A Simulation Tool for Real-Time Systems Using Environmental Energy Harvesting

M. Chetto, *Member IACSIT*, H. Zhang

***Abstract*—Energy constrained systems such as sensor networks can increase their usable lifetimes by extracting energy from their environment. This is known as energy harvesting. Task scheduling at the single nodes should account for the properties of the regenerative energy source which fluctuates, capacity of the energy storage as well as deadlines of the time critical tasks. Designing efficient scheduling strategies is significantly more complex compared to conventional real-time scheduling. In this paper, we present a simulator that enables to construct an optimal schedule using the so-called LSA algorithm, for any task set, battery capacity and energy source profile.**

***Index Terms*—Energy harvesting, real-time, scheduling, simulation tool, monoprocessor.**

I. Introduction

In the design process of a Real-Time Embedded System there are several important attributes the developer has to take care about: first of all, the final product should do the right thing i.e. it must be functionally correct. Secondly, the performance should be adequate, expressed in terms of response times. Thirdly, when relying on a battery as power source, the system must behave in an energy aware manner. Since the alteration of one of the attributes surely also affects the other two, an integrated framework where all aspects can be evaluated and balanced is really desirable.

In this paper, we present a simulation tool for real-time systems using environmental energy harvesting. Energy harvesting otherwise known as energy scavenging is the conversion of ambient energy into electricity to power small electric and electronic devices, making them self-sufficient, often for decades.

A key consideration that affects power management in an energy harvesting system is that instead of minimizing the energy consumption and maximizing the lifetime achieved as in classical energy storage operated devices, the system operates in a so-called energy neutral mode by consuming only as much energy as harvested. The simulator enables to construct an optimal schedule for any task set, battery capacity and energy source profile. The simulator includes two real-time scheduling algorithms: LSA (Lazy Scheduling Algorithm) [1], [2] and EDF (Earliest Deadline First) [3]. According to results of a simulation, engineers can analyze and judge the performance of a real-time system. And they can dimension the energy storage to optimally play out the variations of the underlying energy source.

Manuscript received October 10, 2012; reviesed November 15, 2012.

M. Chetto and H. Zhang are with IRCCyN Institute (e-mail: maryline.chetto@ univ-nantes.fr), LUNAM University, Nantes, France.

The rest of the paper is organized as follows: In section 2, we will present some issues in energy harvesting systems. Section 3 describes real-time scheduling strategies under energy constraints. An optimal mono-processor scheduling algorithm, namely LSA will be introduced in section 4. We will describe our software simulator in section 5. Section 6 includes some experimental results generated by the simulator. Section 7 concludes the paper.

II. Issues in Energy Harvesting Systems

In the aim of bringing solutions to the energy problem and extending the system operating duration, energy harvesting technology has been explored very recently [4], [5]. Possible energy harvesting sources include solar, thermal, vibrational and kinetic energy, etc. Among these ones, solar energy offers the highest energy density, lowest production cost, and highest availability. An energy harvesting system draws parts or all of its operating energy from its ambient energy sources. Consequently, it has potential to overcome the energy constraint imposed by traditional battery-powered embedded systems and may operate perennially. Unfortunately, most wireless sensing systems built up until now do not make the most of power. It results that they use a much larger solar panel than necessary to guarantee permanent operation or they rely on a larger, more expensive, higher capacity battery than needed.

Furthermore, the main problem to solve will come from the instantaneous power level that tends to vary over a wide. The autonomous nature of operation makes it imperative that the system adapts its power consumption accordingly. Goal of this adaptation is to maximize the utility of the application in a long-term perspective. Then, the crucial issue is to find scheduling mechanisms that can adapt the performance to the available energy profile. Up to now, when designing a real-time embedded system, the first concern has been usually time. Now, the primary concern is that power from solar panels or other free sources that cannot be stored (or stored with limited capacity) should be fully consumed greedily, or else this energy will be wasted. With solar energy, during the day, the real-time tasks are executed while the battery is recharging. But during the night, the system must rely entirely on the energy that has been collected, and stocked during the day in the battery with a capacity that must be sufficient.

