A Neuro-Fuzzy Fan Speed Controller for Dynamic Thermal management of Multi-core Processors

ABSTRACT

Request for more computing power steadily forces designers to provide more powerful processors by using the number of cores on a single chip. The increasing complexity of processors leads to higher integration density, power density and temperature. Cooling equipment is one Thermal management technique that reduces the thermal resistance of the heat sink without any performance degradation. However, higher fan speed produces a lower thermal resistance, but at the expense of higher power consumption. The increase in fan speed causes an increase of power consumption, large noise levels in the system, and decrease of fan lifetime that may impact reliability. Our proposed Neuro-fuzzy (NF) fan controller (NFSC), minimizes fan power consumption while avoiding the temperature violation from the temperature threshold. NFSC estimates the minimum fan speed that holds the temperature close to the temperature threshold with the aim of saving power. The experimental results indicate that our proposed model can significantly decrease the average fan power consumption about 30% with increase of average temperature by 5% compared to the traditional fan controller.

Categories and Subject Descriptors

energy efficiency, Reliability

General Terms

Experimentation, power, Temperature, Performance, Active Cooling

Keywords

Power Management, Close-loop Control, Dynamic Thermal Management, Temperature, Nero-fuzzy

1. INTRODUCTION

The high demand for more computing power grows steadily which leads to increasing integration density of modern processors. The high power and energy consumed in a small area die size results in rising power density and generated heat. Moreover, higher temperature potentially threatens system reliability and decreases both transistor age and transition speed [1]. Dynamic Thermal Management (DTM) techniques are proposed to mitigate the aforementioned problems. DTM is a set of techniques that control processor temperature at run-time in order to prevent temperature from violating the system temperature threshold. DTM techniques can be categorized into in - band and out-of-bound approaches. In-bound thermal management techniques operate in the critical path of the application execution. Stop-and-go is one of the primitive hardware-based in-bound DTM techniques, where a processor core enters into a sleep state at temperature threshold and resumes execution once the temperature returns to a normal operating range. Dynamic Voltage and Frequency Scaling (DVFS) is another hardware-based in-bound DTM technique that dynamically adjusts the processor voltage and frequency to manage power and temperature. Task scheduling [2-3] and task migration [4-8] are two software-based in-bound methods. The aim of the task scheduling technique is to distribute tasks among different cores to prevent hot spots [2-3]. Task migration is another technique that moves tasks from a hot core to an appropriate core in order to control and manage the overall temperature [4-8]. Although all of these approaches decrease peak and average temperature greatly, they lengthen execution time, thus degrading overall system performance [1].

On the other hand, out-of-band techniques (e.g. CPU cooling fans) operate completely outside the critical performance path of applications. Out-of-band techniques, which use processor cooling equipment, reduce the thermal resistance of the heat sink without any performance degradation by the use of active devices such as fans. The thermal resistance of a heat sink is controllable and inversely proportional to the rotational fan speed. Higher fan speed produces a lower thermal resistance at the expense of higher power consumption. The considerable increase in cooling power due to the cubic relationship between fan speed and its power is one the most serious challenges [9]. Furthermore, the increase in fan speed causes large noise levels in the system that may impact reliability. Lowering fan speed has the additional advantage of improving the fan lifetime. The results in [10] show that a 35% reduction in fan speed improves the fan lifetime by a factor of two.

Despite the effects of fan speed on power consumption, noise level, reliability, etc., intelligent fan control is not performed on the fan speed. The traditional fan controllers on the system are reactive and use a static mapping between the fan speed and the processor temperature [11]. This approach increases the air flow rate to match to a corresponding rise in temperature.

Recently, some intelligent fan control techniques have been proposed in order to optimize system parameters such as performance, power and energy under thermal constrained. As an example, a unified framework consisting of both in-band and out-of-band techniques has been proposed in [12] to determine an optimal configuration to maximize performance/watt of multi-core processor. In [11] integration of an out-of band method (fan control) and an in-band method (DVFS) has been performed to balance temperature, power consumption, and performance. They found that using a less powerful fan can achieve the same thermal efficiency as a more powerful fan if we have more intelligent fan controllers. A DTM framework has been presented in [13] to minimize the total system power consumption, while avoiding the thermal emergency. The proposed optimization framework obtains the energy-optimal die temperature with considering throughput constraint, while conventional DTM techniques hold the die temperature as close as possible to the thermal emergency temperature to get the maximum throughput. The authors of [14] proposed a new workload scheduling technique that minimizes the cooling energy costs of fan subsystem in a multi-socket CPU platform. At the core level, a new band limited predictor was employed for proactive thermal management that minimizes the cooling costs and reduces the frequency of thermal emergencies. The work of [15] is a simulation-based method that examines the unified workload, power and cooling management for saving energy and capping power of a blade enclosure system. They proposed a hierarchical control for data center level energy management. Their Fan Controller (FC) level of their proposed architecture for a blade enclosure tunes the fan speeds dynamically to maintain server temperature below their threshold while minimizing power.

