IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 3, No 4, May 2010
ISSN (Online): 1694-0784
ISSN (Print): 1694-0814

Energy Efficient Real-Time Scheduling in Distributed Systems

Santhi Baskaran[1] and P. Thambidurai[2]

**[1] Department of information Technology, Pondicherry Engineering College
Puducherry – 605 008, India**

**[2] Department of Computer Science and Engineering, Pondicherry Engineering College
Puducherry – 605 008, India**

Abstract

Battery powered real-time systems have been widely used in many applications. As the quantity and the functional complexity of battery powered devices continue to increase, energy efficient design of such devices has become important. Also these real-time systems have to concurrently perform a multitude of complex tasks with strict time constraints. Thus, minimizing power consumption and extending battery life while guaranteeing the timing constraints has become a critical aspect in designing such systems. Moreover, energy consumption is also a critical issue in parallel and distributed systems. We present novel algorithms for energy efficient scheduling of Directed Acyclic Graph (DAG) based applications on Dynamic Voltage Scaling (DVS) enabled systems. Experimental results show that our proposed algorithms give better results than the existing algorithms.

Keywords: *Real-time, slack, energy reduction, scheduling*

1. Introduction

Energy consumption reduction is becoming nowadays an issue reflected in most aspects of our lives. The obvious driving force behind addressing energy consumption in digital systems is the development of portable communication and computation. The consumers demand better performance and more functionality from the hand-held devices, but this also means higher power and energy consumption. Hence energy efficiency is an important optimization goal in digital system design, and most of these are in fact time critical systems.

Timeliness and energy efficiency are often seen as conflicting goals. When designing a real-time system, the first concern is usually time. Yet, with the right methods energy efficiency can also be achieved. Energy-efficient architectures may be selected, while still meeting the timing constraints.

The ready availability of inexpensive processors with large memory capacities and communication networks with large bandwidth are making it more attractive to use distributed (computer) systems for many of the real-time applications. Distributed real-time systems have emerged as a popular platform for applications such as multimedia, mobile computing and information appliances. For hard real-time systems, application deadlines must be met at all time. However, early completion (before the dead- line) of the applications may not bring the systems extra benefits. In this case, we can trade this extra performance for other valuable system resources, for example, energy consumption. Dynamic voltage scaling (DVS), which varies the processor speed and supply voltage according to the workloads at run-time, can achieve the highest possible energy efficiency for time-varying computational loads while meeting the deadline constraints [1]. DVS takes advantage of the quadratic relationship between supply voltage and energy consumption [2], which can result in significant energy savings. There are many commercially available voltage-scalable processors, including Intel's Xscale [3], Transmeta's Crusoe [4], and AMD's mobile processors with AMD PowerNow! technology support [5].

An important problem that arises from using distributed systems is how to assign tasks and resources to the processors so as to fully utilize the processors, while ensuring that the timing constraints of the tasks are met along with energy efficiency. Hence in this paper we address energy efficiency in the context of distributed real-time systems, targeting variable speed processor architectures.

In energy efficient scheduling, the set of tasks will have certain deadline before which they should finish their

ISSN (Online): 1694-0784
ISSN (Print): 1694-0814

execution and hence there is always a time gap between the actual execution time and the deadline. This time gap is called slack. In this paper, we have proposed algorithms to allocate this slack in an efficient way therefore resulting in more energy savings.

Following this introduction, Section 2 gives a brief overview of basics and related work. Details of system model are presented in Section 3. In Section 4**,** we present an overview of the existing algorithms. In Section 5**,** we describe our energy-efficient algorithm in detail. Experimental results are presented in Section 6. Finally, we conclude in Section 7.

2. Basics and related work

Operating system is responsible for functioning of the entire system, including task constraints and status, resource usage, etc. Therefore, it is one of the most effective and efficient approaches to reduce energy consumption with proper task scheduling algorithms.

