Exploration of Memory-Aware Dynamic Voltage Scheduling for Soft Real-Time Applications

ABSTRACT

Dynamic voltage scaling (DVS) and dynamic power management (DPM) are widely-used techniques to reduce energy consumption in modern computing systems. Although combining these techniques can save more energy, there has not been much work focused on energy-optimal combination of these techniques under variable memory clock frequencies. In this paper, we explore system-wide energy-optimal frequency space, employing a memory-aware DVS technique with a stochastic memory access model on frequency-variable memory devices for an MPEG-4 application. In addition, we propose a simple but practical DVS method, which can be applied to an actual platform.

I. INTRODUCTION

Recently mobile information terminals become more and more popular due to the user's requirement of immediacy in accessing information and multimedia. These devices are actually battery-operated and efficient use of restricted battery resources may be requisite on them. To prolong the lifetimes of batteries in use, we need effective power management techniques. Dynamic power management (DPM) and dynamic voltage scaling (DVS) are most representative low-power techniques controlling power consumption.

DPM is a technique reducing power dissipation by selectively powering down based on the usage patterns for such non-voltage scalable devices as memory chip, disk, etc. In these devices energy consumption depends on variation of not voltages but currents. In the meantime, DVS is a low-power technique to achieve a quadratic energy saving while tolerating linear performance degradation according to the variation of supply voltages in variable voltage processors [1]. As processor energy may take a high percentage of the total system energy consumption and memory chips are reported to be high power consumers in the portable computing systems, DVS algorithms considering power management of external memories have much been studied for the optimal system-wide energy consumption [2, 3, 4]. In [2], memory behaviors in setting CPU speeds are considered and a polynomial-type relation between the clock frequency and the power dissipation is proposed. In [3], synergetic effects are investigated in combining DVS techniques and DPM policies without or with a power-aware memory.

In [4], an elegant integration of DVS and DPM is studied on the basis of the streaming frame requests instead of the external memory requests. In these previous works, they assume external memories are usually supplied with a fixed supply voltage and a fixed clock frequency. However, some recent embedded processors (e.g. XScale) allow limited variation of the clock frequency for the external memories due to synchronization between the main bus linked to the processor and the memory control bus. Therefore, integration of DVS and DPM for frequency-variable memory devices needs to be studied from the viewpoint of the system-wide optimal energy consumption. In [5], a merge of DVS and DPM based on a realistic energy model of external memory devices is studied, and they also propose an analytical energy model depending on not only the processor clock frequency but also the memory clock frequency considering external memory behaviors.

In this paper, we address the problem of memory aware DVS for systems with variable memory clock frequencies. To the best of our knowledge, [5] is the only published work which addressed the combined DVS and DPM for the systems with variable memory clock frequency. However, [5] never considers memory access patterns and adopts a naive DPM scheme for the slack time after completion of a task till a deadline. Our work explores the energy-optimal frequency space using a DVS technique with a threshold-based DPM scheme depending on a stochastic model of memory accesses. In addition, we propose a simple but practical memory-aware DVS method. Experimental results using an MPEG-4 application shows the feasible energy-optimal frequencies vary depending on the degree of DPM energy saving. They also show our technique reduces the total energy consumption by more than 34% over the existing technique.

The rest of this paper is organized as follows. We describe a main motivation of our work in Section II while the energy model for memory-aware DVS and DPM is presented in Section III. We explore the feasible energy-optimal frequency space for an MPEG-4 application in Section IV. Section V concludes with a summary.

II. MOTIVATIONAL EXAMPLE

Feasible energy-optimal frequency assignment with a frequency-variable memory device indicates that lower processor frequency and memory frequency do not always result in lower total energy consumption [5]. This is because memory energy consumption in the used energy model is affected by a trade-off between static energy and the dynamic energy of an external memory device according to the variation of both frequencies. The memory energy consumption is shown to be convex due to static memory energy, and the total system-wide energy consumption is governed by the convexity. However, in general different memory energy model or memory access patterns may affect the total system-wide energy critically with the memory frequency varying.

As a motivational example, we will investigate what the feasible energy-optimal frequency space becomes if a deeper DPM mechanism is applied to the memory device model of the previous work. In Fig. 1, the left picture shows the original feasible solution space of the energy-optimal clock frequency setting on a MPEG-4 decoder program [5]. In this picture, the center of the contours means a globally energy-optimal point and each contour line does the same total energy consumption value. The solid curve shows the relation between the processor frequency and the memory frequency meeting the deadline constraint of a one frame decoding, that is, the deadline is always met in the upper region of this curve. (a) is a frequency assignment ignoring both memory energy and access time. (b) and (c) are frequency assignments considering only memory access time and considering both memory energy and access time, respectively. As (c) is nearest to the center and is above the solid curve, the frequency pair at (c) is the feasible energy-optimal solution. We can notice that the processor frequency at (c) is higher than that at (b), but the energy consumption of (c) is rather lower than that of (b). And, we also notice that the global energy-optimal point is outside the boundary of the feasible solution space.

To obtain a deeper DPM mechanism we augmented the original energy model [5] to enable a memory device to transition into a lower power state after its staying at the idle state during a specified threshold time irrespective of execution of the task. Because memory traces are not available we estimated average interarrival time using the processor execution cycles, memory access count and the deadline published in the previous work, and we employed a stochastically energy-minimizing policy based on an exponential distribution.

