Energy-Efficient Scheduling of Real-Time Tasks in Reconfigurable Homogeneous Multi-core Platforms

Aymen Gammoudi, Adel Benzina, Mohamed Khalgui, Daniel Chillet

Energy-Efficient Scheduling of Real-Time Tasks in Reconfigurable Homogeneous Multi-Core Platforms

Aymen Gammoudi, Adel Benzina, Mohamed Khalgui and Daniel Chillet

***Abstract*—This paper deals with real-time scheduling of homogeneous multi-core platforms powered by a battery to be periodically recharged. A system is composed of reconfigurable real-time dependent and periodic tasks to be assigned to different cores interconnected by a network-on-chip (NoC). The system is subject to reconfigurations, which are automatic operations allowing the addition and/or removal of tasks as well as their exchanged messages on the NoC. Consequently, any reconfiguration can violate real-time and energy constraints on cores as well as the NoC when the energy is unavailable until the next recharge. A novel periodic task model based on elastic coefficients and a new scheduling strategy are proposed to compute useful temporal parameters allowing for tasks and messages to meet the related constraints while controlling the communication cost on the NoC. This strategy is compared with an ILP-based optimal solution, and a tool named *OptimalMappingTasks* is developed to run different simulations that prove the originality of the paper's contribution.**

***Index Terms*—Homogeneous multi-core architecture; Real-time system, Low-power scheduling; Reconfiguration; Task mapping.**

A. Gammoudi is with the School of Electrical and Information Engineering, Jinan University (Zhuhai Campus), Zhuhai 519070, China, University of Rennes 1, France, and also with LISI Lab (INSAT), University of Carthage, Tunis 1080, Tunisia.
E-mail: aymen.gammoudi@irisa.fr

A. Benzina is with Tunisia Polytechnic School, and also with the National Institute of Applied Sciences and Technology, University of Carthage, Tunis 1080, Tunisia
E-mail: Adel.Benzina@isd.rnu.tn

M. Khalgui is with the School of Electrical and Information Engineering, Jinan University, Zhuhai 519070, China, and also with the National Institute of Applied Sciences and Technology, University of Carthage, Tunis 1080, Tunisia
E-mail: khalgui.mohamed@gmail.com

D. Chillet is with Institut National de Recherche en Informatique et Automatique (INRIA), and also with Ecole Nationale Superieure des Sciences Appliquees et de Technologies (ENSSAT), University of Rennes1, Lannion 22300, France
E-mail: daniel.chillet@irisa.fr

NOMENCLATURE

Π	Periodic task set;
τ_i	Periodic task;
W_i	Worst-case execution time (WCET) of τ_i;
D_i	Relative deadline of τ_i;
A_i	Release time of τ_i;
T_i	Period of τ_i;
$Tmax_i$	Maximum period of τ_i;
I_i	Importance factor of τ_i;
C_j	j^{th} Core handling periodic tasks;
U_{C_j}	Processor utilization of core C_j;
$M_{i,j}$	Message to be exchanged between τ_i and τ_j;
$TM_{i,j}$	Period of $M_{i,j}$;
$WM_{i,j}$	Worst-case transmission time (WCTT) of $M_{i,j}$;
$DM_{i,j}$	Absolute deadline of $M_{i,j}$;
$SM_{i,j}$	Relative size of $M_{i,j}$;
$RTSys$	Real-time system;
P	Power consumption;
P_{Total}	Total power consumption of the system;
P_{Limit}	Limit power consumption;
P_{Proc}	Power consumed by the processors;
P_{Com}	Power consumed by the communication traffic;
E	Energy available in the battery;
$t_{recharge}$	Time remaining until the next recharge;
Γ	Homogeneous core set;
N_1	Number of dependent periodic tasks;
N_2	Number of tasks added by the reconfiguration;
M	Number of tasks after the reconfiguration scenario;
p	Number of cores;
H	Mapping matrix;
ΔT_i	Integer value that extends the period of τ_i;
ΔW_i	Integer value that reduces the WCET of τ_i;
ΔTM_i	Integer value that extends the period of $M_{i,j}$;
ΔWM_i	Integer value that reduces the WCTT of $M_{i,j}$;
EU	Energy Unit;
QoS	Quality of Service;
$RTOS$	Real-Time Operating System;
$MPEG$	Moving Picture Experts Group;
EDF	Earliest Deadline First;
RM	Rate Monotonic;

I. Introduction

The number of reconfigurable real-time embedded platforms powered by batteries is constantly increasing with applications in aeronautics, mobile communication systems, automotive, robotics, telecommunications, environmental monitoring, consumer electronics, and so on [7]-[29]-[30]. In such applications, it is necessary to guarantee the stability of the controlled

system. Based on the environment conditions and also the required performance, the system designer imposes that each periodic task must be executed at a constant rate in order to guarantee the stability. For critical applications (where missing a deadline is a total system failure), the schedulability of the system has to be satisfied a priori and no task period modification is allowed at run-time. Such a critical application in which periodic tasks operate is determined by the schedulability analysis that must be done according to the task set to ensure its feasibility under constraints, such that real-time and/or energy. Feasibility analysis can be based on the work reported in [17] on the RM and the EDF policies, a periodic task τ_i is characterized by two parameters, its WCET W_i and its period T_i, which are assumed as constants for all task instances (homogeneous multi-core architecture). This is a fair supposition for hard real-time systems; however it might be too restrictive for other applications.

For instance, timing constraints are more flexible and dynamic in multimedia systems than control applications. Tasks such as voice sampling, image acquisition, data compression, and video playing are executed periodically, but their execution rates are not as strict as in control applications. Thus missing a deadline while displaying an MPEG video may degrade the quality of service (QoS), but this will not engender the systems failure. According to the requested QoS, the tasks may increase or decrease their execution rate to fit the requirements of other concurrent activities. The variation of the tasks rates would increase the flexibility of the system while managing the overload conditions and providing a more general admission control mechanism. When the system cannot guarantee the execution of a new task, it can reduce the rate of the other tasks (by increasing their periods in a controlled fashion) to decrease the processor utilization and accommodate the new request instead of rejecting it. In this context, several research studies reported in [5]-[14]-[15]-[28]-[31]-[32] have focused on the dynamic reconfiguration of real-time systems. Concerning the dynamic reconfiguration, two policies are defined in the literature: manual, applied by users [24]; and automatic, applied by intelligent control agents [16]. The most promising approach is the elastic scheduling reported in [5]-[6]-[12]-[23].

A system, denoted $RTSys$, can be implemented by N_1 dependent and periodic tasks to be assigned to p cores linked by a network-on-chip (NoC) [27] in order to ensure the communication between dependent tasks. Reducing the cost of communications between cores becomes a major concern to ensure high-performance and reliability of such systems. Several academic studies reported in [33] and [34] propose mapping algorithms using task clustering while considering both inter-task communication and execution costs. The basic idea of the clustering based algorithm reported in [33] and [34] is to group heavily communicating tasks into the same cluster. The tasks that are grouped into the same cluster are assigned to the same processor to avoid communication costs. These algorithms are interesting since the communication cost is reduced. Nevertheless, the authors are not interested in the verification of the processor utilization after assigning the tasks, i.e. the processor utilization exceeds 100% in certain cases. In addition, they do not check the overflow on the communication medium. To solve these two problems, effective solutions based on the modification of WCETs, deadlines, and periods of tasks to manage the processor utilization after any assignment of tasks are reported in [1], [5], [6], [12]. Similarily to the research works reported in [5]-[12]-[23], we consider a more flexible task model, in which the tasks can operate within a given range of periods with different performances. We consider that the system $RTSys$ is running on a reconfigurable platform which is powered by a battery and is recharged periodically with a well-known recharging period. A reconfiguration scenario is defined, in this paper, as any internal or external event that leads to the addition and/or removal of dependent periodic tasks as well as their exchanged messages to adapt the system's behavior to its environment [8].

After any reconfiguration scenario, the real-time constraints can become no more satisfied i.e. a task misses its deadline and a message takes a long time to arrive to its destination. Moreover, a reconfiguration scenario may increase the energy consumption and the system $RTSys$ can deplete the available energy before the next recharge. Therefore, the system $RTSys$ will be unfeasible and should provide optimal real-time solutions with a controlled power consumption to maintain the system up until the next planned recharge of the battery. An $RTSys$ is feasible if and only if it satisfies under different constraints: (i) the deadlines of the tasks are ensured/respected (real-time constraints), (ii) the messages deadlines on the communication medium are ensured (communication constraints), and (iii) the energy constraint is sufficient to ensure the execution until the next periodic recharge (energy constraints).

In real-time scheduling, effective solutions based on the modification of WCETs, deadlines, and periods of tasks in order to re-obtain the system feasibility after any reconfiguration scenario are reported in [1]-[5]-[6]-[9]-[12]. The work presented in [6] proposes a feasible low-power dynamic reconfiguration of real-time systems where addition and removal of tasks are applied at run-time. Three solutions are presented to cut-down the energy consumption after any reconfiguration scenario. The authors propose to extend the periods (or reduce the WCETs) of tasks by assigning a single value to all the tasks in order to re-obtain the system feasibility. Recently, the work reported in [1] improves these solutions by proposing a new approach based on grouping tasks in packs and modifying task parameters by assigning a unique period (or WCET) to all the tasks of the same pack. Each pack is a group of tasks having "similar" periods (or WCETs). In order to satisfy the real-time and the energy constraints after any reconfiguration scenario, specific modifications are to be undergone on the parameters of the packs and their related tasks. Each modification of periods (or WCETs) of the tasks has a cost in terms of delay. The cost of each modification on a task is the difference between the parameter's value before and after the reconfiguration. The total cost of the modification on the system is the sum of all these costs. In the research work reported in [3], the authors develop the $Reconf-Pack$ simulator to evaluate and compare these two approaches ([1] and [6]) by generating a large number of random systems and random reconfiguration scenarios for each one. The authors show that the total cost

delay introduced by applying the solution presented in [6] in 88% of the randomly generated cases is drastically higher than that introduced by the method reported in [1]. To show that the pack-based solution, proposed in [1], can be implemented in a real application, the work reported in [2] proposes a new reconfigurable middleware, named $Reconf - Middleware$, that presents a plugin implemented in RTLinux and describes the transition from the pack theory to the implementation. An extension of the technique reported in [1] to homogeneous multi-core platforms is described in [4]. The authors propose four solutions to schedule a real-time application composed of independent periodic tasks under energy constraints. The authors develop a new strategy for allocating N tasks to p cores of a given distributed system using task clustering by considering the load balancing over the different cores. The proposed strategy guarantees that, when a task is mapped into the system and accepted, then it is correctly executed prior to its deadline. Nevertheless, they do not consider the case of dependent periodic software tasks and they do not treat the influence of communication on the energy consumption.

