Stochastic Dynamic Thermal Management: A Markovian Decision-based Approach

Hwisung Jung, Massoud Pedram
Department of Electrical Engineering, University of Southern California
{hwijung, pedram}@usc.edu

Abstract - **This paper proposes a stochastic dynamic thermal management (DTM) technique in high-performance VLSI system with especial attention to the uncertainty in temperature observation. More specifically, we propose a stochastic thermal management framework to improve the accuracy of decision making in DTM, which performs dynamic voltage and frequency scaling to minimize total power dissipation and on-chip temperature. A key characteristic of the framework is that thermal states are controlled by stochastic processes, i.e., partially observable semi-Markov decision processes. Collaborative optimization is considered with mathematical programming formulations to reduce operating temperature by using multi-objective design optimization methods. Experimental results with 32bit embedded RISC processor demonstrate the effectiveness of the technique and show that the proposed algorithm ensures thermal safety under performance constraints.**

I. INTRODUCTION

"Smaller and faster" are the chief demands driving today's electronic designs because they generally mean higher performance. However, they also translate into high power densities, higher operating temperatures and lower circuit reliability. Furthermore, local hot spots, which have much higher temperatures compared to the average die temperature, are becoming more prevalent in VLSI circuits.

With component packages becoming more compact and having smaller physical profiles, it is no longer sufficient to merely add "a bigger fan" as a downstream fix for thermal problems. Because heat flow must be planned and thermal resistances minimized, thermal management is best accomplished when it is incorporated starting at the beginning of the design cycle. In addition, although worst-case heating conditions seldom arise in a circuit during its lifetime, when they do arise, they can cause significant problems, ranging from circuit transient timing errors to complete catastrophic burnout. A package designed for the worst case is excessive. Any applications that generate heat should engage an alternative, *runtime* thermal-management technique (*dynamic thermal management* or DTM). Since typical high-power applications still operate 20% or more below the worst case [1], this can lead to dramatic savings. This is the philosophy behind the thermal design of the Intel Pentium 4.

As evidenced in the recent literature [2][3][4][5], increasing interest has focused on dynamic thermal management. The work presented in [2] studies the architectural-level thermal model, HotSpot, and management based on an equivalent circuit models that corresponds to micro-architecture blocks. The trigger mechanism used to cool the microprocessor's temperature with DTM has been derived in [3] by using Wattch. Predictive DTM [4], which exploits certain properties of multimedia applications, is an example of online strategies for thermal management. In reference [5], authors have tackled the performance optimization problem for disk drives by studying the inter-relationship between capacity, performance, and thermal characteristics of disk drives. A good summary of research that combines thermal management techniques and potential risks is given in [6].

Although most of the previous works on DTM has concentrated on thermal modeling and simulation at circuit, gate, and architecture levels, less attention has been paid to thermal management at system level. Furthermore, these DTM techniques, which depend on the use of temperature sensors, can hardly observe the peak power dissipation and resulting peak temperature due to the non-uniform power density across a chip. Thus, important aspects contributing to the advancement of thermal-managed system design is the abilities to fully model and characterize the thermal behavior of the system. In particular, since temperature variation across a chip can result in significant uncertainty in temperature observation, improving the accuracy of decision making in thermal management by modeling and assessing the uncertainty is an important step to guarantee the quality of the consumer electronics.

It is believed that prompt regulation of on-chip temperature by DTM must be combined with dynamic voltage and frequency scaling (DVFS), which has proven to be an effective technique for reducing total power consumption [7], to ensure higher thermal safety of the chips. In literature, several works [8][9] have been proposed on combining DVFS and dynamic power management based on stochastic processes. However, to the best of our knowledge, no proposed research work is conducted on combining DTM and DVFS techniques in stochastic modeling with the uncertainty in temperature observation. This is specifically the contribution of the present paper.

