Low-Energy Standby-Sparing for Hard Real-Time Systems

Alireza Ejlali, Bashir M. Al-Hashimi, *Fellow, IEEE,* and Petru Eles, *Member, IEEE*

***Abstract*— Time-redundancy techniques are commonly used in real-time systems to achieve fault tolerance without incurring high energy overhead. However, reliability requirements of hard real-time systems that are used in safety-critical applications are so stringent that time-redundancy techniques are sometimes unable to achieve them. Standby sparing as a hardware-redundancy technique can be used to meet high reliability requirements of safety-critical applications. However, conventional standby-sparing techniques are not suitable for low-energy hard real-time systems as they either impose considerable energy overheads or are not proper for hard timing constraints. In this paper we provide a technique to use standby sparing for hard real-time systems with limited energy budgets. The principal contribution of this work is an online energy-management technique which is specifically developed for standby-sparing systems that are used in hard real-time applications. This technique operates at runtime and exploits dynamic slacks to reduce the energy consumption while guaranteeing hard deadlines. We compared the low-energy standby-sparing (LESS) system with a low-energy time-redundancy system (from a previous work). The results show that for relaxed time constraints, the LESS system is more reliable and provides about 26% energy saving as compared to the time-redundancy system. For tight deadlines when the time-redundancy system is not sufficiently reliable (for safety-critical application), the LESS system preserves its reliability but with about 49% more energy consumption.**

***Index Terms*— Fault tolerance, Low-power design, Real-time and embedded systems**

I. Introduction

Time-redundancy techniques (e.g., rollback-recovery) [1-5] and hardware-redundancy techniques (e.g., triple modular redundancy) [7][16-18] are commonly used in real-time systems to achieve fault tolerance. One advantage of time redundancy is that it usually requires less hardware as compared to hardware redundancy [7][18], and hence from an energy consumption point of view, time redundancy is generally more preferable than hardware redundancy. However, time-redundancy techniques may not be able to achieve the required reliability of hard real-time systems that are used in safety-critical applications [17]. To achieve the high reliability requirements of these systems, the use of hardware redundancy (also called hardware fault-tolerance [7] or replication [16]) is a must [16]. Indeed, in time-redundancy techniques, system reliability largely depends on the available amount of slack time so that when deadlines are tight, the reliability decreases significantly [6][7][17]. In contrast, the use of hardware redundancy decouples the fault tolerance from the slack time and can provide high reliability even when deadlines are tight. However, hardware redundancy usually incurs considerable energy overhead [7]. For example, triple modular redundancy (TMR) and duplication are two well-known hardware-redundancy techniques that clearly increase the energy consumption by 200% and 100% respectively [18]. Therefore, in systems with limited energy budgets, if we want to benefit from the high reliability offered by hardware redundancy, energy consumption should be carefully taken into account. In this paper we describe how a low-energy hardware-redundancy technique, based on standby-sparing [18], can be used for hard real-time systems. We will argue in this paper (Section II) that conventional standby-sparing techniques, i.e., hot and cold spares [18], cannot be used for low-energy hard-real time systems, and hence we have developed a different standby-sparing technique for this purpose. In a system which uses our proposed technique (throughout this paper we call this system LESS, which stands for Low-Energy Standby-Sparing), dynamic voltage scaling (DVS) is used to reduce the energy consumption of the primary unit. However, we do not use DVS for the spare and use dynamic power management (DPM) [10] instead. Indeed we will argue (Sections II, V, and VI) that in the LESS system, from both energy consumption and reliability viewpoints, DVS is not suitable for the spare. In this paper we provide an online energy-management technique which is specifically developed for the LESS system and uses slack reclamation (i.e., exploits dynamic slacks [2]) to reduce the energy consumption.

Some research works, e.g., [2][3][8][29], have addressed

Manuscript received on 5th May, 2011 and revised on 8th August, 2011. This work was supported by Engineering and Physical Sciences Research Council (EPSRC), UK, under grant no. EP/E035965/1. It is also partially supported by the Research Vice-Presidency of Sharif University of Technology.

A. Ejlali is with the Department of Computer Engineering, Sharif University of Technology, Tehran 14588-89694, Iran. E-mail: ejlali@sharif.edu.

B. M. Al-Hashimi is with the School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U.K. E-mail: bmah@ecs.soton.ac.uk.

P. Eles is with the Department of Computer and Information Science, Linköping University, S-58333 Linköping, Sweden. E-mail: petel@ida.liu.se.

both fault tolerance and low energy-consumption in fault-tolerant real-time systems that are based on time-redundancy. However, these works have focused on time redundancy and have not considered hardware redundancy. [15] has proposed a technique to exploit voltage scaling to reduce the energy overhead of TMR when it is used for real-time systems. This technique exploits system static slacks to reduce the energy overhead of a TMR system to a level comparable to that of a duplicated system and has not considered the use of dynamic slacks (slack reclamation). To achieve energy-aware fault-tolerance for hard real-time systems, [30] has proposed a static scheduling technique for executing independent tasks on a multi-processor system. This scheduling technique attempts to reduce the need for executing secondary (backup) tasks to conserve energy. However this work has not considered DVS and also has not considered any slack reclamation scheme. Furthermore, unlike our proposed technique, the technique of [30] cannot be used for dependent tasks. [6] has proposed to use a combination of information redundancy and time redundancy to address the resource conflict between time-redundancy and DVS on slack. However, this work has not considered hardware redundancy and does not provide any energy-management technique for fault-tolerant real-time systems.

This paper substantially extends our previous work in [28], where we proposed the basic idea of LESS, as follows:

1) We have considered the following five issues in the proposed energy management technique: *i*) Voltage transition overhead (both time and energy overheads) imposed by DVS, *ii*) Activation overhead (both time and energy overheads) imposed by DPM, *iii*) Static energy consumption, *iv*) The time overhead of the energy manager task itself, and *v*) The best-case execution times of the tasks. To consider these five issues, we revised the analytical energy models and the energy management algorithm.
2) By conducting new simulation experiments, we have compared the LESS system (with the revised energy management algorithm) with a recent low-energy time-redundancy system (presented in [29]) with respect to the energy consumption and reliability.
3) We have developed an analytical reliability model for the LESS system. Using this model, we have evaluated the reliability of the LESS system and compared it with the time-redundancy system.

The results show that for relaxed time constraints, both the LESS and time-redundancy systems can achieve the reliability requirements of safety-critical applications (indeed the LESS system is slightly more reliable in this case), but the LESS system provides about 26% energy saving as compared to the time-redundancy system. For tight deadlines, it can be observed that while the time-redundancy system cannot achieve the reliability requirement of safety-critical applications, the LESS system preserves its high reliability but with about 49% more energy consumption.

The rest of the paper is organized as follows. In Section II we describe how the LESS system operates. In Section III, we have developed analytical energy models for the LESS system. In Section IV, we have used the analytical models of Section III to provide an online energy management technique for the LESS system. In Section V, we have developed analytical reliability models for the LESS system. In Section VI simulation results are presented and discussed. Finally, Section VII concludes the paper.

II. Proposed Low-Energy Standby-Sparing Technique

In this paper we consider a standby-sparing system (called the low-energy standby-sparing or the LESS system) which is composed of two identical processors. One of them is called the primary unit and operates as the main processor, while the other one is called the spare unit and replaces the primary unit when it becomes faulty. There are two conventional types of standby sparing: hot and cold [18], but none of these two techniques is suitable for low-energy hard real-time systems. A hot spare unit operates in parallel with the primary unit and therefore imposes considerable energy overhead as the spare is always operational. In contrast to a hot spare, a cold spare is idle until a fault is detected in the primary unit. Therefore a cold spare does not consume power until needed to replace the primary unit. However, we have shown in [28] that in a hard real-time system, sometimes the spare must be activated even before the primary unit becomes faulty; otherwise a fault in the primary unit may result in missing a deadline. Therefore, the cold standby-sparing technique is not proper for hard real-time systems. In the LESS system, the spare is neither a cold spare nor a hot one. Rather, dynamic power management (DPM) [10] is used to reduce the energy consumption of the spare, i.e., it is kept idle for as long as possible, taking into account the hard real-time constraints. To reduce the energy consumption of the primary unit, dynamic voltage scaling (DVS) is used. However, we do not use DVS for the spare because, as we will see later in this paper, unlike the primary unit, the spare usually does not need to execute the tasks completely and can be kept idle (i.e. DPM), which makes it unnecessary for the spare to use DVS. Furthermore, we have shown in Section V that the reliability of the spare is a lower bound on the reliability of the whole LESS system and this lower bound would be degraded if we used DVS for the spare. For both the DVS and DPM techniques, we use a slack reclamation scheme, i.e., dynamically created slacks are exploited to achieve energy saving. Dynamic slacks result at runtime when tasks consume less than their worst-case execution time [2]. The use of dynamic slacks helps to achieve more energy saving as compared to the techniques that only use static slack time which is the difference between the deadline and the worst-case execution time [2][19]. It should be noted that the LESS system does not use its slack time for fault tolerance. The fault tolerance is achieved by using the spare and the slack time is only used for reducing the energy consumption. Therefore, unlike time-redundancy techniques, the LESS system preserves its fault-tolerance even when the available slack is small.

Clearly, a standby-sparing system requires an error

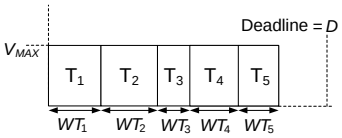

Fig. 1. A simple static schedule for a single-processor system with five dependent tasks and a common deadline (frame size) D

detection mechanism to determine if a task finishes successfully or not. In the context of fault-tolerant real-time systems, an error detection mechanism is usually assumed to be part of the software architecture and the error detection overhead is considered as part of the task execution time [1][5]. Similarly, in this paper we assume that an error detection mechanism is part of the software architecture. We focus only on transient faults as it has been observed that transient faults are the major source of concern in safety-critical applications [27][29][31]. In this paper, we consider frame-based real-time applications [29][31] with hard timing constraints where a set of dependent tasks, e.g., $\Gamma=\{T_1, T_2, T_3, \ldots, T_n\}$, is executed within each frame and must be completed before the end of the frame (deadline) [31]. The LESS system does not need any dedicated scheduler. Indeed it is assumed that a static schedule, like the one shown in Fig. 1, already exists for a single processor system which has no fault-tolerance or energy-management mechanism and the LESS system uses this same schedule to run the given application. When the LESS system is executing such a schedule, it does not change the temporal order of the tasks to avoid violating task dependencies. It should be noted that as we consider sequences of dependent tasks, different tasks (i.e., tasks T_i and T_j, where $i \neq j$) cannot be executed in parallel in the LESS system. Therefore we do not use dual-processor scheduling methods that schedule the tasks among the processors to execute different tasks in parallel. In the following, we describe how the LESS system operates.

