DTACU: a global DVFS algorithm based on piece energy model

Longfei Li, Zhanzhuang He, Juli Zhang, Xupei Zhang

Xi'an Microelectronics Technology Institute, Xi'an, 710065, China

{longfeisos,hzz771,juli_zh,xupeijack}@163.com

Keywords: global DVFS algorithm, multicore processor, piece energy model, low power design

Abstract. Considering the exclusive policy and poor energy benefits for multicore processor by Ondemand algorithm, a global DVFS (Dynamic Voltage and Frequency Scaling) algorithm DTACU based on piece energy model is proposed. In order to translate multicore problems to single core problems, an equivalent energy model, called piece energy model, was set up. It divided the time into several periods in which no task arrive or finish in each core, hence the multicore processor can be seen as a single core processor. Based on theoretical research on energy consumption and current published papers, analytical expressions of equivalent single core energy consumption were deduced. DTACU was proposed through analyzing the source of energy consumption in expression and implemented based on Userspace. Taking the number of active cores and dynamic thresholds into account, DTACU only change CPU speed during one period to both reduce energy consumption and satisfy the demand of task deadline. The final result of experiments show that DTACU can achieve lower energy consumption without sacrificing system performance compared with Ondemand.

Introduction

Nowadays, reducing the energy consumption is considered as one of the major research topics for computing system, especially for mobile computing system. Comparing with low-power component design and manufacturing technology, processor low-power design at system-level is also meaningful. Dynamic Voltage and Frequency Scaling (DVFS) is one of feasible means of reducing energy consumption. It is a hardware technology used to dynamically scale up or down the frequency or voltage of processor according to the governor policy and the workload demand at the cost of an increased latency.

For multicore processor, there are two main means of DVFS control, namely local DVFS and global DVFS. Local DVFS changes the clock frequency per core, while global DVFS makes these changes for the entire chip. Recently, clustered DVFS, which uses multiple of on-chip regulators drive a set of DVFS domains, was proposed for multicore processors[1]. Theoretically, local DVFS is the most effective way to reducing energy consumption. However, in practice global DVFS is the most common in processor design, since it is cheaper to implement, but there are still some issues on global DVFS, such as low energy benefits and complex algorithms.

In this paper, we explore and propose a global DVFS algorithm DTACU (Dynamic Threshold and Active Cores based on Userspace) by analyzing the relationship between frequency and energy consumption based on piece energy model. Section 2 introduces the related works about global DVFS. After presenting the detailed process of setting up the piece energy model, the description and implementation of DTACU are presented in Section 4. The results of experiment are shown in section 5, and final conclusion is given in section 6.

Related Works

Many DVFS power management policies were proposed in the past. Some of them have been integrated in many kinds of processor, but there are no known optimal algorithm that minimize the energy consumption on global DVFS. Jean-Philippe et al. [2] study reactive DVFS control for multicore processors and realize FoREST, a new DVFS runtime controller independent form any performance model, in multicore system. Although FoREST does not use any performance model, it must under the user-defined slowdown constraints. In other words, it may generate a huge latency that is based on the programs running on the cores. Semeraro et al. [3] proposed to periodically reduce CPU frequency until an impact on execution time is suspected from hardware observation, but the proposed mechanism is not able to control its impact on slowdown as opposed to more recent solutions. Besides, some of algorithm focus on reducing the energy consumption of a specific program [4]. The optimal global DVFS algorithm should apply to all programs, so vifrification to host programs is the most basic elements for global DVFS.

Piece Energy Model Introduction

Multicore Power model.

The power consumption, P, of a CMOS-based multicore processor is related to its system clock frequency f and operating voltage V as $P \propto V^2 \bullet f$. Also, the relation $V \propto f$ holds for these processors [5]. In this paper, the model follows the commonly accepted modeling assumptions from the literature. Because of widespread use of CMOS technology, so the power consumption of a multicore processor, it was agreed, consists of the static power consumption and dynamic power consumption. Static power consumption refers to the power consumption from the leakage current. The static power of a processor is often approximated by a linear function of the voltage in other literature, so it can be expressed as $P_s = cV + c_2$. It is a common assumption that the voltage is linearly related to the clock frequency in the power management literature [6], so we simplify the static power as $P_s = c_1 f + c_2$.

