Energy/Reliability Trade-offs in Fault-Tolerant Event-Triggered Distributed Embedded Systems

Junhe Gan[1] Flavius Gruian[2] Paul Pop[1] Jan Madsen[1]

[1] Department of Informatics and Mathematical Modelling
Technical University of Denmark, Denmark
{juga, pop, jan}@imm.dtu.dk

[2] Department of Computer Science
Lund University, Sweden
flavius.gruian@cs.lth.se

Abstract— This paper presents an approach to the synthesis of low-power fault-tolerant hard real-time applications mapped on distributed heterogeneous embedded systems. Our synthesis approach decides the mapping of tasks to processing elements, as well as the voltage and frequency levels for executing each task, such that transient faults are tolerated, the timing constraints of the application are satisfied, and the energy consumed is minimized. Tasks are scheduled using fixed-priority preemptive scheduling, while replication is used for recovery from multiple transient faults. Addressing energy and reliability simultaneously is especially challenging, since lowering the voltage to reduce the energy consumption has been shown to increase the transient fault rate. We presented a Tabu Search-based approach which uses an energy/reliability trade-off model to find reliable and schedulable implementations with limited energy and hardware resources. We evaluated the algorithm proposed using several synthetic and real-life benchmarks.

I. Introduction

Safety-critical applications have to function correctly, meet their timing constraints and be energy-efficient even in the presence of faults. Such faults might be permanent (e.g., damaged microcontrollers or communication links), transient (e.g., caused by electromagnetic interference), or intermittent (appear and disappear repeatedly). The transient faults are the most common [1], and their number is increasing due to the rising level of integration in semiconductors.

Researchers have proposed several hardware architecture solutions, such as TTA [2], that rely on hardware replication to tolerate a single permanent fault in any of the components of a fault-tolerant unit. Such approaches can be used for tolerating transient faults as well, but they incur great hardware costs. Alternatives to such purely hardware-based solutions are approaches such as re-execution, replication, and checkpointing. Several researchers have shown how the schedulability of an application can also be guaranteed with appropriate levels of fault-tolerance [3–5].

With regard to energy minimization, the most common approach that allows energy/performance trade-offs during runtime of the application is dynamic voltage and frequency scaling (DVFS) [6]. DVFS aims at reducing the dynamic power consumption by scaling down operational frequency and circuit supply voltage. A considerable amount of work has been done on DVFS. See [6] for a survey.

Incipient research has analyzed the interplay of energy/performance trade-offs and fault-tolerance techniques [7–9]. Redundancy-based fault-tolerance techniques (such as re-execution and replication) and DVFS-based low-power techniques compete for the available slack. The interplay of power management and fault recovery has been addressed in [8], where checkpointing policies were evaluated with respect to energy. In [7], time redundancy was used in conjunction with information redundancy, which does not compete with DVFS for slack, to tolerate transient faults. In [9], fault tolerance and dynamic power management were studied, and rollback recovery with checkpointing was used to tolerate multiple transient faults in distributed systems.

Addressing energy and reliability simultaneously is especially challenging because lowering the voltage to reduce energy consumption has been shown to increase the number of transient faults exponentially [10]. The main reason for such an increase is that, with lower voltages, even very low energy particles are likely to create a critical charge that leads to a transient fault. However, this aspect has received very limited attention. Zhu [11] has proposed a reliability-aware DVFS heuristic for uni-processor systems, and a single-task checkpointing scheme was evaluated in [10]. Researchers have also addressed reliability in the context of temperature-aware design. Their focus has been on limiting the operating temperature in order to increase the life-time of the system [12]. Such a technique could be used in conjunction with our approach.

In [13], we show how re-execution and active replication can be combined in an optimized implementation that leads to a schedulable fault-tolerant application without increasing the resources required. In [14], we consider the energy versus reliability trade-offs in the context of distributed time-triggered systems, where tasks and messages are scheduled based on a static-cyclic scheduling policy, and transient faults are tolerated using task re-execution.

In this paper, we consider heterogeneous distributed event-triggered systems, where tasks are scheduled using fixed-priority preemptive scheduling, and messages are scheduled using fixed-priority non-preemptive scheduling. Transient faults are tolerated through task replication. In this context, we propose an optimization approach that decides offline the mapping of tasks to processing elements and the voltage and frequency levels for executing each task, such that transient faults are tolerated, the timing constraints of the application are satisfied, and the energy consumed is minimized. We have developed a novel Tabu Search-based approach for the synthesis of fault-tolerant event-driven systems that takes into account the influence of voltage and frequency scaling on reliability. Our offline approach can be used in conjunction with existing online DVFS

978-1-4244-7516-2/11/$26.00 ©2011 IEEE

techniques [15] to further reduce the energy consumption under reliability constraints.