Primary Unit: Suppose that a schedule like the one in Fig. 1 exists for a single processor system operating at the maximum supply voltage V_{MAX}. Fig. 2 shows how such a schedule is executed on the LESS system. When tasks are executed at the supply voltage V_{MAX}, each task T_i has a worst-case execution time WT_i, and an actual execution time AT_i. Each task T_i is executed on the primary unit at a supply voltage V_i, which may be less than V_{MAX}. For each task T_i, we define the normalized supply voltage ρ_i as follows:

$$\rho_i = \frac{V_i}{V_{MAX}} \tag{1}$$

When a reduced supply voltage is used for a task T_i, the worst-case execution time is prolonged from WT_i to WT_i/ρ_i and the actual execution time is prolonged from AT_i to AT_i/ρ_i [11]. Before starting each task T_i, we execute a relatively short task, called energy manager, on the primary unit (hachured tasks on the primary unit in Fig. 2) that determines the proper supply voltage V_i for the task T_i. After the energy manager determines the supply voltage V_i, the supply voltage must change from V_{i-1} to V_i. This voltage transition takes the time [25]:

$$TR_i = K \cdot |V_i - V_{i-1}| = K \cdot V_{MAX} \cdot |\rho_i - \rho_{i-1}| \tag{2}$$

where K is a constant factor.

To avoid additional voltage transitions, we never change the supply voltage for executing the energy manager task itself and it is executed at the same voltage as that of the previous task, i.e., V_{i-1}. Let τ_{EM} be the time it takes to execute the energy manager task at the supply voltage V_{MAX}, then the time overhead due to executing the energy manager task and changing the supply voltage is:

$$\tau_i = \frac{\tau_{EM}}{\rho_{i-1}} + TR_i = \frac{\tau_{EM}}{\rho_{i-1}} + K \cdot V_{MAX} \cdot |\rho_i - \rho_{i-1}| \tag{3}$$

Throughout this paper, we call τ_i the energy management overhead for the task T_i.

As reducing the supply voltage increases the execution time, the supply voltage of a task T_i cannot be reduced arbitrarily as it may result in missing the deadline. To guarantee that the supply voltage never decreases to a risky level with the possibility of deadline miss, we use the following rule:

Rule 1: The normalized supply voltage ρ_i for each task T_i should be determined so that we can always meet the deadline if we execute the subsequent tasks T_{i+1} through T_n at the maximum supply voltage.

Using this rule, we can provide a minimum value for ρ_i which guarantees that the deadline will not be missed. Let ST_i be the time at which the task T_{i-1} finishes running on the primary unit and the energy manager task starts running on this unit to determine the supply voltage for the next task T_i (Fig. 2). We have (Fig. 2):

$$ST_i = \sum_{j=1}^{i-1} (\tau_j + \frac{AT_j}{\rho_j}) \tag{4}$$

Let τ_M be the maximum possible value of τ_i. When the task T_i finishes, if we want to execute the subsequent tasks T_{i+1} through T_n at the maximum supply voltage V_{MAX}, the task T_{i+1} may take the worst-case time τ_M+WT_{i+1}. However, each of the tasks T_{i+2} through T_n take the worst-case time $\tau_{EM}+WT_j$ ($i+2 \leq j \leq n$). This is because when all the tasks T_{i+1} through T_n are executed at V_{MAX}, no voltage transition occurs for the tasks T_{i+2} through T_n and the energy manager task takes the time τ_{EM} when executed at V_{MAX}. Therefore, the worst-case time it takes to execute the subsequent tasks T_{i+1} through T_n at V_{MAX} is:

$$WCST_i = (\tau_M + WT_{i+1}) + \sum_{j=i+2}^{n} (\tau_{EM} + WT_j) \tag{5}$$

From (4), and (5), we can conclude that the maximum possible time which is available for executing the task T_i (and its associated energy management overhead τ_i) without violating Rule 1 is:

$$\begin{aligned} MT_i &= D - ST_i - WCST_i = \\ & D - \sum_{j=1}^{i-1} (\tau_j + \frac{AT_j}{\rho_j}) - (\tau_M + WT_{i+1}) - \sum_{j=i+2}^{n} (\tau_{EM} + WT_j) \end{aligned} \tag{6}$$

If we execute the task T_i at the maximum supply voltage, the worst-case time it takes will be τ_M+WT_i and since the time which is available to execute the task T_i without violating

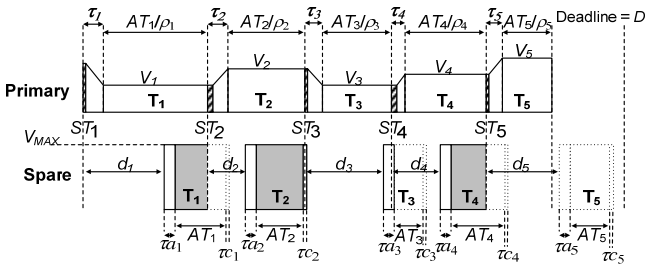

Fig. 2. The schedule of Fig. 1 running on the low-energy standby-sparing (LESS) system, which uses the proposed technique

Rule 1 is MT_i, the slack time which is available to the task T_i and can be exploited by DVS to reduce the task energy consumption is:

$$SL_i = MT_i - (\tau_M + WT_i) \tag{7}$$

When we exploit DVS, the worst-case time τ_M+WT_i increases to τ_M+WT_i/ρ_i, but this time increase should not be greater than the available slack, i.e.:

$$(\frac{WT_i}{\rho_i} - WT_i) \le SL_i \tag{8}$$

Inequality (8) can also be rewritten as:

$$\frac{WT_i}{WT_i + SL_i} \le \rho_i \tag{9}$$

which gives the minimum value of ρ_i. It should be noted that for each task T_i, the slack time SL_i is the entire slack which is available to the task, i.e., it includes both the dynamic and static slacks. To gain more insight into how the slack time SL_i includes both types of slack, we use (6) to rewrite (7) as follows:

$$SL_i = D - \sum_{j=1}^{i-1}(\tau_j + \frac{AT_j}{\rho_j}) - (\tau_M + WT_i) - (\tau_M + WT_{i+1}) - \sum_{j=i+2}^{n}(\tau_{EM} + WT_j) \tag{10}$$

Equation (10) implies that if the previous tasks (i.e., the tasks T_1 through T_{i-1}) consume less execution time (i.e., lower AT_j values, for $1 \le j \le i-1$), more slack SL_i will be available to the task T_i. The slack obtained in this way is by definition dynamic [2][19] because the task T_i can use the slack time which is obtained at runtime from its previous tasks. Equation (10) also implies that if the worst-case execution times of the tasks were smaller, more slack SL_i would be available. This slack is by definition static [2][19] as it depends on the worst-case execution times of the tasks. Although in this paper we exploit both dynamic and static slacks (i.e., the slack SL_i) to conserve energy, we do not need to provide separate analytical models for dynamic and static slacks, because in our proposed technique we do not intend to treat these two types of slack differently.

Spare Unit: In the spare unit the backup copy of each task T_i is executed at the maximum supply voltage, but with a delay d_i. During the delay time d_i the spare unit is in idle mode to conserve energy. As shown in Fig. 2, after the delay d_i, the activation of the spare unit (for executing the backup task T_i) takes the activation time τa_i. This activation time includes the communication time c_i which is required to communicate with the spare in order to activate it, plus the wake-up delay d_w which is the time it takes to change the spare mode from idle to operational, i.e., $\tau a_i = c_i + d_w$.

Whenever a task T_i which is being executed on the primary unit finishes successfully, the backup copy of this task, which is being executed on the spare unit, is dropped as it is no longer required. In this case, regardless of the supply voltage (energy consumption) of the primary unit, the later we start the backup copy, the greater fraction of the backup copy is dropped, which results in more energy saving. Therefore in the LESS system we start backup copies as late as possible, i.e., we always set d_i to the maximum possible delay. To calculate the maximum possible delay d_i, it should be noted that the delay cannot be increased arbitrarily as it may result in missing the deadline if a fault occurs in the primary unit. To guarantee that the delay never increases to a risky level with the possibility of deadline miss, we use the following rule:

Rule 2: The delay d_i for each task T_i should be determined so that if a fault occurs in the primary unit during the execution of the original task T_i, we can always meet the deadline if we continue executing (and do not drop) the backup task T_i on the spare and execute the subsequent tasks T_{i+1} through T_n at the maximum supply voltage on the primary unit.

Using this rule, we can find the maximum delay d_i which guarantees that the deadline will not be missed. Suppose that during the execution of the task T_i on the primary unit, a fault occurs. In this case, the backup copy of the task T_i which is being executed on the spare will not be dropped and its execution will be continued. Once the backup copy T_i finishes, it communicates back to the primary unit so that the primary unit can start executing the next task T_{i+1}. As shown in Fig. 2, this communication takes the time τc_i. Since the backup task T_i starts at the time ST_i+d_i (Fig. 2), the time duration $D-(ST_i+d_i)$ should be enough to finish not only the backup task T_i (with the worst-case time $\tau a_i+WT_i+\tau c_i$), but also all the subsequent tasks T_{i+1} through T_n which are executed on the primary unit at the maximum supply voltage (with the worst-case execution time $WCST_i$ given by (5)). Therefore, we have:

$$(\tau a_i + WT_i + \tau c_i) + WCST_i \le D - (ST_i + d_i) \tag{11}$$

From (11), we conclude that the maximum delay d_i is:

$$d_i = D - ST_i - WCST_i - (\tau a_i + WT_i + \tau c_i) \tag{12}$$

The execution of each backup task T_i on the spare should be delayed by the time d_i (given by Eq. 4) to achieve energy saving for the spare without missing the deadline. Although the slack time SL_i (given by (10)) has been defined for the original tasks on the primary unit, it is noteworthy that the slack time SL_i is also quite important for the backup tasks on the spare. This is because from (10) and (12) we can conclude that:

$$d_i = SL_i + \tau_M - (\tau a_i + \tau c_i) \tag{13}$$

This implies that when more slack time is available to a task T_i, d_i becomes greater, which results in more energy saving for the backup task on the spare.

The energy manager task is not only responsible for determining the proper value of ρ_i, but is also responsible for

determining the value of d_i using (12) (Section IV). The problem which is considered in the rest of this paper is how, for each task T_i, the parameters ρ_i should be determined by the energy manager task so that the energy consumption becomes minimized while guaranteeing that the deadline will not be missed.

III. Energy Consumption Model

In this section we develop analytical models for the energy consumption of the LESS system. It is argued in Appendix C that to analyze the average energy consumption of a fault-tolerant system, we do not need to consider the cases where the system tolerates a fault. Therefore, in this section, we only consider the fault-free scenario where no fault occurs in the LESS system.

