Journal of Theoretical and Applied Information Technology
15th December 2022. Vol.100. No 23
© 2022 Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A FRAMEWORK FOR SINGLE-CHIP MULTIPLE PROCESS MICROARCHITECTURE FOR DYNAMIC IOT COMMUNICATION

NITESH GAIKWAD[1], Dr. SHIYAMALA. S[2]

[1]Research Scholar, Electronics and Communication Engineering, Vel Tech Rangarajan

Dr. Sagunthala R&D Institute of Science and Technology.

[2]Professor, Electronics and Communication Engineering, Vel Tech Rangarajan

Dr. Sagunthala R&D Institute of Science and Technology

E-mail: niteshgaikwad78@gmail.com, drshiyamala@veltech.edu.in

ABSTRACT

The presence of high-performance multicore embedded architectures, energy efficiency remains a research black hole. Previous algorithms like Dynamic-based Frequency Scaling models and mapping-based thread models were implemented in microarchitecture to efficiently use clock frequency and energy. Unfortunately, these methods raise the problem of data traffic and high execution time. To overcome these drawbacks, a solution model analyzed load in the multiprocessor and migrated them to the proper Core to give an efficient data transfer rate. To perform this, an African Buffalo Load Migration Communication (ABLMC) system model has been presented in this paper. This unit analyses the workload characteristics of the multicore processor in the dynamic IoT environment application layer. Workload parameters are taken as features and computed for their similarity with minimum execution time using optimization. The predicted score value has been then given to the ABLMC model to take migrating decisions on loads. Communication requires a high data transfer rate that was achieved in a multicore processor using this ABLMC model. The proposed framework has been implemented and tested in a MATLAB environment with the performance matrices of high Data transfer rate and minimum execution time of 1.6 s. Thus, the proposed framework has excellence in real-time applications.

Keywords: *Microarchitecture, Multiprocessor, Core, Optimization, Iot Communication, And Workload Characteristics.*

1. INTRODUCTION

The internet has proliferated in consumer and enterprise markets [1]. The Internet of Things (IoT) develops an invisible, intelligent network fabric which can be sensed, programmed, and controlled [2]. The IoT allows the object in the ecosystem of IoT to communicate indirectly or directly with each other through the internet [3]. In the scope of IoT, the things are IoT-enabled objects which contain actuating and sensing elements along with embedded software and hardware components that facilitate network connectivity, security and data aggregation [4]. To perform a specific application task using gathered information, every IoT enabled things was designed [5]. The IoT objects massive deployment results in large volumes of information production [6]. In the IoT ecosystem, the data processing, communication, security, and real-time identification of such data's in the larger volumes are important problems that needed to be resolved for effective growth [7]. In the present IoT model, the end-devices of IoT are designed as cost-effective and simple as possible [8]. Thus, it was designed with limited processing abilities; clouds offload data and just enough for secure connection [9].

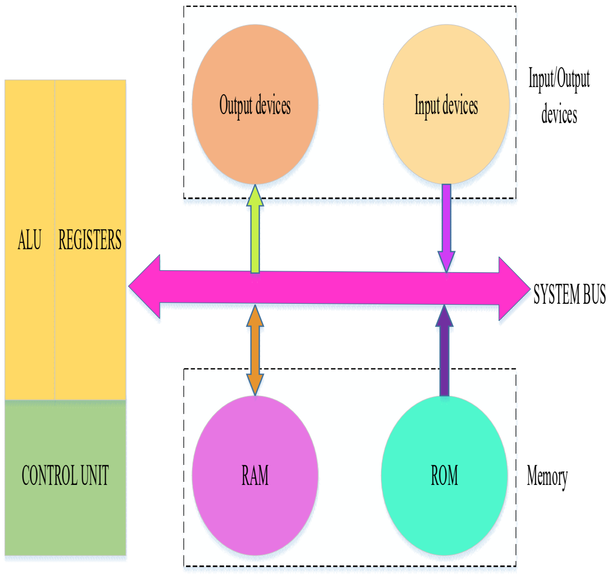

Fig.1 Block Diagram Of Microarchitecture

Moreover, the functionalities of complex information management like filtering of data and identification are delegated to the data centers in the cloud [10]. The processing units connected with IoT objects need an optimal balance among performance and power [11]. The microarchitecture block diagram is represented in Fig.1. Moreover, many of the IoT things were battery-powered; these things operate their entire life with battery. In modern electronics, the requirements of IoT were not satisfied by batteries and battery technologies [12]. So, the optimization of power was considered in the optimization of performance in parallel [13]. In IoT objects, designing effective embedded processors with optimized power performance is an unvaried process [14]. In the market, the challenge in the design was preventing the processors' high performance from the violation of requirements of the power budget [15].

The opportunities in processor design optimization for power were the greatest at the architecture level [16]. While defining the processor's microarchitecture configuration, the optimization of power and performance was done [17]. The microarchitecture configuration contains various design parameters of processors, which selected based on effects in overall performance and management of processor [18]. Selection of microarchitecture configuration includes various space explorations in design over a space of research for tunable design parameters for processors [19]. The determination of power was done by microarchitectural configuration in embedded IoT objects used in processors [20].

Several research works have been done in the past, such as level of microarchitectural configuration [21], dynamic pipeline-based PMDC prototype [22], intellectual completion-based tracking system [23], etc. But still, there is no significant improvement in the microarchitecture processors. These issues were motivated towards this research work.

