Simulation of Efficient Real-time Scheduling and Power Optimization

Rym Cheour[1], Richard Urnuella[2] ,Yvon Trinquet[2] and Mohamed Abid[1]

[1] Computer and Embedded System Lab, National School of Engineers of Sfax
Sfax, Tunisia

[2] IRCCyN, University of Nantes
Nantes, France

Abstract

Sophisticated applications turn out to be executed upon more than one CPU for practical and economic reasons. Due to advances in circuit technology and performance limitation, multi-core technology has become the mainstream in CPU designs. However, the most serious limitation of these devices is the battery lifetime since battery technology is not keeping up with the rest of the power-hungry processors and peripherals used in today's mobile devices. As a solution, many investigations have turned toward the algorithms of power management combined with some scheduling policies. They can make significant energy saving while preserving the temporal constraints of these embedded systems. Reducing energy, especially, affect not only the battery lifetime, but also aim to reduce the heat generated by real-time embedded controller in various products or even to decrease the conditions of cooling and the costs, in the large scale, of giant multiprocessor computers. To assess the behavior and performance of the strategy of scheduling a flexible multiprocessor scheduling simulation and evaluation platform is needed. This paper puts forth the claim that the STORM simulator improves application quality both in terms of execution time and energy consumption for a high performance mobile computing embedded system design.

Keywords: Scheduling, Power management, DVFS, DPM, simulation, EDF.

1. Introduction

In recent years interest in simulation modeling has greatly increased. Indeed, a simulation environment is essential, since the test of the hardware and software is hard and difficult. For instance, new systems, services and protocols present challenges for testing and require large and complex environments [1]. The simulations represent the problem concretely, provide a broader context allowing a deeper understanding of the situation and handle problems which are difficult or impossible to solve analytically. Beyond cost and time savings, the simulator offers the great opportunity to enhance the functionalities and the features of the real-time systems and to achieve the required Quality of Service (QoS) [2].
The simulator allows you to check on the model, the behavior of the system and therefore the deadlines.
As those simulators are legion, a very interesting open-source tool we came across was the STORM simulator [3]. STORM (Simulation TOol for Real time Multiprocessor scheduling) is able to analyze the behavior and to evaluate the performances of the policies of scheduling while taking into account the algorithms of energy management. We aim to develop scheduling algorithms to minimize the energy/power consumption using the energy conservation method called Dynamic Power Management (DPM) or Dynamic Voltage and Frequency Scaling (DVFS) on one or more processors. Our target is to achieve scheduling techniques for a variety of systems configurations and scheduling policies, e.g. Earliest Deadline First (EDF), while taking into account actual processor limitations such as time/energy. The time (energy) required to change the processor speed is very small compared to that required to complete a task [4]. We will integrate voltage, frequency (DVFS) or state selection (DPM) and task scheduling together in order to maximize the energy saved when executing an independent task set on one or more processors. Our goal is to meet deadlines requirements taking into account the demands of energy and processing time of all the tasks. To do so a set of tasks will be generated with desired statistical properties. The characteristics of the tasks are

known in advance. Then, the selected scheduler will take as input the task set and the physical processor description. The preemption is allowed to adjust the energy consumption based on the system performance requirements. The preemption is when a process in execution is interrupted when a higher priority process arrives [5].

Since, using well defined algorithms to schedule tasks and resources in a real-time system yields an understandable scheduling solution we have chosen to implement an EDF scheduler [7]. The higher resource utilization and the greater flexibility in handling aperiodic requests make EDF highly desirable for real-time embedded systems [7] [8]. Our work is more specifically to implement and assess energy management policies in the Linux operating system and test it via the STORM Simulator. We implemented our algorithms like plug-ins for the core Linux 2.2.16. Linux is easily extended through modules and provides a robust multiprocessor environment [2]. Linux is a multitasking, multi-platform and multi-user operating system whose main asset is its portability [9].

The paper is organized as follows. The first section is about the motivation behind this work. Then, we will present an overview of the STORM simulator. In section 3, we will describe the model and the states of the tasks considered in the simulation. Besides, we will present the EDf scheduler and outline some power management techniques. As DVFS and DPM play a key role in saving the energy consumption, they will be presented. Next, we will expose our simulation results. Finally, we are going to conclude.

2. Motivation of our work

Most of the embedded system are powered by batteries and store a limited amount of energy. So minimizing the overall energy consumption meanwhile avoiding the deadline violations is crucial to achieve high performances and to enhance the reliability of the system [4]. Indeed, there is a growing realization that simulation should be more rigorously considered.

Although a large research investment in low-energy circuit design and hardware level energy management and real-time scheduling has led to more energy-efficient architectures, few tools are available to apply them.

