A COMPARISON OF REAL-TIME TASK SCHEDULING METHODS IN SPACECRAFT SIMULATION

Mehmet Emin Güllüoğlu[1] and Mehmet Reşit Tolun[2]

[1]Department of Computer Engineering, Baskent University, Ankara, Turkey
TAI, Turkish Aerospace Industries, Ankara, Turkey
[2]Department of Computer Engineering, Baskent University, Ankara, Turkey

ABSTRACT

Today, embedded real-time applications play an important role in modern life. Satellites are also robust embedded real-time applications. A satellite project can cost over three-hundred million dollars. As many satellite manufacturers validate their satellites before launching, satellite simulators play the most valuable role in validation infrastructures. Specifically, satellite flight software validation has become more important. In this paper, we focused on the round robin (RR), rate monotonic (RM), and event driven (ED) real-time scheduling task methods with respect to their CPU usage performance for satellite simulator infrastructures. The tasks are evaluated and tested by real-time executive for multiprocessor systems (RTEMS). Those scheduling tasks are used in polling mode in the simulation setup. In this study, we compared three task scheduler methods for attitude orbit control system tasks and MIL-STD 1553 bus data distribution controller tasks in a spacecraft simulator environment. The results were close and the values were not segregated, thus, we chose RR and ED, because RR was easy to implement and ED allowed for full control of the tasks.

KEYWORDS

Real-time embedded systems, Real-time operating system, Rate monotonic task, Round robin task, Event driven task handling, and Satellite simulations

1. INTRODUCTION

Spacecraft development has been remarkably changed and optimized by modern simulation methods. Spacecraft manufacturers need to be sure the spacecraft can achieve its mission. Satellite simulators play the most valuable role in validation infrastructures. Hence, satellite simulation is an important issue in space craft simulations, such as how well you can simulate your system, based on models and the controller. Simulation frameworks are provided to run models. The satellite equipment, space environment, and satellite dynamics have to be represented by the models. The aims of a consummate simulation contain the representative models that are given at below:

- Satellite equipment models
- Space environment model
- Satellite dynamic model

The general purpose of running satellite models in a simulation framework is spacecraft flight software validation.

David C. Wyld et al. (Eds) : NeCoM, SEAS, SP, CMCA - 2018
pp. 75–84, 2018. © CS & IT-CSCP 2018 DOI : 10.5121/csit.2018.80906

Spacecraft software is run on real-time embedded system called an onboard computer (OBC). All control algorithms are run from there and it handles data management via a uniprocessor. The spacecraft controller is run on a real-time operating system (RTOS) on the OBC. The RTOS was provided with a task scheduler method to use our flight software. Task optimization and performance is directly related to the flight software performance. Therefore, the aim of this study was to compare the real-time task scheduler method in a spacecraft simulation.

In our study, we focused on two controllers, the 1553 bus data distribution controller (1553) and attitude orbit control system (AOCS), and the three task scheduler methods chosen for comparison were:

- Round Robin (RR)
- Rate Monotonic (RM)
- Event Driven (ED)

Tasks for the 1553 and AOCS were run for the three scheduler methods.

2. THE SPACECRAFT SIMULATION

Space mission projects are unique and require a big budget, thus, it must be validated for a successfully mission life before launching. Simulation is the most used process method to validate equipment and verify the satellite system's level of requirement. Validation infrastructures vary according to their purpose.

2.1. Functional Verification Bench (FVB)

For most spacecraft projects, this is used for limited algorithm verification of the AOCS. The OBC is not represented in this model; only the control algorithms are designed and its functionality is tested (Figure 1) [1]

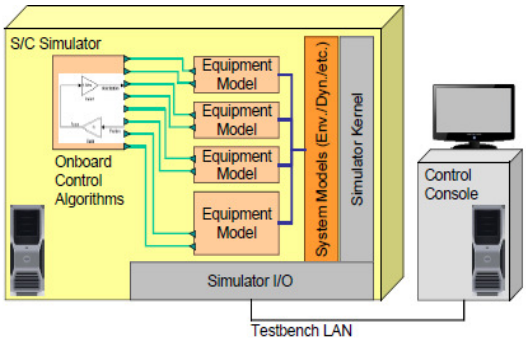

Figure 1 Functional Verification Bench

2.2. Software Verification Facility (SVF)

This facility is used for verification of the interfaces, connections, and standards to be sure that they work with each other properly. The OBC could be a model or emulator. The emulator is a perfect replica the OBC (Figure 2). [1]

