Middle-East Journal of Scientific Research 18 (6): 779-787, 2013
ISSN 1990-9233

DOI: 10.5829/idosi.mejsr.2013.18.6.12487

EC-A: A Task Allocation Algorithm for Energy Minimization in Multiprocessor Systems

Anju S. Pillai and T.B. Isha

Department of Electrical and Electronics Engineering,
Amrita Vishwa Vidyapeetham, Coimbatore, India

Abstract: With the necessity of increased processing capacity with less energy consumption; power aware multiprocessor system has gained more attention in the recent future. One of the additional challenges that is to be solved in a multi-processor system when compared to uni-processor system is job allocation. This paper presents a novel task dependent job allocation algorithm: Energy centric- Allocation (Ec-A) and Rate Monotonic (RM) scheduling to minimize energy consumption in a multiprocessor system. A simulation analysis is carried out to verify the performance increase with reduction in energy consumption and required number of processors in the system.

Key words: Energy Consumption · Job allocation · Multiprocessor systems · Task dependent

INTRODUCTION

Multi processor systems are preferred over uni-processor systems due to a number of reasons viz. increased performance in terms of increased throughput, reduced time to complete the jobs, increased reliability and dependability. Multiprocessor system may be either homogeneous or heterogeneous, based on the application for which it is designed. Even though there are many advantages using multiprocessor systems, the implementation of such systems becomes difficult due to the complexity and difficulty in inter-processor communication and synchronization. The additional overhead that is to be addressed for a multiprocessor system when compared to a uni-processor system includes: task-processor allocation, task communication and synchronization, increased complexity and cost. Task assignment is the process of partitioning the available system of tasks and resources into modules and assignment of these modules into the available processors. Normally task assignment is carried out offline. Because, if the system is static and the complete knowledge of the application is known a priori, then after identifying different modules, the tasks are assigned and bound to processors in the system. Normally, finding an optimal assignment is impractical and is NP-hard [1]. For multiprocessor systems, there are available different algorithms for task-allocation namely: Next-fit algorithm, Bin-packing algorithm, Utilization balancing algorithms, Myopic offline scheduling algorithm, Focused, Buddy strategy and Assignment with precedence constraints etc. [2]. Once the tasks are assigned to the processor, they need to be scheduled. The traditional algorithms which are used for scheduling include: Rate Monotonic (RM) scheduling, Earliest Deadline First (EDF) policy, Deadline Monotonic (DM) etc. [3, 4].

The major issue of most of portable embedded devices is the amount of duration for which battery charge can be retained. This is a major concern as it affects the system usefulness. Energy issues are not only critical for portable devices, but also for standalone systems too. Because this induces problems associated with cooling, power delivery and result in high power densities. Most of today's processors support either chip-multiprocessor or chip-threading techniques and is very popular in the market. Due to the wide spread use of multiprocessor systems, these systems can now be found in desktop applications and in small embedded devices [5]. While ensuring the computational needs, energy consumption issues need more attention to make it a best solution for all sorts of applications. Therefore, most of today's systems are power aware systems.

Corresponding Author: Anju S. Pillai, Department of Electrical and Electronics Engineering,
Amrita Vishwa Vidyapeetham, Coimbatore, India.

Power consumption reduction can be attempted at different levels and by different methods. Software, hardware and power-aware embedded solutions are popular in the field with significant contributions by the software approaches [6]. From the literature, it can be found that, there are plenty of power aware algorithms and protocols available to reduce the power consumption in both uniprocessor and multiprocessor systems [7]. Most of the works carried out for the uniprocessor system are implementable for multiprocessors as well. But, the additional complexities of multiprocessors like: task allocation, inter-processor communication and synchronization have to be exclusively solved for such systems. In this paper, an algorithm for job allocation to minimize the energy consumption of the system by grouping the tasks based on the dependencies is presented.

