# EdgeCoolingMode: An Agent Based Thermal Management Mechanism for DVFS Enabled Heterogeneous MPSoCs

Somdip Dey[1], Enrique Zaragoza Guajardo[1], Karunakar Reddy Basireddy[2],
Xiaohang Wang[3], Amit Kumar Singh[1], Klaus McDonald-Maier[1]
[1]Embedded and Intelligent Systems Laboratory, *University of Essex*
[2]Cyber Physical Systems, *University of Southampton*; [3]School of Software Engineering, *South China University of Technology*
{ [1] somdip.dey, [1] ez17847, [1] a.k.singh, [1] kdm }@essex.ac.uk; [2] KR.Basireddy@soton.ac.uk; [2] baikeina@163.com

***Abstract*—Thermal cycling as well as temperature gradient in time and space affects the lifetime reliability and performance of heterogeneous multiprocessor systems-on-chips (MPSoCs). Conventional temperature management techniques are not intelligent enough to cater for performance, energy efficiency as well as operating temperature of the system. In this paper we propose a light-weight novel thermal management mechanism in the form of intelligent software agent, which monitors and regulates the operating temperature of the CPU cores to improve reliability of the system. We validated our methodology on the Odroid-XU4 SoC and it has been successful to reduce the operating temperature by 6.32% while improving performance by 7.96% and reducing power consumption by 9.45% than the state-of-the-art.**

***Index Terms*—Lifetime reliability, multiprocessor systems-on- chip (MPSoCs), thermal management, thermal cycling, DVFS**

## I. Introduction and Motivation

Modern embedded systems employ heterogeneous Multi-Processor Systems-on-Chips (MPSoCs), where several types of processing cores are available within a single chip, to deliver power as well as energy efficient computing. Majority of research and development have been focused on developing algorithm in such system-on-chip to provide energy efficiency while catering for performance [1]–[3]. One of the most popular heterogeneous MPSoC is the Samsung Exynos 5422 [4], which employs 4 ARM Cortex A-15 (big) CPUs and 4 ARM Cortex A-7 (LITTLE) CPUs and 6 ARM Mali-T628 GPU cores, implementing ARM's big.LITTLE technology. Comparatively little research has been performed on developing algorithm and mechanisms to provide temperature-aware energy efficient allocation of computing resources while meeting end-user demands for performance on such MPSoCs.

Elevated temperatures have adverse effects on Integrated Circuits (ICs) and reliability of electronic products can be heavily influenced by spatial or temporal gradients, or absolute temperatures [5]. To mitigate such reliability issues most mobile devices often come with thermal capping. Now a days devices often come with hardwired Thermal Management Units (TMUs) as well as software TMUs which regulates the energy consumption as well as the temperature of the associated components. But so far none of these state-of-the-art thermal management units have been proven to be effective enough (see Sec. II) to cater for performance, energy efficiency as well as thermal-regulation.

The Exynos 5422 SoC also supports Dynamic Voltage Frequency Scaling ($DVFS$)[1] capabilities, which could be used to reduce dynamic power consumption ($P \propto V^2 f$) [6]–[8]. In order to cater for performance several resource mapping and partitioning mechanisms using DVFS [1], [2], [6], [9]–[11] have been proposed while keeping energy consumption low. Since applications can be classified into three categories [7]: compute intensive, memory intensive and mixed load (both compute and memory intensive), given most applications in real-world fall under the mixed load category, we had the following observations:

**Observation 1:** ARM Cortex A-15 CPU being an out-of-order sustained triple-issue processor, is capable of providing maximum performance, but when we ran a mixed load program on such CPU, the core temperature rises very fast due to executing workload that are both compute and memory intensive.

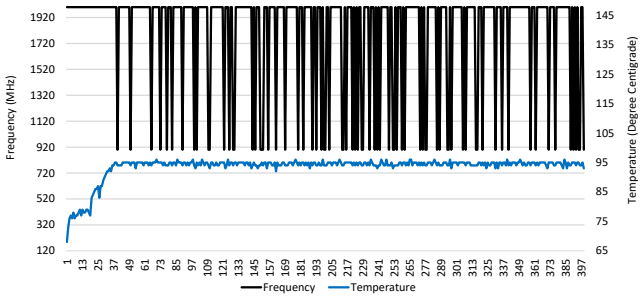


Fig. 1. Frequency and Temperature vs Execution Time Comparison of Streamcluster

Fig. 1 shows the motivation to design a dynamic thermal manager for executing applications on a heterogeneous multi-core architecture (Exynos 5422) containing two types of cores in clusters (big.LITTLE), where 4b and 4L cores are present. The horizontal axis shows the execution time steps whereas the vertical primary axis (left-hand side) reflects the Frequency (in MHz) for the big cores on which the program is executing and the vertical secondary axis reflects the temperature (in °Centigrade) of the big core, which has the worst temperature behavior[2], while executing the workload. We choose Streamcluster[3] from the PARSEC benchmark suit [12] to be our mixed workload on the big cluster. The benchmark completed its execution in 433.12 secs, but Fig. 1 only shows the variance result for first 39.9 secs of the execution because there was repetition in the behavior of the results. From the figure, it could be noticed that the peak temperature[4] on the big core was around 95° centigrades and because of this the thermal management unit (TMU) of the OS starts to regulate the temperature of the cores by CPU throttling[5]. For none of our experiments, we could profile the temperature behavior of LITTLE cores not just because they are less powerful (1/4 size of the big ones and operates at a lower frequency) but there is no

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

[1]DVFS helps to reduce the energy consumption by executing the workload over extra time at a lower voltage and frequency, which could be accounted for reduced power consumption.

