School of Electrical, Electronic & Computer Engineering

Stochastic Modeling Of Dynamic Power Management Policies And Analysis Of Their Power-Latency Tradeoffs

Yuan Chen, Fei Xia, Alex Yakovlev

Technical Report Series
NCL-EECE-MSD-TR-2007-123

November 2007

Contact:
yuan.chen1@ncl.ac.uk

Partially Supported by EPSRC grant EP/E044662/1

NCL-EECE-MSD-TR-2007-123
Copyright c 2007 University of Newcastle upon Tyne

School of Electrical, Electronic & Computer Engineering,
Merz Court,
University of Newcastle upon Tyne,
Newcastle upon Tyne, NE1 7RU, UK

http://async.org.uk/

Stochastic Modeling Of Dynamic Power Management Policies And Analysis Of Their Power-Latency Tradeoffs

Yuan Chen, Fei Xia, Alex Yakovlev

November 2007

Abstract

Dynamic Power Management (DPM) is one of the main system level low power technologies for Systems On Chip (SOCs). This paper presents a series of Markov models for various DPM policies and provides accurate analysis about their power and latency performances with full consideration of the shutdown and wakeup processes. The new models make it possible to incorporate latency analysis in terms of deadline satisfaction.

1. Introduction

With processors becoming more sophisticated and more processors integrated onto a single chip, power becomes the bottleneck to system performance and a lot of research concentrates on low power design in SOCs (Systems On Chip). Dynamic Power Management (DPM) has been explored as one of the main system level energy saving technologies [1]. The power on-off control employed by DPM is especially effective in asynchronous systems. For instance, in an on-chip system built in GALS (Globally Asynchronous Locally Synchronous) [2] architecture, DPM can reduce the power consumption by shutting down and waking up individual blocks according to the task and data flow.

The simplest policy implemented in DPM systems is called *greedy policy* [5] which shuts a processor down as soon as the execution of the last task in the processor is completed and wakes up the processor as soon as a new task arrives. Although simple to implement, this policy only works well if the time and energy overheads of waking up and shutting down can be ignored.

More sophisticated DPM policies have been proposed. *Prediction policy*, for example, estimates the arrival moment of the next task when the processor has completed executing the last task and shuts down the processor according to this estimation. Usually, a timeout threshold is set by such DPM systems [6, 7] and the processor will be shut down only when its estimated idle time is longer than this threshold. The prediction quality significantly influences the system performance. More sophisticated prediction algorithms may help achieve lower power consumption, but they may also cost more power and latency to implement.

Another policy, *accumulation and fire* (A&F), described in [4] as the "N-policy", activates the processor when a set number of tasks have been accumulated. This is similar to the *integrate and fire* mechanism found in biological neural systems [13]. This scheme trades latency for power savings.

Modern processors can have more operation states with different levels of speed and power consumption [16]. This provides system engineers extensive options in different and complex DPM policies. This variety and complexity also makes it difficult to analyze the effectiveness and impact of these schemes.

Recent research tends to consider DPM as Markov processes [3] and use stochastic models in analyzing the performance of DPM systems. At present, the M\M\1 model in queuing theory [10] serves as the basic and primary model used in power analysis. However, without integrating power management (PM) transitions (shutdown and wakeup) into its state transition diagram, the analysis results achieved from this model is rather sketchy and inaccurate. Attempts have been made to improve this model. In [5], Ren et al consider the energy cost of PM transitions only when the processor is in task-free period and make its power estimation of greedy policy useful only when the transition cost is very small. The state transition diagram of [8] gives a thorough analysis of the time overhead of PM transitions while the energy cost of these transitions is ignored. When the energy cost is analyzed in Qiu's model [4], the effect of new task arriving during PM transitions is not taken into consideration. Most of these omissions and simplifications are too restrictive and/or theoretically inconsistent (e.g. PM transitions may be included somewhere in the model but disregarded elsewhere in the same model). Furthermore, little attention has so far been paid to the computational complexity of solving these models in analysis with numerical methods. With the increased potential sophistication in power saving policies and more accurate and complex models, more efficient and reliable solution techniques are needed.

DPM policies, while helping to reduce power consumption, may increase system latency. Existing modeling and analysis of DPM policies [4, 5] tend to use the length of queue of waiting tasks as the measure of latency. This is because it is directly represented in the Markov models and thus easy to study. However, how well systems satisfy task deadline requirements is a much better indication of latency performance and traditionally much more widely used in system design. Unfortunately, deadline satisfaction rates have not been studied using Markov models so far.

The main contributions of this paper are: We provide a method of completely representing PM transition states in Markov models. This helps us build numerically well-behaved models for different DPM policies in a standard fashion. The resulting models avoid the inconsistent and unreliable omissions and approximations in previous work. By using such models on three different DPM policies we demonstrate their applicability and discover the behaviors of these policies under different operating conditions with better accuracy for the first time. We also analyze latency in terms of deadline satisfaction for the first time, and show that our models can be effectively used to carry out unified power and latency analysis.