Primary Unit: In digital circuits, power consumption has two main components: dynamic switching power and static leakage power [26]. The dynamic power consumption of each task T_i on the primary unit is [6][14]:

$$P_{PR-Dyn}(T_i)=C_{eff}V_i^2 f_i \tag{14}$$

where C_{eff} is the average switched capacitance for the primary unit, and V_i and f_i are respectively the supply voltage and the operational frequency during the execution of the task T_i. The static power consumption of each task T_i on the primary unit can be written as [26]:

$$P_{PR-Stat}(T_i)=I_{Sub}V_i=I_0 e^{\frac{V_{th}}{\eta V_T}}\cdot V_i \tag{15}$$

where I_{Sub} is the leakage current, I_0 depends on technology parameters and device geometries, η is a technology parameter, V_{th} is the threshold voltage of transistors, and V_T is the thermal voltage. As mentioned in Section II, for each task T_i on the primary unit, first the supply voltage changes from V_{i-1} to V_i and then the supply voltage V_i is used to execute the task T_i along with the energy manager task that determines the supply voltage V_{i+1}. Therefore, the energy consumption of each task T_i on the primary unit, including the voltage transition overhead and the energy consumption of the energy manager task, is:

$$E_{PR}(T_i) = ET_i + (I_{Sub}V_i + C_{eff}V_i^2 f_i)\cdot\left(\frac{AT_i+\tau_{EM}}{\rho_i}\right) \tag{16}$$

where ET_i is the energy consumed due to the voltage transition from V_{i-1} to V_i, and $(AT_i+\tau_{EM})/\rho_i$ is the time it takes to execute the task T_i along with the subsequent energy manager task. The energy overhead ET_i is given by [25]:

$$ET_i = C_r\cdot(V_i-V_{i-1})^2 = C_r V_{MAX}^2(\rho_i-\rho_{i-1})^2 \tag{17}$$

where C_r denotes power rail capacitance. For the DVS technique, it can be assumed that there is an almost linear relationship between V_i and f_i [11], therefore using (1) we can write $\rho_i = V_i/V_{MAX} = f_i/f_{MAX}$, where f_{MAX} is the operation frequency associated to the supply voltage V_{MAX}. Hence, the energy $E_{PR}(T_i)$ of (16) can also be written as:

$$E_{PR}(T_i) = ET_i + (I_{Sub}V_{MAX} + C_{eff}V_{MAX}^2 f_{MAX}\rho_i^2)\cdot(AT_i+\tau_{EM}) \tag{18}$$

Let P_S be $I_{Sub}V_{MAX}$ and P_D be $C_{eff}V_{MAX}^2 f_{MAX}$ then we can rewrite (18) as:

$$E_{PR}(T_i) = ET_i + (P_S + P_D\rho_i^2)\cdot(AT_i+\tau_{EM}) \tag{19}$$

It should be noted that, based on (14) and (15), P_S and P_D are respectively the static and dynamic power consumption of the primary unit operating at the maximum supply voltage.

Spare Unit: To calculate the energy consumption of the backup task T_i on the spare, we consider four possible cases:

Case 1: The original copy of T_i finishes before activating the spare unit to execute the backup copy of T_i.

In this case, since "The finish time of the original copy of T_i" ≤ "The time at which the spare starts becoming activated to execute the backup copy of T_i", we have (Fig. 2):

$$ST_i+\tau_i+\frac{AT_i}{\rho_i} \le ST_i+d_i \quad \equiv \quad \tau_i+\frac{AT_i}{\rho_i}\le d_i \tag{20}$$

In this case, if the original copy finishes successfully, the spare unit will not be activated as the backup copy is not required. Such a scenario has occurred for the task T_5 in Fig. 2. For this case, the energy consumption of the spare is:

$$E_{SPR}(T_i)=0 \quad when \quad \tau_i+\frac{AT_i}{\rho_i}\le d_i \tag{21}$$

Case 2: The original copy of T_i finishes after the spare unit starts becoming activated to execute the backup copy, however the original copy finishes before the spare unit starts executing the backup copy.

In this case, since "The time at which the spare starts becoming activated to execute the backup copy of T_i" < "The finish time of the original copy of T_i", we have (Fig. 2):

$$ST_i+d_i < ST_i+\tau_i+\frac{AT_i}{\rho_i} \quad \equiv \quad d_i < \tau_i+\frac{AT_i}{\rho_i} \tag{22}$$

Also, since "The finish time of the original copy of T_i" ≤ "The start time of the backup copy of T_i", we have (Fig. 2):

$$ST_i+\tau_i+\frac{AT_i}{\rho_i}\le ST_i+d_i+\pi a_i \quad \equiv \quad \tau_i+\frac{AT_i}{\rho_i}-\pi a_i\le d_i \tag{23}$$

Inequalities (22) and (23) can be written together as:

$$\tau_i+\frac{AT_i}{\rho_i}-\pi a_i\le d_i<\tau_i+\frac{AT_i}{\rho_i} \tag{24}$$

In this case, the spare unit becomes activated which requires the activation energy Ea_i. However, before the backup copy starts, the original copy has finished successfully and hence the backup copy is not required. Such a scenario has occurred for the task T_3 in Fig. 2. For this case, the energy consumption of the spare is:

$$E_{SPR}(T_i)=Ea_i \quad when \quad \tau_i+\frac{AT_i}{\rho_i}-\pi a_i\le d_i<\tau_i+\frac{AT_i}{\rho_i} \tag{25}$$

The activation energy Ea_i includes both the communication energy Ecm_i which is required to communicate with the spare in order to activate it, and the wake-up energy E_w which is required to change the spare mode from idle to operational, i.e., $Ea_i = Ecm_i+E_w$.

Case 3: The original copy of T_i finishes after the backup copy of T_i starts, however the original copy finishes before the backup copy finishes and hence the rest of the backup copy is dropped.

In this case, since "The start time of the backup copy of T_i" < "The finish time of the original copy of T_i", we have (Fig. 2):

$$ST_i + d_i + \pi a_i < ST_i + \tau_i + \frac{AT_i}{\rho_i} \quad \equiv \quad d_i < \tau_i + \frac{AT_i}{\rho_i} - \pi a_i \tag{26}$$

Also, since "The finish time of the original copy of T_i" < "The finish time of the backup copy of T_i", we have (Fig. 2):

$$ST_i + \tau_i + \frac{AT_i}{\rho_i} < ST_i + d_i + \pi a_i + AT_i \quad \equiv \quad \tau_i + \frac{AT_i}{\rho_i} - AT_i - \pi a_i < d_i \tag{27}$$

Inequalities (26) and (27) can be written together as:

$$\tau_i + \frac{AT_i}{\rho_i} - AT_i - \pi a_i < d_i < \tau_i + \frac{AT_i}{\rho_i} - \pi a_i \tag{28}$$

In this case, unlike Cases 1 and 2, as the backup copy starts before the original copy finishes, a part from the beginning of the backup copy is executed (the shaded areas in Fig. 2). However, as the original copy finishes before the backup copy finishes, the backup copy is not executed completely and is dropped once the original copy finishes so that the backup copy is executed only for a duration $(\tau_i+AT_i/\rho_i)-(d_i+\pi a_i)$. Such a scenario has occurred for the tasks T_1 and T_4 in Fig. 2. For this case, the energy consumption of the spare is:

$$E_{SPR}(T_i) = Ea_i + P\cdot(\tau_i + \frac{AT_i}{\rho_i} - d_i - \pi a_i) \quad when \quad \tau_i + \frac{AT_i}{\rho_i} - AT_i - \pi a_i < d_i < \tau_i + \frac{AT_i}{\rho_i} - \pi a_i \tag{29}$$

where P is the power consumption of the spare unit when it is active. It should be noted that as the spare does not exploit DVS and always operates at the maximum supply voltage, we have $P=P_S+P_D=I_{Sub}V_{MAX}+C_{eff}V_{MAX}^2f_{MAX}$.

Case 4: The original copy does not finish before the backup copy finishes.

In this case, since "The finish time of the backup copy of T_i" ≤ "The finish time of the original copy of T_i", we have (Fig. 2):

$$ST_i + d_i + \pi a_i + AT_i \le ST_i + \tau_i + \frac{AT_i}{\rho_i} \quad \equiv \quad d_i \le \tau_i + \frac{AT_i}{\rho_i} - AT_i - \pi a_i \tag{30}$$

We have also proved in Theorem 1 in Appendix A that d_i is not less than $\tau_i+AT_i/\rho_i-AT_i-(\pi a_i+\pi c_i)$, i.e., we have:

$$\tau_i + \frac{AT_i}{\rho_i} - AT_i - (\pi a_i + \pi c_i) \le d_i \tag{31}$$

Inequalities (30) and (31) can be written together as:

$$\tau_i + \frac{AT_i}{\rho_i} - AT_i - (\pi a_i + \pi c_i) \le d_i < \tau_i + \frac{AT_i}{\rho_i} - AT_i - \pi a_i \tag{32}$$

In this case, since the original copy does not finish before the backup copy finishes, the backup copy is not dropped and is executed completely. Also, once the backup copy T_i finishes, it communicates back to the primary unit which requires the communication energy Ec_i. Such a scenario has occurred for the task T_2 in Fig. 2. For this case, the energy consumption of the spare is:

$$E_{SPR}(T_i) = Ea_i + P\cdot AT_i + Ec_i \quad when \quad \tau_i + \frac{AT_i}{\rho_i} - AT_i - (\pi a_i + \pi c_i) \le d_i < \tau_i + \frac{AT_i}{\rho_i} - AT_i - \pi a_i \tag{33}$$

It can be seen from the above discussion that in Cases 1, 2, 3, and 4 we have considered the values of d_i that are greater than or equal to $\tau_i+AT_i/\rho_i-AT_i-(\pi a_i+\pi c_i)$. We have proved in Theorem 1 in Appendix A that d_i is not less than $\tau_i+AT_i/\rho_i-AT_i-(\pi a_i+\pi c_i)$, and hence we do not need to consider the case where $d_i<\tau_i+AT_i/\rho_i-AT_i-(\pi a_i+\pi c_i)$. This means that all the possible values of the parameter d_i have been considered in the above four cases and hence these cases are collectively exhaustive.

It is noteworthy that in the LESS system we never have the opportunity to drop original tasks that are executed on the primary unit. This is because, as it can be seen from Fig. 2, the backup copy of a task T_i finishes at the time $ST_i+d_i+\pi a_i+AT_i$, and then starts communicating back to the primary unit, which finishes at the time $ST_i+d_i+\pi a_i+AT_i+\pi c_i$. After the time $ST_i+d_i+\pi a_i+AT_i+\pi c_i$, if the original copy was still running, it would be no longer required and could be dropped. Since the finish time of the original copy is $ST_i+\tau_i+AT_i/\rho_i$, this case could only occur if we had:

$$ST_i + d_i + \pi a_i + AT_i + \pi c_i < ST_i + \tau_i + \frac{AT_i}{\rho_i} \quad \equiv \quad d_i < \tau_i + \frac{AT_i}{\rho_i} - AT_i - (\pi a_i + \pi c_i) \tag{34}$$

However, we have proved in Theorem 1 in Appendix A that d_i is not less than $\tau_i+AT_i/\rho_i-AT_i-(\pi a_i+\pi c_i)$. This means that in the LESS system it is impossible that we can drop original tasks. Considering all the four possible cases, the energy consumption of each backup task T_i on the spare is:

$$E_{SPR}(T_i) = \begin{cases} 0 & \tau_i + \frac{AT_i}{\rho_i} \le d_i \\ Ea_i & \tau_i + \frac{AT_i}{\rho_i} - \pi a_i \le d_i < \tau_i + \frac{AT_i}{\rho_i} \\ Ea_i + P\cdot(\tau_i + \frac{AT_i}{\rho_i} - d_i - \pi a_i) & \tau_i + \frac{AT_i}{\rho_i} - AT_i - \pi a_i \le d_i < \tau_i + \frac{AT_i}{\rho_i} - \pi a_i \\ Ea_i + P\cdot AT_i + Ec_i & \tau_i + \frac{AT_i}{\rho_i} - AT_i - (\pi a_i + \pi c_i) \le d_i < \tau_i + \frac{AT_i}{\rho_i} - AT_i - \pi a_i \end{cases} \tag{35}$$

Considering both the primary and spare units, the energy which is consumed by the system to execute a task T_i is:

$$E(T_i) = E_{PR}(T_i) + E_{SPR}(T_i) \tag{36}$$

where $E_{PR}(T_i)$ is given by (19) and $E_{SPR}(T_i)$ is given by (35).

