DOI. 10.1109/SACI.2015.7208213

Analysis and Improvements in Energy Consumption Models for RTS

Cristina S. Stângaciu*, Andreea M. Horvath*, Mihai V. Micea*, Vladimir I. Creţu*, Voicu Groza**
* Computer and Software Engineering Department, Politehnica University Timişoara, Romania
** School of Electrical Engineering and Computer Science University of Ottawa
certejan@dsplabs.cs.upt.ro, mah.maria.horvath@gmail.com, mihai.micea@cs.upt.ro, vladimir.cretu@cs.upt.ro, groza@site.uottawa.cs

***Abstract*—In this paper we perform an analysis of the main software methods for consumption reduction in real time systems (RTS). The study covers both scheduling methods and energy consumption models for RTS, where we focus on sensor nodes. Further on, we propose a new consumption model for a large number of node configurations, which suites most of the main scheduling algorithms. We also introduce a software environment which integrates the new model and provides a set of reports and graphical results to compare the power consumption efficiency of the scheduling algorithms**

I. INTRODUCTION

Nowadays, time and energy behavior are two of the most important factors in real time application design. This aspect is emphasized by more and more studies around these two topics. As a result of the research in this area, different components of a device used in RTSs, like CPU, memory, network interfaces became subjects of the studies about energy consumption [1, 2]. The studies are more popular around the CPU component, which contributed to the development of the two most popular power aware scheduling techniques: dynamic voltage scaling (DVS) which is based on reducing energy consumption of the CPU by adjusting the processor speed and/or supply voltage and dynamic power management (DPM) which is based on the use of low power energy states (like sleep or stand-by) every time the processor is idle. A brief overview on the main scheduling techniques can be found in [3].

This article approaches the domain of real time power-aware systems with application on the area of sensor nodes, making analysis and improvements in energy consumption models for real time systems. The paper is structured as follows: A comparative analysis of the major real time power aware scheduling methods is presented in Chapter II - section A; an analysis of the existing models for real time power aware systems can be found in Chapter II - section B; an analysis of real time systems from hardware point of view with exemplification on sensor nodes is described in Chapter III; an unified energy consumption model which can be applied for any configuration of a sensor node is presented in Chapter IV; the implementation of the model in a simulator called INVERTA is presented in Chapter V - section A; graphic evaluation and results are presented in Chapter V - section B and conclusion in Chapter VI.

II. RELATED WORK

This chapter contains a comparative analysis of the main scheduling algorithms with the related existing consumption models, along with a comparison between these models.

A. *Scheduling Mechanisms for Real-Time Power-Aware Systems*

Based on the relatively new development of processors and other hardware components which have the capability to dynamically adjust their power consumption, numerous power-aware scheduling techniques have been developed. The most studied and popular techniques to decrease the energy consumptions are: dynamic power management (DPM) and dynamic voltage scaling (DVS) [3].

Dynamic Power Management is a mechanism which dynamically reconfigures a system to provide the requested services and tasks at the same performance level but, with a minimum number of active components or a minimum load on such components [4]. Mechanisms which change the system state dynamically in order to reduce the power consumption are widely applied not only in real-time systems. On physical platforms the state transitions have a non-negligible cost in terms of time and power consumption. So, when turning off a certain device we have to consider also the transition time, otherwise DPM may cause performance degradation. Switching from the active mode to the sleep mode and then back to the active mode takes time and requires additional energy overhead. For deciding whether to switch to another state, we have to check if the energy consumed by switching to that state is lower than if remaining in the current state. A study on the impact of the energy overhead is provided in [5]. In the literature are presented a number of approaches which take into consideration this aspect of energy overhead [6, 7].

On the other hand, the solution provided by the DVS method is to dynamically adjust the frequency of the device in such a way that the tasks finish execution before their deadline. The idea is to find the ideal frequency for lower power consumption while also ensuring that the tasks are finishing their execution before deadline. The DVS scheduling methods target the reduction of the dynamic power which is given by the next formula [8]:

$$P = C \cdot V_{dd}^2 \cdot f \quad (1)$$

where, P represents the power, C is the output load capacitance, V_{dd} the supply voltage and f the switching frequency.

