ENERGY EFFICIENT SCHEDULING SIMULATOR FOR DISTRIBUTED REAL-TIME SYSTEMS

SANTHI BASKARAN\textsuperscript{1}, VARUN KUMAR P.\textsuperscript{2}, VEVAKE B.\textsuperscript{2} & KARTHIKEYAN A.\textsuperscript{2}

\textsuperscript{1}Assistant professor, Department of Information Technology
\textsuperscript{2}Students, Final Year Information Technology
Pondicherry Engineering College, Puducherry, India
(E-mail: vvaarruunn24@gmail.com, vvek.9291@gmail.com, kar17thi@gmail.com)

Abstract: The most critical design issue in real-time systems, especially in battery-operated systems where functional complexity continues to expand is Energy Consumption. The major area of focus is to provide a real time scheduling simulator which provides features to check if a real-time application meets its temporal constraints while minimizing system energy consumption using several feasibility tests in preemptive/non preemptive cases. Feasibility tests allow a user to study a real time application/system without performing actual scheduling. With this simulator tool, a real-time application is defined by a set of processors, tasks, shared resources and messages. The simulator has a feature to dynamically adjust the processor speed according to the algorithm fed, given input of a hardware profile and workload. The flexible simulation engine also allows the designer to describe and run simulations of specific real-time systems while providing dynamic support for real time, precedence and resource constrained tasks.

Keywords: Dynamic voltage scaling, real-time scheduling, simulation tool, voltage scalable processor.

I. INTRODUCTION

Most of the real-time systems in the world are powered by batteries. They are extensively used for performing multitude of tasks. The rise of functional complexity and quantity of battery powered devices makes the energy efficient design of such devices increasingly important. Also these kind of real-time systems are subjected to concurrently perform a wide variety of complex tasks under stringent time constraints. Hence, minimizing the power consumption of real-time systems while satisfying the stringent timing constraints have become an integral aspect in designing such systems.

Scaling is very essential for saving energy. Dynamic Voltage Scaling (DVS) is used to dynamically change speed and Voltage at run time by spreading run cycles into idle time. Because of the non-linear relationship between CPU speed and power consumption, it is better to spread execution by reducing cycle time and voltage, than to run the CPU at full speed for short burst and then changing to idle mode. Energy reduction can be obtained using lower voltage and frequency. The need for reducing the processor voltage and frequency to a very low level is eliminated but instead power aware real-time scheduling can be used. This scheduling process satisfies the timing constraints of real-time tasks. The voltage and frequency are dynamically adjusted according to some optimization criteria such as high throughput or low energy consumption. However, reduction of processor voltage and frequency increases the circuit delay, causing slowdown in the execution of programs. Hence, energy efficient real-time scheduling makes a trade-off between energy saving and system performance.

The two kinds of scaling that allow adjusting processor voltage and frequency at runtime are Dynamic Voltage Scaling (DVS) and Dynamic Frequency Scaling (DFS). The higher the processor voltage and frequency, the higher will be the system throughput. Energy reduction can be obtained using lower voltage and frequency.
Few of the recently popular trends in modern processor architecture which provide support for DVS and DFS mechanisms are Intel Mobile Processors with SpeedStep technology [1], AMD Mobile Processors with Power Now! Technology [2], etc... DVS and DFS can be implemented at various levels of a system, such as in the processor, in the OS scheduler, in the compiler or in the application.

Enhanced Intel SpeedStep Technology allows the processor performance and power consumption levels to be modified while a system is functioning. This is accomplished via application software, which changes the bus-to-core frequency ratio and the processor core voltage (Vcc). Enhanced Intel SpeedStep Technology is supported on current and future generations of Intel Pentium® M Processors. The Intel Pentium M Processor at 1.6 GHz supports six frequency and voltage operating points as below. The top and bottom modes are commonly known as high frequency mode (HFM) and low frequency mode (LFM).

