Energy-Optimal Dynamic Voltage Scaling in Multicore Platforms with Reconfigurable Power Distribution Network

Juyeon Kim Taewhan Kim
School of Electrical and Computer Engineering, Seoul National University

Abstract—**This work addresses a new problem of dynamic voltage scaling (DVS) in multicore platforms. We solve the multicore DVS problem, i.e., simultaneously scheduling execution of tasks assigned to cores and determining dynamically-varying voltage levels, with the objective of minimizing total energy consumption of the cores and voltage regulators (VRs) in the reconfigurable VR-to-core power distribution network (PDN) of platform while meeting the arrival/deadline constraint of tasks. Here, the key factors to be exploited for energy saving are (1) *available voltage levels*, (2) *power conversion efficiency curve of VRs*, and (3) *turning on/off VRs*. Specifically, we *formulate the problem of task scheduling with the relation between factors 1, 2, and 3 into a linear programming problem and solve optimally in polynomial time*.**

I. Introduction

One of the most effective techniques to reduce power consumption of CMPs (chip multicore processors) is to dynamically adjust the supply voltage and operating frequency (e.g., [1], [2]). Since the amount of power consumption quadratically decreases as the supply voltage decreases, lowering down the voltage can save power, which is known as dynamic voltage scaling (DVS) technique. Numerous works have developed DVS techniques that were best suited for their underlying target systems with different operating conditions. The target systems considered can be roughly classified according to the availability of single/multiple core(s) and no/single/multiple voltage regulator(s) (VRs). (VR performs a role of converting the voltage level of power source to the voltage level requested by the target core(s).) This work focuses on the DVS problem in multicore platforms with multiple VRs.

VRs can be generally classified into three types by the implementation style and operation mechanism: low-dropout (LDO) regulators, switched-capacitor (SC) regulators, and inductive switching (IS) regulators. (The characteristics of the behavior of three types of VRs can be found in [3]–[5].) One common feature of VRs is that the power loss by VRs is closely correlated with both the amount of output load current and the output voltage level of the VRs. In addition, the power loss is significant, ranging 15%$\sim$60% of the total power of the system. Thus, it is necessary to consider the minimization of power consumption by VRs as well as by cores in DVS.

Choi *et al.* [5] considered the power conversion efficiency of VR to determine voltage levels to minimize the total energy consumption of the system, but the work is limited to single core platforms. Later, Kim *et al.* [6] showed that the potential energy saving can be accomplished if a dedicated VR is allocated to each core in multicore platforms. They formulated the DVS problem into an integer linear programming (ILP). However, this method did not consider the power consumed by the mandatory excessive number of VRs to enable per-core DVS. To overcome the limitation of per-core DVS in [6], Kolpe *et al.* [7] proposed to group the cores in the same voltage-rail, in which they used *K-means clustering* to group cores with similar DVS levels, so that the number of VRs to be installed is reduced. However, constraining the connection of VRs to groups of cores will lose the possibility of energy saving by non-uniform DVS (i.e., different voltage levels) to the cores in the same group. Recently, Lee, Wang, and Pedram [8] attempted to overcome the limitation of the work in [7] by suggesting multicore platforms with reconfigurable VR-to-core power distribution network (PDN), where they tried to make use of the power efficiency curve of VRs and the capability of tuning on/off VRs. Under these platforms, they proposed two optimization methods to maximize the system-wide energy savings: (i) *reactive VR consolidation*, which reconfigures the PDN for maximizing the power conversion efficiency of the VRs in PDN by examining the pre-determined DVS levels for the cores and grouping the cores with the same DVS levels and (ii) *proactive VR consolidation*, which modifies the input DVS levels so that the total energy consumed by the cores and VRs can be saved further without degrading performance. They employed a greedy scheme to the reactive and proactive VR consolation methods.

In this paper, we propose a new DVS algorithm combined with dynamic reconfiguration of VR-to-core distribution network in multicore platforms, so that the total energy consumed by the cores and VRs is minimized. Like the work in [8], for saving energy we exploit the three factors: (1) multiple (available) voltage levels, (2) power conversion efficiency curve of VRs, and (3) turning on/off VRs. However, rather than resorting to greedy *local* approach in [8], we solve the problem *globally* by formulating the DVS problem combined with VR-to-core reconfiguration, considering the relation between factors 1, 2, and 3, into a linear programming (LP) problem and solve it optimally in polynomial time. It should be noted that the strength of our (optimal) LP formulation can be easily extended by slight modifications to other multicore platforms such as having heterogeneous VRs in PDN, arbitrary VR-to-core reconfigurability, arbitrary shape of power conversion

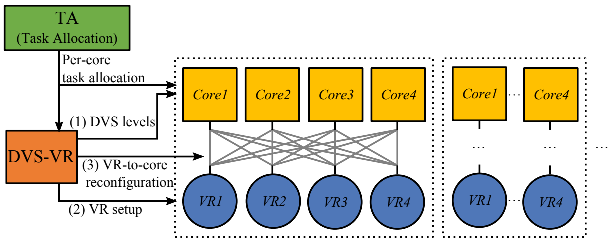


Fig. 1: The conceptual structure of the proposed multicore platform. The structure is identical to that in [8] except that our module DVS-VR *completely* sets the DVS levels and VR-to-core reconfigurations for the per-core tasks allocated by TA (task allocation) module, whereas the work in [8] *modifies* the DVS levels recommended from PM (power management) module to fit into the platform.