[2]Usually the cores exhibit worse temperature behavior which are residing close to the memory.

[3]We choose the *native* option of the Streamcluster to mimic real-world data-mining algorithms, which are both compute intensive and memory intensive.

[4]For our study we have fetched the peak temperature out of the 4 big cores on the SoC. More details are provided in Section IV.

[5]Adjusting the clock speed of the CPU to use less energy consumption and reduce CPU temperature for improved reliability.

2380-6923/19/$31.00 ©2019 IEEE
DOI 10.1109/VLSID.2019.00022



IEEE computer society

temperature sensor for LITTLE cores onboard as well. Therefore all our results only focused on temperature behavior and CPU throttling on the big cores. Although CPU throttling is good in terms of reliable operations but the stock thermal management algorithm of the OS is not intelligent enough to provide performance at the same time. In Fig. 1 we could notice that the clock speed varies from 2000 MHz to 900 MHz at a periodic time step, which causes a drop in performance of the executing application(s). Thus it is necessary to design a thermal manager, which is intelligent enough to cater for performance as well as maintain a desirable temperature of the cores.

**Observation 2:** In a study [7] by Basireddy et al., the researchers have proposed a novel workload management system, which classifies workloads of the executing applications based on Memory Reads Per Instruction (MRPI) metric and manages DVFS levels of cores based on it. This method has resulted to 33% more energy efficiency as compared to modern state-of-the-art workload management approaches and Fig. 2 shows that efficient workload management could also result in energy efficiency as well as performance. For this experimentation, we executed the same Streamcluster benchmark, which completed its execution in 409.596 secs, but Fig. 2 only reflects the variance result for first 39.9 secs of the execution because there was repetition in the behavior of the results. Another interesting thing that could be noticed is that the peak temperature of the big cores also reduced when the workload was mapped appropriately between the CPU cores executed with appropriate DVFS levels for similar mixed-load applications. In some studies [13], [14] it has been found that by reducing the operating temperature by 10-15° centigrades could improve the lifespan of the device by 2x. Since the reliability of the device is highly dependent on the operating temperature of the device, devising temperature-aware mechanisms capable of reducing the operating temperature can help improve the life-span of the SoC device. Therefore, there is a desperate need to implement dynamic thermal manager, which is capable of regulating the operating temperature of the system on top of the state-of-the-art workload and resource management mechanisms so that we could cater not just only energy efficiency and performance but also the overall reliability of the MPSoC.

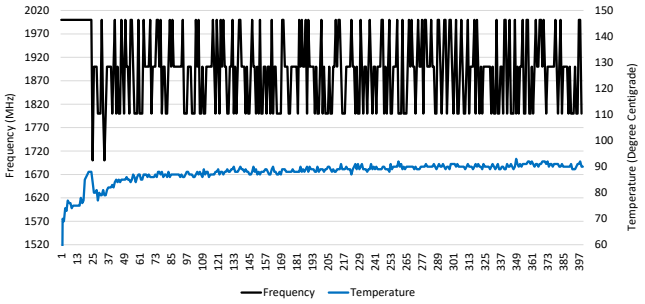


Fig. 2. Frequency and Temperature vs Execution Time Comparison of Streamcluster Using MRPI

In order to overcome the limitations of the existing approaches towards addressing temperature-aware workload management mechanisms, which are capable of catering for both performance and reduced energy consumption, we propose a novel light-weight dynamic thermal management mechanism utilizing the concept of intelligent software agents based on a fast heuristic approach. This thermal management mechanism could be used as module sitting in the application layer to reap its benefits. We call our proposed methodology, "EdgeCoolingMode", since the intelligent software agents monitors and regulates the operating temperature of the system. To this end, this paper makes the following contributions:

1) A light-weight mechanism in the form of intelligent software agent to monitor and regulate temperature of the CPU cores to improve reliability of the system.
2) Validation of the proposed thermal management intelligent agent on a real hardware platform, the Odroid-XU4 [15].

The rest of the paper is organized as follows. Section II presents the state-of-the-art work pursued in the same field. Section III describes our proposed methodology and its implementation. Section IV shows the system model describing the hardware and software infrastructures used for our experiments, the experimental results and validation of our proposed approach. In Section V we explore some related discussion on the proposed methodology and finally, Section VI concludes the paper.

