Energy-Aware Fault-Tolerant Real-Time Scheduling for Embedded Systems

Hussein El Ghor[1](✉), Julia Hage[2], Nizar Hamadeh[1], and Rafic Hage Chehade[1]

[1] LENS Laboratory, Faculty of Technology, Lebanese University, B.P. 813, Saida, Lebanon
husseinelghor@ul.edu.lb

[2] Faculty of Technology, Lebanese University, B.P. 813, Saida, Lebanon

Abstract. In this paper, we investigated the problem of developing scheduling techniques for uniprocessor real-time systems that enhances energy saving while still tolerating up to k transient faults to preserve the system's reliability. Two scheduling algorithms are proposed: The first scheduler is an extension of an optimal fault-free energy-efficient scheduling algorithm, named ES-DVFS. The second algorithm aims to decrease the consumption of energy by using the slack time for the recovery operation when faults occur. The experimental results show that the proposed approach significantly reduces the consumption of energy when compared to the previous schedulers.

Keywords: Real-time scheduling · Fault-tolerant · Checkpointing · Energy management · Energy harvesting

1 Introduction

Many embedded real-time systems usually operate in harsh environments. To function correctly, they have to respect the timing constraints and at the same time decrease energy consumption even in the presence of faults. Therefore, besides their timing and energy constraints, these systems usually have serious fault-tolerant limitations.

Energy management is achieved by the most popular solution, namely dynamic voltage and frequency scaling (DVFS) [1,2], with the aim to reduce energy consumption during system operation and to prolong the battery lifetime by dynamically scaling down the processor supply voltage as much as possible and without violating the tasks deadlines.

In reality, processor faults can be categorized as: transient and permanent faults [3]. We focus in this paper on the transient fault since, in most computing systems, the majority of errors are due to transient faults [4]. In the case of an energy-efficient system, reliability also means ensuring that the system will never be short of energy to ensure its treatment. Anticipation of possible cases of energy can, again, be implemented on the basis of the flexibility offered by the system at the level of execution of tasks.

In this work, we focused on the problem of real-time scheduling under reliability and energy constraints. Its about considering real-time jobs that have needs which are expressed, on the one hand, in terms of processing time and energy consumed by the processor and, on the other hand, in terms of the number of tolerated faults. A job configuration is energy overloaded, this means that the amount of energy consumed is greater than the amount of available energy. In addition, the amount of execution time requested is smaller than the available capacity, the system will therefore typically be able to meet all its deadlines or else catastrophic consequences will occur. A major question that needs to be answered is: how to schedule real-time jobs with energy constraints where the system keeps reliable and able to tolerate up to k faults.

To answer this question, we first propose a uniprocessor Earliest Deadline First (EDF) scheduling algorithm and we then derive an exact and efficient feasibility condition by considering energy management and fault-tolerance. Second, the proposed algorithm is derived to achieve energy autonomous utilization of the processor while respecting deadlines of each task in the task set.

The rest of the paper is organized as follows. In the next section, we summarize the related work. In Sect. 3, we present the model and terminology. The fault tolerant speed schedule was then presented in Sect. 4. Section 5 shows through experimental results the energy savings of the proposed algorithms and Sect. 6 concludes the paper.

2 Related Work

In both industry and academia fields, researchers have found some techniques to enhance energy saving in embedded systems. Among these, DVFS has risen as one of the best framework level methods for energy consumption. DVFS scheduling reduces the supply voltage and frequency when conceivable for preserving energy consumption. Subsequently, a large number of procedures considering the issue of limiting the consumption of the needed energy without violating the timing constraints on uniprocessor systems are widely presented in literature for different task models. Many of the previous work that studied the problem of energy efficient frameworks for real-time embedded systems apply the DVFS technique to reduce the processor energy consumption [5–8].

In [6], authors proposed a DVFS scheme under EDF scheduling policy to decrease dynamic power consumption for real-time systems. In [8], we settle the hypothesis for energy consumption in real-time systems, we proposed an energy efficient real-time scheduling algorithm of aperiodic tasks for wireless sensors. Specifically, we applied the concept of DVFS technique to the process of real-time scheduling. Further, we proposed in [9] an energy guarantee real-time scheduling algorithm that applies the DVFS technique targeting energy harvesting systems. We show that our scheduler achieves capacity savings when compared to other schedulers.

