REHLib: New Optimal Implementation of Reconfigurable Energy Harvesting Multiprocessor Systems

Wiem Housseyni[1], Olfa Mosbahi[1] and Mohamed Khalgui[1,2]
[1]*National Institute of Applied Sciences and Technology (INSAT), University of Carthage, Tunis 1080, Tunisia*
[2]*School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China*

Keywords: Real-time Scheduling, Distributed Architecture, Energy Harvesting, Dynamic Reconfiguration, RTOS, Posix-based Implementation.

Abstract: The designs of reconfigurable embedded real-time energy harvesting multiprocessor systems are evolving for higher energy efficiency, high-performance and flexible computing. Energy management has long been a limiting factor in real-time embedded systems. A reconfiguration is defined as a dynamic operation that offers to the system the capability to adjust and adapt its behavior i.e., scheduling policy, power consumption, or to modify the applicative functions i.e., add-remove-update software tasks, according to environment and the fluctuating behavior of renewable source. This paper provides an implementation of reconfigurable multiprocessor energy harvesting systems. The objective of this work is to develop software components for the design of real-time operating systems. We propose a novel adaptive approach in order to address the limitations in energy harvesting systems. We develop a reconfigurable real-time energy harvesting system based on POSIX implementation. The proposed approach is assessed from two aspects, energy management and real-time scheduling. Experimental results show the effectiveness of the proposed approach compared with state-of-the-art techniques.

1 INTRODUCTION

The uses of multiprocessor embedded real-time systems have grown tremendously in recent years in variety of applications from our daily lives to industry production. This is due to the fact that multiprocessor systems fulfill the growing demand of scalable, high-performance, and highly reliable systems. Multiprocessor systems are widely used in applications involving wireless sensor networks, air traffic control, and battle field surveillance. However, their usefulness is severely limited by the battery capacity. The battery lifetime is a major challenge in the design of embedded systems particularly hard-real-time systems. In recent years, energy scavenging or harvesting technology from renewable sources such as photovoltaic cells, and piezoelectric vibrations emerges as new alternative to ensure sustainable autonomy and perpetual function of the system. By the same token, the literature has revealed a substantial interest in scheduling research for energy aware and power management scheduling for real-time systems. Still, there is sufficient scope for research, although uni-processor real-time scheduling for energy harvesting based systems is well studied. On the other hand, scheduling techniques for the reconfigurable multiprocessor energy harvesting systems are not mature enough to either be applicable or optimal as much as currently available uni-processor real-time scheduling techniques. Reconfigurable systems are solutions to providing both higher energy efficiency, and high-performance and flexibility. From the literature we can drive different definitions of a reconfigurable system. The authors of (Grichi et al., 2015) define a reconfiguration of a distributed system as any addition/ removal/update of one/more software-hardware elements. In this work, we define a reconfiguration as a dynamic operation that offers to the system the capability to adjust and adapt its behavior i.e., scheduling policy, power consumption, according to environment and the fluctuating behavior of renewable source, or to modify the applicative functions i.e., add-remove-update software tasks. Almost of embedded systems are real-time constrained. A real-time system involves a set of tasks where each task performs a computational activity according to deadline constraints. The main purpose of a real-time system is to produce not only the required results but also within strict time constraints. The

Housseyni, W., Mosbahi, O. and Khalgui, M.
REHLib: New Optimal Implementation of Reconfigurable Energy Harvesting Multiprocessor Systems.
DOI: 10.5220/0006432203460354
In *Proceedings of the 12th International Conference on Software Technologies (ICSOFT 2017)*, pages 346-354
ISBN: 978-989-758-262-2