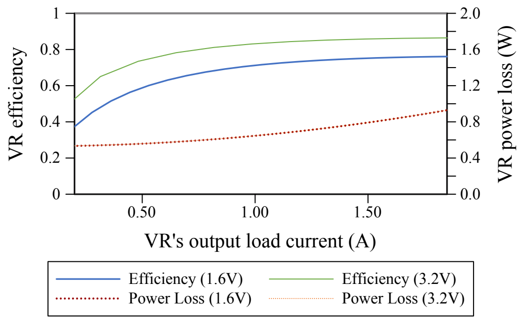


Fig. 2: VR power efficiency (solid lines) and power loss (dotted lines) graphs with respect to the load current.

curve of VRs, and heterogeneous cores.

II. Preliminary and Motivation

Fig. 1 shows a conceptual structure of multicore platform with reconfigurable PDN, in which the first four cores are grouped with the first four VRs, the second four cores with the second four VRs, and finally the last four cores with the last four VRs. The reconfigurable VR-to-core network allocated to each group can deliver power for each core from any VR in the same group. Thus, the reconfigurable PDN provides arbitrary connections from the power output line of any VR to the input power line of any core in the same group. The task allocation (TA) module in Fig. 1 partitions the set of all tasks and distributes the partitioned tasks to cores. The partition and distribution will be guided by several factors such as data/cache coherency, memory hierarchy in cores, and communication protocol between cores. (This work assumes that the per-core task allocation have already been done.) The DVS-VR module in Fig. 1, which we develop in this work, receives three inputs *per-core task allocation*, *per-core power function*, and *per-VR power efficiency curve*[1] and produces three outputs (1) DVS levels (i.e., voltage levels and task schedules), (2) VR setup (i.e., voltage/current values and on/off signal), and (3) VR-to-core reconfiguration.

Since it is generally known that the core power function follows monotonicaly increasing convexity with respect to the applied voltage, but the VR power efficiency curve may not, we assume in this work the VR power efficiency curve can be arbitrary. According to the implementation and operation principle of VRs, the power efficiency function of a type of VR can be derived in terms of the input/output voltages (V_{in}, V_{out}) and currents (I_{in}, I_{out}). For example, for an inductive switching regulator, the power loss (denoted by P_{VR_loss}), in the VR can be expressed as [8]:

$$P_{VR\_loss} = (\alpha_1 D + \alpha_2) I_{out}^2 + \alpha_3 V_{in} + \alpha_4 D + \alpha_5, \quad (1)$$

$$D = \frac{V_{out} + (R_{M2} + R_L) I_{out}}{V_{in} - (R_{M1} + R_{M2}) I_{out}} \quad (2)$$

where R_{M1}, R_{M2}, and R_L are the resistances of the two switches and inductor in the VR, respectively. α_1 through α_5 are the parasitic values specific to the VR. Then, the VR power efficiency (denoted by P_{VR_eff}) is computed by:

$$P_{VR\_eff}(\%) = \frac{Output\_Power}{Input\_Power} = \frac{V_{out} I_{out}}{V_{out} I_{out} + P_{VR\_loss}} \cdot 100\%. \quad (3)$$

Thus, the power efficiency of VR depends on not only the power loss but also the output load current and voltage it produces. For example, Fig. 2 shows how the values of power loss and efficiency for a VR of inductive switching type change as the output load current changes for two output voltages of 1.6V and 3.2V. We can see that the power efficiency of VR is low for the light load current and gradually increases as the load current increases, but settles down beyond 1.0A. Furthermore, For higher output voltages, the power efficiency is better. This observation of VR's power efficiency change implies that the overall power efficiency of VRs in multicore platform can be maximized by checking the values of input load currents and voltages demanded by the cores and properly reconfiguring the VR-to-core power network in a way that some cores are grouped to receive current from a single VR to increase the VR's power efficiency while some VRs are turned off.

