Thermal energy aware proportionate scheduler for multiprocessor systems

Ramesh Pasupuleti 1*, Ramachandraiah Uppu 2

1 Assistant Professor, Department of Electronics and Communication Engineering, HITS, Chennai, India
2 Professor & HOD, School of Electrical Sciences, HITS, Chennai, India
*Corresponding author E-mail: pasupuleti.ramesh87@gmail.com

Abstract

As per Moore’s law, the power consumption and heat solidity of the multiprocessor systems are increasing proportionately. Higher temperature increases the leakage power consumption of the processor and thus probably escort to thermal runaway. Efficiently managing the energy consumption of the multiprocessor systems in order to increase the battery lifetime is a major challenge in multiprocessor platforms. This article presents Thermal Energy aware proportionate scheduler (TEAPS) to reduce leakage power consumption. Simulation experiment illustrates that TEAPS reduces 16% of energy consumption with respect to Mixed Proportionate Fair (PFAIR-M) and 36% of energy consumption with respect to Proportionate Fair (PFAIR) Schedulers on the system consisting of 20 processors under full load condition.

Keywords: Multiprocessor Systems; Energy Aware Scheduling; Thermal Runaway; Leakage Awareness; Critical Speed.

1. Introduction

Power density of the multiprocessor systems have doubled for every three years and this rate is expected to increase [1]. The resulting high temperature can lead to reliability issues and increase the cost of packaging and cooling. A 10⁵ C to 15⁰ C increase in temperature could lead to twice reductions in the lifetime of the device [2] due to the non-uniform power distribution of the multiprocessor system. Localized heating climbs much quickly than the entire chip, escorting to non-uniform temperature distribution on the chip with high-temperature hot spots and spatial gradients [3]. In order to control the on-chip temperature conventional methods to utilize enhanced packaging and cooling techniques such as dynamic fan cooling, water nipping and heat pipe. These methods are not appropriate due to space restrictions. Electronic design automation (EDA) flow permits the system to deal with the thermal contact on various on-chip parameters and integrate the consequences of non-uniform thermal profiles during IC design progression. These techniques are not suitable to compact with runtime situations. This article focus on software looms for thermal energy management. These approaches are flexible and eliminate the limitations described above.

In Dynamic Voltage Scaling (DVS) technology, a processor has a set of operating frequencies mapped to the supply voltage [4]. Slowdown approach is used to reduce dynamic power consumption through DVS mechanism. Slowdown techniques are generally categorized as static slowdown based on the off-line investigation and dynamic slowdown based on the on-line investigation of the task set characteristics [5]. This paper offers a thermal energy aware scheduling mechanism to minimize the leakage power consumption in the multiprocessor system. It assures resource utilization, guaranteed fairness and accuracy. This paper is structured as pursues: Section 2 gives the concise evaluation of earlier works. Section 3 illustrates the system model and TEAPS working principle. The functioning and simulation results for ample datasets are offered in Section 4. At very last, Section 5 presents the conclusion.

2. Related works

Dynamic power management (DPM) stays the system components into shutdown/ sleep states whenever they are idle [6]. Bringing back the system components into active state require more energy hence DPM is efficient only when the idle interval is greater than the threshold level. CMOS power consumption mainly depends on the supply voltage and frequency. Wayne wolf suggested that the power consumption of CMOS can be decreased by varying the voltage and frequency by maintain the entailed performance. DVFS (Dynamic Voltage Frequency Scaling) scheduler reduces the power consumption of the system with the help of Wayne wolf principle [7], [8]. Later combination of more than one voltage–frequency is used to reduce the system energy consumption [9]. Amalgamation of DVFS and Power down technologies is represented in [10-13]. It provides better energy savings when compared with DVS alone [14]. Hybrid Power Management (DPM+DVFS) Scheduling offers enhanced performance and energy savings [15]. PFAIR scheduler provides slot based approach. It is optimal for periodic tasks on multiprocessor systems [16]. To improve the quantum boundaries of the PFAIR scheduler two optimal techniques are proposed namely PF-M [17] and PD [18]. Anderson proposed optimal early release fair scheduler (ERFAIR) and tendered novel algorithm labeled PD². It is also termed as mixed PFAIR or modified PFAIR (PFAIR-M) [19].

