Mapping of Periodic Tasks in Reconfigurable Heterogeneous Multi-core Platforms

Aymen Gammoudi[1,2,3], Daniel Chillet[1], Mohamed Khalgui[2,4] and Adel Benzina[2,3]
[1]*IRISA Lab, University of Rennes1, Rennes, France*
[2]*LISI Lab, University of Carthage, Tunis, Tunisia*
[3]*Tunisia Polytechnic School, University of Carthage, Tunis, Tunisia*
[4]*School of Electrical and Information Engineering, Jinan University (Zhuhai Campus), Zhuhai 519070, China*

Keywords: Heterogeneous Multi-core Platform, Reconfiguration, Task Mapping and Scheduling.

Abstract: *Multi-core Real-time Systems* (*MRS*) powered by a battery have been adopted for a wide range of high performance applications, such as mobile communication and automotive systems. A system is composed of *N* dependent and periodic *Operating System* (OS) tasks to be assigned to *p* heterogeneous cores linked by a network-on-chip (NoC). This paper deals with the problem of task allocation in *MRS* in such a way that the cost of communication between cores is minimized by trying to place the dependent tasks as close as possible to each other. The main objective is to develop a new strategy for allocating *N* tasks to *p* cores of a given distributed system using task clustering by considering both the cost of inter task communication and that of communication between cores. The proposed strategy guarantees that, when a task is mapped into the system and accepted, then it is correctly executed prior to the task deadline. A novel periodic task model based on elastic coefficients is proposed to compute useful temporal parameters allowing to assign all tasks to *p* cores, by minimizing the traffic between cores. Experimental results reveal the effectiveness of the proposed strategy by comparing the derived solutions with the optimal ones, obtained by solving an *Integer Linear Program* (ILP).

1 INTRODUCTION

Multi-core real-time platforms typically are composed of multiple processors (processing units), memories, and a communication infrastructure. Heterogeneous multi-core platforms contain different types of processing units. Therefore, the system designers can take advantage of their properties when mapping tasks to specific processor types and optimize criteria such as computational performance, cost and energy consumption (Schranzhofer et al., 2010). Several academic and industrial studies reported in (Wang et al., 2016), (Cecilio and Furtado, 2014), (Li et al., 2014) have addressed the dynamic reconfiguration of real-time systems. These approaches can be divided into two categories: manual applied by users (Rooker et al., 2007); and automatic applied by intelligent control agents (Vrba and Marik, 2010). The most promising approach is the elastic scheduling reported in (Wang et al., 2016), (Wang et al., 2015), (Marinoni and Buttazzo, 2007), (Buttazzo et al., 1998).

A multi-core real-time system, denoted *MRS*, can be implemented by *N* reconfigurable real-time dependent and periodic tasks to be assigned to *p* heterogeneous cores linked by a network-on-chip (NoC) (Hu and Marculescu, 2005) in order to ensure the communication between dependent tasks. Since we consider a heterogeneous platform, each task cannot be executed by all cores, it can be supported only by one or more specific cores according to system specification. Similarly to the research works reported in (Wang et al., 2016), (Marinoni and Buttazzo, 2007), we consider a more flexible task model, in which the tasks can operate within a given range of periods with different performances. A reconfiguration scenario is defined in this paper as any internal or external event that leads to the addition and/or removal of periodic tasks as well as their exchanged messages to adapt the system's behavior to its environment (Quadri et al., 2012). In this paper, we are interested basically in reconfiguration scenarios that add new software tasks. New tasks must be assigned to their appropriate cores by trying to place the depen-

Gammoudi, A., Chillet, D., Khalgui, M. and Benzina, A.
Mapping of Periodic Tasks in Reconfigurable Heterogeneous Multi-core Platforms.
DOI: 10.5220/0006698500990110
In *Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2018)*, pages 99-110
ISBN: 978-989-758-300-1

dent tasks as close as possible to each other in order to reduce the traffic on the NoC. Therefore the energy consumption, proportional to the data communication volume, will be decreased. As partitioned scheduling converts the problem of scheduling real-time tasks into a set of uniprocessor scheduling problems (Baruah and Goossens, 2004), we decided to select earliest deadline first (EDF) scheduler to schedule the local tasks for each core.

For two new added dependent tasks assigned to different cores, a new periodic message is added automatically to the NoC. Consequently, the real-time constraints can be not satisfied when a task misses the related deadline and a message can take a long time to arrive to its destination. Therefore, the system *MRS* will be unfeasible. An *MRS* is feasible if and only if it satisfies two constraints: (i) the deadlines of the tasks, and (ii) the messages' deadlines on the communication medium. Moreover, the provided solutions must be optimal real-time ones that minimize the power consumption of system.

The general goal of this paper is to propose a new mapping strategy based on grouping heavily communicating tasks into the same cluster. The tasks that are grouped into the same cluster are assigned to the same core while keeping its processor utilization coefficient less than or equal to 1 and the communication bus is feasible. Whenever these constraints are not satisfied, the proposed strategy offers two solutions (A and B) based on the modification of temporal parameters of the processed tasks to meet the real-time and communication constraints. If after assigning the new task to core C_j, the real-time constraint is violated, then the proposed strategy starts by modifying the periods of tasks and the exchanged messages until reaching their specific maximal periods (Solution A) in order to decrease the processor utilization of C_j. If the second constraint is also violated after assigning one or more new tasks, then the proposed strategy applies the second solution (Solution B) that modifies the message's period and the associated tasks to slow down the communicating tasks to satisfy the communication constraint. To evaluate this strategy, we formalize the problem as an optimization problem by using integer linear programming (ILP) and compare the proposed solutions with the optimal results provided by the CPLEX solver (CPLEX, 2013). The originality of this paper is to propose a new strategy of mapping and scheduling tasks in reconfigurable architectures. The proposed strategy looks for the near optimal placement of tasks on cores after any reconfiguration scenario in order to minimize the traffic on the NoC. The proposed strategy is deterministic since it is able to find the suitable solutions for each situation.

