Simulation and Analysis of In-Orbit Applications under Radiation Effects on COTS Platforms

Nandinbaatar Tsog*, **Saad Mubeen***, **Moris Behnam***, **Mikael Sjödin***, **Fredrik Bruhn***†
*Mälardalen University
Box 883, 721 23 Västerås, Sweden
firstname.lastname@mdh.se
†Unibap AB (publ)
Svartbäcksgatan 5, 753 20 Uppsala, Sweden
f@unibap.com

Abstract—**Radiation effects research is crucial as it defines risk to both human bodies and spacecraft. Employing radiation-hardened products is one way to mitigate radiation effects on in-orbit systems. However, radiation effects prohibit most of the state-of-the-art commercial off-the-shelf (COTS) technologies from use in space. Furthermore, radiation effects on software components are less studied compared to hardware components. In this work, we introduce a simulation tool that simulates and performs post-simulation analysis of the impact of radiation effects on schedulability of the software task sets that execute on COTS system-on-chip (SoC) platforms within in-orbit systems. In order to provide a meaningful verification environment, single-event effects (SEEs) are introduced as aleatory disturbances characterized by probability distribution of occurrence using their predefined models. The tool supports interoperability with several other tools as it uses the extensible markup language (XML) model files for input and output, i.e., for importing input task sets and radiation effects and exporting the simulation and analysis results. The proposed tool is extensively by running simulations using a use case of an in-orbit onboard monitoring system.**

TABLE OF CONTENTS

1. INTRODUCTION

Radiation effects increase the complexity of space explorations. The radiation challenge is crucial to consider risks on both biological and mechanical systems, including equipment used in orbit such as onboard computers. As radiation effects are cumulative on the one hand, although the dose of space radiation is mostly low, its risk increases by the total time traveled in space [1], [2]. This characteristic is described by total dose of radiation, i.e., total ionizing dose (TID). Developing shielding materials or radiation-hardened products in order to mitigate radiation effects in orbit components could worsen the other limitations such as size, weight, and power (SWaP), cost, and development time. On the other hand, particles such as electrons cause electrostatic discharge, single-event effects (SEEs). Therefore, radiation effects can hinder the usage of commercial off-the-shelf (COTS) technologies that have been successful in the systems used the earth, such as COTS system on chip (SoC), including the use of integrated graphics processing units (GPUs), which improve the quality of onboard data processing [3].

Technology advancements of COTS SoC accelerators bring the possibility of intelligent onboard data processing instead of transmitting all massive raw data to ground stations via narrow downlink. Examples of onboard data processing include image processing and smart decisions based on artificial intelligence (AI), to mention a few. However, system developers need to tackle radiation risks to the systems that use COTS SoCs. A task set is said to be schedulable if all tasks complete their executions before the corresponding deadlines. Schedualbility of the task set is its property that determines if the task set is schedulable or not.

The radiation environment of deep space and on the earth's surface or in low earth orbit (LEO) are different. The radiation in LEO even varies as the reason for solar activity fluctuations [4]. The study of radiation effects on the human body and materials of components used in-orbit systems, including hardware [5], [6], is well-known and on-going. However, the study of how the radiation effects impact at the software application level is a spotless research area due to their complex and broad characteristics covering various radiation types, different types of hardware, and die revision changes through each family of hardware [7].

A. Contributions

This paper aims at investigating and demonstrating the impact of radiation effects on the schedulability of task sets that run on COTS SoC platforms consisting of heterogeneous processing units. In this regard, the paper introduces a simulation tool, namely Mälardalen-Unibap Simulation Tool (MUST). The tool supports several types of probability distributions and models to describe the radiation effects in the simulation. Furthermore, the simulation tool is able to add processing units such as central processing unit (CPU), graphics processing unit (GPU), field-programmable gate array (FPGA) as using their settings. Our aim in this paper is to identify how the probability distributions of radiation affect the timing schedulability of the task sets using the simulation tool. Note that we consider the challenges arising due to radiation effects at the software (program) level, particularly at the granularity of operating system task and task sets. Hence, the proposed tool, MUST, can be useful for simulating the schedulability of task set under aleatory disturbances of radiations to devices when developers need to start considering unknown working

Swedish special characters (å, ä, ö) in the email addresses are replaced by a and o.
978-1-7281-7436-5/21/$31.00 ©2021 IEEE

environment such as space.

B. Organization

The rest of the paper is organized as follows. In Section 2, the system architecture of MUST and other background information are provided. The layout and implementation of the tool is discussed in Section 3. Experimental evaluation and a discussion of the tool is followed in Section 4. Related work and related tools are introduced in Section 5. Section 6 concludes the paper and discusses future work.

