A Place-and-Route Tool for Heterogeneous FPGAs

Laura Beck (lwb4@cornell.edu)

Contents:

Project Description:

I have been modifying an FPGA placer/router tool called Versatile Placer/Router (VPR) [1] to handle heterogeneous as well as homogeneous FPGA architectures. We want to experiment with heterogeneous FPGAs with embedded hard macros such as processors or multipliers to determine what FPGA routing channel structure allows for the greatest routability around these hard macros.

Introduction:

A field programmable gate array (FPGA) is a programmable device that can implement many different digital circuit designs. FPGAs contain four types of elements called: logic blocks,

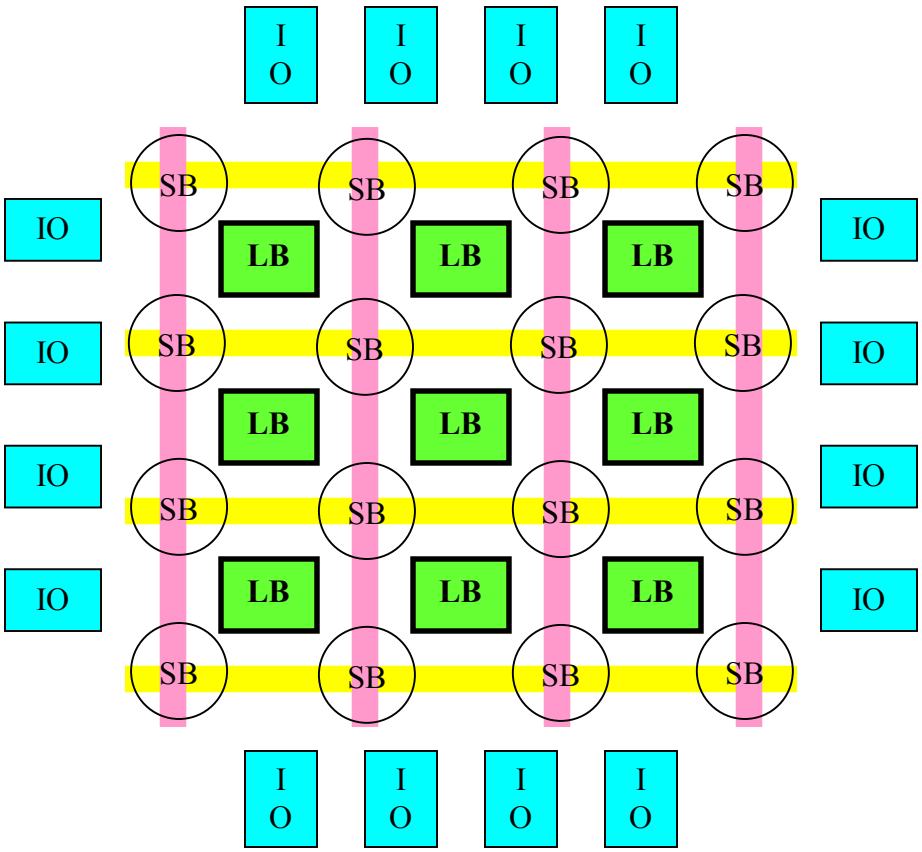

Figure 1. FPGA Structure

routing channels, switch boxes, and IO pins, arranged as shown in Figure 1. The body of the FPGA is a grid of logic blocks, colored green in Figure 1. The grid in Figure 1 is small for

clarity- the grids in most actual FPGAs contain many more logic blocks. The spaces between the logic blocks are called routing channels, marked yellow for horizontal channels and pink for vertical channels in Figure 1. Switch boxes appear wherever vertical and horizontal channels intersect, and as with other chips, the FPGA is bordered by IO pins (shown in blue) allowing for communication with the external world.