The next two sections present the system architecture and the application model, respectively. The energy and reliability models are presented in Section IV. Section V presents our problem formulation and a motivational example. Section VI outlines our proposed reliability-aware mapping, voltage and frequency scaling approach (MVFS). An evaluation of the proposed algorithm is presented in the last section.

II. System Model

We consider hardware architectures consisting of a set $\mathcal{N}$ of heterogeneous processing elements (PEs) and interconnected by a communication channel. Tasks are scheduled using fixed-priority preemptive scheduling. We have shown [16] how realistic protocols such as FlexRay can be taken into account during the analysis and synthesis. However, for simplicity, in this paper we use a simple non-preemptive fixed-priority bus. We ignore the energy consumption of the bus, but we do take the bus timing into account during schedulability analysis.

In this paper we address the energy consumption in PE, which is a major component of the system-level energy consumption, and use the system-level power model from [17]. Thus, a processing element N_i is characterized by a set of operating modes, where each operating mode, $\Lambda_j^{N_i} = (f_j^{N_i}, v_j^{N_i}, p_j^{N_i})$, is described by three parameters: $f_j^{N_i}$ is the operating clock frequency of N_i running in mode j, which is measured in Hz; $v_j^{N_i}$ is the supply voltage measured in Volts; $p_j^{N_i}$ is the power spent measured in Watts. We denote normalized frequency F and normalized voltage V where $F_j^{N_i} = f_j^{N_i}/f_{max}^{N_i}$ and $V_j^{N_i} = v_j^{N_i}/v_{max}^{N_i}$, respectively.

Switching between operating modes requires time and energy. We take this overhead into account through a matrix of time overheads X and a matrix of energy overheads Y. Each element $X_{j\ell}$, $j \neq \ell$, is the time overhead required to mode-switch from j to ℓ. Similarly, each element $Y_{j\ell}$, $j \neq \ell$, denotes the energy consumption when switching modes from j to ℓ.

In this paper we are interested in tolerating transient faults, which are the most common faults in today's embedded systems [1]. The fault rate λ has been reported [10] in the range 10^{-8}–10^{-6} faults per second on each chip. We assume that faults are detected at the completion of a task's execution, and the overhead of fault detection is included in the task's worst-case execution time (WCET). In [13] we have shown how re-execution, which provides time-redundancy, and active replication, which provides spatial-redundancy, can be combined in an optimized implementation that reduces the fault-tolerance overheads. In this paper, we use active replication to tolerate transient faults.

III. Application Model

We model an application as a set Γ of periodic real-time tasks. The mapping of a task τ_i to N_j is captured by the function $\mathcal{M}: \Gamma \rightarrow \mathcal{N}$, i.e., $\mathcal{M}(\tau_i) = N_j$. This mapping is not yet known and will be decided by our approach presented in Section VI. For each task τ_i we know the WCETs of $C_i^{N_j}$ (in the maximum speed operating mode) with respect to each N_j where it could be mapped and executed, the period T_i and deadline D_i. We assume that the deadlines are smaller or equal to the periods, i.e., $D_i \leq T_i$. Each task τ_i is assigned a unique priority.

Tasks are divided into two categories: critical and non-critical. To prevent the application failure, critical tasks have to tolerate transient faults. We assume that for each critical task τ_i, the designer will specify a desired *redundancy level* k_i, i.e., how many replications of τ_i have to be introduced. The desired redundancy level depends on the functionality and implementation of the task, and is captured by the function $\mathcal{F}: \Gamma \rightarrow \mathbb{N}$, i.e., $\mathcal{F}(\tau_i) = k_i$. If $k_i = 0$, the task is non-critical. A critical task and its replicas could be mapped on the same PE and run at different modes, or mapped on different PEs.

Each execution of a periodic task is called a job. The jth job of task τ_i is referred to as J_{ij}. In this paper, we assume that all the jobs J_{ij} of task τ_i are assigned offline to the same operating mode[1] (i.e., run in the same operating voltage and frequency). The assigned mode of task τ_i is captured by the function $\mathcal{L}$: $\Gamma \rightarrow \left\{\Lambda_\ell^{\mathcal{M}(\tau_i)}\right\}$. For simplicity, we do not consider the situation when an intermediate mode can be obtained by using two modes applied to different execution segments of the same task, as in [17]. However, our approach can be extended to consider this aspect.

Tasks communicate asynchronously through buffers, i.e., a reader task will block if the buffer is empty and a writer task will block if the buffer is full. We assume that the buffer size have been determined such that there is no overflow or underflow [18], and we know the size of each message.

IV. Energy/Reliability Trade-off Model

The reliability of executing a task, R_i, is defined as the probability of its successful execution, and it is captured by the exponential failure law [19],

$$R_i = e^{-\lambda c} \tag{1}$$

where λ is the average fault rate, and c is the execution time of the task.

