TEEM: Online Thermal- and Energy-Efficiency Management on CPU-GPU MPSoCs

Samuel Isuwa*†1, Somdip Dey†2, Amit Kumar Singh†3, and Klaus McDonald-Maier†4
†School of Computer Science and Electronics Engineering, University of Essex, UK
Email: {1si17378, 2somdip.dey, 3a.k.singh, 4kdm }@essex.ac.uk
*Computer Engineering Department, University of Maiduguri, Borno State, Nigeria
Email: samuelisuwa@unimaid.edu.ng

***Abstract*—Heterogeneous Multiprocessor System-on-Chip (MPSoC) are progressively becoming predominant in most modern mobile devices. These devices are required to perform processing of applications within thermal, energy and performance constraints. However, most stock power and thermal management mechanisms either neglect some of these constraints or rely on frequency scaling to achieve energy-efficiency and temperature reduction on the device. Although this inefficient technique can reduce temporal thermal gradient, but at the same time hurts the performance of the executing task. In this paper, we propose a thermal and energy management mechanism which achieves reduction in thermal gradient as well as energy-efficiency through resource mapping and thread-partitioning of applications with online optimization in heterogeneous MPSoCs. The efficacy of the proposed approach is experimentally appraised using different applications from Polybench benchmark suite on Odroid-XU4 developmental platform. Results show 28% performance improvement, 28.32% energy saving and reduced thermal variance of over 76% when compared to the existing approaches. Additionally, the method is able to free more than 90% in memory storage on the MPSoC, which would have been previously utilized to store several task-to-thread mapping configurations.**

I. Introduction

Multiprocessor (many processing units or core on a single chip) System-on-Chip (MPSoC) are progressively becoming predominant in most modern computing devices ranging from mobile phones to sophisticated system. The emergence of these multiprocessor technologies is due to the successes recorded in technological advancement which made it possible for more than one processing element (core) to be integrated on the same chip [1]. Earlier multi-core systems are made up of a collection of identical cores – Homogeneous multi-core systems [2]. Modern systems now use different types of cores within a single chip (heterogeneous multi-core system) due to the evolving workloads and varied applications that we have today. These Heterogeneous Multi-core systems have proven to provide more benefits in terms of area, core(s) to application(s) matching for improved performance, power and workload coverage [2], [3]. For instance, Samsung Exynos 5422, commercially utilized in recent smartphones like Galaxy S5 [4], comprises a multi-core ARM Cortex-A15 (big) and Cortex-A7 (LITTLE) CPU clusters alongside ARM Mali-T628 MP6 GPU [4].

Despite the improvement towards efficient and versatile computation in mobile devices, thermal and power challenge remain the most significant threat to this evolution [5]. Increasingly smaller chip size and functional complexity often increase the power density which in turn leads to a higher temperature for the different core. Thermal management as

This work is supported by the UK Engineering and Physical Sciences Research Council EPSRC [EP/R02572X/1 and EP/P017487/1].

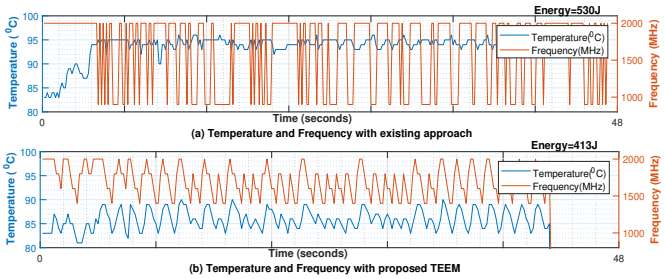

Fig. 1. (a) Existing vs (b) Proposed Approach

oppose to energy saving approach ensures that the temperature of the chip does not go beyond the thermal limit. In most of these approaches, throttling of various devices that have reached temperature limit is performed to tackle the thermal issues [5]. However, this approach tends to be reactive as often in time the thermal limits for these devices are reached or exceeded before the throttling occurs. This inefficient technique often leads to sizeable temporal temperature variation, which not only affects the performance but also impairs on the reliability of the device [1], [6]–[8].

Even though thermal and energy-aware mapping involving CPU and GPU has been studied by [9], energy and performance are often traded-off for improved temperature behavior and vice-versa. Also, the approach only considers design-time (offline) optimization. Therefore, there is a need for a smart approach to efficiently handle the trade-off between power, performance and temperature.

This paper proposes an online thermal- and energy-efficiency management (TEEM) mechanism for executing applications in CPU-GPU mobile MPSoC. The work focuses on the multithreaded applications because of the need to concurrently exploit the CPU and the GPU cores of the heterogeneous SoC during execution.