The efficiency of a DVS algorithm depends very much on the performance of the estimation methods used by it. If we reduce the speed of the processor or of another device, the execution time of the tasks increases, which may lead to deadline violation. In hard real-time systems, violating task deadlines can have extreme consequences; that is why DVS can use only slack time and idle time to decrease the voltage levels and frequency. Some of the most important and

studied scheduling algorithms are: *EDF*, *RM*, and combinations or variations of these two, like *IDLE TIME*, *LPEDF* [9] or *SYS-CLOCK RM* [10]. *IDLE TIME* is a DVS method which adjusts the frequency of the device to the minimum value, when there is no task running. EDF adjusts the frequency of the device using an optimal value, which is the minimum working frequency to guarantee that no deadline is violated. For this scheduling algorithm, the optimal frequency scaling factor is given by the CPU load [8]. RM is similar to the EDF but the scaling factor is different [11]. The *Sys-Clock RM* algorithm determines an optimal working frequency by considering the maximum of the frequencies computed for each task considering its workload and its blocking time [10] and *LPEDF* is based on the idea that every task is running at each scheduling point with a different frequency, optimized for that particular time [9].

B. Existing Energy Consumption Models

The authors in [12] describe two scheduling algorithms: *Dual Speed Algorithm (DSA)*, which is a blocking aware scheduling algorithm with non-preemptive critical sections and *Enhanced Dual Speed Algorithm (EDSA)*, which is derivate from DSA algorithm. The task model considered here is described by the following parameters:

$$\theta_i \equiv \{r, T, D, C, B\} \quad (2)$$

where, θ_i represents the task *i*, *r* represents the release time, *T* the period, *D* the relative deadline, *C* the worst case execution time, *B* the blocking time.

For the DPM algorithm described in the previous chapter, there are models specified at the device level and not at the task level. For example in [7], for each device there are defined three states, each state is characterized by its power *P*. The device can transit from one state into another in a time marked as t_{sw}, with an energy overhead of E_{sw}.

For the DVS, statically applied with the classical EDF and RM scheduling algorithms, as well as for the SYS-RM technique, the task model can be described as [8], [10, 11]:

$$\theta_i \equiv \{T, C, D, f\} \quad (3)$$

where, θ_i represents the task *i*, *T* the period, *C* the worst case execution time, *D* the relative deadline, *f* the working frequency.

The system model used in the LPEDF method has the following parameters [9], with the same meaning as in equation (2):

$$\theta_i \equiv \{r, C, D\} \quad (4)$$

The following table summarizes the task model parameters used in different scheduling techniques:

TABLE I. PARAMETERS USED FOR EACH SCHEDULING ALGORITHM

Parameter /Algorithm	*T*	*D*	*C*	*R*	*B*	O_e	O_t	P_w	*F*
DVS EDF	X	X	X						X
DVS RM	X	X	X						X
DVS LPEDF		X	X	X					
DVS SYS-RM	X	X	X						X
DPM HAD			X	X		X	X	X	
DVS DSA	X	X	X	X	X				
DPM			X	X		X	X	X	

In TABLE I, *T* represents the period, *D* the relative deadline, *C* the execution time or worst case execution time, *R* the release time, *B* the blocking time, O_e the energy overhead, O_t the time overhead, P_w the power, *f* the working frequency. An attempt to use a unified power consumption model for both DVS and DPM was presented in [13], which models the processor power as a polynomial function of the processor working frequency (speed). On the other hand, this model does not consider the time and energy overheads at all.

III. TARGET SYSTEM

A relevant example of target system is the sensor node. The architecture and main components of the sensor nodes are presented in Fig. 1.

A sensor node can be defined as a low-power device that integrates computing, wireless communication and sensing capabilities. The main task of a sensor node in a sensor network is to monitor events: collect data, perform quick local data preprocessing and aggregation, and then transmit the data. Therefore, power consumption can be divided into three main domains: sensing, processing and communication. The model can be extended by adding extra modules like mobility or actuator modules. In wireless sensor networks, energy efficiency is crucial to achieve satisfactory network lifetime. In order to reduce the energy consumption of a node significantly, its radio module needs to be turned off as much as possible. Yet, this cannot be the case for most of the communication protocols implemented in sensor networks [14]. The best way to reduce the power consumption is to use specific methods for every component of a sensor, at every level: processor, radio module, sensing components, etc.