The general goal of this paper is to propose a comprehensive strategy based on the pack-based solutions reported in [1]-[4]. It is applied in order to re-obtain the $RTSys$ feasibility after any reconfiguration scenario. The proposed strategy offers five solutions (A, B, C, D and E) based on the modification of temporal parameters of the processed tasks to meet the real-time, communication and energy constraints. If some constraints are violated after any reconfiguration scenario, then the proposed strategy starts first by modifying the periods of tasks and also the periods of the exchanged messages until reaching their specific maximal periods (Solution A or C). If $RTSys$ is still unfeasible, then it applies another solution to decrease the WCETs of tasks (Solution B). In the worst case, the strategy applies the last solution that removes some tasks according to a functional importance factor I_i defined for each task (Solution E). The proposed strategy applies these solutions in this order since the modification of periods (Solution A or C) does not have a significant influence on the consumed energy. Nevertheless, the application of Solution B or D increases implicitly the energy consumption.

To evaluate the performance of the proposed strategy, we perform different simulations scenario. We then compare these solutions with the research work reported in [6] and show that the cost in terms of delaying tasks is significantly reduced. We formalize this strategy as an optimization problem by using integer linear programming (ILP) and we compare the proposed solutions with the optimal results provided by the CPLEX solver [25]. The originality of this paper is twofold, the first one is to propose a new tool named $OptimalMappingTasks$ to look for the optimal placement of tasks on cores after any reconfiguration scenario in order to minimize the traffic on the NoC by trying to place the dependent tasks as close as possible to each other. The second is to propose a new scheduling strategy to meet the related constraints after any reconfiguration in order to re-obtain the $RTSys$ feasibility by modifying the parameters of tasks and the exchanged messages or by removing some of them. Even if the three constraints are simultaneously violated, the proposed strategy is deterministic since it is able to re-obtain the feasibility of the system by reducing less delay than that introduced by the method reported in [6]. The proposed algorithm is compared with two others EDF-based algorithms reported in [33] and [34] and we show that the traffic on the NoC is minimized.

The rest of this paper is organized as follows. Section II presents the system model and the used terminologies. We formalize the considered problem in Section III. The proposed strategy is presented in Section IV. This section deals also with the modification of temporal parameters of the processed tasks to guarantee the related real-time and energy constraints after any run-time event-based reconfiguration scenario. The strategy is implemented and simulated in Section IV-F. The performance evaluation of the proposed contribution is reported in Section IV-F. Some interesting results are discussed in Section VII. Finally, Section VIII concludes this work.

II. System Model and Assumptions

In this section, we present the task model and the platform architecture that consists of a reconfigurable real-time system $RTSys$ composed of periodic tasks. $RTSys$ is powered by a battery to be recharged periodically, and running on an architecture composed of p homogeneous cores. We also present, in this section, the models of hardware devices, including the processor architecture, battery and the communication between cores.

A. Task Model

The feasibility analysis of periodic tasks under earliest deadline first (EDF) scheduling was proposed by Liu and Layland [17]. EDF is a dynamic scheduling policy used in real-time operating systems to allocate tasks in a priority queue. Whenever a scheduling event occurs, the next task is extracted from the queue by considering the timing to the deadline. The task with the closest deadline will be extracted first, i.e., the task is the next to be scheduled for execution. With a preemptive scheme, among all scheduling policies, EDF is optimal [17]. Assuming that the period is equal to the deadline for each task, the necessary and sufficient condition for schedulability is to verify that the processor utilization rate is less than 1 [17]. Hence, in this paper, we apply EDF to schedule the tasks. We suppose that $RTSys$ processes a task set Π containing N_1 periodic tasks, i.e., $\Pi = \{\tau_1, \tau_2, ..., \tau_{N_1}\}$. Some tasks are considered as flexible, whose utilization can be modified by changing their periods within a specified range. The elastic model introduced by [12]-[23] considers task parameters as flexible as a spring with a given rigidity coefficient. In particular, we consider in this paper the task's period T_i as an elastic parameter whose value can be changed within a specified range. According to [5]-[12]-[23], a periodic task τ_i, $i \in \{1, 2, ..., N_1\}$, is characterized by: (i) a release time A_i, (ii) a worst-case execution time (WCET) W_i that represents the time elapsed from the time task acquires the processor to the end of the task without being interrupted. The WCETs are considered in this paper as inputs, (iii) a relative deadline D_i, (iv) a period T_i, (v) a maximum tolerable period $Tmax_i$, and (vi) a functional importance factor I_i to

be fixed manually by users. Such that more I_i is bigger, less τ_i is important. If $RTSys$ is not feasible or consumes too much energy than the available, then the tasks with the highest value of I_i should be removed to keep as many tasks as possible running on the system. The relative deadline of a periodic task is considered as a hard deadline if its missing is unacceptable [21].

B. Multi-Core Processor Model

We suppose that a homogeneous multi-core platform contains a set Γ of p cores Γ=$\{C_1, C_2, ..., C_p\}$. We assume that, each core C_j ($j \in [1..p]$) executes its local tasks by using the EDF algorithm. We assume that the cores can be reconfigured dynamically by authorizing the addition and/or removal of periodic tasks. We also assume that the frequency of each core can be changed in order to modify the execution time of tasks. To ensure that all the tasks can be executed on the cores, we define the mapping matrix H where

$$H_{i,j} = \begin{cases} 1 & \text{if task } \tau_i \text{ is running on core } C_j. \\ 0 & \text{otherwise.} \end{cases}$$

Furthermore, to guarantee that each task is executed at most by one core, it is necessary to verify that

$$\sum_{j=1}^{p} H_{i,j} = 1; \forall i \in [1..N_1] \tag{1}$$

C. Communication Model

Several distinguished studies deal with network-on-chip (NoC) and its different typologies [22]-[27]. In this paper, we do not consider a specific NoC architecture, but we present a general formalization which is adaptable to different architectures. As defined in subsection II-A, we consider a set Π of dependent periodic tasks, each one is affected to a particular core. We denote in the following by $M_{i,j}$, the message to be exchanged between a pair of tasks τ_i and τ_j. According to [18] the communication model is based on: (i) the regular inter-arrival time $TM_{i,j}$, (ii) the spent time to transmit a message $WM_{i,j}$, (iii) the absolute deadline $DM_{i,j}$, and (iv) the amount of exchanged data $SM_{i,j}$. We note that period $TM_{i,j}$ of message $M_{i,j}$ is equal to period T_i of sending task τ_i ($TM_{i,j}$=T_i). If the task's period is changed, then the message's period will be changed implicitly.
The NoC architecture implies a communication cost between each pair of cores depending on the distance between them. We model each NoC architecture by a specific cost matrix $Cost$ such as

$$Cost_{C_k,C_l} = \begin{cases} X_{C_k,C_l} & \text{if } k \neq l, \forall\ k,l \in [1..p]. \\ 0 & \text{otherwise.} \end{cases}$$

where X_{C_k,C_l} is a constant giving the cost of transferring one bit from core C_k to core C_l.
To calculate the cost of communication between each pair of cores C_k and C_l, it is necessary to multiply the volume of exchanged data by $Cost_{C_k,C_l}$. Then, the total cost of communications is given by

$$CommCost = \sum_{k=1}^{p}\sum_{l=1}^{p}\sum_{i=1}^{N_1}\sum_{j=1}^{N_1} SM_{i,j} * Cost_{C_k,C_l} * H_{i,k} * H_{j,l} \tag{2}$$

We suppose that the energy consumed by the communication is proportional to the distance between cores.

D. Processor Utilization Model

According to [17] the processor utilization U_{C_j} of core C_j is given by

$$U_{C_j} = \sum_{i=1}^{N_1} \frac{W_i}{T_i} * H_{i,j}; \forall j \in [1..p] \tag{3}$$

The core C_j is feasible under the EDF policy, if and only if

$$U_{C_j} \leq 1; \forall j \in [1..p] \tag{4}$$

Eq. 4 is the real-time constraint, denoted **RTConst**.
According to the works reported in [18]-[20]-[22], to evaluate the communication feasibility between two cores, the related medium is considered as a virtual processor. Based on this hypothesis, the utilization of the communication medium between two cores C_k and C_l, denoted $U_{Com}(C_k, C_l)$, is given by

$$U_{Com}(C_k, C_l) = \sum_{i=1}^{N_1}\sum_{j=1}^{N_1} \frac{WM_{i,j}}{TM_{i,j}} * H_{i,k} * H_{j,l}; \forall k,l \in [1..p] \tag{5}$$

According to the work reported in [22], to ensure that the communication between cores that are physically close is feasible, it is necessary to check

$$U_{Com}(C_k, C_l) \leq 1; \forall k,l \in [1..p] \mid Cost_{C_k,C_l} = 1 \tag{6}$$

Eq. 6 is the communication constraint, denoted **CommConst**.

