Est. 1841 | YORK ST JOHN UNIVERSITY

Dey, Somdip, Isuwa, Samuel, Saha, Suman, Singh, Amit Kumar and McDonald-Maier, Klaus (2022) CPU-GPU-Memory DVFS for Power-Efficient MPSoC in Mobile Cyber Physical Systems. Future Internet, 14 (3). p. 91.

Downloaded from: https://ray.yorksj.ac.uk/id/eprint/8673/

The version presented here may differ from the published version or version of record. If you intend to cite from the work you are advised to consult the publisher's version: http://dx.doi.org/10.3390/fi14030091

Research at York St John (RaY) is an institutional repository. It supports the principles of open access by making the research outputs of the University available in digital form. Copyright of the items stored in RaY reside with the authors and/or other copyright owners. Users may access full text items free of charge, and may download a copy for private study or non-commercial research. For further reuse terms, see licence terms governing individual outputs. Institutional Repositories Policy Statement

RaY

Research at the University of York St John

For more information please contact RaY at ray@yorksj.ac.uk

future internet

MDPI

Article

CPU-GPU-Memory DVFS for Power-Efficient MPSoC in Mobile Cyber Physical Systems

Somdip Dey [1,2,*], Samuel Isuwa [3], Suman Saha [2], Amit Kumar Singh [1] and Klaus McDonald-Maier [1]

[1] School of Computer Science and Electronic Engineering (CSEE), University of Essex, Colchester CO4 3SQ, UK; a.k.singh@essex.ac.uk (A.K.S.); kdm@essex.ac.uk (K.M.-M.)
[2] Nosh Technologies, Colchester CO4 3SL, UK; suman.saha@nosh.tech
[3] School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK; s.isuwa@soton.ac.uk
* Correspondence: somdip.dey@essex.ac.uk or dey@nosh.tech

Abstract: Most modern mobile cyber-physical systems such as smartphones come equipped with multi-processor systems-on-chip (MPSoCs) with variant computing capacity both to cater to performance requirements and reduce power consumption when executing an application. In this paper, we propose a novel approach to dynamic voltage and frequency scaling (DVFS) on CPU, GPU and RAM in a mobile MPSoC, which caters to the performance requirements of the executing application while consuming low power. We evaluate our methodology on a real hardware platform, Odroid XU4, and the experimental results prove the approach to be 26% more power-efficient and 21% more thermal-efficient compared to the state-of-the-art system.

Keywords: CPU; GPU; RAM; memory; DVFS; power consumption; multi-processor system-on-chip; MPSoC; peak temperature; thermal behaviour

check for updates

Citation: Dey, S.; Isuwa, S.; Saha, S.; Singh, A.K.; McDonald-Maier, K. CPU-GPU-Memory DVFS for Power-Efficient MPSoC in Mobile Cyber Physical Systems. *Future Internet* **2022**, *14*, 91. https://doi.org/10.3390/fi14030091

Academic Editor: Athanasios Panagopoulos

Received: 5 February 2022
Accepted: 12 March 2022
Published: 14 March 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

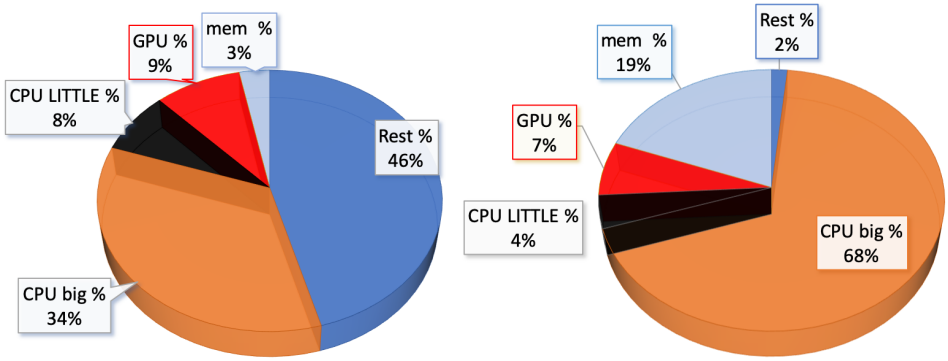
Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction and Motivation

Mobile cyber-physical systems such as smartphones (mobile phones) have become an integral part of our daily life and we use them for a range of applications: browsing the internet, playing games, capturing and editing videos, staying connected with friends and family over social media, etc. To improve the versatility of mobile phones to be able to cater to any type of application being executed on such a device, mobile phones come equipped with heterogeneous multi-processor systems-on-chip (MPSoCs), which consist of different types of processing elements (PEs) such as CPU (big and LITTLE varieties, with big CPUs traditionally having a powerful computational capacity and LITTLE CPUs being comparatively more power-efficient with a lower computational capacity [1]) and GPUs with different processing capabilities. These heterogeneous multi-processor systems have proven to provide more benefits in terms of area and core-to-application matching for improved performance, power and workload coverage [2,3]. On the other hand, given the fact that these mobile devices are battery-operated and that users expect such devices to be operable without the need for frequent charging, optimised power consumption on such devices is an important concern [4,5]. Furthermore, the PEs in these MPSoCs support dynamic voltage and frequency scaling (DVFS), which can be used to reduce dynamic power consumption ($P \propto V^2 f$, where P represents dynamic power consumption, V represents the voltage of the CMOS and f represents the operating frequency) [5–7]. This helps to reduce the power consumption by executing the workload over extra time at a lower voltage and frequency.

In most modern MPSoCs, CPU, GPU and RAM support DVFS, with each of these components affecting the total power consumption of the device differently for different types of applications. For example, when we observed the power consumption due to the effects of DVFS in CPU, GPU and RAM (denoted as memory only) in an Odroid XU4 [8],

utilizing Exynos 5422 MPSoC [9], when it was idle (when no other application was being executed other than the background processes of the OS), we noticed that big CPUs, LITTLE CPUs, GPU and memory consumed 34%, 8%, 9% and 3% of the total power consumption, respectively, of the device on average (as shown in Figure 1a). The total power consumption of the MPSoC when idle was 3.534 W. In this case, the Exynos 5422 MPSoC utilised ARM's big.LITTLE processor technology [10], in which two different types of CPUs (big and LITTLE) are utilised to cater for the performance and power consumption requirements of executing applications. In order to observe the effect of DVFS on each of the major components (big CPUs, LITTLE CPUs, GPU and memory) of the MPSoC we recorded the power consumption during the operation of each of these components in their maximum operating frequency and then in their minimum operating frequency, consecutively, to measure the percentage of total power consumption that is attributed to the maximum and minimum frequency. Figure 1b illustrates the percentage of total power consumption of big CPUs, LITTLE CPUs, GPU and memory in the Exynos 5422 MPSoC when executing the Streamcluster benchmark (in native mode) from the PARSEC benchmark suite [11]. We chose the Streamcluster benchmark because it reflects a mixed workload (both compute-intensive and memory-intensive) [12] to mimic the workload of most of the popular applications used by users. The maximum power consumption of the MPSoC when executing Streamcluster was 10.11 W. As shown in Figure 1, one interesting observation was that in a mixed workload application the memory can contribute to 19% of the total power consumption, which is still a significant amount, and hence, DVFS in memory plays an important role in regard to the total power consumption of the device.