## II. Related Work

In several earlier studies, many researchers have focused on designing methodologies and frameworks to optimize power and operating temperature of MPSoCs. One such noteworthy study is performed by Ghasemazar et al. [16] where the researchers proposed a hierarchical framework leveraging DVFS capabilities of the processing cores to find the optimal voltage-frequency to cater for power consumption and temperature. This methodology was successful in achieving 20% performance boost without impacting the overall operating temperature but their experiments focused mainly on CISC architectures and all results were based on a MATLAB-based Chip Multiprocessor simulator. In another paper [17] Kamal et al. proposed a heuristic based thermal stress-aware mechanism for management of power and temperature in MPSoCs formulated in a convex optimization problem. This approach was implemented on Sniper multicore simulator [18] and was able to reduce spatial and temporal thermal gradients by 7% and 18% respectively when compared to the work in [16]. In another work by Iranfar et al. [19], the researchers proposed a multi-tier hierarchical thermal stress-aware power and temperature management framework for MPSoCs, where the methodology used similar convex optimization solution in multi-layer to improve mean time to failure (MTTF)[6]. The effectiveness of this methodology is again proved based on simulations performed on Sniper multicore simulator. All the aforementioned noteworthy studies were implemented and experimented in simulations[7] instead of experimenting on real devices, which could differ a lot from the practical results achieved from real devices because there are so many factors that account for the operating temperatures such as ambient surrounding temperature, chemical reactions on the devices due to weather change, etc.

In [20] Sigla et al. present a predictor using power sensors to predict the next power consumption based on the following frequency setting is developed. Their technique uses a leakage power model of the ARM's big.LITTLE architecture on the Odroid-XU3 to validate its predictor and Dynamic Power and Frequency Management technique. An Extension of this work has also been published in [21]. Both [20], [21] methods involve predicting the future core temperatures to adjust the workloads or frequencies before exceeding a set threshold, but computation of such predictive temperatures increases the performance overhead of such methods.

In our approach, we utilize the concept of intelligent software agents and linear regression based supervised machine learning to design a light-weight fast heuristic based operating temperature regulator, which could be used on top of any scheduler or governor or

[6]Mean time to failure (MTTF) is the length of time a device is expected to operate/last till failure.

[7]Although modern simulators such as Sniper multicore provides simulation results very close to real devices but there are many unaccountable factors that could happen when executed on a real device instead of simulation, especially in case of temperature variance.

any other types of system resource manager to achieve efficient operating temperature without loosing performance and energy efficiency criteria.

## III. Proposed Methodology: EdgeCoolingMode

### A. Overview of EdgeCoolingMode

There are four schools of thought for artificial intelligence [22]: Thinking Rationally [23], Acting Rationally [24], Thinking Humanly [25] & Acting Humanly [26], where intelligent agents are built or designed to portray each school of thought. In our EdgeCoolingMode we propose an intelligent software agent that would monitor and regulate the thermal behavior of the system by *Thinking and Acting Rationally* based on our heuristic approach.

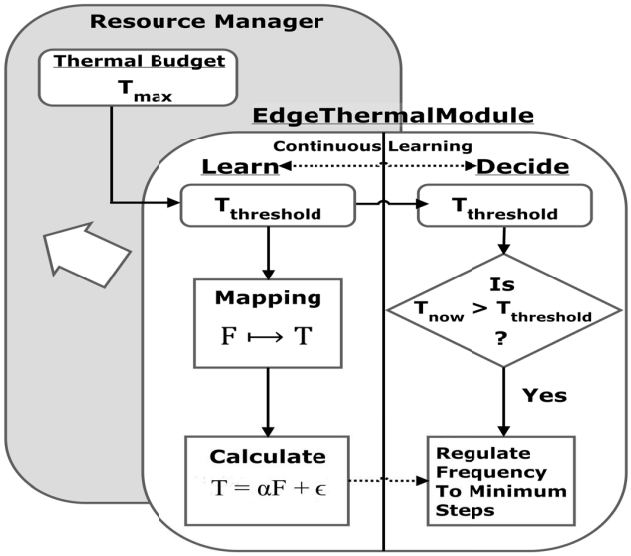


Fig. 3. EdgeCoolingMode Software Agent

Fig. 3 reflects the working of EdgeCoolingMode, which starts by defining a thermal budget by the user. EdgeCoolingMode agent has two distinct modules: Learning module, which monitors the operating frequency & temperature of the cores and creates a relationship variable ($\alpha$) based on linear regression. Although relationship between power and temperature [5], therefore frequency ($P \propto V^2 f$) and temperature does not follow a linear relationship in practice but to make our heuristic approach fast we make the assumption that the relationship is linear and try to reduce the error capacity of our approach by defining an error variable ($\varepsilon$). Based on the user-defined maximum threshold of the operating temperature of the system, the Learning module updates the value of the relationship ($\alpha$) and error ($\varepsilon$) variables. In the Decision module, the agent adjusts the operating temperature of the system by deciding by how much the operating frequency needs to be reduced based on the value of $\alpha$ deduced in the Learning module. In order to make our EdgeCoolingMode agent more intelligent and adaptive, we have also adopted the concept of $ContinuousLearning$ [27]–[29], which is bio-inspired. A human-being continuously learns from its environment and tries to adapt to the surrounding through continuously learning from it. We replicate the same concept in our EdgeCoolingMode agent so that with changing environment and conditions, the agent could adjust the $\alpha$ and $\varepsilon$ variables. Our proposed approach can work with the existing Governor or dynamic thermal and power management (DTPM) of the OS or any other resource manager that is installed on the system or is working in co-habitation.