As simulation is becoming a popular option for conducting studies, STORM has matured a capability of analysis of the System behaviour and performances as well as for taking into account many features of hardware architecture such as multicore design, multiprocessor architecture with shared memory, distributed architecture with communication network, memory architecture (L1 and L2 caches, banked memory)[10].

The development of STORM came from the works of the PHERMA research project 'Parallel Heterogeneous Energy efficient Real-time Multiprocessor Architecture' [11]. Indeed, this simulator is open allowing the implementation of new algorithms for scheduling in an external way to the simulation kernel. It also provides a set of means to develop metrics for characterization of the scheduler. Therefore, STORM is an excellent tool to evaluate the behavior and performance of new strategy for scheduling [10][12]. The scheduler must coordinate resources to meet the timing constraints of the physical system. This implies that the scheduler must be able to predict the behavior of all tasks within the system [13]. The output will be a set of screen shots of the system behavior. Hence, we will be able to sort out the relevant information from the execution of the tasks and their interaction with the different scheduling policy, the architecture chosen, the available resources…

3. Presentation of STORM

To verify the scheduling results, we have used a simulation tool called STORM (Simulation TOol for Real time Multiprocessor scheduling). The original need to develop STORM came from the works of the IR CCyN research unit [3]. This simulator considers the requirements of tasks, the characteristics and execution conditions of hardware components and the scheduling rules. Depending on the scheduling policy and the resources described in an XML file, it runs every task over a specified time interval [10][12]. The results of the simulation are a set of diagrams as illustrated in figure 1. All these diagrams help to analyze the behavior of the system (tasks, processors, performances ...).

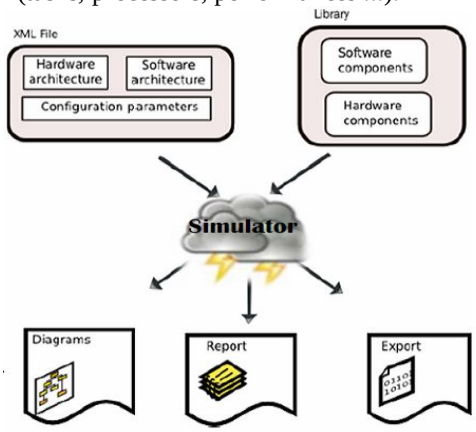

Fig. 1 The STORM simulator [3].

A window displays a Gantt diagram of every task over an interval from 0 to 50 (default values). The title of the window refers to the name given to the task in the XML file (PTASKT1, PTASKT2…). In two other diagrams, we can observe the tasks assigned to processors CPUA and CPUB over the same interval. We can verify the allocation of the tasks on processors according to their availability and to the priorities.

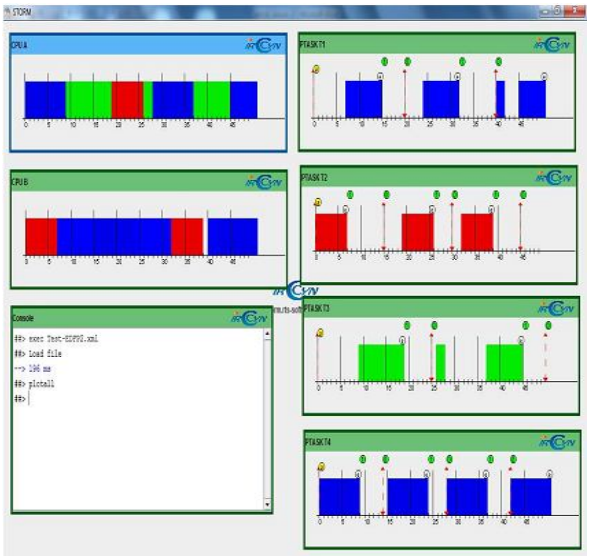

Fig. 2 Graphical results with STORM

Moreover, STORM allows multiprocessor simulation and analyzes energy consumption based on estimations. This simulator also provides support for DPM and DVFS techniques. Being under development, a preliminary study of this tool has been necessary to determine its operation before any action of implementation. To facilitate the development and the checking of the best performance, the development of the scheduler was based on the specification of the EDF scheduler of STORM.

4. Task Management

We consider the preemptive scheduling of a soft real-time system where occasional violation of deadline constraints may not result in a useless execution of the application or calamitous consequences, but decreases utilization [14].The preemption reduces the latency of the system when reacting to real-time or interactive events by allowing low priority processes to be preempted[15]. Preemption helps also to satisfy the constraints especially, the real time constraint. So if the load is very high, the system will slow down its response time [16]. We consider a set of n tasks that will be executed upon m identical CPUs.