Voltage scaling has been widely acknowledged as a powerful and feasible technique for trading off energy consumption for execution time. There is a rich literature addressing variable-voltage scheduling for a set of independent tasks with hard deadlines on a single processor [6]. Hardware–software co-synthesis with variable-voltage scaling for single-processor core-based systems is considered for independent tasks [7]. An offline algorithm, which generates a minimum-energy preemptive schedule [8] is also proposed for a set of independent tasks. This schedule, is based on an earliest deadline first (EDF) scheme, and tries to achieve a uniform scaling of voltage levels of different tasks. Heuristic is provided for a similar problem as in [8] for fixed-priority static scheduling [9]. An energy-priority heuristic is provided for non preemptive scheduling of independent real-time tasks [10]. An iterative slack-allocation algorithm [11] proposed based on the Lagrange Multiplier method, points out that variations in power consumption among tasks invalidate the conclusion in [8] that a uniform scaling is optimal. Though most of the works on energy efficient real-time scheduling concentrate on independent tasks over uniprocessor, research is also now being focused on dependent tasks over distributed systems.

Real-time scheduling for tasks with precedence relationships on distributed systems has been addressed in [12]–[15]. There is also work addressing variable-voltage scaling for such systems [16]–[19], [20]. Hybrid search strategies are used for dynamic voltage scaling [16]. It uses a genetic algorithm with simulated healing for global search and hill climbing and Monte Carlo techniques for local search. The energy efficient real-time scheduling problem is formulated as a linear programming (LP) problem [17] for continuous voltage levels, which can be solved in polynomial time. Algorithm based on a list-scheduling heuristic with a special priority function to tradeoff energy reduction for delay is also proposed for distributed systems [18]. The work proposed in [19] uses a genetic algorithm to optimize task assignment, a genetic-list scheduling algorithm to optimize the task execution order, and an iterative slack allocation scheme, which allocates a small time unit to the task that, leads to the most energy reduction in each step. The performance and complexity of this approach are dependent on the size of the time unit, which however, cannot be determined systematically. The usage of a small time unit for task extension can lead to large computational complexity.

For distributed systems, allocation/assignment and scheduling have each been proven to be NP-complete. For variable-voltage scheduling, the problem is more challenging since the supply voltages for executing tasks have to be optimized to maximize power savings. This requires intelligent slack allocation among tasks. The previous work using global/search strategy [16] and integer linear programming (ILP) [17] can lead to large computational complexity. The problem is further complicated when slack allocation has to consider variations in power consumption among different tasks. Among previous works, those in [11], [17], and [19] consider variations in power consumption among different tasks

The Linear Programming (LP) based algorithms proposed in the literature for battery-powered multiprocessor or distributed DVS systems allocate slack greedily to tasks based on decreasing or increasing order of their finish time [20], or allocates evenly to all tasks [21]. These algorithms use integer linear programming based formulations that can minimize the energy [17].

The slack allocation algorithms assume that an assignment of tasks to processors has already been made. Variable amount of slack is allocated to each task so that the total energy is minimized while the deadlines can still be met. The continuous voltage case is formulated as Linear Programming [17] problem where the objective is, minimization of total energy consumption. The constraints include deadline constraints, precedence constraints in original DAG and precedence constraints among tasks on the same processor after assignment.

ISSN (Online): 1694-0784
ISSN (Print): 1694-0814

Since the scheduling algorithm in [17] does not consider the communication time among tasks, it is extended by considering the communication time when representing precedence relationships among tasks in the path based algorithm proposed in [22]. This approach for energy minimization is an iterative approach that allocates a small amount of slack (called unit slack) to a subset of tasks. The selection of tasks for this unit slack allocation is such that the total energy consumption is minimized while the deadline constraint is also met.

The above process is iteratively applied till all the slack is used. These algorithms do not utilize the slack generated dynamically due to the difference in worst case execution time and actual execution time of real-time tasks. So the existing algorithms have been modified in an efficient way, to schedule the dynamic slack that is present between the total execution time and deadline. New modified LP and modified path-based algorithms are proposed, to effectively use the slack so that energy efficiency of the processor increases.

