New Pack Oriented Solutions for Energy-Aware Feasible Adaptive Real-Time Systems

Aymen Gammoudi[1,2(✉)], Adel Benzina[1,2], Mohamed Khalgui[1], and Daniel Chillet[3]

[1] LISI Laboratory, INSAT, University of Carthage, Tunis, Tunisia
{aymen.gammoudi1,benzina.adel,khalgui.mohamed}@gmail.com
[2] Tunisia Polytechnic School, University of Carthage, Tunis, Tunisia
[3] IRISA Laboratory, ENSSAT, University of Rennes 1, Rennes, France
daniel.chillet@irisa.fr

Abstract. This paper addresses the management of tasks execution for real-time reconfigurable systems powered by battery. In this context, one of major problem concerns the management of battery life between two different recharges. For this type of systems, a reconfiguration scenario means the addition, removal or update of tasks in order to manage the whole system at the occurrence of hardware/software faults, or also to improve its performance at run-time. When such a scenario is applied, the system risks a fatal increase in energy consumption, a violation of real time constraints or a memory saturation. To prevent this type of problems during the execution, a new scheduling strategy is necessary. Our proposal is based on the definition of packs of tasks and the management of different parameters of these packs. For each reconfiguration scenario, modifications will be performed on packs/tasks parameters in order to respect the memory, real-time and energy constraints.

Keywords: Embedded system · Reconfiguration · Real-time and low-power scheduling · OS software optimization · Software analysis

1 Introduction

Nowadays, reconfigurable real-time embedded systems are found in diverse application areas including; avionics, automotive electronics, telecommunications, sensor networks, and consumer electronics. In all of these areas, there is rapid technological progress, yet, energy concerns are still the bottleneck. The minimization of energy consumption is an important criterion for development of real-time embedded systems due to limitations in the capacity of their batteries; in addition battery life can be extended by reducing power consumption [11]. The new generation of real-time embedded systems is addressing new criteria such as flexibility and agility [5]. For these reasons, there is a need to define strategy/methodology in embedded software engineering and dynamic reconfigurable embedded technologies as an independent discipline. Concerning the reconfiguration, two policies are defined in the literature: static and dynamic

© Springer International Publishing Switzerland 2015
H. Fujita and G. Guizzi (Eds.): SoMeT 2015, CCIS 532, pp. 73–86, 2015.
DOI: 10.1007/978-3-319-22689-7_6

reconfigurations. Static reconfigurations are applied off-line to apply changes before the system cold start for a required functional safety [4], whereas dynamic reconfigurations are applied during the execution (on-line) of the application, i.e. at run-time. Dynamic reconfiguration can be manually applied by users [9] or automatically applied by Intelligent Agents [6].

We consider here dynamic reconfiguration and we assume that the system executes n real-time tasks initially feasible towards real-time scheduling. We also assume that the system battery is recharged periodically with a recharge period RP. The general goal of this paper is to ensure that any reconfiguration scenario changing the implementation of the embedded system does not violate real-time constraints and does not result in fatal energy over consumption or in memory saturation. Several research studies [11,12] have focused on the modification of periods or WCETs of tasks in order to decrease the processor utilization. These studies are interesting, but the authors are not interested in the computation cost of the new parameters since they perform heavy calculations after any reconfiguration scenario. Moreover, non-logical values of parameters that do not meet user requirements can be generated. Finally, they do not consider the memory overflow problem after any reconfiguration scenario. Unlike [11,12], we are interested in this paper in deterministic solutions to control the computation cost of parameters that should be realistic while controlling energy and memory constraints.

As a major contribution of this paper, to respect the memory, real-time and energy constraints, a new strategy is defined where after each reconfiguration scenario, suitable and acceptable modifications are performed on parameters of tasks by using well-defined formulas. After each reconfiguration scenario, [12] proposes some solutions to be applied in an arbitrary manner in order to minimize the energy consumption, but it is hard to implement the approach proposed in an embedded platform because it is too complex to be executed on-line. In this paper, we propose a methodological strategy that solves this drawback. According to system and battery state, this strategy proposes quantitative techniques to modify periods, reduce execution times of tasks or remove some of them to ensure real-time feasability, avoiding memory overflow and ensuring a rational use of remaining energy until next recharge.

This paper is organized as follows: Sect. 2 presents the state of the art of reconfigurable embedded systems, low power consumption and real-time scheduling. The third section explains the formalization and a case study. In Sect. 4 we present the different proposed solutions. We evaluate this solution in Sect. 5. Finally, we conclude and present our future works in Sect. 6.

2 State of the Art

Several papers in recent years considered real-time and low-power scheduling policies [8,12,13].