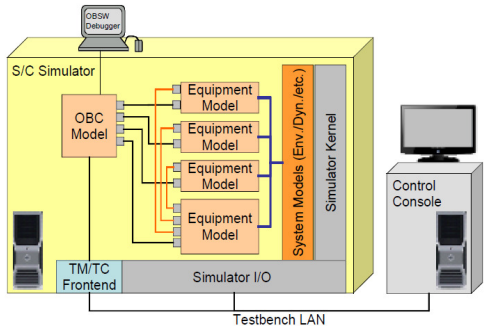

Figure 2 Software Verification Facility

2.3. Hybrid Verification Infrastructure

The hybrid verification infrastructure generally meets the real hardware OBC for verification of the satellite software on the real hardware. This infrastructure has the advantage of a directly tested I/O interface (Figure 3). [1]

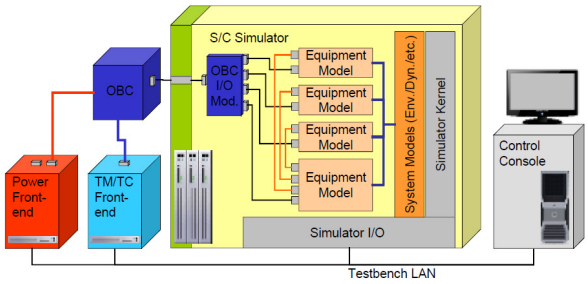

Figure 3 Hybrid Verification Infrastructure

2.4. Avionic Test Bench (ATB)

The ATB is the most improved simulation for the satellite. Verification engineers carry out their design. This bench can be added not only the equipment model, but also the real equipment that is needed to project the constraints (Figure 4). [1]

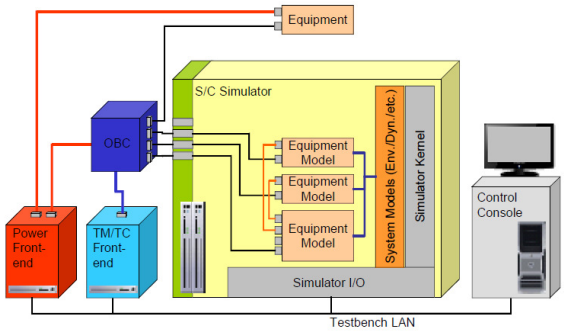

Figure 4 Avionic Test Bench (ATB)

2.5. Dynamic Satellite Simulator (DSS)

This simulator is generally used for telemetry and tele-command verification. Satellite operators are always educated about the DSSs, as it is very important in the launch and early orbit phase

(LEOP), because it aids in satellite survival. When the LEOP is not pre-worked, it causes satellite loss. The DSS provides pre-work before the LEOP, lunching, or nominal operations (Figure 5). [1]

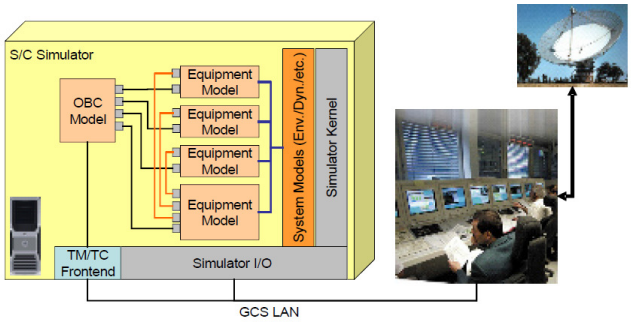

Figure 5 Dynamic Satellite Simulator (DSS)

3. RTEMS (REAL-TIME EXECUTIVE FOR MULTIPROCESSOR SYSTEMS)

RTOSs run in the OBC a part of the flight software. Hard RTOSs provide a restricted time in case the safety critical systems use hard RTOSs. Satellites are safety-critical systems and any unexpected delays may cause a catastrophic output, which can result in satellite loss.