The rest of this paper is organized as follows: Section 2 reviews some related works. Section 3 presents the system model and the used terminologies. We formalize the considered problem in Section 4. The proposed strategy is presented in Section 5 that deals with the modification of temporal parameters of the processed tasks to guarantee the related real-time and communication constraints after any reconfiguration scenario. The strategy is implemented, simulated and analyzed in Section 6. Finally, Section 7 concludes this work.

2 RELATED WORK

Reducing the cost of communication between cores becomes a major concern for the high-performance and reliability of such systems. Several academic studies reported in (Carle et al., 2015), (Bhardwaj and Kumar, 2013), (Tosun et al., 2009) have addressed the mapping algorithms using task clustering by taking both inter task communication and execution costs. The basic idea of clustering based algorithm in (Bhardwaj and Kumar, 2013), (Tosun et al., 2009) 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 in an effort to avoid communication costs. These algorithms are interesting since the communication cost is reduced, but the authors are not interested in the verification of the processor utilization after assigning the tasks. In certain cases, the processor utilization exceeds 100%. In addition, they are not interested to check the overflow on the communication medium. To resolve these two problems, effective solutions based on the modification of WCETs, deadlines, and periods of tasks in order to verify the processor utilization after any assignment of tasks are reported in (Gammoudi et al., 2015), (Wang et al., 2016), (Wang et al., 2015), (Gharsellaoui et al., 2013), (Marinoni and Buttazzo, 2007).

The work presented in (Wang et al., 2015) 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 prolong 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 (Gammoudi et al., 2015) improves these solutions by proposing a new approach based on the modification of task parameters with classification in packs by assigning a unique period

(or WCET) to all the tasks related to the same pack. Each pack is a group of tasks having "similar" periods (or WCETs). For each reconfiguration scenario, specific modifications are performed on the parameters of the packs and their related tasks in order to meet the real-time and energy constraints. In the research work reported in (Gammoudi et al., 2016c), the authors develop the $Reconf-Pack$ simulator to evaluate and compare these two approaches by generating a large number of random systems and reconfiguration scenarios for each one. The authors show that the total cost introduced by applying the solution presented in (Wang et al., 2015) in 88% of random cases is largely higher than that introduced by the method reported in (Gammoudi et al., 2015). To show that the pack solution, proposed in (Gammoudi et al., 2015), can be implemented in a real application, the work reported in (Gammoudi et al., 2016a) 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 (Gammoudi et al., 2015) to homogeneous multi-core platforms is described in (Gammoudi et al., 2016b). The authors propose four solutions to schedule a real-time application composed of independent periodic tasks under energy constraints. Nevertheless, they do not consider the scheduling of dependent tasks onto heterogeneous platforms and they do not treat the influence of the task mapping on the communication and therefore on the energy consumption.

3 SYSTEM MODEL AND ASSUMPTIONS

In this section, we present the task model and the platform architecture that consists of a reconfigurable *MRS* composed of N periodic OS tasks executed by p heterogeneous cores. We also present the models of the processor architecture and the communication between cores.

3.1 Task Model

We suppose that *MRS* processes a task set Π containing N periodic tasks, i.e., $\Pi = \{\tau_1, \tau_2, ..., \tau_N\}$. Some tasks are considered as flexible, whose utilization can be modified by changing their periods within a specified range. According to (Wang et al., 2016), (Marinoni and Buttazzo, 2007), (Buttazzo et al., 1998), a periodic task τ_i is characterized by: (i) a release time A_i, (ii) a worst-case execution time (WCET) W_i, (iii) a relative deadline D_i, (iv) a period T_i, and (v) a maximum tolerable period $T_{i_{max}}$. The relative deadline of a periodic task is considered as a hard deadline if its missing is unacceptable (Baruah and Goossens, 2004). According to (Liu and Layland, 1973), assuming that the period is equal to the deadline for each task, the tasks are schedulable if and only if the processor utilization coefficient is less than 1. Hence, we apply in this paper the EDF policy to schedule the local tasks for each core.

3.2 Platform Model

Let the considered *MRS* consists of a set of p heterogeneous cores $\Gamma=\{C_1, C_2, ..., C_p\}$, interconnected by communication links. Since we consider heterogeneous platforms, we suppose that each task τ_i can be executed by one or more specific cores according to system specification. Hence, we define the matrix γ that represents the different possibilities to execute tasks on cores.

$$\gamma_{i,j} = \begin{cases} 1 & \tau_i \text{ can be executed by } C_j. \\ 0 & \text{otherwise.} \end{cases}$$

In order to ensure that all the tasks are mapped on the cores, we define the mapping matrix H where

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

3.3 Communication Model

Each core runs periodic OS tasks which can exchange messages on the NoC (Khemaissia et al., 2016), (Hu and Marculescu, 2005). 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 (Bui et al., 2012) the communication model is based on: (i) a regular inter-arrival time $TM_{i,j}$, (ii) a spent time to transmit a message $WM_{i,j}$, (iii) an absolute deadline $DM_{i,j}$, and (iv) an amount of exchanged data $SM_{i,j}$ (in bits). We consider that the period $TM_{i,j}$ of message $M_{i,j}$ is equal to the period T_i of the related sending task τ_i ($TM_{i,j}$=T_i). If the task's period is changed, then the periods of the related messages 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 called $Cost$ 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 the cost (e.g. energy) of sending 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 (in units of cost) is given by

$$TotalCost = \sum_{k=1}^{p}\sum_{l=1}^{p}\sum_{i=1}^{N}\sum_{j=1}^{N} SM_{i,j} * Cost_{C_k,C_l} * H_{i,k} * H_{j,l} \quad (1)$$

We suppose, in this paper, that the energy consumed by the communication is proportional to the distance between cores.