The significant contributions of this research are as follows

- To propose an ABLMC unit that performs load migration for improved data transfer rate in dynamic IoT.
- A Framework of microarchitecture with ABLMC unit in the multicore processor for increased data transmission in IoT
- Analyzing workload parameters and extracting features from them to make a selection of minimum processing time using an optimization algorithm for the minimum response time of the proposed framework

By considering all these contributions, a proposed framework has been developed and so as detailed in this paper. This paper has been organized as follows: Section 2 explains the recent literatures based on microarchitecture, Section 3 explains the proposed methodology, and the simulated results and discussions are described in section 4. The conclusion of the paper is described in section 5.

2. RELATED WORK

Some of the recent literatures based on microarchitecture for IOT communication are described as follows:

In IoT applications, the two important design goals in a processor are high calculation performance and less power consumption. In one processor, achieving these two goals is challenging because of their high requirements. To solve this issue, Wei-Pau Kiat *et al.* [21] introduced a new technique i.e. reconfigurable microarchitectural level, which allows a RISC (Reduced Instruction Set computation) processor for supporting IoT applications with various energy trade-offs and performance requirements. The result demonstrates that the technique reduced the consumption of dynamic energy, and in IoT applications, it has a better energy tradeoff. However, it is not platform-specific, i.e., this device has no reconfiguration feature.

The PMDC prototype with dynamic pipeline stages was developed by Wasim Ghder Soliman *et al*. [22]. Using a reconfigurable computational system, the microarchitecture was implemented with the utilization of the FPGA board. In IoT integration procedure, for computing edge level and for optimal controller design, a PSO (Particle Swarm Optimization) is used. The result indicates that the developed method has better latency and throughput. Moreover, the connection needs motors in multiple numbers.

In network-on-chip (NoC), the number of processing elements was increased, and it demands higher throughput and lower latency with constraints in a chip. In NoC, the performance-hungry and fundamental blocks were Network Interface. To handle the multiple outstanding transactions, Sudeep P et al. [23] presented an intellectual completion-based tracking system. As a result, the transaction capability of multiple outstanding creates a cumulative impact on the NoC performance. Furthermore, improving the entire NoC's functionality, it requires more process.

The high-performance processors designed with less requirement of power is the major goal of many futuristic and contemporary applications. Satyajit Bora and Roy Paily [24] presented a novel microarchitecture processor; it has the capability to achieve that requirement. The benchmark of core mark was tested for the designed Core. The result demonstrated that the designed Core could outperform many other existing open-source and commercial cores. Moreover, the extension of the processor depends on specific applications.

Rahul Shrestha [25] has presented a decoding algorithm i.e. maximum a posteriori (MAP) which operates in multiple modes of the radix to hard decode. For MAP decoding-based algorithm, a new architecture for the digital decoder is designed. The presented algorithms' performance was analyzed in a white-Gaussian channel environment. As a result, the presented method has higher throughput and less latency. However, it cannot be utilized as an independent decoder channel.

3. PROPOSED METHODOLOGY

Due to their evident performance and cost advantages, Multiprocessor server processors were more prevalent in the embedded-based mobile systems and systems. With breakthroughs in microprocessor design, a device with processing cores tens has been accessible commercially, and the usage of feasible processing cores in larger numbers on a chip will expand rapidly. When these multicores are used in a dynamic IoT context, there is a lot of data flow and energy usage. To overcome these issues, an African Buffalo-based Load Migrating Chip (ABLMC) unit has also been presented in the microarchitecture to boost the communication acceleration ratio utilizing this multicore processor. The overall architecture of this proposed model ABLMC has been given in figure 2. This study presents an approach that incorporates input workload characteristics and core allocation based on workload. Workload optimization, extraction, and characterization utilizing ABO and prediction score calculations are parts of the proposed microchip unit.

3.1. Functional Units of Dynamic IoT

A dynamic IoT system comprises various functional blocks that help with identification, communication, management, actuation, and sensing, among other things. Devices in the system can exchange the information with other applications and connected devices, and then the collected information is processed locally and sent the information to the cloud-based application or centralized servers. In back-ends, the tasks are processed or performed locally with other IoT infrastructure, which is dependent on the spatial and temporal constraints. Many application components are captured in the functional units of dynamic IoT by using abstract architectural concerns and real deployment objects to capture technological decisions dependent on workload settings. Because there are frequently many Thread Processors (TP) available to realize a specific application component, ABLMC optimizes workload migration from one Core to another by predicting it in the architecture. Sensor I/O interfaces, audio/video interfaces, storage and memory interfaces, and Internet connectivity interfaces are just a few examples. The communication block was used to perform the communication among devices and distant servers. The data connection, network, transport, and application layers are all used in IoT communication protocols.

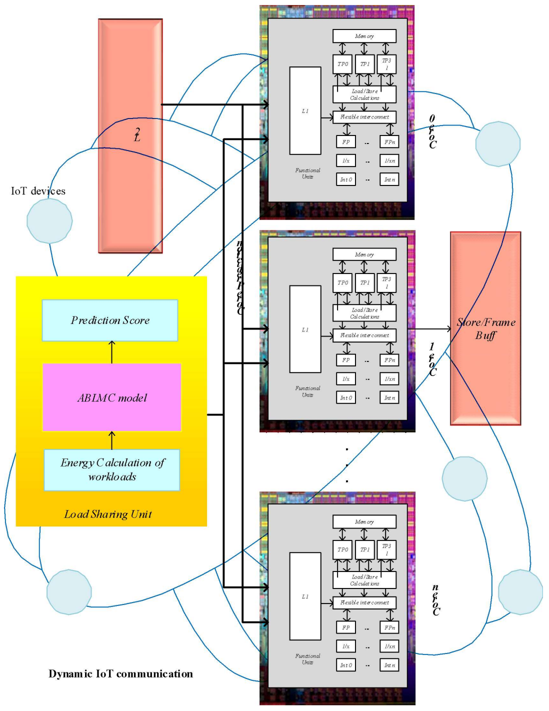

Fig.2 Overall Architecture Of Proposed Methodology And ABLMC Unit

The ABLMC model was created to address the issue of data traffic and energy consumption in multicore processors used in IoT communication. This has been treated as an optimization problem formulated by taking workload of size m and multicores as n numbers. Here, the problem is to find the Core that the load is to be migrated to and then analyse dynamic consumption of energy and execution time. Linear functions of load sizes can simulate energy consumption and time: one for loads lower than accessible core memory and the other for loads more significant than it.

