ELectronics EXpress

IEICE Electronics Express, Vol.16, No.7, 1–6

LETTER

A design of adaptive PID controller based on thermal characteristics of mobile application processors

Hyun Hak Cho[1], **Chang Min Eun**[1], **and Ok Hyun Jeong**[1a)]

Abstract With the advance of process technology, the power density of chipsets has rapidly increased, which has resulted in thermal issues. Modern mobile devices use a thermal management model (TMM) to solve such thermal issues. In this paper, we propose an adaptive proportional-integral-derivative (PID) controller based on mobile CPU system characteristics. In the proposed method, we dynamically adjust the PID constants according to the thermal characteristics. We evaluate thermal mitigation capability and performance using a commercial smartphone. The results show that our scheme decreases the overall system temperature by 5.13% and simultaneously improves the system performance by 5.51%.

Keywords: thermal management model, thermal mitigation algorithm, PID controller, thermal characteristics, application processor, mobile device

Classification: Electron devices, circuits and modules

1. Introduction

The power and thermal issues of mobile devices have been emphasized over the years [1, 2]. With the advance of process technology, the power density of chipsets has increased, resulting in rises in on-chip temperature [3, 4, 5]. Many researchers have studied various thermal management techniques to solve the thermal problems that occur when the on-chip temperature exceeds the thermal design limits of chipsets [6, 7]. In particular, most mobile devices primarily adopt software-based technologies due to their compact size and power budgets [8].

J. Zhou *et al.* designed a two-stage temperature-aware task scheduling scheme that reduces system energy consumption using a task allocation scheme and slack distribution policy [9]. The authors in [10] presented a temperature-aware dynamic voltage and frequency scaling (DVFS) technique to co-optimize the power and performance through frequency stabilization. In previous work [11], the authors proposed and validated a surface temperature-aware thermal management technique using precise surface temperature estimation based on thermal RC-network model.

In this paper, we propose a novel adaptive proportional-integral-derivative (PID) controller for mobile processors with non-linear thermal characteristics because the conventional PID controller is usually not effective for controlling higher-order systems, non-linear systems, or complex systems [12, 13]. In addition, we experimentally derive the thermal characteristics of mobile processor and calculate PID constants based on it. The major contributions of this paper are summarized as follows:

- Adaptive PID controller considering *CPU operation pattern*[1]: The thermal characteristics of the target mobile processor system change in real time according to the *CPU operation pattern*. Therefore, we propose an adaptive PID controller that dynamically changes the PID constants according to the *CPU operation pattern* in real time.
- PID constants based on system characteristics: We experimentally derive the power and thermal characteristics of a commercial mobile application processor (AP). Subsequently, we calculate the PID constants based on the system characteristics using first order plus time delay (FOPTD) model and Chien, Hrones, and Reswick (CHR) method.
- Validation in real mobile systems: We use an off-the-shelf smartphone to verify thermal mitigation effect and system performance. We improve the average core temperature by 0.83%, the average crystal oscillator temperature by 5.13%, and the performance by 5.51%.

2. Background

2.1 Typical mobile thermal management model

The TMM used in mobile devices is a configurable framework designed by an AP chipset vendor. The purpose of the mobile TMM is to manage the temperature so that the chipset does not exceed the thermal design limits while achieving high performance. This framework uses on-chip temperature sensors to monitor the temperatures of various thermal management devices (i.e., CPUs, GPUs, PAs, and displays). If the temperatures from different thermally active points exceed the preconfigured *set-point*, the TMM controls the thermal responses of the entire system according to the thermal mitigation algorithm (TMA) [14, 15].

2.2 Thermal mitigation algorithm

The TMA defines how to adjust the device performance mitigation level to reduce the on-chip temperature when the temperatures of the thermal management devices exceed the *set-point*. In particular, the TMA controls the *maximum*

[1]Department of Electronic Engineering, Sogang University, 35 Baekbeom-ro, Mapo-gu, Seoul 04107, Republic of Korea
a) ohjeong@sogang.ac.kr

DOI: 10.1587/elex.16.20190084
Received February 13, 2019
Accepted February 27, 2019
Publicized March 11, 2019
Copyedited April 10, 2019

[1]The *CPU operation pattern* indicates the combination of the number of active cores and the operating frequency of each core.

allowable CPU frequency to alleviate CPU thermal problems[2]. Since the CPU operates at the highest frequency available to handle the workload, the TMA can have a direct impact on the CPU operating frequency by adjusting the *maximum allowable CPU frequency*. In this manner, the TMA manages the thermal risk and performance of the CPU [15].

