Power models for multicore processor simulators with multiple levels of abstraction

Author: Josep Triviño Valls

Degree in *Informatics Engineering*
Specialization in *Computer Engineering*
Facultat d'Informàtica de Barcelona (FIB)
Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

Director:
Miquel Moretó
Department:
BSC-CNS

Co-director:
Marc Casas
Department:
BSC-CNS

Deponent:
Mateo Valero
Department:
Computer Architecture

UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC) - BarcelonaTech
BARCELONA SUPERCOMPUTING CENTER (BSC-CNS)

June 30, 2015

Abstract

Nowadays, the power consumption is one of the biggest problems of the supercomputing centers because of the big consumption those have. In this project, the issue has been addressed and I have tried to find a way to reduce it. A good manner to reduce it, is to study the energetic behavior of the applications and adapt them to our hardware. In order to do this, we can use different tools, for example, an architecture simulator to obtain the hardware statistics or a framework to obtain power information.

In this project I have created a tool capable of generating power models for different applications given a determined hardware. With this tool, it is possible to obtain the total consumption of any application and the different components involved in the execution. Moreover, we can observe the consumption of the different application tasks and how it is distributed along the execution. It is possible to use this information to optimize any application in terms of power consumption or to find the optimal hardware in order to obtain the best performance.

Resum

Avui en dia, el consum energètic és un dels grans problemes dels centres de supercomputació degut al gran consum que tenen. En aquest treball s'ha adreçat el problema i s'ha intentat buscar algun mètode per tal de reduir-lo. Una bona manera de reduir aquest consum és estudiar el comportament energètic de les aplicacions i adaptar-les al nostre hardware. Per tal d'estudiar aquest comportament, es poden utilitzar diferents eines, com per exemple, simuladors d'arquitectura per tal d'obtenir estadístiques hardware o utilitzar un framework per tal d'obtenir informació energètica.

En aquest projecte he creat una eina capaç de generar models energètics per diferents aplicacions donat un hardware determinat. Amb aquesta eina, es pot obtenir el consum energètic total de qualsevol aplicació i dels diferents components involucrats en l'execució. A més a més de poder observar el consum de les diferents tasques d'una aplicació i com aquest està distribuït al llarg de l'execució. De manera que es pot utilitzar aquesta informació per optimitzar qualsevol aplicació en termes de consum energétic o buscar el hardware òptim per extreure'n el màxim rendiment.

Resumen

Hoy en día, el consumo energético es uno de los grandes problemas de los centros de supercomputación debido al gran consumo que tienen. En este trabajo se ha tratado el problema y se ha intentado buscar algún método para reducirlo. Una buena manera de reducirlo es estudiar el comportamiento energético de nuestras aplicaciones y adaptarlas a nuestro hardware. Para estudiar este comportamiento, podemos utilizar diferentes herramientas, como por ejemplo, simuladores de arquitectura para obtener estadísticas hardware o utilizar un framework para obtener información energética.

En este proyecto he creado una herramienta capaz de generar modelos energéticos para diferentes aplicaciones dado un hardware determinado. Con esta herramienta, se puede obtener el consumo energético total de cualquier aplicación y de los diferentes componentes involucrados en la ejecución. Además de poder observar el consumo de las diferentes tascas de una aplicación y como este está distribuido al largo de la ejecución. De manera que podemos utilizar esta información para optimizar cualquier aplicación en términos de consumo energético o buscar el hardware óptimo para conseguir el máximo rendimiento.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

The project is a TFG (final degree project) of the Facultat d'Informàtica from Barcelona and has been developed in collaboration with Barcelona Supercomputing Center (BSC) inside the RoMoL team (Riding on Moore's Law) which is a 5-year project funded by an European Research Council (ERC) Advanced Grant.

The power consumption of the hardware has always been a headache for the supercomputing centers and the different hardware companies. In order to reduce this power consumption we have to know how it is distributed during the application execution or which are the most power consuming components. This projects aims to create a tool capable of observing the power consumption using different applications from BSC.

Nowadays, there are different multicore processor simulators which can be used to collect information from an application running on them. With this statistics we are able to check the behaviour of the hardware which is being simulated or figure out its most time consuming parts (dependencies, cache misses...) and improve them. With these kind of statistics we can go further and use them to obtain more information, for example power models.

The main idea of this project comes out from McPAT[15], a state-of-the-art framework which can create power models for a program with the hardware statistics that are obtained by a simulator. With the power models we can observe our application behaviour in terms of power. This means that with this knowledge we should be able to observe the power consumption of the hardware components involved in the application's execution.

Profiling tools allow us to obtain the hardware statistics by task and how they are distributed along the execution. As we have a framework which allow us to compute the power consumption given hardware statistics, we can also compute how it is distributed during the execution.

The selected simulator is TaskSim [19], a trace-driven cycle-accurate simulator that is still being developed at BSC. The profiling tool is Paraver[8], a tool that has been also developed at BSC. We are going to see them more accurately on the state of art section.

This project is motivated by the fact that power consumption is nowadays a big problem that it is needed to be solved. Creating a tool capable of showing the power consumption of the hardware components and applications is a good contribution to the cause.

1.2 Stakeholders

This section aims to describe all the people involved in the project, directly or indirectly.

1.2.1 Developer

The person in charge to carry out the project, is the most important one because he is the only one. In this case I have been the developer.

1.2.2 Director, co-director and support

These are the persons in charge to help and guide the developer and supervise that the objectives are being accomplished. In this project the director and co-director have been Miquel Moretó and Marc Casas respectively.

1.2.3 RoMoL team

As I have commented, the project has been done inside the RoMoL team, which is a research group in BSC involved in the research for new computer architectures and parallel software. There are some researchers that are using TaskSim and Paraver in their research, this project can allow them to have more tools in order to obtain more information from their applications and the hardware used.

1.2.4 TaskSim developers

The TaskSim developers can be interested in this project in order to integrate it inside the TaskSim project. Moreover, having other people working with TaskSim has helped to find bugs.

1.2.5 Users

Nowadays the only persons who can use TaskSim are the members of BSC, so they are the users of the project and the main beneficiaries as well. Once TaskSim has been realised, everybody interested in the project can become a user.

1.3 Document structure

The document can be divided in two different parts. The first part is where all the project management has been explained, including the budget with all the costs, the scope with the objectives, methodology and the tools and finally the sustainability. The second part is the technical report. There I have explained what I have done and the different results obtained. Finally I have added the project conclusions and the future work to do.

Chapter 2

State of art

2.1 Computer architecture simulators

A computer architecture simulator is a piece of software which is used to predict the behaviour of hardware models given an input[10]. Architectural simulators have different purposes, per example:

- Compare different hardware designs without having physically the hardware.
- Obtain detailed performance metrics.
- Debug the code in real-time.

The simulators can be classified depending on the context:

- Micro-architecture or full-system simulators. A simulator can model the behaviour of the whole computer system (processor, a memory system) or can be more limited and focus in a specific part.
- Functional or performance simulators. Performance simulators try to reproduce the performance features of the hardware (when it is done) while functional simulators try to be as similar as possible to component's function (what it is done).
- Trace-driven or execution-driven simulators. Traces are prerecorded streams of instructions with fixed input. Execution-driven simulators allow dynamic change of instructions to be executed depending on different input data.

2.1.1 Gem5

Gem5 simulator[2] is a modular discrete event driven computer system simulator for computer system architecture research, capable of evaluate different systems. This simulator can simulate different CPU models, system execution modes and memory system models.

The gem5 simulator[2] is the union of two simulators, each one with different functions. The first one is called M5[3] and provides a highly configurable simulation framework, focused on the network simulation and different CPU models. GEMS[16], a general execution-driven multiprocessor simulation, complements these features with a detailed memory system simulation, including support for multiple cache coherence protocols and interconnect models.

2.1.2 TaskSim

The simulator used in this project is TaskSim[19] which is still under-development. We can say that TaskSim is a state-of-art trace-driven computer architecture simulator designed for:

- Architecture exploration of multicore multiprocessors with large numbers of cores.
- Research on parallel programming models.

TaskSim arose from the problem that the current simulators(for example gem5) have with the increasing gap performance between simulation and real execution. TaskSim allows different levels of abstraction to deal with this problem. The highest level of abstraction, burst mode, is useful for scalability studies to hundreds of cores. The lower-level abstractions, memory mode, provides accurate simulation at a greater level of detail.

2.2 Profiling tools

In order to obtain more information about our applications we can use profiling tools or profilers. Profiling is a form of dynamic program analysis that measures our applications in terms of memory, time or usage of particular instructions... We can use this information to optimize our applications. Profiling tools use different techniques to obtain data, for example hardware interrupts, code instrumentation or performance counters.

2.2.1 Paraver and Extrae

Paraver[8] is a performance analyzer based on traces with a great flexibility to explore the collected data by visual inspection. The principal features of Paraver are:

- Detailed analysis of program performance.
- Customizable semantic of the visualized information.
- Concurrent comparative analysis of different traces.

Paraver has been proofed to be very useful for performance analysis studies, giving much more details about applications behaviour than most performance tools.

In order to generate Paraver traces, Extrae[5] is required. Extrae is the package created to generate Paraver traces for a posterior analysis. Extrae is a tool which translates all internal events into a Paraver events, this way, Extrae gathers all the information regarding the application performance. Extrae is configured through a XML file.

2.3 McPAT

McPAT (Multicore Power, Area, and Timing)[14] is the first integrated power, area, and timing modeling framework for multithreaded and multicore/many-core processors. It is designed to work with a variety of performance simulators over many technology generations. McPAT allows the user to specify low-level configuration details. Also includes models for the components of a complete chip multiprocessor, including in-order and out-of-order processor cores, networks-on-chip, shared caches, and integrated memory controllers.

The McPAT idea comes out from a tool called Wattch[4] which was created in the year 2000. This tool is able to compute the CPU power consumption in order to make power analysis of the applications. However, several factors drive the need for new tools to address changes in architecture and technology, for example the need to accurately model multicore and many-core architectures or the need to model and evaluate power, area, and timing simultaneously. And McPAT addresses these challenges.

2.4 Related work

As this project is specific for a simulator, there is not any project involved in the same simulator, but there are similar projects for other simulators, for example a project that has focused in gem5 and McPAT[12] instead of TaskSim and McPAT. We can say that the objectives of both projects are the same (obtain power models) but the code part is different because of the input and output of each simulator. Although the code, the tools used in both projects are similar and also the problems that happened. Before starting the project, I made a thoroughly evaluation of the project in order to learn about the used tools and the problems which occurred in the similar project mentioned above.

There are other simulators which have integrated McPAT inside of the program, for example the Graphite multicore simulator[17]. This simulator, additionally of giving the statistics of the hardware simulation, gives the power consumption of the processor. Graphite uses McPAT for core and cache power modeling and DSENT[20] for network power modeling. Graphite has been created by Massachusetts Institute of Technology (MIT) and it has been designed for exploration of future multicore processors containing thousands of cores. It provides high performance for fast design space exploration and software development.

Moreover, there are other simulators like Graphite, for example Sniper simulator[9]. Sniper multicore simulator has integrated McPAT inside the structure too. Sniper simulator is based on the interval core model[13] and the Graphite simulation infrastructure.

Chapter 3

Project scope

This project aims to generate power models and study the results in order to know how the power consumption is distributed between the different hardware components or the different application tasks. In order to generate them, it is necessary to create a tool and different benchmarks in order to test it. To do this, I have divided the project scope in different steps:

1. The first part is based only on the TaskSim statistics. TaskSim nowadays just provides hierarchical memory and burst statistics, so the power models are based only on them. To obtain the power models, a bridge between the simulator and the framework has been created. This bridge or tool joins all the simulator information (the output and the simulation configuration file), so that a summary is presented at the end of the execution and parsed for the McPAT framework.
2. Create different benchmarks in order to generate different power models. Explore different processor specifications (different sizes of DL1 and L2) for each benchmark and check the optimal combination for each one.
3. Use Paraver in order to observe how power consumption is distributed along the execution. Represent it in different plots and study the results.

3.1 Objectives

The main objectives of this project are:

- To create a tool to parse information from TaskSim to McPAT and create power models.

- To study the behaviour of the hardware components in terms of power consumption.
- To create a Paraver configuration file to plot the TaskSim simulations.

3.2 Requirements

The requirements of this project are:

- To develop the tool using open-source tools.
- The tool has to execute McPAT without errors, has to be easy to use, consistent, fast and adaptable to user input/output.
- The code has to be reliable, maintainable, well-structured and readable.

3.3 Methodology

In order to achieve all the objectives, I have used the scrum methodology. The scrum methodology has allowed me to split the project in different goals and focus on an specific goal for 2-4 weeks (in scrum terminology is called sprint). Although in this project I have not had meetings every day, there have been meetings once a week.

3.4 Tools

The tools which have been used in this project can be categorized as hardware tools or software tools.

3.4.1 Hardware

The hardware tools that I have used to carry on this project have been the indispensable ones to develop the code and to launch the executions. I have used a desktop in BSC to code the scripts. I have also used a virtual machine on MareNostrum III to launch the TaskSim executions. MareNostrum III has been really useful because I could parallelize the different executions.

3.4.2 Software

The project has been developed in a Linux distribution, OpenSuSe for being precise. The Linux distribution has all the tools to compile, execute or edit. Apart from the usual tools, I have used the tools detailed in the state-of-art chapter(TaskSim, McPAT and Paraver). Moreover, I have used other tools such as:

3.4.2.1 Jinja2

I have used Jinja2 to parse information between a Python script and a xml template. Jinja2 is a modern and designer-friendly templating language for Python.

3.4.2.2 Gnuplot

In order to generate heat maps I have used Gnuplot, a portable command-line driven graphing utility for Linux and other SO.

3.4.2.3 GREASY

GREASY is a tool developed by BSC which is able to run in parallel a group of different tasks, schedule them and run them using the available resources. This is really useful when we have to execute different tasks in MareNostrum III and we do not want to execute them sequentially.

3.5 Revision control

There have been two types of revision control in this project:

- In order to control all the application versions, a version control has been used to be able to recover the data or version in case a black down had happened or an error had appeared in a late version.
- The project has been supervised and reviewed once a week by the project director.

3.6 Validation methods

In order to check if the power models have been created properly and the results make sense, I have used an example of McPAT configuration file given by HP. These configuration files are extracted from real processors specifications. I have filled the example file with the information from TaskSim and I have generated the power model. I have compared my power model (generated from my template) with the power model generated by the example. This comparison has allowed me to generate power models that are similar to the power models generated by a real processor.

I have also tested my tool with different verified benchmarks in order to check if the behaviour of the power consumption was correct.

3.6.1 Benchmarks

I have used different benchmarks for this project in order to generate different power models and to check the correctness of the tool.

- **Vector-Matrix multiplication:** Useful benchmark to check how the code can affect the power consumption (going throw a matrix by rows or by columns).
- **Copy vector:**
  - **Increasing vector size:** Application test that considers different vector sizes in order to show how the power consumption increases also in the different components.
  - **Increasing the memory footprint of DL1 and L2:** Increase the number of misses in both caches and how that affect the power consumption.
- **PARSEC:** Different tests from a benchmark suite composed of multithreaded programs.

3.7 Risk management

There have been different risks on this project.

- TaskSim is a tool which is still being developed, this means that it has been constantly evolving during the project so more functions (extra statistic information) have been added. Besides, the application has not been a stable version so problems and bugs have been found.

- Although I have had almost all the applications in a local desktop, I have used Marenostrum III to execute TaskSim or to use some tools which are only installed there. If an Internet black down had happened, I would not have been able to do so.

- McPAT is a really complex framework, the input of the program is huge and has a lot of variables in order to configure it. The simulator does not provide all the necessary information to configure McPAT, so in some fields it was necessary to make some approximations. These approximations have been taken from real processors to be as accurate as possible.

Chapter 4

Project management, budget and sustainability

4.1 Project management

4.1.1 Planning

In the following chapter we are going to see the overview of the planning. The project has been done during the spring semester of 2015. It started on February the first and has been finished at the end of June. I have disposed of 5 months to work in it. I have dedicated 5 hours a day from Monday to Friday which makes a total of 500 hours.

I have divided the project in three main tasks:

1. **Work environment:** Set up the entire work environment in order to start the project with all the tools working.
2. **Coding:** Code the tool in a way that satisfies all necessities and achieves all the objectives. Besides, I had to create power models and study the results.
3. **Documentation:** Write down the work done in order to be presented as a thesis.

Finally, I have split all the main tasks in sub-tasks in order to detail every step clearly:

4.1.1.1 Work environment

1. **Set up work environment:** Get into the project, talk about it with the project director and introduce myself to the work team. Other tasks such as picking up the key and the credentials can also be included here.

The resources used have been the PC and a junior developer.

2. **Learn how to use the work environment:** Configure the work environment. Using MareNostrum III is not a trivial matter because it works with a queue system which has to be learned before using it. The executions have to be sent to a job queue using specific commands and there are some papers which have to be read in order to achieve the knowledge to use it.

   The resources used have been the PC and a junior developer.

3. **Configure the desktop:** Configure the desktop in order to make the work easier, for example, install another console or editor to which I am more used.

   The resources used have been the PC and a junior developer.

4. **Install and learn McPAT:** Install McPAT from the HP website. Read the papers in order to know how it works. A member of the team has taught me more features about it. I have also studied the input of the program in order to learn how to run it.

   The resources used have been the PC, a junior developer and a senior developer.

5. **Install and learn TaskSim:** Install TaskSim from the repository. A member of the team has showed me how TaskSim works because there is not much information available. Besides, a more precise study about the output of the program has been done.

   The resources used have been the PC, a junior developer and a senior developer.

6. **Learn python:** Learn how to code in python and search for a tool to parse information from TaskSim to a xml file.

   The resources used have been the PC and a junior developer.

7. **Learn about Paraver:** Learn how Paraver works, how to get information from it and how to create configuration files.

   The resources used have been the PC and a junior developer.

4.1.1.2 Coding

8. **Create the Template:** Create a xml template file with the structure needed to launch McPAT.

   The resources used have been the PC and a junior developer.

9. **Code the script:** Code the tool using the python script language. The tool generates the input of McPAT given the TaskSim statistics and creates the power model.

   The resources used have been the PC and a junior developer.

10. **Debug:** Debug the application in order to find bugs or polish some template values to generate more consistent power models.

    The resources used have been the PC and a junior developer.

11. **Create power models:** Create different benchmarks in order to generate power models. Study and represent the different results.

    The resources used have been the PC and a junior developer.

12. **Integrate Paraver with the script:** Add the information showed by Paraver to the project.

    The resources used have been the PC and a junior developer.

4.1.1.3 Documentation

13. **GEP:** Write down the introduction of the project through 7 deliverables at the beginning of the project.

    The resources used have been the PC and a junior developer.

14. **Write the thesis:** Write down all the work done and the results. The result of this task has been added to the paper obtained from GEP.

    The resources used have been the PC and a junior developer.

15. **Defense the project:** Create the slides to defense the project, prepare the defense and finally defense the project in front of the tribunal.

    The resources used have been the PC and a junior developer.

4.1.1.4 Task dependencies

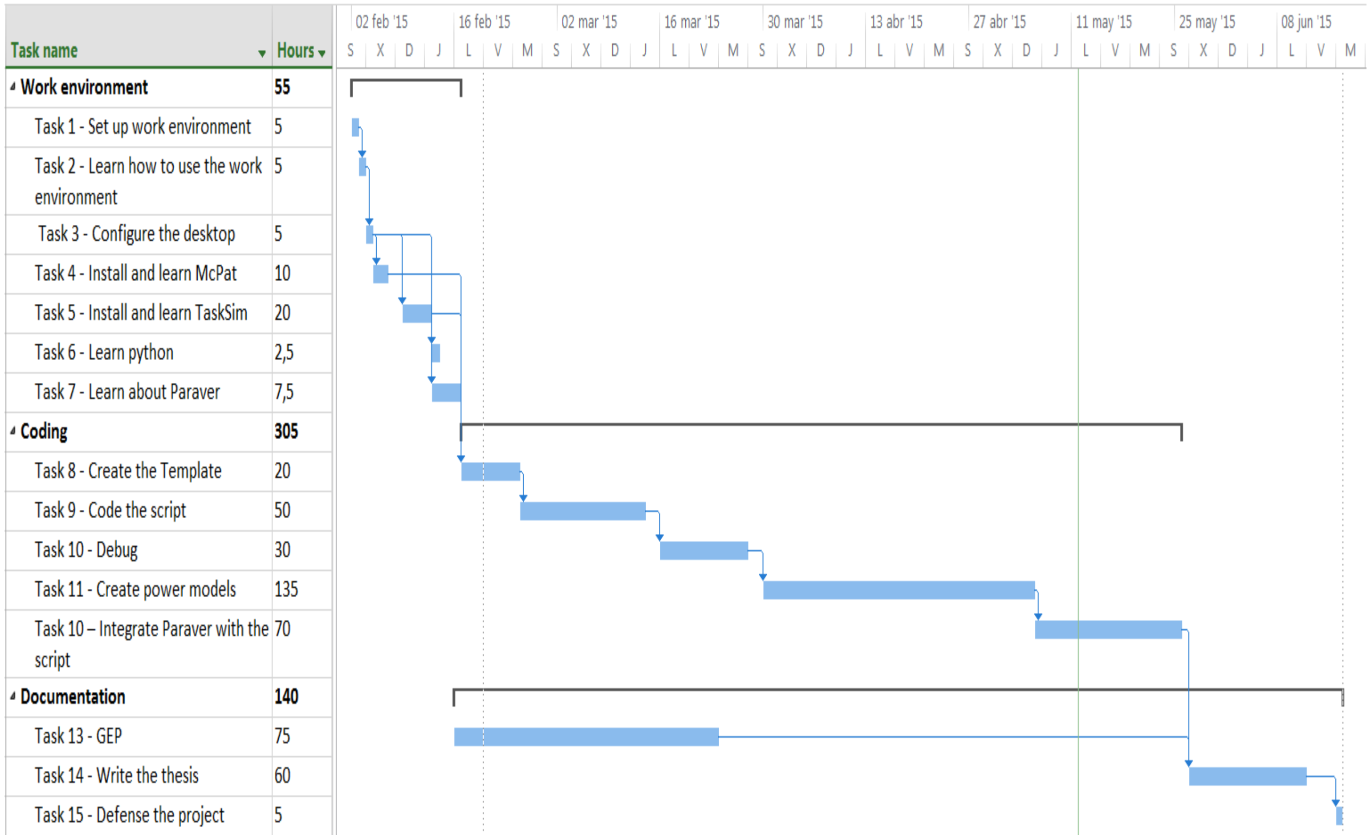
Sub-tasks have the following dependencies(dependencies are represented as $\leftarrow$) :

T1 $\leftarrow T2 \leftarrow T3 \leftarrow T4, T5, T6, T7 \leftarrow T8 \leftarrow T9 \leftarrow T10 \leftarrow T11 \leftarrow T12$.

T12, T13 $\leftarrow T14 \leftarrow T15$.

There are some tasks which could have been done at the same time, but because of resources reasons (the project has been a one person project) they could not have been done in parallel.

4.1.1.5 Gantt chart



Task name	Hours
Work environment	**55**
Task 1 - Set up work environment	5
Task 2 - Learn how to use the work environment	5
Task 3 - Configure the desktop	5
Task 4 - Install and learn McPat	10
Task 5 - Install and learn TaskSim	20
Task 6 - Learn python	2,5
Task 7 - Learn about Paraver	7,5
Coding	**305**
Task 8 - Create the Template	20
Task 9 - Code the script	50
Task 10 - Debug	30
Task 11 - Create power models	135
Task 10 – Integrate Paraver with the script	70
Documentation	**140**
Task 13 - GEP	75
Task 14 - Write the thesis	60
Task 15 - Defense the project	5

Figure 4.1: Gantt chart with the time duration of each task

4.1.2 Alternatives and work plan

As in any project, I created an alternative plan before starting the project in order to deal with the different problems that could had happened during the project. The main objectives of the project had to be met in order to finish the project, so cutting the objectives was not contemplated as a solution. In case that a huge problem appeared, my plan was to work some extra hours in order to deal with it, for example, instead of assigning 5h a day, increment these hours to 7 or 8.

I had also an alternative plan regarding the simulator. If there had been any problems, I would have changed it for another one, for example Gem5. Other tools like Paraver or McPAT were used in similar projects, so that their correct working was demonstrated.

The alternative plan has worked for me. Actually, 500 hours have been enough to finish the project on time, including the dealing of eventual problems. Moreover, the simulator has worked as expected.

4.2 Budget control

In order to check the viability of the project, I computed the total cost of the project before starting. In this way, I was able to see if the budget was enough to cover all the expenses.

4.2.1 Budget

In this chapter we can see the costs of all the resources involved in the project. The resources have been split in five types:

- **Human:** The cost of the staff involved in the project.
- **Software:** The cost of the software tools used in the project.
- **Hardware:** The cost of the hardware resources needed to carry out the project.
- **General expenses:** Involves the indirect resources used in the project.
- **Taxes:** The taxes paid during the project.

4.2.1.1 Human resources

The biggest part of the cost of human resources in the project has been the junior developer. Also we have to take into account that sometimes a senior developer has been required to help the junior developer as well as the hours that the project director has spent supervising the project (half an hour a week).

Human resource	Task	Hours(h)	Cost(€/h)	Total(€)
Junior developer				
	1-Set up work environment	5	10	50
	2-Learn how to use the work environment	5	10	50
	3-Configure the desktop	5	10	50
	4-Install and learn McPAT	10	10	100
	5-Install and learn TaskSim	20	10	200
	6-Learn python	2.5	10	25
	7-Learn about Paraver	7.5	10	75
	8-Create the Template	20	10	200
	9-Code the script	50	10	500
	10-Debug	30	10	300
	11-Create power models	135	10	1350
	12-Integrate Paraver with the script	70	10	700
	13-GEP	75	10	750
	14-Write the thesis	60	10	600
	15-Defense the project	5	10	50
Senior developer				
	Teaching	10	20	200
Project director				
	Supervising the project	10	30	300
Total cost				5500

Table 4.1: Costs of human resources

4.2.1.2 Software

All the software needed to carry out the project has been free (from the OS which is OpenSuse up to the more specific software such McPAT), so the total cost of the software has been 0.

4.2.1.3 Hardware

As we could see on the previous chapter, I have spent 500 hours in the project. All the hours have been done in the same hardware (BSC desktop). I have not computed the work done in Marenostrum III because it is a public resource which can be used free by the members of the team. To compute the deprecation of the hardware, I have chosen 240 working days during the whole year. I have used the following equation:

$$Deprecation = Time \times \frac{Cost}{UsefulLife(y)} \times \frac{1(y)}{240(d)} \times \frac{1(d)}{8(h)} \tag{4.1}$$

The results can be seen in table 4.2.

Hardware resource	Cost(€)	Useful life(y)	Time(h)	Total(€)
Desktop	900	4	500	58.6
Mouse	8.95	4	500	0.58
Screen	84.95	4	500	5.53
Keyboard	9.95	4	500	0.65
Total cost				65.36

Table 4.2: Costs of hardware resources

4.2.1.4 General expenses

There have been two kinds of expenses here:

- **Indirect costs**

  The most costly part here has been the electricity. Nowadays a kwh costs 0.14€ in Spain. There have been only two hardware resources that can be used to compute electricity cost, these hardware resources are the desktop and the screen.

Hardware resource	Power(W)	Price(€/kwh)	Hours(h)	Total(€)
Desktop	0.4	0.14	500	28
Screen	0.03	0.14	500	2.1
Total cost				30.1

Table 4.3: Costs of electricity

- **Unforeseen costs**

  To deal with eventual problems and unexpected events which could had occurred, I decided to increment the total cost in a 5%. In this way, If something had gone wrong (a hardware resource breaks down or a payment program) I would have had some margin.

4.2.1.5 Taxes

The revenue tax in Spain is 21%.

4.2.1.6 Total cost

The total cost can be seen in table 4.4.

Resource type	Total(€)
Human	5500
Hardware	65.36
Software	0
General expenses (without contingency)	30.1
Total cost (without contingency and without taxes)	5595.46
Total cost (with 5% of contingency and without taxes)	5875.23
Taxes (21% of the total cost)	1233.80
Total cost	7109.03

Table 4.4: Costs summary

4.2.2 Control management

At the end of each task, in order to calculate the derivation, I compared the planned cost and the real cost. If the derivation of the task cost was bigger than expected then I made a more accurately study about what had happened during the task in order to avoid that the same issues happen again.

As 500 hours have been enough to finish the project and there has not been any problem involving the hardware, there has not been any derivation of the planned budget.

4.3 Sustainability

It is necessary to evaluate the project's impact on society and in which way it is going to improve the people's quality of life. To study the sustainability, I have focused on economic, social and environmental points of view. In each point of view, I have obtained a score which is computed following the Socratic Method.

4.3.1 Economic

- As we could see in previous chapters, all the costs have been assessed and properly explained.
- In order to deal with unexpected issues, a contingency percent was added to the total cost.
- The project could have been done with less time by an experienced developer but the cost per hour of an experienced developer would has been higher than the present developer. So the total cost of the project probably would has been increased rather than decreased. The resources would not have varied.
- The most important parts have been the most time consuming parts as we can see in the Gantt diagram.
- The project has been conducted in cooperation with BSC and it is linked to a bigger project.

4.3.2 Social

- The present social situation of the country is correct and the political situation stable. The sector which involves the project is healthy and growing.
- The project is not going to influence the social and political situation in any way.
- The project was a need from BSC to have a tool to create power models.
- The project is neither going to improve consumer's quality of life, nor is it going to influence the consumer. This project has just added a new feature to the BSC project.
- No collective is going to be harmed by the project.

4.3.3 Environmental

- The resources involving environmental impact have been electricity and the desktop components.
- The environmental impact of the activity if I had not conducted the project would has been the same.
- The desktop, mouse and keyboard have been reused from older projects.

- The same as in the previous point, all the hardware resources involved in the project are going to be reused in future projects.
- The project has used manufactured products (desktop, mouse, keyboard) which are made from raw resources extracted from problematic countries(coltan). These resources have been extracted in poor working conditions.
- The only pollution generated during the project has been for the production of electricity.
- The project is about the power consumption of our programs, If we know the consumption of our programs, we can improve them to be energetically more efficient, so the project might influence the energetically footprint by reducing it.

4.3.4 Sustainability evaluation

Sustainable?	Economic	Social	Environmental
Planification	Economical viability	Improvement in life quality	Resource analysis
Valuation	5	8	9
Results	Final cost versus prediction	Social environment impact	Resource usage
Valuation	10	9	9
Risks	Adaptation to changes in scenario	Social damage	Environmental damage
Valuation	0	0	0
Total valuation	50		

Table 4.5: Sustainability matrix

Chapter 5

Experimental setup

This chapters aim to explain how the different tools have been used and the setup used in the experiments.

5.1 TaskSim

The architecture simulator to run the tests and obtain the statistics has been TaskSim. The main information about TaskSim has been explained in previous chapters, but there is still information which has to be cleared up.

TaskSim is a trace-driven architecture simulator which means that works with traces. Before executing TaskSim, it is necessary to instrument the code (automatically done by the compiler Mercurium[6]) and to generate the trace with Nanox[7]) (Nanox provides a basic tracing infrastructure to ease the task of creating and reading different kinds of traces). The traces are a collection of events (run-time events, memory accesses, ...) which are sequentially stored. Once we have the trace, we can execute TaskSim with the trace and a configuration file in order to simulate the application and obtain the statistics.

The statistics generated by TaskSim can have different levels of abstractions. In the TaskSim configuration file, we can select which level we wish. We can select burst mode or memory mode. The burst mode only generates CPU bursts and memory mode generates CPU bursts and simulates the memory accesses when available.

Nowadays, TaskSim does not provide information about the entire processor, so I have not computed the power consumption of a processor with all the components involved in it. I have selected those components that TaskSim simulates. These components are mainly the components involved in memory accesses.

5.1.1 Configuration file

As we have seen in the previous section, there is a configuration file in order to configure TaskSim with the processor specifications. The configuration file is composed by different parts such as DRAM, MMU, Cache... Each part is a component that can be configured by the user.

The variables that I have used in this project and also to configure McPAT are the following ones:

- Number of cores.
- ROB size (Re-order Buffer).
- DRAM RAS size (Return Address Stack).
- Cache configuration (DL1 and L2).
  - Size.
  - Associativity.
  - Block width.
  - Latency.
  - Number of w/r ports.
- MSHR (Miss status holding registers).
- Virtual memory page size.

Moreover of this values, there are other important ones which I have not parsed to McPAT. These values have a direct impact to total number of cycles:

- Memory CPU commit rate.
- Memory latency.
- Processor idle cycles: How many cycles will be inserted whenever the scheduler cannot provide a task for a CPU.

I have included an example of TaskSim configuration file in appendix A.

5.1.2 Results file

The results file generated by TaskSim gives us all the information collected during the simulation. The information is given in an specific format. I have included an example of results file in the appendix B.

The information which I have extracted from the results file and I have used to create the power models can be seen below.

- Committed instructions.
- Total number of cycles.
- For each core:
  - Number of loads/stores.
  - For each L2 and DL1:
    - Number of write/read.
    - Number of hit/miss.

5.1.3 Additional information

I have used some additional information in order to configure McPAT that does not appear on the results file or configuration file. This information is used during the TaskSim execution and although it is not provided by the user, it is part of TaskSim infrastructure.

- **Frequency:** TaskSim works at 1GHz.
- **Cache policy:** The cache policy used by TaskSim is copy back.
- **Private or shared L2:** TaskSim allows using private L2 or shared L2. Nowadays, the selection has to be done through the TaskSim code. I have selected private L2 to do the tests.
- **Heterogeneous cores:** Each core differs from others.
- **Out of order cores:** TaskSim simulates out-of-order processors.
- **Additional cache levels:** TaskSim can incorporate as many cache levels (private or not) as the user wants. These cache levels can be private or shared. In this project, I have used two private cache levels (DL1 and L2).

5.2 McPAT

As I have explained in the previous chapters, McPAT can be considered a really complex framework which allows us to calculate the power consumption of a processor. As it has been a really important tool for the project, I have to explain how McPAT works in order to get a rough idea of what I have done with it.

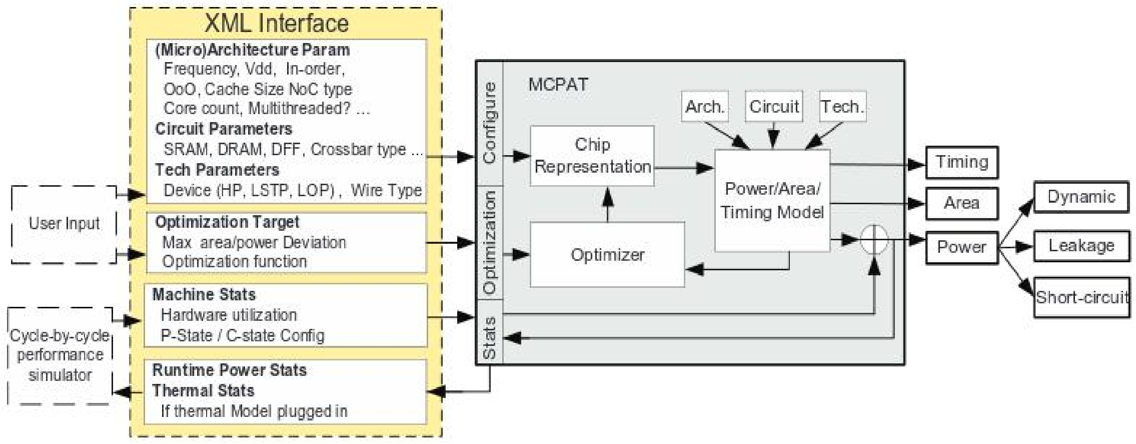


FIGURE 5.1: Block diagram of the McPAT framework

As we can see in the figure, given a configuration file, McPAT uses a sequence of steps to generate the timing, area and power (not just dynamic power but also static and short-circuit power) of a multicore processor. To generate all this information, McPAT models the chip technology, architecture and circuits.

