Energy-Efficiency on a Variable-Bitrate Device

Yung-Hen Lee[1], Jian-Jia Chen[2], and Tei-Wei Kuo[2]

[1] Advanced Micro Devices (AMD)
Email: henryyh.lee@amd.com
[2] Department of Computer Science and Information Engineering
National Taiwan University, Taiwan.
Email:{r90079, ktw}@csie.ntu.edu.tw

Abstract. Dynamic power management has been adopted in many systems to reduce the power consumption by changing the system state dynamically. This paper explores energy efficiency for systems equipped with PCI Express devices, which are designed for low power consumption and high performance, compared to corresponding PCI devices. We propose dynamic power management mechanism and a management policy for energy-efficient considerations. A case study for a variable-bit-rate LAN device under the PCI Express specification is exploited to provide supports for dynamic packet transmission. Simulation results show that the proposed mechanism and policy would reduce the system energy consumption substantially.

1 Introduction

The designs of high-performance hardware have always been in a strong demand in the past decades. The performance of microprocessors has been improved dramatically, and the improvement process continues for the following foreseeable future. Recently, the needs of energy efficiency in various system components trigger the exploring of the trade-off between the system performance and the energy consumption. Different techniques in dynamic power management (DPM) [11], dynamic voltage scaling (DVS) [21], and dynamic cache re-sizing are proposed in different contexts and for different applications. DPM aims at the reducing of the power consumption dynamically by changing the system state, and DVS changes the supply voltage of the electronic circuits dynamically for considerations of energy-efficiency.

Energy-efficient real-time scheduling has been an active research topic in the past decade for DVS systems. Researchers have proposed various scheduling algorithms to minimize the energy consumption for periodic hard real-time tasks under different assumptions, e.g., [1, 2, 14]. When fixed-priority scheduling is considered, various energy-efficient scheduling algorithms were proposed based on heuristics [20, 22, 23, 25, 29]. When energy-efficient scheduling of aperiodic real-time tasks is considered, energy-efficient scheduling for uniprocessor environments with a continuous speed spectrum was explored in [3, 8, 27, 28]. Scheduling algorithms were also proposed in the minimization of the energy consumption when there is a finite number of speeds for a processor with negligible speed transition overheads [5–7, 9, 10].

Under DPM, a device must be in the active state to serve requests, and it might go into the idle or sleep state to save energy. Requests might be issued by applications or respond

to external events, such as the arrival of network packets. Many works on power management mainly focus on the prediction of the duration of each idle period and often assumes that the arrival times of requests cannot be changed [11, 13]. However, the duration of an idle period can be changed by the scheduling (or even delaying) of requests in reality. A common approach is to cluster several short idle periods into a long one such that a device with DPM support could be idle or sleep for a long period of time. There have been some excellent results proposed for processor DPM support, such as those in [12, 19, 26], or for the considerations of real-time task scheduling, such as those in [4, 24]. Existing works mostly consider single service provider or device, e.g., a processor, where how to extend them to the support of multiple devices or a real device is not clear.

This paper explores energy efficiency for systems equipped with Peripheral Component Interconnect (PCI) Express devices, which are designed for low power consumption and high performance, compared to PCI devices [17]. Note that the PCI Express (PCI-E) interface provides control mechanism of functions and parameters for PCI-E devices, such as the supply voltage, the load capacitance, the frequency, and the transfer link [16]. In this work, we propose dynamic power management mechanism for considerations of energy-efficiency. A greedy algorithm for on-line scheduling is proposed to facilitate the power management for a device by re-ordering requests and by reducing the numbers of bitrate changes. We show how to integrate the proposed algorithm and mechanism into existing system implementations. A case study is exploited for a variable-bit-rate local-area-network (LAN) device under the PCI Express specification to provide supports for dynamic packet transmission. The proposed algorithms were evaluated by extensive simulations over networking traces. The experimental results show that the proposed mechanism and policy would reduce the system energy consumption substantially.

The rest of this paper is organized as follows: Section 2 presents the system architecture. Section 3 provides the motivation of this work and define the problem, following the mechanism and the policy in our energy-efficient design for variable-bitrate devices. Section 4 presents the simulation results. Section 5 is the conclusion.

