement and correlation. *New Trends and Issues Proceedings on Advances in Pure and Applied Sciences.* (12), 36–43.

Received date January 5, 2020; revised date March 20, 2020; accepted date April 12, 2020.
Selection and peer review under responsibility of Prof. Dr. Dogan Ibrahim, Near East University, Cyprus.

Abstract

For functional safety, the scheduler should perform all time critical tasks in an order and within predefined deadlines in embedded systems. Scheduling of time critical tasks is determined by estimating their worst-case execution times. To justify the model design of task scheduling, it is required to simulate and visualise the task execution and scheduling maps. This helps to figure out possible problems before deploying the schedule model to real hardware. The simulation tools which are used by companies in an industry perform scheduling simulation and visualisation of all time critical tasks to design and verify the model. All of them lack the capability of comparing simulation results versus real results to achieve the optimised scheduling design. This sometimes leads the overestimated worst-case execution times and increased system cost. The aim of our study is to decrease the system cost with optimisation of scheduled tasks via using the static analysing method.

Keywords: Schedule visualisation, scheduler optimisation, functional safety, real-time systems, scheduler.

* ADDRESS FOR CORRESPONDENCE: **Berkay Saydam,** Red Pine Software, Akdeniz, 1353. Sk. No:2, 35210 Konak/İzmir, Turkey.
E-mail address: berkay.saydam@redpine.com.tr

1. Introduction

Safety is very important issue in the automotive industry. In a system that depends on commands and functions which should be executed correctly, functional safety is the critical. Therefore, functional safety should be provided at the process of development and integration of automotive functionalities [3]. ISO 26262 defines many safety features and introduces safety mechanisms and techniques. Memory protection is used for avoiding undesired changing of data of Automotive Safety Integrity Level (ASIL) relevant applications. The scheduler provides safety by ensuring the tasks activated are scheduled by priority. The operating system also should be protected and undesired modification should be avoided [4]. Related works are mentioned in Section 2and safety issues in the automotive industry are mentioned Section 3 of this article.

Today, real-time systems are common with increasing complex systems, like automotive systems. Producing result values at the correct time is the main task of these systems. Late reactions could cause dangerous events. Therefore, systems are separated according to impression of late reaction, such as hard real-time, firm real-time and soft real-time. Detailed information about real-time systems is provided in Section 4of this article [1].

The scheduler is a management software that is responsible for the process of determining which process will use a resource for execution. Its scheduling policy depends on the safety level of systems. The scheduler processes a policy in terms of constraint such as timing, precedence and resource. The parameter of time is very important for real-time systems and it can be used to classify real-time systems. Scheduling terms are explained in another study [9].

Schedule tables are created in compile time theoretically; however, operations do not come true as hoped in real time. There should be measured values in runtime to overcome this condition. These values also help us to optimise the behaviour of systems. In our work, we determined two steps to achieve an optimised scheduling model. As the first step, the theoretical scheduling map is generated and visualised according to worst-case execution times values of the schedule model. In the second step, which is different and unavailable in any other solution, we run the schedule model on a real-time platform for a hyper period and it outputs a measured schedule map relative to the theoretical scheduling map. This relative map gives us the delta of the excepted worst-case execution times versus measured execution times and execution order. The schedule map is used to optimise the predefined schedule model to have well-balanced execution times and correct order for the critical tasks leading to a cost-effective system which complies with functional safety. There are implementation steps suggested in Section 6 of this article, and we conclude this study in Section 7.

2. Related work

There are various scheduling analysis tools with visualisation support. They have pros and cons [10]. STRESS is a good tool, but it only supports hard real-time systems. Therefore, it cannot be used for soft and firm real-time systems. MAST is interested with theoretical concepts and checks whether a given input can be schedulable or not. ARTISST shows the result of the simulation that takes the context switch into consideration; however, it is not capable of supporting shared resources. Cheddar supports shared resources and has a very powerful visualisation interface, but real simulation is a future study for this tool.