2.1 Reconfiguration of Embedded Systems

Nowadays, a fair amount of research has been done to develop reconfigurable embedded systems. In [11] Wang et al. propose a study for feasible low power dynamic reconfigurations of real-time systems where additions and removals of real-time tasks are applied at run-time. They aim to minimize the energy consumption after any reconfiguration scenario. The research in [3] proposes an agent-based reconfiguration approach to save the whole system when faults occur at run-time. [1] develops an ontology-based agent to perform system reconfigurations that adapt changes in requirements and also in environment. They are interested in studying reconfigurations of control systems when hardware faults occur at run-time. Although these rich and useful contributions provide interesting results, no one is reported to address the problem of dynamic reconfigurations under memory, real-time feasability and energy constraints simultaneously.

2.2 Real-Time Scheduling

Real-time scheduling has been extensively studied in the last three decades [2]. These studies propose several Feasibility Conditions for the dimensioning of real-time systems. These conditions are defined to enable a designer to grant that timeliness constraints associated with an application are always met for all possible configurations. In this paper, Two main classical scheduling are generally used in real-time embedded systems: RM and EDF. Firstly, EDF is a dynamic scheduling algorithm used in real-time operating systems. EDF is an optimal scheduling algorithm on preemptive uniprocessors, in the following sense: if a collection of independent jobs (each one characterized by an arrival time, an execution requirement, and a deadline) can be scheduled (by any algorithm) such that all the jobs complete by their deadlines, then the EDF will schedule this collection of jobs such that all of them complete by their deadlines. On the other hand, if a set of tasks is not schedulable under EDF, then no other scheduling algorithm can feasibly schedule this task set. So, compared to fixed priority scheduling techniques like Rate-Monotonic scheduling, EDF can guarantee all the deadlines in the system at higher loading. When scheduling periodic processes that have deadlines equal to their periods, and when the context switching time is negligible, EDF has a utilization bound of 100 %. The necessary and sufficient condition for the schedulability of the tasks follows that for a given set of n tasks, τ_1, τ_2,..., τ_n with time periods T_1, T_2, ..., T_n, and computation times (worst case execution time, WCET) of C_1, C_2, ..., C_n assuming that $T_i = D_i$ (period equals to deadline) for each task, the deadline driven schedule algorithm is feasible if and only if $U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$, [7]. Secondly, RM is an on-line preemptive static priority scheduling strategy for periodic and independent tasks assuming that $T_i = D_i$ (period equals to deadline) for each task τ_i. The idea is to determine fixed priorities by task frequencies: tasks with higher rates (shorter periods) are assigned with a higher priority. The necessary and sufficient condition for the schedulability of the tasks follows that for a given set of n tasks, τ_1, τ_2,..., τ_n with time periods T_1, T_2, ..., T_n, and computation

times of C_1, C_2, ..., C_n, the deadline driven scheduling algorithm is feasible if $U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{\frac{1}{n}} - 1)$. In our current work, to ensure the availability of energy after each reconfiguration scenario, we focus on adapting task parameters T_i or C_i. We propose to apply dynamic policy EDF when the performance of the system is well, otherwise the static policy RM with limited characteristics. We use as a notation for this real-time feasibility condition: $U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq \alpha_{policy}$,where $\alpha_{policy} = 1$ for EDF scheduling and $\alpha_{policy} = n(2^{\frac{1}{n}} - 1)$ for RM scheduling.

2.3 Low-Power Scheduling

Power reduction techniques can be classified into two categories: static [10] and dynamic. In [11], the power consumption P is proportional to the processor utilization U. If the processor utilization is minimized, then the power consumption is automatically minimized: $P = k.U^2$. Based on the previous formula, Wang et al. in [11,12], present a simple run-time strategy that reduces the energy consumption. They propose to modify the tasks period T_i, assigning a single value to all tasks which is not reasonable in practice [11]. Another solution proposed is to reduce WCETs (C_i) assigning a single value to all tasks which is not reasonable in practice [11]. The formulas proposed in [11,12] are simple with soft calculation, but the main disadvantage is that it is not acceptable for a real-time system to change the period of tasks more than a certain limit according to user requirement. Moreover if tasks have very diverse periods T_i, tasks that have small periods will be too much affected if they will be aligned with tasks that have large periods. The system overall will look like a synchronous system driven by the slowest task. [11,12] propose the same principle to modify WCETs.

To address this problem, we propose to group the tasks that have "similar" periods in packs by assigning a unique period to all tasks of a pack. This idea is formalized in Sect. 3. To reduce energy consumption, [11,12] propose to remove some tasks when the system lacks energy without any reasonable strategy. This is a suitable approach, but if we have to remove a task, we shall preserve critical real-time tasks and remove less important ones first. The complete formulation of this strategy is given in the next sections of this paper. To verify the system's behavior, we use the real-time simulator Cheddar.

