RewardProfiler: A **Reward** Based Design Space **Profiler** on DVFS Enabled MPSoCs

Somdip Dey[1,3], Amit Kumar Singh[1], Sangeet Saha[1],
Xiaohang Wang[2], and Klaus Dieter McDonald-Maier[1]
[1]*University of Essex, U.K.*
[2]*South China University of Technology, China.*
[3]*Samsung R&D Institute, U.K.*
Corresponding E-mail: somdip.dey@essex.ac.uk

***Abstract*—Resource mapping on a heterogeneous multiprocessor system-on-chip (MPSoC) imposes enormous challenges such as identifying important design points for appropriate resource mapping for improved efficiency or performance, time consumption of exploring all the important design points for each profiled applications, etc. Moreover, incorporating a profiler into integrated development environments (IDEs) in order to achieve more detailed and accurate profiling information on the application being targeted during runtime such that improved efficiency or performance while executing the application is achieved, the runtime resource management decision to achieve such improved "reward" has to be utilized in a certain way. In this paper, we propose a hybrid approach of resource mapping technique on DVFS enabled MPSoC, which is suitable for IDE integration due to the reduced design points in our methodology resulting in significant reduction in profiling time. We coined our approach as "RewardProfiler" (a Reward based design space Profiler), which is well capable of reducing the design space exploration without losing most of the important design points based on our heuristic approach. In our strategy, an application has to be mapped onto the available resources in such a way so that the "reward" obtained can be maximized. Our approach can also be utilized to maximize multiple "rewards" (*Multivariate Reward Maximization*) while executing an application. Implementation of our *RewardProfiler* on the Exynos 5422 MPSoC reveals the efficacy of our proposed approach under various experimental test cases and has a potential of saving 170x more time in profiling for our chosen MPSoC compared to the state-of-the-art methodologies.**

***Index Terms*—Reward, profiler, IDE, design space exploration, multiprocessor systems-on-chip (MPSoCs)**

I. Introduction and Motivation

Fig. 1: Increase in design points using conventional profiler with increase in frequency scaling steps and number of CPU cores in MPSoC with 2 clusters

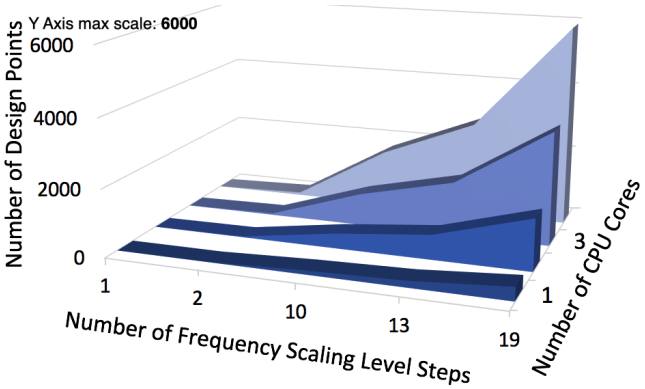
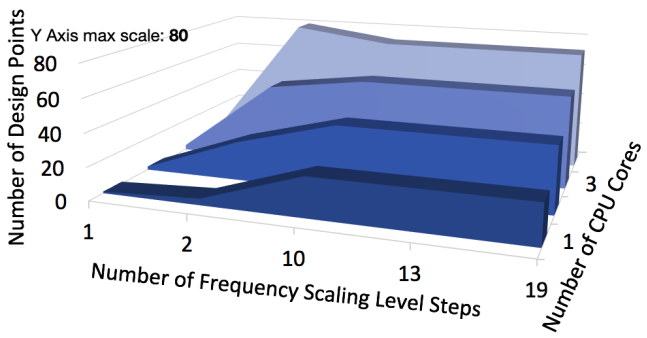


Fig. 2: Increase in design points using *RewardProfiler* with increase in frequency scaling steps and number of CPU cores in MPSoC with 2 clusters



Back in the decade of 2000 when Pentium processors were still dominating the commercial market share, profiling a software performance on such homogeneous processors based systems was easier than it is today. In a patent by Gove et al. [1], the proposed work was one of the earliest recorded methodology on automated profiling information in an integrated development environment (IDE). The workflow for this automated profiler was simple and efficient. However, over the decades, thanks to Moore's Law [2], now we can not just fit many cores on a single IC but we can also fit cores of different processing capabilities onto the same chip to better fit our needs. We could see an extensive use of heterogeneous multiprocessors systems-on-chips, where several types of processing cores are available within a single chip, to deliver performance as well as energy efficient computing, on embedded devices nowadays. Moreover the performance and energy efficiency demands of embedded applications has increased substantially [3] which could only be satisfied by executing them on such heterogeneous MPSoCs.

Now we could consider that mapping applications i.e. allocating resources such as CPU cores, memory, etc. to applications, could be classified under three categories [3]–[8]: Design Time mapping, On-the-fly (runtime) mapping and Hybrid mapping (combining both design time and on-the-fly mapping techniques). Each of these mapping techniques has their own benefits and disadvantages, such as design time mapping requires mapping the tasks in all

possible combinations of the available processing elements and given a large number of processing elements on a modern MPSoC the possible combination space is huge and hence mapping could not be performed at run-time. Whereas mapping tasks at run-time in most cases can not provide the most optimized resource allocation since it could be missing some combination of resources, which could lead to better suitability of the requirements such as performance, energy consumption, etc. In the hybrid mapping techniques we could see that researchers are leveraging benefits of both design time and run-time mapping to allocate resources to applications.

Since, energy efficiency are some of the biggest concerns of embedded computing devices, majority of the state-of-the-art methodologies utilize a combination of the aforementioned mapping policies to increase the energy efficiency of the system, but according to [9] there are five popular methods leading to energy reduction in the system, which includes:

1) *Dynamic Power Management (DPM)* allows idle processing elements or other idle components of the system to be suspended if required in order to reduce energy consumption [10].
2) *Dynamic Voltage Frequency Scaling (DVFS)* allows processors to operate at variable voltage-frequency (v-f) levels [11].
3) Customization of processors to match the processing needed of a task on an MPSoC [12].
4) Customizing cache based memory access [13].
5) Mapping tasks of an application to the processors so that workload could be balanced across all processors in an MPSoC. This improves utilisation of PEs effectively and reduces energy consumption [14].

At the operating system level, we could only use DPM and DVFS based methodologies for energy consumption regulation, which could further add to resource mapping and allocation techniques. Therefore, software based profiling systems mostly utilize these two methodologies to profile energy consumption for a set of applications on heterogeneous MPSoCs. Moreover, if we consider frequency scaling level steps[1] of DVFS then as the number of CPU cores increases in the MPSoC, the number of design points also increases polynomially. For example, if we consider similar CPU cores are clustered together into heterogeneous architecture where cluster wise frequency scaling is available then Fig. 1 shows the increase in the number of design points as the number of CPU cores in the clusters and the frequency scaling steps increases for a heterogeneous MPSoC with 2 CPU clusters.