Another concern will be to control over time the activity of the processor. In a real-time environment where tasks have to meet deadlines and execute periodically, energy harvesting and task scheduling are consequently strongly dependent since they have to handle timing constraints and variability of

DOI: 10.7763/IJCEE.2013.V5.681

available energy.

III. Real-Time Scheduling Under Energy Constraints

Many wireless sensor network applications have real-time requirements where sensed data must be delivered to a base station within a deadline and before the data becomes old. For example, a system that monitors temperature in a nuclear power plant would require that the readings be reported to a base station within enough time for a proper response to be made to a rapid increase in the temperature.

Such real-time applications require periodic activities that have to be cyclically executed at fixed rates and within specific deadlines. Typically, each periodic instance is assigned a relative deadline equal to the task period and is treated as a hard job. Thus, a periodic task is executed only if all its instances are guaranteed to complete within their deadlines. Schedulability analysis of periodic task sets can easily be performed both under fixed and dynamic priority assignments.

In particular, a lot of work has been done for the Rate Monotonic (RM) and the Earliest Deadline First (EDF) algorithms [3]. Schedulability analysis has also been extended for the case in which tasks use shared resources or run in the presence of aperiodic activities, under fixed priority scheduling and in dynamic priority systems as well [6], [7].

Despite the significant body of results in real-time scheduling, many real world problems are not easily supported, including energy harvesting. While Earliest Deadline First (dynamic priority depending on urgency) and Rate Monotonic (fixed priority depending on period) can support sophisticated task set characteristics such as deadlines, precedence constraints, shared resources, jitter, etc., they are all open loop scheduling algorithms. Open loop refers to the fact that once schedules are created they are not "adjusted" based on continuous feedback.

Systems with open-loop schedulers are usually designed based on worst-case parameters. Such an approach can result in a highly underutilized system based on extremely pessimistic estimation of workload (or energy). While open-loop scheduling algorithms can perform well when the workload and the processing performance are accurately modeled, they perform poorly in unpredictable dynamic systems including regenerative energy dependent ones.

Only in the past decade, researchers started to address power and scheduling issues with the objective of either minimizing power usage under timing constraints or maximizing the system performance under the energy constraints but do not consider the rechargeability of the batteries. For example, EDF and RM scheduling have been extended to variable-voltage processors. The idea is to save power by slowing down the processor just enough to meet the deadlines [8]. But solely applying these techniques has limitations in energy harvesting systems because they minimize CPU power, rather than they dynamically manage power according to the profiles of both available energy and processor workload.

IV. An Optimal Mono-Processor Scheduling Algorithm

In [1], C. Moser et al. propose a real-time scheduling algorithm, called Lazy Scheduling Algorithm (LSA), and based on the work of A. Kansal, in 2006 [4], that uses task postponement. Algorithm LSA is energy-clairvoyant, i.e., the generated energy in the future is known. Taking into account available time as well as processable energy, an optimal task ordering can be determined based on the prediction of the available energy in the future.

Their work deals with a real-time system built around a mono-processor architecture that draws the energy from storage and uses it to process tasks (periodic or non periodic ones) with arrival time, deadline, and worst case execution time. The worst case execution time corresponds to the maximum energy demand of the task. The arrival time of the task is not known beforehand. The deadline as well as the worst case execution time of the task is unknown before it is released. However, as long as the task is released, all these parameters are determined. They assume that tasks are preemptable and executed according to the Earliest Deadline First policy. At any time, the energy source module harvests the energy from its ambient environment and then converts it into electrical energy. The electrical energy can be stored in the energy storage (e.g. battery), whose capacity is precisely known. The stored energy is assumed to be known at the system level at any time instant and is no more than the storage capacity. It is assumed that the energy storage is ideal and the battery can be recharged up to its capacity. Likewise, it can be completely discharged to zero. If the stored energy reaches the capacity, the incoming harvested energy overflows the storage and is discarded.