Among the presented techniques of cooling management, are theoretical and derived based on simplified mathematical thermal models and idealized assumptions. Using thermal models causes increasing computational intensity and runtime overhead that makes it difficult to use them on the off-the-shelf processors [16]. Previous fan controllers try to keep the temperature lower than the temperature threshold; however, it could reduce fan speed to keep the temperature around temperature threshold, resulting in reduce of power consumption because of the cubic relationship between fan speed and its power.

Motivated by these facts, this paper uses a Neuro-fuzzy (NF) based fan controller to minimize power consumed by the fan and at the same time avoid violating from the temperature threshold. Our proposed method estimates the minimum fan speed that holds the temperature close to the temperature threshold with the aim of saving power. To authors' best knowledge, no prior attempts have been made to minimize fan power for saving system power on commercial quad-core desktop products under Linux environment.

The experimental results on Intel's Core i7-950 running two to seven benchmarks indicate that our proposed method, NFSC (Neuro-Fuzzy Fan Speed Controller), outperforms static mapping controller in reducing fan power without a substantial increase in temperature. The main contributions of this paper are summarized as follows:

• This paper uses a neuro-fuzzy based model to minimize fan power consumption by controlling fan speed to keep the temperature around temperature threshold.

• Our experimental results on commercial processors indicate that our proposed approach outperforms the static mapping fan controller in saving fan power, considering the temperature threshold.

• There is no additional hardware unit required for our proposed model and thermal-aware algorithm.

The remainder of this paper is organized as follows: The preliminaries of our algorithm are presented in Section 2. Section 3 describes our proposed algorithm. Implementation and analysis results are shown in Section 4 and final conclusions are drawn in Section 5.

2. Preliminary

In this section, the preliminary of proposed algorithm is presented.

2.1 Problem description

In this paper a multi-core processor with N cores, denoted {core1, core2,…,coreN}, that each core has a dedicated physical thermal sensor, has been used. It has a forced-convection air-cooled heat sink that create by a low thermal resistance material and a cooling fan that circulates ambient air through the heat sink. CPU thermal sensors provide feedback signal for closed loop controllers to control fan speed [17].

The problem discussed in this paper is how to control fan speed dynamically, while minimizing fan power and temperature does not violate temperature threshold. In this direction, the proposed method is introduced, which determines the appropriate future fan speed by considering the current fan speed, current temperature and temperature threshold of the processor.

2.2 The Fan speed predictor

In this section, we present our fan speed model based on NF models to determine the next speed of fan processor. NF inference systems have some advantage that motivates us to use them for a modeling fan for thermal management. These systems are relatively fast and robust in its essence. Proposed fan model has three input parameters and one output parameter. In order to learn our model, we need to produce a dataset consist of the effects of changing fan speed on various temperature. Since producing a complete dataset is time-consuming and sometimes difficult, we only produce some portion of complete dataset. NF is applicable for models with a small number of input parameters and useful for small data sets. Since NF models have smooth outputs, they are suitable for fan speed controllers to prevent fluctuating fan speed which decreases fan lifetime.

The Locally Linear Neuro-Fuzzy Models (LLNFM) falls into Takagi-Sugeno (TS) [18] method, where its fuzzy

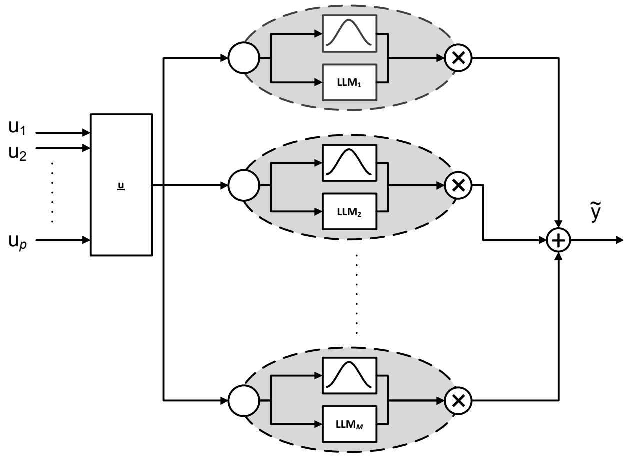


Figure 1-LLNFM structure for P inputs and M neurons.

consequents are a function of the inputs. Figure 1 shows the network structure of LLNFM. It is assumed that M is the number of neurons, $\underline{u} = [u_1 \quad u_2 \quad \cdots \quad u_p]$ is input vector, and P is the number of inputs. As depicted in Figure 1, the network structure consists of one hidden layer and an adder in the output layer, which simply calculates the summation of Locally Linear Models (LLMs) [19].

