DAT105 Computer Architecture
Study Period I, 2022
Department of Computer Science and Engineering
Chalmers University of Technology, Göteborg, Sweden

Laboratory Assignment 3: Power-Performance Trade-offs

Contributors: Mehrzad Nejat, Mohammad Waqar Azhar, Per Stenstrom

Background: In the previous lab assignments, you studied the effect of different architectural parameters of a processor with a simple memory subsystem on performance. You started with a simple in-order single-issue processor with a small single-level cache and improved the performance step-by-step. First, you improved the instruction and data cache, then you increased the branch prediction accuracy, and finally you exploited ILP by allowing out-of-order execution and multiple issue in a superscalar processor. You have managed to reduce the program execution time significantly by increasing the resources. But, it doesn't come for free. Adding resources increases the energy consumption of the processor.

Learning objective: In this assignment you will study the power/energy costs of adding resources. For this purpose, you are going to design a high performance processor using the Sniper simulator [1]. Then, you will perform power/energy measurements using McPAT [3]. There is usually an upper limit on the power consumption of a processor called the power budget. If the power goes beyond this limit, the cooling system fails to keep the chip temperature within a safe bound. So, in the next step you will review your design choices to reduce the chip power consumption below the limits. Furthermore, higher energy consumption means lower battery lifetime for portable devices and higher electricity cost for high-performance computer systems. Thus, in the final step you will study the energy-performance trade-off in your processor model with the goal of improving energy efficiency.

At the end of this assignment, you should be able to:

- Measure and report the performance of a single-core processor model with the Sniper simulator.
- Measure and report power/energy consumption of different components of a processor and a memory hierarchy using McPAT
- Understand how design choices affect the power consumption in order to meet the power limits
- Understand energy efficiency and the trade-off between energy consumption and performance

References:

[1] <http://snipersim.org/w/The_Sniper_Multi-Core_Simulator>

[2] <https://groups.google.com/forum/#!forum/snipersim>

[3] <http://www.hpl.hp.com/research/mcpat/>

Evaluation: You must write a report that includes: A brief description of the problem and the method you used, important assumptions you have made, simulation results and observations, your design choices and the reason behind them and limitations of the methodology. Detailed report guidelines will be available on Canvas.

Simulation Environment

In this assignment, you are going to use a new simulation environment. Since you are already familiar with Simple-Scalar, it should be easy for you to start working with Sniper + McPAT. You only need to use the basic Sniper simulator with "--power" option that will generate power/energy results using McPAT.

There are more advanced capabilities such as simulator APIs[1] that enable you to interact with the simulator from the program code under simulation or several tools already available in 'tools' and 'scripts' directories. But, you are not required to use these tools here. If you are interested, you can study the references for more details.

The basic simulation environment is depicted in Figure 1. If you look carefully at this figure, you will notice that there are several configuration files. That is because you can pass several configurations each containing a subset of the full configuration to Sniper using "-c" in the command line[2]. It is also possible to set a single parameter from the command line using "-g". For further details, check the simulator manual[3]. The simulator will generate a configuration file named "sim.cfg" in the simulation folder that contains the final settings of all the parameters.

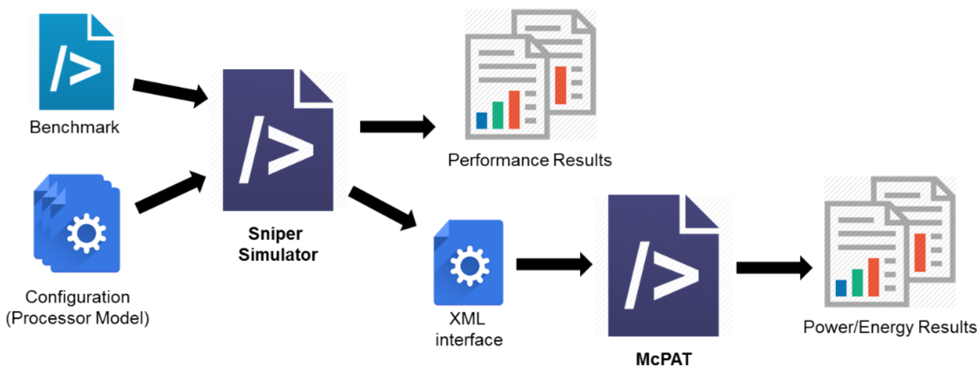

Figure 1: Simulation Environment.

Similar to the previous simulations, there is a script that will run the simulator and pass the required configurations and settings. The script is named "runsim.sh" and it could be found under the "Lab3" directory. You must enter the simulation home directory and the name of the configuration file (without ".cfg") as the first and second command line arguments when executing this script. You can deactivate the

[1] Application Programming Interface

[2] If you provide multiple configurations for the same parameter, the later one will overwrite the previous ones.