2. MUST: System architecture

A. System Model

The system model considered in Mälardalen-Unibap Simulation Tool (MUST) consists of a system S. The system S comprises of radiation effect χ, m numbers of devices $\{P_m\}$ employed in onboard computer platforms including heterogeneous processing units (such as CPU and GPU), and a task set Γ. A task set means a set of programs/applications such as threads in Linux. The system is represented by the following tuple:

$$S = \langle \chi, \{P_m\}, \Gamma \rangle$$

We consider the fixed priority preemptive scheduling policy for CPU scheduler and non-preemptive fixed priority scheduling policy for GPU scheduler [8].

B. Task Model

Each task $\tau_i \in \Gamma$ is executed periodically and described with its worst case execution time C_i, its activation period T_i, and its relative deadline D_i (the deadline considered from the beginning of its activation period), i.e.,

$$\tau_i = \langle C_i, T_i, D_i \rangle$$

In order to simulate the task set Γ and check its schedulability, all tasks will be executed for the time interval that is equal to the hyperperiod of all tasks, which is calculated as the least common multiple of periods of all tasks, i.e.,

$$HP(\Gamma) = LCM(T_i)$$

for all $\tau_i \in \Gamma$. In other words, a task τ_i could be executed several times during the hyperperiod $HP(\Gamma)$. Therefore, we consider jobs of task τ_i and j^{th} job of task τ_i is denoted as $\tau_{i,j}$. Jobs are the released execution instances of a periodic task in each period. Every task consists of sequential, parallel, sequential segments. Sequential segments highlighted with blue color can be executed only sequential manner using CPU, while parallel segments highlighted with red color can be executed either on CPU sequential or on GPU parallel manner as depicted in Figure 1. C_i is expressed with an idea of alternative executions for parallel segments [9] on heterogeneous processing units $\{P_m\}$.

As illustrated in Figure 1, this idea means that any parallel segment of a task can be executed on different processing units for different jobs, i.e., the execution of a parallel segment of a particular task is not always allocated to one particular processing unit. For example, while a parallel segment of job $\tau_{i,j}$ executes on CPU, the same parallel segment of job $\tau_{i,k}$ may execute on GPU in order to avoid using one particular processing unit. Because, the intensive use of one particular processing unit can consequence a bottle neck.

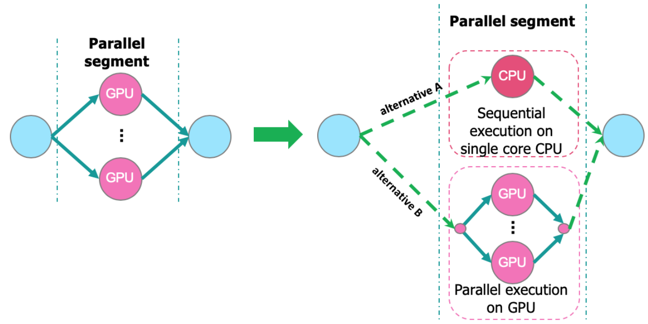


Figure 1. Alternative executions of parallel segment

C. Radiation Effect Model

The occurrence of a radiation effect generally follows the fault burst model [10], [11] in real-time systems as illustrated in Figure 2. The fault burst model describes the occurrence of multiple single radiation effects in radiation effect interval. This means that their distributions and total amount of radiation effects in the radiation effect interval can differ in each time, however, the burst of radiation effects can be bounded by the radiation effect interval.

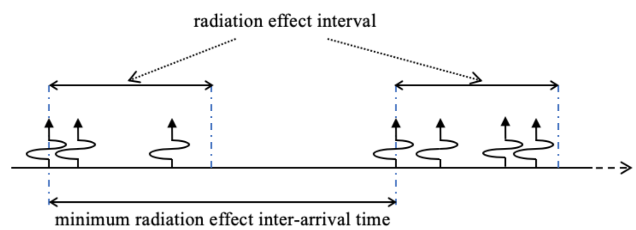


Figure 2. The fault burst model expressing radiation effects

Based on this model, we propose the following extended model of radiation effects employing probability distributions to radiations of environments. We consider that the system S performs under continuous radiation effects. However, the strength of radiation effect is distributed with the given probability distribution. Moreover, each device in the system S has different level of tolerances against radiations. We define it as a radiation tolerance of a device P_l (where $l \leq m$) and denote it as σ_l. Hence, the system can be executed normally under the following condition:

$$\chi(t) \leq \sigma_l \quad (1)$$

where t is the current clock tick and l is an index of the busy device P_l as executing a job $\tau_{i,j}$. A job $\tau_{i,j}$ needs to be re-executed if it does not satisfy the condition described in Equation 1.

In this paper, we consider the following four well-known probability distributions for radiation effect χ: i) uniform distribution, ii) normal distribution, iii) triangular distribution, and iv) exponential distribution.