2 System Architecture

2.1 PCI and PCI-Express Specifications

The Peripheral Component Interconnect (PCI) Local Bus is a high-performance 32-bit or 64-bit bus with multiplexed address and data lines. The bus is used for interconnection between highly integrated peripheral controller components, peripheral add-in cards, processors, and memory systems. In the PCI Local Bus Specification, Revision 2.1 [17], states are defined for all PCI functions, i.e., D0, D1, D2, or D3hot. Although, state transition and conditions of power management are defined in the PCI Bus Power Management Interface Specification [15], how to achieve energy efficiency in the hardware (or even software) implementation is unclear in the specification.

PCI defines a device as a physical load on the PCI bus. Each PCI device can host multiple functions, and each device has its own PCI Configuration Space. Since each PCI function is an independent entity to the software, each function must implement its own power management interface. Each PCI function can be in one of four power management

states, i.e., D0, D1, D2, and D3. As defined in the PCI Local Bus Specification, Revision 2.1, all PCI functions must support states D0, D3hot, and D3cold. Power management states provide different levels of power savings, and each state is denoted by a state number. Note that D1 and D2 are optional power management states. These intermediate states are intended to provide system designers more flexibility in balancing power saving, restore time, and performance. For example, the D1 state would consume more energy than the D2 state; however, the D1 state does provide a quicker restore time, compared to the D2 state. The D3 state is belonging to a special category in power management, and a PCI function could be transitted from any state into D3 by a command issued by software code or an action, due to the physical removing of the power from its host PCI device.

The PCI-Express (PCI-E) specification was designed to trade performance for energy consumption. PCI-E adopts control mechanism of functions to do power management. According to the system workload and performance metrics, a PCI-E device might dynamically adjust its supply voltage, transfer link, or frequency to satisfy the system requirements. To apply DPM to a PCI-E device, a power manager (PM) is required in the system to decide the state changes of the device. PM wakes up a device to serve requests and shuts it down to save power. However, any state transition incurs overheads in both energy consumption and latency. Consequently, a device should be shut down only if it can sleep long enough to compensate the performance and energy overhead.

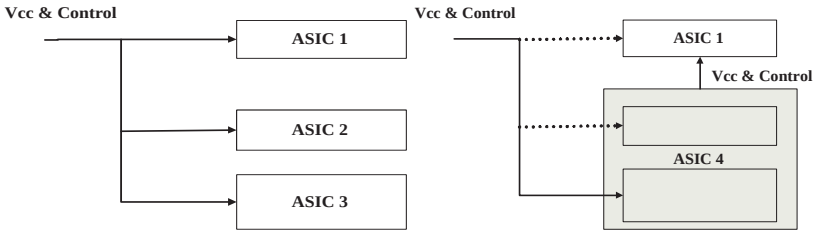
In particular, PM provides the following services [16]:

1. Mechanism to identify power management capabilities of a given function.
2. The ability to turn a function into a certain power management state.
3. Notifications of the current power management state of a function.
4. The option to wake up the system on a specific event.

In addition to the power management of functions, PM also provides Link power management so that the PCI-E physical link could let a device get to an active state, i.e., an initial state, or enable state transition. PCI-E Link states are not visible directly to legacy bus drivers but are derived from the power management states of the components residing on those links. The link states defined in the PCI-E specification are L0, L0s, L1, L2, and L3. The larger the subscript is, the more the power saving. PCI-E components are permitted to wake up the system by using wake-up mechanism, followed by a power management event (PME) message. Even when the main power supply of a device is turned off, a system with the corresponding PCI-E device might be waken up by providing the optional auxiliary power supply (Vaux) needed for the wake-up operation.

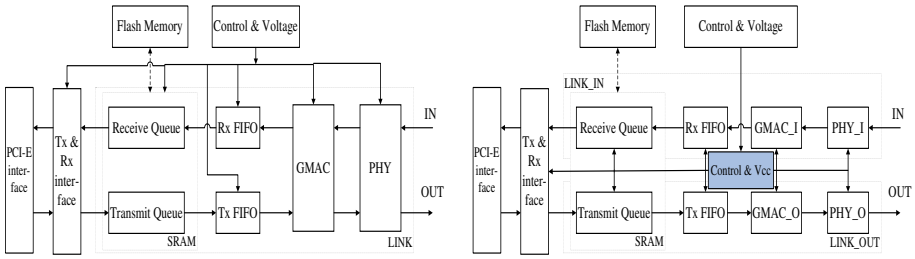
2.2 Variable-Bitrate PCI-Express LAN Devices

A system device is, in general, an integration of several application-specific integrated circuits (ASICs). In chip designs, the supply voltage (Vcc) usually supplies voltage to each component or function, as shown in Figure 1(a), where one purpose in the combination of ASICs is to reduce power consumption [18]. ASIC2 and ASIC3 might be merged or redesigned into an integration circuit (IC) because of changes in the design. For example, when several passive units, such as ASIC1 and the rest in Figure 1(a), have some dependent relationship or control sequence, the supply voltage circuits can be changed, as



(a) Three ASICs before integration (b) Integration with new Vcc support

Fig. 1. Low-Power ASICs designs.



