Variation Aware Dynamic Power Management for Chip Multiprocessor Architectures

Mohammad Ghasemazar and Massoud Pedram
University of Southern California
Department of Electrical Engineering
Los Angeles, CA 90089, U.S.A.
{ghasemaz, pedram}@usc.edu

***Abstract*— with the increasing levels of variability in the characteristics of VLSI circuits and continued uncertainty in the operating conditions of processors, achieving predictable power efficiency and high performance in the electronic systems has become a daunting, yet vital, task. This paper tackles the problem of system-level dynamic power management (DPM) in the state-of-the-art chip multiprocessor (CMP) architectures that are manufactured in nanoscale CMOS technologies with large process variations or are operated under widely varying environmental conditions over their lifetime. We adopt a Markovian Decision Process based approach to CMP power management problem. The proposed technique models the underlying variability and uncertainty of parameters in system level as a partially observable MDP, and finds the optimal policy that stochastically minimizes energy per request. Experimental results demonstrate the high efficacy of the proposed power management framework.**

Keywords - Chip multiprocessor; Dynamic power management; partially observable Markovian decision process;

1. Introduction

With the increasing levels of variability in the characteristics of nanoscale CMOS devices and VLSI interconnects and continued uncertainty in the operating conditions of VLSI circuits, achieving power efficiency and high performance in electronic systems under process, voltage, and temperature (PVT) variations as well as current stress, device aging, and interconnect wear-out phenomena has become a daunting, yet vital, task. This paper tackles the problem of system-level dynamic power management (DPM) in chip multiprocessors (CMP) (a.k.a multicore systems) which are manufactured in nanoscale CMOS technologies and are operated under widely varying conditions over the lifetime of system. Such systems are greatly affected by increasing levels of process variations typically materializing as intrinsic (random) or systematic sources of variability and aging effects in device and interconnect characteristics, and widely varying workloads usually appearing as a source of uncertainty. Variations have a randomizing effect on the performance and power dissipation of a particular processor chip. At the same time, measurements made about the state of the processor and predictions about its future state tend to be imperfect, which gives rise to uncertainty about the system state. At the system level this variability and uncertainty is beginning to undermine the effectiveness of traditional DPM approaches.

This research was sponsored in part by a grant from the National Science Foundation under award number SHF-1018980.

A lot of research has been devoted to optimizing DPM policies, resulting in both heuristics and stochastic approaches. While the heuristic approaches are easy to implement, they do not provide any power/performance assurances. In contrast, the stochastic approaches guarantee optimality under performance constraints although they are more complex to implement. An approach based on discrete-time Markovian decision processes (DT-MDP) was proposed in [1]. This approach outperforms the previous heuristic techniques because of its solid theoretical framework for system modeling and policy optimization. A power management approach based on continuous-time MDP (CTMDP) was introduced in [2]. The policy change framework of this model is asynchronous and thus more suitable for implementation as part of a real-time operating system environment. Reference [3] improved on the modeling technique of [1] by using time-indexed semi-Markovian decision processes (SMDP). A non-stationary process based power management technique is introduced in [4], where the workload is modeled as Markov-modulated stochastic process. Reference [5] presented a modeling and optimization stochastic framework using partially observable MDP. It conveniently formulated and solved it as a quadratic constrained linear program. Authors of [6] presented a POMDP based framework to improve accuracy of decisions in power/thermal management.

Our proposed power manager interacts with an uncertain environment and statistically changing state variables and immediate cost function and tries to minimize the discounted cost in the limit by choosing appropriate actions. The decision making in a partially observable environment is achieved by combining aspects of Hidden Markov Models (HMMs) and MDPs. More specifically, we adopt a Partially Observable MDP to consider the uncertainty and variability in parameter observation.

The remainder of this paper is organized as follows. In section 2, the preliminaries of the paper are presented. The details of the stochastic uncertainty management framework are given in section 3. Section 4 presents variation-aware dynamic power management techniques. Experimental results and conclusion are provided in section 5 and 6, respectively.

2. Preleminaries

A. Variability and uncertainty

Randomness is defined as a lack of pattern or regularity. Two sources of randomness are generally recognized:

978-3-9810801-7-9/DATE11/©2011 EDAA

variability and uncertainty [7]. The first one, variability, is an inherent irregularity in the phenomenon being observed. It means that different situations produce different numerical values for a quantity. Variability in the delay and power consumption of VLSI circuits arises from scaling circuit technologies beyond the ability to control specific performance-dependent and power-dependent parameters [8]. In multicore systems spatially correlated intra-die process variations manifest themselves as core-to-core variations. The existence of variability implies that a single action or strategy may not emerge as optimal for every individual case.