3 Problem Formalization for Reconfigurable Real-Time Systems

This section defines a formalization of the problems exposed above illustrated by different case studies.

3.1 Task Model

We assume in this paper that a real-time embedded system *Sys* is composed of a set of tasks that should meet real-time constraints defined in user requirements:

$Sys = \{\tau_1, \tau_2, ..., \tau_n\}$. Like in [7], Each task τ_i of Sys is defined by (i) a release time R_i, (ii) its worst case execution time (WCETs) C_i, (iii) a period T_i, (iv) a maximum period T_imax, (v) a deadline D_i, (vi) an importance factor I_i and vii) a memory footprint MF_i. Let us explain some parameters: (a) T_imax: is the maximum period I_i can not exceed according to system specification, (b) I_i: is an Integer variable (between 0 and 15), called "importance factor" according to user functional requirements. If a task has a very high value I_i, then the task is less important, else the task is paramount. In case the embedded system has a low energy, so it should to remove some tasks according to their importance factor. Tasks that have $I_i = 0$ are considered critical real-time tasks that can not admit change in their parameters. Finally (c) MF_i: the memory space used by the task τ_i. In this paper, we assume that $T_i = D_i$, then each task τ_i will be described by: $\tau_i = \{R_i, C_i, T_i, T_imax, I_i, MF_i\}$. After each reconfiguration scenario, it is necessary to check the feasibility of real-time scheduling by verifying the equation: $\sum_{i=1}^{n} \frac{C_i}{T_i} \leq \alpha_{policy}$.

3.2 Energy Model

We consider that a real-time embedded system is periodically fully recharged, the energy model is characterized by (i) a quantity of energy available at full recharge E_{max}, (ii) an energy available at time $t : \Delta E(t)$, (iii) a recharge period RP and (iv) a time remaining until the next recharge Δt. As define in Sect. 2.3, the power consumption P is proportional to the processor utilization U. So, $P \alpha U$, it means $P = k.U^2$. Then the power consumption is calculated by:

$$P = k.U^2 = k.(\sum_{i=1}^{n} \frac{C_i}{T_i})^2 \tag{1}$$

We assume in this paper that $k = 1$. To ensure that the system will run correctly until the next recharge, it is necessary that at time t:

$$P(t).\Delta t \leq \Delta E(t) \tag{2}$$

$P(t)$ is the power consumption at t, that means the power consumption $P(t) \leq \frac{\Delta E(t)}{\Delta t}$. We define $P_{limit}(t) = \frac{\Delta E(t)}{\Delta t}$. After each reconfiguration scenario, we have to ensure that: $P(t) \leq P_{limit}(t)$: This is the Energy Constraint.

3.3 Memory Model

We suppose that the memory model in a real-time embedded system is characterized by (i) a memory size MS and (ii) an Available memory at time t, $AM(t)$. Each task occupies at run-time MF_i amount of memory. After each reconfiguration scenario, we must ensure that: $\sum_{i=1}^{n} MF_i < AM(t)$. This is the Memory Constraint.

3.4 Reconfiguration Problem

We suppose that Sys is initially composed of n tasks at t_1, $Sys(t_1) = \{\tau_1, \tau_2, ..., \tau_n\}$, we also suppose that $Sys(t_1)$ is feasible. We assume in the following that the system Sys is dynamically reconfigured at run-time such that its new implementation is $Sys(t_2) = \{\tau_1, \tau_2, ..., \tau_n, \tau_{n+1}, ..., \tau_m\}$. The subset $\{\tau_{n+1}, ..., \tau_m\}$ is added to the initial implementation $\{\tau_1, \tau_2, ..., \tau_n\}$. To ensure that the system will run correctly after this reconfiguration scenario, at time t, it is necessary to check whether the new configuration respects these three constraints:

1. Real-time scheduling feasability constraint, denoted FeasibleC, must verify

$$\sum_{i=1}^{m} \frac{C_i}{T_i} \leq \alpha_{policy} \tag{3}$$

2. Energy constraint, denoted EnergyC, must verify

$$P(t) \leq P_{limit}(t) \tag{4}$$

3. Memory constraint, denoted MemoryC, must verify

$$\sum_{i=1}^{m} MF_i < AM(t) \tag{5}$$

After each reconfiguration scenario, one or more of these constraints can be violated, we have to find the suitable solution to each problem.

