Energy-Efficient Considerations on a Variable-Bitrate PCI-Express Device

J Sign Process Syst (2010) 59:57–69
DOI 10.1007/s11265-008-0280-9

Energy-Efficient Considerations on a Variable-Bitrate PCI-Express Device

Yung-Hen Lee · Jian-Jia Chen · Chi-Sheng Shih

Received: 26 March 2008 / Revised: 10 September 2008 / Accepted: 11 September 2008 / Published online: 1 November 2008

Abstract Dynamic power management has been adopted in many systems to reduce the power/energy 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 local-area-network 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.

Keywords PCI-express devices · Energy-efficient · Variable-bitrate devices · Dynamic power management

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 tradeoff between the system performance and the energy consumption. Different techniques in dynamic power management (DPM) [13], dynamic voltage scaling (DVS) [24], 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, 11, 17]. When fixed-priority scheduling is considered, various energy-efficient scheduling algorithms were proposed based on heuristics [23, 25, 26, 28, 32]. 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, 9, 14, 30, 31]. Scheduling

This work was done when Mr. Lee and Mr. Chen were students at National Taiwan University.

Y.-H. Lee
Dell Corporation, Taipei, Taiwan
e-mail: Henry_Lee@Dell.com

J.-J. Chen (✉)
Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology Zurich (ETH Zurich), Zurich, Switzerland
e-mail: jchen@tik.ee.ethz.ch

C.-S. Shih
Department of Computer Science and Information Engineering and Graduate Institute of Networking and Multimedia, National Taiwan University, Taipei, Taiwan
e-mail: cshih@csie.ntu.edu.tw

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 [6–8, 10, 12]. Recent study by Chen and Kuo [5] provides a comprehensive survey for energy-efficient scheduling of real-time tasks in DVS systems.

Distinct from the DVS technique to adjust the supply voltages, the DPM technique changes the system state dynamically to reduce the power/energy consumption. In DPM, a device must be in the active state to serve requests, and it might go to 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 [13, 16]. 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 [15, 22, 29], or for the considerations of real-time task scheduling, such as those in [4, 27].

In reality, most embedded systems and server systems are with peripheral devices. For battery-driven embedded systems, the reduction of energy consumption can prolong the lifetime of the battery. For server systems, the effective use of energy/power can significantly reduce the power bills. As peripheral devices often make a significant contribution to the power consumption of the entire system, it is necessary to reduce the energy consumption without sacrificing too much performance. Peripheral Component Interconnect (PCI) Express devices are designed for low power consumption and high performance, compared to PCI devices [20]. This paper explores how to achieve energy efficiency for systems equipped with PCI Express devices. By applying the interfaces provided by PCI-Express for control mechanism of functions and parameters, such as the supply voltage, the load capacitance, the frequency, and the transfer link, system designers can balance the energy/power consumption and performance [19].

In this work, we propose DPM 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. As a result, adopting the variable-bite-rate mechanism can significantly improve the energy consumption, compared to the schedule by staying at the maximum bit-rate, with slight increasing of response time.

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

The purpose of this section is to provide a summary of the PCI and PCI-Express specifications and the architecture of a system equipped with a variable-bitrate PCI-Express LAN device. The driver layers of an I/O system in a typical operating system could be abstracted as those shown in Fig. 1. The variable bit-rate device (VBD) driver provides a communication interface between a targeted PCI-Express device and the operating system. Driver-Host is a layer that provides a communication interface between application and the VDB driver, and Driver-Target is a layer for the communications between Driver-Host and a PCI-Express device via the PCI-Express local bus. In this work, we revise the two existing layers in the VBD driver, i.e., Driver-Host and Driver-Target. The objective is to propose a low-power and variable-bitrate design for a typical PCI-Express LAN device.

2.1 PCI and PCI-Express Specifications

The 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 [20], 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 [18], how to achieve