control of real-time systems is ensured by a software called Real-Time Operating System (RTOS). The objective of the RTOS is to manage and control the assignment of system tasks in order to meet real-time requirements. Over the last years, several real-time operating systems have been ported RTLinux (Barabanov and Yodaiken, 1996), RTAI (Mantegazza et al., 2000), Xenomai (Gerum, 2004), and LITMUSRT (Calandrino et al., 2006). However, none of these approaches provide support for real-time reconfigurable multiprocessor energy harvesting based applications yet. New challenges raised by real-time reconfigurable embedded energy harvesting multiprocessors systems (REEHMSs) for RTOS in terms of reconfigurability, scheduling, and energy harvesting management. Based on these motivations, we investigate in this paper the challenges and viability of implementing a reconfigurable real-time energy harvesting scheduler in OS. There are many programming languages designed for the development of real-time systems such as POSIX (Portable Operating System Interface) (Mueller et al., 1993). POSIX standard promotes portability of applications across different operating system platforms. The purpose of this work is to address the scheduling issue of real-time tasks in REEHMSs. For this aim, the main contributions of this paper is summarized as follows: i) A new Energy-Based-Utilization task partitioning heuristic is proposed for the initial task assignment executed offline, ii) A new dynamic adjustment algorithm is proposed for performing system reconfiguration in case of infeasible execution, iii) A new framework REHLib: Reconfigurable Energy Harvesting Library for the design of real-time operating systems is developed to address reconfigurability and energy harvesting limitation in RTOS with POSIX-based implementation. As far as we know, this is the first work that attempt to implement the real-time scheduling for reconfigurable energy harvesting multiprocessor systems in RTOS. The remaining of this paper is structured as follows. Section 2 summarizes a state of the art relative first to real-time scheduling in energy harvesting based embedded systems, then to reconfigurable embedded systems. The system architecture is given in Section 3. Section 4 details the proposed approach. Section 5 presents the implementation of the proposed approach and its integration in Linux kernel. Performance evaluation is studied in Section 6. Section 7 concludes the paper and gives some new directions of work.

2 STATE OF THE ART

In this section, we present a state of the art dealing first with real-time scheduling in energy harvesting systems, then with reconfigurable embedded systems.

2.1 Real-time Scheduling in Energy Harvesting Based Systems

Uni-processor real-time scheduling for energy harvesting based systems is well studied (Allavena and Mossé, 2001), (Chetto, 2014). In (Ghor et al., 2011) the authors proposed the EDeg (Earliest Deadline with energy guarantee) algorithm. According to EDeg algorithm, the processor executes tasks as soon as possible according to the EDF rule, as long as the system can perform without energy failure. The authors define the notion of slack energy which is an extension of the notion of slack time. Slack energy permits to quantify the energy consumed by future jobs and to prevent them from violating their deadlines because of energy shortage. Then, as soon as future energy failure is detected, the system is suspended as long as possible depending on slack time or until the energy storage unit is full. In (Wei et al., 2010), the authors proposed an adaptive energy efficient task allocation scheme for a multiprocessor system-on-a-chip (SoC) in real-time energy harvesting systems. Recently, in (Lu and Qiu, 2011), the authors propose a novel technique of task partitioning for Energy Harvesting Real-Time Embedded System (EH-RTES) multicore based on DVFS-capability. No work from the state of the art deals with unpredictable behavior of reconfigurable computing systems.

2.2 Reconfigurable Real-time Embedded System

The real-time scheduling for reconfigurable embedded systems is well addressed in (Wang et al., 2016), (Wang et al., 2015). In (Wang et al., 2010), a study for low power dynamic reconfigurations of real-time embedded systems is proposed. An agent-based architecture is proposed where an intelligent software agent is developed to check each dynamic reconfiguration scenario and to suggest for users useful technical solutions that minimize the energy consumption. It proposes to modify periods, reduce execution times of tasks or remove some of them. Taken by the same scope, the authors in (Khemaissia et al., 2014) propose to develop an intermediate layer which presents the middleware that will be in interaction with the kernel Linux. The proposed middleware manages

the addition/removal/update of the periodic and aperiodic tasks with precedence constraints and sharing resources. We remind that all related works do not model the reconfiguration in real-time multiprocessor energy harvesting systems. Recently, in our previous work (Housseyni et al., 2016) (Housseyni et al., 2015) we interested to dynamic software reconfiguration in distributed embedded energy harvesting systems. We proposed dynamic run-time reconfiguration scenarios, i.e, task parameters modification, migration, degradation of quality of services, and task removal, in accordance with user requirements in reaction to unpredictable events from the environment or hardware failures. The main purpose is to guarantee a feasible system where real-time and energy harvesting requirements are respected, with the consideration of system performance optimization. While, in the current work we propose dynamic software and hardware reconfiguration. To the best of our knowledge, this is the first work that deals with dynamic software and hardware reconfiguration for real-time scheduling in a reconfigurable multiprocessor embedded system with energy harvesting assumptions.