IV. Energy Management Technique

In this section we aim at providing a technique to determine the parameters ρ_i and d_i to reduce the energy consumption of the LESS system. As mentioned in Sections I and II, in the proposed energy management technique, we want to exploit dynamic slacks to save energy. Therefore, since dynamic slacks result at runtime, the energy-management technique should be online and applied at runtime. Thus, an important requirement for the energy management technique is to incur a low overhead. To deal with this issue, in the proposed technique, we adopt a greedy strategy where for each task T_i, the parameters ρ_i and d_i are determined at the start of the task T_i with the aim of reducing the energy $E(T_i)$. It is noteworthy that the greedy strategy is not an optimal strategy and does not result in the minimum energy consumption. In the greedy strategy when we want to determine ρ_i and d_i, we only focus on reducing the energy consumption of the task T_i. However,

in a more sophisticated strategy, to determine the optimum values of ρ_i and d_i we not only consider the energy consumption of T_i, but also we consider the energy consumption of all the other tasks. This implies that an optimal strategy is much more complex than the greedy strategy. Since we want to exploit the strategy at runtime, the overhead (both time and energy overheads) of the strategy is very prominent. Therefore we have adopted a greedy strategy as it is simpler. Despite the simplicity of the greedy strategy, we will see in Section VI that it is quite effective to reduce the system energy consumption.

To determine the parameter d_i, as mentioned in Section II, we use (12), i.e., we set d_i to the maximum possible delay value. It should be noted that when we use (12), $[D\text{-}WCST_i\text{-}(\tau a_i+WT_i+\tau c_i)]$ is not needed to be calculated online and can be easily calculated offline for each task and stored to be used at runtime because D, τa_i, τc_i, and WT_i are known at design time and $WCST_i$ can be calculated offline at design time using (5). ST_i is the time at which the task T_{i-1} finishes running on the primary unit and the energy manager task starts running to determine ρ_i and d_i; hence ST_i is simply the current time that the internal clock of the system shows at the time the energy manager task starts.

While the parameter d_i can be simply determined at runtime using (12), the online estimation of the parameter ρ_i is not trivial. We have proved in Theorem 2 in Appendix A that the optimum value of ρ_i which minimizes the energy $E(T_i)$ depends on the actual execution time AT_i. However AT_i is not known at the start of the task T_i, which means that it is impossible to calculate the optimum value of ρ_i at the start of the task T_i. Therefore, the problem of minimizing the energy $E(T_i)$ by adjusting the parameter ρ_i is indeed an optimization problem under stochastic uncertainties (we are uncertain about the value of AT_i). One effective way to minimize such a function is to minimize the expected value of the function rather than the function itself [12]. The expected value of the energy $E(T_i)$ is:

$$E[E(T_i)] = E[E_{PR}(T_i)] + E[E_{SPR}(T_i)] \tag{37}$$

Assuming that AT_i is uniformly distributed between BT_i and WT_i, the expected value of the energy $E_{PR}(T_i)$ of (19) is:

$$E[E_{PR}(T_i)] = ET_i + (P_S + P_D\rho_i^2)\cdot\left(\frac{BT_i + WT_i}{2} + \tau_{EM}\right) \tag{38}$$

and the expected value of the energy $E_{SPR}(T_i)$ of (35) is (the related calculations have been provided in Appendix B):

$$E[E_{SPR}(T_i)] = Ea_i\cdot g(\rho_i\delta_i) + \frac{P}{\rho_i}\cdot h(\rho_i\gamma_i) + \frac{P}{\mu_i}\cdot h(\mu_i\gamma_i) + (2P\gamma_i + Ec_i)g(\mu_i\gamma_i) \tag{39}$$

where

$$h(t) = \begin{cases} 0 & WT_i \le t \\ \dfrac{(WT_i - t)^2}{2(WT_i - BT_i)} & BT_i \le t < WT_i \\ \dfrac{WT_i + BT_i}{2} - t & t < BT_i \end{cases} \quad g(t) = \begin{cases} 0 & WT_i \le t \\ \dfrac{WT_i - t}{WT_i - BT_i} & BT_i \le t < WT_i \\ 1 & t < BT_i \end{cases} \tag{40}$$

$$\delta_i \overset{\Delta}{=} d_i - \tau_i \quad , \quad \gamma_i \overset{\Delta}{=} d_i - \tau_i + \tau a_i \quad , \quad \mu_i \overset{\Delta}{=} \rho_i/(1-\rho_i) \tag{41}$$

In DVS-enabled processors the supply voltage can only take a value from a finite set of possible voltage values [20]. In our proposed online energy management technique, at the start of each task, (37) (the sum of (38) and (39)) is calculated for all the possible values of ρ_i, and then the parameter ρ_i is set to the voltage value which gives the least value for $E[E(T_i)]$. It should be noted that most of the calculations required by (38), (39), (40) and (41) can be performed offline for each task and stored to be used at runtime. For this purpose, let p, q_i, r_i, s_i, u_i, v_i, w_i, x_i, y_i, and z_i be defined as follows:

$$\begin{gathered} p \overset{\Delta}{=} 2P,\ q_i \overset{\Delta}{=} \frac{P}{\rho_i},\ r_i \overset{\Delta}{=} \frac{P}{\mu_i},\ s_i \overset{\Delta}{=} (P_S + P_D\rho_i^2)\cdot\left(\frac{BT_i + WT_i}{2} + \tau_{EM}\right) \\ u_i \overset{\Delta}{=} \frac{WT_i}{\sqrt{2(WT_i - BT_i)}},\ v_i \overset{\Delta}{=} \frac{-1}{\sqrt{2(WT_i - BT_i)}},\ w_i \overset{\Delta}{=} \frac{WT_i + BT_i}{2}, \\ x_i \overset{\Delta}{=} \frac{WT_i}{WT_i - BT_i},\ y_i \overset{\Delta}{=} \frac{-1}{WT_i - BT_i},\ z_i \overset{\Delta}{=} \frac{\tau_{EM}}{\rho_i} \end{gathered} \tag{42}$$

It can be concluded from (42) that the parameters p, q_i, r_i, s_i, u_i, v_i, w_i, x_i, y_i, and z_i can be calculated offline for each task and for each possible supply voltage (Note that all the possible values of ρ_i are known at design time). Indeed to calculate these ten parameters we need to know WT_i (task worst-case execution time), BT_i (task best-case execution time), τ_{EM} (execution time of the energy manager at V_{MAX}), P_S (static power of the primary unit at V_{MAX}), P_D (dynamic power of the primary unit at V_{MAX}), and P (spare power consumption) that all can be determined offline at design time. Using these ten parameters, (38) can be rewritten as:

$$E[E_{PR}(T_i)] = ET_i + s_i \tag{43}$$

(39) can be rewritten as:

$$E[E_{SPR}(T_i)] = Ea_i\cdot g(\rho_i\delta_i) + q_i\cdot h(\rho_i\gamma_i) + r_i\cdot h(\mu_i\gamma_i) + (p\gamma_i + Ec_i)g(\mu_i\gamma_i) \tag{44}$$

where μ_i (given by (41)) can be calculated offline for each possible supply voltage, and δ_i and γ_i are:

$$\delta_i = d_i - \tau_i = d_i - (z_{i-1} + TR_i), \quad \gamma_i = d_i - \tau_i + \tau a_i = \delta_i + \tau a_i \tag{45}$$

TR_i is given by (2), and z_i is one of the ten parameters defined in (42).

Also, (40) can be rewritten as:

$$h(t) = \begin{cases} 0 & WT_i \le t \\ (u_i + v_i t)^2 & BT_i \le t < WT_i \\ w_i - t & t < BT_i \end{cases} \quad g(t) = \begin{cases} 0 & WT_i \le t \\ x_i + y_i t & BT_i \le t < WT_i \\ 1 & t < BT_i \end{cases} \tag{46}$$

Clearly, the online calculation of (43), (44) and (46) imposes less overhead as compared to (38), (39) and (40). Fig. 3 shows the pseudo code of the proposed online energy management technique. In this code, we first determine the parameter d_i (line 1) using (12). Then we start from the minimum possible value of ρ_i (calculated in line 2 using (9) and (13)) and for each possible supply voltage we use (43) and (44) to calculate the expected energies $E[E_{PR}(T_i)]$ (lines 7 and 8), and $E[E_{SPR}(T_i)]$ (lines 9-12) respectively. The expected energy consumption of the whole system is $E[E(T_i)]=E[E_{PR}(T_i)]+E[E_{SPR}(T_i)]$ which is calculated in line

```
Inputs:
- μ[j], q_i[j], r_i[j], s_i[j], and z_i[j] where 1≤j≤NV and NV is the number of possible supply voltages.
- p, u_i, v_i, w_i, x_i, y_i, BT_i, WT_i, ρ_{i-1}, Ea_i, Ec_i, τa_i, (D-WCST_i-τa_i-WT_i-τc_i), (WT_i-τ_M+τa_i+τc_i), (K·V_MAX), (C_r·V_MAX^2), ST_i, and P

Outputs:
- ρ_i and d_i
```

```
//ρ[j] (1≤j≤NV) is the array which holds the possible supply voltages in ascending order.
//m is the index of the voltage ρ_i in the array ρ[j], i.e., ρ[m]=ρ_i and pm is the index of the
//voltage ρ_{i-1} in the array ρ[j], i.e., ρ[pm]=ρ_{i-1}.
//EP is the expected energy of the primary unit, ES is the expected energy of the spare, and E
//is the expected energy of the whole system.
//μ[j], q_i[j], r_i[j], s_i[j], and z_i[j] have been calculated offline for 1≤j≤NV, using (41)and(42).
//p, u_i, v_i, w_i, x_i, y_i, K1=(D-WCST_i-τa_i-WT_i-τc_i), K2=(WT_i-τ_M+τa_i+τc_i), K3=K·V_MAX, K4=C_r·V_MAX^2 have been
//also calculated offline.
//ST_i is the current time and is received from the system internal clock.

1:   d_i:=K1-ST_i; //Equation (12)
2:   ρ_min:=WT_i/(d_i+K2);  //Inequality (9) and Equation (13)
3:   I:=1;
4:   while(ρ[I]< ρ_min){I:=I+1;}
5:   E:=∞; // Initialize E with a very big value
6:   for j:=I to NV {
7:     dρ:=ρ[j]-ρ_{i-1};
8:     EP:=K4*dρ*dρ+s_i[j]; //Equations (17) and (43)
9:     δ_i:=d_i-r_{i-1}[pm]-K3*|dρ|;  //Equations (2) and (45)
10:    γ_i:=δ_i+τa_i;  //Equation (45)
11:    T1:=ρ[j]*δ_i; T2:=ρ[j]*γ_i; T3:=μ[j]*γ_i;
12:    ES:= Ea_i*g(T1)+q_i[j]*h(T2)+r_i[j]*h(T3)+(P*γ_i+Ec_i)*g(T3); //Equation (44)
13:    ETMP:=EP+ES;// Equation (37)
14:    if(ETMP<E) {E:=ETMP; m:=j;}}
15:  ρ_i=ρ[m]; //End of the energy manager
16:  function h(t) {
17:    if(WT_i<=t) f:=0; elseif(BT_i<=t) T:=u_i+v_i*t; f:=T*T; else f:=w_i-t;} //Equation (46)
18:  function g(t) {
19:    if(WT_i<=t) g:=0; elseif(BT_i<=t) g:=x_i+y_i*t; else g:=1;} //Equation (46)
```

