ECE-611
Chein-Hung Su
G#00998786

Project 3th Dynamically Changing Cache Associativity and Size in L1/L2

Introduction

In this project, the main goal is to analyze the impact that different cache associativity and size affect the efficiency or target function (IPC/POWER, IPC^2/POWER), of processor with different applications. The reason to analyze the impact between cache changing and power issue is to search the most efficient cache configuration so that with the most efficient power we can provide better performance in processor. Therefore, in the following sections, I will introduce the methodology, implementation, results and conclusion.

Methodology

In this project, there are some materials and equipment used in the research: SMTSIM code, ubuntu system, python software, website for cache power and excel table for analysis. Generally, this methodology has three parts.
First, for SMTSIM code, this project supposes to generate the every 10k instructions and specific cycle time in order to calculate the IPC value. From the given code, I have to find out the code logic then report the number of instructions and cycle time. Therefore, I can find out the most efficient IPC in 1000 samples.
Second, for python software, this project needs four important scripts: extract number of instructions and cycle time, calculate the IPC value from number of instructions and cycle time, retrieval cache power from website and generate the excel table for analysis. Specifically, in the script of calculation for IPC value, the main concept of IPC value is "instruction per cycle". Therefore, in that scripts the most vital thing is calculating "# instructions/ cycle time".
Finally, excel table is to collect the IPC value to extract the most efficient one. Besides, collecting the cache power is critical. Based on IPC value and cache power, it easily calculates two kind of target functions. Therefore, according to target functions, I could generate graphs for analysis.

Implementation

The implementation can be grouped many divisions. First, I find out the most important parameter "states.instrs" which is in the queue.c. The way to find out the number of instructions is according to key word "IPC" in the final state. After that I go to the run.c to print the number of instruction and cycle time for IPC with some conditions in order to generate every 10k instructions and specific cycle time. Second, one core python script needs to modify because this project requires: L1 cache size with 32KB and 64KB, L1 cache associativity with 1-way and 2-way, L2 cache size with 128KB and

256KB, L2 cache associativity with 4-way and 8-way, L3 cache size with 4MB and L3 cache associativity with 16-way. Accordingly, each benchmark has to generate 16 file based on these configurations. Third, I write the python scripts to capture the number of instructions and cycle time in statistic period and calculate IPC value for each 10k instructions. The key point to capture data is to search the word “cyc” which I printed in the results. Therefore, with the number of instructions and cycle time, I could calculate the IPC value in specific given time. Fourth, another important script is to automatically extract the dynamic cache power from website “http://quid.hpl.hp.-com:9081/cacti/index.y?new”. In this work, the key approach is to use important library, mechanize. This library provides easy way to search key word and give value for that word in html code such as “br["cache_size"]”. Thus, the response results can simply be store in the file and grasp the specific cache power. Finally, I combine the IPC table and cache power table for generating two graphs in each benchmark. These benchmarks are applu, bzip and bwaves and results are presenting as following sections.

Result

There are three benchmarks in this section. First, target function(IPC^2/POWER) for applu in figure1 presents that changing size and associativity reduce the performance because although the IPC value increases based on changing size and associativity, the IPC value still has less effect than power consumption. In other words, we can see figure 1. There are four similar repeating bars for example, 1-4, 5-8. These two present only changed L1 size and associativity. So, it is easy to know that changing size and associativity for L2 does really reduce the performance but not too much. On the other hand, the rest groups of bars are changing L2 size and associativity. These two do decrease more performance than changing L1 configurations. In the figure 2, it is different target function to present performance. This performance is similar to figure but the key different is to presents different weighted performance. Figure 2 presents more weighted on power compared with figure 1.

