IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 17, 2019, accepted February 6, 2019, date of publication February 22, 2019, date of current version March 8, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2901010

Run-Time Resource Management Controller for Power Efficiency of GP-GPU Architecture

SHAHERYAR NAJAM[1], JAMEEL AHMED[1], SAAD MASOOD[2], AND CHUADHRY MUJEEB AHMED[3]

[1]Riphah International University, Islamabad 44000, Pakistan
[2]Pakistan Institute of Engineering and Technology, Multan 66000, Pakistan
[3]Singapore University of Technology and Design 487372, Singapore

Corresponding author: Shaheryar Najam (shaheryar.najam@riphah.edu.pk)

This work was supported in part by the Department of Electrical Engineering, Faculty of Engineering and Applied Sciences, Riphah International University, under the seed money program, and in part by the Department of Information Systems Technology and Design, Singapore University of Technology and Design (SUTD).

ABSTRACT The demand for high-performance computing (HPC) has been increasing since the invention of computing technology. This led to the invocation of sophisticated multi/many-core processors with high performance. Graphical processing units (GPUs) have emerged as an important innovation in the many-core era as it features a high number of processors. The GPU acts as a computational accelerator that can significantly reduce the computational time of the HPC, as it can offer standout parallelism for high-end computing applications such as graphics designing. However, increasing the resources has resulted in higher power consumption and heat dissipation which has been a challenging problem for modern HPC Units. On the other hand, because of the dynamic nature of workload, a large amount of parallelism, offered by these many-core processors, is often underutilized. An ideal system would be smart enough to efficiently utilize resources and save power where less workload is available. Reducing the resources dynamically has direct implications on the performance of the system. However, if less workload is available, reducing the resources would not harm the performance, rather it would save power with less to no trade-off in overall throughput of the system. In this paper, a smart power and performance efficient resource management controller for general purpose-GPU architecture is presented. The proposed controller, based on a feedback mechanism, keeps on analyzing the current frequency of central processing unit (CPU) and GPU, number of active cores of the CPU and utilization of CPU and GPU. On the basis of collected data, the proposed controller which features fuzzy type-2 as an optimizing mechanism tries to create a balance between performance and power consumption. The results are evaluated against various benchmarks on NVIDIA TK1 GPU kit and by using dynamic voltage and frequency scaling and core gating, up to 47 % reduction in power consumption has been achieved.

INDEX TERMS HPC, CPU, GP-GPU, GPU, Nvidia, fuzzy type-2.

I. INTRODUCTION

To achieve the ever increasing demand for High Performance Computing (HPC), designers have opted to enhance and further increase the design complexity and resources available to the HPC units. However, due to this fact, saturation has stepped in and further increasing design complexity and clock frequency seems to be unrealizable. This saturation resulted in the emergence of multi/many-core architectures. Presently, many core systems along with the capability of multi-threaded computing, have not only attracted the masses, due to the high level of parallelism available to these units but are also considered to be de facto standard for HPC units. In this regard, GPUs are being considered as a crucial unit for HPC, due to the significantly high number of cores and parallelism and are also solving the problem of computationally analyzing big data in servers, graphical data in Earth & Life Science Groups and solving larger algorithms and complex techniques to find solutions more quickly and higher accuracy. However, there are power

The associate editor coordinating the review of this manuscript and approving it for publication was Ran Cheng.

consumption and heat dissipation issues related to these GPUs specifically, in the context of portable HPC systems. Further, the large amount of parallelism available to these GPUs is often underutilized [2], [3]. Due to the mentioned issue, it acts as a liability rather than a benefit.

On the other hand, throughput and the chip area have been the main driving forces behind microprocessor design. However, the invention of the transistor has realized the portability of the computers and later on, the ever increasing range of battery operated devices with often complex functionality has led to a paradigm shift in design optimizations towards greater power efficiency in complement with the increased throughput. This strive for low power processing has been a challenge for the designers and researchers to optimize each component of the computing unit. However, optimization in terms of power usually results in the expense of the throughput, and which may depreciate the over-all gain of an HPC unit.

Power efficiency has become a crucial challenge due to power constraints of HPC and it has now become a primary architectural design constraint for computing platforms ranging from power optimized embedded processors to high performance parallel computing architectures [5]. The increasing computing power of GPUs has provided higher throughput in comparison with central processing Units (CPUs). Therefore, GPUs are not restricted for graphics application and many programmers have been incorporating GPUs for various kind of HPC. However, optimizing GPU kernel to achieve high performance along with better power efficiency is still a challenge.

At the same time it is worth mentioning that since the beginning of computer technology, Moore's law has been followed, which predicted the doubling of transistors in a core after every 18 months [4]. This increase in the transistor density has further been complimented by increasing operational clock frequency. However, as a result, power consumption and heat dissipation have increased at the same rate till the verge, where a further increase in the clock frequency of HPC units seemed to be unrealizable.

A smart solution to the above mentioned problems will be a system that is smart enough to reconfigure itself according to the amount and parallelism available in the workload. Therefore, we propose the development of run-time GPU based power/performance optimizing algorithms in such a way that GPUs would be able to implement larger algorithms and complex techniques to find solutions more quickly, with greater accuracy, and at a less power consumption.

This paper presents Fuzzy Type 2 Based Resource Management Controller (FBRMC-2) for power efficiency of HPC units. The core aim of this research work was to minimize power consumption when processing of CPU & GPU is not fully utilized. To achieve the required target, FBMRC-2 being the control unit, incorporated two well-known power efficient techniques which are Dynamic voltage and Frequency Scaling (DVFS) and Core Gating (CG). Fuzzy logic was an appropriate choice as it does not require detailed computing modeling [6], [7].

Both above mentioned techniques acquire their own importance in minimizing power utilization of computing devices for series and parallel nature of workload. Because of this fact, recent research is focusing on algorithms incorporating the DVFS and CG techniques to bring down power consumption of computing units for such workload, where full capacity of the computing unit, in terms of performance, is not required [18], [32], [33]. DVFS has a significant impact on performance, utilization and power consumption of CPU-GPU based architectures and it is a widely used technique for power optimization in computer architectures [34]–[36]. The working principle of DVFS is to minimize power-consumption in scenarios where limited processing capabilities are required. In many cases, the requirement of advanced processing capability are not high and little amount of processor's throughput will be sufficient to complete the task. In such cases, executing tasks at the maximum frequency is not beneficial, rather adjusting the operational frequency of processor as per requirement of user's workload will produce better results. Thus, creating a wide opportunity window to reduce the overall power consumption of the device. In such cases, incorporating DVFS technique has the potential to minimize both power consumption and heat dissipation. However, it is worth mentioning here that for power saving, minimum frequency is not the optimal frequency all the times [37]. This is because of memory latency issue; therefore, determining the optimal frequency for any task is crucial. Various algorithms exploiting DVFS scheme have been presented by research community [8]–[14].

Second targeted technique for power optimization is CG (shutting the cores amount of available workload is not high). Since the overall trend in computing has experienced a significant shift towards multi/many cores approach where the number of cores in computing systems is increasing significantly. Because of this fact, increase in power consumption and heat dissipation has been observed at an unprecedented scale. This increasing number of cores provides the high level of parallelism which can only be exploited when the workload is distributed in parallel [12]. In such scenarios, where the workload is not distributed in a parallel manner, gating the specific core and running the processors with less number of cores can be very beneficial for power optimization. Therefore, reconfiguration and smart utilization of resources need to be introduced. The research community has presented various algorithms that exploit the CG technique for scenarios where work either workload or parallelism available in workload is limited [38], [39]. In addition to this, it was observed while analyzing the results of CG that for series natured benchmarks, the impact of CG on the reduction of power consumption is significantly high as compared to parallel benchmarks [39]. The key contribution of this research work is designing the FBRMC-2, a run-time controller, for power optimization in computing devices by selecting appropriate operating frequency of CPU & GPU and the active

number of CPU's cores using DVFS and CG techniques respectively.

Rest of the paper is arranged as follows. Section II presents the detailed literature survey along with a comparison with the proposed idea. Section III presents the methodology about proposed idea and implementation of a system. Section IV presents details of the experimental setup and evaluated results. Finally, the conclusion is presented in Section V.