3. MUST: Design and implementation

This section briefly discusses the inputs and outputs of the tool, design of the user interface of the tool, simulation

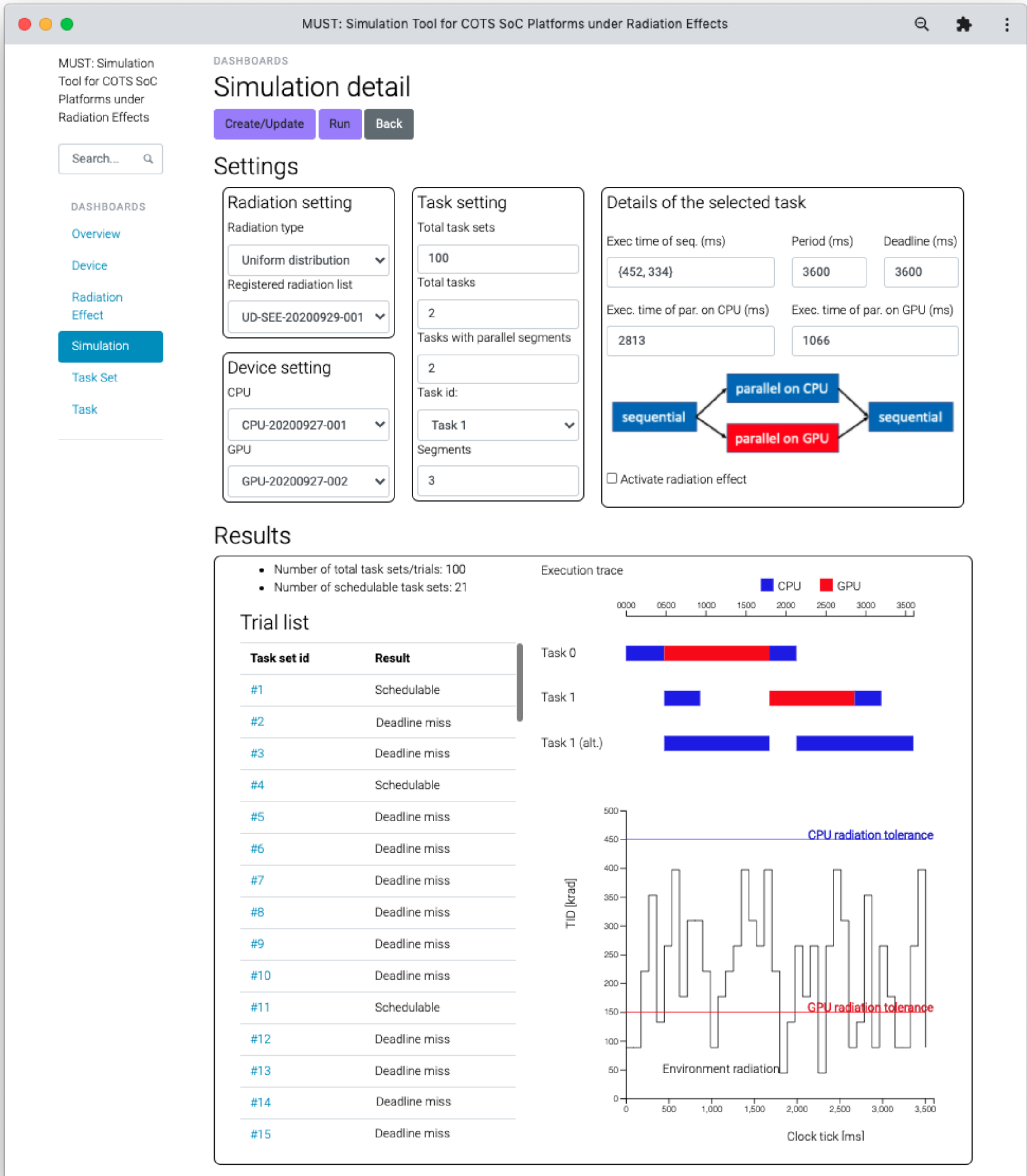


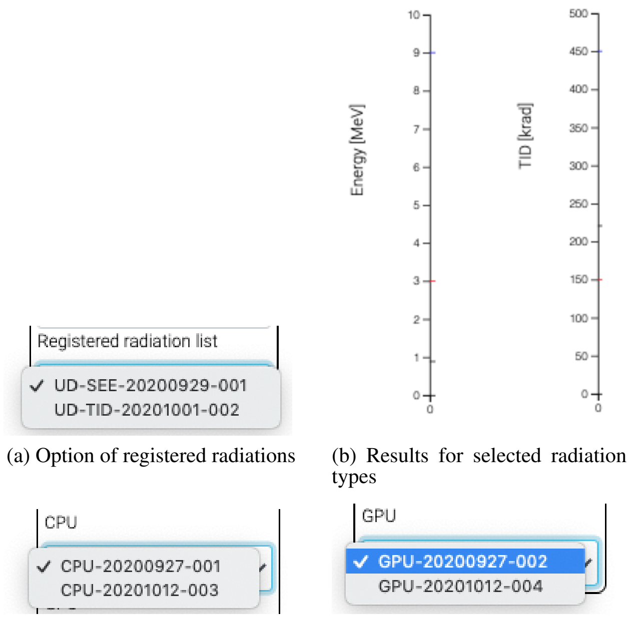
Figure 3. Layout of the MUST simulation tool under radiation effects

mechanism, and implementation and distribution of the tool.