The reliability of a critical task, R_i^{rep}, is increased through introducing k_i replicas. R_i^{rep} is expressed as the probability of *not* having all these tasks fail,

$$R_i^{rep} = 1 - \prod_{i=1}^{k_i+1}(1 - R_i) \tag{2}$$

Note that when $k_i = 0$ (i.e., the task is non-critical), $R_i^{rep} = R_i$. Considering independent faults, the reliability R_s of the system is calculated by,

$$R_s = \prod_{i=1}^{|\Gamma'|} R_i^{rep} \tag{3}$$

where $|\Gamma'|$ denotes the number of tasks in Γ', the complete set of tasks, including the replicas.

The assigned redundancy levels are only valid under the assumption that the system reliability R_s does not fall below a

[1] Jobs might execute in different modes depending on the online power management scheme.

specified reliability goal R_g. If R_s is below R_g, the fault rate might be higher and thus more redundancy might be necessary to tolerate the increased number of faults.

For a task τ_i mapped on N_j and executed in mode ℓ, its WCET c_i is calculated by: $c_i = C_i^{N_j}/F_\ell^{N_j}$. The application is executed periodically with a hyperperiod[2] $T_{\Gamma'}$. The energy consumption E_S of the task set Γ' within $T_{\Gamma'}$ is calculated by:

$$E_S = \sum_{\tau_i \in \Gamma'} \left\lceil \frac{T_{\Gamma'}}{T_i} \right\rceil \cdot p_\ell^{N_j} \cdot c_i + O \tag{4}$$

where $p_\ell^{N_j} \cdot c_i$ is the energy consumed by each job of τ_i mapped on N_j and run in mode ℓ, $\left\lceil \frac{T_{\Gamma'}}{T_i} \right\rceil$ is the number of τ_i's jobs within $T_{\Gamma'}$, and O is the sum of mode switching overheads detected by the schedulability analysis.

However, the energy is also linked to reliability, not only performance. The fault rate is determined by the operating mode which the system runs in, i.e., the supply voltage v and the operating frequency f. In [10], it is assumed that when PE runs in the minimum voltage and frequency level, the fault rate increases 10^d times compared to that of PE running in the maximum voltage and frequency level.

$$\lambda_{max} = \lambda(\Lambda_{min}^{N_j}) = \lambda(\Lambda_{max}^{N_j}) \cdot 10^d = \lambda_{min} \cdot 10^d \tag{5}$$

where d (> 0) is an PE-specific constant.

We derive Eq. 6 from [10][3]. Recall that normalized frequency F and normalized voltage V are used.

$$\lambda(\Lambda_\ell^{N_j}) = \lambda_0 \cdot F^\alpha \cdot 10^{-\beta V} \tag{6}$$

where $\lambda_0 = \lambda_{min}$ is the average fault rate of a system running on the maximum speed operating mode with the highest operating frequency and supply voltage, α and β could be calculated by Eq. 5 and Eq. 6.

We can now update the Eq. 1. When a periodic task τ_i is mapped on N_j and executed in mode ℓ, its reliability R_i is calculated by:

$$R_i = e^{-\lambda(\Lambda_\ell^{N_j})\left\lceil \frac{T_{\Gamma'}}{T_i} \right\rceil c_i} \tag{7}$$

Since the fault rate increases exponentially when supply voltage and operating frequency decrease, switching the operating mode has an impact on the system reliability.

V. Problem Formulation

The problem we are addressing in this paper can be formulated as follows. Given (1) an application modeled as a set Γ of tasks, with associated redundancy levels captured by function $\mathcal{F}$, (2) an architecture consisting of a set $\mathcal{N}$ of heterogeneous PEs that work under different operating modes Λ, and (3) a reliability goal R_g which bounds the system reliability R_s, we are interested in synthesizing an implementation $\mathcal{S}$, such that the deadlines of all tasks are satisfied, the system reliability is within the imposed reliability goal, i.e., $R_s \geq R_g$, and the energy consumption is minimized.

Synthesizing an implementation $\mathcal{S} = (\mathcal{M}, \mathcal{L})$ means deciding offline (1) the mapping $\mathcal{M}$ of tasks and replicas to the PEs and (2) the operating mode $\mathcal{L}$ for executing each task.

A. Motivational Example

Let us consider the example in Fig.1 where we have a task set of six tasks (four tasks and two replicas) mapped on an architecture of two PEs. The WCETs, periods, deadlines and priorities of all tasks are presented in Fig.1(a). τ_1 and τ_2 are critical tasks and they are replicated once. We denote the replicated task of τ_i as τ_i' and its jobs as J_{ij}'. Recall that tasks are periodic and they are executed in accordance to fixed-priority preemptive scheduling (e.g. RM).

