Application-Driven Low-Power Techniques Using Dynamic Voltage Scaling

Taewhan Kim
School of Electrical Engineering and Computer Science
Seoul National University, Seoul, KOREA
tkim AT snu.ac.kr

Abstract

It is generally accepted that dynamic voltage scaling (DVS) is one of the most effective techniques of energy minimization for real-time applications. The effectiveness comes from the fact that the amount of energy consumption is quadractically proportional to the voltage applied to the processor. The penalty is the execution delay, which is linearly and inversely proportional to the voltage. According to the granularity of units to which voltage scaling is applied, the DVS problem is divided into two subproblems: inter-task DVS problem, in which the determination of the voltage is carried out on a task-by-task basis and the voltage assigned to the task is unchanged during the whole execution of the task, and intra-task DVS problem, in which the operating voltage of a task is dynamically adjusted according to the execution behavior to reflect the changes of the required number of cycles to finish the task before the deadline. Frequent voltage transitions may cause an adverse effect on energy minimizatuion due to the increase of the overhead of transition time and energy. In this paper, we survey and describe, in a theoretical aspect, state-of-art techniques of dynamic voltage scaling problems, which include: (1) inter-task DVS problem, (2) intra-task DVS problem, (3) integrated inter-task and intra-task DVS problem, and (4) transition-aware DVS problem.

1 Introduction

Over the past decades there have been enormous efforts to minimize the energy consumption of CMOS circuit systems. Dynamic voltage scaling (DVS), involving dynamic adjustments of the supply voltage and the corresponding operating clock frequency, has emerged as one of the most effective energy minimization techniques. A one to one correspondence between the supply voltage and the clock frequency in CMOS circuits imposes an inherent constraint to DVS techniques to ensure that voltage adjustments do not violate the target system's timing constraints.

Many previous works have focused on hard real-time systems with multiple tasks. Their primary concern is to assign a proper operating voltage to each task while satisfying the task's timing constraint. In these techniques, determination of the voltage is carried out on a task-by-task basis and the voltage assigned to the task is unchanged during the whole execution of the task, which is refered to as *inter-task voltage scheduling*. Yao *et al.* [1] proposed an optimal inter-task voltage scheduling algorithm for independent tasks in which a task is characterized by its arrival time, deadline and required CPU cycles. The proposed scheduling technique computes the speed of execution at any given time (and thus automatically determines each task's starting and ending times) so that the total energy consumption is minimized. Although they formulated the problem without the constraint that the task should be assigned to a single operating voltage, by the convexity of the power function each task is given only one 'middle' voltage that is proved to be optimal. This is because of the underlying assumption that the speed of any specific time is constant, which is not true since the required number of processor cycles, on which the calculation of the speed may vary depending on the behavior of the task. That means, the proposed inter-task scheduling technique is optimal only if each execution of the task follows the worst-case execution path. Leaving the same assumption untouched, many works in the literature have tried to formulate new inter-task scheduling problems considering other issues. Some instances of such issues include tasks with dependency relations [2, 3, 4, 5, 6], discretely variable voltage processors [7, 5], multi-processor environments [2, 3, 4, 5, 6], voltage transition overheads [6]. Recently, it has been reported [8] that the *voltage transition overhaeds* are not trivial in terms of energy and delay, and should be taken into account in DVS as well.

On the other hand, a number of studies of other direction (e.g., [9, 10]) have added a new dimension to the voltage scheduling problem, by considering energy saving opportunities within the task boundary. In their approach, the operating voltage of the task is dynamically adjusted

Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)
0-7695-2676-4/06 $20.00 © 2006 IEEE

according to the execution behavior to accurately reflect the changes of the required number of cycles to finish the task before the deadline, which is referred to as *intra-task voltage scheduling*. Shin *et al.* [9] proposed a remaining worst-case path-based algorithm which achieves the best granularity by executing the basic blocks with possibly different operating voltages. To obtain tight operating points that lead to a minimum energy consumption, the algorithm updates the remaining path length as soon as the execution deviates from the previous remaining worst-case path. More recently, a profile-based optimal intra-task voltage scheduling technique was presented in [10]. It shows the best energy savings by incorporating the task's execution profile into the calculation of the operating voltages. This algorithm is proved to be optimal in that when the task is executed repeatedly or periodically, it achieves a minimum average energy consumption. A recent work [11] attempts to solve *the inter-task and intra-task DVS problems simultaneously* is so as not to miss the energy saving opportunity at each granularity of inter-task and intra-task DVS.