3. System Model

The Directed Acyclic Graph (DAG) represents the flow among dependant tasks in a real-time application. In a DAG a node represents a task and a directed edge between nodes represents the precedence relationship among tasks. The assignment of tasks in a DAG to the available processors is done through scheduling algorithms, so that the finish time of DAG is less than or equal to the application deadline. Here we use the task's execution time at the maximum supply voltage during assignment to guarantee deadline constraints. An assignment DAG represents the direct workflow among tasks after processor assignment. The direct precedence relationships of tasks in an assignment DAG may change from its original DAG.

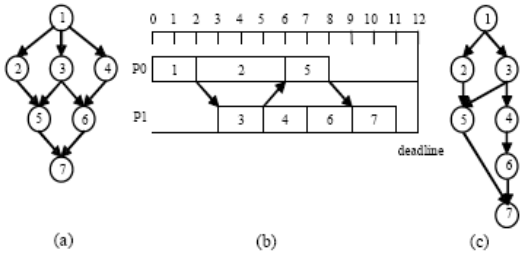


Figure 1 a) DAG b) Assignment on two processor c) Assignment DAG

Figure 1(b) depicts the assignment of tasks for the DAG of Figure 1(a). Figure 1(c) represents the assignment DAG, which is the direct flow among tasks generated after assignment.

The Number of cycles, N_τ, that a task τ needs to finish remains a constant during voltage selection (VS), while processor's cycle time (CT) changes with the supply voltage. For example if processors can operate on a maximum voltage V_h and minimum voltage V_l , we assume CT at V_h is 1 time unit and CT at V_l is 2 time unit. This means slowing down the system increases execution time of tasks with reduced energy.

4. Existing slack allocation algorithms

All slack allocation algorithms allocate variable amount of slack to each task so that the total energy is minimized while the deadlines can still be met. Several slack allocation algorithms are discussed in the literature. Among these, the following algorithms provide close to optimal solutions. Hence, these two algorithms are considered for improvement in this paper.

4.1 LP algorithm

The continuous voltage case is formulated as Linear Programming (LP) [17] problem where the objective is minimization of total energy consumption. The constraints include deadline constraints and precedence constraints in original DAG and assignment DAG.

This is presented as a two-phase framework that integrates task scheduling and voltage selection together to achieve the maximum energy saving of executing dependent tasks on one or multiple variable voltage processors. In the first phase, the EDF scheduling is used for a single processor which is optimal in providing slowdown opportunities and the priority based scheduling is used for multiple processors that provide more slowing down opportunities than a baseline scheduling. The LP formulation in the second phase is the first exact algorithm for the voltage selection (VS) problem. The LP can be solved optimally in polynomial-time for the continuous voltage or solved efficiently for general discrete voltages. This algorithm compared to the baseline scheduling along with scaling VS technique provides 14% more slowing down opportunities. But it does not consider the communication time among tasks, and hence extended as modified LP algorithm for discrete voltage case with resource reclaiming by considering the communication time in assignment DAG.

ISSN (Online): 1694-0784
ISSN (Print): 1694-0814

4.2 Path based algorithm

The path based algorithm [22] presented for continuous voltage parallel and distributed system, is not only considerably better than simplistic schemes but also comparable to LP based algorithm, which provides near optimal solutions.

The LP algorithm ignores the variable energy profiles of tasks on different processors during slack allocation and lead to poor energy reduction. Usage of this variable energy profile can lead to more reduction in energy requirements [19], [21]. However, because of the dependency relationships among tasks in an assignment, the sum of energy reduction of several tasks executed in parallel may be higher than the highest energy reduction of a single task. The path based algorithm effectively addresses this issue and determines a set of multiple independent tasks that together have the maximum energy reduction. In Figure 1, the slack of time 5 to 6 is considered for the slack allocation from start time to total finish time. The slack can be allocated only to task $\tau 2$. However, the slack of time 8 to 9 at Phase 2 can be allocated to a subset of tasks (e.g., $\tau 1$, $\tau 2$ & $\tau 3$, or $\tau 4$). The execution of Phase1 precedes the execution of Phase2 to expect more energy saving by reducing the possibility of redundant slack allocation to the same tasks.

