Application Specific Power Management in Multi-core Architecture

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

AKASH GUPTA **TCS15B004**

MODUKURU TEJA RAMANA **TCS15B020**

Supervisor(s)

Jaynarayan T Tudu

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY TIRUPATI

9^{th} 2019

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Place: Tirupati
Date: 19-05-2019

Signature
Akash Gupta
Roll Number

Modukru Teja Ramana
Roll Number

BONA FIDE CERTIFICATE

This is to certify that the dissertaion titled **APPLICATION SPECIFIC POWER MANAGEMENT IN MULTICORE ARCHITECTURE** submitted by **Akash Gupta** and **Modukuru Teja Ramana**, to the Indian Institute of Technology, Tirupati, for the award of the degree of **Bachelor of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Place: Tirupati
Date: 19-05-2019

Jaynarayan T Tudu
Guide
Assistant Professor
Department of Computer Science and Engineering
IIT Tirupati - 517501

ACKNOWLEDGEMENTS

Our sincere gratitude is reserved for Dr. Jaynarayan T Tudu for his invaluable insights and suggestions. We really appreciate his willingness to help and meet us whenever we need some clarification. we are amazed that even during his busy schedule, he will never say no, instead he was always willing to help us with our thesis. The useful discussion and comments that he suggests us widens our knowledge in various fields of the subject throughout the course of my study.

ABSTRACT

KEYWORDS: Multi-core Architecture; Power dissipation; Glitch power; Dynamic instruction count;

In the multi-core architecture, the power consumption problem has emerged as a roadblock for performance enhancement. There are two reasons how the power dissipation affects the performance: the first reason is the power dissipation limits the number of core that can be functional simultaneously, and the second reason is that the power dissipation limits to increase the clock frequency of a processor. The multiple cores and higher clock frequency have been the two most effective approaches to increase the performance. Currently, most of the general purpose processors and high performance server processor are operated at 2-3 GHz clock frequency and most of the modern processors are having a number of cores ranging from 4 to 36.

In this project we have carried out a study on power dissipation of two different processors for different applications suits from PARSEC and SPLASH benchmark. We have also proposed a solution to solve the power dissipation solution by reducing the over all power dissipation without affecting performance.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

IQ	Instruction Queue
LSQ	Load/Store Queue
ALU	Arithmetic Logic Unit
FP	Floating Point
INT	integer
I-cache	Instruction cache
D-Cache	Data Cache
D-RAM	Dynamic Random Access Memory
ROI	Region Of Interest
PARSEC	Princeton Application Repository for Shared-Memory Computers
MCPAT	Multicore Power Area and Timing
TLB	Translation Local Buffer
V	Volatage
W	watt
GHz	Giga Hertz
ns	Nano Seconds

CHAPTER 1

INTRODUCTION

1.1 Introduction and Background

Although the performance level of out-of-order superscalar processors is high,they consume much more energy than do in-order superscalar processors,because a large amount of energy is consumed by hardware for dynamic instruction scheduling such as an instruction queue(IQ) and a load/store queue (LSQ), which comprises mainly heavily multi-ported memories.

The energy consumption per access of multi-ported memory. The energy consumption per access of multi-ported memory is proportional to its capacity and the number of its ports. Moreover, the number of accesses is also increased with the issue width. Consequently, its energy consumption is very large. We are going to develop a power-aware Adaptive superscalar Architecture which consumes lower energy and saves more power than the existed architectures.

1.2 Power Dissipation

In this section we will provide a background on power dissipation. There are two type of power dissipation Henessy and Patterson (2012):

- Dynamic power dissipation
  - Glitch power
  - Functional power
- Static power dissipation

The dynamic power is defined as the switching power that is dissipated when the processor is in operation. Mathematically, the dynamic power is computed as the total energy divided by a given time duration. The dynamic power comes mainly from two sources: *true functional activity* and *glitch related activities*. The true functional activity which is required for the computation is the compulsory power that the processor has

to dissipate in order to perform the computational task. Whereas the glitch activity are not the essential activity but they comes due to inherent limitation in circuit design. In this work, our objective is to find out the architectural level solutions to reduce the total power dissipation by reducing the static power and glitch power.

The other type of power dissipation that has gain importance is the static power. The static power is also known as leakage because it happens irrespective of any change in inputs. So far there is power supply on, the leakage power continue to dissipate.

1.3 The Power Dissipation in Multi-core Architecture

The excessive power dissipation puts a limit on performance improvement of processor. We address the dynamic power problem by exploring the possibility of reducing the glitch activity in Execution unit of superscalar processor core.

We have carried out a set of experiment to evaluate the power dissipation for difference benchmark from PARSEC and SPLASH suit. The experiment has been carried out to understand and observe the power dissipation profile of each benchmark program for different configuration of cores. Further, the experiment has also been carried out to observe the power dissipation of each units at micro-architecture level. From the plot it can be observed that the power dissipation in Execution unit which are labaled in the plots as Core-fp, Core-ALU, and Core-int are relatively high compared to the other stages of super-scalar pipeline.

Also from the plot of total power dissipation for entire core it can be observed that the power dissipation for different benchmark vary differently. This observation leads us to the understanding that to reduce the over all power dissipation by some means the problem needs to addressed on the basis of application (application here we mean the benchmark program). Therefore we looked for the solution which will be addressing the power problem on the basis of application. Which leads us to the solution which is of dynamic in nature.

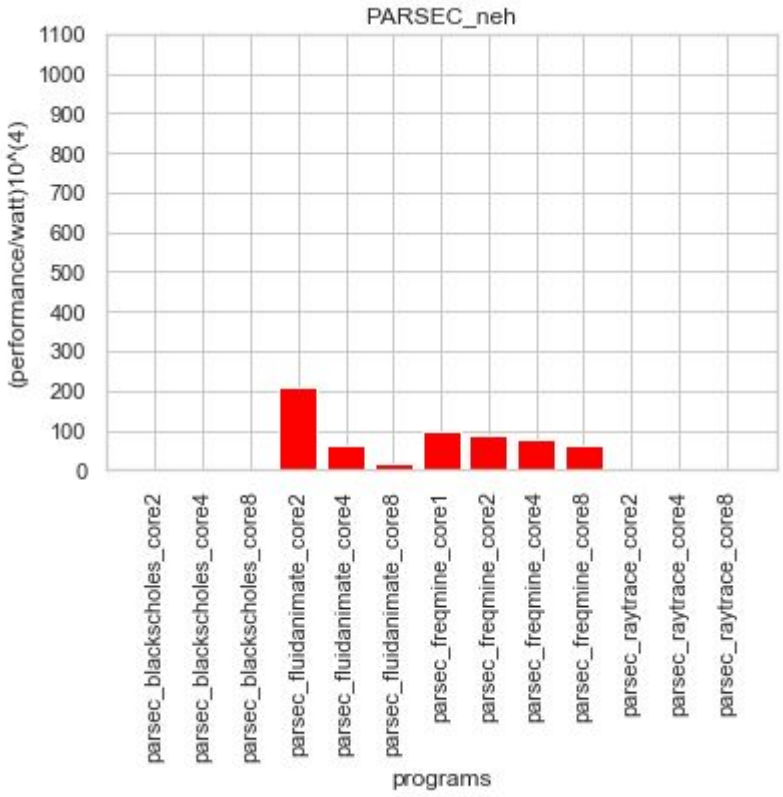


Figure 1.1: Total power dissipation for PARSEC benchmark with varying number of cores