In this paper, we survey and describe, in a theoretical aspect, state-of-art techniques of dynamic voltage scaling problems, which include: (1) inter-task DVS problem, (2) intra-task DVS problem, (3) integrated inter-task and intra-task DVS problem, and (4) transition-aware DVS problem.

2 Intra-task DVS techniques

The amount of energy dissipation for the execution of a task is

$$E \propto {V_{DD}}^2 \times N_{tot} \quad (1)$$

where N_{tot} is the total number of instruction cycles executed for a task. Thus, the *intra-task voltage scheduling* problem is to assign a proper voltage to each basic block of the task so that the energy consumption in *Eq.(1)* is minimized.

The relationship between clock frequency and voltage in CMOS circuits is

$$f_{CLK} \propto (V_{DD} - V_T)^{\alpha}/V_{DD}. \quad (2)$$

where V_T is the threshold voltage and α is the velocity saturation index. If the value of V_T is small enough, the expression is reduced to ${f_{CLK}} \propto {V_{DD}}^{\alpha-1}$.

Since the clock frequency determines the voltage, the scheduling problem can be stated as:

(Intra-Task DVS Problem) *The intra-task voltage scheduling problem for a task's CFG is to determine the clock frequency for each node (i.e., basic block) of the CFG so that the total energy by the task is minimized while satisfying the timing constraint of the task.*

The key problem to solve is to determine what clock frequency should be set to the entry point of each basic block so that the overall energy consumption of the task is minimized. Existing intra-task DVS techniques can be classified according to the way of determining the lowest clock frequency to be set at the entry point of each basic block.

1. (*RWCEP based DVS*): This techique [9] uses the lowest clock frequency by which the remaining WCEP (worst case execution path) can be completed within the deadline of the task.
2. (*RACEP based DVS*): This techique [12] uses the lowest clock frequency by which the remaining ACEP (average case execution path) can be completed within the deadline of the task.
3. (*ROCEP based DVS*): This techique [10] uses the lowest clock frequency by which the remaining OCEP (energy-optimal case execution path) can be completed within the deadline of the task.

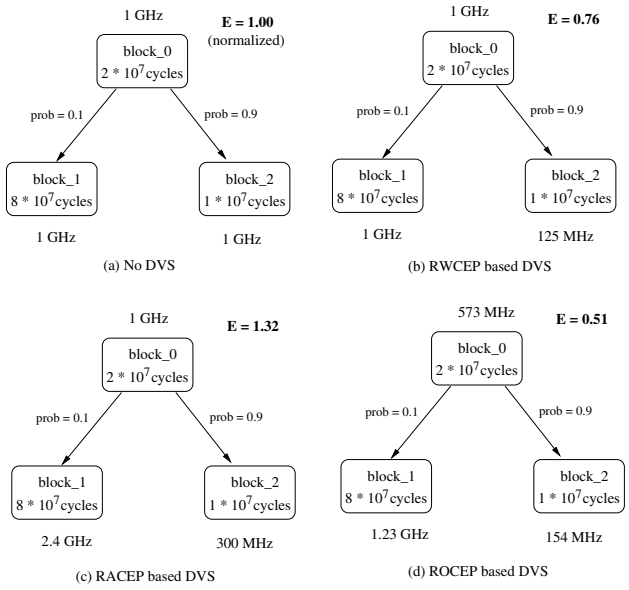


Figure 1: **Example showing the calculations of clock frequency in basic block_0.**

For example, consider a simple hard real-time task with deadline of 100 ms and three basic blocks. Its control flow graph is shown in Figure 1(a) where the number inside each node indicates the number of execution cycles of the block and the number assigned to each arc indicates the probability that the control flow follows the edge. If DVS technique is not used, the speed for the task should be set tightly to (length of the critical path) / (remaining time to deadline)

Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)
0-7695-2676-4/06 $20.00 © 2006 IEEE

= $[(2+8)10^7 cycles]/(100 \cdot 10^{-3}s) = 1GHz$. (See Figure 1(a).) If DVS technique follows the worst case execution path, the speed of block_2 will be set to (length of the (RWCEP from block_2) / (remaining time to deadline) = $[1 \cdot 10^7 cycles]/(80 \cdot 10^{-3}s) = 125MHz$. (See Figure 1(b).) On the other hand, if DVS technique follows the average case execution path, the speed is set to (length of the (RACEP from block_0) / (remaining time to deadline) = $[(2+1) \cdot 10^7 cycles]/(100 \cdot 10^{-3}s) = 300MHz$. (See Figure 1(c).) Finally, if DVS follows energy-optimal execution path, the speed to be set in block_0 is calculated based on the probabilities of its succeeding basic blocks. Here, the speed is 573 MHz. See Figure 1(d).) In summary, we can see that The ROCEP based DVS technique outperforms the others because the clock speed used at each basic block alwyas leads to the total energy consumption which is optimal on the average.

