New Methodology for Feasible Reconfigurable Real-time Network-on-Chip NoC

Imen Khemaissia[1,2], Olfa Mosbahi[1], Mohamed Khalgui[1,4] and Zhiwu Li[3,4]
[1]*National Institute of Applied Sciences and Technology, INSAT, University of Carthage, Carthage, Tunisia*
[2]*Faculty of Sciences, Tunis El-Manar University, Tunis, Tunisia*
[3]*Institute of Systems Engineering, Macau University of Science and Technology, Taipa, Macau*
[4]*School of Electro-Mechanical Engineering, Xidian University, Xian 710071, China*

Keywords: Embedded System, Reconfigurable MPSoC, Multi-agent, Real-time and Low-power Scheduling.

Abstract: The current research paper is interested in flexible reconfigurable real-time Network-on-Chip (NoC) in Multiprocessors System-on-Chip MPSoC architectures. A NoC is composed of several nodes where each one consists of a processor and a router. The reconfiguration of a processor is any operation that permits the addition-removal-update of periodic dependent OS (Operating System) tasks that are sharing resources. For two added dependent tasks assigned to different processors, a message is added automatically on the NoC. After any reconfiguration scenario, several real-time constraints cannot be satisfied since a task can miss its deadline and a message can take a long time to arrive to its destination. In order to re-obtain the system feasibility, we propose a new approach that is called CRM (abrev. Cynapsys Reconfigurable MPSoC). A multi-agent architecture based on a master/slave model is defined where a slave agent is assigned to each node to control its local feasibility after any reconfiguration scenario, and a master is proposed for the whole architecture if any perturbation occurs at run-time by proposing software or hardware solutions. A developed tool at LISI laboratory and Cynapsys is implemented for a real case study in order to evaluate the paper's contribution.

1 INTRODUCTION

Recently, the embedded systems are based on the MPSoC oriented technologies since they meet the required performance of various applications in industry (Z.Hajduk and J.Sadolewski, 2015). An MPSoC is a system-on-chip (SoC) which is composed of numerous processors commonly dedicated for embedded applications. Reducing the power consumption becomes a major concern for the high-performance and reliability of such systems. The MPSoC can be adapted to its environment after any external/internal events that allow to add, remove or update the system tasks or messages to be exchanged between the processors of the chip. A hardware reconfiguration allows the activation/deactivation of a processor of the architecture. We found in the literature various interesting research works which deal with reconfigurable real-time embedded systems (X. Wang and Li, 2011), (J. F. Zhang, 2015), (George and Courbin,), more specifically the reconfiguration of MPSoC architectures (P.K.F. Holzenspies, 2007), (A. Samahi, 2007). Note that various research works on low-power execution of MPSoC architectures are found in (H. Javaid and Parameswaran, 2011), (R. Ben Atitallah and Blouin, 2013), (Salehi and Ejlali, 2015). Multiple algorithms are dedicated to schedule the OS tasks of embedded systems (N.Q. Wu and Li, 2015), (Baker, 1991), (Chetto and Chetto, 1989) (Burns and Wellings, 2001), (T.P.Baker, 1990), (Liu and Layland, 1973). Although all of them are interesting, no one in the related works deals with the real-time reconfiguration of an MPSoC architecture under low-power and low-memory constraints. This work is original since there is no related work that treats the low-power-reconfiguration of MPSoC oriented applications implemented by periodic OS tasks/messages under precedence constraints and with shared resources. In this paper, we assume an MPSoC-based application denoted in the following by *RSys* and composed of *n* nodes such that each one gathers a processor and a router. The different processors of *RSys* execute periodic OS tasks. They are assumed to be under precedence constraints and with shared resources. We use the well-known scheduling policy earliest deadline first (EDF) to schedule the tasks that implement the

Khemaissia, I., Mosbahi, O., Khalgui, M. and Li, Z.
New Methodology for Feasible Reconfigurable Real-Time Network-on-Chip NoC.
DOI: 10.5220/0005992002490257
In *Proceedings of the 11th International Joint Conference on Software Technologies (ICSOFT 2016) - Volume 1: ICSOFT-EA*, pages 249-257
ISBN: 978-989-758-194-6

different processors of *RSys*. The immediate priority ceiling protocol IPCP (Burns and Wellings, 2001) is utilized to deal with the precedence constraints of dependent tasks. The initial system is considered as feasible, i.e., the utilization of each processor is equal or less than 1. Several messages will be added too after the addition of tasks. After the application of successive reconfiguration scenarios, *RSys* becomes infeasible. Also, a message can take a long time to arrive to its destination. In order to resolve all these problems, a new approach called CRM (Cynapsys Reconfigurable MPSoC) is developed at Cynapsys [1] which is a professional company in the embedded technologies. Indeed, all the real-time and precedence constraints should be satisfied. We propose a multi-agent architecture based on the master/slave model to handle feasible reconfigurations of *RSys*. We propose two types of agents: i) A master agent: Controls the whole architecture of *RSys* after applying any reconfiguration scenario. If it receives a disapproval from the slave agent, then it proposes the modification of real-time parameters of tasks or their assignment to other processors of the same MPSoC architecture, or also the removal of some of them. ii) A slave agent: Is defined for each processor to inform the master agent if the energy is increased or the real-time constraints are violated. We choose to apply the paper's contribution to FPGA Stratix III and a tool is proposed to handle all the services provided by the different intelligent agents. The remainder of the paper is organized as follows: the next Section reviews the related works. Section 3 formalizes the reconfigurable MPSoC architectures followed by a case study. Section 4 proposes a methodology for a reconfigurable feasible real-time application. The implementation, simulation, and analysis are found in Section 5. Finally, the last Section summarizes this work with the presentation of future works.

