A Variation-Aware Adaptive Fuzzy Control System for Thermal Management of Microprocessors

Yingnan Cui, Wei Zhang, Bingsheng He

***Abstract*—Thermal failures pose severe threats to the reliability and the performance of modern microprocessors. To deal with the threats, various kinds of thermal management techniques are proposed. For systems with strict temperature constraints, closed-loop thermal controllers are preferred. Because they have the advantages of high control accuracy and high speed of response. However, most closed-loop thermal controllers proposed in previous studies are sensitive to the variations of the system model, which are inevitable due to process variation, sensor noises and environmental influences. In this paper, we propose an adaptive fuzzy controller for the thermal management of microprocessors. The adaptive fuzzy controller has low sensitivity to noises and high adaptivity to system model variations. In the experiments, our adaptive fuzzy controller maintains the control quality when faced with severe variations of the system model. It achieves up to 18.5% better performance and up to 14.2% longer lifespan of the microprocessors when compared with other state-of-the-art thermal controllers.**

I. INTRODUCTION

As the semiconductor technology steps into deep-sub-micron era, power density of microprocessors has nearly reached a prohibitive level [1]. High power density has caused many problematic issues, and high processor temperature is one of the most severe problems. Due to the physical nature of semiconductor devices, high temperature could cause many thermal-related failures to processors [2], [3], [4]. As a result, efficient thermal management techniques are highly demanded by modern microprocessors.

Among various thermal management techinques, dynamic voltage and frequency scaling (DVFS) is widely adopted. The DVFS-based techniques scale down the voltage and frequency level of a microprocessor during thermal emergencies to prevent the processor from being over-heated. In the view of control theory, most DVFS-based thermal management techniques [5], [6], [7] can be categorized as open-loop control systems. However, the open-loop control systems suffer from three major disadvantages. First, the speed of response to thermal emergencies is slow. As a result, the processors work under high temperature for longer time, which harms the reliability and lifespan of the microprocessors. Second, the temperature of processors with open-loop thermal control systems fluctuates severely, which also could cause malfunction of processors [4]. Finally, open-loop thermal control systems cannot guarantee the upper bound of peak temperature of processors. In order to avoid thermal-failures, open-loop thermal control systems usually set large margins between the theoretical threshold temperature and the real threshold temperature applied in the processors. With the tighter thermal constraint, the performance of the processors becomes worse.

To achieve higher control quality, closed-loop DVFS-based thermal control systems are proposed. Guaranteed by control theory, closed-loop thermal control systems provide higher temperature control accuracy, faster speed of response and less temperature fluctuation, when compared to the open-loop thermal control systems. The first closed-loop thermal controller for microprocessors is proposed in [8], which adopts the PID controller design. As one of the most classic designs in control theory, the PID controller has low complexity and is feasible for a wide range of systems. [9] proposes an optimal controller design for the thermal management of microprocessors. The optimal controller achieves optimal performance of a processor with a temperature constraint.

However, the PID controller and the optimal controller are both sensitive to the variations of the thermal model of the processors. When the thermal model of the real processor deviates from the norminal model used during the design phase, the performance of the thermal controller could be significantly affected. In reality, the variations of the thermal model of a processor is inevitable and unpredictable. There are three major factors that lead to the variations of the thermal model. Firstly, CMOS thermal sensors implemented within the processors usually have a considerable amount of noises [10]. Secondly, in the deep-sub-micron era, the process variation brings uncertainty to the dynamic and leakage power model of a processor. Finally, the ambient temperature of the surronding envrionment also affects the heat dissipation speed of a processor.

Due to the variations of the thermal model, it is difficult to guarantee the performance of a thermal controller when a model-sensitive design isadpoted. However, this challenge could be solved by using the fuzzy controller. Firstly, the design of a fuzzy controller does not rely on the thermal model of a system. Secondly, according to control theory, the fuzzy controller has higher robustness in nature. In other words, the fuzzy controller is less sensitive to the variations of the thermal model compared to the PID controller and the optimal controller. Thirdly, the complexity of the fuzzy controller is much lower than other kinds of advanced controllers with high robustness and adaptiveness, like the neural-network-based or the Kalmen-filter-based controllers. Coskun *et al.* [11] first apply the fuzzy controller in a 3D-integrated processor to control the temperature of the processor through DVFS and micro-fluid cooling. This study does not consider the variations of the thermal model and the fuzzy controller proposed in the study does not have adaptivity.

To further deal with the variations of the thermal model, we propose an adaptive fuzzy controller for the thermal

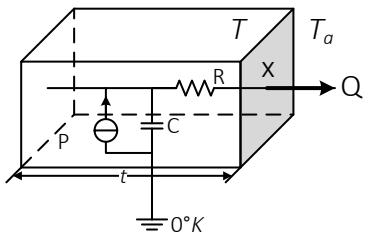


Fig. 1: The thermodynamic model for general solid materials

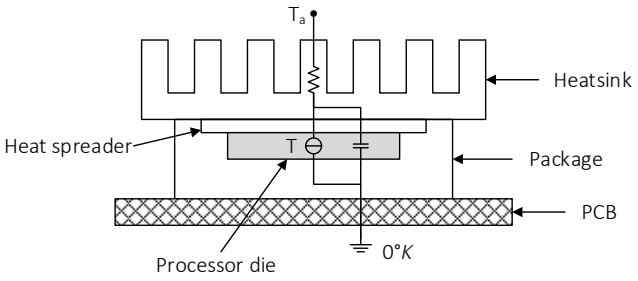


Fig. 2: Side view of a microprocessor and its thermal model.

management of microprocessors. In our design, the parameters of the controller are automatically adjusted according to the feedback of the temperature measurement and best fit the variantions of the thermal model caused by sensor noises, process variation and the environment change. To our best knowledge, our work is the first adaptive fuzzy controller for thermal management of microprocessors. In the experiments, our adaptive fuzzy controller maintains the control quality when faced with severe variations of the system model. It achieves up to 18.5% better performance and up to 14.2% longer lifespan of the microprocessors when compared with other state-of-the-art thermal controllers.

The rest of this paper is organized as follows: Section II discusses the necessity of adopting fuzzy controller. Section III gives a panoramic view of the fuzzy control system. Section IV gives the detailed description of our adaptive fuzzy controller. Section V extends the use of the adaptive fuzzy controller into multi-core processors. Section VI evaluates the performance of our fuzzy controller and compares our design with other state-of-the-art thermal controllers. Section VII introduces the related studies of this work. Finally, Section VIII concludes this paper.

II. Motivation

In this section, we briefly introduce the thermal model, power model and process variation model for microprocessors. Based on these models, we build a model for the whole thermal control system of the microprocessors. We then show how the various kinds of variations of the thermal model affects the performance of the controller. Finally, we explain the reasons of adopting the adaptive fuzzy controller in this work.

A. Thermal Model for Microprocessors

The basic equations to describe thermodynamics of solid materials share the same mathematical forms with the equations of electrical circuits. Based on this duality, the thermal model of solid materials can be visualized as an equivalent circuit. Fig. 1 shows the thermal model of a small cube of materials in the form of an circuit. For ease of discussion, we assume that in the cube all the surfaces are heat-insulate, except for the surface labeled as x. In other words, the heat can only passes through surface x. Inside the cube, there exists a source which produces heat inside the cube. The heat source is modeled by the current source in Fig. 1. The heat generated by the source in unit time is denoted as P. Besides, the thermal resistance of the cube is denoted by the resistor R and the thermal capacitance is denoted by the capacitor C in the circuit. According to thermodynamics theory, the speed of the heat flowing through a surface is affected by the temperature difference between two sides of the surface. We assume the temperature of the cube is T, and the temperature of the ambient environment is T_a. With the above assumptions, the thermodynamics equation of the cube of materials is defined by Eq. (1).

$$P = \frac{T - T_a}{R} + C \cdot \frac{dT}{dt} \tag{1}$$

From Eq. (1), we can see the heat generated inside a cube has two destinations. One part of the heat passes through surface x and dissipates into the ambient environment. The amount of the heat dissipated in unit time is defined by the first term in Eq. (1). The rest of the heat is absorbed by the materials in the cube which leads to the temperature rising of the cube. The second term in Eq. (1) defines the relationship between the amount of heat absorbed by the cube and the temperature rising speed. Assume in Fig. 1, the area of surface x is A, and the thickness of the cube is t. We can compute the thermal resistance and thermal capacitance of the cube according to Eq. (2) and Eq. (3) respectively, where k is the thermal resistivity of the materials, and c is the thermal capacity. Both k and c are defined by the physical nature of the materials.

$$R = \frac{t}{k \cdot A} \tag{2}$$

$$C = c \cdot t \cdot A \tag{3}$$

Based on Eq. (1) to Eq. (3), we can build the thermal model for a microprocessor. Fig. 2 shows the side view of a microprocessor with its package. The die of the processor is surrounded by its package. Within the package, there is a metal-made heat spreader that is directly attached to the die to faciliate the heat dissipation. Outside the chip, there is a heatsink to further accelerate heat dissipation speed. For ease of discussion, we assume the heat produced by the microprocessor can only be dissipated through the heatsink into the air. Based on this assumption we build a thermal model for the chip which is represented by the RC circuit in Fig. 2.

