International Journal of Innovation and Scientific Research
ISSN 2351-8014 Vol. 25 No. 1 Jun. 2016, pp. 32-37
© 2015 Innovative Space of Scientific Research Journals
http://www.ijisr.issr-journals.org/

A heuristic Energy aware Real-Time scheduling approach for periodic tasks in homogenous multi-core systems

***K. Vasanthamani*[1] *and P. Visalakshi*[2]**

[1]Department of Electronics & Communication Engineering,
PSG College of Technology,
Coimbatore, Tamilnadu, India

[2]Department of Electronics & Communication Engineering,
PSG College of Technology,
Coimbatore, Tamilnadu, India

ABSTRACT: Energy efficient operation of any portable devices ultimately increases the lifetime of the devices between every battery charging. The multi-core processors are used widely today for increasing the performance of computing. Efficient scheduling of the real-time tasks becomes necessary to effectively utilize all the cores in a multi-core processor. This work focuses on energy efficient task scheduling of the real-time tasks in homogenous multi-core processor environment. To increase the utilization and reduce the energy consumption of the each core, Simulated Annealing based approach is used to slowly converge to a better solution.

KEYWORDS: Non-preemptive scheduling. Simulated Annealing, Rate Monotonic Scheduling, Slack reduction.

1 INTRODUCTION

Multi-core processors find their usage in the battery operated electronic devices like mobile phones, tablets. To increase the life of the battery operation, the cores are to be effectively utilized in a energy efficient way. Dynamic Voltage and Frequency Scaling (DVFS) is one of the popular technique used to control the clock frequency of processor or the supply voltage on the fly through the software. Most of the processors available in the market have this feature, DVFS is implemented using adjustable voltage regulators and processors with Phase-Lock Loop (PLL) controlled clocks. In this work, the tasks are scheduled to the cores using Leakage Aware Largest Task First algorithm. The frequency of operation of the tasks is reduced using the Uniform Slow down with Frequency Inheritance (USFI). The tasks are iteratively shuffled between the cores till the utilization of the cores is balanced. Then the slack time of the tasks are further utilized by increasing the slow down factor thereby reducing the frequency of operation. Reducing the frequency of operating thereby reduces the energy consumption.

2 RELATED WORK

Heuristic algorithms are used to get an optimized solution in the search space in Operations Research and find their application in Job shop scheduling and real time task scheduling. In [1], the author has addressed the job shop scheduling problem with the objective of minimizing lateness. They have pre-processed using constraint propagation technique and then applied Simulated Annealing to get an optimal solution. The authors in [2] have explored energy efficient flow shop scheduling using Genetic-Simulated Annealing Algorithm by considering make-span and energy consumption as objectives.

The combination of Dynamic Voltage and Frequency Scaling (DVFS) and Dynamic Power Management (DPM) [3] has been used for periodic dependent non-preemptive tasks on multi-core systems for optimizing energy consumption. They have formulated the problem using Mixed Integer Linear Programming. Integer Linear Program has been for energy efficient scheduling for the multi-core processors with Dynamic Voltage Scaling (DVS) feature [4]. The authors [5] have applied SA for optimization of time partitions for mixed criticality real time distributed embedded systems. Offline and online techniques [6] has been proposed to reduce the number of preemptions and to increase the idle time for the processors. The authors have [7] used SA for energy aware approach for hard real time systems on multi-core platforms. The tasks were reallocated using the penalty value calculated based on the critical speed and unbalanced time of the tasks to a new core which is comparatively less utilized. The algorithm was iterative and task reallocation was done during every hyper period.

3 System Model

3.1 Processor and Task Model

A multi core processor with L homogeneous cores is considered for analysis. The ith core of the processor is represented by Li. Each core is assumed to have identical structure and characteristics.