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.6 GHz HFM</td>
<td>1.484 V</td>
</tr>
<tr>
<td>1.4 GHz</td>
<td>1.420 V</td>
</tr>
<tr>
<td>1.2 GHz</td>
<td>1.276 V</td>
</tr>
<tr>
<td>1.0 GHz</td>
<td>1.164 V</td>
</tr>
<tr>
<td>800 MHz</td>
<td>1.036 V</td>
</tr>
<tr>
<td>600 MHz (LFM)</td>
<td>0.956 V</td>
</tr>
</tbody>
</table>

The framework of simulator tool is based on the real-time scheduling theory. Energy efficient real-time simulator tool provides services to study the temporal behavior and energy consumption of real-time applications. Scheduling simulation requires, first to compute a scheduling on a given time interval and second, to look for timing properties in this computed scheduling. For a real-time simulator tool an application is defined as processors, tasks, messages and shared resources. The processor definition may be mono, multi or distributed. The tasks may be defined with timing, precedence, resource and energy constraints. From a set of tasks, for a given system, two kinds of analysis can be performed: Scheduling simulation and feasibility tests.

Scheduling simulation consists in predicting for each unit of time, the task(s) to which the processor(s) to be allocated. Checking if tasks meet their deadline is done by analyzing the computed scheduling. For a given task set, if a scheduling simulation is very long to compute, feasibility tests can be applied instead.

### 1.1. Motivation

This work was motivated by the lack of free, flexible and open energy-efficient scheduling tools for real-time systems. The design and development of this work is carried out to fulfil three main requirements. First, to provide a framework, which implements most of the classical real-time scheduling algorithms with energy efficiency. Second, to apply feasibility tests for uni-processor, multiprocessor and distributed real-time systems. Third, to develop and add new energy-efficient scheduling algorithms to this simulator tool.

### 1.2. Objectives

The objective is to create energy efficient scheduling tool for investigating scheduling in real-time application with,

- Periodic, aperiodic and mixed independent task sets
- Dependent tasks having precedence and resource constraints that are normally represented by a task graph.

The objective is also extended to provide energy efficient scheduling algorithms for real-time tasks in uni-processor, multiprocessor and distributed computers for soft, hard and weakly-hard real-time systems.

### 2. BACKGROUND CONCEPTS

Operating system is the only component with an overview of the entire system, including task constraints and status, resource usage, etc. Therefore, it is believed that, this would be one of the most effective and efficient approaches to reduce energy consumption with proper task scheduling algorithms. The real-time scheduling
problem with power optimization constraints is NP-hard [5]. It is time consuming to find an optimal schedule where energy consumption is minimized and all timing constraints are met. Many previous works either proposed offline scheduling for large energy reduction, or used heuristic methods to reduce scheduling overhead. However, while the former approaches are inflexible and too costly to store in memory, the latter ones may not realize the full potential of energy savings.

Energy-efficient real-time technique is then used online [6] based on the information, such as task start times, finish times, pre-emption times, etc, to make better scheduling decisions. Furthermore, profiling tool [7] is implemented to provide time information at program runtime for scheduler to adjust schedule online according to the actual execution time (AET). Slack time (time between task completion time and task deadline) thus can be better utilized for energy-efficient real-time scheduling.

Extensive power aware scheduling techniques have been published for energy reduction, but most of them have been focused solely on reducing the processor energy consumption. While the processor is one of the major power hungry units in the system, other peripherals such as network interface card, memory banks, disks also consume significant amount of power.

By using the Dynamic Power Down (DPD) technique it is possible to shut down a processing unit and save power when it is idle. There is a minimal time interval that the device can be feasibly shut down with positive energy-saving gain. Only when the slack time or idle time is more than the minimum time interval the DPD is used.

3. PROBLEM DEFINITION

To develop an energy-efficient real-time scheduling simulator with two independent parts: a graphical editor used to describe a real-time application/system, and a framework which includes most of classical real-time scheduling/feasibility algorithms/ tests with energy consideration.

