Technical Report LA-UR 05-3104, February 2005.

When Discreteness Meets Continuity: Energy-Optimal DVS Scheduling Revisited *

Chung-hsing Hsu and Wu-chun Feng
Los Alamos National Laboratory
Los Alamos, NM 87545
{chunghsu,feng}@lanl.gov

Abstract

The energy-optimal DVS scheduling problem seeks to create a frequency-voltage schedule for the CPU that can achieve energy minimization with tolerable performance loss. Prior solutions to the problem assume that the CPU can run across a continuous range of frequencies and voltages, but today's DVS-enabled microprocessors can only support a discrete set. As a result, the energy-optimal results in the continuous case may no longer be valid in the discrete case. This paper bridges the two cases by showing that the optimality can be retained through emulation. The result is also applicable to systems that consider leakage and/or system energy usage.

1. Introduction

For battery-powered embedded and mobile systems, DVS (Dynamic Voltage Scaling) has been recognized as an effective mechanism to prolong battery life. The mechanism allows the supply voltage (and operating frequency) of the system's microprocessor(s) to be scaled down dynamically. Since a processor's energy consumption exhibits a super-linear relationship with its supply voltage, the scaling-down effect necessarily results in a considerable amount of energy savings. However, the scaling-down effect also introduces potential performance degradation because of a lower maximum operating frequency that can be driven by a scaled-down supply voltage. Thus, knowing how to apply DVS to a set of tasks — so that the CPU energy consumption is minimized while none of the tasks experience intolerable performance loss — becomes a key research issue, which we refer to as the *DVS scheduling problem* in this paper.

Research on the DVS scheduling problem can be traced back to as early as 1994-1995 [12, 14], a couple of years before the production of real DVS-enabled microprocessors. In these early works, a DVS-enabled microprocessor was assumed to be able to run at a *continuous* range of voltages and frequencies with a cubic power-frequency relationship. Unfortunately, today's DVS-enabled microprocessors can only run at a *discrete* set of voltages and frequencies. As a result, although these early works are still applicable, e.g., rounding up the calculated speed to the next available clock speed, their optimality with respect to CPU energy consumption no longer holds.

What is more detrimental is that these early works may not even have sub-optimal results due to the *ideal* assumption of the cubic power-speed relationship at low clock speeds. Today's DVS-enabled microprocessors exhibit a linear power-frequency relationship at lower clock speeds. Thus, the ideal assumption over-states the energy-reduction potential at low clock speeds, and any DVS scheduling algorithm based on such an assumption will execute tasks at low clock speeds whenever possible. However, executing a task at a low speed will increase the task's execution time, thereby consuming more standby energy [4]. If the real energy savings by running at low CPU speeds cannot exceed the energy increases due to a longer execution time, a task may end up using more energy in DVS executions than in non-DVS executions. Therefore, it is desirable to derive an energy-optimal DVS scheduling solution based on the assumptions that hold for today's DVS-enabled microprocessors.

A straightforward approach in deriving an energy-optimal DVS scheduling solution for today's DVS-enabled microprocessors is to cast the problem as an integer programming problem and solve it via existing tools that are generally based on exhaustive search.

* This work was supported by DOE Laboratory-Directed Research & Development through LANL contract W-7405-ENG-36. Available as LANL report: LA-UR 05-3104.

While the optimal solution can be easily (but computationally expensively) computed this way, we argue that the structure of the optimal schedule is more or less destroyed. We argue that retaining the structure of the optimal solution is beneficial in that the structure may provide useful hints to refine heuristics that are designed to solve the DVS scheduling problem efficiently.

For example, Kwon and Kim recently showed that the optimal solution derived by Yao et al.'s polynomial-time algorithm for a set of non-periodic, preemptable real-time tasks in the continuous setting [14] can be easily and quickly transformed into an optimal solution in the discrete setting [6]. In the original optimal schedule, each task is assigned a single clock frequency. Kwon and Kim proved that, if every such frequency is emulated with two bounding supported frequencies, then the resulting schedule is optimal. In other words, heuristics that are shown to be effective for the problem addressed by Yao et al. will probably remain effective for the discrete version of the problem.

