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

ESE535, Spring 2011

Assignment #4

Monday, February 21

Due: Wednesday, March 2, 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 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. See details below on what you need to turn in and the format.

Assignment 4 Task Develop and implement an algorithm to schedule and place a circuit netlist, C, onto a time-multiplexed, programmable substrate:

- meeting a specified bound on the number of netlist nodes assigned to a PE
- achieving a target makespan (time-bounded scheduling) accounting for PE to PE physical delay based on both inter-cluster net delay and Manhattan travel distance
- while minimizing the energy required to evaluate the netlist C:

$$\begin{aligned} E &= E_{inter\_cluster} \times \sum_{\text{PEs } p} (p.inputs + p.outputs) + E_{intra\_cluster} \times \sum_{\text{Blocks } b} (b.inputs) \\ &\quad + E_{manhattan\_hop} \times \sum_{\text{Blocks } b} \sum_{\text{PEs } p} \left(c_{b,p} \left(|b.x - p.x| + |b.y - p.y| \right) \right) \end{aligned} \quad (1)$$

For this week, we model the fact that distance between PEs makes communication slower and higher energy. Aside from the change in delay and energy costs, the architecture is unchanged from Assignments 2 and 3. The new term charges energy proportional to the Manhattan distance between each net source and each PE that it must drive ($c_{b,p}$ is 1 if block b provides an input to PE p, 0 otherwise). If you have succeeded in clustering multiple consumers of a net into a PE (a goal from clustering that might still be valuable here), then you only pay the interconnect energy once moving the signal to the PE. `compute_energy` has been updated to include the new Manhattan interconnect energy costs. $E_{manhattan\_hop}$ is new and also provided as an argument input to `sched_main`.

Code Base Changes:

Pickup the code in `assign4.tar` from `~ese535/spring2011/assign4.tar` on `eniac`. Changes have been made to `mesh.c` and `sched_main.c`.

Notably:

- `interconnect_delay` has been updated to account for the additional delay required to move data between PEs.
- `compute_energy` has been updated to compute the energy associated with interconnect between PEs based on distance.

Please keep your assignment 2 scheduler and assignment 3 cluster+scheduler for comparison and add a new placer+scheduler for this assignment. The test `Makefile` is setup to assume your assignment 2 scheduler is option 2, your assignment 3 cluster+scheduler is option 3, and your new placer+scheduler is option 4. Comparing your assignment 2 and 3 mapping results to your assignment 4 placer+scheduler allows you to quantify the benefits of directly addressing the clustering and placement goals versus ignoring them.

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.

- **assign4-writeup**: a single PDF with
  1. What are the pros and cons for each of the following general strategies:
     (a) Phase ordered: cluster then place – e.g. run your assign 3 solution to produce clusters of blocks that will go on a single PE, then run a second algorithm to assign the particular PE (a unique x, y location in the mesh) for each cluster.
     (b) Phase ordered: place then cluster – e.g. run placement on a larger mesh where each PE holds a single block (so mesh is larger by a factor of the cluster size) then group together sets of adjacent PEs to produce the clustered PEs in this architecture.
     (c) Combined clustering and placement – e.g. treat as a three-dimensional placement problem (where the dimensions are x, y, slot (timestep)).
  2. Description of your algorithm and the code that supports it.
     (a) A high-level, English text description of your strategy. Use references if appropriate. The description should relate your solution to the previous question (possibly distinguishing it form all 3, if appropriate).
     (b) Define any cost functions used by your algorithm.
        i. For your algorithm, is it important to compute the changes to these cost functions based on an incremental move or potential assignment?
        ii. How does the cost function reward an incremental move or assignment toward the overall goal of energy minimization?
        iii. How expensive (in runtime) is it to compute the change in cost for an incremental move or assignment?
     (c) Provide pseudocode for your algorithm.

3. Complete a version of this comparison table with your energy results based on the energy costs in `test/Makefile`:

Design	Energy @ 10.0×ASAP delay						
	ASAP	assign2	assign3	assign4	$\Delta_A\%$	$\Delta_2\%$	$\Delta_3\%$
example	236						
e64	18034						
s1423	8180						
alu4-em4	99105						
frisc-em4	160547						
s298-em4	152048						

   $\Delta_A\%$ is the percent improvement of your assignment 4 algorithm over the ASAP baseline:

$$\Delta\% \text{ improve} = \frac{ASAP\_Energy - Assign4\_Energy}{ASAP\_Energy}$$

   $\Delta_2\%$ is the percent improvement of your assignment 4 algorithm over your assignment 2 algorithm:

$$\Delta\% \text{ improve} = \frac{Assign2\_Energy - Assign4\_Energy}{Assign2\_Energy}$$

   $\Delta_3\%$ is the percent improvement of your assignment 4 algorithm over your assignment 3 algorithm:

$$\Delta\% \text{ improve} = \frac{Assign3\_Energy - Assign4\_Energy}{Assign3\_Energy}$$

- **assign4-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; this means the make file should be updated to build the application with any additional source files you may have added)
  - run `make clean` in both the code and test directories
  - use `make assign4.tar` to create the tar file
  - test that you can unpack your `assign4.tar` and build and run tests from there before you upload to blackboard