Meanwhile, the emergence of more and more intelligent integrated development environments (IDEs) [15], [16] enables profiling for both performance and energy efficiency of the application. However, the issue is none of the hybrid resource mapping techniques are suitable enough to be utilized in such IDEs due to the fact that most of the state-of-the-art hybrid resource mapping techniques still incorporate a huge number of design space combinations [17], [18], which could drastically increase the profiling time. On the other hand given more and more application development teams are adopting scrum based agile software development [19], [20], it would be very useful to develop efficient and close to accurate resource mapping and profiling techniques for applications incorporated into IDEs. Although it could be argued that any state-of-the-art hybrid resource mapping could be adopted in IDEs to get profiling results and then design applications accordingly but given the fast development environment of Agile[2], it is crucial to develop hybrid mapping techniques, which are fast both in design and run-time. For example, we could consider the case from Fig. 1, if 7 applications (consisting of a set of tasks) requires to be profiled for several devices consisting of different number of CPU cores and frequency scaling levels then there are 139,440 total design point mappings, which would take 38.74 hours (approx.) to profile exhaustively considering it takes 1000 ms (1 sec) to evaluate one mapping. This means that each application or set of tasks requires 5.6 hours (approx.) of profiling time and even the slightest modification to the tasks would also require similar amount of time in profiling every time a modification is made. Given the constrained time period for Agile application development, profiling time of 5.6 hours for a set of tasks is unacceptable. Whereas, in comparison when we use our approach we achieve 5,194 design points for the same 7 applications (see Fig. 2), which would take 1.44 hours (approx.) instead. It also means that each application would roughly take 12.37 minutes for profiling and hence, we could save 28x more time in profiling for this test case compared to traditional approach. The way we achieve this speedup in profiling time is by clustering the frequency scaling levels based on approximate computing [22], [23] using our heuristic approach. Later, we also show in Sec. IV that using this approach of profiling we could further reduce the profiling time to 170x in comparison for Exynos 5422 MPSoC [24] and thus each application would take 2.04 minutes (approx.).

According to our current knowledge there is no available scientific document, which focuses on developing hybrid resource mapping and profiling techniques on heterogeneous MPSoCs that is more catered towards being incorporated into IDEs. So the challenges that we face in this research are as follows:

1) Reduce the time to profile combination of resources during the design space exploration (DSE) without loosing important design points.
2) Design a hybrid resource mapping technique on heterogeneous MPSoCs, which could incorporate reduced design space exploration and then utilize the knowledge for efficient run-time resource management decision.

In order to address these challenges, we propose a hybrid resource mapping technique on DVFS enabled MPSoC, “RewardProfiler", where we reduce the design space explo-

[1]Using DVFS the frequency and voltage could be regulated to control energy consumption, where each changing frequency level is called frequency scaling level and the number of operating frequencies that could be changed is called frequency scaling level steps.

[2]In Scrum Sprint Agile development the maximum development time for a feature (iteration) in the application could vary from 2-6 weeks [21]

ration and making it suitable for IDE integration. We coined the proposed methodology as RewardProfiler for it being a “reward" based design space exploration profiler. Here, the term “reward" is a beneficial objective that needs to be achieved during the profiling period of the applications or set of tasks such as execution time, energy consumption, thermal gradient, etc. (see Sec. II-C). To this end, this paper makes the following novel contributions:

1) Design a hybrid mapping and profiler methodology addressing the aforementioned challenges such that it could be incorporated into IDEs.
2) Validate the methodology on Exynos 5422 SoC [24], which is a popular heterogeneous MPSoC.

The rest of the paper is organized as follows. Section II discusses about the hardware infrastructure used and problem formulation. In Section III we discuss about our proposed method RewardProfiler and in Section IV we validate our approach with different experiments. Finally, we discuss some noteworthy findings of this study in Section V and conclude in Section VI.

II. System Model and Problem Formulation

A. Hardware Infrastructure

Now a days 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 ISA of the core is 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 [25]–[28]. Specifically, for our study we chose the Odroid XU4 board [29], which employs the Samsung Exynos 5422 [24] MPSoC. This MPSoC 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 DVFS 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. This MPSoC also contains 6 ARM Mali-T628 GPU shader cores based on the “Midgard" architecture and a DRAM of 2GB LPDDR3.

B. Exploration of Combined Design Points

To gain maximum benefit of task, thread and data parallelism of applications to be executed on a CPU-GPU heterogeneous multi-processor systems-on-chip several threads could be partitioned between the CPU and GPU, using different combinations of the big/LITTLE CPU cores and/or the GPU cores. If we consider ARM’s big.LITTLE architecture and assume that there are n_b big and n_L LITTLE cores then the total number of mappings (M_{CPU}) [17], [31]:

$$M_{CPU} = (n_b + n_L) + (n_b \times n_L) \tag{1}$$

Since, at the moment we are not able to segregate the workload on the GPU and map them onto individual cores separately due to limited driver support by ARM, we would consider the total number of mapping on the GPU (M_{GPU}) as 1. Now, since Exynos 5422 supports cluster wide DVFS, the cores within each cluster can run on a chosen voltage-frequency[3] from a pre-defined set of voltage-frequency value pairs. If we assume that there are f_b, f_L and f_{GPU} number of frequency scaling levels for the big, LITTLE and GPU cluster, respectively, then considering the voltage-frequency scaling levels for the CPU and GPU cores, the mapping design space for CPU ($M_{CPU_{VF}}$) and GPU ($M_{GPU_{VF}}$) will consist of the following:

$$M_{CPU_{VF}} = ((n_b \times f_b) + (n_L \times f_L)) + (n_b \times f_b \times n_L \times f_L) \tag{2}$$

$$M_{GPU_{VF}} = 1 \times f_{GPU} \tag{3}$$

Thus from the equations Eq. 1, 2 and 3 we could derive the total number of Combined Design Points (CDP) considering both the CPU and GPU cores are as follows:

$$\begin{aligned} CDP &= M_{CPU_{VF}} \times M_{GPU_{VF}} \\ &= \{((n_b \times f_b) + (n_L \times f_L)) + (n_b \times f_b \times n_L \times f_L)\} \times (f_{GPU}) \end{aligned} \tag{4}$$

From the equation Eq. 4 we could generalize combined design points for all asymmetric multicore processors systems-on-chip having clustered cores with DVFS capabilities[4] and the GPU on such system acts as single unit for operation[5] due to GPU driver limitations. If we consider N as the number of clusters in such AMPSoC, f_C be the number of frequency scaling levels for the whole cluster[6] and n_C be the number of cores in each cluster then the generalized equation governing such combined design point (CDP) consideration is as follows:

$$\begin{aligned} CDP = (\sum_{C=1}^{N} n_C \times f_C + \prod_{C=1}^{N} n_C \times f_C) \times f_{GPU}, \\ where \quad 1 \leq N \leq n_C \end{aligned} \tag{5}$$

In Eq. 5, the governing equation for CDP only works for DVFS enabled multi-core architecture, which means n_C and f_C is always more than 1. It could be inferred from the Eq. 5 that the number of clusters (N) present in the system is either less than or equal to the number of cores (n_C) present in each cluster. The reason to provide this constraint is because from design point of view it is more practical to have more number of cores on the die than the number of clusters due to die size constraint.

For ease of understanding of our heuristic based *RewardProfiler*, we would be using a simplified version of Eq. 5,

[3] We only choose the frequency and the firmware automatically adjusts the voltage based on pre-set value pairs of the voltage-frequency.

[4] We also assume that each clustered cores run at the same frequency as the other cores in the cluster

[5] n_{GPU} is equal to 1

[6] For this instance, we are only considering cluster wise DVFS capability.

where we only consider the MPSoC consisting of heterogeneous multiple CPU cores, and the modified equation could be represented as follows:

$$CDP = \sum_{C=1}^{N} n_C \times f_C + \prod_{C=1}^{N} n_C \times f_C, \quad where \;\; 1 \leq N \leq n_C \tag{6}$$

Nevertheless, our *RewardProfiler* approach could be extended to MPSoC having CPU/GPU cores on-chip, henceforth utilizing Eq. 5.