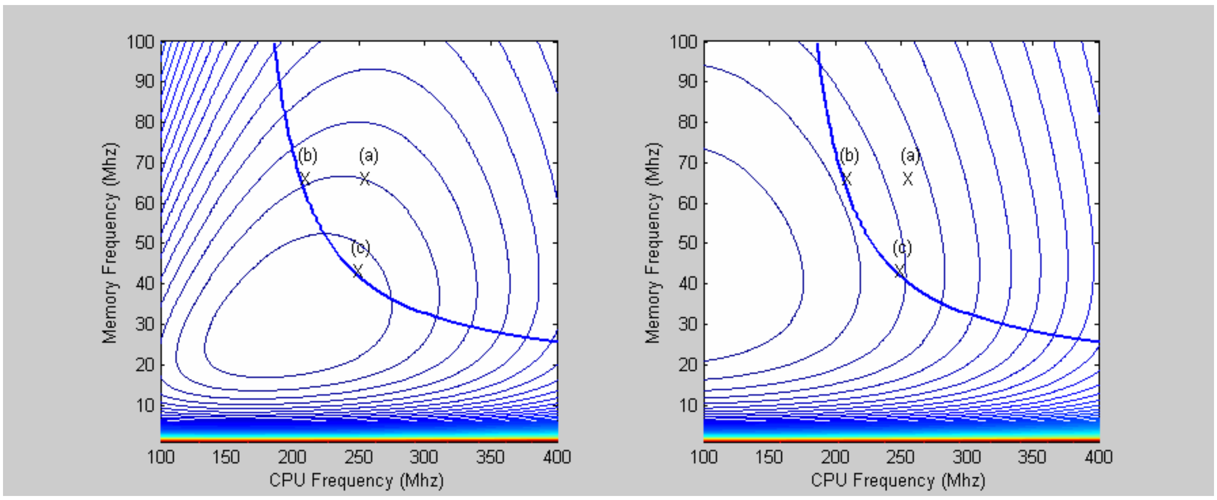


Fig. 1. Original[5] and changed feasible solution space with deeper DPM

The right picture in Fig. 1 shows the changed frequency space when a memory energy model including a threshold-based DPM technique is used. We notice the center of the contours is moved towards a left Y-axis and also notice (b) becomes nearer to the energy-optimal point than (c) at a short glance. That is, although the frequency pairs are the same as those in the original frequency space energy consumption at (b) becomes lower than at (c) actually in the right figure. This phenomenon occurs because more DPM reduces static memory energy and the convexity of the total energy on the frequency pair changes such that a minimum point exists for the processor frequency, however, this does not occur for the memory frequency.

We observed that a deeper DPM mechanism affected the feasible frequency space very much when the memory frequency was varying and it resulted in the movement of the feasible energy-optimal frequency pair. Therefore, it seems important and meaningful to take into consideration which DPM mechanism and how much further DPM we should employ in building an efficient energy consumption model. In the remaining Sections, among many DPM techniques we will study an analytical energy consumption model considering a threshold-based DPM technique based on a stochastic memory access model for a frequency-variable memory device.

III. MEMORY-AWARE COMBININATION OF DVS AND DPM

In this Section, assuming that memory traces are given we first introduce memory states and a memory access model. And, we build a processor energy model and a memory energy model based on them with a memory-aware low power technique combining DVS and DPM considered. Then, we describe how to find the system-wide energy-optimal solution analytically.

1. Memory States and Access Model

We assume that external memory is an SDRAM device which has usually three power states in terms of power dissipation: an active state, an idle (or standby) state, and a powerdown state as shown in Fig. 2.

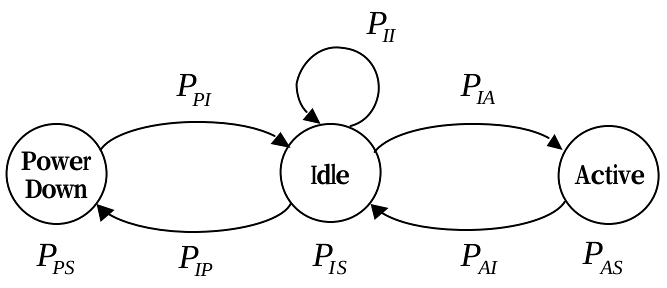


Fig. 2. Power states of an SDRAM device

In Fig. 2, P_{AI} and P_{IA} are the transition power from the active state to the idle state and vice versa respectively. Similarly, P_{IP} and P_{PI} are the transition power from the idle state to the powerdown state and vice versa respectively. P_{II} is the dynamic power due to the necessity for clock propagation during the idle state. P_{AS}, P_{IS}, and P_{PS} are the static power of the active state, the idle state, and the powerdown state respectively. The powerdown state is assumed not to perform any refresh operations.

For the modeling of the memory access patterns, [4] shows approaches purely based on exponential distributions may not model well real system behaviors. However, [6] says that using exponential as an approximation of the real interarival time distribution is sufficient as it produces results consistent with simulation. Based on the observation of [6] we assume that the interarrival time between memory accesses follows an exponential distribution and hence the count of memory accesses follows a Poisson Process, say, N(t). (In practice modeling memory transactions with another stochastic processes is possible, but it is beyond the range and concern of this paper.) Let P[N(t)=n] be the probability that n arrivals occur in time interval of t, then it is presented as the next equation [7].

$$P[N(t)=n]=\frac{(l\,t)^n}{n!}e^{-l\,t} \quad \text{, where } l \text{ is the access arrival rate}$$

If we let T be the random variable of the interarrival time of a Poisson process, its distribution function and density function and are respectively

$$F(x) = P[T \le t] = 1 - e^{-l\,t}\,,\quad f(x)=\frac{dF(x)}{dx}=l\,e^{-l\,t}$$

This means that the interarrival time T is exponentially distributed.