(**a**) Idle (**b**) Executing streamcluster

Figure 1. Percentage of average power consumed by the big CPUs, LITTLE CPUs, GPUs, memory and the rest of the components such as the fan (active cooling), hardware storage and on-chip communication network of the Exynos 5422 MPSoC when idle vs executing the streamcluster benchmark.

There has been a series of published studies on the effects of performing DVFS on CPU or GPU or memory separately or using a combination of two of these components [4,5,13–15]; however, to the best of our knowledge there have not been any studies on the effects of performing DVFS on CPU, GPU and memory together in order to optimise the performance and power consumption of the execution of applications in mobile MPSoCs. Moreover, it is quite attractive to employ methods such as reinforcement learning (RL) to perform CPU/GPU/Memory DVFS since such methods could be application-agnostic. However, for dynamic applications in which the CPU, GPU and memory usage vary dynamically, if RL methods are not allowed to explore the system long enough then the achieved power consumption could be sub-optimal [16]. We utilised the RL method (denoted as *RL* (*CPU-GPU*)) in [4] to perform CPU-GPU DVFS and extended the method to perform CPU-GPU-memory DVFS (denoted as *RL* (*CPU-GPU-RAM*)) to compare the power consumption with our proposed method, denoted as *CGM-DVFS*, to perform CPU-GPU-memory DVFS. Figure 2 shows the average power consumption in Watts on the Exynos 5422 MPSoC when executing different benchmark applications on different approaches.

The benchmark applications were object detection using YOLO (*yolo*) [17], the blackscholes benchmarkfrom PARSEC [11] and fft from Splash-2 [18]. From Figure 2, it is evident that application-agnostic approaches to performing DVFS on CPU-GPU-memory might not lead to close-to-optimal power consumption and therefore indicates that DVFS in the CPU, GPU and memory in mobile MPSoCs more challenging.

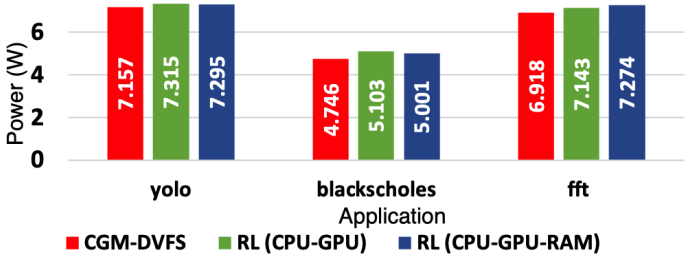


Figure 2. Average power consumption (Watts) when executing different benchmark applications using different approaches.

In this paper, we study the effect of DVFS on memory towards the total power consumption in a mobile MPSoC for different types of applications and we also propose a novel approach, called *CGM-DVFS* (**C**PU-**G**PU-**M**emory **DVFS**), to perform DVFS on CPU (big and LITTLE), GPU and memory in mobile MPSoCs to cater for the performance requirements of the execution of applications while consuming the least amount of power. To this extent, the concrete contributions of this paper are as follows.

1. Studying the effect of DVFS on memory in regard to the total power consumption and performance of executing applications in a mobile MPSoC.
2. Proposing a novel approach—CGM-DVFS—to perform DVFS on CPU-GPU-memory in a mobile MPSoC to cater for the performance requirements of executing applications, while consuming the least amount power.
3. An experimental evaluation of CGM-DVFS on a real hardware platform, Odroid XU4, and a comparative study between CGM-DVFS and state-of-the-art approaches to optimise power consumption.
4. A comparative study and analysis between CGM-DVFS and state-of-the-art delayed reinforcement learning approaches to show that CGM-DVFS is better suited to achieving close-to-optimal power consumption.

The rest of the paper is organised as follows. In Section 2, we show the effect of DVFS on memory in terms of power consumption as a motivational case study. In Section 3, we mention the related works, whereas in Section 4 we provide details of the hardware and software infrastructure used in this study, along with the problem formulation, on the basis of which our proposed method was designed. In Section 5, we provide details on our proposed methodology–*CGM-DVFS*, whereas in Section 6 we show the efficacy of our proposed method through an experimental evaluation, along with a comparative study with the state-of-the-art approach. Finally, we conclude the paper in Section 7.

2. Effect of DVFS on Memory

To observe the effect of DVFS on memory in regard to the total power consumption and performance of different types of executing applications in a mobile MPSoC, we chose benchmark applications from PARSEC [11], Whetstone [19,20] and Splash-2 [18] benchmark suites, as well as RSA encryption [21] and streaming Youtube videos in the Chromium browser. Given the fact that streaming video on Youtube is one of the most popular applications/workloads on mobile devices [22], we chose this workload along with the other benchmark applications. Due to the popularity of RSA encryption for key exchange [23] in most of the secured applications, we chose to perform RSA for 512, 1024, 2048 and 4096 bit encryption and observed the effect of DVFS on memory. Based on the parallelisation, the size of the working set and the data usage of the different

types of benchmark applications from PARSEC and Splash-2, the applications (workload) were segregated into three types [12]: compute-intensive (denoted as Compute), memory-intensive (denoted as memory) and mixed-workload (denoted as Mixed), in which the workload is both compute- and memory-intensive. Table 1 shows the abbreviations of the different types of benchmark applications for our study of the effect of DVFS on memory. Note: Given the compute- and memory-intensive nature of RSA encryption and Youtube video streaming based on [12], both of these applications were also considered to be part of the mixed-workload category.

Table 1. Abbreviations of different types of benchmark applications.

