Predictive Dynamic Thermal and Power Management for Heterogeneous Mobile Platforms

Gaurav Singla*, Gurinderjit Kaur*, Ali K. Unver† and Umit Y. Ogras*
*School of Electrical, Computer, and Energy Engineering, Arizona State University
† Assembly & Test Technology Development, Intel Corporation

***Abstract*—Heterogeneous multiprocessor systems-on-chip (MPSoCs) powering mobile platforms integrate multiple asymmetric CPU cores, a GPU, and many specialized processors. When the MPSoC operates close to its peak performance, power dissipation easily increases the temperature, hence adversely impacts reliability. Since using a fan is not a viable solution for hand-held devices, there is a strong need for dynamic thermal and power management (DTPM) algorithms that can regulate temperature with minimal performance impact. This paper presents a DTPM algorithm based on a practical temperature prediction methodology using system identification. The DTPM algorithm dynamically computes a power budget using the predicted temperature, and controls the types and number of active processors as well as their frequencies. Experiments on an octa-core big.LITTLE processor and common Android apps demonstrate that the proposed technique predicts temperature within 3% accuracy, while the DTPM algorithm provides around 6× reduction in temperature variance, and as large as 16% reduction in *total platform power* compared to using a fan.**

I. Introduction

The abundance of logic and interconnect resources that can be integrated on a single chip pushes the limits of MPSoCs, which power the vast majority of mobile devices. Meanwhile, MPSoC design is driven by the persistent demand for faster and more powerful devices. On the one hand, the number and capacity of the CPU cores increase at a steady rate. On the other hand, the degree of heterogeneity is growing with the inclusion of asymmetric cores and accelerators such as, GPU, video codecs, digital signal processors and display processing engine. The boost in computational power inevitably increases the power dissipation, which in turn reduces the battery lifetime and raises the chip temperature. Recent results indeed reveal that the skin temperature, hence the power consumption, is the performance limiter in mobile devices [1, 2]. Furthermore, rapid changes in power and temperature also deteriorate reliability [3].

Competing requirements between performance and power consumption are addressed by a variety of design and run-time approaches that aim at maximum performance during busy periods and minimum power when there is little activity. For instance, idle power management determines the number of active cores, while dynamic voltage-frequency scaling (DVFS) controls the operating frequency of active resources to match the system performance to the application requirements [4]. Newly emerged big.LITTLE processing works in tandem with these techniques by combining high performance (big) and energy efficient (little) clusters [5]. Big cores are utilized when high performance is needed, while little cores are used during low activity periods. A recent instance of this architecture is the Samsung Exynos 5410 chip, which hosts four A15 (big) and four A7 (little) cores. Our measurements on this chip show 10× dynamic range in performance and 30× range in power consumption between the highest performance and lowest power configurations. Furthermore, moderate to high activity workloads demand big cores and raise the temperature easily beyond acceptable levels, as shown in Figure 1. The experimental platform we employ uses a fan to address this problem [6]. However, fan is *not a viable option* for mobile platforms such as smartphones, where heterogeneous MPSoCs have widespread use. Hence, there is a strong need for DTPM approaches for big.LITTLE architectures to effectively regulate temperature with minimal performance impact.

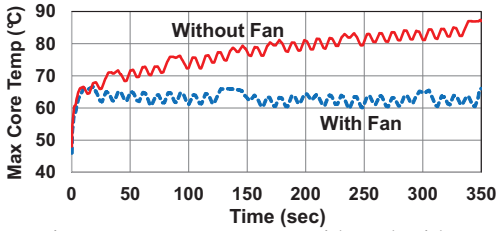

Fig. 1: Maximum core temperature with and without the fan.

The major degrees of freedom offered by the big.LITTLE architectures are controlling the CPU cluster (big or little), number of active cores, operating frequency (hence voltage) of the cores, frequency of GPU, and set the state of active accelerators, such as audio and image processors. Since the use of accelerators is largely governed by the application code and compiler, we focus on rest of the knobs. Constraining the maximum frequency to limit the temperature, while passively waiting for thermal violations and reacting by throttling the cores, impairs performance as well as reliability by causing large temperature variations [7]. In contrast, predictive approaches can take advantage of rich set of dynamic configuration capabilities to manage temperature effectively [8].