3.4 Processor Utilization Model

According to (Liu and Layland, 1973) the processor utilization U_{C_j} of core C_j is given by

$$U_{C_j} = \sum_{i=1}^{N} \frac{W_i}{T_i} * H_{i,j}; \forall j \in [1..p] \quad (2)$$

The core C_j is feasible, if and only if

$$U_{C_j} \leq 1; \forall j \in [1..p] \quad (3)$$

Eq. 3 is the real-time constraint.
According to the works reported in (Bui et al., 2012), (Khemaissia et al., 2014), (Khemaissia et al., 2016), 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

$$U_{Com}(C_k,C_l) = \sum_{i=1}^{N}\sum_{j=1}^{N} \frac{WM_{i,j}}{TM_{i,j}} * H_{i,k} * H_{j,l}; \forall k,l \in [1..p] \quad (4)$$

According to the work reported in (Khemaissia et al., 2016), to ensure that the communication between cores that are physically close is feasible, it is necessary to check

$$U_{Com}(C_k,C_l) \leq 1; \forall k,l \in [1..p] \mid Cost_{C_k,C_l} = 1 \quad (5)$$

Eq. 5 is the communication constraint.

4 PROBLEM FORMULATION

The problem, being addressed in this paper, is concerned with an optimal allocation of the tasks to the cores after any reconfiguration scenario. An optimal allocation is considered as one that minimizes the cost of communication between cores such that the real-time and communication constraints are satisfied. We present, in this section, the formulation of the mapping problem and formalize each constraint as an optimization problem by using ILP.

4.1 Mapping Problem

We suppose that at time t_i *MRS* is composed of $\Gamma(t_i) = \{C_1, C_2, ..., C_p\}$ and $\Pi(t_i) = \{\tau_1, \tau_2, ..., \tau_{n_1}\}$. An *MRS* is feasible if and only if it satisfies the two constraints (real-time and communication constraints). We assume in the following that *MRS* is dynamically reconfigured at time t_k ($t_k > t_i$) by adding n_2 tasks. Therefore, the new implementation of tasks is $\Pi(t_k) = \{\tau_1, \tau_2, ..., \tau_{n_1}, \tau_{n_1+1}, ..., \tau_N\}$, with $N = n_1 + n_2$. The new tasks provided by this reconfiguration must be mapped to their appropriate cores.
We consider, in this paper, that the initial mapping at time t_1 is also a reconfiguration scenario such that $\Pi(t_0) = \{\varnothing\}$ and $\Pi(t_1) \neq \{\varnothing\}$.
To guarantee that each task is executed at most by one of its appropriate cores, it is necessary to satisfy

$$\sum_{j=1}^{p} H_{i,j} * \gamma_{i,j} = 1; \forall i \in [1..N] \quad (6)$$

We need to assign all the old/new tasks to their appropriate cores by minimizing the cost of communication between them. This idea is formalized by

$$RE \begin{cases} Minimize\ TotalCost \\ s.t. \\ U_{C_j} \leq 1, \forall\ j \in [1..p] \quad (Eq.3) \\ U_{Com}(C_k,C_l) \leq 1 \mid Cost_{C_k,C_l} = 1 \ (Eq.5) \\ \forall\ k,l \in [1..p] \\ \sum_{j=1}^{p} H_{i,j} * \gamma_{i,j} = 1, \forall\ i \in [1..N] \quad (Eq.6) \end{cases}$$

where, *TotalCost* is calculated by $Eq.1$. $Eq.3$ satisfies the real-time scheduling under the EDF policy for each core, $Eq.5$ ensures that the communication between the cores C_k and C_l is feasible and $Eq.6$ ensures that each task is executed at most by one of its appropriate cores. The *RE* problem can be solved by integer linear programming solvers such as CPLEX (CPLEX, 2013), 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. We suppose that ILP is only used to compute the optimal solution to have a comparison reference for the heuristic we will propose later. After different reconfiguration scenarios, one or more of these constraints can be violated and ILP cannot provide a solution for that situation.

4.2 Real-time Problem

After assigning one or more tasks on C_j, the processor utilization of C_j will increase and it can be unfeasible ($U_{C_j} > 1$). Since $U_{C_j} = \sum_{i=1}^{N} \frac{W_i}{T_i} * H_{i,j}; \forall j \in [1..p]$, we propose first of all to modify the periods of the local OS tasks in order to decrease the processor utilization. The idea is to extend the periods of tasks until they reach $T_{i_{max}}$. Using the ILP model, we formalize this problem by $\forall\ j \in [1..p] \mid U_{C_j} > 1$

$$Pb1 = \begin{cases} Minimize \sum_{i=1}^{N} \Delta T_i * H_{i,j} \\ s.t. \\ \sum_{i=1}^{N} \frac{W_i}{T_i + \Delta T_i} * H_{i,j} \leq 1 \\ T_i + \Delta T_i \leq T_{i_{max}}, \forall\ i \in [1..N] \end{cases}$$

where ΔT_i is an integer that extends the period of the real-time tasks. This formalization exploits the possible flexibility of the periods in order to reduce the processor utilization.

4.3 Communication Problem

After the addition of a pair of dependent tasks on two cores, 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. 5, 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. The idea here is to slow down the communicating tasks to satisfy the communication constraint. We can formalize this problem by $\forall\ k,l \in [1..p] \mid U_{Com}(C_k,C_l) > 1$

$$Pb2 = \begin{cases} Minimize \sum_{i=1}^{N} \sum_{j=1}^{N} \Delta TM_{i,j} * H_{i,k} * H_{i,l} \\ s.t. \\ \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{WM_{i,j}}{TM_{i,j} + \Delta TM_{i,j}} * H_{i,k} * H_{i,l} \leq 1 \\ TM_{i,j} + \Delta TM_{i,j} \leq T_{i_{max}}, \forall\ i,j \in [1..N] \end{cases}$$

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

*Pb*1 and *Pb*2 can be solved by CPLEX (CPLEX, 2013) 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. Therefore, we present in the next Section some efficient and implementable heuristics and we compare them to the optimal solutions provided by the CPLEX solver to solve these problems.