This algorithm compared to LP based algorithm, provides near optimal solutions and also improves energy reduction by 9-29 %.

5. Proposed Algorithms

Both the proposed algorithms integrate task scheduling and voltage selection together and achieve maximum energy saving on distributed system comprised of variable voltage processors, with discrete voltage case. The proposed algorithms are improved over the existing ones by considering new slacks generated due to AET of tasks at runtime. This is achieved through resource reclaiming procedure added to the existing algorithms.

5.1. Modified LP Algorithm

This algorithm integrates task scheduling and voltage selection together to minimize energy consumption of dependent tasks on systems with a given number of variable voltage processors. The algorithm should also guarantee that after the VS technique is applied, tasks with deadlines still finish before their deadlines.

If d_τ is the task's delay , T_τ execution time of task τ at highest voltage V_h and S_τ is the tasks start time, then the timing constraints for multiprocessors can be modeled as

$S_\tau + d_\tau \leq dl_\tau \; \forall \tau$ with deadline

$d_\tau \geq T_\tau$, $S_\tau \geq 0$, int, $\forall \tau$

For a feasible scheduling, the above constraints guarantee that tasks with deadlines will finish before their deadlines.

The objective of VS is to minimize the sum of each task τ's energy consumption by slowing down each task τ without violating timing constraints. To trade the increase of delay for energy saving, a relationship should be established between d_τ and E_τ. In the continuous voltage case, E_τ is a convex function of d_τ [17].

In the discrete voltage case, only a certain number of voltages are available. This implies that a task's delay can only take discrete values. In this case the d_τ and E_τ are computed as linear function of $N_{\tau,i}$, where $N_{\tau,i}$ is the number of cycles that task τ is executed at a discrete voltage $V_i < V_h$.

The task scheduling policy used in this algorithm provides the maximum slowing down potentials for voltage selection to utilize energy saving. To provide more energy saving opportunities in the scheduling for VS to utilize, we use a priority based on task's deadline, dependencies and the usage of processors in the system. The latest finish time (LFT) for a task is assigned based on their deadline or task's successors to meet their deadline. Leaf task's LFT is its deadline or overall DAG's deadline. Starting from leaf tasks and traversing the DAG backward, we can assign each task a latest finish time. Task τ latest finish time LFT_τ is defined as

$$LFT_\tau = \min (dl_\tau, \min (LFT_{s\tau} - T_{s\tau})),$$

where $LFT_{s\tau}$ and $T_{s\tau}$ are the latest finish time and execution time of direct successor tasks of task τ in the DAG respectively. Let us denote task τ's ready time when all its predecessors finish as R_τ, its earliest start time when it is ready and there is a processor available as EST_τ, and i^{th} processor's available time as PAT_i. These values are updated during the scheduling as we build the scheduling step by step. At each step, task τ's priority PR_τ, is evaluated as

$$PR_\tau = LFT_\tau + EST_\tau$$

$EST_\tau = \max (R_\tau, \min (PAT_i))$, and i= 1 to N (N is Number of processors in the system).

A task with smallest priority value is assigned to a processor Pi, if Pi is available at the same time when task is ready. If more than one processor is readily available before the task is ready then select the latest among all processors available when the task becomes ready. If no processor is available, when task is ready select the

ISSN (Online): 1694-0784
ISSN (Print): 1694-0814

processor that is available the earliest. This selection and assignment of tasks repeat until all tasks are scheduled.

