EE-194: Advanced VLSI
HW #3: simple binning

In this homework, you will write code to decide how to adjust the voltage for a chip to place it in the bin that sells for the most money. The file binning_simple.cxx already contains several functions:

- *main*(). It creates a chip at a random process corner and gives the chip to you to decide what bin to place it in.
- int *best_bin* (const Chip &*chip*) is the function that you must write. Using the functions just described, you decide which bin to place *chip* into, and return that bin (specifically, an index into the global vector *g_bins* described below). Note that you must also fill in *g_bins*[*your-best-bin*].*fpv.V* with the correct voltage.
- *run_test_pass* (const Chip &*chip*, int *V*, int *freq*, bool **pass*, double **power*). This function allows you to test the chip at the given voltage and frequency. It returns *pass*, which tells you if the test passed. It also returns how much *power* the chip used when running the test. Even though this function is the only way you get information about the chip, one of your goals is to be clever and call *run_test_pass*() as few times as possible.
- *data_collect* (int *V*, int *freq*, bool *pass*). Once you have run *run_test_pass*(), you may call *data_collect*() to save away the results for future reference. Specifically, you may then call *data_bound* (FPV *fpv*, int **Vmin_pass*, int **Vmax_fail*). The *fpv* tells what voltage and frequency you intend to run a new sequence of tests at; *data_bound*() looks at all of the data you've given it and returns two values. *Vmin_pass* is the minimum voltage at which the test will definitely pass (and is the minimum voltage for which you've already told *data_collect*() that the test passes as this frequency). Similary, *Vmax_fail* is the maximum voltage at which the test is guaranteed to fail at this frequency. Note that you do not need to use these functions; however, they may help you minimize your calls to *run_test_pass*().

The file also contains various data structures:

- struct *Bin* describes a particular bin. It starts with a *price*; i.e., how much money a part in this bin sells for. It then has an *fpv* structure (which stands for frequency/power/voltage). The *fpv.freq* field describe the frequency that chips in this bin run at, and *fpv.power* states the maximum power chips in this bin are allowed to dissipate. Finally, *fpv.V* says what voltage the chip should run at in order to meet the frequency and power specs. Note that while *freq* and *power* are provided to you, you must set *V* yourself. Furthermore, you must set it to the minimum value that meets frequency, so as to minimize power as much as possible.
- vector<Bin> *g_bins* is a vector that lists all of the possible bins a chip can be placed in. You must choose from one of these bins.
- struct *Chip* is the structure for a chip. Its instance variables describe the chip's process corner. However, you are not allowed to look at them; you can only call the various functions already described.

***Logistics*:**

- Remember to compile with the line **c++ -std=c++11 binning_simple.cxx**. You may add debugging flags as needed.

- Submit your binning_simple.cxx via **provide**.
- You may use any Linux machine you choose; either any box in any of the Linux labs, or the dell24 homework server.