We illustrate how the VR-to-core reconfiguration related to DVS can reduce the total energy consumption of the system by using the example shown in Fig. 3. Suppose we have three tasks *Task1*, *Task2*, and *Task3* to be executed in two cores *Core1* and *Core2* whose input currents can be received from *any* of two VRs *VR1* and *VR2*, as specified in Fig. 3(a). The arrival time, deadline, and workload of each task are given as well. For example, *Task1* has arrival time of 0, deadline of 100, and workload of 300 cycles. Let $P(s_1, s_2)$ denote the total power consumed by *Core1*, *Core2*, *VR1*, and *VR2* when *Core1* and *Core2* are supplied with voltages that generate clock frequencies (i.e., clock speeds) s_1 and s_2, respectively. Fig. 3(b) shows the case where *Core1* and *Core2* operate in the same clock speed of 3. Thus, only *VR1* can be used to supply voltage to the two cores[2] and *VR2* can be turned off. Consequently, the total power $P(3, 3) = 220$ is computed as shown at the bottom in Fig. 3(b). Similarly, Fig. 3(c) shows the VR-to-core configuration for computing $P(3, 2) = 180$. Next, let us consider DVS for the tasks. Fig. 3(d) shows the

[1]For simplicity of presentation, we assume that every core has the same power function and every VR has the same power efficiency curve.

[2]Here, we assume that the sum of input currents required by the two cores is within the range in which VR can produce.

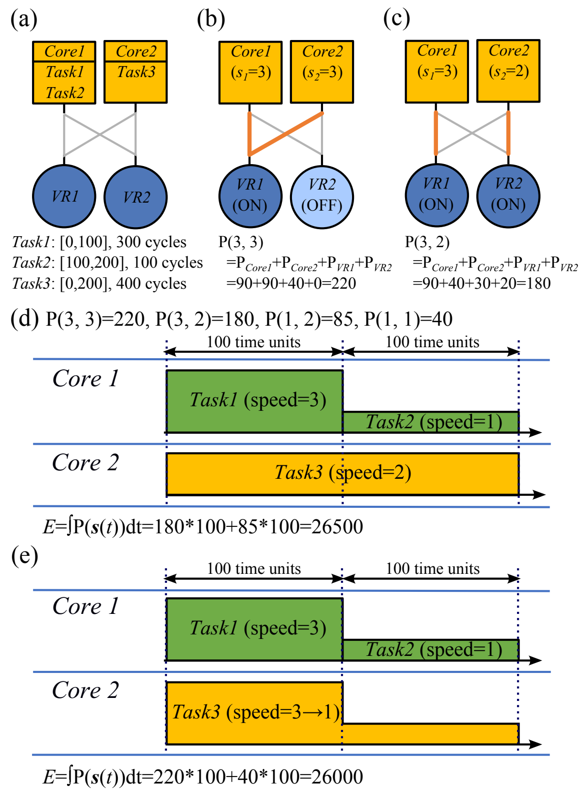

(a)
(b)
(c)
Core1
Task1
Task2
Core2
Task3
VR1
VR2
Core1 (s_1=3)
Core2 (s_2=3)
VR1 (ON)
VR2 (OFF)
Core1 (s_1=3)
Core2 (s_2=2)
VR1 (ON)
VR2 (ON)
Task1: [0,100], 300 cycles
Task2: [100,200], 100 cycles
Task3: [0,200], 400 cycles
P(3, 3) =$P_{Core1}+P_{Core2}+P_{VR1}+P_{VR2}$ =90+90+40+0=220
P(3, 2) =$P_{Core1}+P_{Core2}+P_{VR1}+P_{VR2}$ =90+40+30+20=180
(d) P(3, 3)=220, P(3, 2)=180, P(1, 2)=85, P(1, 1)=40
100 time units
100 time units
Core 1
Task1 (speed=3)
Task2 (speed=1)
Core 2
Task3 (speed=2)
E=∫P(s(t))dt=180*100+85*100=26500
(e)
Task3 (speed=3→1)
E=∫P(s(t))dt=220*100+40*100=26000

Fig. 3: An example of DVS in a multicore platform with reconfigurable PDN. (a) Three tasks with timing constraints and workloads. (b) VR-to-core configuration when both of *Core1* and *Core2* are executed in the clockspeed of 3, and the total power consumed by the cores and VRs. (c) VR-to-core configuration when *Core1* and *Core2* are executed in the clock speeds of 3 and 2, respectively, and the total power consumed by the cores and VRs. (d) *Per-core (VR-unaware) energy optimal* DVS result, and the computation of total energy consumed by the cores and VRs. (e) *Total energy-optimal (VR-aware)* DVS result, and the computation of total energy consumed by the cores and VRs.