For power consumption reduction, there are available a good deal of algorithms exploiting Dynamic Voltage Scaling (DVS) techniques. The possible power reductions are by utilizing the processor slack to slowdown the tasks by reducing the supply voltage and clock frequency [8], [9]. In the field, DVS techniques still stands as one of the prime method for energy saving.

In this paper, a task allocation algorithm: *Ec-A* based on task dependencies and precedence constraints are presented to minimize the system energy consumption. Here a new task grouping and allocation strategy is presented which is beneficial for power consumption reduction and number of processors to execute the given set of tasks. The major contributions of the work include:

- New task allocation algorithm for energy minimization – *Energy centric- Allocation (Ec-A) Algorithm*
- Implementation of task allocation and scheduling in power aware multiprocessor system

Related Workdone: The field of real time systems is enriched with sufficient algorithms and protocols solving most of the existing problems. As the computing needs vary from time to time and application to application; more and more features needs to be incorporated into the existing embedded devices to meet the current performance requirements. Due to this, there is a shift from uni-processor model to multi-core and multiprocessor systems. In the field, there are available many algorithms for task allocation and scheduling. The different algorithms vary in terms of the cost which is minimized while performing the assignment. The popular task assignment algorithms are: Utilization balancing algorithm, Next-Fit algorithm for RM, Bin packing algorithm for EDF, Buddy algorithm, Myopic algorithm etc. The simulation analysis and validation of each of these algorithms are carried out clearly in the literature [2].

The popular and widely used scheduling algorithms are RM, DM and EDF which are actually optimal only for uniprocessor systems. Although EDF is not optimal for multiprocessors, it still remains as a good algorithm for scheduling tasks in multiprocessors. Some of the reasons favoring EDF is, its work conserving nature and bounded preemptions and inter-processor task migrations [10].

There are available a class of algorithms named: *fair algorithms* introduced by Baruah et al. which are found optimal for multiprocessor scheduling. These algorithms are used to guarantee quality of service in computing systems and the first quantum based optimal global scheduling algorithm for multiprocessor real time systems was proposed: Proportionate fair (Pfair) algorithm [11]. Tasks are separated based on the utilization as light (Utilization $\leq$ 50%) and heavy (utilization > 50%) and another algorithm named: PD was also proposed [12]. A simpler algorithm proposed is PD^2 [13] which use less parameter to compare the priorities of the tasks when compared to PD. A variant of Pfair scheduling, early-release scheduling, was also proposed in [14]. But, the above algorithms considered periodic tasks for scheduling. In [15], sporadic tasks and intra-sporadic tasks were introduced in the system. Anyway, the scheduling overheads in Pfair algorithms were quite high and so, another algorithm: Boundary fair (Bfair) is implemented in [16] where the number of time instants at which the scheduling decisions are made was considerably reduced. This algorithm reduces the number of context switches and task migrations and run time overhead during scheduling the tasks. The task scheduling in the multiprocessor system is found to be NP-hard [1]. In order to optimize such systems for different cost functions, certain heuristic based methods have been used rather than conventional techniques. Conventional techniques take reasonable amount of time when compared to heuristic approach.

The need for increased performance of the system is met with architectural supports like multicore and multiprocessors. Additionally, software techniques like multi threading, parallel processing, pipelining, exploiting parallelism at instruction and data levels etc. are also incorporated in today's processors to increase the performance and throughput. To meet the performance needs in such Multi Processors on Chip (MPoC) devices; the energy issues are to be much stringent. If the field of

real time systems is explored, a variety of techniques for energy consumption reduction attempted in all possible ways can be found. Out of these techniques, energy aware task scheduling is one of the prime. A reliability aware energy management is proposed in [17]. Also, energy minimization is attempted by life time reliability aware scheduling in [18]. In one of the recent papers, a joint effort for reducing communication cost, throughput degradation and migration overhead is attempted [19].