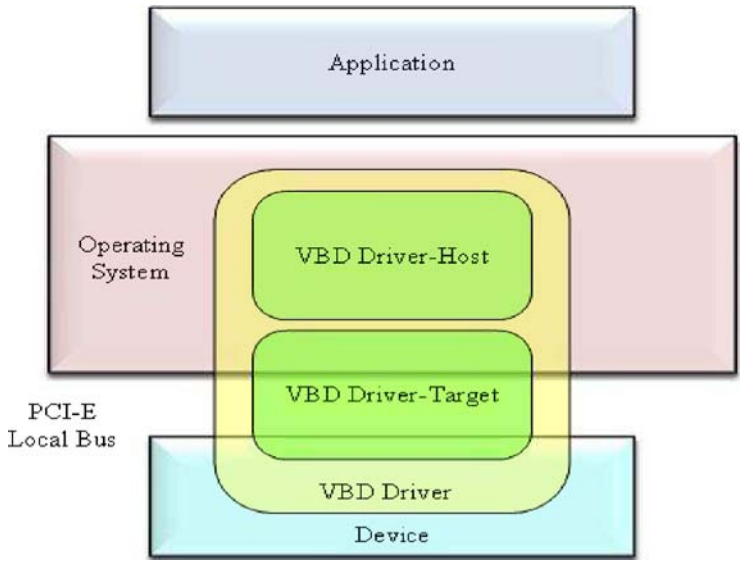


Figure 1 The overview of a variable-bitrate driver between the operating system and devices.

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 transmitted 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. Because of the two different transitions, the two new D3 states are designated as D3hot and D3cold, where the subscripts refer to the presence or absence of Vcc, respectively. Functions in D3hot can be transmitted to an uninitialized D0 state via software by writing to the function's PMCSR register or by having its Bus Segment Reset (PCI RSTpin) asserted. Functions in the D3cold state can only be transmitted to an uninitialized D0 state by reapplying Vcc and by asserting Bus Segment Reset (RSTpin) to the function's host PCI device.

The PCI-Express specification was designed to trade performance for energy consumption. PCI-Express adopts control mechanism of functions to do power management. According to the system workload and performance metrics, a PCI-Express device might dynamically adjust its supply voltage, transfer link, or frequency to satisfy the system requirements. To apply DPM to a PCI-Express 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 [19]:

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-Express physical link could let a device get to an active state, i.e., an initial state, or enable state transition. PCI-Express 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-Express specification are L0, L0s, L1, L2, and L3. The larger the subscript is, the more the power saving. PCI-Express components are permitted to wake up the system by using wake-up mechanism, followed by a power management event message. Even when the main power supply of a device is turned off, a system with the corresponding PCI-Express device might be waken up by providing the optional auxiliary power supply (Vaux) needed for the wake-up operation.

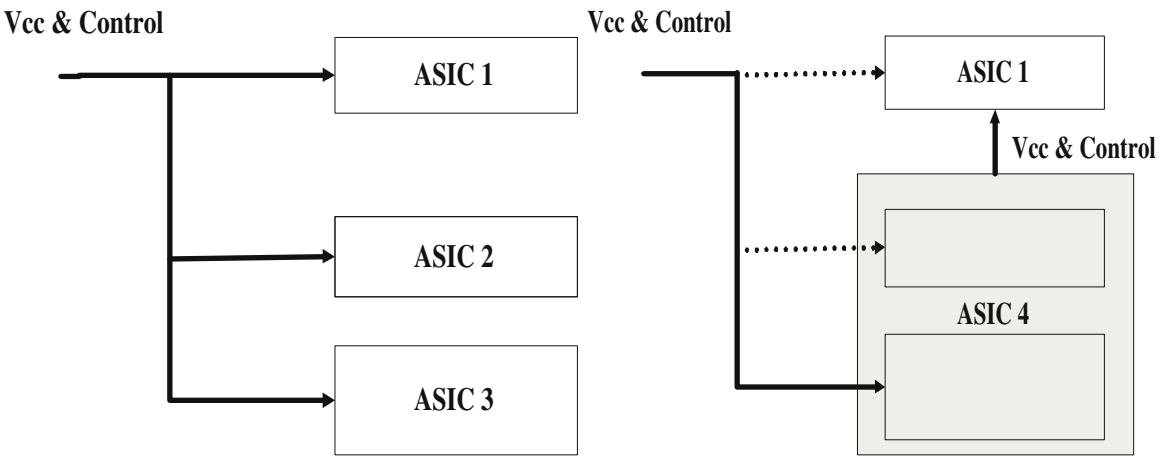
2.2 Variable-Bitrate PCI-Express LAN Devices

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 Fig. 2a, where one purpose in the combination of ASICs is to reduce power consumption [21]. 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 Fig. 2a, have some dependent relationship or control sequence, the supply voltage circuits can be changed, as shown in Fig. 2b, 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-Express specification [19]. It should not only manage state transition for power management but also save power. A typical design block diagram of a PCI-Express LAN device is shown in Fig. 3a. From right to left in the IN port in Fig. 3a, 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 PHY. The GMAC layer translates the protocol between different interfaces. On the other hand, from left to right in Fig. 3a in the OUT port in Fig. 3a, we have a transmission queue and first-in-first-out transmission buffer (Tx FIFO). Our proposed architecture is shown in Fig. 3b. 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 Fig. 3b.

