Energy-aware Task Scheduling for Multi-Processor Systems considering Battery Lifetime

Il-jong Jung[1] and Jong-wha Chong[2]
[1]College of Information & Communications, Hanyang University, Seoul, Korea
702ho R&D Building, 17 Haengdang-Dong, Seongdong-Gu, Seoul, 133-791, Korea
[2]College of Information & Communications, Hanyang University, Seoul, Korea
718ho R&D Building, 17 Haengdang-Dong, Seongdong-Gu, Seoul, 133-791, Korea
E-mail : [1]iljongs@naver.com, [2]jchong@hanyang.ac.kr

Abstract: Increasing needs of large-scale tasks, using single processor in a system is inefficient because it causes waste of energy consumption due to high operating frequency. To overcome the defect of a single processor, multi-processor systems which can operate tasks on low frequency are required to minimize energy consumption. Since the multi-processor systems are expanded to portable devices such like PDAs, laptop computers and cellular phones, it is necessary to not only minimize energy consumption but also optimize battery lifetime because these devices are powered by batteries. This paper presents two solutions to maximize the portable device lifetime. First, the tasks are partitioned to subtasks have the different operating frequencies, which are higher than the previous subtask in a task. The operating frequencies and the subtasks' execution cycles are decided by the formulation to guarantee the deadlines and the energy efficiency. This method maximizes the energy reduction because the latter subtasks operated by the higher frequency are not executed if tasks' execution cycles are decreased. To minimize the average energy consumption, the tasks' execution times are extended to the deadlines to lower the operating frequencies. Second, the frequencies of tasks scheduled in processors are adjusted by Dynamic Voltage Scaling (DVS) to avoid that the maximum frequency of each task overlaps. This process optimizes the battery lifetime, satisfying recovery effect. The simulation results show that the proposed method saves 15%~50% more energy than the existing method.

1. Introduction

Energy consumption is a important factor to decide a lifetime of portable devices. The optimal method is needed to reduce the energy increases because the devices consist of multi-processor architecture to process heavy workload[1]. Also, battery optimization must be considered due to battery discharge characteristic called 'recovery effect[2]. It causes the battery lifetime reduction that the workload which is required to high frequency is generated continuously by the task scheduler in real-time operating system(RTOS)[2]. To profit from the recovery effect, the workload profiles of tasks scheduled in each processor is non-increasing[3]. Therefore, the energy minimization and the battery optimization must be achieved to maximize the lifetime of the portable devices at once.
The basic technique of energy minimization and battery optimization is Dynamic Voltage Frequency Scaling(DVFS). By scaling frequency, the execution time and energy consumption of the tasks can be controlled[4].

This paper presents two solutions to maximize the portable device lifetime. First, the tasks are partitioned to subtasks have the different operating frequencies, which are higher than the previous subtask in a task. The operating frequencies and the subtasks' execution cycles are decided by the formulation to guarantee the deadlines and the energy efficiency. This method maximizes the energy reduction because the latter subtasks operated by the higher frequency are not executed if tasks' execution cycles are decreased. To minimize the average energy consumption, the tasks' execution times are extended to the deadlines to lower the operating frequencies. Second, the frequencies of tasks scheduled in processors are adjusted by Dynamic Voltage Scaling (DVS) to avoid that the maximum frequency of each task overlaps. This process optimizes the battery lifetime, satisfying recovery effect.

2. Proposed Algorithm

2.1 Minimize Energy consumption

Energy minimization can be achieved through the two steps. First, the task is divided to subtasks which have same execution cycles and the operation frequency of each task has different value. The operating frequency of the subtask is higher than the previous subtask. More energy reduction is possible because the latter subtask has more energy. Second, each task is scheduled to processors by the Earliest Deadline First (EDF) and calculated workload. The sum of tasks' execution time is adjusted to their deadline and the execution time of each task is in proportion to its workload. This step makes every task be workload-balanced.