With regard to related work, all of them have some disadvantages. Different problems arise which are not expected from hardware, and these tools cannot test this situation. In our approach, which is different and unavailable in any other solution, we run the schedule model on a real-time platform for a hyper period. The measured scheduling map is an output from the hardware side. Actually, a comparison between these theoretical scheduling map and theme asured scheduling map is provided by our approach.

3. Safety in the automotive industry

ISO 26262 defines standards to provide functional safety in automotive applications. A hazardous event is a relevant combination of a vehicle situation that has the potential to lead to an accident or a vehicle-level hazard if not controlled by the timely action of the driver. A safety goal is a requirement that has the purpose of reducing the risk of hazardous events to a tolerable level [5].

3.1. Automotive safety integrity levels

ASILs represent risk-based classification of a safety goal in the automotive industry. The ASILs range from ASIL D to quality management level (QMlevel). ASIL D is the highest classification of the risk of injury. QM represents applications that cause no automotive hazards [6]. For instance, the braking system plays a crucial role for safety topics in the automotive industry. The elements of this system can be determined as ASIL D. On the other hand, in-vehicle infotainment systems do not provide reliability in the automotive industry. These systems can be marked as ASIL A or QM.

3.2. V-model

V-Model is a development process which includes the verification and validation phases. Phases verify each other with this model. First of all, requirements are determined by the customer. The general architecture is designed in the next phase. This architecture has modules and they are called units. Another step is the designing of units one by one. These steps are carried out by the architect and developers. Implementation phase is started after all units are designed and this phase is the last phase for developers. In the next stage, a tester tests to these units. After that, all the systems are tested by the tester in the integration testing phase. The last phase is the verification of safety requirements. In this step, all requirements, which are determined in first phase, are verified.

4. Real-time systems

A system is a black box that has a set of one or more inputs and outputs. A real-time system is defined as a system that must react within precise time constraints to events. Therefore, time is the most significant factor to be managed. A deadline is a time constraint which represents the time for successful process completion [8].

Real-time systems have different characteristics and specifications. These systems have higher precision than other systems with regard to time constraint. They can give fast response to any reaction. These characteristics are used in safety-critical systems and, of course, these systems cause an extra cost. Therefore, these systems are used in critical systems.

There are three types of real-time systems, depending on the consequences of a missed deadline. If missing a deadline still provides some utility, then it is a soft real-time system. A firm real-time system does not cause damage to the system by missing its deadline, but its output has no value. The last one is the hard real-time system that has a very strict time constraint. If its deadline is missed, then it might cause disastrous consequences [1]. Some scheduling algorithms also are created to support this parameter. The earliest deadline first scheduling algorithm takes deadlines as an argument to determine tasks according to its rules.

Pre-emption is very important method in hard real-time systems. A system can have periodic, a periodic and sporadic tasks. The scheduler cannot handle sporadic tasks or missed deadlines. In these conditions, pre-emption can be used in the safety critical systems. Pre-emption supports us by taking a high priority task to the running state when a low priority task is run and the running task is retained in the waiting state at that moment. It runs again after the high priority task is completed. Thus, this condition can be handled by the scheduler.

In reliable real-time systems, worst-case execution times (WCETs) is the maximum length of time for the execution of a task. It is defined for each task individually. WCET can be used to ensure pre-allocated timing budgets and be used as an input to scheduling analysis [2]If it defines more than necessary, it increases the system's cost.

5. Scheduling analysis in hard real-time systems

Scheduler is a manager that determines which process should be run on the central processing unit (CPU). Each embedded system has a CPU to manage their tasks. Tasks are carried out according to their states. They start in the hard drive as a program and are then passed to the ready queue by the CPU. Now, it is called a process. Then the scheduler selects the process which is desired according to the scheduling algorithm. The dispatcher takes that task to the running state. If there is a pre-emption because of a high priority task, then it goes to the ready state again. If it waits for an input/output (I/O) or event, it goes to the waiting state. When that process is finished, it is terminated. It uses scheduling algorithms to make decisions. Scheduling algorithms are designed according to the purpose of the system. Scheduling algorithms can be classified as static and dynamic. In static scheduling, some parameters of task are given, such as frequency, WCET before run-time. It requires the knowledge of task execution characteristic, but it provides low run-time overhead on the CPU [2].