2 STATE OF THE ART

We expose and analyze several research works which are related to the current contribution. Since this paper addresses the reconfigurable feasible NoC in adaptive MPSoC architectures, we start first by presenting the characteristics of MPSoC and NoC. Then we review some interesting related papers dealing with the real-time scheduling of OS tasks.

2.1 MPSoC and NoC Characteristics

The MPSoC uses diverse processors usually addressed to embedded applications. It is used by structure that are composed of multiple heterogeneous processing elements with particular services indicating the requirement of the expected application area (J. Sepulveda and Strum, 2012). These architectures meet the performance needs of many applications in different domains such as multimedia, telecommunication and network security. Because of their comparatively high performance, flexibility, and power efficiency, the MPSoC is based on NoC solutions (Hansson and Goossens, 2007),(Stensgaard and Sparso, 2008), (F. Martinez Vallina and saniie, 2007). Network on chip (NoC) is known as a new paradigm assigned for the interconnections within a system on chip (SoC) that presents a viable communication infrastructure (Bobda and Ahmadinia, 2005). Although all of them are interesting, there is no related work that deals with the real-time reconfigurable NoC in MPSoC architectures under low-power constraints.

2.2 Real-time Scheduling

Several successful studies in the literature deal with the real-time scheduling of OS tasks. The work in (Liu and Layland, 1973) proposes the earliest deadline first (EDF) and the rate monotonic (RM) to schedule periodic tasks. The work in (Burns and Wellings, 2001) presents the original priority ceiling protocol OPCP and immediate priority ceiling protocol IPCP in order to solve the scheduling problem of the tasks that share resources. The research work in (T.P.Baker, 1990) proposes the stack resource policy SRP that allows processes with different priorities to share a single run-time stack. In this paper, we use the EDF for the scheduling of periodic tasks since it is optimal under some assumptions.
In summary, a lot of successful investigations have been done in the domain of reconfigurable MPSoC-based embedded technologies. None of the previous works takes into account the feasibility at run-time of NoC in an MPSoC architecture under real-time and energy constraints.

3 RECONFIGURABLE MPSoC *RSys*

In this section, we start by formalizing the reconfigurable MPSoC *RSys* before exposing a case study that explains the the problem under consideration.

[1] Cynapsys company: http://www.cynapsys.de/

3.1 Formalization

We assume that *RSys* consists of the matrix N_{i*j} of nodes, i.e., $RSys = \{N_{1,1}, N_{1,2}, \ldots, N_{2,1}, N_{2,2}, N_{i,j}, \ldots, N_{l,c}\}$ ($i \in [1..l]$ and $j \in [1..c]$) where l and c represent respectively the numbers of the rows and columns of the network on chip NoC that is used to connect all the nodes of *RSys* (**?**). *RSys* is considered to be reconfigurable and adapted to its environment by adding/updating/removing OS tasks to/from the processors. We assume that each node $N_{i,j}$ is composed of: (a) Processor $Pr_{i,j}$: Executes periodic OS tasks $\tau_{i,j,k}$ ($i \in [1..l]$, $j \in [1..c]$ and $k \in [1..n_{i,j}]$) that share resources and under precedence constraints, (b) Router $R_{i,j}$: Is responsible of the message's forwarding in the NoC. The latter has a buffer that contains the list of messages to be added from a source node to a destination one. We note that all the processors of *RSys* share data in a global memory M_G.

According to Liu and Layland in (Liu and Layland, 1973), each periodic task $\tau_{i,j,k}$ ($i \in [1..l]$, $j \in [1..c]$ and $k \in [1..n_{i,j}]$) may generate many jobs. It is characterized by: a) Release time $R_{i,j,k}$: The time when a job starts its execution. If the tasks are synchronous, i.e, $R_{i,j,k} = 0$ b) Period $T_{i,j,k}$: Is the regular inter-arrival time, c) Deadline $D_{i,j,k}$: The absolute deadline that is equal to the sum of the release time and the relative deadline, d) WCET $C_{i,j,k}$: the time required to execute a job, and e) static priority $S_{i,j,k}$: The greatest static priority is equal to 1, i.e., $S_k = 1$ represents $\tau_{i,j,k}$ with the highest static priority. We consider that the tasks of *RSys* are sharing resources and are with precedence constraints. We assume that $T_{i,j,k} = D_{i,j,k}$. Each task in *RSys* is characterized by inclusion and exclusion sets according to user requirements respectively where:

- $Inclusion_{set}(\tau_{i,j,k})$: the set of processors that can handle the execution of $\tau_{i,j,k}$.
- $Exlusion_{set}(\tau_{i,j,k})$: the set of processors that cannot handle the execution of $\tau_{i,j,k}$.