(a) A normal block diagram for PCI-E (b) Separation of IN/OUT into two LINK components

Fig. 2. The block diagram of new LAN devices in which all control and voltage supply belong to the ”Control & Vcc” unit.

shown in Figure 1(b), in which the supply voltage of ASIC1 comes from the integrated IC (ASIC4) of ASIC2 and ASIC3.

This paper revises the existing architecture of LAN card devices based on the PCI-E specification [16]. It should not only manage state transition for power management but also save power. A typical design block diagram of a PCI-E LAN device is shown in Figure 2(a). From right to left in the IN port in Figure 2(a), the components are the physical layer (PHY), the global media access control (GMAC) layer, and the first-in-first-out reception buffer (Rx FIFO). PHY translates the protocol between the signal layer and the physical layer. The GMAC layer translates the protocol between different interfaces. On the other hand, from left to right in Figure 2(a) in the OUT port in Figure 2(a), we have a transmission queue and first-in-first-out transmission buffer (Tx FIFO). Our proposed architecture is shown in Figure 2(b). To reduce the power consumption of the GMAC and PHY layers, we design a control unit to control each function unit or component. Take the LAN card as an example: About a half (IN or OUT transport) of the power consumption is required when only one direction transmission occurs. There are two advantages in this architecture: First, a new control unit for Vcc supply is created, and different voltage supplies could be given to different units based on different needs (if the hardware is properly implemented), e.g., state/frequency changes. Secondly, the new architecture separates IN and OUT into two functions, such as LINK_IN and LINK_OUT in Figure 2(b).

The power consumption of the proposed variable bit-rate device (VBD) LAN device is summarized in Table 1. When the bit-rate of the device changes from 1000Mb (1Gb) to 100Mb, the state transition will take about 860 mJ. When the bit-rate of the device

LAN Speed (Megabit/Sec)	Transmission Mode	Current (mA)	Power (mW)
1000Mb (1Gb)	Normal Run (Functional Test)	350.9	1157
	Link Up (Idle)	314.5	1137
	Link Down	139.2	459
100Mb	Normal Run (Functional Test)	147.7	487
	Link Up (Idle)	131.8	434
	Link Down	120.5	398
10Mb	Normal Run (Functional Test)	116.4	384
	Link Up (Idle)	95.3	314
	Link Down	87.5	298

Table 1. Parameters in different LAN bit-rate settings.

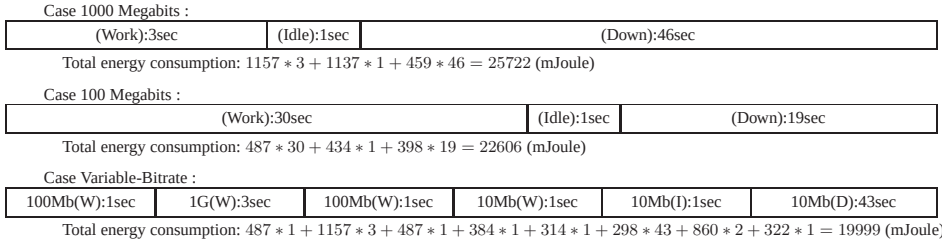


Fig. 3. An example in the transmissions of 300 MB of data in 50 seconds by a PCI-E LAN device.

changes from 100Mb to 10Mb, state transition will take about 322 mJ. The state transition overhead between different D states could be considered negligible, since it takes less than 10 mW power consumption with negligible timing overhead.

3 An Energy-Efficient Design for Variable-Bitrate PCI-E Devices

This section shows an energy-efficient design for variable-bitrate devices. we first describe the problem definition and provide an example. Secondly, we propose mechanism for state transition on the variable-bitrate device. Thirdly, we design a policy to make variable-bitrate device work properly.

3.1 Problem Definition

Suppose that we are required to transmit 300Mb of data from a host computer via a PCI-E LAN device to another computer in 50 seconds. Let the actual data transmission rate in the networking environment be 0.1 times of the network transmission bit-rate.