In this paper, we propose a stochastic dynamic thermal management (SDTM) technique to control the temperature of the system and its power dissipation. we present a systematic approach for modeling a stochastic thermal management

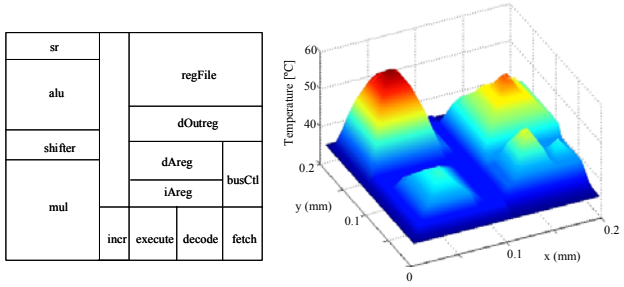
1-4244-9707-X/06/$20.00 ©2006 IEEE

framework, which is based on i) partially observable Markov decision process [10] to model the uncertainty in temperature observation and ii) semi-Markov decision process to model decision making for DVFS. Markov decision process model offers a robust theoretical framework which enables one to apply strong mathematical optimization techniques in order to derive optimal thermal management policies for thermal-managed system. Furthermore, multi-objective design optimization method, which involves the coordination of multiple disciplinary performance constraints, is used to realize more effective solution.

The remainder of this paper is organized as follows. Section 2 provides the brief background on temperature profile identification. The detail of stochastic thermal management framework is presented in section 3. Section 4 shows a dynamic thermal management algorithm. Experimental results and conclusion are given in section 5 and section 6.

II. BACKGROUND: TEMPERATURE PROFILE IDENTIFICATION

Temperature reading can be performed by either external or internal temperature sensors. External temperature sensors, so-called thermo couples, suffer a time-delay in the temperature reading due to the thermal constant from the integrated circuit to the external sensor. Thus, internal temperature sensors, e.g., analog CMOS sensors or ring oscillators, which can be deployed in large number across a chip, have been widely used in pursuit of great accuracy in measuring temperature with quick response time. However, this approach still has clear limitations. For example, ring oscillator-based thermal sensors have problems in large area size and low accuracy. Analog CMOS sensors, although very successful in accuracy and size, are in general sensitive to noise on power and ground lines. Furthermore, sensor output is affected by process variation on low temperature [11]. To compensate for these calibration problems, we can deploy multiple thermal sensors across a chip, but we add on the overall power consumption as well as area size. It is clear that there is a trade-off between efficiency and accuracy in thermal management due to the large number of heat sources in a chip. The non-uniform on-chip temperature is realized due to large temperature variations across a chip, which can be affected by the different heat diffusion length [2].



(a) (b)

Figure 1. Temperature profile identification: (a) floorplan. (b) temperature distribution.

A critical problem in temperature profile identification induced by temperature variation is as follows: suppose that CMOS thermal sensors, which have characteristics as explained above, are implemented along with the processors or functional blocks in a chip. Due to *a)* the limitation of sensor deployment in the chip, e.g., the number, size, and location of the sensors, and *b)* non-uniform temperature distribution across a chip, these sensors may be remote from the hot spots of interest, which incurs the uncertainty in temperature observation. Figure 1 shows the example of non-uniform temperature distribution obtained by running *gcc* (SPECint2000) [12] on a RISC processor designed by the authors. The values of thermal parameters were extracted from the commercial data sheet for a QFP package. The average temperature can be estimated using the following equation,

$$T_{chip} = T_a + \theta_{ja}\left(P_{s/w} + P_{s/c} + P_{static} + P_{int}\right) \quad (1)$$

where T_a is the ambient temperature (25°C), θ_{ja} is the thermal resistivity [°C/W] of the chosen package, and $P_{s/w}$ is the switching component of power, $P_{s/c}$ is the short-circuit power, P_{static} is the leakage power, and P_{int} is the internal charge/discharge power.

III. THERMAL MANAGEMENT FRAMEWORK

We present a theoretical framework to construct a thermal management process under the temperature uncertainty by combining the stochastic processes.

A. POSMDP Formulation

The uncertainty problem in temperature observation, where a thermal manager cannot reliably identify the thermal state of the chip during thermal management, can be solved by modeling decision making by stochastic processes. Note that a thermal manager, which observes the overall thermal state and issues commands (or actions) to control the thermal states of the system, makes a decision at each event occurrence, called *decision epochs*. These actions and thermal states determine the probability distribution over possible next states. Thus, the sequence of thermal states of the system can be modeled as a stochastic process.