The work in [10] contains the detailed procedure of energy-optimal speed calculation of basic blocks. Here, we give a summary of the speed calculation. A task τ is represented with its CFG $G_\tau = (V, A)$, where V is the set of basic blocks in the task and A is the set of directed edges which impose precedence relations between basic blocks. (For example, see Fig. 2(a).) The set of immediate successor basic blocks of any $b_i \in V$ is denoted by $succ(b_i)$. Each basic block b_i is annotated with its non-zero number of execution cycles n_i and each arc (b_i, b_j) is given a probability p_j that the execution follows the arc.

Given a task's CFG and its execution profile that offers the probabilities, we execute each basic block b_i at a speed of $\delta_i/(remaining\ time)$ and adjust the supply voltage accordingly, where δ_i is defined as:

$$\delta_i = \begin{cases} n_i, & \text{if } succ(b_i) = \emptyset \\ n_i + \sqrt[3]{\sum_{\forall b_j \in succ(b_i)} p_j {\delta_j}^3}, & \text{otherwise} \end{cases} \quad (3)$$

The corresponding average energy consumption is proved to be optimal and expressed as [10]:

$$E_{intra} = C \cdot \Big(\frac{\delta_0}{(relative)\ deadline}\Big)^2 \cdot \delta_0 \quad (4)$$

where C is a system-dependent constant, and δ_0 is the δ value of the top basic block b_0 and called *energy-optimal path length* of the task. One interesting interpretation of *Eq.(4)* is that it can be considered as the energy consumed in the execution of δ_0 cycles at a speed of $\delta_0/deadline$. Note that $\delta_0/deadline$ is the initial speed of the task.

For example, consider a real-time task τ_{simple} shown in Fig. 2(a). Fig. 2(b) shows the procedure of calculating each δ_i value according to *Eq.(3)*. Once we have performed the calculation, the operating speed of basic block b_i is simply obtained by dividing δ_i by the remaining time to deadline.

For example, suppose that $deadline$ = 10 (unit time) and the execution follows the path $(b_0, b_2, b_3, b_5, b_6, b_8)$. Then, the corresponding speed/ending time changes as follows:

$$\begin{aligned} \text{Speed}: &\ 2.93 > 3.24 > 3.14 > 3.14 > 2.26 > 2.26 \\ \text{Ending Time}: &\ 2.05 > 3.29 > 3.92 > 4.24 > 7.78 > 10 \end{aligned}$$

In Figure. 2(b), the thick, dotted, and regular arrows respectively indicate the increase, decrease and no change of the processor speed, and the basic blocks with changed operating speed are marked with gray-color.

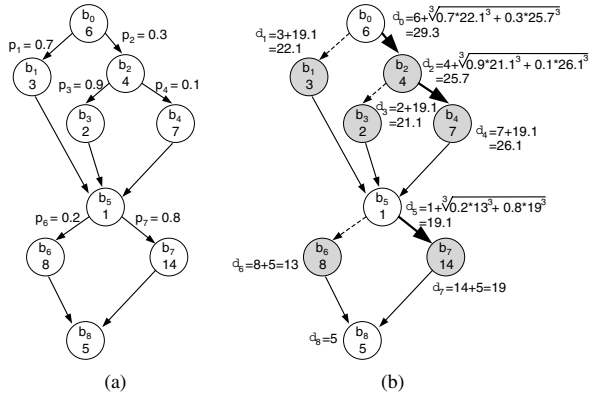


Figure 2: **(a) CFG of a task τ_{simple}; (b) Calculation of δ values.**

3 Inter-task DVS

The most DVS works belong to the inter-task DVS. The inter-task DVS problem can be stated as:

(Inter-Task DVS Problem) *Given an instance of tasks with deadlines and voltages, find a feasible task-level schedule and task-level voltage allocation to tasks that minimizes the total energy consumptin while satisfying the deadline constraints of tasks.*

Table 1 summaizes the current status of the energy-optimal works for the constrained cases of inter-task DVS problems. Note that in addition to the voltage, the amount of energy consumption is affected by the switched capacitance of the task. The value of capacitance is determined according to the execution characteristics of the task: If the task requires hardware components with high switched capacitance, such as multipliers, for execution, the capacitance value will be large, and vice versa. Consequently, to reduce the total energy consumed by tasks, it is desirable to execute the tasks with low switched capacitance using high supply voltages while the tasks with high switched capacitance using low supply voltages. The unsolved case is that of continuous voltages with nonuniform capacitance

of tasks. The work by Kwon *et al.* [7] for uniform capacitance case actually makes use of the optimal-work by Ishihara *et al.* [14] and Yao *et al.* [1]. The work by Kwon *et al.* [7] for nonuniform capacitance case uses a linear programming (LP) formulation. Here, we show the algorithm of Kwon *et al.* [7] for uniform capacitance case.