A. Input & Output

This tool performs simulations of the systems based on the input provided by the user. The input consists of number and type of devices, task sets, number of tasks and their properties in each task set, and radiation effects according to the system, task and radiation effect models discussed in Section 3. The overview of the simulation page of this simulation tool is illustrated in Figure 3. The necessary setting parameters are located at the upper part of the simulation page of the tool.

As depicted in Figures 3, 4(a), 4(c), and 4(d), both radiation and device settings are easily selected from the list of registered radiation types and devices, respectively. On the pages of device and radiation effect, a user can register their related information such as the radiation tolerance for devices, and probabilistic distribution and radiation type (SEE or TID) for radiations. The tool allows the users to register new devices and radiation effects. This allows the users to utilize the tool for the devices with known properties as well as for the prospective devices that the users expect to have. Regarding creating a new radiation effect, any probabilistic distributions can be used along with the existing radiation models provided



(a) Option of registered radiations

(b) Results for selected radiation types

(c) Option of registered CPUs

(d) Option of registered GPUs

Figure 4. Option of settings

in the tool. For example, the tool has the default radiation effects based on AP-8 and AE-8 models [12].

As the main output of the tool, a number of schedulable task sets (i.e., the task sets in which all tasks completed their executions before the corresponding deadlines) is reported based on the simulation results. Furthermore, the detailed results of each simulation trial are illustrated in the result part of the tool. This includes the execution trace of each simulation and radiation graph with respect to the clock tick used in the execution trace. As shown in Figure 4(b), the radiation graph can be illustrated with either ”TID [krad]” or ”Energy [MeV]” on the vertical axis.

B. Design

The layout of the main pages are depicted in Figure 5. The tool consists of 3 parts, the input/setting part, the operation part, and the output/monitoring/archive part. As we discussed in the previous subsection, the device and radiation effect settings belong to the input part as illustrated in Figures 5(b) and 5(c), respectively. Based on the data received from the input part, the operation part performs simulations and produces analysis results as outputs. Hence, as depicted in Figure 3, the simulation page belongs to the operation part. The information generated through simulations is stored in the task lists (see Figure 5(d)). The created task set and radiation effects can be exported as an extensible markup language (XML) model file, which can be input to any other tool conforming to the XML format.

Further, as illustrated in Figure 4(a), the overview page shows the basic statistics of the tool such as the total number of the performed simulations and the created radiation effects.

C. Simulation Mechanism

Each task created in the tool has its own priority that depends on the setting. The different priority assignment policies such as rate monotonic (RM), deadline monotonic (DM) and earliest deadline first (EDF) can be used. In this paper, we consider only EDF priority assignment policy.

1) Generating a task set—First of all, the tool creates a task set with the assigned number of tasks. Each task is assigned an execution time, period, and deadline.

2) Assigning priorities—Since, we consider EDF, the priorities of tasks are dynamic. Thus, a job of the task with the earliest deadline gets the highest priority. In the case if two or more jobs of different tasks have the same deadlines then the priorities are assigned according to the ID numbers of the corresponding tasks, i.e., higher the ID of the task higher the priority of the corresponding job in case multiple jobs have the same deadlines. In order to perform a simulation, the allocation of parallel segments of each task to the appropriate processing unit should be handled.

3) Simulation—The simulation process continues until the clock tick reaches the time equal to the hyperperiod of the task set, $HP(\Gamma)$. At every clock tick, the simulator checks the priorities of tasks and selects the task that should be executed at this clock tick. The job of a task that has completed its execution is not considered until the release time of the next job that occurs periodically. Before executing a task, the simulator generates the environment radiation based on the type of radiation effect setting (TID or SEE). As the environment radiation is smaller than the radiation tolerance of the allocated processing unit (device) in the case of SEE, the simulator executes the task with 1 clock tick. In the case of TID, if the total exposure including current TID is still smaller than the radiation tolerance of the allocated device, the simulator executes the task with 1 clock tick as well. Otherwise, the simulator resets all the execution until this moment. This means that the task set starts from the beginning of the next clock tick, however, the current clock tick will continue. Then the simulator checks the deadline of the job. If the job misses its deadline, the simulator counts the deadline miss and ends this simulation trial.