B. Power Model

In a thermal control system, the controller adjusts the power consumption of the microprocessor by changing the voltage level. Therefore, it is necessary to understand the relationship between the voltage level and the power consumption of the processor. We note that in a processor, the functional circuits and memory cells have different power models. Again, for the ease of discussion, we only introduce the power model for functional circuits. The power of an IC chip contains two parts: the dynamic power and the static power. The dynamic power is the power consumed by the switching of the gates. The static power is mainly the power consumed by the leakages of the circuit.

Eq. (4) shows the definition of dynamic power of a chip (denoted by P_d) [12], where C is the capacitance of the load that the circuit drives, V_{dd} is the supply power of the circuit, f is the clock frequency of the circuit, and α is the activity factor that describes the average ratio of gate switches in the whole circuit. When a microprocessor is at work, α can vary according to the different types of instructions that the processor executes.

$$P_d = \alpha \cdot C \cdot V_{dd}^2 \cdot f \tag{4}$$

The static power of a microprocessor contains three kinds of leakage power: the subthreshold leakage, the gate leakage and the junction leakage. The gate leakage and junction leakage are negligible when compared to the subthreshold leakage [12]. Therefore, we only discuss the subthreshold leakage in this study. According to the model proposed in [13], Eq. (5) defines the subthreshold leakage of a integrated circuit, denoted by P_{sub}. In the equation, N is the number of gates in the circuit and I_{sub} is the average subthreshold leakage current per gate. The definition of the average subthreshold leakage current I_{sub} is defined in Eq. (6) and Eq. (7), where V_{gs}, V_{sb} and V_{ds} are the gate-source, source-bulk and drain-source voltages respectively, V_T is the zero-biased threshold voltage, V_{th} is the thermal voltage kT/q, γ' is the linearized body-effect coefficient, η is the drain induced barrier lowering (DIBL) coefficient, μ_0 is the carrier mobility, C_{ox} is gate capacitance per area, W is the width of the gate and L_{eff} is the effective length of the gate. According to the quations, the average subthreshold leakage current per gate is affected by the supply voltage V_{dd}, the chip temperature T and the geometery characteristics of the CMOS gates.

$$P_{sub} = N \cdot I_{sub} \cdot V_{dd} \tag{5}$$

$$I_{sub} = A \cdot e^{\frac{(V_{gs} - V_T - \gamma' V_{sb} + \eta V_{ds})}{n V_{th}}} (1 - e^{-\frac{V_{ds}}{V_{th}}}) \tag{6}$$

$$A = \mu_0 \cdot C_{ox} \cdot \frac{W}{L_{eff}} \cdot V_{th}^2 \cdot e^{1.8} \tag{7}$$

C. Process Variations

Fig. 3 shows the cross-section view of a nMOS transistor. The geometry parameters of a gate differ from the designed values due to the process variation. The variations in factors like gate length (L), gate oxide thickness (t_{ox}) and channel doping (N_{sub}) affect the electronic characteristics of the transistor. As the feature size of the CMOS circuits shrinks, the influence of the process variation is becoming more and more important.

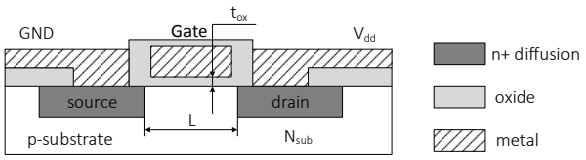


Fig. 3: Cross-section view of nMOS transistor.

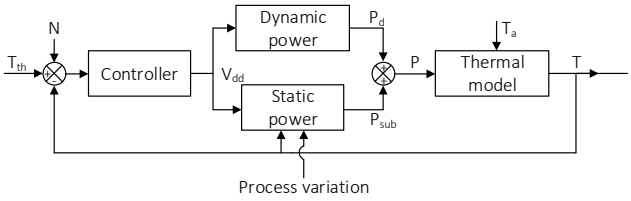


Fig. 4: The block diagram for the thermal control system.

Due to short-channel effects, the leakage power of a transistor is most sensitive to process variation. In studies like [14], the distribution of the geometery parameters of the transistors is usually assumed to be Gaussian. The average change in the leakage power could be evaluated through Monte Carlo simulations using Hspice. Experimental results in [3] show that when 10% of variation is assumed for the three parameters (L, t_{ox} and N_{sub}), the average subthreshold leakage current (I_{sub}) changes up to 31.1%.

The statistical model for process variation is too complicated for analysing and experiments in this study. Therefore, we simply denote the influence of the process variation on the subthreshold leakage power using a parameter δ. We assume the subthreshold leakage power of a microprocessor is δP_{sub}, where P_{sub} is the nominal subthreshold leakage power without the process variation.

D. Control System Model

The block diagram of a closed-loop thermal control system for a microprocessor is shown in Fig. 4. The machanism of the system is as follows. First we set a threshold temperature for the microprocessor, denoted as T_{th}. The controller then decides the supply voltage level V_{dd} according to the difference between the system temperature, denoted by T, and the threshold temperature. The voltage level determines the power consumption of the processor, including both the dynamic power and the static power. Finally, according to the thermal model shown in Section II-A, the power consumption determines the temperature of the processor. According to Eq. (6), the temperature also affects the leakage power of the processor. This relationship is shown by the inner loop between the static power and the thermal model in Fig. 4. In Fig. 4, the variations of the model are also identified. Firstly, the temperture sensor of the processor contains a noise signal, denoted by N. Secondly, the process variation has significant

yet unpredictable influence on the static power. Finally, the ambient temperature T_a may vary due to environment change.

The system shown in Fig. 4 is a non-linear system, which makes it difficult to analyze within limited pages. For ease of discussion, we linearize the system model. The nonlinearity of the model comes from the power model. According to Eq. (4), the dynamic power model is linearized as follows. We use a new variable X to denote the output of the controller, where $X = V_{dd}^2$. In Fig. 4, the dynamic power model block becomes a simple proportional module, where the proportional parameter is denoted by K_1. For the static power, we linearize the model by applying first order taylor expansion on Eq. (6) around the voltage level V_0. The result of the taylor expansion is shown by Eq. (8). We note that P_{sub} is dependent on temperature. But in a closed-loop thermal control system, the temperature of the processor is stable around the threshold temperature. Therefore, we assume the temperature is a constant in the leakage power model. For modern processors, V_{dd} is usually smaller than $1V$ [1], it is reasonable to set $V_0 = 0$ in the taylor expansion. According to Eq. (6), when $V_{dd} = 0$, there is $I_{sub} = 0$. Then, by substituting the I_{sub} in Eq. (5) by Eq. (8), we get Eq. (9). By using the variable X to denote the controller output, we transform the static power model to another proportional module with the proportional parameter denoted by K_2.

$$I_{sub} = I_{sub}(V_0) + I'_{sub}(V_0) \cdot V_{dd} \tag{8}$$

$$P_{sub} = N \cdot I'_{sub}(0) \cdot V_{dd}^2 \tag{9}$$

After the linearization, we apply the Laplace transformation to the system model and get a new system model shown in Fig. 5. In control theory, by applying the Laplace tranformation on the system model, we get the so-called *transfer function* model. The convience of using the transfer function model is that we can build the system model by directly multiplying the tranfer functions of each block along the signal passing direction in the block diagram of the system. For instance, the closed-loop system model of Fig. 5 is as follows:

$$(T_{th}(s) - T(s)) \cdot H(s) \cdot (K_1 + K_2) \cdot G(s) = T(s) \tag{10}$$

We can further transfrom this function into the following equation:

$$\frac{T_{th}(s)}{T(s)} = \frac{(K_1 + K_2) \cdot H(s) \cdot G(s)}{1 + (K_1 + K_2) \cdot H(s) \cdot G(s)} \tag{11}$$

The transfer function describes the characteristics of a system in the frequency field. The variable s in the transfer function represents the frequency level of a signal. A well-performing and stable closed-loop thermal control system should meet the following two standards. Firstly, the system should be able to maintain the temperature of the system around the threshold temperature. Secondly, the temperature of the system should not be sensitive to the noises in the system. To achieve the goals, we want the transfer function of the system to have the following features. Firstly, when s is small, we should make $(K_1 + K_2) \cdot H(s) \cdot G(s) \gg 1$. Because when we apply Laplace tranformation to the input signal of the system, which is the threshold temperature, we can find that the input singal is mainly composed of singals with low frequency levels. According to Eq. (11), when $(K_1 + K_2) \cdot H(s) \cdot G(s) \gg 1$, $T_th(s)/T(s) \approx 1$. That means the output temperature is equal to the input singal (the threshold temperature). Secondly, when s is large, we should make $(K_1 + K_2) \cdot H(s) \cdot G(s) \ll 1$. This is because the noises usually contains more high frequency signals, and we do not want the temperature of the system to trace these high frequency signals.