II. LITERATURE REVIEW

Under the prevailing demand for high-end computing, GPUs are gaining increasing importance due to its large number of parallel processors. Because of this fact, various research communities have shifted their focus toward GPUs and considers it to be a solution for this increasing demand for HPC. However, there are power and heat dissipation issues associated with modern GPUs. Various techniques have been proposed to solve the above mentioned problem. An optimal algorithm for multi-processor allocation in GPU system is presented by Jararweh *et al.* [16]. The proposed algorithm reduces power consumption while maintaining the application required performance. Cost of electricity and cooling infrastructure has been rising and due to this fact, power consumption and heat dissipation have been realized as the critical issue in modern HPC units. The Multi-Processor Allocation (MP-Alloc) algorithm tries to decide the suitable number of processors that can decrease power consumption, resources over-provisioning, and maintain the performance simultaneously. Memory Bandwidth Utilization (BWU) metric was used to predict the number of processors required for the application. Evaluation results predicted CG as a promising technique for power saving without compromising on the overall performance of the system.

Similarly, an Integrated Power and Performance (IPP) prediction model for a GPU architecture has been proposed by Hong and Kim [15]. The main idea is that using more cores does not result in performance improvement when an application reaches the peak memory bandwidth. Unlike most of the previously available models, an empirical power model for the GPU require measured execution times, hardware performance counters, or architectural simulations, was proposed in which IPP predicts execution times to calculate dynamic power events. The outcome of IPP was then used to control the number of running cores. The relation between the increases in power consumption with the increases in temperature was also modeled. With the predicted optimal number of active cores, results have shown, run-time GPU energy consumption was reduced by 22.09%.

Impact of DVFS, on the performance and energy efficiency for various applications, running on GPUs and comparison with those of DVFS for CPU computing is presented by Ge *et al.* [17]. Impact of Dynamic Voltage and frequency scaling (DVFS) is different from the DVFS on CPU computing. For example, if GPU is running compute-bound high-performance and high-throughput workload, then the system power consumption and system performance are approximately in direct relation with the GPU frequency. Hence, with a permitted power limit, increasing the GPU frequency results in better performance without increasing energy to a greater extent. This research further gives detailed analytical explanations of observed trends and exceptions along with their causes. The findings presented in this paper have shown significant potential to impact future CPU and GPU architectures to acquire better efficiency in terms of energy consumption. This research has also pointed out directions for designing effective DVFS schedulers for heterogeneous systems.

In order to reduce power consumption of embedded GPUs, software-based DVFS framework was presented by You and Chung [18]. In this work, an on-demand DVFS policy module was implemented that incorporate a work queue mechanism. Dynamic Warp Resizing (DWR) proposed by Lashgar *et al.* [19]. In modern GPUs synchronize threads are usually grouped in warps. Various parameters like divergence, synchronization overhead, and the efficiency of memory access coalescing are affected by the number of threads included in each warp (or warp size). Performance penalty associated with branch and memory divergence can be reduced by keeping the wrap size small. However, this is achieved at the expense of a reduction in memory coalescing. Large warps significantly enhance memory coalescing. On the other hand, it also increases branch and memory divergence. In determining the warp size, returning best performance, Dynamic workload behavior, including branch/memory divergence and coalescing, is an important factor. DWR, proposed in this work, outperforms static warp size decisions, up to 2.28X.

Power optimizing techniques and algorithms are not limited to GPUs only. Rather, they have their own benefits in MPSoC as well. For full FPGA reconfiguration, two periodic real-time task scheduling algorithms were proposed by Danne and Platzner [20] Concept of Earliest Deadline First (EDF) was used in the first algorithm and named it EDF-NF. On the other side, the second algorithm used the model of servers that implies reserving execution time and area for other tasks which is also known as Merge Server Distribute Load (MSDL). In comparing both techniques, it was observed that EDF-NF was better for system utilization but keeping feasibly for larger real-time tasks sets in mind, it was MSDL which outrun EDF-NF. Research communities have presented various other scheduling techniques along with applications. [21]–[24]

Run-time energy-aware scheduling has also been investigated in various research articles specially for Multi-Processor System on Chip (MPSoCs) Thread scheduling is generally classified into two main classes; balanced and unbalanced scheduling. In balanced scheduling, an equal amount of threads are distributed among the cores. Performance of various scheduling schemes has been analyzed by DeVuyst *et al.* [25], considering both energy and performance. It has been observed in results that the scheduling

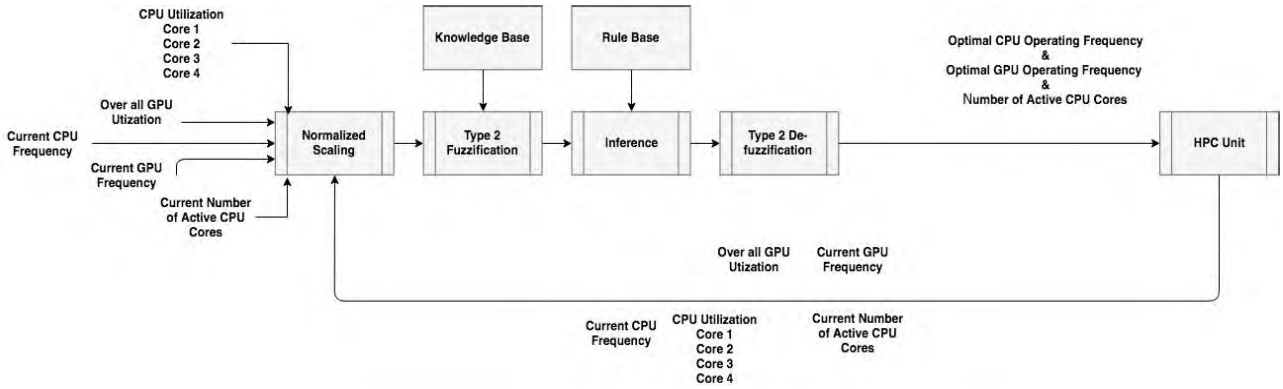


FIGURE 1. Block diagram of FBRMC-2.

of threads in an uneven manner often outclasses balanced scheduling model. This is due to the fact that greater performance can be achieved by linking the certain number of threads together on single core rather than by allocating them to several cores. Communication between dynamically changing tasks and their synchronization have also been challenging issues for reconfigurable multicore platforms. In work presented in [26], a detailed analysis of the performance of various task scheduling algorithms for minimizing schedule length combined along with energy constraints has been performed. This work also analyzed the algorithms for reducing overall energy consumption combined with a constraint of schedule length on DVFS supported multiprocessor systems. For scheduling of concurrent tasks on multicore platforms, Yang *et al.* [27] presented a model that used the combination of both online and offline scheduling approaches to exploit the run-time tradeoff between power and performance. This presented work was executed as an extension of another research work on power-efficient scheduling which was based on grey box modeling. Later scheme of scheduling was presented by Prayati *et al.* [28].

Ma *et al.* [29] discussed scheduling scheme, based on design time and run-time, for management of concurrent task in the real-time application running on the heterogeneous multicore platform. At the design time, the set of schedule and assignment was assigned to each task using Petro curves, and a lightweight scheduler was incorporated to carefully choose the optimal working points making full use of dynamic and non-deterministic behavior of the system, at run-time. For an MPEG4 texture decoder application, results have shown that using this approach, a significant reduction on power can be achieved while maintaining the overall performance.

III. FUZZY TYPE 2 BASED RESOURCE MANAGEMENT CONTROLLER

FBRMC-2, based on fuzzy logic type-2 optimizing algorithm is designed on a feedback mechanism as shown in Fig.1. FBRMC-2 continuously analyze the stats of processor and based on those stats, an algorithm is designed to find the optimal set of resources to create a balance between performance and power consumption.

A. INPUT PARAMETERS

The decision making ability of FBRMC-2 is dependent on its input parameters; therefore, input parameters were selected very carefully. Usually, decreasing the resources of processor results in significant reduction in power consumption. However, it is very crucial to select the optimal resources; otherwise, the probability of creating lag in performance of the system is significantly high. Therefore run-time stats of processor have a critical impact on this task. Following stats of processors were selected as input parameters for FBRMC-2.

1) UTILIZATION OF PROCESSOR

The utilization of processor plays important role in deciding the optimal resources for any task. The balance between performance and power consumption is only possible in cases where the utilization of the processor is maximized. Therefore, FBRMC-2 always keeps on tracking the utilization of processors and tries to maximize it by carefully reducing resources. Utilization was measured by using *tegrastats* utility, provided for Tegra based devices like NVIDIA's embedded Tegra K1 SOC. There are two types of utilization that have been incorporated as input parameters for FBMRC-2.

a: UTILIZATION OF CPU

FBRMC-2 incorporates DVFS and core gating techniques to reduce the power consumption of HPC. However, reducing power has a direct relation with maximizing utilization of available resources. Tracking the CPU utilization makes the selection of optimal CPU frequency and the active number of CPU cores more accurate.

b: UTILIZATION OF GPU

FBRMC-2 also scales the GPU frequency to reduce the power consumption. Therefore, FBRMC-2 keeps on monitoring the utilization of GPU to select the optimal frequency which in return maximizes the utilization of GPU.

