

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**

February 2010

J. Anderson

**Assignment Date:** February 19, 2010  
**Due Date:** March 3, 2010 (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 and cost functions; and, 2) to experiment with the negotiated congestion routing algorithm described in class (PathFinder).

The placement and routing (P&R) tool you will use in this exercise is called “VPR”. It is a tool first developed here at UofT by Vaughn Betz as part of his Ph.D research. VPR has been enhanced extensively since then by graduate students, and continues to be extended. 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 textual description of an 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 VPR code can be found on the course web page. Un-tar the code, enter the `VPR_HET` directory, and type `make` to compile the code. I verified that it compiles and runs on `mint.eecg` and `ug132.eecg` [i.e. it works on *both* of the research networks].

#### **Instruction Manual for VPR**

The VPR instruction manual is provided on the course web page. You need only to consult **Section 5** of the manual, which has the parameter information you’ll need. The complete manual describes other aspects of VPR and T-VPACK (a tool that packs LUTs into clusters).

#### **Netlist File and Architecture File to Use**

Use the test circuit files (`alu4.net`, `ex1010.net`, `spla.net`) and architecture file (`k4-n10.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 - Placement and Simulated Annealing**

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: “`-place_only`” (prevent routing) and “`-place_algorithm bounding_box`” (minimize bounding box perimeter length).

1. Run the `vpr` program once, with its default parameters, on the three test circuits. Plot the score (cost function) versus the temperature. If you turn on the “toggle nets” option, you can see the rat’s nest of wires become less tangled (use the “`-auto 1`” option). Indicate on the plots the temperature (roughly) at which placement score “freezes”. **Note:** to avoid the graphics display, use the “`-nodisp`” option.
2. Run VPR 5 times for each circuit with different random seeds. Calculate the mean and standard deviation of the resulting final scores. Comment on the sensitivity of the scores to the random seeds; how many runs (with different seeds) are needed so that the mean final score is not changing much? Do all circuits (ex1010.net, alu4.net, spla.net) exhibit the same sensitivity to the random seed?
3. Using no more than 30 runs of VPR for each circuit, generate a plot of starting temperature versus final score achieved. Choose only 4-5 different starting temperatures so that you have enough runs per temperature to get a reasonable value for the mean, standard deviation at each temperature. You need to choose your temperatures carefully so that the effect of the parameter is apparent. As above, use different random seeds so that you get a set of results for each temperature. Comment on the results and state why starting temperature affects placement quality. For this step, use the parameter “`-exit_t 0.00005`” (stop when temperature drops below 0.00005)
4. In the `SRC` directory, in the file `place.c`, around at line 546, you will see the main annealing outer loop. The function call to “`exit_crit`” returns a non-zero value when annealing freezes. Alter the code and implement a “temperature bumping” technique as follows: when annealing freezes, instead of exiting, bump up the temperature (say by 2X) and allow annealing to continue until it freezes again. Repeat the bumping a fixed number of times (say 3 times) before finally terminating. Repeat Step #2 above and report the mean and standard deviation of final scores when your bumping technique is on. Experiment with various bumping strategies and briefly describe what you tried. **FYI:** the variable `t` represents the annealing temperature in the VPR program.

## Exercise B - Timing-Driven Routing

**Note:** For this part of the Exercise, use the original version of VPR and not your altered version from Exercise A.

1. VPR also implements timing-driven routing using an improved version of the PathFinder approach described in class. Run the placement tool with the default options *and* the timing-driven router. **NOTE:** Do not set “`-place_only`” or “`-place_algorithm`” for Exercise B. By default, VPR finds the minimum number of tracks per channel (`W`) to route a circuit. Report the post-routing critical path delay for each test circuit after routing with the minimum `W`. Now, re-route each circuit with the # tracks / channel (`W`) fixed to a large value (200) using “`-route_chan_width 200`” on the VPR command line. Report the critical path delay for each circuit with `W = 200`. Compare the two critical path delays for each circuit, and comment on what is being evaluated through this experiment.
2. For this question, use `W = 44` for alu4.net, `W = 70` for spla.net and `W = 70` for ex1010.net. Explore the different parameters of the router and try to improve the critical path delay for the circuits. 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. A key goal of negotiated congestion routing is to reduce the impact of net/connection ordering. Alter the code in `route_timing.c` (around line 93) to reverse the order in which nets are routed and repeat Step #2 above. Was the critical path delay affected by the net ordering? (**FYI:** The outer loop at around line 90 of `route_timing.c` implements an iteration of Nair’s algorithm as described in class.)