Benchmark Applications		
Type	**Name (Execution Option)**	**Abbreviation**
Compute	Whetstone	wht.
Compute	blackscholes (native)	blks.
Memory	x264 (simlarge)	x264
Memory	dedup (simlarge)	ded.
Memory	canneal (simlarge)	cann.
Mixed	FFT (simlarge)	fft
Mixed	facesim (simlarge)	fsim.
Mixed	streamcluster (native)	strm.
Mixed	Youtube in Chromium browser	ytub.
Mixed	RSA	rsa

In Odroid XU4, there are nine available operating frequencies for memory and we chose the highest (825 MHz), the middle (413 MHz) and the lowest (138 MHz) operating frequency levels to observe the effect of DVFS on the power consumption and performance (execution time) of the executed benchmark applications mentioned in Table 1. We executed the benchmark applications five times on the aforementioned three operating frequencies of the memory and observed the average power consumption and performance (execution time), which are shown in Table 2. We also observed the power consumption for the aforementioned three operating frequencies of the memory while the system was idle (only executing the background processes of the OS), which is also denoted as idle, running with a Linux performance governor.This serves as a baseline to evaluate the effect of DVFS on memory in an idle Odroid XU4 system running with a performance governor. In Table 2 we can note that using DVFS in the memory can improve the power savings by 25.124% based on the type of application being executed and hence this calls for an approach that is capable of performing DVFS on CPU, GPU and memory to cater for the performance requirements of the applications, while consuming the least power.

Table 2. Power consumption (Pow. max) of different benchmark applications (App) when executing the application on the maximum operating frequency of the memory. *Pow. save middle* (%) and *Pow. save min* (%) are the improvements in power savings when executing the application on the middle operating frequency and minimum operating frequency, respectively. *Perf. middle* (%) and *Perf. min* (%) indicate the loss in performance for executing the application on the middle operating frequency and minimum operating frequency, respectively.

App	Pow. Max (W)	Pow. Save. Middle (%)	Pow. Save. Min (%)	Perf. Middle (%)	Perf. Min (%)
idle	3.313	5.192	5.886	-	-
wht.	3.556	4.415	5.202	−2.121	−3.638
blks.	5.474	5.298	9.81	−2.483	−7.823
x264	8.748	13.649	20.085	−6.486	−16.993
ded.	7.893	11.136	18.282	−7.674	−14.598
cann.	7.919	10.317	16.782	−7.847	−12.773
fft	7.41	4.575	14.008	−2.939	−15.834
fsim.	5.378	4.574	9.967	−3.475	−7.558
strm.	10.11	1.82	**25.124**	−2.116	**−16.883**
ytub.	7.014	1.725	7.214	-	-
rsa	6.119	1.994	4.935	−1.032	−1.894

3. Related Works

Power-saving mechanisms within performance constraints utilizing DVFS capabilities on heterogeneous MPSoCs have been considered in many studies [4–6,13,14,24–36]. Given the fact that power consumption in a heterogeneous MPSoC can be significantly affected by big CPUs, LITTLE CPUs, GPUs and memory, most of the published studies have proposed power-saving approaches utilizing DVFS of different aforementioned components of the MPSoC but have not considered performing DVFS on all these components to achieve more reduced power consumption while catering to performance constraints.

In [5,6,24,25,28], different approaches to performing DVFS on CPUs to contribute to the reduction of power consumption were proposed. On the other hand, many studies [4,13,14,25,26,30] have considered utilizing DVFS in CPU and GPU to achieve power efficiency in MPSoCs. In [28], David et al. proposed an on-line power management algorithm based on DVFS in a single-chip cloud computer (SCC) platform with multiple cores, in which voltage and frequency could be scaled for each individual tiles. In [29], Bogdan et al. examined a DVFS-based power optimisation mechanism in which a controller for fractal workloads with precise constraints on state and control variables and specific time bounds was utilised. In [6], Reddy et al. performed thread-to-core mapping and DVFS on the cores in relation to workloads that were classified based on a metric, memory reads per instruction (MRPI), and in our study we denoted this methodology as MRPI.In [7], Dey et al. performed DVFS on cores based on the desired reward, which was chosen to be reduced power consumption on the device in our case, and we denoted this methodology as RewardProfiler. In [4], Dey et al. proposed *Next*, which performs DVFS on CPU and GPU based on the user's interaction with the device using Q-Learning (reinforcement learning). In [25], Mandal et al. proposed an imitation-learning-based framework for dynamically controlling the big and LITTLE CPUs, CPU number and the frequencies of active cores in heterogeneous mobile processors. Additionally, there have been extensive studies [31–33], in which DVFS was performed on memory to improve power efficiency either in general-purpose computers or server systems. Only a handful of studies [34–36] have performed DVFS on CPU and memory together to benefit from combined power efficiency in a mobile platform. However, none of these studies attempted to combine the benefits of performing DVFS on CPUs, GPUs and memory in conjunction in a mobile MPSoC to improve power efficiency while catering for performance constraints and hence, this paper addresses this gap in the literature.

4. System Model and Problem Formulation

In this section, we provide details on the hardware and software infrastructure used in this study, along with the problem formulation on the basis of which our proposed method was designed.

4.1. Hardware and Software Infrastructure

We chose the Odroid XU4 [8] development board to implement our CPU-GPU-Memory DVFS. Odroid XU4 employs the Samsung Exynos 5422 [9] MPSoC, which is popularly used in Samsung mobile devices, especially the Samsung Galaxy S5. The Odroid XU4 is a representational development board of the Galaxy S5 smart-phone. The Exynos 5422 MPSoC contains clusters of big (4 Cortex A-15) and LITTLE CPU cores (4 Cortex A-7). This MPSoC provides DVFS features per cluster, with the big CPU cluster having nineteen frequency scaling levels, ranging from 200 MHz to 2000 MHz with steps of 100 MHz, and the LITTLE CPU cluster having thirteen frequency scaling levels ranging from 200 MHz to 1400 MHz with steps of 100 MHz. Exynos 5422 comes equipped with a GPU cluster, called Mali-T628 MP6 GPU, consisting of six shader cores and has seven frequency-scaling levels as follows: 600, 543, 480, 420, 350, 266 and 177 MHz, respectively. This MPSoC supports 2 GB RAM, which has the following nine frequency scaling levels: 825, 728, 633, 543, 413, 275, 206, 165 and 138 MHz, respectively. DVFS in big and LITTLE CPUs in Exynos 5422 is performed cluster-wise and the voltage value for a particular frequency is fixed for that frequency. It should also be noted that below some frequencies, voltage remains the same, but above a certain point, the voltage increases linearly [37]. Examples of this include that for A7 (LITTLE) CPUs frequencies of 200–500 MHz have the voltage of 0.913 V, whereas for A15 (big) CPUs, frequencies of 200–700 MHz have a voltage of 0.9125 V.

The Exynos 5422 MPSoC also has five temperature sensors, four of which are located on four big CPUs and one on the GPU. The Odroid XU4 board does not have an internal power sensor on-board; hence, Odroid SmartPower2 [38], which is an external power monitor with networking capabilities over WIFI, was used in this study to take power consumption readings.

The Odroid XU4 was running on UbuntuMate version 14.04 (Linux Odroid Kernel: 3.10.105) and executing the performance governor. During the time of implementing and conducting our experiments the average ambient temperature of the room was 21 °C.

4.2. Problem Formulation

In this subsection, we define the problem formulation on which our proposed method was based.