We use a semi-Markov decision process (SMDP) to model the event-driven decision making, but also combine it with a partially observable Markov decision process (POMDP) to consider the uncertainty in temperature observation. Notice that the time spent in a particular state in the SMDP follows an arbitrary probability distribution, which is a more realistic assumption than an exponential distribution. A partially observable semi-Markov decision process (POSMDP) extends the SMDP model by incorporating an observation model, which is defined as follows.

Definition 1: Partially Observable Semi-Markov Decision Process. A POSMDP is a tuple (S, A, O, T, R, Z) such that

1) S is a finite set of states.
2) A is a finite set of actions.
3) O is a finite set of observations.
4) T is a transition probability function. T: $S \times A \rightarrow \Delta(S)$
5) R is a reward function. R: $S \times A \rightarrow \Re$
6) Z is an observation function. Z: $S \times A \rightarrow \Delta(Z)$

where $\Delta(\cdot)$ denotes the set of probability distributions.

Figure 2 shows the POSMDP framework for the dynamic thermal management. At a particular instance in time, a chip temperature state is defined as a range of temperatures. The ranges are in turn defined by the temperature thresholds from the

ACPI (Advanced Configuration and Power Interface) specification [13]. For example, given a single temperature sensor on a chip, and assuming a temperature range of 50 to 90°C, then we define the chip temperature state at time t to be one of three states: s_1, s_2, or s_3 as shown in the figure. The chip temperature evolves over time (i.e., the temperature state of the chip changes) from time t to t+1. Note that the time units are abstractly defined and the task of casting them in absolute time units (micro or milli seconds) is achieved by the system developer based on the time constants associated with heat diffusion among other factors.

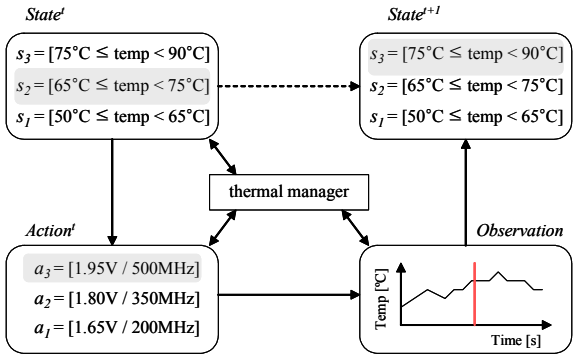


Figure 2. POSMDP Framework.

We assume that the thermal management occurs only when the system is in its active mode. The thermal manager can choose an action from a finite set of actions, i.e., dynamic voltage and frequency scaling (DVFS) sets, at decision epochs. For example, in Figure 2, the action set includes three possible actions corresponding to 1.95V supply and 500 MHz CPU clock speed, 1.80V and 350MHz, etc. The distribution of the time duration between the decision epochs, which is state and action dependent, is given by

$$F(t \mid s, a) = G(t) \quad \text{for} \quad t \geq 0 \tag{2}$$

where $G(t)$ is an arbitrary distribution function arising in turn from characteristics of workload of the system. In partially observable environments, observations are probabilistically dependent on the underlying chip temperature. Transition probability function determines the probability of a transition from thermal state s to thermal state s' after executing action a, i.e., $T(s', a, s) = Prob(s^{t+1} = s' \mid a^t = a, s^t = s)$[1]. Let $p_a(s, s', t)$ denote the probability that as a consequence of choosing action a when the system thermal state is s, the state equals s' after time t. Now, we can compute the probability that the system will be in thermal state s' for the next decision epoch after choosing action a in state s as

$$\begin{aligned} Prob(s' \mid s, a) &= \int_0^\infty p_a(s, s', t)\gamma^t dt \\ &= T(s', a, s)\int_0^\infty F(t \mid s, a)\gamma^t dt \end{aligned} \tag{3}$$

where γ is a *discount factor*, $0 \leq \gamma < 1$. $p_a(s, s', t)$ can be used to calculate the expected transition time between decision epochs. Considering a *cost function*, we assume the conventional approach whereby the expected cost is a lump-sum cost $k(s, a)$ incurred when action a is chosen in state s [10]. An *observation function*, which specifies the relationship between the thermal states and observations, can be defined as the probability of making observation o' in s' after performing a, i.e., $Z(o', s', a) = Prob(o^{t+1} = o' \mid a^t = a, s^{t+1} = s')$.

[1] In this paper, subscripts denote state information whereas superscripts denote time stamp.