According to (I. Khemaissia and Khalgui, 2014), (Baker, 1991), the processor utilization of periodic tasks that share resources is calculated as follows :

$$U_{per}(Pr_{i,j}) = (\sum_{k=1}^{n_{i,j}} \frac{C_{i,j,k}}{T_{i,j,k}} + \frac{B_{i,j,k}}{T_{i,j,k}}), \forall k \in [1..n_{i,j}] \quad (1)$$

where $B_{i,j,k}$ is the blocking factor that is defined as the time to spend by a task with a higher priority when blocked. It waits the termination of a task with a lower priority. In this work, we assume that the blocking factor $B_{i,j,k}$ is assumed to be equal to 1 or 0 (Baker, 1991). The EDF algorithm is used to schedule the independent tasks. For the schedule of tasks that share resources, we use the IPCP. The technique proposed in (I. Khemaissia and Khalgui, 2014) (Chetto and Chetto, 1989) is utilized to deal with the dependent tasks. For example, let us define two dependant tasks $\tau_{i,j,e}$ and $\tau_{i,j,f}$ such that $\tau_{i,j,e}$ precedes $\tau_{i,j,f}$. The deadlines of the tasks are attributed as follows:
if $\tau_{i,j,e}$ precedes $\tau_{i,j,f}$ **then** $S_{i,j,e} < S_{i,j,f}$. The deadline $D_{i,j,e}$ is given by:

$$D_{i,j,e} = min(D_{i,j,e}, (D_{i,j,f} - C_{i,j,e})) \quad (2)$$

By using Eq. (2), the precedence constraints will be satisfied. We note that the initial utilization of each processor U_{bef} is equal to:

$$U_{bef}(Pr_{i,j}) = U_{per}(Pr_{i,j}) \quad (3)$$

According to (X. Wang and Zhou, 2015) (I. khemaissia and Bouzayen, 2014), the energy to be consumed by a processor is proportional to the processor utilization. It is given by:

$$P_{i,j} \propto U_{bef}(Pr_{i,j})^2 \quad (4)$$

Let we assume a reconfiguration scenario at a specific time t. A software reconfiguration is applied by adding or removing OS tasks and U_{bef} increases to be U_{aft}. We assume that the source task $\tau_{i,j,h}$ exchanges a message $m_p(\tau_{i,j,h}; \tau_{a,b,k})$ with a target task destination $\tau_{a,b,k}$. A message $m_p(\tau_{i,j,h}; \tau_{a,b,k})$ in (I. Khemaissia and Khalgui, 2014) is characterized by: (i) A size $S_{mp}(\tau_{i,j,h}; \tau_{a,b,k})$, (ii) A transmission period $T_{mp}(\tau_{i,j,h}; \tau_{a,b,k})$, (iii) A deadline $D_{mp}(\tau_{i,j,h}; \tau_{a,b,k})$, (iv) A Worst Case Transmission Time $WCTT_p$ $C_{mp}(\tau_{i,j,h}; \tau_{a,b,k})$, and (v) A static priority $SP_{mp}(\tau_{i,j,h}; \tau_{a,b,k})$ where:

$$WCTT = S_{mp}(\tau_{i,j,h}; \tau_{a,b,k}) / debit_{NoC} \quad (5)$$

$$D_{mp}(\tau_{i,j,h}; \tau_{a,b,k}) = (D_{a,b,k} - C_{a,b,k}) \quad (6)$$

In this work, we assume that after each addition of a pair of tasks a new message is added automatically. The purpose of this research is to seek the optimal path between the source task and the destination one under real-time constraints. According to (B.D. Bui and Caccamo, 2005), the bus utilization is calculated as follows:

$$U_{bus}(m_p(\tau_{i,j,h}; \tau_{a,b,k})) = \sum_{p=1}^{m} \frac{C_{mp}(\tau_{i,j,h}; \tau_{a,b,k})}{T_{mp}(\tau_{i,j,h}; \tau_{a,b,k})} \quad (7)$$

Note that m is the messages number.

3.2 Case Study

The proposed approach in the current paper is applied to an FPGA Stratix III (Z-A. Obaid and Hamidon,

2009). Thus, we illustrate *RSys* through a running example in order to explain the proposed methodology by using theoretical tasks. Suppose that a startix FPGA board is composed three Nios II processors $Pr_{1,1}$, $Pr_{1,2}$ and $Pr_{2,1}$. We define the NoC as the communication architecture between the components of the MPSoC. Initially, *RSys* does not miss its real-time constraints and low-power properties. Table 1 lists the parameters of the different processors. We assume that its initial utilization $U_{per}(Pr_{2,1})$, $U_{per}(Pr_{1,1})$ and $U_{per}(Pr_{1,2})$ is equal to 0.8, 0.55 and 0.7, respectively, and all the tasks are released at the time $R_i = 0$ with $T_i = D_i$. We consider that $S(\tau_1) > S(\tau_2)$ and $S(\tau_5) < S(\tau_7)$. By using Eq. (2), D_1, D_2, D_5 and D_7 are equal to 16, 20, 25 and 40, respectively. The utilization of each processor is equal to 0.8, 0.55 and 0.7, respectively. Then, the initial energy consumption is 0.36, 0.30 and 0.49, respectively. Thus, the whole initial system is feasible. We assume also that the NoC can initially support all the added messages since its utilization is equal to 0.8.