Copyright © 2018 Ramesh Pasupuleti, Ramachandraiah Uppu. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.
3. TEAPS algorithm

Thermal Energy aware Proportionate scheduler (TEAPS) is proposed in order to increase the battery life time of the multiprocessor systems by minimizing the thermal leakage energy of the system. This work considers independent periodic task set with hard deadlines over homogeneous multiprocessor platform under symmetric shared memory architecture. It integrates the DVFS with PFAIR scheduler. This model adopts global scheduling approach under preemption mechanism.

Generally dissipation of electric energy, heats the device. Temperature rise of device depends on amount of energy dissipation, device heat capacity and heat flow between the device and environment. As per Moore’s law the power consumption and heat solidity are increasing proportionately for the multiprocessor systems. Bar of cross section A, length l, connecting heat source with temperature $T_{source}$ and heat sink with temperature $T_{sink}$. Heat flow $P_{heat}$ across the bar between two bodies can be described as

$$P_{heat} = \frac{H_t A (T_{source} - T_{sink})}{t}$$

Here $H_t$ represents bar heat conductivity. Energy consumption due to thermal leakage ($E_{heat}$) can be represented in terms of $P_{heat}$ as

$$E_{heat} = P_{heat} \times t$$

Here t represents the time period of the processor. It clearly indicates the energy consumption of the system is directly proportional to the power consumption.

Temperature difference between die and environment $\Delta T$ is determined by the power dissipation in the die and thermal conductivity of the package $H_{package}$. Thermal conductivity of the package depends on the package geometry, size and material used. Its value is in between 0.1 and 1 W/C.

$$\Delta T = \frac{P_{die}}{H_{package}}$$

Here $P_{die}$ represents total power dissipated in the die.

3.1. System model

The system consist ‘m’ homogeneous processors $P_1$, $P_2$, ....... $P_m$. It consist a set of ‘N’ independent periodic tasks { $\tau_1$, $\tau_2$, ...... $\tau_N$ } with hard deadlines. The distinctiveness of each task is characterized as $(C_i, p_i, d_i)$ here $C_i$ indicates task worst case execution time, $p_i$ denotes the time period and $d_i$ represents the deadline of the task. The priorities of the tasks are assigned based on their deadlines. It allows preemption mechanism. It tries to schedule tasks such that in t time slots from arrival of task $i$, at least $\frac{C_i}{p_i}$ a unit of execution time is given. The total utilization of the task set

$$U = \sum_{i=1}^{N} (\frac{C_i}{p_i}) \leq m$$

Here $(C_i/p_i)$ represents weight of the task. The challenge in integrating DVFS with the PFAIR algorithm is that a task’s parameter must change when scale down the task in order to reduce the thermal energy consumption. When a task is scaled down, its execution time changes according to the new frequency and the PFAIR scheduling algorithm needs to modify the weight of a task as per the new frequency. To maintain acceptability of the PFAIR scheduling algorithm and to accommodate the notion of variable weights; TEAPS calculate the lag and characteristic string of a task incrementally. The weight of task $\tau_i$ in slot t is defined as $w_{t \rightarrow i}$. The lag of a task at time $t$ is computed incrementally as

$$\text{lag}(\tau_i, t) = \text{lag}(\tau_i, t - 1) + w_{t \rightarrow i - 1}$$

For all $t>0$ and its value becomes zero when $t=0$.

The ideal allocation (id) of a task at a time t is

$$\text{id}_i = \text{id}_{t-1} + w_{t-i}$$

The characteristic string of a task at a time t is

$$\alpha_i(t) = \text{sign}(\text{id}_{t+1} - |\text{id}_i| - 1)$$

3.2. Working principle

TEAPS enables the processors to function with different voltage levels in order to eliminate the temperature emergency to diminish the energy consumption of the system. TEAPS minimizes the execution of the task without defy the timing limitations.

