Efficient Resource Management of Clustered Multi-Processor Systems Through Formal Property Exploration

Ourania Spantidi*, Iraklis Anagnostopoulos*, and Georgios Fainekos†
*Department of Electrical, Computer and Biomedical Engineering, Southern Illinois University, Carbondale, U.S.A.
†School of Computing, Informatics and Decision Systems Engineering, Arizona State University, Tempe, U.S.A.
*{ourania.spantidi,iraklis.anagno}@siu.edu, ‡fainekos@asu.edu

***Abstract*—Modern embedded systems have adopted the clustered Chip Multi-Processor (CMP) paradigm in conjunction with dynamic frequency scaling techniques to improve application performance and power consumption. Nonetheless, modern applications are becoming more aggressive in terms of computational power. At the same time, the integration of multiple cores in the same cluster has resulted in significant increase of power consumption creating thermal hotspots. Conventional design approaches consider fixed power and temperature constraints, which are mostly extracted experimentally leading many times to pessimistic run-time decisions and performance losses. In this paper, we present a unified framework for efficient resource management of clustered CMPs by enabling formal property exploration and integrating robustness analysis. Specifically, we bridge the gap between run-time decisions and design-time exploration by using Parametric Signal Temporal Logic (PSTL) for mining the values of system constraints. Then, we utilize the extracted values to enhance the decisions of the run-time resource manager. Results on the Odroid-XU3 show that the proposed methodology offers more coarse- and fine-grain optimizations.**

***Index Terms*—chip multiprocessors, resource management, DVFS, parametric specification mining**

I. Introduction

Modern embedded systems have adopted the clustered Chip MultiProcessor (CMP) paradigm clustering together cores of the same type and sharing a last level cache and memory controller. Additionally, Dynamic Voltage and Frequency Scaling (DVFS) mechanisms are used to control and adjust power consumption. However, while this architecture allows for considerable performance gains for modern applications, it presents major challenges in balancing power-performance constraints [1].

Previous approaches try to satisfy power/temperature thresholds, without providing any insights on how they determined these values [2], [3]. However, these values may be not be optimal as different applications have different effect on the system. Particularly, Thermal Design Power (TDP) is used as a power threshold during application execution. When applications exceed TDP, the resource manager limits the CPU speed and it can even deactivate specific cores, thus significantly impacting system throughput. However, TDP is not the maximum achievable power that ensures thermal safety. Hence, considering a single and constant power budget for clustered CMPs often leads to pessimistic decisions and performance losses [4]. Similarly, temperature guardbands are used for frequency control when the temperature reaches a critical value. While, this mechanism provides a safety net, it greatly affects performance as the operating frequency will be conservatively bounded to the worst-case temperature. Furthermore, for embedded devices (e.g. mobile phones), due to lack of active cooling, the worst-case temperature is pessimistically estimated without considering the actual properties of the architecture and the workload [5].

As a motivation, Figure 1 presents the power and temperature for two application mixes running on the A15 cluster of the Odroid-XU3 platform. Each mix contains four concurrent executing applications that exert pressure on the memory subsystem in different ways and utilize all cores in the cluster. Additionally, for each mix, we considered two cases: (1) executing under TDP constraint keeping power under $4W$ following [2], and (2) executing under a CPU temperature threshold of $60°C$, which is used in modern mobile devices. While under TDP execution, both mixes satisfy the $4W$ constraint. However Mix 2 had almost $10°C$ higher temperature than Mix 1. Thus, even though both mixes consumed the same power, the fact that Mix 2 had a greater impact on temperature, requires the TDP to be very pessimistic in order to guarantee thermal safety. In the case of temperature threshold ($60°C$), both mixes satisfy it. However, Mix 2 managed to execute on a higher frequency consuming also more power but without any significant impact on temperature. Additionally, it had 8% increase in performance comparing to the TDP scenario. Concluding, a single and constant power budget for clustered CMPs can lead to pessimistic decisions and performance losses, making necessity an investigation and in depth analysis of more complex properties. The difficulty of choosing the threshold values for the constraints lies in the facts that (i) there is no *systematic approach* to investigate and accelerate an in-depth analysis of more complex properties, and (ii) *automatically* infer the ranges of the constraint parameters.