On the other side, fault tolerance objectives are of uppermost importance for embedded systems [10]: system failures can occur in real-time computing systems and can result in hardware errors and/or deadline misses. It was found that the most common errors in computing systems are soft errors, and hence most researches target their work on soft errors to present fault tolerant systems. Such research efforts produced scheduling algorithms with the joint consideration of energy and timing constraints in fault tolerant systems.

More recently, Zhao et al. [11] presented the Generalized Shared Recovery (GSHR) technique to reserve computing resources, which can be used by other tasks to enhance the energy efficiency. Later, this work was extended to be applied to a real-time periodic task model [10]. The proposed algorithms aim to determine the processor scaling factor and the reserved resources for every task to enhance the minimization of energy while still guaranteeing the reliability requirement at the task-level. The advantage of the GSHR scheduler comes from the fact that the reliability of the system can be increased when applying the DVFS technique.

Recently, Han et al. developed effective scheduling algorithms that can save energy when considering that the proposed real-time system can tolerate up to k failures when scheduling a set of aperiodic tasks on a single processor under the EDF policy [12]. For this sake, authors proposed three algorithms: The first two algorithms are based on the previous work performed in [6]. The third algorithm extends the first two by considering that the computing resources are no longer reserved and hence better energy saving performance can be achieved. The main drawback of this work is that the problem of improving the system reliability in presence of failures cannot be solved by a simple modification to the work done in [6].

3 Model and Terminology

We consider the system model and their corresponding notations. Then, we present the problem formulation.

3.1 Task Model

We consider a set of n independent aperiodic real-time jobs $\mathcal{J} = \{J_1, J_2, \cdots, J_n\}$, where J_i denotes the i^{th} job in $\mathcal{J}$ and is characterized by a three tuple (a_i, c_i, d_i). The definition of these parameters are as follows:

- a_i is referred to the arrival time, this means that the time when job J_i is ready for execution.
- c_i is referred to the worst case execution time (WCET) under the maximum available speed S_{max} of the processor.
- d_i is considered as the absolute deadline of job J_i.

We denote the laxity of the job J_i by $d_i - (a_i - c_i)$. We consider that the job set $\mathcal{J}$ is said to be feasible in the real-time manner and under fault-free scenario. In other words, there exists a feasible schedule for $\mathcal{J}$ in abscence of energy considerations, where all deadlines in are respected.

3.2 Power and Energy Model

We assume the speed/frequency of the processor is equipped with a DVFS-enabled with N discrete frequencies f ranging from $f_{min} = f_1 \leq f_2 \leq \cdots \leq f_N = f_{max}$. We consider the notation processor speed S_N, or slowdown factor, as the ratio of the computed speed to the highest processor speed, this means that $S_N = f_N/f_{max}$. The CPU speed can be changed continuously in $[S_{min}, S_{max}]$. Consequently, when a job J_i is executed under speed S_i, the worst case execution time of J_i becomes equal to c_i/S_i.

For embedded systems, the processor and off-chip devices such as memory, I/O interfaces and underlying circuits mainly consume the major part of the energy [13]. In this paper, we distinguish between frequency-dependent and frequency-independent components of the consumed power. Specifically, we adopt the overall power consumption (P) at a slowdown factor S as follows:

$$P = P_{ind} + P_{dep} = P_{ind} + C_{ef}S^{\alpha} \tag{1}$$

Where P_{ind} stands for the frequency-independent power that includes the constant leakage power and the power consumed by off-chip devices [12], which is independent of the system frequency and supply voltage. C_{ef} is denoted as the effective switching capacitance. α is the dynamic power exponent, which is a constant usually larger than or equal to 2.

P_{dep} is considered to be the frequency-dependent active power, which includes not only the processor power, but also any power that depends on the processing speed S. Consequently, the energy consumption of a job J_i that runs at the speed S_i, denoted as $E_i(S_i)$, can be expressed as:

$$E_i(S_i) = (P_{ind} + C_{ef}S_i^{\alpha}).\frac{c_i}{S_i} \tag{2}$$

3.3 Energy Storage Model

The used system relies on an energy storage unit (battery or supercapacitor) with an ideal capacity, namely C, that corresponds to a maximum stored energy. The energy level of the battery must remain between two predefined boundaries, namely C_{min} and C_{max}, where $C = C_{max} - C_{min}$. We consider that $C(t)$ stands for the level of energy in the battery at time t. We state that the stored energy at any time is less than the ideal storage capacity, this means

$$C(t) \leq C \quad \forall\, t \tag{3}$$

3.4 Fault Model

During the execution of any operation on a computing system, both transient and permanent faults may affect the system due to various reasons, like hardware defects or system errors. In this paper, we consider only transient faults since it has been shown to be dominant over permanent faults especially with scaled technology sizes [14].

