M-DTM: Migration-based Dynamic Thermal Management for Heterogeneous Mobile Multi-core Processors

Young Geun Kim, Minyong Kim, Jae Min Kim, and Sung Woo Chung
Dept. of Computer and Radio Communication Engineering
Korea University, Seoul, South Korea 136-713
E-mail: {carrotyone, mkim05, joist, swchung}@korea.ac.kr

***Abstract*—Recently, mobile devices have employed heterogeneous multi-core processors which consist of high-performance big cores and low-power small cores. In heterogeneous mobile multi-core processors, the conventional DVFS (Dynamic Voltage and Frequency Scaling) -based DTM (Dynamic Thermal Management) is still adopted; it does not actively utilize the small cores to resolve thermal problem. In this paper, we propose M-DTM (Migration-based DTM) for heterogeneous mobile multi-core processors. In case of thermal emergency of the big cores, M-DTM migrates applications to the small cores instead of lowering the voltage and frequency of the big cores. In this way, M-DTM allows more time for the applications to run at the highest frequency of the big cores by cooling down the big cores more rapidly, compared to the conventional DTM. Through real measurement, we show that M-DTM improves performance by 10.6% and saves system-wide energy (not CPU energy) by 3.6%, on average, compared to the conventional DTM.**

I. INTRODUCTION

With the advance of process technology, the feature size of microprocessors has significantly decreased. Thanks to the decreased feature size, the performance of microprocessors has been improved with low power consumption. However, the decreased feature size has led to the increased power density, which in turn causes thermal problems even in microprocessors of mobile devices [1].

To solve the thermal problems, several techniques have been proposed from the mechanical side, such as air cooling (a faster fan, larger heat sink, and so on) [2], liquid cooling [3], and thermoelectric cooling [4]. Though the techniques are powerful cooling solutions, they are not feasible for microprocessors of mobile devices, since mobile devices have limited space. Instead, mobile devices mostly utilize OS level DTM (Dynamic Thermal Management) to cool down their microprocessors. Specifically, they adopt the DTM that gradually lowers the voltage and frequency, when on-chip temperature exceeds the pre-defined thermal threshold.

Off-the-shelf mobile devices have employed heterogeneous multi-core processors[1] such as ARM's big.LITTLE [5], which have high-performance big cores and low-power small cores. Recent trend in heterogeneous multi-core processors is operating different types of cores at the same time. However, due to the limited software support, most heterogeneous multi-core processors in off-the-shelf mobile devices operate only one type of cores at a time. Though there are two different types of cores in heterogeneous mobile multi-core processors, the conventional DTM does not actively utilize the small cores to deal with the thermal problems. Rather, it primarily relies on DVFS (Dynamic Voltage and Frequency Scaling), as it did in homogeneous multi-core processors. When the on-chip temperature of the big cores exceeds the thermal threshold, it gradually lowers the frequency of the big cores. Only when the conventional DTM cannot lower the frequency of the big cores any more, it inevitably migrates applications from the big cores to the small cores. However, since the conventional DTM does not stop using the big cores immediately in case of thermal emergency, it does not resolve the thermal emergency rapidly. Due to the reason, with the conventional DTM, an application is provided with limited time at the highest frequency of the big cores, which eventually leads to severe performance degradation.

To tackle the problem, we propose M-DTM (Migration-based DTM) for heterogeneous mobile multi-core processors. M-DTM actively migrates applications to the small cores to cool down the big cores rapidly, as soon as the on-chip temperature of the big cores exceeds the pre-defined thermal threshold. When the on-chip temperature of the big cores decreases below the thermal threshold, M-DTM migrates the applications back to the big cores and operates the big cores at the highest frequency again. In this way, it allows more time for the applications to run at the highest frequency of the big cores, compared to the conventional DTM, leading to performance improvement of the applications. Especially, M-DTM achieves more significant performance improvement for the applications that do not severely slow down on the small cores. Since the small cores have less number of pipeline stages and smaller cache, they have shorter branch misprediction penalty and cache access latency than the big cores. Due to the reason, applications which have a relatively large number of branch mispredictions or cache accesses do not severely degrade their performance on the small cores [6]. To evaluate M-DTM, we perform our experiments based on real measurement. In our evaluation, we show that M-DTM improves performance and energy efficiency while satisfying thermal constraints.

The rest of the paper is organized as follows. In Section II, we introduce previous studies on thermal management techniques. In Section III, we explain M-DTM. In Section IV, we describe our evaluation environment. In Section V, we evaluate M-DTM in terms of performance, temperature, and system-wide energy. Finally, we conclude our work in Section VI.