DVS result of task schedule and voltage scaling, producing a minimal energy consumption of the cores. (We applied the per-core energy-optimal DVS algorithms in [9], [10] to the tasks.) The DVS result shows that *Core1* consumes energy minimally when *Task1* and *Task2* are executed in the speeds of 3 and 1 for 100 time units, respectively while *Core2* consumes energy minimally when *Task3* is executed in the speed of 2 for 200 time units. Thus, the total energy consumed by the two cores and two VRs is $P(3,2)*100 + P(1,2)*100 = 26500$. On the other hand, Fig. 3(e) shows another DVS result which is aware of the energy consumption on VRs. It is seen that *Task3* is scheduled to be executed in the clock speed of 3 for the first 100 time units and then in the speed of 1 for the remaining 100 time units. The total energy consumption is them $P(3,3)*100 + P(1,1)*100 = 26000$, spending less energy than the VR-unaware DVS result in Fig. 3(d). This example clearly implies that by carefully scheduling tasks and scaling voltage while taking into account the energy by VRs as well as by cores, it is possible to considerably reduce the system's total energy consumption if a large number of VRs are deployed in the power distribution network.

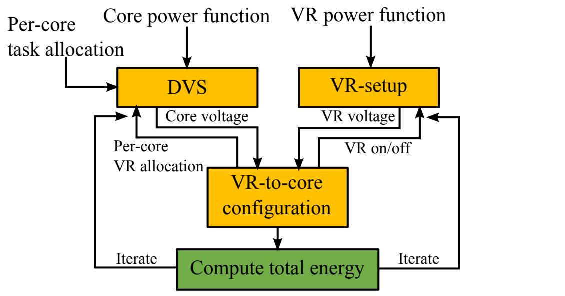


Fig. 4: Diagram showing the interactions of three modules *DVS*, *VR-setup*, and *VR-to-core configuration* associated with the VR-aware DVS problem.

III. VR-AWARE ENERGY-OPTIMAL DVS TECHNIQUE

Problem 1 (**VR-aware DVS**) *For a set of tasks which have been allocated to the cores in a multicore platform with reconfigurable PDN, and a power function of cores and a power efficiency curve of VRs (e.g., Eq.(3)), find (1) execution schedule of tasks, (2) voltage levels of cores, and (3) VR reconfigurations together with the status of VR turning on/off that minimizes the total amount of energy consumed by cores and VRs.*[3]

The VR-aware DVS problem is involved with three inter-related modules. Those modules are *DVS*, *VR-setup*, and *VR-to-core configuration*, as indicated in Fig. 4. The *DVS* initially has the information of per-core task allocation and a core power function, and produces task schedules and voltage levels; the *VR-setup* has a VR power function and produces VR output voltage levels and currents; *VR-to-core configuration* examines the outputs from *DVS* and *VR-setup* and informs the information of per-core VR allocation and the status of VR turning on/off to *DVS* and *VR-setup*, respectively. If there is a further room to save the total energy, the interaction between the three modules will continue.

In the following, we propose a DVS technique, called DVS-VR, which is able to solve the VR-aware DVS problem optimally by formulating the interactions between *DVS*, *VR-setup*, and *VR-to-core configuration* into an LP problem. DVS-VR performs the following three steps.

• **Step 1** (*Preprocessing: Splitting interval and calculating power function*): We divide the whole time intervals assigned to the cores by the arrival times and deadlines of the tasks, for example, as shown in Fig. 5. Let $I_1, \cdots, I_i \cdots, I_{N_I}$ be the resultant sub-intervals. This process has the implication that the interval I_i has a unique set of tasks whose intervals of [arrival time, deadline] overlap with I_i. Thus, we are able to explore the task schedule and voltage assignment on the sub-interval basis without violating energy optimality or the tasks' timing constraint.

In addition, we calculate the set of optimal PDN (= cores + VRs + power network) power values, which will be used in Step 2, for all possible configuration of voltage levels for the cores in the platform. Note that if more than one core in a configuration use the same level of voltage, setting

[3]Note that our DVS technique can handle more than one power function for cores and more than one power efficiency curve for VRs, which will vary depending on the implemented cores and VRs. For simplicity, our presentation uses one power function and one power efficiency curve.

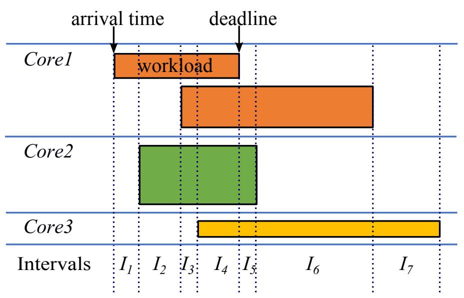


Fig. 5: Splitting the full execution interval into a set of sub-intervals, using arrival times and deadlines as delimiters.

some of VRs can be powered off if sharing a single VR by multiple cores leads to less total power. For example, Fig. 6(a) shows a core-to-voltage configuration, $Conf_1$ = $(V_{Core1}, V_{Core2}, V_{Core3})$ = $(1.0V, 1.2V, 1.0V)$, before setting the turning on/off of VRs. With the consideration of VR turning on/off and the maximum current bound for VRs, a setting of VR turning on/off with optimal power for $Conf_1$ is shown in Fig. 6(b), in which *VR2* is turned off. For N_C number of cores and N_V number of available clock speeds (i.e., voltage levels), $N_V^{N_C}$ is the number of core-to-speed configurations for which we calculate power values together with setting of VRs' turning on/off.