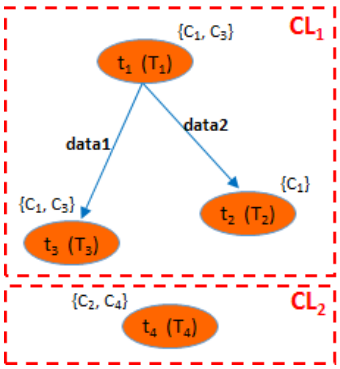

Figure 1: Example of application.

5 CONTRIBUTION

When the configuration (or reconfiguration) scenario is applied, each new task must be assigned to its appropriate core by minimizing the cost of communication. We present in the next subsection a mapping strategy based on grouping heavily communicating tasks into the same cluster. The tasks that are grouped into the same cluster are assigned to the same core such that its processor utilization coefficient is less than or equal to 1 and the communication bus is feasible.

5.1 Proposed Task Allocation Algorithm

Before presenting the allocation algorithm, we introduce some definitions:

SingleCore Task: any task that can be executed only by one specific core. *Example* : τ_2 in Fig. 1.

Cluster (CL): a group of communicating tasks. *Example* : τ_1, τ_2 and τ_3 are grouped in CL_1 (Fig. 1).

Common Processor (CP): a processor that can execute all tasks of a cluster *CL* such that its processor utilization is less than or equal to 1. *Example* : C_1 can be the *CP* of CL_1. C_2 and/or C_4 can be the *CP* of CL_2 (Fig. 1).

CPCL: we denote by *CPCL* each cluster *CL* that has a common processor *CP*.

Data Traffic (TD): is the total volume of data exchanged between tasks in *CL*. *Example* : $TD(CL_1) =$

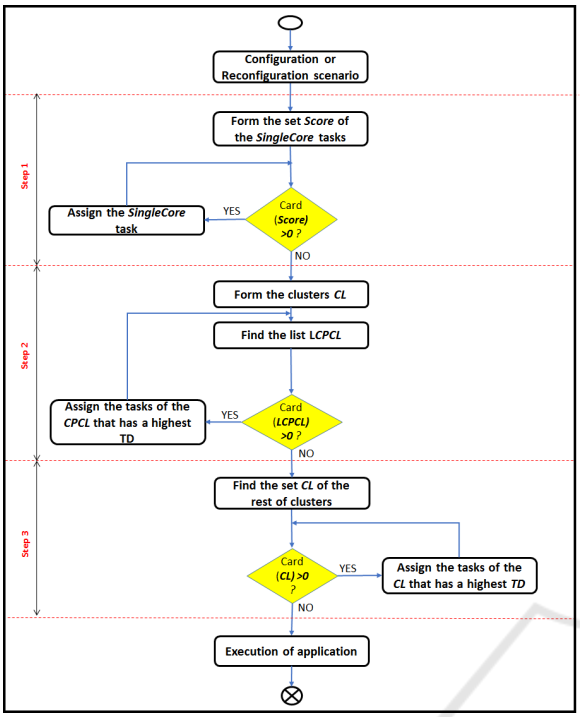

Figure 2: Global *MappingTasks* algorithm.

$TU * (\frac{data1 + data2}{T_1})$ and $TD(CL_2) = 0$ (Fig. 1).

The global structure of the proposed strategy is presented in Fig. 2 and is based on three main steps:

Step 1: Assign all the *SingleCore* tasks.

Step 2: Assigning the communicating tasks that have a common processor *CP*:

Step 2.1: Each group of communicating tasks are grouped in a cluster *CL*.

Step 2.2: Calculate the data traffic (*TD*) for each *CL*.

Step 2.3: Sort all the *CL* in a descending order according to their *TD*.

Step 2.4: A *CL* that has a *CP* will be denoted by *CPCL*. The list of all *CPCLs* called *LCPCL*.

Step 2.5: Assign the tasks of each *CPCL*.

Step 3: Assigning the tasks of *CL* as close as possible.

Step 3.1: Take the *CL* that has a highest *TD*.

Step 3.2: Place each task on the best position (in term of communication cost).

The detailed structure of the proposed strategy is presented in Fig. 3. Its objective is to find the near 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. The proposed strategy solves the problem formalized by *RE* in sub-section 4.1. As shown in this figure, the proposed strategy takes the task's characteristics, the matrix γ, the matrix of architecture *Cost* and the size of exchanged messages as inputs and outputs the mapping of the tasks on the architecture.

After assigning the new tasks, the processor utilization of certain cores will certainly increase. If the new utilization is greater than 1, then the real-time constraint is violated as reported in (Wang et al., 2015), (Liu and Layland, 1973). Besides, the communication constraint can be violated after the addition of tasks and consequently *MRS* will be unfeasible. To solve these two problems that have been formalized in 4.2 and 4.3, we present two solutions in the next subsection.

5.2 Task Scheduling for Feasible Reconfigurable Platforms

We propose two heuristics based on grouping the tasks that have "similar" periods in several packs, denoted *Pack*, to re-obtain the system feasibility. After a reconfiguration, we do not calculate a new period for each task, but assign a new one period to all tasks of the first pack $Pack_{1,j}$ ($j \in [1..p]$). Moreover, all the new periods affected to the tasks of pack $Pack_{x,j}$ are multiple of the one affected to the tasks of $Pk_{1,j}$. Hence, we have only to compute the suitable value $Pk_{1,j}$. We show in the following solutions how the tasks are grouped in packs.