The locally linear modeling approach is based on Divide-and-Conquer (D&C) strategy. A complex nonlinear system space input is divided into small sub-problems with fuzzy validity functions, until these sub-spaces can be covered directly. This network model consists of non-linear parameters, which include mean and deviation of the Gaussian used in validity functions, and linear parameters. According to Fig.1 and the basis of the local linear modeling approach, following equations can be extracted for each fuzzy neuron [19]:

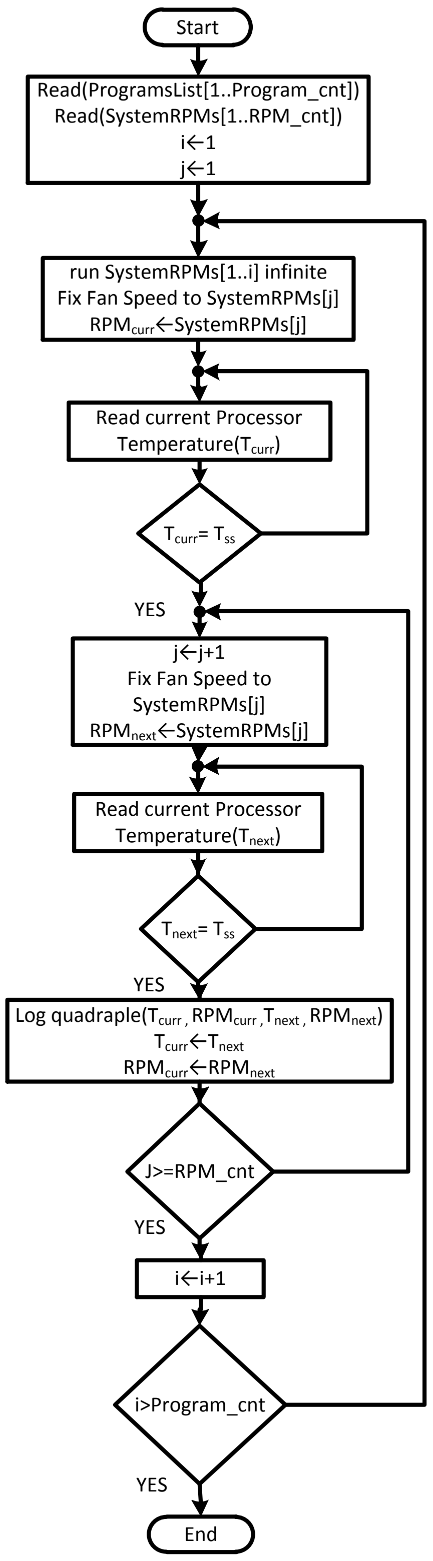
$$\mu_i(\underline{u}) = \exp\left(\frac{-1}{2}\left(\frac{(u_1 - c_{i1})^2}{\sigma_{i1}^2} + \cdots + \frac{(u_p - c_{ip})^2}{\sigma_{ip}^2}\right)\right) \tag{1}$$

$$\varphi_i = \frac{\mu_i(\underline{u})}{\sum_{i=1}^{M} \mu_i(\underline{u})} \tag{2}$$

$$\tilde{y}_i = \omega_{i0} + \omega_{i1}u_{i1} + \omega_{i2}u_{i2} + \cdots + \omega_{ip}u_{ip} \tag{3}$$

$$\tilde{y} = \sum_{i=1}^{M} \varphi_i \tilde{y}_i \tag{4}$$

Where i is the index of the neuron. Equation (1) computes the non-linear parts and it is normalized by Eq. (2). Eq. (3) calculates the linear parts and finally the overall model's output is obtained by applying Eq. (4). The centroid $c_{ij}$and standard deviationσ_{ij} are M×P parameters of nonlinear hidden layer, which are defined as variables of Gaussian activation functions. The learning process is delineated as adjusting the weights of LLM$\omega_{ij}$and premise parameters of activation functions, c_{ij} and σ_{ij}.