C. Reward Based Mapping

Now, let us assume that an application (A) under the system consideration consists of m tasks $A = \{T_1, T_2, ..., T_m\}$, which is required to be scheduled on CPU_{Total} ($CPU_{Total} = \sum_{C=1}^{N} n_C$) heterogeneous CPU cores on the MPSoC such that we receive a reward R_{App}. Here, a reward R_{App} corresponding to an application is the user defined parameter of the system, which may represent any such following terms: performance of an application or energy consumption or operating temperature of the CPU cores. We could achieve the maximum value of the reward ($R_{App_{Max}}$) if and only if we schedule the tasks of the application appropriately to the CPU cores. Here, mapping a set of tasks of the application to appropriate CPU cores also solves the problem of symmetry-elimination[7] in DSE [32]. We also need to keep in mind that for some chosen reward type the objective would change to minimizing the reward instead in order to gain most benefit for the overall system. For example, if we choose low operating temperature as the reward, which we want to gain from appropriate task-to-core mapping, then the beneficial objective would be to achieve the least possible operating temperature of the cores.

Given the Eq. 5 the number of possible combinations of mapping over design space is huge and if we consider that an instance of combined design point mapping could be represented as CDP_i then each such instance could lead to a possible reward R_i (see Eq. 7). NOTE: While defining the concept of reward we have restricted resource mapping to CPU cores at the moment for the ease of understanding. However, the same approach could be extended for the GPU cores in the MPSoC as well.

$$CDP_i \longmapsto R_i \;\; , \;\; \forall R_i \in R_{App} \tag{7}$$

D. Problem Formulation

Given a set (App^s) comprising of s applications such that $App^s = \{App^1, App^2, ...App^s\}$, where $App^i \in App^s$ and App^i is an instance of an application. We can now mathematically formulate the problem with two objectives as follows:

Objective Function:

1) *Maximize* R_{App^s}, i.e. achieve R_{Max}
   Where $R_{Max} = max(\; R_{App^1_{Max}}, R_{App^2_{Max}}, ..., R_{App^s_{Max}} \;)$
2) *Reduce CDP*, i.e. achieve $CDP_{Reduced}$, which are the reduced design space points (mapping)

Subject to following **Constraint**:

- Resource allocation does not exceed available MPSoC resources.

Note: Depending on chosen reward, the objective might be to reduce the reward instead of maximizing so that overall benefit could be achieved.

[7]Symmetry-elimination means eliminating the design point mappings which are redundant in DSE.

III. Proposed Methodology: RewardProfiler

A. Overview of RewardProfiler

In our *RewardProfiler* we reduce the number of design space points (mapping) (see Eq. 5) by segregating the design space into small clusters, which could lead to almost similar reward generation. We could think of this with an example of taking a slab of glass and then hammering it to break it into pieces. Then we group the broken pieces of glasses based on almost similar sizes.

Since *RewardProfiler* is considering DVFS for this approach, our profiler only modifies the number of CPU core allocation from the design space and the frequency level at which the cores run. Based on approximate computing we have clustered the different frequency scaling levels into four major groups. From our experimental data and analysis we have observed that the human eye is able to differentiating a video of 30 frames per second (fps) with 15 fps very easily, whereas it becomes very difficult to differentiate between videos of 13 fps and 15 fps. The same observation could be extended about execution time as well. Until and unless a workload execution is time-critical, difference between execution time of 39 secs and 40 secs is minimal, whereas, the difference of execution time for the same workload of 39 secs and 50 secs is quite noticeable. From our experiments we have noticed that some frequency scaling level steps produce almost similar *Reward* generation and we have clustered such frequency scaling levels together to represent the outcome i.e. *Reward* generation, of a group of frequency levels. Therefore, we are utilizing the core essence of Approximate Computing [23] in our approach and the four different frequency scaling levels which are utilized in our approach are as follows: highest frequency (F_h), medium high frequency (F_{mh}), medium low frequency (F_{ml}) and lowest frequency (F_l). Here, highest frequency is the maximum frequency at which the cores could run whereas the lowest frequency is self explanatory as well. In order to decide on medium high frequency and medium low frequency we have defined two equations (Eq. 8 and Eq. 9).

$$F_{mh} = frequency(\; \lceil (f_C/2) \rceil + 1 \;) \tag{8}$$

$$F_{ml} = frequency(\; \lfloor (f_C/2) \rfloor - 1 \;) \tag{9}$$

We could notice from Eq. 8 and Eq. 9 that we take the frequency at ($\lceil (f_C/2) \rceil + 1$) and ($\lfloor (f_C/2) \rfloor - 1$) frequency levels for F_{mh} and F_{ml} respectively. From our extensive experiments we have also observed that merely selecting just the middle frequency scaling level does not accurately represent the clustering of *Reward* being generated in the set of frequency

levels in the middle. Selecting just the middle could lead to eliminating many important design points and associated *Reward* generation. Hence, for *RewardProfiler* F_{mh} and F_{ml} are selected. Section V shades some light on this ideology.

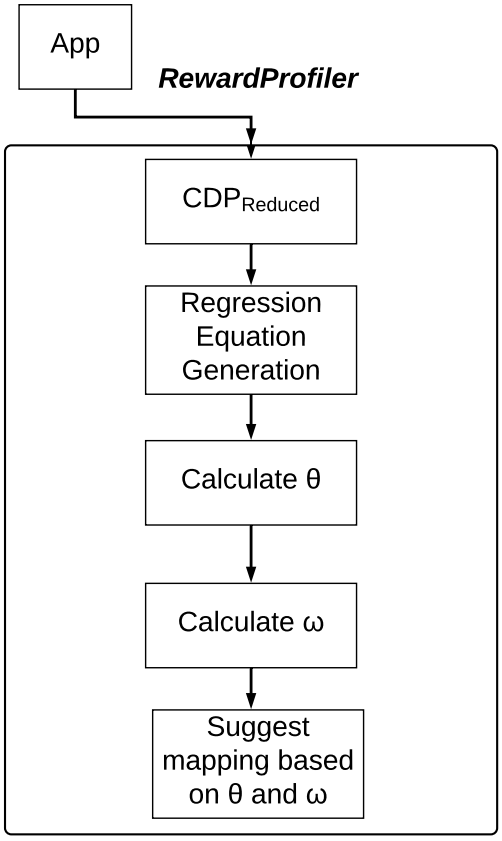


Fig. 3: RewardProfiler Workflow

In our methodology we reduce the design points by only considering the aforementioned 4 frequency levels[8]. Therefore, from Eq. 6 we could deduce the combined design points as:

$$CDP_{Reduced} = \{((n_b \times 4) + (n_L \times 4)) + (n_b \times 4 \times n_L \times 4)\} \quad (10)$$

We can notice in Eq. 10 that we have not considered the GPU cores at all and the reason being that to execute application on both CPU and GPU cores specialized code or framework[9] is required to identify the part of the code or set of tasks to execute on the GPU or CPU.

Now, our *RewardProfiler* have to execute the application on different number of CPU core configuration with the aforementioned frequency levels and check for the values of different reward (R_i) achieved for every configuration. However, before evaluating different reward the user inputs the threshold reward ($R_{threshold}$), which is the minimum value of the reward that is acceptable by the user for a desirable quality of service and is user defined during the profiling period. If the reward (R_i) achieved is equal or more than the threshold reward ($R_{threshold}$) then we consider that configuration for further computation or else the resultant values (R_i, operating frequency F_i) are stored in a table. In case we have several configurations, which are capable of generating rewards either equal or more than the threshold reward then the *RewardProfiler* could calculate the difference in rewards to get the best possible configuration and reward combination.