Dynamic power is the power caused by capacitance charging and discharging when signals changing in the circuit, which is a major part of circuit power. In this paper, we assume that dynamic power is only used when a core is active. Under this assumption, the dynamic power can be expressed as $P_d = \alpha m C V^2 f$, where C denotes load-capacitance in the circuit, α is technology dependent constant and m is the number of active cores. Just because we assume that the voltage is linearly related to the clock frequency, so dynamic power can be changed as $P_d = m c_3 f^{\beta}$. The total power model of the multicore processor is given by:

$$\mathrm{P} = c_1 f + c_2 + m c_3 f^{\beta}, \tag{1}$$

which is only a function of system frequency. This is what we expect, because we will analyze clock frequency and show our frequency selection policy in the next chapter. After getting power model, we will introduce piece energy model to calculate the energy consumption of multicore processors.

Piece Energy Model.

We obtain the total energy consumption by integrating power over time, namely when exactly m cores are active during the time interval $[t_1, t_2]$, the energy consumption for this time interval is

$$\mathrm{E} = \int_{t_1}^{t_2} P\varphi(t)\,dt . \tag{2}$$

In order to obtain a more common, intuitive and simpler energy consumption model for multicore processor, we have several assumptions which are feasible for our model. First, we assume that the tasks are large enough, so we can neglect the overhead of changing clock frequency. Second, task schedule, we have mentioned before, is given and feasible, so we do not formalize the definitions of tasks. Third, we assume that the clock frequency is only changed when tasks start or finish. This assumption is based on observation that the number of clock frequency changes of the optimal policy grows slowly with the number of tasks and we will discuss it later.

Before we introduce the model, we should know a definition of piece, which is a maximal interval [a, b] that no task starts or finishes in (a, b). Under this definition, we can see the total energy consumption as several pieces of energy consumption, so we call such energy model as piece energy model.

Each task has its arrival time and deadline, therefor it should be started at arrival time and finished before the deadline. To easily understanding, we show an example in figure 1 to explain the insights of the piece model. From the example we can see that there are 7 tasks running on 4 cores in the processor. The tasks have precedence constraints, and the amount of work of the tasks $T_1, T_2 \ldots T_7$ is given. Each task has its deadline respectively, thus there are 5 pieces in the example. A given schedule, it will be noticed, uniquely subdivides into K pieces, and P_k denotes the k-th piece.

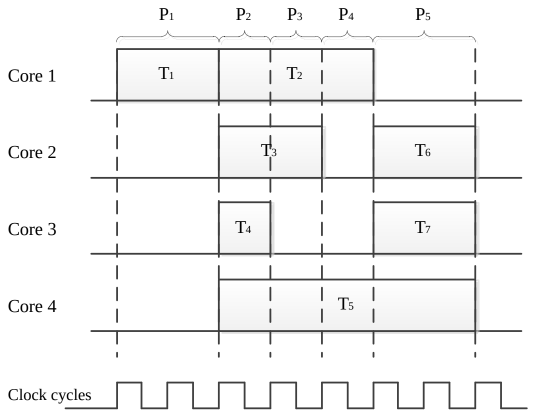


Fig. 1. An example of tasks schedule

Let m_k denote the number of active cores during piece P_k and w_k denote the total workload in the period of P_k. Note that the number of active cores during P_k can't change, this is because no task starts or finishes during this piece. Hence $m_k w_k$ work is executed for piece P_k. We can obtain total work W of the application by summing the work of all tasks, and it can be expressed by $\mathrm{W} = \sum_{k=1}^{K} m_k w_k$. For the example shown in figure 1, it takes 7 clock cycles to execute 7 tasks, which the total workload W equals to 17.