The static initial scheduling of tasks is based on the worst case execution time of the tasks. But, it is possible that a task executes less than its worst case computation time, because of data-dependent loops and conditional statements in the task code or architectural features of the system, such as cache hits and branch predictions or both. Hence the modified LP algorithm is improved over the basic LP algorithm to provide resource reclaiming at program runtime for scheduler to adjust the schedule online according to the actual execution time (AET). This newly generated slack time thus can be better utilized for more energy savings. The algorithm is given below.

```
Modified LP Algorithm()
begin
  Order the tasks in non decreasing order of priority in the
  task queue.
  For processors P=1 to N do
                {N is the total number of processors}
  begin
       Get the first task τf from task queue.
       If τf can be executed then
            start execution.
       While task queue is not empty do
       begin
           If a task is under execution on P then
           begin
                Wait for its completion
                Invoke resource reclaiming
                Get next task from queue
                Apply VS for energy saving
           end
       end
  end
end
```

5.2. Modified path based algorithm

This algorithm is developed for discrete voltage based slack allocation for DAG's on distributed system. First tasks are assigned to processors based on priority scheduling discussed in modified LP algorithm, resulting in an assignment DAG. Modified path based slack allocation algorithm applied to it for energy minimization. This is an iterative approach that allocates a small amount of slack (called unit slack) in each iteration to a subset of suitable tasks so that the total energy consumption is minimized while the deadline constraint is also met.

The above process is iteratively applied till all the slacks are used. The purpose of each iteration to find a weighted maximal independent set of tasks, where the weight is given by the amount of energy reduction by allocating unit slack. The dependency relationships in an assignment DAG [23] contains total slack which can be allocated to the different tasks. For instance, in Figure 1, consider an example in which one unit of slack can be allocated. The total tasks that can be allocated for one unit of slack is one or two:

If task τ_7 (or τ_1) is allocated the slack; no other task can use this slack to meet the deadline. Tasks τ_2 and τ_3 (or τ_2 & τ_4, τ_2& τ_6, τ_4 & τ_5, τ_5 & τ_6) can use this slack concurrently as they are not dependent on each other and both can be slowed down. The appropriate option to choose between the two choices depends on the energy reduction in task τ_7 versus the sum of energy reduction for tasks τ_2 and τ_3. This slack allocation algorithm considers the overall assignment-based task dependency relationship [23], while the most other algorithms ignore them. This algorithm has two phases.

In phase1 slack available from start time to total finish time based on a given assignment is considered for maximum energy reduction by delaying appropriate tasks. In this case the slack can be allocated to only a subset of tasks that are not on the critical path. In phase2 slack available from total finish time to deadline is considered for further energy reduction. In this case the slack can potentially be allocated to all the tasks.

For each of the two phases, this algorithm iteratively allocates one unit of slack, called *unitSlack*. The size of the *unitSlack* can be reduced to a level where further reducing it does not significantly improve the energy requirements. For Phase1, at each iteration only tasks with the maximum available slack are considered since the number of slack allocable tasks is limited and the amount of available slack for each task is different. The maximum available slack of task τ is $slack_\tau$, and is defined as

$slack_\tau = LST_\tau - EST_\tau$ and

$LST_\tau = LFT_\tau - T_\tau$, where LST_τ is the latest start time of task τ.

The set of tasks which can share *unitslack* simultaneously is found using compatible task matrix. If task τ_i and task τ_j are in the same assignment-based path, elements m_{ij} and m_{ji} in compatible matrix M are set to zero. Otherwise the elements are set to one. This matrix can be easily generated by performing a transitive closure on the assignment DAG and then taking compliment of that matrix. Here set of tasks considered at each iteration may be changed. This process is iteratively executed till there is no task which can use slack until total finish time.

Meanwhile, at Phase2, all tasks are considered for slack allocation at each iteration. The number of iterations at Phase2 is equal to *totalSlack* divided by *unitSlack*, where