2. Energy Model

Let us assume that while an soft real-time periodic task executes within the deadline, t_d, the processor runs for N_c cycles at the clock frequency of f_c and the external memory is accessed K times as shown in Fig. 3. Here, t_{acc} is the average delay of accessing the external memory when each cache miss occurs and is approximated as the time between when the external memory request arrives and when the request is completely served. The external memory transitions from the active state to the idle state after serving the memory request and does from the idle state to the powerdown state after staying at the idle state for the specified threshold, t_{th}. $t_{interarriva}$ is the difference of time between the present memory request and the next one, and occurs K times during execution of a task. And, the external memory is assumed to be supplied with the memory clock frequency, f_m and have a burst mode with M_b, number of memory clock cycles for a burst-mode transition. Then, the total execution time, t_{exe} can be approximated as $t_{exe} = \frac{KM_b}{f_m} + \frac{N_c}{f_c}$.

2.1 Processor Energy Model

We assume the energy consumption of the processor is proportional to the square of supplied clock frequency. For the total processor energy, we only consider the dynamic processor energy. Then, the processor energy is

$$E_c = a\, C_c\, k^2\, f_c^2\, N_c$$

Where, a is the switching activity, C_c is the switching capacitance, k is the proportional coefficient between the supply voltage (V_{dd}) and the clock frequency (f_c), and N_c is the total execution cycles with no external memory accesses.

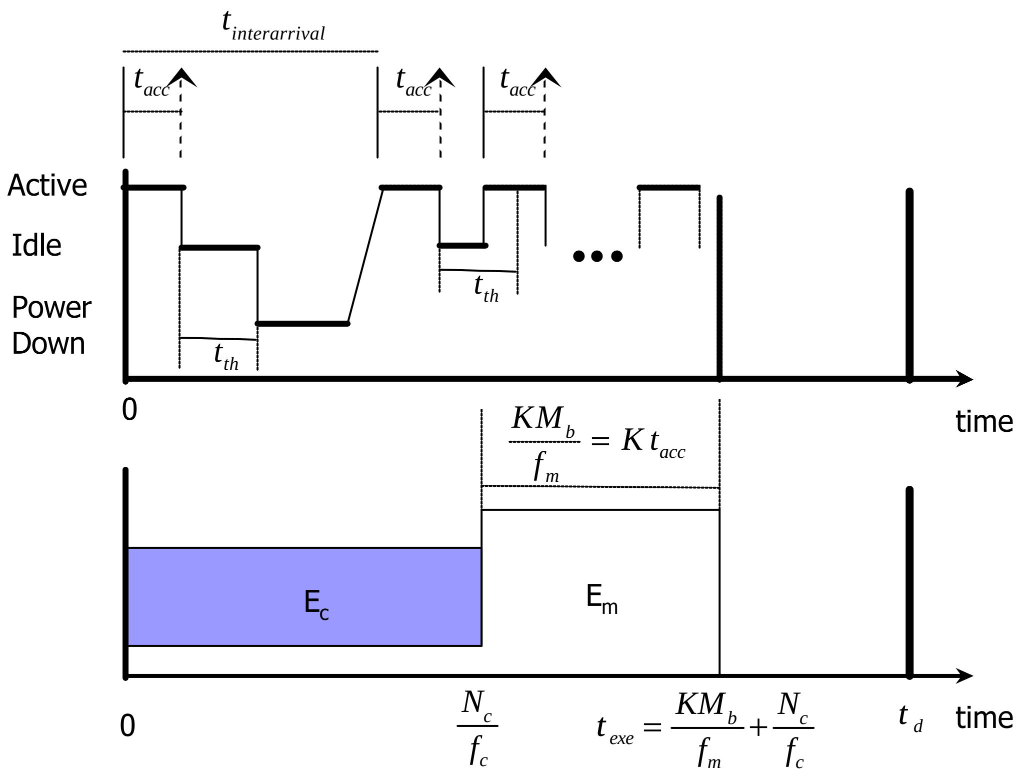


Fig. 3. Processor energy, external memory energy, and memory behaviors

2.2 Memory Energy Model

For the tractable memory energy model, we assume the total number of external memory accesses is given at a fixed value, K and the distribution of the memory access pattern follows the exponential distribution with the access arrival rate, l . And assume that T is a random variable of the interarrival time between memory accesses, as described in subsection 1. It may be more comprehensible to classify the total energy during memory operation into two categories: dynamic energy and static energy.

o Dynamic Energy

Since memory accesses occur *K* times while a task execute we can get the energies $E_{IA}K$ and $E_{AI}K$ easily, which are needed to transition between the idle state and the active state. Next, to derive the dynamic energy at the idle state is a little more elaborate and state and it is assumed as a consumed energy value per elapsed cycle. When the clock frequency of the processor is scaled to f_c as a task runs, the pattern of memory accesses is still maintained but the arrival rate of memory accesses is slowed down and the distribution is spread by the amount of f_c / f_{cmax}. Therefore, the total average interarrival time becomes $\frac{K}{l}\frac{f_{cmax}}{f_c}$ at the frequency f_c while it is $\frac{K}{l}$ at the frequency f_{cmax} . As the number of the total average cycles at the idle state is $f_m P[T \le t_{acc} + t_{th}]\frac{K}{l}\frac{f_{cmax}}{f_c}$, we can obtain the average dynamic energy at the idle state as $E_{II} f_m P[T \le t_{acc} + t_{th}]\frac{K}{l}\frac{f_{cmax}}{f_c}$. We can calculate the transitioning energies between the idle state and the powerdown state as $E_{IP}K\,P[T > t_{acc} + t_{th}]$ and $E_{PI}K\,P[T > t_{acc} + t_{th}]$ respectively. Therefore, the total dynamic memory energy is