The literature supports a few task allocation algorithms in multiprocessor systems which intend to minimize the energy consumption of the system. ETAHM [20] is one such allocation algorithm, which combines task scheduling, mapping, DVS utilization and ant colony optimization while allocating jobs to heterogeneous multiprocessors to reduce the energy consumption. To reduce the energy required to complete a given work load, a robust static resource allocation heuristic was proposed in [21]. The energy consumption reduction is implemented using DVS technique in this method also. ERTJA algorithm is used in sensor networks to reduce communication energy and problems associated with temporal characteristics of the nodes. Energy consumption reduction of the cluster is carried out by reducing the activation of the sleeping nodes [22]. A time and energy aware starvation free allocation algorithm: SCATE for wireless sensor nodes are presented in [23]. An energy aware task allocation for RM scheduling was proposed by Tarek *et al*. [24], which include RM admission control, partition heuristics and speed adjustment to conserve the energy. A method to calculate optimal work load assignment at run time by scalable implementation is shown in [25].

This paper proposes a job allocation for multiprocessor applications, whose primary objective is to minimize the energy consumption of the system while allocating jobs to the processors. For the implementation, a novel task assignment algorithm: *Ec-A* is developed and RM scheduling policy is used for task scheduling.

System Model

Problem Statement: For a set Γ with n independent and periodic tasks and P homogeneous processors in the system, the goal is to implement a task allocation and scheduling in multi-processor system for energy minimization subject to the condition:

- The total utilization of the task set is less than or equal to available processor capacity.
- Individual processor assignment is done without violating the schedulability by RM policy.

Solution: The solution steps include details about the task-processor assignment algorithm and scheduling policy. These details are described in the following sub sections.

Task-Processor Allocation Algorithm: Once the number of tasks and the task attributes are known, tasks are to be assigned to an optimal number of processors for scheduling. The main objective of the work is to minimize the power consumption of the embedded multiprocessor system during task allocation. For this, an appropriate task assignment strategy is devised. The functioning of the proposed new task allocation algorithm and the significance of task allocation based on dependency and precedence constraints are described in detail in the following subsection.

E-Centric *Allocation* Algorithm (*Ec-A* Algorithm): Various task allocation algorithms are available in the literature. Some of these algorithms aim for energy consumption reduction too. Here, a new task allocation strategy for multiprocessor systems is presented whose primary objective is to group the tasks based on the dependency and allocating those dependent tasks on the same processor so as to reduce the communication cost and using only the minimum peripherals as possible, to conserve energy. Due to the dependent nature and task precedence relations, a task grouping strategy is formed which is beneficial for energy consumption reduction. In the proposed *Ec-A* algorithm, once the required peripherals and ports of the processor/controller are identified, the clock inputs to the other peripherals are cut down to ensure protection against associated energy wastage. The use of this algorithm for energy and processor reduction can be better realized when used for actual hardware implementation.

In the field of multiprocessor systems, there are two techniques which are widely used; one is *global scheduling* and the other one is *partitioned scheduling*. In global scheduling, only one main queue is maintained where each task is present and ordered according to some priority assignment policy. In normal approach, from the main queue, few highest priorities, say p tasks are assigned to q processors and so on until all tasks are assigned to the available number of processors. In this approach, the overhead of task admission control and ordering is limited to one main queue alone. But, in partitioned approach, the tasks are assigned to processors as it arrives. Tasks remain in the assigned processor only, no task migration is allowed thereafter. Once the tasks are assigned to different processors, tasks

ordering and scheduling is done by the individual queue available in each processor. From the literature it can be found that it is inappropriate to compare the two different approaches, but due to the simplicity in managing the queue at processor level, partitioned approach is more widely used [26].

The proposed *Ec-A* algorithm uses the global scheduling approach for task allocation and the main queue is created by considering the task dependence relations. Task dependencies can be due to:

- *Data dependency* – The result of a task execution is the input to the other task.
- *Resource requirement* – Two tasks τ_i and τ_j need the same peripherals like: PORTS, LCD display, printers etc. for the execution. In such cases, based on the precedence relations, tasks can be assigned to the same processor for scheduling.
- *Nature of application* – Based on application nature, tasks may depend on each other.
- *Control/conditional dependency* – A task execution may be activated only when a particular condition is TRUE in another task. Eg. for a sensor task, if the input temperature goes above a particular threshold, activate a Buzzer task for warning.