5.2.1 Configuration file

In order to configure McPAT and compute the power, it is necessary to create a xml file. In the configuration file, McPAT allows us to specify the low-level configuration details of our processor. It also provides default values If we want to specify only high-level architectural parameters. The xml file has to contain all components of our processor from which we want to obtain the power consumption with McPAT.

We can see an example of McPAT configuration file in the appendix C. The example just contains the components that I have simulated.

5.2.2 Results file

For each component and sub-component (each component also has different sub-components), McPAT gives us the following statistics:

- **Area (mm^2):** Size of the component.
- **Peak Dynamic (W):** The absolute worst case power (maximum number of accesses).
- **Subthreshold Leakage (W):** The power wasted when a transistor that is supposedly in the off state actually allows a small current to pass between its source and drain.
- **Subthreshold Leakage with power gating (W):** The subthreshold leakage with a technique to reduce the power consumption.
- **Gate Leakage (W):** The current that leaks through the gate terminal.
- **Runtime Dynamic (W):** The power spent in charging and discharging the capacitive loads when the circuit switches state.

I have included an example of McPAT results file in the appendix D.

As the timing of different executions with different cache sizes or number of CPUs it is not equal, I have decided to compute the runtime dynamic energy for each test too. As I have the frequency in which the processor works and the total number of cycles (both things extracted from TaskSim) I can apply the formula below.

$$E(J) = P(W) \times T(s) = P(W) \times \frac{TotalCycles}{Frequency(Hz)} \tag{5.1}$$

5.2.2.1 Power gating

McPAT can report static power (subthreshold leakage) with power gating. Power gating is a technique to reduce the power consumption. The technique consist in shutting off the current to blocks of the circuit that are not in use. I have used static power with power gating instead of without it due to the fact that nowadays all the actual processors work with it and I have wanted to be as accurate as possible to a real consumption of a processor.

Usually, the static power with power gating is the half of static power without power gating. Here is an example:

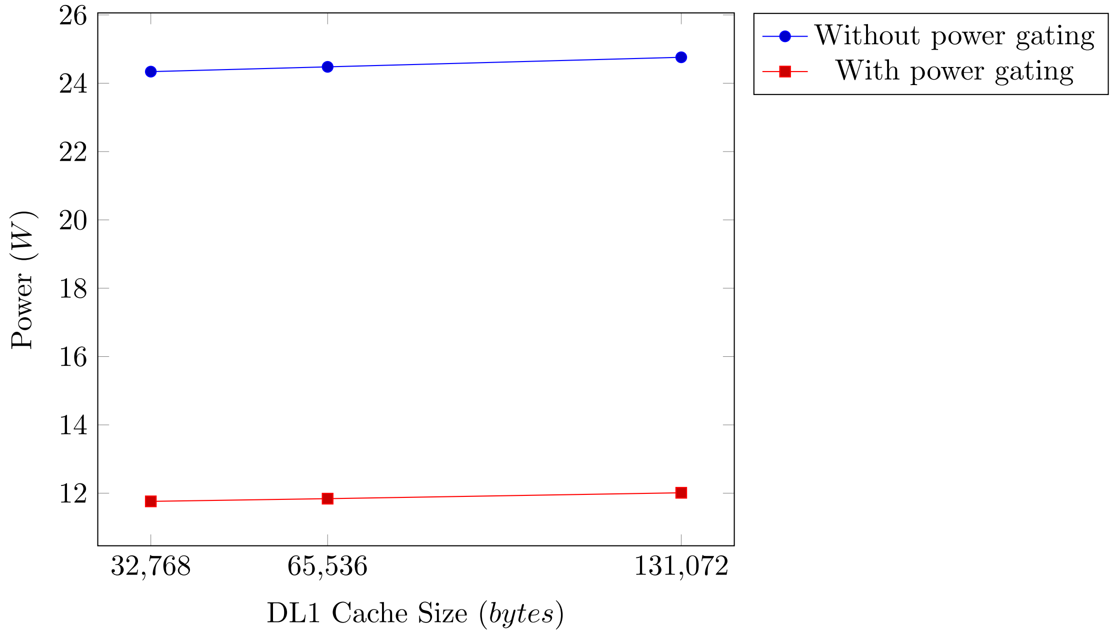


FIGURE 5.2: Static power comparison

5.2.2.2 Processor modeling

The main components and sub-components which TaskSim models and I have used can be seen below.

- Renaming Unit (RU)
- Memory Management Unit (MMU)
  - Itlb
  - Dtlb
- Execution Unit (EU)
  - ALU
  - Floating Point Unit
  - Register Files
  - Instruction Scheduler
- Instruction Fetch Unit (IFU)
  - Instruction Cache
  - Branch Target Buffer
  - Branch Predictor

  - Instruction Buffer
  - Instruction Decoder

- Load Store Unit (LSU)
  - Data Cache (DL1)
  - LoadQ
  - StoreQ

- L2 Cache

5.2.2.3 Cache modeling

In this project, I am going to focus on the power consumption of the memory components (Cache DL1 and L2). To compute the power model of a cache, McPAT uses a tool called CACTI[21]. CACTI is an integrated cache and memory access time, cycle time, area, leakage and dynamic power model. Cacti is in charge to compute the power consumption of data/tag cache array when it is read, searched or written.

In order to configure the cache, McPAT has the following variables:

- Capacity
- Block width
- Associativity
- Bank
- Throughput w.r.t core clock
- Latency w.r.t core clock
- Output width
- Cache policy

Also, for each cache it is needed:

- Number of write accesses.
- Number of read accesses.
- Number of read misses.

- Number of hit misses.

Besides, McPAT models the buffers involved in the cache, these buffers are:

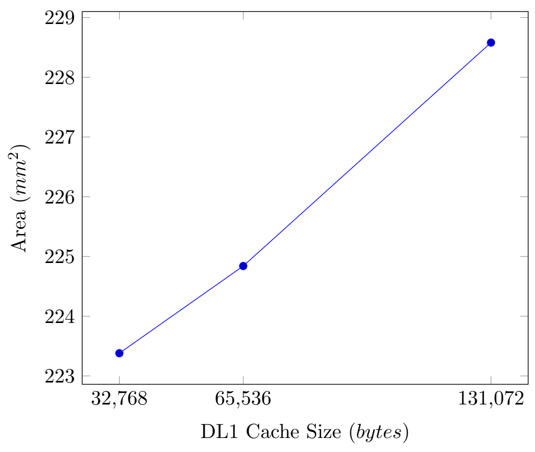
- MSHR buffer.
- Write back buffer.
- Fill buffer.
- Prefetch buffer.

All the elements affect the power consumption of the cache. For example, when the cache size is increased, the number of transistors increases too, so the area, peak dynamic and the static power rise. It also increases the cache dynamic consumption although we might be reducing the number of cache misses. The reason is because we are also increasing the cache access time and the number of transistors to read/write the tag.

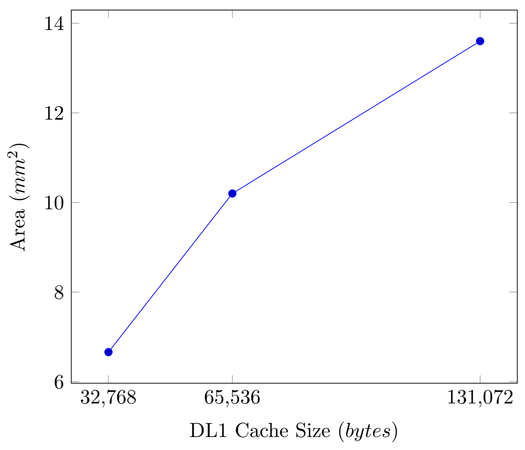
There are other variables which just affect the dynamic consumption, for example the variables involved in cache accesses.

I have done a test in order to check the behaviour of the cache when cache size is increased (All cache sizes have the same number of accesses).

First of all, how area is increased:



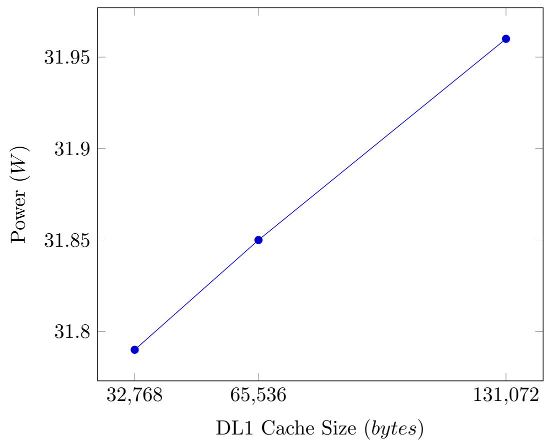
(A) Processor Total Area



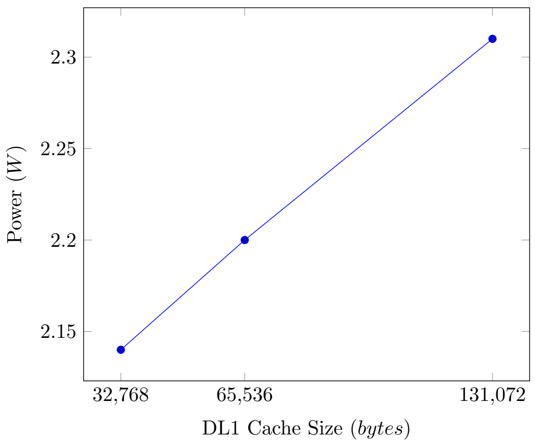
(B) DL1 Total Area

FIGURE 5.3: Area comparison

Secondly, how power consumption is increased:

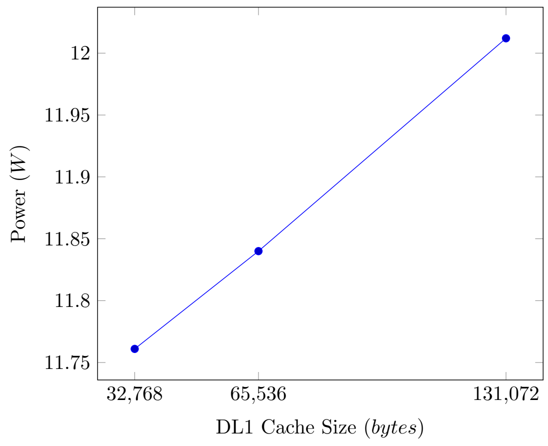


(A) Processor Peak dynamic

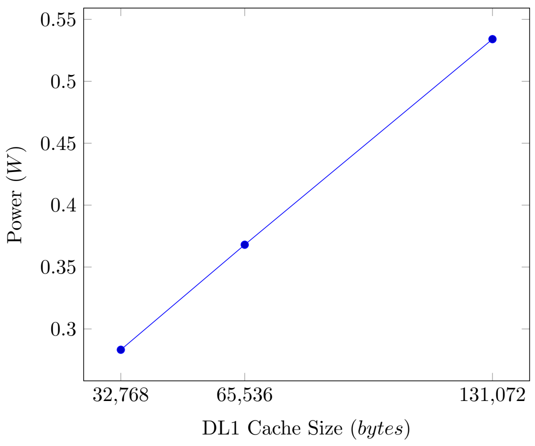


(B) DL1 Peak dynamic

FIGURE 5.4: Peak dynamic power difference

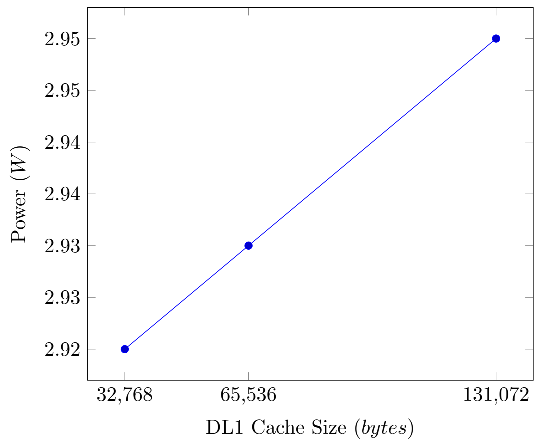


(A) Processor Static power

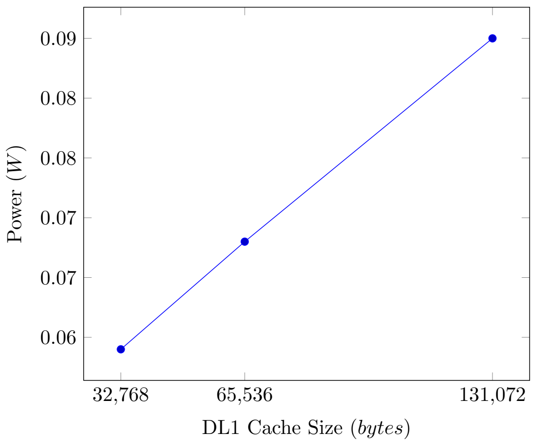


(B) DL1 Static power

FIGURE 5.5: Static power comparison

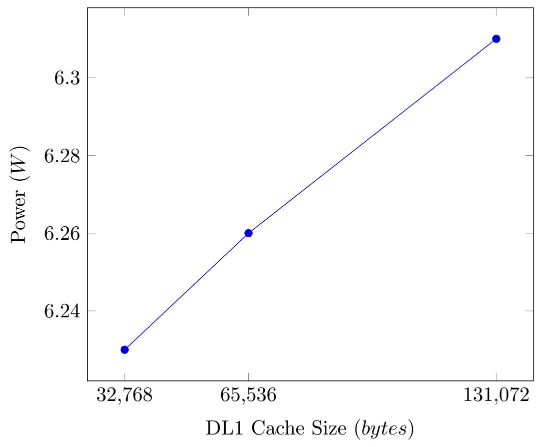


(A) Processor Gate leakage

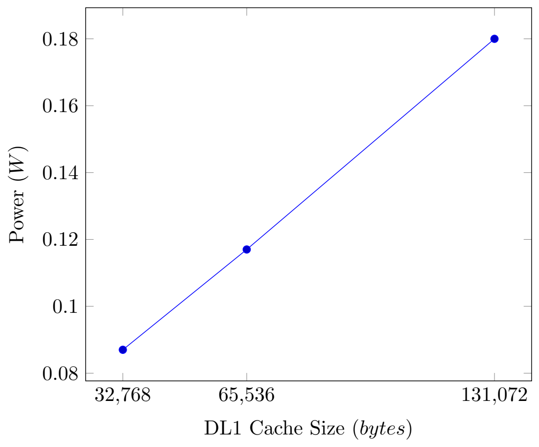


(B) DL1 Gate leakage

FIGURE 5.6: Gate leakage comparison



(A) Processor Dynamic power



(B) DL1 Dynamic power

FIGURE 5.7: Dynamic power comparison

5.3 TaskSim-McPAT script

The main coding part of this project came from creating a script to parse the information from TaskSim to McPAT. The script had to read the parameters from a TaskSim results file and configuration file, and generate a McPAT configuration file with all the information required to run it without problems.

First of all, I created a template with the required information to execute McPAT. The template contained the basic components to run McPAT without problems, moreover, I deleted the unnecessary components (network, cache directories, etc...).

After that, I created a python script to collect the information from TaskSim and store it in a two-level dictionary. The first dictionary level contained the data roots (core, configuration file or information of the whole system). The second level contained the data name.

In order to add the information to the template I used a templating language for Python called Jinja2. Jinja2 allowed me to generate a xml file with all the information from a python dictionary.

All the information I collected from TaskSim was not enough to run McPAT, so more information had to be added in order to run it. The biggest part of this information was about the processor and core configuration. For example, the processor technology used or the number of ALUs for each core.

McPAT has different configuration files extracted from real processor specifications. I used real processors configuration files to fill the unknown information from my template. The most part of the unknown information comes from an Intel Xeon.

5.4 Paraver script

With the TaskSim-McPAT script it is possible to know the total power consumption of our application and the components involved in the execution. TaskSim also has a feature which allows us to generate a Paraver trace. We can extract from Paraver trace all the cache accesses for task and calculate the power consumption for each task. I added this feature to the project, which allows us to observe the behavior of the different application tasks in terms of power and how the power consumption is distributed along the execution.

In order to do this, I have created a Paraver configuration file with the different cache accesses of the different cache levels, then I have multiplied the different cache accesses for the power consumption of a cache access (calculated by McPAT).

For each trace, the Paraver configuration file generates four different plots:

- Gradient color dynamic power consumption (W).
- Function line dynamic power consumption (W).
- Aggregated dynamic power consumption (W).
- Core burst.

Chapter 6

Evaluation

The results obtained from the different benchmarks have been explained in the following chapter.

6.1 Vector copy

In all the following tests I have used a vector copy code. Vector copy has allowed me to have for each vector iteration, a write and a read access.

6.1.1 Increasing vector size with private L2 cache

I have created this benchmark to check the behaviour of the power consumption from the different components of a processor when the vector size is increased.

I have used the code below.

```
void computo (int *x,int *y, int times){
    int k, i;
    for (k = 0; k < times ; k++ )
        for (i = 0 ; i < vecELEMENTS ; i++ )
            x[i] = y[i];
}
```

Listing 6.1: Increasing vector size code

I have increased the vecELEMENTS size gradually, from 32768 elements to 262144 (each element is an integer of 4 bytes). I have done the same test for 1 core, 4 cores and 8 cores. Moreover, I have selected a DL1 cache size of 128KB and a L2 cache size of 1MB.

I have also transformed the power consumption into energy consumption because the difference of time between 1, 4 and 8 cores is different. The energy consumption for the whole processor and the different components involved can be seen below.

VecSize	Total Energy	IFU	RU	LSU	MMU	EU	L2
32768	0.112	0.011	0.013	0.034	0.008	0.044	0.001
65536	0.220	0.022	0.026	0.068	0.016	0.086	0.002
131072	0.444	0.044	0.053	0.137	0.033	0.174	0.003
262144	1.412	0.112	0.202	0.298	0.126	0.666	0.007

TABLE 6.1: Increasing vector size with 1 core

VecSize	Total Energy	IFU	RU	LSU	MMU	EU	L2
32768	0.144	0.012	0.019	0.036	0.012	0.063	0.001
65536	0.275	0.024	0.036	0.071	0.023	0.119	0.002
131072	0.569	0.050	0.076	0.143	0.047	0.250	0.003
262144	1.752	0.127	0.265	0.314	0.166	0.873	0.007

TABLE 6.2: Increasing vector size with 4 core

VecSize	Total Energy	IFU	RU	LSU	MMU	EU	L2
32768	0.196	0.015	0.029	0.038	0.018	0.095	0.001
65536	0.367	0.029	0.053	0.075	0.033	0.175	0.002
131072	0.774	0.059	0.114	0.152	0.071	0.375	0.003
262144	2.378	0.156	0.381	0.343	0.238	1.253	0.007

TABLE 6.3: Increasing vector size with 8 core

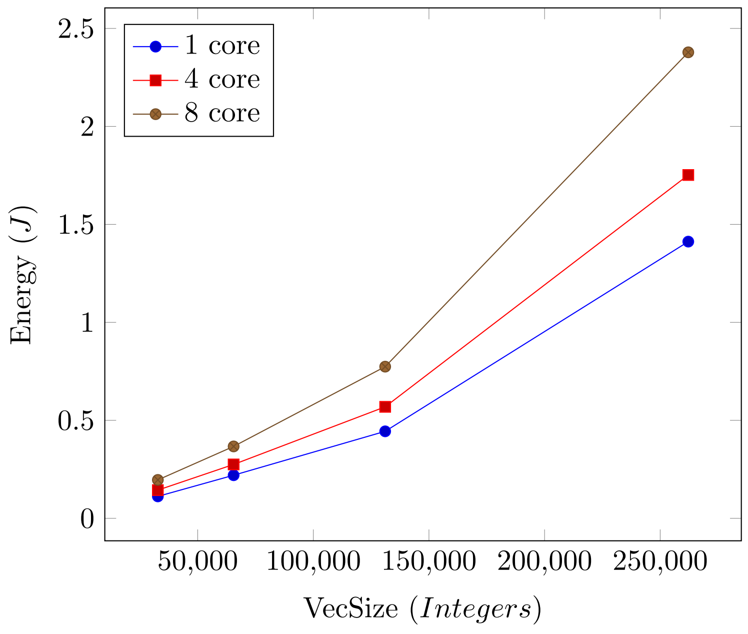


FIGURE 6.1: Processor dynamic energy comparison

Usually the test with most number of cores has the most energy consumption, this is due the fact that the speed up obtained by that number of cores it is not the maximum speed up possible.

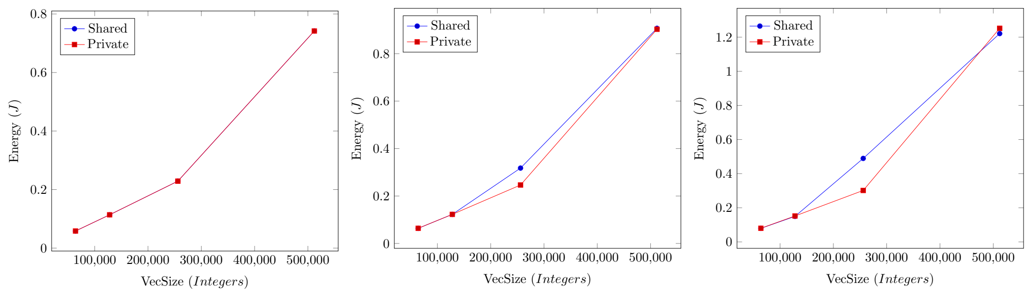
6.1.2 Comparison between private and shared L2 cache

In this test I have changed the TaskSim code in order to test the same application with shared L2 and with private L2. I have tested the application seen before with extra vectors in order to create different tasks with different workloads.

```
for (i = 0 ; i < TASKS ; i++ ){
    computo( x, y, repsOUT );
}
computo( a, b, repsOUT );
```

Listing 6.2: Shared and private cache code

The main difference between both configurations is the number of L2s, while a shared L2 is one cache L2 shared between all the cores, in the private L2 case, all the cores have their own L2 caches.



(A) 1 core (B) 4 core (C) 8 core

Figure 6.2: Processor dynamic energy comparison

We can see that shared L2 gets problems when all the cores share the same cache and are accessing to different vectors. The main difference between both configurations is when both vectors do not fit into the cache L2, although one vector can fit.

Whether one of the vectors does not fit into the cache, or whether both of them fit, the result is the same for all the configurations.

We can also see the difference of static power.

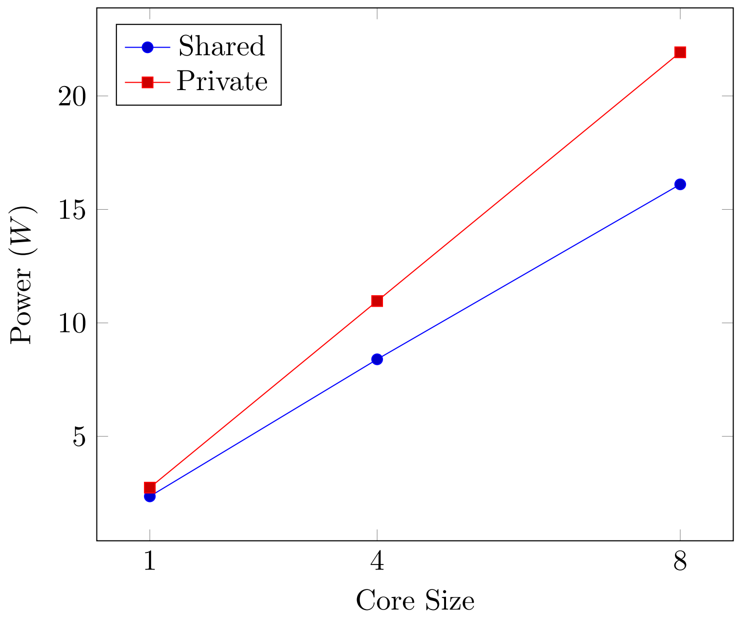


FIGURE 6.3: Processor static power comparison

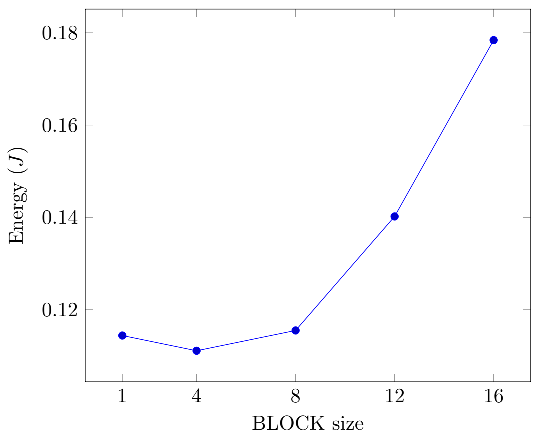
6.1.3 Increasing the memory footprint

I have created a benchmark in order to increase the memory footprint for both cache levels (DL1 and L2). The code which I have used can be seen below.

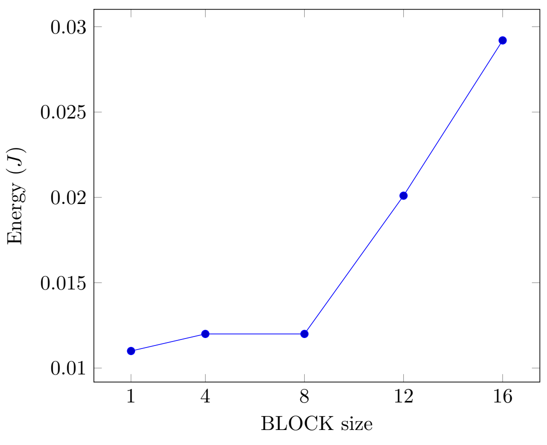
```
for (k = 0; k < reps ; k++ )
    for (i = 0 ; i < vecELEMENTS ; i++ )
        x[i*BLOCK] = y[i*BLOCK];
```

LISTING 6.3: Increasing the memory footprint code

In this code, I have increased the variable BLOCK until the number of misses have been the same as the number of accesses. The results for both cache levels can bee seen below.

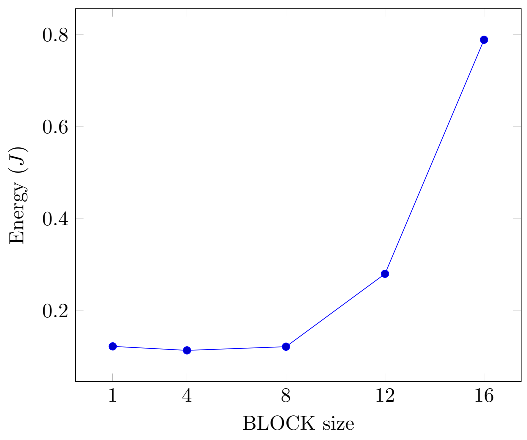


(A) Processor dynamic energy

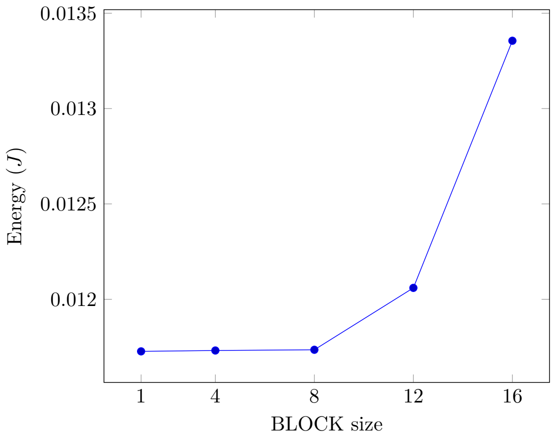


(B) DL1 dynamic energy

FIGURE 6.4: Increasing the memory footprint in DL1 dynamic energy consumption



(A) Processor dynamic energy



(B) L2 dynamic energy

FIGURE 6.5: Increasing the memory footprint in L2 dynamic energy consumption

Both caches have more or less the same figure. We can see that in BLOCK size 1, 4 and 8 the number of misses does not increase too much and also that the vector still fits into the cache. After BLOCK size 8, the vector does not fit inside the cache and the energy consumption rises.

The big difference between DL1 and L2 is the processor energy consumption. While a miss in cache DL1 means an access to L2, a miss in L2 means a memory access, which is much slower than an access to L2. This is why the difference on figure 11.1.A between BLOCK size 1 and BLOCK size 16 is smaller than on figure 11.2.A.

Moreover, we can see that L2 is less consuming than DL1. This is due the fact that in a copy back policy, DL1 has to update in more levels than L2.

6.2 Vector Matrix multiplication

In this section, I have compared the power consumption between going throw a matrix by columns and going throw it by rows. In order to compare both things, I have created a vector-matrix multiplication test, which allows me to test that.

I have used the code below.

```
for (i = 0 ; i < elements ; i++ )
    for(j = 0 ; j < elements ; j++ )
        z[i] += x[j] * y[j*column+i];
```

LISTING 6.4: Columns code

```
for(j = 0 ; j < elements ; j++ )
    for (i = 0 ; i < elements ; i++ )
        z[i] += x[j] * y[j*column+i];
```

Listing 6.5: Rows code

In this test, I have used the same configuration for both simulations, so the static power is the same for both. The power dynamic is also not representative because of the time difference between both tests.

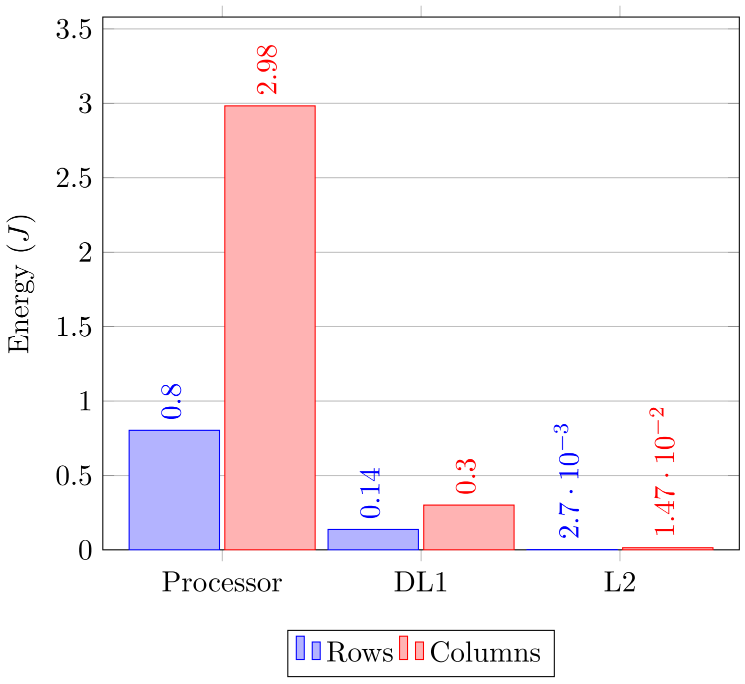


Figure 6.6: Dynamic energy comparison

We can see that the difference between both tests is huge. I have used a 4 core processor, this is why the difference between the total dynamic energy and DL1 or L2 is so high.

6.3 PARSEC Benchmarks

I have created power models from the PARSEC (Princeton Application Repository for Shared-Memory Computers) suite[1]. The suite focuses on emerging workloads and was designed to be representative of the next-generation shared-memory programs for chip-multiprocessors.

I have selected the following PARSEC benchmarks:

- **Ferret:** This application is used for content-based similarity search.

- **Blackscholes:** This application calculates the prices for a portfolio of European options analytically with the Black-Scholes partial differential equation.
- **Swaptions:** This application uses the Heath-Jarrow-Morton framework to price a portfolio of swaptions. Swaptions employs Monte Carlo simulation to compute the prices.
- **Dedup:** This application compresses a data stream with a combination of global and local compression.
- **Streamcluster:** Solves the online clustering problem.

For each benchmark, I have done the following:

- I have executed each benchmark for 1, 4 and 16 cores (I have not been able to create power models of 4 and 16 cores for Dedup and Ferret benchmarks).
- For each core size, I have configured TaskSim with different L2 sizes (128KB, 256KB, 516KB, 1MB and 2MB).
- For each L2 size, I have executed TaskSim with different DL1 sizes (2KB, 4KB, 8KB, 16KB, 32KB, 64KB and 128KB).
- Moreover, I have done the test two times. The first time I have done it without changing the latency and associativity values of cache DL1, those values have been 2 and 4 respectively. In the second time I have changed these values according to the cache size:
  - 2KB and 4KB: Latency 1 and associativity 2.
  - 8KB and 16KB: Latency 2 and associativity 4.
  - 32KB and 64KB: Latency 4 and associativity 8.
  - 128KB: Latency 8 and associativity 16.

For the L2, I have used always the same values. The latency value has been 20 and the associativity value has been 16.

For each benchmark I have created a heat map of:

- Processor dynamic energy consumption (J)
- Processor dynamic power consumption (W)
- Processor static power consumption (W)

- DL1 dynamic energy consumption (J)
- L2 dynamic energy consumption (J)

All the heat maps can be seen in appendix E. In the following sections, I have just included the processor dynamic energy consumption for 1 core with fixed associativity and latency and without it.