Fig. 7(b) shows all possible core-to-voltage configurations when the multicore platform has two cores and two VRs with voltages 1.0V and 0.8V available. The core-to-voltage configurations can be converted to the corresponding core-to-speed configuration $Conf_i = (s_{i,1}, s_{i,2})$ where $s_{i,1}$ and $s_{i,2}$ indicate the clock speeds supplied to $Core_1$ and $Core_2$ in $Conf_i$, respectively. By using the computed minimal power value together with VR setting for each configuration, the next step is to determine the time length, $x_{i,j}$, for which $Conf_j$ should be run on interval I_i to achieve a minimal total energy consumption.

• **Step 2** (*Linear programming formulation*): As mentioned before, the DVS problem can be reduced to the problem of finding the time duration of each core-to-speed configuration for every interval I_i, $i = 1, \cdots N_I$. The notations used in our LP formulation are summarized as:

- N_C : The number of cores.
- N_S : The number of clock speed configurations for cores.
- N_P : The number of tasks.
- N_I : The number of intervals.
- (a_k, d_k) : The arrival time and deadline of task k.
- W_k : The total workload of task k.
- C_k : The core to which task k is allocated for execution.
- I_i : The i-th interval.
- T_i : The time length of I_i.
- $\vec{s_j}$ (=$Conf_j$) : The j-th core-to-speed configuration ($\vec{s_j} = (s_{j,1}, \cdots, s_{j,N_C})$).
- $P(\vec{s_j})$: The power consumed by the cores with core-to-speed configuration $\vec{s_j}$.
- $x_{i,j}$: The length of time interval spent by $\vec{s_j}$ on I_i.
- $w_{i,k}$: The workload of task k processed in I_i.

Note that $x_{i,j}$ and $w_{i,k}$ are variables and the rest are constants. The objective function and the set of constraints to be satisfied between the variables and constants are expressed below.

1. The objective function is to minimize the total energy consumption:

$$\Sigma_{i=1}^{N_I} \Sigma_{j=1}^{N_S} P(\vec{s_j}) * x_{i,j} \qquad (4)$$

2. $x_{i,j}$ and $w_{j,k}$ should have non-negative numbers. Further, the total time spent by all core-to-speed configurations should equals the length of interval I_i:

$$x_{i,j} \geq 0 \quad \text{for } 1 \leq i \leq N_I,\ 1 \leq j \leq N_S \qquad (5)$$

$$w_{i,k} \geq 0 \quad \text{for } 1 \leq i \leq N_I,\ 1 \leq k \leq N_P \qquad (6)$$

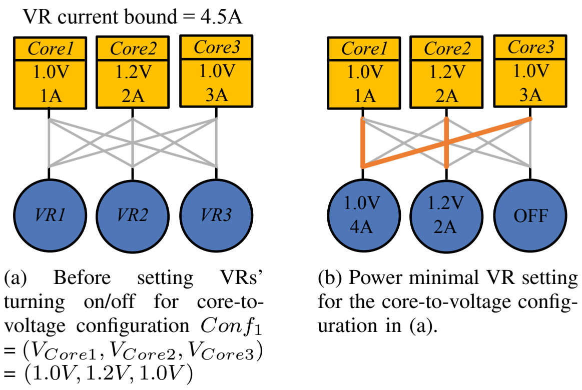
$$\Sigma_{j=1}^{N_S} x_{i,j} = T_i \quad \text{for } 1 \leq i \leq N_I \qquad (7)$$

3. The total workload processed in interval I_i cannot be more than the total sum of the quantities of the clock speed multiplied by the time duration.[4] In addition, the total sum of workload processed on I_i should satisfy the workload constraint:

$$\Sigma_{j=1}^{N_S} s_{j,p} * x_{i,j} \geq \Sigma_{C_k = p} w_{i,k} \quad \text{for } 1 \leq i \leq N_I,\ 1 \leq p \leq N_C \qquad (8)$$

$$\Sigma_{I_i \in (a_k, d_k)} w_{i,k} = W_k \quad \text{for } 1 \leq k \leq N_P \qquad (9)$$

Fig. 7(c) shows the LP formulation for a DVS problem instance in Figs. 7(a) and (b), in which tasks 1 and 2 are allocated to $Core1$ and task 3 to $Core2$, four core-to-speed configurations (in vector notation) and two VRs are used. Fig. 7(d) shows an LP solution of $x_{3,-}$ for I_3 produced from the equations in Fig. 7(c), in which $x_{3,1} = 40$, $x_{3,2} = 50$, and $x_{3,4} = 20$ correspond to the times spent by $Conf_1$, $Conf_2$, and $Conf_4$, respectively. Fig. 7(e) depicts the configurations used for minimal energy consumption across all intervals. Note that both of *Tasks* 1 and 2 should be run in $Core1$ for some time period in interval I_3, which is referred to as a *collective schedule* of multiple tasks.