Table 1: **Summary of intra-task DVS techniques.**

Voltage	Tasks		optimal? / ref.
cont. voltage	single task		Yes / trivial
	multiple tasks	uniform cap.	Yes / Yao *et al.* [1]
		nonuniform cap.	unknown
disc. voltage	single task		Yes / Ishihara *et al.* [14]
	multiple tasks	uniform cap.	Yes / Kwon *et al.* [7]
		nonuniform cap.	Yes / Kwon *et al.* [7]

The procedure starts from the results of the possibly invalid voltage allocation with the feasible task schedule obtained from Yao *et al.*'s algorithm [1], and transforms it into that of valid voltage allocation with a feasible schedule. More precisely, the schedule of tasks during transformation, but change the voltages so that they are all valid. Then, the question is what and how the valid supply voltages are selected and used. A valid voltage for each (scheduled) task is determined by performing the following three steps: (Step 1: *Merge time intervals*) All the scheduled time intervals that were allotted to execute the task are merged into one; (Step 2: *Voltage reallocation*) The invalid supply voltage is replaced with a set of valid voltages. (Step 3: *Split time interval*) The merged time interval is then split into the original time intervals.

For example, suppose that we have three voltages 7.0V, 5.0V, and 3.0V available for use and their corresponding clock speeds are 70MHZ, 50MHZ, and 30MHZ, respectively. Then, for each scheduled task with the ideal voltage in Figure 3(a), the three steps are applied. Figure 4 shows the results of three steps for task J_1. Initially, J_1 is scheduled to be executed in two time intervals [0,3] and [8,9] with the voltage being 3.75V, as shown in Figure 4(a). (According to the results in [1] each task always uses the same voltage.) Consequently, in Step 1 the time intervals are merged into [0,4] as shown in Figure 4(b). The supply voltage in Step 2 is then updated. To do this, we make use of Ishihara and Yasuura's results [14]: For a given ideal (optimal) voltage for a task, the valid (optimal) voltage allocation is to use the two immediately neighboring valid voltages to the ideal voltage. (For details on how to find the time point at which the clock speed changes, see [14].) Figure 4(c) shows the result of voltage reallocation where two voltages 3.0V and 5.0V are used because the ideal voltage

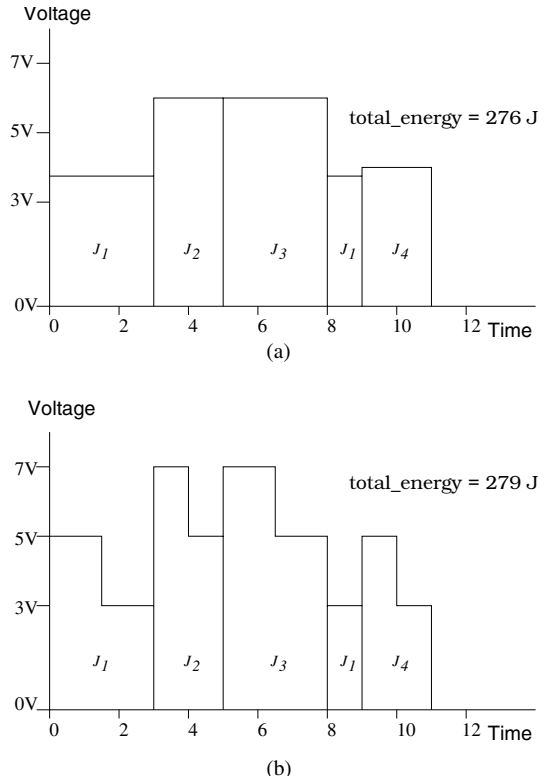


Figure 3: **An example illustrating the transformation of continuously variable voltage allocation into discontinuously variable voltage allocation. (a) A continuously variable voltage allocation for tasks; (b) A discontinuously variable voltage allocation derived from (a).**

(=3.75V) is in between 3.0V and 5.0V, and no other valid voltages are in the interval. Finally, in Step 3 we restore the time intervals while preserving the voltage reallocation obtained in Step 2, as shown in Figure 4(d). By repeating these three steps for J_2, J_3, and J_4 in Figure 3(a), a voltage allocation for all tasks with a feasible schedule is obtained, as shown in Figure 3(b). Note that because we used only a number of discrete voltages, the energy consumption, which is $279J$, increases from that in Figure 3(a), which is $276J$. However, according to [7], the amount of the increase is minimal.