The proposed system can afford a maximum of k transient faults. The used system is usually able to detect faults when a job ends its execution. We assume that the energy and time overhead caused by fault detection, denoted as EO_i and TO_i respectively, are not negligible and are independent of the variations in the processor frequency.

Generally, there is not restriction on the occurrence of faults during the execution of jobs and multiple faults may occur when executing a single job [12]. The fault recovery scheme in this paper is based on re-executing the affected job. Consequently, R_i stands for the maximum recovery overhead for executing a job J_i under the maximum speed S_{max}, which is equal to c_i, or $R_i = c_i$. When a fault occurs during any job execution, say J_i, a recovery job of the same deadline d_i is released, which is subject to preemption as well.

4 Fault Tolerant Speed Schedule

4.1 Overview of the Scheduling Scheme

In this section, we present a fault-tolerant DVFS scheduling approach for a dynamic-priority real-time job set on uniprocessor systems to enhance energy saving while still guaranteeing the timing constraints. The proposed algorithm is based on the Energy Saving - Dynamic Voltage and Frequency Scaling (ES-DVFS) algorithm that we previously proposed in [8].

To better understand tthe proposed approach and before proceeding, we first state some basic definitions and then briefly reiterate the general concept of ES-DVFS.

Definition 1. *Given a real-time job set $\mathcal{J}$ of n independent aperiodic jobs such that $\mathcal{J} = \{J_1, J_2, \cdots, J_n\}$.*

- *$\mathcal{J}(t_s, t_f)$ denotes the job set contained in the time interval $\phi = [t_s, t_f]$, i.e jobs that are ready to be executed at time t_s and with deadlines smaller than or equal to t_f. $\mathcal{J}(\phi) = \{J_i \mid t_s \leq a_i < d_i \leq t_f\}$.*
- *$W(\phi)$ denotes the overall amount of the jobs' workload in $\mathcal{J}(\phi)$ in the time interval $[t_s, t_f]$, that means that the total worst case processing time of jobs completely embedded in the time interval,*

$$W(\phi) = \sum_{J_i \epsilon \mathcal{J}(\phi)} c_i \tag{4}$$

- *The processor load $h(\phi)$ over an interval $\phi = [t_s, t_f]$ is defined as*

$$h(\phi) = \frac{W(\phi)}{t_f - t_s} \tag{5}$$

- *The intensity of jobs in the time interval $\phi = [t_s, t_f]$, denoted as $I(\phi)$, is defined as*

$$I(\phi) = \max_{J_j \epsilon \mathcal{J}(\phi)} \left(\frac{\sum_{d_i \leq d_j} c_i}{d_j - (t_f - t_s)} \right) \tag{6}$$

– *We consider that the* fault-related overhead *of a time interval* $\phi = [t_s, t_f]$, *denoted as* $W_k(\phi)$ *is*

$$W_k(\phi) = W_r(\phi) + W_{TO}(\phi) \tag{7}$$

Where $W_r(\phi)$ *stands for the worst-case workload that is reserved to be used in case of recovery, i.e.* $W_r(\phi) = k \times (R_l + TO_l)$ *and* l *represents the index of the job with the maximum recovery time in* $\mathcal{J}(\phi)$. $J_l = \{J_i \mid max(R_i + TO_i),\ J_i \epsilon \mathcal{J}(t\phi)\}$ *and* $W_{TO}(\phi)$ *is considered as the overhead due to fault detection from regular jobs, i.e.*

$$W_{TO}(\phi) = \sum_{J_i \epsilon \mathcal{J}(\phi)} TO_i \tag{8}$$

Further, $W_k(\phi) \geq W_{k-1}(\phi)$ *for* $k \geq 1$, *since all recovery of jobs have positive execution times. For this sake, we restrict our work to a k-fault tolerant system that can exactly tolerate k faults when investigating the worst-case reserved recovery of fault scenarios.*

– *The energy demand of a job set* $\mathcal{J}$ *in the interval* $\phi = [t_s, t_f]$ *is*

$$g(\phi) = \sum_{t_s \leq r_k, d_k \leq t_f} E_k(S_k) \tag{9}$$

Given a real-time job set $\mathcal{J}$, ES-DVFS was provably optimal in minimizing energy consumption in on-line energy-constrained setting by providing sound dynamic speed reduction mechanisms [8]. The ES-DVFS approach can provide a feasible energy efficient technique, which is function of the processor frequency where the time constraints of all jobs in $\mathcal{J}$ are still respected. Under this assumption, the ready jobs in the used interval are not executed with fixed speed as the previous work in [6], but are dynamically adjusted on the fly.