3.2. Workload parameters extraction using Pin tools

Pin tools are used to measure workload parameters, and these tools have been adjusted to operate with several ARM-based multicore architectures. The pin tool is a binary instrumentation tool that, depending on the application, can be rewritable. The user can rebuild apps using the API supplied, operating as a source compatible with any architectural instruction sets. Pin tools automatically insert the function calls at any time in the program without impacting the application registers utilized in the application programs described by processors pin tools. The initial phase in the workload study was to identify inefficiencies in the various workloads and the commonalities between them. Pin avoids costs by storing previously saved compliant instrumented codes in a code buffer. Because the pin tools are designed for Intel-based designs, dealing with ARM architectures is a challenge. Some of the workloads considered in this study are given in the table.1

Table.1 Considered Work Load Features

Sl.no	Workload Features
1	Arithmetic Metric Measurement
2	Logical Measurement instructions
3	Load and store instruction
4	Read and write instruction
5	Branch instruction
6	ILP mechanisms
7	Register created traffics
8	System call Traffics
9	Branch predictable instructions
10	Unconditional Branch Instructions
11	Instruction per count
12	Cache access and miss ratio
13	Branch miss prediction ratio
14	Average power consumption ratio
15	Overall execution time of workload value

The workload parameters in that the microarchitecture are employed are identified by the user given labels. For the architecture-dependent workload parameters, independent workloads were used. With the help of these pin tools, the core speed and its execution time have been analysed. While a core executes at a minimum of zero speed value, there was no task to perform. Let $\vartheta_i = \frac{1}{\tau_i} = T_i/\bar{s}$ be the mean value of service rate measured. The average number of tasks that the core processor can complete has been in T_iTime unit. Utilization of Core for each task can be given by eqn. (1)

$$\rho_i = \frac{\varphi_i}{n_i\vartheta_i} = \frac{\varphi_i\bar{x}_i}{n_i} \tag{1}$$

Equation (1) shows the mean percentage of time with busy core information. Concerning the power for speed value, average energy consumed by the Core in one unit of time can be given using eqn. (2)

$$\rho_I {T_I}^{\tau} = \left(\frac{\varphi_i}{n_i}\right)\bar{r}{T_i}^{\tau-1} \tag{2}$$

In these equations (1) and (2), ρ_i is the Core of microarchitecture and φ_i be the power of speed and $\bar{x}_i$ represents the service rate. The current state and previous state of the multicore processor have been given as $\tau - 1$ and τ. Let the probability of task waiting time to get executed can be given by eqn. (3)

$$PB_{k,i} = \frac{\rho_i ni}{1-\rho_i} = \rho_{i,0}\frac{ni^{ni}}{ni!}\cdot\frac{{\rho_i}^{ni}}{1-\rho_i} \tag{3}$$

Similarly, the total number of tasks concerning their average is expressed in eqn. (4)

$$T_i = \sum_{k=0}^{\emptyset} O\rho_i n = n_i\rho_i + \frac{\rho_i}{1-\rho_i} \tag{4}$$

The execution time of the average tasks can be calculated using eqn. (5)

$$T_t = \frac{O_i}{\varphi_i} = \bar{x}_\iota\left(n_i\rho_i + \frac{\rho_i}{1-\rho_i}\right) \tag{5}$$

While writing the equation concerning multicore processor, equation (5) becomes eqn. (6)

$$T_t = \bar{x}_\iota\left(n_i\rho_i + \frac{\rho_i}{1-\rho_i}\cdot\frac{{\rho_i}^{ni}}{(1-\rho_i)^2}\right) \tag{6}$$

This equation is then simplified and it is expressed in eqn. (7)

$$T = \frac{\varphi_1}{\varphi}T_1 + \frac{\varphi_2}{\varphi}T_2 + \cdots + \frac{\varphi_n}{\varphi}T_n \tag{7}$$

This equation (7) has been treated as a function of load distribution. One of the first things we noticed when examining the various workloads was that they all had a single processor loop that took up most of the computing time.

3.3. ABLMC model in load sharing unit and its working

The user-defined labels identify the workload characteristics for which the microarchitecture is used. Independent workloads were used to test the architecture-specific workload characteristics. ABLMC unit has been presented in this paper to find the prediction score of each available core microarchitecture. Working load features listed in Table 1 have been considered as inputs for the optimization algorithm. The cores for which the load data has to be migrated will be selected by predicting the best quality and its respective workloads. Let us consider the prediction score value to be found out be ω, and T is the tasks execution time. The total functions executed in the Core can be given as $\varphi1, \varphi2, \ldots \varphi n$ in the core processor of $n1, n2, \ldots .n$. The fitness function for the evaluation of prediction score was calculated using eqn. (8)

$$f(x) = Minimize\ T \tag{8}$$

Where, T represents the Multicore processors average tasks execution time.