5.2.1 Solution A: Modification of Periods under Real-Time Constraint

If C_j ($j \in [1..p]$) is not feasible after assigning some new tasks to it, 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 pack $Pack^T_{x,j}$ according to its period. To construct the packs, we need to find the suitable new period $Pk^T_{1,j}$ that minimizes the cost of the new solution for the whole system. We formalize this problem by

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

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

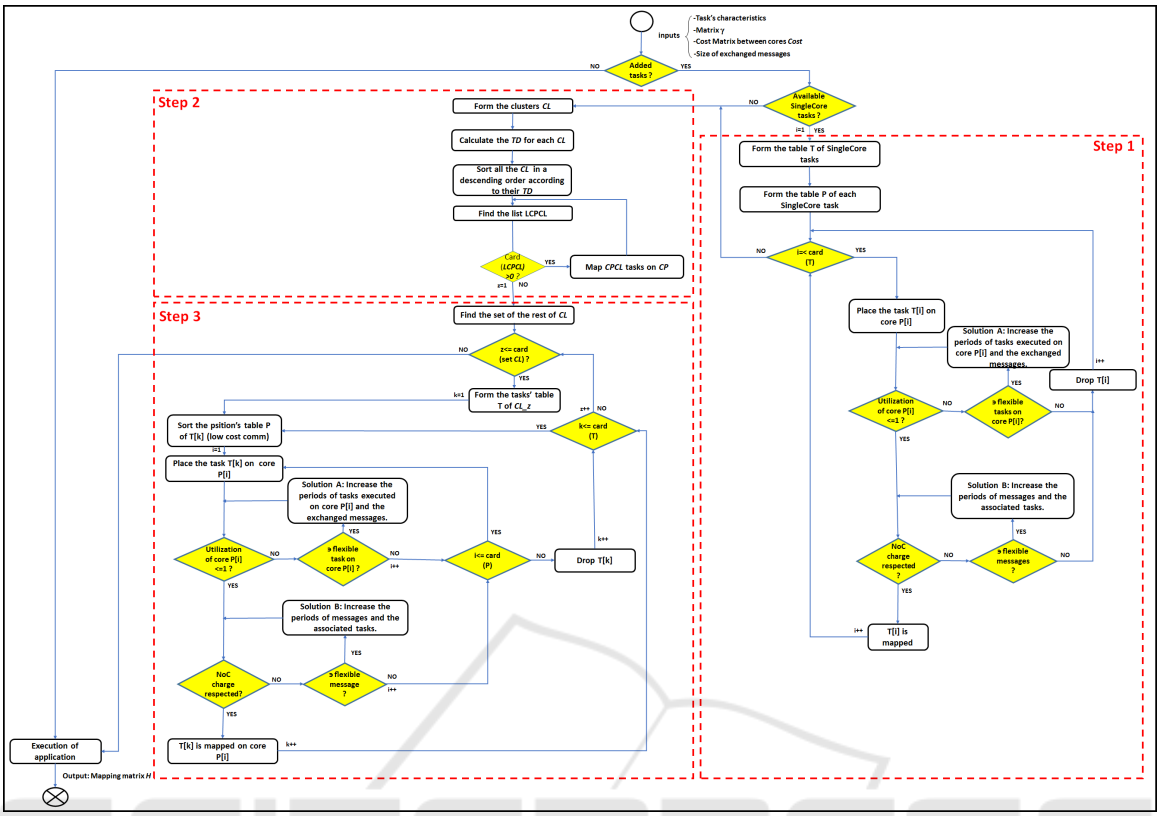

Figure 3: *MappingTasks* algorithm.

incremented. Hence, $Pack^T_{1,j}$ includes the tasks that have a period in the range $[1 \;;\; Pk^T_{1,j}]$, $Pack^T_{2,j}$ includes the tasks that have a period in the range $[Pk^T_{1,j}+1 \;;\; 2*Pk^T_{1,j}]$, etc.

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

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

thus,

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

then,

$$Pk^T_{1,j} \geq \sum_{\tau_i \in Pack^T_{1,j}} W_i + \ldots + \sum_{\tau_i \in Pack^T_{x,j}} \frac{W_i}{x}$$

Since the periods are integer values, then,

$$Pk^T_{1,j} = \left\lceil \sum_{\tau_i \in Pack^T_{1,j}} W_i + \ldots + \sum_{\tau_i \in Pack^T_{x,j}} \frac{W_i}{x} \right\rceil \quad (7)$$

where $Pk^T_{1,j}$ is the new period affected to tasks of the first pack $Pack^T_{1,j}$, $x*Pk^T_{1,j}$ to the tasks of x^{th} pack $Pack^T_{x,j}$. 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 3.3, 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.

5.2.2 Solution B: Modification of Message's Period under Communication Constraint

For two added dependent tasks assigned to different processors, a message is added automatically on the NoC, we are interested in this part in the communication feasibility. 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 exchanged messages 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

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

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

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 $[(x-1) * Pk_{x,k,l}^{TM} + 1 \ ; \ x * Pk_{1,k,l}^{TM}]$, $x \in \mathbb{N}^{+}$.
To satisfy the communication constraint, it is necessary that $\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\frac{WM_{i,j}}{TM_{i,j}} * H_{i,k} * H_{j,l}\right) \leq 1$. 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,

$$Pk_{1,k,l}^{TM} = \left\lceil \sum_{M_{i,j} \in Pack_{1,k,l}^{TM}} WM_{i,j} + ... + \sum_{M_{i,j} \in Pack_{x,k,l}^{TM}} \frac{WM_{i,j}}{x} \right\rceil \quad (8)$$

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.

6 EXPERIMENTAL STUDY

We present in the first subsection a case study in which we calculate the communication cost through the use of the proposed strategy. In the second, we evaluate the performance of the strategy by generating a large number of random tasks.