E. Power Consumption Model

In the works reported in [6] and [19], the authors show that the power consumed by a DVS processor is directly proportional with at least the squared value of its utilization, i.e., $P\ \alpha\ U^2$. We consider that $RTSsys$ is periodically fully recharged. Thus, the power consumption of core C_j is given by

$$P_{C_j} = k * U_{C_j}^2; \forall j \in [1..p] \tag{7}$$

Hence, the power consumption of all the tasks running in the p cores is given by

$$P_{Proc} = \sum_{j=1}^{p} P_{C_j} \tag{8}$$

Similarly, we define the power consumption due to the communication on the NoC by

$$P_{Com} = k \sum_{k=1}^{p}\sum_{l=1}^{p} U_{Com}^2(C_k, C_l) \tag{9}$$

Therefore, the total power consumption of $RTSys$ is

$$P_{Total} = P_{Proc} + P_{Com} \quad (10)$$

Since, we consider periodic tasks, it is possible to compute the energy needed to support their until the next battery recharge. We suppose that $E(t)$ is the available energy in the battery at time t and $t_{recharge}$ is the time remaining until the next recharge. We note that these two parameters are supposed known at any time t. Therefore, to ensure that $RTSys$ runs correctly until the next recharge, assuming that the configuration will remain the same, it is necessary that at time t, i.e.,

$$P_{Total}(t).t_{recharge} \leq E(t) \quad (11)$$

We define the variable P_{Limit} by the quantity $\frac{E(t)}{t_{recharge}}$. It is the maximum power consumption tolerable at time t otherwise $RTsys$ will be off due to energy shortage before the next recharge. Thus, after each reconfiguration scenario, we have to satisfy the following power consumption constraint, i.e.,

$$P_{Total} \leq P_{Limit} \quad (12)$$

Eq. 12 is the energy constraint, denoted **EnergyConst**.

III. Problem Position

We present in this section the problem of reconfiguration and we describe and formalize each constraint as an optimization problem by using ILP. This section presents first the general ILP formulation of the problem, and continues with the ILP formulations of the proposed heuristics.

A. Reconfiguration Problem

We suppose that at time t_0 $RTSys$ is initially composed of $\Gamma(t_0) = \{C_1, C_2, ..., C_p\}$ and $\Pi(t_0) = \{\tau_1, \tau_2, ..., \tau_{N_1}\}$. Assuming that $RTSys$ at t_0 is feasible implies that all three constraints of real-time, energy and communication must be satisfied. In the following, we assume that $RTSys$ is dynamically reconfigured at time t_1 by adding N_2 tasks. Therefore, the new implementation of tasks is $\Pi(t_1) = \{\tau_1, \tau_2, ..., \tau_{N_1}, \tau_{N_1+1}, ..., \tau_M\}$, with $M = N_1 + N_2$. This reconfiguration scenario is appointed in the following by $RE\text{-}CONF$. To ensure that the system runs correctly after $RE\text{-}CONF$, it is necessary to guarantee that the new configuration satisfies the constraints of the following model

$$RE \begin{cases} Minimize\ CommCost & (Eq.2) \\ s.t. & \\ \sum_{j=1}^{p} H_{i,j} = 1, \forall\ i \in [1..M] & (Eq.1) \\ U_{C_j} \leq 1, \forall\ j \in [1..p] & (Eq.4) \\ U_{Com}(C_k, C_l) \leq 1 \ \mid Cost_{C_k,C_l} = 1 & (Eq.6) \\ \forall\ k, l \in [1..p] & \\ P_{Total} \leq P_{Limit} & (Eq.12) \end{cases}$$

where $Eq.2$ calculates the total cost of the communications, $Eq.1$ guarantees that each task can be executed at most by one core, $Eq.4$ satisfies the real-time scheduling under the EDF policy for each core, $Eq.6$ ensures that the communication between the cores C_k and C_l is feasible and $Eq.12$ ensures a rational use of the remaining energy until the next recharge. The RE problem can be solved by integer programming solvers such as CPLEX [25], which is hard to be implemented in an embedded platform since it is too complex to be executed on-line. We note that this solver does not produce any solution if one or more constraints are not satisfied. To solve the problem RE, it is necessary that all the constraints are satisfied. Nevertheless, after each $RE\text{-}CONF$ scenario, one or more of these constraints can be violated. We formalize first of all each constraint as an optimization problem by using the ILP formulation. Then, we present a heuristic to solve them in an efficient way.

B. Real-Time Problem

After the $RE\text{-}CONF$ scenario, if core C_j is not feasible ($U_{C_j} > 1$) and since $U_{C_j} = \sum_{i=1}^{M} \frac{W_i}{T_i} * H_{i,j}; \forall j \in [1..p]$, we propose first of all to modify the periods of the local real-time tasks in order to decrease the processor utilization. The principle is to extend the periods of tasks until they reach $Tmax_i$. If U_{C_j} is still greater than 1, then we have to reduce the WCETs in order to satisfy $U_{C_j} \leq 1$. In the worst case, we apply another alternative that removes some tasks according to their importance factor I_i. Using the ILP model, we formalize this problem by $\forall\ j \in [1..p] \mid U_{C_j} > 1$

$$Pb1 = \begin{cases} Minimize \sum_{i=1}^{M} \Delta W_i * H_{i,j} \\ s.t. \\ \sum_{i=1}^{M} \frac{W_i - \Delta W_i}{T_i + \Delta T_i} * H_{i,j} \leq 1 \\ W_i - \Delta W_i > 0, \forall\ i \in [1..M] \\ T_i + \Delta T_i \leq Tmax_i, \forall\ i \in [1..M] \end{cases}$$

where ΔT_i is an integer that extends the period of the real-time tasks and ΔW_i is an integer that reduces their WCETs. This formalization permits to minimize the WCETs modification and exploits the possible flexibility of the periods.

C. Energy Problem

After the reconfiguration scenario $RE\text{-}CONF$, the processor utilization of certain cores will be increased due to the added tasks. According to the work reported in [6], the power consumption P is quadratically dependent on the processor utilization. The new power consumption, in the worst case, may surpass the available energy budget. Therefore, $RTSys$ can violate the energy constraint. In this context, we suggest to decrease the processor utilization of core C_{high} ($high \in [1..p]$) that has the highest processor utilization until the energy constraint is satisfied. To decrease the processor utilization, we propose to adapt the periods of real-time tasks, running

on C_{high}, until they reach $Tmax_i$ in order to ensure that the system $RTSys$ will run correctly until the next battery recharge. This idea is formalized by

$$Pb2 \begin{cases} Minimize \sum_{i=1}^{M} \Delta T_i * H_{i,high} \\ s.t. \\ \sum_{i=1}^{M} \frac{W_i}{T_i + \Delta T_i} * H_{i,high} \leq 1 \\ T_i + \Delta T_i \leq Tmax_i, \forall\ i \in [1..M] \\ P_{Total} \leq P_{Limit} \qquad (Eq.12) \end{cases}$$

After adapting the task parameters, running on C_{high}, the processor utilization $U_{C_{high}}$ is decreased and the energy constraint (Eq. 12) can be satisfied. If it is still violated, then we propose to remove some tasks according to their importance factor I_i.

D. Communication Problem

After the addition of a pair of dependent tasks on two cores that can be caused by $RE\text{-}CONF$, a periodic message will be added to the NoC and should respect a corresponding deadline related to these tasks. Then, it is necessary to verify the feasibility of the communication between each pair of cores C_k and C_l ($k, l \in [1..p]$). According to Eq. 6, if $U_{Com}(C_k, C_l)$ is greater than 1, then the communication between these two cores is not feasible. In order to solve this problem, we propose to modify the period of messages exchanged between the tasks running on cores C_k and C_l. We can formalize this problem by

$\forall\ k, l \in [1..p] \mid U_{Com}(C_k, C_l) > 1$

$$Pb3 = \begin{cases} Minimize \sum_{i=1}^{M} \sum_{j=1}^{M} \Delta TM_{i,j} * H_{i,k} * H_{i,l} \\ s.t. \\ \sum_{i=1}^{M} \sum_{j=1}^{M} \frac{WM_{i,j}}{TM_{i,j} + \Delta TM_{i,j}} * H_{i,k} * H_{i,l} \leq 1 \\ TM_{i,j} + \Delta TM_{i,j} \leq Tmax_i, \forall\ i, j \in [1..M] \end{cases}$$

where $\Delta TM_{i,j}$ is an integer value that extends the period of message $M_{i,j}$.

$Pb1$, $Pb2$ and $Pb3$ can be solved by CPLEX [25] that provides an optimal solution since it seeks for each task and message the suitable and exact modification with a minimal cost. Nevertheless, it is not reasonable to implement an ILP solver in an embedded system. In other words and since we propose a new strategy to schedule real-time tasks in an operating system, it is too complex to execute on-line an ILP solver, but it is useful to calculate the optimal solution that will be considered as a reference target for any implementable heuristic. Therefore, we present in the next Section some efficient and implementable heuristics compared with the optimal solutions provided by the CPLEX solver to solve these problems.

IV. Contribution

When the $RE\text{-}CONF$ scenario is applied at run-time, the processor utilization of certain cores will be certainly increased. If the new utilization is greater than 1, then the real-time constraint is violated as reported in [6]-[17]. In addition, the energy and/or communication constraints can be violated after the addition of tasks and consequently $RTSys$ will be unfeasible. In this section, we present in Fig. 1 a new scheduling strategy based on adapting task parameters T_i and/or W_i to fulfill the feasibility condition. If some tasks of the new execution model violate the related deadlines, or if the power consumption is increased, or the charge of the communications between two cores is greater than 1, then this strategy proposes new solutions in order to re-obtain the $RTSys$ feasibility with a low power consumption. It proposes also to remove some tasks according to their importance factor I_i. Each modification of periods or WCETs of tasks has a cost in terms of delay. The cost of each modification on a task is the difference between the parameter's value before and after this reconfiguration scenario. Therefore, the objective is to re-obtain the system feasibility while minimizing the modification cost. We present in the remainder of this section some new solutions with various costs and with different implementation complexities.