4.1 Tasks models

Tasks can be grouped into three families: periodic, aperiodic and sporadic. The periodic tasks execute critical control activities with hard timing constraints aimed at guaranteeing regular activation rates. Aperiodic tasks respond to randomly arriving events. The aperiodic tasks cannot be guaranteed to be served within a hard deadline (the deadlines must be soft) [17]. Aperiodic tasks, typically used to handle the processing requirements of random events such as operator requests. However, the sporadic tasks can arrive at the system at arbitrary points in time, but with defined minimum inter-arrival times between two consecutive invocations [18].
The simplest and the most fundamental model is provided by the periodic task model of Liu and Layland [19]. The periodic tasks are those whose processing is repeated on a regular basis such as the regular monitoring of the state of a physical sensor or sampling of the serial communication line.
T_i a periodic task is characterized by the quadruplet (O_i, T_i, D_i, C_i)[16], where :

- The date of arrival O_i, is the moment of the first activation of the task τ_i
- Time of execution C_i specifies an upper limit on the time of execution of each task τ_i
- The relative deadline D_i denotes the separation between the arrival of the task and the deadline (a task that arrives at time t has a deadline at $t+D_i$);
- A period T_i denoting the duration between two successive activations of the same task.

Each task is independent from the other tasks. This means, the temporal behavior of each task (its ability to meet its deadlines) is not affected by the behavior of any other task in the system.

4.2 States of tasks

As the multitasking system runs, we assign for each task one of these four states: Running, Ready for execution, Waiting or Unexisting as shown in figure 3[2]. The transition from one state to another is done through system calls or a decision made by the scheduler[13]. When a multitasking kernel decides to move the running task to another state and to give control of the CPU to a new task, a context switch should be performed [13].

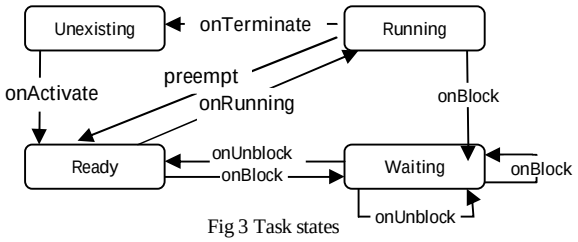

Fig 3 Task states

Each time a task enters the 'Ready' state (its methods onActivate() and onUnBlock()), it has to be added to the end of this list by calling its addLast() method. Each time a task leaves the 'Running' state (its methods onTerminat() and onBlock()), it has to be removed from this list by calling its remove() method. Besides, the first activation (onActivate) and the following activations (onUnblock) will add the corresponding task to the ready queue, whereas the events of termination of jobs (onBlock) or of task (onTerminate) correspond to a rejection of the corresponding task from the ready tasks queue. When a state changes from 'Ready' to 'Running' for such a task, it simply requires calling the onRunning method of its equivalent object[3].

5. EDF scheduler

The algorithm "Earliest Deadline First" (EDF) is a real-time scheduling algorithm[19] . It assigns priority to each task depending on the deadline and always select the highest priority task to execute, preempting lower priority tasks when necessary [2] [13]. EDF is particularly beneficial for time-sensitive workloads, such as multimedia and/or control applications [20]. Since EDF is more efficient than fixed-priority in term of schedulability, it will be easier for it to achieve high utilization. This algorithm is proved to be optimal in the sense that if a system of tasks can be sequenced using any policy of assigning priorities, the system can also be sequenced with the EDF algorithm [20]. The study of schedulability gives a necessary and sufficient condition formulated by the following theorem: a system of periodic tasks can be sequenced using the EDF algorithm if and only if:

$$\sum_{i=1}^{n} \left(\frac{C_i}{T_i}\right) \leq 1 \qquad (1)$$

T_i represents the period, C_i the worst execution time. Also, EDF ensures a maximum occupancy of the CPU up to an upper limit of 100 % CPU utilization.

6. Power management technique

The ultimate goal of any power management technique is to reduce an entity's consumption. When power management techniques are applied to microprocessors, they may improve the imminent energy shortage in portable equipment[21]. In order to extract the highest possible performance while simultaneously avoiding any associated instabilities, dynamic optimization algorithms must be based upon a deep and accurate understanding of the system behavior [22]. We cite the main used technique which are the DPM (Dynamic Power Management) and the DVFS (Dynamic Voltage and Frequency Scaling). Detailed description of power management technique is exposed in the next section.

6.1 DPM

The basic idea of DPM is to stop devices including the processor when they are not required and to wake them up when they are[24]. A Wireless Sensor Networks test-bed developed in cooperation with the start-up company SeNet s.r.l. for agricultural monitoring has measured the consumption of the nodes in differnt states [23]. Each node is composed of an IEEE 802.15.4 radio transceiver based on Chipcon CC2430 operating in the 2.4 GHz band, photovoltaic panels, rechargeable batteries, temperature and humidity sensors. It has shown that in sleeping mode, the consumption of the system is only 0.5 mA, whereas during the activity mode the overall consumption is 30 mA[22][24]. Currently, the DPM is done thanks to the ability of the hardware to support mechanisms of sleep state ranging from total activity of the system up to full sleep implementation or disconnection of the system. Indeed, idle state transitions and implementation cost is running a bit expensive in the point of view of energy.