6.1 Implementation of the Proposed Strategy

Example - 1 In this example, we have considered a typical program made up of 10 executable tasks

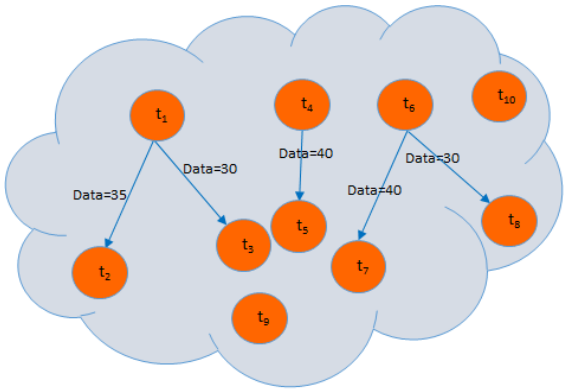

Figure 4: Inter task communication graph of example 1 (in kilo bits).

$\{\tau_1,\tau_2,...,\tau_{10}\}$ to be executed on a *MRS* having three cores $\{C_1,C_2,C_3\}$. The tasks' characteristics are described in Tab. 1 and the inter task communication graph is illustrated in Fig. 4. The cores connections matrix *Cost* is shown as follow:

$$Cost = \begin{array}{c} \begin{matrix} C_1 & C_2 & C_3 \\ \downarrow & \downarrow & \downarrow \end{matrix} \\ \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \end{array} \begin{matrix} \\ \leftarrow C_1 \\ \leftarrow C_2 \\ \leftarrow C_3 \end{matrix}$$

Table 1: Tasks' characteristics.

Task	T_i	$T_{i_{max}}$
τ_1	5	6
τ_2	6	8
τ_3	10	10
τ_4	4	6
τ_5	12	22
τ_6	5	16
τ_7	8	21
τ_8	8	32
τ_9	6	14
τ_{10}	7	12

The WCET W_i of each task is presented in Tab. 2

Table 2: WCET W_i of tasks.

WCET	C_1	C_2	C_3
W_1	1	3	0
W_2	1	0	2
W_3	2	0	0
W_4	1	2	0
W_5	2	0	0
W_6	0	2	0
W_7	0	0	3
W_8	0	2	1
W_9	0	0	2
W_{10}	0	2	1

The matrix $\gamma_{i,j}$ that represents the different possi-

bilities to execute tasks on cores is

$$\gamma = \begin{array}{c} \begin{matrix} C_1 & C_2 & C_3 \\ \downarrow & \downarrow & \downarrow \end{matrix} \\ \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \end{array} \begin{array}{l} \leftarrow \tau_1 \\ \leftarrow \tau_2 \\ \leftarrow \tau_3 \\ \leftarrow \tau_4 \\ \leftarrow \tau_5 \\ \leftarrow \tau_6 \\ \leftarrow \tau_7 \\ \leftarrow \tau_8 \\ \leftarrow \tau_9 \\ \leftarrow \tau_{10} \end{array}$$

Step 1 Assign the *SingleCore tasks*:
τ_3, τ_5, τ_6, τ_7 and τ_9 are *SingleCore tasks* and they must be assigned.
Step 2 Assign the tasks of *CPCL* to their related common processor *CP*:
Tab. 3 shows the clusters *CPCL* that have common processors *CP* and their data traffic *TD*.

Table 3: *CPCL* and *CL* clusters.

Clusters	Tasks	TD	CP
$CPCL_1$	τ_1, τ_2, τ_3	13	C_1
$CPCL_2$	τ_4, τ_5	10	C_1
CL_1	τ_6, τ_7, τ_8	14	-
CL_2	τ_9	0	C_3
CL_3	τ_{10}	0	C_2, C_3

The proposed strategy assigns $CPCL_1$ and $CPCL_2$ on C_1.
Step 3 Assign the tasks of the *CL*:
We start to assign the tasks of CL_1 because it has the highest *TD*. Since τ_6 and τ_7 are already mapped in step 1 to C_3, the proposed strategy assigns τ_8 to C_2 (same core of the related task τ_6).
The output of the proposed strategy is the mapping matix H

$$H = \begin{array}{c} \begin{matrix} C_1 & C_2 & C_3 \\ \downarrow & \downarrow & \downarrow \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \end{array} \begin{array}{l} \leftarrow \tau_1 \\ \leftarrow \tau_2 \\ \leftarrow \tau_3 \\ \leftarrow \tau_4 \\ \leftarrow \tau_5 \\ \leftarrow \tau_6 \\ \leftarrow \tau_7 \\ \leftarrow \tau_8 \\ \leftarrow \tau_9 \\ \leftarrow \tau_{10} \end{array}$$

The total cost of communication *TotalCost* is 40 units of cost.
The processor utilization of the cores after assigning tasks is presented in Tab. 4.

Table 4: Processor utilization for each core.

	$Core_1$	$Core_2$	$Core_3$
Processor utilization U	0.983	0.935	0.708

At a particular time, the system undergoes the first reconfiguration by adding two tasks that are presented in Tab. 5.

Table 5: Added tasks.

Task	T_i	$T_{i_{max}}$
τ_{11}	12	20
τ_{12}	4	15

The WCET W_i of each new task is presented in Tab. 6

Table 6: WCET W_i of each new task.

WCET	C_1	C_2	C_3
W_{11}	0	2	0
W_{12}	1	0	1

$$\gamma = \begin{array}{c} \begin{matrix} C_1 & C_2 & C_3 \\ \downarrow & \downarrow & \downarrow \end{matrix} \\ \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \end{array} \begin{array}{l} \leftarrow \tau_{11} \\ \leftarrow \tau_{12} \end{array}$$

The new inter task communication graph is illustrated in Fig. 5.

