CS 6810 Homework #4

Due: 9:10 a.m. Oct. 29 (no late submissions will be graded)

General instructions: Same as usual. This homework will require that you run some code for experiments but the code will be provided for you and so what you will hand in will be a written description of your experimental setup, the results generated by your experiments, and your conclusions.

Background: Caches are extremely important for single core processor performance and as we now move into the study of multi-core and multi-processor systems they will become even more important. In some sense making multi-whatever systems is getting simpler from the processor core perspective but in order for such systems to achieve parallel performance gains the design of the memory system and the interconnect are becoming more critical to system performance than things like pipleline organization and execution unit structure.

A lot of the academic research being done on both memory organizations and interconnect strategies has been based on 2 tools: **CACTI** & **ORION**. CACTI is a memory modeling tool that currently comes from HP Labs and ORION is an interconnect modeling tool that comes from Li Shuan Peh's group at Princeton. Both of these tools can be used stand-alone in small studies (like this homework) but are more typically coupled with a larger architectural simulation framework such as SimpleScalar, M5, or Simics since both the architecture and a variety of benchmark codes are necessary for a more complete picture of architectural merit.

CACTI is a tool that allows you to explore the performance, area, and power impacts of certain organizational choices. CACTI has been around for a long time and there have been many improvements. As you might expect in today's world where power (both active and leakage), the relatively weak scaling of wires with respect to transistors, and performance are all critically important. Last week, the latest version or CACTI 6.5 was released and you'll be using this tool for this homework. CACTI 6.5 has much more accurate wire models, power analysis, and provides a greater range of process technology options. It also allows caches to be built from standard SRAM style arrays as well as embedded DRAM which allows higher data density but suffers from the need to periodically refresh the memory array. It has also been enhanced to model main memories as well. The whole purpose of this assignment is to introduce you to his with this useful research tool as well as to quantify some of the more conceptual aspects of cache organization that you've hopefully been studying.

Caution: since everybody will be running the same experiments, there may be a strong temptation to use the data gathered by somebody else – needless to say **DON'T GIVE IN TO THIS TEMPTATION** – besides there's enough room for creativity that it will be easy to spot a non-original copy. As an incentive to do the right thing there will be a CACTI specific question on the next mid-term and if you

don't answer it properly then your grade for HW4 will become 0. Hopefully an even better motivation to do the right thing is this is a chance to become familiar with an important research tool. In fact this assignment asks you to do some research to answer the questions and hence you'll need to figure out how to use CACTI to quantify your answers. It would be a simple robot like chore to just run CACTI with pre-specified parameters and then ask you to cut and paste the results. At the graduate level this is silly so you won't be asked to do that.

Setup – the CACTI tool has been installed on all of the CADE machines and is linked to /usr/local/bin and the full build is in /usr/local/stow/cacti/cacti65. Copy /usr/local/stow/cacti/cacti65/cache.cfg to your own home directory. You should also take a look at the README file in the same directory. In order to run cacti for a particular experiment you will need to modify the appropriate parameters in the cache.cfg file and save it in some experiment specific file, which let's say is expt21.cfg. Bottom line here is that you don't want to modify the original cache.cfg in case your subsequent edits somehow result in an illegal .cfg file. You can then run cacti by typing:

```
./cacti –infile expt21.cfg > expt21_out.txt
```

to get the results in a file that you name but here is expt21_out.txt. Also download the technical report for CACTI 6.0 which is also on the web page. Note that the tech report for 6.5 isn't available yet but the 6.0 version will help you understand what is going on a bit better. If you're still confused you might want to Google the CACTI 5.3 tech report and download that as well but I don't think you'll need this.

Basically what CACTI does is explore the entire design space as constrained by the .cfg file to give you the best design. So it's important to figure out what the parameters mean in the .cfg file. Some configurations truly won't make sense and in these cases CACTI will throw an error saying it can't find a solution. One example is if you ask for a design which has the absolute minimum delay and the absolute minimum power then no such solution exists since fast and low power are in conflict – it will find a minimum delay version and it will find a minimum power version but they won't be the same design.