Algorithm 1 describes TEAPS mechanism. Scheduler assigns ready high priority task to the available processor. Initially processor runs under maximum frequency ($f_{max}$) condition. TEAPS periodically computes the predicted temperature of the processor ($T_{Pk}$) depend on the current temperature ($T_{Ck}$), neighbor processor temperature ($T_i$), power consumption of the processor ($P_k$) and processor floor planning parameters $\alpha$ and $\beta$. TEAPS compares predicted temperature with the threshold temperature of the processor ($Tn$).

$$T_{Pk} = T_{Ck} + \sum_{k=1}^{i} T_i - T_{Ck} + \beta P_k$$

If the predicted temperature is exceeding the threshold temperature value then the scheduler scale down the speed of the processor to diminish the thermal energy leakage on the processor. TEAPS runs the processor at minimum frequency ($f_{min}$) and it updates the weight of the task as per new frequency is

$$w_{t \rightarrow i} = \frac{C_{i \rightarrow j}^{\text{new}}}{P_i}$$

Here $C_{i \rightarrow j}^{\text{new}}$ is the run time of a task with workload at a frequency ‘$f_{new}$’. TEAPS always guarantees that the minimization of the speed is not at all located beneath the critical speed. The speed of the processor under minimum frequency condition is called critical speed. To implement the task beneath the critical speed will increases the chances of missing the deadline and it’s not energy efficient. Else the predicted temperature is below the threshold temperature value then the scheduler allows the processor to run under maximum frequency ($f_{max}$) and it updates the weight of the task as per the frequency.

3.3. Processor model

It considers ARM7TDMI based LPC2148 microcontrollers for implementation. The system consists of 20 ARM7TDMI systems which are connected under symmetric shared memory architecture. It represents homogeneous multiprocessor platform. Operating points of ARM7TDMI are represented in the table 1. ARM7TDMI processors are equipped with thermal sensors to detect the over-heating on chip. The processors will be throttled if the on-chip temperature exceeds a certain predefined threshold. Thermal sensors produce noise in the data which may result inaccuracy in the final output. Cochran and Reda proposed Nyquist-Shannon sampling technique to estimate temperature characteristics of the whole chip by using [20]. The absolute estimation error of temperature characteristics of the chip is only 0.6%. TEAPS control the high temperature on the processor in order to reduce the leakage power consumption.
Table 1: Operating Points of ARM7TDMI

| Operating Points | Clock Frequency (MHz) | Voltage (V) |
|------------------|-----------------------|-------------|
| $f_{\text{min}}$ | 12                    | 2.8         |
| -                | 24                    | 3.0         |
| -                | 48                    | 3.1         |
| $f_{\text{max}}$| 60                    | 3.3         |

Algorithm 1: Sample TEAPS Algorithm

for $i$=1 to N do
  for $k$=1 to m do
    Assign $f \leftarrow f_{\text{max}}$
    Calculate $T_P = T_C + \sum_{i=1}^{m} \alpha_k (T_i - T_C) + \beta_k P_k$
    if ($T_P > T_{th}$) then $f_{\text{new}} \leftarrow f_{\text{min}}$
    Update weight of task as per new frequency
    $w_i' = \frac{C_i}{P_i}$
    else $f_{\text{new}} \leftarrow f_{\text{max}}$
    Update weight of task as per new frequency
    $w_i' = \frac{C_i}{P_i}$
  end if
end for
end for

4. Results

Simulation based trials were performed to compare the efficiency of the proposed scheduler with PFAIR and PFAIR-M schedulers. In Proportionate fair (PFAIR) scheduler, each task is performed at an estimated uniform rate by breaking it into a series of quantum length subtasks. Mixed PFAIR (PFAIR-M) algorithm provides task partitioning approach. It reduces overhead and improves the performance of PFAIR scheduler. Both of these algorithms don’t concentrate on thermal leakage energy consumption of the system. The proposed TEAPS algorithm curtails the leakage power due to high temperature conditions in order to exploit the system battery life time.

4.1. Datasets

Randomly produced hypothetical task sets are used for simulations. The weights of the tasks were generated using rand fixed sum method [21]. The period of the tasks was taken from normal distributions with $\sigma = \mu/5$ here $\mu = 100$ hence $\sigma = 20$. Number of tasks was ranging from 10 to 100 and number of processors varying from 2 to 20. Different system load values are considered for simulation. ARM7TDMI processor power model is taken into consideration.

4.2. Simulations

This segment investigates the performance of TEAPS algorithm in comparison to PFAIR-M and PFAIR algorithms. System parameters are processors (m), tasks (N), system load (L) and task period (P). The span of a time slot is 1 msec. Results in this segment are standard over 100 runs of the scheduler for 100000 time slots of schedule duration on a structure with 20 processors and 30 tasks.