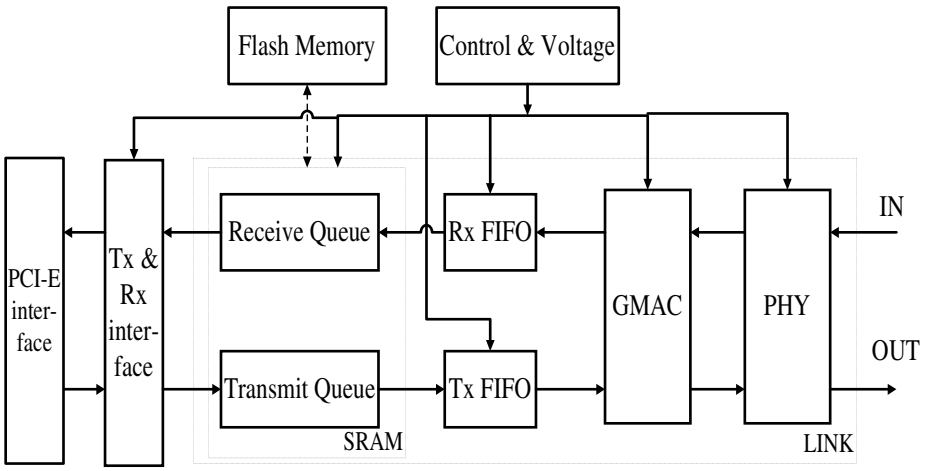
The power consumption of the proposed VBD LAN device is summarized in Table 1. When the bit-rate of the device changes from 1000 Mb (1 Gb) to 100 Mb, the state transition will take about 860 mJ. When the bit-rate of the device changes from 100 to 10 Mb, state transition will take about 322 mJ. The state transition overhead between different D states could be considered negligible, since it takes less than 10 mW power consumption with negligible timing overhead.

Even though PCI-Express LAN devices provide hardware mechanism for low power design, system designers have to determine how to dynamically adopt

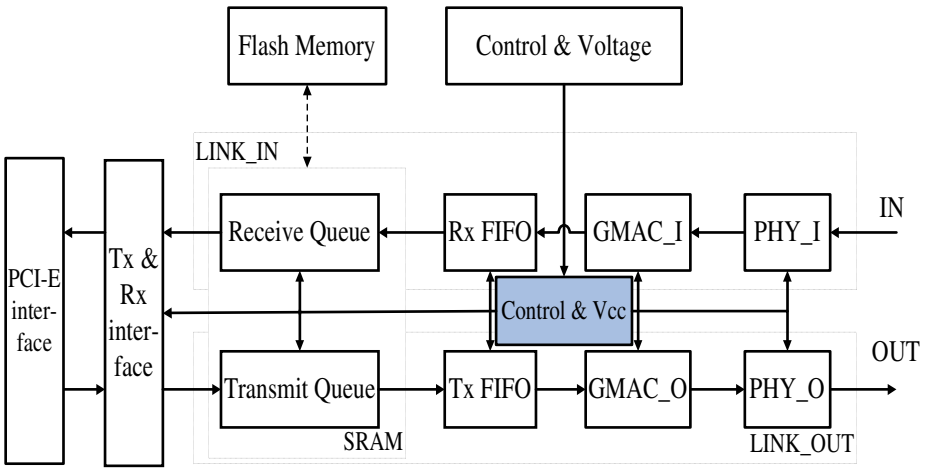


(a) Three ASICs before integration (b) Integration with new Vcc support

Figure 2 Low-power ASICs designs (**a**, **b**).



(a) A normal block diagram for PCI-Express



(b) Separation of IN/OUT into two LINK components

Figure 3 The block diagram of new LAN devices in which all control and voltage supply belong to the "Control & Vcc" unit (**a**, **b**).

the bit-rate changes or mode changes. For example, to reduce the energy, we can turn off the device when it is idle, but turning the device on later will consume energy. Therefore, the decision to change the mode or change the bit-rate mush be done carefully. Section 3 will present our proposed algorithms to decide when and how to change the bit-rate or mode.

3 An Energy-Efficient Design for Variable-Bitrate PCI-Express 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-

Table 1 Parameters in different LAN bit-rate settings.