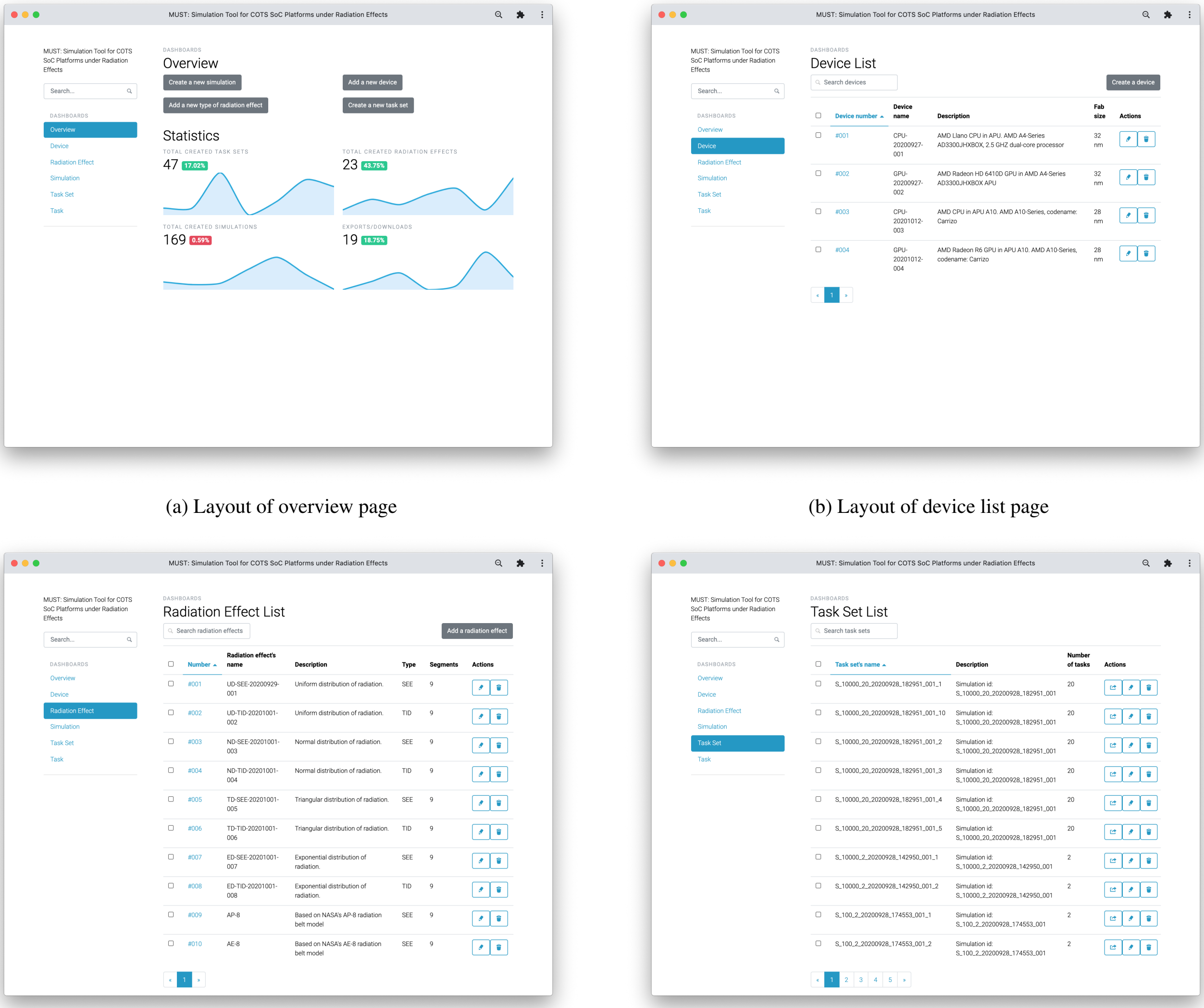
D. Implementation

The user interface of the tool is developed considering the web browser based solution using the MERN stack[2] in order to be less platform dependent. The MERN stack is a combination of four web technologies, MongoDB[3], Express JS[4], React JS[5] and Node JS[6]. The user interface is based on [13], where the detailed development guide of the MERN stack can be found.

The back-end of the simulator is implemented in Python programming language[7] using MongoClient, datetime, logging, random, time, and count libraries. Hence, the simulator is less dependent on underlying platforms. As using MongoDB as NoSQL, the data structure can be extended and customized easily without destroying data in the current experiment. This means that the tool can be easily adapted for new devices and radiation models.

The tool is published and distributed to GitLab repository page[8] freely. The tool can be extended with different types of radiation models, devices and new execution models, for example, the adaption of CRÈME96 [14], SPENVIS [15],

[2] https://www.mongodb.com/mern-stack
[3] https://www.mongodb.com/
[4] https://expressjs.com/
[5] https://reactjs.org/
[6] https://nodejs.org/
[7] https://www.python.org/
[8] https://gitlab.com/nabarja/must_aeroconf2021



(a) Layout of overview page

(b) Layout of device list page

(c) Layout of radiation effect list

(d) Layout of task set list

Figure 5. Layout of various pages in the MUST tool

TASTE [16], and Radeon™ GPU Profiler[9]. All changes regarding these extensions can be tracked on this page.