B. Policy Representation

In a partially observable environment, since a thermal manager cannot fully observe the underlying temperature profile of the processor, it can make decisions based on the observable system history H, where the system history is a sequence of state and action pair such as $<s^0, a^0>, <s^1, a^1>, \ldots, <s^t, a^t>$, making this non-Markovian process. However, by using the belief state space B, a properly updated probability distribution over the thermal state S, we can convert the original POSMDP into a fully observable SMDP, so-called *belief state* SMDP [14]. The belief state for state s is denoted as $b(s)$, and the sum of belief states over all state is equal to 1. A properly updated belief state $b'(s')$, after action a and observation o, can be calculated from the previous belief state $b(s)$ as follows.

$$b'(s') = \frac{\sum_{s \in S} Prob(s', o \mid s, a) b(s)}{\sum_{s \in S} \sum_{s' \in S} Prob(s', o \mid s, a) b(s)} \tag{4}$$

In this formula, $Prob(s', o|s, a)$ represents the probability of the system moving to state s' when a thermal manager observes temperature o if the previous state is s and the executed action is a, i.e., $Prob(s', o \mid s, a) = Z(o, s', a)T(s', a, s)$.

A thermal manager's goal is to choose a policy that minimizes a cost function, C that is defined on the set of system history H. Let $\pi : B \rightarrow A$ represent a stationary policy that maps probability distribution over states to actions. By incorporating expectation over actions, the cost of a stationary policy π can be determined by using Bellman equation [15] as

$$\begin{aligned} C^\pi(b) = &\sum_{s \in S} b(s) k(s, a) \\ &+ \gamma \sum_{o \in O} \sum_{s' \in S} Z(o, s', a) \sum_{s \in S} T(s', a, s)\, C^\pi(b') \end{aligned} \tag{5}$$

Simply speaking, a thermal manager must execute the action a prescribed by policy π, and then update its probability distribution over the system's thermal states according to (4). The optimal action to take at b is obtained by

$$\pi^*(b) = \arg\min_{a \in A} C^\pi(b) \tag{6}$$

A standard method of finding the optimal infinite policy π is to iterate cost function for SMDP by using a sequence of optimal finite cost functions.

Table 1. Parameter values for a given example.

Action	Description	cost	State , Observation	Description
		[$k(s_1, a)$ $k(s_2, a)$ $k(s_3, a)$]		
a_1	[1.65V / 200MHz]	[1.5 1 0]	s_1 , o_1	[50°C ≤ temp < 65°C]
a_2	[1.80V / 350MHz]	[1 0 1]	s_2 , o_2	[65°C ≤ temp < 75°C]
a_3	[1.95V / 500MHz]	[0 1 1.5]	s_3 , o_3	[75°C ≤ temp < 90°C]

An example of cost iteration for a POSMDP model is given as follows. Consider the POSMDP model of a thermal manager with three states, $S = \{s_1, s_2, s_3\}$, three actions, $A = \{a_1, a_2, a_3\}$, and three observations, $O = \{o_1, o_2, o_3\}$, where parameter values are given in Table 1. Assume that the current belief state is [0.3 0.5 0.2] as shown in Figure 3 (a), where $\Sigma_{s \in S} b_s = 1$. Then, the

value of doing action a_1 in this belief state is 0.95 (= 0.3*1.5 + 0.5*1 + 0.2*0) based on (5). Similarly, actions a_2 and a_3 have values 0.5 and 0.8, respectively. Thus, action a_2 is chosen. Furthermore, since we have a finite number of actions and observations, there are a finite number of possible next belief states. Figure 3 (b) shows the belief state evolution when we fix our first action to be a_2. Even though we know the action with the certainty, the observation is not known in advance since the observations are probabilistic. The reason why there are three possible observations for a given action, e.g., a_2 = [1.8V / 350MHz], is depicted in Figure 4 with the following three cases: a) the system remains in the active mode with steady workload after a_2 is chosen, resulting in the same chip temperature. b) The system remains in the active mode, but temperature increases due to heavy workload after a_2 is chosen. c) The system enters into the idle mode after a_2 is chosen, resulting in low temperature. For a given belief state, each observation has a certain probability associated with it. Suppose that we compute the probability of getting each of the three observations and the costs of the resulting belief states when action a_2 is chosen such as $Prob(s_1,o_1 \mid s_2,a_2)$=0.2, $Prob(s_2,o_2 \mid s_2,a_2)$=0.5, $Prob(s_3,o_3 \mid s_2,a_2)$=0.3 and $C^*(b_1)$=0.7, $C^*(b_2)$=1.2, $C^*(b_3)$=0.8. Then, the cost of belief state when a system chooses a_2 is 1.93, including the immediate cost. Thus, given an initial belief state b^0, the optimal policy can be found by going through the set of all useful policy π and finding the one whose cost function is minimized with respect to b^0.