The operating modes of two PEs are given in Fig.1(b), which also contains λ_0, α, and β (we assume the same values for both PEs in this example, but they should be PE-specific). The hyperperiod of the application is $T_\Gamma = 100$. In Fig.1(c) and (d), the height of the rectangle indicates the operating mode which the task is running in. The timeline on each PE considers the scenario when all tasks are released at $t = 0$ and executed up to their WCETs. However, tasks might be released at different times and could be finished earlier. In Section VI.A, we discuss the schedulability analysis and the assumptions considered.

For the discussion, we use a reference which runs all the tasks in the maximum speed operating mode and attempts to reduce the energy consumption by mapping the tasks on the low energy consumed PEs, without missing the deadlines. We present the referenced energy consumption E_0 and system reliability R_s^0 after (a) and (b). The reliability goal R_g is given as well.

The typical approach for energy minimization is to decide the mapping and the operating mode for each task such that

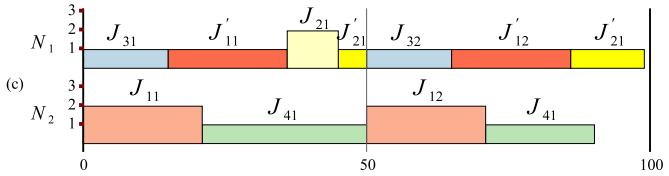
(a)

Γ'	$C_i^{N_1}$	$C_i^{N_2}$	$T_i = D_i$	Priority
τ_1	7	14	50	1
τ_2	6	12	100	4
τ_3	5	10	50	2
τ_4	8	16	100	5
τ_1'	7	14	50	3
τ_2'	6	12	100	6

(b)

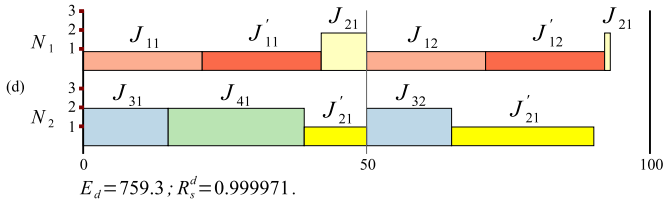
Operating Modes	N_1 Freq. [MHz]	N_1 Volt. [V]	N_1 Power [W]	N_2 Freq. [MHz]	N_2 Volt. [V]	N_2 Power [W]
1	333	1.2	4	166	1.1	2
2	666	1.4	12	333	1.25	4.5
3	1000	1.6	25	500	1.5	11
	$\lambda_0 = 10^{-6}, \alpha = -4, \beta = -0.04$					

E_0=1312 ; R_s^0=0.999996. R_g=0.99996.



E_c=753.4 ; R_s^c=0.999362.

Energy reduction by 42.58%. Prob. of failure increased by 160 times.



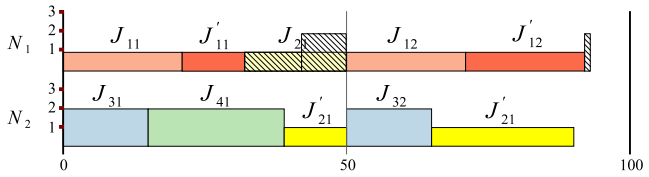
E_d=759.3 ; R_s^d=0.999971.

Energy reduction by 42.13%. Prob. of failure increased by 7 times.

Fig. 1 Mapping and VFS Example

[2]hyperperiod is equal to the least common multiple of all tasks' periods.

[3] [10] provides the fault rate formula when a system runs in normalized frequency F and the corresponding normalized voltage $V = F \cdot V_{max} = F$. But we also consider the case $V \neq F$.



E_s=647.3 ; R_s=0.999962.

Energy reduction by 50.66%. Prob. of failure increased by 10 times.

Fig. 2 Runtime behavior for an arbitrary instance

the energy consumed is minimized and the deadlines are satisfied. Such an implementation, denoted with $\mathcal{S}_c$, is shown in Fig.1(c). $\mathcal{S}_c$ meets all tasks' deadlines and reduces the energy by 42.58%, compared to E_0. However, $\mathcal{S}_c$ does not meet the system reliability goal. Since $1 - R_s$ captures the probability of failure, we use Eq. 8 to measure the degeneration of the system reliability:

$$\theta = \frac{1-R_s}{1-R_s^0} \tag{8}$$

where R_s^0 is the system reliability of initial solution. According to Eq. 8, the probability of failure for $\mathcal{S}_c$ is 160 times greater than that of the initial solution.

By carefully deciding the mapping $\mathcal{M}$ and operating modes $\mathcal{L}$, it is possible to reduce the negative impact on reliability without a significant loss of energy savings. Another implementation $\mathcal{S}_d$ shown in Fig.1(d) fulfills all timing and reliability requirements, with only 0.45% loss in energy savings while comparing the energy reduction result between $\mathcal{S}_c$ and $\mathcal{S}_d$.