Start
Read(ProgramsList[1..Program_cnt])
Read(SystemRPMs[1..RPM_cnt])
i←1
j←1
run SystemRPMs[1..i] infinite
Fix Fan Speed to SystemRPMs[j]
RPMcurr←SystemRPMs[j]
Read current Processor Temperature(Tcurr)
Tcurr= Tss
YES
j←j+1
Fix Fan Speed to SystemRPMs[j]
RPMnext←SystemRPMs[j]
Read current Processor Temperature(Tnext)
Tnext= Tss
YES
Log quadraple(Tcurr, RPMcurr, Tnext, RPMnext)
Tcurr←Tnext
RPMcurr←RPMnext
J>=RPM_cnt
YES
i←i+1
i>Program_cnt
YES
End

Figure 2- The flowchart of the offline phase of our proposed method.

In this work, Locally Linear Modeling Tree (LoLiMoT) algorithm is used for the network learning process. LoLiMoT, which is an incremental tree-construction algorithm, uses axis-orthogonal splits to partition the input space. In each algorithm iteration, the validity functions that correspond to actual partitioning of the input space are computed and after that the corresponding rule consequences are optimized by a local weighted least squared technique.

3. Fan speed controller

As mentioned before, previous fan controllers try to keep the temperature below the temperature threshold; however, it could reduce fan speed to keep the temperature around temperature threshold, resulting in reduce of power consumption because of the cubic relationship between fan speed and its power. This fact persuaded us to propose a fan speed controller.

Our proposed method consists of offline and online phases. In the offline phase, we evaluate the effect of different fan speeds on various temperatures for making dataset and the LLNFM is learned by importing produced dataset. During the online phase, fan speed selection is done by using the extracted architecture and parameters of the previous phase.

3.1 Offline phase

This subsection discusses the offline phase of our proposed model. The flowchart of offline phase is depicted in Figure 2. In this phase, at first dataset is generated, and then LLNFM is trained by the obtained dataset. Creating dataset required several steps which we fully explain how it being generated. In the first step, the list of used programs in generating dataset is specified. After that, the list of RPMs is created by running appropriate procedure of the fan control toolbox. In the next step, fan speed set to the first value of RPMs list. Then, some programs are selected to run infinitely. The temperature increases until reaching the steady state temperature (T_{SS}), which is defined as a temperature that the system would reach if the application is executed infinitely [7]. In this state, T_{SS} and fan speed save as current temperature and current RPM consequently. After that, fan speed set to next fan speed in the RPMs list that led to changing temperature to another T_{SS}. Selected fan speed and obtained T_{SS} set as a next RPM and next temperature consequently. Current temperature, current RPM, next temperature and next RPM store as a record of dataset. Then, fan speed is set to the next RPM in the RPMs list to find temperature changes and save them as a record in the dataset and this process is continued for all different RPMs. After checking the effect of changing RPM in temperature from one program, we add programs steadily to find these variations on other number of programs. When the dataset is ready, LLNFM is learned by generating dataset to produce the architecture and parameters of the network.

3.2 Online phase

During the online phase, fan speed selection is done by using extracted architecture and parameters from the previous phase. Figure 3 shows the block diagram of our proposed NF based fan speed model. The inputs of LLNFM are current fan speed, current temperature, and the temperature threshold. The output of LLNFM is the future fan speed which is an appropriate fan speed that holds temperature around temperature threshold while minimizing fan power consumption due to optimized selected fan speed.

Our proposed controller works in two states: where there is at least one running program that tries to hold temperature as close to the temperature threshold (T_{thr}). In case there is not any program, the controller tries to hold temperature around the idle temperature threshold (T_{idle}). At each iteration, we predict the appropriate fan speed for each core and then select the maximum value among the predicted speeds of all cores as a current fan speed.

The continuous CPU fan speeds can be divided into discrete distinct speeds due to the discontinuous PWM (Pulse Width Modulation) values, which are used for changing fan speed. By the result of discretization of fan speeds, a small change of them may fluctuate between two consecutive discrete values. To avoid this phenomenon, new selected fan speed should compare with the current fan speed. If the difference between them is more than a specified threshold value, fan speed change according to the selected value. Our proposed Fan Speed Controller depicted in Algorithm 1.

4. Experimental Results

This section provides experimental results for different applications from SPEC CPU2006 benchmarks. In the rest of this section we represent the experimental environment and analyze the obtained results.

4.1 Fan power measurement

For each discrete fan speed PWM, which is used in proposed algorithm, we measured fan power through voltage and current measurement. Voltage and current of

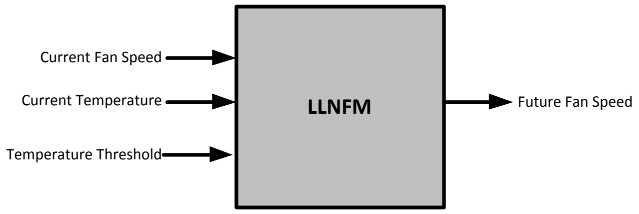


Figure 3-The proposed NF fan speed model.