Fig. 3. The pseudo code of the proposed online energy management technique (The energy manager task)

13. Then we find the ρ_i value which gives the least value for the expected energy $E[E(T_i)]$ (line 14). Finally, in line 15, we set ρ_i to the normalized voltage that gives the least energy. Line 15 is indeed the last line of the code of the energy management technique and the next lines (lines 16-19) define the functions h(t) and g(t) using (46) that are called in line 12 to calculate the expected energy $E[E_{SPR}(T_i)]$.

To detect transient faults during the execution of the energy manager task, we use temporal duplication for this task [7][18]. When a fault is detected, we do not use the results of the energy manager task and to avoid violating Rules 1 and 2 (Section II), we set the supply voltage to its maximum value (ρ_i=1) and set the delay d_i to its minimum value (d_i=0). Of course, these values (for d_i and ρ_i) do not result in any energy saving for the task T_i, but it should be noted that this situation only happens when a fault occurs during the execution of the energy manger task which is a rare event and, as mentioned in Appendix C, does not have a considerable impact on the system average energy consumption. It should also be noted that, as we will see in Section VI, the energy manager task has a negligible overhead, so that the temporal duplication of this task imposes negligible time and energy overheads.

It can be concluded from Fig. 3 that τ_{EM} (the execution time of the energy manager task at V_{MAX}) is not constant and may vary in different runs. However, for the sake of simplicity, throughout this paper, we have always assumed that τ_{EM} is constant. Indeed we have assumed that the energy manager task always consumes its worst-case execution time which is constant. It should be noted that as the energy manager task has relatively short execution times (Section VI), it does not make a significant difference if we assume that this task always consumes its worst-case execution time. Furthermore, our study has shown that if we considered the variations in τ_{EM}, the energy manager task would be much more complex and energy consuming, so that it is not worth considering the variations in τ_{EM}. Another important issue is that although (38) and (39) are derived with the assumption that AT_i is uniformly distributed, we will show in Section VI, through simulation, that this technique is quite effective to reduce the energy consumption of the LESS system even when AT_i has other distributions.

V. Reliability Model

By definition, if a safety-critical (hard) real-time system fails, a catastrophe can result [17][18]. Therefore, the probability that such a system fails, called "failure probability" [7], must be kept very low, e.g., for some applications the failure probability must be less than 10^{-9} [18]. In this section we develop an analytical model for the failure probability of the LESS system. This analytical model is used in Section VI to analyze the reliability of the LESS system and to compare it with a time-redundancy system. Note that, as discussed in Section II, here we focus only on transient faults. The occurrence of transient faults in digital systems usually follows a Poisson process [6][7][29]. It has been observed that in DVS-enabled systems the rate of transient faults increases exponentially as the supply voltage decreases, so that the fault rate can be expressed as [32]:

TABLE 1
THE ENERGY CONSUMPTION AND EXECUTION TIME OF THE BENCHMARK TASKS

Benchmark	Voltage, Frequency	Execution time (ms)	Energy Consumption (μJ)
qsort	1V,200MHz	453.93	14065.11
	0.58V,100MHz	881.56	4751.64
basicmath	1V,200MHz	707.61	20852.51
	0.58V,100MHz	1310.29	7044.64
bitcount	1V,200MHz	497.21	15883.70
	0.58V,100MHz	1009.17	5366.02
susan (smoothing)	1V,200MHz	258.68	8047.77
	0.58V,100MHz	503.35	2718.79
susan (edges)	1V,200MHz	18.89	588.03
	0.58V,100MHz	37.32	198.66
susan (corners)	1V,200MHz	10.96	337.56
	0.58V,100MHz	21.70	114.04
Energy manager task (Fig. 3)*	1V,200MHz	0.0922	2.9359
	0.58V,100Mz	0.1789	0.9658

* The reported results are for the duplicated execution of this task (Section IV).

$$\lambda(V_i) = \lambda_0 \cdot 10^{\frac{V_{MAX} - V_i}{S}} \quad (47)$$

where λ_0 is the fault rate corresponding to $V_i = V_{MAX}$, and S is the value that when the supply voltage decreases by, the fault rate increases by one order of magnitude (we assume $\lambda_0=10^{-6}$ faults/s and S=1 V [6]). Let $P_i(V_i)$ be the probability that the execution of a task T_i at the supply voltage V_i becomes faulty. Based on Poisson distribution, $P_i(V_i)$ is given by [6][27]:

$$P_i(V_i) = 1 - e^{-\lambda(V_i)\frac{AT_i}{\rho_i}} \quad (48)$$

where AT_i/ρ_i is the actual execution time of the task T_i when executed at the supply voltage V_i (Section II), and $\lambda(V_i)$ is given by (47). In the LESS system, the system fails to execute a task T_i when the original copy of T_i on the primary unit becomes faulty with the probability of $P_i(V_i)$, and also the backup copy of T_i on the spare becomes faulty with the probability of $P_i(V_{MAX})$ (note that the spare operates at V_{MAX} (Section II)). Therefore, the probability that the LESS system fails to execute a task T_i is:

$$FP(T_i) = P_i(V_i) \cdot P_i(V_{MAX}) \quad (49)$$

As $0 \le P_i(V_i) \le 1$, we conclude from (49) that:

$$FP(T_i) \le P_i(V_{MAX}) \quad \Rightarrow \quad 1 - FP(T_i) \ge 1 - P_i(V_{MAX}) \quad (50)$$

In (50), $1\text{-}FP(T_i)$ is the probability that the LESS system successfully executes the task T_i, and $1\text{-}P_i(V_{MAX})$ is the probability that when the backup copy of T_i is executed on the spare, the spare successfully executes the backup task. Hence, (50) indeed implies that the reliability of the spare is a lower bound on the reliability of the whole LESS system. Since we do not use DVS for the spare, this lower bound will not be degraded by reduced supply voltages.

Using (49), when a group of n tasks T_1 through T_n is executed on the LESS system, the probability that the system fails to execute the n tasks is:

$$FP_{Standby-Sparing} = 1 - \prod_{i=1}^{n} [1 - FP(T_i)] \quad (51)$$

We will use (51) in Section VI to calculate and compare the failure probability of the LESS system with that of a time-redundancy system.

VI. SIMULATION RESULTS

To evaluate the LESS system, we have conducted several experiments using MiBench benchmarks (Auto./Industrial set) [22], and numerous synthetic schedules. The MPARM tool [21] (cycle-accurate simulator for ARM7TDMI processor proposed in [23]) was used to obtain the power consumption and execution times. In the experiments, the processor could have five different supply voltages: 1V (200MHz), 0.86V (167MHz), 0.76V (143MHz), 0.69V (125MHz), 0.58V (100MHz), and it was assumed that the supply voltage of the RAM and cache units is scaled in proportion to the supply voltage of the processor core [11]. A 32-bit AMBA AHB interconnect [21] was used for the communication between the processors. To execute the benchmarks, we used the RTEMS embedded operating system [24]. The first set of experiments was conducted in order to investigate the energy and execution time overhead of the proposed online energy management technique. Table 1 shows the energy consumption and execution time of the benchmark tasks when executed at the supply voltages 1V, and 0.58V (the maximum and minimum supply voltages). Although temporal duplication is used for the energy manager task (Section IV), it can be seen from Table 1 that, as compared to the MiBench benchmarks, the energy and execution time overhead of this task is always less than 0.91%, which is quite negligible.

To evaluate the effectiveness of the proposed technique, we conducted another set of experiments where we compared the LESS system with a time-redundancy system. In the related literature, various implementations of time-redundancy systems have been considered (e.g., [1][2][3][5][6]). These various implementations differ in energy management technique, recovery execution policy, and slack time management. It is beyond the scope of this paper to compare the LESS system with various implementations of time-redundancy systems. Rather, we consider only one possible implementation of time-redundancy, which has been recently proposed in [29]. The main reason to choose this time-redundancy system is that it has similar restrictions and conditions to the LESS system, such as: hard real-time constraints, the use of DVS, the use of dynamic slacks, the execution of frame-based applications, and the simultaneous consideration of low energy consumption and high reliability. It is noteworthy that although [29] uses its proposed time-redundancy system to execute independent tasks, it can also be used to execute dependent tasks. Indeed, in our experiments, we used this time-redundancy system to execute dependent tasks without demanding any changes to its design. Note that we do not compare the LESS system with other hardware-redundancy techniques because: *i*) With respect to reliability, we do not claim that our proposed technique provides a better reliability than other hardware-redundancy techniques, *ii*) With respect to energy consumption, there are