Note: Depending on the goal of reward generation the motive behind the threshold reward ($R_{threshold}$) could also change. For example, if a performance constraint is required to be met then the associated threshold reward is the maximum value of allowed execution time for the profiled application. In this case the threshold reward is the maximum value of the reward that is acceptable by the user for a desirable quality of service.

Let us consider R_i as the reward for an instance of a configuration. We could then represent R_i as a linear function (see Eq. 11) of the operating frequency (F_i), since we could only regulate the frequency of the running CPU cores[10].

Algorithm 1: RewardProfiler Execution

Input:
1. App^s: set of s applications
2. R_{App^s}: set of rewards chosen for each application in App^s (see Sec. II-D)

Output:
1. Mapping Combination: *Combination of frequency level, operating cores (big / LITTLE) such that* R_{max} *is achieved, where* $R_{max} = max(\ R_{App^1_{Max}}, R_{App^2_{Max}}, ..., R_{App^s_{Max}}\ )$

begin
- Compute the reduce number of $CDP_{Reduced}$ using Eq. 9 ▷ Each instance of CDP will represent a combination of chosen cores and frequency level, say one application chosen 1 big and 1 LITTLE cores with frequency (F_i) then this instance will map to a particular reward value R_i
- **for** *Each application* $App^i \in App^s$ **do**
  - Obtain reward (R_i) corresponding to each CDP_i as per Eq. 6
  - **for** *Each obtained reward* (R_i) **do**
    - **if** *Achieved* $R_i \geq R_{threshold}$ **then**
      - Store in a table with the acheived R_i;
      - Calculate the angular degree (θ_i)using Eq.11;
    - **else**
      - Store in a table with the acheived R_i
  - Find the difference (ω_i) between the angles using Eq. 12;
  - Find the minimum value (ω_{least}) from all instances of ω mentioned above;
  - /* $\omega_{least} = \theta_{least_i} - \theta_{least_{i-1}}$ */
  - Find $\theta_{least} = \min(\ \theta_{least_i}, \theta_{least_{i-1}})$ and then fetch corresponding mapping combination of the minimum value of θ_{least}; ▷ This particular step is carried out in function $fecthMapping()$

[8] 4 frequency scaling levels are F_h, F_{mh}, F_{ml}, F_l.

[9] Frameworks such as CUDA or OpenCL are required to execute tasks on GPU cores.

[10] The firmware in the system automatically adjusts the voltage based on pre-set value pairs of the voltage-frequency once the frequency is chosen.

$$R_i = \alpha_i \times F_i + \beta_i, \forall F_i \in \{F_h, F_{mh}, F_{ml}, F_l\} \quad (11)$$

Using the Eq. 11 if we assume that for a different operating frequency (F_{i+1}) a new reward (R_{i+1}) is generated but the α ($\alpha_i = \alpha_{i+1}$) and β ($\beta_i = \beta_{i+1}$) values remain the same during the profiling period, the value of α and β could be evaluated from the linear functions of R_i and R_{i+1}. After computing the α and β values from the Eq. 11 for every possible frequencies[11] *RewardProfiler* have to calculate the angular degree of the slope of the linear function (see Eq. 12). The reason to calculate the angular degree (θ) of the slope is to check for the rate of decrease or increase in angular degree between different slopes ($\alpha_{i-1}, \alpha_i, \alpha_{i+1}, ...$) achieved through different reward ($R_{i-1}, R_i, R_{i+1}, ...$) generation in order to decide which frequency level to choose for maximized reward while minimizing design space for resource allocation.

$$\theta_i = \arctan(\alpha_i) \quad (12)$$

In Eq. 12 θ_i is the angular degree of slope, α_i, of reward, R_i. After calculating θ for every possible combination from $CDP_{Reduced}$, given that the reward generated is equal to or more than the threshold reward ($R_{threshold}$), the profiler fetches the difference between two adjacent θ values. Now, we could assume that the difference between two adjacent θ to be ω. Therefore, if we assume that the profiler needs to calculate ω_i for every θ_{i-1} and θ_i (see Eq. 13), then after calculating ω for all desired rewards the profiler would suggest the mapping combination from $CDP_{Reduced}$ where ω is the least (ω_{least}, see Eq. 14). For instance a higher value of ω_i would mean that the difference in slope of two different rewards (R_{i-1}, R_i) is high and means that for CDP_{i-1} & CDP_i the rewards (R_{i-1}, R_i) generated are significantly different. However, in comparison if the value of ω_i is minimal then it means that for CDP_{i-1} & CDP_i the rewards generated do not defer a lot and hence, it is more beneficial to choose the CDP_{i-1}. Using this mathematical approach we can guarantee selection of almost minimized design space for resource allocation for almost maximized reward generation using linear regression based technique.

$$\omega_i = \theta_i - \theta_{i-1} \quad (13)$$

$$\omega_{least} = min(\ \omega_1, \omega_2, ...\omega_i\ , ...\omega_n) \quad (14)$$

To predict our heuristic based threads-to-core mapping, the profiler suggests the mapping combination from the combined design point ($CDP_{Reduced}$) that generates θ_{i-1}, which leads to ω_{least}, and the operation that leads to this suggestion of mapping resources is called $fetchMapping$ (see Eq. 15).

$$Mapping\ Combination = fetchMapping(\ \omega_{least}\ ) \quad (15)$$

The aforementioned work-flow and pseudo-code of the proposed *RewardProfiler* could be found from Fig. 3 and Algo. 1 respectively.

[11] 4 different frequencies for each cluster of cores in our case.

B. Multivariate Reward Profiler & Multivariate Reward Maximization

The proposed approach of *RewardProfiler* also works for multiple types of reward maximization and we call this approach of the *RewardProfiler* as *Multivariate Reward Maximization* (MRM). In *Multivariate Reward Maximization* different types of "reward" with associated reward threshold are selected such that the *RewardProfiler* could try to maximize each type of reward for the profiled application while trying to reduce the design space for resource allocation. An example of such use case of *Multivariate Reward Maximization* is provided in Sec. IV along with its experimental results.

Note: Although the term *Multivariate Reward Maximization* is used in this case but for rewards where the benefit for the system is to minimize the reward during profiling period such as reducing thermal gradient of the system while executing an application, then the *RewardProfiler*'s goal is to minimize the value of reward in that case while reducing design space for resource allocation.

IV. Experimentation and Validation Results

A. Experimental Setup

Applications can be classified into three categories [17]: compute intensive, memory intensive and mixed load (both compute and memory intensive). To validate our proposed methodology we implemented the *RewardProfiler* on Exynos 5422 [24] (see Sec. II-A).

In order to validate the effectiveness of our *RewardProfiler*, Streamcluster (a data-mining benchmark of PARSEC [33]) with *native* & *simlarge* options is chosen. The reason behind this selection lies on the fact that the Streamcluster represents a real world mixed load application and its execution period is quite long to observe different operations required for chosen reward type in the system. Along with Streamcluster we also utilized facesim (with *simdev* & *simlarge* options), vips and x264 benchmarks from PARSEC to validate our methodology. We have also chosen another real world mixed-load application to get profiling suggestion from the *RewardProfiler*. We have used a face detection algorithm based on Haar Cascade [34] to detect faces and eyes on a video of 320 × 240 dimension for the aforementioned purpose. We have ran all our experiments for validation on UbuntuMate version 18.04 LTS (Linux Odroid kernel: 4.14.37-135).