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

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 300 Mb of data from a host computer via a PCI-Express LAN device to another computer in 50 s. 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 1 Gb, as shown in the first item in Fig. 4. Three seconds are used to transmit data, and 1 s is used to have state transition of the device to the idle state. The rest of 46 s is for the device to stay at the "Down" state. Another alternative is to transmit data at 100 Mb for 30 s and then let the device go into the idle state. The device would stay at the "Down" state for the rest 20 s, as shown in the second item in Fig. 4. 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 Fig. 4. 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 10 Mb because we could not finish it in 50 s. 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-Express 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 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 VBD. 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. $F_{b,1}$ denotes the actual

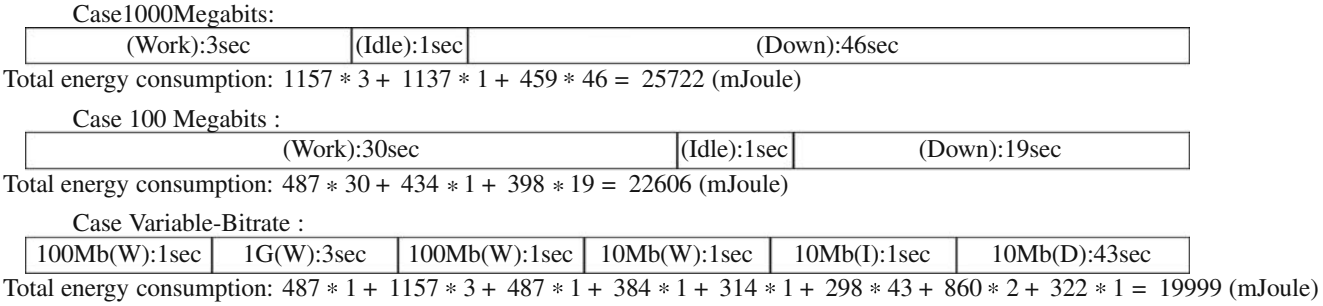


Figure 4 An example in the transmissions of 300 MB of data in 50 s by a PCI-Express LAN device.

bit-rate in the previous time slice, where $F_{b,2}$ is the actual bit-rate in the time slice before the previous time slice. Initially, let $D_c = D0$, F_c =100 Mb, and F_b =1 Mb, 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 10, 100, and 1000 Mb. 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.