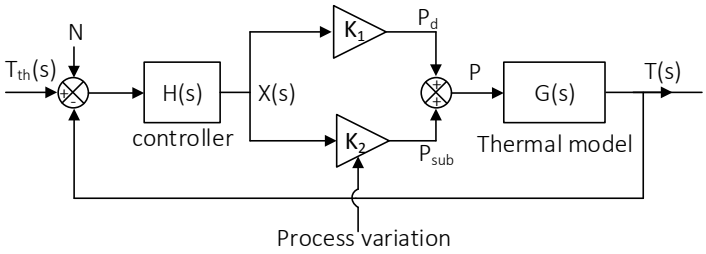


Fig. 5: The transfer function model for the thermal control system.

Based on the above discussion, we can deduce how the variation in thermal model affects the performance of the system. From Fig. 5, we can see that among the three major sources of variations, the process variation has directly changes the transfer function of the system. Therefore, we first discuss the influence of the process variation.

We assume that for the system without the variations, a perfectly designed thermal controller has the transfer function denoted by $H(s)$. The process variation changes the parameter K_2 in Fig. 5 to a new value K'_2. The influence of the change in the parameter K_2 is as follows. According to the previous discussion, if $K'_2 > K_2$, the system becomes more sensitive to noises compared to the original system. And of $K'_2 < K_2$, the system becomes less accurate in tracing the input signal of the system.

According to previous studies, in 65nm or smaller technology, the static power of a chip is nearly equal to the dynamic power [15]. Which means in the model shown in Fig. 5, there is $K_1 \approx K_2$. According to previous studies [16], the process variation can lead to the changing of K_2 by more than 50%. In that case, the controller designed without considering variations in the system model can lead to malfunction in real processors.

The influence of the variation in ambient temperature and the sensor noises can be explained as follows. In the system model, we now assume the output of the system is $T - T_a$. This assumption does not change the tranfer function of the sytem because in Eq. (1), the derivative of $T - T_a$ is the same as T. With the new assumption, our system model changes to Fig. 6. In the new model, the input of the system is $T_{th} - T_a$. It means that the output of the system must trace $T_{th} - T_a$. We assume that the variation of T_a is denoted by ΔT_a. Adding ΔT_a with the sensor noise N, we get a combined noise signal $\Delta T_a + N$. The influence of this combined noise signal to the system depends on the sensitivity of the system model to noises. When the system model changes due to the process

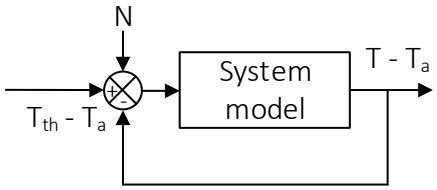


Fig. 6: The system model with $T - T_a$ as output.

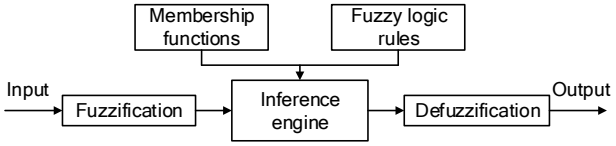


Fig. 7: The standard structure of a fuzzy controller.

variation, the influence of this noise signal also changes.

The above discussions are all based on the linearized system model. In reality, the nonlinearity of the system model could worsen the case. Therefore, a adaptive controller is required to deal with the variations of the system model. In addition, since the variations of the system model is nearly impossible to be accurately measured during run-time. It makes the model-based controllers like the PID controller and the optimal controller very difficult to design. The design of the fuzzy controller does not require an accurate system model. As a result, an adaptive fuzzy controller could be an ideal solution to the microprocessors with various sources of variations.

III. Preliminaries of Fuzzy Controllers

The fuzzy control theory is based on the fuzzy set theory which is proposed by L. A. Zadeh in 1965. As the word "fuzzy" indicates, the fuzzy set theory is designed to depict problems with vagueness where traditional mathematical theories are difficult to be applied. Based on the fuzzy set theory, the fuzzy controller imitates the reasoning process of a human.

Fig. 7 shows the standard structure of the fuzzy controller. The fuzzy controller is composed of three major modules: the fuzzification module, the inference engine and the defuzzification module. The fuzzification module converts the input variables into the so-called *linguistic variables*, which are sent into the inference engine. The inference engine decides the output of the controller according to the fuzzy logic rules and the membership functions. The output of the inference engine is also a linguistic variable. The defuzzification module converts the output linguistic variable into a numeric variable, which is the final output value of the fuzzy controller.

In a fuzzy controller, all the operations are related to the linguistic variables. As indicated by the name, the values of a linguistic variable are a group of words which are used to describe amount or size, like "High" , "Low", and *etc.* For a linguitic variable, each of its possible value is assigned with a membership function, which shows the level of accuracy of using that value to describe the related numeric variable. For example, assume there is a linguistic variable denoted by A. A is used to evaluate the value of a numerical variable α.

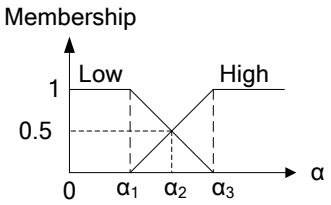


Fig. 8: The membership functions for linguistic variable A.

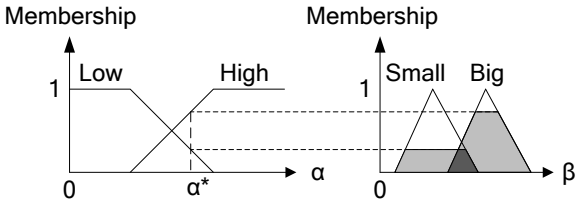


Fig. 9: Inference process of a fuzzy controller.

Assume A has two linguistic values: "High" and "Low". Fig. 8 shows the membership functions of the linguistic variable A. For each possible value of α, the membership function defines the membership value of α for the related value of the linguistic variable. When $\alpha < \alpha_1$, the membership of α for "High" is 0 and the membership for "Low" is 1. In this case, we can say that α is 100% "High" and 0% "Low". And when $\alpha = \alpha_2$, we can say that α is 50% "High" as well as 50% "Low".

The nature of the linguistic variable makes it capable of describing things with vagueness. For instance, when we talk about the temperature in our lifes, we do not have an accurate standard for defining high temperature and low temperature. In extream cases like when temperature is over $40°C$ or under $0°C$, we have the clear notions of high and low. But for the temperature between this range, the definition of high and low becomes vague. The linguistic variable quantifies this vagueness through the membership function and thus provides a tool for converting human thinking process into mathematical operations.

After the fuzzification module converts the input of the fuzzy controller into a linguistic variable, the inference engine then decides the output of the fuzzy controller. The inference process is based on the fuzzy logic rules, which are a group of conditional statements defining the output of the controller under different input values. The input and output are both represented by linguistic variables. For example, assume in a fuzzy controller, the input linguistic variable is A and the output linguistic variable is B. The possible value of A is "High" or "Low" and the possible value of B is "Big" or "Small". Then a possible set of fuzzy logic rules of this controller could be as follows:

- If A is "High", then B is "Big".
- If A is "Low", then B is "Small".