As a result, the general architecture of a sensor node consists on the following main components (see Figure 1):

1. Processing and storage unit: processes the data, executes the program code, controls the entire system, and stores the code and data;

2. Sensors and/or actuators: the interface to the physical world. They monitor and control the environment;

3. Communication unit: provides support for the information exchange between the nodes;

4. Power supply unit: provides the energy for the entire node.

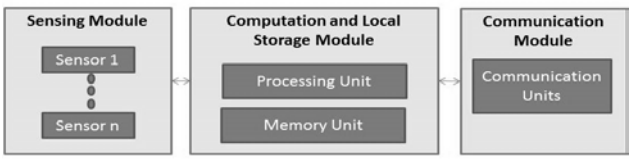

Figure 1. System architecture of the sensor node

A comparison between real-life examples of existing sensor nodes studied in the literature is summarized in TABLE II:

TABLE II. COMPARISON OF STATE OF THE ART SENSOR NODES

Node Model	Processor	Communication	Memory	Ref.
µAMPS	StrongARM SA-1100	LMX3162	512KB Flash	[15]
WINS	StrongARM SA-1100	Connexant's RDSSS9M	4MB Flash	[15]
Pico Node	DW8051	Proprietary		[15]
PushPin	Cygnal C8051F016	IrDA transceiver 83F8851		[15]
Eyes	MSP430F149	TR1000	8Mbit	[15]
WeC	AT90LS8535	TR1000	Chip: 24lC256 Connection: I2C, Size: 32KB EEPROM	[15]
Rene	AT90LS8535	TR1000	Chip: 24lC256 Connection: I2C, Size: 32KB EEPROM	[16]
Rene2	Atmega163	TR1000	Chip: 24lC256 Connection: I2C, Size: 32KB EEPROM	[16]
Dot	Atmega163	TR1000	Chip: 24lC256 Connection: I2C, Size: 32KB EEPROM	[16]
Mica	ATmega103L	TR1000	Chip: AT45DB041B Connection: SPI, Size: 512KB Flash	[15]
Mica2	ATmega128L	CC1000	Chip: AT45DB041B Connection: SPI, Size: 4 MB Flash	[15]
Mica2Dot	ATmega128L	CC1000	Chip: AT45DB041B Connection: SPI, Size: 512KB Flash	[16]
MicaZ	ATmega128L	CC2420		[17]
Medusa MK-2	ATmega128L AT91FR4081 ARMTHUMB	TR1000	1 MB Flash	[15]
iBadge	ATmega 103L TMS320VC5416	TR1000 Bluetooth ROK101007		[15]
XYZ	OKI ML67Q5002 ARM7TDMI	CC2420	256 K Flash, 32 K RAM, 4 K boot ROM, 2Mbits external ROM	[18]
Telos	TI MSP430	CC2420	Chip: STM25P80 Connection type: SPI Size: 1024 KB	[16]

IV. PROPOSED MODEL

Further on, we present a new energy consumption model which can be applied to a generic configuration of sensor nodes and can be used by both DVS and DPM scheduling algorithms. This new model was implemented in a C based simulator called INVERTA [19].

The proposed real time energy consumption model contains, besides the time parameters previously discussed, a set of parameters that describe the energy consumption of a task:

$$\theta_i \equiv \{T_i, C_i, D_i, \varphi_i, \Psi_i\} \quad (4)$$

where: θ_i represents the task i from the task set, defined by the parameters: T_i– the task period, C_i– the task execution time, D_i– the deadline, φ_i– the start time and– Ψ_i a vector which contains all the modules (devices) used by task i.

Regarding the task temporal parameters: the task period, execution time and start time must be specified absolutely: whether they are specified in measure units in international system (submultiples of second), whether they are specified in CPU clock cycles for a well-defined frequency, chosen as a reference and called nominal frequency. Usually this is the maximum operating frequency of the processor, under well-defined operating conditions. Next, we consider the temporal parameters of the task specified by the nominal frequency, which we choose to be equal to the maximum working frequency.

$$\Psi_i \in \{\psi_0, ..\psi_j, \psi_{j+1}, ...\psi_m\} \quad (5)$$

where: ψ_j is the basic module/device j.

Observation**:** The processor or the processor core, if it is further divided into submodules, has the index 0.

We define a basic module as the module which is not further divided into other components and for which the energy consumed in its various states of operation can be directly determined.

Each module is characterized by a set of states. So, for each device j of task i, there will be a vector of states. Each state is described by the following set of parameters:

$$\psi_{i,j} \equiv \{S_{i,j}^k, S_{i,j}^{k+1}, ..., S_{i,j}^n\} \quad (6)$$

where: $\psi_{i,j}$ represents the module/device j of task i and $S_{i,j}^k$ represents the energy state of the module $\psi_{i,j}$ while running the job k of task i.