According to LSA, the processor executes all tasks at full power; and the system starts executing a task if the task is ready and has the earliest deadline among all ready tasks and the system is able to keep on running at the maximum power until the deadline of the task.

In contrast to greedy scheduling algorithms such as EDF, LSA hesitates to power tasks until it is necessary to respect timing constraints. In that sense, tasks are executed neither as soon as possible nor as late as possible, so the purpose of LSA is to find an optimal start time s_i to begin execution of a task τ_i. In [1], [2], the authors of LSA also discuss an admittance test that decides, whether a set of real-time tasks can be scheduled without violating deadlines. Another crucial question which has been solved is how to dimension the capacity of the battery that ensures continuous operation. The performance evaluation study demonstrates that achievable capacity savings between 20% and 45% are obtained compared with the classical EDF algorithm.

While optimal in the case of a single speed processor, the main drawback of the LSA algorithm is that a task is executed at the maximum power and may be finished well before its deadline. In this case, the task slack would be wasted; more importantly, the limited energy is unnecessarily squandered. As a consequence, future tasks have to violate their deadlines because of energy shortage.

V. Description of the Simulator

A. Objective of the Simulator

The simulator has been designed specifically for any periodic task set under energy harvesting constraints. By using it, we can report details of the schedule produced for any task set with given energy storage capacity and energy source profile. According to the system output results, the user can easily evaluate the performances of a real-time embedded system and verify the temporally-feasibility of a task set. More precisely, we can analyze the Quality of Service given by the percentage of satisfied deadlines under various battery capacities and various energy source profiles using two schedulers, namely EDF and LSA schedulers.

Moreover, the execution sequence of tasks is visually displayed on the screen. If we want to change the simulation conditions or some values, it can be easily achieved by the friendly interface, see Fig. 2 and Fig. 3. Such a simulator reduces the number of physical prototypes that need to be built and tested. So, it is playing an essential role in the verification phase.

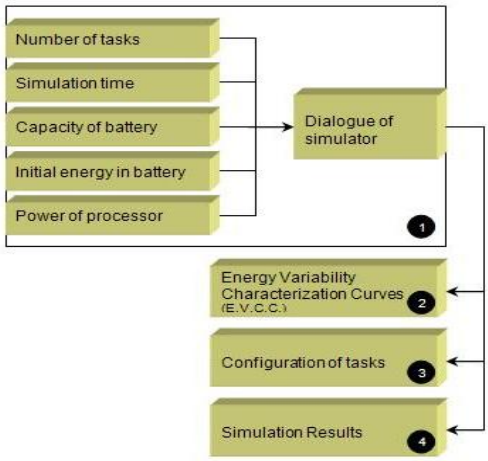

Fig. 1. Framework of the simulator

B. Components of the Simulator

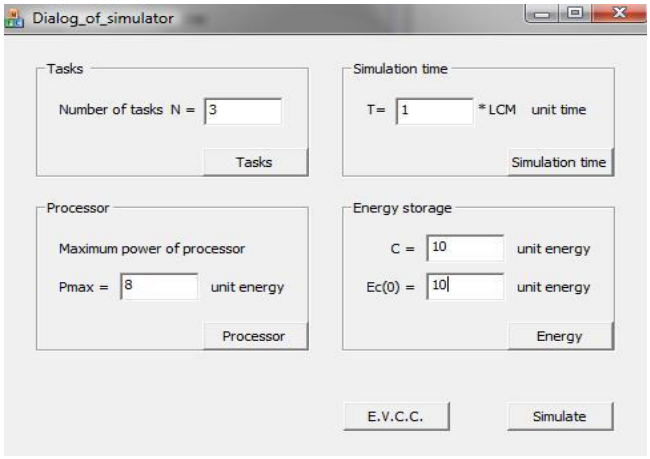

Fig. 2. Dialogue box of the simulator

This software simulation tool consists of four main components: a module for dialogue with the user, a module for energy variability characterization curves called E.V.C.C (Energy Variability Characterization Curves) in [2], a module for tasks configuration and finally a module for memorizing the results. The simulator structure is presented in Fig. 1.