In the the validation results (see Sec. IV-B) and in the discussion sections (see Sec. V) we have only provided a snapshot of the total execution time due to repetitive pattern of result in all the graphs. We executed the benchmark applications on all eight cores (big.LITTLE) of our Exynos 5422 SoC platform for different frequencies (frequency belonging to F_h, F_{mh}, F_{ml}&F_l). Due to this approach we could save at least 14.17x more time than usual offline resource mapping techniques using combined design space mapping for our chosen platform. For Exynos 5422 considering big and LITTLE CPU cores, and GPU core as unity with only one frequency level, there could be 4080 (see Eq. 16) combined design points (CDP) whereas, for our reduced

CDP ($CPD_{reduced}$) it would be just 288 points (see 17) based on our experimental setup.

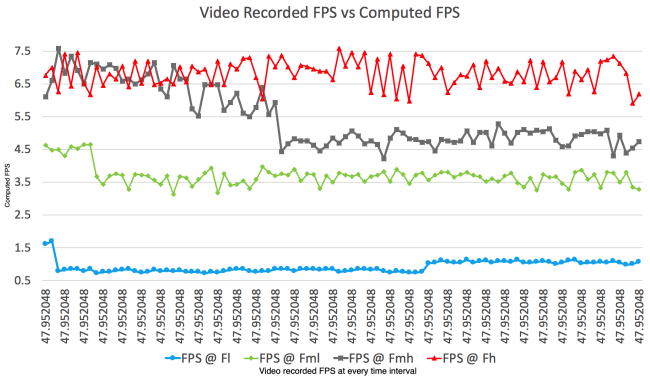
$$CDP_{Reduced}Cal. = \{((4 \times 19) + (4 \times 13)) + (4 \times 19 \times 4 \times 13)\} \quad where,\ n_b = 4, n_L = 4,\ f_b = 19, f_L = 13 \tag{16}$$

$$CDP_{Reduced}Cal. = \{((4 \times 4) + (4 \times 4)) + (4 \times 4 \times 4 \times 4)\} \quad where,\ n_b = 4, n_L = 4 \tag{17}$$

We could further reduce these combined design points by allocating set of tasks to the clusters of big.LITLLE instead of allocating them to each CPU cores. Therefore, $n_b = 1, n_L = 1$ instead. This would also ensure that symmetry-elimination [32] is solved and the *RewardProfiler* do not waste time in profiling similar resource allocation more than once. Thus, the total number of CDP that would be profiled by the *RewardProfiler* in this case would be only 24 as shown in Eq. 18 using Eq. 6. Using this approach we are saving 170x ($= \frac{4080}{24}$) more time in profiling than usual offline resource mapping techniques using combined design space mapping for our chosen platform.

$$CDP_{Reduced}Cal. = \{((1 \times 4) + (1 \times 4)) + (1 \times 4 \times 1 \times 4)\} \tag{18}$$

Fig. 4: Computed fps at 4 different frequency levels: F_h, F_{mh}, F_{ml}, F_l



B. Validation Results

1) Using RewardProfiler with single reward: For Streamcluster in $native$ mode we chose operating temperature as our reward and the objective of the *RewardProfiler* was to get the best possible combination of CPU cores and frequency level for which we could achieve low operating temperature (threshold of 90°C is selected, which is the operating thermal cap of the system) without compromising much on performance based on our heuristic approach. We achieved an operating temperature of 89.79°C (avg.) at F_{mh} frequency level (big cores running at 1200 MHz and LITTLE cores running at 900 MHz) for Streamcluster with an execution time of 393.64 seconds. *RewardProfiler* was able to reduce the operating temperature (reward) to 89.79°C (avg.), which was within 2.034% (avg.) of the optimal value

TABLE I: Avg. fps at different frequency levels

Freq. Levels	Avg. fps
F_h	8.709
F_{mh}	6.171
F_{ml}	4.440
F_l	0.914

TABLE II: α, β, θ at different frequency levels

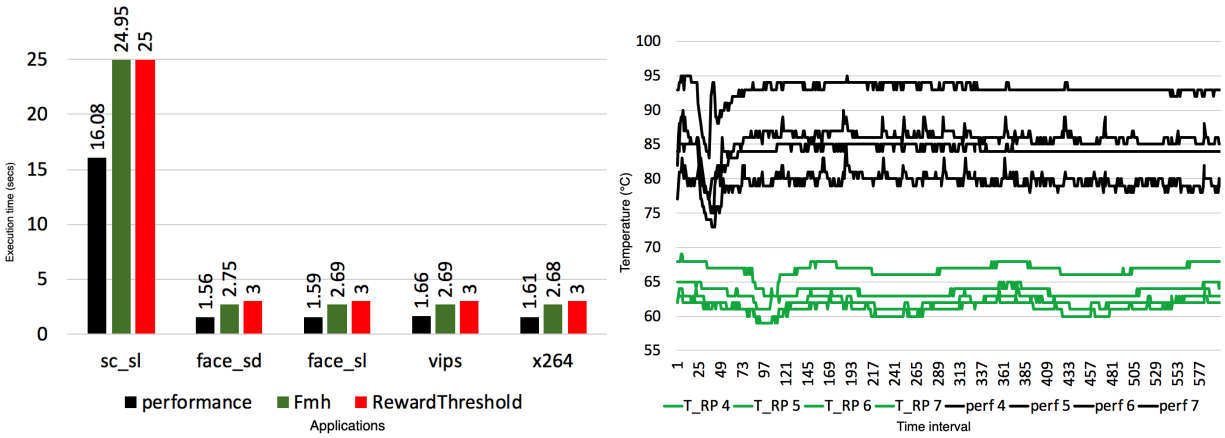
Frequency Levels	α	β	θ
F_h	0.0001	8.6613	0.001
F_{mh}	0.0002	5.4304	0.011
F_{ml}	0.0013	3.9309	0.074
F_l	0.0027	0.7853	0.154

(88°C avg.) of the operating temperature on the Odroid XU4 along with optimal execution time (392.01 seconds). *RewardProfiler* only required to profile the Streamcluster application 24 times (see Eq. 18) instead of profiling the application for 4080 times (see Eq. 16) to find a reward close to the optimal reward (~2.034% difference) and hence, reducing the profiling time by 170x.

For face detection application we chose computed fps (frames per second) to be our reward and the objective of the *RewardProfiler* is to maximize fps while keeping resource mapping to least. For this application we chose our threshold value to be 6 fps. We found out that according to *RewardProfiler* the best frequency level is at F_{mh} and running the application on all eight CPU (big.LITTLE) cores. Tables I and II show the corresponding average computed fps and different values of parameters of regression expression (see Eq. 11) and θ for different frequency levels. Fig. 4 reflects the aforementioned average fps values in a graphical representation for 4 different frequency levels (F_h, F_{mh}, F_{ml}&F_l). In this experiment, both the big and LITTLE clusters were set to F_h, F_{mh}, F_{ml}&F_l respectively at the same time and the results in Fig. 4 reflect the achieved fps for these operating frequency selection. In the figures (4, 6 & 7) the video recorded fps, which represents the fps of the original video when recorded, is presented on the X axis and the Y axis reflects the computed fps during profiling.