The other source of randomness, *uncertainty*, is related to a generalized lack of knowledge about the processes involved. When making observations of past events or speculating about the future, imperfect knowledge is the rule rather than the exception. For example, the workload attributed to a particular functional unit is not directly recorded very often. Furthermore direct measurements may be noisy or erroneous. Uncertainty implies that we might make a non-optimal choice since we may expect one outcome different from what actually occurs.

Variations, especially those due to manufacturing process and aging phenomena, have a randomizing effect on the performance and power dissipation of a particular processor chip. At the same time, measurements made about the state of the processor and predictions about its future state tend to be imperfect, which gives rise to uncertainty about system state. We propose a power manager that interacts with statistically changing parameters and an uncertain environment.

B. System model

This paper targets current multiprocessor architectures, in single chip (CMP a.k.a. MPSoC) realizations supporting Dynamic Voltage and Frequency Scaling (DVFS). The availability of DVFS knob varies by implementation, from *independent* voltage and frequency scaling for each core to *universal* (a.k.a. global) voltage and frequency scaling for all cores in a CMP system. Without loss of generality, we consider a state-of-the-art CMP system (AMD's K10 family [9] or Intel's Nahalem or Sandy Bridge [10]) in which a single voltage source supplies power to all the cores (i.e., although the voltage setting can vary over time, the same value is applied to all cores at any time instance); However, the frequency setting of each core can be independently varied (subject to not exceeding the maximum frequency threshold allowed by the voltage setting). In addition, each individual core can be turned on or off independently.

Consider a CMP system with N processing cores with k global voltage levels, and l frequency levels for each core when the core is *active* (note that we assume a one-to-one relationship between the operating voltage and maximum clock frequency of the core, however the core can operate at a frequency lower than the maximum one.) Each core also has a sleep state. Commands issued by a power manager are various f settings (including sleep) that cause transitions among different system states. The global voltage level is determined by the highest voltage applied to any individual core.

A CMP's performance may be defined by metrics such as the number of requests serviced, denoted by H (can be measure in a time epoch), instructions-per-second (IPS), and even more elaborate measures based on accurate clocks-per-instruction (CPI) stacks that are useful for data placement and application software optimizations [11]. In our problem formulation, it is more appropriate to use a high-level metric; hence, we adopt the number of requests serviced per second as the system's performance metric.

We assume a set of application programs run on the multicore system, each giving rise to a specific, but known, *request arrival rate,* λ. This is modeled by introducing multiple rate modes for applications [12]. In addition the requests have different levels of instruction-level parallelism and access to memory resulting in different IPS values. We assume a distributed task assignment mechanism where incoming requests to the CMP enter a *ready task queue* and are held there until they are picked by the next available core. That is, every time a core becomes available, it fetches the request in front of the ready queue and starts executing it (or adds the request to its own local queue; however it would not change the proposed mathematical model).

3. Markov Decision Process-based DPM

A (time-average) Markov decision process (MDP) model facilitates reasoning in domains where actions change the system states and a cost (or reward) is utilized to optimize the system performance. The simple MDP is directly observable in the sense that its execution hinges on the assumption that the current system state can be determined without any errors and the cost (reward) of an action can be calculated exactly. In a multicore system, the *true system state* is determined by a three-tuple, <*workload, power, performance*>. It is difficult to determine the true state of a system directly, because none of the true state variables are directly measurable at run time and furthermore, they are all affected by one or more sources of randomness. Therefore, we define an *observable system state* which consists of observable variables and can be used to infer the true state of the system. The observable system state, which is the joint global state of the request generators, the ready queue, the cores, is given by the tuple: $< \lambda, w, q, v, \overline{f}, \overline{U} >$. Table 1 provides their definition.

Table 1 - Definition of system parameters

λ is the arrival rate of requests
w is the request size
q is the occupancy level of the ready queue
v is the chip-wide operating voltage of the cores
$\overline{f}$ is the operating frequency vector of the cores
$\overline{U}$ is the utilization vector of the cores

In partially observable environments, where states of the system cannot be identified exactly, observations made by a power manager about the state of the system are indirect and may even be noisy, and therefore, they only provide incomplete information. One way to deal with uncertainty under a wide range of operating conditions and environments is to rely on the history of previous actions and observations to disambiguate the current state. For example, a hidden Markov model (HMM) [5][13] can be adopted, where the state is not directly observable but variables influenced by the state are observable, to learn a model of the hidden states. Thus, a power manager in HMM reasons about the system state indirectly through these observed variables, which capture

complex system dynamics that are not completely observable. The decision making in a partially observable environment is achieved by combining aspects of HMMs and MDPs. Specifically, we utilize a MDP, which models the decision making strategy, combined with a HMM, which considers the uncertainty in parameter observation.