3 SYSTEM FORMALIZATION

We present in this section a formal description of the considered reconfigurable energy harvesting multiprocessor system. The investigated system model is depicted in Figure 1.

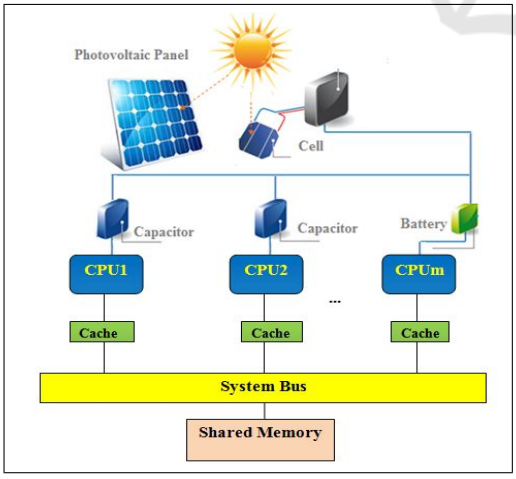

Figure 1: System model.

3.1 Hardware Architecture

We consider a symmetric multiprocessor (SMP) architecture. The considered SMP configuration *Sys*, is constructed of m identical CPU *Sys*: $\{P_1,..,P_m\}$, where preemption and migration of tasks are authorized. The CPUs lie close to each other and are physically connected over a common high-speed bus. The processors share a global memory module (shared memory) and peripheral devices through a common I/O bus interface. Each processor P_j, $j \in \{1,..,m\}$ is connected to a rechargeable energy storage with limited capacity B_j. We assume that the energy storage is supplied by a renewable energy source such as the photovoltaic energy.

3.2 Energy Considerations

The battery B_j, $j \in \{1,..,m\}$, is characterized by a capacity C_j. We assume hereafter that the energy storage can be completely depleted to as little as zero. The energy available in the storage B_j at time t is denoted by $E_{B_j}(t)$. We also assume that each energy storage can be charged up to its capacity. Let $P_{h_j}(t)$ be the instantaneous power of harvesting energy of the battery associated to processor P_j. The harvested energy in the interval time $[t_1, t_2]$ in the battery associated to the processor P_j, denoted $E_{h_j}(t_1, t_2)$ is calculated as follows:

$$E_{h_j}(t_1,t_2) = \int_{t_1}^{t_2} P_{h_j}(t)\,dt \quad (1)$$

We suppose that the incoming power received by the storage unit associated to processor P_j is a constant in time ($\forall$ t, $P_{h_j}(t) = P_{h_j}$).

3.3 Real-time Tasks Model

We assume that *Sys* performs a set of n periodic independent task set. In this work, a task can be a thread or a process. Each task τ_i is characterized by: i) Period T_i, ii) Worst case execution time (WCET) C_i in conformance with the classical task model of Liu and Layland (Liu and Layland, 1973), iii) Worst case energy consumption (WCEC) En_i expressed in Joules, and iv) A degree of criticality dc_i that defines its applicative importance. The degree of criticality is defined as the functional and operational importance of a task. The designer of the system defines (manually) the degree of criticality of each task in the system. We consider that tasks have implicit deadlines, i.e. deadlines are equal to periods. In addition we define task CPU utilization calculated as follows: $U_{\tau_i} = \frac{C_i}{T_i}$

4 RECONFIGURABLE ENERGY HARVESTING AWARE SCHEDULER

The task scheduling problem on a REEHMS constrained by both real-time and energy constraints with uncertainty on the energy availability is known to be NP-complete and considered as one of the most challenging problems in parallel computing (Michael and David, 1979). To address these challenges, we propose a dynamic scheduling framework for REEHMSs.