Given: Let us consider a system that has a set of applications, $S_{App} = \{App_1, App_2, \ldots App_i\}$, where App_i is the ith application and App_i consists of a set of tasks, $S_{task} = \{tsk_1, tsk_2, \ldots tsk_i\}$, where S_{task} always generates a fixed performance output Prf_i for the fixed DVFS configuration values R_i while executing App_i on the system. Here, R_i consists of the combination of the DVFS values for big CPUs ($DVFS_{b_i}$), LITTLE CPUs ($DVFS_{L_i}$), GPUs ($DVFS_{g_i}$) and memory ($DVFS_{m_i}$) such that $R_i =< DVFS_{b_i}, DVFS_{L_i}, DVFS_{g_i}, DVFS_{m_i} >$ leads to a fixed performance output Prf_i. Now, we can consider $Prf_{desired}$ as the desired value of the performance output for the execution of App_i.

Find: The desired DVFS configuration values ($R_{desired}$) are the combination of the desired DVFS values for big CPUs ($DVFS_{b_{desired}}$), LITTLE CPUs ($DVFS_{L_{desired}}$), GPUs ($DVFS_{g_{desired}}$) and memory ($DVFS_{m_{desired}}$).

Subject to: Meeting the desired performance $Prf_{desired}$ while consuming the least power (P_{least}) during the execution of App_i on $R_{desired}$.

5. Proposed Methodology: CGM-DVFS

5.1. Overview of CGM-DVFS

Figure 3 presents a block diagram of our proposed CGM-DVFS methodology. CGM-DVFS is not just an approach, but also an automated agent that sets the appropriate

DVFS on CPU, GPU and memory to achieve the desired performance of the executing application while consuming the least amount of power. For each App_i in S_{App}, the profiling of App_i (this step is denoted as *Profiling*) is performed such that for different combinations of $DVFS_{b_i}$, $DVFS_{L_i}$, $DVFS_{g_i}$ and $DVFS_{m_i}$, the corresponding value of Prf_i, the corresponding peak temperature instance (T_i) and the corresponding power consumption (P_i) of the device are recorded and stored on the disk storage memory. More on Profiling is provided in Section 5.2. From the set containing the profiled values of $S_{Prf} = \{Prf_1, Prf_2, \ldots Prf_i\}$, the desired performance $Prf_{desired}$ is searched based on the equation: $Prf_{desired} \in S_{Prf}$; where $Prf_i \geq Prf_{desired}$. Now, for all the possible values of Prf_i that are equal or greater than $Prf_{desired}$ from S_{Prf}, the agent searches for the value with the least power consumption such that $P_{least} = min(S_P)$; where $S_P = \{P_1, P_2, \ldots P_i\}$ ($P_1, P_2, \ldots P_i$ are the corresponding power consumption of $Prf_1, Prf_2, \ldots Prf_i$). The agent then fetches the associated $DVFS_{b_i}$, $DVFS_{L_i}$, $DVFS_{g_i}$ and $DVFS_{m_i}$ configuration (this step is denoted as *Fetch desired config*), and then the desired DVFS values of big CPUs ($DVFS_{b_{desired}}$), LITTLE CPUs ($DVFS_{L_{desired}}$), GPUs ($DVFS_{g_{desired}}$) and memory ($DVFS_{m_{desired}}$) are set to this configuration (this step is denoted as *Set desired DVFS*).

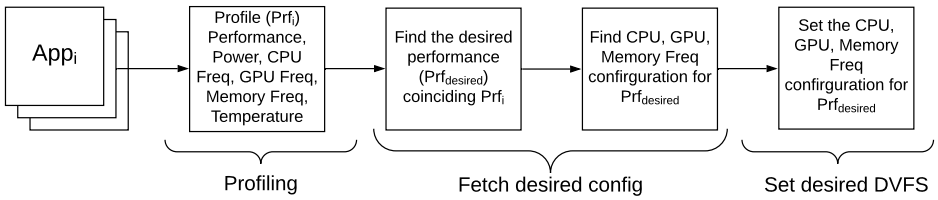


Figure 3. Block diagram illustrating the CGM-DVFS methodology.

Note: During *Profiling*, if the executing application (App_i) is not executed for a long enough time to observe all the DVFS combinations of CPU, GPU and memory to profile the performance and power consumption, then the profiled values will be a subset of S_{Prf}, denoted by S'_{Prf}, where $S'_{Prf} = \{Prf_1, Prf_2, \ldots Prf_j\}$ and $S'_{Prf} \subset S_{Prf}$. In such a case, the agent searches $Prf_{desired}$ from S'_{Prf} based on the value with the least power consumption.

5.2. Steps in Detail: Profiling, Fetch Desired Config and Set Desired DVFS

In the profiling step, we utilise the concept of clustering performance for a range of DVFS, as introduced in [7], in which Dey et al. proposed that for a group of DVFS values for the same processing element the performance outcome remains similar. For example, for App_i a set of consecutive DVFS values could lead to more or less the same performance output Prf_i and hence, instead of selecting each of these DVFS values during the design space exploration (Profiling), only one representative DVFS value from the set is selected and then profiled only for that value. In this way, the agent can reduce the number of configurations that it has to profile. For our experimental device, Odroid XU4, we chose the following DVFS configurations: four DVFS levels for big CPUs (2 GHz, 1.4 GHz, 0.8 GHz, 0.2 GHz); four DVFS levels for LITTLE CPUs (1.4 GHz, 1 GHz, 0.8 GHz, 0.2 GHz); three DVFS levels for GPUs (600 MHz, 420 MHz, 177 MHz) and three DVFS levels for memory (825 MHz, 413 MHz, 138 MHz). In [7], the equation for the combined design point (CDP) is provided for an MPSoC where DVFS capability is only considered in big CPUs, LITTLE CPUs and GPUs. Since, in this paper, we also consider DVFS in the memory, the equation for CDP is modified to incorporate the operating frequency levels of memory as well and is represented in Equation (1). In Equation (1), n_b and n_L represent the number of big CPUs and LITTLE CPUs respectively, whereas, f_b, f_L, f_{GPU}, f_{mem} represent the number of operating frequency levels for big CPUs, LITTLE CPUs, GPUs and memory, respectively.

$$CDP = \{((n_b \times f_b) + (n_L \times f_L)) + (n_b \times f_b \times n_L \times f_L)\} \times f_{GPU} \times f_{mem} \quad (1)$$

Since, in our chosen platform and methodology, DVFS in big and LITTLE CPUs is performed cluster-wise, the total number of reduced CDPs for the aforementioned configuration, as per Equation (1), is 216 ($\{((1 \times 4) + (1 \times 4)) + (1 \times 4 \times 1 \times 4)\} \times 3 \times 3$). The agent starts the profiling process by selecting the maximum DVFS level for big CPUs, LITTLE CPUs, GPUs and memory; records the performance output, temperature and power consumption for that configuration and then selects the next-lowest DVFS level in the configuration to record the same. The agent uses a waterfall method in which the DVFS levels are selected from high to low on big CPUs first, then on the LITTLE CPUs, then on the GPUs and then on the memory. Based on our empirical data, we noticed that to profile accurately it is best to profile each of the reduced CDPs every 100 milliseconds for 1 s and hence the total number of profiling points become 2160 (216×10). Once all the 2160 profiling points are traversed and configurations are recorded and stored on the disk memory, these configurations will be used (as in the "Fetch desired config" and "Set desired DVFS" steps) by the agent to find $Prf_{desired}$, in which the system consumes the least amount of power (P_{least}) and set the DVFS values accordingly.