3.5 Case Study Problems Illustrations

We present in this section a case study that can show the different problems. We use this notation to represent certain tasks $\tau_i = \{C_i, T_i, T_i max, I_i\}$. Let us assume that the system supports the following tasks: $\tau_1 = \{4, 40, 90, 1\}$, $\tau_2 = \{6,$ 15, 50, 1$\}$, $\tau_3 = \{3, 29, 80, 2\}$ and $\tau_4 = \{4, 40, 70, 4\}$. It is assumed that we use the EDF scheduling ($\alpha_{policy} = 1$). We verify the system feasibility condition:

$$U = \sum_{i=1}^{4} \frac{C_i}{T_i} = 0.7034 \leq 1 \tag{6}$$

then the system is feasible. We suppose that at this time t, $P_{limit}(t) = 1.2\,\mathrm{W}$. It is assumed also that $k = 1$, then we calculate the power consumption at t:

$$P(t) = k * U^2 = 1 * (\sum_{i=1}^{4} \frac{C_i}{T_i})^2 = 0.4947\,\mathrm{W} \tag{7}$$

$P(t)$ is less than $P_{limit}(t)$, then the Energy constraint is respected.

We suppose now that after a certain execution time, a first reconfiguration is performed. For this reconfiguration, two tasks $\tau_5 = \{5, 20, 50, 1\}$ and

$\tau_6 = \{6, 25, 50, 5\}$ are added. Due to this reconfiguration, we must verify if the system respects the feasibility condition. We then compute U as,

$$U = \sum_{i=1}^{6} \frac{C_i}{T_i} = 1.193 > 1 \tag{8}$$

Because the value of U is greater than 1, the system is no more feasible after the reconfiguration. Furthermore, we must also verify the Energy constraint at this time t:

$$P(t) = k * U^2 = 1 * (\sum_{i=1}^{6} \frac{C_i}{T_i})^2 = 1.423\,\mathrm{W} \tag{9}$$

As $P(t)$ is higher than $P_{limit}(t)$, then the Energy constraint is not respected.
So, for this reconfiguration scenario, two constraints are then violated:

- *Problem1: Real-Time Constraint is violated.*
- *Problem2: Energy Constraint is violated.*

4 Solutions for Feasible Reconfigurable Real-Time Systems

In this section, we present the different solutions that we propose to extend [11,12]. These solutions are mainly based on the modification of the periods (T_i) or the WCETs (C_i) of tasks in order to ensure that the system will run correctly until the next battery recharge after each reconfiguration scenario and to satisfy the real-time feasibility and memory constraints. In fact if we take Eqs. 1 and 3, we can see that T_i and C_i are parameters that can be adapted to apply a new configuration that respects the energy and feasibility constraints. To ensure that the system is feasible, Wang et al. in [11,12] propose an approche to modify the tasks period T_i assigning the same value to all tasks [11]. Another solution proposed is to reduce WCET while assigning also the same value to all tasks [11]. As stated in Sect. 2.3, this approach presents two main drawbacks and cannot be applied in practice. In this paper, we propose to group the tasks that have "similar" periods in several Packs, denoted Pk, by assigning a unique new period T^{New} to all tasks of the first pack Pk_1. Moreover all new periods affected to pack Pk_j are multiples of T^{New}, the period affected to tasks belonging to pack Pk_1. We have only to compute the suitable T^{New}. This solution controls the complexity of the problem.

4.1 Pack Model

Let us note that each time a new period T^{New} is affected to a task that has originally a period T_i, the cost is a delay penalty for this task of T^{New} - T_i. This is applicable for tasks of Pack Pk_1. For other packs Pk_j the period is j* T^{New}. So the cost for each task of Pk_j is: (T^{New}-(T_i mod T^{New})) mod T^{New}. The total

cost for the approach is the sum of all these costs. We need to seek the value T^{New} that minimizes the cost of the new solution for the whole system:

$$\sum_{i=1}^{m}((T^{New} - (T_i \bmod T^{New})) \bmod T^{New}) \; is \; minimal, \; with \; T^{New} \geq Min(T_i) \tag{10}$$

***Running Example 1*: (Case study).** We have 6 tasks. According to Eq. 10 we seek a value of T^{New} that leads to a minimum cost. Possible values of T^{New} range from 15 s to 40 s. We found that $T^{New} = 26$ s is the optimal solution.

The same approach is applied when WCET is modified. It is necessary to seek C^{New} such that:

$$\sum_{i=1}^{m}((C^{New} - (C_i \bmod C^{New})) \bmod C^{New}) \, is \, minimal, \, with \, C^{New} \geq Min(C_i) \tag{11}$$

***Running Example 2*: (Case study).** We have 6 tasks. According to Eq. 11 we seek C^{New}, then we start the calculation of costs, with $C^{New} = 3$ until $C^{New} = 6$. So $C^{New} = 3$ is minimal with a cost equals to 5.

This approach leads to 2 solutions to make the system feasible denoted (T_{RT}^{New} and C_{RT}^{New}) and 2 solutions to be sure that the system respects the energy constraint denoted (T_{Eg}^{New} and C_{Eg}^{New}). We present the proposed solutions for each problem apart.