A. Motivational case study

An experimental case study is used to demonstrate the advantages of the proposed TEEM approach. Fig. 1(a) illustrates execution by a typical Linux ondemand thermal management technique in Exynos 5422 while running COVARIANCE application from the Polybench benchmark suite on 2L+3B mapping (two LITTLE and three Big cores of the CPU cluster) and GPU cluster at partition 1024 (workloads shared evenly across the CPU-GPU cores). The figure clearly shows the temperature of core-6 (the core with the highest temperature within the big CPU cluster) and the throttling of the big CPU cluster, i.e. the stepping down of the frequency level of the Cortex A15-Cores from 2000 MHz to 900 MHz to reduce the dissipation of power and consequently reduction in

temperature. This is done whenever the temperature exceeds or reaches the thermal limit, in this case, 95°C. This technique often infringes on the performance of the Mobile SoC as the device might end up working at lower frequency mode (900 MHz as against 2000 MHz) thus negatively affecting the need for incorporating the powerful core initially. Also, the variable temperature spread over time as a result of the throttling also impact on the reliability of the device. It can also be seen in Fig. 1(a) that the Linux ondemand has an execution time of 48seconds, average temperature of 93.7°C and energy consumption of 530J (Joules) with a peak operating temperature of 96°C.

In contrast, rather than allowing the SoC to reach its thermal limit and thereby throttling to the much lower frequency (900 MHz), the proposed approach maintains the SoC at a much lower operating temperature by setting a lower threshold (85°C in this case). This was achieved by progressively stepping down the frequency of the A15 cluster, whenever the temperature reaches the threshold. This frequency scaling was possible because the cluster supports multiple frequency levels which ranges from 200 MHz to 2000 MHz at a step of 100 MHz [4]. Our systematic approach as shown in Fig. 1(b) resulted in lower execution time of 39.6seconds, an average temperature of 85.8°C and reduced energy consumption of 413J with a peak temperature of 90°C. Therefore, efficient thermal management was achieved without trading-off performance or energy efficiency, thus, translating into more reliable system with more extended battery life and improved performance.

In implementing the proposed approach, this paper makes the following novel contributions:

1. Detailed evaluation of the CPU and GPU characteristics while running different applications at varying frequency setting on Samsung Exynos 5422 MPSoC with consideration on the temperature, energy consumption and performance constraint.
2. A model is developed from the obtained characteristics using linear regression in R to determine the mapping and fraction of application workload on the CPU-GPU cores while satisfying the requirement.
3. An online thermal- and energy-efficient mechanism that meets performance requirement.
4. Implementation of the mechanism on real-life mobile MPSoC present in the Odroid-XU4 board and evaluation using different applications.

II. Related Work and Problem Formulation

Energy saving mechanisms within performance constraint on heterogeneous MPSoC have been considered by authors in [10]–[12], [29]. Even though the results obtained show energy saving, the approaches can only be applied on Single-ISA heterogeneous MPSoC thus cannot be used for application running on CPU-GPU cores with different ISA. Furthermore, authors in [13] studied performance and energy saving in scheduling priority-constrained real-time applications on multiprocessor systems applying DVFS. However, the approach did not consider different applications and the heterogeneity of the microprocessors in minimizing the energy consumption.

Conversely, authors in [3], [14]–[16] examined energy-efficient mapping and partitioning of applications on heterogeneous platforms using CPU-GPU SoCs. [16] considers partitioning of application exploring different domains, different ISA and operating system on an extremely heterogeneous platform. Authors in [14] achieve energy efficiency by focusing on remapping and migration of task in multi-application scenario. Authors in [3] and [15] employed OpenCL to explore the simultaneous CPU and GPU exploitation for an application. [3] proposed static partitioning of applications through optimum power-performance trade-off of the CPU and GPU clusters by careful exploitation of their performance and functional heterogeneity. Similarly [15] considers run-time mapping and thread partitioning of application within performance constraints of the applications. Although the works above achieved energy saving, temperature behaviour which may not only lead to degraded system performance but also reliability issues, was not considered.

Proactive thermal management in MPSoC that uses predictors was proposed by [17], [18]. The authors in [17] predict temperature and modify the task allocation. Reduction of temperature variance and thermal hot spots were achieved with little performance cost. Similarly, authors in [19] proposed a runtime thermal management that helps in reducing hot spot, average thermal cycle and spatial gradient. The approaches achieved significant improvement compared to other reactive methods. However, only homogeneous MPSoC was considered.

Design-time thermal optimization in single core systems using DVFS was pioneered by [19]. Equally, the work was further extended by [20], [21] employing real-time applications on heterogeneous microprocessor systems. Authors in [20] considers only independent real-time workloads. Authors of [21] on the other hand, consider energy efficient approach in different real-time applications with priority constrained. The approach is especially suitable for heterogeneous multi-scale systems. However, OpenCL which supports concurrent exploitation of CPU and GPU cores was not employed.

