ARTE: a Simulation Tool for Reconfigurable Energy Harvesting Real-time Embedded Systems

Wiem Housseyni, Maryline Chetto

ARTE: a Simulation Tool for Reconfigurable Energy Harvesting Real-time Embedded Systems

Wiem HOUSSEYNI[1,2,3] and Maryline CHETTO[1]

[1] University of Nantes, LS2N UMR CNRS 6597 44321 Nantes, France
[2] LISI Laboratory, INSAT, University Of Carthage, Tunisia
[3] Tunisia Polytechnic School, University Of Carthage, Tunisia
maryline.chetto@univ-nantes.fr
wiem.housseyni@univ-nantes.fr

Abstract. This report presents ARTE a new free Java based software, which we have developed for simulation and evaluation of reconfigurable energy harvesting real-time embedded systems. It is designed in the aim to generate, compare, and evaluate different real-time scheduling strategies on their schedulability performance as well as energy efficiency in the presence of external unpredictable reconfiguration scenarios. A reconfiguration scenario is defined as unpredictable events from the environment such as addition-remove of software tasks, and increase-decrease of the power rate. The occurrence of such events may evolve the system towards an overload state. For this purpose, ARTE provides a task sets generator, reconfiguration scenarios generator, and a simulator. It also offers to the designer the possibility to generate and run simulation for specific systems. In the actual version, ARTE can simulate the execution of periodic independent and (m,k)-firm constrained task sets on monoprocessor architecture. The energy consumption is considered as a scheduling parameter in the same manner as Worst Case Execution Time (WCET).

Table of Contents

1 Introduction

Since decades, the literature has revealed a substantial interest in the real-time scheduling problem, where several approaches and algorithms have been ported over the years to optimize the scheduling of real-time tasks on single and multiple processor systems. The energy constraint has emerged as a major issue in the design of such systems, despite there are big research efforts addressing this problem. Research works have focused on either dynamic energy awareness algorithm in order to reduce system energy consumption or energy harvesting technology. In order to enhance the lifespan and to achieve energy autonomy in such system, there is a tremendous interest in the energy harvesting technologies recently.

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 the 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 energy harvesting real-time systems are not mature enough to either be applicable or optimal as much as currently available uni-processor real-time scheduling techniques.

Nowadays, new criteria such as energy efficiency, high-performance and flexible computing arises the need of new generation of real-time systems. As a result, such systems need to solve a number of problems that so far have not been addressed by traditional real-time systems. Reconfigurable systems are solutions to providing both higher energy efficiency, and high-performance and flexibility. In recent years, a substantial number of research works from both academia and industry have been made to develop reconfigurable control systems [1], [2]. Reconfiguration is usually performed in response to both user requirements and dynamic changes in its environment such as unpredictable arrival of new tasks, and hardware or software failures. Some examples of such systems are multi-robot systems [3], and wireless sensor networks [4]. From the literature we can drive different definitions of a reconfigurable system. The authors of [5] 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. In order to check whether a set of tasks respects its temporal constraints when a scheduling algorithm is used or to evaluate the efficiency of a new approach, software simulation against other algorithms is considered as a valid comparison technique and it is commonly used in the evaluation of real-time systems. Over the last years, several simulation tools have been ported MAST [6], CHEDDAR [7], STORM [8], FORTAS [9] and YARTISS [10]. However, none of these approaches provide support for reconfigurable based applications, yet. More recently, new simulation tools are proposed for reconfigurable computing systems Reconf-Pack [15]. However, this attempt do not support energy harvesting requirements.

New challenges raised by reconfigurable real-time embedded energy harvesting systems in terms of reconfigurability, scheduling, and energy harvesting management. Indeed, in such a context, it is very difficult to evaluate and compare scheduling algorithms on their schedulability performance as well as energy efficiency. Unlike any prior work, we formulate and solve the challenging problem of scheduling real-time applications on reconfigurable energy-harvesting embedded system platforms. Based on these motivations, we investigate in this report the challenges and viability of designing an open and flexible simulation tool able to generate and simulate accurately the behavior of such systems.
In this report, we introduce ARTE, a new simulation tool for reconfigurable energy harvesting real-time embedded systems, which provides various functions to simulate the scheduling process of real-time task sets and their temporal behavior when a scheduling policy is used. It provides the classical real-time scheduling policy EDF, the optimal scheduling algorithm for energy harvesting systems EDH, EDF scheduling for (m,k)-constrained task sets, and a new scheduling policy which is an extension of EDH for (m,k)-firm constrained task sets EDH-MK, and finally a new hierarchical approach Reconf-Algorithm. It implements also classical and new feasibility tests for both real-time and energy harvesting requirements. The main aim of this research work is to guarantee a feasible execution in the whole system in the presence of unpredictable events while satisfying a quality of service (QoS) measured first in term of the percentage of satisfied deadline, second the percentage of satisfied deadline while considering the degree of importance of tasks, and finally the overhead introduced by the proposed approach.

The remainder of this report is as follows: we review related works and examples of real-time simulators in Section 2. Section 3 presents a background about the EDF, EDH scheduling algorithm, and the (m,k)-firm model. In section 4 we detail the proposed new scheduling approach for reconfigurable energy harvesting real-time systems. Then in Section 5 we present the various functionalities of our simulation tool. A case study is given in Section 6. Finally, we discuss our future work in Section 7 and Section 8 brings a conclusion to this report.

2 Research Aims

New challenges raised by reconfigurable real-time embedded energy harvesting systems in terms of reconfigurability, scheduling, and energy harvesting management. Such systems work in dynamic environment where unpredictable events may occur arrival-remove of software tasks or increase-decrease of power rate. However, when unpredictable events from the environment occur, the system may evolve towards an unfeasible state (processor/energy overload) and the actual operation mode is no longer optimal. For this aim, the main focus of this research is on how to achieve system feasibility while satisfying a graceful QoS degradation. For this purpose we define three operating modes:

- **Normal mode**: where all the tasks in the system execute 100% of their instances while meeting all the deadlines
- **Degradation mode level 1**: This is the case where the normal mode is not feasible. The K least important tasks execute in degraded mode according to the model (m,k)-firm and other tasks execute normally. The schedulability test is performed by considering iteratively the tasks according to their importance.
- **Degradation mode level 2**. This is the case where the degradation mode level 1 is not feasible. Abandonable tasks are gradually eliminated by increasing importance.

When a system executing under an operating mode and an external event occurs it is imperative to verify the schedubility test. We identify three cases:

- The system may continue to execute in the actual operating mode,
- the system may be executed under degraded mode,
- the system may executed under normal mode.

The quality of service (QoS) is measured in term of:

- The percentage of satisfied deadline,
- the percentage of satisfied deadline while considering the degree of importance of tasks,
- the overhead introduced by the proposed approach.

3 Related Works

In this section we outline the existing works related to simulation of the scheduling of real-time tasks.