Algorithm 1 A Time-Slice-Based Transition Algorithm

Input: $(F_c, D_c, F_{b,1}, F_{b,2})$
Output: The setting of the state D_c and the bit-rate F_c for this 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
    end if
  end if
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 F_{b,1} < F_{b,2} then
      Downgrade F_c with one degree
    else
      F_c remains
    end if
  end if
end if
```

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 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 $F_{b,1} < F_{b,2}$, 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 Fig. 5. 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 revised algorithm will refer to only three recent time slices, and is applied only when the previous three time slices at time t are in the *Normal Run*. Note that it is possible to refer to more than 3 time slices to make a decision for setting the bit-rates. However, checking the previous three time slices can report quite representative results to see the network situation at this moment. If the network is with low performance,

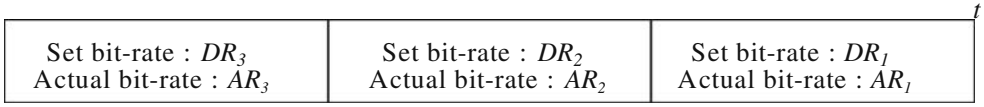


Figure 5 The notations of the bit-rates of the time slices ahead of the current time slice.

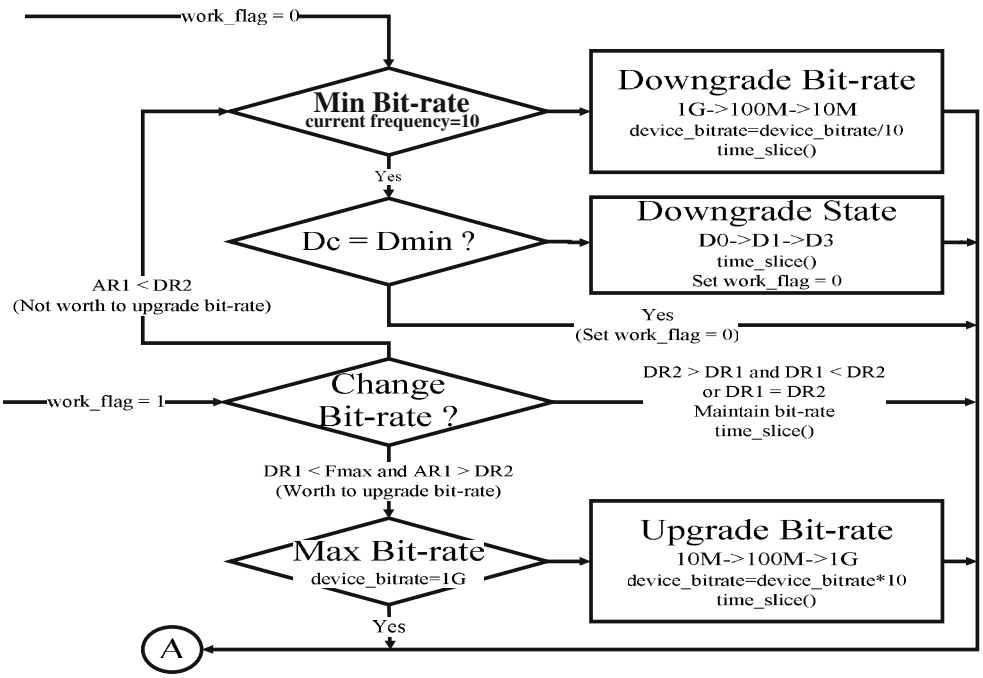


Figure 6 The flowchart of the variable bit-rate algorithm.

one should not use high bit-rate due to the significant energy/power waste. If more than three time slices are referred, the decision must be made carefully, as the transmission history might not be useful for the moment. Some factors, like aging, should be taken into considerations. Here, we will present our solution by referring three time slices.

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

1. The state and the bit-rate of the device remain as the same as their corresponding ones in the

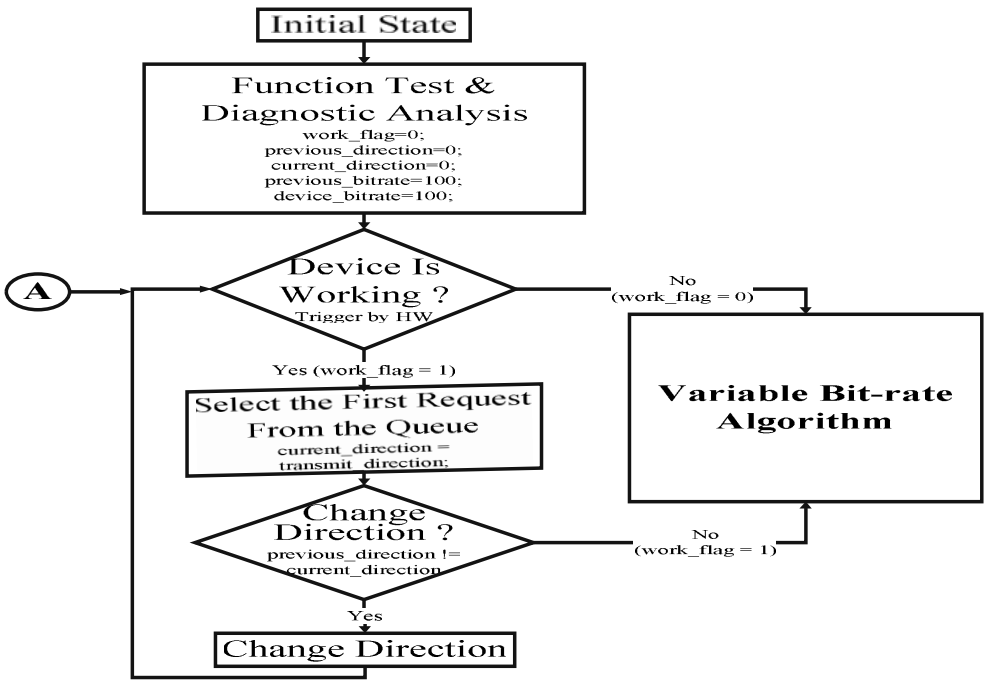


Figure 7 The flowchart of the variable bit-rate algorithm.

SB: 10Mb	SB: 100Mb	SB: 1000Mb	SB: 100Mb	SB: 100Mb	SB: 100Mb	SB: 100Mb
AB: 5Mb	AB: 30Mb	AB: 50Mb	AB: 25Mb	AB: 30Mb	AB: 30Mb	AB: 30Mb

Figure 8 An example for the revised algorithm, where SB and AB stand for the set bit-rate and the actual bit-rate of a time slice, respectively.

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 $DR_2 = DR_1$.

2. 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 $AR_1 < DR_2$.

3. 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.

4. 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 $AR_1 > DR_2$.

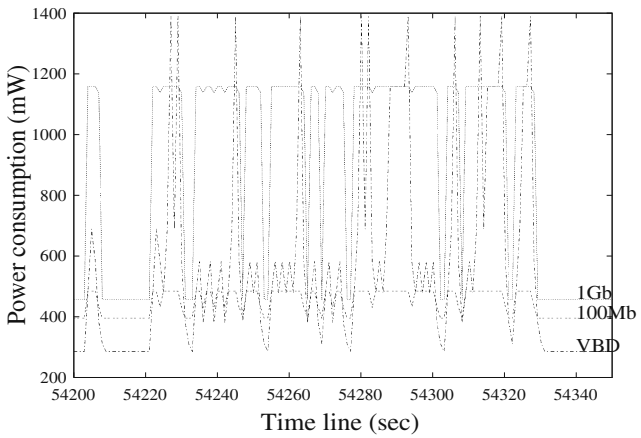
5. Upgrade the device state D_c if the following condition is satisfied:

The device is not working, but a new request arrives.

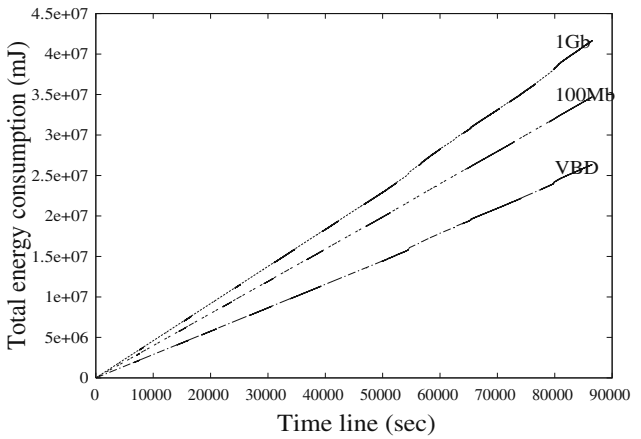
The flowchart of the rules is illustrated in Fig. 6, where $work\_flag = 0$ means that the device is not working; otherwise, it is working.

The operating of the device is shown in Fig. 7. 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 Fig. 3b, where terms of "frequency" and "bit-rate" are used for the same meaning. 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



(a) Power consumption



(b) Energy consumption

Figure 9 Power consumption and energy consumption in different bit-rate settings (**a**, **b**).

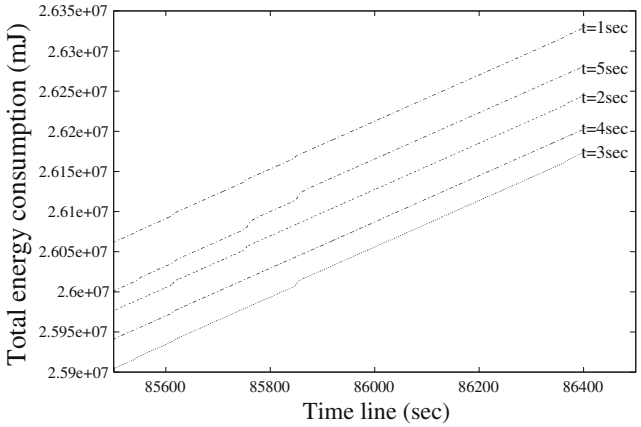


Figure 10 The energy consumption of the VBD strategy in different settings of time slices.

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 *D*0 of the bit-rate 1000 Mb. 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 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 1 Gb and 100 Mb at the *Normal Run* state is 2.376, there is no point to move to 1 Gb from 100 Mb 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.