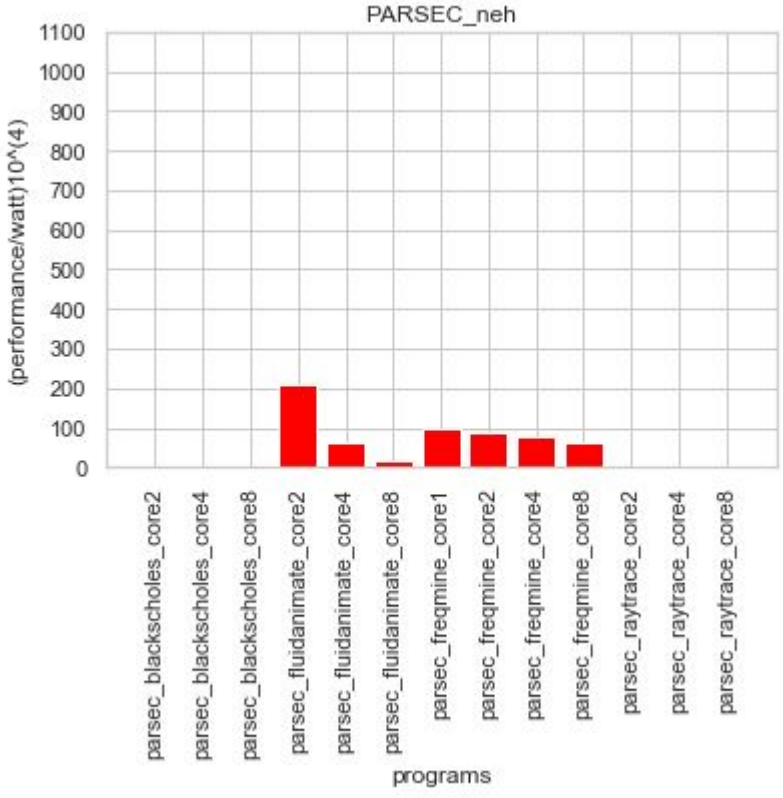


Figure 1.2: Total power dissipation for SPLASH benchmark with varying number of cores

1.4 The Contributions

- Experiments to identify the power dissipation problem in Execution unit
- Designing a configuration for atom-like processor core
- Architecture for reducing glitch activity in Execution unit

1.5 Chapter Organisation

The Chapter II provides literature review on power dissipation problem in multi-core architecture. The Chapter III explains in detail the problem of interest and the proposed solution. The Chapter IV gives experimental results and report concludes in Chapter V. Additional information on sniper simulator, bench marks, and few programs are provided in Appendix.

CHAPTER 2

REVIEW OF PREVIOUS WORK

The power dissipation problem has emerged as one of the major hindrance for performance improvement in modern processor. There have been several approaches proposed so far to mitigate the power problem. Different ideas from the various areas like compiler design, architecture, and circuit design have been proposed to solve not only the dynamic power but also the static power.

We have carried out a small amount of literature review to gain the background knowledge on power issues in processor architecture. The literature study also gave us the understanding about the proposed methodology and the unsolved problems.

2.1 Micro-architecture Approaches

One of the area that have been studied seriously with respect to managing the power consumption in processor is to design an efficient power management unit Isci *et al.* (2006). The main idea of having an efficient power management unit is to detect the critical level of power consumption in terms of chip temperature, peak power, or battery life and them schedule the decide to reduce the work load for a particular core or completely turn-off the core for a given amount of time. The work of Canturk et al have rigorously analysed the power management policies and proposed an efficient policy that maintains the power and performance trade-off much better than the earlier proposed techniques. There have been a work done by Reda et al Reda and Belouchrani (2017) on determining the processor's component to measure the power consumption efficient for it to be used by power management unit.

To perform the power simulation at micro-architecture level there have been a few simulators developed, one which we use, called **McPAT**, models the power consumption for a processor architecture with reasonably accepted accuracy Li *et al.* (2009). The McPAT in integration with the performance simulator **Sniper** measures the total power dissipation as well as dynamic and static power dissipation.

There have been approach to reduce the power consumption in a single core, and thereby reduce the overall power in multi-core system. The main idea behind such kind of approaches is to design the single core in such a way that it minimizes the wastage of computational resources such as Execution units, Register file, Reorder buffer etc. Shioya et al proposed a micro-acrhitectural technique called FXA, which segregate the execution units into two parts: front-end execution and back-end execution Shioya *et al.* (2016). The front-end execution unit is a light weight in-order execution of instruction whereas the back-end unit is the outof order execution. The idea is to reduce the load on back-end execution which consumes large amount of power compared to the front-end execution due to its large que sizes and scheduler. The front-end execution unit is simple as it does need to have any que because it process all the independent instructions. This technique improves the performance per watt for SPEC benchmark program.

2.2 Circuit Design Approaches

Two approaches have been explored to minimize the processor power dissipation at circuit level. Both of the approaches fundamentally comes from the equation for dynamic and static power consumption. The dynamic power is $0.5CV_{dd}^2f$ where C output load, V_{dd} is supply and f is the clock frequency. The two techniques explored are: Power gating and Dynamic voltage and frequency scaling (DVFS). The power gating techniques basically cut-off the power supply ie V_{dd} and therby it controls both the Dynamic and Static power. Whereas, the $DVFS$ technique dynamically controls the voltage and frequency to keep the power consumption under a threshold level. The McPAT Li *et al.* (2009) simulator models both the technique to simulation.

A recent work which designs a power management unit having fine-grained $DVFS$ is reported by Haghbayan *et al.* (2014). The DVFS proposed there have capability to scale the voltage and frequency to finer level, meaning the voltage be scaled down to the near-threshold limit.

2.3 Objectives and Scope of the Project

The scope of the project is limited to the power consumption analysis for different applications from PARSEC and SPLASH benchmark suit. The objective of the power analysis is to figure out the power dissipation and performance of individual application with respect to variation in cores and utilisation of execution units. We have carried out the power analysis by the process of architectural simulation using the simulators: Sniper and McPAT. Apart from the power analysis, we also propose an architectural mechanism to manage the power by turn-off and turn-on of selective functionals units based on a set of criteria.

The implementation at hardware level, simulation with detail power gating, and functional simulation with our proposed design are beyond the scope of this work. The work is also based on some of the assumption such as the turn-on and turn-off circuitry are efficient enough that they won't take time more than 5-10 seconds.

CHAPTER 3

THE PROPOSED POWER MANAGEMENT UNIT

This chapter presents a detailed description of the problem and the proposed solution. We have carried out an experiment to study the power dissipation problem for different set of applications on two different processor architecture. The problem we identified is the excessive power dissipation in the Execution unit of a core. We addressed this problem by proposing a solution which provides a provision for turning-off and turning-on the selected functional units dynamically at any given point of time.

3.1 Analysis of Problem

This section presents and analyses the problem of interest. We carried out set of experiments observe the power problem at different level of architecture for different applications from PARSEC and SPLASH suits. The experiment has been done for both the configuration of architecture, the Nehalam which is pre-configured and the atom which is configured by us based on the understanding of atom architecture.

3.2 The Proposed Architecture

3.3 Working Principle

The PIID acts a detector and stops the power consuming process based on priority if consuming process has high priority it allows it to execute. But suppose it has low priority or there is no priority it executes low power instruction first by turning it off. We are not continuously turning on and off first we will take the instructions which is in binary format we will separate 0's and 1's and we first execute 0's sequence and next 1's. In this case we will have number of turnoff and on process will be less.