There are several alternatives in executing the data transmission: We might choose to transmit data in 1Gb, as shown in the first item in Figure 3. Three seconds are used to transmit data, and one second is used to have state transition of the device to the idle state. The rest of 46 seconds is for the device to stay at the ”Down” state. Another alternative is to transmit data at 100Mb for 30 seconds and then let the device go into the idle state. The device would stay at the ”Down” state for the rest 20 seconds, as shown in the second item in Figure 3. The other alternative is intelligently exploit the flexibility in the switching of

bit-rates. For example, we could do bit-rate adjustments, as shown in the third item in Figure 3. In terms of the energy consumption, the third alternative is the best among the presented alternatives, and the first is the worst. The third could save more than 20% of the total energy consumption, compared to the first. More than 10% saving of the total energy consumption could be achieved by the third alternative, compared to the second case. Note that it is not feasible to transmit the data at 10Mb because we could not finish it in 50 seconds. The example shows the advantage of adaptive adjustments of transmission bit-rates in energy consumption and provides a motivation for our work. Note that it is infeasible to have an optimal schedule unless the future is predictable.

This paper explores the management of state transition and transmission-bit-rate adjustments for the scheduling of requests. Each request to the device under considerations is characterized by three parameters: its Input/Output type, start-time, and request size. Our objective is to minimize the energy consumption in servicing the requests such that the task response time is acceptable. In the following subsections, we shall propose state transition mechanism based on existing system implementations and the PCI-E specification. We will then propose a policy in the management of state transition and transmission-bit-rate adjustments with the considerations of the scheduling of requests.

3.2 A Time-Slice-Based Transition Algorithm: The Basic Approach

The main data structure in the variable-bitrate driver is a queue. When a new request arrives, the request is inserted into the queue with the specification of its own transmission direction, starting time, and request size. This queue will be processed by applying the shortest-job-first (SJF) order for better performance since it tends to minimize the average response time of requests. Each request is associated with a status variable to record its service status. A request is removed from the queue after its service is completed.

We exploit the idea of *time slice* for the servicing of requests to a variable bit-rate device. The operating time of a device is divided into fixed time slices (of a specified length T) such that both the bit-rate and the power management state of the device are required to remain unchanged within each time slice. The rationale behind the time-slice idea is to reduce the number of bit-rate switchings to save energy consumption when requests are interleaved with short inter-arrival time. Another incentive is to keep the device working when some request finishes before the expiration of the time slice so that any immediately incoming request within the time-slice period would be serviced instantly.

Let D_c and F_c be the device state and the bit-rate state of the device, respectively. F_b denotes the actual bit-rate in the previous time slice. Initially, let $D_c = D0$, F_c =100Mb, and F_b =1Mb, regardless of what the network transmission bit-rate is. At the starting of each time slice, F_c and D_c are set as the actual device transmission bit-rate and the device state in the previous time slice, respectively, and $F_b = F_c$. The device transmission bit-rate (referred to as the bit-rate) and the state is checked up, as shown in Algorithm 1. F_c could be one of the three bit-rates 10Mb, 100Mb, and 1000Mb. D_c could be one of the three states: D0 (working state), D1 (idle state), and D3 (link down state, also abbreviated as D_{min}). Given a variable bit-rate LAN device, let the minimum transmission bit-rate F_{min} and the maximum transmission bit-rate F_{max} be 10Mb and 1Gb, respectively.

We adopt a greedy algorithm to set up the bit-rate and the state of a device at the starting of each time slice. The basic idea is as follows: If the device is not working, and

Algorithm 1 A Time-Slice-Based Transition Algorithm

Input: (F_c, D_c, F_b)

Output: The setting of the state D_c and the bit-rate F_c for this time slice, where F_b is the bit-rate for the previous time slice

```
if the device is not working then
    if F_c > F_min then
        Downgrade F_c with one degree
    else
        if D_c > D_min then
            Downgrade D_c with one degree
else
    if (F_c can be upgraded with one degree and F_c < F_max) then
        Upgrade F_c with one degree
    else
        if the actual bit-rate in the previous time slice is less than F_b then
            Downgrade F_c with one degree
        else
            F_c remains
    F_b is set as the actual bit-rate in the previous time slice;
```

the current transmission bit-rate F_c is higher than the minimum bit-rate F_{min}, then we downgrade the bit-rate. If the current device state D_c is higher than the minimum device state D_{min}, then we turn the device into a deeper power saving state. On the contrary, if the device is working (in default, the device will recover to the $D0$ state), and the bit-rate could be upgraded, then we shall pull the device bit-rate to a higher level for the performance considerations. If the actual bit-rate in the previous time slice is less than F_b, then the upgrading of the bit-rate would not improve the performance. As a result, we downgrade F_c with one degree. When we downgrade (upgrade) F_c with one degree, we mean that we move down (up) the bit-rate to the next level of the available bit-rate settings. Similarly, when we downgrade (upgrade) D_c with one degree, we mean that we move down (up) the state to the next level of the available D-state settings.