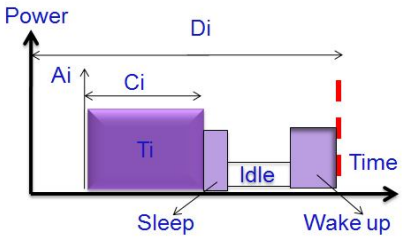

Fig 4 Optimization of consumption with DPM

6.2 Dynamic voltage and frequency Scaling

Dynamic voltage and frequency scaling (DVFS) is an efficient technique for reducing CPU energy. Most micro-processor systems are characterized by a time-varying computational load. It is better to run the processor at the weakest frequency compatible with the necessary performance level. When used at a reduced frequency, the processor can operate at a lower supply voltage [24]. DVFS exploits the CMOS property as shown in equation 2 that a linear reduction in the supply voltage results in a cubic reduction in the power consumption at the expense of a linear slowdown in the processor frequency.

$$P=A*C*V^{2}*F \quad (2)$$

P is the power consumed, A is the activity factor, i.e., the fraction of the circuit that is switching, C is the switched capacitance, V is the supply voltage, and F is the clock frequency.

Lowering only the operating frequency can reduce the power consumption but the energy consumption remains the same because the computation needs more time to finish. Lowering the supply voltage can reduce a significant amount of energy because of the quadratic relation between the power and the voltage. A number of modern microprocessors such as Intel's XScale and Transmeta's Cruso are equipped with the DVFS functionality [2].

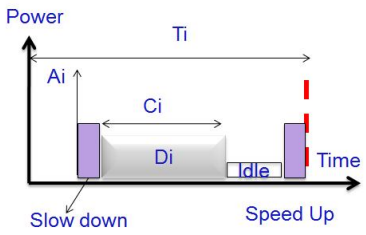

Fig 5 Optimization of consumption with DVFS

7. Test and results

In this section, we will expose the results obtained by the simulation of the DPM and DVFS techniques via STORM. The basic idea of the DMP is to stop the devices when they are not required and to wake them up when they are. The DVFS method modulates the voltage and frequency used by the CPU. Most of the time, the processor does not have to run at maximum speed, we can thus slow and largely reduce the consumed power, without loss of performance. A STORM strength is that we can easily extend the algorithm using an external java code and using the correct name in the appropriate "classname" tag. The method already implemented in STORM with the name dpm_leat is in fact, to declare the characteristics of the type leat_processors. This processor is PXA270. The PXA270 controller is based on the ARM core of the family of Marvell XScale (Ex-Intel). It has all the necessary features for a typical embedded application. It supports the dynamic adjustment of the power and the performance of the processor based on CPU demand. The frequency of the CPU can take the following values: 312/416 / 520/624 MHz. This kind of processor supports some low power consumption states for the DPM (example: sleep, deep sleep, standby etc). Admittedly, we have advocated the use of this processor for test purposes.

7.1 Experimentation

We based our work on the EDF scheduler "edf_p_scheduler" described in STORM. EDF is not real-time under Linux because of the time allocated by the kernel for decision-making. However, STORM is for the real-time execution, in the sense of the scheduling decisions taken by the kernel. In basic STORM EDF, when a processor is not used in select(), we choose a particular state such as sleep, deepsleep, standby, etc for PXA270. We have considered the following parameters which are also introduced in the XML file in figure 6. A number of temporal parameters are attached to each task in the simulation. They are either fixed or variable and updated over the events of the simulation.

```
- <!-- EDF_P Earlieast Deadline First Preemptive -->
- <!-- MULTIPROCESSOR -->
- <SIMULATION duration="1000">
  <SCHED className="storm.Schedulers.EDF_P_Scheduler" />
  <DPM className="storm.DPMS.DPM_P_LEAT" />
- <CPUS>
    <CPU classname="storm.Processors.leatprocessor" name="CPU A" id="0"
    idle_to_standby="1" idle_to_sleep="300" />
    <CPU classname="storm.Processors.leatprocessor" name="CPU B" id="1"
    idle_to_standby="1" idle_to_sleep="300" />
    <CPU classname="storm.Processors.leatprocessor" name="CPU C" id="2"
    idle_to_standby="1" idle_to_sleep="300" />
    </CPUS>
- <TASKS>
   <TASK classname="storm.Tasks.ptask_NAM" name="PTASK T1" id="1" period="80"
    activationdate="0" WCET="50" BCET="30" deadline="80" />
     <TASK classname="storm.Tasks.ptask_NAM" name="PTASK T2" id="2"
    period="100" activationdate="0" WCET="55" BCET="35" deadline="100" />
     <TASK classname="storm.Tasks.ptask_NAM" name="PTASK T3" id="3"
    period="120" activationdate="0" WCET="60" BCET="40" deadline="120" />
     <TASK classname="storm.Tasks.ptask_NAM" name="PTASK T4" id="4"
    period="150" activationdate="0" WCET="60" BCET="30" deadline="150" />
     <TASK classname="storm.Tasks.ptask_NAM" name="PTASK T5" id="5"
    period="200" activationdate="0" WCET="85" BCET="45" deadline="200" />
     <TASK classname="storm.Tasks.ptask_NAM" name="PTASK T6" id="6"
    period="250" activationdate="0" WCET="120" BCET="60" deadline="250" />
    </TASKS>
            </SIMULATION>
```