The rest of this paper is organized as follows: Section 2 describes the new Markov models for greedy (Section 2.1), predictive (Section 2.2) and A&F (Section 2.3) policies respectively. This part also includes the comparison of power performance of different policies. Section 3 is about the analysis of system latency for different DPM policies and investigates the combined power and latency performance. Conclusions and future work plans are given in Section 4.

2. Markov Models for DPM Systems

2.1 Greedy Policy

The transition-state-flow diagram of our new Markov model for greedy policy is given in Figure 1. A processor is in state 0 when it has no tasks to be executed. In this state, the processor is sleeping to save power until a new task arrives. Similar to the M\M\1 model, we assume the arriving rate of new tasks to be λ. However, when the time overhead of wakeup is considered, the processor does not move directly from state 0 to state 1 when a new task arrives as in the M\M\1 model. In our model, the waking up process is described by a series of explicit *wakeup* states wu1, wu2 and so on. These wakeup states represent the behavior of the processor when a number of tasks arrive before the completion of the wakeup process.

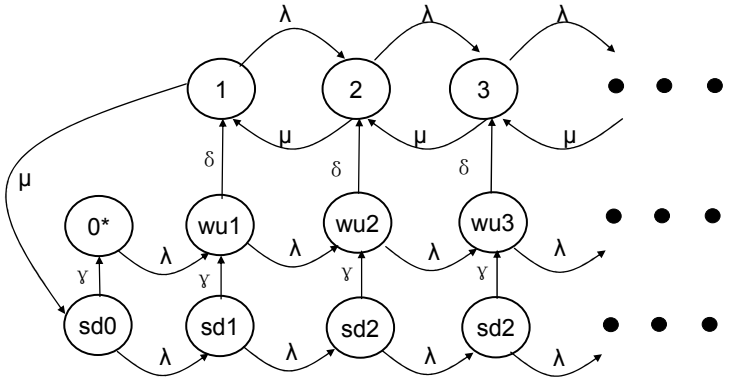

Figure 1: New model for greedy policy

Power consumption behavior is most relevant in portable devices where batteries serve as the main energy source. The voltage and operating frequency that a battery can support depend on a large number of non-deterministic factors such as the battery recovery effect [9]. More recently, energy harvesting [18] has gained popularity and this type of power source is also non-deterministic depending on multiple independent factors. In addition to being affected by power supply uncertainties, wakeup and shutdown also depend on data (e.g. the duration of a shutdown may depend on the amount of data needing to be saved) which should best be viewed as non-deterministic at system design time. Therefore, similar to [4], we also consider the wakeup and shutdown processes to be stochastic, similar to the task arrival and service processes. In Figure 1, γ is the rate of shutdown and δ is the rate of wakeup.

In this model, the processor will move from state 0 first to wu1 when a new task arrives, thus starting its wakeup process. If tasks arrive during the wakeup period, the processor will move across the wakeup states to wu2, wu3 and so on. After the wakeup period finishes, the processor starts its execution of all tasks accumulated.

In the paper, states n (n>0) are *active* states because the processor is active when it stays in any of these states. And state 0 is the *inactive* state. When the last task is completed, the processor will move from state 1 to *shutdown* states sd0, sd1 and so on. Similar to wakeup states, the index number in shutdown states represents the number of tasks accumulated so far during the shutdown process. When the processor is in state sd0, it moves to state 0 if no tasks have arrived on completion of the shutdown. Otherwise it may move to state sd1 or further. If the shutdown process is completed *not* in sd0, the processor will immediately enter the appropriate wakeup state.

If S_i, S_a, S_{sd} and S_{wu} are the sums of probabilities of all inactive, active, shutdown and wakeup states respectively, the estimated average power consumption can be expressed in E2-1.

$$\overline{P} = S_i * P_s + S_a * P_w + S_{wu} * P_{wu} + S_{sd} * P_{sd} \quad \text{(E2-1)}$$

In E2-1, the power consumption when the processor is working, sleeping, waking up and shutting down are P_w, P_s P_{wu} and P_{sd} respectively. In order to differentiate power and probability in this paper, Q is used to represent probability or probability distribution while P is used for power consumption.

This kind of Markov model allows the probabilities of all states to be expressed in terms of one of them. For example, we can get $Q_{sd0} = \mu Q_1 /(\gamma + \lambda)$, where Q_{sd0} and Q_1 are the probabilities of state sd0 and 1 respectively. This will significantly reduce the computational complexity of solving the model numerically [10]. The regular structure of the model in **Erreur ! Source du renvoi introuvable.** allows us to readily express the probabilities of all states in terms of Q_1, the probability of state 1. E2-2 to E2-5 are the expressions for the probability of every state in different groups. The sums of probabilities in different groups are given in E2-6 to E2-8. Finally the value of Q_1 used in these equations is derived by setting the total probability of all states to 1 and this is given in E2-9.