[3] Manuals are available in Canvas files. You can easily find them online as well.

simulation of each benchmark application by adding a "#" sign at the beginning of the corresponding lines in the script. You must be able to understand how this simple script works. If you have any problem understanding the parameters ask the teaching assistants and/or check the simulator manual.

Simulation Results: The performance results could be found in a file named "sim.out" while the power results are available in "power.txt" and a summary of Power/Energy measurements are printed on the standard output which is redirected to a text file[1]. The energy breakdown is also plotted in "power.png". But, you will mostly use "sim.out" and "power.txt".

Task 1: Design a high performance processor in Sniper

You have learnt how different architectural parameters of a processor and a cache impacts on the performance in the first two lab assignments. In this assignment you will use Sniper. The available configuration parameters are slightly different. Therefore, you cannot directly use the values from previous lab assignments. You will be provided a new base configuration file for Sniper named "base4.cfg". One major difference is that this base configuration uses a two-level cache hierarchy where the second level is a combined instruction and data cache reflecting what is typically used on the market.

1.1. Simulate the base configuration

Task 1.1. Assess the performance of the base model.

Here, the goal is to establish the performance of the base model in terms of execution time and relevant statistics to understand what limits its performance (Hint: CPI=1/IPC, Number of instructions executed and speedup relative to base). Collect performance results in a table such as Table 1 using the benchmarks listed there. **Devise a suitable methodology for this task.**

Create a directory for your simulation and place the configuration file there. Run the script by passing the directory address and configuration file name as two command line arguments. Once the simulation is done, you can find the results for each benchmark application inside the corresponding sub-directory. Open one "sim.cfg" file and check the full list of configuration settings. Notice how the values of the "gainestown" architecture[2] are overwritten by the base configuration. Also check data and tag access times to each cache. Open one "sim.out" file and check the available performance results.

[1] The gsm benchmark generates unreadable output on stdout which was redirected to a file that you never opened in the previous labs. But, now we also have Power/Energy summaries on stdout as well. So, don't be alarmed if you see unreadable characters when you open the file that contains the redirected stdout for gsm. Actually, if you don't see these characters, it means that something didn't work!

[2] Some basic configurations such as "gainestown" are provided with the simulator in a directory named "config"

If you compare the instruction counts with the previous lab assignments, you will notice some difference. The reason is that the benchmarks were compiled differently for SimpleScalar simulations. But the instruction counts should be relatively the same.

Table 1: Example report of performance results

	Application	dijkstra	qsort	stringsearch	gsm-untoast	jpeg-cjpeg	**GM***
base	Instructions						NA
	Time						NA
	CPI						NA
HPP	Time						NA
	CPI						NA
	SP**						
HPP+PW	Time						NA
	CPI						NA
	SP						

*Geometric Mean
** Speedup relative to base (use CPI values)

1.2. Design a high performance processor

Task 1.2:

Configure and assess the performance of a high-performance processor (HPP).

Here, the goal is to configure HPP to maximize the performance by establishing the execution time and use relevant statistics to understand what limit its performance (Hint: CPI=1/IPC, Number of instructions executed and speedup relative to base). **Devise a suitable methodology for this task.**

Using the experience from previous assignments, modify the processor parameters to maximize the performance in a new configuration named "HPP[1]". The configuration space is shown in Table 2. It differs from the configurations space for SimpleScalar in lab assignments 1 and 2. Note for example that you cannot change the branch prediction method. On the hand, it is possible to change the parameters of the second-level cache. Therefore, keep the level-1 cache parameters unchanged except for block size that should be the same for all the caches.

There are a few differences compared to previous assignments. In this case, you are not modifying the branch predictor. Some other parameters such as number of functional units are hard coded in the simulator and you are not modifying them either[2]. Table 1 shows a suggested way of reporting the performance of the base configuration and HPP.

However, you have a second-level cache that can further improve the memory sub-system. Therefore, keep the level-1 cache parameters unchanged with the exception

[1] High Performance Processor

[2] It is possible to modify the simulator since the source code is available. You can even find some suggestions and hints from the designers in the google group [2]

of the block size. First, the block size of all caches must be the same. Second, the number of sets (size / (associativity*block size)) must be greater than or equal to the block size. So, occasionally, you might need to increase the L1 size if you use bigger block sizes.

Hint: In order to reduce the simulation time, you can deactivate the simulation of the longest benchmark when doing some of the tests. Removing "--power" argument would also save you some simulation time, but make sure that you are not going to need the power measurement results for that simulation later. You can also run the simulations in parallel to speed up. One way to do that is to create one copy of the script for each benchmark application and deactivate the rest of the applications in that script. Then run each script in a separate terminal. Another method is to execute each simulation in background.