In this paper, we first present a broadly applicable methodology for generating power and thermal models for heterogeneous mobile platforms. This methodology starts from the first principles and generates mathematical models that enable *accurate power/thermal predictions tailored* to the mobile platform of interest. After empirically validating these models, we present a novel run-time technique to periodically compute the power budget that is *guaranteed* to keep the temperature within permissible limits. Finally, this power budget is used for determining the CPU cluster, number of active cores, and their frequencies to regulate temperature with minimal performance impact. The major contributions of this paper are as follows:

978-3-9815370-4-8/DATE15/©2015 EDAA

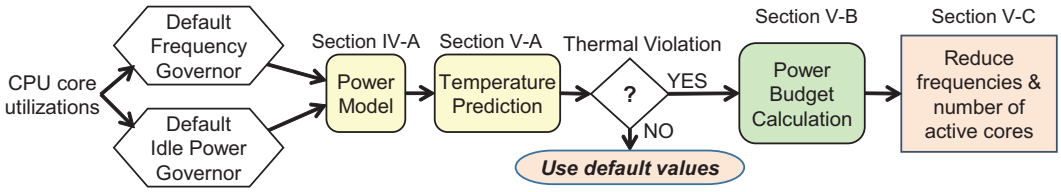

Fig. 2: High level description of the DTPM algorithm. Sections detailing individual blocks are annotated.

- A methodology for generating power and thermal models for heterogeneous MPSoCs, and experimental validation using one of the first commercial big.LITTLE architectures [6],
- A novel approach for dynamically computing power budget using temperature prediction, and an effective DTPM algorithm based on this approach,
- Exhaustive experimental evaluation which demonstrates effective thermal regulation with 6× smaller variance and as much as 16% reduction in *total platform power*.

The rest of the paper is organized as follows. Related work is presented in Section II. Overview of the DTPM technique is explained in Section III. Power and thermal model generation methodology and corresponding empirical validation appear in Section IV. Thermal prediction and DTPM algorithm based on thermal prediction are presented in Section V. Finally, extensive experimental evaluation using Samsung Exynos 5410 octa-core chip and a wide range of benchmarks is presented in Section VI, while conclusions appear in Section VII.

II. Related Work

Thermal modeling and dynamic thermal management have received significant attention due to increased power densities and reliability implications of temperature. In particular, poor performance of reactive approaches led researchers to develop compact thermal models [7, 9] and thermal prediction techniques [8, 10, 11] which can be used in pro-active thermal management methods. The thermal model presented in this paper is similar to these approaches in using a linear time invariant system to predict temperature. However, instead of relying on material and design parameters to find the model coefficients, we use actual power/temperature measurements and system identification tools to find the parameters of the model. Increasing the usage of temperature sensors and power meters make our approach feasible and accurate.

Thermal models are commonly employed for temperature control by voltage/frequency assignment and task scheduling/migration. For example, the work presented in [12] presents temperature control techniques for homogeneous multicore systems through DVFS. Similarly, temperature aware task scheduling and migration techniques are presented in [13, 14]. Model predictive and optimal control theory have been recently employed for thermal management to achieve smooth control with minimal performance loss [15, 16]. Due to scalability problems of centralized control, an agent-based thermal management technique is proposed in [17]. Finally, a hierarchical power management technique for asymmetric processors is presented in [4], where the authors try to optimize the energy/performance trade-off under thermal design power constraints. These techniques can benefit from our thermal predictor and power budget calculation approach.

While most of the thermal management techniques are implemented and validated in a simulation environment, we demonstrate our technique on a commercial big.LITTLE platform [6].Since developing simulation models for new processors is obstructed by the difficulties in finding exact floorplan, heat sink information and parameter values, researchers are usually limited to few examples such as simple XScale core and Alpha processor [8, 14]. We plan to make our power and thermal models public to enable research on emerging heterogeneous platforms.