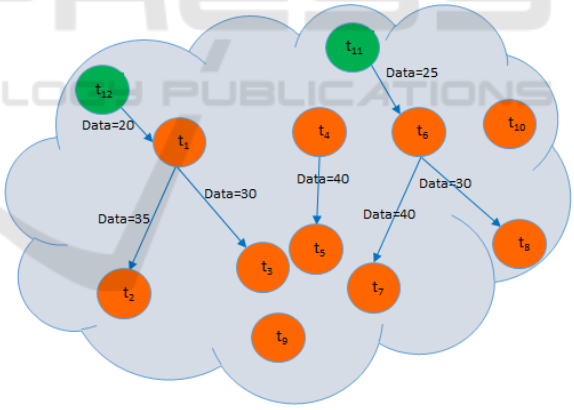

Figure 5: Inter task communication graph application's example after adding tasks (in kilo bits).

After assigning τ_{11} on C_2, since it is a *SingleCore task*, the processor utilization $U_{C_2} = 1.102 > 1$. The proposed strategy decreases U_{C_2} by applying Solution A, i.e., U_{C_2} equals to 0.933, and the new periods of tasks running on C_2 are presented in Tab. 7.
Since the task τ_{12} communicates with τ_1, then the best position to place it, is C_1 (same core to τ_1). Thus, the processor utilization of C_1 after assigning τ_{12} will be equal to 1.233. The proposed strategy begins by applying Solution A to decrease the processor utilization. After calculating the new periods, it appears that

Table 7: Modification of period's tasks running on C_2.

Task	T_i	$T_{i_{max}}$
τ_6	5	16
τ_8	10	32
τ_{10}	10	12
τ_{11}	15	20

Solution A can not increase the periods of tasks running on C_1 because their $T_{i_{max}}$ are not enough. Consequently, the proposed strategy proposes to assign τ_{12} to the next core C_3 and $U_{C_3} = 0.958$.
Thus, the new mapping matrix H is given by

$$H = \begin{array}{c} \begin{matrix} C_1 & C_2 & C_3 \\ \downarrow & \downarrow & \downarrow \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{array} \begin{array}{l} \leftarrow \tau_1 \\ \leftarrow \tau_2 \\ \leftarrow \tau_3 \\ \leftarrow \tau_4 \\ \leftarrow \tau_5 \\ \leftarrow \tau_6 \\ \leftarrow \tau_7 \\ \leftarrow \tau_8 \\ \leftarrow \tau_9 \\ \leftarrow \tau_{10} \\ \leftarrow \tau_{11} \\ \leftarrow \tau_{12} \end{array}$$

The new total cost of communication *TotalCost* is equal to 40+20=60 units of cost.

6.2 Evaluation of Performance

In order to generalize the performance evaluation of the proposed strategy, we generate randomly tasks by using the developed tool $Task - Generator$ presented in (Gammoudi et al., 2016c). The proposed strategy tries to form clusters of tasks and then allocate these clusters to the cores. The effectiveness of the proposed algorithm is compared with the optimal solution provided by the CPLEX solver (CPLEX, 2013). For several sets of input data (N, p), the comparison is shown in a tabular form as well as in a graphical form. Tabs. 8 and 9 illustrated in the form of Figs. 6 and 7 respectively.

It can be observed from Fig. 6 that the values of total cost obtained by the proposed strategy are near to those obtained by the optimal solution, in the case, when the number of cores is kept fixed and number of tasks are taken in increasing order. The similar observation can also be made from Fig. 7 in the case when the number of tasks is fixed and that of cores is taken in an increased order. Thus, it is concluded that the proposed strategy results in a near optimal cost in both cases.

Table 8: Communication cost when number of tasks increases.

Tasks N	Cores p	Proposed algo	Optimal sol
10	4	68	66
11	4	92	88
12	4	131	126
13	4	146	138
14	4	188	177
15	4	235	219
16	4	292	280
17	4	355	336

Table 9: Communication cost when number of cores increases.

Tasks N	Cores p	Proposed algo	Optimal sol
17	4	355	336
17	5	385	358
17	6	372	359
17	7	385	361
17	8	390	372
17	9	402	386
17	10	423	399
17	11	430	415

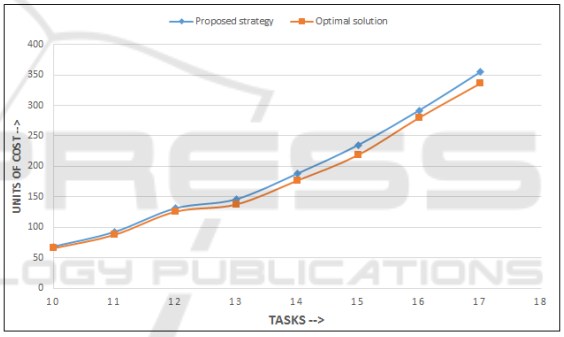

Figure 6: Communication cost of our strategy when tasks are in increasing order and number of cores is 4.

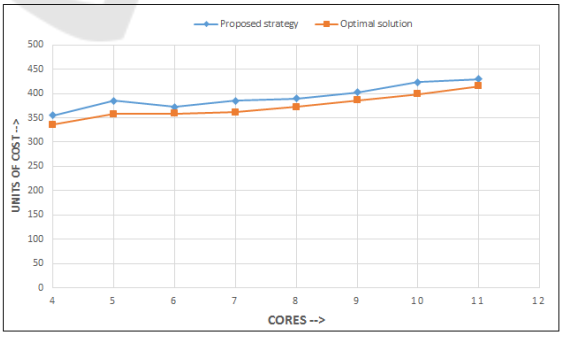

Figure 7: Communication cost of our strategy when cores are in increasing order and number of tasks is 17.

7 CONCLUSIONS

Task allocation problem for reconfigurable multicore systems, using task clustering, is discussed.