For ease of discussion, we continue to use this example to introduce the inference engine of the fuzzy controllers. Assume at a time, the input of the fuzzy controller α has the value α^*. The fuzzification module converts the input α^* to a linguistic variable. The values of the membership functions of

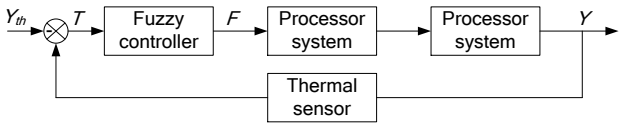


Fig. 10: The structure of the self-adaptive fuzzy control system for thermal management of microprocessors.

the linguistic variable are denoted by $M_{High}(\alpha^*)$ (for "High") and $M_{Low}(\alpha^*)$ (for "Low"). Based on the fuzzy logic rules the inference engine changes the membership functions of B as defined in Eq. (12) and Eq. (13). In Eq. (12), $M^*_{Big}(\beta)$ is the membership function of value "Big" after inference and $M_{Big}(\beta)$ is the original membership function. Similarly, in Eq. (13), $M^*_{Small}(\beta)$ is the new membership function and the $M_{Small}(\beta)$ is the original one. Fig. 9 shows the inference process. The new membership functions of B is represented by the shaded areas.

$$M^*_{Big}(\beta) = \begin{cases} M_{Big}(\beta),\ M_{Big}(\beta) < M_{High}(\alpha^*) \\ M_{High}(\alpha^*),\ M_{Big}(\beta) > M_{High}(\alpha^*) \end{cases} \tag{12}$$

$$M^*_{Small}(\beta) = \begin{cases} M_{Small}(\beta),\ M_{Small}(\beta) < M_{Low}(\alpha^*) \\ M_{Low}(\alpha^*),\ M_{Big}(\beta) > M_{Low}(\alpha^*) \end{cases} \tag{13}$$

The output of the inference engine is the linguistic variable B with the new membership functions $M^*_{Big}(\beta)$ and $M^*_{Small}(\beta)$. The defuzzification module converts B into a numberic value, which is the final output of the fuzzy controller. A commonly used defuzzification method is the center of gravity method (COG), which use the horizontal coordinate of the center of gravity of the shaded area shown in Fig. 9 as the output of the fuzzy controller. Assume the outline of the shaded area is defined by a function $U(\beta)$, then the output of the fuzzy controller,β^*, given by COG method is defined in

$$\beta^* = \frac{\int U(\beta) \cdot \beta d\beta}{\int U(\beta) d\beta} \tag{14}$$

IV. Self-Adaptive Fuzzy Controller Design

As discussed in Section II-A, we have three major reasons to use adaptive fuzzy controller in this paper. Firstly, factors like temperature sensor noises, the process variation and the ambient temperature makes it difficult to build an accurate thermal model for the microprocessor. Compared with model-depending controllers like the PID controller and the optimal controller, the fuzzy controller does not rely on the model of the system. Secondly, the fuzzy controller has high robustness. It is less sensitive to the noises in the system than many other kinds of controllers. Finally, the variations of the thermal model of microprocessors require adaptivity for the thermal controllers.

Fig. 10 shows the structure of the closed-loop thermal control system with the proposed adaptive fuzzy controller. The closed-loop system works as follows. Assume the current processor temperature is Y. The measured temperature by thermal sensor, denoted as Y_m, is compared with the threshold temperature, denoted by Y_{th}. According to the difference of Y_m and Y_{th}, denoted as T, the fuzzy controller decides the frequency level of the processor, denoted as F. The DVFS control module regulates the frequency and voltage level of the processor according to F. This changes the power profile of the mircoprocessor and therefore controls the temperature of the chip.

In the process, the adaptive module keeps track of T and Y_m to detect the variations in the output responses of the microprocessors. By analyzing T and Y_m, the self-adaptive module decides how to set the parameters in the fuzzy controller to adapt to the variations in thermal related parameters of the microprocessor. In designing the self-adaptive control system, we have two objectives. First, the control quality should be guaranteed; and second, the complexity of the controller should be minimized. The following two sections respectively introduce our design of the fuzzy controller and its self-adaptive modules.

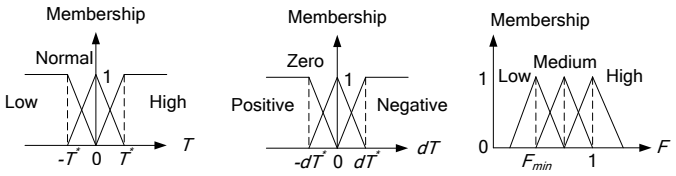
A. Fuzzy Controller Design

The basic structure of the fuzzy controller is introduced in Section III. We design the fuzzy controller in the following order. Firstly, we decide the input and output of the controller. Secondly, we design the linguistic variables and the related membership functions for the input and the output of the controller. Third, we design the fuzzy logic rules. Finally, we select the best parameters for the fuzzy controller.

As shown in Fig. 10, the difference between the measured processor temperature and the temperature threshold T, should be an input of the fuzzy controller. Besides, we also use the temperature changing speed as another input for the fuzzy controller, denoted by dT. dT reflects the trend in the changing of the temperature of the processor. Using dT as an additional input, the controller could predict the temperature change and take proactive moves to avoid thermal emergencies. The output of the fuzzy controller is the frequency level of the processor, denoted by F. F is normalized to maximum frequency level of the system.

The linguistic variables and the related membership functions of the fuzzy controller is shown in Fig. 11. Each linguistic variable is assigned with three values. For the input linguistic variables, three linguistic variables are the minimum number of linguistic values that cover all the fundamental situations. Increasing the number of values for each linguistic variable could significantly increase the complexity of the system. For the membership functions of the linguistic variables, we adopt the symmetric triangular function, which is one of the most commonly used kind of membership functions. The membership functions of the input and the output of the controller is shown in Fig. 11. F^* in Fig. 11a and dF^* in Fig. 11b can be adjusted to achieve better performance of the system. F_{min} in Fig. 11c is the minimal frequency level of the processor. Table I shows the fuzzy logic rules for the controller, which are designed by the "common sense" of people. For the defuzzification module, we adopt the COG defuzzification method as mentioned in Section III.

In searching the optimal parameters of the fuzzy controller T^* and dT^*, we adopt the random search methods. This is



(a) T (b) dT (c) F

Fig. 11: Membership functions of the linguistic variables.

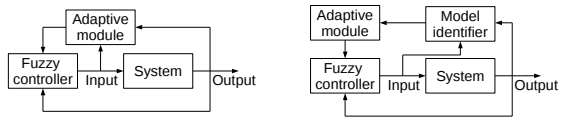
TABLE I: The fuzzy logic rules for our fuzzy controller.

If		Then
T is	dT is	F is
Low	Negative	High
Low	Zero	High
Low	Positive	Medium
Normal	Negative	High
Normal	Zero	Medium
Normal	Positive	Low
High	Negative	Medium
High	Zero	Low
High	Positive	Low

because the non-linear feature of fuzzy control systems makes it impossible to optimize the parameters of the fuzzy controller analytically. First we set the searching ranges for T^* and dT^*. The searching range of T^* is set to $[0, T_{max}]$. Where T_{max} is the highest temperature overflow compared to the threshold temperature allowed by the system. T_{max} is usually a design objective of the thermal controller. The searching range of dT^* is set to $[0, dT_{max}]$. Where dT_{max} represents the highest possible temperature rising speed in a processor. We use the Monte Carlo methods to search the optimal parameters in the searching range. The objective of the random search is to minimize the overflow of the processor temperature over the threshold temperature.

B. Self-Adaptive Module Design

As mentioned in Section I, the objective of the adaptive module of the fuzzy controller is to make the controller adaptive to the variations of the thermal model. To achieve the goal, the adaptive control theory provides two different kinds of adaptive modules [17]. Shown in Fig. 12a, the first kind of adaptive modules analyze the variations in system model and adjust controllers to eliminate the influence of the variations. This kind of adaptive modules contain model identifiers which are too complicated to be implemented in microprocessors. Shown in Fig. 12b, the second kind of adaptive modules compare the output of the system with an ideal output produced by a reference model, and adjust the controller to make the system output to trace the ideal output.



(a) The model-identifier-based adaptive module. (b) The reference-model-based adaptive module.

Fig. 12: The structures of adaptive modules.

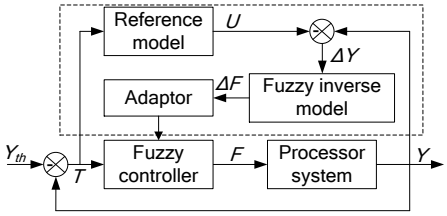


Fig. 13: The block diagram of the adaptive fuzzy controller.

The reference model mentioned here is the model of an ideal system with makes the closed-loop control system stable and accurate. This kind of adaptive modules are called model reference adaptive (MRA) modules.

Fig. 13 shows the structure of the MRA-based adaptive fuzzy controller. The mechanism of the MRA module is as follows. Based on the same input of the fuzzy thermal control system, the reference model produces the ideal output for the thermal control system to trace. Then the difference of the real temperature of the system and the ideal output of the reference module is sent to a fuzzy inverse model which use fuzzy logics to decide the additional amount of frequency that should be adjusted to the processor to eliminates the difference between the real output of the system and the ideal output of the reference model. Finally, the adapter applies the additional amount of the frequency to the output of the fuzzy controller by adjusting the parameters of the fuzzy controller.