To illustrate the above algorithm, we will use an example for demonstration, as shown in Fig. 8. Suppose that we are required to transmit 200 Mb of data from a host computer via a PCI-Express LAN device to another computer. The time slice is 1 s. We assume that at the beginning the device is in the Link Up mode at bit-rate 10 Mb. Therefore, the device is turned to the Normal Run at time 1 s with 10 Mb. The actual bit-rate for the device is 5 Mb. Then, at 2 s, we will change to bit-rate 100 Mb (with actual bit-rate 30 Mb) by Algorithm 1. Then, at 3 s, we will change to bit-rate 1000 Mb (with actual bit-rate 50 Mb) by Algorithm 1. As $50 < 100$ Mb, we change the bit-rate to 100 Mb (with actual bit-rate 25 Mb) at 4 s. At 5 s, as the first condition stands, for the rest of scheduling of the transmission, the device will be set to transmit at bit-rate 100 Mb with actual bit-rate 30 Mb.

4 Performance Evaluation

The proposed algorithm was evaluated over a trace collected at an FTP server for 2 weeks. The arrival times of transmission requests were translated into their start times in the trace. The range of time slices varied from 1 to 5 s. 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 1 Gb but with possible state transitions (denoted to as 1 Gb), Setting of the transmission bit-rate fixed at 100 Mb but with possible with state transition (denoted to as 100 Mb), and our proposed variable-bit-rate algorithm (denoted to as VBD).

Figure 9 shows the power consumption under the 1 Gb strategy, the 100 Mb strategy, and the VBD strategy with a 1-s time slice. As astute readers might point out, the 1 Gb 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 9b 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 1 Gb strategy. Compared to the 100 Mb strategy, the VBD strategy

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

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

could save roughly 15% of the energy consumption. The trace for the experiments were for 2 weeks!

Figure 10 shows the experimental results of the VBD strategy by varying the duration of a time slice from 1 to 5 s. In the experiments, the VBD strategy with 3-s 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 1 Gb 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 1 Gb strategy. The average response time of the VBD strategy with different durations of a time slice and the 1 Gb strategy are shown in Table 2. Although the average response time of the VBD strategy was worse than that of the 1 Gb 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-Express 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.

Acknowledgements We would like to thank Prof. Tei-Wei Kuo at National Taiwan University for his valuable inputs and the reviewers for their valuable feedbacks.

Moreover, this work is supported in part by a grant from the NSC program 96-2219-E-002-005, in part by a grant from the NSC program 95-2221-E-002-093-MY3, and in part by Excellent Research Projects of National Taiwan University, 97R0062-05.

References

1. Aydin, H., Melhem, R., Mossé, D., & Mejía-Alvarez, P. (2001). Determining optimal processor speeds for periodic real-time tasks with different power characteristics. In *Proceedings of the IEEE EuroMicro conference on real-time systems* (pp. 225–232).
2. Aydin, H., Melhem, R., Mossé, D., & Mejía-Alvarez, P. (2001). Dynamic and aggressive scheduling techniques for power-aware real-time systems. In *Proceedings of the 22nd IEEE real-time systems symposium* (pp. 95–105).