4.2 Solution A: Modification of Periods Under Real-Time Scheduling Constraint:

Proposition 1. The extended T_i of the task τ_i is multiple of T_{RT}^{New}:

$$T_{RT}^{New} = \left\lceil \frac{\sum_{Pk_1} C_i + \sum_{Pk_2} \frac{C_i}{2} + ... + \sum_{Pk_j} \frac{C_i}{j}}{\alpha_{policy}} \right\rceil \tag{12}$$

Proof. In order to respect the real-time scheduling constraint according to a scheduling policy "α_{policy}": $\sum_{i=1}^{m} \frac{C_i}{T_i} \leq \alpha_{policy}$. We assign each task to its Pack Pk_j according to its period T_i, Then:

$$\sum_{Pk_1} \frac{C_i}{T} + \sum_{Pk_2} \frac{C_i}{2.T} + ... + \sum_{Pk_j} \frac{C_i}{j.T} \leq \alpha_{policy}$$

So,

$$\frac{1}{T}.(\sum_{Pk_1} C_i + \sum_{Pk_2} \frac{C_i}{2} + ... + \sum_{Pk_j} \frac{C_i}{j}) \leq \alpha_{policy}$$

Then,

$$T_{RT}^{New} = \frac{\sum_{Pk_1} C_i + \sum_{Pk_2} \frac{C_i}{2} + ... + \sum_{Pk_j} \frac{C_i}{j}}{\alpha_{policy}}$$

Since the periods are integer:

$$T_{RT}^{New} = \left\lfloor \frac{\sum_{Pk_1} C_i + \sum_{Pk_2} \frac{C_i}{2} + ... + \sum_{Pk_j} \frac{C_i}{j}}{\alpha_{policy}} \right\rfloor$$

Now, we assign T_{RT}^{New} to tasks of Pk_1, $2 * T_{RT}^{New}$ to tasks of Pk_2, ..., $j * T_{RT}^{New}$ to tasks of Pk_j. After the modification of the periods, the processor utilization of tasks is reduced, and can satisfy the real-time scheduling.

Running Example 3: **Problem 1 (Case study).** According to Eq. 10, the optimal value of T^{New} is 15 s. Then we have three Packs: Pk_1 groups tasks that have periods between 1 and 15, Pk_2 groups tasks that have periods between 16 and 30 and Pk_3 groups tasks that have periods between 31 and 45. The new period T_{RT}^{New} that satisfies the real-time constraint is equal to 16 according to Eq. 12. Then, U is equal to $0.9791 \leq 1$. It is obvious that the real-time constraint is respected after applying a reconfiguration scenario.

4.3 Solution B: Modification of WCETs Under Real-Time Scheduling Constraint:

Proposition 2. The extended WCET C_i of task τ_i is multiple of C_{RT}^{New}:

$$C_{RT}^{New} = \left\lfloor \frac{\alpha_{policy}}{\sum_{Pk_1} \frac{1}{T_i} + \sum_{Pk_2} \frac{2}{T_i} + ... + \sum_{Pk_j} \frac{j}{T_i}} \right\rfloor \tag{13}$$

Proof. We followed the same used technique to calculate the new WCETs. After we reconfigure the WCETs, we should get $\sum_{i=1}^{m} \frac{C_i}{T_i} \leq \alpha_{policy}$. We assign each task to its Pack Pk_j according to its WCETs C_i, Then:

$$\sum_{Pk_1} \frac{C}{T_i} + \sum_{Pk_2} \frac{2.C}{T_i} + ... + \sum_{Pk_j} \frac{j.C}{T_i} \leq \alpha_{policy}$$

So,

$$C. \left(\sum_{Pk_1} \frac{1}{T_i} + \sum_{Pk_2} \frac{2}{T_i} + ... + \sum_{Pk_j} \frac{j}{T_i} \right) \leq \alpha_{policy}$$

Then,

$$C_{RT}^{New} = \left\lfloor \frac{\alpha_{policy}}{\sum_{Pk_1} \frac{1}{T_i} + \sum_{Pk_2} \frac{2}{T_i} + ... + \sum_{Pk_j} \frac{j}{T_i}} \right\rfloor$$

We assign C_{RT}^{New} to tasks of Pk_1, $2 * C_{RT}^{New}$ to tasks of Pk_2, ..., $j * C_{RT}^{New}$ to tasks of Pk_j. After the modification of the WCETs, the processor utilization of tasks is reduced, and can satisfy the real-time scheduling.

***Running Example 4*: Problem 1 (Case study).** According to Eq. 11, the optimal value of C^{New} is 3 s. Then we have two Packs: Pk_1 groups tasks that have WCETs between 1 and 3 and Pk_2 groups tasks that have WCETs between 4 and 6. The new WCET C_{RT}^{New} satisfies the real-time constraint is equal to 2 according to Eq. 13. Then, U is equal to $0.895 \leq 1$. It is obvious that the real-time constraint is respected after applying a reconfiguration scenario.