4. MUST: Use case

This section evaluates the MUST tool using a monitoring use case of an in-orbit system and discusses how the schedulability of task sets can be improved.

A. Use Case Description

The evaluation of the MUST tool is considered to apply a use case of monitoring system. This use case is inspired by a smallsat computer system [17], which presents a pre-operational task-set and logging software. Satellites consist of several peripherals and it is significant to monitor their abnormal activity. As depicted in Figure 6, an onboard computer (OBC) handles a monitoring system with two tasks, namely τ_0 and τ_1. Task τ_0 detects and collects status of peripherals to the storage employed in the OBC. Task τ_1 analyzes the stored logs and makes an appropriate decision such as sending report to the ground station, rebooting the system, and restarting a particular peripheral, and so on.

[9]https://gpuopen.com/rgp/

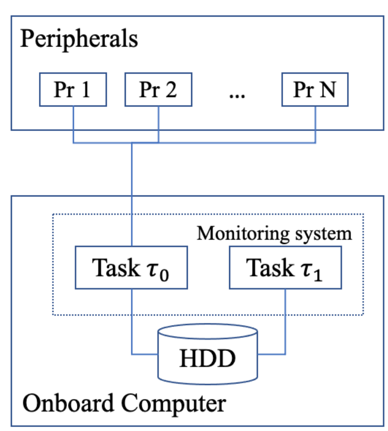


Figure 6. Use case: Monitoring system

The periods of τ_0 and τ_1 are 900 ms and 1200 ms, respectively. We consider implicit deadlines for the tasks. This means that the deadline of a task is considered to be equal to its period. Thus, the hyperperiod of the task set is 3600 ms. Each task has a parallel segment which is allocated to CPU. The execution times of the sequential, parallel, sequential segments of τ_0 and τ_1 are {20 ms, 20 ms, 30 ms} and {40 ms, 40 ms, 50 ms} respectively. The scheduling policy is EDF. Hence, the utilization of the task set is $U = 70/900 + 130/1200 = 0.186$, which satisfies the schedulability condition of EDF: $U \leq 1$.

The radiation tolerances of the devices, CPU and GPU, are given as 10 MeV. The environment radiation is created using the uniform distribution with the ranges' [1 MeV; 10.5 Mev]. This means that the environment radiation exceeds only 5% probability of the devices' radiation tolerance.

B. Evaluation and Discussion

Table 1 reports the simulation results of the experiment. There are 4 and 3 jobs of tasks τ_0 and τ_1 in the hyperperiod of 3600 ms. The table shows that each of two jobs of τ_0 within the hyperperiod complete their executions before and after the corresponding deadlines. In the case of τ_1, only one job completes its execution before the deadline, while two of its jobs miss their deadlines in the hyperperiod. Thus, the task set is not schedulable under the given radiation effect.

Table 1. Experiment results

Job $\tau_{i,j}$	**Execution time**	**Response time**	**Result**
$\tau_{0,1}$	70 ms	483 ms	pass
$\tau_{0,2}$	70 ms	>864 ms	misses deadline
$\tau_{0,3}$	70 ms	>268 ms	misses deadline
$\tau_{0,4}$	70 ms	169 ms	pass
$\tau_{1,1}$	130 ms	451 ms	pass
$\tau_{1,2}$	130 ms	600 ms	misses deadline
$\tau_{1,3}$	130 ms	729 ms	misses deadline

Let us focus on the job $\tau_{0,1}$. During 483 ms, the first and second segments of the job restarted 26 and 5 times, respectively. Furthermore, in this simulation, the first segment of any job needs to restart even if the second or the third segments experience the radiation effect while they are executing.

This experiment reveals that the smaller the size of the segment the lesser it is affected by the radiation effect. Hence, an important take-away from this experiment is that the schedulability of the systems under radiation effects can be improved if the the execution times of the jobs are partitioned into smaller chunks so that the partial execution results can be more frequently saved. Developing such a technique to determine optimized chunks of the executions is left for the future work.

5. RELATED WORK AND TOOLS

Space missions are limited to bring the technological advances in COTS platforms due to the radiation effects. There are many works that focus on measuring the behaviour of COTS platforms under radiation effects [7], [18], [5]. These works consider the effect of radiation regarding total ionizing dose (TID) and single-event effects (SEEs) on in-orbit hardware and materials used in the spacecraft. Miller et al. [6] and Troxel [7] consider the radiation effect on commercial DRAMs. The exposed particle can damage hardware, which can end up with data loss as well. Moreover, the authors mention the changes of chip revision within each family can be another concern of radiation effects. Therefore, the current state of the art focuses on how radiation effects can affect materials of hardware that, in turn damages the stored data. There is a lack of research on investigation of radiation effects on the execution behavior of applications that are stored in the hardware.