4.2 Concepts for the EMES-DVFS Model

ES-DVFS is optimal in case of fault-free conditions. Hence, To make the above ES-DVFS fault-tolerant, we adopt a scheduler (we call it MES-DVFS) is to take into consideration the fault recovery when calculating the effective processor load and intensity in any interval $\phi = [t_s, t_f]$, i.e. to replace $h(\phi)$ and $I(\phi)$ with $h_m(\phi)$ and $I_m(\phi)$ respectively, such that

$$h_m(\phi) = \frac{\sum\limits_{J_i \epsilon \mathcal{J}(\phi)} c_i + k \times R_l}{d_{max} - W_{TO}(\phi) - k \times TO_l} \tag{10}$$

Where d_{max} is the longest deadline in $\mathcal{J}(\phi)$ and $W_{TO}(\phi)$ stands for the overall overheads due to fault-detection for original jobs as defined in Definition 1.

In addition, the intensity of the jobs in $\mathcal{J}(\phi)$ at current time t is

$$I_m(t) = \max_{J_j \epsilon \mathcal{J}(\phi)} \left(\frac{\sum\limits_{d_i \leq d_j} c_i + k \times R_l}{d_j - t - W_{TO}(\phi) - k \times TO_l)} \right) \tag{11}$$

When a fault is detected, and for the sake of reducing the total energy consumption for the regular jobs and their recovery copies, MES-DVFS runs the copy of the recovered job using a defined processor speed ($S_i \leq S_{max}$). However, this may not be energy efficient since, in practice, the fault rate is considered to be very low.

An extended approach for MES-DVFS (we call it EMES-DVFS), is to execute the recovery copies under the highest possible processor speed, usually at S_{max}.

Hence, the intensity calculation of the jobs in $\mathcal{J}(\phi)$ can be modified correspondingly, as Eq. 12

$$I_e(t) = \max_{J_j \epsilon \mathcal{J}(\phi)} \left(\frac{\sum\limits_{d_i \leq d_j} c_i}{d_j - t - W_k(\phi)} \right) \tag{12}$$

Further, the effective processor load of the jobs in $\mathcal{J}(\phi)$ can also be modified correspondingly, as Eq. 13

$$h_e(\phi) = \frac{\sum\limits_{J_i \epsilon \mathcal{J}(\phi)} c_i}{d_{max} - W_k(\phi)} \tag{13}$$

4.3 Description of the EMES-DVFS Scheduler

We consider a job set $\mathcal{J}$ of n jobs $\mathcal{J} = \{J_1, J_2, \cdots, J_n\}$ that can tolerate up to k faults. Let $\mathcal{Q}(\phi)$ be the list of ready but uncompleted jobs for execution in the time interval $\phi = [t_s, t_f]$. We can formulate our EMLPEDF algorithm to obey the following rules:

Rule 1: The EDF scheduler is used to select the future running jobs in $\mathcal{Q}(\phi)$.
Rule 2: The processor is imperatively idle in $[t_s, t_s + 1)$ if $\mathcal{Q}(\phi)$ is empty.
Rule 3: The processor is imperatively busy in $[t_s, t_s + 1)$ if $\mathcal{Q}(\phi)$ is not empty and $0 < C(t_s) \leq C$. Hence, the following steps must be performed:

1. Select the job, say J_i with the highest priority.
2. Calculate the effective processor load $h_e(\phi)$ and intensity $I_e(\phi)$ using Eqs. 13 and 12 respectively.
3. Set the speed S_{ei} of job J_i to the maximum between $h_e(\phi)$ and $I_e(\phi)$.