6.3.1 Ferret

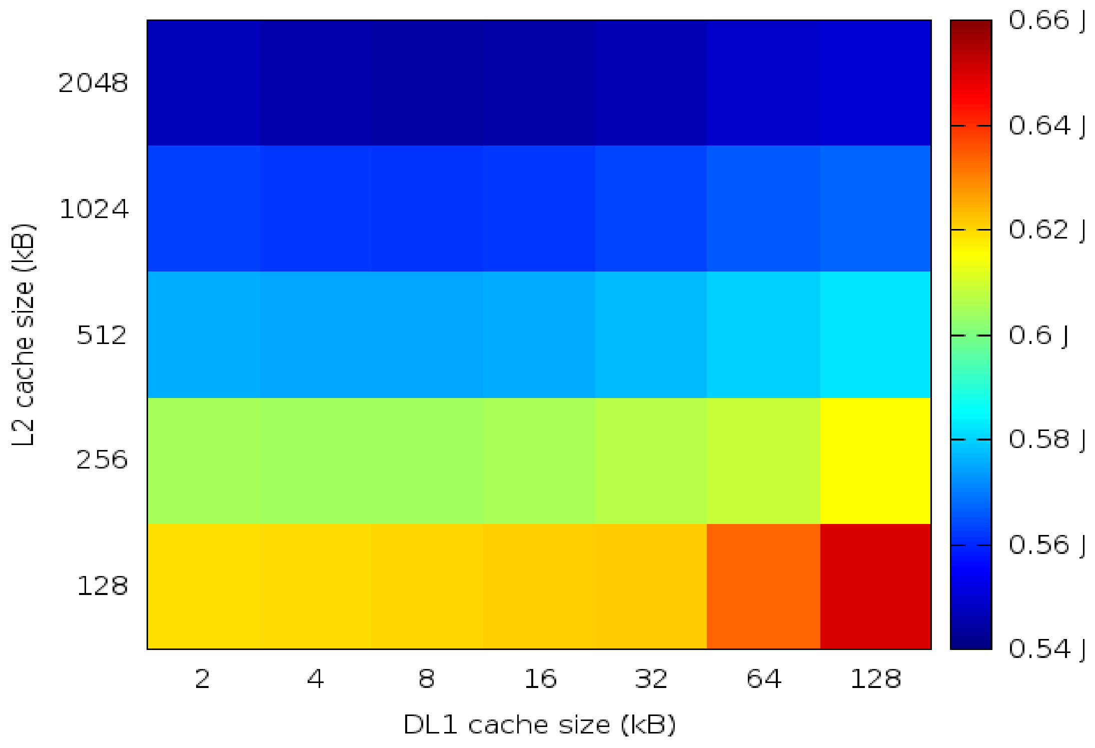


Figure 6.7: Ferret processor dynamic energy consumption with fixed associativity and latency

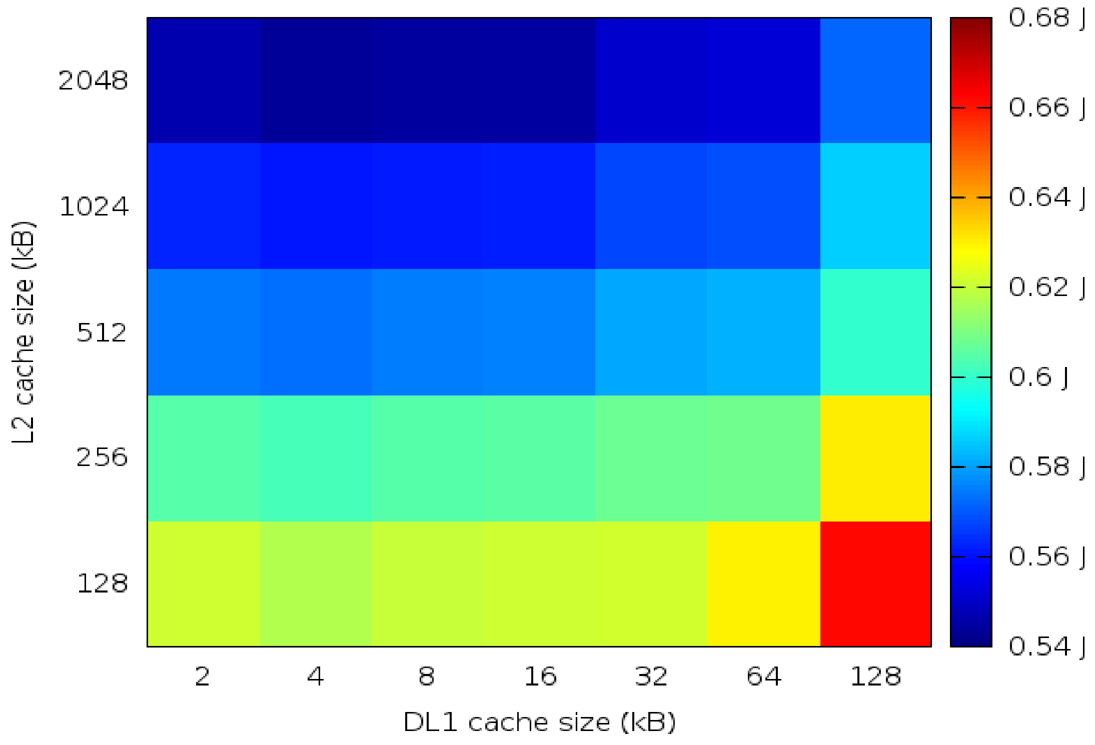


FIGURE 6.8: Ferret processor dynamic energy consumption with unfixed associativity and latency

The graphic result it is due to the fact that the misses in cache L2 are being reduced while L2 size increases. In small L2 size, there are a lot of cache misses so the memory access time increases considerably so this affects the total number of program cycles. So I can say, that in this benchmark, the best L2 cache size is the biggest one possible (2MB in this case).

If we are looking for the best DL1 cache size, it is 8KB. In the DL1 8KB cache the misses are reduced considerably from the previous smaller cache size. After this cache size, cache misses reduce gradually but there is not an improvement due to the fact that the power consumption of the cache rises too.

6.3.2 Blackscholes

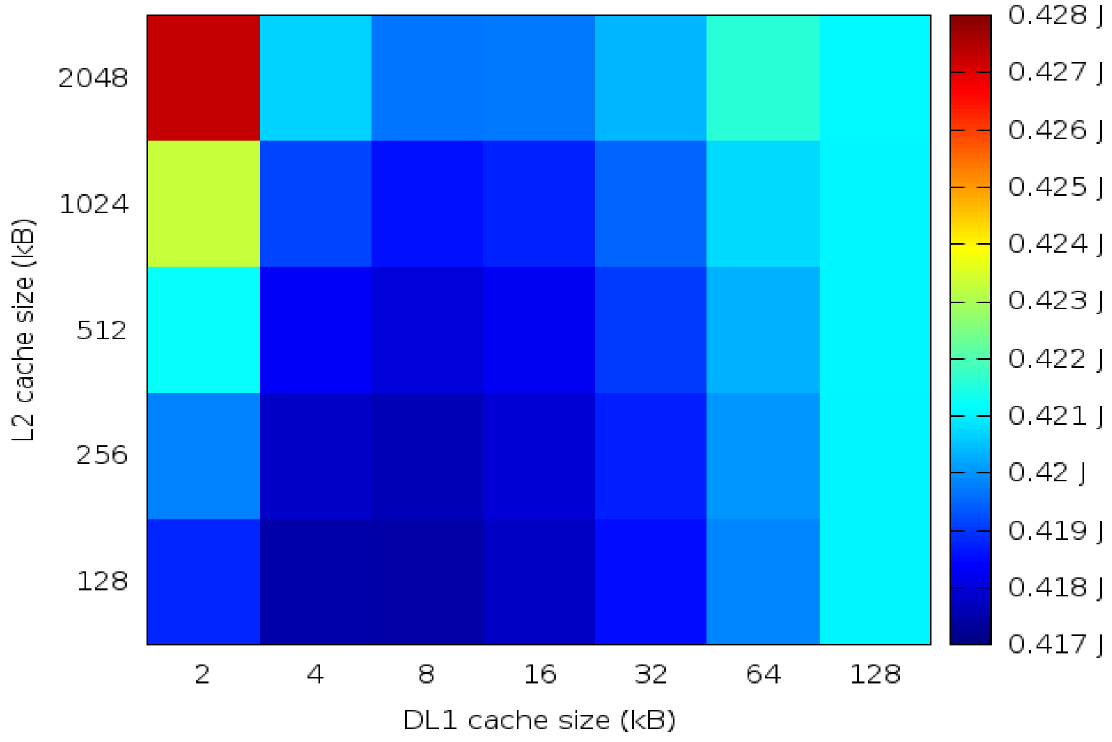


FIGURE 6.9: Blackscholes processor dynamic energy consumption with fixed associativity and latency

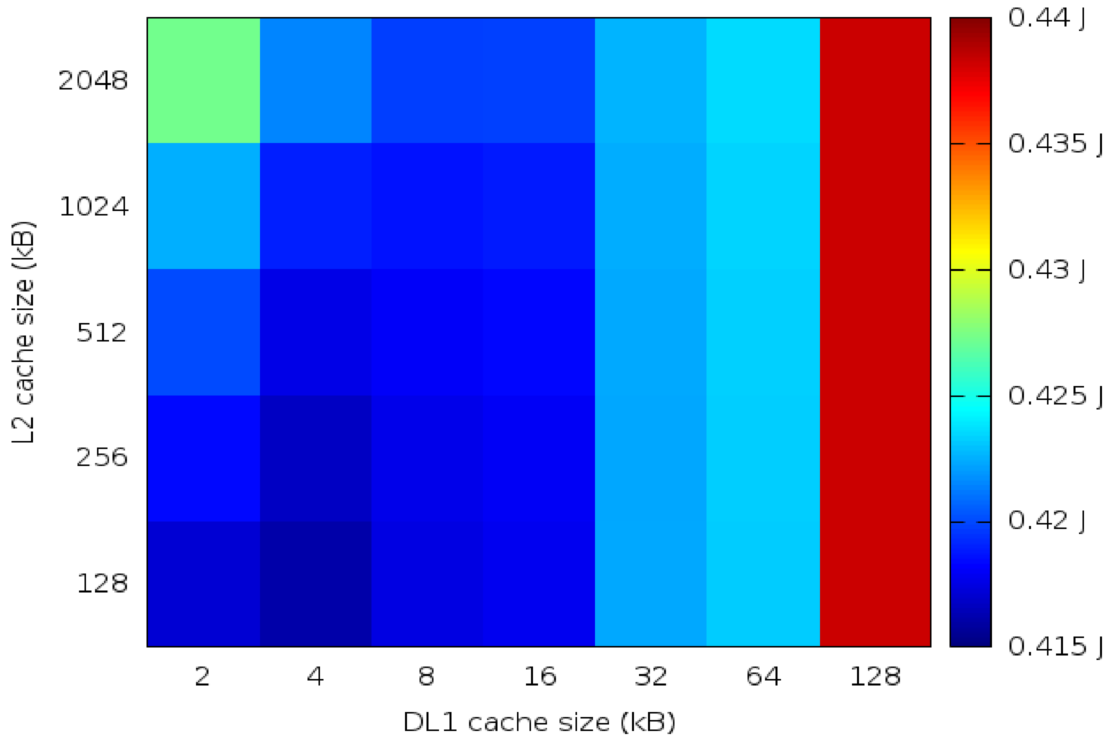


Figure 6.10: Blackscholes processor dynamic energy consumption with unfixed associativity and latency

In this benchmark it is easy to observe the best cache configuration. First of all, cache misses in L2 are stable from the beginning (128KB size). So in this application only matters the number of DL1 misses. In cache size 8KB the number of misses stops decreasing. In the unfixed graphic, the best cache size is 4KB, this is due to the fact that has smaller latency and associativity than 8KB.

6.3.3 Swaptions

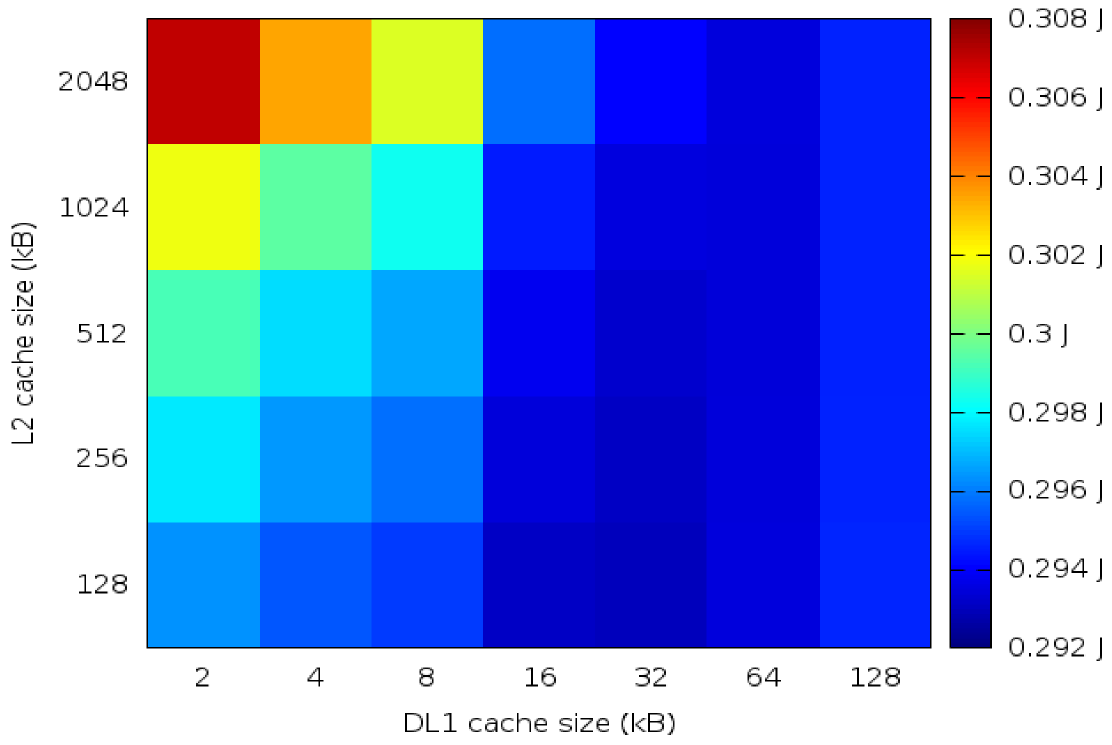


FIGURE 6.11: Swaptions processor dynamic energy consumption with fixed associativity and latency

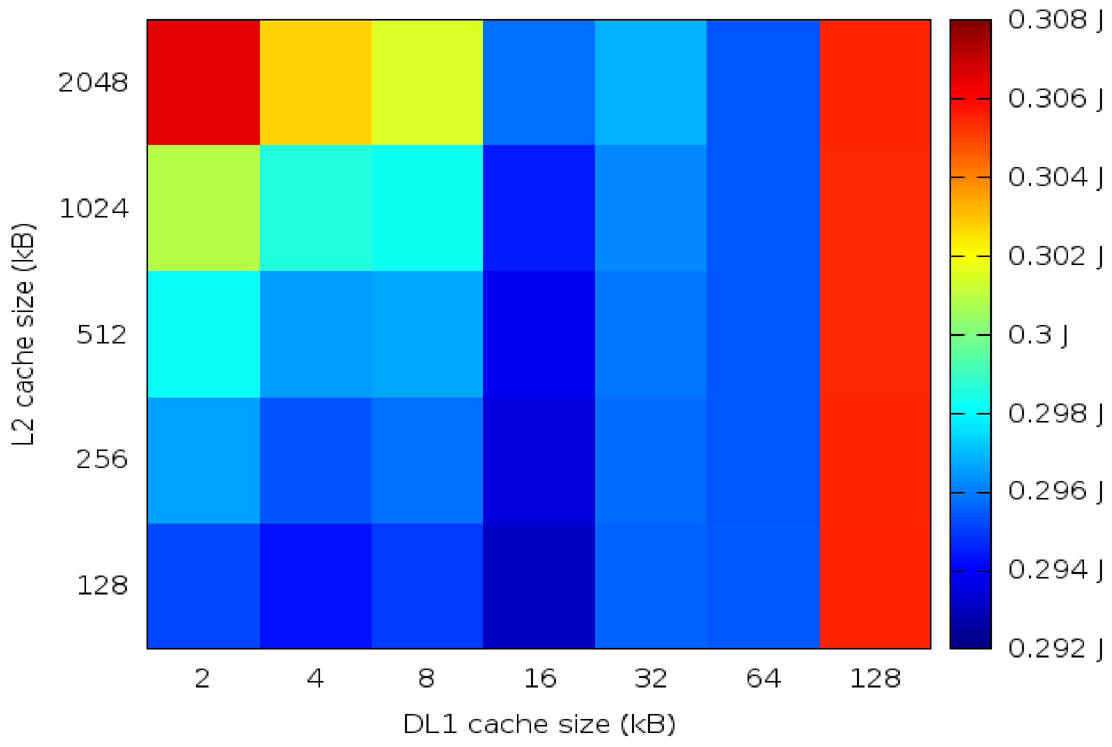


FIGURE 6.12: Swaptions processor dynamic energy consumption with unfixed associativity and latency

In this application we can observe that L2 size is not important, due to the fact that the number of misses remains stable from the very beginning. The number of DL1 misses is reduced by half while the cache size is increased. The best size if we are taking into account the associativity and latency is 16KB, while if we are not, the best solution is 32KB.

6.3.4 Dedup

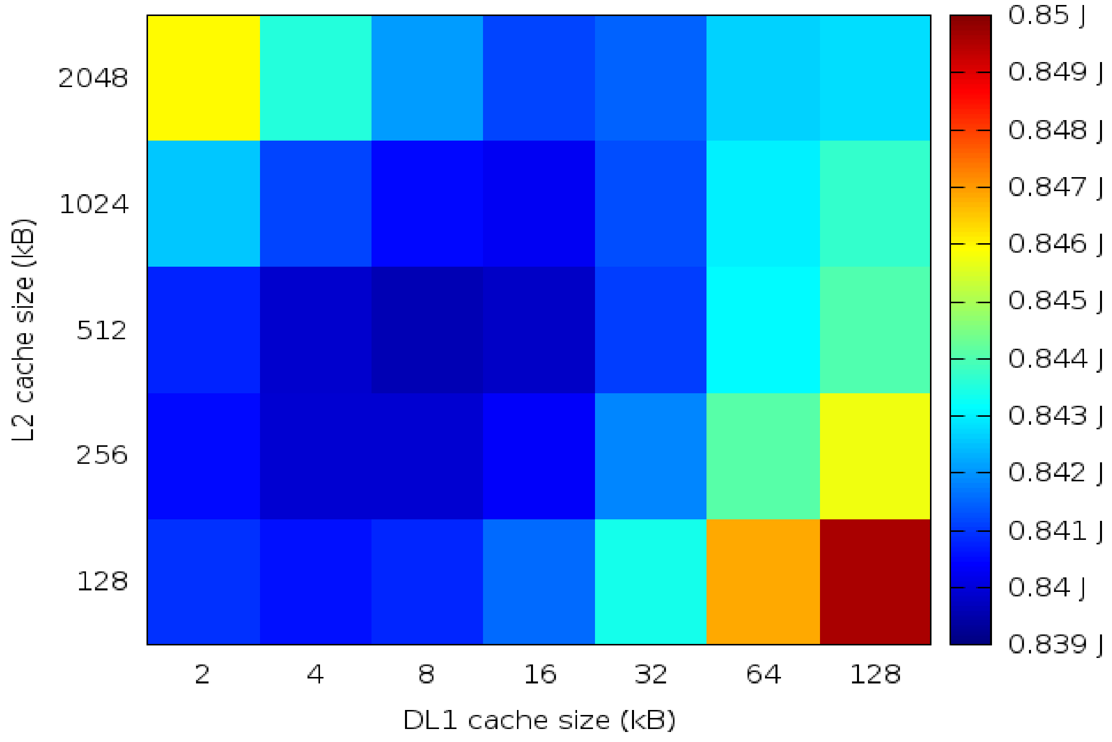


FIGURE 6.13: Dedup processor dynamic energy consumption with fixed associativity and latency

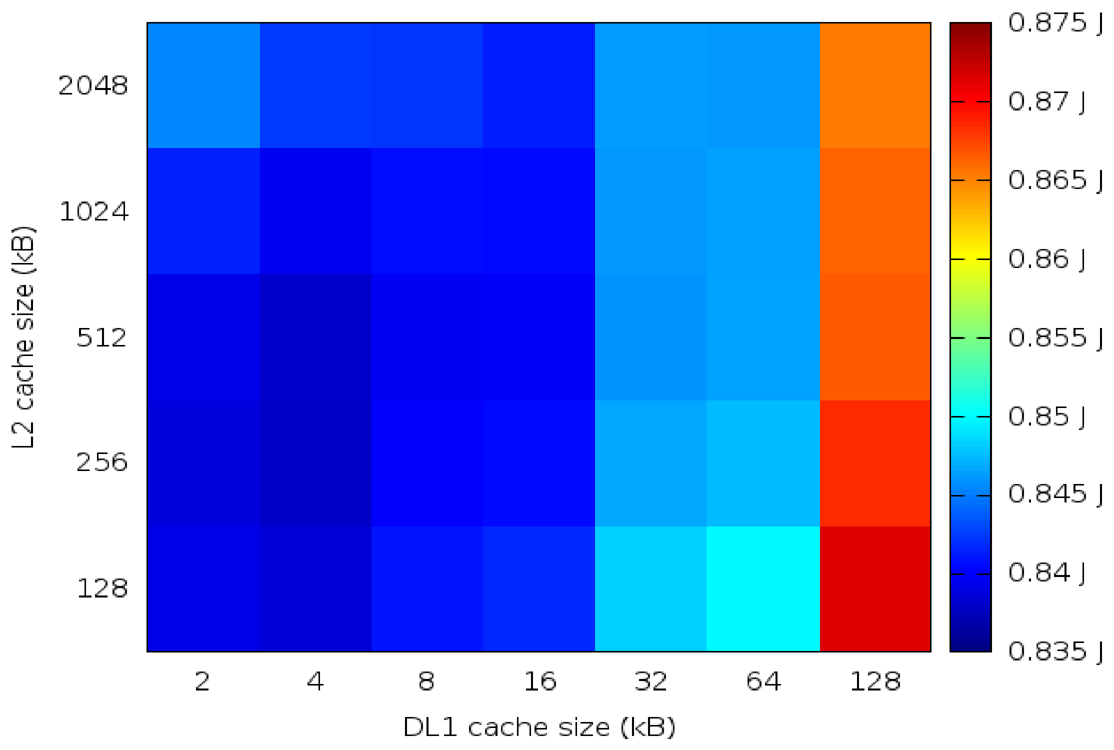


FIGURE 6.14: Dedup processor dynamic energy consumption with unfixed associativity and latency

In Dedup application both caches are important. The misses of L2 stop decreasing in 512KB size. We can see that bigger cache sizes are not the best ones. In cache DL1, the number of misses is reduced by half as the cache size is increased. The best size for the unfixed associativity and latency is 4KB while for fixed associativity and latency is 8KB.

6.3.5 Streamcluster

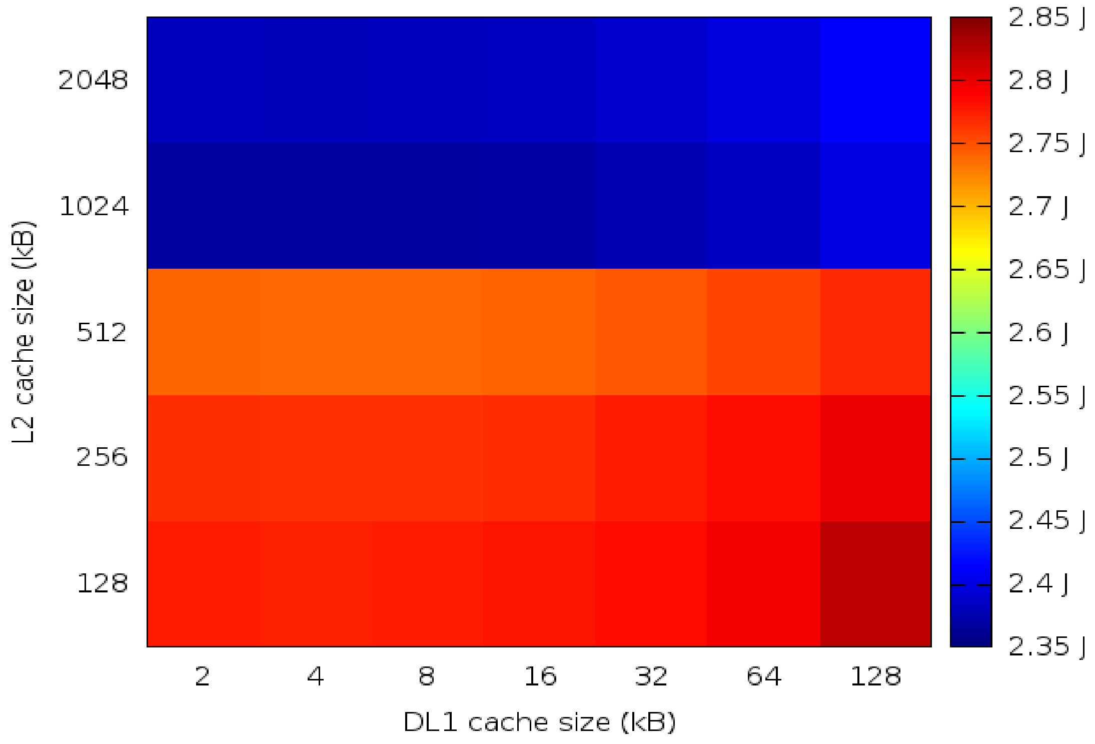


FIGURE 6.15: Streamcluster processor dynamic energy consumption with fixed associativity and latency

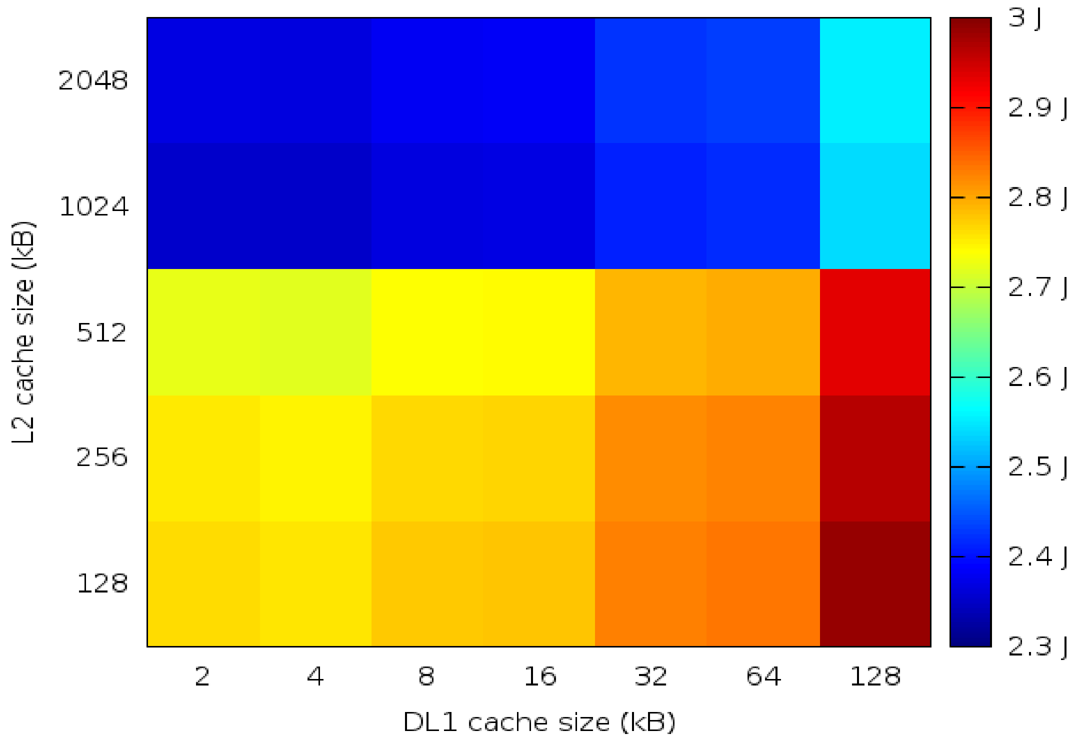


Figure 6.16: Streamcluster processor dynamic energy consumption with unfixed associativity and latency

In this application, L2 misses are the most important thing. As we can see, in cache sizes smaller than 1MB, the number of misses does not decrease really much. Once L2 size reaches 1MB, the number of misses decrease and becomes stable. For 1MB and 2MB the number of misses are the same.

In order to choose a DL1 size, the best solution would be to pick up the smaller one, because the number of misses remains equal for all the DL1 cache sizes.

6.3.6 Average

I have computed the average of the processor power and energy consumption within the different benchmarks (the average of processor power can be seen in the appendix).

Before doing this, and in order to compute the average energy consumption, I have normalized its values.

With the average, we can check the best cache size combination for the PARSEC suite.

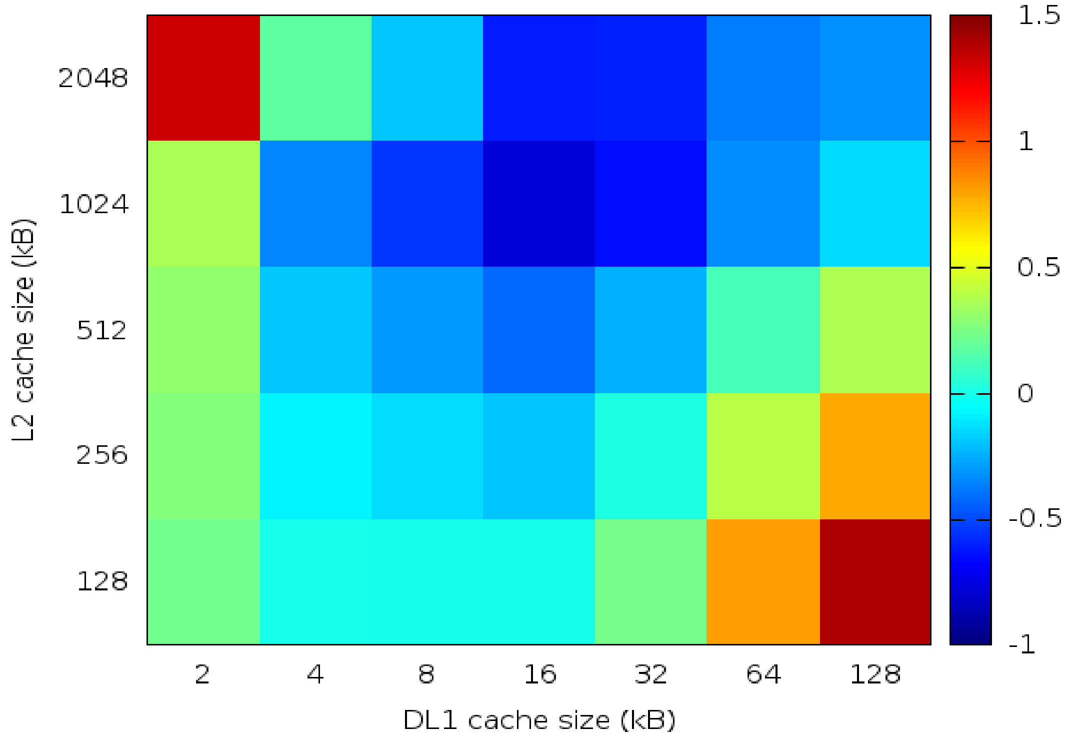


FIGURE 6.17: Average processor dynamic energy consumption with fixed associativity and latency

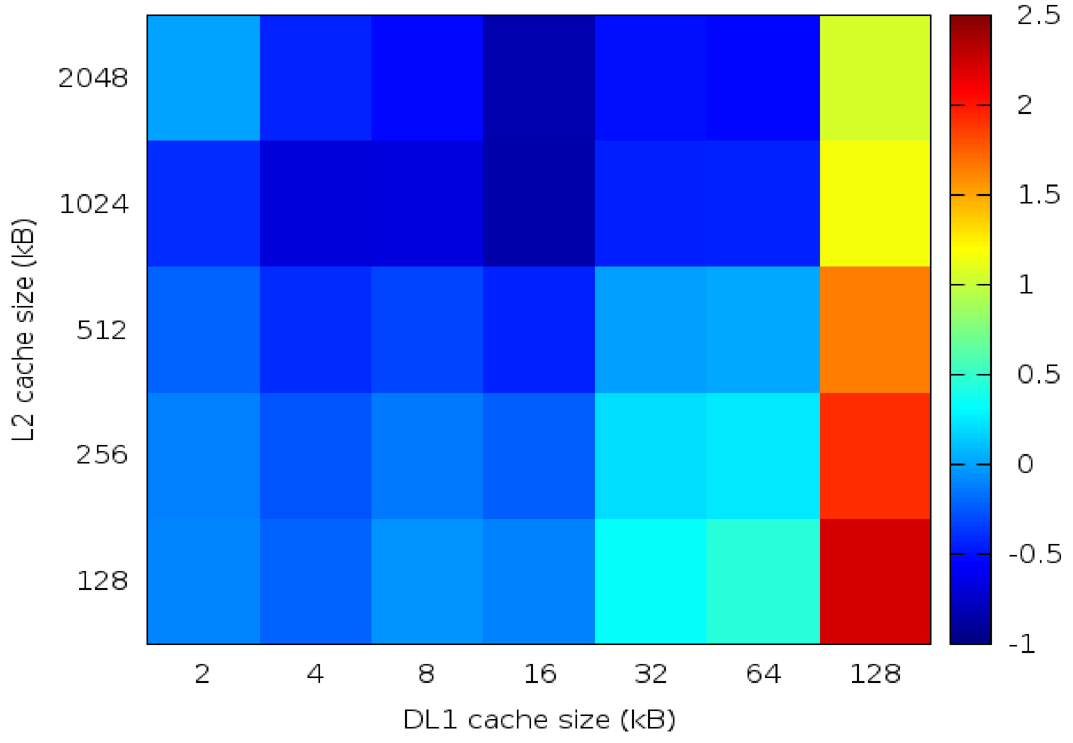


FIGURE 6.18: Average processor dynamic energy consumption with unfixed associativity and latency

6.3.7 Further results

All the graphics included in this section, are the same for all benchmarks.

6.3.7.1 Static power

As bigger caches are used, more transistors are used, so more static power is consumed.

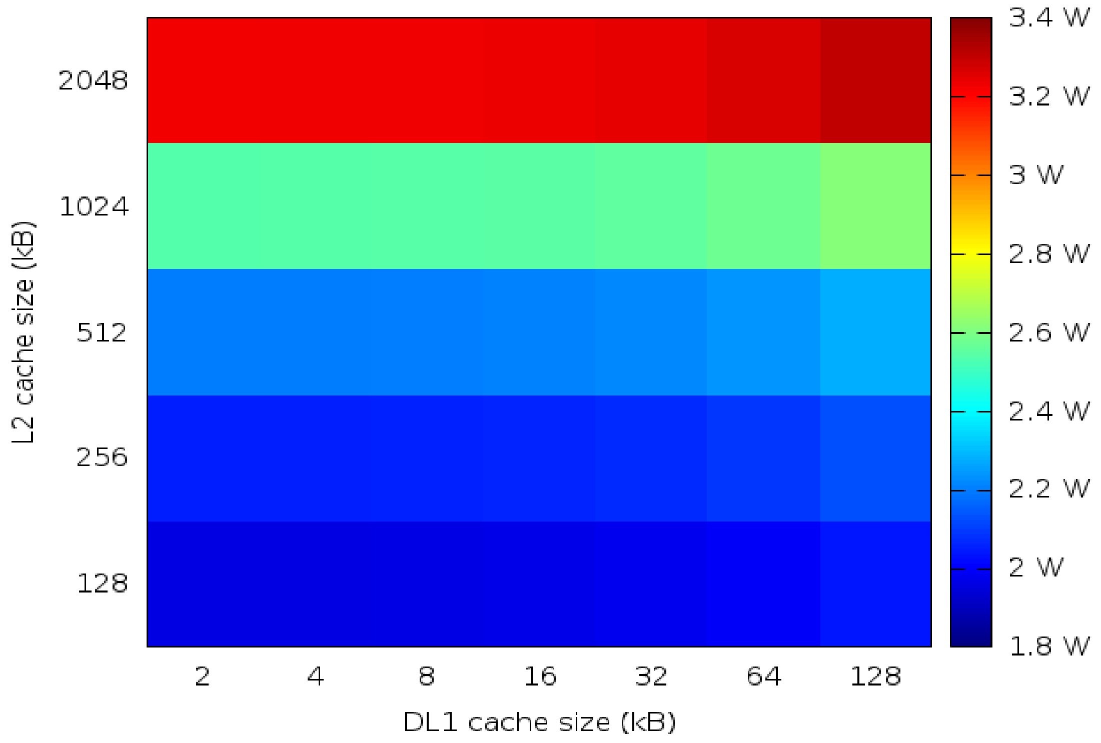


FIGURE 6.19: Processor static power

6.3.7.2 DL1 dynamic energy consumption

We can see that a smaller cache is usually the best option. The rate misses/hits has not much influence on DL1 energy consumption although it has on other factors (L2).