Additional energy savings can be obtained at runtime by exploiting the available slack, because, for instance, a task may finish before its WCET. Several such online management schemes have been proposed, such as [15]. Our offline optimization can work in conjunction with any of them. The concern is whether we can obtain further gains without impacting the reliability goal. Any online DVFS scheme would have to use R_g as a constraint.

Fig.2 considers the case in Fig.1(d) where J'_{11} finished earlier. An online DVFS scheme can, in principle, avoid executing replicas if the current job of the original task does not experience a fault, but such a discussion is beyond the scope of this paper. The shadow of the rectangle shows the offline optimized solution of Fig.1(d). At runtime, the available slack will be exploited by scaling voltage and frequency to execute J_{21}, which further increases the energy savings to 50.66%. Although the reliability is slightly decreased to 0.999961, it is still within R_g. In general, a task that finishes earlier will increase reliability (Eq. 7) and this can be translated into energy gains.

VI. Tabu Search-based Algorithm

The problem presented in the previous section is NP-hard. To solve it, we propose a Tabu Search-based approach, which decides offline the mapping $\mathcal{M}$ and operating mode $\mathcal{L}$ for executing each task. Tabu Search (TS) [20] is an optimization metaheuristic which iteratively explores the solutions in the vicinity (neighborhood) of the current solution, selecting the ones which optimize the *cost* funtion. TS would avoid being stuck in a local optimum by allowing selection of non-improving solutions. Our proposed *cost* funtion in Eq. 9 captures the energy minimization under timing and reliability constraints:

$$cost(\mathcal{S}) = E_{\mathcal{S}} + W_R \cdot \max(0, R_g - R_s) + W_r \cdot \max(0, r_{\mathcal{S}}) \tag{9}$$

where the first term is the energy consumption of running the implementation $\mathcal{S}$ within $T_{\Gamma'}$, the second term is the reliability constraint and the third term represents the timing constraint. Instead of considering solutions which do not meet timing and reliability constraints as infeasible, we chose to penalize them heavily in the *cost* function by giving large values to W_R and W_r. This allows us to explore infeasible regions of the search space and drive the search towards feasible regions. $r_{\mathcal{S}}$ is "degree of schedulability" which is presented in Section VI.A.

Algorithm 1 presents our reliability-aware mapping, voltage and frequency scaling optimization (MVFS). MVFS takes the complete the task set Γ' and the architecture $\mathcal{N}$ as inputs, and returns the implementation $\mathcal{S} = (\mathcal{M}, \mathcal{L})$ which minimizes the *cost* function. The search starts from an initial solution $\mathcal{S}^0$ (line 1) which is a mapping such that the utilization of the PEs is balanced and the communications are minimized. All tasks are assigned the maximum speed operating mode.

The neighborhood of the current solution is generated using design transformations (moves) that change the current system implementation $\mathcal{S}^{now}$ (lines 3 and 5). The neighborhood might be very large, thus we consider a limited number of neighboring solutions called a *candidate set*, denoted with $\mathcal{C}$. We perform two types of design moves in MVFS: (1) mapping moves ($\mathcal{M}$-moves) and (2) voltage and frequency scaling moves ($\mathcal{L}$-moves).

From a current solution, $\mathcal{M}$-moves are performed by moving a task from one PE to another PE, or swapping two tasks between two PEs. Since evaluating all neighboring mappings can be extremely expensive for large task set, we use a probability of performing $\mathcal{M}$-moves ($P_{\mathcal{M}}$). Although it is possible to miss excellent solutions, the computational overhead is significantly reduced and randomness is introduced which can act as an anti-cycling mechanism. An $\mathcal{L}$-move is generated by randomly assigning operating mode for each task. For each neighboring mapping, a number of $\mathcal{L}$-moves (n$\mathcal{L}$) are added to the candidate set $\mathcal{C}$.

We evaluate all candidates in $\mathcal{C}$ (line 7). For selecting a new solution, we first attempt to select an improved solution with