Step by step process for calculating and minimizing T can be given as

1 Consider input from workload features, average workload, and average execution time of workloads

2 Find the current state of each core (microarchitecture) processor and analyse the busy Core processor execution time

3 Calculate $t, \tau\ and \omega$

4 Now twice the ω value and for each time of iteration find whether the limit of waiting has exceeded the allowable limit or not.

5 The average value of the execution of the total workload in the core processor is then compared to the obtained execution time value.

6 The search of the Core that has minimum execution time is selected, and then the workload is migrated to that core processor

7 ABLMC model then collects this prediction core result and analysed for the selection of core processor in the microarchitecture

By implementing all these steps, the ABLMC unit solves the problem of data traffic and energy consumption.

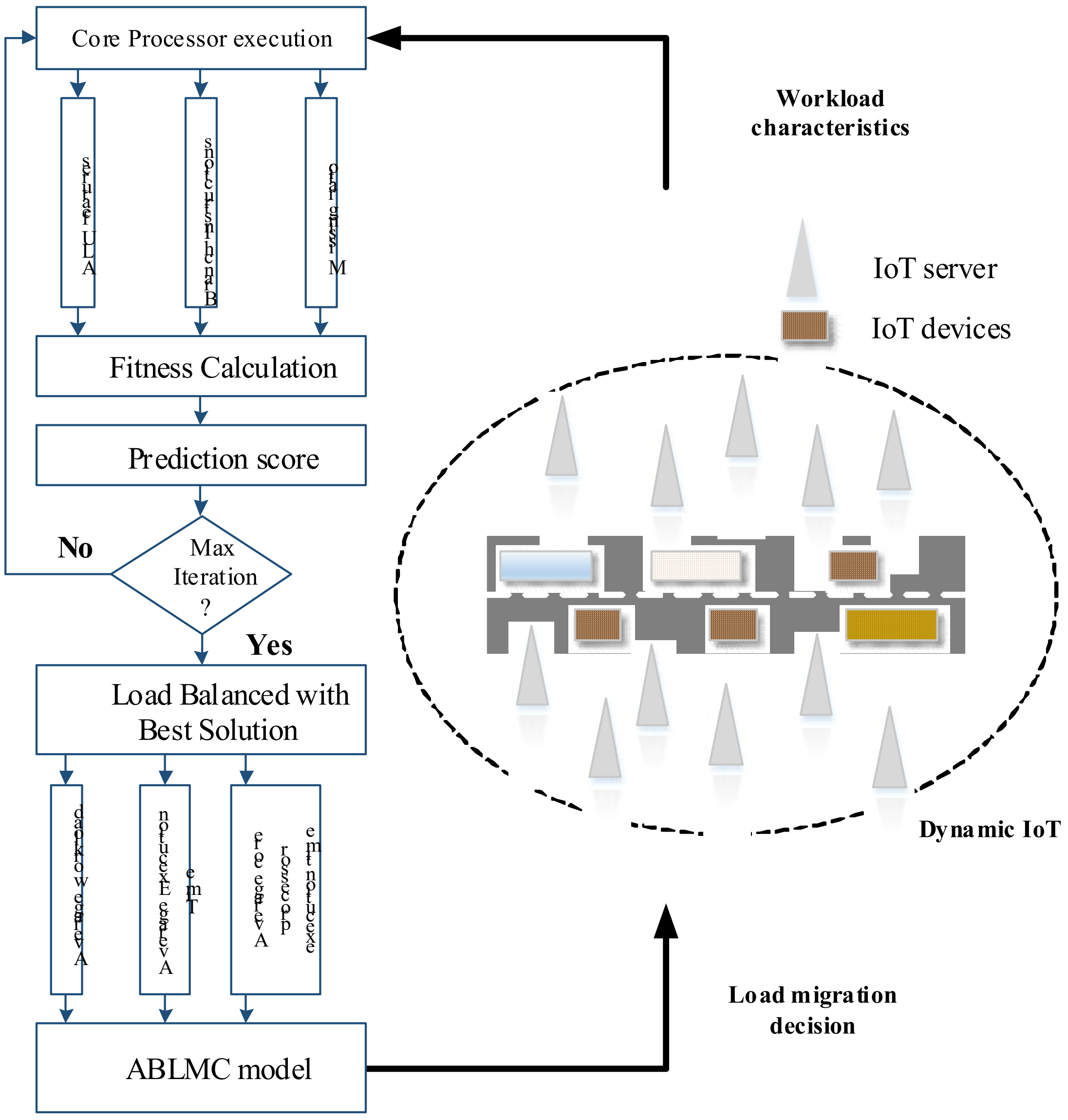

Fig. 3 Load Migration Using The ABLMC Model

A. Core processor workload update phase

As stated above, initially, all buffaloes have an equal amount of Core processor workload solution space. Based on the glow-worm position existing value, the workload of core processor is updated. When the updated workload phase starts, every glow-worm was added to the Core processor workload's previous level. Core processor workload fraction value is deducted to show the delay in Core processor workload value with time and it is expressed in eqn. (9).

$$l_i(t+1) = (1-\rho)l_i(t) + \gamma J(x_i(t+1)) \quad (9)$$

In this equation (9), $l_i(t)$ represents the Core processor workload level associated with glow worm i^{th} iteration at time t. ρ represents Core processor workload decay constant $(0 < \rho < 1)$. Core processor workload constant and $f(x(t))$denotes the value of an objective function at agents i^{th}location with time$t$. For each glow worm, the value of an objective function is computed using equation (9). Core processor workload gets updated based on intensity value of buffalo's position and expressed in eqn. (10).

$$I(dist) = I_0 \exp(-\gamma dist^2) \quad (10)$$

In equation (10), $dist$ is distance, γ is denoted as light absorption efficient $\in [0, \alpha]$ and I_0 is the light source intensity. The light intensity value is computed from the current position's objective function, and to achieve better value for the objective position, buffaloes move towards a better deal.