III. Overview of the Proposed Framework

State of the art mobile platforms are highly integrated closed systems where different hardware and software modules interact very tightly. Therefore, techniques targeting these platforms cannot be designed or evaluated in isolation. Hence, all the models and algorithms presented in this work are incorporated with the existing software infrastructure, as outlined in Figure 2. Existing frequency and idle state governors, as well as the device specific drivers, *e.g.*, GPU driver, remain intact and feed their outputs to the proposed framework. The power consumption predictions are fed to the thermal model to predict the resulting temperature if these actions were taken. Hence, different governor or device specific optimizations implemented in dedicated drivers can work in coordination with the proposed framework. Unless a thermal violation is predicted, the decisions of the default drivers such as the core and GPU frequencies, choice of big or little cluster and number of active cores, are affirmed. Thus, the proposed DTPM approach is non-intrusive when the temperature is within permissible levels. We start with the temperature constraint and work backwards to determine the maximum power consumption that can be tolerated. Finally, the available budget is used to overwrite the set of active resources and their frequencies such that the temperature constraint violation can be prevented. These steps are detailed in the following sections as annotated in Figure 2.

IV. Power/Thermal Modeling Methodology

Effective management of power and temperature depends critically on accurate analytical models that can be evaluated at run-time. Therefore, we start with presenting our modeling methodology that leverages thermal and power sensors. In particular, power consumption of big core cluster P_{A15}, little core cluster P_{A7}, GPU P_{GPU} and memory P_{mem} are read using power sensors. If the power consumption of a target resource cannot be measured individually, it needs to be considered

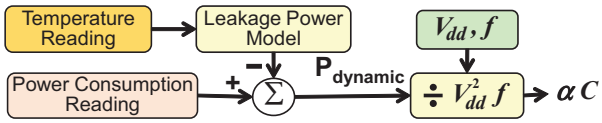

Fig. 3: Run-time computations for computing the product of the activity factor and switching capacitance.

as a part of a bigger block whose power consumption can be measured. Likewise, temperature of each big core is read through temperature sensors.

A. Power Modeling

Power models for major components such as CPU, GPU, display and battery exists in literature [3]. Therefore, we detail only our empirical approach to extract the leakage current and switching capacitance. The total power consumption can be expressed as

$$\begin{aligned} P_{total} &= P_{dynamic} + P_{leakage} \\ P_{total} &= \alpha C V_{dd}^2 f + V_{dd} I_{leakage} \end{aligned} \quad (1)$$

where α and C are the activity factor and switching capacitance, respectively. Leakage current can be extended as:

$$\begin{aligned} I_{leakage} &= A_s \frac{W}{L} (\frac{kT}{q})^2 e^{\frac{q(V_{GS}-V_{th})}{nkT}} + I_{gate} \\ I_{leakage} &= c_1 T^2 e^{\frac{c_2}{T}} + I_{gate} \end{aligned} \quad (2)$$

where A_s is a technology dependent constant, L and W are channel length and width, k is the Boltzmann constant, T is the temperature, q is the charge, V_{GS} is the gate to source voltage, V_{th} is the threshold voltage, n is the sub-threshold swing coefficient, and I_{gate} is the gate leakage current [18, 19]. These technology and device parameters are condensed into generic parameters denoted by c_1 and c_2. We employ the procedure below to find the unknown parameters assuming that dynamic power shows negligible variation with temperature.

Leakage Power Characterization: Using equations 1 and 2, the total power consumption is written as

$$P_{total} = \alpha C V_{dd}^2 f + V_{dd}(c_1 T^2 e^{\frac{c_2}{T}} + I_{gate}) \quad (3)$$

For a given operating frequency, voltage and temperature, Equation 3 has four unknowns, αC, c_1, c_2 and I_{gate}. We placed the target platform in a furnace and swept the temperature from $40°C$ to $80°C$ in increments of $10°C$. During the tests, we used a light workload running only on the big cores with fixed f and V_{dd} such that the dynamic power did not increase the temperature. Then, multiple power measurements were taken to find the unknowns and model the leakage power. The leakage power of the big core cluster with temperature is shown in Figure 4(a). In general, this analysis is repeated for each computing resource such as the little cores and GPU to extract the corresponding leakage power model.