There are a lot of tools to test and visualize the temporal and execution behavior of real-time systems, and they are divided mainly into two categories: the execution analyzer frameworks and the simulation software. MAST [6] is a modeling and analysis suite for real-time applications that is developed in 2000. MAST is an event-driven scheduling simulator that permits modeling of distributed real-time systems and offers a set of tools to e.g. test their feasibility or perform sensitive analysis. Another known simulator is Cheddar [20, 19] which is developed in 2004 and it handles the scheduling of real-time tasks on multiprocessor systems. It provides many implementations of scheduling, partitioning and analysis of algorithms, and it comes with a friendly Graphical User Interface (GUI). Unfortunately, no API documentation is available to help with the implementation of new algorithms and to facilitate its extensibility. Moreover, Cheddar is written in Ada programming language [14] which is used mainly in embedded systems and it has strong features such as modularity mechanisms and parallel processing. Ada is often the language of choice for large systems that require real-time processing, but in general, it is not a common language among developers. We believe that the choice of Ada as the base of Cheddar reduces the potential contributions to the software from average developers and researchers. Finally, STORM [8] FORTAS [9], and YARTISS [10] are tools which are written in Java. In 2009, STORM is released and it is described as a simulation tool for Real time multiprocessor scheduling. It has modular architectures (both software and hardware) which simulate the scheduling of task sets on multiprocessor systems based on the rules of a chosen scheduling policy. The specifications of the simulation parameters and the scheduling policies are modifiable using an XML file. However, the simulator tool lacks a detailed documentation and description of the available scheduling methods and the features of the software. On the other hand, FORTAS is a real-time simulation and analysis framework which targets uniprocessor and multiprocessor systems. It is developed mainly to facilitate the comparison process between the different scheduling algorithms, and it includes features such as task generators and computation of results of each tested and compared scheduling algorithm. FORTAS represents valuable contributions in the effort towards providing an open and modular tool. Unfortunately, it seems to suffer from the following issues: its development is not open to other developers for now, we can only download .class files, no documentation is yet provided and it seems that no new version has been released to public since its presentation in [9]. More recently, YARTISS is proposed as a modular and

extensible tool. It is a real-time multiprocessor scheduling simulator which provides various functions to simulate the scheduling process of real-time task sets and their temporal behavior when a scheduling policy is used. Functionalities of YARTISS: 1) simulate a task set on one or several processors while monitoring the system energy consumption, 2) concurrently simulate a large number of tasksets and present the results in a user friendly way that permits us to isolate interesting cases, and 3) randomly generate a large number of task sets. However, none of these simulation tools provide support for unpredictable reconfiguration scenarios yet. To date, only a few recent works target the real-time scheduling issue in reconfigurable systems. In [15] a simulator tool is proposed for Reconfigurable Battery-Powered Real-Time Systems Reconf-Pack. Reconf-Pack is a simulation tool for analyzing a reconfiguration and applying the proposed strategy for real-time systems. It is based upon another tool Task-Generator which generates random tasks. According to the state of the system after a reconfiguration, Reconf-Pack calculates dynamically a deterministic solution. Moreover, it compares the pack-based solutions to related works. However, it seems to suffer from the following issues: its development is not open to other developers for now, and isn't available for download.

During the development of ARTE, we learned from those existing tools and we included some of their features in addition to others of our own. Our aim is to provide a simulation tool for reconfigurable energy harvesting real-time systems that is easily it can be used to generate, compare, simulate the scheduling of real-time tasks on reconfigurable energy harvesting real-time systems.

4 Background

This section gives a background about first the EDF scheduling algorithm, then EDH scheduling algorithm, and finally the (M,K)-model.

4.1 Earliest Deadline First scheduling EDF

EDF is probably the most famous dynamic priority scheduler. As a consequence of its optimality for preemptive uniprocessor scheduling of independent jobs, the run-time scheduling problem is perfectly solved if we assume there exists no additional constraints on the jobs. EDF is the scheduler of choice since any feasible set of jobs is guaranteed to have a valid EDF schedule [18].
Time-feasibility: A set ψ of n tasks $\tau_i = \{C_i, T_i, E_i\}$ is time feasible if and only if

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

However, when considering energy harvesting requirements EDF is no longer optimal and a necessary condition but non sufficient of schedulability is as follows:
Time-feasibility: A set ψ of n tasks is time feasible if and only if

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1 \tag{2}$$

Energy-feasibility: A set ψ of n tasks is energy feasible if

$$\frac{\sum_{i=1}^{n} E_i}{C + E_P(0,t)} \leq 1 \tag{3}$$

Let E_p(0,t) be the amount of energy that will be produced by the source between 0 and t and C is the energy storage unit (supercapacitor or battery) capacity.

4.2 Earliest Deadline First-Energy Harvesting EDH

Dertouzos [16] shows that the Earliest Deadline First Algorithm (EDF) is optimal. EDF schedules at each instant of time t, that job ready for execution whose deadline is closest to t. But the problem with EDF is that it does not consider future jobs arrivals and their energy requirements. In [17], the authors prove that EDF is no longer optimal for RTEH systems. Jobs are processed as soon as possible thus consuming the available energy greedily.Although noncompetitive, EDF turns out to remain the best non-idling scheduler for uniprocessor RTEH platforms [17].
In [18], the authors describe the Earliest Deadline-Harvesting ED-H scheduling algorithm which is proved to be optimal for the scheduling problem of Energy-Harvesting Real-time systems. ED-H is an extension of the EDF algorithm that adds energy awareness capabilities by using the notion of slack-time and slack-energy. The idea behind ED-H is to order the jobs according to the EDF rule since the jobs issued from the periodic tasks have hard deadlines. Executing them in accordance with their relative urgency appears to be the best approach even if they are not systematically executed as soon as possible due to possible energy shortage. The difference between ED-H and classical EDF is to decide when to execute a job and when to let the processor idle. Before authorizing any job to execute, the energy level of the storage must be sufficient so that all future occurring jobs execute timely with no energy starvation, considering their timing and energy requirements and the replenishment rate of the storage unit. According to EDH a processor P_j and battery B_j, that performs tasks set ψ_j should satisfy the schedulability tests described follows:

- **Time-feasibility**: [18] ψ_j is time feasible if and only if

$$U_{P_j} = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1 \tag{4}$$

- **Energy-feasibility**: [18] ψ_j is energy feasible if and only if

$$Ue_j \leq 1 \tag{5}$$

  Ue_j is the energy load of the set of tasks ψ_j assigned to processor P_j

$$Ue_j = \sup_{0 \leq t_1 \leq t_2 \leq H} \frac{E_{c_j}(t_1, t_2)}{C + E_{P_j}(t_1, t_2)}$$

Theorem 1 [18] The set of tasks ψ_j assigned to processor P_j is feasible if and only if

$$U_{P_j} \leq 1 \; and \; Ue_j \leq 1.$$

Theorem 1 gives a necessary and sufficient condition of schedulability.

4.3 (M,K)-model

For its intuitiveness and capability of capturing not only statistical but also deterministic quality of service QoS requirements, the *(m,k)*-model has been widely studied, e.g., [19], [20], [21], [22], and [23]. The *(m,k)*-model was originally proposed by Hamdaoui et al. [20]. According to this model, a repetitive task of the system is associated with an *(m,k)* (0<m<k) constraint requiring that m out of any k consecutive job instances of the task meet their deadlines. A dynamic failure occurs, which implies that the temporal quality of service QoS constraint is violated and the scheduler is thus

considered failed, if, within any k consecutive jobs, more than *(k-m)* job instances miss their deadlines. Based on this *(m,k)*-model, Ramanathan et al. [23] proposed to partition the jobs into mandatory and optional jobs. So long as all of the mandatory jobs can meet their deadlines, the *(m,k)*-constraints can be ensured. The mandatory jobs are the jobs that must meet their deadlines in order to satisfy the -constraints, while the optional jobs can be executed to further improve the quality of the service or simply be dropped to save computing resources. Quan et al. [22] formally proved that the problem of scheduling with the *(m,k)*-guarantee for an arbitrary value of $m$and k is NP-hard in the strong sense. They further proposed to improve the mandatory/optional partitioning by reducing the maximal interference between mandatory jobs.