Table 1: Characteristics of the initial periodic tasks.

τ_i	C_k	T_k	$Pr_{i,j}$	τ_i	C_k	T_k	$Pr_{i,j}$
τ_1	4	20	$Pr_{1,1}$	τ_5	5	25	$Pr_{1,2}$
τ_2	1	20	$Pr_{1,1}$	τ_6	3	30	$Pr_{1,2}$
τ_3	3	20	$Pr_{1,1}$	τ_7	6	40	$Pr_{1,2}$
τ_4	6	40	$Pr_{1,1}$	τ_8	5	20	$Pr_{1,2}$

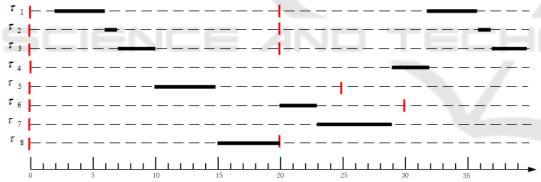

Figure 1: Initial scheduling of $Pr_{1,1}$ and $Pr_{1,2}$.

Table 2: Characteristics of the initial periodic messages.

Messages	C_{mp}	D_{mp}/T_{mp}	Messages	C_{mp}	D_{mp}/T_{mp}
$m1(\tau_1,\tau_2)$	3	30	$m5(\tau_1,\tau_5)$	1	10
$m2(\tau_1,\tau_3)$	2	20	$m6(\tau_3,\tau_2)$	5	25
$m3(\tau_2,\tau_4)$	2	20	$m7(\tau_4,\tau_6)$	3	30
$m4(\tau_3,\tau_1)$	2	20	$m8(\tau_7,\tau_8)$	2	20

Table 3 indicates the parameters of the added periodic tasks. Furthermore, we activate a new Nios II processor $Pr_{2,1}$. The processor utilization after the addition of OS tasks $U_{aft}(Pr_{1,1})$, $U_{aft}(Pr_{1,2})$ and $U_{aft}(Pr_{2,1})$ will be 1.05, 1.05 and 1.1, respectively. We deduct that the system is infeasible since the processor utilization of each processor exceeds 1. Also the energy consumption increases to be 1.1, 1.1 and 1.21, respectively.

After the addition of several messages, the bus utilization becomes equal to 1.2 and several messages

Table 3: Characteristics of the added periodic tasks.

τ_k	C_k	T_k	$SetInc(\tau_{i,j,k})$
τ_9	4	10	$Pr_{1,1}$, $Pr_{2,1}$
τ_{10}	2	20	$Pr_{1,1}$, $Pr_{1,2}$
τ_{11}	4	40	$Pr_{1,1}$, $Pr_{2,1}$
τ_{12}	9	30	$Pr_{1,2}$, $Pr_{2,1}$
τ_{13}	4	20	$Pr_{1,2}$, $Pr_{2,1}$
τ_{14}	2	40	$Pr_{1,2}$, $Pr_{2,1}$

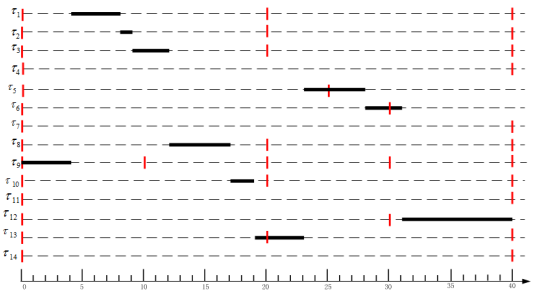

Figure 2: Scheduling after a reconfiguration scenario.

Table 4: Characteristics of the added periodic messages.

Messages	C_{mp}	D_{mp}/T_{mp}	Messages	C_{mp}	D_{mp}/T_{mp}
$m(\tau_{A9},\tau_{10})$	4	40	$m(\tau_{10},\tau_{13})$	2	20
$m(\tau_{10},\tau_{12})$	4	10	$m(\tau_9,\tau_{14})$	3	30
$m(\tau_{A8},\tau_{12})$	2	20	$m(\tau_{13},\tau_{14})$	2	10
$m(\tau_{11},\tau_{14})$	8	40	$m(\tau_{12},\tau_{14})$	3	10

cannot be supported by the NoC. Moreover, the messages take a long time to be routed. For that new software solutions are proposed for the NoC feasibility.

In the sections below, we propose new technical software solutions to satisfy the real-time constraints and to reduce the power consumption.

4 CRM: METHODOLOGY FOR FEASIBLE RECONFIGURABLE REAL-TIME EMBEDDED SYSTEMS

We propose in the current paper a new methodology CRM that deals with the feasible real-time reconfigurable MPSoC architectures. We aim to extend the work in (I. khemaissia and Khalgui, 2014) which is not interested in the feasibility of the NoC. The different steps of this methodology are stated as follows: (i) Application of reconfiguration scenarios in different processors, and (ii) Verification of the system's feasibility, i.e., feasibility of each processor and in NoC.

Before we describe these steps in details, let us define the proposed multi-agent architecture.