2.1.1 Subtask Frequency Formula

The task scheduler can obtain information of the tasks from the RTOS, which is worst case execution cycle(WCEC), deadline, probability distribution of execution cycle and so on. This information is used to determine the operating frequency.
The operating frequency of a task F is execution cycles C / deadline Td and each subtask's operating frequency fi is F * r. the r is the ratio factor that determine how higher or lower the operating frequency is than F. in DVFS, the dynamic power is in proportion to voltage^2*frequency and the energy is power*time, in result, the energy consumption is execution cycle * frequency^2. Because the execution time is C/F and the voltage is in proportion to frequency. The energy consumption of the subtask may expressed as

$$e_i = c_i * f_i^2 = \frac{C}{n} * F^2 \quad (1)$$

$$c_i = \frac{C}{n} * \frac{F^2}{f^2} \quad (2)$$

If the number of subtask is n, the subtask's energy consumption $c_i*f_i^2$ may be equal to the task's energy consumption divided by n. Therefore, the execution cycle of the subtask c_i can be expressed as equation (2).
The below formulas are the way to obtain the ratio r, and the summation of each subtasks' cycles is equal to cycles of the task. The formula is

$$C = \sum_{i=1}^{n} c_i \quad (3)$$

ci can be substituted to equation (2), and it is expressed as

$$C = \sum_{i=1}^{n} \frac{C}{n} * \frac{F^2}{f^2} = \sum_{i=1}^{n} \frac{C}{n} * \frac{F^2}{F^2 * r_i^2} \quad (4)$$

Since the subtask frequency f is F * r, it can be simplified as

$$C = \sum_{i=1}^{n} \frac{C}{n} * \frac{1}{r_i^2} \quad (5)$$

$$\therefore n = \sum_{i=1}^{n} \frac{1}{r_i^2} \quad (6)$$

Finally, the adequate 'r's are chosen according to equation (6) because n is a constant. If the execution cycles of the task are reduced with the high probability, the last subtask's r is even larger than the first subtask's.

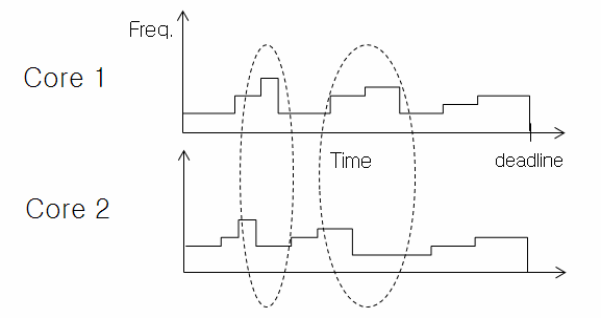

figure 1. Battery Optimization Methodology

2.1.2 Workload-Balanced Scheduling

The tasks are scheduled according to F, which represents workload. It minimizes energy consumption that all processors have the same energies. If some task is waiting for being scheduled in queue of task scheduler, workloads of each core are calculated by scheduler and then the task will be allocated to the core that has the minimum energy. This simple method distributes the tasks to the cores as possible as workload-balanced.

2.2.1 Battery Optimization

The figure 1 describes the battery optimization methodology. Each subtask has the different frequency determined by the equation (6). By up-scaling the previous subtask using DVFS, The overlap of the maximum frequencies among the processors can be avoided. This is the key point to optimize battery lifetime considering recovery effect. The energy consumption of the battery is the summation of each processor's energy, so avoiding the overlap of the maximum frequencies, the battery provides to the processors until their maximum lifetime.

2.2.2 Execution Time Adjustment

As stated above, the operating frequency can be up or down-scaling depended on the constraint, which is that each subtask's maximum frequency must not be overlapped among the processors. Although this method maximizes the battery lifetime, deadline violation can happen on a case by case basis. So, the deadline will be guaranteed after passing the below algorithm.