$$Q_{0^*} = S_i = \frac{\mu\gamma}{\lambda(\lambda+\gamma)} Q_1 \tag{E2-2}$$

$$Q_{sd(n)} = \frac{\mu}{\lambda+\gamma} * (\frac{\lambda}{\lambda+\gamma})^n Q_1 \qquad (n \geq 0) \tag{E2-3}$$

$$Q_{wu(n)} = \frac{\gamma}{\lambda+\delta} Q_{sd1} * \sum_{k=0}^{n-1} [(\frac{\lambda}{\lambda+\gamma})^k (\frac{\lambda}{\lambda+\delta})^{n-k}] + (\frac{\lambda}{\lambda+\delta})^n Q_{0^*} \qquad (n \geq 1) \tag{E2-4}$$

$$Q_n = \sum_{k=1}^{n} (\frac{\lambda}{\mu})^{k-1} Q_1 - \frac{\delta}{\mu} [\sum_{k=1}^{n} \sum_{s=k}^{n} (\frac{\lambda}{\mu})^{n-s} Q_{wu(k)}] \tag{E2-5}$$

$$S_a = \sum_{n=1}^{\infty} Q_n = \frac{\mu}{\mu-\lambda} [1 + \frac{\lambda}{\delta} + \frac{\lambda^2}{(\lambda+\gamma)\gamma}] Q_1 \tag{E2-6}$$

$$S_{wu} = \sum_{n=1}^{\infty} Q_{wu(n)} = \frac{\mu}{\delta} Q_1 \tag{E2-7}$$

$$S_{sd} = \sum_{n=0}^{\infty} Q_{sd(n)} = \frac{\mu}{\gamma} Q_1 \tag{E2-8}$$

$$Q_1 = \frac{1}{\frac{\mu}{\gamma} + \frac{\mu\gamma}{\lambda(\lambda+\gamma)} + \frac{\mu}{\delta} + \frac{\mu}{\mu-\lambda}[1 + \frac{\lambda}{\delta} + \frac{\lambda^2}{(\lambda+\gamma)\gamma}]} \tag{E2-9}$$

The average power consumption of greedy policy can be derived by integrating E2-6 to E2-8 into E2-1, and when the time overhead of both shutdown and wakeup process can be ignored ($\gamma \to \infty, \delta \to \infty$), E2-1 becomes E2-10 which is the average power consumption obtained from the M\M\1 model.

$$\overline{P} = \frac{\lambda}{\mu} P_w + \frac{\mu-\lambda}{\mu} P_s \tag{E2-10}$$

Unlike previous work, our model does not assume a limited number of shutdown and wakeup states because potentially an unlimited number of tasks may arrive during a single shutdown or wakeup period. This removes one of the key inconsistencies suffered by previous models. And it enables us to derive closed form analytical formulas such as E2-7 and E2-8 which behave very well numerically. The inconsistent simplifications of previous work are consequently difficult to justify.

2.2 Prediction Policy

Greedy policy is relevant when the time and energy overheads of both shutdown and wakeup processes can be ignored. It may increase system power consumption when the average power consumption of both shutdown

and wakeup processes is greater than the processor's working power consumption ($P_{sd} + P_{wu} > 2P_w$) [11]. In this case, the energy gained by short sleeping gaps cannot overcome the energy overhead of shutdown and wakeup.

To avoid such short sleeping gaps, prediction policy shuts down the processor only when the estimated arriving moment of next task will be later than some threshold value. One common prediction scheme [17] shuts down the processor when a time gap of τ has passed after the completion of the last task without a new task arriving. Advanced prediction policies can dynamically adjust the value of τ and some of them have been successfully implemented into chip designs [11].

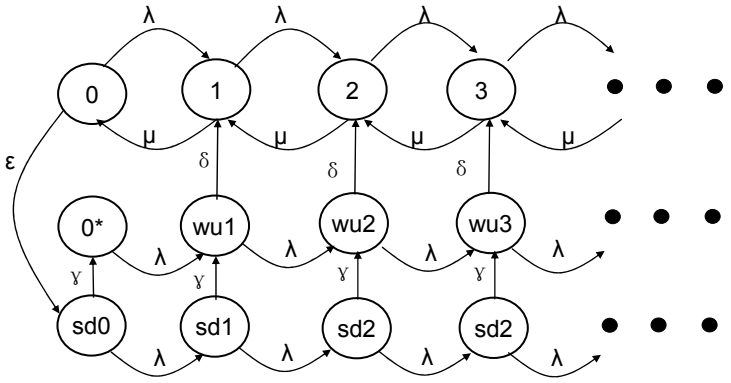

Figure 2: New model for prediction policy

Figure 2 gives our analysis model of prediction policy. Compared to the greedy model in Figure 1, a new state 0 is added in Figure 2. This new state represents the situation when the processor is not shut down but has no tasks to process. In order to differentiate this state from the state when the processor is sleeping, a star mark (*) is attached to the true sleeping state. When the processor is in state 0, it may be shut down if the estimated task free period is long enough and $\varepsilon=\tau^{-1}$ represents the rate of long task free period distribution. If the processor is to be shut down, its actions are similar to that in the model in Section 2.1. If the estimated task free period is very small, the processor will stay in state 0 waiting for a new task to arrive before it moves to state 1 again.