4 Integration of Intra-task DVS

The combined problem of the inter- and intra-task DVS problems can be described as:

(Combined DVS Problem) *Given an instance of tasks with deadlines and voltages, find a feasible task-level*

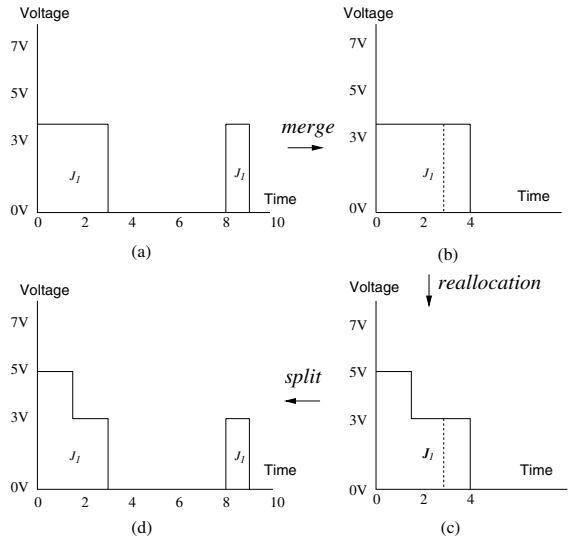


Figure 4: **The three steps of voltage allocation procedure [7] for task J_1 in Figure 3. (a) An initial schedule in Figure 3(a); (b) The result after merging time intervals; (c) The result after voltage reallocation; (d) The result after splitting the time interval.**

schedule and task-level voltage allocation to tasks that minimizes the total energy consumptin while satisfying the deadline constraints of tasks.

There are two optimal results in the literature related to the combied DVS problem: (i) an optimal inter-task DVS scheme [1] that determines the operating voltage of each task assuming the worst-case execution path and (ii) an optimal intra-task DVS scheme [10] that determines the operating voltage of each basic block in a single task. From the analysis of the procedures of (i) and (ii), the work in [11] found that an energy-optimal integration of (i) and (ii) is possible with a slight modification of the procedures. The proposed integrated DVS approach is a two-step method:

1. Statically determine energy-optimal starting and ending times (s_i and e_i) for each task τ_i.

2. Execute τ_i within $[s_i, e_i]$ while varying the processor speed according to the voltage scales obtained by an existing optimal intra-task DVS scheme.

The key concern is to develop a new inter-task scheduling algorithm that finds starting and ending times (s_i and e_i) for each task, which leads to a minimum value of total energy consumption when an optimal intra-task scheme is applied to the tasks. For the further details on the optimal algorithm, it can be referred to the work in [11].

5 Transition-Aware DVS

During the execution of tasks, three types of transition overhead are encountered for each voltage transition : *transition cycle*, *transition interval* and *transition energy*.

A *transition cycle* (denoted as Δn) is the number of instruction cycles of the transition code itself. We assume that the energy for executing the transition instruction varies depending on the voltages in transition.

A *transition interval* (denoted as Δt) is the time taken during the voltage transition, which is not constant. Note that in the current commercial designs, the Phase-Locked Loop that is used to set the clock frequency requires a fixed amount of time to lock on a new frequency. This locking time is known to be independent of the source and target frequencies, and is typically much smaller than the time it takes for the voltage to change [15]. Therefore, it is desirable to assume that the transition interval Δt is not a fixed value. A reasonable assumption for the variable voltage processor is that the transition interval is proportional to the difference between the starting and ending transition voltages [16].

A *transition energy* (denoted as ΔE) is the amount of energy consumed during the transition interval by the systems. The value of ΔE may vary depending on the starting and ending voltage levels. The DVS problem with the additional consideration of transition energy is even more difficult to solve because the generalized model of the transition energy for various systems/processors is hard to obtain and even a simplified version of the problem with the assumption that the transition energy is a constant looks a non-trivial task to solve. Let $\Delta E_{(v_i \rightarrow v_j)}$ denote the energy dissipated by the transition of voltage from v_i to v_j. We assume that the processor has a set $\mathcal{V}$, called *voltage set*, of voltages available to use, i.e., $\mathcal{V} = \{v_1, v_2, \cdots, v_M\}$ ($v_1 < v_2 < \cdots < v_M$). In addition, we assume that the values of $\Delta E_{(v_i \rightarrow v_j)}$, $v_i, v_j \in \mathcal{V}$ have already been given, and the values are stored in a table, called *transition-energy table*. Furthermore, we reasonably assume that $\Delta E_{(v_i \rightarrow v_j)} < \Delta E_{(v_i \rightarrow v_{j'})}$ if $|v_i - v_j| < |v_i - v_{j'}|$.