3. Bansal, N., Kimbrel, T., & Pruhs, K. (2004). Dynamic speed scaling to manage energy and temperature. In *Proceedings of the symposium on foundations of computer science* (pp. 520–529).
4. Brown, J. J., Chen, D. Z., Greenwood, G. W., Hu, X., & Taylor, R. W. (1997). Scheduling for power reduction in a real-time system. In *International symposium on low power electronics and design* (pp. 84–87).
5. Chen, J.-J., & Kuo, C.-F. (2007). Energy-efficient scheduling for real-time systems on dynamic voltage scaling (DVS) platforms. In *RTCSA* (pp. 28–38).
6. Chen, J.-J., & Kuo, T.-W. (2006). 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)* (pp. 153–162).
7. Chen, J.-J., Kuo, T.-W., & Lu, H.-I. (2005). Power-saving scheduling for weakly dynamic voltage scaling devices. In *Workshop on algorithms and data structures (WADS)* (pp. 338–349).
8. Chen, J.-J., Kuo, T.-W., & Shih, C.-S., (2005). $1+\epsilon$ approximation clock rate assignment for periodic real-time tasks on a voltage-scaling processor. In *The 2nd ACM conference on embedded software (EMSOFT)* (pp. 247–250).
9. Irani, S., Shukla, S., & Gupta, R. (2003). Algorithms for power savings. In *Proceedings of the fourteenth annual ACM-SIAM symposium on discrete algorithms* (pp. 37–46). Society for Industrial and Applied Mathematics.

10. Ishihara, T., & Yasuura, H. (1998). Voltage scheduling problems for dynamically variable voltage processors. In *Proceedings of the international symposium on low power electronics and design* (pp. 197–202).
11. Jejurikar, R., Pereira, C., & Gupta, R. (2004). Leakage aware dynamic voltage scaling for real-time embedded systems. In *Proceedings of the design automation conference* (pp. 275–280).
12. Kwon, W.-C., & Kim, T. (2003). Optimal voltage allocation techniques for dynamically variable voltage processors. In *Proceedings of the 40th design automation conference* (pp. 125–130).
13. Benini, L., Bogliolo, A., & Micheli, G.D. (2000). A survey of design techniques for system-level dynamic power management. *IEEE transactions on Very Large Scale Integration Systems, 8*, 299–316.
14. Li, M., & Yao, F. F. (2005). An efficient algorithm for computing optimal discrete voltage schedules. *SIAM Journal on Computing, 35*(3), 658–671.
15. Lorch, J. R., & Smith, A. J. (1997). Scheduling techniques for reducing processor energy use in macos. *Wireless Networks, 3*, 311–324.
16. Lu, Y.-H., Chung, E.-Y., Simunic, T., Benini, L., & Micheli, G. D. (2000). Quantitative comparison of power management algorithms. In *Design Automation and Test in Europe*.
17. Mejía-Alvarez, P., Levner, E., & Mossé, D. (2004). Adaptive scheduling server for power-aware real-time tasks. *ACM Transactions on Embedded Computing Systems, 3*(2), 284–306.
18. PCI Bus Power Management Interface Specification 1.1. (1998) December.
19. PCI Express Base Specification 1.0a. (2003) April.
20. PCI Local Bus Specification 2.3. (2002) March.
21. Putting it All Together: Intels Wireless-Internet-on-a-Chip. (2001) June.
22. Qu, G., & Potkonjak, M. (1999). Power minimization using system-level partitioning of applications with quality of service requirements. In *ICCAD* (pp. 343–346).
23. Quan, G., & Hu, X. (2002). Minimum energy fixed-priority scheduling for variable voltage processor. In *Proceedings of the design automation and test Europe conference* (pp. 782–787).
24. Rabaey, J. M., Chandrakasan, A., & Nikolic, B. (2002). *Digital integrated circuits* (2nd ed.). Englewood Cliffs: Prentice Hall.
25. Shin, D., Kim, J., & Lee, S. (2001). Low-energy intra-task voltage scheduling using static timing analysis. In *Proceedings of the 38th conference on design automation* (pp. 438–443). New York: ACM.
