Efficient Power Management Technique Of Multicore Architecture For Real Time Visual Data

Alan Janson, Vinayak Bhogan, Akash Name, Julie Dsouza

Abstract— With increasing need of higher computational capacity and speed for visual data such as images & videos, Multicore computing technology has become a viable solution. Graphite multicore architecture simulator provides a necessary parallel computing environment not only for computational test but also optimized modelling. This paper discusses about the efficient model for multi-core architecture based on image processing algorithm incorporating both Dynamic Voltage Frequency Scaling (DVFS) & concept of heterogeneity.

Index Terms— Graphite Multicore Processor, Dynamic Voltage & Frequency Scaling [DVFS], Heterogeneity, Multithreading, Image Averaging Filter, Parallel Computation, Power Management.

1. INTRODUCTION

Images & video data is core of visual multimedia. Being considerably large data, higher computational speed is a crucial factor. Fast processing response is the major requirement in many real-time processing applications. Processors have become more powerful, processing has shifted towards digital domain, although parallel computing seems to be an ideal solution but actual hardware has much more complex factors like Power, area & memory resources are difficult to realize and implement. This basic issue of implementation of parallel computing algorithms can be handled effectively by Multicore architecture.

Multicore processors have a higher edge on single core processors in terms of various parameters like processing speed and computational time. The increase number of counts achieves the reasonable performance of the programming models for the need of multi-thread working in union with multicore. But as we increase the number of cores the power for each core also increases resulting in large power utilization. Power consumption is one of the major issues that needs to be resolved for efficient implementation of multicore architecture. With increasing performance minimizing the power consumption is crucial. Thus on architectural basis, we propose a new model based on both DVFS & heterogeneity which can be incorporated to reduce power consumption considerably while still maintaining overall performance drop under tolerable limits[3][4][6].

- *Alan Janson, EXTC Engineer from Mumbai University, India*
  E-mail: alanjannson@gmail.com
- *Vinayak Bhogan, EXTC Engineer from Mumbai University, India*
  E-mail: vinayakfrom25593@gmail.com
- *Akash Name, EXTC Engineer from Mumbai University, India*
  E-mail: akash.name07@gmail.com
- *Julie Dsouza, EXTC Engineer from Mumbai University, India*
  E-mail: julie.dsouza19@gmail.com

2. ARCHITECTURE AND BACKGROUND

2.1 Conventional Multicore Design

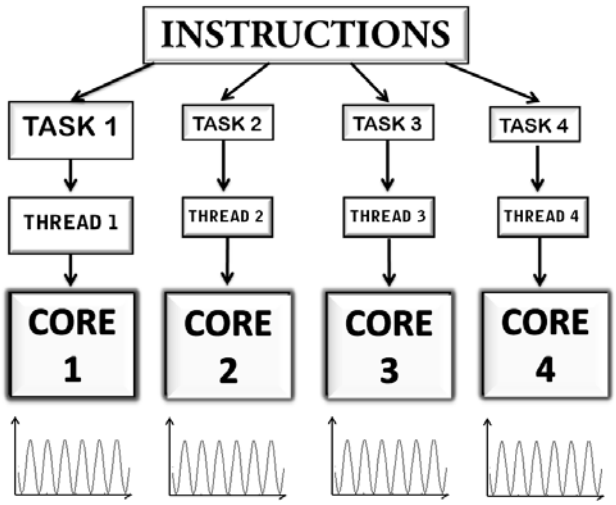

Fig. 1 Conventional Multicore Architecture.

This Fig. 1 shows the basic conventional multicore model. It contains homogeneous core which are same in size and configuration. This homogeneous core processes data irrespective of their variable complexity with similar computational processing. The supply voltage and operating frequency are also similar throughout the core. As the number of core increases the amount of power also increase resulting in large power consumption and heat dissipation[9][10].

Different schemes to reduce power are:

1) Dynamic Voltage & Frequency Scaling [DVFS]

DVFS is one of the common power management techniques. Complex workload need high speed and supply voltage for computation compared to simple

instruction. The overall power [P] mainly depends on supply voltage [V] and operating frequency [F] which is given as $P = CV^2F$. Therefore by scaling voltage and frequency with respect to workload, power can be reduced.

Multicore offers individual DVFS for each core, this is referred to as per-core DVFS (PC-DVFS). Per-core DVFS has extended flexibility in controlling power. Power has its own primary performance problem in high-performance computing (HPC). Thus reducing power consumption, lead to significant reduction in the energy required for a computation, particularly for memory-bound workloads[8].