4.4 Solution C: Modification of Periods Under Energy Constraint:

Proposition 3. The extended T_i of task τ_i is multiple of T_{Eg}^{New}

$$T_{Eg}^{New} = \left\lceil \frac{\sum_{Pk_1} C_i + \sum_{Pk_2} \frac{C_i}{2} + ... + \sum_{Pk_j} \frac{C_i}{j}}{\sqrt{\frac{P_{limit}(t)}{k}}} \right\rceil \tag{14}$$

Proof. It is necessary that the current power $P(t) = k.U^2$ should be less than the critical power P_{limit}, with $P_{limit(t)} = \frac{\Delta E(t)}{\Delta t}$, then we should get $k.U^2 \leq P_{limit}(t)$ U $\leq \sqrt{\frac{P_{limit}(t)}{k}}$. So, $\sum_{i=1}^{m} \frac{C_i}{T_i} \leq \sqrt{\frac{P_{limit}(t)}{k}}$. We assign each task to its Pack Pk_j according to its period, Then:

$$\sum_{Pk_1} \frac{C_i}{T} + \sum_{Pk_2} \frac{C_i}{2.T} + ... + \sum_{Pk_j} \frac{C_i}{j.T} \leq \sqrt{\frac{P_{limit}(t)}{k}}$$

$$\frac{1}{T}.(\sum_{Pk_1} C_i + \sum_{Pk_2} \frac{C_i}{2} + ... + \sum_{Pk_j} \frac{C_i}{j}) \leq \sqrt{\frac{P_{limit}(t)}{k}}$$

So,

$$T_{Eg}^{New} = \left\lceil \frac{(\sum_{Pk_1} C_i + \sum_{Pk_2} \frac{C_i}{2} + ... + \sum_{Pk_j} \frac{C_i}{j})}{\sqrt{\frac{P_{limit}(t)}{k}}} \right\rceil$$

We assign T_{Eg}^{New} to tasks of Pk_1, $2 * T_{Eg}^{New}$ to tasks of Pk_2, ..., $j * T_{Eg}^{New}$ to tasks of Pk_j. After the modification of the periods, the processor utilization of tasks is reduced which can respect the energy constraint.

***Running Example 5*: Problem 2 (Case study).** According to Eq. 10, the optimal value of T^{New} is 15 s. Then we have three Packs: Pk_1 groups tasks that have periods between 1 and 15, Pk_2 groups tasks that have periods between 16 and 30 and Pk_3 groups tasks that have periods between 31 and 45. The new period T_{Eg}^{New} that satisfies the energy constraint remains equal to 15 according to Eq. 14. Then, U is equal to 1.0392. So, $P = k.U^2 = 1 * U^2 = 1.08\,\text{W} \leq 1.2\,\text{W}$. It is obvious that the energy constraint is respected after applying a reconfiguration scenario.

Note: In this running example the real-time constraint is violated because $U = 1.0392 > 1$, we should seek another period by using the solution A and choose the maximum to satisfy the two constraints.

4.5 Solution D: Modification of Periods Under Energy Constraint:

Proposition 4. The extended WCET C_i of task τ_i is multiple of C_{Eg}^{New}:

$$C_{Eg}^{New} = \left\lfloor \frac{\sqrt{\frac{P_{limit}(t)}{k}}}{(\sum_{Pk_1} \frac{1}{T_i} + \sum_{Pk_2} \frac{2}{T_i} + ... + \sum_{Pk_j} \frac{j}{T_i})} \right\rfloor \tag{15}$$

Proof. It is necessary that the current power $P(t) = k.U^2$ should be less than the critical power P_{limit}, with $P_{limit(t)}=\frac{\Delta E(t)}{\Delta t}$, then we should get $k.U^2 \leq P_{limit}(t)$ U $\leq \sqrt{\frac{P_{limit}(t)}{k}}$. So, $\sum_{i=1}^{m} \frac{C_i}{T_i} \leq \sqrt{\frac{P_{limit}(t)}{k}}$.