Also, task precedence relations are considered to order and group the tasks in the main queue. Assigning dependent tasks to the same processor is advantageous due to:

- *Use of resource constrained processors to reduce the system cost* - Processors with limited resources like peripherals and other hardware devices can be appropriately used. If a particular device is available only in one of the processor, then the tasks which require those peripherals can be assigned to that processor. All the resources need not be present in all processors in the system.
- *Reduction in inter-processor communication* – If dependent tasks are assigned to different processors, those tasks need to communicate across the processors for sharing data, information regarding execution completion or condition checking etc. This external communications can be avoided if these tasks are assigned to the same processor.
- *Synchronization at task level and processor level* – If tasks are dependent on each other, they need to be synchronized. Otherwise, it may result in inconsistency. Thus, if dependent tasks are allocated to different processors, synchronization between the tasks becomes hard to achieve. In addition to task synchronization, in such cases processor synchronization also becomes essential. By assigning the dependent tasks to the same processor, this overhead can be eliminated.
- *Reduction in time delay* – The delay in communication and sharing of data can be avoided if dependent tasks are assigned to the same processor.
- *Creation of feasible schedule* – A feasible schedule can be created when tasks precedence and dependency relations are known a priori and the tasks are available in the same processor.

In addition to the above mentioned advantages, dependent task grouping and allocation to the same processor is beneficial in terms of energy consumption reduction. Energy is consumed:

- When all the peripherals and hardware devices are present in all the processors, even though those devices are not used by the application task, there is some energy wastage associated, due to the availability of system clock input to the peripherals.
- Due to the need of task communication and synchronization for sharing data and control signals, across the processors, the processor utilization increases due to the extra time processor expends in waiting for an acknowledgement from the other processor which is executing the dependent task. The increased utilization means the processor is not idle but busy doing some work. This will increase the energy consumption of the processor.
- Also, increased time delay will add up to increased power consumption.

Thus, assigning dependent tasks to the same processor is vital for energy saving in multiprocessor systems. Energy saving in the order of milli Joules (mJ) is also significant, because when it comes to the implementation of large and complex applications like sensor networks or smart grids; where thousands of such processors are executing throughout the time, the energy saving can be significant.

The functioning of the proposed algorithm involves two stages.

Task Dependency Identification and Queue Creation: The tasks are arranged in the main queue by considering the task dependencies. If a task τ_is operation depends on τ_j task then, tasks (τ_i, τ_j) are grouped together to assign to

the same processor. In this phase, in addition to task dependency identification and grouping, the total number of processors required in the system is also calculated and fixed. By knowing the total task set utilization, the number of required processors is fixed as:

$$U_{total} = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

where: C_i is the execution time of the task and T_i the time period.

The required optimal number of processors, $n = [U_{total}]$

Task Assignment: Upon the identification of task dependencies, the next step is task assignment. Task assignment is carried out satisfying the following conditions:

- After the formation of dependent task groups, the first dependent group d_i is assigned to the first processor P_i. Next dependent group is assigned to the next processor and so on, until all the processors are assigned with one dependent group each.

$$d_i \rightarrow P_i, d_j \rightarrow P_j, \ldots d_m \rightarrow P_m$$

where: d_i is the group of dependent tasks represented by (τ_p, τ_q) After the assignment, find individual processor utilization: U_i, U_j,... .... U_m Then, check whether the feasibility condition of RM policy holds good after the assignment. The sufficient condition for schedulability by RM policy is given by the upper bound condition.

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{1/n} - 1)$$

where: n is the total number of tasks in the system. However, the above utilization test is not optimal as the workload on the processor would be overestimated. The sufficient and necessary condition for schedulability by RM is given by the Response Time Analysis (RTA) [27] as below:

$$R_i^{n+1} = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i^n}{T_j} \right\rceil C_j$$

The response time of each task is iteratively calculated until it converges to a fixed value. Finally, if $R_i \leq D_i$ then the given task set is schedulable by RM priority assignment.