4.1 Static Task Assignment Heuristic

In this section, the energy-based utilization (EBU) scheduling algorithm is described. Energy-based utilization achieves proportional energy sharing among tasks by managing energy as the first-class resource and scheduling tasks based on their energy consumption. In this paper, the energy is correlated to the period by introducing the concept of energy utilization. The energy utilization, U_{e_i}, is defined as the worst case energy consumption WCEC of task τ_i E_{n_i} divided by its deadline D_i.

$$U_{e_i} = \frac{En_i}{D_i} \tag{2}$$

The EBU selects initially the task with lowest energy utilization to be scheduled on a processor in *Sys* according to the Best Fit heuristic. Therefore, *Sys* is sorted in decreasing order of the energy availability in the storage, $E_{B_j}(t)$. The task τ_i is assigned to processor P_j if the task set ψ_j is schedulable according to EDF, and the storage B_j has the least residual energy. Ideally, the residual energy in the storage should be zero. Each task set ψ_j assigned to processor P_j should satisfy the following constraints:

Real-time Constraints:

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1 \tag{3}$$

Energy Harvesting Constraints:

$$\sum_{i=1}^{n} U_{e_i} \leq P_{h_j} \tag{4}$$

4.2 Real-time Energy Harvesting Multiprocessor Systems Reconfiguration

Reconfigurable computing systems greatly have the ability to respond to environmental changes including hardware/software defects, resource changes, and non-continual feature usage. Such as, in this paper the flexibility and adaptability is modeled by the addition of new tasks to the system in order to dynamically adapt the system's behavior to outside stimuli in accordance with user requirements. Thereafter, the occurrence of such unpredictable external events may involves the system toward an infeasible state, where the real-time and energy constraints may be violated, task's deadlines missed. Such situation involves an adequate reconfiguration scenario so as to cope with online modification of the processing load. We propose a reconfiguration procedure based on software hardware reconfiguration to re-establish the feasibility in the whole system. Figure 2 explains the proposed reconfiguration procedure step-by-step.

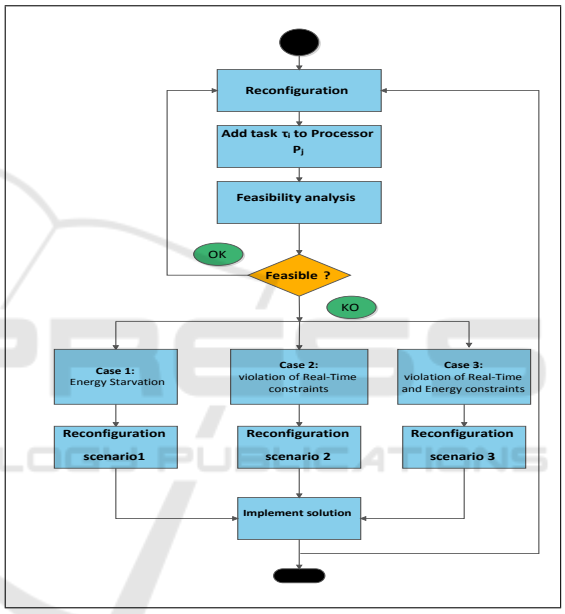

Figure 2: Activity diagram of the reconfiguration strategy.

4.2.1 Energy-based Migration

We identify three cases of failure:
Case 1: The real-time constraints are satisfied but the energy constraints are violated.
Case 2: The energy constraints are satisfied but the real-time constraints are violated.
Case 3: Both the real-time and the energy constraints are violated.
In order to deal with the three forms of failure, we propose a hardware/software reconfiguration scenario to each case in the purpose to re-establish the system feasibility. The basic idea is to define a lower utilization threshold for processors and keep the total utilization of the CPU by all the tasks allocated to the processor between the lower utilization threshold and the schedulability utilization U_x equals to 1 defined by

EDF.
Reconfiguration scenario 1: We propose to compare the CPU utilization to a predefined threshold. If the CPU utilization of the faulty processor falls below the lower threshold, the task set has to be migrated from this processor and the processor has to be switched to the idle mode for recharging the storage unit.
Reconfiguration scenario 2: If the CPU utilization exceeds 1, some tasks have to be migrated from the faulty processor to reduce the utilization factor.
Reconfiguration scenario 3: The task set has to be migrated from the processor which has to be switched to the idle mode for recharging the storage unit.
The Energy-Based Migration algorithm is carried out in two steps: i) At the first step: selection of tasks to be migrated, ii) At the second step: processor selection where migrated tasks will be affected.

4.2.2 Tasks Selection Policy

The proposed policy selects the minimum number of tasks needed to migrate from a faulty processor to the lower CPU utilization below the schedulability utilization U_x threshold 1. The algorithm sorts the list of tasks affected to the faulty processor in the decreasing order of the CPU utilization. Then, it selects a task which satisfies two conditions. First, the task should have the utilization higher than the difference between the processor utilization and the schedulability utilization U_x equals to 1. Second, the task provides when migrated, the minimum difference between the schedulability utilization U_x and the new utilization across the values provided by all the tasks. If there is no such a task, the algorithm selects the task with the highest utilization. The algorithm stops when the new processor utilization is below the schedulability utilization U_x.