The TMA can be classified into threshold control algorithms and dynamic control algorithms, according to the method of controlling the performance mitigation level. The threshold control algorithm limits the performance mitigation level to a specified level when the thermal problems occur. Once the temperature falls below a certain level (*clear-point*), the mitigation level is again set to the initial value (maximum performance level). On the other hand, when the on-chip temperature exceeds the *set-point*, the dynamic control algorithm starts to reduce the performance mitigation level by one step at every sampling time until the temperature decreases below the *set-point* again. As the temperature decreases below the *set-point*, the algorithm increases the mitigation level one step at a time, ensuring that the device can provide better performance.

2.3 Conventional PID controller

The PID controller is widely used and the most common control algorithm in the process control industry because of its simplicity and robustness [17, 18]. The general discrete-time PID controller output $u(k)$ can be calculated as

$$u(k) = K_p e(k) + K_i T_s \sum_{i=0}^{k} e(i) + \frac{K_d}{T_s}[e(k) - e(k-1)]. \quad (1)$$

where $e(k)$ is the error value, k is the discrete-time index, T_s is the sampling period, K_p is the proportional constant, K_i is the integral constant, and K_d is the derivative constant [19]. In this case, the $e(k)$ is the difference between a desired response $r(k)$ and an actual response $c(k)$. The PID controller output depends on the PID constants, the current error value, the sum of historic error values, and the change rate of the error value. Therefore, the values of the PID constants must be carefully selected based on the target process characteristics to guarantee the effectiveness of the control mechanism [20, 21].

The integral component of the conventional PID controller tends to accumulate large errors, which can cause overshooting or undershooting control. This is known as integral windup [22]. To solve this problem, the PID controller output is calculated using the first derivative of Eq. (1). The following equation is the velocity form of the discrete-time PID controller [17, 19]:

$$u(k) = u(k-1) + K_p[e(k) - e(k-1)] + K_i T_s e(k) + \frac{K_d}{T_s}[e(k) - 2e(k-1) + e(k-2)]. \quad (2)$$

In this paper, we use the velocity form of the PID controller. As shown in Eq. (2), the PID constants are multiplied by the change of the error value, the current error value, and the rate of the error change, respectively.

[2]The CPU operating frequency is set to a value below the *maximum allowable CPU frequency* by a Linux governor [16].

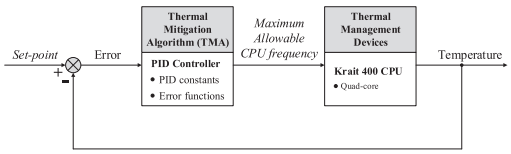


Fig. 1. Block diagram of TMM when using PID control algorithm.

3. Adaptive PID controller

3.1 Target AP and CPU

In this paper, we select the Snapdragon 801, a high-performance mobile AP, as the target system. The Snapdragon 801 has quad Krait 400 CPU cores of up to 2.4 GHz. The Krait 400 CPU can independently operate at 15 discrete frequencies with a granularity of approximately 150 MHz in the range of 300–2450 MHz [15].

The TMM of the target AP can run three TMAs: threshold, simple dynamic, and PID control algorithm. The PID control algorithm of the target system uses conventional PID controller architecture using pre-determined PID constants. Fig. 1 shows a block diagram of the TMM when the PID control algorithm is used for CPU thermal management. The *set-point* is used as the desired response, the actual system response is on-chip temperature, and the error function is used as the controller input. The PID controller calculates the *maximum allowable CPU frequency* according to the error function and the PID constants.

3.2 Motivation for proposed PID controller

To use the PID controller, the PID constants must be calculated based on the characteristics of the target system. In this paper, we characterize the target system based on power consumption and temperature variation. Considering the multi-core platform, there is a significant difference in power consumption and temperature variation according to the *CPU operation pattern* [23, 24, 25, 26].

The Snapdragon 801 has over 60 thousand different *CPU operation patterns* because the quad-core can independently operate at 15 discrete frequencies. Therefore, we empirically derive the power and thermal characteristics that vary with the *CPU operation pattern* and find the PID constants for each pattern. Finally, we design an adaptive PID controller that adjusts the PID constants based on the *CPU operation pattern* in real time.