[1] In this paper, we define heterogeneous multi-core processors as multi-core processors composed of the different types of cores with different performance.

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (NRF-2012R1A2A2A01013816) and KSSRC program [An Efficient Task Scheduling Technique to Improve Performance and Reduce Power & Temperature of big.LITTLE Cores in HMP]. Sung Woo Chung is the corresponding author of this paper.

II. Related Work

To resolve thermal problems in microprocessors, many techniques have been proposed [7]. In software level, techniques based on DVFS have been proposed to satisfy thermal constraints [8][9]. Lee et al. proposed a temperature-aware DVFS technique which predicts on-chip temperature with performance counters [8]. Based on the predicted on-chip temperature, their technique addresses thermal problems caused by the limited number of thermal sensors by scaling the frequency and voltage of cores. Hanumaiah and Vrudhula proposed another temperature-aware DVFS technique for hard real time applications, which scales the voltage and frequency of cores based on the power and thermal model of the system [9]. Other software level thermal management techniques have been proposed based on task scheduling [10][11]. Merkel et al. tried to balance hot tasks across the cores in symmetric multi-core processors to avoid thermal emergency [10]. Choi et al. proposed a thermal aware task scheduling technique which assigns hot tasks to multiple cores in a balanced manner and delays hot tasks to be executed later than cool tasks [11]. However, all the above techniques do not consider the heterogeneous mobile multi-core processors which have different types of cores.

For mobile multi-core processors, Kim et al. proposed a temperature-aware DVFS technique [12]. According to the Jensen's inequality [13], the average of cubic frequencies is larger than the cube of average frequency, when the variance of the frequencies is not zero. Since power consumption is roughly proportional to the frequency cubed [14], the average power consumption in different frequencies is also larger than the power consumption in their average frequency. For example, operating cores at two different frequencies (repeatedly changing between 1600 MHz and 800 MHz) consumes more power compared to operating them at their average frequency (1200 MHz). Additionally, frequency and voltage transition causes delay and energy overhead [15]. Based on the fact, their technique stabilizes frequency and voltage to the average frequency and voltage to save power and enhance performance under thermal constraints. However, their technique does not consider the heterogeneous multi-core processors.

For heterogeneous multi-core processors, several techniques have been proposed [16][17]. Sharifi et al. proposed a thermal management technique that assigns workloads to different types of cores and determines the frequency of cores based on the performance requirement of workloads [16]. To identify the performance requirement of workloads, their technique profiles the deadline of the workloads in the off-line phase. Sharifi et al. also proposed TempoMP which contains a thermal prediction model for heterogeneous multi-core processors [17]. The thermal prediction model predicts the on-chip temperature with thermal parameters, such as die thickness, convection capacitance, etc. Based on the predicted temperature and the performance requirement of workloads, it assigns workloads to different types of cores and scales the frequency of cores. However, the techniques do not actively utilize the small cores to cool down the big cores. Hence, they cannot allow much time for workloads to run at the highest frequency of the big cores in case of thermal emergency, which leads to performance degradation and energy inefficiency.

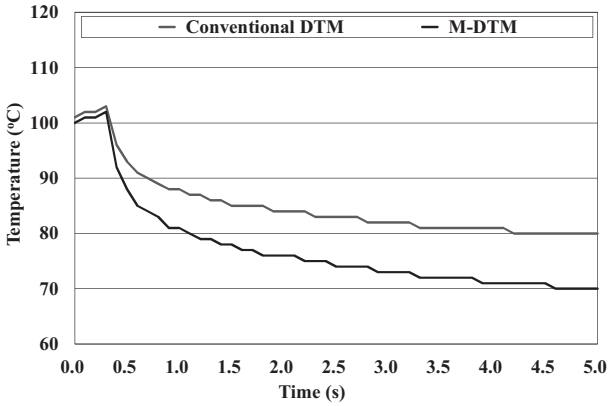


Fig. 1. On-chip temperature of the big cores with the conventional DTM and M-DTM in case of thermal emergency. Note matrix01 from the EEMBC benchmark is used for the experiment.