$$\begin{aligned} E_{md} = {} & E_{IA}K + E_{AI}K + E_{II}\, f_m\, P[T \le t_{acc} + t_{th}]\frac{f_{cmax}}{f_c}\frac{K}{l} + \\ & E_{IP}K\,P[T > t_{acc} + t_{th}] + E_{PI}K\,P[T > t_{acc} + t_{th}] + E_{IP} + E_{PI} \end{aligned}$$

o Static Energy

First, the static energy at the active state during the memory access can be calculated by $P_{AS}\frac{KM_b}{f_m}$, where KM_b is the total number of memory access cycles and $\frac{KM_b}{f_m}$ is the total memory access time at the active state. Using the probability the average static energy at the idle state is $P_{IS}P[T \le t_{acc} + t_{th}]\frac{K}{l}\frac{f_{cmax}}{f_c}$. Similarly, the average static energy at the powerdown state is $P_{PS}P[T \le t_{acc} + t_{th}]\frac{K}{l}\frac{f_{cmax}}{f_c}$. As during the slack time the memory will stay at the powerdown state, the static energy is calculated as $P_{PS}(t_d - t_{exe}) = P_{PS}(t_d - \frac{N_c}{f_c} - \frac{KM_b}{f_m})$. Therefore, the total static energy is

$$\begin{aligned} E_{ms} = {} & P_{AS}\frac{KM_b}{f_m} + P_{IS}K\,P[T \le t_{acc} + t_{th}]\frac{f_{cmax}}{f_c}\frac{1}{l} + \\ & P_{PS}K\,P[T > t_{acc} + t_{th}]\frac{f_{cmax}}{f_c}\frac{1}{l} + P_{PS}(t_d - \frac{N_c}{f_c} - \frac{KM_b}{f_m}) \end{aligned}$$

Consequently, the total system-wide energy is

$$\begin{aligned} E_t = {} & E_c + E_m = E_c + E_{md} + E_{ms} \\ = {} & a\,C_c k^2 f_c^2 N_c + \\ & E_{IA}K + E_{AI}K + E_{II}\,f_m\,P[T \le t_{acc} + t_{th}]\frac{f_{cmax}}{f_c}\frac{K}{l} + \\ & E_{IP}K\,P[T > t_{acc} + t_{th}] + E_{PI}K\,P[T > t_{acc} + t_{th}] + \\ & P_{AS}\frac{KM_b}{f_m} + P_{IS}K\,P[T \le t_{acc} + t_{th}]\frac{f_{cmax}}{f_c}\frac{1}{l} + \\ & P_{PS}K\,P[T > t_{acc} + t_{th}]\frac{f_{cmax}}{f_c}\frac{1}{l} + E_{IP} + E_{PI} + P_{PS}(t_d - \frac{N_c}{f_c} - \frac{KM_b}{f_m}) \end{aligned}$$

For the exponential distribution, employing $P[T > t] = e^{-\frac{f_c}{f_{cmax}} l t}$ and an approximated formula from the Maclaurin Series [8], the total energy can be rearranged as follows.

$$
\begin{aligned}
E_t \; = \; & \left(a\, C_c\, k^2 N_c + E_{IP}\frac{Kl^2}{2 f_{cmax}^2}(t_{acc}+t_{th})^2 + E_{PI}\frac{Kl^2}{2 f_{cmax}^2}(t_{acc}+t_{th})^2 \right) f_c^2 \\
& + \left(P_{PS}\frac{Kl}{2 f_{cmax}}(t_{acc}+t_{th})^2 - P_{IS}\frac{Kl}{2 f_{cmax}}(t_{acc}+t_{th})^2 - E_{II}\, f_m \frac{Kl}{2 f_{cmax}}(t_{acc}+t_{th})^2 \right) f_c \\
& - \left(E_{IP}\frac{Kl}{f_{cmax}}(t_{acc}+t_{th}) + E_{PI}\frac{Kl}{f_{cmax}}(t_{acc}+t_{th}) \right) f_c + P_{PS}\left(\frac{Kf_{cmax}}{l} - N_c \right)\frac{1}{f_c} + E_{IA}K + E_{AI}K \qquad (1) \\
& + E_{II}\, f_m K(t_{acc}+t_{th}) + P_{AS}\frac{KM_b}{f_m} + P_{IS}K(t_{acc}+t_{th}) - P_{PS}K(t_{acc}+t_{th}) \\
& + E_{IP}(K+1) + E_{PI}(K+1) + P_{PS}t_d - P_{PS}\frac{KM_b}{f_m}
\end{aligned}
$$

3. Finding Energy-Optimal Solutions

To get the optimal energy savings while guaranteeing the deadline of a soft real-time task is met we should delay the total execution time until it reaches the deadline, that is,

$$
t_{exe} = \frac{KM_b}{f_m} + \frac{N_c}{f_c} \le t_d \qquad (2)
$$

The problem to solve is comprised of the total energy function (1) and the time constraint (2) and the total energy is a function of not only the processor clock frequency but also the memory clock frequency. This formula can be considered as a generic form extending [5] in that we make an effort to employ much more chances to power down for the optimal energy savings. Since the total system-wide energy depends on both the processor frequency and the memory frequency and is a convex function of these two variables we can derive the energy-optimal frequencies by the equations $\partial E_t / \partial f_c = 0$ and $\partial E_t / \partial f_m = 0$. By solving these two equations simultaneously we can find the optimal-energy frequency pair (f_c , f_m).

IV. EXPLORING THE FEASIBLE FREQUENCY SPACE

In Section III, we derived analytical system-wide energy-optimal frequency equations reflecting variation of both the processor frequency and the memory frequency. The closed optimal solution can be obtained by solving of high degree simultaneous nonlinear equations. However, this is very complex and incurs large computational overhead, and hence this method is not suitable for online voltage scheduling in actual embedded systems. Therefore, instead of a closed solution we intend to explore the solution space extensively to comprehend the impacts on the energy-optimal frequency pair from variable system behaviour parameters and memory access parameters, and propose a practical memory-aware DVS method based on searching the energy-optimal feasible frequency space.