2) Heterogeneity

Heterogeneity is the technique in which the configurations of the processing cores are modified based on the task incorporated. Increase in core size results in large power dissipation. Therefore using heterogeneous core, the core configuration are modified according to the given complex or simple task resulting in comparatively low power consumption.

Heterogeneous multi-core processor architecture can expand the features for powerful computing to provide system acceleration. They currently offer an efficient and powerful integrated processor moreover it also supports data parallelism, bringing several innovative changes in functionality of the system[2][7][5].

2.2 Modified Design

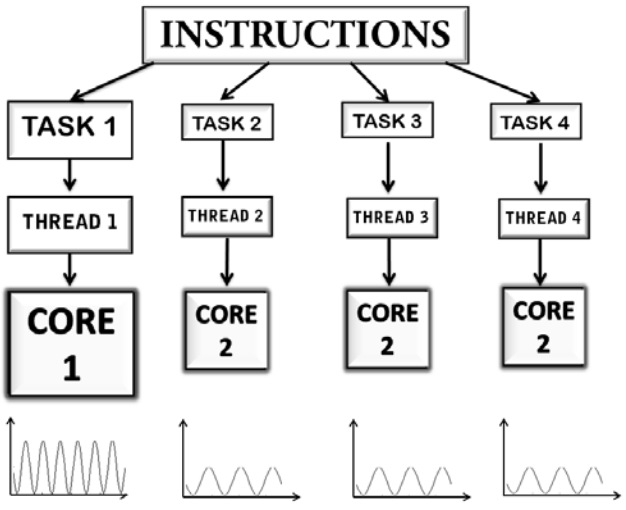

Fig. 2 Modified Architecture.

This Fig. 2 illustrates the modified model using DVFS and heterogeneity technique together for the efficient power management of the processing unit. Parameters like size of the processing core, supply voltage and operating frequency are made flexible in order to minimize power dissipation compared with conventional multicore design. In order to implement this modified multicore design, graphite multicore architecture is reliable and efficient for implementation.

3. IMPLEMENTATION

3.1 Simulation Environment

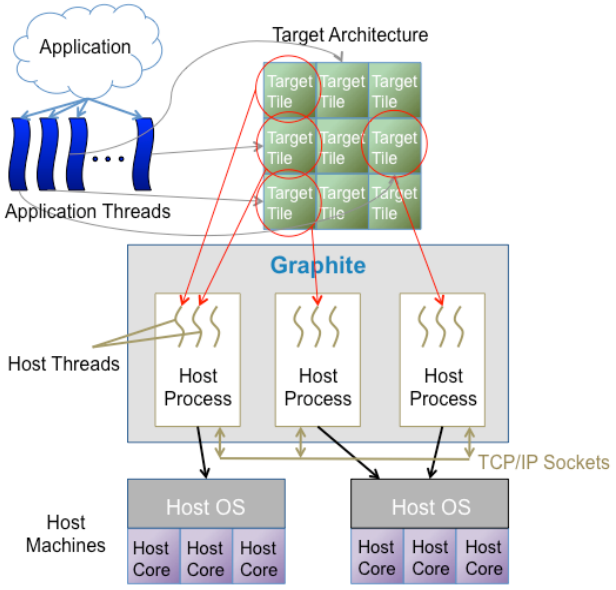

Fig.3 Graphite Architecture.

Graphite is an open source API (application program interface) used to explicitly direct multithreaded, shared memory parallelism, for distributed simulator infrastructure designed to enable rapid high-level architectural evaluation and software development for future multicore architectures. Graphite multicore architecture is used on multicore simulator, designed to boost the power and process in parallelism for multicore engine. It provides both functional and performance modelling for cores, on-chip networks and memory subsystems including cache hierarchies with full cache coherence.

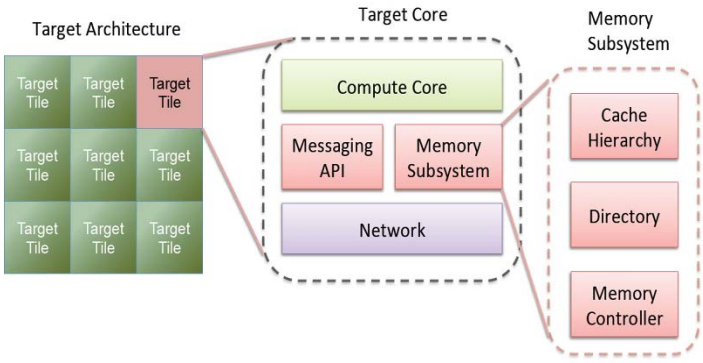

Fig. 4 Target Architecture.