3.3 System characteristics of target CPU

It is time-inefficient to derive the system characteristics for all *CPU operation patterns* of the target AP. Therefore, we derive the system characteristics assuming that all CPU cores operate at the same frequency. In other words, we analyze the system characteristics using only 60 *CPU operation patterns*. We also classify the operation patterns with similar system characteristics into groups to reduce the complexity of implementing the adaptive PID controller.

The analysis of the system characteristics proceeds in two stages: In the first stage, we classify the entire system

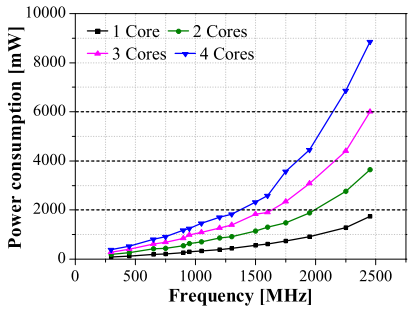


Fig. 2. Power consumption of target CPU according to *CPU operation pattern*.

based on the power characteristics. In the second stage, we derive the thermal characteristics of each group and calculate the PID constants.

3.3.1 Target system group classification

To derive the power characteristics of the target CPU, we measure the CPU power consumption for all *CPU operation patterns*. We use the test scenario ("fast discharger" application) that allocates almost 100% of the workload to each core.

Fig. 2 shows the measurement results of CPU power consumption according to the *CPU operation pattern*. In this paper, we divide the target system into four groups based on the power consumption of the cases using the maximum frequency (2450 MHz) on a single-core system. Group A includes cases that consume low power (2,000 mW or less). Group B includes cases that consume a medium level of power (2,000–4,000 mW). Group C includes cases that consume a high level of power (4,000–6,000 mW). Group D includes cases that consume an extreme amount of power (over 6,000 mW). As CPU thermal problems mainly occur when operating in a medium- or higher-power consumption cases, we analyze the thermal characteristics by subdividing Groups B–D according to the number of active cores.

3.3.2 PID constants based on system characteristics

We use the FOPTD model and CHR method to select the PID constants. The PID constants for each group are calculated as the average value of the cases belonging to the group.

The FOPTD model is widely used for the analysis of various processes because it enables the simple characterization of the target process [21, 27]. Fig. 3 shows the system parameters (steady-state gain (K_m), time constant (T), and time delay (L)) of the FOPTD model [13, 22]. The system parameters can be obtained by drawing a tangent line at the inflection point of the step response of the target system, as shown in Fig. 3 [20]. The initial PID constants are then calculated using the CHR method [21, 28]. Table I summarizes the CHR method formulae. There are two kinds of CHR method tuning formulae depending on the degree of overshoot [29]. We calculate the PID constants using the "with 0% overshoot" formula.

To confirm the CPU thermal characteristics, we measure temperature variation for all *CPU operation patterns*

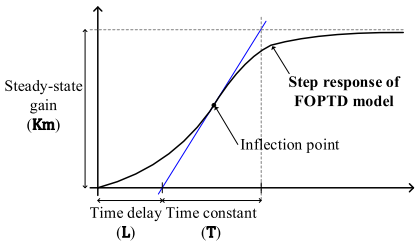
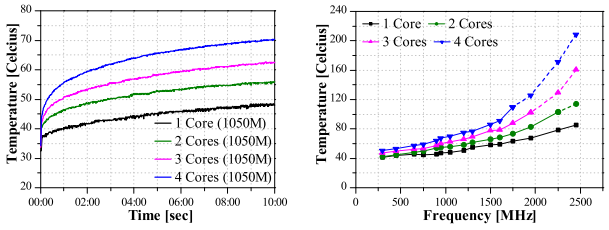


Fig. 3. Step response of FOPTD model and system parameters.