Rule 4: If $S_{ei} < S_{min}$, then $S_{ei} = S_{min}$ $\forall$ $J_i \epsilon \mathcal{J}(\phi)$.
Rule 5: If job, say J_j is released with $d_j < d_i$, then update S_{ei} by **Rule 3**.
Rule 6: If job, say J_k is released with $d_k > d_i$, then complete the execution of J_i.
Rule 7: If job, say J_k is released with $d_k > d_i$, and $c_k > d_k - d_i$ then update S_{ei} by **Rule 3**.
Rule 8: Calculate the energy consumption $E_i(S_{ei})$ according to Eq. (2).
Rule 9: Calculate the energy level in the battery when the job ends its execution.
Rule 10: Remove job J_i from the queue $\mathcal{Q}(\phi)$.
Rule 11: Repeat step (1)–(8) until the queue $\mathcal{Q}$ is empty.

5 Simulation Results

We compare the performance of four scheduling algorithms: EMES-DVFS, MES-DVFS, NPM and LPSSR proposed in [12]. NPM scheme executes jobs with maximum frequency and does not scale down the voltage/frequency. We developed a discrete event-driven simulator in C that generates a job set $\mathcal{J}$ where the number of jobs varies from 10 to 50. The simulation is repeated 100 times for the same number of jobs.

For the sake of clarity, we use NPM as a reference schedule that represents the schedule of given set of jobs $\mathcal{J}$ without incorporating DVFS. We consider that all the plotted energy consumptions are normalized to NPM. We consider that $\alpha = 2$, $C_{ef} = 1$, $P_{ind} = 0.05$, and S_{min} is equal to 0.25.

We compute the simulation results by using a discrete DVFS processor that operates on 8 frequency levels $\{1.00, 0.86, 0.76, 0.67, 0.57, 0.47, 0.38, 0.28\}$ as in the PentiumM processor.

5.1 Experiment 1: Energy Consumption by Varying the Number of Jobs

First, we take interest in how energy consumption of the processor changes when we vary the number of jobs. We report here the results of four simulation studies where the fault rate is set to 10^{-5} and the number of jobs varies from 10 to 50. Further, we consider that the number of faults is strictly equal to 1 ($k = 1$). In each of the four schedulers, we compute the normalized energy consumption metric of the used speed. Figure 1 shows the expected energy consumption of EMES-DVFS and MES-DVFS versus previous schedulers like NPM and LPSSR.

From Fig. 1, we find that the energy consumption of the four schedulers increases as the number of jobs becomes larger. This is reasonable since the likelihood of having large slack time that can be used for DVFS is diminishing as we increase the number of jobs. Further, a significant amount of energy saving is gained by EMES-DVFS and MES-DVFS schemes since it can benefit from the significant amount of slack time that decreases the expected consumption of energy. In other words, EMES-DVFS and MES-DVFS can effectively assign the speeds for every job such that the job set becomes feasible at a speed closest to the critical speed.

When the number of jobs is low, we find that the reduction in energy consumption that is achieved by the tested algorithms are approximately the same. This is because most jobs are executed at the lowest speed. With the increasing number of jobs, our scheduler demonstrates its great advantage in achieving a high energy saving. As an average, EMES-DVFS can achieve an additional 51% and 20% of energy saving when we compare it with NPM and LPSSR, respectively. Further, the difference in energy consumption is around 12% between EMES-DVFS can MES-DVFS, since in the EMES-DVFS scenario, the recovery from one faulty job is performed at maximum processor speed and subsequent slack is left for DVFS.

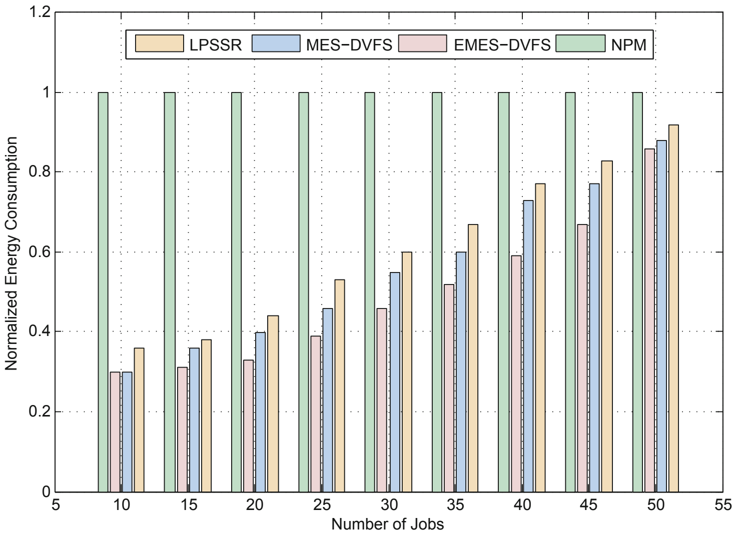

Fig. 1. Energy savings by varying the numbers of jobs, $k = 1$

We conclude that our approach gains more energy savings in such a way that it can benefit from the slacks generated during execution and hence it can use all the available slack time.