Let us describe in detail, the function associated to each module.

1) Dialogue Module

Through this interface, engineers can set the simulation with various input data: number of tasks, simulation period, power of processor, etc. If some input data is missing, default values for the missing data are 0. Fig. 2 shows a view of the user interface.

2) Energy Characterization Module

In this phase, we launch a process to generate various E.V.C.C. curves. Each curve represents a typical regenerative energy source. As seen in fig.3, we can choose any one from five different curves through the buttons.

There are constant energy sources, periodic energy sources (to model piezoelectric sources, etc), energy sources with average distribution (e.g. electromagnetic sources, mechanical vibration sources, etc.), and some energy sources with normal distribution (e.g. photovoltaic sources).

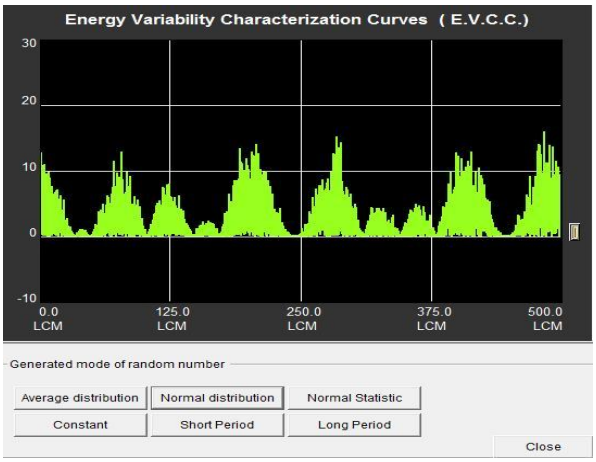

Fig. 3. Generation of E.V.C.C. curves

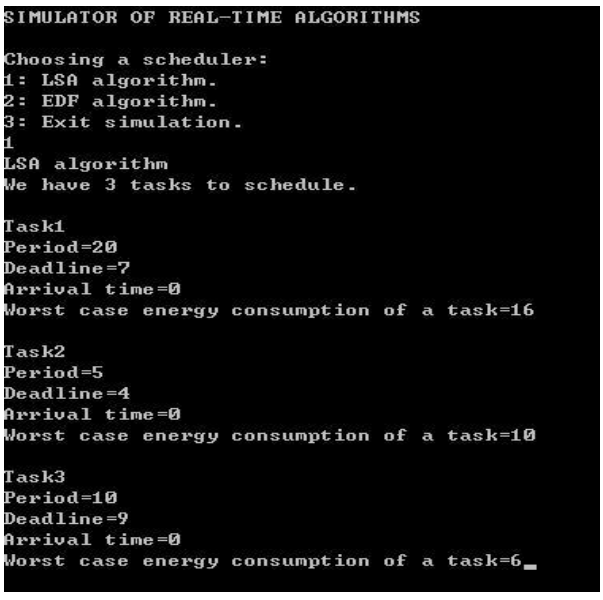

Fig. 4. Configuration module

3) Configuration Module

This component permits the user to achieve two objectives: describing a task set and choosing a real-time scheduler. All operations relative to configuration are completed through DOS prompt. In DOS window, we input all data through the keyboard and all data are displayed on the screen. Fig. 4 shows the user interface.

4) Result Module

After all necessary settings, the engineer can launch the calculation engine in order to simulate the scheduling process of a real-time system. Some results related to system performance are calculated, and not only displayed by graph, but also registered in an Excel file. The following parameters are calculated or monitored in the simulation process:

- Least Common Multiple of the periods for a task set (LCM);
- processor utilization factor (U);
- start time of task τ_i (s_i);
- level of energy in the storage at time t (Ec(t));
- percentage of passed deadlines;
- execution sequence of tasks.