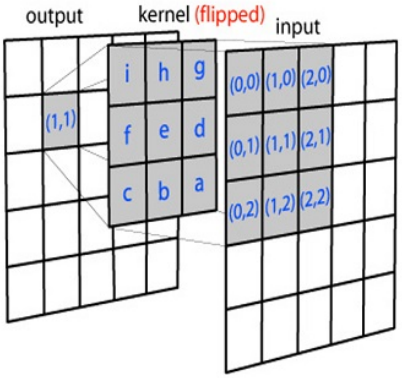

Fig. 5 Two Dimensional Convolution.

This Fig. 4 illustrates the various components present in the target architecture. This target tiled multicore architecture is a thread executing units which are based on multithreaded application. There are various target architecture components such as:

1) Tiles: It contain a processing core, network model and memory components.
2) Core: Basic processing unit for modelling multithreaded application.
3) Memory model: Flexible model responsible for the memory subsystem, which is composed of caches and DRAM at different levels.
4) Network model: Networking model based on routing of packets over the on-chip network and accounts for various delays encountered due to contention and routing overheads[1].

3.2 Algorithm

A. IMAGE AVERAGING FILTER

Image filtering is one of the most commonly used techniques of image processing & enhancement. Neighbourhood processing is prominent concept in image that enables us to extract & modify the image itself as per the requirement.

Image filtering is based on mathematical concept of 2D spatial convolution that is carried out on pixel intensities. Average filtering is one of the most commonly used filters that is utilized to add blurring effect on image as well as to enhance the edge detection.

Fig. 5 explains clearly about how the filtering operation is carried out in spatial domain. For mask of size m x n the averaging filter technique can be represented mathematically as,

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} k(s,t) f(x+s, y+t)$$

Here, m = 2a + 1 & n = 2b+1 with a and b as positive integers.
K(s,t) = Averaging kernel matrix
Average filter kernel of size m x n can be represented as,

$$K_N = \frac{1}{mn} \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}_{m\,x\,n}$$

It is assumed that the kernel coefficient k(0,0) is aligned with centre of the pixel under consideration as.

a. First at the edges of image zero padding is done then appropriate mask is applied for each & every pixel in image
b. 2D convolution is achieved by multiplication & addition of weighted neighbourhood values and value of pixel under consideration is replaced by output of the convolution.
c. Mask is then sheeted row-wise or column-wise and filtering is carried out by processing same steps as 1 and 2 over entire image.

B. IMAGE AVERAGING ALGORITHM

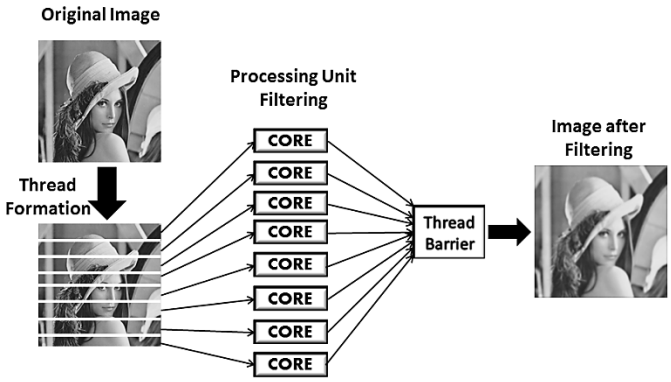

Fig. 6 Image Filtering Algorithm.

This Fig. 6 illustrates the image averaging filtering algorithm as an application on graphite multicore architecture. There are 2 different size images considered such as 786x786 and 1024x1024 for image averaging application. Each image is divided into row by row equivalent to the number of threads allocated. Each thread is given to each core for processing and is retransformed into image with averaging. The process can be explained below

a. Read an image.
b. Divide image into N number of threads.
c. Distribute these threads into the cores. Cores are generated corresponding to the amount of threads. Each thread is responsible for processing included in its core
d. Threads are processed by applying average filtering to them.
e. These Processed threads are given to thread barrier for the reconstruction of the image.
f. Reconstructed image is written in the memory.