An algorithm can produce the same output for a particular input, it is called deterministic algorithm. Offline algorithms can be used to provide determinism. In offline scheduling, there is a table which has lists of tasks and their activation times. This table is called the schedule table. A period is defined in the schedule table, and it is called hyper period. Tasks of a scheduling table repeats suring the hyper period, continuously [8].

Scheduling analysis is an analysis that provides testing of the scheduling algorithm and the scheduler system. Real-time systems should be tested and verified for critical sections and there should be time calculations for a task to carry out the scheduling analysis. The schedule table can be used for time inputs and a theoretical schedule map will be output of scheduling analysis. This process can be repeated for task execution in real-time. The output of this scheduling analysis can be called a measured schedule table [7].

Software development life cycle (SDLC) is a process that is used in software development to produce high-quality software and maintain it easily. The scheduling analysis tool makes the steps of the SDLC easier. A comparison between theoretical scheduling map and measured scheduling map can be used to optimise the system. Some tasks might be overestimated at compile time to ensure functional safety and it increases system cost. Rearrangement of predefined schedule model with balanced execution times can be used to decrease this cost.

6. Implementation and results

In the automotive industry, software and hardware developments are tedious processes. For that reason, an SDLC is followed up. We can say that a general SDLC is the V-model. In this model, there is certain phases which are mentioned in Section3. Some static analyse tools are used before runtime to decrease the developer or tester's effort. The visualisation system is very important to display the scheduling decisions of a real-time scheduler because there are many safety critical tasks. Scheduling algorithms for the real-time scheduler can be investigated and compared by their performance using the values of this visualisation. For these reasons, our work consists of two steps.

First is creating a theoretical scheduling map. In this step, the tasks are determined to provide functionality. These tasks have some attributes, such as name, id, priority, core-id, period and WCET. Each task has an id which is defined and unique. This id is used by the scheduler to access the task's attributes. For instance, Task 1can be accessed with id number of 62. Core-id is used to determine the running core of the related task. While period determines the frequency of the working operation,

WCET determines the deadline of that task. They have a unit section to give the measurement unit. Thus, this file can be manageable. There is a schedule table file that includes WCET values of tasks. The format of this file is shown in Figure 1.

```
<SOURCE-FILE-DATA>
  <TASKS>
    <TASK>
      <NAME>Task1</NAME>
      <ID>62</ID>
      <PRIORITY>0</PRIORITY>
      <CORE-ID>0</CORE-ID>
      <PERIOD unit="ms">50</PERIOD>
      <WCET unit="us">34500</WCET>
    </TASK>
    <TASK>
      <NAME>Task2</NAME>
      <ID>63</ID>
      <PRIORITY>0</PRIORITY>
      <CORE-ID>1</CORE-ID>
      <PERIOD unit="ms">20</PERIOD>
      <WCET unit="us">3400</WCET>
    </TASK>
    <TASK>
      <NAME>Task3</NAME>
      <ID>65</ID>
      <PRIORITY>0</PRIORITY>
      <CORE-ID>1</CORE-ID>
      <PERIOD unit="ms">10</PERIOD>
      <WCET unit="us">1600</WCET>
    </TASK>
    <TASK>
      <NAME>Task4</NAME>
      <ID>66</ID>
      <PRIORITY>0</PRIORITY>
      <CORE-ID>1</CORE-ID>
      <PERIOD unit="ms">48</PERIOD>
      <WCET unit="us">4000</WCET>
    </TASK>
```

Figure 1. The format of the schedule table

Necessary data are parsed, like the name of task, ID, priority and WCET, from this document. These values are used to generate a theoretical scheduling map. The visualisation of a generated map for a hyper period is shown in Figure 2. In Figure 2, the cores are classified as tabbing. The core-id variables of the schedule table are used to provide this classification. A timetable is found in the upper area of the visualisation tool. The measurement unit of a timetable, which is determined with unit variable in the schedule table, is given in the timetable. Tasks line up according to the related core from up to down. The name of tasks is given line by line. Working time of tasks are shown in their timelines. If there is a conflict between tasks, it can be seen here. Developers can do maintenance easily and can also design units with aid of this tool.