Run-time temperature prediction and dynamic thermal and power management approach for single-ISA heterogeneous MPSoC was presented by [7]. Also, run-time thermal and energy efficient mapping using OpenCL on heterogeneous MPSoCs was considered by [9]. The approach in [9] does not apply any online procedure to optimize the temperature, energy and or the performance of the MPSoC when the behavior of the cores change.

Summarily, majority of the works in this field consider energy optimization without thermal consideration [3], [10]–[16] which not only affects the reliability and lifespan of the system but also negatively affects the performance as well. The few works on heterogeneous MPSoC [9] do not employ online optimization process when the behavior of the cores change which often leads to a substantial difference in temperature variance that affects the reliability of the SoC.

In contrast, this paper focuses on thermal management process which increases reduction in thermal gradient as well as energy-efficiency through resource mapping and thread-partitioning of application workloads with online optimization in heterogeneous MPSoCs.

A. Problem Formulation

Given: User requirement, T_{REQ} (time required) and AT (average temperature), for a performance constrained application execution on a heterogeneous MPSoC using CPU-GPU cores that support dynamic voltage and frequency scaling.

Find: Model that determines the mapping and fraction of application workload on the CPU-GPU cores of the Mobile SoC using linear regression in R. Also, an online thermal- and energy-efficient mechanism.

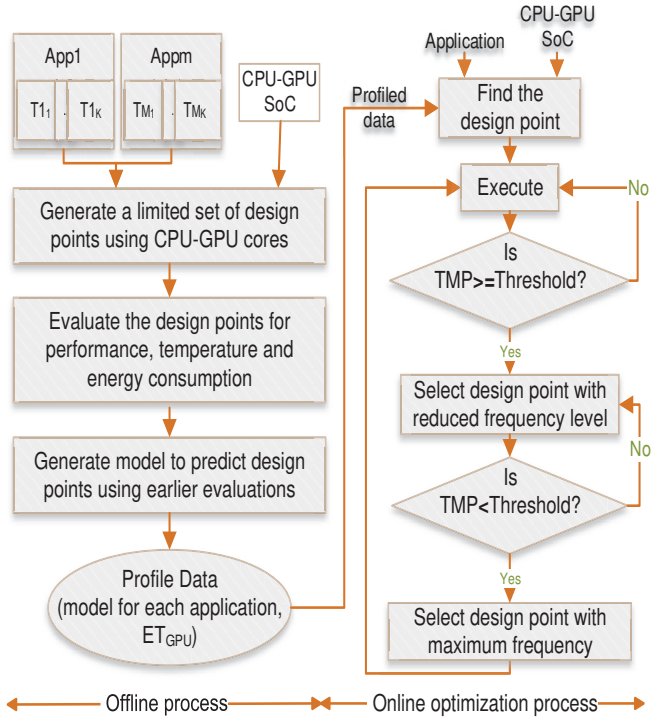

Fig. 2. Proposed online Thermal and Energy-Efficiency Management (TEEM)

Subject to: Meeting the performance- and thermal-constraints of each application within the SoC available resources.

III. Proposed Approach

Fig. 2 shows an outline of the runtime (online) thermal- and energy-efficiency management (TEEM) approach to execute applications in CPU-GPU MPSoC. The approach has offline and online procedures. Details of the procedures are as follows.

A. Offline Process

1) Generation of design points using CPU and GPU cores: For each application App1 to Appm, a set of mapping and partition of work-items between CPU and GPU is determined as design points. Equation (1) is used to determine the number of mappings for the considered platform on the CPU clusters (M_{CPU}) [15]. Similarly, only one possible mapping is obtainable on the GPU cluster.

$$M_{CPU} = N_b + N_L + (N_b \times N_L) \quad (1)$$

Where N_b and N_L represent the number of cores on the big and LITTLE CPU clusters, respectively. However, Exynos 5422 supports cluster-wise voltage-frequency scaling, hence using the different frequency setting for the big (19), LITTLE (13) and GPU (7) clusters, the maximum design points (M_{DP}) for the appraised MPSoC was obtained using (2) [15].

$$M_{DP} = \{(N_b \times F_b) + (N_L \times F_L) + (N_b \times F_b \times N_L \times F_L)\} \times \{1 \times F_g\} \quad (2)$$

Where F_b, F_L, and F_g represent the frequencies of the big, LITTLE and the GPU cores respectively.

For each of the mapping generated, nine different partitions of work-item are also generated. The fraction of the application workload is varied from 0 to 1 on the CPU cores with partition 0 representing all the workloads on the GPU, and partition 1 denotes all the workloads are executed on the CPU cores. Other intermediate values across different grains to represent the various fraction of workloads on the CPU and GPU include 1/8, 2/8 (1/4), 3/8, 4/8 (1/2), 5/8 (10/16), 6/8 (3/4), and 7/8. Therefore, the design points considered include the different mapping as well as the partition of work-items on the CPU and GPU cores. However, not all the 28,560 mappings (obtained from (1)) × 9 partitions, which equates to 257,040 design points were considered. Specifically, 10,368 design points that cover a diverse mapping represented as used big and LITTLE cores and various partitions were used.