Definition 1: A Partially Observable Markov Decision Process, POMDP, is a tuple (*S, A, O, T, C, Z*) such that:

S is a finite set of states.
A is a finite set of actions.
O is a finite set of observations.
T is a transition probability function. T: $S \times A \rightarrow (S)$
C is a cost function. C: $S \times A \rightarrow \Re$
Z is an observation function. Z: $S \times A \rightarrow \Delta(Z)$

The state space *S* comprises of a finite set of states, where $s \in S$ can be defined as global (observable) state of the system. In our problem context, a system state is characterized by the multi-tuple: $< \lambda, w, q, v, \bar{f}, \bar{U} >$.

The action space *A* consists of a finite set of actions $a \in A$, that are issued by the power manager to change the processor state, and hence cause transitions into and out of the system states. These actions include commands to turn individual cores on/off and set the chip-wide voltage setting for all active cores, as well as setting their frequency level. They affect *v* and $\bar{f}$ directly, and $\bar{U}$ indirectly.

The observation space *O* contains a finite set of observations $o \in O$, namely, value of λ for current requests reported by the operating system, occupancy level of the ready queue, *q*, chip-wide operating voltage of cores, *v*, operating frequency vector of cores, $\bar{f}$, utilization vector of cores, $\bar{U}$, and the request size, *w,* reported by built-in sensors on the chip.

The state transition probability function, $T(s^{t+1}, a^t, s^t)$ [1], determines the probability of a transition from a state s^t to another state s^{t+1} after executing action a^t, i.e., system transits to state s^{t+1} at time t+1 with the probability given by (1).

$$Pr(s^{t+1} | s^t, a^t) = T(s^{t+1}, a^t, s^t) \quad (1)$$

An observation function, $Z(o^{t+1}, s^{t+1}, a^t)$, which captures the relationship between the actual state and the observation, is defined as probability of making observation o^{t+1} after taking action a^t that has landed the system in state s^{t+1}, i.e., state s^{t+1} generates observation o^{t+1} at time t+1 with probability

$$Pr(o^{t+1} | s^{t+1}, a^t) = Z(o^{t+1}, s^{t+1}, a^t) \quad (2)$$

We consider a bounded cost function that assigns a statistical cost value to each state-action pair whereby an *immediate cost*, $C(s^t, a^t)$, is incurred when action a^t is executed in state s^t. In this work, the immediate costs are calculated as average *energy per request* in state s^t during the epoch leading to time *t* plus the *energy* cost of the transition from state s^t to s^{t+1} (this includes for example the energy consumed for adjusting the voltage of the processor chip),

$$C(s^t, a^t) = \frac{1}{H(s^t)} [Energy(s^t) + \sum_{s^{t+1} \in S} \left(Energy(s^t \rightarrow s^{t+1}) \cdot T(s^{t+1}, a^t, s^t) \right)] \quad (3)$$

[1] In this paper, we denote the time stamp of states, actions, and observations with a superscript on the variable, for instance s^t denotes state *s* at time *t*.

This way of defining the immediate cost of state-action pairs is motivated by the fact we can account for the cost of a state only after it is accurately (not probabilistically) known that the system has landed there. Hence, the cost would be calculated at the next decision epoch, to account for the energy consumption of processor in the state s^t during one decision epoch, plus transition cost of an action.