Kwon and Kim's result [6] provided a big step towards solving the scheduling problem for today's DVS-enabled microprocessors, but unfortunately, it relies on an assumption that does not yet hold for today's DVS technology. Specifically, Kwon and Kim's work is based on the highly-cited theoretical result from Ishihara and Yasuura [3], a result that was derived under an ideal assumption of the power-frequency relationship. Hence, the optimality result that Kwon and Kim derived may not hold for today's DVS-enabled microprocessors. But we will show that Kwon and Kim's optimality result *can* hold for today's DVS-enabled microprocessors. In fact, we will prove that a less strict power-frequency relationship is needed for Kwon and Kim's optimality result to remain true, and this power-frequency relationship can be easily satisfied through a careful selection of operating points in DVS-enabled microprocessors.

In short, we extend the highly-cited DVS scheduling theory from Ishihara and Yasuura so that it can be applied to today's DVS-enabled microprocessors. The extended result does not make any assumption about the power-frequency relationship. Hence, the extended result can be used for leakage-aware CPU energy minimization as well as system-wide energy minimization, given appropriate power consumption values. Moreover, based on our extension of the above scheduling theory and subsequent analysis, a unified framework that subsumes many important existing works in DVS scheduling theory for the discrete case can be derived.

The remainder of the paper is organized as follows. We first give a brief review of the established results in energy-optimal DVS scheduling. Then in Section 3, we present our main theorem. The proof of how this theorem extends previous work is shown in Section 4. After that, we demonstrate in Section 5 a use of the theorem by taking into account the leakage power. In Section 6 concluding remarks and future work are presented.

2. Background and Related Work

In the continuous version of the energy-optimal DVS scheduling problem, there is a fundamental theorem [14] that has been used extensively and is stated as follows.

> If the CPU power $P(s)$ is a convex[1] function of its speed s, then the schedule that executes the entire job at a constant speed and completes the task right at its deadline D is energy-optimal.

Mathematically, the theorem gives a description about the structure of the optimal solution $s^{opt}(t)$ for the following problem:

$$\min \left\{ \int_0^D P(s(t))dt : \int_0^D s(t)dt = W, s(t) \geq 0 \right\} \quad (1)$$

where $s(t)$ is the processor speed at time t and W is the required CPU clock cycles to complete the task; the theorem says, if $P(s)$ is convex, then $s^{opt}(t) = \frac{W}{D}$.

As mentioned in Section 1, early work on DVS scheduling assumes a cubic power-frequency relationship, which is based on an approximation of the alpha-power law [11] with $v_t = 0$ and $\alpha = 2$.

$$s \propto \frac{(v - v_t)^\alpha}{v} \qquad \text{and} \qquad P(s) \propto s \cdot v^2 \quad (2)$$

where v and s are the supply voltage and operating frequency of a DVS-enabled microprocessor, respectively. For today's technology, the threshold voltage v_t is never zero and the technology parameter α is around 1.3 [15]. Hence, the cubic power-frequency relationship $P(s) \propto s^3$ over-estimates the power-reduction potential at lower CPU speeds, even though the relationship makes $P(s)$ a convex function so that the above theorem can be applied.

Ishihara and Yasuura's work in [3] is one of the first to tackle the issue of discrete operating points in DVS-enabled microprocessors. Assuming that only a limited

1 A function $P(s) : R \rightarrow R$ is *convex* on an interval $I = [a, b]$ if for all points s_1 and s_2 in I and $0 \leq \lambda \leq 1$,

$$P(\lambda s_1 + (1 - \lambda)s_2) \leq \lambda P(s_1) + (1 - \lambda)P(s_2).$$

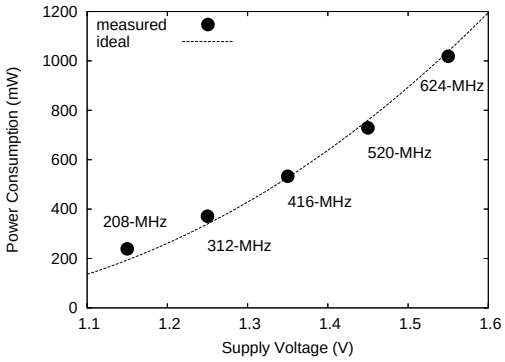


Figure 1. A DVS-enabled Intel PXA270.