ECE-611
Chein-Hung Su
G#00998786

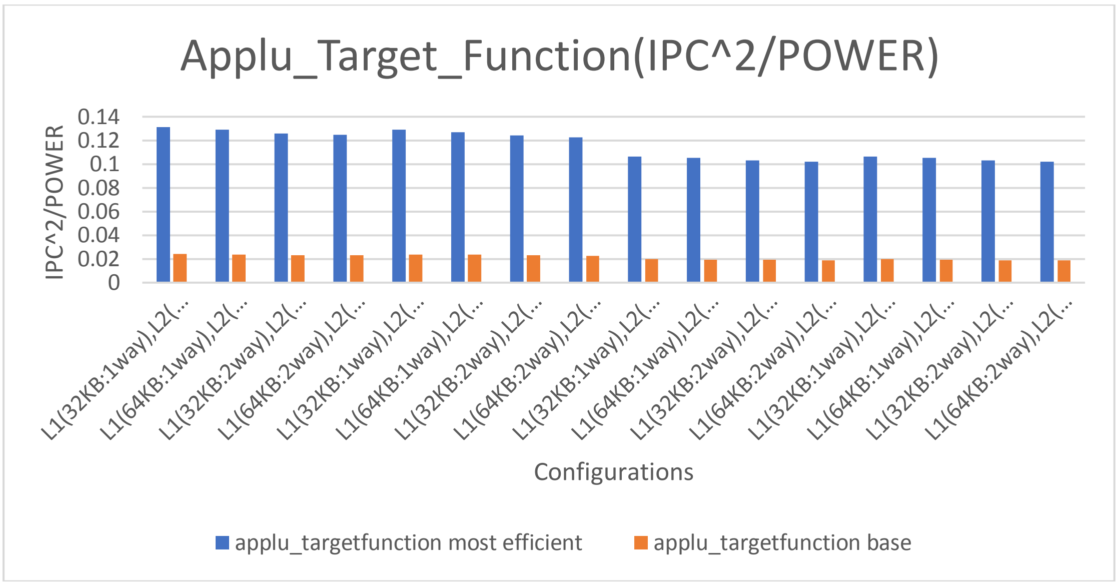

Figure 1 Applu Target Function(IPC^2/POWER)

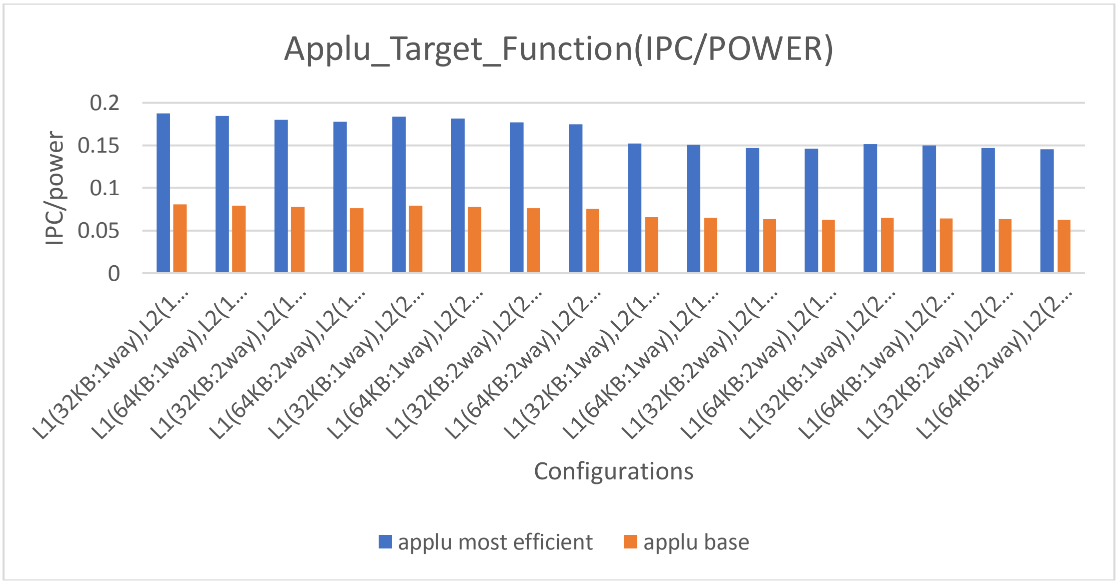

Figure 2 Applu Target Function(IPC/POWER)

Second, the target function(IPC^2/POWER) for Bzip in figure 3 presents that changing size and associativity reduce the performance because although the IPC value increases based on changing size and associativity, the IPC value has less impact than power consumption. In other words, we can see figure 3. There are four similar repeating bars for example, 1-4, 5-8. These two present only changed L1 size and associativity. So, it is easy to know that changing size and associativity for L2 does really reduce the performance but not too much. On the other hand, the rest groups of bars are changing L2 size and associativity. These two do decrease more performance than changing L1 configurations. In the figure 4, it is different target function to present performance. This performance is similar to figure 3 but the key different is to present more weighted on power.