A. Belief state

Instead of making decisions based on the current perceived state of the system, the POMDP maintains a *belief*, i.e., a probability distribution over the possible states of the system, and makes decisions based on its current belief. The optimal POMDP solution is Markovian over the belief space, *B* [6]. Hence, by using the belief space, we convert original POMDP into a completely observable, regular (albeit continuous state space) MDP, the so-called belief state MDP. The *belief state* at time *t* is a $|S| \times 1$ vector of probabilities defined as [13]:

$$b^t := [b^t(s)] \quad \forall s \in S \ , \sum_{s \in S} b^t(s) = 1 \quad (4)$$

where $b^t(s)$ is the posterior probability distribution of state *s* at time *t*. Note that the $|S|$-dimensional belief state is continuous. Based on the belief state, an action a^t is chosen by the policy $\pi = \{\pi^t\}$, that maps the belief states to actions. Given belief state b^t and an action a^t resulting in observation o^{t+1}, using Bayes' rule, we can compute the successor belief state b^{t+1} as follows:

$$b^{t+1}(s) = Pr(s | o^{t+1}, a^t, b^t) = \frac{Z(o^{t+1}, s, a^t) \cdot \sum_{s'} \left(b^t(s') T(s, a^t, s') \right)}{\sum_{s''} \left(Z(o^{t+1}, s'', a^t) \cdot \sum_{s'} \left(b^t(s') T(s'', a^t, s) \right) \right)} \quad (5)$$

The belief state transition function, $T_b(b^{t+1}, a^t, b^t)$, which provides the probability of a transition from current belief state b^t to next belief state b^{t+1} after executing action a^t, is given by:

$$T_b(b^{t+1}, a^t, b^t) = Pr(b^{t+1} | a^t, b^t) = \sum_o \left(Pr(b^{t+1} | a^t, b^t, o) \cdot Pr(o | a^t, b^t) \right) \quad (6)$$

The cost model presented in (7), denotes the immediate cost incurred by action a^t issued in current state b^t.

$$C_b(b^t, a^t) = \sum_{s \in S} \left(b^t(s) \cdot C(s, a^t) \right) \quad (7)$$

We have thus transformed the problem based on the POMDP model to one based on belief-state MDP model. The optimal policy $\pi^*(b)$ of the belief-state MDP representation is also optimal for the physical-state POMDP representation.

4. Proposed Variation Aware DPM

As stated earlier, the proposed power manager interacts with an uncertain environment and statistically variable state variables and immediate cost function and tries to minimize the discounted cost in the limit by choosing appropriate actions at regularly scheduled decision epochs. Figure 1 illustrates the basic structure of our proposed Variation Aware Dynamic Power Manager. The actions issued by the power manager change the system state and lead to quantifiable cost.

The power manager consists of two functional components. The first component is the belief state estimator block which predicts the system's next belief state, b^{t+1}, based on the current belief state, b^t, action taken a^t, and observation, o^t. The second component is policy maker that calculates statistical total cost function and assigns optimal actions, a^{t+1}, based on a policy optimization algorithm. In online VA-DPM, the belief

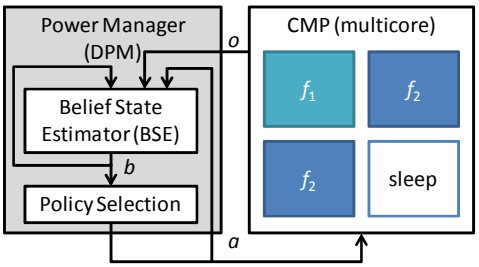

Figure 1. Structure of variability-aware DPM

state estimator utilizes a Kalman Filter [14], and the policy optimization block makes decisions to minimize a cost function (c.f. section A) by using value-iteration method [15].

A. *Cost-fucntion of VA-DPM*

Equation (3) defines the cost function associated with state-action pairs. In this equation, the energy consumption of CMP during an epoch, D_{epoch}, is the product of its power consumption and the duration of decision epoch. On the other hand, since the average transition energy between the states is negligible compared to the energy dissipation due to execution of request, we drop the corresponding term.

Moreover, to account for the performance degradation, we incorporate a penalty in the cost function which depends on occupancy level of the system's ready queue, q, in state s. This penalty is a weighted linear function of the difference of q and a target occupancy level, q_0 (which is set statically by the designer or adaptively by the DPM itself– a typical value of q_0 is 30% to 75% of the max queue capacity – values outside the range tend to result in large performance penalty or even request loss going above the range or resource underutilization and energy inefficiency due to "non energy proportional" nature of today's servers going below the range [16].) Thus, $C(s,a^t)$ may be rewritten as:

$$C(s,a^t) = E\left[\frac{P(s)\cdot D_{epoch}}{H(s)}\right] + \alpha|q(s) - q_0| \qquad (8)$$

where $P(s)$ represents the power consumption at state s, D_{epoch} indicates the duration of decision epoch, H denotes number of completed (serviced) requests during the last epoch. Note E[.] represents the expected value (average value).