ISSN (Online): 1694-0784
ISSN (Print): 1694-0814

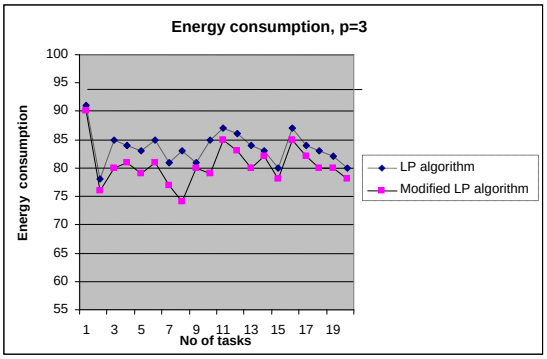
totalSlack is the slack available between actual deadline and total finish time. At each iteration, one *unitSlack* is allocated to one or more tasks that lead to maximum energy reduction.
The energy reduction of a task is defined by the difference between its original energy and its energy expected after allocating a *unitslack* to the task. A branch and bound algorithm is used to search all the compatible solutions to determine the one that has maximum energy reduction. The complete modified path based algorithm is stated informally as below.

Modified Path based Algorithm()
begin
Assign tasks to N processors using Priority based scheduling and obtain assignment DAG.
While unitslack exists in phase1 *do*
begin
Find tasks with maximum available slack
Generate compatible matrix to find slack sharable tasks
Find task set with maximum energy reduction using branch and bound search
Invoke resource reclaiming
end
While unitslack exists in phase2 *do*
begin
Delay all last tasks on processors to extend up to deadline
Invoke resource reclaiming
end
end

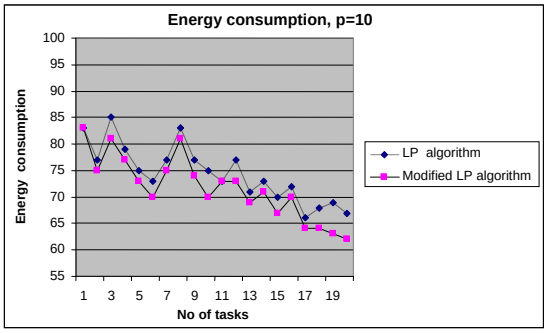
6. Results and Discussions

To evaluate the efficiency of our algorithms in energy saving, we conducted experiments through simulation on various task graphs consisting of 2 to 20 tasks, generated randomly. The comparison of energy consumption for the above task graphs between the existing LP and modified LP algorithms, as well as existing path based and modified path based algorithms are analyzed. For each set of tasks the number of processors is kept constant and the energy consumption for a minimum of ten DAGs are noted. The average values of all those DAGs were calculated. Each point in the above graphs is the average value of such DAGs. This method was repeated by changing the number of processors (varied between 2 to 10) and comparisons were made between the existing and proposed algorithms. A few sample results of those comparisons are shown in the graphs below.

From the graphs it is inferred that path based algorithms provide more energy savings than LP based algorithms. The modified algorithms outperform existing ones by reducing on an average 5% more energy. This is due to the addition of resource reclaiming technique in the proposed algorithms.

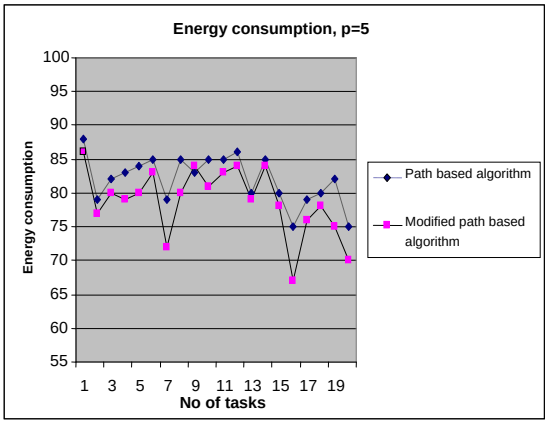


a) Number of Processors = 3



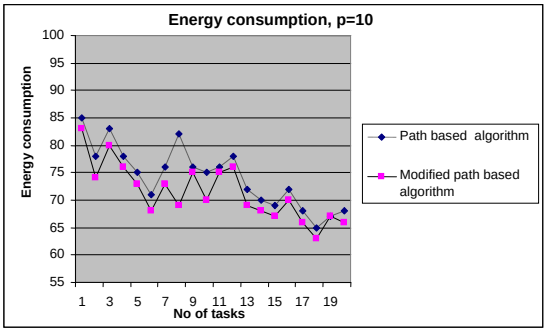
b) Number of Processors = 10

Figure 2. Comparison of energy consumption(in %) between LP and modified LP algorithms



a) Number of Processors = 5

ISSN (Online): 1694-0784
ISSN (Print): 1694-0814



b) Number of Processors = 10

Figure 3. Comparison of energy consumption (in %) between path based and modified path based algorithms