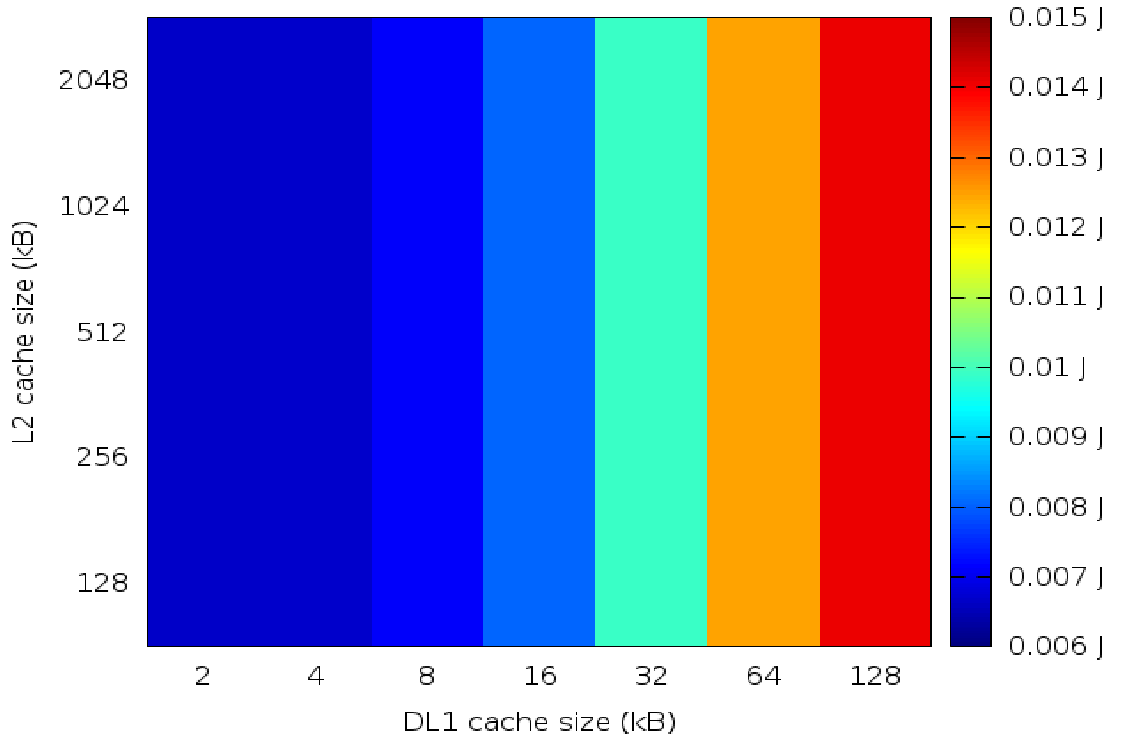


Figure 6.20: DL1 dynamic energy consumption

Moreover, when misses are reduced by half in each cache size (if the number is really big), then something like this can happen:

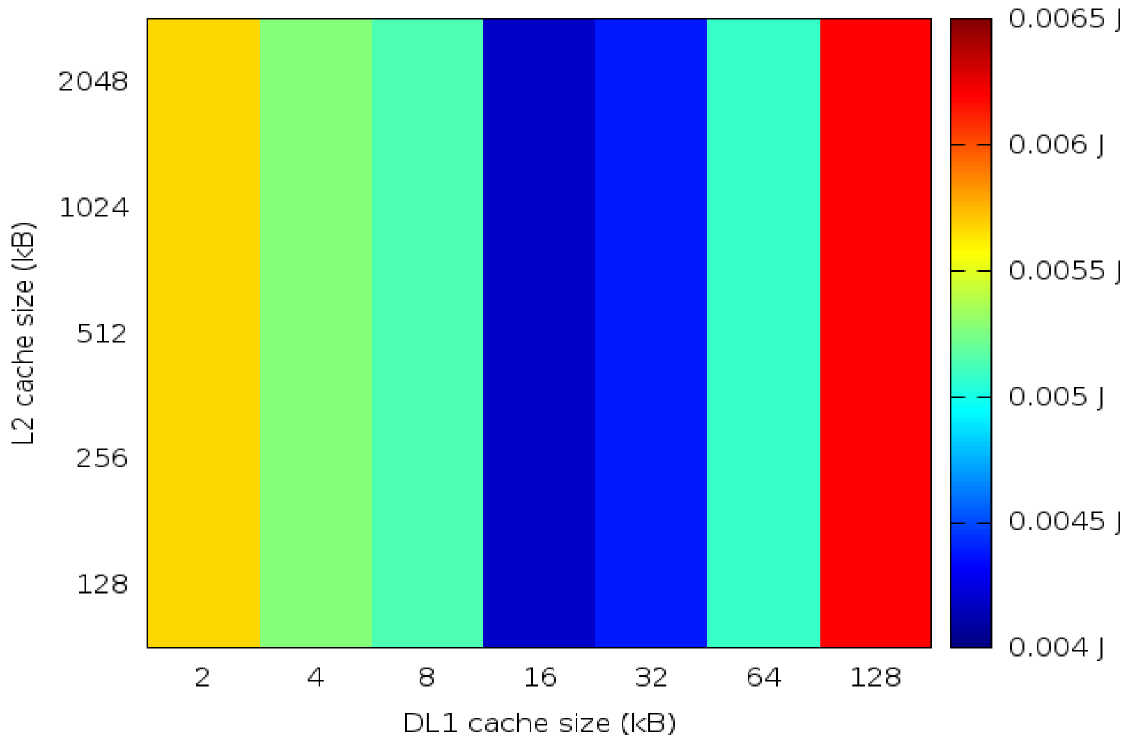


Figure 6.21: Swaptions DL1 dynamic energy consumption

6.3.7.3 L2 dynamic energy consumption

A bigger cache means a bigger consumption. In this graphic, for the same number of accesses (vertical line), the best solution is always the smaller one. Moreover, we can see that as we increase the DL1 size, we are decreasing the cache L2 number of accesses (horizontal line).

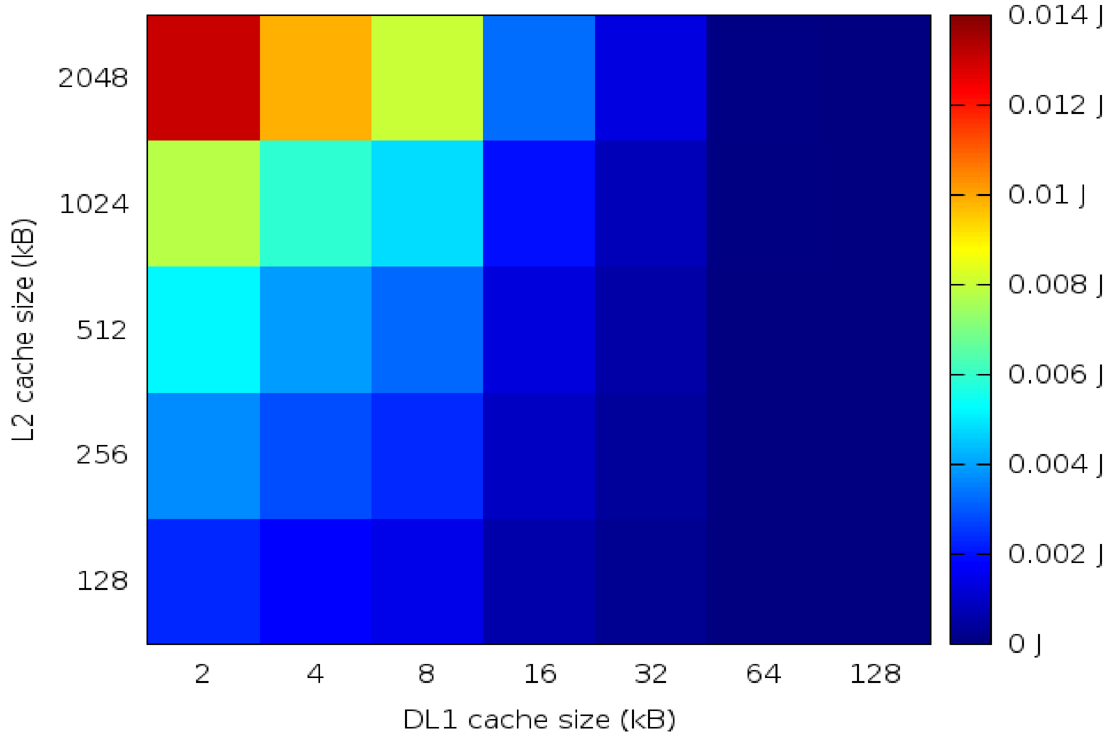


FIGURE 6.22: L2 dynamic energy consumption

This is the result when the number of accesses in L2 remains equal in all DL1 sizes (we have seen this in the streamcluster benchmark).

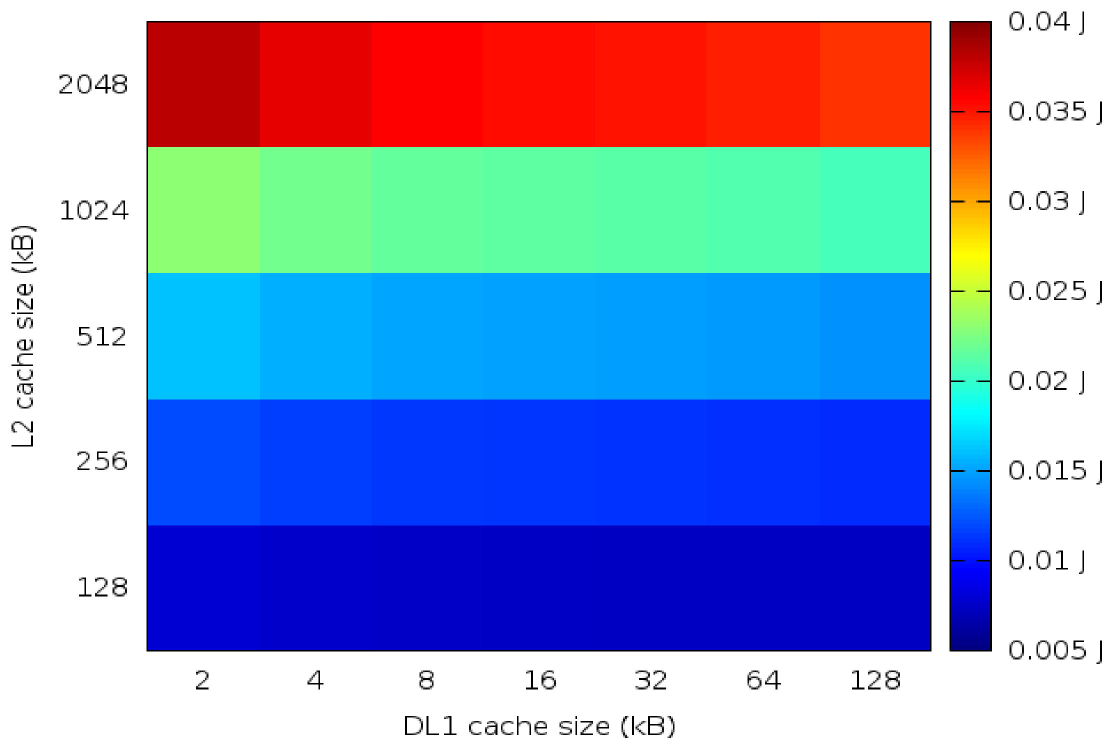


FIGURE 6.23: Streamcluster L2 dynamic energy consumption

6.4 Paraver visualization

6.4.1 Copy vector with different vector sizes

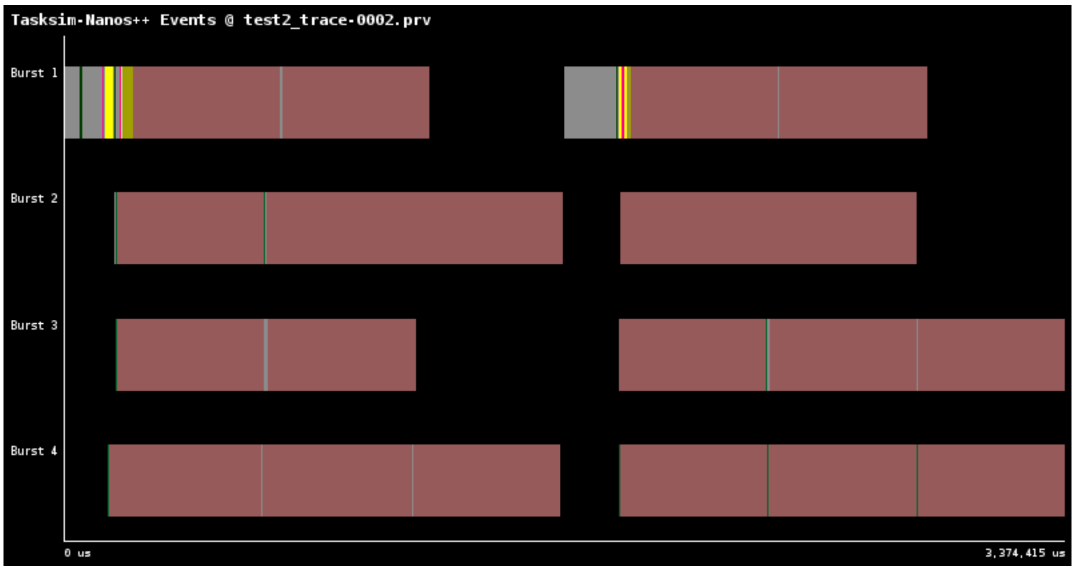
I have selected this test in order to check the behaviour of the power consumption along the execution. This test allows me to see what happens in the different tasks when a vector fits into the DL1 cache and when not. I have used the following code:

```
 for (i = 0 ; i < TASKS ; i++ )
     computo( x, y, repsOUT );
#pragma omp taskwait
 for (i = 0 ; i < TASKS ; i++ )
     notask( x, y);
 for (i = 0 ; i < TASKS ; i++ )
     computo( x, y, repsOUT );
#pragma omp taskwait
```

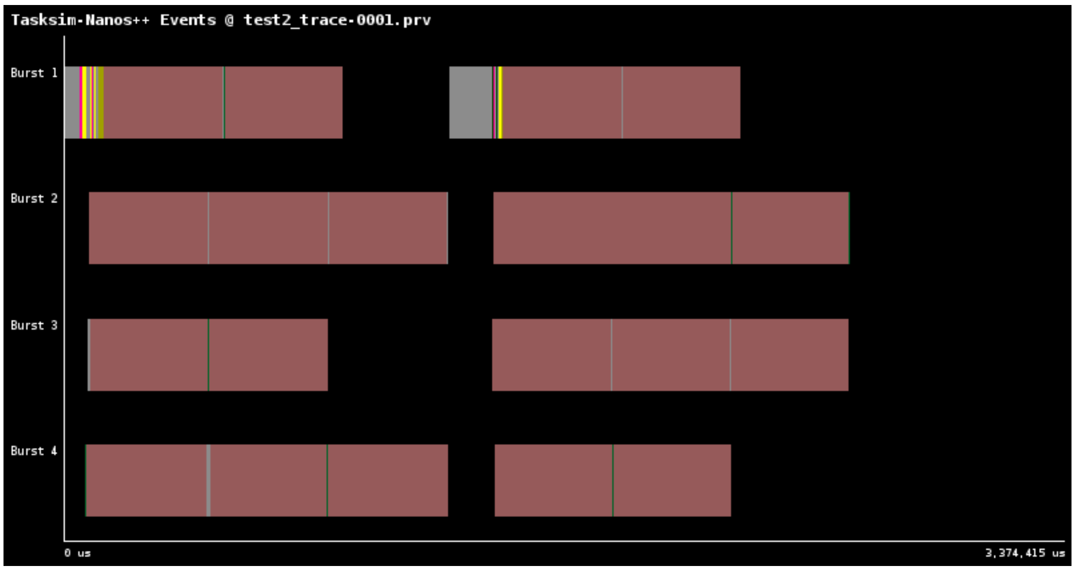
LISTING 6.6: Paraver code

I have selected a 4 core processor with shared L2 cache. I have selected a shared L2 cache because it is easier to observe what happens into the cache. I have also reduced the vector size due to the fact that the Paraver trace is much bigger than the TaskSim trace.

The first graphic is the core burst, in this graphic we can see how the tasks are divided and how they are executed in the different CPUs.



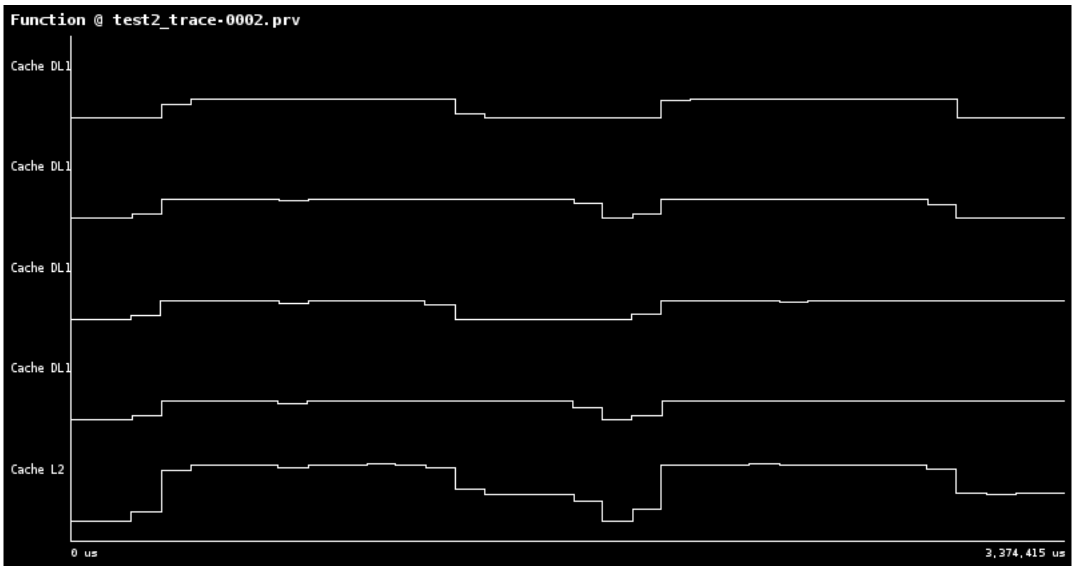
(A) Vector does not fit



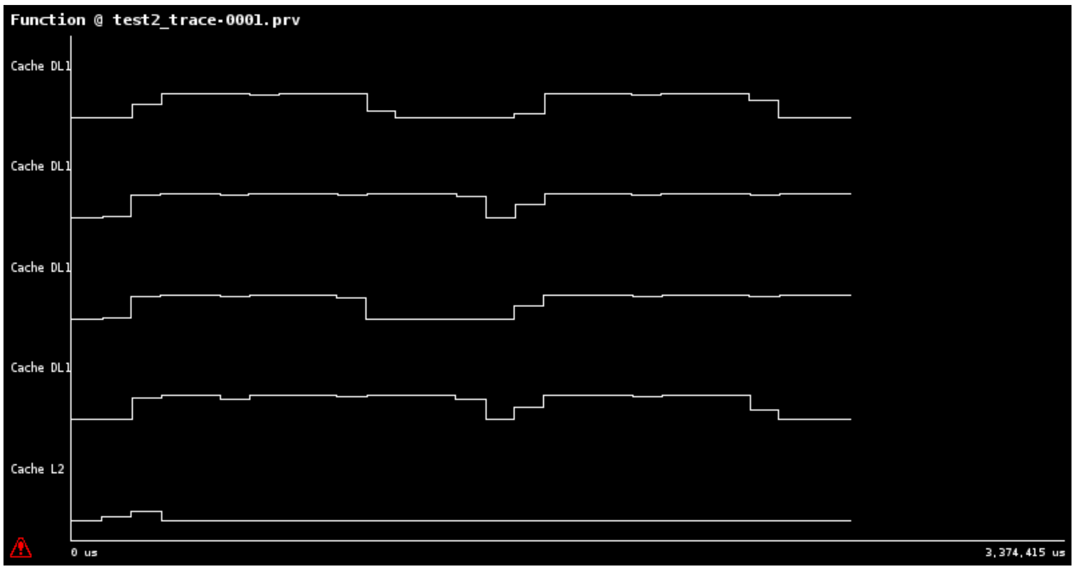
(B) Vector fits

FIGURE 6.24: Core burst

In the following figure, we can check how the cache power consumption is distributed.



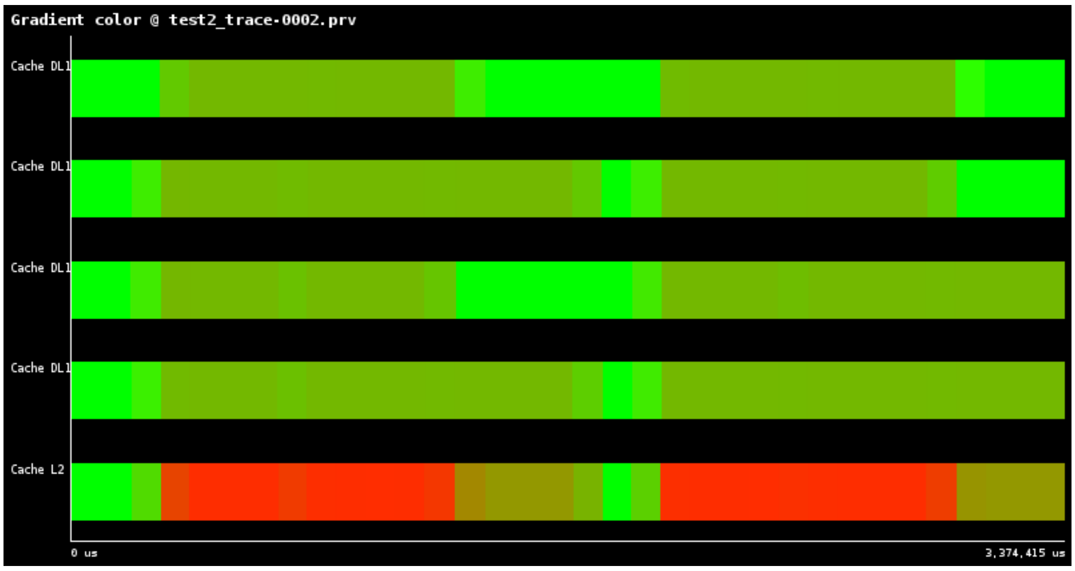
(A) Vector does not fit



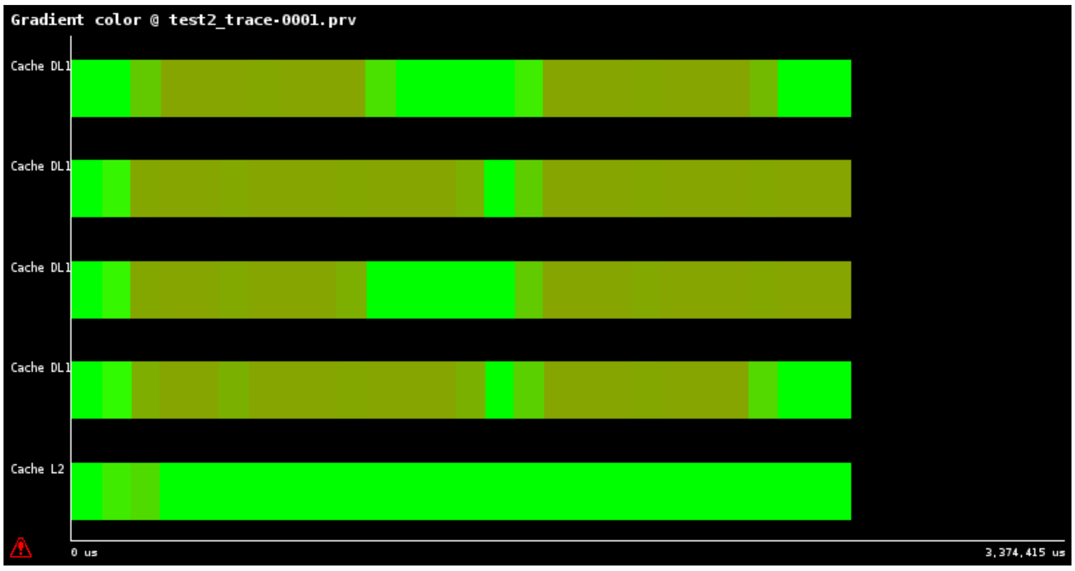
(B) Vector fits

FIGURE 6.25: Function line dynamic power consumption (0W - 0.094W)

Paraver can also create a gradient color plot.



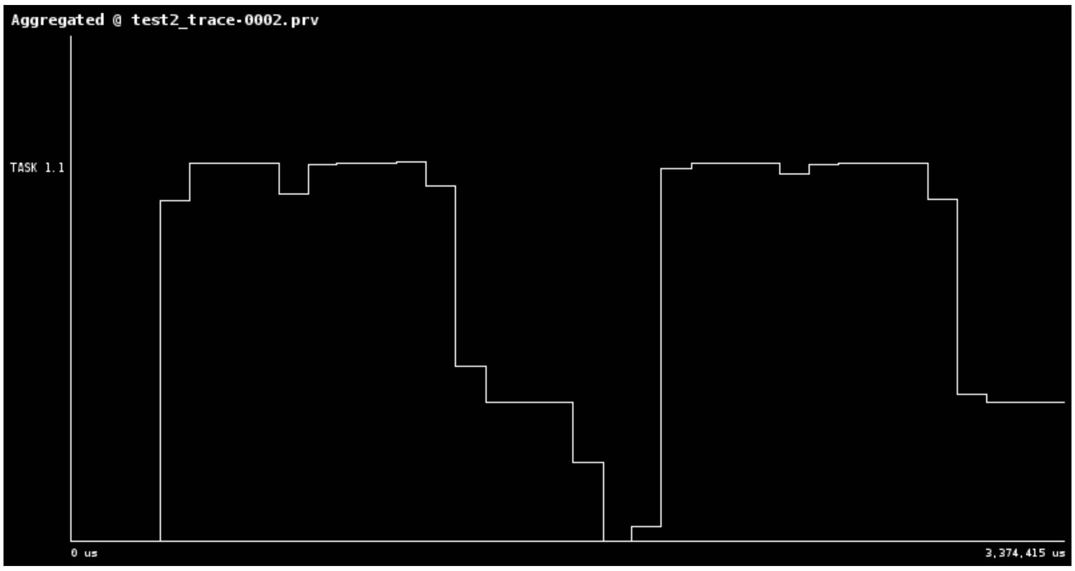
(A) Vector does not fit



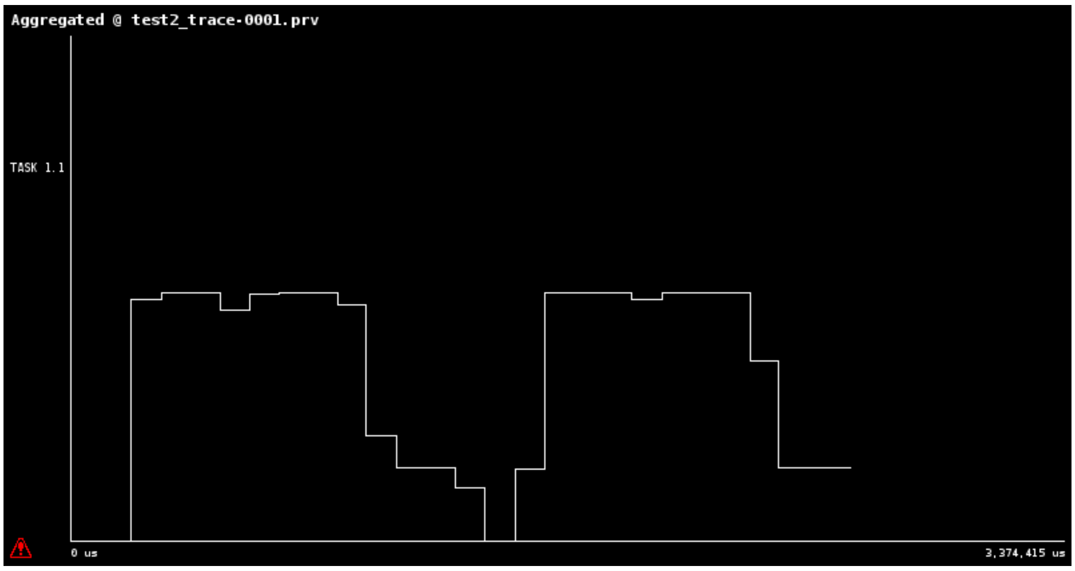
(B) Vector fits

FIGURE 6.26: Gradient color dynamic power consumption (0W - 0.094W)

Finally, I have generated the aggregated power consumption of the caches.



(A) Vector does not fit

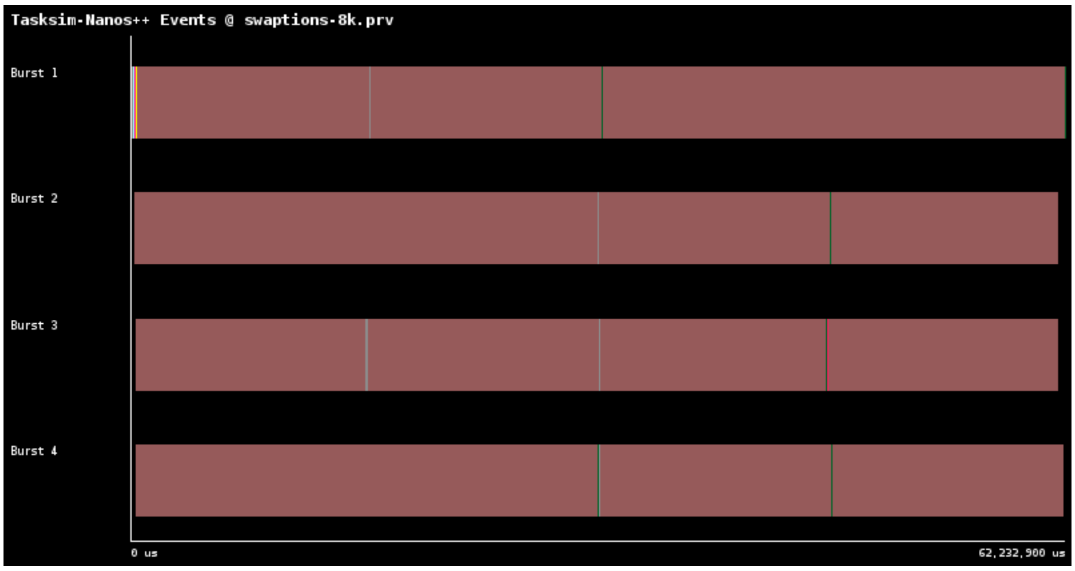


(B) Vector fits

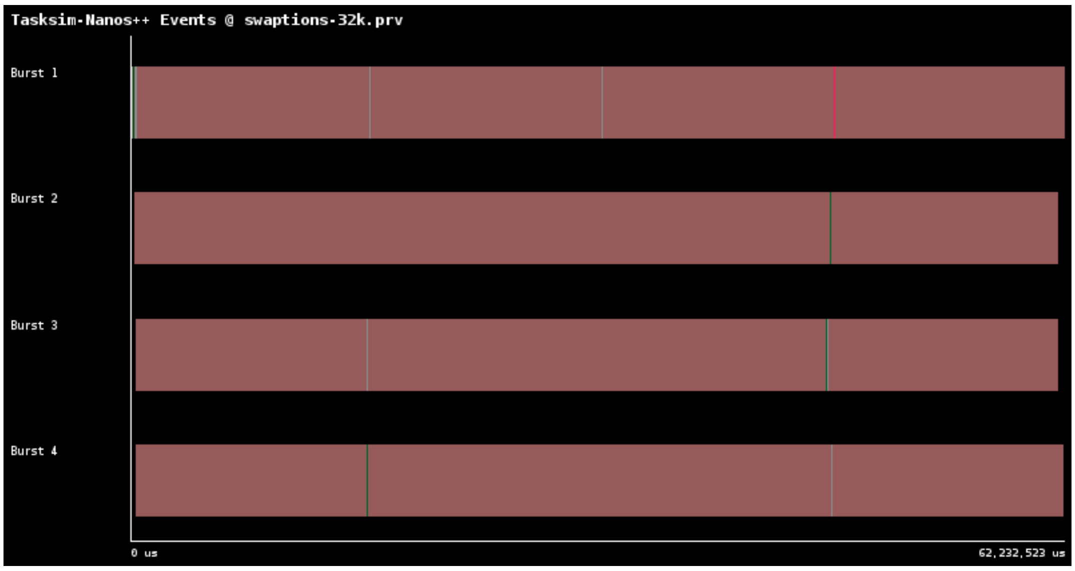
FIGURE 6.27: Aggregated dynamic power consumption (0W - 0.226W)

6.4.2 Swaptions with different cache sizes

I have extracted the power consumption distribution from swaptions. In this test I have selected a 4 core processor with private L2 cache. The size of L2 cache has been 1MB. I have done a test with a DL1 cache of 8KB and another one with a DL1 cache of 32KB in order to corroborate the results seen in the previous chapter.