Algorithm 1 Fan Speed Controller

```
1: initialize $T_{thr}$ , $T_{idle}$
2:while(True)do
3: If (there is at least one program) then
4:$T_{target}$=$T_{thr}$
5:     else
6:$T_{target}$=$T_{idle}$
7:endif
8:     $RPM_{curr}$=read current fan speed
9: for i =1 to Core_count do
10:          $T_{curr}$ = read $core_i$ temperature
11:     PredictedFanSpead [i]=Predict ($RPM_{curr}$,$T_{target}$,$T_{curr}$)
12: endfor
13: SelectedFanSpeed=max(PredictedFanSpead);
14: if abs(OldFanSpeed- SelectedFanSpeed)>= $S_{gate}$then
15:          ChangeFanSpeed(SelectedFanSpeed);
16:          OldFanSpeed= SelectedFanSpeed;
17:endif
18:endwhile
```

each PWM is measured by clamp data that depicted in Table 1.

Table 1.
Voltage, current, and power consumption of each PWM.

PWM	Voltage (V)	Current (mA)	Power (mW)
75	3.83	29.86	114.3638
90	4.58	30.32	138.8656
105	5.23	30.78	160.9794
120	5.99	30.88	184.9712
135	6.66	31.2	207.792
150	7.42	31.88	236.5496
165	8.09	34.16	276.3544
180	8.86	37.25	330.035
195	9.54	38.1	363.474
210	10.32	39.4	406.608
225	11	42	462
240	11.79	44.6	525.834
255	12.17	45.2	550.084

4.2 Experimental setup

We present a fan controller that dynamically sets the fan speed according to processor temperature, fan speed and specific temperature threshold. To evaluate the proposed fan controller, we conduct our experiments with two to seven SPEC2006 benchmarks executed simultaneously on an Intel's Core i7-950. The size of the main memory of the system is 6 GB. The Linux kernel version is 3.8.0. The LM sensor [20] is used to read core temperatures and change the fan speed. The range of fan PWMs of our experiments is from 75 to 255.We use cpufreq Linux subsystem to fix the processor frequency.

In order to compare our experiments, we rebuild the traditional fan controller. It uses a static mapping between the fan speed and the processor temperature. The fan speed is set toPWM_{min} when the temperature is no more than T_{min}, and increase linearly with temperature to full speed PWM_{max} when the temperature reaches T_{max}. The parameter values in our platform are: PWM_{min}=60, T_{min}=35°C and T_{max}=75°C. The maximum allowed fan speed for traditional fan control and our presented method is also set to 75% nominal speed. Temperature threshold is set to different values in our model to examine applicability in different situations.

4.3 Experimental analysis

In this section, we evaluate the proposed controller against the traditional controller scheme. Figure 4 and Figure 5 illustrate fan power consumption and temperature of the cores for traditional fan controller and our proposed method (NFSC) on Intel core i7-950with simultaneous running four and six programs respectively. The temperatures and fan power are sampled every second. As shown in the figures, our proposed method overcomes traditional controller in the fan power reduction by holding temperature around the temperature threshold. Figure 6 and Figure 7 show total power consumption in 500 seconds and average power consumption of fan and average and peak temperature, respectively for traditional controller and our proposed method with various number of programs. Also, for each experiment, different temperature threshold are used(e.g. 63˚C is temperature threshold when running five programs while for six programs experiment, temperature threshold is 58˚C).After running a different set of programs, it is observable that our proposed technique (NFSC) reduces the total fan power consumption almost30% (350W) in 2500 seconds with 2.6◦C average temperature increase compared to the traditional fan controller, in average. It should be noted that the average temperature is the mean of four core temperature running programs simultaneously from beginning to the end.

Table 2 summarizes the comparison results of our proposed method against the traditional fan controller that are the mean values extracted at run-time for two to seven attempted applications. Hence, compared to traditional fan controller, our proposed method indeed leads to more fan power reduction by holding temperature around the temperature threshold.