4.4 Mandatory/Optional Job Partitioning With (M,K)-Pattern

The (m,k)-pattern of task τ_i, denoted by Π_i, is a binary string $\Pi_i = \{\pi_{i0}, \pi_{i1}\pi_{i(k_i-1)})\}$ which satisfies the following: 1) π_{ij} is a mandatory job if $\pi_{ij} = 1$ and optional if $\pi_{ij} = 0$ and 2)$\sum_{j=0}^{k_i-1} \pi_{ij} = m_i$.

By repeating the (m,k)-pattern , we get a mandatory job pattern for τ_i. It is not difficult to see that the (m,k)-constraint for τ_i can be satisfied if the mandatory jobs of τ_i are selected accordingly.

Evenly Distributed Pattern (Even Pattern) Even Pattern strategy was proposed by Ramanathan et al. [23] as follows: the first release is always mandatory and subsequent distribution of mandatory and optional alternating. Mathematically,

$$\Pi_j^i = \begin{cases} 1, if j = \lfloor \lceil \frac{j \times m_i}{k_i} \rceil \times \frac{k_i}{m_i} \rfloor, for j = 0, 1, .., k_i \\ 0, otherwise \end{cases} \tag{6}$$

In [24] a necessary and sufficient feasibility test for (m,k)-constrained tasks executing under EDF scheduling policy is proposed.

Theorem 2: Let system T = $\{\tau_0, \tau_i, .., \tau_{n-1}\}$, where τ_i=$\{C_i, T_i, D_i, m_i, k_i\}$ and Ψ be the mandatory job set according to their E-patterns. Also, let L represent either the ending point of the first busy period when scheduling only the mandatory jobs or the LCM of T(i) , i = 0, ..., (n-1), whichever is smaller. Then, Ψ is schedulable with EDF if and only if (iff) all the mandatory jobs arriving within [0, L] can meet their deadlines, i.e.,

$$\sum_i W_i(0,t) = \sum_i (\lceil \frac{m_i}{k_i} \times \lceil \frac{t - D_i}{T_i} \rceil^+ \rceil) \times C_i \leq t \tag{7}$$

5 New Scheduling Approach for Reconfigurable Energy Harvesting Real-Time Systems

Reconfiguration is usually performed in response to both user requirements and dynamic changes in its environment such as unpredictable activation of new tasks, removal of tasks, or increase-decrease of power supply of the system. Some examples of reconfigurable systems are multi-robot systems [3] and wireless sensor networks [4]. At run-time, the occurrence of unpredictable task's activation makes the static schedule no longer optimal and may evolve the system towards an unfeasible state due to energy and processing overloads. Thereafter, some existing or added tasks may violate

deadlines. The system has to dynamically adjust and adapt the task allocation and scheduling in order to cope with unpredictable reconfiguration scenarios. We identify mainly two function modes:

- **Normal mode**: where all the tasks in the system execute 100% of their instances while meeting all the deadlines
- **Degradation mode level 1**: This is the case where the normal mode is not feasible. The K least important tasks execute in degraded mode according to the model (m,k)-firm and other tasks execute normally. The schedulability test is performed by considering iteratively the tasks according to their importance.
- **Degradation mode level 2**. This is the case where the degradation mode level 1 is not feasible. Abandonable tasks are gradually eliminated by increasing importance.

At any instant, external unpredictable reconfigurations may occur to add or remove software tasks or to increase-decrease power supply on the system. The occurrence of such events provoke the execution of schedulability tests to identify in which mode the tasks should be executed.

5.1 Normal Mode

In the normal mode tasks are assumed to be executed under the optimal scheduler for real-time energy harvesting systems EDH algorithm. Then the tasks set should satisfy the following theorem:
Theorem 1 [18] The set of tasks ψ_j assigned to processor P_j is feasible if and only if

$$U_{P_j} \leq 1 \; and \; Ue_j \leq 1.$$

5.2 Degradation mode level 1: EDH-MK Algorithm

We propose in this work a new real-time scheduler for reconfigurale energy harvesting real-time systems EDH-MK. The proposed algorithm EDH-MK is an extension of the EDH algorithm with (m,k)-firm guarantee. When it is impossible to execute the tasks set in normal mode due to processor and/or energy overloads, we propose to execute the K least important tasks in degraded mode according to the model (m,k)-firm and other tasks execute normally. In this work we propose the necessary and sufficient schedulability conditions for the EDH-MK algorithm.
Definition 1: The static slack time of a mandatory job set Ψ on the time interval $[t_1, t_2)$ is

$$SST_{\Psi}(t_1, t_2) = t_1 - t_2 - \sum_i W(t_1, t_2) \tag{8}$$

$SST_{\Psi}(t_1, t_2)$ gives the longest time that could be made available within $[t_1, t_2)$ after executing mandatory jobs of Ψ with release time at or after t_1 and deadline at or before t_2.
Definition 2: The total mandatory energy demand within interval [0,t] is

$$g(0,t) = \sum_i (\lceil \frac{m_i}{k_i} \times \lfloor \frac{t}{T_i} \rfloor \rceil) \times En_i \tag{9}$$

Proof:
As shown [23] that, if the mandatory jobs are determined according to (6), for the first p_i jobs of τ_i, there are $l_i(t) = \lceil \frac{m_i}{k_i} \times p_i \rceil$ jobs that are mandatory. Therefore, the

total mandatory energy load within interval [0,t] that has to be finished by time t, denoted by g(0,t), can be formulated as follows:

$$g(0,t) = \sum_i (\lceil \frac{m_i}{k_i} \times \lceil \frac{t - D_i}{T_i} \rceil^+ \rceil) \times En_i \tag{10}$$

$$g(0,t) = \sum_i (\lceil \frac{m_i}{k_i} \times \lfloor \frac{t}{T_i} \rfloor \rceil) \times En_i \tag{11}$$

Let E_p(0,t) be the amount of energy that will be produced by the source between 0 and t and C is the energy storage unit (supercapacitor or battery) capacity.
Definition 3: The total mandatory static slack energy on the time interval [0, t] is

$$SSE_\Psi(0,t) = C + E_p(0,t) - g(0,t) \tag{12}$$

SSE_Ψ(0,t) gives the largest energy that could be made available within [0,t] after executing mandatory jobs with release time at or after 0 and deadline at or before t.
Definition 4: Let d be the deadline of the active job at current time t_c. The preemption slack energy of a mandatory job set Ψ at t_c is