B. Selection of Core microarchitecture using minimized execution time Phase

Buffalo undergoes a movement phase where each buffalo utilize a probabilistic approach for decision making, and it moves towards the neighbor buffaloes to make the Core processor workload value is more significant than the present value. Buffaloes get attracted through the neighborhood buffaloes, which glows brighter as the Core processor workload gets updated based on the pixel's intensity value. Hence specific buffaloes might move towards brighter buffaloes B, probability of i moving towards neighbor B can be given as shown in equation (11)

$$Prob_{ij}(t) = \frac{l_B(t) - l_i(t)}{\sum_{A \in G(t)} l_A(t) - l_i(t)} \quad (11)$$

In equation (11), $B \in G(t)$, $G(t) = \{B: dist_{iB}(t) < range^i_{dist}(t); I_i(t) < I_B(t)\}$defined set of any buffaloes neighbor i with time t, $dist_{iB}(t)$ represents the Euclidean distance among buffaloes i and B, $range^i_d(t)$ represents the range of variable neighborhoods associated with buffaloes$i$ at time t. If the buffaloes$i$ choose another buffalo $B \in G(t)$ with $prob_{iB}(t)$. The movement of buffaloes has been modeled using a discrete-time model that can be stated as shown in equation (11).

$$y_i(t+1) = y_i(t) + S\left(\frac{y_B(t) - y_i(t)}{\|y_B(t) - y_i(t)\|}\right) \quad (12)$$

In equation (12), $y_i(t) \in \mathbb{R}^n$ represents the buffalo's location$i$, in the n-dimensional space $\mathbb{R}^n$, $\|.\|$ represents the Euclidean norm operator and S denote step size.

Algorithm 1

```
Create Dynamic IoT network
//*Perform ABLMC algorithm technique for getting prediction score*//
Declare an integer variable $y and y_{max}, y_1, y_2, y_3$
Initialize y to zero
Initialize $ymax$ to hundred
Initialize $y1, y2 and y3$
//* Int T function*//
Compute $S_{(q,j)}$
Using eqn (1)
//Objective function//
If
Load distribution function, using eqn (7)
{
Core processor is selected
        Else
        {
Update
T value as new
}
}
End if
// Updating //
T function
Int $T$
for
Microprocessors avg. execution time (8)
End for
//* Fitness function termination*//
Repeat
Objective function $\longrightarrow$ n times
Set limit $\longrightarrow$ n times
Terminate
if
limit $\Longrightarrow$ exceeded
Use eqn (8)
End loop
}
```

Multiprocessor server processors have grown more common in mobile embedded systems and systems due to their cost advantages and apparent performance. A device with processing cores tens has been available commercially thanks to improvements in microprocessor design, and the possible processing cores on a chip will expand rapidly. There is a lot of data flow and energy utilization when these multicores are used in a dynamic IoT setting. To address this challenge, the microarchitecture includes an African Buffalo-based Load Migrating Chip (ABLMC) unit that boosts the communication acceleration ratio while using this multicore CPU.

4. Results

The proposed ABLMC unit has been employed along with the Microarchitecture of Multicore processor; this model has been simulated and analysed for their performance. This model has been implemented in the MATLAB2020 tool in the windows 10 platform. The proposed model has been tested with a self-adaptable data partitioning algorithm solving bi-objective optimization problem for performance and energy (ADAPTALEPH) [26], feedback-driven priority-based CPU scheduling algorithm (FDPBCSA) [27], workload dependent dynamic power management model (WDDPMM) [28] and energy-efficient and dynamic frequency scaling for multi-core embedded architectures in an IoT environment (PODS) [29]. Performance parameters include Execution Time, Data transfer rate, and Power consumption.

4.1. Dataset

Various workloads are derived from and illustrated in fig 3, several sensors like cameras, and from the experimental setup, the image processing approaches are gathered. The suggested ABLMC framework was trained using around 1600 datasets. We used IoT benchmarks to test and evaluate the proposed approach, which covers applications in a wide range like arithmetic and logic functions, branch instructions, and missing ratios.

4.2 Analysis of ABLMC unit in multiprocessor for Chicago dataset

Dataset has been considered to be executed in the IoT communication model made up of Multicore processor that employs an ABLMC unit. Dataset has been obtained from [26] that has the count of taxi trips that occurred in Chicago city. It was collected as sensor data and communicated via IoT devices to the system for execution. Collection of this data and performing functions at the application layer of IoT and causes data traffic as there is more data to be handled. To overcome this proposed framework, has an ABLMC unit within them that performs the selection of minimum execution time of core microarchitecture by analysing average tasks and average workloads. By this prediction score, core is predicted so as to perform the execution.