5.3. Justification of the Design Choices

In the majority of commercial smartphones utilizing MPSoCs, due to constraints on the display size, most consumers utilise one application at any time period [39]. Henceforth, we have considered profiling one application at a time to make the proposed method more commercially applicable. Moreover, later in Section 6.3 we also show that application-agnostic approaches such as delayed reinforcement learning could lead to sub-optimal or worse power consumption than application-specific profiling approaches such as CGM-DVFS. Additionally, since different DVFS configurations for dynamic applications (tasks) could lead to dynamic profiling outputs such as performance and power consumption, we invoke CGM-DVFS at random time periods to update the profiling configurations and save them on the memory to perform the *Fetch desired config* & *Set desired DVFS* steps.

6. Experimental Results

6.1. Experimental Applications

To evaluate the efficacy of CGM-DVFS, we modified some of the existing popular applications, thus mimicking a mixed workload as utilised by users, such that the agent is capable of recording the performance output during the profiling step. The following applications were chosen for the experimental evaluation:

Face detection: Face detection using a Haar-cascade [40] is utilised, in which faces are detected based on the presence of Haar features in the video image frame. This application is denoted as *face*.

YOLO object detection: Object detection using the You Only Look Once (YOLO) approach [17] is utilised, in which objects are detected based on different regions in the video image frame. This application is denoted as *yolo*.

Video rendering: A video rendering program is utilised, in which each video image frame is converted to a greyscale image and then the text, "Hello, World¡' is rendered on top of the video image frame to be shown as the output. This application is denoted as *render*.

On-device streaming: A video streaming application is utilised, in which the video is streamed from the on-device storage. This application is denoted as *stream*.

Traffic sign detection: An application to detect traffic signs using a Haar cascade [41] is utilised, in which Haar features for traffic signs are being detected. This application is denoted as *traffic*.

MobileNet object classification: An application to classify dogs and cats in video image frames using the MobileNet CNN model [42] is utilised. This application is denoted as *classify*.

For the aforementioned applications (face, yolo, render, stream, traffic and classify), since all of them are computer-vision-based, we chose frames per second (FPS) to be the performance output and therefore the CGM-DVFS agent recorded the FPS as Prf_i, as

mentioned in the profiling step. In our experiments, we chose the desired FPS ($Prf_{desried}$) to be 60.

Additional benchmark applications: Since benchmark applications from PARSEC and SPLASH-2 benchmark suites do not allow one to observe the intermediate performance (execution time) of the application when executing it without the use of a performance counter, we executed blackscholes (denoted as blks.) from PARSEC, streamcluster (denoted as strm.) from PARSEC and fft from SPLASH-2 216 times (as per reduced CDP) such that each execution was performed on each configuration from the reduced CDP. The minimum execution time out of 216 executions of the respective benchmark application (228.18 seconds for blks., 368.15 seconds for strm.& 12.58 seconds for fft) was chosen as the $Prf_{desired}$ for that application. We chose to perform this experimentation method to prove the scalability and efficacy of CGM-DVFS across different types of applications and not just for computer-vision-based applications.
Note: Since we chose the minimum (best) execution time for the additional benchmark applications and given the fact that the media-based benchmark applications such as face, yolo, render, stream, traffic and classify do not have a specific execution time since they are continuously executed, the power consumption here is proportional to the energy consumption (energy = power $\times$ execution time) for executing the respective applications, since the execution time is constant in this case.

6.2. Evaluation and Comparative Study

We evaluated CGM-DVFS for each aforementioned experimental application fifteen times and observed the average power consumption of the MPSoC and the average peak temperature of the big CPUs. We chose to observe the peak temperature of the big CPUs since they tend to be the hottest hot spot in the MPSoC [43]. We also evaluated the average power consumption of the MPSoC and the average peak temperature of big CPUs achieved using the performance governor (denoted as performance), the interactive governor (denoted as interactive) and the state-of-the-art approaches as proposed in [4,6,7] (mentioned in Section 3). In [4], the proposed Q-Learning (reinforcement learning)-based DVFS is based on a reward function, as shown in Equation (2), which is based on Equation (3). We also denote this methodology as Next in our comparative study. In Equation (2), the reward function attempts to maximise the value of $PPDW$, which is a metric, *performance per degree watt*, incorporating the performance (FPS_i), temperature (ΔT, where ΔT is the difference between the current temperature, T_i, and the ambient temperature, T_a) and power consumption (P_i) of the device. The agent in Next has the following states: big_CPU_{freq}, $LITTLE_CPU_{freq}$, GPU_{freq}, $FPS_{current}$, $Target_FPS$, $Power_{current}$, $Temperature_{big}$ and $Temperature_{device}$; where big_CPU_{freq} is the frequency of the big CPU, $LITTLE_CPU_{freq}$ is the frequency of the LITTLE CPU, GPU_{freq} is the frequency of the GPU, $FPS_{current}$ is the current performance in terms of FPS, $Target_FPS$ is the desired performance in terms of FPS, $Power_{current}$ is the current power consumption and $Temperature_{big}$ and $Temperature_{device}$ are the temperature of the big CPU and the whole device, respectively. The actions performed by the Next agent are as follows: big frequency up, big frequency down, do not change big frequency, LITTLE frequency up, LITTLE frequency down, do not change LITTLE frequency, GPU frequency up, GPU frequency down and do not change GPU frequency. We modified Equation (3) to incorporate the performance of all types of applications, not only FPS-based ones, and the modified equation for $PPDW$ is Equation (4). Moreover, we also extended [4], denoted as Next_Mod, to incorporate memory DVFS along with CPU and GPU such that we could undertake a comparative study between Next and CGM-DVFS. In Next_Mod, the agent has a new state, RAM_{freq}, frequency of memory, and three more new actions: RAM frequency up, RAM frequency down and do not change RAM frequency. Both Next and Next_Mod were invoked every 100 ms. Exploration sessions for face, yolo, render, stream, traffic and classify applications for Next and Next_Mod were 5 min, whereas blks., strm. and fft were executed for their execution lifespan for Next and Next_Mod to explore. [4,6,7]