A systematic method to mine the values for the parameters of a system is via the utilization of syntax and semantics of Parametric Signal Temporal Logic (PSTL) formulas. PSTL extends Signal Temporal Logic (STL) [6], a specification formalism to express temporal properties. Once an STL formula is expressed, a robustness analysis [7] is used in order to evaluate in which cases the properties do not hold true. Building multiple STL expressions and checking system robustness makes the

978-3-9819263-5-4/DATE21/©2021 EDAA

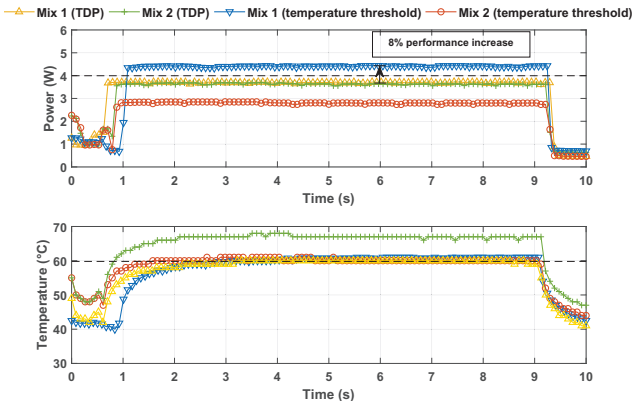

Fig. 1: Motivation example

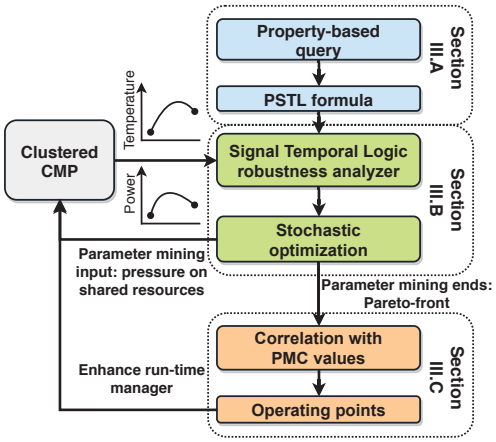

Fig. 2: Overall methodology

exploration more systematic; however, it is not scalable as each formula must be very specific and covers a small part of the design space. PSTL solves this problem by replacing threshold constants with parameters that must be inferred.

In this paper, instead of designing resource managers under fixed and unexplored constraints, we present an approach to infer the ranges of parameters for specifications under different types of workloads. Particularly, we present a unified framework that utilizes PSTL to express run-time constraints and mine multiple parameters in order to create optimized system- and application-aware operating points. These operating points are used to enhance the run-time resource manager in order to trigger DVFS efficiently and satisfy system, user, and throughput requirements at the same time. To the best of our knowledge, this is the first work that couples the concept of PSTL and run-time resource management on CMPs. Overall, the contributions of the proposed framework are:

(1) We systematically infer the constraint parameters of clustered CMPs in order to increase system power efficiency.

(2) We go beyond traditional design space exploration by utilizing robustness metrics to determine system properties.

(3) We demonstrate how the inferred parameters can be used in order to enhance run-time resource allocation decisions.

(4) We evaluate the whole framework on Odroid-XU3 board over varying workload conditions.

II. Related work

Resource management of modern CMPs is a well studied topic due to its importance and to the increased number of actuation knobs that modern platforms offer. Previous research works can be split into two categories. In the first one, single objective optimization techniques have been developed. Particularly, such approaches developed run-time resource managers that focus only on one metric such as application performance/system throughput [8], [9], energy consumption minimization [10], [11], or resource utilization [12]. In order to satisfy the objective, the aforementioned approaches search the configuration space and create policies for the run-time managers. To overcome the exploration time bottleneck, there have been efforts to accelerate the process by developing heuristics [13], [14]. Alternatively, there have also been efforts to develop multi-objective approaches that try to optimize both execution time and energy/power consumption [15], [16], [17]. Since these objectives are often conflicting, a variation of multi-objective optimization is to rank the objectives and assign priorities and goals to specific metrics [18]. In that way, the run-time manager monitors the different goals and utilizes the available actuators in order to enforce resource allocation policy to satisfy the hierarchy [2]. However, all the aforementioned resource allocators and controllers work considering fixed power, temperature, and performance constraints which are mostly extracted experimentally. Specifically, there is no justification why meeting a power threshold will always keep temperature within safe values since applications utilize on-chip resources in a different way. Furthermore, keeping a low power threshold will always satisfy temperature constraints; however, it may not allow the resource allocators to fully exploit the computing resources of the platform.