### B. Learning Module

In the study [5] through experiments on real devices it was found that on heterogeneous MPSoCs such as Odroid platforms, power and temperature follows a relationship of quadratic function or exponential function based on various other dependable factors[8]. We also validate this relationship in Sec. V. However, to make our heuristic approach fast during run time to take necessary decision for thermal regulation we assume that frequency and temperature follows a linear relationship represented by Eq. 1. In Sec. IV also show the effectiveness of using such linear regression based methodology, which has yielded amazing results in terms of performance and reduction in temporal temperature gradients considerably.

$$\begin{aligned} T_i &= \alpha \times F_i + \epsilon \ , \\ where\ F_i &: \text{Operating frequency at time instance i,} \\ T_i &: \text{Operating temperature at time instance i,} \\ \alpha &: \text{Relationship variable,} \\ \epsilon &: \text{Error variable} \end{aligned} \tag{1}$$

In the Learning module the agent tries to monitor the thermal changes using Eq. 1 and evaluate the value of $\alpha$ and $\epsilon$ based on Eq. 2. If we consider that $\alpha$ and $\epsilon$ remain constant between two temperature variance (to find the variance relationship between frequency and temperature) and, $F_2$ and $T_2$ as the operating frequency and operating temperature at the time instance, whereas $F_1$ and $T_1$ as the operating frequency and operating temperature at previous time instance then we could derive:

$$\begin{aligned} T_2 &= \alpha \times F_2 + \epsilon \ \ , \\ T_1 &= \alpha \times F_1 + \epsilon \ \ , \\ \therefore \alpha &= (T_2 - T_1)/(F_2 - F_1) \end{aligned} \tag{2}$$

From Eq. 2 we could solve for $\alpha$. Thus for every instance of operating frequency of the cluster, i.e. $F_i$, the agent maps it to a peak operating temperature of the cluster cores ($T_i$) (see Eq. 3).

$$F_i \longmapsto T_i \tag{3}$$

Now from Eq. 3 and 2 and by reducing the frequency (using DVFS capabilities) by a certain number of frequency scaling levels[9], different $F_i$, $T_i$, $\alpha$, $\epsilon$ & frequency scaling level reduction steps ($l$) are recorded by the agent, which would be used to decide which frequency to drop to in the Decision module when the operating temperature reaches the threshold value[10]. The algorithm for Learning module is provided in Algo. 1.

### C. Decision Module

In the decision module the EdgeCoolingMode agent uses different relationship variable ($\alpha$), error variable ($\epsilon$), frequency scaling level steps $l_i$ computed from Algo. 1 and then calculate the desired frequency using the following equation:

$$\begin{aligned} F_i - F_{desired} &= (T_i - T_{desired})/(\alpha) \ \ , \\ where\ \ T_{desried} &< T_{threshold} \end{aligned} \tag{4}$$

[8]Power/Temperature relationship could vary because of whereabouts of the temperature sensors onboard or distance between temperature sensors and hotspot, etc.

[9]For Exynos 5422 [4] big core cluster has 19 frequency scaling levels with 100MHz each step.

[10]Here, temperature threshold is the thermal cap of the CPU cores that the EdgeCoolingMode agent regulates such that $T_i \leq T_{threshold}$.

**Algorithm 1:** Learning Module Execution

**Input:**
1. $T_{max}$: threshold value of operating temperature
2. $n$: number of different frequency scaling levels

**Output:** $S(\alpha, \epsilon, l)$: set of $\alpha$ values for $s$ frequency scaling levels
**Initialize:** $T_{threshold}$ = $T_{max}$;

**Mapping Frequency with Temperature:**

Write($F_i$, $T_i$, map.csv); ▷ Monitor and track different frequency & temperature readings (Eq. 3)

**Calculate $\alpha$ & $\epsilon$:**

Read(map.csv); ▷ Monitor map.txt file for changes in $T_i$
**if** $T_{i-1} \geq T_{threshold}$ **then**
  **for** *each frequency scaling level* $l_i$ *in* $n$ **do**
    $< \alpha_i, \epsilon_i >$ = CalculateAlphaEpsilon($l$, $F_i$, $T_i$, $F_{i-1}$, $T_{i-1}$); ▷ Compute $\alpha$ & $\epsilon$ using the Eq. 2
    Write($S(\alpha_i, \epsilon_i, l_i)$, alphas.txt);
      ▷ Write our $\alpha$, $\epsilon$ & $l$ values so that it could be later updated through conituous learning return $S(\alpha_i, \epsilon_i, l_i)$;
**else**
  return void();

**Algorithm 2:** Decision Module Execution

**Input:**
1. $T_{max}$: threshold value of operating temperature
2. $n$: number of different frequency scaling level steps

**Output:** $(F_i - F_{desired})_{least}$: least desired operating frequency to drop to
**Initialize:** $T_{threshold}$ = $T_{max}$;