On completion of dependent task assignment, next is the independent task assignment in the system. The independent task with higher utilization is assigned to the processor with lesser utilization. The assignment is continued until all the independent tasks are assigned to the processors or without violating the feasibility condition. In case, if any of the independent task cannot be allocated to the available processors then, task splitting-up is carried out. i.e. independent task is split into two or more smaller functions without affecting the functionality of the task. Each of the smaller functions is assigned to different processors so that the processor can still schedule the dependent and independent tasks without affecting the schedulability. Task splitting strategy helps to pack the processors more effectively and thus result in reduced number of processors when compared to Next-fit or any such algorithms used in the multiprocessor domain.

If task τ_m is an independent task whose utilization is U_m and is not possible to allocate to any of the processors as the schedulability condition fails. Under this condition, task τ_m is portioned into two parts initially as: τ_{m1} and τ_{m2}. These two parts are assigned to the processors P_i and P_j which are the least used processors among all the available processors.

$$\tau_{m1} \rightarrow P_i \text{ and } \tau_{m2} \rightarrow P_j$$

The time complexity of *Ec-A* algorithm for allocating n periodic tasks to the processor is O(n log(n)). The E-cA algorithm steps are shown in Fig. 1.

Scheduling of Tasks: Tasks are scheduled based on the well known fixed priority assignment policy RM. The priority assignment rule is *shorter the time period, higher the priority*. Upon the priority assignment, a task holds the assigned priority until the end of its execution. Also, different instances of the same tasks hold the same priority. After allocating the jobs to the required optimum number of processors in the system, tasks are scheduled on individual processors. Task assignment is done without violating the schedulability of the system.