1. Exploring the Energy-Optimal Frequency Space

The energy-optimal frequency solution is very likely to vary depending on a used energy model, as is seen in the motivational example. In other words, the feasible solution space itself is changeable depending on the memory clock frequency as well as the degree of DPM on external memory devices, memory energy consumption models, memory access patterns, etc. Therefore, we will explore transition of the optimal solution space while altering several variable parameters in the proposed memory energy model of Section III.

First, we assumed the target system consists of a 32-bit RISC processor whose frequency can vary between [100Mhz, 400Mhz] and an SDRAM device with a maximum frequency of 100Mhz. And then, we got memory traces with Simplescalar-ARM available from the University of Michigan [9] for an MPEG-4 decoder selected as a target soft real-time application. While the MPEG decoder ran carphone.mp4 with 33 ms/frame on this simulator, we profiled memory traces and built an exponential model based on the traces, which has 5488.23ns as average interarrival time. The energy consumption or power dissipation values of an SDRAM device from [5] were used. Table 1 summarizes all the basic parameters of the SDRAM device model, the memory access model, and so on. Here, we assume the memory access time to be 60 ns because this value is a usually baseline speed for SDRAM devices.

We investigated the frequency space with the processor frequency and the memory frequency varying with regard to the threshold, the average interarrival time, the processor execution cycles with cache hits, and the number of memory accesses. The parameters in Table 1 were basically applied to all the simulations except the very varying parameter whose impact should be observed.

Table 1. Basic experimental parameters

E_{IA}	110.8 (nJ/access)	E_{AI}	15.8 (nJ/access)
E_{II}	3.14 (nJ/cycle)	E_{IP}	0
E_{PI}	0	P_{AS}	0.151 (W)
P_{IS}	0.072 (W)	P_{PS}	0.0116 (W)
t_{acc}	60 (ns)	t_{th}	300 (ns)
M_b	9	$f_{m,\,max}$	100 (Mhz)
$aC_c k^2$	1×10^{-17} (nJ/hz^2)	N_c	7.09×10^6 (cycles)
K	2,755	$1/l$	5488.23 (ns)

Fig. 4 shows the transition of the energy-optimal point in the frequency space while the threshold t_{th} is changed. In Fig 4, the left upper graph shows the energy-optimal frequency space when the previous DVS method of [5] is applied. The right upper, the left lower, and the last remaining graph show the frequency spaces of the proposed DVS method respectively with 10μs, 1 μs, and 300ns as t_{th}. The graph of the previous method can be regarded as a special case of the proposed method with $t_{th}=\infty$. In other words, our energy model is a generic version of the previous energy model and the powe rdown state is never entered before the slack time occurs in the previous model. As t_{th} becomes smaller we can notice that the total energy becomes a convex function of only the processor frequency.

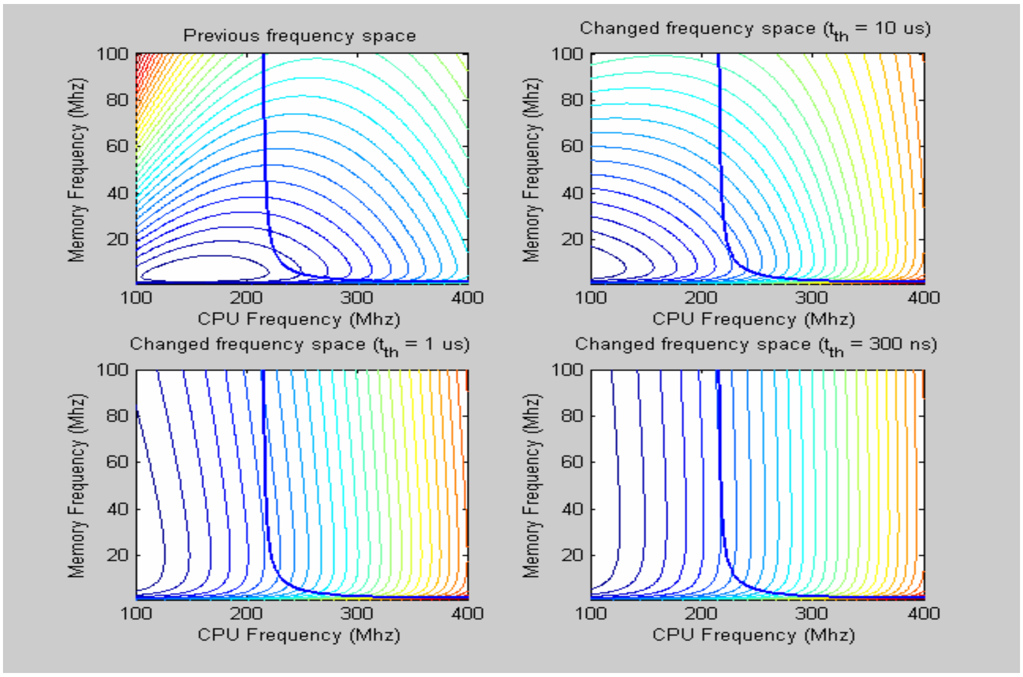


Fig. 4. Variation of the energy-optimal frequency space with the threshold changed