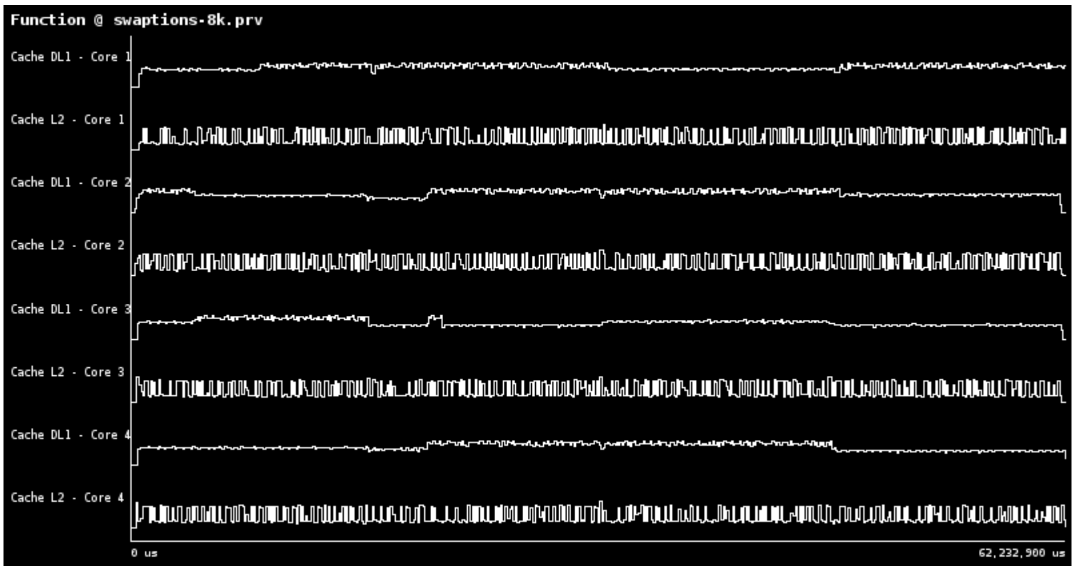


(A) 8KB DL1 Cache

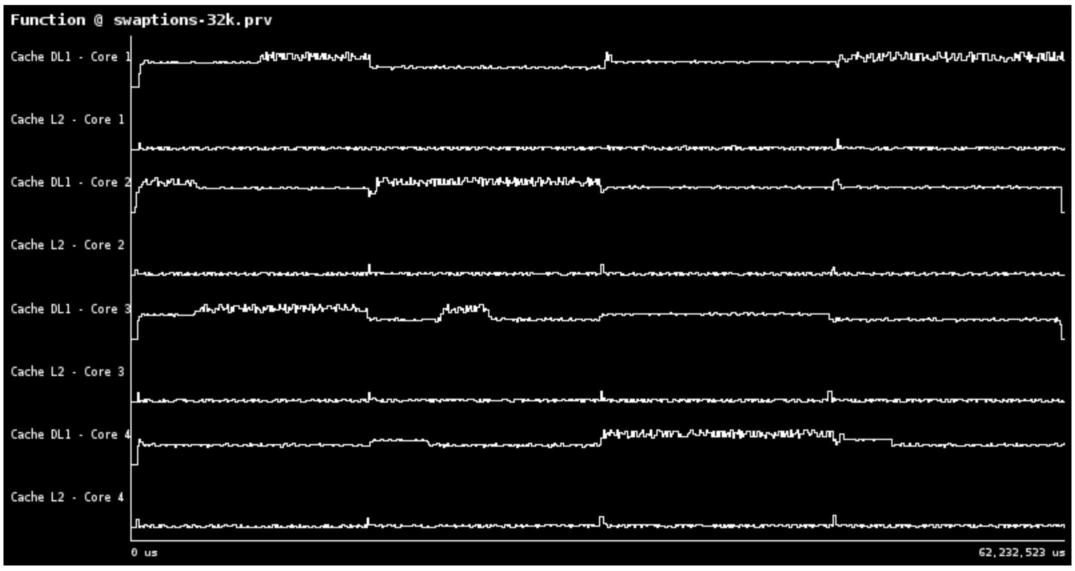


(B) 32KB DL1 Cache

FIGURE 6.28: Core burst

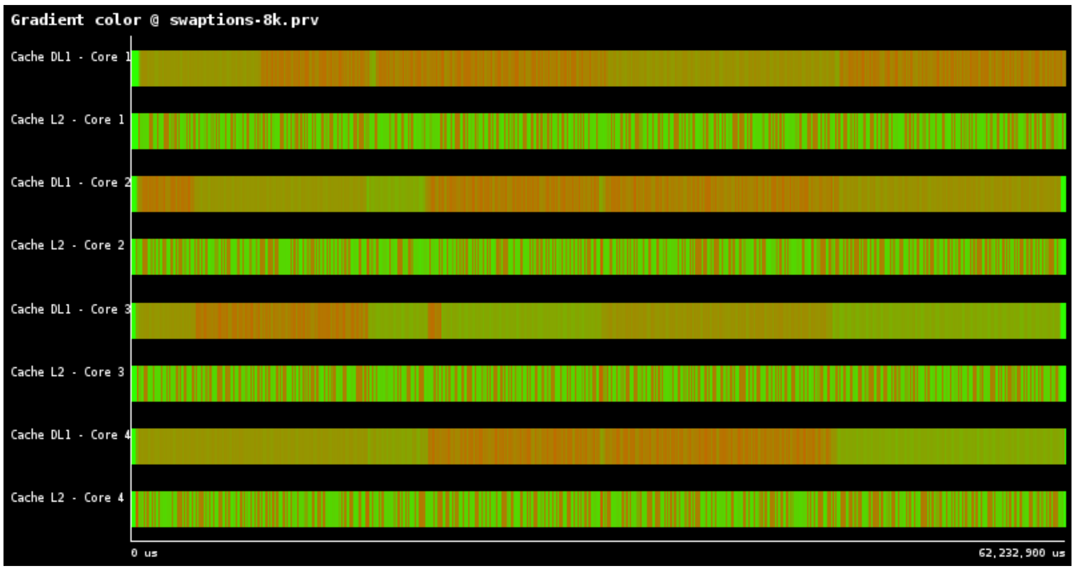


(A) 8KB DL1 Cache

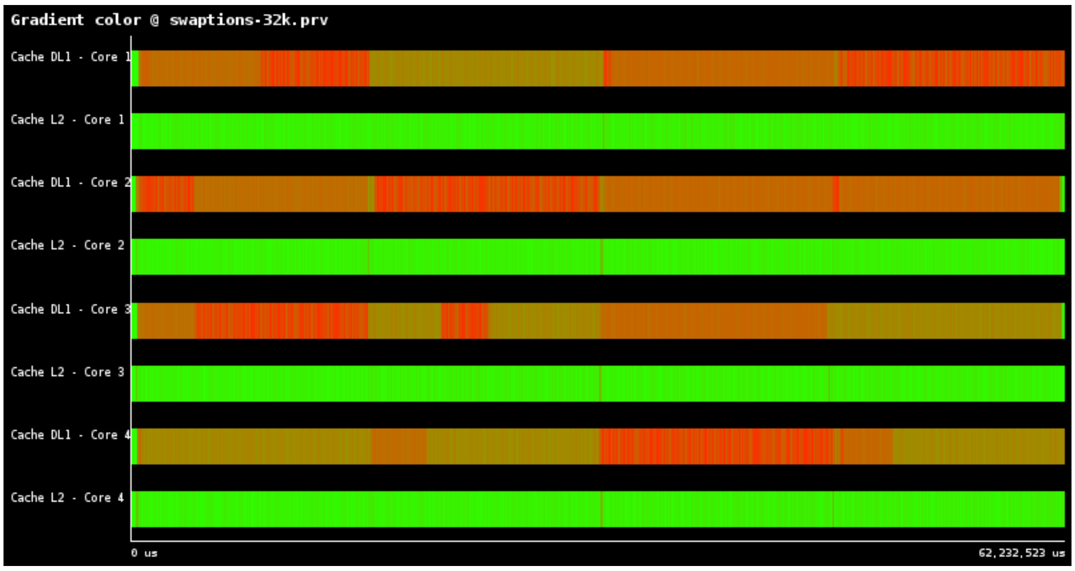


(B) 32KB DL1 Cache

Figure 6.29: Function line dynamic power consumption (0W - 0.056W)

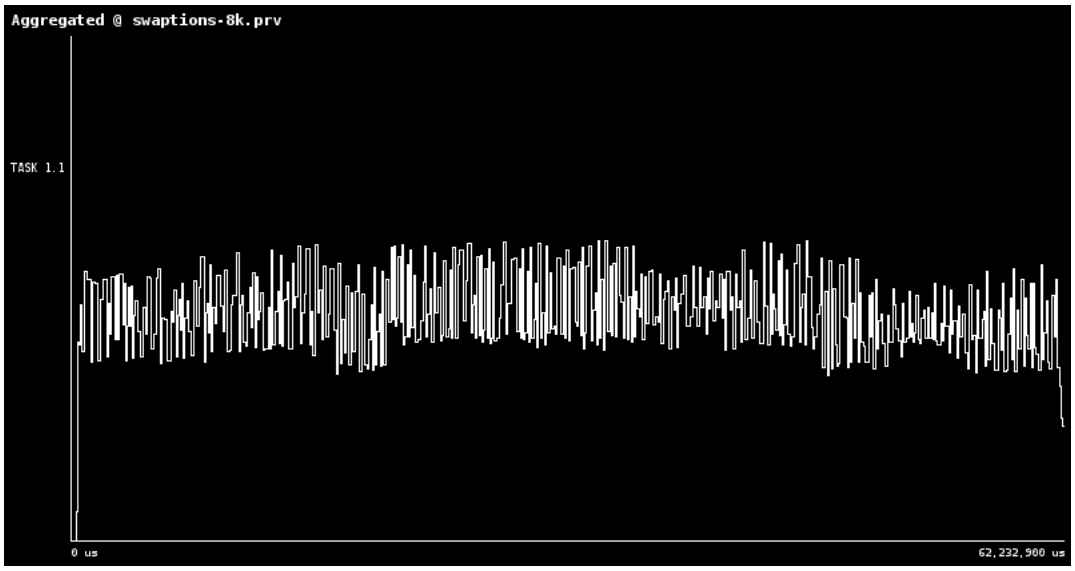


(A) 8KB DL1 Cache

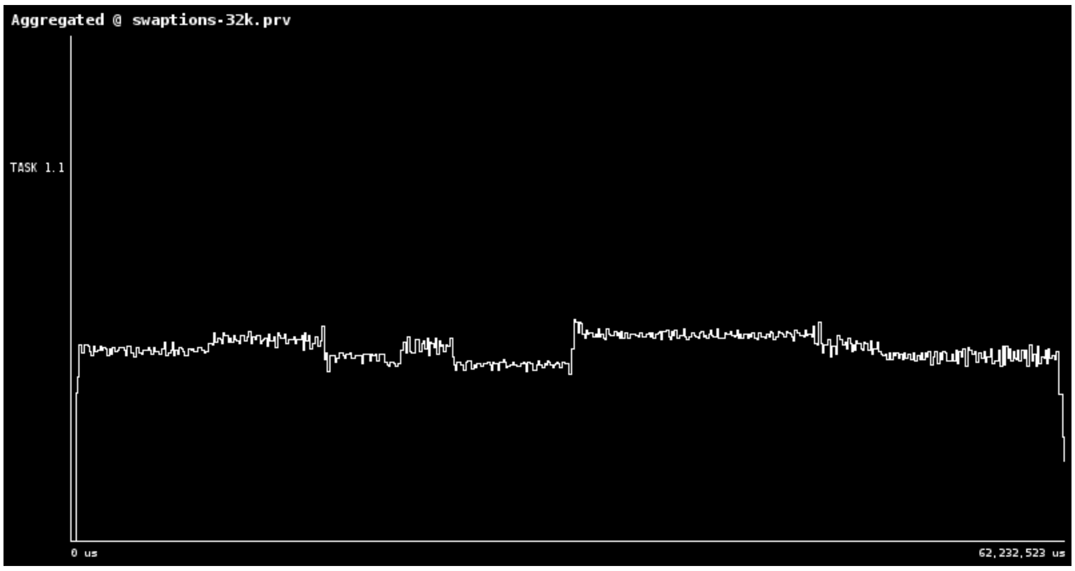


(B) 32KB DL1 Cache

FIGURE 6.30: Gradient color dynamic power consumption (0W - 0.056W)



(A) 8KB DL1 Cache



(B) 32KB DL1 Cache

FIGURE 6.31: Aggregated dynamic power consumption (0W - 0.381W)

As we can see in the function line figure and in the gradient figure, the power consumption of 32KB DL1 cache is bigger than the 8KB cache although that the miss rate is lower. This is due to the fact that the power consumption for access is bigger in 32KB than in 8KB cache.

The aggregate power consumption is bigger in cache 8KB because of the power consumption of L2. We can check the difference of L2 power consumption in the gradient figure and in the function line figure.

Chapter 7

Conclusions

With all the work done, I have accomplished all the main objectives mentioned on the project scope chapter:

- First of all, I have created a consistent tool in order to generate power models for different applications and secondly I have evaluate the correct working of the tool with different benchmarks as we can see in the evaluation chapter.
- I have explored different applications with different workloads. Moreover, I have explored the best hardware for the PARSEC benchmark.
- I have created a tool in order to use Paraver to compute the power consumption.

Moreover, as we can see in the benchmarks, the variable which more influences the power consumption of an application is the number of accesses (in the cache component). But as we can observe in PARSEC benchmarks, we can vary our hardware in order to improve the power consumption. Although in the test I have only varied the cache size, there are a plenty of variables which can be modified in order to be more optimal. The tool allows us to easily explore the best hardware suite to launch our executions in power terms.

But if what we want is to change the behaviour of our application in order to reduce the power consumption, we can check with Paraver the most power consuming parts and try to optimize them.

7.1 Future work

Although I have done what has been possible with TaskSim, in a near future, TaskSim is going to offer more functions like pipelining operations. The future work to do would be to maintain the tool in order to add all the new statistics to the tool. With more information about the executions, more accurate and real power models could be generated.

Moreover, it would be possible to integrate McPAT inside the TaskSim code, like other simulators we have seen before. This would allow us to generate the power consumption of an application immediately and give to TaskSim more functions.

It would be also reasonable to test the tool with all the verified benchmarks that TaskSim project has. In this way, we could analyze which is the best hardware in order to support all the benchmarks.

Appendices

Appendix A

TaskSim configuration file

```
ncpus = 1
mode_selector = MEMORY
idle_cycles = 0
forward_task_in_out_to_cpu = false
measure = FULL_APPLICATION
master_speedup_ratio = 1.0

[Burst]
perf_ratio = 1

[MemCPU]
out_buff_size = 4
rob_size = 192
commit_rate = 4

[DL1Cache]
mshr = DL1MSHR
level = 1
num-ports = 1
latency = 2
size = 131072
line-size = 32
word-size = 4
assoc = 1
victim-lines = 0
#victim-cache-affinity = ANY

[DL1MSHR]
size = 128

[L2Cache]
mshr = L2MSHR
```

```
level = 2
num-ports = 1
latency = 10
size = 1048576
line-size = 32
word-size = 4
assoc = 1
victim-lines = 0

[L2MSHR]
size = 128

[Memory]
latency = 10000

[L2Bus]
latency = 1
width = 8

[DRAM]
frequency-divider = 5
cas = 16
ras = 16
precharge = 16
access-mask = rrrrrrrrrrrrrRRRRBBBBhhhhhhhhhhbbb
input-buffer = 4

[MMU]
page-size = 8192
access-priority = total
allocation-policy = dynamic_2
empty-page-threshold = 16
backward-migration-factor = 0.4

[Paraver]
buffer_size = 1000
hardware_sampling_interval = 100000
```

Listing A.1: TaskSim configuration file

Appendix B

TaskSim results file

```
Final cycle count: 762130109
0:1:33:4:27:3:2:3:18:4:45:2:17:2:42:2:32:15:burst_count:Burst:0
0:underflows:Memory CPU store size histogram:Memory:1
35240785:0 to 2:Memory CPU store size histogram:Memory:1
0:2 to 4:Memory CPU store size histogram:Memory:1
0:4 to 6:Memory CPU store size histogram:Memory:1
0:6 to 8:Memory CPU store size histogram:Memory:1
0:8 to 10:Memory CPU store size histogram:Memory:1
0:10 to 12:Memory CPU store size histogram:Memory:1
0:12 to 14:Memory CPU store size histogram:Memory:1
0:14 to 16:Memory CPU store size histogram:Memory:1
0:overflows:Memory CPU store size histogram:Memory:1
:Memory CPU store size histogram:Memory:1
0:underflows:Memory CPU load size histogram:Memory:1
35767668:0 to 2:Memory CPU load size histogram:Memory:1
0:2 to 4:Memory CPU load size histogram:Memory:1
0:4 to 6:Memory CPU load size histogram:Memory:1
0:6 to 8:Memory CPU load size histogram:Memory:1
0:8 to 10:Memory CPU load size histogram:Memory:1
0:10 to 12:Memory CPU load size histogram:Memory:1
0:12 to 14:Memory CPU load size histogram:Memory:1
0:14 to 16:Memory CPU load size histogram:Memory:1
0:overflows:Memory CPU load size histogram:Memory:1
:Memory CPU load size histogram:Memory:1
243743511:ROA number of cycles spent waiting for load responses:ROA:0
1:ROA total number of output full:ROA:0
2565160:ROA total number of stalled cycles:ROA:0
344871603:ROA total number of committed instructions:ROA:0
35240785:ROA number of stores sent:ROA:0
35241435:ROA number of stores total:ROA:0
35767668:ROA number of loads sent:ROA:0
92295283:ROA number of loads total:ROA:0
```

```
0:setup−task−io−requests:Cache:2
71008453:requests−processed:Cache:2
71008453:acks−sent:Cache:2
7994975:requests−sent:Cache:2
7994975:acks−received:Cache:2
71008453:requests−received:Cache:2
22.0646:miss−time:Cache:2
0:write_noncompulsory_miss:Cache:2
0:write_compulsory_miss:Cache:2
905377:write_half_miss:Cache:2
897348:write_miss:Cache:2
0:write_shared_hit:Cache:2
0:write_exclusive_hit:Cache:2
33438060:write_hit:Cache:2
0:read_noncompulsory_miss:Cache:2
0:read_compulsory_miss:Cache:2
1101851:read_half_miss:Cache:2
4976888:read_miss:Cache:2
0:read_shared_hit:Cache:2
0:read_exclusive_hit:Cache:2
29688929:read_hit:Cache:2
0:setup−task−io−requests:Cache:3
7994975:requests−processed:Cache:3
7994975:acks−sent:Cache:3
44270:requests−sent:Cache:3
44270:acks−received:Cache:3
7994975:requests−received:Cache:3
200.359:miss−time:Cache:3
0:write_noncompulsory_miss:Cache:3
0:write_compulsory_miss:Cache:3
0:write_half_miss:Cache:3
0:write_miss:Cache:3
0:write_shared_hit:Cache:3
0:write_exclusive_hit:Cache:3
2120739:write_hit:Cache:3
0:read_noncompulsory_miss:Cache:3
0:read_compulsory_miss:Cache:3
2560:read_half_miss:Cache:3
33901:read_miss:Cache:3
0:read_shared_hit:Cache:3
0:read_exclusive_hit:Cache:3
5837775:read_hit:Cache:3
206.554:access−time:DRAM:5
35016:open:DRAM:5
35016:precharge:DRAM:5
10369:write:DRAM:5
33901:read:DRAM:5
0:AvgDataSize:Bus:4
```

```
0:Ack sent:Bus:4
88540:Ack received:Bus:4
88540:Requests sent:Bus:4
0:Requests received:Bus:4
Clear count: 0
Forward migration count: 0
Backward migration count: 0
Total footprint: 0
```

Listing B.1: TaskSim results file

Appendix C

McPAT configuration file

```

<?xml version="1.0" ?>
<component id="root" name="root">
  <component id="system" name="system">

    <param name="number_of_cores" value="1"/>
    <param name="number_of_L1Directories" value="0"/>
    <param name="number_of_L2Directories" value="0"/>
    <param name="number_of_L2s" value="1"/>
    <!--1 Private, 0 shared/coherent -->
    <param name="Private_L2" value="1"/>
    <param name="number_of_L3s" value="0"/>
    <param name="number_of_NoCs" value="0"/>
    <!--1 means homo -->
    <param name="homogeneous_cores" value="0"/>
    <param name="homogeneous_L2s" value="0"/>
    <param name="homogeneous_L1Directories" value="0"/>
    <param name="homogeneous_L2Directories" value="0"/>
    <param name="homogeneous_L3s" value="0"/>
    <!--cache coherence hardware -->
    <param name="homogeneous_ccs" value="1"/>
    <param name="homogeneous_NoCs" value="0"/>
    <param name="core_tech_node" value="45"/><!-- nm -->
    <param name="target_core_clockrate" value="1000"/> <!-- MHz -->
    <param name="temperature" value="380"/> <!-- Kelvin -->
    <param name="number_cache_levels" value="2"/>
    <!--0: aggressive wire technology; 1: conservative wire technology -->
    <param name="interconnect_projection_type" value="0"/>
    <!--0: HP(High Performance Type); 1: LSTP(Low standby power) 2: LOP (
    Low Operating Power) -->
    <param name="device_type" value="0"/>
    <param name="longer_channel_device" value="1"/>
  <!-- 0 not enabled; 1 enabled -->
  <param name="power_gating" value="1"/>
  <param name="machine_bits" value="64"/>
  <param name="virtual_address_width" value="64"/>
  <param name="physical_address_width" value="52"/>
  <param name="virtual_memory_page_size" value="8192"/>
  <stat name="total_cycles" value="247189664"/>
  <stat name="idle_cycles" value="100"/>
  <stat name="busy_cycles"  value="247189564"/>

  <!-- *********************** cores ****************** -->
  <component id="core0" name="core0">
    <!-- Core property -->
    <param name="clock_rate" value="1000"/>
    <!-- 0 means using ITRS default vdd -->
    <param name="vdd" value="0"/>
    <!-- "-1" means using default power gating virtual power supply
  voltage constrained by technology and computed automatically -->
    <param name="power_gating_vcc" value="-1"/>
            <!-- for cores with unknown timing, set to 0 to force off the
  opt flag -->
    <param name="opt_local" value="0"/>
    <param name="instruction_length" value="32"/>
    <param name="opcode_width" value="16"/>
    <param name="x86" value="1"/>
    <param name="micro_opcode_width" value="8"/>
    <!-- inorder/OoO; 1 inorder; 0 OOO-->
    <param name="machine_type" value="0"/>
    <param name="number_hardware_threads" value="1"/>
    <param name="fetch_width" value="4"/>
    <param name="number_instruction_fetch_ports" value="1"/>
    <param name="decode_width" value="4"/>
    <param name="issue_width" value="4"/>
    <param name="peak_issue_width" value="6"/>
    <!-- commit_width determines the number of ports of register files
  -->
    <param name="commit_width" value="4"/>
    <param name="fp_issue_width" value="2"/>
    <param name="prediction_width" value="1"/>
    <param name="pipelines_per_core" value="1,1"/>
    <param name="pipeline_depth" value="31,31"/>
    <param name="ALU_per_core" value="6"/>
    <param name="MUL_per_core" value="1"/>
    <!-- For MUL and Div -->
    <param name="FPU_per_core" value="2"/>
    <!-- buffer between IF and ID stage -->
    <param name="instruction_buffer_size" value="32"/>
    <!-- buffer between ID and sche/exe stage -->
```

```
      <param name="decoded_stream_buffer_size" value="16"/>
      <!-- 0 PHYREG based, 1 RSBASED-->
      <param name="instruction_window_scheme" value="0"/>
      <!-- McPAT support 2 types of OoO cores, RS based and physical reg
    based-->
      <param name="instruction_window_size" value="64"/>
      <param name="fp_instruction_window_size" value="64"/>
      <!-- the instruction issue Q as in Alpha 21264; The RS as in Intel P6
     -->
      <param name="ROB_size" value="192"/>
      <!-- each in-flight instruction has an entry in ROB -->
      <!-- registers -->
      <!-- X86-64 has 16GPR -->
      <param name="archi_Regs_IRF_size" value="16"/>
      <!-- MMX + XMM -->
      <param name="archi_Regs_FRF_size" value="32"/>
      <!--  if OoO processor, phy_reg number is needed for renaming logic,
      renaming logic is for both integer and floating point insts.  -->
      <param name="phy_Regs_IRF_size" value="256"/>
      <param name="phy_Regs_FRF_size" value="256"/>
      <!-- rename logic -->
      <param name="rename_scheme" value="0"/>
      <param name="register_windows_size" value="0"/>
      <param name="LSU_order" value="out-of-order"/>
      <param name="store_buffer_size" value="96"/>
      <param name="load_buffer_size" value="48"/>
      <!-- number of ports refer to sustain-able concurrent memory accesses
     -->
      <param name="memory_ports" value="1"/>
      <param name="RAS_size" value="16"/>
      <stat name="total_instructions" value="246614721"/>
      <stat name="load_instructions" value="45505417"/>
      <stat name="store_instructions" value="11327085"/>
      <stat name="committed_instructions" value="246614721"/>
      <stat name="pipeline_duty_cycle" value="1"/>
      <stat name="total_cycles" value="247189664"/>
      <stat name="idle_cycles" value="100"/>
      <stat name="busy_cycles"  value="247189564"/>
      <stat name="IFU_duty_cycle" value="0.25"/>
      <stat name="LSU_duty_cycle" value="0.25"/>
      <stat name="MemManU_I_duty_cycle" value="0.25"/>
      <stat name="MemManU_D_duty_cycle" value="0.25"/>
      <stat name="ALU_duty_cycle" value="1"/>
      <stat name="MUL_duty_cycle" value="0.3"/>
      <stat name="FPU_duty_cycle" value="0.3"/>
      <stat name="ALU_cdb_duty_cycle" value="1"/>
      <stat name="MUL_cdb_duty_cycle" value="0.3"/>
      <stat name="FPU_cdb_duty_cycle" value="0.3"/>
      <param name="number_of_BPT" value="0"/>

      <component id="core0.icache" name="icache">
        <!-- the parameters are capacity,block_width, associativity, bank,
    throughput w.r.t. core clock, latency w.r.t. core clock,output_width,
    cache policy, -->
        <!-- cache_policy;//0 no write or write-though with non-write
    allocate;1 write-back with write-allocate -->
        <param name="icache_config" value="131072,32,8,1,8,3,32,1"/>
        <!-- cache controller buffer sizes: miss_buffer_size(MSHR),
    fill_buffer_size,prefetch_buffer_size,wb_buffer_size-->
        <param name="buffer_sizes" value="16, 16, 16, 16"/>
      </component>

      <component id="core0.dcache" name="dcache">
        <param name="dcache_config" value="4096, 32, 4, 1, 4, 2, 32, 1"/>
        <param name="buffer_sizes" value="128,128,128,128"/>
        <stat name="read_accesses" value="22596345"/>
        <stat name="write_accesses" value="12902258"/>
        <stat name="read_misses" value="1578157"/>
        <stat name="write_misses" value="3528872"/>
        <stat name="conflicts" value="0"/>
      </component>
      <param name="number_of_BTB" value="0"/>
      <component id="system.core0.BTB" name="BTB">
        <param name="BTB_config" value="5120,4,2,1,1,3"/>
      </component>
  </component>

    <component id="system.L20" name="L20">
        <param name="L2_config" value="524288, 64, 16, 1, 16, 20, 64, 1"/>
        <param name="buffer_sizes" value="128,128,128,128"/>
        <!-- number of r, w, and rw ports -->
        <param name="ports" value="1,1,1"/>
        <param name="clockrate" value="1000"/>
        <param name="vdd" value="0"/><!-- 0 means using ITRS default vdd
    -->
        <param name="power_gating_vcc" value="-1"/>
        <param name="device_type" value="0"/>
        <stat name="read_accesses" value="5107029"/>
        <stat name="write_accesses" value="3595533"/>
        <stat name="read_misses" value="6327"/>
        <stat name="write_misses" value="0"/>
        <stat name="conflicts" value="0"/>
        <stat name="duty_cycle" value="0.5"/>
    </component>


```

```
<!--************************************************************************-->

    <component id="system.mc" name="mc">
      <param name="number_mcs" value="0"/>
    </component>
<!--************************************************************************-->

    <component id="system.niu" name="niu">
      <param name="number_units" value="0"/>
    </component>
<!--************************************************************************-->

    <component id="system.pcie" name="pcie">
      <param name="number_units" value="0"/>
    </component>
<!--************************************************************************-->

    <component id="system.flashc" name="flashc">
        <param name="number_flashcs" value="0"/>
    </component>
<!--************************************************************************-->

    </component>
</component>
```

Listing C.1: McPAT configuration file

Appendix D

McPAT results file

```
McPAT (version 1.3 of Feb, 2015) results (current print level is 3, please
    increase print level to see the details in components):
*****************************************************************************************
  Technology 45 nm
  Using Long Channel Devices When Appropriate
  Interconnect metal projection= aggressive interconnect technology
    projection
  Core clock Rate(MHz) 1000

Processor:
  Area = 23.2491 mm^2
  Peak Power = 10.0036 W
  Total Leakage = 4.72842 W
  Peak Dynamic = 5.27521 W
  Subthreshold Leakage = 4.4027 W
  Subthreshold Leakage with power gating = 2.04596 W
  Gate Leakage = 0.325718 W
  Runtime Dynamic = 1.20404 W

  Total Cores: 1 cores
  Device Type= ITRS high performance device type
    Area = 23.2491 mm^2
    Peak Dynamic = 5.27521 W
    Subthreshold Leakage = 4.4027 W
    Subthreshold Leakage with power gating = 2.04596 W
    Gate Leakage = 0.325718 W
    Runtime Dynamic = 1.20404 W

Core:
      Area = 23.2491 mm^2
      Peak Dynamic = 5.27521 W
      Subthreshold Leakage = 4.4027 W
      Subthreshold Leakage with power gating = 2.04596 W
      Gate Leakage = 0.325718 W
      Runtime Dynamic = 1.20404 W

      Instruction Fetch Unit:
        Area = 3.78963 mm^2
        Peak Dynamic = 0.741947 W
        Subthreshold Leakage = 0.569678 W
        Subthreshold Leakage with power gating = 0.279599 W
        Gate Leakage = 0.0342316 W
        Runtime Dynamic = 0.162322 W

          Instruction Cache:
            Area = 1.31432 mm^2
            Peak Dynamic = 0.150314 W
            Subthreshold Leakage = 0.184268 W
            Subthreshold Leakage with power gating = 0.102491 W
            Gate Leakage = 0.00843643 W
            Runtime Dynamic = 3.58793e-09 W

          Branch Target Buffer:
            Area = 0.478297 mm^2
            Peak Dynamic = 0.0219604 W
            Subthreshold Leakage = 0.0252612 W
            Subthreshold Leakage with power gating = 0.0148829 W
            Gate Leakage = 0.000985866 W
            Runtime Dynamic = 2.89928e-10 W

          Branch Predictor:
            Area = 0.0142151 mm^2
            Peak Dynamic = 0.00360149 W
            Subthreshold Leakage = 0.000765153 W
            Subthreshold Leakage with power gating = 0.000414063 W
            Gate Leakage = 6.53926e-05 W
            Runtime Dynamic = 3.6871e-11 W

          Instruction Buffer:
            Area = 0.0245435 mm^2
            Peak Dynamic = 0.161309 W
            Subthreshold Leakage = 0.0011165 W
            Subthreshold Leakage with power gating = 0.000590575 W
            Gate Leakage = 5.69632e-05 W
            Runtime Dynamic = 0.0268189 W

          Instruction Decoder:
            Area = 1.85799 mm^2
            Peak Dynamic = 0.358719 W
            Subthreshold Leakage = 0.325606 W
            Subthreshold Leakage with power gating = 0.146523 W
            Gate Leakage = 0.0185411 W
            Runtime Dynamic = 0.0894599 W

      Renaming Unit:
        Area = 0.23512 mm^2
        Peak Dynamic = 0.414558 W
        Subthreshold Leakage = 0.0349715 W
        Subthreshold Leakage with power gating = 0.0160445 W
        Gate Leakage = 0.00629456 W
        Runtime Dynamic = 0.184171 W

      Load Store Unit:
        Area = 1.25411 mm^2
        Peak Dynamic = 0.362141 W
        Subthreshold Leakage = 0.139991 W
        Subthreshold Leakage with power gating = 0.0632416 W
        Gate Leakage = 0.0133767 W
        Runtime Dynamic = 0.121196 W

          Data Cache:
            Area = 0.864812 mm^2
            Peak Dynamic = 0.279442 W
            Subthreshold Leakage = 0.0910443 W
            Subthreshold Leakage with power gating = 0.0412158 W
            Gate Leakage = 0.00610289 W
            Runtime Dynamic = 0.0228989 W

          LoadQ:
            Area = 0.083675 mm^2
            Peak Dynamic = 0.0164852 W
            Subthreshold Leakage = 0.00692433 W
            Subthreshold Leakage with power gating = 0.00311595 W
            Gate Leakage = 0.000499133 W
            Runtime Dynamic = 0.0151588 W

          StoreQ:
            Area = 0.100465 mm^2
            Peak Dynamic = 0.0201706 W
            Subthreshold Leakage = 0.00936056 W
            Subthreshold Leakage with power gating = 0.00421225 W
            Gate Leakage = 0.000628884 W
            Runtime Dynamic = 0.0370955 W

      Memory Management Unit:
        Area = 0.166096 mm^2
        Peak Dynamic = 0.119143 W
```

```
        Subthreshold Leakage = 0.0377489 W
        Subthreshold Leakage with power gating = 0.016987 W
        Runtime Dynamic = 0.115107 W

          Itlb:
            Area = 0.0329161 mm^2
            Peak Dynamic = 0.00201826 W
            Subthreshold Leakage = 0.00254371 W
            Subthreshold Leakage with power gating = 0.00114467 W
            Gate Leakage = 0.000193075 W
            Runtime Dynamic = 6.90441e-11 W

          Dtlb:
            Area = 0.0329161 mm^2
            Peak Dynamic = 0.00201826 W
            Subthreshold Leakage = 0.00254371 W
            Subthreshold Leakage with power gating = 0.00114467 W
            Gate Leakage = 0.000193075 W
            Runtime Dynamic = 6.90441e-11 W

      Execution Unit:
        Area = 8.20112 mm^2
        Peak Dynamic = 2.88025 W
        Subthreshold Leakage = 1.30495 W
        Subthreshold Leakage with power gating = 0.591728 W
        Runtime Dynamic = 0.606439 W

          Integer ALUs (Count: 6 ):
            Area = 0.47087 mm^2
            Peak Dynamic = 0.651519 W
            Subthreshold Leakage = 0.295671 W
            Subthreshold Leakage with power gating = 0.133052 W
            Gate Leakage = 0.0221076 W
            Runtime Dynamic = 0.070378 W

          Floating Point Units (FPUs) (Count: 2 ):
            Area = 4.6585 mm^2
            Peak Dynamic = 0.237683 W
            Subthreshold Leakage = 0.731296 W
            Subthreshold Leakage with power gating = 0.329083 W
            Gate Leakage = 0.0546797 W
            Runtime Dynamic = 0.211134 W

          Complex ALUs (Mul/Div) (Count: 1 ):
            Area = 0.235435 mm^2
            Peak Dynamic = 0.100341 W
            Subthreshold Leakage = 0.147835 W
            Subthreshold Leakage with power gating = 0.066526 W
            Gate Leakage = 0.0110538 W
            Runtime Dynamic = 0.140756 W

    L2
    Area = 3.67669 mm^2
    Peak Dynamic = 0.757168 W
    Subthreshold Leakage = 0.454718 W
    Subthreshold Leakage with power gating = 0.241067 W
    Gate Leakage = 0.0256912 W
    Runtime Dynamic = 0.0148072 W
```