A periodic task set T with N tasks whose period is given by P_i and Worst Case Execution Time (WCET) by C_i is taken for analysis. The tasks has deadline D_i, which is assumed to be equal to its period. A task is characterized by a tuple, (T_i, D_i, C_i). The task requests its j^{th} execution is at time $t_{i,j}$ = (k – 1)T_i where k = {1, 2, 3 …}. B_i is the blocking time of the task and computed as B_i = max j>i C_j. The utilization u_i of the each task is calculated by u_i=C_i/P_i. The tasks are scheduled in the processor with dynamic adjustment of frequency. The speed of the processor can be varied over a discrete range from f_{min} to f_{max}. The ratio between the current processor frequency to the maximum processor frequency is called as slow down factor. The slowdown is also discrete and ranges from η_{min} to 1. η_{min} is the slow down factor calculated based on the minimum frequency level with which the processor can be operated.

4 Simulated Annealing Algorithm

Simulated Annealing proposed in [8], is a probabilistic method used for solving optimization problems. Simulated Annealing (SA) is based on the physical annealing process of metals. The temperature of the molten metal is brought down slowly for ordered alignment of the atoms. The SA algorithm starts the iteration from the set temperature T and it is reduced using a reduction factor r during every iteration. The algorithm is terminated if the desired temperature is reached. It is used in other engineering problems for which the initial solution is slowly improved by making small changes till it reaches a better solution. Simulated Annealing has been part of the Operations Research to solve large number of optimization problems such as job flow shop scheduling [9], task scheduling [10], water management for farm irrigation [11] and scheduling in Grid computing systems [12].

4.1 Simulated Annealing in scheduling of real-time tasks

In this work, Simulated Annealing has been applied for real time task scheduling of non-preemptive periodic tasks in homogeneous multi-core processor environment. The initial solution is obtained using LA + LTF [13] algorithm. Then the processor clock frequency is reduced to minimize the energy consumption using Uniform Slow down with Frequency Inheritance. The slow down factor for each task and for each core is calculated. The tasks run in parallel in the multiple cores. As in the homogenous multi-core system, all the cores run in a single frequency. So, the highest frequency among the parallel tasks is selected for operation. The laxity of each core is calculated. The slow down factor is slowly refined using SA to reach an appropriate solution.

Consider the number of cores, L = 2 and the number of tasks, N = 3 and the task set is τ_1 = (5, 5, 2), τ_2 = (20, 20, 3) and τ_3 = (30, 30, 8). The utilization of each task is 0.4, 0.15 and 0.266 respectively. After the initial population, the tasks 1 was allocated to Core 1 and tasks 2 and 3 were allocated to Core 2. The utilization of the two cores is almost equal. The tasks are allocated to the cores using Leakage Aware (LA) and Largest Task First (LTF) In LA & LTF, the tasks are sorted in the non-increasing order based on the utilization. The tasks are allocated to the core with lowest utilization. The utilization of the individual core is the sum of the utilization of all the tasks allocated to that core. The slow down for the tasks scheduled in each core is calculated using the equation 1 [14].

$$\left(\sum_{1\le r\le q}\frac{C_r}{\eta_r}\left\lceil\frac{S_{i\,j}}{T_r}\right\rceil\right)+\frac{1}{\eta_{ij}}\left(B_i+\sum_{q\le p\le i}C_p\left\lceil\frac{S_{i\,j}}{T_p}\right\rceil\right)=S_{i\,j} \tag{1}$$

With the example task set shown in Table 1, the slow down factor for the first task is 0.4 and for the second and the third tasks are 0.55 and 0.367. The maximum of the slow down factor is 0.55 and is taken for consideration. Let the maximum frequency f_{max} of operation be 20 MHz. After slow down the frequency has reduced to 11 MHz. The execution time for the tasks has changed to 3.6, 5.45 and 14.54 respectively. The new utilization of the tasks has become 0.72 and 0.7577 for first and the second core respectively. Then the tasks are scheduled using the popular Rate Monotonic Algorithm (RMA). The maximum utilization that can be achieved using RMA based on hyperbolic bound test is given by the equation 2.

$$\prod_{i=1}^{n}(U_i+1)\le 2 \tag{2}$$

Table 1. Utilization for the cores after scheduling and after applying Simulated Annealing