The corresponding states can be predefined or user defined, a particular case being the predefined states for low consumption of a processor (low-power states). The parameters which define the energetic state of a device are defined below. Let k be the index that represents the job of a task θ_i, the energetic state of the device j while running the job k of task i becomes:

$$S_{i,j}^k = \begin{Bmatrix} f_{i,j\min}, o_{i,j\_act\_\max}, o_{i,j\_inact\_\max,}, P_{i,j}^k, \\ \eta_{i,j\_act\_\max}^k, \eta_{i,j\_inact\_\max}^k \end{Bmatrix} \quad (7)$$

where: f_{i,j_min} represents the minimum frequency at which the device must run so that no time constraints are violated; o_{i,j_act_max} represents the time for the worst case, due to a transition from one active configuration into the maximum active configuration (in terms of energy consumption); o_{i,j_inact_max} represents the worst case time overhead, due to a possible entry into an inactive configuration (the sum of the

maximum transition time from the maxim active configuration to the current inactive configuration and the maximum transition time from that inactive configuration back to the active one, for a specific device); $P_{i,j}^k$ represents the power consumption for the state $S_{i,j}^k$; η^k_{i,j_act_max} represents the maximum energy consumed during the transition from the energetic state corresponding to the current active state to the energetic state $S_{i,j}^k$ and η^k_{i,j_inact_max} represents the sum between the maximum energy consumed during a possible entry to an inactive state and the maximum energy consumed during a possible exit from that inactive state.

According to the energy formula, the energy consumed by device *j* during the execution time C_i of a job of task θ_i, is:

$$E_{i,j}(0,C_i) = \int_0^{C_i} P_i(t)dt \quad (8)$$

If the power function is constant, then the equation becomes:

$$E_{i,j} = P_j \cdot C_i \quad (9)$$

The worst case energy overhead η_i due to the transition in the state $S_{i,j}^k$ defined before, is added to this energy consumption. Thus, the total energy consumption of a module during the execution of a job of task *i*, becomes:

$$E_{i,j\_tot} = E_{i,j} + \eta_j \quad (10)$$

where: $\eta_j = \max(\eta_{i,j\_act}, \eta_{i,j\_inact})$, η_{i,j_act}, is equal to 0 if there is no transition from previous state to the current one, or equal to η^k_{i,j_act_max} otherwise. Similarly, η_{i,j_inact} is equal to 0 if there is no transition to an inactive state after the task finish its execution, being equal to η^k_{i,j_inact_max}, otherwise.

V. MODEL IMPLEMENTATION AND RESULTS ANALYSIS

The proposed consumption model has been implemented and integrated in a simulator called INVERTA [19]. It has been applied to the study of six different scheduling algorithms regarding their power efficiency for various configurations of sensor nodes. INVERTA (Integrated Visual Environment for Real-Time Application Analysis and Development) is a framework used for the design, analysis and validation of real-time applications for specific target platforms. The main scope of this module of the simulator is to find for a certain platform the configuration and the best scheduling algorithm in terms of time and energy performance. INVERTA also includes a graphical module which provides reports and graphs of valuable use for energy efficiency studies and comparisons. Fig. 2 depicts the main modules of INVERTA, with dark background being the ones which integrate the new model. Not only has the structure of a task been updated in order to integrate the model, but also all the scheduling algorithms have been modified to consider an entire sensor node with its different components, and not just the processor. Both the DVS and the DPM algorithms have been implemented to be studied and compared.

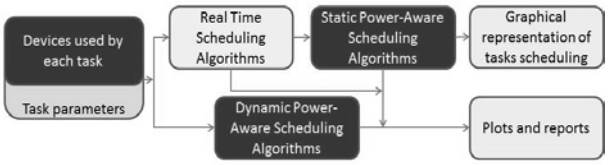

Figure 2. INVERTA modules

C. Model Implementation

The energy consumption model described in the previous chapter is used to describe the application tasks. Using this model, there were implemented in INVERTA, the following scheduling algorithms:

- DVS:
  - RM [11]
  - EDF [8]
  - IDLE TIME
  - LPEDF [9]
  - SYS-CLOCK RM [10]
- DPM [7]

For DVS, algorithms RM, EDF, SYS-CLOCK RM and IDLE TIME, the simulator computes the frequency scaling factor according to the selected algorithm and at each scheduling step all the devices that are used by the current task are targeted and the most suitable state $S_{i,j}^k$ for each device is selected. Depending on the computed frequency scaling factor value, the algorithm goes through all the states for each device and choses the state with the frequency equal to the value given by the scaling factor or the next higher frequency (See Code Sequence 1).