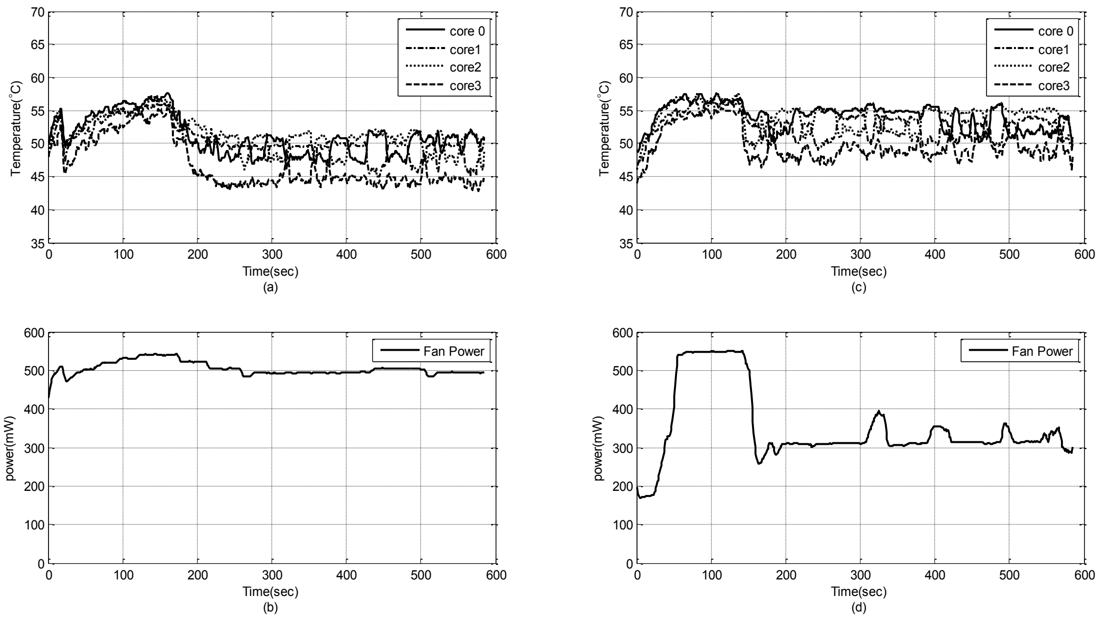


Figure 4-(a) Core temperature and (b) power consumption of traditional fan controller and (c) cores, temperature and (d) power consumption of our proposed method by running four programs with a temperature threshold of 58 ◦C.