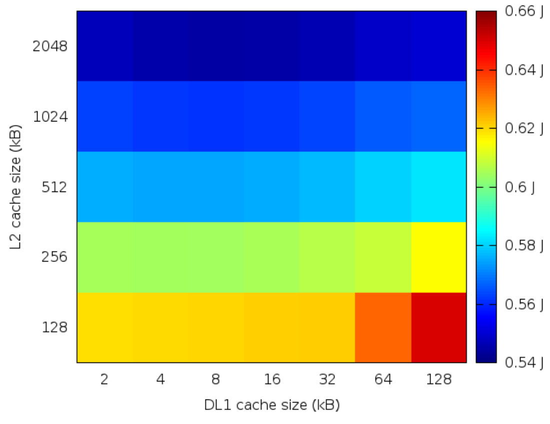
Listing D.1: McPAT results file

Appendix E

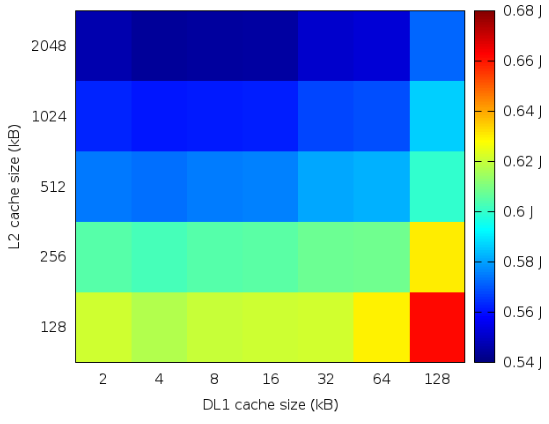
PARSEC benchmark graphics

E.1 Ferret

E.1.1 1 core

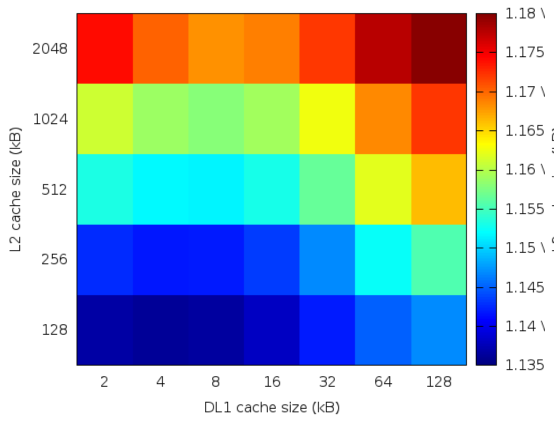


(A) Fixed associativity and latency

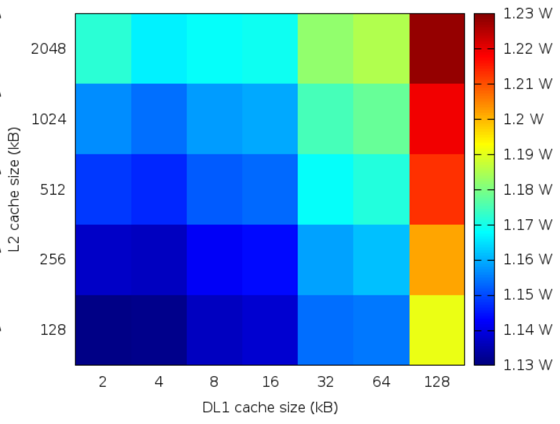


(B) Unfixed associativity and latency

FIGURE E.1: Ferret 1 core processor dynamic energy consumption

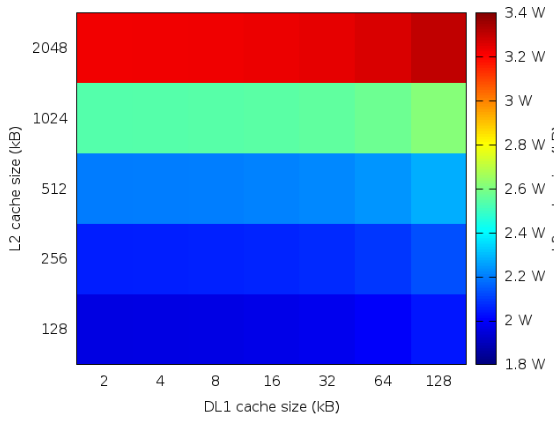


(A) Fixed associativity and latency

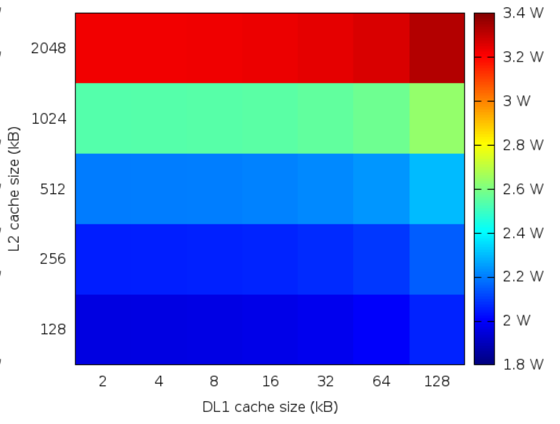


(B) Unfixed associativity and latency

FIGURE E.2: Ferret 1 core processor dynamic power consumption

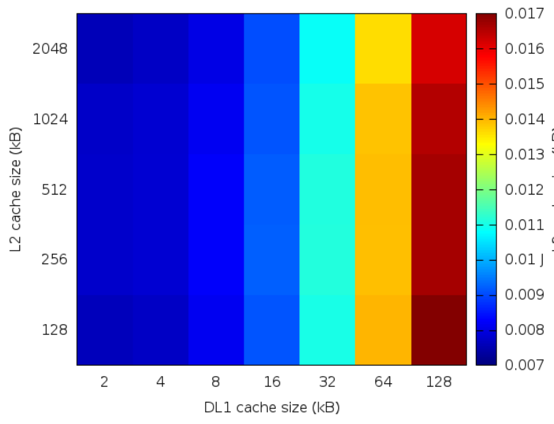


(A) Fixed associativity and latency

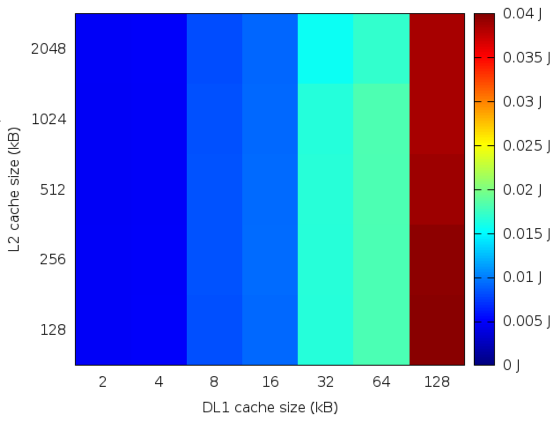


(B) Unfixed associativity and latency

FIGURE E.3: Ferret 1 core processor static power consumption

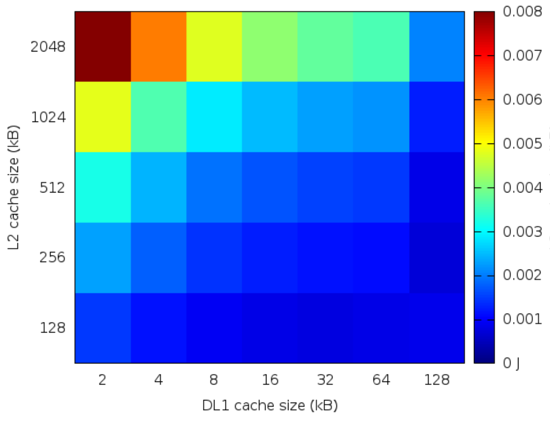


(A) Fixed associativity and latency

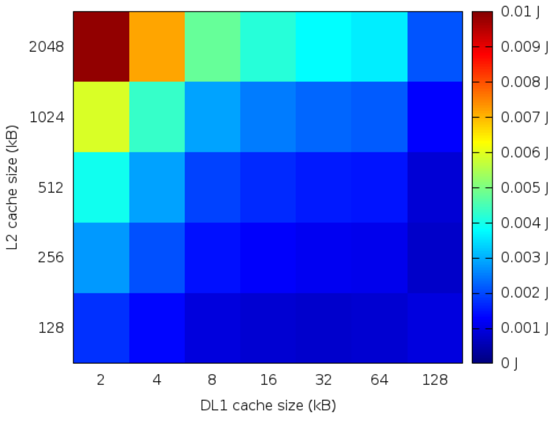


(B) Unfixed associativity and latency

FIGURE E.4: Ferret 1 core DL1 dynamic energy consumption



(A) Fixed associativity and latency

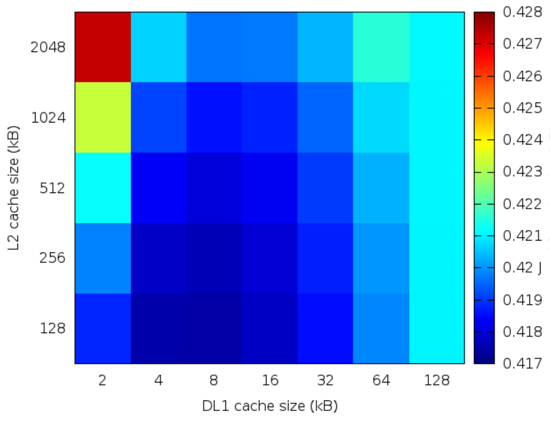


(B) Unfixed associativity and latency

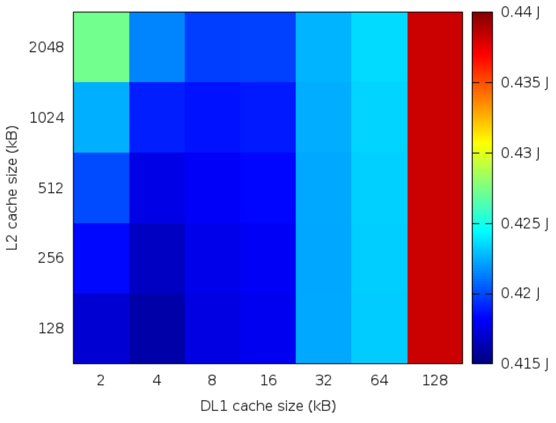
FIGURE E.5: Ferret 1 core L2 dynamic energy consumption

E.2 Blackscholes

E.2.1 1 core

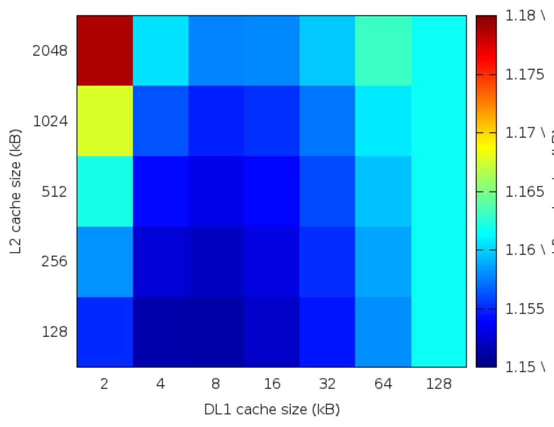


(A) Fixed associativity and latency

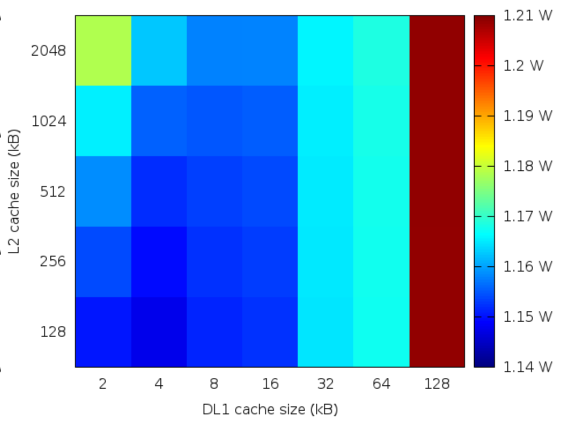


(B) Unfixed associativity and latency

FIGURE E.6: Blackscholes 1 core processor dynamic energy consumption

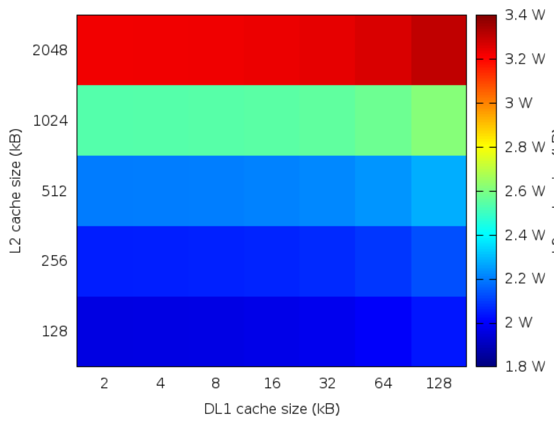


(A) Fixed associativity and latency

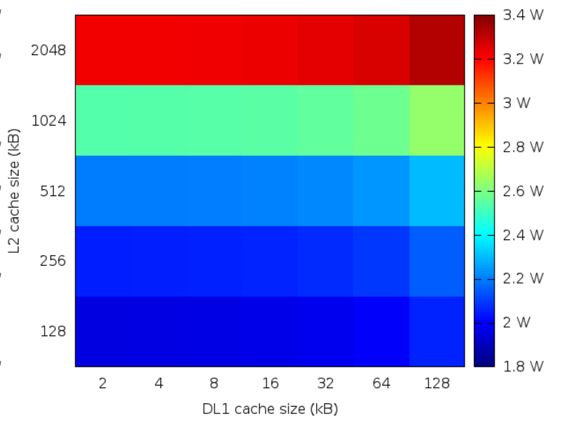


(B) Unfixed associativity and latency

FIGURE E.7: Blackscholes 1 core processor dynamic power consumption

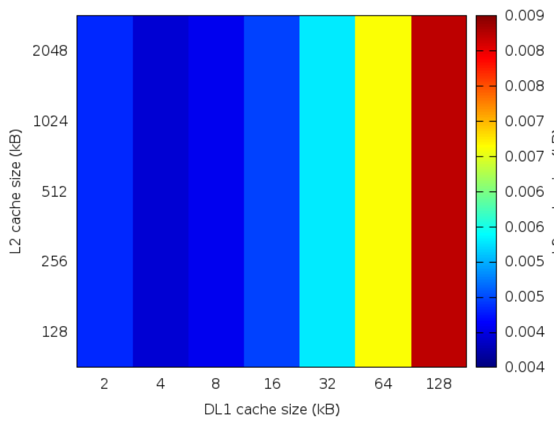


(A) Fixed associativity and latency

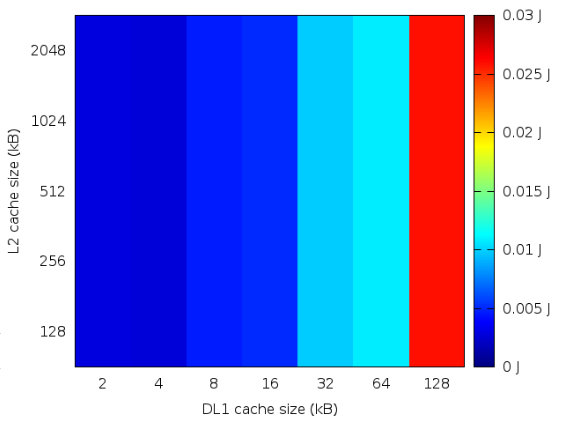


(B) Unfixed associativity and latency

FIGURE E.8: Blackscholes 1 core processor static power consumption

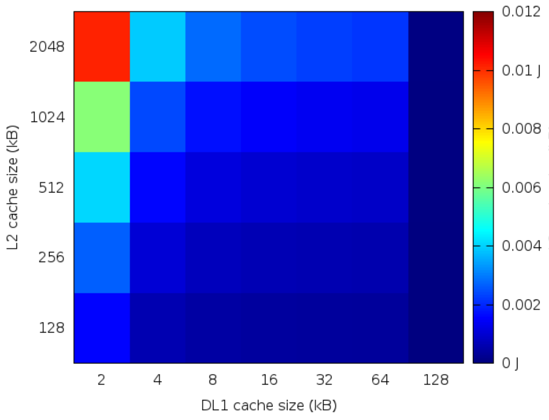


(A) Fixed associativity and latency

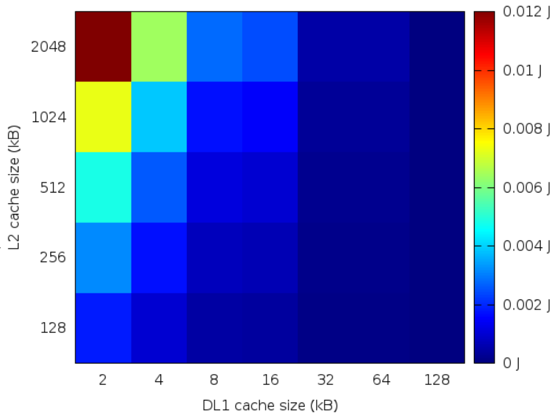


(B) Unfixed associativity and latency

FIGURE E.9: Blackscholes 1 core DL1 dynamic energy consumption



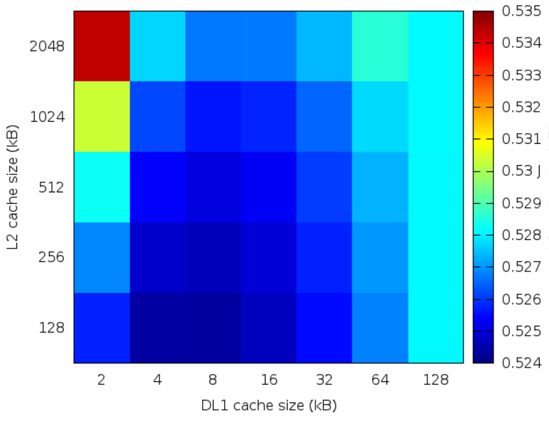
(A) Fixed associativity and latency



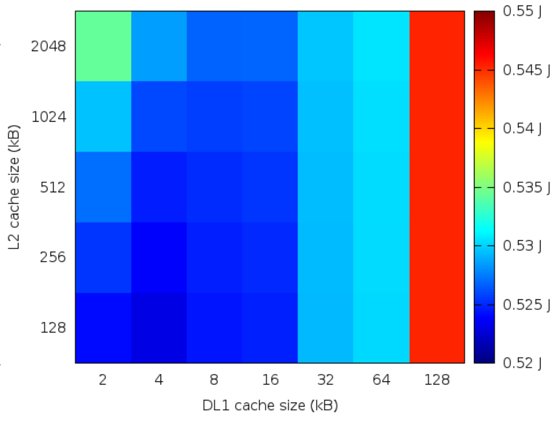
(B) Unfixed associativity and latency

FIGURE E.10: Blackscholes 1 core L2 dynamic energy consumption

E.2.2 4 core

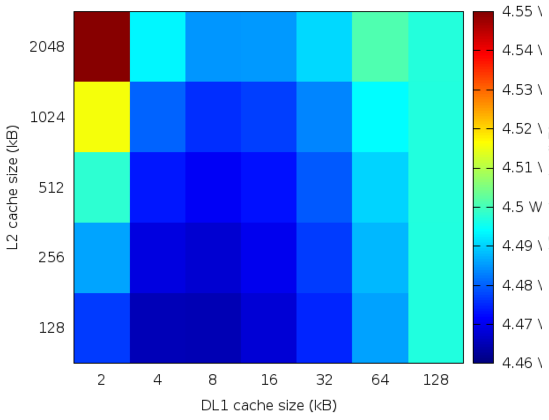


(A) Fixed associativity and latency

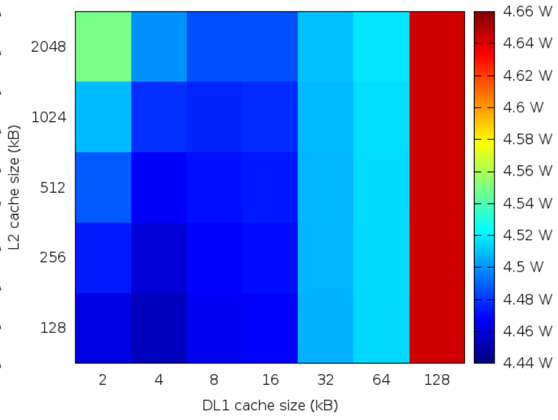


(B) Unfixed associativity and latency

FIGURE E.11: Blackscholes 4 core processor dynamic energy consumption

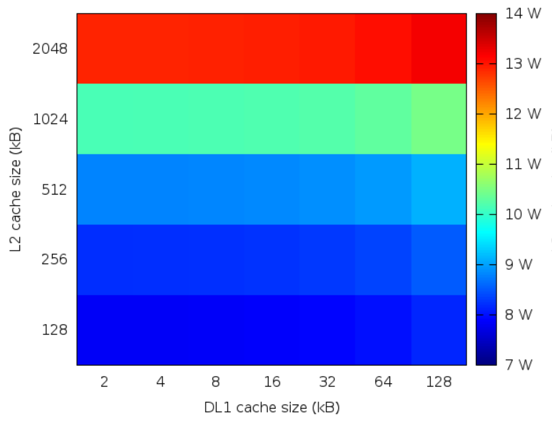


(A) Fixed associativity and latency

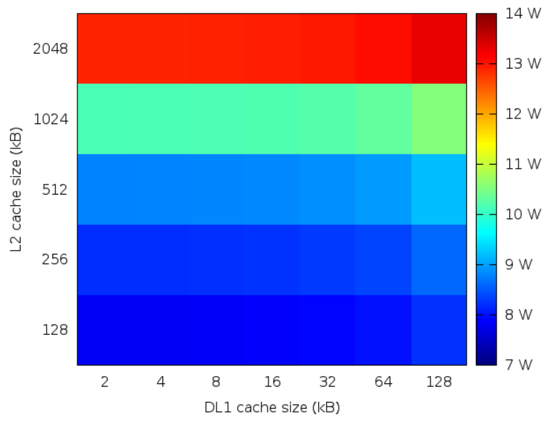


(B) Unfixed associativity and latency

FIGURE E.12: Blackscholes 4 core processor dynamic power consumption

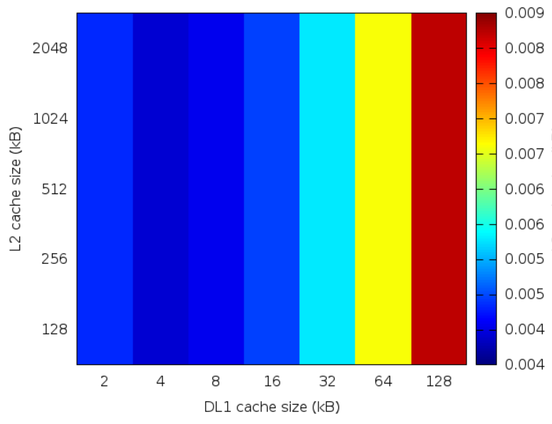


(A) Fixed associativity and latency

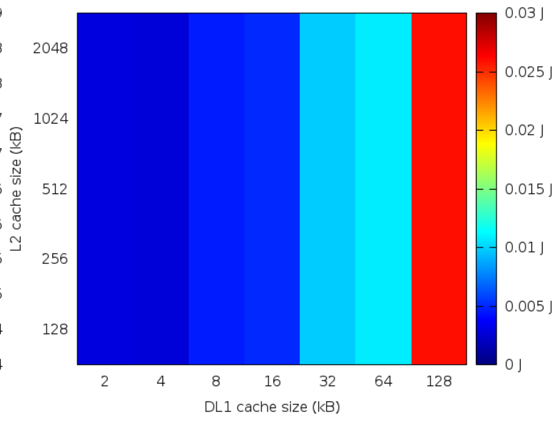


(B) Unfixed associativity and latency

FIGURE E.13: Blackscholes 4 core processor static power consumption

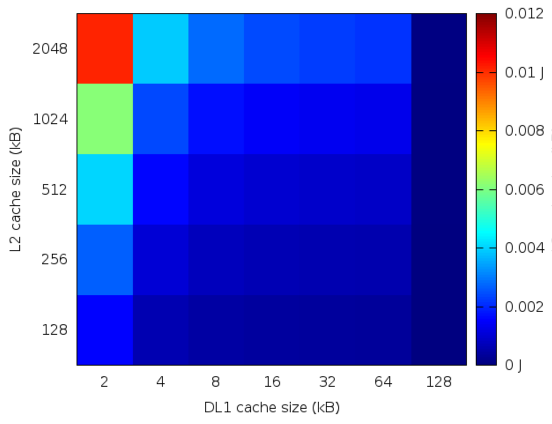


(A) Fixed associativity and latency

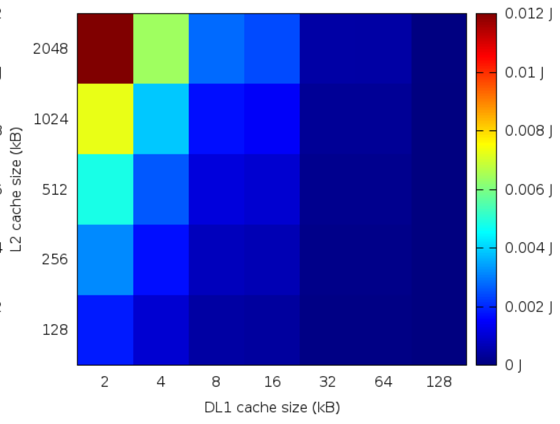


(B) Unfixed associativity and latency

FIGURE E.14: Blackscholes 4 core DL1 dynamic energy consumption



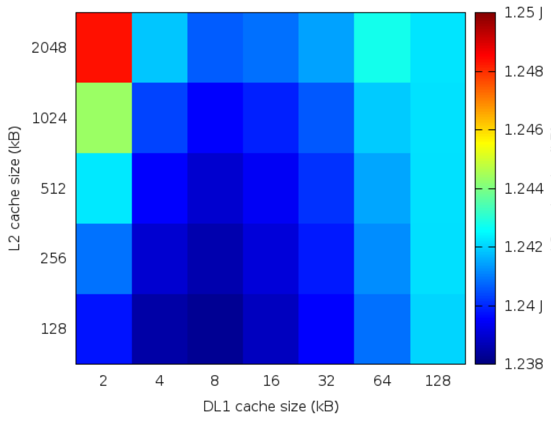
(A) Fixed associativity and latency



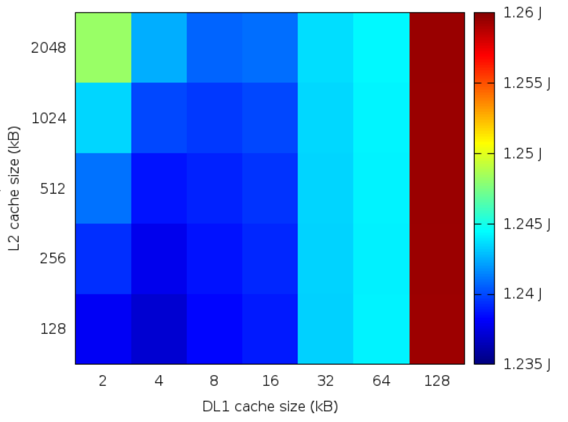
(B) Unfixed associativity and latency

FIGURE E.15: Blackscholes 4 core L2 dynamic energy consumption

E.2.3 16 core

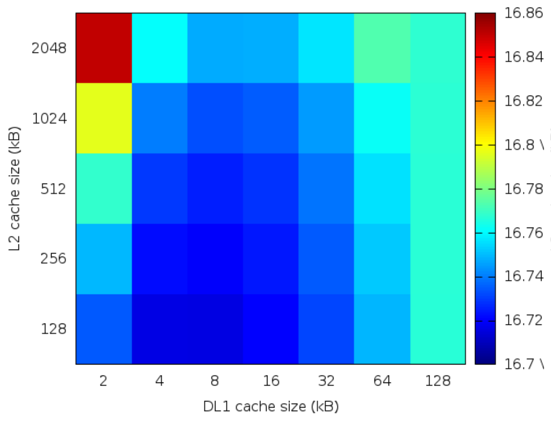


(A) Fixed associativity and latency

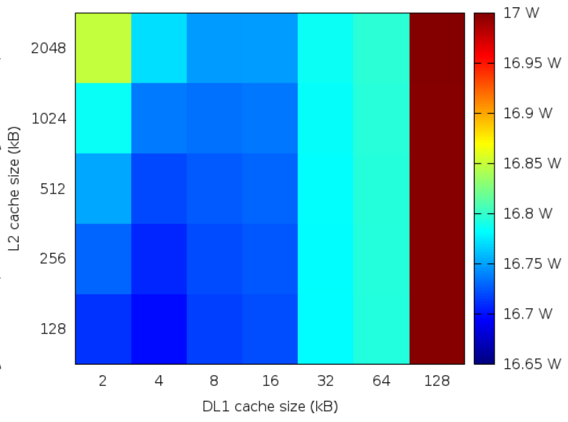


(B) Unfixed associativity and latency

FIGURE E.16: Blackscholes 16 core processor dynamic energy consumption

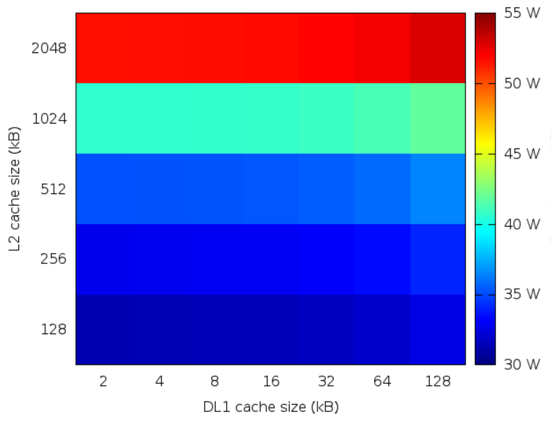


(A) Fixed associativity and latency

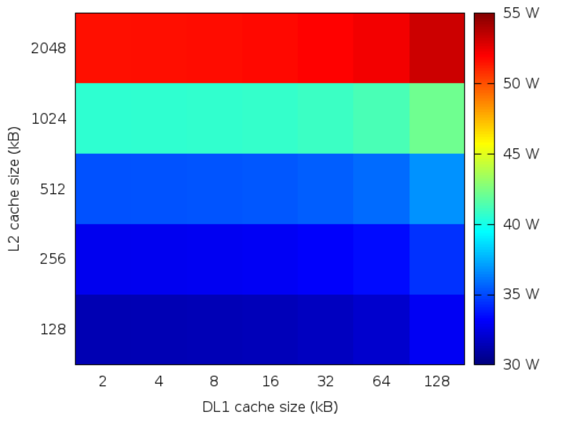


(B) Unfixed associativity and latency

FIGURE E.17: Blackscholes 16 core processor dynamic power consumption

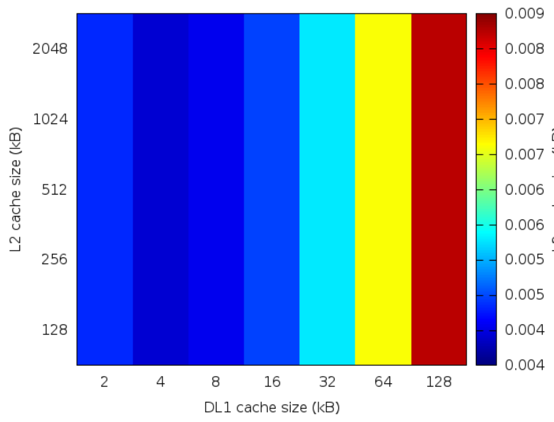


(A) Fixed associativity and latency

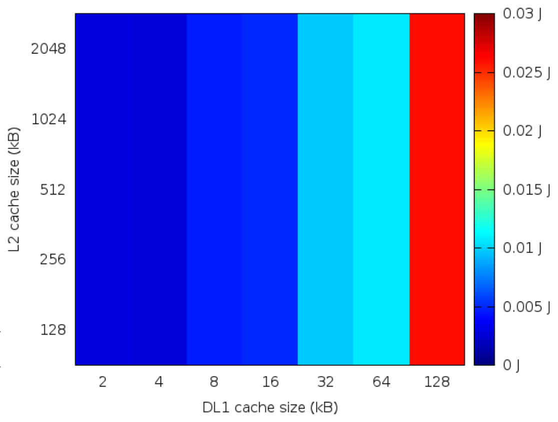


(B) Unfixed associativity and latency

FIGURE E.18: Blackscholes 16 core processor static power consumption

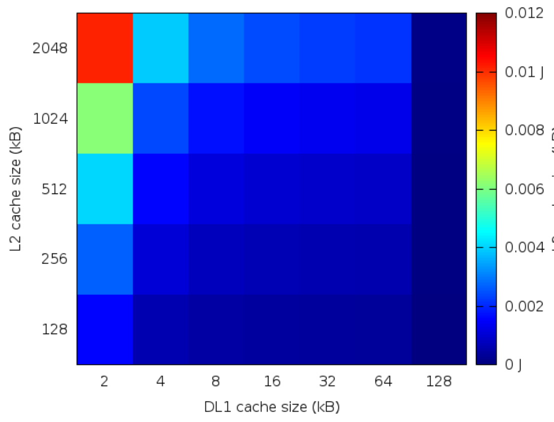


(A) Fixed associativity and latency

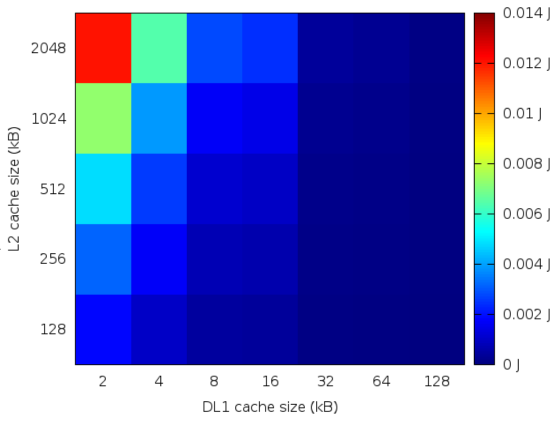


(B) Unfixed associativity and latency

Figure E.19: Blackscholes 16 core DL1 dynamic energy consumption



(A) Fixed associativity and latency

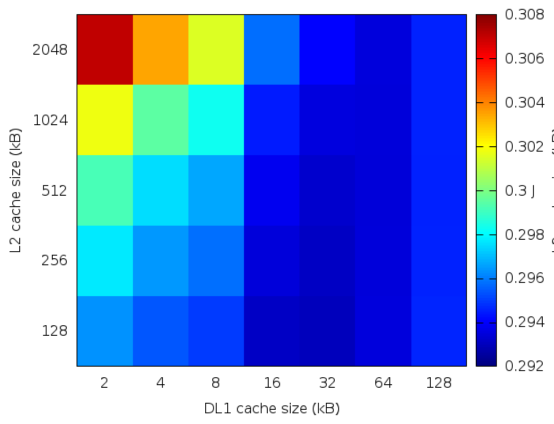


(B) Unfixed associativity and latency

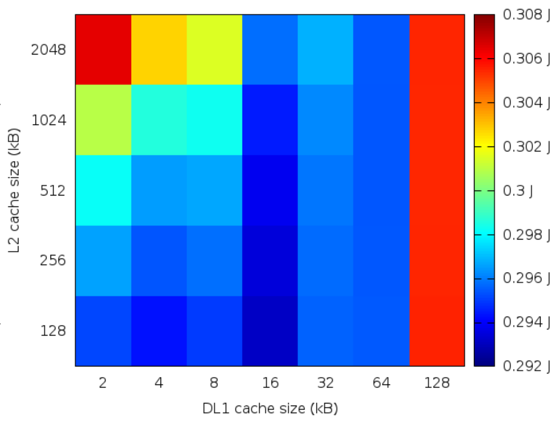
Figure E.20: Blackscholes 16 core L2 dynamic energy consumption