Transition energy aware DVS problem: *Given a fixed schedule of tasks, and voltage set $\mathcal{V}$ and the associated transition-energy table, assign the voltages to the tasks so that the total energy consumption for the execution of tasks, together with the energy consumed by the voltage transitions, is minimized.*

Unfortunately, most of the DVS methods never take into account the minimization of the amount of energy consumed during the voltage transition. To our knowledge, Mochocki, Hu, and Quan [13] and Shin and Kim [8] addressed the DVS problem with the consideration of transition overheads and discrete voltages. The work in [13] put the primary emphasis on the consideration of transition in-

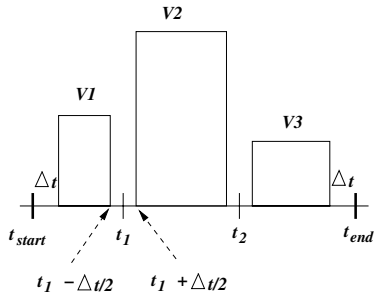
tervals by modifying the optimal scheduling algorithm in [1] together with a simple treatment on both the transition energy and discrete voltages. On the other hand, the work in [8] attempted the limitation of the work in [13] in that it tries to solve the probelm optimaly for a constrained case. We review the work by [8] here. The key contribution of the work is the network formulation of the problem. Here, we only show the procedure of single task case. It was natually exyend to the caes of multiple tasks.

Suppose an instance of TE-VA has a task τ_1 with the schedule interval $[t_{start}, t_{end}]$ and R cycles to be executed, If the overheads of voltage transition are not taken into account, an optimal result can be obtained by using the voltage assignment technique in [14], i.e., the optimal voltage assignment is to use the two voltages in $\mathcal{V}$ that are the immediate neighbors to the (ideal) voltage corresponding to the lowest possible clock speed which results in an execution of task τ_1, exactly starting at time t_{start} and ending at time t_{end}; (The ideal voltage can be obtained according to the voltage and delay (clock speed) relation [14].

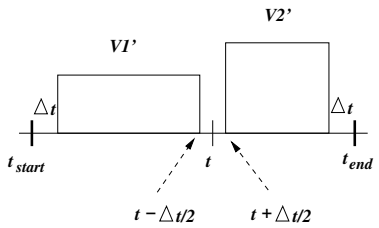
Lemma I *For an instance of the transition-aware DVS problem with a single task, an energy-optimal voltage assignment uses at most two voltages for the execution of task.* (The proof described below becomes the foundation of the proposed network formulation.)

proof. Suppose there is an optimal voltage assignment $\mathcal{VA}$ which uses more than two voltages for the execution of the task during the scheduled interval of $[t_{start}, t_{end}]$. Let $v_1, \cdots, v_K$ ($K > 2$) be the sequence of voltages applied to the task, starting from t_{start} to t_{end}, by the optimal voltage assignment, and let $[t_{start} + \Delta t, t_2 - \Delta t/2]$, $[t_2 + \Delta t/2, t_3 - \Delta t/2]$, $\cdots$, $[\Delta t/2 + t_r, t_{end} - \Delta t]$ be the corresponding execution intervals. (See Figure 5(a) for an example.) Note that the length of the actual execution interval, not including the transition interval, is $T_L = t_{end} - t_{start} - (r+1)\Delta t$. Since the voltages before and after the execution of the task are 0V, transition interval Δt is required from 0V to v_1 at the beginning (i.e., $[t_{start} + \Delta t, t_2 - \Delta t/2]$) and from v_r to 0V at the end (i.e., $[\Delta t/2 + t_r, t_{end} - \Delta t]$) Also, a transition interval is required between the two consecutive execution intervals (e.g., Δt around time t_i in $[t_{i-1} + \Delta t/2, t_i - \Delta t/2]$ and $[t_i + \Delta t/2, t_{i+1} - \Delta t/2]$).

