# EE382M – 15: Assignment 4

Professor: Lizy K. John
TA: Jee Ho Ryoo
Department of Electrical and Computer Engineering
University of Texas, Austin

Due: 11:59PM November 6th, 2014

## 1. Introduction

This assignment will give you some high level exposure to the simulator infrastructures and tools used by the larger computer architecture community. Specifically, we will be having you work with various tools to estimate power. More specifically, you will gain experience using McPAT (through the simulator GEM5). We will be using a benchmark suite designed specifically for embedded workloads called MiBench.

This is not a lab you will be able to complete the night before it is due. Working with simulators requires a great deal of patience, as runtime frequently eclipses several hours per workload. **Please start this lab as early as possible.** It will take you longer than you think. Write scripts to automate the tedious aspects of simulation runs and data reporting.

### 1.1 Background

#### 1.1.1 McPAT

McPAT is a multicore power simulator developed by HP Labs that is widely accepted and validated by the architecture community. It is capable of estimating static and dynamic power consumption for technology nodes from 90nm to 22nm. Essentially, it is implemented as a wrapper around the CACTI [1] tool developed by Norm Jouppi. CACTI provides detailed circuit characteristics for various memory structures and types. McPAT [2] uses these circuit models, a description of the target machine, and a description of the workload to estimate power at a functional unit level.

#### 1.1.2 Gem5

Gem5 [3] is a full-system simulation infrastructure that attempts to merge the best aspects of the M5 [4] and GEMS [5] simulators. Its main selling point is its pervasive object oriented nature that allows for easy modification and addition of new components. It has 2 basic modes of operation: SE and FS modes. FS (full-system) mode simulates all aspects of program execution, including the system calls to the operating system. SE (syscall emulation) mode has gem5 emulate the operating system calls. For our lab, we will be using SE mode exclusively.

#### 1.1.3 MiBench

MiBench [6] is an older embedded workload suite developed as an open source alternative for the costly EEMBC suite develop at the University of Michigan. It comprises a number of kernels and workloads from a variety of different embedded application domains, such as automobiles and network routers. We are using MiBench as opposed to more traditional workloads, such as SPEC CPU2006, solely because MiBench's short runtime will make your lives much easier.

**1.2 Setup and Requirements**

- You will need access to a Linux machine with root so you can install any needed libraries and dependencies.

- You will need to download and make McPAT. The easiest way is using the version control system Mercurial (alternatively you can download the code directly):

  ```
  hg clone https://code.google.com/p/mcpat/
  ```

  - Follow the directions in the README to install McPAT. You might need to get rid of the –m32 flag in mcpat.mk if you are on a 64 bit machine without the 32 bit development libraries installed.

- You will need to download and make gem5. Please use the stable version:

  ```
  http://repo.gem5.org/gem5-stable
  ```

  - Follow the directions in the README to compile gem5. Perform an optimized build in SE mode for x86. You will more than likely need to install several packages from your linux distributions package manager (apt-get, yum, etc).

**1.3 Deliverables**

The following deliverables must be completed for this assignment. You will submit a tarball that is named <lastname>-a4.tar.gz and has the following working directory structure:

```
/report.pdf            // results for 3 MiBench workloads,
                       // the warm-up questions, and contest discussions
/scripts/*
/contest/energy/energy_se.py          // related files for the energy contest
/contest/energy/stats.txt
/contest/energy/mcpat_out.txt
/contest/power/power_se.py            // related files for the power contest
/contest/power/stats.txt
/contest/power/mcpat_out.txt
/contest/performance/perf_se.py       // related files for the performance contest
/contest/performance/stats.txt
/contest/performance/mcpat_out.txt
```

Your report will be in PDF format and will contain:

- Problem solutions in Section 2
- Graphs and discussion in Section 3
- Discussion in Section 4

Any scripts used in the lab must be put in the /scripts directory. Additionally, you must submit the final values of the energy, power, and performance system configurations as reported by Gem5.

## 2. Warm-Up

We really are serious about reading the references, documents, papers for gem5 and McPAT. Answer the following questions using the literature:

1.) How was McPAT validated in the original paper? What were the validating processors?

2.) What processor models are supported in GEM5? What memory models are supported? How would I decide which models to use for a research project?

3.) What is a Domain Specific Language (DSL)? What types of DSLs does gem5 support?