C. Instruction of the Simulator

The simulator is very simple to use. The users should first open the dialogue box (as shown in fig. 2), and input the values following the commentaries. Then, after pressing the "E.V.C.C." button, an interface is displayed on the screen (as shown in fig. 3). The user can arbitrarily choose an energy source for simulation. And if we press the "Simulate" button of the dialogue box, a window appears, as shown in fig. 4. The user then configures a task set following the DOS prompt. The user hits the "Enter" key on the keyboard and the results appear on the screen, and are saved in an Excel file.

VI. Illustration on a Practical Example

A. Necessary Configuration for the Simulator

In the following example, we study a real-time system which is composed of periodic tasks, denoted as follows: $\tau = \{\tau_1, \tau_2, \ldots \tau_n\}$. A four-tuple ($T_i$, D_i, A_i, E_i) is associated with each τ_i, where T_i is the period, D_i is the deadline, A_i, is the arrival time and E_i represents the energy consumption in the worst case situation.

Consider the three following tasks: τ_1 (20, 5, 0, 16), τ_2 (5, 4, 0, 10) and τ_3 (10, 9, 0, 6). We assume that the capacity of the energy storage is E = 10. The storage is assumed to be full initially, i.e. Ec(0) = E. The recharging power P_s is constant and equal to 4. The processing power, called P_{max}, is equal to 8. And we assume that the duration of simulation, T, is given by LCM (T_1, T_2, T_3) here equal to 20.

B. Task Processing by LSA Scheduler

Let us assume that we select the LSA scheduler. Fig. 5 shows some simulation outputs, the details of the execution sequence and the energy level in the battery during the whole simulation time. Displaying such results on the screen is very beneficial to the feasibility analysis.

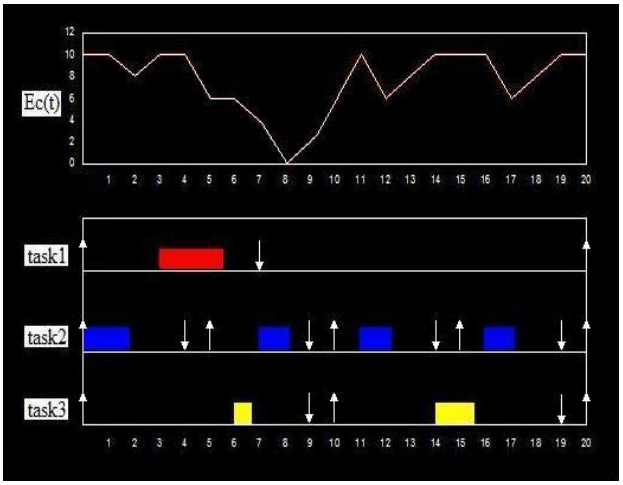

Fig. 5. Output results for the LSA scheduler

Based on the resulting statistical data, we can get the percentage of deadlines which are satisfied along the simulation time interval. It is shown in the last line of the screen through DOS prompt, (see fig. 6). Thanks to this parameter, we can verify the temporal feasibility of the task set τ.

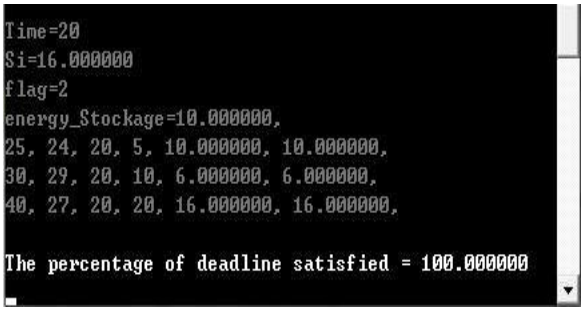

Fig. 6. Ratio of deadline missing

Value 100 means that LSA can feasibly schedule the given task set without any deadline violation, be given the capacity of the battery and the power source profile.

VII. Conclusion

There is a specific need to develop technology and methods for harvesting energy from the environment. Efficient use of energy and waste reduction are the political, social and technical challenge of the next decade.