We use t_k^b and t_k^c to denote the start time and complete time respectively for each piece P_k. Because of $t_k = w_k / f_k$ and the power model we have obtained before, thus the total energy consumption can now be expressed in terms of pieces as follow:

$$\mathrm{E} = c_2\left(t^c - t^b\right) + \sum_{k=1}^{K} m_k c_3 f_k^{\beta-1} w_k \tag{3}$$

where t^c represents the complete time of the last piece and t^b represents the start time of the first piece. It can be seen from the above equation that the total energy consumption also consists of the static and dynamic energy.

It has been proven that when at time t_1 there are n active cores and at time t_2 there are m active cores, the ratio between the optimal clock frequencies is $\sqrt[\beta]{m/n}$. We let n equals to 1 then we get $\widehat{f_k} = f_k \sqrt[\beta]{m_k}$. Similarly, $\widehat{w_k} = w_k \sqrt[\beta]{m_k}$. We substitute the variables for clock frequencies and work to energy equation and m_k is disappear. Thus, the multicore energy consumption model becomes an equivalent single core model. The substitution into the energy function is given as follows.

$$\mathrm{E} = c_2\left(t^c - t^b\right) + \sum_{k=1}^{K} c_3 \left(\widehat{f_k}\right)^{\beta-1} \widehat{w_k} \tag{4}$$

DTACU Description and Implementation

Equivalent Model Analysis.

Frequency changes are not always effective to power consumption, because energy saving achieved by frequency change is almost as large as the energy consumption by DVFS in some case. We can see from the total energy consumption function that dynamic energy consumption increases as the number of frequency change increase. Hence, when to change frequency for global DVFS is an important problem. Because we subdivide tasks into pieces, and there is no active core changes during the piece, so it is similar with an equivalent single core. From the model we can see that it is crucial to global DVFS that the clock frequency is only changed when piece start in multicore processor. Although flexible frequency change can save much more energy by rational use of DVFS, this principle is more feasible considering the energy consumption and risks of DVFS。

In the total energy consumption model, it seems that there is no influence between frequency and static energy. Conversely, frequency indeed influences the static energy. Because tasks, it is true, can be completed before the deadline. This means E_{static} is the worst case and actual static energy consumption may less lower. So an appropriate increase in frequency would get a notable energy saving that may larger than energy consumption by dynamic power.

DTACU Description.

DTACU try to reduce the number of frequency change based on the analysis above. The frequency would be changed after the first interval of the piece. During the interval, CPU utilization can be obtained and appropriate frequency can be confirmed. Because of no change of the number of active cores in the piece, the CPU utilization can represent the demand of multicore tasks in the interval.

Note that the number of active cores changes for each piece, so active cores should be taken into consideration of global policy. The frequency would be ramped up to the maximum if the CPU utilization exceeds the 90% when all cores are active. This is a general agreement considering user experience. However, it is unnecessary to increase the frequency to the maximum when not all of cores are active because frequency variation is not linear.

Based on aforementioned model and analysis, we put forward three principles of DTACU as follow:

(1).The frequency is only changed when active cores changing;

(2).Different policies for different number of active cores;

(3).Different threshold utilizations depending on the number of CPU core.

When all cores are active, we can regard CPU as a single core, so the total CPU utilization is a representation of the system demand and we do not need to consider the single core utilization at all. In other cases, not all cores are active, single core utilization becomes a determining factor, so the DTACU needs to deal with different situations respectively. For dual-core processor, the total utilization is 50% when one core is full load and another is idle. If the same task was running on dual-core, then the total utilization is still 50%, but the single utilizations of both cores are 50% as well, so total utilization is not meaningful for not-all core active cases. On the other hand, we define different threshold utilizations depending on the number of CPU core in order to achieve better schedule performance. Basically, 50% or 75% are usually chose for up-threshold and 40% for down-threshold based on the Ondemand [7].

DTACU Implementation.

DTACU was implemented based on Userspace governor, which allows user to manually set processor frequency in Linux [8]. In order to change CPU frequency, as an example, cpufreq that inside the kernel of Linux allows the user to set the desired frequency at any time.The pseudocode of DTACU is shown below, where the up-threshold and down-threshold will be determined by the number of CPU core.