26. Shin, Y., & Choi, K. (1999). Power conscious fixed priority scheduling for hard real-time systems. In *Proceedings of the 36th ACM/IEEE conference on design automation conference* (pp. 134–139). New York: ACM.
27. Shin, Y., & Choi, K. (1999). Power conscious fixed priority scheduling for hard real-time systems. In *DAC* (pp. 134–139).
28. Shin, Y., Choi, K., & Sakurai, T. (2000). Power optimization of real-time embedded systems on variable speed processors. In *Proceedings of the 2000 IEEE/ACM international conference on computer-aided design* (pp. 365–368). Piscataway: IEEE.
29. Weiser, M., Welch, B., Demers, A., & Shenker, S. (1994). Scheduling for reduced cpu energy. In *Symposium on operating systems design and implementation* (pp. 13–23).
30. Yang, C.-Y., Chen, J.-J., & Kuo, T.-W. (2007). 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)*.
31. Yao, F., Demers, A., & Shenker, S. (1995). A scheduling model for reduced CPU energy. In *Proceedings of the 36th annual symposium on foundations of computer science* (pp. 374–382). Piscataway: IEEE.
32. Yun, H.-S., & Kim, J. (2003). On energy-optimal voltage scheduling for fixed-priority hard real-time systems. *ACM Transactions on Embedded Computing Systems, 2*(3), 393–430, Aug.

Yung-Hen (Henry) Lee received his B.S. degree from the Department of Computer Science Engineering in National Chung Cheng University, Taiwan, in 1996. After his compulsory service, in 1998, he joined MITAC as a software engineer till 2000. Then, he had been in charge of silicon porting and chipset development in Phoenix Technology (PTEC) till 2005. He received his M.S. degree from the Department of Computer Science and Information Engineering National Taiwan University, Taiwan, in 2006. In the mean while, he worked for processor development and technical supports in Advance Micro Device(AMD) till 2008. Now, he is in DELL for server development and BIOS enablement.

Dr. Jian-Jia Chen received his Ph.D. degree from Department of Computer Science and Information Engineering at National Taiwan University in 2006. He received his B.S. degree from the Department of Chemistry at National Taiwan University 2001. Since 2008, after completing his compulsory military service, he has been a postdoc researcher at Computer Engineering and Networks Laboratory (TIK) Swiss Federal Institute of Technology (ETH) Zurich, Switzerland. His research interests include real-time systems, energy-efficient scheduling, power-aware designs, embedded systems, temperature-aware scheduling, distributed computing, and algorithmic analysis. He won the Best Paper Award on IEEE RTCSA 2005 and Institute of Information and Computing Machinery (IICM) Doctoral Dissertation Award in 2006.

Dr. Chi-Sheng Shih has been an assistant professor at the Graduate Institute of Networking and Multimedia and Department of Computer Science and Information Engineering at National Taiwan University since February 2004. He received the B.S. in Engineering Science and M.S. in Computer Science from National Cheng Kung University in 1993 and 1995, respectively. In 2003, he received his Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign. His main research interests are embedded systems, hardware/software codesign, real-time systems, and database systems. Specifically, his main research interests focus on real-time operating systems, real-time scheduling theory, embedded software, and software/hardware co-design for system-on-a-chip. He won the Best Paper Award on IEEE RTCSA 2005 and Best Student Paper Award on IEEE RTSS 2004.