New applications of energy harvesting technology for embedded systems are beginning to drive economic development in many sectors. It concerns as well the high technology sectors as the general public products in which wireless sensor networks are used in a variety of embedded applications, such as environmental applications, military applications…

In this paper, we have presented a tool in order to simulate the behavior of embedded harvesting systems. It has been designed for checking the schedulability of a task set under temporal constraints expressed in terms of deadlines and energy constraints expressed in terms of storage capacity and source power variation. Harvesting systems constructed to

date extract power efficiently from the source but do not use it adequately under real-time running conditions. As a result, they need a much larger harvester (e.g. solar panel) than necessary to yield the same level of power as a more efficient one, or they rely on a larger, more expensive, higher capacity battery than needed in order to sustain extended operation.

Our simulator integrates two different schedulers: a classical one which greedily consumes energy, EDF, and an optimal one, LSA which has been specifically designed to optimize dimensioning and scheduling.

Here, all tasks are assumed to be periodic with deadlines less than or equal to periods, and the execution time of tasks is proportional to its energy requirement. Through this simulator, the user easily determines the feasibility of a task set. In the whole simulation, we mainly focus on the following two metrics: the energy variation in the energy storage unit along time and the percentage of deadlines which are missed.

REFERENCES

[1] C. Moser, D. Brunelli, L. Thiele, and L. Benini, "Real-time scheduling with regenerative energy," in Proc. *of the 18th Euromicro Conference on Real-time Systems (ECRTS06)*, pp. 261-270, Dresden, Germany, October, 2006.

[2] C. Moser, D. Brunelli, L. Thiele, and L. Benini, "Real-time scheduling for energy harvesting sensor nodes," *Journal of Real Time Systems*, vol. 37, no. 3, pp. 233-260, 2007

[3] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in Hard Real-time Environment," *Journal of ACM* vol. 20, no. 1, pp. 46-61, 1973.

[4] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, *Power management in energy harvesting sensor networks*, NESL Technical Report Number: TR-UCLA-NESL-200605-01, 2006.

[5] S. Roundy, D. Steingart, L. Frechette, P. K. Wright, and J. M. Rabaey, "Power sources for wireless sensor networks," *First Europeean Workshop, EWSN 2004,* Lecture Notes in Computer Science, pp. 1-17, Berlin, January 2004.

[6] H. Chetto and M. Chetto, "Some Results of the Earliest Deadline Scheduling Algorithm," *IEEE Transactions on Software Engineering*, vol. 15, no. 10, pp. 1261-1269, 1989.

[7] J. W. S. Liu, Real-Time Systems, Prentice Hall, pp. 610, March 2000.

[8] G. Quan and X. S. Hu, *Designing Embedded Processors, A Low Power Perspective*, Springer Netherlands, Chapter 10, 2007

Maryline Chetto was born in 1959 at Illiers, France. She received her M. Sc. Degree in automatic control, the Ph. D. Degree in computer science, and the HDR (Habilitation à Diriger des Recherches) from the University of Nantes, France, respectively in 1982, 1984, and 1993.

She is currently professor at the University of Nantes in the Institute of Technology, namely IUT de Nantes. Her research is conducted in the Group of Real-Time Systems of the Research Institute of Communications and Cybernetics (IRCCyN).

Prof. Chetto has been the leader of a French national R&D project, namely Cleopatre, supported by the French government, which aims to provide free open source real-time solutions. Prof. Chetto has published more than 70 journal articles and conference papers in the area of real-time operating systems. Her research interests include scheduling in real-time systems. Her current studies concern real-time energy harvesting applications.

Hui Zhang was born in Zibo, China. He received his M.Sc. degree in Robotics and Automatic Control from the University of Montpellier 2, France in 2008.

From October 2008 to June 2012, he has been working toward the Ph.D. degree in Computer Science at the Institut de Recherche en Communications et Cybernétique de Nantes (IRCCyN – UMR CNRS 6597).

He is conducting his research in the area of real-time scheduling with renewable energy constraints. More precisely, he studies the relative performance of non-idling scheduling heuristics compared with the idling optimal one for uni-processor platforms.