III. Methodology

Figure 2 presents an overview of the proposed framework, which consists of three steps. **Step 1:** Express a property-based query using Parametric Signal Temporal Logic (PSTL) and define the output trajectories (e.g., power, temperature, throughput) to monitor and analyze. Example of such queries are "What is the maximum power threshold and overall system throughput we can achieve while maintaining thermal safety", or "What is the maximum power threshold and overall system throughput so that the application performance is satisfied, while maintaining thermal safety". **Step 2**: Once the query has been expressed in PSTL, we trigger the parameter mining phase. Initially, a random workload is assigned to each core, utilizing a developed library of micro-benchmarks, while the output trajectories are monitored by an analyzer. Then, based on the robustness of the trajectories, a stochastic sampler selects a new workload for each core, trying to find operating conditions that falsify the query. At the end, we get the Pareto-front in the parameter space for which the requirement is guaranteed to not be satisfied. Note that, frequency is not a part of the search space. Step 2 is repeated for all the frequencies in the range 1.4 - $2.0GHz$. **Step 3**: The mined values of the Pareto-front are used to build run-time operating points and replace static power thresholds. In that way, the manager decides at run-time the

power budget and based on the current status of the platform adjusts the operating frequency accordingly.

A. Preliminaries of PSTL

STL is used to express, in a compact and formal way, temporal properties of a cyber-physical system. Let x be a vector variable, i.e., $x = [x_1, \ldots, x_n]^T$, $p(x)$ be a predicate over the reals, and $\mathcal{I}$ be any non-empty interval of $\mathbb{R}_{\geq 0}$. The syntax for STL formulas is provided by the grammar:

$$\phi ::= \mathbf{T} \mid p(x) \geq 0 \mid \neg\phi \mid \phi \vee \phi \mid \bigcirc_{\mathcal{I}} \phi \mid \phi \mathcal{U}_{\mathcal{I}} \phi$$

where $\mathbf{T}$ is *true*, $\neg$ and $\vee$ are the Boolean operators for negation and conjunction respectively, $\bigcirc_{\mathcal{I}}$ is the next sample operator, $\mathcal{U}_{\mathcal{I}}$ is the until operator, and $\mathcal{I}$ represents a physical time interval. Other common Boolean and temporal operators can be defined as syntactic abbreviations [19]:

- $\Diamond_{\mathcal{I}}\phi \equiv \mathbf{T}\mathcal{U}_{\mathcal{I}}\phi$ stands for eventually at some time in the time interval $\mathcal{I}$, ϕ should be true, and
- $\Box_{\mathcal{I}}\phi \equiv \neg\Diamond_{\mathcal{I}}\neg\phi$ stands for always during the interval $\mathcal{I}$, ϕ should be true.

When $\mathcal{I} = [0, \infty)$, we will be dropping $\mathcal{I}$ from the notation, e.g., $\Box_{[0,\infty)}\phi \equiv \Box\phi$.

Following the STL semantics, we map a formula φ and a trace σ to a value drawn from the set $\mathbb{R}_{\geq 0}$. Robustness of the formula φ, over the trace σ at sample i, is a quantitative measure that shows how far the trace σ at time i is from satisfaction of the STL formula [20].

In many applications, it is important to explore the properties that a signal satisfies as opposed to issuing a verdict that a signal satisfies or not a given specification φ. For example, instead of asking the question, does σ satisfy the specification that the power is always less than $6W$ in the first 5sec, i.e., $\varphi = \Box_{[0,5]}(Power \leq 6)$, we may leave the time constraint as parameter θ to be mined, e.g., $\varphi[\theta] = \Box_{[0,\theta]}(Power \leq 6)$. In other words, we want to learn what is the longest time θ for which the requirement holds. Parametric Signal Temporal Logic (PSTL) was introduced in [21] as a formal language to capture such questions. The parameter mining problem is formally defined as:

Problem III.1 (STL Parameter Mining)**.** *Given a PSTL formula $\phi[\theta]$ with a vector of m unknown parameters $\theta \in \Theta = [\underline{\theta}, \overline{\theta}]$ and a system Σ, find the set $\Psi = \{\theta^* \in \Theta \mid \Sigma \not\models \phi[\theta^*]\}$.*

The notation $\Sigma \not\models \phi$ means that there exists a trajectory σ of the system Σ such that σ does not satisfy φ. Hence, the goal of parameter mining is to identify the boundary of Ψ as close as possible and, then use the notion of temporal logic robustness to provide a measure of robust system performance.