Besides performing radiation testing, the space missions predict the radiation effects using several existing tools. CRÈME96 is a state-of-the-art prediction tool for SEEs based on the Cosmic Ray on Micro-Electronics code that provides better description of the environment with ionizing radiations and improved calculations of single-event upsets (SEUs) [14]. CRÈME96 provides the prediction models that predict how cosmic ray affects microelectronics. The European Space Agency (ESA) provides space environment information system (SPENVIS) [15]. SPENVIS provides models of the hazardous space environment including cosmic rays, radiation belts, solar energetic particles, among others. OMERE[10] is a freeware radiation software dedicated to radiation effects to electronic devices in space environment. This tool is developed with the support of the National Centre for Space Studies (CNES) based on the industrial requirements from several organizations and companies. OMERE computes particle fluxes as the space environment, and dose, displacement damage, SEEs and solar cell degradation as radiation effects on electronic devices. We plan to add models from both CRÈME96, SPENVIS, and OMERE in future release of our tool.

Under the ESA initiative, TASTE is developed as a development tool-chain that targets heterogeneous, real-time, and embedded systems [16]. TASTE supports model-based development and provides early verification and testing of generated applications. We consider that bringing a possibility to import the TASTE generated applications as tasks to our tool can broaden the usability of it. Further, Radeon™ GPU Profiler provides the detailed execution trace of tasks on GPU computing. We plan to include it in the future release of our tool as this will help to consider the reality of task segments.

6. CONCLUSIONS

This paper introduced the architecture, design, implementation and simulation mechanisms of a new simulation tool for the task sets running on heterogeneous processing units that are subject to radiation effects. Furthermore, the tool performs post-simulation analysis to check the schedulability of the task set. The occurrence of radiation effects in this work is described with common probability distributions. As one of the outputs, the tool provides the rate of deadline misses among simulated task sets. The tool is designed to support interoperability with other tools that use the XML format for inter-tool communication. That is, the task sets and radiation models can be exchanged with the XML model files.

The preliminary experiment using the tool shows that a technique splits a task into small segments and guarantee to save their executed results from radiation effects can improve schedulability of task sets. As future work, the tool can

[10] https://www.trad.fr/en/space/omere-software/

be extended to interoperate with the existing tools such as the cosmic ray effects on micro-electronics CRÈME 96, ESA's space environment information system SPENVIS, and Radeon™ GPU Profiler. Further, although the tool includes the simplified NASA radiation belt models, AP8 and AE8, we continue to improve this simplified model in the tool.

Acknowledgments

The work in this paper is supported by the Swedish Knowledge Foundation through the projects DPAC and HERO. We thank all of our industrial partners involved in these projects, in particular Unibap AB (publ). Further on, this work is supported in part by a corporate scholarship of the TESO Corporation for Nandinbaatar Tsog.

References

[1] L. Walsh, U. Schneider, A. Fogtman, C. Kausch, S. McKenna-Lawlor, L. Narici, J. Ngo-Anh, G. Reitz, L. Sabatier, G. Santin, L. Sihver, U. Straube, U. Weber, and M. Durante, "Research plans in Europe for radiation health hazard assessment in exploratory space missions," *Life Sciences in Space Research*, vol. 21, no. March, pp. 73–82, 2019. [Online]. Available: https://doi.org/10.1016/j.lssr.2019.04.002

[2] "Space Faring The Radiation Challenge," https://www.nasa.gov/pdf/284273main_Radiation_HS_Mod1.pdf, accessed: 2020-02-04.

[3] L. L. Bello, R. Mariani, S. Mubeen, and S. Saponara, "Recent advances and trends in on-board embedded and networked automotive systems," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 1038–1051, 2019.

[4] L. M. S. Martines, "Analysis of LEO Radiation Environment and its Effects on Spacecraft's Critical Electronic Devices," *Dissertations and Theses*, p. 102, 2011. [Online]. Available: https://commons.erau.edu/edt/102

[5] R. Kingsbury, F. Schmidt, W. Blackwell, I. Osarentin, R. Legge, K. Cahoy, and D. Sklair, "Tid tolerance of popular cubesat components," in *2013 IEEE Radiation Effects Data Workshop (REDW)*, 2013, pp. 1–4.

[6] C. Miller, R. Owen, M. Rose, P. M. Rutt, J. Schaefer, and I. A. Troxel, "Trends in radiation susceptibility of commercial drams for space systems," in *2009 IEEE Aerospace conference*, 2009, pp. 1–12.

[7] I. Troxel, "Memory technology for space," *Military and Aerospace Programmable Logic Devices (MAPLD)*, 2009.

[8] L. Sha, T. Abdelzaher, K.-E. A. rzén, A. Cervin, T. P. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. P. Lehoczky, and A. K. Mok, "Real Time Scheduling Theory: A Historical Perspective," *Real-Time Systems*, vol. 28, no. 2/3, pp. 101–155, 2004.

[9] N. Tsog, M. Becker, F. Bruhn, M. Behnam, and M. Sjödin, "Static allocation of parallel tasks to improve schedulability in cpu-gpu heterogeneous real-time systems," in *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*, vol. 1, 2019, pp. 4516–4522.