Code Sequence 1. Procedure for the DVS method

Procedure for DVS:

```
Input: a set of tasks with all the parameters described before, the Hyperperiod of the tasks, a scheduling list with the time stamps for launching each job into execution (TimeStamp).
For IDLE time:
Traverse the scheduling list
For each TimeStamp do:
    Check if exist idle time at current step:
    If yes,
        Find fmin for each device
        Calculate the energy with fmin
    If not,
        Calculate de energy with the corresponding frequency
  End IDLE time
For EDF, RM, Sys-RM:
Calculate de value for coef.
For EDF: coef = Σ(Ci/Ti)
For RM: coef = scaling factor which is calculated according to [11]
For Sys-RM: optimal frequency [10]
Calculate newHyperperiod = oldHyperperiod/coef.
Traverse the scheduling list until currentTime == newHyperperiod;
For each TimeStamp do:
     Find fmax for each device;
     Calculate fc = coef * fmax;
     Check if exist idle time at current step:
     If yes,
```

```
            newTimeStamp        =        max{curentTime,
prevoiusTimeStamp final time}
        If not,     newTimeStamp = oldTimeStamp / coef;
        Calculate Energy consumed.
```

For DPM each time a task is starting or ending its execution the algorithm takes the decision whether to go to the corresponding active or sleep state, whether to remain in the current state. This decision takes in consideration the time and energy needed to transit to the other state. If the energy needed to transit to other state is larger than the energy that would be consumed if the device remains in the current state, then the state is not changed yet [7]. The decision is taking by computing the so called Break-Even Time (BET). Only if the device idle time is greater than the BET computed for a certain inactive state, the device will be put into that corresponding inactive state (See Code Sequence 2).

Code Sequence 2. Procedure for the DPM method

Procedure for DPM:

```
Input: a set of tasks
```

Transit from active to sleep mode:

```
First use a scheduling algorithm to schedule the tasks
   For each device j:
      Traverse the scheduling list:
      For each TimeStamp
         Check if device j is used by the current task
         If yes,
            St = start time; Et = end time;
            Calculate BET = max{O_inact, eta_inactive /
(Pactive - Psleep)}
            If (St - previousEt) > BET
               Transit to the selected inactive state;
               Calculate Energy consumed in the selected
inactive state
            else,
               Remain in active state;
               Calculate energy consumed in the current
active state
          If not,
             Go to next TimeStamp.
```

Transit from sleep to active mode:

```
  First use a scheduling algorithm to schedule the set of
tasks
  For each device j:
      Traverse the scheduling list:
      For each TimeStamp
         Check if device j is used by the current task
         If yes,
             Calculate energy consumed in active state
         If not,
             Go to the next TimeStamp.
```

D. Results analysis

In this chapter we present a set of graphics and results which were obtained after using the simulator. In order to do this, first, we consider the next set of tasks (Task set 1):

Task 0: Period 15, WCET 2, Deadline 15

Task 1: Period 20, WCET 3, Deadline 20

Task 2: Period 30, WCET 4, Deadline 30

For each task we have considered two devices that the corresponding task uses to perform its execution. Task 0 and task 2 uses both devices and task 1 uses only the processor. For these inputs you can see below the schedules for each technique (Fig. 3), where the *id0* represents task 0, *id1*, task 1 and so on, the time is represented in nominal CPU clock cycles, and the rectangles represent the execution of each time: the height represents the processor frequency at which, each task is executed, and the length represents the actual execution time:

TABLE III. PARAMETER VALUE EXAMPLES

State Code	Descrip-tion	P [mW]	Freq. [MHz]	o_{act_max} [us]	o_{inact_max} [us]	η [uJ]
EFM32G890F128 Microcontroller						
S1	EM1	5,6	32	-	7.37	114.16
A1	EM0	18.54	8	24.25	-	61.83
A2	EM0	21.09	32	-	-	-
LIS3DH Accelerometer						
S1	Power-down	$1.25 \cdot 10^{-3}$	-	-	180	4.725
A1	Low-Power	$15 \cdot 10^{-3}$	$50 \cdot 10^{-6}$	180	-	2.25
A2	Normal	$27.5 \cdot 10^{-3}$	$50 \cdot 10^{-6}$	-	-	-