C. Implementation in Graphite

This Fig 7 illustrates the graphite multicore architecture while executing image average filtering algorithm based on the modified design that is combining DVFS and heterogeneity. The executions on graphite multicore in various phases are as follows:

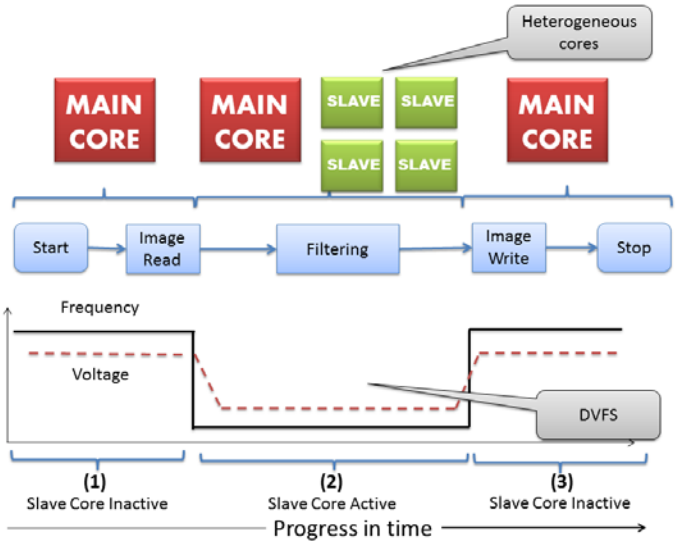

Fig. 7 Graphite Implementation.

1) The execution starts with the main core where the image is read. Here the supply voltage and operating frequency is high. The slave cores are inactive.
2) At the time of filtering, the slave cores become active. The voltage and frequency are made less with the slave cores having half the size of the main core because the processing part in comparatively less.
3) After filtering, the slave cores become inactive. The image is write back and the main core stops execution.

4. Results

Results are generated based on Graphite simulation with following assumptions:

- Infinite DRAM storage
- Negligible DRAM access latency
- Negligible latency during core switching ON or OFF
- Negligible loss of power in connecting network
- Latency while switching voltage & frequency is negligible.
- Completion time for program is assumed to be equal to maximum time taken by core to complete given task

Following specifications are maintained while performing simulation:

- Technology node – 45nm
- Temperature - 300 K
- Maximum operating frequency - 2.0 GHz
- On chip voltage regulator separate for each tile.
- Range of number of cores used - 2 to 32

For performance analysis of modified architecture basic parameters that are observed are processing time & power consumed by entire architecture. Processing time is direct measure of performance while power consumed represents power efficiency of system.

Variations of both parameters that is Processing time & power with respect to change in number of cores used with different frequencies and different images. Following images are considered for performance analysis.

Fig. 8.1 Windmill [Size: 786x786].

Fig. 8.2 Boat [Size: 1024x1024].

Two scenarios are considered while making simulative analysis.

1) Normal scenario that Describes execution under following specification :
   - All executing cores are equal i.e. having same operating frequency of 1GHz
   - Same cache size viz.
     - L1 I-cache size = 16 KB
     - L2 D-cache size = 32 KB
     - L2-cache size = 512 KB

2) Modified scenario that describes execution under following specification
   - Only main core remains as it is & operating at frequency fs = 1Ghz
   - Slave cores operating at frequency fs =0.65 GHz or fs=0.77 GHz
   - Size of slave cache
     - L1 I-cache size =16 KB
     - L2 D-cache size =16 KB
     - L2-cache size =256 KB

A. Following graphs represents time & power variations with respect to number of cores used for different images & frequencies.

1) For Windmill Image
   - Power & Time graph at slave frequency fs=0.77 GHz

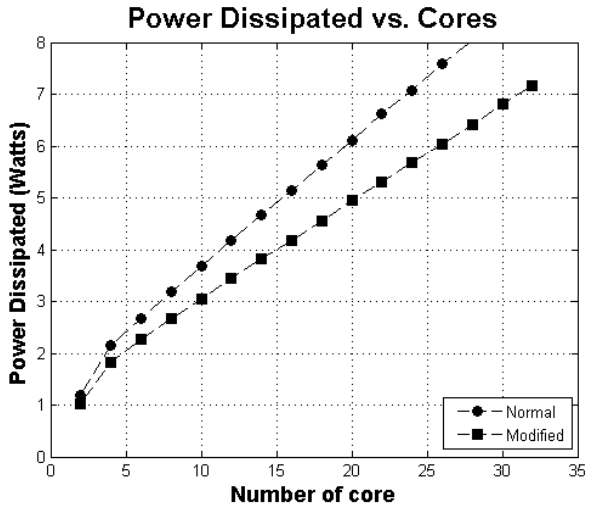

Fig. 9.1 Power Dissipated Vs Cores for Windmill Image at fs = 0.77.