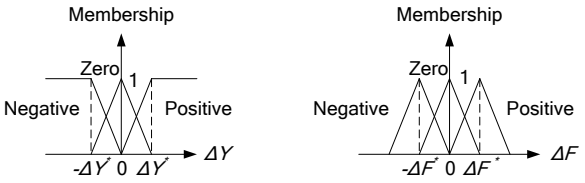
In the design, a good reference model is critical for the performance of the adaptive fuzzy control system. As shown in Fig. 13, the input of the reference model is the difference between system temperature and the threshold temperature, denoted as T. And the ideal output of the reference model is denoted by U. As discussed in Section II-D, the transfer function of an ideal system, denoted by $G(s)$, should meet the condition. When $s \ll 1$, there is $G(s) \gg 1$ and when $s \gg 1$, there is $G(s) \ll 1$. Eq. (15) defines the transfer function for the reference model. We use this reference model for two main reasons. Firstly, as long as $\alpha \ll 1$, the transfer function meets the requirement mentioned above. Secondly, the $G(s)$ defined in Eq. (15) is similar to the thermal model mentioned in Section II, which makes it easy for the controller to trace the ideal output of the reference model.

$$\frac{T(s)}{U(s)} = G(s) = \frac{1}{s+\alpha} \tag{15}$$

The transfer function shows the frequency response of the system. Implementing the reference model based on the transfer function is not straightforward. We convert the transfer function into a differential equation to faciliate the implementation of the reference model. Eq. (16) shows the differential equation.

$$T = U' + \alpha \cdot U \tag{16}$$

The fuzzy inverse model decides the amount of the output of the fuzzy controller that should be adjusted based on ΔY. For the sake of system complexity, we also set three values



(a) Input (ΔY) (b) Output (ΔF)

Fig. 14: The linguistic variables and the related membership functions for the fuzzy inverse model.

TABLE II: Fuzzy logic rules of fuzzy inverse model.

If ΔY is	**Then** ΔF is
Positive	Negative
Zero	Zero
Negative	Positive

for each linguistic variable in the fuzzy inverse model. The linguistic variables and the related membership functions are defined in Fig. 14. As mention in Section IV-A, we also use Monte Carlo method to find the best parameters for ΔY^* and ΔF^*. As we all know, when $\Delta Y > 0$, F should be reduced in order to reduce the temperature of the system to reduce ΔY, and vice versa. Therefore, the fuzzy logic rules for the fuzzy inverse model can be easily designed. The complete set of fuzzy logic rules for the fuzzy inverse model is shown in Table II. The defuzzification module of the fuzzy inverse model also adopts the COG method.

The adapter changes the parameters in the fuzzy controller to apply the ΔF decided by the fuzzy inverse model. For the sake of efficiency, the adapter should not directly add ΔF to the output of the fuzzy controller. If ΔF is directly applied to the output of the fuzzy controller, in the next cycle ΔY becomes smaller, and so does ΔF. But since the output F of the fuzzy controller is unchanged, it causes ΔY to rise again in the cycle that follows. In this way, the adaptive module always amends the output of the fuzzy controller. However, if ΔY is realized by changing the parameters in the fuzzy controller, after the system becomes stable, the fuzzy controller does not require further adjustment. In such case, the system could shutdown the adaptive module and save considerable amount of time and energy. For the adapter to apply the change, the easiest way is to horizontally shift the membership functions of the linguistic variables of the output F. Fig. 15 shows the process of shifting membership functions of F activated by the adapter.

The complexity of the fuzzy controller is $O(n)$, where n denotes the number of fuzzy logic rules. Since the number of fuzzy logic rules are usually fixed, the complexity of the fuzzy controller can be viewed as $O(1)$. The complexity of the self-adaptive module is also $O(1)$. Although the reference model contains a differential equation, with efficient discrete algorithms like Runge-Kutta methods, the computing complexity is negligible.

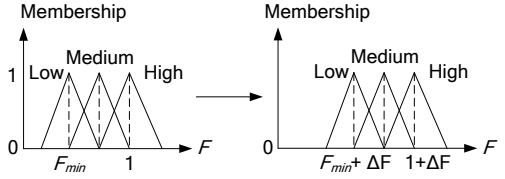


Fig. 15: Membership function adjusted by the adapter.

Algorithm 1 Coolest first scheduling algorithm

Input: ready tasks, the cores in the processor
Output: the schedule
if a time slack terminates **then**
  sort the cores according to current temperature.
  sort ready tasks according to power consumption.
  repeat
    pick the task with highest power
    schedule the task to the coolest core
  until all the cores are occupied
end if

V. Extension to Multi-core Processors

The previous section discusses the design of the adaptive fuzzy controller in a unicore processor. When used in multi-core processors, the advantages of adaptive fuzzy controller become more significant. Firstly, in a multi-core processor, the thermal model of each core is coupled with each other due to the inter-chip heat transfer. It makes the thermal model of a multi-core processor difficult to build compared to the single-core processor case. In addition, in multi-core processors with the asymmetrical floorplan, the thermal model of each core depends on its location in the die. Only controllers with adaptive abilities to deal with the situation. Secondly, the in-die process variation worsens the situation for building an accurate thermal model for the multi-core processor. It further enlarges the difference in the thermal model of each individual core.

The increase in the number of cores in the system brings higher flexibility in thermal control. For multi-core processors, thermal-aware scheduling is an efficient technique to reduce the peak temperature of the system [18]. Since our fuzzy controller is a core-throttling thermal management technique, it should be combined with an efficient thermal-aware scheduling algorithm to achieve higher performance. When combined with a thermal-aware scheduling algorithm, the potential possibility of thermal emergencies significantly reduces, which leads to less switches of the voltage and frequency level of the system.

An efficient thermal-aware scheduling algorithm should balance the power consumption on the processor both spatially and temporally. In this study, the thermal-aware scheduling algorithm we selected to work cooperatively with the adaptive fuzzy controller is the coolest first algorithm [19]. The pseudo code of the algorithm is shown in Alg. 1. The algorithm aims at scheduling the task with highest power consumption to the processor with coolest temperature. The effectiveness of the algorithm in reducing the number of thermal emergencies has been proved in the previous study [19]. We note that the power consumption of the tasks could either be acquired off-line through profiling or be measured on-line using performance counters.

VI. Evaluation

A. Experiment Setup

In the experiments, we use HotSpot 5.02 as our thermal simulation platform for the microprocessor [20]. The experiments are carried out on both single-core and multi-core processors. The single-core processor is a 45nm ARM11 processor. The multi-core processor is a 45nm ARM Cortex-A7 processor with four cores. The DVFS module is assumed to be able to adjust the frequency from 0.9GHz to 1.5GHz. We use two benchmarks in the experiments. The first benchmark is a endless loop of floating point computation, which is used to test the step response of the closed-loop control system because the power consumption of is stable. The second benchmark is the SPEC CPU2006 suite, which is used to test the performance of system real-world applications. We use the GEM5 + MCPAT simulation platform to collect the power trace of the benchmarks [21], [22] .

In the experiments, three different kinds of thermal controllers are adopted for comparision. First, we use a simple threshold-trigger DVFS controller as the baseline of the experiments. Second, we adopt a PID controller which is proposed in [23]. The PID controller is the most commonly adopted controllers in control systems. Finally, we adopt a state-of-the-art thermal controller design from [9]. The controller is designed using the optimal control theory, therefore it refered to as the optimal controller. For the three types of controllers, the duration of one control cycle is set to 10 ms. The temperature traces of the processors without any thermal control methods are aslo shown in the experiments.

In the experiments, we set the variation of the system for three different sources. The first source of variation is the noises of the temperature sensors. The noise singals contained by the output of the temperature sensor is set to be a white noise with a amplitude of $0.5°C$. This setting is based on the senser noise model proposed in [10]. The second source of variation is the ambient temperature. We set the different ambient temperature levels in different experiments. The third source of variation is the process variation of the processors. As discussed in Section II-C, with process variation, the leakage power can be computed by δP_{sub}, where P_{sub} is the subthreshold leakage power of the processor without process variation and δ is a factor defined by the process variation. For the single core processor, we set $\delta = 1.5$, which is within the reasonal range according to [16]. For the multi-core processor, we set δ for the four cores using a normal distribution with mean of 1 and standard deviation of 0.3. This is also based on the discussion in [16].

B. Step Response

In control theories, step response is a commonly used metric to examine the control qualities of control systems. The step response of a control system is defined as the output of the system under the input of a step function. Since the input of the system is very simple, the step response clearly shows the native characterisitics of the control system. In the experiments, the single core processor is tested. The benchmark is the first benchmark mentioned in Section VI-A which produces a stable power consumption. The initial temperture of the processor is set to be $60°C$. The threshold temperature of the processor is set to $75°C$. The ambient temperature of the system is set to $25°C$.

