University of Toronto, Faculty of Applied Science and Engineering
Department of Electrical and Computer Engineering

ECE 1387 - CAD for Digital Circuit Synthesis and Layout

Exercise #1 - Simulated Annealing-based Placement and Timing-Driven Negotiated Congestion Routing

November 2013 J. Anderson

Assignment Date: Friday, November 1, 2013
Due Date: Friday, November 8, 2013 (before lecture begins)
Late Penalty: **-1 mark per day late, with total marks available = 10**

The purpose of this exercise is two-fold: 1) to use an automatic placement tool based on the Simulated Annealing optimization strategy, and to gain some familiarity with the properties of that strategy, in particular, the cooling schedule, cost and acceptance functions; and, 2) to experiment with the negotiated congestion routing algorithm described in class.

The placement and routing (P&R) tool you will use in this exercise is called "VPR". It is a tool originally created here at U of T by Vaughn Betz as part of his Ph.D research in the late 90s. The tool has since been enhanced extensively by other graduate students. VPR is now part of the (larger) VTR (Verilog-to-Routing) project which is a complete open-source flow from RTL synthesis, to logic synthesis, to placement and routing for FPGAs. You can check out the VTR paper on the course reading list.

As well as being a P&R tool, VPR is a framework for FPGA CAD and architectural research, used by researchers all around the world. VPR takes a text description of the target FPGA architecture as input. The description specifies, among other things, the make-up of the FPGA logic blocks, the routing segment lengths, the switch block style, and the routing delays. By changing the architecture description file, one can evaluate the speed and area-efficiency of different FPGAs.

Access to Software

The VTR (version 7) code (which includes VPR) can be found at this link:

http://www.eecg.utoronto.ca/vtr/terms.html
Unzip and untar the code, and go into the vpr subdirectory of the distribution. Type make to compile the vpr binary. Note that the VTR release also includes RTL synthesis and logic synthesis, which you will not need to use for this exercise.

Instruction Manual for VPR

The VPR instruction manual is provided in the doc subdirectory of the VTR distribution. You need only to consult **Section 2** of the manual, which has the parameter information you'll need. The complete manual describes other aspects of VPR.

Netlist Files and Architecture File to Use

Use the three test circuit files (*.blif) and architecture file (k6_N10_mem32K_40nm.xml) provided on the course web page. You may find reading the architecture file interesting, as it gives a description of an FPGA architecture.

Exercise A - Simulated Annealing Placement

The VPR program allows the user to set various Simulated Annealing parameters: the starting temperature, the ending temperature, the rate at which the temperature decreases, the number of moves per temperature, and the initial random seed. NOTE: For **all** steps in Exercise A, run VPR with the following parameters: "--pack" (pack LUTs/FFs into logic blocks) "--place" (place design) and "--place_algorithm bounding_box" (minimize HPWL only, not timing). An example command line is:

./vpr k6_N10_mem32K_40nm.xml diffeq1.blif --pack --place --place_algorithm bounding_box

1. Run the VPR program once (see example command line above), on each of the three circuit netlists. Plot the score (cost function) vs. the temperature. **Note:** to avoid VPR's graphics display, you may need to use the "--nodisp" option. Also note that in the placer's text output, you can see the acceptance rate, total # of moves, and alpha (the temperature scaling factor).

2. Run VPR 5 times for each circuit with different random seeds (use the --seed command-line option). Calculate and report the mean and standard deviation of the resulting final scores. Comment on the sensitivity of the scores to the random seeds? Do all circuits exhibit the same sensitivity to the random seed?

3. For each circuit, run VPR 5 times (with different seeds) with 5 different starting temperatures (25 * 3 circuits = 75 runs in total) (use the --init_t command-line option for starting temperature). Be sure to choose some starting temperatures that are low enough to actually impact the placement results (hint: you may need to use starting temperatures considerably less than 1). For each circuit, report the mean and standard deviation of the final score for each starting temperature.

4. One of the ideas that was proposed during the ECE1387 lecture on October 25 was to change the accept function in simulated annealing so that if the computed probability of acceptance is > 0.5, then *always* accept; otherwise reject. That is, in the proposed approach, moves are no longer accepted probabilistically. Modify the accept function in VPR to implement this idea (check around line 1620 of `vpr/SRC/place/place.c`). Repeat step #2 above. Compare the results with those achieved for step #2.

5. VPR implements "range windowing", like the Timberwolf algorithm described in class. The windowing aims to make annealing more "efficient" by reducing the window in which cells may move as temperature decreases. Starting with a fresh version of VPR (not the version from #4 above), disable the windowing by hacking the `update_rlim(…)` function in `place.c` to return immediately (without adjusting the window). Repeat step #2 above. Does eliminating the range windowing impact the HPWL results for these circuits?

Exercise B - Timing-Driven Routing

Note: For this part of the Exercise, use the original version of VPR (not the one you modified in the previous step).

1. VPR also implements timing-driven routing using an improved version of the PathFinder approach described in class. Run the VPR tool with the timing-driven router *and* the timing-driven placer. Report the post-routing critical path delay for each of the test circuits after routing.

   For this step, set the # tracks per channel (W) to 66 using "--route_chan_width 66" for diffeq1, and set W to 42 and 72 for sha, or1200, respectively. An example command line: ./vpr k6_N10_mem32K_40nm.xml or1200.blif --route_chan_width 72

2. Using the same parameters as in the previous step, explore the different parameters of the router and try to improve the critical path delay for the two circuits. Check the VPR manual to find out which parameters are available to change. Comment on the parameters you experimented with and their effect on the critical path delay. Be sure to vary the "present congestion" ("--pres_fac_mult") and "historical congestion" ("--acc_fac") parameters talked about in class.

3. Evaluate the extent to which the placer vs. the router affects the final circuit performance. Repeat step #1 of Exercise B, but run the placer in non-timing-driven mode (--place_algorithm bounding_box) instead of timing-driven mode. Report the critical path delay for each circuit and compute the degradation vs. that achieved in step #1.