TABLE 2
THE ENERGY CONSUMPTION AND FAILURE PROBABILITY OF THE LESS AND TIME-REDUNDANCY SYSTEMS*

		Relaxed time constraints: Static Slack= the biggest WT (worst case execution time) in the schedule					Tight time constraints: Static Slack= 0				
		Energy Consumption			Failure Probability		Energy Consumption			Failure Probability	
Distribution of the actual execution time	# of tasks in the schedule	Time-Redundancy system (mJ)	LESS System (mJ)	Energy Ratio$^{\Psi}$	Time-Redundancy System	LESS System	Time-Redundancy system (mJ)	LESS System (mJ)	Energy Ratio$^{\Psi}$	Time-Redundancy System	LESS System
Uniform from BT to WT	5	54.96	37.82	0.69	$10^{-11.06}$	$10^{-11.33}$	54.96	89.93	1.64	$10^{-5.27}$	$10^{-11.68}$
	10	89.17	65.85	0.74	$10^{-10.47}$	$10^{-11.00}$	89.17	130.42	1.46	$10^{-4.86}$	$10^{-11.19}$
	15	118.11	94.15	0.80	$10^{-10.14}$	$10^{-10.84}$	118.11	157.73	1.34	$10^{-4.67}$	$10^{-10.98}$
Exponential♦ Mean=(BT+WT)/2	5	54.51	35.63	0.65	$10^{-11.12}$	$10^{-11.37}$	54.51	93.95	1.72	$10^{-5.23}$	$10^{-11.65}$
	10	90.64	68.55	0.76	$10^{-10.43}$	$10^{-10.99}$	90.64	135.94	1.50	$10^{-4.89}$	$10^{-11.21}$
	15	117.27	93.64	0.80	$10^{-10.03}$	$10^{-10.79}$	117.27	158.33	1.35	$10^{-4.65}$	$10^{-10.96}$
Normal♦ Mean=(BT+WT)/2 σ=(WT-BT)/6	5	51.41	35.73	0.70	$10^{-11.03}$	$10^{-11.36}$	51.41	86.28	1.68	$10^{-5.24}$	$10^{-11.70}$
	10	85.63	65.05	0.76	$10^{-10.45}$	$10^{-11.02}$	85.63	123.42	1.44	$10^{-4.89}$	$10^{-11.26}$
	15	118.46	93.83	0.79	$10^{-10.12}$	$10^{-10.82}$	118.46	150.18	1.27	$10^{-4.67}$	$10^{-11.00}$

* For all the three distributions, it was assumed that the task worst-case execution times (i.e., WT) are uniformly distributed from 20ms to 1500ms and the task best-case execution times (i.e., BT) are uniformly distributed from 0 to WT.

♦ Exponentially and normally distributed execution times were assumed to be bounded between BT and WT [14].

Ψ Energy Ratio = Energy of the LESS system / Energy of the time-redundancy system.

very few works that have simultaneously considered hardware redundancy and low energy consumption (e.g., [15] and [30]), and these works considerably differ from ours in assumptions and application models (e.g., a single task within each frame in [15] and the requirement of independent tasks in [30]), so that it is not meaningful to compare theses works with the LESS system.

To measure the reliability of the LESS system, we have developed a software program that calculates the failure probability of the system using the analytical models provided in Section V. This software takes the required information (e.g., task supply voltages) as input from the MPARM simulator and calculates the failure probabilities. We have also developed a similar software program to measure the failure probability of the time-redundancy system. To compare the two systems, 99 static schedules similar to the schedule of Fig. 1 were generated randomly and used in the experiments. Out of these 99 random schedules, one third were generated with 5 tasks within each frame, one third with 10 tasks within each frame, and one third with 15 tasks within each frame. To generate random schedules, the worst-case execution times of the tasks were generated randomly using uniform distribution. It was assumed that the worst-case execution times of the tasks could be any value from 20ms to 1500ms. It was also assumed that the best-case execution time of each task is uniformly distributed from 0 to its worst-case execution time. For the static slack times we considered two cases: 1) relaxed time constraints: when the static slack is equal to the biggest worst-case task execution time. In this case the time-redundancy system will have enough time to re-execute any of the tasks if a fault occurs [29], 2) tight time constraints: when the static slack is so small that the time-redundancy system cannot use the static slack to re-execute any of the tasks.

For generating random static schedules we used uniform distribution (for WT_i and BT_i) as we wanted all schedules to be equally probable to be considered. However, for a specific schedule, the actual execution times of the tasks AT_i may have different probability distributions based on the system application [13]. Some research works have considered the uniform, normal, or exponential distributions for the actual execution times of the tasks [13][14]. Similarly, in our experiments, we considered these three distributions for the actual execution times AT_i. It should be noted that in the experiments the same static schedules were used for all the three distributions. Indeed at first we randomly generated 99 static schedules and then we used these static schedules with various distributions for the actual execution times AT_i. We randomly generated AT_i values before conducting simulations and then during the simulations we used these values that had been generated offline. However, during simulation experiments, the energy manager task never used these AT_i values, since we discussed in Section IV that in reality the actual execution time AT_i is not known at the start of the task T_i. Indeed, during simulation experiments, whenever a task T_i finished, it released the dynamic slack WT_i-AT_i, and this dynamic slack was exploited by the energy manager task as explained in Section IV. In all the experiments, the tasks in the synthetic schedules were selected from the MiBench benchmarks; however as we wanted to evaluate the impact of AT_i distribution, each task T_i was executed only for a duration of AT_i (AT_i / ρ_i when voltage scaling is used). With respect to the transition and activation overheads, we assumed that C_r=10μF (Equation (17)), K=10μS/V (Equation (2)), E_w=2μJ, and d_w=1ms (Section II). It should be noted that these assumptions are only used by way of example and as observed in Section IV the proposed energy management technique does not require any specific assumption about transition and activation overhead values. With respect to the communication overheads (Ecm_i, c_i, Ec_i and τc_i in Section II), it is noteworthy that our experiments show that for MiBench

benchmarks, the communication energies (Ecm_i and Ec_i) vary between 28.8 pJ (for bitcount) and 57.6 nJ (for susan). Also the communication time overheads (c_i and τc_i) vary between 20ns (for bitcount) and 41μs (for susan). Table 2 shows the energy consumption and the reliability (failure probability) of the schedules when executed on the LESS and time-redundancy systems. The following three interesting observations can be made from Table 2:

- For tight deadlines, the LESS system consumes in average 49% more energy than the time-redundancy system. However, in this case, the LESS system is far more reliable than the time-redundancy system, so that the failure probability of the LESS system is smaller than that of the time-redundancy system by a factor of about 1 million. This is because when deadlines are tight, the time-redundancy system does not have enough time for re-executing the tasks, while the LESS system is still fault tolerant as its fault-tolerance is achieved through the use of the spare and is independent from the available amount of slack time (Section II). To underline the difference between the reliabilities of the two systems, note that safety-critical real-time systems may easily require failure probabilities be less than 10^{-9} [18]. It can be seen from Table 2 that when deadlines are tight, the time-redundancy system cannot achieve the reliability required by safety-critical applications, whereas the LESS system can.
- For relaxed time constraints, both the systems can achieve the reliability required by safety-critical applications. However, the LESS system provides in average about 26% (up to 36%) energy saving as compared to the time-redundancy system. This is because, in this case, the time-redundancy system does not exploit much of its static slack and reserves it for fault tolerance (re-execution). However, in the LESS system, fault tolerance is decoupled from the slack time (Section II), hence the static slack is exploited only to reduce the energy consumption. It should also be noted that, for relaxed time constraints, the spare can be usually kept idle; hence it consumes very little energy. Indeed our experiments show that for relaxed time constraints the energy consumption of the spare is less than 3% of the whole LESS system energy consumption. This indicates how DPM is effective for the spare and we do not need to use DVS for the spare.
- Even when timing constraints are relaxed and the time-redundancy system has enough time to re-execute any of the tasks, the LESS system is still more reliable than the time-redundancy system (the failure probability of the LESS system is between 2 to 6 times smaller than that of the time-redundancy system). This is because, in the time-redundancy system, slack time is a limited resource which is shared among the tasks of a schedule [29], so that if one task consumes more slack (for re-executions), less slack will be left for the fault tolerance of the other tasks. However, in the LESS system, the spare is always available for each task to tolerate its faults, regardless of whether the spare is used for the other tasks or not. Indeed, unlike the slack time, the spare is not a consumable resource and may not be used up.

While in the experiments, we used random schedules (applications), the LESS system can be used for practical applications that require high reliability, hard real-time operation and low energy consumption. Examples of these applications are autonomous airborne (or seaborne) systems working on limited battery supply [2], automated surveillance systems [2], and implantable devices [33]. For example, many implantable devices can be considered as frame-based real-time systems where during each frame (period) a sequence of dependent tasks must be executed. For instance, a sequence of dependent tasks that must be executed on the processor of a typical implantable device, used for heart disease, is [33]: 1) Filtering, 2) Preprocessing, 3) Detection and measurement of QRS, 4) Detection of abnormalities, 5) Compression and transmission (or storage) of the results. We believe that the LESS system is a possible design candidate to implement such systems. Nevertheless, in this paper, we do not aim at providing a case study. Instead, we aim at analyzing how effective, in general, the LESS system is and this is why we have evaluated the LESS system for numerous random schedules (applications) with different number of tasks and various task execution times.

It is also worth mentioning that, while the observations in this section show that the LESS system is preferable to the time-redundancy system from energy-consumption and reliability viewpoints, this superiority comes at the price of redundant hardware resources that the LESS system uses as compared to the time-redundancy system. However, for the applications where both fault-tolerance (high reliability) and low energy consumption are required (e.g., implantable devices [33]), we believe that we have to pay this price.

VII. Summary and Conclusion

The use of hardware-redundancy techniques for real-time systems is necessary when high reliability is the primary concern. However, hardware-redundancy (e.g., duplication and TMR [18]) can increase the energy consumption by a factor of 2 or 3. In this paper we describe how a low-energy hardware-redundancy technique based on standby-sparing [18] can be used for hard real-time systems. Through an analytical approach, we have developed an online energy management technique for the low-energy standby-sparing (LESS) system which can exploit dynamic slacks to reduce the energy consumption. In this online energy management technique, we have considered voltage transition and activation overheads imposed by DVS and DPM. We have also considered both the dynamic and static energy consumptions. The experimental results show that the energy and execution time overhead of the proposed online energy management technique when applied to MiBench benchmarks (Auto./Industrial set) is always less than 0.91%, which is quite negligible. We also compared the LESS system with a low-energy time-redundancy system proposed in [29]. The results show that for relaxed time constraints, the

LESS system consumes about 26% less energy than the time-redundancy system. For tight deadlines when the time-redundancy system cannot achieve high reliability requirements of safety-critical applications, the LESS system still preserves its fault tolerance and can be used for safety-critical applications, but consumes about 49% more energy than the time-redundancy system.

APPENDIX A

Theorem 1. For each task T_i, d_i (given by (12)) is not less than $\tau_i+AT_i/\rho_i-AT_i-(\pi a_i+\pi c_i)$.

Proof. Using (9) we can write:

$$\frac{WT_i}{\rho_i}-WT_i\le SL_i \quad\Rightarrow\quad \tau_i+\frac{WT_i}{\rho_i}-WT_i-(\pi a_i+\pi c_i)\le\tau_i+SL_i-(\pi a_i+\pi c_i) \tag{52}$$

Considering that $\tau_i\le\tau_M$ (Section II), we conclude from (52):

$$\tau_i+\frac{WT_i}{\rho_i}-WT_i-(\pi a_i+\pi c_i)\le\tau_M+SL_i-(\pi a_i+\pi c_i) \tag{53}$$

The right hand side of (53) is equal to d_i (Equation (13)). Therefore, we have:

$$\tau_i+\frac{WT_i}{\rho_i}-WT_i-(\pi a_i+\pi c_i)\le d_i \tag{54}$$

It can be simply shown that:

$$AT_i\le WT_i \quad\Rightarrow\quad \tau_i+\frac{AT_i}{\rho_i}-AT_i-(\pi a_i+\pi c_i)\le\tau_i+\frac{WT_i}{\rho_i}-WT_i-(\pi a_i+\pi c_i) \tag{55}$$

Based on (54) and (55), we have:

$$\tau_i+\frac{AT_i}{\rho_i}-AT_i-(\pi a_i+\pi c_i)\le d_i \tag{56}$$

and the theorem is proved. ■

Theorem 2: The optimum value of ρ_i which minimizes the energy $E(T_i)$ (given by (36)) cannot be calculated at the start of the task T_i.