Run-time computation of αC: At run-time, power/thermal sensors in the platform are used to measure the power consumption and temperature of the resource of interest. Then, the dynamic power consumption is found by subtracting the leakage power from the total power, as described in Figure 3. Finally, operating frequency and voltage at the time of the

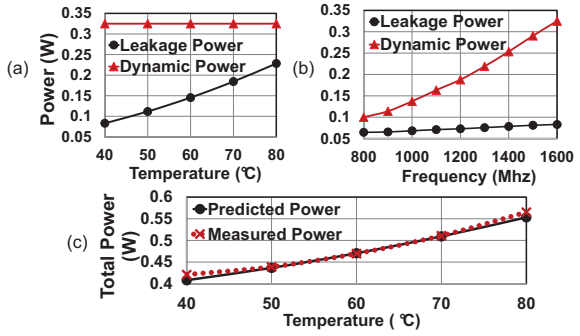

Fig. 4: Variation of leakage and power model with (a) temperature, and (b) frequency, (c) Predicted and measured power.

computation are used to extract the product of the activity factor and switching capacitance. This model is used to predict the dynamic power consumption before any decision on the frequency is made. Figure 4(b) and (c) demonstrate the dynamic power model and the accuracy of total power prediction for low activity workload, respectively. Our power model resulted in less than 4% average prediction error compared to measurement across a wide range of workloads.

B. Thermal Modeling

Using the duality between the thermal and electrical networks, one can model the dynamics of the temperature using a state-space model [8]. Suppose that there are N nodes in the network, whose temperature and power consumption are given by $[T(t)]_{N\times 1}$ and $[P(t)]_{N\times 1}$, respectively. Then,

$$C_t \frac{dT}{dt} = -G_t T(t) + P(t) \quad (4)$$

where C_t and G_t are the thermal capacitance and conductance matrices [7]. Since power/temperature measurement and control are performed periodically in OS kernels or firmware in practice, we discretize Equation 4 assuming a sampling period of T_s seconds:

$$\begin{aligned} T[k+1] &= (I - T_s C_t^{-1} G_t) T[k] + T_s C_t^{-1} P[k] \\ T[k+1] &= A_s T[k] + B_s P[k] \end{aligned} \quad (5)$$

Finding the thermal conductance and capacitance matrices (A_s and B_s) using finite element simulations or a thermal modeling framework like Hotspot [7] would require detailed design information such as floorplan, heat sink geometry, and material properties, which are either not public or very hard to obtain. Furthermore, validating the thermal model would still require actual power and temperature measurements. Therefore, we start directly with actual measurements, and employ system identification to find C_t and G_t, as detailed next.

System Identification: The input to the difference equation is $P(k)$, which is the power consumption of the major resources. For instance, $P = [P_{A7}, P_{A15}, P_{GPU}, P_{mem}]^T$ for our system, where P_{A7} and P_{A15} correspond to the little and big core clusters, respectively. In order to obtain an accurate characterization, we controlled *each of these four sources separately* while keeping the others at a minimum value. More precisely, we oscillated the frequency of big cores between the minimum and maximum values using a pseudo-random bit

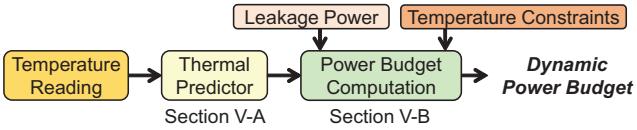

Fig. 5: Temperature prediction and power budget computation.

sequence (PRBS), and measured the temperature. The PRBS input is generated to cover a frequency spectrum, which is much broader than that excited by an arbitrary application. Then, we recorded the input $P[k]$ and output $T[k]$ time series. Finally, we used the system identification toolbox of Matlab and the input/output signals to find A_s and B_s in Equation 5.

V. Dynamic Thermal and Power Management

In the absence of a DTPM algorithm, the OS kernel wakes-up more processors and increases their frequencies as the workload intensifies. Consequently, increased power consumption elevates the temperature, which eventually results in a thermal violation. The proposed approach utilizes the power and thermal models introduced in Section IV to dynamically compute a power budget, as outlined in Figure 5. Staying within this budget guarantees that no thermal violation will happen. Then, this power budget is used at run-time to limit the types, number and frequencies of active resources.