Here, the probabilities of all states are expressed in terms of Q_0 which is the probability of state 0. E2-11 to E2-14 gives the probabilities of different groups and E2-15 is the expression for Q_0.

$$S_i = \frac{\gamma\varepsilon}{\lambda(\lambda+\gamma)}Q_0 \qquad \text{(E2-11)}$$

$$S_a = \{\frac{\varepsilon}{\mu-\lambda}[1+\frac{\lambda}{\lambda+\gamma}(\frac{\lambda}{\delta}+\frac{\lambda}{\gamma}+\frac{\gamma}{\delta})]+\frac{\mu}{\mu-\lambda}\}Q_0 \qquad \text{(E2-12)}$$

$$S_{sd} = \frac{\varepsilon}{\gamma}Q_0 \qquad S_{wu} = \frac{\varepsilon}{\delta}Q_0 \qquad \text{(E2-13,14)}$$

$$Q_0 = \frac{1}{\frac{\varepsilon}{\gamma}+\frac{\varepsilon\gamma}{\lambda(\lambda+\gamma)}+\frac{\varepsilon}{\delta}+\{\frac{\varepsilon}{\mu-\lambda}[1+\frac{\lambda}{\lambda+\gamma}(\frac{\lambda}{\delta}+\frac{\lambda}{\gamma}+\frac{\gamma}{\delta})]+\frac{\mu}{\mu-\lambda}\}} \qquad \text{(E2-15)}$$

Next we apply these models on a real example processor. We choose the FUJI MHF 2043AT which was used in both [11] and [12]. Table 1 gives the key parameters of this processor.

P_s Watt	P_w Watt	T_{wu} sec	T_{sd} sec	P_{wu} Watt	P_{sd} Watt
0.13	0.95	1.61	0.67	2.85	0.54

Table 1: Fujitsu MHF2043AT

Here, we use $\gamma=1/T_{sd}$, $\delta=1/T_{wu}$, and we pick two different values of τ ($\tau_1=2\lambda^{-1}$ from [4] and τ_2=30s [11]) for the prediction policy. In our analysis, the average execution rate of the processor μ has been normalized to 1 and Figure 3 shows the variation of average power consumption $\overline{P}$ with the change of λ.

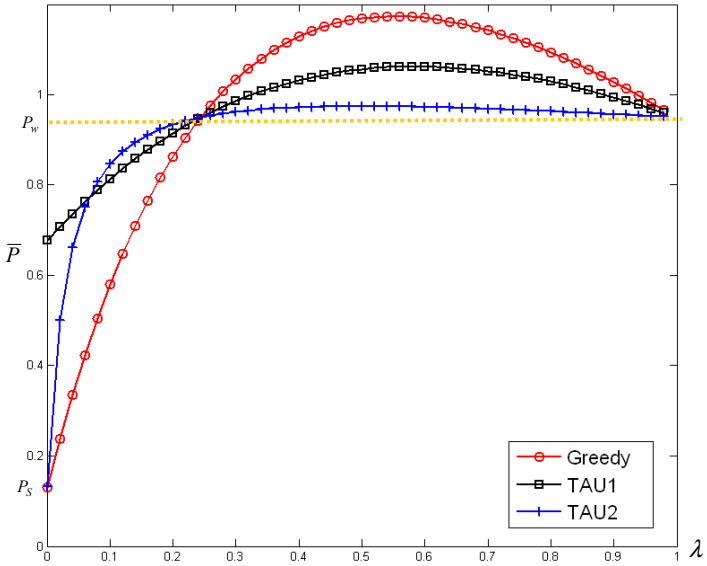

Figure 3: $\overline{P}$ in different PM policies

From Figure 3, we can see that the curves cross P_w at the same point. When λ is small, τ prediction policy is worse than greedy policy because it may fail to shut down even when the actual inactive period is long. Both greedy and constant τ policies consume the same power P_s (0.13W) when λ approaches 0 because the system sleeps completely, but λ^{-1}-dependent τ does not have this feature.

The parameters of MHF2043AT satisfy $P_{sd}+P_{wu}>2P_w$ (Table 1). Therefore, the power curves go above P_w (0.95W) when task arrival is fast enough. From here on, prediction policy shows its advantage over greedy policy. It can be seen from Figure 3 that with prediction the system still cannot reduce its energy consumption under P_w because τ prediction is not perfect. When λ approaches μ, the processor becomes too busy to execute all arriving tasks and has no time to sleep. The curves therefore join at P_w.

For τ prediction, the value of τ affects the power behavior. Increasing τ trades more loss when λ is low for gains when λ is high and λ^{-1}-dependent τ pays too much a price when λ is low. Very large τ implies that the system is almost never shut down with power close to P_w for all λ. In the rest of this paper we use constant τ_2=30s (TAU2 in the figure legends) as a prediction policy example.