2) Using RewardProfiler with multiple rewards: We also chose multiple rewards to profile the following benchmarks from PARSEC: Streamcluster with *simlarge* option, faceism with simdev & simlarge options, vips and x246. For multiple rewards we chose the execution time and the thermal gradient of the system as the *Rewards*. For execution time we selected different reward threshold for each applications, which are being profiled. In this test case, instead of choosing the reward threshold as the value for which the *RewardProfiler* should try to generate rewards, which is more than the threshold, the threshold is the maximum allowed value for that specific reward. Therefore, for the chosen reward threshold if the reward generated surpasses the threshold value then the *RewardProfiler* do not consider those rewards. Even for the reward where thermal gradient of the system is chosen, the threshold value is the operating thermal cap of the system. The goal of the *RewardProfiler* is to find the operating frequency for which the system can achieve an overall reduction in thermal gradient (spatial and temporal) of the system while meeting performance deadline (execution time) for the respective benchmark applications. Therefore, for these test cases the *RewardProfiler* has to meet both performance and thermal constraints and hence, these are a multi-reward examples.

Fig. 5 shows the resultant rewards generated for each



(a) Reward: Execution Time

(b) Reward: Thermal Gradient

Fig. 5: *RewardProfiler*: Profiling with multiple rewards (Execution Time & Thermal Gradient)

benchmark applications, which were profiled. In the Fig. 5.(a) *sc_sl* represents the results for Streamcluster with *simlarge* option, *face_sd* represents the results for facesim with *simdev* option, *face_sl* represents the results for facesim with *simlarge* option, *vips* represents the results for vips benchmark and *x264* for the x264 benchmark. The execution time in Fig. 5.(a) are in seconds. For the reduction in thermal gradient the chosen reward threshold was 70° C, which acted as the thermal cap for the system. In the Fig. 5.(b) perf 4, 5, 6, 7 represent the operating temperature of the 4 ARM Cortex A-15 big CPU cores respectively while profiling the aforementioned benchmarks in performance governor of linux, whereas the *T_RP* 4, 5, 6, 7 represent the operating temperature of the 4 ARM Cortex A-15 big CPU cores respectively while profiling the benchmarks in F_{mh} for big CPU cluster and F_{ml} for the LITTLE CPU cluster using *RewardProfiler*. From Fig. 5 we could observe that *RewardProfiler* is able to reduce the thermal gradient of the system by 27.37% and reduce thermal cycle by 72.31% while achieving execution time by performance deadline (reward threshold for execution time). All the temperature readings are in ° Centigrades and we could only profile the temperatures of the 4 ARM Cortex A-15 big CPU cores because the temperature sensors are only available on the 4 big CPU cores and 1 on the GPU of the Odroid XU4 platform.

We also compared our *RewardProfiler* with state-of-the-art TheSPot methodology [35], which is a thermal stress-aware power and temperature management approach for MPSoCs. Table III shows the average reduction in thermal gradient achieved using TheSPot methodology for facesim, vips and x264 benchmarks. Since we profiled several applications (Streamcluster, facesim, vips and x264) in the same profiling session using *RewardProfiler*, Table III reflects the average reduction in thermal gradient (spatial and temporal) for the whole profiling session instead of individual profiling of each benchmark application. In the study by Iranfar et al. [35] TheSPot approach is implemented in two ways: Optimal TheSPot and Heuristic TheSPot methodologies. In the Table **??**, O.TheSPot represents the results for Optimal TheSPot methodology, H.TheSPot represents the results for Heuristic TheSPot methodology, whereas, R.Profiler represents the results for our *RewardProfiler*. Since TheSPot is able to achieve different temperature reduction for spatial and temporal thermal gradient, the table reflects the average reduction of spatial and temporal thermal reduction in percentage (%). We could observe that TheSPot method is able to reduce the thermal gradient by 15.83% on an average if we take average of all the results, whereas, our *RewardProfiler* is able to achieve 27.37% reduction in thermal gradient while meeting the performance constraint (execution time threshold). Therefore, *RewardProfiler* is able to outperform TheSPot DPTM[12] by 72.87% for this experimental setup.

TABLE III: Average reduction in thermal gradient comparison between TheSPot and *RewardProfiler*

Thermal gradient reduction (%)			
	O.TheSPot	**H.TheSPot**	***RProfiler***
facesim	16	21.5	27.37
vips	9.5	13.5	27.37
x264	13	21.5	27.37

3) Comparative study: Although in Sec. IV-B2 we have compared a test case of using *RewardProfiler* with multiple rewards to the TheSPot thermal management methodology to show the efficacy of the proposed methodology in reducing thermal gradient, being selected as one of the chosen rewards. In this section we compare *RewardProfiler* with some of the state-of-the-art methodologies dealing with reduction in design points in design space exploration (DSE) [32], [36], [37].

In the study [36], Shahid et al. propose a methodology to utilize approximate computing in DSE. Since majority of proposed studies in traditional DSE [38]–[40] try to optimize energy consumption while achieving the best performance/throughput as the multi-objectives, [36] also

[12]DPTM stands for Dynamic Power and Thermal management methodology.

focuses on the same objectives. Unfortunately, modern MPSoCs utilized in embedded devices are now capable of doing much more due to immense improvement in chip technology. Now a days we could see that devices from smart-phones to laptops, which have become an integral part of most human beings' life, can utilize similar MPSoCs as the computational resource of the device. Therefore, DSE should not just focus on optimizing energy consumption while achieving the best throughput but should also consider optimizing other factors such as thermal gradient reduction for improved device reliability and hardware security for improved confidence in communication and storage of data on the device and between devices. In [36], the methodology executes several optimization techniques used in DSE in time-parallel to fetch the inputs to be used to train a neural network, which in terms generate an approximate solution. Here, the approximate solution is the optimized configuration/design points which generates the reduced energy consumption while maximizing throughput. Comparatively, *RewardProfiler* is capable of accommodating multiple objectives, which in terms is known as the rewards for the methodology, and is not just limited to minimizing energy consumption and maximizing performance. *RewardProfiler* is also faster in execution because every time when the program code changes or the hardware architecture of the device changes then the neural network has to be retrained, which is itself a time consuming process. Moreover, the methodology provided in [36] is evaluated through simulation, whereas, *RewardProfiler* is implemented on real hardware platform, which provides more confidence in the methodology and it's efficacy.

In the study [37], Rosvall et al. propose a methodology to find design point implementations for a set of streaming applications on a shared multiprocessor platform which guarantee required performance by leveraging constraint programming approach and data-flow to find optimal mappings. Hence, [37] achieves to meet performance requirements of streaming applications by micro-managing mapping, scheduling, and performance of each such application. Additionally, the proposed methodology in [37] is only restricted for streaming applications and is evaluated on simulation platform. Whereas, *RewardProfiler* works for any type of application as shown in Sec. IV and focuses on resource allocation based on the chosen reward constraints, hence, avoiding micro-management of scheduling problems, which would add more overhead to the current solution.

In [32], Schwarzer et al. propose a methodology to eliminate symmetry by clustering tasks and mapping them using Integer Linear Program (ILP) which eliminates all architectural as well as encoding symmetries from the search space. Moreover, efficacy of [32] is shown through simulation platform and one of the biggest issue of the proposed methodology is that as the number of task clusters grow, the time to solve the ILP also increases proportionately. Since, in *RewardProfiler* each application is treated as a clustered task and resource allocation and frequency scaling is used to maximize rewards as a whole for each application, symmetry elimination as well as the number of design point combination remains same for a particular hardware platform and hence computation time required for the application also remains same.