A. Thermal Prediction and Validation

Equation 5 not only describes how the temperature evolves but it also enables temperature prediction at an arbitrary number of time steps ahead. In particular, the temperature at time step $k+n$ can be derived as:

$$T[k+n] = A_s^n T[k] + B_s \sum_{i=0}^{n-1} A_s^i P[k+n-i-1] \quad (6)$$

This equation predicts the temperature at a future time step for a given power consumption trajectory. Before changing the frequency of a CPU core, its power consumption can be computed using Equation 1 and plugged to this equation to predict the resulting temperature. We implemented this predictor in Linux kernel, and validated its accuracy by comparing against actual measurements. We observed that the average prediction error is less than 3% ($1°C$) up to 1 second, while the error is within 7% ($2.5°C$) for as long as 5 seconds for the Templerun gaming benchmark, as shown in Figure 6. Further evaluation is presented in Section VI.

B. Run-time Power Budget Computation

Suppose the temperature constraint for each thermal hotspot is given by $[T_{constr}]_{N\times 1}$, where each entry gives the maximum permissible temperature. Using Equation 5, we write

$$\begin{aligned} |T[k+1]| &\leq |T_{constr}| \\ |A_s T[k] + B_s P[k]| &\leq |T_{constr}| \end{aligned} \quad (7)$$

where the $|\cdot|$ represents the norm operation. Since thermal control algorithms typically use the maximum temperature, we employ L_∞ norm and denote $|T_{constr}|_\infty = T_{max}$ to re-write the temperature constraint as

$$\begin{aligned} |A_s T[k] + B_s P[k]|_\infty &\leq T_{max} \\ max\ \{A_{s,i} T[k] + B_{s,i} P[k]\} &\leq T_{max} \quad 1 \leq i \leq N \end{aligned} \quad (8)$$

where $A_{s,i}$ and $B_{s,i}$ denote the i^{th} row of matrices A_s and B_s, respectively. Hence, we convert the matrix inequality into a set of scalar inequalities, one for each thermal hotspot. Temperature constraints can be written as:

$$B_{s,i} P[k] \leq T_{max} - A_{s,i} T[k] \quad 1 \leq i \leq N \quad (9)$$

The right hand-side is known since we obtained A_s and B_s through system identification, and measure $T[k]$. Consequently, run-time decisions are made such that $P(k)$ satisfies the power budget constraint given by Equation 9.

C. DTPM Algorithm Implementation

The proposed algorithm is incorporated with the existing governors in the Linux kernel, as explained in Section III and illustrated in Figure 2. The set of active cores and their frequencies, which are determined by the default drivers, are used to predict the power consumption and temperature, as explained in Section IV. In this work, we use a prediction interval of "1s" since it is sufficient to control the temperature of our target platform. In general, accurate predictions up to "5s" can be made, as depicted in Figure 6. If the predicted temperature exceeds the specified constraint for any one of the hotspots, the power budget is computed, as explained in Section V-B. To minimize the impact on performance, the proposed algorithm first finds the maximum feasible frequencies under the available power budget. If the power budget cannot be met with the current number of active cores, then the hottest core is put to sleep, and the tasks running on this core are migrated to the other cores by the kernel. Finally, when the power budget is so small that it cannot be satisfied even with a single big core, then all the active tasks are migrated to the little cluster and big cores are put to sleep. Moving to the little cluster and reducing the GPU frequency (if GPU is active) are used as the last resort, since they have the biggest performance impact and the migrating across clusters has a larger overhead based on our empirical evaluations. The detailed results are omitted due to space considerations.

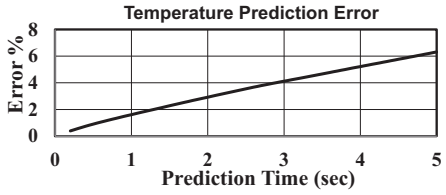

Fig. 6: Avg. temperature prediction error for Templerun game.