To estimate the power consumption of CMP in each state, we use the model presented in [18]. It provides accurate power and performance models for a high performance multi-core server system, based on measurements. It characterizes total power consumption of CMP as a function of core utilization,

$$P(s) = c_1(s)\cdot\sum_{i=1}^{N} u_i + c_0(s) \qquad (9)$$

where c_1(s) and c_0(s) are regression coefficients for the power macro-model at state s. These coefficients can be extracted by characterization of any particular CMP; Authors of [18] report their values for an Intel Xeon processor 5400 series.

The average number of serviced requests in any state, s, can be computed by dividing summation of useful clock cycles of all cores by the average request size in that state, $w(s)$,

$$H(s) = \frac{D_{epoch}\cdot\sum_{i=1}^{N}(f_i(s)\cdot u_i(s))}{w(s)} \qquad (10)$$

where f_i and u_i denote the clock frequency and utilization of i-th core, respectively. Note that $w(s)$, which is an input to our problem, captures the size and CPI of the arriving requests in state s. This can be done by dynamic profiling with the aid of built-in performance monitoring units in the processor [18].

B. *Policy optimization by value-iteration*

Finding an optimal power management policy requires a decision-making strategy which maps belief states to actions. The goal is to minimize some cumulative function of the costs, typically infinite-horizon sum under a discounting factor γ (usually just under 1), which may be computed by (11).

$$\Phi^*(\pi) = \min_{\pi}\left(\sum_{t=0}^{\infty}\sum_{a^t,b^t}\gamma^t T_b(b^{t+1},a^t,b^t)C_b(b^t,a^t)\right) \qquad (11)$$

The standard family of algorithms to calculate the policy requires storage for two arrays indexed by state: *cost* $\boldsymbol{\Phi}$, which contains real values, and *policy* π which contains actions. At the end of the algorithm, π will contain the solution and $\boldsymbol{\Phi}(b_0)$ will contain the discounted sum of the costs to be accrued (on average) by following that solution. The algorithm then has the following two steps, which are repeated in some order for all the states until no further changes take place.

$$\pi(b) = \underset{a}{\operatorname{argmin}}\sum_{b'\epsilon B}\left(T_b(b',a,b)\cdot\Phi(b')\right) \qquad (12)$$

$$\Phi(b) = C_b(b,a) + \gamma\sum_{b'\epsilon B}\left(T_b(b',\pi(b),b)\cdot\Phi(b')\right) \qquad (13)$$

In value iteration [15], the π array is not used; instead, the value of $\pi(b)$ is calculated whenever it is needed. Substituting the calculation of $\pi(b)$ and (7) into the calculation of $\boldsymbol{\Phi}(b)$ gives the combined step:

$$\Phi^*(b) = \min_{a}\left(\begin{matrix}\sum_{s\epsilon S}\left(b(s)\cdot C(s,a)\right)\\ +\gamma\sum_{b'\epsilon B}\left(T_b(b',a,b)\cdot\Phi^*(b')\right)\end{matrix}\right) \qquad (14)$$

Given the optimal cost function, the optimal policy is:

$$\pi^*(b) = \underset{a}{\operatorname{argmin}}\left(\begin{matrix}\sum_{s\epsilon S}\left(b(s)\cdot C(s,a)\right)\\ +\gamma\sum_{b'\epsilon B}\left(T_b(b',a,b)\cdot\Phi^*(b')\right)\end{matrix}\right) \qquad (15)$$

Let b_0 denote the initial state of system, then the goal of VA-DPM is to minimize$\Phi(b_0)$, as formulated below using value-iteration method,

$$\begin{cases} Minimize\ \Phi(b_0) \\ \text{subject to:} \\ \Phi(b) = \min_a\left(C_b(b,a) + \gamma\sum_{b'\epsilon B}\left(T_b\left(b',\pi(b),b\right)\Phi\left(b'\right)\right)\right) \\ a = \pi(b) = \underset{a}{\operatorname{argmin}}\sum_{b'\epsilon B}\left(T_b(b',a,b)\cdot\Phi(b')\right) \\ C_b(b,a^t) = \sum_s\left(b(s)\left(E\left[\frac{P(s)\cdot D_{epoch}}{H(s)}\right] + \alpha|q(s)-q_0|\right)\right) \end{cases} \qquad (16)$$

Note that C_b and T_b are known in this problem statement; immediate cost of transitions between states can be calculated once states are defined; Finally, T_b is determined by some estimation methods to be discussed later.

C. *Belief state estimator (BSE)*

We present an online prediction-based DPM technique, which is analytically and statistically tractable.