Proof: Let $\hat{\rho}_i$ be the optimum value of ρ_i which minimizes the energy $E(T_i)$. From calculus we know that this optimum value is obtained either when the derivative of $E(T_i)$ with respect to ρ_i does not exist or when this derivative is 0. From (35), it can be concluded that the values of ρ_i at which the derivative does not exist are given by:

$$\tau_i+\frac{AT_i}{\rho_i}=d_i \tag{57}$$

$$\tau_i+\frac{AT_i}{\rho_i}-\pi a_i=d_i \tag{58}$$

$$\tau_i+\frac{AT_i}{\rho_i}-AT_i-\pi a_i=d_i \tag{59}$$

$$\tau_i+\frac{AT_i}{\rho_i}-AT_i-(\pi a_i+\pi c_i)=d_i \tag{60}$$

Also, from (19), (35), and (36), it can be concluded that the values of ρ_i at which the derivative is 0 are given by:

$$C_rV_{MAX}\rho_i+P_D(AT_i+\tau_{EM})\rho_i+\frac{P}{2}KV_{MAX}\,\mathrm{sgn}(\rho_i)-P\frac{AT_i}{2\rho_i^2}=0 \tag{61}$$

We know from calculus that the optimum value $\hat{\rho}_i$ must satisfy one of the equations (57), (58), (59), (60) or (61). Now we prove that regardless of which equation ((57), (58), (59), (60) or (61)) is satisfied by $\hat{\rho}_i$, the value of $\hat{\rho}_i$ depends on the parameter AT_i (actual execution time). We know from calculus that when a variable y is independent of a variable x, we have $\partial y/\partial x=0$. Therefore, to see if $\hat{\rho}_i$ is independent of AT_i, we calculate $\partial\hat{\rho}_i/\partial AT_i$ for all Equations (57), (58), (59), (60), and (61). By calculating $\partial\hat{\rho}_i/\partial AT_i$ for (57), and (58), we obtain:

$$\frac{\partial\hat{\rho}_i}{\partial AT_i}=\frac{\hat{\rho}_i}{AT_i} \tag{62}$$

which can never be 0 as we never have $\hat{\rho}_i=0$. By calculating $\partial\hat{\rho}_i/\partial AT_i$ for (59), and (60), we obtain:

$$\frac{\partial\hat{\rho}_i}{\partial AT_i}=\frac{\hat{\rho}_i-1}{AT_i} \tag{63}$$

which can be 0 only when $\hat{\rho}_i=1$. However, when the value $\hat{\rho}_i=1$ satisfies (59) or (60), by substituting 1 for $\hat{\rho}_i$ in (59) and (60), we respectively obtain:

$$\tau_i-\pi a_i=d_i \tag{64}$$

$$\tau_i-(\pi a_i+\pi c_i)=d_i \tag{65}$$

It can be simply concluded from (13) (Section II) that the necessary condition for (64) holds is $SL_i\le\pi c_i$, and the necessary condition for (65) holds is $SL_i=0$. This implies that one possible case where $\hat{\rho}_i$ is independent of AT_i is when no slack time is available (or the slack time is very small) and hence the supply voltage must be set to its maximum value $\hat{\rho}_i=1$ regardless of the value of AT_i.

Finally, by calculating $\partial\hat{\rho}_i/\partial AT_i$ for (61), we obtain:

$$\frac{\partial\hat{\rho}_i}{\partial AT_i}=\frac{2P_D\hat{\rho}_i-P/\hat{\rho}_i^2}{2C_rV_{MAX}+2P_D(AT_i+\tau_{EM})} \tag{66}$$

which would be 0, if the optimum value could be $\hat{\rho}_i=\sqrt[3]{P/2P_D}$. However, $\hat{\rho}_i=\sqrt[3]{P/2P_D}$ does not satisfy (61) and hence (66) never becomes 0.

In short, except for when there is no slack time (or the slack time is very small and $SL_i\le\pi c_i$), we have $\partial\hat{\rho}_i/\partial AT_i\neq0$, which means that the optimum value $\hat{\rho}_i$ depends on the actual execution time AT_i. However the actual execution time is not known at the start of the task T_i. Hence, it is impossible to calculate the optimum value $\hat{\rho}_i$ at the start of the task T_i. ■

APPENDIX B

In this appendix, we consider the energy $E_{SPR}(T_i)$ of (35) as a function of the random variable AT_i and calculate the expected value of $E_{SPR}(T_i)$. For simplicity, let δ_i be defined as $\delta_i=d_i-\tau_i$, γ_i be defined as $\gamma_i=d_i-\tau_i+\pi a_i$, η_i be defined as $\eta_i=d_i-\tau_i+(\pi a_i+\pi c_i)$, and μ_i be defined as $\mu_i=\rho_i/(1-\rho_i)$, then we can rewrite (37) as:

$$E_{SPR}(T_i)=\begin{cases}0 & AT_i\le\rho_i\delta_i\\ Ea_i & \rho_i\delta_i<AT_i\le\rho_i\gamma_i\\ Ea_i+P(\dfrac{AT_i}{\rho_i}-\gamma_i) & \rho_i\gamma_i<AT_i\le\mu_i\gamma_i\\ Ea_i+P\cdot AT_i+Ec_i & \mu_i\gamma_i<AT_i\le\mu_i\eta_i\end{cases} \tag{67}$$

Let $u(t)$ and $r(t)$ be the unit step and unit ramp functions respectively, i.e.:

$$u(t)\overset{\Delta}{=}\begin{cases}1 & t>0\\ 0 & t\le 0\end{cases} \qquad r(t)\overset{\Delta}{=}t\cdot u(t) \tag{68}$$

Using the $u(t)$ and $r(t)$ functions we can rewrite (67) as:

$$E_{SPR}(\mathrm{T}_i)=Ea_i\cdot u(AT_i-\rho_i\delta_i)+\frac{P}{\rho_i}\cdot r(AT_i-\rho_i\gamma_i)+ \\ \frac{P}{\mu_i}r(AT_i-\mu_i\gamma_i)+(2P\gamma_i+Ec_i)u(AT_i-\mu_i\gamma_i) \tag{69}$$

(69) has been obtained with the assumption that AT_i is never greater than $\mu_i\eta_i$. This assumption is valid because we have:

$$AT_i\le\mu_i\eta_i \quad\equiv\quad \tau_i+\frac{AT_i}{\rho_i}-AT_i-(\varpi_i+\varpi_i)\le d_i \tag{70}$$

and we have proved in Theorem 1 in Appendix A that this inequality always holds.

Let T be a random variable with the probability density function $f(t)$ and t_0 be a possible value of T. Then the expected values of $u(T\text{-}t_0)$ and $r(T\text{-}t_0)$ can be calculated as follows:

$$E[u(T-t_0)]=\int_{-\infty}^{+\infty}f(t)u(t-t_0)dt=\int_{t_0}^{+\infty}f(t)dt=P(T>t_0) \tag{71}$$

$$E[r(T-t_0)]=E\left[\begin{cases}T-t_0 & T>t_0\\ 0 & T\le t_0\end{cases}\right]= \\ E[T-t_0\,|\,T>t_0]\cdot P(T>t_0)+0\cdot P(T\le t_0)= \\ E[T-t_0\,|\,T>t_0]\cdot P(T>t_0) \tag{72}$$

where P(I) is the probability that the inequality I holds.

We use (71) and (72) to calculate the expected value of the energy $E_{SPR}(\mathrm{T}_i)$ of (69) as follows:

$$E[E_{SPR}(\mathrm{T}_i)]= \\ Ea_i\cdot P(AT_i>\rho_i\delta_i)+\frac{P}{\rho_i}\cdot E[AT_i-\rho_i\gamma_i\,|\,AT_i>\rho_i\gamma_i]\cdot P(AT_i>\rho_i\gamma_i)+ \\ \frac{P}{\mu_i}\cdot E[AT_i-\mu_i\gamma_i\,|\,AT_i>\mu_i\gamma_i]\cdot P(AT_i>\mu_i\gamma_i)+(2P\gamma_i+Ec_i)P(AT_i>\mu_i\gamma_i) \tag{73}$$

Assuming that AT_i is uniformly distributed between BT_i and WT_i, we have:

$$P(AT_i>t)=\begin{cases}0 & WT_i\le t\\ \dfrac{WT_i-t}{WT_i-BT_i} & BT_i\le t<WT_i\\ 1 & t<BT_i\end{cases} \tag{74}$$

$$E[AT_i-t\,|\,AT_i>t]=\begin{cases}0 & WT_i\le t\\ \dfrac{WT_i-t}{2} & BT_i\le t<WT_i\\ \dfrac{WT_i+BT_i}{2}-t & t<BT_i\end{cases} \tag{75}$$

and hence we have:

$$E[AT_i-t\,|\,AT_i>t]\cdot P(AT_i>t)=\begin{cases}0 & WT_i\le t\\ \dfrac{(WT_i-t)^2}{2(WT_i-BT_i)} & BT_i\le t<WT_i\\ \dfrac{WT_i+BT_i}{2}-t & t<BT_i\end{cases} \tag{76}$$

Let $g(t)$ be defined as P(AT_i>t) and $h(t)$ be defined as $E[AT_i\text{-}t|AT_i>t]$.P(AT_i>t), then we can rewrite (73) as:

$$E[E_{SPR}(\mathrm{T}_i)]=Ea_i\cdot g(\rho_i\delta_i)+\frac{P}{\rho_i}\cdot h(\rho_i\gamma_i)+\frac{P}{\mu_i}\cdot h(\mu_i\gamma_i)+(2P\gamma_i+Ec_i)g(\mu_i\gamma_i) \tag{77}$$

APPENDIX C

In this appendix, by means of a numerical example, we show that to calculate the energy consumption of fault-tolerant systems, we do not need to consider the cases where the system tolerates a fault, because such cases are rare and hence have a negligible impact on the average energy consumption. Suppose that a fault tolerant system consumes 5mJ to execute a task T, and if a fault occurs during the execution of this task, the system requires an extra energy of 7mJ to tolerate the fault. Also assume that the probability that a fault occurs during the execution of this task is 10^{-6}. This assumption means that if no fault-tolerance mechanism were used, the failure probability would be 10^{-6} which is a reasonable assumptions based on the observations in Section VI. In this example, considering the extra energy consumption of fault tolerance, the expected value (average) of the energy which is consumed by the system to execute the task is:

$$E=(1-10^{-6})\times 5\mathrm{mJ}+10^{-6}\times 12\mathrm{mJ}=5.000007\,\mathrm{mJ}$$

which is almost equal to the energy consumed in the fault-free case, i.e. 5mJ. It is noteworthy that while a failure probability of 10^{-6} is considerable from a reliability point of view (quite bigger than acceptable values of failure probability [18]), it is negligible from an energy consumption point of view so that we can assume no fault occurs when analyzing the energy consumption.