4.1 Multi-agent based Architecture

We define a multi-agent architecture following the Master-Slave model: (i) Master Agent Ag_M: Responsible of the whole system and the NoC feasibility, (ii) Slave Agent $Ag_{i,j}$: Defined for each node $N_{i,j}$ to inform Ag_M if the energy increases or if the local real-time constraints are not satisfied. Fig. 3 summarizes the whole contribution of this current research. It shows the different states of the system and the proposed agents to resolve any destabilization that can occur before and after applying any reconfiguration scenario.

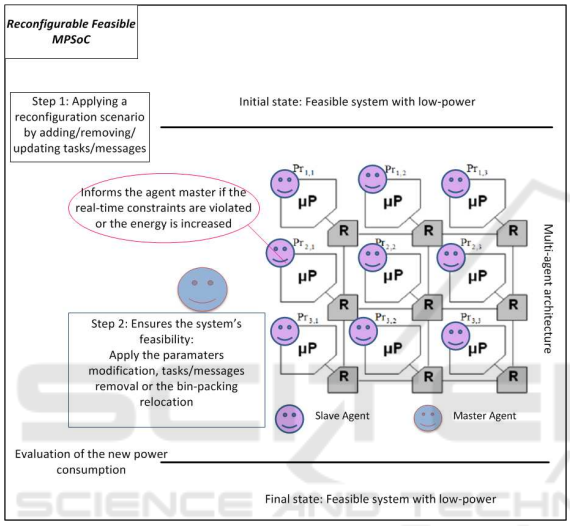

Figure 3: CRM Methodology.

4.2 System Feasibility

Many solutions are proposed to re-obtain the system feasibility in all the processors/NoC of *RSys* after applying any reconfiguration scenario that violates real-time or energy constraints.

4.2.1 Solution 1: Modification of Parameters

The utilization must be modified to ensure the maintain of the energy consumption. A technical solution to be proposed by Ag_M allows to change the parameters of all the initial and new tasks. We suggest in the current work the modification of periods as a solution 1.1 or WCETs as a solution 1.2 to guarantee the satisfaction of all the constraints after a such scenario. According to (I. khemaissia and Khalgui, 2014), the new periods become:

$$T_{i,j,k}^{(r)} = \left\lceil \left(\sum_{k=1}^{n} \left(\frac{(C_{i,j,k} + B_{i,j,k})}{U_{per}(Pr_{i,j})} \right) \right) \right\rceil \quad (8)$$

Once the periods are modified, we calculate the new processor utilization of periodic tasks $U_{perT}^{(r)}$ as follows:

$$U_{perT}^{(r)}(Pr_{i,j}) = \sum_{k=1}^{n} \frac{C_{i,j,k}}{T_{i,j,k}^{(r)}} + \frac{B_{i,j,k}}{T_{i,j,k}^{(r)}} \quad (9)$$

where n is the number of periodic tasks and r is the reconfiguration number. Then, the new power consumption is given by:

$$P_{perT}^{(r)}(Pr_{i,j}) \propto (U_{perT}^{(r)}(Pr_{i,j})^2) \quad (10)$$

The new values of the constant period of initial and new tasks to be executed by $Pr_{1,1}$, that are calculated by Eq. (8), are equal to 33 Time Units. It is equal to 49 and 12 for the tasks of $Pr_{1,2}$ and $Pr_{2,1}$. Then the new processor utilizations of $Pr_{1,1}$, $Pr_{1,2}$ and $Pr_{2,1}$ are equal to 0.54, 0.69 and 0.5, respectively. The periods modification can maintain the utilization of all the processors of *RSys* and can stabilize the power consumption. If we modify the WCET of tasks, then the new $C_k^{(r)}$ are given by:

$$C_{i,j,k}^{(r)} = \begin{cases} \left\lfloor \frac{U_{per}(Pr_{i,j}) - \sum_{k=1}^{n} \frac{B_{i,j,k}}{T_{i,j,k}}}{\sum_{k=1}^{n} \frac{1}{T_{i,j,k}}} \right\rfloor \\ 1, if \frac{U_{per}(Pr_{i,j}) - \sum_{k=1}^{n} \frac{B_{i,j,k}}{T_{i,j,k}}}{\sum_{k=1}^{n} \frac{1}{T_{i,j,k}}} \leq 0 \end{cases} \quad (11)$$

After the modification of the WCETs, the new processor utilization $U_{perC}^{(r)}$ is given by:

$$U_{perC}^{(r)}(Pr_{i,j}) = \left(\sum_{k=1}^{n} \frac{C_{i,j,k}^{(r)}}{T_{i,j,k}} + \frac{B_{i,j,k}}{T_{i,j,k}} \right) \quad (12)$$