ECE-611
Chein-Hung Su
G#00998786

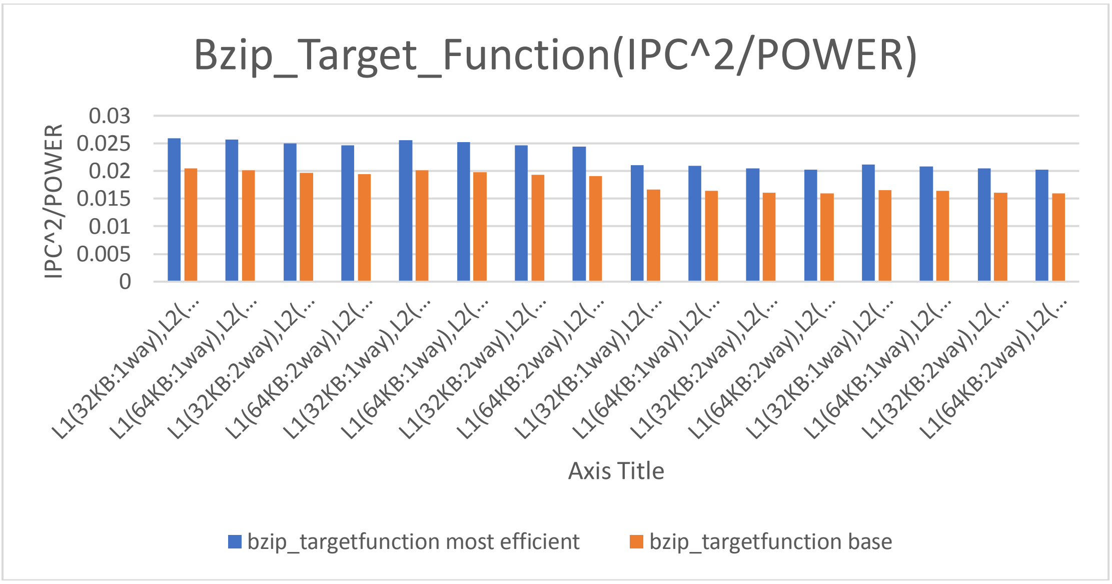

Figure 3 Bzip Target Function(IPC^2/POWER)

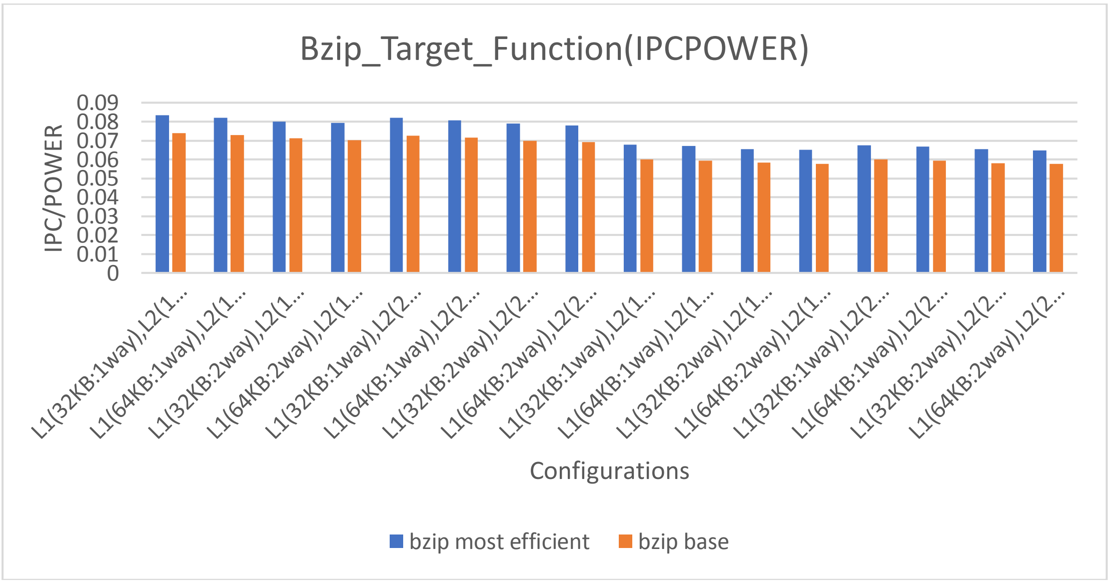

Figure 4 Bzip Target Function(IPC/POWER)

Third, the target function(IPC^2/POWER) for Bwaves in figure 5 presents that changing size and associativity reduce the performance because although the IPC value increases based on changing size and associativity, the IPC value has less influence than power consumption. In other words, we can see figure 5. There are four similar repeating bars for example, 1-4, 5-8. These two present only changed L1 size and associativity. So, it is easy to know that changing size and associativity for L2 does really reduce the performance but not too much. On the other hand, the rest groups of bars are changing L2 size and associativity. These two do decrease more performance than changing L1 configurations. In the figure 6, it is different target function to present performance. This performance is similar to figure 5 but the key different is to present more weighted on power.

