ECE636: Reconfigurable Computing
Spring 2020 **Handout #2**

Homework # 2

FPGA Placement and Routing: Understanding the Tradeoffs

1 Introduction

Perhaps the most challenging part of implementing a new FPGA architecture is developing an appropriate set of CAD tools to map a design consisting of logic gates to a physical device implementation. While many companies have succeeded in designing FPGA devices that have an abundance of logic gates coupled with ultra-fast routing resources, only those that have developed decent CAD software support have been able to survive. In the last problem set we evaluated a set of metrics that indicated how we should apportion VLSI area in an FPGA device. In this assignment we evaluate CAD algorithm tradeoffs of algorithm run time versus performance.

We've seen that four main steps are needed to translate a circuit from a generic logic representation consisting of simple 2-input gates to a successful implementation in a physical device: technology-independent optimization, technology mapping, placement, and routing. In the following exercises we primarily focus on the latter two issues through the application of two standard algorithms, simulated annealing-based placement and maze routing, and evaluate various tradeoffs of run time versus placement and routing quality. The CAD algorithm implementations considered here are similar to those found in software tools for island-style devices from Xilinx Corporation and Intel/Altera Corporation.

2 FPGA Placement

In this first set of exercises you will get a chance to learn more about simulated annealing, an iterative FPGA placement technique we have talked about in class. Before starting the execises, look at the material related to FPGA placement in Section 3 in [1] and in [3] to elaborate on FPGA placement issues and standard techniques for addressing them.

While the basic goal of simulated annealing is to locate a low-cost placement based on a pre-specified cost, a number of different parameters can be used to control the algorithm's operation.

Ex 1: Simulated Annealing Parameters
In several short paragraphs, summarize Section 3 of [1] especially focussing on the discussion of annealing parameters. Specifically, how is the start temperature for simulated

annealing determined? How is the temperature update scheme implemented for VPR? Note that these parameters are also discussed in [3].

Ex 2: Inner Number Variation
Based on Figure 4-3 of [3], how does β effect wire lengths? Approximately how does run time change as β is varied?

Ex 3: Start Temperature Variation
Based on Figure 4-3 of [3], how does $StartT$ effect wire lengths? Approximately how does run time change as $StartT$ is varied?

Ex 4: Fixed Temperature Update Schedule
Based on Figure 4-3 of [3], how does *alpha_t* effect wire lengths? Approximately how does run time change as *alpha_t* is varied?

Ex 5: Bounding Box Scaling
In class, we discussed that net bounding box sizes are sometimes adjusted during placement and routing to a value which is larger than the Manhattan distance between the most-distant net points. Why is this adjustment performed and how are the adjustments determined? (Hint: note reference [2])

3 A* Routing

For the next set of exercises, you will have the opportunity to learn more about routing for FPGAs by varying a set of parameters that control router run time. In class, we learned that *maze routing* can be thought of as an A* search that evaluates possible routing paths from a net source to net destinations. Typically, this search is performed in such a way as to minimize a prespecified cost function at intermediate points in routing paths. In VPR, the per-net intermediate cost along a path is:

$$Cost = Cost_{prev} + C_0 + \alpha(\Delta D) \tag{1}$$

where $Cost_{prev}$ is the cost of previous track segments, C_0 is the cost of the current track under evaluation, and ΔD is the Manhattan distance from the current track to the destination [2]. α is a tunable parameter which indicates which part of the cost function should be weighted most heavily. A large α value indicates that closeness to the destination is the overriding factor in selecting a new track to add to an existing route while an α value near 0 indicates that track congestion is the more important factor. More complete descriptions of this cost function and maze routing in general can be found in Section 2.2.1 of [2].

Ex 6: Using the A* parameter, α
Summarize the results in Table 4 and Figure 4 in [2] in several paragraphs. Is the use of depth-first routing (e.g. using α to direct routing) effective?

Ex 7: Net Ordering

Most routing algorithms perform routing using a largest-first approach. Why does routing the longest and highest fanout nets first help routing? Keep in mind that VPR typically requires multiple routing iterations in which each net is ripped up and rerouted.

Ex 8: Power-aware Placement

In class, we discussed a cost function that allows for a balance between timing-driven and area-driven placement. How can the cost function of the VPR placer be modified to also consider power consumption? Please provide an example cost equation. Limit your answer to several paragraphs.

Ex 9: Timing-driven Routing

In class, we discussed a cost function that allows for a balance between timing-driven and area-driven placement. How can the cost function of the VPR router be modified to allow for this same type of balance? Please provide an example cost equation. Limit your answer to several paragraphs.

References

[1] V. Betz and J. Rose. VPR: A New Packing, Placement, and Routing Tool for FPGA Research. In *Proceedings, Field Programmable Logic, Seventh International Workshop*, Oxford, UK, Sept. 1997.

[2] J. Swartz, V. Betz, and J. Rose. A Fast Routability-Driven Router for FPGAs. In *6th International Workshop on Field-Programmable Gate Arrays*, Monterey, Ca, Feb. 1998.

[3] R. Tessier. *Fast Place and Route Approaches for FPGAs.* PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 1999. Chapter 4.