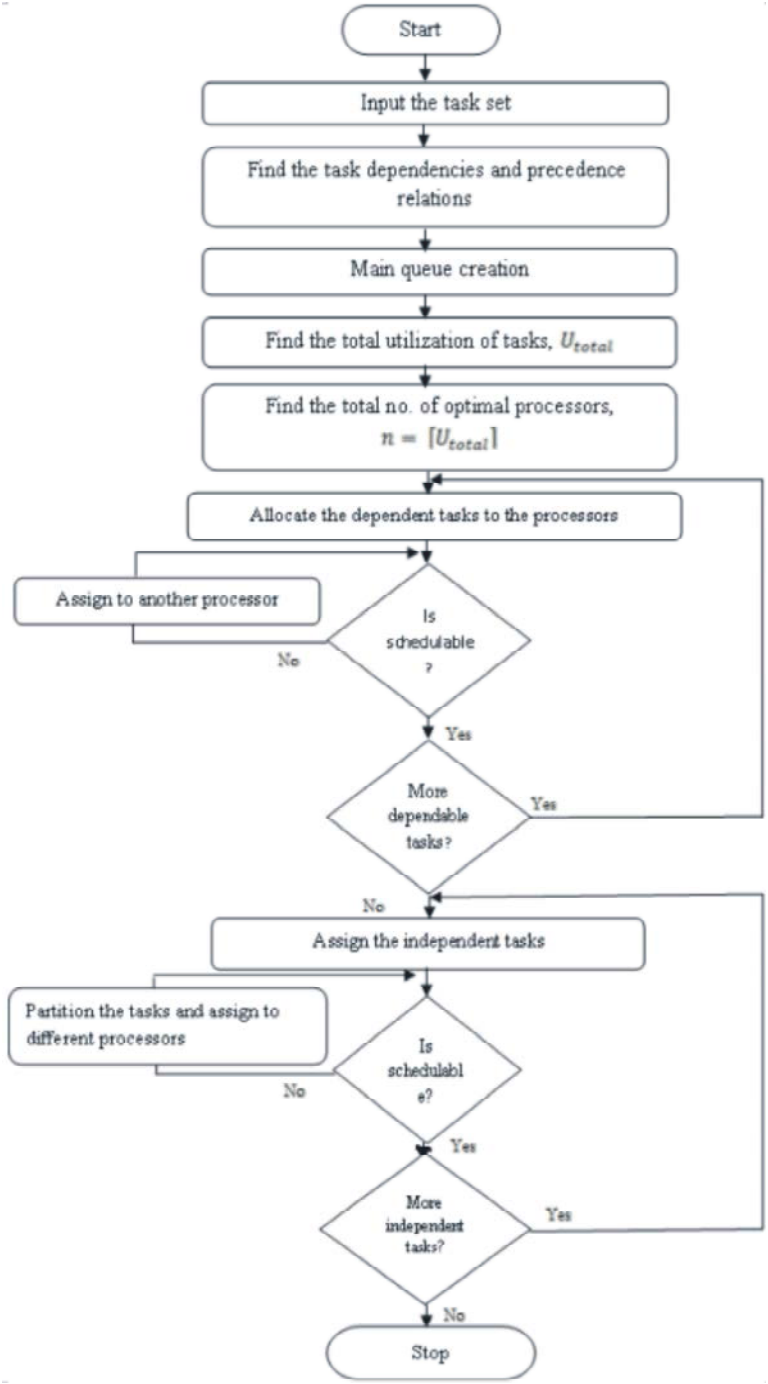

Fig. 1: Flow chart describing the steps of *Ec-A* algorithm

Simulation Results: Performance evaluation is carried out using MATLAB. The simulations are done with synthetic tasks generated by UUnifast algorithm [28]. Task number varying from 10 to 20 with varying utilizations of 2 to 10 is considered for evaluation. For analysis, ARM7 LPC2148 microcontroller is used, whose rated operating voltage is 3.3V and clock frequency 60MHz. The energy consumption of the processor is calculated using the equation:

$$E_{processor} = \sum_{i=1}^{n} C_i * v_i^2 * f_i * t$$

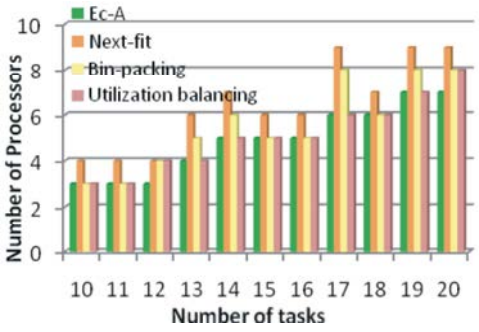

Fig. 2: Variation in no. of processors with no. of tasks

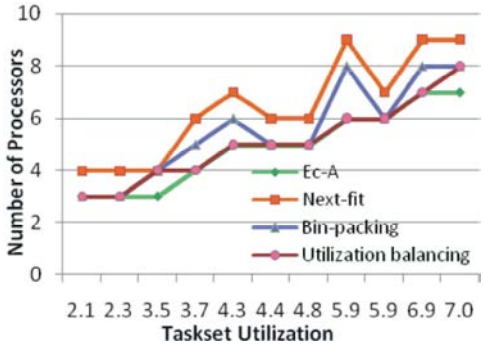

Fig. 3: Variation in no. of processors with task set utiliztion

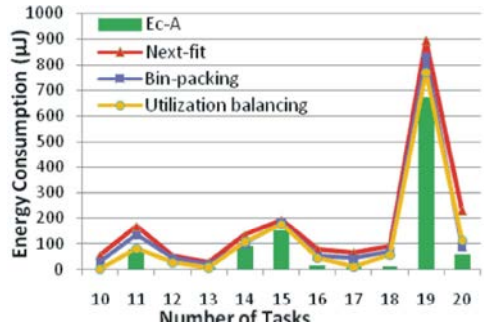

Fig. 4: Number of tasks Vs Energy consumption

where: the effective capacitance, the supply voltage and the clock frequency. *t* is the time period during which the processor is executing the tasks.