A. Technical Overview

In order to re-obtain the system feasibility, we propose a heuristic based on grouping the tasks that have "similar" periods (or WCETs) in several packs, denoted $Pack$. As described in the second part of Section I, after a reconfiguration, we do not calculate a new period for each task, but we assign a unique new period (or new WCET) to all tasks of the first pack $Pack_{1,j}$ ($j \in$ [1..p]). Moreover, all the new periods (or WCETs) affected to the tasks of pack $Pack_{x,j}$ are multiple of the one affected to the tasks of $Pack_{1,j}$. Hence, we have only to compute the suitable value $Pk_{1,j}$. In order to satisfy the real-time and the energy constraints after any reconfiguration scenario, specific modifications are to be undergone on the parameters of the packs and their related tasks. Each modification of periods (or WCETs) of tasks has a cost in terms of delay. The cost of each modification on a task is the difference between the parameter's value before and after a reconfiguration. Full details can be found in [1].

B. Solution A: Periods' Modification under Real-Time Constraint

If C_j ($j \in$ [1..p]) is not feasible after the $RE\text{-}CONF$ scenario, then we propose to extend the periods of tasks that run on C_j. In order to satisfy the real-time constraints, we assign each task to the pack $Pack^T_{x,j}$ according to its period. To construct the packs, we need to seek the suitable new period $Pk^T_{1,j}$ that minimizes the cost of the new solution for the whole

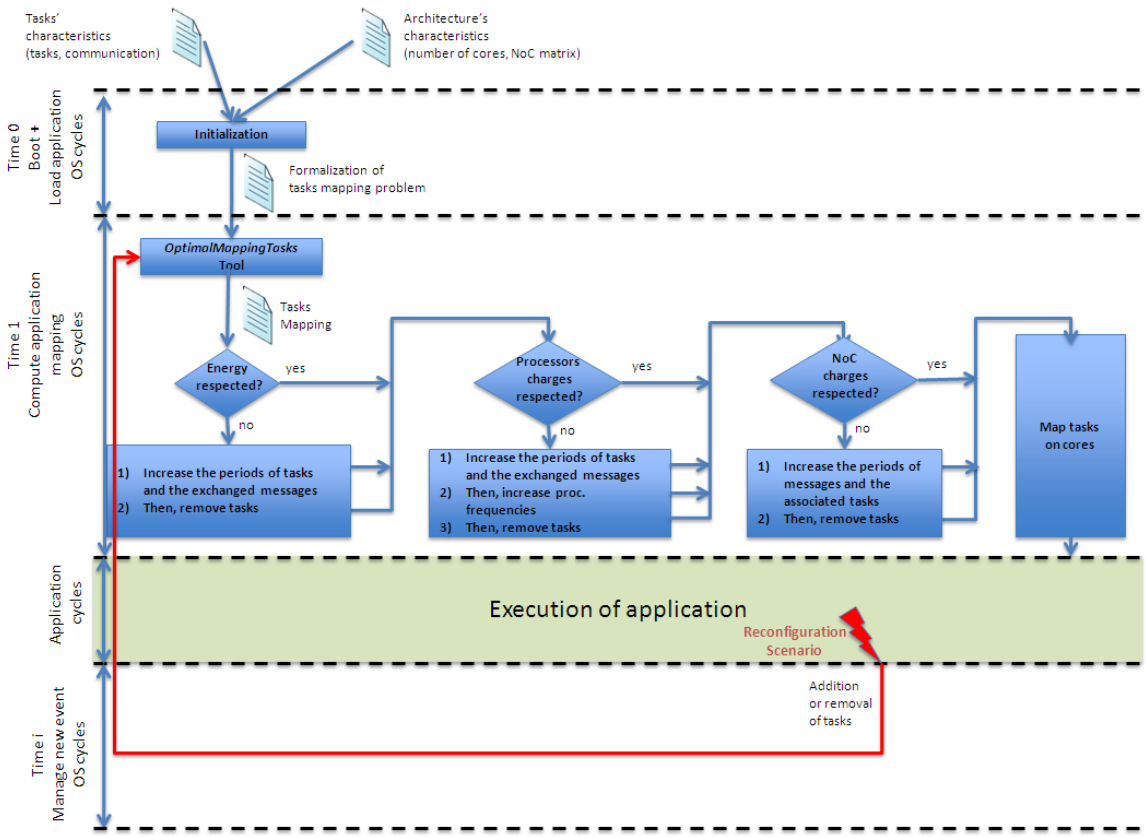

Fig. 1. Steps of strategy.

system. We formalize this problem by

$$\begin{cases} Min \sum_{i=1}^{M} \left((Pk_{1,j}^T - (T_i \ mod \ Pk_{1,j}^T)) \ mod \ Pk_{1,j}^T \right) * H_{i,j} \\ s.t. \\ Pk_{1,j}^T \geq Min(T_i), \forall \ i \in [1..M] \mid H_{i,j} = 1 \end{cases}$$

Once $Pk_{1,j}^T$ is calculated, we construct the packs of the system tasks as follows: $Pack_{x,j}^T, x \geq 1$, includes the tasks that have a period in the range $[(x-1) * Pk_{1,j}^T + 1 \quad ; \quad x * Pk_{1,j}^T]$, $x \in \mathbb{N}^+$. The first value of x is 1 and when all tasks are affected, x is no more incremented.

Example:

$Pack_{1,j}^T$ includes the tasks that have a period in the range $[1 \quad ; \quad Pk_{1,j}^T]$, $Pack_{2,j}^T$ includes the tasks that have a period in the range $[Pk_{1,j}^T + 1 \quad ; \quad 2 * Pk_{1,j}^T]$, etc.

To satisfy the real-time constraints under the EDF policy, it is necessary that $\sum_{i=1}^{M} \left(\frac{W_i}{T_i} * H_{i,j} \right) \leq 1$. Since the periods are now multiple of the same value $Pk_{1,j}^T$, we get

$$\sum_{\tau_i \in Pack_{1,j}^T} \frac{W_i}{Pk_{1,j}^T} + ... + \sum_{\tau_i \in Pack_{x,j}^T} \frac{W_i}{x.Pk_{1,j}^T} \leq 1$$

thus,

$$\frac{1}{Pk_{1,j}^T} * \left(\sum_{\tau_i \in Pack_{1,j}^T} W_i + ... + \sum_{\tau_i \in Pack_{x,j}^T} \frac{W_i}{x} \right) \leq 1$$

then,

$$Pk_{1,j}^T \geq \sum_{\tau_i \in Pack_{1,j}^T} W_i + ... + \sum_{\tau_i \in Pack_{x,j}^T} \frac{W_i}{x}$$

Since the periods are integer values, then,

$$Pk_{1,j}^T = \left\lceil \sum_{\tau_i \in Pack_{1,j}^T} W_i + ... + \sum_{\tau_i \in Pack_{x,j}^T} \frac{W_i}{x} \right\rceil \qquad (13)$$

where $Pk_{1,j}^T$ is the new period affected to tasks of the first pack $Pack_{1,j}^T$, $x * Pk_{1,j}^T$ to the tasks of x^{th} pack $Pack_{x,j}^T$. After adapting the tasks' parameters running on C_j, all the tasks are executed without missing their deadlines. Therefore, the real-time constraint can be satisfied. As described in subsection II-C, if the task's period is changed implicitly, then the message's period will be changed since $TM_{i,j}$=T_i. For that, the periods of messages that are associated to the adapted tasks will be changed.

C. Solution B: WCETs' Modification under Real-Time Constraint

In this part, as an alternative solution, we adjust the WCETs of tasks to re-obtain the feasibility of C_j. We use the same technique presented in Solution A to calculate the new WCETs of tasks that run on core C_j ($j \in$ [1..p]). First of all, we seek the value $Pk^W_{1,j}$ in order to group the tasks according to their WCETs. We formalize it by

$$\begin{cases} Min \sum_{i=1}^{M} \left((Pk^W_{1,j} - (W_i \ mod \ Pk^W_{1,j})) \ mod \ Pk^W_{1,j}\right) * H_{i,j} \\ s.t. \\ Pk^W_{1,j} \geq Min(W_i), \forall \text{ i} \in [1..M] \mid H_{i,j} = 1 \end{cases}$$

We construct the packs of tasks as follows: $Pack^W_{x,j}$ includes the tasks having a WCET in the range $[(x-1) * Pk^W_{1,j} + 1 \quad ; \quad x * Pk^W_{1,j}]$. The objective is the same to satisfy the real time constraint $\sum_{i=1}^{M} \left(\frac{W_i}{T_i} * H_{i,j}\right) \leq 1$. Since the WCETs W_i are multiple of the same value $Pk^W_{1,j}$, we get

$$\sum_{\tau_i \in Pack^W_{1,j}} \frac{Pk^W_{1,j}}{T_i} + ... + \sum_{\tau_i \in Pack^W_{x,j}} \frac{x.Pk^W_{1,j}}{T_i} \leq 1$$

thus,

$$Pk^W_{1,j} * \left(\sum_{\tau_i \in Pack^W_{1,j}} \frac{1}{T_i} + ... + \sum_{\tau_i \in Pack^W_{x,j}} \frac{x}{T_i}\right) \leq 1$$

therefore,

$$Pk^W_{1,j} = \left\lfloor \frac{1}{\sum_{\tau_i \in Pack^W_{1,j}} \frac{1}{T_i} + ... + \sum_{\tau_i \in Pack^W_{x,j}} \frac{x}{T_i}} \right\rfloor \quad (14)$$

where $Pk^W_{1,j}$ is the new WCET affected to the tasks of the first pack $Pack^W_{1,j}$. $\tau_i \in Pack^W_{x,j}$ if and only if $(x-1) * Pk^W_{1,j} + 1 \leq W_i \leq x * Pk^W_{1,j}$
After the modification of the WCETs, the processor utilization of C_j is reduced and can satisfy the real-time scheduling.