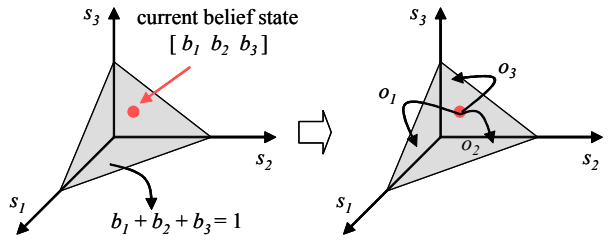
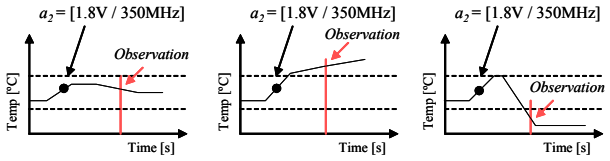


Figure 3. A graphical representation of belief state: (a) current belief state. (b) new belief state when a_2 is chosen.



(a) (b) (c)

Figure 4. Example of three possible observations.

The process of maintaining the belief state is Markovian, which means that the belief state SMDP problem can be solved by adapting the value iteration algorithm. The detail of the value iteration problem for the SMDP, which can be solved in a similar manner as in [10], is omitted to save space.

IV. SDTM

In this section, we present a stochastic dynamic thermal management (SDTM) algorithm based on the POSMDP framework to control the temperature of the system and its power dissipation.

A. SDTM Algorithm

Since the number of actions and observations for the POSMDP is finite, the set of all policies for system history H can be represented by the set of policy trees as shown in Figure 5 (a). Assuming N_o number of observation states, N_a choices of actions, and t-stages policy tree, the size of all possible H-policy tree can be calculated as

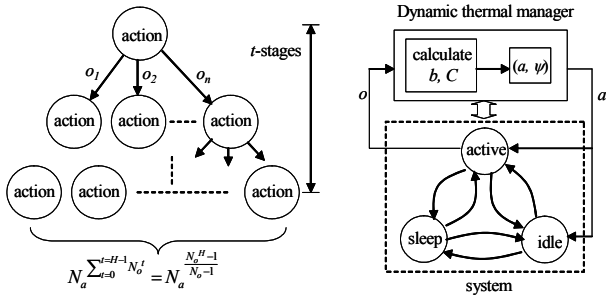
$$N_a^{\sum_{t=0}^{t=H-1} N_o^t} \tag{7}$$

The diagram of SDTM is shown in Figure 5 (b), where the states of the system are divided into three modes. In the active mode, the system can switch between different speed levels, i.e., power-saving states, managed by a thermal manager, resulting in power minimization. In the diagram, we use an observation strategy Λ, a set of pair (a, ψ), which is defined as follows.

Definition 2: Observation strategy. In policy tree, an observation strategy is defined as $\psi : O \rightarrow \Lambda$ such that

1) O is a finite set of observations
2) $\Lambda = \{(a, \psi) \mid a \in A, \psi \in \Lambda^o\}$
   where A is a finite set of actions, and Λ^o is a finite set of all policy trees.

A particular observation strategy tells the SDTM what action to perform, and what to do next contingent on an observation received. Since the observation strategy is stage-dependent, i.e., the observation strategy of t stage can be defined in terms of observation strategy of t-1 stage, a policy tree therefore corresponds to an observation strategy.



(a) (b)

Figure 5. SDTM: (a) policy tree. (b) the diagram of thermal management.

Figure 6 shows the stochastic dynamic thermal management algorithm based on an observation strategy. The SDTM algorithm with n states takes $O(n)$ and $O(n)$ times for finding an optimal action and updating the system history, respectively, which results in total $O(n^2)$ running time.