The structure and complexity of an FPGA's logic blocks varies from device to device. However, in look-up-table (LUT) based FPGAs, the type we are studying, logic blocks always contain at least one LUT - which can be programmed to implement any Boolean function of some small number of variables - and one flip-flop [2]. Thus, each logic block can be programmed to either perform a small amount of logic, store a small amount of data, or sometimes both. The FPGA's vertical and horizontal routing channels are made up of banks of wires, through which signals can travel between the logic blocks and between logic blocks and IO pins. Many vertical and horizontal wires go into each of the FPGA's switch boxes, and these switch boxes are programmed to form only some of the many possible connections between these wires, causing the signals in the FPGA to be routed to the correct places.

When working with an FPGA, one designs a large digital circuit, and then a software synthesis tool breaks it down into pieces small enough to be implemented by single logic blocks. Then a second computer-aided design (CAD) tool, called a placer/router, is needed to determine exactly where on the FPGA each part of the design should be located. As the name implies, a placer/router's work has two stages: placement and routing. Placement determines exactly which FPGA logic blocks should implement which logic-block sized pieces of the circuit design, and routing decides how the FPGA's switch boxes should be configured so that the circuit's signals will travel to the appropriate places through the FPGA's channels. The placer tries to arrange the logic blocks in a way that will minimize the distance signals in the circuit will have to travel. The router tries to minimize the amount of wiring needed and the speed the final circuit will run at. Routing is generally much more time-consuming than placement. My research involves modifying the C-code for a placer/router tool called Versatile Place and Route (VPR) that was developed at the University of Toronto [1].

Most FPGAs look somewhat like Figure 1 in that all of their logic blocks have the same structure, all of their routing channels are the same size, and basically their entire structure is uniform. Such FPGAs are called homogeneous FPGAs. We are interested in studying heterogeneous FPGAs – those in which some parts of the FPGA are different. Particularly, we want to study FPGAs with embedded hard macros: non-reconfigurable hardware such as processors and multipliers inserted somewhere in the FPGA [3]. An example of such a device is the Xilinx Virtex II Pro FPGA, equipped with embedded memory blocks, a multiplier, and two processors. Our goal is to determine whether non-uniform channel sizes would be useful in FPGAs containing hard macros. We plan to investigate this by using VPR to place and route circuits from the twenty circuit MCNC Benchmark Suite on FPGA architectures containing hard macros with several different channel structures. The VPR program was not designed to work with this type of heterogeneous FPGA, so I am currently modifying VPR to handle these architectures.

A related study was conducted in 2001 by P. Hallschmid and S. Wilton [4]. Their study differs from ours in that it focused on embedded memory blocks whereas we are considering general hard macros, and their study used detailed routing, while we are using global routing. The difference between detailed and global routing concerns switch boxes. The simplest (and most expensive) type of switch box one could build would be capable of making a connection between any two of the wires entering it. This requires more hardware than is found in actual switch boxes – which usually are only able to form some of these connections. In detailed routing, one gives specific information about the switch boxes of the FPGA being used to produce a route that could be downloaded to an actual FPGA. In global routing, one assumes ideal switch boxes that can form any possible connection. Global routing is good for experimenting with hypothetical FPGA architectures that do not yet have any specific switch box structure. At this stage, we are not concerned with switch behavior, so global routing is appropriate for our work.

Contributions:

1) The first modification I made to VPR was to allow a user to specify any channel in the FPGA and determine what size that channel would be, either in absolute units, or relative to the size of a typical channel. The relative size is useful because the size of a typical channel is not usually known before routing takes place. VPR performs a binary search to determine the smallest size for typical channels that will not make routing impossible. Figure 2 shows VPR's display of an FPGA with some channels wider than others. The gray squares represent logic blocks and the subdivided gray squares along the border represent IO pins.