(a) Before setting VRs' turning on/off for core-to-voltage configuration $Conf_1$ = $(V_{Core1}, V_{Core2}, V_{Core3})$ = $(1.0V, 1.2V, 1.0V)$

(b) Power minimal VR setting for the core-to-voltage configuration in (a).

Fig. 6: Example illustrating the platform's power-minimal VR setting for a core-to-voltage configuration $(V_{Core1}, V_{Core2}, V_{Core3})$ = $(1.0V, 1.2V, 1.0V)$.

[4]We assumed that the average switched capacitances of the tasks are identical.

The number of variables used in LP formulation is $N_I \cdot N_S + N_I \cdot N_P$, and the number of constraints is $N_I \cdot N_S + N_I \cdot N_P$ (for ≥ 0 constraints) added by $N_I + N_I \cdot N_C + N_P$. The next step is to generate an *individual schedule* of every task from the collective schedules while preserving the energy optimality.

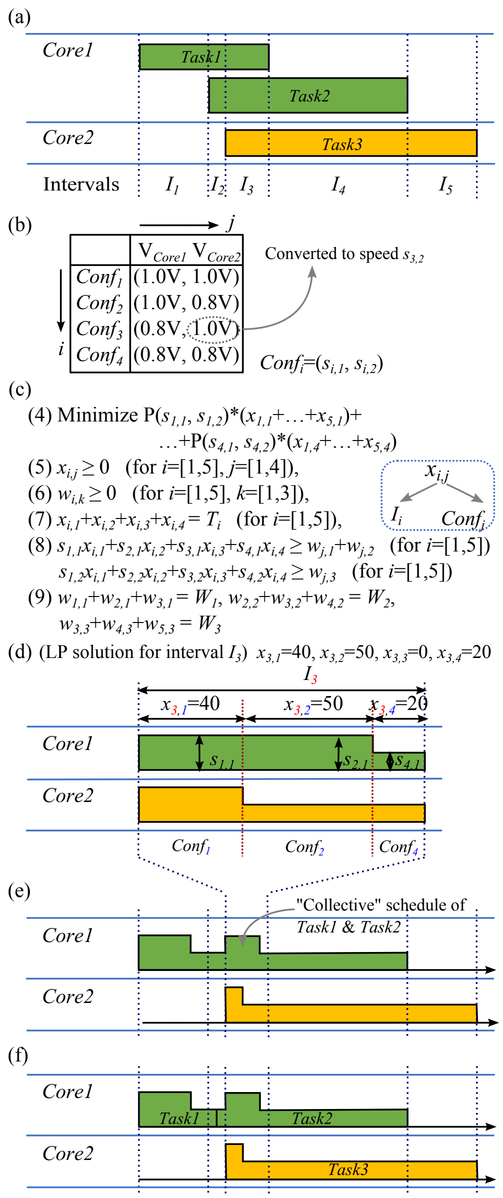

(a)
Core1
Task1
Task2
Core2
Task3
Intervals
I_1 I_2 I_3 I_4 I_5
(b)
j
V_{Core1} V_{Core2}
$Conf_1$ (1.0V, 1.0V)
$Conf_2$ (1.0V, 0.8V)
$Conf_3$ (0.8V, 1.0V)
$Conf_4$ (0.8V, 0.8V)
i
Converted to speed $s_{3,2}$
$Conf_i=(s_{i,1}, s_{i,2})$
(c)
(4) Minimize $P(s_{1,1}, s_{1,2})*(x_{1,1}+\ldots+x_{5,1})+\ldots+P(s_{4,1}, s_{4,2})*(x_{1,4}+\ldots+x_{5,4})$
(5) $x_{i,j} \geq 0$ (for i=[1,5], j=[1,4]),
(6) $w_{i,k} \geq 0$ (for i=[1,5], k=[1,3]),
(7) $x_{i,1}+x_{i,2}+x_{i,3}+x_{i,4} = T_i$ (for i=[1,5]),
(8) $s_{1,1}x_{i,1}+s_{2,1}x_{i,2}+s_{3,1}x_{i,3}+s_{4,1}x_{i,4} \geq w_{j,1}+w_{j,2}$ (for i=[1,5])
$s_{1,2}x_{i,1}+s_{2,2}x_{i,2}+s_{3,2}x_{i,3}+s_{4,2}x_{i,4} \geq w_{j,3}$ (for i=[1,5])
(9) $w_{1,1}+w_{2,1}+w_{3,1} = W_1$, $w_{2,2}+w_{3,2}+w_{4,2} = W_2$,
$w_{3,3}+w_{4,3}+w_{5,3} = W_3$
$x_{i,j}$ I_i $Conf_j$
(d) (LP solution for interval I_3) $x_{3,1}$=40, $x_{3,2}$=50, $x_{3,3}$=0, $x_{3,4}$=20
I_3
$x_{3,1}$=40 $x_{3,2}$=50 $x_{3,4}$=20
Core1
$s_{1,1}$ $s_{2,1}$ $s_{4,1}$
Core2
$Conf_1$ $Conf_2$ $Conf_4$
(e)
"Collective" schedule of Task1 & Task2
Core1
Core2
(f)
Core1
Task1
Task2
Core2
Task3