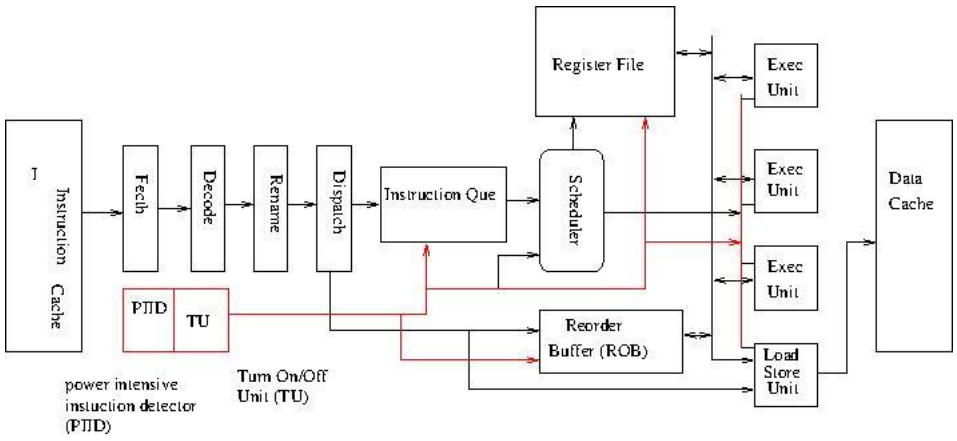


Figure 3.1: A core with super scalar processor architecture with a proposed power reducing unit

Integer adder/sub	counter 1	threshold 1
Load/Store	counter 2	threshold 2
Integer Multiplication 1	counter 3	threshold 3
Integer Multiplication 2	counter 4	threshold 4
Integer adder/sub 2	counter 5	threshold 5
FP Divider	counter 6	threshold 6
FP Multiplication	counter 7	threshold 7

Figure 3.2: A unit to track the threshold for deciding the power-off based on counter value

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Experiment for Nehalem and Atom Comparison

Here we have compared power consumption and performance of different applications used in 2 benchmarks splash and parsec for both Atom and Nehalem architecture. FFM,FFT,RAYTRACE are the applications used in splash. Blacksholes,Fluidanimate,Freqmine, RAYTRACE are the applications used in parsec

parsec will run minimum 2 cores at a time.

In Figure 4.1 we can see power(in watt) consumption of different applications in 2 benchmarks splash and parsec for Atom for core1,core2 and core4 in splash and core2,core4,core8 in parsec.we can visualize that FMM application for core4 is consuming more power compared to other applications in splash.And Fluidanimate application for core 8 is consuming more power compared to others applications in parsec.

In Figure 4.2 we can see performance(in nanoseconds) of different applications in 2 benchmarks splash and parsec for Atom for core1,core2 and core4 in splash and core2,core4,core8 in parsec.we can see that some applications in both benchmarks are taking less nanoseconds(in time) i.e less than zero.you can see in application Freqmine in parsec performance is not improving much even though it is taking more number of cores.

Figure 4.3 and Figure 4.4 are separate graphs of parsec and splash in atom.

In Figure 4.5 we can see power(in watt) consumption of different applications in 2 benchmarks splash and parsec for Nehalem for core1,core2 and core4 in splash and core2,core4,core8 in parsec.we can visualize that FMM application for core4 is consuming more power compared to other applications in splash.And Fluidanimate application for core 8 is consuming more power compared to others applications in parsec.

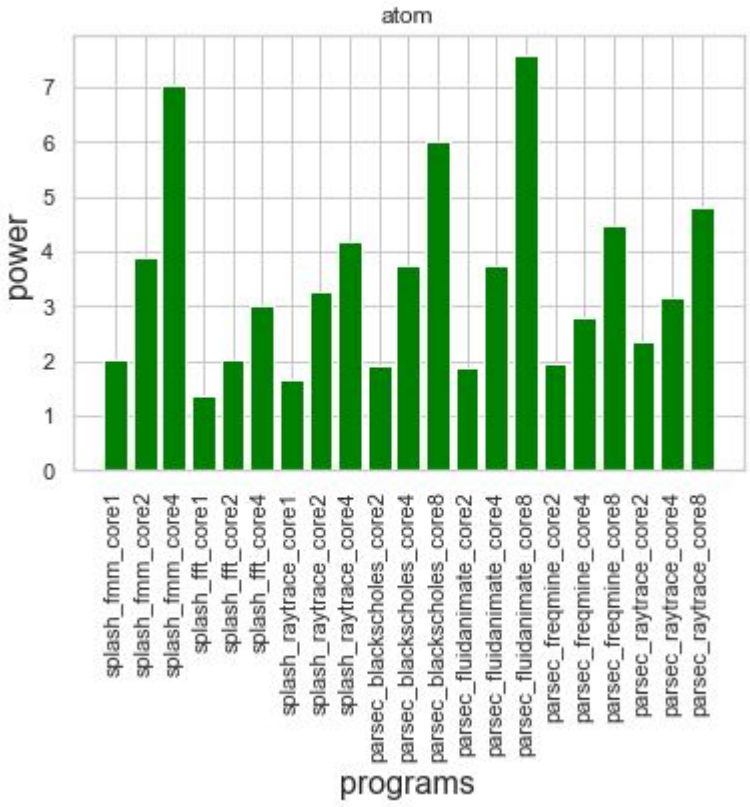


Figure 4.1: Power consumption in atom architecture (Here power is measured in watt)

In Figure 4.6 we can see performance(in nanoseconds) of different applications in 2 benchmarks splash and parsec for Nehalem for core1,core2 and core4 in splash and core2,core4,core8 in parsec.we can see that some applications in both benchmarks are taking less nanoseconds(in time) i.e less than zero.you can see in application Freqmine in parsec performance is not improving much even though it is taking more number of cores

In Figure 4.7 for we plotted pie chart for both Atom and Nehalem for 2 different bench marks parsec and splash we compared the power consumption of each unit like Alu, Integer(int), Fetch, Floating point(fp) for Atom we reduced frequency from 2.4GHz to 2.2GHz and for Nehalem we reduced voltage from 1V to 0.8V.we can see that for some units power is decreasing and for some units power is increasing.

4.2 Core Level Power Off

In Figure 4.8 We can see that for RAYTRACE Application in parsec benchmark instructions are present in only one core in second core we have zero instructions but

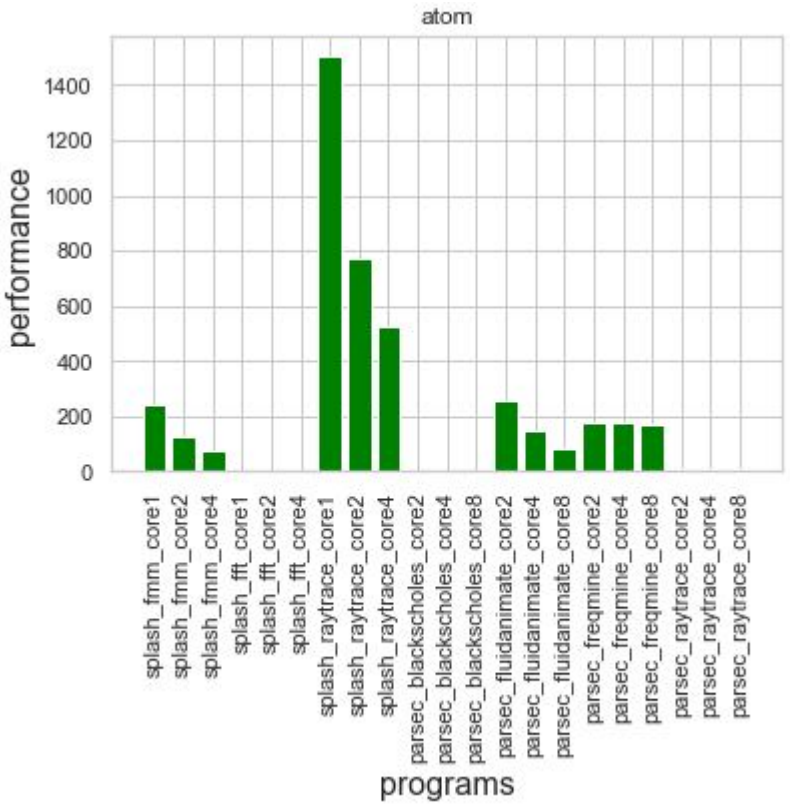


Figure 4.2: Performance of atom processor (performance is measured in terms of execution time)

it is running both the cores.In this case it is better to turnoff the second core. So We have turned off the whole core when there are zero instructions in second core for RAYTRACE Application.