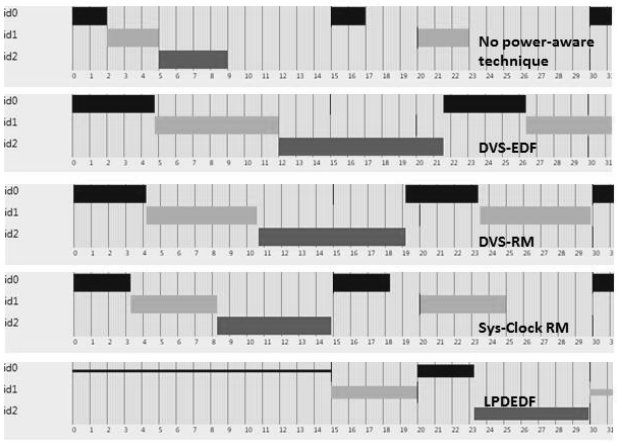

Figure 3.Comparative Representation of Task Scheduling with Different Algorithms

In Fig. 4 we can see a comparison regarding the energy consumption when running the task set specified above. The task set was simulated with the same device configuration. Fig. 5-7 represent the energy efficiency of a certain algorithm in function of the CPU load. Such studies were already presented in the literature and are out of the scope of this article. The figures are presented only as usage example of the proposed model.

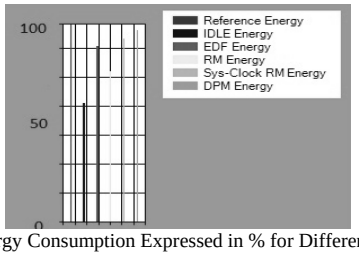

Figure 4. Energy Consumption Expressed in % for Different Algorithms

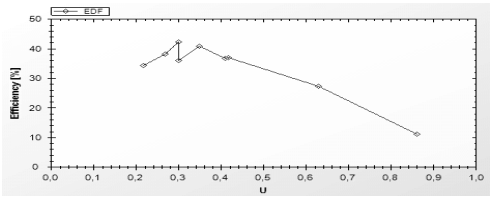

Figure 5. Energy Efficiency function of CPU Load for DVS-EFM

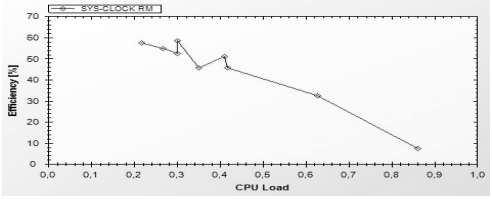

Figure 6. Energy Efficiency function of CPU Load for SysClock RM

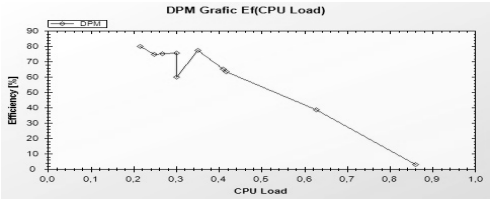

Figure 7. Energy Efficiency function of CPU Load for DPM

VI. CONCLUSIONS

The paper provides an analysis of the most popular task and device models used in different power-aware scheduling techniques, it proposes a unified and original model for specifying the temporal and energy behavior of a task and presents a set of reports and graphical results which can be used in comparing the energy consumption of different power-aware scheduling techniques.

The proposed model can be easily integrated in most of the existing power-aware scheduling algorithms. It also has led to the development of a unified simulator which was used for analyzing and comparing the time and energy behavior of different power-aware scheduling algorithms running on different architectures, having the only condition that the devices are specified by the requested parameters from the model. Examples of such comparisons and analysis can be seen in Fig. 5-7.

The model can also be used for developing hybrid power-aware scheduling algorithms which take into consideration not only the processor, but practically all the devices which can be specified according to the model.

ACKNOWLEDGMENT

This work was partially supported by the strategic grant POSDRU/159/1.5/S/137070 (2014) of the Ministry of National Education, Romania, co-financed by the European Social Fund – Investing in People, within the Sectoral Operational Programme Human Resources Development 2007-2013.