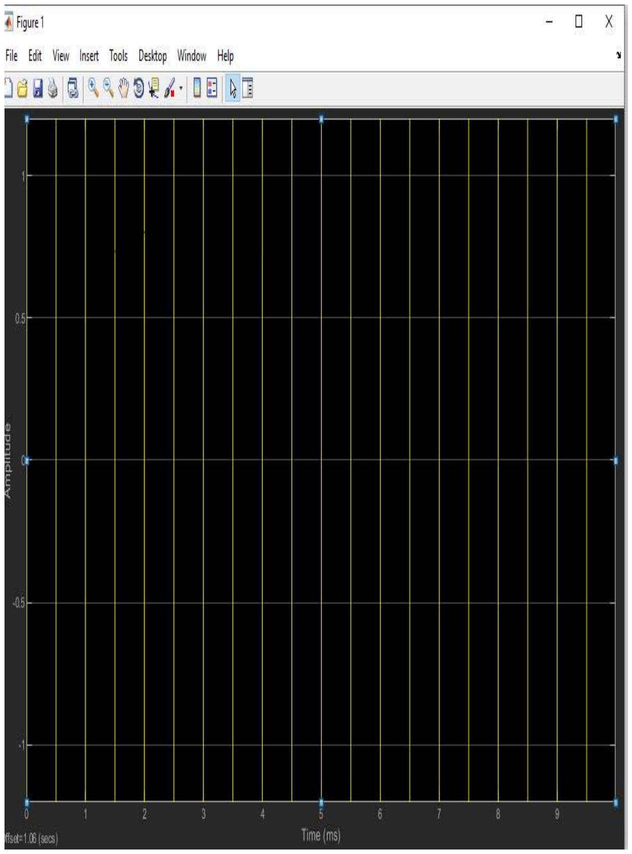

Fig.4 Error rate

Error rate was the missed communication between the IoT device and the system. This error rate is shown in figure 4. The data transfer rate of the processor that used the ABLMC unit is shown in figure 5. From this figure, it is clearly visible that the processor takes extremely minimum time for running waiting and preempted. This data processing unit has shown that it has acquired minimum memory consumption and power consumption properties.

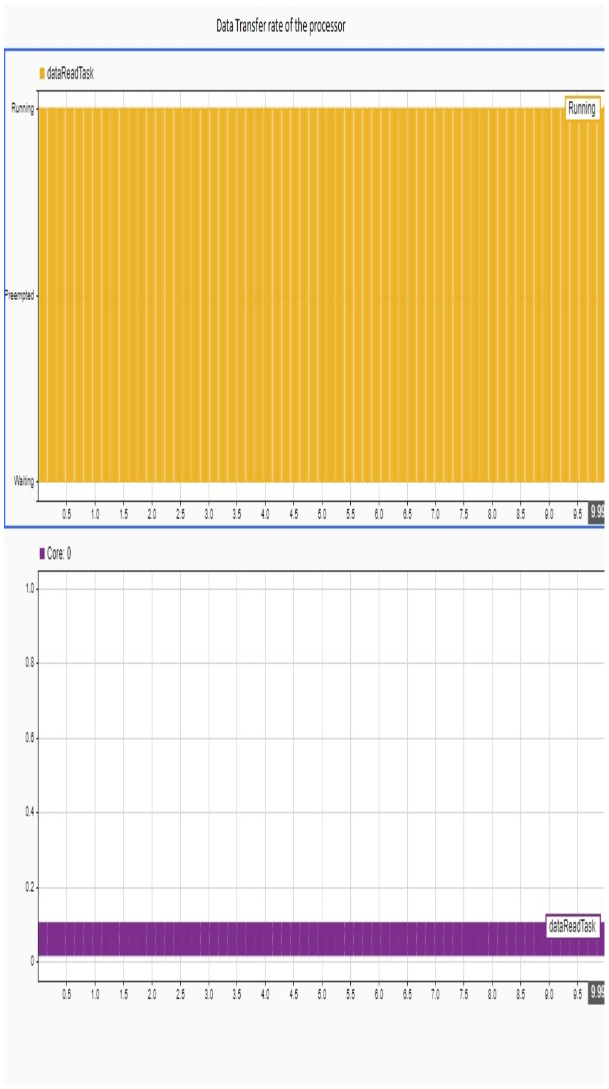

Fig.5 Data Transfer Rate Of The Processor

While executing read and writes instructions in the IoT communication, the data error rate of the processor is presented in this figure 6. Error rate for executing these instructions as branch instructions and arithmetic logic instructions was measured by acquiring memory values from the processor. This error rate measurement shows that the proposed framework has high data transfer rate and minimum error rate from the graph given below.

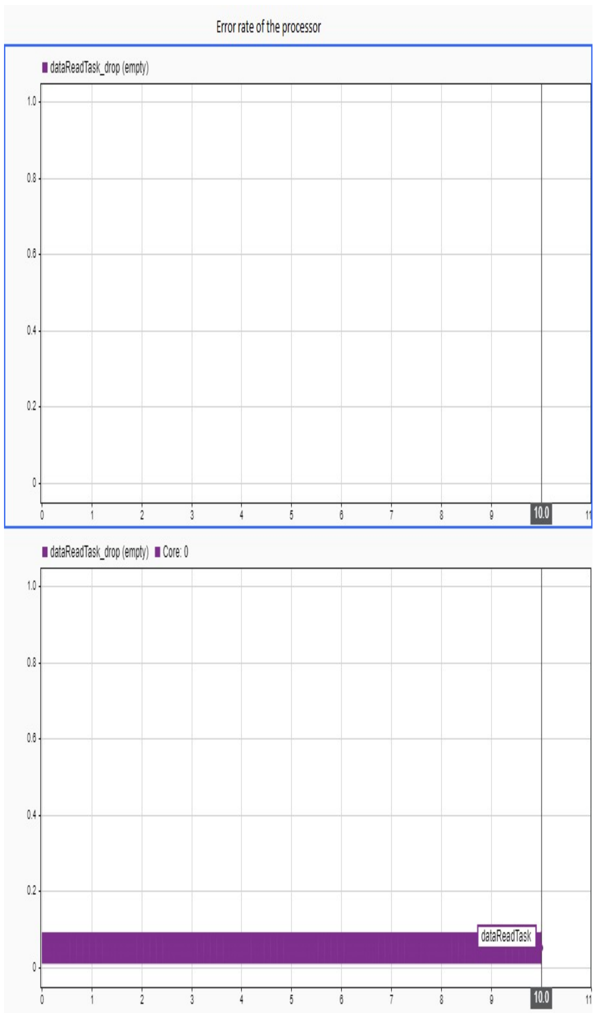

Fig.6 Error Rate Of The Processor

4.3. Comparison graphs of proposed with previous techniques

By comparing the parameters like processing time, data transfer rate, and power consumption with the previous techniques like ADAPTALEPH, FDPBCSA, WDDPMM, and PODS. Comparison graphs have been given in the below section.