$$PSE_\Psi(t_c) = \min_{t_c \le r_i \le d_i \le d} SE_{\tau_i}(t_c) \tag{13}$$

Lemma 1: If d_1 is missed in the EDH-MK schedule because of energy starvation there exists a time instant t such that g(t, d_1) > C + E_p(t, d_1) and no schedule exists where d_1 and all earlier deadlines are met.
Proof: Recall that we have to consider the energy starvation case where d_1 is missed with E(d_1) = 0. Let t_0 be the latest time before d_1 where a mandatory job with deadline after d_1 releases, no other mandatory job is ready just before t_0 and the energy storage unit is fully charged i.e. E(t_0) = C. The initialization time can be such time. The processor is idle within [t_0-1, t_0) since no mandatory jobs are ready. As no energy is wasted except when there are no ready jobs, the processor is busy at least from time t_0 to t_0 + 1. We consider two cases:
Case 1: No mandatory job with deadline after d_1 executes within [t_0, d_1). Consequently, all the mandatory jobs that execute within [t_0, d_1). have release time at or after t_0 and deadline at or before d_1. The amount of energy required by these mandatory jobs is g(t_0, d_1). As is feasible, g(t_0, d_1) is no more than the maximum storable energy plus all the incoming energy i.e., C + E_p(t_0, d_1). As E(t_0) = C, we conclude that all mandatory jobs ready within [t_0, d_1) can be executed with no energy starvation which contradicts the deadline violation at d_1 with E(d_1) = 0.
Case 2: At least one mandatory job with deadline after d_1 executes within [t_0, d_1). Let t_2 be the latest time where a mandatory job, say τ_2, with deadline after d_1 is executed. As d_1 is lower than d_2 and mandatory jobs are executed according to the earliest deadline rule in EDH-MK, we have $r_2 < r_1$. At time t_2, one of the following situations occurs.
Case 2a: The processor is busy all the times in [t_0, d_1). τ_2 is preempted by a higher priority job, say τ_3, with $d_3 \le d_1$. From rule 4.2 in [18], PSE_Ψ (r_3) > 0 which implies that SE_{τ_1} (r_3) > 0 and in consequence g(r_3, d_1) < E(r_3) + E_p(r_3, d_1). All mandatory jobs that are executed within [r_3, d_1) have release time at or after r_3 and deadline at or before d_1. Consequently, the amount of energy they require is at most g(r_3, d_1). That contradicts deadline violation and E(d_1) = 0.
Case 2b: The processor is idle in $[t_3 - 1, t_3)$ with $t_3 > t_2$ and busy all the times in [t_3, d_1). The processor stops idle at time t_3 imperatively by rule 4.1 [18] if E(t_3) = C. By hypothesis, there is no mandatory job waiting with deadline at or before d_1

at t_3 because t_0 is the latest one. Furthermore, no mandatory job with deadline after d_1 is executed after t_2 and consequently after t_3. In order not to waste energy, all the energy which arrives from the source is used to advance mandatory jobs with deadline after d_1. The processor continuously commutes from active state to inactive state. The storage is maintained at maximum level until τ_1 releases. Consequently, we have E(r_1) = C. As τ_1 is feasible, g(r_3, d_1) ≤ C + $E_p(r_1, d_1)$. Thus, E(r_1) + $E_p(r_1, d_1)$ ≥ g(r_1, d_1). That contradicts deadline violation and E(d_1) = 0.
Lemma 2: The set Ψ is energy-feasible if and only if

$$SSE_{\Psi} > 0 \tag{14}$$

Proof:
"**If**": Directly follows from Lemma 2.
"**Only If**": Since is energy-feasible, let us consider an energy-valid schedule produced within [0, d_{Max}). The amount of energy demanded in each interval of time [t_1, t_2), g(t_1, t_2), is necessarily less than or equal to the actual energy available in [t_1, t_2) given by E(t_1) + $E_p(t_1, t_2)$. An upper bound on E(t_1) is the maximum storable energy at time t_1, that is C. Consequently, g(t_1, t_2)is lower than or equal to C + $E_p(t_1, t_2)$. This leads to ∀ (t_1, t_2) ∈ [0, d_{Max}), g(t_1, t_2) ≤ C + $E_p(t_1, t_2)$ i.e. SSE (t_1, t_2) ≥ 0. Thus, SSE ≥ 0.
Lemma 3: Ψ is energy-feasible if and only if

$$UE_{\Psi} \leq 1 \tag{15}$$

Proof: As proof of Lemma since $SSE_{\Psi}(t_1, t_2) \leq 0$ amounts to $UE_{\Psi}(t_1, t_2) \leq 1$.
The necessary and sufficient schedulability conditions falls into two constraints which should be respected.

- **Real-time constraints**: For each processor P_j the tasks set Ψ_j assigned to P_j should satisfy their deadlines. From equation (7)
  Time-feasibility: The set Ψ_j is time feasible if and only if

$$\sum_i W_i(0,t) = \sum_i (\lceil \frac{m_i}{k_i} \times \lfloor \frac{1}{T_i} \rfloor \rceil) \times C_i \leq 1 \tag{16}$$

- **Energy constraints**: Each processor P_j must not, at any moment, lack energy to execute the tasks set assigned to processor P_j. Energy feasibility: P_j is energy feasible if and only if

$$UE_{\Psi} \leq 1 \tag{17}$$

We give a necessary and sufficient condition for EDH-MK schedulability and feasibility.
Theorem 3: Let Ψ be the mandatory job set according to their E-patterns. Also, let L represent either the ending point of the first busy period when scheduling only the mandatory jobs or the LCM of T(i) , i = 0, ..., (n-1), whichever is smaller. Then, Ψ is schedulable with EDH-MK if and only if (iff) all the mandatory jobs arriving within [0, L] can meet their deadlines, i.e.,

$$\sum_i W(0,t) \leq 1 and SSE_{\Psi} \leq 0 \tag{18}$$

Proof: "**If**": We suppose that constraint (17) is satisfied and Ψ is not schedulable by EDH-MK. Let us show a contradiction. First, we assume that Ψ is not schedulable by

EDH-MK because of time starvation. of energy starvation. Lemma 2 states that there exists a time interval $[t_0, d_1)$ such that $g(t_0, d_1) > \mathrm{C} + E_p(t_0, d_1)$ i.e., C + $E_p(t_0, d_1)$ - g$(t_0, d_1) < 0$. Thus, $SSE_{\Psi} < 0$ and condition 17 in Theorem 2 is violated.
"**Only if**": Suppose that Ψ is feasible. Thus, Ψ is time-feasible and energy feasible. From constraint (7) in theorem 2 and constraint (14) in Lemma 2, it is the case that constraint (17) is satisfied.

5.3 Degradation mode level 2: Removal Algorithm

This is the case where the degradation mode level 1 is not feasible. Abandonable tasks are gradually eliminated by increasing importance. We sort all the abandonable tasks in an ascending order of degree of importance such that we can reject those with less importance one by one until establishing the system feasibility.
Theorem 4: The set of tasks Ψ_j assigned to processor P_j is feasible under degradation mode level 2: Removal algorithm if and only if the set of non abandonable tasks set Ψ_j^{na}

$$U_{\Psi_j^{na}} \leq 1 \; and \; Ue_{\Psi_j^{na}} \leq 1. \tag{19}$$

Proof: Directly follows from the proof of the theorem 4 in [18]

5.4 Reconf-Algorithm

To adjust the framework to cope with any unpredictable external event such as new task arrivals, task removal, and increase-decrease power supply, we characterize a reconfiguration as any procedure that permits to reconfigure the system to be feasible, i.e., satisfying its real-time and energy constraints with the consideration of system performance optimization. We propose an approach with two successive adaptation strategies to reconfigure the system at run-time. The two adaptation strategies are performed in a hierarchical order as depicted in Fig. 1.

- Degradation mode level 1: EDH-MK Algorithm
- Degradation mode level 2: Removal Algorithm

6 Functionalities

In this section we explain all functionalities of ARTE in details while showing their specifications and various characteristics regarding the problem of real-time scheduling in reconfigurable energy harvesting systems.