III. M-DTM: Migration-based DTM for Heterogeneous Mobile Multi-core Processors

A. Benefit of M-DTM

M-DTM provides more time for the applications to run at the highest frequency of the big cores by rapidly cooling down the big cores using the small cores. In case of thermal emergency of the big cores, M-DTM immediately moves applications from the big cores to the small cores and operates the small cores at the highest frequency; the low-power small cores do not incur thermal emergency in any case. Fig. 1 shows on-chip temperature of the big cores with the conventional DTM and M-DTM in case of thermal emergency. As shown in Fig. 1, M-DTM cools down the big cores more rapidly, since there is no running application on the big cores. When the on-chip temperature of the big cores decreases below the pre-defined thermal threshold, M-DTM migrates the applications back to the big cores and operates the big cores at the highest frequency again. In this way, M-DTM provides more time for the applications to run at the highest frequency of the big cores, compared to the conventional DTM. Since M-DTM always operates the big and small cores at the highest frequency, the execution time of an application with M-DTM is sum of the time when the application runs on the big and small cores at the highest frequency. Hence, the execution time of an application with M-DTM (denoted as $t_{\text{M-DTM}}$) is formulated as (1), where $t_{\text{big}}(f)$ and $t_{\text{small}}(f)$ are the time when the application runs on the big and on the small cores at frequency f; $f_{\text{big\_max}}$ and $f_{\text{small\_max}}$ are the highest frequency of the big and small cores, respectively.

$$t_{\text{M-DTM}} = t_{\text{big}}(f_{\text{big\_max}}) + t_{\text{small}}(f_{\text{small\_max}}) \quad (1)$$

Since the conventional DTM lowers the frequency of the big and small cores, it operates the big and small cores at all frequency steps. Due to the reason, the execution time of an application with the conventional DTM is sum of the time when the application runs on the big and small cores at all frequency steps. Hence, the execution time of an application with the conventional DTM (denoted as $t_{\text{conventional\_DTM}}$) is formulated as (2), where $f_{\text{big\_min}}$ and $f_{\text{small\_min}}$ are the lowest frequency of the big and small cores, respectively.

$$t_{\text{conventional\_DTM}} = \sum_{i=f_{\text{big\_min}}}^{f_{\text{big\_max}}} t_{\text{big}}(i) + \sum_{i=f_{\text{small\_min}}}^{f_{\text{small\_max}}} t_{\text{small}}(i) \quad (2)$$

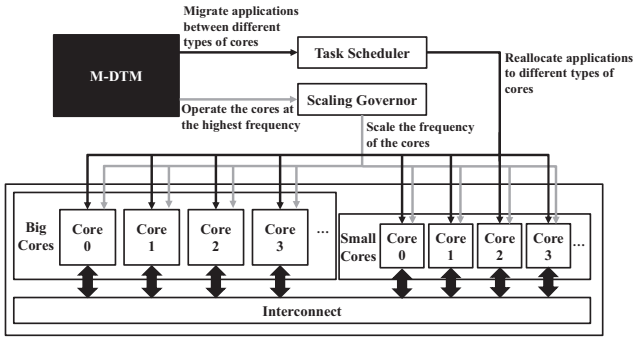


Fig. 2. Interaction between M-DTM and OS modules.

From (1) and (2), M-DTM outperforms the conventional DTM, when (3) is satisfied.

$$t_{\text{big}}(f_{\text{big\_max}}) + t_{\text{small}}(f_{\text{small\_max}}) < \sum_{i=f_{\text{big\_min}}}^{f_{\text{big\_max}}} t_{\text{big}}(i) + \sum_{i=f_{\text{small\_min}}}^{f_{\text{small\_max}}} t_{\text{small}}(i) \quad (3)$$

B. Interaction between M-DTM and OS Modules

Fig. 2 shows the interaction between M-DTM and OS (Operating System) modules in case of thermal emergency of the big cores. As shown in Fig. 2, M-DTM cooperates with a task scheduler and a scaling governor (an operating system module that scales the frequency of the cores) to deal with thermal emergency of the big cores. In thermal emergency of the big cores, M-DTM commands the task scheduler to migrate applications from the big cores to the small cores and the scaling governor to operate the small cores at the highest frequency by using shared memory. Taking the command, the task scheduler assigns the applications to the small cores through the interconnect and the scaling governor scales the frequency of the small cores to the highest frequency. When the on-chip temperature decreases below the thermal threshold, M-DTM migrates applications back to the big cores through the interconnect and operates the big cores at the highest frequency by giving commands to the task scheduler and the scaling governor.