Algorithm 1 MVFS

1: $\mathcal{S}^0 \leftarrow$ InitialSolution(Γ', $\mathcal{N}$); $\mathcal{S}^{now} \leftarrow \mathcal{S}^0$
2: **while** *max_iterations* is not exhausted **do**
3: $\mathcal{C} \leftarrow \mathcal{M}$-moves($\mathcal{S}^{now}$, $P_{\mathcal{M}}$)
4: **for** each solution $\mathcal{S}_i^{new} \in \mathcal{C}$ **do**
5: $\mathcal{C} \leftarrow \mathcal{L}$-moves($\mathcal{S}_i^{new}$, n$\mathcal{L}$)
6: **for** each solution $\mathcal{S}_j^{new} \in \mathcal{C}$ **do**
7: Calculate $\Delta_j = cost(\mathcal{S}^{now}) - cost(\mathcal{S}_j^{new})$
8: **end for**
9: **end for**
10: $\mathcal{S}^{now} \leftarrow$ SelectNewSolution(Δ_j, *length_tl*)
11: $\mathcal{S} \leftarrow$ SaveBestSolution($\mathcal{S}^{now}$)
12: **end while**
13: **return** $\mathcal{S}$

the largest improvement compared to current solution, as long as it is not on the tabu list or the tabu status can be *aspirated*[4]. If no such improved solution exists, we randomly select a non-improved solution to be new solution as long as it is on non-tabu status. Randomly accepting non-improving moves introduces the diversification, which forces the search into previously unexplored areas of the design space.

We record the last *length_tl* design transformations (both $\mathcal{M}$-moves and $\mathcal{L}$-moves) to the tabu list in order to prohibit reverse transformations. The selection of new solution and the maintenance of tabu list are done inside the SelectNewSolution function (line 10). Then, SaveBestSolution function (line 11) saves the new solution if it is the best solution so far. After *max_iterations* without an improvement, the algorithm stops and the best-found-solution $\mathcal{S}$ is reported.

A. Schedulability Analysis

To determine the schedulability of an implementation we use response-time analysis [21] to calculate the worst-case response time r_i of every task τ_i, which is compared to the deadline D_i. The basic analysis presented in [21] has been extended over the years. For example, the state-of-the-art analysis in [22] considers arbitrary arrival times and deadlines, offsets and synchronous inter-task communication (where a receiving task has to wait for the input of the sender task), and the analysis in [17] takes into account the time needed for the mode-switching overheads.

Our focus in this paper is exploring the trade-off between energy and reliability. Hence, we decide to use the basic analysis from [21]. We do, however, take communication into account to make sure that the mapping solutions do not create too much bus traffic. We consider a non-preemptive fixed-priority bus and during the design space exploration we mark as infeasible solutions which result in a bus utilization over 100%. The utilization is calculated from the bus speed, the size of the messages exchanged over the bus and period of the sender tasks.

The schedulability of an implementation $\mathcal{S}$ is captured using the "degree of schedulability", $r_{\mathcal{S}}$ over all tasks:

$$r_{\mathcal{S}} = \begin{cases} d_1 = \sum_i \max(0, r_i - D_i) & if \quad d_1 > 0 \\ d_2 = \sum_i (r_i - D_i) & if \quad d_1 = 0 \end{cases} \tag{10}$$

If the application is not schedulable, there exists at least one r_i greater than the deadline D_i, therefore the term d_1 of the function will be positive. In this case $r_{\mathcal{S}}$ is equal to d_1. However, if the application is schedulable, then each r_i is smaller than the corresponding deadline D_i. In the case $d_1 = 0$ and we use d_2 as the $r_{\mathcal{S}}$, as it is able to differentiate between two design alternatives, both leading to a schedulable application.

VII. Experimental Results

For the evaluation of our approach, we used ten synthetic benchmarks and five real-life case studies. In all benchmarks, a quarter of the tasks were considered critical and for each critical task we introduced one redundancy. We used three types of PEs described in Table 1. The minimum failure rate (when system runs at the maximum speed operating mode) of all PEs was set to $\lambda_0 = 10^{-6}$. We used a reference for all experiments, the system energy consumption $E^0_{\mathcal{S}}$ and reliability R^0_s obtained by our MVFS approach, but without performing $\mathcal{L}$-moves (i.e., no mode changes that all tasks are run with the highest frequency and voltage). The reliability goal R_g is set such that we accept a probability of failure which is ten times larger than R^0_s, i.e., $R_g = 1 - 10(1 - R^0_s)$. We tuned the MVFS parameters so that the results were as close as possible to optimal (i.e., no improvements were seen with a longer run-time).

TABLE 1 THREE TYPES OF *PEs*

Fast PE			Medium PE			Slow PE		
Freq. [MHz]	Volt. [V]	Power [W]	Freq. [MHz]	Volt. [V]	Power [W]	Freq. [MHz]	Volt. [V]	Power [W]
500	1.2	9.2	300	1.4	4.3	133	1.1	1.36
600	1.25	12	350	1.5	5.6	166	1.2	1.9
700	1.3	15.1	400	1.6	7.1	200	1.3	2.58
800	1.35	18.6	450	1.7	8.95	233	1.4	3.4
1000	1.4	25	500	1.8	11.4	266	1.5	4.4
$\alpha = -6.5, \beta = -0.039$			$\alpha = -8.9, \beta = -0.038$			$\alpha = -6.5, \beta = -0.039$		