2) Evaluation of design points: For the design points (D) considered, the metrics, i.e. temperature profile, energy consumption, frequency and execution time were obtained.

The execution time (ET) is governed by the cluster (CPU or GPU) taking more time as shown using (3) or (4).

$$ET = \max\{WG_{CPU} \times ET_{CPU}, (1 - WG_{CPU}) \times ET_{GPU}\} \quad (3)$$

$$ET = \max\{WG_{CPU} \times ET_{CPU}, WG_{GPU} \times ET_{GPU}\} \quad (4)$$

Where WG_{CPU} and WG_{GPU} are the fractions of workload on the CPU and GPU cores respectively, ET_{CPU} and ET_{GPU} represent the approximated execution times when workload is executed on only CPU and only GPU respectively.

The temperatures for the design points are determined with the help of the temperature sensors on the big CPU and GPU Clusters. At each point in time, the highest temperature value was taken for the two clusters (big and GPU) and both the peak and average of the peak-temperatures over time were evaluated.

The frequency of the different clusters was also captured at finer grains to evaluate the effect of throttling on each of the clusters. It was observed that only the frequency of the big cluster A15 core was affected by the Linux ondemand thermal management whereas the LITTLE and GPU clusters are not affected. For each of the different frequency/voltage settings, the changes in each of the metrics were captured, compared and evaluated.

The power readings were taken using Odroid smart power 2. The value read include voltage, current, power (Watt) and kWh. The power (Watt) is multiplied by the execution time of the application to determine the actual energy consumption in Joules of the MPSoC.

3) Model to predict the mapping and partition: Linear regression in R was used to determine the model. Fig. 3 shows the preliminary analysis of the data using scattered matrix plot to visualize the relationship between the multi-predictor variables.

For the multiple linear regression model in this work, (5) is the consideration:

$$M = \beta_0 + (\beta_1 \times AT) + (\beta_2 \times ET) + (\beta_3 \times PT) + (\beta_4 \times EC) \quad (5)$$

Where M represents the mapping for the big and LITTLE cores, AT and PT are the average and peak temperatures respectively, ET represents the execution time and EC represents the energy consumption. M is the response variable while AT, PT, ET and EC are the multiple-predictor variables generated using different application when varying the mapping from 1L+1B to 4L+4B. The intercept β_0 and the slopes: β_1, β_2, β_3 and β_4 of each predictor is estimated by the model with all the predictor variables as shown in Table I.

From the results of Fig. 3 and Table I, the following explanations can be given:

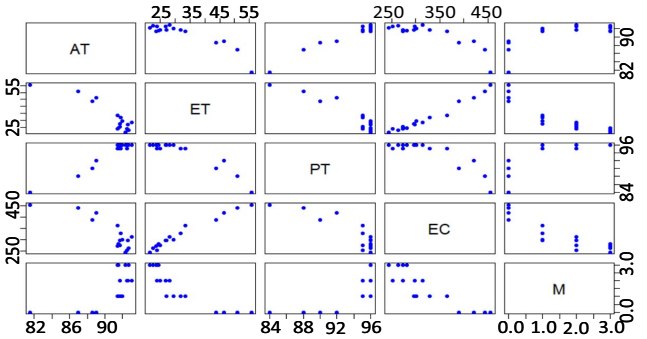

Fig. 3. Matrix scatterplot of respond and predictor variables

• For any time specified by the user, there will be an average reduction in the number of used big.LITTLE cores by -0.33165. Similarly, for any given average temperature, and peak temperature there will also be a reduction in the number of cores by -0.22377 and -0.20229 respectively. Only energy consumption will lead to increase in the mapping by 0.02017.

• It can also be noticed that energy consumption, average and peak temperature values have insignificant p-values (Probability). From Fig. 3, it is clear the pattern mapping takes when regressed on average temperature and peak temperature; they are closely associated with each other, likewise energy consumption and execution time. Therefore, when combined in a model, they masked (become insignificant after adjusting to) each other. This often results in collinear problem whenever two or more predictors are strongly correlated. To avoid this, peak temperature and energy consumption are dropped, and the model is rerun.

After running the model with average temperature and execution time as the predictor variables, a very strong level of significance is observed. However, there is little or no improvement in adjusted R-squared value and the residual error probably due to outlier points. Hence, the response variable was transformed using logarithm to base ten to fit the model better.

Table II shows the output of the modified model using logarithm. The model reflects an improved F-statistic (ratio of mean regression to the mean error, the higher the value, the better the result) from 40.66 to 76.71 on 2 and 13 degrees of freedom, improved multiple and adjusted R-squared values of 0.92 and 0.91 respectively.