Table 1: Comparisons of overall energy consumption for all four algorithms

Algorithm	Average Energy Consumption (%)
LP	85
Modified LP	79
Path based	82
Modified Path based	77

From Table 1, it is noted that in existing algorithms, Path based algorithm consumes less energy than LP algorithm. This is due to the concept of unit slack introduced. It can also be observed from the results that our proposed algorithms have less energy consumption than the existing ones by 5%, thus leading to more energy efficiency.

7. Conclusions

In this paper, two modified DVS algorithms for discrete voltage case and distributed systems are presented. These algorithms handle task graphs with precedence and timing constraints along with communication time. The resource reclaiming component is used to improve energy reduction over the existing LP and path based optimal slack allocation algorithms. Experimental results show that the proposed algorithms are comparable to exiting algorithms in providing optimal solutions as well as consume energy 5% less than the than the existing ones. The task graphs considered in this paper are assumed to be free of resource constraints which arise due to the use of non-sharable resources by the tasks. So this algorithm can be extended by considering resource constraints in addition to the precedence and timing constraints for dependant tasks.

References

[1] T. D. Burd, T. Pering, A. Stratakos, and R. Brodersen, A dynamic voltage scaled microprocessor system", IEEE J. Solid-State Circuits, Vol. 35, pp. 1571{1580, 2000.

[2] T. Burd and R. Brodersen. Energy Efficient CMOS Microprocessor Design. In Proceedings of 28th Annual Hawaii International Conference on System Sciences., pages 288–297, January 1995.

[3] *Intel Xscale*. [Online]. Available: http://www.intel.com/design/intelxscale/

[4] *Transmeta Crusoe*. [Online]. Available: http://www.transmeta.com

[5] *AMD PowerNow!* [Online]. Available: http://www.amd.com

[6] N. K. Jha, "Low power system scheduling and synthesis," in *Proc. Int. Conf. Comput.-Aided Des.*, Nov. 2001, pp. 259–263.

[7] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M. B. Srivastava, "Power optimization of variable-voltage core-based systems," *IEEE Trans. Comput.-Aided Design Integer Circuits Syst.*, vol. 18, no. 12, pp. 1702–1714, Dec. 1999.

[8] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced CPU energy," in *Proc. Symp. Foundations Comput. Sci.*, Oct. 1995, pp. 374–382.

[9] P. B. Jorgensen and J. Madsen, "Critical path driven co synthesis for heterogeneous target architectures," in *Proc. Int. Workshop Hardware/ Software Code.*, Mar. 1997, pp. 15–19.

[10] J. Pouwelse, K. Langendoen, and H. Sips, "Energy priority scheduling for variable voltage processors," in *Proc. Int. Symp. Low-Power Electron. and Des.*, Aug. 2001, pp. 26–31.

[11] A. Manzak and C. Chakrabarti, "Variable voltage task scheduling algorithms for minimizing energy," in *Proc. Int. Symp. Low Power Electron. and Des.*, Aug. 2001, pp. 279–282.

[12] Y. Kwok and I. Ahmad, "Dynamic critical-path scheduling: An effective technique for allocating task graphs to multiprocessors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 7, no. 5, pp. 506–521, May 1996.

[13] P. Eles, A. Doboli, P. Pop, and Z. Peng, "Scheduling with bus access optimization for distributed embedded systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 8, no. 5, pp. 472–491, Oct. 2000.