In the first experiment we wanted to evaluate the quality of MVFS as the systems become larger. Table 2 presents the experimental setup details and the results. We used five synthetic benchmarks of 10 to 105 tasks (including replicated tasks) mapped on architectures with 2 to 6 different types of PEs. The results obtained with MVFS are presented in the last two columns. The increase in the failure probability θ was calculated using Eq. 8 and had to be smaller or equal to 10 in order to meet the reliability goal R_g. Alongside the MVFS results, we also present the results obtained with MVFS$^-$. This is an implementation which minimizes the energy, but without concern for reliability (the reliability constraint, the second term in Eq. 9, is removed). As we can see from Table 2, columns 5 and 6, minimizing the energy without considering reliability leads to a dramatic increase in the probability of failure, which increases more than 100 times in most cases. However, our MVFS approach is able to keep the system reliability R_s within the specified R_g, without a significant loss in energy savings (last column) compared to MVFS$^-$ (column 6).

In the second experiment we wanted to determine the ability of MVFS to find good quality solutions as the utilization of the system increases. The experimental setup and the results obtained are presented in Table 3. We used a synthetic benchmark with 25 tasks (20 tasks and 5 replicated tasks) mapped on a heterogeneous architecture with 3 different types of PEs. We varied the execution times resulting in five cases corresponding to various initial utilization (column 5), from 27.21% to 71.98%. More energy can be saved in the less utilized systems since more slack could be used for lowering operating voltage

TABLE 2 SYNTHETIC BENCHMARKS: DIFFERENT SYSTEM SIZES

Test Set	Numbers of PEs	Orig. Tasks	Repl. Tasks	MVFS$^-$ θ [times]	MVFS$^-$ Saved E [%]	MVFS θ [times]	MVFS Saved E [%]
1	2	8	2	166	28.23	10	24.64
2	4	31	8	112	28.56	10	25.28
3	4	42	11	137	30.47	10	26.04
4	6	63	16	104	25.92	10	21.92
5	6	84	21	57	22.78	10	20.57

[4]A tabu status can be *aspirated* when the solution has a better *cost* function than that of the current best-known solution, since it will not produce any cycling in the exploration of the design space.

TABLE 3 SYNTHETIC BENCHMARKS: VARYING INITIAL UTILIZATION

Test Set	Numbers of PEs	Orig. Tasks	Repl. Tasks	Initial Util. [%]	MVFS$^-$ θ [times]	MVFS$^-$ Saved E [%]	MVFS θ [times]	MVFS Saved E [%]
1	3	20	5	27.21	198	29.57	10	25.00
2	3	20	5	41.00	121	26.55	8	23.02
3	3	20	5	52.73	101	25.26	9	21.05
4	3	20	5	61.56	72	22.94	10	20.09
5	3	20	5	71.98	7	12.76	7	12.76

TABLE 4 REAL-LIFE BENCHMARKS

Benchmarks	Numbers of PEs	Orig. Tasks	Repl. Tasks	MVFS$^-$ θ [times]	MVFS$^-$ Saved E [%]	MVFS θ [times]	MVFS Saved E [%]
networking-cords	2	13	3	141	28.01	10	20.49
auto-indust-cords	4	24	6	77	22.68	10	17.87
telecom-cords	4	30	8	129	28.16	9	19.57
3 Apps together	6	67	17	64	15.26	10	13.86
Smart-phone	2	61	16	60	18.71	9	15.23

and frequency without missing the deadlines. As expected, using MVFS is especially important where the energy saving potential is greater, because reliability is significantly impaired without it.

Finally, we evaluated MVFS in real-life case studies. Five benchmarks were selected from the Embedded System Synthesis Benchmark Suite (E3S), version 0.9 [23], and SmartPhone Benchmarks [6][5]. The experimental setup details and the results obtained are presented in Table 4. As can be seen, this evaluation confirms the results obtained from the synthetic benchmarks. This means that by using MVFS we are able to eliminate the negative impact of energy minimization on reliability with minimal loss in energy savings.

VIII. CONCLUSION

In this paper we addressed the mapping, voltage and frequency scaling for fault-tolerant hard real-time applications mapped on distributed embedded systems where tasks and messages are scheduled using an event-driven scheduling policy. We captured the effect of voltage and frequency scaling on system reliability, and we showed that if the supply voltage and the operating frequency are lowered to reduce energy consumption, reliability is significantly reduced. That is why we proposed a Tabu Search-based approach that takes reliability into account when performing task mapping and operating mode assignment. As the experimental results show, our Tabu Search-based strategy is able to produce energy-efficient implementations which are both schedulable and fault-tolerant. By carefully deciding the mapping, operating voltage and frequency of each task, we showed that it is possible to eliminate the negative impact of energy minimization on reliability with minimal decrease in energy savings.