2) CURRENT FREQUENCY OF CPU & GPU

FBRMC-2 also keeps monitoring the current CPU and GPU frequency. As FBRMC-2 is designed to select an optimal frequency of CPU and GPU, it is important to keep the record of current frequency. Keeping the record of current frequency of CPU and GPU, along with their respective utilization, helps in evaluating the overall scenario in terms of amount and parallelism available in workload which in return brings FBRMC-2 in a better position to maximize the utilization by reducing resources.

3) NUMBER OF ACTIVE CPU CORES

FBRMC-2 use core gating technique to reduce the number of active core of CPU only. FBRMC-2 reduces the active core of CPU when either workload is less or less or not parallelism is available in workload. Increasing the number of core is only beneficial when the workload is distributed in parallel. Therefore, reducing the active core of CPU will not produce drastic effects on overall performance when the nature of workload is series. However, before reducing the active cores, it is important to analyze the current number of active CPU cores and its utilization so that proposed active number of cores may produce effective results in balancing the performance and power consumption.

B. WORKING OF FBRMC-2

Dynamic and run-time reconfiguration of hardware is not a new concept, recent research suggests the reconfiguration of the system is based on performance statistics obtained from the underlying platform but there is a need to go beyond traditional reconfiguration schemes in order to save power while keeping throughput conserved. This is where FBRMC-2 system can be deployed as an intelligent controller to decide the optimal settings of reconfigurable parameters. Fuzzy logic type-2 is one of the suitable candidates to design a bridge between human logic and computer. In design exploration, fuzzy logic has significant application for finding optimization [40], [41]. Furthermore, the capability of fuzzy inference system to interpret linguistic rules and to defuzzify the result to a crisp set of values, without deploying a sophisticated mathematical model has made fuzzy logic a smart tool for attaining the reconfiguration of energy and performance aware multi/man core system. In previous work, Shaheryar *et al.* [30] presented the fuzzy logic type-1 based optimizing controller for MPSoC and results were significantly better than available governors of Ubuntu operating system. However, results predicted even better response for GP-GPU based HPC units like Nvidia TK1 GPU kit. Furthermore, fuzzy type-2 has been incorporated as very often, in order to design the rules in the fuzzy logic system, incorporated knowledge is uncertain. This uncertainty leads to rules having uncertain antecedents and/or consequents, which in turn translates into uncertain antecedent and/or consequent membership functions [31]. These issues of uncertainty have further been improved by fuzzy type-2 model. Feedback mechanism has also been improved with better input parameters that can define the overall available workload scenario and the current configuration of the processor more precisely. On the other hand, an additional technique of scaling GPU's frequency along with other techniques (core gating and scaling CPU's frequency) incorporated in previous research work, also plays a critical role in minimizing power and heat dissipation.

The proposed power optimizing scheme, make use of closed-loop feedback based fuzzy type-2 and it is interesting to note that the proposed system is based on the typical control model but the main advantage of using fuzzy type-2 is that precise impact modeling of input parameters over outputs is not required, as the fuzzy type-2 based controllers have natural ability to deal with vague information based on a rule base of linguistic terms. Reconfigurable parameters were identified in order to bring down energy consumption without disturbing the throughput to a greater extent. In this work, the targeted parameters are *CPU's operating frequency*, *number of active CPU's cores* and *GPU's operating frequency*. FBMRC-2 keeps on monitoring overall CPU utilization, GPU utilization, CPU's operating frequency, GPU's operating frequency, active number of CPU's cores.

Initially, in fuzzification step, three different fuzzy subsets were defined and assigned to their membership function respectively i.e. low, middle and high boundary and are denoted by μ_{A1}, μ_{B1} and μ_{C1} respectively. For example, utilization (in percentage) of CPU of the processor varies from 0 to 1. Lower membership function μ_{A1} is bounded between 0 and 40%, μ_{B1} membership means moderate utilization and it is bounded between 25% and 75%, μ_{C1} membership means high utilization and it is bounded between 60% and 100%. As FBRMC-2 works on fuzzy type-2 model and fuzzy type-2 contain 3 dimensional membership; therefore, membership function has further been classified as *membership function 1* and *membership function 2*. In similar approach, membership function for rest of input and output parameters were designed and details are provided in Table 1 and 2. Establishment of a relationship between input and output parameters of the system is critical, subsequent to detailed analysis, separate fuzzy logic rules were designed for power optimization of CPU and GPU as shown in Table 3 & 4 respectively. The rules were designed carefully to make sure new resource configuration would lead to optimized power consumption. For example, in Rule 12, Table.3: if CPU's utilization is *medium*, CPU's current frequency is *low* and the number of active CPU's core is *high*, then it shows that computing device needs to improve utilization by reducing the resource. In this case, CPU's current frequency is already low, therefore; only option to reduce resources is by reducing the number of active CPU cores. Because of this reason, the number of active CPU cores has been reduced from high to medium. Similarly, in Rule 13, CPU's current frequency was reduced from medium to low as numbers of active CPU's cores were already low. For further understanding, consider

TABLE 1. Membership functions of Fuzzy type-2 for input variables.

$$\mu_{A1}=\begin{cases}0 & if\ 40\% \le \text{CPU/GPU Utilization} \le 0\%\\ \frac{40-x}{40} & if\ 0\% \le \text{CPU/GPU Utilization} \le 40\%\end{cases}$$

$$\mu_{A2}=\begin{cases}0 & if\ 40\% \le \text{CPU/GPU Utilization} \le 0\%\\ \frac{40-x}{50} & if\ 0\% \le \text{CPU/GPU Utilization} \le 40\%\end{cases}$$

$$\mu_{B1}=\begin{cases}0 & if\ 75\% \le \text{CPU/GPU Utilization} \le 0\%\\ \frac{x-25}{25} & if\ 25\% \le \text{CPU/GPU Utilization} \le 50\%\\ \frac{75-x}{25} & if\ 50\% \le \text{CPU/GPU Utilization} \le 75\%\end{cases}$$

$$\mu_{B2}=\begin{cases}0 & if\ 75\% \le \text{CPU/GPU Utilization} \le 0\%\\ \frac{x-25}{31.25} & if\ 25\% \le \text{CPU/GPU Utilization} \le 50\%\\ \frac{75-x}{31.25} & if\ 50\% \le \text{CPU/GPU Utilization} \le 75\%\end{cases}$$

$$\mu_{C1}=\begin{cases}0 & if\ 100\% \le \text{CPU/GPU Utilization} \le 60\%\\ \frac{x-60}{40} & if\ 60\% \le \text{CPU/GPU Utilization} \le 100\%\end{cases}$$

$$\mu_{C2}=\begin{cases}0 & if\ 100\% \le \text{CPU/GPU Utilization} \le 60\%\\ \frac{x-60}{50} & if\ 60\% \le \text{CPU/GPU Utilization} \le 100\%\end{cases}$$

CPU/GPU Utilization

$$\mu_{A1}=\begin{cases}0 & if\ 0.35 \le \text{CPU's Current Frequency} \le 0\\ \frac{0.35-x}{0.35} & if\ 0 \le \text{CPU's Current Frequency} \le 0.35\end{cases}$$

$$\mu_{A2}=\begin{cases}0 & if\ 0.35 \le \text{CPU's Current Frequency} \le 0\\ \frac{0.35-x}{0.4375} & if\ 0 \le \text{CPU's Current Frequency} \le 0.35\end{cases}$$

$$\mu_{B1}=\begin{cases}0 & if\ 0.8 \le \text{CPU's Current Frequency} \le 0\\ \frac{x-0.2}{0.3} & if\ 0.2 \le \text{CPU's Current Frequency} \le 0.5\\ \frac{0.8-x}{0.3} & if\ 0.5 \le \text{CPU's Current Frequency} \le 0.8\end{cases}$$

$$\mu_{B2}=\begin{cases}0 & if\ 0.8 \le \text{CPU's Current Frequency} \le 0\\ \frac{x-0.2}{0.375} & if\ 0.2 \le \text{CPU's Current Frequency} \le 0.5\\ \frac{0.8-x}{0.375} & if\ 0.5 \le \text{CPU's Current Frequency} \le 0.8\end{cases}$$

$$\mu_{C1}=\begin{cases}0 & if\ 1.0 \le \text{CPU's Current Frequency} \le 0.65\\ \frac{x-0.65}{0.35} & if\ 0.65 \le \text{CPU's Current Frequency} \le 1.0\end{cases}$$

$$\mu_{C2}=\begin{cases}0 & if\ 1.0 \le \text{CPU's Current Frequency} \le 0.65\\ \frac{x-0.65}{0.4375} & if\ 0.65 \le \text{CPU's Current Frequency} \le 1.0\end{cases}$$