**Decide:**

$S(\alpha_i, \epsilon_i, l_i)$ = Read(alphas.txt);
  ▷ Read the $\alpha_i$ and $\epsilon_i$ for each $l_i$
**if** $T_i \geq T_{threshold}$ **then**
  **for** *each frequency scaling level* $l_i$ *in* $n$ **do**
    $l_{this}$ = CalculateLeastFrequency($\alpha_i$, $F_i$, $T_i$, $F_{i-1}$, $T_{i-1}$);
      ▷ Compute $l_{this}$, which is $(F_i - F_{desired})$
    **if** $l_{this} \leq l_i$ && $l_{this} \geq l_{prev}$ **then**
      SetOperatingFrequency($F_{desired}$);
        ▷ Set the operating frequency to the desired optimal one
      return $(F_i - F_{desired})_{least}$;
    $l_{prev} = l_i$;
**else**
  return void();

The EdgeCoolingMode agent tries to find the least value of $F_i - F_{desired}$ i.e. the least number of frequency scaling level, it should drop to so that the operating temperature could be reduced without affecting the overall performance of the executing application as opposed to what generic TMUs or DPTMs[11] do. Majority of stock TMUs and DPTMs reduce the frequency of the cores drastically to achieve thermal and power budget but that also reduces the performance of the executing application drastically. Although EdgeCoolingMode does not monitor performance directly but since for most computational application operating frequency is directly proportional to performance and hence by reducing the frequency by least scaling level the agent not just reduces the operating temperature but also try to affect performance the least. Therefore, from the Eq. 4 the agent has to find the least value of $F_i - F_{desired}$ by computing the predictive operating temperature using the values of $\alpha_i$ & $\epsilon_i$ for every frequency scaling level steps ($l_i$).

$$\forall\{l_i \in n\} : l_{desried} = (F_i - F_{desired})_{least} \quad , \\ where \;\; l_i : \text{each frequency scaling level} \\ n : \text{number of different frequency scaling level steps} \tag{5}$$

The algorithm for this module is provided in Algo. 2.

### D. Continuous Learning

The Learning module keeps running and keeps updating the value of relationship variable ($\alpha$), error variable ($\epsilon$) for different associated frequency scaling levels for which the operating frequency should be reduced to reduce spatial and temporal thermal gradient.

## IV. Experimental Results

### A. System

*1) Hardware Infrastructure:* Nowadays heterogeneous MPSoCs consist of different types of cores, either having the same or different instruction set architecture (ISA). Moreover, the number of cores of each type of ISA can vary based on MPSoCs and are usually clustered if the types of cores are similar. For this research, we have chosen an Asymmetric Multicore Processors (AMPs) system-on-chip (AMPSoC), which is a special case of heterogeneous MPSoC and has clustered cores on the system. Our study was pursued on the Odroid XU4 board [15], which employs the Samsung Exynos 5422 [4] MPSoC. Exynos 5422 is based on ARM's big.LITTLE technology [30] and contains cluster of 4 ARM Cortex-A15 (big) CPU cores and another of 4 ARM Cortex-A7 (LITTLE) CPU cores, where each core implements the ARM v7A ISA. This MPSoC provides dynamic voltage frequency scaling feature per cluster, where the big core cluster has 19 frequency scaling levels, ranging from 200 MHz to 2000 MHz with each step of 100 MHz and the LITTLE cluster has 13 frequency scaling levels, ranging from 200 MHz to 1400 MHz, with each step of 100 MHz. Additionally, each core on the cluster has a private L1 instruction and data cache, and a L2 cache, which is shared across all the cores within a cluster.

Since Odroid XU4 board does not have an internal power sensor onboard, hence an external power monitor [31] with networking capabilities over WIFI is used to take power consumption readings. Although the ARM Cortex-A7 (LITTLE) CPU cores on Odroid XU4 do not have temperature sensor but our intelligent agent approach is scalable and works for heterogeneous cluster cores.

*2) Software Infrastructure:* For multi-core systems, multi-threaded applications are heavily used in recent times to represent workloads as they could leverage concurrency and parallel processing. Examples of such applications are available in several benchmarks such as PARSEC [12]. For our experiments we have tried several applications from the PARSEC benchmark such as Streamcluster, Facesim, x264, etc. but to validate the effectiveness of our $EdgeCoolingMode$ mechanism we chose Streamclsuter with $native$ option because it closely represented a real-world mixed load application and the execution period was long enough to observe thermal regulation in the system. We also validated our approach for Whetstones benchmark [32]. We have run all our experiments on UbuntuMate version 14.04 (Linux Odroid Kernel: 3.10.105).

### B. Experimental Setup

We implemented our proposed EdgeCoolingMode software agent on top of the MRPI based Mapping and Resource Manager [7] as well as on stock Linux Governors to perform the experiments and validate the effectiveness of our methodology. For all our experiments we did not modify anything else on the system, hardware or software, so that the agent's standalone effectiveness could be determined. We

[11] DPTM is Dynamic Power and Thermal Management techniques that regulates both power and temperature.

choose 7 different operating temperature thresholds: 95°, 94°, 93°, 92°, 91°, 90°and 89°.