Fig. 7: Example showing the LP formulation for a DVS problem instance by DVS-VR. (a) Core-to-voltage (core-to-speed) configurations. (b) Generation of time intervals for a DVS problem instance in Step 1. (c) An LP formulation for (a) and (b) in Step 2. (d) One-to-one correspondence between $x_{i,-}$ values produced from Step 2 and core-to-speed configuration in I_i. (e) *Collective schedule* over all intervals produced from Step 2 for (a) and (b). (f) *Individual schedule* produced from Step 3 for the schedule in (e).

• **Step 3** (*Generation of individual task schedule*): Solving LP formulation in Step 2 produces $x_{i,j}$ values. We simply apply the earliest deadline first scheduling to the schedule determined by the $x_{i,j}$ values. It can be easily check that since the schedule never changes the voltage levels imposed by the $x_{i,j}$ values, no change of energy consumption or timing violation occurs. When a new task arrives, we pick the task which has the earliest deadline among the tasks in hand and schedule the selected task first. Since the set of all clock speeds (i.e., voltage levels) of every core determined by the $x_{i,j}$ values is enough to execute the whole tasks, all the timing constraints will be met. Fig. 7(f) shows the individual schedule of tasks from the collective schedule in Fig. 7(e).

IV. Experimental Results

TABLE I: Design specification modeled for the experiments.

Configuration of voltage regulator	
DC-DC controller	TPS40009, PWM (Texas Instruments)
MOSFET	Si4946EY, N-type (Vishay Siliconix)
Inductor	DR74-6R8, 6.8μH (Coiltronics)
Capacitor	20TQC22M, 22μF (Sanyo)
Configuration of CPU core	
Supply voltage	0.8$\sim$3.2V
Operating frequency	100$\sim$400MHz
Static current	100mA
Base power	150mW
Power @ max. supply voltage	5.85W

To verify the effectiveness of the proposed algorithm DVS-VR (VR-aware DVS for multicore platform with reconfigurable PDN), we compared the result produced by DVS-VR with that by the state-of-the-art VR-aware multicore DVS algorithm VRCon [8]. VRCon in [8] receives recommendation of voltage levels for each core from the power management module. Based on the voltage level recommendation, it configures the connection from the voltage regulators to cores to save power consumption. It sometimes increases the voltage level for some cores if it is predicted that the increase enables to save power in the forthcoming execution. The main algorithmic difference between DVS-VR and VRCon [8] is that DVS-VR is an energy optimal DVS while VRCon is heuristic.

Since the efficiency of VRCon is highly dependent on the degree of the accuracy of the voltage recommendation in the power management module and the accuracy can be improved if the module knows the information of as many tasks as possible in advance, for fair comparison we applied a single-core energy optimal DVS algorithm in [10] to find the voltage recommendation, and set the VR-to-core configurations in a way to minimize the total power consumption of the multicore platform with reconfigurable PDN. The modified VRCon also supports the capability of checking if a voltage increase for some cores will save the power and increases the voltage if it does. Our DVS-VR and the modified VRCon [8] were implemented in C on a Linux machine with 8 cores of 3.50GHz Intel i7 CPU and 16GB memory. LP programs were solved using *GLPK*.

The experiments are conducted using three applications TS1 for testing tasks with loose timing, TS2 for testing tasks with moderate timing, and TS3 for testing tasks with tight timing. The tasks in the applications were randomly generated to include 5, 10, and 20 tasks for TS1, TS2, and TS3, respectively. The number of cores installed in the platform is set to 4. (If the platform contains more than 4 cores, they are grouped to have less than 5 in a group. Then, DVS-VR or VRCon can be applied to each group.) The power numbers

TABLE II: Comparison of the energy consumption and run time used by the modified version of VR-aware VRCon [8] and our VR-aware DVS-VR. On average, DVS-VR used 3.8% less energy than VRCon. For tasks with tight timing, the energy saving is 6.1%.