set of supply voltages is supported, they found that not only can an unsupported CPU frequency be easily *emulated* through a combination of supported frequencies,[2] but also using only the two immediately-neighboring frequencies to emulate the unsupported frequency will result in the lowest power consumption among all possible combinations. In other words, the above theorem remains valid for the discrete case in the following way:

> $[\cdots]$ If the desired CPU frequency is not directly supported, the two immediately-neighboring CPU frequencies can be used to emulate the desired CPU frequency and result in an energy-optimal DVS schedule.

Ishihara and Yasuura's optimality result is based on an assumption that Equation (2) holds for $v_t \geq 0$ and $1 \leq \alpha \leq 2$. However, this equation is not always valid for today's DVS-enabled microprocessors. First, these processors only provide a discrete set of CPU frequencies, whereas the equation requires a continuous range to be directly supported. Second, the equation only models a processor's dynamic power consumption and ignores other power consumption sources. As a result, Equation (2) over-states the energy-reduction potential at low clock speeds, as illustrated in Figure 1. The data is taken from Intel's application note [1].

In the following, we will show that Ishihara and Yasuura's optimality result can still hold for today's DVS-enabled microprocessors. The extension comes from a key theorem that assumes a discrete set of operating points in a DVS-enabled microprocessor but does not make any assumption about the power-frequency relationship.

2 i.e., the 1.5-GHz clock speed can be emulated by running half of the time at the 1.0-GHz clock speed and half the time at the 2.0-GHz clock speed.

3. A Fundamental Theorem

The discrete version of the energy-optimal DVS scheduling problem is formulated as follows: Given a microprocessor that only exports n operating points $\{(s_i, P_i) : 1 \leq i \leq n\}$ where P_i is the power consumption of the CPU speed s_i and $0 \geq s_1 < s_2 < \cdots < s_n$, we seek to find a solution $(t_1^{opt}, \cdots, t_n^{opt})$ for the following problem.

$$\min \left\{ \sum_i P_i t_i : \sum_i s_i t_i = W, \sum_i t_i = D, \ t_i \geq 0 \right\} \tag{3}$$

If we re-write (3) by replacing t_i with $D \cdot r_i$, we can derive an equivalent problem formulation as follows.

$$D \cdot P_{min}(\frac{W}{D}) \tag{4}$$

where $P_{min}(s) \stackrel{def}{=}$

$$\min \left\{ \sum_i P_i r_i : \sum_i s_i r_i = s, \sum_i r_i = 1, \ r_i \geq 0 \right\} \tag{5}$$

The function $P_{min}(s)$ represents the lowest power consumption with the best possible combination of the n operating points that *emulates* the processor speed s. Equation (4) reveals important information about the energy-optimal DVS schedule for the discrete case; that is, the optimal schedule will run at (possibly emulated) speed W/D throughout the entire job execution. In other words, the structure of an energy-optimal DVS schedule for the continuous case and for the discrete case are very similar.

In fact, the similarity result is not surprising since we can prove that $P_{min}(s)$ is a convex function on $[s_1, s_n]$. Because of this, we can still apply the main theorem for the continuous case to the discrete case and, hence, get an energy-optimal DVS schedule whose structure is very similar to the structure of optimal DVS schedule in the continuous case. Note that the convexity of $P_{min}(s)$ does *not* require any particular power-frequency relationship. That is, the common assumption that $P_i \equiv \mathcal{P}(s_i)$ for some convex function $\mathcal{P}(s)$ is not needed. As we will discuss later, the generality of Theorem 1 provides tremendous help in the construction of an energy-optimal DVS schedule since the assumption $P_i \equiv \mathcal{P}(s_i)$ does not always hold in practice.