[10] F. Many and D. Doose, "Fault Tolerance Evaluation and Schedulability Analysis ," in *Proceedings of the 2011 ACM Symposium on Applied Computing*, ser. SAC '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 729–734. [Online]. Available: https://doi.org/10.1145/1982185.1982344

[11] A. Burns, R. Davis, and S. Punnekkat, "Feasibility analysis of fault-tolerant real-time task sets," in *Proceedings of the Eighth Euromicro Workshop on Real-Time Systems*, 1996, pp. 29–33.

[12] C. E. Jordan, "NASA radiation belt models AP-8 and AE-8," RADEX INC BEDFORD MA, Tech. Rep., 1989.

[13] C. Bonneau, "Internship report," *Mälardalen University, Sweden*, 2020.

[14] A. J. Tylka, J. H. Adams, P. R. Boberg, B. Brownstein, W. F. Dietrich, E. O. Flueckiger, E. L. Petersen, M. A. Shea, D. F. Smart, and E. C. Smith, "CREME96: A revision of the cosmic ray effects on micro-electronics code," *IEEE Transactions on Nuclear Science*, vol. 44, no. 6, pp. 2150–2160, 1997.

[15] D. Heynderickx, B. Quaghebeur, E. Speelman, and E. Daly, "ESA's Space Environment Information System (SPENVIS)-A WWW interface to models of the space environment and its effects," in *38th Aerospace Sciences Meeting and Exhibit*, 2000, p. 371.

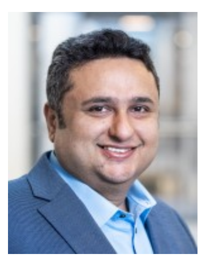
[16] M. Perrotin, E. Conquet, J. Delange, A. Schiele, and T. Tsiodras, "Taste: a real-time software engineering tool-chain overview, status, and future," in *International SDL Forum*. Springer, 2011, pp. 26–37.

[17] C. R. Julien, B. J. LaMeres, and R. J. Weber, "An fpga-based radiation tolerant smallsat computer system," in *2017 IEEE Aerospace Conference*, 2017, pp. 1–13.

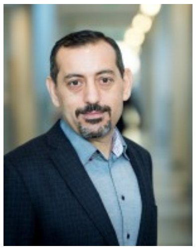
[18] D. Sinclair and J. Dyer, "Radiation effects and cots parts in smallsats," 2013.

Biography

Nandinbaatar Tsog *is a PhD candidate at Mälardalen University, Sweden. He received his B.S. in computer science from University of Electro-Communications, Japan in 2006. After few years of industrial experience, he finished M.S. in computer science specialized in intelligent embedded systems from Mälardalen University in 2016. The aim of his research is to investigate predictability in heterogeneous architectures such as HSA.*



Saad Mubeen *is an Associate Professor at the School of Innovation, Design and Engineering at Mälardalen University Sweden since March 2018. He holds a Master degree in Electrical Engineering specialization in Embedded Systems from Jönköping University, in 2009. He received his Licentiate and PhD degrees in Computer Science and Engineering from Mälardalen University in 2012 and 2014 respectively. He received his Docent title in Computer Science with a focus on "Developing Predictable Vehicular Embedded Systems" in 2018.*



Moris Behnam *is a professor in computer science with focus on Cyber Physical Systems. He has awarded a B.Eng., and M.Sc. in Computer and Control Engineering at the University of Technology, Iraq, and also M.Sc., Licentiate, and PhD in Computer Science and Engineering at Mälardalen University, Sweden, in 1995, 1998, 2005, 2008 and 2010 respectively. Moris has been a visiting researcher at Wayne State University, USA in 2009 and he has been a Postdoctoral Researcher at University of Porto in 2011. His research interests include real-time scheduling, synchronization protocols, multicore/multiprocessor systems, distributed embedded real-time sysems, using control theories in real-time scheduling, indutrial cloud computing and internet of things.*

Mikael Sjödin *received his PhD in computer systems 2000 from Uppsala University (Sweden). Since then he has been working in both academia and in industry with embedded systems, real-time systems, and embedded communications. The current research goal is to find methods that will make software development cheaper, faster and yield software with higher quality. Concurrently, Mikael is also been pursuing research in analysis of real-time systems, where the goal is to find theoretical models for real-time systems that will allow their timing behavior and memory consumption to be calculated.*

Fredrik Bruhn *received a Ph.D. in Microsystems Technologies from Uppsala University, Uppsala, Sweden in 2005 and a Masters of Science in Atomic and Molecular Physics from Uppsala University in 2000. He graduated from International Space University Summer Session Program in 2001. He has been with Mälardalen University since 2013 as Adjunct Professor in Robotics and Avionics. He has been a guest researcher at JPL and entrepreneur starting several high technology companies in robotics and space applications.*