We assign each task to its Pack Pk_j according to its WCETs C_i, Then:

$$\sum_{Pk_1} \frac{C}{T_i} + \sum_{Pk_2} \frac{2.C}{T_i} + ... + \sum_{Pk_j} \frac{j.C}{T_i} \leq \sqrt{\frac{P_{limit}}{k}}$$

$$C.\left(\sum_{Pk_1} \frac{1}{T_i} + \sum_{Pk_2} \frac{2}{T_i} + ... + \sum_{Pk_j} \frac{j}{T_i} \right) \leq \sqrt{\frac{P_{limit}}{k}}$$

So,

$$C_{Eg}^{New} = \left\lfloor \frac{\sqrt{\frac{Plimit}{k}}}{\sum_{Pk_1} \frac{1}{T_i} + \sum_{Pk_2} \frac{2}{T_i} + ... + \sum_{Pk_j} \frac{j}{T_i}} \right\rfloor$$

We assign C_{Eg}^{New} to tasks of Pk_1, $2 * C_{Eg}^{New}$ to tasks of Pk_2, ..., $j * C_{Eg}^{New}$ to tasks of Pk_j. After the modification of the WCETs, the processor utilization of tasks is reduced, and can respect the energy constraint.

Running Example 6: **Problem 2 (Case study).** According to Eq. 11, the optimal value of C^{New} is 3 s. Then we have two Packs: Pk_1 groups tasks that have WCETs between 1 and 3 and Pk_2 groups tasks that have WCETs between 4 and 6. The new WCET C_{Eg}^{New} that satisfies the energy constraint is equal to 2 according to Eq. 15. Then, U is equal to 0.894, then $Pk = k.U^2 = 1 * U^2 = 0.799 \leq 1.2$ W. It is obvious that the energy constraint is respected after applying a reconfiguration scenario.

Note: If the real-time constraint is violated ($U > 1$), we should seek another WCET by using the solution B and choose the minimum to satisfy the two constraints.

4.6 Solution E: Removal of Tasks

This solution proposes the removal of less important tasks according to the importance factor I_i in order to minimize the energy consumption after any reconfiguration scenario of an embedded system that affects the energy constraint.

4.7 New Deterministic Solution for Real-Time and Low-Power Scheduling of Reconfigurable Embedded Systems Under Memory Constraints

We can implement our approach by this algorithm with complexity $O(n)$. We use the following functions: $ProcessorUtilization(k)$: It is a function that returns the processor utilization value when it runs with a given tasks parameters denoted k. $Execution(k)$: System execution by applying k, $Execution()$: Regular execution, $Max(a, b)$: It is a function that returns the maximum between a and b, $Min(a, b)$: It is a function that returns the minimum between a and b.

Algorithm 1. Decision Strategy

```
while Reconfiguration do
  if (!MemoryC) then
    Execution(SolutionE)
  else if (FeasibleC) AND (EnergyC) then
    Execution()
  else if (!FeasibleC) AND (EnergyC) then
    if (ProcessorUtilization(SolutionA) < ProcessorUtilization(SolutionB)) then
      Execution(SolutionA)
    else
      Execution(SolutionB)
    end if
  else if (FeasibleC) AND (!EnergyC) then
    if (ProcessorUtilization(SolutionC) < ProcessorUtilization(SolutionD)) then
      Execution(SolutionC)
    else
      Execution(SolutionD)
    end if
  else
    if (ProcessorUtilization(Max{SolutionA, SolutionC}) <
    ProcessorUtilization(Min{SolutionB, SolutionD}) then
      Execution(Max{SolutionA, SolutionC})
    else
      Execution(Min{SolutionB, SolutionD})
    end if
  end if
end while
```

5 Evaluation of Performance

To evaluate the current paper's contribution to the related works (RW) in [11,12]. We assume a case of a system composed of 100 tasks that can be reconfigured at run-time under memory and energy constraints. For this purpose we adopted the same set of tasks used in [11] to evaluate this algorithm. We calculate the

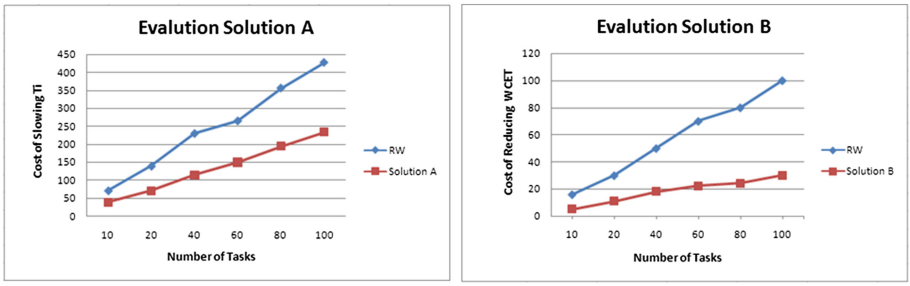

Fig. 1. Cost of modication of periods T_i (Solution A) and WCETs C_i (Solution B).

cost of our solutions compared to the proposed solution in [11,12]. The cost of a solution is the total delay introduced to periods T_i or to WCETs C_i as explained in Sect. 4.1. In Fig. 1, we show a comparison with RW when we apply Solution A (Fig. 1 left side) and Solution B (Fig. 1 right side). For each reconfiguration scenario, modifications will be performed on packs/tasks parameters in order to respect the memory, real-time and energy constraints. Thanks to this concept of packs, we can notice that our solution is less costly in both cases A and B than RW. Moreover, our solutions are implemented by an algorithm with complexity $O(n)$, but the complexity of the algorithm of these related works [12] is $O(n^2)$ (two nested for-loops).