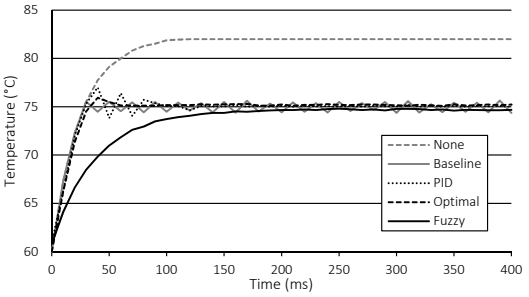


Fig. 16: Step responses in ideal situation.

In the first experiment, we tested the performance of the control system without the variation of the thermal model. 16 shows the step responses of different control systems under the ideal situation. We can see that without thermal control, the processor temperature reaches a stable level of around $82°$. Among the four controllers, the performance of the baseline controller is the worst. The temperature of the processor fluctuates periodically around the threshold temprature. This is because the baseline controller considers too little information to decide the control output. It only reacts when the temperature is above the threshold. The PID controller, which is designed according to the system model, achieves much better result. However, when compared to the optimal controller and the adaptive fuzzy controller, the output of the PID controller shows higher overflow above the threshold and more severe fluctuation. This is because the structure of the PID controller is too simple and limits its ability to adjust the system characteristics. With more advanced structures, the optimal controller and our adaptive fuzzy controller both achieves good performance in temperature control. But when the temperature is under the threshold, the temperature controlled by the optimal controller rises faster than our fuzzy controller. This means with the optimal controller, the processor runs faster than with the fuzzy controller, which further results in better performances. According to [9], the optimal controller is able to optimize the performance of the processor with a temperature constraint. The experimental results also supports this point.

In the second experiment, we test the tolerance of the control systems to the sensor error. Fig. 17 shows the temperature traces of the processors with different controllers after temperature becomes relatively stable. It is obvious that the baseline methods is seriously affected by the sensor error. This is because the control decision of the baseline controller solely relies on the output of the temperature sensor and there is no method to identify the noise signals from the output. For the other three controllers, the PID controller shows highest sensitivity to the noises. This is also due to the simplicity in the structure of the PID controller. The optimal controller and the fuzzy controller both show high tolerance to the sensor noises.

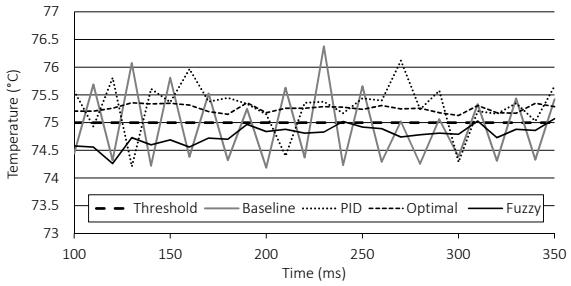


Fig. 17: Step responses with temperature sensor error.

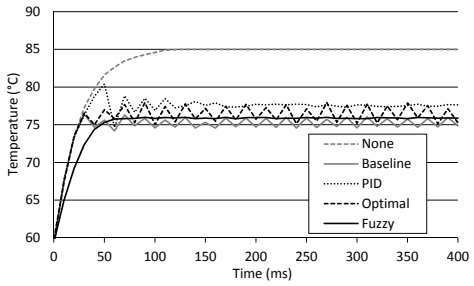


Fig. 18: Step responses in parameter variation.

Finally, we test the performance of the controllers under process variation and ambient temperature change. First, we set the ambient temperature as $35°C$, which represents a outdoor environment in summer time. Second, we set the δ of the subthreshold leakage power model as 1.5. Fig. 18 shows the related temperature traces. Due to the rise of the ambient temperature and the leakage power, the processor temperature reaches a higher level than in the previous experiments. First we can see that the temperature trace of the baseline controller is not significantly affected. This is because the change of system characteristics does not affect the control policy of the baseline controller. The only difference is that with increased leakage power, the temperature rises faster under the high voltage level and decrease slower under the low voltage level. Totally, the baseline controller achieves a slightly higher average temperture than in the previous experiment. Second, the result of PID controller shows a steady error when compared to the threshold. According to Section II-D, the rise in leakage power makes the control system more sensitive to noises. Refering to Fig. 6, the ambient temperature acts as a noise of the input signal of the system. When the two factors are combined together, the output of the PID controller deviates from the pre-defined threshold for around $2.5°C$. Third, the result of optimal controller fluctuates severely. This is because the optimal controller is structured based on a more delicate system model [9] and the variation of the system model have a significant affect on the performance of the controller. Finally, the adaptive fuzzy controller achieves a nearly unchanged power trace compared to the previous experiment. The result shows that our design have a good adaptivity to the variation of the system model.

C. Real Applications

We test the performance of the single core ARM processor with different kinds of thermal controllers under real-world workloads using the SPEC CPU2006 benchmark suite. In the experiments, The initial temperture of the processor is set to be $60°C$. The threshold temperature of the processor is set to $75°C$. The ambient temperature of the system is set to $25°C$.

First we test the performance of the processor without the variations. Table III shows the execution time of the benchmarks in the processors using different kinds of thermal controllers. The performance of the processor without any thermal control methods is always the best since the benchmarks are executed with the highest voltage level. For different kinds of thermal controllers, the average increases ofn the execution time of the benchmarks are 27.5% (baseline), 19.5% (PID), 8.1% (optimal) and 11.7% (fuzzy) respectively. The baseline controller results in the worst performance due to its continuous switching of the voltage levels. The optimal controller achieves best performance among the thermal controllers due to its optimization due to its performance-aware optimization. The fuzzy controller outperforms the PID controller because the PID controller leads to a more fluctuating temperature trace which results in more switches of the voltage level.

TABLE III: Execution times of the benchmarks (without variations).

Bench.	Execution time (s)				
	None	Baseline	PID	Optimal	Fuzzy
400	1040.76	1307.41	1248.25	1085.14	1148.16
401	1353.56	1726.88	1552.42	1416.57	1540.00
403	905.22	1144.55	1059.74	959.1	1006.15
429	3386.09	4203.11	4030.73	3826.21	3765.57
445	260.29	310.87	309.79	274.03	293.09
456	997.97	1283.68	1157.42	1113.99	1153.32
458	9406.80	12948.49	11615.81	9992.4	10350.70
462	111.64	133.74	132.60	120.45	127.10
464	67426.00	84066.01	81638.28	78398.56	72916.57
471	1153.36	1591.83	1390.07	1262.79	1331.50
473	16433.83	21378.5	20314.06	17878.81	18050.46
483	276.77	356.62	334.71	287.86	301.07

Table IV shows the execution time of the benchmarks with variations. In the experiments, the three sources of variations are combined together. The execution time of the processor without thermal controllers keeps the same. We note that process variation also affects the execution time of the processor, but to focus our discussion on thermal control, we do not consider this influence in this study. For different kinds of thermal controllers, the average increases of the execution time of the benchmarks are 29.9% (baseline), 36.1% (PID), 28.5% (optimal) and 17.6% (fuzzy), respectively. When compared with the results in Table III, the baseline controller is the least affected by the variations. As discussed in Section VI-B, this is because the model change does not affect the decision making procedure of the baseline controller. The increase in the execution time mostly comes from the rising of the leakage power of the processor. The PID controller and the optimal controller are both severely affected by the variations of the thermal model. Due to their lack of adaptivity, these two controllers cannot perform as well as designed. Our fuzzy controller, on the other hand, is capable of adapting to the variations of the system model and achieve the best performance among the four kinds of c ontrollers.

As mentioned in Section I, the control quality of the thermal

TABLE IV: Execution times of the benchmarks (with variations).