Understanding the .cfg file parameters

In order that they appear in the cache.cfg file:

-size (bytes) nnn – where nnn must be a power of 2 – if you pick a number that's too big then you'll get an error – it's represents the cache capacity

block size is the size of a line in a direct mapped cache or a way in a set-associative cache

-associativity can be 0 or some power of 2:

- 0 means fully associative (it's not strange since an integer meaning infinite is hard to represent)
- 1 means direct mapped
- other powers of 2 are the number of ways in the set

Ports are all differential and the single ended read port parameter must be 0 since these are no longer supported since in today's world they don't make sense due to signal integrity issues. Note that caches will typically have only 1 read-write port, but a register file may have multiple read exclusive ports and multiple write exclusive ports.

We discussed banking in class – if all banks are connected to the same bus then this organization is known as UCA for uniform cache access – UCA bank count specifies the number of banks – must be a power of 2. For reasons that you don't need to worry about in this homework that bus is an H-tree.

Technology is the process technology used for fabrication in microns – e.g. 0.032 is a 32 nm process – today Intel's Nehalem is fabricated in 45 nm.

CACTI is also useful for modeling main memory organizations and the next 3 parameters are only meaningful in this context which we will ignore in this homework: namely page size, burst length, and prefetch width. Just leave these values unchanged since you will be selecting a subsequent parameter which will casue them to be ignored.

In today's world there are multiple transistor types that are available and CACTI gives you the choices of what to use:

- itrs-hp: high performance
- itrs-lstp: low standby power (a.k.a. leakage)
- itrs-lop: low operating power (a.k.a. dynamic power)
- lp-dram: this means a dram cell built from a logic process transistor (a.k.a. embedded DRAM)
- comm.-dram: this means a dram cell using a dram process transistor which is used to build the standard commodity drams that you can buy.

All caches have both tag and data storage arrays. In general the peripheral logic should be itrs-hp since this logic gets the bits from the storage mats (or sub-arrays) to and from the output drivers. The choice of cell type varies – for small memories hp makes sense, for large memories where power is a concern then lstp, or lop will make sense. For very large memories area becomes a concern and you will need to consider lp-dram. If you're modeling a cache or memory that is off-chip then a non-logic process can make sense in which case comm.-dram might be considered.

For this homework where we are only considering caches, it's important to match the bus width to the line size. In more general terms if the bus size is bigger than the line size it means you've made a mistake. If the bus size is smaller than the line size then you will get erroneous power and access numbers since they will represent the time and energy required to do one bus-sized transaction rather than the delivery of the entire line.

Operating temperature in the CACTI 6.5 only affects leakage currents. In general it also affects transistor speed to some extent but this isn't currently modeled. The value represents degrees Kelvin.

Cache type is a bit of a misnomer but allows you to control which kind of memory structure you are modeling:

- Cache – means a cache which will have a tag array
- Ram means something like a register-file, scratch RAM where there is no need for a tag array.
- Main-memory means that there is no tag array and every transaction will occur in page-sized chunks.

Tag size can be specified as default which will work for basic caches, but if you're modeling something more real like a known virtual address size and a known physical address space then you'll need to figure out how many tag bits you need. This will also be the case if you were to model distributed shared memory directories, branch target buffers, etc.

There are 3 access modes:

- Fast – data and tag accesses happen in parallel and the whole set is broadcast over the H-tree bus
- Sequential – tag access happens first and then the data array access as would be the case in a physically tagged physically addressed cache
- Normal – data and tag accesses are done in parallel but the tag is then used to select which way is to be delivered over the H-tree bus

Deviate will not be used in this homework but in general is allows you to prune the CACTI search space in terms of % of deviation from the best possible design.

In any realistic design, a combined metric of merit is important. Usually both energy consumption and delay are the main considerations although other things like area, leakage or dynamic power may be important. For a mobile device that is mostly idle such as a cell phone, leakage is critical to battery life. For a device that is mostly active, then dynamic power becomes important. These parameters can come into play with the deviate parameter in which case the Optimize parameter must be set to "NONE" – however in this homework you will use ED (energy x delay) or ED^2 (energy * delay^2) optimizations.