First, assuming we know the distribution of PVT variations and observation noise, we can define the state and observation models in accordance with our framework as follows:

$$b^{t+1} = \mathbf{X}b^t + \mathbf{Y}a^t + u^t, \qquad u^t \sim N(0, Q^t) \qquad (17)$$

$$o^{t+1} = \mathbf{Z}b^{t+1} + v^{t+1}, \qquad v^{t+1} \sim N(0, R^t) \qquad (18)$$

where u^t is a state noise induced by PVT variation which is normally distributed with zero mean and variance Q^t; v^{t+1} is an observation noise normally distributed with zero mean and variance R^t. The state transition matrix **X** includes the probabilities of transitioning from state b^t to another state b^{t+1} when action a^t is taken. The action-input matrix **Y** relates the action input to the state, whereas the observation matrix **Z** contains the probabilities of making observation o^{t+1} when action a^t is taken, leading the system to enter state s^{t+1}. In practice, **X**, **Y**, and **Z** might change with each time step or measurement, but here we assume they are constant.

The estimation algorithm performs the state estimation based on Kalman Filter (KF). It produces estimates of the true values of measurements and their associated calculated values by predicting a value, estimating uncertainty of predicted value, and computing a weighted average of predicted and measured value. We skip its details due to space limitation (cf. [14]).

Figure 2 illustrates one decision (timestamp t) of the online VA-DPM algorithm to compute the optimal policy given by (14). It estimates the next belief state based on the Kalman filter technique, and computes the belief-state transition probabilities by simply deriving the maximum likelihood estimates, and then finds the optimal policy by utilizing value-iteration algorithm, as explained in section 4.B.

1 make observation o^{t+1}
2 estimate the next state, $b^{t+1} = BSE(o^{t+1}, a^t, b^t)$
2.1 predict: $\tilde{b}^{t+1} = \mathbf{X}b^t + \mathbf{Y}a^t + u^t$
2.2 update: $b^{t+1} = \tilde{b}^{t+1} + \mathbf{K}^{t+1}(o^{t+1} - \mathbf{Z}\tilde{b}^{t+1})$
3 compute T_b by maximum likelihood method
4 find optimal a^{t+1} through value-iteration
5 return $a^{t+1}=\pi(b^{t+1})$

Figure 2. Online algorithm for variability-aware DPM

The immediate cost is provided in constant time e.g., in a table-lookup. With n states and m actions, this algorithm takes $O(n)$ and $O(m)$ times for finding a belief state and an optimal action, respectively, which results in total $O(nm)$ running time.

5. Experimental Results

We evaluated the effectiveness of the proposed VA-DPM in terms of energy saving and performance by a series of experiments. We implemented our optimization algorithms as well as a real-time simulator (to model the operation of a CMP and its power management) in MATLAB [20]. The simulator models a CMP (c.f. section 2.B), queuing and execution of requests, and performing commands issued by power manager. Requests are randomly generated and sent to CMP, with a random inter-arrival time (following an exponential distribution.) Size of requests are based on two different workloads in SPECWeb2005 benchmark suite [21]. In order to model uncertainty and variability of the parameters, we applied a Gaussian disturbance to of request characteristics at runtime. The processor in our experiments is a dual core CMP. It has two global voltage levels, and each individual core can operate at high (2.3GHz) or low frequency (2.0GHz) or sleep mode. The power-performance relationship presented in [18] is used to model the characteristic of our CMP, and without loss of generality, we use the power coefficients reported in this work as an example; its parameters may affect the values of our results, but not the nature of our solution. According to this work, mixing high and low frequency settings would not save any power; hence we define the set of actions to be A={L0, LL, H0, HH},where L,H,0 denote low, high frequency and sleep mode, respectively. Note that CMP's voltage is set automatically by hardware corresponding to the highest employed frequency. Continuous parameters such as arrival rate, request size, queue occupancy, and utilization are divided into intervals to become discrete; i.e., the range of utilization levels is divided into discrete space of {U_L=(0,45%], U_M=(45%,80%], U_H=(80%,100%]}, while space of queue occupancy is {q_L=[0,30%], q_M=(30%,65%] q_H=(65%,100%]}. Finally, λ and w are profiled per application into two modes only. We next construct the corresponding discrete state space, S, and observation space, O, by Cartesian product of < λ, w, q, v, $\bar{f}$, $\bar{U}$> in which v, $\bar{f}$ do not have any uncertainty. We set γ=0.9 here, however, it is a user defined constant, and can be set after profiling through different applications.