### C. Temperature Results

In table I we provide the different $\alpha$ and $\epsilon$ values for different threshold (90-95°)[12] for drop of frequency by two-step levels (reduction of 200 $MHz$). Here each $\alpha$ and $\epsilon$ are the values for the instance when operating temperature reached the threshold value and the agent reduced frequency by two-steps to reduce the operating temperature.

TABLE I
RESULTS: VALUES OF $\alpha$ & $\epsilon$ FOR DIFFERENT TEMPERATURE THRESHOLDS

| **95°** | **94°** | **93°** | **92°** | **91°** | **90°** |
|---|---|---|---|---|---|
| $\alpha = 0.02, \epsilon = 55$ | $\alpha = 0.01, \epsilon = 74$ | $\alpha = 0.01, \epsilon = 73$ | $\alpha = 0.01, \epsilon = 73$ | $\alpha = 0.02, \epsilon = 55$ | $\alpha = 0.02, \epsilon = 55$ |
| $\alpha = 0.02, \epsilon = 55$ | $\alpha = 0.01, \epsilon = 74$ | $\alpha = 0.01, \epsilon = 73$ | $\alpha = 0.01, \epsilon = 73$ | $\alpha = 0.02, \epsilon = 55$ | $\alpha = 0.02, \epsilon = 52$ |
| $\alpha = 0.005, \epsilon = 85$ | $\alpha = 0.01, \epsilon = 74$ | $\alpha = 0.01, \epsilon = 73$ | $\alpha = 0.01, \epsilon = 73$ | $\alpha = 0.01, \epsilon = 73$ | $\alpha = 0.02, \epsilon = 55$ |
| $\alpha = 0.01, \epsilon = 75$ | $\alpha = 0.02, \epsilon = 56$ | $\alpha = 0.02, \epsilon = 55$ | $\alpha = 0.02, \epsilon = 55$ | $\alpha = 0.02, \epsilon = 55$ | $\alpha = 0.01, \epsilon = 71$ |

Fig. 4 and 5 show the operating temperature & frequency respectively of the big core, which was monitored during the execution of benchmarks with our proposed EdgeCoolingMode agent in effect as well without utilizing the agent. We choose to show only values for 5 temperature threshold: 89°, 91°, 93°& 95°, so that it could be easily understood from the graph.

We ran Streamcluster from PARSEC five times each for different temperature threshold values (89°, 90°, 91°, 92°, 93°, 94°& 95°) and for MRPI, Linux governor in performance mode & Linux governor in ondemand mode. Fig. 6 summarizes the average execution time (in secs), average peak temperature (in ° centigrade), average power consumption (in Watt) and the average number of times CPU throttling took place. The average is computed by taking all 5 execution for each temperature budget into account. From the table it could be noticed that using our EdgeCoolingMode we have achieved 6.32% reduction in peak temperature[13] while improving performance by 7.96% and reducing power consumption by 9.45%.

We executed Whetstones benchmark for five times and here, we present the average result of all five executions. For Whetstones the total execution time, when run with the Linux's stock governor ondemand, is 121.56 secs with 206.8 times CPU throttling and an average temperature of 93.95° centigrade. With our approach, we achieved 4.11% reduction in temperature with a performance boost of 1.65% and no CPU throttling. We have also noticed that while using linear regression of temperature vs frequency instead of quadratic equation, on an average the computation takes 193 ms whereas the quadratic one takes 237 ms and hence has a speedup of 1.295x with no loss in accuracy over time.

## V. DISCUSSION

During our experiments, we noticed that power/temperature relationship was following a quadratic equation instead of exponential as is the well-known case. There could be several reasons for such behavior. One possibility is that we have run our experiments with the active cooling mechanism (cooling fan with heatsink) on board of the

[12]We only showed from 90-95° centigrades for this experiment to show the computations of $\alpha$ and $\epsilon$ for space constraint in this paper.

[13]Considering that Linux's thermal cap is at 95°centigrades.