Utilization of the cores after Simulated Annealing									
Task no.	**C_i**	**U_i**	**Core**	**Core Utilization**	**Slow Down factor**	**Freq in MHz**	**C_i after SD**	**U_i after SD**	**Core Utilization after SD**
1	3.6	0.72	1	0.72	0.447	8.94	4.47	0.8946	0.8946
2	5.45	0.273	2	0.7577	0.441	8.94	6.79	0.3395	0.9645
3	14.54	0.4847	2		0.426	8.94	18.75	0.625	
Utilization of the cores before slow down (SD) and after slow down									
Task no.	**C_i**	**U_i**	**Core**	**Core Utilization**	**Slow Down factor**	**Freq in MHz**	**C_i after SD**	**U_i after SD**	**Core Utilization after SD**
1	2	0.4	1	0.4	0.4	8	3.60	0.72	0.72
2	3	0.15	2	0.416	0.55	11	5.45	0.273	0.7577
3	8	0.266	2		0.367	7.34	14.54	0.4847	

After initial slow down, the cores are still under-utilized, because of the blocking factor of the tasks, using equation 1, the slow down cannot be further reduced. So by applying SA, mapping the ΔE to ΔU, the utilization of the core is slowly increased in the steps of 0.05. The utilization of the core 1 has increased from 0.72 to 0.8946 and that of the core 2 from 0.7577 to 0.9645.

5 Performance Evaluation

The purpose of the scheduling algorithm in multi-core environment is to ensure that none of the core is underutilized. In this energy efficient scheduling, the algorithm balances the tasks loads between the cores evenly so that the total utilization of per core is nearest same. Consider the task set, (5, 5, 2), (15, 15, 2), (20, 20, 1), (35, 35, 5) and (45, 45, 5) is to be scheduled to a multi-core processor with 2 cores. The energy efficient schedule of two cores is shown in Fig. 1 and Fig. 2.

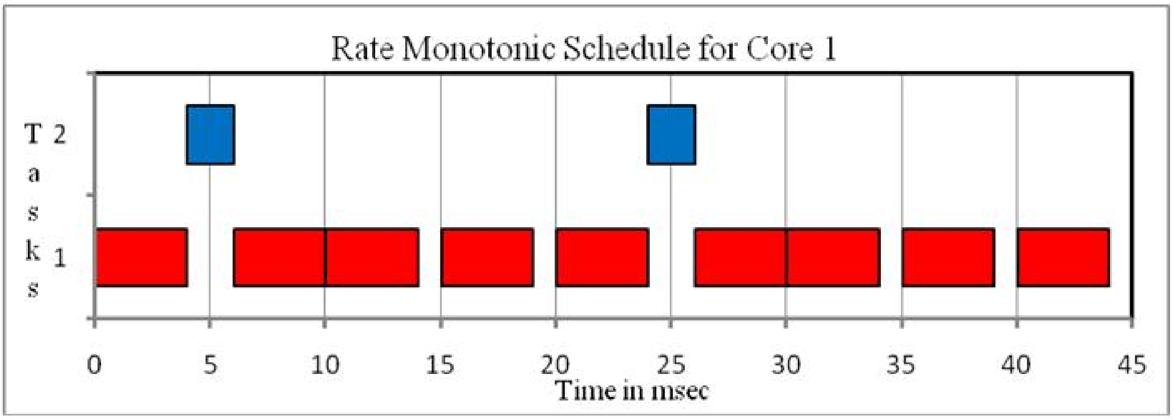

Fig. 1. Rate Monotonic Schedule for Core 1

The performance of the proposed scheduling solution using Simulated Algorithm is evaluated using utilization of the cores, normalized laxity and normalized energy of each core. The maximum frequency of operation of the processors with DVFS feature depends on the supply voltage and is given by the equation 3 [15].

$$f = k\frac{(V_{dd} - V_t)^2}{V_{dd}} \tag{3}$$

Where Vdd – Supply voltage, Vt – Threshold voltage, f – Clock frequency, k – constant and can be written as equation 4.