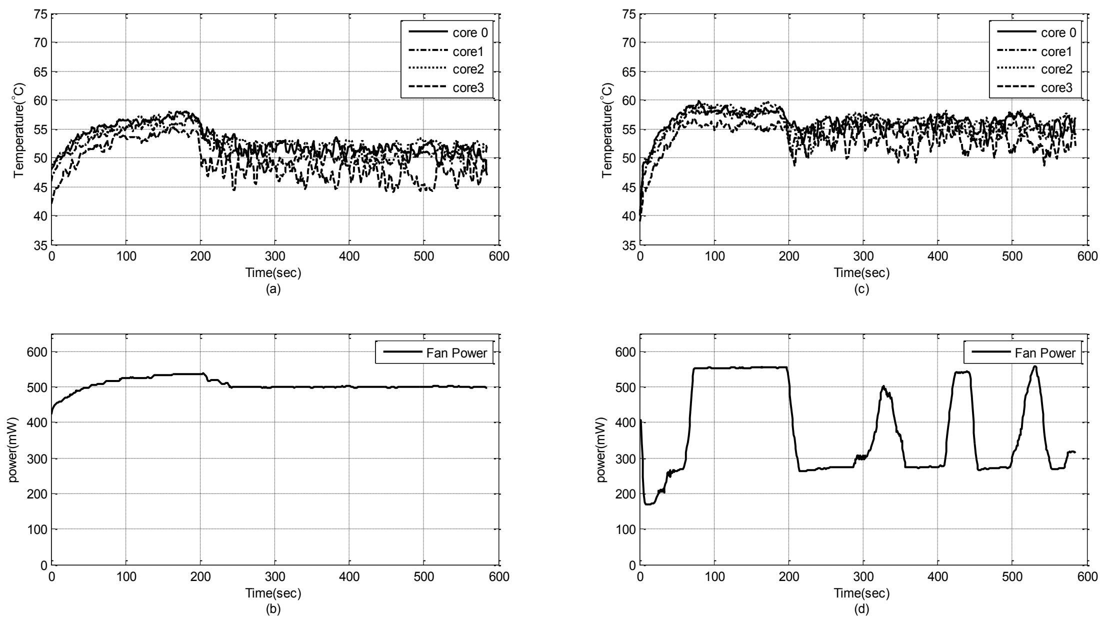
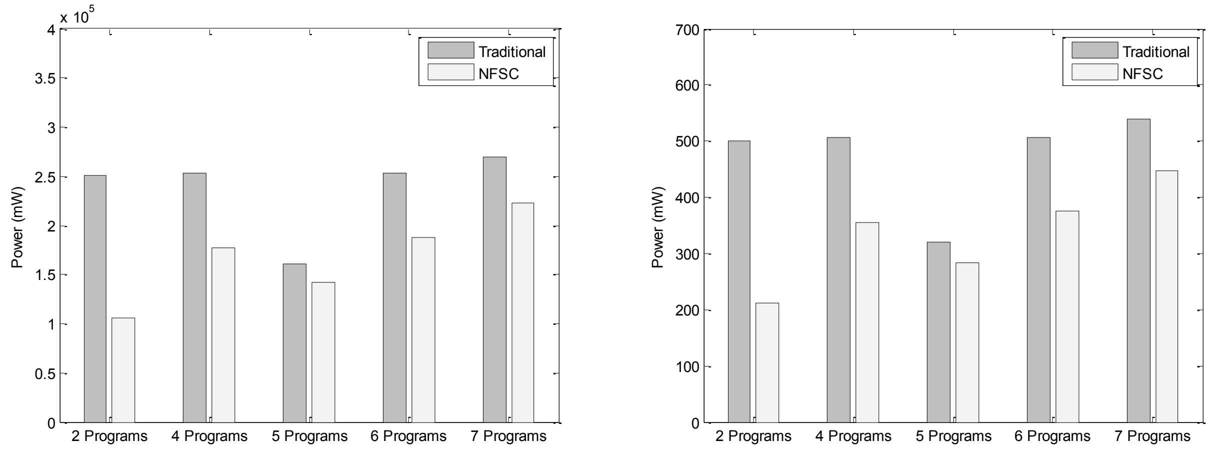
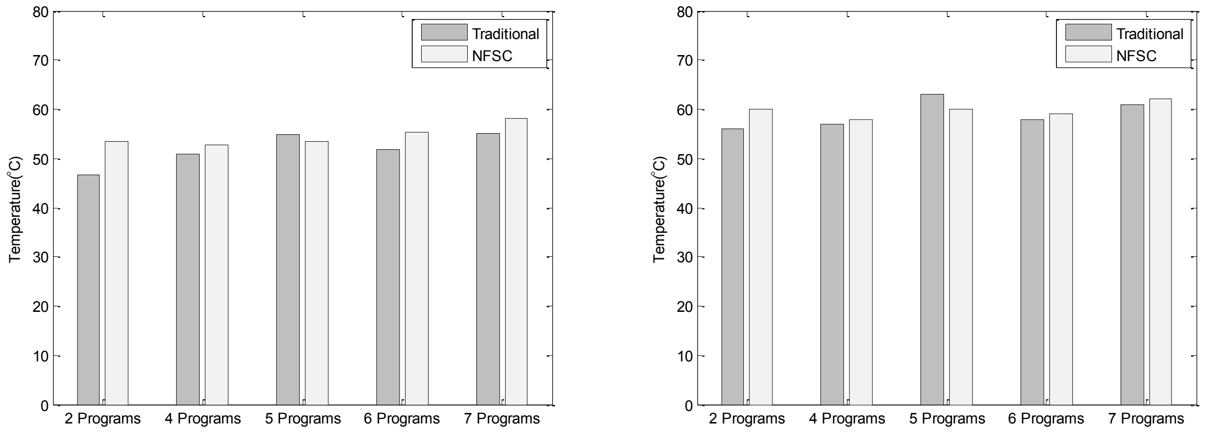


Figure 5-(a) Core temperature and (b) power consumption of traditional fan controller and (c) cores, temperature and (d) power consumption of our proposed method by running six programs with a temperature threshold of 58◦C.



(a) (b)

Figure 6 -Comparison of different algorithms with various number of programs in terms of (a) total and (b) average power consumption.



(a) (b)

Figure 7 -Comparison of different algorithms with various number of programs in terms of (a) average and (b) peak temperature.

Table 2
Average and total power consumption and average temperature in 2500 seconds.

Method	Average power (mW)	Total power (W)	Average temperature (°C)
Traditional	475	1187.28	51.95
NFSC	334.7	836.74	53.61

5. Conclusion

In this paper, a fan speed control model based on Neuro-fuzzy (NF) to minimizing power consumption of fan and at the same time avoids violating from the temperature threshold for multicore processors is presented. As demonstrated, an appropriate fan speed controller is extremely important in saving fan power. Experimental results based on practical benchmarks (SPEC CPU2006) running on a desktop platform (Intel Core i7-950) indicate that our proposed model can overcome traditional fan controller in saving power with negligible increase of average temperature without violating temperature threshold. Our proposed method improved average power consumption of fans about 30% relate to traditional fan controller with an almost 5% (2.6°C) average temperature overhead.