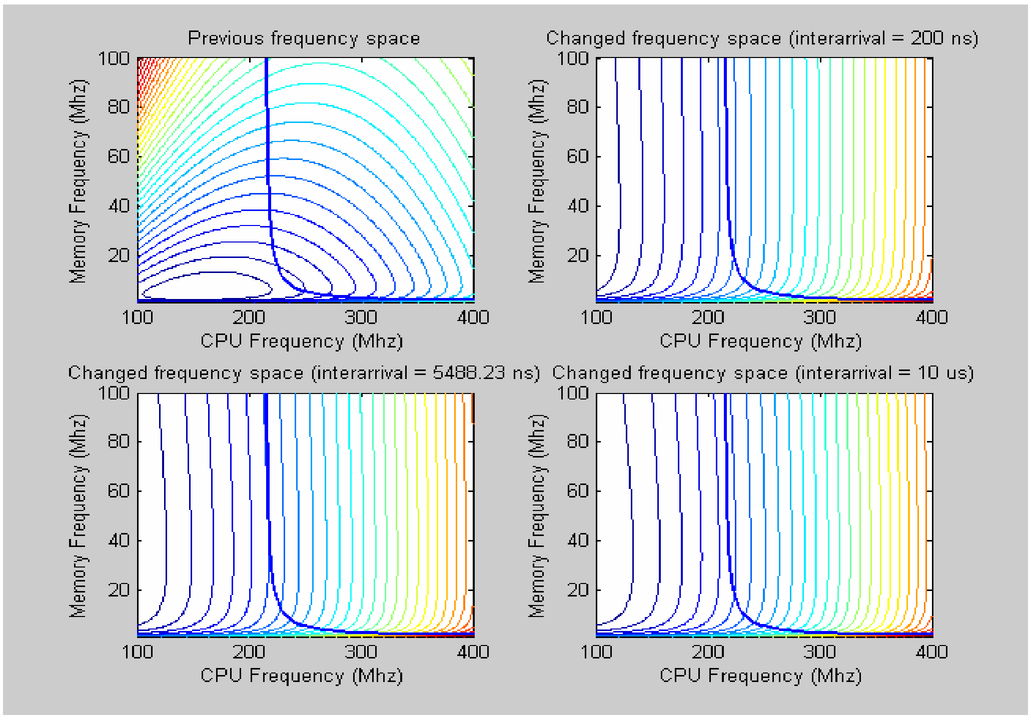


Fig. 5. Variation of the energy-optimal frequency space with the average interarrival time changed

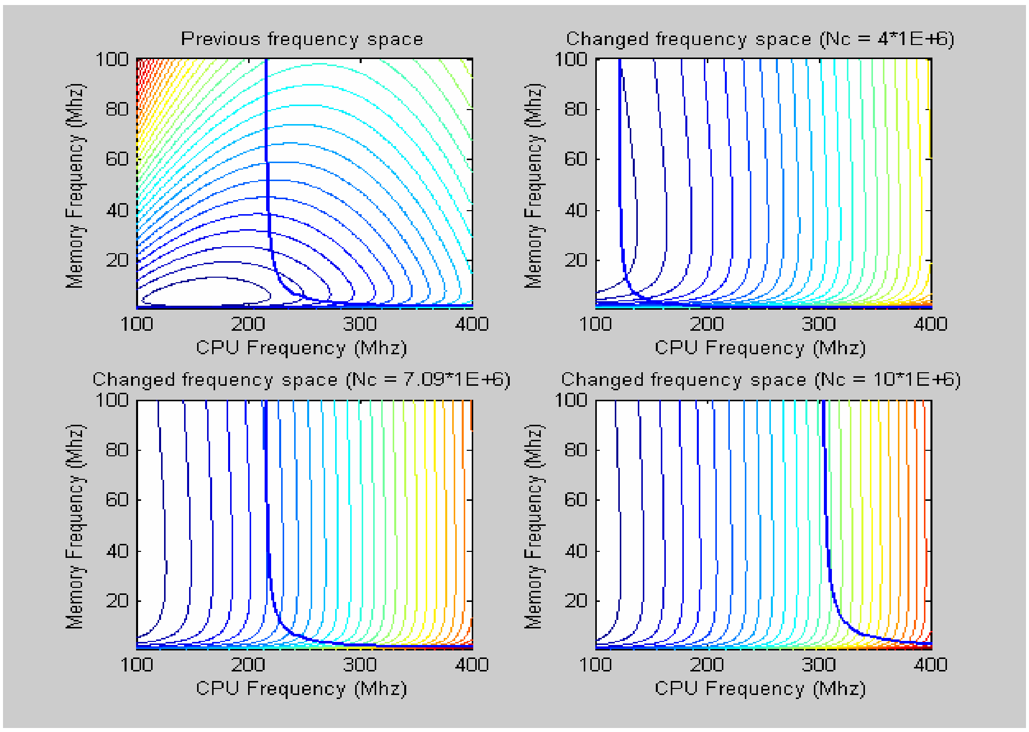


Fig. 6. Variation of the energy-optimal frequency space with the processor cycles changed