The new power consumption is:

$$P_{perC}^{(r)}(Pr_{i,j}) \propto (U_{perC}^{(r)}(Pr_{i,j})^2) \quad (13)$$

By using Eq. (11), the new constant WCET of old and new tasks to be executed by $Pr_{1,1}$ and $Pr_{1,2}$ is equal to 2 Time Units. It is equal to T6 Time Units for the tasks of $Pr_{2,1}$. Hence, the new utilizations of $Pr_{1,1}$, $Pr_{1,2}$ and $Pr_{2,1}$ are equal to 0.55, 0.51 and 0.45, respectively. The WCETs modification can reduce the processor utilization of all the processors of *RSys* and the power consumption is minimized too. According to (I. Khemaissia and Khalgui, 2014), if the bus cannot support the added messages, then we can modify the parameters of the messages or remove the unimportant ones. The new period or the WCTT of each message is calculated according to (I. Khemaissia and Khalgui, 2014) as follows:

$$T_{mp} = \left\lceil \frac{\sum_{i=1}^{m} C_{mp}}{U_{NoC}(periodicmessages)} \right\rceil \quad (14)$$

or

$$C_{mp} = \begin{cases} 1,0 < \frac{U_{NoC}(periodicmessages)}{\sum_{i=1}^{m} \frac{1}{T_{mp}}} \leq 1 \\ \lfloor \frac{U_{NoC}(periodicmessages)}{\sum_{i=1}^{m} \frac{1}{T_{mp}}} \rfloor, \frac{U_{NoC}(periodicmessages)}{\sum_{i=1}^{m} \frac{1}{T_{mp}}} > 1 \end{cases} \quad (15)$$

where m denotes the number of messages.
After applying Eq. (14), the new utilization of NoC becomes equal to 0.8. Then the NoC is considered feasible.

4.2.2 Solution 2: Tasks/Message Removal

As a second solution, Ag_M suggests to remove some OS tasks/messages according to their priorities. If we need to remove a task with precedence constraints, then it is necessary to verify if the tasks that depends on it have a lower priority or not. In this case, the dependent tasks must be removed from the system. Otherwise, it is not allowed to remove this task. For example, if we assume to remove these tasks: A_1 and A_4 from $Pr_{1.1}$, then its processor utilization becomes equal to 1. If we remove the tasks with the lowest priority from all the processors, thus we can reduce the processor utilization. It is similar for the messages, i.e., if we remove the messages with the lowest priority, then the utilization of NoC will be reduced. This solution can provide a feasible real-time system after any reconfiguration scenario but the values of the processor/NoC utilization depend on the number of the removed tasks/messages.

4.2.3 Solution 3: Relocation of Tasks according to the Bin-packing

We use the bin-packing algorithm to relocate the tasks according to several conditions that will be described below (Davis, 2006). The system can be reconfigured by Ag_M on two levels in order to be temporally feasible with a low-power. Since we can calculate the processor utilization of each task, two steps must be done: (a) **Step 1:** We relocate the tasks by using one of the proposed algorithms of the bin-packing, and (b) **Step 2:** We modify their parameters by following Solution 1 if the current utilization of a processor exceeds 1. Before applying the bin-packing, we should start by ordering the tasks in an ascending order and the processors in a descending order according to their utilization. Two conditions should be satisfied when we re-locate a task $\tau_{i,j,k}$ to $Pr_{i,j}$: (i) **Condition 1:** the processor $Pr_{i,j} \in Inclusion_{set}(\tau_{i,j,k})$, and (ii) **Condition 2:** $U_{bef}(Pr_{i,j}) \leq 1$. Algorithm 1 describes the different followed steps in order to relocate the tasks according to the next fit descending NFD.

Algorithm 1: Relocation of new tasks according to NFD.

```
Order the processors in an ascending order;
Order the tasks in a descending order;
for (each processor Pr_{i,j}) do
  for (each task τ_{i,j,k}) do
    if (Pr_{i,j} is active) and (τ_{i,j,k} ∈ Inclusion_set(τ_{i,j,k})) then
      Calculate the current processor utilization
      U_bef(Pr_{i,j}) after the addition of τ_{i,j,k};
    else
      if ((U_bef(Pr_{i,j}) ≥ 1) then
        Assign τ_{i,j,k} to Pr_{i,j} that satisfies the conditions (conditions 1, 2 and 3);
        Re-Calculate U_bef(Pr_{i,j}) after the addition of τ_{i,j,k} to Pr_{i,j};
        break;
      else
        Close the current processor and open the next one;
        Re-Calculate the utilization of the newest opened processor;
      end if
    end if
  end for
end for
```

In order to apply the NFD on the running example, we start by ordering the tasks and the processors in a descending order and an ascending one, respectively. We have $U_{bef}(Pr_{1,1}) < U_{bef}(Pr_{1,2}) < U_{bef}(Pr_{2,1})$. If we re-order the periodic tasks, we get τ_9, τ_{12}, τ_{13}, τ_{10}, τ_{11}, τ_{14}. Every time we add a task, we should verify if the processor can include it. The NFD is applied as follows: τ_9 is packed into $Pr_{1,1}$ and will be closed. Since $Pr_{1,1} \notin SetInc(\tau_{12})$, it will be closed and the current added task is put into $Pr_{1,2}$. The utilization of $Pr_{1,2}$ becomes equal to 1. Then, it will be closed and $Pr_{2,1}$ is open to support τ_{13}. For τ_{10}, it will be packed into $Pr_{1,1}$ that is re-opened since $Pr_{2,1}$ is not mentioned in $SetInc(\tau_{10})$. If we add τ_{11} into $Pr_{1,1}$, its utilization becomes equal to 1.1. In this case, Ag_M applies the periods modification. The same solution is applied when τ_{14} is added into $Pr_{1,2}$.