4.) What ISAs does GEM5 currently support? On what ISA(s) can I run the InOrder processor model with the classic memory system in FS mode?

## 3. GEM5 + McPAT

### 3.1 Overview

This section will have you use GEM5 and McPAT together with MiBench programs to discover relationships between performance, power, and energy. You will be looking at three MiBench programs: susan, bitcnt, and qsort

### 3.2 Help, Assumptions, and Rules

In your lab packet you are given:

- Precompiled MiBench programs modified to work with gem5 at different optimization levels
- mcpat_template.xml file

You will be running each benchmark through gem5, and passing the gem5 statistics to McPAT to glean power. You will then analyze all of these results to determine the effect on performance, power, and energy.

**MiBench:**

We have provided you with three of the benchmarks from the automotive suite of MiBench that have been modified to work with gem5: susan, bitcnt, and qsort. We have provided a runme.sh script showing how to run the program outside of the simulator environment.

**Running gem5:**

We will be running gem5 in SE mode on x86. The easiest way to accomplish this is with the provided se.py script in the gem5 directory. Since we care about power, we will need a more detailed CPU model than the default gem5 simulator selects.

For this section of the lab, we want you to configure gem5 to run in system emulation mode with the following flags:

```
--cpu-type=detailed
--caches
--l2cache
```

These settings have gem5 run an OoO core with 2-levels of cache and default sizes. Everything else should be kept at their default values. We will explore the breadth of what gem5 can simulate in subsequent sections.

**Running McPAT:**
McPAT is not fully integrated into the gem5 simulator environment, so you will have to do a little legwork here. Essentially you will need to convert the output of a gem5 run (stats.txt) to a valid input file for McPAT. McPAT takes input files in the xml file format, and the input files are a combination of a system topology/configuration and runtime statistics.

One thing you will notice when working with McPAT is that it is not always clear how to map the output of a simulator to the input of McPAT. Therefore we have provided the **mcpat_template.xml** file dictating how to map the output of gem5 into the input of McPAT. Let us go over the example of the template layout, using the L2 cache as a reference. The following template illustrates how we have laid out the input file:

```
<component id="system.L20" name="L20
  <param name="L2_config"
       value="REPLACE{config.system.l2.size,config.system.l2.tags.block_size,
                     config.system.l2.assoc,1,1,config.system.l2.hit_latency,
                     config.system.l2.tags.block_size,1}"/>
 <param name="buffer_sizes"
       value=" REPLACE{config.system.l2.mshrs,config.system.l2.mshrs,
       config.system.l2.mshrs,config.system.l2.mshrs}"/>
<param name="vdd" value="0"/>
<param name="clockrate"
       value="REPLACE{1e-6/( config.system.clk_domain.clock * 1e-12)}"/>
<param name="ports" value="1,1,1"/>
<param name="device_type" value="0"/>
<stat name="read_accesses"
       value="REPLACE{stats.system.l2.ReadReq_accesses::total}"/>
<stat name="write_accesses"
       value="REPLACE{stats.system.l2.ReadExReq_accesses::total}"/>
<stat name="read_misses"
       value="REPLACE{stats.system.l2.ReadReq_misses::total}"/>
<stat name="write_misses"
       value="REPLACE{stats.system.l2.ReadExReq_misses::total}"/>
<stat name="conflicts"
       value="REPLACE{stats.system.l2.tags.replacements}"/>
<stat name="duty_cycle"
       value="1.0"/></component>
```

Every value that has a ***REPLACE*** in front of a {} needs to be replaced by the appropriate statistic from either the gem5 configuration file or the stats file. Everything else should be left at the default value.

For example, `REPLACE{stats.system.l2.ReadExReq_accesses::total}"/>` should be replaced by the value of `system.l2.ReadExReq_accesses::total` from the gem5 stats file for a given run. Please note that the {} can contain:

- a basic stat (read_accesses)
- a formula that needs to be evaluated from the basic stats (eg. clockrate)
- a comma separated list of basic stats (e.g. l2_config).

Once you have configured your input file for McPAT you can go ahead and run the tool using the directions in the readme. Please use the "optimize for clock" option, and ignore any warnings about not being able to meet the target clock rate.

**HINT**: You may want to write a script (python and perl suggested) to do the text replacement for the McPAT configuration file. You really don't want to be doing this be hand, as there are a lot of runs, a lot of variables, and a lot of places to mistype values.