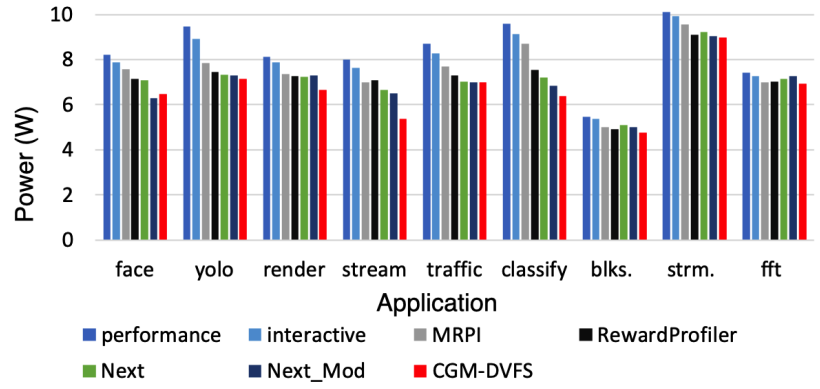
and Next_Mod were chosen for the comparative study because these methods perform DVFS on a combination of CPU, GPU and memory or all of the above.

$$max\ R(s_i, a_i) = max(PPDW_i), \quad \text{where} \quad max(PPDW_i) = PPDW_{best} \geq PPDW_i > PPDW_{worst} \tag{2}$$

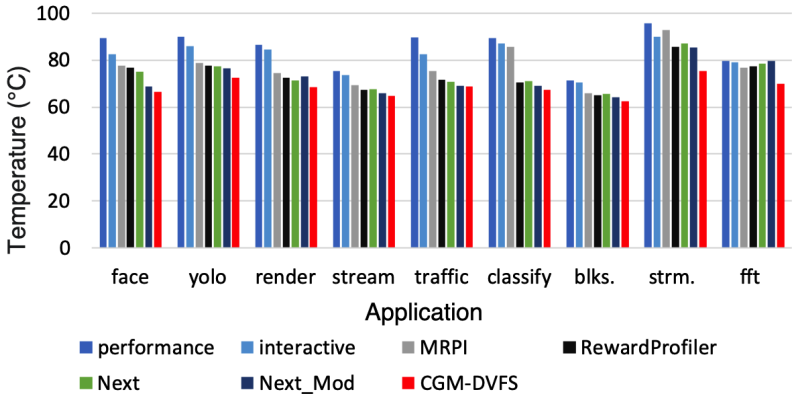
$$PPDW_i = \frac{FPS_i}{\Delta T \times P_i}, \text{where } \Delta T = T_i - T_a \tag{3}$$

$$PPDW_i = \frac{Perf_i}{\Delta T \times P_i}, \text{where } \Delta T = T_i - T_a \tag{4}$$

Figure 4 shows the average power consumption of the device (see Figure 4a) and the average peak temperature of big CPUs (see Figure 4b) while executing the aforementioned benchmark applications using different DVFS methodologies: performance, interactive, MRPI, RewardProfiler, Next, Next_Mod and CGM-DVFS. Tables 3 and 4 show the improvement in power-savings (%) and the reduction in peak temperature (%), respectively, of CGM-DVFS compared to performance, MRPI, RewardProfiler, interactive, Next & Next_Mod. Based on the tables, CGM-DVFS is capable of saving 33.476% more power compared to the performance governor, whereas it is capable of saving 26.796% more power compared to the state-of-the-art approach, MRPI. On the other hand, CGM-DVFS is also capable of reducing the peak temperature of big CPUs by 25.567% compared to the performance governor and by 21.238% compared to MRPI.



(**a**) Average power consumption in Watts



(**b**) Average peak temperature in °C

Figure 4. Average power consumption (Watts) and average peak temperature (°C) when executing different applications on different methodologies: performance, interactive, MRPI, RewardProfiler, Next, Next_Mod and CGM-DVFS.

Table 3. Improvement in power-savings (%) of CGM-DVFS compared to performance (perf.), MRPI, RewardProfiler (RProfiler.), interactive (inter.), Next and Next_Mod (N_Mod.).

App	Perf.	MRPI	RProfiler.	Inter.	Next	N_Mod.
face	21.08	14.59	9.28	17.81	8.61	−3.17
yolo	24.46	8.79	4.049	19.82	2.16	1.89
render	18.00	9.40	8.39	15.34	8.12	8.89
stream	32.81	23.17	24.34	29.55	19.40	17.58
traffic	19.74	9.17	4.02	15.57	0.48	−0.03
classify	**33.48**	**26.80**	15.36	30.2	11.42	6.93
blks.	12.43	5.13	3.83	11.34	6.10	5.10
strm.	21.20	18.80	11.97	15.99	2.62	0.60
fft	12.21	9.04	9.42	11.40	3.15	4.89

Table 4. Reduction in peak temperature of big CPUs (%) of CGM-DVFS compared to performance (perf.), MRPI, RewardProfiler (RProfiler.), interactive (inter.), Next and Next_Mod (N_Mod.).

App	Perf.	MRPI	RProfiler.	Inter.	Next	N_Mod.
face	25.57	14.16	13.18	19.38	11.29	3.04
yolo	19.43	8.13	6.74	15.67	6.44	5.19
render	20.80	8.06	5.38	18.80	3.90	6.27
stream	13.93	6.428	3.83	12.16	3.99	1.82
traffic	23.25	8.71	3.96	16.58	2.63	0.38
classify	**24.50**	**21.24**	4.30	22.42	4.92	2.29
blks.	12.43	5.13	3.83	11.34	4.46	2.49
strm.	21.20	18.80	11.97	15.99	13.39	11.67
fft	12.21	9.04	9.42	11.40	10.95	12.08

Overhead analysis: We also evaluated the overhead analysis of executing our proposed method. In our empirical data, we noted that the average overhead to read the profiled data (2160 profiling points) in the *Fetch desired config* step was 29.507 milliseconds and the overhead to search for the desired DVFS configuration in this same step was 0.145 milliseconds.

6.3. Comparative Study between CGM-DVFS and Delayed-Reinforcement-Learning Approaches

In this subsection, we provide a comparative study of our proposed method with the current state of the art.