B. Parameter mining on clustered CMPs

In this work, we utilize PSTL to systematically infer the constraint parameters of clustered CMPs in order to increase system power efficiency (coarse-grain optimization) and user-defined throughput objectives (fine-grain optimization). To that end, we consider three levels of objectives such as *system*, *application*, and *throughput* and two queries to mine parameters for. *System* is the primary objective which ensures that the power consumption will keep the temperature under a safe value $Temp_{safe}$. This objective is of high priority in order to preserve thermal safety. The *application* is the secondary objective which targets the user experience. In modern embedded systems, users require specific applications to achieve a minimum of performance. Finally, *throughput* is a tertiary objective and targets a minimum throughput for the overall platform in order to avoid starvation for background processes and services running on an embedded device. The first query (Q1) targets to optimize the power efficiency of the system, while avoiding system under-utilization. The second query (Q2) takes into consideration specific application requirements imposed by the user:

Q1: *For a thermal safety temperature value $Temp_{safe}$, what is the maximum achievable power P_{thr} and maximum overall system throughput Th_{thr} before facing temperature issues?*

$$\varphi^{Q1}[\theta] = \varphi_1^{Q1}[\theta_1] \wedge \varphi_3^{Q1}[\theta_3] \tag{1}$$

where

$$\begin{aligned}\varphi_1^{Q1}[\theta_1] =& \Box(Power \leq \theta_1)\\ \varphi_3^{Q1}[\theta_3] =& \Box(\text{Throughput} \leq \theta_3) \implies \\ & \Box(\text{Temp} \leq Temp_{\text{safe}})\end{aligned}$$

Q2: *For a given user application and a thermal safety temperature value $Temp_{safe}$, what is the maximum achievable power P_{thr}, maximum user application performance $Perf_{thr}$, and maximum overall system throughput Th_{thr} before facing temperature issues?*

$$\varphi^{Q1}[\theta] = \varphi_1^{Q1}[\theta_1] \wedge \varphi_2^{Q1}[\theta_2] \wedge \varphi_3^{Q1}[\theta_3] \tag{2}$$

where

$$\begin{aligned}\varphi_1^{Q1}[\theta_1] =& \Box(Power \leq \theta_1)\\ \varphi_2^{Q1}[\theta_2] =& \Box(\text{Perf}_{app} \leq \theta_2)\\ \varphi_3^{Q1}[\theta_3] =& \Box(\text{Throughput} \leq \theta_3) \implies \\ & \Box(\text{Temp} \leq Temp_{\text{safe}})\end{aligned}$$

In both queries, θ is the vector of parameters to be inferred, while temperature ($^\circ C$), power (W), application performance (Instructions per Second (IPS)) in Q2, and system throughput (total IPS) are the output trajectories. As shown in [20], each of the sub-queries $\varphi_i^{Qi}[\theta]$ could be explored independently by formulating an optimization problem over the corresponding single parameter θ_i. In this work, we utilized the software toolbox S-TALIRO [22], which automatically formulates and solves the optimization problem for a given parametric specification $\varphi[\theta]$.

In clustered CMPs, cores are not fully independent processors. Simultaneous executing applications fight for shared resources such as memory controller and last level cache, significantly affecting temperature, power, application performance, and overall throughput. Consequently, the return values of the parameter mining phase depend on the actual workload. For that reason, we created a library of micro-benchmarks which consists of memory-, cache-, and compute-intensive micro-benchmarks that put tunable pressure on the shared resources. Specifically, the memory- and cache-intensive benchmarks have constant memory bandwidth with configurable size, while the compute-intensive ones pressure the integer and floating point units of the cores. Additionally, each micro-benchmark takes as input a value that dictates its working dataset. If the dataset value is less than the L1 cache, then a compute-intensive benchmark is

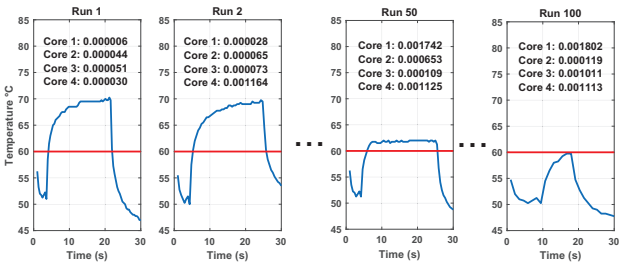

Fig. 3: Example regarding the PSTL parameter mining and the functionality of the stochastic optimizer

triggered. If the dataset value is less than the double size of L2 cache, then a cache-intensive is used that heavily utilizes the data in L2 cache. In all other cases a memory-intensive micro-benchmark is triggered putting pressure on memory controller and thrashing the L2 cache. This value of the dataset is the control knob of the stochastic optimizer.