3.3 A Revised Algorithm

The purpose of this subsection is to further improve the time-slice-based transition algorithm with the considerations of the bit-rate of the three time slices ahead of the current time slice: The revised version of the algorithm is referred to as the *variable bit-rate algorithm*. Let t denote the starting time of the current time slice. We shall determine the device state D_c and the bit-rate state F_c of the device. Let DR_x and AR_x denote the set bit-rate and the actual bit-rate of the device in the x-th time slice ahead of the current time slice, respectively, as shown in Figure 4. Note that even if we set the bit-rate of a device at a value, the actual bit-rate might be lower because the device and the environment might not allow such as a bit-rate. The variable bit-rate algorithm is a greedy algorithm based on the idea of the time-slice-based transition algorithm.

The rules in the upgrading and downgrading of the bit-rate for the variable bit-rate algorithm is defined as follows:

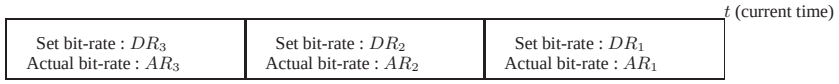


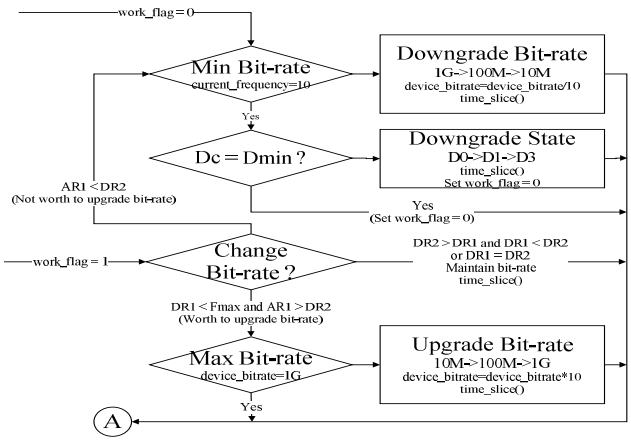
Fig. 4. The notations of the bit-rates of the time slices ahead of the current time slice.

1. Downgrade the transmission bit-rate F_c if any of the following two conditions is satisfied:
   – The device is not working, and $DR_1 >$ minimum transmission bit-rate.
   – The device is working, and $AR_1 < DR_2$.
2. Downgrade the device state D_c if both of the following two conditions are satisfied:
   – The device is operating at the minimum transmission bit-rate.
   – The current state is over the minimum device state.
3. Upgrade the transmission bit-rate F_c if the following condition is satisfied:
   – The device is working, DR_1 is lower than the maximum device bit-rate, and $AR_1 > DR_2$.
4. Upgrade the device state D_c if the following condition is satisfied:
   – The device is not working, but a new request arrives.
5. The state and the bit-rate of the device remain as the same as their corresponding ones in the previous time slice if any of the following two conditions is satisfied:
   – The device is working, $DR_2 > DR_3$, and $DR_1 < DR_2$.
   – The device is working, and $DR_2 = DR_1$.

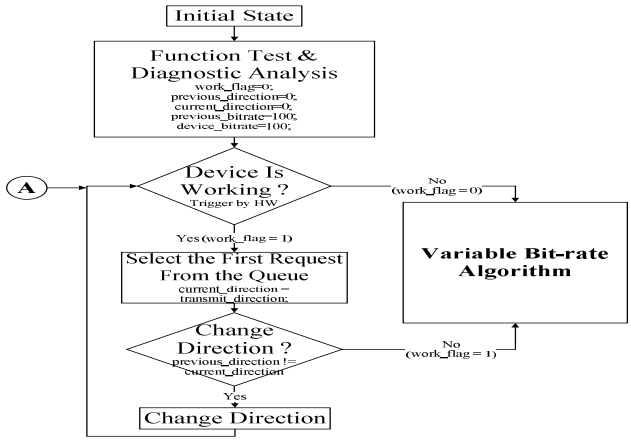
The flowchart of the rules is illustrated in Figure 5(a), where $work\_flag = 0$ means that the device is not working; otherwise, it is working.