D. Solution C: Periods' Modification under Energy Constraint

Let us assume that the core C_{high} violates the energy constraint after the $RE\text{-}CONF$ scenario. This solution proposes to extend the periods of tasks running on C_{high} ($high \in$ [1..p]) in order to ensure that the current power should be less than the critical one.
The energy constraint is $P_{Total} \leq P_{Limit}$ and Eq. 10 is

$$P_{Proc} + P_{Com} \leq P_{Limit}$$

i.e.,

$$\sum_{j=1}^{p} P_{C_j} \leq P_{Limit} - P_{Com}$$

$$P_{C_{high}} + \sum_{j=1}^{p-1} P_{C_j} \leq P_{Limit} - P_{Com}$$

thus,

$$P_{C_{high}} \leq P_{Limit} - P_{Com} - \sum_{j=1}^{p-1} P_{C_j}$$

According to Eq. 7

$$k * U^2_{C_{high}} \leq P_{Limit} - P_{Com} - \sum_{j=1}^{p-1} P_{C_j}$$

then,

$$U_{C_{high}} \leq \sqrt{\frac{P_{Limit} - P_{Com} - \sum_{j=1}^{p-1} P_{C_j}}{k}}$$

We suppose that $\alpha = \sqrt{\frac{P_{Limit} - P_{Com} - \sum_{j=1}^{p-1} P_{C_j}}{k}}$
then,

$$\sum_{i=1}^{M} \left(\frac{W_i}{T_i} * H_{i,high}\right) \leq \alpha$$

$Pack^T_{x,high}$ is constructed to include the tasks that have a period in the range $[(x-1) * Pk^T_{1,high} + 1 \quad ; \quad x * Pk^T_{1,high}]$, i.e.,

$$\sum_{\tau_i \in Pack^T_{1,high}} \frac{W_i}{Pk^T_{1,high}} + ... + \sum_{\tau_i \in Pack^T_{x,high}} \frac{W_i}{x.Pk^T_{1,high}} \leq \alpha$$

thus,

$$\frac{1}{Pk^T_{1,high}} * \left(\sum_{\tau_i \in Pack^T_{1,high}} W_i + ... + \sum_{\tau_i \in Pack^T_{x,high}} \frac{W_i}{x}\right) \leq \alpha$$

finally,

$$Pk^T_{1,high} = \left\lceil \frac{\sum_{\tau_i \in Pack^T_{1,high}} W_i + ... + \sum_{\tau_i \in Pack^T_{x,high}} \frac{W_i}{x}}{\alpha} \right\rceil \quad (15)$$

where $Pk^T_{1,high}$ is the new period affected to the tasks of $Pack^T_{1,high}$ and the other packs are multiple. After adapting the task' parameters, the processor utilization of C_{high} is decreased. Therefore, the system can run correctly until the next recharge. We note that the period $TM_{i,j}$ of message $M_{i,j}$ is equal to the period T_i of sending task τ_i. Since this solution proposes to adapt the tasks' periods, thus the periods of messages associated to the modified tasks are changed implicitly.

E. Solution D: Modification of Message's Period under Communication Constraint

We are interested in this part in the communication feasibility. After the $RE\text{-}CONF$ scenario, a message are added to the NoC whenever two related added dependent tasks are assigned to different cores. If $U_{Com}(C_k, C_l)$ ($\forall \ k, l \in$ [1..p]) is greater than 1, then the communication between the cores is not feasible. We propose to extend the periods of messages exchanged between the tasks running on cores C_k and C_l. In order to satisfy the communication constraint, we assign each message to $Pack^{TM}_{x,k,l}$ according to its period. We formalize

this problem by
$\forall\ k,l \in [1..p] \mid U_{Com}(C_k, C_l) > 1$

$$\begin{cases} Min\ \sum_{i=1}^{M}\sum_{j=1}^{M}(Pk_{1,k,l}^{TM} - (TM_{i,j}\ mod\ Pk_{1,k,l}^{TM}))\ mod \\ \quad Pk_{1,k,l}^{TM} * H_{i,k} * H_{i,l} \\ s.t. \\ Pk_{1,k,l}^{TM} \geq Min(TM_{i,j}),\ \forall\ i,j \in [1..M] \end{cases}$$

Once $Pk_{1,k,l}^{TM}$ is calculated, the packs of messages are processed such that $Pack_{x,k,l}^{TM}$ includes the messages that have a period in the range $[(x-1) * Pk_{x,k,l}^{TM} + 1 \quad ; \quad x * Pk_{1,k,l}^{TM}]$, $x \in \mathbb{N}^+$.
To satisfy the communication constraint, it is necessary that $\sum_{i=1}^{M}\sum_{j=1}^{M}\left(\frac{WM_{i,j}}{TM_{i,j}} * H_{i,k} * H_{j,l}\right) \leq 1$. Since the periods $TM_{i,j}$ are now multiple of the same value $Pk_{1,k,l}^{TM}$, we get

$$\sum_{M_{i,j} \in Pack_{1,k,l}^{TM}} \frac{WM_{i,j}}{Pk_{1,k,l}^{TM}} + ... + \sum_{M_{i,j} \in Pack_{x,k,l}^{TM}} \frac{WM_{i,j}}{x.Pk_{1,k,l}^{TM}} \leq 1$$

Since the periods are integer, then,

$$Pk_{1,k,l}^{TM} = \left\lceil \sum_{M_{i,j} \in Pack_{1,k,l}^{TM}} WM_{i,j} + ... + \sum_{M_{i,j} \in Pack_{x,k,l}^{TM}} \frac{WM_{i,j}}{x} \right\rceil \quad (16)$$

where $Pk_{1,k,l}^{TM}$ is the new period affected to the messages of the first pack $Pack_{1,k,l}^{TM}$, $x * Pk_{x,k,l}^{TM}$ to the messages of x^{th} pack $Pack_{x,k,l}^{TM}$. After adapting the parameters of messages running on the communication bus between cores C_k and C_l, $U_{Com}(C_k, C_l)$ will be smaller than 1. Therefore, the communication constraint can be satisfied. Since the periods of messages are modified, it is necessary to adapt the periods of their sending tasks.

F. Solution E: Removal of Tasks

This is the last chance solution proposed by the strategy described in Fig. 1. It allows the removal of tasks in order to cut-down the processor utilization after the $RE\text{-}CONF$ scenario. By defining for each task τ_i an importance factor I_i, the strategy suggests removing tasks that have the higher I_i because their removal can be useful for a low power reconfiguration of the system.

G. Deterministic Strategy of Reconfigurable Platforms

In this section, we present the operating mode to allow a feasible multi-core system $RTSys$. We therefore need to develop a tool to find the optimal mapping of the tasks on cores in order to optimize the energy consumption while meeting temporal and communication constraints. Communicating tasks are placed as close as possible. This tool, named $OptimalMappingTasks$ (Fig. 2), solves the problem formalized by RE in sub-section III-A. It provides a graph that presents the energy cost for each possible placement and it chooses just the possibilities that have a minimal cost of energy consumption and that occupy a minimum number of cores. After any reconfiguration scenario, $OptimalMappingTasks$ places all the tasks on cores. After this step, the real-time, communication and/or energy constraints can be violated. In this section, the defined strategy proposes a deterministic choice between the different solutions (A, B, C, D or E). It starts first by modifying the periods of tasks and the exchanged messages until reaching their specific allowed maximal periods (Solution A or C). If $RTSys$ is still unfeasible, then it decreases the WCETs of tasks (Solution B). In the worst case, the strategy applies the last solution that removes some tasks according to an importance functional factor that we name I_i (Solution E). Algorithm 1 explains these steps and uses three functions: (i) **Feasible(RTSys)** that returns true if the system $RTSys$ is feasible, (ii) **Max(Solution X, Solution Y)** selects the solution that has the highest value $Pk_{1,j}^{T}$ from solutions X and Y, and (iii) **Apply(Solution X)** which applies the solution X.

Algorithm 1 Deterministic Strategy

```
Input RTSys after any configuration/reconfiguration scenario
Seek the mapping by OptimalMappingTasks tool
if ((!RTConst) and (!EnergyConst)) then
    SolutionRTEg = Max{Solution A, Solution C}
    Apply (SolutionRTEg) // increases the periods of tasks and the exchanged messages
    if (!Feasible(RTSys)) then
        if (!CommConst) then
            Apply (Solution D) // increases the periods of messages and the associated tasks.
        else
            Apply (Solution B) // increases the processor's frequencies
        end if
    end if
else if (!RTConst) and (EnergyConst) then
    Apply (Solution A) // increases the periods of tasks and the exchanged messages
    if (!Feasible(RTSys)) then
        if (!CommConst) then
            Apply (Solution D) // increases the periods of messages and the associated tasks.
        else
            Apply (Solution B) // increases the processor's frequencies
        end if
    end if
else if (RTConst) and (!EnergyConst) then
    Apply (Solution C) // increases the periods of tasks and the exchanged messages
    if (!CommConst) and (!Feasible(RTSys)) then
        Apply (Solution D) // increases the periods of messages and the associated tasks.
    end if
else if (RTConst) and (EnergyConst) and (!CommConst) then
    Apply (Solution D) // increases the periods of messages and the associated tasks.
end if
if (!Feasible(RTSys)) then
    Apply (Solution E) // removal of tasks
end if
Output RTSys is feasible
```

V. Implementation and Experimental Study

The goal of this section is to simulate the proposed solutions by following the presented Algorithm. 1.