Initially, a random micro-benchmark is loaded on each core (random dataset value). In the case of query Q2, one core always executes the user-defined application and the rest random micro-benchmarks. After the end of the execution, we retrieve the trajectories of temperature, power consumption, application performance, and system throughput. In order to correlate the output trajectories with the activity of the cluster and the behavior of the concurrent executing applications, we use the per core and total Memory Reads Per Instruction (MRPI) as metrics, which can be monitored at run-time (MRPI = last level cache misses/instructions retired) [11]. The MRPI is a frequency-less metric that has been proved to provide better performance insights for concurrent executing applications [11]. According to the PSTL expression used, the output trajectories are analyzed and the robustness value of the system is calculated. The stochastic optimizer uses this robustness value and correlates it with the per core dataset values that produced that output. Then, it decides the next workload by modifying the dataset value per core, thus modifying the pressure on the shared resources. By tuning the dataset value or micro-benchmarks per core, the stochastic optimizer aims to push system behavior as close as possible to the specified constraint boundaries. The parameter exploration terminates once a maximum number of tests is completed and it is performed for all the available frequencies. At the end, the framework returns the mined values of θ along with the maximum MRPI value under which the values were mined. This value shows how compute- or memory-intensive the workloads can be, in order to satisfy the initially given constraints. Low values of MRPI indicate a more compute-intensive behavior, while larger values indicate a more memory-intensive one.

Figure 3 shows an example of the mining process for query Q1 considering $Temp_{safe} = 60°C$. For this example, we set the maximum number of tests for the stochastic optimizer to 100. For each run, Figure 3 shows the output temperature signal, as well as the MRPI value of each core (4 cores total). Initially (Run 1), random micro-benchmarks are assigned on each core. At this point the PSTL formula is not satisfied. The robustness value of the system is negative (since we have exceeded $Temp_{safe}$). Thus, the stochastic optimizer selects to modify the dataset value for the cores. In the second run, the temperature drops slightly, since the sampler picked a more memory-intensive workload for Core 4. As the procedure continues and the stochastic optimizer keeps modifying the dataset, at run 50 the temperature has dropped $15°C$, and the MRPI values of Core 1 and Core 4 correspond to more memory-intensive workloads. At this point, the robustness value of the system is still negative, however it is approaching zero as the distance of the trajectory from the threshold becomes smaller. Finally, at run 100 which is the last one, the stochastic optimizer has selected such values, that the output temperature of the system does not exceed $60°C$. The robustness value of the system is now positive with an absolute value close to zero. The resulting mined parameter values in this example were θ_1, θ_2, that correspond to the Pareto-front of maximum power and system throughput that satisfied the temperature constraint.

C. Enhancing run-time manager decisions

After the exploration phase is done for each frequency value, we create a look-up table which contains tuples of the frequency of the mining phase, and the respective workloads associated with their MRPI values. We built our run-time manager on top of the goal manager presented in [2] as this method follows the same principle with our design, having three hierarchical goals (1) thermal safety, (2) user experience, (3) and performance. The thermal safety goal has the highest priority and it should always be satisfied in terms of a fixed power threshold. An urgency metric is introduced in order to capture the extent of the violation of a constraint [2]:

$$U_{Pow} = \frac{P_{curr}}{P_{ref}}, U_{Perf} = \frac{perf_{max} - perf_{curr}}{perf_{max} - perf_{ref}} \quad (3)$$

where P_{curr} is the current power consumption and $perf_{curr}$ is the current application performance, while values of urgency equal or greater than one indicate a constraint violation. However, for all the constraints the reference values are fixed and the same for all applications (e.g., $P_{ref} = 4W$ (TDP)). As a way to adjust frequency at run-time taking into consideration the effect of each application and the properties of the platform, the run-time manager monitors periodically the per core and total MRPI values, finds in the look-up table the corresponding mined power threshold (P_{ref} in Equation 3), and adjusts the operating frequency accordingly so as not to exceed it. Thus, the proposed method allows the run-time manager to associate the power threshold with the current workload, ensuring better thermal safety and increasing power efficiency.