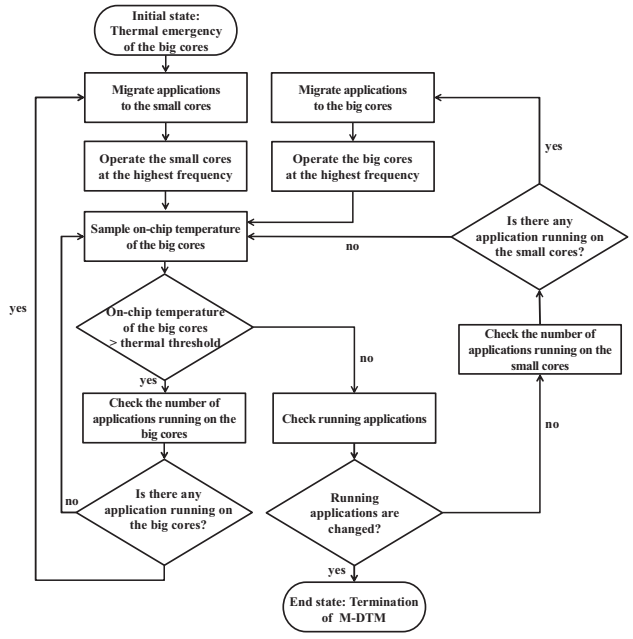


Fig. 3. Detailed Algorithm of M-DTM.

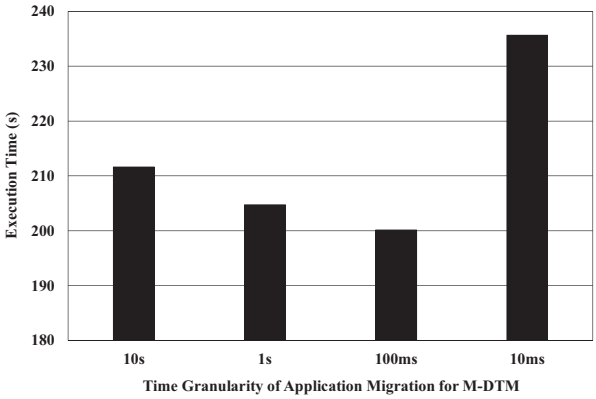


Fig. 4. Execution time of matrix01 with different time granularities of application migration for M-DTM.

C. Detailed Algorithm of M-DTM

Fig. 3 shows the detailed algorithm of M-DTM. In case of thermal emergency of the big cores, M-DTM migrates applications to the small cores to cool down the big cores. In this case, though applications usually show lower performance on the small cores, M-DTM is able to operate the small cores at the highest frequency to preserve the performance of the applications as much as possible; through experiments on off-the-shelf heterogeneous mobile multi-core processors, we found the small cores never incur thermal emergency.

M-DTM samples the on-chip temperature of the big cores. When the sampled on-chip temperature is higher than the pre-defined thermal threshold, M-DTM checks the number of applications running on the big cores. In case that there is no application running on the big cores, M-DTM samples the on-chip temperature again. Otherwise, M-DTM migrates the applications to the small cores and operates the small cores at their highest frequency again.

When the sampled on-chip temperature is lower than the pre-defined thermal threshold, M-DTM checks running applications. Unless the running applications are changed, M-DTM checks the number of applications running on the small cores. In case that there is no application running on the small cores, M-DTM samples the on-chip temperature again. Otherwise, it migrates the applications back to the big cores. In this case, M-DTM operates the big cores at the highest frequency to enhance the performance of the applications. In case that running applications are changed, M-DTM is terminated until the big cores incur thermal problem again. Note that when running applications are changed, the on-chip temperature of the big cores varies, since different applications have different thermal characteristics.

D. Determining Time Granularity of Application Migration for M-DTM

Fig. 4 shows the execution time of matrix01 with different time granularities of application migration for M-DTM. As shown in Fig. 4, the execution time of matrix01 is shortest, when the time granularity of application migration is 100 ms. In case that the time granularity is longer than 100 ms, the on-chip temperature of the big cores increases much higher than

TABLE I
EEMBC BENCHMARK APPLICATIONS FOR EVALUATION

Application	Description
iirflt01	Infinite impulse response filter algorithm
matrix01	Matrix calculations
pntrch01	Pointer chasing algorithm
puwmod01	Pulse-width modulation algorithm
rspeed01	Road speed calculation algorithm
tblook01	Table lookup algorithm
ttsprk01	Tooth-to-spark test algorithm
cjpeg	Standard compression algorithm for JPEG
djpeg	Standard decompression algorithm for JPEG
rgbcmy01	RGB-to-CMYK conversion algorithm
rgbhpg01	High pass gray-scale filter algorithm
rgbyiq01	RGB-to-YIQ conversion algorithm
ospf	OSPF Dijkstra's algorithm
pktflowb512k	Packet flow algorithm with packet size of 512Kb
routelookup	Route lookup algorithm
bezier01fixed	Bezier curve algorithm with integers
bezier01float	Bezier curve algorithm with floating points
dither01	Floyd-Steinberg error diffusion dithering algorithm
rotate01	Bitmap rotation algorithm
text01	Text parsing algorithm

