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

ESE535, Spring 2015

Project Assignment

Wednesday, March 25

Due: Proposal: Thursday, April 2, 10PM.
Due: Milestone 1: Thursday, April 9, 10PM.
Due: Milestone 2: Thursday, April 16, 10PM.
Due: Milestone 3: Thursday, April 23, 10PM.
Due: Final Report: Wednesday, April 29, 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 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 will consider coupled projects. However, there should be clear pieces of the project that are the work of each individual. It should be possible to evaluate each piece independently in addition to evaluating the composite effect of the pieces. For example, assignment 6 and assignment 8 could be considered coupled but independently evaluatable. You could evaluate placement with global routing (as we did) and routing with the dumb placement from assignment 2. Then, you could evaluate the composite effect of using good packing and good placement together.

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.

Nature of Project The project is a more open-ended optimization of your choosing. The first deliverable is the project formulation proposal—essentially an assignment statement—for your project. Including the proposal development, you have a total of 5 weeks for the project. As a discipline for you and an opportunity for feedback for the instructor, there will be weekly milestones starting with the proposal. You will select the nature of the other two milestones.

The idea here is to take mapping for the heterogeneous multicontext computing array further in some way and to some depth. We're leaving it up to you to select how. Some examples:

- Make it more real in some ways – we've deliberately made simplifying assumptions to keep the scope of the assignments down. There are several direction that you would want to take this to better match a realistic component or to explore details of the architecture. Examples include:
  - Limited locations for inputs and outputs.
  - Support limited "inputs" at the leaf PEs (take an architectural parameter that can be less than K*luts_per_pe)
  - Properly support independent memory banks per LUT input
  - Properly support latches

  Examples above are intended to be concrete and illustrative. You are welcome to identify other issues that you believe are inadequately addressed by the assignment 2–8 flow for which you could develop suitable models and optimizations.
- Address some piece of the flow we have not tackled in class – we have focused mostly on physical optimizations (placement, routing); you may want to explore some other piece that we've covered in class but not on assignments. Examples include:
  - high-level synthesis to this target (e.g., SPICE netlists [2], GraphMachine [1])
  - logic clustering (e.g., reduce the number of network-crossing routing waves from exercise on assignment 4)
- Develop a better solution to some piece of the flow or a composite optimization that simultaneously attacks multiple parts of the flow – since we only had one or two weeks for each assignment, it was not possible to explore all options and was likely not possible to explore the best approaches for each problem. Furthermore, we have covered more techniques since some assignments were given. You could use this assignment as an opportunity to explore a more aggressive optimization for one of the tasks previously addressed. Examples include:
  - Use SAT, ILP, or pruning search for placement or routing.
  - Develop a proper Pathfinder-style router [4].
  - Integrate or interleave scheduling or global routing along with placement.
  - Adaptive/refined placement.
  - Simulated Annealing placement.

You are free to consider techniques not introduced in the course, including starting with algorithms from the literature that we did not read for the course. One goal of the course is to enable you to read the literature to follow new ideas as they are published.

Project Formulation Proposal Project formulation must include:

- Defining the (potentially revised) architecture model
- Defining the final evaluation cost function
- Defining the optimization task and goal
- Defining any new solution legality checking code for your revised architecture model
- Defining the schedule and milestones
- Defining the evaluation experiments (including benchmark set and targets or parameters to the optimization problem)

Your Project Formulation Proposal will be much like the assignments we have been providing you for assignments 2–8. You are, in essence, creating an assignment for yourself for the project. Reviewing assignments 2, 3–6, 7–8, you will see that all of the above items are defined in them (with assignments 3, 4, 5 and 7 being milestones). Being able to formulate problems is an important skill to develop and one of the goals of this course.

Milestones Since the project is a four week project after you turn in the Project Formulation Proposal, you should identify what you should target completing at each of the intermediate three weeks. What exactly the milestones are will be project dependent, but they should be related to development of the code and evaluation of solutions. Some things you might think about when selecting milestones:

- is there support code (e.g. cost function calculations, legality checks, modifications to key data structures) that should be done early? (during the first week)?
- are there examples you should work by hand first? (like the warmup exercise you did for earlier assignments)
- is there a very simple version you can get running early as a baseline or starting point? This might be particularly important if you are attacking a piece of the flow we did not target in class; this is similar to doing a simple bisection cut (assignment 4) before doing recursive bisection (assignment 5).
- are there experiments you need to run early to help decide how to tune your algorithm?
- perhaps you could target a running implementation by the second week, giving you time to tune and improve it during weeks 3 and 4?

Feedback We will make an effort to get you rapid feedback on the Project Formulation Proposal.

Algorithm Selection and Tuning As a longer, more open-ended project, you should be exploring alternative algorithms/approaches and tuning the parameters (*e.g.* cooling scheduling in simulated annealing, weights in optimization cost functions, ordering priorities for list scheduling). Your final writeup should describe what you explored and what you learned. This may include additional result graphs comparing algorithms and parameters. A good example of an article showing the exploration of tuning parameters is [3]; notably Tables 1.1–1.5 and Figure 4 show the impact of various parameters in their algorithm.

Turnin

Assignment	Points	Pieces	Description
Proposal	100	PDF only	As identified above under **Project Formulation**, like an assignment statement
Milestone 1	50	PDF and code	As you define in Proposal
Milestone 2	50	PDF and code	As you define in Proposal
Milestone 3	50	PDF and code	As you define in Proposal
Final Report	250	PDF and code	Writeup will likely end up looking similar to your answers to assignments 6 and 8 with an additional section on **Algorithm Selection and Tuning** as described above.

References

[1] Michael deLorimier, Nachiket Kapre, Nikil Mehta, and André DeHon. Spatial hardware implementation for sparse graph algorithms in GraphStep. *ACM Transactions on Autonomous and Adapative Systems*, 6(3):17:1–17:20, September 2011.

[2] Nachiket Kapre and André DeHon. SPICE^2: Spatial Processors Interconnected for Concurrent Execution for Accelerating the SPICE Circuit Simulator Using an FPGA. *IEEE Transactions on Computed-Aided Design for Integrated Circuits and Systems*, 31(1):9–22, January 2012.

[3] Alexander Marquardt, Vaughn Betz, and Jonathan Rose. Timing-driven placement for FPGAs. In *Proceedings of the International Symposium on Field-Programmable Gate Arrays*, pages 203–213, 2000.

[4] Larry McMurchie and Carl Ebeling. PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs. In *Proceedings of the International Symposium on Field-Programmable Gate Arrays*, pages 111–117, 1995.