IV. Evaluation

The developed automation framework was evaluated on top of the A15-cluster of Odroid-XU3 board utilizing the S-TaLiRo [22], [23] tool for PSTL mining. All workloads run on the Odroid platform, and the output observations (e.g., temperature, power) are sent to a host computer which executes S-TaLiRo and performs the parameter mining (i.e., robustness analyzer and stochastic optimization). Finally, we utilized the underlying Performance Monitoring Counters (PMCs) to monitor and record the retired instructions and L2 cache misses (MRPI), while the in-built Power Monitoring Unit (PMU) was used to record power consumption and temperature.

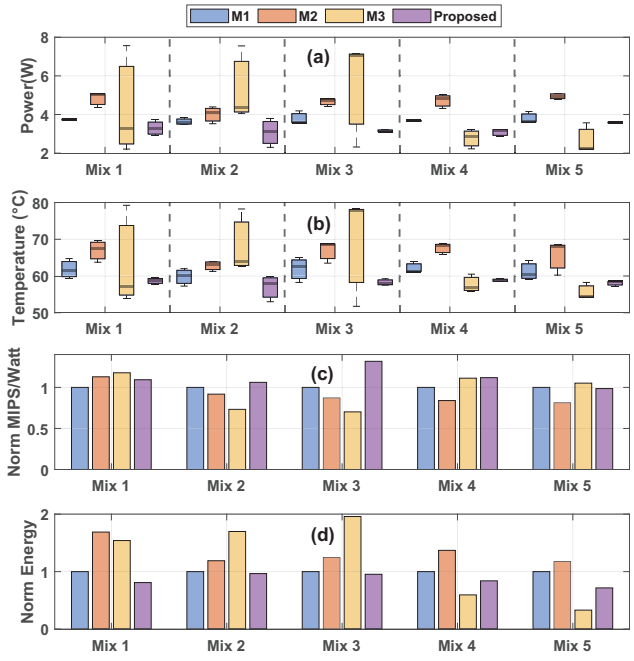

Fig. 4: Comparison of the proposed approach in terms of (a) power consumption; (b) temperature; (c) power efficiency (normalised MIPS/Watt); and (d) normalised energy consumption

A. Results on coarse-grain optimization for power efficiency

Regarding query Q1 (Equation 1), we performed the parameter mining considering $Temp_{safe} = 60°C$. We selected to run the parameter mining process by setting 100 iterations for the stochastic optimizer. This process requires approximately one hour on the Odroid-XU3. In order to extract a more optimized Pareto-front of solutions, we repeated this process 10 times. At this point, it is important to mention this step is required only once and it is performed offline. Once the power threshold values have been extracted, we enhance the run-time allocator. We have experimentally observed that we are able to serve any incoming workload, regardless of whether it has been previously encountered. Finally, the mining process can be applied similarly to any other platform, as long as we have access to performance counters, temperature, and power monitoring units.

Regarding the evaluation of query Q1 (Equation 1), we created five workload mixes. Each mix consists of twelve applications from the Polybench [24] and Stream [25] benchmark suites and it is executed in the following way. Initially, four applications are launched on the four cores of the board. Once an application finishes its execution, it is replaced by another one until all twelve applications terminate. In that way, we are able to create operating conditions with varying pressure on the shared resources. Figures 4(a)-(d) depict the behavior of our method in terms of power consumption (W), temperature ($°C$), power efficiency (normalised $MIPS/Watt$), and normalised energy consumption against:

M1: The goal-driven method in [2]. This work utilizes a TDP of $4W$, justified by the need for thermal safety. This method is also the baseline of our experiments in Figures 4(c)-(d);

M2: method in [3], which uses Multi-Layer Perceptrons (MLPs) to set the operating frequency under specific power budgets. This method uses MLPs as classifiers, where their inputs are the run-time values of specific PMCs. In our experiments, we target the minimum power threshold of $5W$ that the authors also considered;

M3: The MRPI-based method in [11], which employs the MRPI as a classification metric to reduce energy consumption.