Theorem 1 *$P_{min}(s)$ is convex on $[s_1, s_n]$.*

Proof Given two frequencies f and g, assume two sets $\{u_i\}$ and $\{v_i\}$ such that

$$P_{min}(f) = \sum_i P_i u_i, \ \sum_i s_i u_i = f, \ \sum_i u_i = 1, \ u_i \geq 0$$

and

$$P_{min}(g) = \sum_i P_i v_i,\ \sum_i s_i v_i = g,\ \sum_i v_i = 1,\ v_i \geq 0$$

For any $\lambda \in [0,1]$, let $r_i \stackrel{def}{=} \lambda u_i + (1-\lambda)v_i$. It should be easy to verify that

- $r_i \geq 0$ for all i,
- $\sum_i r_i = 1$, and
- $\sum s_i r_i = \lambda f + (1-\lambda)g$.

Also, we can derive that

$$P_{min}(\lambda f + (1-\lambda)g) \leq \sum_i P_i r_i = \lambda P_{min}(f) + (1-\lambda)P_{min}(g)$$

Thus $P_{min}(s)$ is convex. ∎

To demonstrate the generality of Theorem 1, let us consider the energy-optimal DVS scheduling problem for a set of preemptable real-time jobs. The continuous version of the problem has been well studied, and Yao et al. [14] already provides one of the first polynomial-time algorithms to derive an optimal DVS schedule, provided $P(s)$ is convex. The derived schedule will execute a job at a single CPU frequency. Because of Theorem 1, we can replace every unsupported frequency s with the optimal DVS schedule for $P_{min}(s)$ and create an energy-optimal DVS schedule for the general discrete case. This approach extends the Kwon and Kim's DVS scheduling algorithm [6] whose optimality result is based on Ishihara and Yasuura's optimality result [3] and thus only holds for a restricted discrete case.

It may be argued that the DVS schedule generated through the $P_{min}(s)$-directed emulation will introduce a considerable amount of transition overhead because each ideal frequency may be emulated using several frequencies and the transition between frequencies (and also the corresponding voltages) in today's DVS-enabled microprocessors is very expensive (approximately, 10-100 μs). However, we will show in the next section that *at most* two frequencies are needed to emulate the desired frequency, and we will give a necessary and sufficient condition for when the two frequencies will be the immediate neighbors of the desired frequency.

In summary, because of Theorem 1, the energy-optimal DVS algorithm for the discrete case is potentially no more complicated than the optimal DVS algorithm for the continuous case.

4. An In-Depth Analysis on $P_{min}(s)$

In the previous section, we presented a key theorem that bridges energy-optimal DVS scheduling in the discrete case with DVS scheduling in the continuous case through a convex function $P_{min}(s)$. As part of the result, we argued that the optimal DVS algorithms for the discrete case and for the continuous case potentially have the same time complexity if $P_{min}(s)$ is not complicated to compute. In this section, we will give an in-depth analysis on $P_{min}(s)$ and we will argue that the function is fairly simple to compute. Moreover, the $P_{min}(s)$ function can derive a unified framework for several important existing work in DVS scheduling theory.

First, $P_{min}(s)$ models an easy linear programming (LP) problem. According to LP research results, *at most two* DVS operating points will be required to construct an energy-optimal DVS schedule. Again, this optimal solution structure does *not* rely on any special setup for DVS settings such as $P_i \equiv \mathcal{P}(s_i)$ for some convex function $\mathcal{P}(s)$. Since at most two operating points will be used, a straightforward exhaustive enumeration of all possible pairs results in an $O(n^2)$-time computation of $P_{min}(s)$ for a single s.

The time complexity of $P_{min}(s)$ computation can be further reduced if additional assumptions are imposed which will narrow down the possible locations of the two operating points in constructing $P_{min}(s)$. Proposition 1 presents one such approach, and Ishihara and Yasuura's optimality result can be considered as an application of Proposition 1.