The operating of the device is shown in Figure 5(b). The device starts at the initial state and does various function test and diagnostic analysis. Flags $work\_flag$, $previous\_direction$, $current\_direction$, $previous\_bitrate$, and $device\_bitrate$ denote the working status (i.e., working or not working), the status of the previous time slice (i.e., read or write), the status of the current time slice (i.e., read or write), the set bit-rate of the previous time slice, and the set bit-rate of the current time slice, respectively. If the device is not working, then call the variable-bit algorithm; otherwise, the request at the front of the queue is selected for data transmission. $current\_direction$ is set as the transmission direction of the request, i.e., read or write. If the direction is not changed, then invoke the variable-bit algorithm; otherwise, the transmission direction is changed by a hardware setting action. Note that circuits for "IN" and "OUT" are separated, as shown in Figure 2(b). The hardware setting action would activate a different circuit and de-activate the original circuit for the service of the previous request. The selected request will be executed by the newly activated circuit. The entire operating of the device will go back to the checking state of the device working status.

The variable bit-rate algorithm provides a framework for the adjustment of the state and the bit-rate of a device. The algorithm could be further improved by considering the tradeoff between the power consumption and the required bit rate. Take Table 1 as an example. The first column of the table shows the three available bit-rates of a variable-bit-rate LAN device, and the second column shows the three available states for each bit-rate, e.g., the *Normal Run* state being as $D0$ of the bit-rate 1000Mb. The third column and the fourth column show the current and the power of each corresponding state for a given bit-rate. We should further improve the conditions in the upgrading and downgrading of



(a)



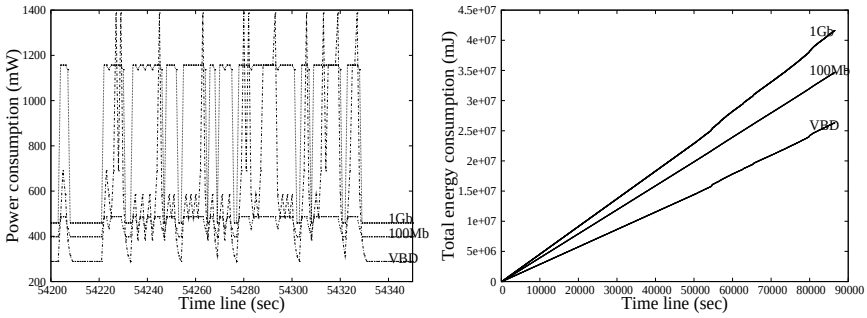
(b)

Fig. 5. The flowchart of the variable bit-rate algorithm.

states/bit-rates by considering the tradeoff between the energy consumption and the bit-rate, i.e., the performance. For example, since the power ratio between 1Gb and 100Mb at the *Normal Run* state is 2.376, there is no point to move to 1Gb from 100Mb if the transmission bit-rate required for a transmission does not need to be 2.376 times faster. Another consideration is on the limitation on the actual transmission bit-rate in the reality. When a device could not reach the transmission bit-rate as being set by the algorithm, the maximum transmission bit-rate should be set accordingly. Such a setting action could be done dynamically as the policy requires, e.g., once per few hours.

4 Performance Evaluation

The proposed algorithm was evaluated over a trace collected at an FTP server for two weeks. The arrival times of transmission requests were translated into their start times in



(a) Power consumption (b) Energy consumption

Fig. 6. Power consumption and energy consumption in different bit-rate settings.

the trace. The range of time slices varied from one second to five seconds. The power consumption values of a bit-rate transition, a D state transition, and data transmissions are as shown in Section 2. Three different strategies were simulated: Setting of the transmission bit-rate fixed at 1Gb but with possible state transitions (denoted to as 1Gb), Setting of the transmission bit-rate fixed at 100Mb but with possible with state transition (denoted to as 100Mb), and our proposed variable-bit-rate algorithm (denoted to as VBD).

Figure 6(a) shows the power consumption under the 1Gb strategy, the 100Mb strategy, and the VBD strategy with a one-second time slice. As astute readers might point out, the 1Gb strategy always had a larger power consumption for most of the time, compared to other strategies. With the VBD strategy, the power consumption was usually smaller, but there did exist some peaks in the experiments because of switchings of the bit-rate. (Note that the power consumption in the switching of states was negligible.) Figure 6(b) shows the total energy consumption of the three strategies under comparisons. The VBD strategy clearly outperformed the other two strategies. The relationship between the energy consumption and the time line was almost linear for each of the three strategies although they had different slopes. The gap between the VBD strategy and others was getting bigger as time went by. By the end of the experiments, the VBD strategy could save roughly 30% of the energy consumption, compared to the 1Gb strategy. Compared to the 100Mb strategy, the VBD strategy could save roughly 15% of the energy consumption.