In Figure 4, Tables 3 and 4 it can be noted that CGM-DVFS outperforms the Q-Learning based reinforcement learning (RL) approach, Next, in which DVFS is only performed on the CPU and GPU. This was expected since CGM-DVFS performs DVFS on the CPU, GPU and RAM to reduce the power consumption even more. However, when compared to Next_Mod, in which DVFS is performed on the CPU, GPU & RAM using Q-Learning, CGM-DVFS outperformed this method for the yolo, render, stream, classify, blks., strm. and fft applications. Interestingly, Next_Mod seemed to produce sub-optimal (worse) results when compared to Next and CGM-DVFS, especially for the render and fft applications. This is due to the fact that for delayed RL approaches such as Q-Learning the agent must explore the dynamic system (dynamic environment) long enough to find the optimal outcome [16]. Although delayed RL approaches are beneficial to optimise the power consumption and temperature of the system in an application-agnostic manner, often, given the number of actions required (actions to perform DVFS on the CPU, GPU and RAM) if the agent is not allowed to explore the dynamic environment for long enough, then the method will result in sub-optimal power consumption. On the other hand, application-specific profiling approaches such as CGM-DVFS will result in close-to-optimal power consumption since these approaches are specific to certain applications.

7. Conclusions

In this paper, we studied the effect of different frequency scaling levels on memory in regard to the total power consumption in a mobile MPSoC. We also proposed CGM-DVFS, an agent designed to perform DVFS on big and LITTLE CPUs, GPUs and RAM on a mobile MPSoC and the experimental results proved the efficacy of CGM-DVFS in reducing power consumption and peak temperature while catering to performance requirements compared to the state-of-the-art approaches. Through our experimental results, we also showed that application-specific profiling approaches such as CGM-DVFS outperform delayed reinforcement learning approaches such as Q-Learning and result in closer-to-optimal power consumption when the system (environment) is dynamic.

Author Contributions: Conceptualisation, S.D.; methodology, S.D.; software, S.D.; validation, S.D.; formal analysis, S.D.; investigation, S.D.; resources, S.D.; data curation, S.D.; writing—original draft preparation, S.D.; writing—review and editing, S.D., S.I., S.S., A.K.S. and K.M.-M.; visualisation, S.D.; supervision, S.D.; project administration, S.D.; funding acquisition, S.D. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by Nosh Technologies under Grant nosh/agri-tech-000001.

Data Availability Statement: Experimental evaluation data are accessible from https://github.com/somdipdey/Data-for-CGM-DVFS (accessed on 14 March 2022).

Conflicts of Interest: This research was pursued such that part of the proposed methodology could be implemented as a feature in the commercial mobile application named nosh–Food Management app: https://nosh.tech, (accessed on 14 March 2022) to improve battery consumption and performance of the said mobile application.

References

1. Cho, H.D.; Engineer, P.D.P.; Chung, K.; Kim, T. Benefits of the Big. LITTLE Architecture. Available online: https://s3.ap-northeast-2.amazonaws.com/global.semi.static/Benefits_of_the_bigLITTLE_Architecture.pdf (accessed on 4 February 2022).
2. Singh, A.K.; Dey, S.; McDonald-Maier, K.; Basireddy, K.R.; Merrett, G.V.; Al-Hashimi, B.M. Dynamic Energy and Thermal Management of Multi-Core Mobile Platforms: A Survey. *IEEE Des. Test* **2020**, *37*, 25–33. [CrossRef]
3. Singh, A.K.; Shafique, M.; Kumar, A.; Henkel, J. Mapping on multi/many-core systems: Survey of current and emerging trends. In Proceedings of the 2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC), Austin, TX, USA, 29 May–7 June 2013; pp. 1–10.
4. Dey, S.; Singh, A.K.; Wang, X.; McDonald-Maier, K. User Interaction Aware Reinforcement Learning for Power and Thermal Efficiency of CPU-GPU Mobile MPSoCs. In Proceedings of the 2020 DATE IEEE, Grenoble, France, 9–13 March 2020.
5. Dey, S.; Singh, A.K.; Wang, X.; McDonald-Maier, K. Edgecoolingmode: An agent based thermal management mechanism for dvfs enabled heterogeneous mpsocs. In Proceedings of the 2019 VLSID IEEE, Delhi, India, 5–9 January 2019.
6. Reddy, B.K.; Singh, A.K.; Biswas, D.; Merrett, G.V.; Al-Hashimi, B.M. Inter-cluster Thread-to-core Mapping and DVFS on Heterogeneous Multi-cores. *IEEE Trans. Multi-Scale Comput. Syst.* **2018**, *4*, 369–382. [CrossRef]
7. Dey, S.; Singh, A.K.; Saha, S.; Wang, X.; McDonald-Maier, K.D. RewardProfiler: A Reward Based Design Space Profiler on DVFS Enabled MPSoCs. In Proceedings of the 2019 CSCloud/2019 EdgeCom IEEE, Paris, France, 21–23 June 2019.
8. Odroid-XU4. Available online: https://www.hardkernel.com/shop/odroid-xu4-special-price/ (accessed on 4 February 2022).
9. Exynos 5 Octa (5422). Available online: https://www.samsung.com/exynos (accessed on 23 July 2018).
10. Kim, M.; Kim, K.; Geraci, J.R.; Hong, S. Utilization-aware load balancing for the energy efficient operation of the big. LITTLE processor. In Proceedings of the 2014 Design, Automation & Test in Europe Conference & Exhibition (DATE) IEEE, Dresden, Germany, 24–28 March 2014.
11. Bienia, C. Benchmarking Modern Multiprocessors. Ph.D. Thesis, Princeton University, Princeton, NJ, USA, 2011.
12. Dey, S.; Singh, A.K.; Prasad, D.K.; Mcdonald-Maier, K.D. SoCodeCNN: Program Source Code for Visual CNN Classification Using Computer Vision Methodology. *IEEE Access* **2019**, *7*, 157158–157172. [CrossRef]
13. Pathania, A.; Jiao, Q.; Prakash, A.; Mitra, T.I. Integrated CPU-GPU power management for 3D mobile games. In Proceedings of the 2014 51st ACM/EDAC/IEEE DAC IEEE, San Francisco, CA, USA, 1–5 June 2014.
14. Isuwa, S.; Dey, S.; Singh, A.K.; McDonald-Maier, K. TEEM: Online Thermal-and Energy-Efficiency Management on CPU-GPU MPSoCs. In Proceedings of the 2019 DATE IEEE, Florence, Italy, 25–29 March 2019; pp. 438–443.
15. Hsieh, C.Y.; Park, J.-G.; Dutt, N.; Lim, S.-S. Memory-aware cooperative CPU-GPU DVFS governor for mobile games. In Proceedings of the 2015 13th IEEE Symposium on Embedded Systems For Real-time Multimedia (ESTIMedia), Amsterdam, The Netherlands, 8–9 October 2015; pp. 1–8.