So far we have assumed that the task arrival rate λ is constant. However, in real SOCs, λ may vary from time to time according to external environment changes. And the comparison of policies should be based on their performance in the full range of λ. Therefore, we have the following equations:

$$\overline{P}_{gre}=\int_l^h \overline{P}_{gre,\lambda}d\lambda/(h-l) \qquad \text{(E2-16)}$$

$$\overline{P}_{pred}=\int_l^h \overline{P}_{pred,\lambda}d\lambda/(h-l) \qquad \text{(E2-17)}$$

$\overline{P}_{gre,\lambda}$ and $\overline{P}_{pred,\lambda}$ represent the average power consumption of greedy and prediction policies which are calculated in Sections 2.1 and 2.2 respectively. And h and l are the upper and lower bound of λ variation respectively. Since DPM is meaningful only when the processor has much more execution capability than

required by the task arrival rate, we choose h=0.1μ and l=0.5μ in our analysis. In this case, $\overline{P}_{gre,\lambda}$=0.964W and $\overline{P}_{pred(\tau 2),\lambda}$=0.943W which shows prediction policy can be more efficient than greedy policy but the advantage is limited. To achieve better than P_w performance in both policy we need to reduce h and/or l. This may cause the prediction policy to perform less well than the greedy one. This is a motivation for more sophisticated policies.

2.3 Accumulation & Fire Policy

Considering the energy overhead of Power Management, a natural improvement is to reduce the frequency of shutdown and wakeup processes. In the A&F policy [4], a processor is not woken up immediately when a new task arrives. Instead tasks will be accumulated. The accumulation will continue until a certain limit (which is called the accumulation limit) is reached. The processor is then activated to batch process all accumulated tasks. And the time of activation is called the *fire* moment in this policy. Greedy policy can be seen as a basic A&F policy whose accumulation limit is 1.

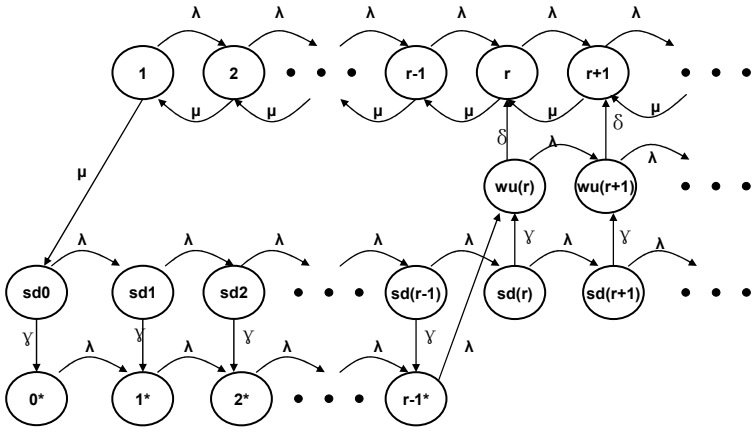

Figure 4: Markov Model for A&F policy

Figure 4 shows the model of A&F policy. Compared to the model given in Figure 1, the major difference is that sleeping extends from only one state (0*) in greedy policy to a group of r states (r>0 is the accumulation limit).

Using the same solution technique given in Sections 2.1 and 2.2, we derived the expressions for the sums of probabilities in different groups in E2-18 to E2-22. All these equations will become their counterparts in Section 2.1 when r=1.

$$S_i = \sum_{n=0}^{r-1} Q_{n^*} = \frac{\mu}{\lambda} * [R - \frac{\lambda}{\gamma}(1 - (\frac{\lambda}{\lambda+\gamma})^R)] * Q_1 \quad \text{(E2-18)}$$

$$S_{sd} = \sum_{n=0}^{\infty} Q_{sd(n)} = \frac{\mu}{\gamma} Q_1 \quad \text{(E2-19)}$$

$$S_{wu} = \sum_{n=r}^{\infty} Q_{wu(n)} = \frac{\mu}{\delta} Q_1 \quad \text{(E2-20)}$$

$$S_a = \sum_{n=r}^{\infty} Q_n = \frac{\mu}{\mu-\lambda}[R + \frac{\lambda}{\delta} + \frac{\lambda^{R+1}}{(\lambda+\gamma)^R \gamma}] Q_1 \quad \text{(E2-21)}$$

$$Q_1 = \frac{1}{\frac{\mu}{\gamma} + \frac{\mu}{\lambda}[R - \frac{\lambda}{\gamma}(1 - (\frac{\lambda}{\lambda+\gamma})^R)] + \frac{\mu}{\delta} + \frac{\mu}{\mu-\lambda}[R + \frac{\lambda}{\delta} + \frac{\lambda^{R+1}}{(\lambda+\gamma)^R \gamma}]} \quad \text{(E2-22)}$$

When we integrate E2-18 to E2-21 into E2-1, we can derive $\overline{P}$ for A&F policy and the effectiveness of all three policies so far is compared in Figure 5. From this figure, It is clear that A&F policy (r>1) has potential advantages over the others. Compared with the other two policies, A&F policy has wider usable range (the range

of λ when $\overline{P} < P_w$), and sometimes lower worst case value (the highest $\overline{P}$ in the full λ range). And this trend continues with the increase of accumulation limit r. Given the same parameters in Section 2.2 when calculating E2-15 or E2-16, the average power consumption of A&F is 0.731W (r=2) and 0.628W (r=3), which is 22.5% and 31.3% lower than that of the τ_2 prediction policy.