Figure 3 demonstrates the performance of our belief state estimator, which estimates state of total utilization (summation of probabilities of states with same utilization level) starting at an initial state U_L (low utilization). As illustrated, b is updated with time and final state tends to be medium utilization (U_M).

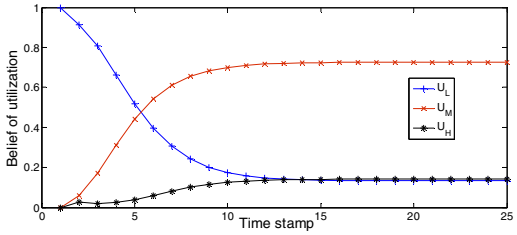

Figure 3. Performance of belief state estimator

Next, we define two baseline power management techniques to establish the comparison basis for overall evaluation of proposed VA-DPM. The first one, called BASE, is a conventional DPM that acts only based on observations without any prediction of underlying state. It uses an optimized offline policy to select the best action for to any observation. The second heuristic, called uncertainty tolerant (UT-DPM), is a POMDP similar to VA-DPM, but without the ability to catch variation related effects. It only captures the uncertainty in observations, by ignoring the state noise in (17).

Figure 4 demonstrates average power consumption of the three mentioned power management algorithms, BASE, UT-DPM, and VA-DPM. As it can be seen, BASE frequently switches between actions, because its decision is independent of PVT variations, hence non-optimal, and also it quickly responds to the uncertain input data, causing excessive power consumption, for any particular throughput. In contrast, VA-DPM takes advantage of estimated unobservable parameters and makes better decisions.

On the other hand, comparison of UT-DPM and VA-DPM in Figure 4 shows the importance of optimal action selection, given an estimate of system state. Note that in this figure, *Action ID* corresponds to the optimum frequency level that is chosen by algorithms that minimizes the cost function in (16). As mentioned, the only difference between these two algorithms is that UT-DPM accounts for observation noise but

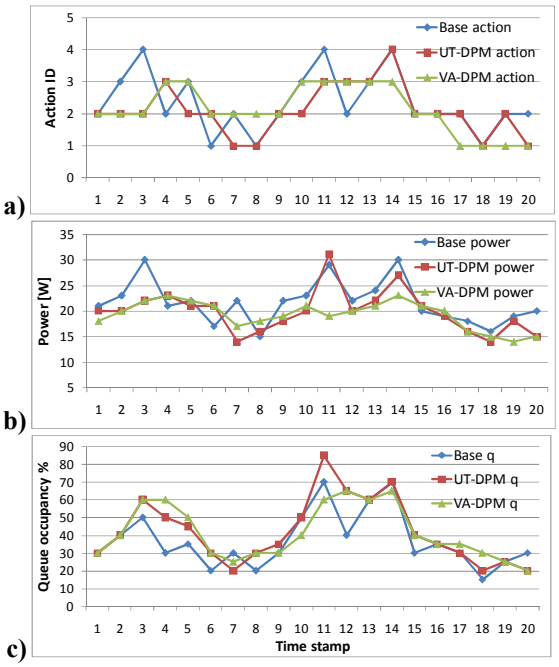

Figure 4. Comparison of BASE, UT-DPM, and VA-DPM a)Action ID b)Power consumption c)queue occupancy percentage

not for variations, while VA-DPM considers both. As it can be seen in Figure 4, the actions taken by VA-DPM result in lower power consumption, on average. Average power consumption and average queue occupancy of three algorithms are shown in Figure 5 (q_0 was set to 40%). VA-DPM reaches an average power saving of 13% with respect to BASE, by optimally handling variability and uncertainty. But the difference between the power consumption of UT-DPM and VA-DPM is negligible (VA-DPM consumes 3% less power than UT-DPM) because they operate the same where system parameters have not been affected by variations, but VA-DPM provides good solutions and guaranteed correctness under all corner cases. Moreover, the variance in power and q values is lowest for VA-DPM among all three methods.

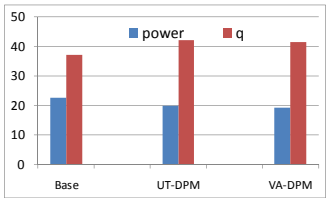

Figure 5. Comparison of BASE, UT-DPM, and VA-DPM

As the final experiment, in order to test the effect of performance penalty, we applied a heavier workload, which has 20% larger average request size, and used a 20% higher α (busy-queue penalty factor; cf. (8)). As a result of this change in performance requirements, overall power consumption of VA-DPM increased by about 11% but the percentage of queue being crowded (i.e., being in state q_H) dropped to about 5% (which is a 6% improvement) which in turn translates to lower request loss rate.