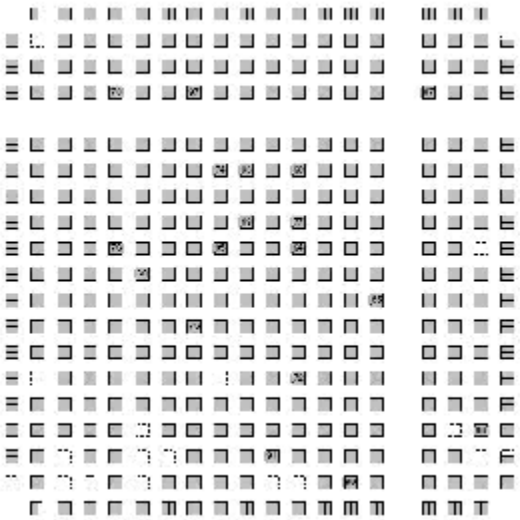

Figure 2. VPR Display of an FPGA with Two Very Wide Channels

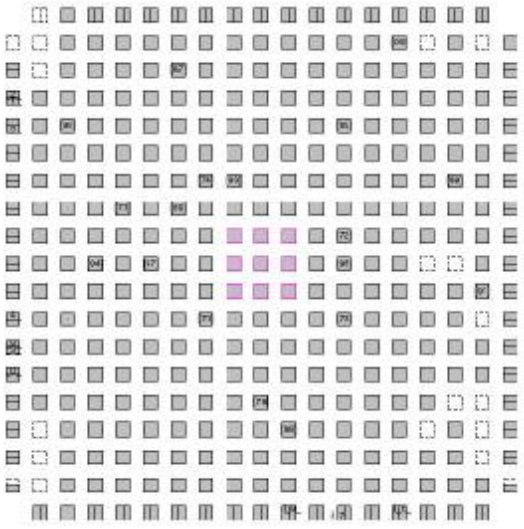

Figure 3. VPR Display of an FPGA with A Hard Macro at its Center

2) Next I integrated some code written by a former student working with my mentor into my modified version of VPR. We wanted to represent a hard macro as a rectangular group of logic blocks that was assigned a specific location by a user and never moved from that spot by VPR. In normal operation, VPR assumes all logic blocks are identical FPGA resources and its placing algorithm moves all of the logic blocks around to find an optimal arrangement. The former

student's code allowed a user to specify logic blocks that would not be moved by VPR's placer. I integrated this portion of his code into my work rather than adding my improvements to a copy of his code from the start because other parts of his code do not work properly.

Figure 3 shows VPR's representation of an FPGA containing a hard macro at its center. Logic blocks within the hard macro are colored pink instead of gray. During placement, one can observe the gray blocks moving around and the pink blocks remaining fixed in place.

3) I then changed VPR so that when a hard macro is present, VPR will automatically find the channels next to the hard macro and make these channels twice as large as a typical channel. The decision to make them twice as large is arbitrary and easily changed. This modification just makes it easy to change all of these channels as a group. Figure 4 shows VPR's display of an FPGA with a hard macro surrounded by wide channels.

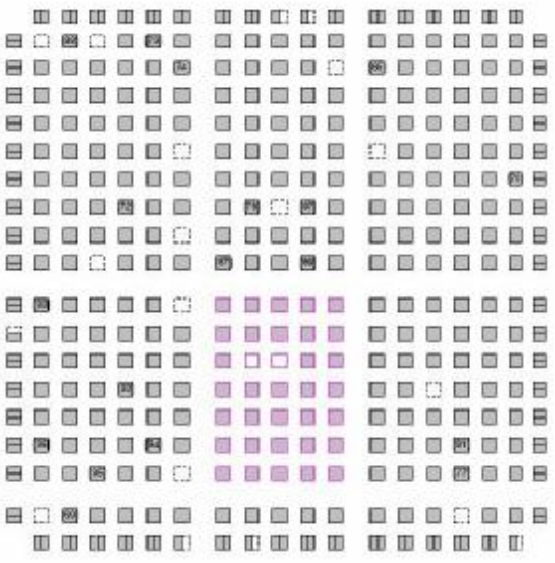

Figure 4. VPR Display of an FPGA with a Hard Macro Surrounded by Wide Channels

Tasks in Progress:

1) I am currently working on getting VPR's router not to route any FPGA signals on the channels in the middle of a hard macro. This modification is necessary because an FPGA containing an actual embedded chip would not be able to route anything through this chip. Figure 5 shows a zoomed-in VPR display of a circuit with that has been routed on an FPGA with a hard macro. The paths through the FPGA's channels that are being used by some signal are drawn in, those connecting to the hard macro's logic blocks in pink,

and others in black. The problem is that currently VPR routes some of the black wires (thickened for clarity in Figure 5) between the hard macro's pink logic blocks.