The transform residual plot shows a relatively good fit with the data randomly scattered as shown in Fig. 4.

Thus, (5) now becomes:

$$\log(\mathrm{M}) = \beta_0 + (\beta_1 \times \mathrm{AT}) + (\beta_2 \times \mathrm{ET}) \quad (6)$$

In summary, the model for predicting the mapping to be used given user's requirement, i.e. average temperature and time has been generated. This solves the issue raised in [15] regarding the huge number of mappings that has to be generated and stored. However, for each application, the model has to be adjusted in order to fit properly as each application behaves differently on the CPU and GPU clusters.

4) Determination of the Fraction of Workloads on the CPU and GPU: To determine the fraction of application workloads to be executed on the CPU and the rest on the GPU or vice versa, (3) and (4) were modified to obtain (7) and (8) for partition of work-items and fraction of workload respectively.

$$\mathrm{ET} = \mathrm{WG_{CPU}} \times \mathrm{ET_{CPU}} \quad (7)$$

$$\mathrm{ET} = (1 - \mathrm{WG_{CPU}}) \times \mathrm{ET_{GPU}} \quad (8)$$

Where ET is the time taken for the application to complete its execution. Since $\mathrm{ET_{GPU}}$ is independent of the mapping used, (8) can be transformed to determine the fraction of workloads on the CPU by changing ET to $\mathrm{T_{REQ}}$ (time required for the application to be executed) which is specified by the user. Equation (8) then becomes:

$$\mathrm{WG_{CPU}} = 1 - (\mathrm{T_{REQ}}/\mathrm{ET_{GPU}}) \quad (9)$$

Equation (9) is used to determine the fraction of workload on the CPU provided $\mathrm{T_{REQ}}$ is less than $\mathrm{ET_{GPU}}$. This is logical because there is no advantage in exploring the heterogeneity of the cores if better performance can be achieved with single-ISA heterogeneous SoC, i.e. either GPU or CPU.

TABLE I
FITTING THE MODEL WITH ALL THE PREDICTOR VARIABLES

Residuals:				
min	**1Q**	**median**	**3Q**	**max**
-0.9163	-0.2417	0.1544	0.3471	0.5374

Coefficients:	**Estimate**	**std. Error**	**t-value**	**Pr(>—t—)**
(Intercept)	44.91228	11.24681	3.993	0.00178**
AT	-0.22377	0.13789	-1.623	0.13060
ET	-0.33165	0.12015	-2.760	0.01727*
PT	-0.20229	0.21002	-0.963	0.35446
EC	0.02017	0.01398	1.443	0.17467

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.5 '.' 0.1 ' ' 1
Residual Standard error: 0.4802 on 12 degrees of freedom
Multiple R-squared: 0.8749, Adjusted R-squared: 0.8332
F-statistics: 20.98 on 4 and 12 DF, p-value: 2.396e-05

TABLE II
THE TRANSFORMED MODEL

Residuals:				
min	**1Q**	**median**	**3Q**	**max**
-0.34646	-0.08481	0.01754	0.11249	0.22634

Coefficients:	**Estimate**	**std. Error**	**t-value**	**Pr(>—t—)**
(Intercept)	10.099046	3.124188	3.233	0.00654**
AT	-0.079174	0.031680	-2.499	0.02663*
ET	-0.065991	0.008635	-7.642	3.68e-06***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.5 '.' 0.1 ' ' 1
Residual standard error: 0.1614 on 13 degrees of freedom
Multiple R-squared: 0.9219, Adjusted R-squared: 0.9019
F-statistics: 76.71 on 2 and 13 DF, p-value: 6.348e-08

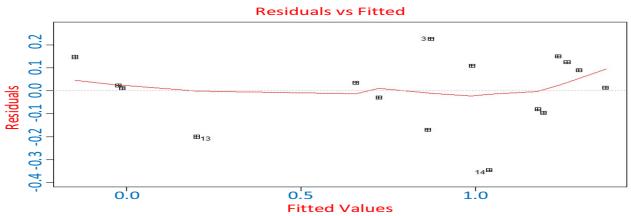

Fig. 4. Residual plot for the transformed model

B. Online Optimization Process

At runtime, the design point D is determined given the required timing ($\mathrm{T_{REQ}}$) and average temperature (AT) as the constraints using the developed model. The fraction of application workload ($\mathrm{WG_{CPU}}$) on the CPU is also determined using (9) and the $\mathrm{ET_{GPU}}$ stored from the offline process. Thereafter, the application is scheduled to execute at maximum frequency for all the clusters (big.LITTLE and GPU).