6 Conclusions

In this work, we proposed and evaluated a new novel approach, which aims to enhance energy savings when scheduling a real-time job set that can tolerate up to k transient faults while still respecting time and energy constraints. We benefit from the slacks generated during run-time to the maximum extent in such a way that all the available slack time is used for energy reduction, which is carried out using dynamic voltage and frequency scaling (DVFS). Under this notion, we propose an algorithm that estimates an optimal speed reduction mechanism which maintains feasibility within predefined timing constraints when no more than k faults occur.

Our scheduler dynamically adjusts the jobs' slowdown factors by using the run-time slacks which may be increased for recovery demands of the system. It differs from the previous approaches where the assignments of job frequencies are predetermined, and hence it is more flexible and adaptive in minimizing energy consumption while still keeping the systems reliability at a desired level. Simulation results proved that the presented scheduler can significantly reduce energy consumption when compared with the existing works.

References

1. Shin, Y., Choi, K., Sakurai, T.: Power optimization of real-time embedded systems on variable speed processors. In: Proceedings of the International Conference on Computer-Aided Design, pp. 365–368 (2000). https://doi.org/10.1109/ICCAD.2000.896499
2. Quan, G., Hu, X.: Energy efficient fixed-priority scheduling for real-time systems on variable voltage processors. In: Proceedings of the Design Automation Conference, pp. 828–833 (2001). https://doi.org/10.1109/DAC.2001.156251
3. Srinivasan, J., Adve, S.V., Bose, P., Rivers, J., Hu, C.K.: Ramp: a model for reliability aware microprocessor design. IBM Research Report, RC23048 (2003)
4. Castillo, X., McConnel, S.R., Siewiorek, D.P.: Derivation and calibration of a transient error reliability model. IEEE Trans. Comput. **31**, 658–671 (1982). https://doi.org/10.1109/TC.1982.1676063
5. Aydin, H., Melhem, R., Mosse, D., Mejia-Alvarez, P.: Power-aware scheduling for periodic real-time tasks. IEEE Trans. Comput. **53**(5), 584–600 (2004)
6. Yao, F., Demers, A., Shenker, S.: A scheduling model for reduced CPU energy. In: Proceedings of the 36th Annual Symposium on Foundations of Computer Science, pp. 374–382, October 1995
7. Zhang, Y., Chakrabarty, K., Swaminathan, V.: Energy-aware fault tolerance in fixed-priority real-time embedded systems. In: Proceedings of the 2003 IEEE/ACM International Conference on Computer-Aided Design, ICCAD 2003 (2003)
8. El Ghor, H., Aggoune, E.M.: Energy efficient scheduler of aperiodic jobs for real-time embedded systems. Int. J. Autom. Comput. 1–11 (2016)
9. EL Ghor, H., Chetto, M.: Energy guarantee scheme for real-time systems with energy harvesting constraints. Int. J. Autom. Comput. (to appear)
10. Zhao, B., Aydin, H., Zhu, D.: Energy management under general task-level reliability constraints. In: IEEE 18th Real Time and Embedded Technology and Applications Symposium (2012)
11. Zhao, B., Aydin, H., Zhu, D.: Generalized reliability-oriented energy management for real-time embedded applications. In: 48th ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 381–386, June 2011
12. Han, Q., Niu, L., Quan, G., Ren, S., Ren, S.: Energy efficient fault-tolerant earliest deadline first scheduling for hard real-time systems. Real-Time Syst. **50**, 592–619 (2014)
13. Burd, T.D., Brodersen, R.W.: Energy efficient CMOS microprocessor design. In: Proceedings of the HICSS Conference, January 1995
14. Hazucha, P., Svensson, C.: Impact of CMOS technology scaling on the atmospheric neutron soft error rate. IEEE Trans. Nuclear Sci. **47**(6), 2586–2594 (2000). https://doi.org/10.1109/23.903813