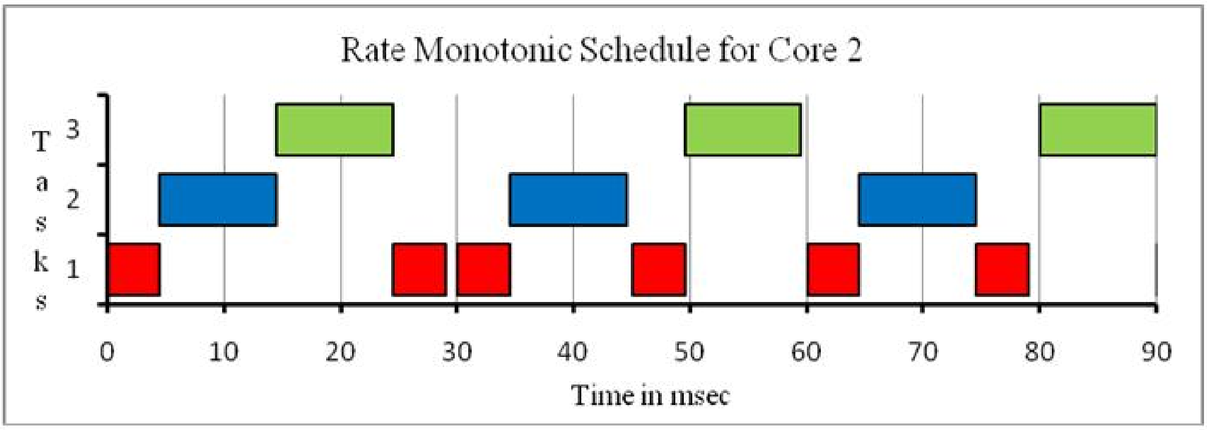

Fig. 2. Rate Monotonic Schedule for Core 2

$$f = aV_{dd} \tag{4}$$

The power consumption is given by equation 5.

$$P = C_{ef}V_{dd}{}^2 f \tag{5}$$

Here C_{eff} is the effective switch capacitance which is constant for that circuit and substituting the value of V_{dd}, the equation 5 becomes as equation 6. The power dissipated depends on the clock frequency of the processor.

$$P = Sf^3 \tag{6}$$

The normalized energy consumption is calculated by setting the energy consumed at the highest speed as 1. The normalized energy consumption and the slow down factor are shown in Fig. 3. The slow down factor for the first task set was 0.447 and that of the second one was 0.5. The cores were underutilized and were around 0.4 for each core for both the task sets. Laxity or Slack Time is the difference between the relative deadline and the response time of the task. Normalized laxity is calculated using the equation 7 [16].

$$\text{Normalized Laxity} = \frac{\text{relative deadline - resp. time}}{\text{task period}} \quad (7)$$

6 CONCLUSION

The energy efficient scheduling of the real-time tasks was performed for homogenous multi-core processor in two stages. In the first stage, USFI was used to slow down the tasks thereby reducing the operating frequency. After which the available laxity was further reduced using Simulated Annealing till the tasks are schedulable using hyperbolic bound feasibility test. After arriving on a feasible slow down factor for both the cores, the tasks are scheduled using Rate Monotonic Algorithm. The energy and the laxity of the cores are reduced as shown in the Fig. 3. This work can be further extended to heterogeneous multi-core processors.

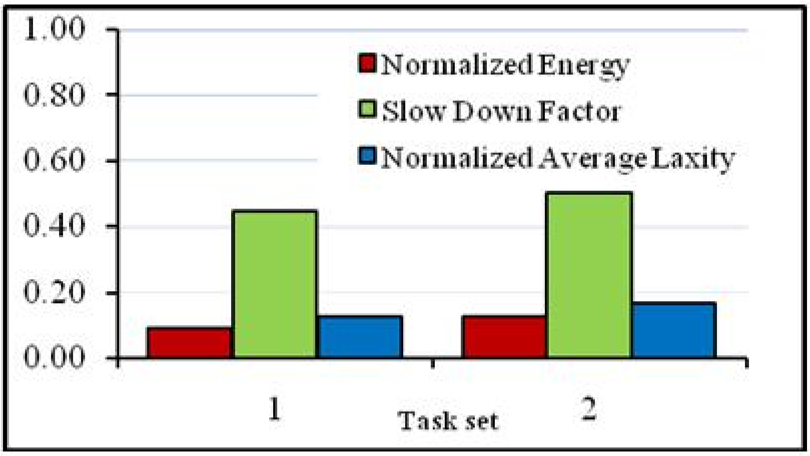

Fig. 3. Normalized energy, Normalized Average laxity and slow down factor after scheduling

REFERENCES

[1] R. Zhang, “A simulated annealing-based heuristic algorithm for job shop scheduling to minimize lateness,” *International Journal of Advanced Robotic Systems,* vol. 10, pp. 1-9, 2013.