4.3 Micro-architecture Level Power Minimization

First, we have counted the number of instructions in each unit.you can see the figure 4.9 and 4.10 which shows how many instructions are present in each unit in each core.we have showed for core1,core2,core3,core4

In Figure 4.11 We have set the timer with a threshold of 5*ns* and 10*ns* how to calculate the static power of a particular unit in the duration of the timer threshold.we can observe from the figure 4.11 that for 5*ns* we are saving power of 40 percent for particular unit and for 10*ns* we are saving more than 80 percent we know the total power of a core, we also know total dynamic power of each unit and core. static power of each unit=total

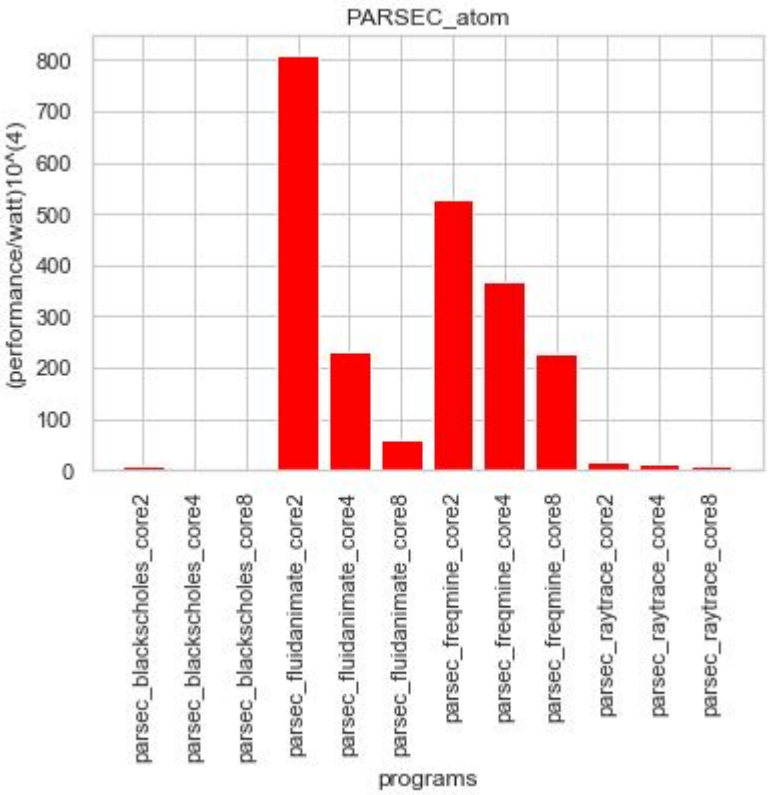


Figure 4.3: Performance of atom processor for PARSEC benchmark

core power-total dynamic power of core/total number of units

$$total_power = dynamic_power + static_power \tag{4.1}$$

static power consuming in threshold duration = (static power of particular unit/total time of core during the execution of application)* threshold time

In Figure 4.12 we can see that switching off particular core for 10ns we are loosing performance compared to switching off unit for 5ns

threshold time = A limits of time which indicates the functional unit has to turn-off.