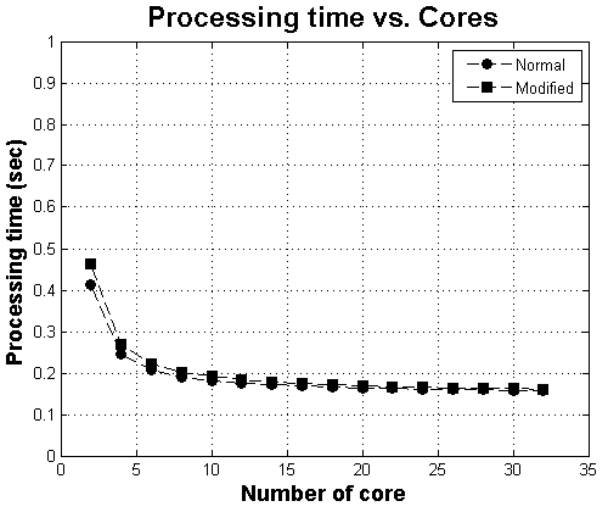

Fig. 9.2 Processing time Vs Cores for Windmill Image at fs = 0.77.

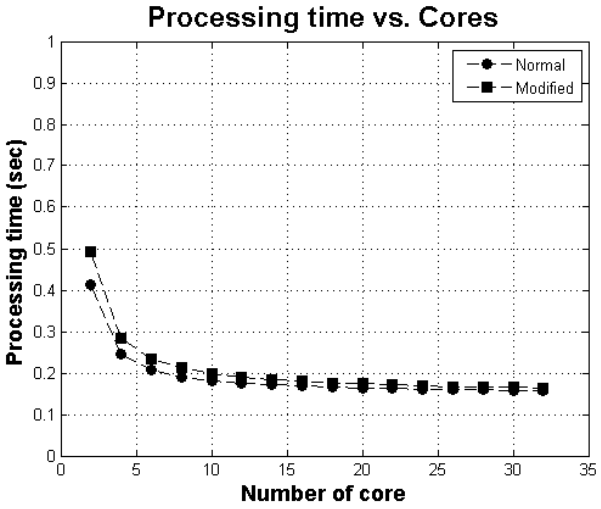

Fig. 10.2 Processing time Vs Cores for Windmill Image at fs = 0.65.

- Power & Time graph at slave frequency fs=0.65 GHz.

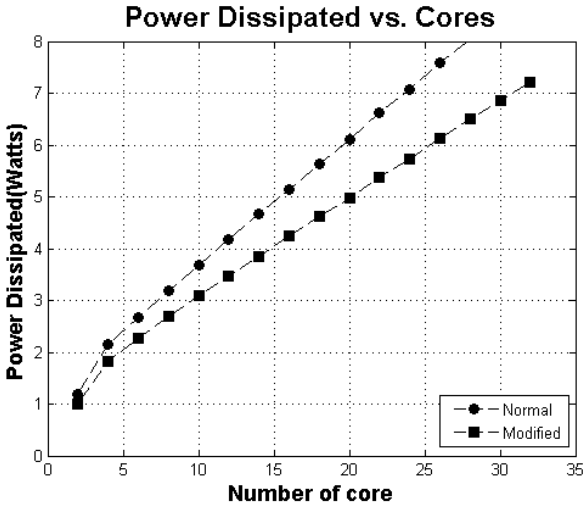

Fig. 10.1 Power Dissipated Vs Cores for Windmill Image at fs = 0.65.