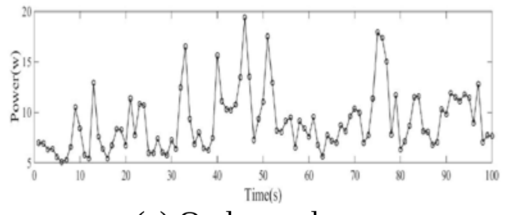
```
Get available frequency
    every internal after new piece
        if(all cores are active)
            if(total utilization > 90%)
                increase frequency to MAX
            if( 90% > total utilization > up-threshold)
                increase frequency to upper level
            if(total utilization < down-threshold)
                decrease frequency to lower level
        else
          if(not all cores are active)
            if(single utilization > up-threshold)
                increase frequency to upper level
            if(single utilization < down-threshold)
                decrease frequency to lower level
```

Experiments

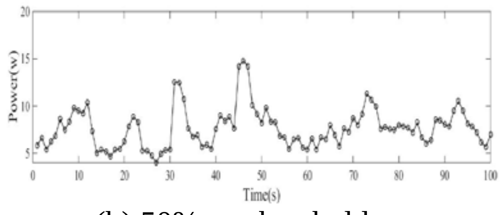
DTACU is implemented for x86_32 processors on Linux Userspace governor. The experiments were run on Intel Core i3-2350M quad-core processor, running Linux 3.5.3. There are ten available frequencies range between 0.8GHz and 2.3GHz except for the frequency below f_{crit}. The benchmark programs is the SPLASH2 running in C class datasets, which consists of twelve programs.

In order to change the frequency for each piece, we predefine the sequence of the programs running on the different cores and it will send a signal to DTACU when one program is done. Thus, DTACU can distinguish each piece and start work to determine the frequency. We use FTA22J-12, a power usage monitoring tool of Monsoon Solutions Company, to monitor the power of processor.

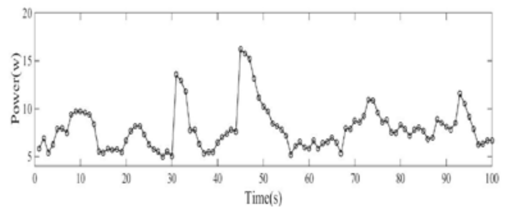
Two groups of experiments were conducted. Note that we predefined two kinds of program sequence for each group of experiment, so we firstly run the experiment with the up-threshold 50% and 75% respectively and down-threshold 40% uniformly under the qual-core situation by disabling two of cores in processor. We capture the power spectrum during the first 100 seconds, and the sample interval is one second. The results are the median value of 5 executions, as shown in Figure 3, and compared to Ondemand, the default Linux DVFS control policy.



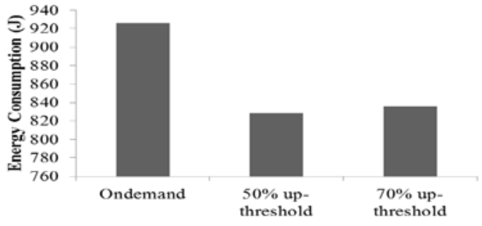
(a) Ondemand



(b) 50% up-threshold



(c) 75% up-threshold

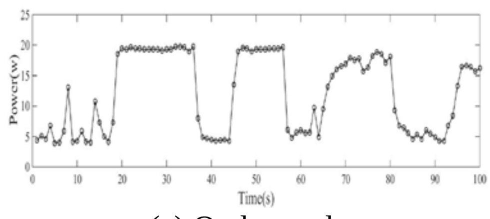


(d) energy consumption

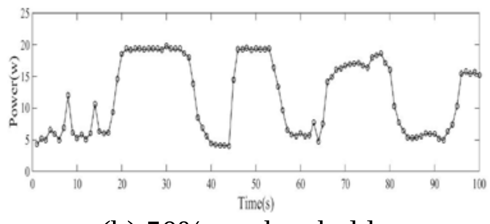
Fig. 3. Power spectrum and energy consumption under dual-core situation

From the figure above we can see that the curve of Ondemand is rougher than which of our policy no matter what value up-threshold is. Besides, the average power values of DTACU are smaller than which of Ondemand. There is no obvious difference between 50% up-threshold and 75% up-threshold, and the average power values are 8.05 w and 8.10 w respectively.