REFERENCES

[1] L. Niu, "Energy efficient scheduling for real-time embedded systems with QoS guarantee," *Real-time systems*, vol. 47, no. 2, pp. 75-108, 2011.

[2] D.-R. Chen;, C.-C. Hsu;, and M.-F. Lai;, "Time-Efficient Power-Aware Scheduling for Periodic Real-Time Tasks," in *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, 2009, pp. 1-8.

[3] C. S. Stangaciu, M. V. Micea, and V. I. Cretu, "Energy efficiency in real-time systems: A brief overview," in *Applied Computational Intelligence and Informatics (SACI), 2013 IEEE 8th International Symposium on*, 2013, pp. 275-280.

[4] L. Benini, A. Bogliolo, and G. De Micheli, "A survey of design techniques for system-level dynamic power management," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 8, no. 3, pp. 299-316, 2000.

[5] M. Bambagini, M. Bertogna, M. Marinoni, and G. Buttazzo, "On the Impact of Runtime Overhead on Energy-Aware Scheduling," presented at the Workshop on Power, Energy, and Temperature Aware Real-time Systems (PETARS 2013), Philadelphia, USA, 2013.

[6] M. A. Haque, H. Aydin, and D. Zhu, "Energy-Aware Standby-Sparing for Fixed-Priority Real-Time Task Sets," *Sustainable Computing: Informatics and Systems*, no. 2014.

[7] K. Huang, L. Santinelli, J.-J. Chen, L. Thiele, and G. C. Buttazzo, "Adaptive Dynamic Power Management for Hard Real-Time Systems," in *Real-Time Systems Symposium, 2009, RTSS 2009. 30th IEEE*, 2009, pp. 23-32.

[8] H. Aydin, R. Melhem, D. Mosse, and P. Mejia-Alvarez, "Dynamic and Aggressive Scheduling Techniques for Power-Aware Real-Time Systems," in *Real-Time Systems Symposium, RTSS* Proceedings. 22nd IEEE 2001, pp. 95- 105.

[9] B. C. Mochocki, "Voltage scheduling techniques for dynamic voltage scaling processors with practical limitations," University of Notre Dame, 2004.

[10] S. Saewong and R. Rajkumar, "Practical Voltage-Scaling for Fixed-Priority RT-Systems," in *Real-Time and Embedded Technology and Applications Symposium*, Proceedings. The 9th IEEE 2003, pp. 106-114.

[11] Y. Shin, K. Choi, and T. Sakurai, "Power Optimization of Real-Time Embedded Systems on Variable Speed Processors," in *Computer Aided Design, 2000. ICCAD-2000. IEEE/ACM International Conference on*, 2000, pp. 365-368.

[12] A. M. Elewi, M. H. Awadalla, and M. I. Eladawy, "ENERGY EFFICIENT REAL TIME SCHEDULING OF DEPENDENT TASKS SHARING RESOURCES," in *Proceedings of the 2008 High Performance Computing and Simulation Conference (HPCS)*, 2008, pp. 107-116.

[13] M. Bambagini, G. Buttazzo, and M. Bertogna, "Energy-Aware Scheduling for Tasks with Mixed Energy Requirements," presented at the Real-Time Scheduling Open Problems Seminar (RTSOPS), 2013.

[14] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella, "Energy conservation in wireless sensor networks: A survey," *Ad Hoc Networks*, vol. 7, no. 3, pp. 537-568, 2009.

[15] M. A. M. Vieira, C. N. Coelho, Jr. , D. C. da Silva, Jr. , and J. M. da Mata, "Survey on wireless sensor network devices " in *Emerging Technologies and Factory Automation, 2003.*, Proceedings. ETFA '03. IEEE Conference, 2003, pp. 537-544.

[16] J. Polastre, R. Szewczyk, and D. Culler, "Telos: enabling ultra-low power wireless research," in *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, 2005, pp. 364-369.

[17] M. Krämer and A. Geraldy, "Energy measurements for micaz node," *5. GI/ITG KuVS Fachgespräch „Drahtlose Sensornetze "*, no. p. 61, 2006.

[18] D. Lymberopoulos and A. Savvides, "XYZ: a motion-enabled, power aware sensor node platform for distributed sensor network applications," in *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, 2005, pp. 449-454.

[19] M. V. Micea, "HARETICK: A real-time compact kernel for critical applications on embedded platforms," in *Proc. 7th Intl. Conf. Development and Applic. Syst., DAS*, 2004, pp. 16-23.