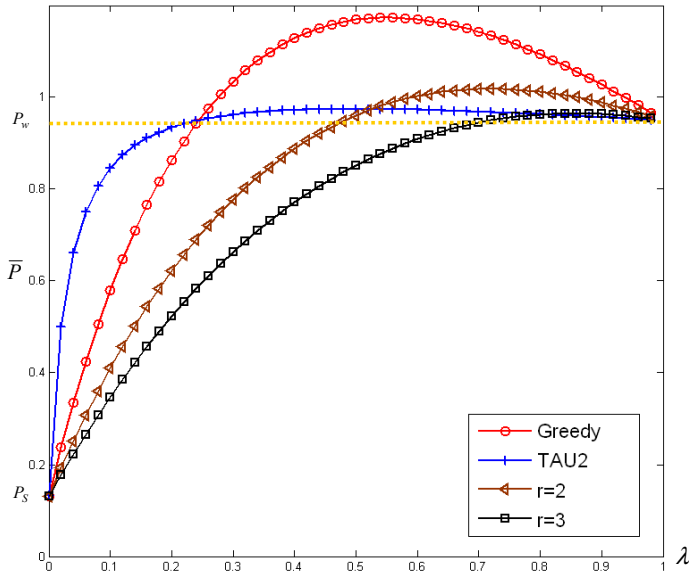

Figure 5: $\overline{P}$ in different policies

Another advantage of A&F policy is its easy implementation in hardware design. When drawing the prediction curve of Figure 5, we do not take the energy cost of prediction computation into consideration. This cost is related to the prediction algorithm and will increase with prediction accuracy. On the other hand, A&F is as simple as the greedy policy to implement and the energy cost of A&F computation (if it cannot be ignored) will not increase with the increase of the accumulation limit r.

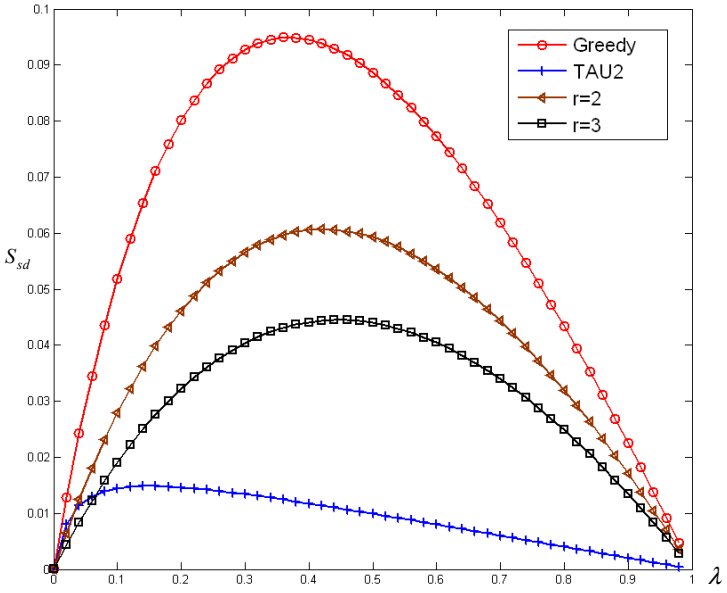

Figure 6: The influence of shutdown process

The inclusion of shutdown and wakeup processes into Markov models provides us with insight about the influence of these transitions in the performance of different policies. Figure 6 shows the variation of probability of shutdown states (S_{sd}) in different policies. From the figure, we can see S_{sd} reaches its top value when λ is near 0.5μ because here, the processor switches frequently from inactive states to active states and back. And it is clear

that the approximate equation given in [5] may cause gross inaccuracy in energy estimation because the top value of S_{sd} may be as high as 10%. Furthermore, the shutdown process has less influence on prediction policy than on greedy policy, and it is the main reason for the former policy gaining more efficiency than the latter. For the A&F policy, the larger the accumulation limit is, the less its curve is influenced by S_{sd}. The variation of probability of wakeup groups (S_{wu}) in different groups shows similar trends.

3. Latency Analysis

Although A&F policy may achieve very low power consumption especially when the accumulation limit is high, this gain is achieved at the cost of longer latency. System engineers need to balance both the gain in power and the cost in latency brought by DPM when they choose the proper low power policy for processors in their systems. For the A&F policy, it is also important to determine a proper value of accumulation limit r for different processors. In order to answer these questions, we must first find ways to measure the latency. In system design, soft deadlines for task execution have been a popular measure for real-world latency performance. For instance, such deadlines have been widely used in DPM or DVS (Dynamic Voltage Scaling) design [15]. Different from hard real-time systems [14], soft deadlines are not compulsory. They serve more as guidelines for execution scheduling to optimize system performance. Violation of such a deadline is not considered a catastrophe. The probability of deadline violation for each new coming task will be used here to mark the latency of different policies. This measure is more realistically expressive than the length of queue of waiting tasks.