Normalized CPU's Current Frequency

$$\mu_{A1}=\begin{cases}0 & if\ 0.35 \le \text{GPU's Current Frequency} \le 0\\ \frac{0.35-x}{0.35} & if\ 0 \le \text{GPU's Current Frequency} \le 0.35\end{cases}$$

$$\mu_{A2}=\begin{cases}0 & if\ 0.35 \le \text{GPU's Current Frequency} \le 0\\ \frac{0.35-x}{0.4375} & if\ 0 \le \text{GPU's Current Frequency} \le 0.35\end{cases}$$

$$\mu_{B1}=\begin{cases}0 & if\ 0.8 \le \text{GPU's Current Frequency} \le 0\\ \frac{x-0.2}{0.3} & if\ 0.2 \le \text{GPU's Current Frequency} \le 0.5\\ \frac{0.8-x}{0.3} & if\ 0.5 \le \text{GPU's Current Frequency} \le 0.8\end{cases}$$

$$\mu_{B2}=\begin{cases}0 & if\ 0.8 \le \text{GPU's Current Frequency} \le 0\\ \frac{x-0.2}{0.375} & if\ 0.2 \le \text{GPU's Current Frequency} \le 0.5\\ \frac{0.8-x}{0.375} & if\ 0.5 \le \text{GPU's Current Frequency} \le 0.8\end{cases}$$

$$\mu_{C1}=\begin{cases}0 & if\ 1.0 \le \text{GPU's Current Frequency} \le 0.65\\ \frac{x-0.65}{0.35} & if\ 0.65 \le \text{GPU's Current Frequency} \le 1.0\end{cases}$$

$$\mu_{C2}=\begin{cases}0 & if\ 1.0 \le \text{GPU's Current Frequency} \le 0.65\\ \frac{x-0.65}{0.4375} & if\ 0.65 \le \text{GPU's Current Frequency} \le 1.0\end{cases}$$

Normalized GPU's Current Frequency

$$\mu_{A}=\begin{cases}0 & if\ \text{Active CPU's cores} \le 1\ or \ge 2\\ 1 & if\ 1 \le \text{Active CPU's cores} \le 2\end{cases}$$

$$\mu_{B}=\begin{cases}0 & if\ \text{Active CPU's cores} \le 1.5\ or \ge 3\\ 1 & if\ 1.5 \le \text{Active CPU's cores} \le 3\end{cases}$$

$$\mu_{C}=\begin{cases}0 & if\ \text{Active CPU's cores} \le 2.5\ or \ge 4\\ 1 & if\ 2.5 \le \text{Active CPU's cores} \le 4\end{cases}$$

Number of CPU cores

Rule 18 where CPU's utilization is *medium*, CPU's current frequency is *high* and number of active CPU's core is *high*.

TABLE 2. Membership functions of Fuzzy type-2 for output variables.

$$\mu_{A1}=\begin{cases}0 & if\ 0.35 \le \text{GPU's New Frequency} \le 0\\ \frac{0.35-x}{0.35} & if\ 0 \le \text{GPU's New Frequency} \le 0.35\end{cases}$$

$$\mu_{A2}=\begin{cases}0 & if\ 0.35 \le \text{GPU's New Frequency} \le 0\\ \frac{0.35-x}{0.4375} & if\ 0 \le \text{GPU's New Frequency} \le 0.35\end{cases}$$

$$\mu_{B1}=\begin{cases}0 & if\ 0.8 \le \text{GPU's New Frequency} \le 0\\ \frac{x-0.2}{0.3} & if\ 0.2 \le \text{GPU's New Frequency} \le 0.5\\ \frac{0.8-x}{0.3} & if\ 0.5 \le \text{GPU's New Frequency} \le 0.8\end{cases}$$

$$\mu_{B2}=\begin{cases}0 & if\ 0.8 \le \text{GPU's New Frequency} \le 0\\ \frac{x-0.2}{0.375} & if\ 0.2 \le \text{GPU's New Frequency} \le 0.5\\ \frac{0.8-x}{0.375} & if\ 0.5 \le \text{GPU's New Frequency} \le 0.8\end{cases}$$

$$\mu_{C1}=\begin{cases}0 & if\ 1.0 \le \text{GPU's New Frequency} \le 0.65\\ \frac{x-0.65}{0.35} & if\ 0.65 \le \text{GPU's New Frequency} \le 1.0\end{cases}$$

$$\mu_{C2}=\begin{cases}0 & if\ 1.0 \le \text{GPU's New Frequency} \le 0.65\\ \frac{x-0.65}{0.4375} & if\ 0.65 \le \text{GPU's New Frequency} \le 1.0\end{cases}$$

Normalized GPU's New Frequency

$$\mu_{A1}=\begin{cases}0 & if\ 0.35 \le \text{CPU's New Frequency} \le 0\\ \frac{0.35-x}{0.35} & if\ 0 \le \text{CPU's New Frequency} \le 0.35\end{cases}$$

$$\mu_{A2}=\begin{cases}0 & if\ 0.35 \le \text{CPU's New Frequency} \le 0\\ \frac{0.35-x}{0.4375} & if\ 0 \le \text{CPU's New Frequency} \le 0.35\end{cases}$$

$$\mu_{B1}=\begin{cases}0 & if\ 0.8 \le \text{CPU's New Frequency} \le 0\\ \frac{x-0.2}{0.3} & if\ 0.2 \le \text{CPU's New Frequency} \le 0.5\\ \frac{0.8-x}{0.3} & if\ 0.5 \le \text{CPU's New Frequency} \le 0.8\end{cases}$$

$$\mu_{B2}=\begin{cases}0 & if\ 0.8 \le \text{CPU's New Frequency} \le 0\\ \frac{x-0.2}{0.375} & if\ 0.2 \le \text{CPU's New Frequency} \le 0.5\\ \frac{0.8-x}{0.375} & if\ 0.5 \le \text{CPU's New Frequency} \le 0.8\end{cases}$$

$$\mu_{C1}=\begin{cases}0 & if\ 1.0 \le \text{CPU's New Frequency} \le 0.65\\ \frac{x-0.65}{0.35} & if\ 0.65 \le \text{CPU's New Frequency} \le 1.0\end{cases}$$

$$\mu_{C2}=\begin{cases}0 & if\ 1.0 \le \text{CPU's New Frequency} \le 0.65\\ \frac{x-0.65}{0.4375} & if\ 0.65 \le \text{CPU's New Frequency} \le 1.0\end{cases}$$

Normalized CPU's New Frequency

$$\mu_{A}=\begin{cases}0 & if\ \text{Active CPU's cores} \le 1\ or \ge 2\\ 1 & if\ 1 \le \text{Active CPU's cores} \le 2\end{cases}$$

$$\mu_{B}=\begin{cases}0 & if\ \text{Active CPU's cores} \le 1.5\ or \ge 3\\ 1 & if\ 1.5 \le \text{Active CPU's cores} \le 3\end{cases}$$

$$\mu_{C}=\begin{cases}0 & if\ \text{Active CPU's cores} \le 2.5\ or \ge 4\\ 1 & if\ 2.5 \le \text{Active CPU's cores} \le 4\end{cases}$$

Number of Active CPU Cores

In this case, the opportunity to reduce resources is greater as compared to previously discussed rules as both the CPU's current frequency and the number of active CPU's cores are high. However, by separately testing the impact of both DVFS and CG on the performance of the system, it was noticed that reducing the active number of CPU's core has a significant impact on increasing utilization with a minimal amount of performance loss. Hence, in this rule, the number of active CPU's core is reduced to low and current CPU's frequency remained the same. Rule 9 also appear to be the same case

TABLE 3. Rules of FBRMC-2 for CPU's optimization.