Fig 6 xml file to introduce the task parameters

The hardware architecture is composed of 3 identical processors. The identifier of the CPU is "LEATProcessor" which is a component of the library. Six tasks are present in the software architecture. Their own period, the date of the first activation, the worst case execution time (

WCET), the amount of time remaining to finish the current task RET (Remaining Execution Time), the EET (Effective Execution Time), the duration of the occupancy of the processor by the task and ET (or Execution Time) and the deadline (if it is not equal to the period) are internal to the simulator. The scheduling algorithm is in charge of fixing the computing time of each task when it is activated. By default, the 'Actual Execution Time' (AET) of a new task on arrival is simply equal to the WCET (the value which is specified in the input file XML).
The table1 shows the task parameters.

Table 1Task parameters

	Period	Activation	WCE	BCE	Deadline
1	80	0	50	30	80
2	100	0	55	35	100
3	120	0	60	40	120
4	150	0	60	30	150
5	200	0	85	45	200
6	250	0	120	60	250

For efficient manipulation, we migrate the tasks among the run queues (PTASKT1 is running on CPUA and the CPUB)in a way such that we always try to have, on an m CPU system, the m earliest deadline ready tasks running on the CPUs (here we have m=3). The temporal behavior of each task (its ability to meet its deadlines) is not affected by the behavior of the other tasks: if a task misbehaves and requires a large execution time, it cannot jeopardize the processor.

7.2 EDF simulation results

Sorting the queue of the ready tasks will be in ascending order of deadlines which are characteristic of tasks entered in the XML specification as shown in figure 6. We note that STORM checks initially the utilization ratio of processors and then assign the tasks so that they can run in parallel (figure 7). However, we note that some tasks switch from one CPU to another unlike their execution such as the task 3. Therefore, we should draw attention to the fact that the algorithm relies on the priority of the tasks more than on the processor affinity. Processor affinity refers to the tendency of a process to get scheduled constantly on the same processor[20].

7.3 DPM simulation results

We have chosen in our example an activation date equal to 0 for all the tasks in the XML specification as shown in figure 6. But, all tasks do not begin at t=0 because the number of the considered tasks is greater than the number of processors available in the simulation. This allocation of processors is established according to their availability and to the priority of the considered task. However, it allows us to browse through the chronogram. We have noticed that the revival of the processor (the transition from the DeepSleep state to Idle) is 261.75 ms what leads to an important delay for the effective execution of the task in its second activation. Thus, over the total duration of simulation, we will accumulate an important delay as illustrated in figure 8. Indeed, the cost of transitions between the states idle and starting is a little expensive from the energy point of view. Consequently, the implementation of the correct strategy of the transition between states is essential for the success of the DPM.

7.4 DVFS simulation results

The approaches suggested by DVFS are based on the principle of slowing down the execution of a task by lowering the frequency of the processor. The slowdown takes into account the deadline of the task. Thus, the choice of the frequency is not arbitrary and is quite related mainly to the policy of the considered scheduling policy and to the technology of the processor and the points of operation it offers. The speed of the processor is recalculated and updated after each activation. The more the task advances during its execution, the more the selected frequency is closer to the optimal one. In a schematic way, decreasing the frequency of a processor lengthens (with respect to the "normal" execution time) the execution time of the processor that is necessary to accomplish a task. Increasing this frequency reduces then the working time previously increased until, at best, finding the "normal" remaining execution time as shown in figure 9. The temporal cost for a change of frequency on this processor is null.