Number of voltages	No. of voltage-to-speed configurations	Task set	VRCon [8]		DVS-VR		Avg. Energy consumption Ratio
			Energy consumption (J)	Run time (s)	Energy consumption (J)	Run time (s)	
4	256	TS1	96.51	<0.01	96.05	0.01	0.995
		TS2	163.89	<0.01	161.96	0.04	0.988
		TS3	277.07	<0.01	266.65	0.15	0.962
6	1296	TS1	99.17	<0.01	95.63	0.06	0.964
		TS2	167.15	<0.01	160.22	0.25	0.958
		TS3	280.44	<0.01	260.90	0.95	0.930
7	2401	TS1	99.16	<0.01	95.69	0.13	0.965
		TS2	166.17	<0.01	159.81	0.53	0.962
		TS3	279.30	<0.01	260.20	1.88	0.932
9	6561	TS1	96.91	0.05	95.37	0.46	0.984
		TS2	163.53	0.05	159.04	1.59	0.973
		TS3	276.76	0.05	258.36	5.28	0.934
Average							0.962

TABLE III: The numbers of voltage transitions by VRCon [8] and DVS-VR, and theoretical bounds.

Task set	#tasks	VRCon [8]	DVS-VR	
		No. trans.	No. trans.	Theor. bound
TS1	5	5.95	5.95	45
TS2	10	13.58	13.73	95
TS3	20	31.85	31.18	195

were calculated according to the equation provided in [5]. Table I summarizes the design specification for the multicore platform we modeled in the experiments.

Table II summarizes the total energies consumed by the task sets and the run times for the DVS schedules produced by VRCon [8] and DVS-VR. On average, DVS-VR used 3.8% less enenrgy than VRCon. Howeverm for tasks with tight timing, the energy saving is 6.1%. This means that as more tasks compete for acquiring cores for execution, the gap of the effectiveness between VRCon and DVS-VR will be larger since VRCon resorts to heuristic DVS while DVS-VR is optimal. Furthermore, the more the number of clock speeds is available, the more DVS-VR will save energy due to the more DVS options. On the other hand, VRCon [8] produced some cases with more energy consumption than necessary as the number of available clock speeds increases. It is because too many clock speed options cause the voltage recommendation module to miss a perfect exploitation of binding multiple cores into a single VR.

Table III summarizes the total numbers of voltage transitions required for the DVS schedules produced by VRCon [8] and DVS-VR, and theoretical bounds. It shows that VRCon and DVS-VR both use much less numbers of transitions over the theoretical bounds. Since VRCon is aware of the minimization of transition overhead, it can be safely concluded that DVS-VR also reasonably well control the voltage transitions.

V. Conclusion

This work proposed an energy-optimal DVS technique for multicore platform with reconfigurable PDN. We showed that the problem was theoretically solvable optimally in polynomial time by successfully formulating it into a linear programming problem. This is the first polynomial-time optimal work that addressed the multi-core DVS problem combined with reconfigurable voltage regulators, and can be used as a basis for solving diverse variants of multicore DVS problem such as selecting types and number of VRs, allocating tasks to cores, and determining a best combination of voltage levels available for VRs. In addition, our DVS result sheded light on the implication that some of variants of the multicore DVS with multiple VRs would be solvable in polynomial time or be effectively solvable. Through experiments, it was confirmed that 3.8$\sim$6.1% reduction of energy consumption was expected over that of the state-of-the-art DVS method for multicore platforms with reconfigurable PDN.

Acknowledgment: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF-2017R1E1A1A03070465).

References

[1] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power cmos digital design," *IEICE Transactions on Electronics*, 1992.
[2] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced cpu energy," in *Mobile Computing*, 1996.
[3] G. A. Rincon-Mora and P. E. Allen, "A low-voltage, low quiescent current, low drop-out regulator," *IEEE JSSC*, 1998.
[4] A. Ioinovici, "Switched-capacitor power electronics circuits," *IEEE Circuits and Systems Magazine*, 2001.
[5] Y. Choi, N. Chang, and T. Kim, "Dc-dc converter-aware power management for low-power embedded systems," *IEEE TCAD*, 2007.
[6] W. Kim, M. Gupta, G.-Y. Wei, and D. Brooks, "System level analysis of fast, per-core dvfs using on-chip switching regulators," in *ACM HPCA*, 2008.
[7] T. Kolpe, A. Zhai, and S. S. Sapatnekar, "Enabling improved power management in multicore processors through clustered dvfs," in *IEEE/ACM DATE*, 2011.
[8] W. Lee, Y. Wang, and M. Pedram, "Optimizing a reconfigurable power distribution network in a multicore platform," *IEEE TCAD*, 2015.
[9] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced cpu energy," in *IEEE Annual Symposium on Foundations of Computer Science*, 1995.
[10] W.-C. Kwon and T. Kim, "Optimal voltage allocation techniques for dynamically variable voltage processors," *ACM TECS*, 2005.