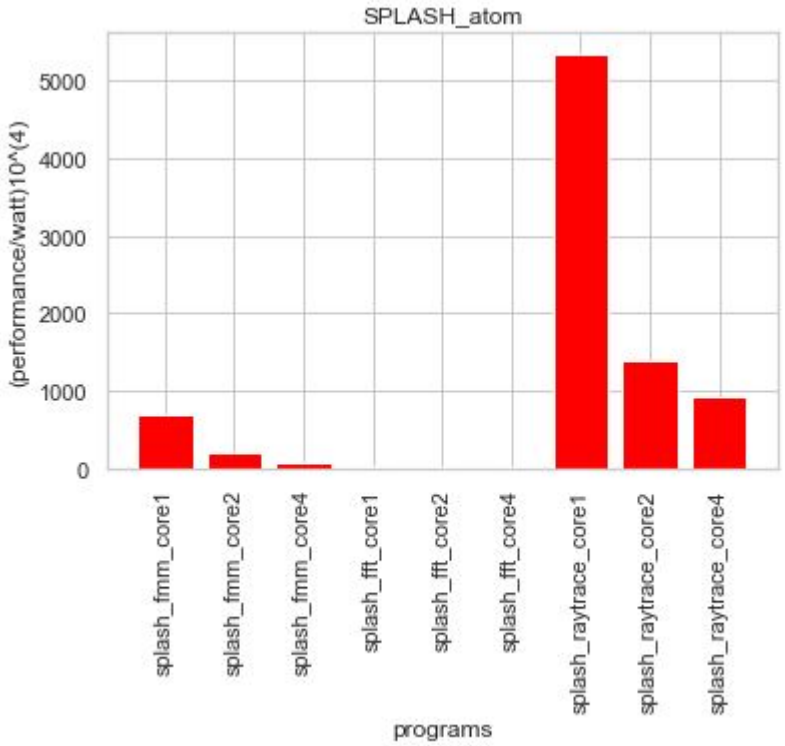


Figure 4.4: Performance of atom processor for SPLASH benchmark

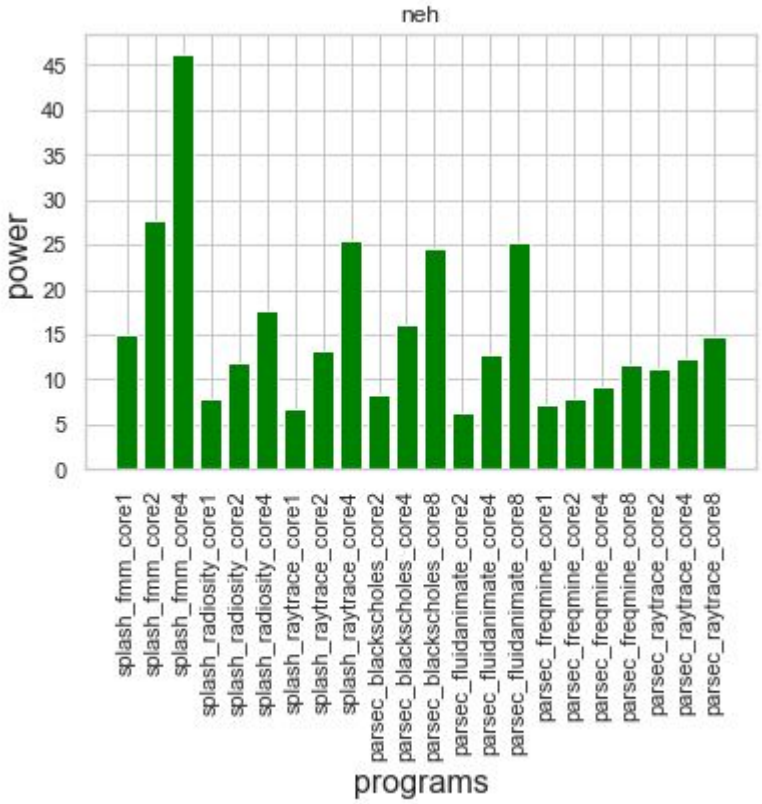


Figure 4.5: Power consumption for Nehalem architecture

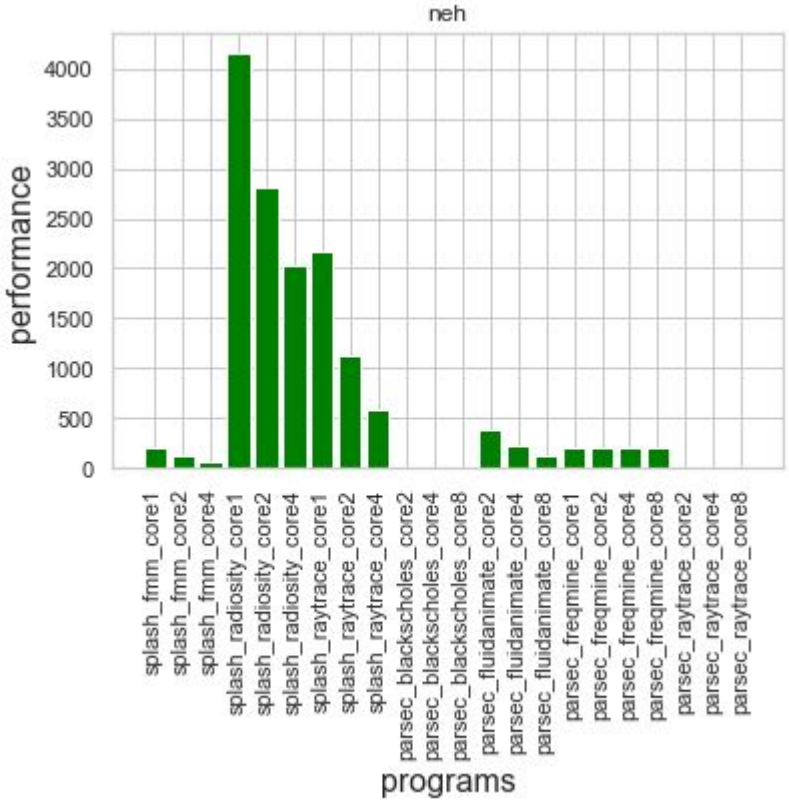


Figure 4.6: Performance for Nehalem architecture

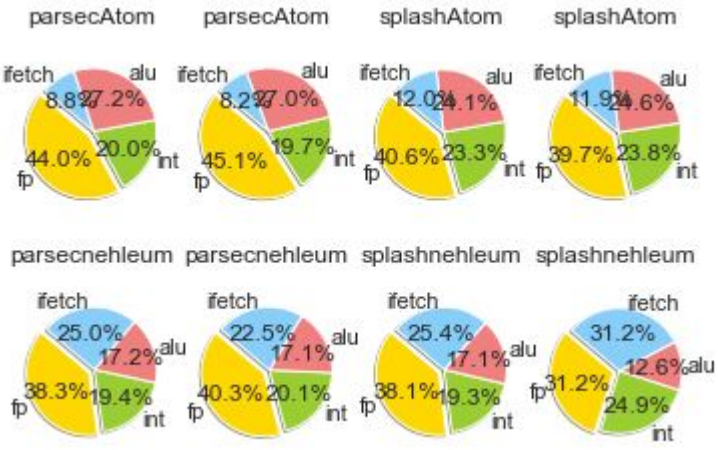


Figure 4.7: Power consumption for each unit

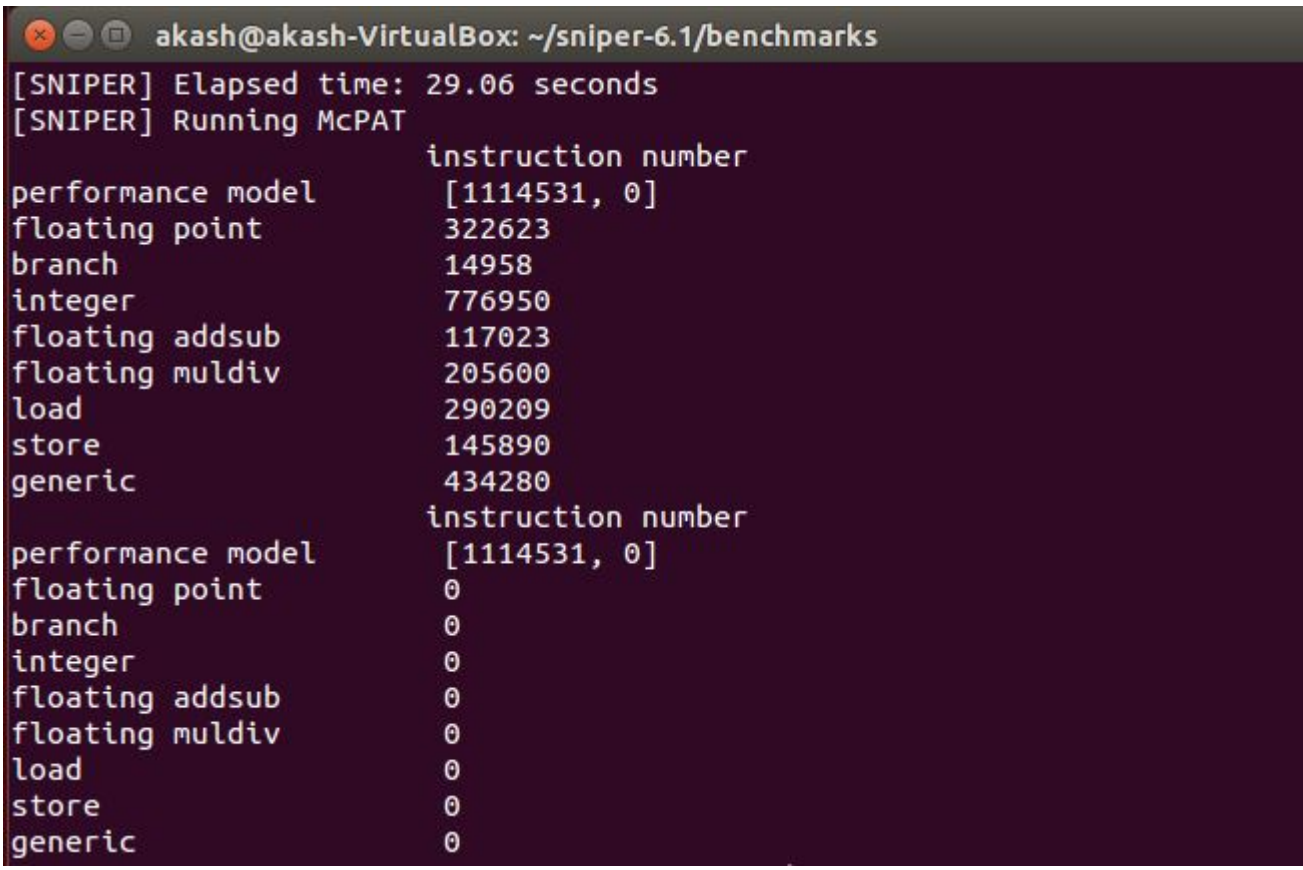


Figure 4.8: Power profile for RAYTRACE application for two-core architecture

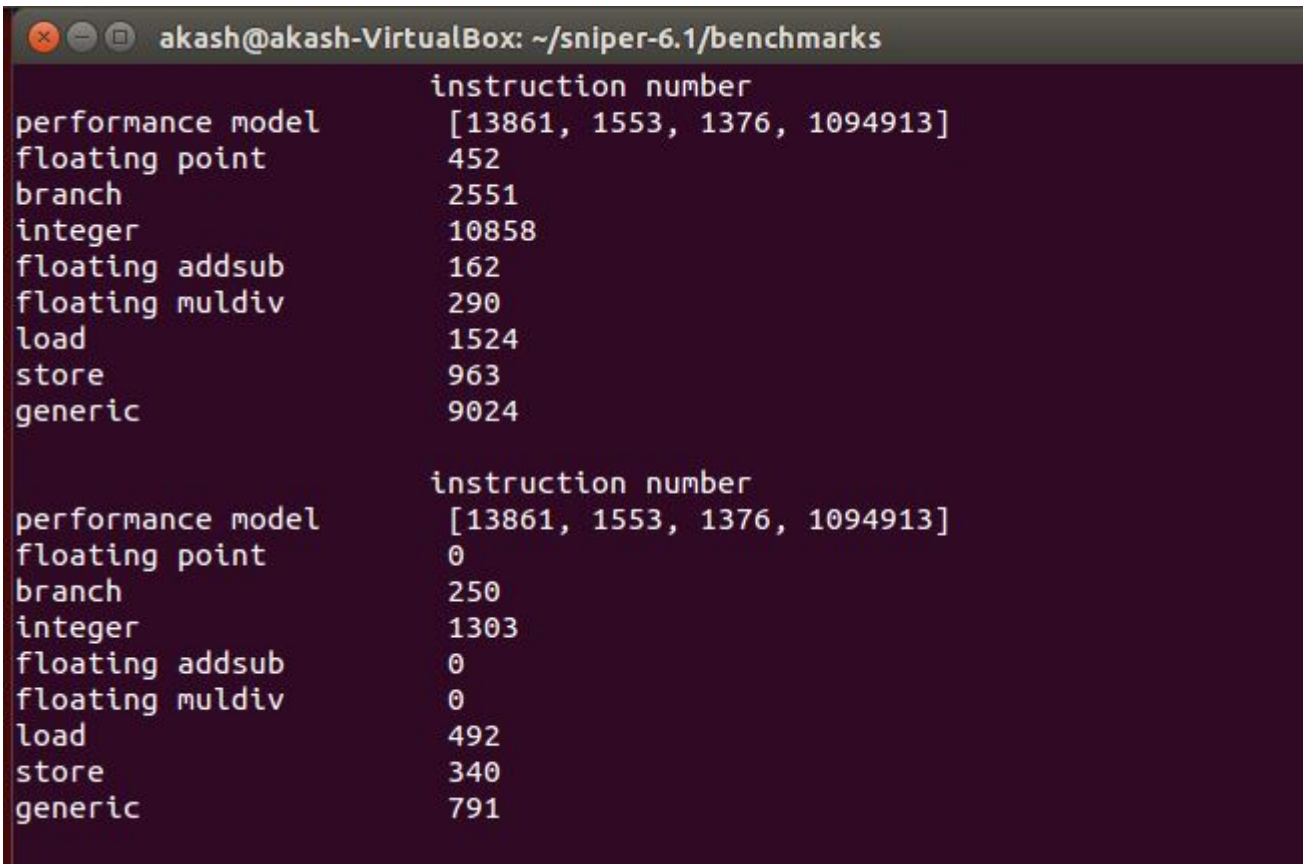


Figure 4.9: Power profile for RAYTRACE application for Four-core architecture

```
akash@akash-VirtualBox: ~/sniper-6.1/benchmarks

                        instruction number
performance model        [13861, 1553, 1376, 1094913]
floating point           0
branch                   226
integer                  1150
floating addsub          0
floating muldiv          0
load                     443
store                    304
generic                  707

                        instruction number
performance model        [13861, 1553, 1376, 1094913]
floating point           322171
branch                   11055
integer                  761687
floating addsub          116861
floating muldiv          205310
load                     290748