4.3. Discussions

Figure 1 demonstrates the variation of Energy Consumption with the numbers of processors in the system by maintaining number of tasks as 30 and system load as 80%. The average period of a task in the experiment is taken as 500 msec. From the plot it can be seen that the TEAPS algorithm illustrates a linear increase in the energy consumption as the number of processors increases.
Figure 2 represents variation of Energy Consumption with the numbers of tasks in the system by maintaining number of processors as 20, system load as 80% and average task period as 500 msec. System queue certainly not turn out to be empty frequently due to increase in the number of tasks which affects the task weights. Due to this energy consumption of TEAPS algorithm increases initially and then runs almost constant. The energy consumption of PFAIR and PFAIR-M algorithms increases with the number of tasks due to variation in task weights. Figure 3 represents the variation of the Energy Consumption with the average task period in the system by maintaining number of processors as 20, system load as 80% and number of tasks as 30. As task period increases the energy consumption of TEAPS increases initially and then runs approximately constant. The energy consumption of PFAIR and PFAIR-M algorithms increases due to increase in the task execution requirements with the increase in the task period. Results illustrate poor performance of PFAIR scheduler compared to PFAIR-M and TEAPS algorithms. Figure 4 represents the variation of the Energy Consumption with the processor load by keeping number of processors as 20 and numbers of tasks as 30. The average period of a task in this trial is taken as 500 msec. The average task weight increases with the system load since the number of tasks are constant. Due to this the energy consumption of TEAPS, PFAIR-M and PFAIR algorithms increases with the increase in processor load. Results illustrate better performance of TEAPS scheduler compared to PFAIR-M and PFAIR algorithms. From the plots it can be noticed that the TEAPS minimizes the thermal energy consumption of the system in order to increase the system battery life time. It reduces 16% of energy consumption compare to PFAIR-M and 36% of energy consumption compare to PFAIR algorithm for the system consisting of 20 processors.

4.4. Correctness of TEAPS

TEAPS is correct in the wisdom that no task missed its deadline due to task reweighting mechanism. Basically, PFAIR was developed with static task weights throughout their life time. Task reweighting was modelled as a combination of join and leave properties. The task with current weight leaves the system when the scheduler terminates the task and a task with new weight joins the system as long as the system total utilization remains within ‘m’ and also task’s utilization remains within 1. Since, it can be certify that TEAPS maintain the correctness during the task reweighting mechanism.

5. Conclusion

In this effort, we offered a novel energy-efficient TEAPS scheduling algorithm to curtail the leakage power due to high temperature. This strategy is applicable in dynamic scenarios. It increases the battery life time of the multiprocessor systems. TEAPS reduces the energy consumption of 16% with PFAIR-M and 36% with PFAIR scheduler on the system consisting of 20 processors under full load. It is feasible and optimal for periodic task sets on multiprocessor systems. It is recommended to enable multiple frequency operating conditions for minimizing the power consumption of multiprocessor systems further more without violating the timing constraints.

Competing interests

The authors declare that they don’t have the competing interests.