Figure 7 shows the experimental results of the VBD strategy by varying the duration of a time slice from one second to five seconds. In the experiments, the VBD strategy with three-second time slice is better than that with others. The difference between any two of the total energy consumption of the five lines was, in fact, less then 1%. The determination of time-slice durations in the experiments was done by a series of experiments and observations. We found that the durations adopted in the experiments were the best for the trace under simulation. However, we must point out that a bad decision for a time-slice duration would not ruin the proposed VBD strategy too much. It was based on the observation in which the performance of the VBD strategy did not change a lot for the five durations. Even if the duration was set as infinity, the VBD strategy became the 1Gb strategy. The determination of time slice could be determined by profiling tools.

In general, the VBD strategy payed the price at a worse response time, compared to the 1Gb strategy. The average response time of the VBD strategy with different durations of a

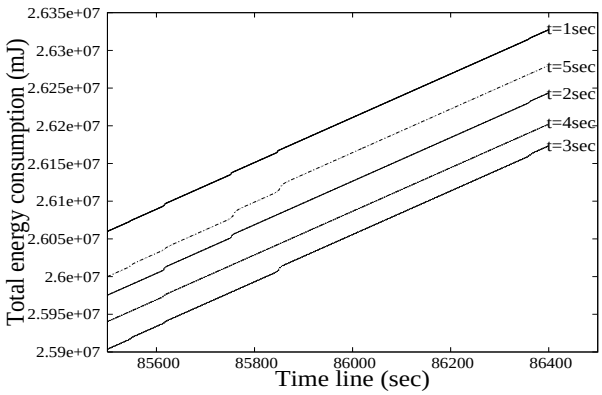


Fig. 7. The energy consumption of the VBD strategy in different setttings of time slices.

Time slice (second)	N/A(1Gb)	1 second	2 second	3 second	4 second	5 second
Average Response Time (sec)	0.206	0.646	0.653	0.658	0.664	0.679

Table 2. The average response time in different durations of a time slice.

time slice and the 1Gb strategy are shown in Table 2. Although the average response time of the VBD strategy was worse than that of the 1Gb strategy, the delay in the transmission of a file was not bad because the delay was only for the transmission of the last piece of the file (when a file was broken into pieces for transmissions).

5 Conclusion and Future work

In this paper, we design a prototype of a variable-bit-rate local-area-networking device over the PCI Express specification. A case study is done over a variable-bit-rate local-area-networking (LAN) device under the PCI Express specification in energy-efficient designs. A greedy on-line scheduling algorithm is developed to minimize the energy consumption with tolerable performance degradation. We propose the concept of time slice to adjust the transmission bit-rate or the idle time of the device. A feasible mechanism is presented based on the implementations of existing systems. The proposed algorithm and mechanism were evaluated by simulations over emulated devices. The experimental results show that the proposed algorithm could reduce from 15% to 30% energy consumption roughly, compared to a typical PCI-E LAN card with normal PM functionality. The increasing of the average response time of requests was reasonable in the experiments.

Energy efficiency has been a highly critical design issue in hardware and software designs. For the future work, we shall further extend the static time-slice approach to a dynamic one to further improve the power saving of the system. The concept and methodology proposed in this work could also be extended to the energy-efficient management designs of complicated devices, such as many VGA, USB, ATAPI and SATA devices. Such management designs could be implemented by either software or hardware, and there is always a tradeoff in terms of cost and performance.