6.1 Task Set Generator

Task Model The current version proposes a task model according to the Liu and Layland task model with energy related parameters. All tasks are considered periodic and independent. Each task τ_i is characterized by i) worst case execution time WCET C_i, ii) worst case energy consumption WCEC E_i, and iii) its period T_i. It is considered that tasks have implicit deadlines, i.e., deadlines are equal to periods. In addition, each task is characterized by a degree of importance L_i which define the functional and operational importance of the execution of the task vis-a-vis the application. Moreover, tasks are considered to be (m,k)-firm constrained deadlines. Tasks are classified into two categories: the first is firm task set with (1,1)-firm deadline constraints; the other is a set of soft tasks with (m,k)-soft deadline constraints. And

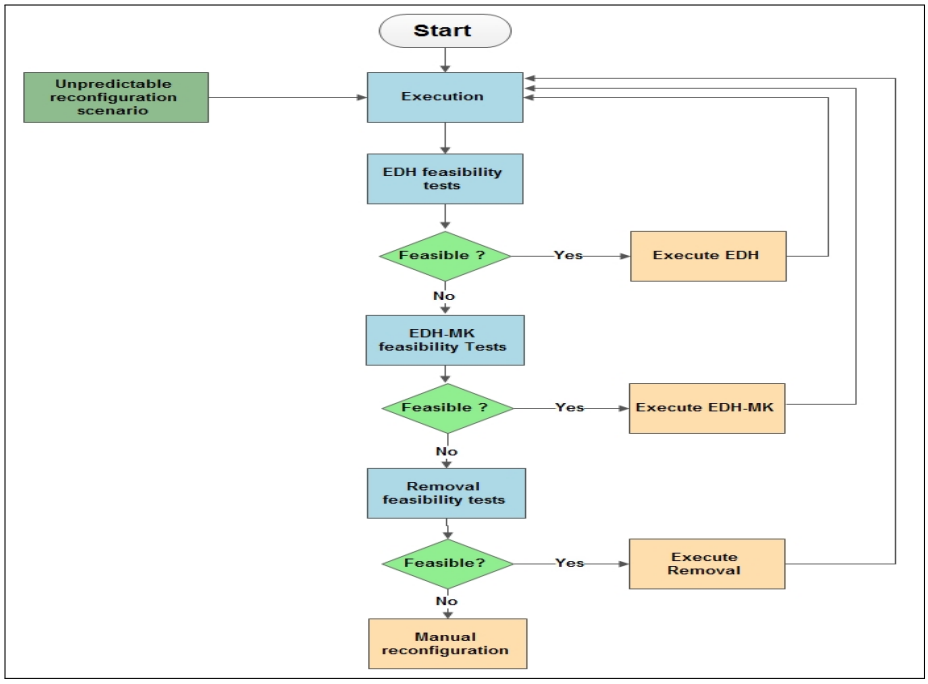

Fig. 1. The reconfiguration scenarios generator tool.

a boolean A_i to determine if a task is abandonnable or not.
The used task sets can be loaded into the simulator either through the GUI by using a file browser or entering the parameters manually, or by using task set generator as depicted in Fig.2 .

For the simulation results to be credible, the used task sets should be randomly generated and varied sufficiently. The current version includes by default a generator based on the UUniFast-Discard algorithm [25] coupled with a hyper-period limitation technique [26] adapted to energy constraints. This algorithm generates task sets by dividing among them the CPU utilization (U = $\sum \frac{C_i}{T_i}$) and the energy utilization ($U_e = \sum \frac{E_i}{T_i Pr}$ where Pr is the recharging function) chosen by the user. The idea behind the algorithm is to distribute the system's utilization on the tasks of the system. When we add the energy cost of tasks to the system, we end up with two parameters to vary and two conditions to satisfy. The algorithm in its current version distributes U and U_e uniformly on the tasks then it finds the 2-tuple (C_i, E_i) which satisfies all the conditions namely U_i, U_e and energy consumption constraints. The operation is repeated several times until the desired 2-tuple approaches the imposed conditions. Finally, the algorithm returns as a result a time and potentially energy feasible system. The (m,k)-firm parameters are randomly generated in the interval [1, 10]. We define three levels of importance then the degree of importance is randomly generated in the interval [1, 3] where 1 is the higher importance level. The parameter A_i is randomly generated in the interval [0, 1].

6.2 Reconfiguration Scenarios Generator

In order to represent as near as possible the real behavior of the physical reconfigurable energy harvesting real-time embedded systems we developed a reconfiguration

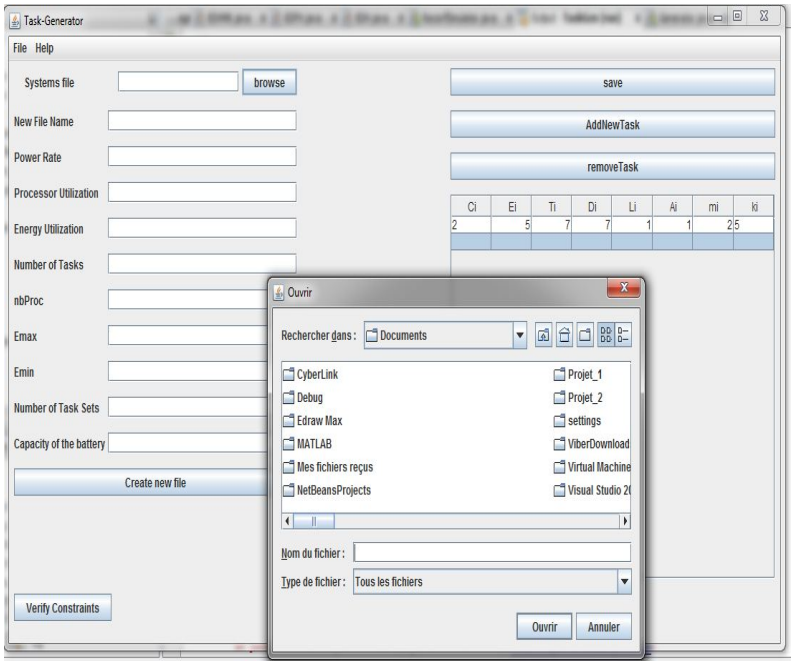

Fig. 2. The task set generator tool.

scenarios generator tool. Through the GUI the user can use personalized reconfiguration scenarios by selecting the user personalization option, or using random reconfiguration scenarios by using the random task set generator as depicted in Fig. 3.
The option user personalization offers to the user the possibility to generate reconfiguration scenarios that modify the applicative functions i.e., add-remove software tasks or increase-decrease the power supply of the system. For the simulation results to be credible, the used reconfiguration scenarios should be randomly generated and varied sufficiently. The current version includes by default a reconfiguration scenarios generator which offers the possibility to generate three kinds of reconfiguration scenarios: i) high dynamic system, ii) medium dynamic system, and iii) low dynamic system. The randomly reconfiguration scenarios algorithm calculates the number of jobs Njobs in the system upon one hyper-period.

- High dynamic system: the generator will add randomly n tasks where n is randomly in the interval [10%, 3%] from Njobs.
- Medium dynamic system: the generator will add randomly where n is randomly in the interval [5%, 1%] from Njobs.
- Low dynamic system: the generator will add randomly where n is randomly in the interval [1%, 0%] from Njobs.