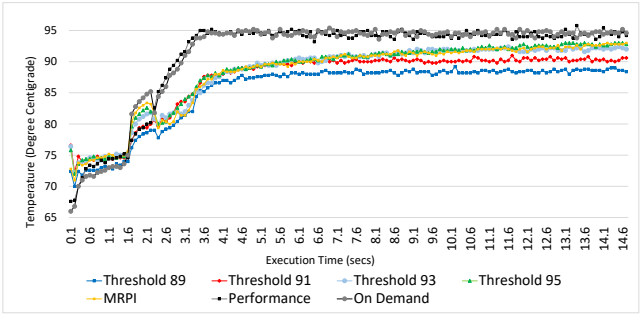


Fig. 4. Execution Time/Temperature Relationship For Streamcluster

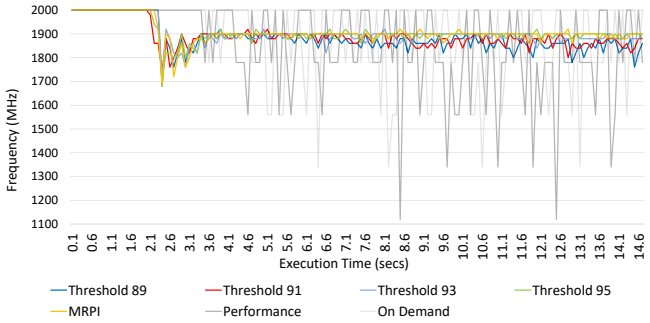


Fig. 5. Execution Time/Frequency Relationship For Streamcluster

Odroid XU4, which tried to physically regulate the temperature thus regulating the spatial thermal gradient as much as possible through active heat dissipation. Another point we have to keep in mind is that since our methodology acts as an intelligent agent to enhance the capacities of thermal regulation of the system instead of acting as a replacement for the already existing thermal management unit on the system, we did not turn off the TMUs for our experiments. We mapped the relationship between power and temperature after executing[14] our EdgeCoolingMode mechanism with a thermal threshold of 90° centigrades and the relationship follows a quadratic function (see Fig. 7). Since our proposed EdgeCoolingMode was acting as a thermal regulator, therefore, regulating the temporal thermal gradient

[14]We ran Stremcluster benchmark from PARSEC and mapped the power consumption with the peak temperature of the big cluster.

| Model | Exec Time (sec) | Avg. Temp. (°C) | Avg. Power (W) | Times Throttling |
|---|---|---|---|---|
| Th. 89 | 398.98 | 88.5 | 9.97 | 0 |
| Th. 90 | 395.56 | 89.26 | 10.07 | 0.2 |
| Th. 91 | 399.57 | 90.37 | 10.23 | 0.6 |
| Th. 92 | 398.6 | 91.25 | 10.39 | 1 |
| Th. 93 | 395.88 | 92.19 | 10.5 | 3.5 |
| Th. 94 | 394.42 | 92.91 | 10.64 | 8.2 |
| Th. 95 | 400.93 | 93.51 | 10.76 | 55.8 |
| MRPI | 406.63 | 93.7 | 10.79 | 119.2 |
| Performance | 433.01 | 94.08 | 10.89 | 649 |
| On-Demand | 429.78 | 94.24 | 11.01 | 630.2 |

Fig. 6. Result for different temperature threshold: Execution time, Avg. temperature, Avg. power, Avg. times throttling for different temperature threshold

and hence from the aforementioned figure (Fig. 7) we could see that the temperature is regulated (temperature variance stabilizes) and maintained below the thermal cap set by the OS.

We have also noticed that we achieve different results based on the time period of the day due to the difference in ambient temperature between night and day[15]. Fig. 6 shows the results of experiments performed during daytime when the ambient temperature was hottest. Thus if the same set of experiments are performed now the results might vary by a little percentage.

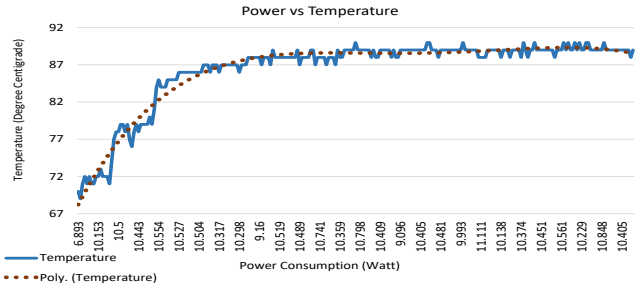


Fig. 7. Power/Temperature Relationship Using EdgeCoolingMode

## VI. Conclusion

In this paper, we have proposed a light-weight thermal management mechanism in the form of an intelligent agent, which is capable of monitoring and regulating the operating temperature of the CPU cores on MPSoCs. The efficacy of the methodology was evaluated by implementing and validating the mechanism on the Odroid-XU4 MPSoC, employing ARM's big.LITTLE architecture. The results of applying the agent-based thermal manager showed that, compared with the state-of-the-art power and temperature management approach, the proposed approach was not only capable of reducing operating temperature of the CPU cores but at the same time improved performance of the executing application as well as reduction in power consumption.

## References


[1] A. K. Singh, P. Dziurzanski, H. R. Mendis, and L. S. Indrusiak, "A survey and comparative study of hard and soft real-time dynamic resource allocation strategies for multi-/many-core systems," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, p. 24, 2017.

[2] A. K. Singh, C. Leech, B. K. Reddy, B. M. Al-Hashimi, and G. V. Merrett, "Learning-based run-time power and energy management of multi/many-core systems: current and future trends," *Journal of Low Power Electronics*, vol. 13, no. 3, pp. 310–325, 2017.

[3] B. K. Reddy, G. V. Merrett, B. M. Al-Hashimi, and A. K. Singh, "Online concurrent workload classification for multi-core energy management," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2018, pp. 621–624.

[4] "Exynos 5 octa (5422)," https://www.samsung.com/exynos, accessed: 2018-07-23.

[5] K. DeVogeleer, G. Memmi, P. Jouvelot, and F. Coelho, "Modeling the temperature bias of power consumption for nanometer-scale cpus in application processors," in *Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV), 2014 International Conference on*. IEEE, 2014, pp. 172–180.