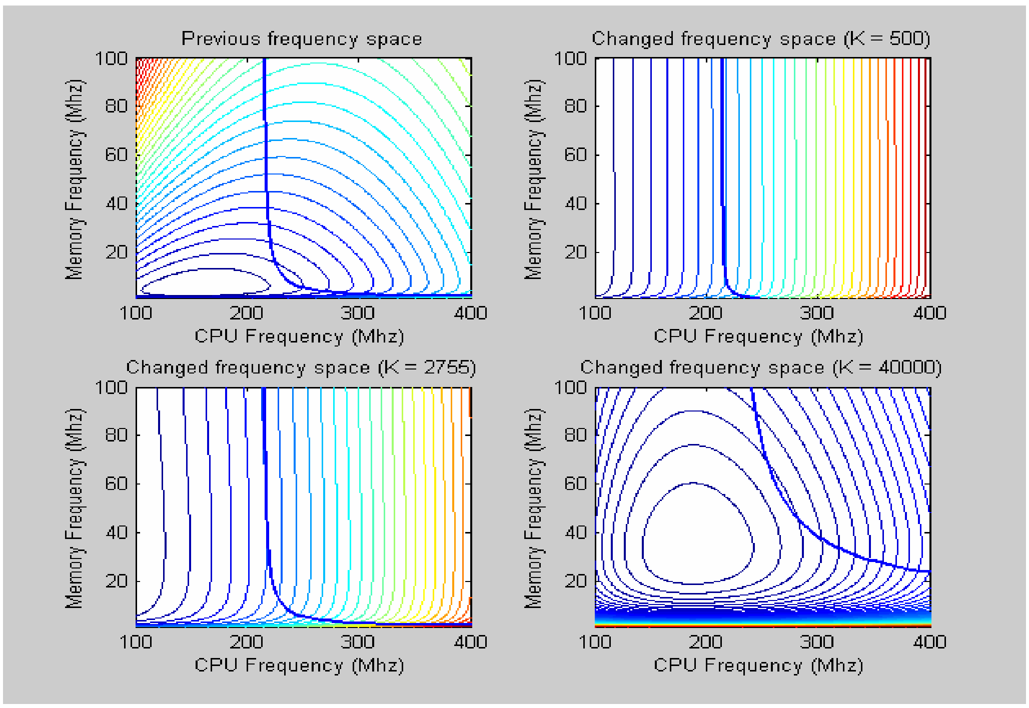


Fig. 7. Variation of the energy-optimal frequency space with the memory access count changed

The solid lines in the whole graphs of Fig. 4 show the relation between the processor frequency and the memory frequency meeting the deadline constraint of eq. (2) as shown in the motivational example. Therefore, we also notice that the feasible frequency solution meeting the deadline is also moved from about (235 Mhz, 10 Mhz) to (220 Mhz, 55 Mhz) roughly when the left upper and the right lower graph are compared. This is caused by a lot of reduction in the static memory energy consumption due to adjustment of the threshold time which decides when the memory device should enter the powerdown state.

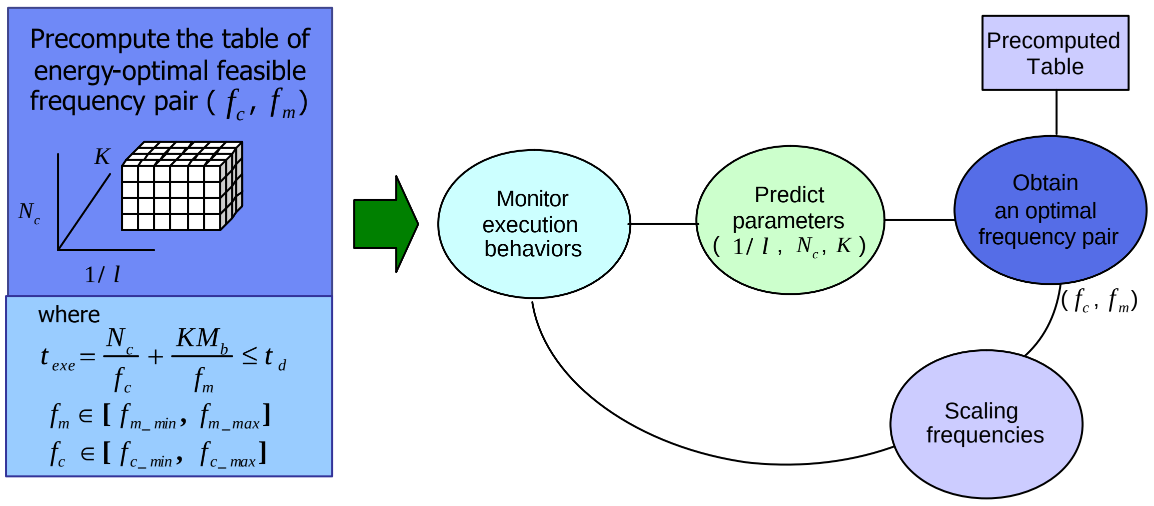
While variation of the threshold can be considered as direct control of the degree of DPM, the remaining parameters are rather related with adjusting the execution behaviours of the target task. Assuming the threshold is already decided, by inspecting relation between the energy-optimal frequency space and these parameters we can get some hints to obtain a close-to-energy-optimal frequency solution available in an actual embedded platform. Fig. 5 shows the movement of the energy-optimal frequency space with the average interarrival time $1/l$ changed. Except the graph of the original frequency space the other graphs don't seem changed much. This indicates DPM on the memory device is well applied by a proper threshold already and the interarrival time may affect not the shape of the contours but the optimal value of the total energy while keeping the shape. In Fig. 6, we notice that the processor execution cycle affects the feasibility of the energy-optimal pair keeping the shape of the space like a case of the interarival time. This is because the processor execution cycle influences directly the limit of the processor frequency, the amount of the processor energy and the static memory energy in eq. (1) and (2). For the varying count of memory accesses, as shown in Fig. 7, when the count is small the space looks like a convex function of only a the processor frequency due to many chances to power down. But, when the count is large in the lower right graph the minimum property appears again. This can be attributed to the insufficient opportunities to be power downed.

2. Practical Memory-Aware DVS Method

Now we propose a simple but practical memory-aware DVS method based on profiling energy-optimal feasible frequency space with memory access pattern parameters varying off-line. Our method is inspired by an event-driven, cruising DVS mechanism per process in [10].