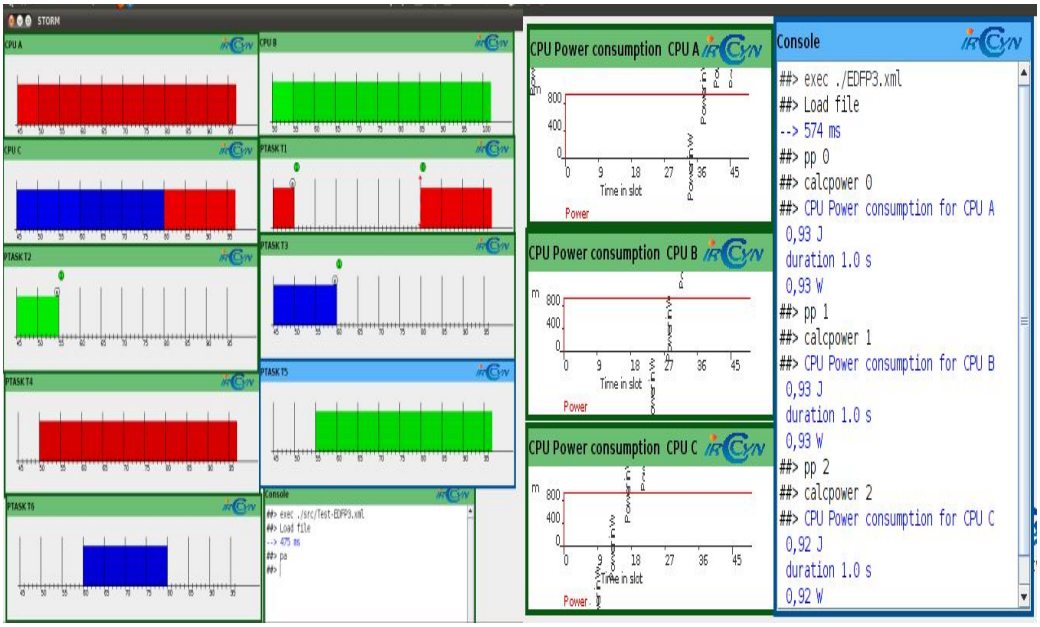

Fig 7 Execution of the tasks with EDF

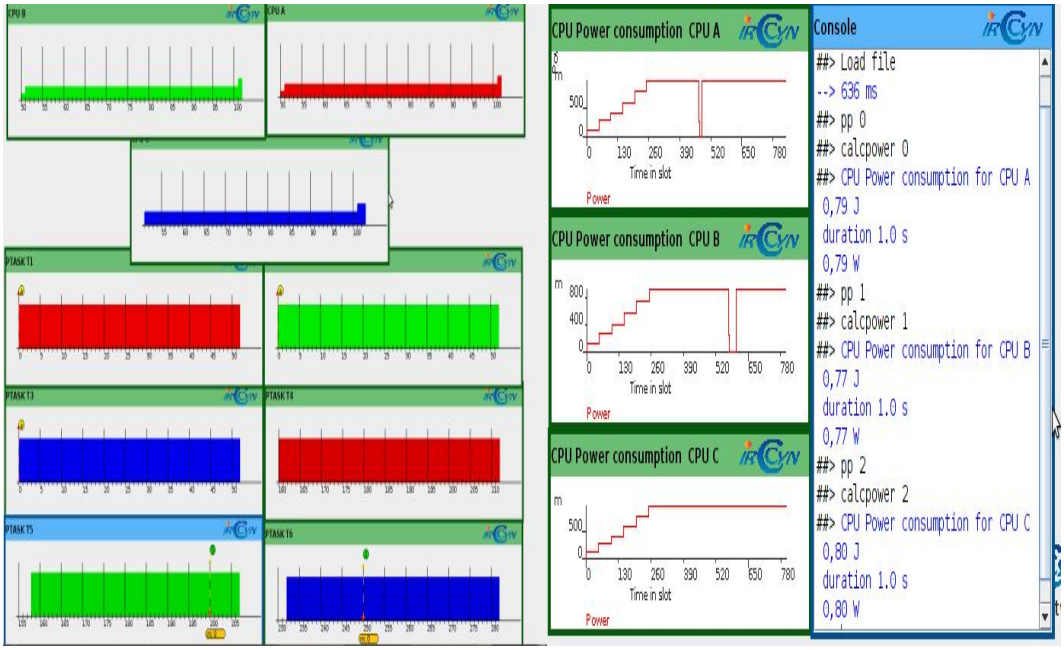

Fig 8 Task execution with DPM

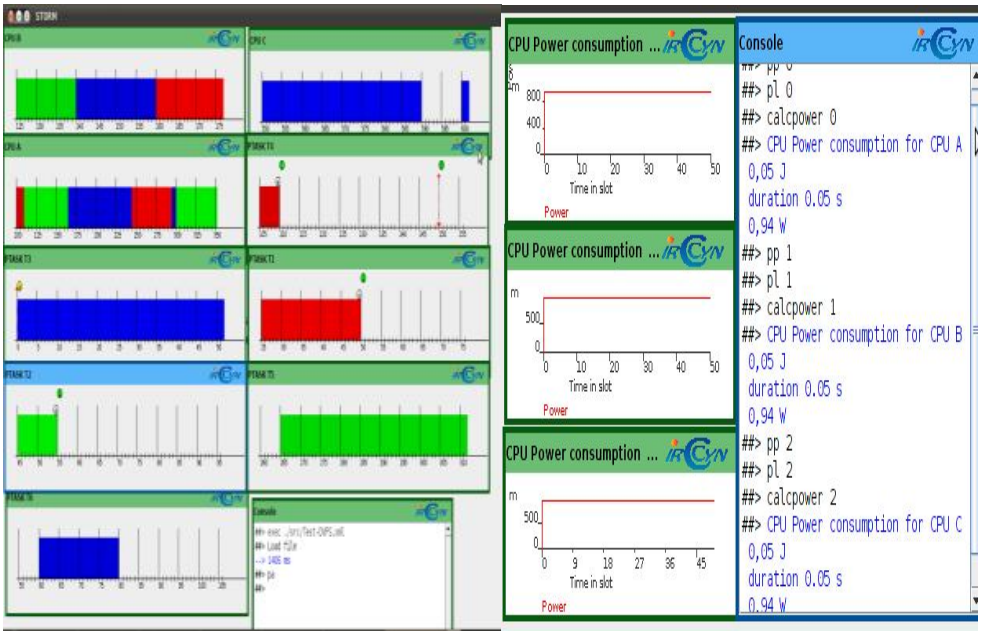

Fig 9 Task execution with DVFS

7.5 Comparison of results

We have proposed an approach which is to describe and compare the energy efficiency of various power management techniques and this combined with an EDF preemptive scheduler. Also, we could raise the energy profile of each processor in addition to the execution of the the tasks in order to give more visiblity concerning the performances of STORM to highlight the energy savings. In addition, we determined, via the simulator, the power consumed by each processor calculated for a period of 1 second. The CPU power consumption diagram of a processor shows over time its electrical power (in watts) computed according to the physical characteristics of its chip and its functioning states. Insofar, as the processors, generally, function with less possible downtime, the energy energy consumption and costs obtained for each technique of power management differ considerably. Energy saving can mostly reach 95% of the costs of the consumption compared to the use of EDF. At first glance, it should be noted that the techniques of energy management contribute contribute significantly to the reduction in power consumption, although the use of DVFS seems more efficient in terms of gain. It is about 0.05J only whereas without energy consideration it reaches 0.93J and this without negative incidence on the performance of the systems.

8. Conclusion

There is a growing recognition within different researchers' communities of the importance of simulation tools that help design and test new hardware and software approaches. That's why we have chosen the STORM simulator which is an under-development framework for real-time multiprocessor scheduling evaluation. The paper outlines several facets of STORM. Indeed, this tool gives a global insight about the execution of the different tasks and allows an evaluation via simulation of the potential gains with scheduling and power management techniques. As it supports multiple types of schedulers, through simulation we can compare scheduling algorithms in terms of schedulability performance as well as of energy efficiency. We have considered, particularly, the EDF scheduler by taking into account the task management service to provide significant energy savings while maintaining real-time deadline guarantees. Future works tries to improve the performances of this new simulator, to explore the memory architecture modeling, the partitioning and the hierarchy of schedulers…

References

[1] K. Lahmar, R. Cheour, M. Abid. Wireless Sensor Networks: Trends, Power Consumption and Simulators. Modelling Symposium, pp. 200-204, 2012.