-Resizing-

1. $Cr_T = \{K_1, K_2, \text{L}, K_n\}$
2. for(i=0; i < n; i++)
Sizing(K in Cr, T_n)
3. compare t_i with T_i
for (i=1, $t_i \leq T_i$, i++)
{ $K_i \rightarrow Cr_S$ }
4. if ($t_i = T_i$)
{ t_i is made equal to T_i by increasing frequency
$K_i \rightarrow Cr_S$}
5. if ($Cr_T = \varnothing$) END
6. else goto 3

-Sizing-

1. for(i=1, i ≤ n, i++)

$$\{t_i = T_n * \frac{b_i}{\sum_{\substack{Ki \in Cr \\ i=1}}^{n} b_i}\}$$

Figure 2. Deadline Guarantee Algorithm

Figure 2 explains the algorithm to guarantee that deadline meets. The number 1 in Resizing algorithm is the tasks are scheduled temporally in the processor. 2 branches out to the Sizing algorithm that each task has the execution time in proportion of its execution cycles within the deadline. For example, let's assume that task K2 and K4 are allocated in processor 2 such like Figure 3. K2's execution cycle is 1million and deadline is 40milli seconds and K4 is 4 million and 50milli seconds. If there is no any fitting process, the workload will be not balanced as shown Figure 3. So, The Sizing algorithm in Figure 2 is needed to balance the workload. 3 is the process to compare the execution time of each task with its deadline. If there is no violation, the task is scheduled to the processor permanently. Unless, the operating frequency is up-scaled to its deadline, and then, the task is scheduled completely. 5 is the end function of this scheduling if there are no tasks in temporal processor, otherwise, the step is back to 3 by 6.

Reduction ratio	Proposed Algorithm	Worst Case Execution time	07' DAC
0%	7.091E+21	7.091E+21	1.559E+22
10%	5.909E21	6.382E+21	1.158E+22
20%	4.924E+21	5.673E+21	7.583E+21
30%	4.136E+21	4.963E+21	4.896E+21

Table 1. Calculated Energy Consumption

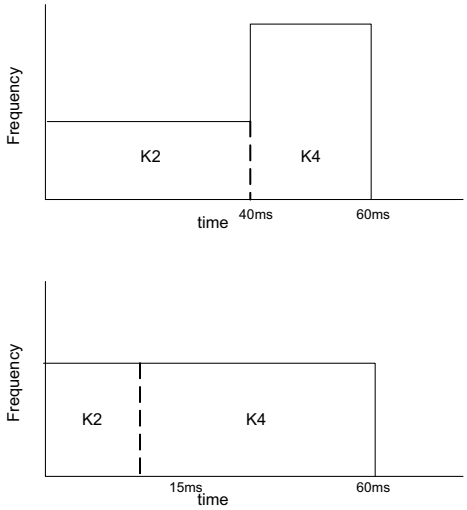

Figure 3. Example : Task Sizing

3. Simulation Result

The simulation Result of the proposed algorithm was compared to the method of task partitioned and assigned speeds based on their probability information and worst case execution time[4].
As expressed above a table 1, the energy consumption was calculated by reduction ratio of execution cycle. Reduction ration 0% means the execution cycle was worst case. The table 1 the proposed algorithm of this paper saved the energy consumption above 15%.

4. Conclusion

As portable devices develop into multi-processor architecture, the optimal task scheduling has been required to minimize the energy consumption. Also, the battery lifetime had to be considered because energy minimization is not equal to battery maximization. So, this paper proposed the method to minimize the energy consumption and optimize the battery lifetime. As result, this paper contributed to save the energy consumption and make the portable device keep the power longer.

Acknowledge

This research was sponsored by Seoul R&BD Program and ETRI SoC Industry Promotion Center, Human Resource Development Project for IT-SoC Architect. This research was also supported by the MKE(Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Assessment) (IITA-2008-C1090-0801-0019). The IDEC provide research facilities for this study.

References

[1] Fei Sun, Srivaths Ravi, Anand Raghunathan, Niraj K. Jha, "Application-Specific Heterogeneous Multiprocessor Synthesis Using Extensible Processors," In IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, 2006.

[2] Anirban Lahiri, Saurabh Agarwal, Anupam Basu, Bhargab B. Bhattacharya, "Recovery-based Real-Time Static Scheduling for Battery Life Optimization," In VLSID'06, 2006.

[3] Princey Chowdhury and Chaitali Chakrabarti, "Static Task-Scheduling Algorithms for Battery-Powered DVS Systems," In IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS, 2005.

[4] Changjiu Xian, Yung-Hsiang Lu, Zhiyuan Li, "Energy-Aware Scheduling for Real-Time Multiprocessor Systems with Uncertain Task Execution Time," In DAC, 2007.