We suppose that we have a set Π of periodic tasks that will be placed on a set of different cores. We use the same case study Figs. 7 and 8 reported in [6] as a subset in Π where all the tasks are considered independent. In order to evaluate the proposed mapping tool $OptimalMappingTasks$, we add some dependency constraints between these tasks (Tab. I). We suppose that the multi-core platform uses the one-dimensional network to exchange all messages between the p cores where the total length of the bus is split into $p-1$ segments. To map

all the tasks, we suppose that we have 5 cores ($p = 5$) and the cost matrix is

$$Cost = \begin{array}{c} \begin{array}{ccccc} C_1 & C_2 & C_3 & C_4 & C_5 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \end{array} \\ \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 0 & 1 & 2 & 3 \\ 2 & 1 & 0 & 1 & 2 \\ 3 & 2 & 1 & 0 & 1 \\ 4 & 3 & 2 & 1 & 0 \end{pmatrix} \end{array} \begin{array}{l} \leftarrow C_1 \\ \leftarrow C_2 \\ \leftarrow C_3 \\ \leftarrow C_4 \\ \leftarrow C_5 \end{array}$$

The objective is to minimize the traffic cost on the NoC under real-time, energy and communication constraints. We suppose that the energy available in the battery is 450 EU (Energy Units). Fig. 2 presents the optimal cost of tasks' placement and shows the corresponding mapping. 1

TABLE I
CHARACTERISTICS OF THE PERIODIC MESSAGES

Message	WM	TM	SM	Message	WM	TM	SM
$M_{1,2}$	3	500	50	$M_{15,16}$	7	680	60
$M_{2,1}$	4	700	30	$M_{10,12}$	4	620	45
$M_{1,3}$	3	500	20	$M_{12,10}$	4	620	40
$M_{3,1}$	5	720	20	$M_{12,13}$	4	620	50
$M_{4,5}$	6	740	55	$M_{13,12}$	5	740	80
$M_{5,4}$	6	760	35	$M_{20,21}$	6	780	80
$M_{16,15}$	3	700	55	$M_{21,20}$	3	800	80

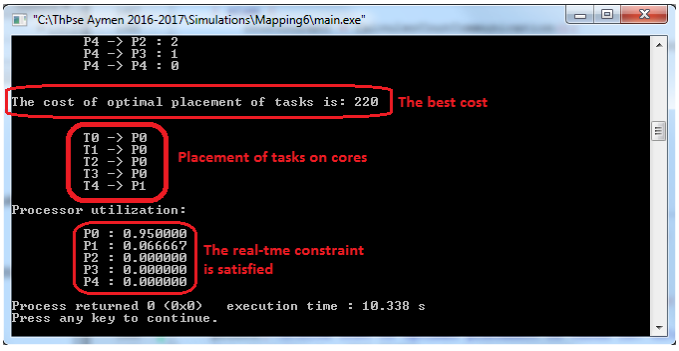

Fig. 2. Presentation of $OptimalMappingTasks$.

Fig. 3 represents the different possibilities to map all the tasks. Each blue circle indicates a possible mapping of tasks and has a corresponding value of energy consumption. We have in this case 78125 possibilities for mapping the given tasks to cores.

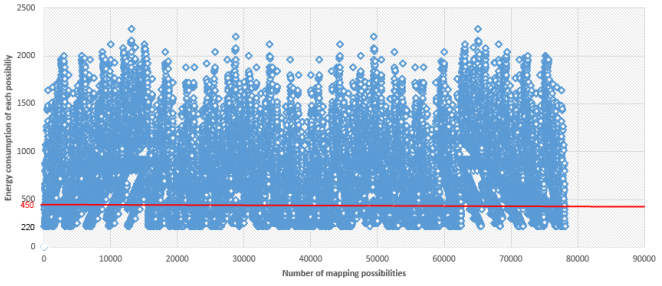

Fig. 3. Energy consumption of all mapping possibilities.

We notice that $OptimalMappingTasks$ provides many possibilities which consume energy less than 450 EU. According to the result produced by this tool, 220 EU is the minimal energy consumption which respects the energy constraint (Fig. 3). We note that there is 93 mapping possibilities that lead to this minimal energy consumption. Therefore, the chosen mapping is one of the possibilities that has the minimal energy consumption (220 EU in Fig. 3).

Once all the tasks are placed on cores, Algorithm. 1 that represents the proposed strategy must check the rest of constraints. If one or more of them are violated, then it must apply the suitable solutions for each core. It proposes to increase the periods or to modify the execution time of tasks or to remove some of them.

Since the proposed solutions can be applied core per core and in order to compare the current paper's contribution to Wang's approach [6], we suppose that the tasks in Figs. 7 and 8 in [6] are executed by the five considered cores. The objective is to apply the proposed solutions and the solutions reported in [6] to the same case study.

After the implementation of these tasks on C_1, the real-time constraint is not satisfied because the processor utilization is greater than 1 ($U_{C_1} = 1.02$). To re-obtain the core feasibility, we explore (i) the heuristic proposed in solution A, (ii) the optimal solution formalized by $Pb1$, and (iii) the solution presented in [6]. These solutions deal with the modification of temporal parameters of the processed tasks where each modification has a cost in terms of delay. The cost of each modification on a task is the difference between the parameter's value before and after a reconfiguration. After the modification of the periods (Solution A), the processor utilization is decreased and can satisfy the real-time scheduling. After the execution of these three solutions, the processor utilization is reduced to 0.997 and satisfies the real-time scheduling. Fig. 4 compares the performance of the proposed solution A (bars in orange) with the approach reported in [6] (bars in grey) and the optimal solution generated by CPLEX (bars in blue). Using the same colors, Fig. 5 compares the performance of the proposed solution B with the approach reported in [6]. We note that the total cost (Fig. 6) of the proposed strategy is equal to 3275 ms, the total cost by using the approach proposed by Wang et al. in [6] is equal to 24550 ms and the total cost by applying the optimal solution is 2491 ms. In other words, the proposed heuristic introduces a delay of 1.3 times ($\frac{3275}{2491}$) the delay introduced by the optimal solution whereas Wang's method introduces 9.8 times ($\frac{24550}{2491}$) the delay of the optimal solution.

In order to evaluate the performance of the proposed strategy, a second tool has been developed and published in [3] to generate random systems and reconfiguration scenarios. We compare the proposed solution with the approach proposed in Wang et al. [6] using 50 randomly generated systems. After observing the results, it is noticed that the delay in 88% of randomly generated tests is reduced thanks to the proposed solution. Moreover, the proposed strategy lowers the average delay to keep the system feasible and up until next recharge. This delay is only 55% of that resulting from Wang's strategy [6].

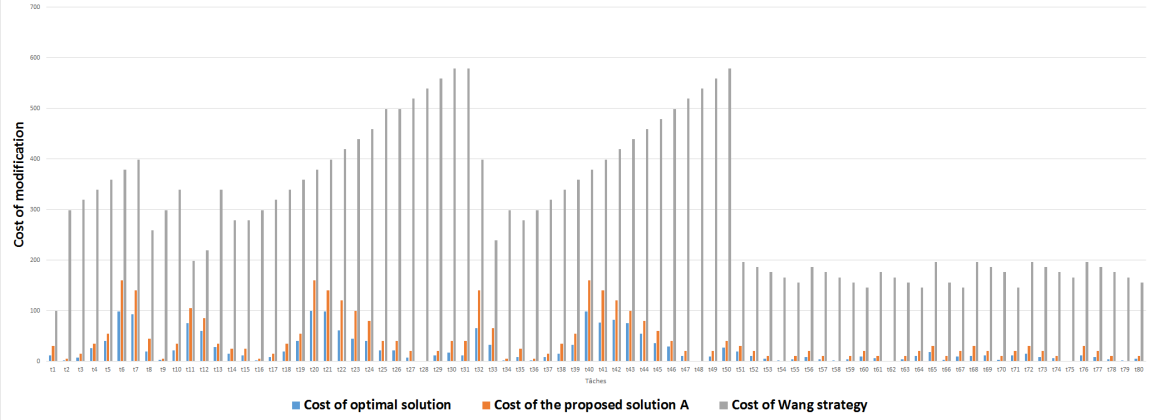

Fig. 4. Modification's cost of periods T_i (Solution A).

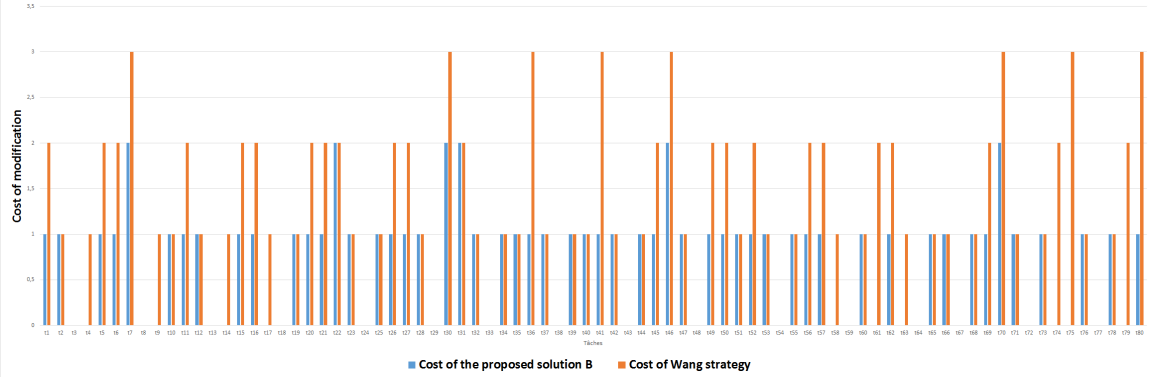

Fig. 5. Modification's cost of WCETs W_i (Solution B).

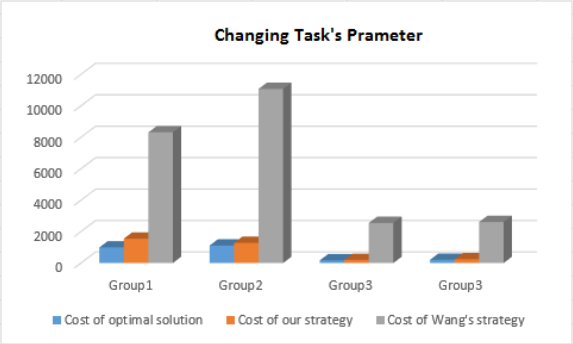

Fig. 6. Cost of parameters' modification.

$$Cost = \begin{array}{c} \begin{array}{cccc} C_1 & C_2 & \cdots & C_p \\ \downarrow & \downarrow & \cdots & \downarrow \end{array} \\ \begin{pmatrix} 0 & 1 & \cdots & p-1 \\ 1 & 0 & \cdots & p-2 \\ \vdots & \ddots & \ddots & \vdots \\ p-1 & \cdots & 1 & 0 \end{pmatrix} \end{array} \begin{array}{l} \\ \leftarrow C_1 \\ \leftarrow C_2 \\ \vdots \\ \leftarrow C_p \end{array}$$

VI. Performance Evaluation

To show the effectiveness of the proposed strategy, we propose to compare it with two EDF-based algorithms: Bhardwaj et al. [33] and Govil et al. [34]. In order to provide a more convincing evaluation, we apply the three approaches to the same case study presented in [33]. The objective is to map all the tasks to the different cores and calculate the total communication cost of each algorithm.
Let us indicate that the cost of the cores connections matrix is shown as follows:

For several sets of input data (N, p), the comparison is shown in a tabular form as well as in a graphical form.