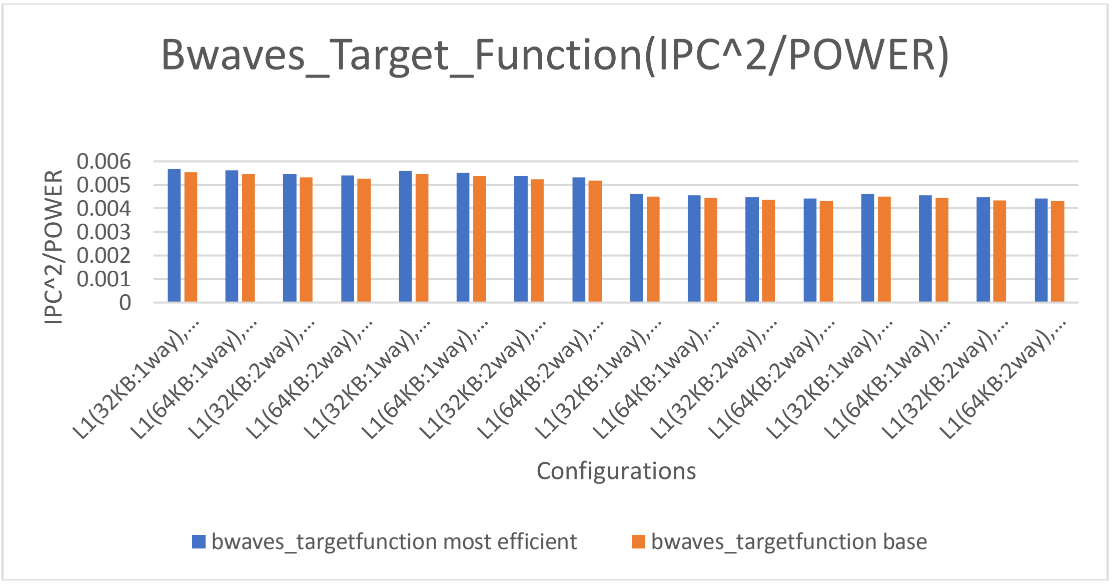

Figure 5 Bwaves Target Function(IPC^2/POWER)

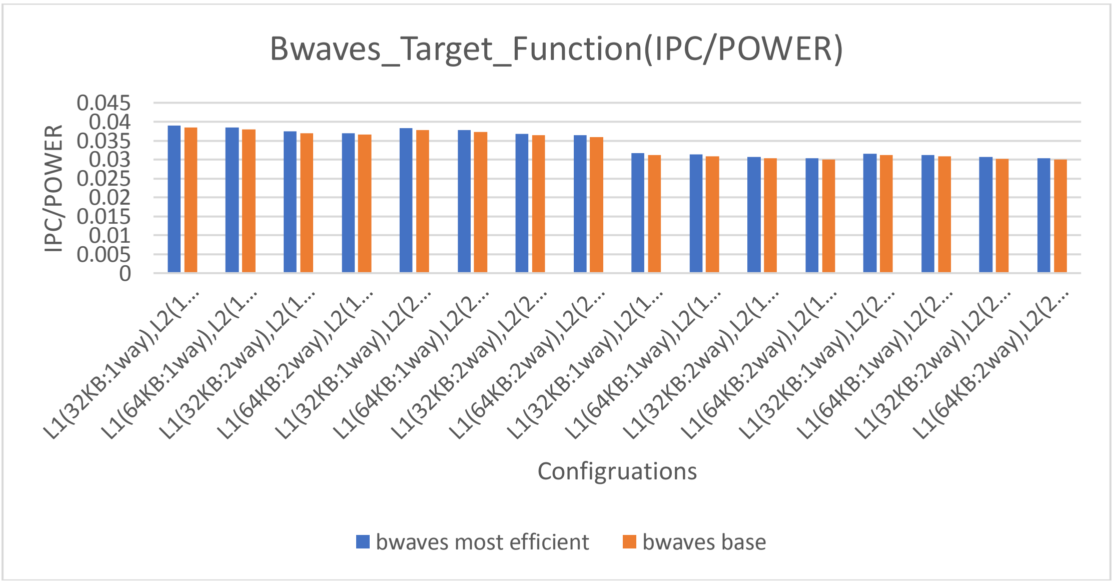

Figure 6 Bwaves Target Function(IPC/POWER)

Compared with these three benchmarks, based on the same cache power, the performance influenced by IPC value. In other words, if benchmark has higher IPC value, it has better performance. Therefore, from three benchmarks, applu is the best performance.

Conclusion

In this project, the simulation presents that larger cache size and associativity influence power consumption. Specifically, cache size influences more than associativity according to comparison of graphs. No matter increase cache size or associativity, it can increase the IPC but in the meanwhile changing cache size or associativity might increase the power consumption. There is a tradeoff between IPC and power consumption to get better performance. Therefore, with this analysis, it helps me to get the most efficient configuration.