## 3.3 Questions

1) Compile each of the 3 benchmarks at the O2 optimization level and run them through McPAT and gem5 as detailed above. Organize the McPAT output into 3 stacked bar graphs showing static and dynamic power consumption for each of the workloads. Each stacked bar graphs should have the following components:
    a. Total static power with power for the entire processor
        i. Please include sub threshold leakage and gate leakage. Use power gated values.
    b. Core dynamic power broken down into:
        i. Instruction Fetch Unit
        ii. Renaming Unit
        iii. Load/Store Unit
        iv. MMU
        v. Execution Unit
    c. L2 dynamic power
    d. MC power
    e. NoC power

2) Now go through and repeat the same experiment with O0, O1, O3, O4 and Os optimization levels. Present your results in the form of a clustered stacked bar graph (a description of what I mean be these graph formats is below). Also present the EDP, IPC and simulated runtime for each benchmark and compiler optimization flag in whatever format you feel best represents the data.

3) Some people have suggested that compiling for performance might, in fact, be the same as compiling for power (or energy to be more precise) [7]. Why might this be the case? What do your results indicate? Use your graphs and data to back up your statements.

   If someone is interested in minimizing energy per workload, what flags should they use? What about minimizing average power per workload?

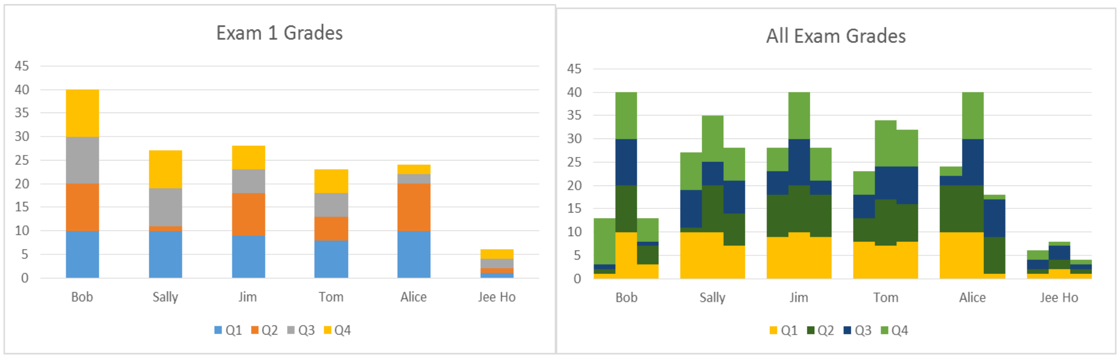

**Figure 1: Stacked Bar and Clustered Stacked Bar Examples**

## 4. Competition

### 4.1 Overview

For this section, you will be competing with your fellow students for the best microarchitecture for a given workload over a variety of definitions for "best". You will be using the gem5 + McPAT flow that you developed in the previous section.

### 4.2 Rules and Requirements

Please provide the best microarchitecture for the provided susan workload from the previous section, compiled at O0 optimization level. You will submit three different microarchitectures that optimize:

- Energy consumption
- Power consumption
- Performance (as measured in simulated runtime)

We are restricting the design space of your exploration to the following parameters, both to make your life easier, as well as ours for grading. You are allowed to change the following parameters listed in Table 1:

**Table 1: Configuration Parameters**

| Parameter | Minimum | Maximum |
|---|---|---|
| CPU Type | DerivO3CPU (detailed) | |
| CPU Clock | 500 MHz | 4Ghz |
| CPU Cores | 0 | 4 |
| L1D Size | 1KB | 64KB |

| | | |
|---|---|---|
| L1I Size | 1KB | 64KB |
| L2 Size | 1KB | 1MB |
| L1D associativity | 1 | 8 |
| L1i associativity | 1 | 8 |
| L2 associativity | 1 | 8 |
| Cache Line Size | 8 | 64 |
| Decode Width | 1 | 8 |
| Issue Width | 1 | 8 |
| Commit Width | 1 | 8 |
| ROB Entries | 32 | 192 |
| SQ Entries | 8 | 64 |
| LQ Entries | 8 | 64 |
| Physical INT Registers | 32 | 512 |
| Physical FP Registers | 32 | 512 |

**All other parameters must be kept at their default values.**

You will notice that not all these configuration parameters can be tweaked from the se.py command line. In fact, when we verify your results, we are only going to run the following:

```
./gem5.fast <your_script>.py –outdir=$ODIR --cpu-type=detailed --
caches --l2cache –c <workload_stuff>
```

This means that you will need to modify the default se.py file to produce the machine configuration that you are interested in simulating. Your script will be copied into the examples/configs folder in gem5 right where se.py currently resides, so you can rely on any of the files from the common library. Please note that every other config other than those listed above must be the same as the default values produced by se.py when run with the above command line!