6. REFERENCES

[1] Kong, J., Chung, S. W., & Skadron, K. (2012). Recent thermal management techniques for microprocessors. *ACM Computing Surveys* (CSUR), 44(3), 13.

[2] Cai, Q., González, J., Magklis, G., Chaparro, P., & González, A. (2011, August). Thread shuffling: Combining DVFS and thread migration to reduce energy consumptions for multi-core systems. In *Low Power Electronics and Design (ISLPED) 2011 International Symposium* on (pp. 379-384). IEEE.

[3] Choi, J., Cher, C. Y., Franke, H., Hamann, H., Weger, A., & Bose, P. (2007, August). Thermal-aware task scheduling at the system software level. In *Proceedings of the 2007 international symposium on Low power electronics and design* (pp. 213-218). ACM.

[4] Liu, G., Fan, M., & Quan, G. (2012, March). Neighbor-aware dynamic thermal management for multi-core platform. In *Design, Automation & Test in Europe Conference & Exhibition* (DATE), 2012 (pp. 187-192). IEEE.

[5] Kumar, A., Shang, L., Peh, L. S., & Jha, N. K. (2006, July). HybDTM: a coordinated hardware-software approach for dynamic thermal management. *In Proceedings of the 43rd annual Design Automation Conference* (pp. 548-553). ACM.

[6] Yeo, I., & Kim, E. J. (2009, April). Temperature-aware scheduler based on thermal behavior grouping in multicore systems. In *Proceedings of the Conference on Design, Automation and Test in Europe* (pp. 946-951). European Design and Automation Association.

[7] Yeo, I., Liu, C. C., & Kim, E. J. (2008, June). Predictive dynamic thermal management for multicore systems. In *Proceedings of the 45th annual Design Automation Conference* (pp. 734-739). ACM.

[8] Liu, Z., Xu, T., Tan, S. X. D., & Wang, H. (2013). Dynamic thermal management for multi-core microprocessors considering transient thermal effects. In *ASP-DAC* (pp. 473-478).

[9] Patterson, M. K. (2008, May). The effect of data center temperature on energy efficiency. In *Thermal and Thermo mechanical Phenomena in Electronic Systems*, 2008. ITHERM 2008. 11th Intersociety Conference on (pp. 1167-1174). IEEE.

[10] Paparrizos, G. (2003). An Integrated fan speed control solution can lower system costs, reduce acoustic noise, power consumption and enhance system reliability. Technical report, *Microchip Technology Inc*.

[11] Li, D., Ge, R., & Cameron, K. (2010, September). System-level, Unified In-band and Out-of-band Dynamic Thermal Control. In *Parallel Processing* (ICPP), 2010 39th International Conference on (pp. 131-140). IEEE.

[12] Hanumaiah, V., & Vrudhula, S. (2012). Energy-efficient Operation of Multi-core Processors by DVFS, Task Migration and Active Cooling.

[13] Shin, D., Chung, S. W., Chung, E. Y., & Chang, N. (2010). Energy-Optimal Dynamic Thermal Management: Computation and Cooling Power Co-Optimization. *Industrial Informatics, IEEE Transactions* on, *6*(3), 340-351.

[14] Ayoub, R., Indukuri, K., & Rosing, T. S. (2011). Temperature aware dynamic workload scheduling in multi socket CPU servers. *Computer-Aided Design of Integrated Circuits and Systems*, *IEEE Transactions* on, 30(9), 1359-1372.

[15] Wang, Z., Tolia, N., & Bash, C. (2010, April). Opportunities and challenges to unify workload, power, and cooling management in data centers. In *Proceedings of the Fifth International Workshop on Feedback Control Implementation and Design in Computing Systems and Networks* (pp. 1-6). ACM.

[16] Hanumaiah, V., Vrudhula, S., & Chatha, K. S. (2011). Performance optimal online dvfs and task migration techniques for thermally constrained multi-core processors. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions* on, *30*(11), 1677-1690.

[17] Ayoub, R., Sharifi, S., & Rosing, T. S. (2010, March). Gentlecool: Cooling aware proactive workload scheduling in multi-machine systems. In *Proceedings of the Conference on Design, Automation and Test in Europe* (pp. 295-298). European Design and Automation Association.

[18] Takagi, T., Sugeno, M., (1985). Fuzzy identification of systems and its application to modeling and control, Syst. Man Cybern, *IEEE Transactions* on, 15 (1), 116–132.

[19] Nelles, O. (2001). Nonlinear system identification: from classical approaches to neural networks and fuzzy models. Springer.

[20] Lm sensors linux hardware monitoring [Online]. Available: http://www.lm-sensors.org.