store                    144536
generic                  419059
```

Figure 4.10: Power profile for RAYTRACE application for Four-core architecture

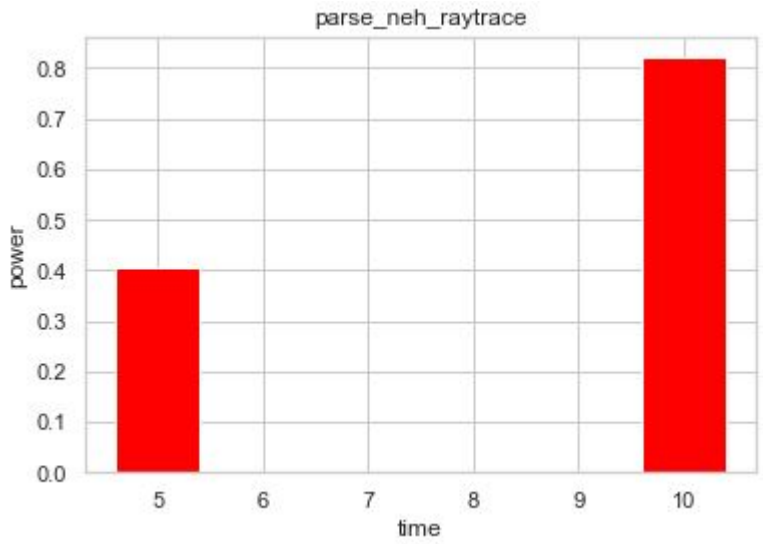


Figure 4.11: Power saving obtained by proposed methodology for RAYTRACE application.

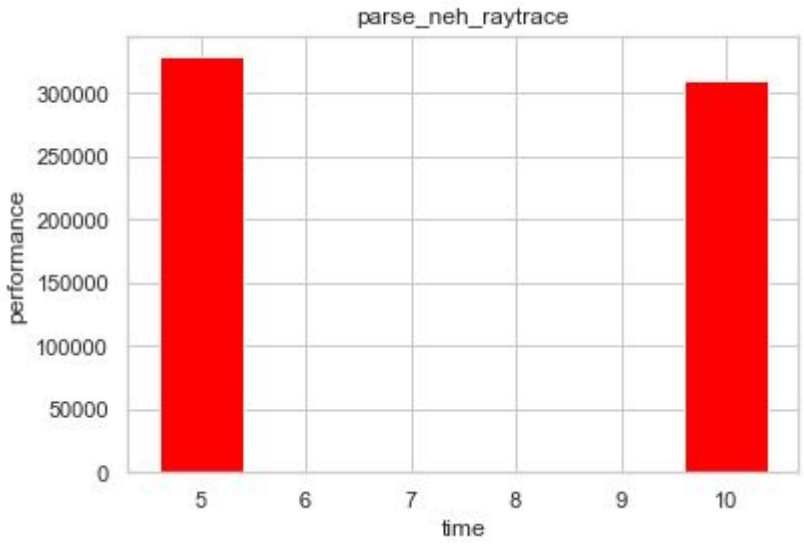


Figure 4.12: Performance loss due to extending the turning off time for RAYTRACE application.

```
                  Power      Energy    Energy %
  core-core       0.50 W     0.22 mJ      6.96%
  core-ifetch     0.11 W     0.05 mJ      1.48%
  core-alu        0.34 W     0.15 mJ      4.68%
  core-int        0.25 W     0.11 mJ      3.42%
  core-fp         0.55 W     0.24 mJ      7.71%
  core-mem        0.13 W     0.06 mJ      1.79%
  core-other      0.13 W     0.06 mJ      1.78%
  icache          0.13 W     0.06 mJ      1.88%
  dcache          0.56 W     0.25 mJ      7.84%
  dram            4.42 W     1.94 mJ     61.62%
  other           0.06 W     0.03 mJ      0.83%

  core            2.00 W     0.88 mJ     27.83%
  cache           0.70 W     0.31 mJ      9.72%
  total           7.17 W     3.15 mJ    100.00%
[PARSEC] [----------    End of output    ----------]
```

Figure 4.13: Power Consumption of RAYTRACE application using parsec benchmark in atom .