5 APPROACH IMPLEMENTATION WITH POSIX AND INTEGRATION INTO A LINUX BASED SYSTEM

We propose to develop a REHLIB: Reconfigurable Energy Harvesting Library based on POSIX-Implementation in the purpose to enhance the real-time services of existing versions of Linux, such as RTLinux (Masmano et al., 2009) or Linux/RTAI (Mantegazza et al., 2000). The REHLIB is composed of two components real-time scheduling with energy harvesting requirements, and the reconfiguration management components. Figure 3 depicts the

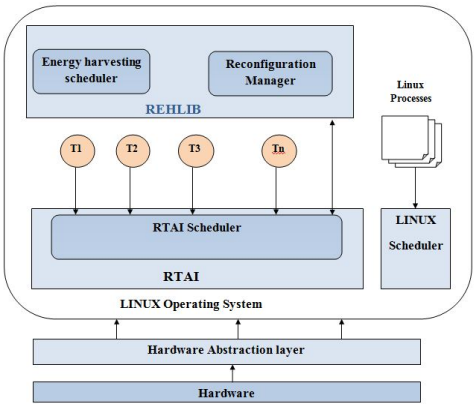

Figure 3: REHLIB Framework.

architecture of the proposed framework REHLIB.

5.1 Middleware Design

In order to model the reconfiguration scenarios in a real-time REEHMS, a metamodel is developed by using UML. We propose in Figure 4 the class diagram for the proposed middleware to be composed of three layers: i) Software layer, ii) Reconfiguration layer, and iii) Hardware layer.
Software Layer: It presents the ERT_task class which inherits the pthread_attr_t. The ERT_task is specified through the real-time parameters: release time, period, deadline, worst case execution time, worst case energy consumption, degree of criticality, and cpu_set.
Reconfiguration Layer: The role of the reconfiguration layer is to guarantee feasible executions of the real-time task set on the whole system. The ERT_schedule class permits to schedule the ERT_task according to the proposed energy harvesting scheduling algorithm. The reconfiguration event class presents the different unpredictable external reconfiguration events that can violate the real-time and/or the energy constraints. The reconfiguration manger class permits to re-establish the system feasibility by applying a hardware/software reconfiguration.
Hardware Layer: It contains three classes: processor, battery, and memory. These classes represent the hardware components physically implemented in the platform.

5.2 Basic Structures

We propose to create a new ERT_thread that extend the POSIX thread. Therefore, we define a new structure of the proposed ERT_thread, which contains the attributes for every thread as described as follows:
typedef struct thread_att_ERT thread_att_ERT;

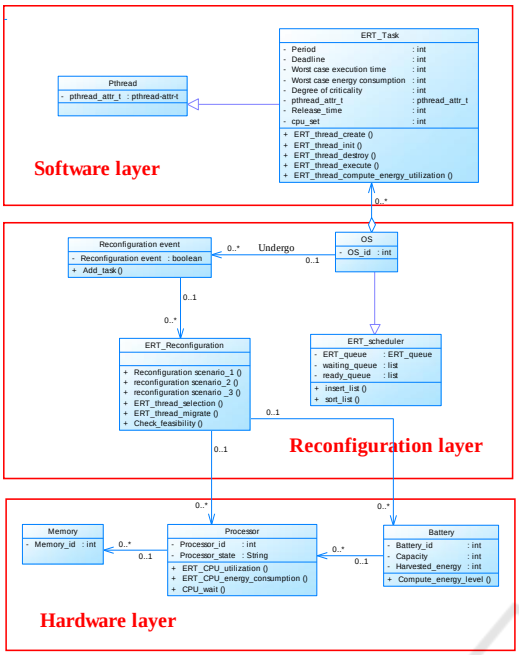

Figure 4: Class diagram of REHLib architecture.