6.3 Simulation Tool

The aim of this tool is to simulate the scheduling of a system according to the parameters and the assumptions of the user, mainly the task set, and the scheduling policy. But the purpose of ARTE is not only restricted to checking the feasibility of a given system but also to simulate and analyze the performance of the scheduling policies when unpredictable reconfiguration scenarios occur in the system. Through

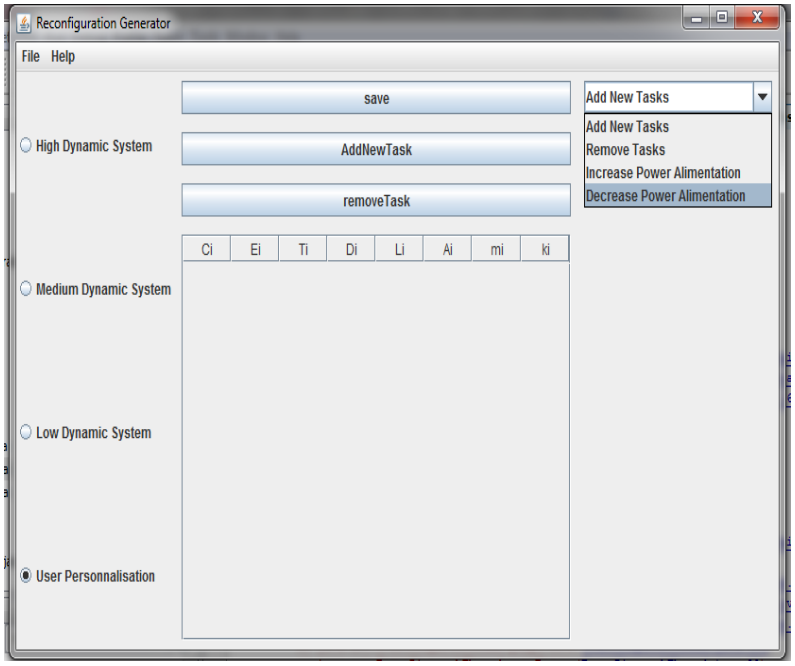

Fig. 3. The reconfiguration scenarios generator tool.

the main interface as depicted in Fig.4 the user have the possibility to use task sets loaded into the simulator either through the GUI by using a file browser or entering the parameters manually, or by using task set generator. When the user creates a system the hyper period will be calculated automatically and displayed in the GUI. For simulation the user choose the scheduling policy as well as the time interval for simulations. Two kinds of analyzes can be performed: scheduling simulation and test feasibility. The user can generate a set of reconfiguration scenarios to the system.

7 Case Study

This section presents a case study through which we can show the different features and functionalities implemented in ARTE as well as to explore the performance of the proposed EDH-MK algorithm and Reconf-Algorithm in order to keep feasible executions with graceful QoS after any external reconfiguration scenario that may evolve the system towards an unfeasible state. For this aim we create a new system where we generate randomly a task set with parameters in table 1. Initially, we have verified the task set feasibility using the simulator tool Fig.5. Then, we have choose to generate a random high dynamic reconfiguration scenario using the reconfiguration scenario tool Fig. 6. Thereafter, the system evolves toward an unfeasible state Fig. 7. In order to analyze the EDH scheduler performance we have run a simulation on 100 time units. The EDH scheduler provides 114 deadline miss Fig. 8. In order to analyze the EDHMK scheduler performance we have run a simulation on 100 time units. The EDHMK scheduler provides 16 deadline miss Fig. 9.

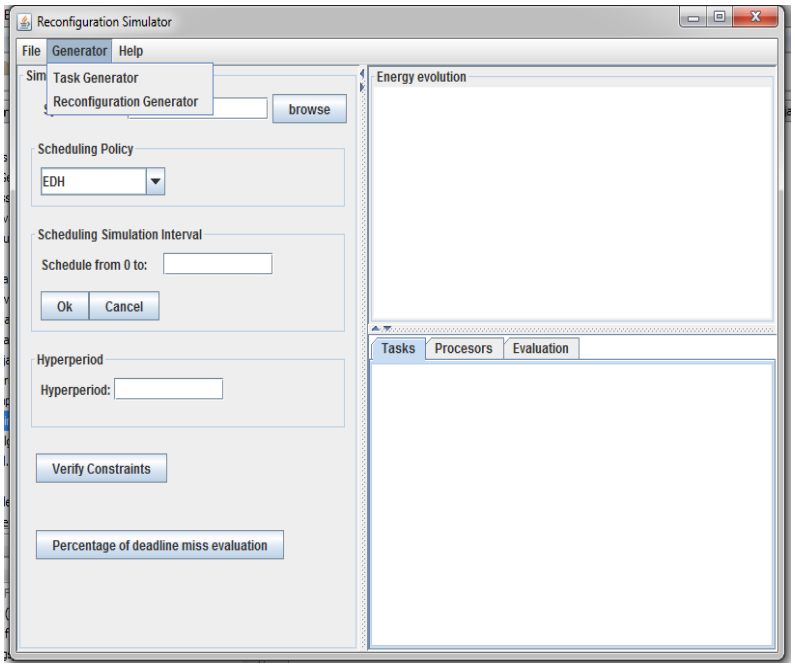

Fig. 4. The simulation tool.

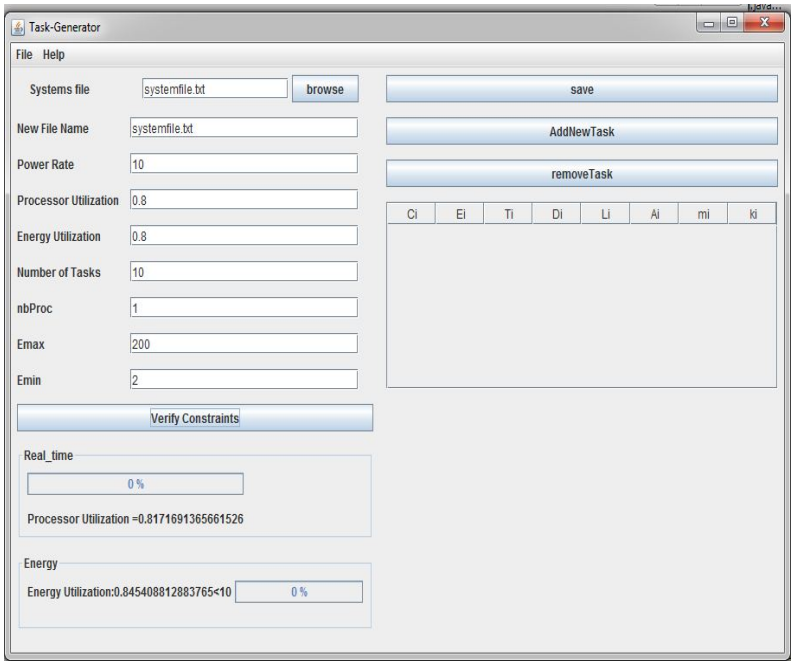

Fig. 5. Random system generation.