Figures 4(a)-(b) depict the shape of power and temperature distribution (respectively) per mix for its whole duration, as well as the corresponding median values observed. As mentioned, M1 considers $TDP = 4W$ and according to the results that it achieves in all mixes, it has a small variation. However, the temperature observations for M1 (Figure 4(b)) along all the selected mixes, show a minimum value of $58°C$ and a maximum value of $66°C$. This difference of $8°C$ validates our motivation that a single TDP value can result in large temperature variations. Similarly to M1, M2 also shows small variation in the resulting power, since it employs the usage of a neural network for different target frequencies, and is also characterized by power-awareness. At this point, it is important to mention that the authors mention in their paper that the MLPs are trained with a target power consumption of $P_{safe} = 4.8W$ as a "safety net" [3]. This leads to a more pessimistic predictions and therefore lower performance observations. Interestingly, M2 shows a minimum observed temperature of $60°C$ and a maximum temperature of $70°C$, which again validates our motivation for mining and setting the power budget based on the behavior of the workload. On the contrary, M3 does not aim to remain below a certain TDP value. This method adjusts the operating frequency by monitoring the MRPI value of the currently executed mix in order to reduce energy consumption without affecting its performance. However, if the current workload contains compute-intensive applications (low MRPI), whose performance is greatly affected by frequency scaling, the MRPI method does not reduce the operating frequency drastically. In our scenario, mixes 1, 2 and 3 contain both compute- and memory-intensive groups of workloads. Hence M3 in these cases shows great variation in power and temperature, since it will choose a lower frequency for compute-intensive groups, and a higher frequency for memory-intensive groups. Therefore the observed temperature variation for just one mix, (e.g. Mix 1), can be up to $20°C$. M3 results in low power and temperature observations for mixes 4, 5, since the MRPI value of these workloads is assigned to very low frequencies. However, they still yield variation in power, without significant gains in MIPS/Watt. Our method managed to keep the temperature variation low by utilizing the mined parameters for different power budget thresholds. Additionally, our method achieved an average gain of 11% regarding power efficiency over the second best M1 (Figure 4(c)). Similarly the energy consumption of our method is on average 15.4% and 35.3% lower than M1 and M3, respectively. Finally, our method decreases throughput by 2.72% compared to M1. Overall, satisfying a global power threshold may result in undesired temperature variations. Our experiments revealed that our systematic method, for associating mined power threshold parameters with the current workload,

TABLE I: Comparison against [2] for used defined applications

	3mm on [2]	3mm Pr.	stream on [2]	stream Pr.
Avg. Temp.	$61°C$	$57°C$	$66°C$	$60°C$
Avg. Pwr.	3.06 W	2.76 W	3.57 W	3.61 W
Norm. IPS gain	1	0.97	1	1.02

ensures thermal safety and increased power efficiency.

B. Results on fine-grain optimization (user-defined applications)

Regarding query Q2 (Equation 2), we selected two applications with different behavior that cover a wide variety of activity in modern embedded systems. Specifically, we utilized the `stream` benchmark [25] and the `3mm` (3 Matrix Multiplications) application from the polybench benchmark suite [24]. For all the selected applications, we performed the parameter mining considering $Temp_{safe} = 60°C$. Finally, we selected to run the parameter mining process by setting 100 iterations for the stochastic optimizer and we repeated this process 10 times. Similary to IV-A, this process is completed within an hour and needs to be done once per application. The evaluation scenarios are as follows. Initially the user starts the system running the desired application on one core along with some random co-runners. The co-runners can terminate at any time and new different applications replace them. For all scenarios, the user-selected application runs always on one core and with an allowable maximum performance drop of 10% ($Perf_{min} = 0.9 \times Perf_{max}$). At the same time, the overall throughput of each core, in terms of IPS, should be at least 2.9×10^6. We set this value according to [10] which utilized the same board. We compared the behavior of our system against the goal-driven method in [2] which also supports application performance thresholds. Table I depicts that, for the `3mm` compute-intensive application, our method dropped the performance by 3% compared to [2], since the achieved average power is lower. This is a direct result of our method's primary goal of thermal safety assurance. Indeed, our method manages to maintain an average temperature of $57°C$. For the `stream` memory-intensive application, we achieved lower temperature with approximately the same power consumption as [2]. In this case, our performance was higher by 2%. We consider the difference in performance negligible for both application scenarios, since our primary goal is to maintain an average temperature less than or equal to $Temp_{safe} = 60°C$.

V. Conclusions

In this paper, we present a framework for formal property exploration on clustered CMPs. Specifically, we utilize PSTL to express run-time constraints and mine multiple parameters in order to create optimized system- and application-aware operating points. Instead of having fixed values for power constraints, we enhance the run-time manager to consider different threshold values based on the mined parameters. At run-time, the allocator utilizes these operating points to select frequency in order to provide thermal safety and satisfy user requirements making reasonable trade-offs wherever necessary. Even though parameters vary for different systems, our method can be similarly applied to other CMPs.