typedef struct thread_attr_ERT { pthread_attr_t * pthread_attr_t; int Period; int Deadline; int Worst Case Execution Time; int Worst Case Energy Consumption; int Degree of Criticality;
} thread_att_ERT;
At initialization time, the designer has to set the thread attributes, the period T_i, deadline D_i, worst case execution time WCET C_i, worst case energy consumption WCEC En_i, and degree of criticality d_{c_i}. We define a data structure ERT_queue consists of a waiting_queue and a release_queue (as well as one lock per queue).
typedef struct ERT_queue ERT_queue;
typedef struct ERT_queue { waiting_queue * waiting_queue; release_queue * release_queue; }ERT_queue;
In addition we define new functions to extend the POSIX pthread Lib:
ERT_thread_create: create a new ERT_thread.
ERT_thread_init: initialize thread attributes.
ERT_thread_destroy: destroys ERT_thread attributes.
ERT_thread_execute: execute ERT_thread.
CPU_wait: put the processor on idle state.
ERT_thread_compute_energy_utilization(): compute the energy utilization of the corresponding ERT_thread.
ERT_CPU_utilization(): compute the processor utilization.
ERT_CPU_energy_consumption(): compute the total processor energy consumption.
Compute_energy_Level(t, B_j): compute the amount of energy available in the storage unit B_j corresponding to processor P_j at time t.
In order to manage the waiting_queue and release_queue, we define new functions:
insert_list(): insert an element into a sorted list.
sort_list(): sort lists.

5.3 Energy Harvesting Scheduler

In order to implement the task assignment algorithm (EBU) for the task set in the SMP architecture *Sys*, we create a new function EBU_ERT_thread_allocation(). The list of waiting tasks (waiting_queue) is sorted in increasing order of energy utilization. Each task is allocated to a processor in *Sys*, according to EBU_ERT_thread_allocation() algorithm. Therefore, we use the **Processor affinity**, in order to enable task allocation according to energy availability in each storage associated to each processor in *Sys*. Processor affinity is a modification of the native central queue scheduling algorithm. Each task (be it process or thread) in the queue has a tag indicating its preferred / kin processor. Algorithm 1 depicts the task assignment EBU_ERT_thread_allocation() algorithm.

Algorithm 1: ERT_scheduler algorithm.

```
ERT_thread ERT_thread; cpu_set_t cpuset;
ERT_queue ERT_queue;
while (ERT_thread=next(ERT_queue −>
waiting_queue)=!0) do
    if cpu != −1 then
        if ERT_CPU_utilization(cpu) ≤ 1 ||
          ERT_CPU_energy_consumption(cpu)
          ≤ E_Bj then
            if Compute_energy_level(cpu) >
              Max_Energy then
                CPU_ZERO(&cpuset);
                CPU_SET( cpu , &cpuset);
                pthread_setaffinity_np(0,
                  sizeof(cpu_set_t), &cpuset);
                  pthread_setschedparam(
                pthread_t target_thread, int
                  policy, sched_EDF);
            end
        end
    end
end
```

The scheduling of the proposed algorithm is performed in the ERT_schedule() routine. The scheduling occurs on timer handler activation. In the implementation the scheduler maintains the two lists waiting_queue and a release_queue (as well as one lock

per queue). ERT_schedule() routine attempts to release tasks from the waiting_queue list. The task set is executed according to EDF policy.

5.4 Reconfiguration Manager

The role of the reconfiguration manager is to guarantee feasible executions of the real-time task set on the whole SMP architecture. Therefore, when external events occur at run-time which add new real-time tasks to be executed on particular processors, the system may evolves towards an infeasible state. In section 4, we identified three cases of failure on a faulty processor. We develop a **ERT_reconfiguration()** routine for the management of reconfiguration scenarios (Algorithm 2). For the implementation of **ERT_reconfiguration()** routine, we use cpu_wait to put the processor in idle mode and **ERT_thread_execute** to put the processor in active mode. For the task migration we develop the function **ERT_thread_Migrate()**.