the thermal threshold, since M-DTM is not able to migrate the applications to the small cores to cool down the big cores faster than the time granularity (> 100 ms). In this case, hardware DTM (where the time granularity is shorter and the thermal threshold is higher, compared to software DTM), such as throttling is invoked to lower the on-chip temperature of the big cores, leading to performance degradation. On the other hand, when the time granularity of application migration for M-DTM is shorter than 100 ms, migration overhead caused by M-DTM is bigger than performance gain from M-DTM; note that the overhead of each migration is at most 0.8 ms, in case that the application uses a large amount of cache space. Hence, M-DTM migrates applications between two different types of cores with the time granularity of 100 ms, which is also the time granularity of DVFS for the conventional DTM [12].

IV. EXPERIMENTAL ENVIRONMENT

We perform our evaluation on Odroid-XU, an off-the-shelf mobile device [18]. Odroid-XU adopts Exynos 5410 [19], which contains four high-performance cores (big cores) and four low-power cores (small cores). The processor has 9 frequency steps (from 1600 MHz to 800 MHz) for the big cores and 8 frequency steps (from 1200 MHz to 500 MHz) for the small cores. The processor in our target device operates only one type of cores between the big and small cores at a time. Odroid-XU runs on Android version 4.2.2 (Jelly-bean) and Linux kernel 3.4.5.

We evaluate M-DTM in the perspective of performance, temperature, and system-wide energy. We keep track of the on-chip temperature by using the on-chip thermal sensors[2]. To measure system-wide energy consumption, we use Monsoon Power Monitor [20], an external power measurement device. In the evaluation, we use the thermal threshold of 70.0 °C by referring to the thermal threshold of real world mobile devices; room temperature is 19.0 °C.

In our evaluation, we use 20 applications from the EEMBC [21], which is one of the most widely used embedded benchmark suite. The detailed description of each application is shown in Table I. Since each application is single-threaded, we concurrently execute four same applications to fully utilize the big or small cores.

[2] Since on-chip thermal sensors incur substantial area and power overhead, most mobile processors employ the limited number of thermal sensors to capture potential hot spot location. We define the on-chip temperature as the highest temperature from the four thermal sensors (located in each big core).

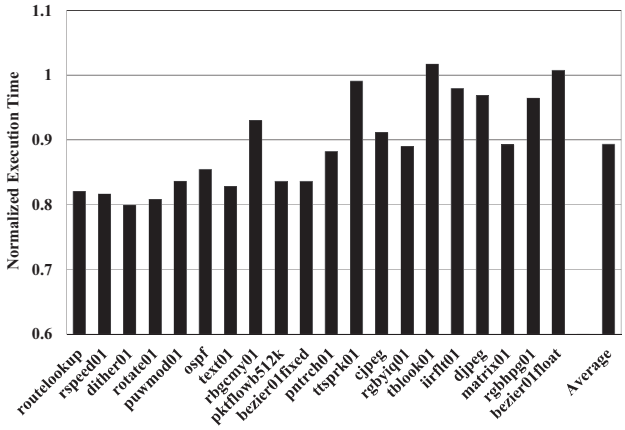


Fig. 5. Normalized execution time of M-DTM.

V. EXPERIMENTAL RESULT

A. Performance

Fig. 5 shows the execution time of applications with M-DTM normalized to that with the conventional DTM. As shown in Fig. 5, M-DTM improves the execution time of the applications by 10.6%, on average, compared to the conventional DTM. Since M-DTM rapidly cools down the big cores by using the small cores, it provides more time for applications to run at the highest frequency of the big cores (20.5% of the entire execution time of the applications, on average), as shown in Fig. 6. On the other hand, since the conventional DTM lowers the frequency of the big cores in case of thermal emergency, it provides less time for applications to run at the highest frequency of the big cores (only 6.6% of the entire execution time of the applications, on