Table I. CHR method tuning formulae for set-point regulation

Design Criteria	K_p	K_i	K_d
With 0% Overshoot	$\frac{0.6T}{K_mL}$	$\frac{0.6}{K_mL}$	$\frac{0.3T}{K_m}$
With 20% Overshoot	$\frac{0.95T}{K_mL}$	$\frac{0.68}{K_mL}$	$\frac{0.45T}{K_m}$

using the fast discharger application[3]. Fig. 4(a) shows the measurement results of the CPU core temperature change for four *CPU operation patterns* as an example. As shown in Fig. 4(b), the tendency of the CPU temperature change is different depending on the *CPU operation pattern*. Therefore, we derive the system parameters using the CPU temperature variation graph for each case, and calculate the PID constants using it. The steady-state temperature of the case where shutdown occurs due to excessive heat generation is estimated using the linear regression model according to the CPU power consumption. The experimental results of the steady-state temperatures of the target system are given in Fig. 4(b). The dotted line indicates the cases in which shutdown occurs.



(a) Examples of CPU core temperature measurement results

(b) Steady-state temperature variation of CPU core

Fig. 4. Results of thermal characteristics experiment of target CPU according to *CPU operation pattern*.

We derive the PID constants based on the thermal characteristics of all *CPU operation patterns* and then calculate the average value for each group. The conventional PID constants continually attempts to minimize the error. However, the TMM must provide high performance while solving the thermal issues, rather than maintaining the on-chip temperature at the *set-point*. Therefore, we empirically validate the performance of the initial PID constants and adjust them to ensure high performance.

[3]We set the chipset to be forced shutdown if the on-chip temperature exceeds 110 °C to prevent excessive heat generation and malfunctioning of the chipset.

As mentioned in Section 2.3, the derivative constant is related to the change rate of the process temperature. Accordingly, the controller output becomes large if the rate of temperature change is fast. We perform the fine tuning of the derivative constant to prevent unnecessary performance limitation.

3.4 Proposal of adaptive PID controller

Fig. 5 shows the procedure in which the *maximum allowable CPU frequency* is set using the adaptive PID control algorithm. Our control algorithm starts to operate when the device is powered on. In its initial state, our scheme sets the *maximum allowable CPU frequency* to the maximum frequency (2450 MHz) of the target CPU and the PID constants to the average value of all groups. The on-chip temperature of all CPU cores are also periodically monitored in each sampling period. In addition, our proposed method monitors the current *CPU operation pattern* in each sampling period and changes the PID constants in real time based on the average of the *CPU operation pattern* during the *sample history length*. The error function calculates the difference between the *set-point* and the current core temperature. The adaptive PID controller output is then calculated according to the error function and the PID constants. The *maximum allowable CPU frequency* of the current sampling time is calculated as the sum of the *maximum allowable CPU frequency* of the previous sampling time and the controller output of the current sampling time. In this manner, our adaptive PID control algorithm manages the CPU thermal issues until the smartphone is terminated.

4. Evaluation

4.1 Experimental environments

In this paper, we use the commercial smartphone LG G3, which adopts the Snapdragon 801 as the AP, to validate in real mobile environments. All the configurations of the experiments are set to be the same, and the default value of the device is used: The *set-point* is set to 85 °C, the sampling period is set to 100 ms, and the performance mitigation level of the threshold control algorithm is set to 1750 MHz[4].

4.2 Experimental methodology

In real mobile environments, the temperatures of various chipsets inside the smartphone are important, but the rear surface temperature is also important [11, 30]. Therefore, we use the on-chip temperature sensors of all CPU cores and the crystal oscillator. The temperature of the crystal oscillator represents the rear surface temperature because the crystal oscillator temperature varies according to the heat diffused by different components, including CPUs. In this paper, we use the average temperature during the test period as the thermal mitigation capability index.

We analyze the overall system performance of the smartphone using the AnTuTu benchmark v4.5.1. This benchmark runs several test programs and represents the system performance as the summation of each program score, which is called the AnTuTu total score. The AnTuTu benchmark v4.5.1 runs the test programs in the following order: runtime, CPU integer, CPU float-point, RAM operation, RAM speed, multitask, database I/O, storage I/O, 2D graphics, and 3D graphics.

In this paper, we repeat a single AnTuTu test to confirm the heat generation effect in continuous-use smartphone environments[5]. We run 10 consecutive single AnTuTu tests so that the temperature of the smartphone adequately saturate. We use the average of the AnTuTu total scores during the AnTuTu back-to-back test as the system performance index.