TABLE IV: Comparative study of AC-DSE [36], Rosvall et al. [37] and Schwarzer et al. [32] with *RewardProfiler*

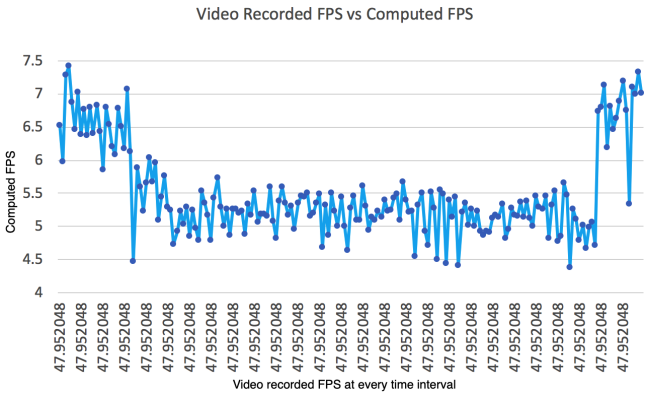
Method	Sim	Flex. S.	M. Obj.
AC-DSE [36]	✓	✓	✗
Rosvall et al. [37]	✓	✗	✗
Schwarzer et al. [32]	✓	✓	✗
RewardProfiler	✗	✓	✓

Table IV summarizes the differences between AC-DSE [36], Rosvall et al. [37], Schwarzer et al. [32] and *RewardProfiler* based on the following criteria:

- **Sim**: Whether the methodology is implemented on a simulator or not. If the implementation and experimentation is on a simulated platform then this feature has ✓in the table.
- **Flex. S.**: Whether the methodology could be implemented for any type of software/application/task set or not. If the implementation works for any type of software/application/task set then this feature has ✓in the table.
- **M. Obj.**: Whether the methodology could be implemented for multiple objectives/rewards such as reducing energy consumption, increasing performance/throughput, reducing thermal gradient, etc. If the methodology is capable of handling multiple objectives/rewards as mentioned above then this feature has ✓in the table.

V. Discussion and Future Scope

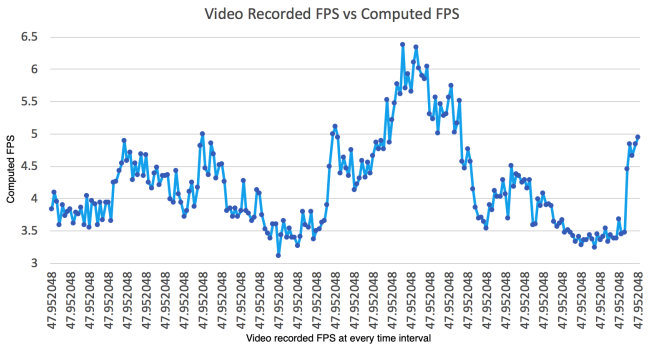
Fig. 6: Computed fps at 1700 MHz for big cores and LITTLE cores at 1100 MHz



From our experimental results we have also validated that the clustering technique of frequency levels was in fact effective due to the fact the differences between the reward gained at intermediate[13] frequency levels weren't

[13]Here intermediate frequencies are other frequency levels that lie between F_h, F_{mh}, F_{ml}&F_l

Fig. 7: Computed fps at 800 MHz for big cores and 400 MHz for LITTLE cores

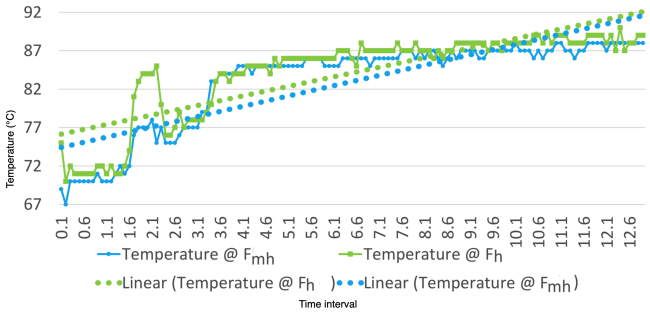


noticeable. Fig. 6 and Fig. 7 show the fps achieved in the face detection application for intermediate frequency levels respectively. From the figures we could see that the difference between the fps achieved by executing the application at F_{mh}[14] (see Fig. 4) is not very different from running the application at 1700 MHz for big cores and LITTLE cores at 1100 MHz. The same could be inferred if we compare the resultant fps achieved by executing the application at 800 MHz for big cores and 400 MHz for LITTLE cores with that of the result of the computed fps achieved by executing the application at F_{ml}.

Another point we need to keep in mind is that the reason to choose four different frequency levels instead, especially middle high and middle low frequency, for clustering reward based design space mapping points is to get four different values[15] of angle of slope for the linear function. The main motivation behind it is to measure the angular increase and decrease between these slopes in order to be able to decide automatically which frequency to select so that maximum reward could be achieved. The same ideology of measuring the decrease and increase in angular difference between the slopes could be used for mapping allocation and suggestion, even without profiling that CDP itself. For example, if we could notice that the values of ω $(....\omega_{i-1},\omega_i,\omega_{i+1}....)$ increase or decrease gradually then from the gradual variation of the values of ω we could develop a prediction model for resource allocation and operating frequency selection based on that knowledge. Since this approach is not within the scope of this research, we could focus on implementing such approach of developing prediction model in a future extension of this work.

We also have to keep in mind that it could be argued, we could just decide to suggest the best possible frequency and maximize reward just from the linear expression. But from the experimental data we could notice that the relationship between reward and frequency is not exactly linear for every chosen reward[16] and hence computing either of their value from the linear equation without considering the difference between the angle achieved from the linear functions into account would lead to high degree of errors. When we executed Streamcluster of PARSEC workbench at F_h and F_{mh} frequencies respectively, we chose temperature as the reward that we wanted to minimize. Here, we want to minimize the reward because least operating temperature results in more reliability of the device thus contributing to overall benefit of the system. The relationship between execution time and operating temperature of the big cores follows a quadratic equation instead of linear. From Fig. 8 we could notice that for different frequencies the operating temperature of big cores is not linear as mentioned earlier and the angle between the linear function lines are different as well.

Fig. 8: Computed temperature of big cores at F_h and F_{mh}



One shortcoming of this proposed approach (*RewardProfiler*) is that if the number of allowed frequency scaling levels on the cluster of CPU cores is less than five then the *RewardProfiler* does not really provide any reduction of CDP since we are selecting 4 frequency scaling levels (F_h, F_{mh}, F_{ml}&F_l). This is evident even in Fig. 2, where for a system with only 2 frequency scaling level there was no reduction in CDP. However, since most of the current MPSoC such as Exynos 5422 SoC [24] used in popular Samsung phones, HiSilicon Kirin 970 SoC [41] used in popular Huawei phones, etc. support more than 13 frequency scaling levels at the least and for such popular MPSoC our *RewardProfiler* would be very beneficial to reduce CDP and suggest appropriate resource mapping for maximized reward generation.

VI. Conclusion

In this paper we proposed a profiling strategy for IDEs to map available resources in a heterogeneous MPSoC device to applications in such a way that we could achieve maximum overall “reward" for the executing applications. In our experimental and validation section we were able to prove that our profiling methodology is not just capable of reducing the design points (mapping) of combined design space by 170x at max for Exynos 5422 MPSoC without loosing any important design points, but at the same time able to outperform the state-of-the-art DPTM approach and yet being able to achieve high reward for executing application as desired by the user.

Acknowledgment

This work has been supported by the UK Engineering and Physical Sciences Research Council EPSRC [EP/R02572X/1

[14]The face detection application was executed on the big cores at 1200 MHz and LITTLE cores at 900 MHz.