2) For Boat Image

- Power & Time graph at slave frequency fs=0.77 GHz

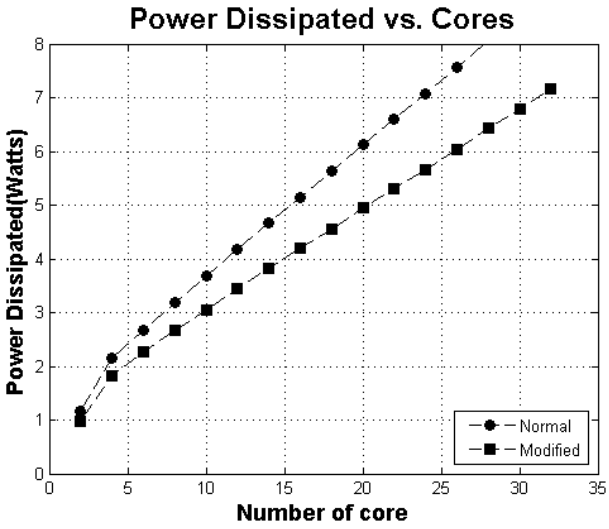

Fig. 11.1 Power Dissipated Vs Cores for Boat Image at fs = 0.77.

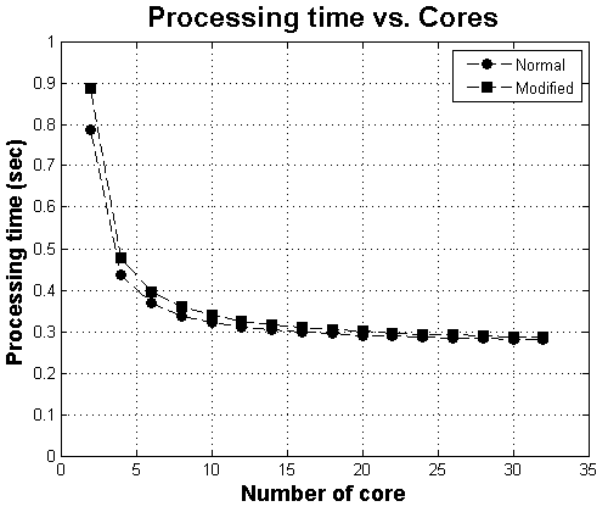

Fig. 11.2 Processing time Vs Cores for Boat Image at fs = 0.77.

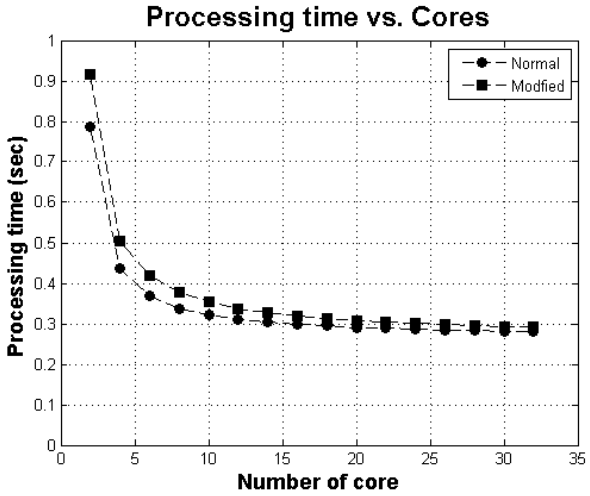

Fig. 12.2 Processing time Vs Cores for Boat Image at fs = 0.65.

- Power & Time graph at slave frequency fs=0.65 GHz

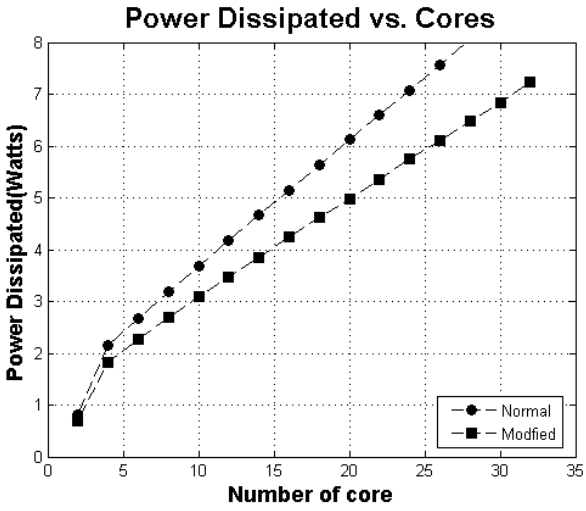

Fig. 12.1 Power Dissipated Vs Cores for Boat Image at fs = 0.65.

B. Comparing all statistical data with simulation under normal execution & following graphs represent percentage variation in parameters as compared to Normal execution.

1) For Windmill image

- At Slave Frequency fs = 0.77 GHz.