The Fig. 2 shows the required number of processors with varying number of tasks.

Fig. 3 shows the variation in the processor requirement with variation in task set utilization and Fig. 4 shows the change in energy consumption of the processor with number of tasks.

When compared to Next-fit allocation algorithm, which is suitable for scheduling tasks with RM policy in multiprocessor systems, the *Ec-A* allocation algorithm takes only O(n log(n)) complexity to assign *n* tasks to the processors. While Next-fit algorithm has a complexity of $O(n^2)$. From the above graphs, it is verified that *Ec-A* algorithm outperforms the other existing task allocation algorithms for multiprocessors, in terms of reduced number of processors and reduced energy consumption and complexity.

CONCLUSION

The present era systems are mostly multiprocessor systems due to the demand of high ended complex application implementation. For such systems, energy consumption is a critical issue due to the resource limitations and need for battery power. An optimal task allocation strategy is vital for obtaining a proper scheduling that can minimize the energy consumption of the system. This paper presents a task allocation algorithm, Energy centric Allocation: *Ec-A*, for energy consumption reduction in multiprocessor systems. The allocation strategy is based on grouping the tasks by the dependencies and precedence relations and by cutting the clock supply to the unused peripherals at all the time when it is not functioning. The simulation results validate the performance enhancement of *Ec-A* algorithm to reduce the number of processors required and also to minimize the energy consumption while scheduling the tasks. The future work intends to implement the *Ec-A* algorithm in a real time application and perform a hardware validation.

REFERENCES

1. Edwin Hou, S.H., Nirwan Ansari and Hong Ren, 1994. A genetic algorithm for multiprocessor scheduling, IEEE Trans. on Parallel and Distributed Systems, 5(2).
2. Jane, W.S. and W. Liu, 2000. Real-Time Systems. Prentice Hall PTR, Upper Saddle River, NJ, USA
3. Liu C.L. and J.W. Layland, 1973. Scheduling algorithms for Multi programming in a hard-real-time environment, Journal of ACM, 20(1): 4661.
4. Buttazzo, G., 2003. Rate monotonic vs. EDF: Judgment day, in Proceedings of 3rd ACM International Conference on Embedded Software, Philadephia, USA.
5. Moreshet Tali, R. Bahar Iris and Herlihy, 2005. Maurice. Energy reduction in multiprocessor systems using transactional memory, ISLPED, editor(s) Roy, Kaushik and Tiwari, Vivek, pp: 331-334.