[15]One value for each frequency level out of four: F_h, F_{mh}, F_{ml}&F_l.

[16]Here, reward could be energy efficiency or performance or operating temperature of the cores, etc.

and EP/P017487/1], the Natural Science Foundation of Guangdong Province No. 2018A030313166, and Pearl River S&T Nova Program of Guangzhou No. 201806010038. Somdip would also like to thank his colleagues from the University of Essex, the Samsung R&D Institute UK, and his parents for their support.

REFERENCES

[1] D. Gove and J. Gove, "Generating and using profile information automatically in an integrated development environment," Jan. 8 2004, uS Patent App. 10/191,355.

[2] G. E. Moore, "Cramming more components onto integrated circuits," *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82–85, 1998.

[3] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel, "Mapping on multi/many-core systems: survey of current and emerging trends," in *Design automation conference (dac), 2013 50th acm/edac/ieee*. IEEE, 2013, pp. 1–10.

[4] P. Van Stralen and A. Pimentel, "Scenario-based design space exploration of mpsocs," in *Computer Design (ICCD), 2010 IEEE International Conference on*. IEEE, 2010, pp. 305–312.

[5] A. Weichslgartner, D. Gangadharan, S. Wildermann, M. Glaß, and J. Teich, "Daarm: Design-time application analysis and run-time mapping for predictable execution in many-core systems," in *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis*. ACM, 2014, p. 34.

[6] W. Quan and A. D. Pimentel, "A hybrid task mapping algorithm for heterogeneous mpsocs," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 14, no. 1, p. 14, 2015.

[7] T. D. Ngo, K. J. Martin, and J.-P. Diguet, "Move based algorithm for runtime mapping of dataflow actors on heterogeneous mpsocs," *Journal of Signal Processing Systems*, vol. 87, no. 1, pp. 63–80, 2017.

[8] A. D. Pimentel, "Exploring exploration: A tutorial introduction to embedded systems design space exploration," *IEEE Design & Test*, vol. 34, no. 1, pp. 77–90, 2017.

[9] T. S. Muthukaruppan, H. Javaid, T. Mitra, and S. Parameswaran, "Energy-aware synthesis of application specific mpsocs," in *Computer Design (ICCD), 2013 IEEE 31st International Conference on*. IEEE, 2013, pp. 62–69.

[10] U. Gupta, C. A. Patil, G. Bhat, P. Mishra, and U. Y. Ogras, "Dypo: Dynamic pareto-optimal configuration selection for heterogeneous mpsocs," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 5s, p. 123, 2017.

[11] Y. Shin, K. Choi, and T. Sakurai, "Power optimization of real-time embedded systems on variable speed processors," in *Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design*. IEEE Press, 2000, pp. 365–368.

[12] H. Lin and Y. Fei, "Exploring custom instruction synthesis for application-specific instruction set processors with multiple design objectives," in *Low-Power Electronics and Design (ISLPED), 2010 ACM/IEEE International Symposium on*. IEEE, 2010, pp. 141–146.

[13] A. Gordon-Ross, F. Vahid, and N. Dutt, "Automatic tuning of two-level caches to embedded applications," in *Proceedings of the conference on Design, automation and test in Europe-Volume 1*. IEEE Computer Society, 2004, p. 10208.

[14] A. Benoit and Y. Robert, "Mapping pipeline skeletons onto heterogeneous platforms," *Journal of Parallel and Distributed Computing*, vol. 68, no. 6, pp. 790–808, 2008.

[15] A. Bergel, F. Banados, R. Robbes, and D. Röthlisberger, "Spy: A flexible code profiling framework," *Computer Languages, Systems & Structures*, vol. 38, no. 1, pp. 16–28, 2012.

[16] M. Palmieri, I. Singh, and A. Cicchetti, "Comparison of cross-platform mobile development tools," in *Intelligence in Next Generation Networks (ICIN), 2012 16th International Conference on*. IEEE, 2012, pp. 179–186.

[17] 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.

[18] A. Abdi and H. R. Zarandi, "Hystery: a hybrid scheduling and mapping approach to optimize temperature, energy consumption and lifetime reliability of heterogeneous multiprocessor systems," *The Journal of Supercomputing*, pp. 1–26, 2018.

[19] I. Inayat, S. S. Salim, S. Marczak, M. Daneva, and S. Shamshirband, "A systematic literature review on agile requirements engineering practices and challenges," *Computers in human behavior*, vol. 51, pp. 915–929, 2015.

[20] E.-M. Schön, J. Thomaschewski, and M. J. Escalona, "Agile requirements engineering: A systematic literature review," *Computer Standards & Interfaces*, vol. 49, pp. 79–91, 2017.

[21] H. Kniberg, *Scrum and XP from the Trenches*. Lulu. com, 2015.

[22] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *Test Symposium (ETS), 2013 18th IEEE European*. IEEE, 2013, pp. 1–6.

[23] Q. Xu, T. Mytkowicz, and N. S. Kim, "Approximate computing: A survey," *IEEE Design & Test*, vol. 33, no. 1, pp. 8–22, 2016.

[24] "Exynos 5 octa (5422)," https://www.samsung.com/exynos, 2018, accessed: 2018-09-27.

[25] S. Dey, E. Z. Guajardo, B. K. Reddy, X. Wang, A. K. Singh, and K. McDonald-Maier, "Edgecoolingmode: An agent based thermal management mechanism for dvfs enabled heterogeneous mpsocs," in *2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems*. IEEE, 2019.

[26] S. Isuwa, S. Dey, A. K. Singh, and K. McDonald-Maier, "Teem: Online thermal- and energy-efficiency management on cpu-gpu mpsocs," in *2019 Design, Automation, and Test in Europe (DATE 2019)*. IEEE, 2019.

[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," in *2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. IEEE, 2018, pp. 291–298.

[28] 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)," in *Adaptive Many-Core Architectures and Systems workshop, York, UK*.

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

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

[31] 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.

[32] T. Schwarzer, A. Weichslgartner, M. Glaß, S. Wildermann, P. Brand, and J. Teich, "Symmetry-eliminating design space exploration for hybrid application mapping on many-core architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 2, pp. 297–310, 2018.

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

[34] G. Bradski and A. Kaehler, "Opencv," *Dr. Dobb's journal of software tools*, 2000.

[35] 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.

[36] A. Shahid, M. Y. Qadri, M. Fleury, H. Waris, A. Ahmad, and N. N. Qadri, "Ac-dse: Approximate computing for the design space exploration of reconfigurable mpsocs," *Journal of Circuits, Systems and Computers*, vol. 27, no. 09, p. 1850145, 2018.

[37] K. Rosvall and I. Sander, "Flexible and tradeoff-aware constraint-based design space exploration for streaming applications on heterogeneous platforms," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 23, no. 2, p. 21, 2018.

[38] T. Givargis and F. Vahid, "Platune: a tuning framework for system-on-a-chip platforms," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 11, pp. 1317–1327, 2002.

[39] G. Mariani, G. Palermo, V. Zaccaria, and C. Silvano, "Oscar: An optimization methodology exploiting spatial correlation in multicore design spaces," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 5, pp. 740–753, 2012.

[40] M. Lukasiewycz, M. Glaß, C. Haubelt, and J. Teich, "Efficient symbolic multi-objective design space exploration," in *Proceedings of the 2008 Asia and South Pacific Design Automation Conference*. IEEE Computer Society Press, 2008, pp. 691–696.

[41] "Kirin - hisilicon," www.hisilicon.com/en/Solutions/Kirin, 2018, accessed: 2018-09-27.