A. Comparison of algorithms by increasing the number of tasks

The comparison is performed using a constant number of cores (4 cores) and an increasing number of tasks (Tab. II). In Fig. 7, the orange curve represents the communication cost by applying the algorithm proposed by Govil et al. [34] and and the blue curve indicates the communication cost after applying the algorithm proposed by Bhardwaj et al. [33]. The cost of communication, when we apply the proposed strategy, is represented by the grey curve.

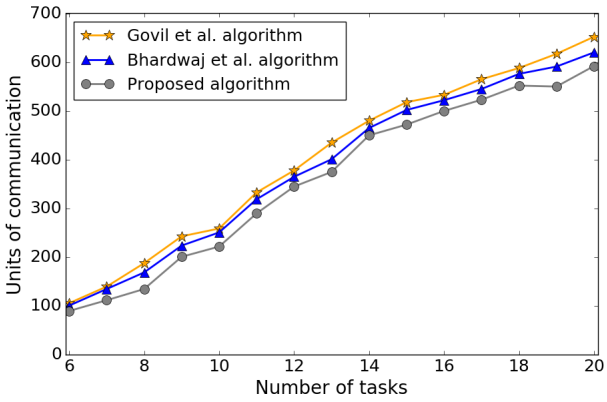

Fig. 7. Communication cost of our strategy when the tasks are in an increasing order and the number of cores is four.

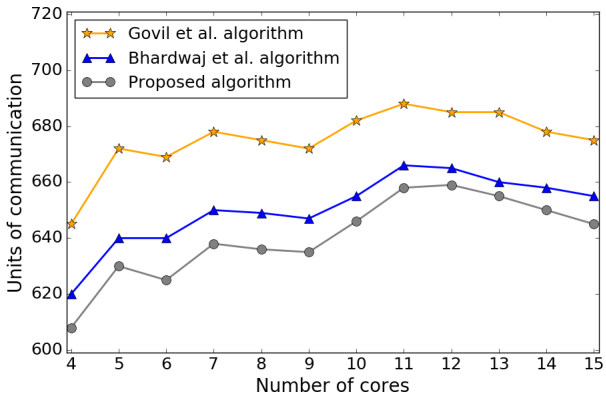

Fig. 8. Communication cost of our strategy when the cores are in an increasing order and the number of tasks is 50.

TABLE II
COMMUNICATION'S COST OF THE THREE ALGORITHMS WHEN THE TASKS ARE IN AN INCREASING ORDER AND THE NUMBER OF CORES IS FOUR.

Tasks N	Cores p	Algo. [33]	Algo. [34]	Our Algo.
6	4	101	106	90
7	4	135	140	112
8	4	169	188	135
9	4	224	243	201
10	4	251	259	222
11	4	319	333	290
12	4	365	378	345
13	4	401	435	375
14	4	465	480	450
15	4	502	518	472
16	4	522	533	500
17	4	545	565	523
18	4	576	588	552
19	4	591	617	550
20	4	620	652	592
Total		5786	6035	5409

The experimental results show that the proposed algorithm offers a gain estimated in average to 10.5% ($\frac{6035-5409}{6035} * 100$) of communication cost compared with [34] and 6.5% of communication cost compared with [33].

B. Comparison of algorithms by increasing the number of cores

In this subsection, the comparison is done with a constant number of tasks (50 tasks) and with an increasing number of cores (Tab. III). By measuring the gain obtained in terms of communication costs, we note that the proposed algorithm offers a little more than 5% ($\frac{8104-7685}{8104} * 100$) of gain compared with the algorithm reported in [34] and 2% ($\frac{7805-7685}{7805} * 100$) of gain compared with the second one reported in [33]. Fig. 8 shows the different communication costs illustrated by Tab. III.

TABLE III
COMMUNICATION COST OF THE THREE ALGORITHMS WHEN THE CORES ARE IN AN INCREASING ORDER AND THE NUMBER OF TASKS IS 50.

Cores p	Tasks N	Algo. [33]	Algo. [34]	Our Algo.
4	50	620	645	608
5	50	640	672	630
6	50	640	669	625
7	50	650	678	638
8	50	649	675	636
9	50	647	672	635
10	50	655	682	646
11	50	666	688	658
12	50	665	685	659
13	50	660	685	655
14	50	658	678	650
15	50	655	675	645
Total		7805	8104	7685

C. Comparison of algorithms in terms of task rejection rate

The application of the three algorithms of task mapping may drive to the rejection of some of them for particular reasons such as the violation of i) real-time constraints, ii) communication constraint, and/or (iii) energy constraints. The objective of this subsection is to calculate the task rejection rate of each algorithm. In order to generalize the performance evaluation of these algorithms, we propose to randomly generate 1000 tasks by using the developed simulator $Task - Generator$ presented in a previous work reported in [3] and calculate the task rejection rate when the number of cores is taken in an increasing order.
Let us indicate that we configure initially the simulator as follows (Tab. IV):

TABLE IV
UNIFORM DISTRIBUTION OF TASK'S PARAMETERS.

Task's parameter	Min	Max
WCET W	1	10
Period T	200	1000
Period max T_{max}	T	2*T

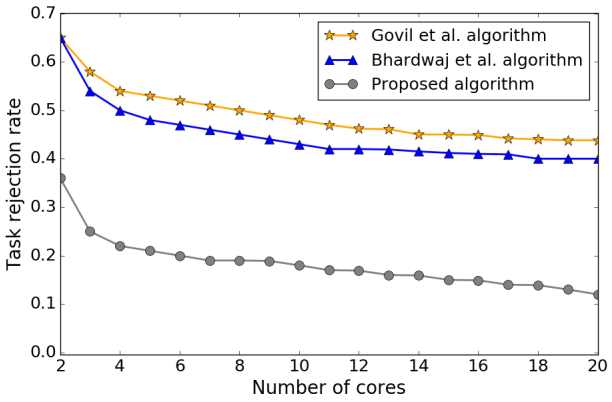

Fig. 9. Evolution of the task rejection rate when the number of cores increases.

Fig. 9 shows the task rejection rate with the number of cores. After observing the results, it is noticed that the proposed algorithm offers a gain estimated in average to 62% of task rejection rate compared with [34] and 59% compared with [33].

To further evaluate the proposed algorithm, we use the $Task-Generator$ simulator and calculate the task rejection rate by increasing the number of tasks and fixing the number of cores equal to four. The idea is to evaluate the proposed algorithm with different loads. Fig. 10 shows that if the number of tasks is less than or equal to 402, then no task will be rejected. Nevertheless, beyond 402 tasks the task rejection rate will be increased. From 1500 tasks, the system rejects almost 48% of tasks. Since the rejection rate is dependent on the tasks' parameters, then it is varied for each set of tasks which are randomly generated. That's why Fig. 10 presents for each task's number, an interval of rejection rate.

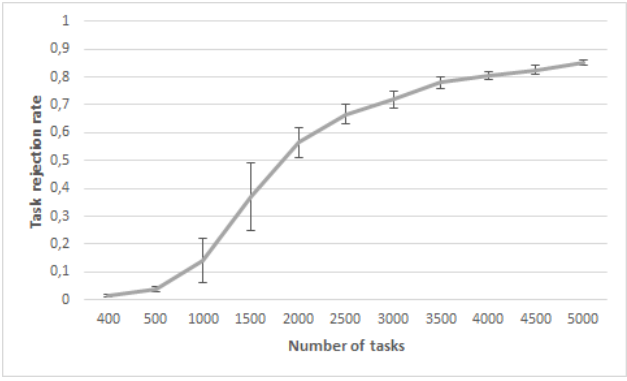

Fig. 10. Task rejection rate when the tasks are in an increasing order and the number of cores is fixed to four.

Concerning the evaluation of the elapsed execution time of the proposed solution, we estimate it when i) the number of tasks is increased from 50 to 500, and ii) the number of cores is fixed to four. Fig. 11 presents the elapsed time to map and construct the packs of tasks. The used processor is an Intel Core 2 Duo (1.3GHz) with 4Go of RAM.

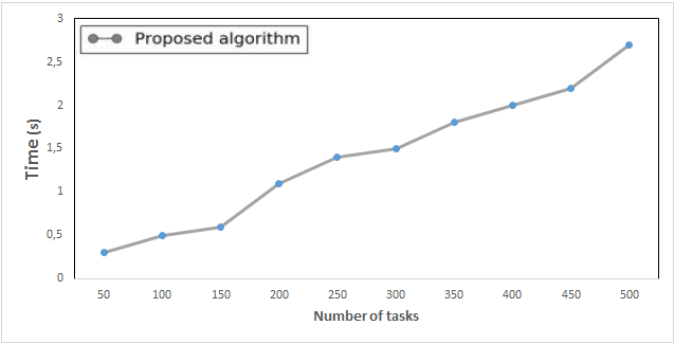

Fig. 11. Elapsed time to map and group the tasks on packs.

Thanks to the grouping of tasks in packs, the placement becomes easier since the tasks of the pack can be mapped to the same core. Consequently, the mapping time of tasks is minimized and acceptable.