The main features to be provided are,

- Framework to do energy-efficient scheduling simulation with classical real time schedulers.
- Features to apply many feasibility tests in the pre-emptive case and in the non-pre-emptive case.
- Support for periodic, aperiodic and mixed tasks.
- Support for precedence and resource constrained tasks.
- Scheduling simulation of applications distributed on several processor and sharing messages.
- Support to design new schedulers, task pattern arrivals and event analyzers.

4. FEATURES OF SIMULATOR

Energy efficient real time scheduler tool developed basically provides two kinds of features:

- Simulator
- Scheduling feasibility test.

Feasibility tests allow a user to study a real time application/system without performing actual scheduling. The simulator dynamically adjusts the processor speed according to the algorithm, given input of a hardware profile and workload. The hardware profile describes the multiple operating points of the processor, each with a separate frequency and power consumption. Unless otherwise specified we use the following hardware profile \{(1.6, 1.484), (1.4, 1.420), (1.2, 1.276), (1.0, 1.164), (800, 1.036), (600, 0.956)\}, where 1.6, 1.4, 1.2, 1.0, 800, 600 denote the processor frequency in GHz and MHz, and 1.484, 1.420, 1.276, 1.164, 1.036, 0.956 denote the corresponding voltage.

This simulator can be used to find which algorithm suits best to any given set of inputs. Since the task set is generated randomly using statistical distributions, the simulation works best at all sorts of situation and also gives the required result.

The real-time scheduling simulator is composed of two parts:

- User Interface to accept inputs and display results in a graphical format.
- Background Execution of algorithms.
The simulator is basically developed using ‘C’ language. The user interface is developed using Visual Basic. This provides the user with screens to give inputs such as number of tasks and number of processors.

The simulator can be used firstly to compute a scheduling and secondly, to compare various real-time scheduling algorithms in the literature. Using this simulator, three existing scheduling algorithms namely Red Tasks Only (RTO) algorithm and Blue When Possible (BWP) algorithm [13] and Red as Late as Possible (RLP) algorithm [14], designed for overloaded real-time systems that allow skips are considered for energy efficiency [15].

Few sample interface forms in the simulator framework are as shown in Figure 3.

5. INPUT DESCRIPTION

The input data like number of tasks, number of processors, etc. are acquired from the user through the user interface, depending on the algorithm chosen from the framework. They are stored in a file and are given to random task generation program. This generates attributes like worst case execution time, arrival time, actual execution time, deadline, and resource constraints etc. depending on the algorithms chosen. The randomly generated inputs are used by all the related algorithms taken into consideration and scheduling simulation is done.

The random task generation program also generates precedence constraints among the given number of tasks, in the form of a Directed Acyclic Graph (DAG). The DAG generated can be used by all the algorithms meant for tasks with precedence constraints. For multiprocessor and distributed systems, the number of processors is varied from 2 to 10.

6. OUTPUT DESCRIPTION

The system output describes graphically how each and every task is being scheduled on different processors. It also contains the details regarding the speed, energy consumed by the variable voltage processor and also the increase in the computation time of tasks to show graphically. The output from the simulator is also stored in files. The output files contain the details about the processor Id and energy consumed by each and every variable voltage scalable processor, if the task sets satisfy the necessary constraints.
The output interface consists of a form which has the output files displayed in a Microsoft Chart control. There are button click events for viewing the energy consumed by the variable voltage scalable processor, for viewing the task sets which contains the information of all the tasks, and for the graphical representation of the output. The simulator framework also has provision to view the comparison of power consumption among variable voltage scalable processors of a distributed or multiprocessor system, for the algorithms selected through the input interface.

7. IMPLEMENTATION AND ANALYSIS OF RESULTS

The simulator developed simulates voltage scalable processors which dynamically adjusts the processor speed according to the proposed algorithm. A continuous voltage scaling model is used and hence the processor speed can be adjusted continuously from its maximum speed to a minimum speed which is assumed to be 25% of its maximum speed.