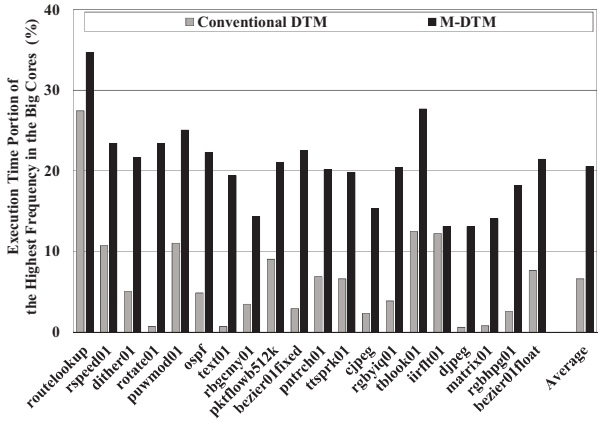
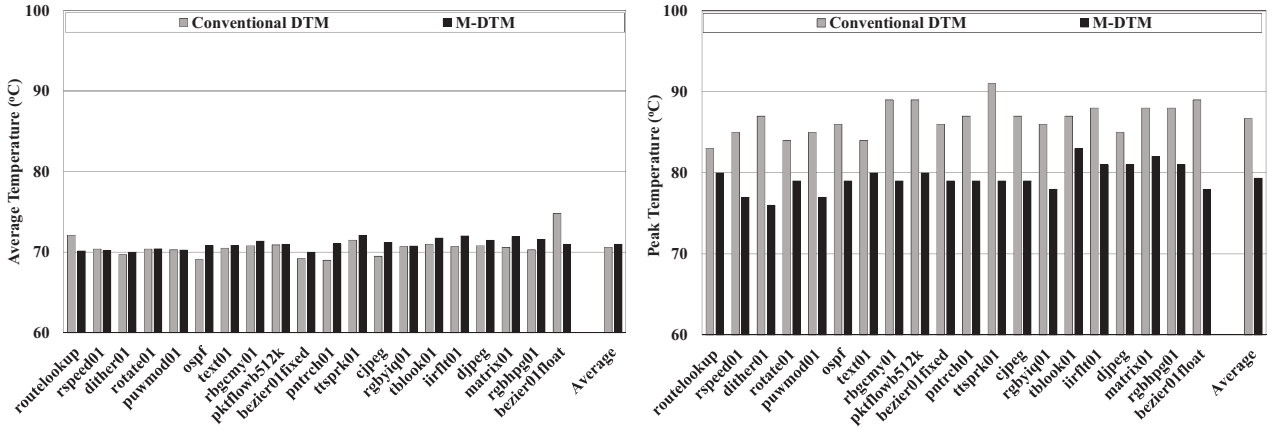


Fig. 6. Execution time portion of the highest frequency in the big cores during total execution time.



(a) Average temperature

(b) Peak temperature

Fig. 7. Average and peak temperature of the conventional DTM and M-DTM.

average), as shown in Fig. 6.

In case of left 10 applications in Fig. 5, the execution time with M-DTM is significantly shorter than that with the conventional DTM by up to 20.0%. The applications commonly have a relatively large number of branch mispredictions or cache accesses. Since the small cores have less number of pipeline stages and smaller cache than the big cores, they have shorter branch misprediction penalty and cache access latency. Due to the reason, the 10 applications, which have a relatively large number of branch mispredictions or cache accesses, face less slowdown, even when M-DTM migrates them to the small cores, compared to the rest of the applications. Hence, M-DTM achieves additional performance improvement for the 10 applications. On the other hand, in case of tblook01 and bezier01float, the execution time with M-DTM is slightly longer than that with the conventional DTM. Since the applications have a relatively small number of branch mispredictions and cache accesses, they severely get slow on the small cores. Since M-DTM deals with thermal emergency by migrating the applications to the small cores, it takes more time to run the applications on the small cores with M-DTM, compared to the conventional DTM.

B. Temperature

Fig. 7 shows average and peak on-chip temperature of the conventional DTM and M-DTM. As shown in Fig. 7(a), the average on-chip temperatures with both DTMs are near the thermal threshold (70.0 °C). However, in case of peak temperature, M-DTM shows lower peak temperature (79.3 °C, on average), compared to the conventional DTM (86.7 °C, on average) for all cases as shown in Fig. 7(b). The reason is that M-DTM more rapidly lowers the on-chip temperature of the big cores by using the small cores through migration, compared to the conventional DTM.

In Fig. 7(b), average peak temperatures of both DTMs are higher than the thermal threshold (70 °C), since the DTMs sample on-chip temperature every 100 ms. In other words, the DTMs do not take any action until the next sampling operation (for 100 ms in the worst case), even when the temperature exceeds the thermal threshold between two consecutive sampling operations. However, as soon as both DTMs invoke their DTM operation, the on-chip temperature decreases below the thermal threshold. Due to the reason, the applications do not run at the peak temperature for a long time. Note that M-DTM does not cause any reliability problems (e.g. soft error and physical damage), though the transient peak temperature exceeds the thermal threshold, since the peak temperature with M-DTM is always lower than the *critical temperature* (the temperature above which the microprocessors incur reliability problems [22] – usually set higher than the temperature that triggers hardware DTM). In our experiment, M-DTM does not invoke the hardware DTM even in the worst case (tblook01). Furthermore, M-DTM does not cause any soft error or physical damage, regardless of our repeated experiments.