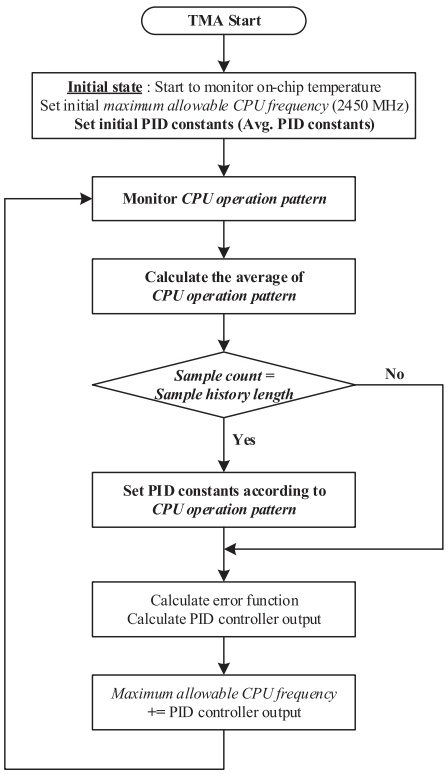


Fig. 5. Overview of CPU thermal mitigation using adaptive PID control algorithm.

4.3 Experimental results

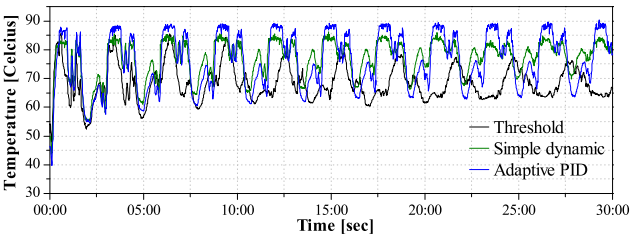
Fig. 6 shows the average temperature of all CPU cores, the average crystal oscillator temperature, and the variation of the single AnTuTu total scores during the AnTuTu back-to-back test. Table II summarizes the thermal mitigation capability and the system performance.

4.3.1 Results of conventional control algorithms

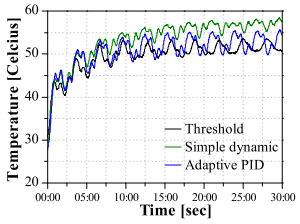
In Fig. 6(a), the CPU core temperature changes periodically according to the single AnTuTu test time (approximately 3 minutes). In addition, since the heat generated by the device accumulates as the AnTuTu back-to-back test progresses, the crystal oscillator temperature gradually increases and the system performance gradually decreases, as shown in Fig. 6(b) and Fig. 6(c).

[4]This mitigation level is the default value of the Snapdragon 801, which is determined by the device manufacturer.

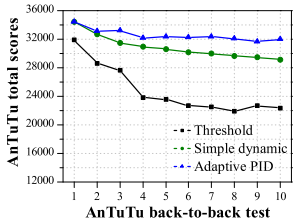
[5]We call this test the AnTuTu back-to-back test.



(a) Average CPU core temperature



(b) Average crystal oscillator temperature



(c) Variation of single AnTuTu total scores

Fig. 6. Experimental results during AnTuTu back-to-back test.

Table II. Comparison of thermal mitigation capability and system performance according to TMA type

TMA Types		Threshold	Simple Dynamic	Adaptive PID
AnTuTu Total Scores		24789.11	30856.18	32557.84
Average Temperature [°C]	Cores	68.37	76.86	76.22
	Crystal Oscillator	49.53	53.45	50.71

When using the threshold control algorithm, the core temperature decreases rapidly after reaching the *set-point* during the AnTuTu back-to-back test. Therefore, the average temperature of the CPU cores is reduced by 12.41% compared with the simple dynamic control algorithm. This means that an excessive performance limitation takes place to dissipate heat. Fig. 6(b) and Table II also show that the crystal oscillator temperature is kept lower by this effect. The average temperature of the crystal oscillator in the case using the threshold control algorithm is 7.93% lower than that of the simple dynamic control algorithm. However, the system performance of the threshold control algorithm is 24.47% lower than that of the simple dynamic algorithm because of excessive performance limitations. Furthermore, it can be seen that performance degradation is larger when using the threshold control algorithm.

The mobile TMM aims to guarantee an environment that can provide high performance while satisfying thermal constraints. Therefore, the simple dynamic control algorithm is the adequate method for managing CPU thermal problems than the threshold control algorithm which evokes severe performance degradation. Hence, we mainly compare the proposed algorithm with the conventional simple dynamic control algorithm, in the next subsection.