Rule Number	CPU Utilization	CPU's Current Frequency.	No. of Active CPU's Cores	CPU's New Frequency	No of. Active CPU's Cores
1	L	L	L	-	-
2	L	L	M	-	L
3	L	L	H	-	L
4	L	M	L	M	-
5	L	M	M	L	L
6	L	M	H	M	L
7	L	H	L	L	-
8	L	H	M	M	L
9	L	H	H	M	L
10	M	L	L	-	-
11	M	L	M	-	L
12	M	L	H	-	M
13	M	M	L	L	-
14	M	M	M	M	L
15	M	M	H	-	M
16	M	H	L	M	-
17	M	H	M	M	L
18	M	H	H	-	L
19	H	L	L	-	-
20	H	L	M	-	-
21	H	L	H	-	-
22	H	M	L	L	-
23	H	M	M	L	-
24	H	M	H	L	-
25	H	H	L	L	-
26	H	H	M	M	-
27	H	H	H	M	M

as both the CPU's current frequency and the number of active CPU's cores are high yet, it is different in terms of CPU's utilization as in this case its *low*. In this rule along with decreasing number of active CPU's core to low, CPU's current frequency has also been reduced to medium. This is because of the reason that CPU's utilization is significantly low and requires drastic steps for maximizing it while balancing the performance. Rules of FBRMC-2 for GPU's optimization was designed with a similar approach and given in Table. 4. FBRMC-2, keeping track of CPU utilization of all cores, overall GPU utilization, current CPU's frequency, current GPU's frequency, and the number of currently active CPU cores, strive to select the optimal frequency of both CPU & GPU and number of active cores for CPU only.

Centroid method, by calculating the center of gravity (COG) of the particular membership function area, defuzzify the output parameter. COG was calculated using the following equation and working of fuzzy logic type-2 for single input is explained in Fig. 2.

$$COG = \frac{\sum_{x=a}^{b} \mu_A(\chi)x}{\sum_{x=a}^{b} \mu_A(\chi)} \tag{1}$$

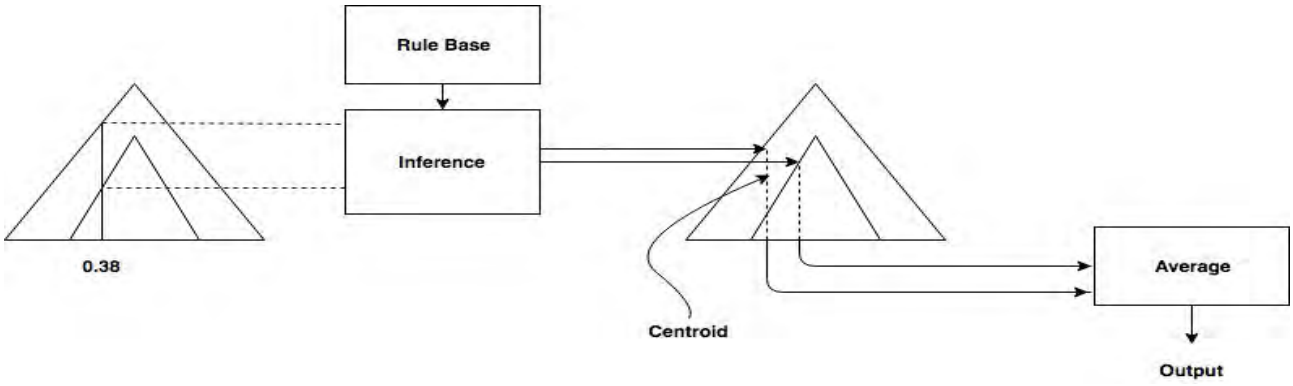


FIGURE 2. Working of Fuzzy logic type-2 for single input.

TABLE 4. Rules of FBRMC-2 for GPU's optimization.

Rule No	GPU's Current Frequency	GPU's Utilization	GPU's New Frequency
1	L	L	-
2	L	M	-
3	L	H	-
4	M	L	L
5	M	M	L
6	M	H	-
7	H	L	L
8	H	M	M
9	H	H	M

The whole process, incorporating the multi-thread approach, runs in parallel. Focal process create three threads on CPU cores using *OpenMP* as shown in Fig. 3, *Thread 1* executes the FBRMC-2 unit which further controls the DVFS and CG module. *Thread 2* executes the *Monitoring Unit 1* for tracking of utilization and current operating frequency of CPU & GPU, while *Thread 3* executes *Monitoring Unit 2* for tracking the active number of CPU cores. Each thread completes the program in a totally parallel approach without any dependencies; therefore, it does not produce any delay in the performance of the overall system. In order to evaluate the performance, standard CUDA based benchmark is executed on GPU in parallel with the execution of FBRMC-2 on CPU.

Fig. 4 represents a sample case study, in this case, CPU's utilization is 45%, normalized CPU's current frequency is 0.35 and CPU is running on 4 active cores. Hence, after reviewing the membership function for input parameters given in Table 1, it is clear that scenario under consideration, satisfies the Rule # 15 given in Table.3 and that is CPU's utilization is medium, normalized CPU's current frequency is medium and active number of CPU's core is high. That means resources are high; however, CPU's utilization can further be improved. This condition predicts that workload available to CPU is not distributed in parallel. Therefore, in order to maximize the utilization, which in return saves power consumption, only active number of cores is reduced from 4 to 2.85 and does not disturb operating frequency. It is important to note that the new number of active cores for CPU, proposed by FBRMC-2 is 2.85 which is not realistic, therefore, FBRMC also quantizes both proposed frequency and the active number of cores to available levels. In this case, 2.85 will be round-off to 3 cores.

Similarity, Fig. 4 also presents a sample case of resource management for GPU in terms of scaling frequency in a way, that should maximize the utilization. In the presented case, GPU utilization is 42% and normalized GPU current frequency is 0.42. This case satisfies the Rule # 5 given in Table.4 and that is GPU's utilization is medium and normalized GPU's current frequency is medium. FBRMC-2 after looking at GPU utilization will try to maximize it by reducing the frequency. Therefore, FBRMC-2 proposed normalized new frequency which is 0.245. Normalization of frequency range was achieved by dividing the actual range of frequency by maximum frequency of GPU valued 852 MHz. Hence, after getting the proposed normalized frequency, FBRMC-2 multiply this value to the maximum frequency ($0.245 \times 852 = 208.74$) and proposed frequency for GPU under this case is 208.74 MHz. Since only available GPU frequency can be implemented and proposed frequency lies between two available frequencies of GPU that are 180 MHz and 252 MHz; hence, FBRMC-2 then quantize the proposed frequency to nearest available frequency. In this case, FBRMC-2 quantized 208.75 MHz to 180 MHz. Frequency is allocated in a homogeneous way and then all core of GPU and CPU run on new GPU's frequency and new CPU's frequency respectively.

C. TARGETED TECHNIQUES FOR POWER OPTIMIZATION

Two well-known power optimization techniques were incorporated with FBRMC-2. These are DVFS and CG.

1) DVFS

Scaling frequency of computing unit produces a significant response in reducing power consumption. However, it is very

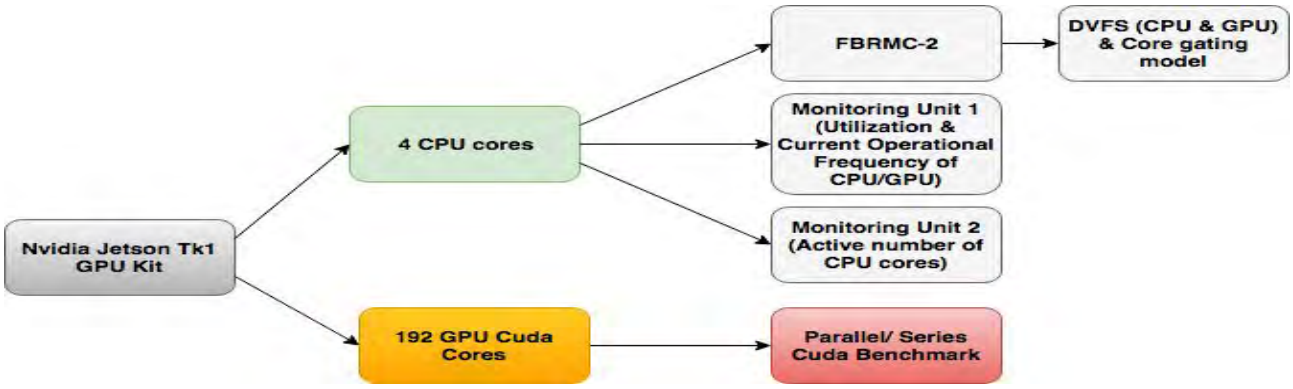


FIGURE 3. Block diagram of multi-threaded execution of proposed system.