E.3 Swaptions

E.3.1 1 core

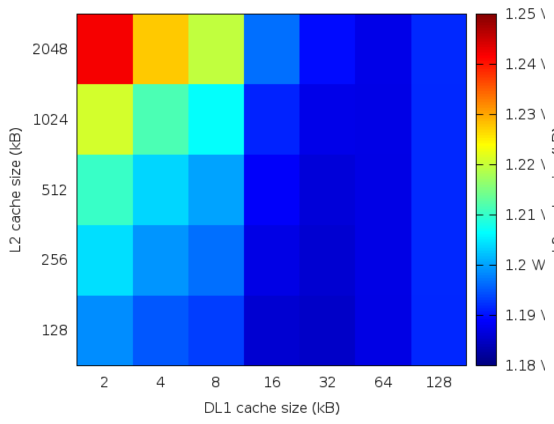


(A) Fixed associativity and latency

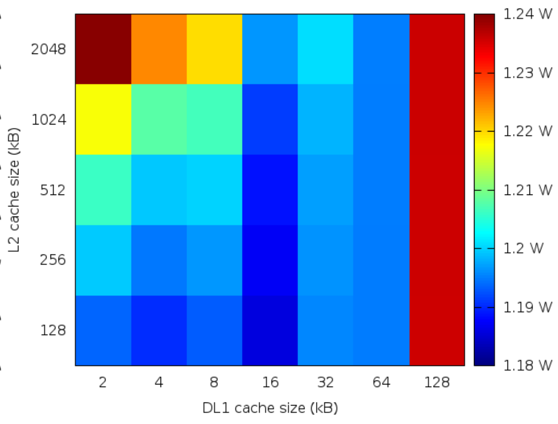


(B) Unfixed associativity and latency

FIGURE E.21: Swaptions 1 core processor dynamic energy consumption

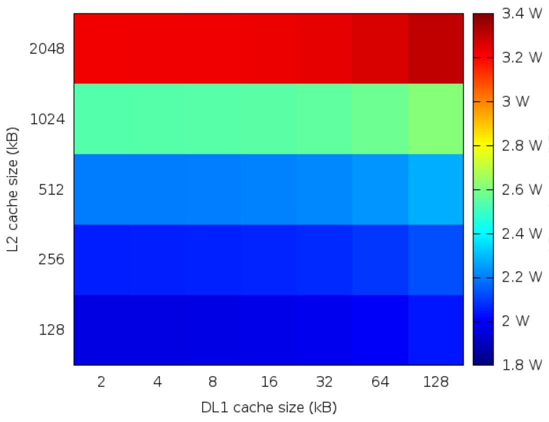


(A) Fixed associativity and latency

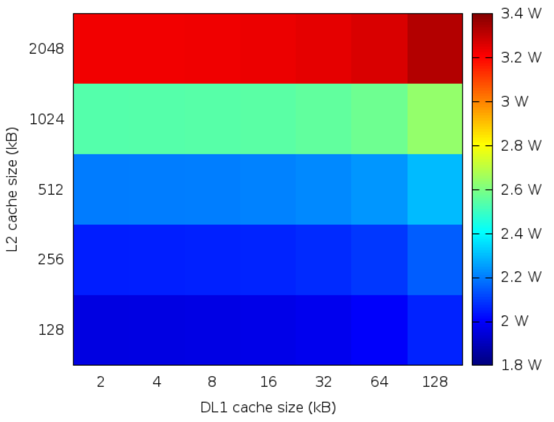


(B) Unfixed associativity and latency

FIGURE E.22: Swaptions 1 core processor dynamic power consumption

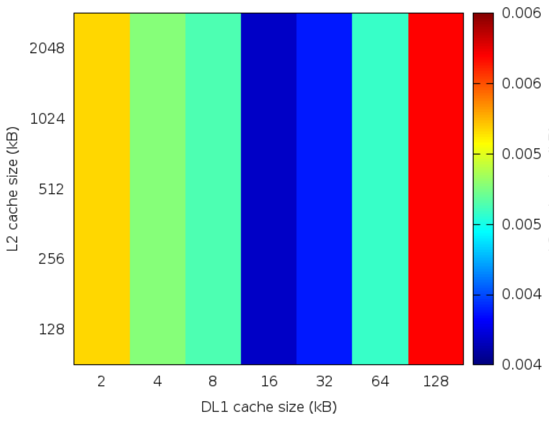


(A) Fixed associativity and latency

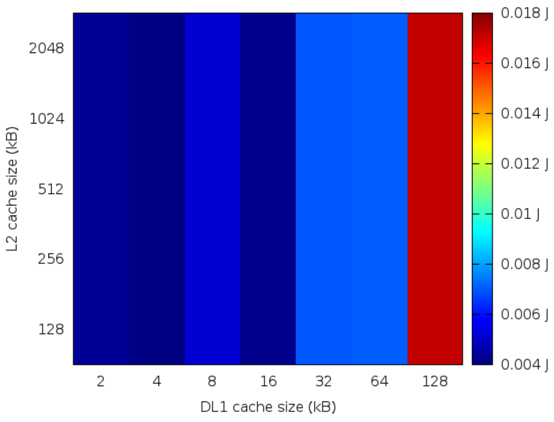


(B) Unfixed associativity and latency

FIGURE E.23: Swaptions 1 core processor static power consumption

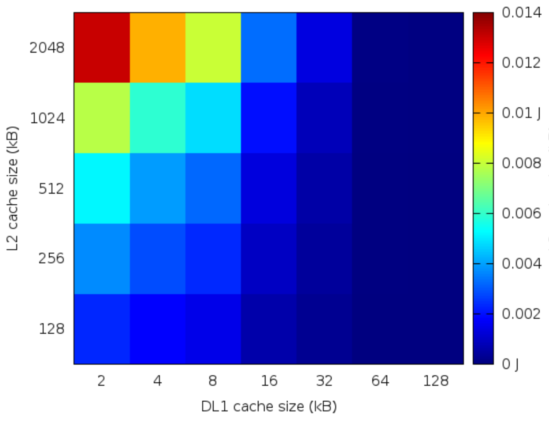


(A) Fixed associativity and latency

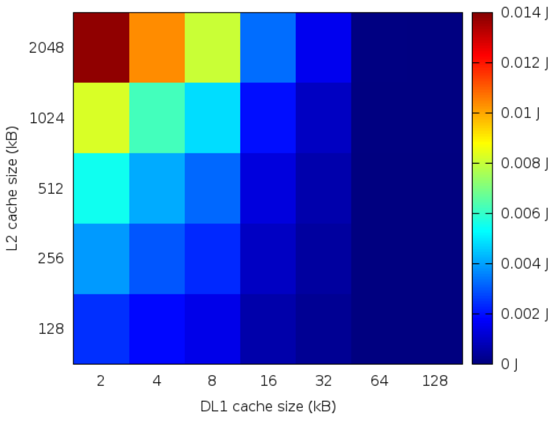


(B) Unfixed associativity and latency

FIGURE E.24: Swaptions 1 core DL1 dynamic energy consumption



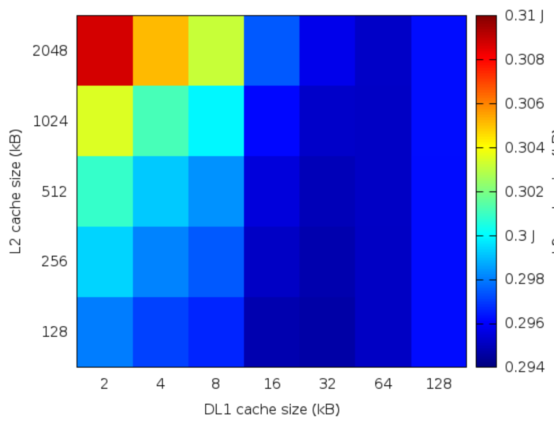
(A) Fixed associativity and latency



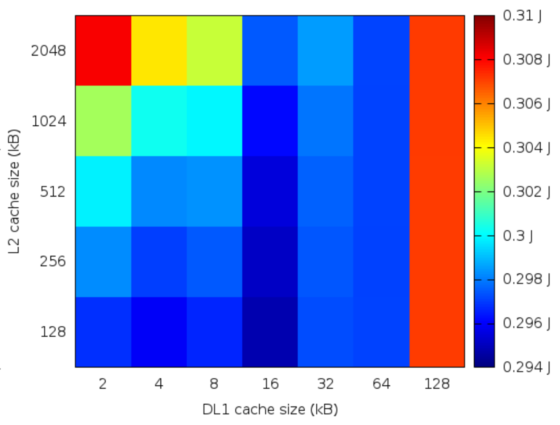
(B) Unfixed associativity and latency

FIGURE E.25: Swaptions 1 core L2 dynamic energy consumption

E.3.2 4 core

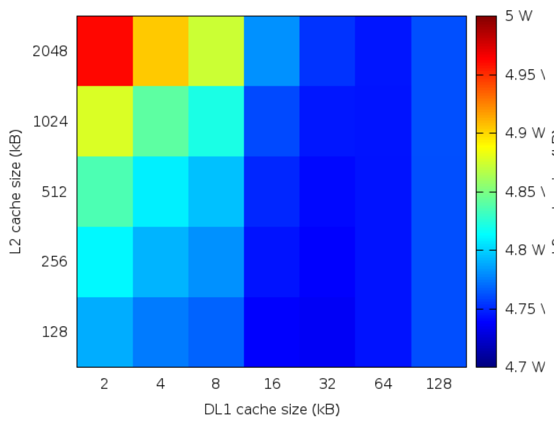


(A) Fixed associativity and latency

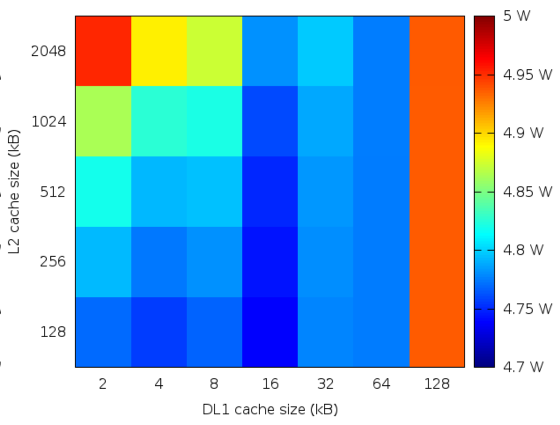


(B) Unfixed associativity and latency

FIGURE E.26: Swaptions 4 core processor dynamic energy consumption

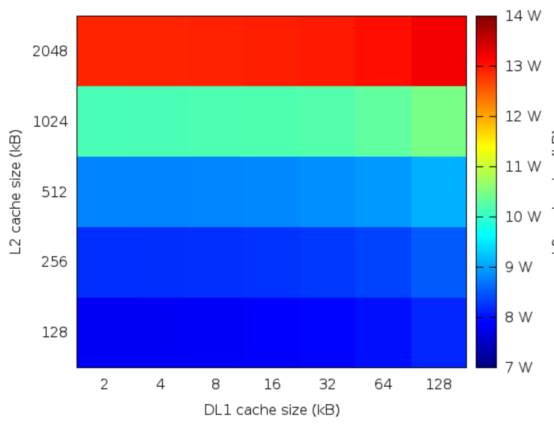


(A) Fixed associativity and latency

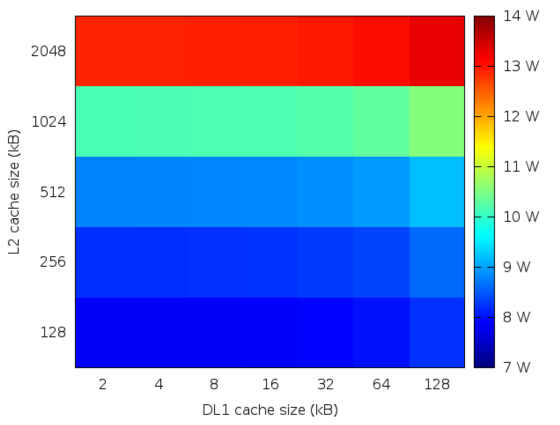


(B) Unfixed associativity and latency

FIGURE E.27: Swaptions 4 core processor dynamic power consumption

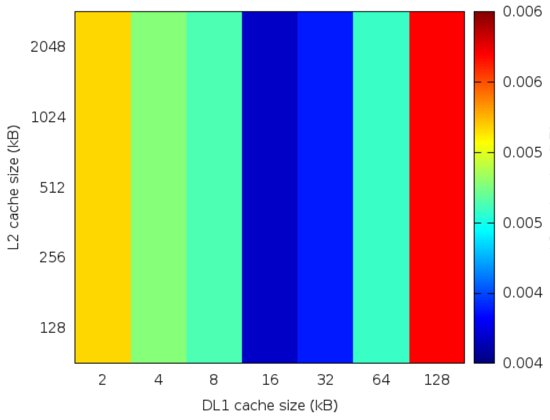


(A) Fixed associativity and latency

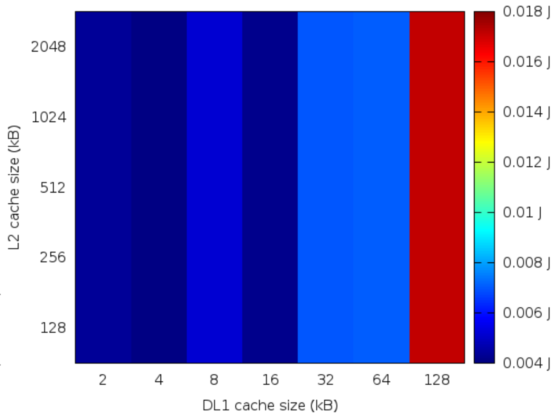


(B) Unfixed associativity and latency

FIGURE E.28: Swaptions 4 core processor static power consumption

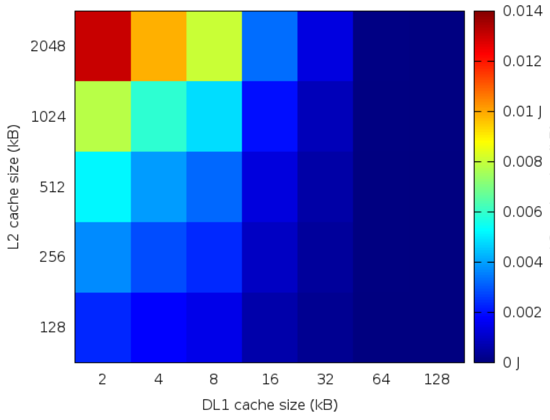


(A) Fixed associativity and latency

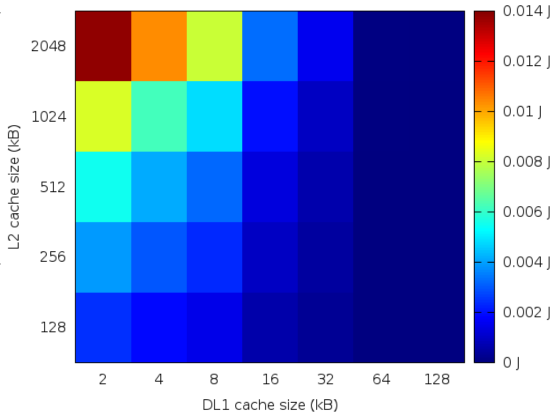


(B) Unfixed associativity and latency

FIGURE E.29: Swaptions 4 core DL1 dynamic energy consumption



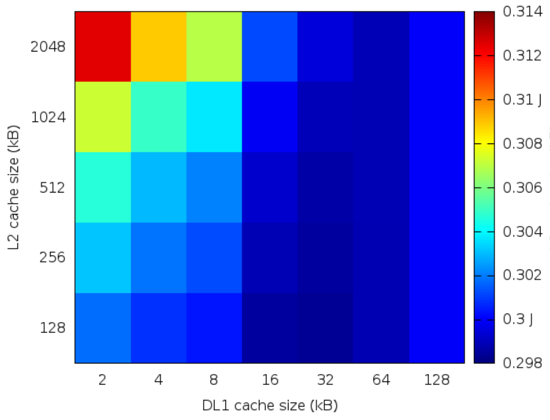
(A) Fixed associativity and latency



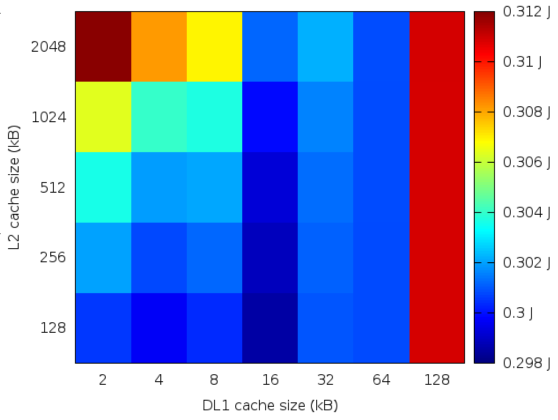
(B) Unfixed associativity and latency

FIGURE E.30: Swaptions 4 core L2 dynamic energy consumption

E.3.3 16 core

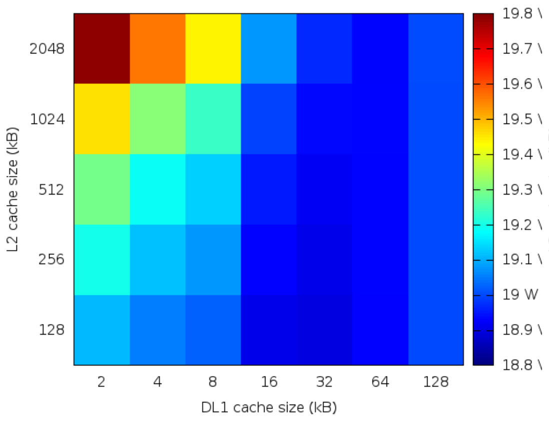


(A) Fixed associativity and latency

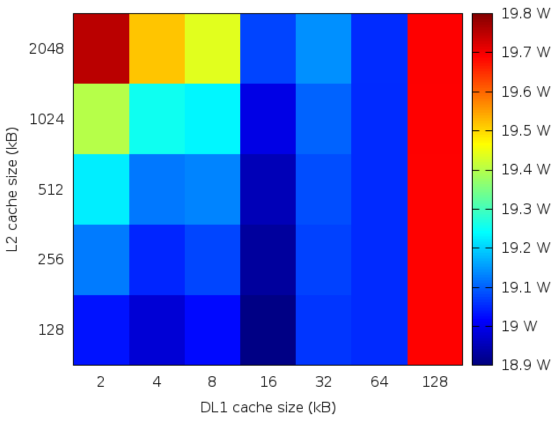


(B) Unfixed associativity and latency

FIGURE E.31: Swaptions 16 core processor dynamic energy consumption

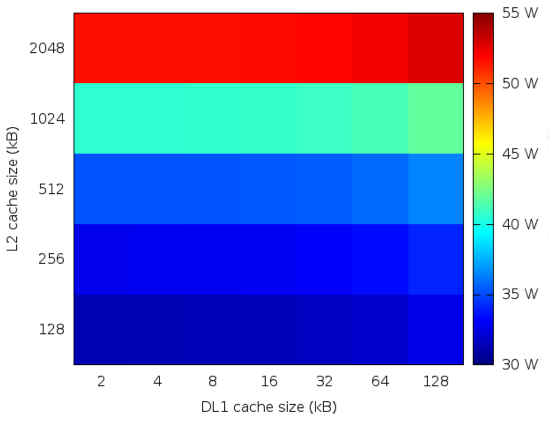


(A) Fixed associativity and latency

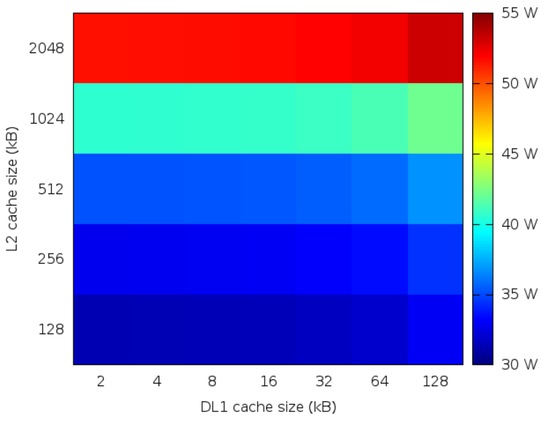


(B) Unfixed associativity and latency

FIGURE E.32: Swaptions 16 core processor dynamic power consumption

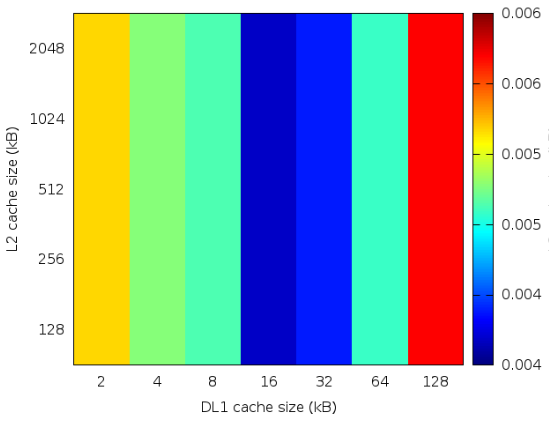


(A) Fixed associativity and latency

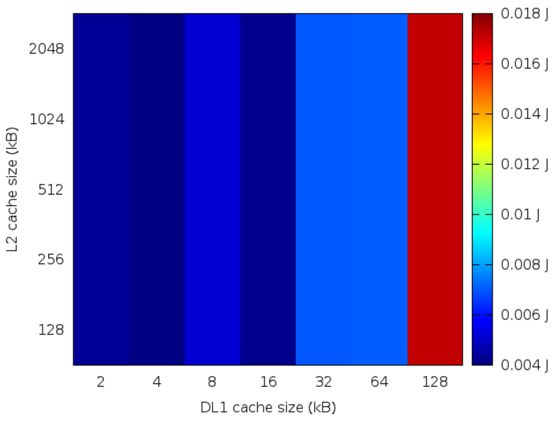


(B) Unfixed associativity and latency

FIGURE E.33: Swaptions 16 core processor static power consumption

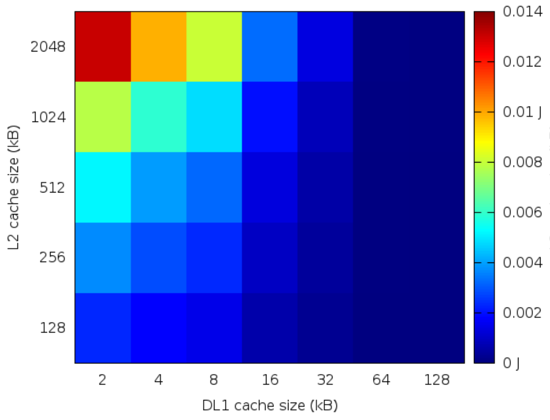


(A) Fixed associativity and latency

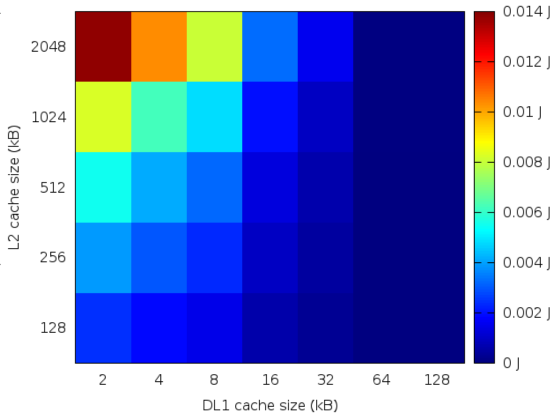


(B) Unfixed associativity and latency

FIGURE E.34: Swaptions 16 core DL1 dynamic energy consumption



(A) Fixed associativity and latency

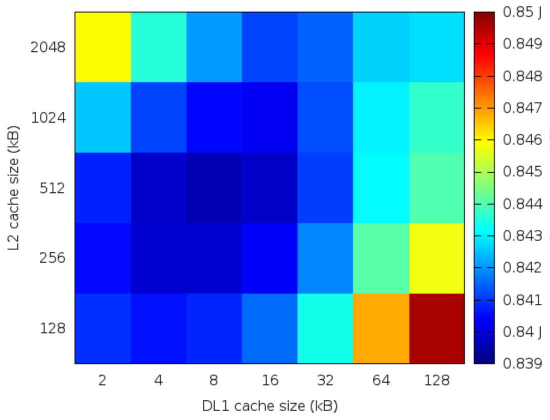


(B) Unfixed associativity and latency

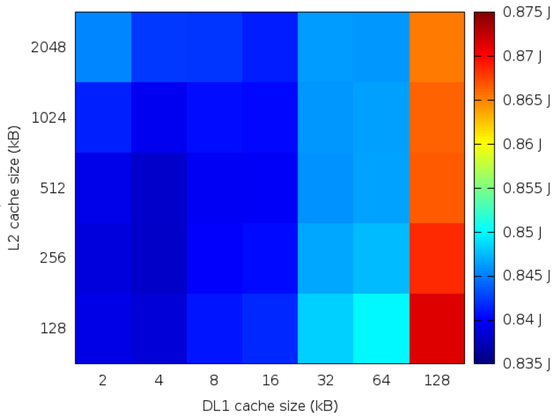
FIGURE E.35: Swaptions 16 core L2 dynamic energy consumption

E.4 Dedup

E.4.1 1 core

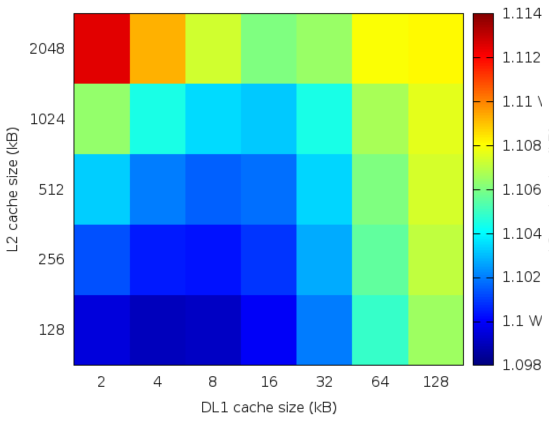


(A) Fixed associativity and latency

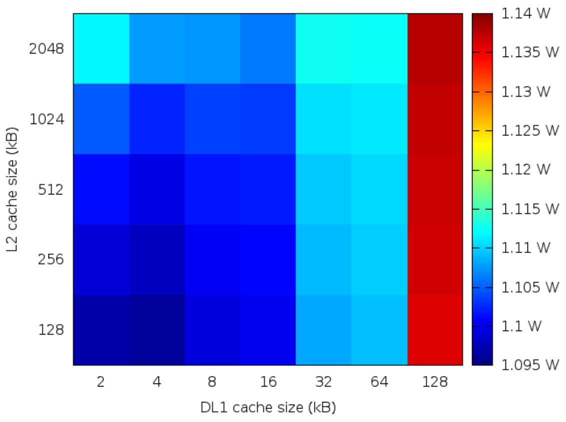


(B) Unfixed associativity and latency

FIGURE E.36: Dedup 1 core processor dynamic energy consumption

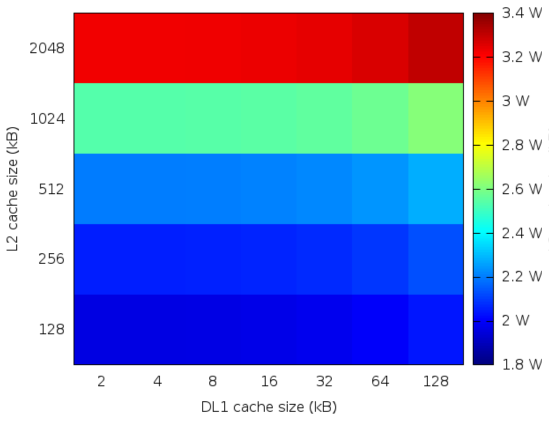


(A) Fixed associativity and latency

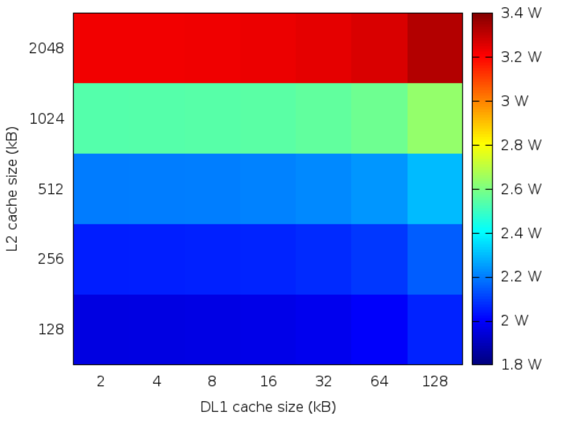


(B) Unfixed associativity and latency

FIGURE E.37: Dedup 1 core processor dynamic power consumption

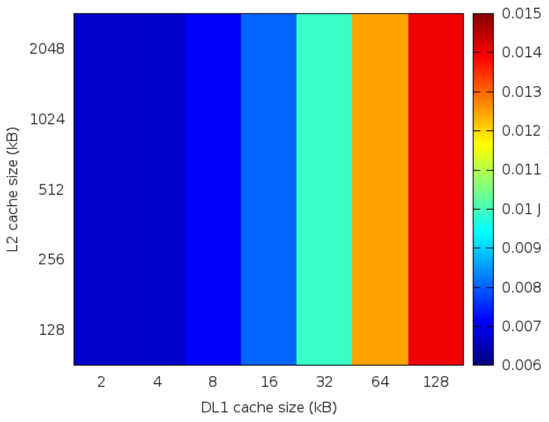


(A) Fixed associativity and latency

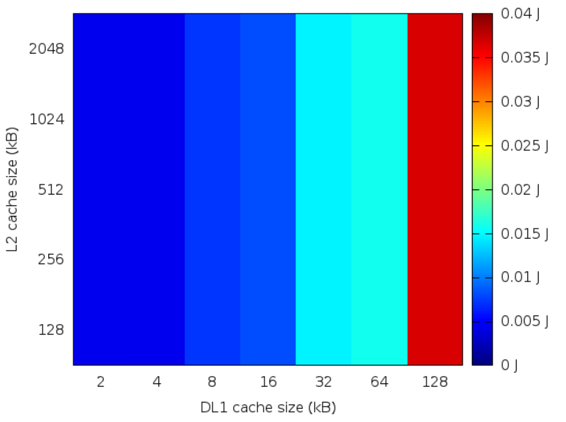


(B) Unfixed associativity and latency

FIGURE E.38: Dedup 1 core processor static power consumption

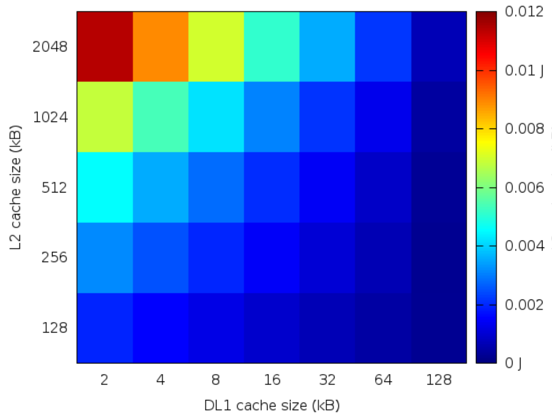


(A) Fixed associativity and latency

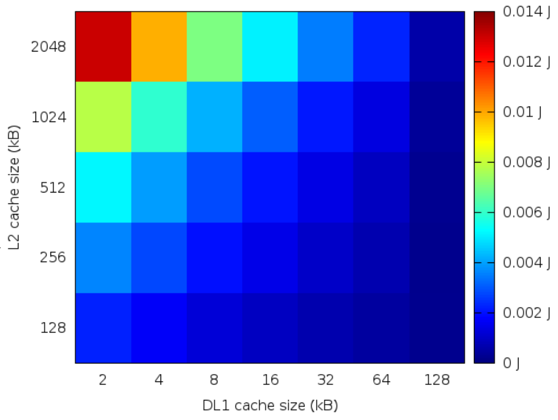


(B) Unfixed associativity and latency

FIGURE E.39: Dedup 1 core DL1 dynamic energy consumption



(A) Fixed associativity and latency

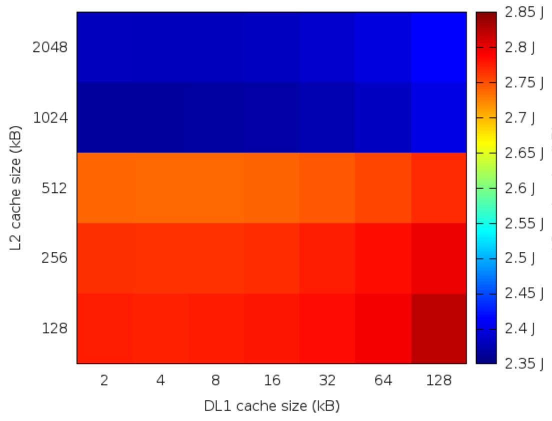


(B) Unfixed associativity and latency

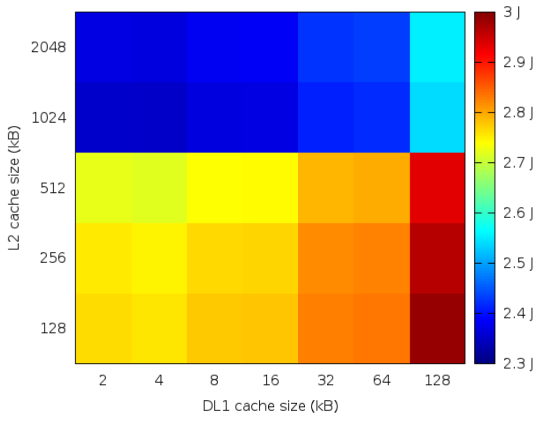
FIGURE E.40: Dedup 1 core L2 dynamic energy consumption