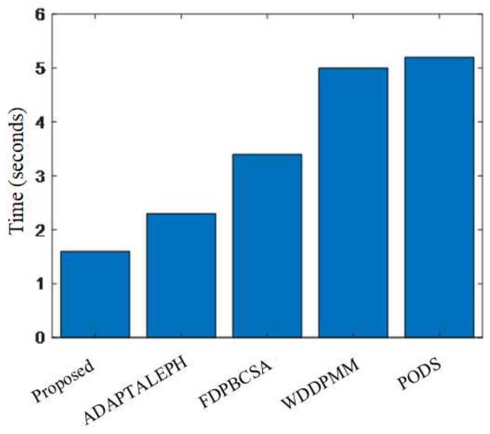

Fig.7 Processing Time Comparison

The time of processing is described as the time taken between the two processes i.e. the interpreting data stream and incoming event stream. The unit of processing time is second. Processing time is the time taken for the whole model to be executed and produce the result. From the graph as shown in figure 7, it has been stated that the proposed model executes the whole process in low time that is, at a time period of 2.5 seconds, other techniques such as ADAPTALEPH and FDPBCSA takes a processing time of 2.8 seconds and 3.2 s respectively, WDDPMM and PODS scheme takes a time of 6.1 seconds and 7.5 seconds. A minimized processing time was obtained for the proposed framework as it was using a highly relied ABLMC unit that gives quick results at the application end.

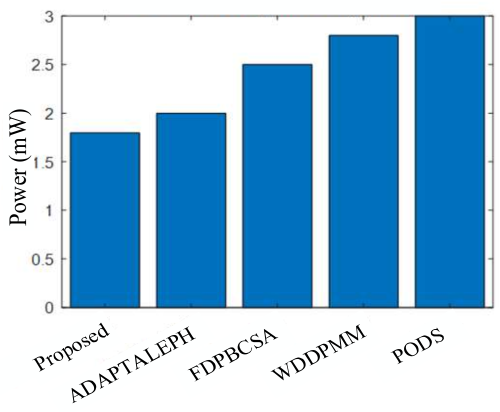

Fig.8 Power Consumption Comparison

Power consumption is the analysis of power consumed by the proposed and other frameworks while executing the process in the multicore processor. The proposed framework has consumed less power with minimum time. From figure 8 it can be clearly visible that the power has been in the value of 1.7mw for a proposed framework that is minimum power consumption compared to other techniques. While ADAPTALEPH, FDPBCSA, WDDPMM, and PODS have given power consumption of 2mw,2.5mw, 2.8mw, and 3mw, respectively. Thus, the proposed framework has been secured the minimum value of power consumption.

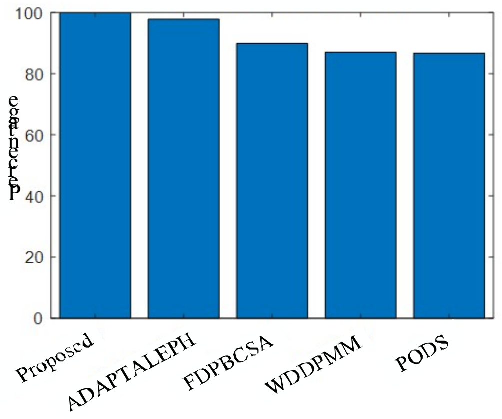

Fig.9 Data Transfer Rate Comparison

From figure 9, it can be clearly understood that the data transfer rate has been measured and plotted in the graph with respect to the communication instructions. ADAPTALEPH, FDPBCSA, WDDPMM and PODS has shown a decreased data transfer rate of 96%,90%,83% and 80% respectively. Rather proposed framework has shown a high data transfer rate of 99.9%, this was due to the ABLMC unit included in the multicore processor.

5. CONCLUSION

This study presents a model of the African Buffalo Load Migration Communication (ABLMC) system. This unit examines the multicore processor's workload characteristics in the dynamic IoT environment application layer. Workload parameters are used as features, and their similarity with the shortest execution time is calculated using optimization. The ABLMC model was then given the projected score value to make load migration decisions. Communication necessitates a fast data transfer rate, which this ABLMC architecture enabled in a multicore CPU. The suggested framework was constructed and tested in a MATLAB environment using performance matrices that required a high data transmission rate and a minimal execution time of 1.6 seconds. As a result, the suggested framework performs exceptionally well in real-time applications. Hence the energy consumption was reduced by

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

implementing the intelligent calculations in frequency scaling, which identifies the way of working in an IoT environment. Further, to attain better performances, the intelligent algorithms combined with Kernel, which intelligently schedule the workload.

DECLARATION OF CONFLICTING INTERESTS:

The Author(s) declared no potential conflicts of interest with respect to the research, authorship and/or publication of this article.