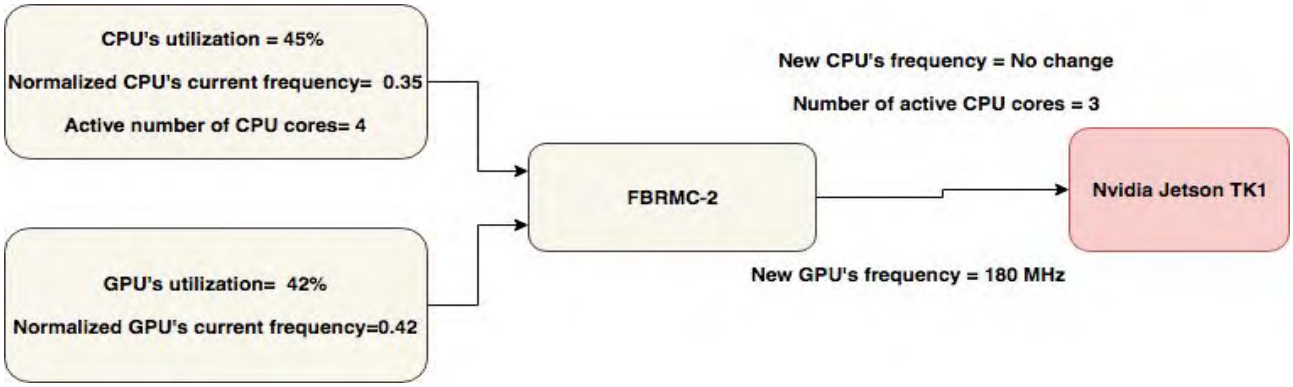


FIGURE 4. Sample description of FBRMC-2.

critical to decide the optimal frequency of the device otherwise, it may produce lag in performance. FBRMC-2 tries to select the most appropriate operating frequency of both CPU & GPU as explained in section III-B. It is important to discuss, FBRMC-2 proposed a separate frequency for CPU & GPU. However, the proposed frequency for CPU or GPU would be implemented homogeneously for all cores of CPU & GPU.

2) CG

FBRMC-2 use core gating technique to reduce the number of active core of CPU only. FBRMC-2 reduces the active core of CPU when either workload is less or parallelism available in workload is less. FBMRC-2 carefully proposed the number of active cores. CPU then runs on the proposed number of cores and rest of the cores are gated. Utilization of cores may vary from each other, therefore gating a core with high utilization may disturb the execution of task and performance of the overall system. Hence, FBRMC-2 along with proposing the active number of cores also suggest the identity of cores that need to be gated.

IV. EXPERIMENTAL RESULTS

The platform used to set-up the experiment was NVIDIA's embedded Tegra K1 SoC (CPU+GPU+ISP on a single processor) as shown in Fig. 5. An operating system used for

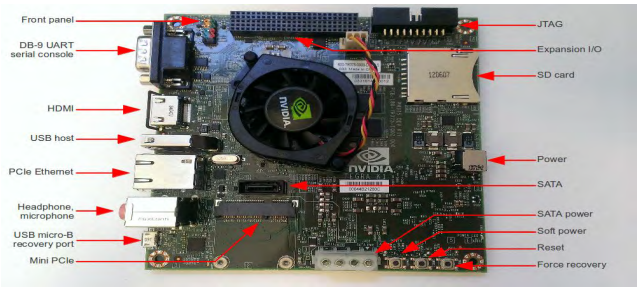
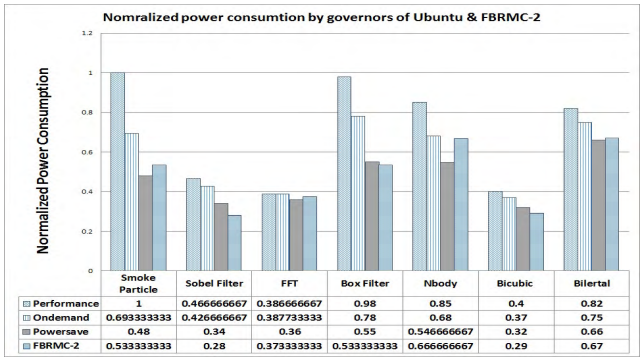


FIGURE 5. Labeled diagram of Nvidia Jetson TK1 GPU board.

this platform was Ubuntu, a Debian based Linux operating system.

In order to evaluate the results for the proposed scheme, various standard benchmarks were selected with different workload and amount of parallelism. For comparison, three well know governors of Ubuntu were used which are *powersave*, *ondemand* and *performance*. *Powersave* minimizes the resource configuration and designed to save power consumption only; while *ondemand* governor, similar to proposed FBRMC-2, tries to predict the workload and based on it, scale down the resources. On the other hand *Performance* governor tries to enhance the performance of the system by scaling up the resource configuration to the maximum



	Smoke Particle	Sobel Filter	FFT	Box Filter	Nbody	Bicubic	Bilertal
Performance	1	0.466666667	0.386666667	0.98	0.85	0.4	0.82
Ondemand	0.693333333	0.426666667	0.387733333	0.78	0.68	0.37	0.75
Powersave	0.48	0.34	0.36	0.55	0.546666667	0.32	0.66
FBRMC-2	0.533333333	0.28	0.373333333	0.533333333	0.666666667	0.29	0.67

FIGURE 6. Power consumption of governors of Ubuntu and FBMRC-2 against various kind of benchmarks.

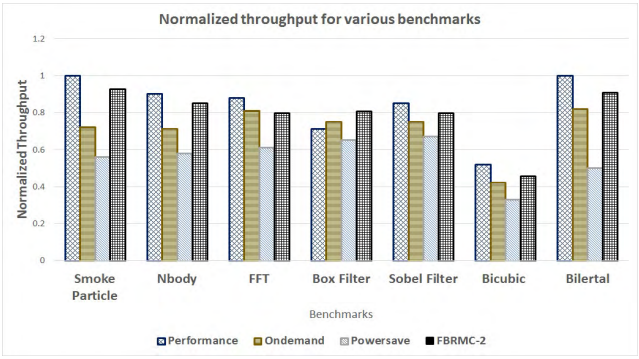


FIGURE 7. Normalized throughput of governors of Ubuntu and FBMRC-2 against various kind of benchmarks.

value and does not care about the power consumption of the device. Proposed FBRMC-2 on the other hand, has adopted a novel approach of using both DVFS (for CPU & GPU both) and CG (for CPU only) techniques together to reduce power consumption. Both power consumed by computing units and its throughput for various benchmarks has been discussed in results. It is important to note that power shown in Fig.6 is normalized power measured by Eq. 2. Here, AP is actual power consumed against said benchmark and MP is maximum power consumed by computing unit. In Fig.6, MP is power consumed by performance governor against *Smoke Particle* benchmark. Similarly, Normalized throughput was obtained by Eq. 3: At is actual throughput and MT is maximum amount of throughput. In Fig.7, MT is throughput of performance governor against *Smoke Particle* benchmark.

$$Nomralized\ Power = AP/MP \tag{2}$$

$$Nomralized\ Throughput = AT/MT \tag{3}$$

In this context, experimental results were evaluated using various benchmarks and results of power, consumed by computing unit, in case of FBRMC-2 and governors of Ubuntu, are shown in Fig.6. *Smoke Particle*, *Bilertal*, *Box Filter* and *NBody* are highly parallel benchmarks that deploy high level of parallelism in instruction level. Amount of workload in such cases appears high which make it difficult to save power and any attempt to save power would lead to a great amount of lag in performance. However, for these parallel benchmarks, FBRMC-2 has reduced the amount of power consumption significantly, without compromising on the throughput of the system to a greater extent as shown Fig.6. In comparison, on-demand & powersave governors showed fair reduction in power consumption in Fig.6. However, the throughput of the system, under these governors, appeared to be considerably low as compared to FBRMC-2 and performance governor shown in Fig.7. Performance governor, on the other hand, gives remarkable throughput shown in Fig.7, but in Fig.6, it is also evident, as performance governor only focus to provide better performance by maximizing the resource configuration, power-consumption for said governor is too high as compared to the proposed scheme. FBRMC-2 attempts to create a balance between power consumption and throughput and tries to minimize power consumption only in such scenarios which are feasible. In such cases, where better performance is required, FBMRC-2 does not attempt to save much power.

For detailed analysis of results, power-consumption and throughput of system for comparatively less parallel benchmarks, like *FFT*, *Sobel Filter*,*Bicubic*, and *Box Filter*, are also given in Fig.6 and Fig.7 respectively. In these benchmarks, the workload is not disseminated in a highly parallel manner hence; running the computing unit with high resource configuration would not be beneficial. Therefore; in such scenarios, FBRMC-2 with the novel approach of using DVFS along with CG saves more power than all above-mentioned governors as shown in Fig.6. Proposed scheme tries to maximize the utilization by reducing the frequency of both CPU & GPU along with reducing the active number of cores of CPU and outclasses the finest governor to reduce power consumption that is *powersave* governor. It is evident in Fig.6 that in case of FBRMC-2, the power consumed by the system is less than all above mentioned governors. On the other hand, due to the fact that workload in these benchmarks is not distributed in a highly parallel manner, improving utilization by reducing resource configuration did not compromise throughput of the system which is evident in Fig.7.