More evaluation work has to be developed through simulation:

- The processor utilization while considering several random distributions of a set of tasks and comparison with [11,12].
- The total delay (Solution cost) also with randomly distributed set of tasks.

6 Conclusion

This paper is interested in reconfigurable real-time embedded systems when the battery recharges are done periodically. Our study concerns specifically the influence of the reconfiguration on memory, energy and real-time feasibility constraints. We propose a new strategy that ensures a low-cost feasible real-time and low-power reconfiguration of embedded systems while meeting memory limits. Thanks to the estimation of available energy after any reconfiguration, the system is temoporally configured to run the embedded tasks with low-cost computation. In addition to the control of memory, our solution is more realistic since it generates logical values of real-time parameters to be assigned to different packs. This original contribution is more useful than related works in [11,12] since it is applicable in practice. In our future works, we will be interested in the implementation of the paper's contribution that will be evaluated by assuming real case studies.

References

1. Al-Safi, Y., Vyatkin, V.: An ontology-based reconfiguration agent for intelligent mechatronic systems. In: Mařík, V., Vyatkin, V., Colombo, A.W. (eds.) HoloMAS 2007. LNCS (LNAI), vol. 4659, pp. 114–126. Springer, Heidelberg (2007)
2. Baruah, S., Goossens, J.: Scheduling real-time tasks: algorithms and complexity. In: Leung, J.Y.T. (ed.) Handbook of Scheduling: Algorithms Models and Performance Analysis. CRC Press, Boca Raton (2003)
3. William Brennan, R., Fletcher, M., Norrie, D.H.: A holonic approach to reconfiguring real-time distributed control systems. In: Mařík, V., Štěpánková, O., Krautwurmová, H., Luck, M. (eds.) ACAI 2001, EASSS 2001, AEMAS 2001, and HoloMAS 2001. LNCS (LNAI), vol. 2322, pp. 323–335. Springer, Heidelberg (2002)
4. Angelov, C., Sierszecki, K., Marian, N.: Design models for reusable and reconfigurable state machines. In: Yang, L.T., Amamiya, M., Liu, Z., Guo, M., Rammig, F.J. (eds.) EUC 2005. LNCS, vol. 3824, pp. 152–163. Springer, Heidelberg (2005)
5. Gharsellaoui, H., Ben Ahmed, S.: Real-time reconfigurable scheduling of sporadic tasks. In: Cordeiro, J., Van Sinderen, M. (eds.) ICSOFT 2013. CCIS, vol. 457, pp. 24–39. Springer, Heidelberg (2014)
6. Khalgui, M., Mosbahi, O., Li, Z., Hanisch, H.: Reconfigurable multi-agent embedded control systems: from modelling to implementation. IEEE Trans. Comput. **60**(4), 538–551 (2010)
7. Liu, C., Layland, J.: Scheduling algorithms for multiprogramming in a hard-real-time environment. J. ACM **20**(1), 46–61 (1973)
8. Quan, G., Hu, X.: Minimum energy fixed-priority scheduling for variable voltage processors. IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. **23**(9), 1062–1071 (2003)
9. Rooker, M.N., Sünder, C., Strasser, T., Zoitl, A., Hummer, O., Ebenhofer, G.: Zero downtime reconfiguration of distributed automation systems: the ϵCEDAC approach. In: Mařík, V., Vyatkin, V., Colombo, A.W. (eds.) HoloMAS 2007. LNCS (LNAI), vol. 4659, pp. 326–337. Springer, Heidelberg (2007)
10. Shin, Y., Choi, K.: Power conscious fixed priority scheduling for hard real-time systems. In: 1999 36th Proceedings of Design Automation Conference, pp. 134–139 (1999)
11. Wang, X., Khalgui, M., Li, Z.: Dynamic low power reconfigurations of real-time embedded systems. In: Proceedings of the 1st International Conference on Pervasive and Embedded Computing and Communication Systems, Portugal (2011)
12. Wang, X., Khemaissia, I., Khalgui, M., Li, Z.: Dynamic low-power reconfiguration of real-time systems with periodic and probabilistic tasks. IEEE Trans. Autom. Sci. Eng. **12**(1), 258–271 (2014)
13. Yao, F., Demers, A., Shenker, S.: A scheduling model for reduced CPU energy. In: 1995 Proceedings of the 36th Annual Symposium on Foundations of Computer Science, pp. 374–382 (1995)