C. System-wide Energy Consumption

Fig. 8 shows the system-wide (not just CPU) energy consumption of M-DTM normalized to that of the conventional DTM. As shown in the Fig. 8, M-DTM saves the system-wide energy consumption by 3.6%, on average (by up to 15.6%), compared to the conventional DTM.

M-DTM saves the system-wide energy consumption, though it dissipates larger power than the conventional DTM. Since M-DTM operates either of the big or small cores at the highest frequency during its entire operation, it consumes more dynamic power than the conventional DTM, on average. On the other hand, M-DTM consumes similar static power to the conventional DTM, since the average on-chip temperature of M-DTM is similar to that of the conventional DTM. Despite

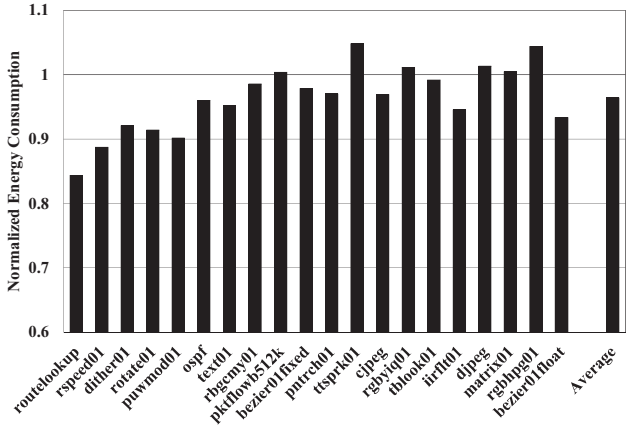


Fig. 8. Normalized system-wide energy consumption of M-DTM.

the increased power consumption, M-DTM saves the system-wide energy consumption, since it reduces the execution time of the applications (as explained in Section V.A.).

M-DTM also saves energy in the several cases where the execution time of the applications with M-DTM is slightly longer than that with the conventional DTM. In case of bezier01float, though its execution time with M-DTM is slightly longer, M-DTM saves system-wide energy, due to the lower average on-chip temperature, which leads to less static power consumption. In case of tblook01, M-DTM saves system-wide energy, despite longer execution time and higher average on-chip temperature. In this case, the conventional DTM operates the big cores at the frequency which is higher than the highest frequency of the small cores during the most time of total execution time. Hence, the conventional DTM consumes more dynamic power than M-DTM.

For some applications (pktflowb512, rgbyiq01, djpeg, and matix01), the system-wide energy consumption of M-DTM is slightly higher than that of the conventional DTM. The reason is that M-DTM shows similar execution time to the conventional DTM while consuming more dynamic power during the execution of the applications.

VI. Conclusion

In heterogeneous mobile multi-core processors, the conventional DTM of off-the-shelf mobile devices primarily relies on DVFS to deal with thermal emergency, as it did in homogeneous multi-core processors. However, since the conventional DTM does not actively utilize the small cores to cool down the big cores, it cannot provide much time for applications to run at the highest frequency of the big cores. In this paper, we propose M-DTM (Migration-based DTM) for heterogeneous mobile multi-core processors. M-DTM actively migrates applications to the small cores to cool down the big cores rapidly. As M-DTM rapidly cools down the big cores, it migrates the applications back to the big cores and operates the big cores at the highest frequency again. In this way, it allows more time for the applications to run at the highest frequency of the big cores, compared to the conventional DTM. Through the real measurement, we show that M-DTM improves performance by 10.6% and saves system-wide energy by 3.6%, on average, compared to the conventional DTM. Furthermore, M-DTM reduces peak temperature by 7.4 ^{o}C, which alleviates the dark silicon problem [23]. In addition, M-DTM shows comparable average on-chip temperature to the conventional DTM, satisfying thermal constraints.

In this paper, we evaluate M-DTM on the heterogeneous mobile multi-core processor which does not operate different types of cores at the same time. However, we believe that M-DTM will also improve performance and save energy for the future heterogeneous mobile multi-core processors supporting simultaneous use of different types of cores. For future work, we plan to develop more sophisticated algorithm for M-DTM, which migrates applications between the big cores considering the application characteristics in case that less than four applications run on the big cores.

References

[1] K. Sekar, "Power and thermal challenges in mobile devices," In *Proceedings of the 19th Annual Conference on Mobile Computing & Networking* (*MobiCom '13*), pp. 363-368, 2013.