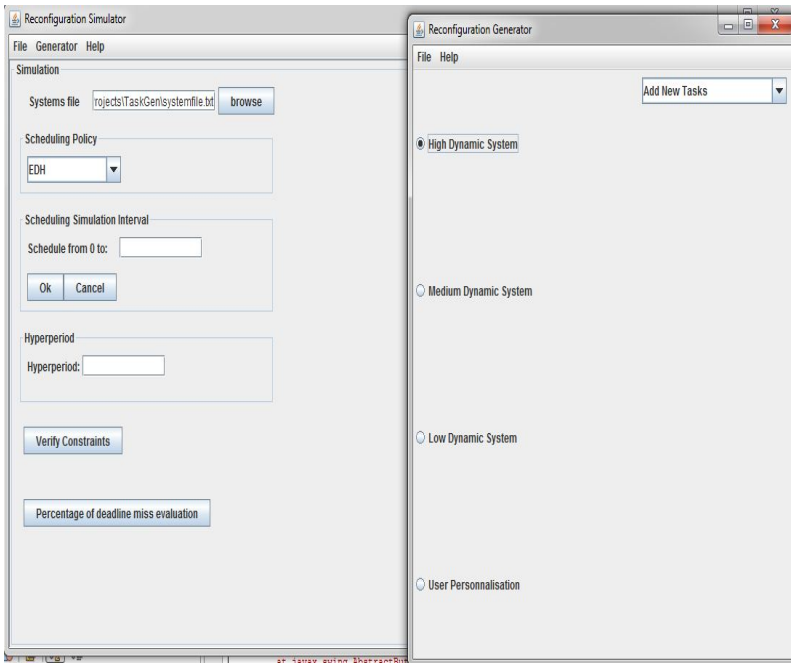

Fig. 6. Random reconfiguration scenario generation.

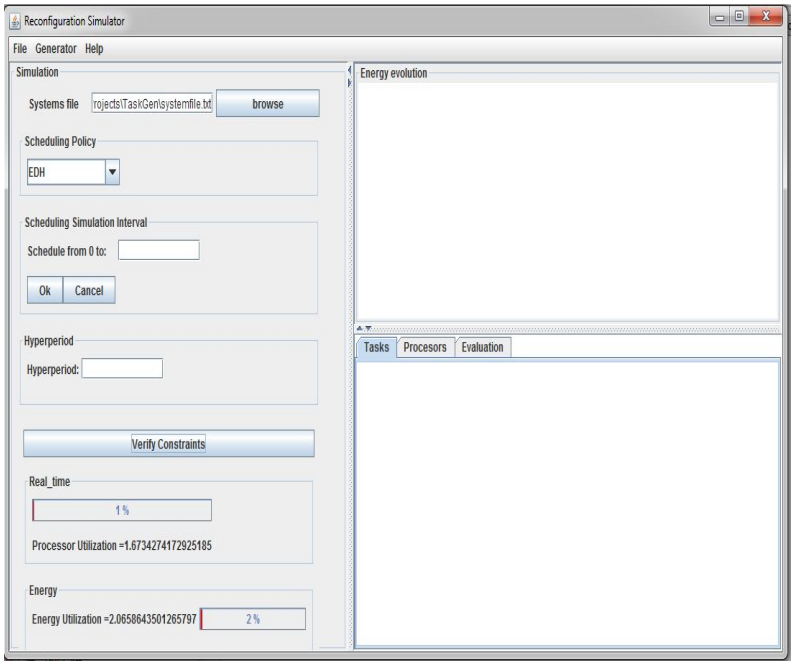

Fig. 7. Test system feasibility.

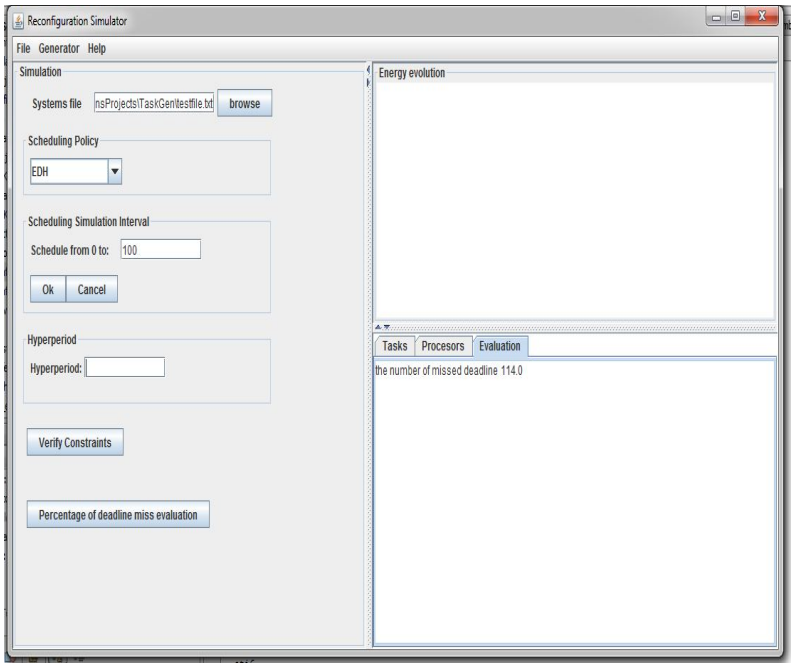

Fig. 8. EDH simulation.

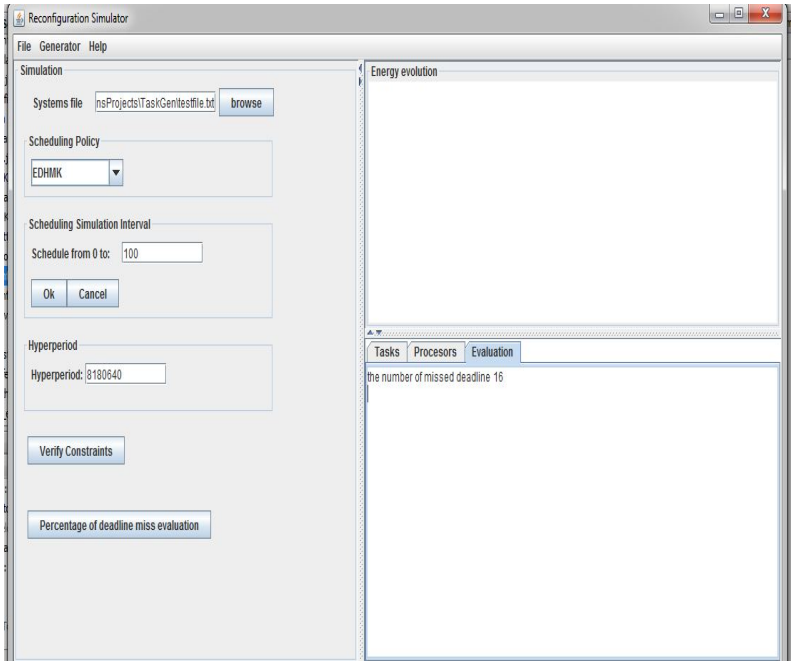

Fig. 9. EDH-MK simulation.

Table 1. Initial System Configuration.

File Name	systemfile.txt
Power Rate	10
Processor Utilization	0.8
Energy Utilization	0.8
Number of tasks	10
Emax	200
Emin	2
Battery Capacity	200
Proc number	1

8 Future Works

The actual release offers many important features where the main purpose is to provide a simulator tool which provide the flexibility and the performance to deal with unpredictable reconfiguration scenarios. But, it has been made in a hurry and improvements are planned to address all features targeted by the proposed simulator tool.
The authors are now working on:

- the development of an extension of the developed graphical user interface to facilitate the use of the simulator by a large number of users, and to provide our tool with a GUI to display the simulation results in an interactive and intuitive way. Three different views envisaged: a time chart, a processor view and an energy curve as well as a comparison results view permits to the user to see the simulation of selected scheduling policies,
- implement other task models in the simulator,
- implement other scheduling approaches in the simulator, such as the fixed priority approaches.
- implement the multiprocessor platforms, and develop new scheduling techniques based on the migration of tasks between the different processors,
- finally, we plan the use of a distributed system decentralizing the control, and more precisely the use of an multi-agent system MAS. We aim through the use of MAS to represent as near as possible the real behavior of the physical networked reconfigurable energy harvesting real-time embedded systems thanks to the developed simulator. Motivated by these considerations, we choose to deploy the intelligent agents to simulate the dynamic behavior of networked reconfigurable energy harvesting real-time embedded systems.