VI. Experimental Evaluation

A. Experimental Setup and Methodology

Experimental Platform The proposed framework is evaluated using the Odroid-XU+E platform [6] powered by Samsung Exynos 5410 MPSoC, which is a single ISA heterogeneous big.Little processor composed of a big core cluster (4 A15 cores), little cluster (4 A7 cores), a GPU, and other basic components. The Odroid platform can activate only the big or little cluster at a given time. Built-in power sensors measure the power consumption of big core cluster, little core cluster,

GPU and memory separately, while external power meters enable logging the total platform power. The platform also provides temperature sensors located on each big core which are the thermal hotspots. *All the results reported in this paper are direct measurements on this platform. Hence, the implementation overheads are included in the results.*

Benchmarks We used 15 benchmarks, 11 from the Mi-Bench embedded benchmark suite [20], 3 frequently used game and video applications and one self written matrix multiplication code, which is mainly used during debugging. The benchmarks and their relevant properties are summarized in Table I. The benchmarks are also categorized according to their comparative CPU power consumption as low, medium and high. Even if a benchmark is single threaded, there are many active threads in the system since the benchmarks run along with Android operating system and all other kernel background processes. Therefore, multiple cores were active during the experiments and this number varied dynamically. Finally, the games and video benchmarks utilized GPU, while the other benchmarks were CPU intensive.

Default configuration (With fan) We ran the benchmarks first with the default configuration of the target platform which uses a fan. The fan is activated when maximum core temperature exceeds $57°C$. Then, the fan speed is increased to 50% and 100% when the temperature passes $63°C$ and $68°C$, respectively. We emphasize that using a fan is *not feasible* when this chip is used in a smartphone or tablet, which is the case for Samsung Galaxy S4. Therefore, we evaluated two more solutions besides the proposed DTPM technique.

Without fan We disabled the fan and re-ran all the benchmarks. Since the fan is not activated when the workload is low, we observe little or no changes for light activity. However, temperature increases quickly for high loads and keeps on increasing continuously. To avoid physical damage to the device, we limited the run time to a few minutes for these workloads. We also implemented a heuristic thermal management algorithm which mimics the fan control algorithm in the default configuration. Instead of activating the increase in fan speed, this heuristic throttles the frequency by 18% and 25% when the temperature passes $63°C$ and $68°C$, respectively.

Proposed DTPM algorithm All the power and temperature models and the proposed DTPM algorithm are implemented in the Linux 3.4.76 kernel. After compiling the whole kernel with our modifications, we flashed it to the device. The kernel function implementing our models is called periodically whenever the CPU frequency driver is executed (once every 100ms). We first ran the modified kernel with the power models and thermal predictor without taking any real action to assess the power and performance overhead. We did not observe any noticeable change in power and performance due to our models.

TABLE I: The benchmarks used in the experiments.

Types	Benchmarks	Category
Security	Blowfish, Sha	Low, Medium
Network	Dijkstra, Patricia,	Low, Medium
Computational	Basicmath, Matrix Multiplication Bitcount, Qsort	High Medium
Telecomm	CRC32, GSM, FFT	Low, Medium, High
Consumer	JPEG	Medium
Games	Angry-Birds, Temple-run	High
Video	Youtube	Low

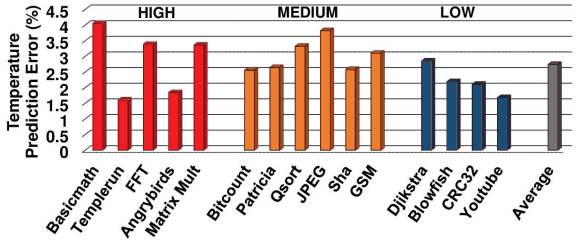

Fig. 7: Temperature prediction error for all the benchmarks.

B. Temperature Prediction Accuracy

We ran each benchmark and predicted the temperature $T[k+10]$ at every control interval $T[k]$. That is, the temperature one second (10 control intervals) ahead of time is predicted using a sliding window, and the predictions are compared to the measured values at the end of each experiment. Figure 7 shows that the average prediction error is less than 3% ($1°C$) and it never exceeds 4% ($1.4°C$). One second prediction window is selected since 10 control intervals are sufficient to regulate the temperature. We also validated that prediction windows as large as "5s" do not result in a noticeable performance impact, while the prediction error increases moderately, as depicted in Figure 6.