During execution, the temperature of the cores (big and GPU) are monitored continuously using the onboard temperature sensors to ensure the threshold set is not exceeded. When the threshold temperature is reached, the frequency level of the A15 core is reduced by a delta (δ) value (200 MHz in this case) and the design point D is selected as

shown in Fig. 2. If the temperature is still equal to or greater than the threshold, further reduction is done by δ, but not below 1400 MHz. 1400 MHz was used due to the observation made while evaluating the effects of various frequencies on performance and temperature. The constant selection of D enables a progressive reduction in the frequency level to be achieved. However, if at any point the temperature is observed to be below the threshold, D with maximum frequency is selected to avoid infringing the performance.

IV. Experimental Setup and Evaluation

A. Experimental setup

1) Hardware Infrastructure: The proposed approach was implemented on Odroid-XU4 development platform [4] that employs Exynos 5422 MPSoC from Samsung. The Exynos 5422 integrates CPU clusters built on ARM's big.LITTLE [23] multi-core architecture and Mali-T628 GPU cluster core on a single chip. The LITTLE CPU cluster comprises of quad-core Cortex-A7 capable of operating at a cluster-wise frequency of 200 MHz to 1400 MHz at discrete intervals of 100 MHz. Similarly, the big cluster consists of powerful ARM Cortex-A15 quad-cores which can be clocked from 200 MHz to 2000 MHz at the same discrete interval of 100 MHz [3], [4], [24]. The GPU cluster, on the other hand, is made up of Mali-T628 MP6 GPU, which consist of 6 shader cores with a clock speed of up to 600 MHz.

Odroid smart power 2 capable of powering single-board devices with power requirement ranging from 4 to 5.3 volts and 5 amp in 100 mV step [25] was used to monitor the power consumed by the SoC. The Wi-Fi enabled access board captures the voltage (volt), current (Ampere), power (watt) and energy consumed (kwH) at default sample rate of 1 Hz (1sec).

2) Software Infrastructure:: The Odroid-XU4 platform runs Ubuntu Linux Kernel 16.04.1 LTS that supports heterogeneous multi-processing. OpenCL and FreeOCL that allows for the full concurrent exploitation of the ARM CPU and Mali GPU were also employed for this work.

The evaluation of TEEM considers applications from the Polybench [26] benchmark suite developed for multicore CPUs, GPU and other accelerators. The applications used cut across the wide application domain of the benchmark suite [27], [28]. For instance, data mining class applications such as correlation (CR) and covariance (CV). Others include the linear algebraic kernels 2MM (2M) and MVT (MV), basic linear algebraic routines GEMM (GE), SYRK (SR) and SYR2K (S2) and the stencils such as 2D-convolution (2D).

B. Evaluation

The proposed approach termed as TEEM was compared with the existing approach of [15] and [9], to demonstrate energy optimization, reduced temperature gradient, performance enhancement, and memory optimization. The approach in [9] performs temperature-aware mapping and partitioning of applications on GPU or CPU-GPU based on some performance and energy trade-off. The approach is known as Reliable Mapping and Partitioning (RMP). In RMP, if better temperature behavior can be obtained by running all the application on GPU with minimal performance trade-off, then the application is mapped on only the GPU, else the partition of work-items on the CPU and GPU cores with minimal performance infringement is determined. Comparing with this shows the potential of the online performance- and thermal-constrained optimization without a trade-off in any of the metrics.

The approach of [15] implements energy-efficient mapping and thread partitioning (EEMP) of applications. In EEMP, suitable voltage/frequency for the cores used is determined via DVFS. The dynamic power management in this means executing at the maximum voltage/frequency and turning off the unused cores. Comparing with this work is imperative to demonstrate the impact of energy optimization approach without thermal consideration.

V. Results and Discussion

A. Energy Optimization

Fig. 5(a) presents the bar graphs of the energy optimization comparison of TEEM with EEMP and RMP for some applications from the Polybench benchmark on mapping 2L+4B. The figure shows the individual energy consumption in Joules when different applications where used for EEMP, RMP and TEEM approaches respectively. The thermal-constrained threshold used for these plots is 85^{o}C. The 85^{o}C threshold was chosen after several results with varying temperature threshold values were explored and evaluated. For instance, they have either high overheads (for high threshold) due to several frequency changes at runtime or miss performance improvement opportunities for low threshold value. This threshold however, shows consistently improved results for all the metrics when different mappings are used.

It can be observed for 2D and GM the overhead is 18.81% and 30.36% respectively when compared to RMP. This is because only the GPU cluster was used for these applications in the RMP approach. However, this is not always the case, in SR application for instance where the GPU was also used in the RMP approach, the proposed approach shows an energy saving of 47.28%. For all the considered cases, an average of 28.32% (125J) and 13.97% (72J) of energy optimization was achieved by the proposed approach when compared to EEMP and RMP respectively. This further attests to the fact that energy-saving approach without thermal consideration can lead to an excessively high temperature [5] which could affect performance and consequently, the energy as established in the results of TEEM vs EEMP. Similar behavior was achieved when different mappings are used.