9 Conclusion

This report presents ARTE, a real-time scheduling simulator for reconfigurable energy harvesting real-time embedded systems. Presently, the ARTE simulator is able to simulate accurately the execution of task sets of a reconfigurable energy harvesting system. We briefly presented existing simulation tools. However, none of the aforementioned efforts respond to the new criteria of flexibility, agility and high-performance. Thus, there is a need to develop a new simulation tool that offers the ability to deal with the dynamic behavior of reconfigurable systems. We have detailed the different features provided by ARTE: 1) scheduling simulation, 2) feasibility tests, and 3) percentage of missed deadline measurement. We have described the three main tools

of ARTE: 1) random generator of task sets, 2) random generator of reconfiguration scenarios sets, and 3) a simulator tool. Finally, we presented some expanding features we will implement.

10 Proposed Thesis Outline

Abstract

Introduction

I State of The Art

1 Real-Time Scheduling

2 Energy Harvesting Systems

3 Real-Time Scheduling for Energy Harvesting Systems

4 Reconfigurable Systems

5 Real-Time Scheduling for Reconfigurable Systems

II Contributions

1 Real-Time Scheduling under Energy constraints for Reconfigurable Systems: Monoprocessor Case

2 Real-Time Scheduling under Energy constraints for Reconfigurable Systems: Multiprocessor Case

3 Case Study: Biomedical Application

4 Simulation Tool: ARTE

General Conclusion

References

1. A. Gharbi, M. Khalgui, and M. A. Khan, "Functional and operational solutions for safety reconfigurable embedded control systems," in *Embedded and Real Time System Development: A Software Engineering Perspective*, pp. 251–282, Springer, 2014.
2. R. M. da Silva, I. F. Benítez-Pina, M. F. Blos, D. J. Santos Filho, and P. E. Miyagi, "Modeling of reconfigurable distributed manufacturing control systems," *Technological Innovation for Cloud-Based Engineering Systems*, vol. 48, no. 3, pp. 1284–1289, 2015.
3. Y. Chen, X. Mao, F. Hou, Q. Wang, and S. Yang, "Combining re-allocating and re-scheduling for dynamic multi-robot task allocation," in *Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on*, pp. 000395–000400, IEEE, 2016.
4. H. Grichi, O. Mosbahi, M. Khalgui, and Z. Li, "Rwin: New methodology for the development of reconfigurable wsn," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 109–125, 2017.

5. H. Grichi, O. Mosbahi, and M. Khalgui, "Rocl: New extensions to ocl for useful verification of flexible software systems," in *Software Technologies (ICSOFT), 2015 10th International Joint Conference on*, vol. 1, pp. 1–8, IEEE, 2015.
6. M. G. Harbour, J. G. García, J. P. Gutiérrez, and J. D. Moyano, "Mast: Modeling and analysis suite for real time applications," in *Real-Time Systems, 13th Euromicro Conference on, 2001.*, pp. 125–134, IEEE, 2001.
7. F. Singhoff, J. Legrand, L. Nana, and L. Marcé, "Cheddar: a flexible real time scheduling framework," in *ACM SIGAda Ada Letters*, vol. 24, pp. 1–8, ACM, 2004.
8. R. Urunuela, A.-M. Déplanche, and Y. Trinquet, "Storm a simulation tool for real-time multiprocessor scheduling evaluation," in *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on*, pp. 1–8, IEEE, 2010.
9. P. Courbin and L. George, "Fortas: Framework for real-time analysis and simulation," *Proc. of WATERS*, pp. 21–26, 2011.
10. Y. Chandarli, M. Qamhieh, F. Fauberteau, and D. Masson, *YARTISS: A Generic, Modular and Energy-Aware Scheduling Simulator for Real-Time Multiprocessor Systems*. PhD thesis, UPE LIGM ESIEE, 2014.
11. S. Kato, R. Rajkumar, and Y. Ishikawa, "A loadable real-time scheduler suite for multicore platforms," *Technical report, Department of Electrical and Comptuter Engineering*, 2009.
12. M. Holenderski, M. Van Den Heuvel, R. J. Bril, and J. J. Lukkien, "Grasp: Tracing, visualizing and measuring the behavior of real-time systems," in *International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, pp. 37–42, 2010.
13. J. M. Calandrino, H. Leontyev, A. Block, U. C. Devi, and J. H. Anderson, "Litmusˆ rt: A testbed for empirically comparing real-time multiprocessor schedulers," in *Real-Time Systems Symposium, 2006. RTSS'06. 27th IEEE International*, pp. 111–126, IEEE, 2006.
14. J. W. McCormick, F. Singhoff, and J. Hugues, *Building parallel, embedded, and real-time applications with Ada*. Cambridge University Press, 2011.
15. A. Gammoudi, A. Benzina, M. Khalgui, D. Chillet, and A. Goubaa, "Reconf-pack: A simulator for reconfigurable battery-powered real-time systems," in *30th European Simulation and Modelling Conference*, 2016.
16. M. L. Dertouzos, "Control robotics: The procedural control of physical processes," in *Proceedings IF IP Congress, 1974*, 1974.
17. M. Chetto and A. Queudet, "A note on edf schedulingfor real-time energy harvesting systems," *IEEE Transactions on Computers*, vol. 63, no. 4, pp. 1037–1040, 2014.
18. M. Chetto, "Optimal scheduling for real-time jobs in energy harvesting computing systems," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 2, pp. 122–133, 2014.
19. G. Bernat and A. Burns, "Combining (/sub m//sup n/)-hard deadlines and dual priority scheduling," in *Real-Time Systems Symposium, 1997. Proceedings., The 18th IEEE*, pp. 46–57, IEEE, 1997.
20. M. Hamdaoui and P. Ramanathan, "A dynamic priority assignment technique for streams with (m, k)-firm deadlines," *IEEE transactions on Computers*, vol. 44, no. 12, pp. 1443–1451, 1995.
21. S. Hua and G. Qu, "Energy-efficient dual-voltage soft real-time system with (m, k)-firm deadline guarantee," in *Proceedings of the 2004 international conference on Compilers, architecture, and synthesis for embedded systems*, pp. 116–123, ACM, 2004.
22. G. Quan and X. Hu, "Enhanced fixed-priority scheduling with (m, k)-firm guarantee," in *Real-Time Systems Symposium, 2000. Proceedings. The 21st IEEE*, pp. 79–88, IEEE, 2000.
23. P. Ramanathan, "Overload management in real-time control applications using (m, k)-firm guarantee," *IEEE Transactions on Parallel and Distributed Systems*, vol. 10, no. 6, pp. 549–559, 1999.
24. L. Niu and G. Quan, "Energy minimization for real-time systems with (m, k)-guarantee," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 7, pp. 717–729, 2006.

25. E. Bini and G. C. Buttazzo, "Measuring the performance of schedulability tests," *Real-Time Systems*, vol. 30, no. 1, pp. 129–154, 2005.
26. J. Goossens and C. Macq, "Limitation of the hyper-period in real-time periodic task set generation," in *In Proceedings of the RTS Embedded System (RTS01*, Citeseer, 2001.