4.3.2 Results of adaptive PID control algorithm

The adaptive PID control algorithm basically prevents excessive performance limitation by adjusting the PID constants according to the *CPU operation pattern* in real time. Furthermore, the fine tuning of the derivative constant reduces performance limitation in the period where the rate of temperature change is fast. Therefore, as shown in Fig. 6(a), the adaptive PID control algorithm provides better performance while maintaining a relatively high temperature during the CPU test period. The AnTuTu total score of the proposed scheme is 32557.84, which is 5.51% higher than that of the simple dynamic control algorithm and 31.34% higher than that of the threshold control algorithm. In addition, Fig. 6(c) shows that the proposed algorithm minimizes performance degradation during the AnTuTu back-to-back test.

When using the proposed control algorithm, the average temperatures of the CPU cores and the crystal oscillator are 76.22 °C and 50.71 °C, respectively. Compared with the simple dynamic control algorithm, the core and crystal oscillator temperatures are 0.83% and 5.13% lower. As shown in Fig. 6(a) and Fig. 6(b), it is possible to keep the crystal oscillator temperature relatively low by reducing the CPU temperature by aggressively throttling CPU performance during the period in which the CPU peak performance is not required. In this paper, we focus on improving the system performance by using the proposed algorithm, but if the *set-point* is lowered according to other design goals, the temperature can be further improved while maintaining the system performance.

5. Conclusion

The mobile TMM aims to ensure that chipsets can provide high performance while meeting thermal constraints. In this paper, we experimentally derive the PID constants based on the system characteristics, and then propose an adaptive PID control algorithm that dynamically adjusts the PID constants to be adequate for the *CPU operation pattern*. The proposed algorithm improves the system performance and temperature simultaneously by considering the system characteristics that vary in real time. Compared with the simple dynamic control algorithm, our method improves the system performance by 5.51%. Moreover, our scheme reduces the average temperatures of the CPU cores and the crystal oscillator by 0.83% and 5.13%, respectively.

In conclusion, our research provides insights into the design and application phase of TMM for mobile processors. Furthermore, the methodology of deriving the system characteristics of mobile AP can provide design flexibility because it can be easily applied to other mobile AP platforms and different chipsets. As a future work, we are planning to improve the TMM of heterogeneous mobile processors and 5G modem.

Acknowledgments

Ok Hyun Jeong is the corresponding author of this paper. This research was supported by the Commercializations Promotion Agency for R&D Outcomes (COMPA) funded by the Ministry of Science, ICT and Future Planning (MSIP) (No. 2017K000348, Development of a circular antenna array system for P2MP transmission over the millimeter-wave environment).

References

[1] K. Dev, *et al.*: "Power mapping and modeling of multi-core processors," Proc. Int. Symp. Low Power Electron. Des. (ISLPED) (2013) 39 (DOI: 10.1109/ISLPED.2013.6629264).

[2] K. Sekar: "Power and thermal challenges in mobile devices," Proc. 19th Annu. Int. Conf. Mob. Comput. Networking (2013) 363 (DOI: 10.1145/2500423.2505320).

[3] D. Brooks and M. Martonosi: "Dynamic thermal management for high-performance microprocessors," Proc. 7th Int. Symp. High-Perform. Comput. Archit. (HPCA) (2001) 171 (DOI: 10.1109/HPCA.2001.903261).

[4] M. Pedram and S. Nazarian: "Thermal modeling, analysis, and management in VLSI circuits: Principles and methods," Proc. IEEE **94** (2006) 1487 (DOI: 10.1109/JPROC.2006.879797).

[5] G. Singla, *et al.*: "Predictive dynamic thermal and power management for heterogeneous mobile platforms," Proc. 2015 Des. Autom. Test Eur. Conf. Exhibition (DATE) (2015) 960 (DOI: 10.7873/DATE.2015.1036).

[6] K. M. Attia, *et al.*: "Dynamic power management techniques in multi-core architectures: A survey study," Ain Shams Eng. J. **8** (2017) 445 (DOI: 10.1016/j.asej.2015.08.010).

[7] J. Kong, *et al.*: "Recent thermal management techniques for microprocessors," ACM Comput. Surv. **44** (2012) 1 (DOI: 10.1145/2187671.2187675).

[8] O. Sahin and A. K. Coskun: "On the impacts of greedy thermal management in mobile devices," IEEE Embed. Syst. Lett. **7** (2015) 55 (DOI: 10.1109/LES.2015.2420664).