A novel task model based on elastic coefficients is proposed. This is to adapt task parameters allowing task allocation and reconfiguration while minimizing communication costs. The task allocation problem is known to be NP-hard. When considering communication costs, the proposed technique gives a near optimal solution while respecting real-time and communication constraints. Optimal solution is obtained by formulating the problem as an ILP problem and we compare the results of the proposed technique with the optimal solution. After some evaluations, we conclude that the proposed technique results in a near optimal ones provided by the CPLEX solver.

In a future work, we will focus on the implementation of the paper's contribution in a real-time operating system that will be evaluated by assuming real case studies.

REFERENCES

Baruah, S. and Goossens, J. (2004). Scheduling real-time tasks: Algorithms and complexity. *Handbook of scheduling: Algorithms, models, and performance analysis*, 3.

Bhardwaj, P. and Kumar, V. (2013). An effective load balancing task allocation algorithm using task clustering. *International Journal of Computer Applications*, 77(7).

Bui, B. D., Pellizzoni, R., and Caccamo, M. (2012). Real-time scheduling of concurrent transactions in multidomain ring buses. *IEEE Transactions on Computers*, 61(9):1311–1324.

Buttazzo, G. C., Lipari, G., and Abeni, L. (1998). Elastic task model for adaptive rate control. In *Real-Time Systems Symposium, 1998. Proceedings. The 19th IEEE*, pages 286–295. IEEE.

Carle, T., Potop-Butucaru, D., Sorel, Y., and Lesens, D. (2015). From dataflow specification to multiprocessor partitioned time-triggered real-time implementation. *Leibniz Transactions on Embedded Systems*, 2(2):01–1.

Cecilio, J. and Furtado, P. (2014). Architecture for uniform (re) configuration and processing over embedded sensor and actuator networks. *IEEE transactions on industrial informatics*, 10(1):53–60.

CPLEX (2013). Ibm ilog cplex optimize. In *http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud/*.

Gammoudi, A., Benzina, A., Chillet, D., and Khalgui, M. (2016a). New reconfigurable middleware for adaptive rtos in ubiquitous devices. In *10th International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*.

Gammoudi, A., Benzina, A., Khalgui, M., and Chillet, D. (2015). New pack oriented solutions for energy-aware feasible adaptive real-time systems. In *International Conference on Intelligent Software Methodologies, Tools, and Techniques*, pages 73–86. Springer.

Gammoudi, A., Benzina, A., Khalgui, M., and Chillet, D. (2016b). Real-time scheduling of reconfigurable battery-powered multi-core platforms. In *Tools with Artificial Intelligence (ICTAI), 2016 IEEE 28th International Conference on*, pages 121–129. IEEE.

Gammoudi, A., Benzina, A., Khalgui, M., Chillet, D., and Goubaa, A. (2016c). Reconf-pack: A simulator for reconfigurable battery-powered real-time systems. In *30th European Simulation and Modelling Conference*.

Gharsellaoui, H., Khalgui, M., and Ahmed, S. B. (2013). Reconfiguration of synchronous real-time operating system. *International Journal of System Dynamics Applications (IJSDA)*, 2(1):114–132.

Hu, J. and Marculescu, R. (2005). Energy-and performance-aware mapping for regular noc architectures. *IEEE Transactions on computer-aided design of integrated circuits and systems*, 24(4):551–562.

Khemaissia, I., Mosbahi, Olfa, K., Mohamed, L., and Zhiwi (2016). New methodology for feasible reconfigurable real-time network-on-chip noc. In *Proc. 11th Int. Conf. Softw. Eng. App on*.

Khemaissia, I., Mosbahi, O., and Khalgui, M. (2014). Reconfigurable can in real-time embedded platforms. In *Informatics in Control, Automation and Robotics (ICINCO), 2014 11th International Conference on*, volume 1, pages 355–362. IEEE.

Li, L., Li, S., and Zhao, S. (2014). Qos-aware scheduling of services-oriented internet of things. *IEEE Transactions on Industrial Informatics*, 10(2):1497–1505.

Liu, C. L. and Layland, J. W. (1973). Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, 20(1):46–61.

Marinoni, M. and Buttazzo, G. (2007). Elastic dvs management in processors with discrete voltage/frequency modes. *IEEE Transactions on industrial informatics*, 3(1):51–62.

Quadri, I. R., Gamatié, A., Boulet, P., Meftali, S., and Dekeyser, J.-L. (2012). Expressing embedded systems configurations at high abstraction levels with uml marte profile: Advantages, limitations and alternatives. *Journal of systems architecture*, 58(5):178–194.

Rooker, M. N., Sünder, C., Strasser, T., Zoitl, A., Hummer, O., and Ebenhofer, G. (2007). Zero downtime reconfiguration of distributed automation systems: The εcedac approach. In *Holonic and Multi-Agent Systems for Manufacturing*, pages 326–337. Springer.

Schranzhofer, A., Chen, J.-J., and Thiele, L. (2010). Dynamic power-aware mapping of applications onto heterogeneous mpsoc platforms. *IEEE Transactions on Industrial Informatics*, 6(4):692–707.

Tosun, S., Ozturk, O., and Ozen, M. (2009). An ilp formulation for application mapping onto network-on-chips. In *Application of Information and Communication Technologies, 2009. AICT 2009. International Conference on*, pages 1–5. IEEE.

Vrba, P. and Marik, V. (2010). Capabilities of dynamic reconfiguration of multiagent-based industrial control

systems. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 40(2):213–223.

Wang, X., Khemaissia, I., Khalgui, M., Li, Z., Mosbahi, O., and Zhou, M. (2015). Dynamic low-power reconfiguration of real-time systems with periodic and probabilistic tasks. *IEEE Transactions on Automation Science and Engineering*, 12(1):258–271.

Wang, X., Li, Z., and Wonham, W. (2016). Dynamic multiple-period reconfiguration of real-time scheduling based on timed des supervisory control. *IEEE Transactions on Industrial Informatics*, 12(1):101–111.