According to Figure 4, the processor may work in any of its states when a new task comes. If the processor is in active state n when the new task comes, it has n+1 tasks so far waiting to be executed and scheduling is needed to select the proper task for the processor from time to time. Here we simply assume that the processor use FIFO (First-Input-First-Output) mechanism to choose tasks in the waiting queue and all time cost in the scheduling procedure can be ignored, the current task will be executed only after the completion of the previous n tasks (other assumptions of scheduling including priority etc. could be readily incorporated into our model). If we assume the execution of every task follows Poisson distribution, the execution of n+1 tasks follows Erlang distribution [10] with parameters n+1 and μ. If t is the entire time cost that the new coming task spent in the processor and $E_r(t;\ n+1,\ \mu)$ is the Erlang probability distribution function (pdf) and Q_n is the probability for the processor staying in state n, the probability of deadline violation in this case can be expressed as E3-1:

$$Q_n(t \geq DL) = Q_n * \int_{DL}^{\infty} E_r(t; n+1, \mu)dt \qquad \text{(E3-1)}$$

Similarly, if the processor is in wakeup state wu(n) when the new task arrives, the task must wait for the completion of the wakeup process and then the execution of the n tasks accumulated before its loading into the processor. If m is the fire moment of the processor and k is the time spent in the wakeup processing, $P_{oisson}(k;\delta)$ represents the corresponding Poisson pdf and the probability of deadline violation in this case can be expressed as E3-2:

$$\begin{aligned} Q_{wu(n)}(t \geq DL) &= 1 - Q_{wu(n)}(t < DL) \\ &= Q_{wu(n)} * (1 - \int_0^{DL} E_r(t-m; n+1, \mu) \int_0^m P_{oisson}(k;\delta)dkdt) \end{aligned} \qquad \text{(E3-2)}$$

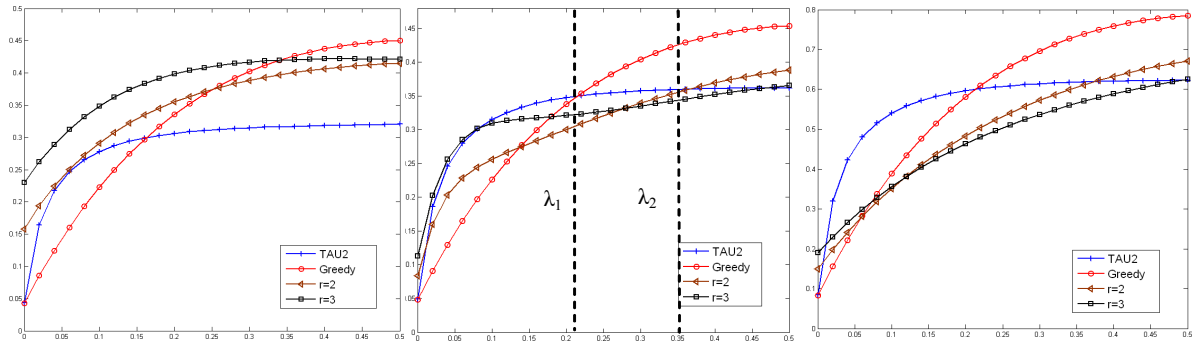

Figure 7: Optimized policy in different conditions. (a) Wp=0.3, DL=10/μ (b) Wp=0.3 DL=30/μ (c) Wp=0.6 DL=10/μ

The equations for inactive states and shutdown states are similar to E3-1 or E3-2. And the entire probability of deadline violation of the processor is given in E3-3:

$$Q(t \geq DL) = \sum_{n=0}^{r-1} Q_{n^*}(t \geq DL) + \sum_{n=0}^{\infty} Q_n(t \geq DL) + \sum_{n=0}^{\infty} Q_{sd(n)}(t \geq DL) + \sum_{n=r}^{\infty} Q_{wu(n)}(t \geq DL) \quad \text{(E3-3)}$$

With the latency expression of E3-3, we can now derive the combined latency-power index (E3-4).

$$Balance(\lambda, r, DL) = W_p * \frac{\overline{P}}{P_w} + (1 - W_p) * Q(t \geq DL) \quad \text{(E3-4)}$$

DL is the average deadline. W_p (0< W_p<1) in E3-4 is the weight of power, which depends on how much relative importance has been given to power saving and can be adjusted by system engineers according to their particular requirements. In order to unify the dimension of the equation, we normalized $\overline{P}$ by P_w which measures how much energy will be saved by the implemented DPM policy. Given the value of W_p, a DPM policy which can minimize E3-4 is the optimal policy which can balance system performance of both latency and power. For a particular processor, we assume constant parameters μ, γ and δ. Therefore the value of equation E3-4 depends on the accumulation limit r, the arrival rate λ and the deadline *DL*. If both λ and *DL* are assumed to be constant as well, the determination of accumulation limit r in A&F policy depends on the minimization of E3-4.

Furthermore, because all DPM policies can behave better than others under certain circumstances, much recent research work focuses on hierarchical architecture of DPM design which can adapt the proper DPM policy to dynamic systems [5]. And a unified cost function like E3-4 can serve as a standard assessment framework of policies and hierarchical DPM can adjust different policies according to the variation of environment parameters such as λ and DL.