Similarly, consider case of two benchmarks *Histogram_ 64* and *Histogram_ 256*. Both benchmarks show the distribution of pixel intensities within an image. These benchmarks demonstrate a simple and high performance implementation of 64-bin histogram and 256-bin histogram respectively. Since Histogram_ 64 does not get affected from bank conflicts (load/store data from/to the same memory bank) and intra wrap branching divergence, performance in terms of processing time is better than Histogram_ 256 which highly depend on input data and that results in bank conflict and intra-wrap branching divergence. Histogram_ 64 being a lighter benchmark than Histogram_ 256, appear to require fewer resources which in results predict better opportunity for power saving by reducing the resources. However, due to bank conflict and intra-wrap branching divergence involve

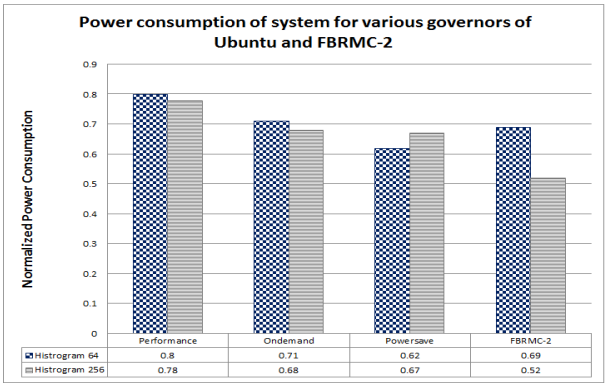


FIGURE 8. Normalized power consumption of governors of Ubuntu and FBMRC-2 for Histogram (64,256) benchmark.

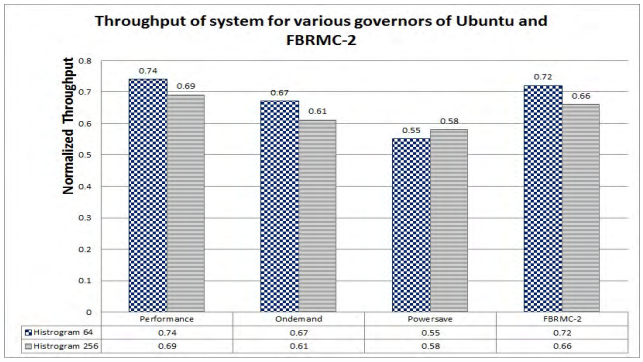


FIGURE 9. Normalized throughput of governors of Ubuntu and FBMRC-2 for Histogram (64,256) benchmark.

in Histogram_ 256, high amount of parallelism and resources were not producing a significant impact on processing time. Therefore, reducing resources in later case produce better results in reducing power consumption without disturbing the throughput to a greater extent. While in the case of Histogram_ 64, reducing resources specially scaling down the frequency of GPU was reducing a significant amount of throughput. Since FBRMC-2 manage resources based on utilization on CPU & GPU, which played a crucial role in estimating the above mentioned scenarios and reduces a significant amount of power without affecting the throughput. This is evident in result of power consumption and throughput for governors of Ubuntu and FBRMC-2 against both benchmarks as shown in Fig. 8 & 9 respectively.

Sometimes reducing power may lead to a lengthening of the task to such an extent that overall power consumption would still appear high. In such cases, reducing power consumption is not enough rather, it is also important to keep monitoring the effect of reduction in power consumption on the execution time of the task. To analyze the results of power optimization, Energy Delay Product (EDP) is an important metric. EDP portrays combine effect of throughput and energy consumption; hence, reducing EDP contain crucial importance for power efficient systems. Results of

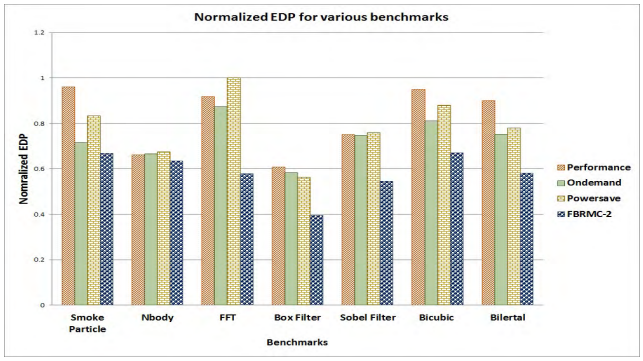


FIGURE 10. Normalized EDP of governors of Ubuntu and FBMRC-2 against various kind of benchmarks.

EDP for FBRMC-2 and governors of Ubuntu are shown in Fig.10 and it is evident that minimum EDP was achieved for FBRMC-2 as compared to available governors of Ubuntu. This is due to the fact that 47% power consumption was reduced by FBRMC-2 with less than 7 % of the reduction in throughput.

V. CONCLUSION

In this paper, FBRMC-2, featuring fuzzy type-2, has been proposed for finding the balance between the power consumption and the performance of HPC. Nvidia Jetson TK1 board was used for runtime implementation. A thorough literature survey is presented to highlight the issues affecting the performance and power consumption of GP-GPU based architectures. Then, in order to authenticate and resolve the effect of recognized challenging issues from literature, a fuzzy logic type-2 based reconfiguration engine, targeting to improve the balance between power and performance of HPC units, is presented. The proposed system by deactivating underutilized resources is capable of optimizing the resource configuration of CPU-GPU based architecture in a way that optimal resources would be allocated according to workload. Targeted techniques for resource management are DVFS (for both CPU and GPU) and CG (for CPU only) and FBRMC-2 intelligently incorporates these techniques to ensures the optimal power consumption. For various benchmarks, it is observed that proposed FBRMC-2 can save up to 47% of power consumption without affecting the performance and outrun all available governors of Ubuntu.

REFERENCES

[1] S. Collange, D. Defour, and A. Tisserand, "Power consumption of GPUs from a software perspective," in *Proc. Int. Conf. Comput. Sci.* Berlin, Germany: Springer, 2009, pp. 914–923.

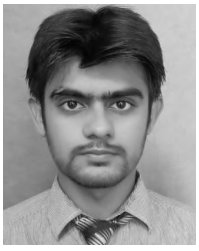
[2] M. Y. Qadri and K. D. McDonald-Maier, "A fuzzy logic reconfiguration engine for symmetric chip multiprocessors," in *Proc. Int. Conf. Complex, Intell. Softw. Intensive Syst. (CISIS)*, Feb. 2010, pp. 937–943.

[3] J. Ahmed, M Y. Siyal, S. Najam, and Z. Najam, "Challenges and issues in modern computer architectures," in *Fuzzy Logic Based Power-Efficient Real-Time Multi-Core System*. Singapore: Springer, 2017, pp. 23–29.