16. John, G.H. When the Best Move Isn't Optimal: Q-Learning with Exploration. Available online: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.6.5630&rep=rep1&type=pdf (accessed on 4 February 2022).
17. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
18. Woo, S.C.; Ohara, M.; Torrie, E.; Singh, J.P.; Gupta, A. The SPLASH-2 programs: Characterization and methodological considerations. *ACM SIGARCH Comput. Archit. News* **1995**, *23*, 24–36. [CrossRef]
19. Roy Longbottom's PC Benchmark Collection. Available online: http://www.roylongbottom.org.uk (accessed on 10 October 2018).
20. Longbottom, R. Whetstone Benchmark History and Results. Available online: https://www.researchgate.net/profile/Roy-Longbottom/publication/318755466_Whetstone_Benchmark_History_and_Results/links/597b906aaca272d568b85fea/Whetstone-Benchmark-History-and-Results.pdf (accessed on 10 October 2018).
21. Rivest, R.L.; Shamir, A.; Adleman, L.M. Cryptographic Communications System and Method. U.S. Patent 4,405,829, 20 September 1983.
22. These Were the 10 Most-Downloaded Apps of the Decade. Available online: https://www.ndtv.com/offbeat/these-were-the-10-most-downloaded-apps-of-the-decade-2150290 (accessed on 10 October 2018).
23. Thakkar, J. Types of Encryption: 5 Encryption Algorithms and How to Choose the Right One. 2020. Available online: https://securityboulevard.com/2020/05/types-of-encryption-5-encryption-algorithms-how-to-choose-the-right-one/ (accessed on accessed on 10 October 2018).
24. Dey, S.; Singh, A.K.; McDonald-Maier, K.D. P-EdgeCoolingMode: An Agent Based Performance Aware Thermal Management Unit for DVFS Enabled Heterogeneous MPSoCs. *IET Comput. Digit. Tech.* **2019**, *13*, 514–523. [CrossRef]
25. Mandal, S.K.; Bhat, G.; Patil, C.A.; Doppa, J.R.; Pande, P.P.; Ogras, U.Y. Dynamic resource management of heterogeneous mobile platforms via imitation learning. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2019**, *27*, 2842–2854. [CrossRef]
26. Singh, A.K.; Basireddy, K.R.; Prakash, A.; Merrett, G.V.; Al-Hashimi, B.M. Collaborative adaptation for energy-efficient heterogeneous mobile SoCs. *IEEE Trans. Comput.* **2019**, *69*, 185–197. [CrossRef]
27. Dey, S.; Singh, A.K.; Wang, X.; McDonald-Maier, K.D. DeadPool: Performance Deadline Based Frequency Pooling and Thermal Management Agent in DVFS Enabled MPSoCs. In Proceedings of the 2019 6th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2019 5th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom), Paris, France, 21–23 June 2019.
28. David, R.; Bogdan, P.; Marculescu, R. Dynamic power management for multicores: Case study using the Intel SCC. In Proceedings of the 2012 IEEE/IFIP 20th International Conference on VLSI and System-on-Chip (VLSI-SoC), Santa Cruz, CA, USA, 7–10 October 2012; pp. 147–152.
29. Bogdan, P.; Marculescu, R.; Jain, S. Dynamic power management for multidomain system-on-chip platforms: An optimal control approach. *ACM Trans. Des. Autom. Electron. Syst. (TODAES)* **2013**, *18*, 1–20. [CrossRef]
30. Li, X.; Li, G. An Adaptive CPU-GPU Governing Framework for Mobile Games on big. LITTLE Architectures. *IEEE Trans. Comput.* **2020**, *70*, 1472–1483. [CrossRef]
31. Deng, Q.; Meisner, D.; Bhattacharjee, A.; Wenisch, T.F.; Bianchini, R. MultiScale: Memory system DVFS with multiple memory controllers. In Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design, Redondo Beach, CA,, USA, 30 July–1 August 2012.
32. Deng, Q.; Meisner, D.; Bhattacharjee, A.; Wenisch, T.F.; Bianchini, R. Coscale: Coordinating cpu and memory system dvfs in server systems. In Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture, Vancouver, BC, Canada, 1–5 December 2012.
33. Chang, K.K.; Yağlıkçı, A.G.; Ghose, S.; Agrawal, A.; Chatterjee, N.; Kashyap, A.;Lee, D.; O'Connor, M. Hassan, H.; Mutlu, O. Understanding reduced-voltage operation in modern DRAM devices: Experimental characterization, analysis, and mechanisms. In Proceedings of the ACM on Measurement and Analysis of Computing Systems, Urbana-Champaign, IL, USA, 5–9 June 2017.
34. Begum, R.; Werner, D.; Hempstead, M.; Prasad, G.; Challen, G. Energy-performance trade-offs on energy-constrained devices with multi-component DVFS. In Proceedings of the 2015 IEEE International Symposium on Workload Characterization, Atlanta, GA, USA, 4–6 October 2015.
35. Begum, R.; Hempstead, M.; Srinivasa, G.P.; Challen, G. Algorithms for CPU and DRAM DVFS under inefficiency constraints. In Proceedings of the 2016 IEEE 34th International Conference on Computer Design (ICCD), Scottsdale, AZ, USA, 2–5 October 2016.
36. Mendis, H.R.; Chen, W.M.; Indrusiak, L.S.; Kuo, T. W.; Hsiu, P.C. Impact of memory frequency scaling on user-centric smartphone workloads. In Proceedings of the 33rd Annual ACM Symposium on Applied Computing, Pau, France, 9–13 April 2018.
37. Gensh, R.; Aalsaud, A.; Rafiev, A.; Xia, F.; Iliasov, A.; Romanovsky, A.; Yakovlev, A. *Experiments with Odroid-xu3 Board*. Available online: https://eprints.ncl.ac.uk/file_store/production/213859/D9F017EA-31CC-4A4A-AEF9-BDA775890FAB.pdf (accessed on 4 February 2022).
38. Odroid SmartPower2. Available online: https://www.odroid.co.uk/odroid-smart-power-2 (accessed on 23 July 2018).
39. Budiu, R. Multitasking on Mobile Devices. In *White Paper of Nielsen Norman Group logoNielsen Norman Group*; NNGroup: Fremont, CA, USA, 2015.
40. Soo, S. *Object Detection Using Haar-Cascade Classifier*; Institute of Computer Science, University of Tartu: Tartu, Estonia 2014; pp. 1–12.

41. Kalafatić, Z. Available online: Traffic Sign Detection and Recognition. Available online: https://shorturl.at/stDO6 (accessed on 4 February 2022).
42. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
43. Iranfar, A.; Kamal, M.; Afzali-Kusha, A.; Pedram, M.; Atienza, D. Thespot: Thermal stress-aware power and temperature management for multiprocessor systems-on-chip. *IEEE Trans. Comput. Des. Integr. Circuits Syst.* **2018**, *37*, 1532–1545. [CrossRef]