VII. Discussion

Low-power scheduling of $RTSys$ after reconfiguration scenarios is the most significant contribution of this study. By dynamically adjusting the tasks parameters, the published research strategies [4], [6], [26] can reduce the power consumption of $RTSys$ to keep the energy until the next recharge. However, they fail to reduce the power consumption and satisfy the feasibility of an $RTSys$ with a high-power caused by task addition scenarios. The contributions of this research are summarized as follows: 1) Proposition of the $OptimalMappingTasks$ tool which seeks the optimal placement of tasks on cores after any reconfiguration scenario in order to minimize the traffic on the NoC by trying to place the dependent tasks as close as possible, 2) Proposition of scheduling solutions grouped in a strategy to re-obtain the feasibility of $RTSys$ after the tasks' mapping by satisfying the real-time, communication and energy constraints.

VIII. Conclusion

In this paper, we first presented a general methodology for modelling the power consumption of periodic tasks on rechargeable homogeneous multi-core architectures. Being reconfigurable, they can violate some constraints such as real-time and/or energy and/or communication. To guarantee that between two recharges the energy is sufficient to execute all the tasks without missing their deadlines, we propose four solutions to modify some parameters of tasks and messages in order to decrease the processor utilization and to re-obtain the system feasibility. If the system is still unfeasible, then we propose a fifth solution that removes some tasks according to their importance factor I_i. All these heuristics are manipulated by a new deterministic strategy. This strategy is based on a pack classification such that after each reconfiguration scenario, suitable and acceptable modifications are performed on the parameters of the packs and their related tasks. We plan in the future work to implement the proposed strategy in an RTOS that will be assessed by considering real case studies. In this paper, the platform is considered homogeneous. Nevertheless the consideration of an extended strategy for battery-powered heterogeneous platforms is necessary since

these are found in diverse applications, such that distributed wireless sensor nodes, telecommunications, etc.

REFERENCES

[1] A. Gammoudi, A. Benzina, M. Khalgui and D. Chillet, "New Pack Oriented Solutions for Energy-Aware Feasible Adaptive Real-Time Systems," in *Proc. 14th Int. Conf. Intell. Softw. Methodologies Tools Techn.*, Naples, Italy, 2015, pp. 73–86.

[2] A. Gammoudi, A. Benzina, M. Khalgui and D. Chillet, "New Reconfigurable Middleware for Adaptive RTOS in Ubiquitous Devices," in *Proc. 10th Int. Conf. Mobile Ubiquitous Comput. Syst. Services and Techn.*, Venice, Italy, 2016, pp. 131–137.

[3] A. Gammoudi, A. Benzina, M. Khalgui and D. Chillet, "Reconf-Pack: A Simulator for Reconfigurable Battery-Powered Real-Time Systems," in *Proc. 30th Europ Mod. Conf.*, University of Las Palmas. Spain, 2016.

[4] A. Gammoudi, A. Benzina, M. Khalgui and D. Chillet, "Real-Time Scheduling of Reconfigurable Battery-Powered Multi-Core Platforms," in *Proc. 28th Int. Conf. Tools Artif. Intell.*, San Jose, USA, 2016.

[5] X. Wang, Z. W. Li and W. M. Wonham, "Dynamic multiple-period reconfiguration of real-time scheduling based on timed DES supervisory control," *IEEE Trans. Ind. Inform.*, vol. 12, no. 1, pp. 101–111, 2016.

[6] X. Wang, I. Khemaissia, M. Khalgui, Z. W. Li, O. Mosbahi and M. C. Zhou, "Dynamic low-power reconfiguration of real-time systems with periodic and probabilistic tasks," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 1, pp. 258–271, 2015.

[7] H. E. Houssein and M. A. E. Hadi, "Energy efficient scheduler of aperiodic jobs for real-time embedded systems," *Int. J. Autom. Comput.*, pp. 1–11, 2016.

[8] I. R. Quadri, A. Gamati, P. Boulet, S. Meftali and J. L. Dekeyser, "Expressing embedded systems configurations at high abstraction levels with UML MARTE profile: Advantages, limitations and alternatives," *J. Syst. Archit.*, vol. 58, no. 5, pp. 178–194, 2012.

[9] H. Gharsellaoui, M. Khalgui and S. B. Ahmed, "Reconfiguration of Synchronous Real-Time Operating System," *Int. J. Syst. Dynamics App.*, vol. 2, no. 1, pp. 114–132, 2013.

[10] C. C. Lin, Y. C. Syu, C. J. Chang, J. J. Wu, P. Liu, P. W. Cheng and W. T. Hsu, "Energy-efficient task scheduling for multi-core platforms with per-core DVFS," *J. Parallel Distrib. Comput.*, vol. 86, no. 1, pp. 71–81, 2015.

[11] C. Jalier, D. Lattard, A. A. Jerraya, G. Sassatelli, P. Benoit and L. Terros, "Heterogeneous vs homogeneous MPSoC approaches for a mobile LTE modem," in *Proc. Conf. Des. Autom. Test. Eur.*, Dresden, Germany, 2010, pp. 184–189.

[12] M. Marinoni and G. Buttazzo, "Elastic DVS management in processors with discrete voltage/frequency modes," *IEEE Trans. Ind. Inform.*, vol. 3, no. 1, pp. 51–62, 2007.

[13] S. R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh and T. Jacob "An 80-tile sub-100-w teraflops processor in 65-nm cmos," *IEEE Trans. Ind. Inform.*, vol. 43, no. 1, pp. 29–41, 2008.

[14] J. Cecilio and P. Furtado, "Architecture for uniform (re) configuration and processing over embedded sensor and actuator networks," *IEEE Trans. Ind. Inform.*, vol. 10, no. 1, pp. 53–60, 2014.

[15] L. Li, S. Shancang and S. Zhao "QoS-aware scheduling of services-oriented internet of things," *IEEE Trans. Ind. Inform.*, vol. 10, no. 2, pp. 1497–1505, 2014.

[16] V. Pavel and V. Marik, "Capabilities of dynamic reconfiguration of multiagent-based industrial control systems," *IEEE Trans. Syst., Man, Cybern.*, vol. 40, no. 2, pp. 213–223, 2010.

[17] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM.*, vol. 20, no. 1, pp. 46–61, 1973.

[18] B. D. Bui, R. Pellizzoni and M. Caccamo, "Real-time scheduling of concurrent transactions in multidomain ring buses," *IEEE Trans. Comput.*, vol. 61, no. 9, pp. 1311–1324, 2012.

[19] G. Quan and S. X. Hu, "Minimal energy fixed-priority scheduling for variable voltage processors," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 8, pp. 1062–1071, 2003.

[20] I. Khemaissia, O. Mosbahi and M. Khalgui, "Reconfigurable CAN in real-time embedded platforms," in *Proc. 11th Int. Conf. Informat. Control. Autom. Robot.*, Vienna, Austria, 2014, pp. 355–362.

[21] S. Baruah and J. Goossens, "Scheduling real-time tasks: Algorithms and complexity," in *Handbook of scheduling: Algorithms, models, and performance analysis*, vol. 22, 2004.

[22] I. Khemaissia, O. Mosbahi, M. Khalgui and Z. W. Li, "New Methodology for Feasible Reconfigurable Real-Time Network-on-Chip NoC," in *Proc. 11th Int. Conf. Softw. Eng. App.*, Lisbon, Portugal, 2016.

[23] G. C.Buttazzo, G. Lipari and L. Abeni, "Elastic task model for adaptive rate control," in *Proc. 19th Symp. Real-time Syst.*, 1998, pp. 286–295.

[24] N. R. Martijn, C. Sunder, T. Strasser, A. Zoitl, O. Hummer and G. Ebenhofer, "Zero downtime reconfiguration of distributed automation systems: The εcedac approach," in *Holonic and Multi-Agent Systems for Manufacturing*, 2007, pp. 326–337.

[25] CPLEX, "IBM ILOG CPLEX Optimize," in *http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud/*, 2013.

[26] T. Yokoyama, G. Zeng, H. Tomiyama and H. Takada, "Static task scheduling algorithms based on greedy heuristics for battery-powered DVS systems," *IEICE trans. Inform. syst.*, vol. 93, no. 10, pp. 2737–2746, 2010.

[27] J. Hu and R. Marculescu, "Energy-and performance-aware mapping for regular NoC architectures," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 4, pp. 551–562, 2005.

[28] H. Grichi, O. Mosbahi, M. Khalgui, and Z. Li, "New Extended Object Constraint Language for Adaptive Discrete Event Systems with Application to Recongurable Wireless Sensor Networks," *IEEE Trans. Syst. Man. Cybern. Syst.* to be published in 2017.

[29] S. Meskina, N. Doggaz, M. Khalgui, and Z. Li, "Multiagent framework for smart grids recovery," *IEEE Trans. Syst. Man. Cybern. Syst.*, vol. 47, no. 7, pp. 1284–1300, 2017.

[30] H. Grichi, O. Mosbahi, M. Khalgui, and Z. Li, "RWiN: new methodology for the development of reconfigurable WSN," *IEEE Trans. Autom. Sci. Eng.* , vol. 14, no. 1, pp. 109–125, 2017.

[31] J. Zhang, M. Khalgui, Z. Li, O. Mosbahi, and Abdulrahman. M. Al-Ahmar, "R-TNCES: A novel formalism for reconfigurable discrete event control systems," *IEEE Trans. Syst. Man. Cybern. Syst.*, vol. 43, no. 4, pp. 757–772, 2013.

[32] I. Ghribi, R. B. Abdallah, M. Khalgui, Z. Li, and K. Alnowibet, and M. Platzner, "R-Codesign: Codesign Methodology for Real-time Reconfigurable Embedded Systems Under Energy Constraints," *IEEE Access*, vol. PP, no. 99, 2018.

[33] P. Bhardwaj and V. Kumar, "An Effective Load Balancing Task Allocation Algorithm using Task Clustering," *Inter. J. Comput. Appl*, vol. 77, no. 7, 2013.

[34] K. Govil, 'A smart algorithm for dynamic task allocation for distributed processing environment," *Inter. J. Comput. Appl*, vol. 28, no. 2, pp.13–19, 2011.