Although you are allowed to change the above gem5 configurations, you ***must not change the MCPAT template file***. That is, you must use the same template file from the previous section in this section.

In your report we expect you to include the following:

- A discussion of what configurations you tried, as well as your intuition for trying them. You are allowed/encouraged to profile the binaries using pin and/or perf, and to look at the MiBench source on the web.
- A concise summary of the performance, power, and energy of your submitted machine configuration at each design point. Graphs and accompanying text are encouraged.

Additionally, for competition (power, performance and energy) you must provide the following:

- A modified se.py file that will run your optimal machine configuration.

- The gem5 stats text file
- The output of McPAT, generated at maximum detail level

You are allowed one submission for each power, performance and energy competition. The delivered files must in the directory structure specified in Section 1.3 ***Deliverables.***

**NOTE:** If you think there is a problem with how we map an architectural feature to McPAT, well, you are probably right! Research tools like these often don't interface with each other with 100% accuracy, and are imperfect approximations. If a mapping problem influences your decisions for this section, then simply make a note of it in the report and work around it. If it is so major that you cannot complete the lab, then please let the TA know so that it can be fixed.

**HINT:** The input space is too broad for you to try every combination and complete this lab on time. Use your intuition to sample reasonable portions of the design space. Iterate through a couple times, using the results of your previous runs to motivate new design space decisions.

### 4.3 Grading

The entire competition is worth a total of 50 pts. Out of these 50 pts, 10 will come from your reasoning and results reporting, and 40 will come from the quality of your results as compared to other students. Your assignment must be compressed in tar.gz format. Please use the following command to tar your submission:

```
% tar –zcvf <your last name>-a4.tar.gz <your working directory>
```

**Make sure that the name of your working directory is named <your last name>-a4.**

**An automated script will be used to grade this assignment. Thus, if the script fails to grade your code due to not following instructions, you will lose substantial part of your grade. If something is unclear, ask!**

There will be a submission link available on the course website, so please submit your tarball by the deadline. Check the following before you tar your directory.

1. Please remove all your temporary files.
2. The report must be in the pdf format.
3. All graphs and figures must be included in the report. Figures not included in the report will not be looked, and thus, will not be grade.

**We will be verifying that your reported stats and your reported configurations match. Don't cheat.**

## 5. References

[1] Shivakumar, P., & Jouppi, N. P. (2001). *Cacti 3.0: An integrated cache timing, power, and area model*. Technical Report 2001/2, Compaq Computer Corporation.

[2] Li, S., Ahn, J. H., Strong, R. D., Brockman, J. B., Tullsen, D. M., & Jouppi, N. P. (2009, December). McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on* (pp. 469-480). IEEE.

[3] Binkert, N., Beckmann, B., Black, G., Reinhardt, S. K., Saidi, A., Basu, A., ... & Wood, D. A. (2011). The gem5 simulator. *ACM SIGARCH Computer Architecture News*, *39*(2), 1-7.

[4] Binkert, N. L., Dreslinski, R. G., Hsu, L. R., Lim, K. T., Saidi, A. G., & Reinhardt, S. K. (2006). The M5 simulator: Modeling networked systems. *IEEE Micro*,*26*(4), 52-60.

[5] Martin, Milo MK, et al. "Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset." *ACM SIGARCH Computer Architecture News* 33.4 (2005): 92-99.

[6] Guthaus, M. R., Ringenberg, J. S., Ernst, D., Austin, T. M., Mudge, T., & Brown, R. B. (2001, December). MiBench: A free, commercially representative embedded benchmark suite. In *Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on* (pp. 3-14). IEEE.

[7] Valluri, M., & John, L. K. (2001). Is Compiling for Performance—Compiling for Power?. In *Interaction between Compilers and Computer Architectures* (pp. 101-115). Springer US.