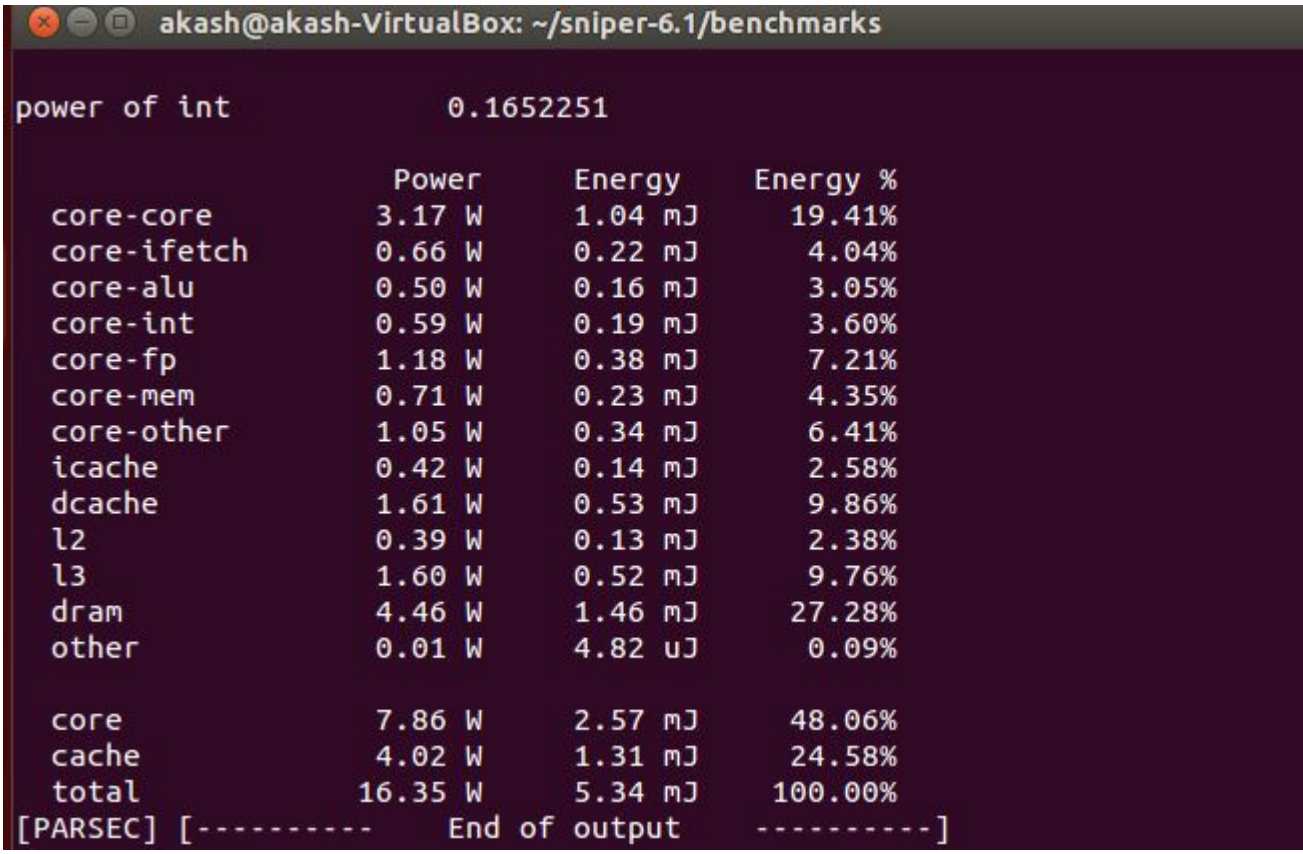


Figure 4.14: Power Consumption of RAYTRACE application using parsec benchmark in Nehalem.

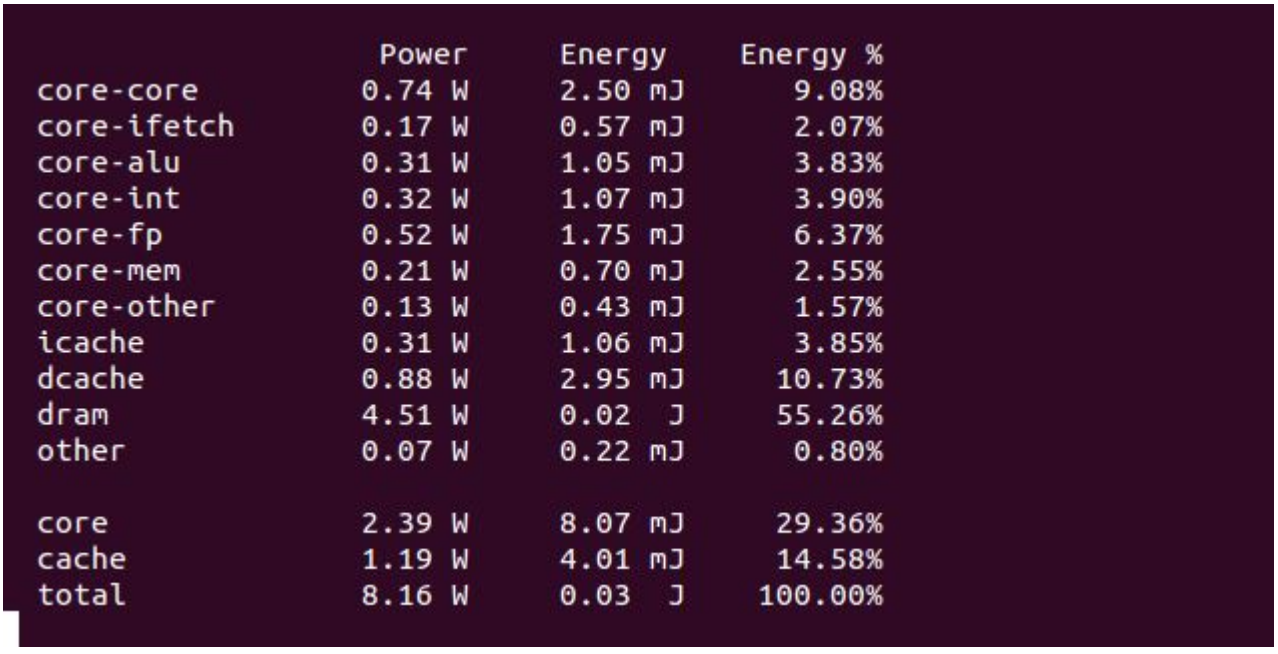
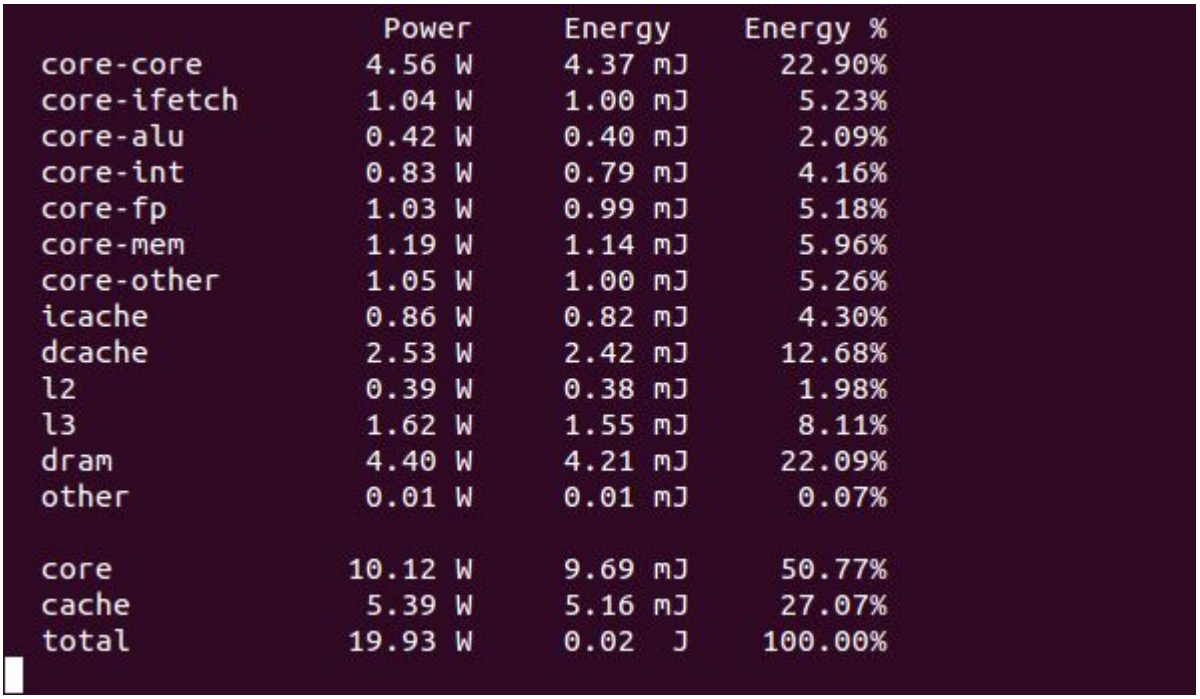


Figure 4.15: Power Consumption of RAYTRACE application using Splash benchmark in atom.