Now, consider another voltage assignment $\mathcal{VA}'$ which uses two voltages for the actual execution length of $T'_L = t_{end} - t_{start} - 3\Delta t$. (See Figure 5(b).) Let v'_1 and v'_2 be the two voltages which lead to a minimum total energy consumption except the transition energy. (Note that the two voltages can be obtained by applying the technique in [14] to the task of R cycles with an execution interval of length T_L.) Then, we want to compare the amount of energy consumptions excluding the transition energy used



(a) An example of more than two voltage assignment for a task execution



(b) An example of two voltage assignment for the same task in (a) (v1' and v2' are adjacent voltages)

Figure 5: **Transition intervals and actual execution intervals for an example of voltage assignment using more than two voltages and voltage assignment using two voltages.**

in $\mathcal{VA}$ and $\mathcal{VA}'$, and compare the amounts of transition energy used in $\mathcal{VA}$ and $\mathcal{VA}'$.

Obviously, we can see that $max\{v_1, \cdots, v_r\} \geq max\{v'_1, v'_2\}$ since the execution interval for $\mathcal{VA}$ is shorter than that for $\mathcal{VA}'$ due to more transition intervals in $\mathcal{VA}$ than that of $\mathcal{VA}'$. This means that the transition energy from 0V at the start to eventually $max\{v_1, \cdots, v_r\}$ is greater than that from 0V to $max\{v'_1, v'_2\}$ by the assumption $\Delta E_{(v_i \rightarrow v_j)} < \Delta E_{(v_i \rightarrow v_{j'})}$ if $|v_i - v_j| < |v_i - v_{j'}|$. Thus, the total transition energy for $\mathcal{VA}$ is greater than that for $\mathcal{VA}'$. On the other hand, since from the fact that v'_1 and v'_2 are the voltages of optimal voltage assignment with time length of T'_L, and $T'_L < T_L$, the total energy consumption without transition energy for $\mathcal{VA}'$ is less than that for $\mathcal{VA}$. Thus, the assumption that $\mathcal{VA}$ is optimal is false. $\diamondsuit$

From Lemma I, the solution space for a single task can be confined to the solutions which only use at most two voltages in $\mathcal{V}$. The remaining issue is to give, for an given execution interval for the task, a technique of finding the two (optimal) voltages and their execution intervals. According to [8]: A network $N(V, A)$ is constructed for a task with R instruction cycles, execution interval $[t_{start}, t_{end}]$, voltage set $\mathcal{V}$ and transition energy table where V is the set of nodes and A is the set of arcs between two nodes. Note that Lemma I tells there is always an optimal voltage

assignment which uses two voltages only. (See the upper figure in Figure 6.) If we know the two voltages used, then the two actual execution intervals can be computed accordingly using the speed and voltage relation [14, 7]. For each of the two execution intervals, $|\mathcal{V}|$ nodes are arranged vertically, each node representing a unique voltage in $\mathcal{V}$. Then two additional nodes are included in V. One is *start-node*, placing at the front of $N(V, A)$ and the other is *end-node* at the end. (See the lower figure in Figure 6.) We insert arcs from the nodes in the first column to those in the second column, from the start-node to the nodes in the first column, and from the nodes in the second column to the end-node. (See the lower figure in Figure 6.) We then assign weight to each arc. The weight of an arc from the start-node to a node labelled as v_i in the first column indicates $\Delta E_{(0V \rightarrow v_i)}$, and the weight of an arc from a node labelled as v_i to the end-node indicates $\Delta E_{(v_i \rightarrow 0V)}$. The weight of an arc from a node labelled as v_i to a node labelled as v_j represents the total sum of the (minimal) energy consumed by the execution of the task (i.e., $E1(v_1) + E2(v_2)$ in Figure 6) and the transition energy $\Delta E_{v_i \rightarrow v_j}$ [1]

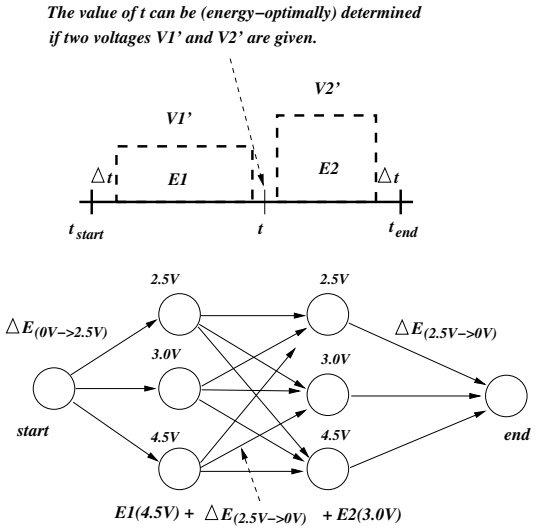


Figure 6: **The construction of network for modeling an instance of the TE-VA problem with a single task.**

Then, from the network $N(V, A)$, a shortest path can be found from the start-node to the end-node. The cost of the shortest path is exactly equivalent to the total amount of energy consumption including the transition energy for the execution of the task. the procedure [8] performs the two steps: (Step 1) Construct network $N(V, A)$ for the task; (Step 2) Find a shortest path (SP) on $N(V, A)$.