Algorithm 2: ERT_reconfiguration algorithm

```
ERT_thread ERT_thread; cpu_set_t cpuset;
 ERT_queue ERT_queue;
if ERT_thread_create() &&
 CPU_ZERO(&cpuset);
CPU_SET( cpu , &cpuset);
pthread_setaffinity_np(0, sizeof(cpu_set_t),
 &cpuset) then
    if (ERT_CPU_utilization(cpu) ≤ 1 &&
     ERT_CPU_energy_consumption(cpu) >
     E_Bj ) then
        if (ERT_CPU_utilization(cpu) < thr)
         then
            ERT_thread_Migrate();
            CPU_wait();
        end
    end
end
if (ERT_CPU_utilization(cpu) > 1 &&
 ERT_CPU_energy_consumption(cpu) < E_Bj)
 then
    ERT_thread_selection();
     ERT_thread_Migrate();
end
if (ERT_CPU_utilization(cpu) > 1 &&
 ERT_CPU_energy_consumption(cpu) > E_Bj)
 then
    ERT_thread_Migrate();
    CPU_wait();
end
ERT_thread_execute;
```

6 EXPERIMENTATION

The performance of the proposed operating system was studied on a 2.50 GHz core i5. In order to assess the performance of the proposed approach, we perform two sets of experiments. First, we evaluate the effectiveness of the proposed approach in term of deadline miss ratio when compared with state-of-the-art techniques. Second, we evaluate the implementation of the proposed REHLib to make a quantitative evaluation of the overhead introduced by the ERT_reconfiguration schedulers. In order to evaluate the performance of the proposed approach in term of deadline success ratio, we consider a set of 50 tasks. We assume a set of unpredictable reconfiguration scenarios applied repeatedly at run-time. Each scenario adds a set of n tasks such that n is randomly chosen between 10 and 50. The period of each task is chosen randomly between 10 and 100. The WCET is randomly chosen between 6 and 20. In the sequel, we compare the proposed approach with (Ghor et al., 2011).

Figure 5 clearly shows that EDeg and ERT algorithms provide the same performance when processor utilization below 0.9. However, the performance of EDeg algorithm degrades significantly when the processor utilization increases upper 0.9 due to a set of reconfiguration scenarios applied repeatedly at run-time. One of the most important performance evaluation is the overhead introduced by the ERT schedulers. By definition, the overhead of an operating system represents the time lost in handling all kernel mechanisms, such as context-switching overhead, task scheduling management overhead and so on. In this experiment, we take interest in the overhead caused by the ERT_reconfiguration routine by calculating the ERT_CPU_energy_consumption(), ERT_CPU_utilization(), ERT_thread_selection. In order to assess the introduced overhead we consider different task set where the number of tasks varying in each set. The considered tasks are generated with a hyperperiod of 3360 ticks and with periods of 10 milliseconds each one. From Figure 6 we can drive that the overhead scales with the number of tasks. However, this time is very low when compared with the whole measurement period with maximum overhead of 8 %.

7 CONCLUSION AND FUTURE WORK

In this paper, we proposed first a new static best fit task partitioning algorithm based on energy utiliza-

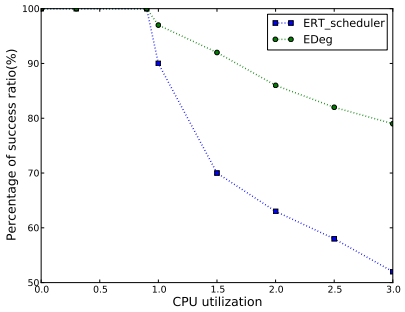

Figure 5: Comparison of deadline success ratio.

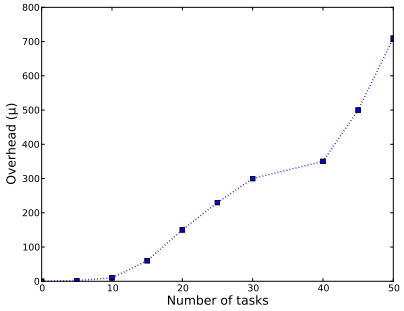

Figure 6: Dynamic overhead of ERT_reconfiguration scheduler.

tion of tasks called BFEBU performed offline. Then, we proposed a dynamic adaption strategy applied online to perform automatic dynamic reconfiguration scenarios to deal with unpredictable external events from the environment or hardware failures, in the purpose to guarantee a feasible executions in the whole multiprocessor system. We identified three cases of processor infeasibility: i) Processor overload, ii) Energy starvation, and iii) Both processor overload and energy starvation. We consider two reconfiguration scenarios: a) Software reconfiguration which consist to migrate tasks from one faulty processor to a non faulty one, and b) Hardware reconfiguration which consists to switch the faulty processor to the idle mode for recharging the storage unit. We developed a new library for reconfigurable energy harvesting multiprocessor systems called REHLib based on POSIX-implementation. The developed REHLib library implement the proposed approach and consists of two software components: reconfiguration manager and energy harvesting scheduler. The proposed approach is assessed from two aspects energy gain, and deadline success ratio. Extensive simulation experiments show the effectiveness of the proposed approach compared with previous works in terms of the percentage of deadline success ratio. A simulation study evaluate the impact of overheads on the relative performance of the proposed approach. The authors are now working on the development of a simulation tool for the reconfigurable real-time energy harvesting multiprocessor systems.