5 EXPERIMENTATION

This section presents an experimentation that applies low-power reconfigurations of MPSoC-based architectures. We present firstly the implementation of the agent-based architecture. After that, theoretical simulations and analysis are shown to highlight the advantages of the proposed contribution. We choose to apply the latter to Stratix III development board.

5.1 Implementation of the Communication Protocol

In this section, we present the main algorithm that applies the proposed methodology. A protocol is defined as a system of rules required to make easier the communication between the different agents of the system. Before describing the algorithm, let us present the following used functions. (i) Send-approval-power($Ag_{i,j}$,Ag_M): If the power consumption is inferior to 1 after a reconfiguration scenario, then $Ag_{i,j}$ sends an approval message to Ag_M, (ii) Send-alert-power($Ag_{i,j}$,Ag_M): If the power consumption is superior to 1 after a reconfiguration scenario, then $Ag_{i,j}$ sends a disapproval message to Ag_M, (iii) Evaluate-power-consumption(Ag_M): Once one of the proposed solutions is applied, Ag_M computes the difference between the power consumption before and after the reconfiguration, (iv) Manage-removal($Ag_{i,j}$): Each agent $Ag_{i,j}$ must update the memory after any scenario allowing the removal of tasks from a processor $Pr_{i,j}$, (v) App-sol1.1(): Periods modification, (vi) App-sol1.2(): WCETs/WCTTs modification, (vii) App-sol2(): Tasks/messages removal, and (viii) App-sol3(): Re-location by applying the bin-packing. Algorithm 2 is developed to control the power consumption by applying new software solutions. It is with complexity $O(n^2)$. First of all, it reads the parameters of the initial tasks. Afterwards, it reads the parameter of the added messages. Finally, it verifies the feasibility of the system after the reconfiguration. If the utilization of a processor/NoC exceeds 1, then the agent suggests one of the proposed solutions that are mentioned previously.

Algorithm 2: Allocations of OS Tasks/Messages to Reconfigurable MPSoCs.

for each reconfiguration scenario **do**
  Compute the utilization U_{aft} and the NoC utilization;
  if $U_{aft} \leq 1$ or $U_{NoC} \leq 1$ **then**
    Send-approval-power($Ag_{i,j}$,Ag_M);
  else
    Send-alert-power($Ag_{i,j}$,Ag_M);
    Call(App-sol1.1()) or Call(App-sol1.2()) or Call(App-sol2())or Call(App-sol3());
    // Applying one of these solutions
  end if
end for
for each processor **do**
  Compute the utilization after reconfiguration;
  Evaluate-power-consumption(Ag_M);
end for

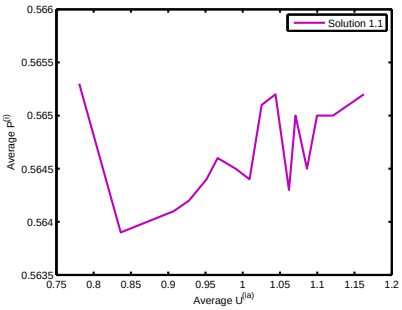

Figure 4: Power consumption after the periods modification.

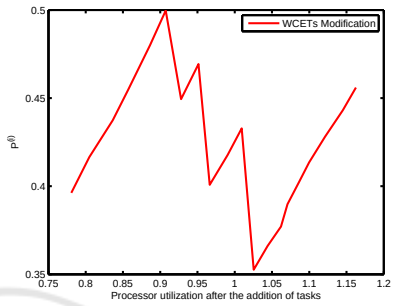

Figure 5: Power consumption after the WCETs modification.

5.2 Simulations

This section presents the obtained results after applying the proposed solutions. The initial system is assumed to be feasible with low-power. The processor utilization of each processor $Pr_{1,1}$, $Pr_{1,2}$, and $Pr_{2,1}$ is equal to 0.696532, 0.751534 and 0.803858, respectively. Fig. 4 depicts the power consumption after the modification of the periods and the WCETs. We can deduce that the periods modification can stabilize the power consumption. But, the WCETs modification can reduce the power consumption since the curves show important variations. We can conclude that this theoretical simulation result by Solution 1.2 is more advantageous than Solution 1.1. Fig. 6 visualizes the simulation result after applying the bin-packing algorithm. The different values of the power consumption are in the closed interval $[0.5635..0.5655]$. Then, we can neglect the variations. This solution is considered as effective as well.

6 CONCLUSION

In this paper, a new approach called CRM is developed for low-power reconfigurable MPSoC-based architectures. Initially, the system is feasible with low-

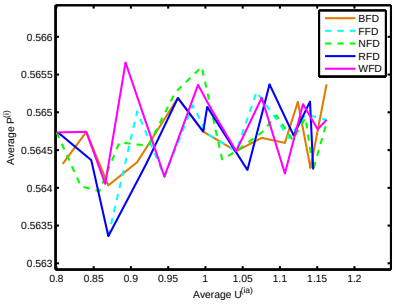

Figure 6: Power consumption after applying the bin-packing.

power. However, after many reconfiguration scenarios, the power consumption becomes bigger and some real-time requirements may not be satisfied. A multi-agent architecture based on the master-slave model is proposed, where software/hardware technical solutions are applied in order to obtain a feasible real-time system guaranteeing the minimization or the maintain of the power consumption. This new methodology is applied to confidential projects at Cynapsys. To our best knowledge, no studies dealing with reconfigurable real-time MPSoC under low-power constraints were suggested before. As a future work, we will be interested in aperiodic/sporadic tasks. Also, we will be interested in the reconfigurable routing of periodic/sporadic messages under low-power and low-memory constraints.