Bench.	Execution time (s)				
	None	Baseline	PID	Optimal	Fuzzy
400	1040.76	1357.87	1485.72	1480.46	1227.49
401	1353.56	1737.82	2036.16	1766.68	1596.54
403	905.22	1146.14	1137.28	1165.38	1077.21
429	3386.09	4219.54	5077.18	4134.8	3837.48
445	260.29	315.8	328.639	336.823	295.12
456	997.97	1399.32	1257.29	1174.41	1188.32
458	9406.80	13165.7	14871.2	13749.8	11231.41
462	111.64	137.08	141.672	136.312	118.60
464	67426.00	84852.01	86681.9	81374.5	86110.65
471	1153.36	1608.89	1371.62	1539.46	1344.37
473	16433.83	21461.41	26156.7	22614.3	21465.20
483	276.77	357.5	335.663	355.728	305.24

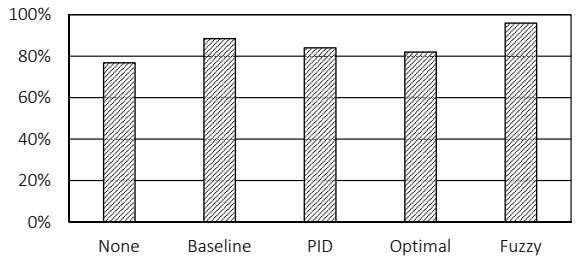


Fig. 19: Normalized lifespan of the processors.

controllers not only affects the performance of the system but also affects the reliability of th processors. High temperature and severe temperature fluctuation both pose negative effects on the lifespan of the processors. According to the reliability model proposed in [24], we estimate the lifespan of the processors with different thermal control solutions. In the estimation, the temperature traces collected from the experiments with process variations is used. Fig. 19 shows the lifespan of the processors with different thermal controllers. The lifespan is normalized to the ideal lifespan of the processor which is resulted from working under a stable temperature level under the threshold. From the figure we can see that our adaptive fuzzy controller achieve longest lifespan when compared to other kinds of controllers. This is because the temperature traces achieved by our adaptive fuzzy controller is more stable than the other controllers. With process variation, the control quality of the optimal controller is more severely damaged and therefore results in the shortest lifespan among the three kinds of controllers.

D. Multi-core Processors

In the end, we evaluates the performance of our adaptive fuzzy controller on multi-core processors. The processor model used in the simulation is the ARM Cortex 7 processor with four individual cores in one chip. We assume each core has an individual thermal control unit. In the simulation, we run all the benchmarks on the processor. The thermal-aware scheduling algorithm mentioned in Section V is used to schedule the tasks on the processor. The three sources of the variation are all applied to the processor.

Fig. 20 shows the execution time of the benchmarks on the processors with different types of thermal controllers. Similar

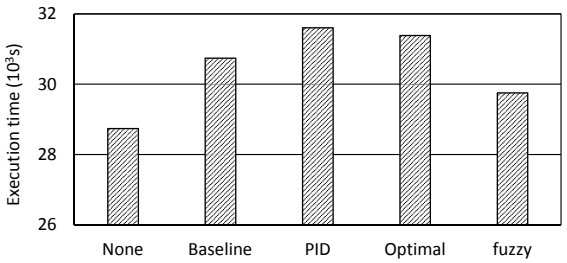


Fig. 20: Execution time of the benchmarks (ARM Cortex 7).

like the experimental results on the single-core processor. Our adaptive fuzzy controller outperforms the PID controller and the Optimal controller in execution time by up to 6.18%. When comparing to the result of the single core processor, the reduction in execution time achieved by our fuzzy controller has decreased. This is because of the following two reasons. Firstly, the area of the processor has increased due to the integration of multiple cores, which makes the heat easier to dissipate. Secondly, the use of the thermal-aware scheduling algorithms efficiently improves the thermal condition of the cores. The two reasons combined results in a more balanced power consumption on the processor and therefore reduces the room of temperature management for the thermal controllers. Therefore the advantages of our adaptive fuzzy controller becomes less obvious on the multi-core processor.

VII. Related Work

Previous works mainly adopted DVFS based techniques and/or task scheduling to manage the processor temperature of CMP. DVFS techniques control the temperature of microprocessors through dynamically adjusting the voltage and frequency levels [8], [6], [5], [25]. Most DVFS-based DTM systems [6], [5], [25] can be categorized as open-loop control systems, because in such works, the decision making processes are usually triggered by thermal emergencies. Open-loop control systems have three major disadvantages. First, the respond speed to thermal emergencies is slow. As a result, the processors will work under high temperature for longer time, which harms the reliability and lifespan of the microprocessors. Second, the temperatures of processors with open-loop thermal control systems fluctuate severely, which also could cause malfunction of processors [4].

To achieve higher control qualities, closed-loop DVFS-based thermal control systems have been proposed. Based on control theory, closed-loop thermal control systems provide higher temperature accuracy, fast response time and significant reduction to temperature fluctuations. In [8], a PID controller was adopted to control the DVFS module. As one of the most classic designs in control theory, PID controllers have low complexity and are feasible for most kinds of systems. Fu et al. [23] proposed an improved design of PID controller. Wang et al. adopted optimal control theory and proposed a closed-loop optimal controller targeting thermal and power control [9]. The optimal controller provides theoretical optimal control quality in a multi-objective optimization problem, however with high complexity. In [11], Sabry et al. proposed fuzzy controller for

thermal control with microfluidic cooling. Fuzzy controllers have the advantages of good control qualities, low complexity, and high endurance to errors and system parameter variations.

Thermal-aware task scheduling is another widely used approach to address the thermal challenges in microprocessors [26], [18], [27]. The fundamental idea is to assign tasks to the cores in such a manner that overheating of cores can be avoided. For instance, Hung et al. [26] proposed several fast heuristics to minimize the run-time peak temperature in multi-core systems. Coskun et al. [18] proposed an integer linear programming (ILP) based algorithm to reduce the number of thermal hotspots on a chip through workload distribution. However, both thermal-aware DVFS and task scheduling based approaches have their own limitations. Pure DVFS-based thermal management may suffer from performance degradation and significant time and energy overheads under high workload. On the other hand, the temperature control quality of pure scheduling algorithms is less effective compared to the DVFS-based solutions.

Thus, hybrid solutions are proposed in order to take the advantage of both techniques [28], [29], [6], [30], [31], [32]. For instance, Liu et al. [28] used clustering algorithm to solve the scheduling and voltage assignment problem for task graphs. Bao et al. [29] used a heuristic to distribute idle slacks among periodic tasks to reduce the peak temperature. Hanumaiah et al. [6] formulated the thermal management problem of multi-core processors into a non-linear optimization problem and proposed optimal task scheduling and voltage scaling policies. Kumar et al. [30] first adopted the hybrid methodology in dynamic thermal management. The proactive task and voltage assignment was according to a fast regression-based thermal model. Rao et al. [31] proposed a new processor speed model and optimized the processors performance based on the model in a thermal-aware system with DVFS. Ma et el. [32] proposed an application assignment algorithm to optimize the performance of multi-core processors with the optimal thermal controller.

However, due to aggressive scaling of technology, the increasing process variations have posed serious challenges to the traditional thermal management solutions. Process variations coupled with environmental variations like ambient temperature fluctuations, increasing noise in sensor temperature readings, etc. result in highly unpredictable system, forcing pessimistic design decisions with larger power and thermal margins compromising performance and energy efficiency [33], [34], [35]. As a result, the above discussed thermal-aware scheduling and DVFS techniques often fail to control rising temperature since despite the existing variations, these techniques treat all processing cores as the same with nominal values [34]. Hence, the impact of process variations on the power, performance and temperature of multi-core processors needs to be carefully investigated. Though, the studies on process variations can be dated back to the 1970s [34], serious consideration to its challenges began only when the device scaling reached 90nm and beyond as the manufacturing processes lacks controllability with miniaturizing of device sizes [33], [34]. Moreover, with more transistors integrated on a single chip, the chip is susceptible to larger variations due to added environmental effects like ambient temperature, signal noises due to proximity effects etc. [16]. Process variations make a deterministic system into statistical or probabilistic in nature, which requires statistical models to describe how transistor parameters vary within a die.

On the thermal aspects, Kursun et al. [36] first studied the influence of process variations on temperature profile of a multi-core processor, and proposed a sensor network which identifies the difference in the power consumption of each processor. Finally, they proposed a variation-aware thermal scheduling algorithm which takes the difference in power consumption of each core into consideration. Most recently, Tavana et al. combined DVFS and task mapping to achieve energy efficiency using simulated annealing solution for reducing energy delay product. Paterna et al. [37] analyzed the influence of ambient temperature variation on the thermal management solutions for embedded processors in smartphones. These approaches all require the measurement or modeling of process variation at design time, which significantly limits the flexibility and incurs extra complexity and cost.

VIII. Conclusion

In this paper, we propose an adaptive fuzzy controller for DVFS-based dynamic thermal management of microprocessors. Based on the fuzzy logic and the adaptive control theory, the controller has the benefits of high control quality, low sensitivity to system noises and the variations of system model. Our adaptive fuzzy controller achieves up to 18.5% better performance and up to 14.2% longer lifespan of the microprocessors when compared with other state-of-the-art thermal controllers.

References

[1] A. B. Kahng, "The itrs design technology and system drivers roadmap: Process and status," in *Proceedings of the 50th Annual Design Automation Conference*, ser. DAC '13, 2013, pp. 34:1–34:6.