[6] A. Aalsaud, R. Shafik, A. Rafiev, F. Xia, S. Yang, and A. Yakovlev, "Power–aware performance adaptation of concurrent applications in heterogeneous many-core systems," in *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*. ACM, 2016, pp. 368–373.

[7] B. K. Reddy, A. Singh, D. Biswas, G. Merrett, and B. Al-Hashimi, "Inter-cluster thread-to-core mapping and dvfs on heterogeneous multi-cores," *IEEE Transactions on Multiscale Computing Systems*, pp. 1–14, 2017.


[15]Ambient temperature between night and day varied by a couple of degrees, hence, affecting the results.


[8] S. Dey, A. K. Singh, and K. McDonald-Maier, "Energy efficiency and reliability of computer vision applications on heterogeneous multi-processor systems-on-chips (mpsocs)."

[9] K. Chandramohan and M. F. O'Boyle, "Partitioning data-parallel programs for heterogeneous mpsocs: time and energy design space exploration," in *ACM SIGPLAN Notices*, vol. 49, no. 5. ACM, 2014, pp. 73–82.

[10] R. Barik, N. Farooqui, B. T. Lewis, C. Hu, and T. Shpeisman, "A black-box approach to energy-aware scheduling on integrated cpu-gpu systems," in *Proceedings of the 2016 International Symposium on Code Generation and Optimization*. ACM, 2016, pp. 70–81.

[11] A. K. Singh, A. Prakash, K. R. Basireddy, G. V. Merrett, and B. M. Al-Hashimi, "Energy-efficient run-time mapping and thread partitioning of concurrent opencl applications on cpu-gpu mpsocs," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 5s, p. 147, 2017.

[12] C. Bienia, "Benchmarking modern multiprocessors," Ph.D. dissertation, Princeton University, January 2011.

[13] T. Chantem, R. P. Dick, and X. S. Hu, "Temperature-aware scheduling and assignment for hard real-time applications on mpsocs," in *Proceedings of the conference on Design, automation and test in Europe*. ACM, 2008, pp. 288–293.

[14] A. K. Coskun, T. S. Rosing, and K. Whisnant, "Temperature aware task scheduling in mpsocs," in *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE'07*. IEEE, 2007, pp. 1–6.

[15] "Odroid-xu4," https://goo.gl/KmHZRG, accessed: 2018-07-23.

[16] M. Ghasemazar, H. Goudarzi, and M. Pedram, "Robust optimization of a chip multiprocessor's performance under power and thermal constraints," in *Computer Design (ICCD), 2012 IEEE 30th International Conference on*. IEEE, 2012, pp. 108–114.

[17] M. Kamal, A. Iranfar, A. Afzali-Kusha, and M. Pedram, "A thermal stress-aware algorithm for power and temperature management of mpsocs," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*. EDA Consortium, 2015, pp. 954–959.

[18] T. E. Carlson, W. Heirman, and L. Eeckhout, "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulations," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, Nov. 2011, pp. 52:1–52:12.

[19] A. Iranfar, M. Kamal, A. Afzali-Kusha, M. Pedram, and D. Atienza, "Thespot: Thermal stress-aware power and temperature management for multiprocessor systems-on-chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 8, 2018.

[20] G. Singla, G. Kaur, A. K. Unver, and U. Y. Ogras, "Predictive dynamic thermal and power management for heterogeneous mobile platforms," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*. EDA Consortium, 2015, pp. 960–965.

[21] G. Bhat, G. Singla, A. K. Unver, and U. Y. Ogras, "Algorithmic optimization of thermal and power management for heterogeneous mobile platforms," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 3, pp. 544–557, 2018.

[22] M. Negnevitsky, *Artificial intelligence: a guide to intelligent systems*. Pearson Education, 2005.

[23] D. McDermott and E. Charniak, "Introduction to artificial intelligence," *Reading: Addison-Wesley*, 1985.

[24] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[25] J. Haugeland, "Artificial intelligence: The very idea. 1985," *Cited on*, p. 26, 1985.

[26] E. Rich and K. Knight, "Artificial intelligence," *McGraw-Hill, New*, 1991.

[27] S. Dey, G. Kalliatakis, S. Saha, A. K. Singh, S. Ehsan, and K. McDonald-Maier, "Mat-cnn-sopc: Motionless analysis of traffic using convolutional neural networks on system-on-a-programmable-chip," *2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2018)*, 2018.

[28] S. Thrun and T. M. Mitchell, "Lifelong robot learning," in *The biology and technology of intelligent autonomous agents*. Springer, 1995, pp. 165–196.

[29] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, and T. M. Mitchell, "Toward an architecture for never-ending language learning." in *AAAI*, vol. 5. Atlanta, 2010, p. 3.

[30] "Arm big.little technology," http://www.arm.com/, accessed: 2018-07-23.

[31] "Odroid smartpower2," https://www.hardkernel.com/main/products/prdt_info.php\?g_code=G148048570542, accessed: 2018-07-23.

[32] H. J. Curnow and B. A. Wichmann, "A synthetic benchmark," *The Computer Journal*, vol. 19, no. 1, pp. 43–49, 1976.