[2] M. Dai, D. Tang, A. Giret, M. A. Salido and W. D. Li, “Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm,” *Robotics and Computer-Integrated Manufacturing*, vol. 29, no. 5, pp. 418-429, 2013.

[3] G. Chen, K. Huang and A. Knoll, “Energy optimization for real-time multiprocessor system-on-chip with optimal DVFS and DPM combination,” *ACM Transactions on Embedded Computing System (TECS),* vol. 13, no. 3s, pp. 111, 2014.

[4] A. Mishra and A. K. Tripathi, “Energy efficient voltage scheduling for multi-core processors with software controlled dynamic voltage scaling,” *Applied Mathematical Modelling*, vol. 38, no. 14, pp. 3456-3466, 2014.

[5] P. K. Saraswat, P. Pop and J. Madsen, *Task Mapping and Bandwidth Reservation for Mixed Hard/Soft Fault-Tolerant Embedded Systems*, In: *Real-Time and Embedded Technology and Application Symposium,* ACM, pp. 89–98, 2010.

[6] V. Legout, M. Jan and L. Pautet, *A scheduling algorithm to reduce the static energy consumption of multiprocessor real-time systems*, In: *Proceedings of International Conference on Real-Time Networks and Systems*, ACM, pp. 99-108, 2013.

[7] D. He and W. Mueller, “A heuristic energy-aware approach for hard real-time systems on multi-core platforms,” Microprocessors and Microsystems, vol. 38, no. 8, pp. 288–295, 2012.

[8] S. Kirkpatrick and M. P. Vecchi, “Optimization by simulated annealing,” *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[9] K. R. Arjun and M. S. Jayamohan, “Application of simulated annealing in flow shop scheduling,” *International Journal of Innovative Research in Science, Engineering and Technology*, vol. 2, pp. 460-469, 2013.

[10] I. Bate and P. Emberson, *Incorporating scenarios and heuristics to improve flexibility in real-time embedded systems*, In: *Proceeding of Real-Time and Embedded Technology and Applications Symposium*, IEEE, pp. 221–230, 2006.

[11] P. D. Brown, T. A. Cochrane and T. D. Krom, “Optimal on-farm irrigation scheduling with seasonal water limit using simulated annealing,” Agricultural Water Management, vol. 97, no. 6, pp. 892-900, 2010.

[12] W. Abdulal, A. Jabas, S. Ramachandram and O. A. Jadaan, Mutation based simulated annealing algorithm for minimizing makespan in grid computing systems, In: *Proceedings of International Conference on Electronics Computer Technology,* IEEE, pp. 90-94, 2011.

[13] Chen, Jian-Jia, H. R. Hsu and T. W. Kuo, *Leakage-aware energy-efficient scheduling of real-time tasks in multiprocessor systems*, In: *Proceedings of Real-Time and Embedded Technology and Applications Symposium,* IEEE, pp. 408–417, 2006.

[14] R. Jejurikar and R. K. Gupta, *Energy aware non-preemptive scheduling for hard real-time systems*, In: *Proceedings of Euromicro Conference on Real-Time Systems,* IEEE, pp. 21–30, 2005.

[15] S. Saha and B. Ravindran, *An experimental evaluation of Real-Time DVFS scheduling algorithms*, In: *Proceedings of International Systems and Storage Conference,* ACM, p. 7, 2012.

[16] J. Lelli, D. Faggioli, T. Cucinotta and G. Lipari, “An experimental comparison of different real-time schedulers on multi-core systems,” *Journal of Systems and Software*, vol. 85, no. 10, pp. 2405-2416, 2012.