University of Pennsylvania
Department of Electrical and Systems Engineering
Electronic Design Automation

ESE535, Spring 2013

Assignment #5

Monday, March 11

Due: Assign 5a: Monday, March 18, beginning of class.
Due: Assign 5b: Monday, March 25, beginning of class.

Resources You are free to use any books, articles, notes, or papers as references. Provide citations in your writeup as appropriate.

Collaboration You may **not** discuss algorithmic and testing approaches. You may give tutorial assistance on using OS, compiler, and debugging tools. All code development should be done independently. You may **not** share code or show each other code solutions. All writeups must be the work of the individual.

Writeup Turn-in assignments on blackboard. See details on course web page. No hand-writing or hand-drawn figures. Details for turnin are at the end.

Project Goal for this phase Place LUTs into specific cluster locations aware of the exact defects on a particular chip so that it will yield while minimizing total linear wire length.

You may use more rows and columns. The wirelength minimization both acts to encourage good placements and to minimize the size of the array you use.

Opportunity Knowing the exact defects for a particular chip, you can avoid placing logical LUTs into a cluster that would be incompatible with the cluster defects. If necessary, you can spread-out the design onto a larger array to access more potentially compatible LUTs.

Assignment 5a Task Produce a legal mapping that will yield.

- The number of LUTs assigned to the cluster will not exceed a specified cluster size.
- The total number of distinct inputs to the cluster will not exceed a specified number of cluster inputs.
- The LUTs assigned to a cluster can be matched to compatible, partially defective physical LUTs in cluster

Write a function for computing the change in wirelength that would result from moving a block to a new (x,y) location in the mesh. This function must **not** use the global wirelen function in `mesh.c`, and it should be efficient. Treat wires as point-to-point Manhattan-distance wiring (no optimization for shared fanout wires on a net) and include both the wirelength of the inputs to the block and the output of the block connected to its successors.

Assignment 5b Task Minimize wirelength while achieving the legal mapping that will yield.

Even with the two components, 5a is probably less than a week's work, while 5b is open ended. You would do well to plan to finish 5a early and start working on 5b before the 5a deadline.

Code Base: We extend the same basic code base we have developed in Assignments 2–4.

Pickup the code in `assign5.tar` from `~ese535/spring2013/assign5.tar` on `eniac`. Unpack it with `tar -xvf assign5.tar`. Run `make` to build. This should produce an executable `dplace` which you can run. The `makefile` in the `test` subdirectory runs `dplace` on the various cases needed for this assignment and provides an example of how to use it. Please use the architecture and target parameters in the `makefile` for producing your results for this assignment.

Representations: No representation changes for this assignmet. For this assignmet the (x,y) positions take on their traditional signficance.

You may want to consult the VPR manual (available in `~ese535/spring2013/manual_430.pdf`) for descriptions of the mesh architecture and placement coordinate system. Particularly Figure 2 shows what the basic module of a LUT and FF looks like. Figure 10 shows the coordinate system.

A quick overview of code:

- `dplace.c` — contains the new `main` function that drives the placement. There are now two different case statements operating on approach inputs. The first is a matching approach and the second is a placement approach. These are selected by the `-approach` (for placement) and `-match` (for matching) options. The makefile in the test directory is setup to provide the options. The cryptic file name `.mXpY` (.m2p0, .m2p1, .m2p3) correspond to these options. We will be using your match (mapping 2) for this assignment.
- `test_chip.c` – reads in test chips, actually calls your placement routine and matching functions, and collects and writes results.
- `sequential_place.c` – will not achieve a defect-free assignment; this has been changed to automatically increase the size of the mesh until it can place everything; this means there is no more need to use `-pside 1`. Note that you must reload the defect map **after** resizing the mesh as shown here.

You need to complete code in:

- `your_place.c` – your_place_legal and your_place_good – this is where your Assignment 5 answers goes.
- `delta_wirelen.c` – you need to fillin the code for `delta_wirelen` here for Assignment 5a.

You will also need your `transform.c` and `cost.c` from Assignment 2 and `match.c` from Assignment 3.

Note that the placement routine (your_place) is called **after** chip defect maps are loaded. This is different from how your_pack was called in Assignment 4.

Caveat: The code not borrowed from t-vpack/vpr (`test_chip`, `mesh`, `sequential_place`) was newly written or heavily revised for this assignment. While we have tried to test it, like any recently developed code it may contain bugs. Let us know if you have any problems. Similarly, we may need to provide updated source as we fix bugs or add additional functionality. The code for Assignment 5 fixes the errors flagged earlier where there was confusion between xdim and ydim.

Assignment 5a turnin: You will need to upload two files. We have created separate assignments on blackboard so that you only need to submit a single file to each assignment.

1. **assign5a-writeup**: a single PDF with
   - Pseudocode for your legal LUT placement algorithm
   - Explanation of your legal LUT placement algorithm
   - Table of results reporting the runtime, array dimensions, and wirelength for each of the benchmarks (7) on each of the provided test chips (9). This is a total of $7 \times 9 = 63$ cases.
   - Pseudocode for your delta_wirelen algorithm
   - Explanation of your delta_wirelen algorithm
   - Assessment of the complexity of your delta_wirelen algorithm in terms of the relevant variables in the program, such as lut_size, num_blocks, mesh x and y dimensions, inputs_per_cluster; many of these are **not** relevant, so you need to identify the ones that are and write an expression in terms of the relevant ones.
2. **assign5a-code**: a single tar file with your code (no binary files, but in an archive like the provided support so it can be unpacked and built)
   - run `make clean` in both the code and test directories
   - use `make assign5.tar` to create the tar file
   - test that you can unpack your `assign5.tar` and build and run tests from there before you upload to blackboard; we will build your code and test it.

Assignment 5b turnin: You will need to upload two files. We have created separate assignments on blackboard so that you only need to submit a single file to each assignment.

- **assign5b-writeup**: a single PDF with
  1. Pseudocode for your wire-length minimizing LUT placement algorithm
  2. Explanation of your wire-length minimizing LUT placement algorithm
  3. Highlights from the tuning and experimentation you used to arrive at your final algorithm. Use graphs and quantitative results to support the conclusions you drew.
  4. Table of results reporting the runtime, array dimensions, and wirelength for each of the benchmarks on each of the provided test chips for **both** your legal placement routine from Assignment 5a and your wire-length-minimizing placement routing from Assignment 5b.
- **assign5b-code**: a single tar file with your code (no binary files, but in an archive like the provided support so it can be unpacked and built – same as above)