C. Temperature Control and Stability

The objective of the DTPM algorithm is to ensure that the temperature is regulated successfully without using a fan. To provide a fair comparison with the default configuration, we used a temperature constraint of $63°C$ which is used in the fan control algorithm. We validated that the proposed algorithm can regulate the temperature for all of the benchmarks with minimal performance impact. As representative examples, the results for Templerun and Basicmath benchmarks are shown in Figure 8 and Figure 9, respectively. First, we observe that the proposed DTPM algorithm successfully limits the temperature to the specified constraint, which is easily violated without the fan. Furthermore, the temperature variation is significantly smaller than the default solution with fan, without using a fan, and the heuristic algorithm, which is not shown for clarity. More precisely, we observe as high as $6\times$ reduction in variance for both of the benchmarks, as summarized in Table II. Superior and smoother operation is achieved since the performance is throttled *only if* a thermal violation is predicted, and *only as much as needed* with the help of precise power budgeting.

D. Power and Performance Evaluation

The proposed DTPM algorithm demotes the frequency and number of active cores only if the default values exceed the

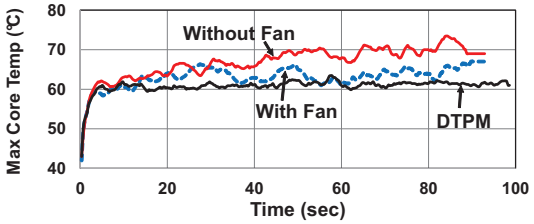

Fig. 8: Temperature control for Templerun benchmark.

TABLE II: Variance and range of temperature change.

		Without Fan	With Fan	DTPM
Variance ($^\circ C$)	Basicmath	9.42	2.68	0.48
	Templerun	12.08	6.8	1.12
Max($^\circ C$)	Basicmath	73.44	67	63.44
	Templerun	76.11	67.11	63.33
Min($^\circ C$)	Basicmath	60.66	59.55	59.44
	Templerun	59.22	58.88	57.22
Average($^\circ C$)	Basicmath	67.54	63.45	61.12
	Templerun	69.59	62.66	61.24

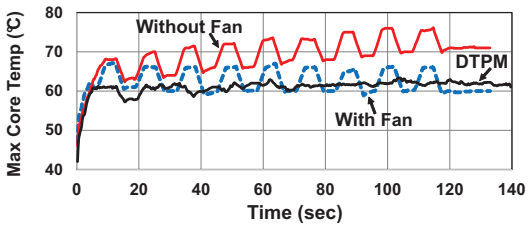

Fig. 9: Temperature control for Basicmath benchmark.

power budget. When the computation load is light, the temperature barely reaches the maximum constraint. Therefore, the proposed algorithm rarely interferes with the system and results in almost no change in performance (less than 1%), as illustrated by the low load benchmarks in Figure 10. For the same reason, there is little change in CPU power, but avoiding the fan, even if it is rarely active, results in around 3% platform power savings which corresponds to about 0.2 W.

As the computational load increases, the number of active cores and their frequencies increase. Hence, both the core and fan power consumption increase. Consequently, the power savings obtained using the proposed approach become more significant. For example, we achieve 8% power savings for medium and 14% savings for high activity benchmarks on average. It is important to note that these savings are significant since they are at the platform level. For example, 14% savings corresponds to 0.7 W savings, which would increase the lifetime of a typical smartphone battery by around 25% from 2h to 2h30m under continuous use. Despite significant power savings, the performance loss on average is only 3.3%. The performance loss hardly reaches 5% even for the most demanding applications. In contrast, the reactive DTPM algorithm that mimics the fan control results in around 20% loss in performance measured by execution time.

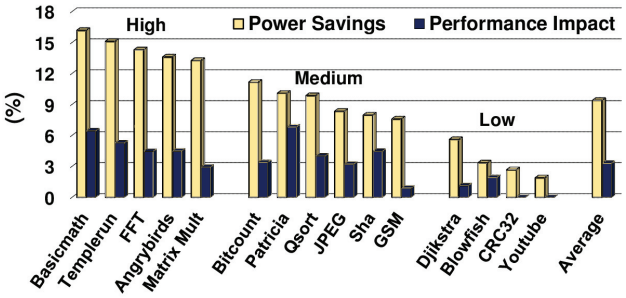

Fig. 10: Power savings and performance loss summary.