Proposition 1 *If*

$$\forall i,\ P_{min}(s_i) = P_i \tag{6}$$

then

$$P_{min}(s) \equiv P_j \cdot r_j + P_{j+1} \cdot (1 - r_j)$$

where

$$r_j = \frac{s_{j+1} - s}{s_{j+1} - s_j} \quad \textit{and} \quad s_j \leq s \leq s_{j+1}$$

Proof Due to convexity of $P_{min}(s)$. ∎

Proposition 2 *If $P_i = \mathcal{P}(s_i)$ for some convex function $\mathcal{P}(s)$, then Condition (6) is true.*

Proof It is obvious that $P_{min}(s_i) \leq P_i$. To prove that $P_{min}(s_i) \geq P_i$, we use the classical Jensen's discrete inequality as follows.

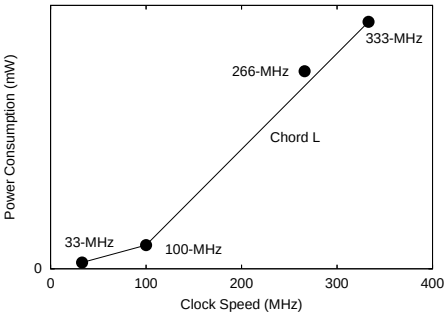
$$\sum_j r_j P_j = \sum_j r_j \mathcal{P}(s_j) \geq \mathcal{P}(\sum_j r_j s_j) = \mathcal{P}(s_i) = P_i.$$

The proof is completed. ∎

Condition (6) is a desirable property to have in a DVS-enabled microprocessor. As pointed out by Lorch and Smith [7], the use of an operating point i that satisfies the inequality $P_{min}(s_i) < P_i$ should be avoided

i	1	2	3	4
s_i (MHz)	33	100	266	333
P_i (mW)	19	72	600	750

(a)



(b)

Figure 2. A DVS-enabled IBM PowerPC 405LP.

since one can get lower power consumption at the same clock speed using other operating points; otherwise, a DVS scheduling algorithm may result in sub-optimal energy usage. we call these unwelcome operating points *power-inefficient* operating points.

Definition 1 *An operating point* (s_i, P_i) *is called* power-inefficient *if* $P_{min}(s_i) < P_i$.

For example, the 266-MHz operating point in a DVS-enabled PowerPC 405LP microprocessor [3], shown in Figure 2(a), is power-inefficient. The 266-MHz clock speed can be emulated by running at 100-MHhz for 2/7 of time and running at 266-MHz for 5/7 of time, and this emulation results in a power consumption of 72 mW $\cdot$ (2/7) + 750 mW $\cdot$ (5/7) = 557 mW. Since the power consumption of the 266-MHz operating point is 600 mW, the 266-MHz operating point is power-inefficient.

To check whether or not Condition (6) is satisfied in a DVS-enabled microprocessor, we derive an equivalent form for Condition (6), stated formally as follows.

Proposition 3 *Condition (6) is true if and only if*

$$\frac{P_2 - P_1}{s_2 - s_1} \leq \frac{P_3 - P_2}{s_3 - s_2} \leq \cdots \leq \frac{P_n - P_{n-1}}{s_n - s_{n-1}} \qquad (7)$$

Proof (The if part) By contradiction. (The only-if part) Due to the convexity of $P_{min}(s)$.

3 The data is taken from [13].

Now consider the PowerPC 405LP example again. We can easily derive that

$$\frac{600 - 72}{266 - 100} = 3.18 > \frac{750 - 600}{333 - 266} = 2.24.$$

If we plot the speed-power curve of this microprocessor as in Figure 2(b), we will find out that the 266-MHz operating point lies above the chord that connects the 100-MHz and 333-MHz operating points. In fact, The chord L represents the power consumption of all emulated speeds between 100-MHz and 333-MHz through the combination of the 100-MHz and 333-MHz operating points. Hence, to identify all power-inefficient operating points, we can construct an underlying convex contour for the speed-power curve, and any operating point that lies above this convex contour is power-inefficient.

In summary, in this section we have given an in-depth analysis on $P_{min}(s)$. We showed that the highly-cited DVS scheduling theory from Ishihara and Yasuura is an instance of a more general optimality result. We also presented a necessary (and also sufficient) condition of when this more general optimality result will be true. The condition is related to the set of operating points in a DVS-enabled microprocessor. It is desirable to have the condition satisfied; otherwise, a DVS scheduling algorithm will be less effective for CPU energy reduction. We provided a way to guarantee that the condition will be satisfied.