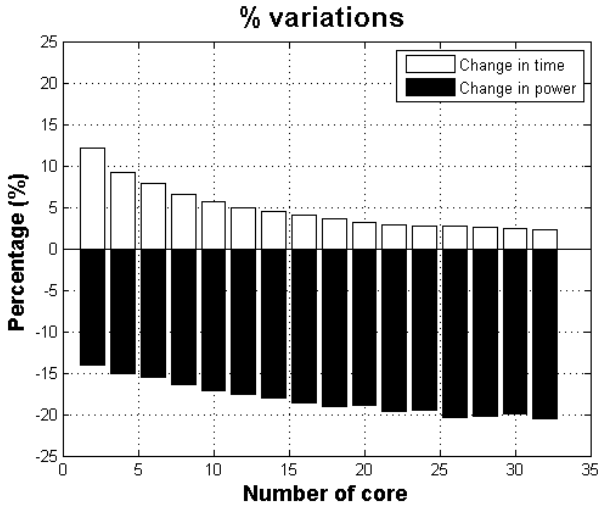

Fig. 13.1 Percentage variation Vs Cores for Windmill Image.

- At Slave Frequency fs = 0.65 GHz

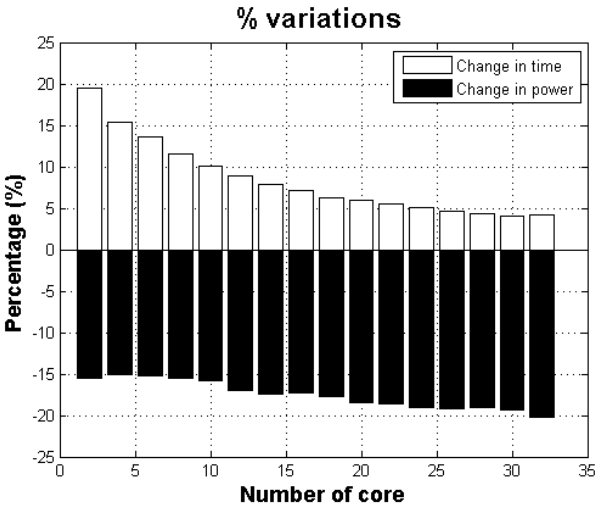

Fig. 13.2 Percentage variation Vs Cores for Windmill Image.

2) For Boat Image

- At Slave Frequency fs = 0.77 GHz

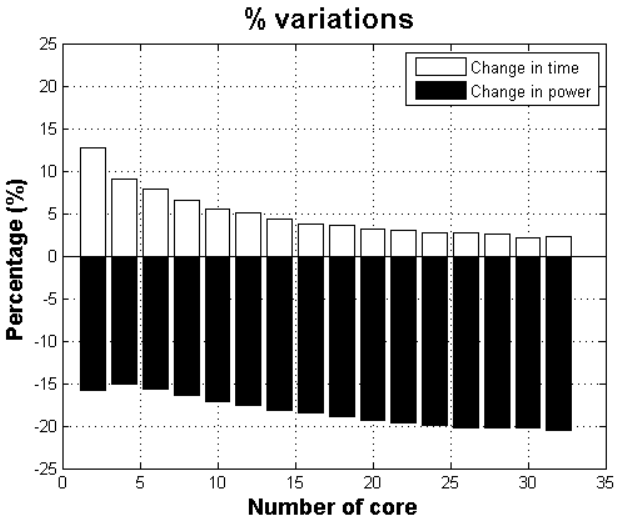

Fig. 14.1 Percentage variation Vs Cores for Boat Image.

- At Slave Frequency fs = 0.65 GHz

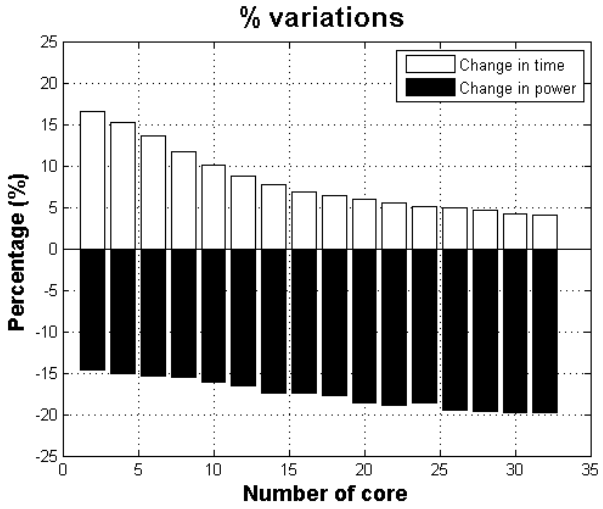

Fig. 14.2 Percentage variation Vs Cores for Boat Image.