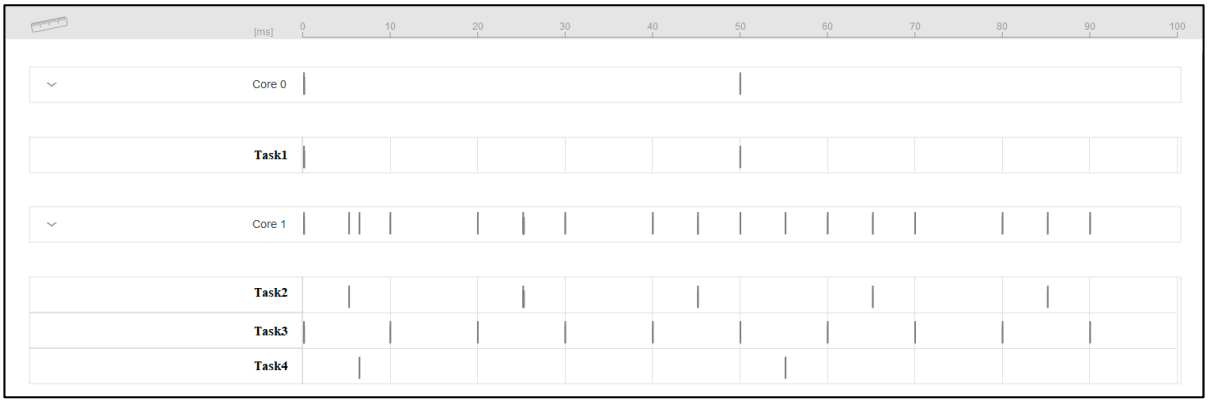

Figure 2. Theoretical scheduling map

In the second step, the schedule model is run on the real-time platform for a hyper period. The measured schedule map is compared with the theoretical scheduling map which is generated at step one and the map is generated that includes values of the measured schedule map relative to theoretical scheduling map. When we select a task in the timeline, the attributes of that task is shown by a tool. This is the best part of this tool. When we look at Figure 3, the start time of this task and end time of task are seen clearly. If we have any delay in this task, this condition is shown in the delay section. We can optimise our task according to this delay part. Block runtime and state are also shown in this section. We can control our task whether is background or foreground. The difference between the two maps is seen in Figure 3 for a task.

Figure 3. Difference between theoretical and measured maps for Task 1

Figure 3 shows that there is a difference in the end time of Task 1 between theoretical and measured (real) maps. WCET of Task1 is higher than the measured value, but it can be decreased. We can make this decision with our tool easily.