6. Conclusion and Future Work

We tackled the problem of system-level dynamic power management (DPM) in CMP architectures subject to process variations and widely varying environmental conditions. Our proposed solution is based on Partially Observable Markovian Decision Process, which finds the optimal policy that stochastically minimizes energy per request plus a performance degradation penalty. It uses well established prediction techniques to estimate system state and take the globally optimal action accordingly. In comparison to the baseline power management algorithms, our technique gains an average power saving of 13%.

This work can be extended in the estimation techniques used, in order to utilize a lighter and faster calculation algorithm for power manager. Also, a mechanism can be adopted to make better use of correlated observations to gather more reliable information, and decrease its uncertainty at the source (sensor). Another restriction of this work is its simulation infrastructure, which would be supported by a physical simulation setup in future for accurate measurements.

References

[1] L. Benini, G. Paleologo, A. Bogliolo, G. De Micheli, "Policy Optimization for Dynamic Power Management," *IEEE Trans. on Computer Aided Design*, Jun. 1999, pp. 813-833.

[2] Q. Qiu, Q. Wu, M. Pedram, "Stochastic Modeling of a Power-Managed System – Construction and Optimization," *IEEE Trans. on Computer-Aided Design*, Oct. 2001, pp. 1200-1217.

[3] T. Simunic, L. Benini, P. Glynn, G. De Micheli, "Event-driven Power Management," *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, Jul. 2001, pp. 840-857.

[4] Z. Ren, B. H. Krogh, R. Marculescu, "Hierarchical Adaptive Dynamic Power Management," *IEEE Trans. on Computers*, Apr. 2005.

[5] Y. Tan, Q. Qui, "A framework of stochastic power management using hidden Markov medel," *Proc. Design Automation Test in Europe, 2008*

[6] H-S. Jung, M. Pedram, "Uncertainty-aware dynamic power management in partially observable domains," *IEEE Trans. VLSI Systems*, Jun. 2009

[7] F. Meyer, M. Schmitt, *Space, Structure and Randomness: Contributions in Honor of Georges Matheron in the Fields of Geostatistics, Random Sets and Mathematical Morphology*, Springer, 2007.

[8] K. Bernstein, *et al.,* "High-performance CMOS variability in the 65-nm regime and beyond," *IBM Journal of Research and development*, 2006.

[9] AMD Corp, http://support.amd.com/us/Processor_TechDocs/40036.pdf

[10] Intel Co, http://www.intel.com/technology/architecture-silicon/next-gen/

[11] S. Eyerman, L. Eeckhout, T. Karkhanis, J. E.Smith, "A performance counter architecture for computing accurate CPI components," *SIGOPS Oper. Syst.* Rev. 40, 5, Oct. 2006, pp. 175-184.

[12] Q. Qiu, M. Pedram, "Dynamic power management based on continuous-time Markov decision processes," *Proc. Design Automation Conf.*, 1999.

[13] A.R. Cassandra, L.P. Kaelbling, M.L. Littman, "Acting Optimally in Partially Observable Stochastic Domains," *Proc. Conf. Artificial Intelligence*, Aug. 1996, pp. 1023-1028.

[14] G. Welch and G. Bishop, "An introduction to the Kalman filter," Technical Report TR 95-041, University of North Carolina, Department of Computer Science, 1995.

[15] J. Pineau, G. Gordon, S. Thru, "Point-based value iteration: An anytime algorithm for POMDPs," *Int'l Joint Conf. Artificial Intelligence*, 2003.

[16] L. A. Barroso, U. Hölzle, "The case for energy-proportional computing," *IEEE Computer*, vol. 40, 2007.

[17] M.L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Publisher, New York, 1994.

[18] I. Hwang, M. Pedram, "Power and performance modeling in a virtualized server system," in *Proc. of Int'l Conf. on Parallel Processing Workshops*, 2010.

[19] S. Eyerman, L. Eeckhout, T. Karkhanis, J. E.Smith, "A performance counter architecture for computing accurate CPI components," SIGOPS Oper. Syst. Rev. 40, 5, Oct. 2006, pp. 175-184.

[20] MATLAB Optimization, http://www.mathworks.com

[21] http://www.spec.org/