[2] B. Li, L.-S. Peh, and P. Patra, "Impact of process and temperature variations on network-on-chip design exploration." in *NOCS*. IEEE Computer Society, 2008, pp. 117–126.

[3] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "The impact of technology scaling on lifetime reliability," in *Dependable Systems and Networks, 2004 International Conference on*. IEEE, 2004, pp. 177–186.

[4] J. S. S. T. Association *et al.*, "Failure mechanisms and models for semiconductor devices," *JEDEC Publication JEP122-B*, 2003.

[5] V. Hanumaiah and S. B. K. Vrudhula, "Temperature-aware dvfs for hard real-time applications on multicore processors." *IEEE Trans. Computers*, vol. 61, no. 10, pp. 1484–1494, 2012.

[6] V. Hanumaiah, S. B. K. Vrudhula, and K. S. Chatha, "Performance optimal online dvfs and task migration techniques for thermally constrained multi-core processors." *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 30, no. 11, pp. 1677–1690, 2011.

[7] J. Lee and N. S. Kim, "Analyzing potential throughput improvement of power- and thermal-constrained multicore processors by exploiting dvfs and pcpg." *IEEE Trans. VLSI Syst.*, vol. 20, no. 2, pp. 225–235, 2012.

[8] K. Skadron, T. F. Abdelzaher, and M. R. Stan, "Control-theoretic techniques and thermal-rc modeling for accurate and localized dynamic thermal management," in *Proceedings of the Eighth International Symposium on High-Performance Computer Architecture (HPCA'02), Boston, Massachusettes, USA, February 2-6, 2002*, 2002, pp. 17–28.

[9] Y. Wang, K. Ma, and X. Wang, "Temperature-constrained power control for chip multiprocessors with online model estimation." in *ISCA*. ACM, 2009, pp. 314–324.

[10] A. Bakker and J. H. Huijsing, *High-accuracy CMOS smart temperature sensors*. Springer, 2000, vol. 595.

[11] M. M. Sabry, A. K. Coskun, and D. Atienza, "Fuzzy control for enforcing energy efficiency in high-performance 3d systems." in *ICCAD*. IEEE, 2010, pp. 642–648.

[12] N. H. Weste and D. M. Harris, *Integrated circuit design*. Pearson, 2011.

[13] A. P. Chandrakasan, W. J. Bowhill, and F. Fox, *Design of high-performance microprocessor circuits*. Wiley-IEEE press, 2000.

[14] A. Srivastava, R. Bai, D. Blaauw, and D. Sylvester, "Modeling and analysis of leakage power considering within-die process variations," in *Proceedings of the 2002 international symposium on Low power electronics and design*. ACM, 2002, pp. 64–67.

[15] H. Su, F. Liu, A. Devgan, E. Acar, and S. Nassif, "Full chip leakage estimation considering power supply and temperature variations," in *Proceedings of the 2003 international symposium on Low power electronics and design*. ACM, 2003, pp. 78–83.

[16] K. Agarwal and S. Nassif, "Characterizing process variation in nanometer cmos," in *Design Automation Conference, 2007. DAC'07. 44th ACM/IEEE*. IEEE, 2007, pp. 396–399.

[17] L. R. M. M. K. A. Landau, I.D., *Apative Control*. Springer-Verlag, 2011.

[18] A. K. Coskun, T. S. Rosing, K. A. Whisnant, and K. C. Gross, "Temperature-aware mpsoc scheduling for reducing hot spots and gradients," in *Proceedings of the 2008 Asia and South Pacific Design Automation Conference*. IEEE Computer Society Press, 2008, pp. 49–54.

[19] K. Stavrou and P. Trancoso, "Thermal-aware scheduling: a solution for future chip multiprocessors thermal problems," in *Digital System Design: Architectures, Methods and Tools, 2006. DSD 2006. 9th EUROMICRO Conference on*. IEEE, 2006, pp. 123–126.

[20] K. Skadron and et al., "Temperature-aware microarchitecture," in *ISCA*, 2003, pp. 2–13.

[21] e. e. Binkert, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[22] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures." in *MICRO*. ACM, 2009, pp. 469–480.

[23] Y. Fu, N. Kottenstette, C. Lu, and X. D. Koutsoukos, "Feedback thermal control of real-time systems on multicore processors." in *EMSOFT*. ACM, 2012, pp. 113–122.

[24] O. Semenov, A. Vassighi, and M. Sachdev, "Impact of self-heating effect on long-term reliability and performance degradation in cmos circuits," *Device and Materials Reliability, IEEE Transactions on*, vol. 6, no. 1, pp. 17–27, 2006.

[25] H. Jung and M. Pedram, "Stochastic dynamic thermal management: A markovian decision-based approach," in *Computer Design, 2006. ICCD 2006. International Conference on*. IEEE, 2007, pp. 452–457.

[26] W.-L. Hung, Y. Xie, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "Thermal-aware task allocation and scheduling for embedded systems," in *Proceedings of the conference on Design, Automation and Test in Europe-Volume 2*. IEEE Computer Society, 2005, pp. 898–899.

[27] A. K. Coskun, T. S. Rosing, and K. Whisnant, "Temperature aware task scheduling in mpsocs," in *Proceedings of the conference on Design, automation and test in Europe*. EDA Consortium, 2007, pp. 1659–1664.

[28] Y. Liu, Y. Yang, and J. Hu, "Clustering-based simultaneous task and voltage scheduling for noc systems," in *Proceedings of the International Conference on Computer-Aided Design*. IEEE Press, 2010, pp. 277–283.

[29] M. Bao, A. Andrei, P. Eles, and Z. Peng, "Temperature-aware idle time distribution for energy optimization with dynamic voltage scaling," in *Proceedings of the Conference on Design, Automation and Test in Europe*. European Design and Automation Association, 2010, pp. 21–26.

[30] A. Kumar, L. Shang, L.-S. Peh, and N. K. Jha, "Hybdtm: a coordinated hardware-software approach for dynamic thermal management," in *Proceedings of the 43rd annual Design Automation Conference*. ACM, 2006, pp. 548–553.

[31] R. Rao and S. Vrudhula, "Efficient online computation of core speeds to maximize the throughput of thermally constrained multi-core processors," in *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, 2008, pp. 537–542.

[32] K. Ma, X. Li, M. Chen, and X. Wang, "Scalable power control for many-core architectures running multi-threaded applications," in *ACM SIGARCH Computer Architecture News*, vol. 39, no. 3. ACM, 2011, pp. 449–460.

[33] S. Herbert and D. Marculescu, "Variation-aware dynamic voltage/frequency scaling," in *High Performance Computer Architecture, 2009. HPCA 2009. IEEE 15th International Symposium on*. IEEE, 2009, pp. 301–312.

[34] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variations and impact on circuits and microarchitecture," in *Proceedings of the 40th annual Design Automation Conference*. ACM, 2003, pp. 338–342.

[35] W. Schemmert and G. Zimmer, "Threshold-voltage sensitivity of ion-implanted mos transistors due to process variations," *Electronics letters*, vol. 10, no. 9, pp. 151–152, 1974.

[36] E. Kursun and C.-Y. Cher, "Temperature variation characterization and thermal management of multicore architectures," *IEEE micro*, vol. 29, no. 1, pp. 0116–126, 2009.

[37] F. Paterna, J. Zanotelli, and T. S. Rosing, "Ambient variation-tolerant and inter components aware thermal management for mobile system on chips," in *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*. IEEE, 2014, pp. 1–6.

Yingnan Cui Yingnan Cui received his Bachelor degree in Automation from Harbin Institute of Technology (2006-2010). He is now a Ph.D. student in School of Computer Engineering, Nanyang Technological University, Singapore (2010-). His research topic focuses on run-time thermal management of multi-core systems.

Wei Zhang Dr. Wei Zhang received her Ph.D. degree in Electrical Engineering from Princeton University. She joins Hong Kong University of Science and Technology in 2013 and establishes Reconfigurable System Lab. She was an assistant professor in School of Computer Engineering at Nanyang Technological University, Singapore (2010-2013). She is a co-investigator of Singapore-MIT Alliance for Research and Technology and works on low-power electronics. She is a collaborator of ASTAR-UIUC Advanced Digital Sciences Center and works on FPGA acceleration for multimedia applications.

Bingsheng He Dr. Bingsheng He received the bachelor degree in computer sicence from Shanghai Jiao Tong University (1999-2003), and the Ph.D. degree in computer science in Hong Kong University of Science and Technology (2003-2008). Dr. He is an assistant professor in School of Computer Engineering of Nanyang Technological University, Singapore. His research interests are high performance computing, cloud computing, and database systems.