[2] Z. Wang, C. Bash, N. Tolia, M. Marwah, and X. Zhu, "Optimal fan speed control for thermal management of servers," In *Proceedings of ASME/Pacific Rim Technical Conference and Exhibition on Packaging and Integration of Electronic and Photonic Systems* (*IPACK '09*), 2009

[3] H. B. Jang, I. Yoon, C. H. Kim, S. Shin, and S. W. Chung, "The impact of liquid cooling on 3D multi-core processors," *IEEE International Conference on Computer Design* (*ICCD '09*), pp. 472-478, 2009.

[4] J. Long, D. Li, and S. O. Memik, "Theory and analysis for optimization of on-chip thermoelectric cooling systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 10, pp. 1628-1632, 2013.

[5] ARM. Advances in big.little technology for power and energy savings [Online] Available: http://www.arm.com/files/pdf/Advances_in_big.LITTLE_Technology_for_Power_and_Energy_Savings.pdf

[6] J. M. Kim, S. K. Seo, and S. W. Chung, "Looking into heterogeneity: when simple is faster," In *2nd International Workshop on Parallelism in Mobile Platforms (PRISM-2),* 2014.

[7] J. Kong, S. W. Chung, and K. Skadron, "Recent thermal management techniques for microprocessors," *ACM Computing Survey*, vol. 44, no. 3, Article 13, pp. 1-42, 2012.

[8] J. S. Lee, K. Skadron and S. W. Chung, "Predictive temperature-aware DVFS," *IEEE Transactions on Computers,* vol. 59, no. 1, pp. 127-133, 2010.

[9] V. Hanumaiah and S. Vrudhula, "Temperature-aware DVFS for hard real-time applications on multicore processors," *IEEE Transactions on Computers*, vol. 61, no. 10, pp. 1484-1494, 2012.

[10] A. Merkel, F. Bellosa, and A. Weissel, "Event-driven thermal management in SMP systems," In *proceedings of the 2nd Workshop on Temperature-Aware Computer Systems* (*TACS '05*), 2005.

[11] J. Choi, et al, "Thermal-aware task scheduling at the system software level," In *proceedings of the 2007 International Symposium on Low Power Electronics and Design* (*ISLPED '07*), pp. 213-218, 2007.

[12] J. M. Kim, Y. G. Kim, and S. W. Chung, "Stabilizing CPU frequency and voltage for temperature-aware DVFS in mobile devices," *IEEE Transactions on Computers*, in press

[13] Wikipedia. Jensen's Inequality. [Online] Available: http://en.m.wikipedia.org/wiki/Jensen's_inequality#section_2

[14] D. M. Brooks, et al, "Power-aware microarchitecture: design and modeling challenges for next-generation microprocessors," *IEEE Micro*, vol. 20, no. 6, pp. 26-44, 2000.

[15] S. Park, et al, "Accurate modeling of the delay and energy overhead of dynamic voltage and frequency scaling in modern microprocessors," *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 5, pp. 695-708, 2013.

[16] S. Sharifi, A. K. Coskun, and T. S. Rosing, "Hybrid dynamic energy and thermal management in heterogeneous embedded multiprocessor SoCs," In *15th Asia and South Pacific Design Automation Conference* (*ASP-DAC '10*), pp. 873-878, 2010.

[17] S. Sharifi, R. Ayoub, and T. S. Rosing, "TempoMP: integrated prediction and management of temperature in heterogeneous MPSoCs," In *Design, Automation & Test in Europe Conference & Exhibition* (*DATE '12*), pp. 593-598, 2012.

[18] Hardkernel. Odroid-xu. [Online] Available: http://www.hardkernel.com/main/products/prdt_info.php?g_code=G137510300620

[19] Samsung Electronics. Exynos 5410. [Online] Available: http://www.samsung.com/global/business/semiconductor/minisite/Exynos/w/solution.html#?v=octa_5410

[20] Monsoon Solutions. Monsoon Power Monitor. [Online] Available: http://www.msoon.com/LabEquipment/PowerMonitor/

[21] EEMBC. The Embedded Microprocessor Benchmark. [Online] Available: http://www.eembc.org

[22] Q. Xie, et al, "Dynamic thermal management in mobile devices considering the thermal coupling between battery and application processor," In *Proceedings of the International Confernece on Computer-Aided Design* (*ICCAD '13*), pp. 242-247, 2013.

[23] M. Shafique, S. Garg, J. Henkel, and D. Marculescu, "The EDA challenges in dark silicon era," In *Proceedings of the 51st Design Automation Conference* (*DAC '14*), pp. 1-6, 2014.