B. Reduction in Thermal Gradient

Fig. 5(b) shows the temperature variation when the various approaches are employed. It is worthy to note that while the temperature variation in EEMP and RMP is large, the proposed approach closely maintains the temperature around the threshold (i.e. 85^{o}C) which results in significant reduction in the thermal gradient over time as shown in the figure. Even though RMP tries to reduce the temperature variation, the proposed TEEM approach is comparably better than the approach with an average thermal gradient reduction of 76% and 45% when compared to EEMP and RMP respectively. The proposed approach also restrict the peak temperature of the chip within the constrained threshold, unlike the EEMP and RMP where the thermal limit of the chip is often reached. These results vividly portray the potential of thermal-constrained management as it mitigates the sharp temperature variation which affects the reliability of the chip. Similar results are also obtained when different mappings were considered e.g. for 2L+3B, 84% and 64% reductions were achieved for EEMP and RMP respectively.

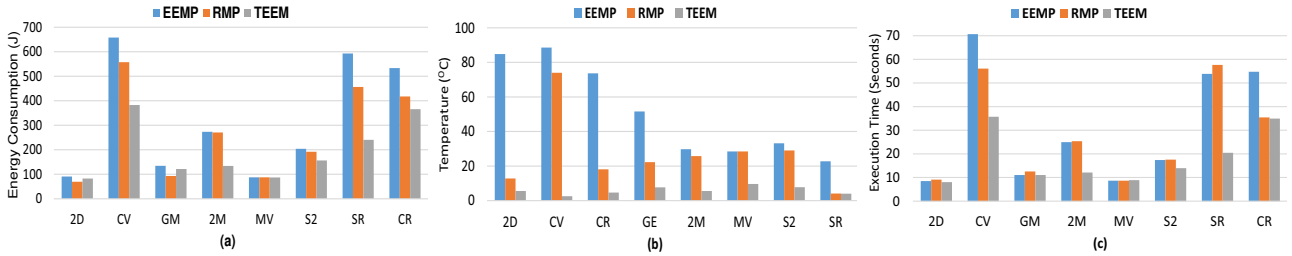

Fig. 5. (a) Energy Consumption, (b) Temperature and (c) Execution Time of different applications when employing various approaches

C. Performance Improvement

In order to substantiate the flexibility of this approach with respect to performance, the results obtained are compared to EEMP and RMP approaches that considered performance in their approaches.

Fig. 5(c) shows the the bar graphs of the performance comparison for some applications from the Polybench benchmark with the best partition of application workload for each of the approaches with 2L+4B mapping. The result from Fig. 5(c) shows that significant improvement in performance metric was achieved for almost all the applications used when compared to EEMP and RMP respectively. It is noteworthy that the proposed approach achieved an average of about 28% and 24% performance improvement when compared to EEMP and RMP respectively. Equally, similar results are obtained with different mappings.

D. Memory Optimization

Given the fact that embedded systems are designed within tight resource constraint, optimum resource utilization is required. For instance, using the approach of [15], 128 design points will need to be generated and stored in the memory for each application which leads to lots of memory overhead if different applications are involved. However, with this approach, only the different models for each application and the GPU execution time (ET_{GPU}) are stored. This gives a total of 2 items compared to 128 items when EEMP approach is used. Hence an overall memory saving of 98.8% is achieved when the proposed approach is employed.

VI. CONCLUSION

This paper proposed an online thermal- and energy-efficiency management (TEEM) of OpenCL applications in CPU-GPU heterogeneous mobile MPSoC. In this work, an exhaustive evaluation of the CPU and GPU temperature, energy and performance characteristics while running different applications at varying frequency setting was carried out. A model was developed using machine learning approach in R to determine the mapping and fraction of workload within user's requirement. Then an online thermal-constrained process that considerably controls the temperature within a threshold is proposed. The proposed approach was evaluated on real-life mobile MPSoC with different applications from Polybench benchmark suite. The results show optimization of energy consumption, peak temperature, performance and memory overhead when compared to the state-of-the-art.

REFERENCES

[1] A. Prakash et al., "Improving mobile gaming performance through cooperative CPU-GPU thermal management," DAC'2016.

[2] T. Mitra, "Heterogeneous Multi-core Architectures," IPSJ Tran. on System LSI Design Methodology, vol. 8, pp. 51-62, 2015.

[3] A. Prakash et al., "Energy-Efficient Execution of Data-Parallel Applications on Heterogeneous Mobile Platforms," IEEE ICCD'15, pp. 208-215.

[4] (2017) Odroid-XU4 Manual. Odroid Magazine. Available: https://magazine.odroid.com/odroid-xu4

[5] K. Sekar, "Power and thermal challenges in mobile devices," presented at the Proceedings of the 19th annual international conference on Mobile computing & networking, Miami, Florida, USA, 2013.