References

[1] Skadron K, Stan M.R, Huang W, Velusamy S, Sankaranarayanan K and Tarjan D, “Temperature-aware computer systems: Opportunities and challenges”, *Micro*, IEEE, 23, (2003), pp. 52–61. https://doi.org/10.1109/MM.2003.1261387.
[2] Semenov O, Vassighi A and Sachdev M, “Impact of self-heating effect on long-term reliability and performance degradation in CMOS circuits”, *IEEE Transactions on Device and Materials Reliability*, 6(2006),pp.17–27. https://doi.org/10.1109/TDMR.2006.870340.
[3] Skadron K, Stan M.R, Huang W, Velusamy S, Sankaranarayanan K and Tarjan D, “Temperature-aware microarchitecture: Modeling and implementation”, *ACM Transactions on Architecture and Code Optimization*, (2004), pp.94–125. https://doi.org/10.1145/980152.980157.
[4] Aydin H, Melhem R, “Determining optimal processor speeds for periodic real-time tasks with different power characteristics”, *Proceedings of Euromicro conference on real-time systems*, (2001), pp. 225–232. https://doi.org/10.1109/EMRTS.2001.934038.
[5] Augustin J, Irani S and Swamy C, “Optimal power-down strategies”, *Proceedings of 45th Annual IEEE Symposium*, (2004), pp. 530 – 539. https://doi.org/10.1109/FOCS.2004.50.
[6] Sandy Irani, Sandeep Shukla and Rajesh Gupta, “Online strategies for dynamic power management in systems with multiple power-saving states”, *ACM Transactions on Embedded Computing System*, Vol.2,No.3,(2003),pp.325–346. https://doi.org/10.1145/860176.860180.
[7] Kim W, Kim J and Min SL, “A dynamic voltage scaling algorithm for dynamic-priority hard real-time systems using slack time analysis”, *Proceedings of the conference on Design, automation and test*, Europe, (2002), https://doi.org/10.1145/1134650.1134673.
[8] Jia Jin Chen, Chun Yue Yang, “Energy-Efficient Real-Time Task Scheduling in Multiprocessor DVS Systems”, *IEEE transactions on computers*, (2007), Page 342.
[9] Nikzad Babaii Rizvandi and Jagid Taheri, “Some observations on optimal frequency selection in DVFS-based energy consumption minimization”, *Journal of Parallel Distributed Computing*, Elsevier, (2011), pp.1154–1164.
[10] Chen J and Kao T, “Procrastination for leakage-aware rate-monotonic scheduling on a dynamic voltage scaling processor”, *Proceedings of the ACM SIGPLAN/SIGBED Conference on Languages, Compilers and Tools for Embedded Systems*, (2006), pp. 153–162.
[11] Jejurikar R and Gupta R, “Procrastination scheduling in fixed priority real-time systems”, *ACM SIGPLAN Notices*, vol. 39, no. 7, (2004), pp. 57–66. https://doi.org/10.1145/998300.997173.
[12] Jejurikar R, Pereira C and Gupta R, “Leakage aware dynamic voltage scaling for real-time embedded systems”, *Proceedings of the 41st annual conference on Design automation*, New York, ACM, (2004), pp. 275–280. https://doi.org/10.1145/996566.996650.
[13] Lu Z, Zhang Y, Stan M, Lach J and Skadron K, “Procrastinating voltage scheduling with discrete frequency sets”, *Proceedings of the conference on Design, automation and test*, Europe, (2006), pp.456–461.
[14] Lorch JR and Smith AJ, “Improving dynamic voltage scaling algorithms with pace”, *Proceedings of the ACM SIGMETRICS international conference on Measurement and modelling of computer systems*,(2001),pp.50-61. https://doi.org/10.1145/378420.378429.
[15] Viney Devadas and Hanan Aydin “On the Interplay of Voltage/Frequency Scaling and Device Power Management for Frame-Based Real-Time Embedded Applications”, *IEEE transactions on computers*, vol. 61, no. 1, (2012), Page 31. https://doi.org/10.1109/TC.2010.248.
[16] Barua S, Cohen N, Plaxton CG, Varvel D, “Proportionate progress: a notion of fairness in resource allocation”, *Algorithmica*,15,(1996),pp.600–625. https://doi.org/10.1007/BF01940853.
[17] Barua S, Gehrke J, Plaxton CG, “Fast scheduling of periodic tasks on multiple resources”, *Proceedings of Ninth International Parallel Processing Symposium,(1995),pp.280–288. https://doi.org/10.1109/IPPS.1995.395946.
[18] Goosens J, Funk S, and Barua S, “Priority-driven scheduling of periodic tasks systems on multiprocessors”, *Real Time Systems*, 25(2-3),(2003),pp.187–205. https://doi.org/10.1023/A:1025120124771.
[19] Anderson JH, Srinivasan A, “Mixed PFAIR/PFAIR scheduling of asynchronous periodic tasks”, Journal of Computer and System Sciences, 68, (2004), pp. 157–204. https://doi.org/10.1016/j.jcss.2003.08.002.

[20] Cochran R and Reda S, “Spectral techniques for high-resolution thermal characterization with limited sensor data”, Proceedings of the Design automation Conference, (2009), pp. 478–483. https://doi.org/10.1145/1629911.1630037.

[21] Emberson P, Stafford R, and Davi R, “Techniques for synthesis of multiprocessor task sets,” workshop at the Euro micro Conference on Real-Time Systems, (2010). pp. 6–11.