References

1. H. Aydin, R. Melhem, D. Mossé, and P. Mejía-Alvarez. Determining optimal processor speeds for periodic real-time tasks with different power characteristics. In *Proceedings of the IEEE EuroMicro Conference on Real-Time Systems*, pages 225–232, 2001.
2. H. Aydin, R. Melhem, D. Mossé, and P. Mejía-Alvarez. Dynamic and aggressive scheduling techniques for power-aware real-time systems. In *Proceedings of the 22nd IEEE Real-Time Systems Symposium*, pages 95–105, 2001.
3. N. Bansal, T. Kimbrel, and K. Pruhs. Dynamic speed scaling to manage energy and temperature. In *Proceedings of the Symposium on Foundations of Computer Science*, pages 520–529, 2004.
4. J. J. Brown, D. Z. Chen, G. W. Greenwood, X. Hu, and R. W. Taylor. Scheduling for power reduction in a real-time system. *In International Symposium on Low Power Electronics and Design*, pages 84–87, 1997.
5. J.-J. Chen and T.-W. Kuo. Procrastination for leakage-aware rate-monotonic scheduling on a dynamic voltage scaling processor. In *ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES)*, pages 153–162, 2006.
6. J.-J. Chen, T.-W. Kuo, and H.-I. Lu. Power-saving scheduling for weakly dynamic voltage scaling devices. In *Workshop on Algorithms and Data Structures (WADS)*, pages 338–349, 2005.
7. J.-J. Chen, T.-W. Kuo, and C.-S. Shih. $1+\epsilon$ approximation clock rate assignment for periodic real-time tasks on a voltage-scaling processor. In *the 2nd ACM Conference on Embedded Software (EMSOFT)*, pages 247–250, 2005.
8. S. Irani, S. Shukla, and R. Gupta. Algorithms for power savings. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 37–46. Society for Industrial and Applied Mathematics, 2003.
9. T. Ishihara and H. Yasuura. Voltage scheduling problems for dynamically variable voltage processors. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 197–202, 1998.
10. W.-C. Kwon and T. Kim. Optimal voltage allocation techniques for dynamically variable voltage processors. In *Proceedings of the 40th Design Automation Conference*, pages 125–130, 2003.
11. L.Benini, A.Bogliolo, and G.D.Micheli. A survey of design techniques for system-level dynamic power management. *IEEE Transactions on VLSI Systems*, 2000.
12. J. R. Lorch and A. J. Smith. Scheduling techniques for reducing processor energy use in macos. *Wireless Networks*, pages 311–324, 1997.
13. Y.-H. Lu, E.-Y. Chung, T. Simunic, L. Benini, and G. D. Micheli. Quantitative comparison of power management algorithms. *In Design Automation and Test in Europe*, 2000.
14. P. Mejía-Alvarez, E. Levner, and D. Mossé. Adaptive scheduling server for power-aware real-time tasks. *ACM Transactions on Embedded Computing Systems*, 3(2):284–306, 2004.
15. PCI BUS POWER MANAGEMENT INTERFACE SPECIFICATION 1.1, December 1998.
16. PCI EXPRESS BASE SPECIFICATION 1.0A, April 2003.
17. PCI LOCAL BUS SPECIFICATION 2.3, March 2002.
18. PUTTING IT ALL TOGETHER:INTELS WIRELESS-INTERNET-ON-A-CHIP, June 2001.
19. G. Qu and M. Potkonjak. Power minimization using system-level partitioning of applications with quality of service requirements. *In ICCAD*, pages 343–346, 1999.
20. G. Quan and X. Hu. Minimum energy fixed-priority scheduling for variable voltage processor. In *Proceedings of the Design Automation and Test Europe Conference*, pages 782–787, 2002.
21. J. M. Rabaey, A. Chandrakasan, and B. Nikolic. *Digital Integrated Circuits*. Prentice Hall, 2nd edition, 2002.
22. D. Shin, J. Kim, and S. Lee. Low-energy intra-task voltage scheduling using static timing analysis. In *Proceedings of the 38th Conference on Design Automation*, pages 438–443. ACM Press, 2001.
23. Y. Shin and K. Choi. Power conscious fixed priority scheduling for hard real-time systems. In *Proceedings of the 36th ACM/IEEE Conference on Design Automation Conference*, pages 134–139. ACM Press, 1999.
24. Y. Shin and K. Choi. Power conscious fixed priority scheduling for hard real-time systems. *In DAC*, pages 134–139, 1999.
25. Y. Shin, K. Choi, and T. Sakurai. Power optimization of real-time embedded systems on variable speed processors. In *Proceedings of the 2000 IEEE/ACM International Conference on Computer-Aided Design*, pages 365–368. IEEE Press, 2000.
26. M. Weiser, B. Welch, A. Demers, and S. Shenker. Scheduling for reduced cpu energy. *In Symposium on Operating Systems Design and Implementation*, pages 13–23, 1994.
27. C.-Y. Yang, J.-J. Chen, and T.-W. Kuo. Preemption control for energy-efficient task scheduling in systems with a DVS processor and Non-DVS devices. In *the 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2007.
28. F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 374–382. IEEE, 1995.
29. H.-S. Yun and J. Kim. On energy-optimal voltage scheduling for fixed-priority hard real-time systems. *ACM Transactions on Embedded Computing Systems*, 2(3):393–430, Aug. 2003.