REFERENCES

[1] Zollo, Lamberto, et al. "What influences consumers' intention to purchase organic personal care products? The role of social reassurance." Journal of Retailing and Consumer Services 60 (2021): 102432.
[2] Nakandhrakumar, R. S., et al. "Internet of Things (IoT) based system development for robotic waste segregation management." Materials Today: Proceedings (2021).
[3] Silvano, Wellington Fernandes, and Roderval Marcelino. "Iota Tangle: A cryptocurrency to communicate Internet-of-Things data." Future Generation Computer Systems 112 (2020): 307-319.
[4] Tuysuz, Mehmet Fatih, and Ramona Trestian. "From serendipity to sustainable green IoT: Technical, industrial and political perspective." Computer Networks 182 (2020): 107469.
[5] Sharma, Avani, et al. "Towards trustworthy Internet of Things: A survey on Trust Management applications and schemes." Computer Communications (2020).
[6] Zhao, Zhiheng, et al. "IoT edge computing-enabled collaborative tracking system for manufacturing resources in industrial park." Advanced Engineering Informatics 43 (2020): 101044.
[7] Li, Yuxi, et al. "Deep learning in security of internet of things." IEEE Internet of Things Journal (2021).
[8] Zahmatkesh, Hadi, and Fadi Al-Turjman. "Fog computing for sustainable smart cities in the IoT era: Caching techniques and enabling technologies-an overview." Sustainable Cities and Society 59 (2020): 102139.
[9] Wang, Haoxin, et al. "Architectural design alternatives based on cloud/edge/fog computing for connected vehicles." IEEE Communications Surveys & Tutorials 22.4 (2020): 2349-2377.
[10] Khashan, Osama A. "Parallel Proxy Re-Encryption Workload Distribution for Efficient Big Data Sharing in Cloud Computing." 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC). IEEE, 2021.
[11] RM, Swarna Priya, et al. "Load balancing of energy cloud using wind driven and firefly algorithms in internet of everything." Journal of parallel and distributed computing 142 (2020): 16-26.
[12] Garrido-Hidalgo, Celia, et al. "The adoption of internet of things in a circular supply chain framework for the recovery of WEEE: The case of lithium-ion electric vehicle battery packs." Waste Management 103 (2020): 32-44.
[13] Song, Yingjie, et al. "MPPCEDE: multi-population parallel co-evolutionary differential evolution for parameter optimization." Energy Conversion and Management 228 (2021): 113661.
[14] Iqbal, Waseem, et al. "An in-depth analysis of IoT security requirements, challenges, and their countermeasures via software-defined security." IEEE Internet of Things Journal 7.10 (2020): 10250-10276.
[15] Ainsworth, Sam, et al. "ParaDox: Eliminating Voltage Margins via Heterogeneous Fault Tolerance." 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA). IEEE, 2021.
[16] Simevski, Aleksandar, et al. "PISA: Power-robust multiprocessor design for space applications." 2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS). IEEE, 2020.
[17] Nejat, Mehrzad, et al. "Coordinated management of processor configuration and cache partitioning to optimize energy under QoS constraints." 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS). IEEE, 2020.
[18] Shafiabadi, Mohammad Hossein, et al. "Comprehensive regression-based model to predict performance of general-purpose graphics processing unit." Cluster Computing 23.2 (2020): 1505-1516.
[19] Khailany, Brucek, et al. "Accelerating chip design with machine learning." IEEE Micro 40.6 (2020): 23-32.
[20] Wright, John Charles, et al. "A Dual-Core RISC-V Vector Processor With On-Chip Fine-Grain Power Management in 28-nm FD-SOI." IEEE Transactions on Very Large Scale

Integration (VLSI) Systems 28.12 (2020): 2721-2725.

[21] Kiat, Wei-Pau, et al. "An energy efficient FPGA partial reconfiguration based micro-architectural technique for IoT applications." Microprocessors and Microsystems 73 (2020): 102966.

[22] Soliman, Wasim Ghder, et al. "Reconfigurable Microarchitecture-Based PMDC Prototype Development for IoT Edge Computing Utilization." IEEE Sensors Journal 21.2 (2020): 2334-2345.

[23] Sudeep, P., et al. "Intellectual Completion Tracking System Micro-architecture for Advanced NoC Designs." 2020 IEEE VLSI DEVICE CIRCUIT AND SYSTEM (VLSI DCS). IEEE, 2020.

[24] Bora, Satyajit, and Roy Paily. "A High-Performance Core Micro-Architecture Based on RISC-V ISA for Low Power Applications." IEEE Transactions on Circuits and Systems II: Express Briefs 68.6 (2020): 2132-2136.

[25] Shrestha, Rahul. "A Multiple-Radix MAP-Decoder Microarchitecture and Its ASIC Implementation for Energy-Efficient and Variable-Throughput Applications." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 29.1 (2020): 65-75.

[26] Reddy Manumachu, Ravi, and Alexey L. Lastovetsky. "Design of self-adaptable data parallel applications on multicore clusters automatically optimized for performance and energy through load distribution." Concurrency and Computation: Practice and Experience 31.4 (2019): e4958.

[27] Sharma, Rashmi, et al. "Priority-based joint edf–rm scheduling algorithm for individual real-time task on distributed systems." The Journal of Supercomputing 77.1 (2021): 890-908.

[28] Chawla, Nikhil, et al. "Securing IoT devices using dynamic power management: machine learning approach." IEEE Internet of Things Journal (2020).

[29] Tamilselvan, K., and P. Thangaraj. "Pods–A novel intelligent energy efficient and dynamic frequency scalings for multi-core embedded architectures in an IoT environment." Microprocessors and Microsystems 72 (2020): 102907.