EE193 – Emerging Memory Technologies
Fall 2021
Prof. Marco Donato

Lab assignment #2 – Evaluate eNVM-based cache design with NVSim

In this assignment you will learn to evaluate the design of different memory architectures based on non-volatile technologies using a memory modeling tool called NVSim[1].

Once you are connected to the `homework` cluster, copy the NVSim source code to your work directory (e.g. your `$HOME`):

```
$ cp –r /ee/193EMT/NVSim .
```

You can then compile the code by running the following command:

```
$ cd NVSim && make
```

NVSim allows for three distinct levels of customization:

1) *Memory Cell Technology:* You can select different memory technologies by either choosing one of the existing memory input files (`*.cell`), or by providing your own cell definition;
2) *Memory Configuration:* You can define the memory design space search by modifying the configuration file (`*.cfg`) to specify design parameters such as Design Target (cache vs scratchpad), Optimization Target (area, latency, energy…), technology node, and other memory array organization parameters;
3) *Memory Model:* You can directly modify the model C++ code to edit or add features to the memory model;

Tips for an efficient simulation flow:

The tool you will use in this assignment relies on input configuration files and a command line interface (as opposed to a GUI) to setup the simulations. This is common to many architecture level modeling tools which target design space exploration. While you can edit the configuration files using a text editor and manually run each simulation, it is usually helpful to automate these mundane tasks. The following are some tips that should help you streamlining this process:

1) It is usually good practice to redirect the simulator output to a log file, which can be parsed at a later stage to extrapolate the required information. You can redirect both `stdout` and `stderr` to a log file *and* diplay the results on the terminal using the following:

   ```
   $ command 2>&1 | tee log_file
   ```

2) Using text manipulation commands (e.g. `sed`) can help you automate the process of updating the simulation parameters across a large set of values and permutations. Alternatively, you can

[1] The main characteristics of this tool are described in this paper: *X. Dong, C. Xu, Y. Xie and N. P. Jouppi, "NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 31, no. 7, pp. 994-1007, July 2012*

use scripting languages (e.g. python, perl…) to generate different variants of the input configuration files and schedule the different simulation runs

3) A convenient way to collect and plot data (especially for NVSim) is to parse the output log file (see (1)) for the desired values and collect results for each simulation run in a comma-separated values (csv) file. This can then be imported in a spreadsheet or parsed with pandas in python to plot your results

Part 1: CMOS SRAM cache baseline

As a first step, you will characterize different CMOS SRAM cache designs. Use the sample configuration file provided in the NVSim work folder (./`cfg_files/SRAM_cache.cfg`) to simulate various cache sizes and associativity configurations. In order to change the memory size and associativity you will have to edit the "Capacity" and "Associativity" parameters in the configuration files (reasonable ranges are 16kB to 2MB for capacity, and up to 16 for associativity). Note that you can also modify the cache block size by editing the "WordWidth" parameter as well. Once you have finalized the configuration, you can run the simulation with the following command:

```
$ ./nvsim ./cfg_files/SRAM_cache.cfg
```

The tool will search the memory design space and print the design features for the resulting memory architecture. For this part of the assignment, plot the **read access time vs cache size and associativity** and **read energy vs cache size and associativity**.

Part 2: eNVM-based caches

Now that you have established a baseline for your experiments, repeat the simulations from Part 1 using different eNVM technologies. The underlying technology used to implement the memory can be set in the configuration file by editing the "MemoryCellInputFile" parameter to point to the desired cell configuration file. For this simulation you can use the files included in the `cell_defs` folder, which describe STT-RAM and RRAM memory cells. In addition to the plots described in Part 1, plot the **total leakage power vs cache size** for the three technologies.

Part 3: eNVM storage density

One of the advantages on eNVM technologies is to provide denser on-chip storage compared to traditional SRAM. A common metric used to define the memory cell area is expressed in terms of F^2 where F represents the minimum feature size for a given technology. For this part of the assignment, we are going to focus on a simpler memory macro to analyze the area of a traditional CMOS SRAM and a RRAM crossbar memory. In order to isolate the analysis to a single memory macro rather than an entire cache you will have to change the "DesignTarget" parameter to "RAM". In addition, as a way of improving the storage density, the simulator can be set to prioritize the memory silicon footprint by changing the "OptimizationTarget" to "Area". In your report, address the following points:

1) Based on the cell definitions used in your simulations, what area reduction would you expect to achieve by replacing a CMOS SRAM memory array with a RRAM crossbar array?

2) Plot the total memory area for the two technologies as a function of the memory size ("Capacity"). Are the results consistent with your prediction? What trends are you able to observe from your simulation results?
3) Can you identify any aspect in the memory architecture design that may cause any unexpected result?

In your report, describe the simulation settings you have used for your experiments. For each of the three parts in this assignment your report should also include a detailed discussion emphasizing any trend that has emerged from the simulation results and highlight whether these are in agreement or in contrast with the expected outcome.