5. The Impact of the Idle Power

In general, if the CPU has no task to execute, it will put itself in an idle state. In the idle state, the CPU either is executing *no-op* instructions at the minimum speed or enters into one of the sleep modes. In any case, the CPU idle power P_{idle} is *never zero*, i.e., $P_{idle} \neq 0$.

In the past, the impact of the idle power is ignored, thereby resulting in a common assumption that energy consumption will be reduced whenever the processor speed is scaled down. However, researchers (e.g., [8, 10, 2]) have observed that this assumption does not always hold for today's DVS-enabled microprocessors such as StrongARM SA-1100 and Transmeta Crusoe processor. The root of the problem is that the idle power has increased to a point that it can no longer be ignored. As a result, the DVS schedule that executes a job at a higher-than-necessary clock speed may consume less energy. In the following, we will use $P_{min}(s)$ and subsequent analysis results to model energy-optimal DVS scheduling in the presence of the nonzero idle power.

First of all, we formulate the energy-optimal DVS scheduling in the presence of the nonzero idle power as follows.

$$\min\{d \cdot P_{min}(\frac{W}{d}) + P_{idle} \cdot (D-d) : \ d \leq D\} \quad (8)$$

Through a few algebraic manipulations, we can derive an equivalent formulation for the problem as follows.

$$\min\{\frac{P_{min}(s) - P_{idle}}{s} \cdot W : \ s \geq \frac{W}{D}\} + P_{idle} \cdot D \quad (9)$$

This new problem formulation reveals two pieces of interesting information. One information is that the idle power P_{idle} can be considered as existence throughout the entire time-interval $[0, D)$, This interpretation assists in solving the optimal DVS scheduling problem with respect to system-wide energy minimization. Problem (9) indicates that the standby power consumed by non-CPU devices in the target system does not matter as long as the amount of the standby power is *fixed*. We can focus on generating an energy-optimal DVS schedule that minimizes CPU energy consumption. Another information is that the optimal solution will minimize the CPU energy consumption per cycle, i.e., $\frac{P_{min}(s)-P_{idle}}{s}$. The (possibly emulated) processor speed used in the optimal solution is called the *critical speed* [2, 5].

5.1. The Critical Speed

The term "critical speed" is first introduced by Irani et al. in [2] when they proposed an energy-optimal DVS scheduling algorithm in the case of the nonzero idle power. In their problem formulation, $P_{min}(s)/s$ is assumed to be a monotonic (and also convex) function. On the other hand, Jejurikar et al. [5] found out that, when the leakage power is also considered, the curve $\frac{P_{min}(s)-P_{idle}}{s}$ with respect to the processor speed s is a U-shaped curve. They use the critical speed, which is the lowest point in the curve, to *explicitly* rule out the use of some operating points in the construction of an energy-optimal DVS schedule. Intuitively, executing below the critical speed consumes more time and energy and should be avoided in creating an energy-optimal DVS schedule. Mathematically speaking, they factor P_{idle} into $P_{min}(s)$ (therefore, P_{idle} does not need to be a constant), and they modify $P_{min}(s)/s$ to be a monotonic function.

Our $P_{min}(s)$-derived framework can easily model both problem formulations. We can model Irani et al.'s problem by adding a new operating point $(0, P_{idle})$ into the set of operating points. For Jejurikar et al.'s problem, we do not need to explicitly restrict the set of operating points because the operating points below the critical speed will never be used in the construction of an energy-optimal DVS schedule due to their high $\frac{P_{min}(s)-P_{idle}}{s}$ ratio. Finally, both problem formulations only consider the continuous version of the problem, whereas Problem (9) models the discrete case as well.

Finally, we would like to point out that the partition of the n operating points by a threshold processor speed has been used in energy-optimal DVS scheduling, and the "critical power slope" technique [8] is one of them. However, the power-efficient operating points described in the previous section is *not* one of them.