NUCA stands for non-uniform cache access which means that an interconnect structure is used to go to the appropriate memory array and local accesses will be faster than more remote accesses. For this homework we will stick to UCA (uniform cache access) organizations.

Differential wire signaling can use the following attributes:
"lowswing" wires use less energy to charge and discharge wire capacitances but require the use of more sensitive sense amplifiers.
"fullswing " wires use simpler sense amplifiers but require more energy to charge or discharge the capacitive load quickly
"default" implies that CACTI should search both options and report the best result

In general big memories are slow and small memories are faster so a common strategy is to build a larger memory out of a smaller number of storage arrays called mats. Wire pitch inside or outside of the mats has influences bandwidth per unit area and delay.

- "global" means full pitch wires (twice the size of semi-global) which reduces bandwidth but improves delay.
- "semi-global" increases bandwidth per cross sectional area but the thinner wires have a higher RC time constant which increases delay.

For this assignment you can leave these values unchanged.

Interconnect projections of what we'll see in future processes varies from conservative to aggressive estimates. The values that CACTI uses are taken from Ron Ho's excellent Ph.D. thesis. Note that in the basic .cfg file neither is commented. The safest thing to do is to make sure for any parameter that only one option is selected. However the operational reality is that if multiple values are selected only the last one will be the value assigned to the parameter just like in any program text.

You can select a print level detail as either concise or detailed. Concise just provides output about the overall values of the design, whereas detailed breaks down the various parameters separately for tag and data arrays for configurations where this makes sense.

ECC can be true or false and while there are numerous ECC styles the current version of CACTI uses a simple parity bit for 8 bits of memory width. This model is consistent with current commercial DRAM practice.

-Print input parameters should always be set to "true" for this assignment – this simply puts a preface in your output files so that you can keep things straight in terms of which results correspond to which CACTI parameters. In a more long term research effort you may want to use scripts to set the .cfg file in which case you wouldn't necessarily need them in the output file.

Since we will stick to UCA for this assignment we will ignore the remaining NUCA parameters.

For each of the following questions:

"Describe your experiments" means specifying which .cfg file parameters you changed from the basic cache.cfg file. You should justify these choices – the answer can be as simple as "Parmeter x was varied in steps of y from q to z." etc.

Problem 1: [40 points] The cost of increased associativity

L1 caches today vary from 16 KB to 128 KB and tend to be highly associative in order to reduce conflict misses. Create a set of experiments to characterize the cost of increased associativity for 4-, 8- and 16-way set associative caches for caches in this capacity range and for all of the available technology options and in terms of ED and ED^2 optimizations.

In writing your answer, provide a description of the experiments that you ran to answer this question, list the experimental setups that you did for each experiment and justify your selection of the appropriate .cfg file parameter values.

For your conclusion, describe how you expect the cost of increased associativity to scale with improved technology.

Problem 2: [20 points] The rule of thumb – fact or fiction

In your text and in lectures you learned that doubling the size of a cache with a fixed level of assoiciativity yields the same conflict miss ratio as doubling the associativity. Note that this rule of thumb does not take into account energy or delay but merely the performance impact of the miss ratio. Construct an experiment using three different L1 cache points to draw conclusions about how cost in terms of energy and delay are also affected. Again as part of your answer describe your experimental setup in terms of .cfg file parameters and base your conclusion the results of your experiments.

Problem 3: [20 points] L2 caches

L2 caches tend to be direct mapped are increasing in size and hence the L2 access delay will grow unless banking is used. For L2 cache sizes between 1 and 8 MB, construct an experiment covering all of the supported process technology steps that considers between 1 and 16 banks. Describe your experiments in terms of the .cfg parameters. For your conclusion, predict how energy and delay vary with banking and process technology.

Problem 4: [20 points] Very large caches

With multi-core processors we will likely see very large L3 caches which are shared between the cores. The caches may well vary from 32 to 128 MB. The huge size of these caches will create both area and power issues. For a 32 nm process find a cache organization that is ED optimized. Describe the experiments that you conducted to find this organization, and describe whether it will be consistent with a goal consuming 15 watts of power at 2.5 GHz with an activity factor of some core accessing the L3 every 4^{th} cycle on average and occupying an area of .5 cm^2.