[9] J. Zhou, *et al.*: "Thermal-aware task scheduling for energy minimization in heterogeneous real-time MPSoC systems," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **35** (2016) 1269 (DOI: 10.1109/TCAD.2015.2501286).

[10] J. M. Kim, *et al.*: "Stabilizing CPU frequency and voltage for temperature-aware DVFS in mobile devices," IEEE Trans. Comput. **64** (2015) 286 (DOI: 10.1109/TC.2013.188).

[11] M. K. Yim, *et al.*: "Surface temperature-aware thermal management technique for mobile devices," IEICE Electron. Express **11** (2014) 20140944 (DOI: 10.1587/elex.11.20140944).

[12] B. M. Mohan and A. Shinha: "Analytical structure and stability analysis of a fuzzy PID controller," Appl. Soft Comput. **8** (2008) 749 (DOI: 10.1016/j.asoc.2007.06.003).

[13] S. W. Sung and I.-B. Lee: *PID Controllers and Automatic Tuning* (A-Jin, Seoul, 1999) 146.

[14] Qualcomm Technologies, Inc.: DragonBoard 410c based on Qualcomm Snapdragon 410E processor thermal debugging guide (2016) https://developer.qualcomm.com.

[15] Qualcomm Technologies, Inc.: Designing mobile devices for low power and thermal efficiency (2013) https://www.qualcomm.com/documents.

[16] V. Pallipadi and A. Starikovskiy: "The ondemand governor: Past, present, and future," Proc. Linux Symp. (2006) 215.

[17] K. J. Åström and T. Hägglund: *Advanced PID Control* (ISA, North Carolina, 2006) 158.

[18] A. R. Laware, *et al.*: "Real time temperature control system using PID controller and supervisory control and data acquisition system (SCADA)," Int. J. Appl. Innovation Eng. Manage. **2** (2013) 88.

[19] F. G. Shinskey: *Feedback Controllers for the Process Industries* (McGraw-Hill, Singapore, 1994) 90.

[20] F. A. Salem: "New efficient model-based PID design method," Eur. Sci. J. **9** (2013) 181.

[21] G. J. Silva, *et al.*: *PID Controllers for Time-Delay Systems* (Birkhäuser, Boston, 2005) 110.

[22] S. W. Sung, *et al.*: *Process Identification and PID Control* (John Wiley & Sons, Singapore, 2009) 129.

[23] E. Glocker and D. Schmitt-Landsiedel: "Modeling of temperature scenarios in a multicore processor system," Adv. Radio Sci. **11** (2013) 219 (DOI: 10.5194/ars-11-219-2013).

[24] G. Liu, *et al.*: "On-line predictive thermal management under peak temperature constraints for practical multi-core platforms," J. Low Power Electron. **8** (2012) 565 (DOI: 10.1166/jolpe.2012.1216).

[25] Q. Xie, *et al.*: "Dynamic thermal management in mobile devices considering the thermal coupling between battery and application processor," Proc. 2013 IEEE/ACM Int. Conf. Comput.-Aided Des. (ICCAD) (2013) 242 (DOI: 10.1109/ICCAD.2013.6691125).

[26] I. Yeo, *et al.*: "Predictive dynamic thermal management for multicore systems," Proc. 45th ACM/IEEE Des. Autom. Conf. (2008) 734.

[27] S. Tavakoli and M. Tavakoli: "Optimal tuning of PID controllers for first order plus time delay models using dimensional analysis," 4th Int. Conf. Control Autom. Proc. (2003) 942 (DOI: 10.1109/ICCA.2003.1595161).

[28] D. T. Korsane, *et al.*: "PID tuning rules for first order plus time delay system," Int. J. Innovative Res. Electr. Electron. Instrum. Control Eng. **2** (2014) 582.

[29] A. Tripathi, *et al.*: "Study and analysis of various tuning methods of PID controller for AVR system," Int. J. Res. Electr. Electron. Eng. **1** (2013) 93.

[30] Q. Xie, *et al.*: "Therminator: A thermal simulator for smartphones producing accurate chip and skin temperature maps," Proc. IEEE/ACM Int. Symp. Low Power Electron. Des. (ISLPED) (2014) 117 (DOI: 10.1145/2627369.2627641).