	Power	Energy	Energy %
core-core	4.56 W	4.37 mJ	22.90%
core-ifetch	1.04 W	1.00 mJ	5.23%
core-alu	0.42 W	0.40 mJ	2.09%
core-int	0.83 W	0.79 mJ	4.16%
core-fp	1.03 W	0.99 mJ	5.18%
core-mem	1.19 W	1.14 mJ	5.96%
core-other	1.05 W	1.00 mJ	5.26%
icache	0.86 W	0.82 mJ	4.30%
dcache	2.53 W	2.42 mJ	12.68%
l2	0.39 W	0.38 mJ	1.98%
l3	1.62 W	1.55 mJ	8.11%
dram	4.40 W	4.21 mJ	22.09%
other	0.01 W	0.01 mJ	0.07%
core	10.12 W	9.69 mJ	50.77%
cache	5.39 W	5.16 mJ	27.07%
total	19.93 W	0.02 J	100.00%

Figure 4.16: Power Consumption of RAYTRACE application using Splash benchmark in Nehalem.

CHAPTER 5

CONTRIBUTIONS AND SUMMARY

We have addressed the power dissipation problem in multi-core processor architecture. We have carried out two important experiments to figure out the power consumption of the applications from PARSEC and SPLASH benchmark. The first experiment is to observe the power dissipation by different applications across varying number of cores and the second experiment is to observe the power dissipation by each individual unit within a core. The experiment reveals that different application have different performance per power profile. The second experiment indicates that the execution stage in pipeline is the maximum power consuming unit within a core. Further, we have conducted the experiment to identify the busy and idle time of each functional unit for different applications.

Our next contribution is the proposed power management unit which facilitate the turn-on and off of the selected functional units. The power management unit proposes the heuristic methodology to detect the time of turning off and on of the selected functional unit. The proposed methodology is then experimentally verified which show a reduction of power dissipation compared to the base architecture for some of the applications. In summary, the project contributes the following:

- Identifying the power dissipation for individual functional units for different applications
- Identifying the idle and busy time of each functional unit
- Proposed a power management unit for dynamically turning-on/off the selected functional units
- Designed a configuration file for Intel Atom processor architecture

The project is limited to the extend that it is based on some of the technological assumption which depends on the circuit design. Therefore, a further research work is needed to experimentally validate the efficiency of proposed work when technological details: such as feasibility of micro-architecture level power gating, time required to turn-on the logic block, and functional simulation of the proposed power management unit are taken into consideration.

APPENDIX A

Simulator,MCPAT,Configuration, Benchmarks,RAYTRACE

A.1 Simulator

It is a software that predict the realistic output performance metrics based on the input given by user,application,device etc.

We have used Simulator called Sniper.

A.2 Sniper

Sniper is a Multicore simulator.Basically it generates the realistic output for the processor based on configuration which is design by user or some standard configuration.It provides performance,behaviour,power of processor and its units.

In this simulator we can define ROI(Region Of Interest) which can target any section of a processor.we can generate visual effects like topology,3D visualization of time,core and instruction per section.

A.3 MCPAT

Basically it is first integrated power,area, and timing modeling framework used for multithreaded and multicore processors.with variety of performance simulators it is designed.If user wants to declare only high level parameters for architectures it provides default values for low level hardware.It uses XML interface for generating performance.It runs separately from a simulator and it only reads performance statics from it.

In this the user specifies area,power deviation,target clock frequency, optimization function and other architectural or circuit or technology parameters.

It is also used to calculate the corresponding power dissipation for the particular period and when performance simulator require it will send it back.

A.4 Configuration

These are changes done between atom and nehalem configuration file.

1.In atom file we changed frequency in comparision to nehalem

2.We changed window size of core interval timer.

3.In I-TLB we changed size and associativity and same thing done in D-TLB also.

4.Voltage and technology node(chip size).

5.In L1 and L2 I cache associativity and its size.

6.we changed data access time of L2 cache.

7.D Ram controller Bandwidth

8.Network Bus Bandwidth.

A.5 Benchmarks

We have 2 different Benchmarks Splash and Parsec

Parsec is a collection of parallel programs and can be used for performance study of multiprocessor.

It's purpose is to build and run packages as well as perform other management operations.

It is composed of software packages which are the benchmark programs and their required libraries and tool and the frame work.

It distinguish between global and local configurations.

It uses name of build configuration and platform for which the program the program is compiled to form a key that is use to distinguish different builds and installation of

package.

Splash

The report generated by the splash describes the structure function's and performance characteristics of each application.

A.6 RAYTRACE

Name: raytrace

Description

It generates visually ,realistic image by tracing the path of light in the whole scene.

It leverages the physical property that the path of light is always reversible to reduce the computational requirements by following the light rays from the eye point through the image plane to the source of the light.

This way only light rays that contribute to the image are considered.

This benchmark is basically develop for real time animations example:computer games.

It is optimized for speed rather than realism.

The computational complexity of the algorithm depends on the resolution of the output image and the scene.

Input/Output:

The input for raytrace is a data file describing a scene that is composed ofa single, complex object.

The program automatically rotates the camera around the object to simulate movement.

The output is a video stream that is displayed in a video.

For the benchmark version output has been disabled.

REFERENCES

1. **Haghbayan, M.**, **A. Rahmani**, **A. Y. Weldezion**, **P. Liljeberg**, **J. Plosila**, **A. Jantsch**, and **H. Tenhunen**, Dark silicon aware power management for manycore systems under dynamic workloads. *In 2014 IEEE 32nd International Conference on Computer Design (ICCD)*. 2014. ISSN 1063-6404.

2. **Henessy, J. L.** and **D. Patterson**, *Computer Architecture: A Qunatitative Approach*. Morgan Kaufman, Elsevier, 2012.

3. **Isci, C.**, **A. Buyuktosunoglu**, **C. Cher**, **P. Bose**, and **M. Martonosi**, An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget. *In 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-39 2006), 9-13 December 2006, Orlando, Florida, USA*. 2006. URL https://doi.org/10.1109/MICRO.2006.8.

4. **Li, S.**, **J. H. Ahn**, **R. D. Strong**, **J. B. Brockman**, **D. M. Tullsen**, and **N. P. Jouppi**, Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures. *In 42st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-42 2009), December 12-16, 2009, New York, New York, USA*. 2009. URL https://doi.org/10.1145/1669112.1669172.

5. **Reda, S.** and **A. Belouchrani**, Blind identification of power sources in processors. *In 27th Design, Automation, and Test in Europre (DATE)*. 2017.

6. **Shioya, R.**, **R. Takami**, **M. Goshima**, and **H. Ando** (2016). FXA: executing instructions in front-end for energy efficiency. *IEICE Transactions*, **99-D**(4), 1092–1107. URL https://doi.org/10.1587/transinf.2015EDP7316.