REFERENCES

[1] V. Izosimov, P. Pop, P. Eles, and Z. Peng, "Scheduling of Fault-Tolerant Embedded Systems with Soft and Hard Timing Constraints", in *Proc. Design, Automation and Test in Europe (DATE '08)*, pp. 915-920, 2008.

[2] R. Melhem, D. Mosse, and E. Elnozahy, "The interplay of power management and fault recovery in real-time systems," *IEEE Trans. Computers*, vol. 53, no. 2, pp. 217-231, 2004.

[3] Y. Zhang and K. Chakrabarty, "Dynamic adaptation for fault tolerance and power management in embedded real-time systems," *ACM Tran. Embedded Computing Systems*, vol. 3, no. 2, pp. 336-360, 2004.

[4] F. Liberato, R. Melhem, and D. Mosse, "Tolerance to multiple transient faults for aperiodic tasks in hard real-time systems," *IEEE Trans. Computers*, vol. 49, no. 9, pp. 906-914, 2000.

[5] P. Eles, V. Izosimov, P. Pop, and Z. Peng, "Synthesis of Fault-Tolerant Embedded Systems", in *Proc. Design, Automation and Test in Europe (DATE '08)*, pp. 1117-1122, 2008.

[6] A. Ejlali, B.M. Al-Hashimi, M.T. Schmitz, P. Rosinger, and S.G. Miremadi, "Combined Time and Information Redundancy for SEU-Tolerance in Energy-Efficient Real-Time Systems", *IEEE Trans. VLSI Sys.*, vol. 14, no. 4, pp. 323-335, 2006.

[7] I. Koren, and C. M. Krishna, *Fault-Tolerant Systems*, Morgan Kaufmann, Elsevier, 2007.

[8] Y. Zhang and K. Chakrabarty, "A Unified Approach for Fault Tolerance and Dynamic Power Management in Fixed-Priority Real-Time Embedded Systems", *IEEE Trans. CAD*, vol. 25, no. 1, pp. 111-125, 2006.

[9] F. Cottet, J. Delacroix, C. Kaiser, and Z. Mammeri, *Scheduling in Real-Time Systems*, John Wiley & Sons, 2002.

[10] M. T. Schmitz, B. M. Al-Hashimi, and P. Eles, *System-Level Design Techniques for Energy-Efficient Embedded Systems*, Norwell, MA: Kluwer, 2004.
[11] T. D. Burd, T. A. Pering, A. J. Stratakos, and R. W. Brodersen, "A dynamic voltage scaled microprocessor system," *IEEE J. Solid-State Circuits*, vol. 35, no. 11, pp. 1571-1580, 2000.
[12] K. Marti, *Stochastic Optimization Methods*, Second Edition, Springer, 2008.
[13] P. Li, and B. Ravindran, "Fast, Best-Effort Real-Time Scheduling Algorithm", *IEEE Trans. Computers*, vol. 53, no. 9, pp. 1159-1175, 2004.
[14] H. Aydin, R. Melhem, D. Mosse, and P. Mejia-Alvarez, "Power-Aware Scheduling for Periodic Real-Time Tasks", *IEEE Trans. Computers*, vol. 53, no. 5, pp. 584-600, 2004.
[15] D. Zhu, R. Melhem, D. Mosse, and E. Elnozahy, "Analysis of an energy efficient optimistic TMR scheme", in *Proc. 10th Int'l Conf. Parallel and Distributed Systems (ICPADS 2004)*, pp. 559-568, 2004.
[16] S. Poledna, *Fault-tolerant real-time systems: The problem of replica determinism*, Kluwer Academic Publishers, 1996.
[17] H. Kopetz, *Real-time systems: Design principles for distributed embedded applications*, Kluwer Academic Publishers, 2002.
[18] D.K. Pradhan, *Fault-tolerant computer system design*, Prentice-Hall, 1996.
[19] R. Jejurikar, and R. Gupta, "Dynamic slack reclamation with procrastination scheduling in real time embedded systems", in *Proc. Design Automation Conference (DAC 2005)*, pp. 111-116, 2005.
[20] *TM5400/TM5600 Data Book*, Transmeta Corp., Santa Clara, CA, 2000.
[21] MPARM Tool, http://www-micrel.deis.unibo.it/sitonew/research/mparm.html. 2005.
[22] M. R. Guthaus, J. S. Ringenberg, D. Ernst,T. M. Austin, T. Mudge, and R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite", in *Proc. IEEE 4th annual Workshop on Workload Characterization*, pp. 83-94, 2001.
[23] L. Benini, D. Bertozzi, A. Bogoliolo, F. Menichelli, and M. Olivieri., "MPARM: Exploring the Multi-Processor SoC Design Space with SystemC", *The Journal of VLSI Signal Processing*, vol. 41, no. 2, pp. 169-182, 2005.
[24] RTEMS Operating System, http://www.rtems.com. 2010.
[25] A. Andrei, P. Eles, Z. Peng, M.T. Schmitz, and B.M. Al-Hashimi, "Energy Optimization of Multiprocessor Systems on Chip by Voltage Selection", *IEEE Trans. VLSI Sys.*, vol. 15, no. 3, pp. 262-275, 2007.
[26] J. Luo, L. Yan, and N. Jha. "Combined Dynamic Voltage Scaling and Adaptive Body Biasing for Heterogeneous Distributed Real-time Embedded Systems", in *Proc. Int'l Conf. Computer-Aided Design (ICCAD'03)*, pp. 30-37, 2003.
[27] P. Pop, K.H. Poulsen, V. Izosimov, and P. Eles, "Scheduling and Voltage Scaling for Energy/Reliability Trade-offs in Fault-Tolerant Time-Triggered Embedded Systems", in *Proc. Int'l Conf. Hardware-Software Codesign and System Synthesis (CODES+ISSS'07)*, pp. 233-238, 2007.
[28] A. Ejlali, B.M. Al-Hashimi, and P. Eles, "A Standby-Sparing Technique with Low Energy-Overhead for Fault-Tolerant Hard Real-Time Systems", in *Proc. 7th Int'l Conf. Hardware/Software Codesign and Sys. Synthesis (CODES+ISSS'09)*, pp. 193-202, 2009.
[29] B. Zhao, H. Aydin, and D. Zhu, "Enhanced Reliability-Aware Power Management through Shared Recovery Technique", in *Proc. Int'l Conf. Computer-Aided Design (ICCAD'09)*, pp. 63-70, 2009.
[30] O.S. Unsal, I. Koren, and C.M. Krishna, "Towards Energy-Aware Software-Based Fault Tolerance in Real-Time Systems", in *Proc. ACM/IEEE Int'l Symp. Low Power Electronics and Design (ISLPED'02)*, pp. 124-129, 2002.
[31] Y. Liu, H. Liang, and K. Wu, "Scheduling for Energy Efficiency and Fault Tolerance in Hard Real-Time Systems", in *Proc. Design, Automation & Test in Europe (DATE 2010)*, pp. 1444-1449, 2010.
[32] D. Zhu, R. Melhem, and D. Mosse, "The Effects of Energy Management on Reliability in Real-Time Embedded Systems," in *Proc. Intl. Conf. Computer Aided Design (ICCAD 2004)*, pp. 35-40, 2004.
[33] M. Kumar Das, *Modern Pacemakers - Present and Future*, Intech Publisher, 2011.

Alireza Ejlali received the PhD degree in computer engineering from Sharif University of Technology in 2006. He is an associate professor of Computer Engineering at Sharif University of Technology, Tehran, Iran. From 2005 to 2006, he was a visiting researcher in the Electronic Systems Design Group, University of Southampton, United Kingdom. He is now the director of Computer Architecture Group and the director of the Embedded Systems Research Laboratory (ESR-LAB) at the Department of computer engineering, Sharif University of Technology. His research interests include low power design, real-time embedded systems, and fault tolerant embedded systems.

Bashir M. Al-Hashimi (M'99-SM'01-F'09) received the B.Sc. degree (with 1st-class classification) in Electrical and Electronics Engineering from the University of Bath, UK, in 1984 and the Ph.D. degree from York University, UK, in 1989. Following this he worked in the microelectronics design industry and in 1999, he joined the School of Electronics and Computer Science, Southampton University, UK, where he holds the Endowment ARM Chair in Computer Engineering, Director of the Pervasive Systems Center, and Dean of Research, Faculty of Physical and Applied Sciences. He has authored one book on SPICE simulation, (CRC Press, 1995), and coauthored two books, Power Constrained Testing of VLSI circuits (Springer, 2002), and System-Level Design Techniques for Energy-Efficient Embedded Systems (Springer, 2004). He edited the book, System-on-Chip: Next Generation Electronics (IEE Press, 2006). He has published over 250 refereed papers in journals conference proceedings.

Prof. Al-Hashimi is a Fellow of the IEE and a Fellow of the British Computer Society. He is the Editor-in-Chief of the IEE Proceedings: Computers and Digital Techniques, and a member of the editorial board of the Journal of Electronic Testing: Theory and Applications (JETTA), and Journal of Low Power Electronics. He was the General Chair of the 11th IEEE European Test Symposium (UK 2006), Technical-Programme Chair of DATE 2009, and the General Chair of DATE 2011. He is the coauthor of two Best Paper Awards: the James Beausang at the ITC 2000, relating to low power BIST for RTL data paths, and at the CODES-ISSS Symposium 2009, relating to low-energy fault-tolerance techniques. He is a co-author of a paper on test data compression which has been selected for a Springer book featuring the most influential work over the ten years of the DATE conference.

Dr. Petru Eles is Professor of Embedded Computer Systems with the Department of Computer and Information Science (IDA), Linköping University. Petru Eles' current research interests include embedded systems, real-time systems, electronic design automation, cyber-physical systems, hardware/software codesign, low power system design, fault-tolerant systems, design for test. He has published a large number of technical papers in these areas and co-authored several books. Petru Eles received two best paper awards at the European Design Automation Conferences (EURO-DAC) in 1992 and 1994, a best paper award at the Design Automation and Test in Europe Conference (DATE) in 2005, a best paper award at the International Conference on Hardware/Software Codesign and System Synthesis (CODES/ISSS) in 2009, and a best presentation award at the 2003 CODES/ISSS.

Petru Eles is an Associate Editor of the IEEE Transactions on Computers, the IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, and of the IET Proceedings - Computers and Digital Techniques. Petru Eles has been an IEEE CAS Distinguished Lecturer for 2004 - 2005.