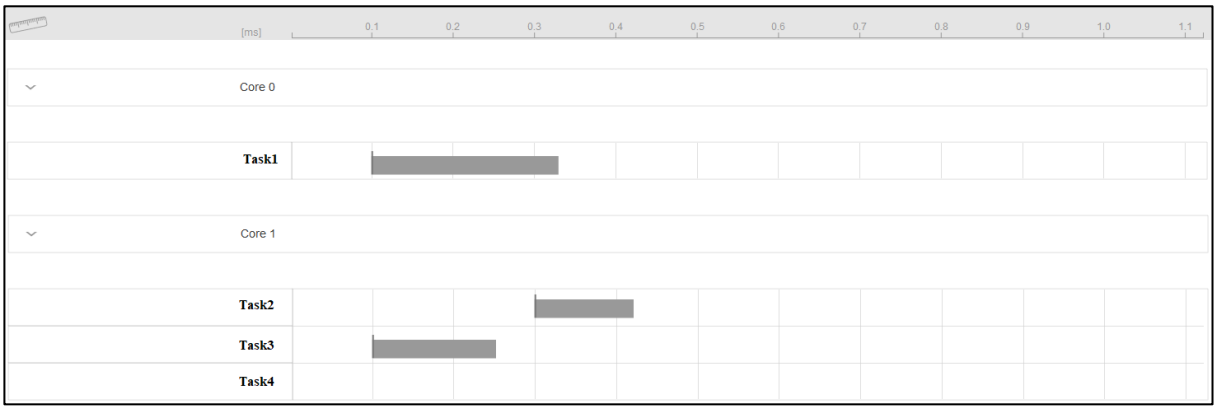

Figure 4. Detecting an incorrect WCET case

In Figure 4, another scenario is taken from the measured scheduling map. In the scheduling model, the WCET values of Task 2 and Task 3 are 0.1 milliseconds and 0.2milliseconds, respectively. The execution time of Task 2 is higher than its WCET value. In addition, the WCET value of Task 3 is overestimated in the theoretical scheduling map. In this analysis, the WCET values can be optimised. Thus, Task 2 will not pass its time by adding time from Task 3.

This situation is not only valid for transferring time values between tasks but these data are also used to choose correct and cost-efficient chips. Moreover, developers are able to make a decision to adding new feature to chip easily.

7. Conclusion

In this study, WCET values for a task are determined and a schedule table is arranged. The schedule model is run on the real-time platform for a hyper period. The measured schedule map is compared with the theoretical scheduling map which is generated at step one and the map is generated that includes values of the measured schedule map relative to theoretical scheduling map. According to the tests, the execution time of the second task is higher than its WCET value. In addition, the WCET value of the third task is overestimated in the theoretical scheduling map. In this analysis, WCET values can be optimised. Thus, the second task will not pass its time by adding time from the third task.

With this study, CPU usage became more efficient and well-balanced execution times and correct order decreased cost. Runtime tasks have been also tested to provide reliability with the measured scheduling map. Today, many systems work with a multicore. The selection of chip and core depends on the WCET of tasks. The most appropriate cost/performance can be determined with selection of chip and core. An appropriate chip can be determined by using the real execution time value of the measured schedule map. With real task distribution and real core utilisation, we will be able to obtain capability of the existing system and what it can do in the future.

For future study, sporadic tasks can be overcome in runtime with machine learning method, which use the values of a measured scheduling map. This study leads to a study that includes the optimisation of system capacity and updating of schedule tables in a certain range dynamically. Multiple schedule tables can be defined by compile time. These schedule tables are exchanged between them according to the current situation of a system at runtime.

References

[1] Buttazzo, G. C. (2011). *Hard Real-Time Computing Systems: predictable scheduling algorithms and applications* (3rd ed.).Boston, MA: Kluwer Academic Publishers.

[2] Hermann, K. (2011). Real-time systems - design principles for distributed embedded application. Berlin, Germany: Springer.

[3] ISO: 26262. (2018a). Road vehicles-function safety part 1: vocabulary.

[4] ISO: 26262. (2018b). Road vehicles-function safety part 6: product development at the software level.

[5] ISO: 26262. (2018c). Road vehicles-function safety part 2: management of functional safety.

[6] ISO: 26262. (2018d). Road vehicles-function safety part 9: automotive safety integrity level (ASIL)-oriented and safety-oriented analyses.

[7] Leung, J. & Zhao, H. (2005). *Real-time scheduling analysis*. Washington**,** DC: Office of Aviation Research and Development, DOT/FAA/AR-05/27.

[8] Phillip, L. (1993). *Real-time systems design and analysis, an engineer's handbook*. Piscataway, NJ: IEEE Press.

[9] Saltzer, J. H. & Frans Kaashoek M. (2009). Principle of computer system design: an introduction. Cambridge, MA: Massachusetts Institute of Technology.

[10] Shabir A., Sehrish M. & Do-Hyeun K. (2018). Comparative analysis of simulation tools with visualization based on real-time task scheduling algorithms for IoT embedded applications. *International Journal of Grid and Distributed Computing, 11*(2), 1–10. doi:10.14257/ijgdc.2018.11.2.01