5.2. Critical Power Slope

One particular technique, called *critical power slope* [8], identifies operating points that should be avoided in energy-optimal DVS scheduling in the case of the nonzero idle power. These operating points are called *energy-inefficient* operating points. Do not get confused with power-inefficient operating points defined in the previous section. The two definitions are *not* equivalent. As we will explain in details later, an energy-efficient operating point can be power-inefficient. In other words, though all operating points of a DVS-enabled microprocessor are energy-efficient, the Ishihara and Yasuura's optimality result cannot be guaranteed.

Formally speaking, an operating point (s_i, P_i) is called *energy-efficient* if the energy consumed by executing W CPU cycles at this operating point is no greater than the energy consumed by any operating point with a higher processor speed, i.e., $\forall \ s_j > s_i$, such that

$$P_i \cdot \frac{W}{s_i} + P_{idle} \cdot (D - \frac{W}{s_i}) \leq P_j \cdot \frac{W}{s_j} + P_{idle} \cdot (D - \frac{W}{s_j}).$$

Otherwise, the operating point is called energy-inefficient. The above inequality can be simplified, resulting in the following equivalent definition for an energy-efficient operating point. The ratio $\frac{P_i - P_{idle}}{s_i}$ is the so-called *critical power slope* for setting (s_i, P_i).

Definition 2 *An operating point* (s_i, P_i) *is called* energy-efficient *if*

$$\forall \ j > i, \ \frac{P_i - P_{idle}}{s_i} \leq \frac{P_j - P_i}{s_j - s_i} \quad (10)$$

Consider the PowerPC 405LP example again. In the previous section, we illustrated that the 266-MHz operating point is power-inefficient because a lower power consumption can be achieved using other operating

points that emulate the 266-MHz clock speed. However, the 266-MHz is energy-efficient when $P_{idle} = 12$ mW, i.e.,

$$\frac{600-12}{266} = 2.21 \leq \frac{750-600}{333-266} = 2.24$$

Note that the 266-MHz operating point is considered energy-inefficient if $P_{idle} = 0$.

From this example, we know that an energy-efficient operating point can be power-inefficient. Why? It turns out that Condition (6) can derive Condition (10) with $s_1 \equiv 0$ and $P_1 \equiv P_{idle}$, In other words, if all the operating points are power-efficient, they will be energy-efficient, and an energy-efficient operating point does not necessarily mean that it will be power-efficient. Specifically, Condition (6) can derive two other conditions according to Proposition 4. These two derived conditions plus Condition (6) can derive Condition (10), as illustrated in Figure 3.

Proposition 4 *If Condition (7) is true, then the following two conditions are true.*

$$\forall 1 \leq i < n,\ \frac{P_{i+1}-P_1}{s_{i+1}-s_1} \leq \frac{P_{i+1}-P_i}{s_{i+1}-s_i} \qquad (11)$$

$$\forall 1 < i < n,\ \frac{P_i-P_1}{s_i-s_1} \leq \frac{P_{i+1}-P_1}{s_{i+1}-s_1} \qquad (12)$$

Proof Omitted.

In sum, to guarantee that the Ishihara and Yasuura's optimality result still holds for today's DVS-enabled microprocessors, all the operating points need to be power-efficient, which is a more strict requirement than for all operating points being energy-efficient, a concept derived from the "critical power slope" technique.

6. Conclusions and Future Work

The energy-optimal DVS scheduling problem seeks to create a DVS schedule for the CPU that can achieve energy minimization with tolerable performance loss. Prior solutions to the problem assume that the CPU can run across a continuous range of frequencies and voltages with a cubic power-frequency relationship. In this paper we presented a key theorem that is not based on these two unrealistic assumptions, yet it still retains the energy-optimal results in these previous works for today's DVS-enabled microprocessors. Moreover, based on our extension of the DVS scheduling theory and subsequent analysis, a unified framework that subsumes many important existing works in DVS scheduling theory for the discrete case can be derived.