VI. Acknowledgment

This research has been supported in part by grant NSF IIP 1361847 from the NSF I/UCRC for Embedded Systems at SIUC.

References

[1] Marisabel Guevara et al. Strategies for anticipating risk in heterogeneous system design. In *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, pages 154–164. IEEE, 2014.

[2] Elham Shamsa et al. Goal-driven autonomy for efficient on-chip resource management: Transforming objectives to goals. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*.

[3] Theodoros Marinakis et al. Meeting power constraints while mitigating contention on clustered multi-processor system. *IEEE Embedded Systems Letters*, 2019.

[4] Santiago Pagani et al. Thermal safe power (tsp): Efficient power budgeting for heterogeneous manycore systems in dark silicon. *IEEE Transactions on Computers*, 66(1):147–162, 2017.

[5] Hussam Amrouch et al. Optimizing temperature guardbands. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pages 175–180. IEEE, 2017.

[6] Oded Maler et al. Monitoring temporal properties of continuous signals. In *Proceedings of FORMATS-FTRTFT*, 2004.

[7] Ezio Bartocci et al. Specification-based monitoring of cyber-physical systems: A survey on theory, tools and applications. In *Lectures on Runtime Verification - Introductory and Advanced Topics*, volume 10457 of *LNCS*, pages 128–168. Springer, 2018.

[8] Sander Stuijk et al. A predictable multiprocessor design flow for streaming applications with dynamic behaviour. In *Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on*, pages 548–555. IEEE, 2010.

[9] Andreas Weichslgartner et al. Daarm: Design-time application analysis and run-time mapping for predictable execution in many-core systems. In *Hardware/Software Codesign and System Synthesis (CODES+ ISSS), 2014 International Conference on*, pages 1–10. IEEE, 2014.

[10] Basireddy Karunakar Reddy et al. Online concurrent workload classification for multi-core energy management. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*.

[11] Basireddy Karunakar Reddy et al. Inter-cluster thread-to-core mapping and dvfs on heterogeneous multi-cores. *IEEE Transactions on Multi-Scale Computing Systems*, 4(3):369–382, 2017.

[12] Jayasimha Sai Koduri et al. Spa: Simple pool architecture for application resource allocation in many-core systems. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*.

[13] Ryan Cochran et al. Pack & cap: adaptive dvfs and thread packing under power caps. In *2011 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 175–185. IEEE, 2011.

[14] Roberta Piscitelli et al. Design space pruning through hybrid analysis in system-level design space exploration. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012*.

[15] Daniel Olsen et al. Performance-aware resource management of multi-threaded applications on many-core systems. In *Proceedings of the on Great Lakes Symposium on VLSI 2017*, pages 119–124, 2017.

[16] Bryan Donyanavard et al. Sparta: Runtime task allocation for energy efficient heterogeneous manycores. In *2016 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS)*, pages 1–10. IEEE, 2016.

[17] Amit Kumar Singh et al. Resource and throughput aware execution trace analysis for efficient run-time mapping on mpsocs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*.

[18] Elham Shamsa et al. Goal formulation: Abstracting dynamic objectives for efficient on-chip resource allocation. In *2018 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*, pages 1–4. IEEE, 2018.

[19] Georgios E. Fainekos et al. Robustness of temporal logic specifications. In *Formal Approaches to Testing and Runtime Verification*, volume 4262 of *LNCS*, pages 178–192. Springer, 2006.

[20] Bardh Hoxha et al. Mining parametric temporal logic properties in model based design for cyber-physical systems. *International Journal on Software Tools for Technology Transfer*, 20:79–93, 2018.

[21] Eugene Asarin et al. Parametric identification of temporal properties. In *Runtime Verification*, pages 147–160. Springer, 2012.

[22] Georgios Fainekos et al. Robustness of specifications and its applications to falsification, parameter mining, and runtime monitoring with s-taliro. In *Runtime Verification (RV)*, 2019.

[23] S-taliro: Temporal logic falsification of cyber-physical systems. https://sites.google.com/a/asu.edu/s-taliro/s-taliro, 2015.

[24] Louis-Noël Pouchet. Polybench: The polyhedral benchmark suite. *URL: http://www. cs. ucla. edu/pouchet/software/polybench*, 2012.

[25] John D McCalpin. A survey of memory bandwidth and machine balance in current high performance computers. *IEEE TCCA Newsletter*, 1995.