REFERENCES

Allavena, A. and Mossé, D. (2001). Scheduling of frame-based embedded systems with rechargeable batteries. In *Workshop on Power Management for Real-time and Embedded systems (in conjunction with RTAS 2001)*.

Barabanov, M. and Yodaiken, V. (1996). Real-time linux. *Linux journal*, 23(4.2):1.

Calandrino, J. M., Leontyev, H., Block, A., Devi, U. C., and Anderson, J. H. (2006). Litmusˆ rt: A testbed for empirically comparing real-time multiprocessor schedulers. In *Real-Time Systems Symposium, 2006. RTSS'06. 27th IEEE International*, pages 111–126. IEEE.

Chetto, M. (2014). Optimal scheduling for real-time jobs in energy harvesting computing systems. *IEEE Trans. Emerging Topics in Computing*, 2(2):122–133.

Gerum, P. (2004). Xenomai-implementing a rtos emulation framework on gnu/linux. *White Paper, Xenomai*, page 81.

Ghor, H. E., Chetto, M., and Chehade, R. H. (2011). A real-time scheduling framework for embedded systems with environmental energy harvesting. *Computers & Electrical Engineering*, 37(4):498–510.

Grichi, H., Mosbahi, O., and Khalgui, M. (2015). Rocl: New extensions to ocl for useful verification of flexible software systems. In *Software Technologies (ICSOFT), 2015 10th International Joint Conference on*, volume 1, pages 1–8. IEEE.

Housseyni, W., Mosbahi, O., Khalgui, M., and Chetto, M. (2015). Real-time task reconfiguration in energy-harvesting based multiprocessor systems. In *29th European Simulation and Modelling Conference-ESM'2015*.

Housseyni, W., Mosbahi, O., Khalgui, M., and Chetto, M. (2016). Real-time scheduling of reconfigurable distributed embedded systems with energy harvesting prediction. In *Distributed Simulation and Real Time Applications (DS-RT), 2016 IEEE/ACM 20th International Symposium on*, pages 145–152. IEEE.

Khemaissia, I., Mosbahi, O., Khalgui, M., and Bouzayen, W. (2014). New reconfigurable middleware for feasible adaptive rt-linux. In *PECCS*, pages 158–167.

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

Lu, J. and Qiu, Q. (2011). Scheduling and mapping of periodic tasks on multi-core embedded systems with energy harvesting. In *Green Computing Conference and Workshops (IGCC), 2011 International*, pages 1–6. IEEE.

Mantegazza, P., Dozio, E., and Papacharalambous, S. (2000). Rtai: Real time application interface. *Linux Journal*, 2000(72es):10.

Masmano, M., Ripoll, I., Crespo, A., and Metge, J. (2009). Xtratum: a hypervisor for safety critical embedded systems. In *11th Real-Time Linux Workshop*, pages 263–272. Citeseer.

Michael, R. G. and David, S. J. ((1979)). Computers and intractability: a guide to the theory of np-completeness. *WH Freeman and Co., San Francisco.*

Mueller, F. et al. (1993). A library implementation of posix threads under unix. In *USENIX Winter*, pages 29–42.

Wang, X., Khalgui, M., and Li, Z. (2010). Dynamic low power reconfigurations of embedded real-time systems. In *Proceedings of the 1st International Conference on Pervasive and Embedded Computing and Communication Systems, Portugal*, volume 6. Citeseer.

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 Trans. 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 Trans. Industrial Informatics*, 12(1):101–111.

Wei, T., Guo, Y., Chen, X., and Hu, S. (2010). Adaptive task allocation for multiprocessor socs. In *Quality Electronic Design (ISQED), 2010 11th International Symposium on*, pages 538–543. IEEE.