The key theorem and subsequent analysis results we presented in this paper assume a very basic setting and, hence, can be extended in various ways. For example, the transition overhead can be taken into account, the deterministic execution time can be replaced with stochastic execution time, and a set of periodic real-time tasks can be used instead of a single job. We also wish to perform simulation studies to evaluate the degree of the benefit in energy-optimal DVS scheduling versus heuristic techniques such as pseudo operating points [9]. The "pseudo operating points" technique pre-defines additional operating points through emulation and rounds up the calculated clock speed to the next (possibly pseudo) operating point. This technique is closely related to the operating-point setup problem [10] in DVS scheduling theory.

References

[1] Intel Corporation. Intel PXA27x processor family power requirements. Application Note 280005-002, 2004.

[2] S. Irani, S. Shukla, and R. Gupta. Algorithms for power savings. In *Proceedings of the 14th Symposium on Discrete Algorithms*, January 2003.

[3] T. Ishihara and H. Yasuura. Voltage scheduling problem for dynamically variable voltage processors. In *International Symposium on Low Power Electronics and Design (ISLPED)*, August 1998.

[4] R. Jejurikar and R. Gupta. Dynamic voltage scaling for system-wide energy minimization in real-time embedded systems. In *International Symposium on Low Power Electronics and Design (ISLPED)*, August 2004.

[5] R. Jejurikar, C. Pereira, and R. Gupta. Leakage aware dynamic voltage scaling for real-time embedded systems. In *Proceedings of the 41th Design Automation Conference (DAC)*, June 2004.

[6] W-C Kwon and T. Kim. Optimal voltage allocation techniques for dynamically variable voltage processors. In *Proceedings of the 40th Design Automation Conference (DAC)*, June 2003.

[7] J. Lorch and A. Smith. Operating system modifications for task-based speed and voltage scheduling. In *The First International Conference on Mobile Systems, Applications, and Services (MobiSys)*, May 2003.

[8] A. Miyoshi, C. Lefurgy, E. Hensbergen, and R. Rajkumar. Critical power slope: Understanding the runtime effects of frequency scaling. In *Proceedings of the 16th Annual ACM International Conference on Supercomputing (ICS)*, June 2002.

[9] V. Rao, G. Singhal, and A. Kumar. Real tim,e dynamic voltage scaling for embedded systems. In *International Conference on VLSI Design (VLSID)*, January 2004.

[10] S. Saewong and R. Rajkumar. Practical voltage-scaling for fixed-priority RT-systems. In *Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposiuam (RTAS)*, May 2003.

[11] T. Sakurai and A. Newton. Alpha-power law MOSFET model and its applications to CMOS inverter delay and

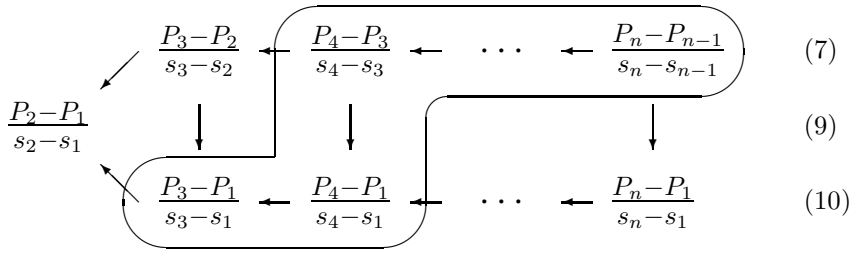


Figure 3. The enclosed area represents the energy-efficiency of the operating point (s_3, P_3).

other formulas. *IEEE Journal of Solid-State Circuits*, 25(4):584–594, April 1990.

[12] M. Weiser, B. Welch, A. Demers, and S. Shenker. Scheduling for reduced CPU energy. In *the 1st Symposium on Operating Systems Design and Implementation (OSDI)*, November 1994.

[13] R. Xu, C. Xi, R. Melhem, and D. Mossé. Pratical PACE for embedded systems. In *the ACM International Conference on Embedded Software (EMSOFT)*, September 2004.

[14] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *IEEE Annual Symposium on Foundations of Computer Science*, October 1995.

[15] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner. Theoretical and practical limits of dynamic voltage scaling. In *Proceedings of the 41th Design Automation Conference (DAC)*, June 2004.