[6] D. Brooks et al., "Power, thermal, and reliability modeling in nanometer-scale microprocessors," IEEE Micro, vol. 27, pp. 49-62, May-Jun 2007.

[7] G. Singla et al., "Predictive Dynamic Thermal and Power Management for Heterogeneous Mobile Platforms," DATE'2015, pp. 960-965, 2015.

[8] Q. Xie et al., "Dynamic Thermal Management in Mobile Devices Considering the Thermal Coupling between Battery and Application Processor," 2013 IEEE/ACM ICCAD, pp. 242-247, 2013.

[9] E. W. Wachter et al., "Reliable mapping and partitioning of performance-constrained OpenCL applications on CPU-GPU MPSoCs," presented at the Proceedings of the 15th IEEE/ACM Symposium on Embedded Systems for Real-Time Multimedia, Seoul, Republic of Korea, 2017.

[10] B. K. Reddy et al., "Inter-cluster Thread-to-core Mapping and DVFS on Heterogeneous Multi-cores," IEEE Tran. on Multi-Scale Computing Systems, pp. 1-1, 2017.

[11] B. Donyanavard et al., "SPARTA: Runtime Task Allocation for Energy Efficient Heterogeneous Many-cores," (Codes+ISSS), 2016.

[12] V. Chau et al., "Energy Efficient Job Scheduling with DVFS for CPU-GPU Heterogeneous Systems," presented at the Proceedings of the Eighth International Conference on Future Energy Systems, Shatin, Hong Kong, 2017.

[13] M. E. T. Gerards et al., "On the interplay between global DVFS and scheduling tasks with precedence constraints," IEEE Tran. on Computers, vol. 64, pp. 1742-1754, 2015.

[14] A. Goens et al., "TETRiS: a Multi-Application Run-Time System for Predictable Execution of Static Mappings," presented at the 20th SCOPES, Sankt Goar, Germany, 2017.

[15] A. K. Singh et al., "Energy-Efficient Run-Time Mapping and Thread Partitioning of Concurrent OpenCL Applications on CPU-GPU MPSoCs," ACM Trans. Embed. Comput. Syst., vol. 16, pp. 1-22, 2017.

[16] K. Chandramohan and M. F. P. O'Boyle, "Partitioning Data-parallel Programs for Heterogeneous MPSoCs : Time and Energy Design Space Exploration," Acm Sigplan Notices, vol. 49, pp. 73-82, May 2014.

[17] A. K. Coskun et al., "Proactive temperature balancing for low cost thermal management in MPSoCs," in ICCAD'08, pp. 250-257.

[18] A. K. Coskun et al., "Utilizing Predictors for Efficient Thermal Management in Multiprocessor SoCs," IEEE TCAD, vol. 28, pp. 1503-1516, Oct 2009.

[19] Y. Liu et al., "Thermal vs Energy Optimization for DVFS-Enabled Processors in Embedded Systems," in 8th ISQED'2007, pp. 204-209.

[20] S. Saha et al., "Thermal-Constrained Energy-Aware Partitioning for Heterogeneous Multi-core Multiprocessor Real-Time Systems," in IEEE RTCSA, 2012, pp. 41-50.

[21] J. Zhou et al., "Thermal-aware correlated two-level scheduling of real-time tasks with reduced processor energy on heterogeneous MPSoCs," Journal of Systems Architecture, vol. 82, pp. 1-11, 2018/01/01/ 2018.

[22] A. K. Singh et al., "A Survey and Comparative Study of Hard and Soft Real-Time Dynamic Resource Allocation Strategies for Multi-/Many-Core Systems," ACM Computing Surveys, vol. 50, Jun 2017.

[23] R. Kumar et al., "Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction," in Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture, 2003.

[24] M. H, Hnel et al., "Heterogeneity by the numbers: a study of the ODROID XU+E big. LITTLE platform," presented at the Proceedings of the 6th USENIX conference on Power-Aware Computing and Systems, Broomfield, CO, 2014.

[25] Odroid. (2017, 10th June). SmartPower2. Available: http://odroid.com/dokuwiki/doku.php?id=en:acc:smartpower2

[26] S. Grauer-Gray et al., "Auto-tuning a high-level language targeted to GPU codes," in INPAR 2012, pp. 1-10.

[27] PolyBench/C. (2018, 10th June). PolyBench/C: The Polyhedral Benchmark suite. Available: https://bit.ly/2xofVPd

[28] L.-N. Pouchet and T. Yuki. (2018, 10th June). PolyBench. Available: https://bit.ly/2NLxKSI

[29] S. Dey et al., "EdgeCoolingMode: An Agent Based Thermal Management Mechanism for DVFS Enabled Heterogeneous MPSoCs," in Proceedings of VLSID 2019.