Scheduling task sets and task graphs are generated using the following approach:
- Task sets are randomly generated with parameters such as arrival time, acet, and wcet.
- The wcet is taken randomly and acet is also randomly generated such that it is 40 to 100 per cent of wcet.
- The overall deadline is generated such that it is always greater than or equal to the sum of acet of all the tasks in DAG.

Task graph is randomly generated using adjacency matrix where 0 represents the tasks that are not dependent on any other tasks and 1 represents the dependency, with varying breadth and depth.

The performance parameter considered for evaluating and comparing our new algorithms with the existing algorithms is overall Power Consumption of the voltage scalable processors used. It is calculated as the average of power consumed by individual processors used to execute the application. The power consumption when a processor runs at maximum speed is 100% and it is calculated proportionately for reduced speed, as the relationship between power dissipation \( P_d \), supply voltage \( V_{dd} \), and frequency \( f \) is represented by
\[
P_d = C_{ef} V_{dd}^2 f \quad \text{and} \quad f = k \frac{(V_{dd} - V_t)^2}{V_{dd}},
\]
where \( C_{ef} \) is the switched capacitance, \( k \) is the constant of circuit, and \( V_t \) is the threshold voltage [16].

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Power Consumption (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Earliest Deadline First</td>
<td>45</td>
</tr>
<tr>
<td>Rate Monotonic Scheduling</td>
<td>44</td>
</tr>
<tr>
<td>Least Laxity First</td>
<td>32</td>
</tr>
</tbody>
</table>

The power consumption in percentage is compared among the existing and proposed algorithms by varying the number of tasks from 2 to 20 and the number of processors from 2 to 10. The maximum number of tasks is taken to be 20, because real-time applications with dependent tasks above 20 are very rare.

8. CONCLUSION

An energy-efficient scheduling simulator tool for real-time systems is designed. This tool is developed for real-time applications that can run on uni-processor, multiprocessor and distributed systems. New scheduling algorithms are added to the framework. Existing classical real-time scheduling algorithms are studied and modified to achieve increased energy efficiency. The scheduling algorithms considered are capable of scheduling periodic, aperiodic tasks with
precedence and resource constraints. In the future
more user-defined schedulers and task activation
patterns can be added to this simulator
framework. Feasibility tests are limited to usual
task models and schedulers. New feasibility tests
for other real-time task models are to be developed
and added to this framework.

References

Dynamic Voltage Scheduling Using Program
Checkpoints”, In Conference on Design, Automation
and Test in Europe, 2002.
Scheduling Algorithms for Minimizing Energy/
Power”, IEEE Transactions on Very Large Scale Integration
Scaling for Low-power Embedded Operating Systems”,
In ACM Symposium on Operating Systems Principles,
[7] C. Shen, K. Ramanritham and J. A. Stankovic,
“Resource Reclaiming in Multiprocessor Real-time
Systems”, IEEE Trans. Parallel and Distributed Systems,
Real-Time Scheduling in Distributed Systems,”
International Journal of Computer Science Issues, Vol. 7,
Scheduling Algorithms for Battery-powered DVS
Systems”, IEEE Trans. On Very large Scale Integration
of Periodic Task Graphs and Aperiodic Tasks in
Distributed Real-time Embedded Systems”, Int. Conf.
Scheduling and Voltage Selection for Energy
Minimization”, Design Automation Conference, pp. 183-
188, June 2002.
for Energy Minimization on Parallel Machines”, 16th
Euromicro Conference on Parallel, Distributed and
Network-Based Processing, pp. 399-406, 2008.
Scheduling of Skippable Periodic Tasks in Weakly-Hard
Real-Time Systems”, IEEE International Conference and
Workshops on the Engineering of Computer-Based
Scheduling of Skippable Periodic Tasks with Energy
Efficiency in Weakly Hard Real-time Systems”,
International Journal of Computer Science & Information