[14] G. Sih and E. A. Lee, "A compile-time scheduling heuristic for interconnection constrained heterogeneous processor architectures," *IEEE Trans. Parallel Distrib. Syst.*, vol. 4, no. 2, pp. 175–187, Feb. 1993.

[15] J. Liu, P. H. Chou, and N. Bagherzadeh, "Communication speed selection for embedded systems with networked voltage-scalable processors," in *Proc. Int. Workshop Hardware/Software Code.*, May 2002, pp. 169–174.

[16] N. Bambha, S. S. Bhattacharyya, J. Teich, and E. Zitzler, "Hybrid search strategies for dynamic voltage scaling in embedded multiprocessors," in *Proc. Int. Workshop Hardware/Software Co-Design*, Apr. 2001, pp. 243–248.

[17] Y. Zhang, X. Hu, and D. Chen, "Task scheduling and voltage selection for energy minimization," in *Proc. Des. Autom. Conf.*, Jun. 2002, pp. 183–188.

[18] F. Gruian and K. Kuchcinski, "LEneS: Task-scheduling for low-energy systems using variable voltage processors," in

Proc. Asian South Pacific Des. Autom. Conf., Jan. 2001, pp. 449–455.

[19] M. T. Schmitz and B. M. Al-Hashimi, "Considering power variations of DVS processing elements for energy minimization in distributed systems," in *Proc. Int. Symp. Syst. Synthesis*, Oct. 2001, pp. 250–255.

[20] P. Chowdhury and C. Chakrabarti, "Static task scheduling algorithms for battery-powered DVS systems", IEEE Trans. On Very large Scale Integration Systems, 13(2), pp. 226-237, Feb. 2005.

[21] J. Luo and N. K. Jha, "Power-conscious joint scheduling of Periodic task graphs and aperiodic tasks in distributed real-time embedded systems", Int. Conf. on Computer-Aided Design, pp. 357-364, Nov. 2000.

[22] Jaeyeon Kang and Sanjay Ranka , "Dynamic Algorithms for Energy Minimization on Parallel Machines", 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing , pp. 399-406, 2008.

[23] Youlin Ruan, Gan Liu, Qinghua Li and Tingyao Jiang, " An Efficient Scheduling Algorithm for Dependent Tasks", Int. Conference on Computer and Information Technology, pp.456-459, May. 2004.

Mrs. Santhi Baskaran received her B.E. degree in Computer Science and Engineering from University of Madras, Chennai, India in 1989 and M.Tech. degree in Computer Science and Engineering from Pondicherry University, Puducherry, India in 1998. She served as Senior Lecturer and Head of the Computer Technology Department, in the Polytechnic Colleges, Puducherry. India, for eleven years, since 1989. She joined Pondicherry Engineering College, Puducherry, India in 2000 and currently working as Associate Professor in the Department of information Technology. Now she is pursuing her PhD degree in Computer Science and Engineering. Her areas of interest include Real-time systems, embedded systems and operating systems. She has published research papers in International and National Conferences. She is a Life member of Indian Society for Technical Education and Computer Society of India.

Prof. Dr. P. Thambidurai is a Member of IEEE Computer Society. He received his PhD degree in Computer science from the Alagappa University, Karaikudi, India in 1995. From 1999, he served as Professor and Head of the Department of Computer Science & Engineering and Information Technology, Pondicherry Engineering College, Puducherry, India, till August 2006. Now he is the Principal for Perunthalaivar Kamarajar Institute of Engineering and Technology (PKIET) an Government institute at Karaikal, India. His areas of interest include Natural Language Processing, Data Compression and Real-time systems. He has published over 50 research papers in International Journals and Conferences. He is a Fellow of Institution of Engineers (India). He is a Life member of Indian Society for Technical Education and Computer Society of India. He served as Chairman of Computer Society of India, Pondicherry Chapter for two years. Prof. P.Thambidurai is serving as an Expert member to All India Council for Technical Education (AICTE) and an Adviser to Union Public Service Commission (UPSC), Govt. of India. He is also an Expert Member of IT Task Force and Implementation of e-Governance in the UT of Puducherry.