Algorithm *Stochastic Dynamic Thermal Management Algorithm*

n: the number of states
input: o
1: observe temperature o
2: for i = 1 to n
3: if ($o_i = o$)
4: calculate belief state b
5: calculate cost C
6: do observation strategy ψ
7: for j = 1 to n
8: if ($s_j = o$)
9: update system history H
10: return a

Figure 6. SDTM algorithm.

B. Multi-objective Design Optimization

A multi-objective design optimization method is used to optimize the performance metrics by formulating mathematical programming models.

Considering power metric, we use a joint cost structure such that the expected cost rate, i.e., power consumption, is a sum of the lump-sum cost $k(s, a)$ and a continuous cost rate $c(s', a, s)$, which is given by

$$cost(s,a) = k(s,a) + c(s',a,s) = pow(s) + \frac{1}{\tau(s,a)} \sum_{s \in S} Prob(s' \mid s,a) ene(s,s') \quad (8)$$

where $pow(s)$ is the power consumption of the system in thermal state s, $ene(s, s')$ is the energy required by the system to transit from thermal state s to s', and $\tau(s, a)$ is the expected duration of the time that the system spent in the state s if action a is chosen. Let a sequence of thermal states $s^0, s^1, \ldots, s^k$ denote a processing path δ from s^0 to s^k of length k with the property that $p(s^0, s^1), \ldots, p(s^{k-1}, s^k) > 0$, where $p(x, y)$ is the probability that the system moves to state y from state x. For a policy π, we define the discounted cost C of a processing path δ of length k as follows.

$$C^{\pi}(\delta) \triangleq \sum_{i=0}^{k} \gamma^{t_i} cost(s^i, a^i) \quad (9)$$

where t_i is the time that the system spent in thermal state s^i before action a^i causes a transition to state s^{i+1}. Considering the expectation with respect to the policy π over the set of processing path starting in thermal state s, we can define the expected cost of the system, given that the system starts in state s by $pow^{\pi}_{avg}(s) = EXP[C^{\pi}(\delta)]$.

The main objective is to minimize the on-chip temperature, which is highly dependent on power consumption. The design objective is a vector $\boldsymbol{J}$ of performance metrics we are trying to minimize or maximize. The design vector $\boldsymbol{a}$ contains action sets, i.e., DVFS sets, which impact the design performance. Parameter a is the variable that can be controlled and influences the multi-component objective functions as shown in Figure 7. Note that $\chi(s, a)$ is the frequency that the system is in thermal state s and action a is issued.

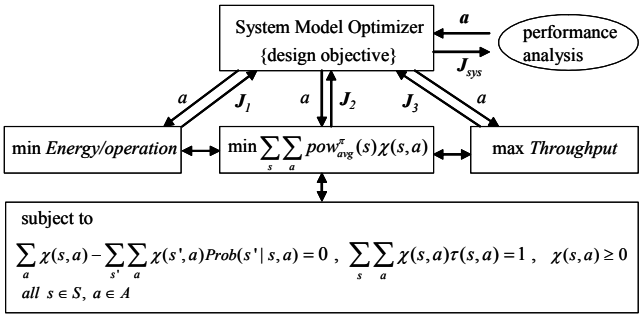


Figure 7. Multi-objective design optimization.

We cannot solve the above optimization problem exactly since there is strong dependency between the various performance metrics. The problem, however, can be solved approximately by using a weighted linear combination of different objectives, where each objective J_i is multiplied by a strictly positive scalar weight λ_i and summed together to form a single-objective optimization problem. We adopt a solution technique in which the weights are *dynamically adjusted* as the optimization proceeds. For example, for min J_1 (energy/operation) and max J_3 (throughput), we formulate the problem as

$$\min f(a) = \lambda_1 J_1(a) + \lambda_3 [J_3(a)]^{-1} \quad (10)$$

λ_i's are set as $\lambda_i = w_i / \sum_j w$, where $w_i = (\partial U / \partial J_i) / (\partial U / \partial J_1)$, and U is the user-specified utility function, i.e., $U = f(\boldsymbol{J})$.

V. EXPERIMENTAL RESULTS

In the experimental setup, we designed a 32bit embedded RISC processor in HDL, which has 3-stage pipeline with basic instruction sets for simplification, and synthesized with 0.18um technology library. The proposed algorithm is implemented in Matlab.

