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

ESE535, Spring 2015 Assignment #7–8 Wednesday, March 4

Due: Assign 7: Thursday, March 19, 10PM
Due: Assign 8: Thursday, March 26, 10PM

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.

We expect everyone to abide by Penn's Code of Academic Integrity. `http://www.upenn.edu/academicintegrity/ai_codeofacademicintegrity.html` If there is any uncertainty, please ask.

Writeup Turn-in assignments on canvas. See details on course web page. No handwriting or hand-drawn figures. See details below on what you need to turn in and the format.

Project Goal for this phase Develop a detailed router that assigns each net to specific switches on specific waves, minimizing the number of waves necessary to route the entire design.

This is a two week project, with the functioning code due at the assignment 8 date. Use your experience from previous assignment to plan. I expect you should have code for a basic level of functionality written and into debugging by the assignment 7 due date, so that you can finish debugging, optimize, and get solid results for the assignment 8 due date.

Assignment 7 exercises:

1. Consider doing simulated-annealing placement of the LUTs (and IOs) onto PEs for the heterogeneous multicontext computing array. Identify a suitable cost function to use with simulated annealing and describe its suitability in terms of:
   (a) Relation to assignment 2 cost functions
   (b) Ability to reward every incremental move that makes progress toward the assignment 2 cost function goals
   (c) Time complexity of computing the Δcost for a proposed moved

2. Thinking of routing on the heterogeneous multicontext computing array as a list-scheduling problem, describe a suitable priority function to order the routing nets? (What properties of nets should you consider? How do you combine or select among these priorities? What is your complete ordering strategy?) [**Hint:** This may be relevant to your routing solution.]

3. Consider using Pathfinder to route for the heterogeneous multicontext computing array. Provide short (one or two paragraph) answers describing key changes or adaptation needed to make Pathfinder applicable:
   (a) Route strictly by level. Each level should be routed in the smallest number of waves, but each level is given its own set of waves. So, level 0 might get waves 0 and 1, level 1 might get waves 2–5, level 2 waves 6 and 7, etc.
   (b) Route by level in sequence from level 0 but allow routes to start as early as predecessors allow, even if it is before the nominal first wave of a level. Continuing the example above, a level 1 net whose inputs are satisfied by wave 0 could attempt routing in wave 1.
   (c) Route all levels together. Include a description of what makes this harder in your answer.

   [**Note:** This is probably not relevant to a routing solution you will develop during the two weeks of this assignment but would be relevant if you wanted to spend more time developing a higher quality router.]

Code notes and instructions:

- Code base is augmented from assignment 3. Pickup updates in `~ese535/spring2015/assign7.tar` on `eniac`.
- `detail_route.c` is where you need to complete your router code.
- `model.c` and `energy.c` provide a new, more detailed cost model structure. Since we provide the code and its invocation, you do not need to work with this directly. Current model.c is a placeholder that we intend to update the week after Spring Break (around the time assignment 7 is due, so that you have the revised version for your assignment 8 experiments).
- `tree_energy.c` applies the energy model and writes out the final cost estimates.
- `tree.c` has been augmented to support the statistics gathering for `tree_energy.c`.
- We have revised `main.c` to invoke the detailed router, check the detail routes, and call the routines in `tree_energy`.
- We broke out `grow.c` to call functions from assignment 2 and return the level growth schedule.
- We do not require that you implement a complete congestion-negotiation router for assignment 8.[1] We simply require that (1) all nets are routed, and (2) you make effort to minimize the waves during routing.
- `heapsort.c` provides a sort implementation that may be useful.
- We added `valid_detail_routes` to `check_route.c` to check the validity of routes you compute.
- Note the `update_timestep` function in `tree.c`. When you route the output of a LUT (or input), use this to record the wave for the route. This should be consistent with the routing wave (domain) used in the interconnect. This is where `valid_detail_route` (called from `valid_detail_routes`) figures out which domain should hold the route. It is also how `check_routing_precedence` (also called from `valid_detail_routes`) checks the schedule of LUT evaluations and routing is reasonable.
- You will likely find the detail route validator above and the global router (`global_route.c`) valuable references for working with the tree data structure and domains. The global router uses the domains as we did in assignment 2,[2] but it does illustrate how to traverse the tree. The detail route validator uses domains as we will use for this assignment; it is the piece of code that validates when routes are properly formed.