[2] R.Cheour, S.Bilavarn, M.Abid, Exploitation of the EDF scheduling in the wireless sensors networks, International Journal on Measurement Technologies and Instrumentation Engineering, Volume 1, Issue 2, pages 14-27, 2011.
[3] STORM. http://storm.rts-software.org/, Sept. 2011.
[4] E.Saad, M.Awadalla, M.Shalan and A.Elewi, Energy-Aware Task Partitioning on Heterogeneous Multiprocessor Platforms. International Journal of Computer Science Issues (IJCSI), 2012.
[5] A.Noon, A.Kalakech, S.Kadry, A New Round Robin Based Scheduling Algorithm for Operating Systems: Dynamic Quantum Using the Mean Average, IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 3, No. 1, May 2011 ISSN (Online): 1694-0814.
[6] J.Sun, Y.Zhang, J.Fan, Providing Real-Time Guarantees to smart car, Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing, Xian, Shaanxi, pp 13-17, 2010.
[7] G. Buttazzo, P. Gai. Efficient EDF implementation for small embedded systems. In Proc. International Workshop on Operating Systems Platforms for Embedded Real-Time Applications. 2006.
[8] G.Buttazzo, Rate Monotonic vs. EDF: Judgment Day, Real-Time Systems, Vol. 28, pp. 1-22,
2005.
[9] R. Love Linux kernel development, Addison-Wesley Professional, 2010.
[10]R.Urunuela, A. Deplanche, Y.Trinquet , Simulation for multiprocessor real-time scheduling evaluation, 7th EUROSIM Congress on Modelling and Simulation, Prague, 2010.
[11] Pherma http://pherma.irccyn.ec-nantes.fr, Sept. 2011.
[12] R.Urunuela, A. Deplanche, Y.Trinquet, A Simulation Tool for Real-time Multiprocessor Scheduling Evaluation, 15th IEEE International Conference on Emerging Technologies and Factory Automation, (ETFA), Bilbao, Spain, 2010.
[13] R.Cheour, S.Bilavarn, M.Abid, EDF Scheduler for wireless sensor networks under Linux: case study, Fourth International Conference on Sensing Technology, Lecce, Italy, pp 530, June 2010.
[14] W. Lifeng and Y. Haibin, Research on a soft real-time scheduling algorithm based on hybrid adaptive control architecture, in Proc. American Control Conf, Lisbon, Portugal, pp. 4022-4027 vol.5, 2003.
[15] A.Youssef, A.Hamdy, R.Ammar, Efficient Processing Power Reservation Approach to Improve Real-Time Task Schedulability and Reliability. International Journal of Computer Science Issues (IJCSI). 9(5):196-205, 2012.
[16] A. Burns and A. J. Wellings. “RealTime Systems and Programming Languages”. Addison Wesley Longman, 4th edition, 2009.
[17] B. Sprunt, L. Sha, and J. P. Lehoczky. Aperiodic task scheduling for hard real-time systems. Real-Time Systems, 1(1):27–60, 1989.
[18] D. Isovic, G. Fohler. “Efficient Scheduling of S. poradic, poradic, Aperiodic, and Periodic Tasks with Complex Constraints” In Proc. Of 21st IEEE Real-Time Systems Symposium, Orlando Florida, USANovember 2000.
[19] C. L. Liu and J. W. Layland. “Scheduling algorithms for multiprogramming in a hard-real-time environment”. ACM, 20(1):46-61, January 1973.-
[20] J. A. Stankovic and M. Spuri and K. Ramamritham and G. Buttazzo, Deadline Scheduling for Real-Time Systems: EDF and Related Algorithms, Kluwer Academic Publishers, 0-7923- 8269-2, 1998.
[21] A.Abdelmotalib and W.Zhibo "Power Management Techniques in Smartphones Operating Systems." IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 3, No 3, May 2012.
[22] K.Lahmar, ‘Study and validation of an energy simulation environment in the WSN’, Master degree, National School of Engineers of Sfax, Tunisia, 2007.
[23] C.Buratti, A.Conti , D.Dardari , R.Verdone An overview on wireless sensor networks technology and evolution. Sensors. 2009;9:6869–6896.
[24] R.Cheour, K.Lahmar, M.Abid: Evolution of wireless sensor networks and necessity of power management technique, IEEE CAS, The 10th edition of Faible Tension Faible Consommation, Morocco, pp82, 2011.

Rym Cheour received the Engineering and M.S. degrees from the National School of Engineers of Sfax, Tunisia in 2008 and 2009, respectively. She is currently working toward the Ph.D. degree in Department of Computer Science at National Engineering School of Sfax, Sfax, Tunisia. Her research interests include Real Time scheduling, power management and wireless sensor networks

Urunuela Richard is a research engineer of the IRCCyN laboratory in the ”Real-Time Systems” group. Since 2003 he is interested in the design of real-time systems and more particularly: operating systems, power management for such systems, and real-time scheduling. At present, he is in charge of the implementation of STORM, a Simulation TOol for Real time Multiprocessor scheduling. He is also focusing on the development of engineering tools for helping to the design of power management policies. Previously he worked around scheduling and power management in the OBASCO research group at the Ecole des Mines of Nantes.

Yvon Trinquet is professor at the University of Nantes (Electrical Engineering department). He was leader of the Real-Time Systems team of IRCCyN laboratory since 1995 until 2012. His research focuses on real-time systems, especially real-time scheduling and simulation.

Mohamed Abid received the Ph.D. degree from the National Institute of Applicated Sciences, Toulouse (France) in 1989 and the “thèse d’état” degree from the National School of Engineering of Tunis (Tunisia) 2000 in the area of Computer Engineering and Microelectronics. He is working now as Professor in the Department of Electrical Engineering at National School ofEngineering of Sfax (Tunisia). Currently he is founding member and responsible of doctoral degree “Computer System Engineering” at ENIS since 2003. In 1992, he was founding responsible of Electronic Systems Synthesis Group at Laboratory of Electronic and Micro-Electronic in Sciences Faculty in Monastir (Tunisia). Since 2000 he is founding member of System on Chip at Computer, Electronic and Smart engineering system Laboratory at National School of Engineering of in Sfax (Tunisia). Since 2005, he has occupied the director position of the laboratory. His current research interests include: hardware–software co-design, System on Chip, Reconfigurable System, and Embedded System, etc. He has also been investigating the design and implementation issues of FPGA embedded systems.