Numerical solutions have been obtained using the parameters of MHF2043AT. We have limited our range to 0<λ<0.5 because as stated in Section 2.2, DPM does not make practical sense for larger λ.

Figure 7 shows a series of results. First of all, we set *DL* as ten times the average execution period (*DL*=10/μ) and set W_p to 0.3 which means latency rather than power is the main concern of the system and obtained Figure 7(a). Greedy and prediction policies are better here because they cause much less deadline violation than their A&F counterparts. If the processor has more powerful execution capability so that the deadline request becomes relatively looser, we obtain Figure 7(b) with *DL*=30/μ. Even with W_p keeping to 0.3, A&F policy can be a good choice. For example, A&F (r=2) is the best choice when λ= λ_1 and A&F (r=3) is the best choice when λ= λ_2. If

we increase W_p to represent those systems where power saving is more important, A&F policy will be chosen frequently even when the deadline requirement is tight. In Figure 7(c)), we raise W_p to 0.6 while DL is 10/μ. It can be seen A&F policy with large accumulation limit (r=3) is the best choice for a wide range of λ. This trend will become more obvious with the increase of W_p or DL.

4. Discussion and Future Work

A series of Markov models of DPM policies is presented in this paper. This modeling approach includes the explicit and complete representation of PM transition states, which enhances the accuracy and reliability of the models. It also produces models of regular shape and pattern with standard representation for any new operating states and the transitional states between them. This makes analytical closed-from solutions easier to derive and computational complexity of numerical solutions easier to reduce. It also makes these models readily extendible to cover the potentially diverse and increasingly sophisticated power management policies of the future. To demonstrate the capabilities of this method, the A&F policy is studied here in accurate detail for the first time with both power and latency analyses based on real-world processor parameters. We plan to further apply this method to study more sophisticated DPM as well as DVS policies in the context of processors (e.g. [16]) with more than two operating states.

It must be said that the philosophy of tau prediction policy is based on the assumption that some knowledge exists of system behaviour so that a "good" tau or tau function can be found at system design time. It is therefore not entirely fair to compare the performance of this scheme with others under a purely stochastic assumption. A detailed study of prediction policies is outside the scope of this report and we plan to investigate this topic in the future.

Acknowledgement

The authors would like to thank Prof. Isi Mitrani of Newcastle University for enlightening and extensive discussions.

Reference

[1] L Benini, G de Micheli, “Dynamic Power Management: Design Techniques and CAD Tools” , Kluwer Academic Publishers Norwell, USA, 1998

[2] S. Moore, G.Taylor, R.Mullins, P.Robinson, “Point to Point GALS Interconnect,” *ASYNC’02*, UK, 2002

[3] J.A.Gubner, "Probability and Random Processes for Electrical and Compter Engineers", Cambridge University Press, 2006

[4] Q. Qiu, M. Pedram, “Dynamic Power Management Based on Continuou-Time Markov Decision Processes”, DAC 1999

[5] Z. Ren, B.H. Krogh, R. Marculescu, “Hierarchical Adaptive Dynamic Power Management”, IEEE Transactions on Computers, 2005

[6] M. Srivastava, A, Chandrakasan, R. Brodersen, “Predictive system shutdown and other architectural techniques for energy efficient programmable computation”, IEEE Transactions on VLSI Systems, 1996

[7] C.-H. hwang and A. Wu, “A predictive system shutdown method for energy saving of event-driven computation”, Proc. of the ICCAD, 1997

[8] T.Simunic, L.Benini, A.Acquaviva, P.Glynn. G.de Micheli, “Dynamic Voltage Scaling and Power Management for Portable Systems”, DAC 2001

[9] C.-F. Chiasserini, R. Ramesh, “Energy Efficient Battery Management”, IEEE Journal on Selected Areas in Communications, Jul 2001

[10] L. Kleinrock “Queuing Systems Volume I: Theory”, John Wiley & Sons, 1975

[11] Yung-Hsiang Lu, Eui-Young Chung, T. Simunic, L. Benini, G. De Micheli “Quantitative Comparison of Power Management Algorithms”, DATE 2000

[12] L.Benini, A.Bogliolo, G. De Micheli “A survey of Design Techniques for System-level Dynamic Power Management” IEEE Transitions on VLSI June 2000

[13] G.Buzsaki, A.Draguhn, “Neuronal Oscillations in Cortical Networks”, Science, June 2004

[14] G.C. Buttazzo, “Hard Real-time Computing Systems: Predictable Scheduling Algorithms and Applications”, Springer, 2004

[15] L. Yuan, G. Qu. “Analysis of energy reduction on dynamic voltage scaling-enabled systems”, IEEE Transactions on CAD, Vol.24, No.12, 2005

[16] SA-1100 Microprocessor Technical Reference Manual, Intel, 1998

[17] R.Golding, P.K.Bosch, J.Wilkes. “Idlesness is not Sloth”, USENIX Winter Conference, 1995

[18] A. Kansal, M. B. Srivastava, “An environmental energy harvesting framework for sensor networks”, ISLPED’03, Seoul, Korea