VII. Conclusion

In this paper, we presented a practical temperature prediction methodology and a DTPM algorithm for heterogeneous MPSoCs. The proposed approach calculates a precise power budget based on temperature predictions at run-time. Then, this budget is used to control the type of cores (big or little), number of active cores and frequency of the cores. Thorough experimental evaluation shows that the proposed approach not only eliminates the need for a fan, which is not a viable choice for mobile devices, but also provides significant power, thermal, and reliability advantages. In particular, it regulates the temperature more effectively than the default configuration which uses a fan, and on average offers 10% platform power savings with 3.3% loss in performance.

References

[1] Q. Xie *et al.*, "Dynamic thermal management in mobile devices considering the thermal coupling between battery and application processor," in *Proc. of ICCAD*, 2013.

[2] D. Kadjo, U. Y. Ogras, R. Ayoub, M. Kishinevsky, and P. Gratz, "Towards platform level power management in mobile systems," in *In Proc. of System-on-Chip Conf*, 2014.

[3] D. Brooks, R. P. Dick, R. Joseph, and L. Shang, "Power, thermal, and reliability modeling in nanometer-scale microprocessors," *IEEE Micro*, vol. 27, no. 3, pp. 49–62, 2007.

[4] T. S. Muthukaruppan *et al.*, "Hierarchical power management for asymmetric multi-core in dark silicon era," in *Proc. of DAC*, 2013.

[5] P. Greenhalgh, "Big. little processing with arm cortex-a15 & cortex-a7," *ARM White Paper*, 2011.

[6] ODROID − XU + E. Http://www.hardkernel.com/main/main.php.

[7] K. Skadron *et al.*, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Trans. on Arch. and Code Optimization*, vol. 1, no. 1, pp. 94–125, 2004.

[8] S. Sharifi, D. Krishnaswamy, and T. S. Rosing, "Prometheus: A proactive method for thermal management of heterogeneous mpsocs," *IEEE Trans. on CAD of Integrated Circuits and Syst.*, pp. 1110–1123, 2013.

[9] W. Huang *et al.*, "Hotspot: A compact thermal modeling methodology for early-stage vlsi design," *IEEE Trans. on Very Large Scale Integration Syst.*, vol. 14, no. 5, pp. 501–513, 2006.

[10] D.-C. Juan, S. Garg, and D. Marculescu, "Statistical thermal evaluation and mitigation techniques for 3d chip-multiprocessors in the presence of process variations," in *Proc. of DATE*, 2011.

[11] I. Yeo, C. C. Liu, and E. J. Kim, "Predictive dynamic thermal management for multicore systems," in *Proc. of DAC*, 2008.

[12] S. Murali *et al.*, "Temperature control of high-performance multi-core platforms using convex optimization," in *Proc. of DATE*, 2008.

[13] A. K. Coskun, J. L. Ayala, D. Atienza, T. S. Rosing, and Y. Leblebici, "Dynamic thermal management in 3d multicore architectures," in *Proc. of DATE*, 2009.

[14] V. Hanumaiah, S. 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 Syst.*, vol. 30, no. 11, pp. 1677–1690, 2011.

[15] F. Zanini, D. Atienza, L. Benini, and G. De Micheli, "Multicore thermal management with model predictive control," in *European Conf. on Circuit Theory and Design*, 2009.

[16] Y. Wang, K. Ma, and X. Wang, "Temperature-constrained power control for chip multiprocessors with online model estimation," in *ACM SIGARCH Comp. Arch. News*, vol. 37, no. 3, 2009, pp. 314–324.

[17] M. A. Al Faruque, J. Jahn, T. Ebi, and J. Henkel, "Runtime thermal management using software agents for multi-and many-core architectures," *IEEE Design & Test of Computers*, vol. 27(6), pp. 58–68, 2010.

[18] Y. Liu, R. P. Dick, L. Shang, and H. Yang, "Accurate temperature-dependent integrated circuit leakage power estimation is easy," in *Proc. of DATE*, 2007.

[19] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, "Leakage current mechanisms and leakage reduction techniques in deep-submicrometer cmos circuits," *Proc. of the IEEE*, vol. 91, no. 2, pp. 305–327, 2003.

[20] M. R. Guthaus *et al.*, "Mibench: A free, commercially representative embedded benchmark suite," in *Proc. of Int. Symp. on Workload Characterization*, 2001.