First, we precompute a cubic table whose contents are the energy-optimal feasible frequency pairs off-line. Among the observed parameters in Section III, we select three factors, $1/l$, N_c, and K which govern the system behaviour and model memory accesses because we assume the threshold is already decided. Actually, the used threshold value is 300ns. While meeting the deadline and bounds of each frequency we build a table for the parameters within proper ranges by specified step sizes. $1/l$ varies from 300 to 6,000 ns with 20 steps, N_c and K does from 4×10^5 to 8×10^6 with 20 steps and from 2,000 to 42,000 with 5 steps respectively. Then, we obtain an energy-optimal frequency pair using the precomputed table with $1/l$, N_c, and K predicted for the next interval and can scale the processor frequency and the memory frequency. The overall procedure of our heuristic method is shown in Fig. 8.

To verify performance of our heuristic DVS method we assume that the predicted parameters of the next frame of the MPEG-4 program should be as follows: $1/l$, N_c, and K belongs to [5400ns, 5700ns], [2000, 12000], and [$6.8\times10^5, 7.2\times10^6$] respectively. Real values of these parameters are shown in Table 1. After we get the frequency pairs from the table using the indices mapped in two sets of the bound values of the above parameters, we just interpolate the frequency pairs to obtain the energy-optimal frequency solution for the next frame. If the size of the precomputed table becomes large, that is, the step sizes of variable parameters are increased, we will obtain a more accurate energy-optimal frequency solution. But this requires more computational and spatial overhead.



Precompute the table of energy-optimal feasible frequency pair (f_c, f_m)
K
N_c
$1/l$
where
$t_{exe} = \frac{N_c}{f_c} + \frac{KM_b}{f_m} \le t_d$
$f_m \in [f_{m\_min}, f_{m\_max}]$
$f_c \in [f_{c\_min}, f_{c\_max}]$
Monitor execution behaviors
Predict parameters ($1/l$, N_c, K)
Precomputed Table
Obtain an optimal frequency pair
(f_c , f_m)
Scaling frequencies

Fig. 8. Procedure of a practical memory-ware DVS method

Table 2. Energy-optimal frequencies and energy consumption

DVS method	f_c(Mhz)	f_m(Mhz)	E_c(μJ)	E_m(μJ)	E_t(μJ)	Energy saving rate (%)
Non-DVS	400	100	11,344	7,400	18,744	0
Previous [5]	221	51	3,463	7,872	11,335	38.4
	235	9	3,916	3,788	7,704	58.9
Proposed	221	51	3,463	957	4,420	76.4
	235	9	3,916	1,148	5,064	72.9

In Table 2, a pair of the processor frequency and the memory frequency, (235Mhz, 9Mhz) is the feasible energy-optimal solution of the previous method of [5]. (221Mhz, 51Mhz) represents the feasible frequency pair of the proposed practical method. As can be expected from the observation in Section III, we notice the optimal energy frequency pair is moved to around f_c = 221Mhz and f_m = 51Mhz in comparison with the previous method. We also notice energy saving rates are much higher than those by the previous method. This is because a low processor frequency allows more chances to transition the memory into the lower power state due to the threshold-based DPM mechanism in our method in contrast to the previous method and static memory energy is much saved, and decrease of the processor energy consumption due to the low processor speed makes a direct effect on the total energy consumption. This result confirms again that when the threshold and other memory behavioural parameters are varied the energy-optimal frequency pair moves from the point obtained by the previous DVS method to another point found by the proposed DVS method. Therefore, our method can find a much more energy-efficient frequency pair than the previous method.

V. CONCLUSIONS

For modern embedded microprocessors with variable memory clock frequency as well as CPU clock frequency, a memory-aware DVS heuristic is necessary. In order to develop such a system-wide heuristic, we explored the energy-optimal frequency pair space using an MPEG-4 application.

The main result of our analytical study is that the optimal frequency pair for a given system can vary significantly depending on the degree of DPM energy saving. In particular, we showed how the optimal frequency pair changes to a more close-to-optimal one using our DVS technique with a threshold changed. We also proposed a simple but practical memory-aware DVS heuristic which saves more than 34% energy consumption over the existing method.

REFERENCES

[1] T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors", in *Proc. of International Symposium On Low Power Electronics and Design*, 1998, pp. 197–202.

[2] T. L. Martin, and D. P. Siewiorek, "Nonideal battery and main memory effects on CPU speed on CPU speed-setting for low power", *IEEE Transactions on VLSI Systems*, vol. 9, no. 1, pp. 29–34, Feb 2001.

[3] X. Fan, C. S. Ellis, and A. R. Lebeck, "The synergy between power-aware memory systems and processor voltage", in *Proc. of Power-Aware Computer Systems*, 2003.

[4] T. Simunic, L. Benini, A. Acquaviva, P. Glynn, and G. De Micheli, "Dynamic Voltage Scaling and Power Management for Portable Systems", in *Proc. of Design Automation Conference*, 2001, pp. 524-529.

[5] Y. Cho and N. Chang, "Memory-Aware Energy-Optimal Frequency Assignment for dynamic Supply Voltage Scaling", in *Proc. of International Symposium On Low Power Electronics and Design*, 2004, pp. 387–392.

[6] X. Fan, C. S. Ellis, and A. R. Lebeck, "Memory Controller Policies for DRAM Power Management", in *Proc. of International Symposium on Low Power Electronics and Design*, Aug. 2001, pp. 129-134.

[7] A. Leon-Garcia, *Probability and Random Processes for Electrical Engineering*, Addison-Wesley, 2nd Edition, 1994.

[8] http://mathworld.wolfram.com/MaclaurinSeries.html.

[9] D. Burger and T.M. Austin, "The Simplescalar Tool Set, Version 2.0," *Computer Architecture News*, pages 13–25, June 1997.

[10] A. Weissel and F. Bellosa, "Process cruise control: event-driven clock scaling for dynamic power management", in *Proc. of International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, Oct. 2002, pp.238-246.