5. CONCLUSION

Conclusions are drawn on basis of following perspective:

1) Effect of Computational load: The performance of multicore architecture is completely dependent on the computation load only. Contents of data to be processed becomes irrelevant as simulation computes same resource utilization for different data of similar computational load
2) Processing time: Exponential decrease in processing time is observed as number of processing cores increases. Whereas with decrease in operating frequency, it is evident that processing time also increases by smaller extent.
3) Power consumption: There is linear increase in power as we increase number of cores. Whereas with decrease in operating frequency of slave cores power reduces by great extent.
4) Multicore perspective: With architecture of small number of cores, modified technique shows deterioration in performance. But when it comes to many-core architecture extensive power savings upto 20% can be achieved while allowing only 5% tolerable loss of performance (increase in processing time).

ACKNOWLEDGMENT

The authors would like to thank the reviewers for helpful feedback George Kurian and George Bezerra (CSAIL, MIT) for significant help with the simulation tools and for helpful discussions. We would also thank Carbon Research Group for their resources.

REFERENCES

[1] J. E. Miller, H. Kasture, G. Kurian, C. Gruenwald, N. Beckmann, C. Celio, J. Eastep, and A. Agarwal, "*Graphite: A distributed parallel simulator for multicores*," in Proceedings of the International Symposium on High Performance Computer Architecture, 2010, pp. 1–12.

[2] Rakesh Kumar, Dean M. Tullsen, Parthasarathy Ranganathan, Norman P. Jouppi, Keith I. Farkas, "*Single-ISA Heterogeneous Multi-Core Architectures for Multithreaded Workload Performance,"* In Proceedings of the 31st International Symposium on Computer Architecture, June, 2004.

[3] Oscar Almer, Igor Bohm, Tobias Edler von Koch, Bjorn Franke, Stephen Kyle, Volker Seeker, Christopher Thompson, and Nigel Topham, "*Scalable Multi-Core Simulation Using Parallel Dynamic Binary Translation*," Embedded Computer Systems (SAMOS), 2011 International Conference on 2011 , Page 190 – 199

[4] Lis M, Keun Sup Shim, Myong Hyon Cho, Devadas S, "*Memory coherence in the age of multicores,"* Computer Design (ICCD), 2011 IEEE 29th International Conference on pages 1 – 8

[5] Calcado, F. ; List Embedded Real Lime Syst. Lab., CEA, Gif-Sur-Yvette ; Louise, S. ; David, V. ; Merigot, A." *Efficient Use of Processing Cores on Heterogeneous Multicore Architecture."* Complex, Intelligent and Software Intensive Systems, 2009. CISIS '09,. International Conference.

[6] Ung Ho Ahn, Sheng Li, Seongil O, Jouppi, N.P, "*McSimA: A manycore simulator with application-level simulation and detailed microarchitecture modelling*," Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on pages: 74 – 85

[7] Saripalli V ,Guangyu Sun, Mishra A, Yuan Xie, Datta S, Narayanan V, "*Exploiting Heterogeneity for Energy Efficiency in Chip Multiprocessors*," Emerging and Selected Topics in Circuits and Systems, IEEE Journal on (Volume:1 , Issue: 2)

[8] Jungseob Lee ; Dept. of Electr. & Comput. Eng., Univ. of Wisconsin - Madison, Madison, WI, USA ; Nam Sung Kim *"Optimizing throughput of power- and thermal-constrained multicore processors using DVFS and per-core power-gating."* Design Automation Conference, 2009. DAC '09. 46th ACM/IEEE.

[9] Kumar R, Zyuban V, Tullsen, "*Interconnections in multi-core architectures: understanding mechanisms, overheads and scaling*", Computer Architecture, 2005. ISCA '05. Proceedings. 32nd International Symposium on 4-8 June 2005 ,408 – 419

[10] Ranger C, Raghuraman R, Penmetsa A, Bradski G, Kozyrakis C, "*Evaluating MapReduce for Multi-core and Multiprocessor Systems*," High Performance Computer Architecture, 2007. HPCA 2007. IEEE 13th International Symposium on 10-14 Feb. 2007 , 13 - 24