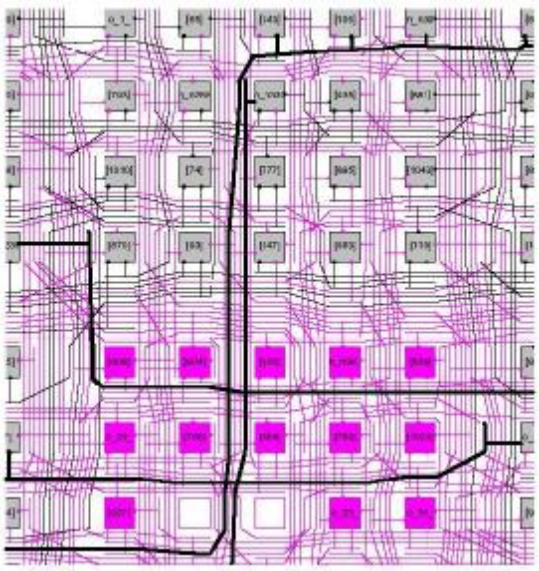

Figure 5. VPR's Current Routing Sends Wires Through the Hard Macro – This Needs to Change

2) Presently, only one hard macro can be specified and recognized by VPR, and its position is given in a header file so that every time the hard macro's size or position is changed, VPR must be entirely recompiled. I want to edit this so that hard macros can be specified in an input file and multiple hard macros can be present within a single FPGA.

3) As shown in Figure 2, VPR's graphical interface displays a diagram of the FPGA being used, and this display has been modified to color logic blocks within a hard macro pink. The pink coloring does not always show up when it should, and I would like to fix this if I have time.

Figure 6, below, shows my entire plan for modifying VPR:

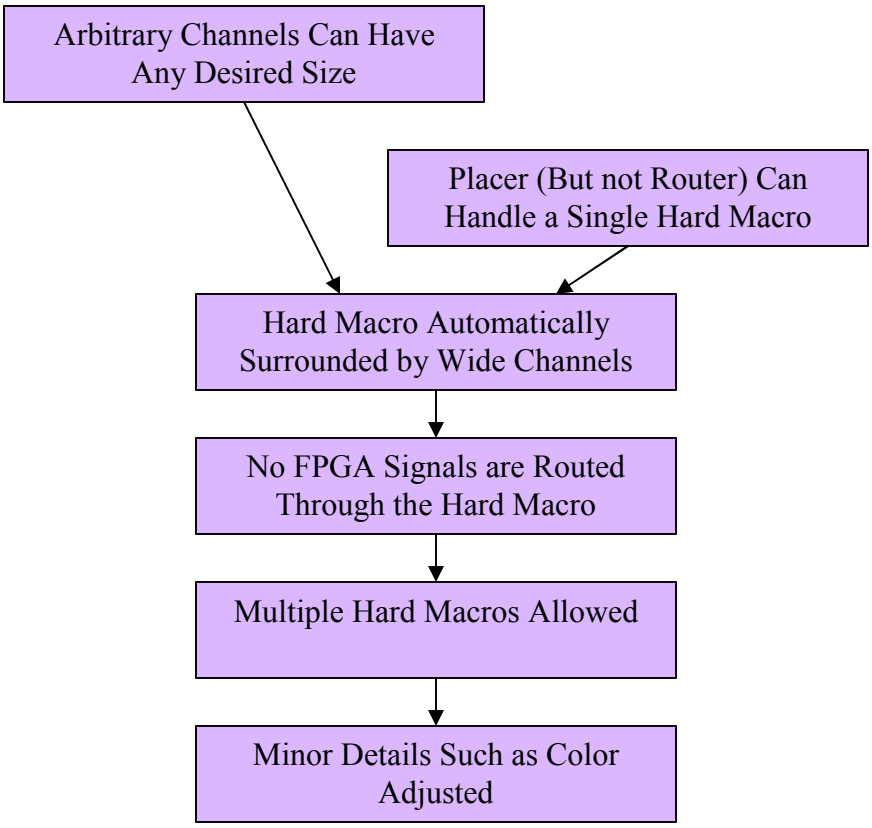

Figure 6. VPR Modification Plan

Experimental Plan:

When the modified version of VPR is usable, I will run VPR to place and route a group of sample circuits on FPGAs with embedded hard macros, experimenting with the size of the channels around these hard macros. A study conducted by the makers of VPR showed that for homogeneous FPGAs it is optimal for all channels to be the same size [5]. My mentor and I predict, however, that for heterogeneous FPGAs containing hard macros, the sample circuits will be routable with a smaller typical channel size if the channels near the hard macro are larger than typical channels. If this is true, I will try to determine how much larger than the typical channels it is favorable for channels around hard macros to be.

Mentorship Experience:

I have been working for my DMP mentorship for six weeks, and I have four weeks of work remaining. My first three weeks were spent discussing possible projects with my mentor, Eli Bozorgzadeh, reading background information and documentation for tools I would need, installing software, getting computers set up and designing my DMP website. The next three weeks were devoted to concentrated work on my research project itself, and preparing this report.

The most exciting thing that has happened was the first accomplishment listed in the contributions section. This was the first improvement I made to the piece of software I'm working with and it took me a week to make it since I was not yet familiar with the code and I tried several things that did not work before finding a good solution. It was very satisfying when I got this working. Likewise, I would say working on this task has been my biggest challenge so far.

I work in a room where about eight CE/CS graduate students have their cubicles. My mentor wanted her DMP students to be near the graduate students so we could get a feel for their working environment and experience and ask them questions. I think it was a good idea – I've gotten some good advice about careers and graduate study from some of those students. I meet with my mentor for a half hour every day to discuss how my project is going. I also update my DMP website every night and Professor Bozorgzadeh checks it regularly. There are no other students working on my project, however there are always other students around me.

My desk is next to that of another DMP student, Padmini Nagaraj, who is also working with my mentor. It's nice having someone my age nearby. There are essentially no undergraduates on campus who do not live in Irvine and have cars so they can spend most of their time off campus. I think being in Irvine for the summer would have been somewhat isolating if I had been the only DMP student here, but because there are two of us, it has been fine.

I live in a single room (I try to get single rooms since I'm a very light sleeper compared to most other college students) in a small dorm that's a twenty-minute bike ride from the building where

I work. This is the only college campus I've been on where the buildings aren't walking distance from each other. I was very surprised at this (and a little concerned) when I first got here, but I was able to get an inexpensive bike at Target, and it's actually been a lot of fun biking around campus. My dorm is in a really beautiful location, and I haven't had any problems with the people there – as I mentioned before, most of them aren't around very much, although they seem nice.

References:

[1] V. Betz and J. Rose, "VPR: A New Packing, Placement and Routing Tool for FPGA Research," *International Workshop on Field Programmable Logic and Applications*, 1997.

[2] J. Wakerly, *Digital Design Principles and Practices*, Prentice Hall, 2001.

[3] W. Wolf, *FPGA-Based System Design*, Prentice Hall, 2004.

[4] P. Hallschmid and S. Wilton, "Detailed Routing Architectures for Embedded Programmable Logic IP Cores," *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey CA*, Feb. 2001, pp. 69-74.

[5] V. Betz and J. Rose, "Effect of the Prefabricated Routing Track Distribution on FPGA Area-Efficiency," *IEEE Transactions on VLSI, Vol. 6, No. 3*, Sept. 1998, pp. 445-456.