REFERENCES

A. Samahi, E. B. (2007). Automated integration and communication synthesis of reconfigurable mpsoc platform. In *Second NASA/ESA Conference on Adaptive Hardware and Systems*, Edinburgh.

Baker, T. (1991). Stack-based scheduling of realtime processes. Journal of Real-Time Systems.

B.D. Bui, R. P. and Caccamo, M. (2005). *Real-time Scheduling of Concurrent Transactions in Multi-domain Ring Buses*. IEEE Transactions on Computers.

Bobda, C. and Ahmadinia, A. (2005). *Dynamic interconnection of reconfigurable modules on reconfigurable devices*. Design and Test of Computers.

Burns, A. and Wellings, A. (2001). Scheduling algorithms for multiprogramming in a hard real time environment. In *Addison Wesley Longmain*.

Chetto, H. and Chetto, M. (1989). Some results of the earliest deadline scheduling algorithm. IEEE Transactions on Software Engineering.

Davis, T. (2006). *Bin Packing*. http://www.geometer.org/mathcircles.

F. Martinez Vallina, N. J. and saniie, J. (2007). *Nova interconnect for dynamically reconfigurable noc systems*. IEEE Electro/Information Technology.

George, L. and Courbin, P. Reconfiguration of uniprocessor sporadic real-time systems: the sensitivity approach. In *chapter in IGI-Global Knowledge on Reconfigurable Embedded*.

H. Javaid, M. Shafique, J. H. and Parameswaran, S. (2011). *System-level application-aware dynamic power management in adaptive pipelined MPSoCs for multimedia*. Computer-Aided Design, San Jose, CA.

Hansson, A. and Goossens, K. (2007). *Trade-offs in the configuration of a network on chip for multiple use-cases*. Procedings of International Symposium on Networks on Chip (NOCS), Princeton, NJ.

I. Khemaissia, O. Mosbahi, M. K. and Bouzayen, W. (2014). *New Reconfigurable Middleware for Feasible Adaptive RT-Linux*. pervasive and computing embedded and communication systems, Lisbon, Portugal.

I. Khemaissia, O. M. and Khalgui, M. (2014). *New automatic agent-based solutions for feasible reconfigurable MP-SoC architectures*. Proceedings of the 14th International Conference on Application of Concurrency to System Design, Tunisia.

I. Khemaissia, O. M. and Khalgui, M. (2014). *Reconfigurable CAN in real-time embedded platforms*. Proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics (ICINCO), Austria.

J. F. Zhang, M. Khalgui, Z. W. L. G. F. O. M. H. B. S. (2015). Reconfigurable coordination of distributed discrete event control systems. In *IEEE Transactions on Control Systems Technology*.

J. Sepulveda, R. Pires, G. G. W. J. C. and Strum, M. (2012). *QoSS hierarchical NoC-based architecture for MP-SoC dynamic protection*. International Journal of Reconfigurable Computing.

Liu, C. L. and Layland, J. W. (1973). *Scheduling algorithms for multiprogramming in a hard real time environment*. J. Assoc. Comput. Mach.

N.Q. Wu, M. Z. and Li, Z. (2015). *Short-term scheduling of crude-oil operations: Petri net-based control-theoretic approach*. IEEE Robotics and Automation Magazine.

P.K.F. Holzenspies, G.J.M. Smit, J. K. (2007). Mapping streaming applications on a reconfigurable mpsoc platform at run-time. In *International Symposium System-on-Chip*, Tampere.

R. Ben Atitallah, E. Senn ; D. Chillet, M. L. and Blouin, D. (2013). *An efficient framework for power-aware design of heterogeneous MPSoC*. IEEE Transactions on Industrial Informatics.

Salehi, M. and Ejlali, A. (2015). *A Hardware platform for evaluating low-energy multiprocessor embedded systems based on COTS devices*. IEEE Transactions on Industrial Electronics.

Stensgaard, M. and Sparso, J. (2008). *Renoc : A network-on-chip architecture with reconfigurable topology*. Second ACM/IEEE International Symposium on Networks-on-Chip.

T.P.Baker (1990). A stack-based resource allocation policy for realtime processes. In *Real-Time Systems Symposium*.

X. Wang, M. K. and Li, Z. W. (2011). Dynamic low power reconfigurations of real-time embedded systems. In *in: Proc. 1st Pervas. Embedded Comput. Commu. Syst*, Portugal.

X. Wang, I. Khemaissia, M. K. Z. W. L. O. M. and Zhou, M. (2015). *Dynamic low-power reconfiguration of real-time systems with periodic and probabilistic tasks*. IEEE Transactions on Automation Science and Engineering.

Z-A. Obaid, A. S. and Hamidon, M. (2009). *FPGA-based implementation of digital logic design using altera DE2 board*. International Journal of Computer Science and Network Security.

Z.Hajduk, B. and J.Sadolewski (2015). Architecture of fpga embedded multiprocessor programmable controller. In *iIEEE Transactions on Industrial Electronics*.