E.5 Streamcluster

E.5.1 1 core

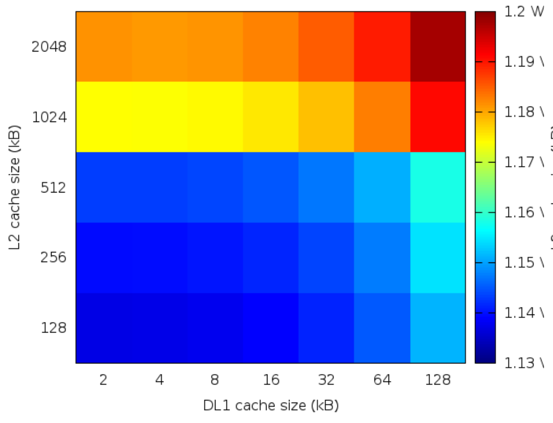


(A) Fixed associativity and latency

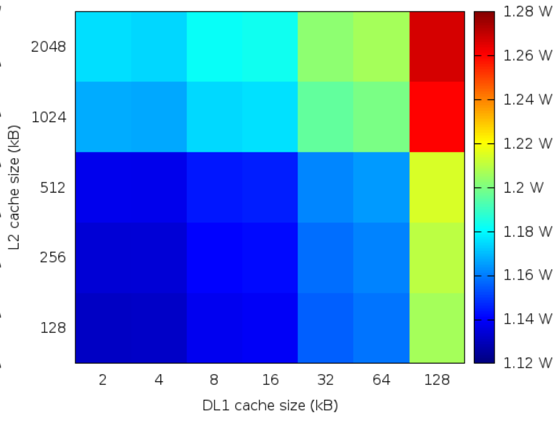


(B) Unfixed associativity and latency

FIGURE E.41: Streamcluster 1 core processor dynamic energy consumption

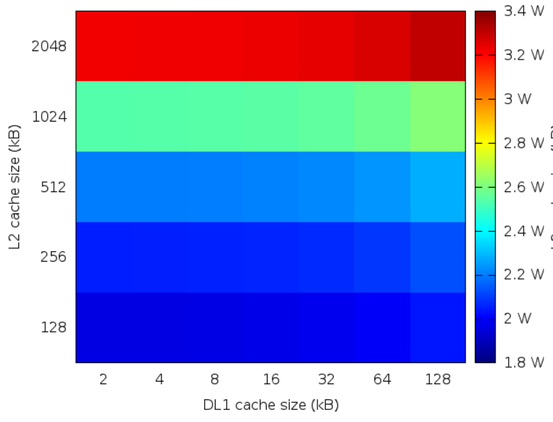


(A) Fixed associativity and latency

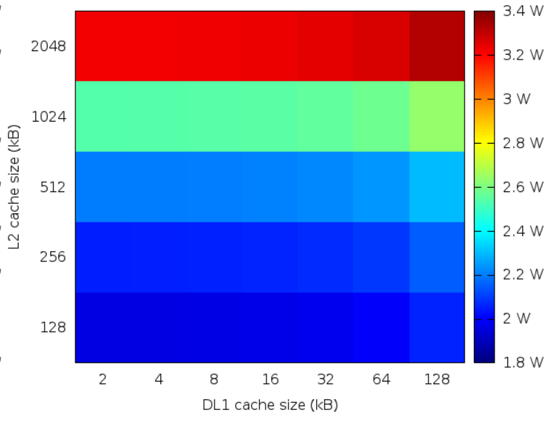


(B) Unfixed associativity and latency

FIGURE E.42: Streamcluster 1 core processor dynamic power consumption

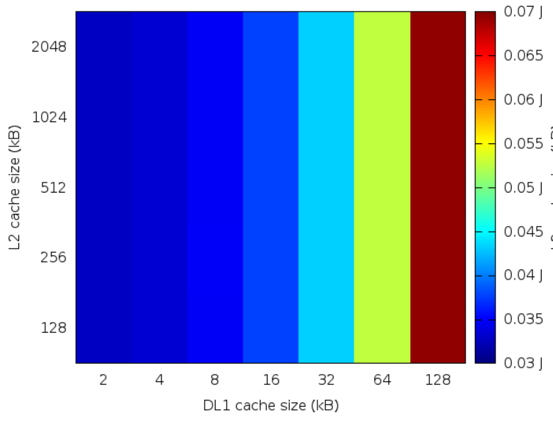


(A) Fixed associativity and latency

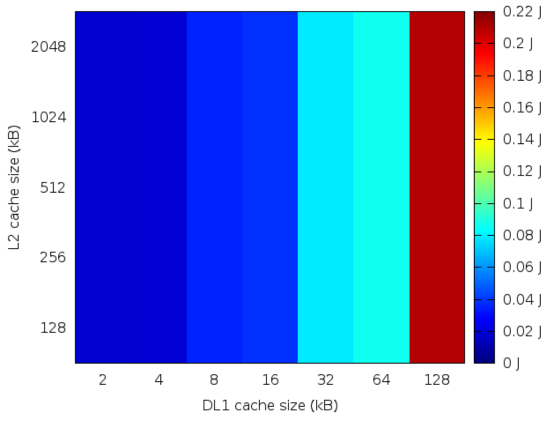


(B) Unfixed associativity and latency

FIGURE E.43: Streamcluster 1 core processor static power consumption

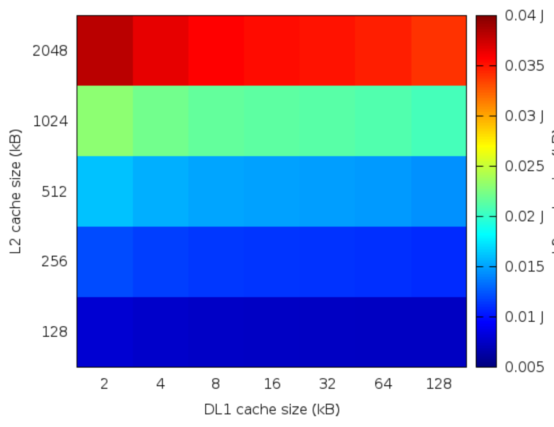


(A) Fixed associativity and latency

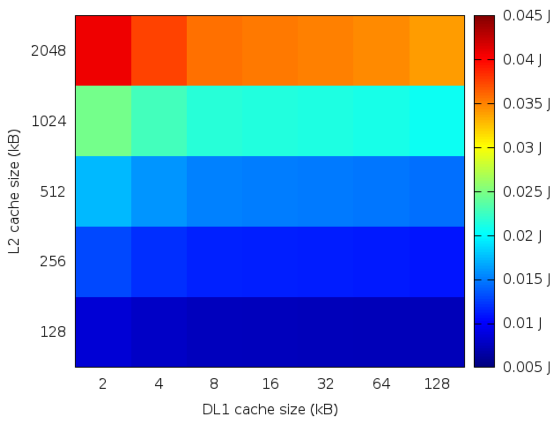


(B) Unfixed associativity and latency

FIGURE E.44: Streamcluster 1 core DL1 dynamic energy consumption



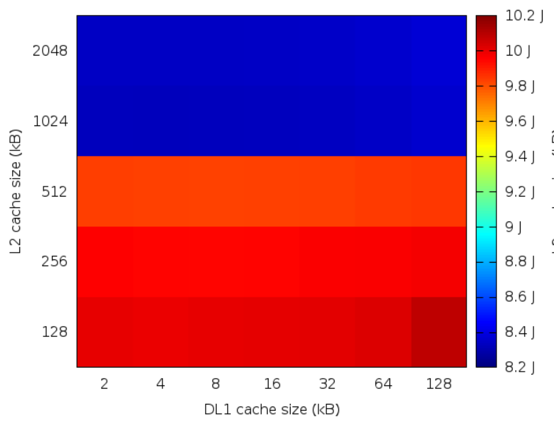
(A) Fixed associativity and latency



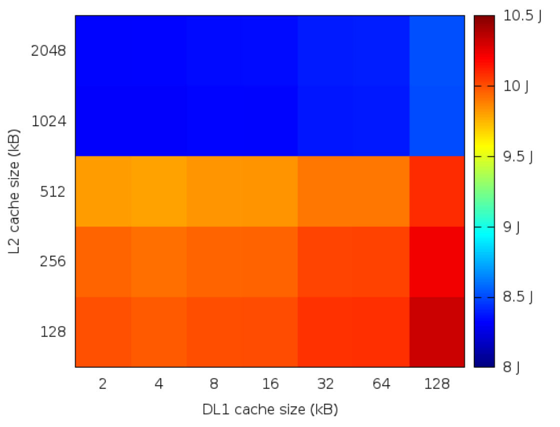
(B) Unfixed associativity and latency

FIGURE E.45: Streamcluster 1 core L2 dynamic energy consumption

E.5.2 4 core

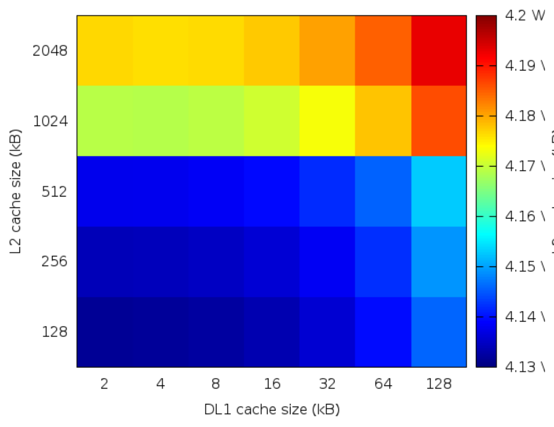


(A) Fixed associativity and latency

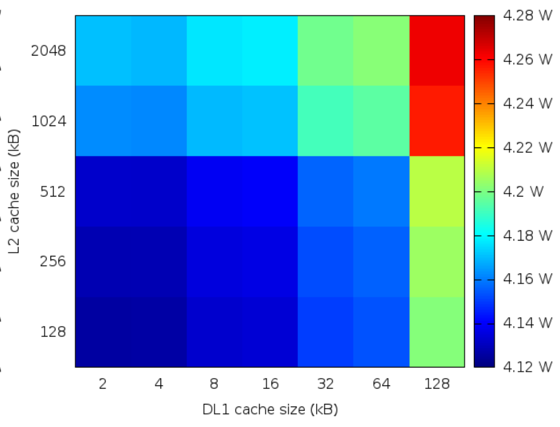


(B) Unfixed associativity and latency

FIGURE E.46: Streamcluster 4 core processor dynamic energy consumption

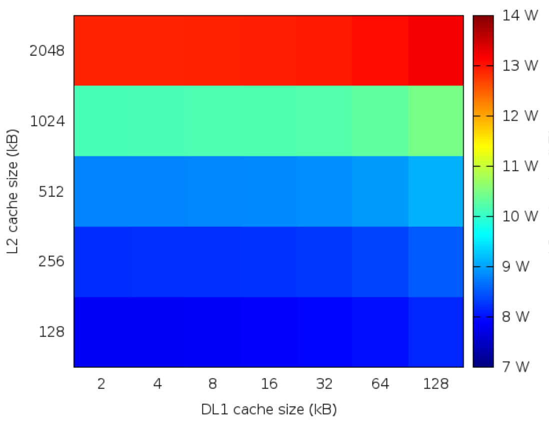


(A) Fixed associativity and latency

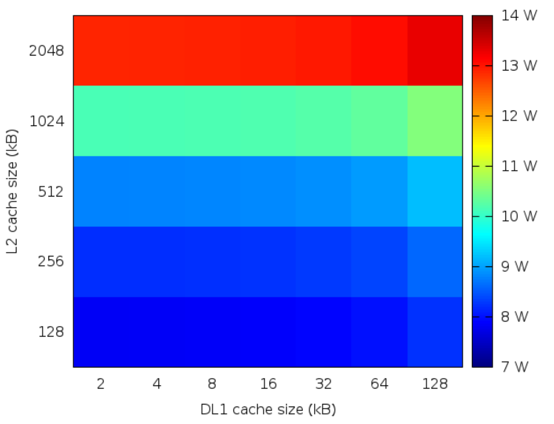


(B) Unfixed associativity and latency

FIGURE E.47: Streamcluster 4 core processor dynamic power consumption

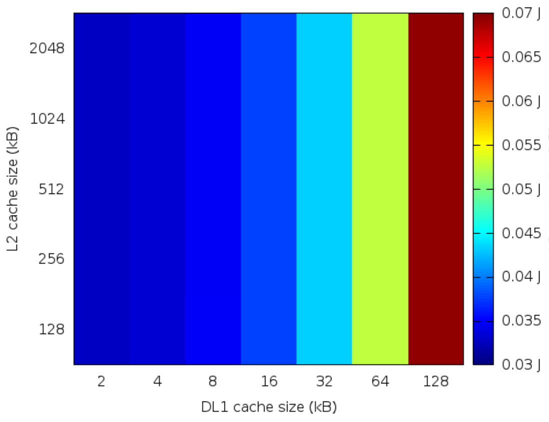


(A) Fixed associativity and latency

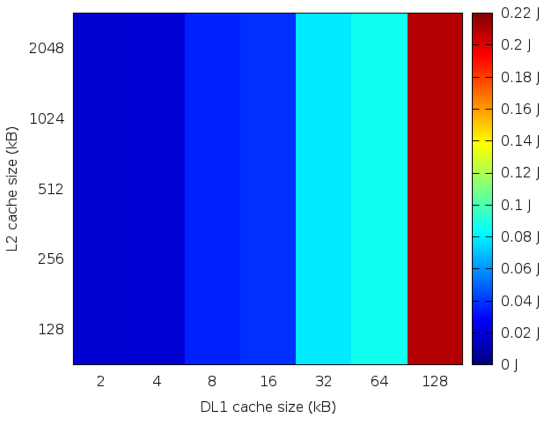


(B) Unfixed associativity and latency

FIGURE E.48: Streamcluster 4 core processor static power consumption

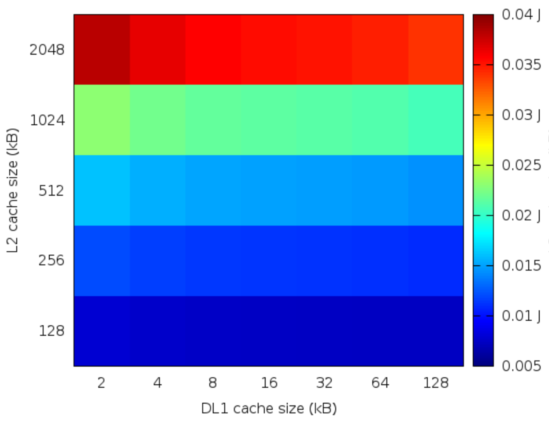


(A) Fixed associativity and latency

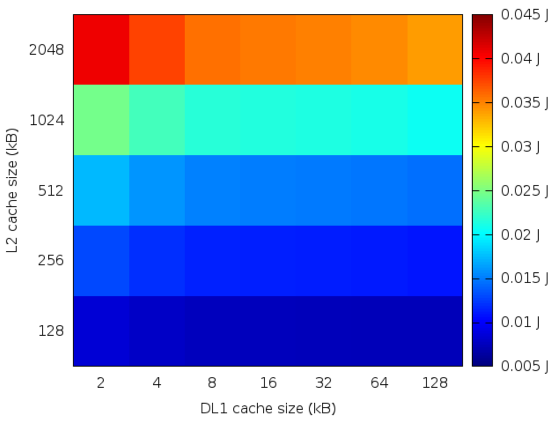


(B) Unfixed associativity and latency

FIGURE E.49: Streamcluster 4 core DL1 dynamic energy consumption



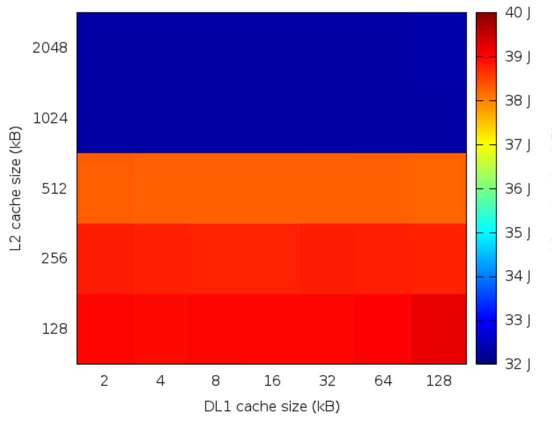
(A) Fixed associativity and latency



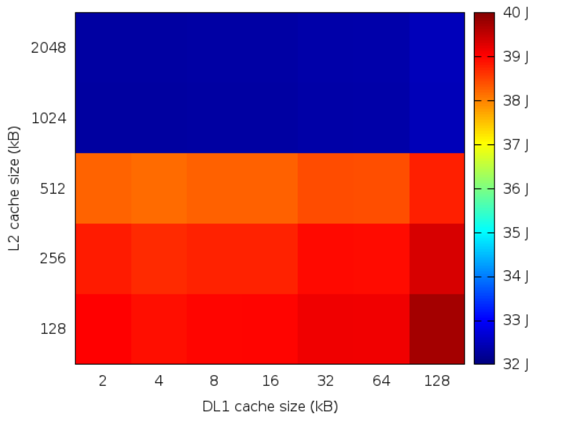
(B) Unfixed associativity and latency

FIGURE E.50: Streamcluster 4 core L2 dynamic energy consumption

E.5.3 16 core

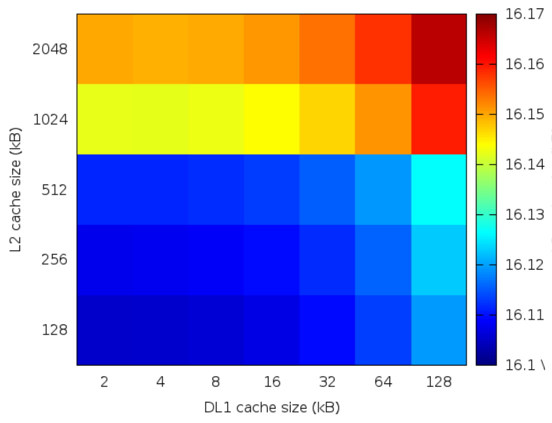


(A) Fixed associativity and latency

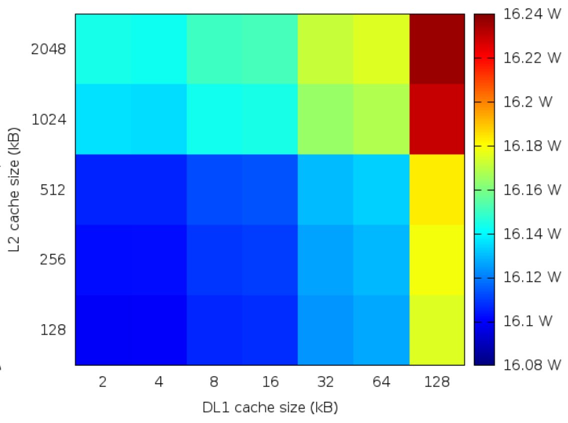


(B) Unfixed associativity and latency

FIGURE E.51: Streamcluster 16 core processor dynamic energy consumption

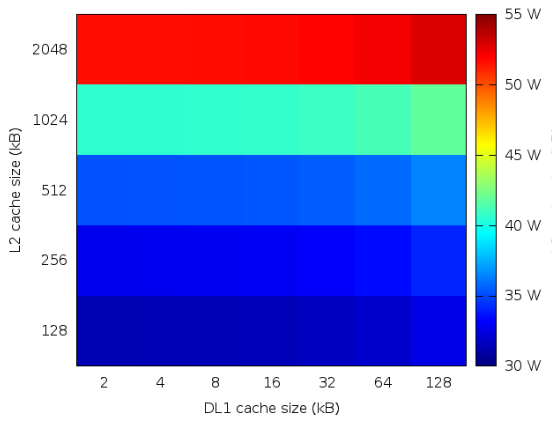


(A) Fixed associativity and latency

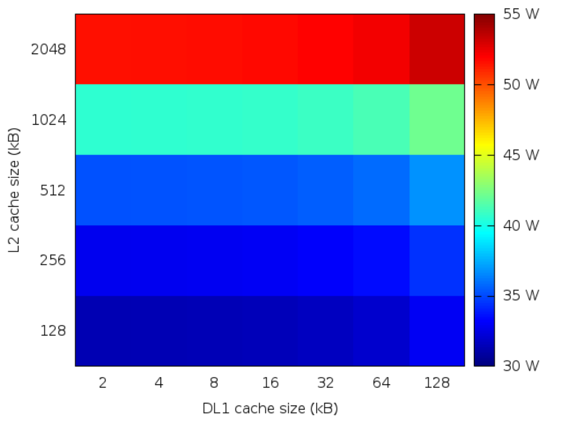


(B) Unfixed associativity and latency

FIGURE E.52: Streamcluster 16 core processor dynamic power consumption

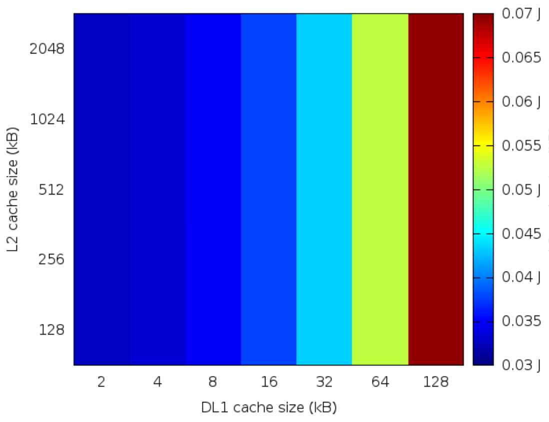


(A) Fixed associativity and latency

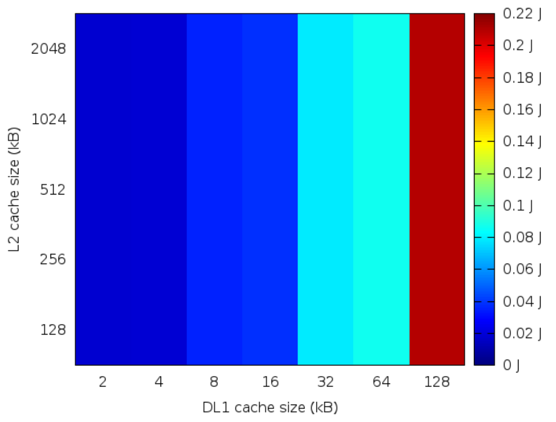


(B) Unfixed associativity and latency

FIGURE E.53: Streamcluster 16 core processor static power consumption

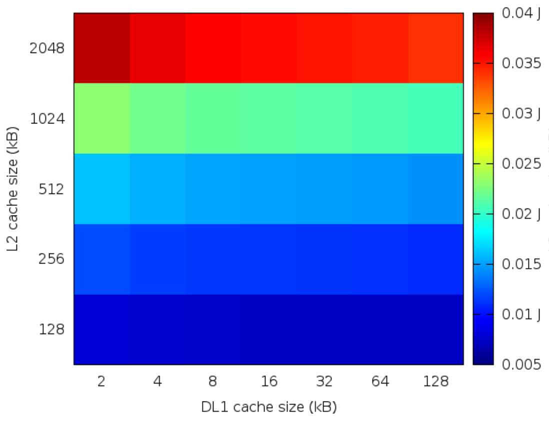


(A) Fixed associativity and latency

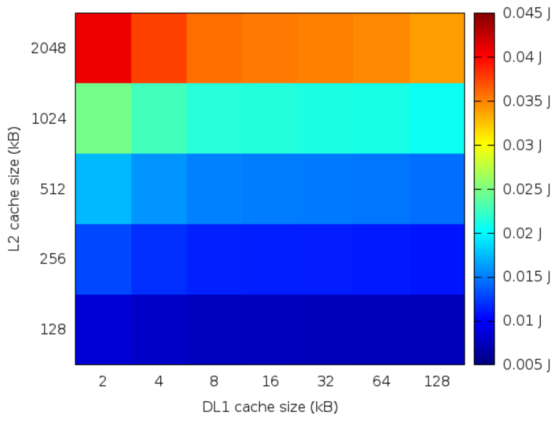


(B) Unfixed associativity and latency

FIGURE E.54: Streamcluster 16 core DL1 dynamic energy consumption



(A) Fixed associativity and latency

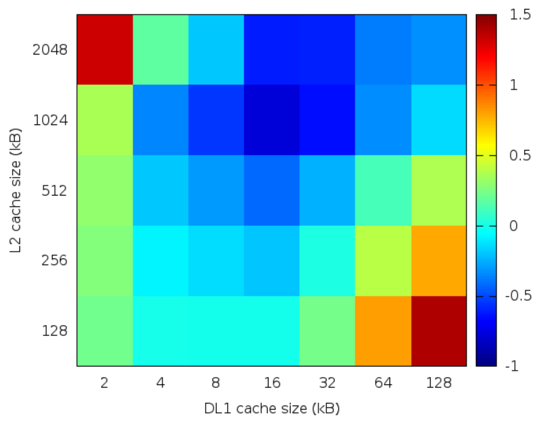


(B) Unfixed associativity and latency

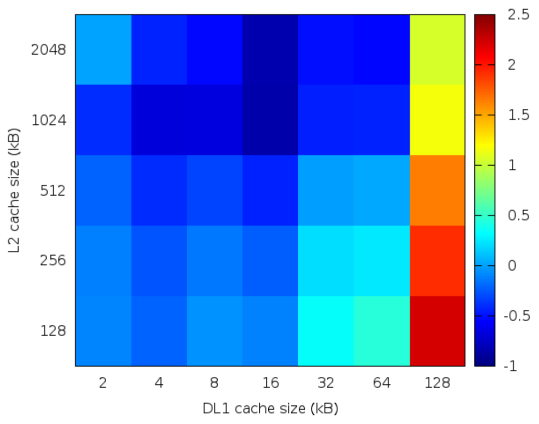
FIGURE E.55: Streamcluster 16 core L2 dynamic energy consumption

E.6 Average

E.6.1 1 core

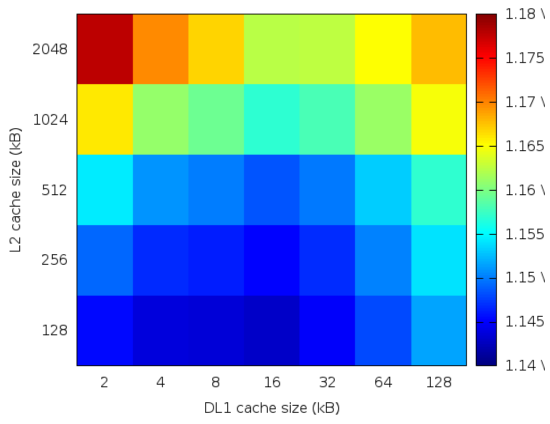


(A) Fixed associativity and latency

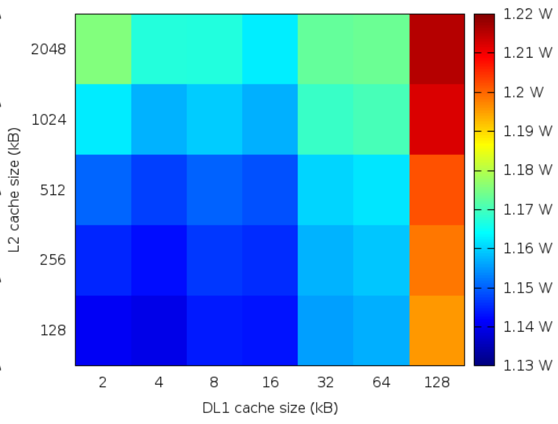


(B) Unfixed associativity and latency

FIGURE E.56: Average 1 core processor dynamic energy consumption



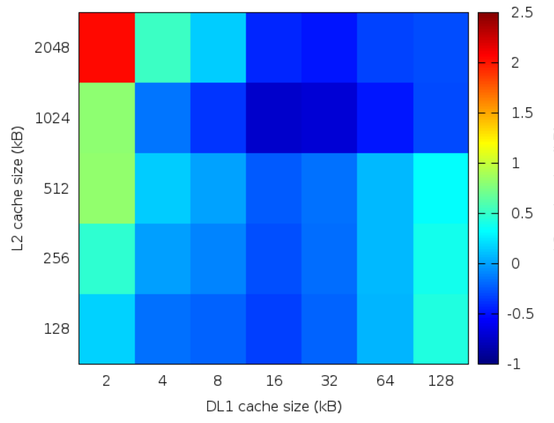
(A) Fixed associativity and latency



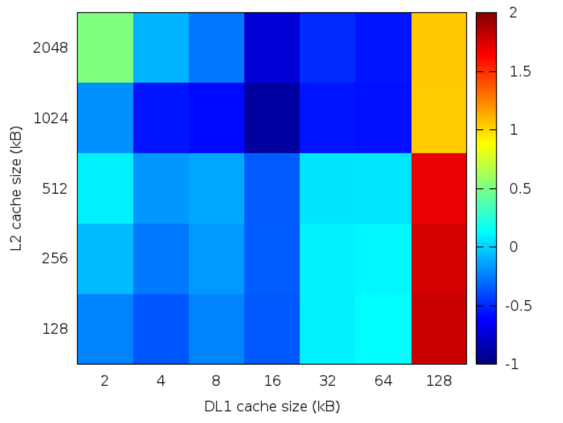
(B) Unfixed associativity and latency

FIGURE E.57: Average 1 core processor dynamic power consumption

E.6.2 4 core

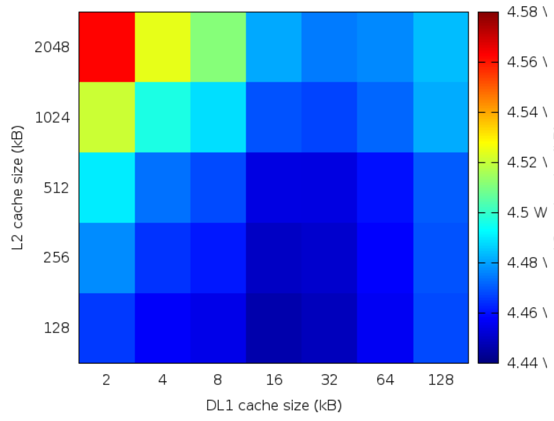


(A) Fixed associativity and latency

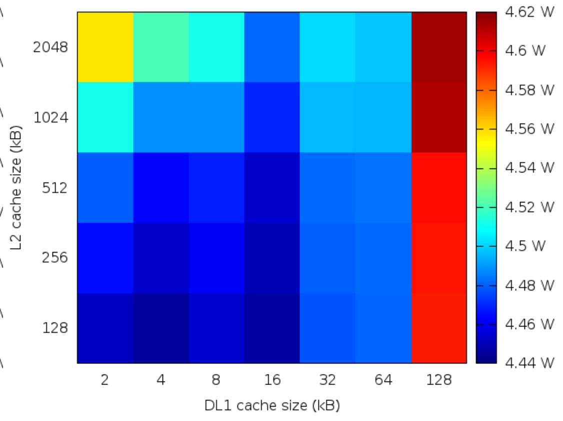


(B) Unfixed associativity and latency

FIGURE E.58: Average 4 core processor dynamic energy consumption



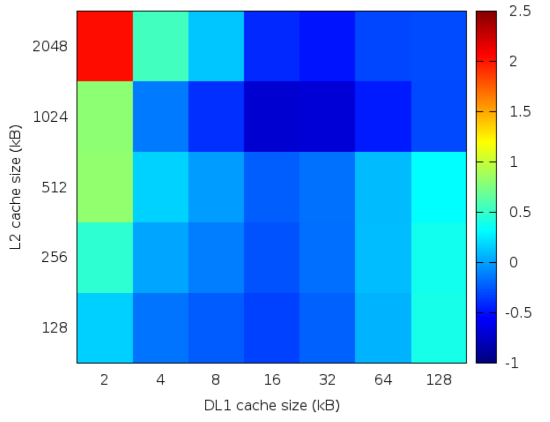
(A) Fixed associativity and latency



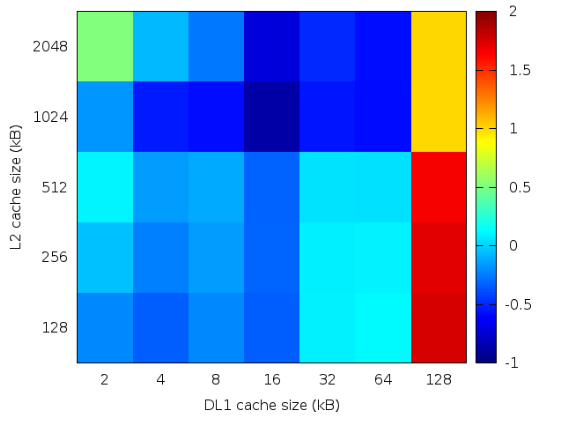
(B) Unfixed associativity and latency

FIGURE E.59: Average 4 core processor dynamic power consumption

E.6.3 16 core

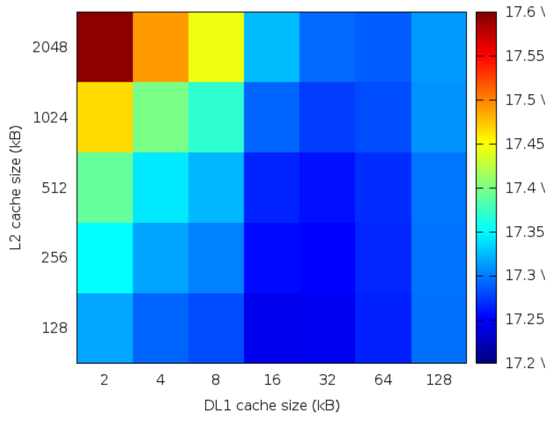


(A) Fixed associativity and latency

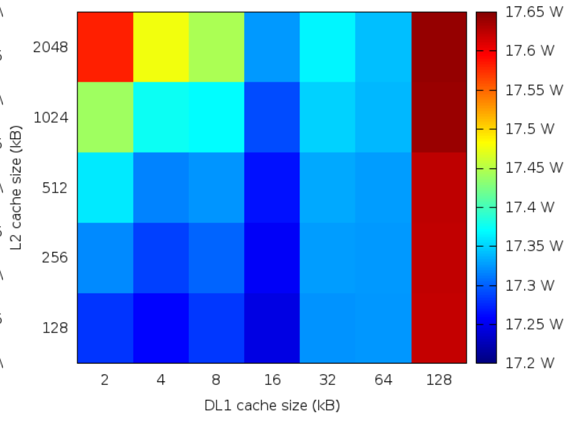


(B) Unfixed associativity and latency

Figure E.60: Average 16 core processor dynamic energy consumption



(A) Fixed associativity and latency



(B) Unfixed associativity and latency

Figure E.61: Average 16 core processor dynamic power consumption

Bibliography

[1] C. Bienia. Benchmarking modern multiprocessors. January 2011.

[2] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood. The gem5 simulator. *SIGARCH Comput.Archit.News*, 39(2):1–7, aug 2011.

[3] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt. The m5 simulator: Modeling networked systems. *Micro, IEEE*, 26(4):52–60, 2006.

[4] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In *Computer Architecture, 2000. Proceedings of the 27th International Symposium on*, pages 83–94, 2000.

[5] BSC. Extrae. http://www.bsc.es/computer-sciences/extrae. Visited on 03/18/2015.

[6] BSC. Mercurium. https://pm.bsc.es/projects/mcxx. Visited on 03/18/2015.

[7] BSC. Nanos. https://pm.bsc.es/projects/nanox. Visited on 03/18/2015.

[8] BSC. Paraver: a flexible performance analysis tool. http://www.bsc.es/computer-sciences/performance-tools/paraver/general-overview. Visited on 03/18/2015.

[9] T. E. Carlson, W. Heirman, and L. Eeckhout. Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In *High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for*, pages 1–12, 2011.

[10] L. J. Colmenar JM, Risco Martin JL. an overview of computer architecture and system simulation. Technical report, 2011.

[11] A. Duran, E. Ayguadé, R. M. Badia, J. Labarta, L. Martinell, X. Martorell, and J. Planas. Ompss: A proposal for programming heterogeneous multi-core architectures. *Parallel Processing Letters*, 21(02):173–193, 2011.

[12] F. A. Endo, C. Damien, and H.-P. Charles. Micro-architectural simulation of embedded core heterogeneity with gem5 and mcpat. In *Proceedings of the 2015 Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools*, RAPIDO '15, pages 7:1–7:6, 2015.

[13] D. Genbrugge, S. Eyerman, and L. Eeckhout. Interval simulation: Raising the level of abstraction in architectural simulation. In *High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on*, pages 1–12, 2010.

[14] S. Li, 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 *Proceedings of the 42Nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, pages 469–480, 2009.

[15] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi. The mcpat framework for multicore and manycore architectures: Simultaneously modeling power, area, and timing. *ACM Trans.Archit.Code Optim.*, 10(1):5:1–5:29, apr 2013.

[16] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood. Multifacet's general execution-driven multiprocessor simulator (gems) toolset. *SIGARCH Comput.Archit.News*, 33(4):92–99, nov 2005.

[17] 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 *High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on*, pages 1–12, 2010.

[18] A. Rico, F. Cabarcas, C. Villavieja, M. Pavlovic, A. Vega, Y. Etsion, A. Ramirez, and M. Valero. On the simulation of large-scale architectures using multiple application abstraction levels. *ACM Trans.Archit.Code Optim.*, 8(4):36:1–36:20, jan 2012.

[19] A. Rico, A. Duran, F. Cabarcas, Y. Etsion, A. Ramirez, and M. Valero. Trace-driven simulation of multithreaded applications. In *Performance Analysis of Systems and Software (ISPASS), 2011 IEEE International Symposium on*, pages 87–96, 2011.

[20] C. Sun, C. H. O. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L.-S. Peh, and V. Stojanovic. Dsent - a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling. In *Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on*, pages 201–210, 2012.

[21] S. J. E. Wilton and N. P. Jouppi. Cacti: an enhanced cache access and cycle time model. *Solid-State Circuits, IEEE Journal of*, 31(5):677–688, 1996.