6. Wei, T., P. Mishra, K. Wu and H. Liang, 2008. Fixed-Priority Allocation and Scheduling for Energy-Efficient Fault Tolerance in Hard Real-Time Multiprocessor Systems, IEEE Transactions on Parallel and Distributed Systems, On Page(s), 19(11): 1511-1526.
7. Sung, I. Park, 2003. The Design of Power Aware Embedded Systems, PhD Thesis, University of California, Los Angeleles.
8. Padmanabhan, Pillai and Kang G. Shin, 2001. Real-Time Dynamic Voltage Scaling for Low Power Embedded Operating Systems, Symposium on Operating Systems Principles'01.
9. Woonseok Kim, Jihong Kim and Sang Lyul-Min, 2003. Dynamic voltage scaling algorithm for fixed-priority real-time systems using work-demand analysis, ISLPED '03. Proceedings of the 2003 International Symposium on Low Power Electronics and Design, pp: 396-401.
10. Goossens, J., S. Funk and S. Baruah, 2003. Priority-driven scheduling of periodic tasks systems on multiprocessors, Real-Time Systems, 25(2-3): 187-205.
11 Baruah, S.K., N.K. Cohen, C.G. Plaxton and D.A. Varel, 1996. Proportionate progress: a notion of fairness in resource allocation, Algorithmica, 15(6): 600-625.
12. Baruah, S.K., J. Gehrke and C.G. Plaxton, 1995. Fast scheduling of periodic tasks on multiple resources, Proc. of The International Parallel Processing Symposium, pp: 280-288.
13. Anderson, J.H. and A. Srinivasan, 2001. Mixed pfair/erfair scheduling of asynchronous periodic tasks, Proc. of the 13th Euromicro Conference on Real-Time Systems, pp: 76-85.
14. Anderson, J.H. and A. Srinivasan, 2000. Early-release fair scheduling, Proc. of the 12th Euromicro Conference on Real-Time Systems, pp: 35-43.
15. Anderson, J.H. and A. Srinivasan, 2000. Pfair scheduling: Beyond periodic task systems, Proc. of the 7th Int'l Workshop on Real-Time Computing Systems and Applications, pp: 297-306.
16. Dakai Zhu, Xuan Qi, Daniel Mosse and Rami Melham, 2011. An optimal boundary fair algorithm scheduling algorithm for multiprocessor real time systems, Journal of parallel and Distributed Computing, pp: 1411-1425.
17. Aydin, H. and D. Zhu, 2011. Reliability-aware energy management for periodic real-time tasks, IEEE Transactions on Computers, 58(10): 1382-1397.
18. Huang, L. and Q. Xu, 2010. Energy-efficient task allocation and scheduling for multimode MPSoCs under lifetime reliability constraint, in: IEEE Conference on Design, Automation and Test in Europe.
19. Das, A., A. Kumar and B. Veeravalli, 2012. Energy-aware communication and remapping of tasks for reliable multimedia multiprocessor systems, in: International Conference on Parallel and Distributed Systems, ICPADS.
20. Chang, Po-Chun, Wu I-Wei, Shann, Jean Jyh-Jiun and Chung Chung-Ping, 2008. ETAHM: an energy-aware task allocation algorithm for heterogeneous multiprocessor, DAC. editor(s) Fix, Limor. 776-779, ACM.
21. Apodaca, Jonathan, Young, Bobby Dalton, Briceno, Luis Diego, Smith, Jay, Pasricha, Sudeep, Maciejewski, Anthony A. Siegel, Howard Jay, Bahirat Shirish, Khemka, Bhavesh, Ramirez, Adrian and Zou Yong, 2011. Stochastically robust static resource allocation for energy minimization with a makespan constraint in a heterogeneous computing environment, AICCSA. editor(s) Siegel, Howard Jay and El-Kadi, Amr., pp: 22-31.
22. Karimi Hamid, Kargahi Mehdi and Yazdani Nasser, 2010. On the Handling of Node Failures: Energy-Efficient Job Allocation Algorithm for Real-time Sensor Networks, In JIPS, 6(3): 413-434.
23. Mohsen Sharifi and Morteza Okhovvat, Scate, 2011. A Scalable Time and Energy Aware Actor Task Allocation Algorithm in Wireless Sensor and Actor Networks, ETRI Journal, 34(3): 330-340. http://dx.doi.org/10.4218/etrij.12.0111.0366.
24. Tarek, A. AlEnawy and Hakan Aydin, 2013. Energy-Aware Task Allocation for Rate Monotonic Scheduling, IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS), pp: 213-223.
25. Paterna, Francesco, Acquaviva Andrea, Caprara, Alberto, Papariello, Francesco, Desoli, Giuseppe and Benini Luca, 2011. An efficient on-line task allocation algorithm for QoS and Energy Efficiency in Multicore Multimedia Platforms, pp: 100-105.

26. Zhu, D., R. Melhem and B. Childers, 2003. Scheduling with dynamic voltage/speed adjustment using slack reclamation in multi-processor real-time systems, IEEE Trans. on Parallel and Distributed Systems, 14(7): 686-700.
27. Joseph, M. and P. Pandya, 1986. Finding response times in a real time system, BCS Computer Journal, 29(5): 390-395.
28. Bini, E.G. and C. Buttazzo, 2004. Biasing Effects in Schedulability Measures, IEEE Proceedings of the 16th Euromicro Conference on Real-Time Systems, Aatania, Italy.