REFERENCES

[1] C. Constantinescu, "Trends and challenges in VLSI circuit reliability", *IEEE Micro*, 23, pp.14-19, 2003.

[2] H. Kopetz and G. Bauer, "The Time-Triggered Architecture", *Proceedings of the IEEE*, 91(1), pp. 112-126, 2003.

[3] A. Bertossi and L. Mancini, "Scheduling Algorithms for Fault-Tolerance in Hard-Real Time Systems", *Real Time Systems*, 7(3), pp. 229-256, 1994.

[4] A. Burns, R. Davis, S. Punnekkat, "Feasibility Analysis for Fault-Tolerant Real-Time Task Sets", *Euromicro Workshop on Real-Time Systems*, pp. 29-33, 1996.

[5] Y. Zhang and K. Chakrabarty, "Energy-Aware Adaptive Checkpointing in Embedded Real-Time Systems", *DATE Conf.*, pp. 918-923, 2003.

[6] M. T. Schmitz, B. M. Al-Hashimi, and P. Eles, *System-Level Design Techniques for Energy-Efficient Embedded Systems*, Springer, 2003.

[7] A. Ejlali, B. M. Al-Hashimi, M. Schmitz, P. Rosinger, and S. G. Miremadi, "Combined Time and Information Redundancy for SEU-Tolerance in Energy-Efficient Real-Time Systems", *Very Large Scale Integration (VLSI) Systems*, 14(4), pp. 323-335, 2006.

[8] R. Melhem, D. Mosse, and E. Elnozahy, "The interplay of power management and fault recovery in real-time systems", *IEEE Transactions on Computers*, 53(2), pp. 217-231, 2004.

[9] Y. Zhang and K. Chakrabarty, "Dynamic adaptation for fault tolerance and power management in embedded real-time systems", *ACM Transactions on Embedded Computing Systems*, vol. 3, pp. 336-360, 2004.

[10] D. Zhu, R. Melhem and D. Mossé, "The Effects of Energy Management on Reliability in Real-Time Embedded Systems", *Proc. of the International Conference on Computer Aided Design*, pp. 35-40, 2004.

[11] D. Zhu and H. Aydin, "Reliability-Aware Energy Management for Periodic Real-Time Tasks", *IEEE Transactions on Computers*, 58(10), pp. 1382 - 1397, 2009.

[12] Coskun, A. K.; Simunic Rosing, T.; Mihic, K.; De Micheli, G.; and Leblebici, Y. "Analysis and Optimization of MPSoC Reliability", *Journal of Low Power Electronics*, vol.2, no. 1, pp. 56-69, 2006.

[13] Paul Pop, Viacheslav Izosimov, Petru Eles, and Zebo Peng, "Design Optimization of Time- and Cost-Constrained Fault-Tolerant Embedded Systems With Checkpointing and Replication", *Very Large Scale Integration (VLSI) Systems, IEEE Transactions*, 17(3), pp.389-402, 2009.

[14] Paul Pop, Kre Harbo Poulsen, Viacheslav Izosimov, and Petru Eles, "Scheduling and voltage scaling for energy/reliability trade-offs in fault-tolerant time-triggered embedded systems", *Hardware/software codesign and system synthesis*, pp. 233-238, 2007.

[15] F. Gruian, "Hard real-time scheduling for low-energy using stochastic data and DVS processors", *International Symposium on Low Power Electronics and Design*, pp. 41-55, 2001.

[16] Pop, T. and Pop, P. and Eles, P. and Peng, Z. and Andrei, A., "Timing analysis of the FlexRay communication protocol", *Real-time systems*, pp. 205-235, 2008.

[17] E. Bini, G. Buttazzo and G. Lipari, "Minimizing CPU energy in real-time systems with discrete speed management", *ACM Transactions on Embedded Computing Systems*, vol.8, no.31, 2009.

[18] S. Manolache, P. Eles, and Z. Peng, "Buffer space optimisation with communication synthesis and traffic shaping fornocs", *Proceedings of Design, automation and test in Europe*, DATE '06, pp. 718-723, 2006.

[19] Barry W. Johnson, "Design and Analysis of Fault-Tolerant Digital Systems", Addison-Wesley, 1989.

[20] F. Glover. "Tabu Search", *ORSA Journal on Computing*, 1(3), pp. 109-206, 1989.

[21] G. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*, Springer, 2005.

[22] J. C. Palencia and M. G. Harbo, "Schedulability Analysis for Tasks with Static and Dynamic Offsets", *Proceedings of the IEEE Real-Time Systems Symposium*, p. 26, 1998.

[23] Homepage: http://ziyang.eecs.umich.edu/dickrp/e3s, *Embedded System Synthesis Benchmark Suite(E3S)*, 2009.

[5] We only used the applications of MP3 decoder, GSM decoder, JPEG encoder and JPEG decoder.