[4] I. Present, "Cramming more components onto integrated circuits," *Readings Comput. Archit.*, vol. 56, 2000.
[5] T. Mudge, "Power: A first class design constraint for future architectures," in *Proc. Int. Conf. High-Perform. Comput.* Berlin, Germany: Springer, 2000, pp. 215–224.
[6] E. H. Mamdani, "Application of fuzzy logic to approximate reasoning using linguistic synthesis," *IEEE Trans. Comput.*, vol. C-26, no. 12, pp. 1182–1191, Dec. 1997.
[7] P. Albertos and A. Sala, "Fuzzy logic controllers. Advantages and drawbacks," in *Proc. 8th Int. Congr. Autom. Control*, vol. 3, 1998, pp. 833–844.
[8] J. Ahmed, M. Y. Siyal, S. Najam, and Z. Najam, "Real-time power and performance-aware system," in *Fuzzy Logic Based Power-Efficient Real-Time Multi-Core System*. Singapore: Springer, 2017, pp. 31–45.
[9] X. Mei, Q. Wang, and X. Chu, "A survey and measurement study of GPU DVFS on energy conservation," *Digit. Commun. Netw.*, vol. 3, no. 2, pp. 89–100, 2017.
[10] W. Kim, M. S. Gupta, G.-Y. Wei, and D. Brooks, "System level analysis of fast, per-core DVFS using on-chip switching regulators," in *Proc. IEEE 14th Int. Symp. High Perform. Comput. Archit.*, Feb. 2008, pp. 123–134.
[11] Y. Liu, H. Yang, R. P. Dick, H. Wang, and L. Shang, "Thermal vs energy optimization for DVFS-enabled processors in embedded systems," in *Proc. 8th Int. Symp. Qual. Electron. Design (ISQED)*, Mar. 2007, pp. 204–209.
[12] J. Leverich, M. Monchiero, V. Talwar, P. Ranganathan, and C. Kozyrakis, "Power management of datacenter workloads using per-core power gating," *IEEE Comput. Archit. Lett.*, vol. 8, no. 2, pp. 48–51, Feb. 2009.
[13] R. Kumar, K. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen, "Processor power reduction via single-ISA heterogeneous multi-core architectures," *IEEE Comput. Archit. Lett.*, vol. 2, no. 1, p. 2, Jan./Dec. 2003.
[14] L. Cai and Y.-H. Lu, "Power reduction of multiple disks using dynamic cache resizing and speed control," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, Oct. 2006, pp. 186–190.
[15] S. Hong and H. Kim, "An integrated GPU power and performance model," *ACM SIGARCH Comput. Archit. News*, vol. 38, no. 3, pp. 280–289, 2010.
[16] Y. Jararweh, S. Alzubi, and S. Hariri, "An optimal multi-processor allocation algorithm for high performance GPU accelerators," in *Proc. IEEE Jordan Conf. Appl. Elect. Eng. Comput. Technol. (AEECT)*, Dec. 2011, pp. 1–6.
[17] R. Ge, R. Vogt, J. Majumder, A. Alam, M. Burtscher, and Z. Zong, "Effects of dynamic voltage and frequency scaling on a K20 GPU," in *Proc. 42nd Int. Conf. Parallel Process.*, Oct. 2013, pp. 826–833.
[18] D. You and K.-S. Chung, "Dynamic voltage and frequency scaling framework for low-power embedded GPUs," *Electron. Lett.*, vol. 48, no. 21, pp. 1333–1334, Oct. 2012.
[19] A. Lashgar, A. Baniasadi, and A. Khonsari, "Dynamic warp resizing: Analysis and benefits in high-performance SIMT," in *Proc. IEEE 30th Int. Conf. Comput. Design (ICCD)*, Sep./Oct. 2012, pp. 502–503.
[20] K. Danne and M. Platzner, "A heuristic approach to schedule periodic real-time tasks on reconfigurable hardware," in *Proc. Int. Conf. Field Program. Logic Appl.*, Aug. 2005, pp. 568–573.
[21] K. Danne, R. Muhlenbernd, and M. Platzner, "Executing hardware tasks on dynamically reconfigurable devices under real-time conditions," in *Proc. Int. Conf. Field Program. Logic Appl.*, Apr. 2006, pp. 1–6.
[22] K. Danne and M. Platzner, "An EDF schedulability test for periodic tasks on reconfigurable hardware devices," *ACM SIGPLAN Notices*, vol. 41, no. 7, pp. 93–102, 2006.
[23] P. Saha and T. El-Ghazawi, "Extending embedded computing scheduling algorithms for reconfigurable computing systems," in *Proc. 3rd Southern Conf. Program. Logic*, Feb. 2007, pp. 87–92.
[24] P. Saha and T. El-Ghazawi, "Software/hardware co-scheduling for reconfigurable computing systems," in *Proc. FCCM*, Apr. 2007, pp. 299–300.
[25] M. De Vuyst, R. Kumar, and D. M. Tullsen, "Exploiting unbalanced thread scheduling for energy and performance on a CMP of SMT processors," in *Proc. 20th IEEE Int. Parallel Distrib. Process. Symp.*, Apr. 2006, p. 10.
[26] K. Li, "Performance analysis of power-aware task scheduling algorithms on multiprocessor computers with dynamic voltage and speed," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 11, pp. 1484–1497, Nov. 2008.
[27] P. Yang *et al.*, "Energy-aware runtime scheduling for embedded-multiprocessor SOCs," *IEEE Design Test Comput.*, vol. 18, no. 5, pp. 46–58, Sep. 2001.
[28] A. Prayati *et al.*, "Task concurrency management experiment for power-efficient speed-up of embedded MPEG4 IM1 player," in *Proc. Int. Workshops Parallel Process.*, Aug. 2000, pp. 453–460.
[29] Z. Ma, C. Wong, P. Yang, J. Vounckx, and F. Catthoor, "Mapping the MPEG-4 visual texture decoder: A system-level design technique based on heterogeneous platforms," *IEEE Signal Process. Mag.*, vol. 22, no. 3, pp. 65–74, May 2005.
[30] N. Shaheryar, Y. Q. Muhammad, N. Zohaib, A. Jameel, and N. Q. Nadia, "A fuzzy logic based power-efficient run-time reconfigurable multicore system," *Chin. J. Electron.*, vol. 27, no. 3, pp. 549–555, 2018.
[31] N. N. Karnik, J. M. Mendel, and Q. Liang, "Type-2 fuzzy logic systems," *IEEE Trans. Fuzzy Syst.*, vol. 7, no. 6, pp. 643–658, Dec. 1999.
[32] B. Dietrich, S. Nunna, D. Goswami, S. Chakraborty, and M. Gries, "LMS-based low-complexity game workload prediction for DVFS," in *Proc. IEEE Int. Conf. Comput. Design (ICCD)*, Oct. 2010, pp. 417–424.
[33] B. Anita, "Dynamic power gating with quality guarantees," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Design*, vol. 27, no. 4, 2009, pp. 377–382.
[34] T. Komoda, S. Hayashi, T. Nakada, S. Miwa, and H. Nakamura, "Power capping of CPU-GPU heterogeneous systems through coordinating DVFS and task mapping," in *Proc. IEEE 31st Int. Conf. Comput. Design (ICCD)*, Oct. 2013, pp. 349–356.
[35] A. Pathania, Q. Jiao, A. Prakash, and T. Mitra, "Integrated CPU-GPU power management for 3D mobile games," in *Proc. 51st ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2014, pp. 1–6.
[36] C.-Y. Hsieh, J.-G. Park, N. Dutt, and S.-S. Lim, "Memory-aware cooperative CPU-GPU DVFS governor for mobile games," in *Proc. 13th IEEE Symp. Embedded Syst. Real-Time Multimedia (ESTIMedia)*, Oct. 2015, pp. 1–8.
[37] Y. Liang, P. Lai, and C. Chiou, "An energy conservation DVFS algorithm for the Android operating system," *J. Converg.*, vol. 1, p. 1, Dec. 2010.
[38] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose, "Microarchitectural techniques for power gating of execution units," in *Proc. Int. Symp. Low Power Electron. Design*, Aug. 2004, pp. 32–37.
[39] A. Pradip, "Dynamic power gating with quality guarantees," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Design*, 2009, pp. 377–382.
[40] L. Wu, X. Su, P. Shi, and J. Qiu, "Model approximation for discrete-time state-delay systems in the T–S fuzzy framework," *IEEE Trans. Fuzzy Syst.*, vol. 19, no. 2, pp. 366–378, Apr. 2011.
[41] X. Zhang, Z. Zhang, and G. Lu, "Fault detection for state-delay fuzzy systems subject to random communication delay," *Int. J. Innov. Comput., Inf. Control*, vol. 8, no. 4, pp. 2439–2451, 2012.

SHAHERYAR NAJAM was born in Pakistan, in 1990. He received the B.S. degree in electronics engineering from the National University of Science and Technology and the M.S. degree in electrical engineering from HITEC University. He is currently pursuing the Ph.D. degree in electrical engineering with the Department of Electrical Engineering, Riphah International University, Pakistan, where he is currently a Senior Lecturer. His current research interest includes power efficient high performance computing architectures.

JAMEEL AHMED received the B.E. degree in electronic engineering from the NED University of Engineering and Technology, Karachi, the M.S. degree in electrical engineering from National University of Science and Technology, and the Ph.D. degree from Pakistan and Nanyang Technological University (NTU), Singapore. Subsequently, he has carried out Post-doctorate Fellowship twice with NTU. He is actively involved in teaching and research for the last 25 years. He is currently a Professor and the Dean with the Faculty of Engineering and Applied Sciences, Riphah International University, Islamabad. He has published more than 50 national and international research publications. In addition, he has authored four international and one national book. He is a Member of NCRC and HEC, and an elected Member of the Governing body of Pakistan Engineering Council.



SAAD MASOOD received the bachelor's degree in electronics engineering from IQRA University, Karachi, Pakistan. He is currently pursuing the M.S. degree in electrical engineering with the Riphah International University, Islamabad, Pakistan. He is currently a Lab Engineer/Research and Development Officer with the Pakistan Institute of Engineering and Technology, Pakistan. His current research interest includes green computing.

CHUADRY MUJEEB AHMED is currently pursuing the Ph.D. degree with the Singapore University of Technology and Design. His research interests include cyber security, wireless and multimedia communications, and power efficient computing.