[1] In case of $v_i = v_j$, there will be no voltage transition and $\Delta E_{(v_i \rightarrow v_j)} = 0$.

6 Conclusions

In this paper, we described the current status of the research works on dynamic voltage scaling (DVS) in view of the optimiality of energy minimization. The DVS problems we covered in the paper were (1) inter-task DVS problem, (2) intra-task DVS problem, (3) integrated inter-task and intra-task DVS problem, and (4) transition-aware DVS problem. It should be mention that, except the techniques described in the paper, there are many other effective DVS tecnhiques which targeted to the other important variants of the DVS problem, such as periodic tasks, fixed priority scheduling, jitter constraints, leakage-current aware DVS, soft deadline, etc.

7 Acknowledgement

This research work has been supported byNano IP/SoC Promotion Group of Seoul R&BD Program in 2006. This work was also partially supported by the Ministry of Science and Technology (MOST) / Korea Sceince and Engineering Foundation (KOSEF) through the Advanced Information Technology Research Center(AITrc).

References

[1] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced CPU energy," *Proc. of IEEE Symposium on Foundations of Computer Science*, 1995.

[2] M. T. Schmitz, B. M. Al-Hashimi, and P. Eles, "Energy-efficient mapping and scheduling for DVS enabled distributed embedded systems," *Prof. of Design Automation and Test in Europe* , 2002.

[3] Y. Zhang, X. Hu, and D. Z. Chen, "Task scheduling and voltage selection for energy minimization," *Proc. of Design Automation Conference*, 2002.

[4] G. Varatkar and R. Marculescu, "Communication-aware task scheduling and voltage selection for total systems energy minimization," *Proc. of International Conference on Computer-Aided Design*, 2003.

[5] B. Gorji-Ara et al., "Fast and efficient voltage scheduling by evolutionary slack distribution," *Proc. of Asia-South Pacific Design Automation Conference*, 2004.

[6] A. Andrei et al., "Overhead-conscious voltage selection for dynamic and leakage energy reduction of time-constrained systems," *Prof. of Design Automation and Test in Europe*, 2004.

[7] W.-C. Kwon and T. Kim, "Optimal voltage allocation techniques for dynamically variable voltage processors," *Proc. of Design Automation Conference*, 2003.

[8] J. Shin and T. Kim, "Technique for Transition Energy-Aware Dynamic Voltage Assignment," *IEEE Transactions on Integrated Circuits and Systems II*, 2006.

[9] D. Shin, J. Kim, and S. Lee, "Intra-task voltage scheduling for low-energy hard real-time applications, *IEEE Design and Test of Computers*, Vol. 18, 2001.

[10] J. Seo, T. Kim, and K. Chung, "Profile-based optimal intra-task voltage scheduling for hard real-time applications," *Proc. of Design Automation Conference*, 2004.

[11] J. Seo, T. Kim, and N. Dutt, "Optimal Integration of Intra and Inter Task Dynamic Voltage Scaling for Hard Real-Time Applications ," *Proc. of International Conference on Computer-Aided Design*, 2005.

[12] D. Shin and J. Kim, "A Profile-Based Energy-Efficient Intra-Task Voltage Scheduling Algorithm for Hard Real-Time Applications", *Proc. of International Symposium on Low-Power Electronics abd Design*, 2001

[13] B. Mochocki, X. S. Hu and G. Quan, "A Realistic Variable Voltage Scheduling Model for Real-Time Applications," *Proc. of International Conference on Computer-Aided Design,* 2002.

[14] T. Ishihara and H. Yasuura, "Voltage Scheduling Problem for Dynamically Variable Voltage Processors," *Proc. of International Symposium on Low Power Electronics and Design*, 1998.

[15] *"http://www.analog.com/library/analogDialogue/archives/30-3/single_chip.html"*

[16] *"http://www.focus.ti.com/lit/an/slyt042/slyt042.pdf"*

Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)
0-7695-2676-4/06 $20.00 © 2006 IEEE