RTEMSs are the most useful hard RTOSs on satellites. They provide high-performance peripherals for embedded real-time systems, offering characteristics to support the development of real-time embedded applications that are available for different platforms and architectures, including SPARC leon3 FT, which is one of the architectures used in this study [6]
RTEMSs provide the following features:

- multitasking
- homogeneous multiprocessor systems
- heterogeneous multiprocessor systems
- interrupt management
- event-driven inter-task communication, priority-based, pre-emptive
- rate monotonic scheduling
- synchronization and inter-task communication
- high level of user configurability
- dynamic memory allocation
- priority inheritance

The internal architecture of RTEMSs can be observed as layered components that work in harmony to provide a set of services for a real-time application system. (Figure 6) [6]

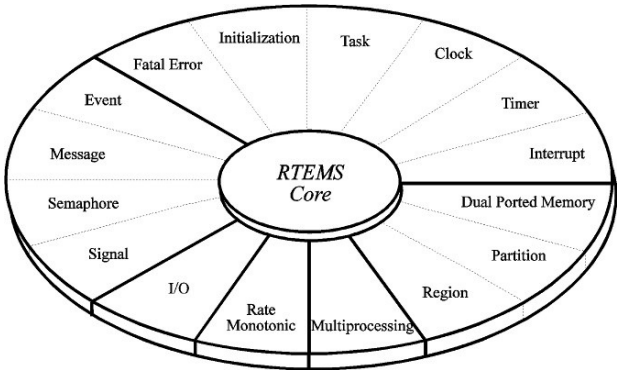

Figure 6 RTEMS resource managers

The functions utilized by multiple managers, such as scheduling, dispatching, and object management, are provided in the executive core. In this study, we focused on task manager services, rate monotonic services, and event services. [6]

4. STUDY ENVIRONMENT

An important issue in space craft simulations is how much simulations are in your system, based on models and the controller. The system must be close-looped and the controllers must be feel the same as in orbit. Hence, we developed the test environment of a hardware OBC, simulation framework, and MIL-STD 1553 data bus to meet the requirements of a spacecraft (Figure 7)

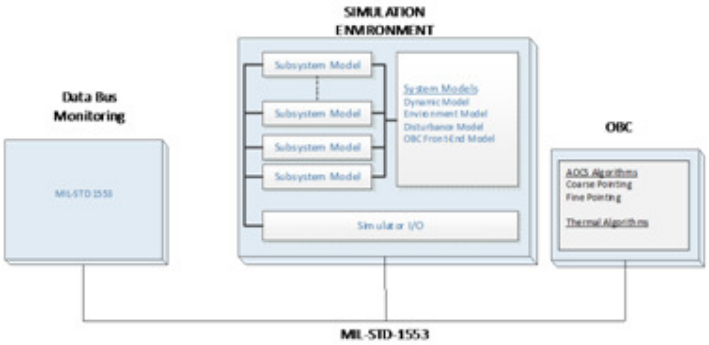

Figure 7 Test Environment

4.1. On Board Computer (OBC)

The OBC is the flight computer of a satellite. All of the control algorithms are run from there, and it handles data management via a uniprocessor [3] SPARC leon3 FT, which is especially designed for space missions. [4]

The LEON3 is a VHDL model of a 32-bit processor that is synthesizable and can accommodate IEEE-1754 (SPARC V8) architecture. LEON3 is an addendum of the LEON2 processor, with a 7-stage pipeline (compared to LEON2's 5-stage pipeline), which supports asymmetric and symmetric multiprocessing (AMP/SMP). A multiprocessing configuration of as much as 16 CPU can be utilized. The LEON3 is a highly-configurable model, which is appropriate for system-on-chip (SoC) designs, featuring the following: [5]

- A SPARC V8 instruction set equipped with V8e extensions,
- A sophisticated 7-stage pipeline
- Advanced on-chip debugging support, equipped with instructions and a data trace buffer
- A local instruction and data scratch pad RAM of 1 to 512 kilobytes
- Robust and fully-synchronous single-edged clock design, high-performance: 1.4 DMIPS/MHz, 1.8 CoreMark/MHz (gcc –4.1.2),
- A wide range of software tools, such as compilers, kernels, simulators, and debugging monitors.
- An AMBA-2.0 AHB bus interface
- Up to 125 MHz in FPGA and 400 MHz on 0.13 um ASIC technology

In the test setup, the OBC featured the following:

- LEON-3 FT core
- RTEMS operating system (OS)
- MIL-STD-1553, CAN, Ethernet
- Control Algorithm Integration:
  - AOCS Algorithm
  - Data Management
  - Sensor Data Analysis
  - Actuator Control
  - Mode management

4.2. MIL-STD 1553 Bus

The Mil-Std-1553B or Milbus has the standard defining characteristics of a serial multiplex data bus. The standard is a set of requirements covering the mechanical, electrical, and functional aspects of the bus. The bus aims at interconnecting via a single-medium avionics subsystem. [6]

MIL-STD-1553 includes three kinds of bus users, known as terminals, including a bus controller (BC), remote terminal (RT), and bus monitor (BM). The bus transaction is a command/response. The BC behaves like a master and begins all of the transactions. The RTs, controlled by the BC, supply the interface between the 1553 bus and the appropriate unit/sub-system. The BM remains passive and is a bus traffic recorder. [6]

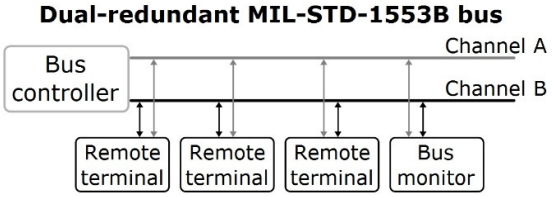

Figure 8 MIL-STD 1553B bus

4.3. Simulation Framework

The simulation framework provides the equipment and environment models in the polling mechanism. It also features the full scope of the problems to be investigated, ranging from the design and to the overall system simulations for the analyses of the dynamic system operation [1]. In our study, the simulation framework was run at ten Hz. The aims of the consummate simulations as contained representative models are given below:

- Star tracker model
- Sun sensor model
- Magnetometer model
- GPS model
- Magnetic toque bar model
- Reaction wheels' model
- Propulsion model
- Communication subsystem model
- Power subsystem model
- Space environment model
- Satellite dynamic model

5. TASK SCHEDULERS

Space craft flight software must include both robustness as well as hard real-time. Moreover, the OS should manage to tasks properly. In space craft flight software, many tasks are used separately in the control flight. The flight software's performance is related to the overall performance of the tasks; therefore, the task scheduler should operate at optimum performance for the flight controller. We implemented three different task scheduler models; round robin (RR), rate monotonic (RM), and event driven (ED). Static priority-driven pre-emptive [3] approaches [7] were also employed in our study.

In our study, all of the tasks were identified as the highest priority. RTEMS OS layers (Figure 9) is showed in the Figure 9. The AOCS and 1553 bus controllers were appointed as the test tasks

that would be used to compare the CPU performances. We focused on two controllers: the 1553 bus data distribution controller (1553) and AOCS, as well as three task scheduler methods chosen for comparison, which were RR, RM, and ED.

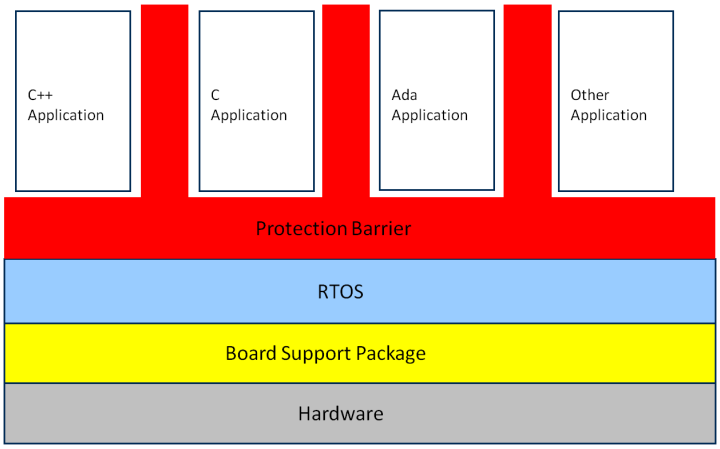

Figure 9 RTOS layer

5.1. Round Robin

RR scheduling is a task scheduling algorithm that is designed to be equitable. It uses time slices, also known as time quanta, which are assigned to each task in the queue. Each task is allowed to use the CPU for a given amount of time, and if it does not finish within the allocated time, it is pre-empted, and then returned to the back of the queue, so the next process in the queue can use the CPU for the same allocated time. [8]

5.2. Rate monotonic

The RM manager supplies facilities for implementing tasks that take place in a periodical manner. It also collects data regarding the implementation of those periods and is able to contribute relevant statistics that can be used when analysing and tuning the application. [2] [8]

5.3. Event driven

The ED is associated with interruptions and event managers that provide a high-performance method of inter-task communication and synchronization. The first task is invoked by interruptions from external devices, and after that, other tasks are invoked by previous tasks. Overload windows are able to miss deadlines; therefore, task times must be pre-calculated and those times must fit into a specific window; otherwise, the deadlines will exceeded [9].