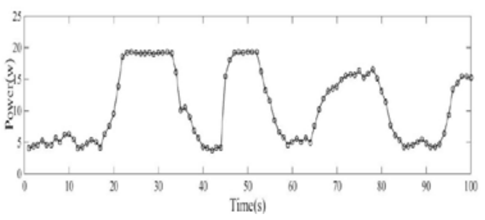
We conduct the similar experiments with the same parameter of 50% and 75% for the qual-core processor. Interestingly, the result is different to the dual-core situation and the up-threshold becomes a critical factor to determine the power consumption. This feature is reflected in Figure 4 as the power variation of 75% up-threshold is gentler than others.



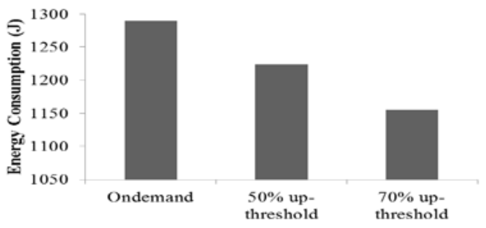
(a) Ondemand



(b) 50% up-threshold



(c) 75% up-threshold



(d) energy consumption

Fig. 4. Power spectrum and energy consumption under qual-core sintuation

DTACU consumes less energy compared with the Ondemand policy in all four cases. This is because Ondemand policy takes many times of frequency change during the experiment. Besides, it is a coarse-grained policy that only based on the total threshold user predefined. Under the qual-core, comparing with Ondemand, the policy with 75% up-threshold can achieve 10.44% lower power consumption but 5.10% for the policy with 50% up-threshold.

Conclusion

This paper puts forward a global DVFS algorithm DTACU for multicore based on the piece period, in which there is no active cores change, so we can see it as a single core. In the implementation, taking the number of active cores and dynamic thresholds into account, DTACU only change CPU speed during one period to both reduce energy consumption and satisfy the demand of task deadline. Although DTACU is experimented under assumptions, the insights of the algorithm are also useful for future work.

References

[1] Kolpe T, Zhai A, Sapatnekar S S. Enabling improved power management in multicore processors through clustered DVFS[C]//Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011. IEEE, 2011: 1-6.

[2] Halimi J P, Pradelle B, Guermouche A, et al. Reactive DVFS Control for Multicore Processors[C]//Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing. IEEE, 2013: 102-109.

[3] Semeraro G, Albonesi D H, Dropsho S G, et al. Dynamic frequency and voltage control for a multiple clock domain microarchitecture[C]//Microarchitecture, 2002.(MICRO-35). Proceedings. 35th Annual IEEE/ACM International Symposium on. IEEE, 2002: 356-367.

[4] Freeh V W, Lowenthal D K. Using multiple energy gears in MPI programs on a power-scalable cluster[C]//Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming. ACM, 2005: 164-173.

[5] Aydin H, Yang Q. Energy-aware partitioning for multiprocessor real-time systems[C]//Parallel and Distributed Processing Symposium, 2003. Proceedings. International. IEEE, 2003: 9 pp.

[6] Park S, Park J, Shin D, et al. Accurate modeling of the delay and energy overhead of dy-namic voltage and frequency scaling in modern microprocessors [J]. Computer-Aided De-sign of Integrated Circuits and Systems, IEEE Transactions on, 2013, 32(5): 695-708.

[7] Liang W Y, Chen Y L, Chang M F. A memory-aware energy saving algorithm with performance consideration for battery-enabled embedded systems[C]//Consumer Electronics (ISCE), 2011 IEEE 15th International Symposium on. IEEE, 2011: 547-551.

[8] Miyakawa D, Ishikawa Y. Process oriented power management[C]//Industrial Embedded Systems, 2007. SIES'07. International Symposium on. IEEE, 2007: 1-8.