- How does a second level cache affect the performance? Is it the same as the first level caches?

Table 2: Configuration Parameters

Parameter	Possible values	HPP	HPP+PW	E-eff
in_order	True/False			
Outstanding loads and stores (LSQ)	1,8,16			
RS* entries	1,16,32			
Interval timer window size (ROB)	16,64,128			
Dispatch & Commit width	1,4,8			
L2 cache size (KB)	4,8,16,32			
L2 cache associativity	1,2,4,8			
block size** (B)	16,32,64			

* Reservation Stations
** block size should be the same for all the caches
Rule: Make sure that **for each cache** the number of sets (size / (associativity*block size)) is bigger than or equal to block size. So, you might need to increase the L1 size if you use bigger block sizes.

Task2: Power Measurements

Here, we want to consider the impact of processor resources on power/energy cost. The two important costs are power/energy consumption and area overhead. Power measurements were already activated in the script by using the "--power" argument. If you check a simulation output directory, you can find an image file that shows the energy break down. Detailed power values could be found in "power.txt" and "power.py" and a final power/energy summery was printed in the standard output and redirected to a text file. However, you are going to collect the power data from "power.txt". You can also find some estimations of area in this file. Notice the hierarchy of components. We are only interested in the core that contains the two levels of cache. So, only focus on the section related to the core. The total values could be found on top of the section followed by the detailed results for different components inside the core.

2.1. Compare the power/energy/area of base and HPP

Task 2.1: Assess the power/energy/area of base and HPP.

Here, we will use Sniper to compare base and HPP with respect to power, energy and area consumed. Devise a suitable methodology for this task by reading the rest of this section and carry out the task.

Look back at the simulation results for the base and HPP configurations. As a hint, you can collect the power and area measurement results for the core (total values) and use a table such as Table 3. You have noticed that there are 3 important power values[1]: peak-dynamic, runtime-dynamic, and leakage. Peak-dynamic is usually used when analyzing whether power is within the power budget and temperature bounds whereas runtime-dynamic is the average power consumption used for energy measurements. Leakage is another component of the power consumption that exists as long as the system is powered up, even if the system is idle with no operation. Power gating in sleep modes is one of the dynamic power management techniques used to reduce the leakage. But, we are not considering this in this assignment.

Use the following formula to calculate the energy:

$$Energy = (P_{runtime\ dynamic} + P_{leakage}) \times Time \qquad (1)$$

- How big are the power, energy, and area costs of the performance improvements obtained by HPP?
- Compare the share of different components in power, energy, and area costs.
- Is your observation the same for different benchmark applications? If not, what could be the reason?

2.2. Meet the power budget

In this step, we introduce a power budget of **11.5 W**. Now, you probably need to make some modifications to your HPP design to make sure the "core Peak power + leakage power" remains below this limit for all the applications[2]. Name your new configuration "HPP+PW" and fill Table 1, Table 2 and Table 3.

- How much performance did you lose when trying to meet the power budget?

[1] Ignore the "power gating" results

[2] Hint: If you manage to reduce the power for the most power hungry application, then other applications will probably meet the power wall limitation as well. So, you don't need to simulate all the applications for intermediate tests. But you need to confirm the final result for all of them.

Table 3: Power, Energy, and Area results

	Application	dijkstra	qsort	stringsearch	gsm-untoast	jpeg-cjpeg
base	Area					
	Peak Dynamic Power					
	Runtime Dynamic Power					
	Subthreshold Leakage Power					
	Energy					
HPP	Area					
	Peak Dynamic Power					
	Runtime Dynamic Power					
	Subthreshold Leakage Power					
	Energy					
	Area Increment* (%)					
	Energy Increment* (%)					
HPP+PW	Area					
	Peak Dynamic Power					
	Energy					

Task3: Energy Efficiency

You have noticed a trade-off between performance and energy consumption during the last task. In this task you will further study this trade-off by using the Energy-Delay-Product (EDP). EDP is one of the common metrics for measuring energy efficiency of a system. It combines both performance and energy improvement in a single metric. Use the following formula for calculating EDP:

$$EDP = Energy \times Execution\ Time \tag{2}$$

First calculate the EDP for the base and HPP configurations. Then, propose one design with lower EDP and call it "E-eff". Use your experience from the last two tasks. You could use a table such as Table 4 to summarize your results. You don't need to consider the power wall for this task.

Table 4: Energy Efficiency Results

	Configuration	dijkstra	qsort	stringsearch	gsm-untoast	jpeg-cjpeg
EDP	base					
	HPP					
	E-eff					

- Is your E-eff design closer to the base or HPP?