[1] That might be something to do for the project.
[2] So, is not an example of how you should use domains for your detail router.

- We added `dump_detail_route_domains` to `tree.c`. This will likely be useful in scrutinizing routes made by your router (and route failures).

For this assignment, we will use the domains in the switches differently. Rather than treating them as a set (as we did for partitioning and global routing), we will use the domains to keep track of signal occupancy during routing waves. That is, each domain will correspond to a particular routing wave. A signal started into the network on a wave, can only use switch outputs (inputs) and wires on that wave, so it can only occupy corresponding domain slots in the tree switches between the source node and the sink node. So, for example, when the PE sends a LUT output out over domain 3, the level 1 switch will receive it on domain 3 and send it out to a parent or sibling on domain 3.

Note that we still run global routing on the binary tree first using the set interpretation in order to identify the appropriate growth schedule (which we extract using the routine in `grow.c`, which in turn calls your functions from assignment 2). Then, we create a new tree using the growth schedule for this detail routing. However, we force the leaf growth (`glevel[0]`) to the value specified by a new command-line option `physical_c` (which we will take as 1 for the primary experiments).

Most nets fanout to multiple sinks. When routing a net, you want to use as few wire segments as possible. This, typically means sharing the wire segments for the prefix of the routes as much as possible. For example, if block A in location 0 is an input to block B in location 7, C in location 6, and D in location 6, then you would prefer to have a single set of wire segment routing up to the least common subtree at height 3. Furthermore, you should be able to take a single wire segment down through the height 2 switch to the 6, 7 tree, where the route would need to split to reach both the 6 and 7 PE. You only need to route A once to location 6 to provide the input for both C and D. This also means you would like to perform this route in a single domain.[3]

Legal routes must respect LUT precedence. The inputs to a LUT must arrive on waves (domains) that precede the LUT output. Your routing effectively (re)schedules your LUTs.[4] In this manner, it may also be able to reduce the number of waves needed by better scheduling the use of routing tracks.

It may be useful to compute an ALAP schedule time for the nodes.

[3]The code is only setup to route a net in a single domain, so don't get fancy and try to split a net across multiple domains.

[4]This is why you need to use `update_timestep`.

Assignment 7 turnin: [50pts] a single PDF with

- Answers to assignment 7 exercises.
- Description of any additional data structure you use for routing.
- Pseudocode for your routing algorithm
- Explanation of your routing algorithm and data structures, including what you do to attempt to minimize the number of waves required.
- Implementation and results status – describe the status of your implementation and summarize any results you have already been able to obtain.

Assignment 8 turnin: [200pts] You will need to upload two files. We have created separate assignments on canvas so that you only need to submit a single file to each assignment

1. **assign8-writeup**: a single PDF with
   - Description of any additional data structure you use for routing.
   - Pseudocode for your routing algorithm
   - Explanation of your routing algorithm and data structures, including what you do to attempt to minimize the number of waves required.
   - Description, data, and graphs from any experimentation and tuning you performed.
   - Table of results for the provided benchmark set reporting the (i) global routing waves, (ii) achieved detail routing waves, (iii) final routed energy cost according to supplied model. Provide (i), (ii), (iii) for **both** the dummy partitioner (cost2 target) and your most advanced, level-oriented, recursive bisection partitioner (assignment 6, cost6 target).
   - Discussion of your results.
2. **assign8-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 assign78.tar` to create the tar file
   - test that you can unpack your `assign78.tar` and build and run tests from there before you upload to canvas; we will build your code and test it.