In the first experiment, we analyzed the characteristics of the designed processor in terms of the power dissipation and the operating temperature by executing SPECint2000 benchmarks which have instruction distributions as shown in Table 2.

Table 2. Instruction characteristics for SPECint2000.

SPECint 2000	# of instructions (in millions)	Total dynamic percentage (0% – 100%)
gcc	6765	
gap	12726	
gzip	74942	

load, store, add/sub, and/or, branch, etc.

In order to achieve accurate power value, we obtained SAIF (Switching Activity Interchange File) [16] by back-annotated RTL simulation, and then executed the Power Compiler [16] to analyze the power dissipation distribution as reported in Table 3. The table shows that certain components of the processor such as execution units and register units have a significant impact on the non-uniform power density, which results in hot spot on a die due to the poor thermal conductivity of silicon.

Table 3. Power dissipation distribution in RISC (no cache).

SPECint 2000	dOutreg	dAreg	iAreg	incr	mul	shifter	alu	sr	reg	decode	fetch	execute	busCtl
gcc	4.6%	14.4%	13.8%	4.1%	2.3%	2.7%	16.5%	2.2%	15.4%	4.1%	4.1%	4.1%	11.7%
gap	4.3%	13.1%	15.7%	4.0%	4.2%	3.1%	14.2%	1.7%	14.8%	4.2%	4.2%	4.2%	12.3%
gzip	4.6%	9.2%	15.4%	4.6%	4.5%	3.8%	18.6%	2.3%	15.1%	4.6%	4.6%	4.6%	8.1%

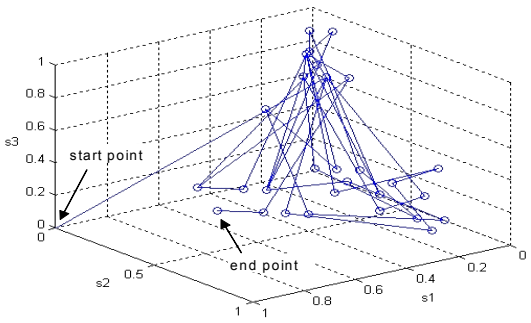


Figure 8. Trace of belief states in temperature observation.

The second experiment is to demonstrate the effectiveness of our proposed SDTM algorithm. First, we randomly chose a sequence of 40 programs which include gcc, gap, and gzip such as gcc_1-$gzip_2$-gap_3-gap_4-…-gcc_{39}-gcc_{40}, where $program_i$ is the i-th program in the sequence. Then, the sequence of programs is executed on the RISC processor to calculate the belief states by

using the parameters that are given in Table 1. The trace of the belief states in temperature observation is depicted in Figure 8.

Simulation results in Figure 9 show the average operating temperature of the processor. As a comparison, we also computed the average temperature of processor which runs at 1) 1.95V / 500MHz, 2) 1.8V / 350MHz, and 3) 1.65V / 200MHz. From the figure, we can see clearly that the processor with the SDTM is selecting the optimal actions, which in turn result in low average temperature as shown in Figure 10.

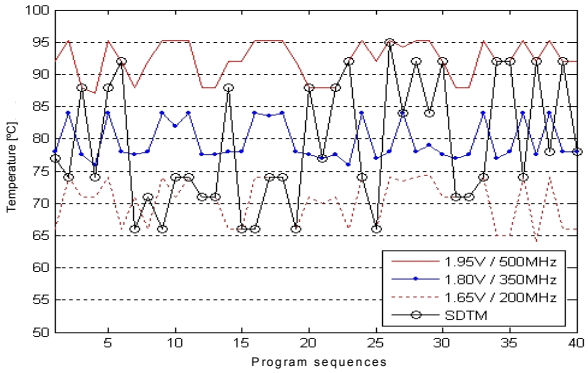


Figure 9. Temperature variation for test scenario.

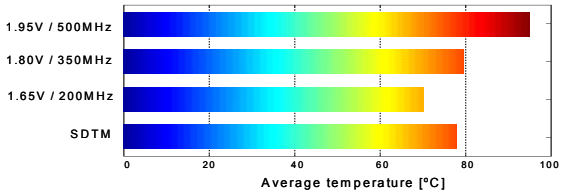


Figure 10. Average temperature for test scenario.