6. DISCUSSION

Herein, we implemented three task schedulers, the RR, RM, and ED. The working frequency was appointed as ten Hz (100 ms). The AOCS and 1553 bus controller were appointed as the test tasks that would have their CPU performances compared. During the tests, every scheduler was run 1000 times and the tasks were identified as the highest priority [7]. The metric of performance was established as the “RTEMS CPU usage service” [8] for comparing the task performances. The results are given in the Table.

Table 1 CPU Usage (%)

	RR%	RM%	ED%
Idle	83.989	82.195	83.342
AOCS	15.727	17.495	16.234
1553	0.001	0.001	0.108

7. CONCLUSION

In this paper, we compared three task scheduler methods in a spacecraft simulator environment. The result were close and the values were not segregated. In this case, the task scheduler methods were investigated and we learned which task was suitable for the study environment and study cases. For example, RR was based on time slicing, so, if RR was chosen, the case should use preemptive, numerous tasks at a short rate. RM [2] always follows the rule of the shorter a task's run period, the higher its priority, or the longer a task's run period, the same is its priority [4]. Hence, the task period of the AOCS was longer than task period of the 1553 with the same priority, so the AOCS was run first. ED provides full control of the task run. If it is chosen, the task designer decides which is first, which is next, and which is last. Hence, we decided to run the 1553 task first and run the AOCS after that. In this study, we chose RR and ED because RR was easy to implement and ED provided full control of the tasks. Future work, we will add one more task, known as the thermal task and we will compare stack (memory) size of tasks.

REFERENCES

[1] J. Eickhoff, "Simulation Tools for System Analysis and Verification" in Simulating Spacecraft Systems, 1st ed., New York, 2009.

[2] C. Liu and J. Layland, "Scheduling algorithms for Multiprogramming in a Hard Real-Time Environment", Journal of the ACM, 20(1), 1973, pp. 46–61.

[3] F.F. Lindh, T. Otnes, and J. Wennerstrom, "Scheduling algorithms for real-time systems", Sch. Comput. Queen's Univ. Tech., 2005.

[4] M. Bashiri S. Ghassem "Performability Comparison of Schedulability Conditions in Real-Time Embedded Systems", 2010 Third International Conference on Dependability, 2010, pp 70–75.

[5] http://www.esa.int/Our_Activities/Space_Engineering_Technology/Onboard_Computer_and_ Data_Handling/Microprocessors.

[6] https://www.esa.int/Our_Activities/Space_Engineering_Technology/Onboard_Computer_and_ Data_Handling/Mil-STD-1553.

[7] K. Ramamritham, J.A. Stankovic "Scheduling Algorithms and Operating Systems Support for Real-Time Systems", IEEE Xplore, 1994.

[8] RTEMS C User's Guide (https://docs.rtems.org/releases/rtems-docs-4.11.2/c-user/index.html).

[9] M. Coutinho, J. Rufino, and C. Almeida. "Control of event handling timeliness in RTEMS". In: Proceedings of the 17th IASTED International Conference on Parallel and Distributed Computing Systems - PDCS 2005, Phoenix, Arizona, USA, 2005. IASTED.

AUTHORS

Mehmet Emin Güllüoğlu is Spacecraft Flight Software System Engineer at Turkish Aerospace Industries (TAI) where he has been since 2007.From 2007 to 2013 he worked at TAI as an Avionic system engineer. From January 2013 to January 2014 he worked at Thales Alenia Space – Cannes, France as an AIVV engineer. From January 2014 to October 2014 he worked at Thales Alenia Space – Cannes, France as a Functional Chain Validation engineer. From October 2014 to today he worked at TAI eventually as a Flight software engineer. His research interests is RTOS task schedulers, Spacecraft simulations and Software integration.