Figure 11 shows the values of possible target actions, i.e., low action means low supply voltage and low frequency, and vice versa, obtained during multi-objective design optimization. This figure indicates that the performance metrics are adjusted gradually and trade-offs between power consumption and others (throughput and energy/operation) become more evident with repeated solutions based on the equation (10).

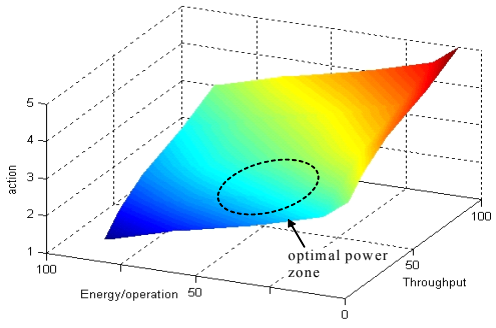


Figure 11. Optimization with weighted sum approach (normalized).

Optimization result with average CPI (Clock Per Instruction) = 1.6, as shown in Table 4, indicates that the SDTM algorithm gives low power consumption and operating temperature with little performance impact on throughput and energy/operation metrics.

Table 4. Normalized values for optimization results.

	1.95V/500MHz	1.80V/350MHz	1.65V/200MHz	SDTM
Average power	1.00	0.85	0.72	0.75
Throughput	1.00	0.69	0.40	0.69
Energy/operation	1.00	1.23	1.79	1.08

VI. CONCLUSION

We propose a stochastic dynamic thermal management technique by providing a stochastic management framework to improve the accuracy of decision making under the uncertainty in temperature observation. The proposed thermal management framework, based on POSMDP, controls the thermal states of the system and makes a decision (i.e., DVFS set) to reduce operating temperature. Experimental results with design optimization formulations demonstrate the effectiveness of our algorithm.

REFERENCES

[1] E. Rohou and M. Smith, "Dynamically managing processor temperature and power," *Proc. of FDDO-2*, Nov. 1999.

[2] K. Skadron, M.R. Stan, et al., "Temperature-Aware Microarchitecture," *Proc. of Int'l Symposium on Computer Architecture*, June 2003.

[3] D. Brooks and M. Martonosi, "Dynamic Thermal Management for High Performance Microprocessor," *Proc. of HPCA*, Jan. 2001.

[4] J.Srinivasan and S.V.Adve, "Predictive Dynamic Thermal Management for Multimedia Applications," *Proc. of 17th Annual ACM Int'l Conference on Supercomputing*. June 2003.

[5] S. Gurumurthi, et al., "Disk Drive Roadmap from the Thermal Perspective: A Case for Dynamic Thermal Management," *ACM SIGARCH Computer Architecture News*. Vol.22, Issue 2. May 2005.

[6] P. Dadvar and K. Skadron, "Potential Thermal Security Risks," *Proc. of 21st IEEE Semi-Therm Symposium,* Mar. 2005.

[7] L. Benini and G. De Micheli, *Dynamic Power Management: Design Techniques and CAD Tools*, Kluwer Academic Publishers, 1997.

[8] Q. Qiu, Q. Wu, and M. Pedram, "Stochastic Modeling of a Power-Managed System - Construction and Optimization," *IEEE Trans. on Computer-Aided Design*, Vol. 20, No. 10, Oct. 2001.

[9] T. Simunic and S. Boyd, "Managing Power Consumption in Networks on Chips," *Proc. of DATE*. March 2002.

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

[11] Y. Cheng, C. Tsai, C. Teng, and S. Kang, *Electrothermal Analysis of VLSI Systems*. Kluwer Academic Publishers, 2000.

[12] http://www.spec.org. CPU SPECint2000 documents.

[13] http://www.acpi.info/spec.htm. Advanced Configuration and Power Interface Specification, Rev. 3.0a. Dec. 2005.

[14] A.R. Cassandra, et al., "Acting Optimally in Partially Observable Stochastic Domains," *Proc. of 12th Conference on AI*, Aug. 1996.

[15] R.E. Bellman, *Dynamic Programming*. Princeton University Press, Princeton, 1957.

[16] http://www.synopsys.com. Synopsys Power Compiler Documents.